A Project Report

on

LOAN PREDICTION ANALYSIS USING MACHINE LEARNING

Submitted in partial fulfillment of the
requirement for the award of the degreeof

Bachelor of Technology

Submitted By:-
Nikhil Kumar Tiwari 20SCSE1010249
Aditya Ranjan 20SCSE1010706

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING /
DEPARTMENT OF COMPUTERAPPLICATION
GALGOTIAS UNIVERSITY, GREATER NOIDA
INDIA.

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA

CANDIDATE'S DECLARATION

We hereby certify that the work which is being presented in the project entitled **"LOAN PREDICTION ANALYSIS USING MACHINE LEARNING"** in partial fulfillment of the requirements for the award of the B.Tech submitted in the School of Computing Science and Engineering of Galgotias University, Greater Noida, is an original work, Department of Computer Science and Engineering/ComputerApplication and Information and Science of School of Computing Science and Engineering , Galgotias University, Greater Noida.

The matter presented in the project has not been submitted by us for the award of any other degree of this or any other places.

Nikhil Kumar Tiwari 20SCSE1010249
Aditya Ranjan 20SCSE1010706

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Reviewer Name and Profession

Abstract

In our banking system the main source of income of any banks is on its credit line despite knowing the fact that they have many products to sell. Because they can earn from interest of those loans which they credit. A bank's profit or a loss depends to a large extent on loans i.e. whether the customers are paying back the loan or defaulting. By predicting the loan defaulters, the bank can reduce its Nonperforming Assets. This makes the study of this phenomenon very important. Previous research in this era has shown that there are so many methods to study the problem of controlling loan default. But as the right predictions are very important for the maximization of profits, it is essential to study the nature of the different methods and their comparison.

A very important approach in predictive analytics is used to study the problem of predicting loan defaulters: The Logistic regression model. The data is collected from the Kaggle for studying and prediction. Logistic Regression models have been performed and the different measures of performances are computed. The models are compared on the basis of the performance measures such as sensitivity and specificity. The final results have shown that the model produce different results. Model is marginally better because it includes variables (personal attributes of customer like age, purpose, credit history, credit amount, credit duration, etc.) other than checking account information (which shows wealth of a customer) that should be taken into account to calculate the probability of default on loan correctly.

Therefore, by using a logistic regression approach, the right customers to be targeted for granting loan can be easily detected by evaluating their likelihood of default on loan. The model concludes that a bank should not only target the rich customers for granting loan but it should assess the other attributes of a customer as well which play a very important part in credit granting decisions and predicting the loan defaulters.

Index Terms– Prediction, loan, outlier, component, Overfitting, Transform

Table of Contents

Introduction

The above Loan status prediction is a clear classification problem as we need to classify whether the Loan Status is yes or no. So, this can be solved by any of the classification techniques like Logistic Regression. Decision Tree Algorithm Random Forest Technique. This paper is divided into four sections

(i) Data Collection

(ii) Comparison of machine learning models on collected data

(iii) Training of system on most promising model

(iv) Testing Data Set The training data set is now supplied to machine learning model, on the basis of this data set the model is trained.

Every new applicant detail filled at the time of application form acts as a test data set. After the operation of testing, model predict whether the new applicant is a fit case for approval of the loan or not based upon the inference it concludes on the basis of the training data sets. The dataset used in this project in loan prediction CSV that contains of 642 data about the borrower's information like gender, income, age etc.

To predict loan safety, the logistic regression algorithm is used. First the data is cleaned so as to avoid the missing values in the data set. To train our model data set of 1500 cases and 10 numerical and 8 categorical attributes has been taken.

Formulation of Problem

The objective of the problem is to pick out which customer will be able to pay the debt and which customer is likely will not be able to pay the debts. Clearly, we have to create a classification model here. We have to use algorithms like logistic regression, decision tree or random forest. We need to create a model that is accurate and the error percentage should be less.

The main objective of this project is to predict whether assigning the loan to particular person will be safe or not. In this project we are

Predicting the loan data by using some machine learning algorithms they are classification, logic regression, Decision Tree and gradient boosting.

- A classification model is run on data attempting to classify whether the person or client is eligible for get loan from any bank with good accuracy of statement.
- Our objectives included some points about this Loan Status Prediction.

Literature Survey

Logistic Regression is a popular and very useful algorithm of machine learning for classification problems. The advantage of logistic regression is that it is a predictive analysis. It is used for description of data and use to explain relationship between a single binary variable and single or multiple nominals, ordinal and ration level variables which are independent in nature. The model development for the prediction is taken in account using the sigmoid function in logistic regression as the outcome is targeted binary either 0 or 1. The dataset of bank customers has been divided into training and test data sets. The train dataset contains approximately 600+ rows and 13+ columns whereas the test dataset contains 300+ rows and 12+ columns, the test dataset does not contain the target variable.

Both the datasets are having missing values in their rows, and the mean, median or mode is used to fill the missing values but not removing the rows completely because the datasets are already small. Using the Feature Engineering techniques, the project is further proceeded and move towards the exploratory data analysis, where the dependent and independent variable is studied through statistics concepts such normal distribution, Probability density function etc. Study of the univariate, bivariate and multivariate analysis will give the view of the inside dependent and independent variable. The model is focusing on to target those customers who are eligible for loans and therefore the logistic regression is enabled using the sigmoid function as it divided the probability into binary output. Therefore the Prediction model can be developed.

Working of Project

When someone borrows some money from someone or some organization, in financial term it is known as loan distribution of the loans is the core business part of almost every banks. The main portion the bank's assets I s directly came from the profit earned from the loans distributed by the banks. The prime objective in banking environment is to invest their assets in safe hands where it is. Today many banks/financial companies approve loan after a regress process of verification and validation but still there is no surety whether the chosen applicant is the deserving right applicant out of all applicants. Through this system we can predict whether that particular applicant is safe or not and the whole process of validation of features is automated by machine learning technique.

The disadvantage of this model is that it emphasizes different weights to each factor but in real life sometime loan can be approved on the basis of single strong factor only, which is not possible through this system.

A. Data Collection

Data has been collected from the Kaggle one of the most data source providers for the learning purpose and hence the data is collected from the Kaggle, which had two data sets one for the training and another testing [12]. The training dataset is used to train the model in which datasets is further divided into two parts such as 80:20 or 70:30 the major datasets is used for the train the model and the minor dataset is used for the test the model and hence the accuracy of our developed model is calculated.

Block diagram and description

Loan Approval Prediction is chosen for prediction of the approval of loan. It uses the training data to for the learning purpose and then predicts on the test data.

***Train Data:** The train data consists of various attributes such as salary, marital status, loan accounts, and loan repayments on time etc. According to these factors we build a required model for loan prediction.

***Test Data:** The test data consist of various attributes such as salary, marital status, loan accounts except the loan approval status. The loan approval status is obtained. When we deploy the test data to the model which is built from the trained data.

The loan status is obtained after the deployment of test data to the model which is built from the trained data. The loan status consists of Customer id and loan status. It indicates for a particular customer loan is approved or not. If loan status is Y (Yes) then the customer is eligible for approval of loan and if it is N (No) then the customer is not eligible for approval of loan.

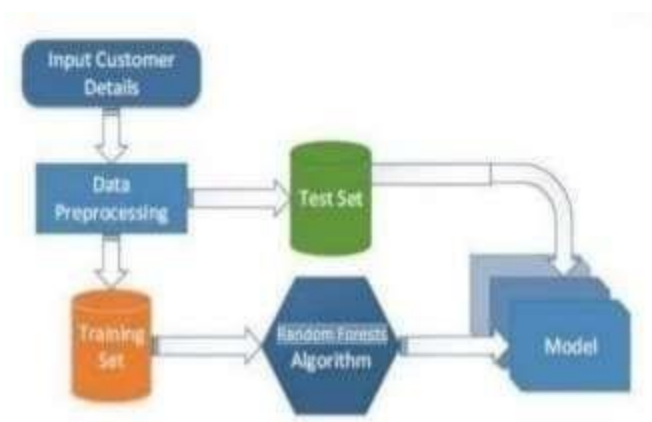


Fig. Block Diagram of Loan Approval Prediction System

The process of selecting a final machine learning model from among a group of candidate machine learning models for aparticular training dataset of Loan customer is called model selection. There are different types of model like logistic regression, SVM, KNN, etc. All these models have some merits and demerits for example predictive error gives the statistical noise in the data, the incompleteness of the sample data, and the limitations of each different model type. The chosen modelmeets the requirements and constraints of the stakeholders (Bank and Customers) project stakeholders. A model should have parameters like • Skillful as compared to naive models. • Skillful relative to other tested models. • Skillful relative to the state-of-the-art. Thus, Prediction of loan approval is a type of a classification problem and hence this model is used.

from sklearn.linear_model import LogisticRegression model = LogisticRegression() model.fit(x_train, y_train)

Future scope

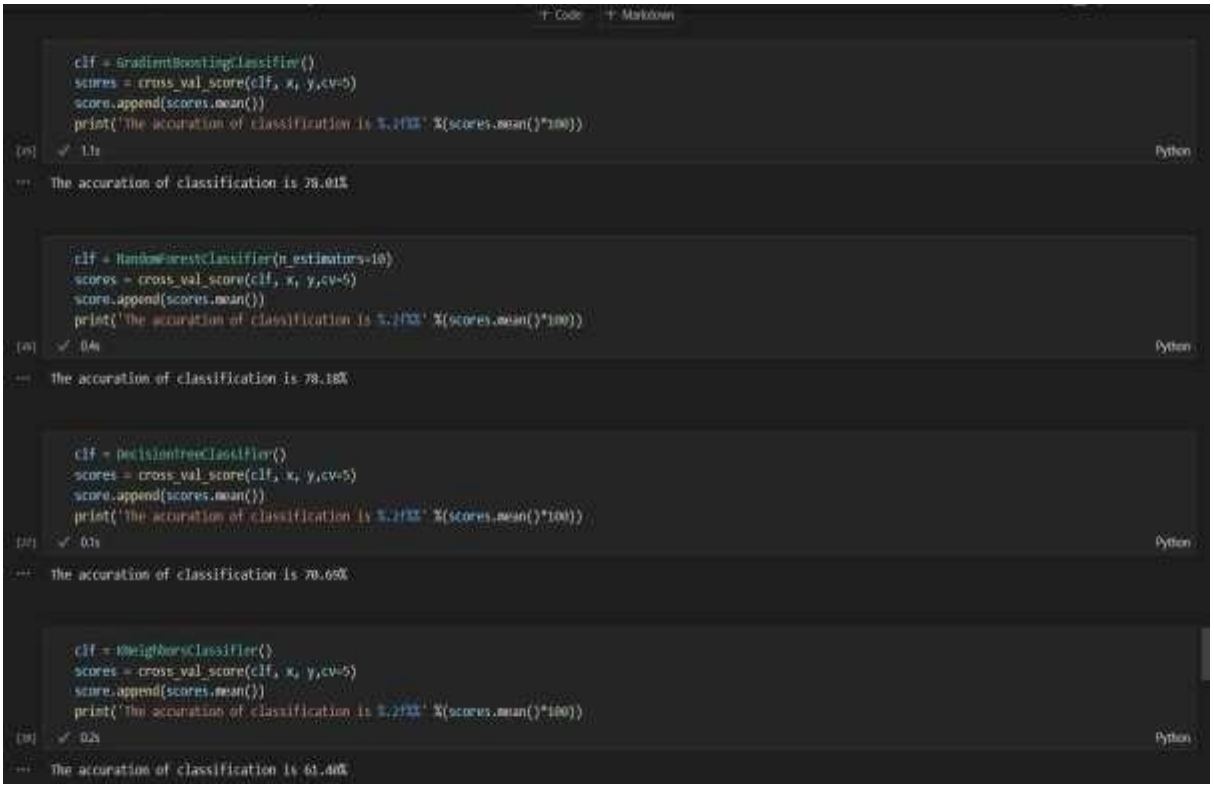
In future, this model can be used to compare various machine learning algorithm generated prediction models and the model which will give higher accuracy will be chosen as the prediction model.The disadvantage of this model is that it emphasizes different weights to each factor but in real life sometime loan can be approved on the basis of single factor only,which is not possible throught this system. So,this paper work can be extended to higher level in future. Predictive model for loans that uses machine learning algorithms, where the results from each graph of the paper can be taken as individual criteria for the machine learning algorithm.

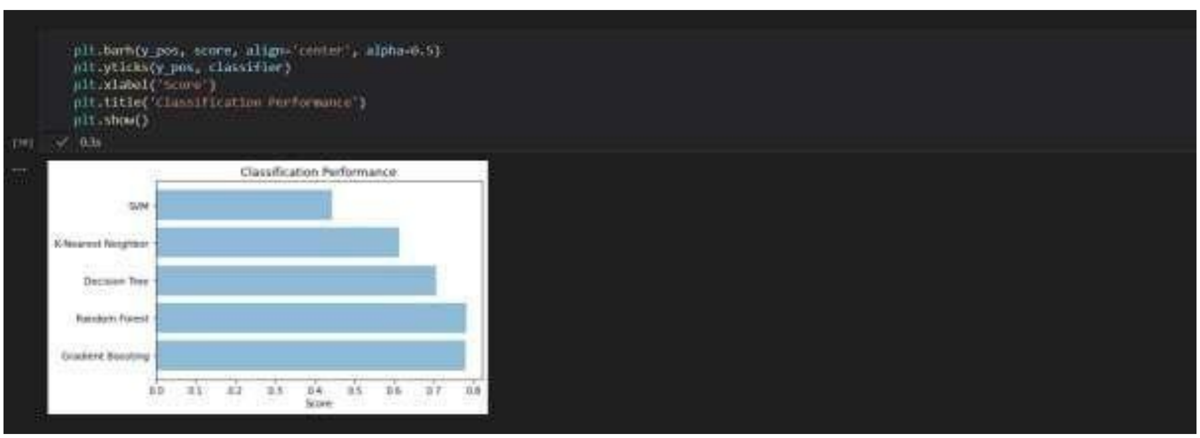
Conclusion

From the proper view of analysis this system can be used for detection of clients who are eligible for approval of loan. It is working perfect and can be used for all banking requirements. This system can be easily uploaded in any operating system. Since the technology is moving towards online, this system has more scope for the upcoming days. This system is more reliable. There is no issue if there are many no of customers applying for loan. This system accepts data for N no. of customers. In future we can add more algorithms to this system for getting more accurate results.

<u>Sample output</u>

These are the required outputs of the different classifiers used in the program:





Program

These are the screenshots of our program.

```
#Import packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import cross_val_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from  sklearn import svm
```

[5] Python

```
data = pd.read_csv("data set.csv")
```

[6] Python

```
data.head()
```

[7] Python

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome
0	LP001002	Male	No	0	Graduate	No	5849
1	LP001003	Male	Yes	1	Graduate	No	4583
2	LP001005	Male	Yes	0	Graduate	Yes	3000
3	LP001006	Male	Yes	0	Not Graduate	No	2583
4	LP001008	Male	No	0	Graduate	No	6000

```
data.info()
```

[8] Python

```
<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
```

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

[9] Python

```
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
```

```
print('Percent of missing "Gender" records is %.2f%%' %((data['Gender'].isnull().
```

[10] Python

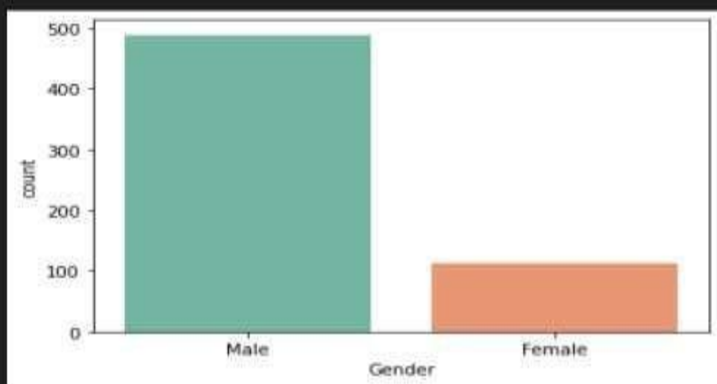
```
Percent of missing "Gender" records is 2.12%
```

```
print("Number of people who take a loan group by gender :")
print(data['Gender'].value_counts())
sns.countplot(x='Gender', data=data, palette = 'Set2')
```

[11] Python

```
Number of people who take a loan group by gender :
Male      489
Female    112
Name: Gender, dtype: int64

<AxesSubplot:xlabel='Gender', ylabel='count'>
```




```
print('Percent of missing "Married" records is %.2f%%' %((data['Married'].isnull(
```

[12] Python

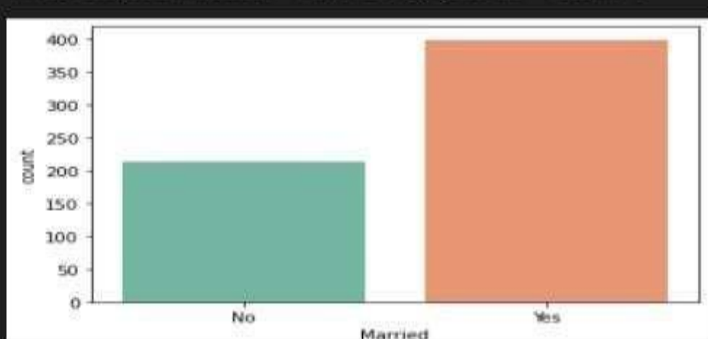
```
Percent of missing "Married" records is 0.49%
```

```
print("Number of people who take a loan group by marital status :")
print(data['Married'].value_counts())
sns.countplot(x='Married', data=data, palette = 'Set2')
```

[13] Python

```
Number of people who take a loan group by marital status :
Yes    398
No     213
Name: Married, dtype: int64

<AxesSubplot:xlabel='Married', ylabel='count'>
```



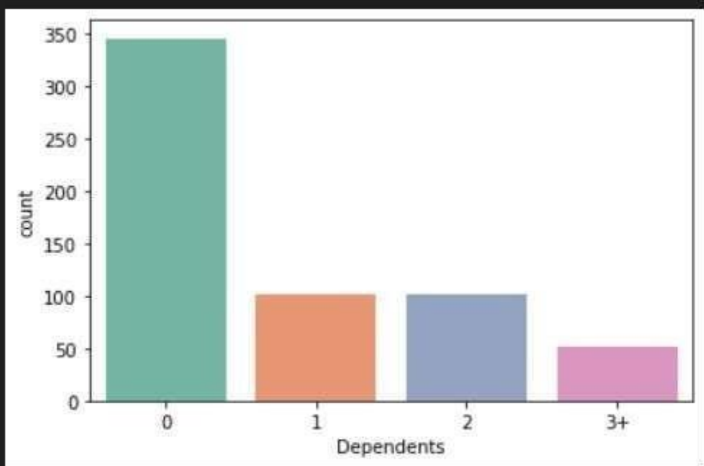
```
[14]
print('Percent of missing "Dependents" records is %.2f%%' %((data['Dependents'].i
```
Python

```
Percent of missing "Dependents" records is 2.44%
```

```
[15]
print("Number of people who take a loan group by dependents :")
print(data['Dependents'].value_counts())
sns.countplot(x='Dependents', data=data, palette = 'Set2')
```
Python

```
Number of people who take a loan group by dependents :
0     345
1     102
2     101
3+     51
Name: Dependents, dtype: int64

<AxesSubplot:xlabel='Dependents', ylabel='count'>
```



```
print('Percent of missing "Self_Employed" records is %.2f%%' %((data['Self_Employ
```

[16] Python

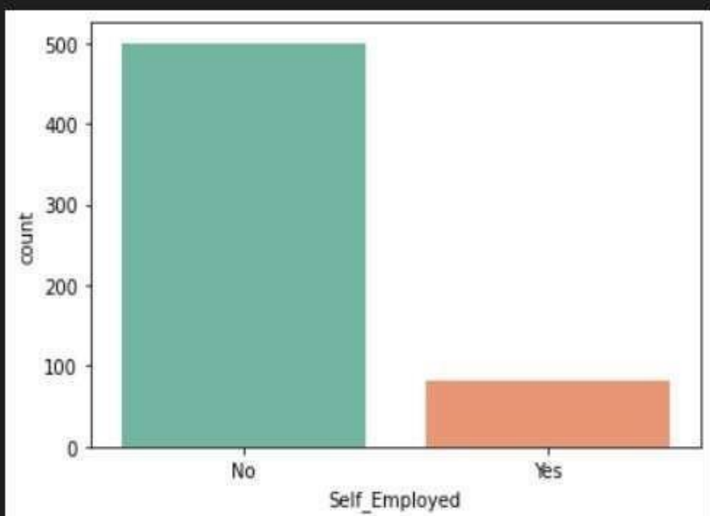
```
Percent of missing "Self_Employed" records is 5.21%
```

```
print("Number of people who take a loan group by self employed :")
print(data['Self_Employed'].value_counts())
sns.countplot(x='Self_Employed', data=data, palette = 'Set2')
```

[17] Python

```
Number of people who take a loan group by self employed :
No     500
Yes     82
Name: Self_Employed, dtype: int64

<AxesSubplot:xlabel='Self_Employed', ylabel='count'>
```



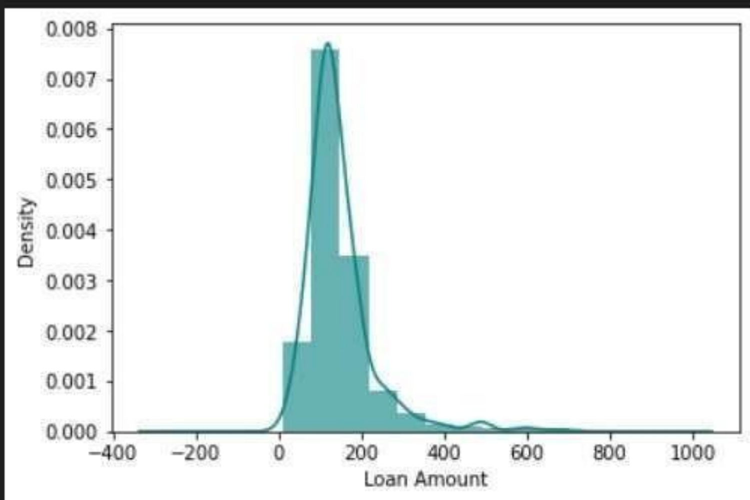
```python
print('Percent of missing "LoanAmount" records is %.2f%%' %((data['LoanAmount'].i
```

[18] Python

```
Percent of missing "LoanAmount" records is 3.58%
```

```python
ax = data["LoanAmount"].hist(density=True, stacked=True, color='teal', alpha=0.6)
data["LoanAmount"].plot(kind='density', color='teal')
ax.set(xlabel='Loan Amount')
plt.show()
```

[19] Python




```python
print('Percent of missing "Loan_Amount_Term" records is %.2f%%' %((data['Loan_Amo
```

[20] Python

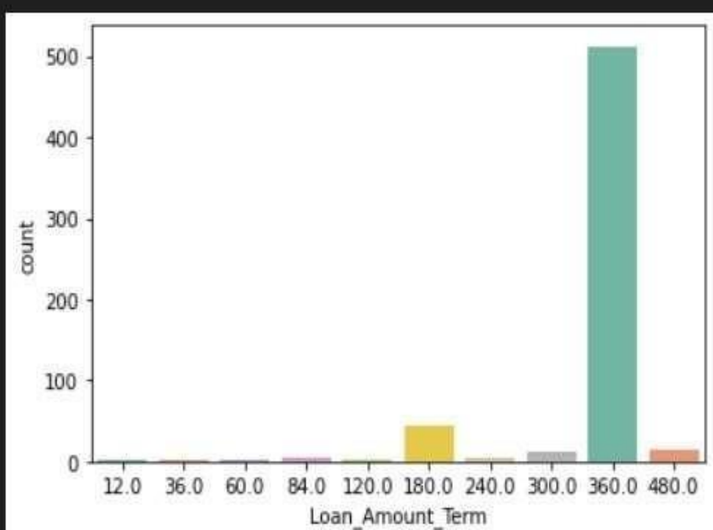
```
Percent of missing "Loan_Amount_Term" records is 2.28%
```

```python
print("Number of people who take a loan group by loan amount term :")
print(data['Loan_Amount_Term'].value_counts())
sns.countplot(x='Loan_Amount_Term', data=data, palette = 'Set2')
```

[21] Python

```
Number of people who take a loan group by loan amount term :
360.0    512
180.0     44
480.0     15
300.0     13
240.0      4
84.0       4
120.0      3
60.0       2
36.0       2
12.0       1
Name: Loan_Amount_Term, dtype: int64

<AxesSubplot:xlabel='Loan_Amount_Term', ylabel='count'>
```



```python
print('Percent of missing "Credit_History" records is %.2f%%' %((data['Credit_His
```

[22] Python

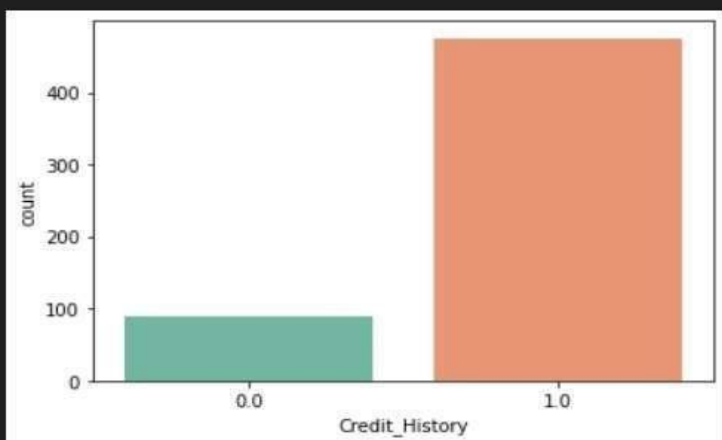
```
Percent of missing "Credit_History" records is 8.14%
```

```python
print("Number of people who take a loan group by credit history :")
print(data['Credit_History'].value_counts())
sns.countplot(x='Credit_History', data=data, palette = 'Set2')
```

[23] Python

```
Number of people who take a loan group by credit history :
1.0    475
0.0     89
Name: Credit_History, dtype: int64

<AxesSubplot:xlabel='Credit_History', ylabel='count'>
```




```python
train_data = data.copy()
train_data['Gender'].fillna(train_data['Gender'].value_counts().idxmax(), inplace=True)
train_data['Married'].fillna(train_data['Married'].value_counts().idxmax(), inplace=True)
train_data['Dependents'].fillna(train_data['Dependents'].value_counts().idxmax(), inplace=True)
train_data['Self_Employed'].fillna(train_data['Self_Employed'].value_counts().idxmax(), inplace=True)
train_data["LoanAmount"].fillna(train_data["LoanAmount"].mean(skipna=True), inplace=True)
train_data['Loan_Amount_Term'].fillna(train_data['Loan_Amount_Term'].value_counts().idxmax(), inplace=True)
train_data['Credit_History'].fillna(train_data['Credit_History'].value_counts().idxmax(), inplace=True)
```

Python

```python
train_data.isnull().sum()
train_data
```

Python

	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	146.412162	360.0	1.0	Urban	Y
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.000000	360.0	1.0	Rural	N
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.000000	360.0	1.0	Urban	Y
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.000000	360.0	1.0	Urban	Y
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.000000	360.0	1.0	Urban	Y
...	...	...	...	...	...	...	...	...	...	...	...	...	...
609	LP002978	Female	No	0	Graduate	No	2900	0.0	71.000000	360.0	1.0	Rural	Y
610	LP002979	Male	Yes	3+	Graduate	No	4106	0.0	40.000000	180.0	1.0	Rural	Y
611	LP002983	Male	Yes	1	Graduate	No	8072	240.0	253.000000	360.0	1.0	Urban	Y
612	LP002984	Male	Yes	2	Graduate	No	7583	0.0	187.000000	360.0	1.0	Urban	Y
613	LP002990	Female	No	0	Graduate	Yes	4583	0.0	133.000000	360.0	0.0	Semiurban	N

614 rows × 13 columns

```python
gender_stat = {"Female": 0, "Male": 1}
yes_no_stat = {'No' : 0,'Yes' : 1}
dependents_stat = {'0':0,'1':1,'2':2,'3+':3}
education_stat = {'Not Graduate' : 0, 'Graduate' : 1}
property_stat = {'Semiurban' : 0, 'Urban' : 1,'Rural' : 2}

train_data['Gender'] = train_data['Gender'].replace(gender_stat)
train_data['Married'] = train_data['Married'].replace(yes_no_stat)
train_data['Dependents'] = train_data['Dependents'].replace(dependents_stat)
train_data['Education'] = train_data['Education'].replace(education_stat)
train_data['Self_Employed'] = train_data['Self_Employed'].replace(yes_no_stat)
train_data['Property_Area'] = train_data['Property_Area'].replace(property_stat)
```

[26] Python

```python
data.info()
data.isnull().sum()
```

[27] Python

```
...
<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
```

```
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
```

```python
x = train_data.iloc[:,1:12]
y = train_data.iloc[:,12]
classifier = ('Gradient Boosting','Random Forest','Decision Tree','K-Nearest Neig
y_pos = np.arange(len(classifier))
score = []
```

[28] Python

```python
clf = GradientBoostingClassifier()
scores = cross_val_score(clf, x, y,cv=5)
score.append(scores.mean())
print('The accuration of classification is %.2f%%' %(scores.mean()*100))
```

[29] Python

```
The accuration of classification is 78.01%
```

```python
clf = RandomForestClassifier(n_estimators=10)
scores = cross_val_score(clf, x, y,cv=5)
score.append(scores.mean())
print('The accuration of classification is %.2f%%' %(scores.mean()*100))
```

[30] Python

```
The accuration of classification is 74.76%
```

```
clf = DecisionTreeClassifier()
scores = cross_val_score(clf, x, y,cv=5)
score.append(scores.mean())
print('The accuration of classification is %.2f%%' %(scores.mean()*100))
```

[31] Python

```
The accuration of classification is 69.87%
```

```
clf = KNeighborsClassifier()
scores = cross_val_score(clf, x, y,cv=5)
score.append(scores.mean())
print('The accuration of classification is %.2f%%' %(scores.mean()*100))
```

[32] Python

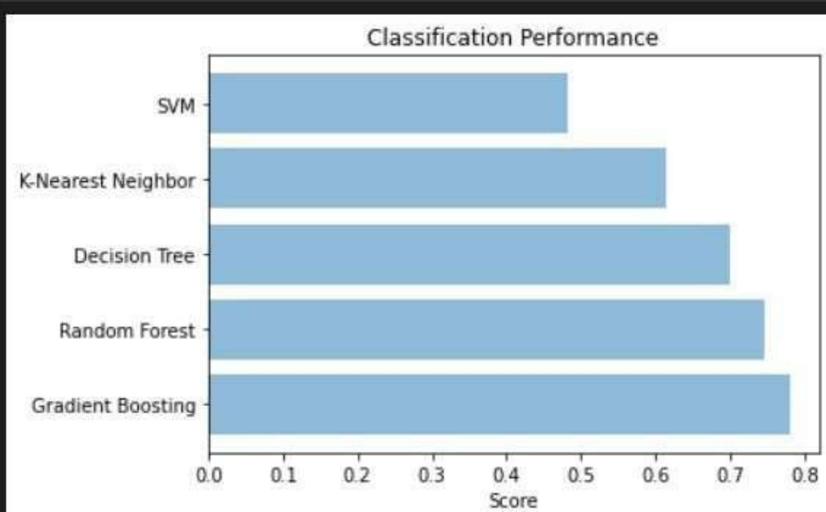
```
The accuration of classification is 61.40%
```

```
clf  =  svm.LinearSVC(max_iter=5000)
scores = cross_val_score(clf, x, y,cv=5)
score.append(scores.mean())
print('The accuration of classification is %.2f%%' %(scores.mean()*100))
```

[33] Python

```
plt.barh(y_pos, score, align='center', alpha=0.5)
plt.yticks(y_pos, classifier)
plt.xlabel('Score')
plt.title('Classification Performance')
plt.show()
```

[34] Python



References

1. Loan Approval Prediction | Kaggle
2. Machine learning - Wikipedia
3. ML | Data Preprocessing in Python - GeeksforGeeks
4. Best prediction with the help of machine learning (skyfilabs.com)