UTT - MS-EBAM - ARI2 - Sujet du projet informatique - 20 Novembre 2020

Pour ce projet, vous devez rendre seulement un notebook python. Ce code doit donc contenir des commentaires (succincts) et doit afficher directement les résultats de votre travail.
Les projets doivent impérativement (1) être remis le au plus tard le Vendredi 24 Janvier 2020, et, (2) être réaliser en binôme (ou tout seul) ; les groupes de plus de 2 verront leur note divisée par le nombre de personnes !
Pour toutes questions, vous pouvez me joindre par email : remi.cogranne@utt.fr

1 Présentation du projet

Le but de ce projet est d'appliquer les méthodes d'apprentissage supervisées vues en cours pour la reconnaissance de la langue d'un morceau de texte.

Les bases de données à utiliser vous sont fournies. Vous pouvez utiliser ces codes comme cela vous semble.
Ce code est surtout intéressant pour changer la taille des block de caractère extrait (plus cette taille est importante, plus la classification sera précise ...)

2 Travail demandé attendu

Le travail qui vous est demandé se décompose en plusieurs étapes. Ces dernières sont résumées succinctement ci-dessous ; notez qu'il s'agit d'une base de travail, tout amélioration ou travail supplémentaire sera bienvenu !

Pour commencer : classification linéaire à 2 classes

Pour commencer, on essayera de classifier deux langages (vous pourrez comparer les différences lorsque vous classifiez des langages proches, *i.e.* Latins (Français / Espagnol) ou Anglo-Saxons (Anglais / Allemand) et des langages de racines différentes. On utilisera les techniques de classification supervisée suivantes : Régression linéaire (Ridge), LASSO et SVM avec noyau linéaire.

Dans ce cas, il vous sera demandé

1. De mettre en place une méthode de validation-croisée de l'hyper-paramètre de régularisation (et d'interpréter le vecteur de projection obtenu) ;

2. D'essayer de réduire la dimension des données (et d'interpréter quelles sont les données les plus pertinentes) en utilisant, par exemple l'Analyse en Composantes Principales (ACP ou PCA en anglais) ;

Pour commencer : classification linéaire à 2 classes

On fera ensuite un travail similaire avec une méthode classification non-linéaire reposant sur l'astuce du noyau.
Il est recommandé d'utiliser pour cela le SVM et le noyau Gaussien / RBF.
Dans ce cas il sera particulièrement important de proposer une validation-croisée des deux hyper-paramètres (noyau Gaussian et régularisation) conjointement et de discuter l'intérêt d'une approche non-linéaire dans un tel contexte.
Vous pourrez également utiliser une méthode de réduction de caractéristiques non-linéaires (kernel PCA / kPCA)

Pour aller plus loin : classification à N classes

On va à présent faire de la classification en utilisant tous les langages simultanément.
On utilisera les 4 mêmes méthodes de classification que précédemment.
Notez que pour gagner du temps on pourra, si vous les souhaitez, utiliser un seul et unique hyper-paramètres appris dans le cas de la classification 2 classes de la section précédente.

Questions 1) Pour chacune des méthodes précédemment citées, comparer avec une matrice de confusion les performances obtenues en utilisant l'approche "one vs all" (nécessitant donc 4 classifieurs) et l'approche "classification 2 à 2" (nécessitant donc 6 classifieurs).
Pensez que le temps de calcul supplémentaire soit justifié au regard des résultats en termes de détection ?

2) Utiliser les 4 classifieurs précédents (déjà entraînés) pour implémenter une méthode multi-classe avec ensemble de classifieurs.
Donner la matrice de confusion et commenter quant au meilleur classifieur dans ce contexte multi classe.+

3 Livrable

Encore une fois, vous devez commenter (succinctement) votre code afin d'expliquer ce que vous faites et, si vous avez fait des choix d'implémentation particuliers, les justifier.
Il vous est également (et surtout) demandé de commenter vos résultats. Un code qui "simplement" produit ce qui est demandé sans aucune analyse sera très mal considéré.

Vous devez seulement envoyer votre notebook par email à l'adresse: remi.cogranne@utt.fr
Ce notebook doit contenir

1. Les codes permettant de résoudre les questions ;
2. Les "traces" / résultats d'une exécution ;
3. Dans le code, des commentaires pour justifier comment est implémenter les parties qui vous semblent intéressantes (et assurer la bonne compréhension) ;
4. Des cellules de discussion des résultats, d'analyse, permettant de montrer que vous pouvez comprendre et interpréter ce que vous obtenez (et c'est clairement là le plus important).