Bagging, Random Forest, Boosting, Gradient Boosting

Nous travaillons avec Python + Scikit-Learn dans cet exercice

Documentation

Ces tutoriels devraient vous aider :

[TUTO 1] Arbres de décision avec Scikit-Learn

http://tutoriels-data-science.blogspot.com/p/tutoriels-en-francais.html#2125873889988820609

[TUTO 2] Random Forest et Boosting avec R et Python

http://tutoriels-data-science.blogspot.com/p/tutoriels-en-francais.html#4180029503214824554

[TUTO 3] Python – Machine learning avec Scikit-Learn

http://tutoriels-data-science.blogspot.com/p/tutoriels-en-francais.html#1766550852310168124

[TUTO 4] Gradient boosting avec R et Python

http://tutoriels-data-science.blogspot.com/p/tutoriels-en-francais.html#855656834581201180

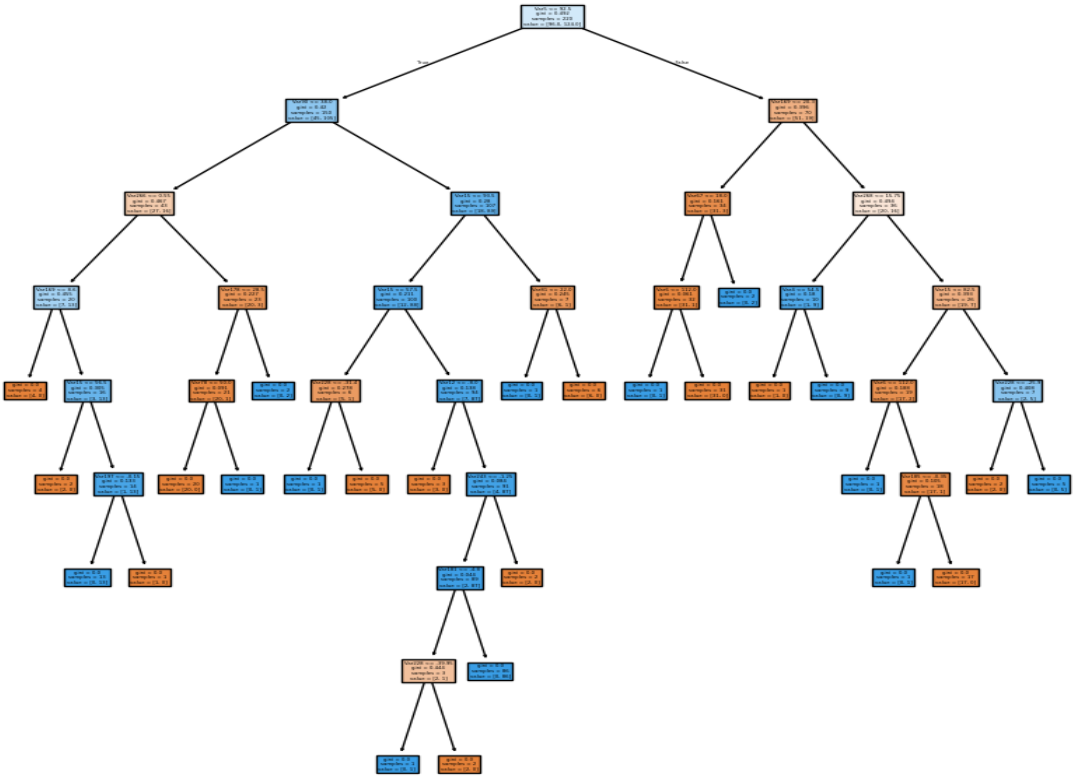
Importation et préparation des données

1. Les données « **arrhythmia.xlsx** » recensent l'apparition ou non de l'arythmie cardiaque (rythme = anormal, négatif ; rythme = normal, positif) chez des personnes décrites par une série de caractéristiques. Chargez les données et affichez-en les propriétés (420 observations ; 279 variables, dont la cible ‘’**rythme**’’ ; ce type de ratio nombre d'observations / nombre de variables [explicatives] n'est pas très favorable, il est propice à l'overfitting, on va surveiller cela).
2. Comptabilisez les effectifs par classe (**TUTO 1**, page 3 ; 237 ‘’normal’’, 183 ‘’anormal’’).
3. Subdivisez les données en échantillons d'apprentissage (220 obs.) et de test (200 obs.). Effectuez un échantillonnage stratifié (stratify) et fixez (random_state = 1) pour que nous ayons les mêmes découpages.
4. A titre de vérification, affichez les fréquences absolues des classes dans les échantillons d'apprentissage (124, 96) et de test (113, 87).

Arbre de décision

5. Affichez la version de « scikit-learn » pour calibrer nos résultats (1.5.1 pour moi, **c'est mieux si nous avons tous une version ≥ 1.0**).

6. Construisez un arbre de décision avec les paramètres par défaut sur l'échantillon d'apprentissage (mettez quand-même random_state = 0 pour que nous ayons les mêmes résultats) (**TUTO 1**, section 2.4 ; avec les paramètres par défaut dans notre cas !).
7. Affichez l'arbre de décision obtenu en coloriant les sommets selon la classe d'appartenance (TUTO 1, section 2.5). Agrandissez l'arbre à l'affichage pour mieux distinguer le contenu des sommets (**TUTO 1**, page 6).



8. Pour disposer d'un autre point de vue, nous affichons l'arbre de manière textuelle (**TUTO 1**, section 2.6).
9. Affichez l'importance des variables (**TUTO 2**, page 24). Quelles sont les 3 variables les plus influentes ? (Var15, Var5, Var90) (**TUTO 1**, section 2.7)
10. Appliquez l'arbre sur l'échantillon test (predict), affichez la matrice de confusion (confusion_matrix) et calculez le taux de reconnaissance (ou taux de succès) (accuracy_score) (**TUTO 1**, section 2.8) (**0.68**).

Bagging

11. Refaites l'analyse ci-dessus en utilisant cette fois-ci un « bagging » d'arbres de décision (**TUTO 2**, page 25 et suivantes). Demandez n_estimators = **100 arbres**, demandez

l'estimation du taux de reconnaissance avec la méthode out-of-bag (oob_score=True), fixez (random_state = 1**)** pour que nous ayons exactement les mêmes résultats, et passez en (estimator) l'arbre construit précédemment pour que nous ayons tous le même paramétrage du modèle sous-jacent.

12. Quel est le taux de succès mesuré avec l'approche out-of-bag ? (oob_score_) (0.795)
13. Quel est le taux de succès en test ? (**0.82**). Quelle conclusion peut-on en tirer par rapport à la performance de l'arbre individuel précédent ? (la combinaison améliore les performances visiblement)
14. D'un côté, OOB-Score vous propose une accuracy de 0.795, de l'autre en test (estimé sur 200 obs.) nous avons 0.82. Quelle autre estimation pouvons-nous produire ? (*j'ai utilisé une 10-CV, j'ai obtenu 0.809 ; la vérité est entre ces différentes valeurs…*)

Random Forest

15. Refaites l'analyse dans les mêmes conditions avec la méthode « random forest » (**TUTO 2**, page 29 et suivantes). Faites calculer (oob_score = True) puis affichez l'erreur « out-of-bag » (oob_score_) (0.80).
16. Affichez l'importance des variables (Var8, Var15, Var233, etc.). Que constatez-vous par rapport à l'arbre individuel ?
17. Le mécanisme spécifique de sélection de variables sur les nœuds inhérent à « random forest » a-t-il porté ses fruits ? (taux de succès en test = 0.765). Pourquoi Random Forest ne se démarque pas par rapport au Bagging dans notre contexte ? (le nombre de variables non-pertinentes est élevé et s'immiscent dans les arbres individuels, dégradant leurs performances, c'est un des points faibles des Random Forest)

Boosting

18. Reproduisez l'analyse précédente avec la méthode « adaboost » (**TUTO 2**, page 30 et suivantes). Demandez explicitement un arbre comme méthode sous-jacente [base_estmator = DecisionTreeClassifier(random_state=0)]. Que pensez-vous des performances ? (taux de succès = 0.685). Qu'est-ce qui pourrait expliquer une telle déconvenue ? (sur-apprentissage vraisemblablement, les arbres individuels sont trop profonds)
19. Pour remédier à cette mauvaise performance, nous décidons de réaliser un « adaboost » de « decision stump ». Le modèle est-il meilleur ? (taux de succès = 0.74). Pourquoi ?

Gradient Boosting

20. Reproduisez l'analyse en utilisant un GradientBoosting (**TUTO 4**, page 16 et suivantes). Fixez (n_estimators = 100) et (random_state = 1). Laissez les autres paramètres à leurs valeurs par défaut.
21. Lancez l'apprentissage (fit). Quelle est la profondeur max des arbres utilisés en interne ? (max_depth = 3) Quel est le taux d'apprentissage ? (learning_rate = 0.1).
22. Quelle est la performance en test du modèle ? (taux de succès = 0.795).
23. On se propose d'optimiser les paramètres de l'algorithme d'apprentissage (**TUTO 4**, page 18 ; attention, GridSearchCV est intégré dans un autre module maintenant https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html). Testez les jeux de paramètres suivants à l'aide d'une validation croisée à (cv = 5) folds :

```
parametres =
{'max_depth':[1,2,3,5,10],'subsample':[1.0,0.8,0.5],'max_features
':[None,'sqrt','log2']}
```

24. Quelle est la combinaison de paramètres qui optimise le taux de reconnaissance ? Avec quelle performance (estimée) ? (0.82).
25. Qu'en est-il réellement sur l'échantillon test ? (0.78)

Tout ça pour ça. Finalement, le premier Bagging était pas mal...