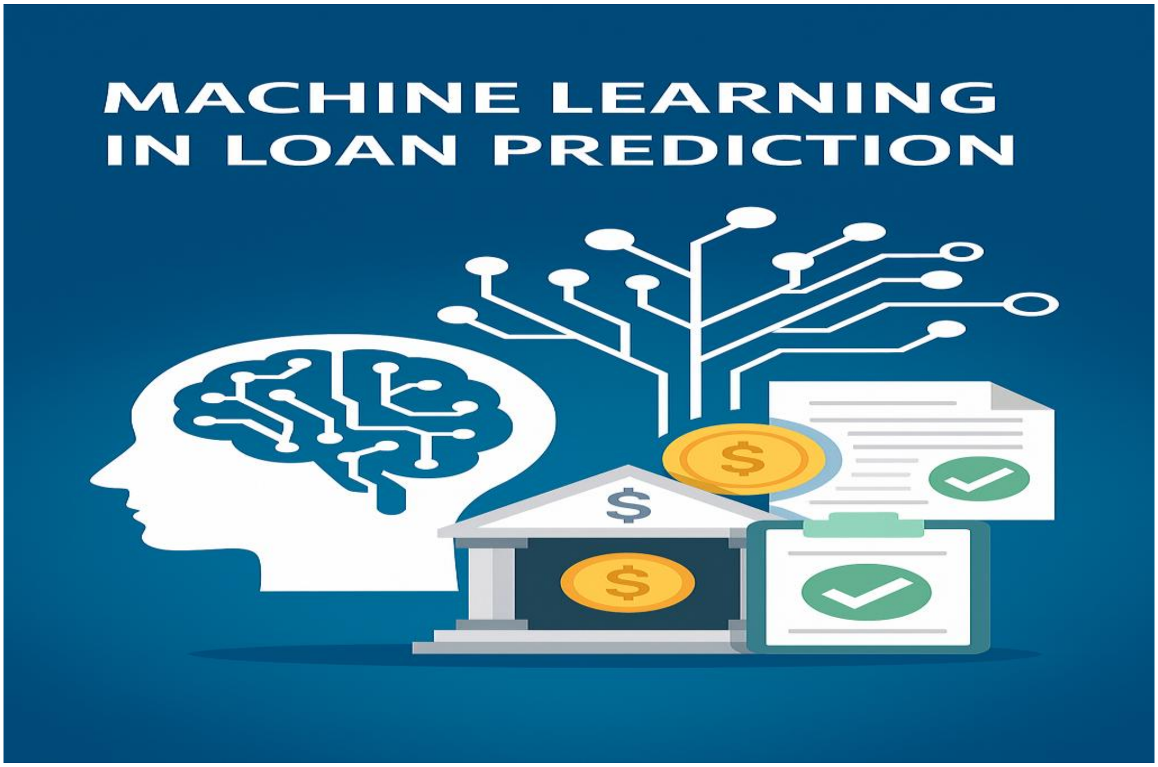
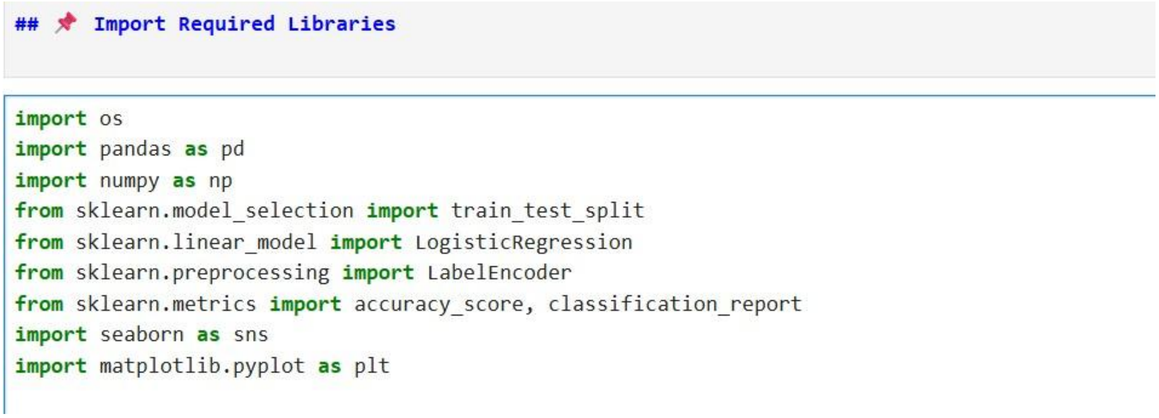
Loan Prediction Using Machine Learning



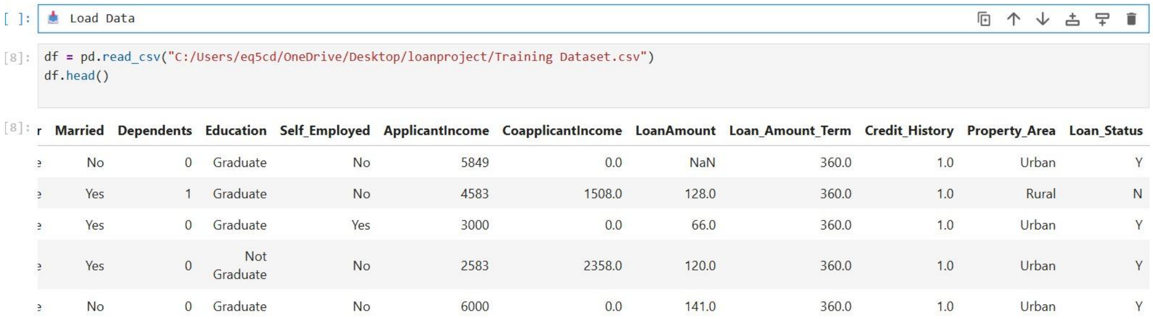
A Data Science Project using Logistic Regression

Presented By: Sahil



```
## 📌 Import Required Libraries
```

```python
import os
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score, classification_report
import seaborn as sns
import matplotlib.pyplot as plt
```

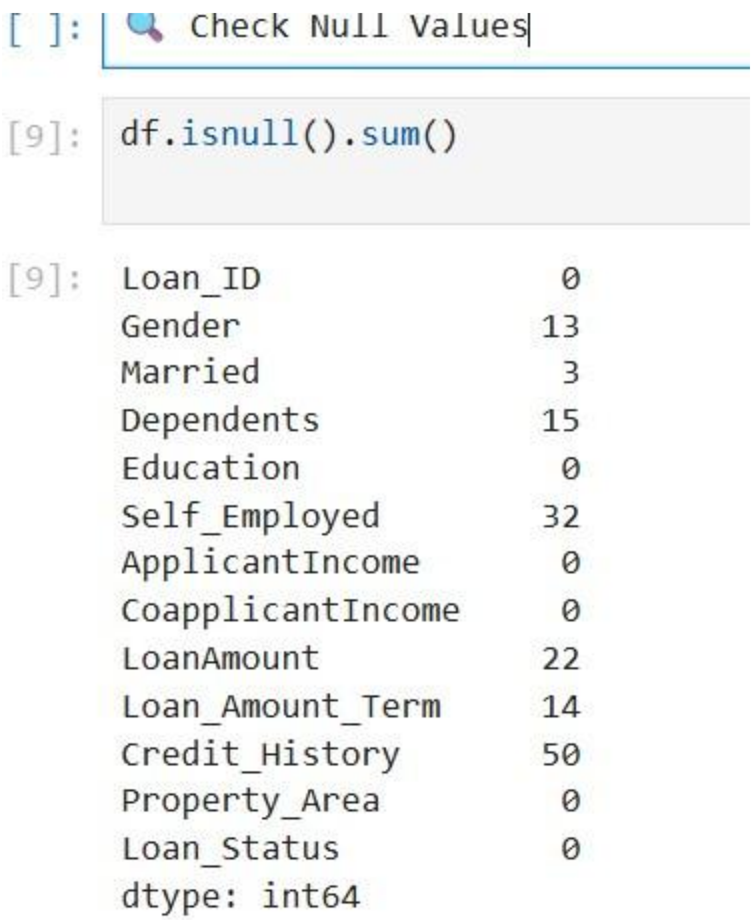


[ ]: Load Data

[8]:
```
df = pd.read_csv("C:/Users/eq5cd/OneDrive/Desktop/loanproject/Training Dataset.csv")
df.head()
```

[8]:

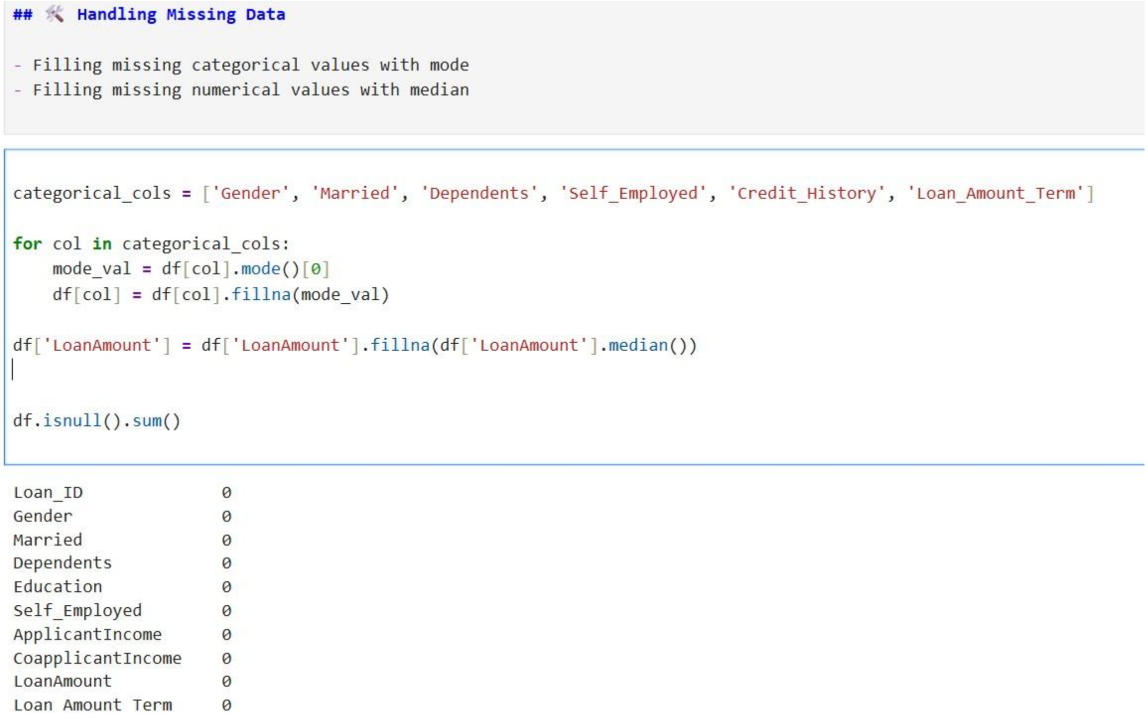
r	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
e	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0	Urban	Y
e	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
e	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
e	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
e	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y



```
[ ]: 🔍 Check Null Values
```

```
[9]: df.isnull().sum()
```

```
[9]: 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
```



```
## 🛠️ Handling Missing Data

- Filling missing categorical values with mode
- Filling missing numerical values with median
```

```python
categorical_cols = ['Gender', 'Married', 'Dependents', 'Self_Employed', 'Credit_History', 'Loan_Amount_Term']

for col in categorical_cols:
    mode_val = df[col].mode()[0]
    df[col] = df[col].fillna(mode_val)

df['LoanAmount'] = df['LoanAmount'].fillna(df['LoanAmount'].median())


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

```
Loan_ID              0
Gender               0
Married              0
Dependents           0
Education            0
Self_Employed        0
ApplicantIncome      0
CoapplicantIncome    0
LoanAmount           0
Loan Amount Term     0
```

```
## 🔤 Encoding Categorical Variables

Using LabelEncoder to convert categorical columns to numeric format.
```

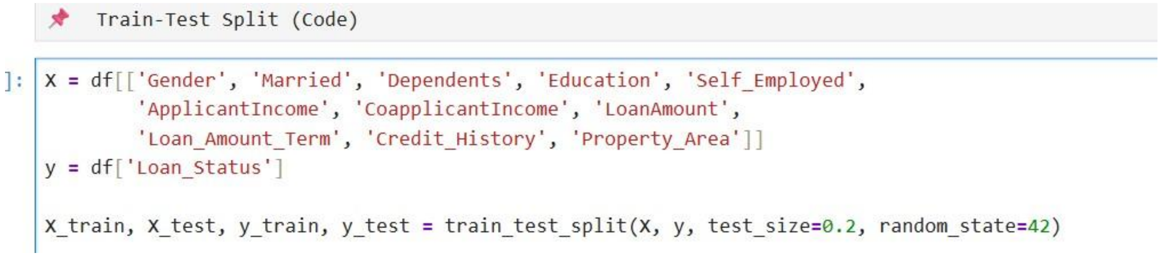
```python
df['Dependents'] = df['Dependents'].replace('3+', 3).astype(int)

cat_cols = ['Gender', 'Married', 'Dependents', 'Education', 'Self_Employed', 'Property_Area', 'Loan_Status']

le = LabelEncoder()
for col in cat_cols:
    df[col] = le.fit_transform(df[col])

df.head()
```

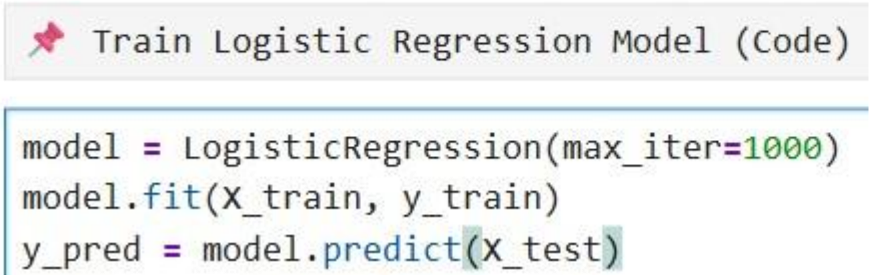
	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Propert
0	LP001002	1	0	0	0	0	5849	0.0	128.0	360.0	1.0	
1	LP001003	1	1	1	0	0	4583	1508.0	128.0	360.0	1.0	
2	LP001005	1	1	0	0	1	3000	0.0	66.0	360.0	1.0	
3	LP001006	1	1	0	1	0	2583	2358.0	120.0	360.0	1.0	
4	LP001008	1	0	0	0	0	6000	0.0	141.0	360.0	1.0	



📌 Train-Test Split (Code)

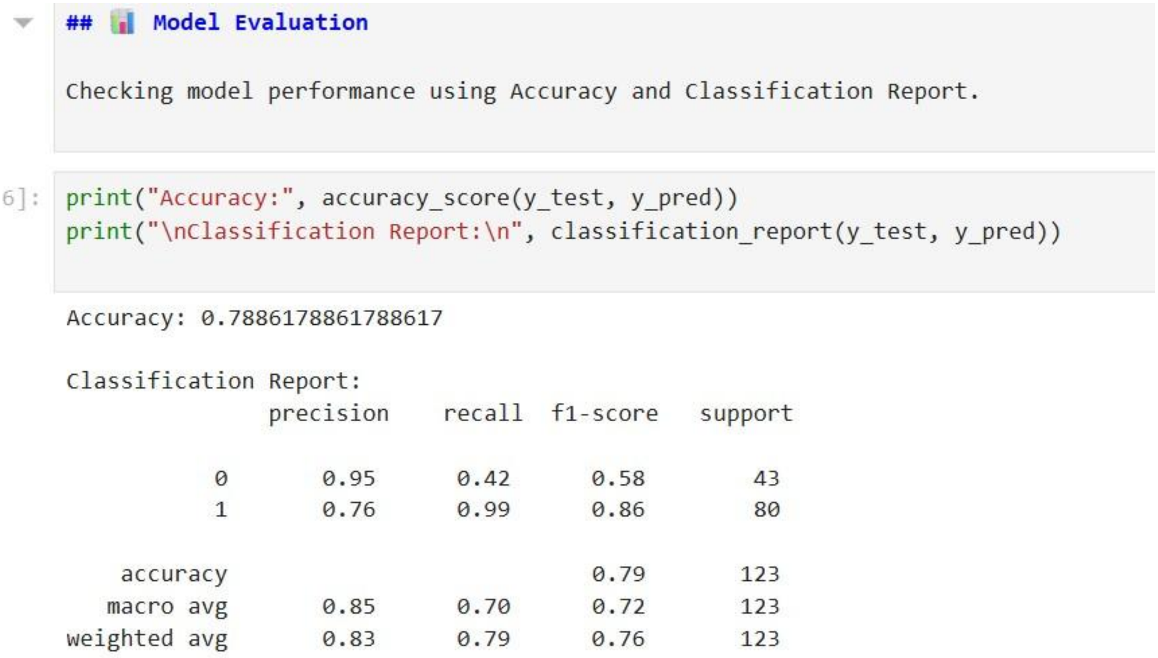
```
X = df[['Gender', 'Married', 'Dependents', 'Education', 'Self_Employed',
        'ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
        'Loan_Amount_Term', 'Credit_History', 'Property_Area']]
y = df['Loan_Status']

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



📌 Train Logistic Regression Model (Code)

```
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```



```
## 📊 Model Evaluation

Checking model performance using Accuracy and Classification Report.
```

6]:

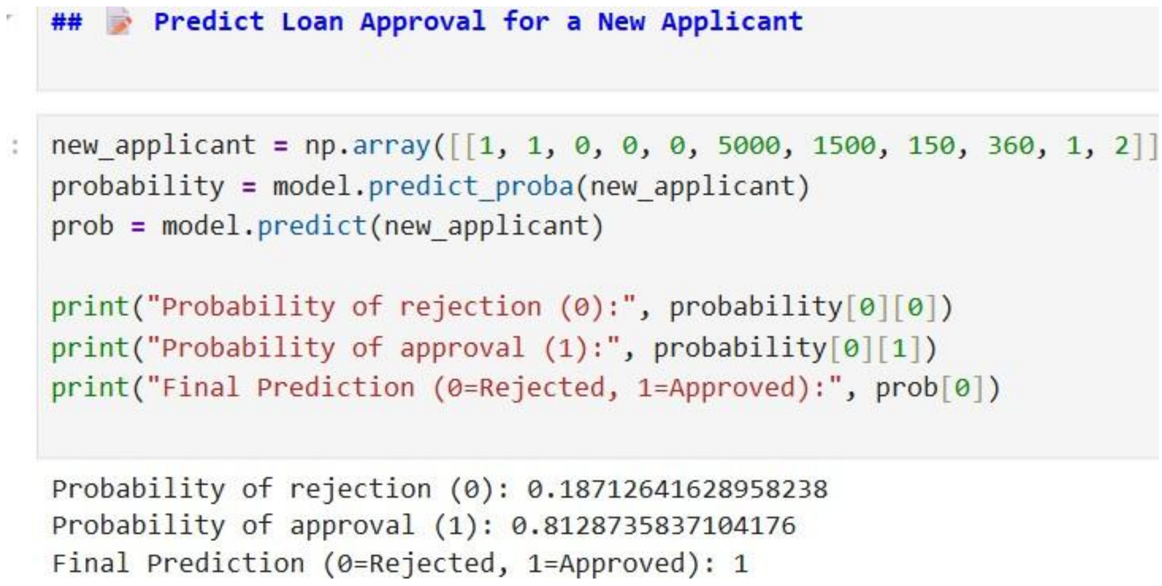
```python
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 0.7886178861788617

Classification Report:
               precision    recall  f1-score   support

           0       0.95      0.42      0.58        43
           1       0.76      0.99      0.86        80

    accuracy                           0.79       123
   macro avg       0.85      0.70      0.72       123
weighted avg       0.83      0.79      0.76       123
```

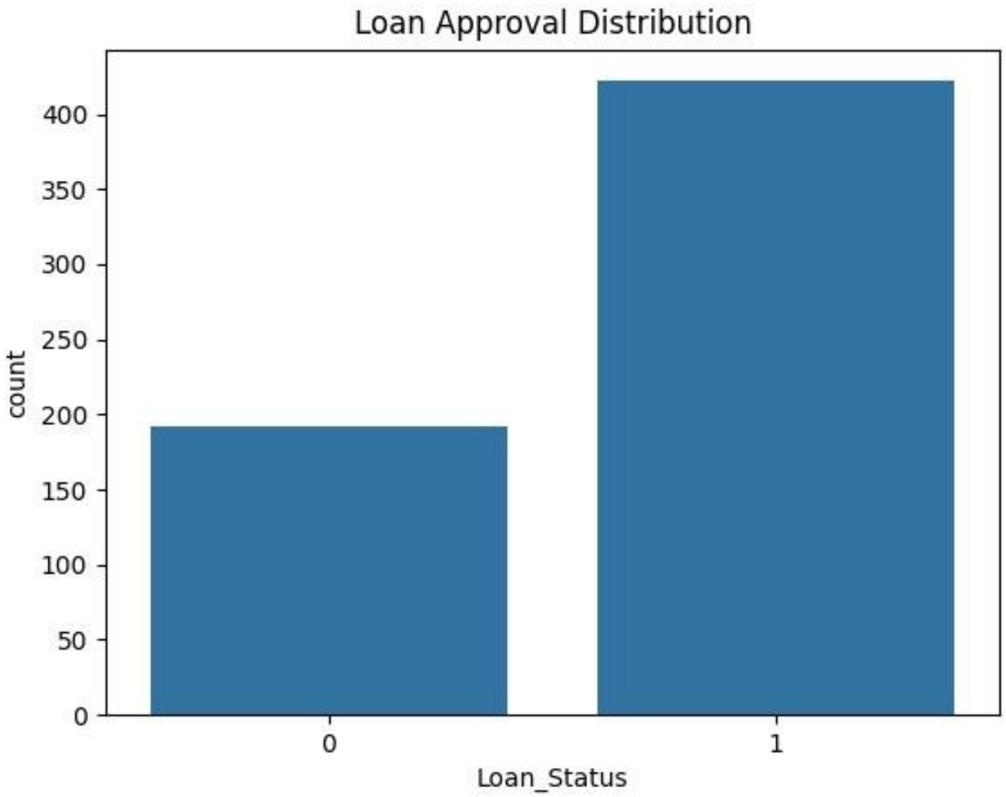


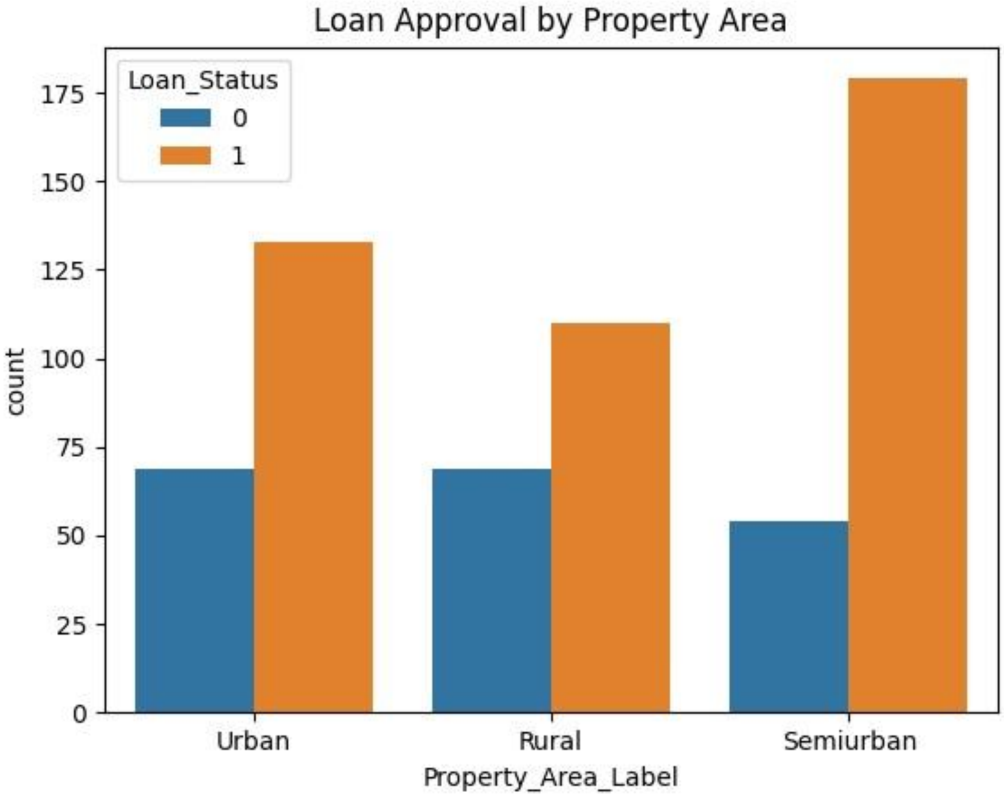
```
## 📝 Predict Loan Approval for a New Applicant
```

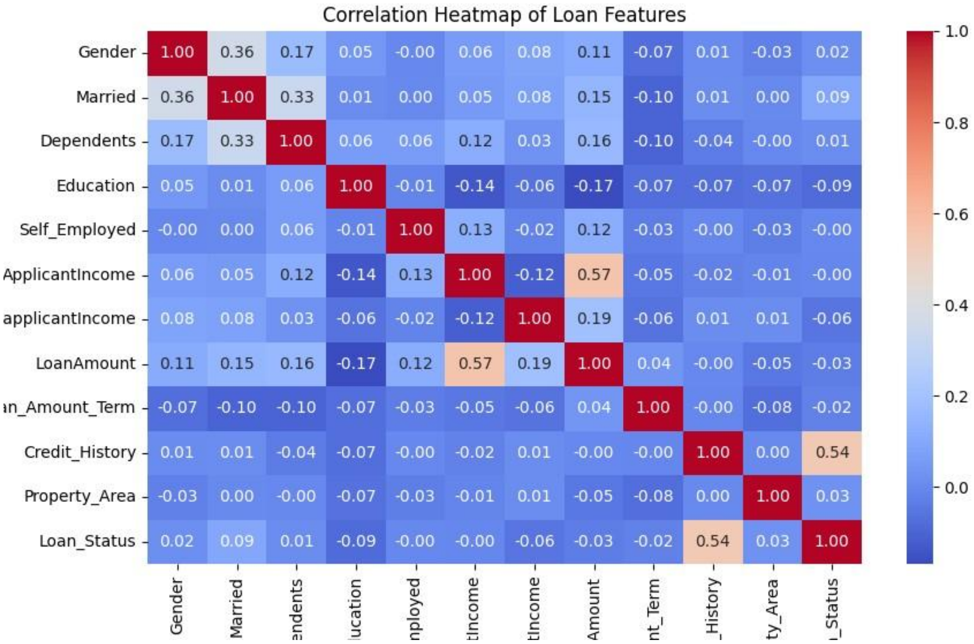
```python
new_applicant = np.array([[1, 1, 0, 0, 0, 5000, 1500, 150, 360, 1, 2]]
probability = model.predict_proba(new_applicant)
prob = model.predict(new_applicant)

print("Probability of rejection (0):", probability[0][0])
print("Probability of approval (1):", probability[0][1])
print("Final Prediction (0=Rejected, 1=Approved):", prob[0])
```

```
Probability of rejection (0): 0.18712641628958238
Probability of approval (1): 0.8128735837104176
Final Prediction (0=Rejected, 1=Approved): 1
```


Loan Approval Distribution
count
0
50
100
150
200
250
300
350
400
0
1
Loan_Status


Loan Approval by Property Area
Loan_Status
0
1
175
150
125
100
75
50
25
0
count
Urban
Rural
Semiurban
Property_Area_Label


Correlation Heatmap of Loan Features