EPITA - Promo 2025

7 Juin 2024

Rapport projet MACH-BDA-DATAVIZ

Benjamin Aupest

Louis Escamilla

Vincent Tardieux

Table des matières

1 Introduction et présentation du projet

Le but du projet est d'analyser les **données** des clients d'un service d'abonnement internet afin de concevoir un modèle pour prédire le risque de départ des clients et ainsi prévoir des actions empêcher cela d'arriver.
On utilise ici le jeu de données "**customer_churn**" disponible sur HuggingFace.
Pour cela, on entraînera un modèle après avoir filtré les données puis on analysera les résultats obtenus.

2 Filtrage et Pré-traitement des données

2.1 Valeurs aberrantes

A l'aide de Spark, on commence par retirer les lignes incomplètes : c'est-à-dire les lignes contenant des champs **"null"**, **"Error"**, etc...
Puis on filtre les valeurs incohérentes selon leur contexte, par exemple : un temps ou un nombre de jours négatifs.
Enfin on retire les colonnes jugées inutiles pour l'entraînement du modèle (genre, numéro de sécurité, etc...) ainsi que celles qui ont une corrélation trop faible avec le **churn_risk_score**.

2.2 Normalisation

On souhaite également pouvoir interpréter les valeurs **non numériques** comme les feedbacks ou la formule d'abonnement. On a essayé d'utiliser la fonction **"OneHotEncoder()"** qui permet de remplacer les différentes entrées d'une colonne en un vecteur avec pour valeurs 0 ou 1.
Cependant nous avons obtenus des résultats inférieurs, on a donc **changé d'approche** en remplaçant directement les strings en valeurs numériques par exemple : on remplace les champs "non" et "oui" par 0 ou 1, de même pour les **feedbacks** une fois qu'on les labélise en bon/mauvais. Pour les **formules d'abonnement** : on les classe en fonction de leurs avantages de 0 pour "Pas d'abonnement" à 5 pour "Abonnement Premium".
On va également appliquer un **"MinMaxScaler()"** pour éviter d'avoir de grands nombres qui déséquilibreraient nos données.

3 Entraînement du modèle

3.1 Premiers résultats

On a d'abord tenté d'entraîner un modèle de **régression logistique** sur le jeu de données **non filtré** et uniquement avec les données numériques, afin d'avoir un premier repère. Nous avions obtenu un score d'accuracy de **0,70**.
Nous avons ensuite essayé avec le jeu de données filtré avec le **OneHotEncoding()** mais nous avons obtenu un résultat inférieur à notre premier test (**0,64**) d'où notre changement d'approche.

3.2 Comparaison des performances

Avec notre nouveau jeu de données, on va étudier la **corrélation** entre le churn_risk_score et les autres variables pour garder uniquement dans le jeu de données celles qui sont les plus corrélées. On obtient les résultats suivants :

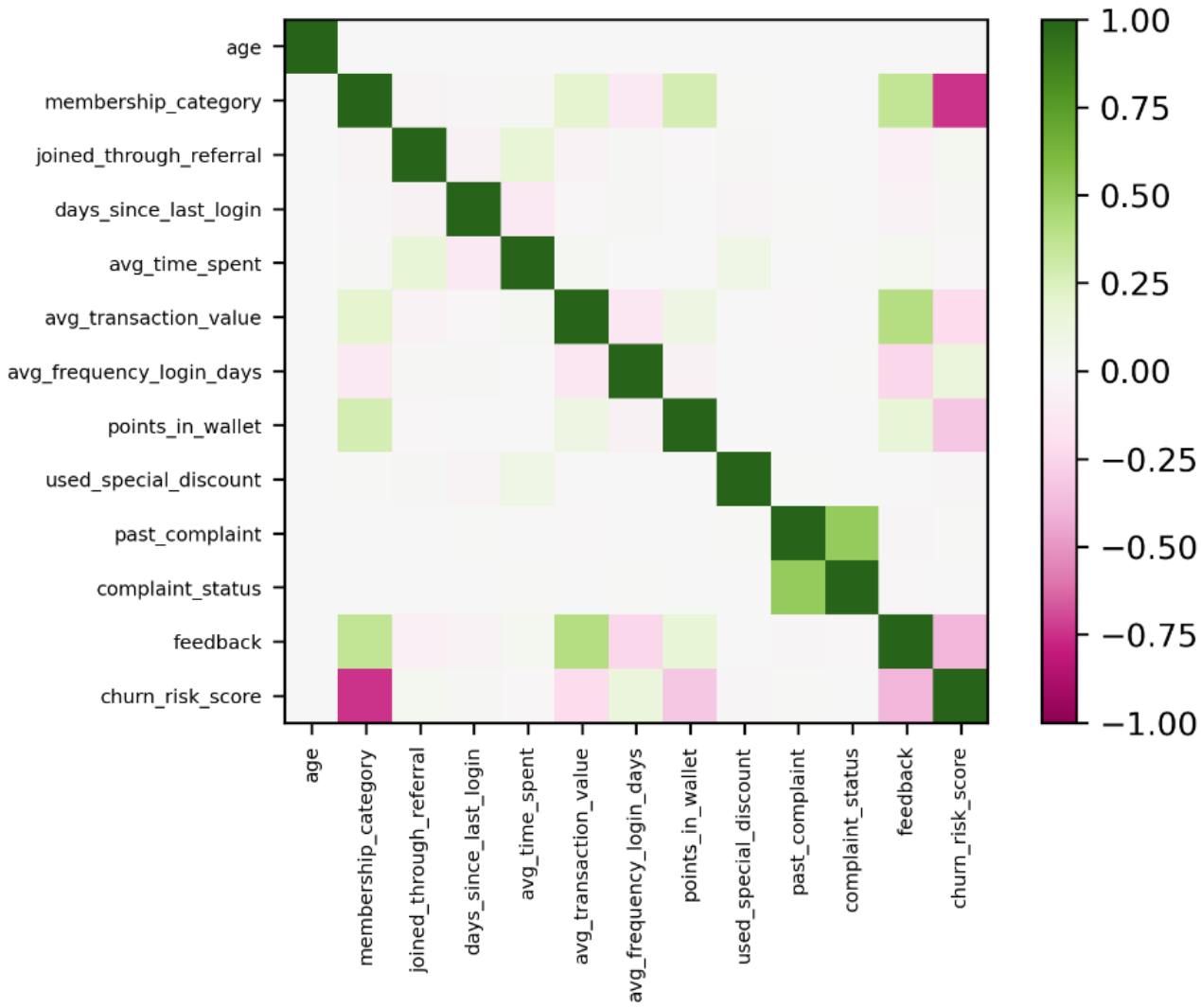

Figure 1 – *Matrice de corrélation du jeu de données*

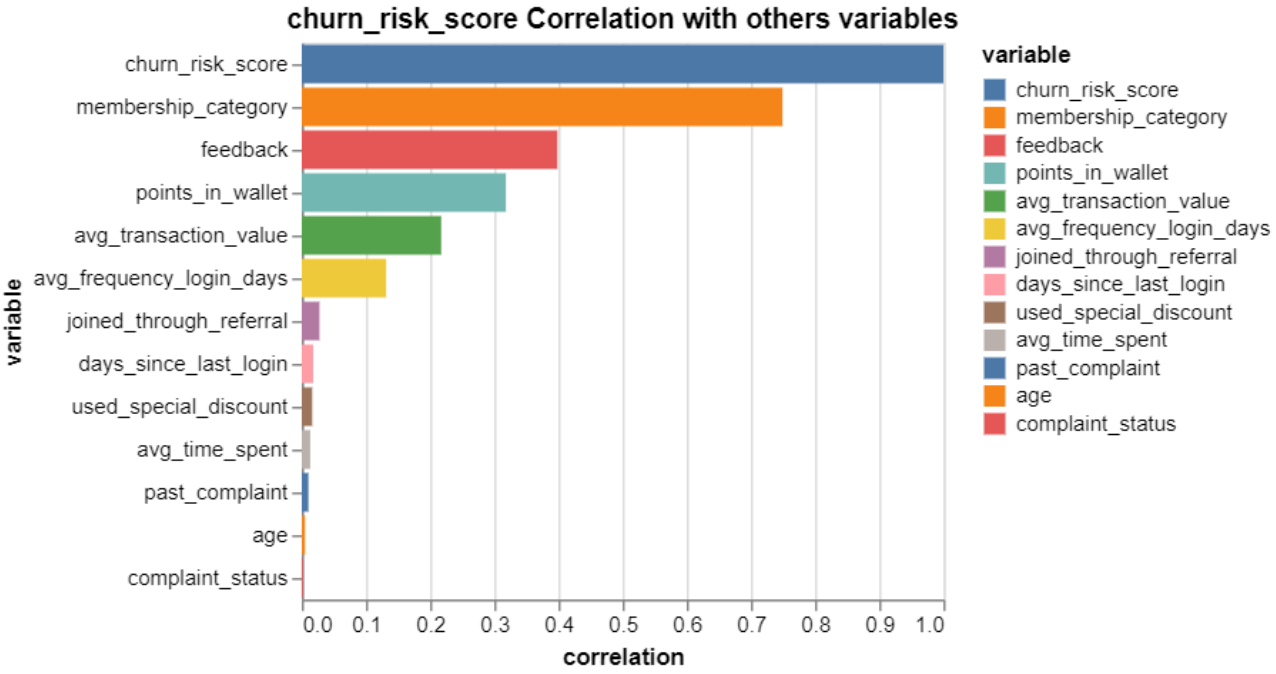

Figure 2 – *corrélation entre le churn_risk_score et les autres variables*

Toutes les autres données non numériques n'ont pas de lien avec le churn_risk_score. Exemple avec les types de région : (Voir le notebook visualization pour tous les visuels)

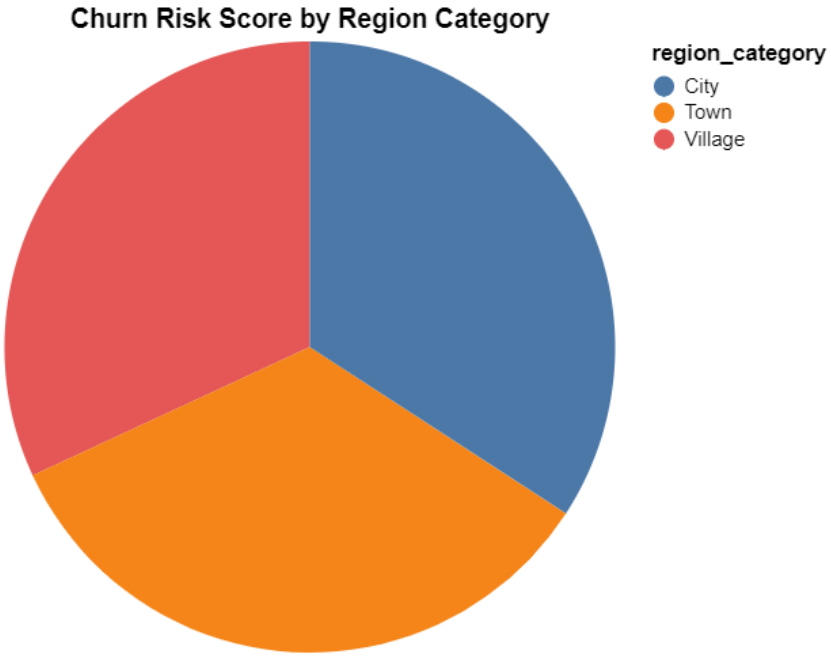

Figure 3 – *churn_risk_score en fonction du type de région*

Nous avons choisi d'étudier le score d'accuracy car il offre un bon compromis entre entre la **precision** et le **recall**, l'entreprise ne proposera donc pas trop de remises aux personnes qui n'en ont pas besoin.
Avec notre nouveau jeu de données, nous avons entraîné les modèles et obtenu les scores d'accuracy suivants :

- LogisticRegression : 0,85
- RandomForestClassifier : 0,94
- SVC : 0,84
- KNeighborsClassifier : 0,83
- DecisionTreeClassifier : 0,92

3.3 Fine-tuning

On a effectuer une **grid-search** avec les paramètres suivants :

- 'n_estimators' : [400, 500, 1000, 2000, 3000]
- 'max_depth' : [5, 10, 20, None]

afin d'affiner les hypers-paramètres ainsi que la **cross-validation** :

- n_splits=10
- random_state=42
- shuffle=True

qui va diviser le training set et adapter le modèle à chaque step.

Ainsi, notre RandomForestClassifier est monté à **0,9464** d'accuracy.

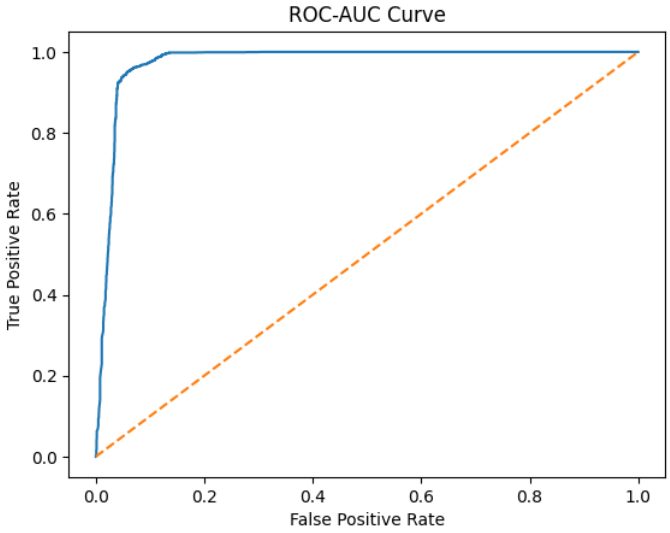

Figure 4 – *Roc curve de notre meilleur modèle*

4 Conclusion

Pour conclure, on a vu que les utilisateurs ont tendance à **rester** s'ils ont des formules d'abonnement avantageuses, s'ils donnent des bons feedbacks, s'ils ont beaucoup de points ou s'ils se connectent régulièrement.
Notre meilleur modèle est un random forest avec pour hyperparamètres changés : n_estimators = 3000 et max_depth = None. La précision obtenue est de **0,9464** ce qui est très **satisfaisant**.
Les prédictions pourront être utilisées par le fournisseur d'accès internet pour envoyer des **offres** spéciales aux clients qui risquent de partir. Pour rappel, nous avons choisi de maximiser la **précision** afin de ne pas proposer des offres à des clients qui seraient de toute façon restés et donc **perdre** de l'argent inutilement.

4.1 Pistes de réflexion

Nous pensions que certaines **variables** seraient corrélées avec le churn_risk_score mais en réalité elle ne le sont **pas**. On pense par exemple à la plateforme utilisée ou à l'option internet, ou encore à l'évolution de celle-ci au cours du temps.
C'est plutôt **bon signe** car les utilisateurs ne s'en vont donc pas à cause d'une mauvaise application smartphone ou un mauvais wi-fi ni à cause du service qui devient obsolète avec le temps.

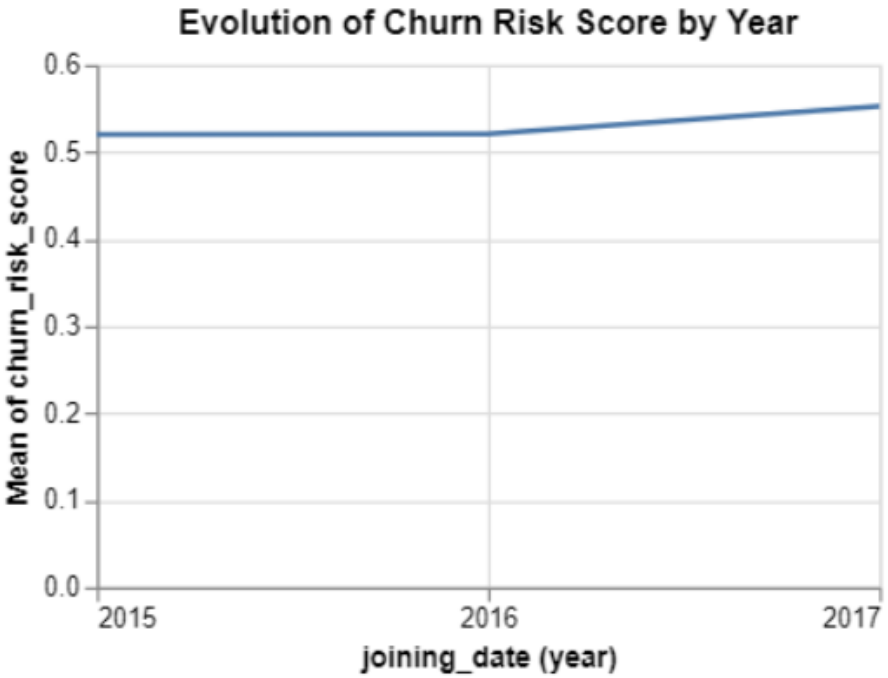

Figure 5 – *Evolution du churn_risk_score au fil du temps*

Il serait donc avantageux pour l'entreprise de **cibler** les utilisateurs susceptibles de s'en aller grâce aux critères mentionnés ci-dessus, puis de leur proposer des **remises** ou des promotions.