A Case Study Report on

Loan Approval Prediction

by

Vansh Jeetendra Jain, 20102156

Ghanshyam Kachhia, 20102026

in the subject

Applied Data Science

Department of Computer Engineering

A. P. SHAH INSTITUTE OF TECHNOLOGY, THANE

G.B. Road, Kasarvadavli, Thane(W)

University of Mumbai

2023-2024

1. Problem Statement:

Large volumes of loan applications are manually reviewed by financial organisations, which can cause irregularities and long processing periods. The goal of this project is to create a machine learning model that uses applicant data to forecast loan approvals. The model will assess variables such as loan amount, credit history, income, and demographics in order to calculate the probability of a successful repayment. This will facilitate data-driven lending decisions, increase efficiency, and streamline the loan approval procedure. The model's ability to identify creditworthy borrowers and predict loan approvals accurately will be used to gauge the project's performance.

2. Description of dataset:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Loan_ID	Gender	Married	Dependen	Education	Self_Empl	ApplicantIn	Coapplicar	LoanAmou	Loan_Amo	Credit_His	Property_A	Loan_Status	
2	LP001002	Male	No	0	Graduate	No	5849	0		360	1	Urban	Y	
3	LP001003	Male	Yes	1	Graduate	No	4583	1508	128	360	1	Rural	N	
4	LP001005	Male	Yes	0	Graduate	Yes	3000	0	66	360	1	Urban	Y	
5	LP001006	Male	Yes	0	Not Gradu	No	2583	2358	120	360	1	Urban	Y	
6	LP001008	Male	No	0	Graduate	No	6000	0	141	360	1	Urban	Y	
7	LP001011	Male	Yes	2	Graduate	Yes	5417	4196	267	360	1	Urban	Y	
8	LP001013	Male	Yes	0	Not Gradu	No	2333	1516	95	360	1	Urban	Y	
9	LP001014	Male	Yes	3+	Graduate	No	3036	2504	158	360	0	Semiurban	N	
10	LP001018	Male	Yes	2	Graduate	No	4006	1526	168	360	1	Urban	Y	
11	LP001020	Male	Yes	1	Graduate	No	12841	10968	349	360	1	Semiurban	N	
12	LP001024	Male	Yes	2	Graduate	No	3200	700	70	360	1	Urban	Y	
13	LP001027	Male	Yes	2	Graduate		2500	1840	109	360	1	Urban	Y	
14	LP001028	Male	Yes	2	Graduate	No	3073	8106	200	360	1	Urban	Y	
15	LP001029	Male	No	0	Graduate	No	1853	2840	114	360	1	Rural	N	
16	LP001030	Male	Yes	2	Graduate	No	1299	1086	17	120	1	Urban	Y	
17	LP001032	Male	No	0	Graduate	No	4950	0	125	360	1	Urban	Y	
18	LP001034	Male	No	1	Not Gradu	No	3596	0	100	240		Urban	Y	
19	LP001036	Female	No	0	Graduate	No	3510	0	76	360	0	Urban	N	
20	LP001038	Male	Yes	0	Not Gradu	No	4887	0	133	360	1	Rural	N	
21	LP001041	Male	Yes	0	Graduate		2600	3500	115		1	Urban	Y	
22	LP001043	Male	Yes	0	Not Gradu	No	7660	0	104	360	0	Urban	N	
23	LP001046	Male	Yes	1	Graduate	No	5955	5625	315	360	1	Urban	Y	
24	LP001047	Male	Yes	0	Not Gradu	No	2600	1911	116	360	0	Semiurban	N	
25	LP001050		Yes	2	Not Gradu	No	3365	1917	112	360	0	Rural	N	
26	LP001052	Male	Yes	1	Graduate		3717	2925	151	360		Semiurban	N	
27	LP001066	Male	Yes	0	Graduate	Yes	9560	0	191	360	1	Semiurban	Y	
28	LP001068	Male	Yes	0	Graduate	No	2799	2253	122	360	1	Semiurban	Y	
29	LP001073	Male	Yes	2	Not Gradu	No	4226	1040	110	360	1	Urban	Y	
30	LP001086	Male	No	0	Not Gradu	No	1442	0	35	360	1	Urban	N	

Fig. 1 Loan Application Dataset.

Size: The dataset comprises a single CSV file, typically with multiple(614) rows (instances) and 13 columns (features).

Features: It includes various features that are potentially useful for predicting loan approval decisions. Some common features in such datasets include:

Applicant ID: Unique identifier for each loan applicant.

Gender: Gender of the applicant (e.g., Male, Female).

Marital Status: Marital status of the applicant (e.g., Married, Single, Divorced).

Education: Educational qualification of the applicant (e.g., Graduate, Not Graduate).

Dependents: Number of dependents of the applicant.

Applicant Income: Income of the applicant.

Coapplicant Income: Income of the co-applicant (if any).

Loan Amount: Amount of the loan requested by the applicant.

Loan Term: Term of the loan (e.g., 15 years, 30 years).

Credit History: Credit history of the applicant (e.g., 1 for good credit history, 0 for poor credit history).

Property Area: Area where the property is located (e.g., Urban, Semiurban, Rural).

Loan Status: The target variable indicating whether the loan was approved or not (e.g., Approved, Not Approved).

Objective: The primary objective of using this dataset is to develop a predictive model that can accurately predict whether a loan application will be approved or denied based on the provided features. This prediction can help financial institutions streamline their loan approval process and make more informed decisions.

3. Descriptive statistics and inferential statistics:

```
[ ] df.describe()
```

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
count	614.000000	614.000000	592.000000	600.00000	564.000000
mean	5403.459283	1621.245798	146.412162	342.00000	0.842199
std	6109.041673	2926.248369	85.587325	65.12041	0.364878
min	150.000000	0.000000	9.000000	12.00000	0.000000
25%	2877.500000	0.000000	100.000000	360.00000	1.000000
50%	3812.500000	1188.500000	128.000000	360.00000	1.000000
75%	5795.000000	2297.250000	168.000000	360.00000	1.000000
max	81000.000000	41667.000000	700.000000	480.00000	1.000000

Fig. 2 Descriptive Statistics.

The image is a table containing summary statistics for several numerical variables.

The first row of the table shows the names of the variables. These include ApplicantIncome, CoapplicantIncome, LoanAmount, Loan_Amount_Term, and Credit_History.

The second row shows the number of non-missing values (count) for each variable. For example, there are 614 non-missing values for ApplicantIncome, but only 592 for LoanAmount.

The third row shows the mean (average) of each variable. For example, the average applicant income is $5403.46.

The fourth row shows the standard deviation (spread) of each variable. For example, there is a standard deviation of $6109.04 in applicant income.

The fifth row shows the minimum value of each variable. For example, the minimum applicant income is $150.

The sixth row shows the 25th percentile (first quartile) for each variable. For example, the 25th percentile of applicant income is $2877.50. This means that 25% of applicants have an income below $2877.50.

The seventh row shows the 50th percentile (median) for each variable. For example, the median applicant income is $3812.50. This means that 50% of applicants have an income below $3812.50 and 50% have an income above $3812.50.

The eighth row shows the 75th percentile (third quartile) for each variable. For example, the 75th percentile of applicant income is $5795.00. This means that 75% of applicants have an income below $5795.00.

The ninth row shows the maximum value of each variable. For example, the maximum applicant income is $81000.00.

Overall, the table provides a summary of the distribution of these financial measures in a dataset of loan applications.

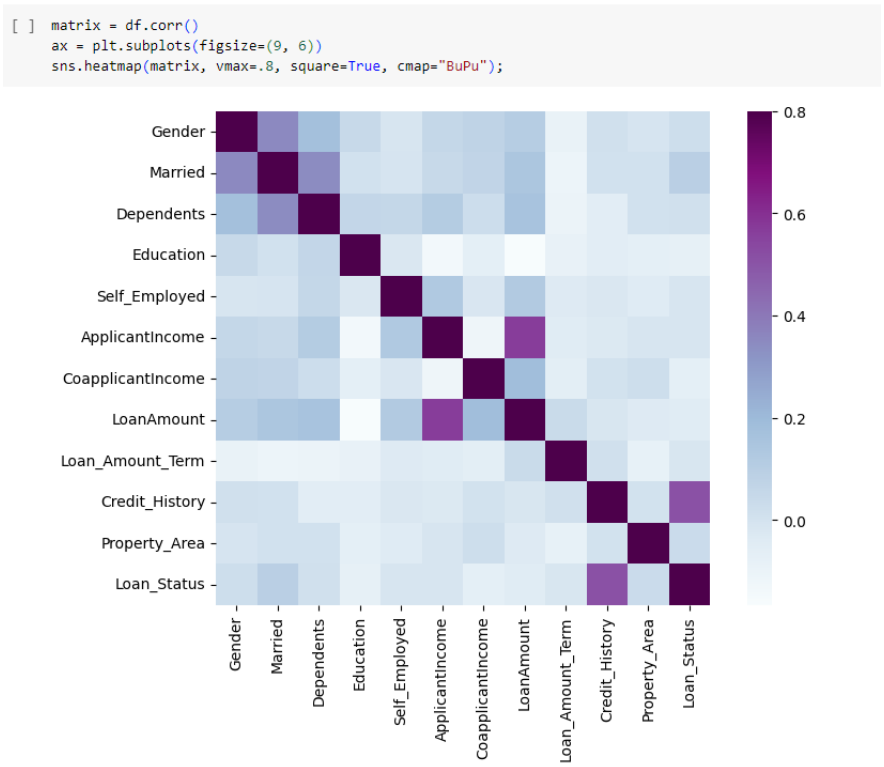


Fig. 3 Inferential Statistics.

The image is a heatmap, which is a graphical representation of a correlation matrix. A correlation matrix is a table that shows the correlation coefficient between all pairs of features in a dataset. The correlation coefficient is a measure of the linear dependence between two variables. It ranges from -1 to 1, with -1 indicating a perfect negative correlation, 0 indicating no correlation, and 1 indicating a perfect positive correlation.

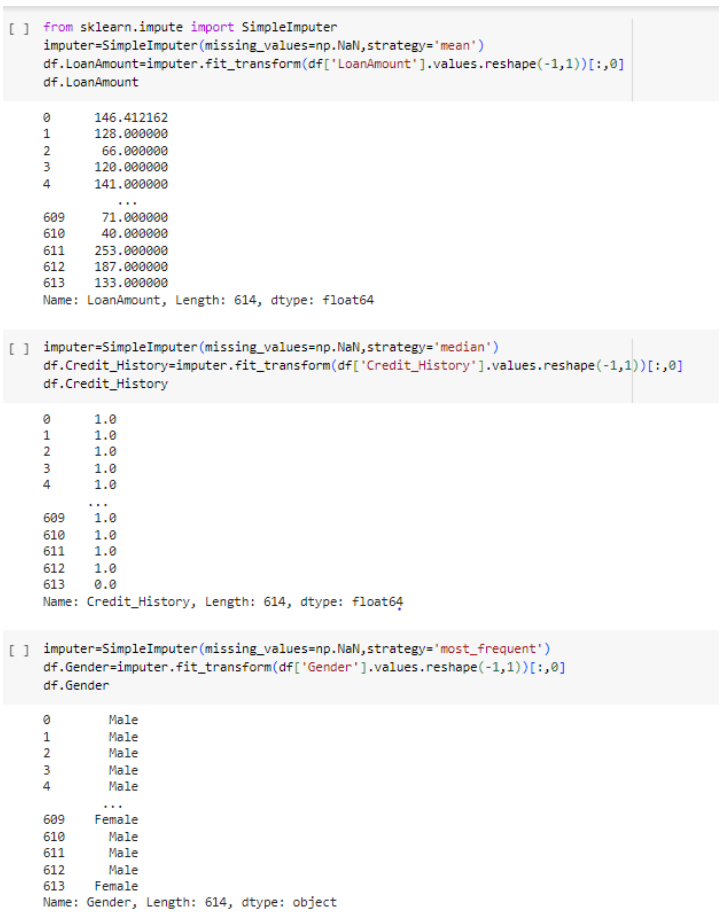
In the heatmap, the color intensity of each cell represents the correlation coefficient between the two features corresponding to the row and column labels of that cell. For instance, the dark blue cell in the upper left corner of the heatmap indicates a strong positive correlation between Gender and Married.

Here's a description of the correlations represented in the heatmap:

- Gender and Married have a strong positive correlation. This means that people who are identified as one gender are more likely to be married.
- Gender and Dependents have a moderate positive correlation. This means there is a weak tendency for people of a particular gender to have more dependents.
- There is a weak positive correlation between Married and Dependents, and Education and Dependents. This means that married couples and more educated people tend to have more dependents.

- Self_Employed has a weak positive correlation with ApplicantIncome and CoapplicantIncome. This means that self-employed people tend to have a slightly higher income than non-self-employed people. There is also a weak positive correlation between ApplicantIncome and CoapplicantIncome, which means that people with higher incomes tend to have co-applicants with higher incomes.
- LoanAmount has a weak positive correlation with ApplicantIncome, CoapplicantIncome, and Loan_Amount_Term. This means that people with higher incomes and co-applicants with higher incomes tend to apply for larger loans for longer terms.
- Credit_History has a weak positive correlation with LoanAmount. This means that people with better credit histories tend to apply for larger loans.
- There is no correlation between Property_Area and Loan_Status. This means that where someone lives has no bearing on whether their loan is approved.

4. Data Preprocessing:



```
[ ] from sklearn.impute import SimpleImputer
    imputer=SimpleImputer(missing_values=np.NaN,strategy='mean')
    df.LoanAmount=imputer.fit_transform(df['LoanAmount'].values.reshape(-1,1))[:,0]
    df.LoanAmount

0      146.412162
1      128.000000
2       66.000000
3      120.000000
4      141.000000
          ...
609     71.000000
610     40.000000
611    253.000000
612    187.000000
613    133.000000
Name: LoanAmount, Length: 614, dtype: float64

[ ] imputer=SimpleImputer(missing_values=np.NaN,strategy='median')
    df.Credit_History=imputer.fit_transform(df['Credit_History'].values.reshape(-1,1))[:,0]
    df.Credit_History

0      1.0
1      1.0
2      1.0
3      1.0
4      1.0
      ...
609    1.0
610    1.0
611    1.0
612    1.0
613    0.0
Name: Credit_History, Length: 614, dtype: float64

[ ] imputer=SimpleImputer(missing_values=np.NaN,strategy='most_frequent')
    df.Gender=imputer.fit_transform(df['Gender'].values.reshape(-1,1))[:,0]
    df.Gender

0        Male
1        Male
2        Male
3        Male
4        Male
        ...
609    Female
610      Male
611      Male
612      Male
613    Female
Name: Gender, Length: 614, dtype: object
```

Fig. 4 Handling Missing Values.

The image is a Python code that addresses missing values in a financial dataset, likely loan applications. It imports tools from scikit-learn to impute missing values in specific columns ("LoanAmount", "Credit_History", "Gender") using different strategies like replacing with the mean, median or most frequent value depending on the column. The code then displays the resulting DataFrame with imputed values.

```
[ ] from sklearn.preprocessing import LabelEncoder
    labelencoder=LabelEncoder()
    df['Loan_Status']=labelencoder.fit_transform(df['Loan_Status'])
    print(df['Loan_Status'])

    0      1
    1      0
    2      1
    3      1
    4      1
           ..
    609    1
    610    1
    611    1
    612    1
    613    0
    Name: Loan_Status, Length: 586, dtype: int64

[ ] df['Loan_Status'].value_counts()

    Loan_Status
    1    406
    0    180
    Name: count, dtype: int64

[ ] labelencoder=LabelEncoder()
    df['Gender']=labelencoder.fit_transform(df['Gender'])
    print(df['Gender'])
    df['Gender'].value_counts()

    0      1
    1      1
    2      1
    3      1
    4      1
           ..
    609    0
    610    1
    611    1
    612    1
    613    0
    Name: Gender, Length: 586, dtype: int64
    Gender
    1    480
    0    106
    Name: count, dtype: int64

[ ] labelencoder=LabelEncoder()
    df['Married']=labelencoder.fit_transform(df['Married'])
    print(df['Married'].unique())
    df['Married'].value_counts()
```

Fig. 5 Label Encoding.

The provided Python code snippet demonstrates label encoding, a technique to convert categorical data into numerical data suitable for machine learning models. It imports the LabelEncoder class from scikit-learn, creates an instance, and fits it to a DataFrame's column (e.g., Loan_Status) to assign unique integers to each category. This process is repeated for other categorical columns, transforming the data for further analysis.

5. Data Visualization:

```
plt.figure(figsize=(8,6))
plt.title("Loan Staus")
sns.countplot(x=df['Loan_Status'],color='orange')
```

```
<Axes: title={'center': 'Loan Staus'}, xlabel='Loan_Status', ylabel='count'>
```

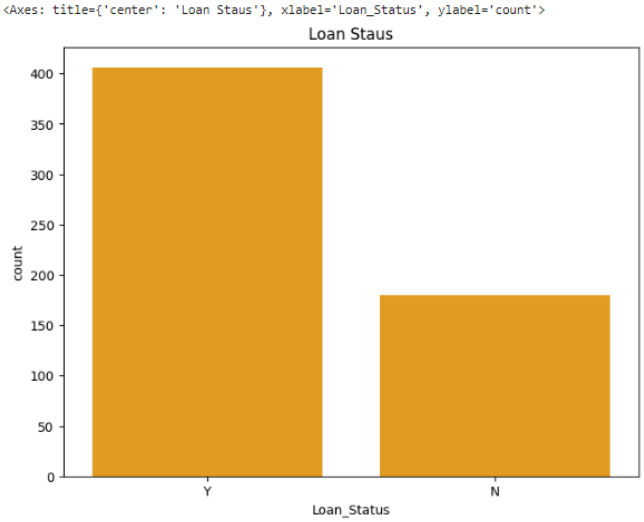


Fig. 6 Visualization of approved and rejected loans.

The image is a countplot which shows the number of loans that were approved and rejected. In this case, it appears that more loans were approved (Y) than rejected (N).

```
Gender=pd.crosstab(df['Gender'],df['Loan_Status'])
Gender.div(Gender.sum(1).astype(float), axis=0).plot(kind="bar", figsize=(4,4))
```

```
<Axes: xlabel='Gender'>
```

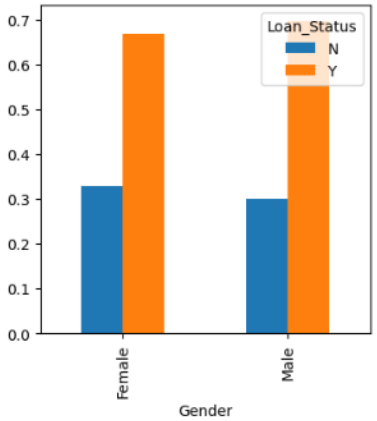


Fig. 7 Loan Approval Rates by Gender

The image you sent is a bar graph that shows the percentage of people who have a loan by gender. The y-axis shows the percentage and the x-axis shows gender. It appears that a higher percentage of females have loans than males.

6. Model Building:

```
[ ]  X = df.iloc[1:542,1:11].values
     y = df.iloc[1:542,11].values
```

```
[ ]  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,random_state=0)
```

Fig. 8 Splitting Data into Training and Testing Sets

This Python code splits data into training and testing sets. It extracts features (X) and target variables (y) from a DataFrame and uses train_test_split to divide them. The test_size parameter allocates 30% of the data for testing, while the remaining 70% goes to training. Setting random_state ensures consistent data splits if the code is run multiple times.

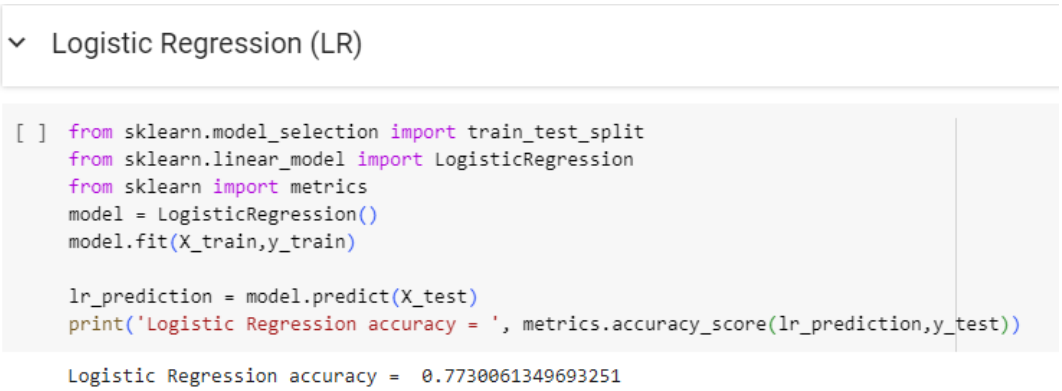


Fig. 9 Logistic Regression Model

This Python code implements logistic regression using scikit-learn. It first imports the LogisticRegression class and creates a model object. Then, it trains the model on the provided features (X_train) and target variables (y_train). The trained model can then predict the target variable for new data (X_test).

7. Model Validation:

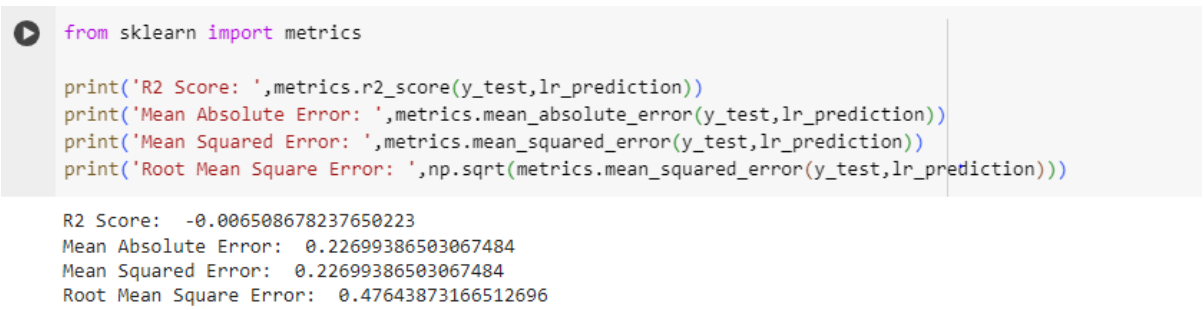


Fig. 10 Evaluating Linear Regression Model Performance

This Python code calculates performance metrics for a linear regression model. It imports functions from scikit-learn to compute the R-squared score (goodness of fit), mean squared error (average squared difference between predictions and actual values), and root mean squared error (square root of MSE, in the same units as predictions) for the model's predictions (lr_prediction) on the test data (y_test). These metrics help assess how well the model generalizes to unseen data.