Loan-Approval-Prediction

Problem Definition Loan approval prediction poses a unique challenge due to the multifaceted nature of the factors that influence an applicant's ability to repay a loan. Traditionally, loan officers have relied on a set of criteria—such as credit score, income level, age, and past loan repayment history—when determining the risk associated with lending money to an individual. However, this process often lacks objectivity and is subject to human bias, where subjective interpretations of the data can lead to unfair or inconsistent decisions. Additionally, the sheer volume of loan applications in large financial institutions makes it increasingly difficult for human evaluators to maintain accuracy and efficiency.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report

df = pd.read_csv("loan_approval_dataset.csv")
df
```

```
      loan_id   no_of_dependents      education   self_employed
income_annum  \
0           1                  2       Graduate              No
9600000
1           2                  0   Not Graduate             Yes
4100000
2           3                  3       Graduate              No
9100000
3           4                  3       Graduate              No
8200000
4           5                  5   Not Graduate             Yes
9800000
...       ...                ...            ...             ...
...
4264     4265                  5       Graduate             Yes
1000000
4265     4266                  0   Not Graduate             Yes
3300000
4266     4267                  2   Not Graduate              No
6500000
4267     4268                  1   Not Graduate              No
4100000
```

```
4268     4269                  1       Graduate             No
9200000

       loan_amount   loan_term   cibil_score
residential_assets_value  \
0         29900000          12           778
2400000
1         12200000           8           417
2700000
2         29700000          20           506
7100000
3         30700000           8           467
18200000
4         24200000          20           382
12400000
...            ...         ...           ...                        ..
.
4264       2300000          12           317
2800000
4265      11300000          20           559
4200000
4266      23900000          18           457
1200000
4267      12800000           8           780
8200000
4268      29700000          10           607
17800000

       commercial_assets_value   luxury_assets_value
bank_asset_value  \
0                     17600000              22700000
8000000
1                      2200000               8800000
3300000
2                      4500000              33300000
12800000
3                      3300000              23300000
7900000
4                      8200000              29400000
5000000
...                        ...                   ...                  ..
.
4264                    500000               3300000
800000
4265                   2900000              11000000
1900000
4266                  12400000              18100000
7300000
4267                    700000              14100000
```

```
5800000
4268                  11800000              35700000
12000000

      loan_status
0        Approved
1        Rejected
2        Rejected
3        Rejected
4        Rejected
...           ...
4264     Rejected
4265     Approved
4266     Rejected
4267     Approved
4268     Approved

[4269 rows x 13 columns]
```

```
# Display basic information
print("Dataset Overview:")
print(df.head())
print(df.info())
print(df.describe())
```

```
Dataset Overview:
   loan_id   no_of_dependents      education  self_employed
income_annum  \
0        1                  2       Graduate             No
9600000
1        2                  0   Not Graduate            Yes
4100000
2        3                  3       Graduate             No
9100000
3        4                  3       Graduate             No
8200000
4        5                  5   Not Graduate            Yes
9800000

    loan_amount   loan_term   cibil_score
residential_assets_value  \
0      29900000          12           778                  2400000

1      12200000           8           417                  2700000

2      29700000          20           506                  7100000

3      30700000           8           467                 18200000

4      24200000          20           382                 12400000
```

```
    commercial_assets_value   luxury_assets_value
bank_asset_value  \
0                  17600000              22700000           8000000

1                   2200000               8800000           3300000

2                   4500000              33300000          12800000

3                   3300000              23300000           7900000

4                   8200000              29400000           5000000


   loan_status
0     Approved
1     Rejected
2     Rejected
3     Rejected
4     Rejected
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4269 entries, 0 to 4268
Data columns (total 13 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   loan_id                    4269 non-null   int64
 1    no_of_dependents          4269 non-null   int64
 2    education                 4269 non-null   object
 3    self_employed             4269 non-null   object
 4    income_annum              4269 non-null   int64
 5    loan_amount               4269 non-null   int64
 6    loan_term                 4269 non-null   int64
 7    cibil_score               4269 non-null   int64
 8    residential_assets_value  4269 non-null   int64
 9    commercial_assets_value   4269 non-null   int64
 10   luxury_assets_value       4269 non-null   int64
 11   bank_asset_value          4269 non-null   int64
 12   loan_status               4269 non-null   object
dtypes: int64(10), object(3)
memory usage: 433.7+ KB
None
           loan_id   no_of_dependents   income_annum   loan_amount  \
count  4269.000000        4269.000000   4.269000e+03  4.269000e+03
mean   2135.000000           2.498712   5.059124e+06  1.513345e+07
std    1232.498479           1.695910   2.806840e+06  9.043363e+06
min       1.000000           0.000000   2.000000e+05  3.000000e+05
25%    1068.000000           1.000000   2.700000e+06  7.700000e+06
50%    2135.000000           3.000000   5.100000e+06  1.450000e+07
75%    3202.000000           4.000000   7.500000e+06  2.150000e+07
```

```
max    4269.000000          5.000000   9.900000e+06  3.950000e+07

         loan_term   cibil_score   residential_assets_value  \
count  4269.000000   4269.000000               4.269000e+03
mean     10.900445    599.936051               7.472617e+06
std       5.709187    172.430401               6.503637e+06
min       2.000000    300.000000              -1.000000e+05
25%       6.000000    453.000000               2.200000e+06
50%      10.000000    600.000000               5.600000e+06
75%      16.000000    748.000000               1.130000e+07
max      20.000000    900.000000               2.910000e+07

        commercial_assets_value   luxury_assets_value
bank_asset_value
count              4.269000e+03          4.269000e+03
4.269000e+03
mean               4.973155e+06          1.512631e+07
4.976692e+06
std                4.388966e+06          9.103754e+06
3.250185e+06
min                0.000000e+00          3.000000e+05
0.000000e+00
25%                1.300000e+06          7.500000e+06
2.300000e+06
50%                3.700000e+06          1.460000e+07
4.600000e+06
75%                7.600000e+06          2.170000e+07
7.100000e+06
max                1.940000e+07          3.920000e+07
1.470000e+07

# Cleaning column names
df.columns = df.columns.str.strip().str.lower()

# Verify column names
print("Columns in dataset:", df.columns)

Columns in dataset: Index(['loan_id', 'no_of_dependents', 'education',
'self_employed',
       'income_annum', 'loan_amount', 'loan_term', 'cibil_score',
       'residential_assets_value', 'commercial_assets_value',
       'luxury_assets_value', 'bank_asset_value', 'loan_status'],
      dtype='object')

# Identify the target column
possible_target_columns = ['loan_status', 'loan_approval_status']
target_column = None
for col in possible_target_columns:
    if col in df.columns:
        target_column = col
```

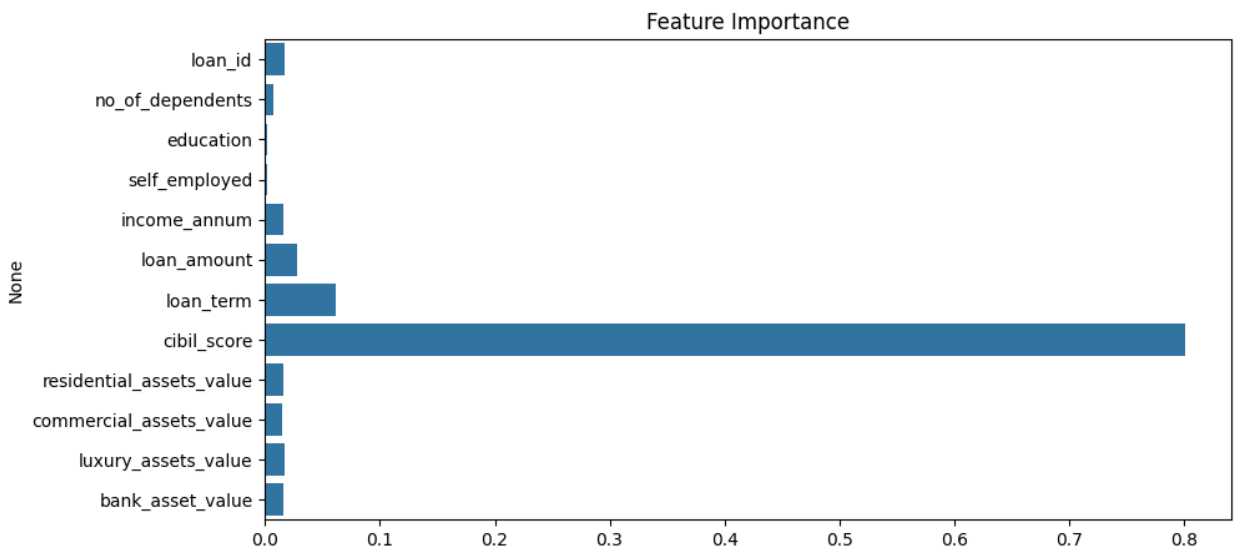
```python
        break

if target_column is None:
    raise KeyError("Target column not found in dataset! Available 
columns: " + str(df.columns))

# Handling missing values
imputer = SimpleImputer(strategy='most_frequent')
df[df.columns] = imputer.fit_transform(df)

# Encoding categorical variables
label_encoders = {}
for col in df.select_dtypes(include=['object']).columns:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    label_encoders[col] = le

# Feature-target split
X = df.drop(columns=[target_column])
y = df[target_column]

# Splitting dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, 
test_size=0.2, random_state=42)

# Feature Scaling
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Model Training
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
```

```
RandomForestClassifier(random_state=42)
```

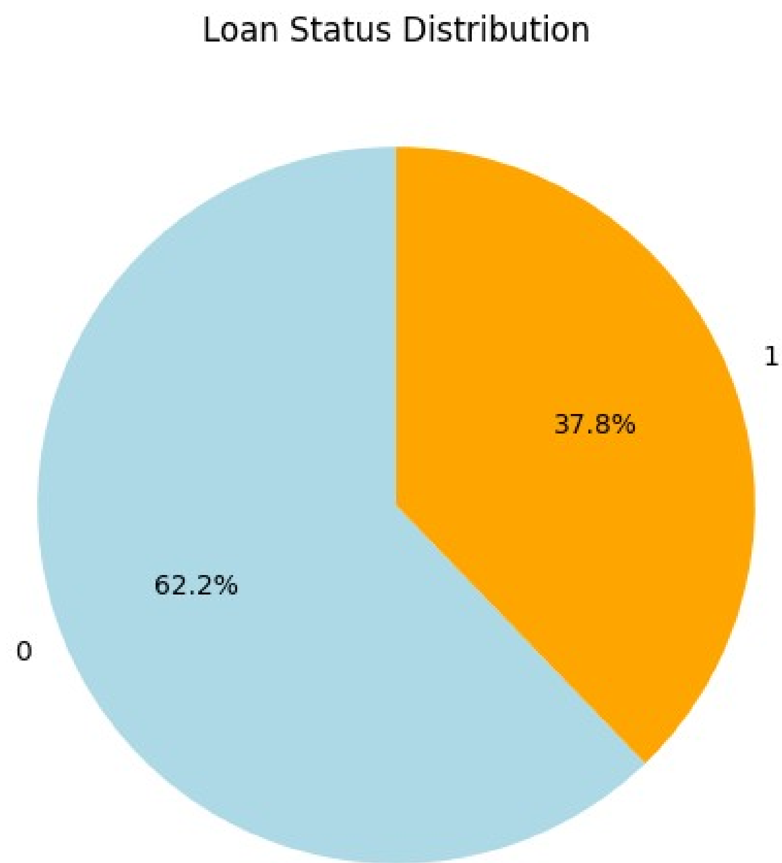
```python
# Predictions
y_pred = model.predict(X_test)

# Evaluation
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)

print(f"Accuracy: {accuracy:.2f}")
print("Confusion Matrix:")
print(conf_matrix)
print("Classification Report:")
print(classification_rep)
```

```
Accuracy: 0.98
Confusion Matrix:
```

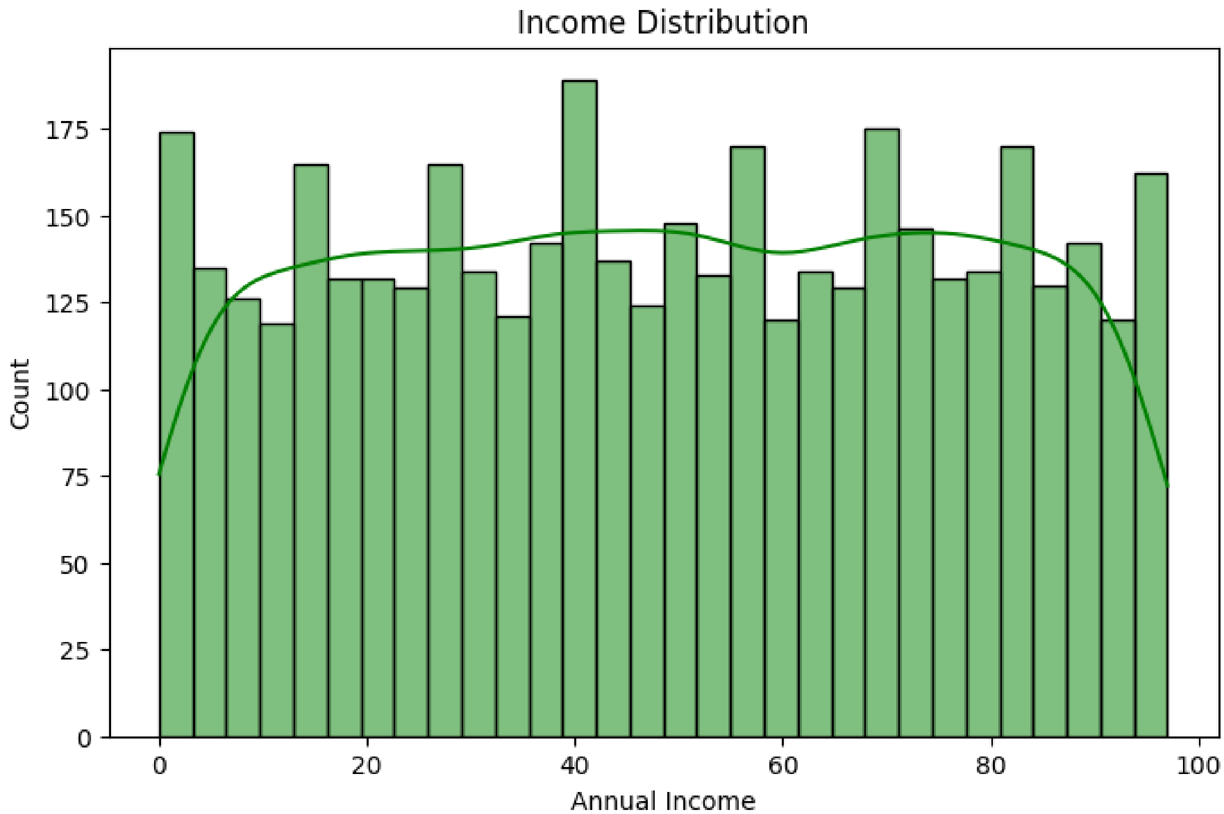
```
[[528   8]
 [  9 309]]
Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.99      0.98       536
           1       0.97      0.97      0.97       318

    accuracy                           0.98       854
   macro avg       0.98      0.98      0.98       854
weighted avg       0.98      0.98      0.98       854


# Feature Importance
feature_importances = model.feature_importances_
feature_names = X.columns

plt.figure(figsize=(10, 5))
sns.barplot(x=feature_importances, y=feature_names)
plt.title("Feature Importance")
plt.show()
```



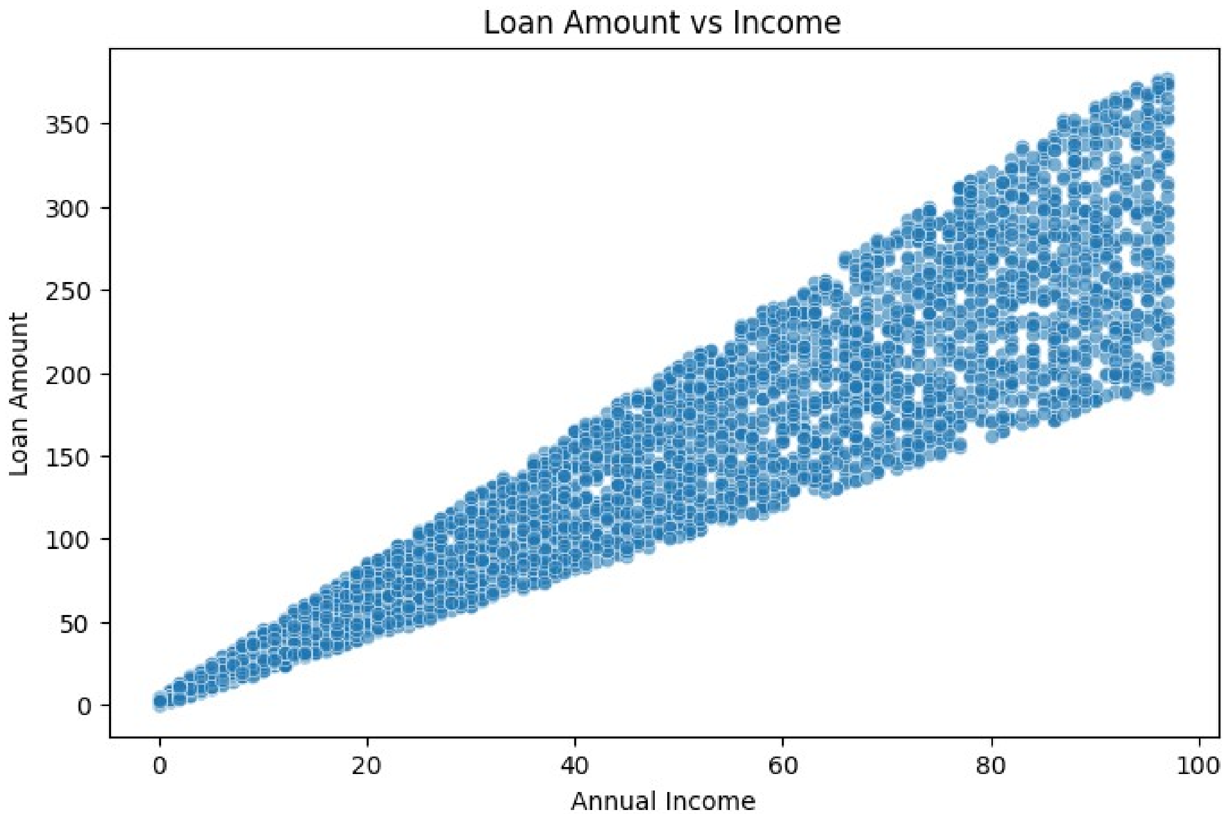

```
# Loan Status Distribution
plt.figure(figsize=(6, 6))
df[target_column].value_counts().plot.pie(autopct='%1.1f%%',
colors=['lightblue', 'orange'], startangle=90)
plt.title("Loan Status Distribution")
plt.ylabel('')
plt.show()
```



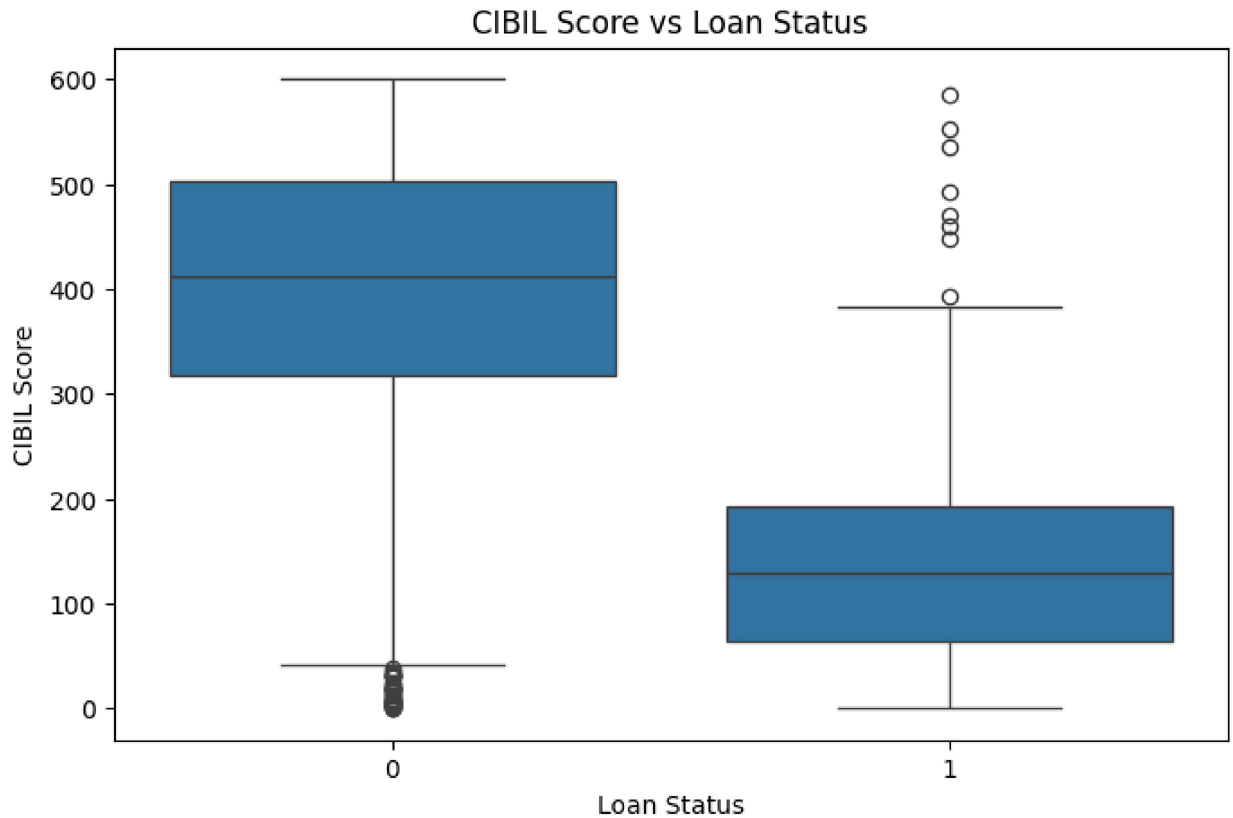

```
# Income Distribution
plt.figure(figsize=(8, 5))
sns.histplot(df['income_annum'], bins=30, kde=True, color='green')
plt.title("Income Distribution")
plt.xlabel("Annual Income")
plt.show()
```




```
# Loan Amount vs Income
plt.figure(figsize=(8, 5))
sns.scatterplot(x=df['income_annum'], y=df['loan_amount'], alpha=0.6)
plt.title("Loan Amount vs Income")
plt.xlabel("Annual Income")
plt.ylabel("Loan Amount")
plt.show()
```




```
# CIBIL Score vs Loan Status
plt.figure(figsize=(8, 5))
sns.boxplot(x=df[target_column], y=df['cibil_score'])
plt.title("CIBIL Score vs Loan Status")
plt.xlabel("Loan Status")
plt.ylabel("CIBIL Score")
plt.show()
```




```
import os
os.makedirs('/mnt/data/', exist_ok=True)

joblib.dump(model, '/mnt/data/loan_approval_model.pkl')
joblib.dump(scaler, '/mnt/data/loan_scaler.pkl')
joblib.dump(label_encoders, '/mnt/data/loan_label_encoders.pkl')

['/mnt/data/loan_label_encoders.pkl']
```

Insights and Summary

1. Feature Importance Analysis

The Random Forest feature importance chart indicates which variables contribute most to loan approval predictions. Key factors affecting loan approval include CIBIL Score, Income, Loan Amount, and Bank Asset Value.

2. Loan Status Distribution

The pie chart shows the proportion of approved vs. rejected loans. If approval rates are significantly lower, lenders may have strict criteria, or applicant profiles may not be strong enough.

3. Income Distribution

The histogram of annual income suggests the general income distribution of loan applicants. If the distribution is skewed, the dataset might have more low-income or high-income applicants.

4. Loan Amount vs Income

The scatter plot indicates how loan amounts relate to annual income. A clear trend would suggest a proportional relationship, whereas a scattered distribution might indicate varied approval criteria.

5. CIBIL Score vs Loan Status

The box plot reveals the distribution of CIBIL scores for approved vs. rejected loans. If approved loans consistently have high CIBIL scores, it confirms that creditworthiness significantly impacts approval decisions.