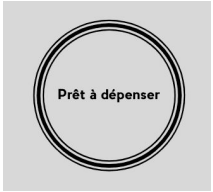


Implémentez un modèle de scoring

Guilhem Berthou - Pierre-Antoine Ganaye (mentor)

https://github.com/guilhembr/p7_scoring

Introduction

Contexte : La société financière **“Prêt à dépenser”**, propose des crédits à la consommation pour des personnes ayant peu ou pas du tout d'historique de prêt.

L’entreprise souhaite mettre en œuvre un outil de “**scoring crédit**” pour calculer la **probabilité qu’un client rembourse son crédit,** puis classifie la demande en **crédit accordé** ou **refusé**.

Traiter les **variables descriptives** d’un prospect

Réaliser une **classification** des clients

Présenter les **résultats** en toute **transparence**

⇒ **Interprétation :** Il s’agit d’un problème de **classification supervisée :**

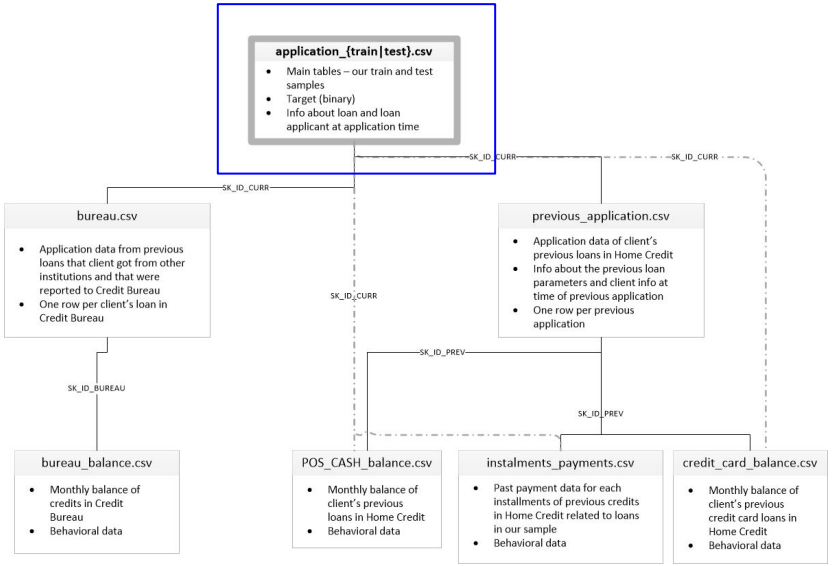
- *Supervisée* : Les données sont étiquetées dans le jeu de données d'entraînement `(application_train)` qui servira de base à l'entraînement d’un modèle afin de prédire la capacité de remboursement (le label) en fonction des features client.
- *Classification* : Le label est une variable binaire, 0 (le client rembourse son crédit), 1 (le client est insolvable).

Introduction

Jeu de données :

Le jeu de données contient 7 sources de données. Pour réaliser un premier prototype nous nous concentrons sur les 2 jeux principaux :

- *application_train/application_test* :
  - Contiennent les informations de chaque demande de prêt.
  - Chaque ligne des datasets correspond à une demande identifiée par `SK_ID_CURR.`
  - Le jeu d'entraînement contient les `TARGET`. A l'inverse le jeu de test n'est pas étiqueté.



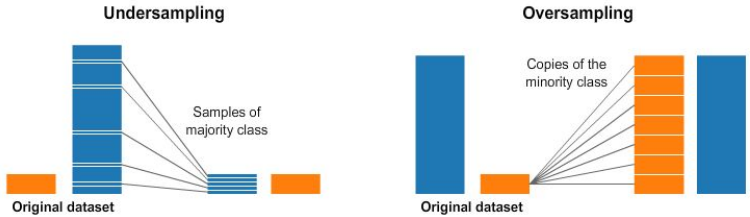
Sommaire

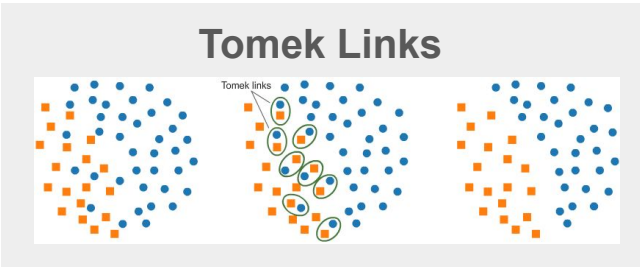
Sommaire

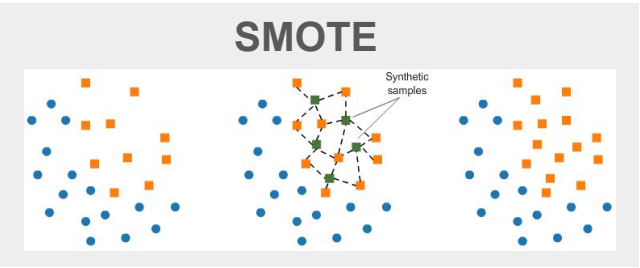
Présentation des enjeux

➢ Déséquilibre des classes

- **Enjeu du sujet :** Nous constatons tout d'abord qu'il s'agit d'un dataset dont les classes de la variable à prédire (`TARGET`) sont déséquilibrées. Il y a ainsi beaucoup plus de lignes correspondant à des remboursements qu'à des défauts.
- **Réponse apportée :**
  - *Option 1 :* **Undersampling / Oversampling**







Présentation des enjeux

➢ Déséquilibre des classes

- **Enjeu du sujet :** Nous constatons tout d'abord qu'il s'agit d'un dataset dont les **classes de la variable à prédire** (`TARGET`) sont ***déséquilibrées***. Il y a ainsi beaucoup **plus de lignes correspondant à des remboursements** qu'à des défauts.
- **Réponse apportée :**
  - *Option 2 :* **Hyperparamètre** `class_weights = balanced`
    - le modèle **ajuste automatiquement** le **nombre d'items** de chaque classe en les **pondérant** de manière inversement proportionnelle à la fréquence de leur classe :

```
n_samples / (n_classes * np.bincount(y))
```

⇒ *Nous avons obtenu les **meilleures performances** en adoptant cette seconde approche.*

Présentation des enjeux

➢ **Feature engineering**

- **Enjeu du sujet**
- **Réponse apportée**

⇒ *XX*

Présentation des enjeux

➢ **Feature selection**

- **Enjeu du sujet**
- **Réponse apportée**

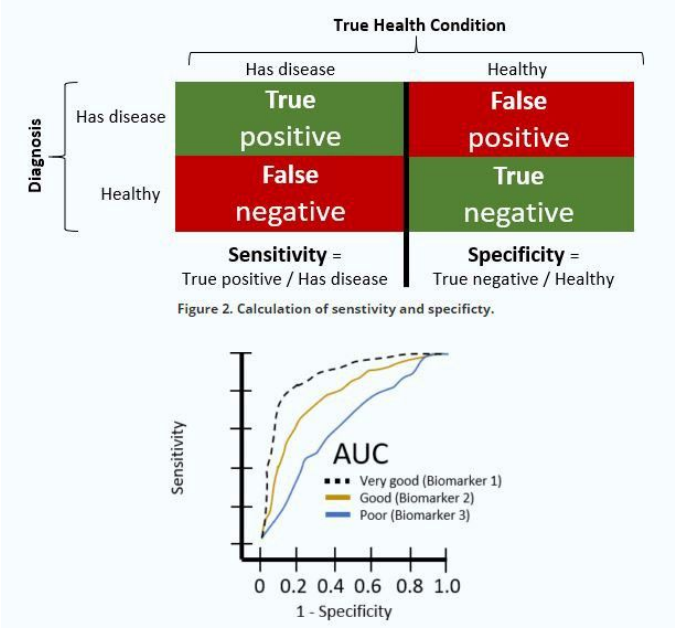
⇒ *XX*

Présentation des enjeux

➢ **Mesure de la performance du modèle**

- **Enjeu du sujet**
- **Réponse apportée**

⇒ *XX*



Présentation des enjeux

➢ **Optimisation du seuil de décision**

- **Enjeu du sujet**
- **Réponse apportée**

⇒ *XX*

Présentation des enjeux

➢ **Récapitulatif des arbitrages pris**

-

⇒ *XX*

Sommaire

Modélisation

➢ **Méthodologie d'entraînement**

- **Encodage**
- **Imputation**
- **Standardisation**

⇒ *XX*

Modélisation

- **Méthodologie d'entraînement**
  - **Recherche d'hyperparamètres via GridSearch CV**
    - **Scoring = roc_auc**

⇒ *XX*

Modélisation

➢ **Comparaison des résultats des modèles**

- Interprétation des métriques

⇒ *XX*

Modélisation

➢ Choix du modèle

⇒ *XX*

Sommaire

Interprétation des prédictions

- **Interprétation globale**
  - **Enjeu du sujet**
  - **Réponse apportée**

⇒ *XX*

Interprétation des prédictions

➢ **Interprétation locale**

- **Enjeu du sujet**
- **Réponse apportée**

⇒ *XX*

Sommaire

Dashboard

➢ **Architecture du dashboard**

- **Schema architecture + github**

⇒ *XX*

Dashboard

➢ Endpoints de l'API

⇒ *XX*

Dashboard

➢ Fonctionnalités du tableau de bord

-

⇒ *XX*

Sommaire

Conclusion

➢ Limites et améliorations

○ Modélisation

- Réduction de dimension (PCA) pour éviter la “curse of dimensionality”
- Amélioration des performances via des modèles plus complexes au dépend du temps de traitement (LGBM, XGBoost)

○ API

- Utilisation de paramètres de requête dans les URL pour une meilleure lisibilité
  - L’ID client devrait être fourni avec la syntaxe ?id=0
- Construction d’une base de donnée afin de permettre à l’utilisateur de filtrer sur les champs souhaités
  - Via une requête SQL dynamique (ex : SELECT * FROM data.db WHERE [*column_to_filter*] =?)
- Création d’un modèle Pydantic dynamique pour encadrer le format des inputs et le documenter automatiquement

○ Dashboard

- Permettre à l’utilisateur de filtrer sur plusieurs dimensions