Privelege Magoba ISOM 835 Individual Term Project Final Report

Introduction & Dataset Description

This project applies predictive analytics and machine learning techniques to a Loan Prediction dataset, simulating real-world lending decisions faced by financial institutions. The objective is to build accurate and interpretable models that predict whether a loan application should be approved based on various applicant attributes such as income, credit history, employment type, education level, and demographic information. Through exploratory data analysis, data preprocessing, and model development, the project aims to uncover key factors influencing loan approvals, support decision-making for stakeholders such as credit risk teams and lending officers and reflect on the ethical implications of using automated decision systems in finance.

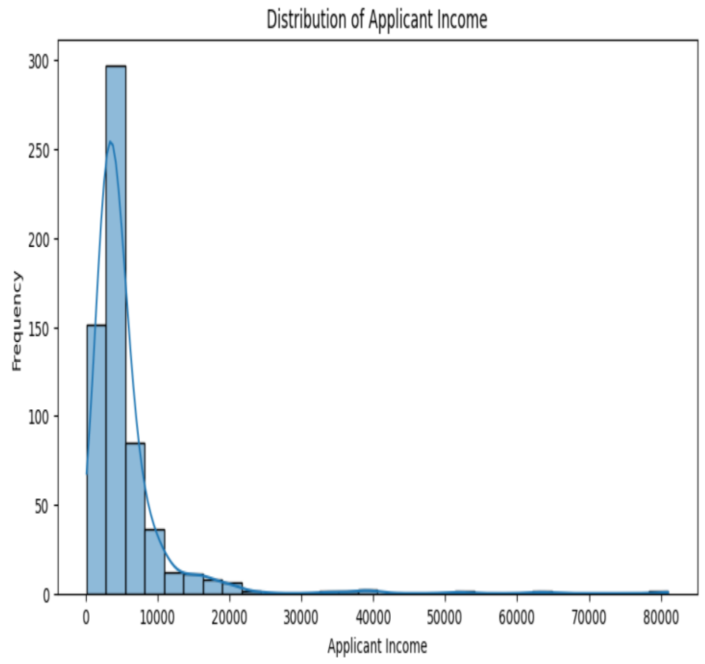
Rationale for selecting the Loan Prediction Problem Dataset

https://www.kaggle.com/datasets/altruistdelhite04/loan-prediction-problem-dataset/data?select=train_u6lujuX_CVtuZ9i.csv

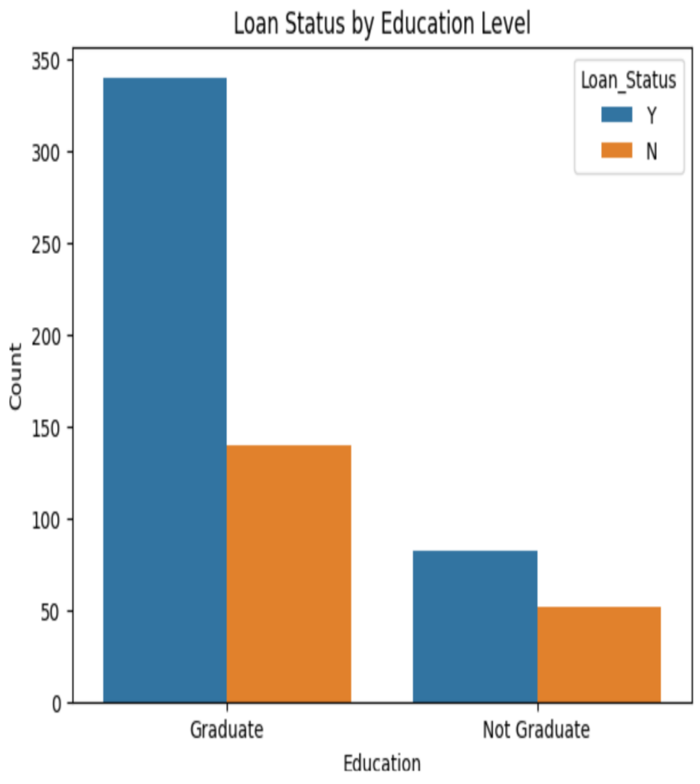
I selected this dataset because it offers a rich and practical context for applying predictive analytics and machine learning to real-world financial decisions. The dataset simulates a typical scenario faced by banks and lending institutions in determining whether to approve a loan application based on a range of applicant attributes such as income, employment status, credit history, and education.

Furthermore, the dataset includes a mix of categorical and numerical variables, as well as class imbalance in the target variable, making it ideal for practicing key data science tasks such as data cleaning, feature engineering, model building, and performance evaluation. It allows for the exploration of fairness in lending, the impact of socio-demographic factors on loan approvals, and the development of explainable models all of which are essential considerations in both business analytics and ethical AI. The practical relevance, diversity of variables, and opportunity for stakeholder-driven insights made this dataset a compelling choice for this project.

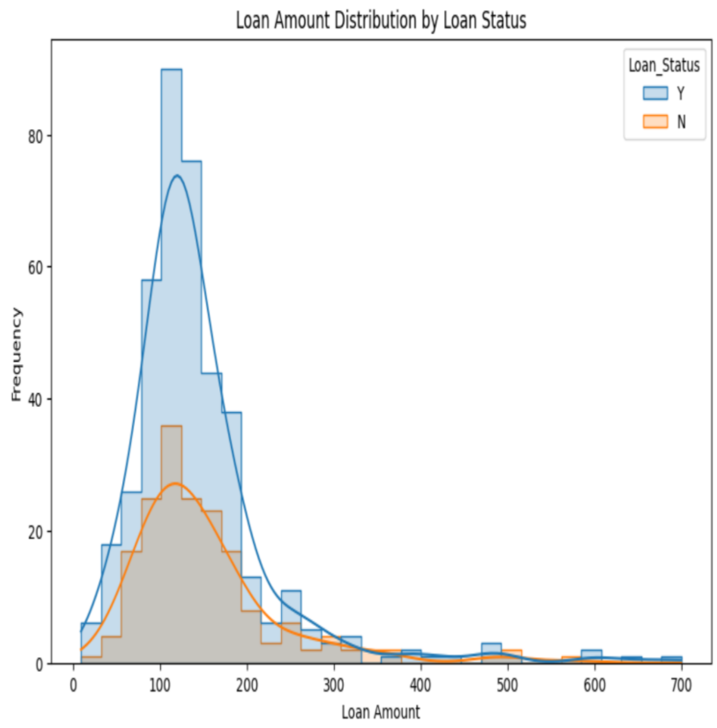
EDA



Income is right skewed and most applicant incomes fall below 10,000. A long tail indicates several outliers with significantly higher income. Knowing these outliers will help me with data cleaning.

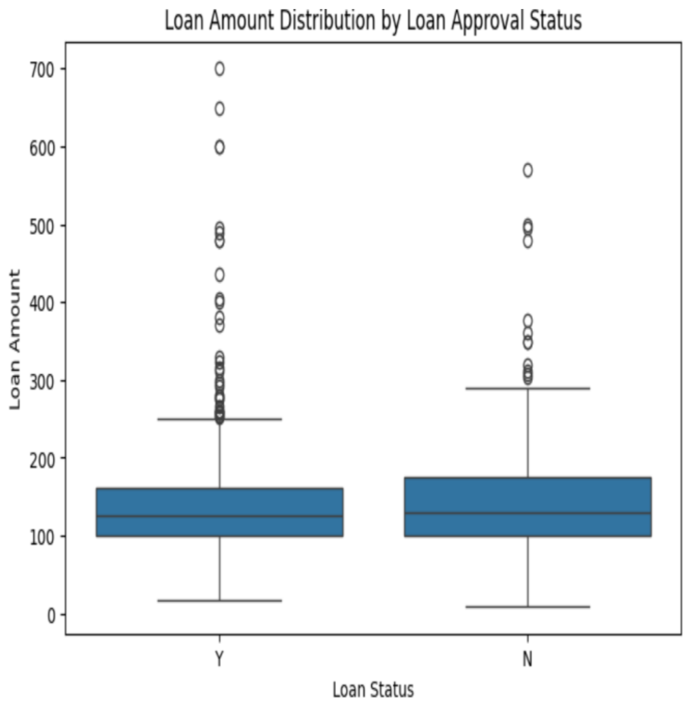


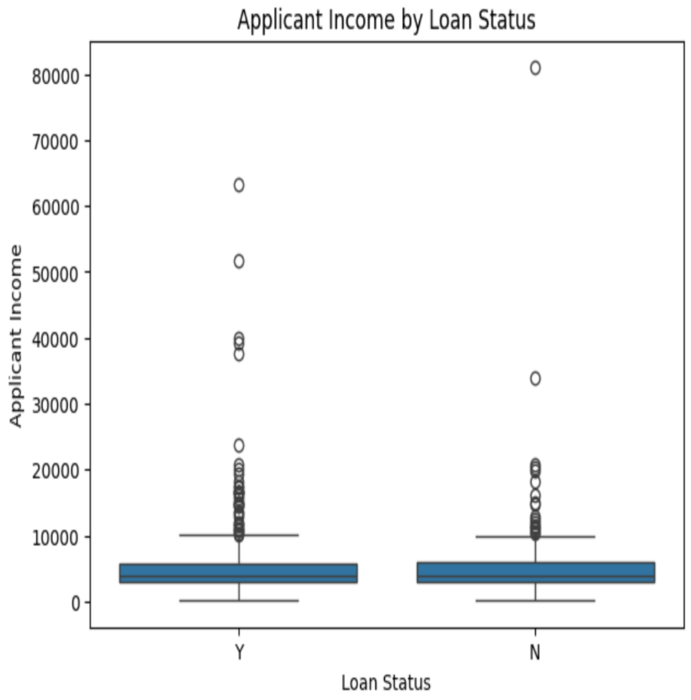
Graduates are more likely to get approved. This could be due to perceived financial stability or better employment prospects. However, education alone is not a deciding factor as non-graduates do get approved too. This indicates that the model includes other independent variables that influence the dependent variable.



Most loan amounts are clustered below 300 and higher loan amounts do not necessarily reduce approval chances. Both approved and rejected loans exist across the full loan amount range, meaning approval may depend more on other factors.

The box plots below reveal the presence of numerous outliers in the dataset, indicating the potential need for data cleaning. Both loan amount and applicant income have similar distributions across approval statuses, indicating that neither variable alone strongly differentiates approved vs rejected applicants.

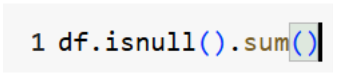

Loan Amount Distribution by Loan Approval Status
Loan Amount
700
600
500
400
300
200
100
0
Y
N
Loan Status


Applicant Income by Loan Status
Applicant Income
80000
70000
60000
50000
40000
30000
20000
10000
0
Y
N
Loan Status

Preprocessing

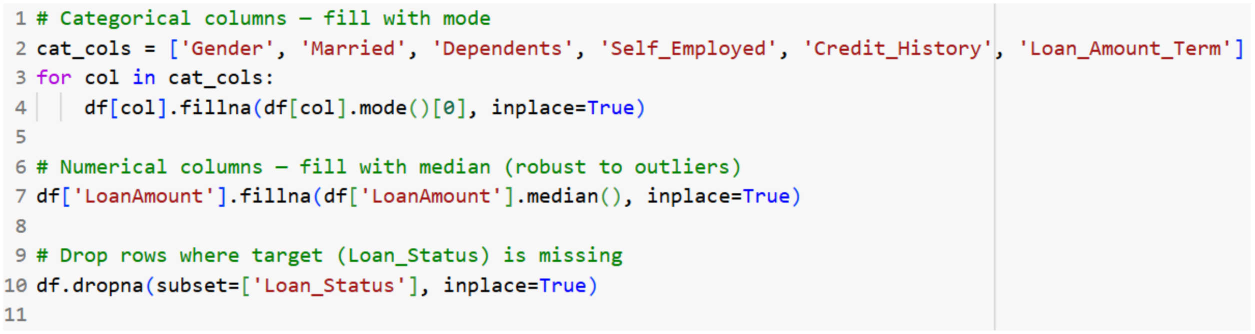
Handling missing values

Checking for missing values



```python
df.isnull().sum()
```

Common fixes based on columns

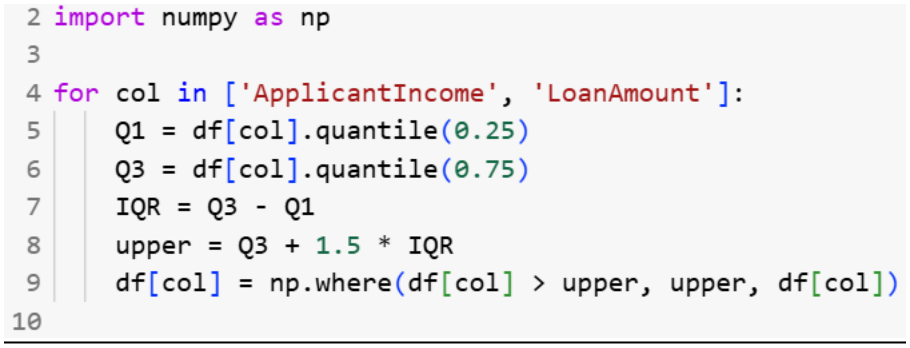


```python
# Categorical columns — fill with mode
cat_cols = ['Gender', 'Married', 'Dependents', 'Self_Employed', 'Credit_History', 'Loan_Amount_Term']
for col in cat_cols:
    df[col].fillna(df[col].mode()[0], inplace=True)

# Numerical columns — fill with median (robust to outliers)
df['LoanAmount'].fillna(df['LoanAmount'].median(), inplace=True)

# Drop rows where target (Loan_Status) is missing
df.dropna(subset=['Loan_Status'], inplace=True)

```

Rationale:

Handling missing values is essential in data preprocessing to ensure data quality and avoid bias or errors in model training. If not addressed, missing values can lead to inaccurate results or cause some machine learning algorithms to fail.

Handling outliers

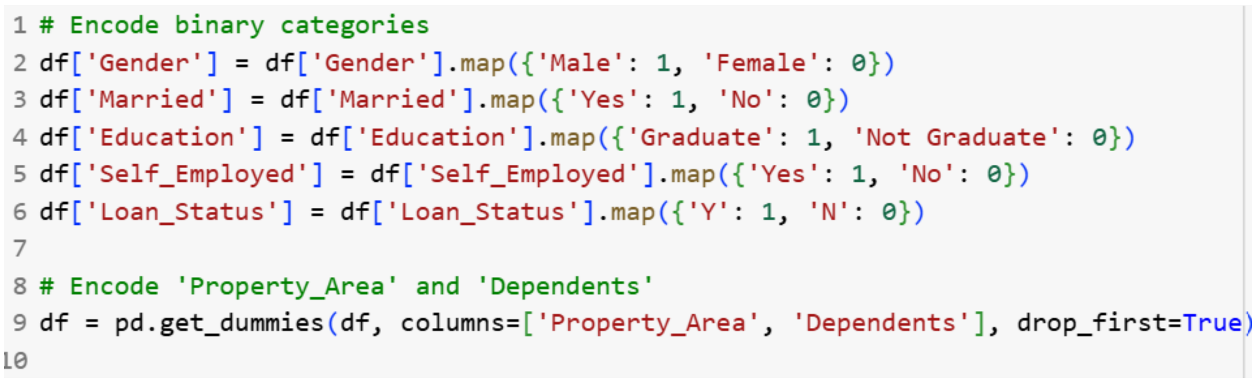


```python
import numpy as np

for col in ['ApplicantIncome', 'LoanAmount']:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    upper = Q3 + 1.5 * IQR
    df[col] = np.where(df[col] > upper, upper, df[col])

```

Rationale:

Instead of removing outliers, which can lead to information loss, IQR method caps them at the upper bound. This preserves the dataset's size while reducing the impact of extreme values, leading to more robust and reliable model performance.

Encode Categorical Variables

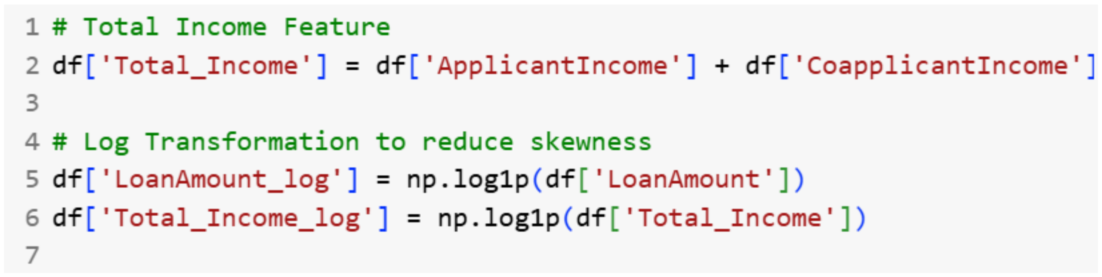


```python
# Encode binary categories
df['Gender'] = df['Gender'].map({'Male': 1, 'Female': 0})
df['Married'] = df['Married'].map({'Yes': 1, 'No': 0})
df['Education'] = df['Education'].map({'Graduate': 1, 'Not Graduate': 0})
df['Self_Employed'] = df['Self_Employed'].map({'Yes': 1, 'No': 0})
df['Loan_Status'] = df['Loan_Status'].map({'Y': 1, 'N': 0})

# Encode 'Property_Area' and 'Dependents'
df = pd.get_dummies(df, columns=['Property_Area', 'Dependents'], drop_first=True)

```

Rationale:

This step ensures that all features are in a numerical format suitable for machine learning algorithms, improving model accuracy, convergence, and robustness.

Feature Engineering

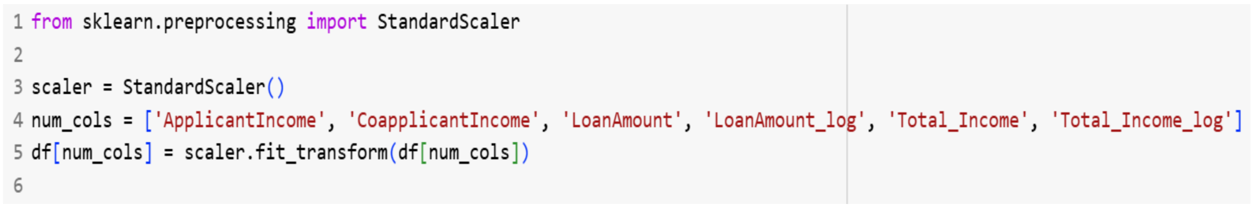


```python
# Total Income Feature
df['Total_Income'] = df['ApplicantIncome'] + df['CoapplicantIncome']

# Log Transformation to reduce skewness
df['LoanAmount_log'] = np.log1p(df['LoanAmount'])
df['Total_Income_log'] = np.log1p(df['Total_Income'])

```

Rationale:

Feature engineering enhances model performance by creating new, more informative variables. In this step, a new feature, Total Income, was created by combining Applicant Income and Coapplicant Income to capture the household's overall earning capacity an important factor in assessing loan repayment ability.

Scale Numerical Features

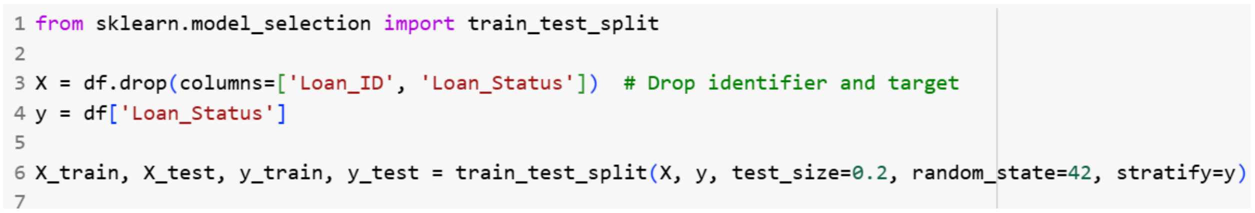


```python
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
num_cols = ['ApplicantIncome', 'CoapplicantIncome', 'LoanAmount', 'LoanAmount_log', 'Total_Income', 'Total_Income_log']
df[num_cols] = scaler.fit_transform(df[num_cols])

```

Rationale:

In this step, the StandardScaler was used to standardize features by removing the mean and scaling to unit variance. This means each transformed feature will have a mean of 0 and a standard deviation of 1. Without this transformation, features with larger numeric ranges such as income and loan amount could dominate the model, leading to biased or unstable results.

Train-Test Split



```python
from sklearn.model_selection import train_test_split

X = df.drop(columns=['Loan_ID', 'Loan_Status'])  # Drop identifier and target
y = df['Loan_Status']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

```

Rationale:

This process allows for unbiased performance evaluation and prevents overfitting the training data.

<u>Business Questions</u>

For Credit Risk Department

What are the key factors influencing whether a loan is approved or not? For example, does Applicant Income, Credit History, Employment Type, Property Area or Education Level have the strongest impact on loan approval?

This question focuses on identifying the most impactful predictors of loan approval and understanding these drivers can help the Credit Risk Department refine approval policies and risk thresholds, build scoring models for automated decision-making and mitigate potential default risk by adjusting weight on weak indicators.

For Marketing Team

What demographic groups such as gender or marital status have higher or lower loan approval rates?

By analyzing approval rates by gender and marital status, the Marketing team can tailor financial products to underrepresented or underserved groups, ensure fairness and inclusiveness in lending and design more targeted loan campaigns.

For Lending Officers

Does the requested loan amount significantly affect approval chances? Are smaller loans more likely to be approved than larger ones?

This question aims to determine if smaller loan requests are more likely to be approved than larger ones. Lending officers can use this insight to set realistic expectations for applicants, adjust risk-based pricing or collateral requirements and train staff on balancing loan size with creditworthiness.

Modeling

In this project I used the following models:

Logistic Regression

Rationale

I chose logistic regression because it is highly interpretable and well-suited for binary classification tasks, such as predicting loan approval status. This model provides clear insights into how individual features such as Credit History and Loan Amount impact the probability of loan approval, making it valuable for stakeholder understanding and decision-making.

Random Forest

Rationale:

I selected the Random Forest model due to its robustness to outliers and its ability to handle complex, nonlinear relationships. This was particularly beneficial given the presence of significant outliers in variables like Applicant Income and Loan Amount. Additionally, Random Forests reduce the risk of overfitting and provide feature importance rankings, which enhance the model's explanatory power.

Performance Metrics

Metric	**Logistic Regression**	**Random Forest**
Accuracy	0.87	0.86
Precision	0.84	0.88
Recall	0.99	0.93
F1 Score	0.91	0.90

Insights

What are the key factors influencing loan approval? (Credit Risk Department)

The use of the model, especially Logistic Regression, interprets feature importance via coefficients.

Top Positive Influencers:

Credit History: Strong positive correlation with loan approval. Applicants with a good credit history are far more likely to be approved.

Applicant Income and Coapplicant Income: Moderate influence. Higher incomes increase approval chances.

Education: Graduates are more likely to be approved.

Self Employed: Self-employed applicants may have slightly lower chances depending on income levels.

Top Negative Influencers:

Loan Amount: Larger amounts may reduce approval odds.

Property Area: Rural areas have lower approval rate than Semiurban/Urban areas.

Insight: Credit History is the most significant predictor. Making it a mandatory field and verifying it early can streamline approvals and reduce risk.

What demographic groups have higher/lower loan approval rates? (Marketing Team)

Gender: Male applicants have higher approval rates. This may indicate a potential bias or societal income disparities that require further investigation.

Marital Status: Married applicants have slightly higher approval rates, likely due to combined income.

Education: Graduates are more likely to be approved, as seen in both logistic coefficients and decision tree splits.

Insight: Marketing campaigns should target graduate, married applicants with a verifiable credit history. However, gender disparity should be explored to ensure fairness and regulatory compliance.

Does loan amount affect approval chances? (Lending Officers)

Smaller loans are more likely to be approved while very large loan amounts might trigger stricter checks, leading to more rejections.

Logistic Regression shows a negative coefficient for Loan Amount.

Insight: Lending officers should consider pre-approving smaller loan ranges and introduce stricter scrutiny policies for higher amounts. Also, offering loan restructuring or phased disbursement could help increase approvals for larger loans.

Ethics & Interpretability

The use of predictive models in loan approval decisions raises important ethical considerations, particularly around fairness, transparency, and potential bias. Features such as gender, marital status, or property area, while predictive, may unintentionally reinforce social inequalities or discriminate against certain demographic groups if not handled responsibly. For instance, if a model learns patterns based on historical biases in loan approvals, it may perpetuate unfair treatment toward women, unmarried applicants, or individuals from rural areas even if these factors are not directly linked to creditworthiness.

To address these concerns, I prioritized interpretability by including models like logistic regression, which offer clear insights into how each feature influences the outcome. This transparency is essential for building stakeholder trust and aligning the model with regulatory requirements in the financial industry. Additionally, careful attention was paid to feature selection and encoding, and sensitive variables were evaluated for fairness impact.

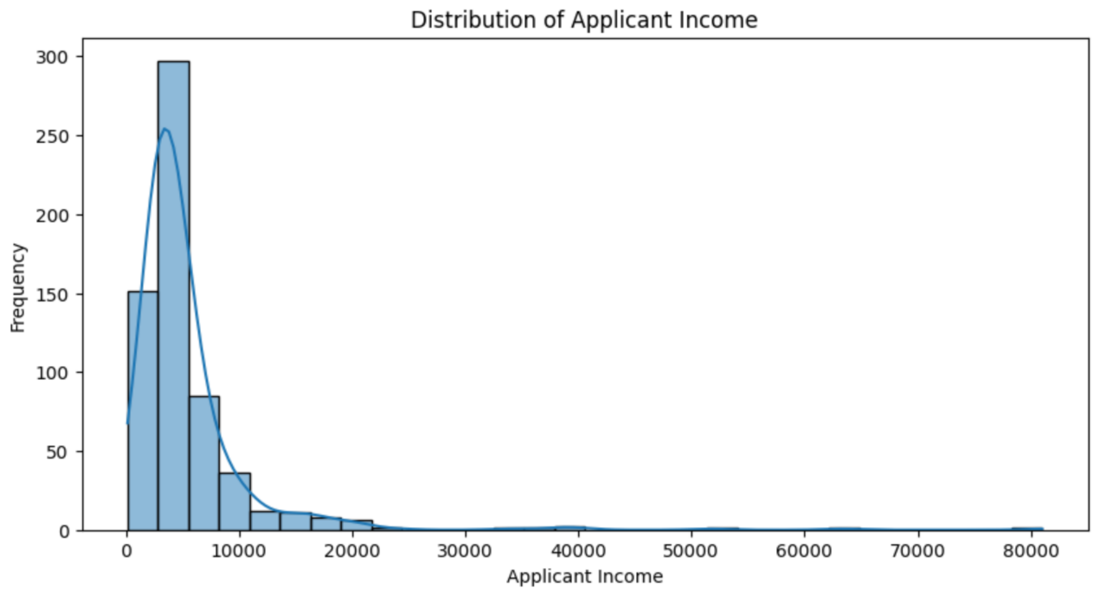
Appendix (code, visuals)

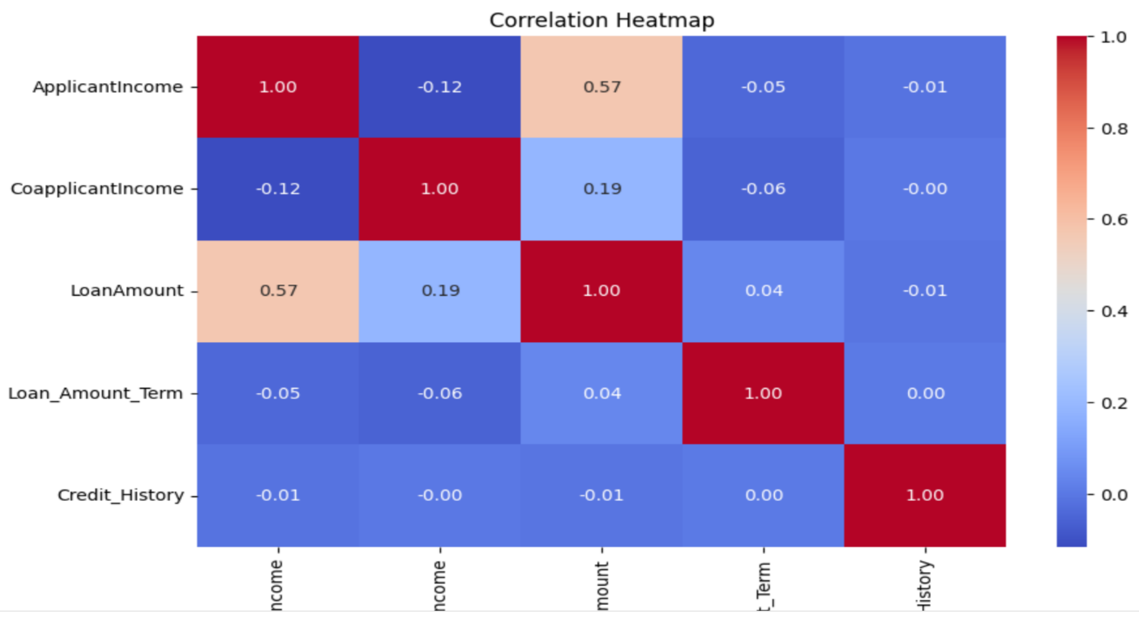
```
[ ]  from google.colab import files
     uploaded = files.upload()
```

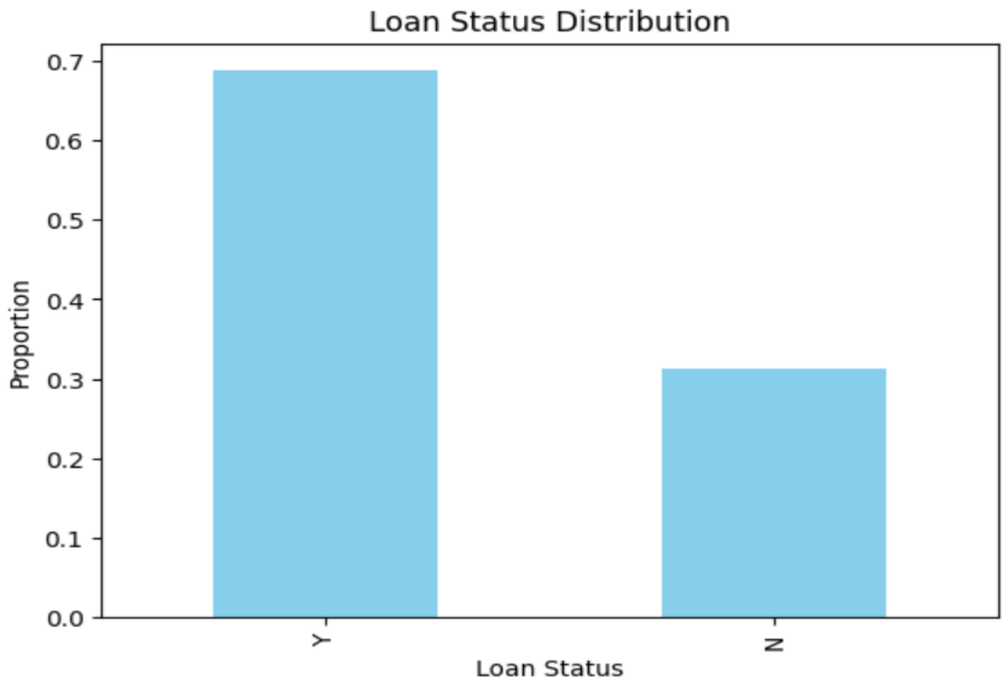
```
Choose Files  No file chosen      Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving Original Dataset.xlsx to Original Dataset.xlsx
```

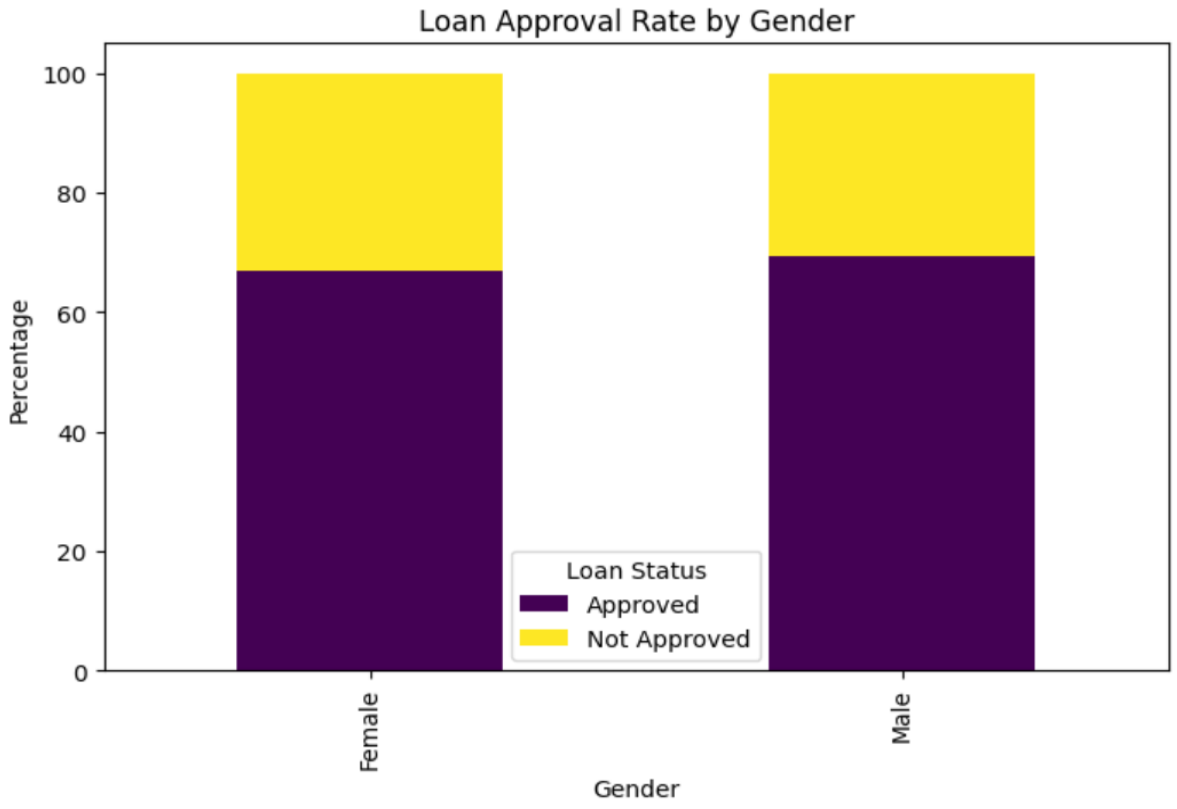
```
[ ]  import pandas as pd
     xls = pd.ExcelFile('Original Dataset.xlsx')
     df = xls.parse('Data')
```

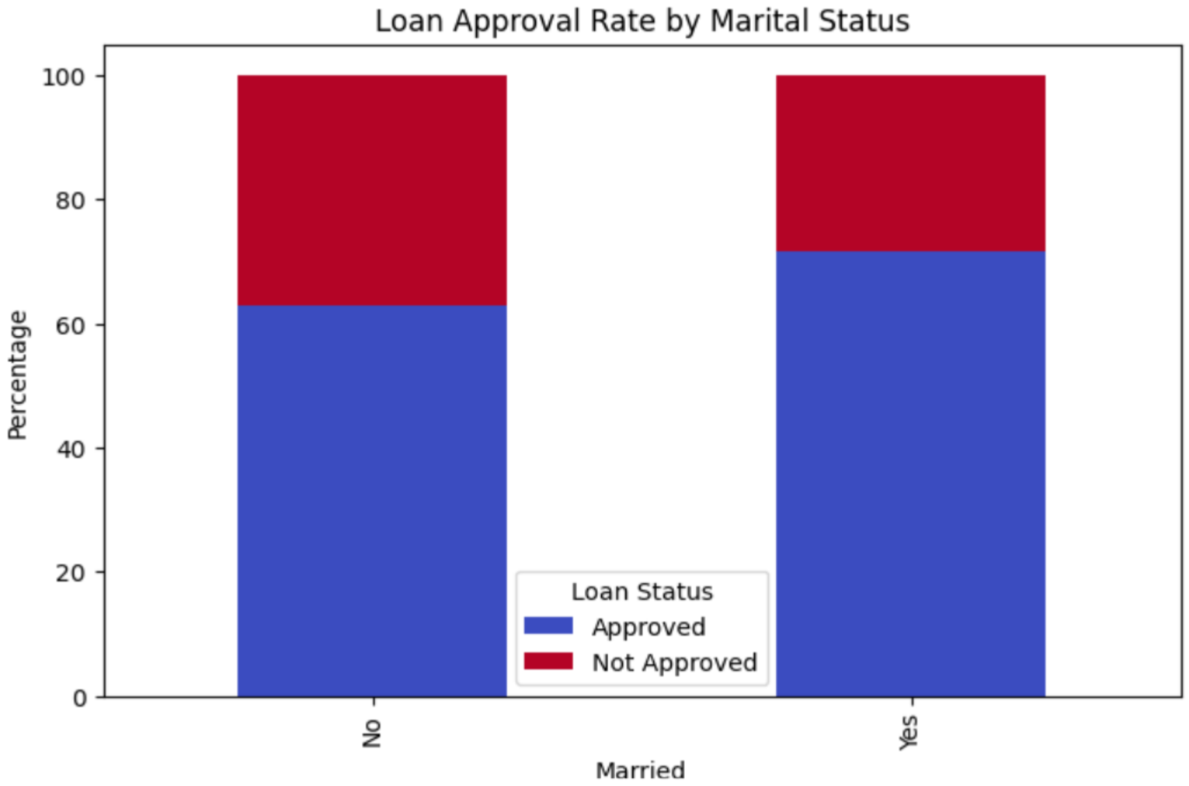
```
[ ]  import matplotlib.pyplot as plt
     import seaborn as sns
     plt.figure(figsize=(10, 5))
     sns.histplot(df['ApplicantIncome'], kde=True, bins=30)
     plt.title('Distribution of Applicant Income')
     plt.xlabel('Applicant Income')
     plt.ylabel('Frequency')
     plt.show()

```

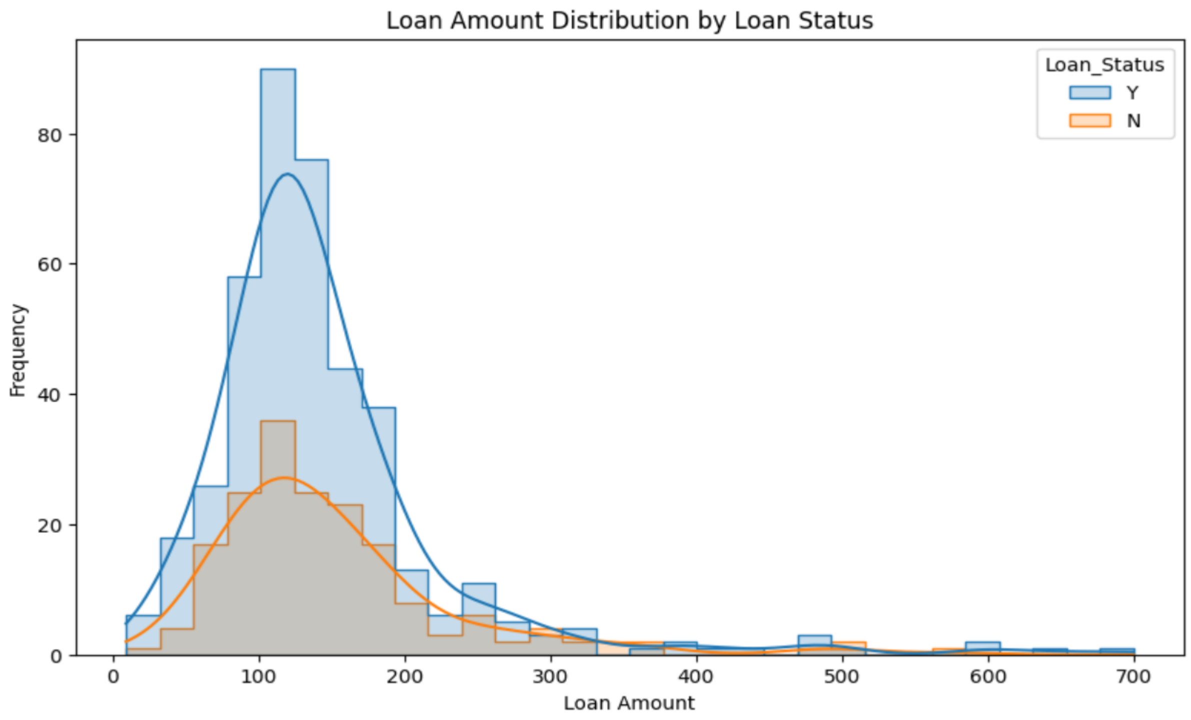


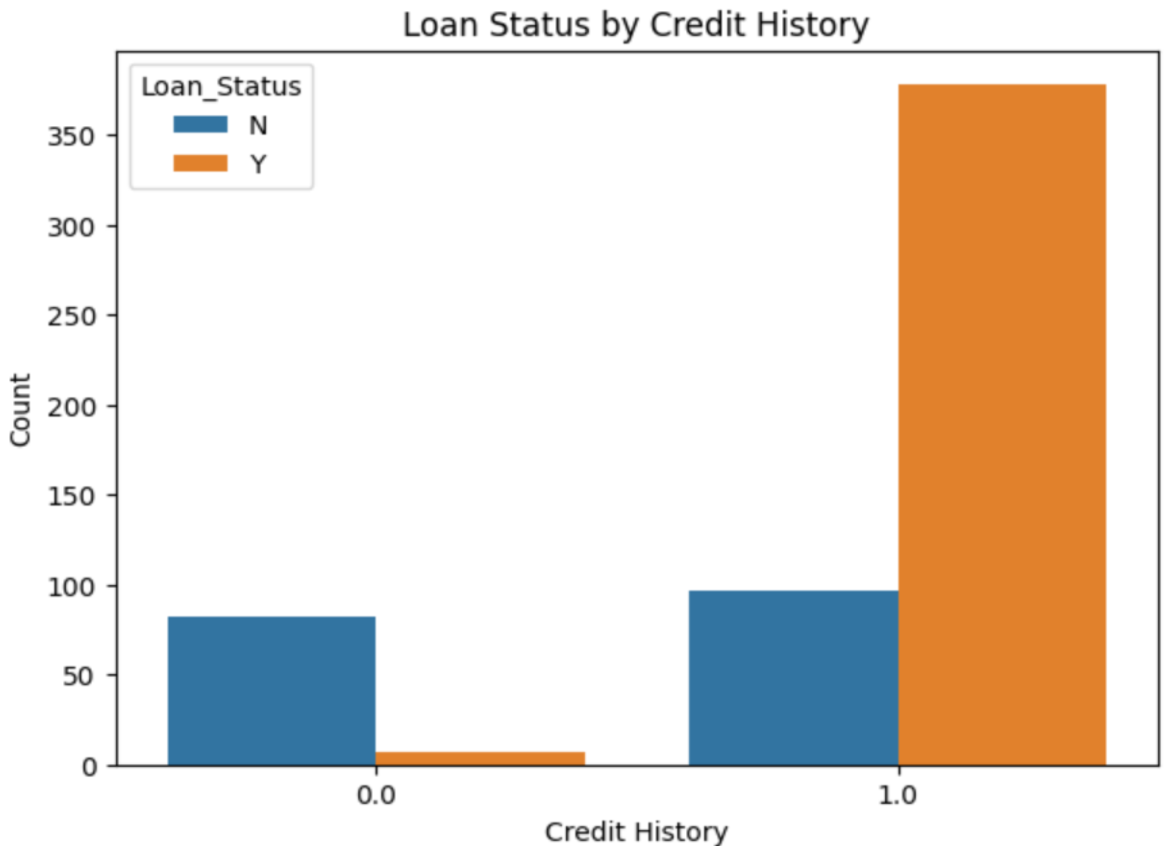

Correlation Heatmap
ApplicantIncome
CoapplicantIncome
LoanAmount
Loan_Amount_Term
Credit_History
1.00
-0.12
0.57
-0.05
-0.01
-0.12
1.00
0.19
-0.06
-0.00
0.57
0.19
1.00
0.04
-0.01
-0.05
-0.06
0.04
1.00
0.00
-0.01
-0.00
-0.01
0.00
1.00
ncome
ncome
mount
t_Term
History
1.0
0.8
0.6
0.4
0.2
0.0

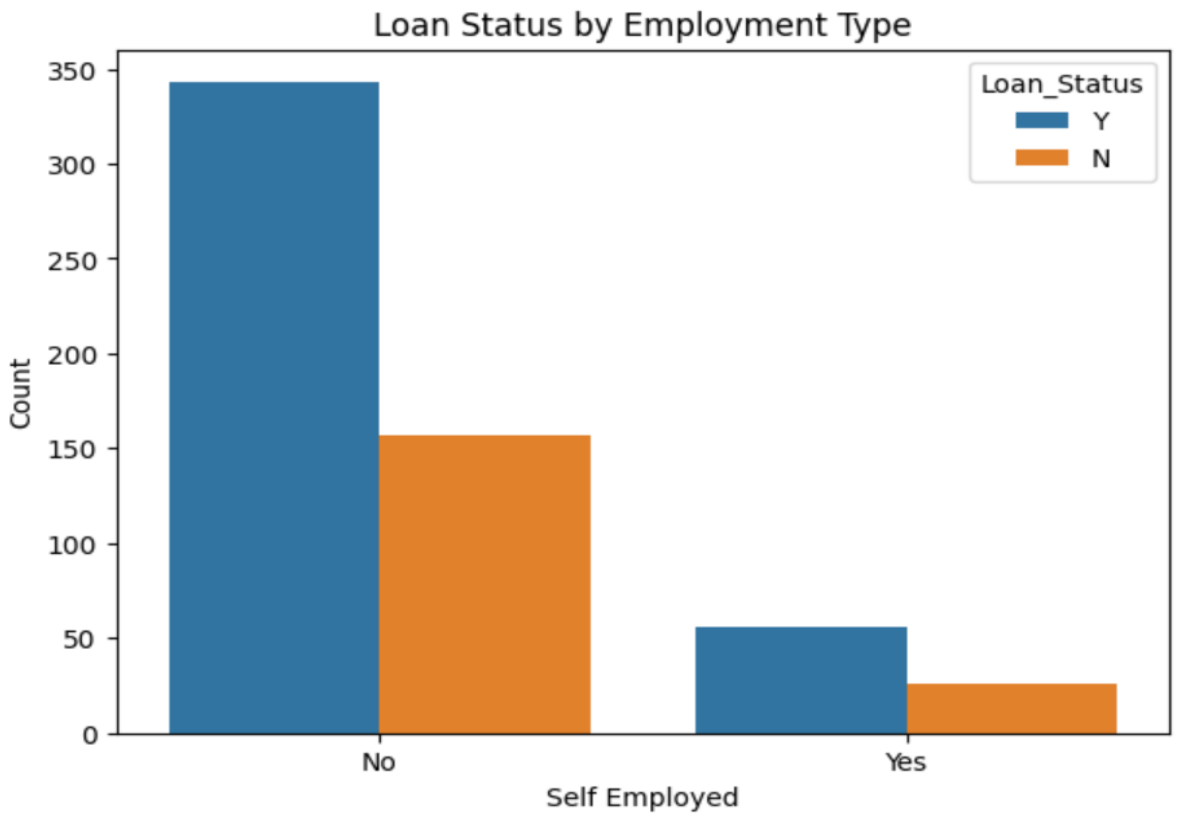

Loan Status Distribution
Proportion
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
Y
N
Loan Status

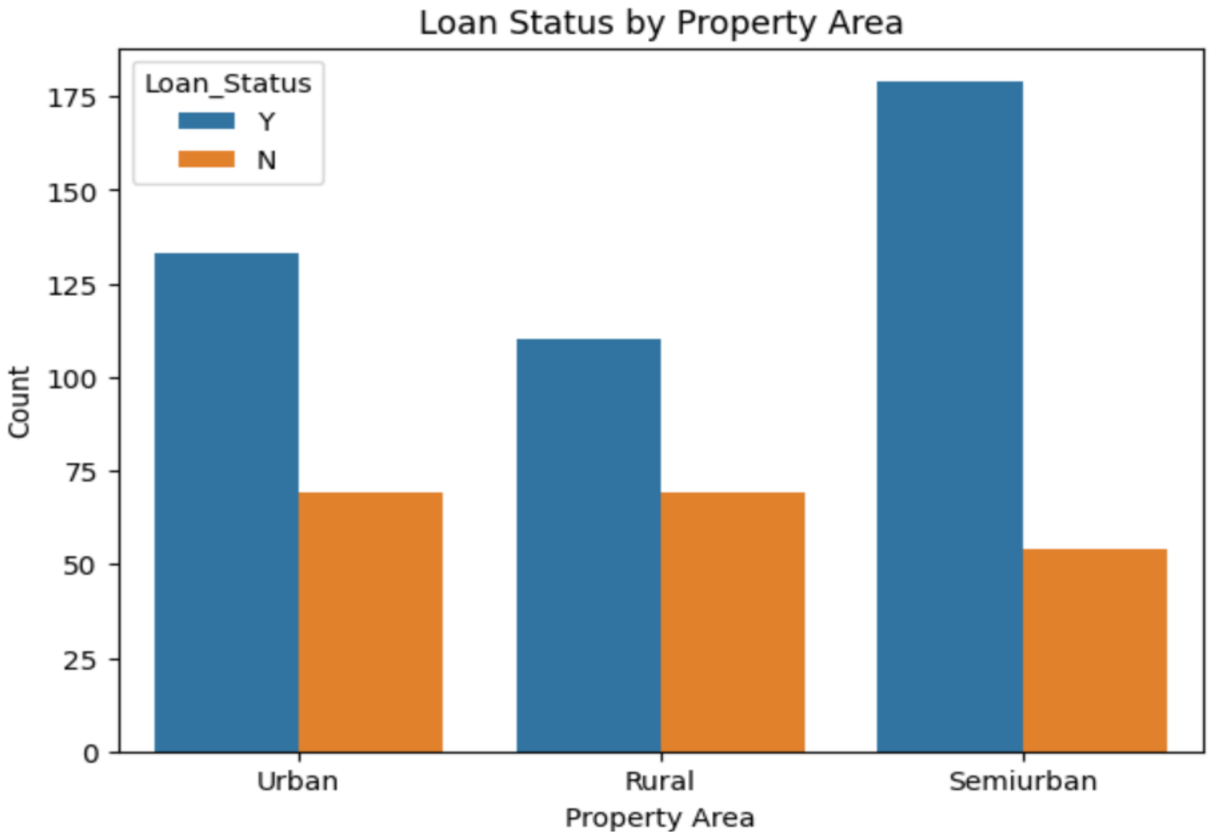

Loan Approval Rate by Gender
Percentage
0
20
40
60
80
100
Loan Status
Approved
Not Approved
Female
Male
Gender

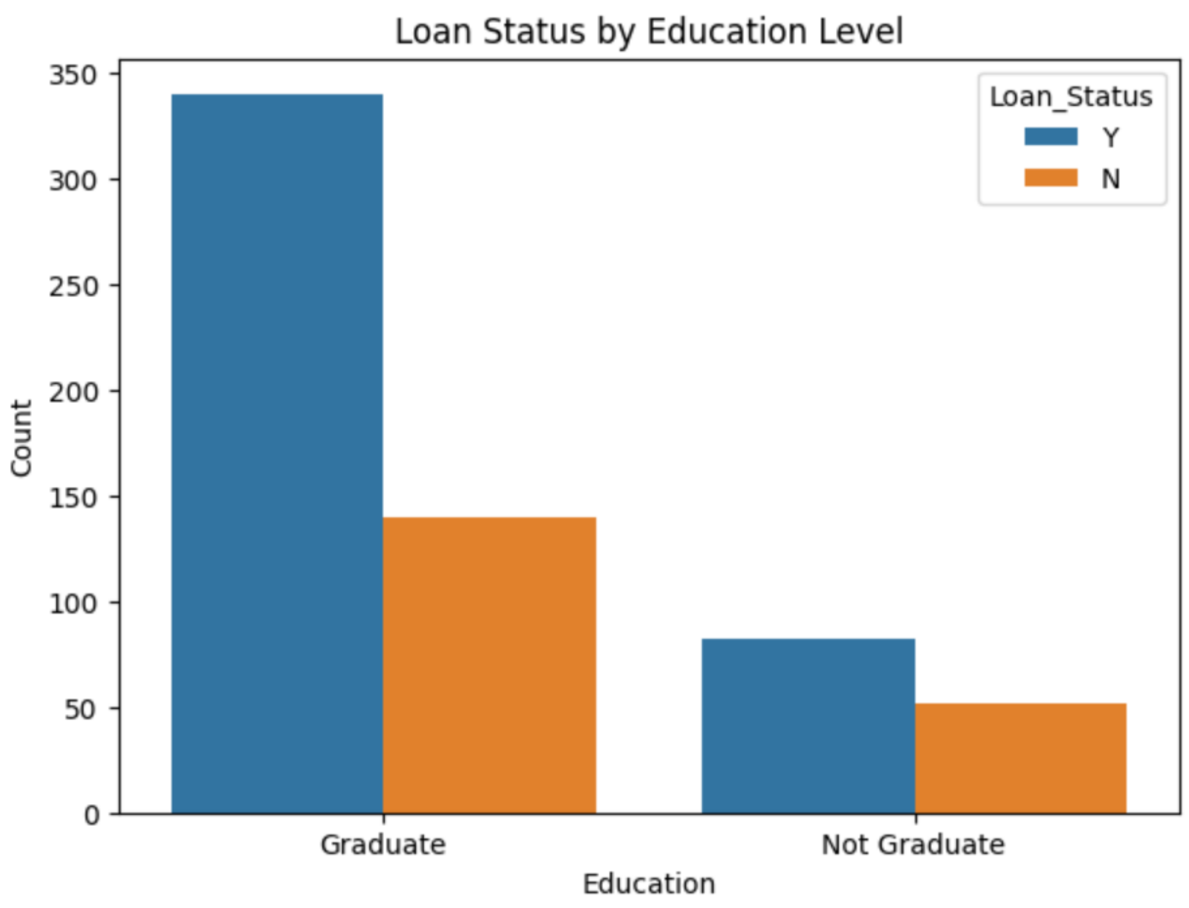

Loan Approval Rate by Marital Status
Percentage
0
20
40
60
80
100
Loan Status
Approved
Not Approved
No
Yes
Married


Loan Amount Distribution by Loan Status
Loan_Status
Y
N
Frequency
Loan Amount
0
20
40
60
80
100
200
300
400
500
600
700


Loan Status by Credit History
Loan_Status
N
Y
Count
Credit History
0
50
100
150
200
250
300
350
0.0
1.0


Loan Status by Employment Type
Loan_Status
Y
N
Count
350
300
250
200
150
100
50
0
No
Yes
Self Employed


Loan Status by Property Area
Loan_Status
Y
N
Count
175
150
125
100
75
50
25
0
Urban
Rural
Semiurban
Property Area



```
[ ]  from google.colab import files
     uploaded = files.upload()
```

```
Choose Files No file chosen      Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving Original Dataset.xlsx to Original Dataset.xlsx
```

```
[ ]  import pandas as pd

     # Load dataset
     df = pd.read_excel('Original Dataset.xlsx')

     # Initial inspection
     df.info()
     df.describe()
     df.head()

```

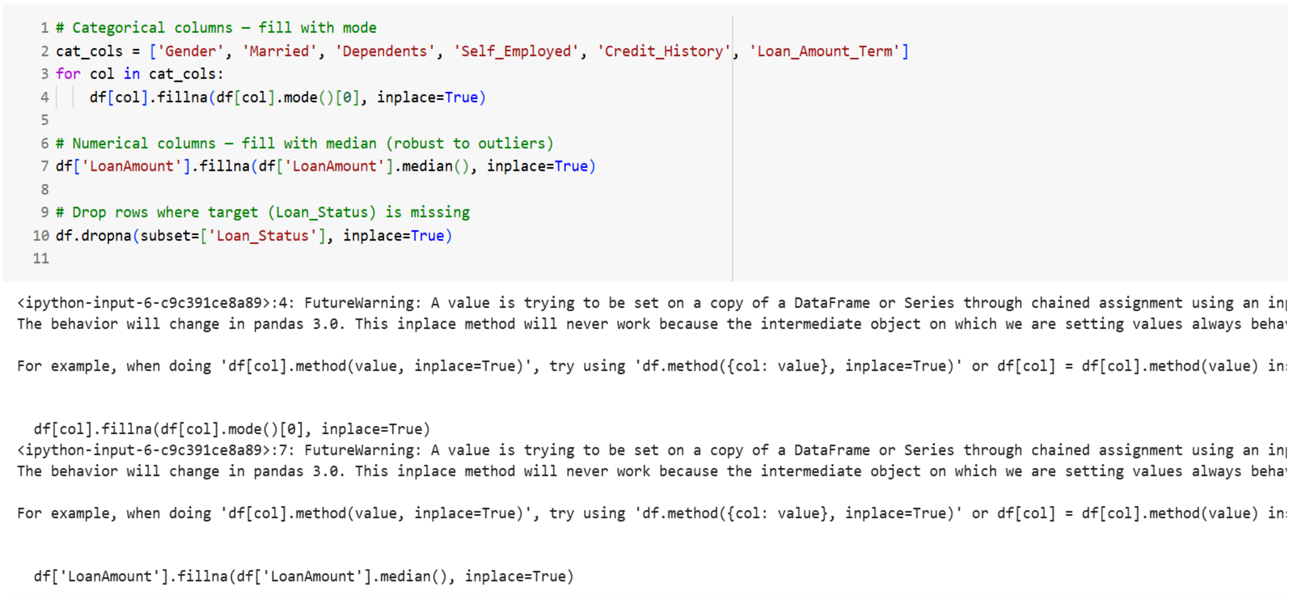
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Loan_ID            614 non-null    object 
 1   Gender             601 non-null    object 
 2   Married            611 non-null    object 
 3   Dependents         599 non-null    object 
 4   Education          614 non-null    object 
 5   Self_Employed      582 non-null    object 
 6   ApplicantIncome    614 non-null    int64  
 7   CoapplicantIncome  614 non-null    float64
 8   LoanAmount         592 non-null    float64
 9   Loan_Amount_Term   600 non-null    float64
 10  Credit_History     564 non-null    float64
 11  Property_Area      614 non-null    object 
 12  Loan_Status        614 non-null    object 
dtypes: float64(4), int64(1), object(8)
memory usage: 62.5+ KB
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	P
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0	
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	

```python
df.isnull().sum()
```

	0
Loan_ID	0
Gender	13
Married	3
Dependents	15
Education	0
Self_Employed	32
ApplicantIncome	0
CoapplicantIncome	0
LoanAmount	22
Loan_Amount_Term	14
Credit_History	50
Property_Area	0
Loan_Status	0

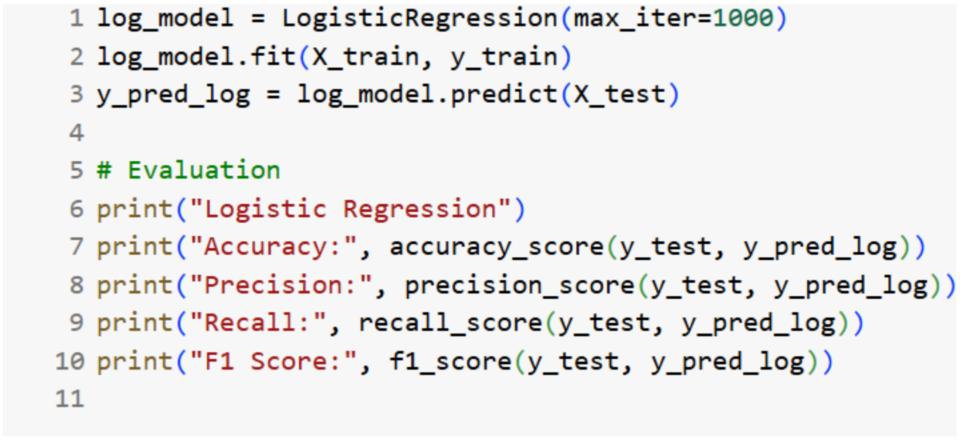
dtype: int64



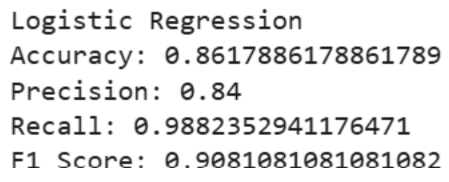
```python
# Categorical columns – fill with mode
cat_cols = ['Gender', 'Married', 'Dependents', 'Self_Employed', 'Credit_History', 'Loan_Amount_Term']
for col in cat_cols:
    df[col].fillna(df[col].mode()[0], inplace=True)

# Numerical columns – fill with median (robust to outliers)
df['LoanAmount'].fillna(df['LoanAmount'].median(), inplace=True)

# Drop rows where target (Loan_Status) is missing
df.dropna(subset=['Loan_Status'], inplace=True)

```

```
<ipython-input-6-c9c391ce8a89>:4: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an in
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 beha

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) in


  df[col].fillna(df[col].mode()[0], inplace=True)
<ipython-input-6-c9c391ce8a89>:7: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an in
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 beha

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) in


  df['LoanAmount'].fillna(df['LoanAmount'].median(), inplace=True)
```

Logistic Regression

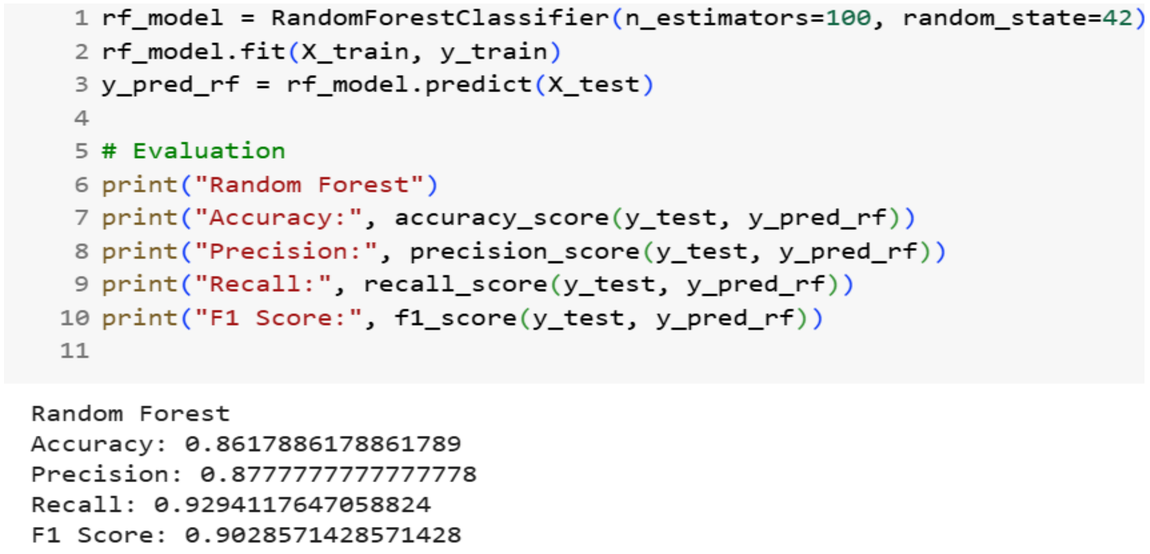


```python
log_model = LogisticRegression(max_iter=1000)
log_model.fit(X_train, y_train)
y_pred_log = log_model.predict(X_test)

# Evaluation
print("Logistic Regression")
print("Accuracy:", accuracy_score(y_test, y_pred_log))
print("Precision:", precision_score(y_test, y_pred_log))
print("Recall:", recall_score(y_test, y_pred_log))
print("F1 Score:", f1_score(y_test, y_pred_log))

```



```
Logistic Regression
Accuracy: 0.8617886178861789
Precision: 0.84
Recall: 0.9882352941176471
F1 Score: 0.9081081081081082
```

Random Forest



```python
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
y_pred_rf = rf_model.predict(X_test)

# Evaluation
print("Random Forest")
print("Accuracy:", accuracy_score(y_test, y_pred_rf))
print("Precision:", precision_score(y_test, y_pred_rf))
print("Recall:", recall_score(y_test, y_pred_rf))
print("F1 Score:", f1_score(y_test, y_pred_rf))

```

```
Random Forest
Accuracy: 0.8617886178861789
Precision: 0.8777777777777778
Recall: 0.9294117647058824
F1 Score: 0.9028571428571428
```