CAR LOAN DEFAULTER

Presentation by Group: 04

Khushi Agrawal

Eksimar Kaur

Diksha Yadav

Tejaswini Pathak

Ankita Roy

THE DATASET

The Data Description

```
[ ] 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_Type	Client_Education
0	12142509	6750	0.0	0.0	1.0	0.0	0.0	61190.55	3416.85	Alone	Commercial	Secondary
1	12138936	20250	1.0	0.0	1.0	NaN	0.0	15282	1826.55	Alone	Service	Graduation
2	12181264	18000	0.0	0.0	1.0	0.0	1.0	59527.35	2788.2	Alone	Service	Graduation dropout
3	12188929	15750	0.0	0.0	1.0	1.0	0.0	53870.4	2295.45	Alone	Retired	Secondary
4	12133385	33750	1.0	0.0	1.0	0.0	2.0	133988.4	3547.35	Alone	Commercial	Secondary

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

The dataset contains
121,856 rows and 40 columns, with a mix of numerical and categorical data, and some columns with missing values.

```
 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
 10  Client_Income_Type          118155 non-null  object
 11  Client_Education            118211 non-null  object
 12  Client_Marital_Status       118383 non-null  object
 13  Client_Gender               119443 non-null  object
 14  Loan_Contract_Type          118205 non-null  object
 15  Client_Housing_Type         118169 non-null  object
 16  Population_Region_Relative  116999 non-null  object
 17  Age_Days                    118256 non-null  object
 18  Employed_Days               118207 non-null  object
 19  Registration_Days           118242 non-null  object
 20  ID_Days                     115888 non-null  object
 21  Own_House_Age               41761 non-null   float64
 22  Mobile_Tag                  121856 non-null  int64
 23  Homephone_Tag               121856 non-null  int64
 24  Workphone_Working           121856 non-null  int64
 25  Client_Occupation           80421 non-null   object
 26  Client_Family_Members       119446 non-null  float64
 27  Cleint_City_Rating          119447 non-null  float64
 28  Application_Process_Day     119428 non-null  float64
 29  Application_Process_Hour    118193 non-null  float64
 30  Client_Permanent_Match_Tag  121856 non-null  object
 31  Client_Contact_Work_Tag     121856 non-null  object
 32  Type_Organization           118247 non-null  object
 33  Score_Source_1              53021 non-null   float64
 34  Score_Source_2              116170 non-null  float64
 35  Score_Source_3              94935 non-null   object
 36  Social_Circle_Default       59928 non-null   float64
 37  Phone_Change                118192 non-null  float64
 38  Credit_Bureau               103316 non-null  float64
 39  Default                     121856 non-null  int64
dtypes: float64(15), int64(5), object(20)
memory usage: 37.2+ MB
```

Data Cleaning

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')
```

Impute missing values in category or object dtype columns with the most frequent value (mode)

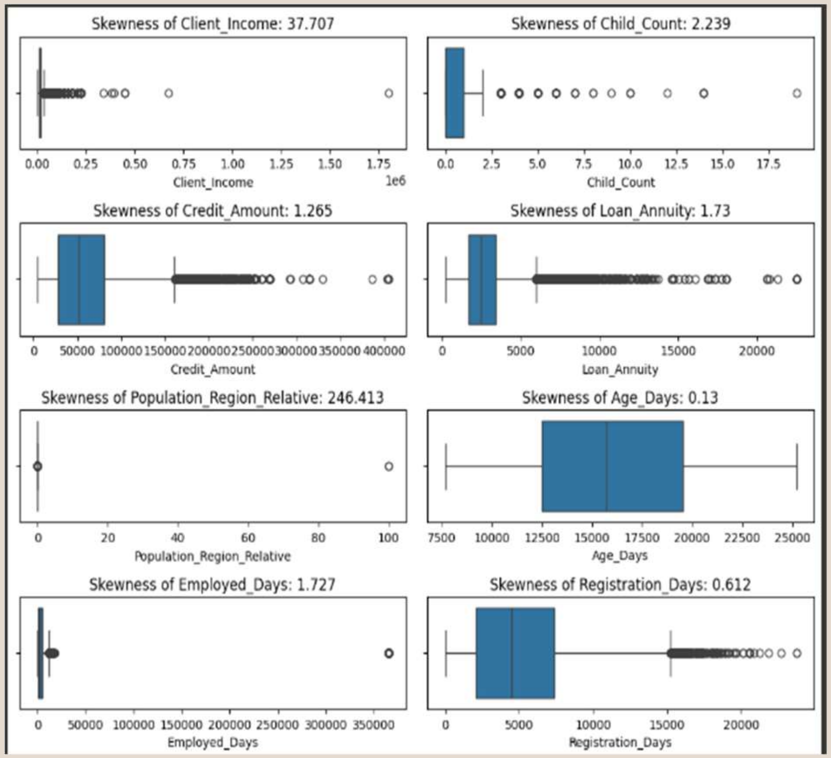
```
for col in object_df.columns:
    if col in df_new.columns:  # Check if column was retained after dropping
        df_new[col].fillna(df_new[col].mode()[0], inplace=True)
print("\nDataFrame after imputing object columns with mode:\n")
df_new.head(1)
```

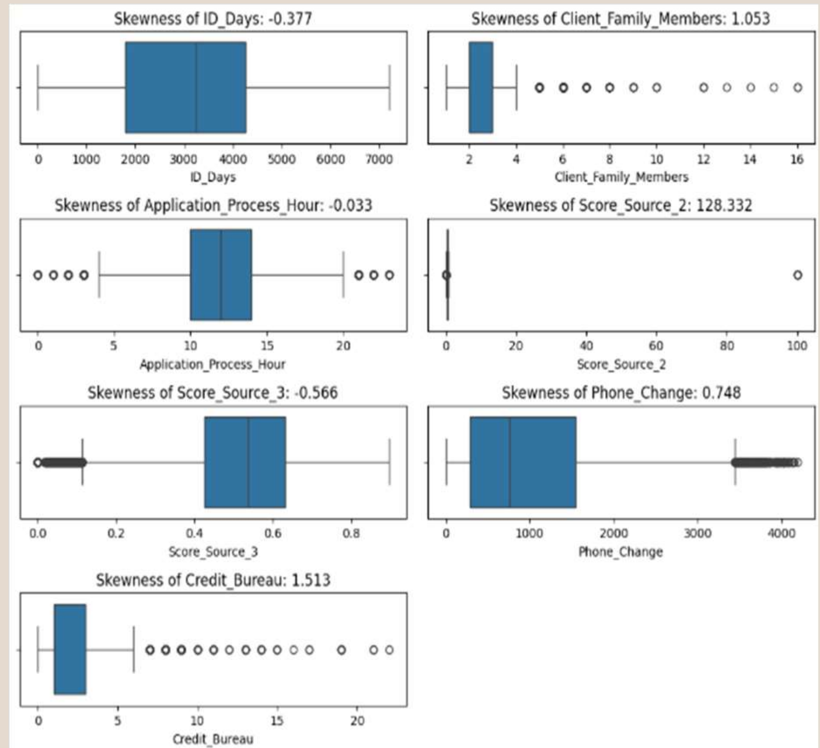
Impute missing values in numeric columns with their median

```
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)
```

EXPLORATORY ANALYSIS OF DATA

Univariate Analysis





Boxplot Analysis Inferences :

Most features show positive skewness with outliers in income, loan amounts, and family size, while age and registration days have nearly symmetrical distributions.

Univariate Analysis of Numerical Column

INFERENCES FROM DISPLOT:

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

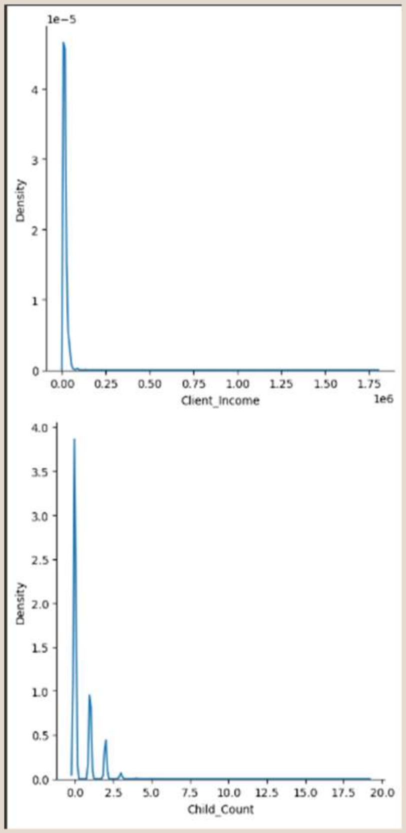
2. Child Count: Most clients have 0-2 children, with very few having more than 3. Outliers with unusually high child counts

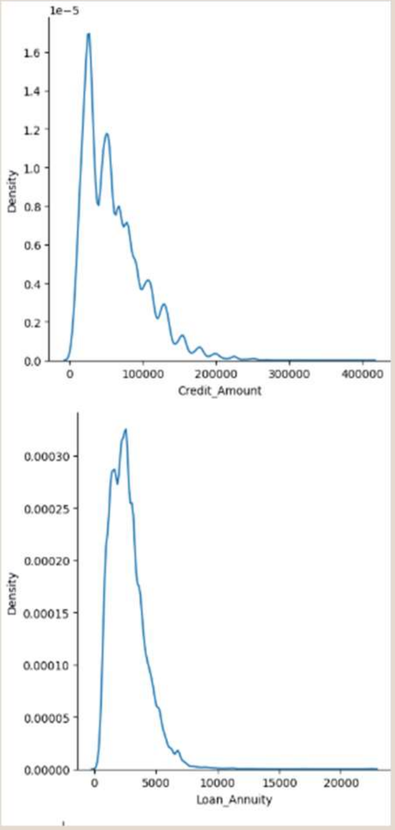
3. Credit Amount: Right-skewed, most loans are small with fewer large loan amounts.

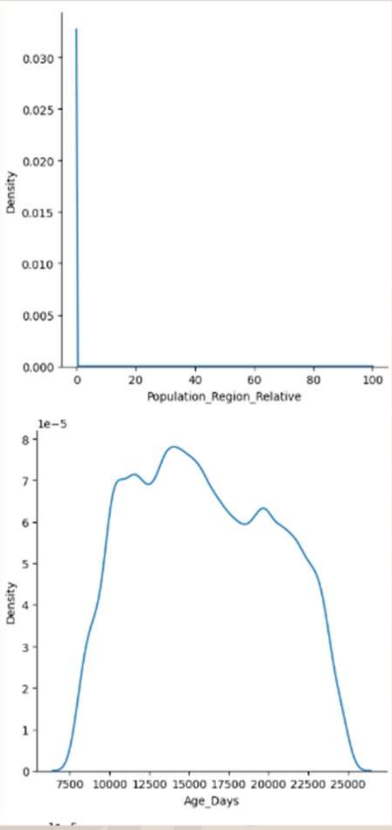
4. Loan Annuity: Right-skewed, most clients have low annuities, with outliers at the higher end.

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

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







Univariate Analysis of Numerical Column

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

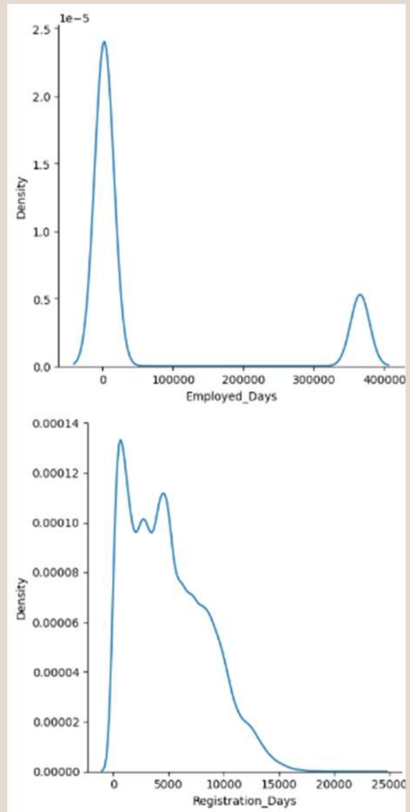
8. Registration Days: Skewed distribution, most clients have been registered for a shorter duration.

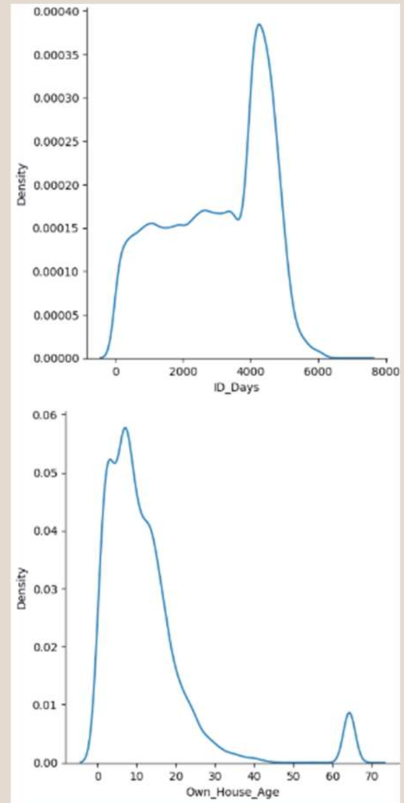
9. ID Days: Indicates issuance/verification timeframes, peaks at specific durations.

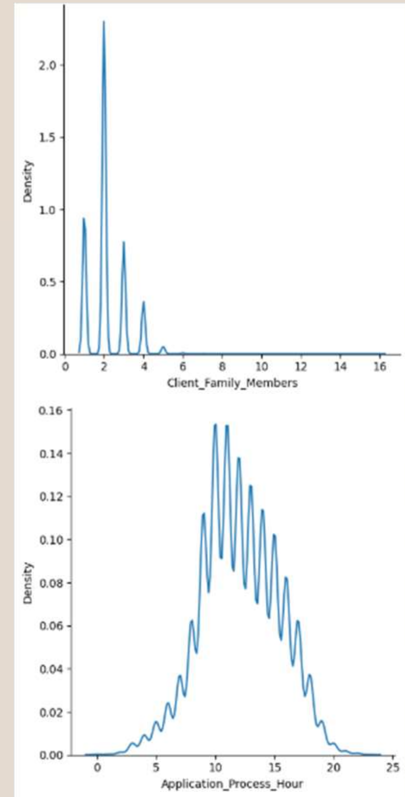
10. Own House Age: Concentrated at lower values, with a peak at older ages of house around 60-70 years.

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

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







Univariate Analysis of Numerical column

13. Score Source 1: Uniform distribution.

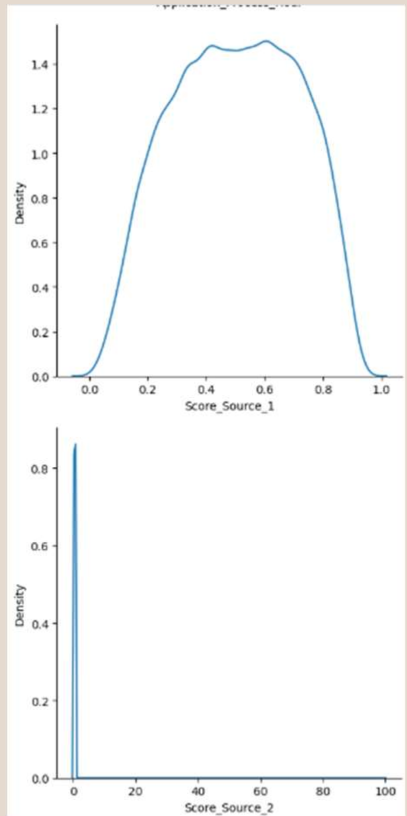
14. Score Source 2: Highly positively skewed after 0.

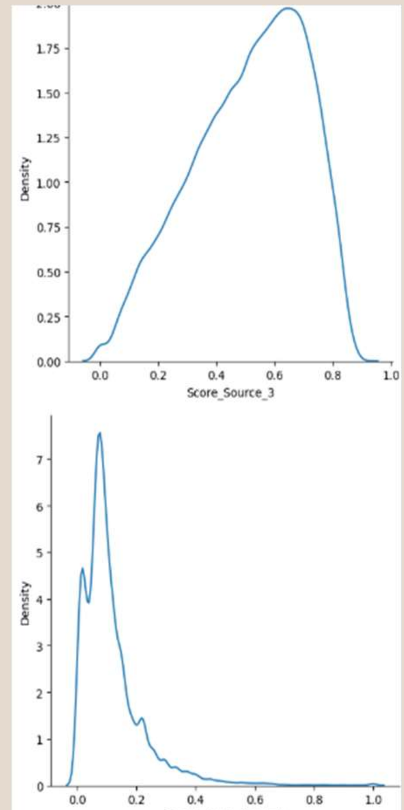
15. Score Source 3: Bell-shaped distribution between 0 and 1.

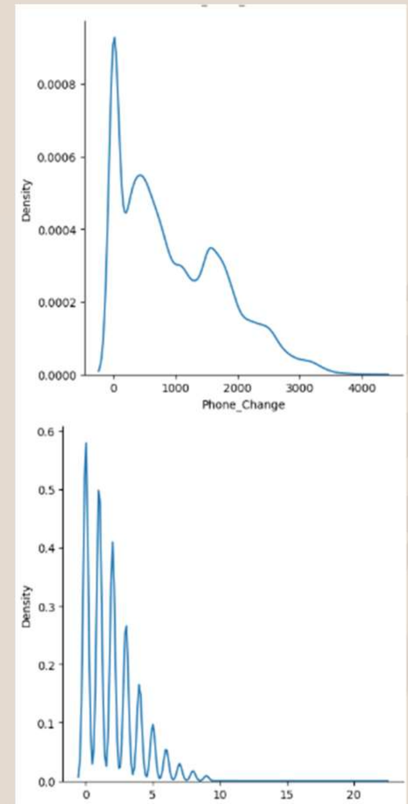
16. Social Circle Default: Right-skewed, higher values may indicate default risk based on social environment.

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

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



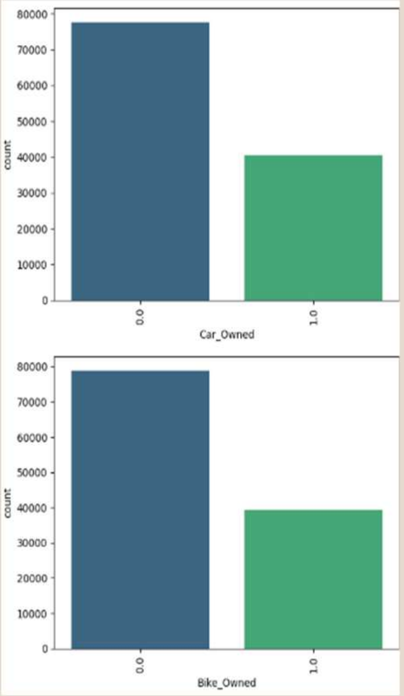


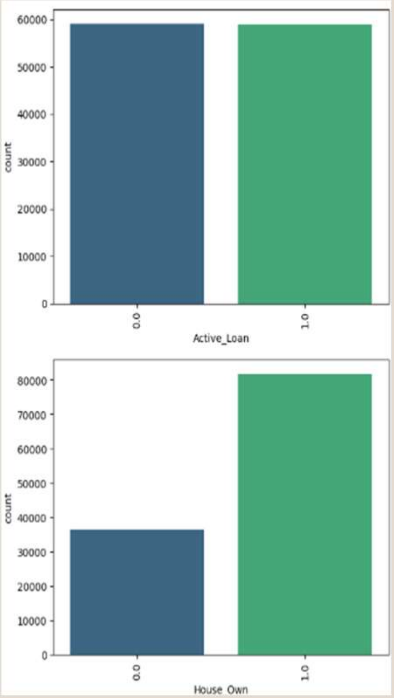


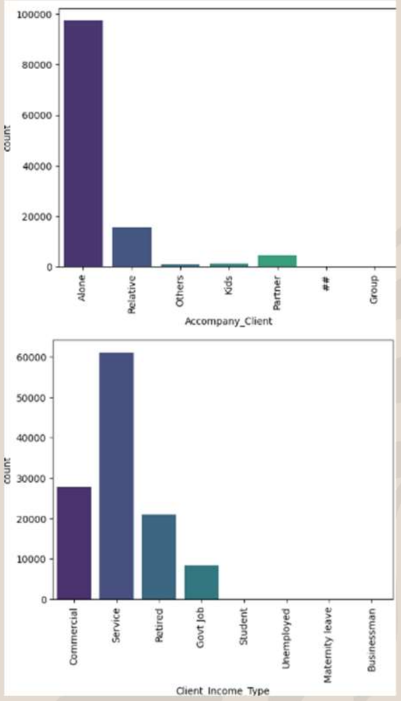
Univariate Analysis of Categorical Column

INFERENCE FROM COUNT PLOT:

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'.







Univariate Analysis of Categorical Column

7. Client_Education: Majority of clients have secondary education, followed by graduates and so on.

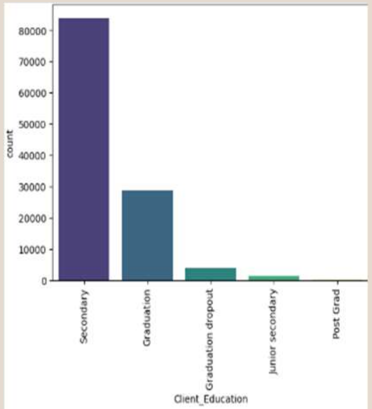
8. Client_Marital_Status: Married clients are the largest group, followed by singles and divorced and widow groups are almost similar.

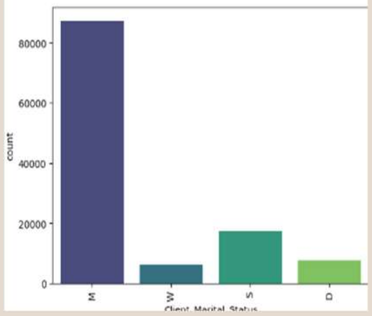
9. Client_Gender: More male clients are there than female clients.

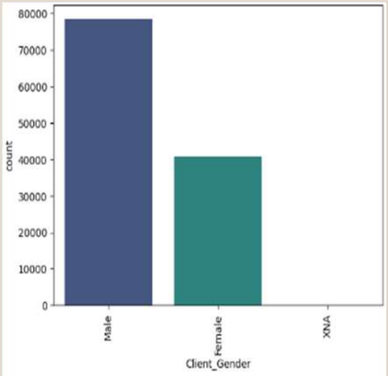
10. Loan_Contract_Type: Most loans are Consumer Loans (CL), fewer are Revolving Loans (RL).

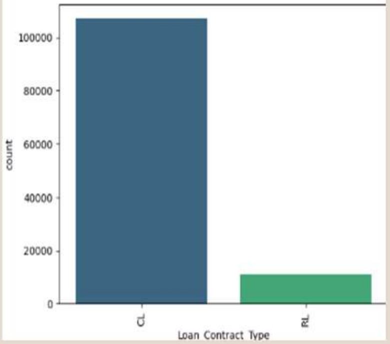
11. Client_Housing_Type: Most clients live in their own homes.

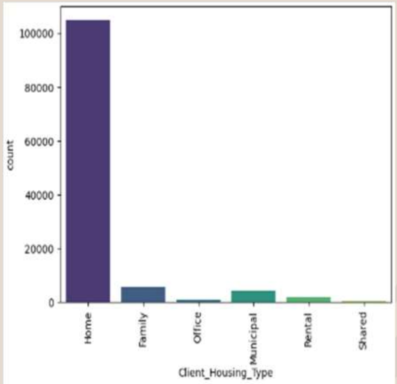
12. Mobile_Tag: Almost all clients have registered mobile tags.

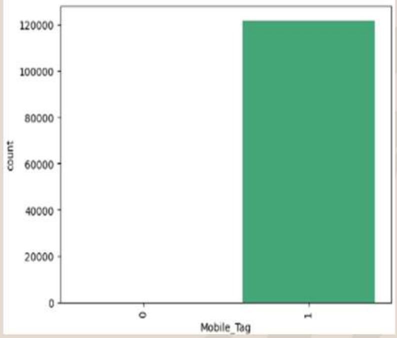












Univariate Analysis of Categorical Column

13. Homephone_Tag: Fewer clients have a registered home phone.

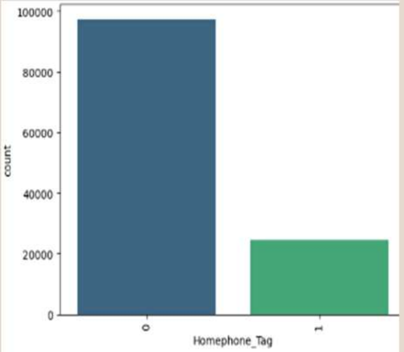
14. Workphone_Working: Fewer clients have provided their workplace contact.

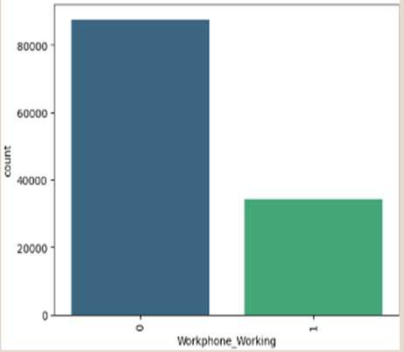
15. Client_Occupation: Dominated by sales, laborers, and core staff followed by others.

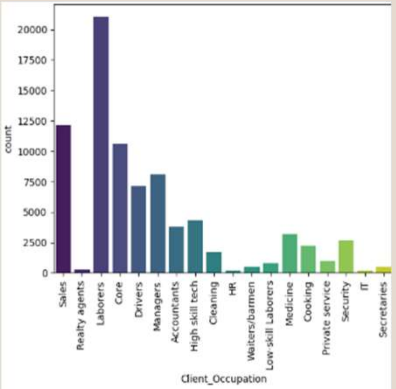
16. Client_City_Rating: Most clients are from cities with a rating of '2', followed by '1' and '3’.

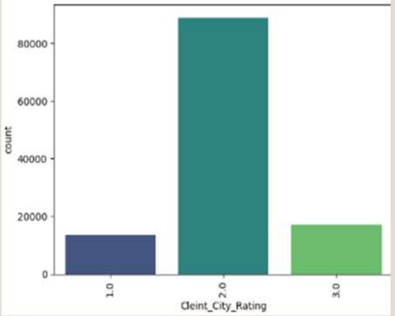
17. Application_Process_Day: Mostly clients processed their applications on week days than weekends and the highest applications were processed on 'Tuesday’.

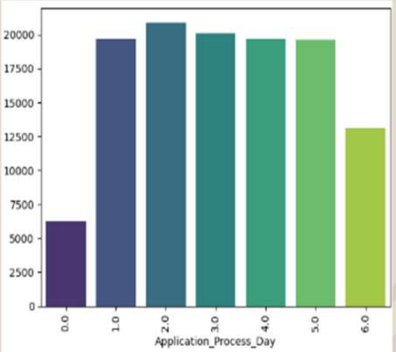
18. Client_Permanent_Match_Tag: Most clients have a permanent match tag.

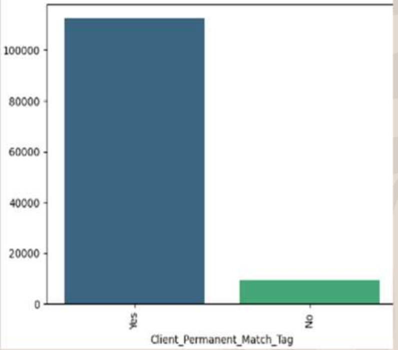










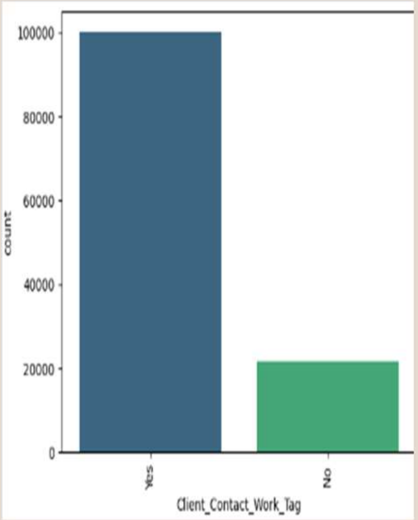


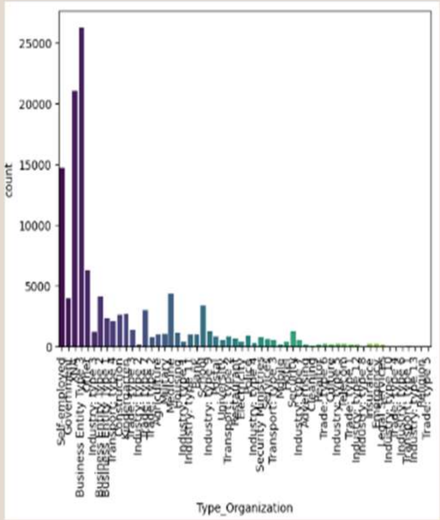
Univariate Analysis of Categorical Column

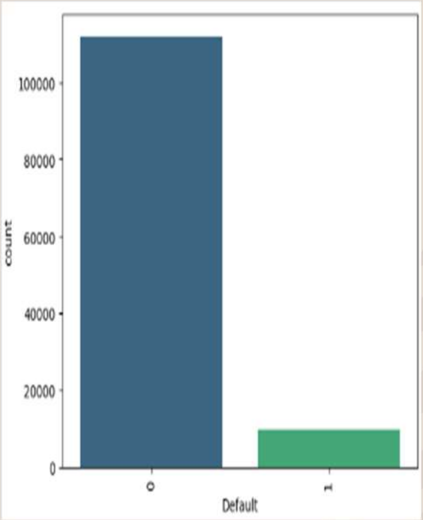
19. Client_Contact_Work_Tag: Majority of clients have a contact work tag.

20. Type_Organization: Majority of clients are 'Business Entity Type 3 followed by 'XNA' and 'Self-employed' and so on.

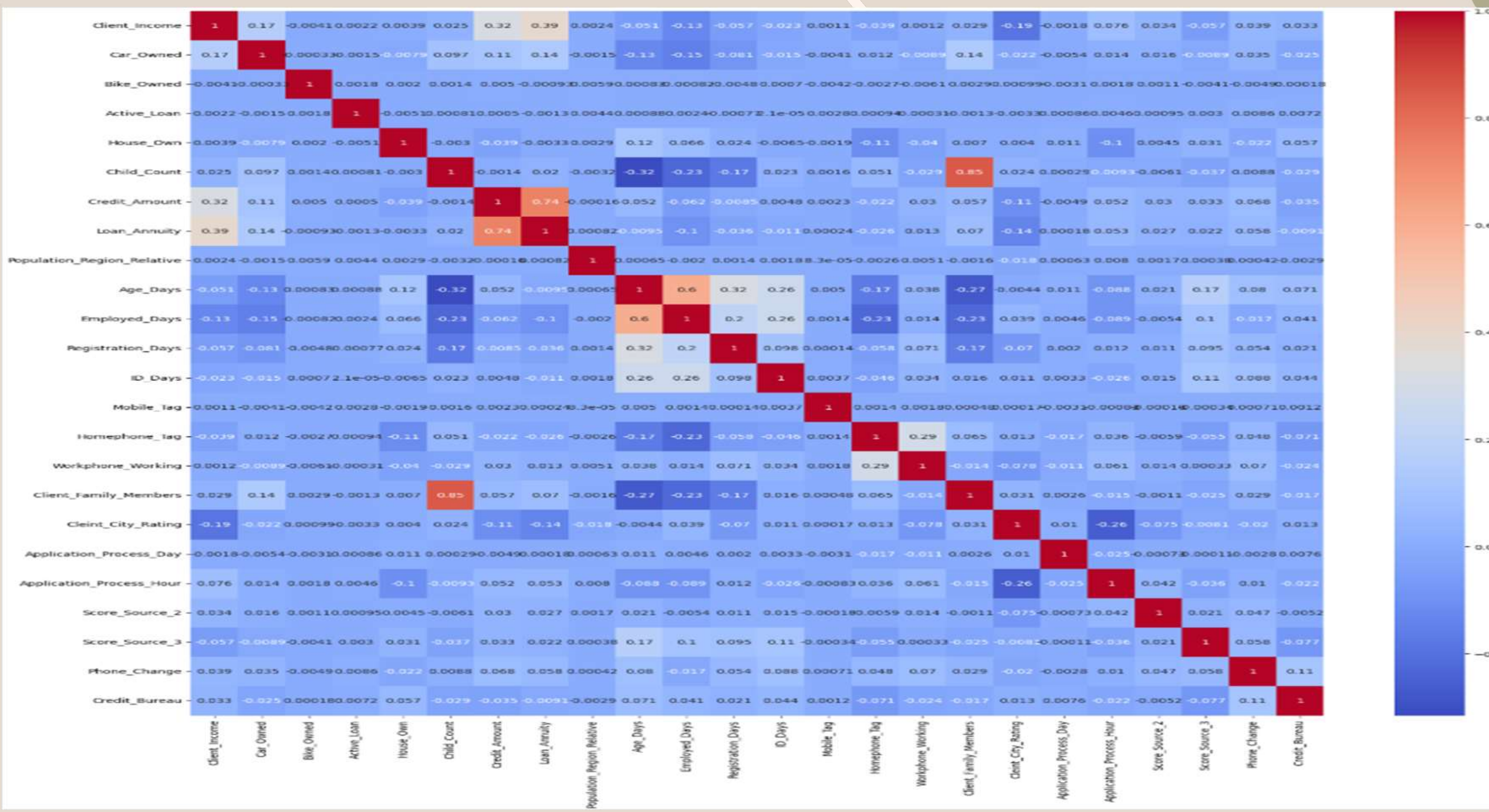
Default: Most of clients have not defaulted their loan







Multivariate Analysis



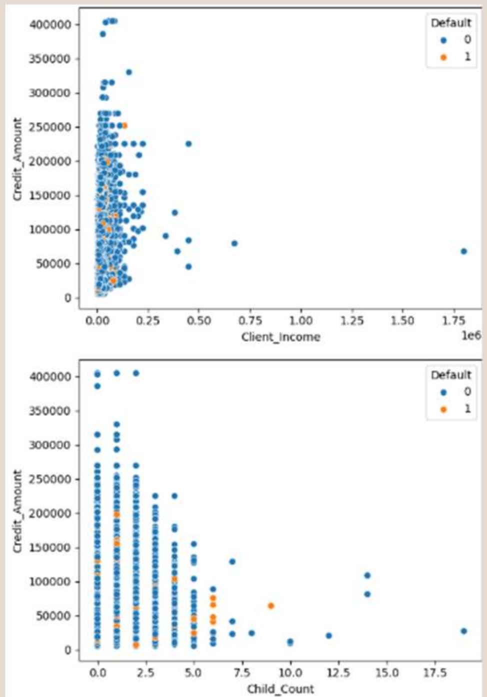
INFERENCES FROM HEATMAP :

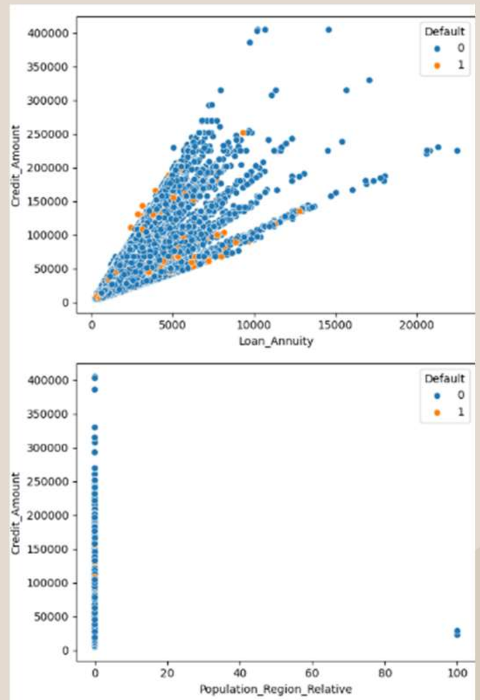
1.Most of the columns have positive correlation.
2.Highly positive correlation is between Client Family Members with Child Count i.e., 0.85 followed by Loan Annuity with Credit Amount i.e., 0.74
3.Highly negative correlation is between Age Days and Child Count i.e., -0.32.

Bivariate Analysis of Numerical Columns

INFERENCES FROM SCATTER PLOT:

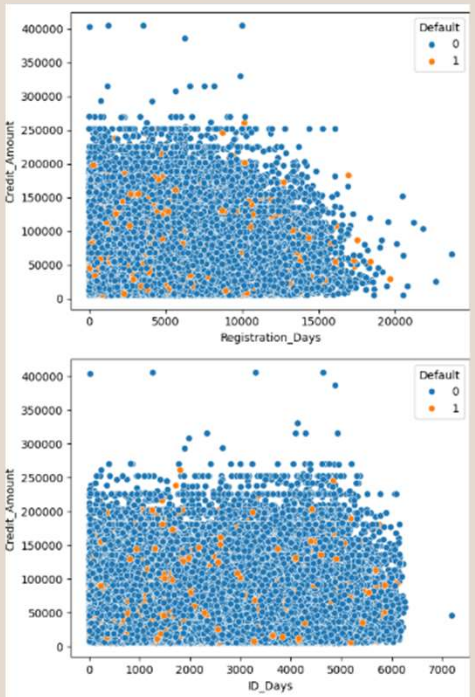
1.Client_Income: Mostly clients have an income between 0 and 0.25 and most of them have not defaulted on their loan. And some of them who defaulted are overlapped.
2.Child_Count: Mostly clients have less than 3 children and they have not defaulted their loan, very few of them defaulted their loan.
3.Loan_Annuity: 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 is overlapped with the non-defaulter category.
4.Population_Region_Relative: Just one or two of the clients have defaulted loan.
5.Age_Days: Mostly clients have not defaulted loans whatever being 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 loan.

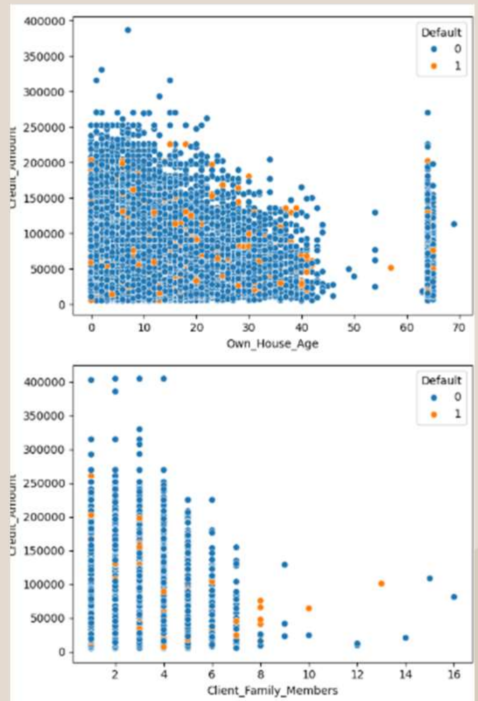


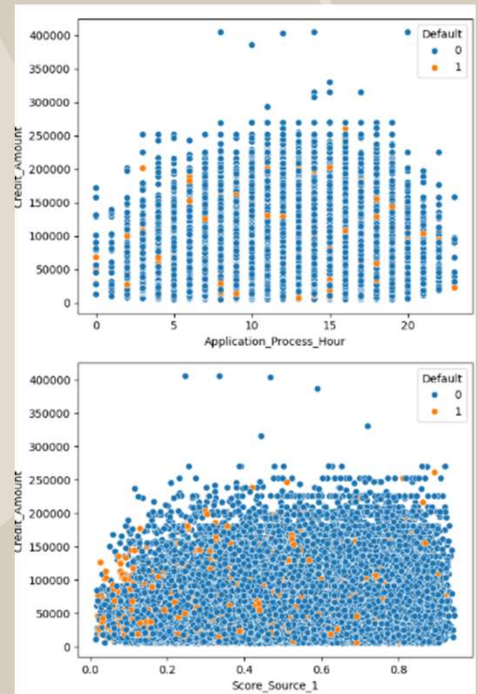


Bivariate Analysis of Numerical Columns

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 lies between 0.0 and 0.2 and in other categories there are less defaulted loans and mostly clients have not defaulted loans.

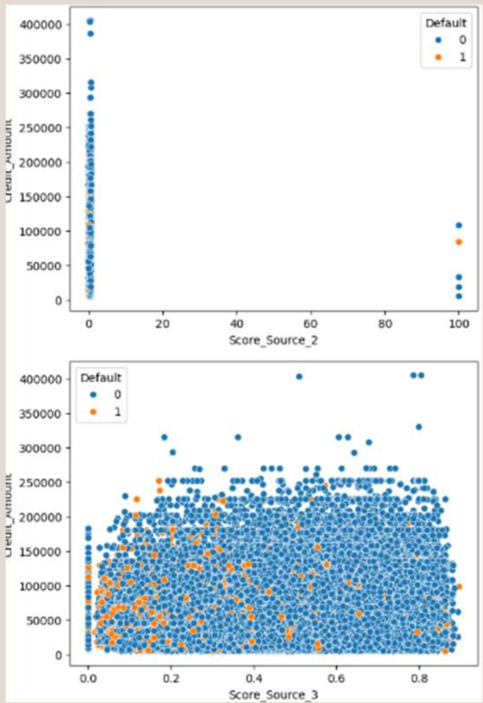


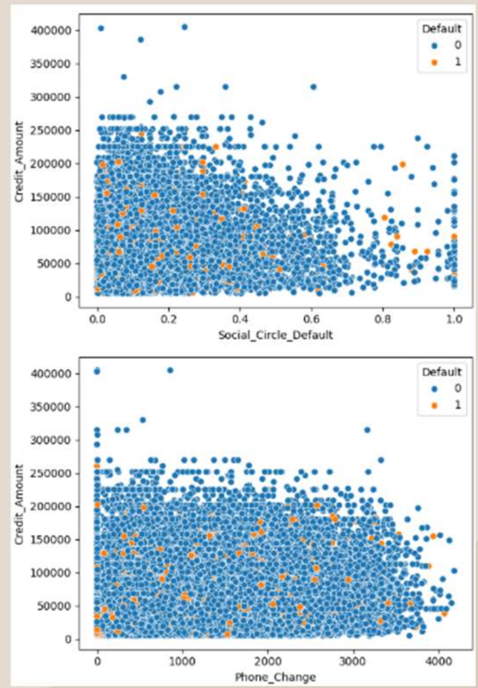


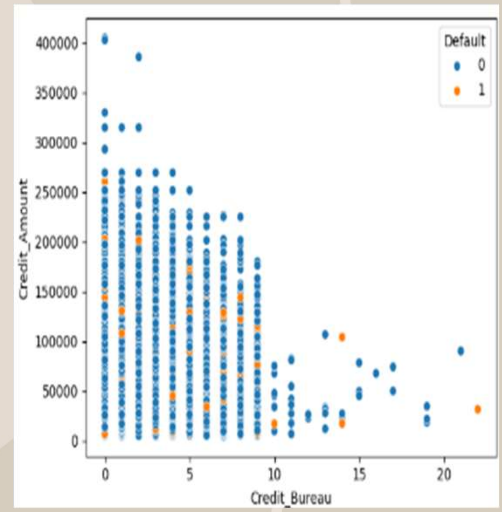


Bivariate Analysis of Numerical Columns

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. which is again overlapped by 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 outlier



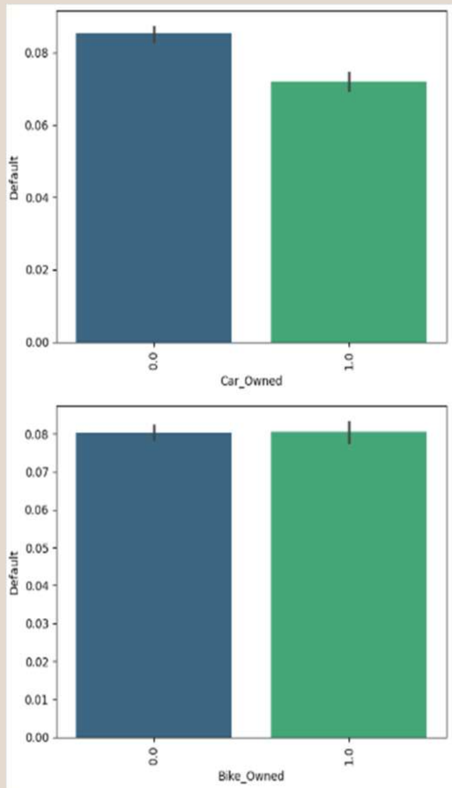


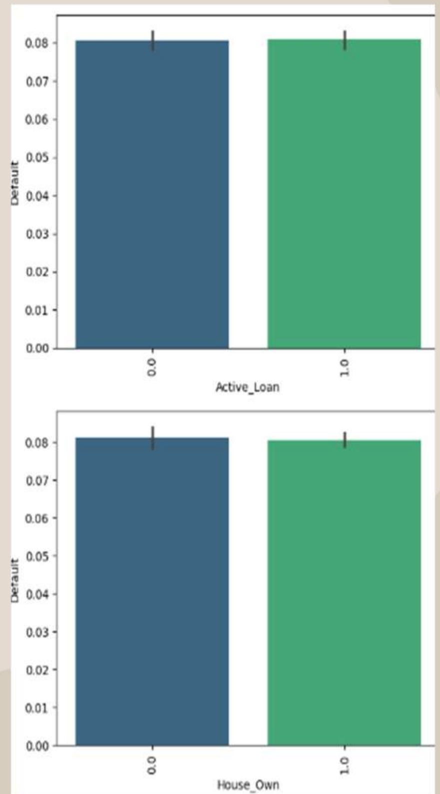


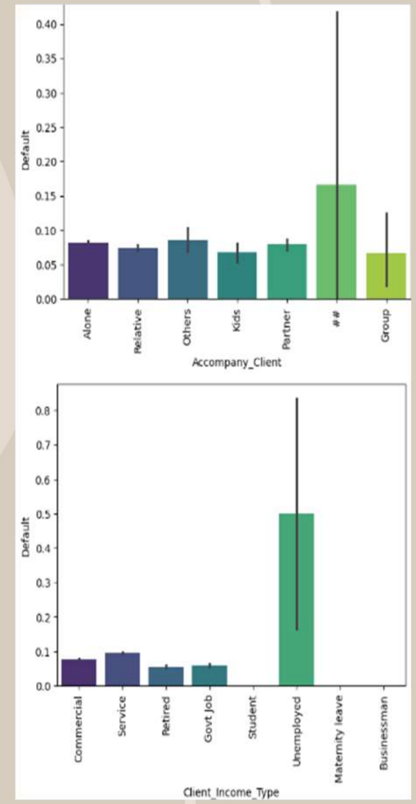
Bivariate Analysis of Numerical Columns

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 weather they an active loan or not.
4. House_Own: Clients who owns a house and who don't have almost equal number of defaulters.
5. Accompany_Client: category of accompany client have maximum number of defaulters followed by 'Others', 'Alone' and 'Partner’.
6. Client_Income_Type: Maximum number of loan defaulters are the 'Unemployed' onesfollowed by 'Service' ones and no defaulters in 'Maternity leave' and 'Businessman' category.

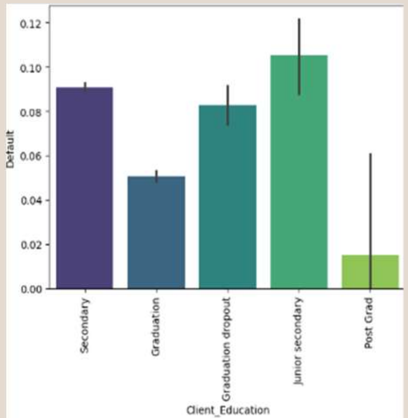


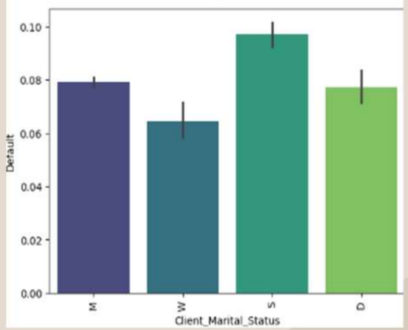


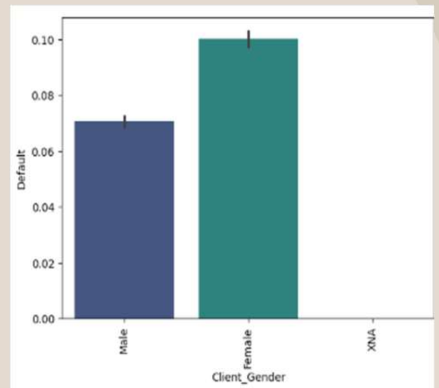


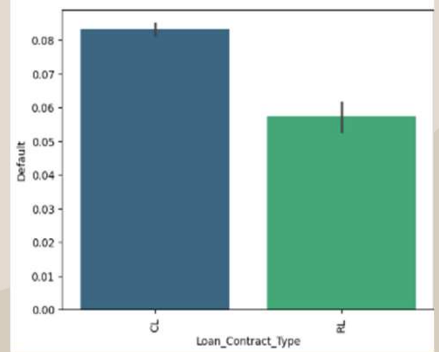
Bivariate Analysis of Numerical Columns

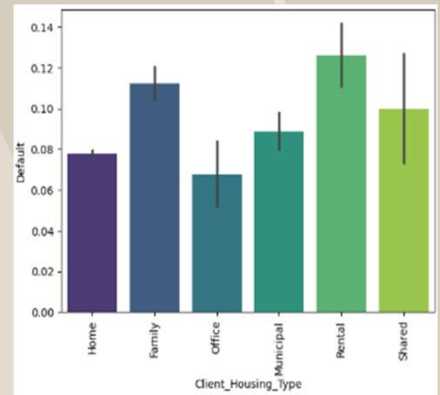
7. Client_Education: Clients with 'Junior Secondary' education are the highest defaulter loans follwed by 'Secondary' education ones.
8. Client_Marital_Status: 'Single' clients have maximum loan defaulters followed by 'Married' and 'Divorced' and then 'Widow'.
9. Client_Gender: 'Female' clients have more defaulted loans than 'Male' clients.
10. Loan_Contract_Type: Maximum number of defaulters are in 'Cash Loan' category.
11. Client_Housing_Type: The 'Rental' one's have maximum number of defaulters followed by 'Family' and then 'Shared' then others.
12. Mobile_Tag: Clients who provided number have maximum number of defaulter loans.

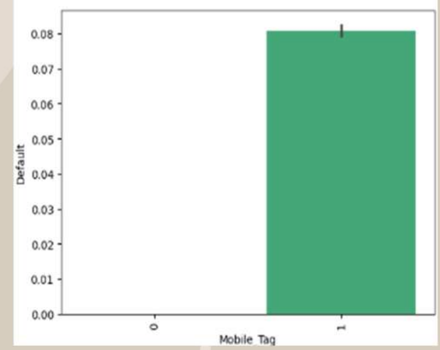






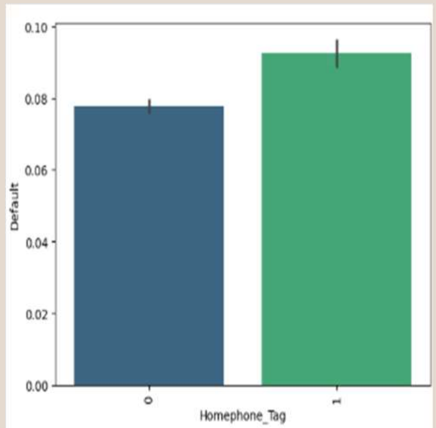


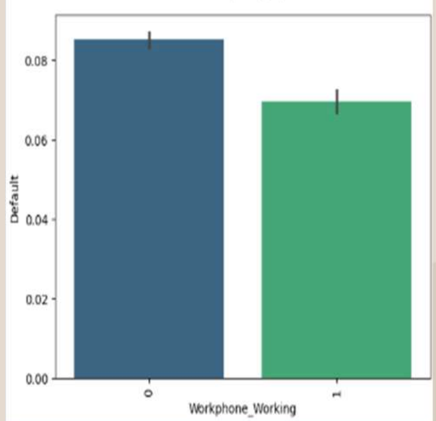


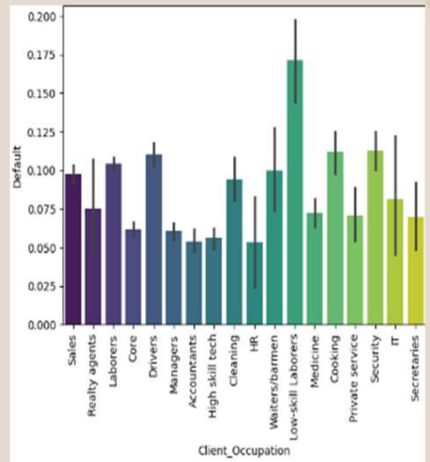


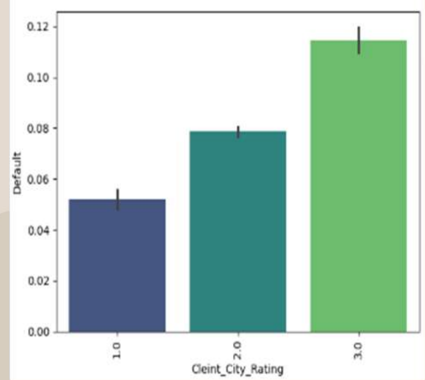
Bivariate Analysis of Numerical Columns

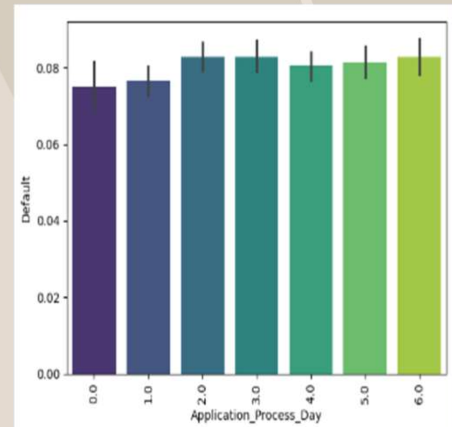
13. Homephone_Tag: Clients who provided home phone number have maximum loan defaulters than those who have not provided phone number.
14. Workphone_Working: Clients whose workphone is not reachable have maximum number of defaulters than those whose workphone is reachable.
15. Client_Occupation: Clients 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 equal number of loan defaulters except 'Tuesday' and 'Wednesday' are being the highest.
18.Client_Permanent_Match_Tag: Client permanent match tag of 'No' category have maximum number of defaulters.

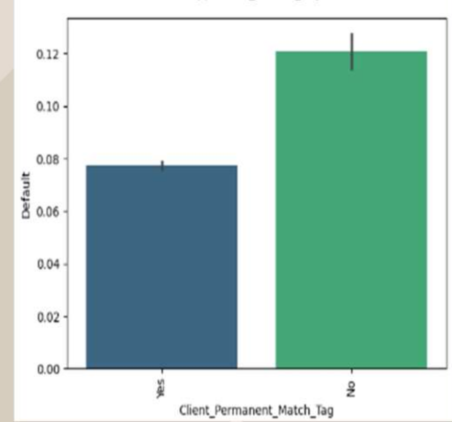












MODEL BUILDING

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	Accompany_Client_Others	Accompany_Client_Partner
12142509	0.0	0.0	1.0	0.0	1	1	0	2.0	6.0	0	1	0	0	0	0
12138936	1.0	0.0	1.0	NaN	1	0	1	2.0	3.0	0	1	0	0	0	0

CONCATENATE NUMERIC & CATEGORICAL COLUMN

```
data = pd.concat([numeric_df, object_encoded], axis=1)
data.head(2)
```

ID	Client_Income	Child_Count	Credit_Amount	Loan_Annuity	Population_Region_Relative	Age_Days	Employed_Days	Registration_Days	ID_Days	Own_House_Age	Client_Family_Members	Application_Process_Hour	Score_Source_1	Score_Source_2	Score_Source_3	Social_Circle_Default
12142509	6750.0	0.0	61190.55	3416.85	0.028663	13957.0	1062.0	6123.0	383.0	NaN	2.0	17.0	0.568066	0.478787	NaN	0.0186
12138936	20250.0	0.0	15282.00	1826.55	0.008575	14162.0	4129.0	7833.0	21.0	0.0	2.0	10.0	0.563360	0.215068	NaN	NaN

TRAIN TEST SPLIT

```
x = data.drop(columns='Default')
y = data['Default']
```

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state=2)
```

```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(97484, 131)
(24372, 131)
(97484,)
(24372,)
```

BASE MODEL

LOGISTIC REGRESSION

```
Accuracy: 0.9196208764155588
Confusion Matrix:
 [[22413     0]
 [ 1959     0]]
Classification Report:
              precision    recall  f1-score   support

           0       0.92      1.00      0.96     22413
           1       0.00      0.00      0.00      1959

    accuracy                           0.92     24372
   macro avg       0.46      0.50      0.48     24372
weighted avg       0.85      0.92      0.88     24372
```

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:

1. **RESAMPLING TECHNIQUES**: APPLYING OVERSAMPLING FOR CLASS "1" OR UNDERSAMPLING FOR CLASS "0" TO BALANCE THE CLASSES.
2. **ADJUSTING CLASS WEIGHTS**: INCREASING THE WEIGHT FOR CLASS "1" IN THE LOGISTIC REGRESSION MODEL TO PENALIZE FALSE NEGATIVES MORE HEAVILY.
3. **EXPLORING OTHER MODELS**: TRYING MORE COMPLEX MODELS LIKE DECISION TREES, RANDOM FORESTS, OR BOOSTING ALGORITHMS, WHICH MAY BETTER CAPTURE MINORITY CLASS PATTERNS.
4. **THRESHOLD TUNING**: EXPERIMENTING WITH THRESHOLD ADJUSTMENTS TO INCREASE SENSITIVITY FOR CLASS "1."

RANDOM FOREST

```
Metrics on Training Data:
Accuracy: 0.9999897419063641
Precision: 0.999989742020853
Recall: 0.9999897419063641
F1-Score: 0.9999897416097664
Confusion Matrix:
 [[89598     0]
 [    1  7885]]
Classification Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00     89598
           1       1.00      1.00      1.00      7886

    accuracy                           1.00     97484
   macro avg       1.00      1.00      1.00     97484
weighted avg       1.00      1.00      1.00     97484


Metrics on Testing Data:
Accuracy: 0.9292630887904152
Precision: 0.9343155159737986
Recall: 0.9292630887904152
F1-Score: 0.9027811715434578
Confusion Matrix:
...
    accuracy                           0.93     24372
   macro avg       0.96      0.56      0.59     24372
weighted avg       0.93      0.93      0.90     24372
```

INFERENCE FROM RANDOM FOREST 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 GRID SEARCH CV OR RANDOMIZED SEARCH CV. THIS WILL HELP OPTIMIZE PARAMETERS SUCH AS MAX DEPTH, N ESTIMATORS, MIN SAMPLES SPLIT, AND MIN SAMPLES LEAF TO IMPROVE GENERALIZATION.

BOOSTING MODELS

```
[ ] !pip install catboost
    #Import required libraries
    from sklearn.metrics import classification_report, roc_auc_score
    from xgboost import XGBClassifier
    from lightgbm import LGBMClassifier
    from catboost import CatBoostClassifier
    from sklearn.ensemble import GradientBoostingClassifier, AdaBoostClassifier
    # Define boosting models with class imbalance handling
    models = {
        "XGBoost": XGBClassifier(use_label_encoder=False, eval_metric='logloss', random_state=42,
                            scale_pos_weight=len(y_train[y_train == 0]) / len(y_train[y_train == 1])),
        "LightGBM": LGBMClassifier(random_state=42,
                            class_weight='balanced'),
        "CatBoost": CatBoostClassifier(verbose=0, random_state=42,
                            class_weights=[1, len(y_train[y_train == 0]) / len(y_train[y_train == 1])]),
        "GradientBoosting": GradientBoostingClassifier(random_state=42,
                            n_estimators=100,
                            subsample=0.8),
        "AdaBoost": AdaBoostClassifier(random_state=42,
                            n_estimators=100)}
```

```
[ ] for n, m in models.items():
        print(f"Training {n}...")
        m.fit(x_train, y_train)  # Train the model on the original data
        y_pred = m.predict(x_test)
        y_prob = m.predict_proba(x_test)[:, 1] if hasattr(m, "predict_proba") else None


        print(f"{n} Classification Report:")
        print(classification_report(y_test, y_pred))
        if y_prob is not None:
            print(f"{n} ROC-AUC Score: {roc_auc_score(y_test, y_prob)}")
        print("-" * 50)
```

Obsevation:

- 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.

TUNING MODELS

Defining function for Scores obtained after Tuning models

```
[ ] mod = []
    accu = []
    rec = []
    pre = []
    f1 = []
    ckap = []
    roc_s = []
    def model_validation(model,xtrain,ytrain,xtest,ytest):
        m = model
        m.fit(xtrain,ytrain)
        pred_h = m.predict(xtest)
        pred_s = m.predict_proba(xtest)[:,1]

        print('Confusion Matrix:\n',confusion_matrix(ytest,pred_h))
        print('\nClassifaction report:\n',classification_report(ytest,pred_h))


        ans = input('Do you want to save the result? Y/N')
        if ans.lower()=='y':
            mod.append(str(model))
            accu.append(accuracy_score(ytest,pred_h))
            pre.append(precision_score(ytest,pred_h))
            rec.append(recall_score(ytest,pred_h))
            f1.append(f1_score(ytest,pred_h))
            ckap.append(cohen_kappa_score(ytest,pred_h))
            roc_s.append(roc_auc_score(ytest,pred_s))
            global scorecard
            scorecard = pd.DataFrame({'Model':mod,'Accuracy':accu,'Precesion':pre,'Recall':rec,
                                      'F1 Score':f1,'Cohen Kappa':ckap, 'ROC AUC':roc_s})

        else:
            return
```

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
```

The model has high precision for class 0 (0.94), but its recall for class 1 is low (0.53), meaning it struggles to identify true positives for class 1. The overall accuracy is 59%, but the F1 score for class 1 is very low (0.17), indicating poor performance in predicting class 1. The model is biased towards predicting class 0, as shown by the significantly higher support for class 0.

DECISION TREE

```
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
```

The model has high precision for class 0 (0.94), but its recall for class 1 is low (0.53), meaning it struggles to identify true positives for class 1. The overall accuracy is 59%, but the F1 score for class 1 is very low (0.17), indicating poor performance in predicting class 1. The model is biased towards predicting class 0, as shown by the significantly higher support for class 0.

RANDOM FOREST

```
[ ] 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
```

The model classifies class 0 (majority class) well with high precision and recall, but it struggles with class 1 (minority class), predicting almost none correctly, as seen in the recall of 0.00. The overall accuracy is 92%, driven by the correct classification of class 0, but the poor performance for class 1 drastically lowers the macro F1-score to 0.48. This indicates severe class imbalance and poor generalization for the minority class.

ADABOOST CLASSIFIER

```
[ ] 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
```

The model achieves good performance for class 0 with a 93% F1-score but struggles significantly with class 1, showing a low precision and recall around 0.25–0.26. Although the overall accuracy is 88%, it is skewed by the dominance of class 0. The macro averages highlight the imbalance issue, indicating the need for improvement in handling minority class predictions.

XGB CLASSIFIER

```
[ ] 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
```

The model performs well for class 0 with a high precision and recall, but struggles with class 1, showing a low recall of 0.03 and an F1-score of 0.06, indicating that most instances of class 1 are misclassified. The overall accuracy of 92% is misleading due to the imbalance, as it heavily favors class 0. The macro averages show a significant drop, revealing poor performance on the minority class despite the high weighted average due to the dominance of class 0.

GRADIENT BOOSTING CLASSIFIER

```
[ ] 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
```

The model performs well for class 0 with a high precision and recall, but struggles with class 1, showing a low recall of 0.03 and an F1-score of 0.06, indicating that most instances of class 1 are misclassified. The overall accuracy of 92% is misleading due to the imbalance, as it heavily favors class 0. The macro averages show a significant drop, revealing poor performance on the minority class despite the high weighted average due to the dominance of class 0.

LGBM CLASSIFIER

```
Confusion Matrix:
 [[16143  6270]
 [  704  1255]]

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
```

The model shows decent recall for class 1 (0.64) but struggles with precision (0.17), leading to many false positives. While class 0 is classified with high precision (0.96), its recall drops to 0.72, resulting in an overall accuracy of 71%. The macro F1-score of 0.54 indicates significant performance disparity between classes, suggesting the model struggles to balance precision and recall effectively across both classes.

CATBOOST CLASSIFIER

```
Confusion Matrix:
 [[20618  1795]
 [ 1154   805]]

Classifaction report:
              precision    recall  f1-score   support

           0       0.95      0.92      0.93     22413
           1       0.31      0.41      0.35      1959

    accuracy                           0.88     24372
   macro avg       0.63      0.67      0.64     24372
weighted avg       0.90      0.88      0.89     24372
```

The model performs well for class 0 with a 93% F1-score but struggles with class 1, achieving low precision (0.31) and a moderate recall (0.41), indicating many false positives. The overall accuracy of 88% is driven mainly by class 0's dominance. The macro averages (F1-score of 0.64) highlight poor balance between the classes, suggesting the model needs improvement in handling minority class predictions.

SCORECARD

Model	Accuracy	Precesion	Recall	F1 Score	Cohen Kappa	ROC AUC
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
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
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
<catboost.core.CatBoostClassifier object at 0x7d4d38b8a680>	0.879000	0.309615	0.410924	0.353148	0.287858	0.752419

Model	Accuracy	Precesion	Recall	F1 Score	Cohen Kappa	ROC AUC
LogisticRegression(class_weight={0: 0.5440076787428291, 1: 6.180826781638347})	0.588339	0.102109	0.528841	0.171169	0.042099	0.581329
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
RandomForestClassifier(max_depth=5, max_features=None, n_estimators=150)	0.919621	0.000000	0.000000	0.000000	0.000000	0.704694

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

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

LIMITATION

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. LowPrecision 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. LowRecall 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: RandomForest 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.

Thank You