

# Smart Lender - Applicant Credibility Prediction for Loan Approval

**Industry Mentor(s) Name:** Nidhi

**Faculty Mentor(s) Name:** K. S. Guruprakash

**Team Members:** Kiruthik Raj R (811519104057)

Hariharan A (811519104038)

Gokulnath K (811519104034)

Kalai Selvan T S (811519104048)

# **Table Of Contents**

## **1.INTRODUCTION**

- 1.1Project Overview
- 1.2Purpose

## **2.LITERATURE SURVEY**

- 2.1Existing problem
- 2.2References
- 2.3Problem Statement Definition

## **3. IDEATION & PROPOSED SOLUTION**

- 3.1 Empathy Map Canvas
- 3.2 Ideation & Brainstorming
- 3.3 Proposed Solution
- 3.4 Problem Solution fit

## **4. REQUIREMENT ANALYSIS**

- 4.1 Functional requirement
- 4.2 Non-Functional requirements

## **5. PROJECT DESIGN**

- 5.1 Data Flow Diagrams
- 5.2 Solution & Technical Architecture
- 5.3 User Stories

## **6. PROJECT PLANNING & SCHEDULING**

- 6.1 Sprint Planning & Estimation
- 6.2 Sprint Delivery Schedule
- 6.3 Reports from JIRA

## **7. CODING & SOLUTIONING (Explain the features added in the project along with code)**

- 7.1 Feature 1
- 7.2 Feature 2
- 7.3 Feature 3
- 7.4 Feature 4

## **8. TESTING**

- 8.1 Test Cases
- 8.2 User Acceptance Testing

## **9. RESULTS**

- 9.1 Performance Metrics

## **10. ADVANTAGES & DISADVANTAGES**

## **11. CONCLUSION**

## **12. FUTURE SCOPE**

## **13. APPENDIX**

## **Introduction**

### **1.1 Overview**

Nowadays, Banks struggle a lot to get an upper hand over each other to enhance overall business due to tight competition. Banks have realized that retaining the customers and preventing fraud must be the strategy tool for healthy competition. Availability of the huge quantity of data, creation of knowledge base and efficient utilization of the same have helped banks to open up efficient delivery channels. Business decisions can be optimized through data mining . Customer segmentation, banking profitability, credit scoring and approval, predicting payment from customers, marketing, detecting fraud transactions, cash management and forecasting operations, optimizing stock portfolios and ranking investments are some of the areas where data mining techniques can be used in the banking industry. Credit risks which account for the risk of loss and loan defaults are the major source of risk encountered by banking industry. Data mining techniques like classification and prediction can be applied to overcome this to a great extent. Decision Tree Induction Algorithm is one of the best technique to achieve this objective. The model thus developed will provide a better credit risk assessment, which will potentially lead to a better allocation of the banks capital.

In this regard, a study is conducted and an efficient prediction model which helps to reduce the proportion of unsafe borrowers is introduced herewith. Due to the significance of credit risk analysis, this study helps banking industry by providing additional information to the loan decision-making process, potentially decreases the cost and time of loan applications appraisal, and decreases the level of uncertainty for loan officers by providing knowledge extracted from previous loans. Decision Tree Induction Algorithm used in this model is the data mining technique for predicting credible customers. The remaining sections of the paper are organized as

follows: In Section 2, a brief review of some of the related works is presented. Research Methodology, Proposed model and the Architecture of Proposed Model are described in Sections 3, 4 and 5 respectively. The experimental results and the prototype for prediction are given in Section 5. The conclusion and future directions are summed.

## **1.2 Purpose**

To develop a smart lending loan approval or rejection application that can automatically approve or reject a loan based on the user's credit score. This increases the efficiency of the loan approval process and makes it less cumbersome and more economical.

# LITERATURE SURVEY

## 2.1 Existing Problem

With the enhancement in the banking sector, lots of people apply for bank loans but the bank has its limited assets which it grants to only limited people, so finding out to whom the loan can be granted is a typical process for the banks. So, in this paper, they tried to reduce this risk by selecting the safe person so as to save lots of bank efforts and assets. It was done by mining the previous records of the people to whom the loan was granted before and on the basis of these records the machine was trained using the machine learning model which gave the most accurate result. The main goal of this paper is to predict if loan assignment to a specific person will be safe or not. This paper has into four sections (i) Collection of data (ii) Comparing the machine learning models on collected data (iii) Training the system on most promising model (iv) Testing the system

Banks are making major part of profits through loans. Loan approval is a very important process for banking organizations. It is very difficult to predict the possibility of payment of loan by the customers because there is an increasing rate of loan defaults and the banking authorities are finding it more difficult to correctly access loan requests and tackle the risks of people defaulting on loans. In the recent years, many researchers have worked on prediction of loan approval systems. Machine learning technique is very useful in predicting outcomes for large amount of data. In this paper, four algorithms are used such as Random Forest algorithm, Decision Tree algorithm, Naive Bayes algorithm, Logistic Regression algorithm to predict the loan approval of customers. All the four algorithms are going to be used on the same dataset and going to find the algorithm with maximum accuracy to deploy the model. Henceforth, we develop bank loan prediction system using machine learning techniques, so that the system automatically selects the eligible candidates to approve the loan.

A loan is the major source of income for the banking sector of financial risk for banks. Large portions of a bank's assets directly come from the interest earned on loans given. The activity of lending loans carry great risks including the inability of borrower to pay back the loan by the stipulated time. It is referred as "credit risk". A candidate's worthiness for loan approval or rejection was based on a numerical score called "credit score". Therefore, the goal of this paper is to discuss the application of different Machine Learning approach which accurately identifies whom to lend loan to and help banks identify the loan defaulters for much-reduced credit risk.

## 2.2 References

- V. Singh, A. Yadav, R. Awasthi and G. N. Partheeban, "Prediction of Modernized Loan Approval System Based on Machine Learning Approach," 2021 International Conference on Intelligent Technologies (CONIT), 2021, pp. 1-4, doi: 10.1109/CONIT51480.2021.9498475.
- [2] H. Ramachandra, G. Balaraju, R. Divyashree and H. Patil, "Design and Simulation of Loan Approval Prediction Model using AWS Platform," 2021 5 International Conference on Emerging Smart Computing and Informatics (ESCI), 2021, pp. 53-56, doi: 10.1109/ESCI50559.2021.9397049.
- [3] P. Tumuluru, L. R. Burra, M. Loukya, S. Bhavana, H. M. H. CSaiBaba and N. Sunanda, "Comparative Analysis of Customer Loan Approval Prediction using Machine Learning Algorithms," 2022 Second International Conference on Artificial Intelligence and Smart Energy (ICAIS), 2022, pp. 349-353, doi: 10.1109/ICAIS53314.2022.9742800.
- [4] S. Barua, D. Gavandi, P. Sangle, L. Shinde and J. Ramteke, "Swindle: Predicting the Probability of Loan Defaults using CatBoost Algorithm," 2021 5th International Conference on Computing Methodologies and Communication (ICCMC), 2021, pp. 1710-1715, doi: 10.1109/ICCMC51019.2021.9418277.
- [5] V. Singh, A. Yadav, R. Awasthi and G. N. Partheeban, "Prediction of Modernized Loan Approval System Based on Machine Learning Approach," 2021 International Conference on Intelligent Technologies (CONIT), 2021, pp. 1-4, doi: 10.1109/CONIT51480.2021.9498475.
- [6] M. Alaradi and S. Hilal, "Tree-Based Methods for Loan Approval," 2020 International Conference on Data Analytics for Business and Industry: Way Towards a Sustainable Economy (ICDABI), 2020, pp. 1-6, doi: 10.1109/ICDABI51230.2020.9325614

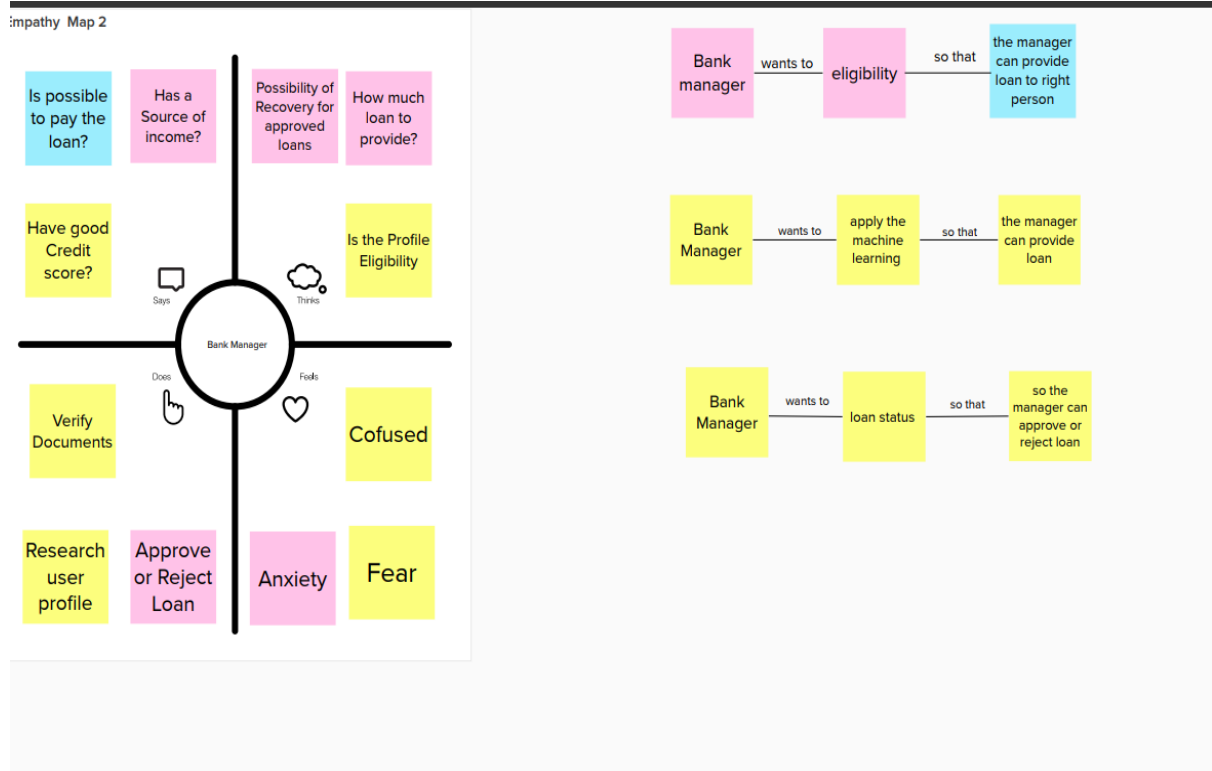
## **2.3 PROBLEM STATEMENT DEFINITION**

- To accurately predict the credibility of a candidate for loan approval.
- Developing a loan approval system that can provide a possible reason for rejection
- Develop a loan approval system that can predict loan approval based on expert data

### 3. IDEATION & PROPOSED SOLUTION


#### 3.1 Empathy Map Canvas

An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviours and attitudes. It is a useful tool to help teams better understand their users. Creating an effective solution requires understanding the true problem and the person who is experiencing it. The exercise of creating the map helps participants consider things from the user's perspective along with his or her goals and challenges.





## 3.2 Ideation & Brainstorming



### Brainstorm & idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

🕒 10 minutes to prepare  
🕒 1 hour to collaborate  
👤 2-8 people recommended

#### Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

🕒 10 minutes

A

**Team gathering**

Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.

B

**Set the goal**

Think about the problem you'll be focusing on solving in the brainstorming session.

C

**Learn how to use the facilitation tools**

Use the Facilitation Superpowers to run a happy and productive session.

[Open article](#) →


#### 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 [your problem statement]?



#### Key rules of brainstorming

To run an smooth and productive session

⬆

Stay in topic.

⬆

Defer judgment.

⬆

Go for volume.

💡

Encourage wild ideas.

👂

Listen to others.

👁

If possible, be visual.

2

### Brainstorm

Write down any ideas that come to mind that address your problem statement.

🕒 10 minutes

#### Kiruthik Raj

- By predicting the loan delinquency, the bank can reduce its Non-Performing Assets
- Eligible applicant for loan can be filtered by cibil score
- Approve loan based on the applicant income
- Verify the loan amount is payable by the applicant based on his income

#### Hariharan

- We the loan based on the movable like car, bike etc
- See the applicant credit card transaction
- If the applicant have working wife and working father.
- Verify the applicant he involved any police issue or violence

#### Kalai Selvan

- If client is married then the loan amount requested is slightly higher than non-married
- If the customer request for the housing loan the area of the house is mentioned properly
- The property area of the customer is mostly equal to the loan amount
- The applicants with very high incomes and co-applied income with a good credit history

#### Gokulnath

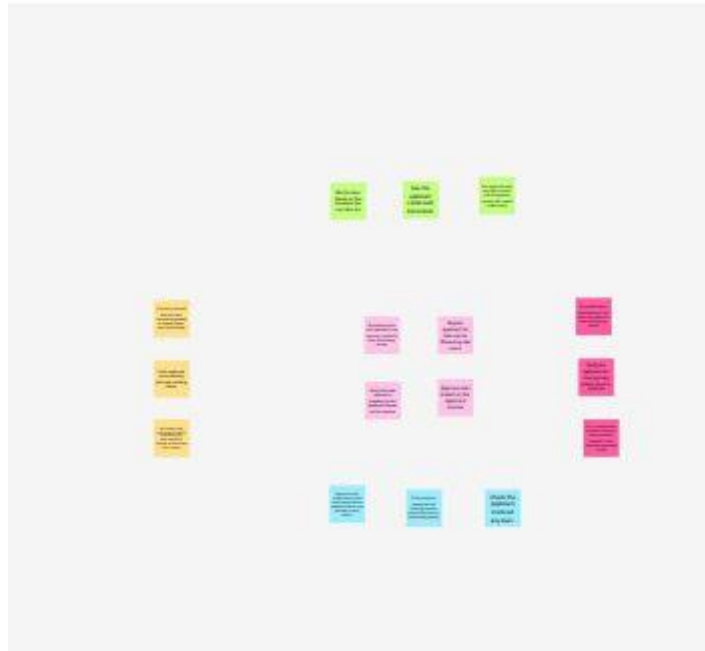
- Approved loans usually have a good credit history, amount requested, and validity in other factors
- The results of the experimental analysis is comparing the loan requested amount are generated in this section
- It is a classification problem where we have to predict whether a loan would be approved or not
- check the applicant involved any loan.

3

**Group ideas**

Take turns sharing your ideas while clustering similar or related notes as you go. In the last 10 minutes, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you can break it up into smaller sub-groups.

⌚ 20 minutes

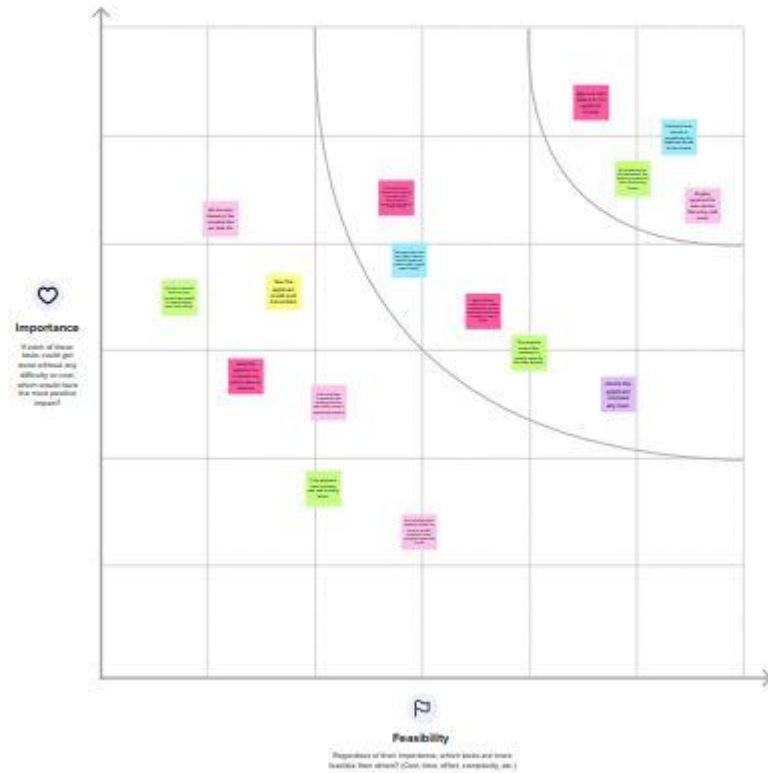


4

### Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

⌚ 30 minutes



### 3.3 Proposed Solution

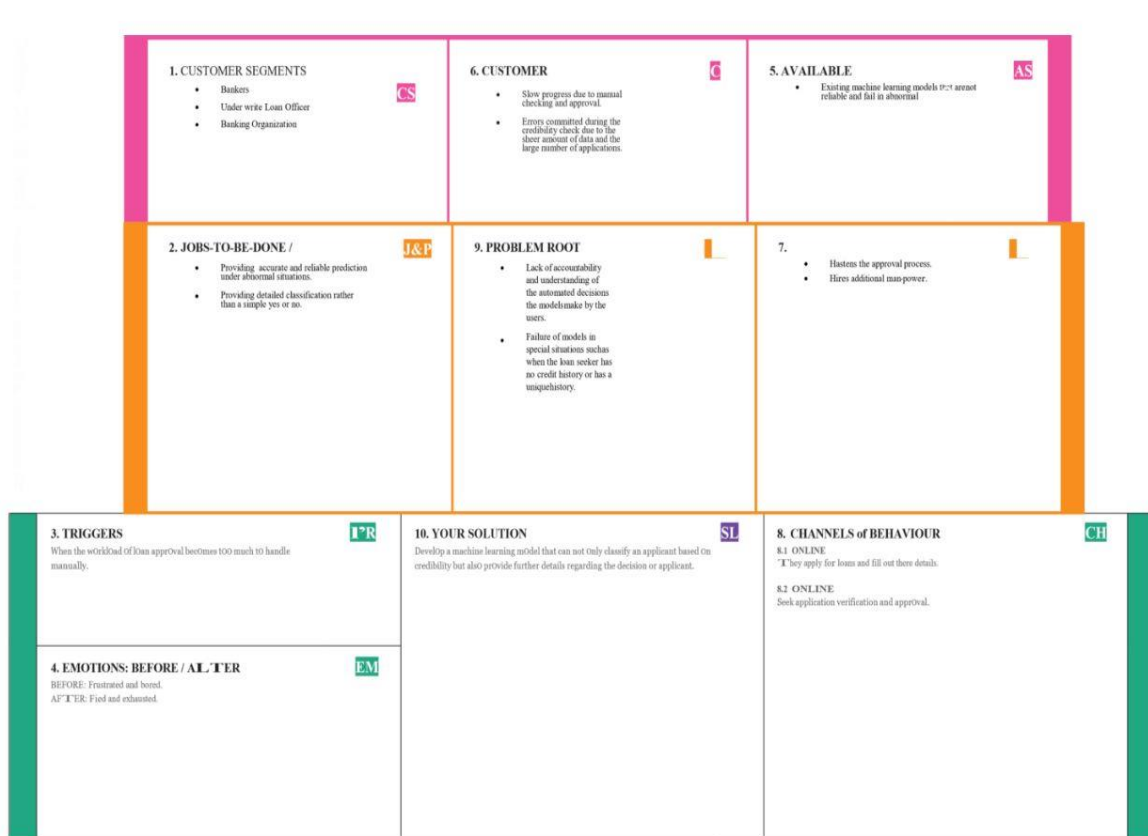
#### Proposed Solution Template:

Project team shall fill the following information in proposed solution template.

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	<p>Understanding the problem statement is the first and foremost step. This would help you give an intuition of what you will face ahead of time. Let us see the problem statement.</p> <p>Dream Housing Finance company deals in all home loans. They have a presence across all urban, semi-urban and rural areas. Customers first apply for a home loan after that company validates the customer's eligibility for a loan. The company wants to automate the loan eligibility process (real-time) based on customer detail provided while filling out the online application form. These details are Gender, Marital Status, Education, Number of Dependents, Income, Loan Amount, Credit History, and others. To automate this process, they have given a problem to identify the customer segments, that are eligible for loan amounts so that they can specifically target these customers.</p>
2.	Idea / Solution description	<p>By predicting the loan defaulters, the bank can reduce its Non- Performing Assets. Eligible applicant for loan can be filtered by cibil score. We the loan based on the movable like car, bike etc. See the applicant credit card transaction. If client is married then the loan amount requested is slightly higher than non-married. If the customer request for the housing loan the area of the house is mentioned properly. Approved loans usually have a good credit history, decent applicant income, and reliability in other factors. The results of the experimental analysis in predicting the loan repayment capacity are presented in this</p>

		section and also The applicants with very high incomes and co-applicant income with a good credit history.
3.	Novelty / Uniqueness	Generally many banks focus on the applicant's property. Whereas we focus on the applicant's income and the family background and also any applicant involved other loan
4.	Social Impact / Customer Satisfaction	This application is working properly and meeting to all Banker requirements. This component can be easily plugged in many other systems. There have been numbers cases of computer glitches, errors in content and most important weight of features is fixed in automated prediction system. This application accuracy level is high. Bank can easily identify the correct applicant and give the loan to the correct applicant.
5.	Business Model (Revenue Model)	Due to the high accuracy, also we spend only limited amount for the application. Many Banks can prefer the our application as we are selling at the low cost with high profit. Using this idea, we can make a profitable revenue
6.	Scalability of the Solution	Our application has better scalability since our analyses all the information provides the best solution. With less chances to get the error. We could achieve the high accuracy

### 3.4 Problem Solution fit



## 4. REQUIREMENT ANALYSIS

FR No.	Non-Functional Requirement	Description
NFR-1	Usability	Simple and understandable UI. Easy to navigate Smooth and seamless Easy to comprehend
NFR-2	Security	Restricted access to data. Login verification Registration verification Upholding privacy of user
NFR-3	Reliability	Backup to prevent data loss Negation of data loss due to lag.

NFR-4	Performance	Web based application. Requires minimum Intel Pentium 4 processor, 4 GB RAM, 1280x1024 screen with application window size 1024x680
-------	-------------	--

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through Gmail Registration through LinkedIn
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	User Application	Filling of application Modification of application Verification of application
FR-4	Loan Issuance	Checking status of loan Loan Approval Loan Rejection
FR-5	Credit history analysis	Credit score auditing Income auditing
FR-6	User management	Choosing appropriate loan program for users Categorising users according to credit history.

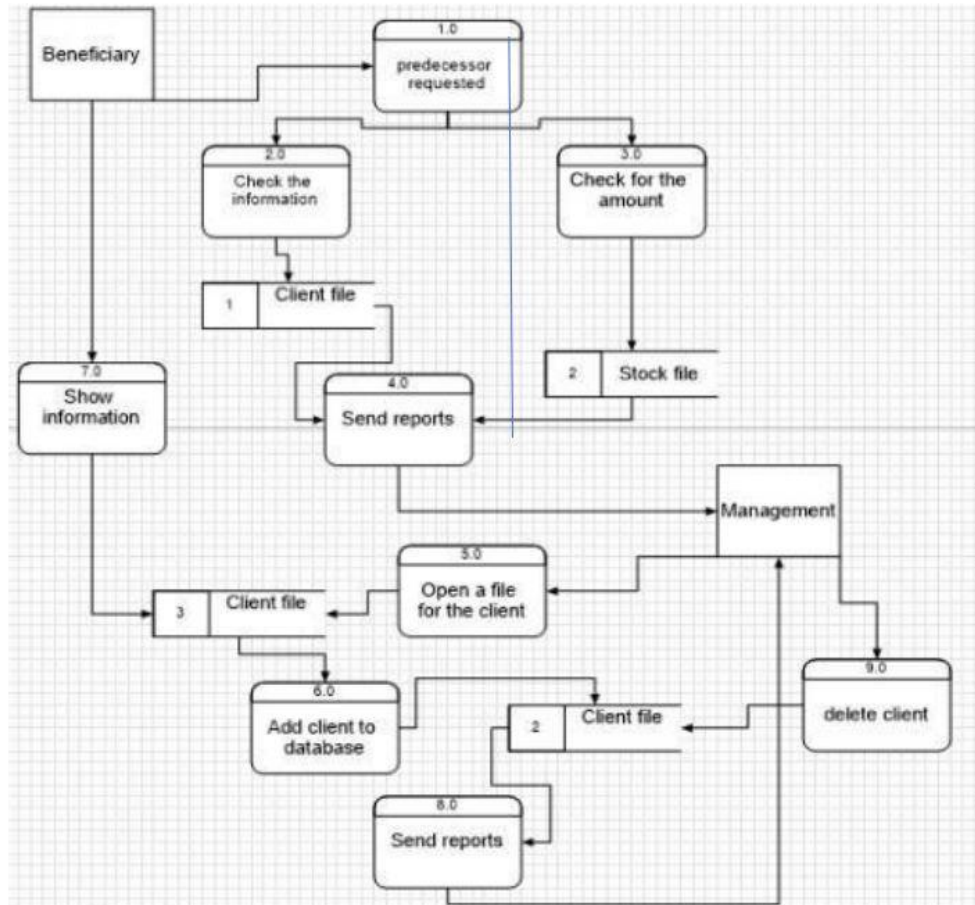
### Non-functional Requirements:

Following are the non-functional requirements of the proposed solution.

NFR-5	Availability	Platform independent support.
NFR-6	Scalability	Can operate efficiently across multiple devices with varying hardware and software specifications.

## 5. PROJECT DESIGN

### 5.1 Data Flow Diagrams





## 5.2 Solution & Technical Architecture

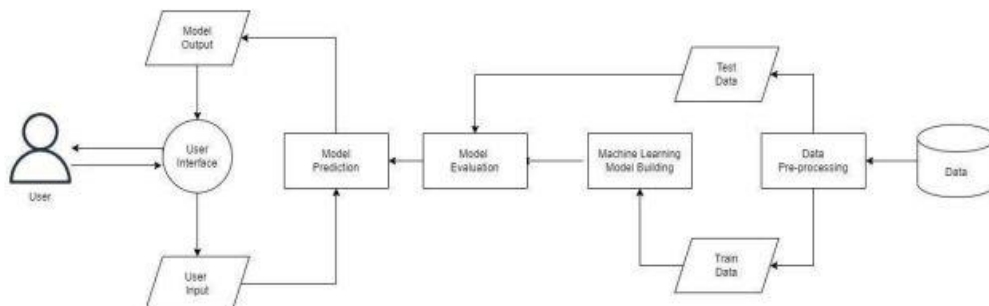
An application should be developed which is able to take the details of the loan applicant, store them and utilize a machine learning model to predict the eligibility for loan approval based on the user application details and credit history. The user should be able to know his/her eligibility upon giving the required details to the application.

### Solution Architecture:

Solution architecture is a complex process – with many sub-processes – that bridges the gap between business problems and technology solutions. Its goals are to:

- Find the best tech solution to solve existing business problems.
- Describe the structure, characteristics, behavior, and other aspects of the software to project stakeholders.
- Define features, development phases, and solution requirements.
- Provide specifications according to which the solution is defined, managed, and delivered.

### Solution Architecture Diagram:



## 5.3 User Stories

### User Stories

Use the below template to list all the user stories for the product.

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
		USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	Sprint-1
		USN-4	As a user, I can register for the application through Gmail	I can receive confirmation email & click confirm	Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password	Able to login	High	Sprint-1
	Dashboard	USN-6	As a user, I should be able to access the dashboard with everything I am allowed to use.	Access the dashboard	Medium	Sprint-1
Customer (Web user)	Registration	USN-7	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
		USN-8	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
		USN-9	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	Sprint-1
		USN-10	As a user, I can register for the application through Gmail	I can receive confirmation email & click confirm	Medium	Sprint-1
	Login	USN-11	As a user, I can log into the application by entering email & password	Able to login	High	Sprint-1

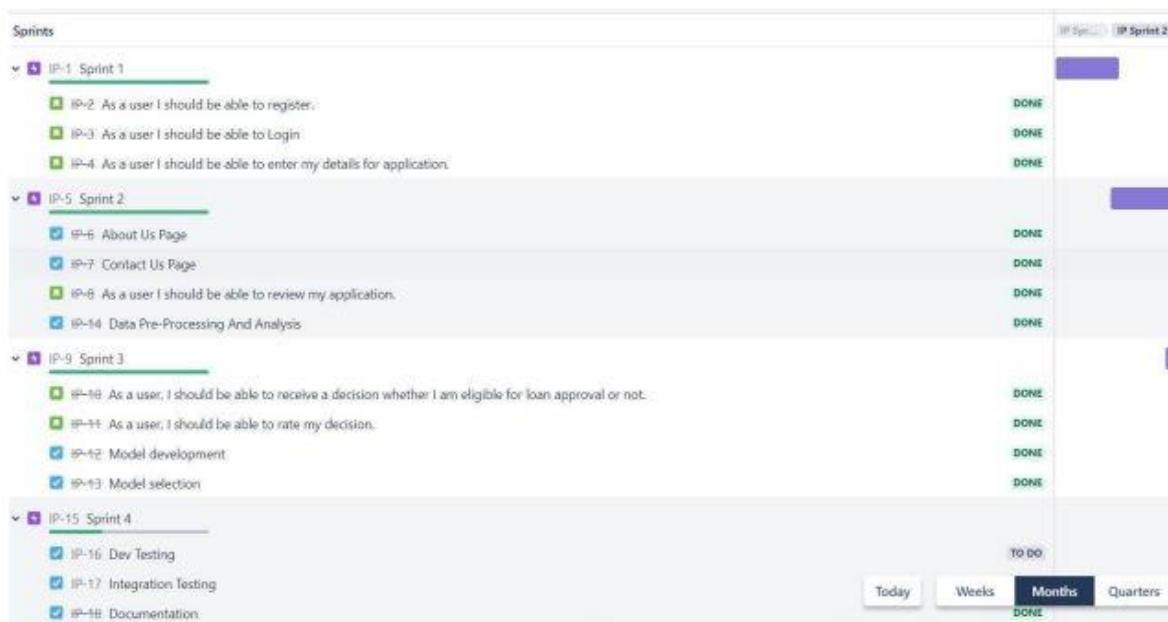
User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
	Dashboard	USN-12	As a user, I should be able to access the dashboard with everything I am allowed to use.	Access the dashboard	Medium	Sprint-1
Loan Approval Officer	Register	USN-13	As a loan approval officer, I should be able to register myself as one using unique email and password.	I can access my account	Medium	Sprint-2
	Login	USN-14	As a loan approval officer I should be able to login myself as one using unique email and password.	Access loan approval dashboard	Medium	Sprint-2
	Automated analysis of credit history	USN-15	As a loan approval officer, I can access the dashboard where I feed application for loan prediction.	I can access the dashboard for loan application prediction.	High	Sprint-3
		USN-16	As a loan approval officer, I can get a decision followed by some details for the decision when I feed an application for loan prediction.	Get a decision for loan prediction with details regarding the decision.	High	Sprint-3
Admin	Register	USN-17	As an admin, I should be able to register myself as one using unique email and password.	I can access my account	Medium	Sprint-4
	Login	USN-18	As an admin I should be able to login myself as one using unique email and password.	Able to login	Medium	Sprint-4
	Dashboard	USN-19	As a admin, I should be able to access the dashboard with everything I am allowed to use.	Access the dashboard	Medium	Sprint-4

## 6. PROJECT PLANNING & SCHEDULING

### 6.1 Sprint Planning & Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Model Building	USN-1	Model Building	2	High	Kiruthik Raj
Sprint-2	HTML webpage	USN-2	Application Building	1	High	Kiruthik Raj Hariharan Kalai Selvan
Sprint-3	Flask API	USN-3	Train the model on IBM	2	Low	Kiruthik Raj Hariharan Kalai Selvan Gokulnath
Sprint-4	Integrate Flask with IBM cloud	USN-4	Integrate flask with scoring end-point	2	Medium	Kiruthik Raj
Sprint-3	HTML webpage	USN-2	Integrate the Flask	1	High	Kiruthik Raj Hariharan
Sprint-2	HTML webpage	USN-2	Enter the input detail	2	Medium	Kiruthik Raj Hariharan

### 6.2 REPORTS FROM JIRA



## CODING AND SOLUTIONING

### 7.1 FEATURE

#### Home.html

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<title>
```

```
    Welcome to Loan Prediction
```

```
</title>
```

```
<style>
```

```
body {
```

```
    background: linear-gradient(-45deg, #ee7752, #e73c7e, #23a6d5, #23d5ab);
```

```
    background-size: 400% 400%;
```

```
    animation: gradient 15s ease infinite;
```

```
    height: 100vh;
```

```
}
```

```
@keyframes gradient {
```

```
    0% {
```

```
        background-position: 0% 50%;
```

```
    }
```

```
    50% {
```

```
        background-position: 100% 50%;
```

```
    }
```

```

100% {
    background-position: 0% 50%;
}
}
</style>
</head>

<h1>Welcome to Loan Prediction</h1>

<h3>Loan Approval is based on lot of this rather than going to a bank and
getting
    rejected. We made it simple that you can get your loan approval prediction
    by our
    machine learning model, for to predict we need some of your
    information</h3>

<h3><center>
<a href="/predict">Predict</a>
</center></h3>

</body>

</html>

```

## 7.2 FEATURE 2

```

<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1">

```

```
<title> Sample Predict </title>
```

```
<style>
```

```
Body {
```

```
    font-family: Calibri, Helvetica, sans-serif;
```

```
    background-color: rgb(130, 172, 107);
```

```
}
```

```
button {
```

```
    background-color: #5fdf63;
```

```
    width: 100%;
```

```
    color: rgb(250, 251, 252);
```

```
    padding: 15px;
```

```
    margin: 10px 0px;
```

```
    border: none;
```

```
    cursor: pointer;
```

```
}
```

```
input {
```

```
/* styling */
```

```
background-color: white;
```

```
border: thin solid rgb(0, 0, 255);
```

```
border-radius: 14px;
```

```
display: inline-block;
```

```
font: inherit;
```

```
line-height: 0.1em;
```

```
padding: 0.5em 3.5em 0.5em 1em;
```

```
/* reset */
```

```
margin: 0;  
-webkit-box-sizing: border-box;  
-moz-box-sizing: border-box;  
box-sizing: border-box;  
-webkit-appearance: none;  
-moz-appearance: none;  
}
```

```
/* arrows */
```

```
select {
```

```
/* styling */  
background-color: rgb(245, 239, 242);  
border: thin solid blue;  
border-radius: 14px;  
display: inline-block;  
font: inherit;  
line-height: 1.5em;  
padding: 0.5em 3.5em 0.5em 1em;
```

```
/* reset */
```

```
margin: 0;
```

```
-webkit-box-sizing: border-box;
-moz-box-sizing: border-box;
box-sizing: border-box;
-webkit-appearance: none;
-moz-appearance: none;
}
```

```
/* arrows */
```

```
select.classic {
background-image:
  linear-gradient(45deg, transparent 50%, rgb(147, 25, 63) 50%),
  linear-gradient(135deg, rgb(68, 198, 57) 50%, transparent 50%),
  linear-gradient(to right, rgb(201, 31, 190), rgb(13, 45, 57));
background-position:
  calc(100% - 20px) calc(1em + 2px),
  calc(100% - 15px) calc(1em + 2px),
  100% 0;
background-size:
  5px 5px,
  5px 5px,
  2.5em 2.5em;
background-repeat: no-repeat;
}
```

```
select.classic:focus {
```



```
background-image:
  linear-gradient(45deg, white 50%, transparent 50%),
  linear-gradient(135deg, transparent 50%, white 50%),
  linear-gradient(to right, gray, gray);
background-position:
  calc(100% - 15px) 1em,
  calc(100% - 20px) 1em,
  100% 0;
background-size:
  5px 5px,
  5px 5px,
  2.5em 2.5em;
background-repeat: no-repeat;
border-color: grey;
outline: 0;
}
```

```
select.round {
background-image:
  linear-gradient(45deg, transparent 50%, gray 50%),
  linear-gradient(135deg, gray 50%, transparent 50%),
  radial-gradient(#ddd 70%, transparent 72%);
background-position:
  calc(100% - 20px) calc(1em + 2px),
```

```
    calc(100% - 15px) calc(1em + 2px),  
    calc(100% - .5em) .5em;  
background-size:  
    5px 5px,  
    5px 5px,  
    1.5em 1.5em;  
background-repeat: no-repeat;  
}
```

```
select.round:focus {  
background-image:  
    linear-gradient(45deg, white 50%, transparent 50%),  
    linear-gradient(135deg, transparent 50%, white 50%),  
    radial-gradient(gray 70%, transparent 72%);  
background-position:  
    calc(100% - 15px) 1em,  
    calc(100% - 20px) 1em,  
    calc(100% - .5em) .5em;  
background-size:  
    5px 5px,  
    5px 5px,  
    1.5em 1.5em;  
background-repeat: no-repeat;  
border-color: green;  
outline: 0;  
}
```

```
select.minimal {  
  background-image:  
    linear-gradient(45deg, transparent 50%, gray 50%),  
    linear-gradient(135deg, gray 50%, transparent 50%),  
    linear-gradient(to right, #ccc, #ccc);  
  background-position:  
    calc(100% - 20px) calc(1em + 2px),  
    calc(100% - 15px) calc(1em + 2px),  
    calc(100% - 2.5em) 0.5em;  
  background-size:  
    5px 5px,  
    5px 5px,  
    1px 1.5em;  
  background-repeat: no-repeat;  
}
```

```
select.minimal:focus {  
  background-image:  
    linear-gradient(45deg, green 50%, transparent 50%),  
    linear-gradient(135deg, transparent 50%, green 50%),  
    linear-gradient(to right, #ccc, #ccc);  
  background-position:  
    calc(100% - 15px) 1em,  
    calc(100% - 20px) 1em,  
    calc(100% - 2.5em) 0.5em;  
  background-size:  
    5px 5px,
```

```
5px 5px,  
1px 1.5em;  
background-repeat: no-repeat;  
border-color: green;  
outline: 0;  
}
```

```
select:-moz-focusing {  
color: transparent;  
text-shadow: 0 0 0 #000;  
}
```

```
form {  
border: 3px solid #db9cea;  
}  
input[type=text], input[type=password]  
{  
width: 100%;  
margin: 8px 0;  
padding: 12px 20px;  
display: inline-block;  
border: 2px solid #d3e970;  
box-sizing: border-box;  
}
```

```
button:hover {  
    opacity: 0.7;  
}
```

```
.cancelbtn {  
    width: auto;  
    padding: 10px 18px;  
    margin: 10px 5px;  
}
```

```
.container {  
    padding: 25px;  
    background-color: rgb(129, 211, 231);  
}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
    <center> <h1> Sample Enter your Details for Loan Approval Prediction  
</h1> </center>
```

```
    <center> <h2>
```

```
    <form action="/output" method="post">
```

```
        <div class="container">
```

```
            <label for="">Choose a Gender</label> <br>
```

```
            <select name="Gender" id="Gendeer" style="width:400px;"  
class="minimal">
```

```
                <option value=1>Men</option>
```

```
                <option value=0>Women</option>
```

```
</select>

<br>

<label for="">Choose a Married</label> <br>

<select name="Married" id="Married" style="width:400px;"
class="minimal">

  <option value=1>Yes</option>

  <option value=0>No</option>

</select>

<br>

<label for="">Choose a Dependents</label> <br>

<select name="Dependents" id="Dependents" style="width:400px;"
class="minimal">

  <option value=0>0</option>

  <option value=1>1</option>

  <option value=2>2</option>

  <option value=3>3</option>

</select>

<br>

<label for="">Choose a Education</label> <br>

<select name="Education" id="Education"
style="width:400px;"class="minimal">

  <option value=0>Graduate</option>

  <option value=1>Not Graduate</option>

</select>

<br>

<label for="">Choose a Self_Employed</label> <br>

<select name="Self_Employed" id="Self_Employed"
style="width:400px;"class="minimal">

  <option value=0>No</option>
```

```
<option value=1>Yes</option>
</select>
<br>
<label>ApplicantIncome </label> <br>
<input type="number" placeholder="Income" name="applicantIncome"
value="4000" style="width:400px;" required>
<br>
<label>CoapplicantIncome</label> <br>
<input type="number" placeholder="Your CoapplicantIncome..."
name="CoapplicantIncome" value="1000" style="width:400px;" required>
<br>
<label>Loan Amount</label> <br>
<input type="number" placeholder="Enter the Loan Amount..."
name="loanamount" value="120" style="width:400px;" required>
<br>
<label for="">Choose a Loan_Amount_Term</label> <br>
<input type="number" placeholder="Enter Loan Amount Term..."
name="Loan_Amount_Term..." value="360" style="width: 400px;" required>
</select>
<br>
<label for="">Choose a Credit_History</label> <br>
<select name="Credit_History" id="Credit_History"
style="width:400px;"class="minimal">
<option value=0>0</option>
<option value=1>1</option>
</select>
<br>
<label for="">Choose a Property_Area</label> <br>
```

```

        <select name="Property_Area" id="Property_Area"
style="width:400px;"class="minimal">
        <option value=0>Urban</option>
        <option value=1>Rural</option>
        <option value=1>Semiurban</option>
        </select>

        <button type="submit">Submit</button>
    </div>
</form> </h2>
</center>
</body>
</html>

```

### 7.3 FEATURE 3

```

<!DOCTYPE html>
<html>

<head>
    <title>Welcome to Loan Prediction</title>
<style>
body {
    background: linear-gradient(-45deg, #ee7752, #e73c7e, #23a6d5, #23d5ab);
    background-size: 400% 400%;
    animation: gradient 15s ease infinite;
    height: 100vh;
}

```



```
@keyframes gradient {
  0% {
    background-position: 0% 50%;
  }

  50% {
    background-position: 100% 50%;
  }

  100% {
    background-position: 0% 50%;
  }
}
</style></head>
<body>
  <center>
    <h1> Loan Approval Prediction</h1>
  </center>
  <br>
  <h3>
    <center>
      {{result}}
    </center>
  </h3>
</body>
</html>
```

## **Model.ipynb**

# Smart Lender

### Import Libraries

```
import pandas as pd
from matplotlib import pyplot as plt
%matplotlib inline
import seaborn as sns
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer
from sklearn.model_selection import train_test_split
import pickle
import numpy as np
from sklearn.metrics import
accuracy_score,classification_report,confusion_matrix,f1_score
from sklearn.preprocessing import StandardScaler
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import RandomizedSearchCV
import numpy
```

### Import Dataset

```
import os, types
```

```

import pandas as pd
from botocore.client import Config
import ibm_boto3

def _iter_(self): return 0

# @hidden_cell

# The following code accesses a file in your IBM Cloud Object Storage. It
includes your credentials.

# You might want to remove those credentials before you share the notebook.
cos_client = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='jbIgUq8OZxntFaOqru6SvInqPaUN8SOBdNN7w1D75-
4Y',
    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3.private.us.cloud-object-storage.appdomain.cloud')

bucket = 'smartlenderapplicantcredibilitypr-donotdelete-pr-ew6lpuekndfnbg'
object_key = 'train_ctrUa4K.csv'

body = cos_client.get_object(Bucket=bucket,Key=object_key)['Body']
# add missing _iter_ method, so pandas accepts body as file-like object
if not hasattr(body, "_iter_"): body.__iter__ = types.MethodType( _iter_, body )

df_data_3 = pd.read_csv(body)
df_data_3.head()
import os, types
import pandas as pd

```

```

from botocore.client import Config
import ibm_boto3

def _iter_(self): return 0

# @hidden_cell

# The following code accesses a file in your IBM Cloud Object Storage. It
includes your credentials.

# You might want to remove those credentials before you share the notebook.
cos_client = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='jbIgUq8OZxntFaOqru6SvInqPaUN8SOBdNN7w1D75-
4Y',
    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3.private.us.cloud-object-storage.appdomain.cloud')

bucket = 'smartlenderapplicantcredibilitypr-donotdelete-pr-ew6lpueknfnbg'
object_key = 'train_ctrUa4K.csv'

body = cos_client.get_object(Bucket=bucket,Key=object_key)['Body']
# add missing _iter_ method, so pandas accepts body as file-like object
if not hasattr(body, "_iter_"): body.__iter_ = types.MethodType( _iter_, body )

df_data_2 = pd.read_csv(body)
df_data_2.head()

import os, types
import pandas as pd
from botocore.client import Config

```

```

import ibm_boto3

def __iter__(self): return 0

# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It
# includes your credentials.
# You might want to remove those credentials before you share the notebook.
cos_client = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='jbIgUq8OZxntFaOqru6SvInqPaUN8SOBdNN7w1D75-
4Y',
    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3.private.us.cloud-object-storage.appdomain.cloud')

bucket = 'smartlenderapplicantcredibilitypr-donotdelete-pr-ew6lpuekndfnbg'
object_key = 'test_1AUu6dG.csv'

body = cos_client.get_object(Bucket=bucket,Key=object_key)['Body']
# add missing __iter__ method, so pandas accepts body as file-like object
if not hasattr(body, "__iter__"): body.__iter__ = types.MethodType( __iter__, body )

df_data_1 = pd.read_csv(body)
df_data_1.head()

data=pd.concat([df_data_1,df_data_2])
data

```

## Univariate Analysis

```
data.Education.value_counts()
```

```
data.columns
```

```
#Univariate Analysis
```

```
plt.figure(figsize=(12,5))
```

```
plt.subplot(121)
```

```
sns.distplot(data['ApplicantIncome'],color='r')
```

```
plt.figure(figsize=(12,5))
```

```
plt.subplot(122)
```

```
sns.distplot(data['Credit_History'])
```

```
plt.show()
```

## Bivariate Analysis

```
#Bivariate Analysis
```

```
plt.figure(figsize=(20,5))
```

```
plt.subplot(131)
```

```
sns.countplot(data['Married'],hue=data['Gender'])
```

```
plt.subplot(132)
```

```
sns.countplot(data['Self_Employed'],hue=data['Education'])
```

```
plt.subplot(133)
```

```
sns.countplot(data['Property_Area'],hue=data['Loan_Amount_Term'])
```

## Multivariate Analysis

#Multivariate Analysis

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

```
data.describe()
```

Perform descriptive statistics on the dataset.

```
data.Credit_History.value_counts()
```

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

```
data.drop(["Loan_ID"],axis=1,inplace=True)
```

```
data
```

Handle the Missing values

```
data["Dependents"].replace({'3+': '3'}, inplace=True)
```

```
data
```

```
data.info()
```

```
for i in [data]:
```

```
    i["Gender"]=i["Gender"].fillna(data.Gender.dropna().mode()[0])
```

```
    i["Married"] = i["Married"].fillna(data.Married.dropna().mode()[0])
```

```

i["Dependents"]=i["Dependents"].fillna(data.Dependents.dropna().mode()[0])

i["Self_Employed"]=i["Self_Employed"].
fillna(data.Self_Employed.dropna().mode()[0])

i["Credit_History"]=i["Credit_History"].fillna(data.Credit_History.dropna().mode()[0])

i["LoanAmount"]=i["LoanAmount"].fillna(data.LoanAmount.dropna().mean())

i["Loan_Amount_Term"]=i["Loan_Amount_Term"].fillna(data.Loan_Amount_Term.dropna().mean())

i["Credit_History"]=i["Credit_History"].fillna(data.Credit_History.dropna().mean())

i["Loan_Status"]=i["Loan_Status"].fillna(data.Loan_Status.dropna().mode()[0])

```

Check for Categorical columns and perform encoding.

```
from sklearn import preprocessing
```

```
# label_encoder object knows how to understand word labels.
```

```
label_encoder = preprocessing.LabelEncoder()
```

```
# Encode labels in column 'species'.
```

```
data["Gender"]= label_encoder.fit_transform(data["Gender"])
```

```
data["Married"]= label_encoder.fit_transform(data["Married"])
```

```
data["Education"]= label_encoder.fit_transform(data["Education"])
```

```
data["Self_Employed"]= label_encoder.fit_transform(data["Self_Employed"])
```

```
data["Credit_History"]= label_encoder.fit_transform(data["Credit_History"])
```



```
data["Property_Area"]= label_encoder.fit_transform(data["Property_Area"])  
data["Loan_Status"]= label_encoder.fit_transform(data["Loan_Status"])
```

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

Split the data into dependent and independent variables

```
x=data.drop(columns=['Loan_Status'],axis=1)
```

```
x
```

```
y=data.iloc[:,11]
```

```
y
```

Scale the independent variables

```
sc=StandardScaler()
```

```
x_bal=sc.fit_transform(x)
```

```
x_bal=pd.DataFrame(x_bal)
```

Split the data into training and testing

```
from sklearn.model_selection import train_test_split
```

```
from sklearn.model_selection import cross_val_score
```

```
x_train,x_test,y_train,y_test=train_test_split(x_bal,y,test_size=0.3)
```

## Build the Model

```
#decision tree
```

```
from sklearn.tree import DecisionTreeClassifier
```

Train the Model and test the Model and Measure the performance using Metrics

```
def decisionTree(x_train,x_test,y_train,y_test):  
    treemodel=DecisionTreeClassifier()  
    treemodel.fit(x_train,y_train)  
    y_pred=treemodel.predict(x_test)  
    cv=cross_val_score(treemodel,x,y,cv=5)  
    return f1_score(y_pred,y_test),np.mean(cv)  
  
print(decisionTree(x_train,x_test,y_train,y_test))
```

```
#Random forest
```

```
rf=RandomForestClassifier()  
rf.fit(x_train,y_train)  
y_pred=rf.predict(x_test)  
cv=cross_val_score(rf,x,y,cv=5)  
print((f1_score(y_pred,y_test),np.mean(cv)))
```

```
#KNeighborsClassifier
```

```
def KNeighbors(x_train,x_test,y_train,y_test):  
    knn_model = KNeighborsClassifier(n_neighbors=5)  
    knn_model.fit(x_train,y_train)  
    yPred = knn_model.predict(x_test)  
    cv=cross_val_score(knn_model,x,y,cv=5)  
    return f1_score(yPred,y_test),np.mean(cv)
```

```
print(KNeighbors(x_train,x_test,y_train,y_test))
```

```
#GradientBoostingClassifier
```

```
def GradientBoosting(x_train,x_test,y_train,y_test):  
    gb=GradientBoostingClassifier()  
    gb.fit(x_train,y_train)  
    yPred=gb.predict(x_test)  
    cv=cross_val_score(gb,x,y,cv=5)  
    return f1_score(yPred,y_test),np.mean(cv)
```

```
print(GradientBoosting(x_train,x_test,y_train,y_test))
```

```
!pip install ibm_watson_machine_learning
```

```
api_key="iFY0zXPpW9WAKkEngxaszWLAMJARuVJzMZgH53lU3o7o"
```

```
location="us-south"
```

```
from ibm_watson_machine_learning import APIClient
```

```
wml_credentials={
```

```
    "apikey":api_key,
```

```
    "url":'https://' + location + '.ml.cloud.ibm.com'
```

```
}
```

```
client=APIClient(wml_credentials)
```

```
def guid_from_space_name(client,space_name):
```

```
    space=client.spaces.get_details()
```

```
    return(next(item for item in space['resources'] if  
item['entity']['name']==space_name)['metadata']['id'])
```

```
space_uid=guid_from_space_name(client,'models')
```

```
print("Space UID=" + space_uid)
```

```
client.set.default_space(space_uid)
```

```
client.software_specifications.list()
```

```
software_spec_uid=client.software_specifications.get_uid_by_name("runtime-  
22.1-py3.9")
```

```
software_spec_uid
```

```
model_details=client.repository.store_model(model=rf,meta_props={
    client.repository.ModelMetaNames.NAME:"Smart_lender",
    client.repository.ModelMetaNames.TYPE:"scikit-learn_1.0",

    client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:software_spec_u
id}

    )
```

```
model_id=client.repository.get_model_uid(model_details)
model_id
```

```
y_pred=rf.predict(x_test)
y_pred
```

```
cv=cross_val_score(rf,x,y,cv=5)
print(f1_score(y_pred,y_test),np.mean(cv))
pickle.dump(rf,open('model.pkl','wb'))
```

```
x=[[1,1,0,0,1,4000,1000,120,360,1,1]]
y=rf.predict(x)
y
```

```
from flask import Flask, render_template, request, jsonify
import pickle
import requests
import json
import numpy as np
import pandas as pd
```

```
API_KEY = "iFY0zXPpW9WAKkEngxaszWLAMJARuVJzMZgH53lU3o7o"
```

```
token_response = requests.post('https://iam.cloud.ibm.com/identity/token',
data={"apikey":API_KEY, "grant_type": 'urn:ibm:params:oauth:grant-
type:apikey'})
```

```
mltoken = token_response.json()["access_token"]
```

```
header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' +
mltoken}
```

```
app=Flask(_name_)
```

```
@app.route('/home')
```

```
def home():
```

```
    return render_template('home.html')
```

```
@app.route('/predict')
```

```
def predict():
```

```
    return render_template('predict.html')
```

```
@app.route('/output',methods=['POST'])
```

```
def output():
```

```
    input_feature=[x for x in request.form.values()]
```

```
    input_feature=[np.array(input_feature)]
```

```
    print(input_feature)
```

```
names=['Gender','Married','Dependents','Education','Self_Employed','ApplicantI
ncome','CoapplicantIncome','LoanAmount','Loan_Amount_Term','Credit_Histo
ry','Property_Area']
```

```
data=pd.DataFrame(input_feature,columns=names)
```

```
print(data)
```

```
response_scoring = requests.post('https://us-
south.ml.cloud.ibm.com/ml/v4/deployments/e858e595-92ed-44da-ba4a-
```

```
f33d452c403c/predictions?version=2022-11-19',  
json=data,headers={'Authorization': 'Bearer ' + mltoken})
```

```
prediction=int(response_scoring.json())
```

```
if(prediction==0):
```

```
    return render_template('submit.html',result="Loan will Not be Approved")
```

```
else:
```

```
    return render_template('submit.html',result="Loan will be Approved")
```

# NOTE: manually define and pass the array(s) of values to be scored in the next line

```
#payload_scoring = {"input_data": [{"field":  
[['Gender','Married','Dependents','Education','Self_Employed','ApplicantIncome'  
, 'CoapplicantIncome','LoanAmount','Loan_Amount_Term','Credit_History','Pro  
perty_Area']],  
  
# "values": [[1,1,0,0,0,4000,1000,120,360,0,0]]}]}
```

## 9.Result

### 9.1 PERFORMANCE METRICS

```
print(classification_report(y_pred,y_test))
```

	precision	recall	f1-score	support
0	0.27	0.62	0.37	26
1	0.96	0.84	0.89	269
accuracy			0.82	295
macro avg	0.61	0.73	0.63	295
weighted avg	0.90	0.82	0.85	295

## 10.ADVANTAGES AND DISADVANTAGES

### 10.1 ADVANTAGES

- The process is now entirely automated.
- Process is streamlined and efficient.
- Prediction have a high degree of accuracy, avoiding errors.
- Predictions are swift and almost instant, reducing time taken to perform the eligibility process.
- Process is decentralized and economical.

### 10.2 DISADVANTAGES

- May give unexpected result for anomalous credit history

CONCLUSION



## CONCLUSION

Loan eligibility determination is a sensitive, vital yet cumbersome process. A lot of resources are drained to ensure the process is continuous without any interruption or error when done manually. With the help of machine learning, the entire process has been fully automated and made more resourceful, economical, convenient and efficient.

## FUTURE SCOPE

- Decentralized integration with banks as an API service.
- Detection of loan frauds.

## APPENDIX

### 13.1 SOURCE CODE

#### Home.html

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<title>
```

```
    Welcome to Loan Prediction
```

```
</title>
```

```
<style>
```

```
body {
```

```
    background: linear-gradient(-45deg, #ee7752, #e73c7e, #23a6d5, #23d5ab);
```

```
    background-size: 400% 400%;
```

```
    animation: gradient 15s ease infinite;
```

```
    height: 100vh;
```

```
}  
@keyframes gradient {  
  0% {  
    background-position: 0% 50%;  
  }  
  
  50% {  
    background-position: 100% 50%;  
  }  
  
  100% {  
    background-position: 0% 50%;  
  }  
}  
</style>  
</head>  
  
<h1>Welcome to Loan Prediction</h1>  
  
<h3>Loan Approval is based on lot of this rather than going to a bank and  
getting  
rejected. We made it simple that you can get your loan approval prediction  
by our  
machine learning model, for to predict we need some of your  
information</h3>  
<h3><center>  
<a href="/predict">Predict</a>  
</center></h3>
```

</body>

</html>

## **Predict.html**

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1">

<title> Sample Predict </title>

<style>

Body {

font-family: Calibri, Helvetica, sans-serif;

background-color: rgb(130, 172, 107);

}

button {

background-color: #5fdf63;

width: 100%;

color: rgb(250, 251, 252);

padding: 15px;

margin: 10px 0px;

border: none;

cursor: pointer;

}

input {

```
/* styling */  
background-color: white;  
border: thin solid rgb(0, 0, 255);  
border-radius: 14px;  
display: inline-block;  
font: inherit;  
line-height: 0.1em;  
padding: 0.5em 3.5em 0.5em 1em;
```

```
/* reset */s
```

```
margin: 0;  
-webkit-box-sizing: border-box;  
-moz-box-sizing: border-box;  
box-sizing: border-box;  
-webkit-appearance: none;  
-moz-appearance: none;  
}
```

```
/* arrows */
```

```
select {
```

```
/* styling */  
background-color: rgb(245, 239, 242);
```

```
border: thin solid blue;
border-radius: 14px;
display: inline-block;
font: inherit;
line-height: 1.5em;
padding: 0.5em 3.5em 0.5em 1em;
```

```
/* reset */
```

```
margin: 0;
-webkit-box-sizing: border-box;
-moz-box-sizing: border-box;
box-sizing: border-box;
-webkit-appearance: none;
-moz-appearance: none;
}
```

```
/* arrows */
```

```
select.classic {
background-image:
  linear-gradient(45deg, transparent 50%, rgb(147, 25, 63) 50%),
  linear-gradient(135deg, rgb(68, 198, 57) 50%, transparent 50%),
  linear-gradient(to right, rgb(201, 31, 190), rgb(13, 45, 57));
background-position:
  calc(100% - 20px) calc(1em + 2px),
```

```
    calc(100% - 15px) calc(1em + 2px),  
    100% 0;  
background-size:  
    5px 5px,  
    5px 5px,  
    2.5em 2.5em;  
background-repeat: no-repeat;  
}
```

```
select.classic:focus {  
background-image:  
    linear-gradient(45deg, white 50%, transparent 50%),  
    linear-gradient(135deg, transparent 50%, white 50%),  
    linear-gradient(to right, gray, gray);  
background-position:  
    calc(100% - 15px) 1em,  
    calc(100% - 20px) 1em,  
    100% 0;  
background-size:  
    5px 5px,  
    5px 5px,  
    2.5em 2.5em;  
background-repeat: no-repeat;  
border-color: grey;  
outline: 0;  
}
```

```
select.round {  
background-image:  
  linear-gradient(45deg, transparent 50%, gray 50%),  
  linear-gradient(135deg, gray 50%, transparent 50%),  
  radial-gradient(#ddd 70%, transparent 72%);  
background-position:  
  calc(100% - 20px) calc(1em + 2px),  
  calc(100% - 15px) calc(1em + 2px),  
  calc(100% - .5em) .5em;  
background-size:  
  5px 5px,  
  5px 5px,  
  1.5em 1.5em;  
background-repeat: no-repeat;  
}
```

```
select.round:focus {  
background-image:  
  linear-gradient(45deg, white 50%, transparent 50%),  
  linear-gradient(135deg, transparent 50%, white 50%),  
  radial-gradient(gray 70%, transparent 72%);  
background-position:  
  calc(100% - 15px) 1em,  
  calc(100% - 20px) 1em,
```

```
    calc(100% - .5em) .5em;
background-size:
    5px 5px,
    5px 5px,
    1.5em 1.5em;
background-repeat: no-repeat;
border-color: green;
outline: 0;
}
```

```
select.minimal {
background-image:
    linear-gradient(45deg, transparent 50%, gray 50%),
    linear-gradient(135deg, gray 50%, transparent 50%),
    linear-gradient(to right, #ccc, #ccc);
background-position:
    calc(100% - 20px) calc(1em + 2px),
    calc(100% - 15px) calc(1em + 2px),
    calc(100% - 2.5em) 0.5em;
background-size:
    5px 5px,
    5px 5px,
    1px 1.5em;
background-repeat: no-repeat;
}
```

```
select.minimal:focus {
```



```
background-image:
  linear-gradient(45deg, green 50%, transparent 50%),
  linear-gradient(135deg, transparent 50%, green 50%),
  linear-gradient(to right, #ccc, #ccc);
background-position:
  calc(100% - 15px) 1em,
  calc(100% - 20px) 1em,
  calc(100% - 2.5em) 0.5em;
background-size:
  5px 5px,
  5px 5px,
  1px 1.5em;
background-repeat: no-repeat;
border-color: green;
outline: 0;
}
```

```
select:-moz-focusing {
color: transparent;
text-shadow: 0 0 0 #000;
}
```

```
form {
  border: 3px solid #db9cea;
```

```
}
input[type=text], input[type=password]
{
    width: 100%;
    margin: 8px 0;
    padding: 12px 20px;
    display: inline-block;
    border: 2px solid #d3e970;
    box-sizing: border-box;
}
button:hover {
    opacity: 0.7;
}
.cancelbtn {
    width: auto;
    padding: 10px 18px;
    margin: 10px 5px;
}

.container {
    padding: 25px;
    background-color: rgb(129, 211, 231);
}
</style>
</head>
<body>
```

```
<center> <h1> Sample Enter your Details for Loan Approval Prediction
</h1> </center>
```

```
<center> <h2>
```

```
<form action="/output" method="post">
```

```
<div class="container">
```

```
<label for="">Choose a Gender</label> <br>
```

```
<select name="Gender" id="Gendeer" style="width:400px;"
class="minimal">
```

```
<option value=1>Men</option>
```

```
<option value=0>Women</option>
```

```
</select>
```

```
<br>
```

```
<label for="">Choose a Married</label> <br>
```

```
<select name="Married" id="Married" style="width:400px;"
class="minimal">
```

```
<option value=1>Yes</option>
```

```
<option value=0>No</option>
```

```
</select>
```

```
<br>
```

```
<label for="">Choose a Dependents</label> <br>
```

```
<select name="Dependents" id="Dependents" style="width:400px;"
class="minimal">
```

```
<option value=0>0</option>
```

```
<option value=1>1</option>
```

```
<option value=2>2</option>
```

```
<option value=3>3</option>
```

```
</select>
```

```
<br>
```

```
<label for="">Choose a Education</label> <br>
<select name="Education" id="Education"
style="width:400px;"class="minimal">
<option value=0>Graduate</option>
<option value=1>Not Graduate</option>
</select>
<br>
<label for="">Choose a Self_Employed</label> <br>
<select name="Self_Employed" id="Self_Employed"
style="width:400px;"class="minimal">
<option value=0>No</option>
<option value=1>Yes</option>
</select>
<br>
<label>ApplicantIncome </label> <br>
<input type="number" placeholder="Income" name="applicantIncome"
value="4000" style="width:400px;" required>
<br>
<label>CoapplicantIncome</label> <br>
<input type="number" placeholder="Your CoapplicantIncome..."
name="CoapplicantIncome" value="1000" style="width:400px;" required>
<br>
<label>Loan Amount</label> <br>
<input type="number" placeholder="Enter the Loan Amount..."
name="loanamount" value="120" style="width:400px;" required>
<br>
<label for="">Choose a Loan_Amount_Term</label> <br>
<input type="number" placeholder="Enter Loan Amount Term..."
name="Loan_Amount_Term..." value="360" style="width: 400px;" required>
```

```

        </select>

        <br>

        <label for="">Choose a Credit_History</label> <br>

        <select name="Credit_History" id="Credit_History"
style="width:400px;"class="minimal">

        <option value=0>0</option>

        <option value=1>1</option>

        </select>

        <br>


        <label for="">Choose a Property_Area</label> <br>

        <select name="Property_Area" id="Property_Area"
style="width:400px;"class="minimal">

        <option value=0>Urban</option>

        <option value=1>Rural</option>

        <option value=1>Semiurban</option>

        </select>


        <button type="submit">Submit</button>

    </div>

</form> </h2>

</center>

</body>

</html>

```

### Submit.html

```
<!DOCTYPE html>
```

```
<html>

<head>
  <title>Welcome to Loan Prediction</title>
<style>
body {
  background: linear-gradient(-45deg, #ee7752, #e73c7e, #23a6d5, #23d5ab);
  background-size: 400% 400%;
  animation: gradient 15s ease infinite;
  height: 100vh;
}
@keyframes gradient {
  0% {
    background-position: 0% 50%;
  }

  50% {
    background-position: 100% 50%;
  }

  100% {
    background-position: 0% 50%;
  }
}
</style></head>

<body>
  <center>
```

```
<h1> Loan Approval Prediction</h1>
</center>
<br>
<h3>
<center>
{{result}}
<center>
</h3>
</body>
</html>
```

## **Model.ipynb**

# Smart Lender

Import Libraries

```
import pandas as pd
from matplotlib import pyplot as plt
%matplotlib inline
import seaborn as sns
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer
from sklearn.model_selection import train_test_split
import pickle
import numpy as np
from sklearn.metrics import
accuracy_score,classification_report,confusion_matrix,f1_score
from sklearn.preprocessing import StandardScaler
```

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import RandomizedSearchCV
import numpy
```

Import Dataset

```
import os, types
import pandas as pd
from botocore.client import Config
import ibm_boto3
```

```
def _iter_(self): return 0
```

```
# @hidden_cell
```

```
# The following code accesses a file in your IBM Cloud Object Storage. It
includes your credentials.
```

```
# You might want to remove those credentials before you share the notebook.
```

```
cos_client = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='jbIgUq8OZxntFaOqru6SvInqPaUN8SOBdNN7w1D75-
4Y',
    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3.private.us.cloud-object-storage.appdomain.cloud')
```



```
bucket = 'smartlenderapplicantcredibilitypr-donotdelete-pr-ew6lpuekndfnbg'
object_key = 'train_ctrUa4K.csv'
```

```
body = cos_client.get_object(Bucket=bucket,Key=object_key)['Body']
# add missing _iter_ method, so pandas accepts body as file-like object
if not hasattr(body, "_iter_"): body.__iter__ = types.MethodType( _iter_, body )
```

```
df_data_3 = pd.read_csv(body)
df_data_3.head()
import os, types
import pandas as pd
from botocore.client import Config
import ibm_boto3
```

```
def _iter_(self): return 0
```

```
# @hidden_cell
```

```
# The following code accesses a file in your IBM Cloud Object Storage. It
includes your credentials.
```

```
# You might want to remove those credentials before you share the notebook.
```

```
cos_client = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='jbIgUq8OZxntFaOqru6SvInqPaUN8SOBdNN7w1D75-4Y',
    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3.private.us.cloud-object-storage.appdomain.cloud')
```

```
bucket = 'smartlenderapplicantcredibilitypr-donotdelete-pr-ew6lpuekndfnbg'
```

```
object_key = 'train_ctrUa4K.csv'
```

```
body = cos_client.get_object(Bucket=bucket,Key=object_key)['Body']
```

```
# add missing _iter_ method, so pandas accepts body as file-like object
```

```
if not hasattr(body, "_iter"): body.__iter__ = types.MethodType( _iter_, body )
```

```
df_data_2 = pd.read_csv(body)
```

```
df_data_2.head()
```

```
import os, types
```

```
import pandas as pd
```

```
from botocore.client import Config
```

```
import ibm_boto3
```

```
def _iter_(self): return 0
```

```
# @hidden_cell
```

```
# The following code accesses a file in your IBM Cloud Object Storage. It  
includes your credentials.
```

```
# You might want to remove those credentials before you share the notebook.
```

```
cos_client = ibm_boto3.client(service_name='s3',
```

```
    ibm_api_key_id='jbIgUq8OZxntFaOqru6SvInqPaUN8SOBdNN7w1D75-  
4Y',
```

```
    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
```

```
    config=Config(signature_version='oauth'),
```

```
    endpoint_url='https://s3.private.us.cloud-object-storage.appdomain.cloud')
```

```
bucket = 'smartlenderapplicantcredibilitypr-donotdelete-pr-ew6lpuekndfnbg'
```

```
object_key = 'test_1AUu6dG.csv'
```

```
body = cos_client.get_object(Bucket=bucket,Key=object_key)['Body']  
# add missing __iter__ method, so pandas accepts body as file-like object  
if not hasattr(body, "__iter__"): body.__iter__ = types.MethodType( __iter__, body )
```

```
df_data_1 = pd.read_csv(body)  
df_data_1.head()
```

```
data=pd.concat([df_data_1,df_data_2])  
data
```

## Univariate Analysis

```
data.Education.value_counts()
```

```
data.columns
```

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

## Bivariate Analysis

#Bivariate Analysis

```
plt.figure(figsize=(20,5))
```

```
plt.subplot(131)
```

```
sns.countplot(data['Married'],hue=data['Gender'])
```

```
plt.subplot(132)
```

```
sns.countplot(data['Self_Employed'],hue=data['Education'])
```

```
plt.subplot(133)
```

```
sns.countplot(data['Property_Area'],hue=data['Loan_Amount_Term'])
```

Multivariate Analysis

#Multivariate Analysis

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

```
data.describe()
```

Perform descriptive statistics on the dataset.

```
data.Credit_History.value_counts()
```

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

```
data.drop(["Loan_ID"],axis=1,inplace=True)
```

```
data
```

Handle the Missing values

```
data["Dependents"].replace({'3+': '3'}, inplace=True)
```

```
data
```

```
data.info()
```

```
for i in [data]:
```

```
    i["Gender"]=i["Gender"].fillna(data.Gender.dropna().mode()[0])
```

```
    i["Married"] = i["Married"].fillna(data.Married.dropna().mode()[0])
```

```
    i["Dependents"]=i["Dependents"].fillna(data.Dependents.dropna().mode()[0])
```

```
    i["Self_Employed"]=i["Self_Employed"].  
    fillna(data.Self_Employed.dropna().mode()[0])
```

```
i["Credit_History"]=i["Credit_History"].fillna(data.Credit_History.dropna().mode()[0])
```

```
i["LoanAmount"]=i["LoanAmount"].fillna(data.LoanAmount.dropna().mean())
```

```
i["Loan_Amount_Term"]=i["Loan_Amount_Term"].fillna(data.Loan_Amount_Term.dropna().mean())
```

```
i["Credit_History"]=i["Credit_History"].fillna(data.Credit_History.dropna().mean())
```

```
i["Loan_Status"]=i["Loan_Status"].fillna(data.Loan_Status.dropna().mode()[0])
```

Check for Categorical columns and perform encoding.

```
from sklearn import preprocessing
```

```
# label_encoder object knows how to understand word labels.
```

```
label_encoder = preprocessing.LabelEncoder()
```

```
# Encode labels in column 'species'.
```

```
data["Gender"]= label_encoder.fit_transform(data["Gender"])
```

```
data["Married"]= label_encoder.fit_transform(data["Married"])
```

```
data["Education"]= label_encoder.fit_transform(data["Education"])
```

```
data["Self_Employed"]= label_encoder.fit_transform(data["Self_Employed"])
```

```
data["Credit_History"]= label_encoder.fit_transform(data["Credit_History"])
```

```
data["Property_Area"]= label_encoder.fit_transform(data["Property_Area"])
```

```
data["Loan_Status"]= label_encoder.fit_transform(data["Loan_Status"])
```

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

Split the data into dependent and independent variables

```
x=data.drop(columns=['Loan_Status'],axis=1)
```

```
x
```

```
y=data.iloc[:,11]
```

```
y
```

Scale the independent variables

```
sc=StandardScaler()
```

```
x_bal=sc.fit_transform(x)
```

```
x_bal=pd.DataFrame(x_bal)
```

Split the data into training and testing

```
from sklearn.model_selection import train_test_split  
from sklearn.model_selection import cross_val_score
```

```
x_train,x_test,y_train,y_test=train_test_split(x_bal,y,test_size=0.3)
```

Build the Model

```
#decision tree
```

```
from sklearn.tree import DecisionTreeClassifier
```

Train the Model and test the Model and Measure the performance using Metrics

```
def decisionTree(x_train,x_test,y_train,y_test):
```

```
    treemodel=DecisionTreeClassifier()
```

```
    treemodel.fit(x_train,y_train)
```

```
    y_pred=treemodel.predict(x_test)
```

```
    cv=cross_val_score(treemodel,x,y,cv=5)
```

```
    return f1_score(y_pred,y_test),np.mean(cv)
```

```
print(decisionTree(x_train,x_test,y_train,y_test))
```

```
#Random forest
```

```
rf=RandomForestClassifier()  
rf.fit(x_train,y_train)  
y_pred=rf.predict(x_test)  
cv=cross_val_score(rf,x,y,cv=5)  
print((f1_score(y_pred,y_test),np.mean(cv)))
```

```
#KNeighborsClassifier
```

```
def KNeighbors(x_train,x_test,y_train,y_test):  
    knn_model = KNeighborsClassifier(n_neighbors=5)  
    knn_model.fit(x_train,y_train)  
    yPred = knn_model.predict(x_test)  
    cv=cross_val_score(knn_model,x,y,cv=5)  
    return f1_score(yPred,y_test),np.mean(cv)  
  
print(KNeighbors(x_train,x_test,y_train,y_test))
```



```
#GradientBoostingClassifier
```

```
def GradientBoosting(x_train,x_test,y_train,y_test):
```

```
    gb=GradientBoostingClassifier()
```

```
    gb.fit(x_train,y_train)
```

```
    yPred=gb.predict(x_test)
```

```
    cv=cross_val_score(gb,x,y,cv=5)
```

```
    return f1_score(yPred,y_test),np.mean(cv)
```

```
print(GradientBoosting(x_train,x_test,y_train,y_test))
```

```
!pip install ibm_watson_machine_learning
```

```
api_key="iFY0zXPpW9WAKkEngxaszWLAMJARuVJzMZgH53lU3o7o"
```

```
location="us-south"
```

```
from ibm_watson_machine_learning import APIClient
```

```
wml_credentials={
```

```
    "apikey":api_key,
```

```
    "url":'https://' + location + '.ml.cloud.ibm.com'
```

```
}
```

```
client=APIClient(wml_credentials)
```

```
def guid_from_space_name(client,space_name):
```

```
    space=client.spaces.get_details()
```

```
    return(next(item for item in space['resources'] if  
item['entity']['name']==space_name)['metadata']['id'])
```

```
space_uid=guid_from_space_name(client,'models')
print("Space UID=" + space_uid)

client.set.default_space(space_uid)

client.software_specifications.list()

software_spec_uid=client.software_specifications.get_uid_by_name("runtime-
22.1-py3.9")
software_spec_uid

model_details=client.repository.store_model(model=rf,meta_props={
    client.repository.ModelMetaNames.NAME:"Smart_lender",
    client.repository.ModelMetaNames.TYPE:"scikit-learn_1.0",
    client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:software_spec_u
id}

)

model_id=client.repository.get_model_uid(model_details)
model_id

y_pred=rf.predict(x_test)
y_pred

cv=cross_val_score(rf,x,y,cv=5)
print(f1_score(y_pred,y_test),np.mean(cv))
pickle.dump(rf,open('model.pkl','wb'))
```

```
x=[[1,1,0,0,1,4000,1000,120,360,1,1]]
```

```
y=rf.predict(x)
```

```
y
```

```
from flask import Flask, render_template, request, jsonify
```

```
import pickle
```

```
import requests
```

```
import json
```

```
import numpy as np
```

```
import pandas as pd
```

```
API_KEY = "iFY0zXPpW9WAKkEngxaszWLAMJARuVJzMZgH53lU3o7o"
```

```
token_response = requests.post('https://iam.cloud.ibm.com/identity/token',  
data={"apikey":API_KEY, "grant_type": 'urn:ibm:params:oauth:grant-  
type:apikey'})
```

```
mltoken = token_response.json()["access_token"]
```

```
header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' +  
mltoken}
```

```
app=Flask(__name__)
```

```
@app.route('/home')
```

```
def home():
```

```
    return render_template('home.html')
```

```
@app.route('/predict')
```

```
def predict():
```

```
    return render_template('predict.html')
```

```
@app.route('/output',methods=['POST'])
```

```

def output():
    input_feature=[x for x in request.form.values()]
    input_feature=[np.array(input_feature)]
    print(input_feature)

names=['Gender','Married','Dependents','Education','Self_Employed','ApplicantIncome','CoapplicantIncome','LoanAmount','Loan_Amount_Term','Credit_History','Property_Area']

data=pd.DataFrame(input_feature,columns=names)
print(data)

response_scoring = requests.post('https://us-south.ml.cloud.ibm.com/ml/v4/deployments/e858e595-92ed-44da-ba4a-f33d452c403c/predictions?version=2022-11-19',
json=data,headers={'Authorization': 'Bearer ' + mltoken})

prediction=int(response_scoring.json())
if(prediction==0):
    return render_template('submit.html',result="Loan will Not be Approved")

else:
    return render_template('submit.html',result="Loan will be Approved")

# NOTE: manually define and pass the array(s) of values to be scored in the
next line

#payload_scoring = {"input_data": [{"field":
[['Gender','Married','Dependents','Education','Self_Employed','ApplicantIncome'

```

```
, 'CoapplicantIncome', 'LoanAmount', 'Loan_Amount_Term', 'Credit_History', 'Property_Area']],
```

```
# "values": [[1,1,0,0,0,4000,1000,120,360,0,0]]]}
```

## **13.2 GITHUB LINK**

<https://github.com/IBM-EPBL/IBM-Project-9118-1658981443>