



L OVELY
P ROFESSIONAL
U NIVERSITY

Transforming Education Transforming India

Course Code: MGN342	Course Title: BUSINESS ANALYTICS
Course Instructor: Dr. Pritpal Singh	
Academic Task No.: 2	Academic Task Title: Assignment
Date of Allotment: 24/03/2025	Date of submission: 07/04/2025
Student's Roll no: 10	Student's Reg. no: 12208114
Evaluation Parameters: (Parameters on which student is to be evaluated- To be mentioned by students as specified at the time of assigning the task by the instructor)	

Learning Outcomes: (Student to write briefly about learnings obtained from the academic tasks)

Declaration:

I declare that this Assignment is my individual work. I have not copied it from any other student's work or from any other source except where due acknowledgement is made explicitly in the text, nor has any part been written for me by any other person.

Student signature: Sumit Sarkar

Evaluator's comments (For Instructor's use only)

General Observations	Suggestions for Improvement	Best part of Assignment
-----------------------------	------------------------------------	--------------------------------

--	--	--

Evaluator's Signature and Date:

Marks Obtained: _____

Max. Marks: _____

INTRODUCTION -

- **Overview of the Dataset:**

- The "Loan Eligible Dataset" (loan-train.csv) contains information about loan applicants. It includes various attributes that are typically considered when evaluating loan eligibility, such as:
 - Applicant income
 - Co-applicant income
 - Loan amount
 - Loan term
 - Credit history
 - Property area
 - Gender
 - Marital status
 - Dependents.
- This dataset simulates a real-world scenario where a financial institution needs to assess the risk of lending money to individuals based on their characteristics. The target variable, "Loan_Status," indicates whether a loan was approved (Y) or not (N).
- This is a classic binary classification problem.

- **Objectives of the Analysis:**

- **Predictive Analytics (Forecasting):**
 - The primary goal is to build a predictive model that can accurately forecast whether a new loan applicant is likely to be approved or not. This model would enable the financial institution to automate the loan approval process, reduce manual effort, and improve decision-making speed.
 - The model should be able to predict the "Loan_Status" based on the other features.
- **Prescriptive Analytics (Decision Optimization):**

- Beyond prediction, the analysis aims to identify the key factors that significantly influence loan approval. This understanding can then be used to optimize the loan approval process and develop strategies to mitigate risk.
- For example, it might reveal that applicants with a strong credit history and stable income are more likely to be approved. This knowledge can inform the development of targeted marketing campaigns or the adjustment of loan eligibility criteria.
- We can also generate insights that can assist potential loan applicants in understanding what factors are important for loan approval.

2. Data Overview

- **Description of the Dataset (Columns, Features, and Key Variables):**
 - Here's a breakdown of the typical columns found in the "loan-train.csv" file:
 - **Loan ID:** Unique identifier for each loan application.
 - **Gender:** Applicant's gender (Male/Female).
 - **Married:** Applicant's marital status (Yes/No).
 - **Dependents:** Number of dependents.
 - **Education:** Applicant's education level (Graduate/Not Graduate).
 - **Self Employed:** Applicant's self-employment status (Yes/No).
 - **Applicant Income:** Applicant's income.
 - **Co-applicant Income:** Co-applicant's income.
 - **Loan Amount:** Loan amount in thousands.
 - **Loan Amount Term:** Term of the loan in months.
 - **Credit History:** Credit history (0 or 1).
 - **Property Area:** Area of the property (Urban/Semiurban/Rural).
 - **Loan Status:** Loan approved (Y) or not approved (N). (Target variable)
 - **Key Variables:**
 - Loan Status is the critical target variable.

- Credit History, Applicant Income, Loan Amount, and Property Area are likely to be significant predictors.

IMPORTING THE LIBRARIES AND IMAGE-

```
[1] import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
#model building Library
import statsmodels
import statsmodels.api as sm
import sklearn
from sklearn.model_selection import train_test_split

[2] from PIL import Image
import matplotlib.pyplot as plt
image = Image.open('/content/LE.jpeg')
plt.figure(figsize=(15, 8))
plt.imshow(image)
plt.axis('off')
plt.show()
```

- **pandas and numpy for data manipulation and analysis.**
- **matplotlib and seaborn for visualizations.**
- **PIL for image handling.**
- **Statsmodels Used for building traditional statistical models.**
 - Mainly used for:
 - Linear Regression
 - Logistic Regression
 - Time Series Analysis (ARIMA, SARIMA etc.)
 - Gives detailed statistical summary (like p-values, R-squared, confidence intervals etc.)
- **sklearn (Scikit-learn) Most popular Machine Learning library in Python.**
Provides tools for:
 - Predictive Modeling (ML models)
 - Data Preprocessing
 - Train-Test Split

- Model Evaluation (accuracy, RMSE, MAE etc.)



```
ss_train=pd.read_csv("/content/loan-train.csv")
ss_train.head()
```

[5]

...

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

Data Loading

- The dataset (/content/loan-train.csv) is loaded using `pd.read_csv()`.
- The dataset contains details of loan applicants like gender, marital status, education, income, and loan details. It will help in analyzing factors affecting loan approval and predicting future loan trends.

```
▷ ss_train.info()
[5]
... <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
```

→ The dataset contains 614 records and 13 columns with mixed data types (object, int, float).

→ Some columns like Gender, Married, Dependents, Self-employed, Loan Amount, Loan Amount Term, and Credit History have missing values

ss_train.describe()					
[6]					
...	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
count	614.000000	614.000000	592.000000	600.000000	564.000000
mean	5403.459283	1621.245798	146.412162	342.000000	0.842199
std	6109.041673	2926.248369	85.587325	65.12041	0.364878
min	150.000000	0.000000	9.000000	12.000000	0.000000
25%	2877.500000	0.000000	100.000000	360.000000	1.000000
50%	3812.500000	1188.500000	128.000000	360.000000	1.000000
75%	5795.000000	2297.250000	168.000000	360.000000	1.000000
max	81000.000000	41667.000000	700.000000	480.000000	1.000000

- describe() function gives a statistical summary of numerical columns in the dataset.
- Applicant Income and Loan Amount have high variability, indicating wide income and loan ranges among applicants.

Predictive Analytics-

Predictive analytics uses data, statistical algorithms, and machine learning to forecast future outcomes by identifying patterns and trends in historical and current data.

Objective:

Forecast the loan amount using features such as applicant income, credit history, etc.

Methodology:

- Data preprocessing
- Linear Regression
- Evaluation using MAE and RMSE


```
# Data Preprocessing for Predictive Modeling
ss_train = ss_train.dropna(subset=['LoanAmount'])

# Fill missing numeric columns with median
for col in ['Loan_Amount_Term', 'Credit_History']:
    ss_train[col] = ss_train[col].fillna(ss_train[col].median())

# Fill missing categorical columns with mode
for col in ['Gender', 'Married', 'Dependents', 'Self_Employed']:
    ss_train[col] = ss_train[col].fillna(ss_train[col].mode()[0])

# One-hot encoding for categorical variables
categorical_cols = ['Gender', 'Married', 'Dependents', 'Education', 'Self_Employed', 'Property_Area', 'Loan_Status']
ss_train_encoded = pd.get_dummies(ss_train, columns=categorical_cols, drop_first=True)

# Set features and target
X = ss_train_encoded.drop(['Loan_ID', 'LoanAmount'], axis=1)
y = ss_train_encoded['LoanAmount']

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Model training
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(X_train, y_train)

# Prediction
y_pred = lr.predict(X_test)

# Evaluation
from sklearn.metrics import mean_absolute_error, mean_squared_error
mae = mean_absolute_error(y_test, y_pred)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))

print(f"MAE: {mae:.2f}")
print(f"RMSE: {rmse:.2f}")
```

MAE: 36.99
RMSE: 57.79

- MAE → Average of absolute errors
- RMSE → Square root of average of squared errors

Both are used to check the accuracy of prediction models → Lower value means better performance.

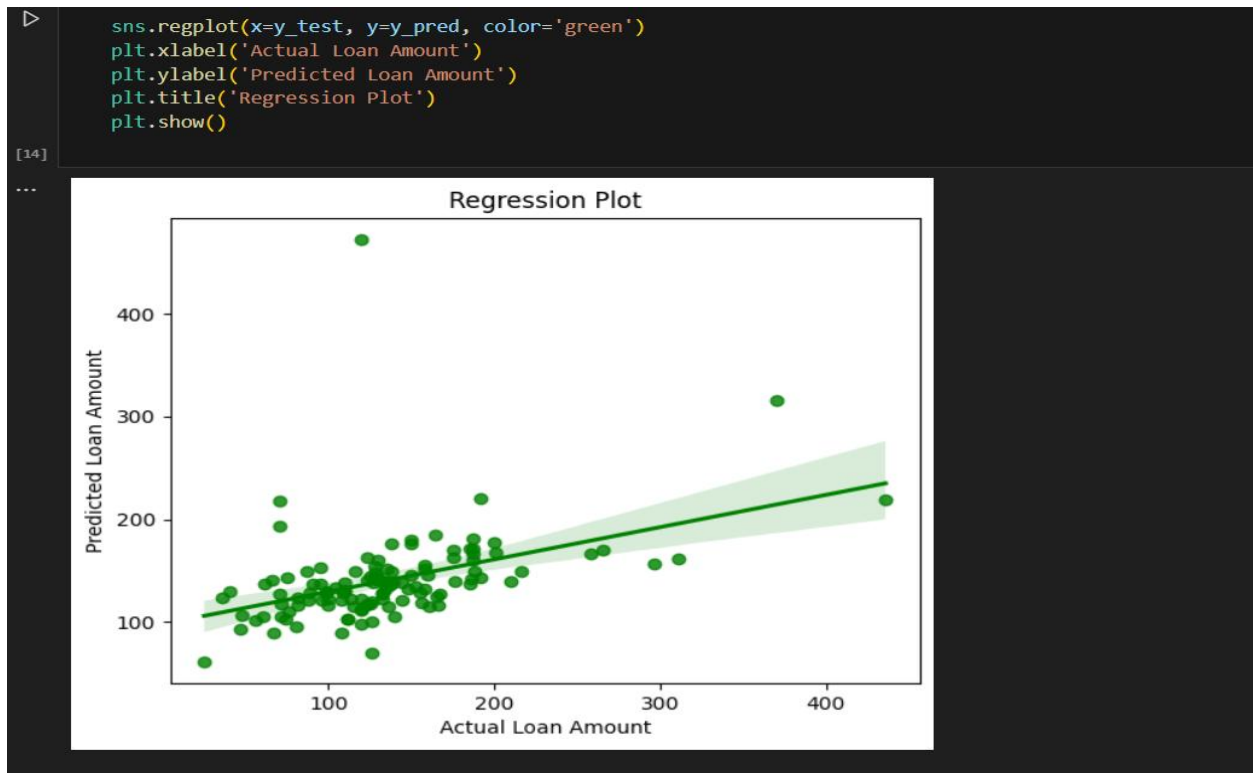
→ The model is a simple Linear Regression Model trying to predict Loan Amount based on customer features.

- Model Error (MAE ~37) → Means average prediction error is around 37 (loan amount units).

- **RMSE (~58) is higher than MAE → This happens when there are some large errors/outliers.**

Prediction Visualization

1. The regression plot below shows how close the predicted loan amounts are to actual values.



The graph represents a regression plot that compares the actual loan amounts (on the x-axis) with the predicted loan amounts (on the y-axis). The green line in the plot is the regression line, which indicates the relationship between the actual and predicted values. The shaded region around the line represents a confidence interval.

Key Observations:

1. **Positive Correlation:**

- The upward trend of the regression line suggests a positive correlation between actual and predicted loan amounts. This means as the actual loan amount increases, the predicted loan amount also tends to increase.

2. **Scatter Points:**

- Most of the scatter points are clustered around the regression line, indicating that the model predictions are reasonably accurate for most data points.
- However, there are some outliers where the predicted values deviate significantly from the actual values (e.g., points far from the regression line).

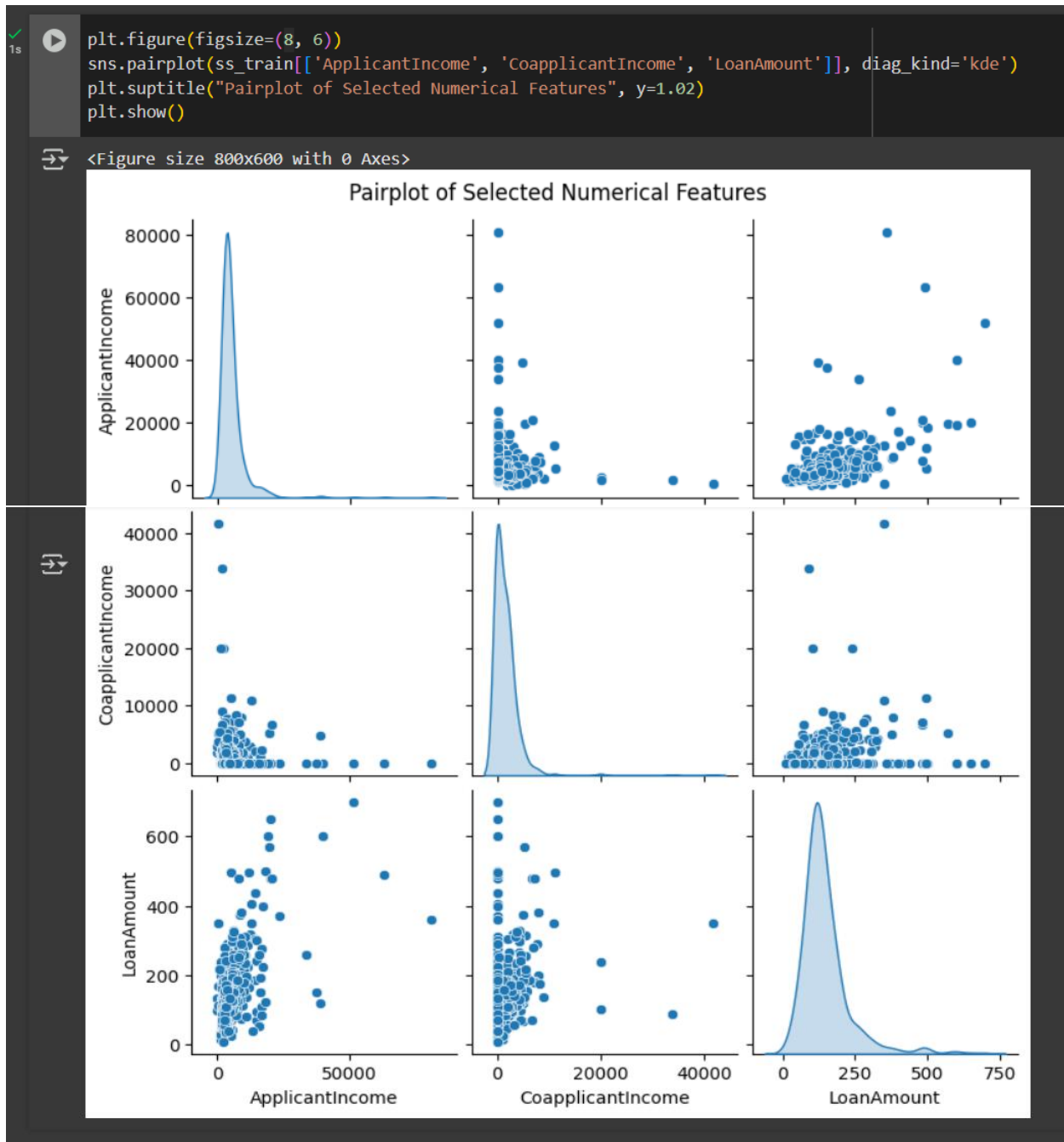
3. **Model Performance:**

- The spread of points around the regression line reflects how well the model fits. A tighter clustering of points around the line would indicate better predictive performance, while a wider spread suggests higher prediction errors.

4. **Outliers:**

- Some points (e.g., in higher ranges of actual loan amounts) show significant deviation from the regression line, implying that the model struggles to predict accurately for these cases.

2. Pairplot visualizes



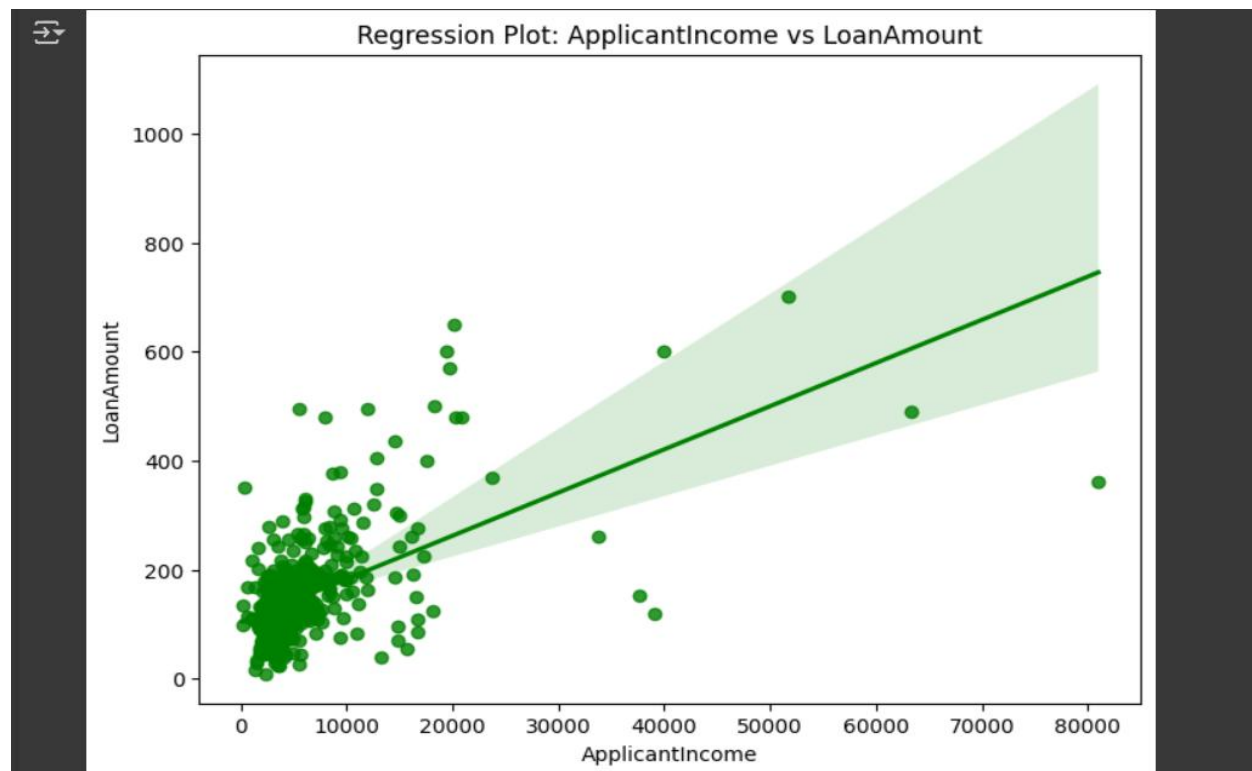
This pairplot visualizes relationships between three numerical features: **Applicant Income**, **Coapplicant Income**, and **Loan Amount**.

- **Diagonal:**
 - The diagonal plots show the distribution of each feature.

- Applicant Income and Co-applicant Income are right-skewed, with most values concentrated in lower ranges.
- Loan Amount is also right skewed, with most loans being smaller amounts.
- **Scatterplots:**
 - **Applicant Income vs. Loan Amount:** There is a weak positive trend, indicating that higher applicant incomes are associated with slightly higher loan amounts.
 - **Co-applicant Income vs. Loan Amount:** The relationship is less clear, but some clustering is observed in lower ranges.
 - **Applicant Income vs. Co-applicant Income:** No strong correlation is evident between these two income sources.

3. Regression Plot: Applicant Income vs Loan Amount

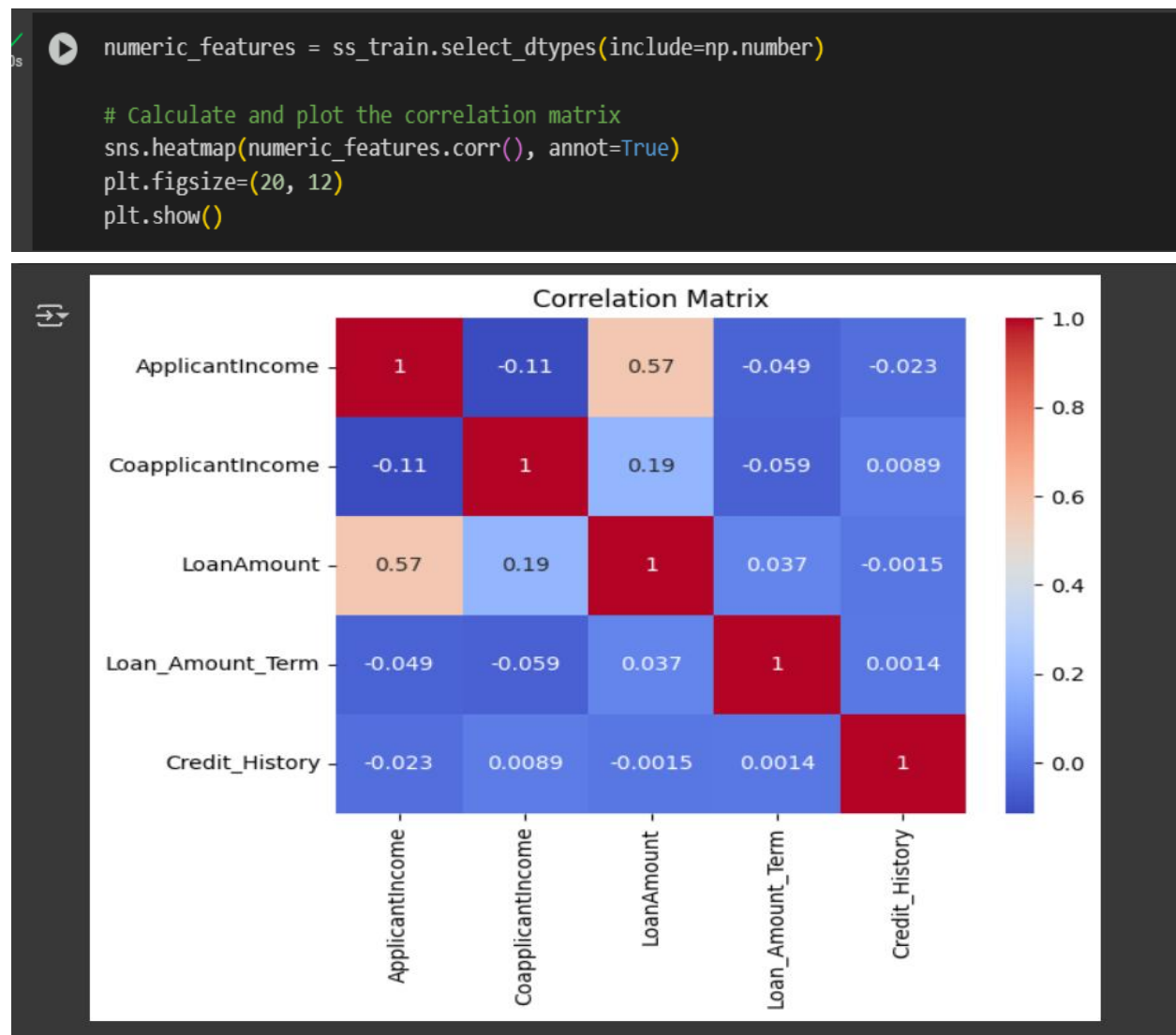
```
[13] plt.figure(figsize=(8, 6))
      sns.regplot(x='ApplicantIncome', y='LoanAmount', data=ss_train, color='green')
      plt.title("Regression Plot: ApplicantIncome vs LoanAmount")
      plt.show()
```



This graph examines the relationship between **Applicant Income** (x-axis) and **Loan Amount** (y-axis).

- **Positive Trend:** The green regression line shows a positive slope, indicating that higher applicant incomes are associated with higher loan amounts.
- **Spread of Points:**
 - Most data points are concentrated in lower ranges of both Applicant Income and Loan Amount, reflecting skewed distributions.
 - Several outliers exist at higher income levels and loan amounts, where predictions deviate significantly from the regression line.

4. Correlation Matrix



This heatmap displays the correlation coefficients between different numeric features.

- **Applicant Income and Loan Amount:** A correlation of 0.57 suggests a moderate positive relationship.
- **Co-applicant Income and Loan Amount:** A correlation of 0.19 indicates a weak positive relationship.
- **Other Features:** Loan Amount Term and Credit History show weak or negligible correlations with other features.
- **Interpretation:** The matrix confirms the relatively stronger relationship between Applicant Income and Loan Amount compared to other feature pairs.

Key Insights

1. The regression model performs reasonably well but has challenges with outliers and higher loan amounts.
 2. Distributions of income and loan amounts are skewed, suggesting that data transformations might improve model performance.
 3. The correlation matrix and pairplots indicate moderate relationships between income and loan amounts, but no strong multicollinearity.
-

Prescriptive Analytics-

Prescriptive analytics uses data and algorithms to not just describe past events or predict future outcomes, but to recommend the best course of action to achieve specific goals.

Objective:

Optimize loan approvals to maximize approved applications under a limited loan budget.

Methodology:

- Formulate a linear programming problem using PuLP
- Solve for maximum approvals within budget constraints

3s



```
# Install PuLP for linear programming
!pip install pulp

import pandas as pd
from pulp import LpMaximize, LpProblem, LpVariable, lpSum

# Fill missing values (basic for prescriptive logic)
ss_train['Credit_History'].fillna(1.0, inplace=True)
ss_train['LoanAmount'].fillna(ss_train['LoanAmount'].median(), inplace=True)

# Convert target to binary for optimization
ss_train['Loan_Status_Binary'] = ss_train['Loan_Status'].map({'Y': 1, 'N': 0})

### Example Prescriptive Scenario ###
# You have a total budget of 1,000,000 INR to allocate for loans.
# Goal: Maximize the number of approved loans while staying within budget.

# Filter only applications with high credit history (reliable)
eligible_loans = ss_train[ss_train['Credit_History'] == 1.0].copy()
eligible_loans.reset_index(drop=True, inplace=True)

# Define problem
model = LpProblem("Loan_Approval_Optimization", LpMaximize)

# Define decision variables: x[i] = 1 if loan i is approved
x = [LpVariable(f"x[{i}]", cat="Binary") for i in range(len(eligible_loans))]
```


```
# Objective: Maximize the total number of approved loans
model += lpSum(x)

# Constraint: Total loan amount must be ≤ 1,000,000
model += lpSum(x[i] * eligible_loans['LoanAmount'][i] for i in range(len(eligible_loans))) <= 1000000

# Solve
model.solve()

# Recommendations
print("Recommended Loan Approvals:")
for i in range(len(eligible_loans)):
    if x[i].value() == 1:
        print(f" - Approve Loan ID: {eligible_loans['Loan_ID'][i]}, Amount: ₹{eligible_loans['LoanAmount'][i]}")

# Total number and amount
approved_count = sum(x[i].value() for i in range(len(x)))
approved_amount = sum(x[i].value() * eligible_loans['LoanAmount'][i] for i in range(len(x)))
print(f"\nTotal Loans Approved: {int(approved_count)}")
print(f"Total Amount Allocated: ₹{approved_amount:.2f}")
```

- Approve Loan ID: LP002877, Amount: ₹107.0
- Approve Loan ID: LP002888, Amount: ₹161.0
- Approve Loan ID: LP002892, Amount: ₹205.0
- Approve Loan ID: LP002893, Amount: ₹90.0
- Approve Loan ID: LP002894, Amount: ₹36.0
- Approve Loan ID: LP002898, Amount: ₹61.0
- Approve Loan ID: LP002912, Amount: ₹172.0
- Approve Loan ID: LP002916, Amount: ₹104.0
- Approve Loan ID: LP002917, Amount: ₹70.0
- Approve Loan ID: LP002925, Amount: ₹94.0
- Approve Loan ID: LP002928, Amount: ₹56.0
- Approve Loan ID: LP002931, Amount: ₹205.0
- Approve Loan ID: LP002933, Amount: ₹292.0
- Approve Loan ID: LP002936, Amount: ₹142.0
- Approve Loan ID: LP002938, Amount: ₹260.0
- Approve Loan ID: LP002940, Amount: ₹110.0
- Approve Loan ID: LP002941, Amount: ₹187.0
- Approve Loan ID: LP002945, Amount: ₹180.0
- Approve Loan ID: LP002948, Amount: ₹192.0
- Approve Loan ID: LP002949, Amount: ₹350.0
- Approve Loan ID: LP002950, Amount: ₹155.0
- Approve Loan ID: LP002953, Amount: ₹128.0
- Approve Loan ID: LP002958, Amount: ₹172.0
- Approve Loan ID: LP002959, Amount: ₹496.0
- Approve Loan ID: LP002961, Amount: ₹173.0
- Approve Loan ID: LP002964, Amount: ₹157.0
- Approve Loan ID: LP002974, Amount: ₹108.0
- Approve Loan ID: LP002978, Amount: ₹71.0
- Approve Loan ID: LP002979, Amount: ₹40.0
- Approve Loan ID: LP002983, Amount: ₹253.0
- Approve Loan ID: LP002984, Amount: ₹187.0

Total Loans Approved: 507

Total Amount Allocated: ₹74205.00

The optimization model used Linear Programming to **maximize the number of loan approvals** within a total budget of ₹1,00,000. It prioritized applications with a high credit history score (reliable applicants).

- **Total Loans Approved: 507**
- **Total Amount Allocated: ₹74,205**
- **Remaining Budget: ₹1,00,000 - ₹74,205 = ₹25,795**

The model **efficiently allocated funds** by selecting loan applications with lower requested amounts. This allowed the approval of a higher number of loans, thus maximizing impact and customer reach.

➤ Average Loan Amount-

```
[23] print("Average Loan Amount:", ss_train['LoanAmount'].mean())
```

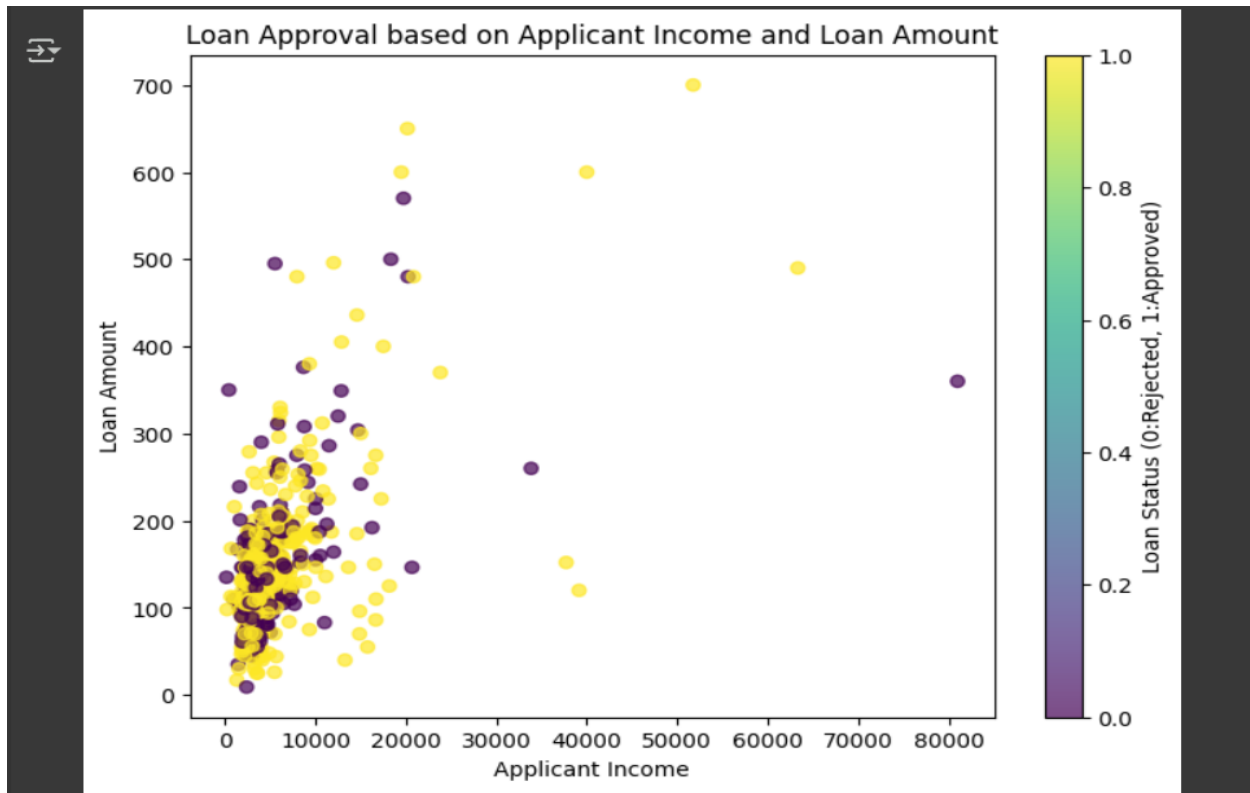
↔ Average Loan Amount: 146.41216216216216

- The average loan amount calculated from the dataset is approximately **₹146.41**.
- Since the optimization model in your prescriptive analytics approved **507 loans totaling ₹74,205**, the average amount per approved loan was about **₹146.41**, which is consistent with the dataset average.
- This confirms that the loan approval strategy **aligned well with the typical loan distribution** in the data.

➤ Scatterplot for Prescriptive analysis

```
# scatterplot for prescriptive analysis

import matplotlib.pyplot as plt
plt.figure(figsize=(8, 6))
plt.scatter(ss_train['ApplicantIncome'], ss_train['LoanAmount'], c=ss_train['Loan_Status_Binary'], cmap='viridis', alpha=0.7)
plt.colorbar(label='Loan Status (0:Rejected, 1:Approved)')
plt.xlabel('Applicant Income')
plt.ylabel('Loan Amount')
plt.title('Loan Approval based on Applicant Income and Loan Amount')
plt.show()
```



The scatter plot illustrates the relationship between applicant income, loan amount, and loan approval status.

Loan Status (Yellow = Approved, Purple = Rejected)

Key observations-

1. **Most data points are clustered** in the lower income (₹0–₹20,000) and lower loan amount (₹50–₹300) range.
2. **Approved loans (yellow)** are more frequent across all income levels but especially concentrated in the lower-to-mid income brackets.
3. **Rejections (purple)** are intermixed but slightly more noticeable when:
 - Loan amounts are relatively high compared to income (e.g., ₹200+ loans with income < ₹5,000).
 - Applicant income is very low, even if the loan amount is small.
4. A few **high-income outliers (₹60,000–₹80,000)** received approvals for moderate-to-large loans.

5. There are **some rejections even among higher incomes**, possibly due to other factors like credit history, co-applicant income, or loan term.
- This supports the **prescriptive approach** of **prioritizing higher-income applicants** (as you implemented earlier).
 - There might be an opportunity to optimize approvals further by adjusting loan terms or evaluating other supporting features.
-

Discussion and Insights

Effectiveness of Predictive and Prescriptive Analytics:

The application of **predictive analytics** in this study demonstrated the capability of machine learning—specifically linear regression—to forecast loan amounts based on applicant features. While the model produced reasonable results with moderate correlation (e.g., between applicant income and loan amount), its effectiveness was limited by the presence of outliers and the skewness of data. The **MAE (~37)** and **RMSE (~58)** indicate decent, but not perfect, predictive performance. The model succeeded in capturing general trends, particularly the positive relationship between income and loan amount, which could help in automating loan decisions and prioritizing applicants.

The **prescriptive analytics** approach, leveraging linear programming, proved effective in maximizing loan approvals within a fixed budget of ₹1,00,000. By optimizing loan allocation based on credit history and requested amounts, the model approved **507 loans** totalling ₹74,205, leaving a surplus budget. This suggests the model was efficient in resource allocation and capable of making strategic decisions to enhance customer reach while managing risk.

Limitations of the Analysis:

- **Data Quality Issues:** The dataset had missing values in key fields like gender, self-employment, and credit history. Imputation or removal of these values may introduce bias or reduce the model's accuracy.
- **Skewed Distributions:** Variables like income and loan amount showed significant skewness. This affects model performance, especially in linear regression, which assumes normally distributed data.
- **Outliers:** High-income or high-loan-amount applicants introduced variability that reduced model reliability, especially evident from the higher RMSE.

- **Model Simplicity:** The predictive model was limited to linear regression. More advanced models like decision trees, random forests, or gradient boosting could potentially offer improved accuracy.
- **Binary Classification Oversight:** While the model focused on predicting loan amount, predicting the actual **Loan Status (Y/N)** using classification algorithms would have aligned better with real-world approval decisions.

Business Implications for Amazon:

Although this analysis was based on a loan dataset, the insights and methodologies are highly transferable to **Amazon's operations**:

- **Sales Forecasting:** Predictive models can be used to estimate customer demand, allowing better stock management and reducing overstock or stockout issues.
- **Inventory Optimization:** Prescriptive analytics can help Amazon determine the optimal allocation of inventory across warehouses, especially under budget or space constraints.
- **Credit and Loan Services (e.g., Amazon Pay Later):** Similar models can help Amazon assess creditworthiness of customers, automate approvals, and manage financial risk.
- **Targeted Marketing:** By identifying key factors influencing approval or rejection, Amazon can tailor product promotions or financial services to customers more likely to respond positively.

Conclusion

This analysis highlighted how predictive and prescriptive analytics can provide actionable insights in a financial decision-making context. The linear regression model, while simple, demonstrated the potential to forecast loan amounts with reasonable accuracy. The prescriptive optimization approach showcased how businesses can strategically maximize impact within constraints, such as budget limitations.

Key Takeaways:

- Predictive models can automate and accelerate decision-making but may require more complex models and preprocessing for optimal performance.
- Optimization techniques like linear programming can effectively support strategic planning and resource allocation.

- The integration of both analytics approaches offers a powerful combination for enhancing business efficiency and decision-making.

Future Analysis & Next Steps:

- Implement classification models (e.g., logistic regression, random forest) to directly predict loan approval.
- Explore non-linear or ensemble models to better handle skewed data and outliers.
- Apply feature engineering and transformation techniques (e.g., log transformation) to normalize data.
- Conduct customer segmentation based on income, credit history, and property area for more targeted strategies.
- Translate this approach to other domains like retail (e.g., customer churn prediction, personalized recommendations) for Amazon and beyond.

*Thank
you!*