DENIS Sarah 12 décembre 2022

dens1704

**Projet de Session IFT712**

**Leaf Classification**

**Introduction**

Ce projet s’inscrit dans le cadre du cours IFT712 - Techniques d’apprentissage. Il consiste à tester six méthodes de classification sur une base de données Kaggle avec la bibliothèque sickit-learn. Nous verrons dans ce rapport quelles ont été les données utilisées pour le projet, quelles méthodes ont été codées pour leur classification, quelle a été la démarche adoptée durant le projet et nous analyserons les résultats obtenus.

Tout d’abord, voici le lien git sur lequel le projet est disponible : <https://github.com/Sari27/IFT712_Projet_de_session>.

**Les données**

Présentation

Les données prises pour ce projet sont disponibles au lien suivant : <https://www.kaggle.com/c/leaf-classification>. Il s’agit de celles suggérées pour la réalisation de ce projet. C’est une base de données de plus de 1500 images en noir et blanc de feuilles d’arbres. On retrouve parmi ces images 99 espèces différentes. Elles sont représentées par des vecteurs numériques contenant 192 attributs. La base de données est séparée en deux sets : un set d’entraînement auquel le nom de l’espèce représentée est associée à chaque vecteur, et un set de tests ou les vecteurs n’ont pas de nom d’espèce associé.

Normalisation

Au commencement du projet, j’ai choisi de travailler sur les données brutes. Ainsi, les premières version disponible sur mon git montrent les résultats d’exécutions sur les données non transformées. Une fois les méthodes codées et la recherche de meilleurs hyperparamètres bien avancées, j’ai décidé de normaliser les données afin de voir si cela peut améliorer les résultats. Ainsi j’ai appliqué la normalisation suivante sur les données :

Avec X étant une colonne du dataframe, ce qui représente un attribut. J’ai appliqué cette normalisation sur le dataframe d’entraînement et le dataframe de test. L’application de cette formule sur les données a mené à l’obtention de résultats différents, nous verrons cela dans la partie analyse.

Nous pouvons noter qu’il existe d’autres formules permettant de normaliser les données, et d’autres méthodes de pré-processing applicables à des données de ce type, telles que la sélection d’attributs.

**Méthodes de classification utilisées et hyperparamètres recherchés**

Le projet de session demandait d’implémenter au minimum six méthodes de classification différentes. J’ai essayé au maximum de prendre des méthodes qui ne se ressemblaient pas : Expliquer pourquoi (type arbre, type ensemble)

Perceptron

Réseaux de neurones

K-voisins

Arbre de décision

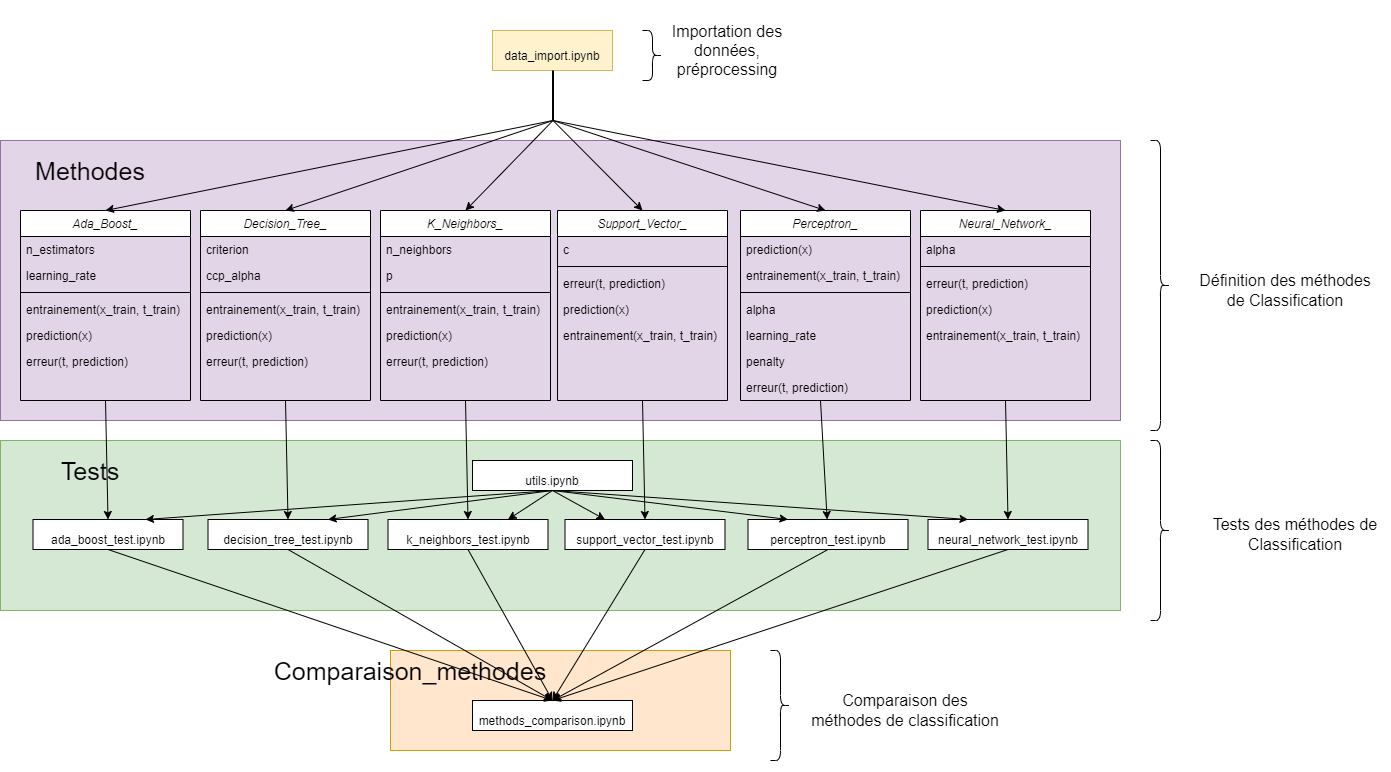
Vecteurs de support

AdaBoost

**Démarche scientifique**

Structure du projet

Diagramme de classes + explications de chaque package



Organisation du travail

Trello ou exel avec dates de rendus

**Analyse des résultats**

Ne pas oublier les différences avec les données prétraitées

Résultats par méthode

* Perceptron
* Réseaux de neurones
* K-voisins
* Arbre de décision
* Vecteurs de support
* AdaBoost

Données normalisées VS non brutes

| Erreur min | Perceptron | Réseaux de neurones | K-voisins | Arbre de décision | Vecteurs de support | AdaBoost |
| --- | --- | --- | --- | --- | --- | --- |
| Données brutes | 0.15 | 0.07 | 0.02 | 0.18 | 0.02 | 0.43 |
| Données prétraitées | 0.036 | 0.005 | 0.008 | 0.17 | 0.004 | 0.43 |

**Avec les hyper-paramètres suivants pour les données brutes :**

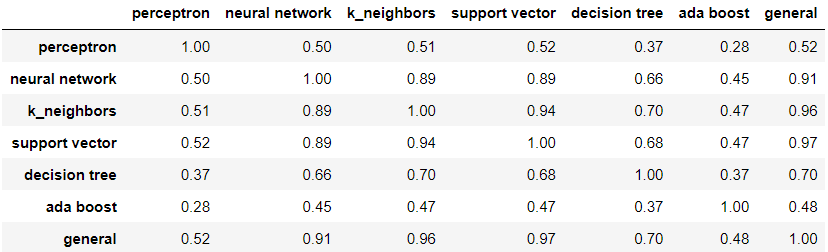
* Perceptron : penalty : l2, learning rate : 0.0001, alpha : 0.001
* Réseaux de neurones : alpha : 1.0e-5
* K voisins : n\_neighbors : 1, p : 1
* Arbre de décision : criterion : gini, ccp\_alpha : 0.0
* Vecteurs de support : c : 4.0
* AdaBoost : n\_estimators : 90, learning\_rate : 0.046

**Avec les hyper-paramètres suivants pour les données normalisées :**

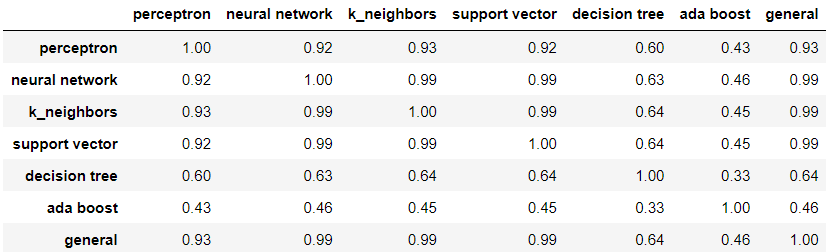
* Perceptron : penalty : l2, learning rate : 0.0001, alpha : 1.0e-7
* Réseaux de neurones : alpha : 0.0046
* K voisins : n\_neighbors : 1, p : 1
* Arbre de décision : criterion : gini, ccp\_alpha : 0.0004
* Vecteurs de support : c : 3.0
* AdaBoost : n\_estimators : 90, learning\_rate : 0.046

Comparaison entre les méthodes

**Avec données brutes**



**Avec données normalisées**



**Conclusion**