Examen Mi-parcours : Fouille de données

Komi Agblodoe – Thomas Vrignaud

20 Janvier 2020

Exercice 1 :

On considère qu'on dispose d'un jeu de données (X,y) lié à une problématique de classification binaire et qu'on a créé un objet GridSearchCV pour optimiser les paramètres d'un classifieur donné.

1. Expliquer la différence entre les deux portions de code ci-dessous :

– code #1 :

*> grid_search.fit(X, y) > acc = grid_search.best_score_*

– code #2 :

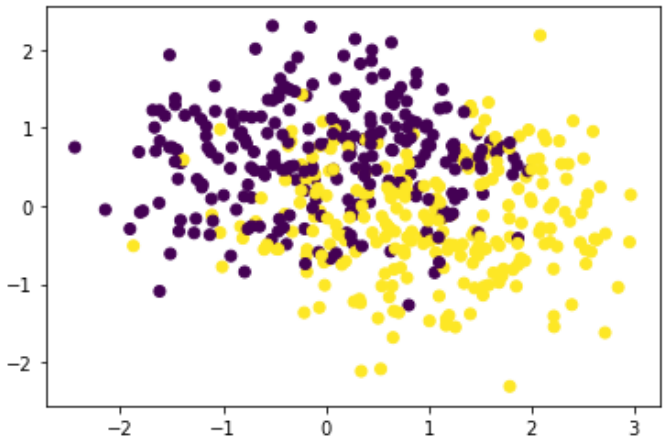
> cv_res = cross_val_score(grid_search, X, y, cv = 10) > acc = cv_res.mean()

Ces deux portions de code ont quelques différences. Dans le 1er code, la fonction *grid_search()* est utilisée seule pour évaluer la performance d'un modèle par validation croisée. Cette fonction va tester toutes les configurations possibles des hyperparamètres et renvoie le meilleur d'entre eux. Le modèle est appris sur l'ensemble du jeu d'apprentissage. La première ligne va effectuer cela ensuite, la deuxième ligne va nous le score pour ce meilleur modèle.

Dans le 2ème code, la fonction *cross_val_score()* est utilisée. Elle permet de calculer le score dans un nombre *k* de folds. Dans notre cas nous utilisons 10 folds. A l'intérieur de cette fonction nous retrouvons la première ligne du code 1, qui nous renvoie la meilleure configuration du modèle. La sortie de cette première ligne sera un vecteur de taille 10. En effet, pour chaque fold le score sera calculé avec le modèle déterminé précédemment. Ensuite, la dernière ligne effectue la moyenne de tous ces scores. Avec cette deuxième méthode une cross-validation est mise en place, ce qui permet d'évaluer les performances du modèle en comparant les valeurs réelles et prédites. Elle permet également de construire et de prédire sur des jeux de données distincts.

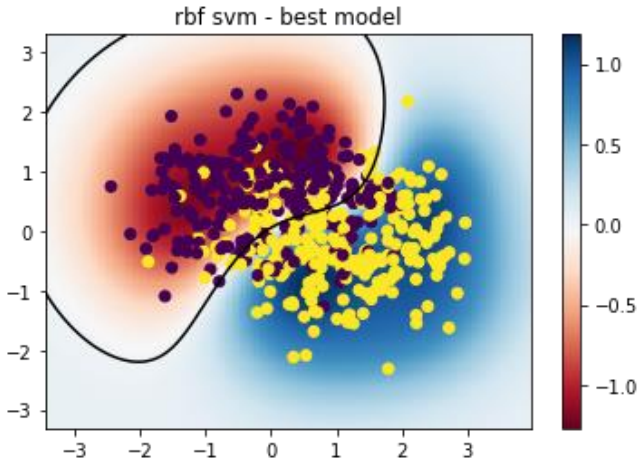
Pour conclure, le même modèle est utilisé dans les deux codes. La différence se situent dans le fait que dans le code 2, une cross validation est mise en place alors que se n'est pas le cas dans le code 1. Dans le code 1, nous pouvons avoir un risque plus grand de sur-apprentissage.

2. Implémenter une procédure basée sur un classifier SVM à noyau RBF et le jeu de données "moons" pour illustrer cette différence. Vous générez le jeu de données grâce à la commande ci-dessous :
> X, y = make_moons(500, noise = 0.6, random_state = 27)



Nous pouvons la représentation de notre jeu de données avec les couleurs associés au groupe auquel ils appartiennent. Nous remarquons qu'une légère tendance semble se démarquer.

Une procédure basée sur un classifier SVM a ensuite été mise en place. Les hyperparamètres ont été choisi avec la fonction *gridsearch().* Vous pouvez voir ci-dessous la forme du modèle.



Enfin, les deux codes ont été mis en place afin de déterminer leurs différences.

La sortie du code N°1 a été le score calculé sur l'ensemble du jeu de données, avec un risque de sur-apprentissage.

```
0.796
```

Pour la sortir du code N°2, nous avons les scores pour les 10 folds. La seconde est la moyenne du score des 10 folds.

```
[0.82 0.78 0.8 0.78 0.72 0.82 0.78 0.88 0.74 0.78]
0.79
```

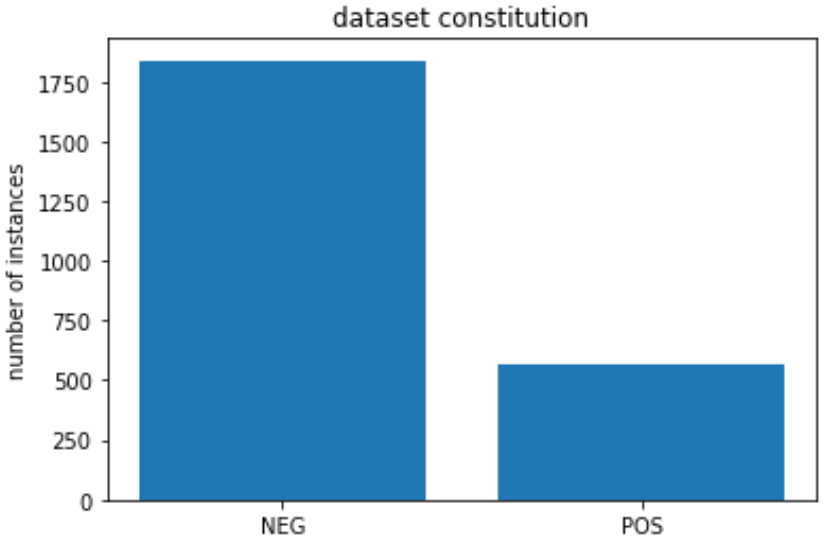
Nous voyons que les deux scores sont tout de même très proches.

Exercice 2 :

1. Charger le jeu de données et fournir quelques éléments d'analyse exploratoire (e.g., statistiques descriptives, analyse ACP) illustrant votre prise en main du jeu de données.

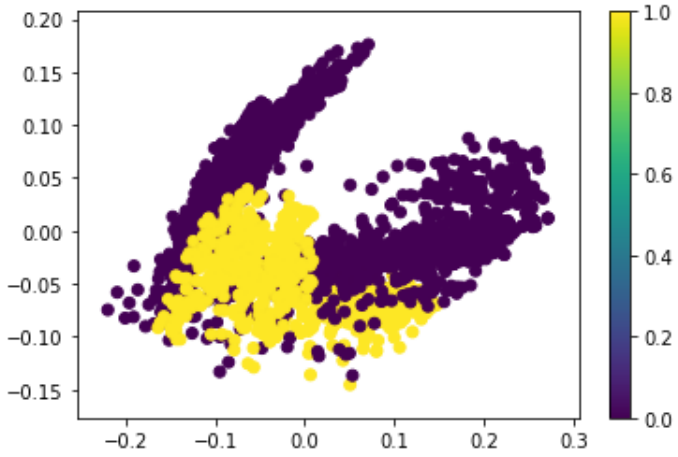
Afin de mieux connaître notre jeu de données, quelques analyses exploratoires vont être mise en place. Notre jeu de données d'entraînement est composé de 240 variables explicatives et de 2403 individus. Nous avons également le Gram de la bactérie pour chacun de ces patients qui est négative ou positive. Concernant le jeu de test, nous avons les mêmes 240 variables explicatives pour 3130 autres patients. Pour le jeu de test, nous ne connaissons pas le Gram de la bactérie.

Ensuite, nous allons voulu connaître la proportion de NEG et de POS dans le jeu d'entraînement.



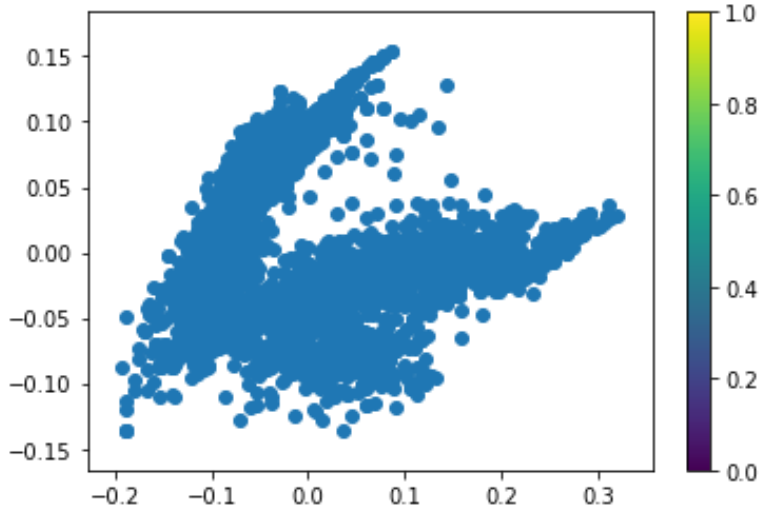
Nous voyons que nous avons beaucoup d'individus avec un gram négative et moins avec un gram positif. Il sera important de l'avoir toujours en tête pour la suite des analyses. Une ACP pour le jeu d'entraînement a été mise en place. La proportion est de 77% de négatif contre 33% de positif.

Nous voyons également que les covariables se ressemblent. Le minimum est autour de 0.015 alors que le maximum est d'environ 0.08. Le tableau des corrélations nous montre que certaines variables sont très fortement corrélées entre elles.



Les points sont représentés en fonction de leur coordonnée obtenue sur les deux premières dimensions de l'ACP. Les couleurs représentent le signe du Gram de la bactérie. Nous pouvons remarquer une légère différence entre les deux groupes. Le violet est pour le Gram négatif et le jaune pour le gram positif.

Pour terminer, une ACP a également été mise en place sur le jeu de test permettant d'observer l'allure de nos données.



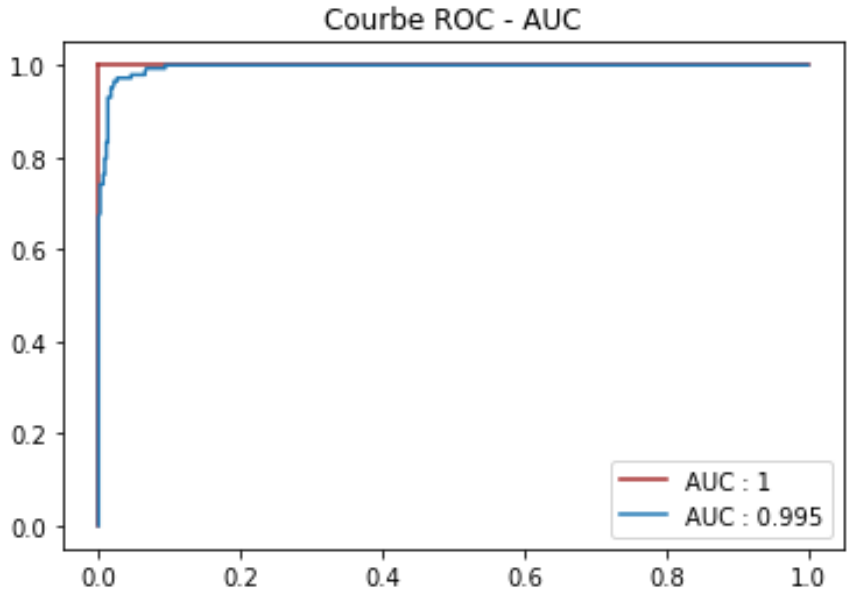
L'abscisse et l'ordonnée sont les mêmes qu'auparavant. Nous n'avons pas de code couleur puisque naturellement nous ne connaissons pas le signe du Gram de la bactérie. Nous pouvons remarquer tout de même que les données ont la même allure que celles du jeu d'entrainement. Ceci va peut-être nous aider à trouver un modèle de qualité.

2. Construire un modèle de prédiction visant à maximiser l'aire sous la courbe ROC.

On va utiliser la méthode de random forest pour ce modèle de prédiction. Cette méthode est basée sur les bagging d'arbres. Nous avons limité à 1000 arbres pour des raisons calculatoires. Les autres paramètres sont sélectionnés par défaut comme le critère de gini ou l'entropie.

```
array([[0.726, 0.274],
       [0.999, 0.001],
       [0.999, 0.001],
       [1.   , 0.   ],
       [0.948, 0.052],
       [0.974, 0.026]])
```

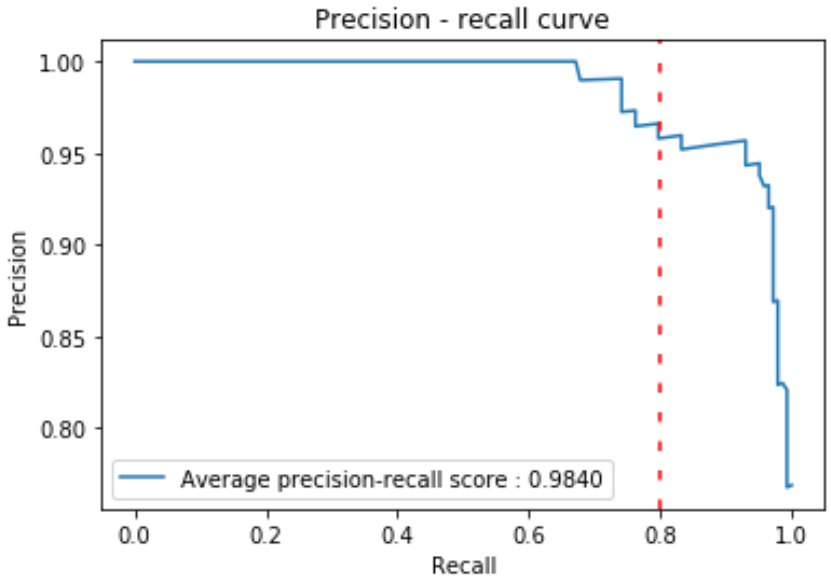
Ci-dessus, un extrait des prédictions fourni par le random forest. Nous voyons que les probabilités sont assez écartées et proches de 0 ou de 1. Il va donc être plus facile de choisir entre un Gram négatif ou positif. Avec ce modèle, nous avons obtenu une accuracy de 97%, ce qui est extrêmement élevé.



Nous voyons également que la courbe ROC est très bon avec une AUC de 0.995, qui est donc très proche de 1.

3. Représenter sur une même figure l'évolution de la spécificité et la précision de votre meilleur modèle quand on fait varier sa sensibilité. Quelles valeurs de spécificité et de précision obtient-on pour une sensibilité de 80% ? Qu'est ce qui explique cette différence ?

Dans un second temps, nous avons évaluer la spécificité et la précision. Ces deux indicateurs sont importants car ils permettent d'avoir une analyse plus fine et précise de la qualité de notre modèle.



Nous pouvons voir avec le graphique ci-dessous que ‘aire sous la courbe est très élevée. De plus, lorsque la sensibilité est de 80%, la précision est de 98.4%.

4. Calculer enfin les prédictions obtenues sur le jeu de test et les enregistrer dans un fichier texte nommé votre-nom_exo-2.txt. Notez que leur qualité sera évaluée en termes d’aire sous la courbe ROC.

Les prédictions calculées ont été enregistrées dans un fichier txt pour pouvoir ensuite les comparer avec les vraies valeurs. Nous avons une proportion de prédiction négative de 76% et 24% de positive.