Techniques d'apprentissage - Projet

Demangeon Matéo (demm1412) Guitteny Martin (guim1106) Rioux Lucas (riol2003)

11 Décembre 2023

Le dépôt de notre projet est disponible à l'adresse suivante : https://github.com/MartinGuitteny/ta-project/tree/main.

1 Introduction

Pour le projet de techniques d'apprentissage IFT712, nous devions implémenter 6 méthodes de classification afin de classifier différentes espèces d'arbres. Le jeu de données est disponible ici, sur Kaggle : https://www.kaggle.com/c/leaf-classification.

2 Démarche

Ce jeu de données est composé d'images de feuilles étiquetées, avec leurs caractéristiques enregistrées dans un fichier csv. Nous nous sommes uniquement servis du fichier `train.csv` qui contient des données étiquetées, contrairement à `test.csv`.

Notre objectif est de créer un classifieur qui utilise ces caractéristiques pour retrouver les labels des feuilles. Nous nous sommes servis des modèles de la librairie `scikit-learn` [1]. Nous avons choisi de comparer les méthodes de classification suivantes :

- Modèle génératif
- Perceptron simple (sans couche cachée)
- Régression logistique
- Méthode des K plus proches voisins (K-nearest neighbors, ou KNN)
- Machine à vecteurs de support (SVC)
- Classificateur à renforcement de gradient (Gradient boosted trees)
- Réseau de neurones

On quantifie la performance de chaque modèle via 3 scores : **précision**, **rappel** et **F1-Score**. Pour chacun de ces modèles, on va rechercher les meilleurs hyperparamètres à l'aide de la fonction `GridSearchCV`, grâce à la validation croisée. Nous avons choisi de diviser le jeu de données d'entraînement en 10 folds.

Avant d'utiliser chaque méthode, nous avons **normalisé les données** pour améliorer l'efficacité des modèles. La normalisation choisie est le Z-Score (classe `StandardScaler` de `scikit-learn`). Nous avons comparé les résultats avec et sans normalisation.

De plus, toutes les classes ne comportaient que 10 individus. Ce faible nombre d'individus a posé des problèmes lors de l'utilisation de `train_test_split` : le nombre d'individus de chaque classe était très variable dans les deux jeux de données, allant de 0 à 10. Nous avons implémenté une méthode permettant de pallier ce problème (`balanced_train_test_split`), et de créer un ensemble d'entraînement avec 8 individus de chaque classe, et un ensemble de test avec 2 individus de chaque classe. Cette technique sera appelée **équilibrage des classes** dans la suite de ce rapport. Suite à cela, nous avons réduit le nombre de folds à 5, sinon notre programme retournait une erreur (on pouvait choisir d'avoir 8 folds au maximum, soit autant que le nombre de données comprises dans chaque classe).

3 Résultats

Les différents scores obtenus par les différents modèles utilisés sont inscrits dans le tableau 1.

Résultats				
Modèle	Précision	Rappel	F1-Score	Temps
Modèle génératif	0.015	0.014	0.004	< 1s
Perceptron	0.903	0.888	0.896	< 1s
Régression logistique	0.998	0.986	0.988	< 1s
Méthode des K plus proches voisins	0.974	0.968	0.971	< 1s
Machine à vecteurs de support	0.987	0.983	0.985	78s
Classif. à renforcement de gradient	0.516	0.452	0.482	< 1s
Réseau de neurones	0.981	0.975	0.978	< 1s

Table 1: Scores obtenus pour chaque modèle, avec équilibrage des jeux de données et avec normalisation

Chaque score est la moyenne d'une centaine de tentatives avec une recherche d'hyperparamètres. Les temps d'exécution sont la moyenne d'une centaine de tentatives, sauf pour la classification à renforcement de gradient, où nous n'avons effectué que 10 essais. Pour les temps d'exécution, nous avons désactivé la recherche d'hyperparamètres.

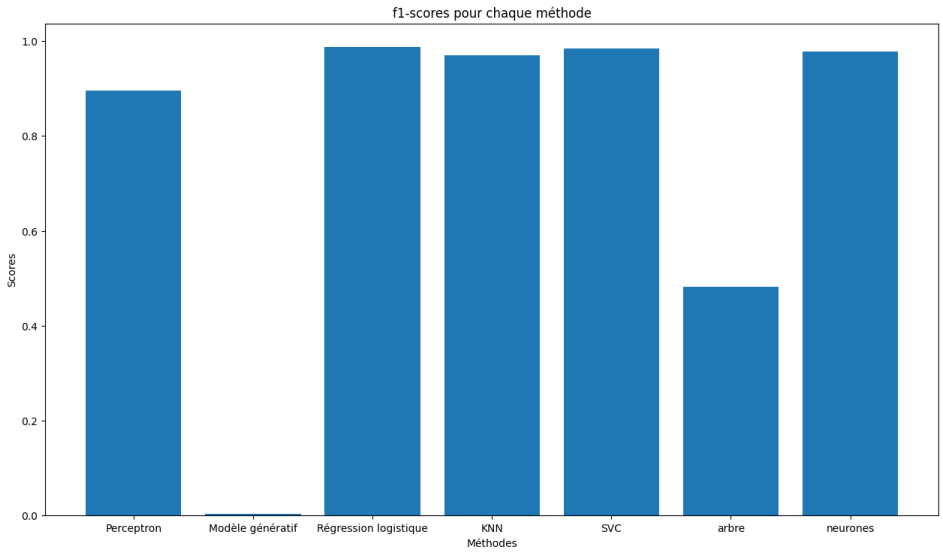

Figure 1: Diagramme des F1-scores de toutes les méthodes

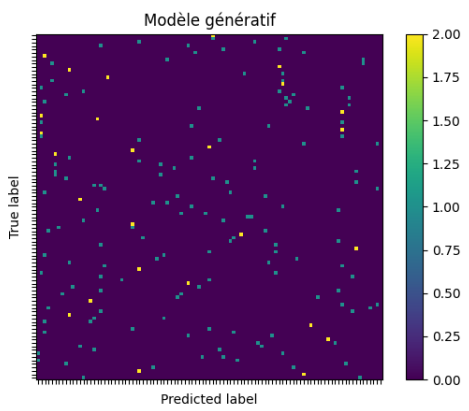

Figure 2: Matrice de confusion du modèle génératif

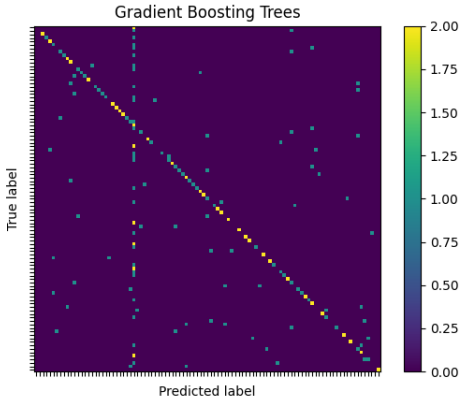

Figure 3: Matrice de confusion du modèle de classification à renforcement de gradient

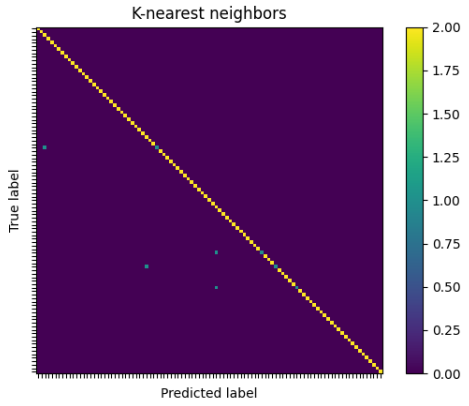

Figure 4: Matrice de confusion du modèle KNN

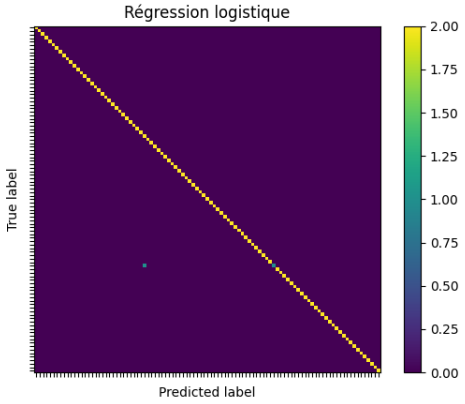

Figure 5: Matrice de confusion de la régression logistique

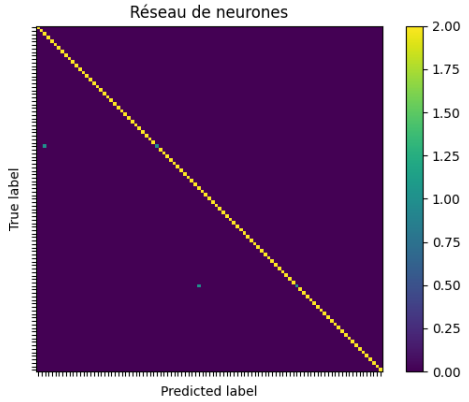

Figure 6: Matrice de confusion du réseau de neurones

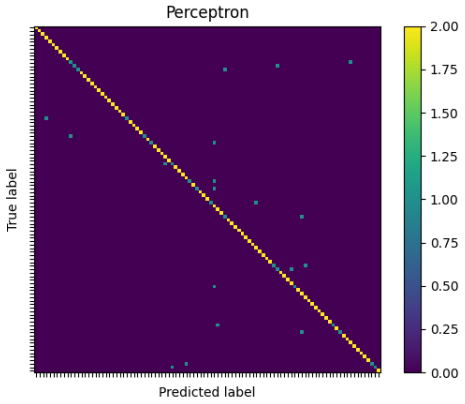

Figure 7: Matrice de confusion du perceptron

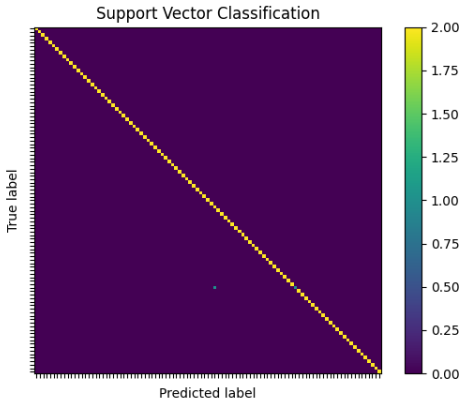

Figure 8: Matrice de confusion du SVC

On peut voir que mis à part le modèle génératif et le modèle de classification à renforcement de gradient, les modèles sont très performants, comme le montre le diagramme 1. En ce qui concerne le modèle génératif, les prédictions semblent proches de l'aléatoire. Cela est peut-être dû au caractère non-gaussien des données. Pour la classification à renforcement de gradient, les prédictions semblent correctes, à l'exception d'une classe qui est souvent prédite alors qu'elle ne devrait pas l'être. Enfin, la machine à vecteurs de support donne de bons résultats, mais au prix d'un temps d'exécution élevé.

Jusqu'ici, on a normalisé les données et on a partitionné le jeu de données de façon à obtenir le nombre de classes adéquat dans l'ensemble d'entraînement (et donc dans l'ensemble de test). On s'interroge sur l'utilité de ces deux opérations : on va donc les désactiver pour identifier leur impact.

Les tables 2, 3 et 4 présentent respectivement la précision, le rappel et le F1-Score des principaux modèles en activant ou non l'équilibrage et la normalisation. Ces scores sont les scores moyens obtenus sur 10 itérations avec recherche d'hyperparamètres. Nous n'avons pas calculé ces nouvelles métriques pour le modèle génératif et le classificateur à renforcement de gradient, que l'on juge trop mauvais.

Précision			
Modèle	Sans normalisation ni équilibrage	Équilibrage seul	Normalisation et équilibrage
Perceptron	0.665	0.708	0.903
Régression logistique	0.928	0.958	0.998
Méthode des K plus proches voisins	0.814	0.891	0.974
Machine à vecteurs de support	0.893	0.939	0.987
Réseau de neurones	0.884	0.934	0.981

Table 2: Précision obtenue pour chaque modèle et chaque situation

Rappel			
Modèle	Sans normalisation ni équilibrage	Équilibrage seul	Normalisation et équilibrage
Perceptron	0.693	0.685	0.888
Régression logistique	0.932	0.949	0.986
Méthode des K plus proches voisins	0.820	0.872	0.968
Machine à vecteurs de support	0.897	0.926	0.983
Réseau de neurones	0.890	0.916	0.975

Table 3: Rappel obtenu pour chaque modèle et chaque situation

F1-Score			
Modèle	Sans normalisation ni équilibrage	Équilibrage seul	Normalisation et équilibrage
Perceptron	0.677	0.696	0.896
Régression logistique	0.930	0.954	0.988
Méthode des K plus proches voisins	0.817	0.881	0.971
Machine à vecteurs de support	0.895	0.933	0.985
Réseau de neurones	0.887	0.925	0.978

Table 4: F1-Score obtenu pour chaque modèle et chaque situation

On constate que l'équilibrage et la normalisation des données augmentent de façon significative tous les scores des principaux modèles. On démontre ainsi leur utilité.

4 Conclusion

Parmi les modèles mis en œuvre, le perceptron, la régression logistique, la méthode des K plus proches voisins et le réseau de neurones semblent être les plus convaincants. En effet, les métriques de ces modèles dépassent systématiquement les 95%. La machine à vecteurs de support donne des résultats satisfaisants, mais au prix d'un temps de calcul élevé. Enfin, la classification à renforcement de gradient et le modèle génératif donnent des résultats médiocres. On a également constaté l'importance de normaliser les données et de bien créer nos ensembles d'entraînement et de test.

Bibliographie

[1] scikit-learn. `https://scikit-learn.org`.