LOAN APPROVAL ANALYSIS

CAPSTONE PROJECT

SWIPE THE PAGE

INTRODUCTION

In the competitive world of finance, lending institutions constantly strive to make informed decisions regarding loan approvals. They want to ensure they approve loans to creditworthy individuals while minimizing the risk of defaults. This is where Exploratory Data Analysis (EDA) becomes a powerful tool.

OBJECTIVE

The key objectives of this project include:

1.Data Understanding: Gain a comprehensive understanding of the structure and characteristics of the dataset. This involves exploring the variables, their types, distributions, and relationships.

2.Data Cleaning: Preprocess the data to handle missing values, outliers, and inconsistencies. This step ensures that the data is ready for analysis and modeling.

3.Exploratory Data Analysis: Perform exploratory analysis techniques such as summary statistics, data visualization, and correlation analysis to uncover patterns, trends, and relationships within the data. This step may involve identifying factors that significantly influence loan approval decisions, such as income level, credit score, employment status, etc.

OBJECTIVE

4. Insights Generation: Extract actionable insights from the analysis that can inform decision making in the loan approval process. These insights may include identifying risk factors associated with loan default, understanding the characteristics of successful loanapplicants, and optimizing approval criteria. Overall, this project aims to leverage exploratory data analysis techniques to gain valuable insights into the loan approval process, ultimately contributing to more informed decision making and improved risk management in financial institutions.

IMPORTING PYTHON LIBRAIRES

1.NumPy

Import Statement: import numpy as np -
Numerical Operations: NumPy is a fundamental library for numerical operations in Python. It provides support for large, multi dimensional arrays and matrices, along with mathematical functions to operate on these arrays.

2.Pandas

Import Statement: import pandas as pd -
Data manipulation and Analysis: Pandas is a powerful library for data manipulation and analysis. It provides data structures like DataFrames and Series, which are essential for handling and analysing structured data.

3.Seaborn

Import Statement: import seaborn as sns -
Statistical Data Visualization: Seaborn is built on top of Matplotlib and provides a high-level interface for creating attractive and informative statistical graphics. It simplifies the creation of complex visualizations.

IMPORTING PYTHON LIBRAIRES

4.Matplotlib

Import Statement: import matplotlib.pyplot as plt-
2D Plotting Library: Matplotlib is a widely used library for creating static, interactive, and animated visualizations in Python. It provides various functions to create different types of plots and charts.

5.Warnings

Import Statement: import warnings
warnings.filterwarnigs('ignore') -
Importing the warning Module is a function call that set ups a filter for warning messages. The parameter 'ignore' specifies that any warning message issued during program exeecution should be ignored, meaning they won't be displayed to the user.

6.Plotly

Import Statement: import plotly.express as px-
Plotly is a Python graphing library that creates interactive, publication-quality graphs. With Plotly, users can generate visually appealing graphs for data analysis, exploration, and presentation, making it a powerful tool for data visualization in fields such as data science, and research.

DATA OVERVIEW

LOADING DATASET

```
In [2]:  1  df=pd.read_csv('loan_sanction_test.csv')
```

I've load the dataset named 'loan_sanction_test.csv' into a variable named 'df'. In order to import the dataset, I've used Pandas Libraries in which I've used 'pd.read_csv' command to import the respective dataset.

SHAPE OF DATASET

```
In [4]:  1  # Lets Check The Shape Of Dataset.
         2  df.shape

Out[4]:  (367, 12)
```

The data set contain a total of 367 rows and 12 columns.

DATA OVERVIEW

Viewing Columns

```
In [5]: 1 df.columns

Out[5]: Index(['Loan_ID', 'Gender', 'Married', 'Dependents', 'Education',
               'Self_Employed', 'ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
               'Loan_Amount_Term', 'Credit_History', 'Property_Area'],
              dtype='object')
```

I've accessed and displayed the columns names of the loaded dataset using 'df.columns'. These columns encompass the various aspects of loan applicants information, providing a detailed overview of the dataset.

Data Set First View

```
In [3]: 1 # Display the first few rows of the dataset
        2 df.head(10)
```

Out[3]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
0	LP001015	Male	Yes	0	Graduate	No	5720	0	110.0	360.0	1.0
1	LP001022	Male	Yes	1	Graduate	No	3076	1500	126.0	360.0	1.0
2	LP001031	Male	Yes	2	Graduate	No	5000	1800	208.0	360.0	1.0
3	LP001035	Male	Yes	2	Graduate	No	2340	2546	100.0	360.0	NaN
4	LP001051	Male	No	0	Not Graduate	No	3276	0	78.0	360.0	1.0
5	LP001054	Male	Yes	0	Not Graduate	Yes	2165	3422	152.0	360.0	1.0
6	LP001055	Female	No	1	Not Graduate	No	2226	0	59.0	360.0	1.0
7	LP001056	Male	Yes	2	Not Graduate	No	3881	0	147.0	360.0	0.0
8	LP001059	Male	Yes	2	Graduate	NaN	13633	0	280.0	240.0	1.0
9	LP001067	Male	No	0	Not Graduate	No	2400	2400	123.0	360.0	1.0

This helps in understanding the dataset’s characteristics and forms the basis for more in depth analysis.

DATA OVERVIEW

Drop Loan_id Column

```
In [8]:  1  df.drop('Loan_ID',axis=1,inplace=True)
```

I've drop 'Loan_id ' column from the dataset because this column do not provide meaningful information for analysis.

Description of Dataset

```
In [9]:  1  # Describing the dataset
         2  df.describe()
```

Out[9]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
count	367.000000	367.000000	362.000000	361.000000	338.000000
mean	4805.599455	1569.577657	136.132597	342.537396	0.825444
std	4910.685399	2334.232099	61.366652	65.156643	0.380150
min	0.000000	0.000000	28.000000	6.000000	0.000000
25%	2864.000000	0.000000	100.250000	360.000000	1.000000
50%	3786.000000	1025.000000	125.000000	360.000000	1.000000
75%	5060.000000	2430.500000	158.000000	360.000000	1.000000
max	72529.000000	24000.000000	550.000000	480.000000	1.000000

I've generated a descriptive statistics for the loaded dataset stored in the variable 'df' using the describe method. This Pandas function provide a statistical summary that include count, mean, standard deviation, minimum, 25th percentile(Q1), median(50th percentile or (Q2), 75th percentile(Q3), and maximum for numerical columns in the DataFrame.

DATA OVERVIEW

Information of Dataset

```
In [10]:
1 # Information of the dataset
2 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 367 entries, 0 to 366
Data columns (total 11 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Gender             356 non-null    object
 1   Married            367 non-null    object
 2   Dependents         357 non-null    object
 3   Education          367 non-null    object
 4   Self_Employed      344 non-null    object
 5   ApplicantIncome    367 non-null    int64
 6   CoapplicantIncome  367 non-null    int64
 7   LoanAmount         362 non-null    float64
 8   Loan_Amount_Term   361 non-null    float64
 9   Credit_History     338 non-null    float64
 10  Property_Area      367 non-null    object
dtypes: float64(3), int64(2), object(6)
memory usage: 31.7+ KB
```

Dataset I've obtained information about the dataset using 'df.info'. This method provides a concise summary of the dataset, including the total number of entries, non null values foreach column, data type of each column and memory usage.

Checking Null Values

```
In [11]:
1 # Check for missing values of dataset
2 df.isnull().sum()
3
```

```
Out[11]:
Gender              11
Married              0
Dependents          10
Education            0
Self_Employed       23
ApplicantIncome      0
CoapplicantIncome    0
LoanAmount           5
Loan_Amount_Term     6
Credit_History      29
Property_Area        0
dtype: int64
```

DATA FILTERATION

I've checked for missing values in the dataset using 'df.isnull ().sum()'. This command provide a column wise count of the null or missing values.

Filling Null Values

```
In [12]:
# Filling The Null Values of Gender Column
df.Gender=df.Gender.fillna(df.Gender.mode()[0])

In [13]:
# Filling The Null Values of Dependents Column
df.Dependents=df.Dependents.fillna(df.Dependents.mode()[0])

In [14]:
# Filling The Null Values of Self_Enployed Column
df.Self_Employed=df.Self_Employed.fillna(df.Self_Employed.mode()[0])

In [15]:
# Filling The Null Values of LoanAmount Column
df.LoanAmount=df.LoanAmount.fillna(df.LoanAmount.median())

In [16]:
# Filling The Null Values of Credit_History Column
df.Credit_History=df.Credit_History.fillna(df.Credit_History.mode()[0])

In [17]:
df.Loan_Amount_Term=df.Loan_Amount_Term.fillna(df.Loan_Amount_Term.median())
```

1. df.Gender, df.Dependents, df.Self_Employed and df.Credit_History: I use mode to fill null values in these categorical columns. Categorical variable slack a natural order, making mean or median is less meaningful. Therefore, mode is more appropriate for maintaining the categorical nature of the data.
2. df.LoanAmount and df.Loan_Amount_Term: I use median to fill null values in ' LoanAmount ' and 'Loan_Amount_Term' column. When dealing with numerical data, median is preferred for filling because median is robust to outliers and less sensitive to extreme values.

DATA FILTERATION

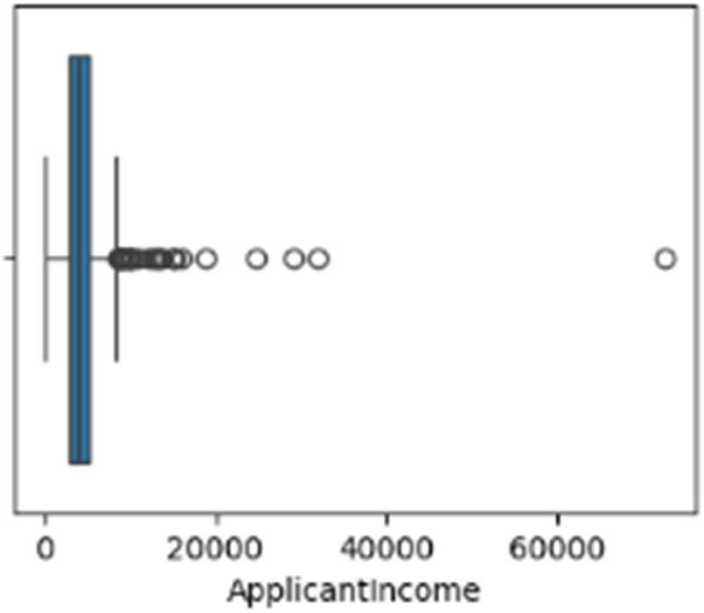
Checking Null Values After Filling

```
In [18]:  1 # Check Null Value After Filling
          2 df.isnull().sum()
```

```
Out[18]: Gender              0
         Married             0
         Dependents          0
         Education           0
         Self_Employed       0
         ApplicantIncome     0
         CoapplicantIncome   0
         LoanAmount          0
         Loan_Amount_Term    0
         Credit_History      0
         Property_Area       0
         dtype: int64
```

All null values are filled as mentioned above.

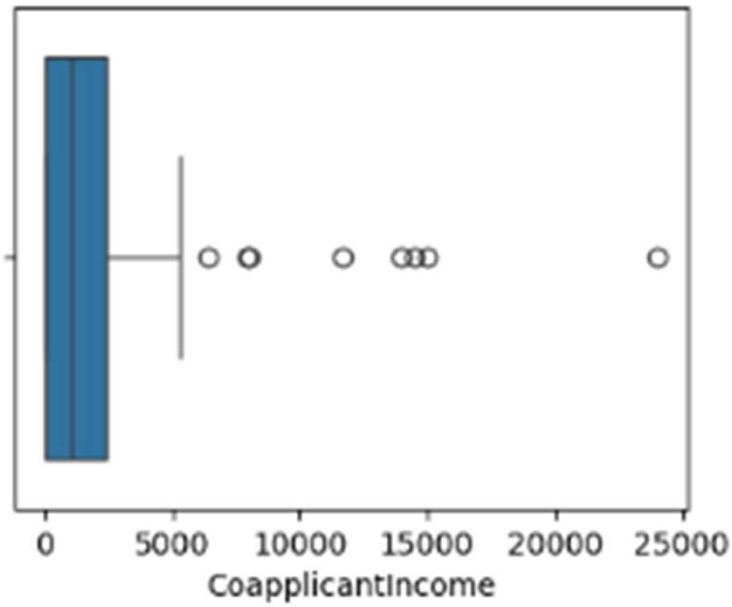
Checking Outlier In Applicant Income Column



The Majority of Outlier in Applicant income columnhas been Identified, and they all exceeds 10000approximately.

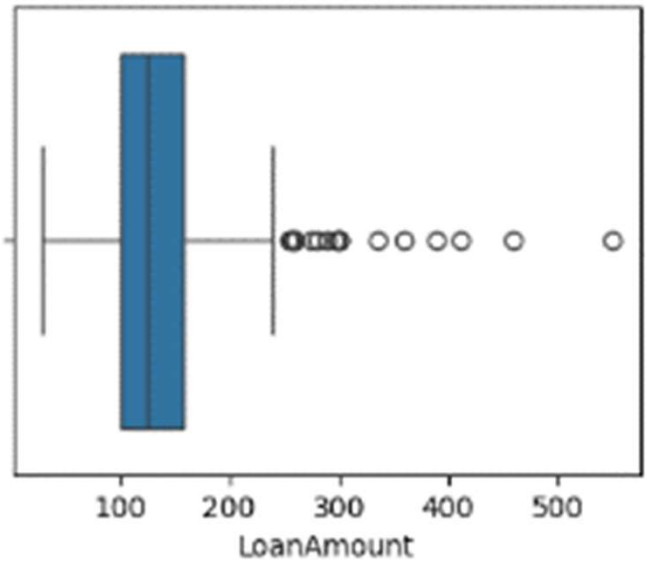
DATA FILTERATION

Checking Outlier In Coapplicant Income Column



The Majority of Outlier in Coapplicant income column has been Identified, and they all exceeds 5200 approximately.

Checking Outlier In Loan Amount Column



The Majority of Outlier in Loan Amount column hasbeen Identified, and they all exceeds 250approximately.

DATA FILTERATION

Removing Outlier In Applicant Income Column

```
In [24]:
# Removing Outlier from the ApplicantIncome column of the dataset.
Q1,Q3 = df.ApplicantIncome.quantile([0.25,0.75])

In [25]:
Q1,Q3
Out[25]: (2864.0, 5060.0)

In [26]:
#  Finding IQR

IQR = Q3-Q1

In [27]:
IQR
Out[27]: 2196.0

In [28]:
# Finding Upper Limit and Lower Limit.

UL = Q3+1.5*(IQR)
LL = Q1-1.5*(IQR)

In [29]:
UL,LL
Out[29]: (8354.0, -430.0)

In [30]:
# TREATING OUTLIERS BY REPLACING THEM WITH
df.ApplicantIncome = np.where(df.ApplicantIncome>UL,UL,df.ApplicantIncome)
```

Outlier is removed successfully in Applicant Income Column.

Removing Outlier In Coapplicant Income Column

```
In [32]:
#  Removing the outlier of CoapplicantIncome column of the dataset
Q1,Q3 = df.CoapplicantIncome.quantile([0.25,0.75])

In [33]:
Q1,Q3
Out[33]: (0.0, 2430.5)

In [34]:
#  Finding IQR

IQR = Q3-Q1

In [35]:
IQR
Out[35]: 2430.5

In [36]:
# Finding Upper Limit and Lower Limit.

UL = Q3+1.5*(IQR)
LL = Q1-1.5*(IQR)

In [37]:
UL,LL
Out[37]: (6076.25, -3645.75)

In [38]:
# TREATING OUTLIERS BY REPLACING THEM WITH

df.CoapplicantIncome = np.where(df.CoapplicantIncome>UL,UL,df.CoapplicantIncome)
```

Outlier is removed successfully in Coapplicant Income Column.

DATA FILTERATION

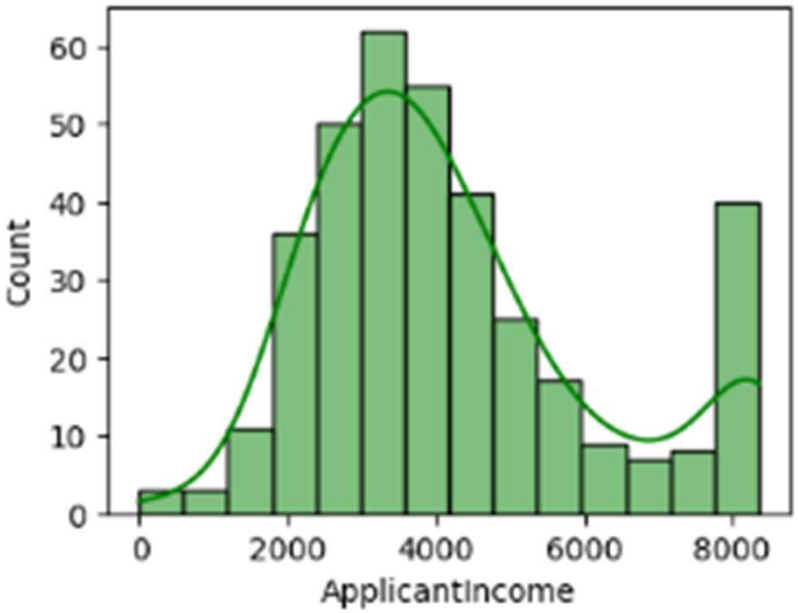
Removing Outlier In Loan Amount Column

```
In [40]:
# Removing the outlier of LoanAmount column of the dataset
Q1,Q3 = df.LoanAmount.quantile([0.25,0.75])

In [41]:
Q1,Q3

Out[41]: (101.0, 157.5)

In [42]:
# Finding IQR
IQR = Q3-Q1

In [43]:
IQR

Out[43]: 56.5

In [44]:
# Finding Upper Limit and Lower Limit.

UL = Q3+1.5*(IQR)
LL = Q1-1.5*(IQR)

In [45]:
# TREATING OUTLIERS BY REPLACING THEM WITH

df.LoanAmount = np.where(df.LoanAmount>UL,UL,df.LoanAmount)
```

Outlier is removed successfully in Loan Amount Column.

DATA VISUALIZATION

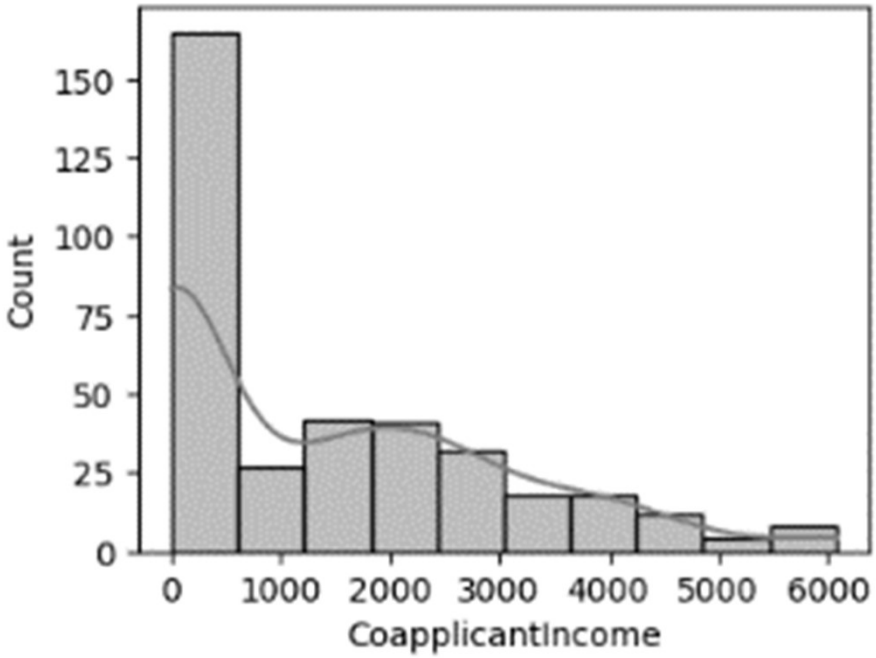
Distribution of Applicant Income



The image shows a histogram overlaid with a line graph, illustrating the distribution of applicant income .The x axis represents ‘Applicant Income’ (ranging from0 to 8000), while the y axis shows Count ”(ranging from 0 to 60). Most applicants fall within the income range of 2000 to 4000 , with an interesting spike at higher incomes. The graph provides insights into the income distribution among a specific group.

DATA VISUALIZATION

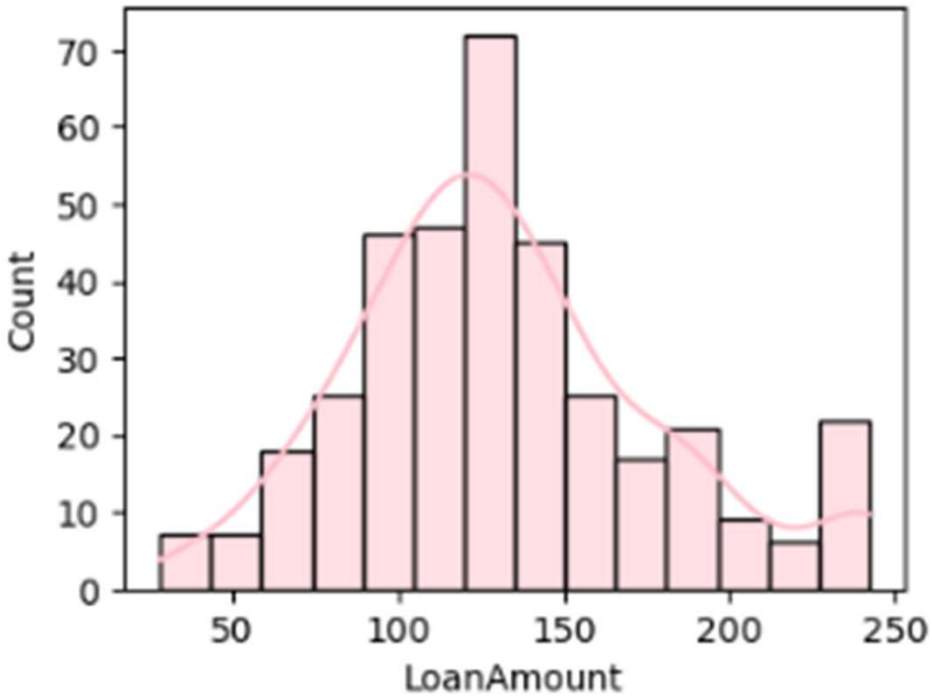
Distribution of Coapplicant Income



This histogram illustrate the distribution of coapplicant income . The x axis represents “Coapplicant Income” (ranging from 0 to 6000), while the y axis shows “Count” (ranging from 0 to 150). A prominent bar reaches up to approximately 150 count for incomes close to zero, indicating many coapplicants have little to no income. The frequency of higher incomes diminishes, with smaller bars representing fewer coapplicants in those income brackets. Overall, this graph provides insights into the financial standing f coapplicants.

DATA VISUALIZATION

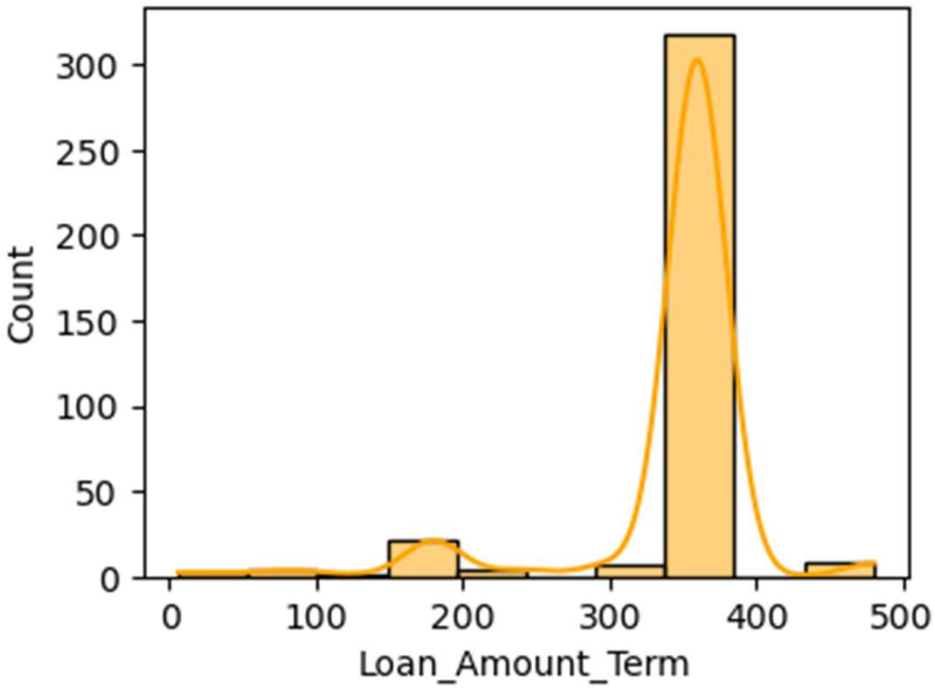
Distribution of Loan Amount



This is a bar graph displaying the distribution of loan amounts . The x axis is Labeled ‘LoanAmount’ and ranges from 0 to 250, while the y axis is labeled “Count” and ranges from 0 to 70. A prominent bar reaches up to approximately 150 count for incomes close to zero, indicating that many coapplicants have little to no income. The frequency of higher incomes diminishes, with smaller bars representing fewer coapplicants in those income brackets. Overall, this graph provides insights into the financial standing of coapplicants.

DATA VISUALIZATION

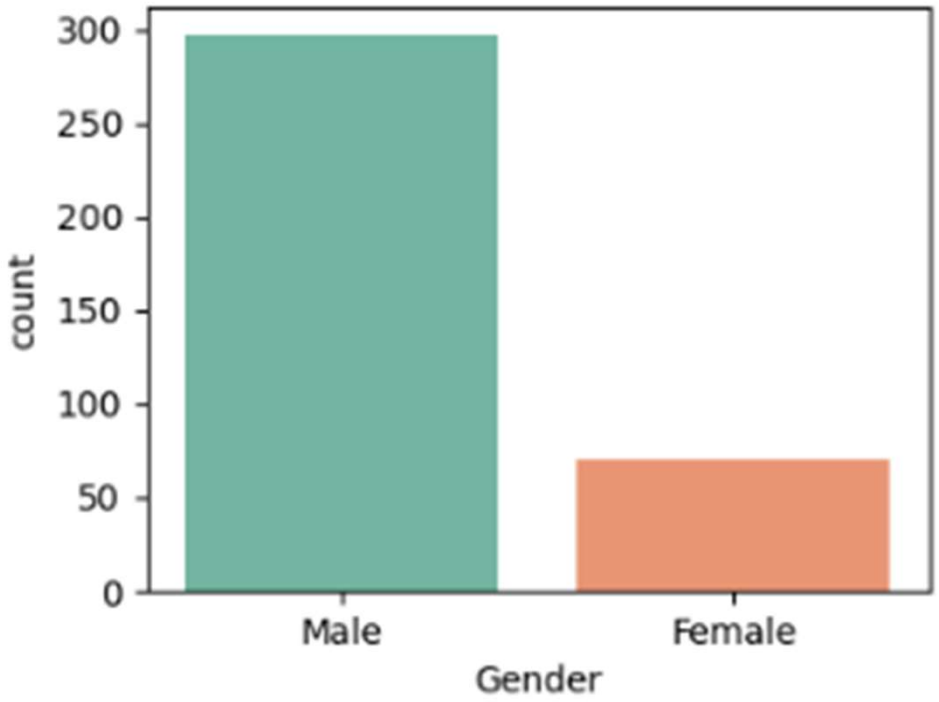
Distribution of Loan Amount Term



The image is a line graph showing the number of loans issued based on the loan term. The x axis labeled "Loan_Amount_Term " likely represents the length of the loan in years. The y axis labeled "Count" represents the number of loans. It appears that shorter loan terms are more common than longer loan terms.

DATA VISUALIZATION

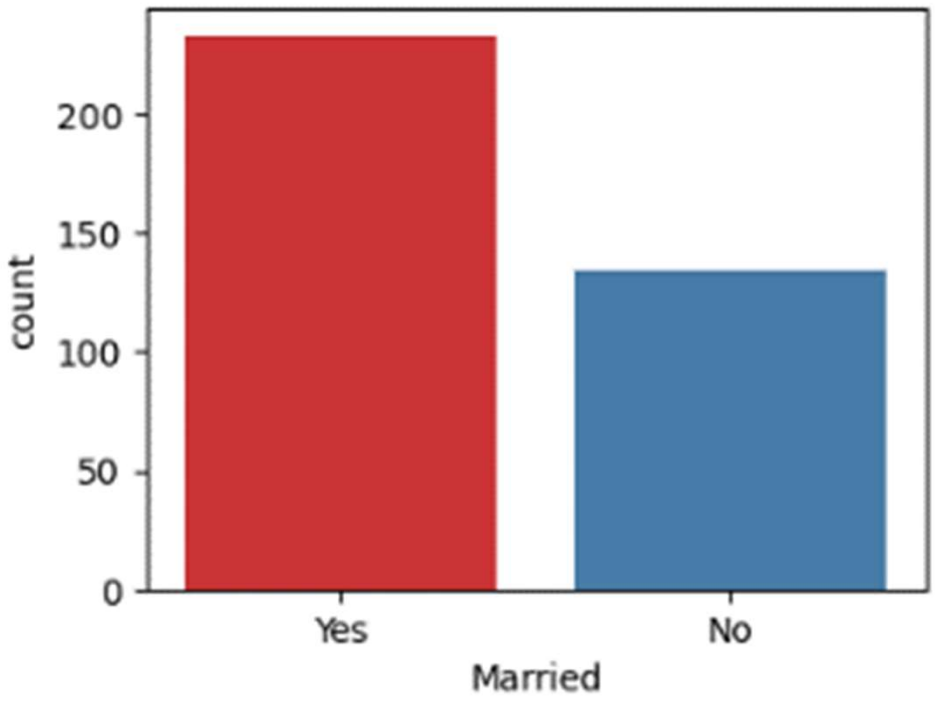
Distribution of Gender



This visualization reveals the distribution of genders among the loan applicants. It appears that there is a noticeable imbalance between male and female applicants. This could be due to various socio-economic factors influencing the willingness of individuals to apply for loans.

DATA VISUALIZATION

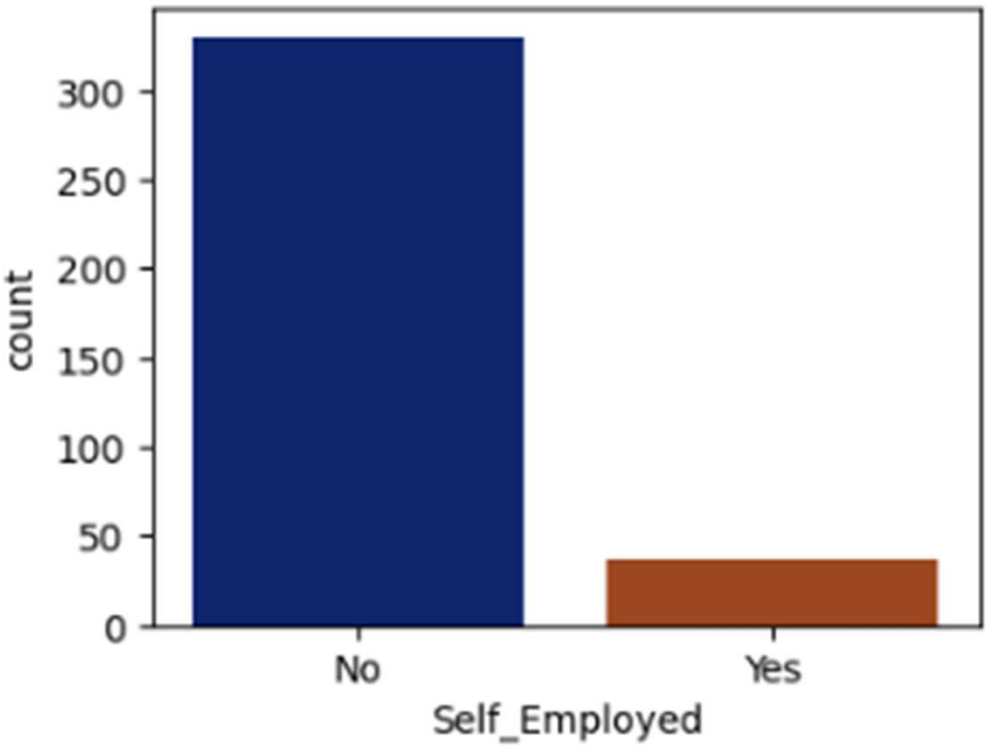
Distribution of Martial Status



This count plot visualizes the distribution of applicants based on their marital status, with categories such as 'Married' and 'Not Married'. There is a noticeable imbalance in the distribution, there are more married applicants compared to unmarried ones. It could reflect societal trends or demographic characteristics of the dataset.

DATA VISUALIZATION

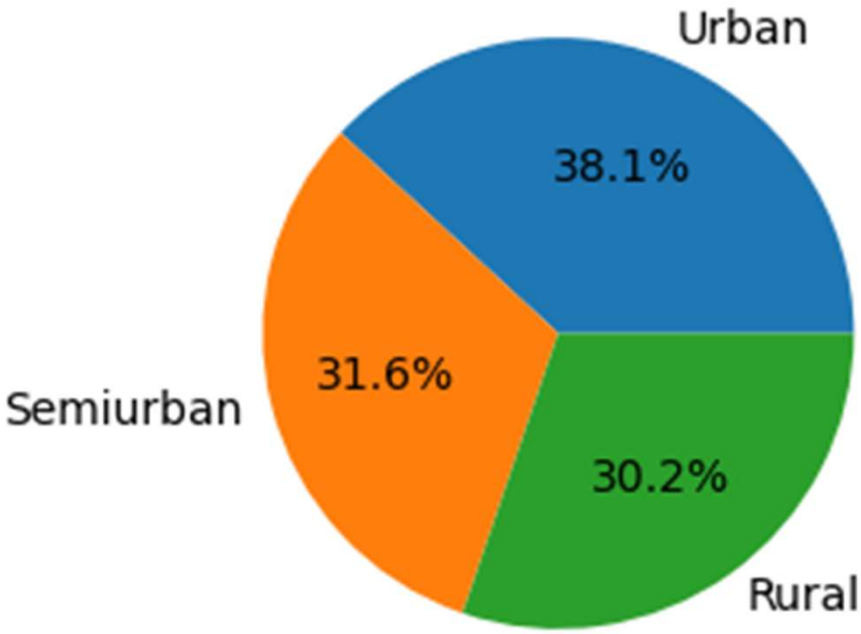
Distribution of Self Employed



This count plot illustrates the distribution of applicants based on their self employment status. Categories may include 'Yes' (indicating self employed) and 'No'(indicating not self employed). There is a significant count in the 'Yes' category, it indicates a not able presence of self-employed individuals or entrepreneurs among the loan applicants. Self employed individuals may have variable income structures compared to salaried individuals. Understanding their distribution provides insights into the diversity of income sources among loan applicants.

DATA VISUALIZATION

Distribution of Property Area



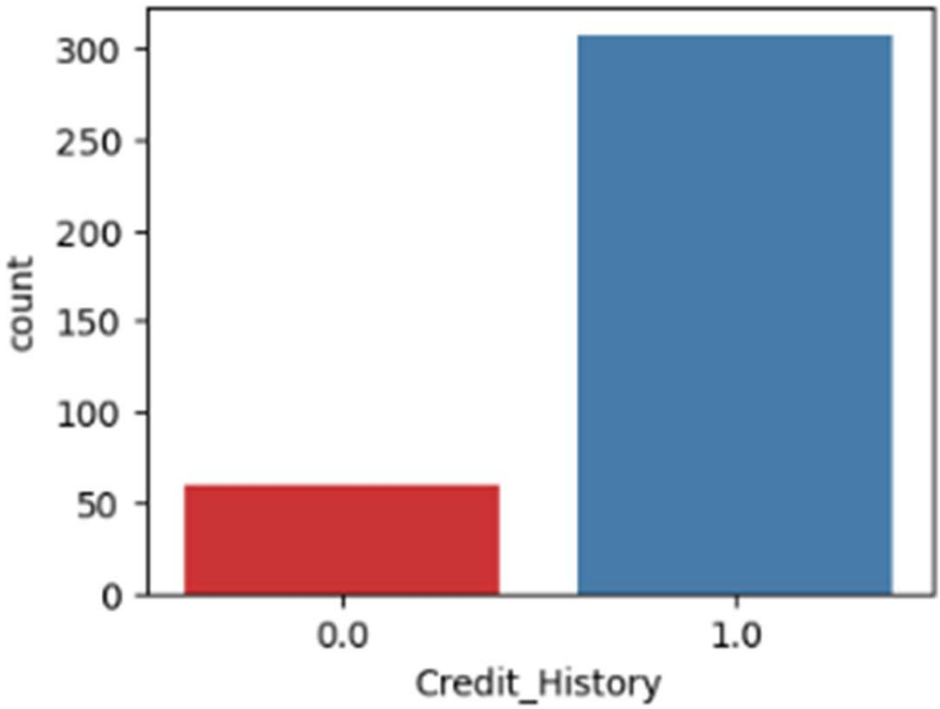
This pie chart divided into three slices labeled "Urban", "Semiurban", and "Rural". The slices represent the distribution of the world's population across these three categories.

* The largest slice, labeled "Urban," is 38.1% of the pie.
* The middle slice, labeled "Semiurban," is 31.6%.
* The smallest slice, labeled "Rural," is 30.2%.

In total, the pie chart adds up to 100%, showing that these three categories encompass the entire world population.

DATA VISUALIZATION

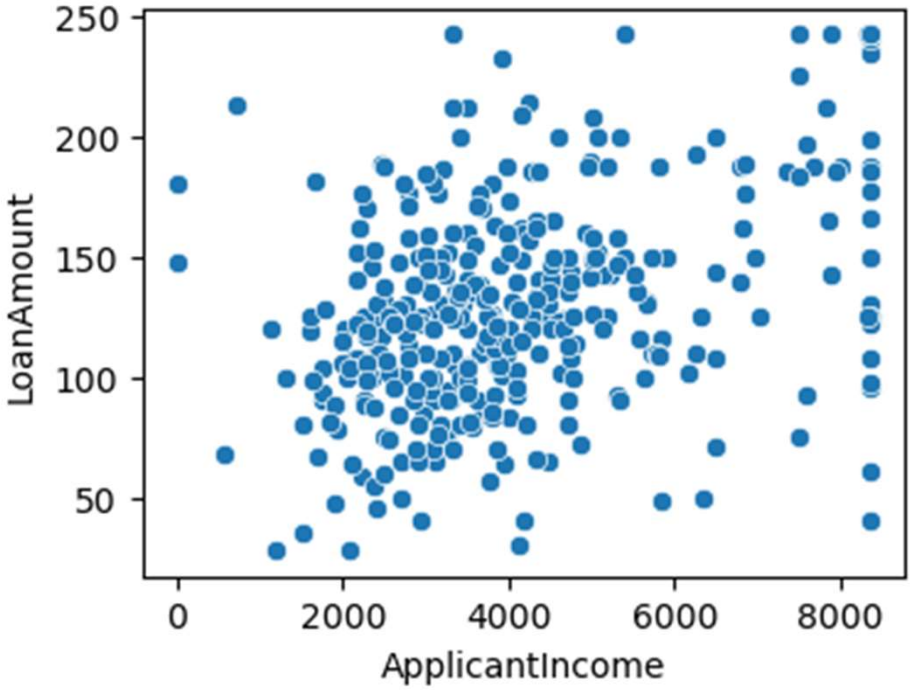
Distribution of Credit History



This count plot illustrates the distribution of applicants based on their credit history. Categories may include1 (indicating a credit history exists) and 'O' (indicating no credit history). A higher count in the ' 1 ' category suggests a larger number of creditworthy applicants with a positive credit history. The distribution provides insights into the prevalence of applicants with and without a credit history. Applicants with no credit history may pose a different risk profile, and this should be considered during the risk assessment process. Credit history is a critical factor in loan approval. Understanding the distribution helps assess the proportion of applicants with a solid credit history, influencing approval rates.

DATA VISUALIZATION

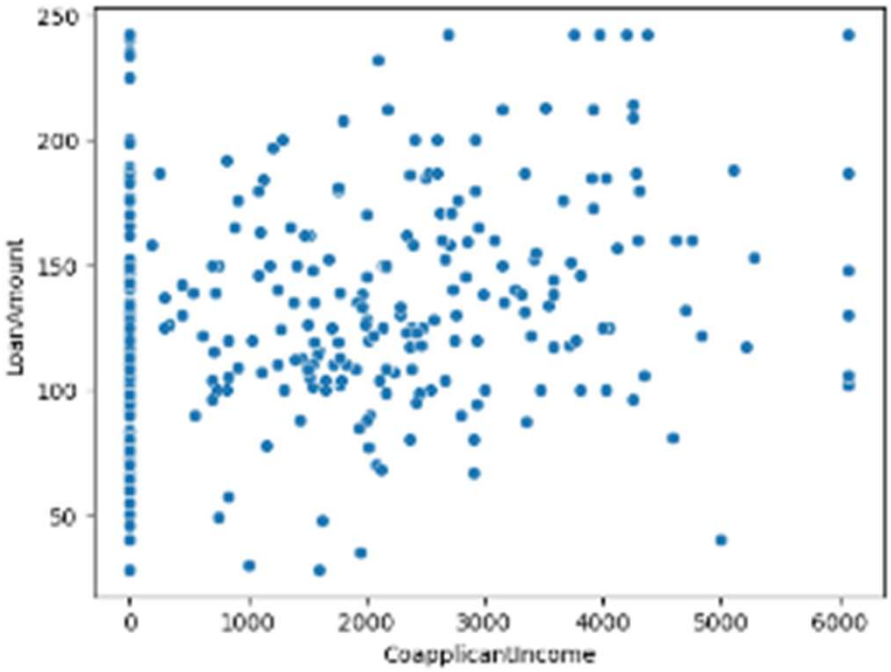
Distribution of Applicant Income and Loan Amount



There is a positive correlation between applicant income and loan amount. This means that as the applicant's income increases, they may be eligible for larger loan amounts. Applicants with higher incomes may qualify for larger loans because they have the financial capacity to repay them.

DATA VISUALIZATION

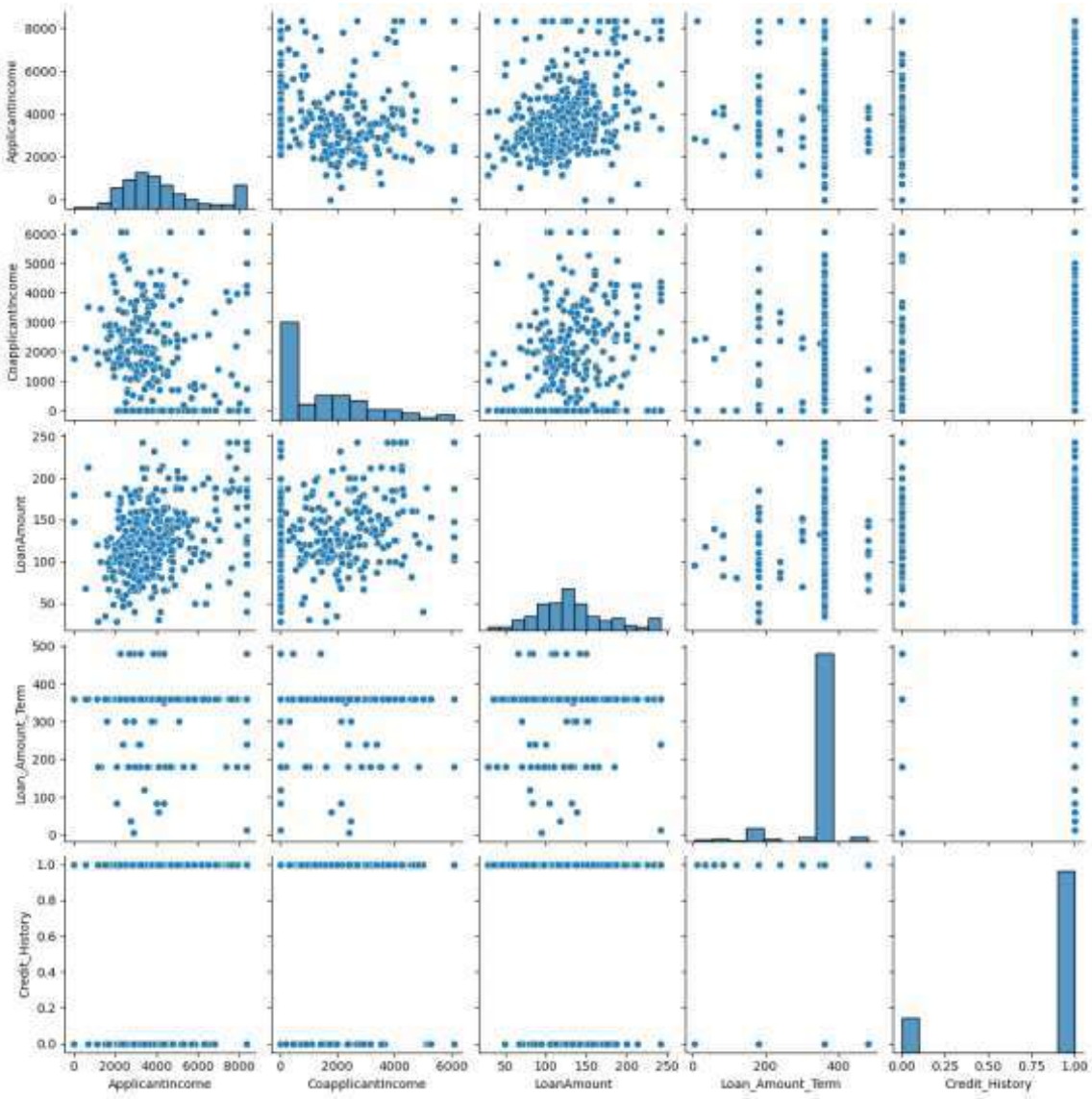
Distribution of Coapplicant Income and Loan Amount



This is a scatter plot graph displaying the relationship between Coapplicant Income and LoanAmount . The y axis represents “ Coapplicant Income ” (ranging from0 to 6000), while the y axis shows “ Loan Amount ”(ranging from 0 to 250). Numerous blue dots represent individual data points scattered across the graph, indicating various combinations of coapplicant income and loan amounts. A scatter plot illustrating the correlation between coapplicant income and loan amount, providing insights into lending trends or patterns

DATA VISUALIZATION

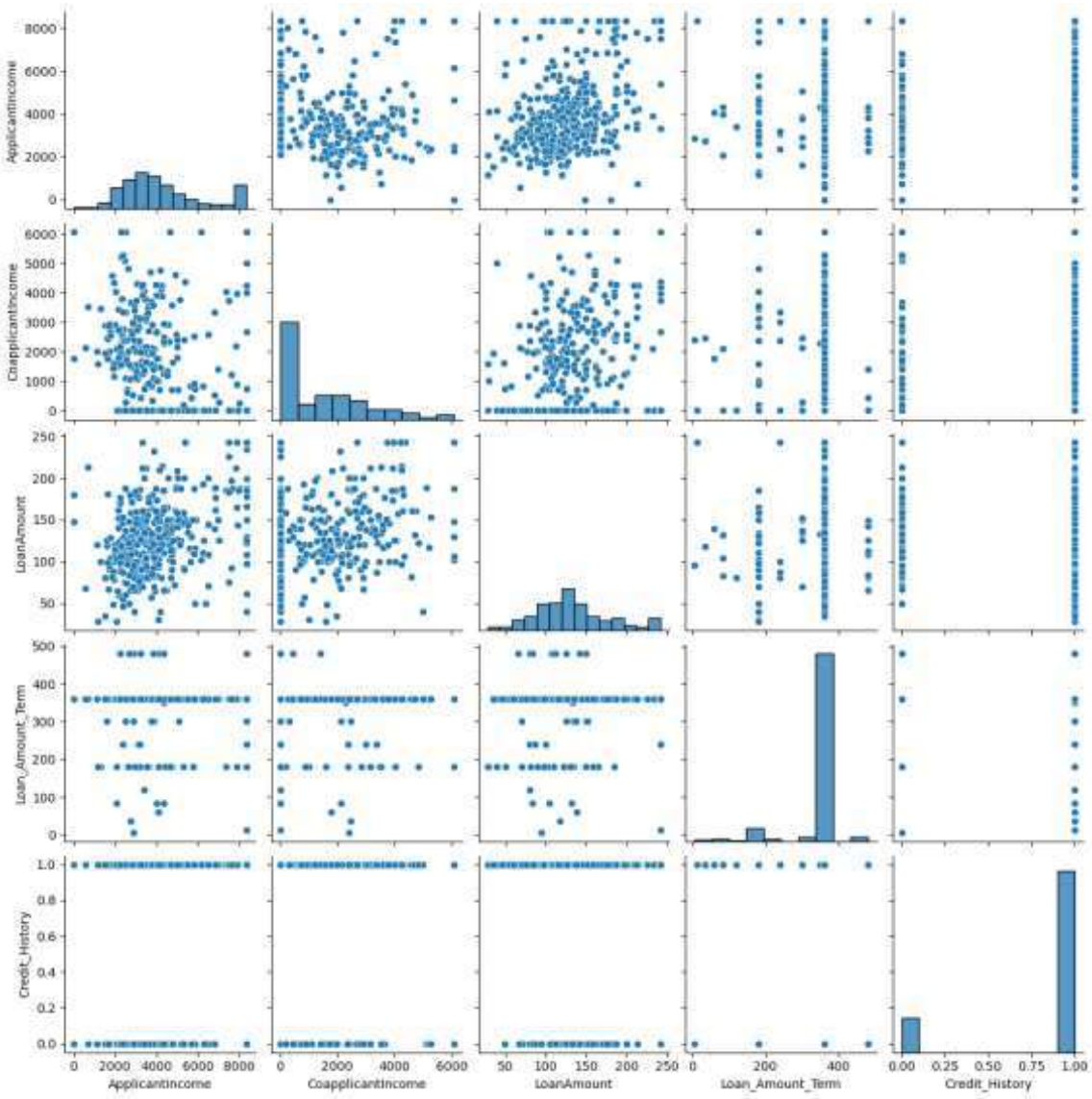
Pair Plot Of Data



The Pair plot analysis reveals a positive correlation between Applicant Income and Loan Amount, suggesting that higher income leads to a higher loan eligibility.

DATA VISUALIZATION

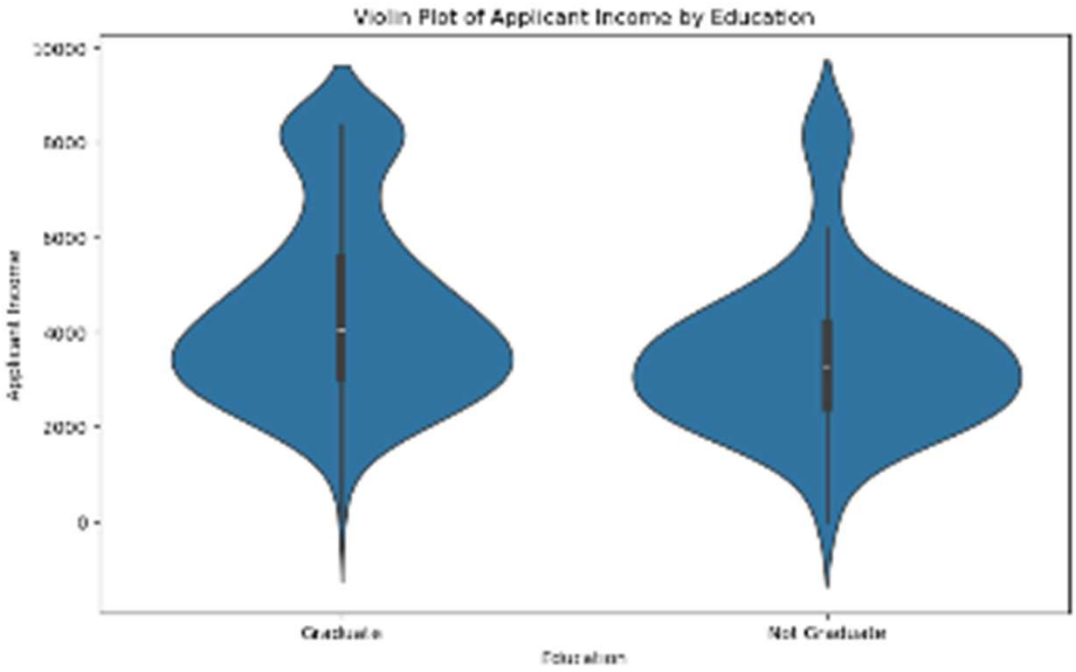
Pair Plot Of Data



The Pair plot analysis reveals a positive correlation between Applicant Income and Loan Amount, suggesting that higher income leads to a higher loan eligibility.

DATA VISUALIZATION

Violin Plot of Applicant Income by Education



This violin plot showing income distribution for applicants with a graduate degree compared to those without. The vertical axis shows income, while the horizontal axis represents education level. The wider the violin shape, the greater the variation in income. The plot suggests that applicants with a graduate degree tend to have higher incomes than those without a graduate degree.

CONCLUSION

Conclusion

After conducting a comprehensive analysis of the loan approval data, several key findings have emerged. Firstly, it is evident that income level and credit score are the most influential factors in determining loan approval. Applicants with higher incomes and better credit scores are more likely to be approved for loans.

Additionally, employment status also plays a significant role, with employed individuals being more likely to receive approval compared to unemployed or self employed applicants. Furthermore, the analysis revealed that debt to income ratio is another crucial factor affecting loan approval, with lower ratios correlating with higher approval rates.

Moreover, there are certain demographic trends worth noting, such as differences in approval rates among various age groups and genders. These findings underscore the importance of fair and unbiased lending practices.

In conclusion, by leveraging the insights gleaned from this analysis, financial institutions can make more informed decisions regarding loan approvals, streamline their processes, and better serve their customers while minimizing risks associated with lending.