Compétition Kaggle: American Express-Default Prediction

Table des matières

I. Introduction

Pour terminer la formation, il est demandé de participer à une compétition Kaggle en cours. Parmi les compétitions ouvertes au début de ce projet, celles servant à l'apprentissage, et donc les compétitions très connues comme la classification des chiffres manuscrits ou la prédiction de survivants sur le Titanic sont écartées. Par conséquent, puisque je m'intéresse plus particulièrement aux données tabulées, la compétition « American Express - Default Prediction » (lien ici) a été choisie.

Le but de cette compétition est de prévoir si un client sera en défaut de paiement, ou s'il sera en capacité de rembourser son crédit, à partir de données sur les précédentes opérations des clients. American Express cherche à travers cette compétition à améliorer ses modèles de prédiction afin d'améliorer les décisions de prêts et donc d'obtenir une meilleure expérience client. Pour cela, la récompense attribuée à cette compétition s'élève à 100k$.

Afin de mener à bien cette modélisation, des données industrielles, c'est-à-dire provenant directement de la société sont à disposition. Dans un souci de confidentialité, les données ont été anonymisées et prétraitées. La base de données a déjà été séparée en un jeu d'apprentissage et un jeu de test, composées dans les deux cas d'un très grand nombre d'observations. La taille de la base de données

est si conséquente que cela va créer un problème de gestion de mémoire, qui sera discuté à différents moments de ce rapport.

Le temps attribué pour ce projet étant assez court, une étude complète n'est pas envisagée. Le choix a été fait de s'inspirer des travaux réalisés par d'autres compétiteurs et partagés sur Kaggle afin d'essayer d'améliorer la performance de leurs modèles. Plus en détails, une première partie sera consacrée à l'exploration des données afin de se familiariser avec les données. Ensuite la partie modélisation sera discutée, en s'appuyant sur un traitement de données proposé par d'autres utilisateurs, en essayant de le modifier légèrement. Une moyenne de prédiction de différents modèles sera réalisée afin d'espérer un gain de performance. Pour finir, la soumission des résultats et les scores obtenus sur différents essais sera présentée.

II. Modification des données

Un des problèmes majeurs de cette compétition est la base de données à disposition. Bien qu'une base de données importante puisse permettre d'atteindre une meilleure performance, cela pose un problème de mémoire à l'utilisation. En effet, les données d'apprentissage ont une taille d'environ 16GO et les données de test de 33GO. Sans manipulation préalable des données, il sera très difficile de passer en mémoire vive ces données et donc de travailler sur ces données. Dans les discussions Kaggle ce sujet revient régulièrement et une solution a retenu mon attention : la modification du type des variables pour diminuer la précision et donc la ressource mémoire nécessaire. Cela consiste par exemple à convertir les types « float64 » en « float16 ». Bien que cela implique une perte de précision, cela permet de diminuer significativement la taille de la base de données (1.7GO et 3.3GB pour le jeu d'apprentissage et le jeu de test respectivement). Parmi les auteurs ayant proposés cette solution, nous utiliserons dans ce projet les données transformées par @RADDAR et téléchargeable ici. Les bases de données sont stockées en parquet, un format binaire accélérant le chargement des données.

III. Exploration

L'exploration des données a été effectuée en utilisant le notebook proposé par @AMBROSM, qui a été légèrement modifié. Il est possible d'accéder à mon notebook grâce à ce lien. Comme discuté précédemment, les données sont chargées en format parquet à partir des modifications apportées par @RADDAR.

L'exploration des labels mets en évidence qu'aucun client est présent deux fois dans la base de données, et que tous les labels sont correctement renseignés (0 ou 1). Environ 25% des clients ont un label 1, correspondant à un défaut de paiement. Ces labels sont combinés à la base de données d'apprentissage contenant l'ensemble des variables. Elle contient 190 variables (en plus des labels)

anonymisées et normalisées appartenant à différentes catégories. Le détail sur le nom de chaque variable est disponible sur la page de la compétition. La base de données contient 5531451 lignes, correspondant à plusieurs opérations pour 458913 clients. Chaque client est caractérisé par un nombre d'opérations compris entre 1 et 13. La majorité des clients ont 13 opérations renseignées. Une moyenne de toutes les opérations pourra être envisagée pour caractériser plus simplement chaque client.

Parmi les 190 variables, 67 contiennent des valeurs nulles. Certaines d'entre elles sont même principalement non renseignées. 17 variables sont renseignées moins de 50% du temps, il pourra être envisagé de supprimer ces variables.

En s'intéressant aux dates des relevés, on s'aperçoit que les périodes couvertes par le jeu d'apprentissage et le jeu de test sont différentes, et donc que la date en tant que telle ne devra pas être utilisée. Par contre la période entre le premier et le dernier relevé sont similaires entre les bases de données, et centrée autour d'un an.

La description des données nous informe qu'elle contient 11 variables catégorielles. La représentation des valeurs de ces variables en fonction du label met en évidence que la majorité des ces variables contiennent de l'information pour distinguer le label, même si certaines semblent moins discriminantes. De plus, chaque variable contient maximum 8 valeurs différentes, et il est donc possible d'appliquer un one-hot encodeur. Concernant les deux variables binaires, elles ne semblent pas utiles, elles prennent toujours la même valeur ou non sont pas renseignées.

Pour finir, concernant les variables continues, elles sont au nombre de 175, avec des formes de distributions différentes. Certaines semblent pouvoir être considérées comme catégorielles puisqu'elles sont réparties en groupes. De même, sur certaines distributions, il semble possible de distinguer des outliers. Cependant nous ne sommes pas sûr et ces observations peuvent être des événements rares, contenant donc de l'information.

IV. Modélisation

L'ensemble du travail de modélisation a été effectué dans un notebook séparé, accessible ici.

La méthode envisagée pour obtenir une performance acceptable est d'utiliser un traitement de données proposé par un autre utilisateur, et de procéder à un moyennage de modèles forts afin de réduire l'erreur. Dans le but d'augmenter le nombre de modèles et donc d'espérer diminuer l'erreur, le traitement de données sera également modifié afin de créer d'autres modèles.

4.1 Traitement de données

Le traitement des données sur ce type de base de données est complexe et peut influencer fortement le résultat final. Dans ce projet le choix a été fait le traitement proposé par @huseyincot qui semble permettre d'obtenir de bonnes performances. Le principe de ce traitement est assez simple. Les variables catégorielles et continues sont traitées différemment. Le but est de combiner l'ensemble des relevés de chaque client afin de créer une seule ligne par client. Pour cela, les données continues sont agrégées en calculant la moyenne, la déviation standard, la valeur minimale, maximale et la dernière valeur. Cela signifie que chaque variable engendre 5 variables dans la base de données traitées. Concernant les variables catégorielles, les données sont agrégées en calculant le nombre d'entrées, la valeur de la dernière entrée, ainsi que le nombre de valeurs uniques possibles. Le nombre de lignes sera donc fortement réduit mais le nombre de variables augmente fortement.

Afin de modifier ce traitement, et donc de créer d'autres modèles légèrement différents, nous l'avons appliqué à la base de données après avoir supprimé les variables étant peu renseignées. En effet, supprimer ces variables ne devrait pas fortement détériorer les performances, mais peu modifier les probabilités de chaque classe prédite par le modèle, ce que nous cherchons à faire lorsque l'on assemble des modèles. Concrètement, les variables avec plus de 20% de valeurs manquantes sont supprimées, ce qui représente 21 variables.

Une dernière méthode a été utilisée dans ce projet est d'appliquer une sélection de variables. Celle-ci consiste à traiter la totalité de la base de données, et d'appliquer une méthode de sélection pour supprimer les variables les moins importantes. Pour cela on utilise un modèle random forest, pré entrainé sur la totalité des données. Ensuite on supprime chaque variable une a une, et on réentraîne le modèle avant de comparer les performances par rapport au modèle initial. On conserve uniquement les variables avec un résultat positif. Compte tenu du temps de calcul nécessaire, cette sélection a été réalisée sur 1/400 des clients.

Pour chaque traitement on sauvegarde la liste des variables utilisée dans un fichier pickle afin de pouvoir être récupérée facilement, notamment lors de la phase de prédiction.

4.2 Création de modèle

Pour chaque traitement présenté précédemment, deux types de modèles sont entrainés : un XGBoost et un CatBoost. Compte tenu de la taille de la base de données, il est impossible d'entrainer les modèles avec la totalité des données. On sépare donc les données en 5 folds, entrainés séparément avec une partie validation. Pour chaque prétraitement, nous obtenons donc 10 modèles (5 XGBoost et 5 CatBoost). La prédiction finale sera donc obtenue par moyenne des 5 modèles. Les modèles entrainés sont également sauvegardés afin de pouvoir être réutilisées pour la prédiction sur le jeu de test.

V. Prédiction

La tache de soumission des prédictions a été réalisée sur un notebook séparé, accessible ici.

L'ensemble des modèles est conservé dans un répertoire permettant d'y accéder facilement et rapidement. Les listes des variables utilisées pour chaque type de traitement sont chargées. La performance de chaque modèle est calculée sur le jeu d'apprentissage mais l'ensemble des modèles sont en léger surapprentissage sur les données d'apprentissage, ne permettant pas de définir le meilleur jeu de modèles.

On décide donc d'effectuer la prédiction sur le jeu de test directement et d'effectuer une soumission pour obtenir le score sur celui-ci. On notera que le jeu de test a été séparé en 5 afin de pouvoir être traiter sans soucis de mémoire. Pour cela, la prédiction est effectuée sur la totalité ou une sélection de modèles avant de combiner les résultats et de les moyenner. En effet, la fonction moyenne ne tient pas compte des valeurs nulles, et peut donc être directement appliquée sur un dataframe contenant l'ensemble de probabilité. Le fichier de soumission contient l'identifiant du client et la probabilité qu'il appartienne à la classe 1 (nombre compris entre 0 et 1).

Plusieurs soumissions ont été réalisées en utilisant une partie ou la totalité de modèle pour observer l'effet sur le score. On a tout d'abord réalisé une soumission rapide, avec des modèles appris rapidement avec un learning rate de 0.1. Ce score a été amélioré en utilisant un learning rate de 0.05. Puisque les modèles XGBoost semblent plus performant que les modèles CatBoost, seules les modèles XGBoost ont été moyennés, avec la totalité des features. Cependant le score est moins élevé qu'en combinant les modèles XGBoost et CatBoost. Une soumission a également été effectuée en n'utilisant pas les modèles avec une sélection de features, puisque ces modèles ne semblent pas aussi performants. Cependant le meilleur score a été obtenue en combinant l'ensemble des modèles avec l'ensemble des traitements, avec un score sur le jeu de test publique de 0.79435. Ce score est amélioré par rapport à ce qui a été proposé dans le notebook de référence avec un score de 0.793. Le score que l'on a obtenu sur le jeu de test privé est de 0.80214, ce qui nous place 2986ème sur 4876. Le gagnant de la compétition a obtenu un score de 0.80977.

VI. PEP8

Dans ce projet il est demandé de respecter les standards PEP8. Ces standards sont des bonnes pratiques pour coder, ce qui est important lorsque le code peut avoir vocation à être partagé, ce qui est le cas ici. Les standards PEP8 servent notamment à standardiser les noms de variables, la mise en page du code, et demande à documenter son code notamment. Différents programmes sont disponibles pour évaluer son code, afin de mettre en évidence les éléments ne respectant pas les

standards. Parmi ceux-ci, j'ai ici utilisé pylint sur le fichier py qui a l'avantage de donner une note finale à notre code. Les notes obtenues sont 9.53, 9.77 et 9.62 sur 10 pour le notebook d'exploration de modélisation et de soumission respectivement. Les erreurs persistantes, ne permettant pas d'obtenir la note maximale, sont un nommage incorrect du notebook et l'absence de documentation du module.

VII. Conclusions

Lors de ce projet j'ai participé à ma première compétition Kaggle, ce qui m'a permis de découvrir ce monde de la compétition. Le choix a été fait de baser mon travail sur ce qui a été proposé par d'autres compétiteurs afin d'améliorer les performances. Pour cela j'ai décidé de combiner différentes méthodes de traitements de données et différents types de modèles afin de faire une moyenne de l'ensemble des prédictions. Cette stratégie a permis d'améliorer légèrement les performances, en passant d'un score de 0.793 à 0.79435. Cette amélioration est faible mais à mettre en parallèle des meilleurs scores obtenus dans la compétition de 0.802. De mon point de vue, compte tenu de la taille de la base de données, je pense que pour obtenir des performances supérieures il est nécessaire de travailler sur des notebooks avec d'avantage de RAM.

Les codes développés dans ce projet respectent tous les standards PEP8, ce qui est important lorsque l'on souhaite partager ses travaux.

Lors de ma prochaine compétition, je pense qu'il serait important de rejoindre la compétition plus tôt afin d'avoir plus de temps pour développer des modèles. Également, il est important de choisir des compétitions avec des bases de données de tailles plus raisonnable pour éviter cette problématique de mémoire. Il me semble que la stratégie d'assembler des modèles forts, issus de différent traitements de données est une bonne stratégie pour améliorer les performances dans le cadre d'une compétition.