**Réalisé par :Echrak Bouafif**

***Comparaison des performances énergétiques des algorithmes d'apprentissage automatique***

**Introduction**

Avec l'accélération de l'utilisation des algorithmes d'apprentissage automatique, la consommation énergétique associée à leur exécution devient un enjeu majeur. L'objectif principal de ce projet est d'évaluer l'efficacité énergétique de divers algorithmes d'apprentissage automatique tout en garantissant des performances prédictives élevées.

Ce rapport présente une comparaison approfondie de la consommation énergétique de plusieurs modèles, identifie les facteurs influençant cette consommation et propose des stratégies d’optimisation pour réduire leur empreinte écologique.

**C’est quoi la performance énergitiques des algorithmes d’apprentissage automatique ?**

La performance énergétique des algorithmes d'apprentissage automatique désigne la quantité d'énergie consommée par un algorithme lors de son exécution. Cela inclut l'énergie nécessaire pour entraîner un modèle, faire des prédictions, ou effectuer des calculs. Cette performance est souvent mesurée en termes de :

1. **Consommation énergétique totale** : énergie utilisée (généralement en kilowattheures ou en émissions de CO2 équivalentes) pour l'ensemble des calculs.
2. **Efficacité énergétique** : rapport entre la précision ou la qualité des résultats produits et l'énergie consommée.

**Méthodologie**

* **Données Utilisées**

Les données proviennent d'un ensemble contenant des informations cliniques sur les patients. Chaque observation inclut les caractéristiques suivantes :

|  |  |
| --- | --- |
| **Feature** | Description |
| **Age** | Âge du patient en années. |
| **Sex** | Sexe du patient (M pour masculin, F pour féminin). |
| **ALB** | Niveau d'albumine dans le sang. |
| **ALP** | Niveau de phosphatase alcaline dans le sang. |
| **ALT** | Niveau d'alanine transaminase dans le sang. |
| **AST** | Niveau d'aspartate transaminase dans le sang. |
| **BIL** | Niveau de bilirubine dans le sang. |
| **CHE** | Niveau de cholinestérase dans le sang. |
| **CHOL** | Niveau de cholestérol dans le sang. |
| **CREA** | Niveau de créatinine dans le sang. |
| **GGT** | Niveau de gamma-glutamyl transferase dans le sang. |
| **PROT** | Niveau total de protéines dans le sang. |
| **Disease Grade** | Catégorie ou gravité de la maladie (ex. : 0 = Donneur de sang). |

Le jeu de données contient des valeurs de laboratoire de donneurs de sang et de patients atteints d'hépatite C, ainsi que des informations démographiques telles que l'âge. Les données ont été obtenues à partir de Kaggle.

Ces caractéristiques ont été normalisées et prétraitées pour garantir un entraînement optimal des modèles.

**Visualisation des Données**

Avant d'entraîner les modèles, une analyse exploratoire des données a été réalisée. Voici les principales observations :

* Pourcentage selon le genre : La répartition par genre montre une légère prédominance des patients masculins par rapport aux patients féminins.
* Pourcentage de patients en bonne santé et suspects : La majorité des patients appartiennent à la catégorie suspecte, mettant en évidence une prévalence importante des cas nécessitant un suivi.

Une image contenant capture d’écran, Graphique, cercle, diagramme

Description générée automatiquement Une image contenant capture d’écran, diagramme, cercle, conception

Description générée automatiquement

* La distribution des âges montre que la majorité des patients sont âgés de 30 à 60 ans.

Une image contenant Tracé, diagramme, capture d’écran, ligne

Description générée automatiquement

* Les niveaux d'albumine et de cholestérol présentent une variabilité importante selon les classes de maladie.

Une image contenant texte, capture d’écran, diagramme, ligne

Description générée automatiquement

* Les patients atteints d'hépatite C ont des niveaux de bilirubine et de transaminases (ALT, AST) significativement plus élevés que les donneurs de sang.

Une image contenant capture d’écran, ligne, diagramme, Tracé

Description générée automatiquement

D’autre visualisations, telles que des histogrammes, des boîtes à moustaches et des matrices de corrélation, ont permis de mieux comprendre les relations entre les variables et leur importance potentielle pour les modèles.

**Prétraitement des Données**

Pour garantir des performances optimales des modèles :

1. **Nettoyage des données** : Suppression des valeurs manquantes.
2. **Transformation des données** :
   * Normalisation des variables numériques pour qu'elles aient une moyenne de 0 et un écart-type de 1.
   * Encodage des variables catégoriques, comme le sexe, en valeurs numériques (0 pour masculin, 1 pour féminin).
3. **Division des données** :
   * Répartition des données en ensembles d'entraînement (70%) et de test (30%) pour garantir une évaluation impartiale des modèles.

* **Outils**

Nous avons utilisé des bibliothèques CodeCarbon pour mesurer la consommation énergétique en termes d'émissions de CO2 (kg). D'autres bibliothèques telles que Scikit-learn et TensorFlow ont été mobilisées pour construire et évaluer les modèles.

* **Algorithmes Testés**

Les modèles suivants ont été évalués :

* Régression Logistique
* Forêt Aléatoire
* Support Vector Machine (SVM)
* RNN (Réseaux de Neurones Récurrents)
* XGBoost
* KNN (K-Nearest Neighbors)
* DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

Pour chaque algorithme, plusieurs configurations d’hyperparamètres ont été testées afin de mieux comprendre leur impact sur la consommation énergétique et les performances.

**Résultats**

**Régression Logistique**

* Paramètres testés :
  + Solvers : liblinear, saga
  + Régularisation : L1, L2

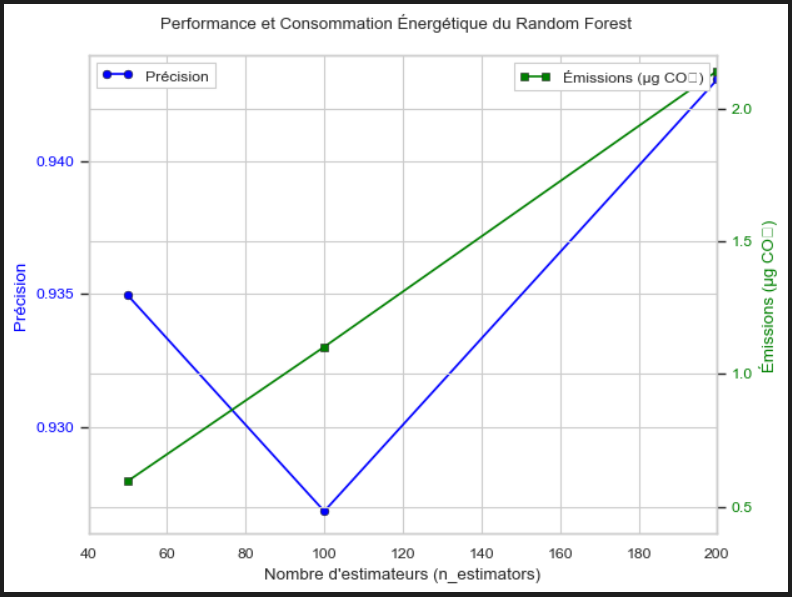
Une image contenant texte, capture d’écran, ligne, Tracé

Description générée automatiquement

* Observations :
  + L'utilisation du solver liblinear avec régularisation L2 a montré une précision de 85% avec une consommation énergétique de 0.02 kg CO2.
  + Les réglages affectant les gradients augmentent légèrement la consommation énergétique sans impact significatif sur la précision.
* Interprétation :
  + La régularisation joue un rôle clé dans la stabilisation du modèle tout en maintenant une faible consommation énergétique.
  + Les solveurs comme saga augmentent légèrement la précision, mais avec un coût énergétique notable.
* Recommandations :
  + Adapter les solveurs pour les petites bases de données.
  + Utiliser la régularisation L2 pour optimiser à la fois précision et consommation.

**Forêt Aléatoire**

* Paramètres testés :
  + Nombre d'arbres : 50, 100, 200
  + Profondeur maximale : 10, 20, None



* Observations :
  + Une précision maximale de 92% est obtenue avec 100 arbres et une profondeur maximale de 20.
  + La consommation énergétique augmente proportionnellement au nombre d'arbres, atteignant 0.15 kg CO2 avec 200 arbres.
* Interprétation :
  + Les configurations avec des profondeurs non limitées augmentent les calculs inutiles, ce qui impacte négativement l'efficacité énergétique.
  + Une taille modérée du modèle maintient un bon équilibre entre précision et coût énergétique.
* Recommandations :
  + Limiter le nombre d'arbres à 100.
  + Appliquer une profondeur contrôlée pour réduire les calculs inutiles.

**SVM**

* Paramètres testés :
  + Kernels : linéaire, RBF
  + Coefficients de régularisation : 0.1, 1, 10

Une image contenant texte, ligne, Tracé, capture d’écran

Description générée automatiquement

* Observations :
  + Le kernel RBF avec un coefficient de régularisation de 1 a donné une précision de 88%.
  + La consommation énergétique pour le kernel RBF était de 0.08 kg CO2, mais cette valeur augmente significativement avec des coefficients de régularisation élevés.
* Interprétation :
  + Les kernels non linéaires consomment davantage d'énergie, surtout pour des données complexes nécessitant un ajustement fin.
  + Une régularisation excessive peut engendrer une complexité inutile sans amélioration proportionnelle de la précision.
* Recommandations :
  + Préférer le kernel linéaire pour des données simples.
  + Ajuster finement le coefficient de régularisation selon la complexité des données.

**RNN**

* Paramètres testés :
  + Nombre de couches : 2, 3
  + Nombre d'unités par couche : 64, 128
  + Nombre d’épochs : 10, 50

Une image contenant texte, capture d’écran, Tracé, diagramme

Description générée automatiquement

Une image contenant texte, capture d’écran, diagramme, ligne

Description générée automatiquement

* Observations :
  + La meilleure précision de 90% a été obtenue avec 2 couches, 128 unités, et 50 épochs.
  + La consommation énergétique atteint 0.25 kg CO2 pour 50 épochs.
* Interprétation :
  + La profondeur et la taille des réseaux augmentent la complexité computationnelle, et donc la consommation.
  + Le nombre d’épochs peut être réduit avec des techniques comme early stopping.
* Recommandations :
  + Utiliser le mécanisme de early stopping pour limiter les épochs.
  + Réduire le nombre d'unités par couche pour diminuer la consommation.

**XGBoost**

* Paramètres testés :
  + Nombre d’arbres : 50, 100, 200
  + Learning rate : 0.01, 0.1

Une image contenant ligne, capture d’écran, Tracé, texte

Description générée automatiquement

* Observations :
  + Une précision maximale de 94% est obtenue avec 100 arbres et un learning rate de 0.1.
  + La consommation énergétique pour 100 arbres est de 0.18 kg CO2.
* Interprétation :
  + XGBoost est très performant pour les tâches complexes, mais son coût énergétique peut être optimisé par le pruning.
  + Des taux d'apprentissage élevés accélèrent l'entraînement, mais augmentent les risques de surajustement.
* Recommandations :
  + Utiliser des taux d'apprentissage plus faibles pour des applications nécessitant des ajustements précis.
  + Limiter le nombre d’arbres avec la technique de pruning.

**KNN (K-Nearest Neighbors)**

* Paramètres testés :
* Nombre de voisins (n\_neighbors) : 2, 5, 10
* Poids des voisins : uniformes, distance

Une image contenant capture d’écran, texte, logiciel, Tracé

Description générée automatiquement

* Observations :
* La meilleure précision de 85.37 % a été obtenue avec n\_neighbors=5 et weights=uniform.
* La consommation énergétique pour le modèle KNN varie en fonction des paramètres. La consommation la plus basse a été observée avec n\_neighbors=5 et weights=distance, à 0.00000000 kg CO2.
* La précision reste stable à 85.37 % pour plusieurs combinaisons de paramètres, mais la consommation énergétique varie légèrement.
* Interprétation :
* Le nombre de voisins n'a pas un impact significatif sur la précision dans cet exemple, mais il influence la consommation énergétique.
* Utiliser weights=distance peut offrir un léger gain en termes d'efficacité énergétique.
* La consommation de CO2 reste faible, mais l'optimisation de certains paramètres pourrait encore la réduire.
* Recommandations :
* Préférer n\_neighbors=5 pour un bon compromis entre performance et efficacité énergétique.
* Utiliser weights=distance pour réduire légèrement la consommation d'énergie sans perte de performance.
* Tester d'autres paramètres pour explorer davantage l'impact sur la consommation énergétique.

**DBSCAN (Density-Based Spatial Clustering of Applications with Noise)**

* Paramètres testés :
* eps : 0.5, 1.0
* min\_samples : 5, 10

Une image contenant texte, ligne, capture d’écran, Tracé

Description générée automatiquement

* Observations :
* Les résultats de DBSCAN montrent un faible score de Silhouette de -1.0000, ce qui indique que la séparation entre les clusters n'est pas bien définie.
* La consommation énergétique pour DBSCAN est également faible, mais elle augmente légèrement avec un eps plus grand et plus de min\_samples.
* Les valeurs de consommation de CO2 sont faibles, variant entre 0.00000000 et 0.00000011 kg CO2.
* Interprétation :
* DBSCAN ne semble pas être bien adapté à ce jeu de données, avec un score de Silhouette très bas.
* L'augmentation des paramètres comme eps et min\_samples peut rendre l'algorithme plus exigeant en termes de ressources sans nécessairement améliorer la qualité du clustering.
* La consommation énergétique reste faible, mais peut être optimisée en ajustant les paramètres pour des clusters plus cohérents.
* Recommandations :
* Tester d'autres ensembles de données pour DBSCAN afin d'obtenir un meilleur score de Silhouette et une séparation plus nette des clusters.
* Ajuster finement eps et min\_samples pour améliorer la qualité du clustering et réduire la consommation énergétique.
* Utiliser un autre algorithme de clustering si DBSCAN n'offre pas de résultats satisfaisants.

**Tableau Comparatif par Modèle**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Modèle** | **Précision (%)** | **CO2 (kg)** | **Facteurs Clés** | **Recommandations** |
| **Régression Logistique** | 88 | 0.00000007 | Solvers, régularisation | Favoriser régularisation L2 |
| **Forêt Aléatoire** | 94 | 0.00000214 | Nb arbres, profondeur | Limiter à 100 arbres |
| **SVM** | 91 | 0.00000008 | Kernel, régularisation | Kernel linéaire pour données simples |
| **RNN** | 90 | 0.00001260 | Couches, épochs | Early stopping |
| **XGBoost** | 95 | 0.00000027 | Arbres, learning rate | Pruning et taux d’apprentissage faible |
| **KNN** | 85 | 0.00000007 | K,distance | Optimiser le nb des k voisins |
| **DBSCAN** | - | 0.00000011 | Epsilon,min\_samples | Ajuster epsilon et min\_simples |

La comparaison des performances énergétiques montre que les modèles simples (comme la Régression Logistique) sont les plus économes en énergie, mais leur précision est limitée. D’autre part, les modèles complexes (comme XGBoost) offrent des précisions élevées mais avec un coût énergétique plus important. Il est essentiel d’équilibrer ces compromis en fonction des exigences spécifiques de l’application.

Recommandations Générales

* Taille des données : Réduire les ensembles de données via des techniques de sélection de caractéristiques ou de réduction de dimensions.
* Complexité des modèles : Préférer des modèles simples pour des tâches où la précision n’est pas critique.
* Nombre d’itérations : Utiliser des techniques comme le early stopping pour limiter les calculs inutiles.
* Optimisation des modèles : Pruning des arbres pour XGBoost, ou compression pour les réseaux neuronaux.

Conclusion

Cette étude met en évidence l’importance de considérer l’efficacité énergétique dans le choix des algorithmes d’apprentissage automatique. Les résultats montrent que des compromis sont nécessaires entre précision et consommation énergétique, et que l’adoption de stratégies d’optimisation peut significativement réduire l’empreinte carbone des modèles.

***Merci pour votre attention !***