# Loan Approval Prediction using Machine Learning

PROJECT COMPLETION REPORT

By:Dhavala Kartikeya Somayaji

**TABLE OF CONTENTS**

Executive Summary: Loan Approval Prediction Project

# Overview

The Loan Approval Prediction project implements machine learning to transform traditional loan assessment processes in the financial sector. The project utilizes Logistic Regression and Random Forest classification algorithms to analyze various applicant variables and predict loan approval decisions.

# Project Implementation

## Data Processing & Analysis

- Conducted comprehensive data preprocessing including:
    - Mean imputation for handling missing values
    - Label encoding for categorical variables
    - Extensive Exploratory Data Analysis (EDA)

## Model Performance

- ***Logistic Regression: 79.67% accuracy***
- ***Random Forest Classifier: 78.05% accuracy***

# Key Findings

## Critical Correlations

- Credit History strongly correlates with Loan Status
- Loan Amount shows significant correlation with Applicant Income
- Loan Amount Term demonstrates notable relationship with overall approval

## Demographic Insights

- Gender distribution shows male applicants as majority
- Most applicants are married and have graduate-level education
- Property location (urban/rural/semi-urban) has minimal impact on approval

### Principal Variables

- Most influential factors for loan approval:
    - Credit History
    - Loan Amount

# Project Impact

The project successfully demonstrates machine learning's capability to:

- Provide consistent loan approval predictions
- Process multiple variables simultaneously
- Eliminate potential human bias in assessment
- Maintain reliable accuracy rates

# Conclusion

The implementation, using a single dataset and primarily focusing on Logistic Regression, achieved satisfactory accuracy rates. The project effectively showcases how machine learning can streamline loan approval processes while maintaining prediction reliability. The results indicate potential for practical application in financial institutions, though there may be opportunities for further optimization.

# INTRODUCTION

Loan approval prediction using machine learning represents a revolutionary approach in the financial sector, transforming how lending institutions evaluate credit risk and make lending decisions. This technology harnesses the power of sophisticated algorithms to automate and enhance the traditional loan assessment process. By analyzing multiple variables simultaneously – including credit scores, income levels, employment history, and spending patterns – machine learning models can identify complex patterns that might escape human observation. These systems offer significant advantages over conventional methods: they reduce processing time, eliminate human bias, and maintain consistent evaluation criteria. Furthermore, the ability of machine learning models to continuously learn from new data ensures increasingly accurate predictions over time, helping financial institutions balance risk management with customer service efficiency.This model only takes one dataset and uses a very useful algorithm knows as Logistic regression to predict if the loan should be approved or not based on the given data. We have taken the raw data for this model and have done the data pre-processing with techniques such as EDA and many more which we will go into later in this report.

# IMPORTING NECESSARY LIBRARIES AND DATASET FOR THE PROJECT:

## The libraries we have used are :

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings('ignore')
pd.options.display.max_columns = None
```

We as we go into the code here and there we have imported certain functions we will be needing for the model

## Dataset we have used is :

The link to the dataset:
https://drive.google.com/file/d/19Dgneg919NbI266OYMw9QtQQ6SaTVbnm/view?usp=sharing

The code to import the dataset into the code :

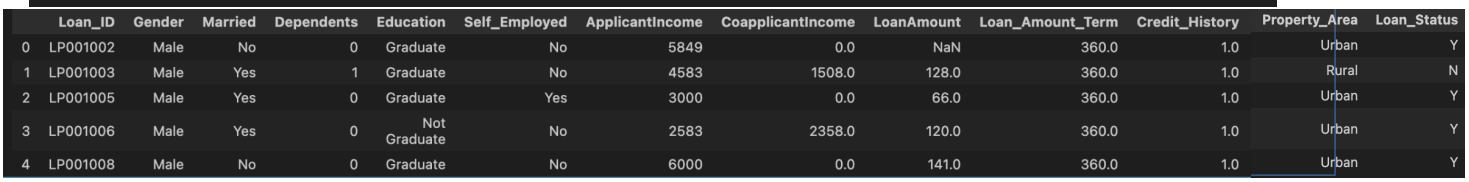
```
df = pd.read_csv(r"/Users/kartikeya/Documents/ML:AI:DL Projects/loan prediction.csv")
```

# BASIC CHECKS:

The basic checks performed were

1.To receive the first 4 rows and all the columns of the dataset:

```
df.head()
```



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

2. To receive the no of  rows and columns in the dataset:

```
df.shape
```

(614, 13)

3.To check the datatypes of all the columns in the dataset:

```
df.dtypes
```

```
Loan_ID              object
Gender               object
Married              object
Dependents           object
Education            object
Self_Employed        object
ApplicantIncome       int64
CoapplicantIncome   float64
LoanAmount          float64
Loan_Amount_Term    float64
Credit_History      float64
Property_Area        object
Loan_Status          object
dtype: object
```

4. To find out all the unique values in the dataset:

```
df.nunique()
```

```
Loan_ID            614
Gender               2
Married              2
Dependents           4
Education            2
Self_Employed        2
ApplicantIncome    505
```

```
CoapplicantIncome    287
LoanAmount           203
Loan_Amount_Term      10
Credit_History         2
Property_Area          3
Loan_Status            2
dtype: int64
```

5.Used to find the number of null values in the dataset to impute the data:

```
df.isna().sum()
```

```
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
```

6. Used to find the important statical values needed to understand the data:

```
df.describe()
```

| | ApplicantIncome | CoapplicantIncome | LoanAmount | Loan_Amount_Term | Credit_History |
|---|---|---|---|---|---|
| count | 614.000000 | 614.000000 | 614.000000 | 614.000000 | 614.000000 |
| mean | 5403.459283 | 1621.245798 | 145.752443 | 342.410423 | 0.855049 |
| std | 6109.041673 | 2926.248369 | 84.107233 | 64.428629 | 0.352339 |
| min | 150.000000 | 0.000000 | 9.000000 | 12.000000 | 0.000000 |
| 25% | 2877.500000 | 0.000000 | 100.250000 | 360.000000 | 1.000000 |
| 50% | 3812.500000 | 1188.500000 | 128.000000 | 360.000000 | 1.000000 |
| 75% | 5795.000000 | 2297.250000 | 164.750000 | 360.000000 | 1.000000 |
| max | 81000.000000 | 41667.000000 | 700.000000 | 480.000000 | 1.000000 |

7. Dropping Loan_id as it does not help the model in prediction

```
df.drop(['Loan_ID'],axis=1,inplace=True)
```

All these are the basic checks to understand the over of the dataset and be able to proceed to further steps with some knowledge on the data and get an accurate and precise model .The most important ones that will be used are df.head() and df.isna().sum(). Where df.head() will be used to understand the modifications done to the data and df.isna().sum() will be used to impuate the null values.

# EDA(Exploratory Data Analysis ):

Now  we are going to do some Data analysis to better understand what really going in the data and find some insights form our POV.

#Number of Categorical values

```
obj = (df.dtypes == 'object')
print("Categorical variables:",len(list(obj[obj].index)))
```

Categorical variables: 7

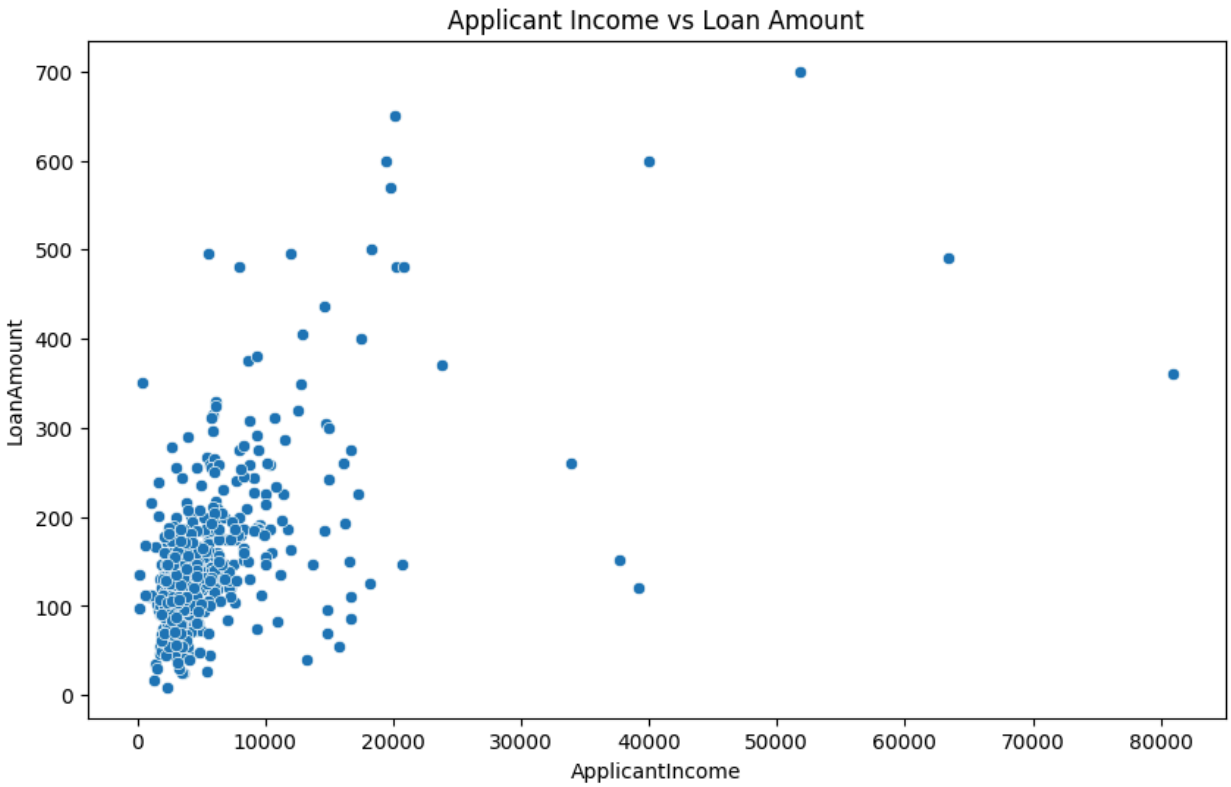
#Both Numerical and Categorical values:

```
categorical_columns = ['Gender', 'Married', 'Dependents', 'Education',
'Self_Employed',  'Property_Area', 'Loan_Status']
numerical = ['ApplicantIncome', 'CoapplicantIncome', 'LoanAmount', 'Loan_Amount_Term',
'Credit_History']
```

#The graphs to understand the dataset:

1.Scatter plot

```
plt.figure(figsize=(10, 6))
sns.scatterplot(x='ApplicantIncome', y='LoanAmount', data=df)
plt.title('Applicant Income vs Loan Amount')
plt.show()
```




Here in this plot we are trying to understand many things like:

**Positive Correlation:**

- If you see that as "Applicant Income" increases, "Loan Amount" also tends to increase, this indicates a positive correlation. Higher income applicants might be eligible for or might request larger loan amounts.

**Clusters or Patterns:**

- You might observe clusters of points indicating different groups of applicants. For example, one cluster might represent lower-income applicants requesting smaller loans, while another represents higher-income applicants requesting larger loans.

**Outliers:**

- Outliers might be present, which are points that deviate significantly from the general pattern. For instance, a low-income applicant requesting a high loan amount or a high-income applicant requesting a small loan amount. Identifying outliers can be useful for further investigation.

**Homogeneity or Variability:**

- The spread of points can indicate the variability in loan amounts requested by applicants with similar incomes. A wide spread suggests high variability, whereas a narrow spread indicates that applicants with similar incomes request similar loan amounts.

**Loan Caps or Limits:**

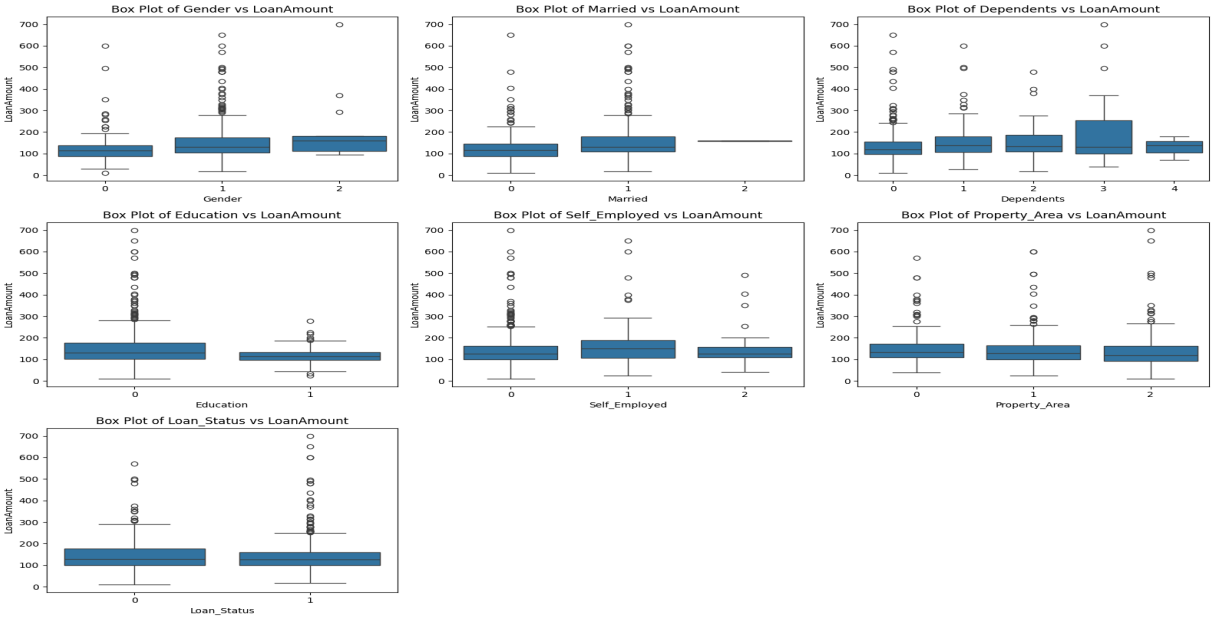
- If there's a visible horizontal line or a cutoff point where loan amounts do not exceed a certain value, this might indicate a cap or limit on loan amounts imposed by the lending institution.

**Income Thresholds:**

- You might observe that below a certain income level, loan amounts are consistently lower, indicating that there might be income thresholds that significantly influence loan eligibility or the amount approved.

2. Box plot:

```
plt.figure(figsize=(16, 12))

for i, feature in enumerate(categorical_columns, 1):

   plt.subplot(3, 3, i)

   sns.boxplot(x=feature, y='LoanAmount', data=df)

   plt.title(f'Box Plot of {feature} vs LoanAmount')

plt.tight_layout()

plt.show()
```




1. Gender vs Loan Amount:

- If the median loan amount for males is higher than for females, it might suggest that males tend to take out larger loans.
- If there are many outliers for one gender, it could indicate that some individuals in that gender category are taking unusually large or small loans.

2. Education vs Loan Amount:

- If graduates have a higher median loan amount than non-graduates, it might suggest that education level influences the loan amount.
- A narrow IQR for graduates might indicate that loan amounts for graduates are more consistent.

3. Self_Employed vs Loan Amount:

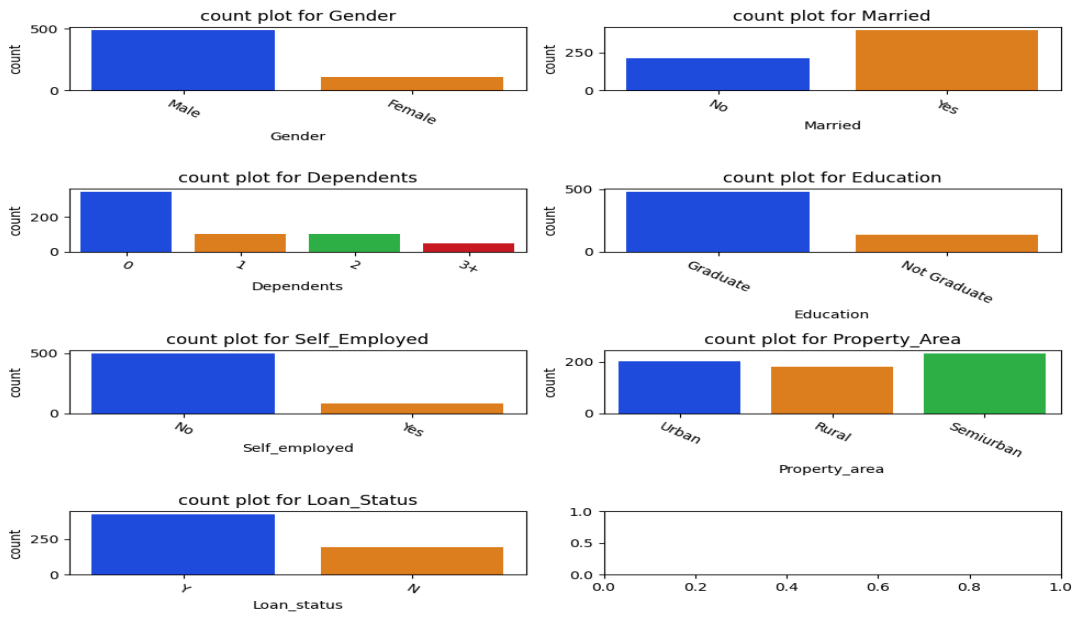
- If self-employed individuals have a higher median loan amount than those who are not self-employed, it could suggest that self-employed individuals tend to take out larger loans.
- A wider IQR for self-employed individuals might indicate more variability in loan amounts for this group.

4. Property_Area vs Loan Amount:

- If urban areas show higher median loan amounts compared to rural areas, it might suggest that property location influences the loan amount.
- Outliers in semiurban areas might indicate a few exceptionally high or low loan amounts.

3.Count plot:

```
fig, axes = plt.subplots(nrows=4, ncols = 2, figsize=(10,8))

axes=axes.ravel()



for i, column in enumerate(categorical_columns):

   sns.countplot(x=df[column],data=df, palette='bright' , ax=axes[i])

   axes[i].set_title(f"count plot for {column}")

   axes[i].set_xticklabels(axes[i].get_xticklabels(), rotation = -30)

   axes[i].set_xlabel(column.capitalize())

   axes[i].set_ylabel('count')



plt.tight_layout()

plt.show()
```



1. Gender vs Count (Gender):

- **Insight:** There are more male applicants than female applicants. The data is heavily skewed toward males.
- **Potential Impact:** This could affect model predictions if gender plays a role in loan approval, as the imbalance may introduce bias.

2. Marital Status vs Count (Married):

- **Insight:** The majority of applicants are married, with fewer unmarried applicants.
- **Potential Impact:** Marital status might be a predictor in loan approval models, and the imbalance might need to be considered when training a model.

3. Dependents vs Count (Dependents):

- **Insight:** Most applicants have 0 dependents, followed by 1 dependent. The number of applicants with 2 or more dependents is significantly lower.
- **Potential Impact:** The feature "Dependents" may not be as useful if the number of dependents is heavily skewed, but it might still provide valuable insights into the applicants' financial obligations.

4. Education vs Count (Education):

- **Insight:** There are more graduates than non-graduates in the dataset. Graduates are the majority group.
- **Potential Impact:** Education level might correlate with loan eligibility, as higher educational attainment can often lead to higher income and better loan repayment ability.

5. Self-Employed vs Count (Self_Employed):

- **Insight:** A smaller number of applicants are self-employed compared to those who are not.
- **Potential Impact:** Self-employment could be a significant feature affecting loan approval, with self-employed individuals possibly facing more scrutiny due to varying income stability.

6. Property Area vs Count (Property_Area):

- **Insight:** The applicants are fairly evenly distributed across the three property areas: urban, semiurban, and rural, with urban and semiurban areas having a slight majority.
- **Potential Impact:** The property area might have an impact on the loan approval process. Urban areas might have different eligibility criteria compared to rural areas.

7. Loan Status vs Count (Loan_Status):

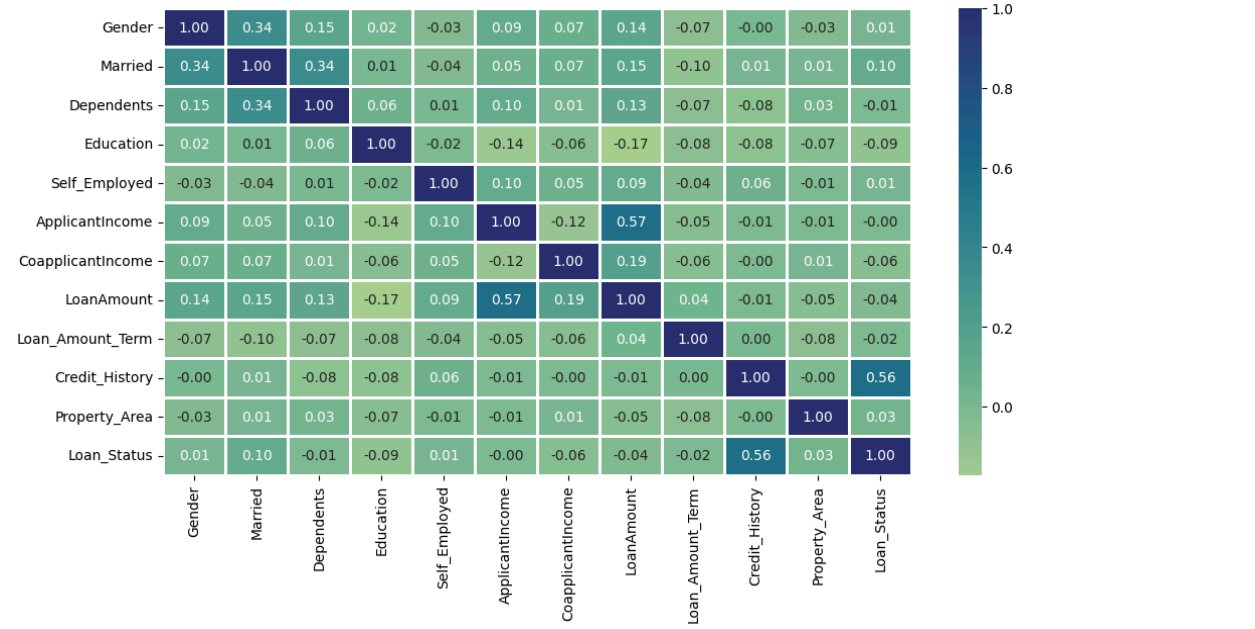
- **Insight:** The data is imbalanced, with a majority of loan approvals (labeled "Y" for yes) compared to rejections (labeled "N" for no).
- **Potential Impact:** This imbalance should be addressed in modeling to avoid bias. Techniques like oversampling, undersampling, or using metrics like F1 score can help balance the model's performance.

8. Missing Values in Loan Status:

- **Insight:** There's an empty or missing bar at the end of the count plot, possibly indicating missing or incomplete loan status data. This may need cleaning or imputation.
- **Potential Impact:** Missing values in critical features like loan status can affect model training and predictions. Proper handling of missing data (e.g., imputation or removal) is necessary for effective modeling.

4.Correlation and Heatmap:

```
plt.figure(figsize=(12,6))

sns.heatmap(df.corr(),cmap='crest',fmt='.2f',
            linewidths=2,annot=True)
```




From correlation we get very useful insights such as:

**Loan Amount and Loan Amount Term:**

- **Correlation:** 0.57 (moderate positive relationship).
- **Insight:** A longer loan term might be associated with larger loan amounts, reflecting the loan structure and repayment plans.

**Applicant Income and Coapplicant Income:**

- **Correlation:** 0.19 (weak positive relationship).
- **Insight:** The incomes of applicants and co-applicants are weakly correlated, suggesting that higher co-applicant income slightly correlates with higher applicant income, but they are largely independent.

**Loan Amount and Applicant Income:**

- **Correlation:** 0.57 (moderate positive relationship).
- **Insight:** Higher applicant income is associated with larger loan amounts, likely because individuals with higher income are eligible for higher loan amounts.

**Credit History and Loan Status:**

- **Correlation:** 0.56 (moderate positive relationship).
- **Insight:** Having a good credit history increases the likelihood of loan approval, suggesting that applicants with a good credit history are more likely to have their loan approved.

**Loan Amount and Loan Status:**

- **Correlation:** 0.56 (moderate positive relationship).
- **Insight:** Higher loan amounts are more likely to be associated with loan approval. This could be due to the criteria for approving higher loans being linked to factors like income and credit history.

**Property Area and Loan Status:**

- **Correlation:** 0.03 (very weak correlation).
- **Insight:** Property area has little to no impact on the loan status, suggesting that the area (urban, semiurban, rural) does not strongly affect whether a loan is approved or denied in this dataset.

**Dependents and Loan Amount:**

- **Correlation:** 0.13 (very weak positive relationship).
- **Insight:** The number of dependents has a slight positive relationship with the loan amount, but this correlation is weak and might not significantly affect loan amount decisions.

**Self-Employed and Other Features:**

- **Correlation:** Weak correlations with other variables, such as ApplicantIncome (0.10) and LoanAmount (0.05).
- **Insight:** Being self-employed has little impact on income or loan amount in this dataset.

**Education and Loan Status:**

- **Correlation:** -0.09 (very weak negative relationship).
- **Insight:** Education has almost no effect on loan approval, suggesting that other factors like income, credit history, and loan amount are more important in determining loan approval.

General Insights from the Heatmap:

- **Strong Predictors of Loan Approval:**
    - Credit_History and LoanAmount are strongly correlated with Loan_Status, indicating their importance in predicting loan approval.
    - LoanAmount and ApplicantIncome show moderate positive correlations, suggesting higher-income applicants are more likely to be approved for larger loan amounts.
- **Moderate Correlations:**
    - Variables like LoanAmount, ApplicantIncome, and CoapplicantIncome show moderate correlations with each other, indicating that higher-income applicants and their co-applicants are more likely to apply for and be approved for higher loan amounts.
- **Weak Correlations:**
    - Features like Property_Area, Education, and Dependents show weak correlations with the target variable (Loan_Status), suggesting they may not be as influential in loan approval predictions compared to other variables.

We can say the most important variables in predicting loan status appear to be **Credit History** and **Loan Amount**, while **Education**, **Property Area**, and **Dependents** have minimal impact.

# FEATURE ENGINEERING(Encoding ,imputation etc):

So from the EDA we had understood which feature are the most important in the prediction of sanction of the loan and which features which are so needed or as useful to predict if the loan will be sanctioned or not .The most important are **Credit History** and **Loan Amount.**

#Encoding

Now what we need to do is to encode the data as the model can not understand letters or words it can only understand numbers so to achive this we are going use label encoding.

```
from sklearn import preprocessing

label_encoder = preprocessing.LabelEncoder()
obj = (df.dtypes == 'object')
for col in list(obj[obj].index):
 df[col] = label_encoder.fit_transform(df[col])
```

So the above code endcodes the data with using the label encoded function which is a function of the the imported fucntion preprocessing from the library sklearn.
So now to check if the data is encoded we will use this programm.

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

for col in categorical_columns:
   print(f"Unique values in '{col}': {df[col].unique()}")
```

Unique values in 'Gender': [1 0 2]
Unique values in 'Married': [0 1 2]
Unique values in 'Dependents': [0 1 2 3 4]
Unique values in 'Education': [0 1]
Unique values in 'Self_Employed': [0 1 2]
Unique values in 'Property_Area': [2 0 1]
Unique values in 'Loan_Status': [1 0]

As we can see that all the Catagorical vaules are encoded.

#Impuatation:

It is the process of filling the null values as the modle can only take continuous data and if null values are present then we wont get an good accuracy from the model.So to do this we are using mean to replace all the null values of this data .

***Before the imputation:***

```
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
```

***After  the imputation:***

```
for col in df.columns:
  df[col] = df[col].fillna(df[col].mean())
 df.isna().sum()
```

Output:

```
Gender             0
Married            0
Dependents         0
Education          0
Self_Employed      0
ApplicantIncome    0
CoapplicantIncome  0
LoanAmount         0
Loan_Amount_Term   0
Credit_History     0
Property_Area      0
Loan_Status        0
dtype: int64
```

Here i felt there was no for combining of features or any sort of extra encoding like one hot encoding creating new columns for each new feature as that would complicate it more.

## TRAIN,TEST,SPLIT:

We have done the tuff part which was the data preprocessing part . Now we have to train the data.So in train test split we first split the data as we wish here we split the dat 20 percent for testing and 80 percent for training the model.

```
from sklearn.model_selection import train_test_split
X = df.drop(['Loan_Status'],axis=1)
Y = df['Loan_Status']
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2,
random_state=1)
```

The column to predict sis Loan_Status so we have it as our y variable.

## PREDICTION USING DIFFERENT ALGORITHMS:

So now after training the data we can make the prediction and get the accuracy of our model We are going to use Random forest Classifier and Logistic Regression algorithm.

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
rfc = RandomForestClassifier(n_estimators = 7,
                             criterion = 'entropy',
                             random_state =7)
lc = LogisticRegression()
for clf in (rfc, lc):
   clf.fit(X_train, Y_train)
   Y_pred = clf.predict(X_test)
   print("Accuracy score of ",
       clf.__class__.__name__,"=",
       100*metrics.accuracy_score(Y_test,Y_pred))
```

Output:

***Accuracy score of  RandomForestClassifier = 78.04878048780488***

***Accuracy score of  LogisticRegression = 79.67479674796748***

So the accuracy of the model is good as there is a good percentage of accuracy and not any abnormal amount this says there was not any overfitting .We got better results compravitvly form the ***LogisticRegression*** algorithm but .

## LESSONS LEARNED:

The main lesson i have learnt i that to keep it simple while solving ML problems for example the first time i had approached this problem i had complicated it by using both label and one hot encoding due to which i had gotten an accuracy of 7 percent using Logistic regression model and 25 percent using the Random forest model.Also i learned how to apply some concepts and also leared when and where they are needed. To keep it simple and build on that is my key take away from this project.

## Vote of Thanks:

I would like to than Abhishek rao sir for teaching me enough to able to complete this project successfully and his guidance played an important role in completing this project without any issues.