Classification multiclasses: légumes secs

Team : Kazou
Virgile Retault : 2164296
Sebastien Foucher : 2162248

April 18, 2022

Prétraitement des attributs

Tout d'abord il faut supprimer la colonne "id" dans les données car elle ne doit pas être utilisée pour la prédiction. Après une analyse des données à l'aide de la librairie panda, il est clair que certains attributs sont fortement corrélés. Ainsi il convient de sélectionner des attributs idoines et non redondants pour l'apprentissage.

Une première approche est de sélectionner manuellement ces attributs. Par exemple on remarque que l'attribut périmètre est très dépendant des attributs longueur des axes. On pourrait donc supprimer ces deux attributs pour réduire la dimension de l'espace de recherche. Cependant, optimiser ce choix est une tâche difficile et laborieuse.

Une technique pour automatiser ce processus consiste à utiliser une analyse en composantes principales pour réduire le nombre d'attributs en les décorrélant les uns des autres. Pour cela il est possible d'utiliser le module de décomposition PCA de sklearn. Après expérimentation le nombre d'attributs optimal à utiliser semble être de 12.

Enfin avant d'entrainer le modèles les attributs sont standardisés en leur retranchant leur moyenne pour les centrer sur 0 avec le module StandardScaler de sklearn.

Si on souhaite entrainer un modèle de réseau de neurones, il faut aussi vectoriser les données de sortie catégoriques avec de l'encodage one-hot.

Méthodologie

Pour évaluer entre eux les différents modèles sans faire de soumission sur Kaggle, les données ont été divisées en 60% de données d'entrainement et 40% de données de validation. La fonction train_test_split de sklearn permet de faire exactement cela.

En ce qui concerne l'optimisation des hyper-paramètres une méthode automatique qui utilise une recherche exhaustive a été implémentée avec le module GridSearchCV de sklearn. Elle permet de tester toutes les combinaisons d'hyper-paramètres spécifiés et d'en extraire la meilleure.

Certaines classes dans les données d'entraînement sont sur-représentées par rapport à d'autres. Il a été testé de ne conserver qu'une partie des données sur-représentées pour avoir une meilleure répartition des classes, mais cela n'a pas été concluant.

Voici les différentes méthodes qui ont été testées :

- Decision Tree
- Random Forest
- SVM
- Gradient Boosting
- Adaboost avec random forest
- K-Neighbors classifiers
- Un réseau de neurone dense de taille (12, 6, 6, 7) entrainé avec Keras
- Utilisation de plusieurs modèles en simultané avec un Stack Classifier

Résultats

Voici les scores obtenus avec ces différentes méthodes:

Méthode	Précision
K-Neighbors Classifier	80%
Decision Tree	89.6%
SVM avec noyau linéaire	90.1%
Gradient Boosting (avec PCA 12)	91.3%
Random Forest (sans PCA)	91.5%
Réseau de neurone (12, 6, 6, 7) (avec PCA 12)	92.2%
Adaboost avec random forest (avec PCA 12)	92.9%
StackClassifier combinant RandomForest, ExtraTree, AdaBoost (avec PCA 12)	93.3%
Random Forest (avec PCA 12)	93.4%

Discussion

La méthode par Random Forest utilisant une analyse par composantes principales avec 12 composantes est celle qui obtient le meilleur score sur Kaggle. Les optimisations faites au niveau du pré traitement des données, notamment la standardisation des attributs et l'analyse en composantes principale permettent ainsi de grandement améliorer les performances du modèles en réduisant le surentrainement.