GROUP - 4

PREDICTING LOAN DEFAULT IN FINANCIAL LENDING - FINAL REPORT

Submission By:

Khushi Agrawal
Eksimar Kaur
Tejaswini
Diksha Yadav
Ankita

Mentor:

Charakani Siva Prasad

PROJECT INTERIM REPORT

Batch details	PGP-DSE APR 24
Team members	Khushi Agrawal Eksimar Kaur Tejaswini Diksha Yadav Ankita
Domain of Project	Finance, Loan Approval
Group Number	4
Team Leader	Khushi Agrawal
Mentor Name	Charakani Siva Prasad

Dataset name: “ Automobile Loan Default **”**

Table of Contents

01. PROJECT DESCRIPTION

In the ever-evolving financial landscape, accurately predicting loan default is paramount for financial institutions. This project aims to develop a sophisticated predictive model capable of identifying borrowers at high risk of default, enabling lenders to make informed decisions and mitigate potential losses.

By harnessing the power of machine learning, we will analyse a comprehensive dataset encompassing a wide range of borrower attributes, financial history, and economic indicators. Through rigorous data preprocessing, exploratory data analysis, and feature engineering, we will uncover underlying patterns and correlations that significantly influence loan default.

The primary objective is to construct a robust predictive model that can assess the probability of default for individual loan applications. This model will serve as a valuable tool for credit risk assessment, empowering financial institutions to:

- **Optimise Credit Decisions:** Make well-informed decisions regarding loan approvals, loan amounts, and interest rates.
- **Enhance Risk Management:** Proactively identify and manage high-risk borrowers.
- **Improve Portfolio Performance:** Minimise loan losses and optimise portfolio returns.

The anticipated impact of this project is far-reaching. By accurately predicting loan defaults, financial institutions can strengthen their credit risk management practices, reduce operational costs, and ultimately enhance overall financial stability. Furthermore, this project contributes to a more transparent and responsible lending environment, benefiting both lenders and borrowers.

1.1 Current Practices in Mitigating Loan Default Risk

Traditional methods for assessing loan default risk often rely on a combination of manual review, credit scoring models, and historical data analysis. While these methods have proven valuable, they may not fully capture the complexities of modern lending environments.

- **Manual Review:** Credit analysts manually assess loan applications, considering factors such as income, employment history, and creditworthiness. However, this process can be time-consuming, subjective, and prone to human error.
- **Credit Scoring Models:** Statistical models, such as FICO scores, are used to evaluate creditworthiness based on historical data. While effective, these models may not fully account for emerging risk factors and may not be as accurate in predicting default in specific scenarios.
- **Historical Data Analysis:** Analysing past loan performance can provide insights into future trends. However, historical data may not always accurately predict future behaviour, especially in rapidly changing economic conditions.

By leveraging advanced machine learning techniques, this project aims to surpass the limitations of traditional methods and provide more accurate and reliable predictions of loan default.

1.2 Background Research :

The financial landscape is characterised by the constant interplay between lenders and borrowers. While lending institutions aim to maximise profits, they also face the inherent risk of loan default. This risk, if not effectively managed, can lead to significant financial losses for lenders. To mitigate these risks, financial institutions have traditionally relied on a combination of credit scoring models, manual review processes, and historical data analysis. However, these methods have limitations in accurately predicting default behaviour, especially in complex and evolving economic conditions.

The increasing volume of loan applications and the sophistication of financial products have further heightened the need for robust and accurate prediction models. As a result, the development of advanced machine learning

techniques has emerged as a promising solution to address the challenge of loan default prediction. By leveraging these techniques, financial institutions can gain valuable insights into borrower behaviour, identify early warning signs of potential default, and make more informed credit decisions.

This project aims to contribute to the advancement of loan default prediction by developing a predictive model capable of assessing the likelihood of default for individual loan applications. By analysing a comprehensive dataset encompassing a wide range of borrower attributes, financial history, and economic indicators, the model will uncover hidden patterns and correlations that significantly influence default behaviour.

Loan default, the failure of a borrower to repay a loan, is a significant challenge for financial institutions. It can lead to substantial financial losses and negatively impact an institution's stability. Traditionally, financial institutions have relied on credit scoring models and manual review processes to assess creditworthiness and mitigate the risk of default. However, these methods have limitations in capturing the nuances of individual borrower behaviour and predicting future trends.

In recent years, machine learning has emerged as a powerful tool for predicting loan default. By leveraging advanced algorithms, machine learning models can analyse large datasets, identify complex patterns, and make accurate predictions. Several studies have demonstrated the superior performance of machine learning models over traditional methods in predicting loan default.

1.3 Literature Survey:

Publications, Application, Past and Undergoing Research:

https://www.kaggle.com/datasets/saurabhbagchi/dish-network-hackathon?select=Data_Dictionary.csv

Predicting Loan Default in Financial Lending

The prediction of loan default has been a significant focus in the field of finance and machine learning. Financial institutions have long sought effective methods to assess credit risk and minimise losses. In recent years, machine learning techniques have emerged as powerful tools for predicting loan default.

A substantial body of research has explored the application of machine learning algorithms to loan default prediction. These studies have demonstrated that machine learning models can outperform traditional statistical models in terms of accuracy and predictive power.

Key findings from the literature:

- **Machine Learning Algorithms:** A variety of machine learning algorithms, including logistic regression, decision trees, random forests, and gradient boosting, have been successfully applied to predict loan default. These algorithms can capture complex patterns and relationships within the data, leading to improved predictive performance.
- **Data Preparation and Feature Engineering:** Data quality and feature engineering are crucial for accurate model predictions. Cleaning and handling missing values, outlier detection, and feature engineering (e.g., creating new features like debt-to-income ratio) are essential steps in preparing the data for modelling.
- **Model Evaluation:** Various metrics can be used to evaluate the performance of a loan default prediction model, including accuracy, precision, recall, F1-score, and ROC curve. The choice of metric depends on the specific business objectives and the cost of false positives and false negatives.
- **Model Interpretability:** Understanding the factors that influence the model's predictions is important for building trust and making informed decisions. Techniques like feature importance analysis and SHAPE values can help interpret the model's decision-making process.
- **Dynamic Economic Conditions:** The performance of loan default prediction models can be affected by changes in economic conditions. It is important to periodically retrain and update models to account for evolving economic factors.

By leveraging machine learning, financial institutions can build more accurate and reliable models to assess creditworthiness and make informed lending decisions. This can help reduce loan defaults, improve risk

management, and enhance overall financial performance.

DATA DICTIONARY

Column No.	Column Name	Description
1	ID	Client Loan application ID
2	Client_Income	Client Income in $
3	Car_Owned	Any Car owned by client before applying for the loan for another car (0 means No and 1 means otherwise)
4	Bike_Owned	Any bike owned by client (0 means No and 1 means otherwise)
5	Active_Loan	Any other active loan at the time of application of loan (0 means No and 1 means otherwise)
7	House_Own	Any house owned by client (0 means No and 1 means otherwise)
8	Child_Count	Number of children the client has
9	Credit_Amount	Credit amount of the loan in $

10	Loan_Annuity	Loan annuity in $
11	Accompany_Client	Who accompanied the client when client applied for the loan
12	Client_Income_Type	Clients income type
13	Client_Education	Highest level of education achieved by client
14	Client_Marital_Status	Marital status of client (D- Divorced, S- Single, M- Married, W- Widowed)
15	Client_Gender	Gender of the Client
16	Loan_Contract_Type	Loan Type (CL- Cash Loan, RL- Revolving Loan)
17	Client_Housing_Type	Client Housing situation
18	Population Region Relative	Relative population of the region where the client is living. Higher value means the client is living in more populated area
19	Age_Days	Age of the client at the time of application submission
20	Employed_Days	Days before the application, the client started earning
21	Registration_Days	Days before the loan application, the client changed his/her registration
22	ID_Days	Days before the loan application, the client changed his/her identity document with which the loan was applied
23	Own_House_Age	Age of Client's house in years
24	Mobile_Tag	Mobile Number provided by Client (1 means Yes and 0 means No)
25	Homephone_Tag	Home Phone Number provided by Client (1 means Yes and 0 means No)
26	Workphone_Working	Was work phone number reachable (1 means Yes and 0 means No)

27	Client_Occupation	Client Occupation type
28	Client_Family_Member	Number of family members does client have
29	Cleint_City_Rating	Client city rating. 3 denotes best and 2 denotes good and 1 denotes average
30	Application_Process_Day	Day of the week on which client applied for the loan (0-Sun, 1-Mon,2-Tues, 3-Wed, 4-Thrus,5-Fri, 6-Sat)
31	Application_Process_Hour	Hour of the day on which client applied for the loan
32	Client_Permanent_Match_Tag	Indication if client contact address does not match permanent address.
33	Client Contact Work Tag	Indication if client work address does not match contact address.
34	Type_Organization	Type of organisation where client works
35	Score_Source_1	Score sourced from another source. This is a normalised score
36	Score_Source_2	Score sourced from another source. This is a normalised score
37	Score_Source_3	Score sourced from another source. This is a normalised score
38	Social_Circle_Default	How many friends/family member of client defaulted on any loan payment in last 60 days
39	Phone_Change	How many days before the loan application, client changed his/her phone
40	Credit_Bureau	Total number of enquiries in last year

41	Default	1 means the client defaulted on loan payments and 0 means otherwise

02. DATA DESCRIPTION

2.1. DATASET AND DATA TYPES

```
[ ] df=pd.read_csv(r'/content/drive/MyDrive/CAPSTONE/Train_Dataset.csv')
```

```
df.head()
```

	ID	Client_Income	Car_Owned	Bike_Owned	Active_Loan	House_Own	Child_Count	Credit_Amount	Loan_Annuity	Accompany_Client	Client_Income_Ty
0	12142509	6750	0.0	0.0	1.0	0.0	0.0	61190.55	3416.85	Alone	Commerc
1	12138936	20250	1.0	0.0	1.0	NaN	0.0	15282	1826.55	Alone	Servi
2	12181264	18000	0.0	0.0	1.0	0.0	1.0	59527.35	2788.2	Alone	Servi
3	12188929	15750	0.0	0.0	1.0	1.0	0.0	53870.4	2295.45	Alone	Retir
4	12133385	33750	1.0	0.0	1.0	0.0	2.0	133988.4	3547.35	Alone	Commerc

The preview shows the first few rows of the dataset, which includes various details about loan applicants such as their income, car ownership, active loans, house ownership, child count, credit amount, loan annuity, and client type.

```
print(f' Number of Columns :',df.shape[1])
print(f' Number of Rows :',df.shape[0])
df.info()
```

```
 Number of Columns : 40
 Number of Rows : 121856
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 121856 entries, 0 to 121855
Data columns (total 40 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   ID                          121856 non-null  int64
 1   Client_Income               118249 non-null  object
 2   Car_Owned                   118275 non-null  float64
 3   Bike_Owned                  118232 non-null  float64
 4   Active_Loan                 118221 non-null  float64
 5   House_Own                   118195 non-null  float64
 6   Child_Count                 118218 non-null  float64
 7   Credit_Amount               118224 non-null  object
 8   Loan_Annuity                117044 non-null  object
 9   Accompany_Client            120110 non-null  object
```

The dataset contains information about 121,856 loan applicants, with 40 features or variables for each applicant. The features include:

Numerical Features:

- **ID:** Unique identifier for each applicant.
- **Car_Owned, Bike_Owned, Active_Loan, House_Own, Child_Count, Mobile_Tag, Homephone_Tag, Workphone_Working, Client_Family_Members, Cleint_City_Rating, Application_Process_Day, Application_Process_Hour, Phone_Change, Credit_Bureau, Default:** These features represent numerical values, likely indicating counts, binary flags, or continuous measurements.

Categorical Features:

- **Client_Income, Credit_Amount, Loan_Annuity, Accompany_Client, Client_Income_Type, Client_Education, Client_Marital_Status, Client_Gender, Loan_Contract_Type, Client_Housing_Type, Population_Region_Relative, Age_Days, Employed_Days, Registration_Days, ID_Days, Own_House_Age, Client_Occupation, Type_Organization, Score_Source_3, Client_Permanent_Match_Tag, Client_Contact_Work_Tag:** These features represent categorical data, likely indicating different categories or groups.

Overall, this dataset provides a rich source of information for analysing loan applicant behaviour and predicting loan default.

2.2. DATA TYPE CONVERSION

Converting incorrect columns dtypes to correct dtypes

```
[ ] # Convert object columns to numeric, errors='coerce' will replace non-numeric values with NaN
    correct_columns = ['Client_Income','Credit_Amount','Loan_Annuity','Population_Region_Relative','Age_Days','Employed_Days','Registration_Days',
                       'ID_Days','Score_Source_3']
    for cols in correct_columns:
        df[cols] = pd.to_numeric(df[cols], errors='coerce')
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 121856 entries, 0 to 121855
Data columns (total 40 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   ID                          121856 non-null  int64
 1   Client_Income               118234 non-null  float64
 2   Car_Owned                   118275 non-null  float64
 3   Bike_Owned                  118232 non-null  float64
 4   Active_Loan                 118221 non-null  float64
 5   House_Own                   118195 non-null  float64
```

The code snippet demonstrates the conversion of several columns from object (string) data type to numeric data type. This is a crucial step in data preprocessing, as it enables numerical calculations and analysis on these columns. By converting these columns to numeric, we can perform operations like calculating summary statistics, visualisations, and potentially using them as features in a machine learning model. The `errors='coerce'` argument in the `pd.to_numeric()` function handles non-numeric values by converting them to NaN (Not a Number), allowing the conversion process to continue without errors.

2.3. DROPPING UNWANTED FEATURES

Since we came accross the % of nulls present into the dataset w.r.t columns the following columns are decided to drop with threshold of 50% and above :

- **Own House Age**
- **Score scource 1**
- **Social Circle Default**

```
[ ] df_new=df.drop(['Own_House_Age','Social_Circle_Default','Score_Source_1'],axis=1)
    df_new.head(1)
```

ID	Client_Income	Car_Owned	Bike_Owned	Active_Loan	House_Own	Child_Count	Credit_Amount	Loan_Annuity	Accompany_Client	Client_Income_Type
12142509	6750.0	0.0	0.0	1.0	0.0	0.0	61190.55	3416.85	Alone	Commercial

The code snippet demonstrates the removal of columns with a high percentage of missing values.

The specific columns being dropped are:

- Own_House_Age
- Score_Source_1
- Social_Circle_Default

These columns are likely removed due to having more than 50% of their data missing, making them unreliable for analysis and potentially introducing bias into the model.

After dropping these columns, the user displays the first few rows of the cleaned dataset to visualise the remaining features. This step is crucial to ensure that the data is ready for further analysis and modelling.

03. NULL VALUE AND OUTLIERS TREATMENT

```
null_values=(df.isnull().sum()/df.shape[0])*100

round(null_values.sort_values(ascending=False),2)
```

	0
Own_House_Age	65.73
Score_Source_1	56.49
Social_Circle_Default	50.82
Client_Occupation	34.00
Score_Source_3	22.09
Credit_Bureau	15.21
ID_Days	4.90
Score_Source_2	4.67
Population_Region_Relative	3.99
Loan_Annuity	3.95

The code calculates the percentage of missing values in each column and sorts them in descending order.

The results indicate that the following columns have a high percentage of missing values:

- Own_House_Age: 65.73%
- Score_Source_1: 56.49%
- Social_Circle_Default: 50.82%
- Client_Occupation: 34.00%
- Score_Source_3: 22.09%
- Credit_Bureau: 15.21%
- ID_Days: 4.90%
- Score_Source_2: 4.67%
- Population_Region_Relative: 3.99%
- Loan_Annuity: 3.95%

These findings suggest that a significant portion of data is missing in these columns. The user may need to consider strategies like imputation, removal, or feature engineering to handle these missing values before proceeding with further analysis or modelling.

Imputing Missing Values

Imputing the missing values

+ Code + Text

```
# Impute missing values in numeric columns with their median
for col in numeric_df.columns:
    # Check if the column exists in df_new before imputing
    if col in df_new.columns:
        df_new[col].fillna(df_new[col].median(), inplace=True)
print("\nDataFrame after imputing numeric columns with median:\n")
df_new.head(1)
```

```
DataFrame after imputing numeric columns with median:
```

ID	Client_Income	Car_Owned	Bike_Owned	Active_Loan	House_Own	Child_Count	Credit_Amount	Loan_Annuity	Accompany_Client	Client_Income_Type
12142509	6750.0	0.0	0.0	1.0	0.0	0.0	61190.55	3416.85	Alone	Commercial

```
[ ] # Impute missing values in category or object dtype columns with the most frequent value (mode)
    for col in object_df.columns:
        if col in df_new.columns:  # Check if column was retained after dropping
            df_new[col].fillna(df_new[col].mode()[0], inplace=True)
    print("\nDataFrame after imputing object columns with mode:\n")
    df_new.head(1)
```

DataFrame after imputing object columns with mode:

ID	Client_Income	Car_Owned	Bike_Owned	Active_Loan	House_Own	Child_Count	Credit_Amount	Loan_Annuity	Accompany_Client	Client_Income_Type
12142509	6750.0	0.0	0.0	1.0	0.0	0.0	61190.55	3416.85	Alone	Commercial

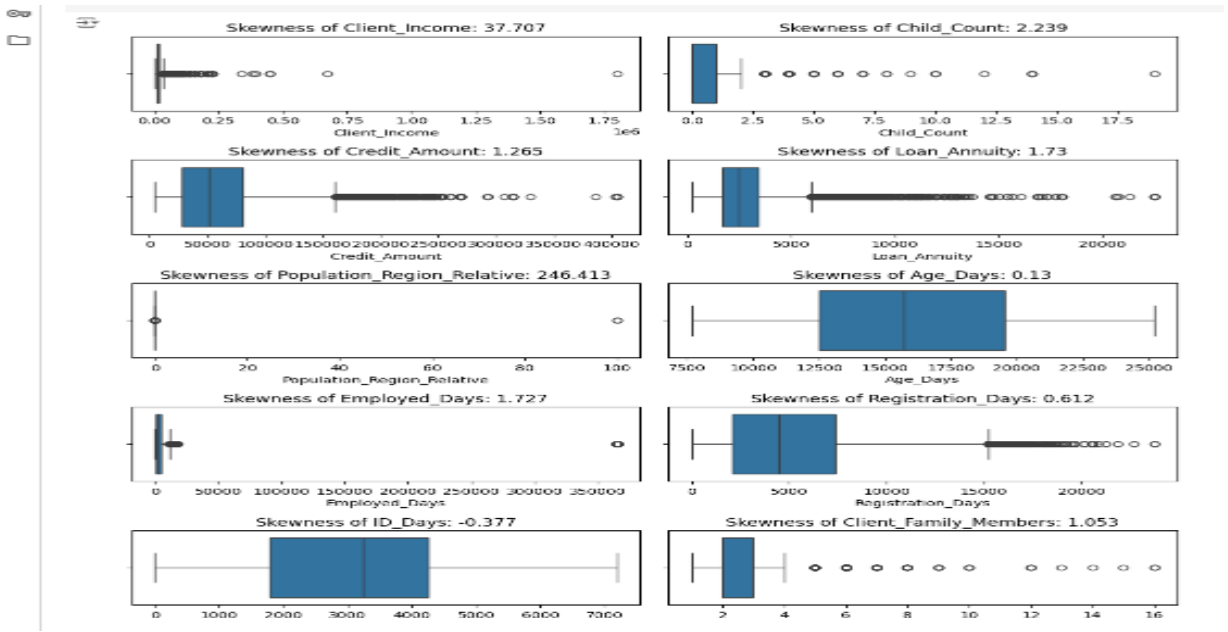
The code snippet imputes missing values in categorical and numerical columns using the most frequent value (mode) and median . This is a common technique for handling missing categorical data.

Finding Outliers

```
t=1
plt.figure(figsize = [10,20])

for i in numeric_df.columns:
  if i in df_new.columns:
    plt.subplot(10,2,t)
    sns.boxplot(data=df_new, x=i)
    plt.title(f'Skewness of {i}: {round(df_new[i].skew(),3)}')
    t+=1
plt.tight_layout()
plt.show()
```

In essence, the code visualises the distribution of two numeric columns using boxplots and provides information about their skewness.

BOXPLOT



INFERENCES FROM BOXPLOT:

1. Client_Income: Extremely high positive skew (37.707), indicating most clients have low income, with a few high-income outliers.
2. Child_Count: Moderate positive skew (2.239), showing most clients have few children, with some high

outliers.

3. Credit_Amount: Moderate positive skew (1.265), with higher loan amounts as outliers.
4. Loan_Annuity: Moderate positive skew (1.73), indicating most clients have low to moderate annuity amounts, with some high-value outliers.
5. Population_Region_Relative: Extremely high positive skewness (246.413), indicating clients are living in less populated areas.
6. Age_Days: Nearly symmetric (0.13), showing an even age distribution with minimal skew.
7. Employed_Days: Moderate positive skewness (1.727), showing mostly employed days are less than 50000 with an extreme outlier at greater than 350000.
8. Registration_Days: Nearly symmetric distribution (0.612), indicating outliers above 15000.
9. ID_Days: Slight negative skew (-0.377) with lower outliers, indicating clients with long-standing IDs.
10. Client_Family_Members: Moderate positive skew (1.053) with outliers, representing clients with larger family sizes.
11. Application_Process_Hour: Nearly symmetric (-0.033), with a few outliers at unusual application times.
12. Score_Source_2: Extremely high positive skew (128.332), suggesting concentration of low scores with high-value outliers, potentially needing transformation.
13. Score_Source_3: Slight negative skew (-0.566) with some low-value outliers.
14. Phone_Change: Moderate positive skew (0.748), with outliers at the high end, possibly indicating clients frequently changing phone numbers.
15. Credit_Bureau: Moderate positive skew (1.513) with higher outliers, potentially indicating high credit activity.

These observations highlight possible outliers in income, child count, loan amount, and annuity.

Treating Outliers

Treating Outliers

```
# Capping

for i in numeric_df:
    # Calculate Q1, Q3, and IQR
    q3, q1 = np.percentile(df[i], [75, 25])
    iqr = q3 - q1

    # Define upper and lower limits for outliers
    ul, ll = q3 + 1.5 * iqr, q1 - 1.5 * iqr

    # Replace outliers with the upper or lower limit
    df[i] = df[i].apply(lambda x: ul if x > ul else ll if x < ll else x)
```

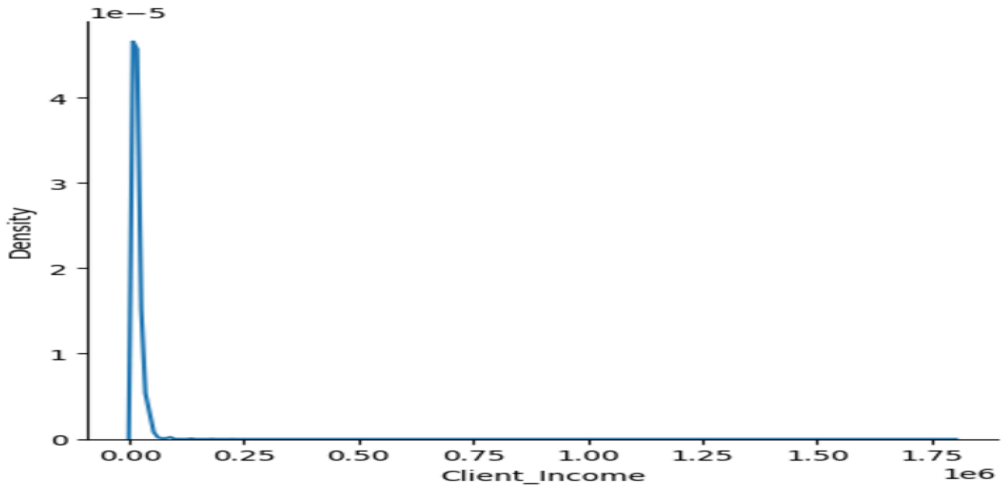
```
[ ] t=1
    plt.figure(figsize = [10,20])

    for i in numeric_df.columns:
      if i in df_new.columns:
        plt.subplot(10,2,t)
        sns.boxplot(data=df_new, x=i)
        plt.title(f'Skewness of {i}: {round(df_new[i].skew(),3)}')
        t+=1
    plt.tight_layout()
    plt.show()
```

By addressing outliers and visualising the data, the project aims to gain better insights and potentially build more accurate models.

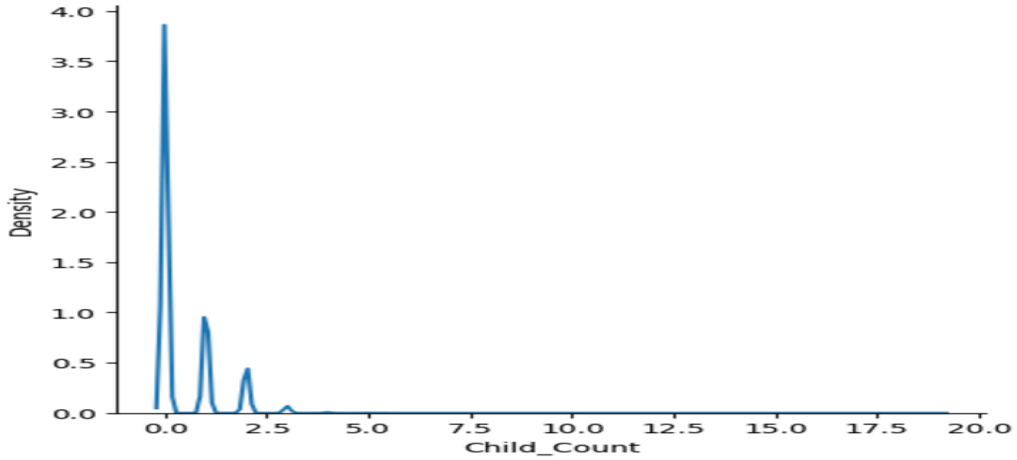
04 . Univariate, Bivariate and Multivariate Analysis

UNIVARIATE ANALYSIS

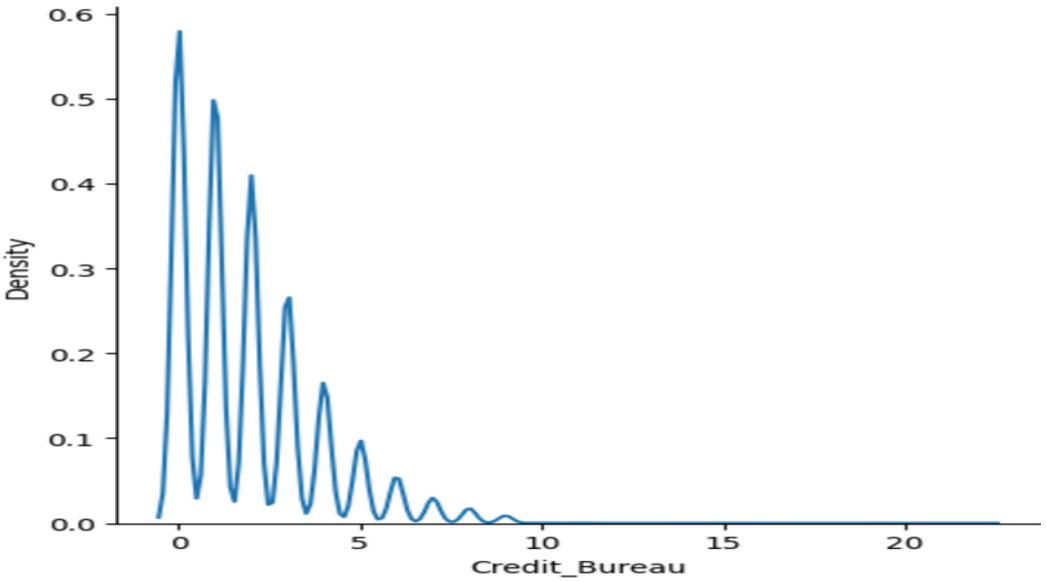
NUMERICAL COLUMN



Client Income: Right-skewed distribution with most incomes concentrated in the lower range. Potential outliers among high-income clients.



Child Count: Most clients have 0-2 children, with very few having more than 3. Outliers with unusually high child counts



Credit Bureau: Discrete values indicating the number of credit checks, higher counts may imply financial distress.

In a similar manner,while viewing all the distplot these are the insights.

Credit Amount: Right-skewed, most loans are small with fewer large loan amounts.

Loan Annuity: Right-skewed, most clients have low annuities, with outliers at the higher end.

Population Region Relative: Majority of clients are from less densely populated areas.

Age (in days): Clients are primarily within a working-age group.

Employment Days: Most clients have short employment tenure, extreme outliers are identified.

Registration Days: Skewed distribution, most clients have been registered for a shorter duration.

ID Days: Indicates issuance/verification timeframes, peaks at specific durations.

Own House Age: Concentrated at lower values, with a peak at older ages of house around 60-70 years.

Client Family Members: Most clients have 1-3 family members, outliers beyond 6 family members present in the dataset.

Application Process Hour: Most application process works are done during business hours (10 AM to 3 PM).

Score Source 1: Uniform distribution.

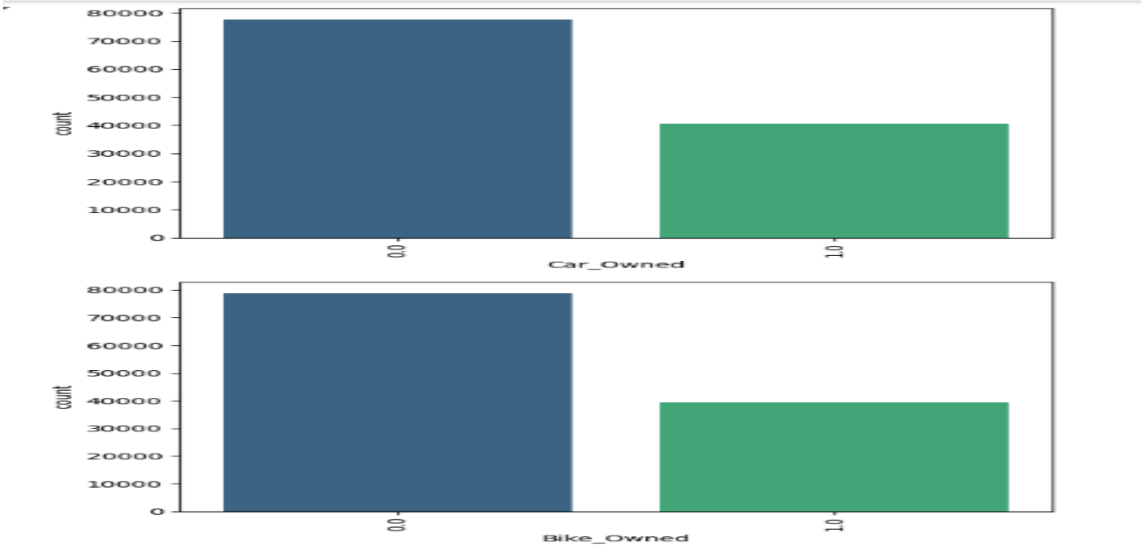
Score Source 2: Highly positively skewed after 0.

Score Source 3: Bell-shaped distribution between 0 and 1.

Social Circle Default: Right-skewed, higher values may indicate default risk based on social environment.

Phone Change: Right-skewed, frequent changes of phone could indicate instability.

CATEGORICAL COLUMN

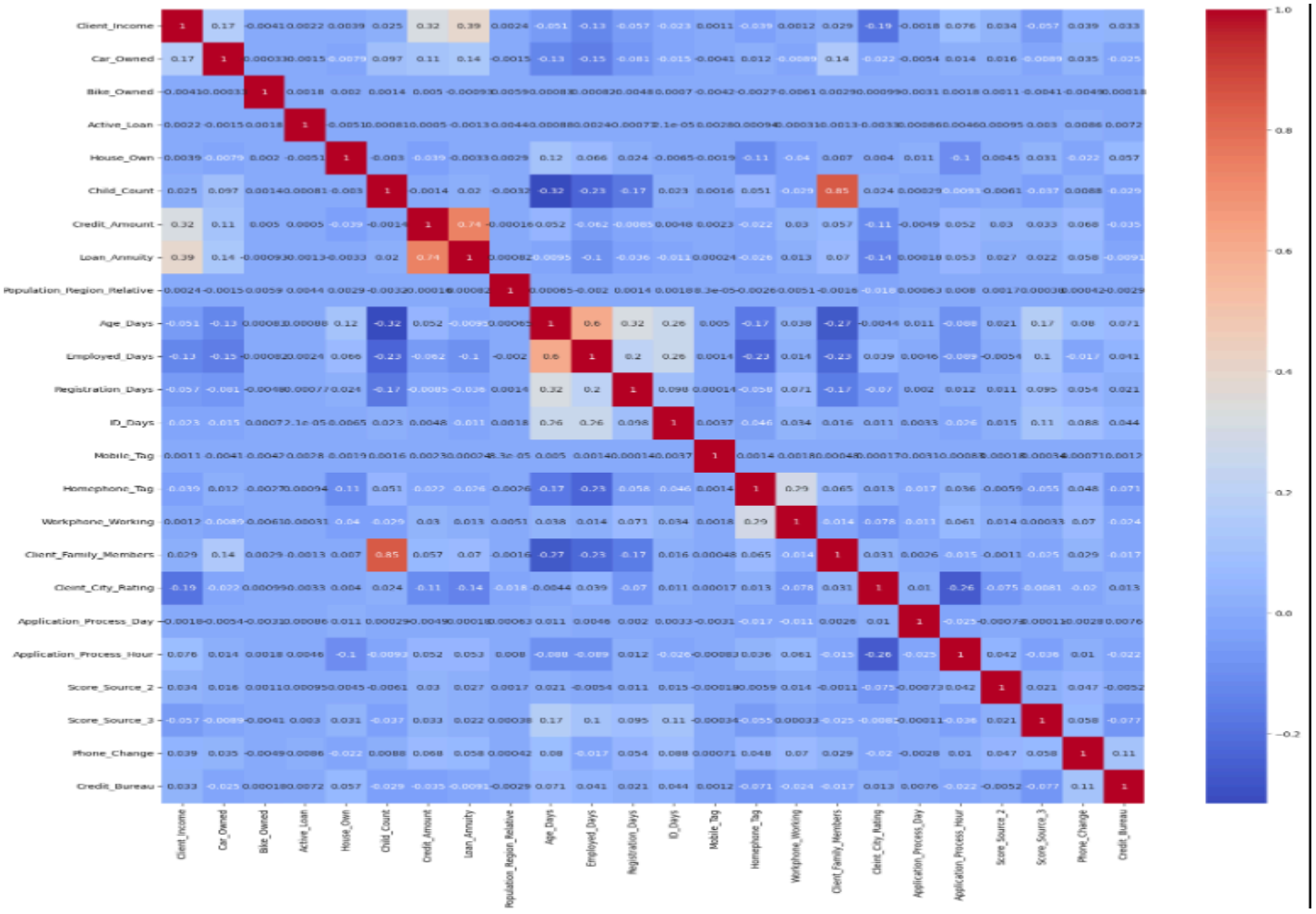


INFERENCES FROM COUNT PLOTS:

1. Car_Owned: Most of the clients do not own a car.
2. Bike_Owned: Majority of clients do not own a bike.

3. Active_Loan: Clients with and without active loans are fairly evenly distributed.
4. House_Own: Most of the clients own a house.
5. Accompany_Client: Mostly clients apply for loans alone, followed by those applying with relatives and then partners and so on.
6. Client_Income_Type: 'Service' and 'Commercial' clients form the largest groups followed by 'Retired' and 'Govt Job'.
7. Client_Education: Majority of clients have secondary education, followed by graduates and so on.
8. Client_Marital_Status: Married clients are the largest group, followed by singles and divorced and widow groups are almost similar.
9. Client_Gender: More male clients are there than female clients.
10. Loan_Contract_Type: Most loans are Consumer Loans (CL), fewer are Revolving Loans (RL).
11. Client_Housing_Type: Most clients live in their own homes.
12. Mobile_Tag: Almost all clients have registered mobile tags.
13. Homephone_Tag: Fewer clients have a registered home phone.
14. Workphone_Working: Fewer clients have provided their workplace contact.
15. Client_Occupation: Dominated by sales, labourers, and core staff followed by others.
16. Client_City_Rating: Most clients are from cities with a rating of '2', followed by '1' and '3'.
17. Application_Process_Day: Most clients processed their applications on weekdays rather than weekends and the highest applications were processed on 'Tuesday'.
18. Client_Permanent_Match_Tag: Most clients have a permanent match tag.
19. Client_Contact_Work_Tag: Majority of clients have a contact work tag.
20. Type_Organization: Majority of clients are 'Business Entity Type 3' followed by 'XNA' and 'Self-employed' and so on.
21. Default: Most of the clients have not defaulted their loan.

MULTIVARIATE ANALYSIS

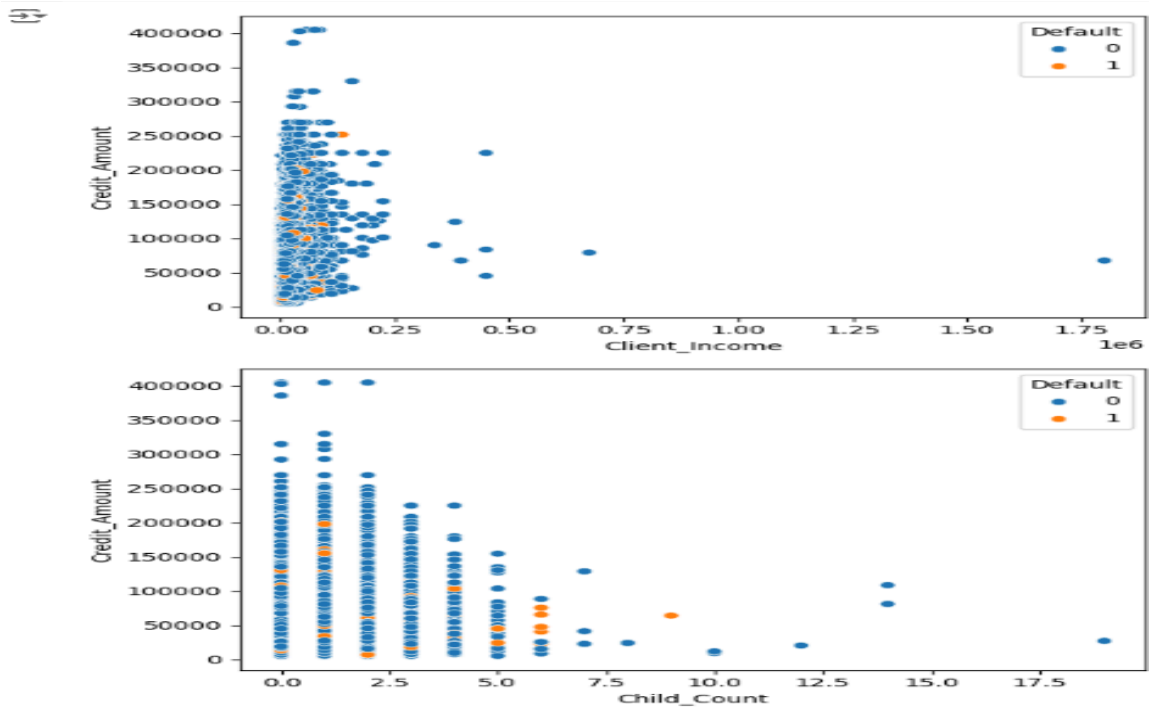


INFERENCES FROM HEATMAP:

1. Most of the columns have positive correlation.
2. Highly positive correlation is between Client Family Members with Child Count i.e., 0.85 followed by Loan Annuity with Credit Amount i.e., 0.74
3. Negative correlation is between Age Days and Child Count i.e., -0.32.

BIVARIATE ANALYSIS

NUMERICAL COLUMN



INFERENCES FROM SCATTERPLOT:

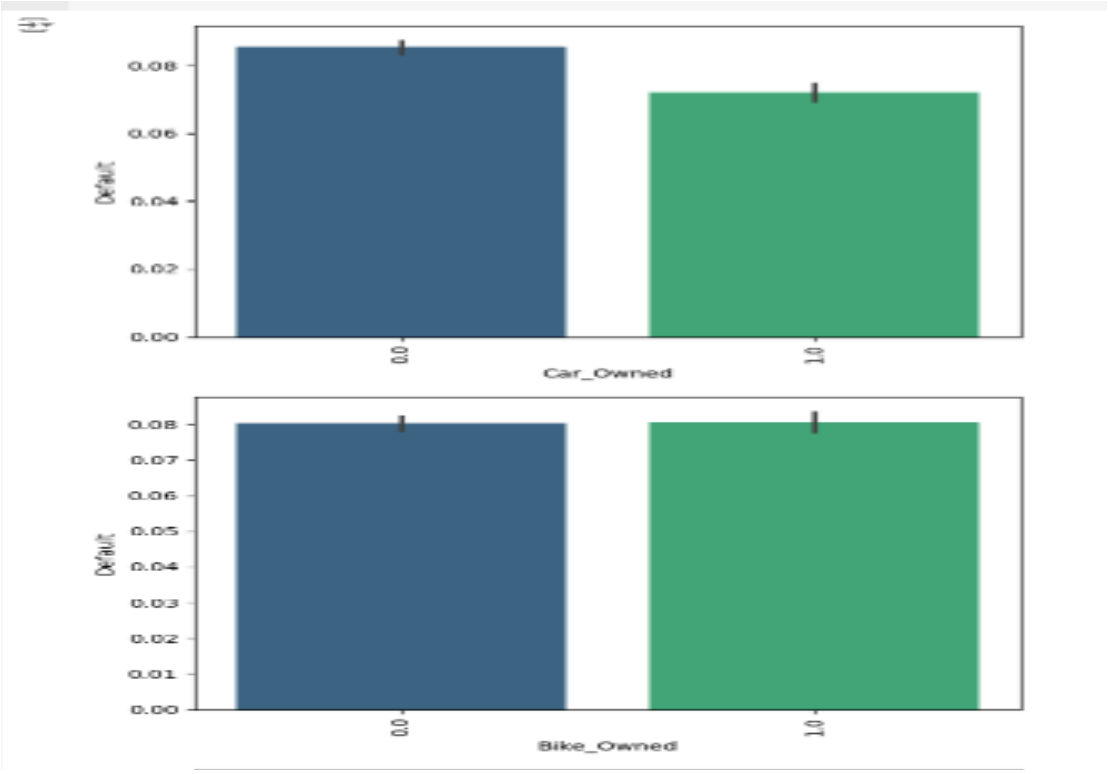
1. Client_Income: Mostly clients have an income between 0 and 0.25 and most of them have not defaulted on their loan. And some of them who defaulted are overlapping.
2. Child_Count: Mostly clients have less than 3 children and they have not defaulted their loan, very few of them defaulted their loan.
3. Loan_Annuity: Most of the clients have a good loan annuity and a good credit amount and have not defaulted loans, some of them have defaulted loans but it overlaps with the non-defaulter category.
4. Population_Region_Relative: Just one or two of the clients have defaulted on their loans.
5. Age_Days: Most clients have not defaulted loans whatever be the age but there are few of them inside the same category who defaulted also.
6. Employed_Days: In 0 employed days and more than 350000 employed days very few clients have defaulted on their loans.
7. Registration_Days: Some clients have defaulted loans but out of that most of them have not defaulted loans.
8. ID_Days: Unequal distribution of loan defaults with no loan default being more and defaulted loan being less and some outliers are also there.
9. Own_House_Age: Minority of clients have defaulted their loan.
10. Client_Family_Members: Mostly clients have not defaulted their loan but clients with 8 family members have defaulted loans.
11. Application_Process_Hour: Few clients have defaulted loans irrespective of their application process hour.
12. Score_Source_1: Most of the defaulted loan clients lie between 0.0 and 0.2 and in other categories there are less defaulted loans and mostly clients have not defaulted loans.
13. Score_Source_2: Minority of clients have defaulted their loans on 0 overlapped with the non-defaulter category at 0.
14. Score_Source_3: Maximum clients who defaulted loans lies between 0.0 and
15. 4 which is again overlapped by the non-defaulter category.
16. Social_Circle_Default: Majority of clients have not defaulted their loans although some clients have

defaulted also.

17. Phone_Change: Maximum clients who changed their phone have not defaulted loans.
18. Credit_Bureau: Some of the clients have defaulted loans with some outliers.

Overall, it's visible that whatever being the category maximum clients have not defaulted their loans.

CATEGORICAL COLUMN



INFERENCES FROM BARPLOT:

1. Car_Owned: Maximum clients who have no car before applying for loan are the defaulters.
2. Bike_Owned: Loan defaulter clients are almost equal who owns a bike and who don't.
3. Active_Loan: Loan defaulter clients are equal whether they have an active loan or not.
4. House_Own: Clients who own a house and who don't have almost an equal number of defaulters.
5. Accompany_Client: the category of accompanied client has a maximum number of defaulters followed by 'Others', 'Alone' and 'Partner'.
6. Client_Income_Type: Maximum number of loan defaulters are the 'Unemployed' one followed by 'Service' ones and no defaulters in 'Maternity leave' and 'Businessman' category.
7. Client_Education: Clients with 'Junior Secondary' education are the highest defaulter loans followed by 'Secondary' education ones.
8. Client_Marital_Status: 'Single' clients have maximum loan defaulters followed by 'Married' and 'Divorced' and then 'Widow'.
9. Client_Gender: 'Female' clients have more defaulted loans than 'Male' clients.
10. Loan_Contract_Type: Maximum number of defaulters are in the 'Cash Loan' category.
11. Client_Housing_Type: The 'Rental' one's have a maximum number of defaulters followed by 'Family' and then 'Shared' then others.
12. Mobile_Tag: Clients who provided numbers have a maximum number of defaulter loans.
13. Homephone_Tag: Clients who provided a home phone number have maximum loan defaulters than those who have not provided a phone number.
14. Workphone_Working: Clients whose work phone is not reachable have a greater number of defaulters than those whose work phone is reachable.
15. Client_Occupation: Clients who are 'Low Skill Labourers' have maximum defaulters than others.
16. Client_City_Rating: Clients whose city rating is 3 means best have maximum number of loan defaulters.
17. Application_Process_Day: Almost all days have an equal number of loan defaulters except 'Tuesday' and 'Wednesday' being the highest.
18. Client_Permanent_Match_Tag: Client permanent match tag of 'No' category has maximum number of

defaulters.

19. Client_Contact_Work_Tag: Client contact work tag of 'No' category has maximum defaulters.
20. Type_Organization: 'Industry: Type 8' and 'Trade: Type 5' have maximum defaulters than others.

05. STATISTICAL ANALYSIS

Chi Square Test

```
# Assuming 'Car_Owned' and 'Default' are the columns to be analyzed
# Ho : Car owned is not dependent on Default
# Ha : Car owned is dependent on Default

observed = pd.crosstab(df_new['Car_Owned'], df_new['Default'])

# Perform the Chi-Square test
chi2, p_value, dof, expected = stats.chi2_contingency(observed)

# Print the values
print(f' Chi-Square Statistics: {chi2}, p_value: {p_value}, dof: {dof}, expected: {expected}')

# Consider alpha as 0.05
alpha = 0.05

if p_value<0.05:
  print('Reject Ho')
else:
  print('Fail to reject Ho')
```

```
 Chi-Square Statistics: 61.909287385348996, p_value: 3.596492698790053e-15, dof: 1, expected: [[74736.19973575  6568.80026425]
 [37274.80026425  3276.19973575]]
Reject Ho
```

```
# Assuming 'Bike_Owned' and 'Default' are the columns to be analyzed
# Ho : Bike owned is not dependent on Default
# Ha : Bike owned is dependent on Default

observed = pd.crosstab(df_new['Bike_Owned'], df_new['Default'])

# Perform the Chi-Square test
chi2, p_value, dof, expected = stats.chi2_contingency(observed)

# Print the values
print(f' Chi-Square Statistics: {chi2}, p_value: {p_value}, dof: {dof}, expected: {expected}')

# Consider alpha as 0.05
alpha = 0.05

if p_value<0.05:
  print('Reject Ho')
else:
  print('Fail to reject Ho')
```

```
 Chi-Square Statistics: 0.020308091977580954, p_value: 0.8866799747906008, dof: 1, expected: [[75900.83616728  6671.16383272]
 [36110.16383272  3173.83616728]]
Fail to reject Ho
```

In Similar manner,after doing all hypothesis testing these are the inferences:

1. Bike Owned is not dependent on default.
2. Active Loan is not dependent on default.
3. House Own is not dependent on default.
4. Mobile Tag is not dependent on default.
5. Application Process Day is not dependent on default.
6. Rest all other columns like Car Owned, Accompany Client, Client Income Type, Client Education, Client Marital Status, Client Gender, Loan Contract Type, Client Housing Type, Home Phone Tag, Work Phone Working, Client Occupation, CLient City Rating, Client Permanent Match Tag, Client Contact Work Tag and Type Organization are dependent on default.

06. ENCODING

Encoding

```
[ ] # n-1 Dummy Encoding

    object_encoded = pd.get_dummies(object_df, drop_first=True, dtype=int)
    object_encoded.head(2)
```

ID	Car_Owned	Bike_Owned	Active_Loan	House_Own	Mobile_Tag	Homephone_Tag	Workphone_Working	Cleint_City_Rating	Application_Process_Day	Default	Accompany_Client_Alone	Accompany_Client_Group	Accompany_Client_Kids	Accompa
12142509	0.0	0.0	1.0	0.0	1	1	0	2.0	6.0	0	1	0	0	
12138936	1.0	0.0	1.0	NaN	1	0	1	2.0	3.0	0	1	0	0	

N-1 Dummy encoding is a technique used in machine learning and data analysis to convert categorical variables into numerical representations that can be understood by algorithms. It involves creating new binary columns (also known as dummy variables) for each category within a categorical variable. Each dummy variable represents the presence or absence of a specific category.

ABSTRACT

The project focuses on addressing the challenge of imbalanced class prediction in a machine learning model, particularly where the minority class ("1") is completely missed. To enhance recall for the minority class and improve overall model performance, fine-tuning techniques were explored. Key approaches included resampling techniques such as oversampling class "1" or undersampling class "0" to balance the class distribution, and adjusting class weights in the logistic regression model to prioritize recall for the minority class. Additionally, the study considered employing more complex models, such as decision trees, random forests, and boosting algorithms, to better capture patterns of the minority class. Threshold tuning was also investigated to increase model sensitivity for class "1." These methods collectively aim to mitigate the impact of class imbalance, ensuring a more robust and equitable predictive performance.

07. MODEL BUILDING

LOGISTIC REGRESSION

Based on the classification report and confusion matrix shown, we can infer the following about the base logistic regression model's performance:

1. High Accuracy: The model has a high accuracy of 91.96%. However, accuracy alone can be misleading, especially when classes are imbalanced.
2. Class Imbalance: There is a significant imbalance in the prediction results, as indicated by the confusion matrix. The model correctly identified all instances of class "0" (with 22,413 true negatives), but failed to predict any instances of class "1," leading to 1,959 false negatives.
3. Recall for Class "1": The recall score for class "1" is 0.0, which is concerning. Given that high recall is critical for this context, this result suggests the model is not performing as required for class "1" and fails to detect any positive instances.
4. Precision and F1-Score for Class "1": Both precision and F1-score for class "1" are also 0.0, indicating the model's inability to capture this class effectively. This affects the overall F1-score and macro average metrics, which are low, suggesting the model's performance across both classes is imbalanced.
5. Weighted vs. Macro Averages: The weighted average metrics are relatively higher than the macro averages because class "0" dominates the dataset. The macro average is lower, reflecting poor performance for class "1."
6. Need for Fine-Tuning: Since the model completely misses class "1," fine-tuning is necessary to improve recall. Possible approaches include:
    - Resampling Techniques: Applying oversampling for class "1" or undersampling for class "0" to balance the classes.
    - Adjusting Class Weights: Increasing the weight for class "1" in the logistic regression model to penalize false negatives more heavily.
    - Exploring Other Models: Trying more complex models like decision trees, random forests, or boosting algorithms, which may better capture minority class patterns.
    - Threshold Tuning: Experimenting with threshold adjustments to increase sensitivity for class "1."

These steps could help achieve better recall for the minority class and improve overall model performance.

RANDOM FOREST MODEL

RANDOM FOREST 1

Inference from Current Model Performance

Based on the classification report and confusion matrix, we can infer the following about the Random Forest model's performance:

Training Data: Accuracy, Precision, Recall, and F1-Score are nearly perfect (~1.0).

- **Confusion Matrix:** Class 0: 89,598 instances, all classified correctly. Class 1: 7,886 instances, with only 1 misclassified. This indicates that the model has almost perfectly fit the training data, which is a sign of overfitting.

Testing Data:

- **Accuracy:** 92.93% (good overall performance).
- **Precision:** Weighted Precision is 93.43%, indicating that predictions for both classes are relatively reliable overall. Precision for class 1 is high (1.0), meaning the model is confident when it predicts class 1, but...
- **Recall:** Weighted Recall is 92.93%, but for class 1, it is only 12%. Out of 1,959 true instances of class 1, the model identifies only 235 correctly, missing the majority (1,724 instances are misclassified as class 0).
- **F1-Score:** For class 1, it is 0.21, which is very low, indicating poor performance in detecting minority class instances.
- **Confusion Matrix:** Class 0 (majority class): 22,413 instances, all correctly classified. Class 1 (minority class): 1,959 instances, with only 235 correctly classified.

The model has overfitted the training data and We will address the overfitting by performing hyperparameter tuning using techniques like GridSearchCV or RandomizedSearchCV. This will help optimize parameters such as max_depth, n_estimators, min_samples_split, and min_samples_leaf to improve generalization.

CLASS IMBALANCE RESOLUTION

Since we know the Target variable is highly imbalanced and our model is also overfitting. Academically we go for SMOTE but considering the real-life like scenarios creating synthetic data points for our purpose fabricates the true nature of the dataset, if SMOTE is used.

So, we decided to go with mentioning class **weights** or **scale_pos_weight** parameters while creating boosting models.

BOOSTING MODELS

Observations:

- Models like XGBoost, LightGBM, and CatBoost show better handling of the minority class compared to Gradient Boosting and AdaBoost.
- CatBoost achieves the best ROC-AUC score and is the most promising model for recall improvement.

Next Steps:

- Prioritize CatBoost, XGBoost, and LightGBM for further optimization since they perform best on minority class recall.
- Explore stacking and custom thresholds to boost recall further.

XGBoost and LightGBM:

- Fine-tune scale_pos_weight and learning rate.
- Experiment with deeper trees (max_depth) and more iterations (n_estimators).

CatBoost:

- Further tweak (class_weights) and (learning rate).
- Use CatBoost support for categorical features directly if applicable.

MODEL TUNING

LOGISTIC REGRESSION

Logistic Regression

```
[ ] model_validation(LogisticRegression(class_weight=weights_dict), x_train, y_train, x_test, y_test)
```

```
Confusion Matrix:
 [[13303  9110]
 [  923  1036]]

Classifaction report:
              precision    recall  f1-score   support

           0       0.94      0.59      0.73     22413
           1       0.10      0.53      0.17      1959

    accuracy                           0.59     24372
   macro avg       0.52      0.56      0.45     24372
weighted avg       0.87      0.59      0.68     24372

Do you want to save the result? Y/Ny
```

```
[ ] scorecard
```

	Model	Accuracy	Precesion	Recall	F1 Score	Cohen Kappa	ROC AUC
0	LogisticRegression(class_weight={0: 0.5440076787428291, 1: 6.180826781638347})	0.588339	0.102109	0.528841	0.171169	0.042099	0.581329

Interpretation:

- **Precision:** The model correctly predicts cases 10% of the time. This means that when the model predicts a customer will be default, it is accurate 10% of the time.
- **Recall:** The model correctly identifies 53% of all the actual positive cases. This means that out of all the customers who actually defaulted, the model correctly predicted 53% of them.
- **F1-score:** The F1-score is moderate for class 1, indicating a balance between precision and recall.
- **Accuracy:** The model correctly predicts 59% of all cases, regardless of whether they are positive or negative. This is a moderate overall accuracy score.
- **Macro Avg:** This is the average of precision, recall, and F1-score across all classes. It gives a sense of the model's performance in each class.
- **Weighted Avg:** This is a weighted average of precision, recall, and F1-score, taking into account the class distribution. It gives a sense of the overall performance of the model, considering the class imbalance.

Overall, the logistic regression model seems to be performing reasonably well. It has good accuracy and recall, but the precision is relatively low. This means that the model might miss some actual positive cases.

DECISION TREE

Decision Tree

```
[ ] from sklearn.tree import DecisionTreeClassifier
```

```
[ ] best_dt = tuning_parameters(grid= {'max_depth' : [5,8,10]}, estimator=DecisionTreeClassifier(class_weight=weights_dict)
                                , x=x_train, y=y_train)
```

```
Fitting 5 folds for each of 3 candidates, totalling 15 fits
```

```
model_validation(DecisionTreeClassifier(**best_dt, class_weight=weights_dict), x_train, y_train, x_test, y_test)
```

```
Confusion Matrix:
 [[13721  8692]
 [  644  1315]]

Classifaction report:
              precision    recall  f1-score   support

           0       0.96      0.61      0.75     22413
           1       0.13      0.67      0.22      1959

    accuracy                           0.62     24372
   macro avg       0.54      0.64      0.48     24372
weighted avg       0.89      0.62      0.70     24372

Do you want to save the result? Y/Ny
```

```
scorecard
```

	Model	Accuracy	Precesion	Recall	F1 Score	Cohen Kappa	ROC AUC
0	LogisticRegression(class_weight={0: 0.5440076787428291, 1: 6.180826781638347})	0.588339	0.102109	0.528841	0.171169	0.042099	0.581329
1	DecisionTreeClassifier(class_weight={0: 0.5440076787428291,\n 1: 6.180826781638347},\n max_depth=5)	0.616937	0.131408	0.671261	0.219789	0.098606	0.690881

Interpretation:

- **Precision:** The model correctly predicts positive cases 13% of the time. This means that when the model predicts a customer will be default, it is accurate 13% of the time.
- **Recall:** The model correctly identifies 67% of all the actual positive cases. This means that out of all the customers who actually defaulted, the model correctly predicted 67% of them.
- **F1-score:** This is the harmonic mean of precision and recall. It provides a balanced measure of the model's performance. A higher F1-score indicates better overall performance. In this case, the F1-score is 0.22, which is relatively low.
- **Accuracy:** The model correctly predicts 62% of all cases, regardless of whether they are positive or negative. This is a moderate accuracy score.
- **Macro Avg:** This is the average of precision, recall, and F1-score across all classes. It gives a sense of the model's performance in each class.
- **Weighted Avg:** This is a weighted average of precision, recall, and F1-score, taking into account the class distribution. It gives a sense of the overall performance of the model, considering the class imbalance.

Overall, the decision tree model seems to have a low precision but a relatively high recall. This suggests that the model might be prone to false positives, but it is good at identifying actual positive cases.

RANDOM FOREST CLASSIFIER

```
[ ] model_validation(RandomForestClassifier(**best_rf, max_features=None), x_train, y_train, x_test, y_test)

Confusion Matrix:
 [[22413     0]
 [ 1958     1]]

Classifaction report:
              precision    recall  f1-score   support

           0       0.92      1.00      0.96     22413
           1       1.00      0.00      0.00      1959

    accuracy                           0.92     24372
   macro avg       0.96      0.50      0.48     24372
weighted avg       0.93      0.92      0.88     24372

Do you want to save the result? Y/Ny
```

```
scorecard
```

	Model	Accuracy	Precesion	Recall	F1 Score	Cohen Kappa	ROC AUC
0	LogisticRegression(class_weight={0: 0.5440076787428291, 1: 6.180826781638347})	0.588339	0.102109	0.528841	0.171169	0.042099	0.581329
1	DecisionTreeClassifier(class_weight={0: 0.5440076787428291,\n 1: 6.180826781638347},\n max_depth=5)	0.616937	0.131408	0.671261	0.219789	0.098606	0.690881
2	RandomForestClassifier(max_depth=5, max_features=None, n_estimators=150)	0.919621	0.000000	0.000000	0.000000	0.000000	0.704694

Interpretation:

- **Precision:** The model correctly predicts positive cases 1% of the time. This means that when the model predicts a customer will be default, it is accurate 1% of the time.
- **Recall:** The model correctly identifies 0% of all the actual positive cases. This means that out of all the customers who actually defaulted, the model correctly predicted 0% of them.
- **F1-score:** This is the harmonic mean of precision and recall. It provides a balanced measure of the model's performance. A higher F1-score indicates better overall performance. In this case, the F1-score is 0.0.
- **Accuracy:** The model correctly predicts 92% of all cases, regardless of whether they are positive or negative. This is a very good overall accuracy score.
- **Macro Avg:** This is the average of precision, recall, and F1-score across all classes. It gives a sense of the model's performance in each class.
- **Weighted Avg:** This is a weighted average of precision, recall, and F1-score, taking into account the class distribution. It gives a sense of the overall performance of the model, considering the class imbalance.

Overall, the Random Forest model seems to be performing moderate. It has low precision, recall, F1-score, and accuracy, indicating that it is not accurate and reliable in predicting customer defaults.

ADABOOST MODEL

```
[ ] best_ada = tuning_parameters(grid={'n_estimators' : [100,150,200], 'learning_rate' :[0.1, 0.2]},
                                 estimator = AdaBoostClassifier(estimator=DecisionTreeClassifier(class_weight=weights_dict)),
                                 x=x_train, y=y_train)

Fitting 5 folds for each of 6 candidates, totalling 30 fits

[ ] best_ada

{'learning_rate': 0.2, 'n_estimators': 200}

model_validation(AdaBoostClassifier(**best_ada, estimator=DecisionTreeClassifier(class_weight=weights_dict)), x_train, y_train,
                 x_test, y_test)

Confusion Matrix:
 [[20841  1572]
 [ 1441   518]]

Classifaction report:
              precision    recall  f1-score   support

           0       0.94      0.93      0.93     22413
           1       0.25      0.26      0.26      1959

    accuracy                           0.88     24372
   macro avg       0.59      0.60      0.59     24372
weighted avg       0.88      0.88      0.88     24372

Do you want to save the result? Y/Ny

scorecard
```

	Model	Accuracy	Precesion	Recall	F1 Score	Cohen Kappa	ROC AUC
0	AdaBoostClassifier(estimator=DecisionTreeClassifier(class_weight={0: 0.5440076787428291,\n 1: 6.180826781638347}),\n learning_rate=0.2, n_estimators=200)	0.876375	0.247847	0.264421	0.255866	0.18853	0.597141

Interpretation:

- **Precision:** The model correctly predicts positive cases 25% of the time. This means that when the model predicts a customer will be default, it is accurate 25% of the time.
- **Recall:** The model correctly identifies 26% of all the actual positive cases. This means that out of all the customers who actually defaulted, the model correctly predicted 26% of them.
- **F1-score:** This is the harmonic mean of precision and recall. It provides a balanced measure of the model's performance. A higher F1-score indicates better overall performance. In this case, the F1-score is 0.26, which is very moderate.
- **Accuracy:** The model correctly predicts 88% of all cases, regardless of whether they are positive or negative. This is a very good overall accuracy score.
- **Macro Avg:** This is the average of precision, recall, and F1-score across all classes. It gives a sense of the model's performance in each class.
- **Weighted Avg:** This is a weighted average of precision, recall, and F1-score, taking into account the class distribution. It gives a sense of the overall performance of the model, considering the class imbalance.

Overall, the AdaBoost model seems to be performing very well. It has low high precision, recall, F1-score, and accuracy, indicating that it is less accurate and reliable in predicting customer defaults.

XGBoost Model

```
[ ] best_xgb = tuning_parameters(grid={'n_estimators' : [100,150,200], 'learning_rate' :[0.1, 0.2], 'gamma' :[1,2,3]},
                                 estimator=XGBClassifier(), x=x_train, y=y_train)
    best_xgb

Fitting 5 folds for each of 18 candidates, totalling 90 fits
{'gamma': 1, 'learning_rate': 0.2, 'n_estimators': 150}

[ ] model_validation(XGBClassifier(**best_xgb), x_train, y_train, x_test, y_test)

Confusion Matrix:
 [[22363    50]
 [ 1897    62]]

Classifaction report:
              precision    recall  f1-score   support

           0       0.92      1.00      0.96     22413
           1       0.55      0.03      0.06      1959

    accuracy                           0.92     24372
   macro avg       0.74      0.51      0.51     24372
weighted avg       0.89      0.92      0.89     24372

Do you want to save the result? Y/Ny

scorecard
```

	Model	Accuracy	Precesion	Recall	F1 Score	Cohen Kappa	ROC AUC
0	AdaBoostClassifier(estimator=DecisionTreeClassifier(class_weight={0: 0.5440076787428291,\n 1: 6.180826781638347}),\n learning_rate=0.2, n_estimators=200)	0.876375	0.247847	0.264421	0.255866	0.188530	0.597141
1	XGBClassifier(base_score=None, booster=None, callbacks=None,\n colsample_bylevel=None, colsample_bynode=None,\n colsample_bytree=None, device=None, early_stopping_rounds=None,\n enable_categorical=False, eval_metric=None, feature_types=None,\n gamma=1, grow_policy=None, importance_type=None,\n interaction_constraints=None, learning_rate=0.2, max_bin=None,\n max_cat_threshold=None, max_cat_to_onehot=None,\n max_delta_step=None, max_depth=None, max_leaves=None,\n min_child_weight=None, missing=nan, monotone_constraints=None,\n multi_strategy=None, n_estimators=150, n_jobs=None,\n num_parallel_tree=None, random_state=None, ...)	0.920113	0.553571	0.031649	0.059874	0.051629	0.755579

Interpretation:

- **Precision:** The model correctly predicts positive cases 55% of the time. This means that when the model predicts a customer will be default, it is accurate 55% of the time.
- **Recall:** The model correctly identifies 3% of all the actual positive cases. This means that out of all the customers who actually defaulted, the model correctly predicted 3% of them.
- **F1-score:** This is the harmonic mean of precision and recall. It provides a balanced measure of the model's performance. A higher F1-score indicates better overall performance. In this case, the F1-score is 0.06, which is very good.
- **Accuracy:** The model correctly predicts 93% of all cases, regardless of whether they are positive or negative. This is a very good overall accuracy score.
- **Macro Avg:** This is the average of precision, recall, and F1-score across all classes. It gives a sense of the model's performance in each class.
- **Weighted Avg:** This is a weighted average of precision, recall, and F1-score, taking into account the class distribution. It gives a sense of the overall performance of the model, considering the class imbalance.

Overall, the XGBoost model seems to be performing moderate. It has high precision, less recall and F1-score, and high accuracy, indicating that it is less accurate and reliable in predicting customer defaults.

LightGBM Model

```
Classifaction report:
              precision    recall  f1-score   support

           0       0.96      0.72      0.82     22413
           1       0.17      0.64      0.26      1959

    accuracy                           0.71     24372
   macro avg       0.56      0.68      0.54     24372
weighted avg       0.89      0.71      0.78     24372

Do you want to save the result? Y/Ny
```

```
[ ] scorecard
```

	Model	Accuracy	Precesion	Recall	F1 Score	Cohen Kappa	ROC AUC
0	AdaBoostClassifier(estimator=DecisionTreeClassifier(class_weight={0: 0.5440076787428291,\n 1: 6.180826781638347}),\n learning_rate=0.2, n_estimators=200)	0.876375	0.247847	0.264421	0.255866	0.188530	0.597141
1	XGBClassifier(base_score=None, booster=None, callbacks=None,\n colsample_bylevel=None, colsample_bynode=None,\n colsample_bytree=None, device=None, early_stopping_rounds=None,\n enable_categorical=False, eval_metric=None, feature_types=None,\n gamma=1, grow_policy=None, importance_type=None,\n interaction_constraints=None, learning_rate=0.2, max_bin=None,\n max_cat_threshold=None, max_cat_to_onehot=None,\n max_delta_step=None, max_depth=None, max_leaves=None,\n min_child_weight=None, missing=nan, monotone_constraints=None,\n multi_strategy=None, n_estimators=150, n_jobs=None,\n num_parallel_tree=None, random_state=None, ...)	0.920113	0.553571	0.031649	0.059874	0.051629	0.755579
2	LGBMClassifier(class_weight={0: 0.5440076787428291, 1: 6.180826781638347},\n max_depth=5, num_leaves=28)	0.713852	0.166777	0.640633	0.264656	0.157149	0.745606

Interpretation:

- **Precision:** The model correctly predicts positive cases 17% of the time. This means that when the model predicts a customer will default, it is accurate 17% of the time.
- **Recall:** The model correctly identifies 64% of all the actual positive cases. This means that out of all the customers who actually defaulted, the model correctly predicted 64% of them.
- **F1-score:** This is the harmonic mean of precision and recall. It provides a balanced measure of the model's performance. A higher F1-score indicates better overall performance. In this case, the F1-score is 0.26, which is very good.
- **Accuracy:** The model correctly predicts 71% of all cases, regardless of whether they are positive or negative. This is a very good overall accuracy score.
- **Macro Avg:** This is the average of precision, recall, and F1-score across all classes. It gives a sense of the model's performance in each class.
- **Weighted Avg:** This is a weighted average of precision, recall, and F1-score, taking into account the class distribution. It gives a sense of the overall performance of the model, considering the class imbalance.

Overall, the LightGBM model seems to be performing very well, with high recall and accuracy. However, the precision is slightly lower, indicating that the model might miss some actual positive cases.

CatBoost Model

```
[ ] scorecard
```

	Model	Accuracy	Precesion	Recall	F1 Score	Cohen Kappa	ROC AUC
0	AdaBoostClassifier(estimator=DecisionTreeClassifier(class_weight={0: 0.5440076787428291,\n 1: 6.180826781638347}),\n learning_rate=0.2, n_estimators=200)	0.876375	0.247847	0.264421	0.255866	0.188530	0.597141
1	XGBClassifier(base_score=None, booster=None, callbacks=None,\n colsample_bylevel=None, colsample_bynode=None,\n colsample_bytree=None, device=None, early_stopping_rounds=None,\n enable_categorical=False, eval_metric=None, feature_types=None,\n gamma=1, grow_policy=None, importance_type=None,\n interaction_constraints=None, learning_rate=0.2, max_bin=None,\n max_cat_threshold=None, max_cat_to_onehot=None,\n max_delta_step=None, max_depth=None, max_leaves=None,\n min_child_weight=None, missing=nan, monotone_constraints=None,\n multi_strategy=None, n_estimators=150, n_jobs=None,\n num_parallel_tree=None, random_state=None, ...)	0.920113	0.553571	0.031649	0.059874	0.051629	0.755579
2	LGBMClassifier(class_weight={0: 0.5440076787428291, 1: 6.180826781638347},\n max_depth=5, num_leaves=28)	0.713852	0.166777	0.640633	0.264656	0.157149	0.745606
3	<catboost.core.CatBoostClassifier object at 0x7d4d38b8a680>	0.879000	0.309615	0.410924	0.353148	0.287858	0.752419

- **Precision**: The model correctly predicts positive cases 30% of the time. This means that when the model predicts a customer will default, it is accurate 30% of the time.
- **Recall**: The model correctly identifies 41% of all the actual positive cases. This means that out of all the customers who actually defaulted, the model correctly predicted 41% of them.
- **F1-score**: This is the harmonic mean of precision and recall. It provides a balanced measure of the model's performance. A higher F1-score indicates better overall performance. In this case, the F1-score is 0.35, which is moderate.
- **Accuracy**: The model correctly predicts 87% of all cases, regardless of whether they are positive or negative. This is a very good overall accuracy score.
- **Macro Avg**: This is the average of precision, recall, and F1-score across all classes. It gives a sense of the model's performance in each class.
- **Weighted Avg**: This is a weighted average of precision, recall, and F1-score, taking into account the class distribution. It gives a sense of the overall performance of the model, considering the class imbalance.

Overall, the CatBoost model seems to be performing moderate in terms of precision and accuracy.

GRADIENT BOOSTING MODEL

GradientBoosting Model

```
[ ] from sklearn.ensemble import GradientBoostingClassifier

[ ] best_gbm = tuning_parameters(grid={'n_estimators' : [100,150,200], 'learning_rate' : [0.1, 0.2], 'max_depth' :[5,10]},
                                 estimator=GradientBoostingClassifier(), x=x_train, y=y_train)
    best_gbm

Fitting 5 folds for each of 12 candidates, totalling 60 fits
{'learning_rate': 0.2, 'max_depth': 10, 'n_estimators': 200}

[ ] model_validation(GradientBoostingClassifier(**best_gbm), x_train, y_train, x_test, y_test)

Confusion Matrix:
 [[22138   275]
 [ 1593   366]]

Classifaction report:
              precision    recall  f1-score   support

           0       0.93      0.99      0.96     22413
           1       0.57      0.19      0.28      1959

    accuracy                           0.92     24372
   macro avg       0.75      0.59      0.62     24372
weighted avg       0.90      0.92      0.91     24372

Do you want to save the result? Y/Ny

scorecard
```

	Model	Accuracy	Precesion	Recall	F1 Score	Cohen Kappa	ROC AUC
0	GradientBoostingClassifier(learning_rate=0.2, max_depth=10, n_estimators=200)	0.923355	0.570983	0.18683	0.281538	0.251888	0.765534

Interpretation:

- **Precision:** The model correctly predicts positive cases 57% of the time. This means that when the model predicts a customer will default, it is accurate 57% of the time.
- **Recall:** The model correctly identifies 19% of all the actual positive cases. This means that out of all the

customers who actually defaulted, the model correctly predicted 19% of them.

- **F1-score:** This is the harmonic mean of precision and recall. It provides a balanced measure of the model's performance. A higher F1-score indicates better overall performance. In this case, the F1-score is 0.28, which is very good.
- **Accuracy:** The model correctly predicts 92% of all cases, regardless of whether they are positive or negative. This is a very good overall accuracy score.
- **Macro Avg:** This is the average of precision, recall, and F1-score across all classes. It gives a sense of the model's performance in each class.
- **Weighted Avg:** This is a weighted average of precision, recall, and F1-score, taking into account the class distribution. It gives a sense of the overall performance of the model, considering the class imbalance.

Overall, the Gradient Boosting model seems to be performing moderate. It has high precision, less recall, F1-score, and accuracy, indicating that it is less accurate and reliable in predicting customer defaults.

LIMITATIONS

1. **Class Imbalance Issue**: Most models show low precision, recall, or F1-scores for predicting defaults (positive cases), which indicates that the class imbalance in the dataset has impacted their ability to correctly classify minority cases.
2. **Low Precision for Positive Class**: Many models, such as Logistic Regression, Decision Tree, and Random Forest, demonstrate low precision, meaning that a high proportion of predicted positive cases are false positives.
3. **Low Recall for Positive Class**: Models like Random Forest and XGBoost have particularly low recall, indicating that they fail to identify a significant number of actual default cases.
4. **Overfitting or Bias**: Models such as Random Forest, XGBoost, and Gradient Boosting have high accuracy scores but perform poorly in terms of recall and F1-score for the positive class, suggesting potential overfitting or bias toward the majority class.
5. **Model-Specific Weaknesses**:
    - Random Forest failed to identify any positive cases (0% recall).
    - AdaBoost and Logistic Regression have moderate F1-scores but still struggle with precision.
6. **Trade-Off Between Precision and Recall**: Models like LightGBM and Gradient Boosting achieve good recall but lower precision, reflecting a trade-off that might lead to many false positives.

0.8 CONCLUSION

Among all the models evaluated, LightGBM stands out as the best-performing model for predicting customer defaults. It achieves a good balance between recall (64%) and accuracy (71%), making it effective in identifying most default cases while maintaining a reasonable overall classification performance. Additionally, LightGBM outperforms other models in handling the trade-off between false positives and false negatives, which is critical in imbalanced datasets like this one.

While its precision (17%) is relatively low, this can be addressed with techniques like oversampling, undersampling, or cost-sensitive learning to further improve the model’s ability to correctly classify positive cases. Overall, LightGBM is a robust and reliable model compared to others in this analysis, demonstrating its suitability for this predictive task with further refinement and optimization.