SMART LENDER - APPLICANT CREDIBILITY PREDICTION FOR LOAN APPROVAL

TEAM ID: PNT2022TMID41056

DEEPANRAJ B (612719104015)

CHANDRU V (612719104014)

POOVARASAN D (612719104048)

TAMILAN M (612719104067)

BACHELOR OF ENGINEERING

Branch : COMPUTER SCIENCE AND ENGINEERING

of Anna University

November 2022

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

THE KAVERY ENGINERING COLLEGE

MECHERI, SALEM.

SALEM – 636453

1. INTRODUCTION

PROJECT OVERVIEW

One of the most important factors which affect our country's economy and financial condition is the credit system governed by the banks. The process of bank credit risk evaluation is recognized at banks across the globe. "As we know credit risk evaluation is very crucial, there is a variety of techniques are used for risk level calculation. In addition, credit risk is one of the main functions of the banking community.

The prediction of credit defaulters is one of the difficult tasks for any bank. But by forecasting the loan defaulters, the banks definitely may reduce their loss by reducing their non-profit assets, so that recovery of approved loans can take place without any loss and it can play as the contributing parameter of the bank statement. This makes the study of this loan approval prediction important. Machine Learning techniques are very crucial and useful in the prediction of these types of data.

We will be using classification algorithms such as Decision tree, Random Forest, KNN, and xgboost. We will train and test the data with these algorithms. From this best model is selected and saved in pkl format. Flask integration and IBM deployment is also be done.

PURPOSE

- **Knowledge of Machine Learning Algorithms.**
- **Knowledge of Python Language with Machine Learning**
- **You'll be able to understand the problem to classify if it is a regression or aclassification kind of problem.**
- **You will be able to know how to pre-process/clean the data using different data pre-processing techniques.**
- **Applying different algorithms according to the dataset and based on visualization.**
- **Real-time Analysis of Project**
- **Building ease of User Interface (UI)**
- **Navigation of ideas towards other projects (creativeness)**
- **Knowledge of building ML models.**
- **How to build web applications using the Flask framework.**

2. LITERATURE SURVEY

EXISTING PROBLEM

- **Manual cross verification of the credit records and other important data is being tried in the past. It is a time consuming process. A lot of labour is required for thistask.**
- **A lot of capital investment is also involved for the labour. The verification processis also prone to human errors hence lacking in accuracy.**
- **Low accuracy of manual credibility inspection which leads to misinterpretation of fraudulent loan applicants as credible ones and vice versa.**

REFERENCES

[1] B. P. Lohani, M. Trivedi, R. J. Singh, V. Bibhu, S. Ranjan and P. K. Kushwaha, "Machine LearningBased Model for Prediction of Loan Approval," 2022 3rd International Conference on Intelligent Engineering and Management (ICIEM), 2022, pp. 465-470, doi: 10.1109/ICIEM54221.2022.9853160.

[2] Soni PM, Varghese Paul, “Algorithm For the Loan Credibility Prediction System”, International Journal of Recent Technology and Engineering (IJRTE), Volume-8, Issue-1S4, June 2019.

[3] Okfalisa, R. Fitriani and Y. Vitriani, "The Comparison of Linear Regression Method and K-Nearest Neighbors in Scholarship Recipient," 2018 19th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), 2018, pp. 194-199, doi: 10.1109/SNPD.2018.8441068.

[4] Kathe Rutika Pramod, Panhale Sakshi Dattatray, Avhad Pooja Prakash, “An Approach For Prediction Of Loan Approval using Machine Learning Algorithm”, International Journal of CreativeResearch Thoughts (IJCRT), Volume 9, Issue 6, June 2021.

[5] Ambika, & Biradar, Santosh. (2021). Survey on Prediction of Loan Approval Using Machine Learning Techniques. International Journal of Advanced Research in Science, Communication and Technology. 449-454. 10.48175/IJARSCT-1165.

[6] Yash Divate, Prashant Rana, Pratik Chavan, “Loan Approval Prediction Using Machine Learning”, International Research Journal of Engineering and Technology (IRJET), Volume: 08 Issue: 05, May 2021.

[7] **Anant Shinde, Yash Patil, Ishan Kotian, Abhinav Shinde, Reshma Gulwani, “Loan Prediction System Using Machine Learning”, International Conference on Automation, Computing and Communication 2022 (ICACC-2022), Volume 44, May 2022.**

[8] **Q. Du, N. Li, S. Yang, D. Sun and W. Liu, "Integrating KNN and Gradient Boosting Decision Tree for Recommendation," 2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), 2021, pp. 2042-2049, doi: 10.1109/IAEAC50856.2021.9390647.**

PROBLEM STATEMENT DEFINITION

- **The prediction of credit defaulters is one of the difficult tasks for any bank.**
- **Machine Learning techniques are very crucial and useful in the prediction of these types of data. Classification algorithms such as Decision tree, Random forest, KNN, and xgboost can be utilized to serve this purpose.**
- **A model must be trained using a dataset to predict the credibility of an applicantaccurately.**

3. IDEATION & PROPOSED SOLUTION

EMPATHY MAP CANVAS

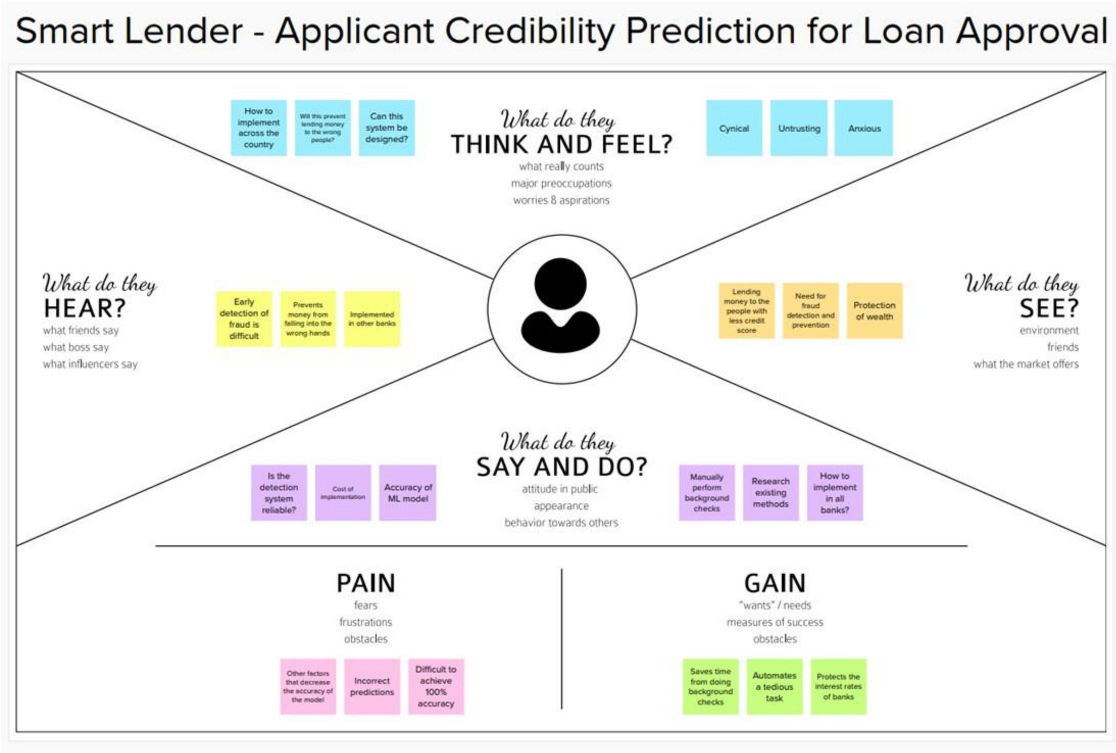

Fig 1. Empathy Map Canvas of our Problem

IDEATION & BRAINSTORMING

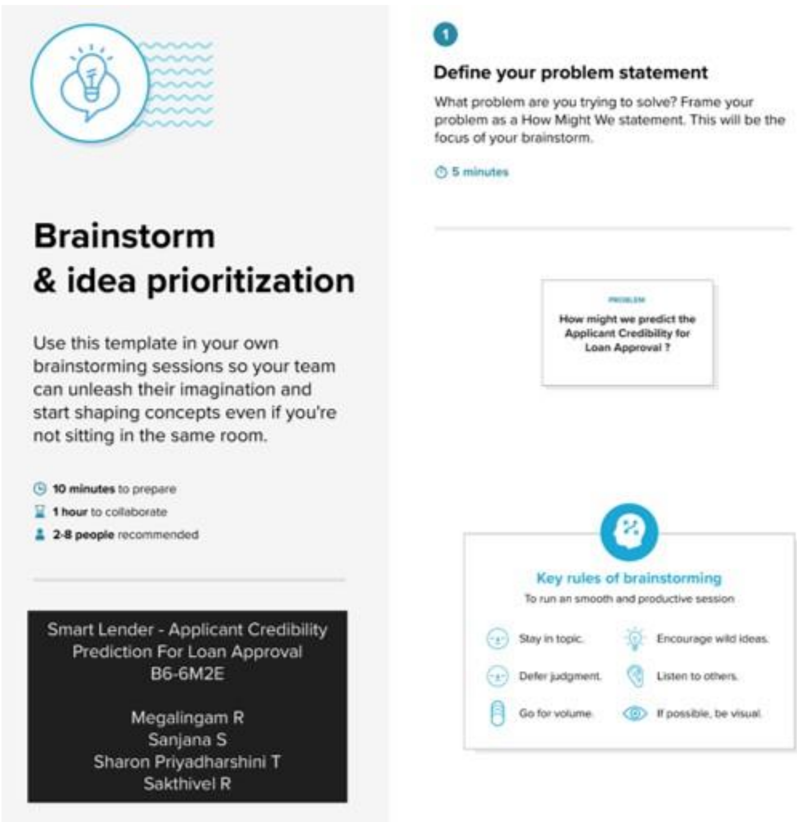

Fig 2. Step-1: Team Gathering, Collaboration and Select the Problem Statement

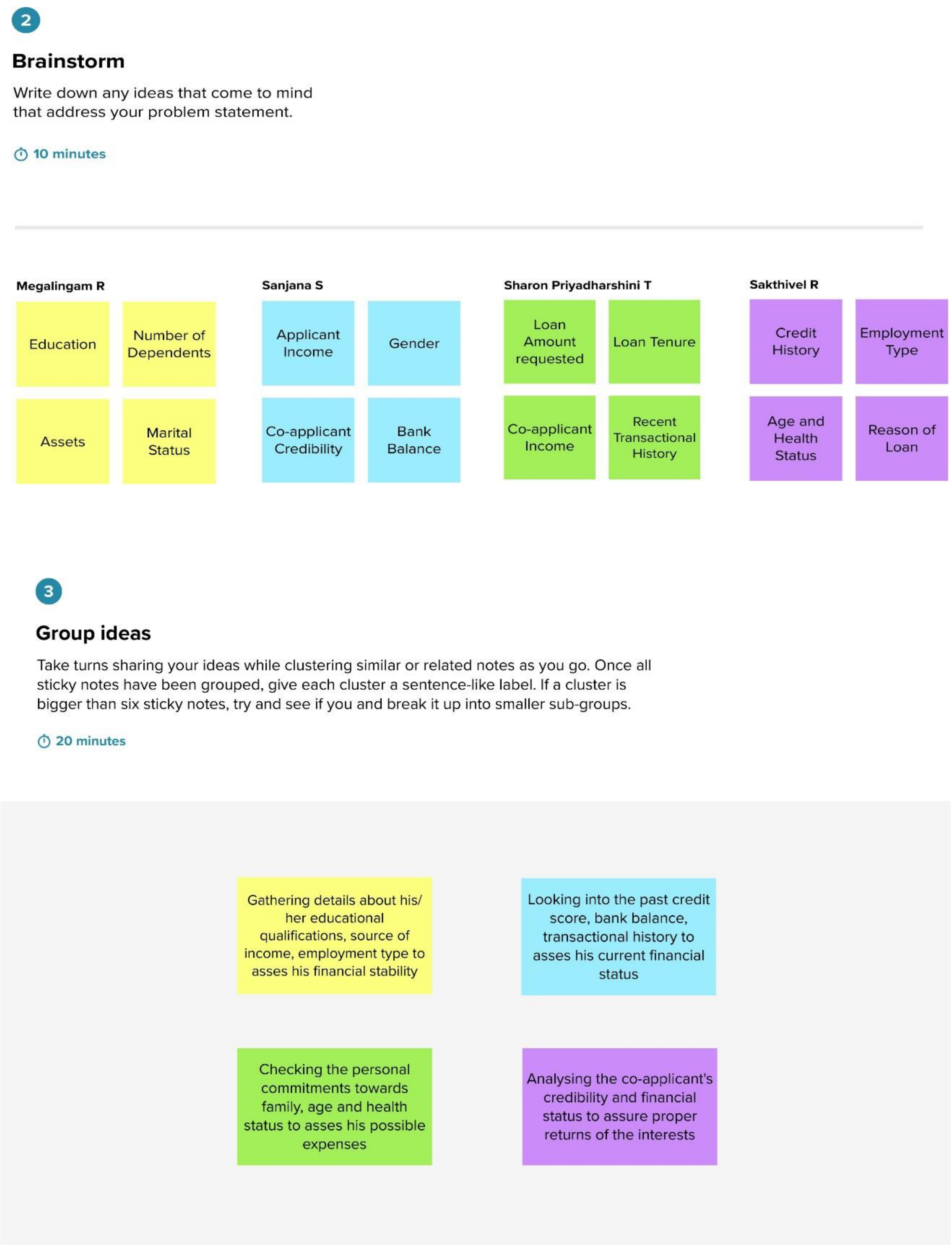

Fig 3. Step-2: Brainstorm, Idea Listing and Grouping

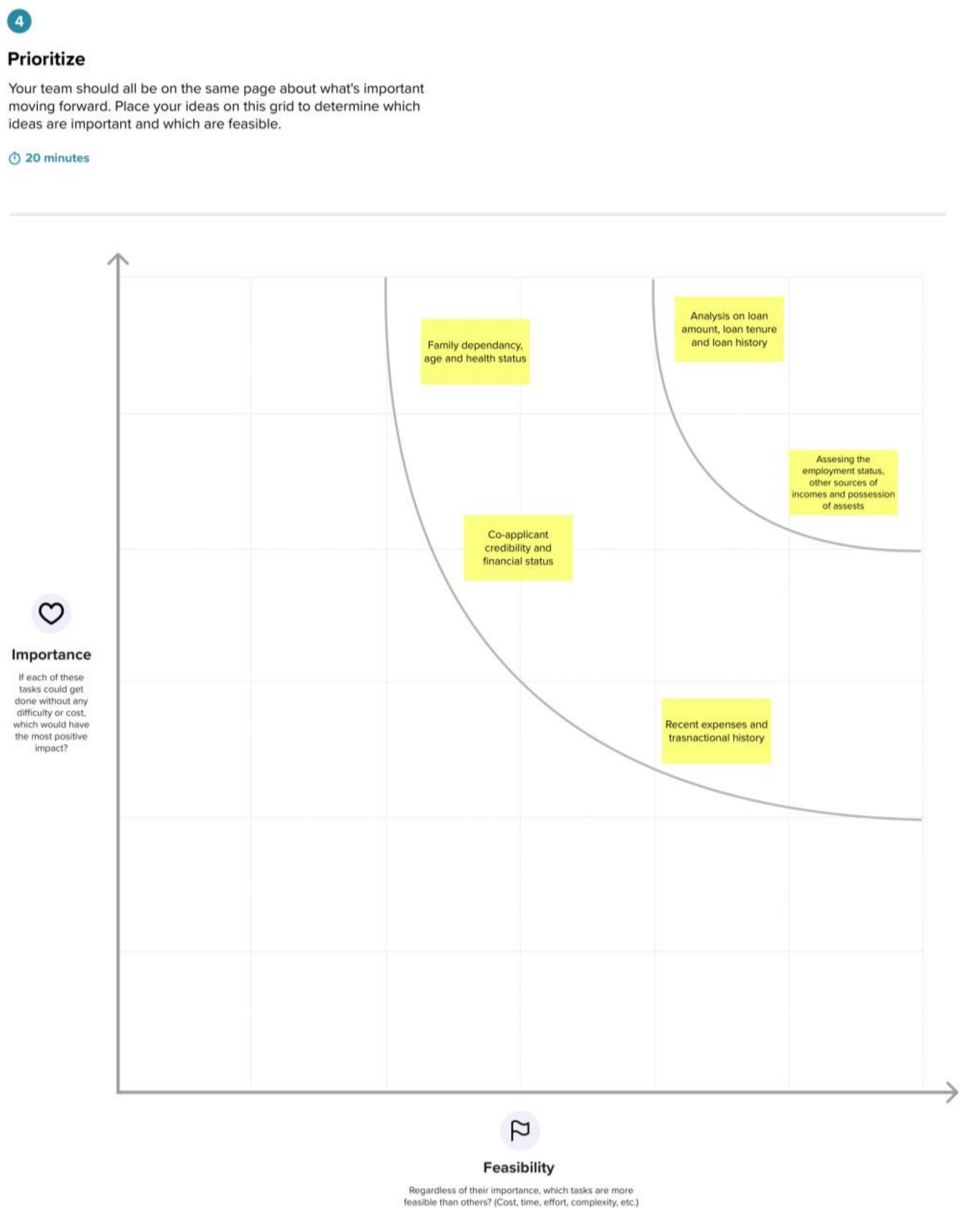

Fig 4. Step-3: Idea Prioritization

PROPOSED SOLUTION

Table 1. Proposed Solution for the given Problem

S. No.	Parameter	Description
1.	Problem Statement (Problemto be solved)	The prediction of credit defaulters is one of the difficult tasks for any bank. But by forecasting the loan defaulters,the banks definitely may reduce their loss by reducing their non-profit assets, so that recovery of approved loans can take place without any loss and it can play as the contributing parameter of the bank statement. But manually assessing the credibility of applicants is a time consuming process and incorrect many a times.

2.	**Idea / Solution description**	**A Machine learning model must be developed to predict the credit defaulters. This model must be trained on previous Loan approval data and their manual credibility checked data. This can be then used to predict the applicant's credibility automatically.**
3.	**Novelty / Uniqueness**	**In this model, the previous manually checked credibility is taken as training data. Once trained it will take Data on Loan history, Financial status and stability, Family status and Co-applicant Credibility as inputs and will provide a Boolean value output for credibility.**
4.	**Social Impact / Customer Satisfaction**	**This model mostly predicts the credibility of a loan applicant accurately, automatically in less time compared to conventional manual checking. This socially helps banks to identify credible loan applicants thus also reduces the loss factor of the Lender (usually Bank). It also speeds up the loan sanctioning process, thus helping the applicants too.**
5.	**Business Model (Revenue Model)**	**A model without human intervention reduces capital investment for the man power and it saves time consumed in this manual process. It will also be accurate than the manual credibility checking process, thus preventing money landing on fraudulent hands.**
6.	**Scalability of the Solution**	**This model can be used with any number of Loan Applicant data and the same algorithm can be used in all the banks or all lenders. With proper organisation and pre-processing of the data about the loan applicant the above proposed solution is completely scalable.**

PROBLEM SOLUTION FIT

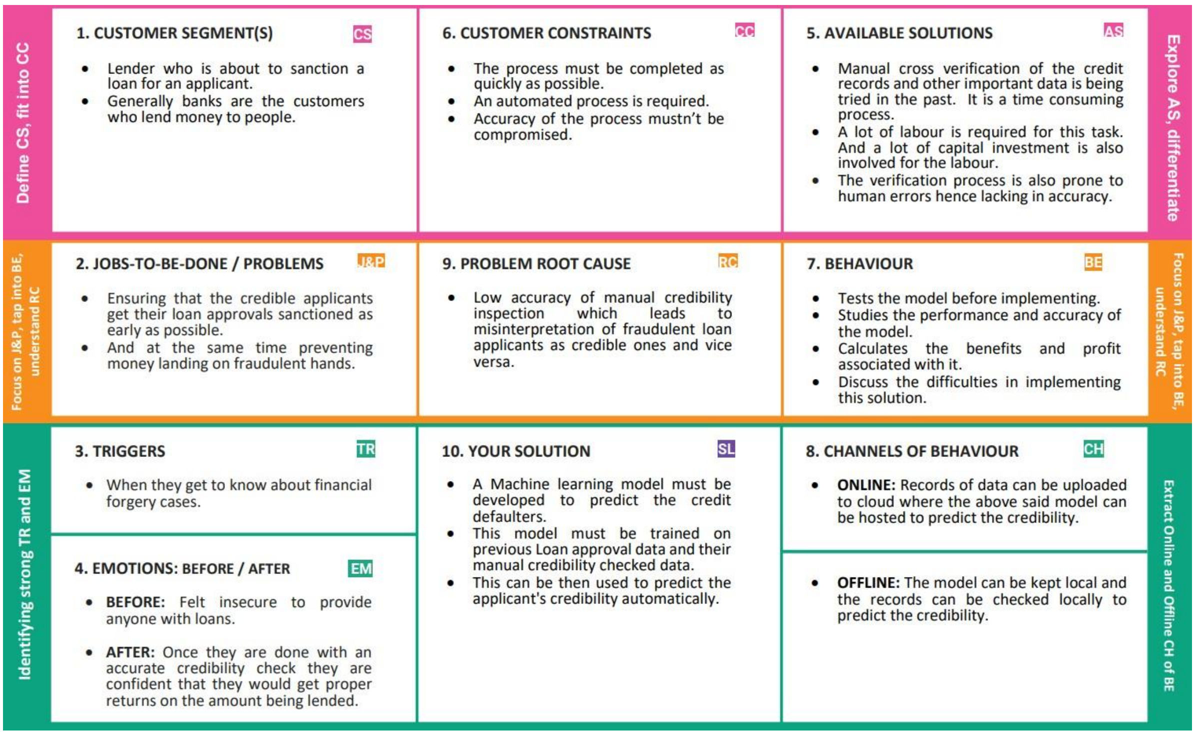

Fig 5. Problem Solution Fit for our Model

4. REQUIREMENT ANALYSIS

FUNCTIONAL REQUIREMENT

Table 2. Functional Requirement for our Model

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 In-Person Confirmation
FR-3	User Requirements	Knowledge on how to Input Data Basic idea on using a Machine Learning Model
FR-4	User Infrastructure	A system with suitable CPU and GPU to support trainingand deployment of ML model.
FR-5	Final Result Visualization	Generated results will be visible to the admin through aWeb server.

NON-FUNCTIONAL REQUIREMENTS

Table 3. Non-Functional Requirement for our Model

FR No.	Non-Functional Requirement	Description
NFR-1	Usability	It can be used by all Lenders (i.e., Banks) and can be trained specifically with respect to the Location’sFinancial Stability.
NFR-2	Security	Storage and Transfer is secure via EncryptionMethodologies.
NFR-3	Reliability	The Predicted Credibility is highly reliable thanManual Identification.
NFR-4	Performance	The Performance relies on Input Dataset.
NFR-5	Availability	Can be made available as a Software.
NFR-6	Scalability	Can be scaled for more branches of the sameLender and the dataset can be shared.

5. PROJECT DESIGN

DATA FLOW DIAGRAMS

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.

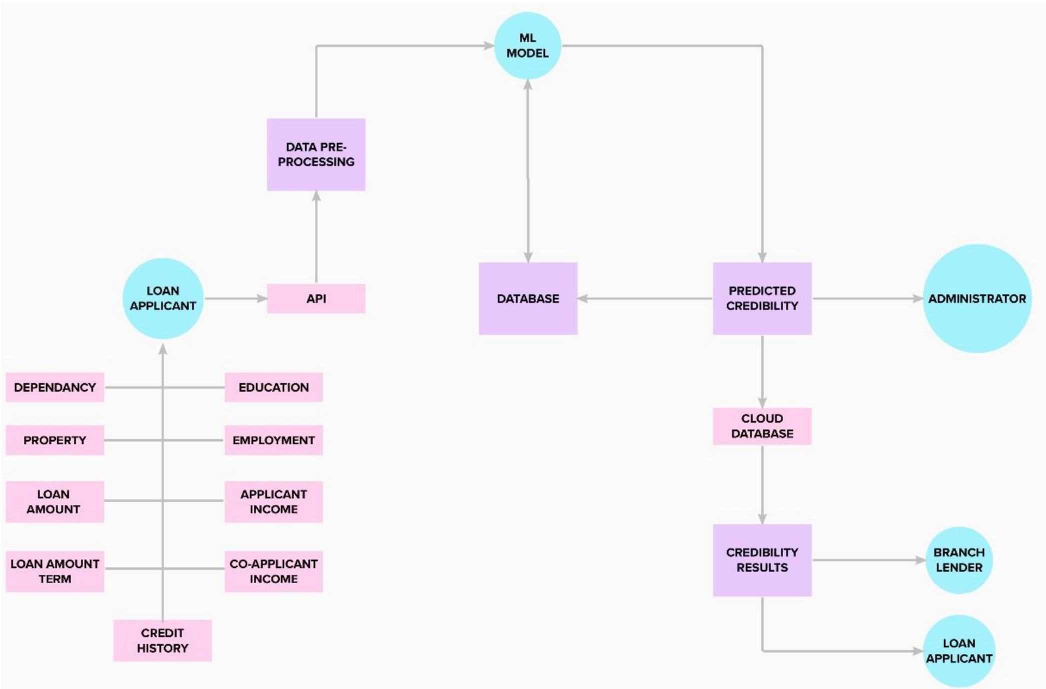

Fig 6. Data Flow diagram of our Proposed Architecture

SOLUTION & TECHNICAL ARCHITECTURE

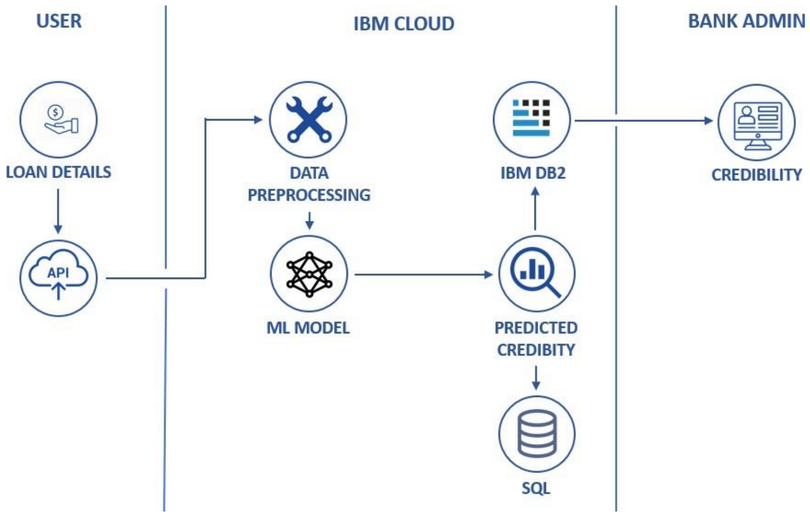

Fig 7. Technical Architecture of our Model

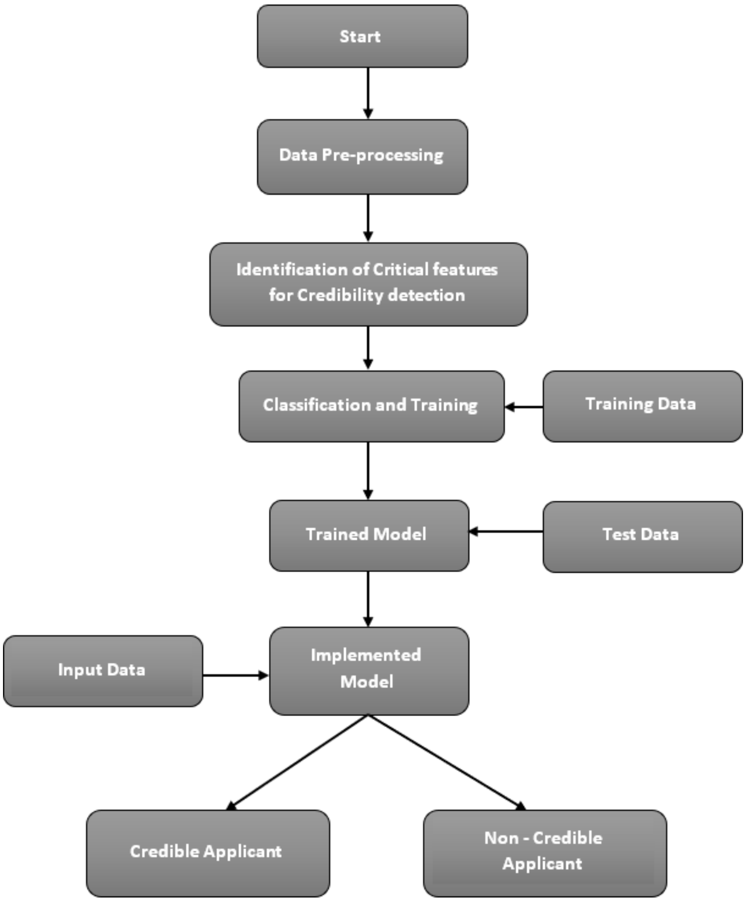

Fig 8. Proposed Solution flow for the Problem Statement

USER STORIES

Table 4. User Stories and Acceptance Criteria

User Type	**Functional Requirement (Epic)**	**User Story Number**	**User Story / Task**	**Acceptance criteria**
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.
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm.
	Login	USN-3	As a user, I can log into the application by entering email & password	I can login to the admin window.
	Dashboard	USN-4	As a user, the credibility results can be checked.	I can visualise the results if details are accurate.
Customer (Web user)	Login	USN-5	As admins, Different Banks and Different branches can have their own admin login and trained model.	Different Admin IDs can be used for different Lender.
	Dashboard	USN-6	As an admin, the credibility result of each loan applicant can be checked.	I can visualise the results if applicant details and database details are accurate.
Customer Care Executive	Help Desk	USN-7	As a Customer Care Executive, I shall attend the calls and guide the user.	I must clearly know the details of the model and the UI.
Administrator	Remote Access	USN-8	I from the main Branch, must have Remote Access to all the Sub-Branches using Cloud.	I manage the data of all the branches and the access must be easy and accurate.
	Maintenance	**USN-9**	**I must be sure that the Servers of all the branches are working in proper condition.**	**I can check the status of servers.**

6. PROJECT PLANNING & SCHEDULING

SPRINT PLANNING & ESTIMATION

Table 5. Sprint Planning and Estimation of our Model

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can registerfor the application by entering my email, password, and confirming my password.	3	High	Deepanraj B Chandru V Poovarasan D Tamilan M
Sprint-1		USN-2	As a user, I will receive confirmation email once Ihave registered for the application	3	High	Deepanraj B Chandru V Poovarasan D Tamilan M
Sprint-1		USN-3	As a user, I can registerfor the application through Gmail	2	Medium	Deepanraj B Chandru V Poovarasan D Tamilan M
Sprint-1	Login	USN-4	As a user, I can log intothe application by entering email & password	2	High	Deepanraj B Chandru V Poovarasan D Tamilan M
Sprint-1	Dashboard	USN-5	As a user, I should be able to access the dashboard with everything I am allowedto use.	2	High	Deepanraj B Chandru V Poovarasan D Tamilan M
Sprint-2	Register	USN-6	As a loan approval officer, I should be able to register myself as oneusing a unique email and password.	3	Medium	Deepanraj B Chandru V Poovarasan D Tamilan M
Sprint-2	Login	USN-7	As a loan approval officer I should be able tologin myself as one using a unique email and password.	3	Medium	Deepanraj B Chandru V Poovarasan D Tamilan M
Sprint-3	Automated analysis of credit history	USN-8	As a loan approval officer, I can access thedashboard where I feedapplications for loan prediction.	2	Medium	Deepanraj B Chandru V Poovarasan D Tamilan M

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-3		USN-9	As a loan approval officer, I can get a decision followed by some details for the decision when I feed anapplication for loan prediction.	3	High	Deepanraj B Chandru V Poovarasan D Tamilan M
Sprint-4	Register	USN-10	As an admin, I should be able to register myself as one using a unique email and password.	2	Medium	Deepanraj B Chandru V Poovarasan D Tamilan M
Sprint-4	Login	USN-11	As an admin I should be able to login myself as one using a unique emailand password.	2	Medium	Deepanraj B Chandru V Poovarasan D Tamilan M
Sprint-4	Dashboard	USN-12	As an admin, I should beable to access the dashboard with everything I am allowed to use.	2	Medium	Deepanraj B Chandru V Poovarasan D Tamilan M

SPRINT DELIVERY SCHEDULE

Table 6. Sprint Delivery Schedule of our Project

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	12	6 Days	27 Oct 2022	01 Nov 2022	12	01 Nov 2022
Sprint-2	6	6 Days	02 Nov 2022	08 Nov 2022	6	08 Nov 2022
Sprint-3	5	6 Days	08 Nov 2022	14 Nov 2022	5	14 Nov 2022
Sprint-4	6	6 Days	14 Nov 2022	19 Nov 2022	6	19 Nov 2022

REPORTS FROM JIRA

7. CODING & SOLUTIONING

FEATURE 1 - Data Set Visualisation

Dataset taken for Training

Loan_ID	Gender	Married	Dependen	Education	Self_Emplo	ApplicantI	Coapplicar	LoanAmou	Loan_Amo	Credit_His	Property_A	Loan_Status
LP001002	Male	No	0	Graduate	No	5849	0		360	1	Urban	Y
LP001003	Male	Yes	1	Graduate	No	4583	1508	128	360	1	Rural	N
LP001005	Male	Yes	0	Graduate	Yes	3000	0	66	360	1	Urban	Y
LP001006	Male	Yes	0	Not Gradu	No	2583	2358	120	360	1	Urban	Y
LP001008	Male	No	0	Graduate	No	6000	0	141	360	1	Urban	Y
LP001011	Male	Yes	2	Graduate	Yes	5417	4196	267	360	1	Urban	Y
LP001013	Male	Yes	0	Not Gradu	No	2333	1516	95	360	1	Urban	Y
LP001014	Male	Yes	3+	Graduate	No	3036	2504	158	360	0	Semiurban	N
LP001018	Male	Yes	2	Graduate	No	4006	1526	168	360	1	Urban	Y
LP001020	Male	Yes	1	Graduate	No	12841	10968	349	360	1	Semiurban	N
LP001024	Male	Yes	2	Graduate	No	3200	700	70	360	1	Urban	Y
LP001027	Male	Yes	2	Graduate		2500	1840	109	360	1	Urban	Y
LP001028	Male	Yes	2	Graduate	No	3073	8106	200	360	1	Urban	Y
LP001029	Male	No	0	Graduate	No	1853	2840	114	360	1	Rural	N
LP001030	Male	Yes	2	Graduate	No	1299	1086	17	120	1	Urban	Y
LP001032	Male	No	0	Graduate	No	4950	0	125	360	1	Urban	Y
LP001034	Male	No	1	Not Gradu	No	3596	0	100	240		Urban	Y
LP001036	Female	No	0	Graduate	No	3510	0	76	360	0	Urban	N
LP001038	Male	Yes	0	Not Gradu	No	4887	0	133	360	1	Rural	N
LP001041	Male	Yes	0	Graduate		2600	3500	115		1	Urban	Y

Fig 10. Visualizing the dataset

This is the Microsoft Excel visualization of the dataset taken for training the model. It has 11 attributes namely Gender, Married, Dependents, Education, Self Employed, Applicant Income, Coapplicant Income, Loan Amount, Loan Amount Term, Credit History, Property Area.

And final result

i.e is the loan status is also stored as a column.

Dataset Description

This below function gives the description of the taken dataset. The description includes features of the dataset like count, mean, std, min, etc.,

```
df.describe()
```

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
count	614.000000	614.000000	592.000000	600.00000	564.000000
mean	5403.459283	1621.245798	146.412162	342.00000	0.842199
std	6109.041673	2926.248369	85.587325	65.12041	0.364878
min	150.000000	0.000000	9.000000	12.00000	0.000000
25%	2877.500000	0.000000	100.000000	360.00000	1.000000
50%	3812.500000	1188.500000	128.000000	360.00000	1.000000
75%	5795.000000	2297.250000	168.000000	360.00000	1.000000
max	81000.000000	41667.000000	700.000000	480.00000	1.000000

Fig 11.Description of the dataset

Univariate Analysis

```python
#plotting the using distplot
plt.figure(figsize=(12,5))
plt.subplot(121)
sns.distplot(df['ApplicantIncome'], color='r')
plt.subplot(122)
sns.distplot(df['Credit_History'])
plt.show()
```

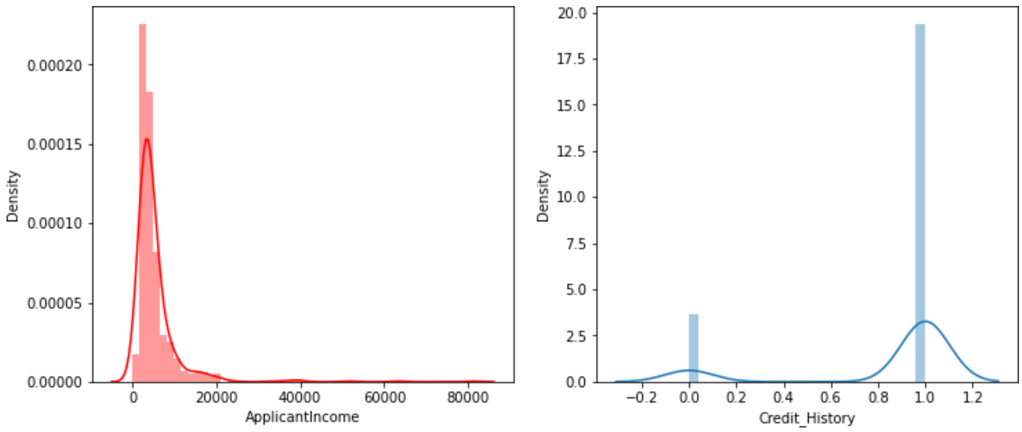

Fig 12. Univariate Analysis - Code and Output

Bivariate Analysis

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

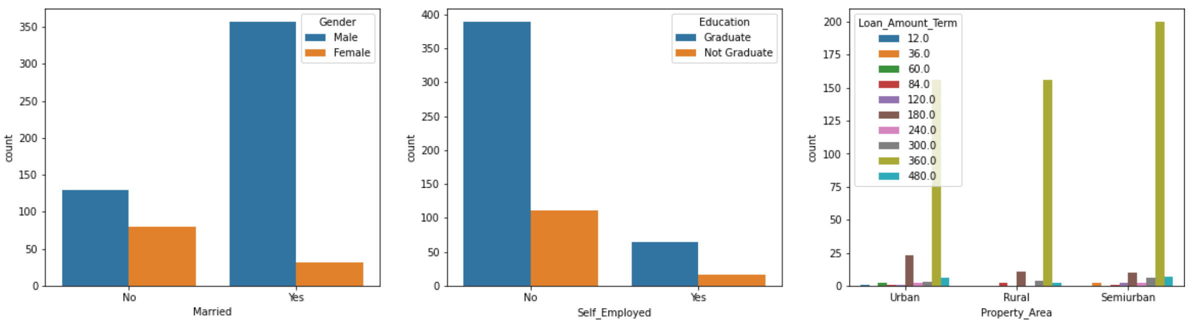

Fig 13. Bivariate Analysis - Code and Output

Multivariate Analysis

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

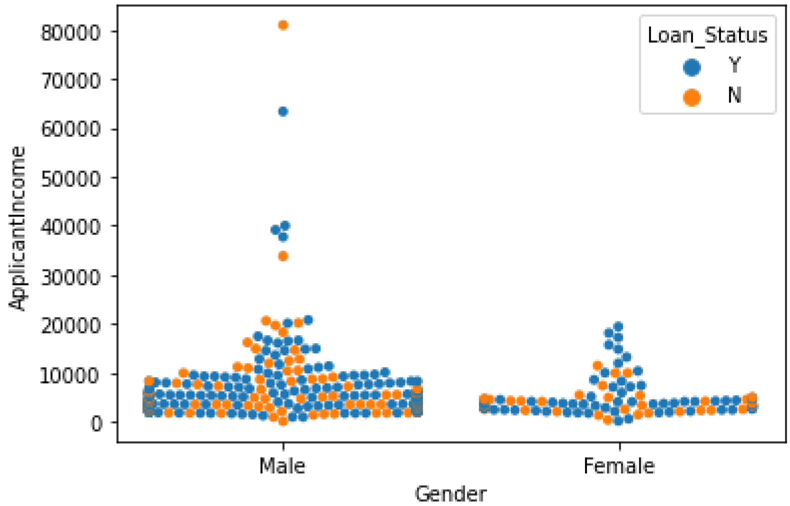

Fig 14. Multivariate Analysis - Code and Output

FEATURE 2 - Data Preprocessing

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Loan_ID            614 non-null    object
 1   Gender             601 non-null    object
 2   Married            611 non-null    object
 3   Dependents         599 non-null    object
 4   Education          614 non-null    object
 5   Self_Employed      582 non-null    object
 6   ApplicantIncome    614 non-null    int64
 7   CoapplicantIncome  614 non-null    float64
 8   LoanAmount         592 non-null    float64
 9   Loan_Amount_Term   600 non-null    float64
 10  Credit_History     564 non-null    float64
 11  Property_Area      614 non-null    object
 12  Loan_Status        614 non-null    object
dtypes: float64(4), int64(1), object(8)
memory usage: 62.5+ KB
```

Fig 15. Information about the dataset

```
df['Gender'] = df['Gender'].fillna(df['Gender'].mode()[0])
df['Married'] = df['Married'].fillna(df['Married'].mode()[0])
#replacing + with space for filling the nan values
df['Dependents']=df['Dependents'].replace('3+',3)
df['Dependents'] = df['Dependents'].fillna(df['Dependents'].mode()[0])
df['Self_Employed'] = df['Self_Employed'].fillna(df['Self_Employed'].mode()[0])
df['LoanAmount'] = df['LoanAmount'].fillna(df['LoanAmount']. mode()[0])
df['Loan_Amount_Term'] = df['Loan_Amount_Term'].fillna(df['Loan_Amount_Term'].mode()[0])
df['Credit_History'] = df['Credit_History'].fillna(df['Credit_History'].mode()[0])
```

Fig 16. Removing Null values in the dataset

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
df.Gender=le.fit_transform(df.Gender)
df.Loan_Status=le.fit_transform(df.Loan_Status)
df.Married=le.fit_transform(df.Married)
df.Education=le.fit_transform(df.Education)
df.Self_Employed=le.fit_transform(df.Self_Employed)
df.Property_Area=le.fit_transform(df.Property_Area)
```

```
#changing the datype of each float column to int
df['Gender']=df['Gender'].astype('int64')
df['Married']=df['Married'].astype('int64')
df['Dependents']=df['Dependents'].astype('int64')
df['Self_Employed']=df['Self_Employed'].astype('int64')
df['CoapplicantIncome']=df['CoapplicantIncome'].astype('int64')
df['LoanAmount']=df['LoanAmount'].astype('int64')
df['Loan_Amount_Term']=df['Loan_Amount_Term'].astype('int64')
df['Credit_History']=df['Credit_History'].astype('int64')
```

Fig 17. Handling Categorical Values

```
#Balancing the dfset by using smote
from imblearn.combine import SMOTETomek
smote = SMOTETomek (0.95)
y = df['Loan_Status']
x = df.drop(columns=["Loan_ID",'Loan_Status'], axis=1)
x_bal,y_bal =smote.fit_resample(x,y)
print(y.value_counts())
print(y_bal.value_counts())
```

```
1    422
0    192
Name: Loan_Status, dtype: int64
1    352
0    330
Name: Loan_Status, dtype: int64
```

Fig 18. Balancing the dataset

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

Fig 19. Scaling the dataset

FEATURE 3 - Training the models

```
train,test = train_test_split(final_df, test_size=0.33, random_state=42)
```

```
train.to_csv('train.csv',encoding='utf-8',index=False)
test.to_csv('test.csv',encoding='utf-8',index=False)
```

```
x=final_df.drop(["Loan_Status"],axis=1)
y=final_df.Loan_Status
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=42)
```

Fig 20. Splitting the dataset

```
import sklearn
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import GradientBoostingClassifier, RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import RandomizedSearchCV
import imblearn
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, f1_score
```

Fig 21. Importing various ML models

Comparing the models

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

```
***DecisionTreeClassifier***
Confusion matrix
[[53 18]
 [12 56]]
Classification report
              precision    recall  f1-score   support

           0       0.82      0.75      0.78        71
           1       0.76      0.82      0.79        68

    accuracy                           0.78       139
   macro avg       0.79      0.79      0.78       139
weighted avg       0.79      0.78      0.78       139

score
0.7841726618705036
```

Fig 22. Decision Tree Classifier

```
randomForest(x_train, x_test, y_train, y_test)

***RandomForestClassifier***
Confusion matrix
[[48 23]
 [ 3 65]]
Classification report
              precision    recall  f1-score   support

           0       0.94      0.68      0.79        71
           1       0.74      0.96      0.83        68

    accuracy                           0.81       139
   macro avg       0.84      0.82      0.81       139
weighted avg       0.84      0.81      0.81       139

score
0.8129496402877698
```

Fig 23. Random Forest Classifier

```
KNN(x_train, x_test, y_train, y_test)

***KNeighborsClassifier***
Confusion matrix
[[50 21]
 [21 47]]
Classification report
              precision    recall  f1-score   support

           0       0.70      0.70      0.70        71
           1       0.69      0.69      0.69        68

    accuracy                           0.70       139
   macro avg       0.70      0.70      0.70       139
weighted avg       0.70      0.70      0.70       139

score
0.697841726618705
```

Fig 24. K Neighbour Classifier

```
xgboost(x_train, x_test, y_train, y_test)

***Gradient BoostingClassifier***
Confusion matrix
[[46 25]
 [ 4 64]]
Classification report
              precision    recall  f1-score   support

           0       0.92      0.65      0.76        71
           1       0.72      0.94      0.82        68

    accuracy                           0.79       139
   macro avg       0.82      0.79      0.79       139
weighted avg       0.82      0.79      0.79       139

score
0.7913669064748201
```

Fig 25. Gradient Boost Classifier

Finalizing the ML model

The Random Forest Classifier model is finalized as the model based on the comparision scores and it is trained and exported as pkl file.

```
from sklearn.model_selection import cross_val_score
rf = RandomForestClassifier()
rf.fit(x_train,y_train)
yPred = rf.predict(x_test)
f1_score(yPred,y_test, average='weighted')
cv = cross_val_score(rf,x,y,cv=5)
np.mean(cv)

0.7998331769367115
```

```
pickle.dump(rf,open('rdf.pkl','wb'))
```

Fig 26. Exporting the final trained model

FEATURE 4 - Deploying the model in IBM Cloud

```
!pip install -U ibm-watson-machine-learning

Requirement already satisfied: ibm-watson-machine-learning in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (1.0.255)
Collecting ibm-watson-machine-learning
  Downloading ibm_watson_machine_learning-1.0.257-py3-none-any.whl (1.8 MB)
     |████████████████████████████████| 1.8 MB 31.6 MB/s eta 0:00:01
```

Fig 27. Installing Necessary Packages

Here the required library of IBM Watson Machine Learning is getting installed.

```
In [20]: from ibm_watson_machine_learning import APIClient
         import json

In [21]: wml_credentials = {
             "apikey" : "yjJRKquYkWeXxU3CGuvFC0Q1f29M8lpidj6PZ8B0RQgY",
             "url" : "https://eu-gb.ml.cloud.ibm.com"
         }

In [22]: wml_client = APIClient(wml_credentials)

In [24]: wml_client.spaces.list()

    Note: 'limit' is not provided. Only first 50 records will be displayed if the number of records exceed 50
    ------------------------------------  ---------------  ------------------------
    ID                                    NAME             CREATED
    99b1a4a9-7cc5-4852-9388-f8907fe20de7  sl_223_ibm_space 2022-11-16T10:44:29.321Z
    ------------------------------------  ---------------  ------------------------

In [25]: SPACE_ID= "99b1a4a9-7cc5-4852-9388-f8907fe20de7"

In [26]: wml_client.set.default_space(SPACE_ID)

Out[26]: 'SUCCESS'
```

Fig 28. Authentication and Space Setting

Using the unique API key generated in IBM Cloud and mentioning our server location. Usingthe API credentials a new space is created in IBM Watson. The space has its unique Space id.

```
In [28]: import sklearn
         sklearn.__version__

Out[28]: '1.0.2'

In [29]: MODEL_NAME = 'Model_building_SL_223_IBM'
         DEPLOYMENT_NAME = 'Smart-Lender_223_IBM'
         DEMO_MODEL = rf

In [30]: software_spec_uid = wml_client.software_specifications.get_id_by_name('runtime-22.1-py3.9')

In [31]: model_props = {
             wml_client.repository.ModelMetaNames.NAME: MODEL_NAME,
             wml_client.repository.ModelMetaNames.TYPE: 'scikit-learn_1.0',
             wml_client.repository.ModelMetaNames.SOFTWARE_SPEC_UID: software_spec_uid
         }

In [32]: model_details = wml_client.repository.store_model(
             model=DEMO_MODEL,
             meta_props=model_props,
             training_data=x_train,
             training_target=y_train
         )
```

Fig 29. Importing the model and setting up it

Downloading the required ML model. Looking for the version that is being supported by IBM and downloading the correct version. Creating a new deployment space for the model. To set up the model requirements and link it to the deployment space. Saving the model to the space by mentioningthe attributes of the model.

```
In [33]: model_details
```

```
Out[33]: {'entity': {'hybrid_pipeline_software_specs': [],
  'label_column': 'Loan_Status',
  'schemas': {'input': [{'fields': [{'name': 'Gender', 'type': 'int64'},
      {'name': 'Married', 'type': 'int64'},
      {'name': 'Dependents', 'type': 'int64'},
      {'name': 'Education', 'type': 'int64'},
      {'name': 'Self_Employed', 'type': 'int64'},
      {'name': 'ApplicantIncome', 'type': 'int64'},
      {'name': 'CoapplicantIncome', 'type': 'int64'},
      {'name': 'LoanAmount', 'type': 'int64'},
      {'name': 'Loan_Amount_Term', 'type': 'int64'},
      {'name': 'Credit_History', 'type': 'int64'},
      {'name': 'Property_Area', 'type': 'int64'}],
     'id': '1',
     'type': 'struct'}],
   'output': []},
  'software_spec': {'id': '12b83a17-24d8-5082-900f-0ab31fbfd3cb',
   'name': 'runtime-22.1-py3.9'},
  'type': 'scikit-learn_1.0'},
 'metadata': {'created_at': '2022-11-16T10:48:43.132Z',
  'id': '03542d22-55b9-4830-af6f-c000da875e4e',
  'modified_at': '2022-11-16T10:48:46.959Z',
  'name': 'Model_building_SL_223_IBM',
  'owner': 'IBMid-6620042VBA',
  'resource_key': 'cdb1c157-cfd2-4271-a4a5-9f28198439ca',
  'space_id': '99b1a4a9-7cc5-4852-9388-f8907fe20de7'},
 'system': {'warnings': []}}
```

Fig 30. Model Details

```
In [34]: model_id = wml_client.repository.get_model_id(model_details)
         model_id

Out[34]: '03542d22-55b9-4830-af6f-c000da875e4e'

In [37]: deployment_props = {
             wml_client.deployments.ConfigurationMetaNames.NAME:DEPLOYMENT_NAME,
             wml_client.deployments.ConfigurationMetaNames.ONLINE: {}
         }

In [38]: deployment = wml_client.deployments.create(
             artifact_uid=model_id,
             meta_props=deployment_props
         )
```

```
#######################################################################################

Synchronous deployment creation for uid: '03542d22-55b9-4830-af6f-c000da875e4e' started

#######################################################################################


initializing
Note: online_url is deprecated and will be removed in a future release. Use serving_urls instead.

ready


------------------------------------------------------------------------------------------------
Successfully finished deployment creation, deployment_uid='84569608-2735-48f9-991a-fb61d5a0c2af'
------------------------------------------------------------------------------------------------
```

Fig 31. Deployment in IBM Cloud

To set the configuration of the deployment. Giving the name for the deployment in IBM Watson. Deploying the model in IBM Cloud using model id. An id is created for the model using which the modelcan be accessed online

Web Page Design

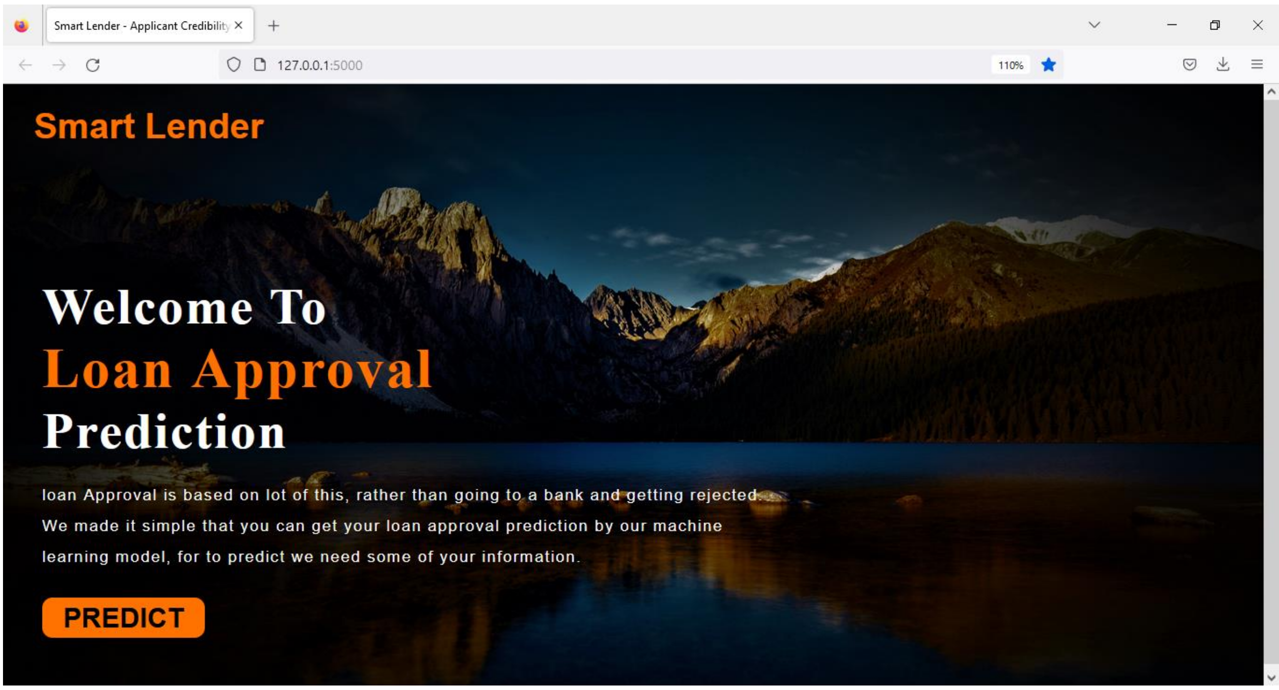

Fig 38. Home Page

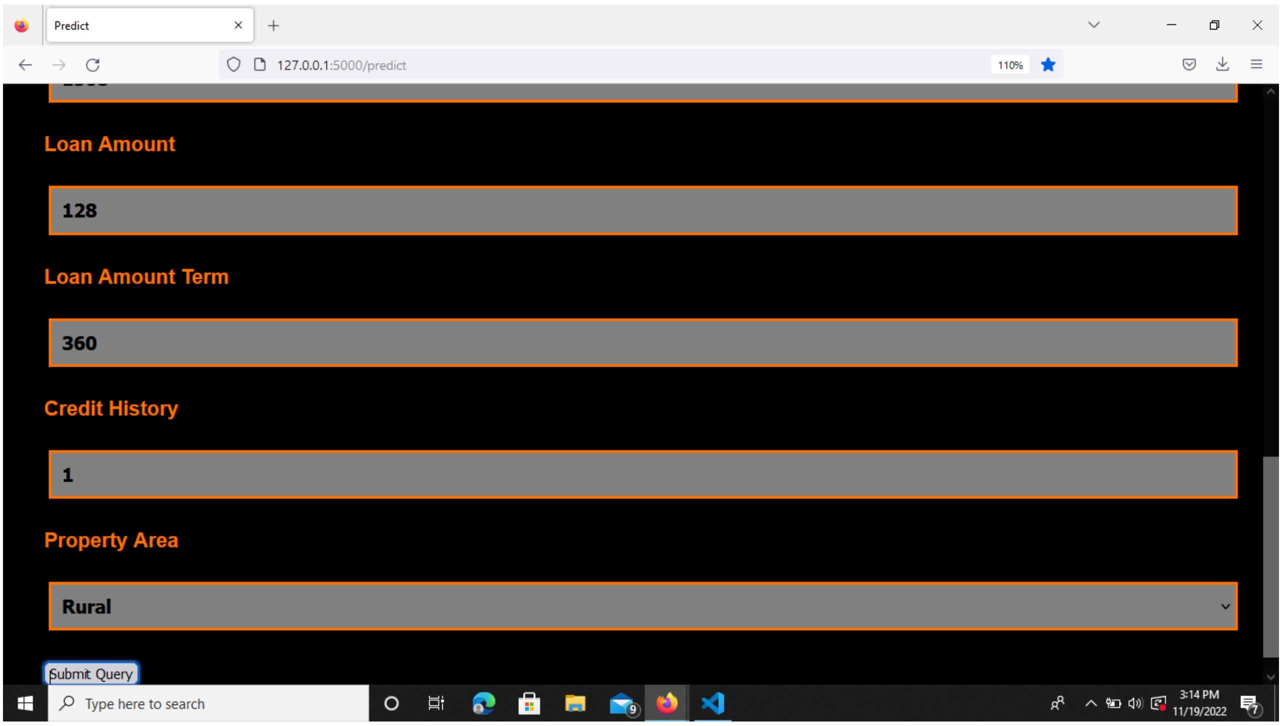

Fig 39. Predict Page

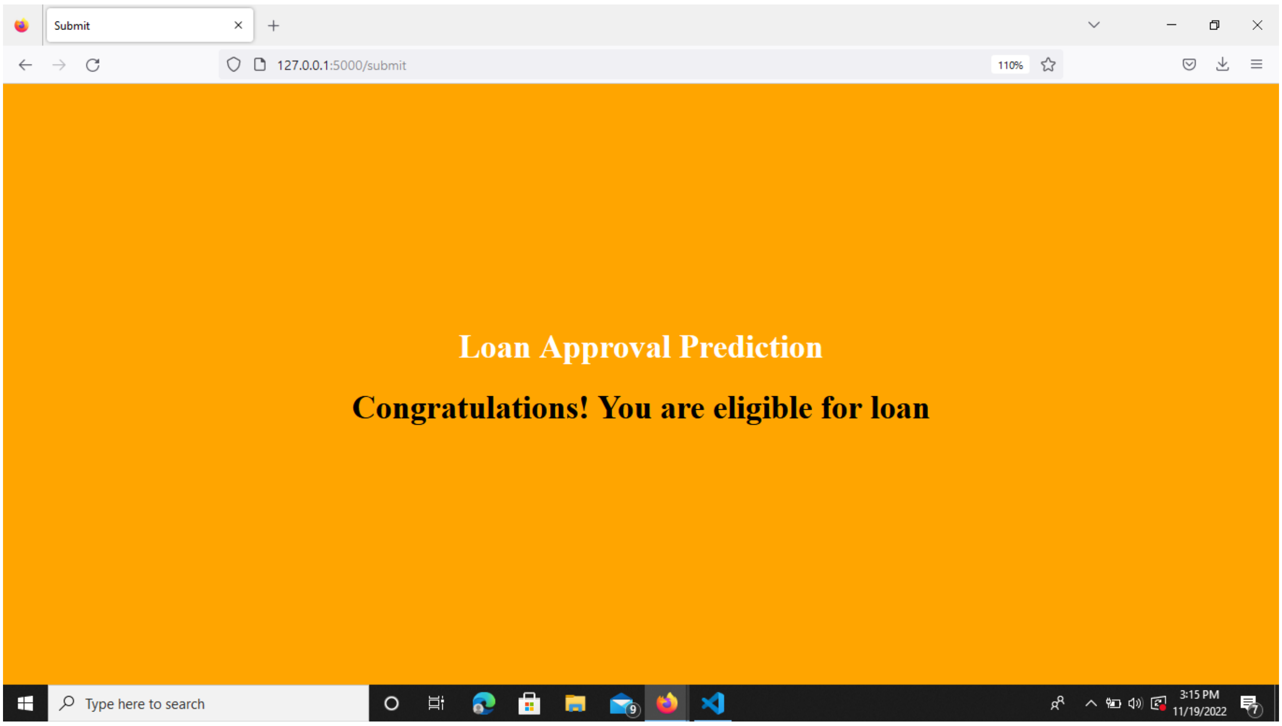

Fig 40. Submit Page

8. TESTING

TEST CASES

Table 7. Sample Test Cases for Testing

Lender/ Applicant	Gender	Marital Status	Dependants	Education	Self Employed	Applicant Income	Co-applicant Income	Loan Amount	Loan Amount Term	Credit History	Property Area	Loan Status
Applicant	Male	No	0	Graduate	No	54170	0	168000	1080	Yes	Urban	Y
Lender	Male	No	0	Graduate	Yes	69500	0	175000	1080	Yes	Semi-urban	Y
Lender	Male	Yes	0	Graduate	No	26980	20340	212000	580	No	Semi-urban	N
Applicant	Male	Yes	2	Graduate	No	11757	0	187000	780	No	Rural	N
Lender	Female	Yes	0	Graduate	No	23300	44860	1000000	360	Yes	Semi-urban	N
Applicant	Female	Yes	2	Graduate	No	14866	0	700000	1500	Yes	Urban	Y
Applicant	Male	Yes	1	Graduate	No	153800	41250	300000	1000	Yes	Urban	Y
Lender	Female	No	0	Graduate	No	10000	16660	225000	500	No	Rural	N
Lender	Male	Yes	0	Graduate	No	48600	83000	1250000	2360	Yes	Semi-urban	Y

USER ACCEPTANCE TESTING

The website has been tested using IBM platform. We had taken inputs from the users who have tested this website and have done modifications to satisfy everones needs. The users found the interface very easy to use. The Web pages were colourful and attractive. There was no unnecessary details in the web page. It was clean and simple that any new user could master it. The data input format was also simple. The user need not enter any unit. He could simply enter the value. The prediction time is fairly low at an average time of 3 seconds. This delay primarily varies depending on the internet connectivity. The model has been hosted in the IBM cloud. Thus with the API available, the model can be accessed remotely from any system provided IBM access key is given. The model predicts the loan status in an more accurate manner. We have two provisions. An applicant can also use this website to predict his loan application acceptance probability. Also a banker/lender can also use this to verify whether the applicant can provided with the loan amount requested. The users are satisfied with the predicted results as they are easier to interrupt. Various inputs have been given by the users to test theconsistency of the model. The model proved itself and all the users accepted the model as a reliable and convenient.

9. RESULTS

PERFORMANCE METRICS

The RandomForestClassifier ML model that we have used here has better performance in speed and accuracy compared to other models. We have compared the performance metrics of 4 models and selected this as the best for the application. The model performed well for all the test cases. The API developed also performed good with no glitches or lag found .

10. ADVANTAGES & DISADVANTAGES

Advantages

This model is trained based on the previous manually assesed loan applicant's datasets. So itasseses new applications more accurately. It takes a lot of parameters as input for prediction, which makes the model more effective in prediction. Since the dataset is balanced the model trained is also balanced and produce more accurate, unbiased results. The user interface is simple and elegant, hence making it easier for the end user to utilise it. It serves as a boon to both the lender and loan applicant in accessing the loan application. It saves a lot of time and manual labour involved in this process. Withthis website's prediction values in hand, the applicant can have a confidence in applying for a loan amount. And it is the same in case of the lender, he can confidently lend money to an applicant.

Disadvantages

This model must extensively reach every person, so that they can make use of this. Massive implementation of this model in all banks might have practical difficulties. Some banks will have some privacy policies which may not allow such implementation in their system. Some banks might need some extra checks before providing loan to a person. In that case they must remodify the model. So asof now this can be a basic gatepass for the lenders to process a loan application.

11. CONCLUSION

The RandomForestClassifier ML model that has been used above performs well for our dataset. The model is fast and consumes less resources. The API developed is also simple and user friendly. By using this model, we could access the credibility of a loan applicant provided the required input data.This saves the time and prevents money landing on fraudulent hands. The model is not 100% accurate but it performs sufficiently well. It can be concluded that the output of this model can be taken as a very important and basic guideline in deciding the credibility of the applicant. Some high priority ground check is unavoidable. So we can proceed to that ground check once we receive a green sign from this model.

12. FUTURE SCOPE

The further works that can be done in this project is to include few more features in model training to study the effect on the prediction. A long history of data (dataset of more than 3 years) can be used for training for increased accuracy. The application can be upgraded such that the input values are fetched directly from the application file and then fed to the model rather than the user entering it manually. A login systems for banks can be developed, so that each bank can have its own login hencemaking their applicants data more secure.

13. APPENDIX

SOURCE CODE DATA PREPROCESSING:

```
from pyexpat import model
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import GradientBoostingClassifier, RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import cross_val_score
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.model_selection import RandomizedSearchCV
import imblearn
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix, f1_score
from imblearn.combine import SMOTETomek
from sklearn.preprocessing import LabelEncoder
import pickle




data = pd.read_csv('loan_dataset.csv')
print(data)


names = ['Gender', 'Married_Status', 'Dependents', 'Education',
'Self_Employed', 'Applicant_Income', 'Co_Applicant_Income', 'Loan_Amount',
  'Loan_Amount_Term', 'Credit_History', 'Property_Area', 'Loan_Status']

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

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

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

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

sns.swarmplot(data['Gender'])
sns.swarmplot(data['Applicant_Income'])
sns.swarmplot(data['Loan_Status'])
plt.show()

print(data.describe())

data.info()


print(data.isnull().sum())

data['Gender'] = data['Gender'].fillna(data['Gender'].mode()[0])
data['Married_Status'] =
data['Married_Status'].fillna(data['Married_Status'].mode()[0])
data['Dependents'] = data['Dependents'].str.replace('+','3')
data['Dependents'] = data['Dependents'].fillna(data['Dependents'].mode()[0])
data['Self_Employed'] =
data['Self_Employed'].fillna(data['Self_Employed'].mode()[0])
data['Loan_Amount'] = data['Loan_Amount'].fillna(data['Loan_Amount'].mode()[0])
data['Loan_Amount_Term'] =
data['Loan_Amount_Term'].fillna(data['Loan_Amount_Term'].mode()[0])
data['Credit_History'] =
data['Credit_History'].fillna(data['Credit_History'].mode()[0])

print(data.isnull().sum())

le=LabelEncoder()
data.Gender=le.fit_transform(data.Gender)
data.Loan_Status=le.fit_transform(data.Loan_Status)
data.Married_Status=le.fit_transform(data.Married_Status)
data.Education=le.fit_transform(data.Education)
data.Self_Employed=le.fit_transform(data.Self_Employed)
data.Property_Area=le.fit_transform(data.Property_Area)

print(data)


data['Gender']=data['Gender'].astype('int64')
data['Married_Status']=data['Married_Status'].astype('int64')
data['Dependents']=data['Dependents'].astype('int64')
data['Self_Employed']=data['Self_Employed'].astype('int64')
```

```
data['Co_Applicant_Income']=data['Co_Applicant_Income'].astype('int64')
data['Loan_Amount']=data['Loan_Amount'].astype('int64')
data['Loan_Amount_Term']=data['Loan_Amount_Term'].astype('int64')
data['Credit_History']=data['Credit_History'].astype('int64')

smote = SMOTETomek (0.95)
y = data['Loan_Status']
x = data.drop(columns=["Loan_ID",'Loan_Status'], axis=1)
x_bal,y_bal =smote.fit_resample(x,y)
print(y.value_counts())
print(y_bal.value_counts())


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

print(x_bal_scaled)


x_train, x_test, y_train, y_test = train_test_split(x_bal,y_bal,
test_size=0.33, random_state=42)

def decisionTree(x_train, x_test, y_train, y_test):
    dt=DecisionTreeClassifier()
    dt.fit(x_train,y_train)
    yPred = dt.predict(x_test)
    print('***DecisionTreeClassifier***')
    print('Confusion matrix')
    print(confusion_matrix(y_test,yPred))
    print('Classification report')
    print(classification_report(y_test,yPred))

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

def rf(x_train, x_test, y_train, y_test):
    rf = RandomForestClassifier()
    rf.fit(x_train, y_train)
    yPred = rf.predict(x_test)
    print('***RandomForestClassification***')
    print('Confusion matrix')
    print(confusion_matrix(y_test,yPred))
    print('Classification report')
    print(classification_report(y_test,yPred))

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

```
def KNN(x_train, x_test, y_train, y_test):
    knn = KNeighborsClassifier()
    knn.fit(x_train,y_train)
    yPred = knn.predict(x_test)
    print('***KNeighborsClassifier***')
    print('Confusion matrix')
    print(confusion_matrix(y_test,yPred))
    print('Classification report')
    print(classification_report(y_test,yPred))

print(KNN(x_train,x_test,y_train,y_test))

def xgboost(x_train, x_test, y_train, y_test):
    xg = GradientBoostingClassifier()
    xg.fit(x_train,y_train)
    yPred = xgboost.Predict(x_test)
    print('***GradientBoostingClassifier***')
    print('Confusion matrix')
    print(confusion_matrix(y_test,yPred))
    print('Classification report')
    print(classification_report(y_test,yPred))

print(xgboost(x_train,x_test,y_train,y_test))


rf = RandomForestClassifier()
rf.fit(x_train,y_train)
yPred =  rf.predict(x_test)
print(f1_score(yPred,y_test,average='weighted'))
cv = cross_val_score(rf,x,y,cv=5)
print(np.mean(cv))

pickle.dump(open("rdf.pkl", "wb"))
```

TRAINING THE MODEL ON IBM:

```
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()
```

```
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.fit_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))
```

```python
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))
```

```python
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))
```

```python
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)
```

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

```
***DecisionTreeClassifier***
Confusion matrix
[[53 18]
 [12 56]]
Classification report
              precision    recall  f1-score   support

           0       0.82      0.75      0.78        71
           1       0.76      0.82      0.79        68

    accuracy                           0.78       139
   macro avg       0.79      0.79      0.78       139
weighted avg       0.79      0.78      0.78       139

score
0.7841726618705036
```

```
randomForest(train_x,test_x,train_y,test_y)
```

```
randomForest(x_train, x_test, y_train, y_test)
```

```
***RandomForestClassifier***
Confusion matrix
[[48 23]
 [ 3 65]]
Classification report
              precision    recall  f1-score   support

           0       0.94      0.68      0.79        71
           1       0.74      0.96      0.83        68

    accuracy                           0.81       139
   macro avg       0.84      0.82      0.81       139
weighted avg       0.84      0.81      0.81       139

score
0.8129496402877698
```

```
knn(train_x,test_x,train_y,test_y)
```

```
KNN(x_train, x_test, y_train, y_test)
```

```
***KNeighborsClassifier***
Confusion matrix
[[50 21]
 [21 47]]
Classification report
              precision    recall  f1-score   support

           0       0.70      0.70      0.70        71
           1       0.69      0.69      0.69        68

    accuracy                           0.70       139
   macro avg       0.70      0.70      0.70       139
weighted avg       0.70      0.70      0.70       139

score
0.697841726618705
```

```
xgboost(train_x,test_x,train_y,test_y)
```

```
xgboost(x_train, x_test, y_train, y_test)
```

```
***Gradient BoostingClassifier***
Confusion matrix
[[46 25]
 [ 4 64]]
Classification report
              precision    recall  f1-score   support

           0       0.92      0.65      0.76        71
           1       0.72      0.94      0.82        68

    accuracy                           0.79       139
   macro avg       0.82      0.79      0.79       139
weighted avg       0.82      0.79      0.79       139

score
0.7913669064748201
```

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

```
f1_score(ypred,test_y,average='weighted')
```

```
0.7742005478857578
```

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

```
np.mean(cv)
```

```
0.8164946303826504
```

```
import joblib
joblib.dump(rf,'model.pkl')
```

```
['model.pkl']
```

```
!tar -zcvf model.tgz "model.pkl"
```

```
!pip install ibm-watson-machine-learning
```

Requirement already satisfied: ibm-watson-machine-learning in c:\users\chandru\anaconda3\lib\site-packages (1.0.257) Requirement already satisfied: certifi in c:\users\chandru\anaconda3\lib\site-packages (from ibm-watson-machine-learning) (2022.9.14) Requirement already satisfied: requests in c:\users\chandru\anaconda3\lib\site-packages (from ibm-watson-machine-learning) (2.28.1) Requirement already satisfied: packaging in c:\users\chandru\anaconda3\lib\site-packages (from ibm-watson-machine-learning) (21.3) Requirement already satisfied: importlib-metadata in c:\users\chandru\anaconda3\lib\site-packages (from ibm-watson-machine-learning) (4.11.3) Requirement already satisfied: tabulate in c:\users\chandru\anaconda3\lib\site-packages (from ibm-watson-machine-learning) (0.8.10) Requirement already satisfied: ibm-cos-sdk==2.11.* in c:\users\chandru\anaconda3\lib\site-packages (from ibm-watson-machine-learning) (2.11.0) Requirement already satisfied: urllib3 in c:\users\chandru\anaconda3\lib\site-packages (from ibm-watson-machine-learning) (1.26.11) Requirement already satisfied: pandas<1.5.0,>=0.24.2 in c:\users\chandru\anaconda3\lib\site-packages (from ibm-watson-machine-learning) (1.4.4) Requirement already satisfied: lomond in c:\users\chandru\anaconda3\lib\site-packages (from ibm-watson-machine-learning) (0.3.3) Requirement already satisfied: ibm-cos-sdk-s3transfer==2.11.0 in c:\users\chandru\anaconda3\lib\site-packages (from ibm-cos-sdk==2.11.*->ibm-watson-machine-learning) (2.11.0) Requirement already satisfied: ibm-cos-sdk-core==2.11.0 in c:\users\chandru\anaconda3\lib\site-packages (from ibm-cos-sdk==2.11.*->ibm-watson-machine-learning) (2.11.0) Requirement already satisfied: jmespath<1.0.0,>=0.7.1 in c:\users\chandru\anaconda3\lib\site-packages (from ibm-cos-sdk==2.11.*->ibm-watson-machine-learning) (0.10.0) Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in c:\users\chandru\anaconda3\lib\site-packages (from ibm-cos-sdk-core==2.11.0->ibm-cos-sdk==2.11.*->ibm-watson-machine-learning) (2.8.2) Requirement already satisfied: numpy>=1.18.5 in c:\users\chandru\anaconda3\lib\site-packages (from pandas<1.5.0,>=0.24.2->ibm-watson-machine-learning) (1.21.5) Requirement already satisfied: pytz>=2020.1 in

c:\users\chandru\anaconda3\lib\site-packages (from pandas<1.5.0,>=0.24.2->ibm-watson-machine-learning) (2022.1) Requirement already satisfied: charset-normalizer<3,>=2 in c:\users\chandru\anaconda3\lib\site-packages (from requests->ibm-watson-machine-learning) (2.0.4) Requirement already satisfied: idna<4,>=2.5 in c:\users\chandru\anaconda3\lib\site-packages (from requests->ibm-watson-machine-learning) (3.3) Requirement already satisfied: zipp>=0.5 in c:\users\chandru\anaconda3\lib\site-packages (from importlib-metadata->ibm-watson-machine-learning) (3.8.0) Requirement already satisfied: six>=1.10.0 in c:\users\chandru\anaconda3\lib\site-packages (from lomond->ibm-watson-machine-learning) (1.16.0) Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in c:\users\chandru\anaconda3\lib\site-packages (from packaging->ibm-watson-machine-learning) (3.0.9)

```
from ibm_watson_machine_learning import APIClient
```

```
wml_credentials = {
    "url": "https://us-south.ml.cloud.ibm.com",
    "apikey": "BfkRcHp3hwoDe60vOFQ5AwdQQOK6ZK28dj6uDbzRsKqN"
}

client = APIClient(wml_credentials)
```

```
# function to store the model in deployment space
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, "space")
print("Space UID - " + space_uid)
```

```
Space UID - 9aaafe72-8656-4fb6-8c50-b0ce72cfd99b
```

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

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

```
Output exceeds the size limit. Open the full output data in a text editor
------------------------------ ------------------------------------ ---- NAME
ASSET_ID TYPE default_py3.6 0062b8c9-8b7d-44a0-a9b9-46c416adcbd9 base kernel-
spark3.2-scala2.12 020d69ce-7ac1-5e68-ac1a-31189867356a base pytorch-onnx_1.3-
py3.7-edt 069ea134-3346-5748-b513-49120e15d288 base scikit-learn_0.20-py3.6
09c5a1d0-9c1e-4473-a344-eb7b665ff687 base spark-mllib_3.0-scala_2.12 09f4cff0-
90a7-5899-b9ed-1ef348aebdee base pytorch-onnx_rt22.1-py3.9 0b848dd4-e681-5599-
be41-b5f6fccc6471 base ai-function_0.1-py3.6 0cdb0f1e-5376-4f4d-92dd-
```

```
da3b69aa9bda base shiny-r3.6 0e6e79df-875e-4f24-8ae9-62dcc2148306 base
tensorflow_2.4-py3.7-horovod 1092590a-307d-563d-9b62-4eb7d64b3f22 base
pytorch_1.1-py3.6 10ac12d6-6b30-4ccd-8392-3e922c096a92 base tensorflow_1.15-
py3.6-ddl 111e41b3-de2d-5422-a4d6-bf776828c4b7 base autoai-kb_rt22.2-py3.10
125b6d9a-5b1f-5e8d-972a-b251688ccf40 base runtime-22.1-py3.9 12b83a17-24d8-
5082-900f-0ab31fbfd3cb base scikit-learn_0.22-py3.6 154010fa-5b3b-4ac1-82af-
4d5ee5abbc85 base default_r3.6 1b70aec3-ab34-4b87-8aa0-a4a3c8296a36 base
pytorch-onnx_1.3-py3.6 1bc6029a-cc97-56da-b8e0-39c3880dbbe7 base kernel-
spark3.3-r3.6 1c9e5454-f216-59dd-a20e-474a5cdf5988 base pytorch-onnx_rt22.1-
py3.9-edt 1d362186-7ad5-5b59-8b6c-9d0880bde37f base tensorflow_2.1-py3.6
1eb25b84-d6ed-5dde-b6a5-3fbdf1665666 base spark-mllib_3.2 20047f72-0a98-58c7-
9ff5-a77b012eb8f5 base tensorflow_2.4-py3.8-horovod 217c16f6-178f-56bf-824a-
b19f20564c49 base runtime-22.1-py3.9-cuda 26215f05-08c3-5a41-a1b0-da66306ce658
base do_py3.8 295addb5-9ef9-547e-9bf4-92ae3563e720 base
...
runtime-22.2-py3.10-xc 5e8cddff-db4a-5a6a-b8aa-2d4af9864dab base autoai-kb_3.1-
py3.7 632d4b22-10aa-5180-88f0-f52dfb6444d7 base ----------------------------- -
----------------------------------- ---- Note: Only first 50 records were
displayed. To display more use 'limit' parameter.
```

```python
software_spec_uid = client.software_specifications.get_uid_by_name("runtime-
22.1-py3.9")
software_spec_uid
'12b83a17-24d8-5082-900f-0ab31fbfd3cb'
```

```python
import sklearn
sklearn.__version__
'1.0.2'
```

```python
MODEL_NAME = 'Model Building'
DEPLOYMENT_NAME = 'space'
DEMO_MODEL = rf
```

```python
model_props = {
    client.repository.ModelMetaNames.NAME: MODEL_NAME,
    client.repository.ModelMetaNames.TYPE: 'scikit-learn_1.0',
    client.repository.ModelMetaNames.SOFTWARE_SPEC_UID: software_spec_uid
}
```

```python
import json
```

```python
model_details = client.repository.store_model(
    model=DEMO_MODEL,
    meta_props=model_props,
    training_data=train_x,
    training_target=train_y
)
```

```
model_details
{'entity': {'hybrid_pipeline_software_specs': [], 'label_column':
'Loan_Status', 'schemas': {'input': [{'fields': [{'name': 'f0', 'type':
'float'}, {'name': 'f1', 'type': 'float'}, {'name': 'f2', 'type': 'float'},
{'name': 'f3', 'type': 'float'}, {'name': 'f4', 'type': 'float'}, {'name':
'f5', 'type': 'float'}, {'name': 'f6', 'type': 'float'}, {'name': 'f7', 'type':
'float'}, {'name': 'f8', 'type': 'float'}, {'name': 'f9', 'type': 'float'},
{'name': 'f10', 'type': 'float'}], 'id': '1', 'type': 'struct'}], 'output':
[]}, 'software_spec': {'id': '12b83a17-24d8-5082-900f-0ab31fbfd3cb', 'name':
'runtime-22.1-py3.9'}, 'type': 'scikit-learn_1.0'}, 'metadata': {'created_at':
'2022-11-19T07:46:23.076Z', 'id': '1f04b280-f2e1-4e63-9700-d230522fe297',
'modified_at': '2022-11-19T07:46:47.418Z', 'name': 'Model Building', 'owner':
'IBMid-6640043Y2R', 'resource_key': '7a380c52-42d2-42cc-b9d0-779ee01e8089',
'space_id': '9aaafe72-8656-4fb6-8c50-b0ce72cfd99b'}, 'system': {'warnings':
[]}}
```

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

```
deployment_props = {
    client.deployments.ConfigurationMetaNames.NAME:DEPLOYMENT_NAME,
    client.deployments.ConfigurationMetaNames.ONLINE: {}
}
```

```
deployment = client.deployments.create(
    artifact_uid=model_id,
    meta_props=deployment_props
```

```
#######################################################################################

Synchronous deployment creation for uid: '1f04b280-f2e1-4e63-9700-d230522fe297'
started

#######################################################################################

initializing Note: online_url is deprecated and will be removed in a future
release. Use serving_urls instead. ready
```

```
--------------------------------------------------------------------------------
Successfully finished deployment creation, deployment_
uid='f223875c-d0df-46db-8796-a66030399dd1'
------------------------------------------------------------------------
```

APPLICATION BUILDING - FLASK:

```
from flask import render_template,Flask,request
import numpy as np
import pickle
from sklearn.preprocessing import scale
app= Flask(__name__, template_folder='templates')

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


@app.route('/')
@app.route('/home')
def home():
    return render_template("home.html")

@app.route('/')
@app.route('/predict')
def predict():
    return render_template("predict.html")


@app.route('/')
@app.route('/submit')
def Submit():
    return render_template("submit.html")


@app.route('/submit',methods = ["GET", "POST"])
def index():
    if request.method=="POST":
        Gender=request.form['Gender']
        Married_Status=request.form['Married_Status']
        Dependents=request.form['Dependents']
        Education=request.form['Education']
        Self_Employed=request.form['Self_Employed']
        Credit_History=request.form['Credit_History']
        Property_Area=request.form['Property_Area']
        Applicant_Income=float(request.form['Applicant_Income'])
        Co_Applicant_Income=float(request.form['Co_Applicant_Income'])
        Loan_Amount=float(request.form['Loan_Amount'])
        Loan_Amount_Term=float(request.form['Loan_Amount_Term'])
```

```
        if Gender == 'Male':
            Gender = 1
        else:
            Gender = 0

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

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

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

        if int(Dependents) >= 3:
            Dependents = 3
        if Credit_History == '1':
            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

        names =
[Gender,Married_Status,int(Dependents),Education,Self_Employed,Applicant_Income
,Co_Applicant_Income,Loan_Amount,Loan_Amount_Term,Credit_History,Property_Area]
        print(names)

        features = [np.array(names)]

        prediction = rf.predict(features)

        print(prediction)

        if prediction == 1:
            return render_template('submit.html', result="Congratulations! You
are eligible for loan")
```

```
        else:
            return render_template('submit.html', result="Sorry! You are not
eligible for loan")

    else: return render_template("predict.html")



if __name__ == "__main__":
    app.run(debug=True, port=5000)
```

HTML CODE:

HOME PAGE:

```
<!DOCTYPE html>
<html lang="en">
<head>

    <title>Smart Lender - Applicant Credibility Prediction For Loan
Approval</title>
    <link rel="stylesheet" type="text/css" href="\static\style.css">

</head>
<body>
    <div class="main">
        <div class="navbar">
            <div class="icon">
                <h2 class="logo">Smart Lender</h2>
            </div>

        </div>
        <div class="content">
            <h1>Welcome To<br><span>Loan Approval</span> <br>Prediction
</h1><br>
            <p class="par">loan Approval is based on lot of this, rather than
going to a bank and getting rejected.<br> We made it simple that you can get
your loan approval prediction by our machine <br>learning model, for to predict
we need some of your information.</p>

            <button class="cn"><h2><a href="predict">PREDICT</a></h2></button>

            </div>



        </div>



    </div>
```

```
  <script src="https://unpkg.com/ionicons@5.4.0/dist/ionicons.js"></script>


</body>
</html>
```

PREDICT PAGE:

```
<!DOCTYPE html>
<html lang="en">
    <title>Predict</title>
    <link rel="stylesheet" type="text/css"  href="\static\dict.css">

<head>
</head>

<body>
 <form action="/submit" method="post">
  <h2 class="heading1">Enter Your Details For Loan Prediction</h2>



    <p2>Gender</p2><br><br>
    <select name="Gender">
      <option>Male</option>
      <option>Female</option>

    </select><br><br>


    <p2>Married Status</p2><br><br>
    <select name="Married_Status">

        <option>Yes</option>
        <option>No</option>

      </select><br><br>



    <p2>Dependents</p2><br><br>
    <input type="text" name="Dependents"><br><br>


    <p2>Education</p2><br><br>
    <select name="Education">

        <option>Graduate</option>
```

```html
        <option>Not Graduate</option>

      </select><br><br>

    <p2>Self employee</p2><br><br>
    <select name="Self_Employed">

        <option>Yes</option>
        <option>No</option>









   </select><br><br>

    <p2>Applicant Income</p2><br><br>
    <input type="text" name="Applicant_Income"><br><br>

    <p2>Co Applicant Income</p2><br><br>
    <input type="text" name="Co_Applicant_Income"><br><br>

    <p2>Loan Amount</p2><br><br>
    <input type="text" name="Loan_Amount"><br><br>

    <p2>Loan Amount Term</p2><br><br>
    <input type="text" name="Loan_Amount_Term"><br><br>

    <p2>Credit History</p2><br><br>
    <input type="text" name="Credit_History"><br><br>

    <p2>Property Area</p2><br><br>
    <select name="Property_Area">

        <option>Urban</option>
        <option>Rural</option>
        <option>semiurban</option>

      </select><br><br>


    <input type="submit">


</form>
```

```
</body>
</html
```

SUBMIT PAGE:

```html
<!DOCTYPE html>
<html lang="en">

<head>

    <title>Submit</title>
    <link rel="stylesheet" type="text/css" href="\static\sub.css">

</head>
<body>

<center style="padding: 210px;">
    <h1 class="heading2">Loan Approval Prediction</h1>
    <h1>{{result}}</h1>
</center>

<h1 style="color:#ff7200 "></h1>





</body>
</html>
```

CSS CODE:

STYLE.CSS

```css
* {
    margin: 0;
    padding: 0;
}

body {
    width: 100%;
    background-image: url(6.jpg);
    background-position: center;
    background-size: cover;
    height: 100vh;
}

.navbar{
    width: 1200px;
    height: 75px;
    margin: 5px;
}
```

```
.icon{
    width: 400px;
    float: left;
    height: 70px;
    margin: 5px;

}

.logo{
    color: #ff7200;
    font-size: 35px;
    font-family: Arial;
    padding-left: 20px;
    float: left;
    padding-top: 10px;
}

.menu{
    width: 400px;
    float: inline-start;
    height: 70px;

}

ul{
    float: left;
    display: flex;
    justify-content: center;
    align-items: center;
    font-size: 20px;
}

ul li{
    list-style: none;
    margin-left: 62px;
    margin-top: 27px;
    font-size: 20px;
    }

ul li a{
    text-decoration: none;
    color: #fff;
    font-family: Arial;
    font-weight: bold;
    transition: 0.4s ease-in-out;
}

ul li a:hover{
    color: #ff7200;
}
```

```
.content{
    width: 1200px;
    height: auto;
    margin: auto;
    color: #fff;
    position: relative;
}

.content .par{
    padding-left: 20px;
    padding-bottom: 25px;
    font-family: Arial;
    letter-spacing: 1.2px;
    line-height: 30px;
}

.content h1{
    font-family: 'Times New Roman', Times;
    font-size: 50px;
    padding-left: 20px;
    margin-top: 9%;
    letter-spacing: 2px;
}

.content .cn{
    width: 160px;
    height: 40px;
    background: #ff7200;
    border: none;
    margin-bottom: 10px;
    margin-left: 20px;
    font-size: 18px;
    border-radius: 10px;
    cursor: pointer;
    font-family: Arial;
}

.content .cn a{
    text-decoration: none;
    color: #000;
    transition: .3s ease;
}

.cn:hover{
    background-color: #fff;
}
```

```css
.content span{
    color: #ff7200;
    font-size: 55px;
}
.form{

    width: 270px;
    height: 300px;
    background: linear-gradient(to top, rgba(0,0,0,0.8)50%,
rgba(0,0,0,0.8)50%);
    margin: auto;
    border-radius: 10px;
    padding: 35px;
    margin-top: 140px;
}

.form h2{
    width: 220px;
    font-family: sans-serif;
    text-align: center;
    color: #fff;
    font: 22px;
    background-color: #ff7200;
    border-radius: 10px;
    margin: 2px;
    padding: 8px;

}

.form input{
    width: 240px;
    height: 35px;
    background: transparent;
    border-bottom: 1px solid #ff7200;
    border-top: none;
    border-right: none;
    border-left: none;
    color: #fff;
    font-size: 15px;
    letter-spacing: 1px;
    margin-top: 30px;
    font-family: sans-serif;
}
.form .btn{
    width: 250px;
    height: 40px;
    background: #ff7200;
    font-size: 20px;
    border-top: none;
    border-right: none;
```

```
    border-left: none;
    border-bottom: none;
    border-radius: 10px;
    color: #fff
}

.form .btn a{
    text-decoration: none;
    color: #000;
    font-weight: bold;
    font-family: 'Times New Roman', Times;
    border-top: none;
    border-right: none;
    border-left: none;




}

.form input:focus{
    outline: none;
}

::placeholder{
    color: #fff;
    font-family: Arial;


}

 .btn:hover{
  background: #fff;
}

.form .link{
    font-family: Arial, Helvetica, sans-serif;
    font-size: 17px;
    padding-top: 20px;
    text-align: center;
}

.form .link a{
    text-decoration: none;
    color: #ff7200;

}
.icon a{
    text-decoration: none;
    color: #fff;
```

```
}

.icon ion-icon{
    color: #ff7200;
    font-size: 30px;
    padding-left: 14px;
    padding-top: 5px;
    transition: 0.3s ease;
}

.icon ion-icon:hover{
    color: #fff;

}

.Liw{
    padding-top: 15px;
    padding-bottom: 10px;
    text-align: center;
}
```

DICT.CSS

```
input[type=text], select{
    width: 100%;
    height: 3rem;
    padding: 10px;
    margin: 5px;
    border: solid #ff7200;
    border-radius: 0px;
    box-sizing: border-box;
    background-color: grey;
    cursor: pointer;
    font-weight: bold;
    font-size: large;


}

form{
    text-align: left;
    margin-left: 10px;

}


body{
    background: black;
    font-weight: bold;
```

```
    font-size: 20px;
    font-family: sans-serif;
    cursor: pointer;
    margin: 30px;
    color: #ff7200;

}

button{
    height: 2.5rem ;
    width: 150px;
    margin: 5px;
    border: solid black;
    font-size: 20px;
    font-family: Arial;
    height: 3rem;
    font-weight: bold;
}

.cn:hover{
    background-color: #fff;
}

.heading1{
    font-family: serif;
    color: #ff7200;
}



.cn{
    background-color: #ff7200;
}

.cn a{
    text-decoration: none;
    color: #000;
    transition: .3s ease;
}
```

SUB.CSS

```css
body{

    background-color: orange;
}

.heading2{
    color: #fff;
}
```

GITHUB & PROJECT DEMO LINK

GITHUB REPO LINK: https://github.com/IBM-EPBL/IBM-Project-41987-1660647118

DEMO VIDEO DRIVE LINK:
https://drive.google.com/file/d/1b5linTcPOs7Vgj03Dl6vfEgBVA8uBw-E/view?usp=share_link