RAPPORT DU PROJET MACHINE LEARNING

Réalisation d'une interface d'analyse de données par apprentissage supervisé

Yanis Benreguig
Chloé Carrié
Léa Regazzetti

M2 SISE
Promotion 2021-2022
Machine Learning sous Python

Introduction

Dans le cadre de notre Master 2 SISE, nous avons eu comme projet, en Machine Learning, de créer une interface graphique permettant à un utilisateur d'appliquer des algorithmes de prédiction sur des jeux de données au choix. Cette interface lui permet de choisir un jeu de données, la variable cible qu'il a envie de prédire et les variables explicatives. L'utilisateur n'a pas besoin d'avoir des connaissances en machine learning, ni en programmation Python.

Les principaux objectifs de notre application sont d'appliquer les bons algorithmes en fonction des caractéristiques de la variable cible sélectionnée et de fournir les bons résultats et sorties afin de permettre à l'utilisateur d'évaluer les performances du modèle. Cette application doit être programmée dans le langage Python. Nous utilisons de nombreuses librairies comme Scikit Learn principalement pour la programmation des algorithmes de classification et de régression, Pandas pour la gestion et la manipulation des données, Numpy pour la manipulation de numériques, Plotly pour les graphiques et Dash pour la création de l'interface.

Comme nous sommes dans le cas de l'apprentissage supervisé, c'est-à-dire que les données sont étiquetées, nous pouvons évaluer les performances de nos modèles. Selon les caractéristiques de la variable cible, des algorithmes de classification ou de régression sont proposés. L'évaluation des modèles est différente si on est dans le cas de la classification ou de la régression mais elle permet de savoir si notre modèle est performant et prédit correctement notre variable. Selon les cas, certains algorithmes ne seront pas adaptés.

Dans une première partie, nous allons décrire l'architecture de l'application et nous allons vous présenter l'utilisation de notre application en seconde partie.

I. Architecture de l'application

Pour illustrer notre application, nous avons une interface graphique qui permet à l'utilisateur d'appliquer des algorithmes de prédiction sur des jeux de données. Deux aspects techniques seront décrits dans cette partie : l'interface graphique qui permet l'interaction avec l'utilisateur et la programmation des algorithmes de prédiction qui se déclenche derrière l'interface. Nous allons commencer par expliquer les algorithmes de prédiction dans une première partie puis l'interface graphique dans une deuxième partie.

1) Algorithmes de prédiction

Concernant les algorithmes, nous avons eu besoin de programmer des algorithmes de classification et des algorithmes de régression selon la variable cible choisie par l'utilisateur. En effet, s'il choisit une variable cible qualitative, seuls les algorithmes de classification lui sont proposés et seuls des algorithmes de régression s'il choisit une variable quantitative. Pour gérer ces deux cas de façon indépendante, nous avons décidé de séparer les algorithmes en deux fichiers python : un pour la classification et un pour la régression.
Pour les algorithmes de classification, nous avons choisi de programmer un modèle de régression logistique, d'arbre de décision et d'analyse discriminante et pour les algorithmes de régression, un modèle d'arbre de régression, de k plus proches voisins et de régression linéaire. Les algorithmes de classification sont regroupés dans le fichier "**Classification.py**" et ceux de régression dans le fichier "**Regression.py.**".

A. Algorithmes de classification

Pour chaque algorithme de classification, la démarche est la même :

- On recode les variables explicatives qualitatives en effectuant un codage disjonctif. Seules k-1 modalités sont obtenues sur les k modalités d'une variable qualitative afin d'éviter les problèmes de multi-colinéarité.
- On scinde les données en deux échantillons : un d'apprentissage et un de test : "**train_test_split**". Nous avons choisi 70% de données pour l'apprentissage et 30% pour le test.
- On centre et réduit les variables explicatives quantitatives X afin d'obtenir la même unité pour chaque variable : "**StandardScaler()**". On applique le centrage-réduction sur l'échantillon d'apprentissage puis nous appliquons exactement les mêmes paramètres de l'échantillon d'apprentissage (moyenne et variance) pour centrer réduire l'échantillon de test.
- Nous stockons l'heure de début d'exécution de notre algorithme : "**start**".

- On instancie notre modèle, on recherche les paramètres optimaux et on applique notre modèle avec les paramètres optimaux sur nos données d'apprentissage : "**fit**" Nous avons choisi de ne pas tester tous les paramètres pour chaque modèle mais uniquement ceux qui nous paraissaient les plus importants.
- On réalise une validation croisée sur les données de base X et Y avec l'accuracy comme méthode de scoring : "**cross_val_score**". Nous faisons la moyenne des réalisations pour obtenir la performance globale en moyenne de notre modèle.
- Après toutes ces étapes, nous pouvons réaliser une prédiction du modèle sur les données de test : "**predict**".

Les sorties de nos algorithmes de classification sont les mêmes pour tous les modèles. Comme nous sommes dans le cas d'algorithmes de classification supervisés, les sorties consistent à comparer les données réelles et les données prédites par nos modèles. Tout d'abord, nous affichons la matrice de confusion qui permet de voir les individus bien classés par notre modèle et ceux qui sont mal classés de façon intuitive.
Puis, nous affichons le rappel et la précision de notre modèle. Le rappel (ou la sensibilité) est le taux de vrais positifs, c'est-à-dire la proportion de positifs que l'on a correctement identifiés. La précision est la proportion de prédictions correctes parmi les points que l'on a prédits positifs. Ces deux mesures nous permettent d'évaluer les performances de nos modèles.
Enfin, nous affichons aussi des sorties graphiques qui permettent de percevoir de façon synthétique la différence entre les prédictions et la réalité. La courbe ROC et l'aire sous cette courbe sont calculées et affichées. La courbe ROC représente la sensibilité en fonction de 1 - la spécificité pour toutes les valeurs seuils possibles du marqueur étudié. La spécificité peut être définie comme étant le taux de vrais négatifs donc nous représentons le taux de vrais positifs en fonction de 1 - le taux de vrais négatifs dans une courbe ROC. L'aire sous cette courbe (AUC) indique la probabilité pour que notre modèle place un positif devant un négatif. Dans le meilleur des cas l'aire vaut 1, si le modèle classe au hasard les individus l'aire sous la courbe sera de 0.5, ce qui signifie en quelque sorte que le modèle n'est pas d'une grande utilité.

De plus, c'est à ce moment là, à la fin de notre programme, que nous arrêtons notre compteur de temps et que nous faisons la différence avec le temps de départ "start" et l'heure de fin "end". Le temps d'exécution de notre programme est donc calculé.

Nos trois fonctions d'algorithmes de classification retournent les mêmes objets : le taux de reconnaissance moyen d'ajustement de nos modèles, les meilleurs paramètres en validation croisée, la matrice de confusion, le rappel, la précision de notre modèle, le temps d'exécution de notre programme, la courbe ROC et son aire sous la courbe.

B. Algorithmes de régression

La démarche pour les algorithmes de régression est la même que pour les modèles de classification. Tout d'abord, nous avons transformé les données comme décrit précédemment en utilisant le codage disjonctif pour les variables qualitatives dans les variables explicatives X et en centrant-réduisant les variables quantitatives du X.
Puis, nous avons scindé en deux échantillons les données : apprentissage et test. Nous stockons l'heure de début de notre programme "start". Nous instancions notre modèle, nous recherchons les paramètres optimaux et nous appliquons notre modèle avec les paramètres optimaux sur nos données d'apprentissage. Nous réalisons une validation croisée afin de connaître les performances moyennes de nos modèles mais ici, pour la régression, nous prenons le coefficient de détermination R2 comme scoring. Le R2 est la mesure de qualité de la prédiction d'une régression, c'est-à-dire, que c'est le rapport de la variance expliquée par la régression sur la variance totale. Enfin, nous pouvons appliquer notre modèle sur les données de test en réalisant une prédiction.

Nous pouvons voir que les étapes sont relativement les mêmes que celles décrites dans la classification mais concernant les métriques, il y a quelques spécificités pour la régression.
En effet, nous ne pouvons pas obtenir de matrice de confusion, rappel et précision dans le cas d'une régression puisque la variable cible est quantitative. Nous avons donc fait le choix de calculer l'erreur quadratique moyenne (MSE - Mean Squared Error) de nos modèles. La MSE mesure la moyenne des carrés des erreurs, c'est-à-dire la différence quadratique moyenne entre les valeurs estimées et les valeurs réelles. Le but est que le MSE soit le plus faible possible puisqu'il représente l'erreur. De plus, une courbe ROC et une aire sous cette courbe ne sont donc pas adaptées ici. Nous avons donc affiché un graphique qui représente les données prédites sous forme de nuage de point avec une courbe qui représente les données réelles. Le but est que le nuage de points s'ajuste au mieux aux données réelles.

Les sorties des modèles de régression sont donc le temps d'exécution de notre programme, le coefficient de détermination moyen de notre modèle, les meilleurs paramètres de notre modèle, le MSE et le graphique qui compare les estimations et la réalité.

2) Interface graphique

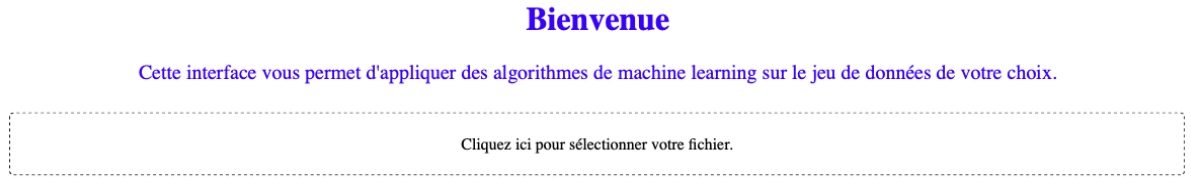
En ce qui concerne l'interface graphique, nous avons eu recours à la bibliothèque Dash pour l'interaction avec l'utilisateur, et à la bibliothèque Plotly pour les graphiques.

En premier lieu, une première page d'accueil est présentée à l'utilisateur afin qu'il puisse comprendre et prendre connaissance de notre application.
Au fur et à mesure des actions de l'utilisateur, différentes propositions lui seront faites, afin notamment qu'il puisse charger le jeu de données de son choix, définir la variable cible à expliquer et les variables explicatives, sélectionner le nombre de folds qu'il souhaite utiliser pour la validation croisée, choisir l'algorithme d'apprentissage supervisé et enfin visualiser les résultats obtenus. Pour mieux comprendre le fonctionnement de l'interface et son utilisation nous avons réalisé un guide d'utilisation.

II. Mini-guide d'utilisation

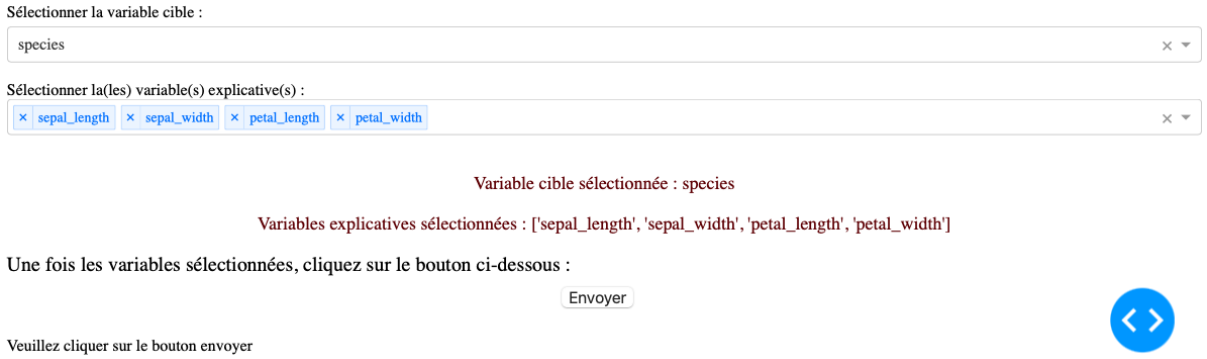
Nous allons ici vous présenter l'utilisation de notre application.
Étant donné que l'application n'est pas déployée, il faut passer par le terminal pour la lancer. De ce fait, il s'agit de définir le répertoire dans lequel se trouve les 3 modules (Interface_Accueil.py, Regression.py, Classification.py) nécessaires à l'utilisation de notre projet. Puis l'exécution passe par le module Interface_Accueil.py.

Sur la page d'accueil, l'utilisateur peut sélectionner le fichier de données qu'il souhaite utiliser. A noter que le fichier doit être dans le format CSV, avec une virgule comme séparateur. Egalement, il ne doit pas présenter de valeurs manquantes. Par ailleurs, les noms de colonnes seront présents sur la première ligne du fichier. Les trois points précédents ayant été définis dans le cahier des charges du projet.

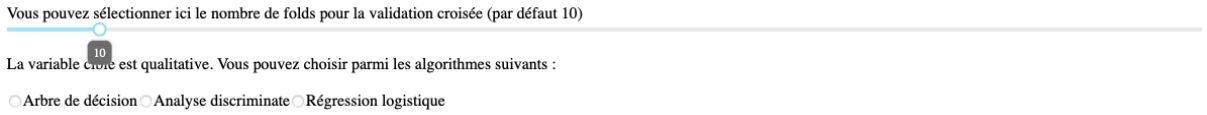


Une fois le fichier sélectionné, un aperçu des 10 premières lignes est présenté à l'utilisateur. Cela lui permet de vérifier que le jeu de données a été bien importé notamment. A la suite de cet affichage, deux composants sont proposés. Le premier est une liste déroulante permettant de sélectionner la variable à prédire. Le deuxième est une liste déroulante à valeurs multiples pour définir les variables explicatives.

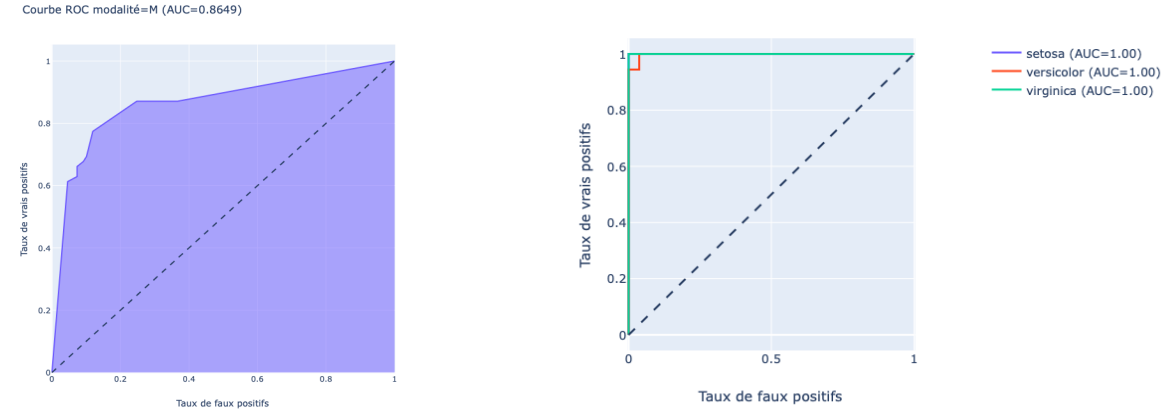
Dès qu'une sélection est effectuée sur l'un des deux composants, un affichage permet de visualiser quelle est la variable cible sélectionnée et quelles sont les variables explicatives sélectionnées. Il est à ce stade possible de changer les valeurs sélectionnées dans les deux composants. Une fois que le choix est fixé, un bouton envoyer doit être pressé.



Selon le type de la variable cible sélectionnée, l'application propose différents algorithmes de classification ou de régression.
Dans le cas d'une variable cible qualitative, il est proposé un arbre de décision, une analyse discriminante ou une régression logistique. Avant cela, un curseur permet à l'utilisateur de choisir le nombre de "blocs" (folds) qu'il souhaite utiliser pour la validation croisée. Par défaut, il est fixé à 10.



Une fois l'algorithme sélectionné, celui-ci s'exécute puis affiche les résultats.
Tout d'abord est présentée la courbe ROC ainsi que la valeur de l'aire sous la courbe. On distingue ici deux cas pour les résultats. Dans le cas où la variable cible ne présente que deux modalités, l'affichage ne se fait que pour une seule des deux modalités, qui est précisée dans le titre. Lorsqu'il y a plus de 2 modalités, l'affichage est réalisé pour chacune des modalités, dans le même graphique.



A la suite, on retrouve un rappel du nombre de folds sélectionnés pour la validation croisée, puis le temps de calcul de l'algorithme. Ensuite, sont affichés les meilleurs paramètres estimés, puis l'accuracy en validation croisée, la précision, le rappel, et enfin la matrice de confusion. Pour le calcul du rappel et de la précision, on distingue à nouveau les deux cas. Si la variable cible présente plus de deux modalités, il s'agit de la moyenne pondérée de la métrique, dans le cas d'une variable cible avec deux modalités, il s'agit de la métrique pour la même modalité que celle de la courbe ROC.

Nombre de folds sélectionnés : 10

Temps de calcul (en secondes) : 15.2

Meilleurs paramètres estimés en fonction de l'accuracy : {'criterion': 'entropy', 'max_depth': 10, 'min_samples_split': 5, 'splitter': 'random'}

Pourcentage de bonnes prédictions en cross-validation (accuracy) : 0.889

Precision : 0.808

Rappel : 0.677

Matrice de confusion :

0	1
99	10
20	42

Si l'on souhaite tester un autre algorithme ou changer le nombre de folds, il suffit de cliquer sur le bouton envoyer, ce qui va réinitialiser les paramètres et permettre de les changer puis de relancer l'exécution de l'algorithme nouvellement choisi.

Dans le cas d'une variable cible quantitative, il est proposé une régression linéaire, un k plus proches voisins ou un arbre de régression.
De la même manière que pour une variable cible qualitative, un curseur permet de sélectionner le nombre de folds.

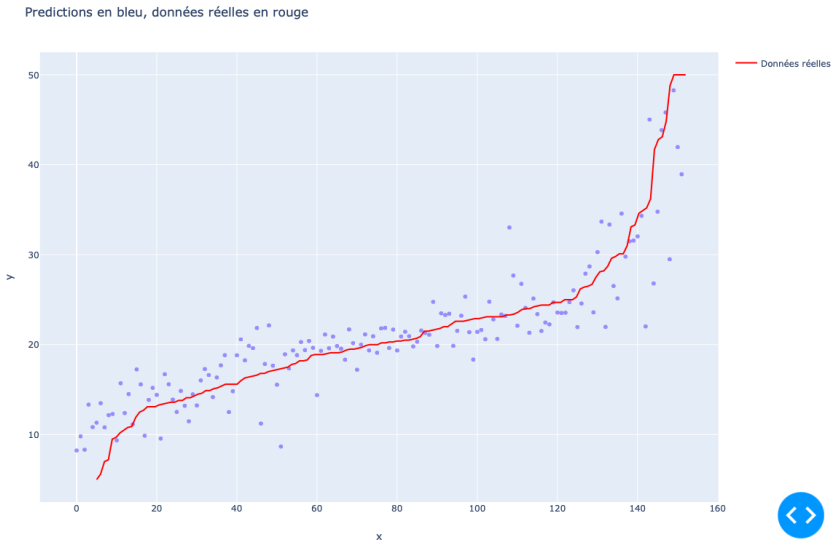
Vous pouvez sélectionner ici le nombre de folds pour la validation croisée (par défaut 10)

15

La variable cible est quantitative. Vous pouvez choisir parmi les algorithmes suivants :

○ Régression linéaire ◉ K plus proches voisins ○ Arbre de régression

En sortie, on trouve un graphique permettant de comparer les prédictions et les données réelles.



Puis on retrouve à nouveau le nombre de folds sélectionnés, le temps d'exécution de l'algorithme et les meilleurs paramètres estimés. Enfin, le coefficient de détermination calculé en validation croisée est affiché, suivi de l'erreur quadratique moyenne, permettant d'évaluer la qualité du modèle.

```
Nombre de folds sélectionnés : 15

Temps de calcul (en secondes) : 4.8

Meilleurs paramètres estimés en fonction du r2 : {'n_neighbors': 4, 'weights': 'distance'}

Coefficient de détermination en cross-validation : -1.524

Erreur quadratique moyenne : 13.973
```

III. Conclusion et perspectives d'évolutions

En conclusion, lors de ce projet, nous avons mis en place une interface dans laquelle l'utilisateur a la possibilité de choisir la variable cible sur laquelle il souhaite réaliser la prédiction ainsi que les variables explicatives servant à prédire le modèle. En fonction de la variable cible, l'interface va lui proposer l'algorithme qu'il souhaite exécuter.
Ce projet nous a apporté une certaine maîtrise de Dash et dans la construction d'une interface, bien que cela n'était pas aisé au début, surtout pour l'intégration des graphiques. Nous avons également eu l'occasion d'en apprendre plus sur les différents algorithmes que nous avons choisis.

Afin d'améliorer notre application, il pourrait être intéressant d'y intégrer une prise en charge d'autres formats de jeux de données, mais également de permettre à l'utilisateur d'interagir avec le choix des paramètres optimaux des algorithmes. De plus, une mise à jour automatique des résultats dès le changement d'un choix serait envisageable ou encore de déployer l'application Dash en API. Nous pourrions également, lorsque la variable cible est choisie, la retirer des variables explicatives proposées afin d'éviter des mauvais résultats, ou encore gérer les valeurs manquantes, comme remplacer celles de variables qualitatives par le mode et celles de variables quantitatives par la moyenne ou la médiane.