**Pierre-Yves Bescon**

**Parcours Data Scientist**

**Projet 7**

**Note méthodologique**

Table des matières

[1 Problématique 4](#__RefHeading___Toc6397_2364427719)

[1.1 Présentation du problème 4](#__RefHeading___Toc6399_2364427719)

[1.2 Jeu de données 4](#__RefHeading___Toc513_1191140541)

[1.3 Un problème d’imbalanced classification 5](#__RefHeading___Toc515_1191140541)

[1.4 Fonction de coût et métrique d’évaluation 6](#__RefHeading___Toc6403_2364427719)

[1.4.1 Fonction de coût 6](#__RefHeading___Toc969_147544)

[1.4.2 Métrique dévaluation 6](#__RefHeading___Toc971_147544)

[1.4.2.1 La courbe ROC 6](#__RefHeading___Toc973_147544)

[1.4.2.2 La courbe PR 6](#__RefHeading___Toc975_147544)

[1.4.2.3 Aire sous la courbe 7](#__RefHeading___Toc977_147544)

[2 Analyse et préparation des données 8](#__RefHeading___Toc519_1191140541)

[2.1 Corrélations 8](#__RefHeading___Toc979_147544)

[2.2 Gestion des outliers 10](#__RefHeading___Toc6401_23644277192)

[2.3 Hiérarchie des données 10](#__RefHeading___Toc981_147544)

[2.4 Feature engineering 11](#__RefHeading___Toc983_147544)

[2.5 Suppression de lignes 12](#__RefHeading___Toc985_147544)

[2.6 Gestion des valeurs manquantes 12](#__RefHeading___Toc6403_23644277192)

[3 Sélection du modèle 13](#__RefHeading___Toc521_1191140541)

[4 Optimisation du modèle 14](#__RefHeading___Toc523_1191140541)

[4.1 Création d'un jeu de validation 14](#__RefHeading___Toc1030_147544)

[4.2 Sélection des variables avec une importance > 0 14](#__RefHeading___Toc1032_147544)

[4.3 Nouvelle optimisation des paramètres 14](#__RefHeading___Toc1034_147544)

[4.4 Optimisation des imputers 17](#__RefHeading___Toc1038_147544)

[4.5 Optimisation des transformers 18](#__RefHeading___Toc1040_147544)

[4.6 Optimisation des scaler 18](#__RefHeading___Toc1042_147544)

[4.7 Optimisation des encoder 19](#__RefHeading___Toc1044_147544)

[4.8 Over et under sampling 19](#__RefHeading___Toc1046_147544)

[4.9 Bilan des métriques 20](#__RefHeading___Toc1048_147544)

[4.9.1 Evolution des métriques sur les jeux d'entrainement et de test 20](#__RefHeading___Toc1050_147544)

[4.9.2 Evolution des métriques sur les jeux d'entrainement et de test 20](#__RefHeading___Toc1052_147544)

[4.9.3 Métriques sur le jeu complet 21](#__RefHeading___Toc1054_147544)

[4.10 Entrainement final 22](#__RefHeading___Toc1056_147544)

[4.10.1 Le modèle 22](#__RefHeading___Toc1058_147544)

[4.10.2 Le pipeline 23](#__RefHeading___Toc1060_147544)

[4.10.3 Entrainement 23](#__RefHeading___Toc1062_147544)

[4.10.4 Interprétabilité du modèle 23](#__RefHeading___Toc1064_147544)

[4.10.5 Calibrage 25](#__RefHeading___Toc1066_147544)

[5 Conception du dashboard 26](#__RefHeading___Toc527_1191140541)

[5.1 Sélection du client 26](#__RefHeading___Toc6399_236442771921121)

[5.2 Sélection du seuil de score 26](#__RefHeading___Toc1079_147544)

[5.3 Risque du client 28](#__RefHeading___Toc1081_147544)

[5.4 Informations du client 28](#__RefHeading___Toc1083_147544)

[5.5 Influence des variables 28](#__RefHeading___Toc1085_147544)

[6 Déploiement du modèle sur le web 30](#__RefHeading___Toc529_1191140541)

[6.1 Déploiement sur AWS EC2 30](#__RefHeading___Toc6399_236442771921122)

[6.1.1 Lancement du modèle à distance 32](#__RefHeading___Toc6401_236442771921122)

[6.1.2 Lancement du dashboard à distance 33](#__RefHeading___Toc1087_147544)

[6.1.3 API 33](#__RefHeading___Toc1089_147544)

[7 Limites et améliorations possibles 34](#__RefHeading___Toc531_1191140541)

[7.1 Limites 34](#__RefHeading___Toc6399_23644277192111)

[7.2 Améliorations possibles 34](#__RefHeading___Toc6403_23644277192111)

## Problématique

## Présentation du problème

L’entreprise « Prêt à dépenser » propose des crédits à la consommation à des personnes ayant peu ou pas d’historique de prêt.

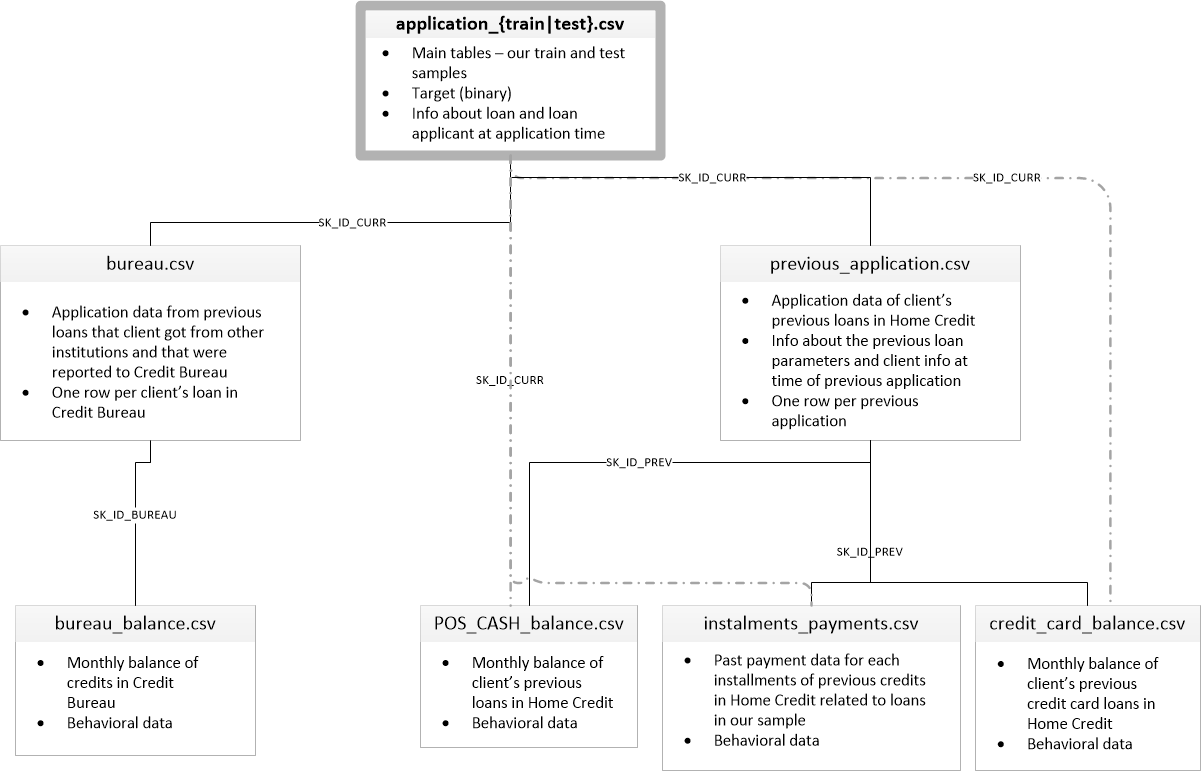
L’entreprise souhaite développer un modèle de scoring de la probabilité de défaut de paiement du client pour étayer la décision d'accorder ou non un prêt à un client potentiel en s’appuyant sur des sources de données variées (données comportementales, données provenant d'autres institutions financières, etc.).

Un dashboard interactif s’appuyant sur les informations des clients et sur le score prédit par le modèle pour chaque client aidera le chargé de clientèle à prendre une décision d’octroi de crédit et à expliquer cette décision au client.

Pour le développement du modèle, l’entreprise dispose d’un historique de 300 000 clients avec leurs informations personnelles, leur historique de prêt et l’information d’un défaut.

## Jeu de données

Le jeu de données est constitué de sept fichiers hiérarchisés comme indiqué dans le schéma suivant



application\_train : informations sur le client et sur le crédit en cours

bureau : informations sur les crédits historiques auprès d’autres établissements

previous\_application: informations sur les crédits historiques auprès de l’entreprise

bureau\_balance : informations sur les échéanciers des crédits auprès d’autres établissements

installment\_payments : informations sur les échéanciers des crédits auprès de l’entreprise

POS\_CASH\_balance : informations spécifiques sur les échéanciers des crédits personnels

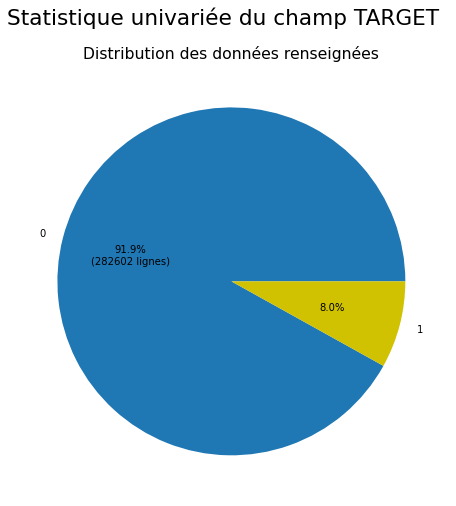
credit\_card\_balance : informations spécifiques sur les échéanciers des crédits revolvings

L’information d’un défaut sur le client est contenue dans application\_train sous forma d’un flag :

* 0 si le client n’a jamais connu de défaut
* 1 si le client a connu un défaut

## Un problème d’imbalanced classification

Le champ indiquant un défaut du client est nommé TARGET et est distribué de la manière suivante :

****

La proportion de client ayant fait un défaut est faible.

Pour ce type de problématique, la classique accuracy n’est pas une métrique d'évaluation pertinente, pour deux raisons:

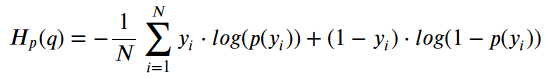
* l'accuracy ne permet pas de mesurer un scoring
* l'accuracy gère mal les problèmes d'imbalanced classification (il suffirait en effet de prédire tout le temps 0 pour avoir une accuracy de 92 %, mais en passant à côté de tous les clients à risque)

## Fonction de coût et métrique d’évaluation

## Fonction de coût

La fonction de coût est utilisée par l'algorithme durant l'entrainement, elle est minimisée afin de réduire les écarts entre les valeurs prédites et les valeurs réelles.

Celle utilisée est la LOG LOSS, fonction de perte par défaut de l'algorithme utilisé, dont voici la formule :

[](https://medium.datadriveninvestor.com/understanding-the-log-loss-function-of-xgboost-8842e99d975d)

## Métrique dévaluation

La métrique d'évaluation est utilisée après l'entraînement, on cherche à la maximiser afin d'améliorer la performance de l'algorithme.

Comme expliqué précédemment, l'accuracy, qui est la métrique classique en classification, n'est pas pertinente lorsque les classes à prédire sont fortement déséquilibrées.

On utilisera donc ici les deux métriques utilisées pour les problèmes d'imbalanced classification: la courbe précision-rappel ainsi que la courbe ROC.

## La courbe ROC

La courbe ROC représente l'évolution de la sensibilité (taux de vrais positifs) en fonction de 1 - spécificité (taux de faux positifs) quand on fait varier le seuil t.

* la sensibilité est : TP / (TP + FN) = TP / P
* la spécificité est : TN / (TN + FP) = TN / N

Elle est adaptée pour mesurer les performances d'un modèle de scoring.

## La courbe PR

La courbe PR (precision-recall) représente l'évolution de la sensibilité (taux de vrais positifs) en fonction de la précision quand on fait varier le seuil t.

* la sensibilité est : TP / (TP + FN) = TP / P
* la précision est : TP / (TP + FP)

Elle est adaptée pour mesurer les performances d'un modèle de scoring déséquilibré (imbalanced classicafication)

.

## Aire sous la courbe

Afin d'avoir une métrique mesurable, l'aire sous les courbes est utilisée pour mesurer la performance du modèle.

## Analyse et préparation des données

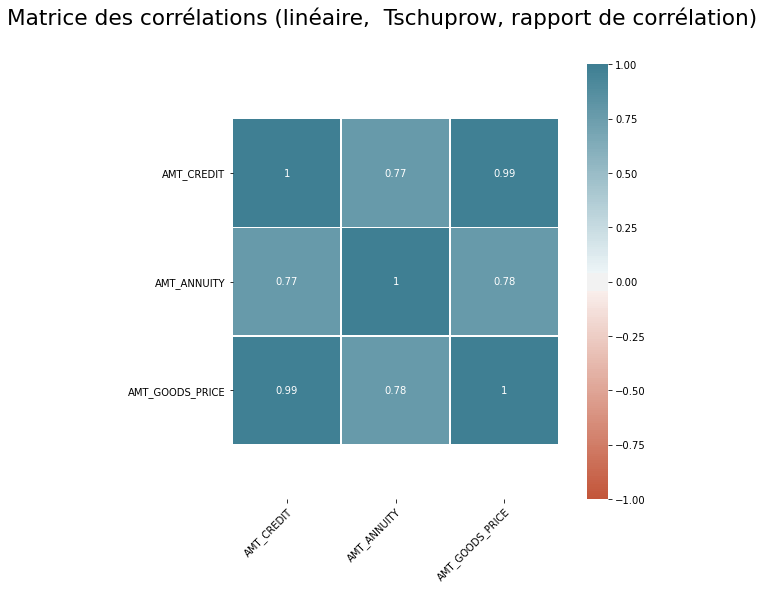
## Corrélations

Il existe des corrélations fortes entre certaines données.

Cependant, comme expliqué dans le chapitre suivant, un modèle basé sur des arbres de décision sera utilisé, qui est insensible aux variables corrélées. La suppression de variables corrélées entraine même une baisse des performances.

Les matrices de corrélations suivantes sont donc montrées à titre indicatif. Toutes les corrélations ne sont pas montrées

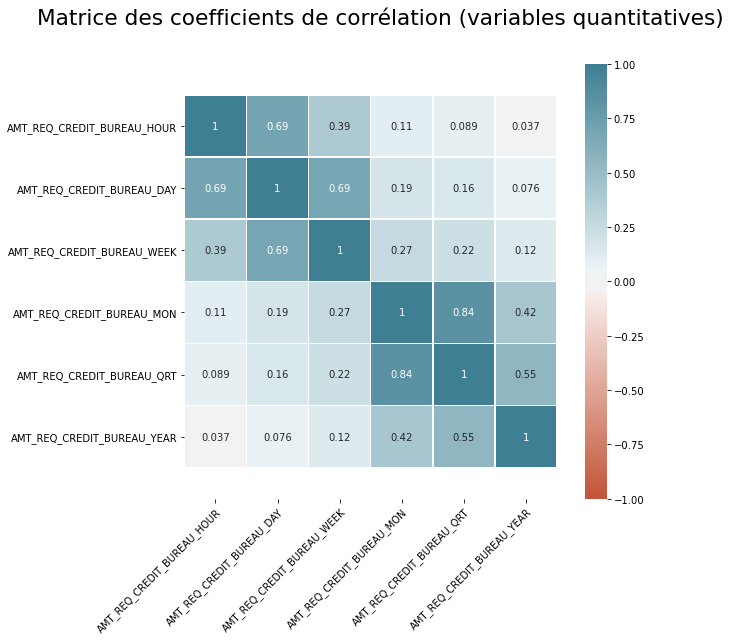
*Corrélation entre les champs amount*

**

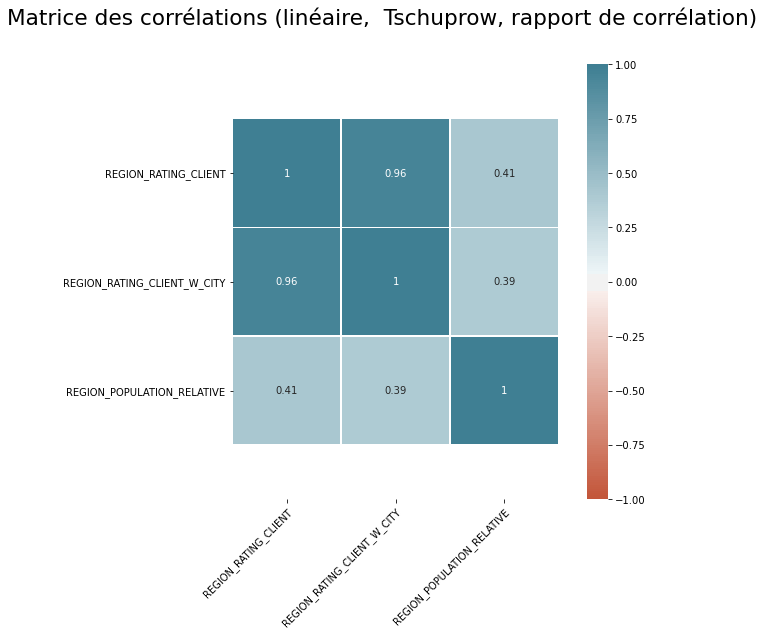
*Corrélation entre les champs days*

**

*Corrélation entre les variables requêtes*



*Corrélation entre les champs rating*



## Gestion des outliers

Plusieurs variables possèdent des outliers.

La suppression des lignes avec des outliers entraine une perte importante de lignes car les outliers des différentes variables sont distribuées sur des lignes différentes.

Par ailleurs, comme expliqué dans le chapitre suivant, un modèle basé sur des arbres de décision sera utilisé, qui est peu sensible aux outliers.

Les outliers sont donc conservé tels quels.

## Hiérarchie des données

Le jeu de données est constitué de sept fichiers hiérarchisés de la manière suivante :

application

------ Bureau

-------------Bureau balance

------Previous application

-------------Pos\_cash\_balance

-------------Installments\_payments

-------------Credit\_card\_balance

Les données de niveau inférieur sont remontées au niveau supérieur en moyennant les variables numériques et en prenant le mode des variables catégorielles.

Il est à noter que presque toutes les lignes de Pos\_cash\_balance et de Credit\_card\_balance ont une ligne correspondante dans Installment\_payments.

## Feature engineering

Quelques variables ont été créées ou modifiées.

*Contract status*

Le contract status de previous application est remplacé par contract status de la dernière ligne correspondant dans Pos\_cash\_balance ou dans Credit\_card\_balance

*Champs requêtes*

Les champs requête existent au niveau jour, trimestre, semaine, mois et année. Le nombre de requête aux différents niveaux devrait augmenter avec le niveau (plus de requêtes sur un an que sur un mois), ce qui n'est pas nécessairement le cas. On peut donc présumer qu'à chaque niveau (ex mois) est retranché le nombre de requêtes du niveau inférieur (ex semaine).

Pour corriger cela, on remplace chaque variable requête par la somme des variables requêtes de niveau inférieur (ex: mois = moi + semaine + jour).

*Champs documents*

Il y a plusieurs flags documents qui indique si un document donné a été fourni par le client. On créé une variable qui donne le nombre de documents fournis par le client.

*Fréquence des crédits passés*

*On créée une variable qui donne le nombre de jours moyen entre deux crédits passés soit auprès de l'entreprise, soit auprès d'une autre institution financière*

*Nombrte de crédits passés*

On créée des variables indiquant le nombre de crédits déjà actifs, remboursés et refusés

*Ecart entre les montants et échéances réels et attendus*

Quatre variables sont créées, donnant la moyenne des écarts positifs et négatifs entre les montants et les échéances réelles et attendus

## Suppression de lignes

Les lignes suivantes sont supprimées du jeu d’entrainement :

* Lignes de previous\_appication avec FLAG\_LAST\_APPL\_PER\_CONTRACT=Y
* Lignes de previous\_application avec NAME\_CONTRACT\_STATUS != 'Approved','Refused'
* Lignes de installement\_payment avec AMT\_Instalment = 0

## Gestion des valeurs manquantes

Les valeurs suivantes sont remplacées par nan: XNA, XAP, unknown

Pour la variable OWN\_CAR\_AGE, les valeurs manquantes sont remplacées par 100

Toutes les valeurs manquantes de variables numériques des tables inférieures (bureau, nureau\_balance, previous\_application

Les valeurs manquantes de applications sont-elles remplacées par la moyenne pour les valeurs numériques et le mode pour les valeurs catégorielles.

## Sélection du modèle

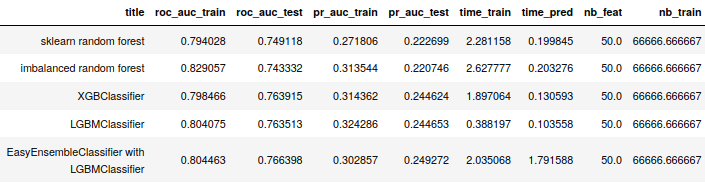
La sélection du modèle s'est faite entre plusieurs modèles basés sur des arbres de décision:

* RandomForestClassifier de sklearn
* BalancedRandomForestClassifier de imblearn
* GradientBoostingClassifier de sklearn
* LGBMClassifier de lightgbm
* XGBClassifier de xgboost
* EasyEnsembleClassifier de imblearn

Les principaux paramètres de tous ces classifiers sont optimisés afin de sélectionner celui qui présente les meilleures performances en terme de métrique et de temps d'entrainement et de prédiction.

Dans le but de limiter les temps de traitement, ces modèles sont optimisés sur un nombre restreint de 50 colonnes et de 100000 lignes avec une cross-validation de 3. Les 50 variables sont la fusion des 30 meilleures variables du RandomForestClassifier et du XGBClassifier, entrainés préalablement après une optimisation du nb\_estimatorts et une fixation du max-depth à 5 pour éviter le sur-apprentissage

Le tableau suivant résume les performances des modèles :



Les trois modèles XGBClassifier, LGBMClassifier et EasyEnsembleClassifier présentent des performances similaires en terme de courbe ROC et de courbe PR, mais LGBMClassifier est nettement plus rapide à entraîner et un peu plus rapide aussi pour la prédiction.

Le modèle sélectionné est donc LGBMClassifier

## Optimisation du modèle

## Création d'un jeu de validation

Le jeu de données (300000 lignes) est divisé en deux : 100000 lignes sont mises de côté pour servir de jeu de validation, et sur les 200000 lignes restantes, des cross-validation de 10 seront effectuées.

Dans les paragraphes qui suivent, les métriques de train sont des moyennes des 10 cross-validation sur les jeux d'entrainement et les métriques de test dont des moyennes des 10 cross-validation sur les jeux de test.

## Sélection des variables avec une importance > 0

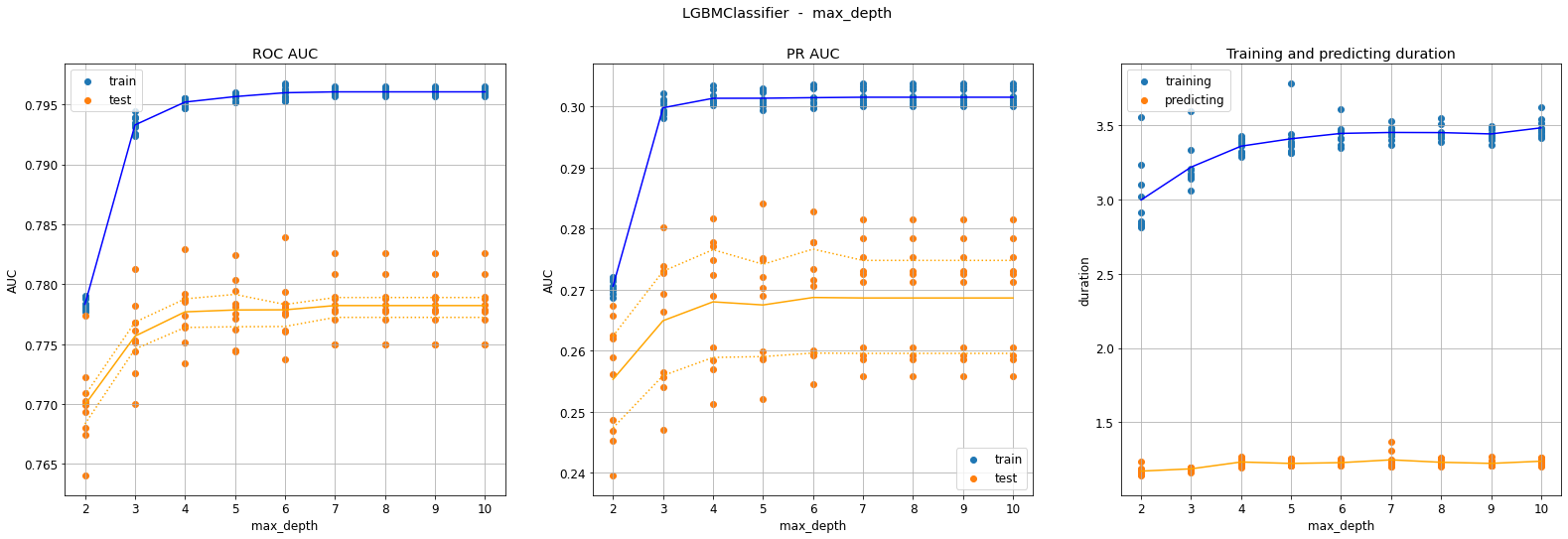
Dans un premier temps, on effectue un calcul de feature importance avec les paramètres optimisés à l'étape précédente. Les variables avec une importance nulle sont écartées. Restent 122 variables sur 229

## Nouvelle optimisation des paramètres

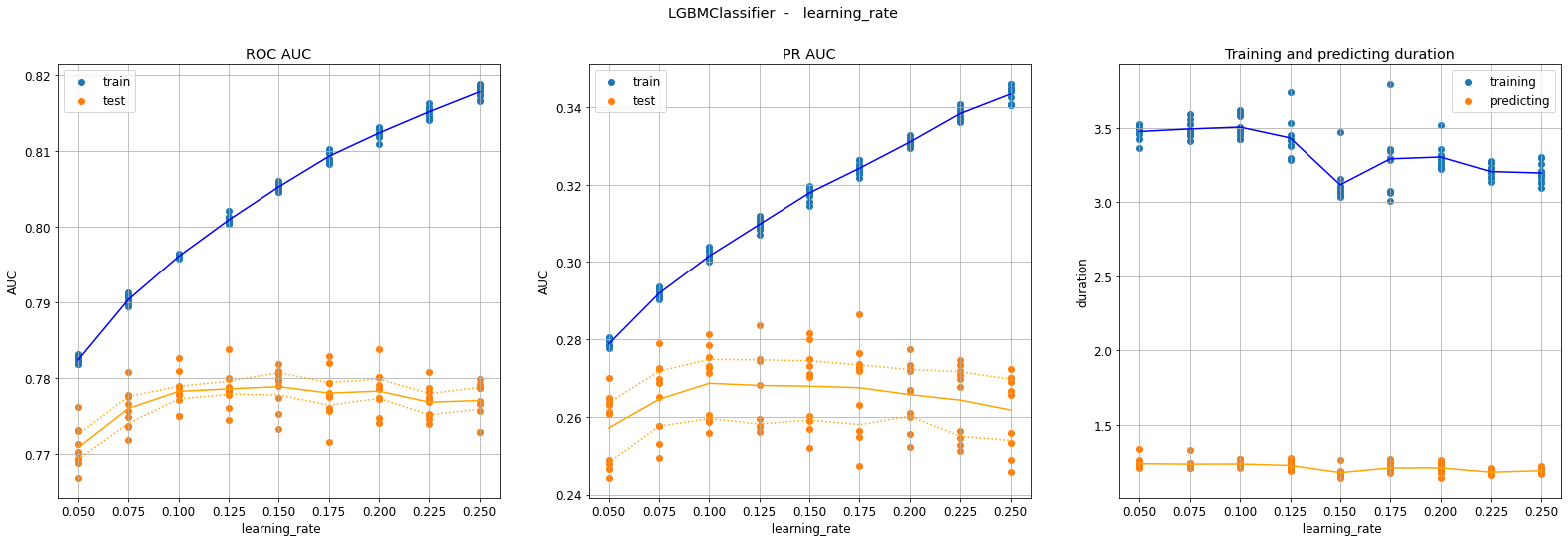
Tous les paramètres du modèle sélectionné sont optimisés une seconde fois, cette fois-ci sur l'ensemble du jeu de données (environ 300000 lignes). Les paramètres sont initialisés avec les valeurs issues de la sélection du modèle.

Pour chaque paramètre, on représente trois courbes représentant, la ROC AUC, la PR AUC et les temps d'entrainement et de prédiction.

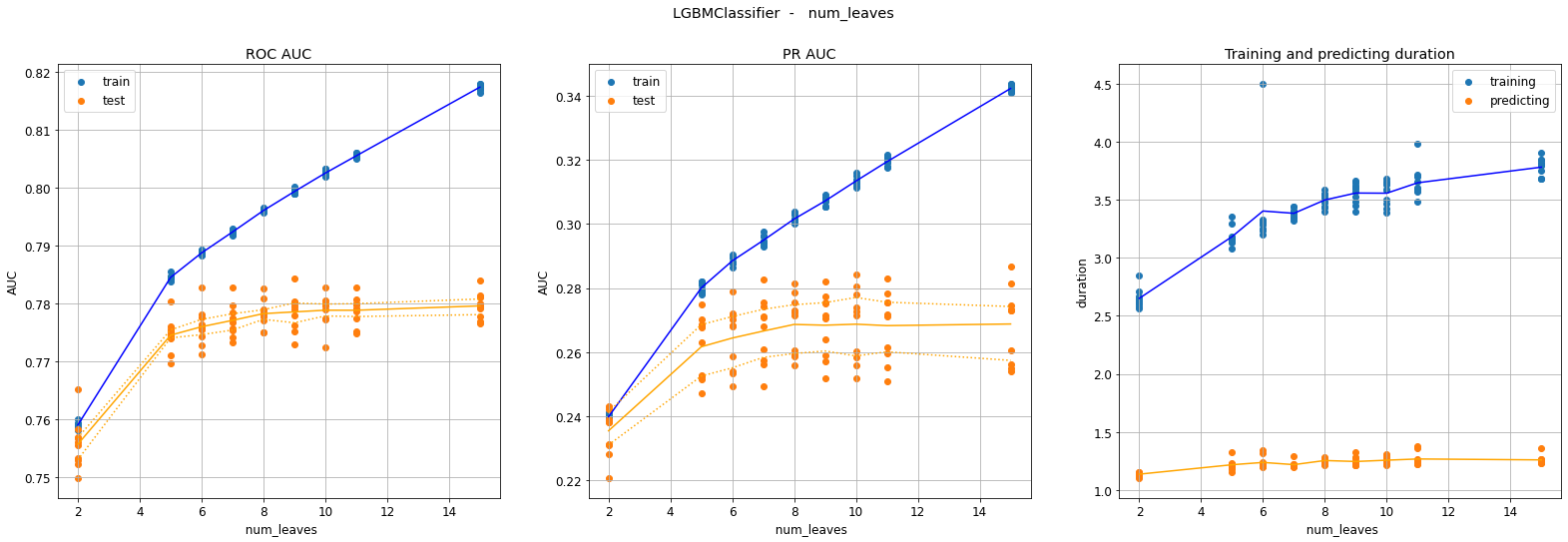
max\_depth:



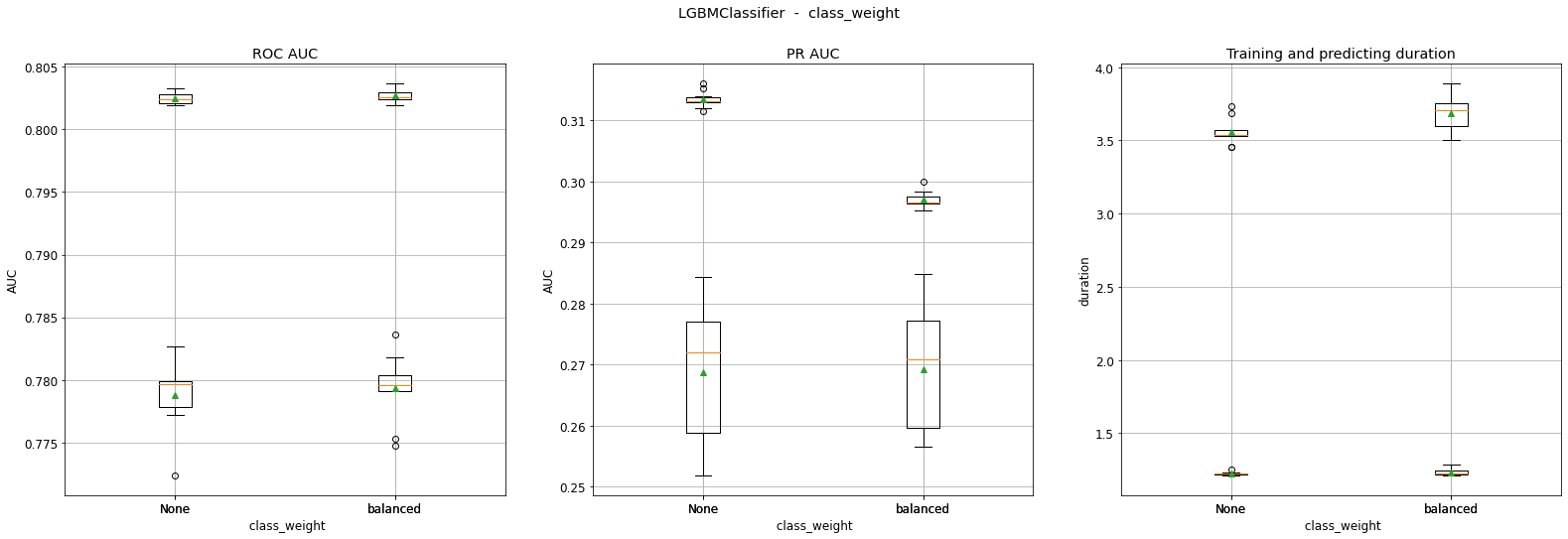
learning\_rate:



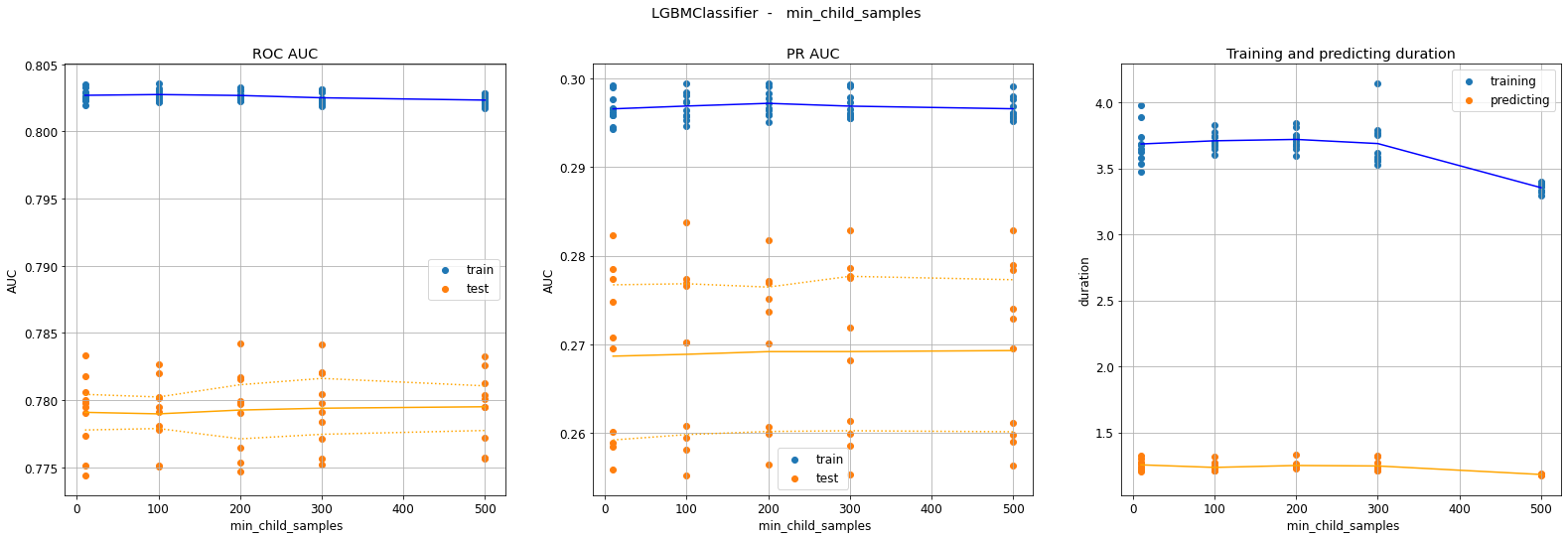
num\_leaves:



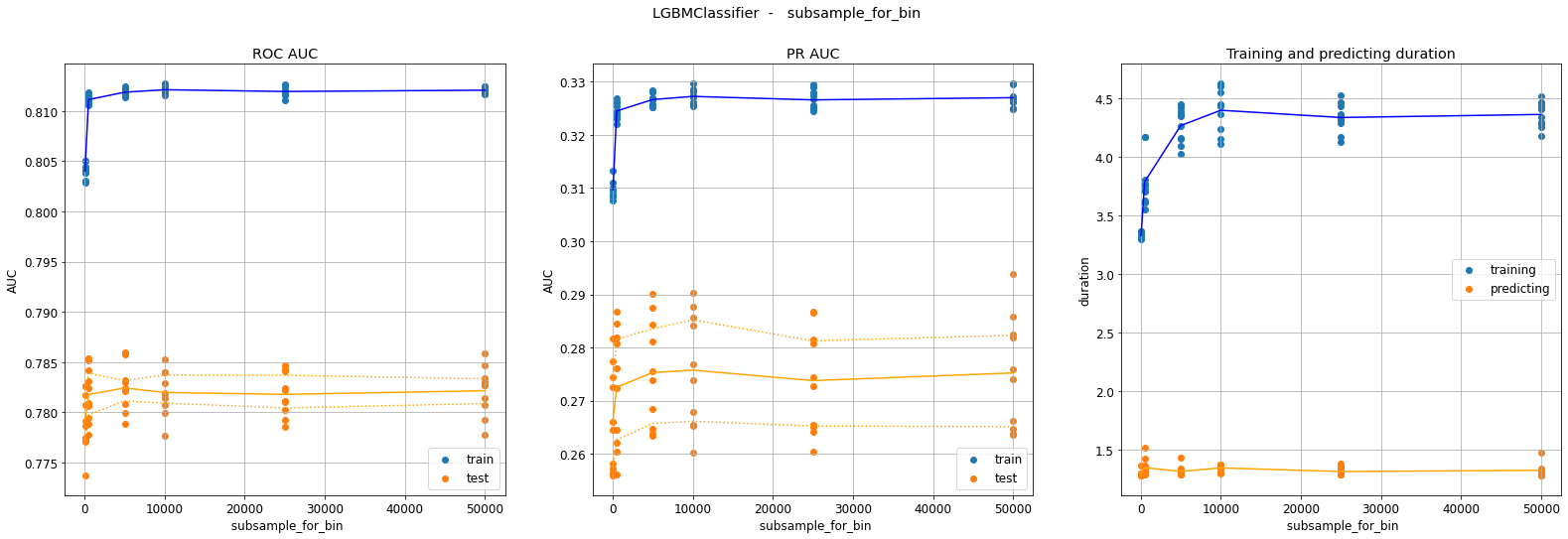
class\_weight:



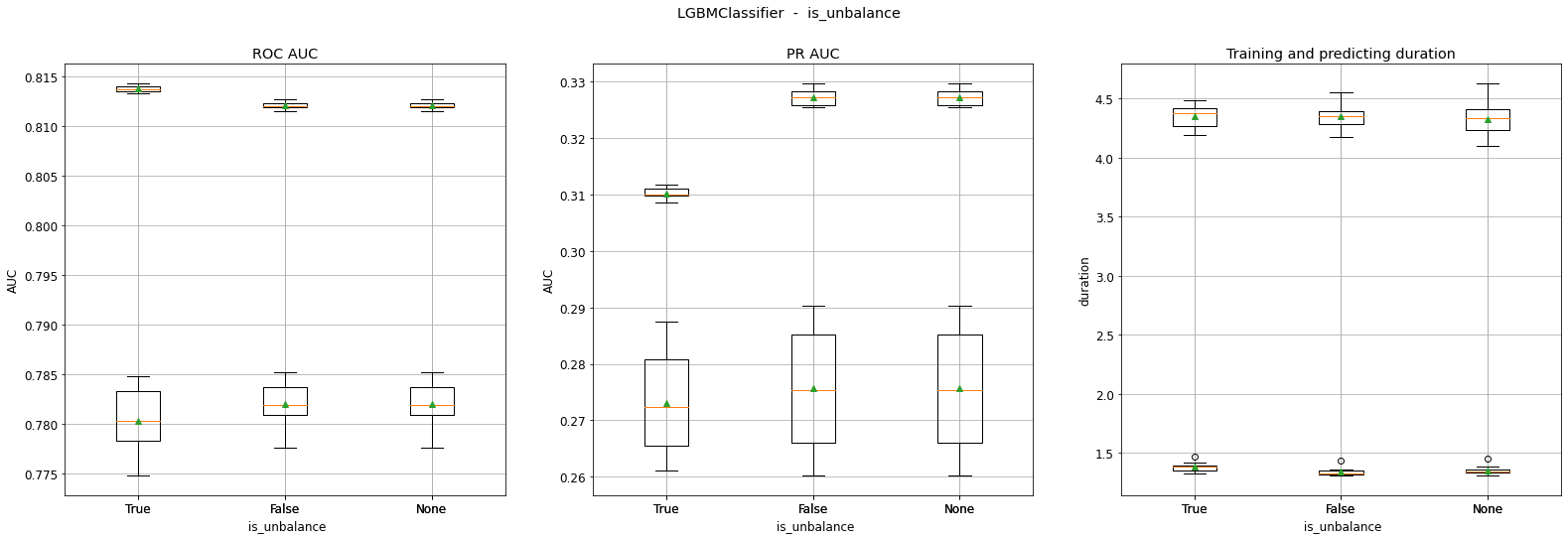
min\_child\_samples:



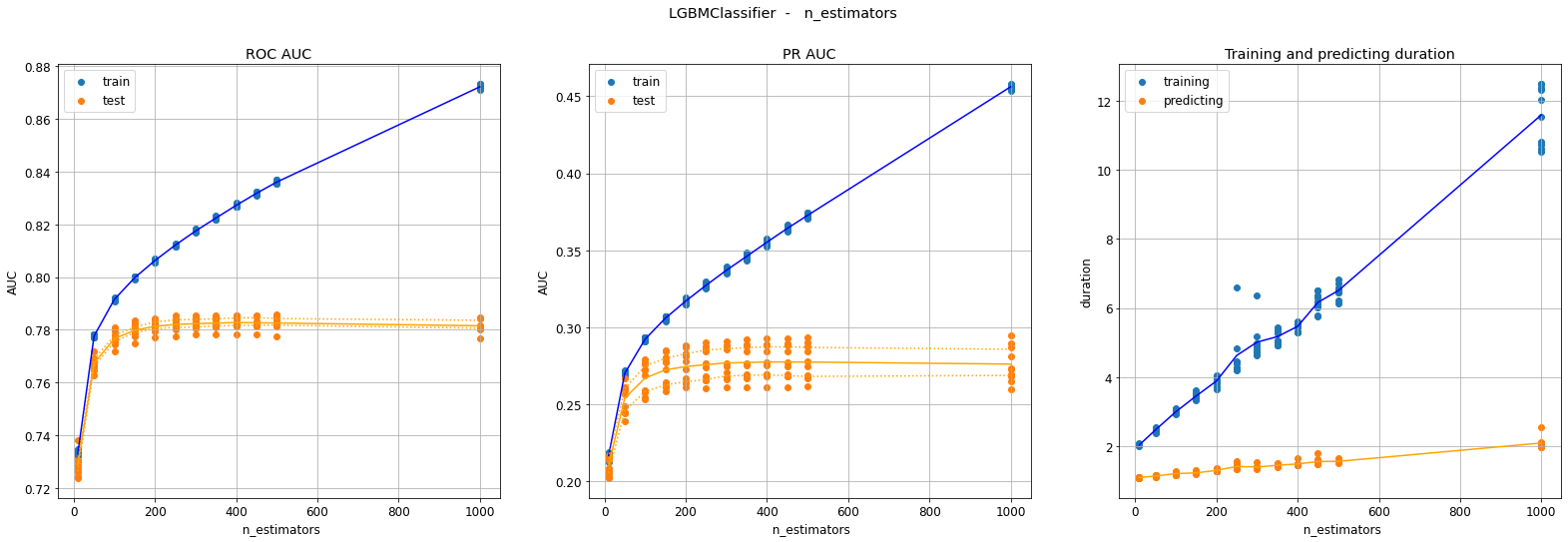
sub\_sample\_for\_bins:



is\_unbalance:

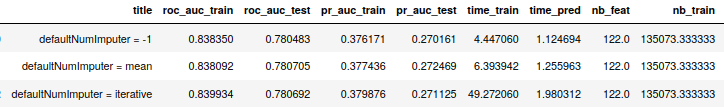


n\_estimtors:

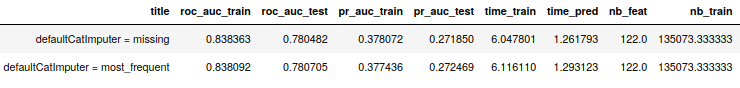
****

## Optimisation des imputers

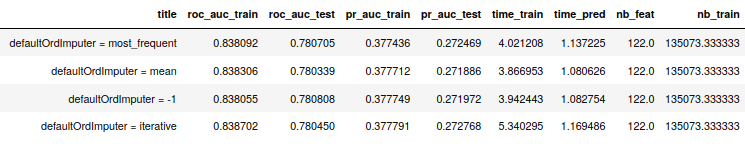
Pour les variables numériques, plusieurs imputers ont été testés (imputer -1, mean imputer, iterative imputer), le plus efficace en terme de performance et de durée de traitement est le mean imputer



Pour les variables catégorielles, deux imputeurs ont été testés: le most\_frequent et marquer par "missing" les valeurs manquantes. Le most\_frequent imputer est retenu.



Pour les variables ordinal, quatre imputer ont été testés: most\_frequent, mean, -1, iterative imputer. Le most\_frequent imputer est retenu.

****

## Optimisation des transformers

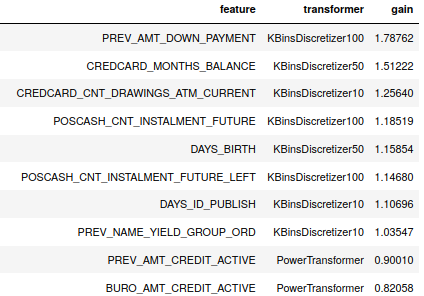
Cinq transformers sont testés sur les variables numériques :

* PowerTransformer
* QuantileTransformer
* KbinsDiscretizer10
* KbinsDiscretizer50
* KbinsDiscretizer100

Pour chaque variable, on calcule le gain en performance sur la courbe ROC et sur la courbe PR et on additionne ces deux gains (ou perte)

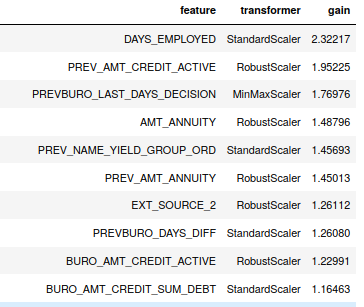
Pour chaque variable, on conserve le meilleur transformer et on l'enregistre dans un pickle.

26 variables sont ainsi transformées, voici les 10 pour lesquelles le gain (\*1000) en performance est le plus important:



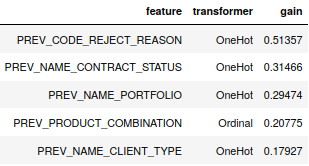
## Optimisation des scaler

La même optimisation est effectuée sur le scaler pour les variables numériques :



## Optimisation des encoder

La même optimisation est effectuée sur le encoders des variables catégorielles :



Le gain en passant en OneHotEncoder est trop faible, toutes les variables catégorielles sont donc encodées avec un ordinal encoder, qui est sans impact sur les modèles basés sur des arbres de décision.

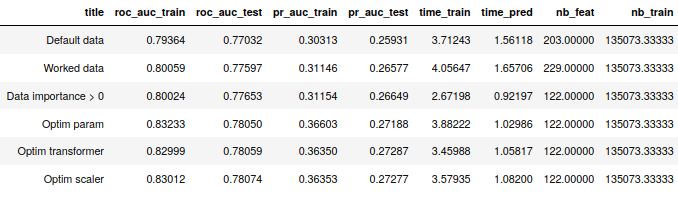
## Over et under sampling

Plusieurs essais d'over et under sampling ont été effectués, sans amélioration des métriques, mais avec une forte dégradation des performance en termes de temps de traitement.

Aucun over/under sampling ne sera donc effectué.

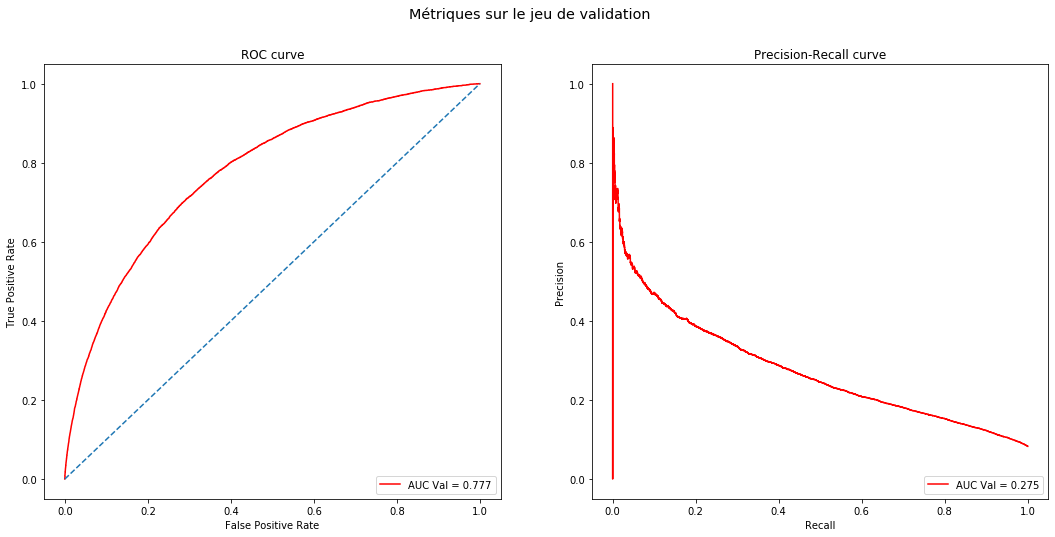
## Bilan des métriques

## Evolution des métriques sur les jeux d'entrainement et de test

Le tableau suivant résume l'ensemble des métriques obtenues avec lightgbm, sur le jeu de données sans feature engineering, sur le jeu de données avec feature engineering, après suppression des variables sans importance, après optimisation des paramètres et après optimisation des transformers

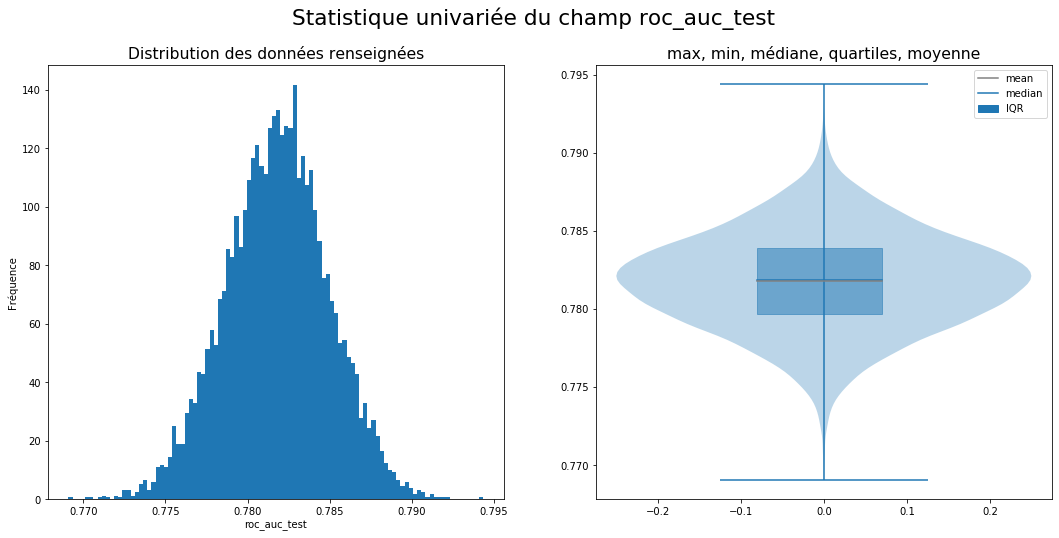
## Evolution des métriques sur les jeux d'entrainement et de test

Les graphiques suivants représentent les courbes ROC et PR sur le jeu de validation

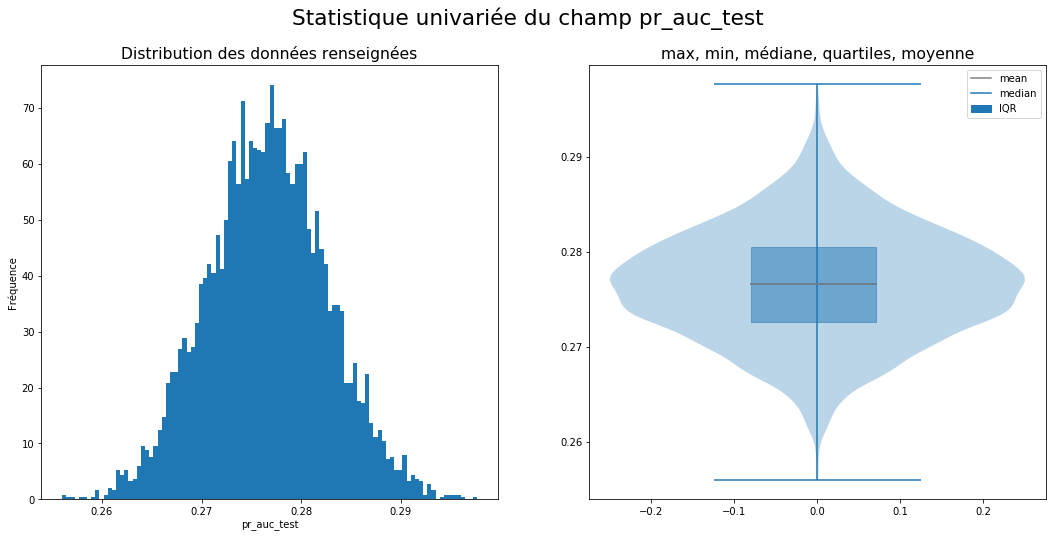
****

## Métriques sur le jeu complet

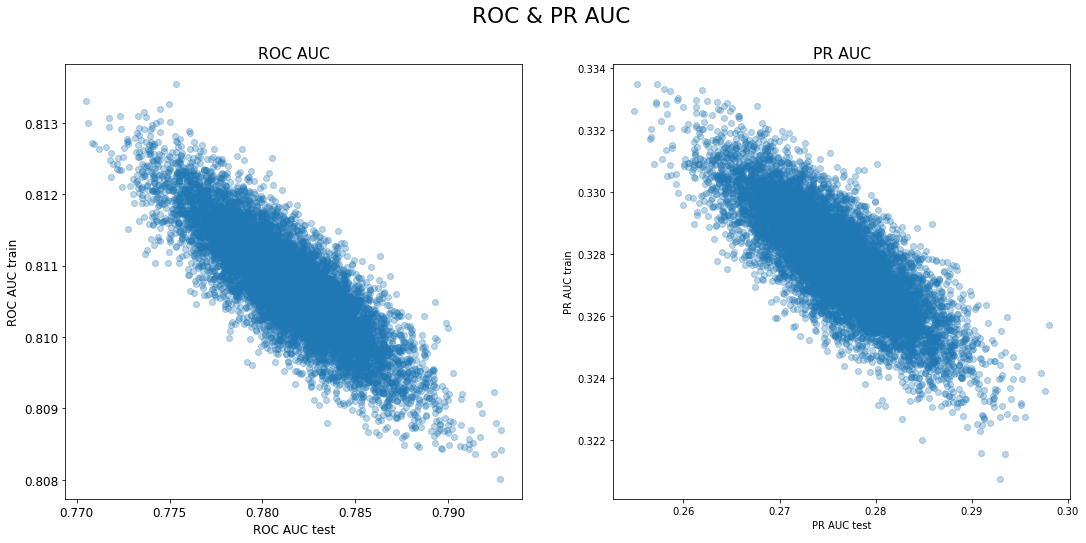
Le graphique suivant représente la distribution des ROC\_AUC après 10000 train-test split et entrainements, sur le jeu de test issu du jeu de données complet avec split de 20%.

****

Le graphique suivant représente la distribution des PR\_AUC après 10000 train-test split et entrainements, sur le jeu de test issu du jeu de données complet avec split de 20%.

****

Enfin les deux nuages de points suivants représentent les ROC\_AUC et PR\_AUC de test en fonction des ROC\_AUC et PR\_AUC d'entrainement

****

## Entrainement final

## Le modèle

Le modèle est le LGBMClassifier avec les paramètres suivants :

model = LGBMClassifier(max\_depth=7,

learning\_rate=0.125,

num\_leaves=8,

min\_child\_samples=300,

subsample\_for\_bin=25000,

n\_estimators=300,

n\_jobs=-1)

## Le pipeline

Le modèle est intégré dans un pipeline avec l'ensemble des transformations, scalers, imputers et encoders:



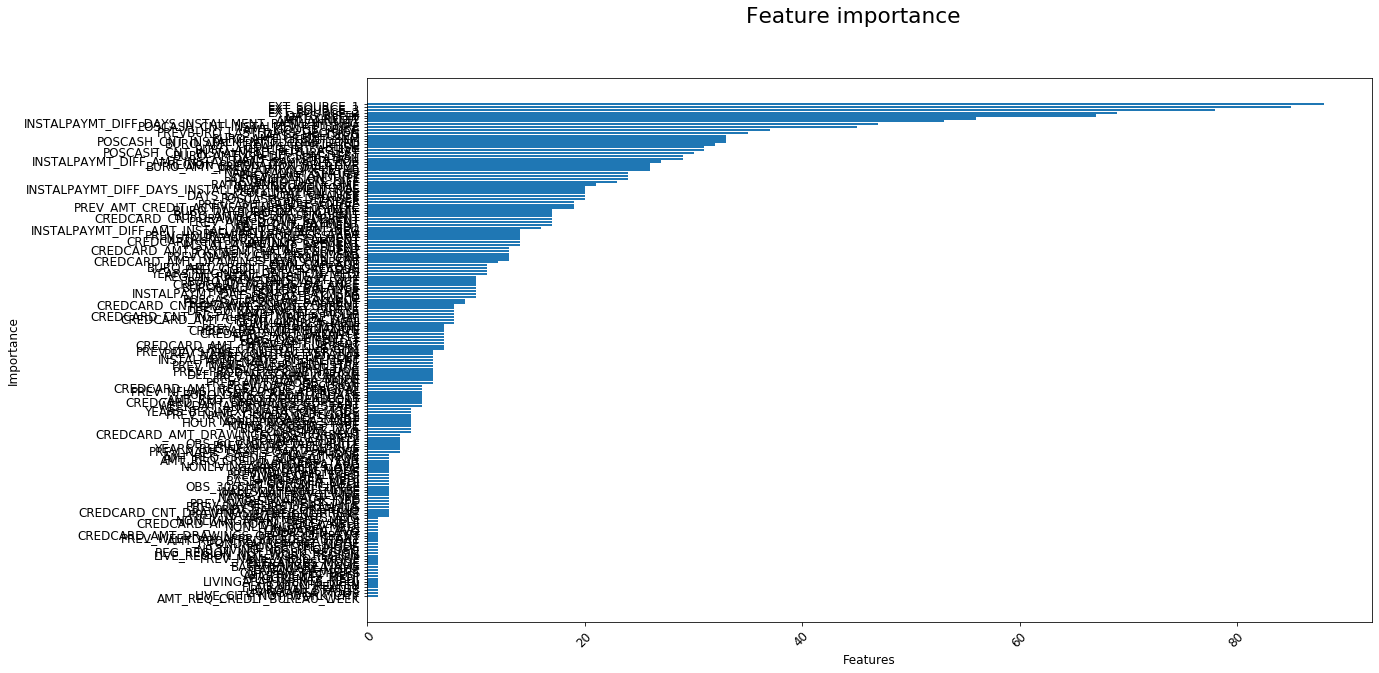
## Entrainement

Le pipeline est entraîné sur l'ensemble du jeu de données (300000 lignes)

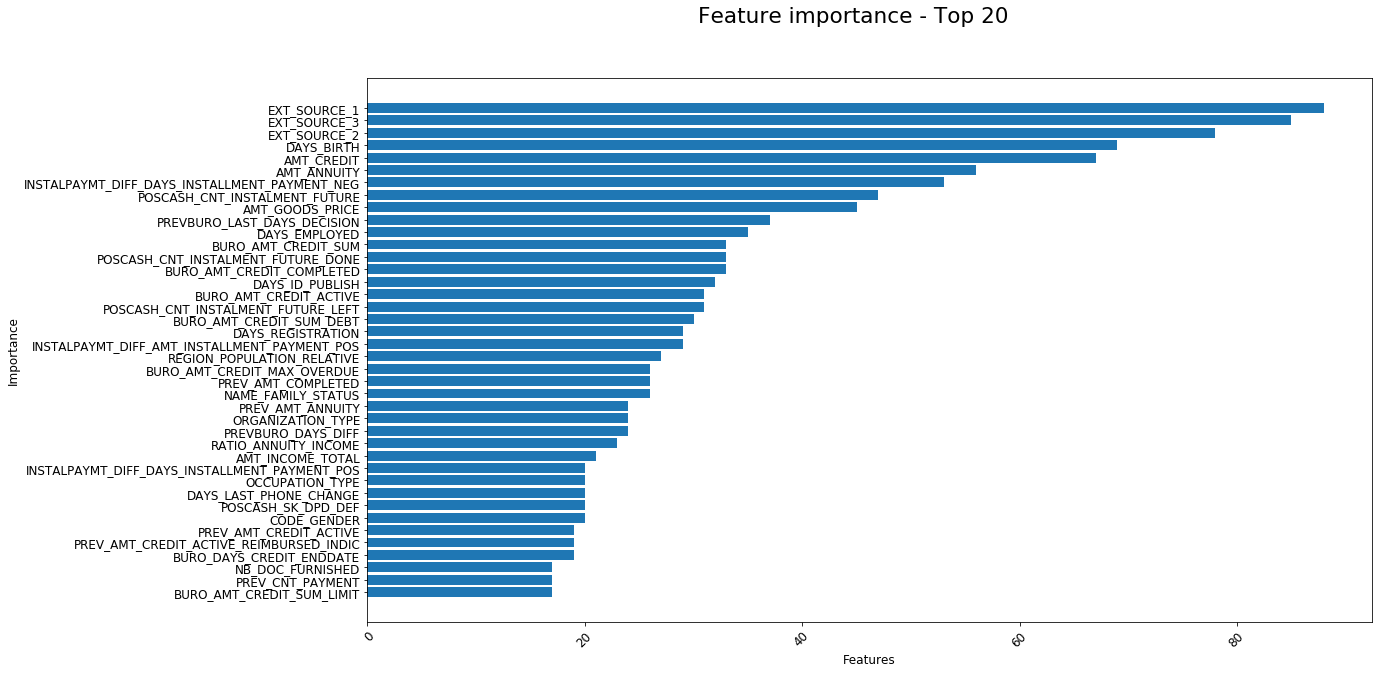
## Interprétabilité du modèle

Le modèle est inter prêté avec ses feature importances.

Voici l'ensemble des features avec leur importance:



Et voici les 20 features les plus importantes:



Pour chacune des features, on représente la distribution de la classe positive en fonction de la valeur de la variable (en effectuant des bins de 1000 pour les variables continues). Les graphiques suivants représentent les quatre variables les plus importantes, et la ligne verte représente le pourcentage réel de client ayant fait défaut. On peut ainsi, pour un client donné interpréter le scoring du modèle en fonction des features les plus importantes.

|  |  |
| --- | --- |
|  |  |
|  |  |

## Calibrage

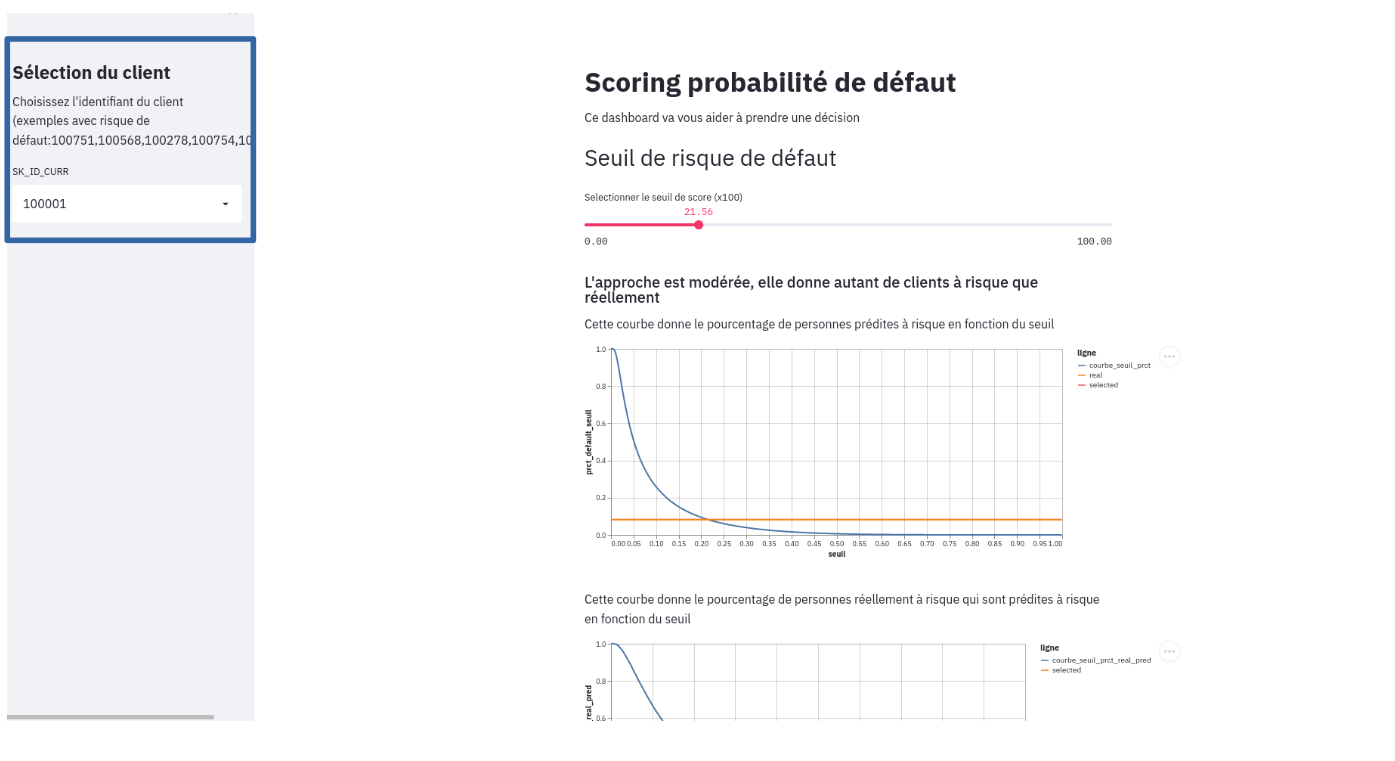
Le calibrage consiste, à fixer un seuil de score au-delà duquel un client sera considéré comme présentant un risque de défaut.

Ce seuil est fixé au niveau qui permet d'avoir 8% de clients en défaut, ce qui correspond au pourcentage réel de personne en défaut.

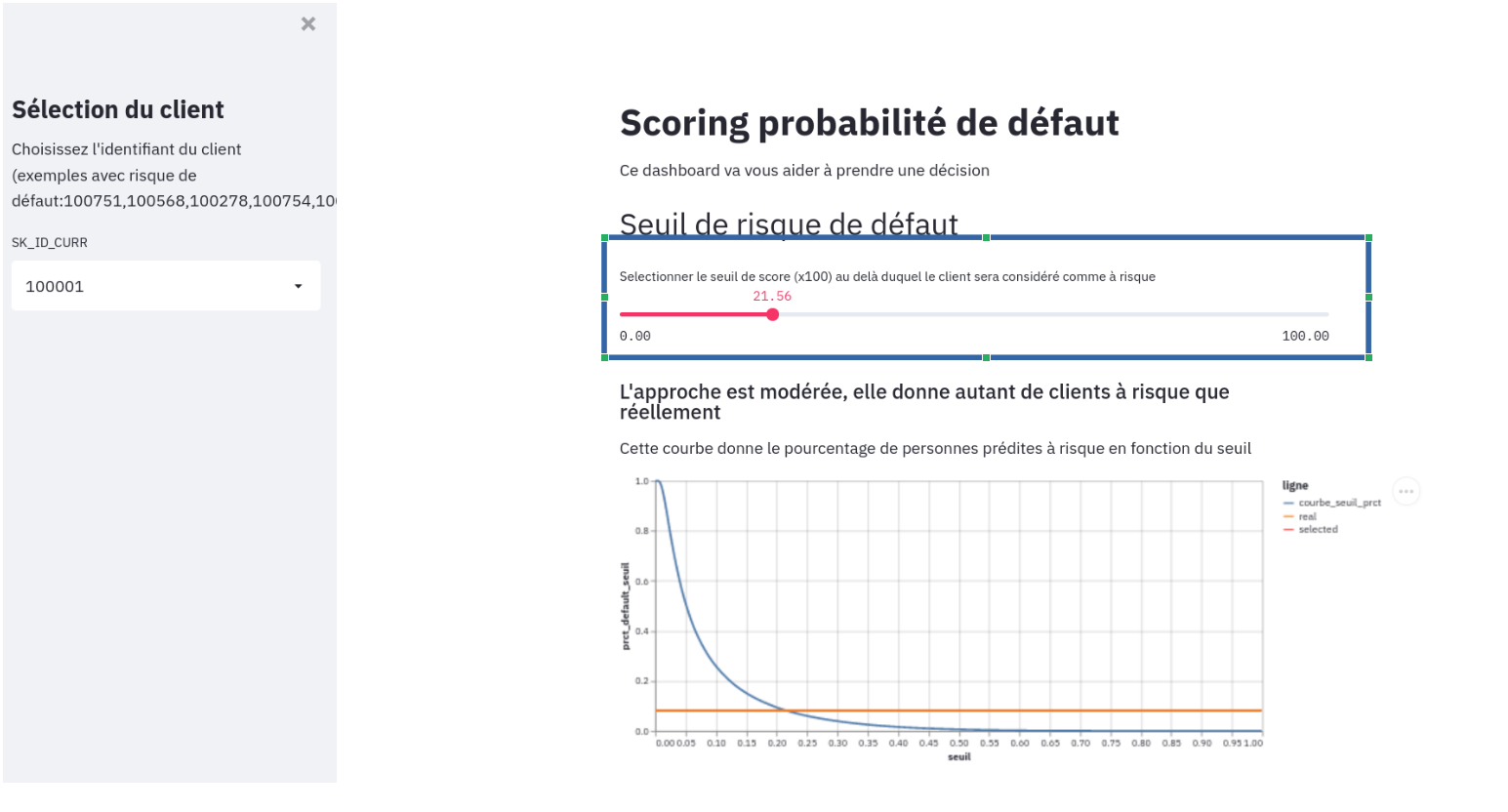
## Conception du dashboard

## Sélection du client

Le client est sélectionnable à gauche du dashboard avec son identifiant:



## 

****

## Sélection du seuil de score

Le seuil de score au-delà duquel le client sera considéré comme à risque est sélectionnable en utilisant un slider. Par défaut il est positionné au niveau qui donne 8% de clients à risque

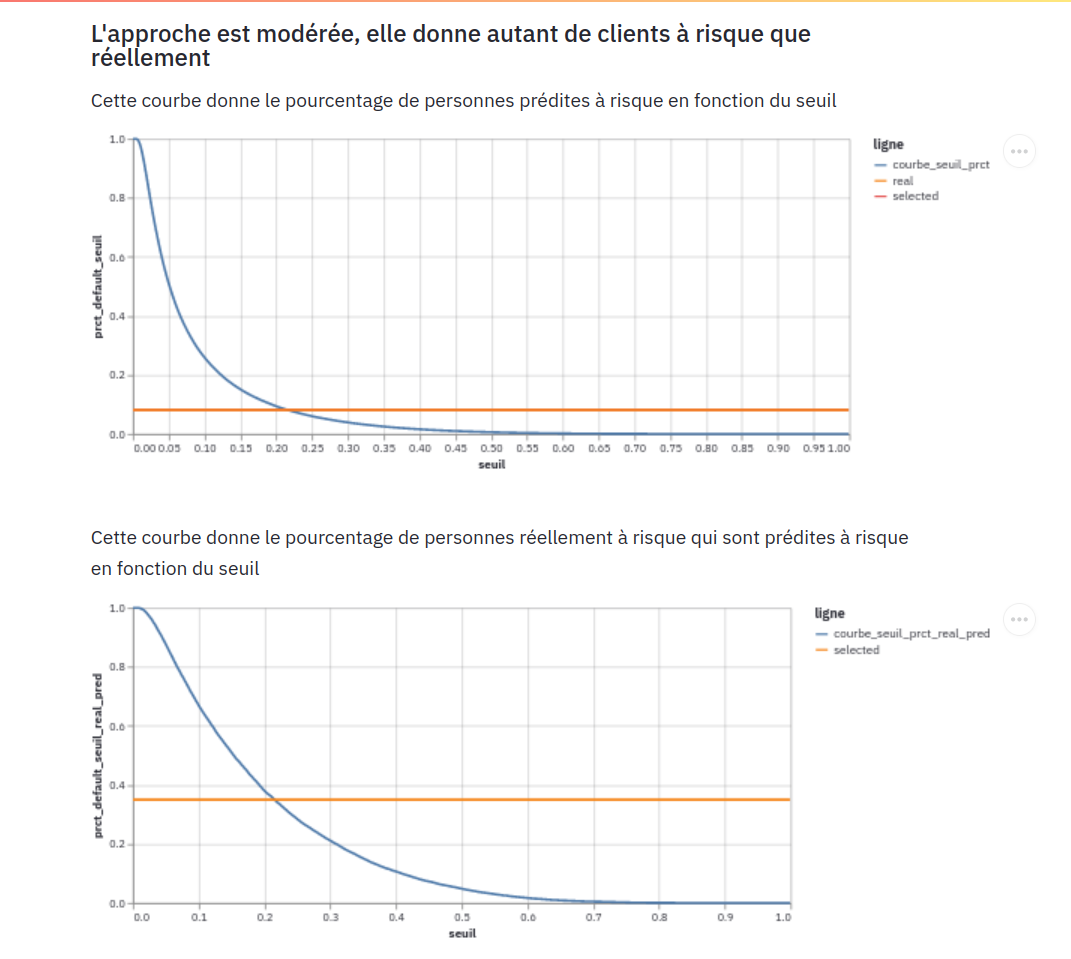
Selon le seuil sélectionné, l'approche est:

* agressive (beaucoup de clients sélectionnés)
* modérée (entre 6,5% et 9,5%)
* risquée pour la banque (peu de client à risque)

Deux graphiques permettent d'interpréter le seuil sélectionné :

* le premier représente en bleu les pourcentages de défaut prédits en fonction du seuil sélectionné
* le second représente en bleu les pourcentages des personnes réellement à risque qui sont prédites comme étant à risque

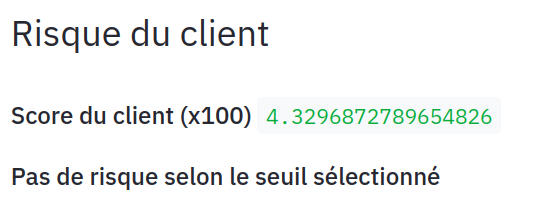
Dans les deux cas, la droite orange représente le pourcentage pour le seuil sélectionné et dans le premier le droite rouge représente le pourcentage réel de clients à risque.

****

## Risque du client

Ensuite, sont indiqués :

* le score du client
* son niveau de risque en fonction du seuil sélectionné



## Informations du client

Ensuite, sont représentées les informations du client, les variables sont ordonnées selon leur importance dans la détermination du score par le modèle



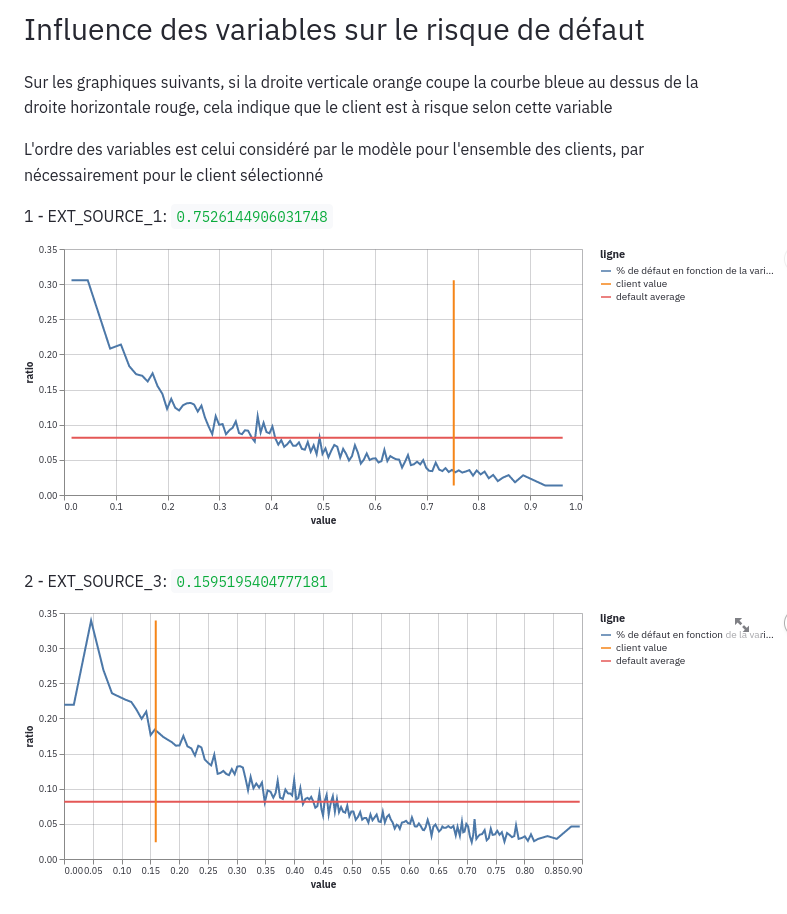
## Influence des variables

Enfin, pour chacune des 11 variables les plus importantes, on représente dans un graphique :

* le pourcentage de défaut en fonction de la valeur prise par la variable (bins 1000 pour les variables continues)
* une ligne horizontale rouge représentant le pourcentage réel de clients à risque (8%)
* une ligne verticale orange indiquant la valeur prise par la variable pour le client

Une variable indique un risque de défaut si la droite verticale orange coupe la courbe bleue au-dessus de la droite horizontale rouge.

Exemple des deux premières variables pour un client à risque :

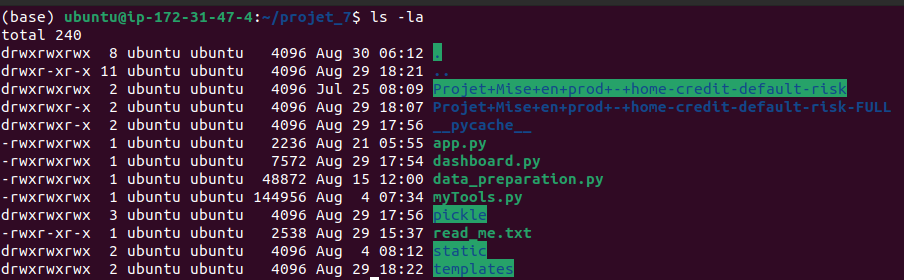


## Déploiement du modèle sur le web

## Déploiement sur AWS EC2

Le projet est déployé sur EC2 selont la structure suivante :

* app.py: modèle de prédiction
* dashboard.py: code du dashboard
* data\_preparation.py: code de préparation des données
* myTools.py: diverses fonctions
* pickle: dossier contenant divers objets dont:
  + le pipeline
  + les feature importances
  + les pourcentage de défaut en fonction du seuil
* templates: contient la template html qui ouvre la page web pour lancer le modèle
* static: vide
* Projet+Mise+en+prod+-+home-credit-default-risk: contient les fichiers csv avec les données de 200 client
* Projet+Mise+en+prod+-+home-credit-default-risk-FULL: contient les fichiers csv avec les données de 45000 client



Etapes à suivre pour l'installation sur EC2 :

# PREPARATION DE L'ENVIRONNEMENT LINUX

# Lancer une instance EC2 Linux Ubuntu

# Dans les groupes de sécurité, ouvrir les ports 8080 et 8501

# Attacher une adresse IP elastic

# Récupérer l'IP publique IPv4 elastic et la mettre dans le fichier projet\_7 (contenu dans projet\_7.zip) après server\_name, dans le fichier templates/index.html (dans projet\_7.zip) ainsi que dans les deux scp (voir transfert de fichiers)

# Exécuter pas à pas les commandes suivantes

# connection shh à l'instance

ssh -i "projet\_7.pem" ubuntu@ec2-13-37-98-86.eu-west-3.compute.amazonaws.com

sudo apt-get update

# TRANSFERT DES FICHIERS

sudo apt-get install unzip

# depuis le poste local, copier le fichier projet\_7.zip sur l'instance

scp -i ~/.ssh/projet\_7.pem projet\_7.zip ubuntu@ec2-13-37-98-86.eu-west-3.compute.amazonaws.com:~/projet\_7.zip

unzip projet\_7.zip

# depuis le poste local, copier le fichier de données complet (à télécharger sur kaggle) en ajoutant "-FULL' à la fin du nom du fichier

scp -i ~/.ssh/projet\_7.pem home-credit-default-risk-FULL.zip ubuntu@ec2-13-37-98-86.eu-west-3.compute.amazonaws.com:~/projet\_7/home-credit-default-risk-FULL.zip

# dans EC2

mv home-credit-default-risk-FULL.zip "Projet+Mise+en+prod+-+home-credit-default-risk-FULL.zip"

mkdir "Projet+Mise+en+prod+-+home-credit-default-risk-FULL"

unzip "Projet+Mise+en+prod+-+home-credit-default-risk-FULL.zip" -d ~/projet\_7

rm "Projet+Mise+en+prod+-+home-credit-default-risk-FULL.zip"

# dans EC2

rm projet\_7.unzip

# CONFIGURATION SERVEUR

sudo apt-get install nginx

sudo nano /etc/nginx/nginx.conf

# ajouter les lignes suivantes dans http

proxy\_read\_timeout 1000;

proxy\_connect\_timeout 1000;

proxy\_send\_timeout 1000;

sudo apt-get install gunicorn3

sudo mv projet\_7 /etc/nginx/sites-enabled/projet\_7

# INSTALLATIONS PYTHON

sudo apt-get install python3-pip

pip3 install flask

pip3 install Cython

pip3 install lightgbm

pip3 install seaborn

pip3 install pandas

pip3 install matplotlib

pip3 install numpy

pip3 install scipy

pip3 install statsmodels

pip3 install imblearn

pip3 install sklearn

pip3 install plotly

pip3 install altair

wget https://repo.anaconda.com/archive/Anaconda3-2021.05-Linux-x86\_64.sh

sh Anaconda3-2021.05-Linux-x86\_64.sh

rm Anaconda3-2021.05-Linux-x86\_64.sh

# se déconnecter puis se reconnecter

pip3 install streamlit

# LANCEMENT

sudo service nginx restart

cd ~/projet\_7

# Ouvrir la page html pour lancer le modèle

screen -S model

gunicorn3 --timeout 1000 app:app

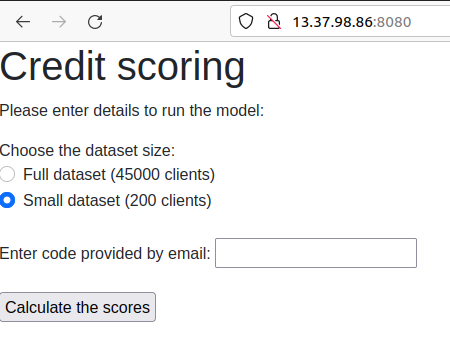
# Ouvrir le dashboard

screen -S dashboard

streamlit run dashboard.py

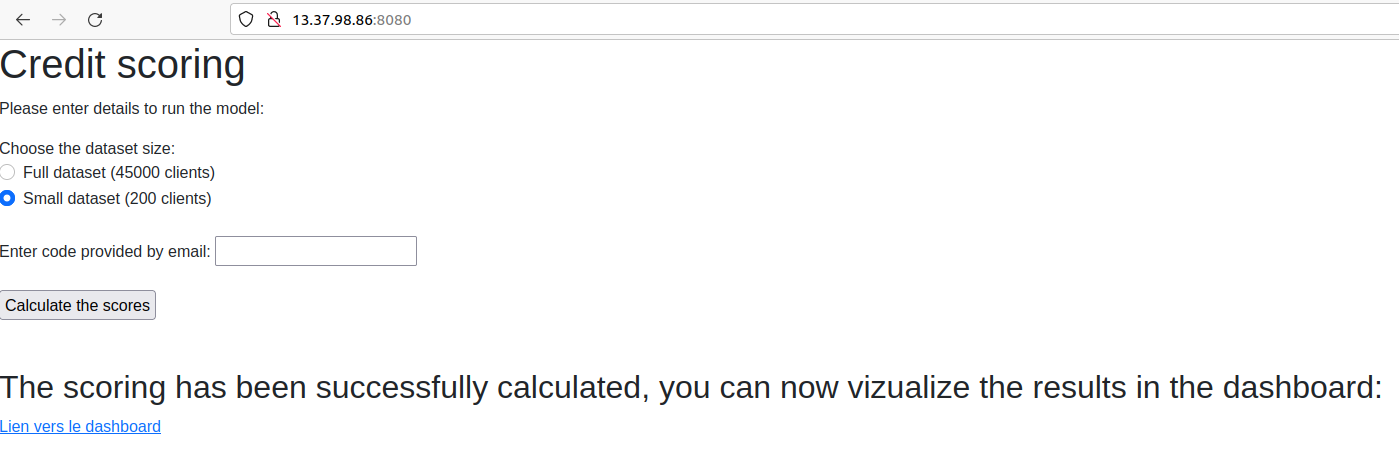
## Lancement du modèle à distance

L'URL [http://13.37.98.86:8080/](http://35.181.74.104:8080/) ouvre une page qui permet de lancer le modèle sur les 45000 clients ou sur un sous-ensemble de 200 clients:



Un code doit être entré pour pouvoir lancer le modèle.

Lorsque le calcul s'est déroulé sans erreur, un message indique qu'on peut lancer le dashboard avec un lien vers l'URL du dashboard(http://13.37.98.86:8501/):

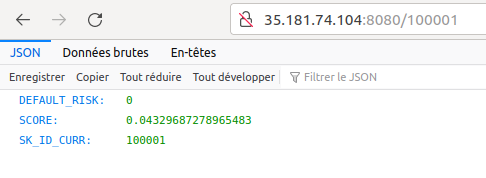


## Lancement du dashboard à distance

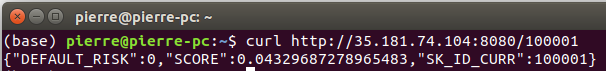
L'URL <http://13.37.98.86:8501/> ouvre la page web du dashboard.

## API

Enfin, une API permet de récupérer le score et le risque de défaut d'un client donné, on l'appelle en utilisant la même URL que pour le modèle, mais en ajoutant un "/" suivi du code du client :



On peut appeler cette API avec une commande CURL :



## Limites et améliorations possibles

## Limites

La principale limite du modèle est inhérente aux problèmes d'imbalanced classification: il y aura toujours un nombre non négligeable de clients à risque non identifié comme tel et il faut diminuer le seuil de scoring pour en capturer le plus possible, avec l'inconvénient de classer un trop grand nombre de client à risque

## Améliorations possibles

Améliorations possibles au niveau du dashboard:

* Mettre des graphiques dynamiques (qui indiquent des valeurs quand on déplace le curseur)
* Lissage des courbes d'interprétation du modèle
* Donner la possibilité de sélectionner une ou plusieurs variables pour interpréter le modèle
* Inclure les variables catégorielles dans les variables sélectionnables pour interpréter le modèle (aucune ne fait partie des 11 plus importantes variables)
* Donner le choix de faire le scoring avec ou sans les variables externes

Améliorations possibles au niveau de l’API :

* Donner la possibilité via l'API de sélectionner un groupe de client selon divers critères

Améliorations possibles au niveau du modèle :

* Tester un réseau de neurones
* Tester les options avancées de XGBoost (telle que l'early stoping)