École Centrale Casablanca

CODING WEEK

SUJET 4 : DÉFAUT DES CLIENTS CARTES DE CRÉDIT

Élèves :
Rydouane SIMBORO
Max Nathan BATIONO
Abdoul-Nasser BARRY
Abdoul Hakim ROUAMBA
Cheick Schakir KAGONE
Sadrac YAMEOGO

Encadrante:
Oumayma BANOUAR

Table des matières

1 Introduction

Dans le cadre de la Coding Week dédiée à machine learning, notre équipe a reçu un projet d'ingénierie visant à explorer l'application de techniques de machine learning dans le domaine de la gestion des risques financiers. Notre projet se concentre spécifiquement sur la prédiction des défauts de paiement des cartes de crédit de clients, une problématique cruciale pour les institutions financières.

Le Machine Learning est devenu un pilier fondamental dans de nombreux secteurs d'activité, offrant des solutions efficaces pour la résolution de problèmes complexes grâce à l'analyse de données et à la génération de prédictions précises. Dans ce contexte, notre projet vise à exploiter ces avancées pour développer un modèle de prédiction robuste capable d'identifier les clients présentant un risque accru de défaut de paiement.

Ce rapport présente en détail notre démarche méthodologique, de l'exploration des données à la construction et à l'évaluation des modèles de machine learning. Nous mettrons en lumière les défis rencontrés, les techniques utilisées et les résultats obtenus, dans le but de démontrer l'efficacité et la pertinence du Machine Learning dans la gestion des risques financiers liés aux cartes de crédit.

À travers ce projet, nous aspirons à contribuer à l'avancement des connaissances dans le domaine du Machine Learning appliqué, tout en fournissant aux entreprises des outils et des insights précieux pour prendre des décisions éclairées en matière de gestion des risques.

2 Contextualisation

Le machine learning est une branche de l'intelligence artificielle qui se concentre sur le développement de techniques permettant aux ordinateurs d'apprendre à exécuter des tâches sans être explicitement programmés pour les réaliser. Il repose sur l'idée que les systèmes peuvent apprendre à partir de données, identifier des modèles et prendre des décisions avec un minimum d'intervention humaine.

2.1 Qu'est ce le Machine Learning ?

Le Machine Learning est un sous-ensemble de l'intelligence artificielle (IA). Cette technologie vise à apprendre aux machines à tirer des enseignements des données et à s'améliorer avec l'expérience, au lieu d'être explicitement programmées pour le faire.

2.2 Quelles sont les différentes approches du Machine Learning ?

2.2.1 Supervised Learning

Dans le supervised learning, le modèle est entraîné sur un ensemble de données étiquetées, où chaque exemple est associé à une étiquette ou à un résultat attendu. Le but est de développer un modèle capable de prédire les étiquettes pour de nouvelles données non étiquetées.

2.2.2 Unsupervised Learning

Contrairement au supervised learning, l'unsupervised learning n'utilise pas d'étiquettes dans les données d'entraînement. Les algorithmes d'unsupervised learning cherchent à découvrir la structure intrinsèque des données sans supervision explicite.

2.2.3 Reinforcement Learning

Dans le reinforcement learning, un agent apprend à interagir avec un environnement en effectuant des actions et en observant les réponses de l'environnement. L'agent reçoit des récompenses ou des pénalités en fonction de ses actions, ce qui lui permet d'apprendre à prendre des décisions pour maximiser les récompenses à long terme.

2.3 Problématique

Comment les institutions financières peuvent-elles prédire de manière efiicace les défauts de paiement potentiels de leurs clients afin de minimiser les risques et de prendre des mesures préventives, face à la montée de l'endettement et des défauts de paiement dans les secteurs bancaires et financiers ?"

2.4 Objectifs

L'objectif principal de notre projet est de développer un modèle prédictif robuste et précis pour identifier les clients présentant un risque élevé de défaut de paiement sur leurs cartes de crédit. Pour ce faire, nous utiliserons des techniques de machine learning pour analyser un ensemble de données comprenant des informations démographiques, des données de crédit et un historique de paiement des clients. Plus spécifiquement, notre projet vise à :

- Explorer et analyser en profondeur l'ensemble de données disponible ;
- Approfondir notre compréhension des fondamentaux de machine learning et de ses applications pratiques ;
- Acquérir une maîtrise des techniques de supervised learning, en mettant l'accent sur la classification déséquilibrée. Comprendre les défis spécifiques posés par les ensembles de données déséquilibrés et explorer des méthodes telles que SMOTE pour surmonter ces défis ;
- Explorer l'application pratique des algorithmes de forêts aléatoires (Random Forest) et d'arbres de décision (Decision Trees) dans la construction de notre modèle prédictif. Comprendre comment ces algorithmes fonctionnent et comment les paramétrer efiicacement pour obtenir les meilleurs résultats ;
- Appliquer la SMOTE (Synthetic Minority Over-sampling Technique) pour résoudre le problème de la classification déséquilibrée. Comprendre en profondeur le fonctionnement de SMOTE et son impact sur l'amélioration des performances des modèles de prédiction ;
- Fournir une analyse approfondie des résultats obtenus et des insights sur les facteurs les plus influents dans la prédiction des défauts de paiement ;
- En atteignant ces objectifs, notre projet vise à acquérir des compétences et fournir aux entreprises du secteur financier un outil puissant pour la gestion proactive des risques de crédit.

3 Aspects théoriques

3.1 Logistic regression et Classification

3.1.1 Logistic regression

La régression logistique est une technique de modélisation statistique utilisée pour prédire la probabilité d'un événement binaire en fonction de plusieurs variables indépendantes. Elle est couramment utilisée pour les problèmes de classification où la variable cible est de nature binaire (par exemple, oui/non, vrai/faux)

3.1.2 Classification

La classification est une tâche de supervised learning où l'objectif est de prédire la classe (ou l'étiquette) d'une observation donnée en fonction de ses caractéristiques. Il existe plusieurs algorithmes de classification, y compris la régression logistique, les arbres de décision, les forêts aléatoires, et bien d'autres.

3.2 Etude du fonctionnement de l'algorithme SMOTE

L'algorithme SMOTE (Synthetic Minority Over-sampling Technique) est une technique d'échantillonnage utilisée pour résoudre les problèmes de déséquilibre de classe dans les ensembles de données d'apprentissage. Il génère des échantillons synthétiques de la classe minoritaire en interpolant de nouveaux exemples entre les instances de la classe minoritaire existante.

3.2.1 Déséquilibre de classe

Dans de nombreux ensembles de données, les classes peuvent être déséquilibrées, c'est-à-dire qu'il peut y avoir beaucoup plus d'exemples dans une classe (majoritaire) que dans une autre (minoritaire). Cela peut entraîner des performances médiocres du modèle, car il peut avoir tendance à prédire la classe majoritaire.

3.2.2 Suréchantillonnage (Over-sampling)

L'approche classique pour traiter le déséquilibre de classe consiste à augmenter artificiellement le nombre d'exemples de la classe minoritaire. Cependant, un simple suréchantillonnage peut entraîner un surapprentissage car il duplique simplement les exemples existants.

3.2.3 Génération synthétique (Synthetic Generation)

SMOTE résout ce problème en générant de nouveaux exemples synthétiques pour la classe minoritaire. Au lieu de simplement dupliquer les exemples existants, SMOTE crée de nouveaux exemples en interpolant entre les exemples de la classe minoritaire existante

3.2.4 Paramètres Principaux :

— sampling strategy : Ce paramètre spécifie le ratio de resampling désiré pour la classe minoritaire par rapport à la classe majoritaire après l'application de SMOTE. Par

exemple, si le déséquilibre initial est 1 :100 (classe minoritaire : classe majoritaire) et que nous voulons atteindre un équilibre de 1 :2, nous fixerons le paramètre sampling strategy à 0.5. Dans le cadre de notre projet, notre sampling strategy est de 1.

— k neighbors : C'est le nombre de voisins à considérer lors de la génération de nouveaux exemples synthétiques. Il détermine combien de voisins les exemples seront pris en compte pour l'interpolation.

3.2.5 Fonctionnement :

— Pour chaque exemple de la classe minoritaire, SMOTE sélectionne k neighbors voisins les plus proches.
— Il sélectionne ensuite aléatoirement un ou plusieurs de ces voisins et crée un exemple synthétique en prenant une combinaison linéaire des caractéristiques de l'exemple initial et de ses voisins sélectionnés.
— Ce processus est répété jusqu'à ce que le ratio de resampling désiré soit atteint.

3.3 Etude du fonctionnement du RANDOM FOREST et du DECISION TREE

3.3.1 DECISION TREE

L'arbre de décision est un modèle de machine learning qui divise récursivement l'ensemble de données en sous-ensembles plus petits basés sur les caractéristiques d'entrée, avec l'objectif de prédire la valeur de la variable cible.

— Structure de l'arbre : Un arbre de décision est une structure arborescente où chaque nœud interne représente une caractéristique (ou un attribut), chaque branche représente une règle de décision basée sur cette caractéristique, et chaque feuille représente le résultat de la classification ou de la régression.
— Nœud Racine : Le nœud racine est le premier nœud de l'arbre, à partir duquel la construction de l'arbre commence. Il représente l'ensemble complet des données d'entraînement.
— Nœuds Internes : Les nœuds internes représentent les caractéristiques qui sont utilisées pour diviser l'ensemble de données en sous-ensembles plus petits. Chaque nœud interne contient une règle de décision basée sur une caractéristique particulière.
— Feuilles : Les feuilles de l'arbre sont les nœuds terminaux où les prédictions sont faites. Chaque feuille représente une classe dans le cas de la classification ou une valeur dans le cas de la régression.
— Division de l'Arbre : L'arbre de décision est construit de manière récursive en divisant répétitivement l'ensemble de données en sous-ensembles basés sur les caractéristiques, jusqu'à ce que certaines conditions d'arrêt soient satisfaites. Ces conditions peuvent inclure une profondeur maximale de l'arbre, un nombre minimum d'échantillons dans un nœud, ou une pureté minimale des feuilles.
— Paramètres Principaux :

max depth : Profondeur maximale de l'arbre. Limite la croissance de l'arbre pour éviter le surapprentissage.

min samples split : Nombre minimum d'échantillons requis pour diviser un nœud interne. Contrôle la croissance de l'arbre et aide à prévenir le surapprentissage.

min samples leaf : Nombre minimum d'échantillons requis pour être à une feuille. Garantit que chaque feuille a un nombre minimum d'échantillons, ce qui peut aider à éviter le surapprentissage.

max features : Nombre maximum de caractéristiques à considérer lors de la recherche de la meilleure division à chaque nœud.

criterion : Méthode utilisée pour mesurer la qualité de la division à chaque nœud. Les critères couramment utilisés incluent "gini" pour l'indice de Gini et "entropy" pour l'entropie.

— Méthode de Prédiction : Pour une nouvelle observation, l'arbre parcourt les règles de décision depuis le nœud racine jusqu'à une feuille, et la prédiction est faite en utilisant la classe majoritaire (dans le cas de la classification) ou la valeur moyenne (dans le cas de la régression) des échantillons dans cette feuille

3.3.2 RANDOM FOREST

La Random Forest est un algorithme de supervised learning utilisé à la fois pour la classification et la régression.Il est basé sur l'ensemble des arbres de décision et construit un grand nombre d'arbres de décision pendant l'entraînement et combine leurs prédictions pour obtenir une prédiction finale.

— Ensemble (Forest) : L'algorithme génère un grand nombre d'arbres de décision aléatoires qui forment un ensemble, d'où le terme "Forest" (forêt). Chaque arbre est construit indépendamment des autres.

— Aléatoire (Random) : La Random Forest introduit de l'aléatoire dans le processus de construction des arbres pour améliorer la généralisation et réduire le surapprentissage. Cela se fait principalement en utilisant un sous-ensemble aléatoire des caractéristiques lors de la construction de chaque arbre.

— Paramètres Principaux : **n estimators** : Nombre d'arbres dans la forêt. Plus il y a d'arbres, mieux c'est, mais cela augmente également le temps de calcul.

max depth : Profondeur maximale de chaque arbre de décision. Contrôle la complexité de chaque arbre et donc le potentiel de surapprentissage.

min samples split : Nombre minimum d'échantillons requis pour diviser un nœud interne. Une valeur plus élevée limite la croissance de l'arbre, évitant ainsi le surapprentissage.

min samples leaf : Nombre minimum d'échantillons requis pour être à une feuille. Similaire à min samples split, mais s'applique aux feuilles.

max features : Nombre de caractéristiques à considérer lors de la recherche de la meilleure division. Une valeur plus élevée introduit plus de variabilité et peut améliorer les performances, en particulier avec de grandes quantités de caractéristiques.

— Bagging (Bootstrap Aggregating) : La Random Forest utilise une technique appelée bagging pour construire chaque arbre. Cela implique de créer plusieurs échantillons bootstrap (sous-ensembles aléatoires d'échantillons d'entraînement avec remplacement) et de construire un arbre sur chacun de ces échantillons.

— Méthode d'agrégation : Pour la classification, les prédictions des arbres individuels sont agrégées par un vote majoritaire. Pour la régression, elles sont généralement

agrégées par une moyenne.

4 Description du data set

Cette recherche avait été réalisée à Taiwan afin de déterminer par diverses méthodes si on pouvait prédire si les clients d'une banque seraient à défaut sur leurs cartes de crédit. Les données du dataset sont réparties en plusieurs attributs. Le premier groupe de variables contient les informations personnelles des clients :

- ID : L'ID de chaque client. Cette information n'est pas nécessaire pour la prédiction.
- LIMITBAL : Le montant de crédit accordé à chaque client en NT dollars.
- SEX : Le sexe du client, C'est une variable qualitative (1=homme, 2=femme)
- EDUCATION : Le niveau d'éducation, variable qualitative(1=hautes études, 2=université, 3=Lycée, 4=autres, 5=in connu, 6=inconnu)
- MARRIAGE : Le statut Marital, variable qualitative(1=marié(e), 2=célibataire, 3=autre)
- AGE : L'age en années, variable numérique Le prochain groupe d'attributs contient les informations à propos du retard de paiement relatif à un mois specifique :
- PAY0 : Etat de paiement en Septembre 2005 (-1=à jour, 1=paiement en retard d'un mois, 2=paiement en retard de 2 mois, ... 8=paiement en retard de 8 mois, 9=paiement en retard de 9 mois et +)
- PAY2 : Etat de paiement en Aout 2005 (Même échelle qu'avant)
- PAY3 : Etat de paiement en Juillet 2005 (Même échelle qu'avant)
- PAY4 :Etat de paiement en Juin 2005 (Même échelle qu'avant)
- PAY5 :Etat de paiement en Mai 2005 (Même échelle qu'avant)
- PAY6 :Etat de paiement en Avril 2005 (Même échelle qu'avant)

Les autres variables considèrent plutôt les informations relatives au montant des dépenses (i.e. un bilan mensuel des dépenses de la carte de credit) :

- BILL AMT1 : bilan mensuel en Septembre 2005(NT dollar)
- BILL AMT2 : bilan mensuel en Aoutt 2005 (NT dollar)
- BILL AMT3 : bilan mensuel en Juillet 2005 (NT dollar)
- BILL AMT4 : bilan mensuel en Juin 2005 (NT dollar)
- BILL AMT5 :bilan mensuel en Mai 2005(NT dollar)
- BILL AMT6 : bilan mensuel en Avril 2005 (NT dollar)

Les données suivantes considèrent le montant de somme remboursée précédemment dans un mois spécifique :

- PAY AMT1 :le montant de somme remboursée précédemment en Septembre 2005 (NT dollar)
- PAY AMT2 : le montant de somme remboursée précédemment en Aout 2005 (NT dollar)
- PAY AMT3 :le montant de somme remboursée précédemment en Juillet 2005 (NT dollar)
- PAY AMT4 :le montant de somme remboursée précédemment en Juin 2005(NT dollar)
- PAY AMT5 :le montant de somme remboursée précédemment en Mai 2005 (NT dollar)
- PAY AMT6 :le montant de somme remboursée précédemment en Avril 2005 (NT dollar)

La dernière variable variable est la donnée à prédire :

Default payment for the next month : Indique si les possesseurs de la carte de crédit sont en defaut ou non (1=oui , 0=non)

5 Etap es explicativ es du code

5.1 Biblioth èques utilisées

```
# determiner la donné majoritaire et minoritaire
import pandas as pd
import numpy as np
from sklearn.neighbors import NearestNeighbors
import random
from sklearn.model_selection import train_test_split
from collections import Counter
from imblearn.over_sampling import SMOTE
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import precision_score,accuracy_score, confusion_matrix,
recall_score, f1_score, confusion_matrix, roc_auc_score, roc_curve
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
```

Figure 1 – Importation des bibliothèques

— pandas (pd) : Pandas est une bibliothèque Python populaire utilisée pour la manipulation et l'analyse de données. Elle offre des structures de données flexibles et des outils pour travailler avec des tableaux de données et des séries temporelles. Elle permet de transformer des fichiers csv et excel en dataframe

— numpy (np) : NumPy est une bibliothèque Python qui fournit un support pour travailler avec des tableaux multidimensionnels et des fonctions mathématiques pour les manipuler efiicacement. Il est largement utilisé en calcul scientifique et en traitemen t des données.

— Scikit-learn est une bibliothèque Python de machine learning qui offre une large gamme d'algorithmes et d'outils pour développer, entraîner et évaluer des modèles prédictifs avec simplicité et efiicacité.

— sklearn.neighbors.NearestNeighbors : Cette classe de scikit-learn implémente des algorithmes pour la recherche des plus proches voisins. Elle est souvent utilisée pour la recherche de voisins les plus proches dans les données non étiquetées ou pour des méthodes basées sur les voisins.

— sklearn.model-selection.train-test-split : Cette fonction de scikit-learn est utilisée pour diviser un ensemble de données en ensembles d'entraînement et de test. Cela permet d'évaluer les performances d'un modèle sur des données non vues et d'éviter le surajustement.

— random : Le module random de Python fournit des fonctions pour générer des nombres aléatoires. Il est souvent utilisé pour l'initialisation aléatoire dans les algorithmes de machine learning ou pour générer des données de manière aléatoire.

— collections.Coun ter : Coun ter est une sous-classe de dictionnaire Python utilisée pour compter les occurrences des éléments dans une collection. Elle est pratique pour effectuer des calculs de fréquence ou pour compter les occurrences dans des ensembles de données.
— imblearn.o ver-sampling.SMOTE : SMOTE (Synt hetic Minorit y Over-sampling Technique) est une métho de d'échan tillonnage utilisée pour résoudre le problème de déséquilibre de classe en créan t des échan till ons synthétiques de la classe minoritaire.
— sklearn.ensem ble.Ra ndomForestClassifier : Rand omForestClassifier est un algorithme de machine learning basé sur les arbres de décision. Il construit plusieurs arbres de décision lors de l'entraînemen t et les combine pour obtenir des prédictions plus robustes et moins sujettes au sura justemen t.
— sklearn.metrics : Ce module de scikit-learn fournit des fonctions pour évaluer les performances des modèles de machine learning. Il inclut des métriques telles que la précision, le rapp el, le score F1, l'aire sous la courb e ROC, etc.
— matplotlib : Matplotlib est une bibliothèque de visualisation en Python. Le sous-module pyplot fournit une interface de style MATLAB pour tracer des graphiques et des visualisations de donn ées.
— sklearn.tree.DecisionT reeClassifier : DecisionTreeClassifier est un algorithme de machine learning basé sur les arbres de décision. Il divise l'ensemble de données en fonction des caractéristiques pour prendre des décisions de classification.
— sklearn.tree : Ce module de scikit-learn fournit des outils pour la manipulation et la visualisation des arbres de décision.

5.2 Imp ortation de la data set

```
arr_data = pd.read_excel("/content/default of credit card clients.xls",header=1,dtype='int8')
#en mettant dtype = int32, on réduit la taille de la mémoire, on divise par 8 la taille de la mémoire
arr_data = arr_data.drop(arr_data.columns[0],axis=1)
```

Figure 2 – Imp ortation des bibliothèques

Ce script charge les données à partir du fichier Excel "default of credit card clients.xls" en utilisan t 'pd.read-excel()' de la bibliothèque pandas.

— Chargemen t des données :
  La fonction 'pd.read-excel()' est utilisée pour charger les données à partir du fichier Excel spécifié. Les paramètres utilisés sont :
  — '"/con ten t/default of credit card clients.xls"' : Chemin vers le fichier Excel.
  — 'header=1' : Indique que la première ligne du fichier Excel contient les noms des colonnes sera pas pris en compte.
  — 'dtype='in t8'' : Spécifie le type de données à utiliser pour les colonnes. En utilisan t ''int8'', chaque valeur est sto ckée en tan t que nombre entier de 8 bits, ce qui réduit considérablemen t l'espace mémoire nécessaire pour sto cker les données par rapp ort à un type de données par défaut comme ''int32''.
— Suppression de la première colonne :Après le chargemen t des données, la première colonne, qui est l'identifian t est supprimée en utilisan t 'arr-data.drop(arr-data.columns[0], axis=1)'.

```
#separation des entrées X et des sorties Y
X = arr_data.drop(arr_data.columns[-1], axis=1)
Y = arr_data[arr_data.columns[-1]]
```

Figure 3 – Séparation des données

Ce script sépare les entrées (variables indép endan tes) des sorties (variable dép endan te) à partir du DataF rame ‘arr-data‘.

— La variable "X" contien t les entrées ou les variables indép endan tes. Pour cela, le DataF rame ‘arr-data‘ est utilisé, et la métho de "drop()" est appliquée sur les colonnes en utili sant "arr-data.columns[-1]" pour exclure la dernière colonne, qui est la variable dép endan te. Ainsi, "X" contien t toutes les colonnes de ‘arr-data‘ sauf la dernière. -

— La variable "Y" contien t les sorties ou la variable dép endan te. Pour cela, une extraction directe est réalisée à partir de "arr-data" en utilisan t "arr-data.columns[-1]", qui représen te la dernière colonne du DataF rame, censée être la variable dép endan te.

5.3 SMOTE from scratc h

```
def compterY(daframe) :  #compte le nombre de majorités et de minorités
  vbin = {}
  dserie = daframe.values[:,-1]
  for i in range(len(dserie)) :
    if f'{dserie[i]}' not in list(vbin.keys()) :
      vbin[f'{dserie[i]}']=1
    else :
      vbin[f'{dserie[i]}'] +=1
  return vbin #les keys sont les valeurs qualitatives et les values sont le nombre de minorité et de majorité
```

Figure 4 – Recensemen t des ma jorités et des minorités

Ce script définit une fonction nommée ‘compterY‘ qui compte le nombre d’occurrences de chaque classe dans la derni ère colonne du DataF rame passé en argumen t.

— Définition de la fonction ‘compterY‘ :
  — La fonction prend en entrée un DataF rame ‘daframe‘. fonction a pour but de compter le nombre d’occurrences de chaque classe dans la dernière colonne du DataF rame, qui est supp osée contenir les étiquettes de classe. fonction retourne un dictionnaire où les clés sont les valeurs uniques dans la dernière colonne du DataF rame (les classes), et les valeurs sont le nombre d’occurrences de chaque classe.
— Boucle de comptage :
  — Un dictionn aire vide ‘vbin‘ est initialis é pour sto cker les comptages des classes dernière colonne du DataF rame est extraite en utilisan t ‘daframe.v alues[:,-1]‘. Cette opération récup ère toutes les lignes de la dernière colonne du DataF ra me.
  — Une b oucle ‘for‘ itère à tra vers les valeurs de cette dernière colonne.
  — Pour chaque valeur dans la dernière colonne, la fonction vérifie si elle existe déjà comme clé dans le dictionnaire ‘vbin‘. Si elle n’existe pas, la valeur est ajoutée

comme clé avec une valeur initiale de 1. Sinon, si la valeur existe déjà comme clé, son comptage est incrémenté de 1.

— Retour du dictionnaire de comptage :
— Une fois que toutes les valeurs de la dernière colonne ont été comptées, le dictionnaire ‘vbin‘ contenant les comptages des classes est retourné.

5.4 Entrainemen t et manipulatio n des donn ées

Ce script définit une fonction nommée ‘pourcentagequalitatif ‘ qui calcule le pourcentage d’occurrence de chaque valeur unique dans une colonne qualitativ e d’un DataF rame.

— Extraction de la colonne :
— La colonne spécifiée par l’indice ‘number‘ est extraite du DataF rame en utilisant ‘dframe[dframe.columns[n umber]]‘. Cette colonne est ensuite convertie en une liste à l’aide de ‘.tolist()‘ et stockée dans la variable ‘col‘.
— Initialisation des variables :
  — Une liste vide ‘liste‘ est initialisée pour stocker les valeurs uniques rencontrées dans la colonne.
  — Un dictionnaire vide ‘num‘ est initialisé pour stocker le nombre d’occurrences de chaque valeur unique.
— Boucle de calcul des pourcentages :
  — Une boucle ‘for‘ itère à tra vers chaque valeur de la colonne ‘col‘.
  — Pour chaque valeur, la fonction vérifie si elle a déjà été rencontrée dans la liste ‘liste‘. Si ce n’est pas le cas, la valeur est ajoutée à la liste et un compteur correspondant est initialisé dans le dictionnaire ‘num‘.
  — Si la valeur a déjà été rencontrée, le compteur correspondant dans le dictionnaire ‘num‘ est incrémenté de 1.
— Calcul des pourcentages :
  — Une fois que tous les comptages ont été effectués, la fonction parcourt les clés du dictionnaire ‘num ‘ et calcule le pourcentage d’occurrence de chaque valeur en le divisant par la taille de la colonne.
  — Retour des pourcentages :
  — La fonction retourne un dictionnaire où les clés sont les valeurs uniques dans la colonne, et les valeurs sont les pourcentages d’occurrence correspondants.

5.5 Explicatio n de l’équilibrag e des données

Ce script concerne l’équilibrage des données à l’aide de l’algorithme SMOTE personnalisé.

— Appel de l’algorithme SMOTE personnalisé :
  — La fonction ‘SMOTEPERSO‘ est appelée pour générer de nouvelles données synthétiques pour la classe minoritaire (contenue dans le DataF rame ‘Minoritytrain‘).
  — Cette fonction prend en entrée le DataF rame des données de la classe minoritaire (‘Minoritytrain‘), le nombre maximal d’occurrences (‘nmax‘) dans une classe et le facteur de suréchantillonnage représentant le nombre de plus proche voisin (‘3‘ dans ce cas).
— Formation du tableau combiné :

- — Les données originales de la classe ma joritaire (sto ckées dans le tableau ‘mtra in‘) et les données synthétiques générées par SMOTE (sto ckées dans le tableau ‘strain‘) sont combinées pour former un seul tableau ‘ftrain‘.
- — La fonction ‘np.vstac k‘ est utilisée pour empiler verticalemen t les deux tableaux.
- — Les données sont mélangées aléatoiremen t en utilisan t ‘np.random.p ermutation‘ pour éviter tout biais intro duit par l’ordre des données.

— Création du nouveau DataF rame :
- — Un nouveau DataF rame ‘ftrain‘ est créé à partir du tableau combiné ‘ftrain‘. Les colonnes sont nommées en utilisan t les noms des colonnes originales provenan t du DataF rame ‘Minorit y-train‘ et ‘Majorit y-train‘.

— Attrib ution des nouvelles données aux variables d’entrée (‘my-x‘) et de sortie (‘my-y‘) : - Les variables d’entrée (‘my-x‘) sont définies comme toutes les colonnes de ‘ftrain‘ sauf la dernière, qui contien t les étiquettes de classe. - La variable de sortie (‘my-y‘) est définie comme la dernière colonne de ‘ftrain‘, contenan t les étiquettes de classe.

— Afiichage du compte des occurrences des classes : - La fonction ‘compterY‘ est utilisée pour compte r le nombre d’occurrences de chaque classe dans le DataF rame équilibré ‘ftrain‘. - Les résultats sont imprimés pour vérifier l’équilibrage des classes.

Ce script utilise la bibliothèque Python ‘imbalanced-learn‘ pour app liquer l’algorithme SMOTE (Synthetic Minorit y Over-sampling Technique) aux données d’entraînemen t.

— Comptage des occurrences des classes avant SMOTE** :
- — La classe ‘Counter‘ est utilisée pour compter le nombre d’occurrences de chaque classe dans ‘ytrain‘, qui représen te les étiquettes de classe des données d’entraînemen t.
- — Ces comptages sont afiichés pour chaque classe avant l’application de SMOTE.

— Application de SMOTE :
- — L’ob jet ‘SMOTE‘ est créé en utilisan t la classe ‘SMOTE‘ de la bibliothèque ‘imbalanced-learn‘.
- — La métho de ‘fitresample‘ est utilisée pour app liquer SMOTE sur les données d’entraînemen t (‘xtrain‘ et ‘ytrain‘), généran t de nouvelles données synth étiques pour équilibrer les classes.
- — - Les nouvelles données d’entraînemen t équilibrées sont sto ckées dans ‘xtrains-mote‘ et ‘ytra insmote ‘.

— Comptage des occurrences des classes après SMOTE :
- — Un nouveau comptage des occurrences de chaque classe est effectu é à l’aide de la classe ‘Counte r‘ pour ‘ytrainsmote‘.
- — Ces comptages sont afiichés pour chaque classe après l’application de SMOTE.

5.6 Implémen tation de la DECISION TREE

Cette partie du script est l’instanciation et l’entraînemen t des trois modèles d’arbre de décision.

— Instanciation des modèles d’arbre de décision :
- — tree-classif : Ce modèle est instancié avec les paramè tres par défaut de scikit-learn, avec un critère de division basé sur l’entropie (‘criterion=’en trop y’‘), un nombre minimum d’échan tillons requis pour diviser un nœud interne (‘min-samples-split=10‘), un algorithme de division optimale (‘splitter =’ best’‘), et

une profondeur maximale de l'arbre de 3 (‘max-depth=3‘).
- ‘tree-classif-smote‘ : Ce modèle est instancié avec les mêmes paramètres que ‘tree-classif‘, mais sera entraîné sur des données équilibrées en utilisan t la technique SMOTE (Synthetic Minorit y Over-sampling Technique).
- ‘tree-classif-smotefsrc‘ : Ce modèle est instancié avec les mêmes paramètres que ‘tree-classif‘, mais sera entraîné sur des donn ées équilibrées en util isant notre implémen tation personnalisée de la technique SMOTE.

— Ent raîn ement des modèles :
- ‘tree-trained‘ : Ce modèle est entra îné sur les données d'entraînemen t (‘x-train‘ et ‘y-train‘) sans l'utilisation de SMOTE. Cela signifie qu'il est entraîné sur les données initiales sans équilibrage des classes.
- ‘tree-trained-smote‘ : Ce modèle est entraîné sur les données d'entraînemen t équilibrées en utilisan t SMOTE (‘x-train-smote‘ et ‘y-train-smote‘).
- ‘tree-trained-smotefsrc‘ : Ce modèle est entraîné sur les don nées d'entraînemen t équilibrées en utilisan t une implémen tation personnalisée de SMOTE (‘my-x‘ et ‘my-y‘).

Ce script réalise la prédiction des classes sur l'ensemble de test à l'aide des modèles d'arbre de décision entraînés précédemmen t.
- Prédiction sans SMOTE : ‘ y-pred‘ : Utilise le modèle entraîné ‘tree-trained‘ pour prédire les classes sur l'ensemble de test (‘x-test‘).
- Prédiction avec SMOTE : ‘y-pred-smote‘ : Utilise le modèle entraîné ‘tree-trained-smote‘ (avec SMOTE) pour prédire les classes sur l'ensemble de test (‘x-test‘).
- Prédiction avec SMOTE from scratc h : ‘y-pred-smotefsrc‘ : Uti lise le modèle entraîné ‘tree-trained-smotefsrc‘ (avec une implémen tation personnalisée de SMOTE) pour prédire les classes sur l'ensemble de test (‘x-test‘).

Ce script calcule et afiiche plusieurs métriques de performance pour évaluer les modèles d'arbre de décision entraînés avec et sans l'utilisation de la technique SMOTE :
- Exactitude (Accuracy) :
  - La précision mesure la prop ortion d'observations correctemen t prédites par le modèle. Elle est calculée en divisan t le nombre total d'observations correctemen t prédites par le nombre total d'observations.
  - ‘accuracy ‘ : Exactitude du modèle sans SMOTE.
  - ‘accuracy -s‘ : Exactitude du modèle avec SMOTE (utilisan t la bibliothèque Python).
  - ‘accuracy -sf‘ : Exactitude du modèle avec SMOTE from scratc h (implémen tation personnalisée).
  - ‘print’ : Ces valeurs sont imprimées pour comparer les performances des différents modèles.
- Précision (Precision) :
  - La précision mesure la prop ortion d'observations prédites comme positives qui sont réellement positives. Elle est calculée en divisan t le nombre de vrais positifs par la somme des vrais positifs et des faux positifs.
  - ‘precision‘ : Précision du modèle sans SMOTE.
  - ‘precision-s‘ : Précision du modèle avec SMOTE (utilisan t la bibliothèque Python).
  - ‘precision-sf‘ : Précision du modèle avec SMOTE from scratc h (implémen tation personnalisée).

— ‘print’ : Ces valeurs sont imprimées pour évaluer la capacité des modèles à minimiser les faux positifs.
— Rapp el (Recall) :
— Le rapp el mesure la prop ortion d’observations positives réellement identifiées par le modèle. Il est calculé en divisan t le nombre de vrais positifs par la somme des vrais positifs et des faux négatifs.
— ‘recall‘ : Rapp el du modèle sans SMOTE.
— ‘recall-s‘ : Rapp el du modèle avec SMOTE (utilisan t la bibliothèque Python).
— ‘recall-sf‘ : Rapp el du modèle avec SMOTE from scratc h (impl émentation personnalisée).
— ‘print’ : Ces valeurs sont imprimées pour évaluer la capacité des modèles à minimiser les faux négatifs.
— Score F1 :
— Le score F1 est une mesure de la précision et du rapp el d’un modèle. Il est calculé comme la moyenne harmonique de la précision et du rapp el.
— ‘f1‘ : Score F1 du modèle sans SMOTE.
— ‘f1-s‘ : Score F1 du modèle avec SMOTE (utilisan t la bibliothèque Python).
— ‘f1-sf‘ : Score F1 du modèle avec SMOTE from scratc h (implémen tation personnalisée).
— ‘print’ : Ces valeurs sont imprimées pour évaluer la balance entre la précision et le rapp el des modèles.

Ce script calcule et afiiche la matrice de confusion pour évaluer les performances des modèles d’arbre de décision sur l’ensemble de te st.

Matrice de Confusion :

— La matrice de confusion est une ta ble qui montre les performances d’un modèle de classification sur un ensemble de données où les valeurs réelles sont connues. Elle présen te le nombre de prédictions correctes et incorrectes classées en fonction de leurs classes réelles et prédites.
— ‘conf-matrix‘ : Matrice de confusion du modèle sans SMOTE .
— ‘conf-matrix-s‘ : Matrice de confusion du modèle avec SMOTE (utilisan t la bibliothèque Python).
— ‘conf-matrix-sf ‘ : Matrice de confusion du modèle avec SMOTE from scratc h (implémen tation personnalisée).
— ‘print’ : Ces matrices sont imprimées pour évaluer la performance des modèles en termes de vrais positifs, faux positifs, vrais négatifs et faux négatifs.

Ce script calcule et afiiche l’aire sous la courb e ROC (AUC) pour évaluer les performances des modèles d’arbre de décision sur l’ensemble de test.

— Aire sous la courb e ROC (AUC)** :
— L’AUC est une mesure de la capacité d’un modèle de classification à distinguer entre les classes positives et négativ es.
— ‘y-pred-proba‘ : Probabilités prédites de la classe positive par le modèle sans SMOTE.
— ‘y-pred-proba-s‘ : Probabilités prédites de la classe positive par le modèle avec SMOTE (utili sant la bibliothèque Python).
— ‘y-pred-proba-sf ‘ : Probabilités prédites de la classe positive par le modèle avec SMOTE from scratc h (implémen tation personnalisée).
— ‘roc-auc‘ : AUC du modèle sans SMOTE.

— ‘roc-auc-s‘ : AUC du modèle avec SMOTE (utilisan t la bibliothèque Python).
— ‘roc-auc-sf‘ : AUC du modèle avec SMOTE from scratc h (implémen tation personnalisée).
— ‘print’ : Ces valeurs sont imprimées pour évaluer la capacité des modèles à classer correctemen t les exemples positifs et négatifs.

5.7 Implémen tation de la RANDOM FOREST

Ce code initialise un modèle de Rand omForestClassifier avec les paramètres suivants :
— n-estimators : Nombre d’arbres dans la forêt. Plus il y a d’arbres, plus le modèle est complexe et peut potentiellement mieux généralise.
— criterion : Critère utilisé pour mesurer la qualité de la séparation des branc hes des arbres. ’Entrop y’ utilise l’entro pie comme mesure de la pureté des nœuds , favorisan t la séparation des classes les plus dispersées.
— max-depth : Détermine la profondeur maxima le des arbres dans la forêt. Une profondeur plus grande permet au modèle de capturer des interactions complexes dans les données, mais peut conduire à un sura justemen t si elle est trop grande.
— min-samples-split : Nombre minimum d’échan tillons requis pour qu’un nœud soit divisé en enfants. Contrôle la croissance de l’arbre en limitan t les divisions qui ne contribuen t pas de manière significativ e à la réduction de l’impureté .
— min-samples-leaf : Nombre minimum d’échan tillons requis pour qu’un nœud soit considéré comme une feuille (une feuille est un nœud qui ne se divise plus). Cela contrôle la taille minimale des feuilles de l’arbre, aidan t à éviter le sura justemen t.
— max-features : Le nombre de fonctionnalités à considérer lors de la recherche de la meilleure division à chaq ue nœud. ’Auto’ utilise toutes les fonctionnalités, mais d’autres options comme ’sqrt’ ou ’log2’ en sélectionnen t une sous-ensemble aléatoire.
— bootstrap : Indique si les échan tillons sont tirés avec remplacemen t pour construire les arbres. Le bootstrap permet d’intro duire de la variabilité dans les échan tillons d’entraînemen t, ce qui peut améliorer la généralisation du modèle.
— random-state : Fixe la graine pour la randomisation afin de garan tir la repro ductibilité des résultats.
— class-weight : Poids associés aux différentes classes dans le problème de classification. Utile pour traiter les ensembles de données déséquilibrés en donnan t plus de poids aux classes minoritaires.
— ccp-alpha : Paramètre pour le coût-complexité minimal (minimal cost-complexit y pruning), contrôlan t la complexité de l’arbre en élaguan t les branc hes avec un coût de complexité élevé.
— En combinan t ces paramètres, le modèle RandomF orestClassifier peut être configuré pour ajuster différents compromis entre la complexité du modèle, la capacité de généralisation et la vitesse de calcul.

Dans cette partie du code une nouvelle foret aleatoire est genere pour etre entrainée sur la donnée equilibré par lalgorythme smote de python

Une autre foret aleatoire est genérée pour etre entrainée sur notre algorythme smote from scratc h

Ces ligne de codes utilisen t la métho de fit() du modèle modele-rf (les RandomF orestClassifier que nous avions défini précédemment) pour entraîner les modèles sur les données

d'entraînemen t x-train des differents cas et les étiquettes corresp ondan tes y-train. Une fois que le modèle est entraîné, il sera prêt à faire des prédictions sur de nouvelles donn ées. Les variable rf-classifier contiendron t les modèles entraînés, prêt à être utilisé pour la prédiction.

Ces lignes de codes utilisen t la métho de fit() du modèle modele-rf (les RandomF orest-Classifier que nous avions défini précédemment) pour entraîner les modèles sur les données d'entraînemen t x-train des differents cas et les étiquettes corresp ondan tes y-train. Une fois que le modèle est entraîné, il sera prêt à faire des prédictions sur de nouvelles donn ées. Les variables rf-classifier contiendron t les modèles entraînés, prêt à être utilisé pour la prédiction.

Ces lignes de codes utilisen t la métho de fit() du modèle modele-rf (les RandomF orest-Classifier que nous avions défini précédemment) pour entraîner les modèles sur les données d'entraînemen t x-train des differents cas et les étiquettes corresp ondan tes y-train. Une fois que le modèle est entraîné, il sera prêt à faire des prédictions sur de nouvelles donn ées. Les variable rf-classifier contiendron t les modèles entraînés, prêt à être utilisé pour la prédiction.

Ces lignes de codes calculent et retourn ent les exactitudes (accuracy) des modèles en comparan t les étiquettes prédites (y-pred) avec les étiquettes réelles des données de test (y-test). Voici ce que fait chaque partie du code : accuracy-score(y-test, y-pred) : C'est une fonction de la bibliothèque scikit-learn (sklearn.metrics) qui calcule l'exactitude en comparan t les étiquettes prédites avec les étiquettes réelles. Elle renvoie le pourcentage d'échan tillons correctemen t classés. cela est realisé pour les 3 cas de figures.

Ces lignes de code calculent et afiichent les précision des modèle en comparan t les étiquettes prédites (y-pred) avec les étiquettes réelles des données de test (y-test). Voici ce que fait chaque partie du code : precision-score(y-test, y-pred) : C'est une fonction de la bibliothèque scikit-learn (sklearn.metrics) qui calcule la précision en comparan t les étiquettes prédites avec les étiquettes réelles. La précision est le rapp ort des vrais positifs (TP) sur la somme des vrais positifs et des faux positifs (FP), c'est-à-dire TP / (TP + FP).

Ces lignes de code calculent et afiichent les rapp els (recall) des modèles en comparan t les étiquettes prédites (y-pred) avec les étiquettes réelles des données de test (y-test). Voici ce que fait chaque partie du code :

— recall-score(y -test, y-pred) : C'est une fonction de la bibliothèque scikit-learn (sklearn.metrics) qui calcule le rapp el en comparan t les étiquettes prédites avec les étiquettes réelles. Le rapp el est le rapp ort des vrais positifs (TP) sur la somme des vrais positifs et des faux négatifs (FN), c'est-à-dire TP / (TP + FN).

— recall = ... : Cette valeur calculée est stockée dans la variable recall pour une utilisation ultérieure ou pour être afiichée.

```
#matrice de confusion
confus_matrix = confusion_matrix(y_test, y_pred)
confus_matrixpy = confusion_matrix(y_test, y_predpy)
confus_matrixscr = confusion_matrix(y_test, y_predscr)
print("Matrice de confusion :")
print(confus_matrix)
print("Matrice de confusion smote python:")
print(confus_matrixpy)
print("Matrice de confusion smote from scratch:")
print(confus_matrixscr)
```

Figure 5 – Matrice de confusion

Ces lignes de code calculent la matrices de confusion des modèle en comparan t les étiquettes prédites (y-pred) avec les étiquettes réelles des données de test (y-test). Voici ce que fait chaque partie du code :

— confusion-matrix(y-test, y-pred) : C'est une fonction de la bibliothèque scikit-learn (sklearn.metrics) qui calcule la matrice de confusion en comparan t les étiquettes prédites avec les étiquettes réelles. Une matrice de confusion est un tableau qui montre le nombre de prédictions correctes et incorrectes classées en fonction des catégories réelles et prédites.

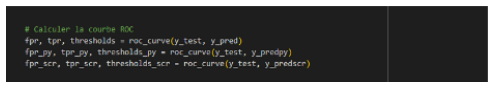


Figure 6 – Calculs Courb e

Ces lignes de code calculent la courb e ROC (Receiver Operating Characteristic) des modèles en comparan t les étiquettes prédites (y-pred) avec les étiquettes réelles des données de test (y-test). Voici ce que fait chaque partie du code :

— roc-curve(y-test, y-pred) : C'est une fonction de la bibliothèque scikit-learn (sklearn.metrics) qui calcule les taux de faux positifs (FPR) et les taux de vrais positifs (TPR) pour différentes valeurs de seuil de classification. La courb e ROC est une représen tat ion graphique du taux de faux positifs par rapp ort au taux de vrais positifs.

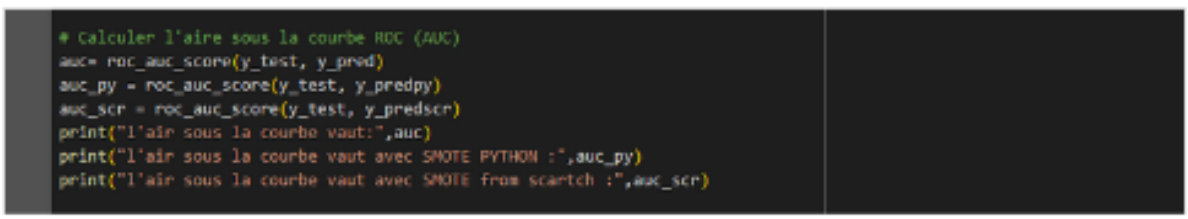


Figure 7 – Calcul Aire

Ces lignes de code calculent la courb e ROC (Receiver Operating Characteristic) des modèles en comparan t les étiquettes prédites (y-pred) avec les étiquettes réelles des données de test (y-test). Voici ce que fait chaque partie du code :

— roc-curve(y-test, y-pred) : C'est une fonction de la bibliothèque scikit-learn (sklearn.metrics) qui calcule les taux de faux positifs (FPR) et les taux de vrais positifs (TPR) pour différentes valeurs de seuil de classification. La courb e ROC est une représen tat ion graphique du taux de faux positifs par rapp ort au taux de vrais positifs.

Ce code crée une figure matplotlib qui afiiche les courb es ROC pour différents modèles, ainsi que la ligne de référence qui indique une classification aléatoire. Voici ce que fait chaque partie du code :

— plt.figure(figsize=(8, 6)) : Crée une nouvelle figure matplotlib avec une taille de 8 pouces de largeur et 6 pouces de hauteur.

— plt.plot(fpr, tpr, color='blue', lw=2, lab el='R OC curve (AUC = %0.2f)' % auc) : Trace la courb e ROC pour le modèle principal (sans SMOTE) avec les taux de faux positifs (FPR) sur l'axe des x et les taux de vrais positifs (TPR) sur l'axe des

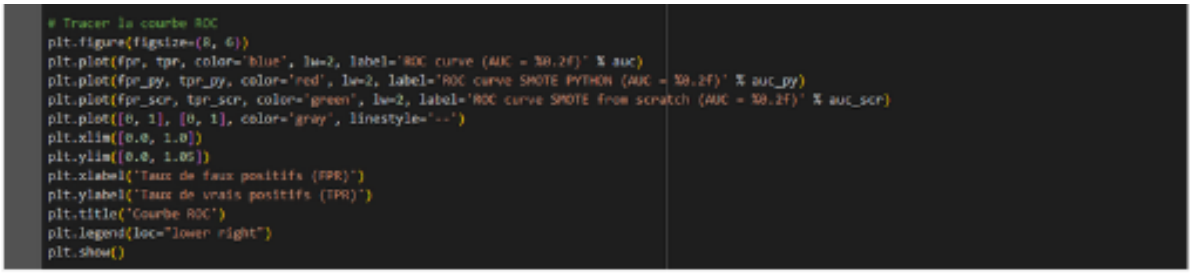


Figure 8 – Tracé Roc

y. La ligne est en bleu avec une largeur de ligne de 2, et une étiquette est ajoutée pour afiicher l'aire sous la courbe ROC (AUC).

— plt.plot(fpr-p y, tpr-p y, color='red', lw=2, label='ROC curve SMOTE PYTHON (AUC = %0.2f)' % auc-py) : Trace la courbe ROC pour le modèle utilisant SMOTE implémenté en Python avec des paramètres similaires à la première courbe, en rouge.
— plt.plot(fpr-scr, tpr-scr, color='green', lw=2, label='ROC curve SMOTE from scratch (AUC = %0.2f)' % auc-scr) : Trace la courbe ROC pour le modèle utilisant SMOTE implémenté à partir de zéro (from scratch), en vert.
— plt.plot([0, 1], [0, 1], color='gray', linestyle='–') : Trace la ligne diagonale de référence pour une classification aléatoire.
— plt.xlim([0.0, 1.0]) : Définit les limites de l'axe x de 0 à 1.
— plt.ylim([0.0, 1.05]) : Définit les limites de l'axe y de 0 à 1.05.
— plt.xlabel('Taux de faux positifs (FPR)') : Ajoute une étiquette à l'axe x.
— plt.ylabel('Taux de vrais positifs (TPR)') : Ajoute une étiquette à l'axe y.
— plt.title('Courbe ROC') : Ajoute un titre à la figure.
— plt.legend(loc="lower right") : Ajoute une légende à la figure, indiquant la signification de chaque courbe.
— plt.show() : Afiiche la figure.

Ainsi, ce code génère une figure matplotlib qui montre les courbes ROC pour différents modèles, avec les aires sous les courbes ROC (AUC) indiquées dans la légende.

6 Matrices de confusion et Métriques

Cette liste montre l'importance et le poids de chaque caractéristique dans le modèle entraîné. On constate que de toutes nos variables, ce sont les "PAY" qui comptent beaucoup

PAY-O	0.744740
PAY-2	0.180635
PAY-3	0.010690
PAY-4	0.045700
PAY-6	0.007890
EDUCATION	0.006910
PAY-AMT4	0.003434

Table 1 – Importance de certaines features

dans la prédiction du défaut de cartes de crédit, surtout PAY-0.

6.1 Accuracy- Precision- Recall- F1 Scores

	Sans SMOTE	Avec SMOTE Python	Avec SMOTE scratch
Accuracy-score	0.821	0.803	0.785
Precision-score	0.664	0.562	0.508
Recall-score	0.367	0.451	0.520
F1-score	0.473	0.500	0.514

Table 2 – Métriques obtenues après utilisation de l'algorithme de DECISION TREE

En conclusion, bien que le modèle non équilibré ait une exactitude plus élevée, il peut manquer de robustesse dans la prédiction de la classe minoritaire. Les modèles SMOTE, en revanche, offrent un compromis entre précision et rappel, ce qui peut être plus favorable dans un contexte de données déséquilibrées. Cependant, il est important de choisir le modèle en fonction des besoins spécifiques du problème et des métriques d'évaluation pertinentes.

	Sans SMOTE	Avec SMOTE Python	Avec SMOTE scratch
Accuracy-score	0.815	0.786	0.782
Precision-score	0.645	0.512	0.503
Recall-score	0.350	0.479	0.507
F1-score	0.454	0.495	0.505

Table 3 – Métriques obtenues après utilisation de l'algorithme de RANDOM FOREST

En conclusion, bien que le modèle non équilibré ait une exactitude légèrement supérieure, les modèles SMOTE, en particulier le modèle SMOTE personnalisé, offrent un meilleur équilibre entre précision et rappel, ce qui est crucial dans un contexte de données déséquilibrées. Cependant, le choix du modèle dépendra des besoins spécifiques du problème et des métriques d'évaluation prioritaires.

- VP (Vrais Positifs) : Il s'agit des cas où le modèle a prédit correctement que le client va faire défaut et effectivement le client a fait défaut.
- FP (Faux Positifs) : Il s'agit des cas où le modèle a prédit à tort que le client va faire défaut mais le client n'a pas fait défaut.
- VN (Vrais Négatifs) : Il s'agit des cas où le modèle a prédit correctement que le client ne va pas faire défaut et effectivement le client n'a pas fait défaut.
- FN (Faux Négatifs) : Il s'agit des cas où le modèle a prédit à tort que le client ne va pas faire défaut alors que le client a fait défaut.

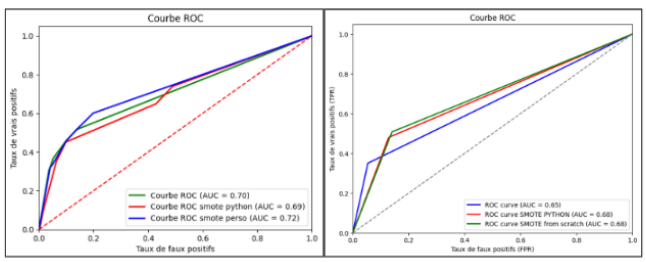
Dans notre cas, nous observons des différences significatives entre les matrices de confusion des modèles non équilibré, SMOTE Python et SMOTE personnalisé. Les modèles SMOTE ont tendance à réduire le nombre de positifs, ce qui est bénéfique dans les cas où la détection des cas positifs est critique, comme dans la prédiction de défauts de paiement.

DT Sans SMOTE SMOTE Python SMOTE Perso	Négatif	Positif
Vrai	4443 4225 4027	244 462 660
Faux	830 720 630	483 593 683

RF Sans SMOTE SMOTE Python SMOTE Perso	Positif	Negatif
Vrai	4434 4088 4030	253 599 657
Faux	853 683 646	460 630 667

Figure 9 – Deux images à la suite

DT **RF**



L'ajout de SMOTE a amélioré la capacité de discrimination du modèle dans le cas du SMOTE personnalisé, ce qui signifie que ce modèle a mieux réussi à générer des données synthétiques adaptées aux particularités du jeu de données. En revanche, dans les autres cas (sans SMOTE ou avec SMOTE Python), l'impact sur la capacité de discrimination était soit négatif, soit neutre. En résumé, le SMOTE personnalisé a mieux fonctionné pour équilibrer les données et améliorer les performances du modèle par rapport aux autres méthodes.

Nous constatons que l'utilisation de SMOTE Python a entraîné une légère amélioration de cet indicateur par rapport au modèle de base (sans SMOTE), avec une valeur de 0.676. Cependant, le SMOTE personnalisé a produit une amélioration plus significative de l'aire sous la courbe, avec une valeur de 0.684. Cela suggère que le modèle avec SMOTE personnalisé est mieux capable de discriminer entre les classes positives et négatives, ce qui peut être attribué à sa capacité à générer des données synthétiques plus adaptées au jeu de données spécifique.

Figure 10 – Deux images à la suite

6.2 Courb es ROC

En comparaison, le Random Forest semble avoir une performance intrinsèque légèrement meilleure que le Decision Tree sans SMOTE, mais l'effet du SMOTE n'est pas explicité pour le Random Forest dans les données fournies.

6.3 Remarques générales

En utilisant des techniques d'équilibrage de données telles que SMOTE, nous avons pu pallier le problème des classes déséquilibrées, ce qui a entraîné une amélioration globale de la performance de nos modèles de machine learning. Cependant, il est important de souligner que l'effet de SMOTE peut varier en fonction de plusieurs facteurs, notamment la spécificité des données et le choix du modèle. L'interprétation des métriques de performance, telles que l'exactitude, la précision, le rappel et le score F1, est cruciale pour comprendre le comportement du modèle. Par exemple, une précision élevée indique que le modèle minimise les faux positifs, tandis qu'un rappel élevé signifie qu'il minimise les faux négatifs. Ces métriques fournissent des informations complémentaires sur la capacité du modèle à prendre des décisions précises et à identifier correctement les instances de

chaque classe. De plus, l'analyse des matrices de confusion nous permet d'avoir une vue détaillée des performances du modèle en termes de vrais positifs, de faux positifs, de vrais négatifs et de faux négatifs. Cette analyse est essentielle pour comprendre comment le modèle classe correctement et incorrectement les instances de chaque classe, ce qui peut aider à identifier les domaines nécessitant des améliorations. En somme, l'utilisation de techniques d'équilibrage de données telles que SMOTE, combinée à une interprétation approfondie des métriques de performance et des matrices de confusion, nous permet de développer des modèles de machine learning plus robustes et précis, capables de prendre des décisions éclairées même dans des scénarios de classes déséquilibrées.

7 Conclusion

En conclusion, ce projet a illustré avec succès l'efiicacité des techniques de machine learning dans la prédiction des défauts de paiement des clients d'une banque. En utilisant des méthodes telles que SMOTE pour gérer les classes déséquilibrées, nos modèles ont démontré leur capacité à identifier les clients à risque de manière précise. Ces résultats soulignent l'importance de sélectionner soigneusement les algorithmes et les techniques de prétraitement des données pour obtenir des prédictions fiables. Ce projet ouvre la voie à des applications futures dans divers secteurs, montrant comment le machine learning peut être utilisé pour résoudre des problèmes commerciaux complexes. En combinant une approche méthodique avec une compréhension approfondie des données, ce projet met en évidence le potentiel du machine learning pour relever des défis dans le monde réel et ouvre de nouvelles perspectives pour la résolution de problèmes commerciaux et sociaux.

8 Bibliographie

1.https://www.analyticsvidhya.com/blog/2020/10/overcoming-class-imbalance-using-smote

2.https://www.bing.com/ck/a?!p=5ec4a52bccb925f8JmltdHM9MTcxMTIzODQwMCZp Z3

3.https: www.bing.com/ck/a?p=5ec4a52bccb925f8JmltdHM9MTcxMTIzODQwMCZp Z3V

4.https://www.stat4decision.com/fr/foret-aleatoire-avec-python/

5.https://youtu.be/UXbCBHfSGDA?si=fs7xYfF1w8y2Gfwy