Prediction of transaction claim status

COIC Quentin & NAAMANE chemsdin

January 4, 2018

1 Présentation du challenge

Nous avons choisi le challenge proposé par *Priceminister-Rakuten* qui consiste à prédire si une transaction résultera sur une réclamation du client, et auquel cas, prédire de quel type de réclamation il s'agit.
Le problème correspond à une classification multiclasse (8 classes). Les données à notre disposition sont composées de 100 000 individus et 22 variables.

2 Traîtement en amont des données

2.1 Traîtement des NaN

Tout d'abord, nous avons évalué le nombre de lignes contenant des 'NaN', avons trouvé un nombre beaucoup trop grand pour un simple 'df.dropna()'. C'est pourquoi nous avons préféré combler une bonne partie des 'NaN' présents dans les donnés.
Premièrement, pour la variable 'SHIPPING_PRICE', on a considéré que les valeurs 'NaN' correspondaient à la valeur '<1'.
Puis, pour 'WARRANTIES_PRICE' qu'ils correspondaient à 0, signifiant qu'aucune assurance n'avait été prise. Ce qui nous a permis de supprimer la variable 'WARRANTIES_FLG'.
Enfin, pour la variable: 'PRICECLUB_STATUS' nous avons considéré les 'NaN' comme valant 'UNSUBSCRIBED'.

2.2 Traîtement des variables qualitatives

Pour les variables qualitatives induisant un ordre, on a directement fait correspondre les valeurs de ces variables à des entiers entre 0 ou 1 et le nombre de variables considérées. Par exemple, pour 'WARRANTIES_PRICE', on a fait correspondre les valeurs aux entiers entre 1 et 5, et comme dit plus haut, on a laissé le 0 pour les Na
Pour les variables n'incluant pas d'ordre, on a simplement créé des dummy variables leur correspondant. Par exemple: On a créé les dummy variables correspondant à la variable 'PRODUCT_FAMILY', on a donc obtenu 12 nouvelles colonnes, correspondants à chaque valeur possible de 'PRODUCT_FAMILY'.

2.3 Traîtements partiuliers

Concernant les départements, on a préféré ne pas inclure la variable pour l'instant, car la création de dummy variables ajoutait trop de variables.
Nous avons préféré considéré 'PRODUCT_FAMILY' plutôt que 'PRODUCT_TYPE' car cette dernière induisait beaucoup de dummy variables.
On a finalement effectué un 'drop.na()' pour se débarrasser des derniers 'NaN' qui restaient dans les données. En effet, après les traîtements donnés ci-dessus, on se retrouvait avec environ 6000 lignes supprimées, plutôt que 90 000 initialement. On a donc au final des données avec 94 159 individus et 80 variables.

3 Application des méthodes

Pour cette partie, la métrique AUC, que l'énoncé du challenge demandait n'était pas directement utilisable pour la classification multiclasse, donc on s'est tourné, pour le moment, vers la métrique 'Accuracy' qu'on a utilisé en cours.

On a pour l'instant testé trois méthode (régression logistique, random forest et réseau de neurones), d'abord uniquement avec les paramètres par défaut, puis en les optimisant à l'aide d'une "grid search".
Pour la régression logistique, on a optimisé le paramètre 'C' de pénalisation.
Pour Random Forest, on a optimisé les paramètres: n_estimator, qui correspond au nombre d'arbres construits, et max_depth, qui correspond à la profondeur maximale de ces arbres.

Méthode	Accuracy avant optimisation	Accuracy après optimisation
Régression logistique	0.5059	0.5085
Random Forest	0.5077	0.5367

Table 1: Performances des méthodes

Nous n'avons pour l'instant pas de résultats concernant les réseaux de neurones car les paramètres par défaut donnaient des résultats bien trop instables et toujours moins bons que ceux des autres méthodes.

4 Idées pour la suite

Au vu des résultats obtenus, on aimerait encore améliorer la performance du Random Forest. Sachant que le GridSearch effectué a donné pour paramètres optimaux: 26 arbres avec une profondeur maximale de 19, sachant que cette profondeur était cherchée entre 1 e 19, on peut se dire qu'il est possible d'améliorer ce paramètre.

De plus, il serait bon de trouver une manière de traîter les variables laissées de côté cette fois ci, et peut-être d'en créer d'autres plus pertinentes.

Concernant les réseaux de neurones, Il serait intéressant de stabiliser les résultats obtenus en optimisant les paramètres (nombre de couches, et nombre de neurones par couches).