Développer une plateforme de classification pour la reconnaissance des formes

FELIX Jérémy, développeur
KALKA-DEBIDINE Ludovic, Coordinateur
LESMOND Elkana, développeur
MOUSSA Issa-Paul, Chef de projet

Table des matières

I. INTRODUCTION 3

II. ORGANISATION 3

III. IDEES 3

A. CHOIX DES CLASSIFIEURS 3
B. CHOIX DE PRETRAITEMENTS DES DONNEES 3
C. FORMAT D'ENTREE DES DONNEES 4
D. LANGAGE DE PROGRAMMATION 4
E. BASE D'EXEMPLE 4
F. INTERFACE 4

IV. DEROULEMENT ET PROBLEMES RENCONTRES 5

V. CONCLUSION 6

I. Introduction

Ce rapport présente les idées et les moyens qui ont permis de concevoir la plateforme de classification. Cette plateforme à pour objectif de réunir plusieurs classifieurs (supervisé ou non, différents types, ...) et de traiter les données (régularisation, outils de prétraitement, ACP, ...) avant de les classifier afin de fournir plusieurs résultats et de choisir les meilleurs en fonction du classifieur et des traitements utilisés sur les données à analyser.

II. Organisation

Afin de mener au mieux ce projet, nous avons décidé d'utiliser certains outils pour faciliter l'échange de fichiers, le développement logiciel et la gestion du projet.
Ces outils sont les suivants :

- **GitHub (+ GitKraken),** comme logiciel collaboratif pour le développement,
- **Google Drive**, pour l'échange de fichiers rapide sans « push »,
- **Gantter**, pour gérer les différentes tâches des membres de l'équipe.

III. Idées

A. Choix des classifieurs

Nous avons initialement choisi d'implémenter les classifieurs vues durant notre cursus du fait qu'ils soient déjà appréhender et donc l'interprétation de leurs résultats serait plus facile et rapide à concevoir.
Voici les différents classifieurs (supervié ou non) qui devraient être implémentés :

- Kppv,
- Bayésien,
- MLP,
- Konhen,
- K-Means,
- PAM,
- SVM.

B. Choix de prétraitements des données

Certains classifieurs peuvent nécessité un prétraitement comme pour MLP qui a besoin d'un vecteur de sortie désiré ou comme normaliser les données pour certains classifieurs. Aussi, dans le cas où il y a beaucoup de dimensions, on peut utiliser des méthodes afin de réduire ces dimensions en prenant celles contenant le plus d'informations comme avec l'ACP ou l'AFD.

C. Format d'entrée des données

Le format d'entrée des données doit être commun aux différentes provenances des données (images, sons, ...). On choisit le format CSV, pour sa simplicité, où l'on inscrira les dimensions de chaque donnée :

Donnée 1 : dim1, dim2, dim3, ... dimn
Donnée 2 : dim1, dim2, dim3, ... dimn
.
.
.

Donnée n : dim1, dim2, dim3, ... dimn

Dans le cas où l'utilisateur choisit un apprentissage supervisé, c'est la dernière dimension qui sera prise en compte comme étant la classe de la donnée. Dans le cas contraire, toutes les dimensions sont prises en compte. Aussi, on considère que les données sont toutes numériques et qu'il n'y a pas de données manquantes.

D. Langage de programmation

Nous avons décidé d'utiliser Java pour la plateforme qui va intégrer des méthodes provenant de différents autres langages (comme R pour K-means et PAM, C pour mlp et kohnen, ...). La raison est que Java est plus facile à utiliser pour produire des interfaces graphiques que d'autres langages et qu'il existe beaucoup d'API ou librairies afin d'utiliser la majorité des langages en Java.

E. Base d'exemple

Deux jeux de données seront utilisés afin de mener les tests :

- **Dataset_2.csv**, qui le taux de cholestérol, le poids et le genre de personnes,
- **bezdekIris_data_v2**, qui est une base simplifiée d'Iris.

F. Interface

L'interface que nous avons imaginé pour cette plateforme se décrit comme une fenêtre divisée en deux (une partie à gauche, et une partir à droite plus grande). La partie de gauche comporterait des éléments écrits (comme des informations sur le classifieur, sur les données et les performances mais aussi des messages propres au logiciel comme les erreurs) et la partie de droite comporterait un graphique correspondant à la performance du classifieur).

C'est à partir des onglets que l'on pourra choisir d'ouvrir un jeu de données, d'appliquer des prétraitements ou d'utiliser un classifieur. On y trouverait aussi la description de certains classifieurs et de leurs paramètres.

Comme on prévoit que certains jeux de données peuvent présenter une première ligne concernant le nom des attributs ou une dernière colonne qui peut ne pas représenter la

classe d'une donnée, une fenêtre lors du chargement de ces jeux de données permet à l'utilisateur de confirmer deux points.

IV. Déroulement et problèmes rencontrés

Avant de passer à la programmation, nous avons regardé quelles libraires possédaient déjà la plupart des classifieurs. Ainsi, le logiciel WEKA et le langage R possédaient déjà des classifieurs et méthodes de prétraitements dans leurs librairies et étaient sous licence libre. Nous avons donc décidé de se concentrer sur ces deux outils.

Cependant, le R a rapidement été préféré du fait qu'on pouvait manipuler comme on le voudrait, les résultats que l'on obtenait à partir d'un appel de fonction de R dans Java, et que pratiquement tous les classifieurs (Kppv, Bayes, MLP, K-means, Pam) étaient déjà intégrer dans R. Avec Weka, c'était semblable à une utilisation unique du logiciel dont le résultat final est ensuite récupéré dans Java (c'est comme si on ouvrait un jeux de données avec Weka, qu'on applique tous les traitements que l'on voudrait et que ce sont les résultats après ceci qui pouvaient être manipuler en Java).

Il y a deux façons d'intégrer R en Java, une comme librairie dont seul les méthodes de bases peuvent êtres utilisées et une autre en « client-serveur » ou le client Java fait des appels à un serveur R qui peut comporter tous les packages existants. C'est donc la deuxième méthode qui est choisie, fonctionnant plus précisément comme suit :

- un code Java se connecte à un serveur et demande à ce dernier d'exécuter un script contenant du code R faisant le traitement souhaité,
- le serveur R exécute ce script et renvoi les résultats au client Java, ces résultats peuvent ensuite être transformer en n'importe quel type Java et être manipuler.

On pouvait demander à R de produire des matrices de confusions, de prédire les classes de la base de généralisation à partir de la base d'apprentissage ou même de produire des images graphiques concernant les performances d'un classifieur.

Comme certains jeux de données sont ordonnés en fonction de leur classe, le programme se charge de mélanger systématiquement ces jeux lors de leur chargement.

Il suffisait donc de charger un jeu de données en Java, de le transformer pour qu'il puisse être communiqué au serveur R afin d'appliquer une méthode que ce langage possède, comme appliquer un classifieur, puis récupérer ces résultats afin de les traiter et de les placer dans l'interface.

Néanmoins, nous avons rencontrés plusieurs problèmes au niveau de l'interprétation des résultats de R en Java. En effet, on s'attendait à ce que ces derniers comportent tous au moins, les classes qui sont attribuées à la base de généralisation afin de pouvoir les comparer et en tirer des performances mais seules certaines méthodes de classifications de R le faisait directement, et pour les autres, on appliquait d'autres méthodes de R permettant d'obtenir des performances mais ces dernières n'étaient pas explicites et ces méthodes ne pouvait pas s'appliquer à tous les classifieurs.

Le manque de documentation concernant ce que chaque méthode renvoyait nous a beaucoup pénalisé. C'est par l'intermédiaire des forums sur le web, des exemples et de l'analyse des résultats que l'on arrivait à comprendre les résultats ou les graphiques. Par exemple, un résultat récupérer en Java après avoir appliquer une méthode de classification, est une liste de plusieurs éléments résumant par exemple les paramètres entrés et donnant un tableau de même taille que la base de généralisation. On déduisait donc qu'il contenant les prédictions des classes, toutefois, chaque méthode de classification avait un retour différent et n'avait pas d'éléments aussi explicites.

V. Conclusion

Au final, les classifieurs implémentés sont les kppv, bayes, MLP, K-means et Pam. Parmi les méthodes de prétraitements, on a pu implémenté que la normalisation des données.

Concernant les images graphiques, nous avons choisi, pour trois classifieurs (les autres ne présentait pas forcément un ou du moins compréhensible), un graphique qui nous semble pertinent.

Pour chaque classifieur, nous mettons dans la partie gauche de l'interface des informations concernant le jeux de données et les paramètres du classifieur avec seul le taux de succès (qui est calculé manuellement en Java) comme performance (à part MLP du fait qu'il ne présentait pas des résultats pouvant le calculer) du fait du manque de documentation concernant l'obtention de d'autres élément de performance.

L'abandon d'un des membres de l'équipe et le temps difficile à gérer ne nous a pas permis de trouver une solution rapide et efficace à ce problème. Il aurait fallu faire une étude plus approfondie des différents outils, librairies ou autres possédant déjà des méthodes de classifications et de prétraitements et comportant précisément ce que l'on obtient comme type de résultat et de peser le pour et le contre même s'il est vrai que le langage R présente plusieurs types de résultats pertinents à condition de savoir les interpréter correctement.