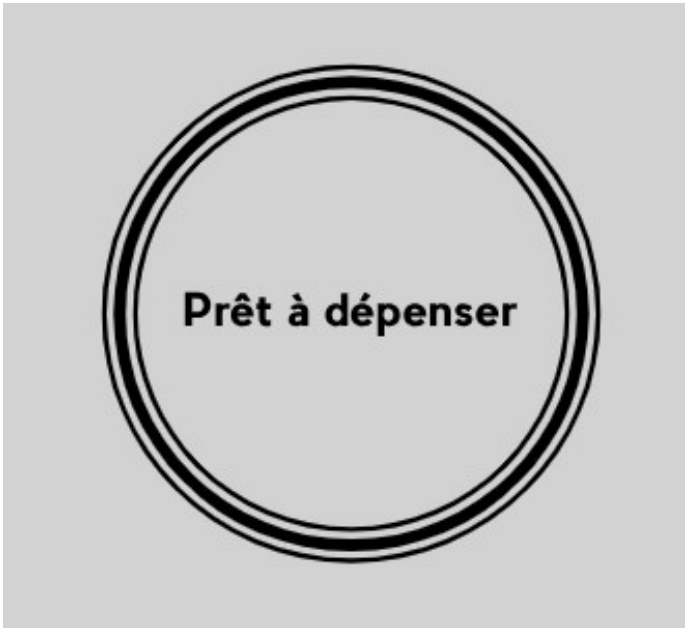
Implémenter un modèle de scoring



Projet 7 Data Scientist

Openclassrooms

Serge Davister 09/2023

Prêt à dépenser

I. Présentation

- Description du projet
- Objectifs de la mission

II. Analyse des données

- Présentation des données
- Analyse exploratoire
- Features selection
- Choix des métriques

III. Modélisation

- Entraînement des modèles
- Analyse des résultats et sélection du modèle

IV. Optimisation

- Tuning des hyper-paramètres
- Importance des caractéristiques
- Interprétation

V. Dashboard

- Création du Dashboard

VI. Conclusion

- Synthèse
- Pistes d'amélioration

I. Présentation

La demande

L'entreprise veut 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. Elle projete de développer un algorithme de classification en s'appuyant sur des sources de données variées (données comportementales, données provenant d'autres institutions financières, etc.).

Parallèlement, les chargés de relation client ont fait remonter le fait que les clients sont de plus en plus demandeurs de transparence vis-à-vis des décisions d'octroi de crédit. Cette demande de transparence des clients va tout à fait dans le sens des valeurs que l'entreprise veut incarner.

Prêt à dépenser souhaite développer un dashboard interactif pour que les chargés de relation client puissent à la fois expliquer de façon la plus transparente possible les décisions d'octroi de crédit, mais également permettre à leurs clients de disposer de leurs informations personnelles et de les explorer facilement.

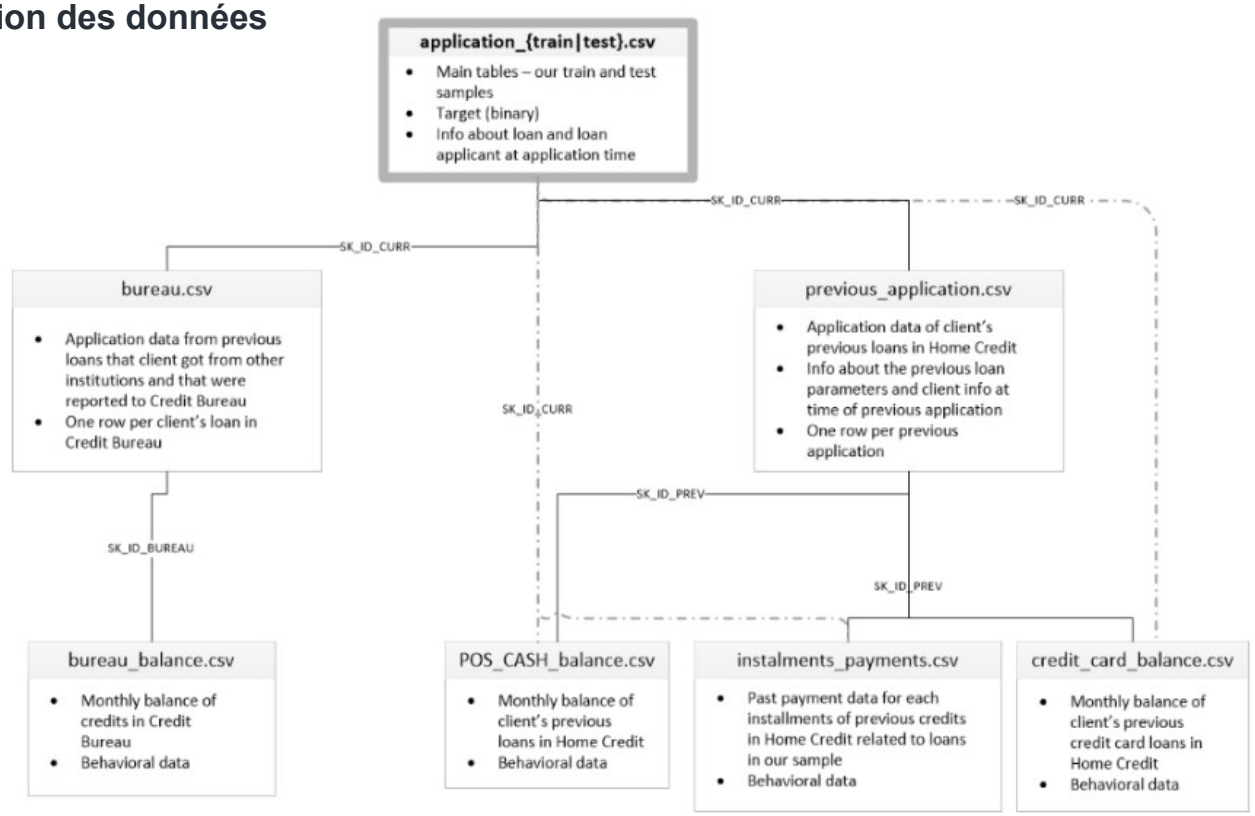
La mission

Construire un modèle de scoring qui donnera une prédiction sur la probabilité de faillite d'un client de façon automatique.
Construire un dashboard interactif à destination des gestionnaires de la relation client permettant d'interpréter les prédictions faites par le modèle, et d'améliorer la connaissance client des chargés de relation client.
Mettre en production le modèle de scoring de prédiction à l'aide d'une API, ainsi que le dashboard interactif qui appelle l'API pour les prédictions.

II. Analyse des données

Prêt à dépenser

Présentation des données

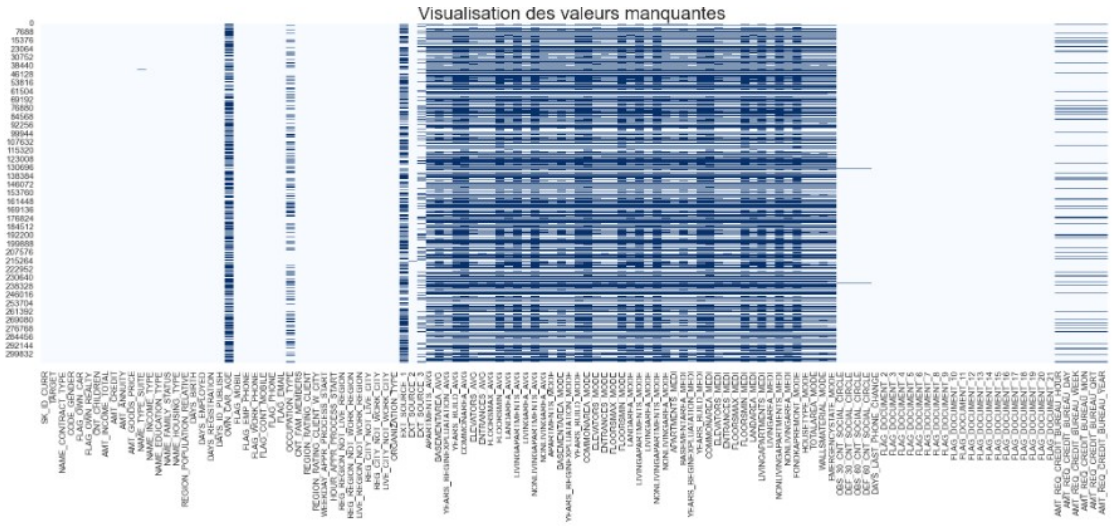


Les données sont réparties dans 7 fichiers .
La table « application » regroupe des informations personnelles ainsi que les données relatives au crédit demandé par les clients.Elle contient 2 jeux de données : l'application **train** regroupant 307 511 clients ayant demandé un crédit pour lesquels on connaît la finalité (accord ou refus de **Prêt à Depenser**)
L'application **test** qui contient les demandes en cours pour lesquelles la décision n'a pas encore été prise .
Les autres fichiers contiennent les données historiques de prêt de ces clients auprès de **Prêt à Depenser** (Previous_application) ou aupès d'autres organismes de prêts (bureau)

Analyse exploratoire - Valeurs manquantes

Prêt à dépenser

Traitement des valeurs manquantes : j'ai conservé uniquement les variables avec nan ≤ 10 %



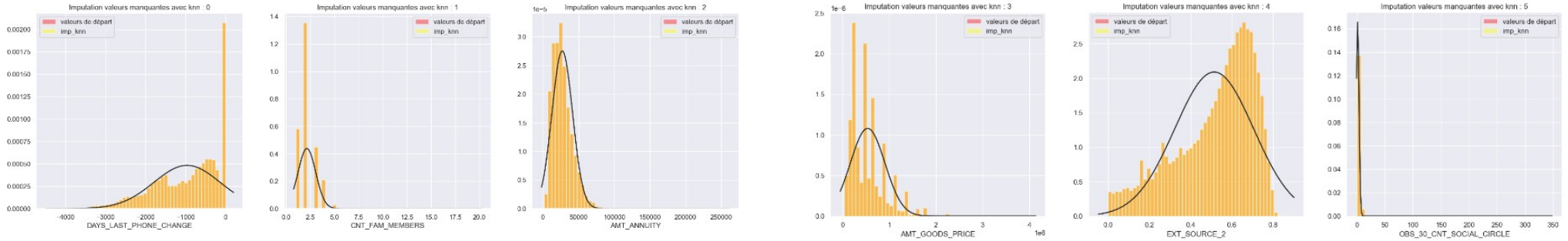
tranche	%_na count
[0-10]	65
]10-20]	7
]20-30]	0
]30-40]	1
]40-50]	8
]50-60]	24
]60-70]	17
]70-80]	0
]80-90]	0
]90-100]	0

	variable	%_na
55	DAYS_LAST_PHONE_CHANGE	0.00
56	CNT_FAM_MEMBERS	0.00
57	AMT_ANNUITY	0.00
58	AMT_GOODS_PRICE	0.09
59	EXT_SOURCE_2	0.21
60	OBS_30_CNT_SOCIAL_CIRCLE	0.33
61	DEF_30_CNT_SOCIAL_CIRCLE	0.33
62	OBS_60_CNT_SOCIAL_CIRCLE	0.33
63	DEF_60_CNT_SOCIAL_CIRCLE	0.33
64	NAME_TYPE_SUITE	0.42

J'ai utilisé :

SimpleImputer pour remplir les nan avec la valeur la plus fréquente pour les variables catégorielles

KnnImputer pour les variables numériques

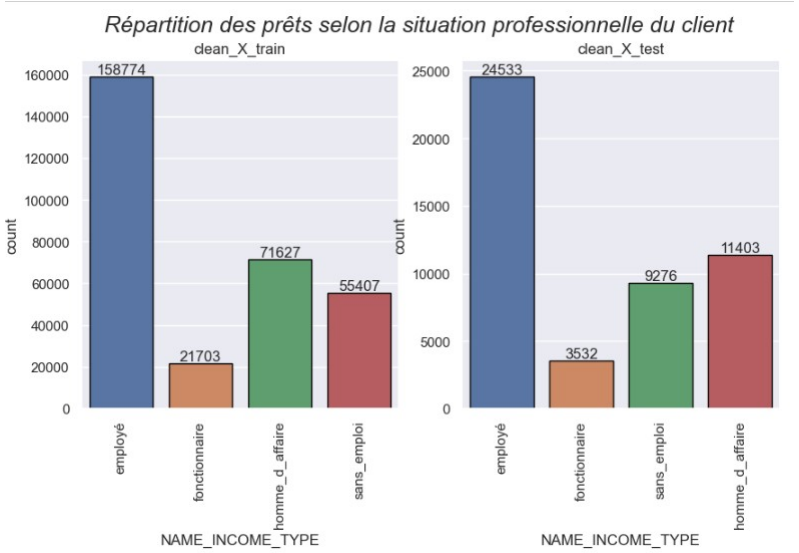


Analyse exploratoire – Transformation de données

Prêt à dépenser

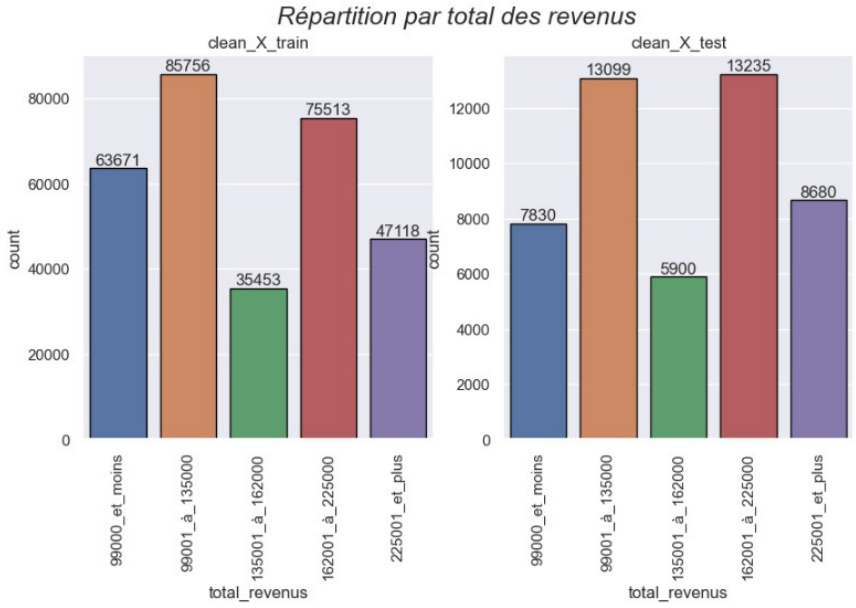
Regroupement des modalités: réduit le nombre de modalités et écarte les valeurs aberrantes

```
map_name_income_type = {'Working' : 'employé',
                        'Commercial associate' : 'homme_d_affaire',
                        'Pensioner' : 'sans_emploi',
                        'State servant' : 'fonctionnaire',
                        'Unemployed' : 'sans_emploi',
                        'Student' : 'sans_emploi',
                        'Businessman':'homme_d_affaire',
                        'Maternity leave' : 'sans_emploi'}
```



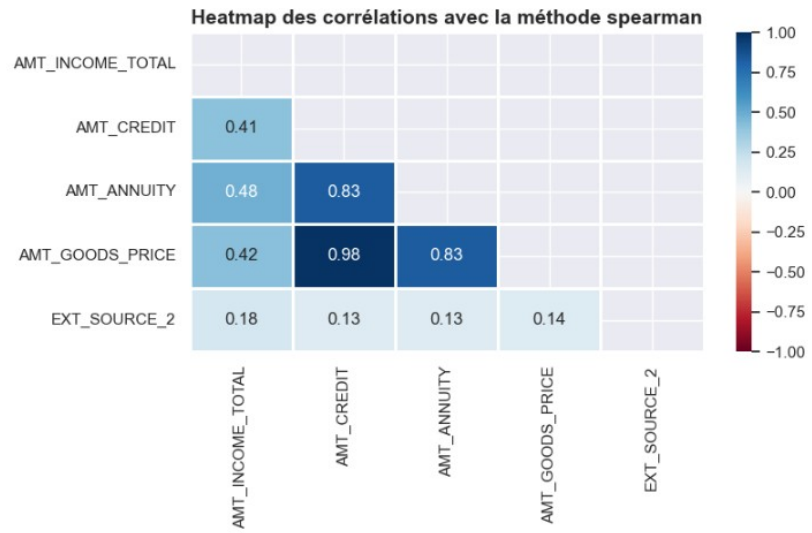
Discrétisation de variables continues: découpage en modalités

```
clean_X_train['total_revenus'] = pd.qcut(clean_X_train.AMT_INCOME_TOTAL,
                                         q=5,
                                         labels=['99000_et_moins',
                                                 '99001_à_135000',
                                                 '135001_à_162000',
                                                 '162001_à_225000',
                                                 '225001_et_plus'])
```

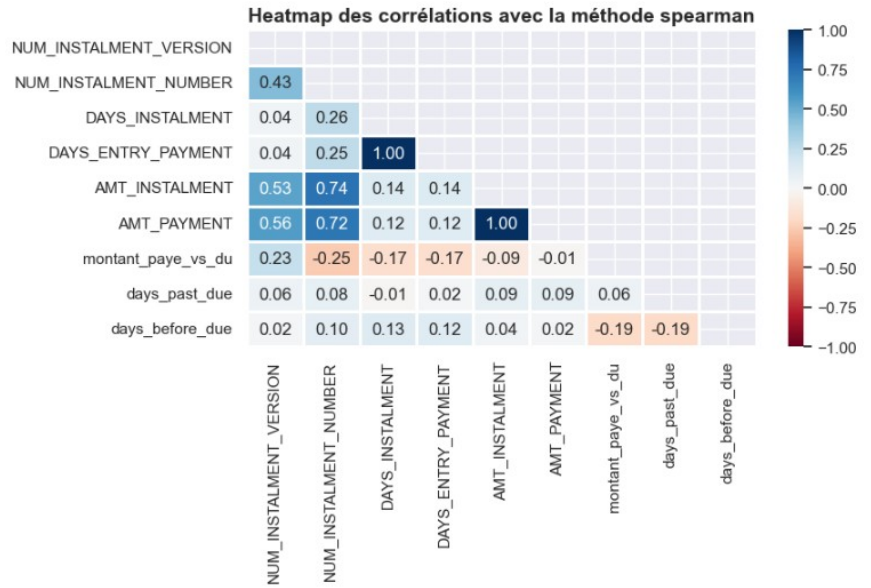


Analyse exploratoire – Réduction de données

J’ai enlevé les caractéristiques avec une corrélation importante.



Très forte corrélation entre AMT_GOODS_PRICE et AMT_CREDIT.Je supprime 'AMT_GOODS_PRICE'(99%).




```
ins_payments.drop(columns=['DAYS_INSTALMENT', 'DAYS_ENTRY_PAYMENT', 'AMT_INSTALMENT'], axis=1, inplace=True)
```

Analyse exploratoire – Feature Selection

Prêt à dépenser

Transformation des variables catégorielles bien ordonnées directement avec LabelEncoder

```
X_train_df : nb_famille
0    1_personne
Name: nb_famille, dtype: category
Categories (6, object): ['1_personne' < '2_personnes' < '3_personnes' < '4_personnes' < '5_personnes' < '6_personnes_et_plus']
```

Les catégories bien ordonnées peuvent être transformées directement en variables numériques

```
cols_ord = ['region_rating_client', 'region_rating_client_w_city',
                'nb_famille', 'age_client', 'taux_endettement',
                'nb_mensualites', 'montant_credit']

# Encode labels of multiple columns at once
#
X_train_df[cols_ord] = X_train_df[cols_ord].apply(LabelEncoder().fit_transform)
X_test[cols_ord] = X_test[cols_ord].apply(LabelEncoder().fit_transform)
#
# Print head
#
X_train_df[cols_ord].head(5)

```

	region_rating_client	region_rating_client_w_city	nb_famille	age_client	taux_endettement	nb_mensualites	montant_credit
0	1	1	0	0	1	0	2
1	0	0	1	3	1	2	5
2	1	1	0	4	0	1	0
3	1	1	1	4	3	0	1
4	1	1	0	4	2	1	2

Pour les autres,il faut indiquer l'ordre correct pour chaque modalité.

```
enf_map = {'sans_enfant': 0,
           'un_enfant': 1,
           'deux_enfants': 2,
           'trois_enfants_et_plus':3}

X_train_df['nb_enfants'] = X_train_df['nb_enfants'].map(enf_map)
X_test['nb_enfants'] = X_test['nb_enfants'].map(enf_map)
```

Analyse exploratoire – Feature Selection

Prêt à dépenser

Recherche de caractéristiques avec une variance faible ou nulles afin de réduire le nombre de caractéristiques peu pertinentes

Utilisation des selecteurs DropConstantFeatures() et VarianceThreshold()

```
drop_constant=DropConstantFeatures(variables=None,tol=0.98)
drop_constant.fit(X_train_df)
```

```
DropConstantFeatures(tol=0.98)
```

Recherche de caractéristiques fortement corrélées avec SmartCorrelatedSelection

```
# # afficher le groupe de variables et les valeurs corrélées
for group in range(len(smart_correl.correlated_feature_sets_)):
    print(smart_correl.correlated_feature_sets_[group])
    print('----------------------------------------------------------------------------------------')
    print(X_train_df[smart_correl.correlated_feature_sets_[group]].std())
    print('   ')
```

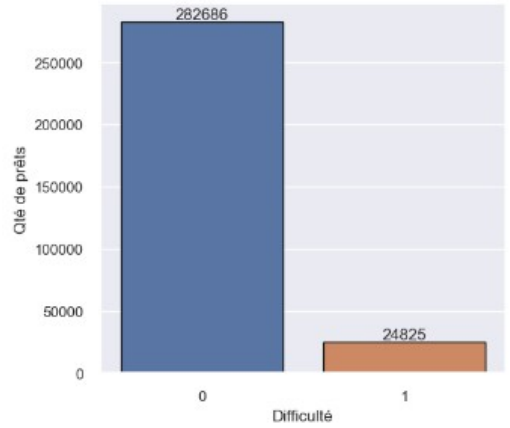
```
{'region_rating_client_w_city', 'region_rating_client'}
----------------------------------------------------------------------------------------
region_rating_client_w_city   0.5027
region_rating_client          0.5090
dtype: float64

{'nb_enfants', 'nb_famille'}
----------------------------------------------------------------------------------------
nb_enfants   0.7071
nb_famille   0.9051
dtype: float64

{'name_type_suite_Unaccompanied', 'name_type_suite_Famille'}
----------------------------------------------------------------------------------------
name_type_suite_Unaccompanied   0.3904
name_type_suite_Famille         0.3826
dtype: float64

{'flag_emp_phone_Oui', 'name_income_type_sans_emploi'}
----------------------------------------------------------------------------------------
flag_emp_phone_Oui             0.3843
name_income_type_sans_emploi   0.3843
dtype: float64

{'reg_city_not_work_city_adresse_identique', 'live_city_not_work_city_adresse_identique'}
----------------------------------------------------------------------------------------
reg_city_not_work_city_adresse_identique    0.4211
live_city_not_work_city_adresse_identique   0.3838
dtype: float64
```

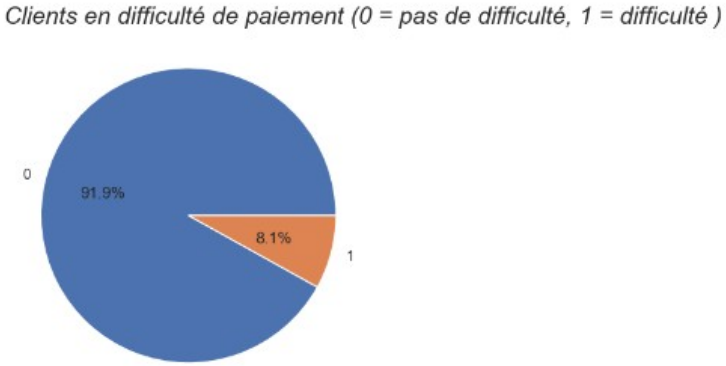
```
#  donne la variable avec plus haute valeur de chaque groupe

list_smart=smart_correl.features_to_drop_
list_smart
```

```
'region_rating_client_w_city',
'nb_enfants',
'name_type_suite_Famille',
'flag_emp_phone_Oui',
'live_city_not_work_city_adresse_identique']
```

Analyse exploratoire – Feature Selection

Nombre de clients « sans difficulté » Vs Nombre de clients « avec difficultés de paiement »





Il y a 8,1 % de clients en difficulté de paiement contre 91.9% sans difficulté. En ML, cette différence importante de classes peut nuire à la fiabilité des prédictions .

1. solution au niveau des données.

On peut ré-équilibrer les classes en augmentant artificiellement le nombre de données dans la classe minoritaire (upsampling ou sur-échantillonnage) ou en réduisant dans la classe majoritaire (downsampling ou sous-échantillonnage)

en downsampling : On peut retirer aléatoirement des points de la classe majoritaire pour réduire le déséquilibre. Une autre possibilité est de créer des paires de points appelées **Tomek link** constituées d'un point de la classe majoritaire proche d'un point de la classe minoritaire et d'éliminer le point de la classe majoritaire. Cette technique permet en outre de supprimer les outliers et de réduire la variance de la classe majoritaire.

en upsampling : on va créer de nouveaux points à partir de points existants de la classe minoritaire (similaire à la DATA AUGMENTATION utilisée pour réduire l'overfitting).On sélectionne des points de la classe minoritaire. Pour chacun des points on recherche les k plus proches voisins et on crée un point synthétique appartenant à la classe minoritaire.
SMOTE : Synthetic Minority Oversampling Technic ou suréchantillonnage minoritaire synthétique
A**DASYN** : ADAptive SYNthetic sampling ou échantillonnage synthétique adaptatif.Ici le nombre d' échantillons synthétiques généré est proportionnel au nombre d'échantillons sélectionnés qui ne sont pas de la classe minoritaire. En conséquence , l'algorithme va générés plus d'exemples synthétiques dans les zones où les échantillons de la classe minoritaire sont rares.

2. solution de résolution au niveau des algorithmes.

On affecte un poids plus important à la classe minoritaire.(apprentissage sensible aux coûts) En pratique, on va spécifier à notre modèle que le fait de bien classer un point de la classe minoritaire est plus important que de bien classer un point de la classe majoritaire. De cette façon on s'arrange pour qu'une erreur de classification d'un point de la classe minoritaire soit considérée par le modèle comme étant plus grave qu'une erreur de classification sur la classe majoritaire. L'objectif des modèles de machine learning étant de toujours d'optimiser un paramètre prédéfini, l'optimisation du gain va inciter l'algorithme à donner plus d'importance à la classe minoritaire et donc à améliorer les prédictions sur celle-ci.

Analyse exploratoire – Choix des métriques

Prêt à dépenser

Avec des données déséquilibrées, il faut opter pour des métriques qui sont moins influencées par la classe majoritaire tels que le **recall** et le **F-score**.Si on ne le fait pas ,on peut penser que le modèle donne de bons résultats parce que le score est élevé alors que les prédictions ne sont correctes que pour la classe majoritaire et qu'une grande partie des points de la classe minoritaire sont mal classifiés. Le score est affecté par le déséquilibre des classes. De plus dans notre cas, une erreur de prédiction est plus dommageable que l'autre.

Les faux positifs sont de bons clients marqués comme mauvais.Le prêt ne leur est pas accordé.La banque perd le client (erreur de type 1)

Les faux négatifs sont de mauvais clients marqués comme bons . Le prêt leur est accordé. Cela représente une perte financière pour la société de crédit (erreur de type 2)

la mesure **Fbeta** calcule une moyenne harmonique pondérée de précision et de rappel tout en privilégiant les scores de rappel supérieurs aux scores de précision.Le modèle minimise les erreurs de classification pour la classe positive tout en privilégiant les modèles qui minimisent les faux négatifs plutôt que les faux positifs.

La **précision** est le nombre de résultats positifs corrects divisé par le nombre de tous les résultats positifs

Le **rappel**(recall) est le nombre de résultats positifs corrects divisé par le nombre de résultats positifs qui auraient dû être renvoyés .

Le **score F1** est une mesure de la précision d'un test. Il prend en compte à la fois la précision et le rappel du test pour calculer le score. Le score F1 peut être interprété comme une moyenne pondérée de la précision et du rappel. **F1 score = 2 * ((precision * recall) / (precision + recall))**

L**'accuracy** [illegible] modèle sur les individus positifs et négatifs de façon symétrique. Elle mesure le taux de prédictions correctes sur l'ensemble des individus : $\frac{TP+TN}{TP+TN+FP+FN}$

La **courbe ROC** (Receiver Operating Charasteristic) et l'**AUC** (Area Under Curve, soit l'aire sous la courbe) sont deux indicateurs permettant d'analyser un modèle et en particulier son seuil de détection.

La courbe ROC (ou courbe des caractéristiques de fonctionnement du récepteur) est une mesure graphique pour évaluer les performances ou la précision d'un classificateur, qui correspond à la proportion totale d'observations correctement classées.

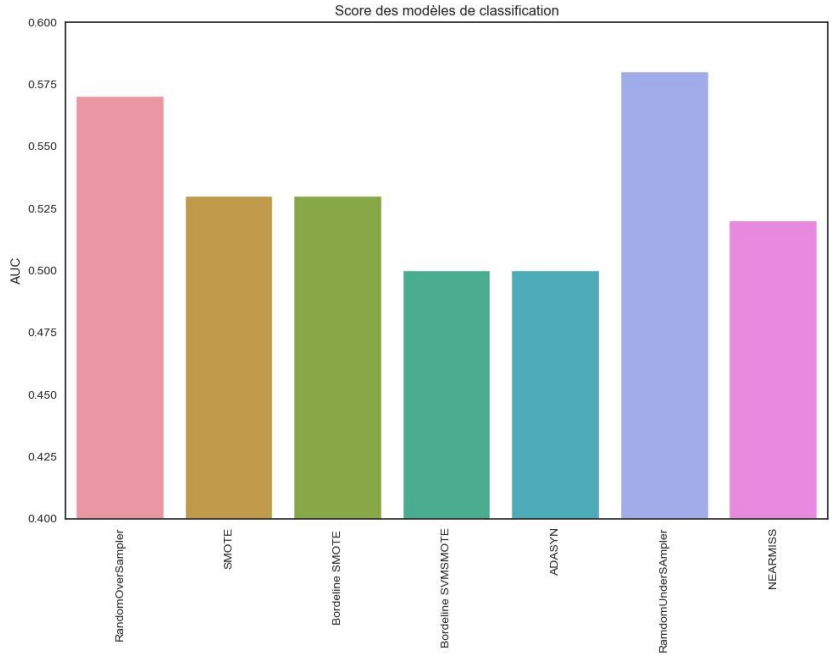
Deux indicateurs sont utilisés pour tracer la courbe ROC : le taux de vrais positifs (TVP) : Le taux de vrais positifs correspond à la proportion de vrais positifs sur l'ensemble des données positives. Il s'agit donc d'un autre nom du rappel ou de la sensibilité.

Analyse exploratoire – Classes balance methods

Prêt à dépenser

Pour évaluer les méthodes d'équilibrage des classes , j'ai utilisé un arbre de décision.

- **RandomOverSampler**
- **SMOTE**
- **Borderline SMOTE** : extension de SMOTE plus sélective dans le choix des échantillons de la classe minoritaire qui sont utilisés pour générer les nouveaux échantillons synthétiquesuniquement le long de la frontière de décision entre les deux classes.
- **Borderline SVMSMOTE** : La zone limite est approximée par les vecteurs de support obtenus après entraînement d'un classificateur SVM standard sur l'ensemble d'entraînement d'origine.
- **ADASYN :** génére des échantillons synthétiques inversement proportionnels à la densité des échantillons dans la classe minoritaire
- **RandomUnderSampler**
- **NEARMISS :** Son principe est de supprimer les observations de la classe majoritaire lorsque des observations associées à des classes différentes sont proches l'une de l'autre.




```python
# SMOTE La génération d'échantillons synthétiques
smo = SMOTE()
X_sm, y_sm = smo.fit_resample(X_train, y_train)
print("y_sm -Resampling SMOTE")
print(pd.DataFrame({
    "COUNT": y_sm.value_counts(),
    "RATIO": y_sm.value_counts() / len(y_sm) * 100}))
print('-----------------------------------------------')
# Entraînement du modèle d'arbre
sm_over_=model1.fit(X_sm, y_sm)
# Affichage des résultats
y_pred = sm_over_.predict(X_val)
print(classification_report_imbalanced(y_val, y_pred))
```

```
y_sm -Resampling SMOTE
    COUNT   RATIO
0  197870 50.0000
1  197870 50.0000
-----------------------------------------------
                   pre       rec       spe        f1       geo       iba       sup

          0       0.93      0.90      0.17      0.91      0.39      0.17     84816
          1       0.13      0.17      0.90      0.15      0.39      0.14      7438

avg / total       0.86      0.84      0.23      0.85      0.39      0.16     92254
```

Analyse exploratoire – Cost-Sensitive Learning

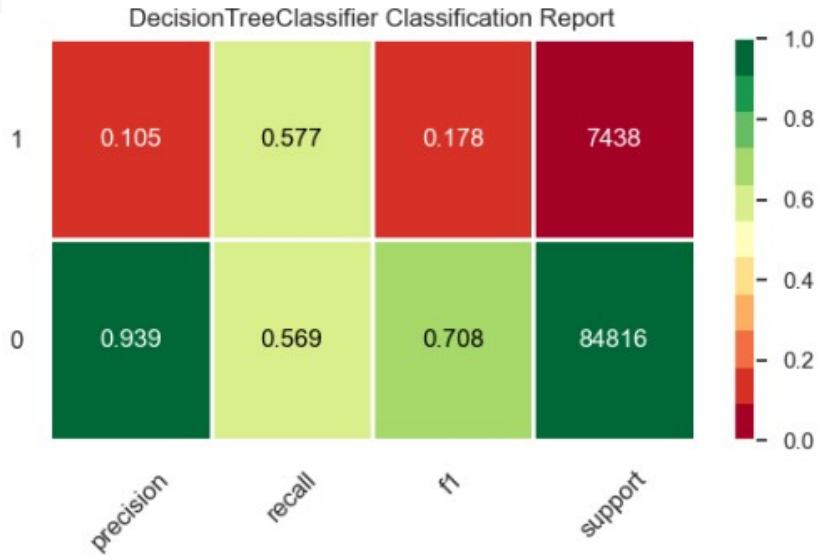
Prêt à dépenser

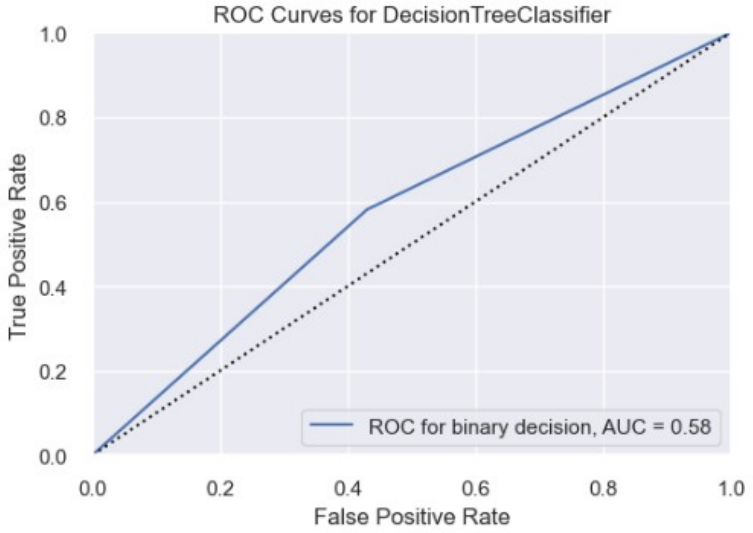
La classification pénalisée permet d'imposer un coût supplémentaire au modèle pour les erreurs de classification commises sur la classe minoritaire pendant l'entraînement. Ces pénalités peuvent biaiser le modèle pour qu'il accorde plus d'attention à la classe minoritaire.
Reprenons notre modèle d'arbre et appliquons lui un coût supplémentaire

```
model2=DecisionTreeClassifier(random_state=1234,class_weight={0:1, 1:10})
```

```
# Sous-échantillonnage
rUs = RandomUnderSampler()
X_ru, y_ru = rUs.fit_resample(X_train, y_train)

print("y_ru -Resampling target")
print(pd.DataFrame({
    "COUNT": y_ru.value_counts(),
    "RATIO": y_ru.value_counts() / len(y_ru) * 100}))
print('-----------------------------------------------')
# Entraînement du modèle d'arbre
rd_under_bal_=model2.fit(X_ru, y_ru)
# Affichage des résultats
y_pred = rd_under_bal_.predict(X_val)
print(classification_report_imbalanced(y_val, y_pred))

```

```
y_ru -Undersampling target
   COUNT   RATIO
0  17387 50.0000
1  17387 50.0000
-----------------------------------------------
                   pre       rec       spe        f1       geo       iba       sup

          0       0.94      0.57      0.58      0.71      0.57      0.33     84816
          1       0.11      0.58      0.57      0.18      0.57      0.33      7438

avg / total       0.87      0.57      0.58      0.67      0.57      0.33     92254
```





III. Modélisation

Prêt à dépenser

J'ai utilisé une cross Validation cv=10 pour l'évaluation des modèles

modèle	Fit time	Durée	Train F1	Test F1	Train FBETA	Test FBETA	Train ROC_AUC	Test ROC_AUC	Train ACCURACY	Test ACCURACY
Dummy_Classifier	0.051093	0.021710	0.000000	0.000000	0.000000	0.000000	0.500000	0.500000	0.919227	0.919227
Logistic_Regression	0.699524	0.030148	0.011967	0.011565	0.007561	0.007307	0.725908	0.724768	0.919025	0.919013
Decision_Tree_Classifier	6.245317	0.040607	1.000000	0.142896	1.000000	0.151202	1.000000	0.532757	1.000000	0.847559
Random_forest Classifier	69.040850	1.387221	0.999776	0.002753	0.999642	0.001724	1.000000	0.707526	0.999964	0.919264
Bagging Classifier	56.898929	0.305051	0.900613	0.035326	0.850036	0.023186	0.999658	0.629017	0.985393	0.916839
Linear_SVC	98.407131	0.034709	0.000460	0.000459	0.000288	0.000287	0.726211	0.725059	0.919190	0.919180
XGB_Classifier	27.729241	0.085213	0.146445	0.051114	0.097672	0.033518	0.852534	0.731719	0.924751	0.918321
Gradient_Boosting_Classifier	113.782099	0.144258	0.015052	0.010805	0.009493	0.006811	0.747273	0.735978	0.919482	0.919213
LGBM-Classifier	3.112440	0.195128	0.040243	0.023355	0.025677	0.014863	0.794201	0.741952	0.920309	0.919190

En prenant en compte le déséquilibre des classes,les résultats sont meilleurs

modèle	Fit time	Durée	Train F1	Test F1	Train FBETA	Test FBETA	Train ROC_AUC	Test ROC_AUC	Train ACCURACY	Test ACCURACY
Dummy_Classifier_balanced	0.054079	0.026107	0.000000	0.000000	0.000000	0.000000	0.500000	0.500000	0.919227	0.919227
Logistic_Regression_balanced	0.676804	0.036756	0.247414	0.246938	0.394980	0.394282	0.726768	0.725487	0.677792	0.677432
Decision_Tree_Classifier_balanced	6.718395	0.047535	1.000000	0.138201	1.000000	0.139127	1.000000	0.531089	1.000000	0.859201
Random_forest Classifier_balanced	77.899738	1.673278	0.999719	0.001608	0.999550	0.001006	1.000000	0.710965	0.999955	0.919245
Bagging Classifier_balanced	56.090249	0.298299	0.900613	0.035326	0.850036	0.023186	0.999658	0.629017	0.985393	0.916839
Linear_SVC_balanced	118.143720	0.057929	0.253522	0.253796	0.389811	0.390055	0.723965	0.723021	0.710979	0.711322
XGB_Classifier_balanced	47.856769	0.117413	0.186907	0.168497	0.364926	0.329448	0.859862	0.697027	0.297412	0.276841
Gradient_Boosting_Classifier_balanced	107.048856	0.139634	0.015052	0.010805	0.009493	0.006811	0.747273	0.735978	0.919482	0.919213
LGBM_Classifier_balanced	3.215434	0.207417	0.289864	0.261739	0.453486	0.409283	0.792369	0.742679	0.712227	0.701199

Je conserve uniquement les 3 meilleurs modèles pour affiner les paramètres de class_weight. Il s'agit de :

LGBM_Classifier_balanced, Logistic_Regression_balanced, XGB_Classifier_Balanced

linear_SVC_balanced a été écarté pour des questions pratiques de temps de calcul trop long

Tuning des hyper-paramètres du Cost Sensitive learning sur les 4 modèles retenus

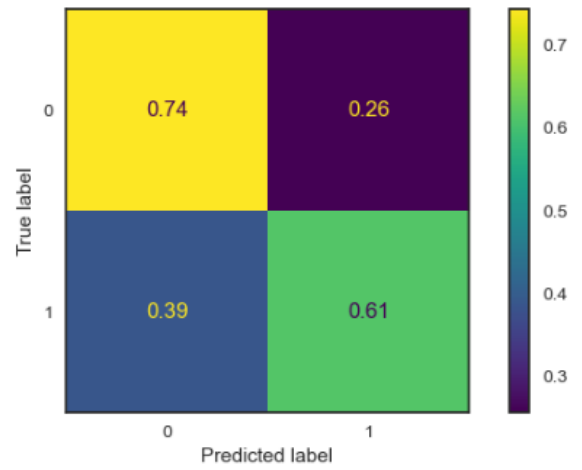
Prêt à dépenser

```
# paramètres
seed=1234
lr_param_grid = [{'LR__class_weight': ['balanced',{0: 50, 1: 1},{0: 10, 1: 1}, {0: 1, 1: 10},{0: 1, 1: 50}]}]

#svm_param_grid = [{'SVM__class_weight': ['balanced',{0: 50, 1: 1},{0: 10, 1: 1}, {0: 1, 1: 10},{0: 1, 1: 50}]}]

lgbm_param_grid=[{'LGBM__scale_pos_weight':[99,50,10]}]

xgb_param_grid = [{'XGB__scale_pos_weight': [1, 10, 25, 50, 75, 99,100,1000]}]

#Pipelines
pipe_lgbm = Pipeline([('LGBM', LGBMClassifier(random_state=seed))])

pipe_lr = Pipeline([('LR',LogisticRegression(random_state=seed))])

#pipe_svm = Pipeline([('SVM', SVC(random_state=seed))])

pipe_xgb = Pipeline([('XGB', XGBClassifier(random_state=seed))])
```

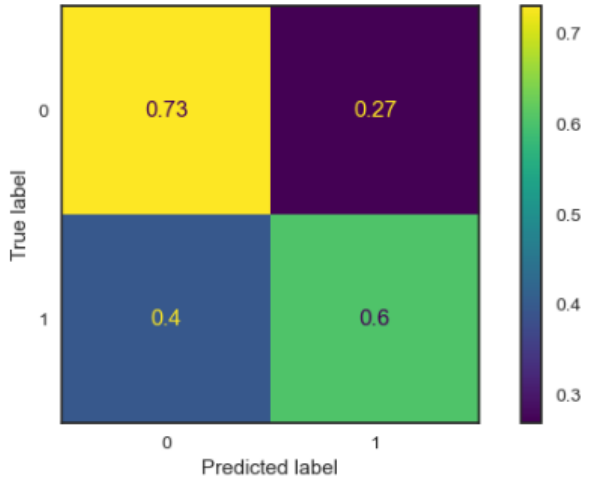
```
scoring = {'f1': 'f1','roc_auc':'roc_auc','accuracy':'accuracy',
           'fbeta' : make_scorer(custom_score)}

lgbm_grid_search = GridSearchCV(estimator=pipe_lgbm,param_grid=lgbm_param_grid,  scoring='f1',   cv=3)

lr_grid_search = GridSearchCV(estimator=pipe_lr, param_grid=lr_param_grid, scoring='f1', cv=3)

#svm_grid_search = GridSearchCV(estimator=pipe_svm, param_grid=svm_param_grid, scoring='f1', cv=3)

xgb_grid_search = GridSearchCV(estimator=pipe_xgb, param_grid=xgb_param_grid, scoring='f1', cv=3)
```

```
grids = [lgbm_grid_search, lr_grid_search, xgb_grid_search]  #, svm_grid_search
for pipe in grids:
    pipe.fit(X_train,y_train)
```

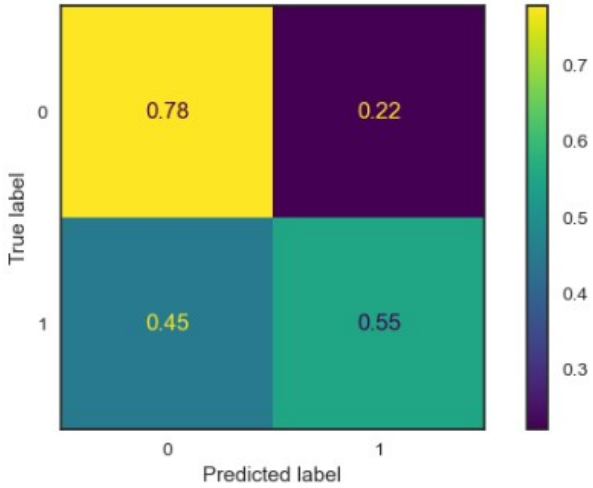
```
LightGBM Valid f1: 0.27
LightGBM Best Params: {'LGBM__scale_pos_weight': 10}
```




```
Logistic Regression Valid f1: 0.26
Logistic Regression Best Params: {'LR__class_weight': {0: 1, 1: 10}}
```




```
XGBoost Valid f1: 0.27
XGBoost Best Params: {'XGB__scale_pos_weight': 10}
```

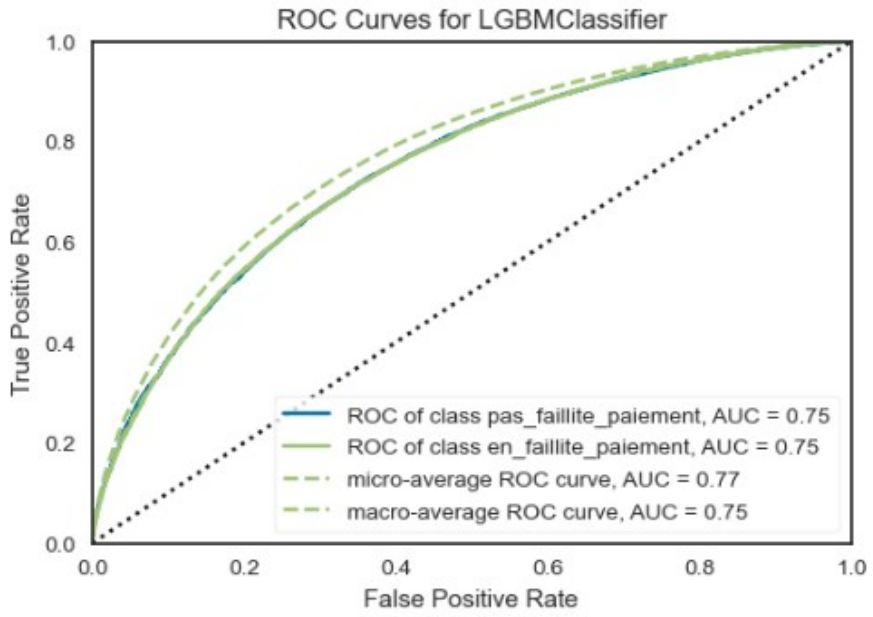


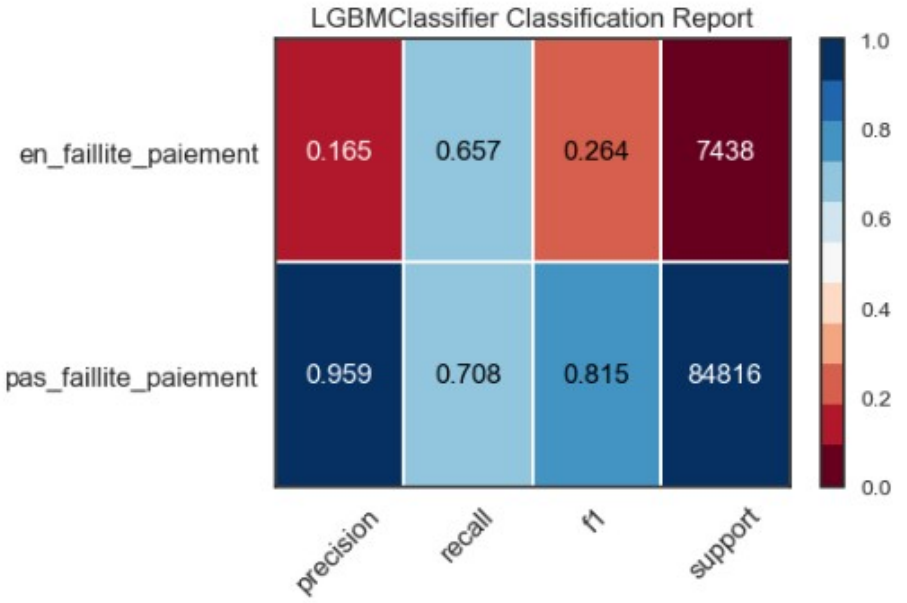
Le modèle LightGBM donne le meilleur résultat avec un sensitive cost de 10

Tuning des hyper-paramètres de LightGBM

```
lgb_model = LGBMClassifier(boosting_type='gbdt',
                           learning_rate=0.15,
                           max_depth=5,
                           num_leaves=2**5,
                           n_estimators=100,
                           objective='binary',
                           class_weight="balanced",
                           random_state=1234,
                           subsample=0.1,
                           reg_alpha=10,
                           reg_lambda=10,

                           )
```

```
Train Accuracy : 71.46%
Test Accuracy : 70.41%

Train Precision : 18.12%
Test Precision : 16.48%

Train Recall : 72.01%
Test Recall : 65.68%

Train f1_score : 28.95%
Test f1_score : 26.35%

Train f2_score : 45.15%
Test f2_score : 41.13%

Train ROC-AUC : 71.71%
Test ROC-AUC : 68.25%
```





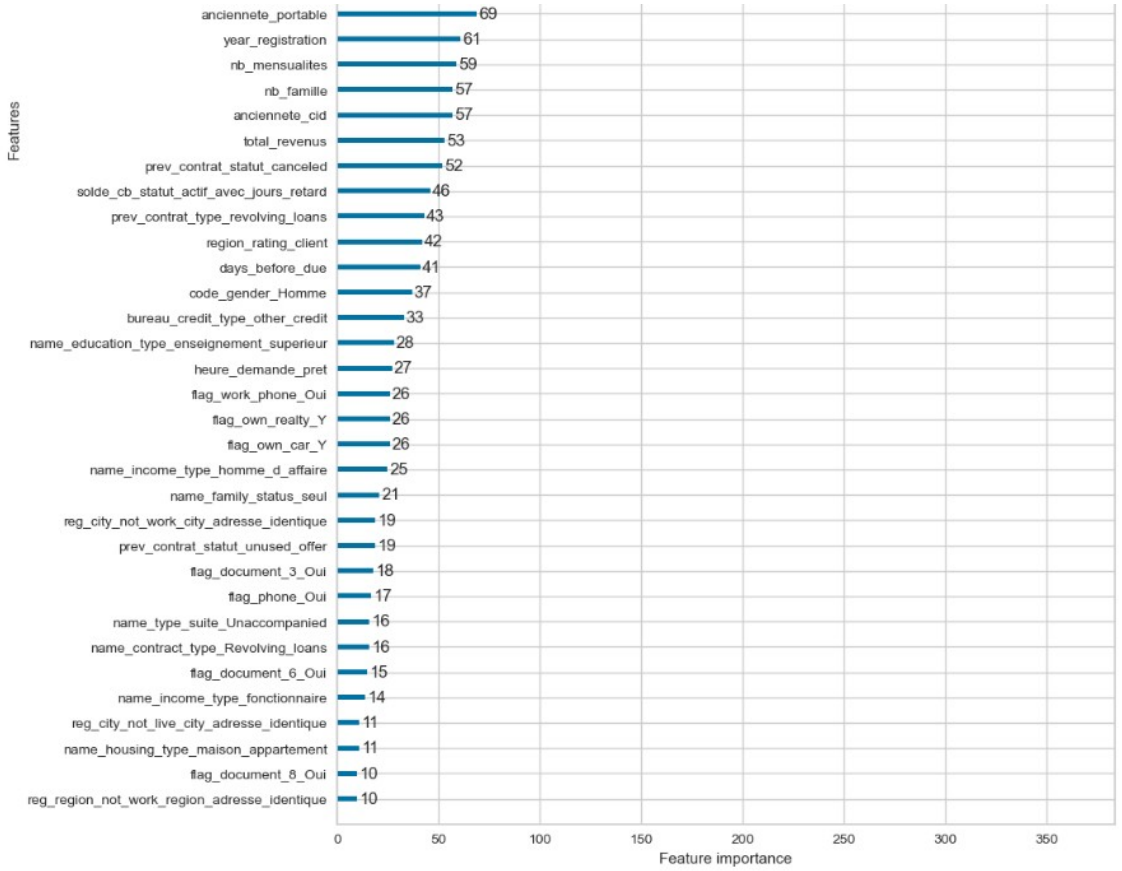

```
<Axes: title={'center': 'LGBMClassifier Classification Report'}>
```

IV. Optimisation

Prêt à dépenser

Avec la fonction feature_importances_ , on peut avoir une valeur qui reflète l'importance de chaque variable.

Avec plot_importance , on trace le graphique avec les variables et les valeurs de coéfficient d'importance



Extrait du graphe plot_importance

Tuning avec OPTUNA

Prêt à dépenser

Optuna est un outil de recherche automatisé permettant l'optimisation des hyper-paramètres des modèles en Machine Learning.

Il faut au préalable définir une fonction coût.La fonction de perte ou de coût est la quantification de l'écart entre les prévisions du modèle et les observations réelles du jeu de donnée utilisé pendant l'entraînement. Une valeur Log Loss inférieure signifie de meilleures prédictions ; la valeur Log Loss d'un modèle parfait serait de 0.

```
param_grid = {
    "objective": "binary",
    "class_weight": "balanced",
    "verbose": -1,
    "boosting_type": "gbdt",
    "n_estimators": trial.suggest_int("n_estimators", 50, 500, step=50),        # N
    "learning_rate": trial.suggest_float("learning_rate", 1e-7, 1.0, log=True), # B
    "max_depth": trial.suggest_int("max_depth", 2, 16, step=2),                 # Max
    "num_leaves": trial.suggest_int("num_leaves", 3, 12, step=2),               # M
    "min_data_in_leaf": trial.suggest_int("min_data_in_leaf", 200, 10000, step=100)
    "min_child_samples": trial.suggest_int("min_child_samples", 5, 100),        # M
    "lambda_l1": trial.suggest_loguniform("lambda_l1", 1e-8, 10.0),
    "lambda_l2": trial.suggest_loguniform("lambda_l2", 1e-8, 10.0),
    "reg_alpha": trial.suggest_loguniform( "reg_alpha",1e-9, 100.0,),       # L1 re
    "min_gain_to_split": trial.suggest_float("min_gain_to_split", 0, 15),
    "subsample":trial.suggest_uniform("subsample", 0.01, 1.0),              # Subsam
    "subsample_freq": trial.suggest_int( "subsample_freq", 0, 10),           # Fre
    "min_child_weight": trial.suggest_uniform("min_child_weight", 0.01, 10.0),
    "feature_fraction": trial.suggest_uniform("feature_fraction", 0.4, 1.0),
    "bagging_fraction": trial.suggest_uniform("bagging_fraction", 0.4, 1.0),
    "bagging_freq": trial.suggest_int("bagging_freq", 1, 7),
    "max_bin":trial.suggest_int ("max_bin",10, 1000),                        # Max n
    "colsample_bytree": trial.suggest_float("colsample_bytree", 0.3, 0.8 , step=1),
}
```

```
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=1234)

cv_scores = np.empty(5)

for idx, (train_index, test_index) in enumerate(cv.split(X, y)):

    X_train, X_val = X.take(list(train_index),axis=0), X.take(list(test_index),axis=0)
    y_train, y_val = y.take(list(train_index),axis=0), y.take(list(test_index),axis=0)

    model = lgbm.LGBMClassifier(**param_grid, n_jobs=-2, random_state=1234)

    model.fit(
        X_train,
        y_train,
        eval_set=[(X_val, y_val)],
        eval_metric="binary_logloss",
        #early_stopping_rounds=100,
        callbacks=[LightGBMPruningCallback(trial, "binary_logloss")])  # Add a pruning callback

    predics = model.predict_proba(X_val)
    cv_scores[idx] = log_loss(y_val, predics)

return np.mean(cv_scores)
```

```
Train Accuracy : 75.75%
Val Accuracy : 72.65%

Train Precision : 22.17%
Val Precision : 16.73%

Train Recall : 79.77%
Val Recall : 60.11%

Train f1_score : 34.70%
Val f1_score : 26.17%

Train f2_score : 52.50%
Val f2_score : 39.58%

Train ROC-AUC : 77.58%
Val ROC-AUC : 66.93%
```

Le modèle est sauvegardé avec joblib

```
from joblib import dump, load
dump(model_LGBM, open('LGBM5_saved.joblib','wb'))
```

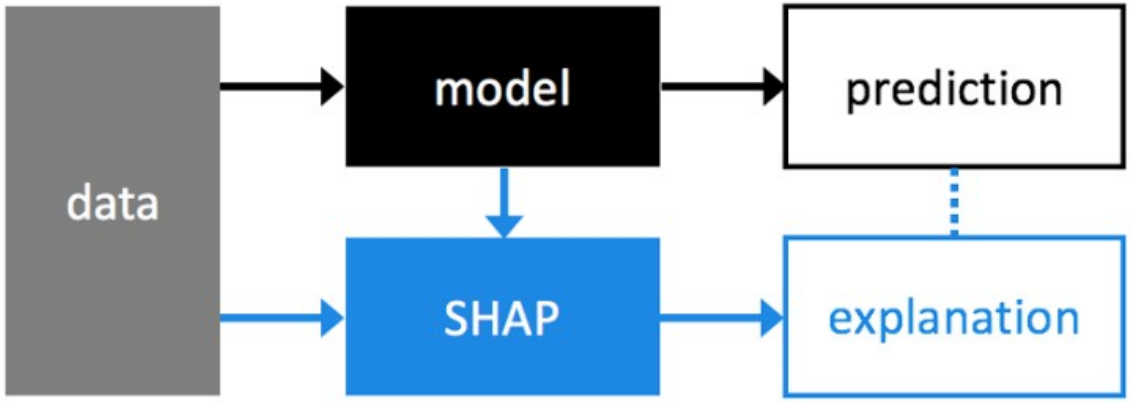
Le modèle est utilisé pour des prédictions sur le jeu de test

Prédictions sur l'ensemble de test

y_test_pred	COUNT	RATIO
0	34881	71.559577
1	13863	28.440423

Interprétation avec SHAP

Cette technique permet d'expliquer les prédictions des modèles de Machine Learning de manière compréhensible aux humains. En attribuant une valeur à chaque caractéristique entrée, il montre comment et dans quelle mesure chaque caractéristique a contribué au résultat final de la prédiction. Il explique comment le modèle a pris sa décision et permet d'identifier les caractéristiques les plus importantes. Il fait cela en calculant la contribution moyenne de chaque caractéristique à la prédiction sur toutes les combinaisons possibles de caractéristiques.



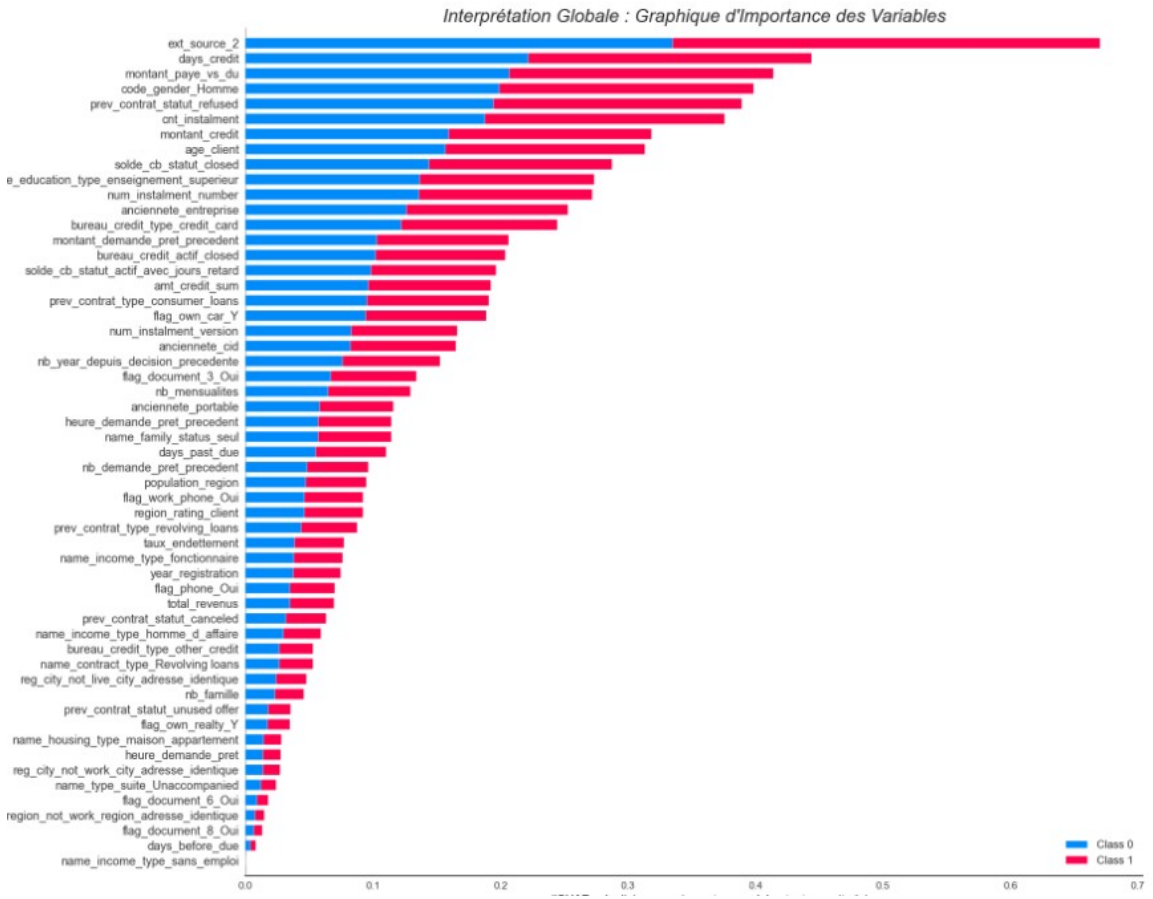
Les shap_values représentent la contribution de chaque caractéristique à la prédiction faite par le modèle pour chaque instance du dataset .

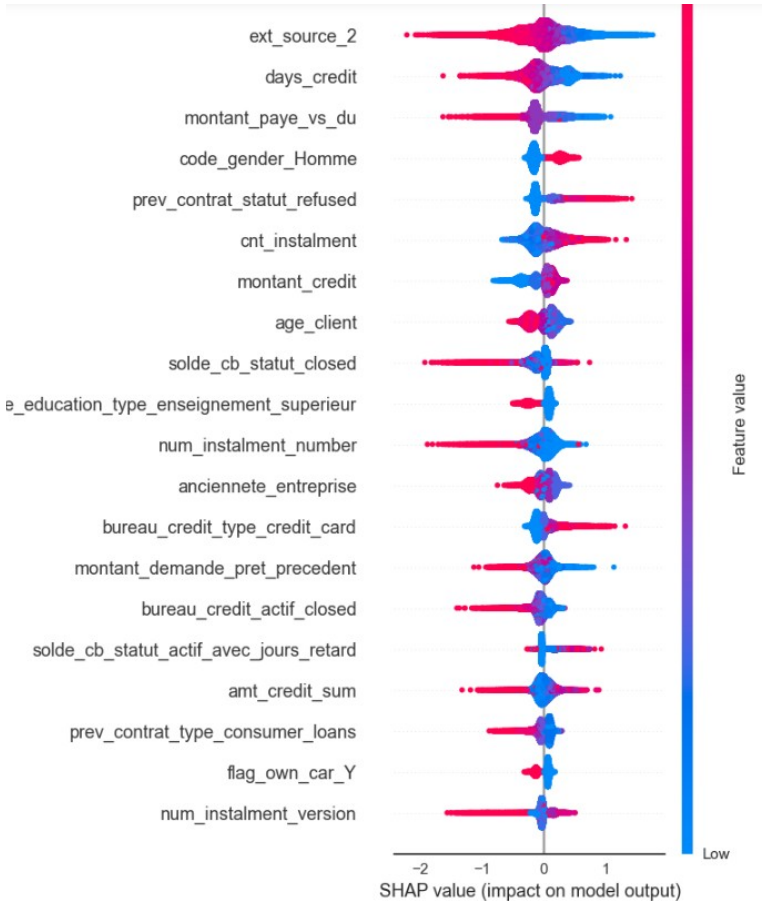
Les valeurs de Shapley calculent l'importance d'une variable en comparant ce qu'un modèle prédit avec et sans cette variable. Cependant, étant donné que l'ordre dans lequel un modèle voit les variables peut affecter ses prédictions,cela se fait dans tous les ordres possibles, afin que les fonctionnalités soient comparées équitablement. Cetteapproche est inspirée de la théorie des jeux.

Interprétabilité globale avec SHAP

Prêt à dépenser

Shap permet à la fois une interprétation globale et une interprétation locale





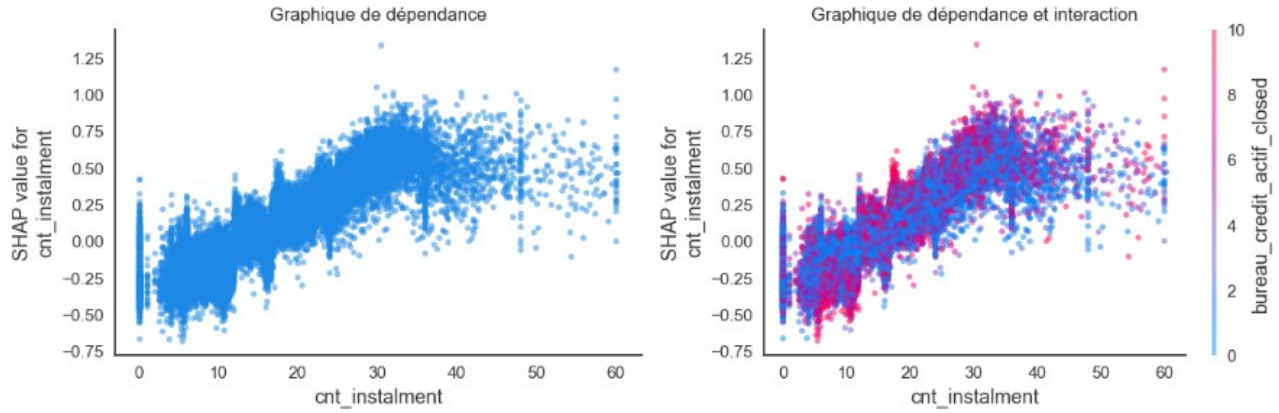
On peut regarder l'importance de chaque caractéristique pour la classe 1

L'interprétation globale cherche à expliquer le modèle dans sa globalité. C'est-à-dire quelles sont les variables les plus importantes pour le modèle. Dans notre cas, quelles caractéristiques affectent le plus le comportement général d'un modèle d'octroi de crédit ?

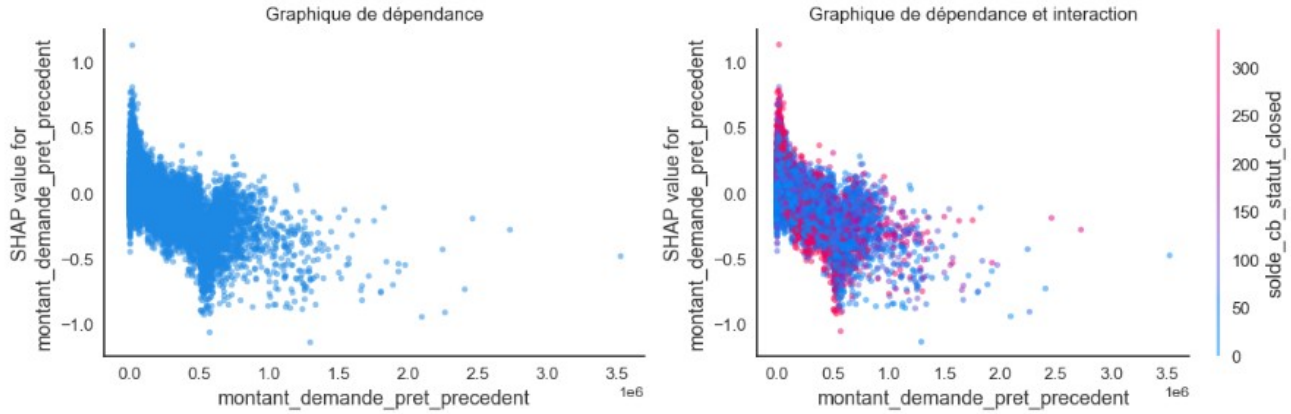
Impact d'une caractéristique sur le modèle avec SHAP

Prêt à dépenser

Impact de chaque caractéristique sur le modèle



Plus la durée du crédit précédent est importante , plus la valeur SHAP augmente et donc plus le modèle prédit que le client aura des difficulté de paiement

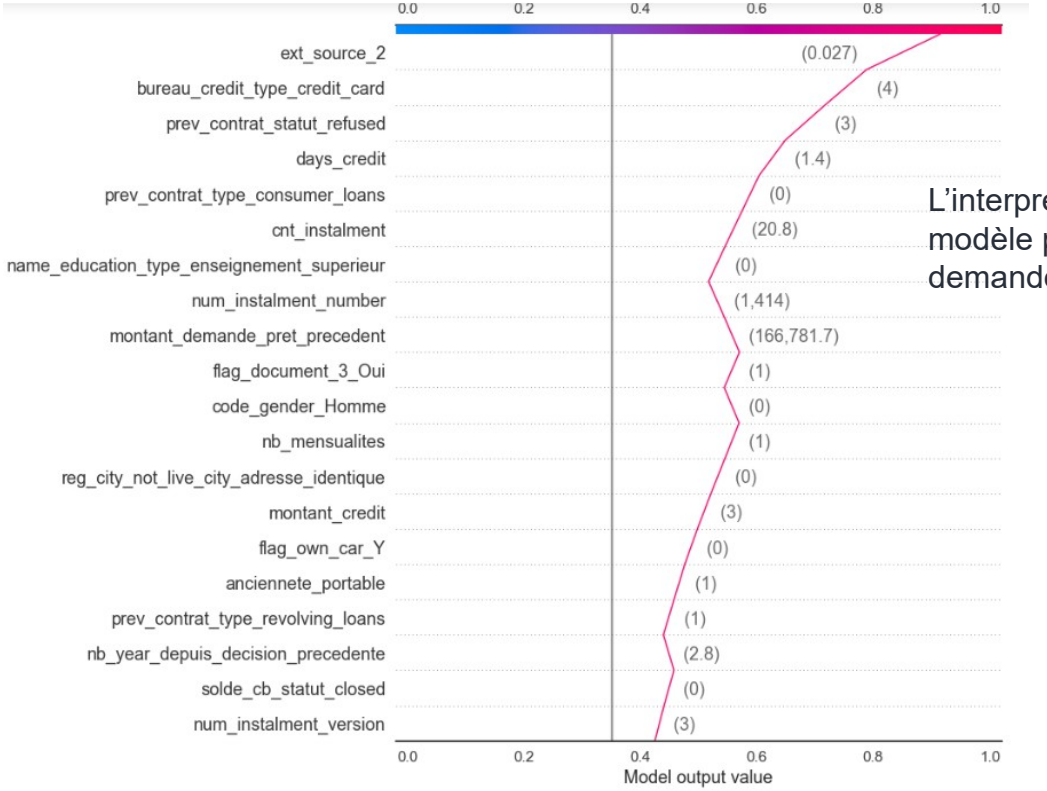


Plus le montant de la demande de prêt précédente est faible,plus la valeur SHAP est importante et donc plus le modèle prédit que le client aura des difficultés de paiement.

Interprétabilité locale avec SHAP

Prêt à dépenser

Le graphique decision_plot est une façon d'expliquer la prédiction. Il donne la valeur et l'impact sur le score de chaque variable par ordre d'importance Il indique la direction prise par la prédiction pour chacune des valeurs des variables affichées.



L'interprétation locale consiste à expliquer la prédiction f(x) du modèle pour un individu x donné. Dans notre cas, pourquoi la demande de prêt du client a-t-elle été approuvée ou rejetée ?

Un graphique force-plot situe la prédiction par rapport à la base value.

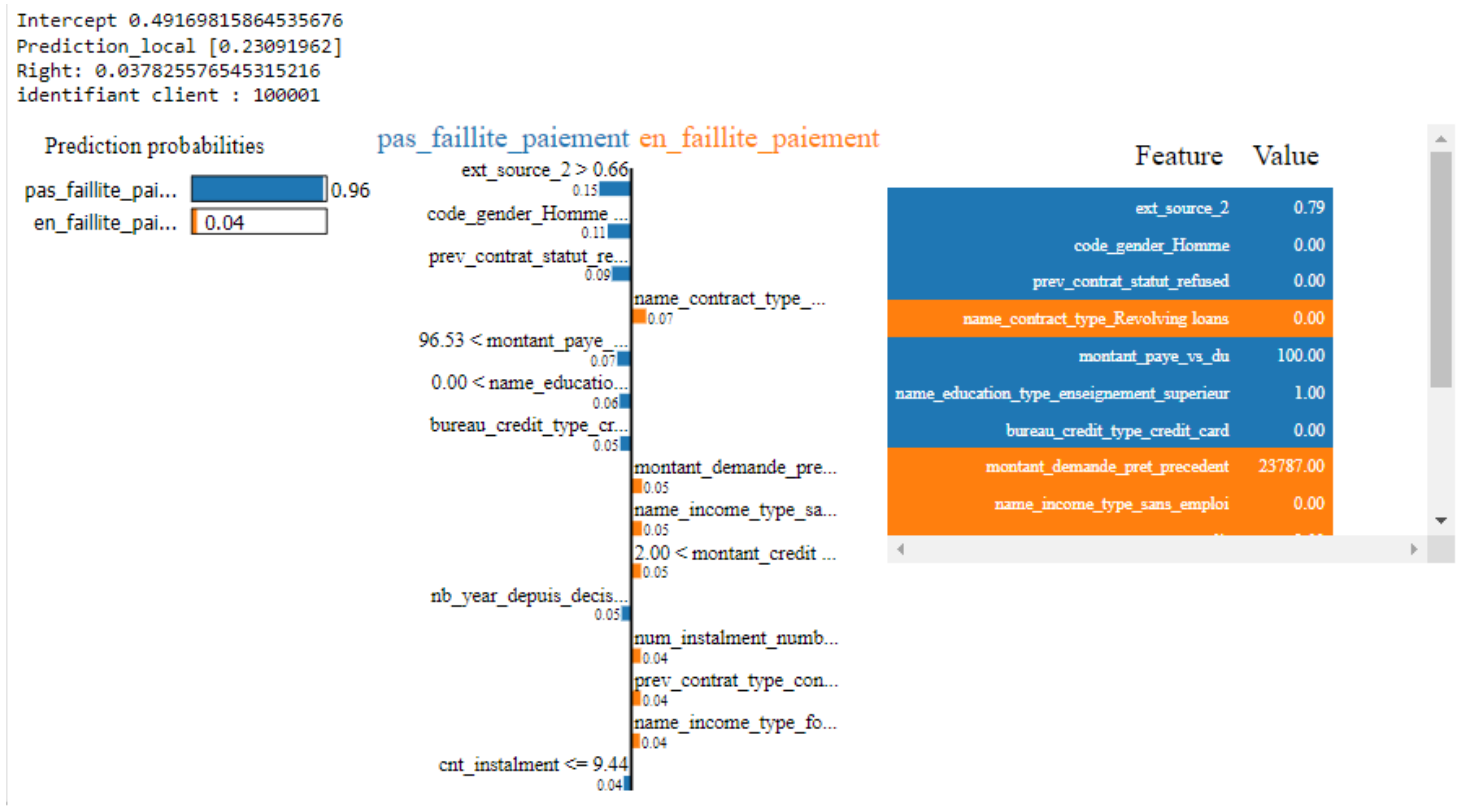
```
shap.plots.force(explainer.expected_value[1], shap_values[1][20, :], X_test.iloc[20, :],matplotlib = True)
```

Il indique les variables qui augmentent la probabilité du client d'être en défaut de paiement (en rouge) et celles qui la diminuent (en bleu), ainsi que la grandeur de cet impact.

Interprétabilité locale avec Lime

Prêt à dépenser

LIME est une technique qui permet de créer un modèle simple autour de la prédiction que nous voulons expliquer et d'utiliser ce modèle pour expliquer la prédiction. En pratique, pour une prédiction donnée , LIME va pertuber les entrées de la prédictions et regarder comment la pertubation a affecté la sortie du modèle. Les attributs qui affectent le plus la sortie lorsqu'ils sont pertubés sont considérés comme ayant une importance élevée pour cette prédiction .On est dans de l'**INTERPRETABILITE LOCALE**



Les 2 techniques donnent des résultats très similaires. LIME est indépendant du modèle et fournit des explications locales et interprétables, tandis que SHAP est spécifique au modèle et fournit des explications globales en utilisant une approche basée sur la théorie des jeux.

V. Dashboard

Prêt à dépenser

Le dashboard a été développé avec streamlit.

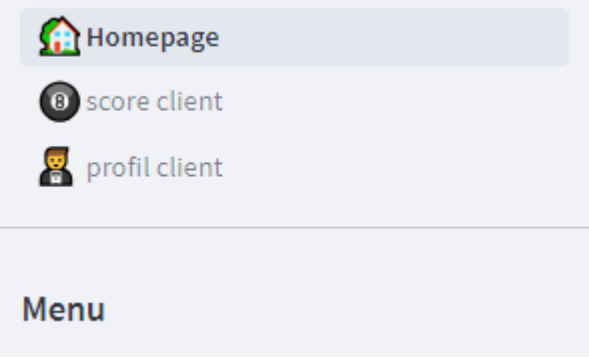
Ce Dashboard a été conçu pour une utilisation en agence. Il permet d'une part d'apporter au client la réponse à sa demande de prêt et d'autre part en cas de réponse négative ,de justifier notre décision en mettant en évidence les éléments concrets qui ont motivé le refus d'octroyer le crédit.

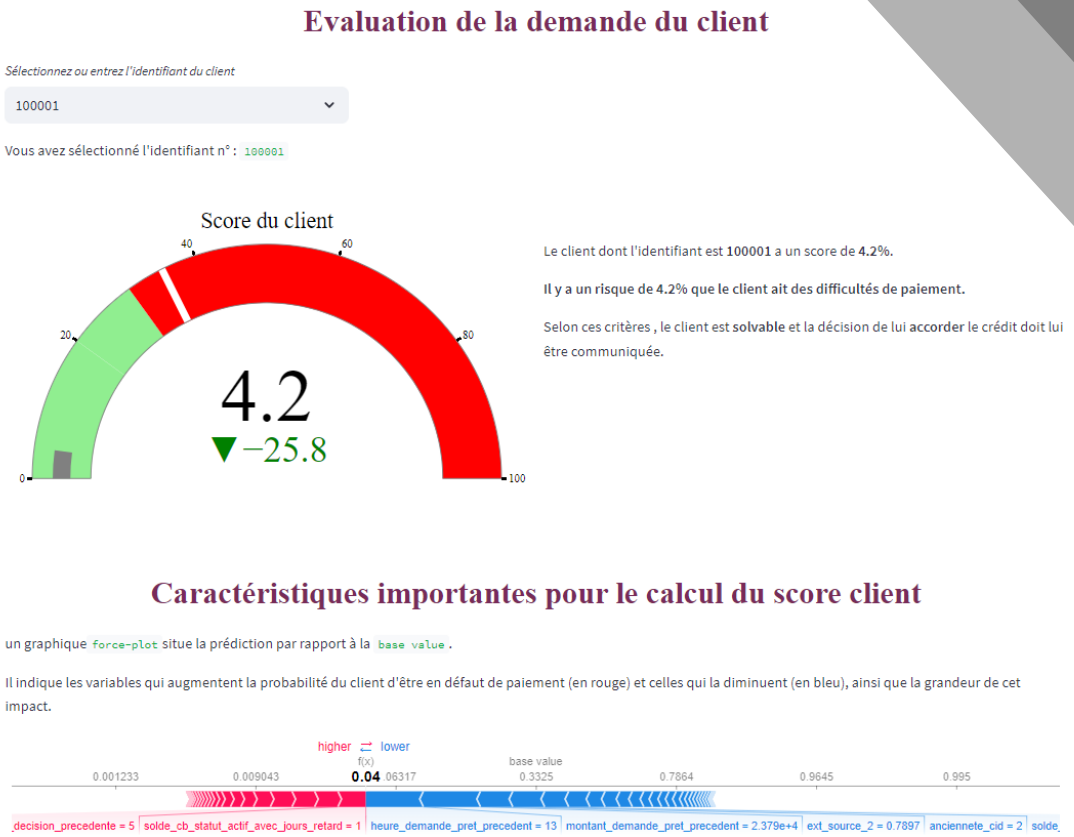
Chaque demande de crédit est soumise à notre modèle de prédiction.Ce modèle a été entrainé avec un jeu de données de 307.511 clients pour lesquels nous savions déjà s'ils avaient ou pas rencontré des difficultés à rembourser leur crédit.
En réponse à chaque demande de crédit ,ce modèle calcule un score qui représente le pourcentage de risque que le client rencontre des difficultés pour rembourser son crédit. En fonction de la valeur de ce score, le prêt sera accordé ou refusé.

Comment l'utiliser ?

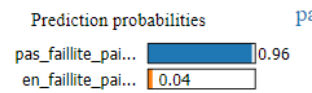
Cliquez sur l'onglet "Score client" et sélectionnez l'identifiant du client dans la liste déroulante. Le score obtenu par le client s'affichera sur la jauge ainsi que la réponse à communiquer au client .Les variables les plus importantes avec leur poids sont mentionnées sous forme de graphiques

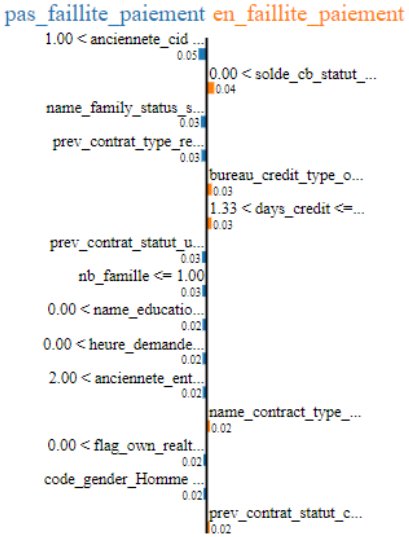
Cliquez sur l'onglet "Profil client" et sélectionnez l'identifiant du client dans la liste déroulante. Les données du profil géneral du client ainsi que certaines données financières relatives aux précédents crédits contractés par le client.



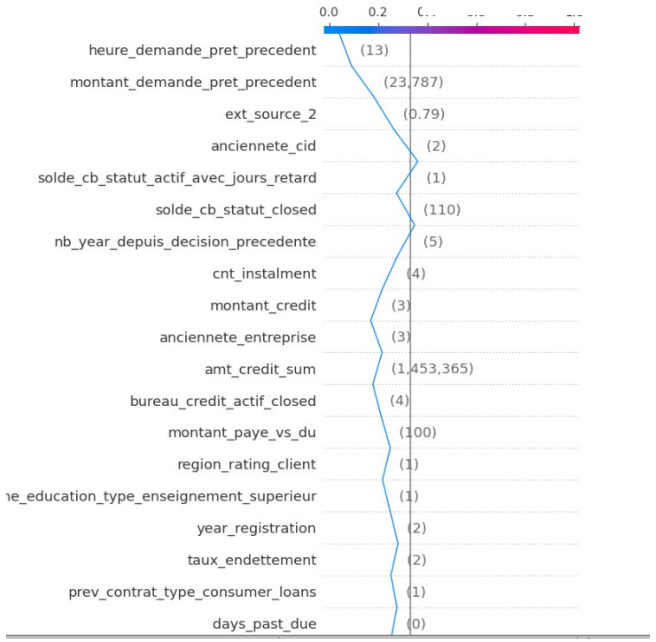


Copie d'ecran du dashboard





Feature	Value
anciennete_cid	2.00
solde_cb_statut_actif_avec_jours_retard	1.00
name_family_status_seul	0.00
prev_contrat_type_revolving_loans	0.00
bureau_credit_type_other_credit	0.00
days_credit	1.86
prev_contrat_statut_unused offer	0.00
nb_famille	1.00
name_education_type_enseignement_superieur	1.00
heure_demande_pret	1.00
anciennete_entreprise	3.00
name_contract_type_Revolving loans	0.00



Données financières de l' emprunteur

Genre : **Femme**

Tranche d'âge : **51_à_58_ans**

Ancienneté de la pièce d'identité : **11_et_plus**

Situation familiale : **couple**

Taille de la famille : **2_personnes**

Nombre d'enfants : **sans_enfant**

Niveau d'études : **enseignement_superieur**

Revenu Total Annuel : **99001_à_135000 $**

Type d'emploi : **employé**

Ancienneté dans l'entreprise actuelle : **6_à_9_ans**

Type d'habitation : **maison_ou_appartement**

Densité de la Population de la région où vit le client : **densité_moyenne**

Evaluation de *'Prêt à dépenser'* de la région où vit le client : **note_de_2**

Type de Crédit demandé par le client : **Cash loans**

Montant du Crédit demandé par le client : **513532_à_679500 $**

Durée de remboursement du crédit : **25_mois_et_plus**

Taux d'endettement : **15_à_19%**

Score normalisé du client à partir d'une source de données externe : **79.0%**

Nombre de demande de prêt réalisée par le client : **1**

Montant des demandes de prêt précédentes du client : **23787.0 $**

Montant payé vs Montant attendu en % : **100.0%**

Durée mensuelle moyenne des crédits précédents : **4.0 mois**

Nombre de Crédit à la Consommation précédent du client : **1**

Nombre de Crédit Revolving précédent du client : **0**

Nombre de Crédit précédent refusé : **0**

Nombre de crédits cloturés enregistrés au bureau du crédit : **4**

Nombre de crédits de type *'carte de crédit'* enregistrés au bureau du crédit : **0**

Nombre d'années écoulées depuis la décision précédente : **5 ans**

VI. Conclusion

Le data scientist dispose d'une panoplie d'outils pour l'aider dans la résolution de problèmes. Toutefois , l'expérience de la personne reste primordiale pour la partie nettoyage ,la sélection des variables ou pour fixer les valeurs de départ pour le tuning des hyper-paramètres.

Ce qui pourrait être amélioré :

Le problème principal a été le manque de connaissances des métiers du secteur bancaire.Le vocabulaire employé n'est pas toujours pleinement compris et cela peut avoir des répercussions sur le choix et la façon de traiter certaines caractéristiques.
Il est clair aussi que le manque de connaissance du métier n'a pas permis de peaufiner la feature engineering.
Le choix des métriques s'est effectué avec le postulat qu'il était préférable de perdre un client plutôt que d'octroyer un prêt à un client susceptible de ne pas rembourser. Le niveau de risque acceptable n'a pas été défini en prenant en compte l'aspect métier.

Les fichiers sont disponibles sur Github