Université Grenoble-Alpes

Développement d'application RShiny

Auteurs
Komi Agblodoe
Azat Aleksanyan
Lucas Chabeau
Mamadou Diamban
Soheil Salmani

Tuteur
Didier Morel

21 février 2020

Remerciements

Nous tenons à remercier particulièrement Monsieur Didier Morel pour leur aide, leur disponibilité et leur écoute tout au long du projet.

Nous remercions aussi le BD pour la confiance qu'ils nous ont accordé avec ce projet.

Table des Matières

1. Introduction

Dans le cadre de notre préparation au diplôme d'ingénieur en statistiques et sciences de données, nous avons été amenés à effectuer notre projet de cinquième année en partenariat avec l'entreprise BD (Becton et Dickinson) et sous l'encadrement de Didier Morel.

Il concerne le développement d'une application RShiny permettant de comparer et de tester des algorithmes de Machine Learning. Dans la mesure où il n'existe apparemment aucune application en ligne permettant de réaliser ce type de travail, il semble très intéressant de mettre en place un tel outil. Ainsi, les expérimentés comme les personnes sans trop de connaissances en apprentissage automatique pourront se servir de l'application pour avoir un aperçu de l'efficacité de différentes méthodes.

Dans ce projet, après avoir identifié le contexte, les besoins et les contraintes, nous avons procédé à la construction de notre application en créant deux fichiers dont un pour le serveur et le second pour l'interface utilisateur. Nous avons implémenté dans notre application, des algorithmes de classification et de régression comme AdaBoost Classification Trees, AdaBoost.M1, CART, les k plus proches voisins (kNN), le randomForest, RfRanger, XGBoost, glmNetL1, glmNetL2. Pour ces différents algorithmes, l'utilisateur a la possibilité de choisir les paramètres voulus. Nous avons ensuite procédé aux tests sur des jeux de données afin de comparer la performance des différents algorithmes de classification. Il faut noter aussi que notre application est capable de charger les jeux de données de type CSV et TSV.

2. Présentation de l'application

Notre application web est constituée d'un menu permanent qui permet de naviguer. Sur le site, une page d'accueil présente l'outil général à l'internaute.

2.1 Présentation succinte de Shiny

Shiny, est un package R qui facilite la construction d'applications web interactives depuis R permettant une sortie en html. Une application shiny se structure en deux parties à savoir :

- Un côté UI (User Interface) qui regroupe tous les éléments de mise en forme et d'affichage de l'interface utilisateur (affichage des inputs et des outputs).
- Un côté Serveur où sont exécutés les codes R qui servent à produire les éléments de sorties (graphiques, tables, traitements, etc..) et à les mettre à jour en cas de changement dans les valeurs d'entrées (inputs).

Les utilisateurs peuvent simplement manipuler l'application pour exécuter et afficher des résultats fournis par du code R.

Notre application a été structurée en deux parties comme décrites ci-haut.

2.2 Rôles de l'application

Le travail qui nous a été demandé par Didier Morel est le développemnt d'une application avec Shiny. Le développement de cette application passe par l'implémentation des algorithmes de machine learning. Cette application doit être capable de:

- Tester l'efficacité des algorithmes (vitesse d'exécution)
- Tester la qualité des modèles fournis par les algorithmes (erreurs/taux de bonne classification)
- Donner une sortie PDF qui résume les résultats des tests d'algorithmes

2.3 Contraintes de l'application

En ce qui concerne notre application web, différentes contraintes inhérentes à sa nature sont prises en compte à savoir:

- L'application doit être compatible avec n'importe quel jeu de données (format TSV ou CSV)
- L'application doit donc être réactive
- L'utilisateur doit pouvoir choisir les variables explicatives et les variables à expliquer
- L'utilisateur choisit quels algorithmes tester sur son jeu de données
- L'application doit être facile à maintenir.

3. Machine Learning Algorithmes

Afin de répondre au mieux à notre problématique nous avons fait le choix d'utiliser plusieurs algorithms différentes pour analyser nos données.

kNN /k-ppv/

L'algorithme des kNN consiste a trouver est les k observations x_i les plus proches de l'observation x' a classifier. Ensuite, il faut definir $f(x')$ en fonction des reponses y_i des kNN. Pour la regression c'est la valeur moyenne et pour la classification - la vote majoritaire. Les questions ouvertes pour cette algorithme est la choix du **critere de distance** et de la **valeur de k**.

- kNN pour la regression:

$$\tilde{f}(x) = moyenne(y_i | i \in N_k(x))$$

$$\Rightarrow \text{approxime directement la fonction de regression } E[Y|X]$$

Nature de l'approximation

1. espérance → moyenne empirique
2. valeur ponctuelle → voisinage (dans le conditionnement)

- kNN pout la classification:

$$\tilde{f}(x) = majorite(y_i | i \in N_k(x)) = arg \max_{l=1,\dots,K} \tilde{P}_l = \frac{1}{k} \sum_{i \in N_k(x))} 1(y_i = l)$$

$$\Rightarrow \text{approxime directement le classifieur de Bayes:}$$

$$arg \max_{l=1,\dots,K} P(Y = C_l | X = x)$$

Nature de l'approximation

1. probabilité → proportion empirique
2. valeur ponctuelle → voisinage (dans le conditionnement)

CART - Classification & Regression Trees

De base, les algorithmes d'arbre de décision ne sont rien d'autre que des instructions if-else qui peuvent être utilisées pour prédire un résultat basé sur des données. La méthodologie CART ou Arbres de classification et de régression fait référence à ces deux types d'arbres de décision.

- arbres de classification

Un arbre de classification est un algorithme dans lequel la variable cible est fixe ou catégorielle. L'algorithme est ensuite utilisé pour identifier la «classe» dans laquelle une variable cible se situerait le plus probablement. Un arbre de classification divise l'ensemble de données en fonction de l'homogénéité des données. Disons, par exemple, qu'il y a deux variables; revenu et âge; qui déterminent si un consommateur achètera ou non un type particulier de téléphone.

Si les données de formation montrent que 95% des personnes de plus de 30 ans ont acheté le téléphone, les données y sont divisées et l'âge devient un nœud supérieur dans l'arbre. Cette division rend les données «pures à 95%». Des mesures d'impuretés comme l'entropie ou l'indice de Gini sont utilisées pour quantifier l'homogénéité des données en ce qui concerne les arbres de classification.

- arbres de régression

Un arbre de régression fait référence à un algorithme dans lequel se trouve la variable cible et l'algorithme utilisé pour prédire sa valeur. Dans un arbre de régression, un modèle de régression est ajusté à la variable cible en utilisant chacune des variables indépendantes. Après cela, les données sont divisées en plusieurs points pour chaque variable indépendante.

À chacun de ces points, l'erreur entre les valeurs prédites et les valeurs réelles est mise au carré pour obtenir une «somme des erreurs au carré» (SEC). Le SEC est comparée entre les variables et la variable ou le point qui a le SEC le plus bas est choisi comme point de partage. Ce processus se poursuit récursivement.

rfRanger ???

Random Forest

À la base, la méthode des forêts aléatoires est basée sur le bagging /aggrégation par bootstrap/, il est très performant sur de nombreux problèmes et facile à paramétrer. Principe d'algorithmes et de découper l'espace d'entrée en région, estimer et prédire une valeur par région. Pour la régression on choisit une valeur moyenne dans la région, et pour la classification on choisit la classe majoritaire. Les avantages de cette méthode sont l'interprétabilité du modèle et le mécanisme de prédiction proche du processus humain.

XGBoost

La bibliothèque XGBoost implémente l'algorithme d'arbre de décision de renforcement du gradient. Le boosting est une technique d'ensemble où de nouveaux modèles sont ajoutés pour corriger les erreurs commises par les modèles existants. Les modèles sont ajoutés séquentiellement jusqu'à ce qu'aucune autre amélioration ne puisse être apportée.

L'amplification du gradient est une approche où de nouveaux modèles sont créés qui prédisent les résidus ou les erreurs des modèles précédents, puis additionnés pour faire la prédiction finale. Il est appelé boosting de gradient car il utilise un algorithme de descente de gradient pour minimiser la perte lors de l'ajout de nouveaux modèles.

Cette approche prend en charge à la fois les problèmes de modélisation prédictive de régression et de classification.

glmNetL2 - Ridge Regression

Régression linéaire avec perte quadratique et pénalité L2 :

$$(w^*, b^*) = \underset{w\in\mathbb{R}^l, b\in\mathbb{R}}{argmin} \sum_{i=1}^{n} (y_i - f(x_i))^2 + \lambda ||w||_2^2$$

$$= \underset{w\in\mathbb{R}^l, b\in\mathbb{R}}{argmin} \sum_{i=1}^{n} (y_i - b - \sum_{j=1}^{p} w_j x_{ij} + \lambda \sum_{j=1}^{p} w_j^2$$

la régression ridge impose donc une contrainte sur les coefficients (w). Le terme de pénalité (λ) régularise les coefficients de telle sorte que si les coefficients prennent de grandes valeurs, la fonction d'optimisation est pénalisée. Ainsi, la régression ridge réduit les coefficients et contribue à réduire la complexité du modèle et la multi-colinéarité.

Pour le résoudre il faut centrer les variables $\rightarrow \tilde{x}_{ij} = x_{ij} - \bar{x}$

1. on estime alors l'intercept b par $b^* = \hat{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$
2. on obtient w avec le meme probleme sans intercept
3. solution : $w^* = (X^T X + \lambda I)^{-1} X^T y$ *ou* $X[i,j] = \tilde{x}_{ij}$

glmNetL1 - Lasso Regression

Régression linéaire avec perte quadratique et pénalité L1 :

$$(w^*, b^*) = \underset{w\in\mathbb{R}^{'}, b\in\mathbb{R}}{argmin} \sum_{i=1}^{n} (y_i - f(x_i))^2 + \lambda ||w||_1$$

$$= \underset{w\in\mathbb{R}^{'}, b\in\mathbb{R}}{argmin} \sum_{i=1}^{n} (y_i - b - \sum_{j=1}^{p} w_j x_{ij} + \lambda \sum_{j=1}^{p} |w_j|$$

Si on regarde l'equation en detail, in peut voir que la seule différence est qu'au lieu de prendre le carré des coefficients, les grandeurs sont prises en compte. Ce type de régularisation (L1) peut conduire à des coefficients nuls, c'est-à-dire que certaines caractéristiques sont complètement négligées pour l'évaluation des résultats. Ainsi, la régression Lasso aide non seulement à réduire le sur-ajustement, mais elle peut également nous aider à sélectionner les variables.

Du coup, par rapport à pénalité ridge :

- même effet de régularisation : shrinkage des coefficients
- mais conduit à des coefficients exactement = 0

⇒ solution parcimonieuse (sparse) : sélection de variables

AdaBoost Classification Trees ????

AdaBoost M1

AdaBoost (boosting adaptatif) est un algorithme d'apprentissage assembliste qui peut être utilisé pour la classification ou la régression. Bien qu'AdaBoost résiste mieux au surajustement qu'un grand nombre d'algorithmes de machine learning, il est parfois sensible aux données bruitées et aux valeurs aberrantes.

AdaBoost est désigné comme adaptatif car il utilise de nombreuses itérations pour générer un apprenant composite fort. AdaBoost crée l'apprenant fort (un classifieur bien corrélé au classifieur correct) en ajoutant de manière itérative des apprenants faibles (un classifieur légèrement corrélé au classifieur correct). Lors de chaque session d'apprentissage, un nouvel apprenant faible est ajouté à l'ensemble et un vecteurpondération est ajusté afin de mettre l'accent sur les exemples ayant été classés de manière incorrecte lors des sessions précédentes. Par conséquent, le classifieur obtenu est doté d'une meilleure précision queclassifieurs des apprenants faibles.

AdaBoost.M1 represente les algorithmes originaux de classification binaire.

Optimisation du modèle - Validation Croisée

L'optimisation du modèle signifie l'optimisation hyperparamétrique, c'est-à-dire il faut trouver le bon niveau de complexité du modèle, qui est en général difficile de choisir a priori. Pour l'optimisation on utilise la methode de la validation croissee.

Au préalable, on définit un jeu de test pour évaluer les performances du modèle. Ensuite, on découpe led'apprentissage en K parties - les folds.

pour $k = 1$ *a* K, il faut :

- mettre de côté la $k - ième$ fold
- apprendre le modèle sur les $K - 1$ folds restantes
- appliquer le modèle sur les données de la $k - ième$ fold

A la fin, on évalue les performances du modèle.

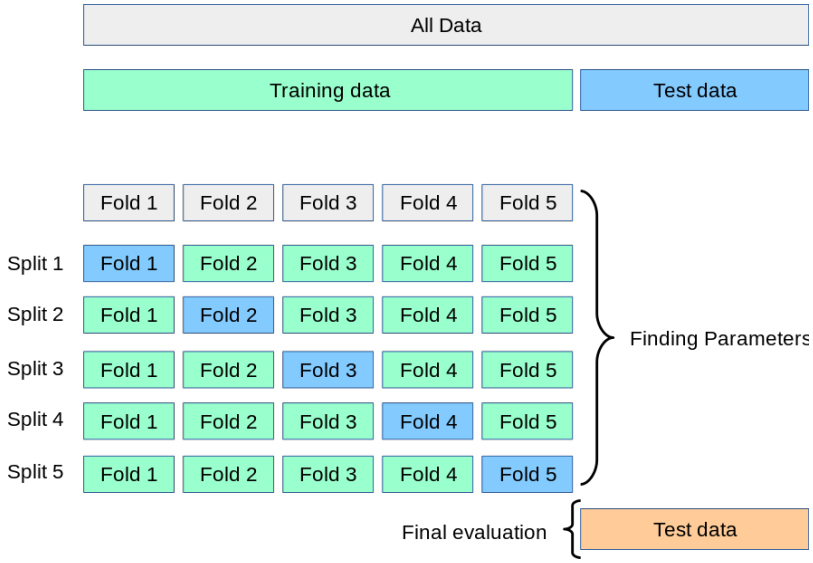

Figure 1: Validation Croisée

4. Développement d'application

L'interface utilisateur a été conçue pour être intuitive et fonctionnelle. Répondant aux contraintes, elle permet à l'utilisateur de charger un jeu de données (de type TSV ou CSV), de choisir les variables explicatives et celles à expliquer, ainsi que les algorithmes et les paramètres de son choix. Outre celà, l'utilisateur a aussi la possibilté de faire des statistiques decriptives pour lui permettre d'explorer son jeu de données.

Ci-dessous une capture d'écran de l'application avec un exemple à l'appui.

IMAGE INPUT - DONT FORGET

4.1 Préparation de donnees

notre application commence par la préparation des données. Dans cette fenêtre, l'utilisateur pourra:

- télécharger des fichiers de données (.csv / .tsv à ce stade de développement)
- choisir la présence de "header", le séparateur et la citation
- et enfin voir un aperçu rapide des données après avoir cliqué sur le bouton *"Upload"*.

4.2 Statistiques Descriptives

Après avoir téléchargé notre fichier de données dans l'application, dans la deuxième fenêtre, nous pourrons voir une statistique descriptive automatisée.

Sur le premier graphique, nous pouvons voir la description des informations de base dans la classe de données d'entrée:

c'est la description du fonction, mais dans l'exapmle de notre appli quelques points sont absents. faut relire

- *lignes*: nombre de lignes
- *colonnes*: nombre de colonnes
- *discrete_columns*: nombre de colonnes discrètes
- *colonnes_continues*: nombre de colonnes continues
- *all_missing_columns*: nombre de colonnes avec tout ce qui manque
- *total_missing_values*: nombre d'observations manquantes
- *complete_rows*: nombre de lignes sans valeurs manquantes
- *total_observations*: nombre total d'observations
- *memory_usage*: allocation de mémoire estimée en octets

Dans le plot du milieu, nous observons la fréquence des valeurs manquantes pour chaque entité. Il y a 4 groupes pour décrire la situation des valeurs manquantes.

- *Good* : quand la proportion des valeurs manquantes ≤ 0.05
- *OK* : $0.05 <$ la proportion des valeurs manquante ≤ 0.4
- *Bad* : $0.4 <$ la proportion des valeurs manquante ≤ 0.8
- *Remove* : $0.8 <$ la proportion des valeurs manquante ≤ 1

Et sur le graphique de droite, nous pouvons voir nos variables une par une. Si la variable se compose de données numériques, un boxplot sera généré, et dans le cas de données non numériques, nous verrons un barplot.

Si dans le jeu de données des valeurs manquantes ont été trouvées, notre application vous proposera de faire l'imputation. Pour le moment, trois méthodes d'imputation sont disponibles:

- pmm */Predictive mean matching (Correspondance moyenne prédictive)/* : Il vise à réduire le biais introduit dans un ensemble de données par imputation, en tirant des valeurs réelles échantillonnées à partir des données.
- norm : Calcule les imputations pour les données manquantes univariées par régression linéaire bayésienne, également connue sous le nom de modèle normal.
- mean : Une autre technique d'imputation consiste à remplacer toute valeur manquante par la moyenne de cette variable pour tous les autres cas, ce qui présente l'avantage de ne pas modifier la moyenne de l'échantillon pour cette variable

Après l'exécution, nous verrons une vue rapide de l'ensemble de données, où nous pouvons être assurés que les valeurs manquantes ont été imputées.

4.3 ML Parameteres

À cet étape, après avoir déjà téléchargé le fichier de données et effectué quelques statistiques descriptives, il est temps pour la première modélisation. Et tout d'abord, l'utilisateur aura la possibilité de choisir des paramètres pour les algorithmes souhaités. Donc, la troisième fenêtre de l'application dédiée aux hyperparamètres et autres spécificités.

Premièrement, on est confrontés au choix entre la régression et la classification.

Rappel

`La classification est la tâche de prédire une étiquette de classe discrète.`
`La régression consiste à prédire une quantité continue.`

Ensuite, c'est le choix des hyperparamètres. Ici, les choix de l'utilisateur sont:

- soit saisir les paramètres définis par les siens
- soit laisser le package ***caret*** de R régler les hyperparamètres */parameter tuning/*

Réglage des hyperparamètres avec caret

Avec l'application, un fichier .csv sera fourni avec les valeurs par défaut pour le réglage des hyperparamètres que vous pouvez voir ci-dessous.

Table 1: Les valeurs par défaut pour le réglage des hyperparamètres

name	method	preProc	trainMetric	trainControl	tuneGrid
CART	rpart		Accuracy	method = 'cv', number = 10	
kNN	knn	center scale	Accuracy	method = 'cv', number = 10	
randomForest	rf		Accuracy	method = 'cv', number = 10	
rfRanger	ranger		Accuracy	method = 'cv', number = 10	
XGBoost	xgbTree		Accuracy	method = 'cv', number = 10	
glmNetL1	glmnet	center scale	Accuracy	method = 'cv', number = 10	alpha = 1, lambda = seq(0, 0.5, 0.001)
glmNetL2	glmnet	center scale	Accuracy	method = 'cv', number = 10	alpha = 0, lambda = seq(0, 0.5, 0.01)

Comme on veux voir:

- pour certains des algorithmes, comme *preProcessing*, les donnees sont centrees et reduites par defaut
- les meilleurs parametres sont choisis en comparant le differents */Accuracy/*
- la qualité des paramètres sera contrôlée par une validation croisée (10 fois)
- pour les algorithmes glmnet */Ridge et Lasso/* α et les valeurs possibles de λ sont définies en complément.

Réglage des hyperparamètres à la main

Si de toute façon l'utilisateur décide d'entrer ses propres paramètres, il doit choisir les algorithmes que l'on souhaite utiliser dans la partie intitulée "Algorithmes" et entrer les paramètres. Après exécution, dans une nouvelle fenêtre on peut voir le choix final des paramètres qui seront utilisés pour la partie principale de modélisation.

4.4 ML Algorithme

Après avoir réglé/importé tous les hyperparamètres, il est temps de visualiser et de comparer la qualité et la précision des algorithmes pour les données.

Ici, l'utilisateur aura la possibilité de :

- choisir la variable d'interet
- confirmer l'option de validation croisée et les folds
- varier la proportion de données d'apprentissage

Ensuite, la comparaison de la précision et de kappa seront affichées sur l'application.