

Implémentez un modèle de scoring

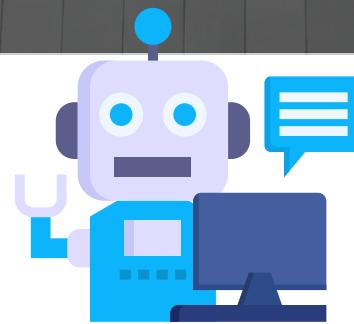
Projet 7

Sommaire

- La problématique
- Le jeu de données
- L'analyse exploratoire et feature engineering
- La méthodologie d'entraînement des modèles
- L'interprétabilité globale et locale du modèle
- L'analyse du Data Drift
- La présentation du dashboard
- Conclusion

La problématique de l'entreprise

"Prêt à dépenser"



Mettre en œuvre un outil de “scoring crédit”

Calculer la probabilité qu'un client rembourse son crédit, puis classifier la demande en crédit accordé ou refusé.

Développer un algorithme de classification

S'appuyer sur des sources de données variées (données comportementales, données provenant d'autres institutions financières, etc.)

Développer un dashboard interactif

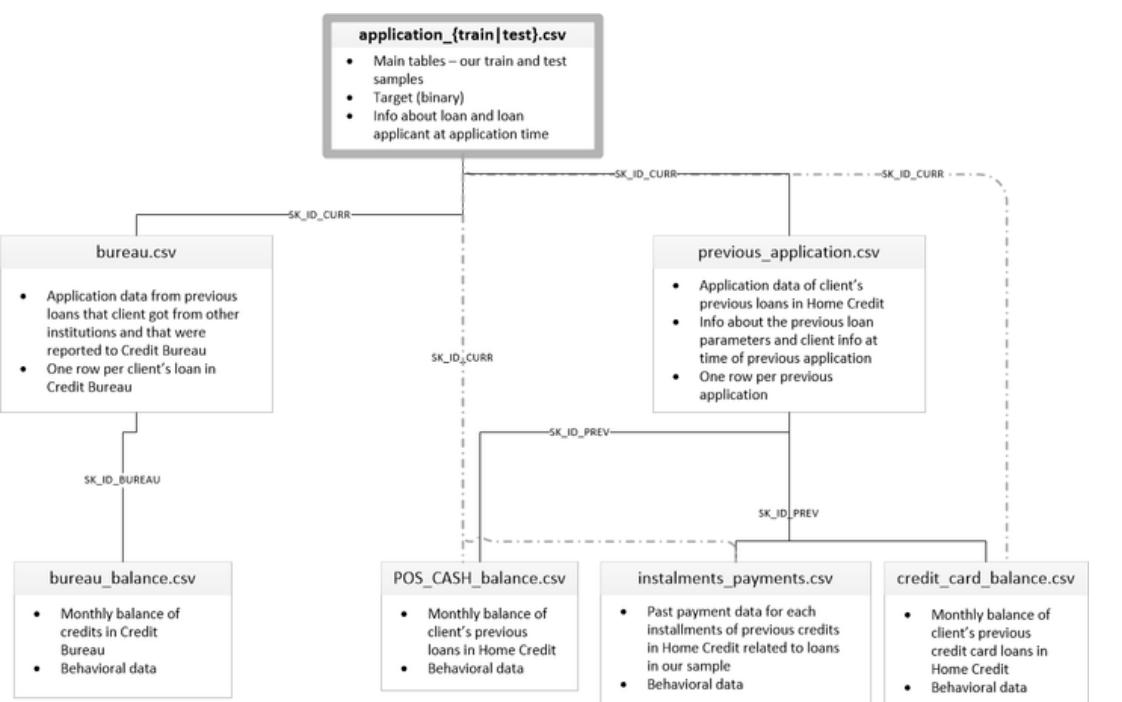
Pour expliquer de façon transparente les décisions d'octroi de crédit, et permettre aux clients de disposer de leurs informations personnelles et de les explorer facilement.

Les données

Kaggle: "Home Credit Default Risk"

```
[2]: application_train = pd.read_csv("application_train.csv")
application_test = pd.read_csv("application_test.csv")
bureau = pd.read_csv("bureau.csv")
bureau_balance = pd.read_csv("bureau_balance.csv")
credit_card_balance = pd.read_csv("credit_card_balance.csv")
installments_payments = pd.read_csv("installments_payments.csv")
previous_application = pd.read_csv("previous_application.csv")
POS_CASH_balance = pd.read_csv("POS_CASH_balance.csv")
```

- 307511 clients

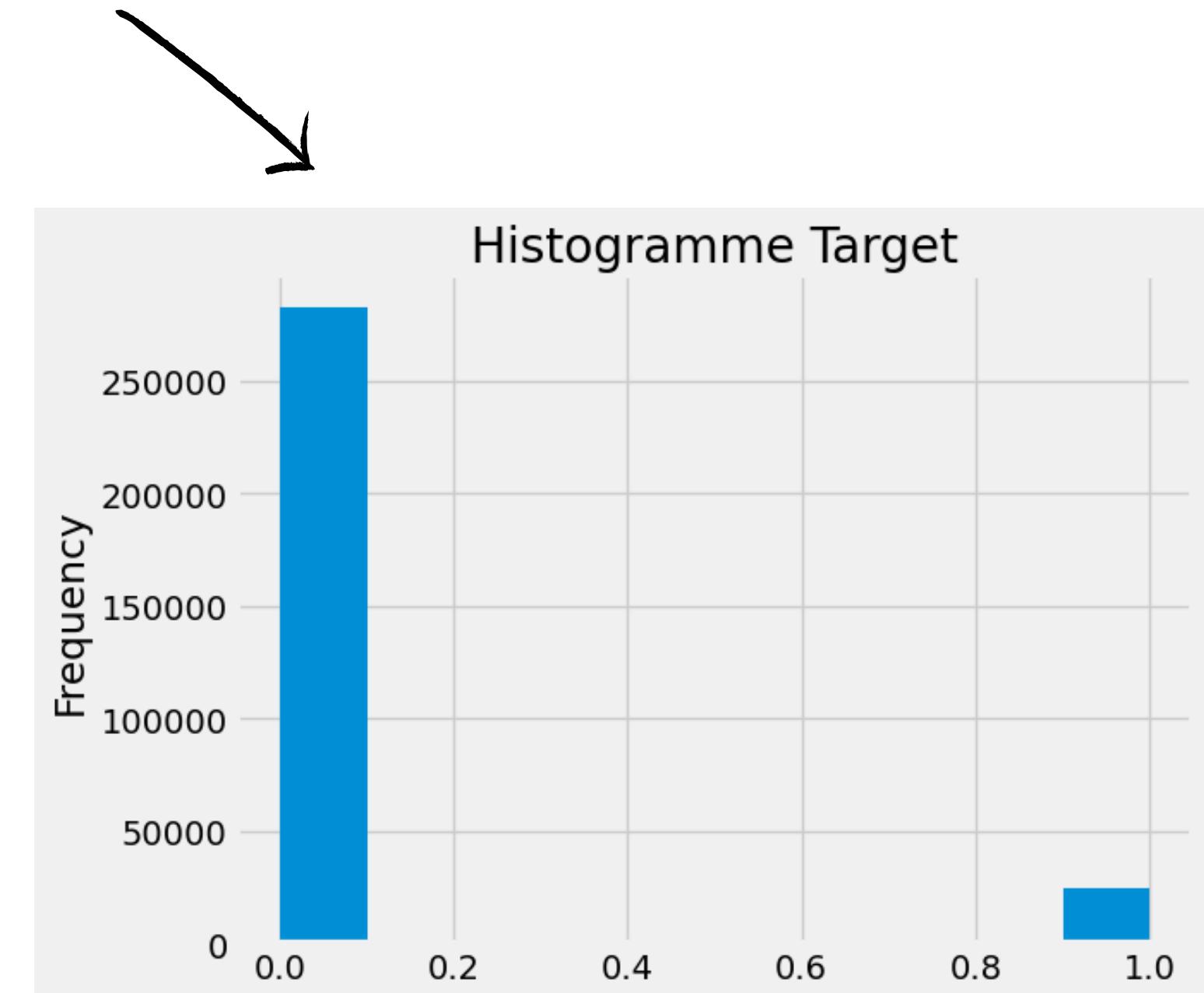


Analyse exploratoire

kernel: "LightGBM with Simple Features"

Pour élaborer un modèle performant, nous devons prendre en compte ce déséquilibre dans notre modélisation en appliquant un traitement de rééquilibrage.

Déséquilibre des classes



Feature engineering

kernel: "LightGBM with Simple Features"

- One-hot encoding des variables catégorielles,
- Préprocessing des différentes tables (NaN, agrégation, ...)
- Jointure des différentes tables

```
[27]: data=main(debug = False)
```

```
Train samples: 307511, test samples: 48744
Bureau df shape: (305811, 116)
Previous applications df shape: (338857, 249)
Pos-cash balance df shape: (337252, 18)
Installments payments df shape: (339587, 26)
Credit card balance df shape: (103558, 141)
```

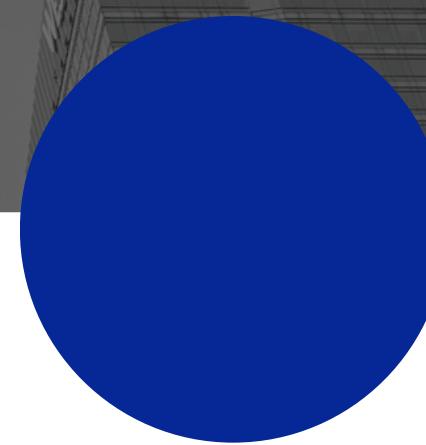
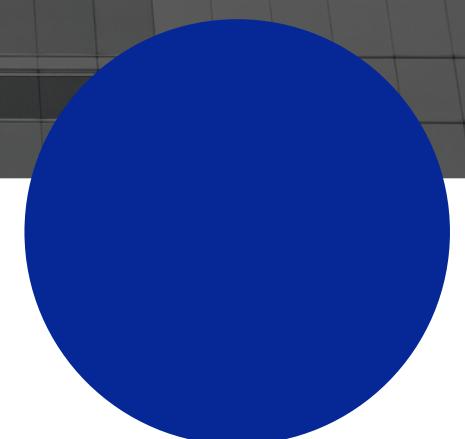
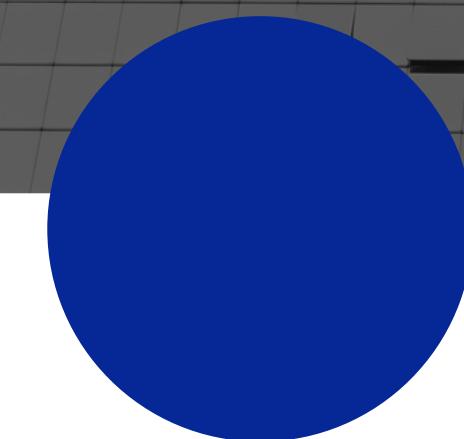
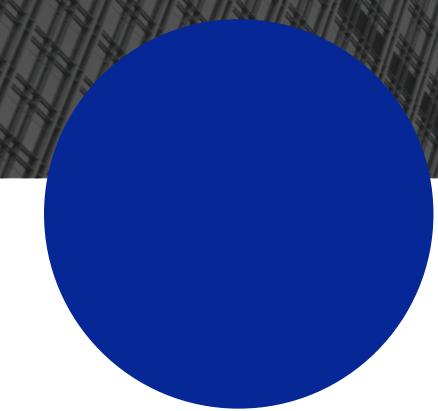
```
[31]: # Divide in training/validation and test data
```

```
train_df = df[df['TARGET'].notnull()]
test_df = df[df['TARGET'].isnull()]
print("Starting Train shape: {}, test shape: {}".format(train_df.shape,
```

```
Starting Train shape: (307507, 798), test shape: (48744, 798)
```



La méthodologie d'entraînement des modèles



Création d'un score métier

Le score “métier” consiste à calculer une fonction de coût métier de type $10*FN + FP$ (où FN = nombre de FN dans la matrice de confusion pour un seuil donné, FP = nombre de FP) et de trouver son minimum pour un seuil donné.

Le traitement du déséquilibre des classes

Méthode par oversampling, via la librairie Smote, pour générer de nouveaux enregistrements de classe 1. L'oversampling se fera pour chaque jeu de données entraîné via la pipeline imblearn.

Le développement et la simulation de modèles

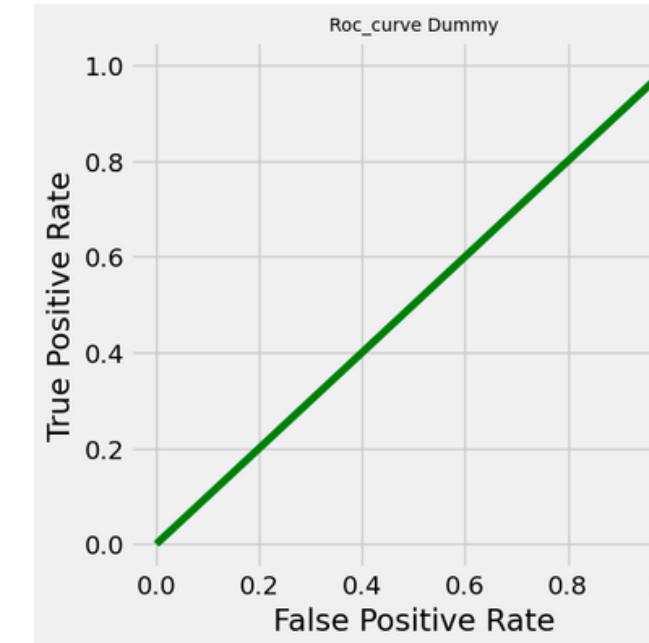
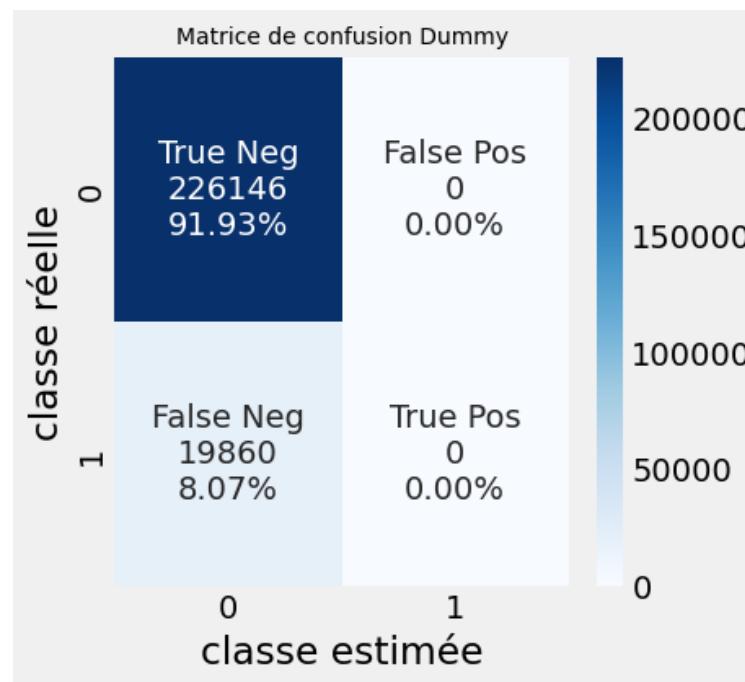
- DummyClassifier
- LogisticRegression
- DecisionTree Classifier
- LGBMClassifier
- CatBoostClassifier

L'interprétabilité globale et locale du modèle

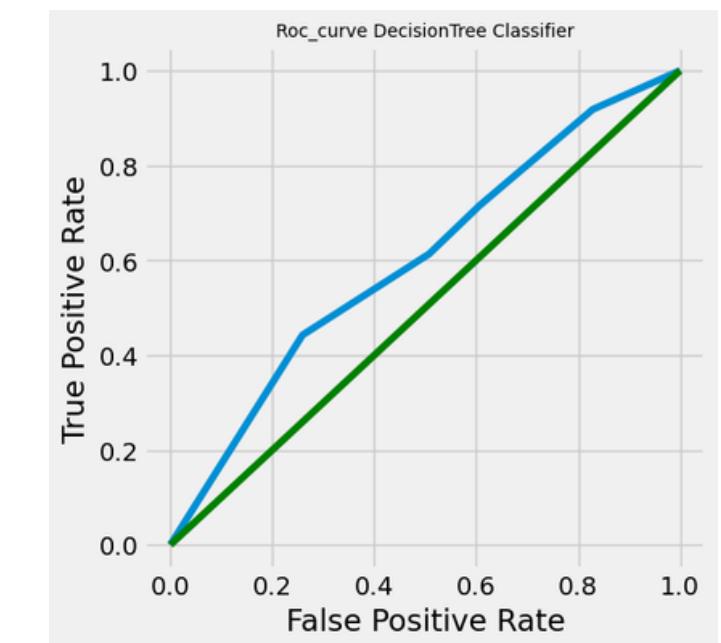
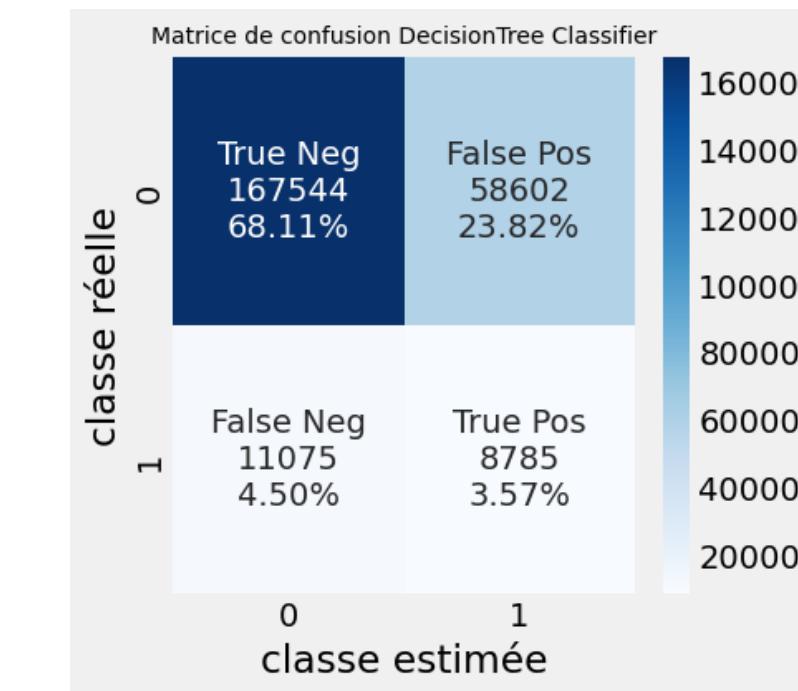
Pour connaître les principales features qui contribuent à l'élaboration du modèle, et pour le client, savoir qu'elle est l'influence de chaque feature dans le calcul de son propre score

Développement et simulation de modèles

```
----- Dummy -----  
Best params: Dummy {'dummy_strategy': 'most_frequent'}  
Score_metier 0.8072973829906588  
score_AUC_train 0.5
```

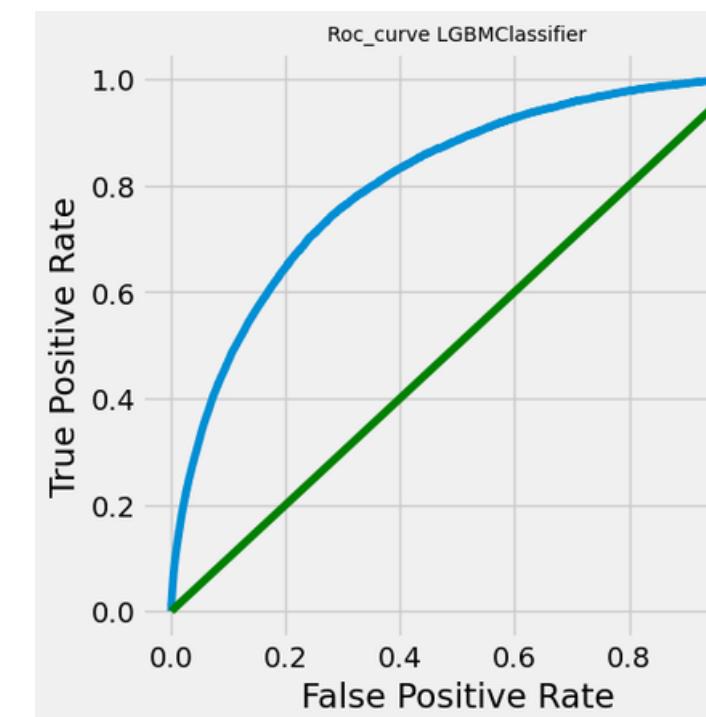
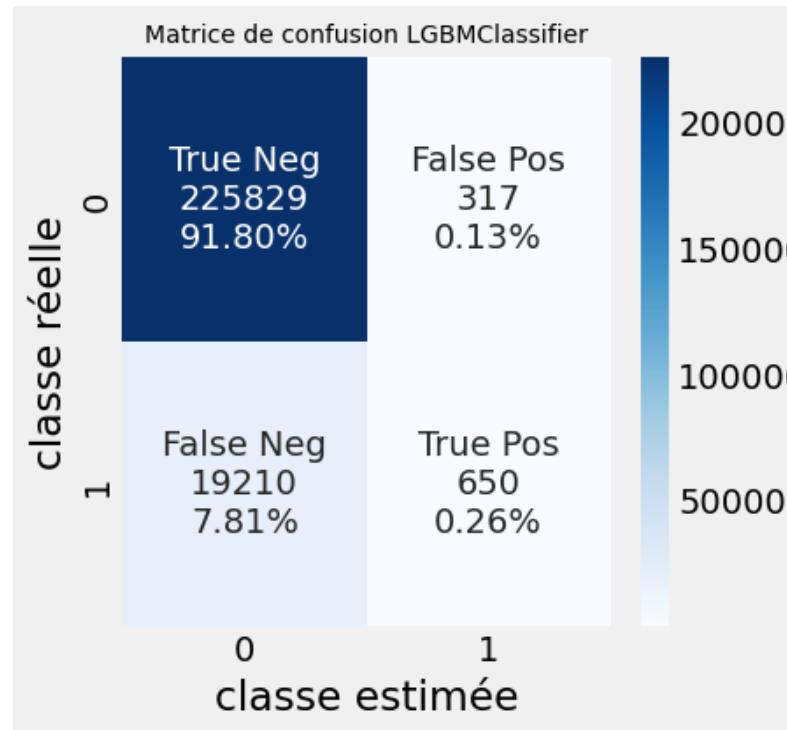


```
----- DecisionTree Classifier -----  
Best params: DecisionTree Classifier {'DecisionTree_criterion': 'entropy', 'DecisionTree_max_depth': 3, 'DecisionTree_min_samples_leaf': 5}  
Score_metier 0.6884059738380365  
score_AUC_train 0.5999673255357345
```

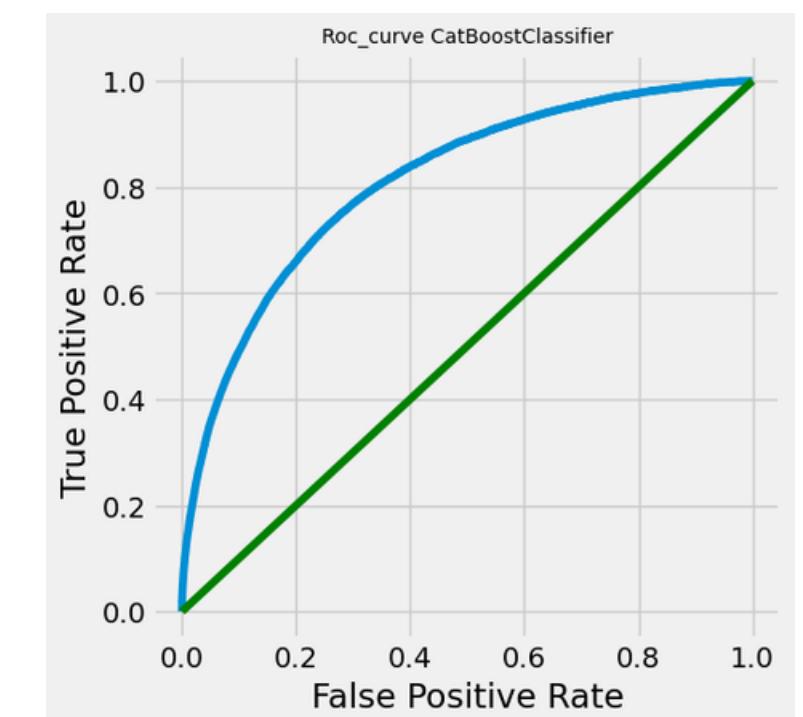
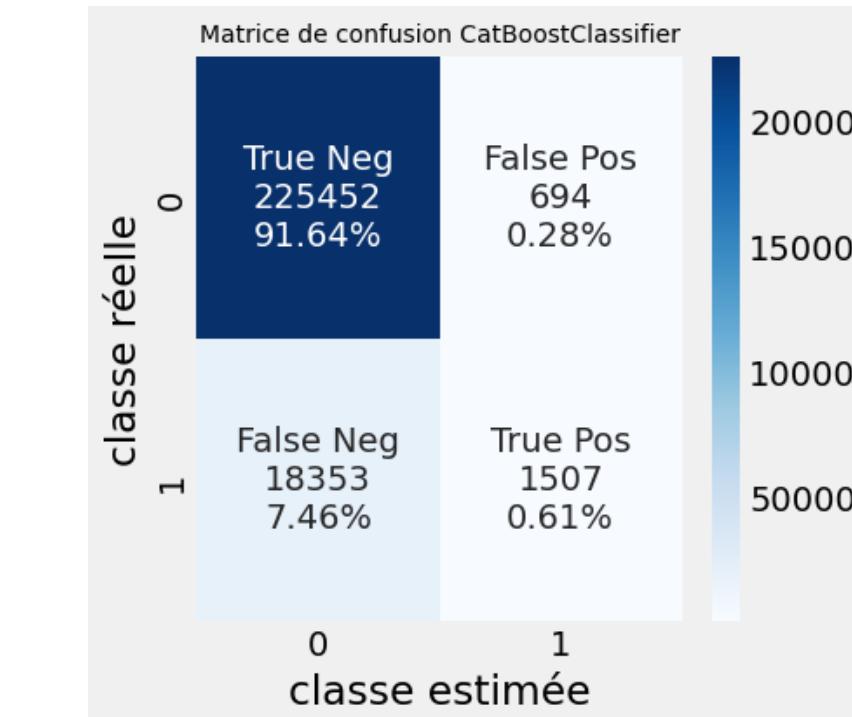


Développement et simulation de modèles

```
----- LGBMClassifier -----  
Best params: LGBMClassifier {'LGBMC__colsample_bytree': 0.97, 'LGBMC__reg_lambda': 2, 'LGBMC__subsample': 0.5}  
Score_metier 0.7821638496622033  
score_AUC_train 0.8033877805984564
```

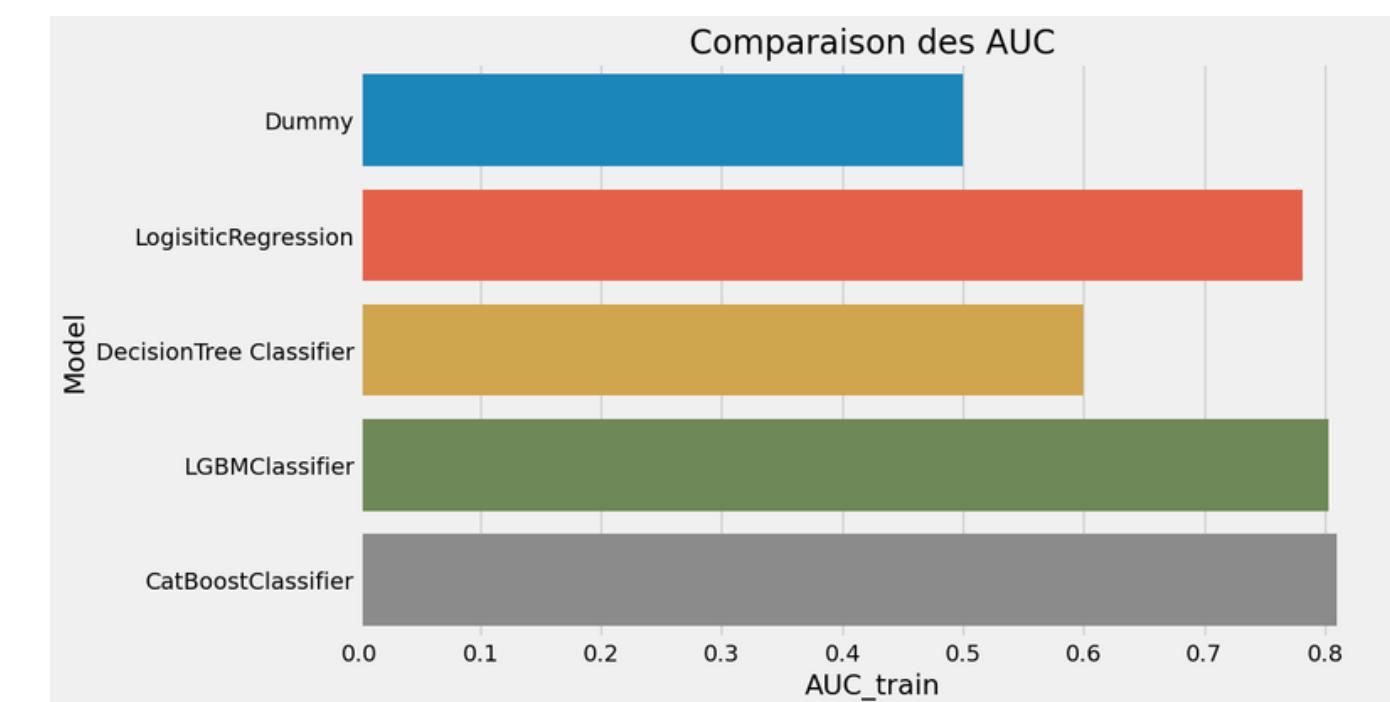
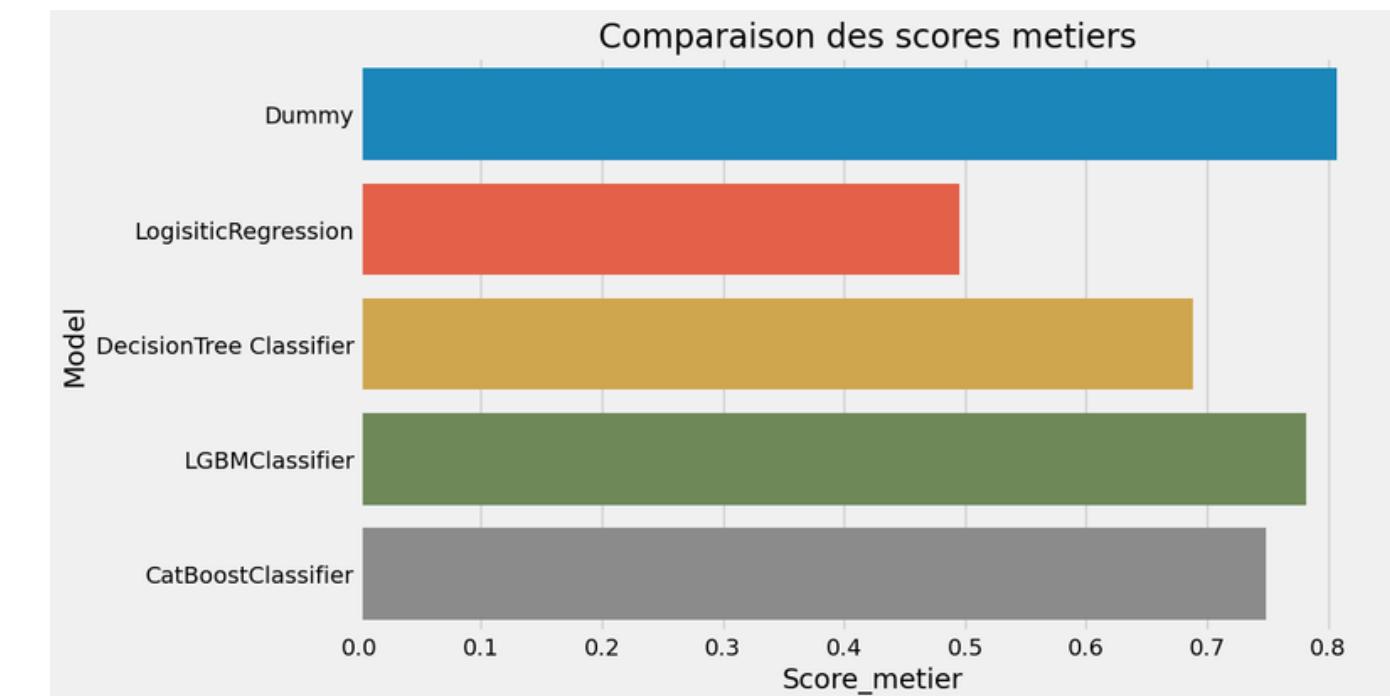
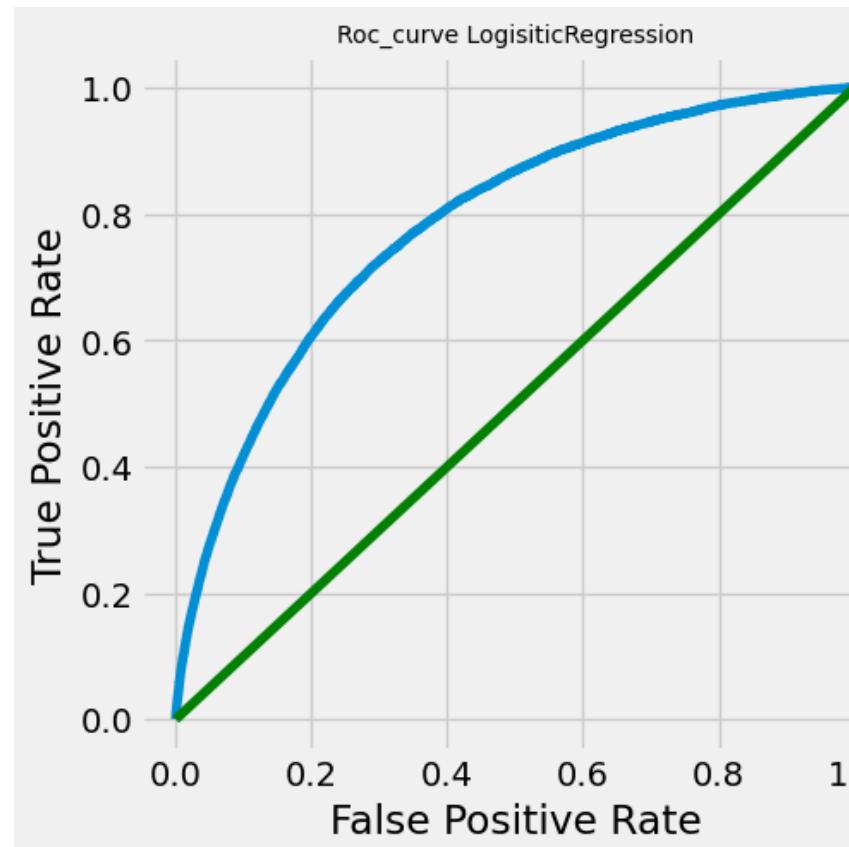
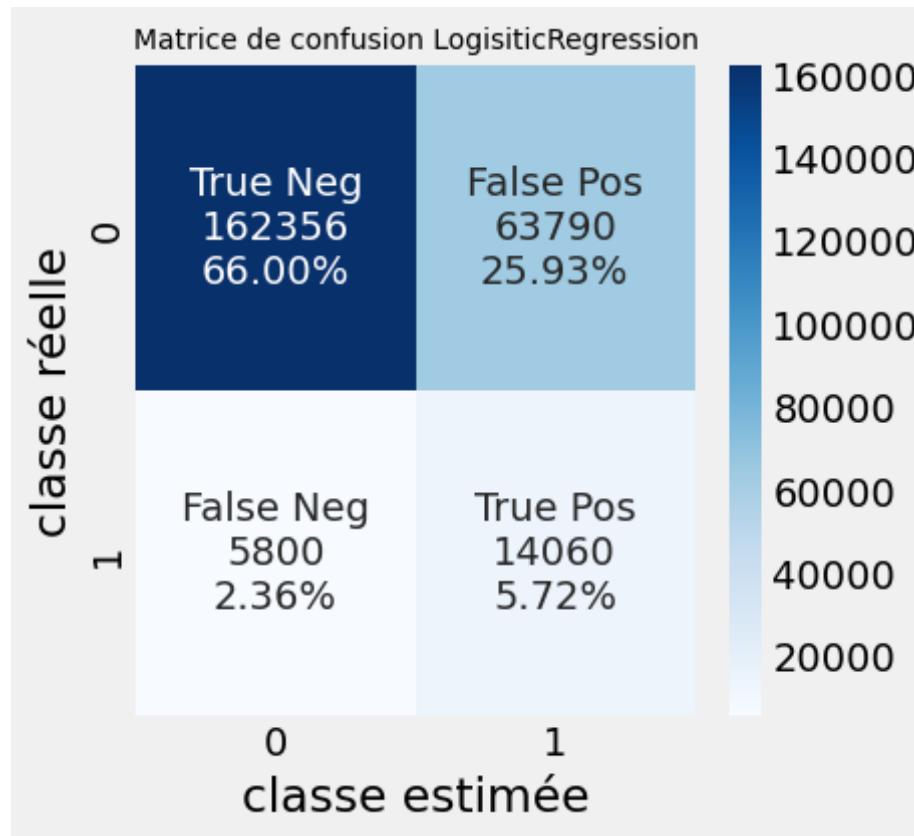


```
----- CatBoostClassifier -----  
Best params: CatBoostClassifier {'CatBoost__max_depth': 3, 'CatBoost__n_estimators': 300}  
Score_metier 0.7488597839077096  
score_AUC_train 0.810142354141741
```



Développement et simulation de modèles

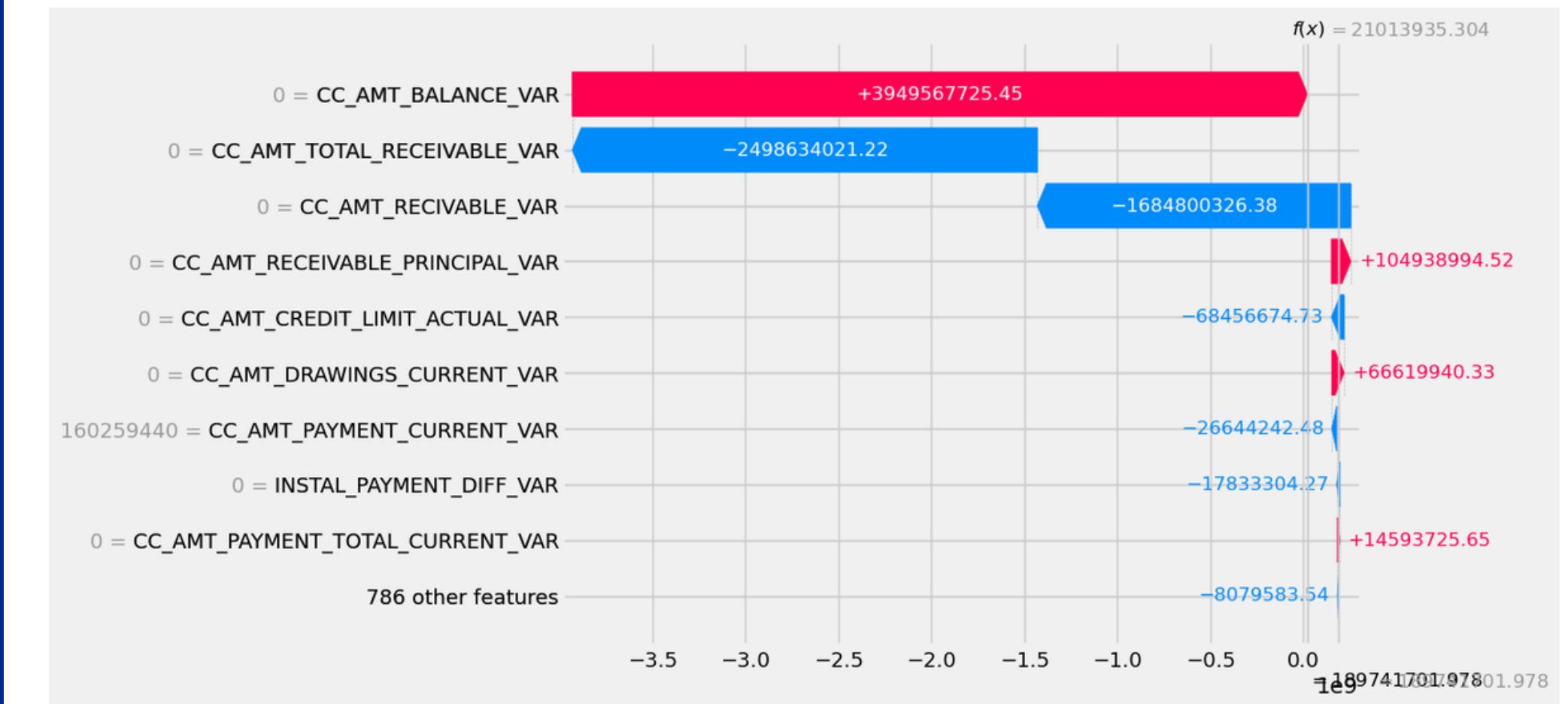
----- LogisticRegression -----
Best params: LogisticRegression {'Logreg_C': 100, 'Logreg_penalty': 'l2'}
Score_metier 0.4950692259538385
score_AUC_train 0.7815978887668652



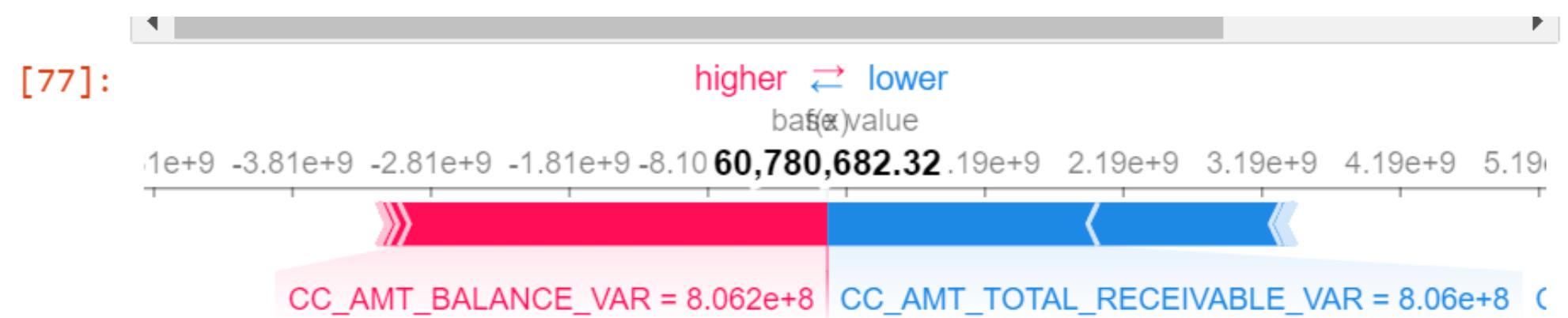
L'interprétabilité globale et locale du modèle

Pour connaître les principales features qui contribuent à l'élaboration du modèle, et pour le client, savoir qu'elle est l'influence de chaque feature dans le calcul de son propre score

L'interprétabilité globale



L'interprétabilité locale



L'analyse du Data Drift

Le Data Drift survient quand on a une grande différence entre les données d'entraînement et les données d'exécution.

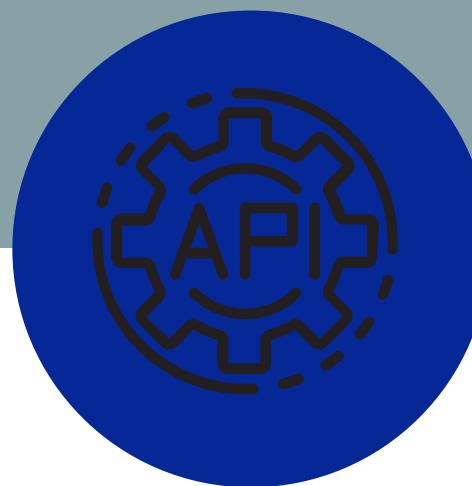
Nous avons utilisé la librairie Evidently pour détecter éventuellement du Data Drift sur les principales features, entre les datas d'entraînement et les datas de production.

```
[9]: report = Report(metrics=[  
    TargetDriftPreset()  
])  
  
report.run(reference_data=train_df, current_data=test_df,  
           #column_mapping=  
           )  
report.save_html('evidently_metrics_report.html')
```

Drift in column 'prediction'

Data drift not detected. Drift detection method: Jensen-Shannon distance. Drift score: 0.036

La présentation du dashboard



Réalisation d'une API

L'application réalisée avec **Fast API** renvoie le score d'un client. Elle réalise le predict à partir du modèle final sauvegardé

Réalisation du dashboard

L'application réalisée avec **Streamlit** appelle l'API via une url pour récupérer le score du client. Elle affiche également la feature importance locale et donne les informations concernant le client

Déploiement sur le cloud de l'API et du dashboard

Gestion des codes de l'API et dashboard avec **GIT**. Mise en production de l'API avec **Render** et du Dashboard sur **Streamlit**

Gestion du code: GIT

https://github.com/Arsekoffi/projet_7

Les codes de l'API et dashboard sont chacun gérés en version en local avec GIT

The screenshot shows a GitHub repository page for 'Arsekoffi / projet_7'. The repository is public, has 0 stars, and 0 forks. The main branch is 'main'. The commit history shows:

- A commit by Arsekoffi titled 'Update dsb.py' from 10 minutes ago, which has 70 reviews.
- A folder named 'app' committed 5 days ago.
- A folder named 'streamlit_code' committed 10 minutes ago.

The screenshot shows a local Git interface for the 'projet_7' repository. The current branch is 'main'. The commit history shows several 'Update dsb.py' commits from Arsekoffi. A detailed view of one commit is shown, comparing the file 'streamlit_code\dsb.py' against its previous state. The diff highlights changes in lines 23 and 24:

```
streamlit_code\dsb.py @@ -20,6 +20,7 @@ shap.initjs()
20 20
21 21 #path= "C:/Users/mr_ar/Downloads/Projet+Mise+en+prod++home-cr
22 22 edit-default-risk/"
23 +application_train=application_train.drop(columns=[ 'Unnamed:
24 24 0','index'],axis=1)
25 25 application_test = pd.read_csv("test_api.csv")
26 26
```

L'API

<https://api-1nb9.onrender.com/docs>

L'application réalisée avec Fast API renvoie le score d'un client. Elle réalise le predict à partir du modèle final sauvegardé

→ C api-1nb9.onrender.com/docs#/de... Mettre à jo

FastAPI 0.1.0 OAS3

/openapi.json

default

GET / Root

POST /predictions Predictions

Parameters

No parameters

Request body required

application/json

```
{ "ID": 100001 }
```

Responses

Curl

```
curl -X 'POST' \
  'https://api-1nb9.onrender.com/predictions' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
    "ID": 100001
}'
```

Request URL

https://api-1nb9.onrender.com/predictions

Server response

Code Details

200 Response body

```
{ "ID": 100001, "prediction": 0, "score": 0.341 }
```

Download Response headers

Le dashboard

<https://arsekoffi-projet-7-streamlit-codedsb-e95zbn.streamlit.app/>

L'application réalisée avec Streamlit appelle l'API via une url pour récupérer le score du client. Elle affiche également la feature importance locale et donne les informations concernant le client

Probabilité de remboursement de crédit

Entrez identifiant client

100001

Prédiction de la probabilité de remboursement du crédit

Reponse from API =
{"ID":100001,"prediction":0,"score":0.341}

Probabilité : 0.341 %

[Crédit accepté](#)

Afficher les données du client qui ont le plus influencé le calcul de son score ?
 Feature importance globale
 Afficher les informations relatives aux clients ?
 Comparer aux autres clients

Le dashboard

<https://arsekoffi-projet-7-streamlit-codedsb-e95zbn.streamlit.app/>

L'application réalisée avec Streamlit appelle l'API via une url pour récupérer le score du client. Elle affiche également la feature importance locale et donne les informations concernant le client

Informations relatives au client

Choisir les informations à afficher

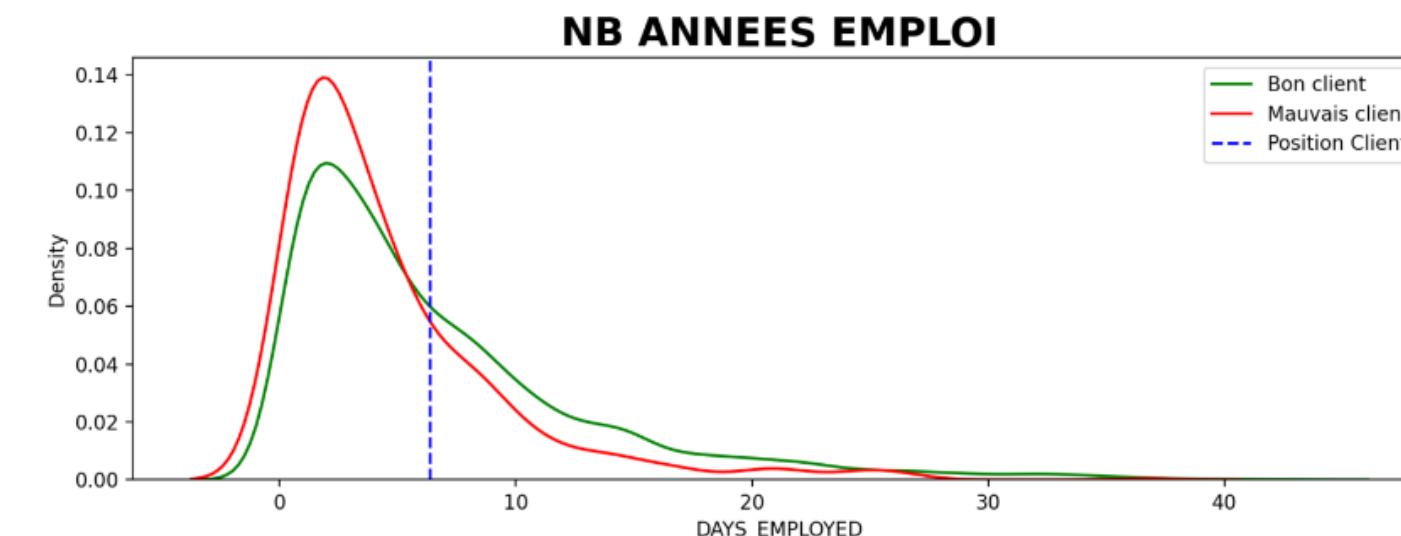
GENRE X AGE X STATUT FAMILIAL X NB ENFANTS X
REVENUS X MONTANT CREDIT X

	0
GENRE	Homme
AGE	52
STATUT FAMILIAL	Married
NB ENFANTS	0
REVENUS	135000.0
MONTANT CREDIT	568800.0

Comparaison aux autres clients

Sélectionner une variable

NB ANNEES EMPLOI



Conclusion



Réalisation et mise en production d'un modèle de prédiction simple et facile d'utilisation.



Le feature engineering:
selectionner les variables les plus pertinentes pour faciliter l'analyse



L'amélioration des paramètres de la construction de la "fonction coût métier" en les adaptant aux exigences du métier, nous pouvons amélioré nos résultats

Les améliorations possibles

Merci !

