Table des matières

Lien vers master branch GitHub, hébergement des fichiers

Contexte

'Prêt à dépenser' est une société financière proposant des crédits à la consommation à des personnes ayant peu ou pas du tout d'historique de prêt.
Les données mises à disposition contiennent diverses informations sur des clients et particulièrement la variable TARGET indiquant si le client honore ses échéances de prêt ou s'il est en difficulté de remboursement.
Il s'agit donc ici d'un problème de classification.

Problématique spécifique

La difficulté de modélisation réside dans le fait que le nombre de clients n'ayant pas de difficultés est très largement majoritaire, ce qui introduit un déséquilibre dans les effectifs des classes à prédire.
Les deux cas défavorables pour la banque sont les suivants :

Si elle accorde un prêt à un client qui va faire défaut.

Si elle est trop restrictive dans ses critères d'attribution et qu'elle refuse des prêts à des client qui auraient remboursé.

Il faut donc trouver une solution pour minimiser ces cas défavorables, afin de maximiser les profits.

1. ANALYSE DES DONNEES

Ligne directrice

L'analyse exploratoire des données a été menée de manière à trouver, ou créer, un nombre raisonnable de variables explicables permettant la modélisation de la problématique.
Pour ce faire des seuils planchers ont été définis comme suit :

- Variables quantitatives
  - Corrélation avec la cible supérieure à 4%
- Variables qualitatives
  - F-value supérieure à 200

  Nombre de valeurs manquantes inférieur à 20%

L'impératif d'explicabilité des choix sur lesquels sont fondés le score, et donc l'acceptation ou le refus du prêt, pourra ainsi respecter au mieux l'impératif d'explicabilité du score pour une personne non experte en data science, sans nuire à la qualité des prédictions, afin de répondre au mieux aux objectifs commerciaux.

Les données retenues pour la modélisation sont 39 variables et la cible, renseignées pour 166 167 clients, avec un ratio de 8,55% de défaut de paiement.

2. MODELISATION

2.1. Création d'une métrique d'évaluation des performances du modèle

Le problème posé induit la création d'une métrique permettant à la modélisation de répondre le plus fidèlement possible aux contraintes métier de l'entreprise.

En effet un modèle qui octroierait des prêts à tous les clients aurait une précision (en termes de classification) de 91,45% mais maximiserait les pertes financières.

Ce qu'il faut ici, c'est maximiser prédictions vraies par rapport à la réalité et minimiser les fausses prédictions (clients à jour étant prédits en défaut et vice versa).

La solution retenue est de créer un score, qui à partir des prédictions du modèle va le récompenser sur les prédictions favorables (clients bien scorés) et le pénaliser dans le cas contraire.

Pour ce faire, en utilisant la matrice de confusion, normalisée suivant les classes réelles, des pondérations retenues en fonction de chacun des cas de figure, on va pouvoir obtenir une métrique répondant aux objectif, les modèles seront évalués et optimisés suivant cette métrique.

2.2. Méthodologie des tests

Afin de pouvoir évaluer les performances de chaque modèle, le jeu de données principal est séparé en un jeu d'entrainement (train) et un jeu de validation (test) en veillant à respecter la même représentation de chaque classe sur les deux sets.

- Jeu d'entrainement :
  - 132930 clients.
  - Repayed / Default : 8.55218030%
- Jeu de validation :
  - 33233 clients.
  - Repayed / Default : 8.55108440%
- Les scores sur le jeu d'entrainement sont calculés par cross validation (5 StratifiedKfolds)
- Le modèle est ensuite entrainé sur train et évalué sur test

Les scores sur train ici présentés à titre de comparaison des performances de chaque modèle.
Les scores sur test permettent quant à eux de valider les capacités de généralisation du modèle.
Pour chacun des deux jeux des matrices de confusion sont calculées, elles affichent les effectifs et les pourcentages des effectifs en fonction des classes réelles.

2.3. Evaluations des différentes solutions mises en place

Pour chacun des modèles, les différentes solutions de rééquilibrage des effectifs des classes sont essayées, les résultats sont évalués sur les matrices de confusion dans cette partie, résultats complets en annexe

2.3.1. Modèles Naïfs

Prédire la classe majoritaire ou la classe minoritaire : scores 50%

2.3.2. Modèles étudiés

Régression Logistique (annexe 5.1)
Random Forest Classifier (annexe 5.2)
Light Gradient Boosting Classifier (Annexe 5.3)

Sans rééquilibrage des effectifs les modèles ne discriminent pas bien les classes : il faut donc trouver une solution afin de rétablir l'équilibre et ainsi d'augmenter les performances

2.3.3. Sous échantillonnage

RandomUnderSampling : On sous échantillonne la classe majoritaire, de manière aléatoire, afin de réduire le déséquilibre, ici elle a été réduite afin qu'il y ait les mêmes effectifs
Cette méthode peut être envisagée ici car la classe minoritaire contient 13091 échantillons appartenant à la classe minoritaire.
Cette solution a pour défaut de nous priver d'une grande quantité de données qui pourraient être utiles, de plus la sélection étant aléatoire il n'y a pas de moyen de quels enregistrements ont été écartés.
Il existe d'autres techniques de sous échantillonnage, avec des critères de choix plus élaborés, mais les temps de calculs sont plus longs et le manque de temps m'a obligé à retenir cette solution en première approche. (Annexe 5.4)

2.3.4. Sur échantillonnage

RandomOverSampling : On sur échantillonne la classe minoritaire en effectuant des tirages aléatoires avec remise jusqu'à obtenir le nombre d'échantillons souhaités
Cette technique peut être utile aux modèles sensibles aux distribution ayant un fort skewness (c'est ici le cas), et où la duplication d'échantillons appartenant à classe minoritaire peut influencer l'apprentissage du modèle.
Cela peut amener le modèle à surapprendre.
De même que précédemment, d'autres méthodes plus évoluées existent, mais leur coût en temps de calcul m'a amené à les écarter. (Annexe 5.5)

2.3.5. Pondération

On impose un coût supplémentaire aux erreurs de classification commises par le modèle sur les différentes classes. Ces pénalités ont pour but de biaiser le modèle afin qu'il accorde plus d'importance à la classe minoritaire au détriment de la classe majoritaire.
Deux solutions sont possibles :

Imposer les poids et chercher le ratio optimal

Utiliser l'argument class_weigth = 'balanced' et laisser le modèle imposer à chaque classe un poids inversement proportionnel à son effectif.

Pour les mêmes raisons de coût en temps de calcul, seule la deuxième option à été ici explorée. (Annexe 5.6)

2.3.6. Approche retenue

Afin de limiter les biais introduits par chaque méthode on va rechercher un compromis.

- Sur-échantillonnage modéré afin de minimiser le biais introduit
- Sous-échantillonnage modéré afin de ne pas omettre trop d'informations
- Le déséquilibre restant entre les effectifs de classes est compensé par pondération

Les résultats sont comparables à ceux obtenus précédemment, tout en minimisant les biais introduits par les méthodes de rééchantillonnage.
C'est un bon compromis, il faut maintenant passer à l'optimisation des différents paramètres, en particulier résoudre le problème de surapprentissage des forêts aléatoires. (Annexe 5.7)

2.3.7. Optimisation des paramètres

Pour chacun des modèles on va tester différents paramètres sur la préparation des données :

Sur-échantillonnage de 50% à 100% de la classe minoritaire.
Sous-échantillonnage de 0 à 50% de la classe majoritaire.
Différentes transformations des distributions des données :

- MinMaxSclaler
- StandardScaler :
- QuantileTransformer :
- RobustScaler :
- PowerTransformer :

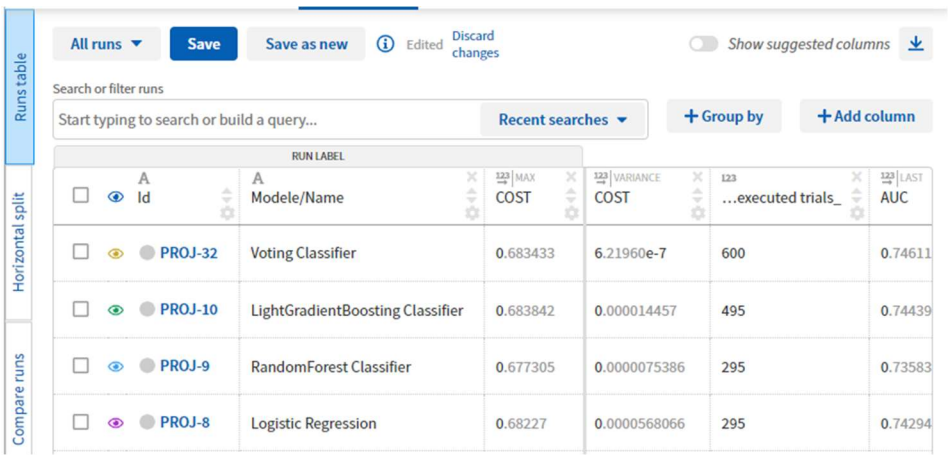
Intégration des différentes étapes dans un pipeline

Pipeline : [Sur échantillonnage, Sous échantillonnage, scaler, Modèle(class_weight='balanced')

Utilisation de la bibliothèque Optuna pour la recherche la meilleure valeur pour les paramètres testés.

- Algorithme d'optimisation TreeParser.

Tous les résultats intermédiaires et finaux sont suivis et enregistrés via Neptune ai (Annexe 5.8)



2.4. Mise en place d'un modèle ensembliste.

Afin de tirer le meilleur parti des étapes précédentes, le choix est fait de construire un méta-modèle.
Pour approfondir cette partie il faudrait aussi jouer sur les random seeds au moment de l'apprentissage des modèles, cette voie pourrait être explorée en gardant en ligne de mire le ratio temp de calcul/ gains en performances.

2.4.1. Concept du Soft voting

Afin de retenir le meilleur de chacun des modèles optimisés et de minimiser leurs biais respectifs, on construit un méta-modèle :
La décision sera prise sur la somme de la probabilité d'appartenance à la classe 1 donnée par chacun des modèles. (Annexes 5.9 et 5.10)

2.4.2. Résultats

	Model name	Cost max	Cost avr	Cost min	Fit time max	Fit time avr	Fit time min	Cost on test
0	lr	0.691204	0.681747	0.677224	12.333586	10.981507	9.408714	0.676302
1	rf	0.682291	0.674362	0.670829	24.107776	23.756953	23.225049	0.673935
2	lgbm	0.685214	0.677858	0.672996	21.914827	21.320651	20.082474	0.679327
3	voting	0.688778	0.681816	0.676641	116.180967	101.697165	74.831378	0.676640
4	voting opt	0.692513	0.683433	0.678822	58.099349	56.372617	52.532451	0.678270

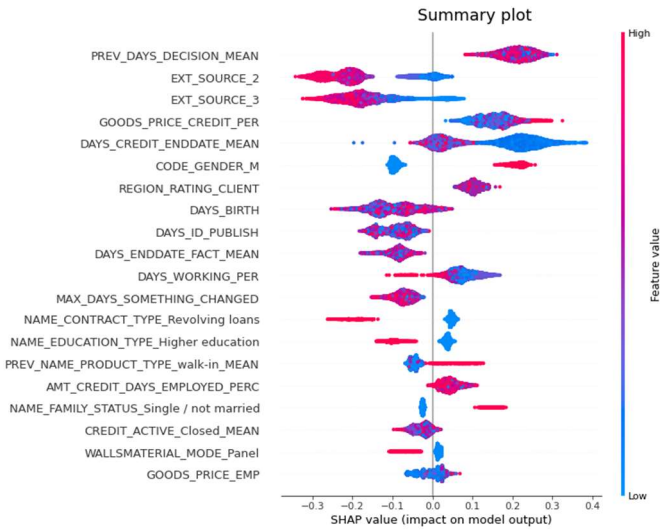
Malgré des scores plus élevés sur le jeu d'entrainement le Voting Classifier généralise moins bien que Lgbm, Ceci est sans doute dû au fait, qu'au terme de la recherche des poids optimums, il accorde plus de crédit aux prédictions de la Régression Logistique qu'a celles de Lgbm.
Soit c'est un biais du modèle, soit il aurait fallu plus d'essais.

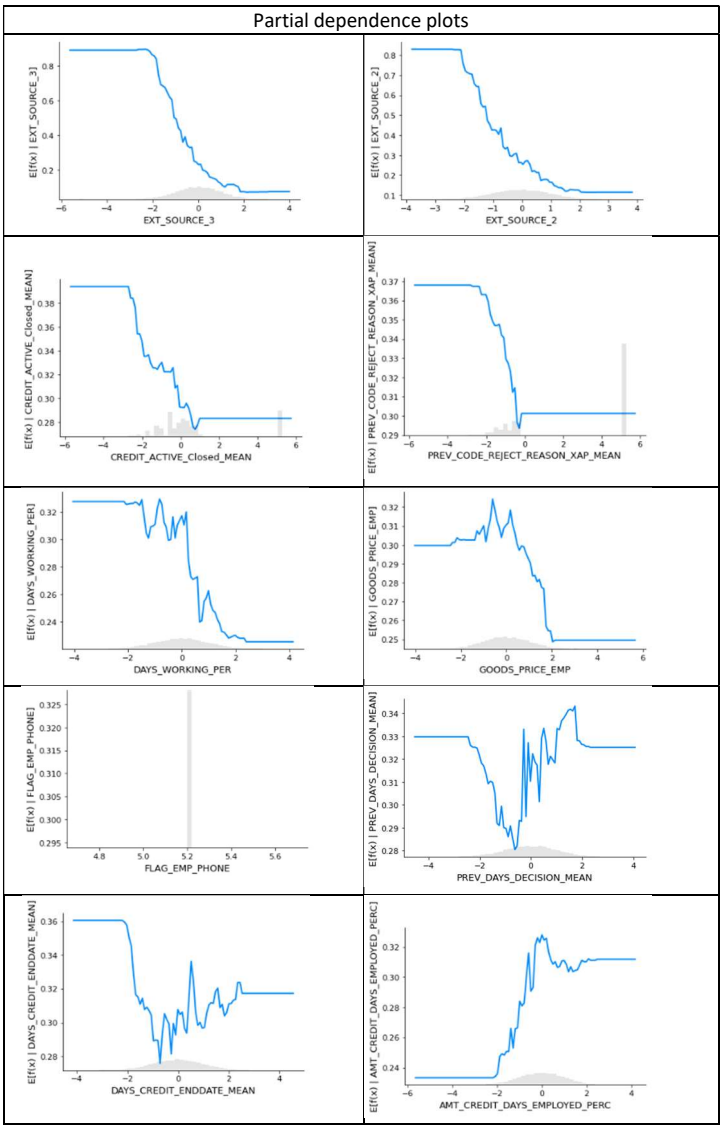
3. Modèle retenu

Pour l'élaboration du tableau de bord, je retiens donc ce Ligth Gradient Boosting Classifier, pour ses capacités de généralisation.

3.1. Interprétabilité du modèle

La bibliothèque SHAP permet d'expliquer la manière dont un modèle s'appuie sur les données fournies pour calculer ses prédictions.





Le summary plot permet de voir sur quelles variables le modèle s'appuie en priorité pour prédire le score.

En abscisses, l'impact de la variable sur les prédictions, les distributions des prédictions de tous les échantillons sont colorées en fonction des valeurs haute et basses de ladite variable.

Pour les variables catégorielles (qui ont été encodées de manière binaire) on peut interpréter facilement l'impact sur la prédiction. Par exemple la variable CODE_GENDER_M indique clairement que le modèle fait une différence nette sur cette variable.

Pour les variables quantitatives on peu observer des différences marquées pour la plupart, mais par exemple le variable PREV_DAYS_DECI SION_MEAN, qui apparait comme la plus importante pour le modèle, l'interprétation n'est pas évidente. C'est sur ce type de représentation que les partial dependence plots apportent un éclairage, ici on s'aperçoit que les valeurs extrêmes ont un impact de 0.33, mais que les valeurs autour de -1 on tendance à avoir un impact plus faible.

Pour pouvoir gagner en interprétabilité il faudrait toutefois, avoir des explications sur ce que représentent les variables EXT_SOURCE_2-3, qui ont un grand pouvoir décisionnaire mais une interprétabilité limitée d'un point de vue relation client, à cause du manque de documentation quant à leur composition.

4. Conclusion

L'optimisation des paramètres du modèle, sur la métrique créée, permet de prédire si un client fera défaut dans 64,55% des cas tout en ne rejetant que 28,69% de bons clients.

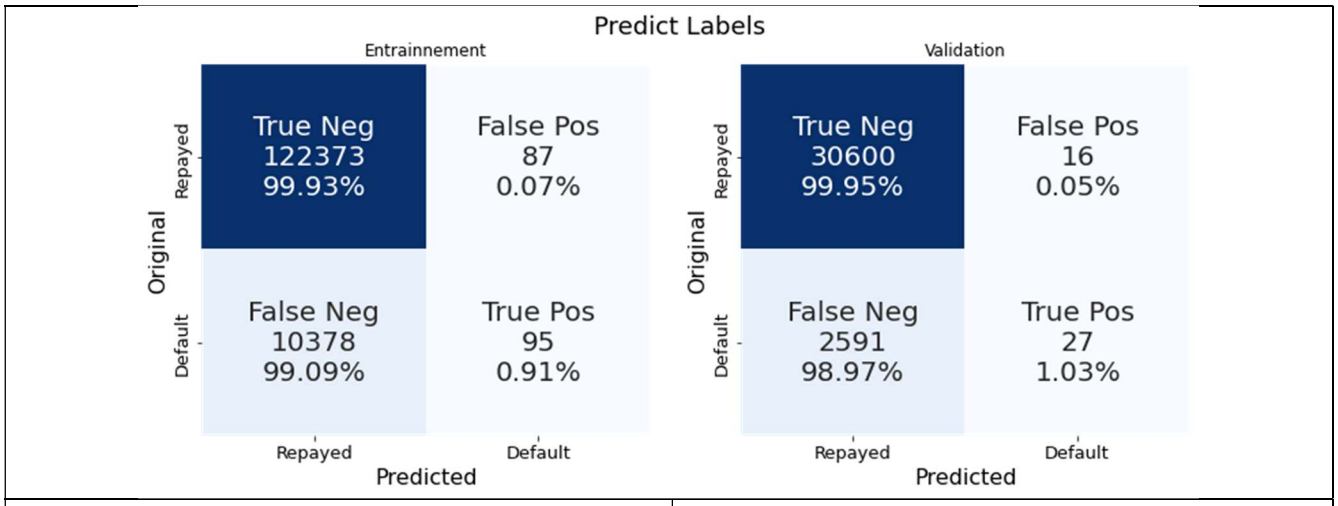
La démarche précédente permet d'obtenir un modèle de scoring maximisant les cas favorables au modèle économique de l'entreprise, étant entendu que la pondération choisie ici pourra évoluer en fonction des objectifs.

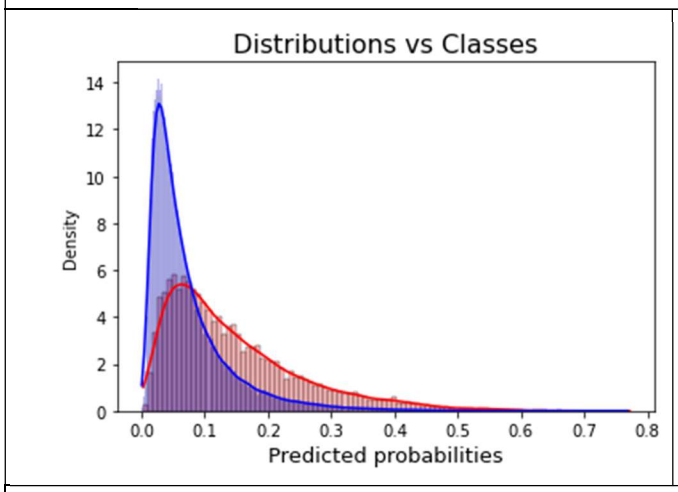
Pour aller plus loin :

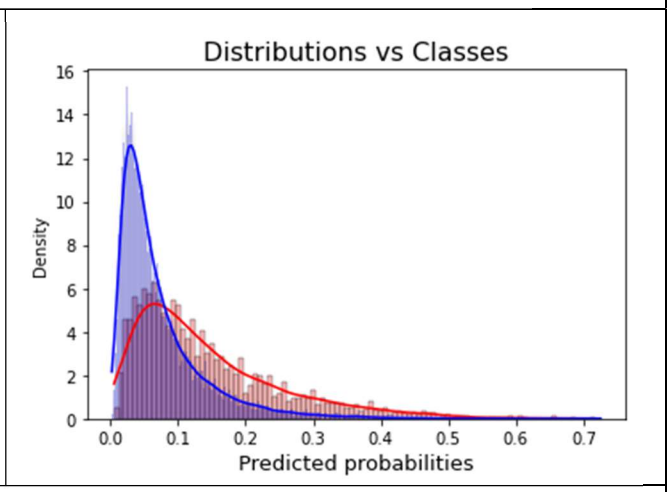
- Il faudrait surement changer de paradigme quant à la sélection des variables (pour des raisons de temps de calcul, le choix à peut-être été trop restrictif)
- L'impact du choix arbitraire du random seed tout au long de cette étude serait à évaluer quant aux performances et qualité de généralisation de la modélisation.
- De la même manière l'élargissement du corps électoral du méta modèle est lui aussi à évaluer sur ses capacités de généralisation, en effet sur la pondération il semble attacher plus d'importance aux prédictions de la régression logistique qu'a celles de ligth gradient boosting en raison de son score plus élevé sur le jeu d'entrainement, ce qui nuit à ses capacités de généralisation.

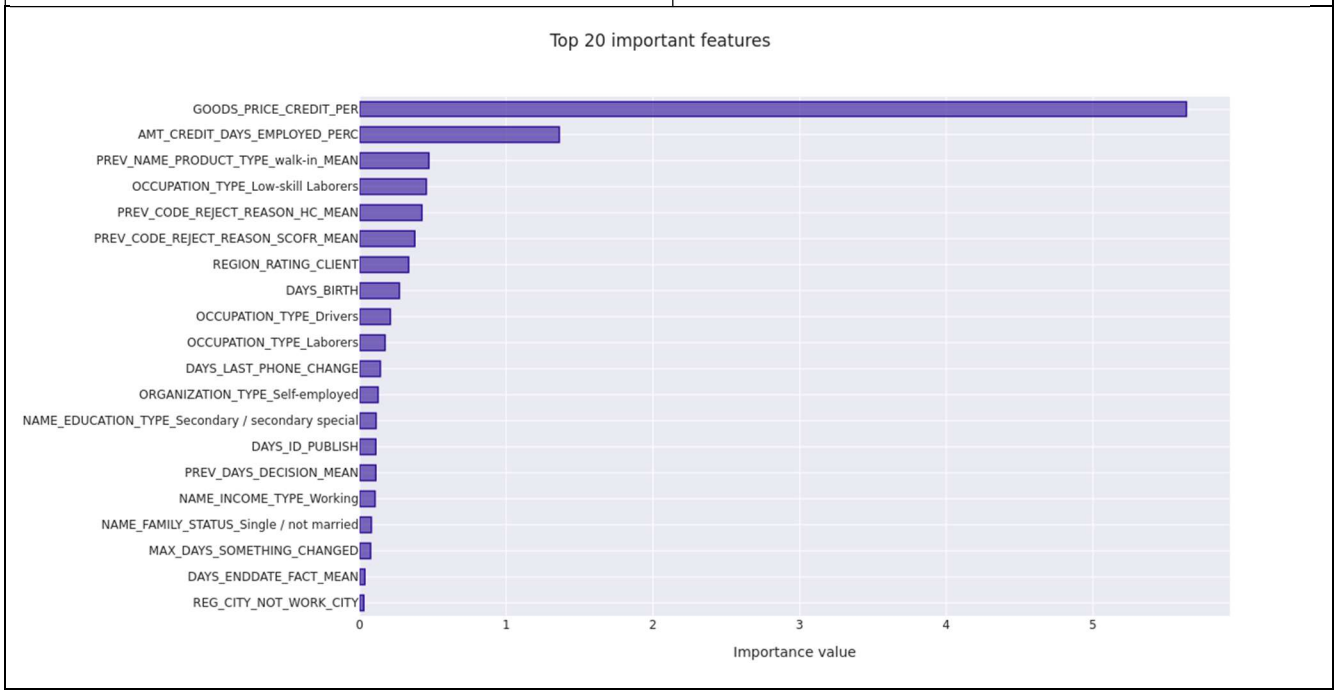
5. ANNEXES

5.1. (Base) Régression logistique

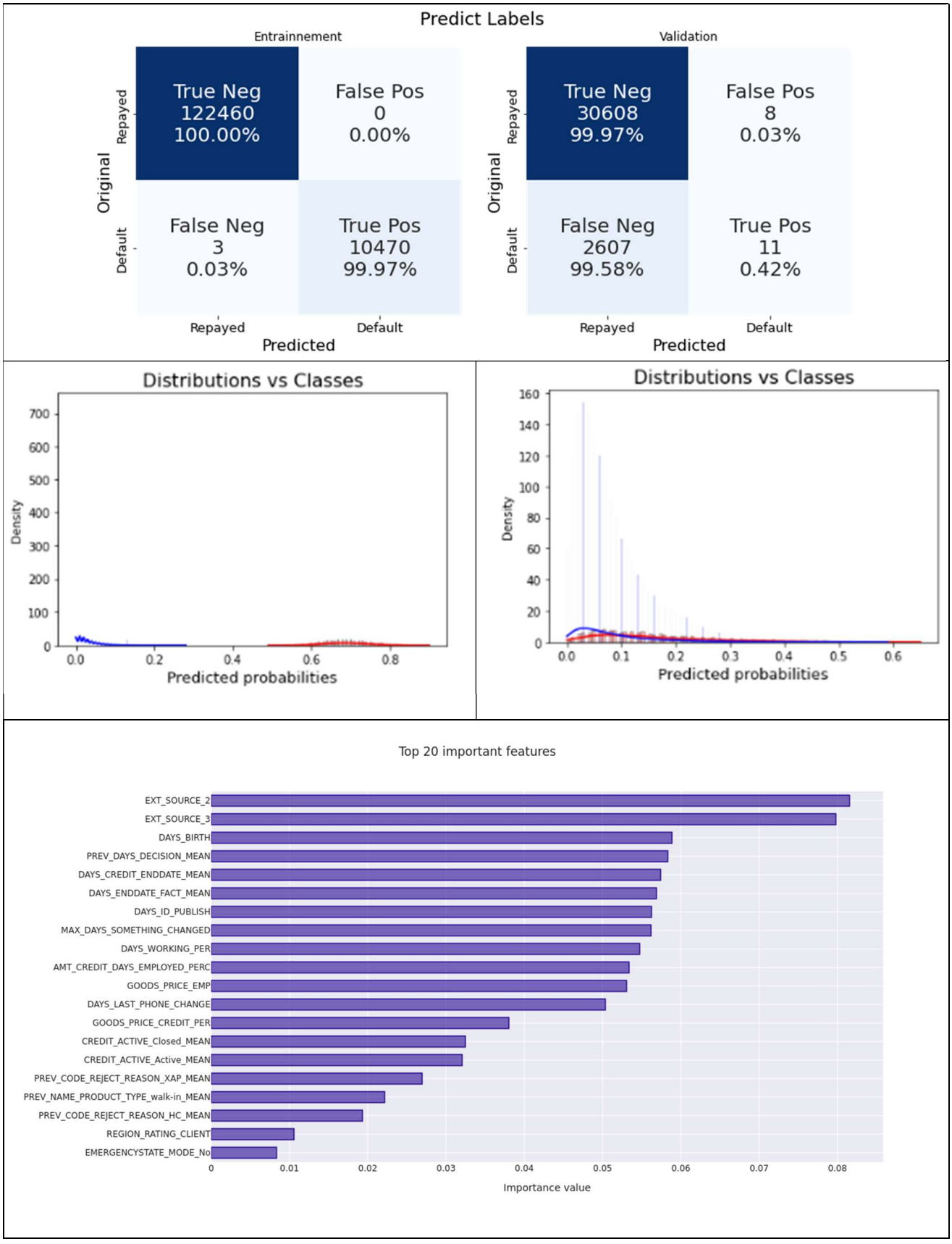




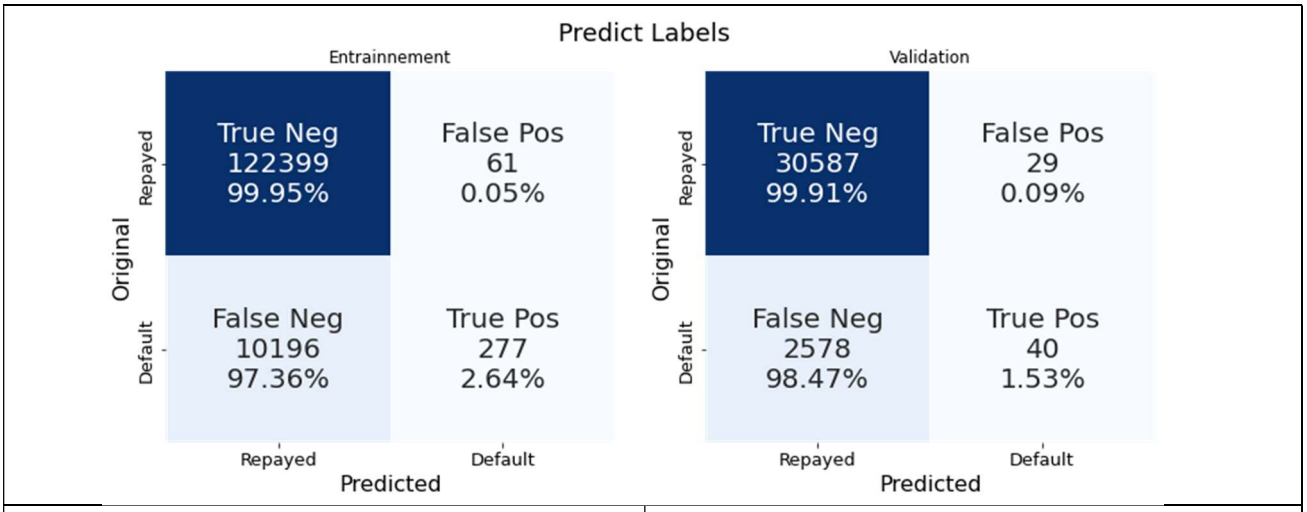


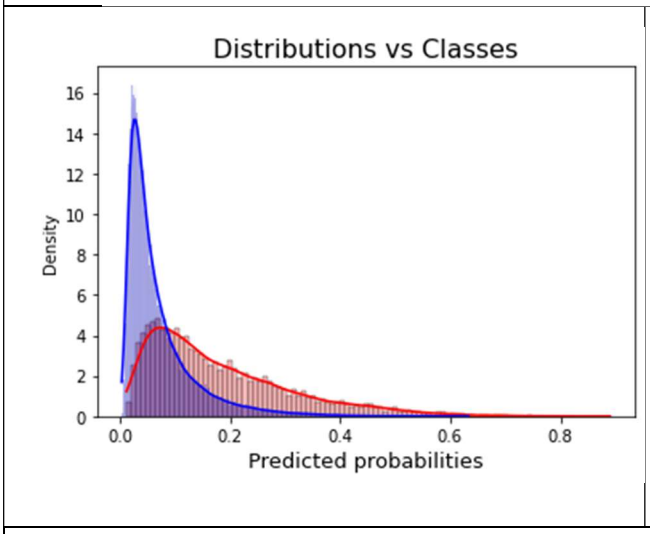


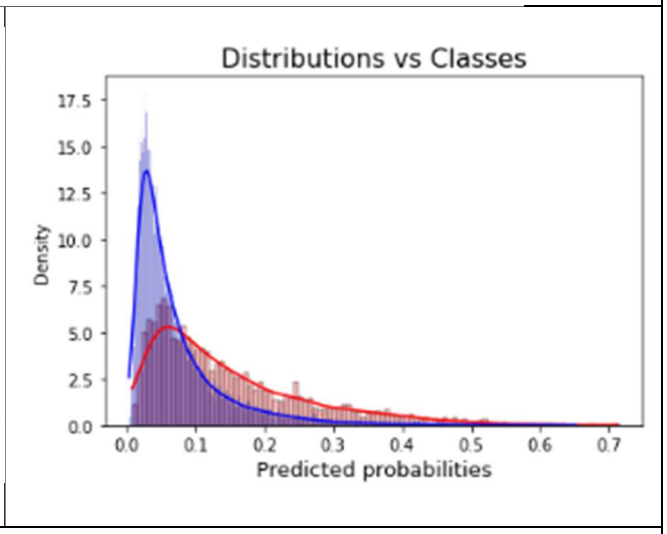
5.2. (Base) Random Forest Classifier

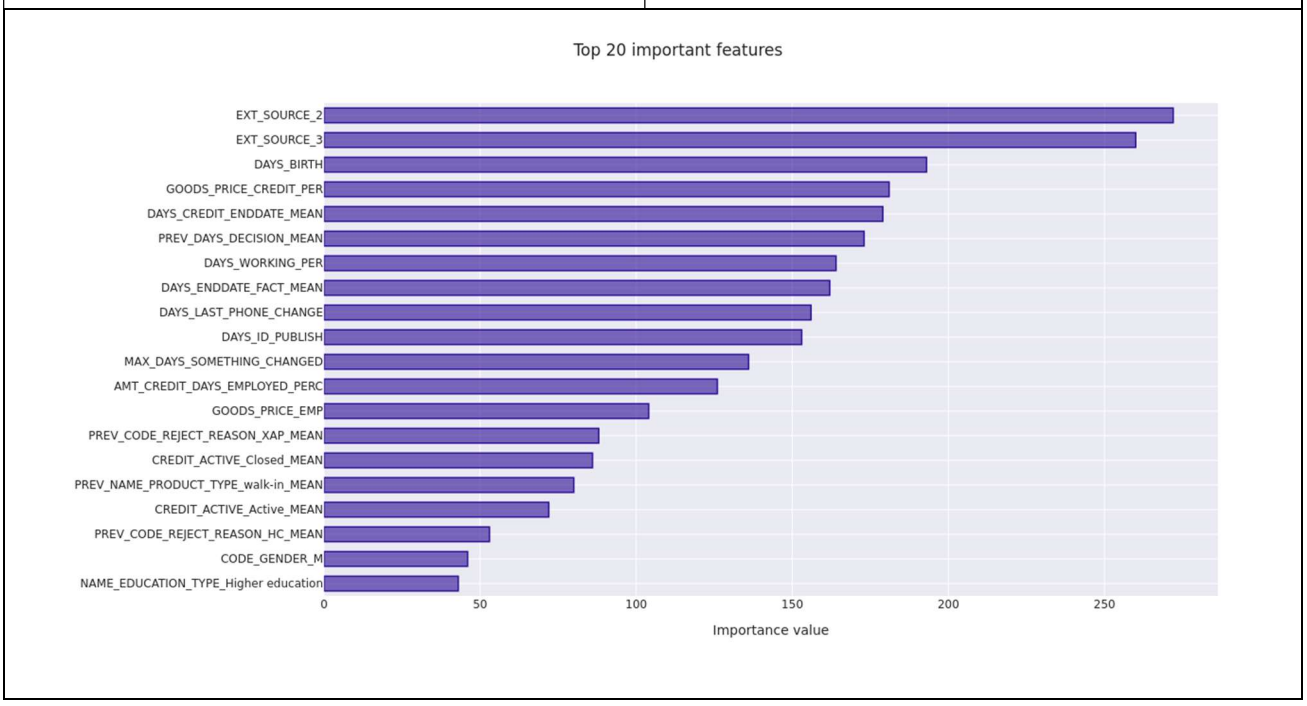


5.3. (Base) Light Gradient Boosting Classifier

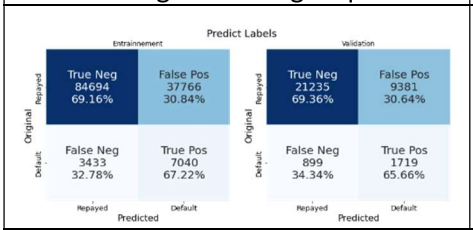
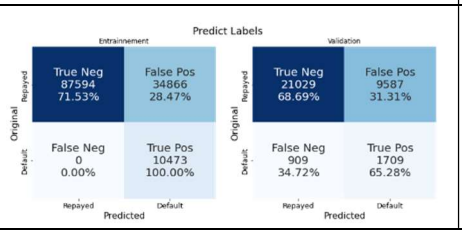
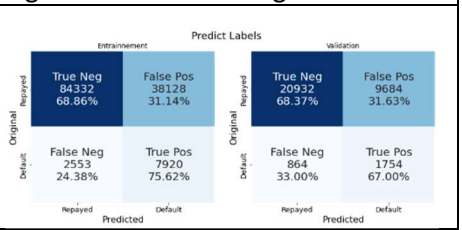
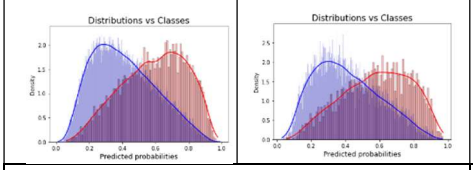
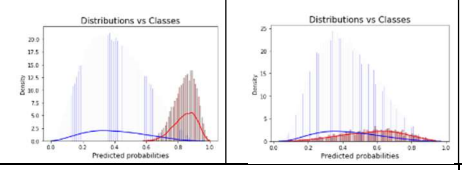
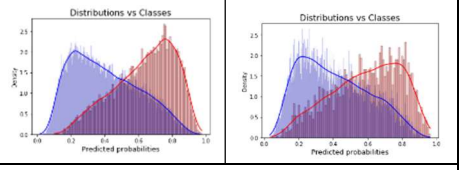
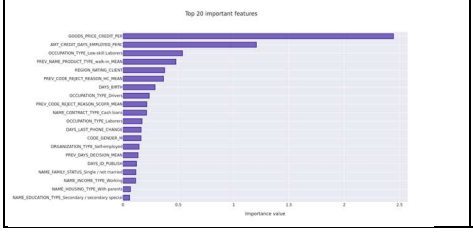
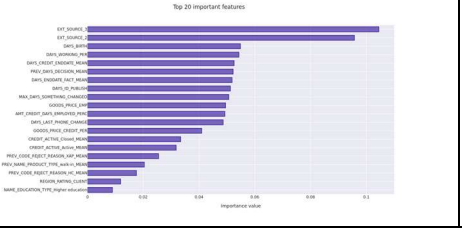
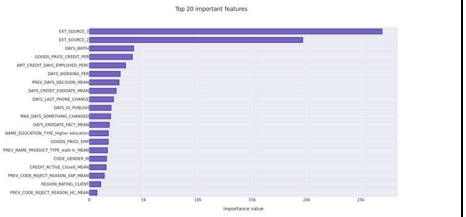




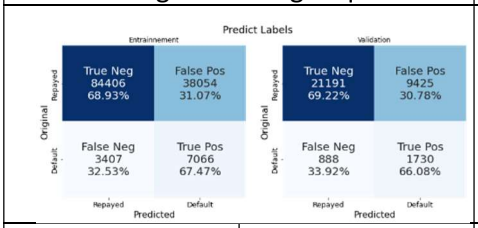
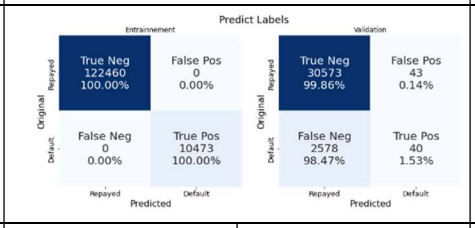
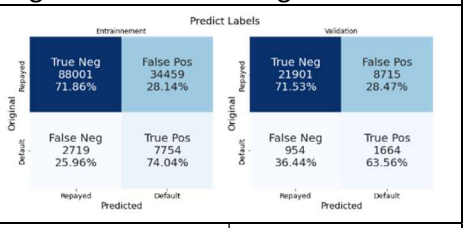
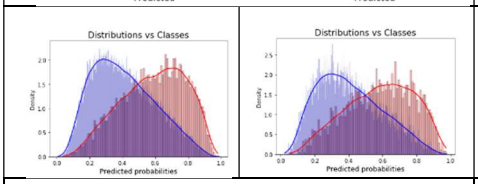
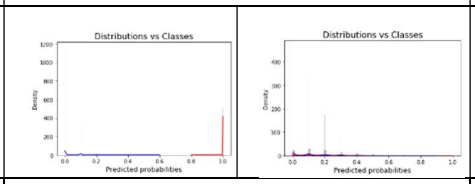
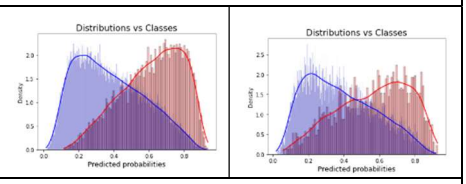
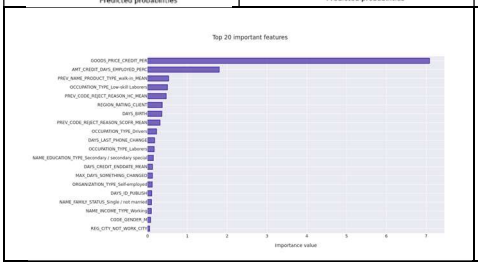
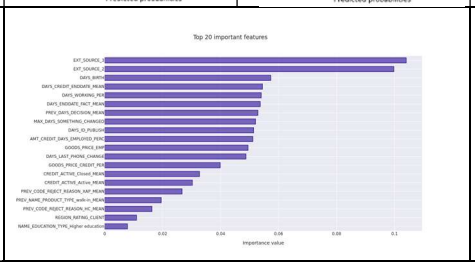
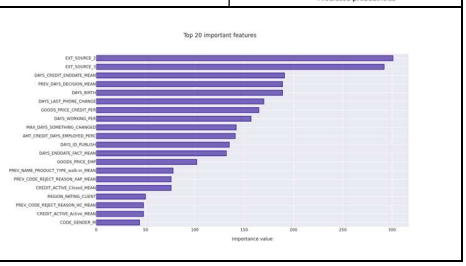




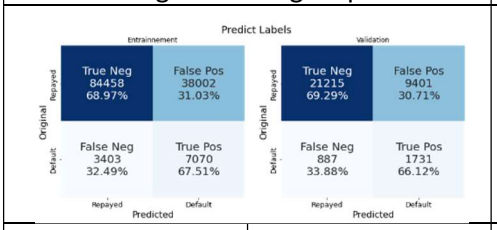
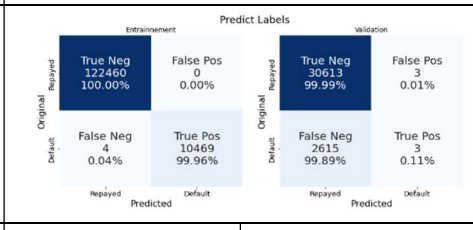
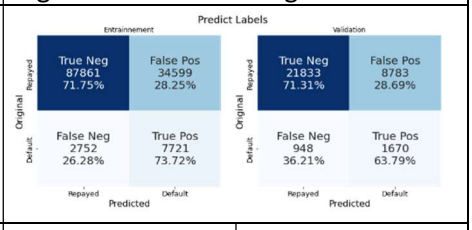
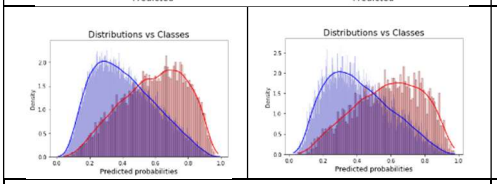
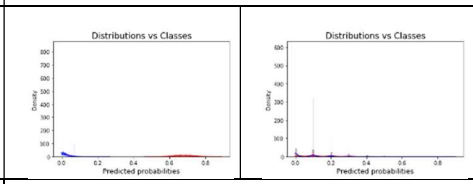
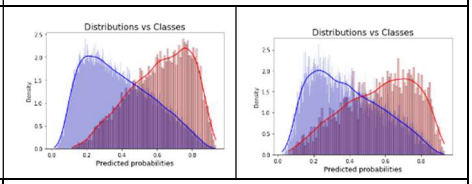
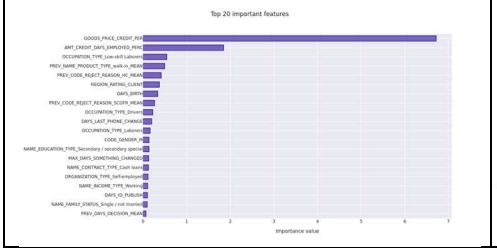
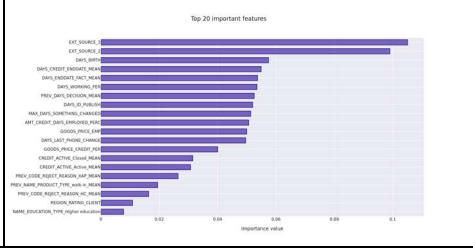
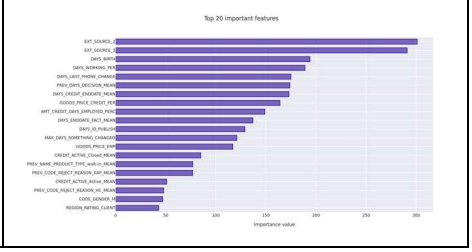
5.4. (Modèles) Under Sampling

Régression Logistique	Random Forest Classifier	Light Gradient Boosting Classifier
		
		
		

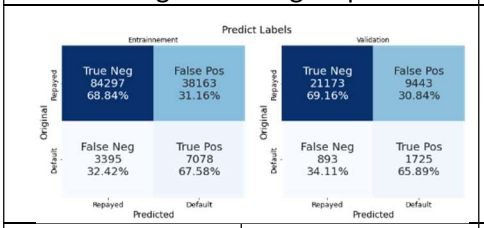
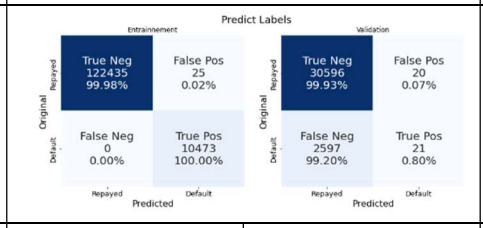
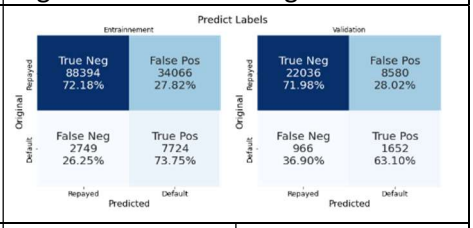
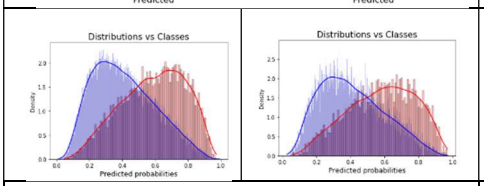
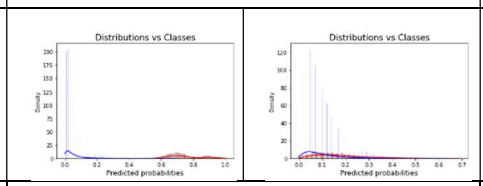
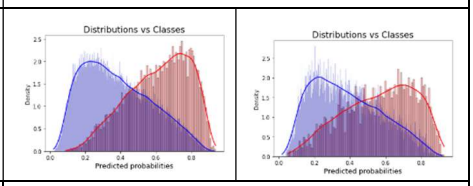
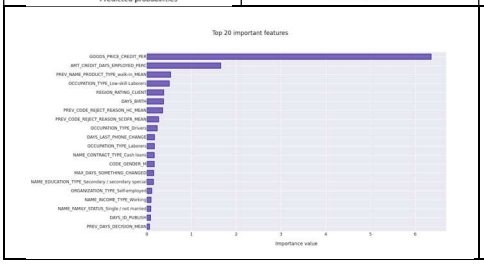
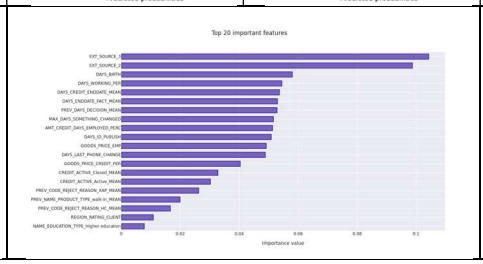
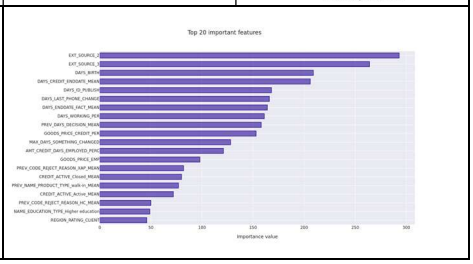
5.5. (Modèles) Over Sampling

Régression Logistique	Random Forest Classifier	Light Gradient Boosting Classifier
		
		
		

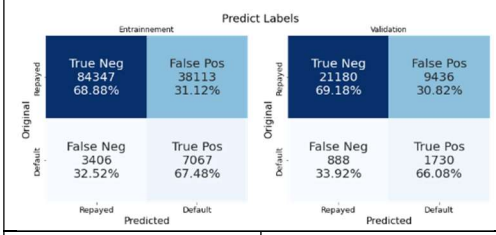
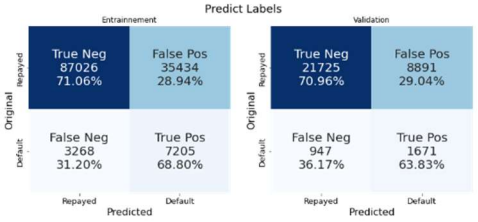
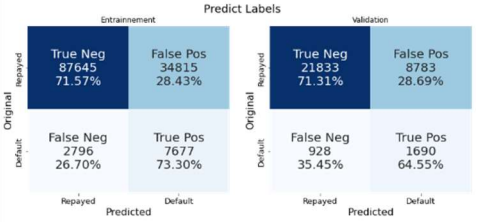
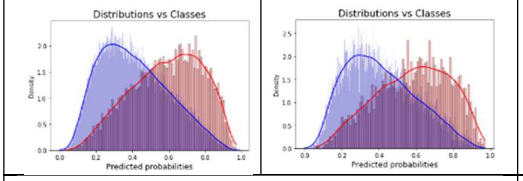
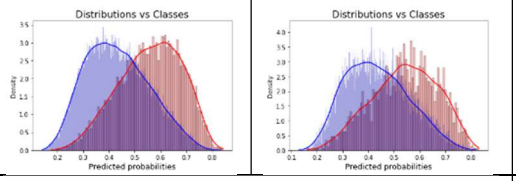
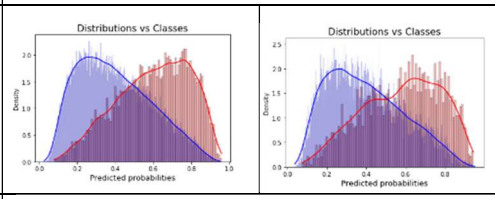
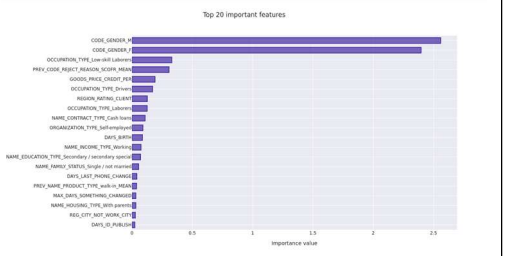
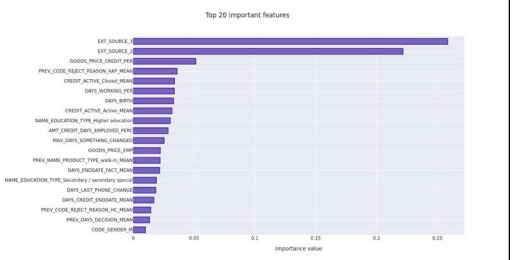
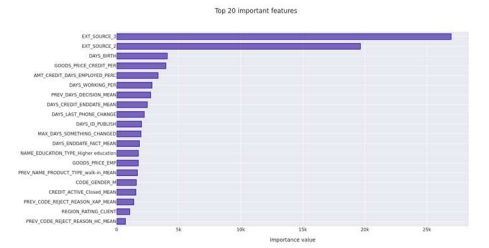
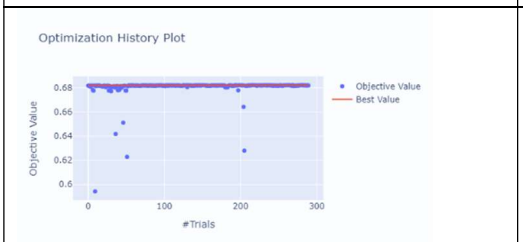
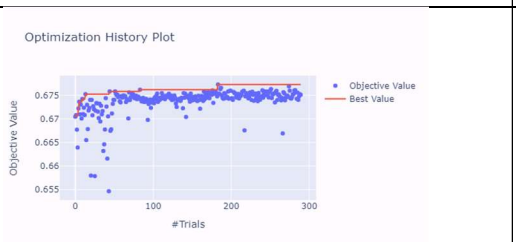
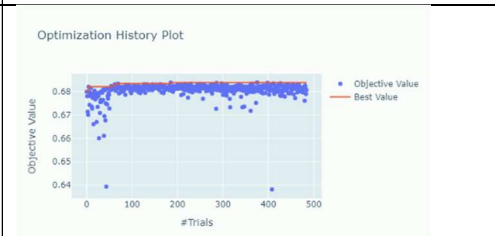
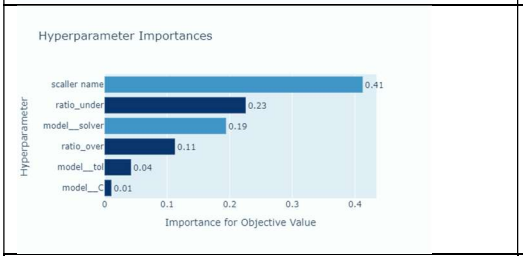
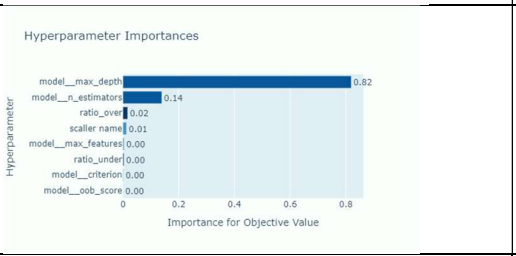
5.6. (Modèles) Pondération

Régression Logistique	Random Forest Classifier	Light Gradient Boosting Classifier
		
		
		

5.7. (Modèles) Combinaison de toutes les solutions

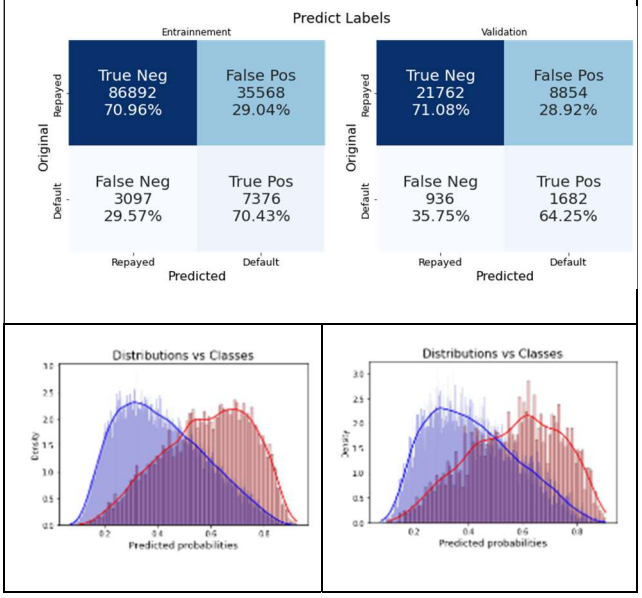
Regression Logistique	Random Forest Classifier	Light Gradient Boosting Classifier
		
		
		

5.8. Optimisation des modèles

Régression Logistique	Random Forest Classifier	Light Gradient Boosting Classifier
		
		
		
		
		
Meilleurs paramètres: ratio_over: 0.46115796940014214 ratio_under: 0.5023533111086315 scaller name: RobustScaler model__C: 387.7896540335958 model__tol: 2.4770441141413036e-05 model__solver: lbfgs	Meilleurs paramètres : ratio_over: 0.44477364180855467 ratio_under: 0.5713605874395502 scaller name: QuantileTransformer model__max_depth: 7 model__n_estimators: 180 model__criterion: gini model__max_features: log2 model__oob_score: True	Meilleurs paramètres : ratio_over: 0.0811606902694105 ratio_under: 0.9023419625243958 scaller name: QuantileTransformer model__boosting_type: dart model__max_depth: 8 model__learning_rate: 0.7132707614188026 model__n_estimators: 185 model__num_leaves: 4 model__min_split_gain: 0.5460401638083019 model__metric: binary_logloss model__subsample: 0.08269078501941346 model__reg_alpha: 63.53249436638803 model__reg_lambda: 11.075030110904528 model__importance_type: gain

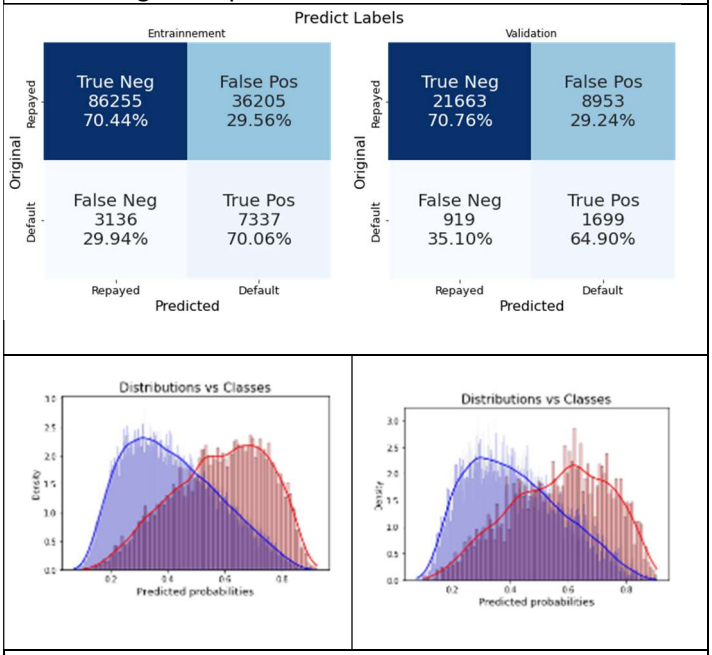
5.9. Voting Classifier sans pondération des votes

Soft Voting votes égaux



5.10. Voting classifier avec pondération des votes

Soft Voting votes pondérés



Meilleurs paramètres :

lr_weight: 0.7444383578110437

rf_weight: 0.09909535803614603

lgbm_weight: 0.4943384500495088