La Cross Validation

Marko ARSIC / Rindra LUTZ

15/11/2020

Introduction

Le monde d'aujourd'hui est un monde connecté. Cette connectivité apporte son lot de d'informations diverses : ce sont les données, ou data en anglais.

Pour traiter ces données, de nouveaux métiers ont vu le jour. Ces nouveaux métiers font appel à des traitements spécifiques dans le domaine des données comme par exemple la data analyse, la data science ou encore le datamining.

Le datamining regroupe des méthodes scientifiques destinées à l'exploration et l'analyse de données, à partir de grands volumes d'informations/de données, dans le but de créer de la valeur, comprendre des phénomènes, comprendre notre monde, et en particulier afin d'aider à prendre des décisions, anticiper des événements et agir.

Dans le datamining, il existe principalement deux grandes familles de méthodes. Ce sont les méthodes descriptives et les méthodes prédictives.

Méthodes descriptives

Une méthode descriptive (dite non supervisée) correspond à la recherche de structure, de relation, de corrélation.

- Permet de mettre en évidence des informations non visibles simplement
- Permet de résumer, synthétiser les données
- Sans variable ou phénomène à expliquer a priori

Ci-dessous, un tableau des méthodes descriptives.

Analyse factorielle ***Projection dans un espace de dimension inférieure***	Analyse en composantes principales (variables continues), Analyse factorielle des correspondances (2 var. qualitatives), Analyse des correspondances multiples (+ 2 var. qualitatives)
Analyse typologique ***Regroupement en classes homogènes***	Méthodes hiérarchiques : classification ascendante/descendante hiérarchique Méthodes de partitionnement : centres mobiles, k-means, nuées dynamiques Classification neuronale (cartes de Kohonen : analyse typologique + réduction de dimensions)
Modèles combinatoires	Classification relationnelle (variables qualitatives) : peu répandu
Modèles à base de règles	Règles d'associations : détection de liens

Méthodes prédictives

Une méthode prédictive (dite supervisée) correspond à la modélisation et la prédiction d'un phénomène donné.

- Permet de définir un pattern (un modèle/une relation) pour expliquer un événement
- Permet d'extrapoler la cible
- Avec une variable/un événement à expliquer

Ci-dessous, un tableau des méthodes prédictives.

Régression (linéaire, logistique, modèle linéaire général GLM, PLS)	
Arbres de décision (ou encore d'induction ou de segmentation)	Modèle à base de règles, sur variable à expliquer continue ou qualitative
Réseaux de neurones	Réseaux à apprentissage supervisé (perceptron multicouche, réseau de neurones profond)
Analyse discriminante	
SVM – Support Vecteur Machine	

Cross-Validation

Traitant le sujet de la cross-validation, qui est une régression logistique, nous nous concentrerons sur les méthodes prédictives, et donc plus particulièrement sur les méthodes de régression.

La régression logistique est une technique prédictive. Elle vise à construire un modèle permettant de prédire/expliquer les valeurs prises par une variable cible qualitative (le plus souvent binaire, on parle alors de régression logistique binaire, et si elle possède plus de 2 modalités, on parle de régression logistique polytomique) à partir d'un ensemble de variables explicatives quantitatives ou qualitatives (un codage est nécessaire dans ce cas).

Une fois que le modèle a été établi grâce à différents outils statistiques, il est alors nécessaire de valider sa fiabilité.

La cross validation pour mesurer la fiabilité du modèle

Lors de toute modélisation, il est nécessaire de définir :

• Une population d'apprentissage (Train) : pour entrainer le modèle
• Une population de test (Test) : pour mesurer, tester la performance et robustesse du modèle

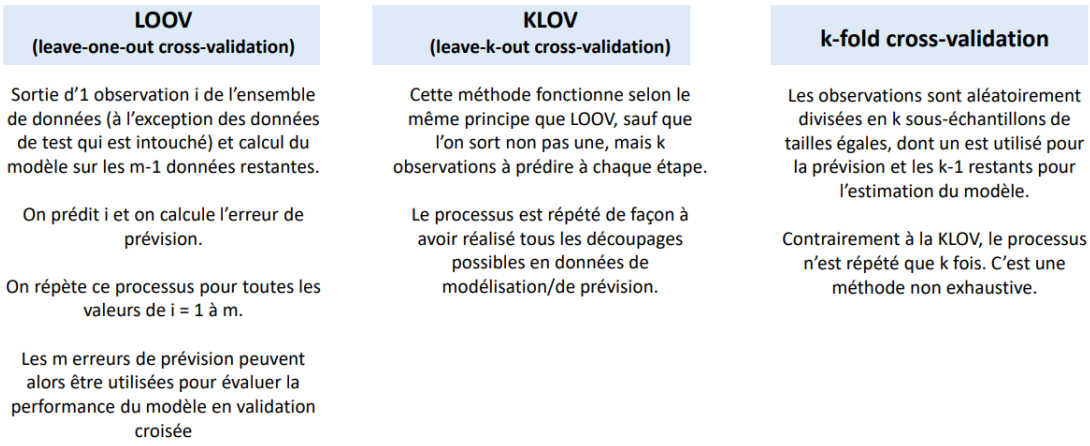
Or, une vérification via la cross-validation demanderait de travailler ici avec 3 types d'échantillons :

• Train
• Valid
• Test

Cet échantillon complémentaire (Valid) permet par exemple de tester plusieurs modèles (en faisant varier les paramètres du modèle ou les variables) : on essaie plusieurs modèles sur le Train et on identifie le plus performant sur le Valid. On teste enfin sur l'échantillon de Test (totalement vierge) le pouvoir de généralisation/sur-apprentissage sur des données toutes fraîches.

Il est possible de sophistiquer la structure des échantillons de Train/Validation au travers de quelques méthodes de validation croisée. En effet, les résultats dépendent de la manière dont ont été construits les 3 sous-ensembles Train/Valid/Test.

Les 3 principales méthodes de validation croisée sont :
• LOOCV (leave-one-out cross-validation)
• LKOCV (leave-k-out cross-validation)
• k-fold cross-validation



La cross validation permet aussi de déterminer des paramètres du modèle : on met en compétition k « sous-modèles » dont on mesure la performance pour déterminer le paramètre testé dont la performance du modèle est la meilleure.

Parlons rapidement du problème du sur-apprentissage, problème largement présent en modélisation.

Le sur apprentissage : problématique majeure en modélisation

Un modèle trop complexe, intégrant trop d'inputs et « épousant » trop les données d'apprentissage amènera donc une très bonne performance sur l'échantillon d'apprentissage (par construction), mais aura trop appris, notamment les bruits ou cas aberrants.
Il sera alors moins performant sur des données qui n'ont pas servi à la construction du modèle, c'est-à-dire sur les données sur lesquelles on souhaite faire la prédiction. L'enjeu est donc de trouver le bon niveau de sophistication pour obtenir un bon niveau de performance sur l'échantillon d'apprentissage et sur l'échantillon de test.

Il n'y a pas sur-apprentissage lorsque la performance du modèle en Test est légèrement plus faible que celle en Train. Un écart trop grand est signe de **sur-apprentissage**.

Exemple pratique

Téléchargeons les packages tidyverse pour une manipulation et une visualisation des données plus faciles, ainsi que caret pour calculer facilement les méthodes de validation croisée (CV). Nous pouvons ensuite les appeler comme suit :

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse
1.3.0 --

## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.4     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0
```

```
## -- Conflicts ---------------------------------------
tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()

library(caret)

## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

Tidyverse est un ensemble de packages fonctionnant en harmonies. Ces packages partagent des représentations de données communes et une conception "API"(interfaces de programmation d'applications).

Caret est un ensemble de fonctions qui tentent de rationaliser le processus de création de modèles prédictifs Ce package contient des outils pour : le fractionnement des données, le prétraitement, la sélection des caractéristiques, le réglage du modèle par rééchantillonnage, l'estimation de l'importance des variables, et de nombreuses autres fonctionnalités.

Nous allons ici utiliser la table swiss, présente sous R. Cette table représente la mesure de la fécondité et des indicateurs socio-économiques standardisés pour chacune des 47 provinces francophones de Suisse vers l'an 1888.

```
# Téléchargement des données
data("swiss")
# Inspecter les données
sample_n(swiss, 3)

##             Fertility Agriculture Examination Education Catholic
## Val de Ruz       77.6        37.6          15         7     4.97
## Herens           77.3        89.7           5         2   100.00
## Rive Gauche      42.8        27.7          22        29    58.33
##             Infant.Mortality
## Val de Ruz              20.0
## Herens                  18.3
## Rive Gauche             19.3
```

Après avoir construit un modèle, nous souhaitons déterminer la précision de ce modèle sur la prédiction du résultat de nouvelles observations non utilisées pour construire le modèle. En d'autres termes, nous voulons estimer l'erreur de prédiction.

Pour ce faire, la stratégie de base consiste donc à :

- Construire le modèle sur un ensemble de données d'entraînement

- Appliquer le modèle sur un nouvel ensemble de données de test pour faire des prévisions

- Calculer les erreurs de prédiction

Il est ensuite temps de passer à l'étape suivante : la validation croisée.

Généralement, on utilise la validation croisée k-fold pour estimer le taux d'erreur de prédiction. Elle peut être utilisée dans les paramètres de régression et de classification. Il est toutefois à noter que d'autres méthodes sont applicables, dont les noms ont été mentionnés plus haut.

Une autre alternative à la validation croisée que nous n'avons pas évoquée est la méthode de rééchantillonnage bootstrap, qui consiste à sélectionner de manière répétée et aléatoire un échantillon de n observations à partir de l'ensemble de données original, et à évaluer la performance du modèle sur chaque copie.

La méthode de validation croisée k-fold évalue la performance du modèle sur différents sous-ensembles de données de formation et calcule ensuite le taux moyen d'erreur de prédiction. L'algorithme est le suivant :

1- Diviser aléatoirement l'ensemble de données en k sous-ensembles (par exemple 5 sous-ensembles)
2- Mettre de côté un sous-ensemble et former le modèle sur tous les autres sous-ensembles
3- Tester le modèle sur le sous-ensemble mis de côté et enregistrer l'erreur de prédiction
4- Répétez ce processus jusqu'à ce que chacun des k sous-ensembles ait servi d'ensemble de test.
5- Calculez la moyenne des k erreurs enregistrées. C'est ce qu'on appelle l'erreur de validation croisée qui sert de mesure de performance pour le modèle.

La validation croisée k-fold est une méthode robuste pour estimer la précision d'un modèle.

L'avantage de la CV k-fold par rapport à la LOOCV est le calcul, moins long. Un avantage moins évident, mais potentiellement plus important de la CV k-fold est qu'elle donne souvent des estimations plus précises du taux d'erreur des tests que la LOOCV.

La question typique est la suivante : comment choisir la bonne valeur de k ?

Une valeur inférieure de K est plus biaisée et donc indésirable. En revanche, une valeur plus élevée de K est moins biaisée, mais peut souffrir d'une grande variabilité. Il n'est pas difficile de voir qu'une valeur plus faible de k (disons k = 2) nous conduit toujours vers une approche de validation d'ensemble, alors qu'une valeur plus élevée de k (disons k = nombre de points de données) nous conduit à une approche de LOOCV.

Dans la pratique, on effectue généralement une validation croisée k-fold en utilisant k = 5 ou k = 10, car il a été démontré empiriquement que ces valeurs donnent des estimations du taux d'erreur de test qui ne souffrent ni d'un biais trop élevé ni d'une variance très élevée.

L'exemple suivant utilise une validation croisée 10-fold pour estimer l'erreur de prédiction.

```r
# Définition de l'échantillon d'entraînement
set.seed(123)
train.control <- trainControl(method = "cv", number = 10)
# Entraîner le modèle
model <- train(Fertility ~., data = swiss, method = "lm",
               trControl = train.control)
# Résultats résumés
print(model)
```

```
## Linear Regression
##
## 47 samples
##  5 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 42, 42, 42, 42, 42, 44, ...
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   7.424916  0.6922072  6.31218
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

La fonction set.seed() définit le numéro de départ utilisé pour générer une séquence de nombres aléatoires. Elle garantit que vous obtenez le même résultat si vous commencez avec ce même numéro chaque fois que vous exécutez le même processus. Par exemple, si j'utilise la fonction sample() immédiatement après avoir défini un numéro, j'obtiendrai toujours le même échantillon.
La fonction trainControl() de **caret** permet de fixer les paramètres du processus d'apprentissage.

Le RMSE, le R2 et le MAE sont tous trois des indicateurs importants.

Plus le R2 (coefficient de corrélation qui représente la part de la variance expliquée par le modèle) est proche de 1, meilleure sera la prédiction. Ici, le R2 est suffisamment élevé pour que l'on considère la prédiction bonne.
Le MAE est l'erreur moyenne absolue. Un des avantages du MAE est qu'il donne une meilleure idée de la qualité de prédiction. Par contre, il n'est pas possible de savoir si le modèle a tendance à sous ou sur-estimer les prédictions.
Un dernier indicateur pertinent est le RMSE (erreur quadratique moyenne). Cet indice fournit une indication par rapport à la dispersion ou la variabilité de la qualité de la prédiction. Le RMSE peut être relié à la variance du modèle.

Aucun des indicateurs précédemment cités n'est clairement meilleur que les autres. Au contraire, toutes ces métriques sont à utiliser ensemble pour mieux comprendre et caractériser la qualité d'un modèle de prédiction.

Le processus de division des données en k-fold peut être répété un certain nombre de fois, c'est ce qu'on appelle la validation croisée k-fold répétée.

L'erreur finale du modèle est considérée comme l'erreur moyenne du nombre de répétitions.

L'exemple suivant utilise une validation croisée 10-fold avec 3 répétitions :

```
# Définiiton de l'échantillon d'entraînement
set.seed(123)
train.control <- trainControl(method = "repeatedcv",
                              number = 10, repeats = 3)
# Entraîner le modèle
model <- train(Fertility ~., data = swiss, method = "lm",
               trControl = train.control)
# Résultats résumés
print(model)
```

```
## Linear Regression
##
## 47 samples
##  5 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 42, 42, 42, 42, 42, 44, ...
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   7.357186  0.6992415  6.15871
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

Ici, notre R2 a très légèrement augmenté, et notre RMSE et MAE ont tous deux très légèrement diminués. On a donc une amélioration de la qualité du modèle très minime avec la répétition de la méthode de cross-validation k-fold, mais cette amélioration peut être bien plus significative dans d'autres cas.

Plus on vérifie ses résultats, mieux la prédiction sera !

A noter :
* Lorsque l'on compare deux modèles, celui qui produit le RMSE le plus faible est le modèle préféré.
* En divisant la RMSE par la valeur moyenne de la variable de résultat, on obtient le taux d'erreur de prédiction, qui doit être aussi faible que possible.

Bibliographie

https://openclassrooms.com/fr/courses/4297211-evaluez-les-performances-dun-modele-de-machine-learning/4308241-mettez-en-place-un-cadre-de-validation-croisee

https://fr.wikipedia.org/wiki

https://fr.slideshare.net/SHUBHAMGUPTA864/kfolds-cross-validation-method

https://scikit-learn.org/stable/modules/cross_validation.html

https://www.youtube.com/watch?v=ymRSxSY1nak&t=289s

http://www.sthda.com/english/articles/38-regression-model-validation/157-cross-validation-essentials-in-r/