Mini Projet Aide à la décision Avec Python

2éme année master professionnel

Filières : Ingénierie économique et financière

Credit-Card-Client

Élaboré par : *BICHIOU Meryem*

DHRIF Fida

Année Universitaire

2023- 2024

Introduction

Depuis 1990, le gouvernement taïwanais autorise la création de nouvelles banques. Afin d'augmenter leur part de marché, ces banques ont émis des liquidités excessives et des cartes de crédit à des demandeurs peu qualifiés. Parallèlement, la plupart des titulaires de cartes, indépendamment de leur capacité de remboursement, ont abusé de leur carte de crédit pour la consommation et accumulé des dettes importantes. Le défaut survient lorsqu'un titulaire de carte de crédit ne peut pas respecter l'obligation légale de remboursement. Cette crise a porté un coup sévère à la confiance dans le crédit à la consommation et représente un défi majeur tant pour les banques que pour les titulaires de cartes.

Le principal objectif de ce projet consiste à mettre en œuvre les compétences acquises en programmation Python, en mettant particulièrement l'accent sur la manipulation de données à l'aide de Pandas, ainsi que sur l'application des concepts d'aide à la décision. L'objectif central est d'analyser des données financières afin de prendre des décisions éclairées.

Environnement

L'analyse a été entièrement réalisée en utilisant le langage Python, en exploitant divers cadres de machine learning et statistiques tels que numpy, pandas, ainsi que d'autres bibliothèques de visualisation des données telles que matplotlib et seaborn.

Description du Jeu de Données

Le jeu de données **"Default of Credit Card Clients"** comprend 30 000 instances de statuts de cartes de crédit collectées à Taïwan d'avril 2005 à septembre 2005.

Le jeu de données utilise la variable binaire **"default payment next month"** comme variable de réponse. Elle indique si les détenteurs de cartes de crédit seront en défaut le mois prochain

(Oui = 1, Non = 0).

Plus précisément, pour chaque enregistrement (c'est-à-dire chaque client), nous disposons d'informations démographiques, de données de crédit, d'historique des paiements et de relevés de factures. Pour être plus précis, voici la liste complète des 23 prédicteurs :

- **Informations personnelles du client :**

1. LIMIT BAL : Montant du crédit accordé (en nouveaux dollars taïwanais) : il inclut à la fois le crédit à la consommation individuel et le crédit familial (supplémentaire).
2. SEXE : 1 = masculin, 2 = féminin
3. ÉDUCATION : 1 = études supérieures ; 2 = université ; 3 = lycée ; 4 = autres.
4. MARIAGE : État civil, 1 = marié ; 2 = célibataire ; 3 = autres.
5. ÂGE : Âge en années.

- **Historique des paiements passés d'avril à septembre 2005, c'est-à-dire le retard du paiement passé se réfère à un mois spécifique :**

6. PAY 0 : État de remboursement en septembre 2005.
7. PAY 2 : État de remboursement en août 2005.
8. PAY 3 : État de remboursement en juillet 2005.
9. PAY 4 : État de remboursement en juin 2005.
10. PAY 5 : État de remboursement en mai 2005.
11. PAY 6 : État de remboursement en avril 2005.

L'échelle de mesure pour l'état de remboursement est la suivante : -1 = paiement régulier ; 1 = retard de paiement d'un mois ; 2 = retard de paiement de deux mois ; ... ; 8 = retard de paiement de huit mois ; 9 = retard de paiement de neuf mois et plus.

- **Montant du relevé de facture (en nouveaux dollars taïwanais), c'est-à-dire un rapport mensuel que les sociétés de cartes de crédit émettent aux titulaires de cartes de crédit dans un mois spécifique :**

12. BILL AMT1 : Montant du relevé de facture en septembre 2005.
13. BILL AMT2 : Montant du relevé de facture en août 2005.
14. BILL AMT3 : Montant du relevé de facturation en juillet 2005.
15. BILL AMT4 : Montant du relevé de facturation en juin 2005.
16. BILL AMT5 : Montant du relevé de facturation en mai 2005.
17. BILL AMT6 : Montant du relevé de facturation en avril 2005.

- **Montant du paiement précédent (en nouveaux dollars taïwanais) :**

18. PAY AMT1 : Montant du paiement précédent en septembre 2005.
19. PAY AMT2 : Montant du paiement précédent en août 2005.
20. PAY AMT3 : Montant du paiement précédent en juillet 2005.
21. PAY AMT4 : Montant du paiement précédent en juin 2005.
22. PAY AMT5 : Montant du paiement précédent en mai 2005.
23. PAY AMT6 : Montant du paiement précédent en avril 2005.

Traitement de données :

- **Premièrement :**

```
Entrée [1]: # Chargement des bibliothèques
            import numpy as np
            import pandas as pd
            import matplotlib.pyplot as plt
            import seaborn as sns
```

Le chargement de différentes bibliothèques Python pour l'analyse de données et la visualisation. On a :

- ✓ **numpy (abrégé en np) :** Une bibliothèque pour le support de tableaux et de matrices multidimensionnels, ainsi que des fonctions mathématiques.
- ✓ **pandas (abrégé en pd) :** Une bibliothèque pour la manipulation et l'analyse de données tabulaires.
- ✓ **matplotlib.pyplot (abrégé en plt) :** Une bibliothèque pour la création de visualisations statiques en deux dimensions.
- ✓ **seaborn (abrégé en sns) :** Une bibliothèque de visualisation de données basée sur Matplotlib qui fournit une interface de haut niveau pour créer des graphiques informatifs et attrayants.

- **Deuxièment :**

```
Entrée [2]: # Importer l'ensemble de données : credit_card_clients.csv
            df = pd.read_csv("C:\\UCI_Credit_Card.csv")
```

```
Entrée [3]: # affecter l'ensemble de données df à une variable data
            data = df
```

```
Entrée [4]: #afficher l'ensemble de données data
            data
```

- On a utilisé la fonction **read_csv** de la bibliothèque Pandas pour lire les données à partir du fichier CSV situé à **"C:\UCI_Credit_Card.csv"** et les stocke dans le DataFrame **df**.
- Puis on à affecter les données du DataFrame **df** à une nouvelle variable appelée **data**. Cela signifie que **data** et **df** pointent vers le même objet DataFrame, et les modifications apportées à l'un seront reflétées dans l'autre.

- Puis on affiche le contenu du DataFrame **data**. Cela pourrait être une sortie de données tabulaires représentant les informations contenues dans le fichier CSV. Et ça se traduire dans la **figure 1**

	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_1	...	PAY_6	BILL_AMT1	...	BILL_AMT6	PAY_AMT1	...	PAY_AMT6	DEFAULT
0	1	20000	2	2	1	24	2	...	-2	3913	...	0	0	...	0	1
1	2	120000	2	2	2	26	-1	...	2	2682	...	3261	0	...	2000	1
2	3	90000	2	2	2	34	0	...	0	29239	...	15549	1518	...	5000	0
3	4	50000	2	2	1	37	0	...	0	46990	...	29547	2000	...	1000	0
4	5	50000	1	2	1	57	-1	...	0	8617	...	19131	2000	...	679	0
5	6	50000	1	1	2	37	0	...	0	64400	...	20024	2500	...	800	0
6	7	500000	1	1	2	29	0	...	0	367965	...	473944	55000	...	13770	0
7	8	100000	2	2	2	23	0	...	-1	11876	...	567	380	...	1542	0
8	9	140000	2	3	1	28	0	...	0	11285	...	3719	3329	...	1000	0
9	10	20000	1	3	2	35	-2	...	-1	0	...	13912	0	...	0	0

Figure 1 : L'ensemble de données original provenant du référentiel UCI Machine Learning, à travers le framework Pandas.

✓ A partir de cette figure nous pouvons comprendre à quoi ressemble la donnée. La variable cible **default.payment.next.month** a été renommée en DEFAULT pour des raisons de concision, tandis que la colonne **PAY 0** a été renommée en **PAY 1**

- **Troisièment :**
  - On a utilisé **la méthode info ()** du DataFrame data pour afficher des informations sur les données. elle peut fournir des détails tels que :

    ✓ La structure générale du DataFrame.
    ✓ Le nombre total d'entrées.
    ✓ Le type de données de chaque colonne.
    ✓ Le nombre de valeurs non nulles dans chaque colonne.
    ✓ L'utilisation de la mémoire par le DataFrame.

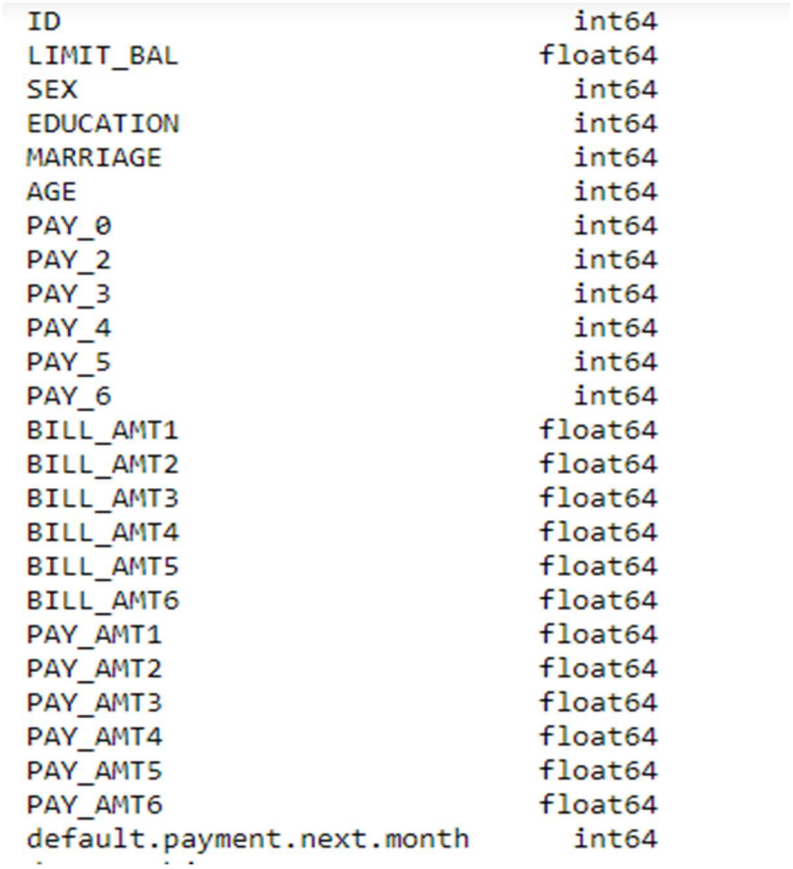
  ➢ Comme le montre la figure suivante :

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30000 entries, 0 to 29999
Data columns (total 25 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   ID                          30000 non-null  int64
 1   LIMIT_BAL                   30000 non-null  float64
 2   SEX                         30000 non-null  int64
 3   EDUCATION                   30000 non-null  int64
 4   MARRIAGE                    30000 non-null  int64
 5   AGE                         30000 non-null  int64
 6   PAY_0                       30000 non-null  int64
 7   PAY_2                       30000 non-null  int64
 8   PAY_3                       30000 non-null  int64
 9   PAY_4                       30000 non-null  int64
 10  PAY_5                       30000 non-null  int64
 11  PAY_6                       30000 non-null  int64
 12  BILL_AMT1                   30000 non-null  float64
 13  BILL_AMT2                   30000 non-null  float64
 14  BILL_AMT3                   30000 non-null  float64
 15  BILL_AMT4                   30000 non-null  float64
 16  BILL_AMT5                   30000 non-null  float64
 17  BILL_AMT6                   30000 non-null  float64
 18  PAY_AMT1                    30000 non-null  float64
 19  PAY_AMT2                    30000 non-null  float64
 20  PAY_AMT3                    30000 non-null  float64
 21  PAY_AMT4                    30000 non-null  float64
 22  PAY_AMT5                    30000 non-null  float64
 23  PAY_AMT6                    30000 non-null  float64
 24  default.payment.next.month  30000 non-null  int64
dtypes: float64(13), int64(12)
memory usage: 5.7 MB
```

Figure 2 : Les informations fournies par Pandas sur le DataFrame

- Puis on a utilisé **data.dtypes** qui répertorie les types de données seulement de chaque colonne dans le DataFrame **data**. Les types de données peuvent inclure des entiers, des nombres à virgule flottante, des chaînes de caractères, etc.

  - Comme le montre la figure suivante :



```
ID                          int64
LIMIT_BAL                 float64
SEX                         int64
EDUCATION                   int64
MARRIAGE                    int64
AGE                         int64
PAY_0                       int64
PAY_2                       int64
PAY_3                       int64
PAY_4                       int64
PAY_5                       int64
PAY_6                       int64
BILL_AMT1                 float64
BILL_AMT2                 float64
BILL_AMT3                 float64
BILL_AMT4                 float64
BILL_AMT5                 float64
BILL_AMT6                 float64
PAY_AMT1                  float64
PAY_AMT2                  float64
PAY_AMT3                  float64
PAY_AMT4                  float64
PAY_AMT5                  float64
PAY_AMT6                  float64
default.payment.next.month  int64
```

Figure 3 : Les types de données fournies par Pandas sur le DataFrame

- Ensuite on a utilisé **data.describe()** qui génère des statistiques descriptives telles que la moyenne, l'écart-type, le minimum, le 25% (Q1), la médiane (50% ou Q2), le 75% (Q3), et le maximum pour chaque colonne numérique du DataFrame **data**. Cela offre un aperçu rapide de la distribution des données.

  Et **data.describe().T** : pour transposer les statistiques, ce qui signifie échanger les lignes et les colonnes. Cela peut rendre la sortie plus lisible en plaçant les noms des variables (colonnes) à la place des indices de lignes.

  - Comme le montre la figure suivante :

	count	mean	std	min	25%	50%	75%	max
ID	30000.00	15000.50	8660.40	1.00	7500.75	15000.50	22500.25	30000.00
LIMIT_BAL	30000.00	167484.32	129747.66	10000.00	50000.00	140000.00	240000.00	1000000.00
SEX	30000.00	1.60	0.49	1.00	1.00	2.00	2.00	2.00
EDUCATION	30000.00	1.85	0.79	0.00	1.00	2.00	2.00	6.00
MARRIAGE	30000.00	1.55	0.52	0.00	1.00	2.00	2.00	3.00
AGE	30000.00	35.49	9.22	21.00	28.00	34.00	41.00	79.00
PAY_0	30000.00	-0.02	1.12	-2.00	-1.00	0.00	0.00	8.00
PAY_2	30000.00	-0.13	1.20	-2.00	-1.00	0.00	0.00	8.00
PAY_3	30000.00	-0.17	1.20	-2.00	-1.00	0.00	0.00	8.00
PAY_4	30000.00	-0.22	1.17	-2.00	-1.00	0.00	0.00	8.00
PAY_5	30000.00	-0.27	1.13	-2.00	-1.00	0.00	0.00	8.00
PAY_6	30000.00	-0.29	1.15	-2.00	-1.00	0.00	0.00	8.00
BILL_AMT1	30000.00	51223.33	73635.86	-165580.00	3558.75	22381.50	67091.00	964511.00
BILL_AMT2	30000.00	49179.08	71173.77	-69777.00	2984.75	21200.00	64006.25	983931.00
BILL_AMT3	30000.00	47013.15	69349.39	-157264.00	2666.25	20088.50	60164.75	1664089.00
BILL_AMT4	30000.00	43262.95	64332.86	-170000.00	2326.75	19052.00	54506.00	891586.00

Figure 4 : les mesures statistiques des données fournies par Pandas sur le DataFrame

- Aussi on a utilisé La **méthode head()** de Pandas qui renvoie par défaut les **cinq premières lignes** du DataFrame sur lequel elle est appliquée. Cela permet d'obtenir un aperçu rapide des données pour comprendre leur structure et leurs premières observations.

	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	...	BILL_AMT4	BILL_AMT5	BILL_AMT6	PAY_AMT1	PAY_AMT2	PAY_A
0	1	20000.0	2	2	1	24	2	2	-1	-1	...	0.0	0.0	0.0	0.0	689.0	
1	2	120000.0	2	2	2	26	-1	2	0	0	...	3272.0	3455.0	3261.0	0.0	1000.0	1(
2	3	90000.0	2	2	2	34	0	0	0	0	...	14331.0	14948.0	15549.0	1518.0	1500.0	1(
3	4	50000.0	2	2	1	37	0	0	0	0	...	28314.0	28959.0	29547.0	2000.0	2019.0	12
4	5	50000.0	1	2	1	57	-1	0	-1	0	...	20940.0	19146.0	19131.0	2000.0	36681.0	10(

5 rows × 25 columns

Et **la méthode tail()** de Pandas qui renvoie par défaut **les cinq dernières lignes** du DataFrame sur lequel elle est appliquée. Cela permet d'obtenir un aperçu rapide des dernières observations des données.

	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	...	BILL_AMT4	BILL_AMT5	BILL_AMT6	PAY_AMT1	PAY_AMT2
29995	29996	220000.0	1	3	1	39	0	0	0	0	...	88004.0	31237.0	15980.0	8500.0	20000.0
29996	29997	150000.0	1	3	2	43	-1	-1	-1	-1	...	8979.0	5190.0	0.0	1837.0	3526.0
29997	29998	30000.0	1	2	2	37	4	3	2	-1	...	20878.0	20582.0	19357.0	0.0	0.0
29998	29999	80000.0	1	3	1	41	1	-1	0	0	...	52774.0	11855.0	48944.0	85900.0	3409.0
29999	30000	50000.0	1	2	1	46	0	0	0	0	...	36535.0	32428.0	15313.0	2078.0	1800.0

5 rows × 25 columns

- Puis on a utilisé **la méthode shape ( )** qui renverra un tuple indiquant le nombre de lignes et de colonnes dans le DataFrame **data**. Dans notre exemple, on a **30 000 lignes** et **25 colonnes** dans le DataFrame.
- Après on a utilisé **la méthode nunique ( )** qui renverra une série Pandas qui répertorie le nombre de valeurs uniques pour chaque colonne du DataFrame **data**. Cela peut être utile pour comprendre la diversité des valeurs dans chaque variable et évaluer la cardinalité des différentes catégories.

```
ID                          30000
LIMIT_BAL                      81
SEX                             2
EDUCATION                       7
MARRIAGE                        4
AGE                            56
PAY_0                          11
PAY_2                          11
PAY_3                          11
PAY_4                          11
PAY_5                          10
PAY_6                          10
BILL_AMT1                   22723
BILL_AMT2                   22346
BILL_AMT3                   22026
BILL_AMT4                   21548
BILL_AMT5                   21010
BILL_AMT6                   20604
PAY_AMT1                     7943
PAY_AMT2                     7899
PAY_AMT3                     7518
PAY_AMT4                     6937
PAY_AMT5                     6897
PAY_AMT6                     6939
default.payment.next.month      2
dtype: int64
```

- Et finalement on a utilisé **la méthode duplicated ( ).sum ( )** qui renverra le nombre total de lignes dupliquées dans le DataFrame **data**. Dans notre exemple, la sortie est zéro, cela signifie qu'il n'y a pas de lignes entièrement identiques dans le DataFrame, ce qui suggère qu'il n'y a pas de valeurs dupliquées

- **Quatrièment :**
  - ✓ **"Structure de données et Nettoyage des données"**

```
trée [13]: #afficher les nombres de valeurs uniques dans la colonne 'EDUCATION' d'un DataFrame
           data['EDUCATION'].value_counts().sort_index()

Out[13]: EDUCATION
         0      14
         1   10585
         2   14030
         3    4917
         4     123
         5     280
         6      51
         Name: count, dtype: int64
```

- Ici Le code Le code **data ['EDUCATION'].value_counts().sort_index()** affiche le nombre d'occurrences de chaque valeur unique dans la colonne 'EDUCATION', triée par les valeurs uniques elles-mêmes. Cela peut être utile pour analyser la distribution des niveaux d'éducation dans le jeu de données.
  - **data['EDUCATION']:** Sélectionne la colonne 'EDUCATION' du DataFrame appelé 'data'.
  - **.value_counts():** Compte le nombre d'occurrences de chaque valeur unique dans la colonne 'EDUCATION'.
  - **.sort_index():** Trie les résultats par index, dans ce cas, par les valeurs uniques de la colonne 'EDUCATION'.

```
Entrée [14]: #afficher les nombres de valeurs uniques dans la colonne 'MARRIAGE' d'un DataFrame
             data['MARRIAGE'].value_counts().sort_index()

Out[14]: MARRIAGE
         0      54
         1   13659
         2   15964
         3     323
         Name: count, dtype: int64
```

- Ici le code **data['MARRIAGE'].value_counts().sort_index()** affiche le nombre d'occurrences de chaque valeur unique dans la colonne 'MARRIAGE', triée par les valeurs uniques elles-mêmes. Cela peut être utile pour analyser la répartition des statuts

matrimoniaux dans le jeu de données, dans notre exemple le nombre de personnes célibataires, mariées et autres

```
Entrée [26]: #supprimer les valeurs inutiles
             data = data.drop(data[data['MARRIAGE']==0].index)
             data = data.drop(data[data['EDUCATION']==0].index)
             data = data.drop(data[data['EDUCATION']==5].index)
             data = data.drop(data[data['EDUCATION']==6].index)
```

- Ici en examinant les valeurs présentes dans les attributs, certaines modifications doivent être apportées :
  - **L'attribut du mariage (marriage)** ne devrait présenter qu'une de ces valeurs : 1, 2, 3 ; cependant, dans l'ensemble de données, certaines entrées ont la valeur 0.
  - **L'attribut de l'éducation (education)** devrait présenter qu'une de ces valeurs : 1, 2, 3, 4 ; cependant, dans l'ensemble de données, certaines entrées ont les valeurs 0, 5, 6.
  - **Les attributs PAY N** devraient présenter qu'une de ces valeurs : -1, 1, 2, 3, 4, 5, 6, 7, 8, 9 ; cependant, dans l'ensemble de données, certaines entrées ont la valeur -2 et 0.
- Dans les deux premiers cas, étant donné qu'il existe un attribut qui représente la classe 'Autre' (respectivement 3 pour le mariage et 4 pour l'éducation), toutes les valeurs non connues sont mappées dans cette catégorie.
- Dans le dernier cas, afin d'utiliser ces attributs en tant qu'attributs numériques, et non pas catégoriels.

➢ Donc la présence d'erreurs dans l'ensemble de données peut être résolue en corrigeant l'attribut incorrect ou en supprimant les lignes associées à l'erreur. Nous pourrions adopter une approche conservatrice et regrouper les catégories non documentées dans d'autres catégories, mais étant donné que les entrées anormales sont relativement peu nombreuses (moins de 399,1.33% du nombre total), nous décidons de les éliminer

```
Entrée [24]: #afficher les nombres de valeurs uniques dans la colonne 'default.payment.next.month' d'un DataFrame
             data['default.payment.next.month'].value_counts()

Out[24]: default.payment.next.month
         0    22996
         1     6605
         Name: count, dtype: int64
```

- Ici le code **data['default.payment.next.month'].value_counts()** qui affiche le nombre d'occurrences de chaque valeur unique dans la colonne 'default.payment.next.month'. Cela est souvent utilisé pour comprendre la distribution des classes dans une colonne, notamment dans le contexte d'un problème de classification où la colonne peut représenter une variable cible (Dans notre exemple, 0 : c'est le nombre de non defaulters qui est 22996 et 1 : c'est le nombre de defaulters qui est de 6605).

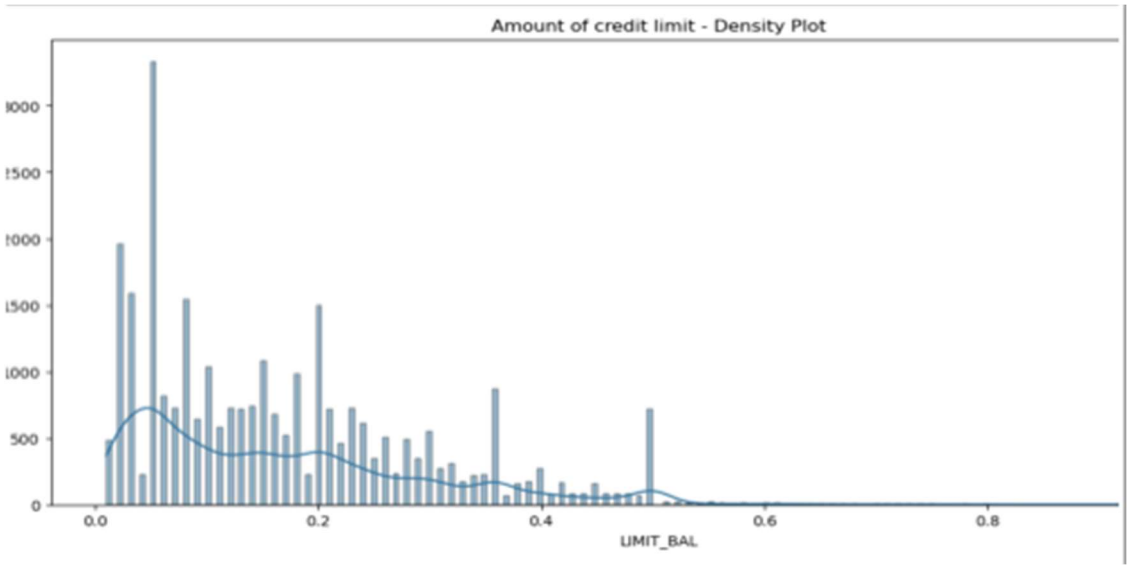
- **Cinquièment**

  - ✓ **Les représentations graphiques**

graphique de distribution des classes basé sur la variable "default.payment.next.month"

- À partir de ce figure , il est possible de voir la distribution de la variable cible **"default payment next month"** (défaut de paiement le mois prochain). Il montre clairement un déséquilibre en faveur de la classe 0 (c'est-à-dire, pas de défaut), avec environ 78% de l'ensemble de données total. Ce problème d'imbalance peut entraîner le fait que les modèles de classification se concentrent davantage sur la classe majoritaire, négligeant ainsi la classe minoritaire s'il n'est pas résolu.

graphique de densité de la distribution des montants de crédit (colonne 'LIMIT_BAL')



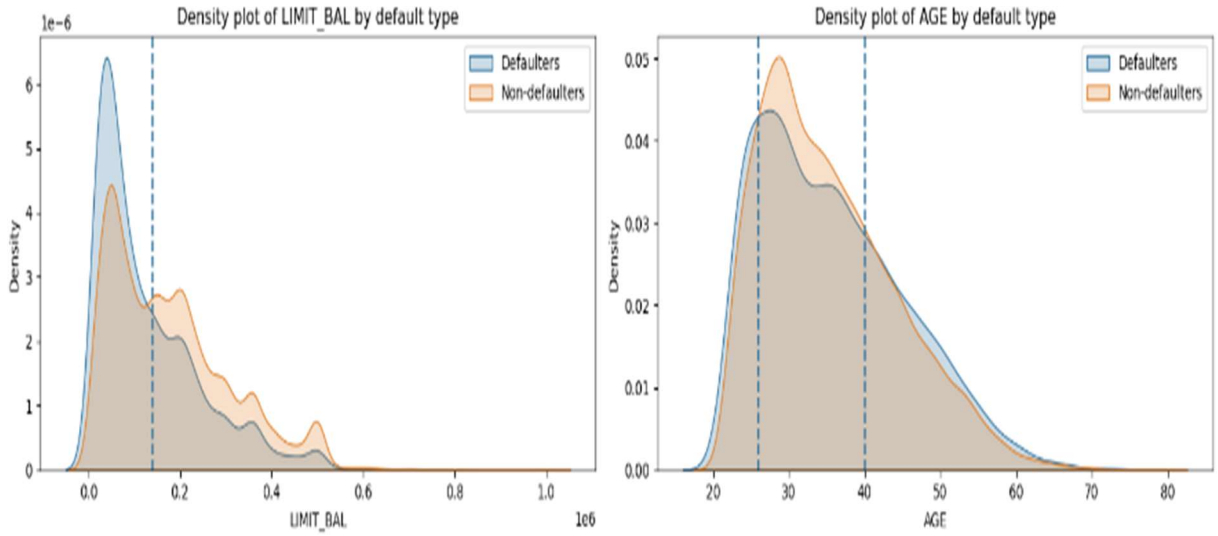
- Ce graphique représente la densité de la distribution des montants de crédit (colonne 'LIMIT_BAL') dans le jeu de données :
  - **L'axe des x** représente les montants de crédit.
  - **L'axe des y** représente la densité, indiquant la concentration relative des montants de crédit à différentes valeurs.

Observations :

- La majorité des montants de crédit semble concentrée dans certaines plages, ce qui est indiqué par les pics dans la distribution.
- On peut observer des zones où la densité est plus élevée, suggérant que certains montants de crédit sont plus fréquents que d'autres.
- La distribution semble être légèrement asymétrique, avec une queue plus longue du côté des montants de crédit plus élevés.

- En résumé, ce graphique de densité fournit un aperçu visuel de la répartition des montants de crédit dans le jeu de données, mettant en évidence les tendances et les concentrations.

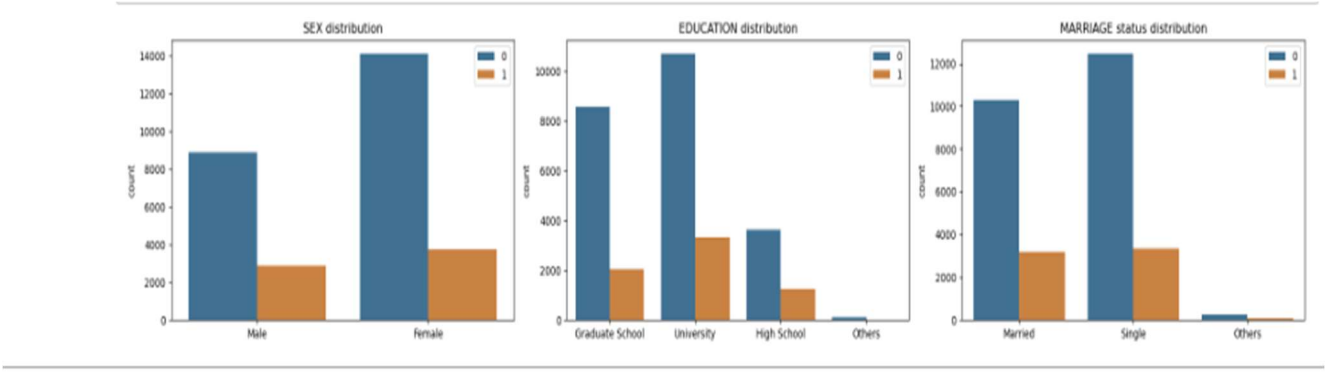
Graphique de Distribution de densité noyau



- En statistiques, l'Estimation de Densité Noyau (KDE) est une technique assez connue pour estimer la fonction de densité de probabilité de manière non paramétrique (c'est-à-dire, elle ne suppose aucune distribution sous-jacente). Ainsi, pour les caractéristiques continues suivantes, nous avons exploré leurs graphiques de densité noyau (KDE). En observant cette figure, on peut remarquer que la plupart des défauts proviennent de crédits avec un montant de **LIMIT BAL** (c'est-à-dire, le montant du crédit) plus bas, en particulier ils sont observés dans une plage allant de 0.1 Les clients au-dessus de ce seuil sont plus susceptibles de rembourser leurs dettes.

- Pour la caractéristique de **l'ÂGE,** une analyse visuelle similaire est réalisée. La probabilité de non-défaut pour les personnes âgées entre environ 25 et 42 ans est plus élevée, ce qui indique que les consommateurs sont plus capables de rembourser les prêts par carte de crédit dans ce groupe d'âge. Une hypothèse pourrait être que leur travail et leur vie familiale tendent à être stables, sans trop de pression.

Graphique des Caractéristiques catégorielles



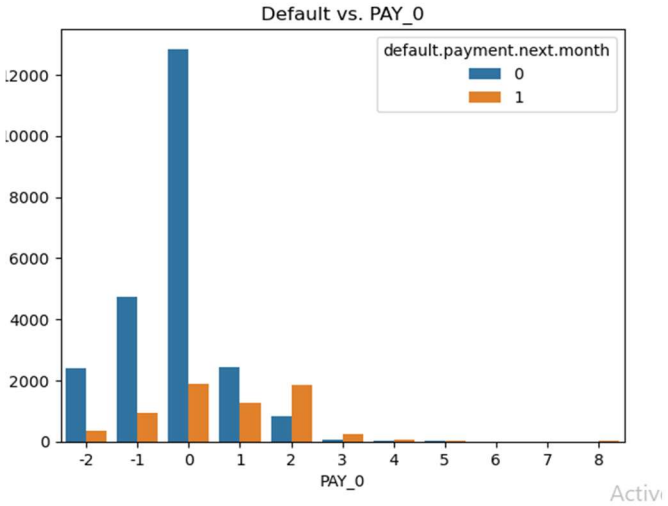
Interprétation des graphiques :

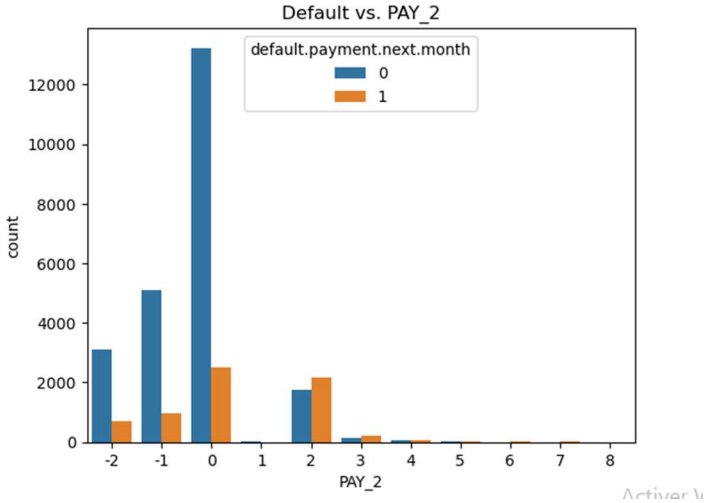
1. **Distribution du SEXE :**
    - Le premier graphique représente la distribution des sexes (Male et Female) dans le jeu de données.
    - Les deux couleurs différentes dans les barres représentent la proportion de personnes en défaut et de non-défaut pour chaque catégorie de sexe.
    - On peut observer que la majorité des individus dans l'ensemble de données sont de sexe féminin. Les hommes semblent avoir une légèrement plus grande probabilité de défaut que les femmes.
2. **Distribution de l'ÉDUCATION :**
    - Le deuxième graphique représente la distribution des niveaux d'éducation (Graduate School, University, High School, Others).
    - De manière similaire, les couleurs différentes indiquent la proportion de personnes en défaut et de non-défaut pour chaque catégorie d'éducation.
    - On remarque que les diplômés universitaires ont une probabilité de défaut relativement plus faible par rapport aux autres catégories d'éducation.
3. **Distribution du STATUT MATRIMONIAL :**
    - Le troisième graphique représente la distribution des statuts matrimoniaux (Married, Single, Others).

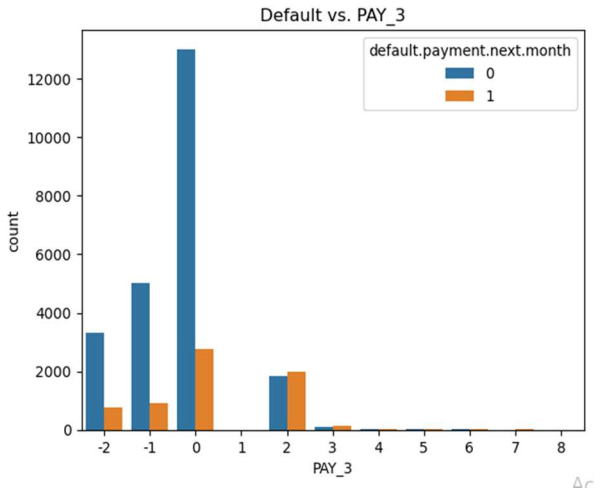
- Encore une fois, les couleurs différentes indiquent la proportion de personnes en défaut et de non-défaut pour chaque catégorie de statut matrimonial.
- On peut observer que les personnes célibataires semblent avoir une probabilité de défaut légèrement plus élevée que les personnes mariées ou autres.

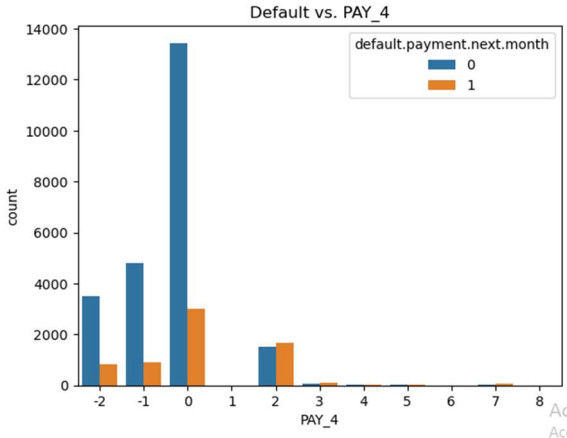
➢ En résumé, ces graphiques offrent un aperçu visuel des distributions des caractéristiques catégorielles par rapport à la variable cible (défaut de paiement le mois prochain), permettant de tirer des observations sur la relation potentielle entre ces variables et la probabilité de défaut.

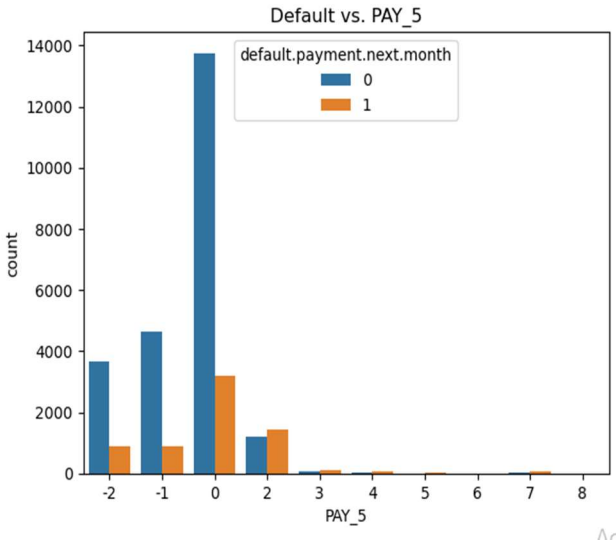
graphiques de l'historique des paiements

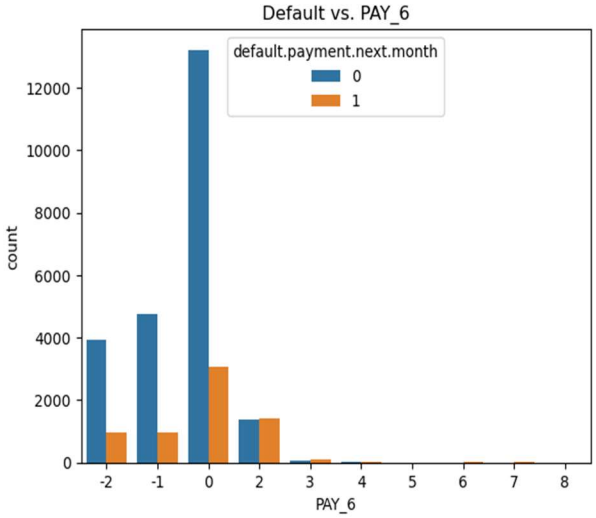












Interprétation des graphiques de l'historique des paiements :

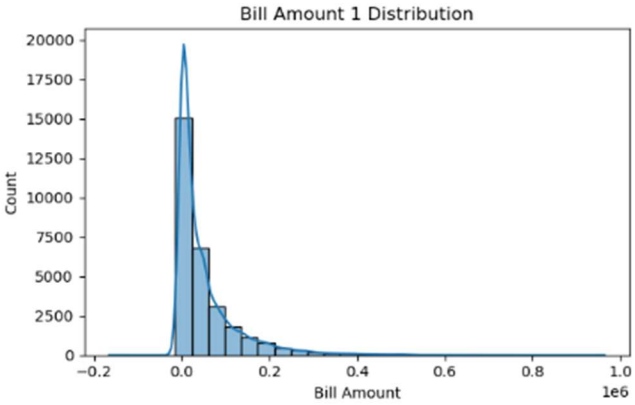
- Les graphiques représentent la distribution de la variable cible "**default.payment.next.month**" (défaut de paiement le mois prochain) pour différentes périodes de paiement (PAY_0 à PAY_6). Voici une interprétation générale :

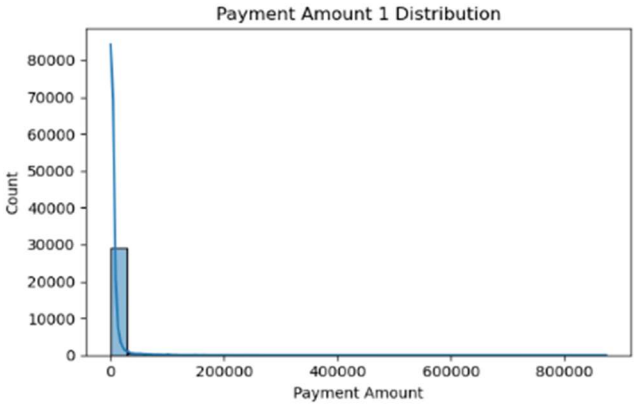
- **PAY_0 à PAY_6 :**
  - Ces graphiques présentent la distribution des états de paiement pour chaque période.
  - Les différentes couleurs dans les barres représentent la proportion de personnes en défaut et de non-défaut pour chaque état de paiement.
  - Plus la valeur de l'état de paiement est positive, plus le retard de paiement est important.

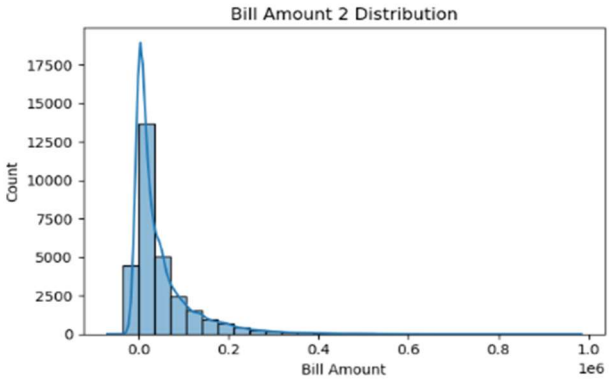
- **Observations générales :**
  - On observe que les personnes ayant des retards de paiement importants (valeurs positives) ont une probabilité plus élevée de défaut le mois suivant.
  - Les personnes ayant des retards de paiement nuls ou négatifs (paiements effectués à temps ou avec un crédit en excédent) présentent une probabilité plus faible de défaut.

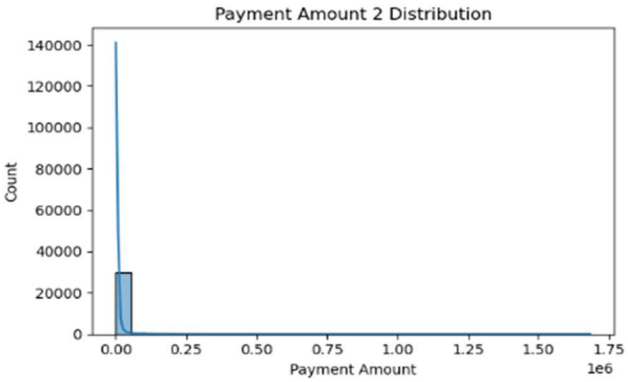
- En résumé, ces graphiques mettent en lumière l'impact de l'historique des paiements sur la probabilité de défaut. Les retards de paiement récurrents semblent être associés à une probabilité plus élevée de défaut, ce qui est cohérent avec l'intuition que les comportements de paiement passés peuvent être des indicateurs importants pour prédire les défauts futurs.

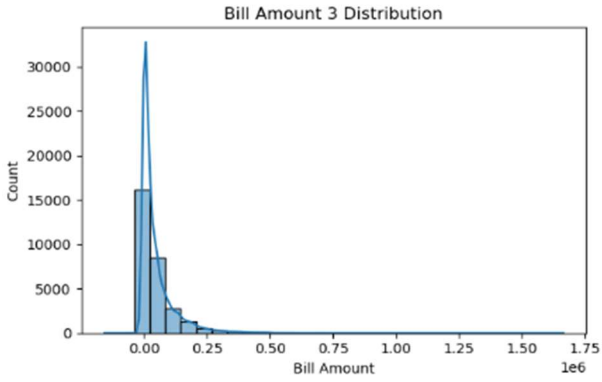
- **Des graphiques des montants de facturation et de paiement :**

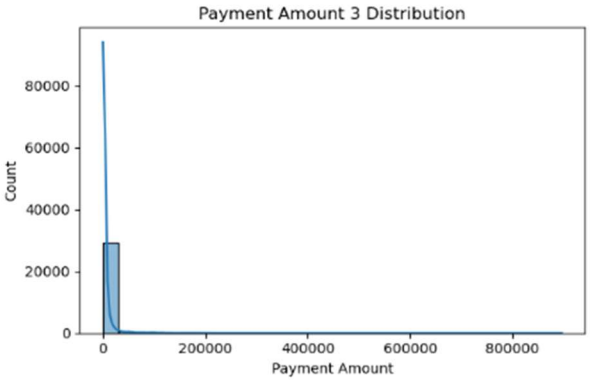


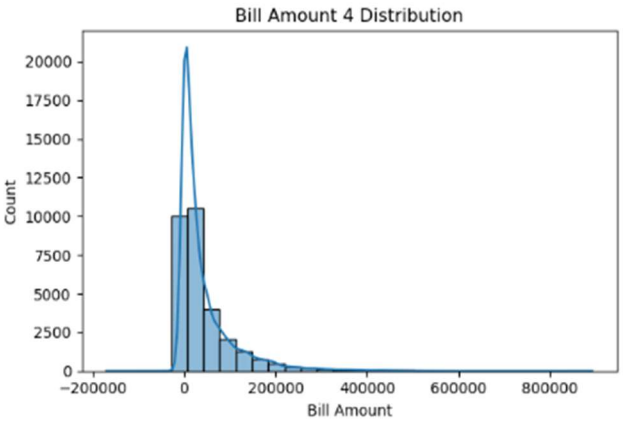


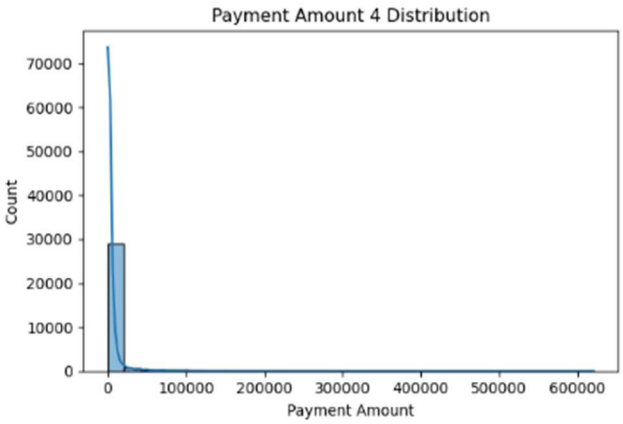


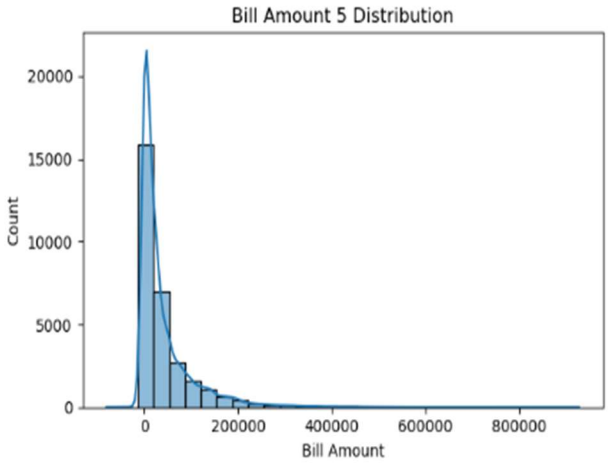


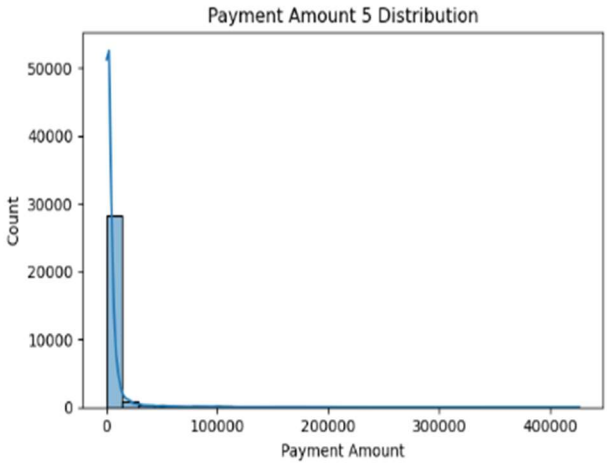


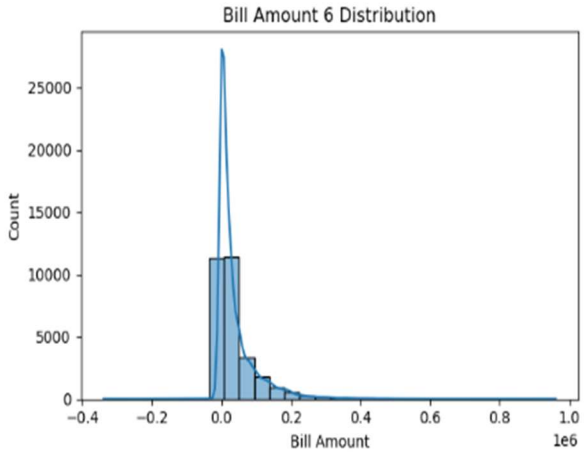


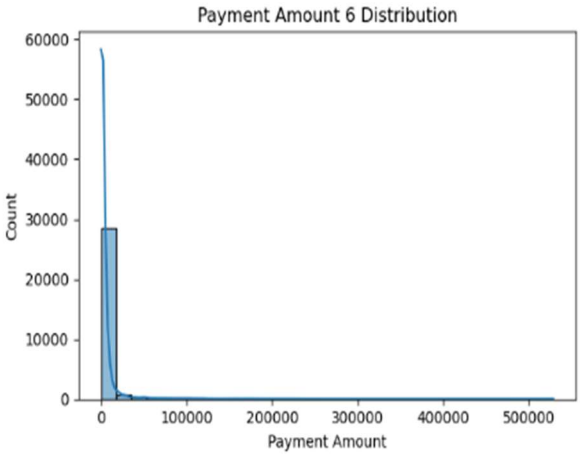

Bill Amount 4 Distribution
Count
Bill Amount
20000
17500
15000
12500
10000
7500
5000
2500
0
−200000
0
200000
400000
600000
800000


Payment Amount 4 Distribution
Count
Payment Amount
70000
60000
50000
40000
30000
20000
10000
0
0
100000
200000
300000
400000
500000
600000


Bill Amount 5 Distribution
Count
Bill Amount
20000
15000
10000
5000
0
0
200000
400000
600000
800000


Payment Amount 5 Distribution
Count
Payment Amount
50000
40000
30000
20000
10000
0
0
100000
200000
300000
400000


Bill Amount 6 Distribution
Count
Bill Amount
1e6
25000
20000
15000
10000
5000
0
−0.4
−0.2
0.0
0.2
0.4
0.6
0.8
1.0


Payment Amount 6 Distribution
Count
Payment Amount
60000
50000
40000
30000
20000
10000
0
0
100000
200000
300000
400000
500000

Interprétation des graphiques des montants de facturation et de paiement :

- Les graphiques représentent la distribution des montants de facturation (BILL_AMT) et des montants de paiement (PAY_AMT) pour différentes périodes (de 1 à 6). Voici une interprétation générale :

Montants de Facturation (BILL_AMT) :

- Les graphiques de gauche (subplot 1) représentent la distribution des montants de facturation pour chaque période.
- On peut observer la tendance générale des montants de facturation, avec une concentration autour de certaines valeurs.

Montants de Paiement (PAY_AMT) :

- Les graphiques de droite (subplot 2) représentent la distribution des montants de paiement pour chaque période.
- On peut observer la variabilité des montants de paiement, avec des concentrations à certaines valeurs.

Observations générales :

- Les montants de facturation (BILL_AMT) présentent une distribution plus étalée, avec des pics à certaines valeurs.
- Les montants de paiement (PAY_AMT) montrent également des concentrations, mais ils semblent plus dispersés que les montants de facturation.
- Ces distributions suggèrent qu'il y a une variabilité importante dans les habitudes de facturation et de paiement des clients.

➢ En résumé, ces graphiques offrent un aperçu visuel des distributions des montants de facturation et de paiement, mettant en évidence les tendances et les concentrations dans les comportements financiers des clients au fil du temps.

Graphique de corrélation

- On a utilisé **Le code data.corr()** qui calcule la corrélation linéaire entre les différentes variables dans le DataFrame, en utilisant le coefficient de corrélation de Pearson par défaut. Ce coefficient mesure la force et la direction d'une relation linéaire entre deux variables. La plage de valeurs va de -1 à 1, où :
  - ✓ 1 : indique une corrélation positive parfaite (quand une variable augmente, l'autre augmente aussi),
  - ✓ -1 : indique une corrélation négative parfaite (quand une variable augmente, l'autre diminue),
  - ✓ 0 : n'indique aucune corrélation linéaire
- La corrélation est un terme statistique décrivant le degré de coordination entre deux variables aléatoires. Intuitivement, si elles évoluent dans la même direction, alors ces variables sont définies comme ayant une corrélation positive, ou vice versa, on définit cela comme une corrélation négative. Le coefficient de corrélation de Pearson (ρ) est l'une des mesures de corrélation linéaire les plus utilisées.
- La formule du coefficient de corrélation de Pearson est donnée par :

$$P(X, Y) = \frac{Cov(X,Y)}{\sigma X \sigma Y} = \frac{E[(X - \mu X)(Y - \mu Y)}{\sigma X \sigma Y]}$$

- Après on a utilisé **le code sns.heatmap** pour créer une heatmap (carte de chaleur) qui représente la corrélation de Pearson entre la variable cible **"default.payment.next.month"** et toutes les autres variables dans le jeu de données. Voici une interprétation générale du graphique :

- **La heatmap** attribue différentes couleurs aux différentes valeurs de corrélation, où une palette de couleurs allant du bleu au rouge est utilisée (**cmap='RdBu_r'**). Les nuances de bleu indiquent une corrélation négative, les nuances de rouge indiquent une corrélation positive, et les couleurs plus pâles indiquent une corrélation plus faible.
- Les annotations (nombres dans les cellules) indiquent la valeur numérique de la corrélation de Pearson entre les paires de variables.

ɹt[70]: <Axes: >

Interprétation spécifique du graphique :

- Les cases annotées indiquent la corrélation entre la variable cible "default.payment.next.month" et chaque autre variable.
- Une valeur positive indique une corrélation positive (augmentation de l'une entraîne une augmentation de l'autre), tandis qu'une valeur négative indique une corrélation négative (augmentation de l'une entraîne une diminution de l'autre).
- La couleur et l'intensité des nuances dans la heatmap fournissent une indication visuelle de la force de la corrélation.

Observations générales :

- Si les cases sont proches de 0, cela suggère une faible corrélation linéaire entre la variable cible et les autres variables.
- Des valeurs proches de 1 ou -1 indiquent une corrélation plus forte, positive ou négative respectivement.

- En résumé, cette heatmap offre une vue d'ensemble rapide des relations linéaires entre la variable cible et les autres variables, permettant d'identifier les associations potentielles.