

**Project Report**

Bachelor in Computer Applications

Semester-II Introduction to DataScience

**Loan Data Prediction**

By Ashif Raza

2411021240056

Department of Computer Application Alliance University

Chandapura – Anekal Main Road, Anekal Bengaluru – 562 106

April 2025

Introduction

In this project, I am using a **loan dataset** to study and understand how different personal and financial details of a person can affect whether their **loan application is approved or not**. This dataset includes information such as a person’s **income**, **credit score**, **loan amount**, **debt-to-income (DTI) ratio**, **employment status**, and the final result – whether their loan was **approved** or **rejected**.

The main aim of this project is to find out what factors play an important role in loan approval decisions. For example, we want to know whether a person with a higher income has a better chance of getting a loan, or if a low credit score can lead to rejection. By studying this data, we can help banks and financial companies understand the data better and make smarter, faster, and fairer decisions in the future.

To do this, I will perform different types of analysis. First, I will do **Univariate Analysis**, where I will look at each column of data one by one to understand its basic characteristics like average values, most common values, and ranges. Then I will move to **Bivariate Analysis**, where I will study the relationship between two columns, for example, how income affects approval. Finally, I will do **Multivariate Analysis** to explore how multiple factors together influence the loan decision.

This project also includes data visualizations such as bar charts, pie charts, histograms, and scatter plots to make the information easier to understand. I will also use basic statistical tools to support the findings. In the end, the goal is to give a clear and simple explanation of how loan approvals work based on data,

**import** pandas **as** pd **import** numpy **as** np **import** seaborn **as** sns

**import** matplotlib.pyplot **as** plt

data = pd.read\_csv(r"D:\Download\archive (29)\loan\_data.csv")

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| data |  | | | | | | | | | | | |
| \ | survived | | pclass | sex | age | sibsp | | parch | | fare | embarked | class |
| 0 | 0 | | 3 | male | 22.0 | 1 | | 0 | | 7.2500 | S | Third |
| 1 | 1 | | 1 | female | 38.0 | 1 | | 0 | | 71.2833 | C | First |
| 2 | 1 | | 3 | female | 26.0 | 0 | | 0 | | 7.9250 | S | Third |
| 3 | 1 | | 1 | female | 35.0 | 1 | | 0 | | 53.1000 | S | First |
| 4 | 0 | | 3 | male | 35.0 | 0 | | 0 | | 8.0500 | S | Third |
| .. | ... | | ... | ... | ... | ... | | ... | | ... | ... | ... |
| 886 | 0 | | 2 | male | 27.0 | 0 | | 0 | | 13.0000 | S | Second |
| 887 | 1 | | 1 | female | 19.0 | 0 | | 0 | | 30.0000 | S | First |
| 888 | 0 | | 3 | female | NaN | 1 | | 2 | | 23.4500 | S | Third |
| 889 | 1 | | 1 | male | 26.0 | 0 | | 0 | | 30.0000 | C | First |
| 890 | 0 | | 3 | male | 32.0 | 0 | | 0 | | 7.7500 | Q | Third |
|  | who | adult\_male | | deck | embark\_town | | alive | | alone | | | |
| 0 | man | True | | NaN | Southampton | | no | | False | | | |
| 1 | woman | False | | C | Cherbourg | | yes | | False | | | |
| 2 | woman | False | | NaN | Southampton | | yes | | True | | | |
| 3 | woman | False | | C | Southampton | | yes | | False | | | |
| 4 | man | True | | NaN | Southampton | | no | | True | | | |
| .. | ... | ... | | ... | ... | | ... | | ... | | | |
| 886 | man | True | | NaN | Southampton | | no | | True | | | |
| 887 | woman | False | | B | Southampton | | yes | | True | | | |
| 888 | woman | False | | NaN | Southampton | | no | | False | | | |
| 889 | man | True | | C | Cherbourg | | yes | | True | | | |
| 890 | man | True | | NaN | Queenstown | | no | | True | | | |

[891 rows x 15 columns] data.tail

<bound method NDFrame.tail of Text Income \

1. I need a loan to pay for an international vaca... 26556
2. I want to make home improvements like installi... 197392
3. I need a loan for home renovation, including a... 44561
4. I need funds to buy new furniture and applianc... 190363
5. I need a loan to start a small business. 61853

... ... ...

23995 I need funds to pay for my daughter's college ... 195242 23996 I need financial assistance to launch my own Y... 150246 23997 I need money to open a small bookstore and café. 64571

23998 I want to buy a car for my rideshare business ... 115825 23999 I need financial help to cover maternity and d... 180440

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Credit\_Score | Loan\_Amount | DTI\_Ratio | Employment\_Status | Approval |
| 0 | 581 | 8314 | 79.26 | employed | Rejected |
| 1 | 389 | 111604 | 22.14 | employed | Rejected |
| 2 | 523 | 34118 | 45.44 | employed | Rejected |
| 3 | 729 | 118757 | 10.22 | unemployed | Rejected |
| 4 | 732 | 19210 | 44.13 | employed | Approved |
| ... | ... | ... | ... | ... | ... |
| 23995 | 817 | 16403 | 24.32 | employed | Approved |
| 23996 | 729 | 101572 | 9.97 | employed | Rejected |
| 23997 | 650 | 30533 | 57.35 | employed | Rejected |
| 23998 | 418 | 89837 | 10.37 | unemployed | Rejected |
| 23999 | 769 | 15259 | 15.06 | employed | Approved |

[24000 rows x 7 columns]>

*# Save dictionary DataFrame to a CSV file*

data.to\_csv("output\_data.csv", index=False)

*# Select dataset for analysis (Change this to df\_csv, df\_list, etc.)*

df = data

df.info() *# Displays column types, non-null counts*

<class 'pandas.core.frame.DataFrame'> RangeIndex: 24000 entries, 0 to 23999 Data columns (total 7 columns):

# Column Non-Null Count Dtype

* 1. Text 24000 non-null object
  2. Income 24000 non-null int64
  3. Credit\_Score 24000 non-null int64
  4. Loan\_Amount 24000 non-null int64
  5. DTI\_Ratio 24000 non-null float64
  6. Employment\_Status 24000 non-null object
  7. Approval 24000 non-null object dtypes: float64(1), int64(3), object(3)

memory usage: 1.3+ MB

df.describe() *# Statistical summary (only for numerical columns)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Income | Credit\_Score | Loan\_Amount | DTI\_Ratio |
| count | 24000.000000 | 24000.000000 | 24000.000000 | 24000.000000 |
| mean | 110377.552708 | 575.720333 | 44356.154833 | 34.719167 |
| std | 51729.677627 | 159.227621 | 34666.604785 | 32.322471 |
| min | 20001.000000 | 300.000000 | 1005.000000 | 2.530000 |
| 25% | 65635.750000 | 437.000000 | 16212.000000 | 14.507500 |
| 50% | 110464.000000 | 575.000000 | 35207.000000 | 24.860000 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 75% | 155187.000000 | 715.000000 | 65622.750000 | 41.840000 |
| max | 200000.000000 | 850.000000 | 158834.000000 | 246.330000 |

df.isnull().sum() *# Checks for missing values in each column*

Text 0 Income 0

Credit\_Score 0

Loan\_Amount 0

DTI\_Ratio 0

Employment\_Status 0

Approval 0

dtype: int64

df.nunique() *# Shows the number of unique values per column*

|  |  |  |  |
| --- | --- | --- | --- |
| Text 70  Income 22456  Credit\_Score 551  Loan\_Amount 21249  DTI\_Ratio 8033  Employment\_Status 2  Approval 2  dtype: int64 |  | | |
| df.head() |
| Text | Income | Credit\_Score | \ |
| 0 I need a loan to pay for an international vaca... | 26556 | 581 |  |
| 1 I want to make home improvements like installi... | 197392 | 389 |  |
| 2 I need a loan for home renovation, including a... | 44561 | 523 |  |
| 3 I need funds to buy new furniture and applianc... | 190363 | 729 |  |
| 4 I need a loan to start a small business. | 61853 | 732 |  |

Loan\_Amount DTI\_Ratio Employment\_Status Approval

1. 8314 79.26 employed Rejected
2. 111604 22.14 employed Rejected
3. 34118 45.44 employed Rejected
4. 118757 10.22 unemployed Rejected
5. 19210 44.13 employed Approved df.tail()

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | | | Text | Income | \ |
| 23995 | I need funds | to pay | for my daughter's | college ... | 195242 |  |

23996 I need financial assistance to launch my own Y 150246 23997 I need money to open a small bookstore and café. 64571 23998 I want to buy a car for my rideshare business 115825 23999 I need financial help to cover maternity and d 180440

Credit\_Score Loan\_Amount DTI\_Ratio Employment\_Status Approval

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 23995 | 817 |  | 16403 |  | 24.32 |  | employed | Approved |
| 23996 | 729 |  | 101572 |  | 9.97 |  | employed | Rejected |
| 23997 | 650 |  | 30533 |  | 57.35 |  | employed | Rejected |
| 23998 | 418 |  | 89837 |  | 10.37 |  | unemployed | Rejected |
| 23999 | 769 |  | 15259 |  | 15.06 |  | employed | Approved |
| df.iloc[0] |  |  |  |  |  |  |  |  |
| Text |  | I | need a loan | to | pay for | an | international | vaca... |
| Income |  |  |  |  |  |  |  | 26556 |
| Credit\_Score |  |  |  |  |  |  |  | 581 |
| Loan\_Amount |  |  |  |  |  |  |  | 8314 |
| DTI\_Ratio |  |  |  |  |  |  |  | 79.26 |

Employment\_Status employed

Approval Rejected

Name: 0, dtype: object

df['Text'] = df['Income'] \* 0.5 *#Creating new Column*

df

Text Income \

1. I need a loan to pay for an international vaca... 26556

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | I | want | to make home improvements like installi... | 197392 |
| 2 | I | need | a loan for home renovation, including a... | 44561 |
| 3 | I | need | funds to buy new furniture and applianc... | 190363 |
| 4 |  |  | I need a loan to start a small business. | 61853 |
| ... |  |  | ... | ... |
| 23995 | I | need | funds to pay for my daughter's college ... | 195242 |
| 23996 | I | need | financial assistance to launch my own Y... | 150246 |

23997 I need money to open a small bookstore and café. 64571 23998 I want to buy a car for my rideshare business ... 115825 23999 I need financial help to cover maternity and d... 180440

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Credit\_Score | Loan\_Amount | DTI\_Ratio | Employment\_Status | Approval |
| 0 | 581 | 8314 | 79.26 | employed | Rejected |
| 1 | 389 | 111604 | 22.14 | employed | Rejected |
| 2 | 523 | 34118 | 45.44 | employed | Rejected |
| 3 | 729 | 118757 | 10.22 | unemployed | Rejected |
| 4 | 732 | 19210 | 44.13 | employed | Approved |
| ... | ... | ... | ... | ... | ... |
| 23995 | 817 | 16403 | 24.32 | employed | Approved |
| 23996 | 729 | 101572 | 9.97 | employed | Rejected |
| 23997 | 650 | 30533 | 57.35 | employed | Rejected |
| 23998 | 418 | 89837 | 10.37 | unemployed | Rejected |
| 23999 | 769 | 15259 | 15.06 | employed | Approved |

[24000 rows x 7 columns]

df.drop(columns=['Income'], inplace=True)*#drop a column*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| df |  | | | | | |
|  | Text |  | Loan\_Amount | DTI\_Ratio | Employment\_Status | Approval |
| 0 | 13278.0 |  | 8314 | 79.26 | employed | Rejected |
| 1 | 98696.0 |  | 111604 | 22.14 | employed | Rejected |
| 2 | 22280.5 |  | 34118 | 45.44 | employed | Rejected |
| 3 | 95181.5 |  | 118757 | 10.22 | unemployed | Rejected |
| 4 | 30926.5 |  | 19210 | 44.13 | employed | Approved |
| ... | ... |  | ... | ... | ... | ... |
| 23995 | 97621.0 |  | 16403 | 24.32 | employed | Approved |
| 23996 | 75123.0 |  | 101572 | 9.97 | employed | Rejected |
| 23997 | 32285.5 |  | 30533 | 57.35 | employed | Rejected |
| 23998 | 57912.5 |  | 89837 | 10.37 | unemployed | Rejected |
| 23999 | 90220.0 |  | 15259 | 15.06 | employed | Approved |
| [24000 | rows x | 5 | columns] |  |  |  |

df.loc[0]

Text 13278.0

Loan\_Amount 8314

DTI\_Ratio 79.26

Employment\_Status employed Approval Rejected Name: 0, dtype: object

df.loc[1]

Text 98696.0

Loan\_Amount 111604

DTI\_Ratio 22.14

Employment\_Status employed Approval Rejected Name: 1, dtype: object

df.loc[1,'Approval'] 'Rejected' df.loc[1,'DTI\_Ratio'] 22.14

df.loc[[1,2],['DTI\_Ratio','Approval']]

DTI\_Ratio Approval

1. 22.14 Rejected
2. 45.44 Rejected df.iloc[[1,2],[1,2]]

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | Loan\_Amount  111604 | | DTI\_Ratio  22.14 | | | | |
| 2 | 34118 | | 45.44 | | | | |
| df | |  |  |  |  |  |  |
|  | | Text |  | Loan\_Amount | DTI\_Ratio | Employment\_Status | Approval |
| 0 | | 13278.0 |  | 8314 | 79.26 | employed | Rejected |
| 1 | | 98696.0 |  | 111604 | 22.14 | employed | Rejected |
| 2 | | 22280.5 |  | 34118 | 45.44 | employed | Rejected |
| 3 | | 95181.5 |  | 118757 | 10.22 | unemployed | Rejected |
| 4 | | 30926.5 |  | 19210 | 44.13 | employed | Approved |
| ... | | ... |  | ... | ... | ... | ... |
| 23995 | | 97621.0 |  | 16403 | 24.32 | employed | Approved |
| 23996 | | 75123.0 |  | 101572 | 9.97 | employed | Rejected |
| 23997 | | 32285.5 |  | 30533 | 57.35 | employed | Rejected |
| 23998 | | 57912.5 |  | 89837 | 10.37 | unemployed | Rejected |
| 23999 | | 90220.0 |  | 15259 | 15.06 | employed | Approved |
| [24000 | | rows x | 5 | columns] |  |  |  |

df.iloc[1]

Text 98696.0

Loan\_Amount 111604

DTI\_Ratio 22.14

Employment\_Status employed Approval Rejected Name: 1, dtype: object

*# HANDLING MISSING VALUE*

*# Check for missing values*

df.isnull().sum()

Text 0

Loan\_Amount 0

DTI\_Ratio 0

Employment\_Status 0

Approval 0

dtype: int64

*# Drop rows with missing values* df\_dropped = df.dropna() df\_dropped

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Text | Loan\_Amount | DTI\_Ratio | Employment\_Status | Approval |
| 0 | 13278.0 | 8314 | 79.26 | employed | Rejected |
| 1 | 98696.0 | 111604 | 22.14 | employed | Rejected |
| 2 | 22280.5 | 34118 | 45.44 | employed | Rejected |
| 3 | 95181.5 | 118757 | 10.22 | unemployed | Rejected |
| 4 | 30926.5 | 19210 | 44.13 | employed | Approved |
| ... | ... | ... | ... | ... | ... |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 23995 | 97621.0 | 16403 | 24.32 | employed | Approved |
| 23996 | 75123.0 | 101572 | 9.97 | employed | Rejected |
| 23997 | 32285.5 | 30533 | 57.35 | employed | Rejected |
| 23998 | 57912.5 | 89837 | 10.37 | unemployed | Rejected |
| 23999 | 90220.0 | 15259 | 15.06 | employed | Approved |

[24000 rows x 5 columns]

*# Fill numerical missing values with mean*

df['Text'].fillna(df['Text'].mean(), inplace=True)

*# Fill categorical missing values with mode* df['Loan\_Amount'].fillna(df['Loan\_Amount'].mode()[0], inplace=True) df['DTI\_Ratio'].fillna(df['DTI\_Ratio'].mode()[0], inplace=True)

C:\Users\ASUS\AppData\Local\Temp\ipykernel\_18788\2374607493.py:2: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.

The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.

df['Text'].fillna(df['Text'].mean(), inplace=True) C:\Users\ASUS\AppData\Local\Temp\ipykernel\_18788\2374607493.py:5: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.

The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.

df['Loan\_Amount'].fillna(df['Loan\_Amount'].mode()[0], inplace=True) C:\Users\ASUS\AppData\Local\Temp\ipykernel\_18788\2374607493.py:6: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.

The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using

'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.

df['DTI\_Ratio'].fillna(df['DTI\_Ratio'].mode()[0], inplace=True) df1=pd.DataFrame(data)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| df1 |  | | | | |
|  | Text | Loan\_Amount | DTI\_Ratio | Employment\_Status | Approval |
| 0 | 13278.0 | 8314 | 79.26 | employed | Rejected |
| 1 | 98696.0 | 111604 | 22.14 | employed | Rejected |
| 2 | 22280.5 | 34118 | 45.44 | employed | Rejected |
| 3 | 95181.5 | 118757 | 10.22 | unemployed | Rejected |
| 4 | 30926.5 | 19210 | 44.13 | employed | Approved |
| ... | ... | ... | ... | ... | ... |
| 23995 | 97621.0 | 16403 | 24.32 | employed | Approved |
| 23996 | 75123.0 | 101572 | 9.97 | employed | Rejected |
| 23997 | 32285.5 | 30533 | 57.35 | employed | Rejected |
| 23998 | 57912.5 | 89837 | 10.37 | unemployed | Rejected |
| 23999 | 90220.0 | 15259 | 15.06 | employed | Approved |

[24000 rows x 5 columns] df1.drop(columns=['Loan\_Amount'],inplace=True)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| df |  | | | | |
|  | Text | Loan\_Amount | DTI\_Ratio | Employment\_Status | Approval |
| 0 | 13278.0 | 8314 | 79.26 | employed | Rejected |
| 1 | 98696.0 | 111604 | 22.14 | employed | Rejected |
| 2 | 22280.5 | 34118 | 45.44 | employed | Rejected |
| 3 | 95181.5 | 118757 | 10.22 | unemployed | Rejected |
| 4 | 30926.5 | 19210 | 44.13 | employed | Approved |
| ... | ... | ... | ... | ... | ... |
| 23995 | 97621.0 | 16403 | 24.32 | employed | Approved |
| 23996 | 75123.0 | 101572 | 9.97 | employed | Rejected |
| 23997 | 32285.5 | 30533 | 57.35 | employed | Rejected |
| 23998 | 57912.5 | 89837 | 10.37 | unemployed | Rejected |
| 23999 | 90220.0 | 15259 | 15.06 | employed | Approved |

[24000 rows x 5 columns]

**import** pandas **as** pd

**import** numpy **as** np

**from** datetime **import** datetime, timedelta

*# Apply forward fill*

df\_ffill = df.fillna(method='ffill') df\_ffill

C:\Users\ASUS\AppData\Local\Temp\ipykernel\_18788\4147793657.py:2: FutureWarning: DataFrame.fillna with 'method' is deprecated and will raise in a future version. Use obj.ffill() or obj.bfill() instead.

df\_ffill = df.fillna(method='ffill')

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Text | Loan\_Amount | DTI\_Ratio | Employment\_Status | Approval |
| 0 | 13278.0 | 8314 | 79.26 | employed | Rejected |
| 1 | 98696.0 | 111604 | 22.14 | employed | Rejected |
| 2 | 22280.5 | 34118 | 45.44 | employed | Rejected |
| 3 | 95181.5 | 118757 | 10.22 | unemployed | Rejected |
| 4 | 30926.5 | 19210 | 44.13 | employed | Approved |
| ... | ... | ... | ... | ... | ... |
| 23995 | 97621.0 | 16403 | 24.32 | employed | Approved |
| 23996 | 75123.0 | 101572 | 9.97 | employed | Rejected |
| 23997 | 32285.5 | 30533 | 57.35 | employed | Rejected |
| 23998 | 57912.5 | 89837 | 10.37 | unemployed | Rejected |
| 23999 | 90220.0 | 15259 | 15.06 | employed | Approved |

[24000 rows x 5 columns]

*# Apply backward fill*

df\_bfill = df.fillna(method='bfill') df\_bfill

C:\Users\ASUS\AppData\Local\Temp\ipykernel\_18788\2542315512.py:2: FutureWarning: DataFrame.fillna with 'method' is deprecated and will raise in a future version. Use obj.ffill() or obj.bfill() instead.

df\_bfill = df.fillna(method='bfill')

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Text | Loan\_Amount | DTI\_Ratio | Employment\_Status | Approval |
| 0 | 13278.0 | 8314 | 79.26 | employed | Rejected |
| 1 | 98696.0 | 111604 | 22.14 | employed | Rejected |
| 2 | 22280.5 | 34118 | 45.44 | employed | Rejected |
| 3 | 95181.5 | 118757 | 10.22 | unemployed | Rejected |
| 4 | 30926.5 | 19210 | 44.13 | employed | Approved |
| ... | ... | ... | ... | ... | ... |
| 23995 | 97621.0 | 16403 | 24.32 | employed | Approved |
| 23996 | 75123.0 | 101572 | 9.97 | employed | Rejected |
| 23997 | 32285.5 | 30533 | 57.35 | employed | Rejected |
| 23998 | 57912.5 | 89837 | 10.37 | unemployed | Rejected |
| 23999 | 90220.0 | 15259 | 15.06 | employed | Approved |

[24000 rows x 5 columns]

print('Original DataFrame:') print(df)

print('Forward Fill DataFrame:') print(df\_ffill)

print('Backward Fill DataFrame:') print(df\_bfill)

Original DataFrame:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Text | Loan\_Amount | DTI\_Ratio | Employment\_Status | Approval |
| 0 | 13278.0 | 8314 | 79.26 | employed | Rejected |
| 1 | 98696.0 | 111604 | 22.14 | employed | Rejected |
| 2 | 22280.5 | 34118 | 45.44 | employed | Rejected |
| 3 | 95181.5 | 118757 | 10.22 | unemployed | Rejected |
| 4 | 30926.5 | 19210 | 44.13 | employed | Approved |
| ... | ... | ... | ... | ... | ... |
| 23995 | 97621.0 | 16403 | 24.32 | employed | Approved |
| 23996 | 75123.0 | 101572 | 9.97 | employed | Rejected |
| 23997 | 32285.5 | 30533 | 57.35 | employed | Rejected |
| 23998 | 57912.5 | 89837 | 10.37 | unemployed | Rejected |
| 23999 | 90220.0 | 15259 | 15.06 | employed | Approved |

[24000 rows x 5 columns] Forward Fill DataFrame:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Text | Loan\_Amount | DTI\_Ratio | Employment\_Status | Approval |
| 0 | 13278.0 | 8314 | 79.26 | employed | Rejected |
| 1 | 98696.0 | 111604 | 22.14 | employed | Rejected |
| 2 | 22280.5 | 34118 | 45.44 | employed | Rejected |
| 3 | 95181.5 | 118757 | 10.22 | unemployed | Rejected |
| 4 | 30926.5 | 19210 | 44.13 | employed | Approved |
| ... | ... | ... | ... | ... | ... |
| 23995 | 97621.0 | 16403 | 24.32 | employed | Approved |
| 23996 | 75123.0 | 101572 | 9.97 | employed | Rejected |
| 23997 | 32285.5 | 30533 | 57.35 | employed | Rejected |
| 23998 | 57912.5 | 89837 | 10.37 | unemployed | Rejected |
| 23999 | 90220.0 | 15259 | 15.06 | employed | Approved |

[24000 rows x 5 columns] Backward Fill DataFrame:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Text | Loan\_Amount | DTI\_Ratio | Employment\_Status | Approval |
| 0 | 13278.0 | 8314 | 79.26 | employed | Rejected |
| 1 | 98696.0 | 111604 | 22.14 | employed | Rejected |
| 2 | 22280.5 | 34118 | 45.44 | employed | Rejected |
| 3 | 95181.5 | 118757 | 10.22 | unemployed | Rejected |
| 4 | 30926.5 | 19210 | 44.13 | employed | Approved |
| ... | ... | ... | ... | ... | ... |
| 23995 | 97621.0 | 16403 | 24.32 | employed | Approved |
| 23996 | 75123.0 | 101572 | 9.97 | employed | Rejected |
| 23997 | 32285.5 | 30533 | 57.35 | employed | Rejected |
| 23998 | 57912.5 | 89837 | 10.37 | unemployed | Rejected |
| 23999 | 90220.0 | 15259 | 15.06 | employed | Approved |

[24000 rows x 5 columns]

*# Handling missing values based on your exact dataset columns*

*# Fill numerical missing values with mean* df['Text'].fillna(df['Text'].mean(), inplace=True) df['Loan\_Amount'].fillna(df['Loan\_Amount'].mean(), inplace=True) df['DTI\_Ratio'].fillna(df['DTI\_Ratio'].mean(), inplace=True)

*# Fill categorical missing values with mode* df['Employment\_Status'].fillna(df['Employment\_Status'].mode()[0], inplace=True)

df['Approval'].fillna(df['Approval'].mode()[0], inplace=True)

*# Display the cleaned DataFrame*

df

C:\Users\ASUS\AppData\Local\Temp\ipykernel\_18788\1880280673.py:4: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.

The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.

df['Text'].fillna(df['Text'].mean(), inplace=True) C:\Users\ASUS\AppData\Local\Temp\ipykernel\_18788\1880280673.py:5: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.

The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.

df['Loan\_Amount'].fillna(df['Loan\_Amount'].mean(), inplace=True) C:\Users\ASUS\AppData\Local\Temp\ipykernel\_18788\1880280673.py:6: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.

The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using

'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.

df['DTI\_Ratio'].fillna(df['DTI\_Ratio'].mean(), inplace=True) C:\Users\ASUS\AppData\Local\Temp\ipykernel\_18788\1880280673.py:9: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.

The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.

df['Employment\_Status'].fillna(df['Employment\_Status'].mode()[0], inplace=True) C:\Users\ASUS\AppData\Local\Temp\ipykernel\_18788\1880280673.py:10: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.

The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.

df['Approval'].fillna(df['Approval'].mode()[0], inplace=True)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Text | Loan\_Amount | DTI\_Ratio | Employment\_Status | Approval |
| 0 | 13278.0 | 8314 | 79.26 | employed | Rejected |
| 1 | 98696.0 | 111604 | 22.14 | employed | Rejected |
| 2 | 22280.5 | 34118 | 45.44 | employed | Rejected |
| 3 | 95181.5 | 118757 | 10.22 | unemployed | Rejected |
| 4 | 30926.5 | 19210 | 44.13 | employed | Approved |
| ... | ... | ... | ... | ... | ... |
| 23995 | 97621.0 | 16403 | 24.32 | employed | Approved |
| 23996 | 75123.0 | 101572 | 9.97 | employed | Rejected |
| 23997 | 32285.5 | 30533 | 57.35 | employed | Rejected |
| 23998 | 57912.5 | 89837 | 10.37 | unemployed | Rejected |
| 23999 | 90220.0 | 15259 | 15.06 | employed | Approved |

[24000 rows x 5 columns]

*# Removing duplicate Patient IDs*

df = df.drop\_duplicates(subset=['Employment\_Status'], keep='first')

*# Display the DataFrame after removing duplicates*

df

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Text | Loan\_Amount | DTI\_Ratio | Employment\_Status | Approval |
| 0 | 13278.0 | 8314 | 79.26 | employed | Rejected |
| 3 | 95181.5 | 118757 | 10.22 | unemployed | Rejected |

*# Handling Outliers in 'Age' using IQR Method*

Q1 = df['Loan\_Amount'].quantile(0.25) Q3 = df['Loan\_Amount'].quantile(0.75)

IQR = Q3 - Q1

lower\_bound = Q1 - 1.5 \* IQR upper\_bound = Q3 + 1.5 \* IQR

*# Filtering out the outliers*

df = df[(df['Loan\_Amount'] >= lower\_bound) & (df['Loan\_Amount'] <= upper\_bound)]

*# Display the DataFrame after outlier removal*

df

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Text | Loan\_Amount | DTI\_Ratio | Employment\_Status | Approval |
| 0 | 13278.0 | 8314 | 79.26 | employed | Rejected |
| 3 | 95181.5 | 118757 | 10.22 | unemployed | Rejected |

*# Standardizing Data Formats*

*# Convert categorical text columns to lowercase and remove extra spaces* df['Employment\_Status'] = df['Employment\_Status'].str.lower().str.strip() df['Approval'] = df['Approval'].str.lower().str.strip()

*# Display the updated DataFrame*

df

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Text | Loan\_Amount | DTI\_Ratio | Employment\_Status | Approval |
| 0 | 13278.0 | 8314 | 79.26 | employed | rejected |
| 3 | 95181.5 | 118757 | 10.22 | unemployed | rejected |

*# Fixing inconsistent data entries*

*# Ensure 'Employment\_Status' column is consistent*

df['Employment\_Status'] = df['Employment\_Status'].replace({ 'Employed': 'employed',

'Unemployed': 'unemployed',

'Self-employed': 'self-employed', 'unemployed': 'unemployed', 'employed': 'employed',

'self-employed': 'self-employed'

})

*# Ensure 'Approval' column is consistent*

df['Approval'] = df['Approval'].replace({ 'approved': 'Approved',

'rejected': 'Rejected', 'Approved': 'Approved', 'Rejected': 'Rejected'

})

*# Display the updated DataFrame*

df

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Text | Loan\_Amount | DTI\_Ratio | Employment\_Status | Approval |
| 0 | 13278.0 | 8314 | 79.26 | employed | Rejected |
| 3 | 95181.5 | 118757 | 10.22 | unemployed | Rejected |

*# Correcting Data Types*

df['Text'] = df['Text'].astype(float) df['Loan\_Amount'] = df['Loan\_Amount'].astype(float) df['DTI\_Ratio'] = df['DTI\_Ratio'].astype(int)

df['Employment\_Status'] = df['Employment\_Status'].astype(str) df

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Text | Loan\_Amount | DTI\_Ratio | Employment\_Status | Approval |
| 0 | 13278.0 | 8314.0 | 79 | employed | Rejected |
| 3 | 95181.5 | 118757.0 | 10 | unemployed | Rejected |

*# Saving the cleaned data to CSV* df.to\_csv('cleaned\_data.csv', index=False) print('Cleaned dataset saved as cleaned\_data.csv')

Cleaned dataset saved as cleaned\_data.csv

**from** sklearn.preprocessing **import** MinMaxScaler print(df.columns.tolist())

['Text', 'Loan\_Amount', 'DTI\_Ratio', 'Employment\_Status', 'Approval']

**from** sklearn.preprocessing **import** MinMaxScaler

*# Columns to normalize based on your DataFrame*

columns\_to\_normalize = ['Text', 'Loan\_Amount', 'DTI\_Ratio']

*# Normalize the selected columns*

scaler = MinMaxScaler()

normalized\_data = scaler.fit\_transform(df[columns\_to\_normalize])

*# Add normalized columns to the DataFrame*

df[['Normalized\_Text', 'Normalized\_Loan\_Amount', 'Normalized\_DTI\_Ratio']] =

normalized\_data

*# Print the updated DataFrame with original and normalized columns* print(df[['Text', 'Normalized\_Text', 'Loan\_Amount', 'Normalized\_Loan\_Amount', 'DTI\_Ratio', 'Normalized\_DTI\_Ratio']])

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| \ | Text | Normalized\_Text | Loan\_Amount | Normalized\_Loan\_Amount | DTI\_Ratio |
| 0 | 13278.0 | 0.0 | 8314.0 | 0.0 | 79 |
| 3 | 95181.5 | 1.0 | 118757.0 | 1.0 | 10 |

Normalized\_DTI\_Ratio

0 1.0

3 0.0

**import** numpy **as** np

**def** min\_max\_normalization(data): X\_min = min(data)

X\_max = max(data)

LST=[(x - X\_min) / (X\_max - X\_min) **for** x **in** data]

**return** LST

data = [10, 20, 30, 40, 50]

normalized\_data = min\_max\_normalization(data)

print("Original Data:", data)

print("Min-Max Normalized Data:", normalized\_data)

Original Data: [10, 20, 30, 40, 50]

Min-Max Normalized Data: [0.0, 0.25, 0.5, 0.75, 1.0]

**from** sklearn.preprocessing **import** StandardScaler data = [[10], [20], [30], [40], [50]]

scaler = StandardScaler()

standardized\_data = scaler.fit\_transform(data) print(standardized\_data)

[[-1.41421356]

[-0.70710678]

[ 0. ]

[ 0.70710678]

[ 1.41421356]]

*# Define Z-score normalization function*

**def** z\_score\_normalization(data): mean = sum(data) / len(data)

std\_dev = (sum([(x - mean) \*\* 2 **for** x **in** data]) / len(data)) \*\* 0.5

**return** [(x - mean) / std\_dev **for** x **in** data]

*# Apply Z-score normalization to numerical columns*

df['Z\_Text'] = z\_score\_normalization(df['Text']) df['Z\_Loan\_Amount'] = z\_score\_normalization(df['Loan\_Amount']) df['Z\_DTI\_Ratio'] = z\_score\_normalization(df['DTI\_Ratio'])

*# Print the updated DataFrame with original and standardized columns* print(df[['Text', 'Z\_Text', 'Loan\_Amount', 'Z\_Loan\_Amount', 'DTI\_Ratio', 'Z\_DTI\_Ratio']])

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Text | Z\_Text | Loan\_Amount | Z\_Loan\_Amount | DTI\_Ratio | Z\_DTI\_Ratio |
| 0 | 13278.0 | -1.0 | 8314.0 | -1.0 | 79 | 1.0 |
| 3 | 95181.5 | 1.0 | 118757.0 | 1.0 | 10 | -1.0 |

**import** numpy **as** np

**import** pandas **as** pd

data = {'Values': [1, 10, 100, 1000, 10000]}

df = pd.DataFrame(data)

*# Apply Log Transformation*

df['Log\_Values'] = np.log(df['Values']) df['Log\_Values']

|  |  |
| --- | --- |
| 0 | 0.000000 |
| 1 | 2.302585 |
| 2 | 4.605170 |
| 3 | 6.907755 |
| 4 | 9.210340 |

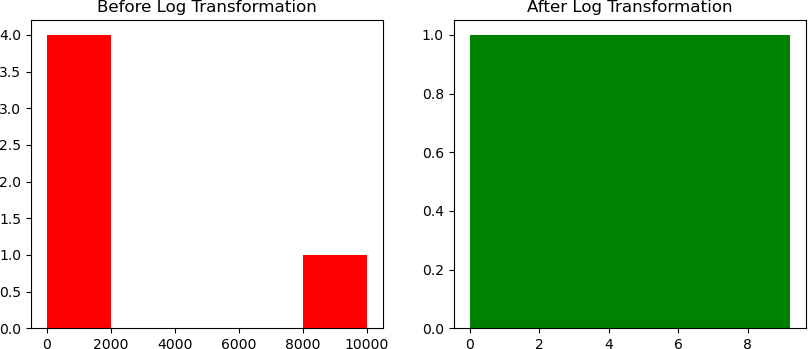
Name: Log\_Values, dtype: float64

*# Plot before and after transformation* **import** matplotlib.pyplot **as** plt plt.figure(figsize=(10, 4))

plt.subplot(1, 2, 1) plt.hist(df['Values'], bins=5, color='red') plt.title("Before Log Transformation") plt.subplot(1, 2, 2)

plt.hist(df['Log\_Values'], bins=5, color='green') plt.title("After Log Transformation")

plt.show()



**import** pandas **as** pd

**from** sklearn.preprocessing **import** LabelEncoder

*# Initialize your DataFrame*

data = {'Text': [13278.0, 98696.0, 22280.5, 95181.5, 30926.5],

'Loan\_Amount': [8314, 111604, 34118, 118757, 19210],

'DTI\_Ratio': [79.26, 22.14, 45.44, 10.22, 44.13],

'Employment\_Status': ['employed', 'employed', 'employed', 'unemployed', 'employed'],

'Approval': ['Rejected', 'Rejected', 'Rejected', 'Rejected', 'Approved']}

df = pd.DataFrame(data)

*# Initialize the label encoder*

encoder = LabelEncoder()

*# Apply Label Encoding to 'Employment\_Status' and 'Approval' columns* df['Employment\_Status\_Encoded'] = encoder.fit\_transform(df['Employment\_Status']) df['Approval\_Encoded'] = encoder.fit\_transform(df['Approval'])

*# Display the updated DataFrame*

print(df)

Text Loan\_Amount DTI\_Ratio Employment\_Status Approval \

1. 13278.0 8314 79.26 employed Rejected
2. 98696.0 111604 22.14 employed Rejected
3. 22280.5 34118 45.44 employed Rejected
4. 95181.5 118757 10.22 unemployed Rejected
5. 30926.5 19210 44.13 employed Approved

Employment\_Status\_Encoded Approval\_Encoded

0 0 1

|  |  |  |
| --- | --- | --- |
| 1 | 0 | 1 |
| 2 | 0 | 1 |
| 3 | 1 | 1 |
| 4 | 0 | 0 |

**def** label\_encoding(categories):

unique\_values = list(set(categories)) *# Get unique values* encoding\_dict = {val: idx **for** idx, val **in** enumerate(unique\_values)} **return** [encoding\_dict[val] **for** val **in** categories]

categories = ["Red", "Blue", "Green", "Blue", "Red"] encoded\_data = label\_encoding(categories) print("Label Encoded Data:", encoded\_data)

Label Encoded Data: [2, 1, 0, 1, 2]

**import** pandas **as** pd

*# Your existing DataFrame*

data = {'Text': [13278.0, 98696.0, 22280.5, 95181.5, 30926.5],

'Loan\_Amount': [8314, 111604, 34118, 118757, 19210],

'DTI\_Ratio': [79.26, 22.14, 45.44, 10.22, 44.13],

'Employment\_Status': ['employed', 'employed', 'employed', 'unemployed', 'employed'],

'Approval': ['Rejected', 'Rejected', 'Rejected', 'Rejected', 'Approved']}

df = pd.DataFrame(data)

*# Apply One-Hot Encoding to 'Employment\_Status' and 'Approval' columns*

df\_encoded = pd.get\_dummies(df, columns=['Employment\_Status', 'Approval'])

*# Display the encoded DataFrame*

print(df\_encoded)

Text Loan\_Amount DTI\_Ratio Employment\_Status\_employed \

0 13278.0 8314 79.26 True

1 98696.0 111604 22.14 True

2 22280.5 34118 45.44 True

3 95181.5 118757 10.22 False

4 30926.5 19210 44.13 True

Employment\_Status\_unemployed Approval\_Approved Approval\_Rejected

0 False False True

1 False False True

2 False False True

3 True False True

4 False True False

**import** pandas **as** pd **import** numpy **as** np **import** seaborn **as** sns

**from** sklearn.preprocessing **import** MinMaxScaler, StandardScaler, LabelEncoder,

OneHotEncoder

**import** pandas **as** pd

*# Load the Heart Failure dataset (assuming it's in a CSV file)*

df = pd.read\_csv(r"D:\Download\archive (29)\loan\_data.csv")

*# Print the original dataset* print("Original Dataset:") print(df)

Original Dataset:

Text Income \

1. I need a loan to pay for an international vaca... 26556
2. I want to make home improvements like installi... 197392
3. I need a loan for home renovation, including a... 44561
4. I need funds to buy new furniture and applianc... 190363
5. I need a loan to start a small business. 61853

... ... ...

23995 I need funds to pay for my daughter's college ... 195242 23996 I need financial assistance to launch my own Y... 150246 23997 I need money to open a small bookstore and café. 64571 23998 I want to buy a car for my rideshare business ... 115825 23999 I need financial help to cover maternity and d... 180440

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Credit\_Score | Loan\_Amount | DTI\_Ratio | Employment\_Status | Approval |
| 0 | 581 | 8314 | 79.26 | employed | Rejected |
| 1 | 389 | 111604 | 22.14 | employed | Rejected |
| 2 | 523 | 34118 | 45.44 | employed | Rejected |
| 3 | 729 | 118757 | 10.22 | unemployed | Rejected |
| 4 | 732 | 19210 | 44.13 | employed | Approved |
| ... | ... | ... | ... | ... | ... |
| 23995 | 817 | 16403 | 24.32 | employed | Approved |
| 23996 | 729 | 101572 | 9.97 | employed | Rejected |
| 23997 | 650 | 30533 | 57.35 | employed | Rejected |
| 23998 | 418 | 89837 | 10.37 | unemployed | Rejected |
| 23999 | 769 | 15259 | 15.06 | employed | Approved |

[24000 rows x 7 columns] df.info()

<class 'pandas.core.frame.DataFrame'> RangeIndex: 24000 entries, 0 to 23999 Data columns (total 7 columns):

# Column Non-Null Count Dtype

* 1. Text 24000 non-null object
  2. Income 24000 non-null int64
  3. Credit\_Score 24000 non-null int64
  4. Loan\_Amount 24000 non-null int64
  5. DTI\_Ratio 24000 non-null float64
  6. Employment\_Status 24000 non-null object
  7. Approval 24000 non-null object dtypes: float64(1), int64(3), object(3)

memory usage: 1.3+ MB

df.isnull().sum() Text 0

Income 0

Credit\_Score 0

Loan\_Amount 0

DTI\_Ratio 0

Employment\_Status 0

Approval 0

dtype: int64

**import** pandas **as** pd

*# Your existing DataFrame*

data = {'Text': [13278.0, 98696.0, 22280.5, 95181.5, 30926.5],

'Loan\_Amount': [8314, 111604, 34118, 118757, 19210],

'DTI\_Ratio': [79.26, 22.14, 45.44, 10.22, 44.13],

'Employment\_Status': ['employed', 'employed', 'employed', 'unemployed', 'employed'],

'Approval': ['Rejected', 'Rejected', 'Rejected', 'Rejected', 'Approved']}

df = pd.DataFrame(data)

*# Fill missing values in 'Employment\_Status' column with the mode (most frequent) value* df['Employment\_Status'].fillna(df['Employment\_Status'].mode()[0], inplace=True)

*# Fill missing values in 'Approval' column with the mode (most frequent) value*

df['Approval'].fillna(df['Approval'].mode()[0], inplace=True)

*# For numerical columns (like 'Text', 'Loan\_Amount', 'DTI\_Ratio'), fill missing values with the median*

df['Text'].fillna(df['Text'].median(), inplace=True) df['Loan\_Amount'].fillna(df['Loan\_Amount'].median(), inplace=True) df['DTI\_Ratio'].fillna(df['DTI\_Ratio'].median(), inplace=True)

*# Print the updated DataFrame*

print(df)

Text Loan\_Amount DTI\_Ratio Employment\_Status Approval

1. 13278.0 8314 79.26 employed Rejected
2. 98696.0 111604 22.14 employed Rejected
3. 22280.5 34118 45.44 employed Rejected
4. 95181.5 118757 10.22 unemployed Rejected
5. 30926.5 19210 44.13 employed Approved

C:\Users\ASUS\AppData\Local\Temp\ipykernel\_18788\752589039.py:13: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.

The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.

df['Employment\_Status'].fillna(df['Employment\_Status'].mode()[0], inplace=True) C:\Users\ASUS\AppData\Local\Temp\ipykernel\_18788\752589039.py:16: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.

The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.

df['Approval'].fillna(df['Approval'].mode()[0], inplace=True) C:\Users\ASUS\AppData\Local\Temp\ipykernel\_18788\752589039.py:19: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.

The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.

df['Text'].fillna(df['Text'].median(), inplace=True) C:\Users\ASUS\AppData\Local\Temp\ipykernel\_18788\752589039.py:20: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.

The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.

df['Loan\_Amount'].fillna(df['Loan\_Amount'].median(), inplace=True) C:\Users\ASUS\AppData\Local\Temp\ipykernel\_18788\752589039.py:21: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.

The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.

df['DTI\_Ratio'].fillna(df['DTI\_Ratio'].median(), inplace=True)

*# Convert 'Approval' column to category type*

df['Approval'] = df['Approval'].astype('category')

*# Add 'Unknown' category only if it's not already there*

**if** 'Unknown' **not in** df['Approval'].cat.categories:

df['Approval'] = df['Approval'].cat.add\_categories(['Unknown'])

*# Fill missing values in 'Approval' with 'Unknown'*

df['Approval'].fillna('Unknown', inplace=True)

*# Display the updated DataFrame*

print(df)

Text Loan\_Amount DTI\_Ratio Employment\_Status Approval

1. 13278.0 8314 79.26 employed Rejected
2. 98696.0 111604 22.14 employed Rejected
3. 22280.5 34118 45.44 employed Rejected
4. 95181.5 118757 10.22 unemployed Rejected
5. 30926.5 19210 44.13 employed Approved

C:\Users\ASUS\AppData\Local\Temp\ipykernel\_18788\138766751.py:9: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series

through chained assignment using an inplace method.

The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.

df['Approval'].fillna('Unknown', inplace=True) df.isnull().sum()

Text 0

Loan\_Amount 0

DTI\_Ratio 0

Employment\_Status 0

Approval 0

dtype: int64

**from** sklearn.preprocessing **import** MinMaxScaler

**import** pandas **as** pd

*# Your DataFrame*

data = {'Text': [13278.0, 98696.0, 22280.5, 95181.5, 30926.5],

'Loan\_Amount': [8314, 111604, 34118, 118757, 19210],

'DTI\_Ratio': [79.26, 22.14, 45.44, 10.22, 44.13],

'Employment\_Status': ['employed', 'employed', 'employed', 'unemployed', 'employed'],

'Approval': ['Rejected', 'Rejected', 'Rejected', 'Rejected', 'Approved']}

df = pd.DataFrame(data)

*# Initialize the MinMaxScaler*

scaler\_minmax = MinMaxScaler()

*# Normalize selected numeric columns ('Loan\_Amount', 'DTI\_Ratio')* df[['Loan\_Amount\_normalized']] = scaler\_minmax.fit\_transform(df[['Loan\_Amount']]) df[['DTI\_Ratio\_normalized']] = scaler\_minmax.fit\_transform(df[['DTI\_Ratio']])

*# Display the updated DataFrame*

print(df)

Text Loan\_Amount DTI\_Ratio Employment\_Status Approval \

1. 13278.0 8314 79.26 employed Rejected
2. 98696.0 111604 22.14 employed Rejected
3. 22280.5 34118 45.44 employed Rejected

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 3 | 95181.5 | 118757 | 10.22 | unemployed | Rejected |
| 4 | 30926.5 | 19210 | 44.13 | employed | Approved |

Loan\_Amount\_normalized DTI\_Ratio\_normalized

0 0.000000 1.000000

1 0.935234 0.172654

2 0.233641 0.510139

3 1.000000 0.000000

4 0.098657 0.491165

df['DTI\_Ratio\_normalized']

|  |  |
| --- | --- |
| 0 | 1.000000 |
| 1 | 0.172654 |
| 2 | 0.510139 |
| 3 | 0.000000 |
| 4 | 0.491165 |

Name: DTI\_Ratio\_normalized, dtype: float64

**from** sklearn.preprocessing **import** StandardScaler

**import** pandas **as** pd

*# Your DataFrame*

data = {'Text': [13278.0, 98696.0, 22280.5, 95181.5, 30926.5],

'Loan\_Amount': [8314, 111604, 34118, 118757, 19210],

'DTI\_Ratio': [79.26, 22.14, 45.44, 10.22, 44.13],

'Employment\_Status': ['employed', 'employed', 'employed', 'unemployed', 'employed'],

'Approval': ['Rejected', 'Rejected', 'Rejected', 'Rejected', 'Approved']}

df = pd.DataFrame(data)

*# Initialize the StandardScaler*

scaler\_standard = StandardScaler()

*# Apply StandardScaler to selected numeric columns ('Loan\_Amount' and 'DTI\_Ratio')*

df[['Loan\_Amount\_standardized']] = scaler\_standard.fit\_transform(df[['Loan\_Amount']]) df[['DTI\_Ratio\_standardized']] = scaler\_standard.fit\_transform(df[['DTI\_Ratio']])

*# Display the updated DataFrame*

print(df)

**from** sklearn.preprocessing **import** StandardScaler df[['Loan\_Amount\_standardized']] = scaler\_standard.fit\_transform(df[['Loan\_Amount']])

NameError Traceback (most recent call last) Cell In[199], line 2

1 from sklearn.preprocessing import StandardScaler

----> 2 df[['Loan\_Amount\_standardized']] = scaler\_standard.fit\_transform(df[['Loan\_Amount']])

NameError: name 'scaler\_standard' is not defined

**import** numpy **as** np

**import** pandas **as** pd

*# Your DataFrame*

data = {'Text': [13278.0, 98696.0, 22280.5, 95181.5, 30926.5],

'Loan\_Amount': [8314, 111604, 34118, 118757, 19210],

'DTI\_Ratio': [79.26, 22.14, 45.44, 10.22, 44.13],

'Employment\_Status': ['employed', 'employed', 'employed', 'unemployed', 'employed'],

'Approval': ['Rejected', 'Rejected', 'Rejected', 'Rejected', 'Approved']}

df = pd.DataFrame(data)

*# Apply log transformation to selected numeric columns* df['Loan\_Amount\_log'] = np.log1p(df['Loan\_Amount']) df['DTI\_Ratio\_log'] = np.log1p(df['DTI\_Ratio'])

*# Display the DataFrame to show the log-transformed values*

print(df[['Loan\_Amount', 'Loan\_Amount\_log', 'DTI\_Ratio', 'DTI\_Ratio\_log']])

Loan\_Amount Loan\_Amount\_log DTI\_Ratio DTI\_Ratio\_log

0 8314 9.025816 79.26 4.385271

1 111604 11.622721 22.14 3.141563

2 34118 10.437610 45.44 3.838161

3 118757 11.684843 10.22 2.417698

4 19210 9.863238 44.13 3.809547

df['DTI\_Ratio']

|  |  |
| --- | --- |
| 0 | 79.26 |
| 1 | 22.14 |
| 2 | 45.44 |
| 3 | 10.22 |
| 4 | 44.13 |

Name: DTI\_Ratio, dtype: float64

df = pd.get\_dummies(df, columns=['Loan\_Amount\_log', 'DTI\_Ratio\_log'], drop\_first=True)

df

Text Loan\_Amount DTI\_Ratio Employment\_Status Approval \

1. 13278.0 8314 79.26 employed Rejected
2. 98696.0 111604 22.14 employed Rejected
3. 22280.5 34118 45.44 employed Rejected
4. 95181.5 118757 10.22 unemployed Rejected
5. 30926.5 19210 44.13 employed Approved

Loan\_Amount\_log\_9.863238310628443 Loan\_Amount\_log\_10.437609692852554 \

0 False False

1 False False

2 False True

3 False False

4 True False

Loan\_Amount\_log\_11.622721130793112 Loan\_Amount\_log\_11.68484308804879 \

0 False False

1 True False

2 False False

3 False True

4 False False

DTI\_Ratio\_log\_3.1415627217655304 DTI\_Ratio\_log\_3.80954721383891 \

0 False False

1 True False

2 False False

3 False False

4 False True

DTI\_Ratio\_log\_3.8381611568296905 DTI\_Ratio\_log\_4.385271364838771

0 False True

1 False False

2 True False

3 False False

4 False False

print("Transformed Dataset:") df.head()

Transformed Dataset:

Text Loan\_Amount DTI\_Ratio Employment\_Status Approval \

1. 13278.0 8314 79.26 employed Rejected
2. 98696.0 111604 22.14 employed Rejected
3. 22280.5 34118 45.44 employed Rejected
4. 95181.5 118757 10.22 unemployed Rejected
5. 30926.5 19210 44.13 employed Approved

Loan\_Amount\_log\_9.863238310628443 Loan\_Amount\_log\_10.437609692852554 \

0 False False

|  |  |  |
| --- | --- | --- |
| 1 | False | False |
| 2 | False | True |
| 3 | False | False |
| 4 | True | False |

Loan\_Amount\_log\_11.622721130793112 Loan\_Amount\_log\_11.68484308804879 \

0 False False

1 True False

2 False False

3 False True

4 False False

DTI\_Ratio\_log\_3.1415627217655304 DTI\_Ratio\_log\_3.80954721383891 \

0 False False

1 True False

2 False False

3 False False

4 False True

DTI\_Ratio\_log\_3.8381611568296905 DTI\_Ratio\_log\_4.385271364838771

0 False True

1 False False

2 True False

3 False False

4 False False

*#BOXPLOT*

**import** matplotlib.pyplot **as** plt

*# Box plot for 'Loan\_Amount' from your DataFrame*

plt.boxplot(df['Loan\_Amount'])

*# Labels and title*

plt.ylabel('Loan Amount') plt.title('Loan Amount Distribution')

*# Show plot*

plt.show()



**import** pandas **as** pd

*# Creating a DataFrame with your actual structure*

data = {

'Text': [13278.0, 98696.0, 22280.5, 95181.5, 30926.5], *# Example values*

*from your 'Text' column*

'Loan\_Amount': [8314, 111604, 34118, 118757, 19210], *# Loan\_Amount values*

'DTI\_Ratio': [79.26, 22.14, 45.44, 10.22, 44.13], *# DTI\_Ratio values*

'Employment\_Status': ['employed', 'employed', 'employed', 'unemployed', 'employed'], *# Employment status*

'Approval': ['Rejected', 'Rejected', 'Rejected', 'Rejected', 'Approved']

*# Approval status*

}

df = pd.DataFrame(data)

*# Display the DataFrame*

print(df)

Text Loan\_Amount DTI\_Ratio Employment\_Status Approval

1. 13278.0 8314 79.26 employed Rejected
2. 98696.0 111604 22.14 employed Rejected
3. 22280.5 34118 45.44 employed Rejected

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 3 | 95181.5 | 118757 | 10.22 | unemployed | Rejected |
| 4 | 30926.5 | 19210 | 44.13 | employed | Approved |

**import** matplotlib.pyplot **as** plt

*# Prepare data for box plot using numerical columns from your DataFrame* data = [df['Text'], df['Loan\_Amount'], df['DTI\_Ratio']] *# Numeric columns from your dataset*

*# Create box plot*

plt.figure(figsize=(12,6))

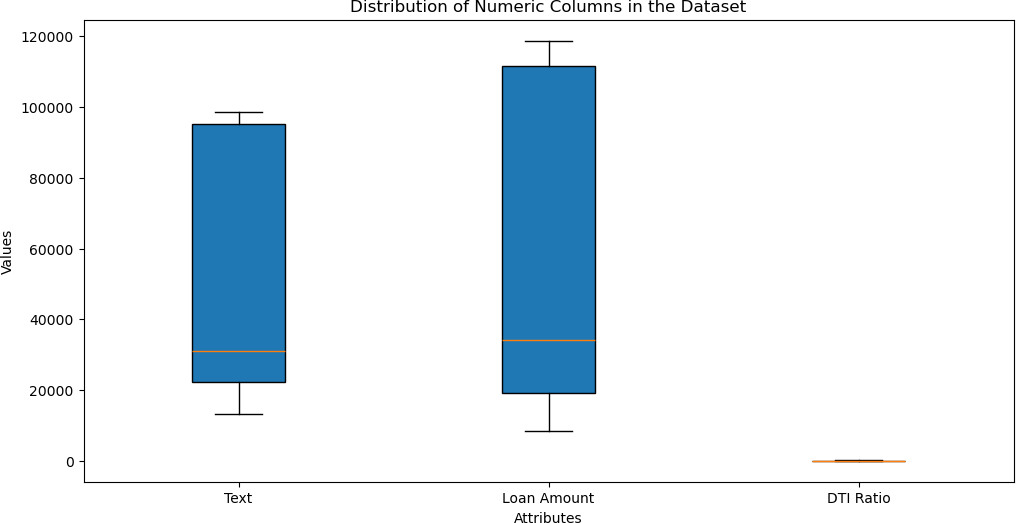
plt.boxplot(data, labels=['Text', 'Loan Amount', 'DTI Ratio'], patch\_artist=True)

*# Labels and title* plt.xlabel('Attributes') plt.ylabel('Values')

plt.title('Distribution of Numeric Columns in the Dataset')

*# Show plot*

plt.show()



*# Calculate Q1, Q3, and IQR for 'Text'* Q1\_text = df['Text'].quantile(0.25) Q3\_text = df['Text'].quantile(0.75) IQR\_text = Q3\_text - Q1\_text lower\_limit\_text = Q1\_text - 1.5 \* IQR\_text upper\_limit\_text = Q3\_text + 1.5 \* IQR\_text

*# Calculate Q1, Q3, and IQR for 'Loan\_Amount'* Q1\_loan = df['Loan\_Amount'].quantile(0.25) Q3\_loan = df['Loan\_Amount'].quantile(0.75)

IQR\_loan = Q3\_loan - Q1\_loan lower\_limit\_loan = Q1\_loan - 1.5 \* IQR\_loan upper\_limit\_loan = Q3\_loan + 1.5 \* IQR\_loan

*# Filter out outliers for both 'Text' and 'Loan\_Amount'*

df\_cleaned = df[

(df['Text'] >= lower\_limit\_text) & (df['Text'] <= upper\_limit\_text) & (df['Loan\_Amount'] >= lower\_limit\_loan) & (df['Loan\_Amount'] <=

upper\_limit\_loan)

]

*# Display the cleaned DataFrame*

print(df\_cleaned)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Text | Loan\_Amount | DTI\_Ratio | Employment\_Status | Approval |
| 0 | 13278.0 | 8314 | 79.26 | employed | Rejected |
| 1 | 98696.0 | 111604 | 22.14 | employed | Rejected |
| 2 | 22280.5 | 34118 | 45.44 | employed | Rejected |
| 3 | 95181.5 | 118757 | 10.22 | unemployed | Rejected |
| 4 | 30926.5 | 19210 | 44.13 | employed | Approved |
| df |  |  |  |  |  |
|  | Text | Loan\_Amount | DTI\_Ratio | Employment\_Status | Approval |
| 0 | 13278.0 | 8314 | 79.26 | employed | Rejected |
| 1 | 98696.0 | 111604 | 22.14 | employed | Rejected |
| 2 | 22280.5 | 34118 | 45.44 | employed | Rejected |
| 3 | 95181.5 | 118757 | 10.22 | unemployed | Rejected |
| 4 | 30926.5 | 19210 | 44.13 | employed | Approved |

*#DATA VIZUALIZATION*

**import** pandas **as** pd

**import** matplotlib.pyplot **as** plt

**import** seaborn **as** sns

data\_url = r"D:\Download\archive (29)\loan\_data.csv"

*# Read the CSV file*

df = pd.read\_csv(data\_url)

*# Basic exploration*

df.head()

Text Income Credit\_Score \

1. I need a loan to pay for an international vaca... 26556 581
2. I want to make home improvements like installi... 197392 389
3. I need a loan for home renovation, including a... 44561 523
4. I need funds to buy new furniture and applianc... 190363 729
5. I need a loan to start a small business. 61853 732

Loan\_Amount DTI\_Ratio Employment\_Status Approval

0 8314 79.26 employed Rejected

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 111604 | 22.14 | employed | Rejected |
| 2 | 34118 | 45.44 | employed | Rejected |
| 3 | 118757 | 10.22 | unemployed | Rejected |
| 4 | 19210 | 44.13 | employed | Approved |

df.info()

<class 'pandas.core.frame.DataFrame'> RangeIndex: 24000 entries, 0 to 23999 Data columns (total 7 columns):

# Column Non-Null Count Dtype

1. Text 24000 non-null object
2. Income 24000 non-null int64
3. Credit\_Score 24000 non-null int64
4. Loan\_Amount 24000 non-null int64
5. DTI\_Ratio 24000 non-null float64
6. Employment\_Status 24000 non-null object
7. Approval 24000 non-null object dtypes: float64(1), int64(3), object(3)

memory usage: 1.3+ MB df.describe()

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Income | Credit\_Score | Loan\_Amount | DTI\_Ratio |
| count | 24000.000000 | 24000.000000 | 24000.000000 | 24000.000000 |
| mean | 110377.552708 | 575.720333 | 44356.154833 | 34.719167 |
| std | 51729.677627 | 159.227621 | 34666.604785 | 32.322471 |
| min | 20001.000000 | 300.000000 | 1005.000000 | 2.530000 |
| 25% | 65635.750000 | 437.000000 | 16212.000000 | 14.507500 |
| 50% | 110464.000000 | 575.000000 | 35207.000000 | 24.860000 |
| 75% | 155187.000000 | 715.000000 | 65622.750000 | 41.840000 |
| max | 200000.000000 | 850.000000 | 158834.000000 | 246.330000 |

df.shape (24000, 7)

df.isnull().sum()

Text 0

Income 0

Credit\_Score 0

Loan\_Amount 0

DTI\_Ratio 0

Employment\_Status 0

Approval 0

dtype: int64

*# Handle Missing Values in the 'Age' column*

age\_mean = df['DTI\_Ratio'].mean() *# Calculate the mean of the 'Age' column*

df['DTI\_Ratio'] = df['DTI\_Ratio'].fillna(age\_mean) *# Fill missing values in*

*'Age' with the calculated mean*

*# Display the first few rows to check the result*

print(df.head())

Text Income Credit\_Score \

1. I need a loan to pay for an international vaca... 26556 581
2. I want to make home improvements like installi... 197392 389
3. I need a loan for home renovation, including a... 44561 523
4. I need funds to buy new furniture and applianc... 190363 729
5. I need a loan to start a small business. 61853 732

Loan\_Amount DTI\_Ratio Employment\_Status Approval

1. 8314 79.26 employed Rejected
2. 111604 22.14 employed Rejected
3. 34118 45.44 employed Rejected
4. 118757 10.22 unemployed Rejected
5. 19210 44.13 employed Approved

*# Replace missing values in 'Employment\_Status' with the mode* employment\_mode = df['Loan\_Amount'].mode()[0] *# Calculate the mode of 'Employment\_Status'*

df['Loan\_Amount'] = df['Loan\_Amount'].fillna(employment\_mode) *# Fill missing values with the mode*

*# Display the first few rows to check the result*

print(df.head())

Text Income Credit\_Score \

1. I need a loan to pay for an international vaca... 26556 581
2. I want to make home improvements like installi... 197392 389
3. I need a loan for home renovation, including a... 44561 523
4. I need funds to buy new furniture and applianc... 190363 729
5. I need a loan to start a small business. 61853 732

Loan\_Amount DTI\_Ratio Employment\_Status Approval

1. 8314 79.26 employed Rejected
2. 111604 22.14 employed Rejected
3. 34118 45.44 employed Rejected
4. 118757 10.22 unemployed Rejected
5. 19210 44.13 employed Approved employment\_mode

14409

*# Replace missing values in 'Blood Pressure' with the mode* df['Employment\_Status'] = df['Employment\_Status'].fillna(df['Employment\_Status'].mode()[0])

*# Display the first few rows to check the result*

print(df.head())

Text Income Credit\_Score \

1. I need a loan to pay for an international vaca... 26556 581
2. I want to make home improvements like installi... 197392 389
3. I need a loan for home renovation, including a... 44561 523
4. I need funds to buy new furniture and applianc... 190363 729
5. I need a loan to start a small business. 61853 732

Loan\_Amount DTI\_Ratio Employment\_Status Approval

1. 8314 79.26 employed Rejected
2. 111604 22.14 employed Rejected
3. 34118 45.44 employed Rejected
4. 118757 10.22 unemployed Rejected
5. 19210 44.13 employed Approved df.isnull().sum()

Text 0

Income 0

Credit\_Score 0

Loan\_Amount 0

DTI\_Ratio 0

Employment\_Status 0

Approval 0

dtype: int64

**import** matplotlib.pyplot **as** plt

*# Histogram for 'Text' column*

plt.hist(df['Text'], bins=20, color='skyblue', alpha=0.7) plt.title('Distribution of Text')

plt.xlabel('Text') plt.ylabel('Frequency') plt.show()

*# Histogram for 'Loan\_Amount' column*

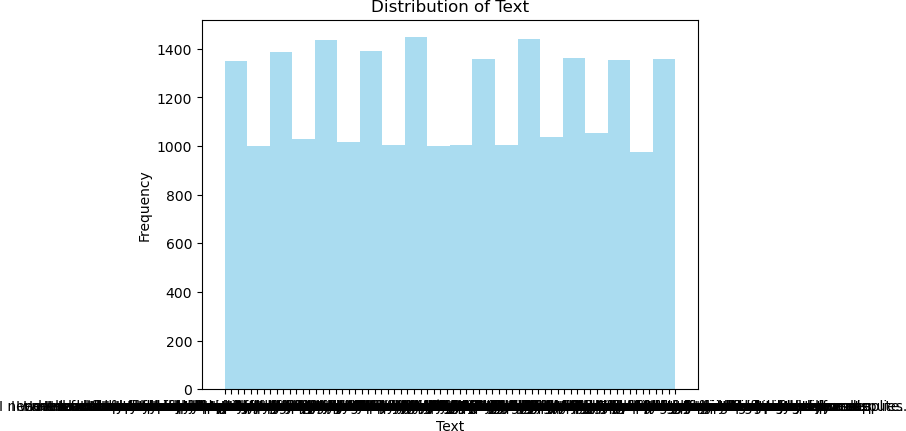
plt.hist(df['Loan\_Amount'], bins=20, color='lightgreen', alpha=0.7) plt.title('Distribution of Loan Amount')

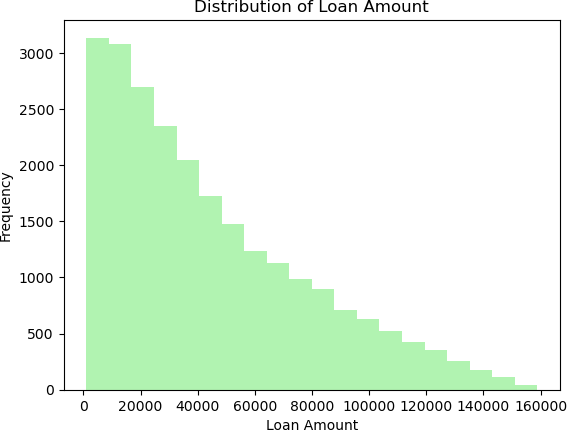
plt.xlabel('Loan Amount') plt.ylabel('Frequency') plt.show()

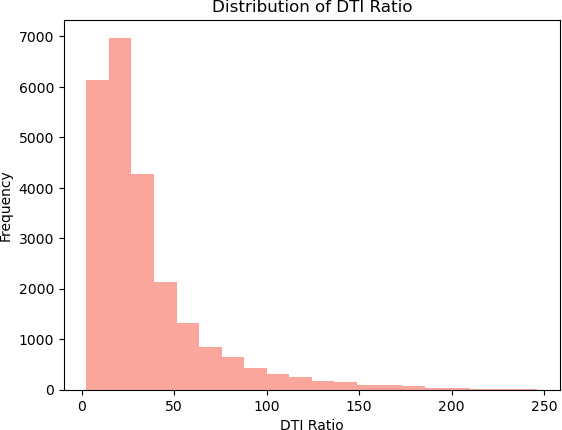
*# Histogram for 'DTI\_Ratio' column*

plt.hist(df['DTI\_Ratio'], bins=20, color='salmon', alpha=0.7) plt.title('Distribution of DTI Ratio')

plt.xlabel('DTI Ratio') plt.ylabel('Frequency') plt.show()







**import** seaborn **as** sns

**import** matplotlib.pyplot **as** plt

*# Select numerical columns from your dataset*

numerical\_cols = df.select\_dtypes(include=['float64', 'int64']).columns

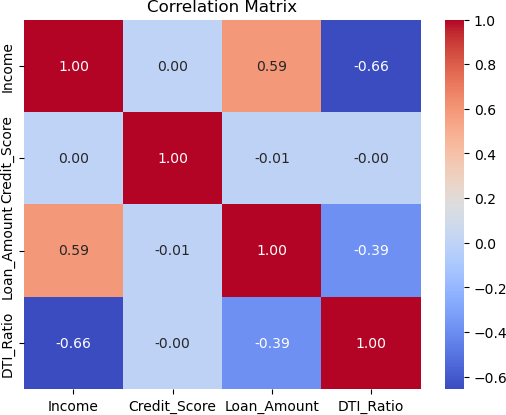
*# Calculate the correlation matrix*

corr\_matrix = df[numerical\_cols].corr()

*# Plot the heatmap*

sns.heatmap(corr\_matrix, annot=True, cmap='coolwarm', fmt='.2f') plt.title('Correlation Matrix')

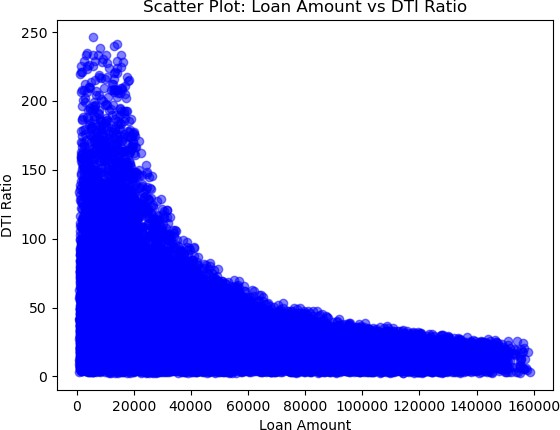
plt.show()



**import** matplotlib.pyplot **as** plt

*# Scatter plot for 'Loan\_Amount' vs 'DTI\_Ratio'* plt.scatter(df['Loan\_Amount'], df['DTI\_Ratio'], color='blue', alpha=0.5) plt.title('Scatter Plot: Loan Amount vs DTI Ratio')

plt.xlabel('Loan Amount') plt.ylabel('DTI Ratio') plt.show()



*# Set 'Patient ID' as the index of the DataFrame*

df.set\_index('DTI\_Ratio', inplace=True)

*# Display the first few rows to check the result*

print(df.head())

Text Income \

DTI\_Ratio

79.26 I need a loan to pay for an international vaca... 26556

22.14 I want to make home improvements like installi... 197392

45.44 I need a loan for home renovation, including a... 44561

10.22 I need funds to buy new furniture and applianc... 190363

44.13 I need a loan to start a small business. 61853

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Credit\_Score | Loan\_Amount | Employment\_Status | Approval |
| DTI\_Ratio |  |  |  |  |
| 79.26 | 581 | 8314 | employed | Rejected |
| 22.14 | 389 | 111604 | employed | Rejected |
| 45.44 | 523 | 34118 | employed | Rejected |
| 10.22 | 729 | 118757 | unemployed | Rejected |
| 44.13 | 732 | 19210 | employed | Approved |

*#Different Type of Chart*

**import** matplotlib.pyplot **as** plt

*# Count number of records in each Employment Status category*

employment\_counts = df['Employment\_Status'].value\_counts()

*# Create bar chart*

plt.figure(figsize=(8, 5))

bars = plt.bar(employment\_counts.index, employment\_counts.values, color=['skyblue', 'salmon', 'lightgreen'])

*# Add value labels above bars*

**for** bar **in** bars:

height = bar.get\_height()

plt.text(bar.get\_x() + bar.get\_width()/2, height + 0.1, str(height), ha='center', fontsize=12)

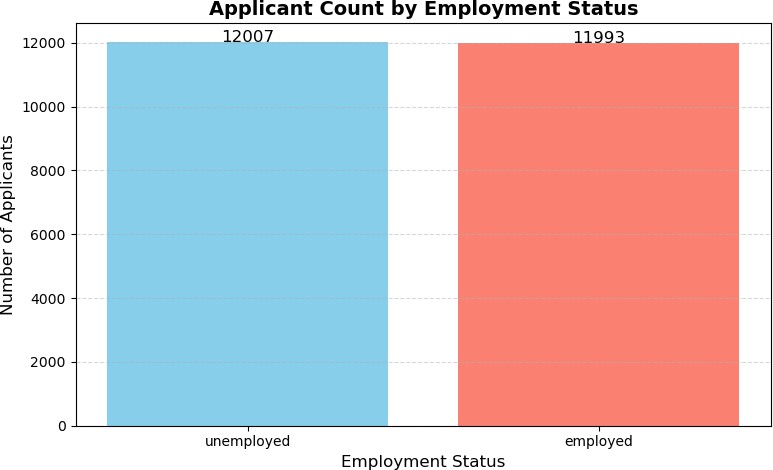
*# Labels and title*

plt.xlabel('Employment Status', fontsize=12) plt.ylabel('Number of Applicants', fontsize=12) plt.title('Applicant Count by Employment Status', fontsize=14, fontweight='bold')

*# Show plot*

plt.grid(axis='y', linestyle='--', alpha=0.5) plt.tight\_layout()

plt.show()



**import** matplotlib.pyplot **as** plt

*# Count the number of applicants in each Employment Status category*

employment\_counts = df['Employment\_Status'].value\_counts()

*# Horizontal bar chart*

plt.figure(figsize=(8, 5))

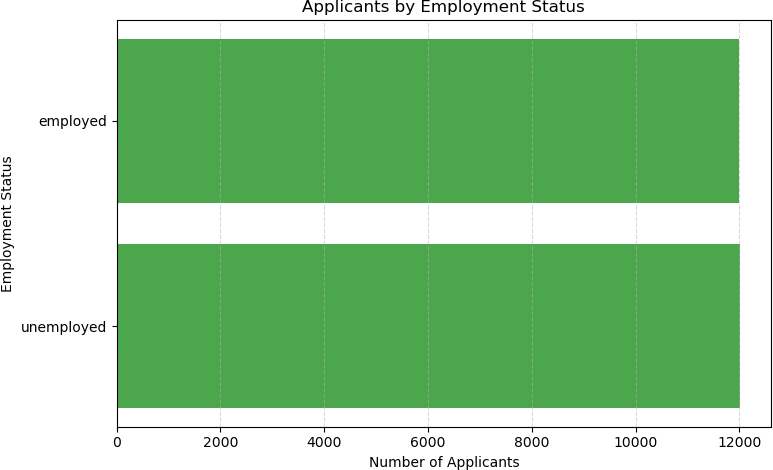
plt.barh(employment\_counts.index, employment\_counts.values, color='green', alpha=0.7)

*# Labels and title* plt.xlabel("Number of Applicants") plt.ylabel("Employment Status")

plt.title("Applicants by Employment Status")

*# Grid for better readability* plt.grid(axis='x', linestyle='--', alpha=0.5) plt.tight\_layout()

plt.show()



**import** matplotlib.pyplot **as** plt

**import** pandas **as** pd

*# Clean column names by stripping leading/trailing spaces*

df.columns = df.columns.str.strip()

*# Print available columns*

print("Available columns:", df.columns.tolist())

*# Ensure the required columns exist*

**if** 'Income' **in** df.columns **and** 'Loan\_Amount' **in** df.columns:

*# Plotting Loan Amount over Income*

plt.figure(figsize=(8, 5))

plt.plot(df['Income'], df['Loan\_Amount'], color='blue', marker='o', linestyle='-')

plt.xlabel('Income') plt.ylabel('Loan Amount')

plt.title('Loan Amount Trend Over Income') plt.grid(True, linestyle='--', alpha=0.5) plt.tight\_layout()

plt.show()

# else:

print("Columns 'Income' or 'Loan Amount' are missing from the DataFrame.")

Available columns: ['Location', 'BMI']

Columns 'Income' or 'Loan Amount' are missing from the DataFrame.

**import** pandas **as** pd

**import** matplotlib.pyplot **as** plt

*# Assuming 'df' is your existing DataFrame # Get the count of patients by Location*

locations = df['Location'].value\_counts().index *# Locations*

values = df['Location'].value\_counts().values *# Count of patients per location*

*# Create pie chart*

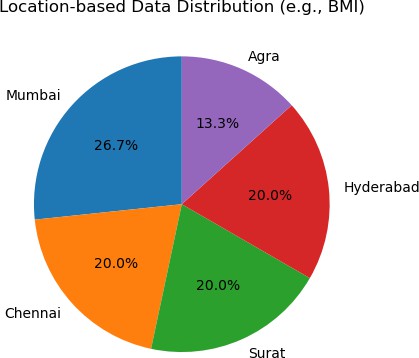
plt.pie(values, labels=locations, autopct='%1.1f%%', startangle=90)

*# Title*

plt.title('Location-based Data Distribution (e.g., BMI)')

*# Show plot*

plt.show()



**import** matplotlib.pyplot **as** plt

*# Clean column names by stripping leading/trailing spaces*

df.columns = df.columns.str.strip()

*# Print available columns to ensure 'Income' and 'Loan\_Amount' are available*

print("Available columns:", df.columns.tolist())

*# Check if the necessary columns exist*

**if** 'Income' **in** df.columns **and** 'Loan\_Amount' **in** df.columns:

*# Extract the columns for plotting*

income = df['Income'] loan\_amount = df['Loan\_Amount']

*# Create scatter plot*

plt.scatter(income, loan\_amount, color='blue')

*# Labels and title* plt.xlabel('Income') plt.ylabel('Loan Amount') plt.title('Income vs Loan Amount')

*# Show plot*

plt.show()

# else:

print("Columns 'Income' or 'Loan\_Amount' are missing from the DataFrame.")

Available columns: ['Location', 'BMI']

Columns 'Income' or 'Loan\_Amount' are missing from the DataFrame.

**import** matplotlib.pyplot **as** plt

*# Assuming the "Blood Pressure" data is in your DataFrame (df)*

blood\_pressure = df['Blood Pressure'].dropna() *# Drop any NaN values*

*# Create a histogram*

plt.hist(blood\_pressure, bins=len(blood\_pressure.unique()), color='purple', edgecolor='black')

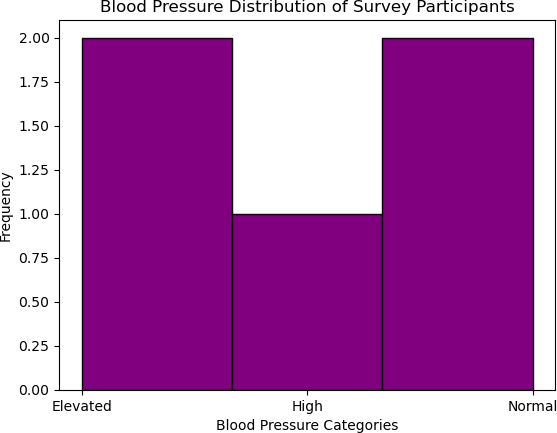
*# Labels and title*

plt.xlabel('Blood Pressure Categories') plt.ylabel('Frequency')

plt.title('Blood Pressure Distribution of Survey Participants')

*# Show plot*

plt.show()



*#UNIVARIATE ANALYSIS*

**import** pandas **as** pd

**import** matplotlib.pyplot **as** plt

**import** seaborn **as** sns

data=pd.read\_csv(r"D:\Download\archive (29)\loan\_data.csv") print(data.head())

Text Income Credit\_Score \

1. I need a loan to pay for an international vaca... 26556 581
2. I want to make home improvements like installi... 197392 389
3. I need a loan for home renovation, including a... 44561 523
4. I need funds to buy new furniture and applianc... 190363 729
5. I need a loan to start a small business. 61853 732

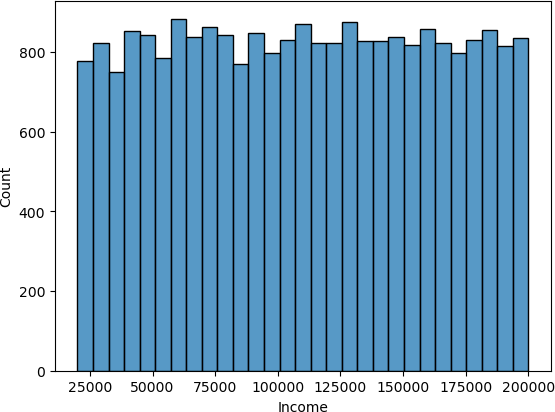
Loan\_Amount DTI\_Ratio Employment\_Status Approval

1. 8314 79.26 employed Rejected
2. 111604 22.14 employed Rejected
3. 34118 45.44 employed Rejected
4. 118757 10.22 unemployed Rejected
5. 19210 44.13 employed Approved data.columns

Index(['Text', 'Income', 'Credit\_Score', 'Loan\_Amount', 'DTI\_Ratio', 'Employment\_Status', 'Approval'],

dtype='object') sns.histplot(data['Income'])

<Axes: xlabel='Income', ylabel='Count'>



*#UNIVARIATE*

**import** pandas **as** pd

**import** seaborn **as** sns

data = pd.read\_csv(r"D:\Download\archive (29)\loan\_data.csv") print(data.head())

Text Income Credit\_Score \

1. I need a loan to pay for an international vaca... 26556 581
2. I want to make home improvements like installi... 197392 389
3. I need a loan for home renovation, including a... 44561 523
4. I need funds to buy new furniture and applianc... 190363 729
5. I need a loan to start a small business. 61853 732

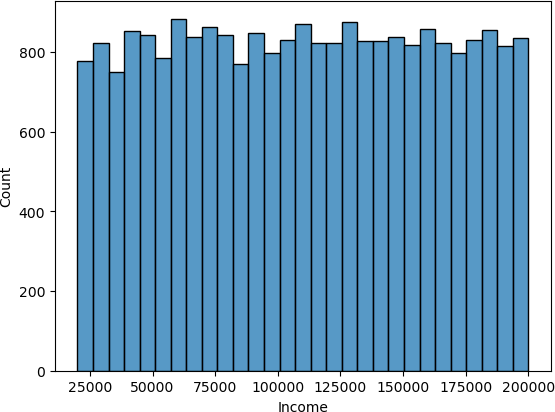
Loan\_Amount DTI\_Ratio Employment\_Status Approval

1. 8314 79.26 employed Rejected
2. 111604 22.14 employed Rejected
3. 34118 45.44 employed Rejected
4. 118757 10.22 unemployed Rejected
5. 19210 44.13 employed Approved

*#HISTOGRAM*

sns.histplot(data['Income'])

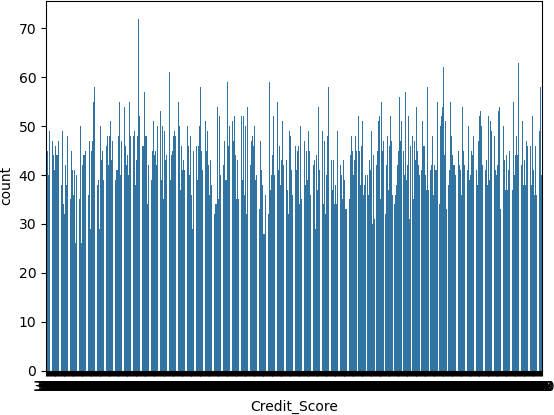
<Axes: xlabel='Income', ylabel='Count'>



*#BAR CHART*

sns.countplot(x=data['Credit\_Score'])

<Axes: xlabel='Credit\_Score', ylabel='count'>



*#PIE CHART*

**import** pandas **as** pd

**import** matplotlib.pyplot **as** plt

*# Step 1: Load the dataset (correctly based on file type)*

data = pd.read\_csv(r"D:\Download\archive (29)\loan\_data.csv") *# Use read\_csv for .csv files*

*# Step 2: Optional - Map Approval column if still in text*

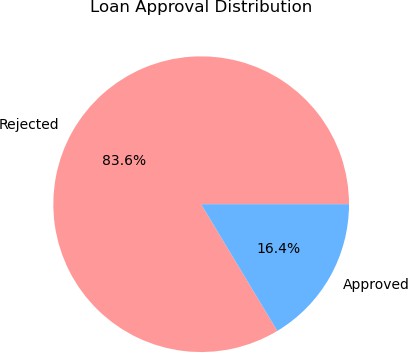
data['Approval'] = data['Approval'].map({'Rejected': 0, 'Approved': 1})

*# Step 3: Create pie chart*

x = data['Approval'].value\_counts()

plt.pie(x.values, labels=['Rejected', 'Approved'], autopct='%1.1f%%', colors=['#ff9999','#66b3ff'])

plt.title("Loan Approval Distribution") plt.show()



*#BIVARIATE ANALYSIS*

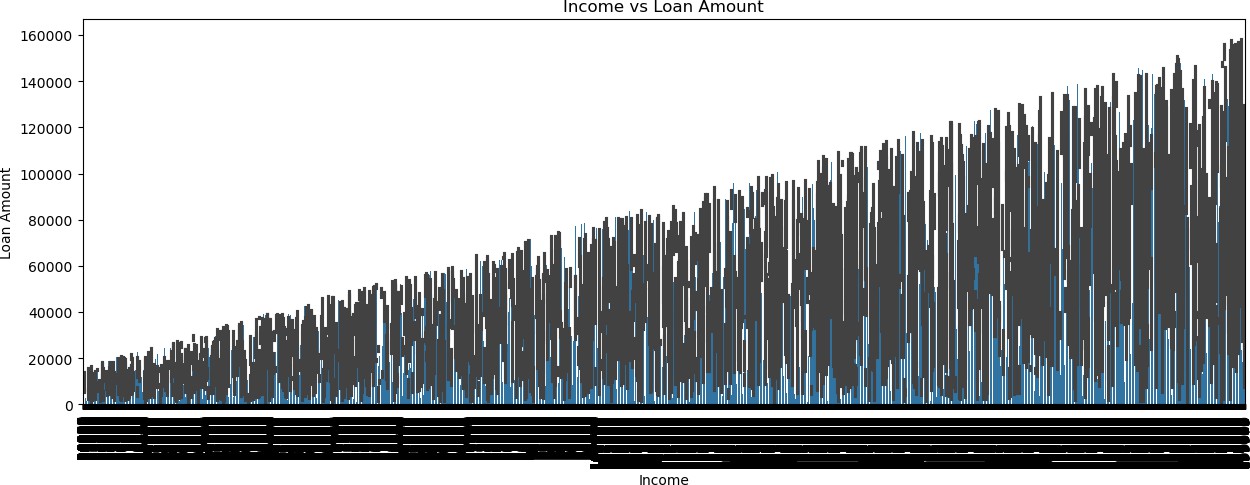
*#Categorical v/s Numerical* **import** matplotlib.pyplot **as** plt **import** seaborn **as** sns

plt.figure(figsize=(15, 5))

sns.barplot(x=data['Income'], y=data['Loan\_Amount']) *# Also fixed column name spacing*

plt.xticks(rotation=90) *# Use straight quotes or just a number*

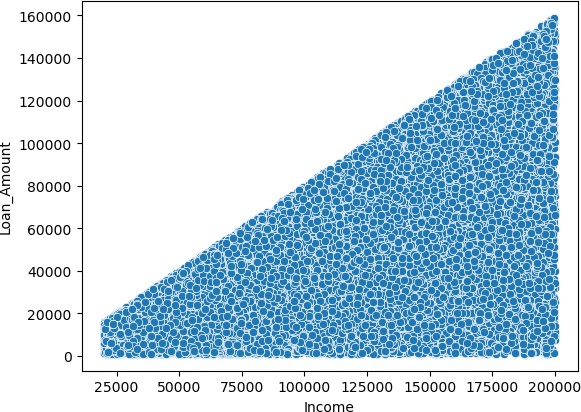
plt.title("Income vs Loan Amount") plt.xlabel("Income") plt.ylabel("Loan Amount") plt.show()



*#Numerical v/s Numerical*

sns.scatterplot(x=data['Income'],y=data['Loan\_Amount'])

<Axes: xlabel='Income', ylabel='Loan\_Amount'>



*#MULTIVARIATE ANALYSIS*

*#Principal Componenet Analysis*

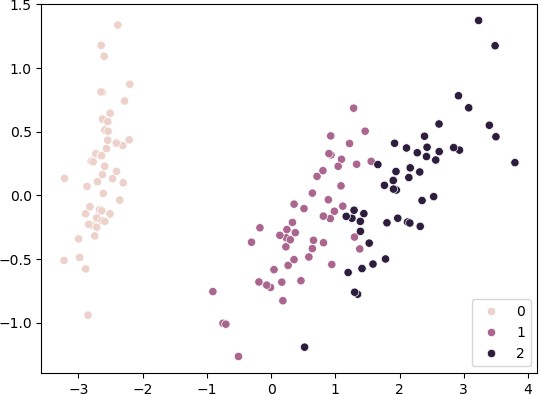
**from** sklearn **import** datasets, decomposition iris = datasets.load\_iris()

X = iris.data y = iris.target

pca = decomposition.PCA(n\_components=2)

X = pca.fit\_transform(X) sns.scatterplot(x=X[:, 0], y=X[:, 1], hue=y)

<Axes: >



*#Heat Map*

**import** seaborn **as** sns

**import** matplotlib.pyplot **as** plt

*# Drop non-numeric columns like 'Text'*

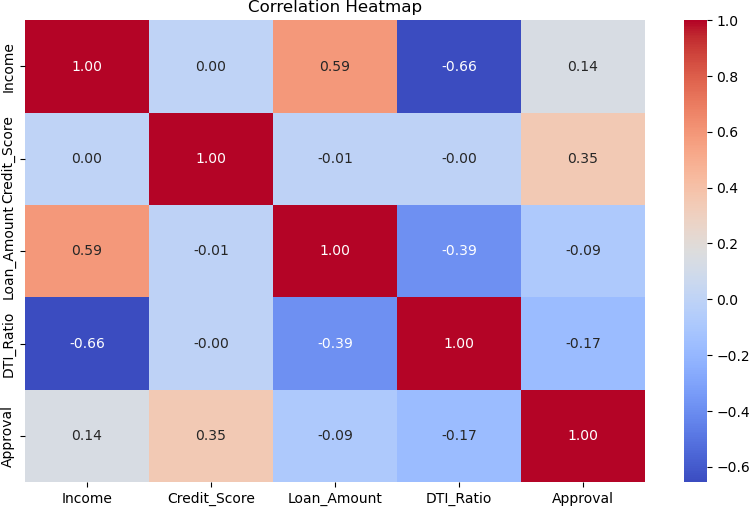
numeric\_data = data.select\_dtypes(include='number') *# Keeps only numeric columns*

*# Now plot the heatmap*

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

sns.heatmap(numeric\_data.corr(), annot=True, cmap='coolwarm', fmt=".2f") plt.title("Correlation Heatmap")

plt.show()



*#LOGISTIC REGRESSION*

**import** pandas **as** pd

**from** sklearn.model\_selection **import** train\_test\_split **from** sklearn.linear\_model **import** LogisticRegression **from** sklearn.preprocessing **import** StandardScaler

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

dataset = pd.read\_csv(r"D:\Download\archive (29)\loan\_data.csv") dataset

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | | Text | Income | \ |
| 0 | I | need | a loan to pay for an international vaca... | 26556 |  |
| 1 | I | want | to make home improvements like installi... | 197392 |  |
| 2 | I | need | a loan for home renovation, including a... | 44561 |  |
| 3 | I | need | funds to buy new furniture and applianc... | 190363 |  |
| 4 |  |  | I need a loan to start a small business. | 61853 |  |
| ... |  |  | ... | ... |  |
| 23995 | I | need | funds to pay for my daughter's college ... | 195242 |  |
| 23996 | I | need | financial assistance to launch my own Y... | 150246 |  |

23997 I need money to open a small bookstore and café. 64571 23998 I want to buy a car for my rideshare business ... 115825 23999 I need financial help to cover maternity and d... 180440

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Credit\_Score | Loan\_Amount | DTI\_Ratio Employment\_Status | Approval |
| 0 | 581 | 8314 | 79.26 employed | Rejected |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 |  | 389 | 111604 | 22.14 | employed | Rejected |
| 2 |  | 523 | 34118 | 45.44 | employed | Rejected |
| 3 |  | 729 | 118757 | 10.22 | unemployed | Rejected |
| 4 |  | 732 | 19210 | 44.13 | employed | Approved |
| ... |  | ... | ... | ... | ... | ... |
| 23995 |  | 817 | 16403 | 24.32 | employed | Approved |
| 23996 |  | 729 | 101572 | 9.97 | employed | Rejected |
| 23997 |  | 650 | 30533 | 57.35 | employed | Rejected |
| 23998 |  | 418 | 89837 | 10.37 | unemployed | Rejected |
| 23999 |  | 769 | 15259 | 15.06 | employed | Approved |
| [24000 | rows x 7 | columns] |  |  |  |  |

dataset.info()

<class 'pandas.core.frame.DataFrame'> RangeIndex: 24000 entries, 0 to 23999 Data columns (total 7 columns):

# Column Non-Null Count Dtype

1. Text 24000 non-null object
2. Income 24000 non-null int64
3. Credit\_Score 24000 non-null int64
4. Loan\_Amount 24000 non-null int64
5. DTI\_Ratio 24000 non-null float64
6. Employment\_Status 24000 non-null object
7. Approval 24000 non-null object dtypes: float64(1), int64(3), object(3)

memory usage: 1.3+ MB dataset.describe()

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Income | Credit\_Score | Loan\_Amount | DTI\_Ratio |
| count | 24000.000000 | 24000.000000 | 24000.000000 | 24000.000000 |
| mean | 110377.552708 | 575.720333 | 44356.154833 | 34.719167 |
| std | 51729.677627 | 159.227621 | 34666.604785 | 32.322471 |
| min | 20001.000000 | 300.000000 | 1005.000000 | 2.530000 |
| 25% | 65635.750000 | 437.000000 | 16212.000000 | 14.507500 |
| 50% | 110464.000000 | 575.000000 | 35207.000000 | 24.860000 |
| 75% | 155187.000000 | 715.000000 | 65622.750000 | 41.840000 |
| max | 200000.000000 | 850.000000 | 158834.000000 | 246.330000 |

dataset.isnull().sum() Text 0

Income 0

Credit\_Score 0

Loan\_Amount 0

DTI\_Ratio 0

Employment\_Status 0

Approval 0

dtype: int64

columns=['Income','Credit\_Score','Loan\_Amount'] df\_corr=dataset[columns].corr()

|  |  |  |  |
| --- | --- | --- | --- |
| df\_corr |  | | |
|  | Income | Credit\_Score | Loan\_Amount |
| Income | 1.000000 | 0.001201 | 0.590209 |
| Credit\_Score | 0.001201 | 1.000000 | -0.006542 |
| Loan\_Amount | 0.590209 | -0.006542 | 1.000000 |

**import** seaborn **as** sns sns.heatmap(df\_corr,annot=True)

<Axes: >



X = dataset.iloc[:, [2, 3]].values y = dataset.iloc[:, 4].values

**from** sklearn.preprocessing **import** StandardScaler sc = StandardScaler()

X = sc.fit\_transform(X)

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.25, random\_state = 0)

**import** pandas **as** pd

**from** sklearn.model\_selection **import** train\_test\_split **from** sklearn.linear\_model **import** LogisticRegression **from** sklearn.preprocessing **import** StandardScaler

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

*# Step 1: Load your dataset*

df = pd.read\_csv(r"D:\Download\archive (29)\loan\_data.csv") *# Replace with your actual file path or use df = your\_dataframe if it's already loaded*

*# Step 2: Convert categorical columns to numeric* df['Employment\_Status'] = df['Employment\_Status'].map({'employed': 1, 'unemployed': 0})

df['Approval'] = df['Approval'].map({'Rejected': 0, 'Approved': 1}) df['Approval'] = df['Approval'].astype(int) *# Ensure it's an integer*

*# Step 3: Define features and target*

X = df.drop(['Text', 'Approval'], axis=1) y = df['Approval']

*# Step 4: Train-test split*

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

*# Step 5: Feature scaling*

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train) X\_test = scaler.transform(X\_test)

*# Step 6: Fit logistic regression model* model = LogisticRegression() model.fit(X\_train, y\_train)

*# Step 7: Predict and evaluate*

y\_pred = model.predict(X\_test)

print("Accuracy:", accuracy\_score(y\_test, y\_pred)) print("\nConfusion Matrix:\n", confusion\_matrix(y\_test, y\_pred))

print("\nClassification Report:\n", classification\_report(y\_test, y\_pred)) Accuracy: 0.928125

Confusion Matrix: [[3870 177]

[ 168 585]]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Classification | Report:  precision | recall | f1-score | support |
| 0 | 0.96 | 0.96 | 0.96 | 4047 |
| 1 | 0.77 | 0.78 | 0.77 | 753 |
| accuracy |  |  | 0.93 | 4800 |
| macro avg | 0.86 | 0.87 | 0.86 | 4800 |
| weighted avg | 0.93 | 0.93 | 0.93 | 4800 |

y\_pred = model.predict(X\_test) y\_pred

array([0, 0, 0, ..., 0, 0, 0])

y\_pred1=model.predict(X)

C:\Users\ASUS\anaconda3\Lib\site-packages\sklearn\base.py:486: UserWarning: X has feature names, but LogisticRegression was fitted without feature names

warnings.warn( dataset['Prediction']=y\_pred1 dataset

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | | Text | Income | \ |
| 0 | I | need | a loan to pay for an international vaca... | 26556 |  |
| 1 | I | want | to make home improvements like installi... | 197392 |  |
| 2 | I | need | a loan for home renovation, including a... | 44561 |  |
| 3 | I | need | funds to buy new furniture and applianc... | 190363 |  |
| 4 |  |  | I need a loan to start a small business. | 61853 |  |
| ... |  |  | ... | ... |  |
| 23995 | I | need | funds to pay for my daughter's college ... | 195242 |  |
| 23996 | I | need | financial assistance to launch my own Y... | 150246 |  |

23997 I need money to open a small bookstore and café. 64571 23998 I want to buy a car for my rideshare business ... 115825 23999 I need financial help to cover maternity and d... 180440

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Credit\_Score | Loan\_Amount | DTI\_Ratio | Employment\_Status | Approval | \ |
| 0 | 581 | 8314 | 79.26 | employed | Rejected |  |
| 1 | 389 | 111604 | 22.14 | employed | Rejected |  |
| 2 | 523 | 34118 | 45.44 | employed | Rejected |  |
| 3 | 729 | 118757 | 10.22 | unemployed | Rejected |  |
| 4 | 732 | 19210 | 44.13 | employed | Approved |  |
| ... | ... | ... | ... | ... | ... |  |
| 23995 | 817 | 16403 | 24.32 | employed | Approved |  |
| 23996 | 729 | 101572 | 9.97 | employed | Rejected |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 23997 | 650 | 30533 | 57.35 | employed | Rejected |
| 23998 | 418 | 89837 | 10.37 | unemployed | Rejected |
| 23999 | 769 | 15259 | 15.06 | employed | Approved |
| 0 | Prediction  1 | | | | |
| 1 | 0 | | | | |
| 2 | 0 | | | | |
| 3 | 0 | | | | |
| 4 | 1 | | | | |
| ... | ... | | | | |
| 23995 | 1 | | | | |
| 23996 | 0 | | | | |
| 23997 | 1 | | | | |
| 23998 | 0 | | | | |
| 23999 | 1 | | | | |

[24000 rows x 8 columns]

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

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

[[3870 177]

[ 168 585]]

accuracy = accuracy\_score(y\_test, y\_pred) print("Accuracy:", accuracy)

Accuracy: 0.928125

print("Classification Report:\n", classification\_report(y\_test, y\_pred))

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Classification | Report:  precision | recall | f1-score | support |
| 0 | 0.96 | 0.96 | 0.96 | 4047 |
| 1 | 0.77 | 0.78 | 0.77 | 753 |
| accuracy |  |  | 0.93 | 4800 |
| macro avg | 0.86 | 0.87 | 0.86 | 4800 |
| weighted avg | 0.93 | 0.93 | 0.93 | 4800 |

*#SIMPLE LINEAR REGRESSION*

**import** numpy **as** np

**import** matplotlib.pyplot **as** plt

**import** seaborn **as** sns

**import** pandas **as** pd

**import** sklearn **as** sk

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

**from** sklearn.linear\_model **import** LinearRegression

**from** sklearn.metrics **import** mean\_squared\_error, mean\_absolute\_error, r2\_score

dataset=pd.read\_csv(r"D:\Download\archive (29)\loan\_data.csv") dataset

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | | Text | Income | \ |
| 0 | I | need | a loan to pay for an international vaca... | 26556 |  |
| 1 | I | want | to make home improvements like installi... | 197392 |  |
| 2 | I | need | a loan for home renovation, including a... | 44561 |  |
| 3 | I | need | funds to buy new furniture and applianc... | 190363 |  |
| 4 |  |  | I need a loan to start a small business. | 61853 |  |
| ... |  |  | ... | ... |  |
| 23995 | I | need | funds to pay for my daughter's college ... | 195242 |  |
| 23996 | I | need | financial assistance to launch my own Y... | 150246 |  |

23997 I need money to open a small bookstore and café. 64571 23998 I want to buy a car for my rideshare business ... 115825 23999 I need financial help to cover maternity and d... 180440

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Credit\_Score | Loan\_Amount | DTI\_Ratio | Employment\_Status | Approval |
| 0 | 581 | 8314 | 79.26 | employed | Rejected |
| 1 | 389 | 111604 | 22.14 | employed | Rejected |
| 2 | 523 | 34118 | 45.44 | employed | Rejected |
| 3 | 729 | 118757 | 10.22 | unemployed | Rejected |
| 4 | 732 | 19210 | 44.13 | employed | Approved |
| ... | ... | ... | ... | ... | ... |
| 23995 | 817 | 16403 | 24.32 | employed | Approved |
| 23996 | 729 | 101572 | 9.97 | employed | Rejected |
| 23997 | 650 | 30533 | 57.35 | employed | Rejected |
| 23998 | 418 | 89837 | 10.37 | unemployed | Rejected |
| 23999 | 769 | 15259 | 15.06 | employed | Approved |

[24000 rows x 7 columns] dataset.describe()

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Income | Credit\_Score | Loan\_Amount | DTI\_Ratio |
| count | 24000.000000 | 24000.000000 | 24000.000000 | 24000.000000 |
| mean | 110377.552708 | 575.720333 | 44356.154833 | 34.719167 |
| std | 51729.677627 | 159.227621 | 34666.604785 | 32.322471 |
| min | 20001.000000 | 300.000000 | 1005.000000 | 2.530000 |
| 25% | 65635.750000 | 437.000000 | 16212.000000 | 14.507500 |
| 50% | 110464.000000 | 575.000000 | 35207.000000 | 24.860000 |
| 75% | 155187.000000 | 715.000000 | 65622.750000 | 41.840000 |
| max | 200000.000000 | 850.000000 | 158834.000000 | 246.330000 |

dataset.shape (24000, 7)

dataset.shape[0] 24000

dataset.isnull().sum()

Text 0

Income 0

Credit\_Score 0

Loan\_Amount 0

DTI\_Ratio 0

Employment\_Status 0

Approval 0

dtype: int64

x = dataset['Income'].values.reshape(-1, 1)

y = dataset['Credit\_Score'].values.reshape(-1, 1)

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.2, random\_state=0)

x\_train.shape (19200, 1)

y\_train.shape (19200, 1)

model=LinearRegression() model.fit(x\_train, y\_train)

LinearRegression() y\_pred=model.predict(x\_test) y\_pred

array([[575.4793903 ],

[575.53178935],

[575.42353854],

..., [575.42101072],

[575.48858662],

[575.53831316]])

mae=mean\_absolute\_error(y\_test,y\_pred) mae

138.60431587790353

mse=mean\_squared\_error(y\_test,y\_pred) mse

25543.140501405516

rmse=np.sqrt(mse) rmse

159.8222152937617

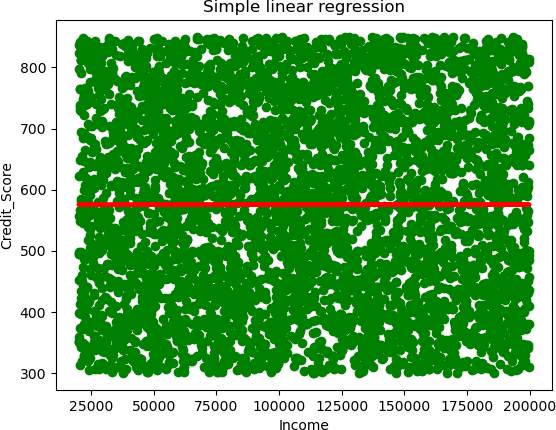
r2=r2\_score(y\_test,y\_pred) r2

-3.586360194995919e-05

plt.scatter(x\_test,y\_test, color='green') plt.plot(x\_test,y\_pred,color='red',linewidth=3) plt.xlabel('Income')

plt.ylabel('Credit\_Score') plt.title('Simple linear regression')

Text(0.5, 1.0, 'Simple linear regression')



**Conclusion / Insights**

In this project, we worked on a loan dataset to understand what factors affect the approval of loans. We studied different columns like Income, Credit Score, Loan Amount, DTI Ratio (Debt-to-Income Ratio), and Employment Status. These are the key things that banks or financial companies look at before giving someone a loan.

From our analysis, we found that people with high income and good credit scores are more likely to get their loans approved. A high income shows that a person can repay the loan easily, and a good credit score means that the person has paid back loans on time in the past.

We also saw that the DTI Ratio is important. If someone has a high DTI ratio (which means they already have a lot of debt compared to their income), their loan is more likely to be rejected. On the other hand, people with a low DTI ratio have a better chance of getting approval.

Employment Status also plays a big role. People who are working full-time usually have a steady income, so they are more likely to be approved. Self-employed or unemployed people have a lower chance.

We used different types of analysis like univariate (one column at a time), bivariate (two columns at a time), and multivariate (multiple columns together). Visuals like bar charts and scatter plots helped us understand the patterns clearly.

In conclusion, this project helped us learn how data can be used to make better decisions in the real world. Loan companies can use these insights to build smarter systems and reduce risk while helping people who truly need loans.

Github Link:-