**MACHINE LEARNING**

ANALYZING LOAN DEFAULT RISK: EXPLORATORY DATA ANALYSIS

**About the dataset:** Banks run into losses when a customer doesn't pay their loans on time. Because of this, every year, banks have losses in crores, and this also impacts the country's economic growth to a large extent. We look at various attributes such as funded amount, location, loan, balance, etc., to predict if a person will be a loan defaulter or not. Grant Group Funding has a dataset of 87,501 rows and 30 columns based on a client in the banking sector.

**Dataset Link:** [**https://www.kaggle.com/datasets/marcbuji/loan-default-prediction/data**](https://www.kaggle.com/datasets/marcbuji/loan-default-prediction/data)

**Column Description:**

* ID: Unique ID
* Asst\_Reg: Value of all the assets registered under the borrowers name
* GGGrade: Grant Group Grade
* Experience: Total year of work experience of the borrower
* Validation: Validation status of the borrower
* Yearly Income : Total yearly income of the borrower
* Home Status: Borrower living status
* Unpaid 2 years : No. of times the Borrower has defaulted in last two years
* "Already
* Defaulted : Number of other loans the borrower was default"
* Designation : Designation of Borrower
* Debt to Income : Debt to Income ratio
* Postal Code : Postal code of borrower
* Lend Amount : Total funded amount to borrower
* "Deprecatory
* Records: An entry that may be considered negative by lenders because it indicates risk and
* hurts your ability to qualify for credit or other services"
* Interest Charged : Interest charged on total amount
* Usage Rate: Processing Charges on the Loan Amount
* Inquiries: Inquiries in Last 6 Months
* Present Balance: Current balance in the borrower account
* Gross Collection: The gross amount payable by way of Settlement or judgment in respect of the Claims, excluding any costs
* Sub GGGrade: Sub Grant Group Grade
* File Status: Status of the loan file
* State: State to which borrower belong
* Account Open: Total number of open accounts in the name of Borrower
* Total Unpaid CL: Unpaid dues on all the other loans
* Duration: Duration for the amount is funded to borrower
* Unpaid Amount: Unpaid balance on the credit card
* Reason: Reason for loan application
* Claim Type: Amongst all Application type what is the borrower Claim Type
* I - Individual Account , J - Joint Account"
* Due Fee: Charges incurred if the payment on loan amount is delayed
* Loan/No Loan: Target Variable

**Dataset:**

A screen shot of a computer

Description automatically generated

**Aim:** Doing a basic EDA on the dataset to check for any null values or drop unnecessary columns and adjust outliers and do data visualization to draw inferences from the various features affecting loan defaults

**Procedure:**

1. Importing necessary Python Libraries
2. Reading dataset
3. Checking of Null Values and Datatypes
4. Checking for missing values as well as datatypes
5. Checking the summary statistics
6. Replacing null values with the mean and median of that column
7. Data visualization using various plots and features
8. Making sure the data is clean by checking for outliers and sum of null values
9. Ensuring data is clean
10. Deciding on the important features that affect loan defaults

**Code:**

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

data = pd.read\_csv("C:/Users/ADMIN/Downloads/archive (1)/Loan Default Prediction Dataset.csv")

data.head()

data.shape

data.dtypes

data.describe()

data.columns

data.nunique()

*#finding the number of missing values*

data.isnull().sum()

df = data.drop(['ID','Asst\_Reg','Validation','Designation','Postal\_Code','Interest\_Charged','Usage\_Rate','Gross\_Collection','Sub\_GGGrade','File\_Status','State','Account\_Open','Duration','Claim\_Type'],axis=1)

df.columns

df.isnull().sum()

*# Load the dataset*

data\_df = pd.read\_csv("C:/Users/ADMIN/Downloads/archive (1)/Loan Default Prediction Dataset.csv")

*# Columns with null values*

columns\_with\_null = ['Yearly\_Income', 'Debt\_to\_Income', 'Total\_Unpaid\_CL', 'Unpaid\_Amount']

*# Filter rows with null values in specified columns*

filtered\_data = data\_df[data\_df[columns\_with\_null].isnull().any(axis=1)]

*# Get statistics for filtered columns*

statistics = filtered\_data[columns\_with\_null].describe().round(2)

print(statistics)

*# Specify the columns you want to replace null values with median*

columns\_to\_replace = ['Yearly\_Income','Total\_Unpaid\_CL','Unpaid\_Amount']

*# Replace null values with mean*

for column in columns\_to\_replace:

median\_value = df[column].median() *# Calculate the median for the column*

df[column].fillna(median\_value, inplace=True) *# Replace null values with median*

*# Specify the columns you want to replace null values with mean*

columns\_to\_replace = ['Debt\_to\_Income']

*# Replace null values with median*

for column in columns\_to\_replace:

mean\_value = df[column].mean() # Calculate the mean for the column

df[column].fillna(mean\_value, inplace=True) # Replace null values with mean

data.describe()

df.dtypes

df.isnull().sum()

df.head()

df.info()

*# Relationship between 'GGGrade' and 'Default'*

sorted\_df = df.sort\_values(by='GGGrade')

sns.countplot(data=sorted\_df,x='GGGrade', hue='Default')

plt.title('Default by GGGrade')

plt.show()

*# Correlation heatmap*

corr = df.corr()

sns.heatmap(corr, annot=True, cmap='coolwarm', fmt=".2f")

plt.title('Correlation Heatmap')

plt.show()

*# Scatter plot*

plt.figure(figsize=(8, 6))

sns.scatterplot(data=df, x='Total\_Unpaid\_CL', y='Unpaid\_Amount')

plt.title('Scatter plot between Total Unpaid CL and Unpaid Amount')

plt.xlabel('Total Unpaid CL')

plt.ylabel('Unpaid Amount')

plt.show()

*# Regression plot*

plt.figure(figsize=(8, 6))

sns.regplot(data=df, x='Total\_Unpaid\_CL', y='Unpaid\_Amount')

plt.title('Regression plot between Total Unpaid CL and Unpaid Amount')

plt.xlabel('Total Unpaid CL')

plt.ylabel('Unpaid Amount')

plt.show()

*# Count plot of 'GGGrade'*

plt.figure(figsize=(8, 6))

sns.countplot(data=df, x='GGGrade')

plt.title('Count of GGGrade')

plt.xlabel('GGGrade')

plt.ylabel('Count')

plt.show()

*# Bar plot*

plt.figure(figsize=(10, 8))

sns.barplot(data=df, x='GGGrade', y='Yearly\_Income', ci=None)

plt.title('Mean Yearly Income by GGGrade')

plt.xlabel('GGGrade')

plt.ylabel('Mean Yearly Income')

plt.show()

*# Pairplot of selected features colored by 'Default'*

selected\_features = ['Yearly\_Income', 'Lend\_Amount', 'Debt\_to\_Income', 'Present\_Balance', 'Default']

sns.pairplot(df[selected\_features], hue='Default', diag\_kind='kde')

plt.suptitle('Pairplot of Selected Features Colored by Default', y=1.02)

plt.show()

*# Heatmap of correlation between selected features*

plt.figure(figsize=(10, 8))

corr = df[selected\_features].corr()

sns.heatmap(corr, annot=True, cmap='coolwarm', fmt=".2f")

plt.title('Correlation Heatmap of Selected Features')

plt.show()

*# Relation plot*

sns.relplot(x='Yearly\_Income',y='Unpaid\_Amount',hue='Home\_Status',data=df)

*# Calculate the distribution of loan defaults*

loan\_defaults\_distribution = df['Default'].value\_counts(normalize=True)

*# Display the distribution*

print("Distribution of Loan Defaults:")

print(loan\_defaults\_distribution)

*# Grouped bar chart*

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

*# Group data by GGGrade and Home\_Status and calculate counts*

grouped\_data = df.groupby(['GGGrade', 'Home\_Status']).size().unstack()

*# Plotting grouped bar chart*

plt.figure(figsize=(10, 6))

grouped\_data.plot(kind='bar', stacked=True)

plt.title('Home Status by GGGrade')

plt.xlabel('GGGrade')

plt.ylabel('Count')

plt.xticks(rotation=45)

plt.legend(title='Home Status')

plt.show()

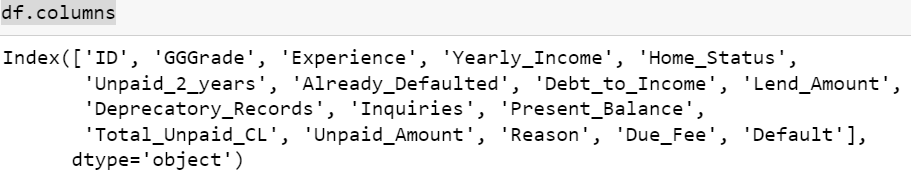
**Output:**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated



A screenshot of a computer

Description automatically generated

**A group of graphs with numbers

Description automatically generated with medium confidence**

***Outcome****: since there are many outliers to the right of Yearly income, Total\_Unpaid CL, Unpaid Amt plots they are skewed hence we replace their null values by their medians and in the case of plot of DTI we see a normal distribution. Hence we replace its null values by the mean value of that column*

A screenshot of a computer screen

Description automatically generated

A white rectangular object with a blue border

Description automatically generated with medium confidence

A graph of different colored bars

Description automatically generated

***Outcome****: GGGrade, or Grant Group Grade, appears to be a grading or rating system used by Grant Group Funding to assess the creditworthiness or risk level of borrowers. It likely indicates the borrower's credit score or credit rating assigned by the lending institution. The GGGrade helps the lender evaluate the likelihood of a borrower defaulting on a loan or their overall credit risk. Borrowers with higher GGGrades are considered more likely to repay their loans on time and are therefore more desirable from the lender's perspective*

*Default : Loan/No Loan: Target Variable*

*'0' typically represents one class (i.e., No Loan, Not Defaulted)*

*'1' typically represents the other class (i.e., Loan, Defaulted)*

*Hence based on these two factors we can observe that. There is a higher probability of having customers with their loan getting defaulted in lower GGGrades 1,2,3 and 4 compared to the higher grades 5,6 and 7 where it decreases.*

A graph of heatmap and a graph of the heatmap

Description automatically generated

***Outcome:*** *This correlation matrix is a table which shows relationship between variables, each cell in the table shows relationship between any 2 variables. Summarises data for advanced analysis*

*We can observe that Total Unpaid CL and Unpaid Amount has a good positive correlation with each other compared to the rest*

*A graph of a scatter plot

Description automatically generated*

***Outcome:*** *A positive correlation between these variables will be indicated by a generally upward trend in the scatter plot.*

* *Total Unpaid CL: If the "Total Unpaid CL" (Total Unpaid Credit Limit) is higher, it can potentially contribute to an increased risk of loan default.*
* *Unpaid Amt: a higher unpaid amount can indicate financial strain and may increase the risk of loan default.*

*So since they both are positively correlated we can observe that, the more theTotal unpaid Cl so is the more unpaid amount. And so is the higher chances of the loan getting defaulted*

**Results*:*** The results includes various visualizations and outcomes we have drawn from our dataset by performing EDA

*A graph of a graph with blue dots

Description automatically generated with medium confidence*

***Outcome:*** *Fitting a regression line to the scatter plot to visualize the trend more clearly*

*A graph of different colored bars

Description automatically generated*

***Outcome****: The bank should seriously consider only giving loans to peeople with a good credit score or GGGrade in order to avoid unnecessary loan defaults. But here we can see that the loan company has many customers with lower GGGrades and very few higher GGGrades.*

A graph of a number of colored bars

Description automatically generated with medium confidence

***Outcome:*** *The GGGrade does not simply increase with the yearly income of the people, it is affected due to the spending habits of people. So the loan company shoudld take into account the spending habits of people i.e. consider people with higher cggrades in order to avoid loan defaults. As we can see here the mean yearly income is almost same for every GGGrade customers in this graph*

A graph of a number of colored dots

Description automatically generated with medium confidence

A graph of different colored dots

Description automatically generated with medium confidence

In [89]:

*A screenshot of a graph

Description automatically generated*

***Outcome:*** *We can see that present balance and Yearly income has a better correlation between each other compared to other variables*

A screenshot of a computer code

Description automatically generated

A graph of a home status

Description automatically generated

***Output:*** *This graph once again proves how the gggrade describes a customers loan worthhiness. As we can see here a lot of cutomers in the lower gggrades have their house mortgaged which increases the risk of them not paying back their loan on time. Whereas in the higher gggrades the people who have their own house or are atleast renting it are relatively high, giving brief satisfction for the loan givers.*

**Results:**In conclusion, Exploratory Data Analysis is a crucial step in understanding loan default patterns, identifying risk factors, and informing decision-making processes for lenders. By leveraging EDA techniques, lenders can improve risk assessment, enhance portfolio management, and mitigate financial losses associated with loan defaults.

Exp No. 3

18-March-2024 *Logistic regression ML model*

**Aim:** The aim of this project is to develop a logistic regression model to analyze and predict loan default risk for a financial institution. Leveraging historical loan data and borrower information, the model will assist in assessing creditworthiness, identifying high-risk loan applicants, and implementing effective risk mitigation strategies. By accurately predicting the likelihood of loan default, the model aims to enhance credit risk management, minimize default losses, and optimize the institution's loan portfolio.

**Procedure:**

1. Data Preprocessing
2. Splitting the data into training and testing sets
3. Select features and select a ML algorithm
4. Training data
5. Testing data
6. Evaluate and make predictions for the new data

**Code:**

*# DATA PREPROCESSING*

*# Let's proceed with encoding the categorical variables.*

from sklearn.preprocessing import LabelEncoder

*# Label encoding for GGGrade, Experience, and Home\_Status*

label\_encoder = LabelEncoder()

df['GGGrade'] = label\_encoder.fit\_transform(df['GGGrade'])

df['Experience'] = label\_encoder.fit\_transform(df['Experience'])

df['Home\_Status'] = label\_encoder.fit\_transform(df['Home\_Status'])

*# One-hot encoding for Reason*

df = pd.get\_dummies(df, columns=['Reason'], drop\_first=True)

*# We can use the LabelEncoder from scikit-learn for label encoding and get\_dummies() function for one-hot encoding.*

*# SPLITTING THE DATA INTO TRAINING AND TESTING SETS*

*# Typically, we reserve a portion of the data (e.g., 20-30%) for testing the model's performance.*

from sklearn.model\_selection import train\_test\_split

*# Separate features (X) and target variable (y)*

X = df.drop('Default', axis=1)

y = df['Default']

*# Split the data into training and testing sets (80% train, 20% test)*

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

*# SELECT FEATURES AND SELECT A ML ALGORITHM*

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix, roc\_auc\_score

*# Initialize the Random Forest Classifier*

rf\_classifier = RandomForestClassifier(random\_state=42)

*# Train the model on the training data*

rf\_classifier.fit(X\_train, y\_train)

*# Predict on the testing set*

y\_pred = rf\_classifier.predict(X\_test)

*# Generate classification report*

print("Classification Report:")

print(classification\_report(y\_test, y\_pred))

*# Generate confusion matrix*

print("\nConfusion Matrix:")

print(confusion\_matrix(y\_test, y\_pred))

*# Calculate AUC-ROC*

auc\_roc = roc\_auc\_score(y\_test, y\_pred)

print("\nAUC-ROC Score:", auc\_roc)

*#TRAINING DATA*

*#1 Initialization of Logistic Regression model: initializes an instance of the Logistic Regression model with a specified random state.*

from sklearn.linear\_model import LogisticRegression

*# Initialize the Logistic Regression model*

logistic\_regression = LogisticRegression(random\_state=42)

*#2 Train the model on the training data: fits the Logistic Regression model to the training data*

*#(X\_train features and y\_train target variable).*

logistic\_regression.fit(X\_train, y\_train)

*#3 Predicting on the testing set: uses the trained model to predict the target variable (y\_pred) for the testing set (X\_test features).*

y\_pred = logistic\_regression.predict(X\_test)

*#4 Evaluating the model:prints the evaluation metrics for the model's performance on the testing set, including precision, recall, F1-score, and support.*

print("Evaluation Metrics:")

print(classification\_report(y\_test, y\_pred))

*#TESTING DATA*

*#example new data entry. Change the values according to the customer*

import pandas as pd

*# Create a DataFrame for the new data*

new\_data = pd.DataFrame({

'GGGrade': ['III'], *# Example value for 'GGGrade'*

'Experience': ['2yrs'], *# Example value for 'Experience'*

'Yearly\_Income': [60000.0], *# Example value for 'Yearly\_Income'*

'Home\_Status': ['RENT'], *# Example value for 'Home\_Status'*

'Unpaid\_2\_years': [0], *# Example value for 'Unpaid\_2\_years'*

'Already\_Defaulted': [1], *# Example value for 'Already\_Defaulted'*

'Debt\_to\_Income': [20.0], *# Example value for 'Debt\_to\_Income'*

'Lend\_Amount': [30000.0], *# Example value for 'Lend\_Amount'*

'Deprecatory\_Records': [0], *# Example value for 'Deprecatory\_Records'*

'Inquiries': [1], *# Example value for 'Inquiries'*

'Present\_Balance': [5000.0], *# Example value for 'Present\_Balance'*

'Total\_Unpaid\_CL': [20000.0], *# Example value for 'Total\_Unpaid\_CL'*

'Unpaid\_Amount': [10000.0], *# Example value for 'Unpaid\_Amount'*

'Reason': ['debt consolidation'], *# Example value for 'Reason'*

'Due\_Fee': [500.0], *# Example value for 'Due\_Fee'*

})

*# Print the new data*

print("New Data:")

print(new\_data)

*# Perform one-hot encoding for categorical variables, ensuring alignment with training data columns*

new\_data\_encoded = pd.get\_dummies(new\_data, columns=['GGGrade', 'Experience', 'Home\_Status', 'Reason'])

*# Align the columns with the training data columns*

new\_data\_encoded = new\_data\_encoded.reindex(columns=X\_train.columns, fill\_value=0)

*# Now, predict loan defaults on the new data*

new\_data\_predictions = logistic\_regression.predict(new\_data\_encoded)

*# Print the predictions*

print("Predictions for new data:")

print(new\_data\_predictions)

**Output:**

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer screen

Description automatically generated

A close-up of a white background

Description automatically generated



**Results:**

* The prediction of 0 indicates that the model predicts that the new data entry is not likely to result in a loan default.
* If the prediction for the new data is 1, it means that the model predicts a higher probability of loan default based on the features provided in the new data entry.
* This prediction aligns with the trained logistic regression model's decision boundary based on the features provided in the new data entry.

**Summary:**

Exploratory Data Analysis is a crucial step in understanding loan default patterns, identifying risk factors, and informing decision-making processes for lenders. By leveraging EDA techniques, lenders can improve risk assessment, enhance portfolio management, and mitigate financial losses associated with loan defaults.Top of Form