Parcours : Data Scientist

Projet7: Implémentez un modèle de scoring

Asma kouaouci

Plan

- **Présentation de la problématique**
- **Jeu de donnée**
- **Explication de l'approche de modélisation**
  - Les modèles
  - la métrique d'évaluation
  - le modèle finale
- **Présentation du dashboard**

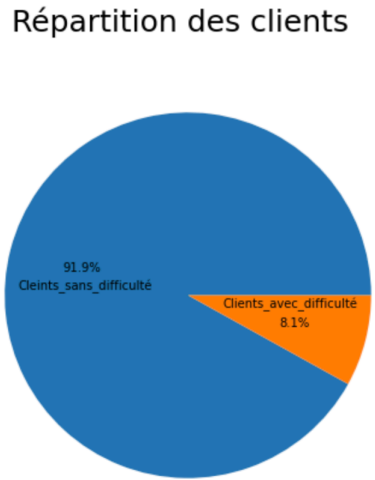
1. Présentation de la problématique

- **Prêt à dépenser:**
  - Société financière qui propose des crédits à la consommation pour des personnes ayant peu ou pas du tout d'historique de prêt
  - Pour l'instant, la décision d'accorder ou non un prêt est effectuée manuellement par les chargés de clientèles

(Black Friday,E-commerce) La crise sanitaire de l'année 2020 a amplifié les changements observés

- **Objectifs :**
  - L'entreprise souhaite développer un modèle de scoring de la probabilité de défaut de paiement du client
  - La méthode la plus fluide possible est donc de développer un **dashboard interactif** pour notre équipe puissent expliquer de façon plus **transparente** possible d'accordé ou Pas le crédit

Données TARGET



Ce graphique montre la répartition des clients selon leurs remboursement

La variable de cible "TARGET " :

le non remboursement (TARGET = 1)
le remboursement à terme (TARGET= 0)

2 . Données imputation et encoding

Une table contenant des informations de nos anciens clients

Chaque ligne correspond à un client avec un identifiant distinct

Variables Quantitatives : Age, Ancienneté de l'emploi actuel, Montant de crédit demandé, Montant de revenu"

Traitement des variables : Remplacement des données manquantes par des valeurs médianes

Variables Catégorielles : Sexe, Statut familial, Niveau éducation etc. »

Traitement des variables :Remplacement des données manquantes par la valeur 'NC'

- LabelEncoding
- OneHotEncoding

Nom	Animal préféré
Jane	Chien
Maxime	Chat
Marie	Souris
Thomas	Chien

Dataset initial

Nom	Animal préféré
Jane	0
Maxime	1
Marie	2
Thomas	0

Dataset
Label Encoding

Nom	Animal préféré chien	Animal préféré chat	Animal préféré souris
Jane	1	0	0
Maxime	0	1	0
Marie	0	0	1
Thomas	1	0	0

Dataset
One Hot Encoding

3. Données variable polynomiale

Variables explicatives corrélées

	correlation	correlation_absolue
TARGET	1.000000	1.000000
EXT_SOURCE_2	-0.160295	0.160295
EXT_SOURCE_3	-0.155892	0.155892
EXT_SOURCE_1	-0.098887	0.098887
DAYS_BIRTH	-0.078239	0.078239

Corrélation avec “TARGET”

	TARGET	EXT_SOURCE_1	EXT_SOURCE_2	EXT_SOURCE_3	DAYS_BIRTH
TARGET	1.000000	-0.098887	-0.160295	-0.155892	-0.078239
EXT_SOURCE_1	-0.098887	1.000000	0.134993	0.109100	0.362185
EXT_SOURCE_2	-0.160295	0.134993	1.000000	0.094147	0.091947
EXT_SOURCE_3	-0.155892	0.109100	0.094147	1.000000	0.178527
DAYS_BIRTH	-0.078239	0.362185	0.091947	0.178527	1.000000

Corrélation entre les variables

Hypothèse: variables explicatives semblent avoir une interaction entre elles Afin de vérifier cette interaction, on décide d'introduire des variables de type polynomial

4. Données variable Crée (Variables supplémentaires)

Les données quantitatives initiales donnent une vision de volume mais ne permettent pas de bien comprendre le sens dans certains cas

```
app_train_poly["YEARS_EMPLOYED_PERCENT"] = app_train_poly["YEARS_EMPLOYED"] / app_train_poly["AGE"]
app_train_poly["CREDIT_ANNUITY_RATIO"] = app_train_poly["AMT_CREDIT"] / app_train_poly["AMT_ANNUITY"]
app_train_poly["INCOME_ANNUITY_RATIO"] = app_train_poly["AMT_INCOME_TOTAL"] / app_train_poly["AMT_ANNUITY"]
app_train_poly["INCOME_CREDIT_RATIO"] = app_train_poly["AMT_INCOME_TOTAL"] / app_train_poly["AMT_CREDIT"]
app_train_poly["CREDIT_GOODS_PRICE_RATIO"] = app_train_poly["AMT_CREDIT"] / app_train_poly["AMT_GOODS_PRICE"]
app_train_poly["CREDIT_DOWNPAYMENT"] = app_train_poly["AMT_GOODS_PRICE"] / app_train_poly["AMT_CREDIT"]
app_train_poly["CREDIT_INCOME_PERCENT"] = app_train_poly["AMT_CREDIT"] / app_train_poly["AMT_INCOME_TOTAL"]
app_train_poly["ANNUITY_INCOME_PERCENT"] = app_train_poly["AMT_ANNUITY"] / app_train_poly["AMT_INCOME_TOTAL"]
app_train_poly["RATIO_CREDIT_GOODS_PRICE"] = app_train_poly["AMT_CREDIT"] / app_train_poly["AMT_GOODS_PRICE"]
app_train_poly["DIFF_GOODS_PRICE_CREDIT"] = app_train_poly["AMT_CREDIT"] - app_train_poly["AMT_GOODS_PRICE"]
app_train_poly['CREDIT_TERM'] = app_train_poly['AMT_ANNUITY'] / app_train_poly['AMT_CREDIT']
```

Exemple : CREDIT_ANNUITY_RATIO le montant d'annuité a un sens mais encore plus de sens si on le comparera avec le montant de revenu annuel d'un client

On crée donc des variables supplémentaires afin de mieux saisir le sens de certaines variables quantitatives:
CREDIT_INCOME_PERCENT: le pourcentage du montant du crédit par rapport aux revenus d'un client
ANNUITY_INCOME_PERCENT: le pourcentage de la rente du prêt par rapport au revenu d'un client
CREDIT_TERM: la durée du versement en mois (puisque la rente est le montant mensuel)
DAYS_EMPLOYED_PERCENT: le pourcentage des jours de travail par rapport à l'âge du client

5 . Modèle explication

- **Dummy Classifier**

Classificateur donne la prédiction selon la règle prédéfinie cette classification prédit la classe la plus fréquente ‘0’

- **Logistic Regression**

Il s'agit d'un modèle linéaire généralisé avec une fonction logistique Elle permet de prédire la cible

- **Decision Tree**

Un arbre de décision permet de construire des règles explicites à partir des variables explicatives pour expliquer la variable cible à chaque étape il sépare les individus en k groupe pour expliquer la variable cible

- **Random Forest**

un ensemble d'arbres de décision (la méthode d’ensemble) sont très similaires entre eux mais à chaque fois légèrement différents Les résultats de tous les arbres sont combinés pour donner une réponse finale celle qui a eu la majorité de vote.

Avant de lancer l’apprentissage des modèles on règle le problème :

- La répartition des valeurs 'TARGET' est fortement déséquilibré (davantage de 0 que 1)
- Les modèles cités plus haut ne sont pas fait pour une classification déséquilibrée
- 1ère solution oversampling : un dataset artificiel
- 2ème solution under sampling : attribution du poids pénalisant des erreurs commises lors de l'apprentissage

5 . Métrique explication

		Classe Réelle	
		-	+
Classe Prédite	-	True Negatives	False Negatives
	+	False Positives	True Positives

Recall = TP / (TP +FN)

Accuracy = (TN + TP) / (TN + FN + TP + FP) Precision =TP / (TP + FP)

• Accuracy : mesure la proportion des clients correctement classés parmi l'ensemble des clients

• F-Mesure : est la moyenne harmonique de Precision et Recall en donnant à chacun la même pondération

F-Mesures = { 2* Precision * Recall) / (Precision + Recall)

• Fbeta-Mesure: La Fbeta-mesure est une généralisation de la F-mesure qui ajoute un paramètre de configuration appelé bêta

Fbeta-Mesures = ((1+beta au 2)* Precision * Recall) / (beta au 2* Precision + Recall)

5 . Modèle retenu

Scénarios	Situation	Gain / Perte
Des clients non payeurs classés comme des clients non payeurs (FP	moyen	refus de crédit —> l'intérêt généré par le prêt vaut 0
Des clients ayant remboursé classés comme des clients ayant remboursé (TN)	bien	accord de crédit —> l'intérêt généré par le prêt
Des clients non payeurs classés comme des clients ayant remboursé (FN)	mauvais	accord de crédit —> non remboursement & perte d'intérêt généré par le prêt
Des clients non payeurs classés comme des clients non payeurs (TP)	moyen	refus de crédit —> l'intérêt généré par le prêt vaut 0

Parmi les 4 scénarios listés celui qui est à éviter en priorité est scénario 3 à un volume important de FN génère un coût important pour l'entreprise

Recall permet de tenir en compte du volume de FN C'est pourquoi on privilége la métrique Recall par rapport à la Precision
Afin d'accorder plus d'importance au Recall dans mon F-Beta-mesure on attribue Beta une valeur 2

Le modèle de **régression logistique** est le modèle qui obtient les meilleurs résultats selon la F-Beta-mesure, c'est pourquoi on décide de retenir ce modèle

6 . Déploiement de l’api sur Heroku

- Création d’un compte gratuit sur Herokuapp.com
- Creation d’une nouvelle Application sur Heroku **oc-scoring**
- Téléchargez et installez la CLI Heroku .

```
$ login heroku
```

Cloner le référentiel
Utilisez Git pour cloner le code source d' oc-scoring sur ma machine locale

```
$ heroku git:clone -a oc-scoring
$ cd oc-scoring
```

Déployez vos modifications
Apportez quelques modifications au code que vous venez de cloner et déployez-les sur Heroku à l'aide de Git.

```
$ git add .
$ git commit -am "message"
$ git push heroku master
```

Présentation de Dashboard

Accédons au Dashboard :

https://oc-scoring.herokuapp.com/

Information

Une note méthodologique a été rédigé afin de communiquer ma démarche de modélisation
Un dossier contenant tous les scripts du projet a été créé dans GitHub
https://github.com/CASMAKOUAOUCI/scoring


Merci Pour Votre Attention