Prêt à dépenser

Implémenter un modèle de scoring

Marwan BOUGHZALA

Juillet 2020

Sommaire

1. Introduction

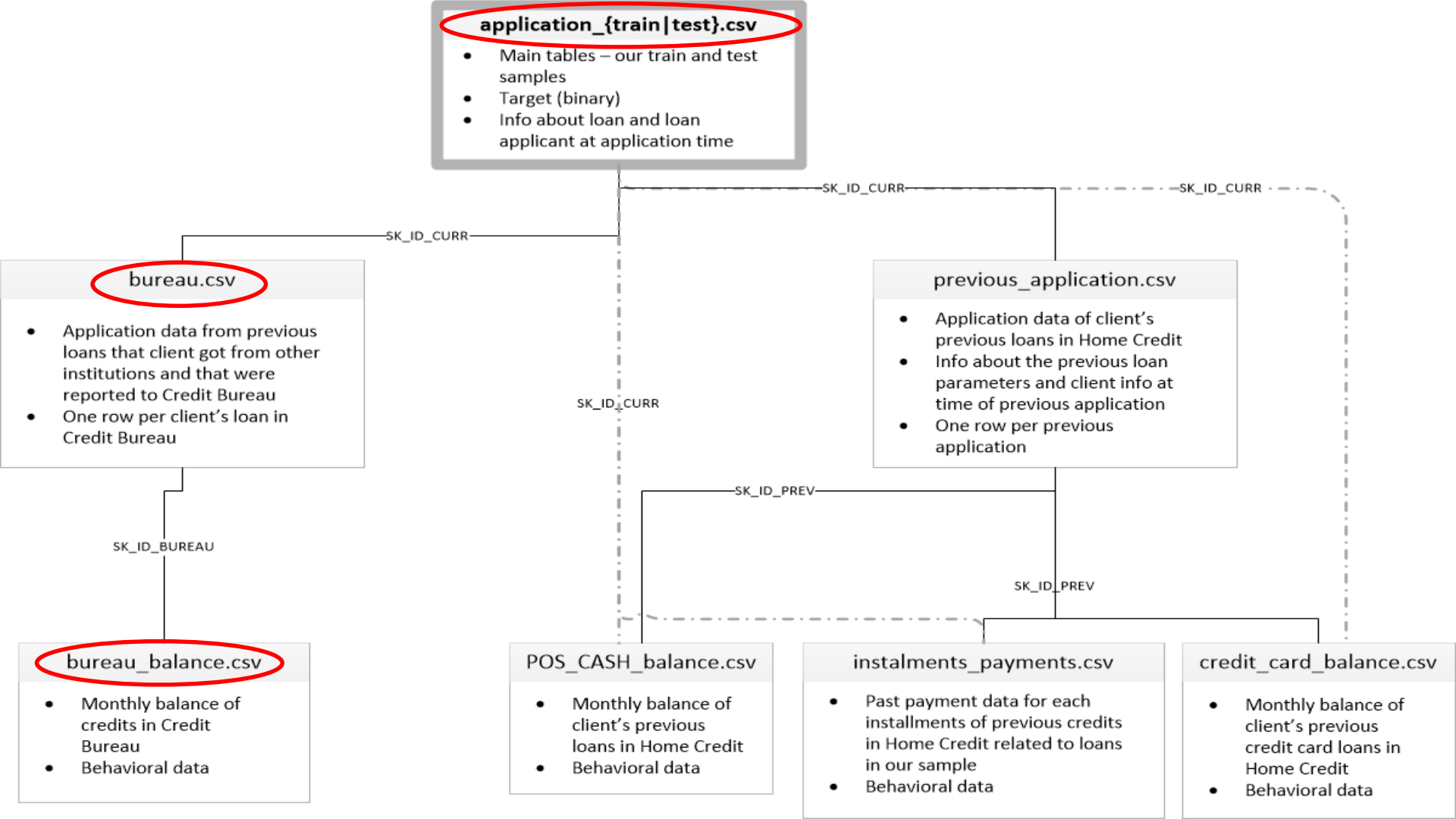
Rappel de la problématique :

- L’entreprise souhaite **développer un modèle de scoring de la probabilité de défaut de paiement du client** pour étayer la décision d'accorder ou non un prêt à un client
- Elle décide donc de développer un **dashboard** interactif
- De plus, les chargés de relation client ont fait remonter le fait que les clients sont de plus en plus demandeurs de **transparence**

1. Introduction

Les données :

- 8 fichiers au format CSV dont 1 fichier « train » et 1 fichier « test » :
  - fichier « train » utilisé pour l'entrainement du modèle
  - fichier « test » utilisé pour simuler les clients réels
- Les fichiers sont sous formes de base de données avec des clés
- 4 fichiers utilisés dans le projet :
  - Application_train.csv
  - Application_test.csv
  - Bureau.csv
  - Bureau_balance.csv


application_{train|test}.csv
Main tables – our train and test samples
Target (binary)
Info about loan and loan applicant at application time
SK_ID_CURR
SK_ID_CURR
SK_ID_CURR
bureau.csv
Application data from previous loans that client got from other institutions and that were reported to Credit Bureau
One row per client's loan in Credit Bureau
SK_ID_CURR
previous_application.csv
Application data of client's previous loans in Home Credit
Info about the previous loan parameters and client info at time of previous application
One row per previous application
SK_ID_PREV
SK_ID_BUREAU
SK_ID_PREV
bureau_balance.csv
Monthly balance of credits in Credit Bureau
Behavioral data
POS_CASH_balance.csv
Monthly balance of client's previous loans in Home Credit
Behavioral data
instalments_payments.csv
Past payment data for each installments of previous credits in Home Credit related to loans in our sample
Behavioral data
credit_card_balance.csv
Monthly balance of client's previous credit card loans in Home Credit
Behavioral data

1. Introduction

Les données :

- Réutilisation d'un kernel de Kaggle avec :
  - Analyse exploratoire,
  - Nettoyage des données
  - Feature engineering
- https://www.kaggle.com/willkoehrsen/introduction-to-manual-feature-engineering
- Données en sortie :
  - Fichier train avec TARGET : (307 511, 199)
  - Fichier test : (48 744, 198)

1. Introduction

Outils utilisés

2. Modélisation

2. Modélisation

1) Equilibre des données

Variable cible :
1 – Client ayant des difficultés de paiement
0 – Client n'ayant pas eu de difficultés de paiement

0 282 686 / 92%
1 24 825 / 8%

- Déséquilibre de la variable cible
- Majorité de « bons clients »
- Création d'une baseline

Rappel
Accuracy : Précision du modèle
Precision : Performance du modèle quand celui-ci déclare une classe.
Recall : Pourcentage de détection des classes.
F1_score : Moyenne harmonique de la précision et du rappel.

2. Modélisation

2) Dummy classifier

- Train test split : Utilisation du paramètre « stratify »
- Stratégie utilisée : most_frequent ➔ Toujours prédire le label le plus présent dans le jeu de donnée
- Résultats :

	precision	recall	f1-score	support
0	0.92	1.00	0.96	84806
1	0.00	0.00	0.00	7448
accuracy			0.92	92254
macro avg	0.46	0.50	0.48	92254
weighted avg	0.85	0.92	0.88	92254

2. Modélisation

3) Gradient boosting

- Résultats :

	precision	recall	f1-score	support
0	0.92	1.00	0.96	84806
1	0.51	0.04	0.08	7448
accuracy			0.92	92254
macro avg	0.72	0.52	0.52	92254
weighted avg	0.89	0.92	0.89	92254

- Equilibrage des classes indispensables
- Utilisation du paramètre « class_weight » : le mode « balanced » utilise les valeurs de y pour ajuster automatiquement les poids inversement proportionnels aux fréquences de classe dans les données d'entrée

2. Modélisation

4) Métrique personnalisée

- Utilisation d'une métrique adapté à la problématique
- 4 possibilités de classements : FN, FP, TN , TP
- Gain totale = FN + FP + TP + TN
- Association de coefficients (arbitraire) à chaque variable :
  - FN_coeff = -10 #Mauvais clients non identifiés ➔ Perte
  - FP_coeff = 0 #Bons clients non identifiés ➔ Manque à gagner
  - TP_coeff = 0 #Mauvais clients identifiés ➔ Argent sauvé
  - TN_coeff = 1 #Bons clients non identifiés ➔ Bénéfice
- Gain totale = -10*FN + 0*FP + 0*TP + 1*TN = **TN – 10 FN**
- **G_normalized = (G - G_min)/(G_max - G_min)**

2. Modélisation

4) Métrique personnalisée

- Résultats :

```
1   Lgb_tun.best_score_
```

```
defaultdict(dict,
            {'train': {'binary_logloss': 0.5322494990826578,
              'credit_metric': 0.7323987622763353},
             'valid': {'binary_logloss': 0.5515642157906153,
              'credit_metric': 0.695848976055648 3}})
```

TN FP

```
([[60979, 23827],
  [ 2462,  4986]])
```

FN TP

	precision	recall	f1-score	support
0	0.96	0.72	0.82	84806
1	0.17	0.67	0.28	7448
accuracy			0.72	92254
macro avg	0.57	0.69	0.55	92254
weighted avg	0.90	0.72	0.78	92254

2. Modélisation

5) Random search

```
param_test ={
            'num_leaves': [10,50,100,1000],
            'min_child_samples': [100,250,500],
            'min_child_weight': [1e-5, 1e-1, 1, 1e1, 1e3],
            'reg_alpha': [0, 1e-1, 1, 5, 10, 50, 100],
            'reg_lambda': [0, 1e-1, 1, 5, 10, 50, 100]
            }
```

- Utilisation d'un échantillon réduit
- Recherches aléatoires d'hyperparamètres
- Dix différentes combinaisons de paramètres
- Utilisation de la métrique métier pour l'évaluation
- ~1h de recherche des meilleurs hyperparamètres
- Légère amélioration après comparaison des modèles avec et sans optimisations

3. Interprétabilité

3. Interprétabilité

1) Lime

- Modèle ➔ Boite noire
- Solution retenue : création d'un modèle d'approximation local linéaire
- **L**ocal **I**nterpretable **M**odel-agnostic **E**xplanations (LIME)
- Avantages :
  - Facile à comprendre et résultats clairs
  - Utilisation sur n'importe quel modèle
  - Explication des caractéristiques même transformé
- Désavantages :
  - Différentes explications entre 2 points très proches
  - Instabilité des explications sur plusieurs cycles
- Conclusion : Très bonne méthode mais encore en phase de développement

4. Dashboard Local

4. Dashboard API

4. Conclusion

- Amélioration du modèle avec :
  - Recherche sur grille
  - Utilisation des autres fichiers csv
  - Améliorer le feature engineering
  - Discuter de la métrique personnalisée avec des équipes du métier
  - Obtenir beaucoup plus de données de la classe en défaut

- Amélioration du dashboard :
  - Créer des fenêtres dynamiques

- Utilisation d’un serveur plus puissant

Merci !