Projet Python

Université Lumière
Lyon2
M2 SISE

DECOSTER Rémi
IBORRA Alexandre
LAINE Pauline

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

bokeh

python™

Année universitaire : 2020 - 2021

Dans le cadre de notre cours de Machine Learning avec Python, nous souhaitons réaliser une interface d'analyse de données avec des méthodes d'apprentissage supervisé.

Cette application devra être facile d'utilisation pour les utilisateurs qui ne connaissent pas la programmation Python.

L'objectif de cette application est de permettre à un utilisateur de réaliser des analyses sur ses données en choisissant la variable qu'il souhaite prédire et les variables prédictives du modèle. Il aura ensuite le choix entre trois algorithmes de prédiction en fonction du type de la variable à prédire : soit une régression, soit une classification.

L'utilisateur pourra accéder au résultat de son modèle avec un graphique présentant les prédictions ainsi que les métriques d'évaluation et le temps de calcul du modèle. Ce qui lui permettra ainsi d'évaluer la qualité de son modèle.

Cette application a été réalisé en Python avec l'outil Bokeh.

Dans la suite de ce rapport, nous verrons comment nous nous sommes organisés pour réaliser cette application. Puis nous vous présenterons l'architecture de notre application. Pour finir vous trouverez un guide d'utilisation avec des exemples.

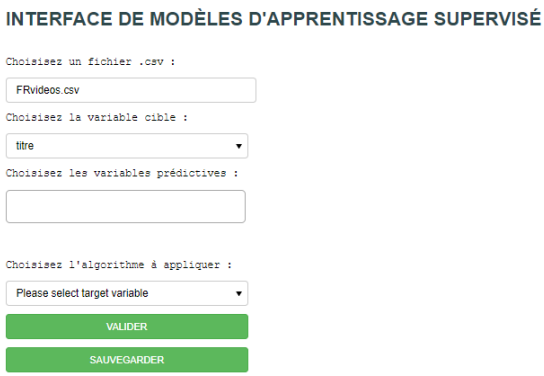
Pour réaliser ce projet nous avons organisé notre travail. Pour commencer nous avons lu le sujet et discuté sur celui-ci afin de bien comprendre les attentes. Nous avons mis en commun ce que nous voulions faire.

Nous avons ainsi réalisé une architecture de l'application sur papier, défini les algorithmes que nous voulions intégrer et nous nous sommes mis d'accord sur l'outil que nous souhaitions utiliser ainsi que sur un jeu de données commun pour faire nos tests. Nous avons donc choisi de réaliser notre application avec l'outil Bokeh et sur le jeu de données FRvideos.csv.

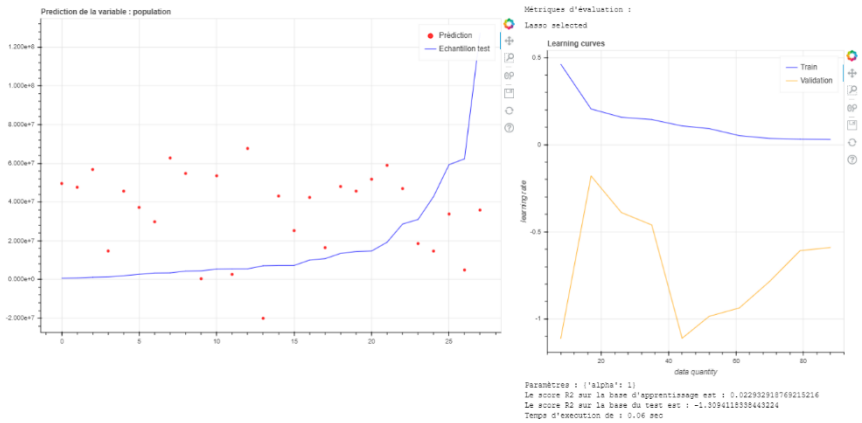
Une fois les étapes de structuration du sujet établies, nous pouvions chacun travailler sur une partie. Un travail que nous mettions en commun régulièrement lors de nos réunions. Cela nous permettait d'échanger sur nos problèmes, sur les améliorations à apporter, et partager les nouvelles tâches.

Pour ce projet nous avons voulu que la répartition soit équitable et que tout le monde puisse toucher à tout. C'est-à-dire que nous avons tous participé à la réalisation du corps de l'application, chacun apportant son expertise, ses améliorations et nous avons réalisé deux algorithmes d'apprentissage supervisé chacun.

Notre application se divise en deux grandes parties. Une partie paramétrage du modèle et une partie avec les résultats. On retrouve dans la structure de notre code les importations, les fonctions et l'affichage visuel des éléments de l'application. De plus un fichier HTML est associé au code de notre application. Ce fichier permet de définir le style de l'application. Ici, il permet de donner un titre à notre application.



Partie paramétrage



Partie résultats

Lorsque nous ouvrons l'application, seule la partie paramétrage est présente. Une zone de texte permet à l'utilisateur de choisir le fichier de données qu'il souhaite importer pour son modèle. Par défaut le jeu de données exporté automatiquement est un fichier sur les vidéos YouTube de France. Le fichier doit être au format CSV, avoir pour séparateur une virgule et la première ligne doit contenir les noms des variables. Derrière cet élément on trouve une fonction (*importation()*) qui va importer le fichier saisi et mettre à jour les deux menus suivants.

On trouve ensuite un menu déroulant qui permet de sélectionner la variable à prédire (variable cible) et un menu permettant de sélectionner une ou plusieurs variables prédictives (features). Ils sont initialiser avec le nom des variables et automatiquement mis à jour lors d'un changement de fichier.

Dans le dernier menu déroulant, l'utilisateur a le choix de l'algorithme de prédiction qu'il souhaite appliquer à son modèle. Il aura le choix entre trois algorithmes. En fonction du type de la variable cible, ce dernier menu change grâce à une fonction (*callback()*) appelée dès lors qu'il y a un changement. Lorsque la variable cible est quantitative, nous avons alors une régression, les trois algorithmes qui seront proposés seront alors : La régression Lasso, la régression Ridge ou la descente de gradient.

Si la variable cible est qualitative, nous sommes dans le cas d'une classification, les trois algorithmes seront alors : l'arbre de décision, les k plus proche voisins et la descente de gradients.

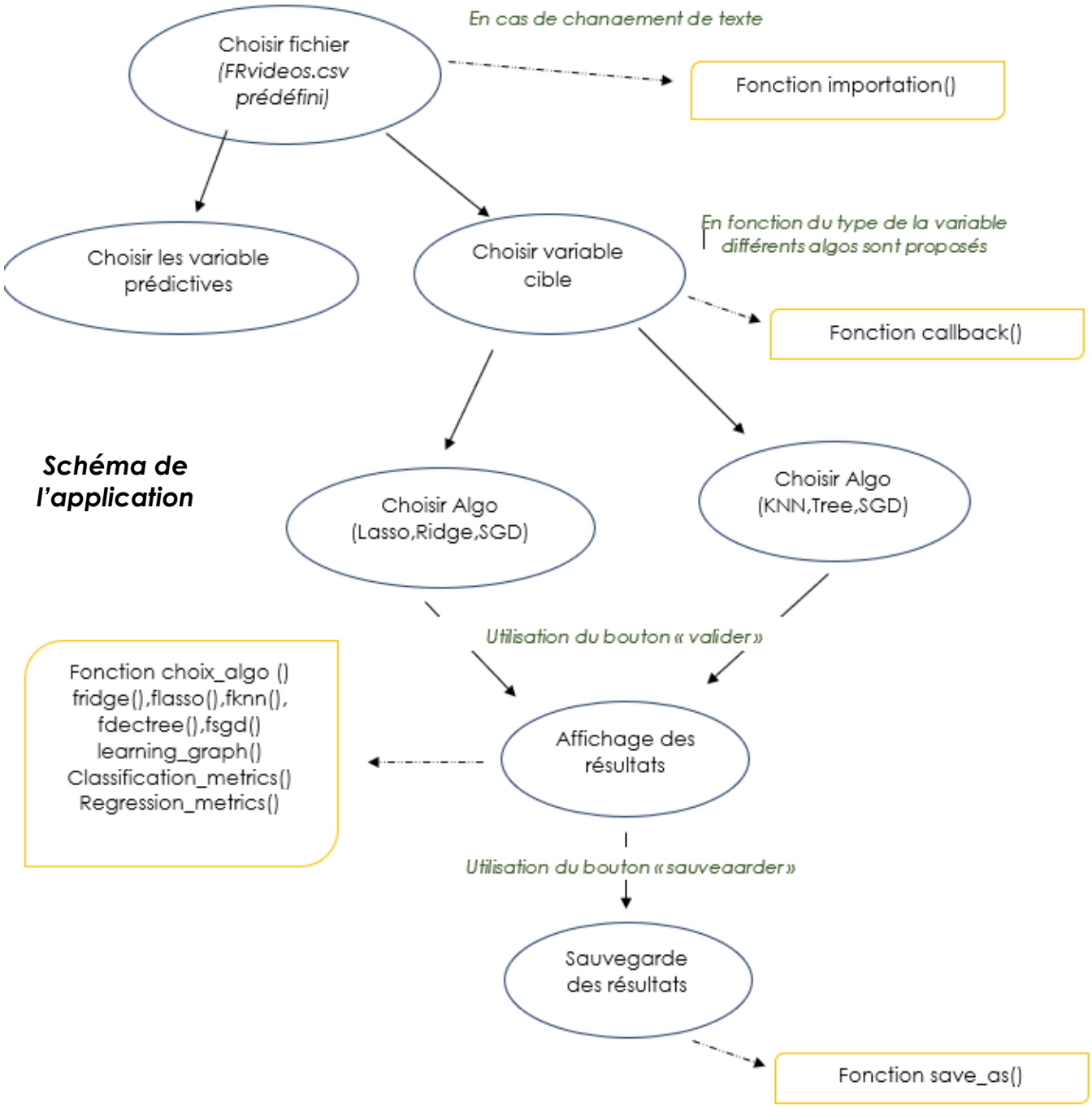
Nous avons voulu choisir des algorithmes que nous connaissions afin d'en comprendre les comportements lors de nos tests. Dans le cas d'un problème de régression comme pour celui de classification, nous proposons une méthode de descente de gradient. Ces méthodes sont particulièrement utiles lorsque le jeu de données contient beaucoup d'échantillons (i.e. > 10^6 d'après la documentation de scikit-learn) et permettent d'obtenir une réponse en un temps raisonnable.

Pour la régression, nous avons choisi d'implémenter une méthode Lasso permettant d'accorder de l'importance à certaines variables seulement (selon la valeur du coefficient alpha). A l'inverse, la méthode Ridge n'a pas pour but d'isoler certaines variables mais de considérer l'ensemble des variables d'entraînement (features) pour la prédiction.

Pour la classification, nous avons choisi une méthode d'arbre de décision qui, en général, fonctionne particulièrement bien lorsque l'on considère des problèmes avec de nombreuses classes. Nous avons aussi choisi une méthode des k-NN car empiriquement, cette dernière fonctionne particulièrement bien lorsque le nombre de classes n'est pas trop grand. Son principal défaut étant que le temps de réponse peut être long lorsque le jeu de données est trop volumineux. Enfin ces méthodes s'expliquent assez bien et c'est pourquoi nous trouvions pertinent d'adopter ces choix pour que l'utilisateur applique une méthode selon ses contraintes et besoins.

Pour finir un bouton « Valider » permet d'éxcecuter le modèle sélectionné. Ce bouton lance une fonction passée en callback (*choix_algo()*), qui permettra de lancer la fonction associée au modèle de l'algorithme selectionné. Lorsque le modèle de prédiction est excecuté, les résultats apparaissent à droite. Cette partie est composée des informations sur le modèle, de l'évaluation qualitative et des résultats graphiques des prédictions. L'évaluation du modèle se fait par une matrice de confusion lorsque nous sommes dans le cas d'une classification. Si le modèle exécuté est une régression, le modèle sera alors évalué avec un score R2. Les résultats peuvent être sauvegardé au format png, avec le bouton « Sauvegarder ».

Schéma de l'application



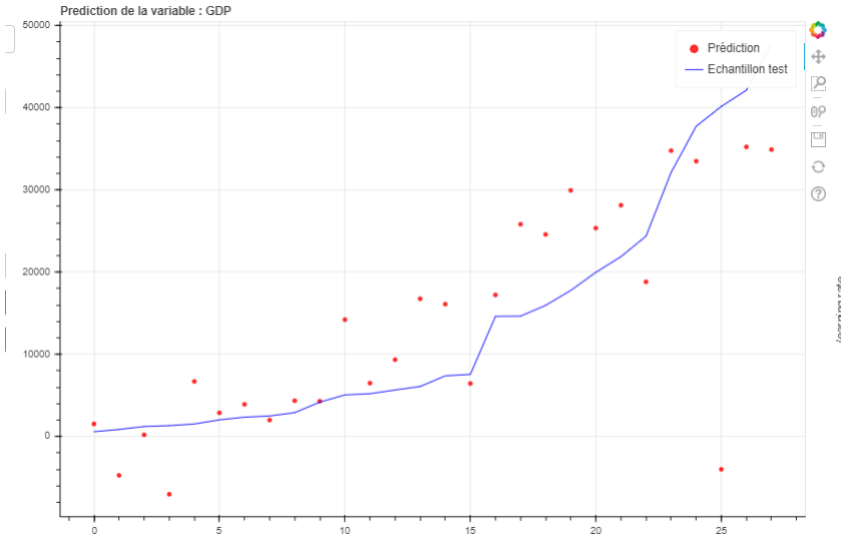
Pour mieux comprendre le fonctionnement de l'application vous trouverez ci-dessous un guide d'utilisation de l'application accompagné de deux exemples. Un pour chaque type d'apprentissage supervisé.

Pour commencer l'utilisation de l'application, il faut commencer par paramétrer le modèle grâce aux éléments présenter précédemment.

Vous devez donc choisir votre fichier, votre variable cible et variables prédictives et votre algorithme.

Voici ci-dessous quelques exemples de résultats :

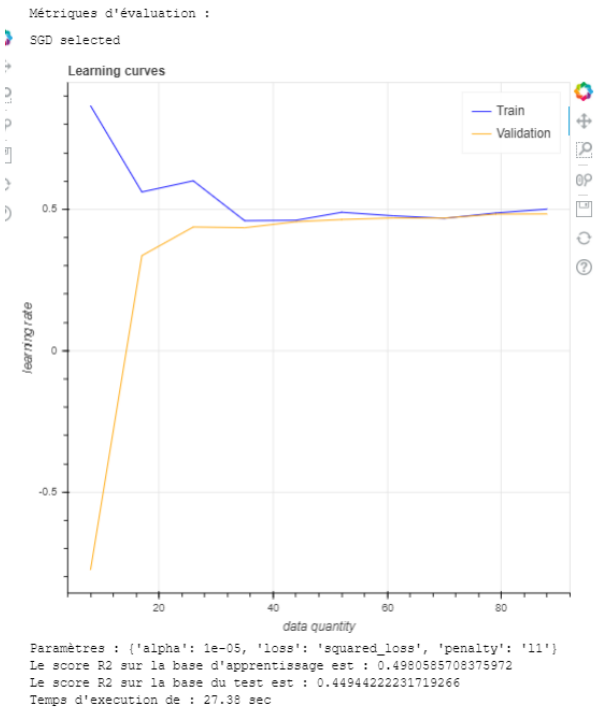
Exemple de régression :



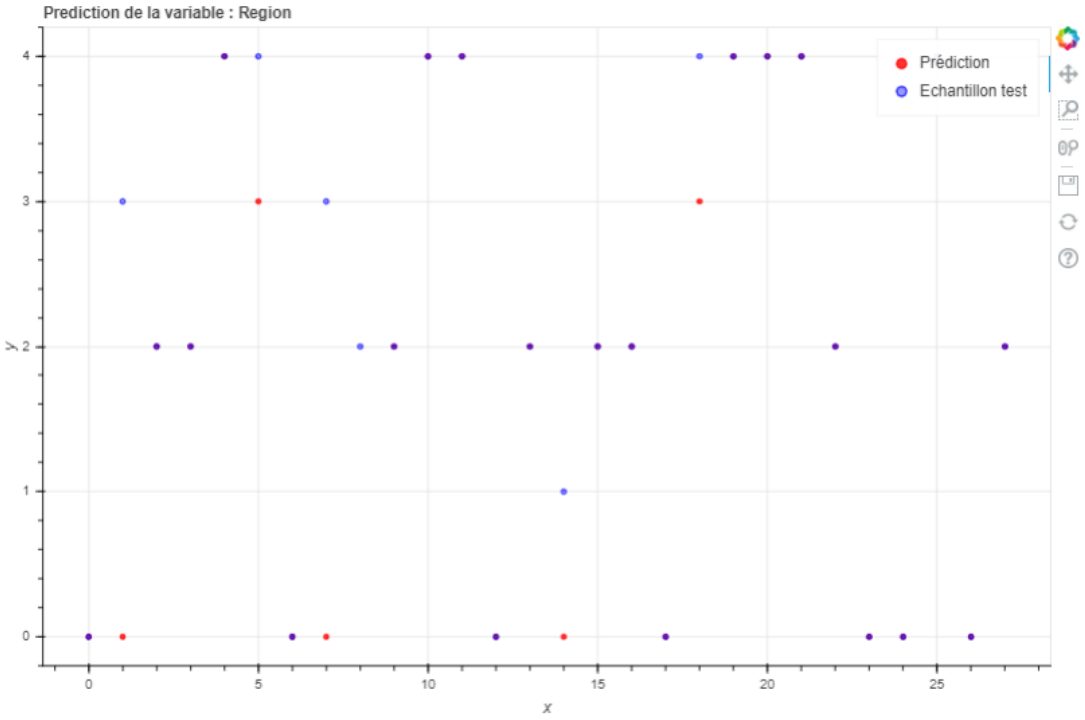
Graphique** pour représenter la **différence** entre les **prédictions** et les **valeurs tests

***Learning curves** :*
Affichent l'évolution de l'apprentissage en fonction de la quantité de données. Elles permettent de constater s'il y a un intérêt à récolter plus de données : vrai si les courbes sont croissantes, faux si elles convergent. Et de détecter les cas d'overfitting si l'écart entre les 2 courbes est important. Ici, l'écart est faible et les courbes convergent.

***Indicateurs pour une régression** :*
Paramètres du modèle
R2 et temps d'exécution



Exemple de classification :



Dictionnaire des correspondances (clef, valeur) : {'America': 0, 'East Asia & Pacific': 1, 'Europe & Central Asia': 2, 'South Asia': 3, 'Sub-Saharan Africa': 4}

Graphique** pour représenter la **différence** entre les **prédictions** et les **valeurs tests

***Indicateurs pour une classification** :*
Paramètres, temps d'exécution, Matrice de confusion et le rapport de classification

```
Paramètres : {'criterion': 'entropy', 'max_depth': 4}
Temps d'execution de : 0.79 sec

Confusion matrix :
[[7 0 0 0 0 0]
 [1 0 0 0 0 0]
 [0 0 8 1 0 0]
 [0 0 0 0 0 0]
 [2 0 0 0 0 0]
 [0 0 0 0 2 7]]
Classification report :
```

index	precision	recall	f1-score	support
America	0.7	1	0.8235294117647058	7
East Asia & Pacific	0	0	0	1
Europe & Central Asia	1	0.8888888888888888	0.9411764705882353	9
Middle East & North Afr	0	0	0	0
South Asia	0	0	0	2
Sub-Saharan Africa	1	0.7777777777777778	0.8750000000000001	9
accuracy	0.7857142857142857	0.7857142857142857	0.7857142857142857	0.7857142857142857
macro avg	0.45	0.4444444444444444	0.4399509803921568	28
weighted avg	0.8178571428571428	0.7857142857142857	0.7896533613445378	28

Résultats :

- L'application que nous avons créée répond au cahier des charges imposé. Elle permet à un utilisateur d'utiliser des algorithmes de Machine Learning avec les features qui lui conviennent et d'en constater les résultats. Les modèles sont testés selon différents paramétrages avec une validation croisée en interne afin de présenter à l'utilisateur la meilleure configuration. Enfin dans un souci de comparaison de modèles, nous avons ajouté un bouton sauvegarder qui appelle une fonction (*save_as()*) afin de sauvergarder les resultats au format png.

Problèmes :

- Compréhension du sujet. Nous n'avions pas tous compris les mêmes attentes de ce projet. Il a donc fallu prendre le temps de bien analyser le sujet et de se mettre en accord sur ce que nous souhaitions faire.

Esprit critique (améliorations) :

- Comme l'application est à destination de personnes qui ne sont pas expertes dans le domaine, nous avons choisi de faire le paramétrage des hyperparamètres automatiquement, en choisissant le meilleur hyperparamètre au sein de nos fonctions. Mais il pourrait être intéressant dans une prochaine version de l'application, de laisser le choix à l'utilisateur d'intégrer ou non ses hyperparamètres. Ce qui permettrait aux utilisateurs qui s'y connaissent de personnaliser leur modèle.
- De plus dans une prochaine version nous pourrions permettre à l'utilisateur de choisir plusieurs algorithmes en même temps et d'afficher les résultats de chaque algorithmes côte à côte.