Problèmatique

La société financière "Pret à dépénser", est une société qui est chargée d'accorder des crédits aux personnes qui ont un faible risque selon leur historique de prêt. Pour ce faire, l'entreprise veut développer un modèle de notation de la probabilité de paiement du client pour décider d'accorder ou non le prêt.

Données

Les données sur les antécédents de crédit sont fournies par Home Credit, une entité qui se consacre aux prêts aux personnes non bancarisées. Ces données sont regroupées en 7 types d'informations différentes.

1) "app train/test" : Il s'agit des principales informations relatives à la formation et aux tests sur chaque demande de crédit faite par un client. Chaque ligne correspond à un client. Chaque client a son numéro d'identification "SK_ID_CURR". Dans les données “app_train " que nous appelons "Data" dans notre cahier, il y a une colonne "Target" qui a un système de classification binaire qui indique "0" pour les prêts payés avec succès et "1" pour les prêts non payés.

2) Bureau : Ces données résument l'historique des prêts antérieurs d'autres institutions financières. Chaque prêt a sa propre ligne, ce qui signifie que chaque ligne de "app_train" peut être associée à plusieurs prêts précédents.

3) Bureau_balance : C'est l'historique mensuel des prêts précédents, ceux-ci sont stockés sur plusieurs lignes dont la quantité dépend du nombre de mois correspondant à chaque prêt.

4) prev_app : Il s'agit des demandes précédentes faites par les clients du "Home Credit". Chaque client peut avoir plusieurs prêts antérieurs représentés par plusieurs lignes identifiées comme "SK_ID_PREV".

5) POS_CASH_BALANCE : Informations mensuelles sur les prêts en espèces que les clients ont contractés avec Home Credit. Chaque ligne correspond à un mois de prêt, de sorte que chaque prêt peut avoir plusieurs lignes.

6) Credil_Card_Balance : Informations mensuelles sur les cartes de crédit que les clients ont déjà eues avec Home Credit. Chaque ligne correspond à un mois de solde de carte de crédit. Chaque crédit peut avoir plusieurs lignes.

7) Installements_payment : Historique des paiements des prêts précédents avec le Crédit immobilier. Il y a une ligne soit pour le paiement effectué, soit pour le non-paiement.

Analyse exploratoire et " Features Engineering "

Dans le cadre de l'analyse exploratoire, chacune des sept informations décrites ci-dessus a été travaillée une par une et leurs observations et transformations seront décrites dans ce qui suit:

Dans le fichier "app_train" ou "data", on observe la variable "Target", qui est une variable de classes de type binaire et de classe déséquilibrée (voir figure 1). Il y a beaucoup plus de prêts portant le label "0" (prêts payés avec succès) que de prêts portant le label "1" (prêts non payés). Nous expliquerons plus tard ce qu'il faut faire dans ces cas-là lorsque nous parlerons de modélisation.

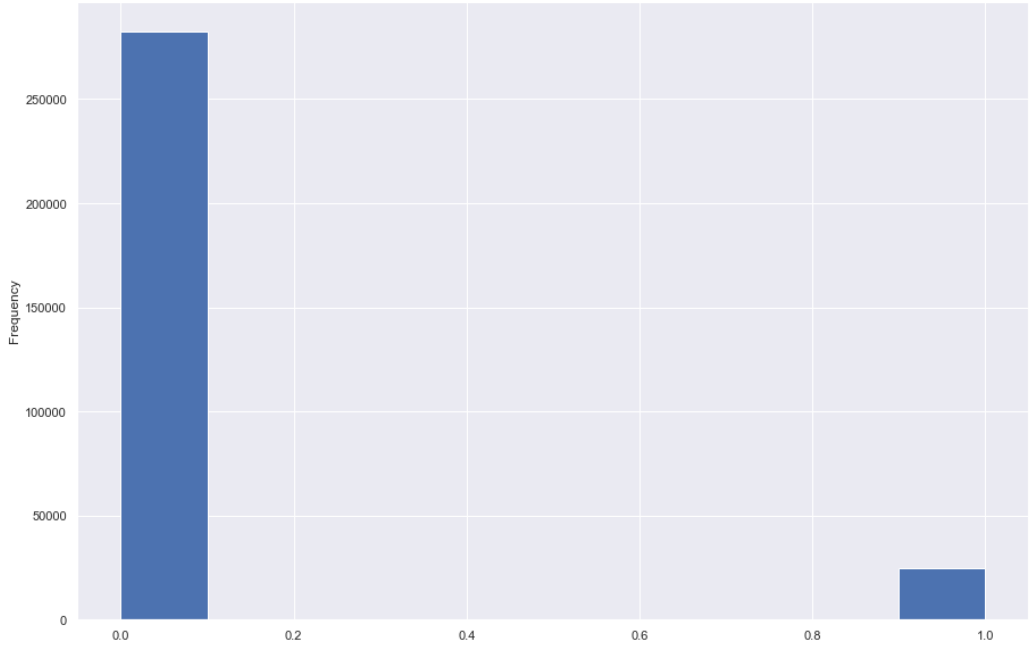

Figure 1. Model desequilibré

En examinant certaines variables temporelles telles que "DAYS_BIRTH", c'est-à-dire une variable qui représente en jours l'âge du client au moment du prêt. Nous remarquons qu'elle est représentée avec une magnitude négative, nous divisons donc cette variable par -365 pour pouvoir la représenter en années positives. Une corrélation PEARSON a ensuite été effectuée pour voir s'il existe une relation entre l'âge du client et la valeur de l'étiquette "Target".

Maintenant, en observant la variable "Days_Employed" où la valeur en jours d'emploi de chaque client est observée, nous constatons à travers l'histogramme suivant (voir figure 2) qu'il y a beaucoup de valeurs anormales qui montrent une valeur constante égale à 365243, qui n'est pas une valeur cohérente ; il a donc fallu les éliminer en obtenant ce nouvel histogramme (voir figure 3).

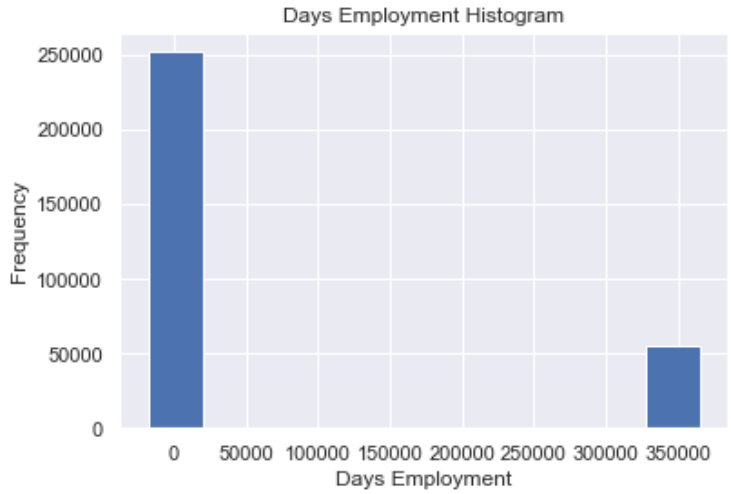

Figure 2. Days Employement Histogram

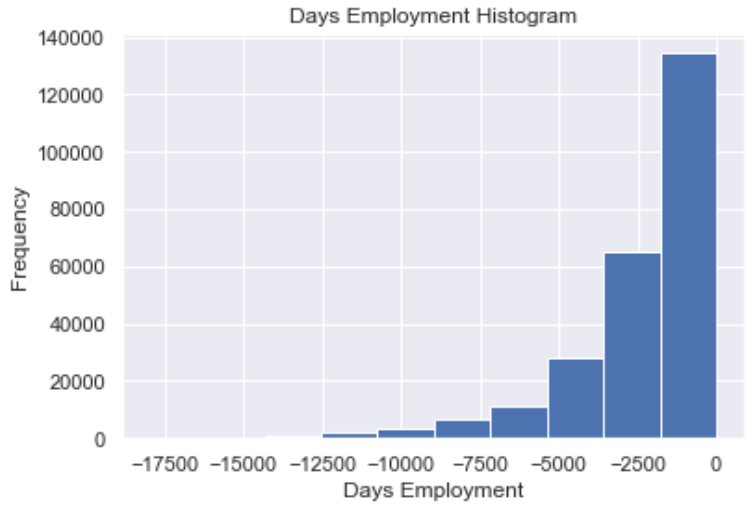

Figure 3. Nouvel Days Employement Histogram

Ensuite, les corrélations de toutes les variables par rapport à "Target" (voir figure 4) sont faites pour explorer les variables qui pourraient avoir plus d'impact sur la modélisation. On observe que le coefficient de corrélation positif le plus élevé à "DAYS_BIRTH", ce qui peut indiquer qu'à mesure que le client vieillit, le pourcentage de non-paiement devient plus faible (en regardant un graphique par tranche d'âge, voir figure 5).

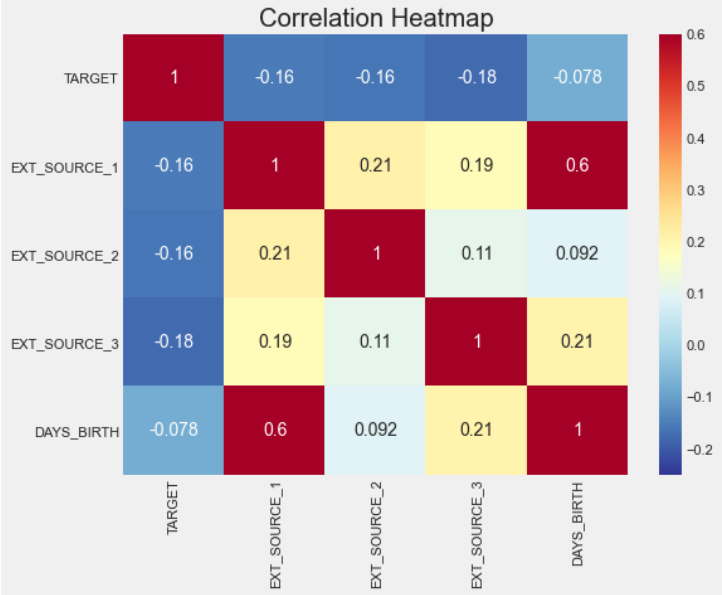

Figure 4. Corrélation Pearson

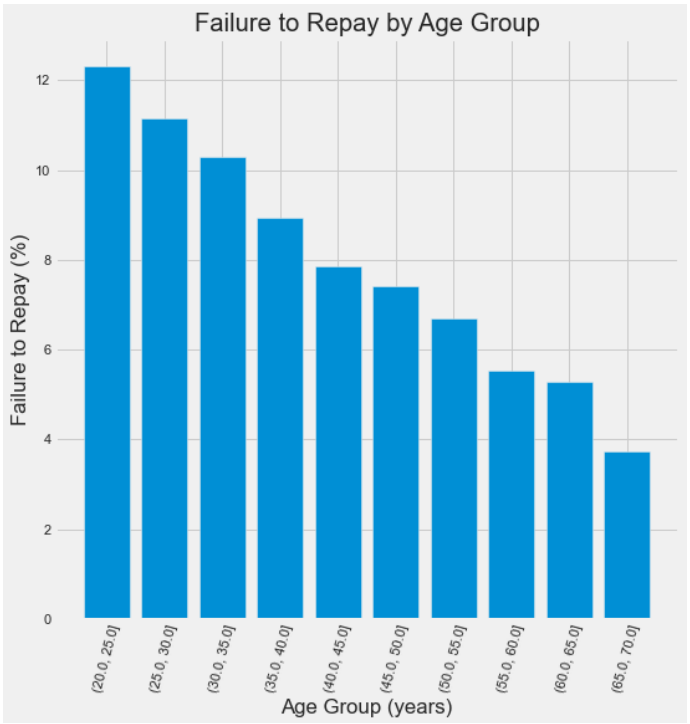

Figure 5. Age vs Défaut Paiement

Cet effet peut être visualisé par "KDE" (voir figure 6), en observant une distribution des densités pour "1" (ligne rouge) et "0" (ligne bleue). De même, nous constatons que les jeunes clients sont ceux qui ont tendance à avoir un peu moins de fiabilité lorsqu'il s'agit de rembourser un crédit.

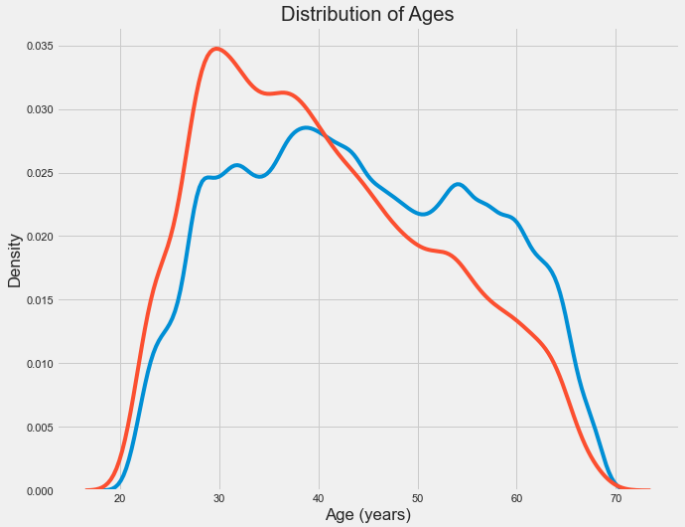

Figure 6. Distribution de Ages.

Les variables "EXT_SOURCE" présentent le plus grand nombre de corrélations négatives (figure 4). Il existe trois variables corrélées négativement, ce qui indique que lorsque leur valeur augmente, la probabilité que le client rembourse le prêt augmente également. Bien que les valeurs de corrélation n'aient pas un impact aussi fort, ces variables pourraient avoir une grande influence lors de la modélisation.

Pour mieux comprendre l'impact des variables "EXT_SOURCE", le graphique en paires suivant (figure 7) compare ces variables avec "YEARS_BIRTH" où l'on observe une très forte relation avec l'âge. Il semble que les variables "EXT_SOURCE" soient des valeurs de score normalisées où l'un des critères de scoring pourrait être l'âge du client. La tendance est à l'augmentation avec l'âge et à l'affichage de valeurs "1" à des âges plus bas. Cela signifie que plus l'âge est jeune et plus le score EXT_SOURCE est bas, moins le client a de chances de rembourser le prêt.

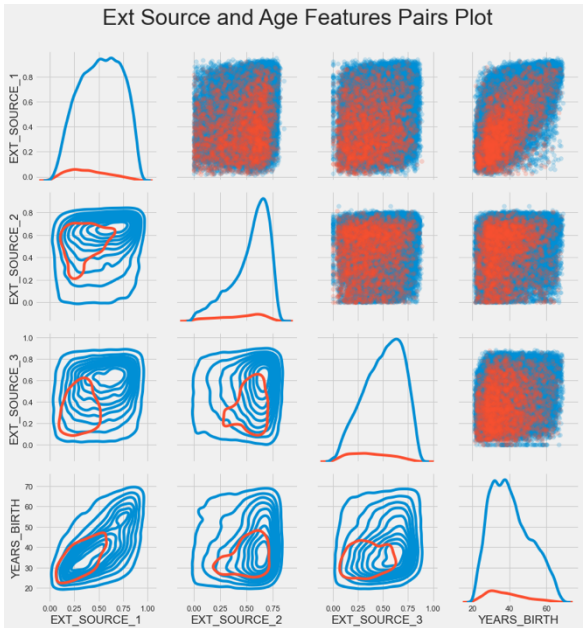

Figure 7. Pair Plot, Age vs EXT_SOURCE

Nous allons maintenant résumer les transformations à effectuer sur les autres informations :

1) Bureau : les données numériques et catégorielles ont été traitées séparément. Les variables numériques ont été transformées en variables statistiques descriptives (moyenne, max, min, somme). Les variables catégorielles sont transformées en variables numériques en créant des variables qui stockent les comptes par catégorie. Une fois les transformations effectuées, les résultats sont regroupés par client "SK_ID_CURR". Une fois cela fait, les corrélations entre les variables sont observées afin d'éviter le "overfitting". Par conséquent, les variables qui dépassent une valeur de corrélation supérieure à 0,8 sont éliminées. Une fois cette procédure terminée, les données sont prêtes à être ajoutées à notre matrice principale de "data".

2) Bureau_Balance : comme ici nous avons des informations mensuelles avant de regrouper les informations par client, celles-ci sont regroupées par prêt précédent "SK_ID_BUREAU". La procédure de traitement est la même, les variables numériques et catégorielles sont transformées comme dans "Bureau", l'union des deux transformations est faite par prêt et enfin regroupées par client. Le filtre de corrélation est refait et ils sont ajoutés à la matrice principale des "data".

3) Pour le calcul des variables (moyenne, max, min, somme) et la transformation des variables catégorielles, deux fonctions "agg_numeric" et "counts_categorical" ont été créées. Pour les autres informations telles que "POSH_CASH_BALANCE", "credit_card_balance" et "Installements_payments", c'est exactement la même procédure qui s'applique. Selon la façon dont les données sont réparties, elles sont d'abord regroupées par prêts, puis par clients avant d'être fusionnées avec la matrice principale.

Valeurs manquantes

Une fois que la matrice avec toutes les informations fusionnées est obtenue, on obtient un tableau de taille (307511X623). Le tableau ci-dessous présente les variables dont 98 % des valeurs sont manquantes. À l'aide de la fonction "remote_missing_columns", les variables comportant plus de 75 % de valeurs manquantes sont éliminées, ce qui permet d'obtenir un tableau de (307511 X 606).

Encodage des variables catégorielles

Enfin, nous devons traiter les variables catégorielles qui proviennent de l'information initiale (voir figure 8) "app_train". Dans ce cas, nous avons la variable "TARGET" qui ne comporte que deux types de catégories pour lesquelles nous pouvons utiliser le "label Encoding", conservant ainsi sa forme initiale.Pour les autres variables catégorielles (avec plus de 2 catégories), "one hoy encoding" est appliqué, donnant une variable ou une colonne pour chaque observation dans chaque catégorie. Il en résulte une matrice plus large (307511 X727). Il semble que la matrice soit assez grande et qu'il soit nécessaire d'utiliser une technique de réduction dimensionnelle avant la modélisation.

NAME_CONTRACT_TYPE	2
CODE_GENDER	3
FLAG_OWN_CAR	2
FLAG_OWN_REALTY	2
NAME_TYPE_SUITE	7
NAME_INCOME_TYPE	8
NAME_EDUCATION_TYPE	5
NAME_FAMILY_STATUS	6
NAME_HOUSING_TYPE	6
OCCUPATION_TYPE	18
WEEKDAY_APPR_PROCESS_START	7
ORGANIZATION_TYPE	58
FONDKAPREMONT_MODE	4
HOUSETYPE_MODE	3
WALLSMATERIAL_MODE	7
EMERGENCYSTATE_MODE	2

Figure 8. Variables Categorielles

Modélisation

Nous utiliserons la matrice "data_model" avec toutes les données agrégées qui contiennent la variable "Target". Il est nécessaire de procéder à une réduction dimensionnelle. Dans ce cas, nous avons utilisé la technique de l'ACP pour obtenir environ 89 % de la variance (voir figure 9) avec seulement 100 composantes principales, ce qui a donné une matrice appelée "Data_model_reduced" de (307511 X 100).

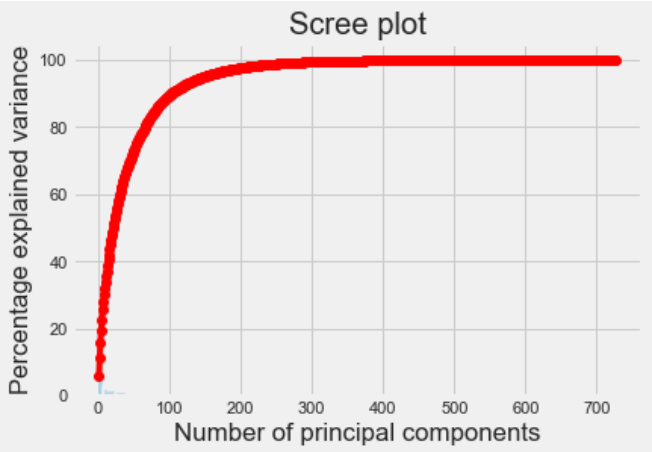

Figure 9. PCA MODEL_REDUCED

Avant de procéder à la réduction, une imputation automatique des valeurs manquantes à partir de la médiane a été effectuée et les valeurs ont été normalisées entre 0 et 1.

Avant de commencer la modélisation, un test a été effectué avec la technique "SMOTE" ("Synthetic Minority Oversampling Technique"). Cette technique consiste essentiellement à créer des échantillons artificiels d'une classe pour compenser le manque d'équilibre entre les classes. Dans ce cas, la variable "Target" a "282686" de valeur "0" et seulement "24835" de valeur "1" qui ne représente que 8% (figure 1). Ces échantillons synthétiques sont créés de façon aléatoire entre des espaces d'échantillons voisins, ce qui compense le déséquilibre entre

les classes. Malheureusement, l'ordinateur disponible (8 Go de RAM, 1600 MHz) est très lent à exécuter cette technique pour un modèle comportant plus de 300 000 échantillons. Il en sera tenu compte dans les éventuelles améliorations à apporter à la modélisation.

Il convient également de noter que pour la modélisation, nous n'utiliserons que la matrice "data_model_reduced", les informations "app_test" ne seront pas intégrées, puisque notre validation des données sera effectuée à partir du X_test créé à partir d'un "split". C'est-à-dire que l'implémentation sera effectuée sur un X_train, X_test, Y_train et Y_test, tous issus de la matrice principale.

Fonction de coût

Avant de mettre en œuvre les algorithmes de modélisation pour la prédiction des scores de probabilité, nous devons introduire le concept de fonction de coût. Nous savons déjà que nous travaillons avec un modèle déséquilibré. Nous devons donc tenir compte de la pénalité ou du coût qu'impliquerait une mauvaise prédiction.

Cette fonction de coût est construite sur la base de la mesure ou de l'estimation de la "Fbeta_Mesure" qui est calculée à partir de la "Précision" et du "Rappel". Précision" étant la variable qui calcule le pourcentage de prédictions correctes pour la classe positive et "Rappel" étant la variable qui calcule le pourcentage de prédictions correctes pour la classe positive par rapport à toutes les possibilités positives existantes.

La mesure F est la moyenne harmonique de la "Précision" et du "Rappel", qui permet de donner du poids à chacune des variables et de décrire la performance du modèle. Fbeta est en fait une paramétrisation de F_mesure où B=1. Une valeur de bêta inférieure à 1 donnerait plus de poids à la précision qu'au rappel. Et un bêta supérieur à 1 donnerait moins de poids à la précision et plus de poids au rappel.

Dans notre modèle, nous avons deux scénarios possibles :

a) Un nombre éventuellement important de FN (faux negative), c'est-à dire un excès de clients considérés comme non éligibles avec defaut de payment (excès de "1"), c'est-à-dire moins de poids à la précision et plus au rappel (bêta superieur à 1).

b) Un éventuel nombre élevé de FP, c'est-à-dire un excès de clients éligibles au paiement ou sans defaut de payment (excès de "0"), c'est-à-dire plus de poids pour la précision et moins de poids pour le rappel (bêta inferieur à 1).

Ces quantités peuvent être observées dans la matrice de confusion une fois que l'algorithme d'"apprentissage automatique" "Régression logistique" a été mis en œuvre.

En analysant les deux cas, il s'agit d'un modèle de classification binaire, il est préférable de minimiser les Faux Positifs car nous voulons minimiser le nombre de personnes qui ne seront pas en mesure de payer le prêt (excès de Faux Positifs). Par conséquent, pour minimiser le FP et maximiser la précision et minimiser le rappel nous utiliserions un bêta supérieur à 1. Dans

cette modélisation, nous avons utilisé un bêta=2,5 en supposant que nous fixions une augmentation de près de deux fois plus de faux négatifs que de faux positifs.

Mise en œuvre de la régression logistique

Une fois notre fonction de coût définie, et après avoir fait notre "séparation" de la matrice "Data_model_reduced" et du vecteur "Target", nous mettons en œuvre l'algorithme "RL" qui effectue une "GridSearch" pour l'optimisation de l'hyperparamètre C qui varie entre 0 et 1 avec CV=5 et l'optimisation de la fonction de coût. Voir les résultats dans le diagramme de confusion (figure 10 et 11). Les données ont été testées avec la matrice X_test et à partir de la méthode "predice_proba() une matrice de probabilité des scores est générée où la probabilité de paiement ou de non-paiement est indiquée. La figure ci-dessous indique par exemple les probabilités que le prêt ne soit pas remboursé.

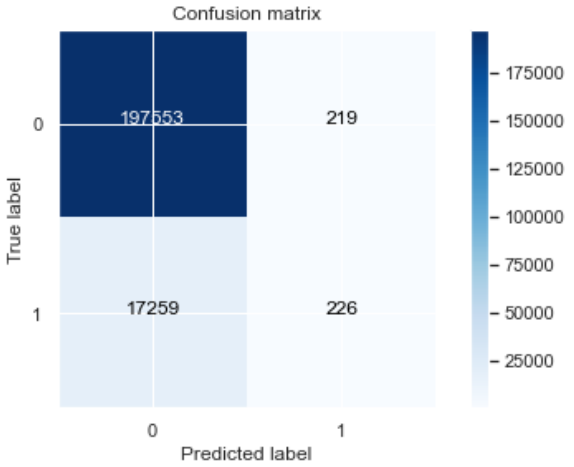

Figure 10. Matrice Confusion X_train LR, Recall=1,29

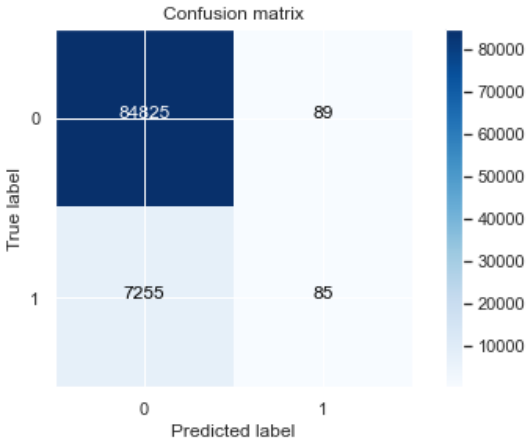

Figure 11. Matrice Confusion X_test LR Recall= 1,15

	SK_ID_CURR	TARGET
42962	149741	0.259151
227307	363290	0.069791
290035	436006	0.040062
239833	377703	0.080450
76427	188624	0.189557

Figure 12. Probabilité de defaut de Paiement X_test RL

Pour interpréter plus clairement les résultats du modèle et faire un diagnostic sur le comportement de notre classificateur binaire, une évaluation a été faite à partir d'un graphique "ROC" (figure 13). La fonction de coût a été travaillée de manière à maximiser la précision et à minimiser le "rappel" où un score AUC = 0,752 a été obtenu, ce qui est un assez bon score de classement.

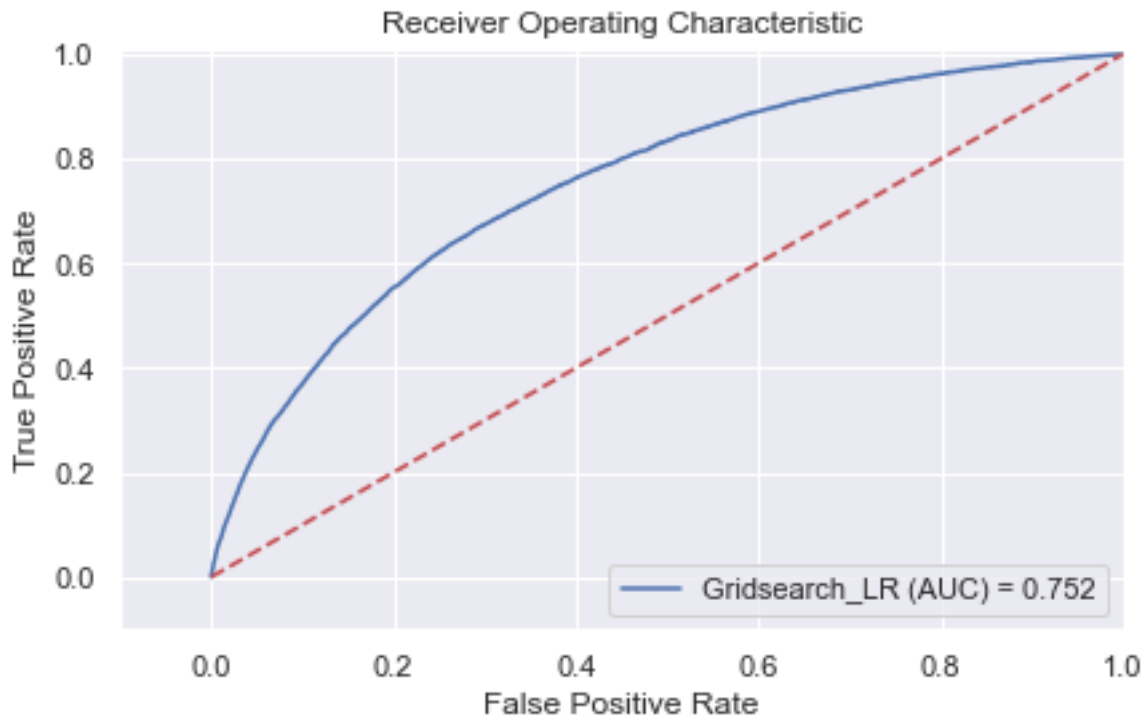

Figure 13 ROC Regression Logistique

Mise en œuvre de l'algorithme "Random Forest"

Nous avons utilisé l'option "random forest classifier" et une "GridSearch" où nous avons optimisé les hyperparamètres entre "n_estimateurs" [10, 100, 200] et "max_depth" [2, 10] avec un cv=5 optimisant la fonction de coût.

Comme pour la "Régression logistique", nous avons une matrice de confusion (figures 14 et 15) et une probabilité de paiement et de non-paiement (figure 16).

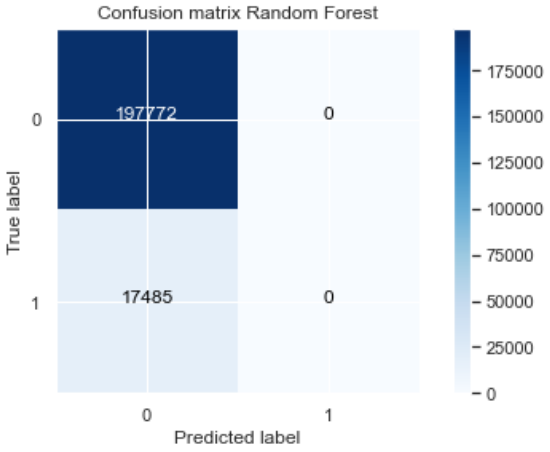

Figure 15. Matrice de Confusion X_train RF Recall=0

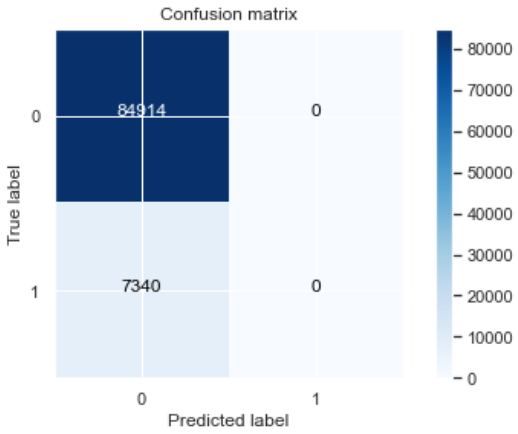

Figure 14 Matrice de Confusion X_test RF Recall=0

	SK_ID_CURR	TARGET
42962	149741	0.105472
227307	363290	0.078783
290035	436006	0.065291
239833	377703	0.079393
76427	188624	0.079742

Figure 16 Probabilité de defaut de paiement X_test RF

De la même manière, nous obtenons une matrice de confusion et une courbe ROC où nous avons obtenu une AUC=0,670 (voir figures).

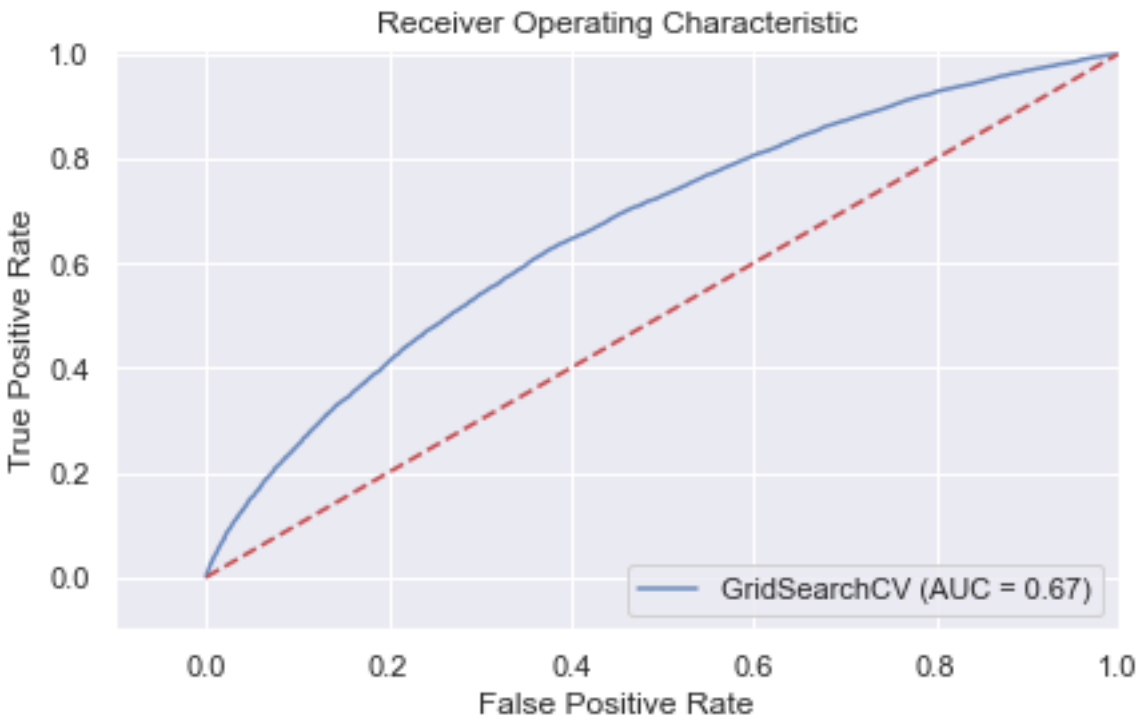

Figure 17. ROC RF

Si nous comparons les deux modèles utilisés (figure 18), nous pouvons constater qu'un meilleur score a été obtenu dans la "LogisticRegession" par rapport à "Random Forest". Il convient de noter que dans la forêt aléatoire, un nombre limité de paramètres ont été utilisés dans la "GridSearch". Une modélisation plus robuste est peut-être nécessaire, mais en termes de temps, la "Régression logistique" est plus rapide.

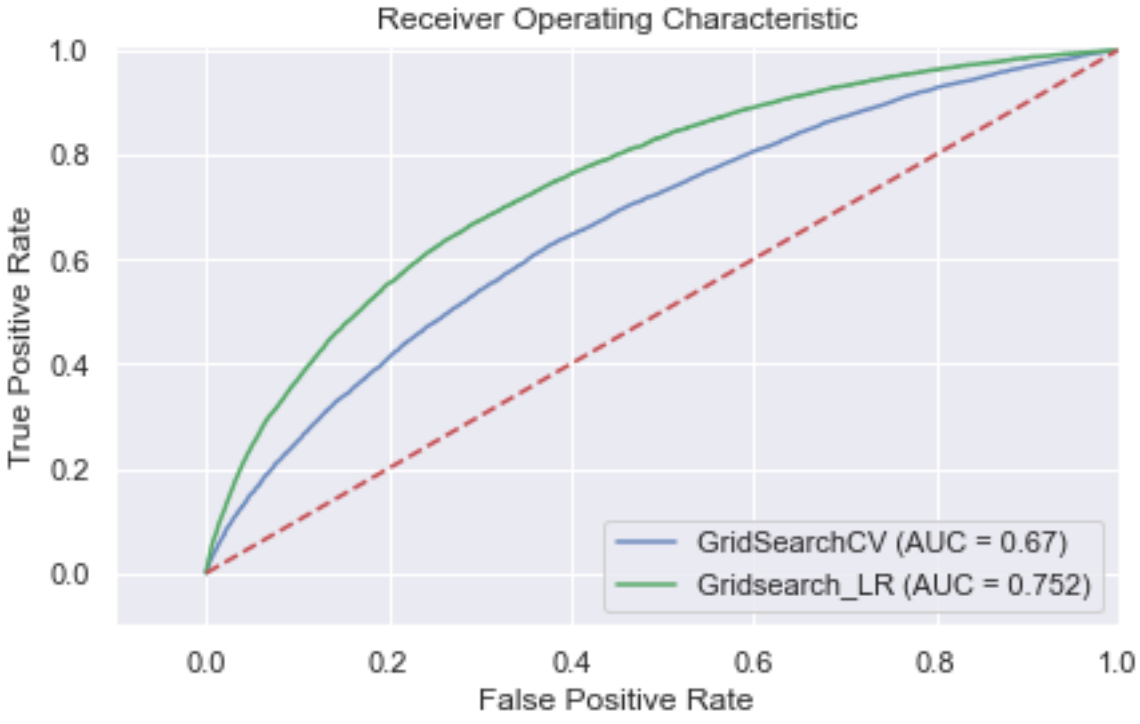

Figure 18. ROC LR VS RF

Bien que nous ayons travaillé avec une matrice réduite où les variables qui ont le plus d'impact sur notre modèle ne peuvent être visualisées. En effectuant une "analyse de l'importance des caractéristiques" avec les composantes de l'ACP, nous avons remarqué que seules quelques variables ont un grand impact sur le modèle et que les variances sont presque bien réparties entre toutes les variables (figure 19).

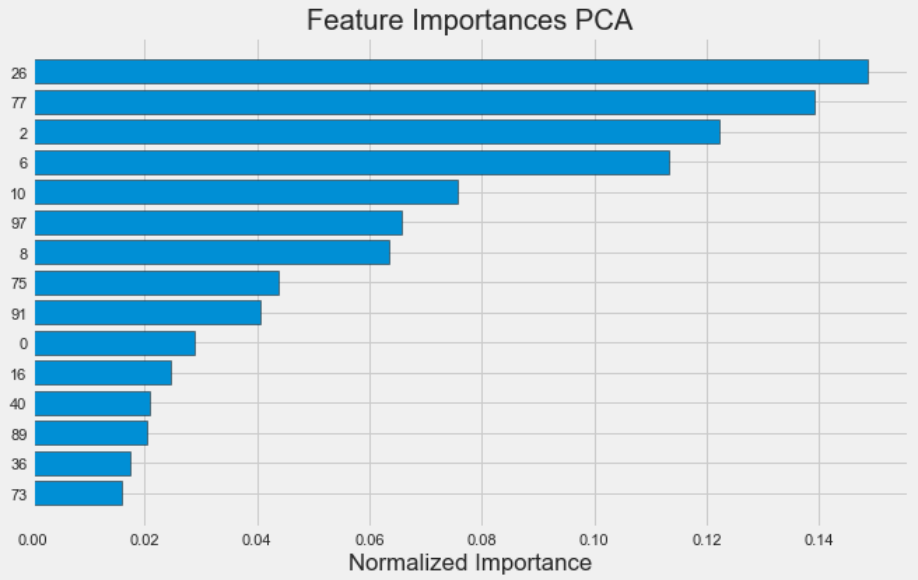

Figure 19. Feature Importances, Grand Importance CP 26, 77,2, 6 et 10)

Comparaison de la figure précédente avec les "Importances des caractéristiques" d'un modèle sans réduction dimensionnelle (figure 20). Nous pouvons voir que les variables EXT_SOURCE et DAYS_BIRTH sont celles qui ont peut-être le plus d'impact sur le modèle.

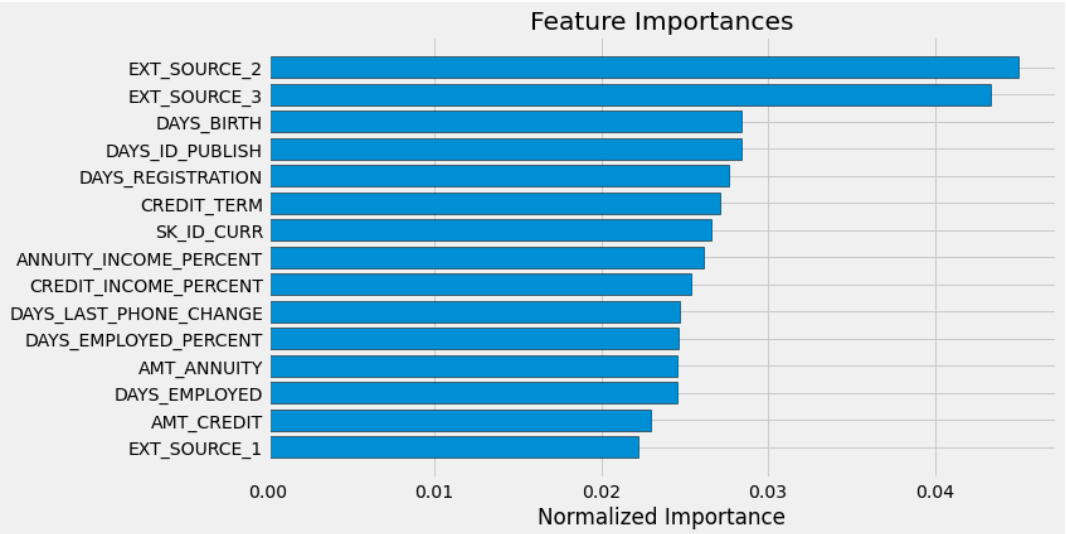

Figure 20. Modelization Random Forest Fait juste avec APP_train Source: Kaggle Start a gentle introduction

Améliorations et désavantages

Comme nous l'avons vu plus haut, ce modèle présente un problème de déséquilibre et il a donc fallu utiliser un algorithme de "Smote" qui pouvait créer des échantillons artificiels qui équilibreraient la quantité de "1" par rapport à "0", ce qui explique probablement pourquoi le modèle a finalement montré une valeur aussi élevée de faux positifs. Malheureusement, en raison des caractéristiques de l'ordinateur, cet algorithme n'a pas pu être mis en œuvre.

Un autre inconvénient de la "Régression logistique" est qu'elle peut présenter des problèmes d'interprétation lorsqu'il existe des "caractéristiques" qui séparent presque parfaitement les deux classes en raison du type de fonction probabiliste qui régit cet algorithme, ce qui pourrait poser des problèmes de convergence.

C'est pourquoi nous voulions mettre en œuvre un modèle "RandomForest", mais en raison de la taille du modèle, nous ne pouvions pas mettre en œuvre un ensemble d'hyperparamètres beaucoup plus robuste qui nous permettrait d'obtenir de meilleures valeurs de CUA. En effet, l'optimisation de ce type de grande modélisation nécessite des ordinateurs plus puissants.