Implémentez un modèle de Scoring

La société financière Prêt à dépenser, propose des crédits à la consommation.
L'entreprise souhaite :

- Mettre en œuvre un outil de Scoring Crédit pour calculer la probabilité qu'un client rembourse son crédit
- 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

Objectif de la mission

La mission consiste a :

- 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.

Analyse exploratoire EDA

```
# Training data
application_train = pd.read_csv(path + 'application_train.csv')
print('Training data shape: ', application_train.shape)
application_train.head()
```

```
Training data shape:  (307511, 122)
```

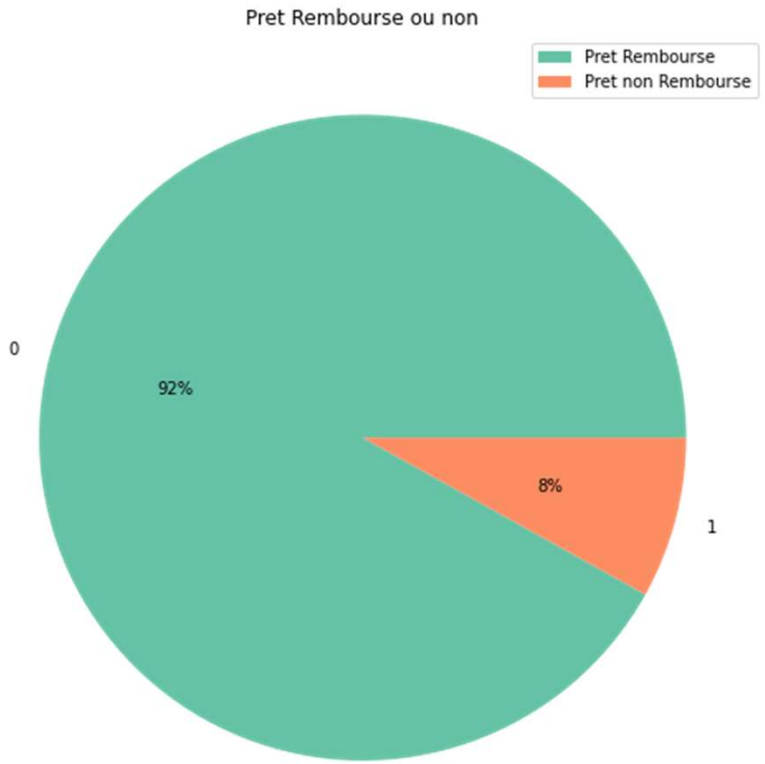
	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREI
0	100002	1	Cash loans	M	N	Y	0	202500.0	40659
1	100003	0	Cash loans	F	N	N	0	270000.0	129350
2	100004	0	Revolving loans	M	Y	Y	0	67500.0	13500
3	100006	0	Cash loans	F	N	Y	0	135000.0	31268
4	100007	0	Cash loans	M	N	Y	0	121500.0	51300

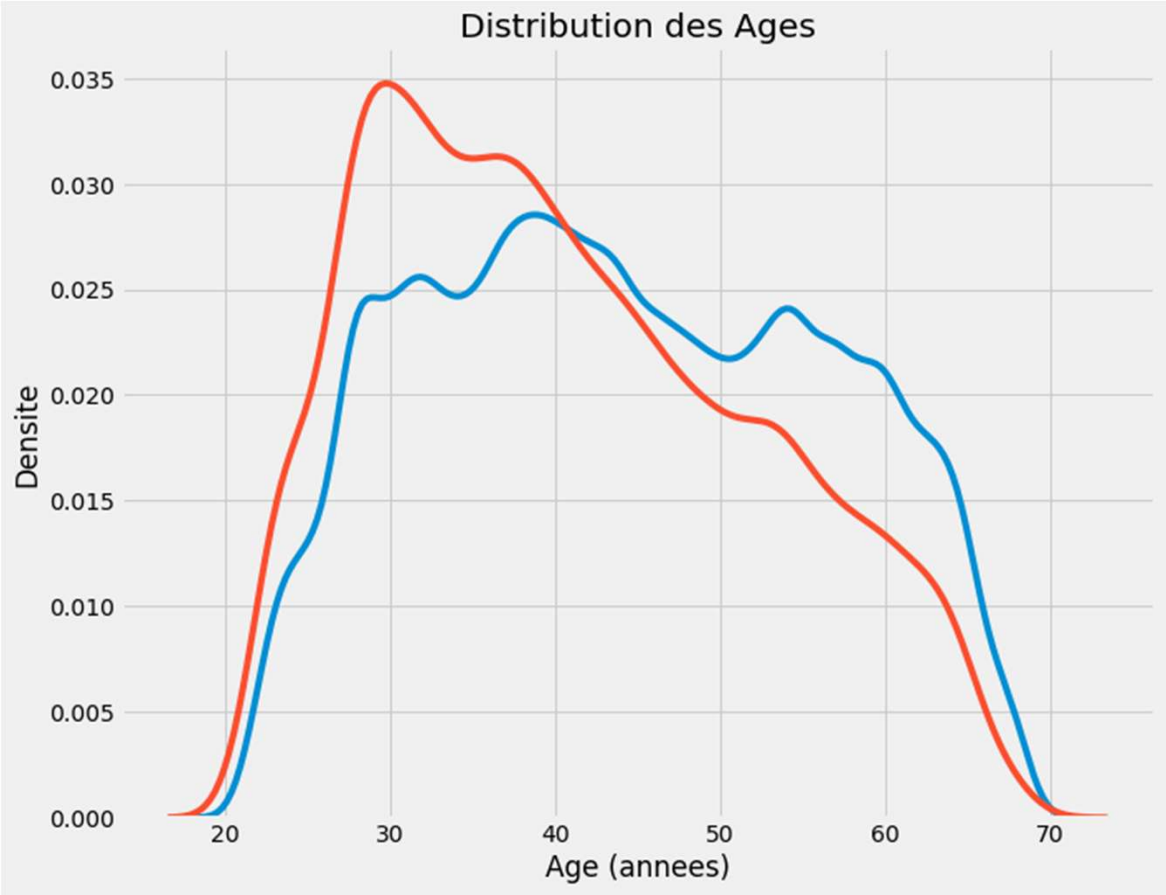
5 rows × 122 columns

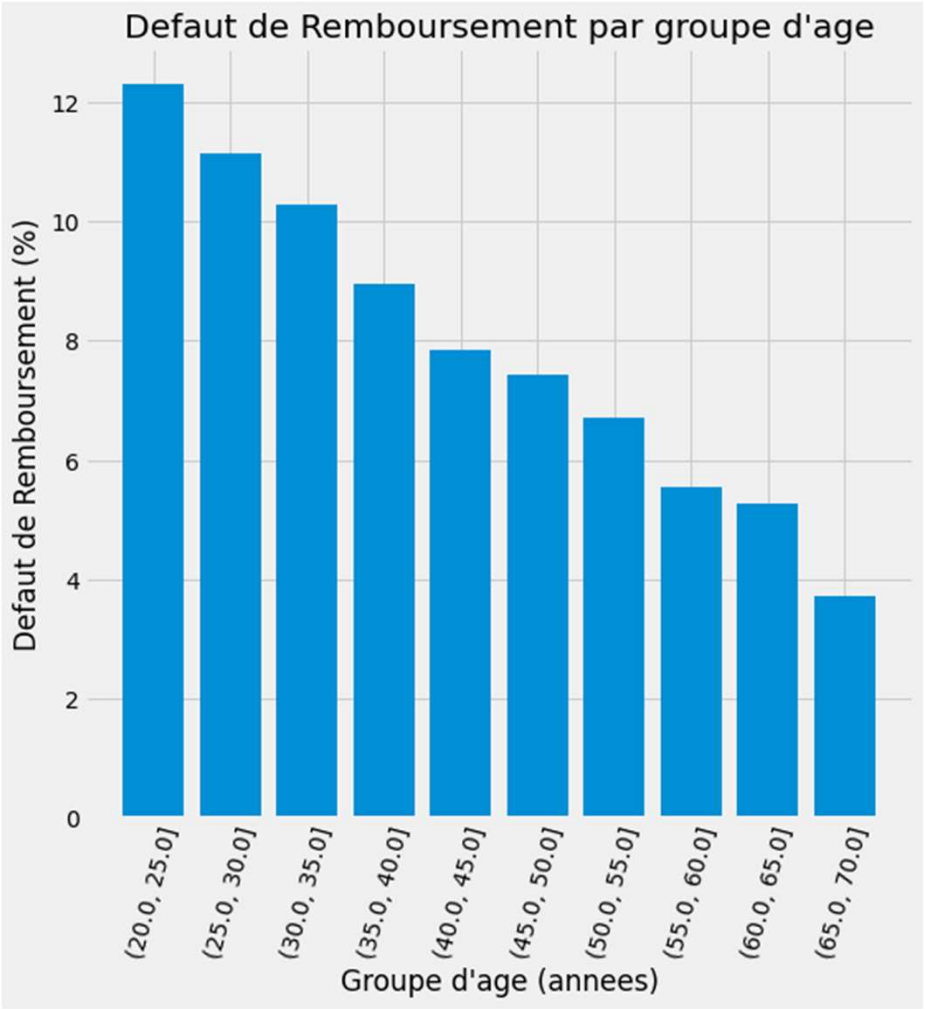
```
Test data shape:  (48744, 121)
```

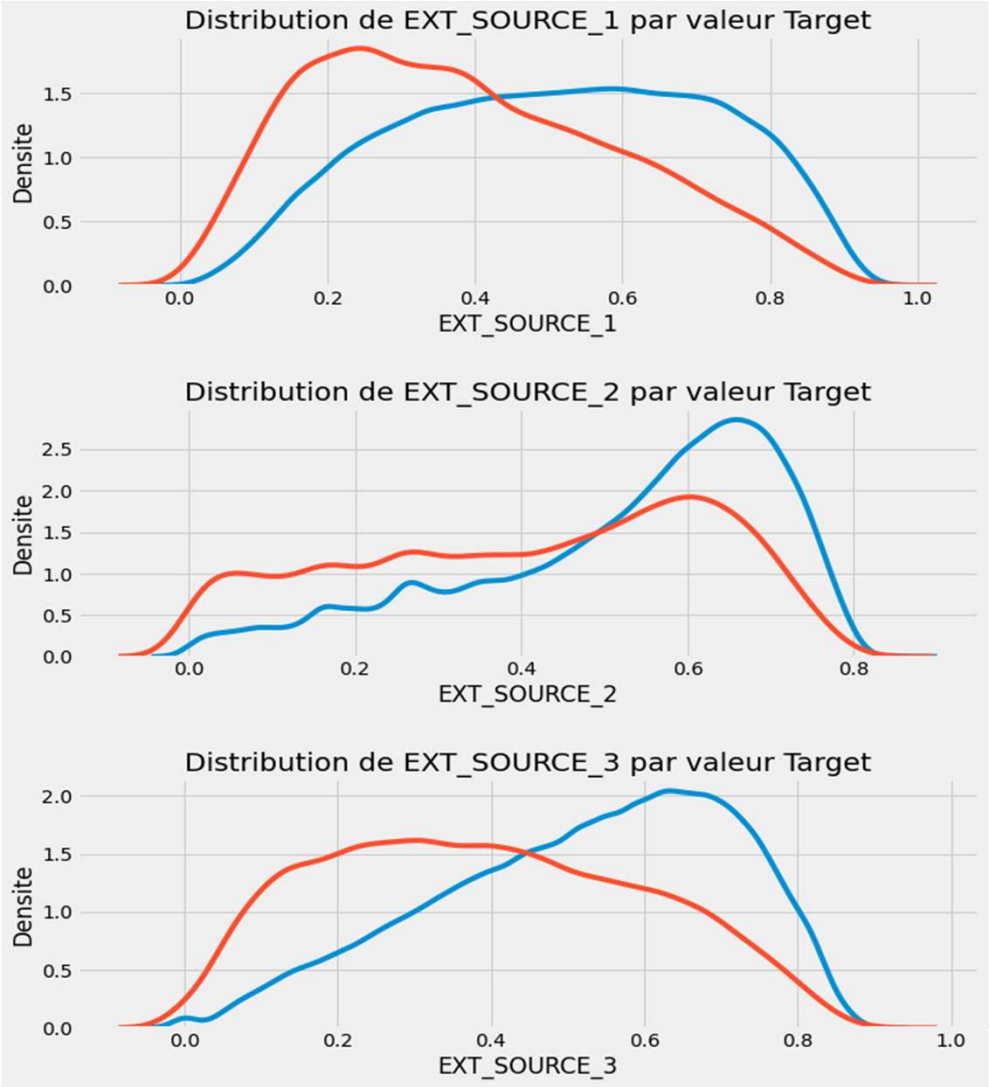
	SK_ID_CURR	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_
0	100001	Cash loans	F	N	Y	0	135000.0	568800.0	
1	100005	Cash loans	M	N	Y	0	99000.0	222768.0	
2	100013	Cash loans	M	Y	Y	0	202500.0	663264.0	
3	100028	Cash loans	F	N	Y	2	315000.0	1575000.0	
4	100038	Cash loans	M	Y	N	1	180000.0	625500.0	

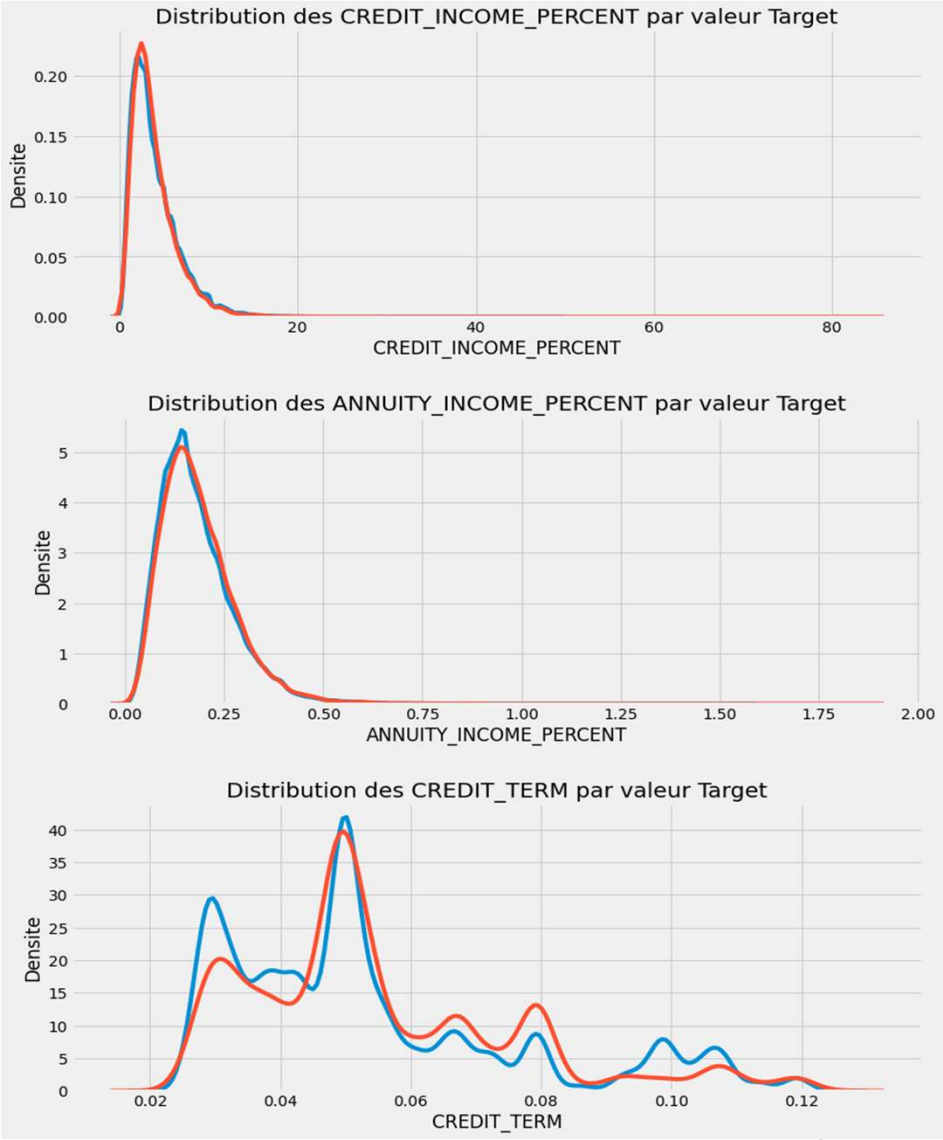
5 rows × 121 columns


Pret Rembourse ou non
Pret Rembourse
Pret non Rembourse
0
92%
8%
1


Distribution des Ages
Densite
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
20
30
40
50
60
70
Age (annees)


Defaut de Remboursement par groupe d'age
Defaut de Remboursement (%)
0
2
4
6
8
10
12
(20.0, 25.0]
(25.0, 30.0]
(30.0, 35.0]
(35.0, 40.0]
(40.0, 45.0]
(45.0, 50.0]
(50.0, 55.0]
(55.0, 60.0]
(60.0, 65.0]
(65.0, 70.0]
Groupe d'age (annees)


Distribution de EXT_SOURCE_1 par valeur Target
Densite
0.0
0.5
1.0
1.5
0.0
0.2
0.4
0.6
0.8
1.0
EXT_SOURCE_1
Distribution de EXT_SOURCE_2 par valeur Target
Densite
0.0
0.5
1.0
1.5
2.0
2.5
0.0
0.2
0.4
0.6
0.8
EXT_SOURCE_2
Distribution de EXT_SOURCE_3 par valeur Target
Densite
0.0
0.5
1.0
1.5
2.0
0.0
0.2
0.4
0.6
0.8
1.0
EXT_SOURCE_3


Distribution des CREDIT_INCOME_PERCENT par valeur Target
Densite
0.20
0.15
0.10
0.05
0.00
0
20
40
60
80
CREDIT_INCOME_PERCENT
Distribution des ANNUITY_INCOME_PERCENT par valeur Target
Densite
5
4
3
2
1
0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
ANNUITY_INCOME_PERCENT
Distribution des CREDIT_TERM par valeur Target
Densite
40
35
30
25
20
15
10
5
0
0.02
0.04
0.06
0.08
0.10
0.12
CREDIT_TERM

Conclusion EDA

- Nous sommes en presence d'un probleme de classification binaire avec un fort desequilibre de classes
- Ce genre de probleme est modelise principalement en utilisant 3 techniques :
  - Oversampling (SMOTE)
  - Undersimpling
  - Class weight

Modelisation

Setting up Environment in PyCaret - Preprocessing

```
from pycaret.classification import *
```

```
exp_credit = setup(data = application_train_domain, target = 'TARGET', session_id=123,
                   normalize = True,
                   transformation = True,
                   ignore_low_variance = True,
                   remove_outliers = True,
                   remove_multicollinearity = True,
                   multicollinearity_threshold = 0.95,
                   fix_imbalance = True #SMOTE
                  )
```

	Description	Value
0	session_id	123
1	Target	TARGET
2	Target Type	Binary
3	Label Encoded	None
4	Original Data	(307511, 127)
5	Missing Values	True
6	Numeric Features	67
7	Categorical Features	59
8	Ordinal Features	False
9	High Cardinality Features	False
10	High Cardinality Method	None
11	Transformed Train Set	(204494, 173)

```
top3 = compare_models(include = estimators)
```

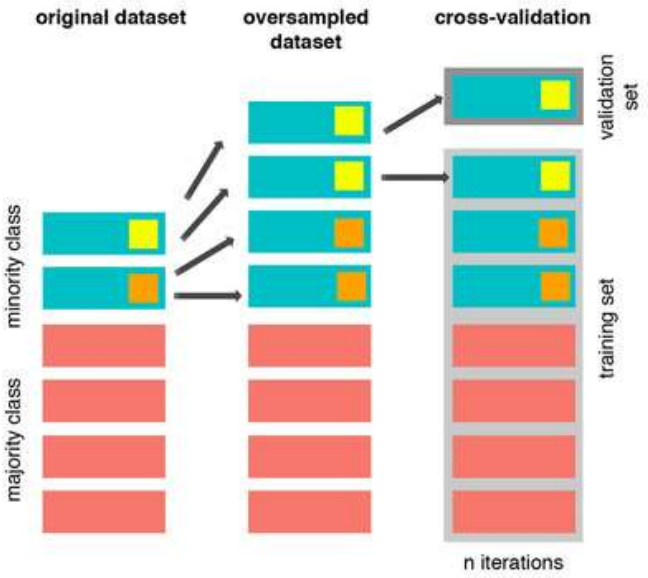
	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
lightgbm	Light Gradient Boosting Machine	0.9195	0.7298	0.0103	0.5158	0.0202	0.0170	0.0637	21.5340
rf	Random Forest Classifier	0.9194	0.6805	0.0006	0.3869	0.0012	0.0009	0.0119	52.6200
gbc	Gradient Boosting Classifier	0.9191	0.6831	0.0089	0.3946	0.0173	0.0138	0.0489	112.0040
ada	Ada Boost Classifier	0.9096	0.6549	0.0420	0.2095	0.0666	0.0417	0.0568	38.6870
dt	Decision Tree Classifier	0.8432	0.5260	0.1480	0.1190	0.1319	0.0469	0.0472	100.2020
lr	Logistic Regression	0.6667	0.7051	0.6286	0.1430	0.2330	0.1172	0.1700	47.0290
ridge	Ridge Classifier	0.6634	0.0000	0.6335	0.1425	0.2326	0.1164	0.1700	31.2730
nb	Naive Bayes	0.3594	0.5734	0.7503	0.0888	0.1587	0.0172	0.0441	18.7800

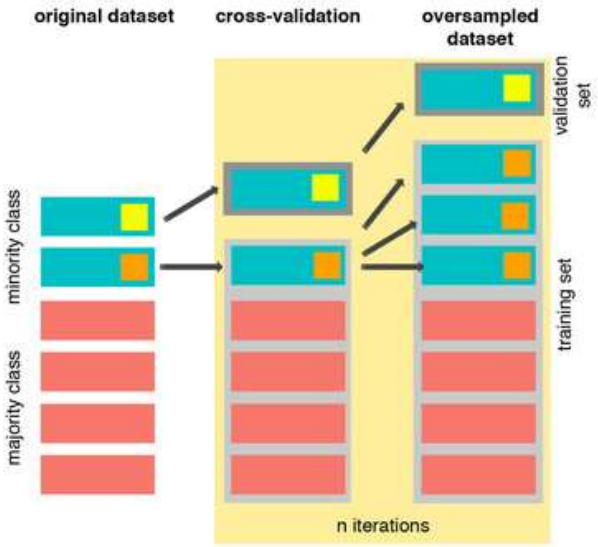
Commentaires sur les resultats :

Nous voyons que :

- les modeles ensemblistes rf, lightgbm, gbc et ada ont un Accuracy trop eleve, ce qui rappelle l'**accuracy paradox** de plus leurs Recall est pratiquement nul ce qui signifie que les faux negatifs sont tres eleves ce qui est contraire au resultat qu'on cherche.
- Par contre LogisticRegression, Ridge et Naive Bayes ont des recall eleve ce qui nous incite a les choisir pour la suite.
- Remarque : Ridge n'offre pas la possibilite de faire un threshold moving car il n'a pas de predict_proba. Les classes sont predites en 0 ou a directement.

Modelisation - RandomSearchCV – pipeline avec SMOTE



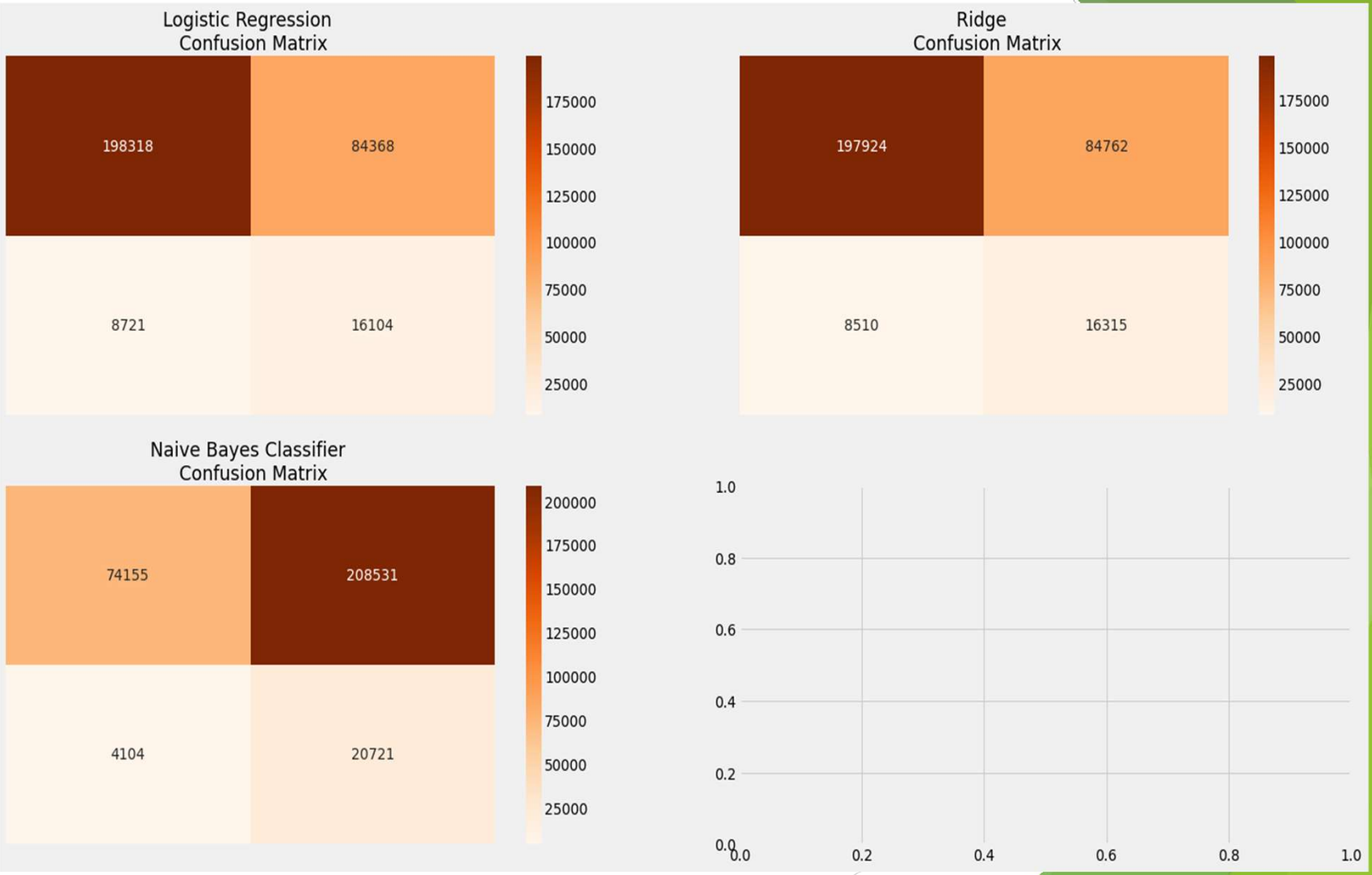


```
def test_SMOTE_model(model, param_distr, X_train , y_train, n_iter, n_splits, Beta):
    # List to append the score and then find the average

    rand_CV = RandomizedSearchCV(model, param_distr, n_iter=n_iter)

    StratKF = StratifiedKFold(n_splits=n_splits)

    for train, test in StratKF.split(X_train, y_train):
        # SMOTE realisee durant la Cross Validation non avant..
        pipeline = imbalanced_make_pipeline(SMOTE(sampling_strategy='minority'), rand_CV)
        model = pipeline.fit(X_train[train], y_train[train])
        best_estim = rand_CV.best_estimator_
        y_pred = best_estim.predict(X_train[test])
        # Scores
        accuracy_lst.append(pipeline.score(X_train[test], y_train[test]))
        auc_lst.append(roc_auc_score(y_train[test], y_pred))
        precision_lst.append(precision_score(y_train[test], y_pred))
        recall_lst.append(recall_score(y_train[test], y_pred))
        f1_lst.append(f1_score(y_train[test], y_pred))
        fbeta_lst.append(fbeta_score(y_train[test], y_pred,beta=Beta))

    avg_accuracy = np.mean(accuracy_lst)
    avg_auc = np.mean(auc_lst)
    avg_precision = np.mean(precision_lst)
    avg_recall = np.mean(recall_lst)
    avg_f1 = np.mean(f1_lst)
    avg_fbeta = np.mean(fbeta_lst)

    return avg_accuracy, avg_auc, avg_precision, avg_recall, avg_f1, avg_fbeta, best_estim
```

Fbeta-measure : Une metrique convenable pour les imbalanced class problem :¶

Fbeta-measure est une métrique à score unique configurable pour évaluer un modèle de classification binaire basé sur les prédictions faites pour la classe positive.
La mesure Fbeta est calculée en utilisant les mesures **Precision** et **Recall**.
•La **Precision** est une métrique qui calcule le pourcentage de prédictions correctes pour la classe positive.
•Le **Recall** calcule le pourcentage de prédictions correctes pour la classe positive sur toutes les prédictions positives qui pourraient être faites.
La maximisation de la **Precision** minimisera les erreurs de faux positifs, tandis que la maximisation du **Recall** minimisera les erreurs de faux négatifs.
La **mesure F1** est calculée comme la moyenne harmonique de la Precision et du Recall, en donnant à chacun la même pondération. Il permet d'évaluer un modèle en tenant compte à la fois de la Precision et du Rappel à l'aide d'un score unique, ce qui est utile pour décrire les performances du modèle et pour comparer les modèles.
La **Fbeta-measure** est une généralisation de la **F-measure** qui ajoute un paramètre de configuration appelé **beta**.
•Une valeur bêta par défaut est 1, ce qui est identique à la mesure F.
•**Une valeur bêta plus petite, telle que 0.5, donne plus de poids à la Precision et moins au Recall**
•tandis qu'une valeur bêta plus grande, telle que 2, donne moins de poids à la Precision et plus de poids au Recall dans le calcul du score.
C'est une métrique utile à utiliser lorsque la Precision et le Recall sont importants mais qu'un peu plus d'attention est nécessaire sur l'un ou l'autre, par exemple lorsque les faux négatifs plus importants que les faux positifs, ou l'inverse.
Rappel :
•negative (class 0) : donc dans note cas les prets rembourses TARGET == 0
•positive (class 1) : les prets non rembourses TARGET == 1
Dans notre cas les faux negatifs doivent etre minimises mais pas trop afin de ne pas perdre de clients donc il faut minimiser les faux negatifs par maximisation du recall.
Nous choisirons une valeur beta egale a 3 pour commencer. Cette valeur donnera plus de poids au recall et moins a la Precision.

	Model	Accuracy_Score	AUC_Score	Precision_Score	Recall_Score	F1_Score	Fbeta_Score
0	RidgeClassifier	0.696047	0.676639	0.160496	0.653494	0.257693	0.499906
1	LogisticRegression	0.698359	0.674737	0.160473	0.646566	0.257117	0.496217
2	GaussianNB	0.300266	0.545956	0.089844	0.838953	0.162275	0.457231


Logistic Regression
Confusion Matrix
198318
84368
8721
16104
175000
150000
125000
100000
75000
50000
25000
Ridge
Confusion Matrix
197924
84762
8510
16315
175000
150000
125000
100000
75000
50000
25000
Naive Bayes Classifier
Confusion Matrix
74155
208531
4104
20721
200000
175000
150000
125000
100000
75000
50000
25000
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

Choix du meilleur modele

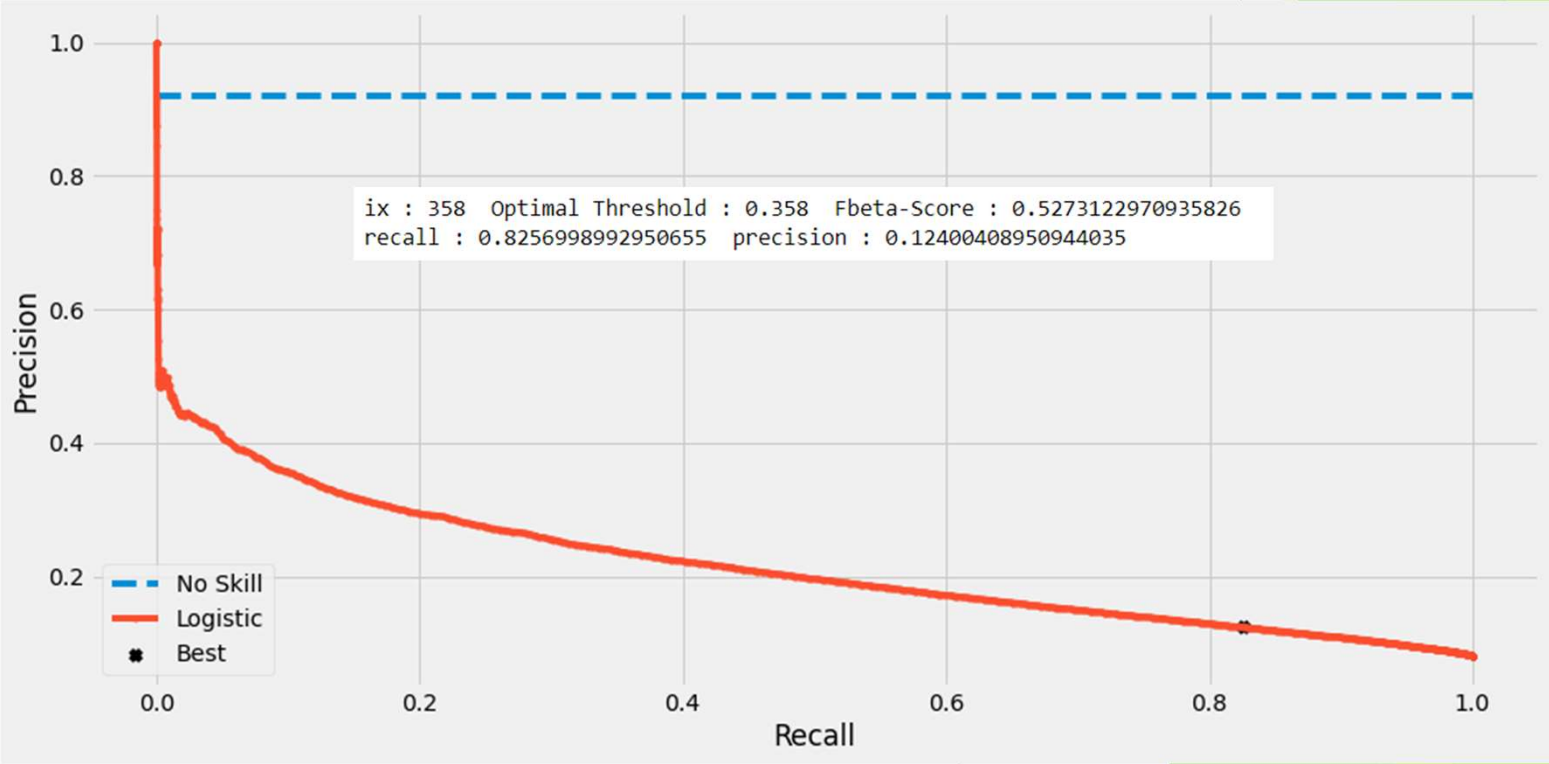
Les faux negatifs sont de :

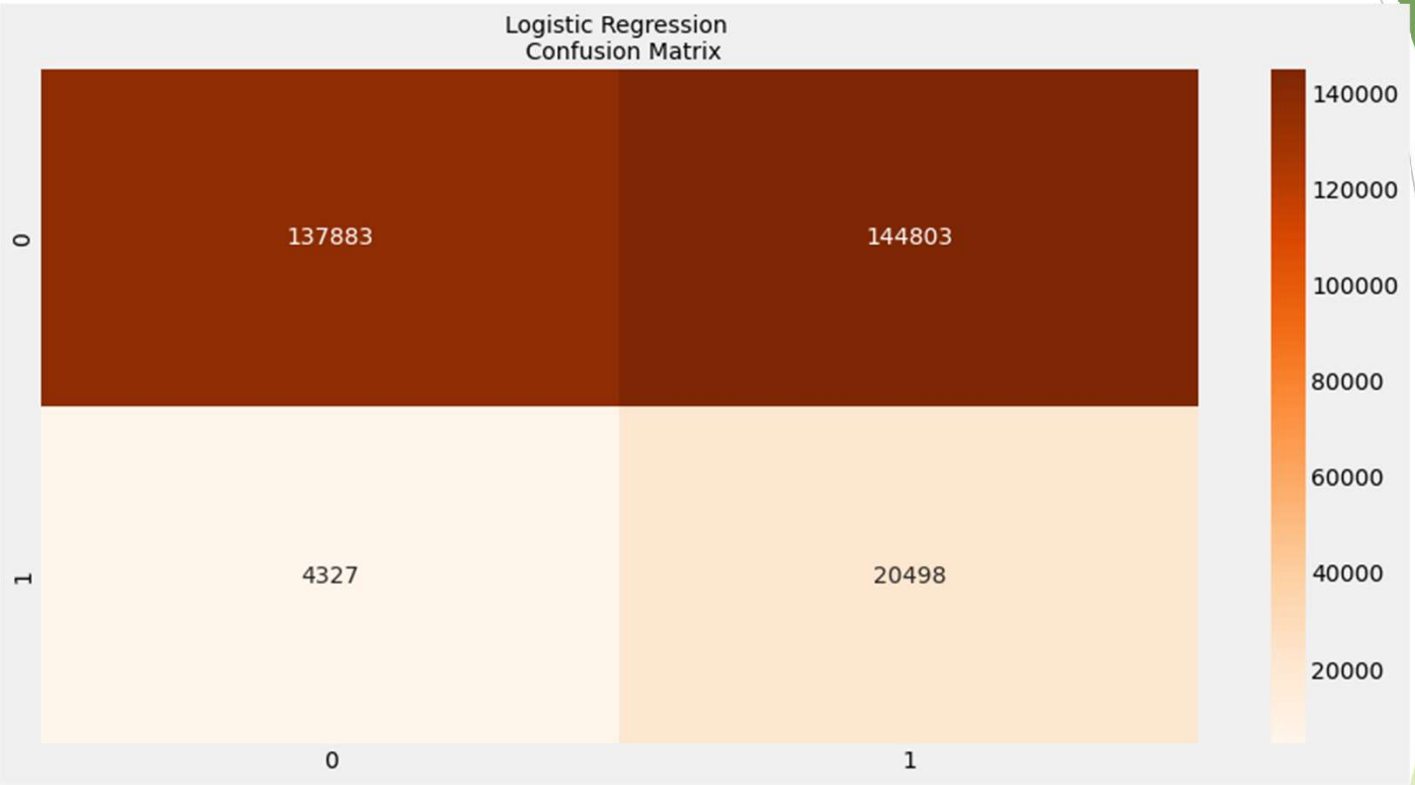
- 8 721 pour lr
- 8 510 pour ridge
- 4 104 pour Naive Bayes

Malgre ca l'etude suivante va montrer que le modele lr est mieux adapte pour notre cas, car ce qui compte aussi c'est de ne pas avoir trop de faux positifs avec une precision trop faible pour le modele Naive Bayes.

Nous preferons le modele lr sur le ridge car ce dernier n'offre pas la possibilite de faire un threshold moving, ces classes etant calcules directement (pas de predict_proba pour ce modele)

Threshold-Moving for Imbalanced Classification





Les faux negatifs sont passes de 8 721 a 4 327 avec threshold optimal

Interpretation : SHAP
(SHapley Additive exPlanations)

What Does the KernelExplainer Do?

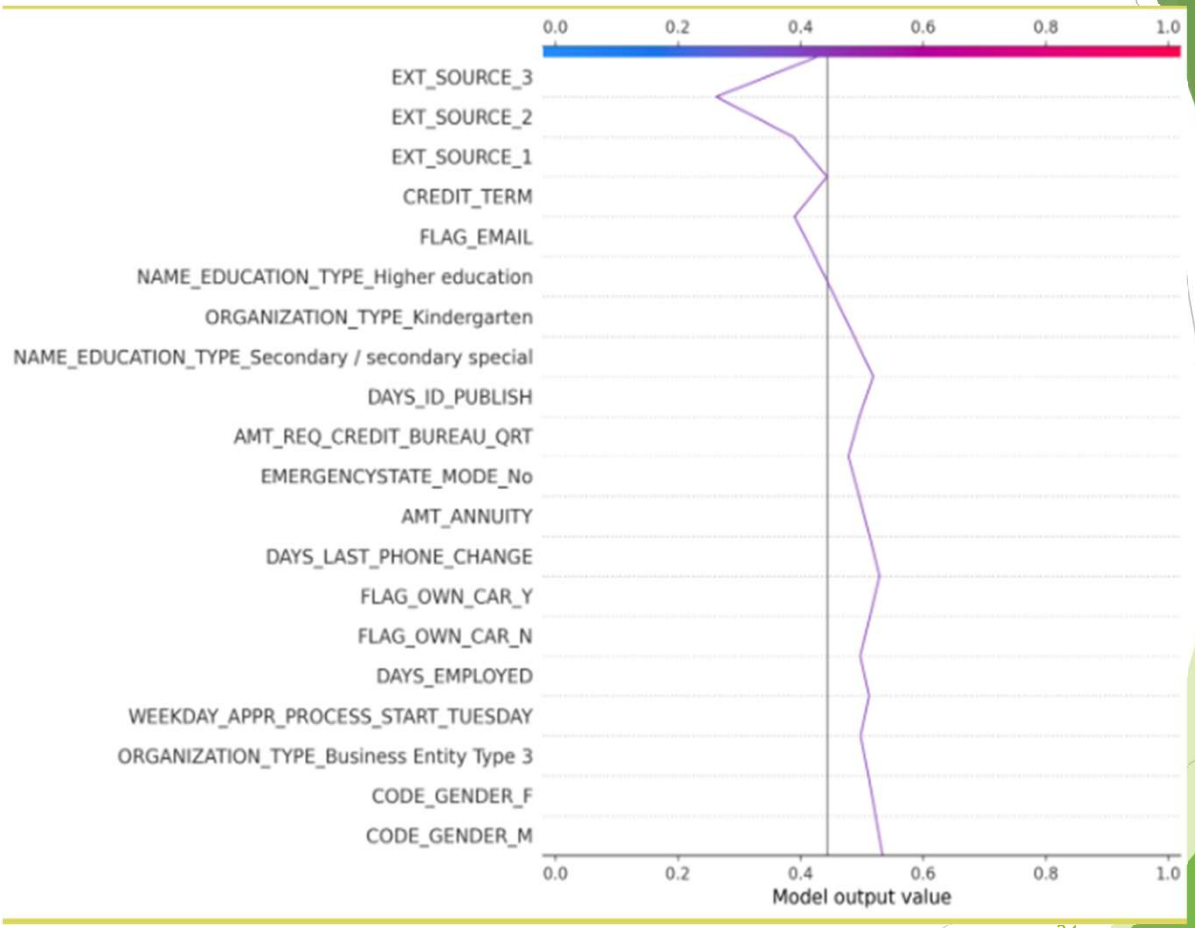
The KernelExplainer builds a weighted linear regression by using your data, your predictions, and whatever function that predicts the predicted values. It computes the variable importance values based on the Shapley values from game theory, and the coefficients from a local linear regression.

Arguments of KernelExplainer() function:

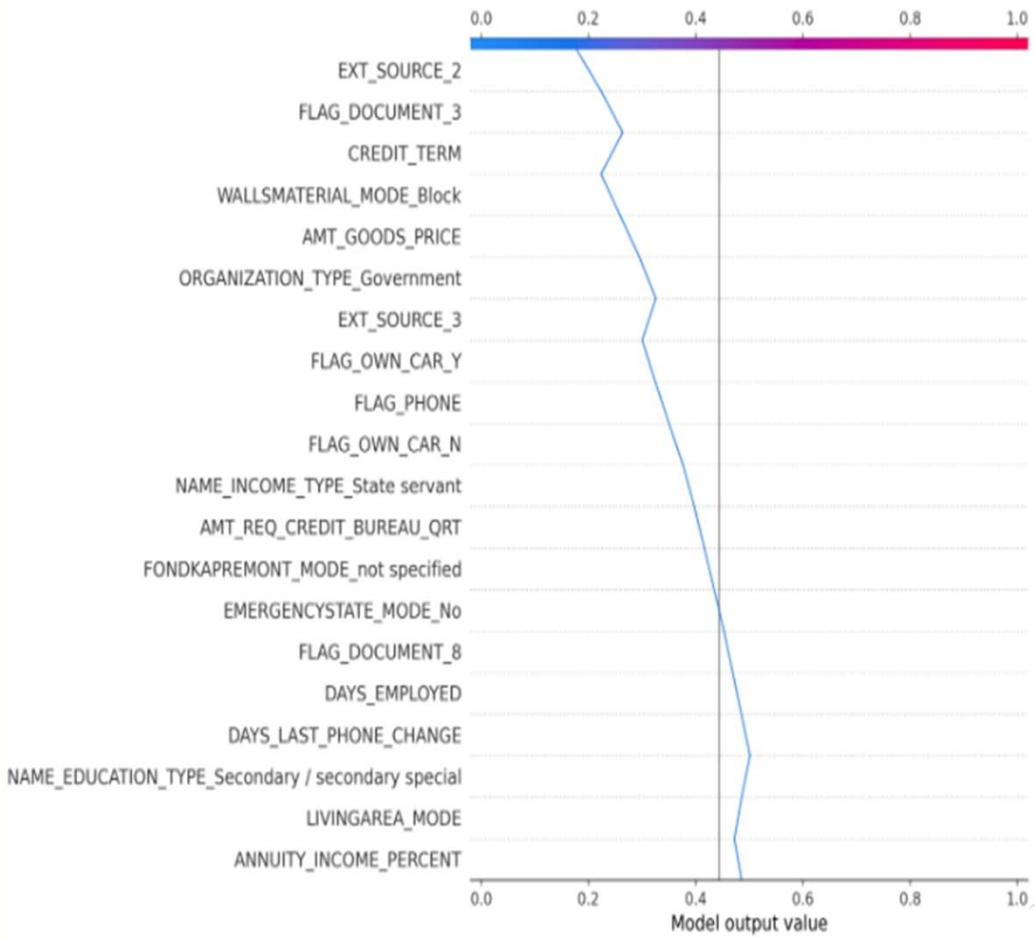
- model: The model to be explained. The output of the model can be a vector of size n_samples or a matrix of size [n_samples x n_output] (for a classification model).
- data: Background dataset to generate the perturbed dataset required for training surrogate models. We simulate "missing" ('0's in z_i) by replacing the feature with the values it takes in the background dataset. So if the background dataset is a simple sample of all zeros, then we would approximate a feature being missing by setting it to zero. For small problems this background dataset can be the whole training set, but for larger problems consider using a single reference value or using the kmeans function to summarize the dataset.
- link: A function to connect feature contribution values to the model output. For a classification model, we generally explain the logit of the predicted probability as a sum of feature contributions. Hence, if the output of the "model" (the first argument) is a probability, we set link = "logit" to get the feature contributions in logit form.

Interpretation : SHAP
(SHapley Additive exPlanations)

Decision Plot



Decision Plot



Dashboard

Spécifications du dashboard

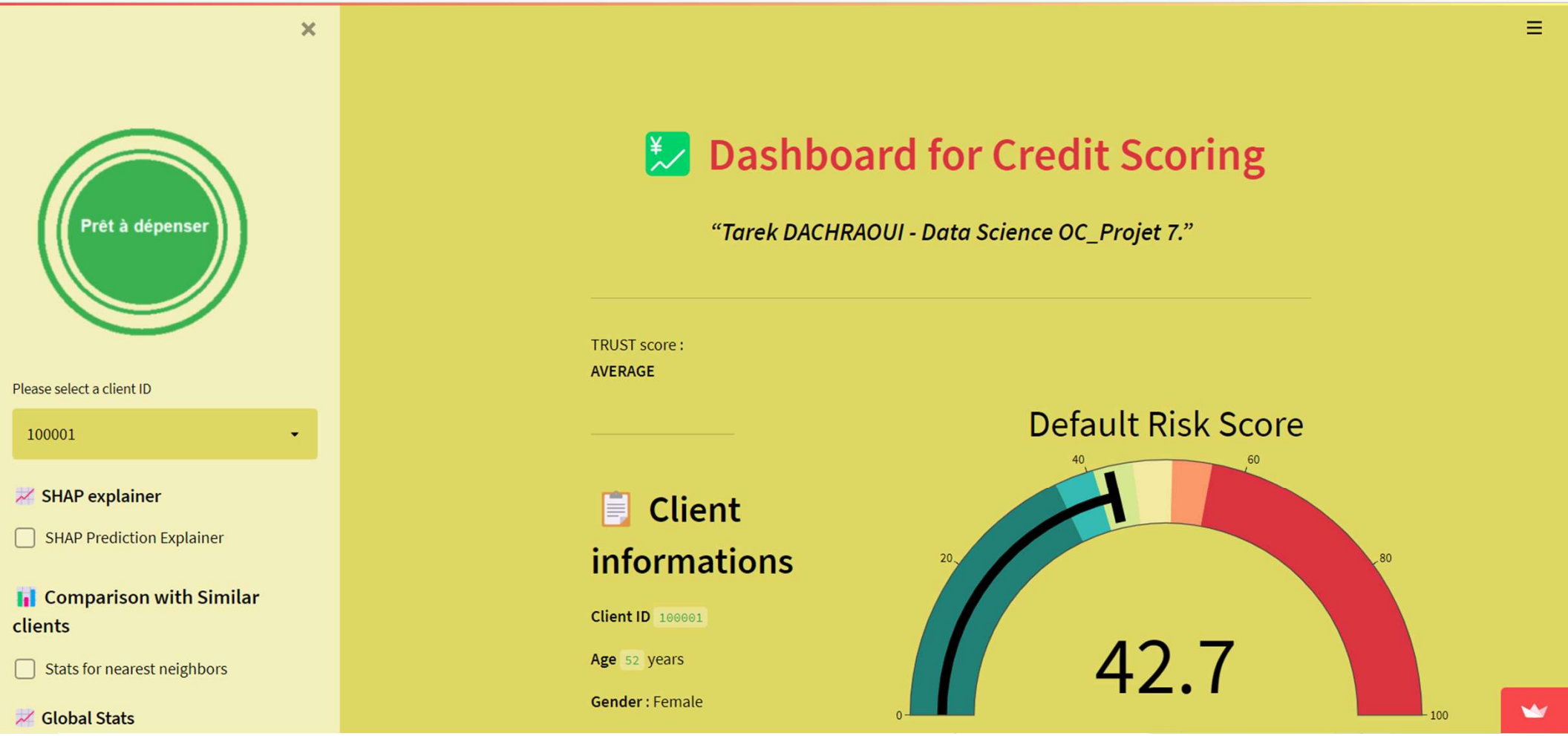
Le dashboard interactif devra contenir au minimum les fonctionnalités suivantes :

- Permettre de visualiser le score et l'interprétation de ce score pour chaque client de façon intelligible pour une personne non experte en data science.
- Permettre de visualiser des informations descriptives relatives à un client (via un système de filtre).
- Permettre de comparer les informations descriptives relatives à un client à l'ensemble des clients ou à un groupe de clients similaires.

Solution technique choisie

- Developpement du dashboard avec Streamlit.
- API : FastAPI

https://tarqdash-oc-projet7-dashboard-td-dashboard-streamlit-e8y7cl.streamlitapp.com/



Prêt à dépenser

Please select a client ID

100001

SHAP explainer

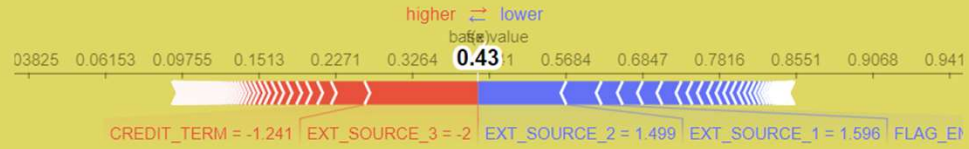
SHAP Prediction Explainer

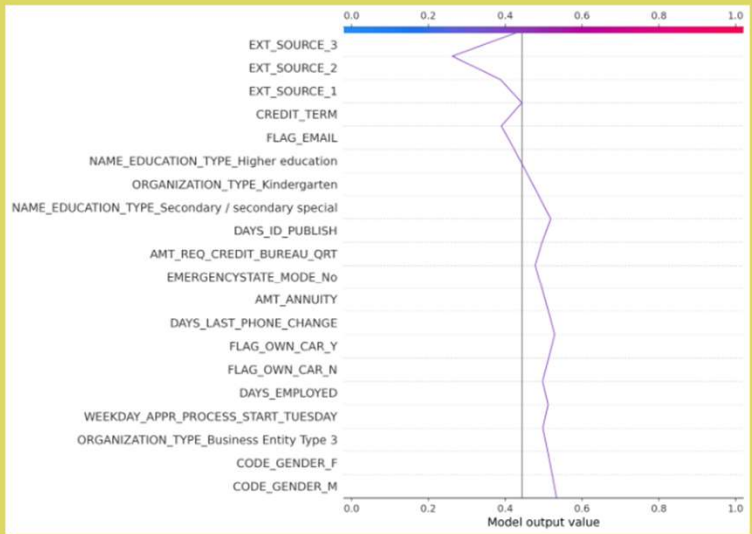
Comparison with Similar clients

Stats for nearest neighbors

Global Stats

SHAP Force & Decision Plots





Please select a feature description

AMT_CREDIT

Feature Description : **['Credit amount of the loan']**

Feature value :

568800.0

Comparison with Similar clients

- [ ] Stats for nearest neighbors

Age

YEARS EMPLOYED

- [ ] Show years employed infos / Hide

Work

Income

Contract Type

Family Status

Gender

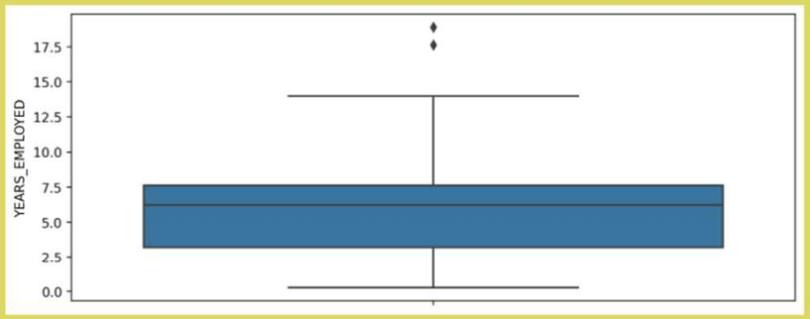
Education Type

Global Stats

- [ ] Global Stats

Income `135000` $

Contract type : Cash loans



Client YEARS EMPLOYED `6` years

Comparison with Similar clients

- [ ] Stats for nearest neighbors

Age

YEARS EMPLOYED

Work

Income

Contract Type

Family Status

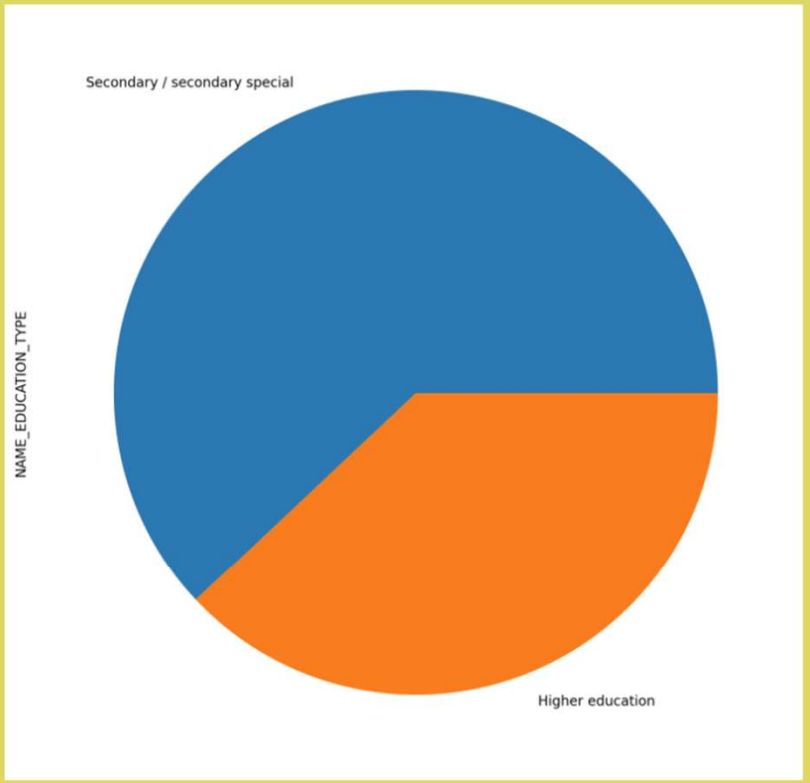
Gender

Education Type

- [ ] Show Education infos / Hide

Global Stats

- [ ] Global Stats



Education
Higher education

100001

SHAP explainer

- [ ] SHAP Prediction Explainer

Comparison with Similar clients

- [ ] Stats for nearest neighbors

Global Stats

- [ ] Global Stats

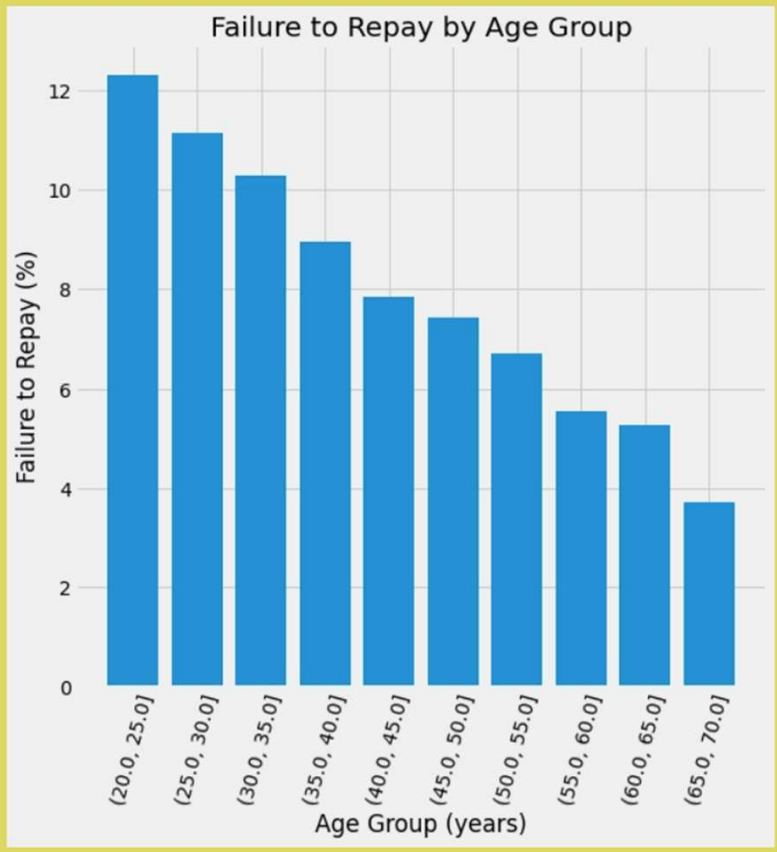
Failure to Repay by Age Group

- [ ] Show Failure to Repay by Age Group / Hide

Distrib. of EXT Sources

Distrib. of Ages

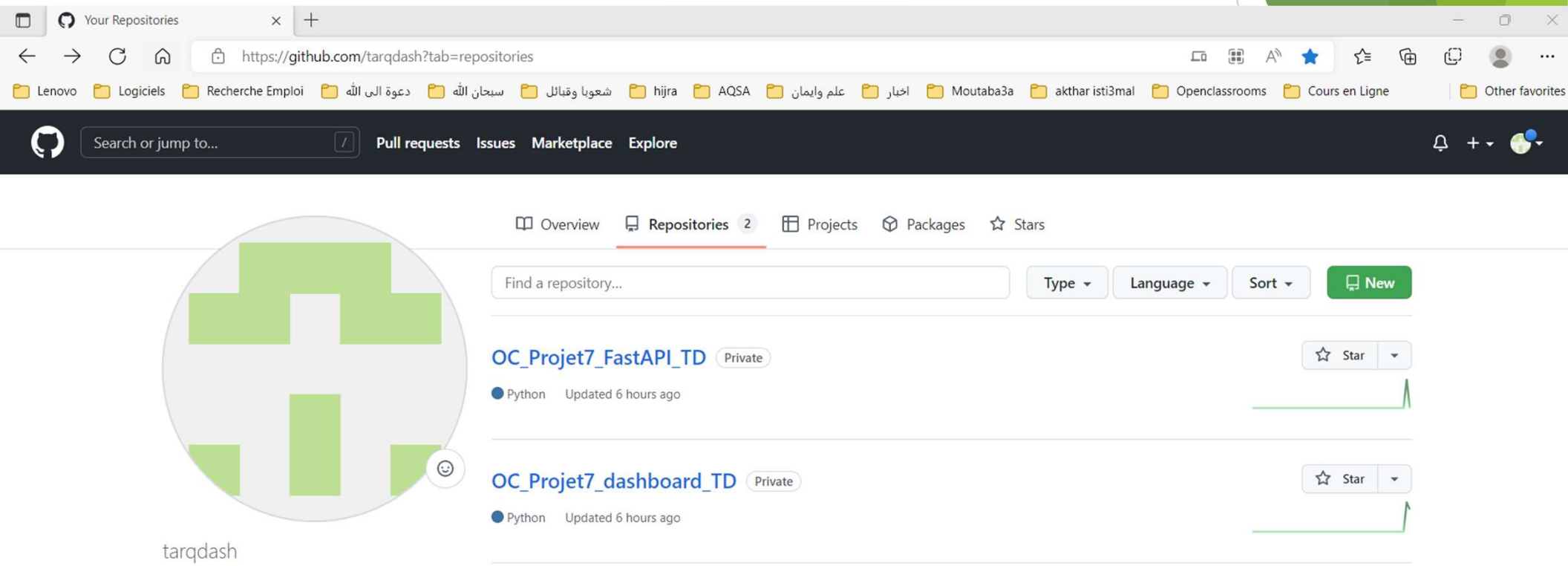
Distrib. of Credit Income



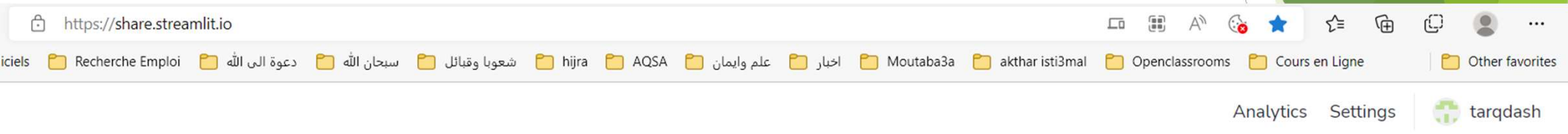
Deploiement

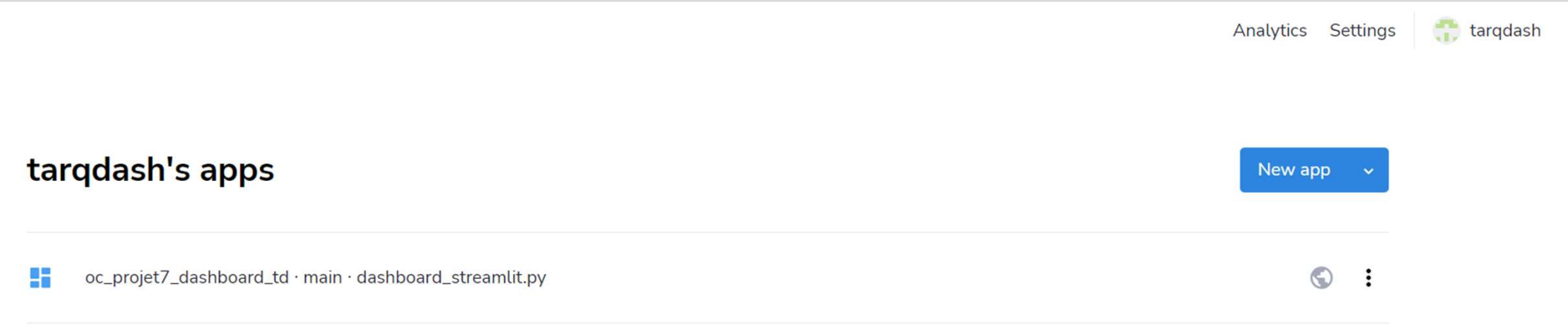
Depots sur github :

- https://github.com/tarqdash?tab=repositories

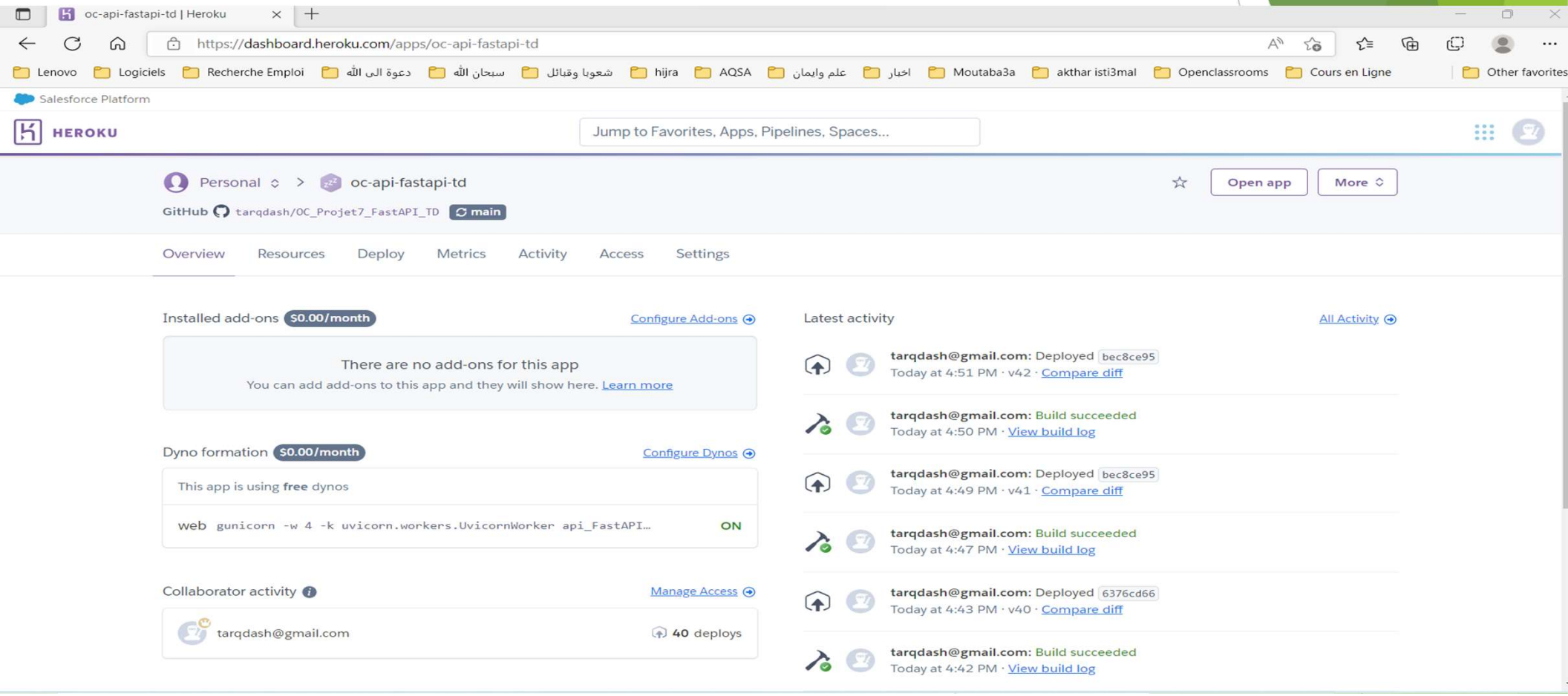


Dashboard sur https://share.streamlit.io/ :





API sur https://dashboard.heroku.com/apps/ :



Les limites et les améliorations possibles

Les limites :

- Temps de calcul important des modèles ensemblistes
- Pour des données plus importantes la technique de Oversampling SMOTE peut consommer beaucoup de ressources
- L'interprétabilité avec SHAP explainer est couteuse en temps de calcul. Pour la réduire nous avons utilise KernelExplainer car le LinearExplainer ne donne pas de résultats satisfaisants

Les améliorations possibles :

- Le modèle **Naive Bayes** peut être une bonne alternative a cause du Recall important qu'elle fournit (>80%) a condition de trouver un moyen algorithmique afin d'améliorer la faible Precision qu'elle donne.
- Comprendre et améliorer le temps de calcul des méthodes ensemblistes qui sont réputées meilleurs en terme de prédiction