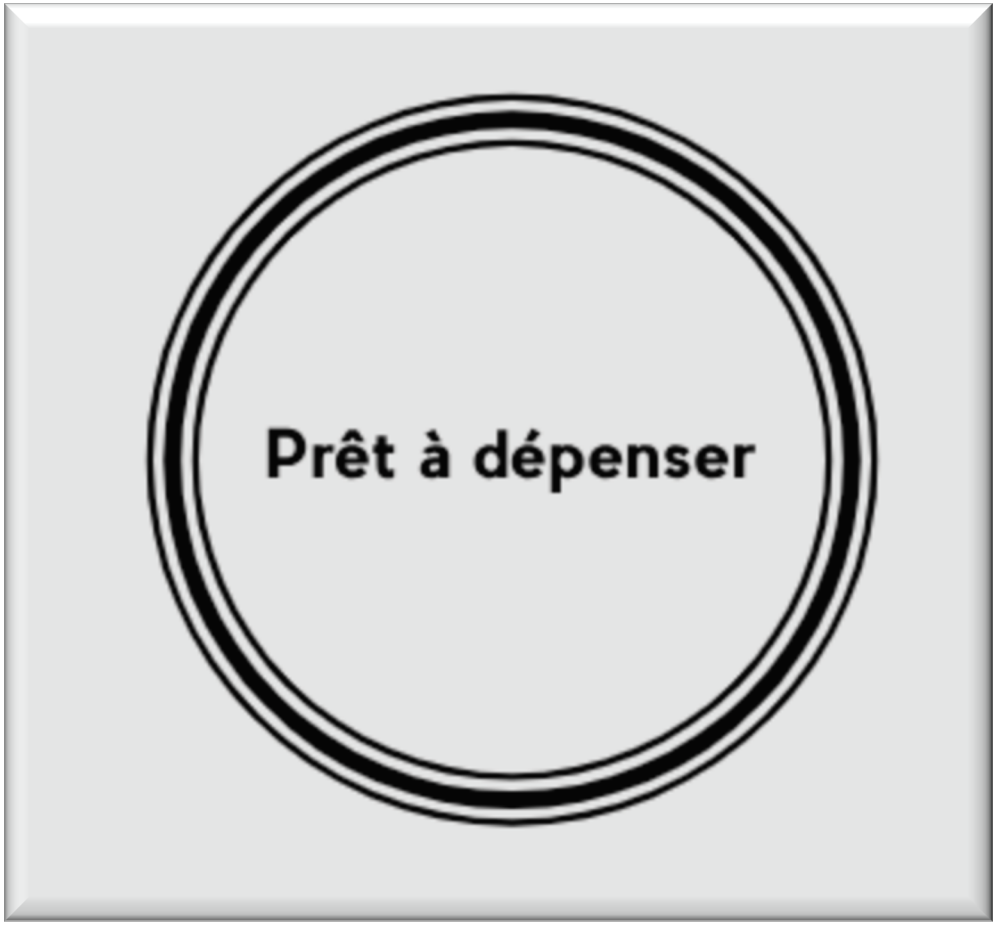
Projet 4

Construire un modèle de scoring



Yana PALYVODA le 27/09/2020

OPENCLASSROOMS

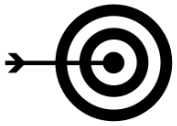
Sommaire

- Contexte du projet et compréhension de la problématique du métier
- Environnement
- Les étapes du projet
- Description du jeu de données
- Transformation du jeu de données
- Comparaison et synthèse des résultats pour les modèles utilisés
- Interprétabilité du modèle
- Conclusion
- Sources

Contexte du projet et compréhension de la problématique du métier

Problématique métier:

La société financière « Prêt à dépenser » propose des crédits à la consommation pour des personnes ayant peu ou pas d'historique de prêt. Il est très important d'évaluer le risque de non-remboursement du crédit pour chaque demandeur du prêt.



L'objectif:

L'objectif de ce projet est de développer un algorithme de scoring pour aider à décider si un prêt peut être accordé à un client.

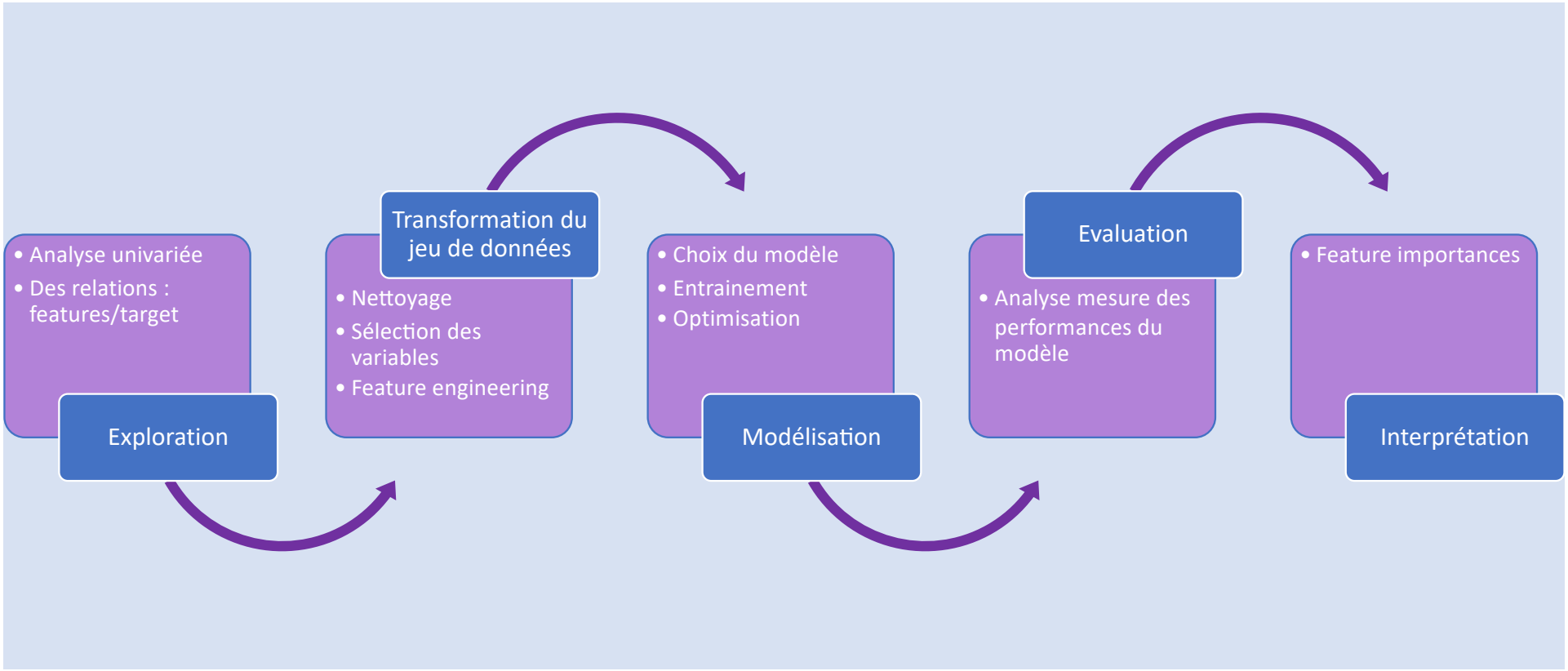
Résultats:

Le modèle doit :

- calculer la probabilité qu'un client rembourse le prêt ou non
- être facilement interprétable par les chargés de relation client
- proposer une mesure de l'importance des variables qui ont poussé le modèle à donner cette probabilité à un client.

Environnement de développement et librairies utilisées

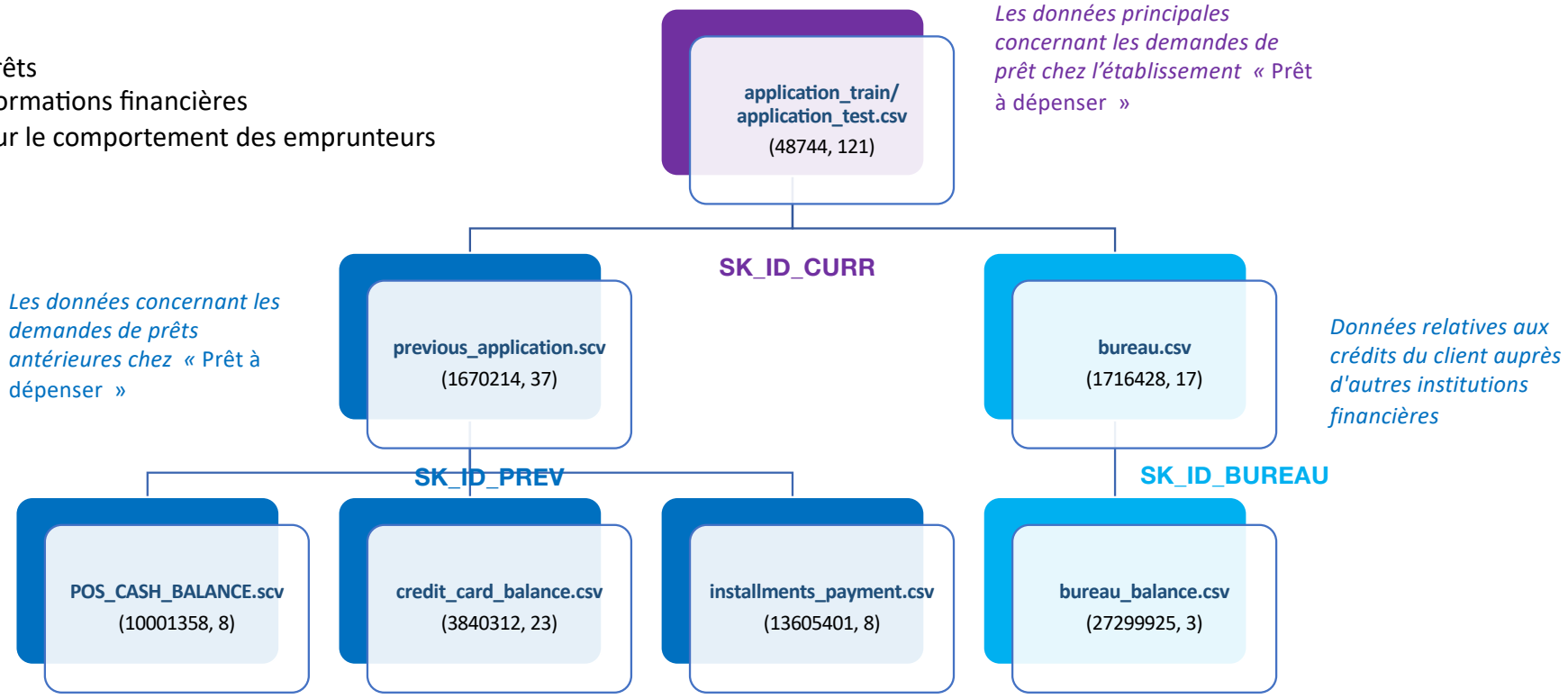
Les étapes du projet



Description du jeu de données

Les données d'origine sont reparties en 7 fichiers différents qui contient :

- un historique de prêts
- un historique d'informations financières
- des informations sur le comportement des emprunteurs



Description du jeu de données

Valeurs manquantes (plus de 60% par colonne)

application_train

	Total	Pct %
COMMONAREA_AVG	214865	69.872297
COMMONAREA_MODE	214865	69.872297
COMMONAREA_MEDI	214865	69.872297
NONLIVINGAPARTMENTS_AVG	213514	69.432963
NONLIVINGAPARTMENTS_MEDI	213514	69.432963
NONLIVINGAPARTMENTS_MODE	213514	69.432963
FONDKAPREMONT_MODE	210295	68.386172
LIVINGAPARTMENTS_MODE	210199	68.354953
LIVINGAPARTMENTS_MEDI	210199	68.354953
LIVINGAPARTMENTS_AVG	210199	68.354953
FLOORSMIN_MEDI	208642	67.848630
FLOORSMIN_AVG	208642	67.848630
FLOORSMIN_MODE	208642	67.848630
YEARS_BUILD_MODE	204488	66.497784
YEARS_BUILD_AVG	204488	66.497784
YEARS_BUILD_MEDI	204488	66.497784
OWN_CAR_AGE	202929	65.990810

Pourcentage de valeurs manquantes: 24.54 %

bureau

	Total	Pct %
AMT_ANNUITY	1226791	71.473490
AMT_CREDIT_MAX_OVERDUE	1124488	65.513264

Pourcentage de valeurs manquantes: 13.50 %

previous_application

	Total	Pct %
RATE_INTEREST_PRIMARY	1664263	99.643698
RATE_INTEREST_PRIVILEGED	1664263	99.643698
NAME_PRODUCT_TYPE	1063666	63.684414

Pourcentage de valeurs manquantes: 26.19 %

Pourcentage de valeurs manquantes:

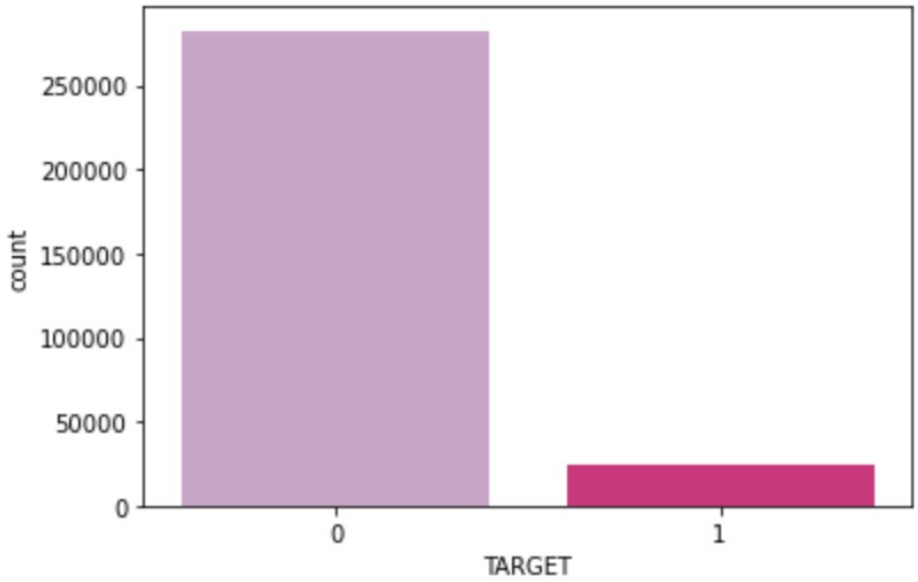
- ✓ bureau_balance: 0.00 %
- ✓ credit_card_balance: 6.65 %,
- ✓ installments_payments: 0.01 %
- ✓ pos_cash_balance: 0.07 %

Description du jeu de données

Variable TARGET

0: le prêt a été remboursé

1: le prêt n'a pas été remboursé

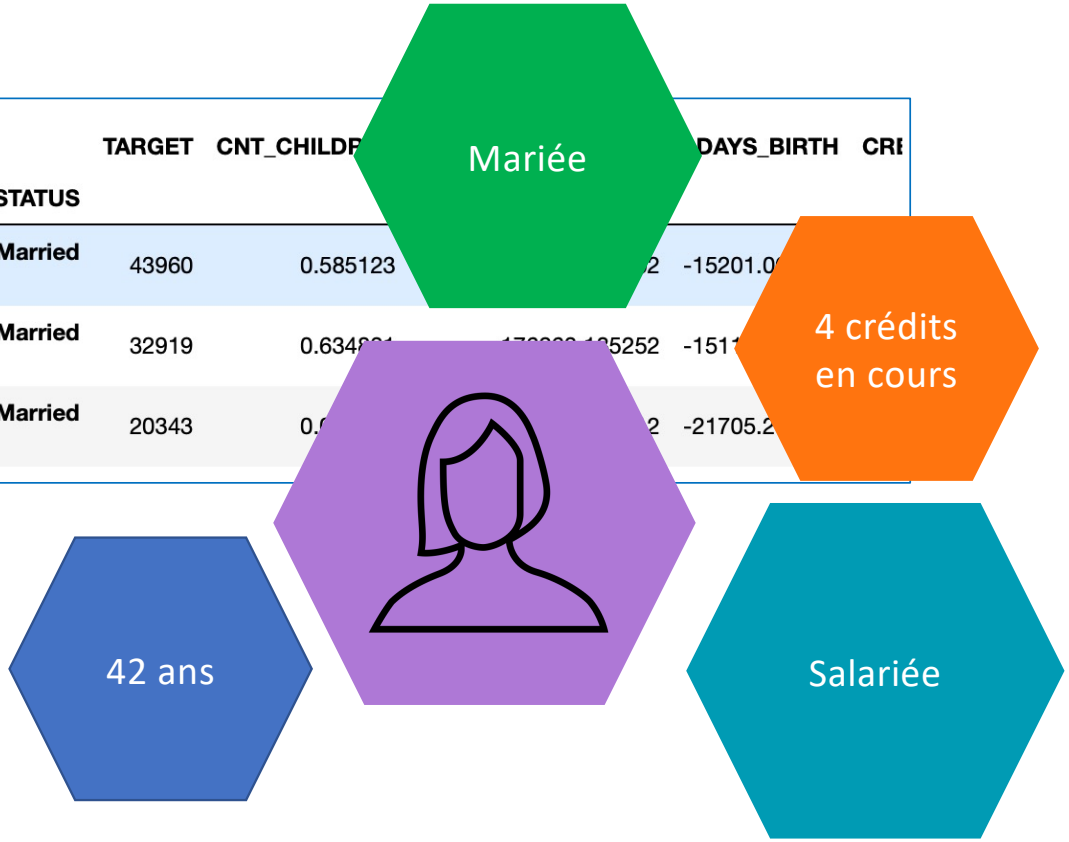


Nous constatons que les valeurs de variable target (1 et 0) n'ont pas les proportions équilibrées. La DataFraime contient 8% des lignes ayant le TARGET à 1. Lors de l'entrainement nous avons besoin d'utiliser une méthode de stratification.

Description du jeu de données

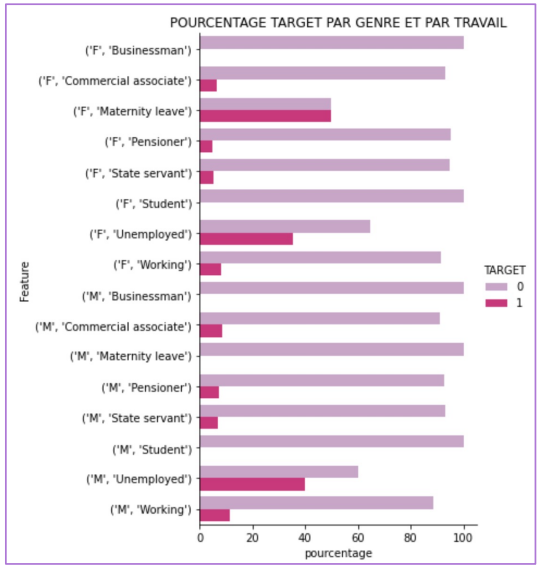
Profil client type de l'établissement

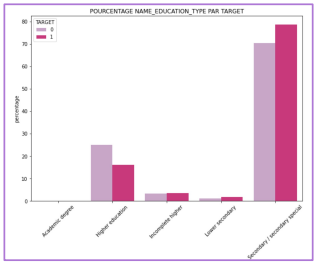
CODE_GENDER	NAME_INCOME_TYPE	NAME_EDUCATION_TYPE	NAME_FAMILY_STATUS	TARGET	CNT_CHILDR	DAYS_BIRTH	CRE
F	Working	Secondary / secondary special	Married	43960	0.585123	-15201.0	
M	Working	Secondary / secondary special	Married	32919	0.634	-151	
F	Pensioner	Secondary / secondary special	Married	20343	0.	-21705.2	

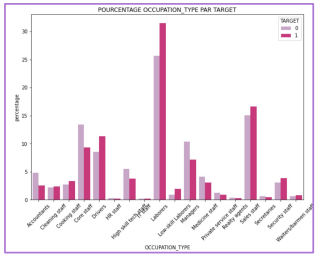


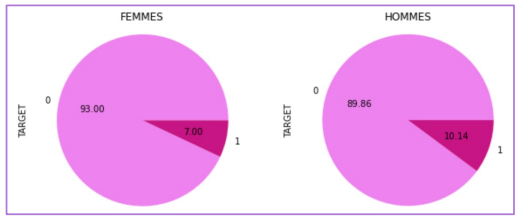
Description du jeu de données

Facteurs de risques









Nous avons constaté également que Data set contient 66% de femmes. Cependant 10% d'hommes ne remboursent pas le crédit contre 7% de femmes.

- Sans emploi
- Ouvrier
- Niveau d'étude: Enseignement secondaire
- Congé maternité

Transformation du jeu de données (nettoyage et feature engineering)

Nettoyage des données

- ✓ Suppression des colonnes ayant plus de 60% de données manquantes.
- ✓ Suppression des colonnes estimées non utiles pour notre analyse.
- ✓ Imputation des valeurs manquantes:
  - ‘mean’ pour quantitatives
  - ‘most freq’ pour qualitatives
    - ** KNNImputer montrait la meilleur résultat sur un échantillon de petit taille. Cependant, l’utilisation KNNImputer est impossible suite à sont temps d’exécution très élevé sur la totalité des données*

Sélection des variables

Utilisation Lasso (*Least Absolute Shrinkage and Selection Operator) :*

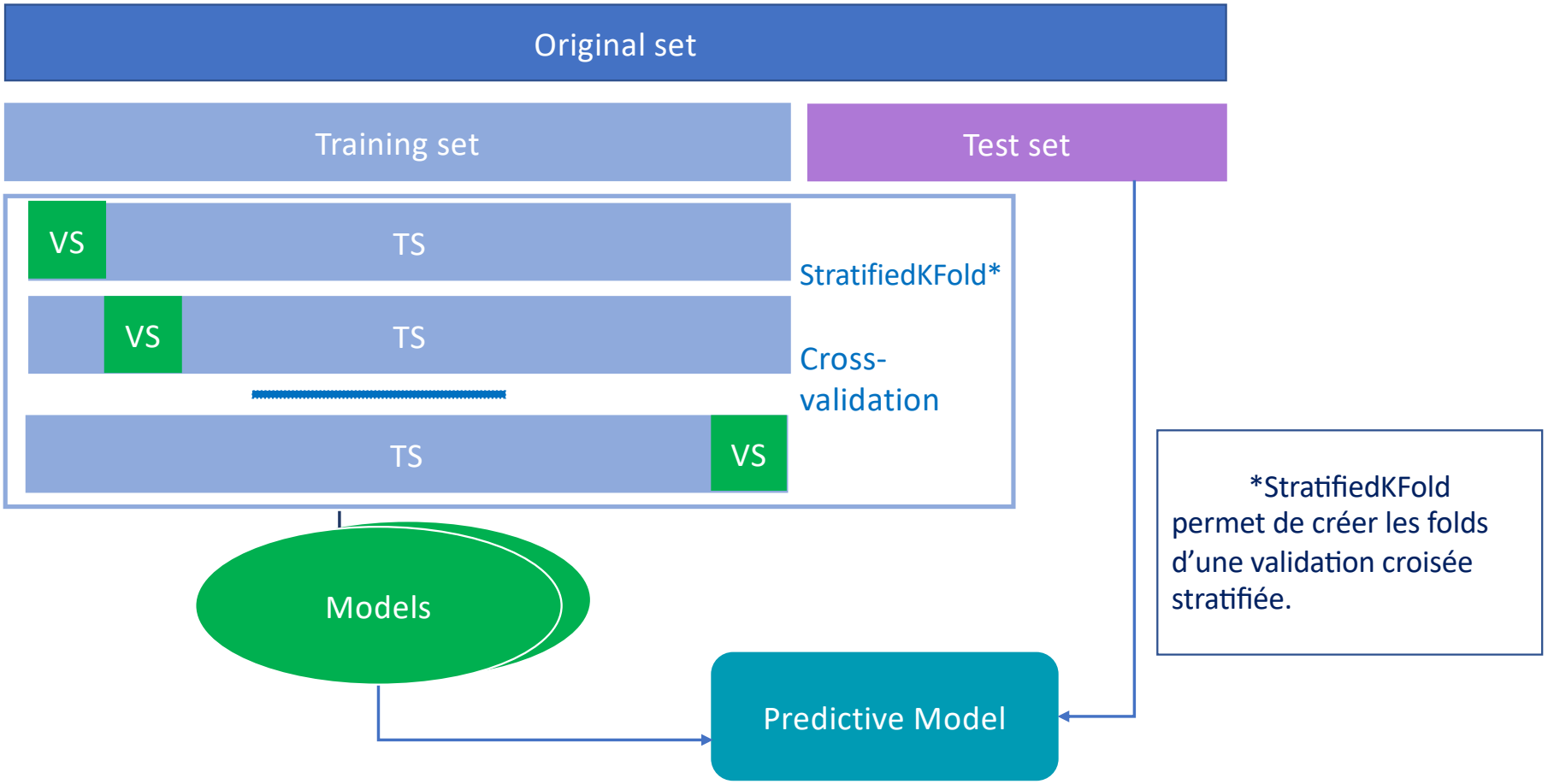
une méthode de sélection de variables et de réduction de dimension supervisée : les variables qui ne sont pas nécessaires à la prédiction de l'étiquette sont éliminées.

Feature engineering

- ✓ Ajout des nouvelles variables:
  - 'CREDIT_TERM': la durée de crédit
  - CREDIT_EN_COURS_APP’: nombre de crédits en cours dans l’établissement
  - 'CREDIT_REFUSED_APP’: crédit refusé par l’établissement
- ✓ Transformation des variables catégorielles en one-hot
- ✓ Normalisation des variables par StandardScaler

Transformation du jeu de données (nettoyage et feature engineering)

Apres la transformation, le jeu de données à été séparé en training set et test set.

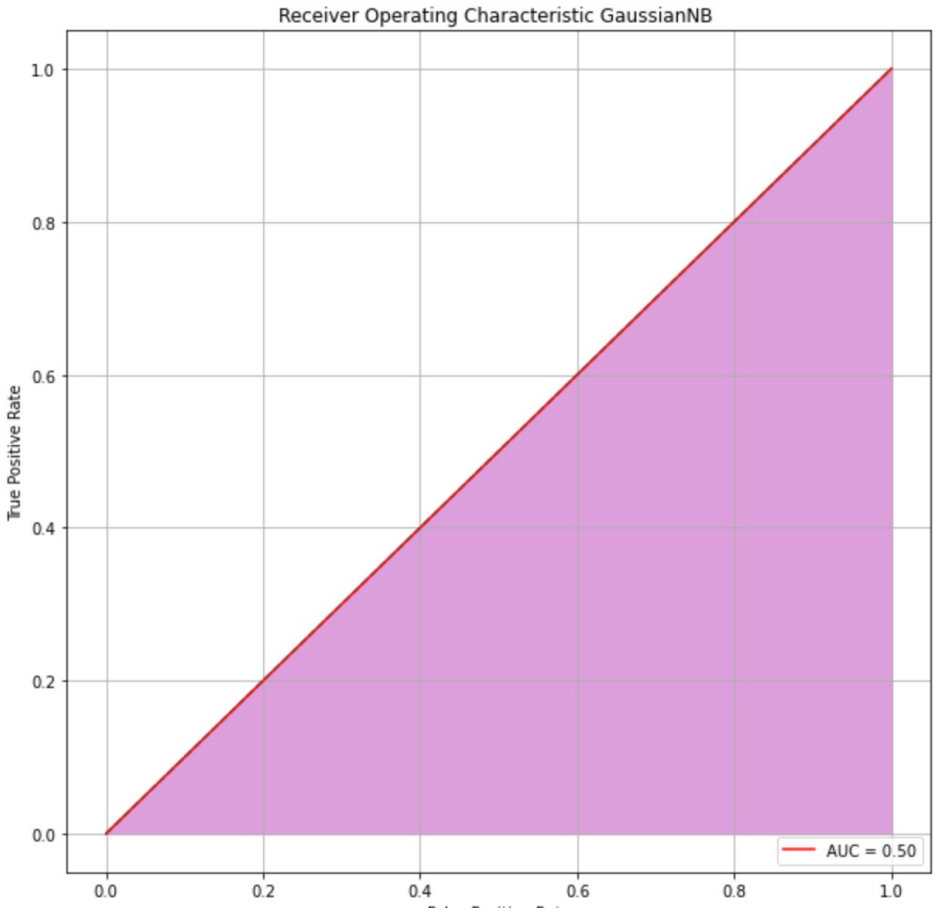


Baseline

Pour évaluer la performance de nos algorithmes, nous avons besoin d'une baseline.

Nous avons choisi le modèle GaussianNB: simple, performant et très rapide.

Une AUROC à 0.5 (classifier aléatoire) sera notre valeur de référence



Comparaison résultats sans et avec nouvelles variables

Avant

	LogisticRegression	DecisionTreeClassifier	SGDClassifier	RandomForestClassifier	Best Score
Accuracy	0.918857	0.852222	0.918357	0.918935	RandomForestClassifier
Precision	0.509925	0.144367	0.444998	0.638961	RandomForestClassifier
Recall	0.012796	0.166658	0.022424	0.003229	DecisionTreeClassifier
F1 Score	0.024954	0.154683	0.042482	0.006422	DecisionTreeClassifier
ROC_AUC	0.747884	0.539720	0.736566	0.725644	LogisticRegression

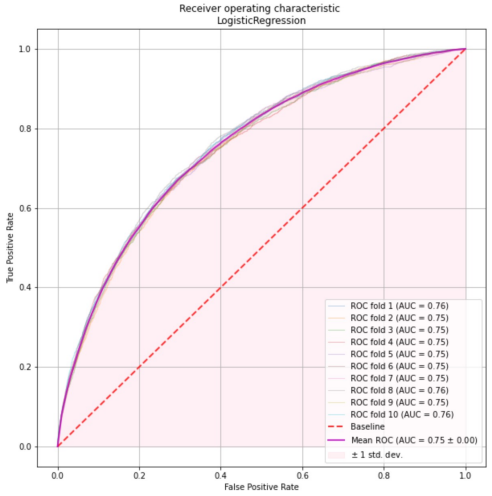
Après

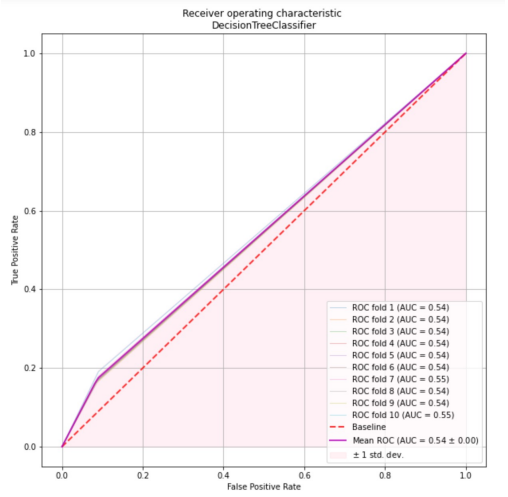
	LogisticRegression	DecisionTreeClassifier	SGDClassifier	RandomForestClassifier	Best Score
Accuracy	0.918935	0.852207	0.918576	0.918964	RandomForestClassifier
Precision	0.520242	0.146778	0.480416	0.723083	RandomForestClassifier
Recall	0.016564	0.170723	0.022543	0.003229	DecisionTreeClassifier
F1 Score	0.032091	0.157817	0.042916	0.006424	DecisionTreeClassifier
ROC_AUC	0.752266	0.541565	0.741725	0.728305	LogisticRegression

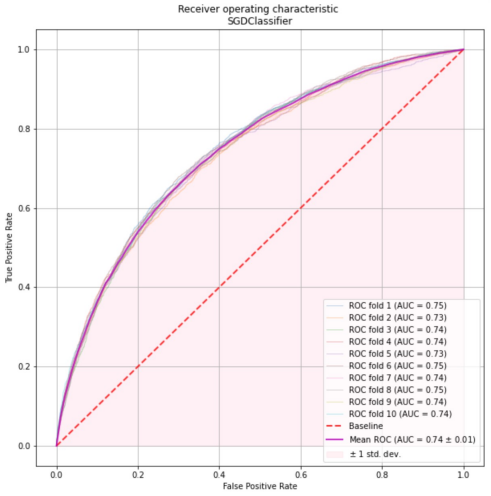
Nous constatons qu'après l'ajout des trois nouvelles variables la valeur de ROC AUC est augmentée

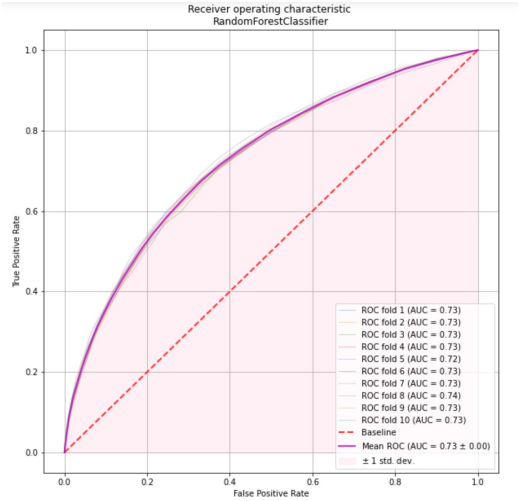
Comparaison et synthèse des résultats pour les modèles utilisés

	LogisticRegression	DecisionTreeClassifier	SGDClassifier	RandomForestClassifier	Best Score
Accuracy	0.918935	0.852207	0.918576	0.918964	RandomForestClassifier
Precision	0.520242	0.146778	0.480416	0.723083	RandomForestClassifier
Recall	0.016564	0.170723	0.022543	0.003229	DecisionTreeClassifier
F1 Score	0.032091	0.157817	0.042916	0.006424	DecisionTreeClassifier
ROC_AUC	0.752266	0.541565	0.741725	0.728305	LogisticRegression

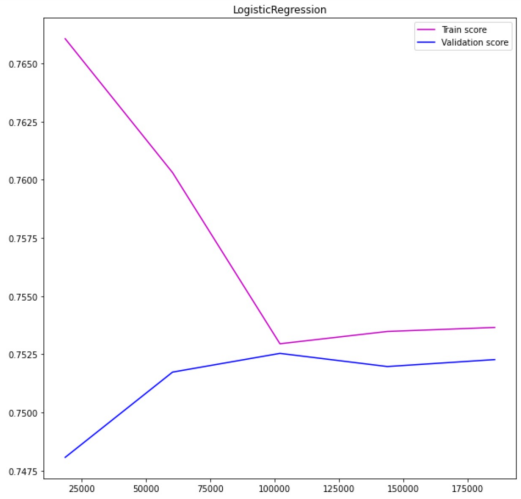


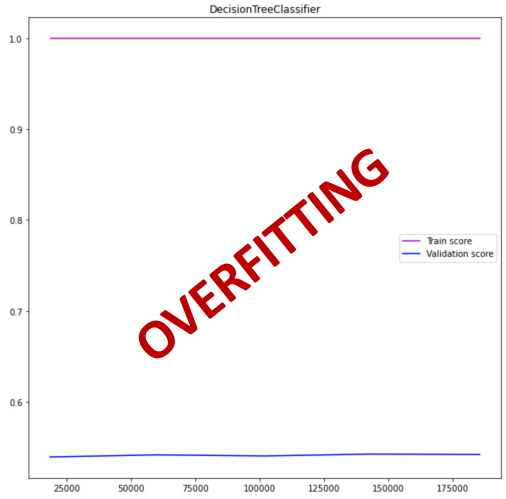


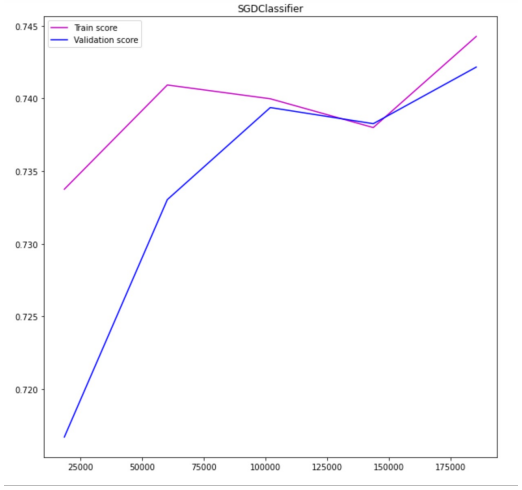


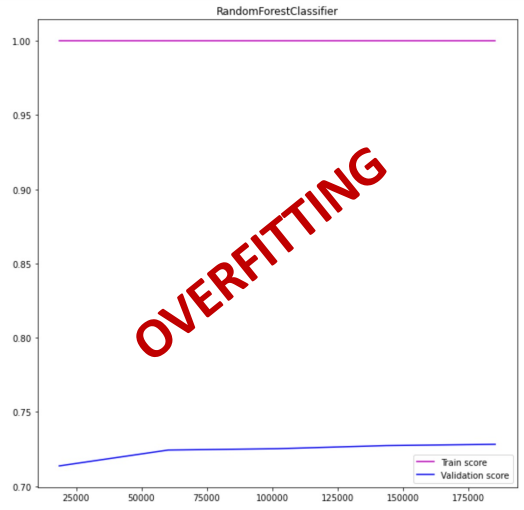


Comparaison learning_curve pour les modèles utilisés









LogisticRegression montre un meilleur résultat d'apprentissage.

Optimisation de hyperparameters

Pour la recherche des paramètres optimaux pour le modèle sélectionné nous avons utilisé le GridSearch.
On indique les paramètres à faire varier, scikit-learn les croise et mesure les performances en validation croisée

Nous avons sélectionné trois paramètres pour améliorer notre modèle:

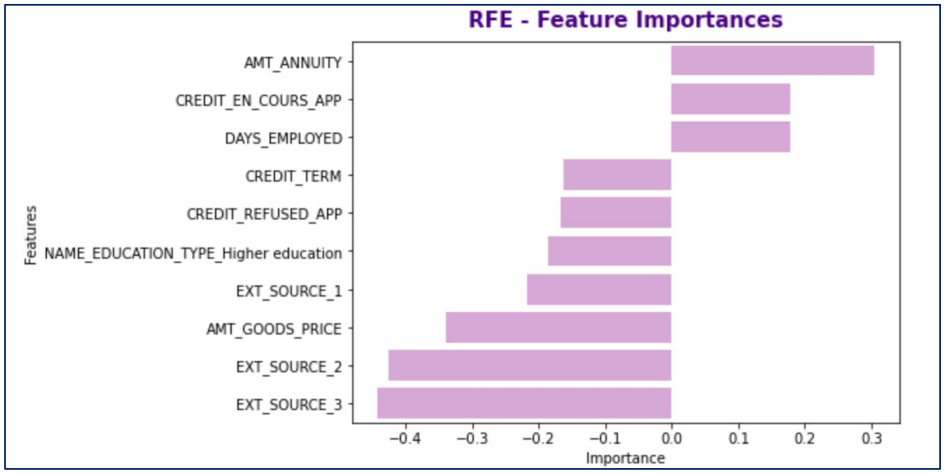
- ✓ -'penalty' : type de regularisation
- ✓ 'C': des valeurs plus petites indiquent une régularisation plus forte
- ✓ -'max_iter': nombre maximal des itérations

```
params ={'penalty':['l2','l1'],'C':[0.001,0.01,0.1],'max_iter':[100,200,300]}
```

Le résultat:

```
Best roc_auc: 0.7523, with best params: {'C': 0.1, 'max_iter': 300, 'penalty': 'l1'}
```

Feature importance - Recursive Feature Elimination

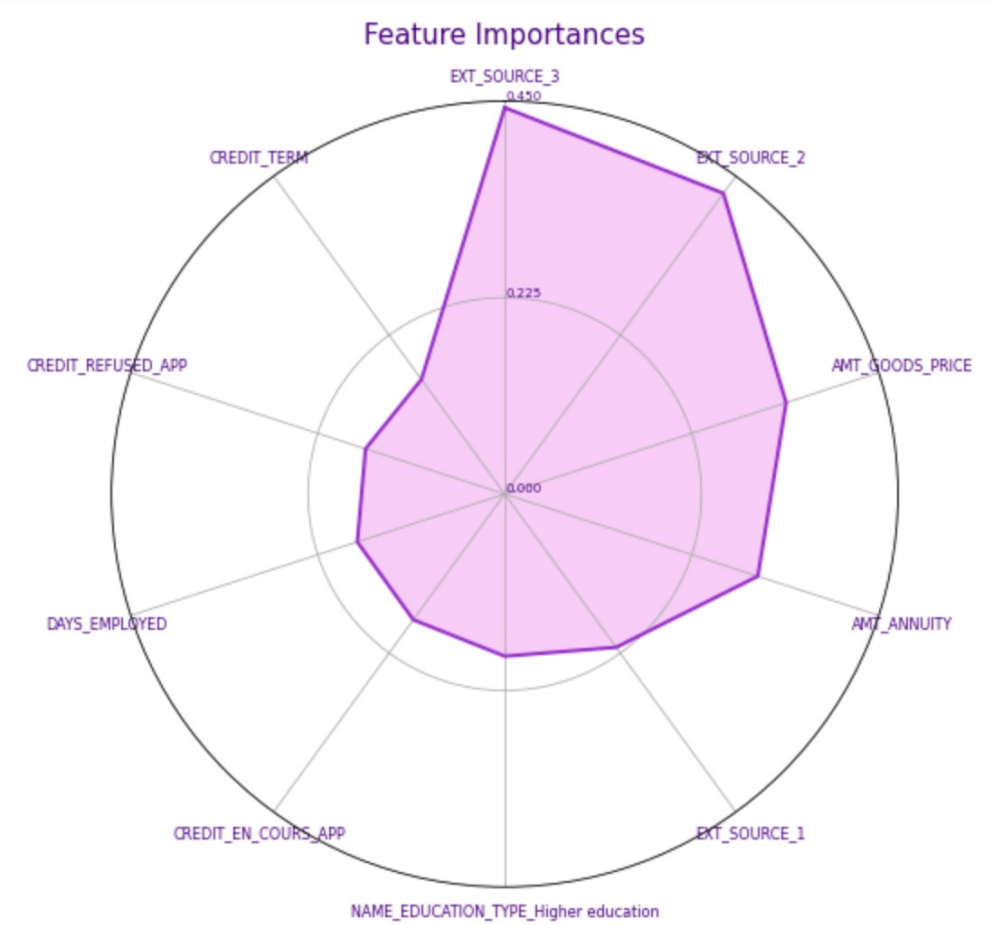


Pour identifier les variables les plus importantes pour notre modèle, nous avons utilisé l'algorithme RFE (Recursive Feature Elimination)

Le principe de cet algorithme est d'éliminer récursivement les variables les moins importantes pour le modèle jusqu'à ce que le nombre de variables défini dans les paramètres soit atteinte.

D'abord, un modèle pour l'ensemble des variables est créé et le score d'importance de chaque variable est calculé. Ensuite, les variables les moins importantes sont supprimées, le modèle est reconstruit et les scores d'importance sont calculés à nouveau.

Interprétabilité du modèle



Les variables les plus importantes pour notre modèle:

- **EXT_SOURCE_3**
- **EXT_SOURCE_2**
- **AMT_GOODS_PRICE**

Conclusion

Les quatre modèles de classifications supervisés qui retournent des scores de probabilité ont été testés:

- *LogisticRegression*
- *DecisionTreeClassifier*
- *SGDClassifier*
- *RandomForestClassifier*

Leurs performances ont étés évaluées en fonction de la valeur AUROC.

LogisticRegression a montré le meilleur résultat parmi ces quatre modèles.

La performance du modèle a été améliorée par la création de nouvelles variables et l'adaptation d'hyperparamètres.

Les variables les plus importantes pour le modèle sélectionné ont étés identifiées à l'aide de la méthode RFE.

La capacité d'apprentissage du modèle peut être améliorée avec l'augmentation du nombre de variables sélectionnées et l'augmentation de la taille de l'échantillon.

Veuillez noter que les modèles très gourmands en ressources n'ont pas pu être testés suite une basse performance de l'équipement.

Sources

- Machine learning avec scikit-learn

http://eric.univlyon2.fr/~ricco/cours/cours_programmation_python.html

- Scikit-learn

https://scikit-learn.org/stable/index.html

- Machine Learnia

https://www.youtube.com/watch?v=82KLS2C_gNQ&list=PLO_fdPEVlfKqMDNmCFzQISI2H_nJcEDJq&index=1&t=2s

- Python

https://www.it-swarm.dev/fr/python/