SY19 A19 TP7 Rapport

YUHUI WANG, XINGJIAN ZHOU et AGHILES HAMACHE

12 Janvier,2020

1. Partie Classification

1.1 Introduction

Dans ce problème de classification, nous devons classifier les 3 différents types d'objets astronomiques à partir de 17 variables. Pour cette partie nous avons fait les analyses ci-dessous :
- PCA
- KNN
- QDA
- Logistic Regression
- Naive Bayes Classifer
- Decision Tree, Bagging et Random Forest
- SVM avec noyaux différents
- Neural Network simple avec une seule couche cachée

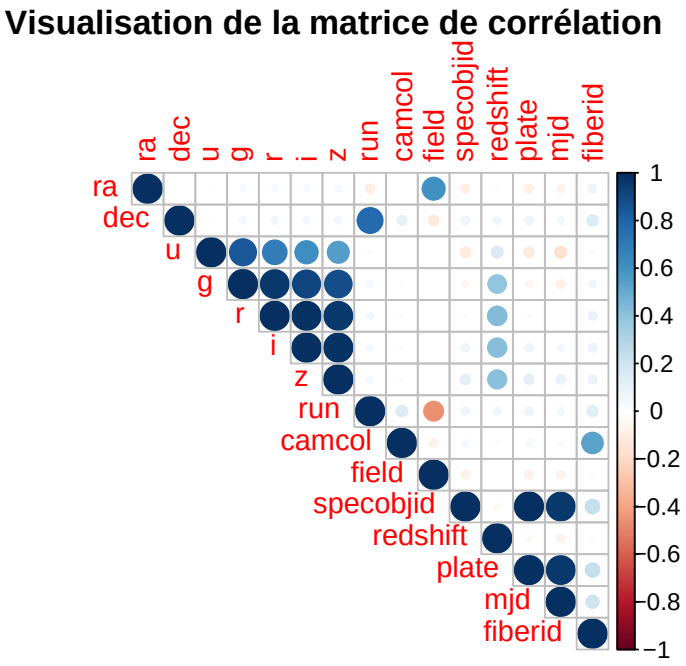
1.2 Préparation, traitement des données

Avant commencer, nous faisons le traitement des données pour que notre algorithmes puissent fonctionner mieux. Et pour les algorithmes basés sur des arbres, nous prenons les données d'orgine, en raison qu'ils n'ont pas besoin de traitement des données.

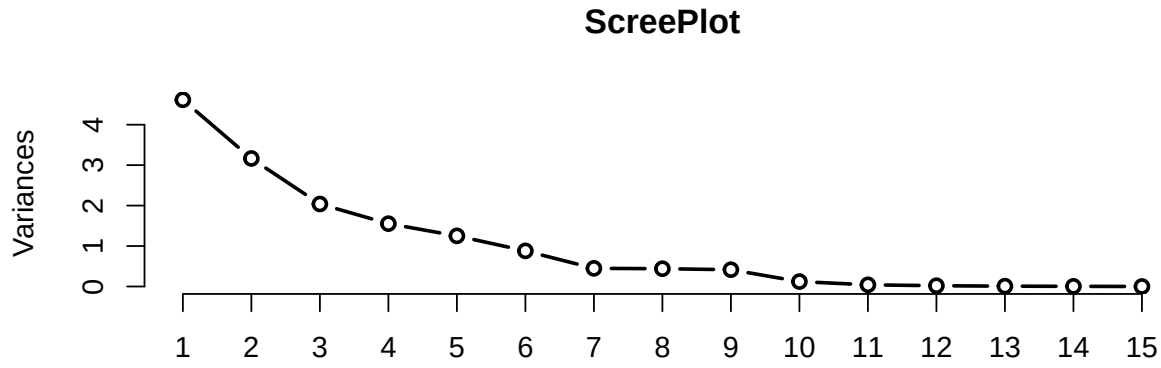
Tout d'abord, nous fasions l'analyse de données. Sachant que la colonne *rerun* et la colonne *objid* sont identiques pour tous les données, nous supprimons ces deux colonnes.

Deuxièmement, vue que la variable *specobjid* est trop grande au nivaeau de 10^{18}, et *redshift* est trop petite au niveau de 10^{-5}. Dans ce condition là, pour équilibrer l'influence des variables différentes, nous utilisons la fonction **scale()** pour normaliser tous les variables.

Troisièmement, nous analysons la dépendence entre les variables en calculant leur covariance. La fonction **corr.test()** de package `psych` est utilisée pour ganer une matrice de corrélation ainsi qu'une matrice des p-values. Et la fonction **corrplot()** du package `corrplot` sert à visualiser graphiquement la matrice de corrélation.



D'après la matrice des p-values et la figure des valeurs corrélation, nous pouvons voir clairement que deux groupes des valeurs *u,g,r,i,z* et *plate,mjd,specobjid* ont une relation linéaire entre eux. Donc nous pouvons prévoir que les méthodes pour réduire la dimension des features seront utiles pour les algorithms qui font hypothèses que les features n'ont pas de relation linéaire. Ainsi nous choisissons de faire l'analyse en composant principal pour l'extraction des features, en utilisant la fonction **prcomp()**.



Nous pouvons faire la conclusion de la figure de **ScreePlot** que nous pouvons choisir les 10 premiers composants principals, et la proportion cumulative est au 0.99521.

Pour pouvoir comparer plus facilement tous les algorithmes, nous utilisons la validation croisée de K-fold. Et après,nous faisons l'intervalle de confiance approximative au nivau 95% avec l'erreurs obtenues en utilisant la fonction **t.test()**.

1.3 Sélection de modèle

1.3.1 Logistic Regression

D'abord, nous faisons la méthode logistic regression, qui est un algorithme très simple pour le problème de classification, en utilisant la fonction **multinom()** de package `nnet` pour la condition avec 3 types.

```
## Erreur au moyen de logistic regression = 0.016     IC = [ 0.01306016 0.01893984 ]
```

1.3.2 QDA

Puis nous utilisons le modèle QDA par la fonction **qda()** dans le package `MASS`. On trouve que logistic regression fonctionne mieux que QDA.

```
## Erreur au moyen de QDA = 0.0232     IC = [ 0.01861576 0.02778424 ]
```

1.3.3 KNN

Pour la méthode de KNN, nous retiendrons la classe la plus représentée parmi les k sorties associées aux k entrées les proches de la nouvelle entrée x. Premièrement, nous appliquons la méthode de la validation croisée pour obtenir un meilleur k. Par la méthode CV, le taux d'erreur reste presque invariant pour k entre 1 et 500. Donc, on choisissons alors K=100.

```
## Erreur au moyen de knn = 0.1132     IC = [ 0.1015925 0.1248075 ]
```

1.3.4 Naive Bayes

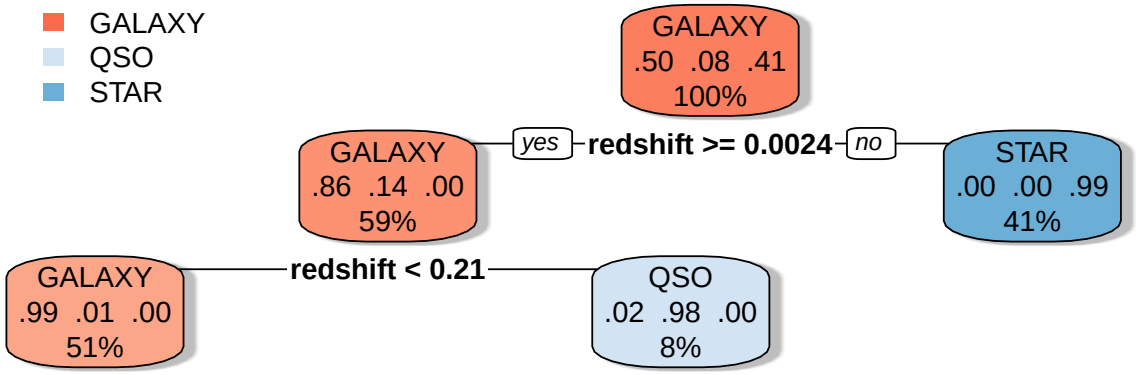
Pour cette méthode, nous faisons la modélisation en utilisant la fonction **naiveBayes()** dans le package `E1071`.

```
## Erreur au moyen de naive bayes = 0.1362     IC = [ 0.1241455 0.1482545 ]
```

1.3.5 Decision Tree, Bagging et Random Forest

Premièrement, nous construissons un arbre de décision sur les données sans traitement, et nous viualiser cette arbre.

```
## Erreur au moyen de decision tree = 0.0114
##     IC = [ 0.008305628 0.01449437 ]
```



Vue que notre modèle est déjà assez simple, nous n'avons pas besoin de appliquer à l'arbre précédent la procédure d'élagage. De plus le tableau de cp a montré que notre modèle est constitué avec une valeur de cp assez petite d'où cp = 0.001.

Alors nous appliquons le bagging et le random forest en utilisant la fonction **randomForest()** du package

`randomForest` d'où le paramètre *mtry = 15* signfie que tous les features sont prend en compte pour le cas bagging et *mtry = nombre de features/3* pour le cas random forest.

```
## Erreur au moyen de bagging = 0.011     IC = [ 0.007290553 0.01470945 ]
```

```
## Erreur au moyen de Random Forest = 0.0112     IC = [ 0.008485411 0.01391459 ]
```

1.3.6 SVM

Premièrement,nous implémintons la méthode SVM avec **ksvm()** dans le package `kernlab`. Nous utilisons une autre cross-validation pour trouver un meilleur cost paramètre C.Nous obtienons que le best_C = 100. Ensuite, nous essayons plusieur kenerls pour savoir s'il y a d'amélioration. ('vanilladot' pour le noyau linéaire, 'polydot' pour le noyau polinomial,'laplacedot' pour le noyau laplacien, et 'rbfdot' pour le noyau gaussien.)

```
##             lineaire polynomial  laplacien  gaussien
## mean error 0.0156000  0.0156000 0.03560000 0.0286000
## IC1        0.0128356  0.0128356 0.03056194 0.0252244
## IC2        0.0183644  0.0183644 0.04063806 0.0319756
```

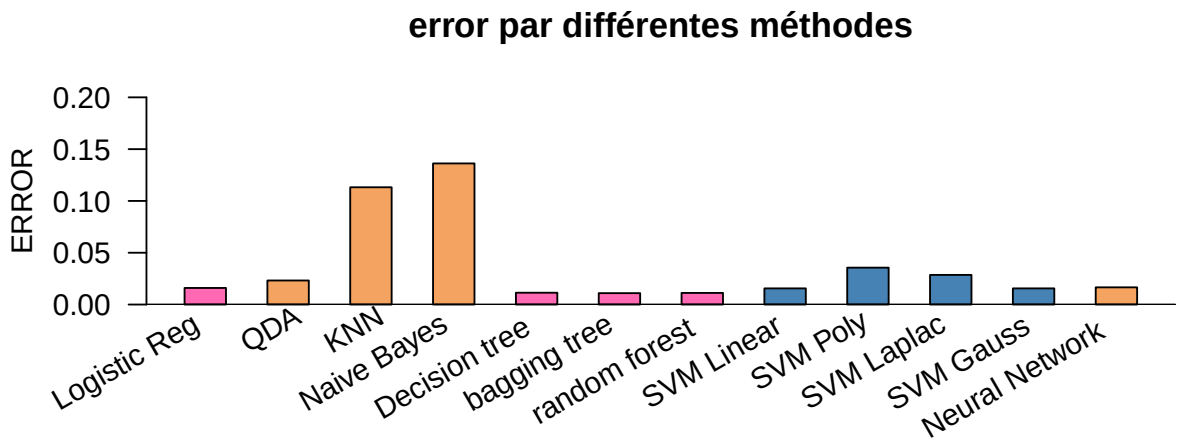
Ici nous povons voir que le noyau linéaire et polinomial ont le même taux d'erreur qui est mieux que les deux autres. Dans ce cas là, nous pouvons faire l'hypothèse que nos données ont des bornes linéaires pour la classification. Et la comparaison entre *KNN*, *Logistic Regression* et *QDA* donne la même conclusion.

1.3.7 Neural Network

Pour le neural network, nous chosissons le moddèle simple avec une seule couche cachée qui contient 15 noeuds dedans. Et nous le réalisons avec **nnet()** dans le package `nnet`.

```
## Erreur au moyen de neural network = 0.0166     IC = [ 0.01358002 0.01961998 ]
```

1.4 Conclusion



D'après tous les algorithmes que nous avons utilisés, en comparant leur erreur moyenne, nous pouvons faire l'hypothèse que nos données ont des bornes linéaires pour la classification. Et parmi tous les modèle que nous avons fait, *Decision Tree* et *Random Forest* ont la meilleure résultat. Et entre les deux, nous avons choisi *Random Forest* en raison que il peut éviter le problème de 'overfit' de *Decision Tree*. Donc nous le choisissons comme le modèle final et faire un entraînement sur tous les données pour obtenir le modèle final.