Leaf classification

BA Amadou[1], YING Xu[2] and ABOU Hamza[3]

***Abstract*— Les végétaux sont fondamentaux pour les êtres humains, il est donc très important de cataloguer et de préserver toutes les espèces végétales. L'identification d'une espèce de plante inconnue n'est pas une tâche simple. Les techniques de traitement automatique des images basées sur la reconnaissance des feuilles peuvent aider à trouver les meilleures caractéristiques utiles pour la représentation et la classification des plantes.**

De nombreuses méthodes sont présentées dans la littérature et utilisent un ensemble restreint et complexe de caractéristiques, souvent extraites d'images binaires ou de la limite de la feuille. Dans ce travail, nous allons implémentrer six modèles de classification en utilisant comme dataset la base de données Leaf-classifier du site Kaggle. Cette base de données est composée en particulier d'un fichier train.csv fournissant 990 feuilles décrites à travers 192 caractéristiques et un dossier d'images binaire. Notre code source est disponible à travers ce lien `https://github.com/AmadouSadouAbihi/Leaf-classification`.

I. INTRODUCTION

La classification des plantes est à la base de la science de la botanique ; c'est aussi le fondement de la génétique végétale, l'écologie, la médecine des plantes et la science des fichiers. Les méthodes traditionnelles de classification des plantes sont principalement dépendantes du jugement du sujet du chercheur. De plus, il est difficile de satisfaire le besoin que les gens veulent identifier rapidement les plantes ; par conséquent, la reconnaissance automatique de l'installation était nécessaire.

Une étude sur la reconnaissance des plantes basée sur le traitement d'images a été développée rapidement par collaborer avec les biologistes, notamment le botaniste et l'informaticien. De nombreux chercheurs ont été leur intérêt pour l'identification automatique des plantes par ordinateur. Différent avec une plante classification traditionnelle, cette méthode est rapide et ne dépend pas du jugement subjectif de la personne. En utilisant les méthodes d'apprentissage automatique de pointe, cette tâche sera plus simple et plus rapide, donc elle pourra aider les gens à mieux connaître les plantes et ceci de façon plus rapide.

La reconnaissance automatique des plantes est toujours une tâche difficile, non seulement dans les phases de classification, mais aussi dans phases d'extraction et de prétraitement d'images. Certains chercheurs ont publié leur étude concentrée sur l'extraction des fonctionnalités. Dans le cadre de notre projet de session, nous avons choisi d'utiliser la base de données Leaf-classifier du site Kaggle. Cette base de données est composée en particulier d'un fichier train.csv fournissant 990 feuilles décrites à travers 192 caractéristiques et un dossier d'images binaire. Chaque feuille est ainsi associée à une image.

[1]BA Amadou est étudiant à l'université de Sherbrooke au Quebec sous le Matricule de 16 187 314 Email: amadou.ba@usherbrooke.ca

[2]YING Xu est étudiante à l'université de Sherbrooke au Quebec sous le Matricule de 18 205 032 Email: xu.ying@usherbrooke.ca

[2]ABOU HAMZA est étudiant à l'université de Sherbrooke au Quebec sous le Matricule de 17 057 836 Email: hamza.abou@usherbrooke.ca

II. *FORMULATION DU PROBLEME*

1) ***Présentation du problème***
   Notre principal objectif sera d'expérimenter quelques méthodes de classification sur cette dataset, à la suite de différentes approches de résolution de problème qui varie selon le type de prétraitement des données effectué pour enfin analyser et interpréter les résultats obtenus.
2) ***Démarche scientifique de résolution du problème***
   Pour résoudre le problème qui nous est soumis, nous avons jugé utile d'adopter trois démarches scientifiques afin d'espérer obtenir les résultats escomptés en fonction des algorithmes de classification qu'on choisira :
   - ***Première approche***
     Dans cette méthodologie de résolution de problème, nous allons faire le minimum de prétraitement des données et sans faire de réduction de dimensionalité de la dataset par analyse en composantes principales (PCA). Pour se faire nous allons travailler qu'avec les données d'entrainement et de test

basiques disponibles dans les fichiers test.csv et train.csv c'est-à-dire les caractéristiques des images associées à chaque donnée ne seront pas prises en compte.

- ***Deuxième approche***
  Cette méthode est fortement similaire à la première approche à une différence près. Dans cette deuxième méthodologie, nous allons faire la réduction de dimensionalité par analyse en composante principale.
- ***Troisième approche***
  Celle-ci se différencie des deux premières approches par la nature des données qui seront mis en jeu. En effet, dans la troisième approche nous avons combinés les données numériques utilisées dans les deux premières approches avec quelques caractéristiques des images qui leur sont associées pour ainsi augmenter la précision dans la prédiction. De même dans la première approche, une analyse en composante principales ne sera pas faite pour réduire la dimensionalité de la dataset
- ***Quatrième approche***
  La dernière méthodologie de résolution est similaire à la troisième dans la nature des données en question et différente dans le nombre de dimensionnalité car ici, en plus des données numériques combinées aux caractéristiques des images qui leur sont associées, une analyse en composantes principales sera faite afin de réduire la dimensionalité de la dataset.

Dans chacune des approches, une recherche d'hyperparamètres sera effectué par cross validation en fonction de la méthode de classification en question.

3) ***Gestion de projet***

Pour mener à bien notre projet, il nous fallait un bon outil de gestion de projet qui serait très facile d'utilisation. Apres concertation avec les différents membres de notre équipe, nous avons retenu Trello, un outil en ligne de gestion de projet reposant sur une organisation des projets en planches listant des cartes, chacune représentant des tâches qui sont assignables à des utilisateurs et sont mobiles d'une planche à l'autre, traduisant leur avancement.

https://trello.com/b/DV0gIPlg/ift-712-machine-learning

Aussi étant appelé à produire des programmes informatiques, il nous indispensable d'utiliser un outil de versionning, pour la gestion des différentes versions de nos programmes mais également la traçabilité de ce qui a été fait et par qui afin de responsabiliser tout un chacun. Etant donné aussi l'existence de différents outils de ce genre, nous avons choisi de travailler avec Git du fait qu'il nous offre comme solution son caractère réparti permettant ainsi chaque programmeur de travailler localement sur une partie du projet pour ensuite publier son travail afin qu'il soit centralisé.

III. EXPERIMENTATIONS

1) *Compréhension de la dataset*
L'ensemble de données sur les feuilles des plantes de l'UCI sur laquelle nous travaillons comprend cent espèces de feuilles, chaque espèce en comptant seize des spécimens distincts. L'image originale est une image en couleur sur fond blanc. Cet ensemble de données très est difficile parce qu'il contient cent quatre vingt quatorses caractéristiques. La classification multi classe est une tâche importante en machine Learning, car de nombreux algorithmes manquent pour classer l'ensemble de données en grandes classes de données. Afin de faire une étude complète, nous allons utiliser quatre approches dont deux vont faire une réduction de dimensionnalité.
Ainsi les données d'entrainement a un total de ***990 lignes pour 194 colomnes***. Par soucis de pédagogie nous allons pas nous amuser à tous les citer dans ce rapport. Cependant , le tableau de la *figure 1* comprend les cinq première ligne de la dataset d'entrainement:

	id	species	margin1	margin2	margin3	shape1	shape2	shape3	texture1	texture2	texture3
0	1	Acer_Opalus	0.007812	0.023438	0.023438	0.000647	0.000609	0.000576	0.049805	0.017578	0.003906
1	2	Pterocarya_Stenoptera	0.005859	0.000000	0.031250	0.000749	0.000695	0.000720	0.000000	0.000000	0.007812
2	3	Quercus_Hartwissiana	0.005859	0.009766	0.019531	0.000973	0.000910	0.000870	0.003906	0.047852	0.008789
3	5	Tilia_Tomentosa	0.000000	0.003906	0.023438	0.000453	0.000465	0.000473	0.023438	0.000977	0.007812
4	6	Quercus_Variabilis	0.005859	0.003906	0.048828	0.000682	0.000598	0.000509	0.039062	0.036133	0.003906

Fig. 1: Cinq premières lignes de l'ensemble d'entrainement.

Étant donné que la variable de réponse est catégorique et se trouve actuellement en format texte, il faudra les encoder sous forme d'étiquette pour que ces réponses puissent être transmises aux classificateurs sans problème. Avec les 99 classes, les étiquettes seraient comprises entre 0 à 98. Pour celà nous allons utiliser la classe ***LabelEncoder*** de la bibiothéque de ***Sklearn***.

Les données de test comportent un total de 594 lignes et 193 colonnes. Les colonnes en test sont exactement les mêmes qu'en train moins la variable de réponse - ***species***. Les 594 lignes représentent 6 échantillons pour chacune des 99 espèces identifiées dans le train.

2) ***Algorithmes de classification utilisés***
Pour avoir de bon résultat, on a choisi d'utiliser des modèles variés. Autant de classifieur qui sont linéaire que de classifieur non linéaire. Ainsi que des modèle combiné comme le "Adaboost classifier". Les sept modèles utilisés sont les suivants :

- ***La régrssion logistique***
  La régression logistique est un modèle de régression linéaire mais au lieu d'optimiser une fonction linéaire, la régression logistique cherche à optimiser une fonction de perte plus complexe "sigmoid". Pour un meilleur apprentissage nous avons fait une recherche du meilleur hyper-paramètre pour l'attribut du terme de régularisation.

- ***Les machines à vecteurs de support (SVM)***
  La classe Svm représente une machine à vecteur de support pouvant utiliser les kernels rbf, sigmoidal et polynomial. Nous avons pu remarquer à travers différentes cross-validations que le kernel le plus optimal pour notre jeu de données était le polynomial. Nous avons aussi joué avec les hyper-paramètres suivants : C paramètre de régularisation; gamma , coefficient multiplicateur du noyau; coef0 ,terme indépendant (i.e. constante) du noyau rbf et polynomial et degree , le degré du noyau polynomial.

- ***Random Forest***
  Une forêt d'arbres décisionnels est un classifieur combinant plusieurs arbres de décision. Dans le cadre de la classification de nos données et pour mieux entraîner notre modèle nous avons appliqué la cross-validation sur les paramètres suivants : la profondeur maximale d'un arbre de décision et le nombre d'arbres de décision.

- ***AdaBoost***
  Tous les algorithmes d'apprentissage tendent à correspondre plus à certains types de problèmes qu'à d'autres, et ont typiquement de nombreux paramètres et configurations différents qu'il est nécessaire d'ajuster pour atteindre une performance optimale sur un ensemble d'apprentissage fourni. AdaBoost (avec des arbres de décision comme classeurs faibles) est souvent désigné comme le meilleur classeur clé-en-main. Afin de mettre en pratique les techniques de boosting, nous avons choisi d'utiliser la classe implémentant l'algorithme AdaBoost sur des arbres décisionnels (aussi appelés, stump decision tree). Ici, nous avons utilisé la cross-validation sur les paramètres suivants :baseestimator : la classe du modèle boosté : en particulier, nous avons testé différentes valeurs pour le paramètres maxdepth des arbres décisionnels; le nombre maximum de modèles combinés et le learning rate .

- ***Neural network***
  Pour effectuer une classification par réseaux de neurones on a appliqué le MLPClassifier de sklearn, en effet le Perceptron MultiCouches (PMC) est un des réseaux de neurones les plus utilisés actuellement en apprentissage supervisé. Dans notre cas, pour améliorer les résultats de notre modèle, nous avons tenté d'effectuer une cross-validation en optimisant les paramètres suivants :le nombre de couches cachées**, le taux d'apprentissage initial**, la ***fonction d'activation*** 'identity', 'logistic', 'tanh', 'relu' et le ***solver*** 'lbfgs', 'sgd', 'adam'

- ***Analyse du discriminant linéaire***
  L'analyse discriminante linéaire est une techniques d'analyse discriminante prédictive généralisant l'analyse discriminante de Fisher. Elle a pour objectif de classifier linéairement des entités à partir de combinaisons linéaires des caractéristiques initiales. Dans ce classifieur, nous pouvons utiliser différents solveur et différentes valeurs pour shrinkage .Cependant, le solveur eigen nécessite que la matrice des données soit inversible (ce qui n'est pas le cas lorsqu'on utilise les données

des images) et le solveur svd impose de ne pas avoir de shrinkage.

3) ***Première approche***

a) ***Prétraitement des données***

L'ensemble de donnée n'est composé que des données numériques fournies par les fichiers *train.csv* et *test.csv*. nous rappelons que ce preprocessing se fait sans analyse en composante principale. Cependant , vu que la variable qui va nous servir de variables de réponse est catégorique (format text) , il nous faudra l'encoder sous forme d' etiquette pour que ces réponses puissent êtretransmises aux classificateurs sans problème. Avec les 99 classes, les etiquettes seraient comprises entre 0'a 98. Pour celà nous allons utiliser la classe LabelEncoder de la bibiothèque Sklearn.

Ensuite, aprés cette encodage, nous allons les "stratifier" et les "spliter" en données d'entraiment (80% des données d'entrainement initiales) soit 792 samples et en données de validation (les 20% restantes) soit les 198 samples restantes. Cette étape sera suivi d'une standarisation des données en utilisant Standard Scaler de Scklearn afin de transformer chacune de ces 192 caractéristiques en en une variance unitaire moyenne nulle "*mean unit variance*".Par la suite , nous procédons à l'entraintement de nos algorithmes.

b) ***Entrainement des algorithmes***

Une fois l'étape précédente terminée, on peut procéder directement à l'entraînement des modèles. Juste dans le but d'avoir des premières métriques. Certes, cette étape d'entraînement permet de sélectionner les bons paramètres de chaque algorithme de classification mais chaque modèle a des "hyperparameter" à choisir astucieusement.

c) ***Cross validation***

La cross-validation permet de sélectionner les bons "hyperparameter" qui vont de pair avec chaque modèle de classification. Comme stratégie de cross-validation, on a utilisé la "k fold cross validation" que l'on peut faire facilement à partir de la méthode "gridsearchcv" (qui prend comme paramètre "stratifiedkfold") dans la bibliothèque sklearn.

4) ***Deuxième approche***

a) ***Prétraitement des données numériques***

Ce preprocessing se différencie de celui de la premiere approche par le fait qu'en plus de ce qui se fait dans la première méthodologie, nous allons procécéder à une réduction de dimensionalité de la dataset en faisant une analyse en composantes principales(ACP).Effectuer l'ACP sur les données transformées, en excluant la variable réponse(species), en capturant au moins 99% de la variabilité des données. Sans préciser au départ le nombre de composantes, le graphique ci-dessous montre la somme cumulée de la variabilité saisie à mesure que la composante principale augmente.

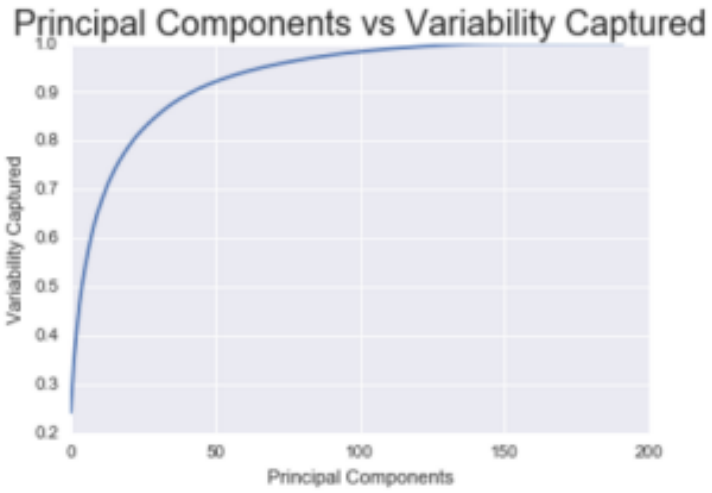

Fig. 2: Variance conservée en fonction du nombre de composantes

Ainsi, nous avons choisi de prendre 0.9 comme pourcentage de variance ce qui diminue le nombre des caractéristiques de 192 à 130. (soit une réduction de 32,29%) en éliminant seulement 0.1% de la variance.

b) ***Entrainement des algorithmes***

Comme dans la première approche après le pré-traitement des données on peut procéder à l'entraînement pour avoir des modèles permettant de généraliser le dataset.

c) ***Cross validation***
Après avoir entraîner les modèles, il est convenable de choisir les ”hyperparameter” adéquats en gardant la même stratégie de cross validation utilisée lors de la première approche.

5) ***Troisième approche***

a) ***Prétraitement des données combinées aux images associés***
Dans cette partie , en plus du prétaitement des données faite dans la première approche, nous allons rajouter aux données certaines caractéristiques extraintes des images qui leur sont associées.Pour chaque image, nous avons choisi d'extraire les caractéristiques suivantes:

- le pourcentage de pixels noits
- le pourcentage de pixels blanc
- le rapport de la largeur de la feuille d'arbre sur la longueur
- le nombre de sommets de la feuille
- l'extentricité de l'ellipse
- l'angle de déviation de l'ellipse
- le gradiant de la droite en approchant le contour par une droite
- l'image de l'abscisse 0 selon l'équation de la droite

En plus de cet ajout de caractérisques aux données , une analyse en composante principale ne sera pas faite

b) ***Entrainement des algorithmes***
Comme dans les autres approches après le pré-traitement des données on peut procéder à l'entraînement pour avoir des modèles permettant de généraliser le dataset.

c) ***Cross validation***
Après avoir entraîner les modèles, il est convenable de choisir les ”hyperparameter” adéquats en gardant la même stratégie de cross validation utilisée lors de la première approche.

6) ***Quatrième approche***

a) ***Prétraitement des données numériques avec PCA combinées aux images associés***
Ce preprocessing se différencie de celui de la précédante approche par le fait qu'en plus de ce qui se fait dans la troisième méthodologie, nous allons procécéder à une réduction de dimensionalité de la dataset en faisant une analyse en composantes principales(ACP)

b) ***Entrainement des algorithmes***
Comme dans les autres approches après le pré-traitement des données on peut procéder à l'entraînement pour avoir des modèles permettant de généraliser le dataset.

c) ***Cross validation***
Après avoir entraîner les modèles, il est convenable de choisir les ”hyperparameter” adéquats en gardant la même stratégie de cross validation utilisée lors de la première approche.

IV. ANALYSE ET INTERPRETATIONS DES RESULTATS

a) ***Régression logistique***
Comme nous le voyons sur la figure ci-dessous,et comme nous pouvions l'imaginer, la regression logistique , qui est un classifieur linéaire parvient difficilemment à classifier nos données qui ne sont pas du tout linéaires et ceci quelque soit l'approche de prétraitement utilisée.

Hyperparameters	penalty	accuracy (train)	accuracy (test)
Première approche	L2	58.7369 %	56.0459 %
Deuxième approche	L2	56.6034 %	57.1209 %
Troisième approche	L2	54.0899 %	51.0023 %
Quatrième approche	L2	51.0873 %	51.0518 %

Fig. 3: Résumé des résultats obtenus pour la régression logistique

b) ***Support vector Machine (SVM)***
comme le montre la figure suivante, la prmière approche avec le noyau polynomiale est beaucoup plus performant que les autres approches.Vu que nos ressources de calcul sont limitées, nous avons pas pu travailler avec le noyau polynomiale pour les approche 1, 2 et 3 car l'eentraitement de ce dernier prenait trop de temps. Nous étions alors contraint de travailler avec le noyau 'rbf' ou 'simoid'

tout en chassant que ces deux noyaux souffre de surapprentissage malgré le choix des hyperparametres par la cross validation.

Hyperparamètres	Kernel	C	Gamma	Coef0	Degree	Accuracy(test)
Première approche	'Poly'	0.0001	0.0001	17.5	9	95.9596 %
Deuxième approche	'Rbf'	2e-137	0.01	0	None	17.6768 %
Troisième approche	'Rbf'	743	2.6e-05	0	None	28.2828 %
Quatrième approche	'Rbf'	743	2.3e-05	0	None	30.0035 %

Fig. 4: Résumé des résultats obtenus pour la SVM

En regardant de prés ces résultats , on peut dire que l'analyse en composantes principales n'est pas forcément bénéfique dans le prétraitement des données.

c) ***Random Forest***

Malgré une dimmunition de l'efficacité avec l'ACP , on remarque que pour ce classifier les résultats ont été trés bonnes. Ceci sans doute du fait qu'ici nous avons une combinaison de modèle (bagging + decison tree).

Hyperparamètres	Max_depth	N_estimators	Accuracy(test)
Première approche	60	99	98.4848%
Deuxième approche	70	88	97.9798%
Troisième approche	50	89	92.9293%
Quatrième approche	75	89	95.0732 %

Fig. 5: Résumé des résultats obtenus pour la Random Forest

d) ***AdaBoost***

D'aprés les résultats obtenus , on remarque amélioration dans la précision quand on utilise le PCA et les caractéristiques de images.

Hyperparamètres	Base_ estimator	N_estimators	Learning_rate	Accuracy(test)
Première approche	Max_depth = 13	88	0.016681	97.9798 %
Deuxième approche	Max_depth = 13	84	0.027825	98.4848 %
Troisième approche	Max_depth = 13	86	0.1	92.4242 %
Quatrième approche	Max_depth = 13	85	0.09	93.0967 %

Fig. 6: Résumé des résultats obtenus pour AdaBoost

e) ***Neural Network***

Comme nous pouvons le constater d'après les résultats ci-dessous, le perceptron multi-couche a beaucoup plus de mal avec les données numériques associés aux images. En effet, dûe à de long temps de convergence, nous n'avons pas réussi à effectuer de cross validation sur les hyperparamètres. Nous avons remarqué que pour un nombre maximum d'itération 10000, le réseau ne converge toujours pas mais nous pouvons atteindre une précision de l'ordre des 75% en test avec toutes les données sans ACP. Nous pouvons donc en conclure que le PCM donne de bons résultats lorsqu'il est entraîné de manière prolongée avec une haute valeur de maxiter. Cependant, pour notre cas d'utilisation, le PCM n'est pas un bon classifieur.

Hyperparamètre	Hidden_layer_sizes	Learning_rate_init	Activation	Solver	Accuracy(test)
Première approche	(104,19)	0.0522	Identity	adam	88.0788%
Deuxième approche	Défaut	1.0	Identity	adam	53.5445%
Troisième approche	Défaut	1.0	Identity	adam	35.6730 %
Quatrième approche	Défaut	1.0	Identity	adam	33.6879 %

Fig. 7: Résumé des résultats obtenus pour Neural Network

f) ***Linear Discriminant Analysis***

Voici des exemples de cross-validation, comme nous pouvons le remarquer, les meilleurs résultats sont obtenus lorsque nous ne réalisons pas d'ACP. Nous pouvons imaginer que cela s'explique par le fonctionnement du classifieur qui réalise, lors de son entraînement, un travail similaire à l'ACP. Ainsi, cet algorithme serait plus performant avec l'ensemble des caractéristiques plutôt qu'avec des caractéristiques appauvri en termes d'informations sur les feuilles.

Hyperparamètres	Solver	Sprinklage	Accuracy (test)
Première approche	'lsqr'	3.0888e-06	98.5848 %
Deuxième approche	'lsqr'	7.1968e-12	98.09798 %
Troisième approche	'Lsqr'	1e-12	95.0495 %
Quatrième approche	'Lsqr'	1e-12	96.2493 %

Fig. 8: Résumé des résultats obtenus pour Linear Discriminant Analysis

V. CONCLUSION

En conclusion, nous pouvons noter que les meilleurs résultats de classification ont été obtenus par les combinaisons de modèles

Random Forest et AdaBoost, ainsi qu'avec l'analyse discriminante linéaire. De plus, contrairement à ce que l'on pourrait penser, l'ACP est rarement bénéfique aux algorithmes de classifications lorsque l'on regarde les performances. Cependant, nous avons remarqué que l'ACP avait tendance à accélérer la vitesse d'entraînement, ce qui peut ainsi parfois contrebalancer la perte de performance.

VI. REFERENCES ET BIBLIOGRAPHIE

- https://github.com/gycggd/leaf-classification/blob/master/src/generate_data.py
- https://github.com/danielangus/leaf-classification/blob/master/code/explore.ipynb
- https://github.com/SeemaKanuri/Leaf-Classification
- https://www.kaggle.com/c/leaf-classification
- https://scikit-learn.org/stable/
- https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_contours/py_contour_features/py_contour_features.html
- https://pillow.readthedocs.io/en/stable/reference/Image.html
- https://sciki-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html
- https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_gui/py_image_display/py_image_display.html
- https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedShuffleSplit.html
- https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html