# *PREDICTING PERSONAL LOAN APPROVAL USING MACHINE LEARNING*

**TEAM MEMBERS NAME : MOHANRAJ P**

**MYTHILI C**

**MYTHILI V**

**NINGARAJ M**

**CLASS : B.SC. COMPUTER SCIENCE–III YEAR**

# TABLE OF INDEX

# 1 .INTRODUCTION

## OVERVIEW

**Predicting Personal Loan Approval Using Machine Learning:**

A loan is a sum of money that is borrowed and repaid over a period of time, typically with interest. There are various types of loans available to individuals and businesses, such as personal loans, mortgages, auto loans, student loans, business loans and many more. They are offered by banks, credit unions, and other financial institutions, and the terms of the loan, such as interest rate, repayment period, and fees, vary depending on the lender and the type of loan.

A personal loan is a type of unsecured loan that can be used for a variety of expenses such as home repairs, medical expenses, debt consolidation, and more. The loan amount, interest rate, and repayment period vary depending on the lender and the borrower's credit worthiness .To qualify for a personal loan, borrowers typically need to provide proof of income and have a good credit score.

Predicting personal loan approval using machine learning analyses a borrower's financial data and credit history to determine the likelihood of loan approval. This can help financial institutions to make more informed decisions about which loan applications to application to approve and which to deny.

## PURPOSE

The use of machine learning algorithms can help predict personal loan approval with a high degree of accuracy. By analyzing various data points such as credit score, income, employment history, and other relevant factors, the model can provide insights into whether an individual is likely to be approved for a loan or not.

Personal loan approval prediction using machine learning can be achieved by analyzing various factors that influence the decision of lenders to approve or reject a loan application. These factors may include the applicant's credit score, income, employment status, loan amount, loan purpose, and loan term.

The benefits of using machine learning for personal loan approval are numerous. Firstly, it can save time and effort for both lenders and borrowers by automating the approval process. Secondly, it can help lenders make more informed decisions about who to lend to, reducing the risk of default and ultimately increasing profitability. Finally, it can help borrowers access loans that they may not have been able to obtain otherwise, improving their financial wellbeing.

By using machine learning algorithms, the model can learn from historical data to identify patterns and trends that are associated with loan approvals. Once the model is trained, it can predict the probability of approval for a new loan application based on the input variables.

Overall, the use of machine learning for personal loan approval has the potential to revolutionize the lending industry and make credit more accessible to a wider range of individuals.

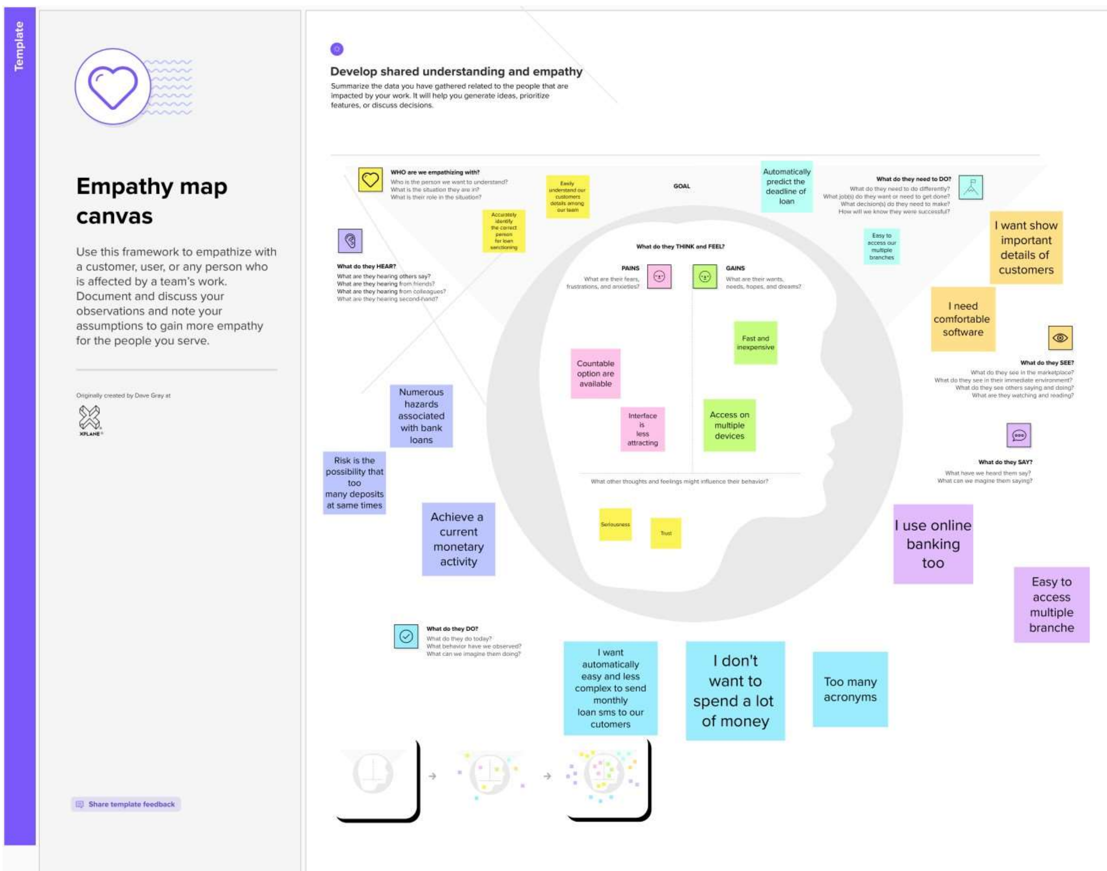
The use of this project can benefit both lenders and borrowers. Lenders can use the model to automate the loan approval process, reduce manual errors, and improve decision-making accuracy. Borrowers can benefit from faster loan processing times, increased transparency, and higher chances of getting approved for a loan.

Overall, personal loan approval prediction using machine learning can help streamline the loan approval process, improve efficiency, and provide a better customer experience.

# 2. PROBLEM DEFINITION & DESIGN THINKING

## EMPATHY MAP

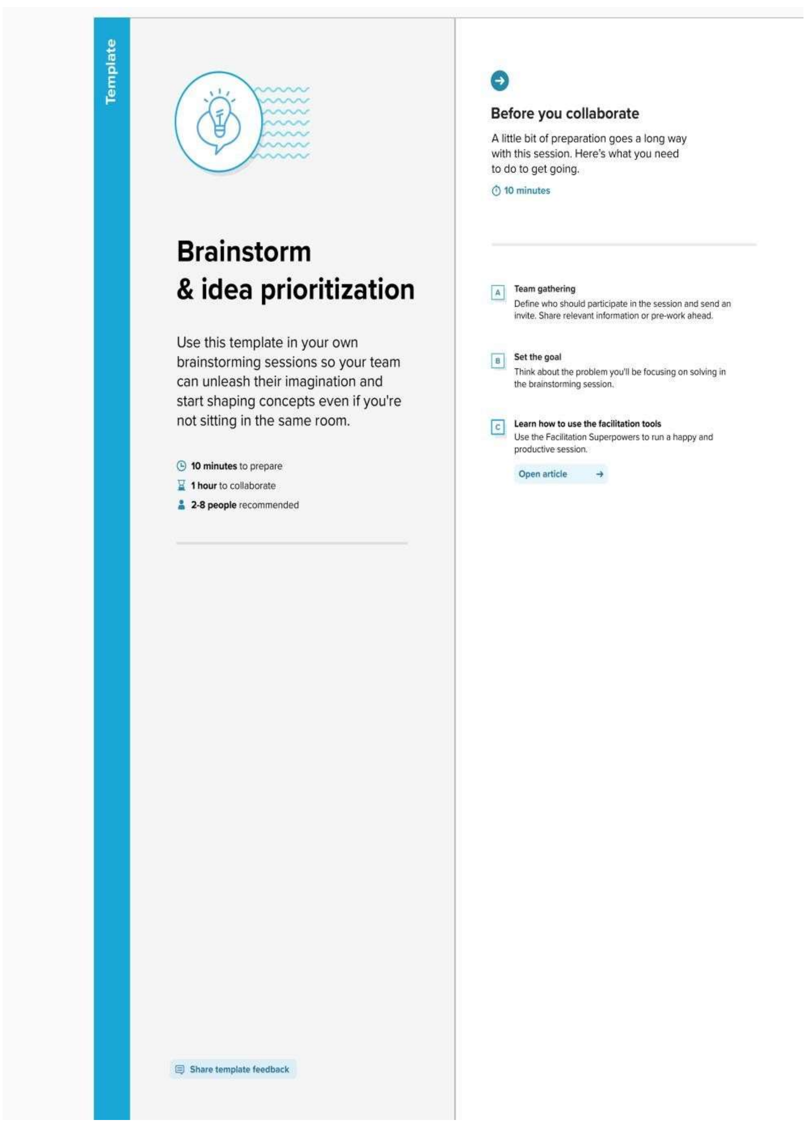
In the ideation face we have empathized as our client and we have acquired the details which are represented in the empathy map given below.



## IDEATION & BRAINSTORMING MAP

**Under this activity our team member have gathered and discussed various ideas to solve our project problem each member contributed 6 to 10 ideas after gathering all ideas we have assessed the impact and feasibility of each point, Finally we have assigned the priority for each point based on this impact values.**

## Step 1: TEAM GATHERING, COLLABRATION AND SELECT THE PROBLEM



1

## Define your problem statement

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

5 minutes

PROBLEM

**How might we predicting personal loan approval using machine learning ?**

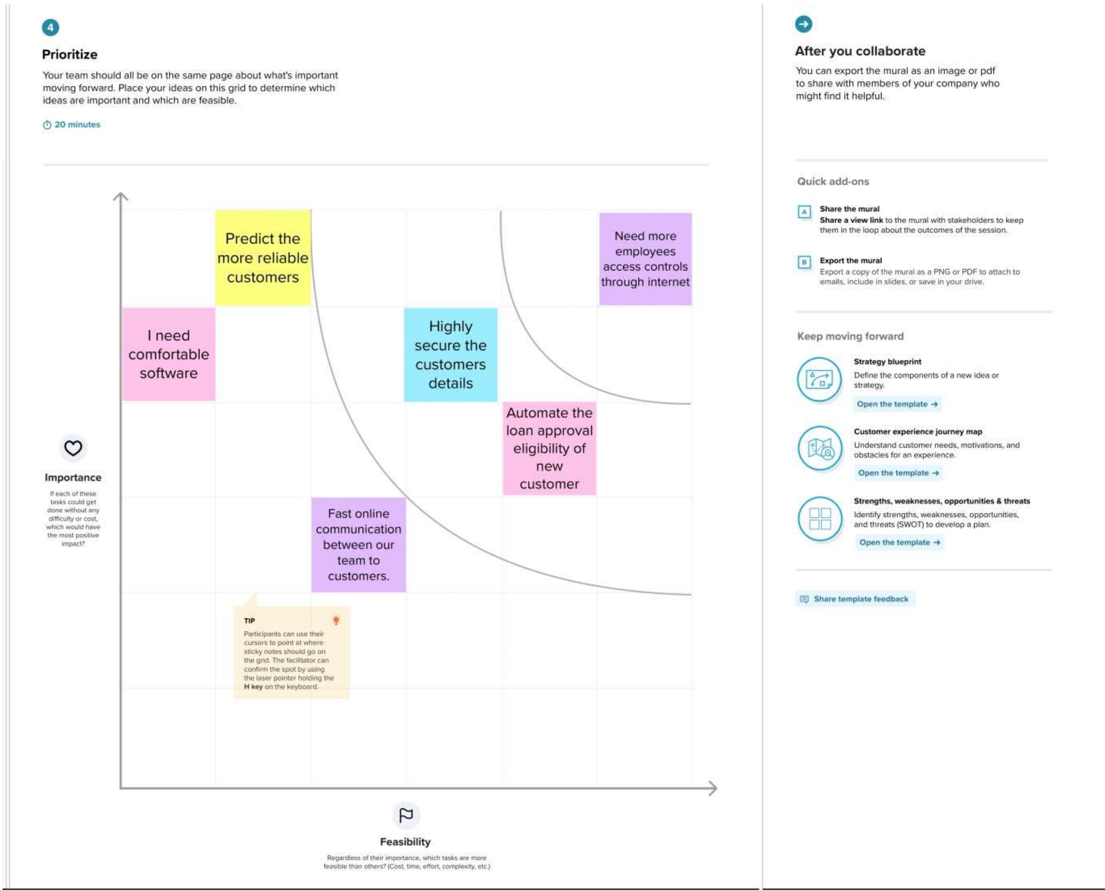
### Key rules of brainstorming

To run an smooth and productive session

- Stay in topic.
- Encourage wild ideas.
- Defer judgment.
- Listen to others.
- Go for volume.
- If possible, be visual.

# STEP2: BRAINSTROM, IDEA LISTING AND GROUPING

# STEP3: IDEA PRIORITIZATION



# 3. RESULT

## READ THE DATASETS

| | 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 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 609 | LP002978 | Female | No | 0 | Graduate | No | 2900 | 0.0 | 71.0 | 360.0 | 1.0 | Rural | Y |
| 610 | LP002979 | Male | Yes | 3+ | Graduate | No | 4106 | 0.0 | 40.0 | 180.0 | 1.0 | Rural | Y |
| 611 | LP002983 | Male | Yes | 1 | Graduate | No | 8072 | 240.0 | 253.0 | 360.0 | 1.0 | Urban | Y |
| 612 | LP002984 | Male | Yes | 2 | Graduate | No | 7583 | 0.0 | 187.0 | 360.0 | 1.0 | Urban | Y |
| 613 | LP002990 | Female | No | 0 | Graduate | Yes | 4583 | 0.0 | 133.0 | 360.0 | 0.0 | Semiurban | N |

614 rows × 13 columns

## HANDLING MISSING VALUES

```
<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
```

```
Index(['Loan_ID', 'Gender', 'Married', 'Dependents', 'Education',
       'Self_Employed', 'Property_Area', 'Loan_Status'],
      dtype='object')
```

| | Loan_ID | Gender | Married | Dependents | Education | Self_Employed | ApplicantIncome | CoapplicantIncome | LoanAmoun |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 5849 | 0.0 | 120. |
| 1 | 1.0 | 1.0 | 1.0 | 1.0 | 0.0 | 0.0 | 4583 | 1508.0 | 128. |
| 2 | 2.0 | 1.0 | 1.0 | 0.0 | 0.0 | 1.0 | 3000 | 0.0 | 66. |

# HANDLING CATEGORICAL VALUES

```
<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    float64
 1   Gender              614 non-null    int64
 2   Married             614 non-null    int64
 3   Dependents          614 non-null    float64
 4   Education           614 non-null    int64
 5   Self_Employed       614 non-null    int64
 6   ApplicantIncome     614 non-null    int64
 7   CoapplicantIncome   614 non-null    int64
 8   LoanAmount          614 non-null    int64
 9   Loan_Amount_Term    614 non-null    int64
 10  Credit_History      614 non-null    int64
 11  Property_Area       614 non-null    float64
 12  Loan_Status         614 non-null    float64
dtypes: float64(4), int64(9)
memory usage: 62.5 KB
```

# HANDLING IMBALANCE DATA

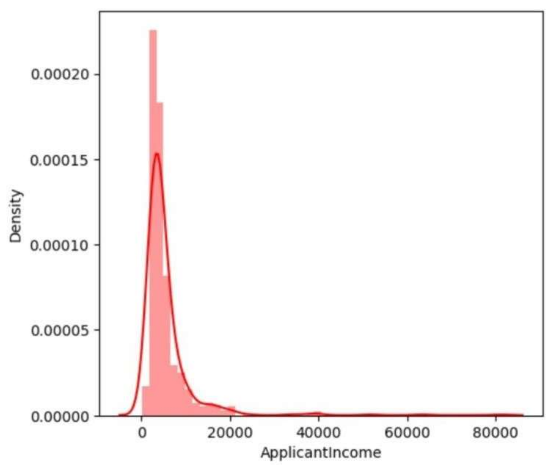
```
1.0    422
0.0    192
Name: Loan_Status, dtype: int64
1.0    366
0.0    366
Name: Loan_Status, dtype: int64
```

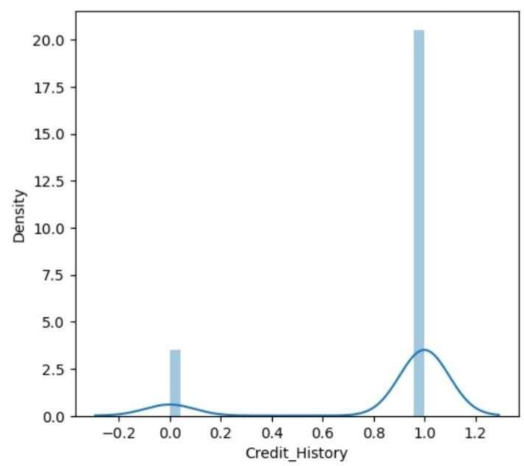
# EXPLORATORY DATA ANALYSIS

## DESCRIPTIVE STATISTICAL

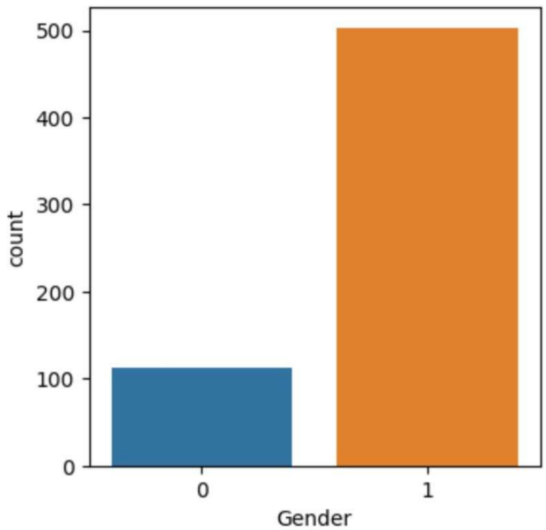
| | Loan_ID | Gender | Married | Dependents | Education | Self_Employed | ApplicantIncome | CoapplicantIncome | LoanAmount | Loan_Amount_Term | Credit_History | Property_Area | Loan_Status |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| count | 614.000000 | 614.000000 | 614.000000 | 614.000000 | 614.000000 | 614.000000 | 614.000000 | 614.00000 | 614.000000 | 614.000000 | 614.000000 | 614.000000 | 614.000000 |
| mean | 306.500000 | 0.817590 | 0.653094 | 0.744300 | 0.218241 | 0.133550 | 5403.459283 | 1621.24430 | 145.465798 | 342.410423 | 0.855049 | 1.037459 | 0.687296 |
| std | 177.390811 | 0.386497 | 0.476373 | 1.009623 | 0.413389 | 0.340446 | 6109.041673 | 2926.24876 | 84.180967 | 64.428629 | 0.352339 | 0.787482 | 0.463973 |
| min | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 150.000000 | 0.00000 | 9.000000 | 12.000000 | 0.000000 | 0.000000 | 0.000000 |
| 25% | 153.250000 | 1.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 2877.500000 | 0.00000 | 100.250000 | 360.000000 | 1.000000 | 0.000000 | 0.000000 |
| 50% | 306.500000 | 1.000000 | 1.000000 | 0.000000 | 0.000000 | 0.000000 | 3812.500000 | 1188.50000 | 125.000000 | 360.000000 | 1.000000 | 1.000000 | 1.000000 |
| 75% | 459.750000 | 1.000000 | 1.000000 | 1.000000 | 0.000000 | 0.000000 | 5795.000000 | 2297.25000 | 164.750000 | 360.000000 | 1.000000 | 2.000000 | 1.000000 |
| max | 613.000000 | 1.000000 | 1.000000 | 3.000000 | 1.000000 | 1.000000 | 81000.000000 | 41667.00000 | 700.000000 | 480.000000 | 1.000000 | 2.000000 | 1.000000 |

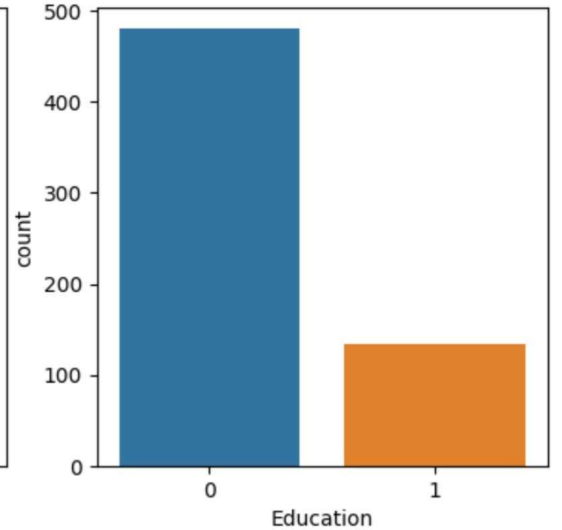
## VISUAL ANALYSIS
## UNIVARIATE ANALYSIS

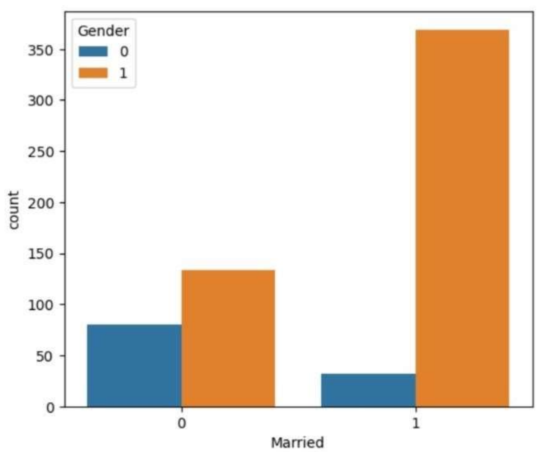


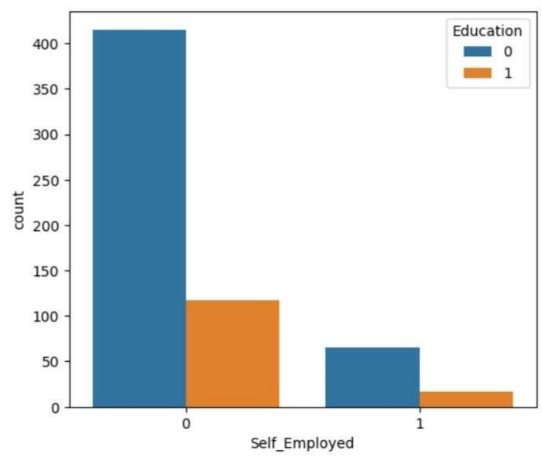



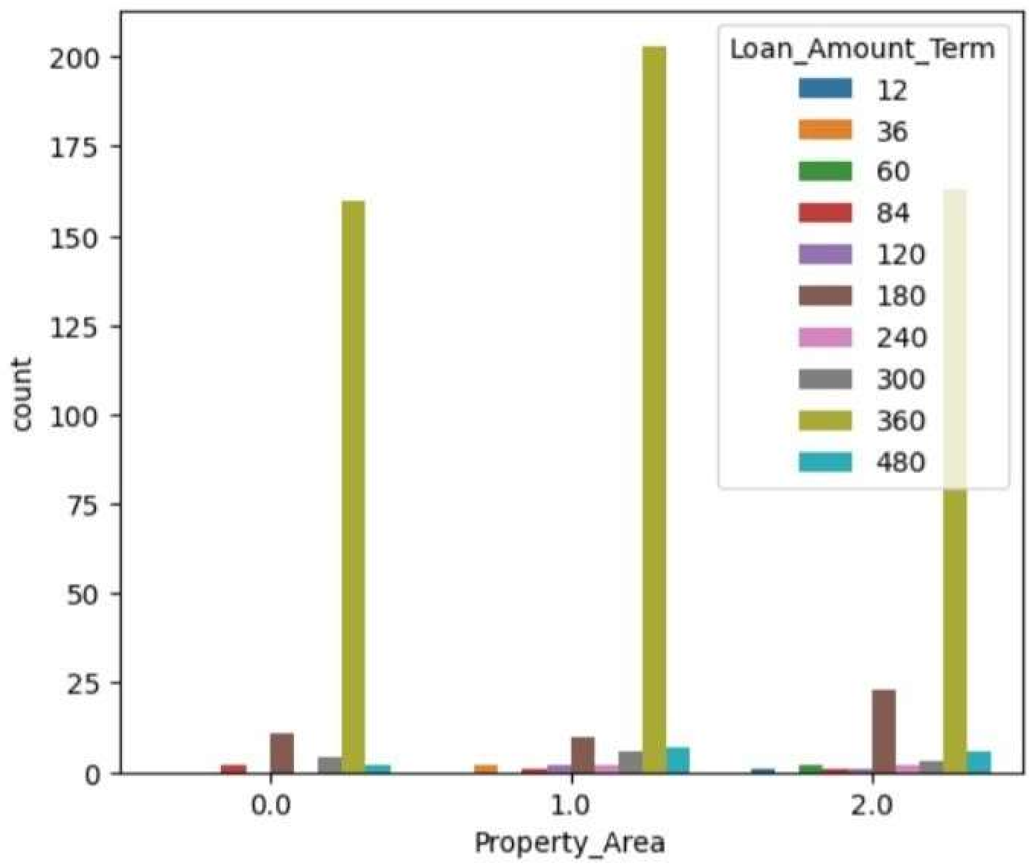
# BIVARIATE ANALYSIS



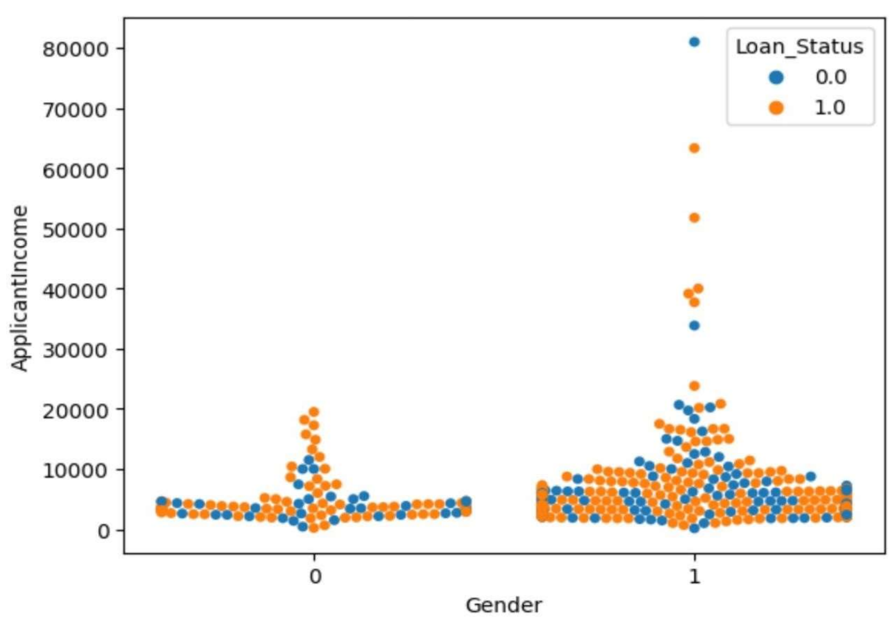






# MULTIVARIATE ANALYSIS



# XGBOOST MODEL

```
 Algorithm is: K Nearest Neighbors
The accuracy is 0.707483:
The Confusion Matrix is: [[43 28]
 [15 61]]
The Classification Report is : ('K Nearest Neighbors', '              precision    recall  f1-score   support\n\n         0.0       0.74      0.61      0.67        71\n         1.0       0.69      0.80      0.74        76\n\n    accuracy

 Algorithm is: Decision Tree
The accuracy is 0.795918:
The Confusion Matrix is: [[56 15]
 [15 61]]
The Classification Report is : ('Decision Tree', '              precision    recall  f1-score   support\n\n         0.0       0.79      0.79      0.79        71\n         1.0       0.80      0.80      0.80        76\n\n    accuracy

 Algorithm is: XGBoost
The accuracy is 0.802721:
The Confusion Matrix is: [[49 22]
 [ 7 69]]
The Classification Report is : ('XGBoost', '              precision    recall  f1-score   support\n\n         0.0       0.88      0.69      0.77        71\n         1.0       0.76      0.91      0.83        76\n\n    accuracy

 Algorithm is: Random Forest
The accuracy is 0.857143:
The Confusion Matrix is: [[56 15]
 [ 6 70]]
The Classification Report is : ('Random Forest', '              precision    recall  f1-score   support\n\n         0.0       0.90      0.79      0.84        71\n         1.0       0.82      0.92      0.87        76\n\n    accuracy
```

# ANN MODEL

```
Epoch 1/100
5/5 [==============================] - 1s 71ms/step - loss: 0.7056 - accuracy: 0.4679 - val_loss: 0.6800 - val_accuracy: 0.5470
Epoch 2/100
5/5 [==============================] - 0s 9ms/step - loss: 0.6564 - accuracy: 0.6774 - val_loss: 0.6374 - val_accuracy: 0.7350
Epoch 3/100
5/5 [==============================] - 0s 10ms/step - loss: 0.6171 - accuracy: 0.7222 - val_loss: 0.5989 - val_accuracy: 0.7863
Epoch 4/100
5/5 [==============================] - 0s 10ms/step - loss: 0.5803 - accuracy: 0.7585 - val_loss: 0.5653 - val_accuracy: 0.8034
Epoch 5/100
5/5 [==============================] - 0s 10ms/step - loss: 0.5459 - accuracy: 0.7821 - val_loss: 0.5337 - val_accuracy: 0.8034
Epoch 6/100
5/5 [==============================] - 0s 10ms/step - loss: 0.5142 - accuracy: 0.8056 - val_loss: 0.5044 - val_accuracy: 0.8034
Epoch 7/100
5/5 [==============================] - 0s 9ms/step - loss: 0.4852 - accuracy: 0.8034 - val_loss: 0.4819 - val_accuracy: 0.8034
Epoch 8/100
5/5 [==============================] - 0s 10ms/step - loss: 0.4604 - accuracy: 0.8141 - val_loss: 0.4657 - val_accuracy: 0.8120
Epoch 9/100
5/5 [==============================] - 0s 11ms/step - loss: 0.4379 - accuracy: 0.8226 - val_loss: 0.4588 - val_accuracy: 0.8205
Epoch 10/100
5/5 [==============================] - 0s 10ms/step - loss: 0.4204 - accuracy: 0.8269 - val_loss: 0.4495 - val_accuracy: 0.8205
Epoch 11/100
5/5 [==============================] - 0s 13ms/step - loss: 0.4053 - accuracy: 0.8333 - val_loss: 0.4427 - val_accuracy: 0.8291
Epoch 12/100
5/5 [==============================] - 0s 10ms/step - loss: 0.3934 - accuracy: 0.8376 - val_loss: 0.4371 - val_accuracy: 0.8291
Epoch 13/100
5/5 [==============================] - 0s 9ms/step - loss: 0.3831 - accuracy: 0.8397 - val_loss: 0.4337 - val_accuracy: 0.8291
```

# TESTING THE MODEL

```
Evaluate model on test data
2/2 [==============================] - 0s 5ms/step - loss: 0.8498 - accuracy: 0.7143
test loss, test acc: [0.8497706651687622, 0.7142857313156128]
Generate a prediction
prediction shape: (1, 1)
```

```
array([[1.42585188e-02],
       [5.22227228e-01],
       [7.19519556e-01],
       [5.47509789e-01],
       [1.45082700e-03],
       [4.63594006e-07],
       [9.02535856e-01],
       [1.23530611e-01],
       [1.81228609e-06],
       [9.33363140e-01],
       [3.97301354e-02],
       [4.01652813e-01],
       [5.30215085e-01],
       [7.53496885e-01],
       [8.49969801e-04],
```

```
array([[False],
       [ True],
       [ True],
       [ True],
       [False],
       [False],
       [ True],
       [False],
       [False],
       [ True],
       [False],
       [False],
       [ True],
       [ True],
       [False],
       [ True],
```

```
     Gender  Married  Dependents  Education  Self_Employed  ApplicantIncome  Coapplican
487       1        1         1.0          0              0            18333
```

```
Prediction: Low chance of Loan approval
/usr/local/lib/python3.9/dist-packages/sklearn/base.py:439: UserWarning: X do
  warnings.warn(
```

# TUNING THE MODEL

# COMPARE THE MODEL

```
 Algorithm is: K Nearest Neighbors
The accuracy is 0.707483:
The Confusion Matrix is:
 [[43 28]
 [15 61]]
The Classification Report is :
              precision    recall  f1-score   support

         0.0       0.74      0.61      0.67        71
         1.0       0.69      0.80      0.74        76

    accuracy                           0.71       147
   macro avg       0.71      0.70      0.70       147
weighted avg       0.71      0.71      0.70       147

-----------------------------------------------------------------------------------------------

 Algorithm is: Decision Tree
The accuracy is 0.768707:
The Confusion Matrix is:
 [[55 16]
 [18 58]]
The Classification Report is :
              precision    recall  f1-score   support

         0.0       0.75      0.77      0.76        71
         1.0       0.78      0.76      0.77        76

    accuracy                           0.77       147
   macro avg       0.77      0.77      0.77       147
weighted avg       0.77      0.77      0.77       147

-----------------------------------------------------------------------------------------------

 Algorithm is: XGBoost
The accuracy is 0.802721:
The Confusion Matrix is:
 [[49 22]
 [ 7 69]]
The Classification Report is :
              precision    recall  f1-score   support
```

```
0.7142857142857143
ANN Model
Confusion_Matrix
[[49 22]
 [20 56]]
Classification Report
              precision    recall  f1-score   support

         0.0       0.71      0.69      0.70        71
         1.0       0.72      0.74      0.73        76

    accuracy                           0.71       147
   macro avg       0.71      0.71      0.71       147
weighted avg       0.71      0.71      0.71       147
```

```
0.8444580839854738
```

```
0.7898840463814475
```

# INTEGRATE WITH WEB FRAMEWORK
# BUILDING HTML PAGE

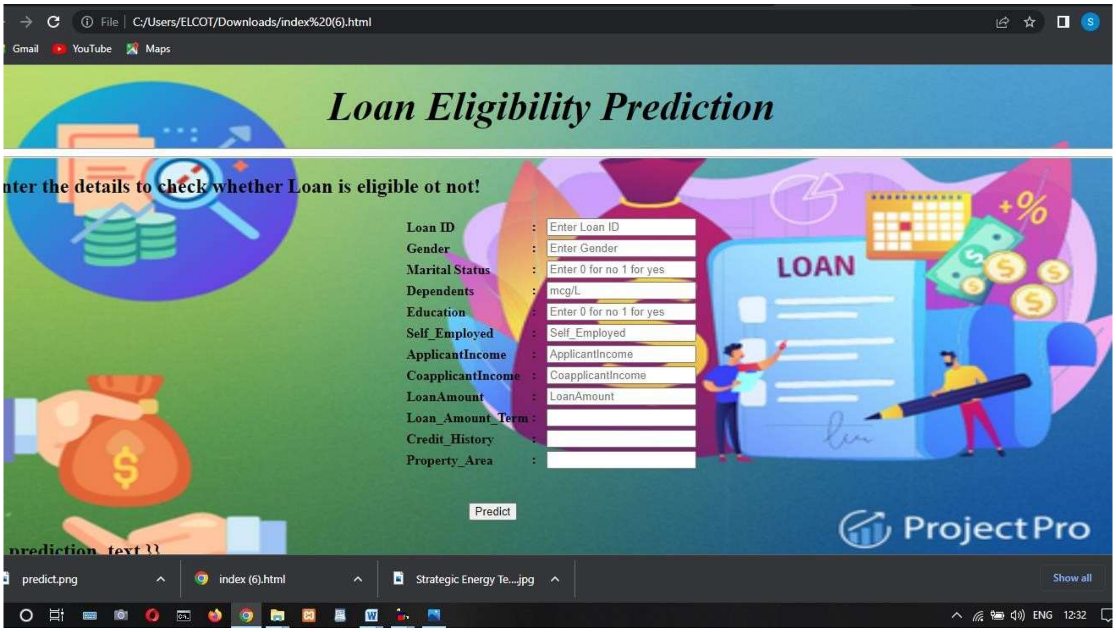
```
<!DOCTYPE html>
<html>
<head>
<title> Loan eligibility prediction</title>
</head>
<body background="predict.png" style="background-repeat:no-repeat; background-
size:100% 100%" text='black'>
<h1>
<b>
<i>
<font size=15>
<center>Loan eligibility Prediction</center>
</font>
</i>
</b>
</h1>
<div style="background-color:white">
<hr>
<hr></div>
<h2> Enter the details to check whether Loan is eligible ot not!</h2>
<h4>
<form action="{{url_for('predict')}}" method="post">
<center>
<table>
<tr>
<td>Loan ID<td>:   <input type='text' name='Loan ID'
placeholder='Enter Loan ID' Enter Numerical part required='required'/><br>
</tr>
<tr>
<td>Gender<td>:   <input type='text' name='Gender'
placeholder='Enter Gender' Enter 0 for Male 1 for Female required='required' /><br>
</tr>
<tr>
<td>Marital Status<td>:   <input type='text' name='Married'
placeholder='Enter 0 for no 1 for yes' required='required'/><br>
</tr>
<tr>
<td>Dependents<td>:   <input type='text' name='Dependents'
placeholder='mcg/L' required='required' /><br>
</tr>
<tr>
<td>Education<td>:   <input type='text' name='Education'
placeholder='Enter 0 for no 1 for yes' required='required' /><br>
</tr>
<tr>
```

```
<td>Self_Employed<td>:   <input type='text' name='Self_Employed '
placeholder='Self_Employed ' required='required' /><br>
</tr>
<tr>
<td>ApplicantIncome<td>:   <input type='text'
name='ApplicantIncome' placeholder='ApplicantIncome' required='required' /><br>
</tr>
<tr>
<td>CoapplicantIncome<td>:   <input type='text'
name='CoapplicantIncome' placeholder='CoapplicantIncome' required='required'/><br>
</tr>
<tr>
<td>LoanAmount<td>:   <input type='text' name='LoanAmount'
placeholder='LoanAmount' required='required' /><br>
</tr>
<tr>
<td>Loan_Amount_Term<td>:   <input type='text'
name='Loan_Amount_Term' placeholder='  ' required='required' /><br>
</tr>
<tr>
<td>Credit_History<td>:   <input type='text' name='Credit_History'
placeholder='  ' required='required' /><br>
</tr>
<tr>
<td>Property_Area<td>:   <input type='text' name='Property_Area'
placeholder='  ' required='required' /><br>
</tr>
<tr>
<td><br><br><center>         &nbs
p     <button type="submit" class="btn btn-primary btn-
block btn-large">Predict</center></button>
</tr>
</form>
</h4>
</table>
</center>
<h2>
<b>
{{ prediction_text }}
</b>
</h2>
</body>
</html>
```

# BUILDING PYTHON CODE

```
import flask
from flask import Flask, render_template, request
import pickle
import numpy as np
import sklearn
from flask_ngrok import run_with_ngrok
import warnings

warnings.filterwarnings('ignore')

app = Flask(_name_)
run_with_ngrok(app)

model = pickle.load(open('rdf.pkl', 'rb'))


@app.route('/', methods=['GET'])
 def home():
    return render_template('index.html')


@app.route('/', methods=['GET', "POST"])
 def predict():
    input_values = [float(x) for x in request.form.values()]
    inp_features = [input_values]
    print(inp_features )
    prediction = model.predict(inp_features)
    if prediction == 1:
        return render_template('index.html', prediction_text='Eligible to loan, Loan will be
sanctioned')
    else:
        return render_template('index.html', prediction_text='Not eligible to loan')


app.run()
```

# RUN THE WEB APPLICATION



# 4. ADVANTAGES & DISADVANTAGES

**Advantages:**

1. Improved accuracy: Machine learning algorithms can analyze a large amount of data and identify patterns that humans may miss, leading to more accurate loan approval predictions.

2. Faster processing time: Automation of the loan approval process can significantly reduce the time it takes to approve or reject a loan application.

3. Reduced manual errors: Automation can reduce the risk of manual errors and ensure consistent decision-making.

4. Increased transparency: Borrowers can better understand the factors that influence loan approval decisions, leading to increased transparency in the lending process.

5. Higher chances of approval: Machine learning algorithms can identify factors that increase the chances of loan approval, leading to higher approval rates for borrowers.

**Disadvantage s:**

1. Limited data: Machine learning models require a large amount of historical data to learn from. If there is limited data available, the model may not be accurate.

2. Biased decision-making: If the historical data used to train the model is biased, the model may make biased decisions, leading to unfair lending practices.

3. Complexity: Machine learning algorithms can be complex and difficult to understand, making it challenging for non-technical users to interpret the results.

4. Lack of human judgment: Machine learning algorithms rely solely on data and may not consider factors that humans would, such as extenuating circumstances or personal relationships.

5. Data privacy concerns: Personal loan applications contain sensitive information, and there may be concerns about how this data is used and protected in a machine learning model.

# 5. APPLICATION

1. Banking and finance: Banks and financial institutions can use machine learning to automate the loan approval process, leading to faster processing times and more accurate decision-making.

2. Fintech startups: Fintech startups can use machine learning to offer personalized loan recommendations to their customers, increasing the chances of approval and improving the customer experience.

3. Credit scoring agencies: Credit scoring agencies can use machine learning to develop more accurate credit scoring models, leading to more precise risk assessments and better lending decisions.

4. Peer-to-peer lending platforms: Peer-to-peer lending platforms can use machine learning to match borrowers with lenders based on their creditworthiness and other factors, leading to more efficient lending.

5. Insurance companies: Insurance companies can use machine learning to predict the likelihood of loan default and adjust their premiums accordingly, leading to more accurate risk assessments and better pricing strategies. 1. Banking and finance: Banks and financial institutions can use machine learning to automate the loan approval process, leading to faster processing times and more accurate decision-making.

6. Traditional banks and financial institutions can use machine learning algorithms to analyze customer data and credit scores to determine the likelihood of loan approval.

7. Online lenders and fintech startups can use machine learning to analyze a borrower's financial history, employment status, and other factors to determine the likelihood of loan approval.

8. Peer-to-peer lending platforms can use machine learning algorithms to match borrowers with lenders based on their creditworthiness and other factors, leading to more efficient lending.

9. Credit unions can use machine learning to analyze member data and credit scores to determine the likelihood of loan approval.

10. Online marketplaces that connect borrowers with lenders can use machine learning to analyze borrower data and credit scores to determine the likelihood of loan approval.

# 6. CONCLUSION

In conclusion, machine learning has revolutionized the process of personal loan approval. Traditional banks, online lenders, fintech startups, credit unions, and peer-to-peer lending platforms can now analyze borrower data and credit scores to determine the likelihood of loan approval. This has led to more efficient and accurate lending, ultimately benefiting both lenders and borrowers. As technology continues to advance, we can expect to see even more innovative uses of machine learning in the financial industry.

Overall, machine learning has transformed the way personal loan approval is conducted. It has enabled lenders to analyze borrower data and credit scores more efficiently and accurately, leading to faster and more reliable lending decisions. This has resulted in benefits for both lenders and borrowers, including improved risk assessment, increased access to credit, and reduced costs. As technology continues to advance, it is likely that machine learning will play an even greater role in the financial industry, leading to further improvements in the lending process.

# 7. FUTURE SCOPE

1. Integration of non-traditional data sources: Currently, personal loan approval models rely heavily on traditional credit bureau data. However, there are many other sources of data that could be integrated into these models, such as social media activity, online shopping behavior, and even biometric data. Incorporating these additional data sources could lead to more accurate risk assessment and better loan approval decisions.

2. Use of deep learning algorithms: Deep learning algorithms can process vast amounts of data and identify complex patterns that may not be apparent to traditional machine learning models. By incorporating deep learning algorithms into personal loan approval models, lenders could improve their accuracy and reduce the risk of default.

3. Real-time decision-making: Currently, most personal loan approval models operate on a batch processing basis, meaning that they analyze data periodically rather than in real-time. By incorporating real-time decision-making capabilities, lenders could make faster and more accurate lending decisions based on the most up-to-date information.

4. Personalized loan products: Machine learning algorithms can also be used to analyze borrower data and identify specific loan products that are best suited to their needs. This could lead to more personalized loan offerings and increased customer satisfaction.

5. Explainable AI: As machine learning models become more complex, it becomes increasingly important to ensure that their decisions can be explained and understood by humans. Explainable AI techniques can be used to provide insight into how machine learning models are making decisions, increasing transparency and trust in the lending process.

6. Integration of alternative credit scoring models: Alternative credit scoring models, such as those based on utility bill payments or rental history, can be integrated into personal loan approval models to provide a more comprehensive view of a borrower's creditworthiness.

7. Integration of natural language processing: Natural language processing can be used to analyze unstructured data sources, such as customer service interactions, to gain additional insights into a borrower's behavior and financial situation.

8. Use of blockchain technology: Blockchain technology can be used to create a secure and transparent lending process, reducing the risk of fraud and increasing trust between lenders and borrowers.

9. Integration of environmental, social, and governance (ESG) criteria: ESG criteria can be integrated into personal loan approval models to ensure that lenders are making socially responsible lending decisions.

10. Collaboration with fintech startups: Collaboration with fintech startups can bring new ideas and technologies to the lending industry, leading to more innovative and effective personal loan approval models.

11. Integration of more data sources: Personal loan approval models can be enhanced by integrating additional data sources, such as social media activity, online purchase history, and mobile phone usage patterns.

12. Use of deep learning algorithms: Deep learning algorithms can be used to analyze complex data sets and identify patterns that may not be visible through traditional machine learning models.

13. Personalized loan offers: Personal loan approval models can be enhanced to provide personalized loan offers based on a borrower's unique financial situation and credit history.

14. Real-time decision making: Personal loan approval models can be enhanced to provide real-time decision making, enabling borrowers to receive loan approvals or denials almost…

# 8. APPENDIX

## A. SOURCE CODE

### Importing the libraries

```
import pandas as pd
import numpy as np
import pickle
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import sklearn
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import GradientBoostingClassifier, RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import RandomizedSearchCV
import imblearn
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, f1_
score
```

### Read the Dataset

```
data = pd.read_csv('/content/drive/MyDrive/PPLA/train_u6lujuX_CVtuZ9i.csv')data
```

### Handling missing values

```
data.info()


data.isnull().sum()
data['Credit_History'] = data['Credit_History'].fillna(data['Credit_History'].median())

data['Self_Employed'] = data['Self_Employed'].fillna(data['Self_Employed'].mode()[0])
data['Dependents'] = data['Dependents'].str.replace('+','')
data['LoanAmount'] = data['LoanAmount'].fillna(data['LoanAmount'].mode()[0])
data['Loan_Amount_Term'] = data['Loan_Amount_Term'].fillna(data['Loan_Amount_Te
rm'].mode()[0]          )
data['Dependents'] = data['Dependents'].fillna(data['Dependents'].mode()[0])
data['Gender'] = data['Gender'].fillna(data['Gender'].mode()[0])
data['Married'] = data['Married'].fillna(data['Married'].mode()[0])

data.info()
```

```
obj_col=data.select_dtypes('object').columns
obj_col

from sklearn.preprocessing import OrdinalEncoder
oe=OrdinalEncoder()
data[obj_col]=oe.fit_transform(data[obj_col])
data.head(3)
```

## Handling Categorical Values

```
data["Gender"]=data["Gender"].astype("int64")
data["Married"]=data["Married"].astype("int64")
data["Self_Employed"]=data["Self_Employed"].astype("int64")
data["Credit_History"]=data["Credit_History"].astype("int64")
data["LoanAmount"]=data["LoanAmount"].astype("int64")
data["Loan_Amount_Term"]=data["Loan_Amount_Term"].astype("int64")
data["Education"]=data[ "Education"].astype("int64")
data["CoapplicantIncome"]=data["CoapplicantIncome"].astype("int64")
data.info()
```

## Handling Imbalance Data

```
from imblearn.combine import*
SEED=2021
smote = SMOTETomek (random_state=SEED)
y=data['Loan_Status']
x=data.drop(columns=['Loan_Status'], axis=1)
x_bal,y_bal=smote.fit_resample(x,y)
print(y.value_counts())
print(y_bal.value_counts())
```

## Descriptive statistical

```
data.describe()
```

## Univariate analysis

```
plt.figure(figsize=(12,5))
plt.subplot(121)
sns.distplot(data['ApplicantIncome'],color='r')
plt.subplot(122)
sns.distplot(data['Credit_History'])
plt.show()
```

## Bivariate analysis

```
plt.figure(figsize=(18,4))
plt.subplot(1,4,1)
sns.countplot(x=data['Gender'])
```

```
plt.subplot(1,4,2)
sns.countplot(x=data['Education'])
plt.show()

plt.figure(figsize=(20,5))
plt.subplot(131)
sns.countplot(x=data['Married'],hue=data['Gender'])
plt.subplot(132)
sns.countplot(x=data['Self_Employed'],hue=data['Education'])
plt.subplot(133)
sns.countplot(x=data['Property_Area'],hue=data['Loan_Amount_Term'])
plt.show()
```

## Multivariate analysis

```
sns.swarmplot(x=data['Gender'],y=data['ApplicantIncome'], hue=data['Loan_Status'])
```

## Scaling the Data

```
names = x.columns
sc=StandardScaler()
x_bal=sc.fit_transform(x_bal)
x_bal=pd.DataFrame(x_bal,columns=names)
```

## Splitting data into train and test

```
X_train, X_test, Y_train, Y_test = train_test_split(x_bal, y_bal, test_size=0.2, random_st
ate=42)
```

## Xgboost model

```
models = []
models.append(('K Nearest Neighbors', KNeighborsClassifier()))
models.append(('Decision Tree', DecisionTreeClassifier()))
models.append(('XGBoost',GradientBoostingClassifier()))
models.append(('Random Forest', RandomForestClassifier()))

for name, algorithm in models:
  model=algorithm
  model.fit(X_train, Y_train)
  prediction = model.predict(X_test)
  print('\n Algorithm is:',name)
  print('The accuracy is %f:'%(accuracy_score(prediction,Y _test)))
  print('The Confusion Matrix is:',(confusion_matrix(Y_test,prediction)))
  print('The Classification Report is :', (name, classification_report(Y_test,preon)))
print('\n')
```

## ANN model

```
import tensorflow as tf
from tensorflow.python import keras
from keras import layers
from keras.layers import Activation,Dense
classifier=keras.Sequential()
classifier.add(Dense(units=100, activation='relu', input_dim=11))
classifier.add(Dense (units=50, activation='relu'))
classifier.add(Dense(units=1, activation='sigmoid'))
classifier.compile(optimizer='adam',loss='binary_crossentropy', metrics=['accuracy'])

model_history = classifier.fit(X_train, Y_train, batch_size=100, validation_split=0.2, epo
chs=100)
```

## Testing the model

```
print("Evaluate model on test data")
results = classifier.evaluate(X_test, Y_test, batch_size=128)
print("test loss, test acc:", results)

# Generate a prediction using model.predict()
# and calculate it's shape:
print("Generate a prediction")
prediction = classifier.predict(X_test[:1])
print("prediction shape:", prediction.shape)

y_pred=classifier.predict(X_test)
y_pred
y_pred=(y_pred>0.5)
y_pred

def predict_x(sample):
  sample=np.array(sample)
  sample=sample.reshape(1,-1)
  sample=sc.transform(sample)
  return classifier.predict(sample)

data1=data.drop('Loan_ID',axis=1)
sample=data1.sample()
sample

if predict_x(sample)>0.5:
  print('Prediction: High chance of Loan approval')
else
  print('Prediction: Low chance of Loan approval')
```

## Compare the model

```
XGB=models[0]
KNN=models[1]
DT=models[2]
RF=models[3]


def compareModel(models):
  for name, algorithm in models:
    model=algorithm
    model.fit(X_train, Y_train)
    prediction = model.predict(X_test)
    print('\n Algorithm is:',name)
    print('The accuracy is %f:'%(accuracy_score(prediction,Y_test)))
    print('The Confusion Matrix is:\n',(confusion_matrix(Y_test,prediction)))
    print('The Classification Report is :\n',classification_report(Y_test,prediction))
    print('-'*100)
compareModel(models)

yPred = classifier.predict(X_test)
print(accuracy_score(y_pred,Y_test))
print("ANN Model")
print("Confusion_Matrix")
print(confusion_matrix(Y_test,y_pred))
print("Classification Report")
print(classification_report(Y_test,y_pred))

from sklearn.model_selection import cross_val_score
rf = RandomForestClassifier()
rf.fit(X_train,Y_train)
yPred = rf.predict(X_test)
f1_score (yPred,Y_test, average='weighted')

cv = cross_val_score(rf,x,y,cv=5)
np.mean(cv)
```