



Implémentez un modèle de scoring

Guilhem Berthou - Pierre-Antoine Ganaye (mentor)

https://github.com/guilhembr/p7 scoring

Introduction

<u>Contexte</u>: 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**

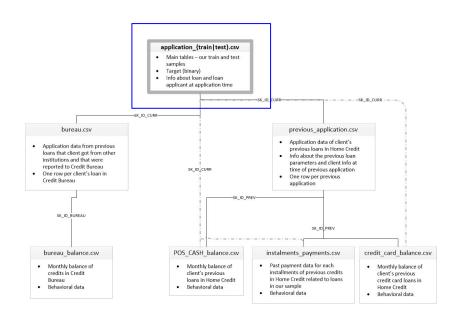
- ⇒ Interprétation : Il s'agit d'un problème de classification supervisée :
 - <u>Supervisée</u>: Les données sont étiquetées dans le jeu de données d'entraînement (application_train) qui servira de base à l'entraînement d'un modèle afin de prédire la capacité de remboursement (le label) en fonction des features client.
 - <u>Classification</u>: 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.



1. Présentation des enjeux

- a. Déséquilibre des classes (Options 1 vs Options 2)
- b. Feature-engineering (connaissance métier)
- c. Feature selection (RFECV)
- d. Mesure de la performance (metrics et make scorer / fonction coût métier)
- e. Optimisation du seuil de décision

2. Modélisation

- Méthode d'entraînement
 - i. Encodage, Imputation et Standardisation
 - i. Recherche des hyperparamètres via GridSearchCV
- b. Comparaison des résultats des modèles
 - i. Lecture des métriques et de la matrice de confusion
 - ji. Justification du choix du modèle

3. Interprétation

- a. Interprétation globale des coefficients du modèle
- b. Interprétation locale des items (calcul des valeurs de Shapley)

4. Dashboard

- a. Architecture (outils utilisés et structure)
- b. Endpoints de l'API
- c. Fonctionnalités du Dashboard

1. Présentation des enjeux

- a. Déséquilibre des classes (Options 1 vs Options 2)
- b. Feature-engineering (connaissance métier)
- c. Feature selection (RFECV)
- d. Mesure de la performance (metrics et make scorer / fonction coût métier)
- e. Optimisation du seuil de décision

2. Modélisation

- a. Méthode d'entraînement
 - Encodage, Imputation et Standardisation
 - i. Recherche des hyperparamètres via GridSearchCV
- b. Comparaison des résultats des modèles
 - Lecture des métriques et de la matrice de confusion
 - ji. Justification du choix du modèle

3. Interprétation

- a. Interprétation globale des coefficients du modèle
- b. Interprétation locale des items (calcul des valeurs de Shapley)

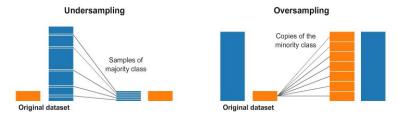
4. Dashboard

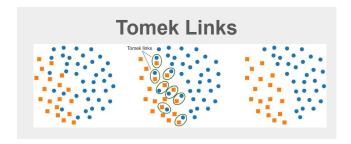
- a. Architecture (outils utilisés et structure
- b. Endpoints de l'AP
- Fonctionnalités du Dashboard

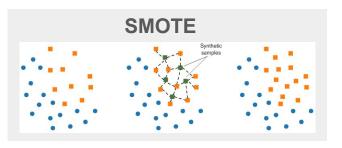
- Présentation des enjeux
 - a. Déséquilibre des classes
 - Feature-engineering
 - c. Feature selection
 - d. Mesure de la performance
 - e. Optimisation du seuil de décision

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
 - a. Déséquilibre des classes
 - b. Feature-engineering
 - c. Feature selection
 - d. Mesure de la performance
 - e. Optimisation du seuil de décision

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.

- Feature engineering
 - Enjeu du sujet
 - Réponse apportée



- Présentation des enjeux
 - Déséquilibre des classe
 - b. Feature-engineering
 - c. Feature selection
 - d. Mesure de la performance
 - e. Optimisation du seuil de décision

- Feature selection
 - Enjeu du sujet
 - Réponse apportée

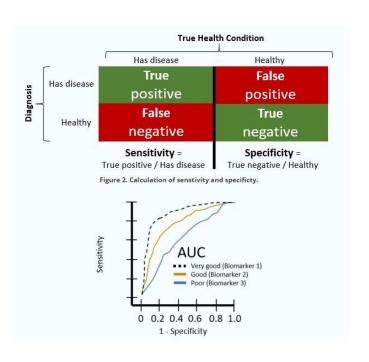


- Présentation des enjeux
 - Déséquilibre des classe
 - Feature-engineering
 - c. Feature selection
 - d. Mesure de la performance
 - Optimisation du seuil de décision

- Mesure de la performance du modèle
 - Enjeu du sujet
 - Réponse apportée



- Présentation des enjeux
 - Déséguilibre des classe
 - b. Feature-engineerin
 - c. Feature selection
 - d. Mesure de la performance
 - e. Optimisation du seuil de décision



- Optimisation du seuil de décision
 - Enjeu du sujet
 - Réponse apportée



- Présentation des enjeux
 - Déséquilibre des classes
 - Feature-engineering
 - c. Feature selection
 - d. Mesure de la performance
 - e. Optimisation du seuil de décision

Récapitulatif des arbitrages pris

 \circ



- Déséquilibre des classe
- Feature-engineering
- c. Feature selection
- Mesure de la performance
- Optimisation du seuil de décision

1. Présentation des enjeux

- a. Déséquilibre des classes (Options 1 vs Options 2)
- b. Feature-engineering (connaissance métier)
- c. Feature selection (RFECV)
- d. Mesure de la performance (metrics et make scorer / fonction coût métier)
- e. Optimisation du seuil de décision

2. Modélisation

- a. Méthode d'entraînement
 - i. Encodage, Imputation et Standardisation
 - . Recherche des hyperparamètres via GridSearchCV
- b. Comparaison des résultats des modèles
 - i. Lecture des métriques et de la matrice de confusion
 - Justification du choix du modèle

3. Interprétation

- a. Interprétation globale des coefficients du modèle
- b. Interprétation locale des items (calcul des valeurs de Shapley)

4. Dashboard

- a. Architecture (outils utilisés et structure
- b. Endpoints de l'AP
- Fonctionnalités du Dashboard

- Méthodologie d'entraînement
 - Encodage
 - Imputation
 - Standardisation



2 Modélisation

. Méthode d'entraînement

. Encodage, Imputation et Standardisation

- ii. Recherche des hyperparamètres via GridSearchCV
- Comparaison des résultats des modèles
 - Lecture des métriques et de la matrice de confusion
 - Justification du choix du modèle

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



2 Modélisation

Méthode d'entraînement

- Encodage, Imputation et Standardisation
- ii. Recherche des hyperparamètres via GridSearchCV
- b. Comparaison des résultats des modèles
 - Lecture des métriques et de la matrice de confusion
 - ii Justification du choix du modèle

- Comparaison des résultats des modèles
 - Interprétation des métriques



2 Modélisation

- Méthode d'entraînement
 - Encodage, Imputation et Standardisation
 - ii. Recherche des hyperparamètres via GridSearchCV
- b. Comparaison des résultats des modèles
 - i. Lecture des métriques et de la matrice de confusion
 - Justification du choix du modèle

Choix du modèle



2 Modélisation

- Méthode d'entraînement
 - Encodage, Imputation et Standardisation
 - Recherche des hyperparamètre via GridSearchCV

. Comparaison des résultats des modèles

- . Lecture des métriques et de la matrice de confusion
- i. Justification du choix du modèle

1. Présentation des enjeux

- a. Déséquilibre des classes (Options 1 vs Options 2)
- b. Feature-engineering (connaissance métier)
- c. Feature selection (RFECV)
- d. Mesure de la performance (metrics et make scorer / fonction coût métier)
- e. Optimisation du seuil de décision

2. Modélisation

- a. Méthode d'entraînement
 - Encodage, Imputation et Standardisation
 - Recherche des hyperparamètres via GridSearchCV
- b. Comparaison des résultats des modèles
 - Lecture des métriques et de la matrice de confusion
 - ji. Justification du choix du modèle

3. Interprétation

- a. Interprétation globale des coefficients du modèle
- b. Interprétation locale des items (calcul des valeurs de Shapley)

4. Dashboard

- a. Architecture (outils utilisés et structure)
- b. Endpoints de l'AP
- Fonctionnalités du Dashboard

Interprétation des prédictions

- Interprétation globale
 - Enjeu du sujet
 - Réponse apportée



3. Interprétation

- a. Interprétation globale des coefficients du modèle
- Interprétation locale des items (calcul des valeurs de Shapley)

Interprétation des prédictions

- Interprétation locale
 - Enjeu du sujet
 - Réponse apportée



3. Interprétation

- Interprétation globale des coefficients du modèle
- Interprétation locale des items (calcul des valeurs de Shapley)

1. Présentation des enjeux

- a. Déséquilibre des classes (Options 1 vs Options 2)
- b. Feature-engineering (connaissance métier)
- c. Feature selection (RFECV)
- d. Mesure de la performance (metrics et make scorer / fonction coût métier)
- e. Optimisation du seuil de décision

2. Modélisation

- a. Méthode d'entraînement
 - Encodage, Imputation et Standardisation
 - Recherche des hyperparamètres via GridSearchCV
- Comparaison des résultats des modèles
 - Lecture des métriques et de la matrice de confusion
 - ji. Justification du choix du modèle

3. Interprétation

- a. Interprétation globale des coefficients du modèle
- b. Interprétation locale des items (calcul des valeurs de Shapley)

4. Dashboard

- a. Architecture (outils utilisés et structure)
- b. Endpoints de l'API
- c. Fonctionnalités du Dashboard

Dashboard

- Architecture du dashboard
 - Schema architecture + github



3. Dashboard

- a. Architecture (outils utilisés et structure)
- Endpoints de l'API
- . Fonctionnalités du Dashboard

Dashboard

➤ Endpoints de l'API



3. Dashboard

- Architecture (outils utilisés et structure
- b. Endpoints de l'API
- Fonctionnalités du Dashboard

Dashboard

> Fonctionnalités du tableau de bord

0



3. Dashboard

- Architecture (outils utilisés et structure
- . Endpoints de l'API
- Fonctionnalités du Dashboard

1. Présentation des enjeux

- a. Déséquilibre des classes (Options 1 vs Options 2)
- b. Feature-engineering (connaissance métier)
- c. Feature selection (RFECV)
- d. Mesure de la performance (metrics et make scorer / fonction coût métier)
- e. Optimisation du seuil de décision

2. Modélisation

- a. Méthode d'entraînement
 - Encodage, Imputation et Standardisation
 - Recherche des hyperparamètres via GridSearchCV
- b. Comparaison des résultats des modèles
 - Lecture des métriques et de la matrice de confusion
 - ji. Justification du choix du modèle

3. Interprétation

- a. Interprétation globale des coefficients du modèle
- b. Interprétation locale des items (calcul des valeurs de Shapley)

4. Dashboard

- a. Architecture (outils utilisés et structure)
- b. Endpoints de l'API
- c. Fonctionnalités du Dashboard

Conclusion 5. 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)

o 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