Project Report

Loan approval analysis is a binary classification problem. The objective of this analysis is to predict the approval of future loan application. It is beneficial for financial institutions to automate the process using machine learning as it reduces the duration of loan application processing up to 40%. It is also less riskier compared to the manual one because manual processing prone to error and fraud.

First, load the required libraries, download and load the dataset which is a csv file into python.

```
[2] import pandas as pd
    import numpy as np
    from sklearn.preprocessing import StandardScaler
    from sklearn.model_selection import train_test_split
    import matplotlib.pyplot as plt
    import seaborn as sns

[3] from google.colab import files
    upload = files.upload()

    Choose Files  LoanAppro...rediction.csv
    • LoanApprovalPrediction.csv(text/csv) - 37135 bytes, last modified: 6/11/2023 - 100% done
    Saving LoanApprovalPrediction.csv to LoanApprovalPrediction.csv

[4] data = pd.read_csv('LoanApprovalPrediction.csv')
    data.head()
```

The dataset used in this analysis contains various background information of applicants that is concluded in 598 observations and 13 features, summarized in the following table.

Feature	**Summary**
Loan_ID	Unique ID of each applicant
Gender	The gender of applicant (Male/Female)
Married	The marital status of applicant (Yes or No)
Dependents	The number of dependents that applicant has (0, 1, 2, 3)
Education	The status of graduation of applicant (Graduate or Not Graduate)
Self_Employed	The status of self-employment of applicant (Yes or No)
ApplicantIncome	The amount of applicant income
CoapplicantIncome	The amount of co-applicant income
LoanAmount	The loan amount (in thousands)
Loan_amount_Term	The duration of loan (in months)
Credit_History	Credit history of applicant' loan repayment
Property_Area	The location of property (Urban, Rural, or Semiurban)
Loan_Status	The status of loan approval (Yes or No)

The target feature here is Loan_Status and the loan eligibility is made based on gender, marital status, the number of dependents, education, loan amount, and credit history.

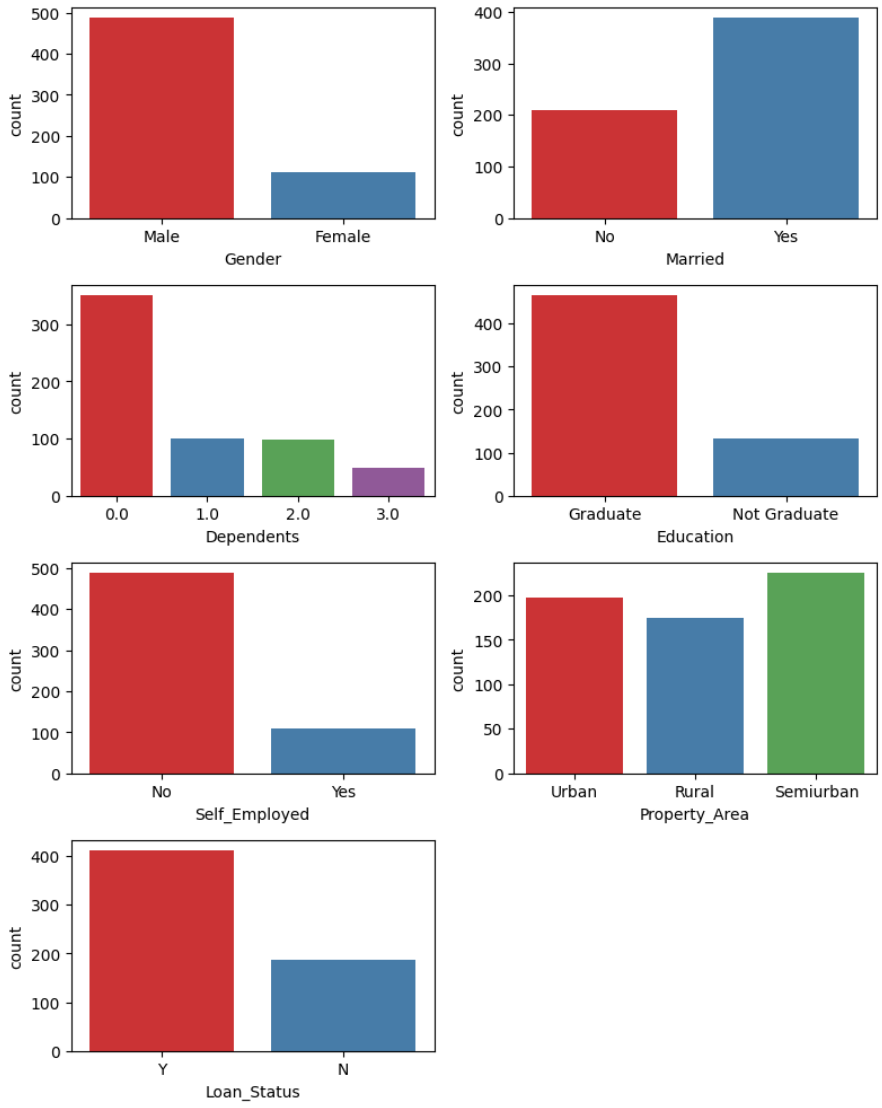
Data Pre-processing. The *isnull()* and *sum()* methods are used to find missing values in the dataset. Only four columns have non zero number of null values, with Credit_History having the most, which is 49, and the least is Dependents, 12 values. Method *mode()* or *mean()* that returns mode or mean of the column values will be used to replace those null values in order to make the dataset fit for analysis. The Loan_ID column is also dropped as it is completely unique and not correlated to other columns.

```
[9] data['Dependents'] = data['Dependents'].fillna(data['Dependents'].dropna().mode().values[0])
    data['LoanAmount'] = data['LoanAmount'].fillna(data['LoanAmount'].dropna().mean())
    data['Loan_Amount_Term'] = data['Loan_Amount_Term'].fillna(data['Loan_Amount_Term'].dropna().mode().values[0])
    data['Credit_History'] = data['Credit_History'].fillna(data['Credit_History'].dropna().mode().values[0])

    # Removing the column which is not correlated to any other columns
    data = data.drop('Loan_ID', axis=1)

    data.isnull().sum()
```

Below is the plot of features that contributed to loan eligibility after missing values are treated.



Converting categorical data into numeric is the next step and shown below.

```
[11] data['Gender'] = data['Gender'].map({'Male':1, 'Female':0}).astype('int')
     data['Married'] = data['Married'].map({'Yes':1, 'No':0}).astype('int')
     data['Education'] = data['Education'].map({'Graduate':1, 'Not Graduate':0}).astype('int')
     data['Self_Employed'] = data['Self_Employed'].map({'Yes':1, 'No':0}).astype('int')
     data['Loan_Status'] = data['Loan_Status'].map({'Y':1, 'N':0}).astype('int')
     data = pd.get_dummies(data, columns =['Dependents', 'Property_Area'])
     data.head()
```

Once all the columns are numeric, we fit a StandardScaler module to normalize the numeric variables.

Creating Train and Test Dataset. Using train_test_split function from sklearn and a split ratio of 80:20, we create the train and test data sets as follows:

```
[15] X = data.drop('Loan_Status', axis=1)
     y = data['Loan_Status']

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

     print(f"Training dataset shape, X_train: {X_train.shape}, y_train: {y_train.shape}")
     print(f"Testing dataset shape, X_test: {X_test.shape}, y_test: {y_test.shape}")
```

We also create function to return error metrics for this analysis and use GridSearchCV to find the best parameter values of each model.

```
[23] from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

     def measure_error(y_true, y_pred, label):
         return pd.Series({'accuracy':accuracy_score(y_true, y_pred),
                           'precision': precision_score(y_true, y_pred),
                           'recall': recall_score(y_true, y_pred),
                           'f1': f1_score(y_true, y_pred)},
                          name=label)
```

```
[24] # We use GridSearchCV to search over specified parameter values of the model.
     from sklearn.model_selection import GridSearchCV
```

Machine Learning Training. There are three models trained for this loan prediction; Decision Tree, Random Forest, and Support Vector Classifier (SVC). The best number of nodes and the maximum depth of the tree for decision Tree Classifier are 3 and 1, respectively. The best parameters for Random Forest are 7 (maximum depth), log2 (maximum features), and 27 (the number of estimators). The last, for SVC, the best value for C (regularization parameter) is 1 and the best kernel use for the model is Gaussian Radian Basis Function (RBF). The following is the result of metric error for each model.

	train	test
accuracy	0.805439	0.816667
precision	0.786241	0.800000
recall	0.981595	0.988235
f1	0.873124	0.884211

Decision Tree

	train	test
accuracy	0.859833	0.816667
precision	0.831202	0.805825
recall	0.996933	0.976471
f1	0.906555	0.882979

Random Forest

	train	test
accuracy	0.826360	0.816667
precision	0.802993	0.805825
recall	0.987730	0.976471
f1	0.885832	0.882979

SVC

From the error metrics presented above, it is clear from the accuracy and f1 value that random forest model is the most suitable to predict the future loan application based on this dataset. However, it looks like the gap between accuracy and f1 value for train dataset and accuracy value and f1 for test dataset is the largest compared the other two models. Hence, it can be concluded that Support Vector Classifier is better to be used in predicting future loan approval based on this dataset.

Even though this analysis found the most suitable model to be used in predicting loan approval, for the better result, we should train this dataset for other traditional classification models such as logistic regression and XGBoost. Different dataset also could help in determining the most suitable for this kind of process since the dataset used in this analysis has less null values compared to other similar datasets.