PROJECT-BASED EXPERIENTIAL LEARNING PROGRAM
(NALAIYA THIRAN)

PROJECT REPORT

SMART LENDER - Applicant Credibility Prediction for Loan Approval

TEAM MEMBERS

1. Adwaid Babu - 113119UG04006
2. Ahmed Rashid.A - 113119UG04008
3. Abhishek Yadav.R - 113119UG04002
4. Ajith.R - 113119UG04009

INDEX

1. INTRODUCTION
Purpose
2. LITERATURE SURVEY
2.1 Existing Problem
2.2 References
3. IDEATION & PROPOSED SOLUTION
3.1 Empathy Map Canvas
3.2 Proposed Solution
3.3 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
7.1 Feature 1
7.2 Feature 2
8. TESTING
8.1 Test Cases
8.2 User Acceptance Testing
9. RESULTS
Performance Metrics
10. ADVANTAGES & DISADVANTAGES
11. CONCLUSION
12. FUTURE SCOPE
13. APPENDIX
Source Code
GitHub & Project Demo Link

1. Introduction

The two most pressing issues in the banking sector are:
1) How risky is the borrower?
2) Should we lend to the borrower given the risk?

The response to the first question dictates the borrower's interest rate. Interest rate, among other things (such as time value of money), tests the riskiness of the borrower, i.e. the higher the interest rate, the riskier the borrower. We will then decide whether the applicant is suitable for the loan based on the interest rate. Lenders (investors) make loans to creditors in return for the guarantee of interest-bearing repayment. That is, the lender only makes a return (interest) if the borrower repays the loan. However, whether he or she does not repay the loan, the lender loses money. Banks make loans to customers in exchange for the guarantee of repayment. Some would default on their debts, unable to repay them for a number of reasons. The bank retains insurance to minimize the possibility of failure in the case of a default. The insured sum can cover the whole loan amount or just a portion of it. Banking processes use manual procedures to determine whether or not a borrower is suitable for a loan based on results. Manual procedures were mostly effective, but they were insufficient when there were a large number of loan applications. At that time, making a decision would take a long time. As a result, the loan prediction machine learning model can be used to assess a customer's loan status and build strategies. This model extracts and introduces the essential features of a borrower that influence the customer's loan status. Finally, it produces the planned performance (loan status). These reports make a bank manager's job simpler and quicker.

PURPOSE

A loan is the core business part of banks. The main portion of the bank's profit directly comes from the profit earned from the loans. Though the bank approves a loan after a regress process of verification and testimonial, still there's no surety whether the chosen hopeful is the right hopeful or not. This process takes new time while doing it manually. We can prophesy whether that
particular hopeful is safe, and the whole testimonial process is automated by machine literacy style. Loan Prognostic is helpful for the retainer of banks as well as for the hopeful also.

However, developing such a model is challenging due to the increased demand for loans. The organizations can use prototypes of the model to make the correct decision to approve or reject the request for the loan customers. This work includes constructing an ensemble model by combining different machine-learning models. Banks struggle to get the upper hand over each other to enhance overall business due to tight competition. Credit Risk assessment is a crucial issue faced by Banks nowadays, which helps them to evaluate if a loan application can be a defaulter at a later stage so that they can go ahead and grant the loan or not. This helps the banks minimize possible losses and increase the volume of credits.

2. Literature Survey

2.1 Existing Problem

Bank employees manually check applicants' details and give an eligible applicant the loan. Checking the details of all applicants takes a lot of time.

Checking the details of all applicants consumes a lot of time and effort. There are chances that human error may occur due to checking all details manually. There is a possibility of assigning loans to an ineligible applicant

2.2 REFERENCES

References were made from the articles and papers and the following has been derived,

S.NO	TITLE	AUTHOR	DESCRIPTION
1.	"Loan Approval Prediction"	Shubham Nalawade, Suraj Andhe, Siddhesh Parab, Prof. Amruta Sankhe - Information Technology, Atharva College of Engineering, Mumbai	These most important features are then used on some selected algorithms, and their performance accuracy is compared with the instance of using all the features. This model can help banks determine which factors are essential for the loan approval procedure. The comparative study clarifies which algorithm will be the best and ignores the rest based on their accuracy.
2.	"Survey on Prediction of Loan Approval Using Machine Learning Techniques"	Ambika and Santosh Biradar/ Department of Computer Engineering, D. Y. Patil College of Engineering, Pune, India	The system is trained on old training datasets in future software that can be made such that new testing dates should also take part

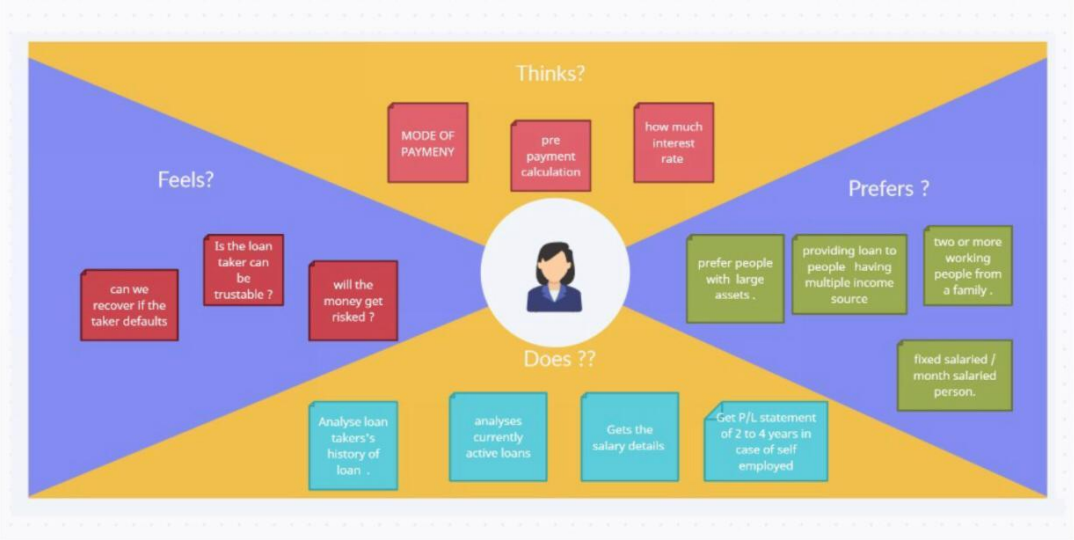
			in training data after some fixed time. Machine learning helps to understand the factors which affect the specific outcomes most. Other models, like neural networks and discriminate analysis, can be used individually or combined for enhanced reliability and accurate prediction.
3.	"Process Evaluation and Improvement: A Case Study of The Loan Approval Process"	MAJAPUSNIK, KATJAKOUS, ANDREJ GODEC and BOASTJAN SUMAK, University of Maribor	As a proposal for optimization, using advanced IT support can optimize the credit approval process to some extent. However, traditional systems like banks are less prone to change and use few available possibilities. Within the research, some potentially replaceable activities were highlighted where they could be included.
4.	Loan Approval Prediction based on Machine Learning"	Kumar Arun, Garg Ishan, Kaur Sanmeet	There have been numerous cases of computer glitches and errors in content. The most important weight of features is fixed in automated prediction systems; in the near future, the so-called software could be made a more

			secure, reliable, and dynamic adjustment. In the near future, this prediction module can be integrated with the module of an automated processing system. The system is trained on old training datasets in future software that can be made such that new testing dates should also take part in training data after some fixed time.
5.	Predict Loan Approval in Banking System Machine Learning Approach for Cooperative Banks Loan Approval"	Amruta S. Aphale, Prof. Dr. Sandeep. R. Shinde Department of Computer Science and Engineering Savitribai Phule Pune University Vishwakarma Institute of Technology, Pune	These most important features are then used on some selected algorithms, and their performance accuracy is compared with the instance of using all the features. This model can help banks determine which factors are essential for the loan approval procedure. The comparative study clarifies which algorithm will be the best and ignores the rest, accuracy.

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
behaviors and attitudes.
It is a valuable tool to help teams better understand their users.
Creating an effective solution requires understanding the problem and the person experiencing it.
Creating the map helps participants consider things from the user's perspective, along with their
goals and challenges.



3.2 PROPOSED SOLUTION

In our proposed system, we combine datasets from different sources to form a generalized data set and use four machine learning algorithms, such as Random Forest, Logistic regression, Decision tree, and Naive Bayes algorithm, on the same data set. The data set we collected for predicting given data is split into the training set and a test set in the ratio of 8:2. The data model, which was created using Machine learning algorithms, applied to the training set and based on a maximum test result
from the four algorithms, the test set prediction is made using the algorithm that has maximum performance. After that, we deploy the model using Flask Framework.
We developed automatic loan prediction using machine learning techniques to deal with the problem. We will train the machine with the previous data set. So machines can analyze and understand the process. Then the machine will check for eligible applicants and give us the result.
Advantages: The time period for loan sanctioning will be reduced. The whole process will be automated so that human error will be avoided. Eligible applicants will be sanctioned loans without any delay.

SOLUTION DESCRIPTION:
Creating a Machine Learning Model and Website to check whether the individual is eligible or not, paperless.

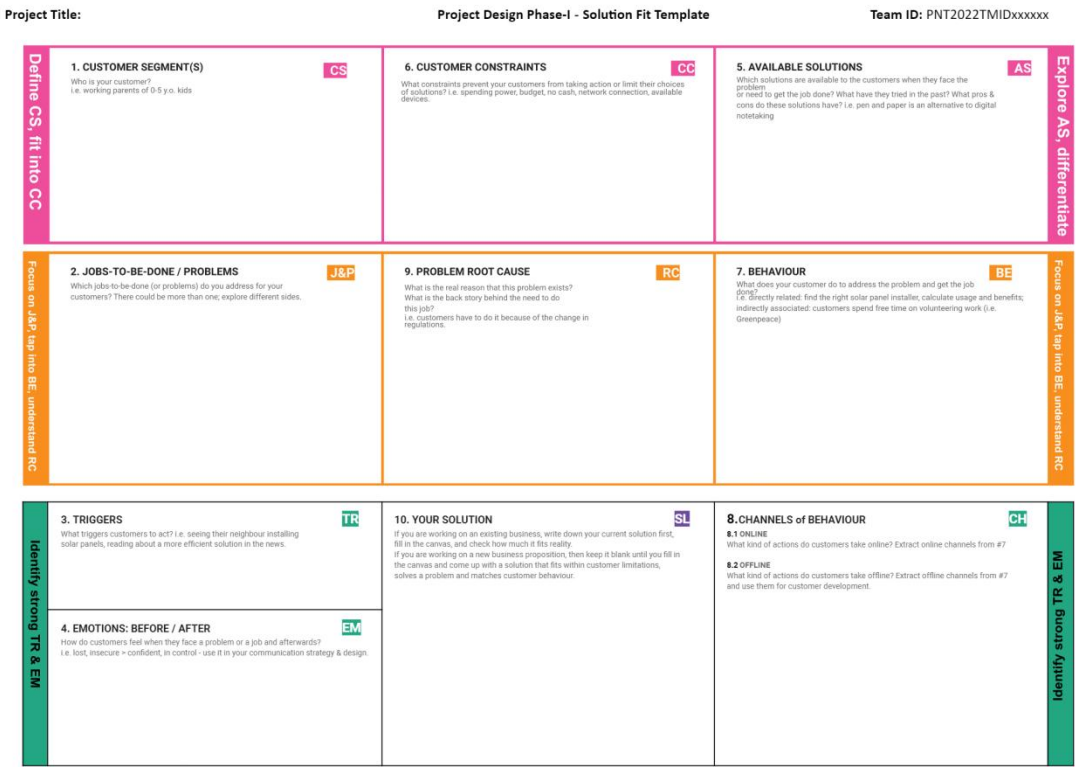
UNIQUENESS /NOVELTY:
Paperless and Hassles process. Also, it would yield maximum positive results for customers. 95% of the customer would be able to get approved for the loan, fulfilling all loan types of customers

SOCIAL IMPACT/CUSTOMER SATISFACTION:
Easy approval of a loan can help an individual to fulfill his dream, reducing depression and rejections.

BUSINESS MODEL:
Friendly on finance, charging less for a monthly subscription or yearly or a one-shot amount for the lifetime subscription

SCALABILITY OF THE SOLUTION:
For now, it's a website, and we can scale it to an API, integrating the API into the Mobile Banking Apps to get the results on the customer's phone, making it even more convenient. Training on further data would yield higher accuracy to the model too.

3.3 PROBLEM SOLUTION FIT



4. REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENTS

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	**Web UI**	Forms like UI to get the data from the User
FR-2	**Predict the Credit Defaulters**	Machine Learning Model is made to predict the Credit Defaulters
FR-3	**Integration of Model and Web UI**	Flask is used to integrate the Model and Web UI
FR-4	**Cloud Deployment**	IBM Cloud Deployment and Ease of access

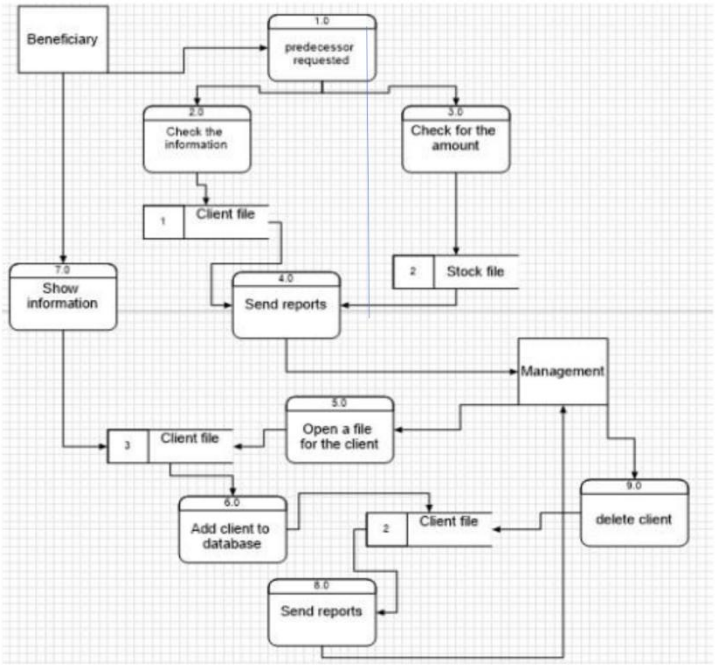
4.2 NON-FUNCTIONAL REQUIREMENTS

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

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	The user interface that serves as a conduit between the user and the cloud-deployed DL Model
NFR-2	**Security**	The cloud-based paradigm should only be accessible to authorised users and should be inaccessible to attackers or terrorists
NFR-3	**Reliability**	The system would be dependable 98 % of the time with >90% accuracy
NFR-4	**Performance**	Within seconds, the model predicts the Credit defaulters with the provided data
NFR-5	**Availability**	IBM Cloud helps in keeping the uptime of the model to 99%
NFR-6	**Scalability**	Can be scaled to an API and can be integrated into Mobile Banking Application

5. PROJECT DESIGN

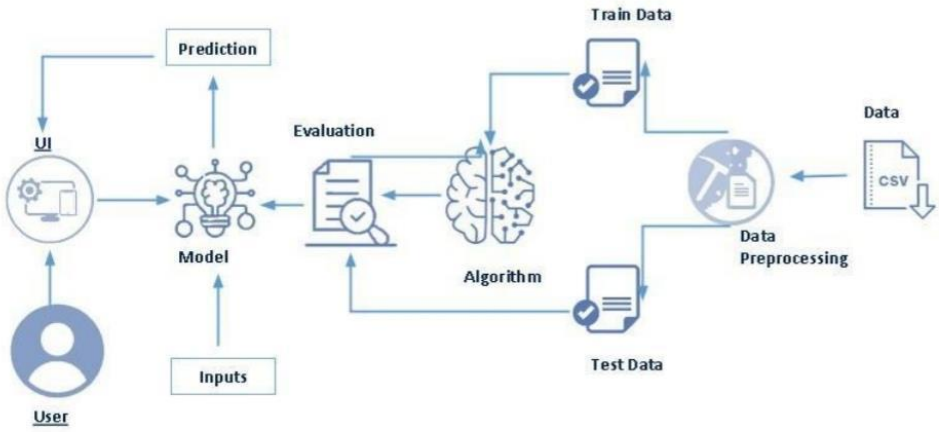
5.1 DATA FLOW DIAGRAM



5.2 SOLUTION AND TECHNICAL ARCHITECTURE

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 software's structure, characteristics, behavior, and other aspects to project stakeholders.
· Define features development phases and solution requirements.
· Provide specifications for the solution's definition, management, and delivery.



Explanation for the Architecture Diagram

1. First, the Model is trained with the obtained dataset. IBM gives the data set
2. Next, the dataset will be pre-processed, and then the data will be split to train and test data.
3. Then then the model would be saved as a PKL file
4. A website would be created for the interaction, and Flask would be used to integrate the
model and website
5. The User would give the input; the inputs would be processed, and then the prediction
would be made.
6. After the prediction is made, the output would be given as “Eligible” or “Not Eligible.”
7. This can be scaled even more as an API and integrated into the Mobile banking application, making it even more convenient for the customer to know the eligibility

5.3 USER STORIES

• Need for the data to be clean enough for Model Prediction

• As a user, I would need a place to enter my data to predict my results.

• As the data is clean now, the data can be used to Train and Evaluate the results.

• Using Flask, we can now integrate the Model. with the input given by the user.

• After Complete integration, now the Model should be deployed in IBM Cloud and put to use.

6. PROJECT PLANNING & SCHEDULING

6.1 SPRINT PLANNING AND ESTIMATION

SPRINT 1

A sprint is an Agile methodology that helps you to complete a set amount of work in a time boxed period. In this, we have done our data preprocessing and loading a dataset, leading to splitting datasets into train sets and test sets. This process is explained as follows,

Dataset

A set of data used to train the model is known as a machine learning dataset. A dataset is used as an example to educate the machine learning algorithm on how to generate predictions

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
2	LP001002	Male	No	0	Graduate	No	5849	0		360	1	Urban	Y
3	LP001003	Male	Yes	1	Graduate	No	4583	1508	128	360	1	Rural	N
4	LP001005	Male	Yes	0	Graduate	Yes	3000	0	66	360	1	Urban	Y
5	LP001006	Male	Yes	0	Not Graduate	No	2583	2358	120	360	1	Urban	Y
6	LP001008	Male	No	0	Graduate	No	6000	0	141	360	1	Urban	Y
7	LP001011	Male	Yes	2	Graduate	Yes	5417	4196	267	360	1	Urban	Y
8	LP001013	Male	Yes	0	Not Graduate	No	2333	1516	95	360	1	Urban	Y
9	LP001014	Male	Yes	3+	Graduate	No	3036	2504	158	360	0	Semiurban	N
10	LP001018	Male	Yes	2	Graduate	No	4006	1526	168	360	1	Urban	Y
11	LP001020	Male	Yes	1	Graduate	No	12841	10968	349	360	1	Semiurban	N
12	LP001024	Male	Yes	2	Graduate	No	3200	700	70	360	1	Urban	Y
13	LP001027	Male	Yes	2	Graduate		2500	1840	109	360	1	Urban	Y
14	LP001028	Male	Yes	2	Graduate	No	3073	8106	200	360	1	Urban	Y
15	LP001029	Male	No	0	Graduate	No	1853	2840	114	360	1	Rural	N
16	LP001030	Male	Yes	2	Graduate	No	1299	1086	17	120	1	Urban	Y
17	LP001032	Male	No	0	Graduate	No	4950	0	125	360	1	Urban	Y
18	LP001034	Male	No	1	Not Graduate	No	3596	0	100	240		Urban	Y
19	LP001036	Female	No	0	Graduate	No	3510	0	76	360	0	Urban	N
20	LP001038	Male	Yes	0	Not Graduate	No	4887	0	133	360	1	Rural	N
21	LP001041	Male	Yes	0	Graduate		2600	3500	115		1	Urban	Y
22	LP001043	Male	Yes	0	Not Graduate	No	7660	0	104	360	0	Urban	N
23	LP001046	Male	Yes	1	Graduate	No	5955	5625	315	360	1	Urban	Y
24	LP001047	Male	Yes	0	Not Graduate	No	2600	1911	116	360	0	Semiurban	N

This is how an original dataset would look and will be provided in a CSV file format which is
then preprocessed by evaluating the data set.

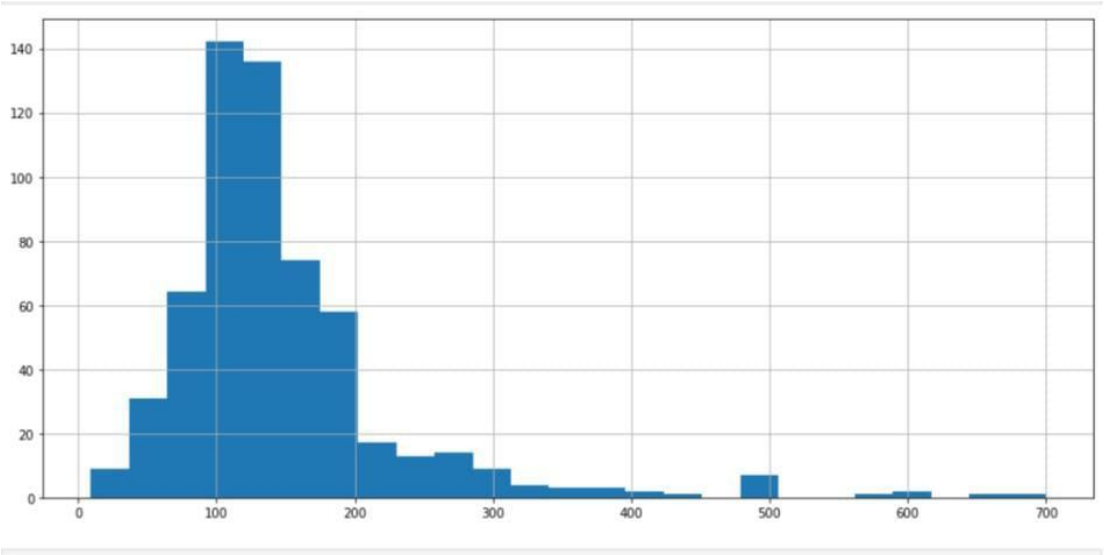
PRE-PROCESSING

Data mining methods are used in preprocessing to normalize the data collected from Kaggle. There is a need to convert because the dataset may have missing values and noisy data. So, we are using a data mining method for the cleaning method. Before using the model selection process, we used preprocessing method to reduce the null values and then recovered the data with the help of train/test split with the help of MinMaxScaler. ***Minmax Scalar***， for each value in every feature, Minmax Scalar cipher the minimum value within the feature and then divided by the vary. The range

is the distinction between the first most and the original minimum. It preserves the shapes of the first original distribution. Data preprocessing is a stage in data analysis that converts raw data into a format that computers and machine learning software can understand and evaluate. Machines like to process neat and orderly information; they interpret data as 1s and 0s. Therefore, it is simple to calculate structured data like whole numbers and percentages. Unstructured data must be organized and cleaned up before analysis, though.
Preprocessing involves specific methods that have to be performed to check whether the given dataset is valid for use. Those methods are explained step by step in the following.

i. UNDERSTANDING THE FEATURES OF DATASET

We will begin at a basic level by identifying the structure of, and then the types of data in front of us. Once we can understand what the data is, we can start to fix problems with the data. For example, we must know how much of our information is missing and what to do when we have missing data.



ii. ANALYSIS OF CATEGORICAL DATA

Categorical data classifies an observation as belonging to one or more categories. For example, an item might be judged as good or bad, or a response to a survey might include categories such as agree, disagree, or no opinion. Stat graphics have many procedures for dealing with such data, including modeling procedures in the Analysis of Variance, Regression Analysis, and Statistical Process Control sections

```
df['Loan_Status'].value_counts()['Y']
```

```
422
```

```
pd.crosstab(df ['Credit_History'], df ['Loan_Status'], margins=True)
```

Loan_Status	N	Y	All
Credit_History			
0.0	82	7	89
1.0	97	378	475
All	179	385	564

```
def percentageConvert(ser):
    return ser/float(ser[-1])
```

```
tabs = pd.crosstab(df ["Credit_History"], df ["Loan_Status"], margins=True).apply(percentageConvert,
tabs
```

Loan_Status	N	Y	All
Credit_History			
0.0	0.921348	0.078652	1.0
1.0	0.204211	0.795789	1.0
All	0.317376	0.682624	1.0

```
app_loan = tabs['Y'][1]
print(f'{app_loan*100:.2f} % applicants got their loans approved')
```

```
79.58 % applicants got their loans approved
```

iii. THE CHOICE OF AN BETTER DATASET TO TRAIN

Machine learning is a subset of AI that trains machines with vast volumes of data to think and act like humans without being explicitly programmed.
The models are available in python open-source software. There are various machine learning methods to perform an algorithm, but we have to choose the best algorithm for the enhancement of the model and for better accuracy.

Decision Trees

The basic algorithmic rule of the call tree needs all attributes or options to be discredited. Feature choice relies on the most significant info gain of possibilities. The data pictured in the call tree will delineate the IF-THEN rules. This model is an associate degree extension of C4.5 classification algorithms represented by Quinlan.

Random Forests

Random forests are a classifying learning framework for characterization (and backslide) that works by building a very large number of Decision trees at planning time and yielding the class that's the mode of the classes surrendered by individual trees.

Support Vector Machine

Used SVM to build and train a model, prepare a demonstration utilizing human cell records, and classify cells to whether the tests are benign (mild state) or dangerous (evil state). Support vector machines are managed learning models that utilize affiliation R-learning calculation, which analyzes attributes and distinguished design information for application classification. SVM can beneficially perform a replacement using the kernel trick, verifiable mapping their inputs into high dimensional attribute spaces.

K-nearest neighbor (KNN)

The KNN algorithm is a simple supervised machine learning algorithm that can unravel classification and replace issues. It is easy to implement and understand but significantly slows as the size of that data on use grows.
In all of these, RANDOM FOREST has given the best accuracy alongside a better result. Also, how this happened is, Random forest is an ensemble learning method for both classification and replacement issues. The advantage of random decision forest is reduced fitting and helps to improve accuracy and
runs efficiently on large datasets and works on both continuous and categorical values, and predicts the analysis of data with the help of test data. We used data mining methodology to predict the likely default from a dataset that contains information about home loan applications, thereby helping the banks for making better decisions in the future. On this basis, the prediction methodology based on the LSTM and SVM methods is proposed, the prediction results are compared with the traditional algorithm, and the feasibility of the model is confirmed. And this random forest has helped to select the best dataset to train, and the further process is carried out.

iv. <u>HANDLING THE DATASET WITH NULL VALUES</u>

There are many techniques for handling null values. Which techniques are appropriate for a given variable can depend strongly on the algorithms you intend to use, as well as statistical patterns in the raw data—in particular, the missing values' ***missingness*** and the randomness of the locations of the missing values. Moreover, different techniques may be appropriate for different variables in a given data set. Sometimes it is helpful to apply several techniques to a single variable. Finally, note that corrupt values are generally treated as nulls.
How to handle bull values
1.Deleting Rows
2.Replacing with Mean/Median/Mode
3.Assigning A Unique Category
4. Predicting the Missing Values
5.Using Algorithms Which Support Missing Values

➢ ***Deleting Rows***

This method is commonly used to handle null values. Here, we either delete a particular row if it has a null value for a particular feature and a particular column if it has more than 70-75% missing values. This method is advised only when enough samples are in the data set.

➢ ***Replacing with Mean/Media/Mode:***

This strategy can be applied to a feature that has numeric data like the age of a person or the ticket fare. We can calculate the mean, median, or mode of the feature and replace it with the missing values.

```
df['Gender'].fillna(df['Gender'].mode()[0],inplace=True)
df['Married'].fillna(df['Married'].mode()[0],inplace=True)
df['Dependents'].fillna(df['Dependents'].mode()[0],inplace=True)
```

```
df['LoanAmount'].fillna(df['LoanAmount'].mean(), inplace=True)
df['Loan_Amount_Term'].fillna(df['Loan_Amount_Term'].mean(), inplace=True)
df['ApplicantIncome'].fillna(df['ApplicantIncome'].mean(), inplace=True)
df['CoapplicantIncome'].fillna(df['CoapplicantIncome'].mean(), inplace=True)
```

```
df['Gender'].fillna(df['Gender'].mode()[0], inplace=True)
df['Married'].fillna(df['Married'].mode()[0], inplace=True)
df['Dependents'].fillna(df['Dependents'].mode()[0], inplace=True)
df['Loan_Amount_Term'].fillna(df['Loan_Amount_Term'].mode()[0], inplace=True)
df['Credit_History'].fillna(df['Credit_History'].mode()[0], inplace=True)
```

➢ ***Assigning A Unique Category***

A categorical feature will have a definite number of possibilities, such as gender. Since they have a definite number of classes, we can assign another class for the missing values.

➢ ***Predicting The Missing Values***

Using the features which do not have missing values, we can predict the nulls with the help of a machine learning algorithm. This method may result in better accuracy unless a missing value is expected to have a high variance.

➢ ***Using Algorithms Which Support Missing Values***

KNN is a machine learning algorithm that works on the distance measure principle. This algorithm can be used when there are nulls present in the dataset. While the algorithm is applied, KNN considers the missing values by taking the majority of the K nearest value.

```
df.isnull().any()
```

```
Gender               False
Married              False
Dependents           False
Education            False
Self_Employed        False
ApplicantIncome      False
CoapplicantIncome    False
LoanAmount           False
Loan_Amount_Term     False
Credit_History       False
Property_Area        False
Loan_Status          False
dtype: bool
```

```
df.head()
```

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property
0	Male	No	0	Graduate	No	8.674026	0.0	4.857444	360.0	1.0	
1	Male	Yes	1	Graduate	No	8.430109	1508.0	4.852030	360.0	1.0	
2	Male	Yes	0	Graduate	Yes	8.006368	0.0	4.189655	360.0	1.0	
3	Male	Yes	0	Not Graduate	No	7.856707	2358.0	4.787492	360.0	1.0	
4	Male	No	0	Graduate	No	8.699515	0.0	4.948760	360.0	1.0	

Training data (or a training dataset) is the initial data used to train machine learning models. Training datasets are fed to machine learning algorithms to teach them how to make predictions or perform a desired task.

Training a dataset depends upon the method we train our dataset; on this basis, we have two divisions supervised and unsupervised learning.

Random forest is a supervised learning algorithm. A random forest is an ensemble of decision trees combined with a bagging technique.

In bagging, decision trees are used as parallel estimators. Combining many decision trees in parallel greatly reduces the risk of over fitting and results in a much more accurate model.

The success of a random forest highly depends on using uncorrelated decision trees. If we use the same or similar trees, the overall result will not be much different than that of a single decision tree. Random forests achieve to have uncorrelated decision trees by bootstrapping and feature randomness.

Bootstrapping is randomly selecting samples from training data with replacement. They are called bootstrap samples. Feature randomness is achieved by selecting features randomly for each decision tree in a random forest. The number of features used for each tree in a random forest can be controlled with the max_features parameter.

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
2	1	1	0	0	0	8.699814745210191	2230.0	5.379729823986222	360.0	1	1	0
3	1	1	0	0	0	7.992268643270745	2900.0	4.875197323201151	360.0	1	1	1
4	1	1	2	0	0	8.740336742730447	1698.0	5.347107330717468	360.0	1	1	1
5	1	1	0	0	0	7.643564441260972	3130.0	4.852030263919617	360.0	1	1	1
6	1	0	0	0	0	8.334711621520917	0.0	4.584967478670572	360.0	0	1	0
7	1	1	0	0	0	9.559595399016661	3266.0	6.345363636828396	360.0	1	0	0
8	1	1	3	1	1	8.11162307830774	2166.0	4.867334430435382	360.0	1	1	1
9	1	1	2	1	0	8.121153242073828	1917.0	4.718495871293094	360.0	0	0	0
10	0	1	0	0	0	8.195609367285773	0.0	4.68213122712422	360.0	1	1	1
11	1	0	0	0	0	9.159858525480083	0.0	5.231105616834357	360.0	1	0	1

v. TESTING THE DATASET

Once we train the model with the training dataset, it's time to test the model with the test dataset. This dataset evaluates the model's performance and ensures that the model can generalize well with the new or unseen dataset. ***The test dataset is another subset of the original data independent of the training dataset***. However, it has some similar features and class probability distribution and uses it as a benchmark for model evaluation once the model training is completed. Test data is a well-organized dataset containing data for each type of scenario for a given problem that the model would face when used in the real world. Usually, the test dataset is approximately 20-25% of the total original data for an ML project.
At this stage, we can also check and compare the testing accuracy with the training accuracy, which means how accurate our model is with the test dataset against the training dataset. If the model's accuracy on training data is more significant than that on testing data, then the model is said to have over fitting.

The testing data should:

- Represent or part of the original dataset.
- It should be large enough to give meaningful predictions.

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
2	1	1	0	0	0	9.114139813302214	0.0	5.429343623934441	360.0	1	1	1
3	1	1	0	0	0	8.368693183097703	0.0	4.867334430433382	360.0	1	1	1
4	1	1	2	0	0	8.334951631422434	1447.0	5.062393033026967	360.0	1	0	1
5	0	0	0	0	0	7.972466019743633	0.0	4.262679877041383	360.0	1	0	1
6	1	0	0	0	0	7.907631394711089	0.0	4.248493242049339	360.0	1	1	1
7	1	1	1	0	0	7.485491608030734	2232.0	4.672828834461906	360.0	1	0	1
8	1	1	2	0	0	8.220672170297232	0.0	4.787491742782046	360.0	1	1	1
9	1	1	0	0	1	8.006367367630246	0.0	4.159654742026423	360.0	1	2	1
10	1	1	3	1	0	7.881152202227102	1587.0	5.153291394497779	360.0	1	0	0
11	1	1	1	0	1	6.907733278982137	3022.0	4.700480365792417	360.0	1	2	0
12	1	1	2	0	0	8.51719319416238	0.0	4.276666119016053	360.0	0	1	0
13	1	1	0	0	0	8.011333109161256	2188.0	5.030437921392435	360.0	1	0	1
14	1	1	3	1	0	7.633758561393383	734.0	4.543294782270004	480.0	1	1	1
15	1	1	2	1	0	8.188966563648876	1390.0	4.537444178729352	360.0	1	0	1
16	1	1	2	0	1	8.106314316253183	3500.0	4.477336814478207	360.0	1	2	1
17	1	1	0	0	0	8.0106913391303	3033.0	4.353876891600341	300.0	1	2	1
18	1	0	0	0	0	7.8136103320351903	0.0	4.077537443903572	360.0	1	2	1

SPRINT 2:

Model Building

Setting up methods for data collection, understanding and paying attention to what is significant in the data to address the questions you are posing, and finding a simulation, statistical, or mathematical model to gain understanding and make predictions are all part of the model-building process.

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import MaxAbsScaler
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.model_selection import cross_val_score
from sklearn.metrics import f1_score
import pickle
```

```
scaler = MaxAbsScaler()
```

```
train = pd.read_csv('train.csv')
```

```
test = pd.read_csv('test.csv')
```

```
train.head()
```

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area
0	1	1	0	0	0	8.699515	2250.0	5.579730	360.0	1	1
1	1	1	0	0	0	7.992269	2900.0	4.875197	360.0	1	1
2	1	1	2	0	0	8.740337	1695.0	5.347108	360.0	1	1
3	1	1	0	0	0	7.641564	3150.0	4.852030	360.0	1	1
4	1	0	0	0	0	8.334712	0.0	4.584967	360.0	0	1

Pickle is primarily used in Python to serialize and deserialize Python object structures. To put it another way, it is the procedure of converting a Python object into a byte stream so that it can be stored in a file or database, have its state preserved across sessions, or be used to transfer data over a network.

Testing & Training

The train/test approach is a way to gauge how accurate your model is.
Because the data set is divided into two sets—a training set and a testing set—this technique is
known as train/test.

SPRINT 3

▪ Web UI

HTML

The most fundamental component of the Web is HTML (HyperText Markup Language). It describes the purpose and organization of web content. Links that join online pages together, either
inside a single website or between websites, are referred to as "hypertext." An essential component of the Web are links. You can participate actively in the World Wide Web by publishing content online and linking it to other people's web pages.

CSS

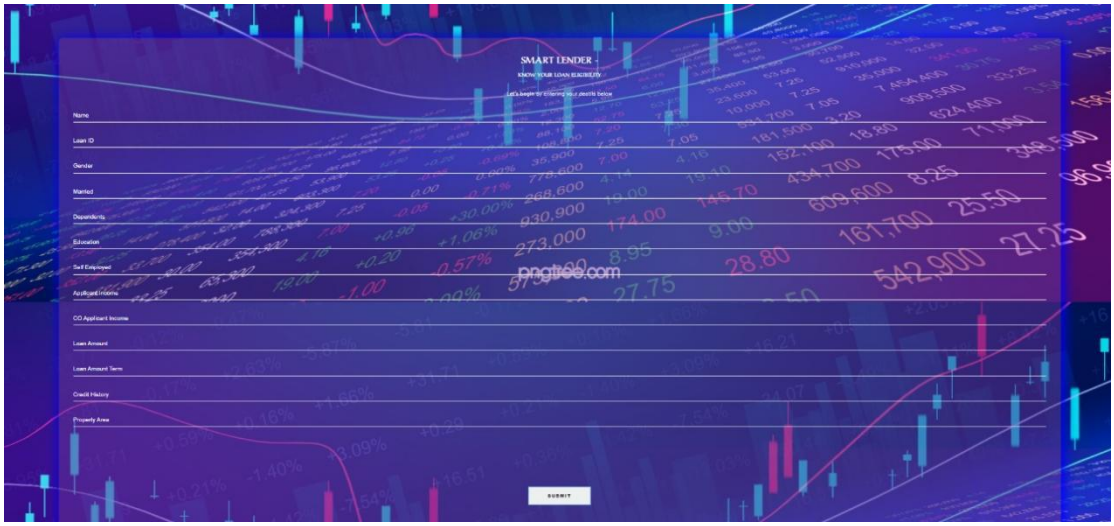
A stylesheet language called Cascading Style Sheets (CSS) is used to specify how a document written in HTML or XML is presented (including XML dialects such as SVG, MathML or XHTML). CSS specifies how items should be shown in various media, including speech, paper, screens, and other media. According to W3C guidelines, CSS is one of the basic languages of the open web and is standardized across all Web browsers.

JS

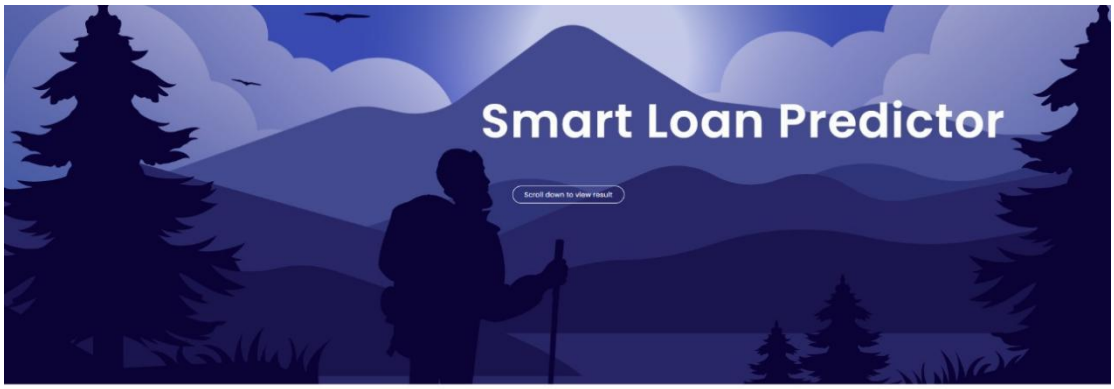
Although it is most famous for being the scripting language for Web pages, JavaScript (commonly abbreviated to JS) is a lightweight, interpreted, object-oriented language with first-class functions that is also utilized in other non-browser applications. It is a dynamic, prototype-based multi paradigm scripting language that supports imperative, functional, and object-oriented programming paradigms. JavaScript, which operates on the client side of the web, may be used to plan or programme how web pages should act in response to an event. JavaScript is a popular scripting language for managing the behavior of web pages because it is simple to learn and extremely effective.We have integrated HTML, CSS & JS to create a website UI

I. HOME PAGE

II. PREDICT PAGE



III. SUBMIT PAGE



STATUS : 'ELIGIBLE'

Web Framework

Web application developers can write applications without having to be concerned about low-level details like protocol, thread management, and other issues thanks to a web framework, also known as a web application framework.

Flask

Python is used to create the Flask web application framework. It was created by Armin Ronacher, who served as the team leader of Pocco, a worldwide group of Python aficionados. The Werkzeg WSGI toolkit and the Jinja2 template engine serve as the foundation for Flask. They're both Pocco projects. Its core is compact and simple to expand.

```
from flask import render_template,Flask,request
import numpy as np
import pickle

app= Flask(__name__, template_folder='templates')

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

@app.route('/')
def home():
    return render_template('index.html')
@app.route('/predict.html')
def formpg():
    return render_template('predict.html')
@app.route('/submit',methods = ['POST'])
def predict():
    loan_num,gender,married,depend,education,self_emp,applicant_income,co_income,loan_amount,loan_term,credit_history,prc
    if gender == 'Male':
        gender = 1
    else:
        gender = 0

    if married == 'Yes':
        married = 1
    else:
        married = 0

    if education == 'Graduate':
        education = 0
    else:
        education = 1

    if self_emp == 'Yes':
        self_emp = 1
    else:
        self_emp = 0

    if depend == '3+':
        depend = 3

    applicant_income = int(applicant_income)
    applicant_income = np.log(applicant_income)
    loan_amount = int(loan_amount)
    loan_amount = np.log(loan_amount)
```

This is how the flask is used for developing web application, and There is a built-in development server and a fast debugger provided which gives a better result.

Pickle

Pickle is primarily used in Python to serialize and deserialize Python object structures. To put it another way, it is the procedure of converting a Python object into a byte stream so that it can be stored in a file or database, have its state preserved across sessions, or be used to transfer data over a network

SPRINT 4

Cloud deployment is the process of deploying an application through one or more hosting models—software as a service (SaaS), platform as a service (PaaS), and infrastructure as a service (IaaS)—that leverage the cloud. This includes architecture, planning, implementing, and operating workloads on the cloud.

Cloud web

Cloud integration connects disparate cloud-based systems into a single platform. By breaking down software silos, cloud integration platforms let you access and manage applications and data from different software systems in one place. Streamline business operations. Improve the efficiency of data management. Reduce costs. Improve customer interactions.

```
from flask import render_template,Flask,request
import numpy as np
import pickle

app= Flask(__name__, template_folder='templates')

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

@app.route('/')
def home():
    return render_template('index.html')
@app.route('/predict.html')
def formpg():
    return render_template('predict.html')
@app.route('/submit',methods = ['POST'])
def predict():
    loan_num,gender,married,depend,education,self_emp,applicant_income,co_income,loan_amount,loan_term,credit_history,prc
    if gender == 'Male':
        gender = 1
    else:
        gender = 0

    if married == 'Yes':
        married = 1
    else:
        married = 0

    if education == 'Graduate':
        education = 0
    else:
        education = 1

    if self_emp == 'Yes':
        self_emp = 1
    else:
        self_emp = 0

    if depend == '3+':
        depend = 3

    applicant_income = int(applicant_income)
    applicant_income = np.log(applicant_income)
    loan_amount = int(loan_amount)
    loan_amount = np.log(loan_amount)
```

IBM Cloud

```

from flask import render_template,Flask,request
import numpy as np
import pickle
import requests

# NOTE: you must manually set API_KEY below using information retrieved from your IBM Cloud account.
API_KEY = "hmIOFhnjuvRGrJaKtFnyvNKEQTINuL4eRrcnbp6K7c8R"
token_response = requests.post('https://iam.cloud.ibm.com/identity/token', data={"apikey":API_KEY, "grant_type": 'urn:ibm
mltoken = token_response.json()["access_token"]

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


app= Flask(__name__, template_folder='templates')

scale = pickle.load(open('scale.pkl','rb'))

@app.route('/')
def home():
    return render_template('index.html')
@app.route('/predict.html')
def formpg():
    return render_template('predict.html')
@app.route('/submit',methods = ['POST'])
def predict():
    loan_num,gender,married,depend,education,self_emp,applicant_income,co_income,loan_amount,loan_term,credit_history,prc
    if gender == 'Male':
```

Cloud Integration

```
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.
client_5158bfd5065b40c4b6cf7e02a60cf879 = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='Rob46tTNo97O_Wdw9cPUe7whW_akOBfAuD9qWugyZBTB',
    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')

body = client_5158bfd5065b40c4b6cf7e02a60cf879.get_object(Bucket='ibmsmartlender-donotdelete-pr-fn1gc
# add missing __iter__ method, so pandas accepts body as file-like object
if not hasattr(body, "__iter__"): body.__iter__ = types.MethodType( __iter__, body )

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

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Cr
0	1	1	0	0	0	9.114160	0.0	5.429346	360.0	
1	1	1	0	0	0	8.368693	0.0	4.867534	360.0	
2	1	1	2	0	0	8.334952	1447.0	5.062595	360.0	
3	0	0	0	0	0	7.972466	0.0	4.262680	360.0	
4	1	0	0	0	0	7.907652	0.0	4.248495	360.0	

6.2 Sprint Delivery Schedule

Project Tracker, Velocity & Burndown Chart: (4 Marks)

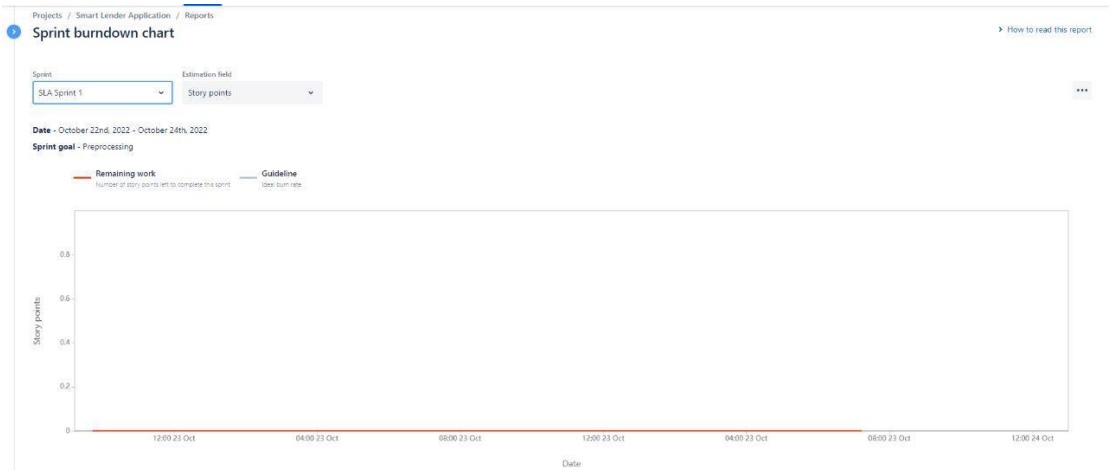
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	6	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	6	6 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	6	13 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	6	19 Nov 2022

Velocity:
Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate the team's average velocity (AV) per iteration unit (story points per day)

$$AV = \frac{sprint\ duration}{velocity} = \frac{20}{10} = 2$$

6.3 Reports From JIRA

SPRINT 1

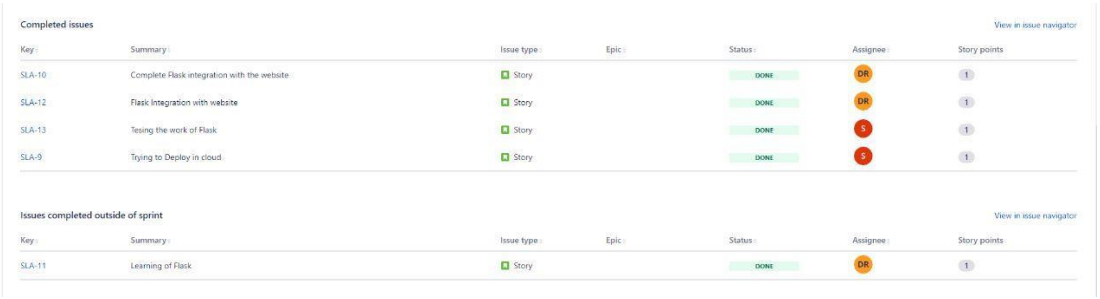


SPRINT 2

SPRINT 3

SPRINT 4

ISSUES COMPLETED OUTSIDE OF SPRINT



Completed issues

View in issue navigator

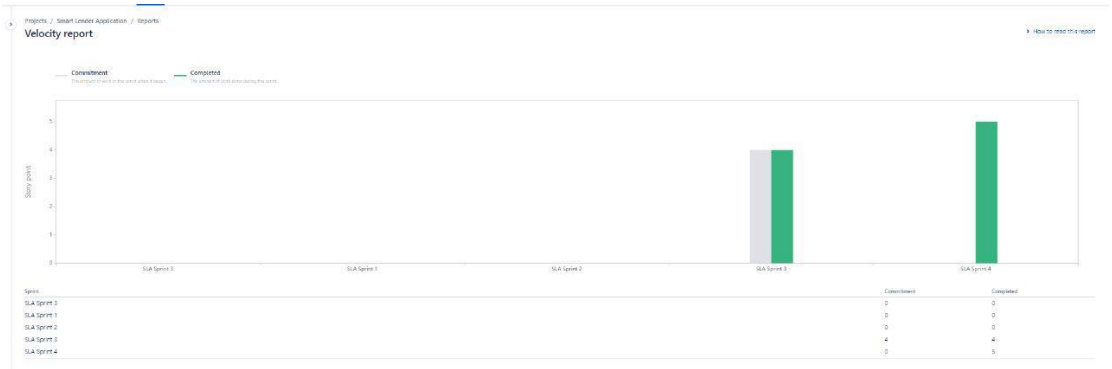
Key	Summary	Issue type	Epic	Status	Assignee	Story points
SLA-10	Complete Flask integration with the website	Story		DONE	DR	1
SLA-12	Flask Integration with website	Story		DONE	DR	1
SLA-13	Tesing the work of Flask	Story		DONE	S	1
SLA-9	Trying to Deploy in cloud	Story		DONE	S	1

Issues completed outside of sprint

View in issue navigator

Key	Summary	Issue type	Epic	Status	Assignee	Story points
SLA-11	Learning of Flask	Story		DONE	DR	1

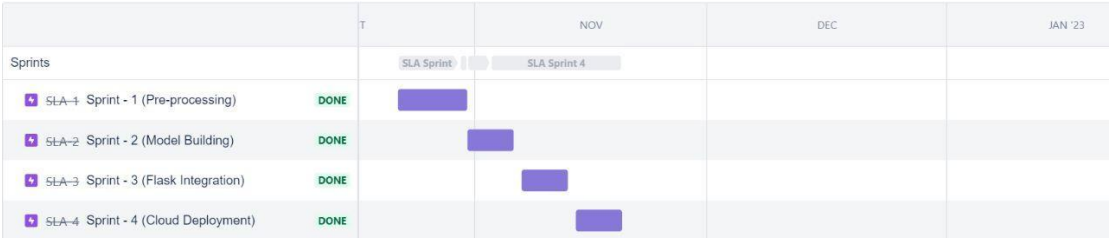
VELOCITY REPORT



CUMULATIVE FLOW DIAGRAM

COMPLETED ISSUES

OVERALL PREPROCESSING



7. CODING & SOLUTIONING

7.1 Feature 1

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

df = pd.read_csv('loan_prediction.csv')
df.head(10)

df.describe()
df.isnull().any()

df.drop('Loan_ID',axis=1,inplace=True)
df.Property_Area.unique()

plt.figure(figsize=(15,7))
df['ApplicantIncome'].hist(bins=25)
plt.show()

df.boxplot(column='ApplicantIncome',figsize=(15,7))
df.boxplot(column='ApplicantIncome', by = 'Education',figsize=(15,7))
df['Property_Area'].value_counts()
```

Analysis of Categorical Values

```
df['Loan_Status'].value_counts()['Y']
pd.crosstab(df ['Credit_History'], df ['Loan_Status'], margins=True)
plt.figure(figsize=(15,7))
df['ApplicantIncome'].hist(bins=20)
plt.show()
df['ApplicantIncome'] = np.log(df['ApplicantIncome'])plt.figure
(figsize=(15,7))
df['ApplicantIncome'].hist(bins=25)
plt
.show()
def percentageConvert(ser):
return ser/float(ser[-1])
tabs = pd.crosstab(df ["Credit_History"], df ["Loan_Status"], margins=True).apply(percentageConvert,
axis=1)
tabs
app_loan = tabs['Y'][1]
print(f'{app_loan*100:.2f} % applicants got their loans approved')
```

So this is a good data set to train with

```
df['Self_Employed'].fillna('No',inplace=True)
#df['TotalIncome'] = df['ApplicantIncome'] + df['CoapplicantIncome']
#df['TotalIncome_log'] = np.log(df['TotalIncome'])
#plt.figure(figsize=(15,7))
#df['TotalIncome'].hist(bins=25)
#plt.show()
#plt.figure(figsize=(15,7))
#df['TotalIncome_log'].hist(bins=25)
#plt.show()
plt.figure(figsize=(15,7))
```

```
df['LoanAmount'].hist(bins=20)
plt.show()
df['LoanAmount'] = np.log(df['LoanAmount'])
plt.figure(figsize=(15,7))
df['LoanAmount'].hist(bins=25)
plt.show()df['LoanAmount'] = np.log(df['LoanAmount'])
plt.figure(figsize=(15,7))
df['LoanAmount'].hist(bins=25)
plt.show()
plt.figure(figsize=(15,7))
df['ApplicantIncome'].hist(bins=20)
plt.show()
df['ApplicantIncome'] =
np.log(df['ApplicantIncome'])
plt.figure(figsize=(15,7))
df['ApplicantIncome'].hist(bins=25)
plt.show()
plt.figure(figsize=(15,7))
df['Loan_Amount_Term'].hist(bins=20)
plt.show()
#df['ApplicantIncome'] = np.log(df['ApplicantIncome'])
#plt.figure(figsize=(15,7))
#df['ApplicantIncome'].hist(bins=25)
#plt.show()
#df.drop('LoanAmount',axis=1,inplace=True)
#df.drop('TotalIncome',axis=1,inplace=True)
df.head()
```

Now to Handle with null values

```
df['Gender'].fillna(df['Gender'].mode()[0],inplace=True)
df['Married'].fillna(df['Married'].mode()[0],inplace=True)

df['Dependents'].fillna(df['Dependents'].mode()[0],inplace=True)
df['LoanAmount'].fillna(df['LoanAmount'].mean(), inplace=True)

df['Loan_Amount_Term'].fillna(df['Loan_Amount_Term'].mean(), inplace=True)
df['ApplicantIncome'].fillna(df['ApplicantIncome'].mean(), inplace=True)

df['CoapplicantIncome'].fillna(df['CoapplicantIncome'].mean(), inplace=True)
df['Gender'].fillna(df['Gender'].mode()[0], inplace=True)

df['Married'].fillna(df['Married'].mode()[0], inplace=True)
df['Dependents'].fillna(df['Dependents'].mode()[0], inplace=True)

df['Loan_Amount_Term'].fillna(df['Loan_Amount_Term'].mode()[0], inplace=True)
df['Credit_History'].fillna(df['Credit_History'].mode()[0], inplace=True)

df.isnull().any()df.head()

cat=['Gender','Married','Dependents','Education','Self_Employed','Credit_History','Property_Area'] target =
['Loan_Status']
all_cols = ['Gender', 'Married', 'Dependents', 'Education',
'Self_Employed','ApplicantIncome','CoapplicantIncome', 'Loan_Amount_Term',
'Credit_History','Property_Area', 'Loan_Status', 'TotalIncome_log', 'LoanAmount_log']

from sklearn.preprocessing import LabelEncoder,OneHotEncoder

for var in cat:
le = LabelEncoder()
df[var]=le.fit_transform(df[var].astype('str'))
print('Done encoding Catergorical Values')
```

```
for tar in target:
oe = OneHotEncoder()
df[tar]=le.fit_transform(df[tar].astype('str'))
print('Done encoding Target Value')
df.head(5)

from sklearn.model_selection import train_test_split

train, test = train_test_split(df,test_size=0.2,random_state=42)
test.to_csv('test.csv',encoding='utf-8',index=False)
train.to_csv('train.csv',encoding='utf-8',index=False)
```

7.2 Feature 2

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import MaxAbsScaler
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.model_selection import cross_val_score
from sklearn.import f1_score
import pickle

scaler = MaxAbsScaler()
train = pd.read_csv('train.csv')
test = pd.read_csv('test.csv')
train.head()

train_y = train.iloc[:,-1]
train_x = train.drop('Loan_Status',axis=1)
test_y = test.iloc[:,-1]
test_x = test.drop('Loan_Status',axis=1)

x = pd.concat([train_x,test_x],axis=0)
y = pd.concat([train_y,test_y],axis=0)

train_x = scaler.fit_transform(train_x)
test_x = scaler.transform(test_x)

def decisionTree(train_x,test_x,train_y,test_y):
dt = DecisionTreeClassifier()
dt.fit(train_x,train_y)
y_pred = dt.predict(test_x)
print("**** Decision Tree Classifier ****")
print('Confusion Matrix')
print(confusion_matrix(test_y,y_pred))
print('Classification Report')
print(classification_report(test_y,y_pred))

def randomForest(train_x,test_x,train_y,test_y):
rf = RandomForestClassifier()
rf.fit(train_x,train_y)
y_pred = rf.predict(test_x)
print("**** Random Forest Classifier ****")
print('Confusion Matrix')
print(confusion_matrix(test_y,y_pred))
print('Classification Report')
print(classification_report(test_y,y_pred))def randomForest(train_x,test_x,train_y,test_y):
rf = RandomForestClassifier()
```

```
rf.fit(train_x,train_y)
y_pred = rf.predict(test_x)

print("**** Random Forest Classifier ****")
print('Confusion Matrix')
print(confusion_matrix(test_y,y_pred))
print('Classification Report')
print(classification_report(test_y,y_pred))

def knn(train_x,test_x,train_y,test_y):
knn = KNeighborsClassifier()
knn.fit(train_x,train_y)
y_pred = knn.predict(test_x)

print("**** KNeighbour Classifier ****")
print('Confusion Matrix')
print(confusion_matrix(test_y,y_pred))
print('Classification Report')
print(classification_report(test_y,y_pred))

def xgboost(train_x,test_x,train_y,test_y):
xg = GradientBoostingClassifier()
xg.fit(train_x,train_y)
y_pred = xg.predict(test_x)

print("**** Gradient Boosting Classifier ****")
print('Confusion Matrix')
print(confusion_matrix(test_y,y_pred))
print('Classification Report')
print(classification_report(test_y,y_pred))

decisionTree(train_x,test_x,train_y,test_y)

def randomForest(train_x,test_x,train_y,test_y):
rf = RandomForestClassifier()
rf.fit(train_x,train_y)
y_pred = rf.predict(test_x)

print("**** Random Forest Classifier ****")
print('Confusion Matrix')
print(confusion_matrix(test_y,y_pred))
print('Classification Report')
print(classification_report(test_y,y_pred))

def knn(train_x,test_x,train_y,test_y):
knn = KNeighborsClassifier()
knn.fit(train_x,train_y)
y_pred = knn.predict(test_x)

print("**** KNeighbour Classifier ****")
print('Confusion Matrix')
print(confusion_matrix(test_y,y_pred))
print('Classification Report')
print(classification_report(test_y,y_pred))

def xgboost(train_x,test_x,train_y,test_y):
xg = GradientBoostingClassifier()
xg.fit(train_x,train_y)
y_pred = xg.predict(test_x)

print("**** Gradient Boosting Classifier ****")
print('Confusion Matrix')
print(confusion_matrix(test_y,y_pred))
```

```
print('Classification Report')
print(classification_report(test_y,y_pred))

decisionTree(train_x,test_x,train_y,test_y)
randomForest(train_x,test_x,train_y,test_y)
knn(train_x,test_x,train_y,test_y)
xgboost(train_x,test_x,train_y,test_y)
rf = RandomForestClassifier()
rf.fit(train_x,train_y)
ypred = rf.predict(test_x)
f1_score(ypred,test_y,average='weighted')
cv = cross_val_score(rf,x,y,cv=5)
np.mean(cv)
pickle.dump(rf,open('rdf.pkl','wb'))
pickle.dump(scaler,open('scale.pkl','wb'))
```

8. TESTING

8.1 Test Cases

Test case ID	Feature Type	Component	Test Scenario	Steps To Execute	Test Data	Expected Result	Actual Result	Status
Home Page_TC_001	UI	Home Page	Verify the user is able to see the button to load for the next page	1.Enter URL and click go 2.Wait for loading and watch	Nill	Button for predict should display	Working as expected	Pass
Home Page_TC_002	Functional	Home Page	Verify the button redirects to another page	1.Enter URL and click go 2.Chech whether the button is visible or not 3.Click on the button to verfiy that redirects into another page	Nill	On clicking, the button should redirect to another page	Working as expected	Pass
Form Page_TC_003	UI	Form Page	Verify the user is able to see input fields to enter the details required	1.Enter URL and click go 2.Go to the form page by clicking the button in the home page 3.Wait until the form page loads 4.Check all the input fields are visible	gender':'male','married':'yes','dependents':'0','education':'graduate','self_employed':'no','applicant_income':'5849','coapplicant_income':'0','loan_amount':'128000','loan_amount_term':'360','credit_history':'1','property_area':'urban'	Input fields should be visible	Working as expected	Pass

Test case ID	Feature Type	Component	Test Scenario	Steps To Execute	Test Data	Expected Result	Actual Result	Status
Form Page_TC_004	Functional	Form Page	Verify the user is able to enter the details without any issues in the form	1.Enter URL and click go 2.Go to the form page by clicking the button in the home page 3.Wait until the form page loads 4.Check all the input fields are visible 5.Click on the input fileds and verify whether they can enter the details	gender':'male','married':'yes','dependents':'0','education':'graduate','self_employed':'no','applicant_income':'5849','coapplicant_income':'0','loan_amount':'128000','loan_amount_term':'360','credit_history':'1','property_area':'urban'	Input fields should be working fine	Working as expected	Pass
Form Page_TC_005	Functional	Form Page	Verify the user is able to click on the submit button to redirect to the other page	1.Enter URL and click go 2.Go to the form page by clicking the button in the home page 3.Wait until the form page loads 4.Check all the input fields are visible 5.Click on the input fileds and verify whether they can enter the details 6.Clock on the submit button to redirect to the result page	gender':'male','married':'yes','dependents':'0','education':'graduate','self_employed':'no','applicant_income':'5849','coapplicant_income':'0','loan_amount':'128000','loan_amount_term':'360','credit_history':'1','property_area':'urban'	The submit button should redirect to another page	Working as expected	Pass

Test case ID	Feature Type	Component	Test Scenario	Steps To Execute	Test Data	Expected Result	Actual Result	Status
Result Page_TC_006	UI	Result Page	Verify the user is able to see the prediction text	1.Enter URL and click go 2.Go to the form page by clicking the button in the home page 3.Wait until the form page loads 4.Check all the input fields are visible 5.Click on the input fileds and verify whether they can enter the details 6.Click on the submit button to redirect to the result page 7. Check whether the prediction text is visible	Nill	The submit button should redirect to another page	Working as expected	Pass

8.2 USER ACCEPTANCE TESTING

DEFECT ANALYSIS

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
UI	10	4	2	3	20
Model	10	8	5	2	25
Integration	9	6	0	0	15
Cloud	11	7	3	0	22
Totals	40	25	10	5	82

9. RESULTS

PERFORMANCE METRICS

S.No.	Parameter	Values
1.	Metrics	**Classification Model:** Confusion Matrix – [[18 25] [2 75]], Accuracy Score – 79% & Classification Report –

	precision	recall	f1-score	support
0	0.90	0.42	0.57	43
1	0.76	0.97	0.85	80
accuracy			0.78	123
macro avg	0.83	0.70	0.71	123
weighted avg	0.81	0.78	0.75	123

In [18]:
```
randomForest(train_x,test_x,train_y,test_y)
```

```
**** Random Forest Classifier ****
Confusion Matrix
[[18 25]
 [ 2 78]]
Classification Report
              precision    recall  f1-score   support

           0       0.90      0.42      0.57        43
           1       0.76      0.97      0.85        80

    accuracy                           0.78       123
   macro avg       0.83      0.70      0.71       123
weighted avg       0.81      0.78      0.75       123
```

In [22]:
```
f1_score(ypred,test_y,average='weighted')
```

Out[22]: 0.7977251407129455

In [23]:
```
cv = cross_val_score(rf,x,y,cv=5)
```

In [24]:
```
np.mean(cv)
```

Out[24]: 0.7915367186458749

10.ADVANTAGES & DISADVANTAGES

ADVANTAGES	DISADVANTAGES
✓ Optimization of Loan Life Cycle. ✓ Digital Lending technology thrives on process speed ✓ Easy capture of Applicant information ✓ Quicker Decision Making ✓ Consistency ✓ Comfort across Devices ✓ Perfect for first time borrowers ✓ Compliance with Rules and Regulations ✓ Power of Analytics	✓ Strict eligibility criteria ✓ One of the major disadvantages of a bank loan is that banks can be cautious about lending to small businesses ✓ Lengthy application process ✓ Not suitable for ongoing expenses ✓ Secured loans carry risk

11. CONCLUSION

We did Exploratory Data Analysis on the features of this data set and saw how each feature is distributed.

We did bi variate and multivariate analysis to see impact of one another on their features using charts.

We analyzed each variable to check if data is cleaned and normally distributed.

We cleaned the data and removed NA values.

We also regenerated hypothesis to prove an association among the independent variables and the target variable. And based in the results, we assumed whether or not there is an association.

We calculated correlation between independent variables and found that applicant income and loan amount have significiant relation.

We created dummy variables for condition structuring the model.

We constructed models taking different variables into account and found through odds ratio that credit history is creating the most impact on loan giving decision.

Finally, we got a model with co-applicant income and credit history as independent variable with highest accuracy.

We tested the data and the accuracy we achieved was 80%

12. FUTUTRE SCOPE

The future of smart lender platforms is bright; there is a massive untapped loan market in the range of 20 lakh crores. Because they are technology-based, these lending platforms have a significant potential for providing novel solutions and disruptive technology for the borrowing and lending business. Today's generation is considerably more knowledgeable and technologically aware and believes in spending money to get varied life experiences. As the habit of consumerization grows, platforms that provide instant approval and paperless loans with minimal time and personal interaction are expected to grow significantly by providing the masses with the opportunity for financial inclusion while also generating higher returns for lenders to encourage them.

SOURCE CODE

Home page

```
<!DOCTYPE html>
        <html lang="en">
        <head>
        <meta charset="UTF-8" />
        <meta http-equiv="X-UA-Compatible" content="IE=edge" />
        <meta name="viewport" content="width=device-width, initial-scale=1.0" />
        <title>Smart Loan Approval Predictor</title>
        <link rel="stylesheet" href="home.css" />

        <style>
        @import url("https://fonts.googleapis.com/css2?family=Aref+Ruqaa+Ink:wght@700&display=swap");
        @import url("https://fonts.googleapis.com/css2?family=Albert+Sans&display=swap");
        @import url("https://fonts.googleapis.com/css2?family=EB+Garamond&display=swap");
        center {
        background: rgba(102, 51, 153, 0.5);
        padding-bottom: 30vh;

        margin: 20;
        padding: 5;
        box-shadow: -5px -5px 30px 5px red, 5px 5px 30px 5px blue;

        }
        body {
        height: 90%;
        margin: 20;
        padding: 5;
        background: pink;


        }
        .container {
        display: flex;
        flex-direction: column;
        text-align: center;

        }
        .btn {
        border: 1px solid #fafafa;
        background: none;
        padding: 20px 100px;
        font-size: 30px;
        font-family: "montserrat";
        cursor: pointer;
        margin: 50px;
        transition: 0.8s;
        position: relative;
        overflow: hidden;

        }
        .btn1 {
        color: black;
        background: white;
        border-color: black;
        border-style: groove;
```

```
border-width: 5px;
border-radius: 20px;
background: pink;
}
.btn1:hover {
color: antiquewhite;
background: grey;
}
.btn::before {
content: "Predict";
position: absolute;
left: 0;

width: 100%;
height: 0%;
background: #ffffff;
z-index: 5;
transition: 0.8s;
}
.btn1::before {
top: 150;
border-radius: 100% 100% 50% 50%;
}
.btn1:hover::before {
height: 180%;
}
html {
height: 100%;
}

h1 {
font-size: 60px;
font-family: "Poppins", sans-serif;
}

h2 {
font-size: 50px;
font-family: "Poppins", sans-serif;
}
h3 {
margin-top: 30px;
font-size: 30px;
font-family: "Poppins", sans-serif;
}
#footer {
margin-bottom: 20px;
}
footer a {
text-decoration: none;
color: white;
border-color: aquamarine;
background: rgb(138, 138, 249);
border-radius: 10px;
padding: 10px;
</style>
</head>

<body>
<main>
```

```
<center>
<h1>Smart Loan Approval Predictor</h1>
<h2>Welcome to loan predictor</h2>
<h3>
Click the Predict button and enter your details to know about your
Loan Approval
</h3>

<div class="container">
<a href="predict.html">
<button class="btn btn1" onclick="predict.html">Predict</button>
</a>
</div>

<footer>
<div id="footer">
<p>Made by</p>
<br />

<a
href="https://www.linkedin.com/in/adwaid-babu-b12070190/"
target="_blank"
>Adwaid Babu
</a>

<span style="font-family: 'Verdana', serif">    </span>
<a
href="https://www.linkedin.com/in/ahmed-rashid-b61546230/"
target="_blank"
>
Ahmed Rashid.A
</a>
<span style="font-family: 'Verdana', serif">    </span>
<a
href="https://www.linkedin.com/in/abhishek-yadav-13222524b"
target="_blank"
>
Abhishek Yadav R</a
>
<span style="font-family: 'Verdana', serif">    </span>
<a
href="https://www.linkedin.com/in/ajith-r-026b1622a"
target="_blank"
>
Ajith R</a
>
</div>
</footer>
</center>
</main>
</body>
</html>
```

Predict Page

```
<!DOCTYPE html>
        <html lang="en">
        <head>
        <meta charset="UTF-8" />
        <meta http-equiv="X-UA-Compatible" content="IE=edge" />
        <meta name="viewport" content="width=device-width, initial-scale=1.0" />
        <title>SMART LENDER</title>
        <style>
        @import url("https://fonts.googleapis.com/css2?family=Aref+Ruqaa+Ink:wght@700&display=swap");
        @import url("https://fonts.googleapis.com/css2?family=Albert+Sans&display=swap");
        @import url("https://fonts.googleapis.com/css2?family=EB+Garamond&display=swap");

        html {
        height: 100%;
        background: pink;
        }

        body {
        margin: 0;
        padding: 0;
        font-family: sans-serif;
        /* background: linear-gradient(#141e30, #243b55);*/
        background: beige;
        background: pink;
        }

        .login-box {
        position: absolute;
        top: 100%;
        left: 50%;
        width: 90%;
        padding: 40px;
        transform: translate(-50%, -50%);
        background: rgba(102, 51, 153, 0.5);
        box-sizing: border-box;
        box-shadow: -5px -5px 30px 5px red, 5px 5px 30px 5px blue;
        border-radius: 10px;
        }

        ::placeholder {
        color: aliceblue;
        }

        .login-box h2 {
        margin: 0 0 30px;
        padding: 0;
        color: #fff;
        text-align: center;
        }

        .fon {
        color: #fff;
        text-align: center;
        font-family: "Albert Sans", sans-serif;
        }
```

```
.login-box .user-box {
position: relative;
}

.login-box .user-box input {
width: 100%;
padding: 10px 0;
font-size: 16px;
color: #fff;
margin-bottom: 30px;
border: none;
border-bottom: 1px solid #fff;
outline: none;
background: transparent;
}

.login-box .user-box label {
position: absolute;
top: 0;
left: 0;
padding: 10px 0;
font-size: 16px;
color: #fff;
pointer-events: none;
transition: 0.5s;
}

.login-box .user-box input:focus ~ label,
.login-box .user-box input:valid ~ label {
top: -20px;
left: 0;
color: #03e9f4;
font-size: 12px;
}

/*--- Button */

.container,
.container:before,
.container:after {
box-sizing: border-box;
padding: 0;
margin: 0;
font: 300 1em/1.5 "Open Sans", "Helvetica Neue", Arial, sans-serif;
text-decoration: none;
color: #111;
}

.btn {
background: rgb(236, 240, 241);
}

.container {
min-width: 500px;
margin: 5% auto;
text-align: center;
}

button:hover {
cursor: pointer;
```

```
}

button {
background: transparent;
outline: none;
position: relative;
border: 3px solid #81c6e8;
padding: 15px 50px;
overflow: hidden;
}

/*button:before (attr data-hover)*/
button:hover:before {
opacity: 1;
transform: translate(0, 0);
}

button:before {
content: attr(data-hover);
position: absolute;
top: 1.1em;
left: 0;
width: 100%;
text-transform: uppercase;
letter-spacing: 3px;
font-weight: 800;
font-size: 0.8em;
opacity: 0;
transform: translate(-100%, 0);
transition: all 0.3s ease-in-out;
}

/*button div (button text before hover)*/
button:hover div {
opacity: 0;
transform: translate(100%, 0);
}

button div {
text-transform: uppercase;
letter-spacing: 3px;
font-weight: 800;
font-size: 0.8em;
transition: all 0.3s ease-in-out;
}

/*--- Footer ---*/

.footer {
margin-top: 203vh;
}

.nav-link {
font-weight: bold;
font-size: 14px;
text-transform: uppercase;
text-decoration: none;
color: #031d44;
padding: 20px 0px;
/* margin: 0px 20px;*/
```

```
display: inline-block;
position: relative;
opacity: 0.75;
}

#d {
margin-top: -40px;
font-family: "EB Garamond", serif;
letter-spacing: 0.5px;
}

#p {
margin-top: -50px;
font-family: "EB Garamond", serif;
letter-spacing: 0.5px;
}

.nav-link:hover {
opacity: 1;
}

.nav-link::before {
transition: 300ms;
height: 3px;
content: "";
position: absolute;
background-color: #031d44;
}

.nav-link-fade-up::before {
width: 100%;
bottom: 5px;
opacity: 0;
}

.nav-link-fade-up:hover::before {
bottom: 10px;
opacity: 1;
}

p {
color: white;
font-family: "Aref Ruqaa Ink", serif;
letter-spacing: 0.5px;
}
</style>
</head>

<body>
<div class="login-box">
<h2
style="text-transform: uppercase; font-family: 'Aref Ruqaa Ink', serif"
>
Smart lender - <br />
<span style="font-size: 14px; color: azure"
>Know your Loan eligibility</span
>
</h2>
<p class="fon" style="font-size: 14px">
Let's begin by entering your deatils below
```

```
</p>
<br />
<form action="/submit" method="post">
<div class="user-box">
<input
type="text"
name=""
required=""
onfocus="this.placeholder='Enter your name'"
onblur="this.placeholder=''"
/>
<label>Name</label>
</div>
<div class="user-box">
<input
type="text"
name="Loan_ID"
required=""
onfocus="this.placeholder='Enter your Loan ID'"
onblur="this.placeholder=''"
/>
<label>Loan ID</label>
</div>
<div class="user-box">
<input
list="gender"
type="data-list"
name="Gender"
required=""
onchange="resetIfInvalid(this);"
onfocus="this.placeholder='Enter your Gender'"
onblur="this.placeholder=''"
/>
<label>Gender</label>
<datalist id="gender" name="gender">
<option value="Male"></option>
<option value="female"></option>
</datalist>
</div>
<div class="user-box">
<input
list="married"
type="text"
name="Married"
required=""
onchange="resetIfInvalid(this);"
onfocus="this.placeholder='Enter your Marital Status'"
onblur="this.placeholder=''"
/>
<label>Married</label>
<datalist id="married" name="married">
<option value="yes"></option>
<option value="no"></option>
</datalist>
</div>
<div class="user-box">
<input
list="dep"
```

```
type="text"
name="Dependents"
required=""
onchange="resetIfInvalid(this);"
onfocus="this.placeholder='Enter your Dependents'"
onblur="this.placeholder=''"
/>
<label>Dependents</label>
<datalist id="dep" name="dep">
<option value="0"></option>
<option value="1"></option>
<option value="2"></option>
<option value="3+"></option>
</datalist>
</div>
<div class="user-box">
<input
list="edu"
type="text"
name="Education"
required=""
onchange="resetIfInvalid(this);"
onfocus="this.placeholder='Enter your Educational Qualification'"
onblur="this.placeholder=''"
/>
<label>Education</label>
<datalist name="edu" id="edu">
<option value="Graduate"></option>
<option value="Non-Graduate"></option>
</datalist>
</div>
<div class="user-box">
<input
list="emp"
type="text"
name="Self_Employes"
required=""
onchange="resetIfInvalid(this);"
onfocus="this.placeholder='Are you self employed?'"
onblur="this.placeholder=''"
/>
<label>Self Employed</label>
<datalist name="emp" id="emp">
<option value="yes"></option>
<option value="no"></option>
</datalist>
</div>
<div class="user-box">
<input
type="number"
name="ApplicantIncome"
required=""
onfocus="this.placeholder='Enter your Income in Dollars'"
onblur="this.placeholder=''"
/>
<label>Applicant Income</label>
</div>
<div class="user-box">
```

```
<input
type="number"
name="CoaaplicantIncome"
required=""
onfocus="this.placeholder='Enter your CO Applicant Income in Dollars'"
onblur="this.placeholder=''"
/>
<label>CO Applicant Income</label>
</div>
<div class="user-box">
<input
type="number"
name="LoanAmount"
required=""
onfocus="this.placeholder='Enter your Loan Amount in Dollars'"
onblur="this.placeholder=''"
/>
<label>Loan Amount</label>
</div>
<div class="user-box">
<input
list="term"
type="text"
name="Loan_Amount_Term"
required=""
onchange="resetIfInvalid(this);"
onfocus="this.placeholder='Enter the loan amount term'"
onblur="this.placeholder=''"
/>
<label>Loan Amount Term</label>
<datalist name="term" id="term">
<option value="480"></option>
<option value="360"></option>
<option value="300"></option>
<option value="240"></option>
<option value="180"></option>
<option value="120"></option>
<option value="84"></option>
<option value="60"></option>
<option value="36"></option>
<option value="12"></option>
</datalist>
</div>
<div class="user-box">
<input
list="credit"
type="text"
name="Credit_History"
required=""
onchange="resetIfInvalid(this);"
onfocus="this.placeholder='Enter your Credit History'"
onblur="this.placeholder=''"
/>
<label>Credit History</label>
<datalist name="credit" id="credit">
<option value="yes"></option>
<option value="no"></option>
</datalist>
```

```
</div>
<div class="user-box">
<input
list="prop"
type="text"
name="Property_Area"
required=""
onchange="resetIfInvalid(this);"
onfocus="this.placeholder='Enter your area of the property'"
onblur="this.placeholder=''"
/>
<label>Property Area</label>
<datalist name="prop" id="prop">
<option value="Urban"></option>
<option value="Rural"></option>
<option value="Semi-Rural"></option>
</datalist>
</div>

<div class="container">
<a href="submit.html">
<button class="btn" data-hover="PREDICT" onclick="submit.html">
<div>SUBMIT</div>
</button>
</a>
</div>
</form>
</div>
</body>

<script>
function resetIfInvalid(el) {
//just for beeing sure that nothing is done if no value selected
if (el.value == "") return;
var options = el.list.options;
for (var i = 0; i < options.length; i++) {
if (el.value == options[i].value)
//option matches: work is done
return;
}
//no match was found: reset the value
el.value = "";
}
</script>
</html>
```

Submit Page

```
<!DOCTYPE html>
        <html lang="en">
        <head>
        <meta charset="UTF-8" />
        <meta http-equiv="X-UA-Compatible" content="IE=edge" />
        <meta name="viewport" content="width=device-width, initial-scale=1.0" />
        <title>Smart loan predictor</title>
        </head>
        <style>
        @import url("https://fonts.googleapis.com/css2?family=Poppins:wght@400;500;600&display=swap");
        * {
        padding: 0;
        margin: 0;
        scroll-behavior: smooth;
        box-sizing: border-box;
        }
        body {
        font-family: "Poppins", sans-serif;
        overflow-x: hidden;
        background: linear-gradient(#39519f, #3939e7);
        z-index: -999;
        }
        img {
        width: 100%;
        }
        ul {
        list-style: none;
        }
        a {
        text-decoration: none;
        }
        .container {
        max-width: 80rem;
        padding: 0 2rem;
        margin: 0 auto;
        }
        nav {
        width: 100%;
        position: absolute;
        top: 0;
        left: 0;
        z-index: 50;
        }
        nav .container {
        display: flex;
        justify-content: space-between;
        align-items: center;
        height: 6rem;
        }
        .logo {
        max-width: 250px;
        display: flex;
        align-items: center;
        }
```

```
.links ul {
display: flex;
}
.links a {
display: inline-block;
padding: 0.7rem 1.2rem;
margin-right: 0.6rem;
color: #ffffff;
font-size: 1.05rem;
text-transform: uppercase;
font-weight: 500;
line-height: 1;
border-radius: 1.5rem;
transition: 0.3s;
}
.links a.active,
.links a:hover {
color: #0b0235;
background: #ffffff;
}
.hamburger-menu {
width: 2.7rem;
height: 3rem;
z-index: 100;
position: relative;
display: none;
align-items: center;
justify-content: flex-end;
}
.hamburger-menu .bar {
position: relative;
width: 2.1rem;
height: 3px;
border-radius: 3px;
background: #ffffff;
}
.bar::before,
.bar::after {
content: "";
position: absolute;
width: 2.1rem;
height: 3px;
border-radius: 3px;
background: #ffffff;
}
.bar::before {
transform: translateY(-9px);
}
.bar::after {
transform: translateY(9px);
}
.hero {
position: relative;
width: 100vw;
height: 100vh;
display: flex;
align-items: center;
justify-content: center;
```

```
overflow: hidden;
}
.hero img {
position: absolute;
top: 0;
left: 0;
width: 100%;
height: 100%;
object-fit: cover;
pointer-events: none;
}
.hero #base {
z-index: 6;
}
.hero #text {
position: absolute;
color: #fefefe;
font-size: clamp(3rem, 7.5vw, 8rem);
white-space: nowrap;
letter-spacing: 3px;
z-index: 5;
right: -850px;
}
#btn {
font-size: 18px;
position: absolute;
color: #fefefe;
text-decoration: none;
padding: 8px 30px;
display: inline-block;
border: 2px solid #fefefe;
border-radius: 25px;
background: transparent;
z-index: 7;
letter-spacing: 0.51px;
transform: translateY(200px);
}
.section-padding {
padding: 80px 0;
background: #fefefe;
}
.grid {
display: grid;
grid-template-columns: repeat(2, 1fr);
align-items: center;
}
.left-content,
.right-content {
padding: 2rem;
}
h4 {
font-weight: 700;
letter-spacing: 0.51px;
line-height: 1.35;
text-align: justify;
color: #0b0235;
font-size: 1.75rem;
}
```

```
.heading {
font-size: 1.15rem;
color: #0b0235;
margin-bottom: 0.5rem;
}
p {
color: #0b0235;
font-size: 0.9rem;
padding-bottom: 2rem;
padding-right: 1rem;
}
footer {
background: #0b0235;
padding: 60px 0;
text-align: center;
}
footer img {
max-width: 400px;
}
@media (max-width: 992px) {
.hamburger-menu {
display: flex;
}
.links ul {
display: none;
}
h4 {
font-size: 1.2rem;
}
.heading {
font-size: 1rem;
}
p {
font-size: 0.8rem;
}
}
@media (max-width: 768px) {
.grid {
grid-template-columns: 1fr;
}
.left-content,
.right-content {
padding: 0.4rem;
}
}
</style>

<body>
<nav>
<div class="container">
<div class="links">
<ul>
<li><a href="#" class="active">Result</a></li>
</ul>
</div>
<div class="hamburger-menu">
<div class="bar"></div>
</div>
```

```
</div>
</nav>
<section class="hero">
<img
src="CSS Images/1.png"
class="translateX"
data-speed="0.25"
id="moon"
alt=""
/>
<img
src="CSS Images/2.png"
class="translateY"
data-speed="0.35"
id="clouds"
alt=""
/>
<img
src="CSS Images/3.png"
class="translateY"
data-speed="0.15"
id="mount"
alt=""
/>
<h1 id="text" class="translateX" data-speed="-3">Smart Loan Predictor</h1>
<a href="#" id="btn">Scroll down to view result</a>
<img
src="CSS Images/4.png"
class="translateY"
data-speed="0"
id="base"
alt=""
/>
</section>
<section id="about" class="section-padding">
<div class="container grid">
<div class="left-content">
<h4>Based on the information given, {{prediction_text}}</h4>
</div>
</div>
<div class="container grid">
<h5 class="heading"><center>{{result}}</center></h5>
</div>
</section>
<footer>
<div class="container">
<a href="#">
<p>SMart Loan Predictor</p>
</a>
</div>
</footer>
<script>
const translateX = document.querySelectorAll(".translateX");
const translateY = document.querySelectorAll(".translateY");
window.addEventListener("scroll", () => {
let scroll = window.pageYOffset;
translateX.forEach((element) => {
let speed = element.dataset.speed;
```

```
element.style.transform = `translateX(${scroll * speed}px)`;
});
translateY.forEach((element) => {
let speed = element.dataset.speed;
element.style.transform = `translateY(${scroll * speed}px)`;
});
});
</script>
</body>
</html>
```

Python App

```
from flask import render_template,Flask,request
import numpy as np
import pickle
import requests


API_KEY = "HwZw7QEADbVJuN0vzxUuA7Z6_JPu3NEVvRrGdVmjY0rj"
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__, template_folder='templates')

scale = pickle.load(open('scale.pkl','rb'))


@app.route('/')
def home():
return render_template('home.html')
@app.route('/predict.html')
def formpg():
return render_template('predict.html')
@app.route('/submit',methods = ['POST'])
def predict():
        loan_num,gender,married,depend,education,self_emp,applicant_income,co_income,loan_amount,loan_term,c
redit_history,property_area = [x for x in request.form.values()]
if gender == 'Male':
        gender = 1
        else:
        gender = 0

if married == 'Yes':
        married = 1
        else:
        married = 0

if education == 'Graduate':
        education = 0
        else:
        education = 1

if self_emp == 'Yes':
        self_emp = 1
        else:
        self_emp = 0

if depend == '3+':
        depend = 3

applicant_income = int(applicant_income)
applicant_income = np.log(applicant_income)
loan_amount = int(loan_amount)
loan_amount = np.log(loan_amount)

if credit_history == 'Yes':
        credit_history = 1
```

```
else:
        credit_history = 0

if property_area == 'Urban':
        property_area = 2

elif property_area == 'Rural':
        property_area = 0
else:
        property_area = 1



features =
[[gender,married,depend,education,self_emp,applicant_income,co_income,loan_amount,loan_term,credit_history,
property_area]]
#con_features = [np.array(features)]
scale_features = scale.fit_transform(features)
sf = scale_features.tolist()

payload_scoring = {"input_data": [{"fields":
['gender','married','depend','education','self_emp','applicant_income','co_income','loan_amount','loan_term
','credit_history','property_area'], "values": sf}]}

response_scoring = requests.post('https://us-south.ml.cloud.ibm.com/ml/v4/deployments/506e9230-fa87-4129-
af59-4c7b7afb6932/predictions?version=2022-11-12', json=payload_scoring,
headers={'Authorization': 'Bearer ' + mltoken})
print("Scoring response")
prediction = response_scoring.json()
predict = prediction['predictions'][0]['values'][0][0]



if predict == 0:
        return render_template('submit.html', prediction_text ='You are eligible for loan')
else:
        return render_template('submit.html',prediction_text = 'Sorry you are not eligible for loan')

if __name__ == "__main__":
        app.run(debug=True)
```

APPENDIX

GITHUB AND PROJECT DEMO VIDEO LINK

Demo Video Link :

https://drive.google.com/file/d/1boPzlCLW1Y_WWgG6eEugjT_0RaMg4Wi_/view?usp=sharing

GitHub Link :

https://github.com/IBM-EPBL/IBM-Project-20514-1659723284