A

MINI PROJECT REPORT

on

"BANK LOAN PREDICTION"

Submitted in partial fulfillment for the completion of

B.E., III Semester

INFORMATION TECHNOLOGY

By

Y. KRISHNA GUPTHA (160120737155)

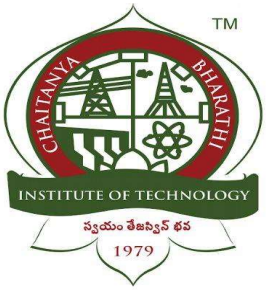
P. MANOJ KUMAR (160120737157)

Under the guidance of

Mr. G. Srikanth,

Asst. professor,

Dept. of IT, CBIT.



DEPARTMENT OF INFORMATION TECHNOLOGY

CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY (A)

(Affiliated to Osmania University; Accredited by NBA(AICTE) and NAAC(UGC), ISO Certified 9001:2015)

KOKAPET(V), GANDIPET(M), RR District HYDERABAD - 75

Website: www.cbit.ac.in

2021-2022

This is to certify that the project work entitled "**BANK LOAN PREDICTION**" submitted to CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY, in partial fulfillment of the requirements for the completion of Mini Project-I of III Semester B.E. in Information Technology, during the Academic Year 2021-2022, is a record of original work done by **Y.KRISHNA GUPTHA (160120737155) and P.MANOJ KUMAR (160120737157)** during the period of study in the Department of IT, CBIT, HYDERABAD, under our guidance.

Project Guide
Mr. G. Srikanth
Assistant Professor, Dept. of IT,
CBIT, Hyderabad.

Head of the Department
Dr. K. Radhika
Professor, Dept. of IT,
CBIT, Hyderabad.

ACKNOWLEDGMENTS

We would like to express our deep and sincere gratitude to our project guide, Mr. G. Srikanth Asst. Professor for allowing us to do the project and providing his invaluable guidance throughout this project. It was a great privilege and honor to work and study under his guidance.

We would like to thank our Head of Department, Dr. K. Radhika, for allowing us to improve our skills, for the constant support, and provided us with the sources that are required for the completion of this project.

Our sincere thanks to the Principal, Dr. P. Ravinder Reddy, as well as the management of the

institute, for giving us a good opportunity to improve our skills with a good learning atmosphere.

We are extremely grateful to our parents for their love, prayers, caring and sacrifices for educating and preparing for our future and supporting us in the completion of this project.

We would like to thank our friends for their constant help and guidance and their support in the completion of this project.

ABSTRACT

The bank plays a vital role in the market economy. The success or failure of an organization largely depends on the industry's ability to evaluate credit risk. Before giving the credit loan to borrowers, the bank decides whether the borrower is bad (defaulter) or good (non-defaulter). The prediction of borrower status i.e. in the future borrower will be defaulter or non-defaulter is a challenging task for any organization or bank. In our project, we used the bank loan prediction dataset provided by kaggle.com. A bank's profit or a loss depends to a large extent on loans i.e., whether the customers are paying back the loan or defaulting. By predicting the loan defaulters, the bank can reduce its Non- Performing Assets. This makes the study of this phenomenon very important.

This dataset consists of 981 samples which are further divided into two parts training set and testing set. The training set has 614 samples and the test set has 367 samples. In the support of this project, we have used libraries viz. scikit to import a machine learning algorithm called Logistic Regression and pandas, matplotlib, seaborn, PIL (python imaging library), joblib, streamlet libraries for the data analytics and data representation respectively. The source code is written under a Python environment. After training the model, To prove real-time efficiency, we have evaluated our prediction model and it was measured with an accuracy of 83%. In this process confusion matrix between tested and predicted values is also analyzed. Further, the deployment was made using the Heroku cloud platform.

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

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

1.1 INTRODUCTION

Nowadays due to the pandemic situations, it is a very risky and time taking process to reach the bank and check the loan eligibility status of any individual so the project "Bank Loan Prediction" is a website aimed at checking the user based on property, employment status, marital status, credit score, monthly income, loan amount, loan duration, and few other factors. It helps users to check their loan eligibility status at a run time.

1.2 MOTIVATION

The motivation for doing this project was that we were primarily interested in undertaking a real-time application of machine learning as an area of research. The secondary motive of this project is to skip documentation parts provided by the bank manually and fill the form online, which positively impacts the customers' and bank employees' by saving time.

1.3 PROBLEM STATEMENT

Banks are facing a significant problem in the approval of the loan. With the advancement in technology, there are so many enhancements in the banking sector also. As the data is increasing daily due to digitization in the banking sector, people want to apply for loans through the internet. Daily there are so many applications that are challenging to manage by the bank employees, and also the chances of some mistakes are high. One mistake can make a massive loss to a bank. Machine Learning a subset of Artificial intelligence (AI), as a typical method for information investigation, has gotten more consideration increasingly.

1.4 OBJECTIVE

The main objective of this project is to design and develop a cloud platform for predicting the individual's loan eligibility status using a machine learning algorithm that can be used to overcome the problem statements above so that a person can prepare before applying for a loan for him/herself. The users who can't visit the bank can check their status of loan eligibility at their fingertips especially in the present pandemic period.

1.5 APPLICATIONS

1. This cloud platform provides the user an easy opportunity to check their loan eligibility status.
2. Instead of approaching the bank manually this platform makes things easier in this pandemic situation.
3. Users can enter their property and income details in a safe and online mode.
4. Users will receive an instant reply i.e., a run-time prediction for users to check the loan eligibility.

1.6 ORGANISATION OF THE REPORT

The mini-project is divided into 6 parts.

1. First part contains an introduction to the project.
2. Second part of the project contains the literature survey which consists of software and hardware requirements and the existing system of the project.
3. Third part contains the methodology of the project and block diagram of the proposed system.
4. Fourth part contains the implementation of the project this consists of the code used for execution.
5. Fifth part contains the testing and results of the project. It consists of screenshots of the execution of the project.
6. Sixth part contains the conclusion of the project and the future of the project.

2. LITERATURE SURVEY

SOFTWARE REQUIREMENTS

Languages used:

Front end development:

1. StreamLit - For creating and designing web apps.

Backend development:

1. Google Drive - To store the data.
2. Heroku - To host a website and provide a permanent domain.
3. Google Collaboratory – For implementing the machine learning model.
4. Sklearn and other python libraries– To import machine learning algorithms.

HARDWARE REQUIREMENTS (minimum)

OS	: Windows 7 SP1
RAM	: 2GB
STORAGE	: 250 MB
INPUT DEVICE	: Keyboard or touch screen.

OTHER REQUIREMENTS: Graphical User Interface (GUI) or integrated video card.

EXISTING SYSTEM

In today's world, manual documentation is required in the bank for checking the loan eligibility status. An intermediate person is required to check the details of every individual who wants to apply for a loan. According to the details the mediator reports within a period which may take a few days which might be a waste of a lot of time and money.

3. METHODOLOGY

The proposed model focuses on predicting the eligibility of customers for loan repayment by analyzing their input factors. On the output from the regression model, the decision on whether to approve or reject the loan eligibility can be made. Using different data analytics tools loan prediction and their severity can be forecasted. In this process, it is required to train the data using different algorithms and then compare user data with trained data to predict the nature of the loan. After training the data to a few algorithms we found that the logistic regression algorithm started resulting in better accuracy. The training data set is now supplied to this model; based on this data set, the model is trained. Every new applicant's details filled at the time of application form acts as a test data set. After the operation of testing, the model predicts whether the new applicant is a fit case for approval of the loan or not based upon the inference it concludes based on the training data sets to extract important information and predict if a customer would be able to repay his loan or not.

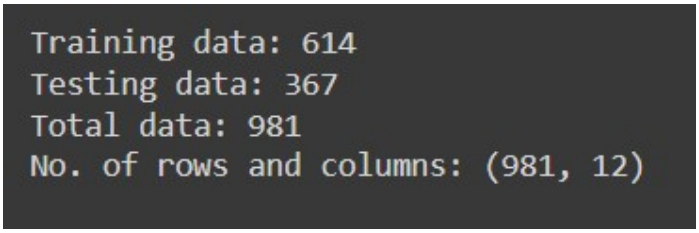


Fig (3.1) COMPOSITION OF DATASET

The major steps that we employed in developing the machine learning algorithm are

- **STEP-1: IMPORTING THE DATASET**
- **STEP-2: PRE-PROCESSING OF DATA**
- **STEP-3: LABEL ENCODING**
- **STEP-4: FILLING THE MISSING VALUES**
- **STEP-5: DROPPING THE UNWANTED VALUES**
- **STEP-6: SPLITTING THE DATA**
- **STEP-7: TRAINING AND TESTING THE MODEL**
- **STEP-8: DEMONSTRATION OF CONFUSION MATRIX**
- **STEP-9: SAVING THE MODEL**
- **STEP-10: EVALUATION OF THE MODEL**

	precision	recall	f1-score	support
0	0.91	0.47	0.62	43
1	0.83	0.98	0.90	111
accuracy			0.84	154
macro avg	0.87	0.72	0.76	154
weighted avg	0.85	0.84	0.82	154

Fig (3.2) EVALUATION METRICS

1. Accuracy:

- It measures how often the classifier is correct for both true positives and true negative cases. Mathematically, it is defined as:

- Accuracy = (True Positive + True Negative)/Total Predictions

2. Recall:

- It measures how many times did the classifier get the true positives correct. Mathematically, it is defined as:

- Recall = True Positive/ (True Positive + False Negative)

3. F1-Score:

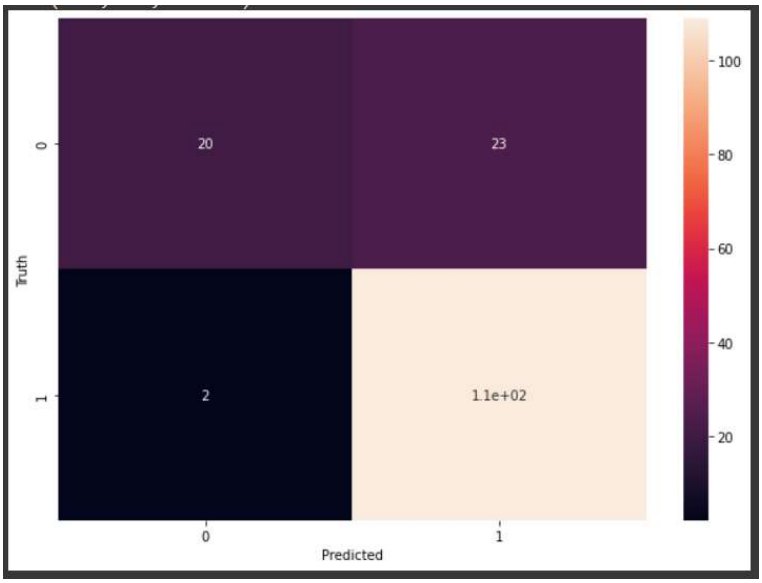
- It conveys the balance between precision and recall.

- F1-Score = 2 * ((precision * recall) / (precision + recall))

4 Precision:

- Precision measures of the total predicted to be positive how many were positive.

- Precision = (True Positive) / (True Positive + False Positive)




```
[[ 20  23]
 [  2 109]]
```

Fig (3.3) LAYOUT OF CONFUSION MATRIX

If T and N denote the number of clients that will default the credit payment and clients that will not default in the payment of their credit respectively, then the total number of the dataset is expressed as T + N. Furthermore, TP (True Positive) and TN (True Negative) represent the total positive and negative cases/instances that are rightly classified, respectively. The FP and FN also denote the number of predicted/classified instances that are incorrectly predicted yes when it is no and the number of instances that are predicted no when it is actually yes, respectively. These constitute the entries to the confusion matrix shown in the above layout.

From the above layout,

True Positive = 109

True Negative = 20

False Positive = 23

False Negative= 2

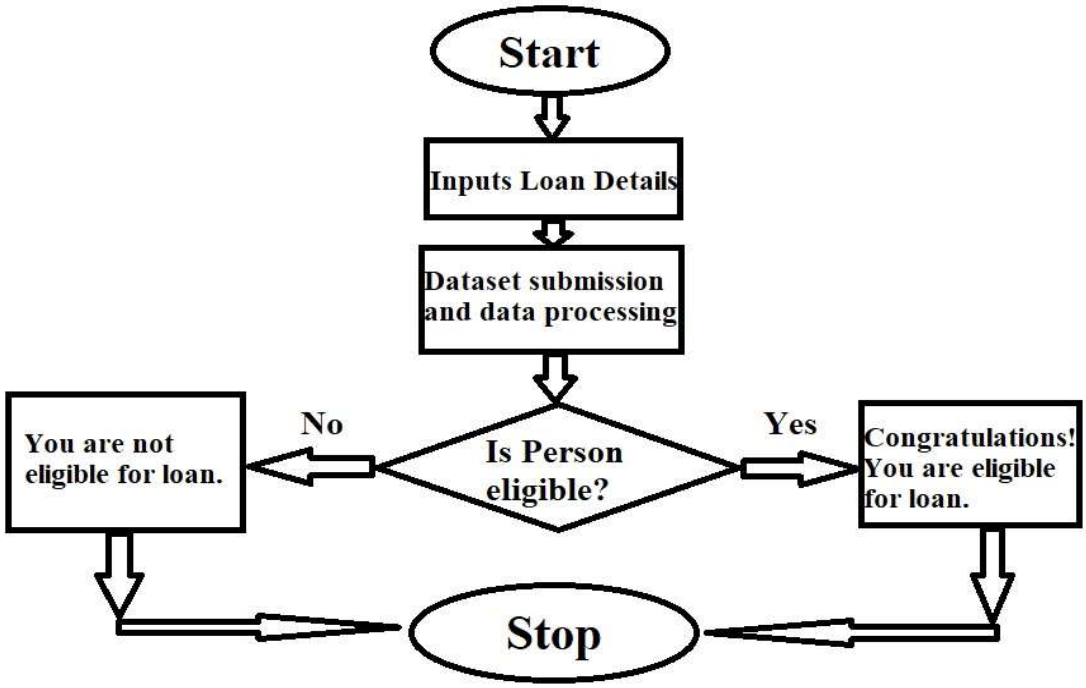


Fig (3.4) BLOCK DIAGRAM

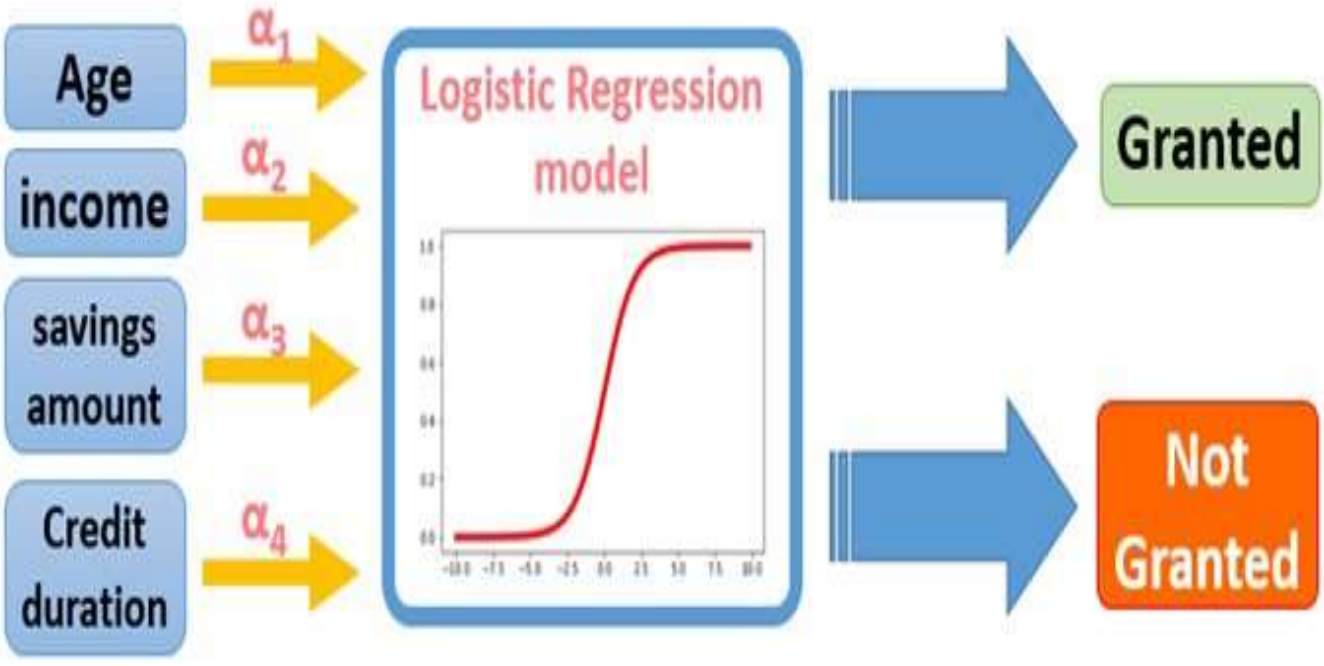


Fig (3.5) OVERALL CLASS DIAGRAM

4. IMPLEMENTATION

We have implemented the website in two sections; the front-end section and the back-end section. The front-end section contains the streamlit page in which the user can input their respective details and when submit button is clicked result is displayed. In the back-end section, the details which are entered by the user are given as an input to the model which predicts the probability of eligibility of loan for that particular user at an instant.

We store the implementation of the front-end part and the back-end part in the Github repository which is a code hosting platform along with a few attributes that are required to create a bridge network with the web hosting platforms. We connect Github and Heroku to host the website with a permanent domain.

Fig (4.1) FRONT-END IMPLEMENTATION

98 lines (81 sloc) | 3.39 KB

Raw Blame

```python
import streamlit as st
from PIL import Image
import pickle

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

def run():
    img1 = Image.open('bank_logo.JPG')
    img1 = img1.resize((156,145))
    st.image(img1,use_column_width=False)
    st.title("Bank Loan Prediction")

    ## Account No
    account_no = st.text_input('Account number')

    ## Full Name
    fn = st.text_input('Full Name')

    ## For gender
    gen_display = ('Female','Male')
    gen_options = list(range(len(gen_display)))
    gen = st.selectbox("Gender",gen_options, format_func=lambda x: gen_display[x])

    ## For Marital Status
    mar_display = ('Unmarried','Married')
    mar_options = list(range(len(mar_display)))
    mar = st.selectbox("Marital Status", mar_options, format_func=lambda x: mar_display[x])

```

```
    ## No of dependets
    dep_display = ('No','One','Two','More than Two')
    dep_options = list(range(len(dep_display)))
    dep = st.selectbox("Dependents",  dep_options, format_func=lambda x: dep_display[x])

    ## For edu
    edu_display = ('Not Graduate','Graduate')
    edu_options = list(range(len(edu_display)))
    edu = st.selectbox("Education",edu_options, format_func=lambda x: edu_display[x])

    ## For emp status
    emp_display = ('Job','Business')
    emp_options = list(range(len(emp_display)))
    emp = st.selectbox("Employment Status",emp_options, format_func=lambda x: emp_display[x])

    ## For Property status
    prop_display = ('Rural','Semi-Urban','Urban')
    prop_options = list(range(len(prop_display)))
    prop = st.selectbox("Property Area",prop_options, format_func=lambda x: prop_display[x])

    ## For Credit Score
    cred_display = ('Between 300 to 500','Above 500')
    cred_options = list(range(len(cred_display)))
    cred = st.selectbox("Credit Score",cred_options, format_func=lambda x: cred_display[x])

    ## Applicant Monthly Income
    mon_income = st.number_input("Applicant's Monthly Income($)",value=0)

    ## Co-Applicant Monthly Income
    co_mon_income = st.number_input("Co-Applicant's Monthly Income($)",value=0)

    ## Loan AMount
    loan_amt = st.number_input("Loan Amount",value=0)


    ## loan duration
    dur_display = ['2 Month','6 Month','8 Month','1 Year','16 Month']
    dur_options = range(len(dur_display))
    dur = st.selectbox("Loan Duration",dur_options, format_func=lambda x: dur_display[x])

    if st.button("Submit"):
        duration = 0
        if dur == 0:
            duration = 60
        if dur == 1:
            duration = 180
        if dur == 2:
            duration = 240
        if dur == 3:
            duration = 360
        if dur == 4:
            duration = 480
        features = [[gen, mar, dep, edu, emp, mon_income, co_mon_income, loan_amt, duration, cred, prop]]
        print(features)
        prediction = model.predict(features)
        lc = [str(i) for i in prediction]
        ans = int("".join(lc))
        if ans == 0:
            st.error(
                "Hello: " + fn +" || "
                "Account number: "+account_no +' || '
                'According to our Calculations, you will not get the loan from Bank'
            )
        else:
            st.success(
                "Hello: " + fn +" || "
                "Account number: "+account_no +' || '
                'Congratulations!! you will get the loan from Bank'
            )

run()
```

Fig (4.2) BACK-END IMPLEMENTATION

STEP-1: IMPORTING THE DATASET

```
[1] import pandas as pd
    import numpy as np
    import os
    os.chdir('/content/drive/MyDrive/MP-1/Loan Eligible Dataset')
```

```
[2] train = pd.read_csv('./loan-train.csv')
    train.Loan_Status = train.Loan_Status.map({'Y':1,'N':0})
```

```
[3] train.head()
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0	Urban	1
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	0
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	1
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	1
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	1

STEP-2: PRE-PROCESSING OF DATA

```
[4] train.isnull().sum()
```

```
Loan_ID               0
Gender               13
Married               3
Dependents           15
Education             0
Self_Employed        32
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           22
Loan_Amount_Term     14
Credit_History       50
Property_Area         0
Loan_Status           0
dtype: int64
```

```
[5] Loan_status = train.Loan_Status
    train.drop('Loan_Status',axis = 1,inplace = True)
    test = pd.read_csv('./loan-test.csv')
    Loan_ID = test.Loan_ID
    data = train.append(test)
    data.head()
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0	Urban
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban

```
[6] print("Training data:",len(train.index))
    print("Testing data:",len(test.index))
    print("Total data:",len(data.index))
    print("No. of rows and columns:",data.shape)
```

```
Training data: 614
Testing data: 367
Total data: 981
No. of rows and columns: (981, 12)
```

STEP-3: LABEL ENCODING

```
[7] #Label encoding for gender
    data.Gender = data.Gender.map({'Male':1,'Female':0})
    data.Gender.value_counts()
```

```
1.0    775
0.0    182
Name: Gender, dtype: int64
```

```
[8] #Label encoding for marital status
    data.Married = data.Married.map({'Yes':1,'No':0})
    data.Married.value_counts()
```

```
1.0    631
0.0    347
Name: Married, dtype: int64
```

```
[9] #Label encoding for Dependents
    data.Dependents = data.Dependents.map({'0':0,'1':1,'2':2,'3+':3})
    data.Dependents.value_counts()
```

```
0.0    545
2.0    160
1.0    160
3.0     91
Name: Dependents, dtype: int64
```

```
[10] #Label encoding for Education Status
     data.Education = data.Education.map({'Graduate':1,'Not Graduate':0})
     data.Education.value_counts()
```

```
1    763
0    218
Name: Education, dtype: int64
```

```
[11] #Label encoding for Employment Status
     data.Self_Employed = data.Self_Employed.map({'Yes':1,'No':0})
     data.Self_Employed.value_counts()
```

```
0.0    807
1.0    119
Name: Self_Employed, dtype: int64
```

```
[12] #Label encoding for Property Area
     data.Property_Area = data.Property_Area.map({'Rural':0,'Semiurban':1,'Urban':2})
     data.Property_Area.value_counts()
```

```
1    349
2    342
0    290
Name: Property_Area, dtype: int64
```

```
[13] data.head()
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area
0	LP001002	1.0	0.0	0.0	1	0.0	5849	0.0	NaN	360.0	1.0	2
1	LP001003	1.0	1.0	1.0	1	0.0	4583	1508.0	128.0	360.0	1.0	0
2	LP001005	1.0	1.0	0.0	1	1.0	3000	0.0	66.0	360.0	1.0	2
3	LP001006	1.0	1.0	0.0	0	0.0	2583	2358.0	120.0	360.0	1.0	2
4	LP001008	1.0	0.0	0.0	1	0.0	6000	0.0	141.0	360.0	1.0	2

STEP-4: FILLING THE MISSING VALUES

```
[14] print("Before filling:\n",data.isnull().sum())
     data.Dependents.fillna(data.Dependents.median(),inplace = True)
     data.Credit_History.fillna(np.random.randint(0,2),inplace = True)
     data.Married.fillna(np.random.randint(0,2),inplace = True)
     data.LoanAmount.fillna(data.LoanAmount.median(),inplace = True)
     data.Loan_Amount_Term.fillna(data.Loan_Amount_Term.mean(),inplace = True)
     data.Self_Employed.fillna(np.random.randint(0,2),inplace = True)
     data.Gender.fillna(np.random.randint(0,2),inplace = True)
     print("After filling:\n",data.isnull().sum())
```

```
Before filling:
 Loan_ID              0
Gender               24
Married               3
Dependents           25
Education             0
Self_Employed        55
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           27
Loan_Amount_Term     20
Credit_History       79
Property_Area         0
dtype: int64
After filling:
 Loan_ID              0
Gender                0
Married               0
Dependents            0
Education             0
Self_Employed         0
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount            0
Loan_Amount_Term      0
Credit_History        0
Property_Area         0
dtype: int64
```

STEP-5: DROPPING THE UNWANTED VALUES

```
data.drop('Loan_ID', inplace = True, axis = 1)
```

```
data.head()
```

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area
0	1.0	0.0	0.0	1	0.0	5849	0.0	126.0	360.0	1.0	2
1	1.0	1.0	1.0	1	0.0	4583	1508.0	128.0	360.0	1.0	0
2	1.0	1.0	0.0	1	1.0	3000	0.0	66.0	360.0	1.0	2
3	1.0	1.0	0.0	0	0.0	2583	2358.0	120.0	360.0	1.0	2
4	1.0	0.0	0.0	1	0.0	6000	0.0	141.0	360.0	1.0	2

STEP-6: SPLITTING THE DATA

```
x = data.iloc[:614,]
y = Loan_status
```

```
from sklearn.model_selection import train_test_split
train_x,test_x,train_y,test_y = train_test_split(x,y,random_state = 0,test_size=0.25)
```

```
print(train_x.shape)
print(test_x.shape)
print(train_y.shape)
print(test_y.shape)
```

```
(460, 11)
(154, 11)
(460,)
(154,)
```

```
[20] test_x.head()
```

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area
454	1.0	0.0	0.0	1	1.0	7085	0.0	84.0	360.0	1.0	1
52	0.0	0.0	0.0	1	0.0	4230	0.0	112.0	360.0	1.0	1
536	1.0	1.0	0.0	1	0.0	6133	3906.0	324.0	360.0	1.0	2
469	1.0	1.0	0.0	1	0.0	4333	2451.0	110.0	360.0	1.0	2
55	1.0	1.0	2.0	1	0.0	2708	1167.0	97.0	360.0	1.0	1

STEP-7: TRAINING AND TESTING THE MODEL

```
[21] from sklearn.linear_model import LogisticRegression
     model = LogisticRegression()
     model = model.fit(train_x,train_y)
```

```
[22] model.score(train_x,train_y)
```

```
0.8043478260869565
```

```
[23] model.score(test_x,test_y)
```

```
0.8376623376623377
```

```
[24] pred_y = model.predict(test_x)
     print('Predicted data:',pred_y)
     print('Testing data:',test_y)
```

```
Predicted data: [1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 0 0 1 1 1 1 1 0 1
 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1
 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 0 0 1 1 1 1 1 0 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1
 0 0 1 1 1 0]
Testing data: 454    1
52     0
536    1
469    0
55     1
      ..
399    0
89     1
271    1
563    1
162    0
Name: Loan_Status, Length: 154, dtype: int64
```

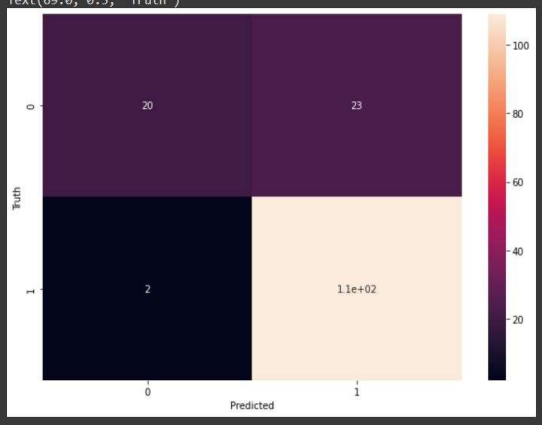
STEP-8: DEMONSTRATION OF CONFUSION MATRIX

```
[25] from sklearn.metrics import confusion_matrix,classification_report
     matrix = confusion_matrix(test_y,pred_y)
     print(matrix)
```

```
[[ 20  23]
 [  2 109]]
```

```
[26] %matplotlib inline
     import matplotlib.pyplot as plt
     import seaborn as sn
     plt.figure(figsize = (10,7))
     sn.heatmap(matrix,annot = True)
     plt.xlabel('Predicted')
     plt.ylabel('Truth')
```

```
Text(69.0, 0.5, 'Truth')
```



STEP-9: SAVING THE MODEL

```
[27] import pickle
     file = '/content/drive/MyDrive/MP-1/Model/ML_Model.pkl'
     with open(file, 'wb') as f:
         pickle.dump(model, f)
```

STEP-10: EVALUATION OF THE MODEL

```
report = classification_report(test_y,pred_y)
print(report)
```

```
              precision    recall  f1-score   support

           0       0.91      0.47      0.62        43
           1       0.83      0.98      0.90       111

    accuracy                           0.84       154
   macro avg       0.87      0.72      0.76       154
weighted avg       0.85      0.84      0.82       154
```

```
[29] with open(file, 'rb') as f:
         model = pickle.load(f)
```

```
[30] #A sample from the dataset is taken to evaluate the model
     decision = model.predict([[0.0, 0.0,  0.0,  1,  0.0,  1811, 1666.0, 54.0, 360.0,  1.0,  2]])
     print(decision)  #true or false
```

```
[1]
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:451: UserWarning: X does not have valid feature names, but LogisticRegression was fitted with feature names
  "X does not have valid feature names, but"
```

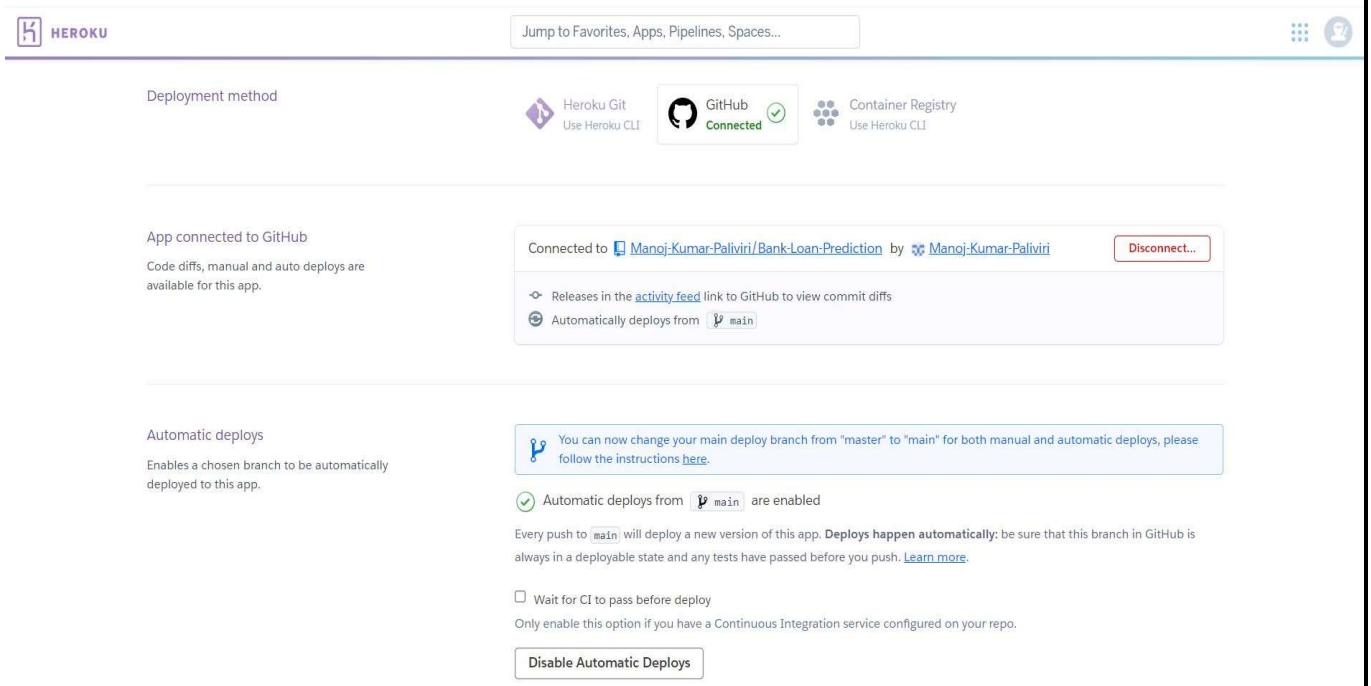


Fig (4.3) CONNECTING THE FRONT-END(HEROKU) AND BACK-END(GITHUB)

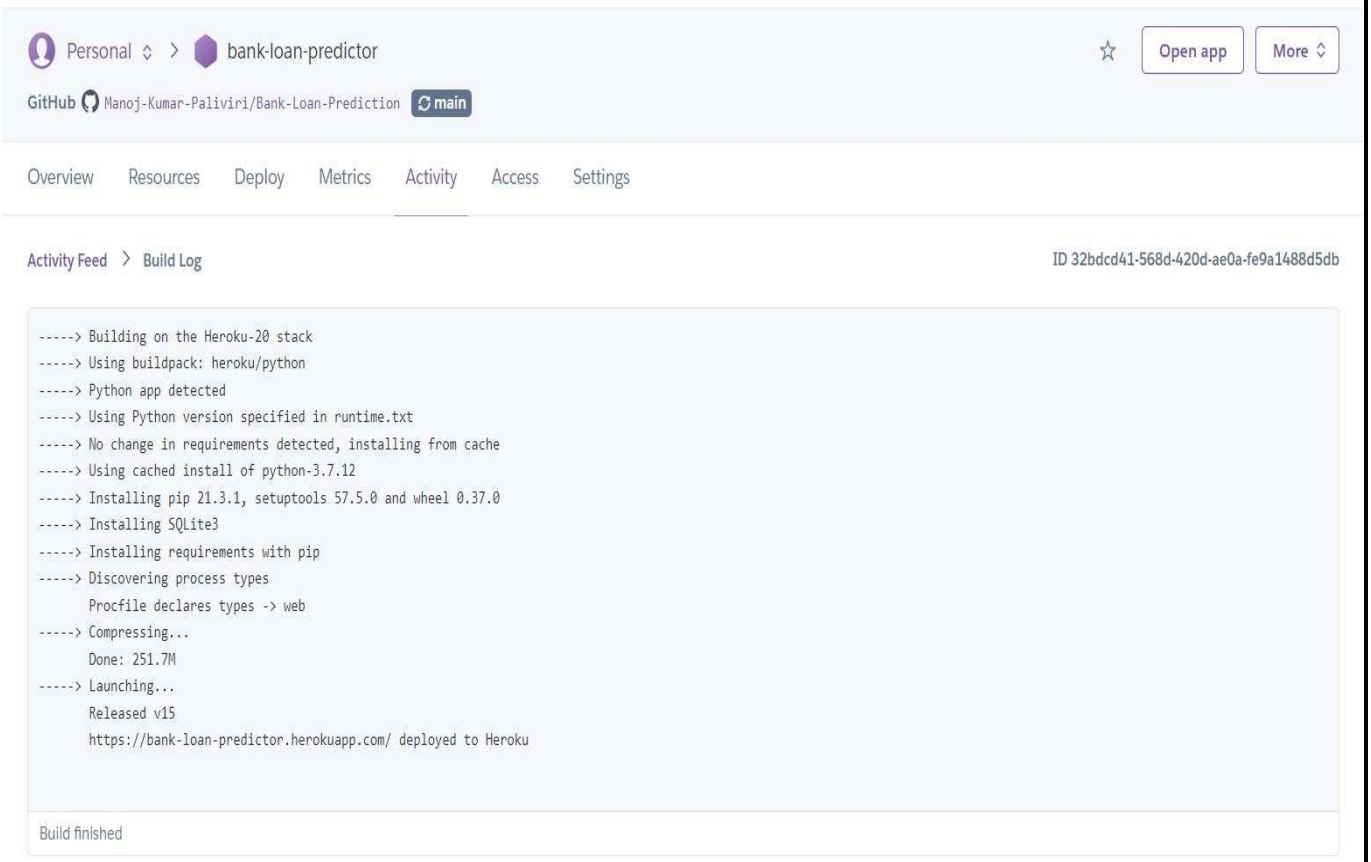
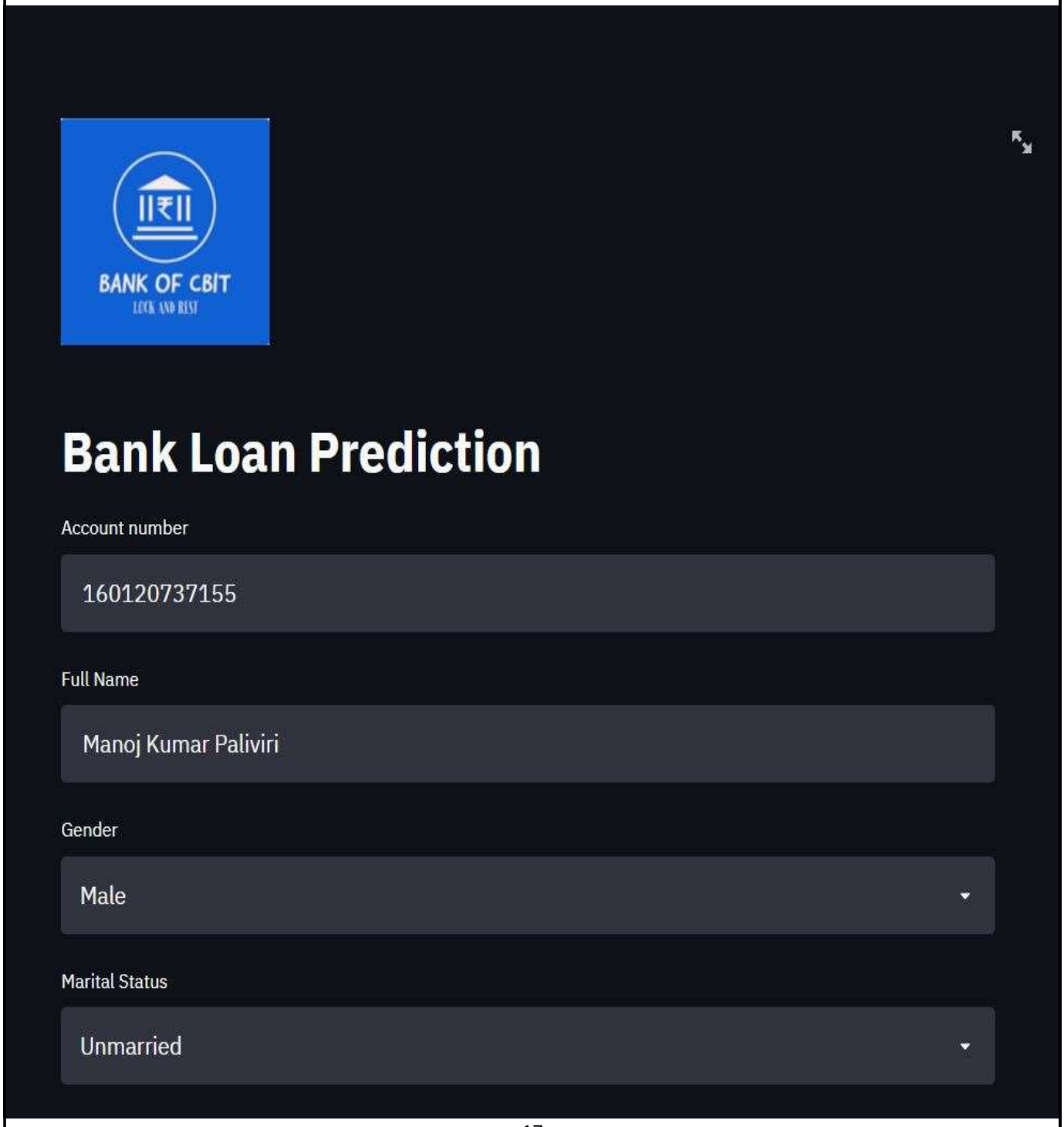


Fig (4.4) DEPLOYING AND HOSTING THE WEB APPLICATION

5.TESTING & RESULTS

Fig (5.1) POSITIVE TEST CASE

The person with a high salary, living in an urban area, who is a graduate and has a high credit score is most likely eligible for the loan.



Dependents

No

Education

Graduate

Employment Status

Business

Property Area

Urban

Credit Score

Above 500

Applicant's Monthly Income($)

3000 − +

Co-Applicant's Monthly Income($)

1500 − +

Loan Amount

500 − +

Loan Duration

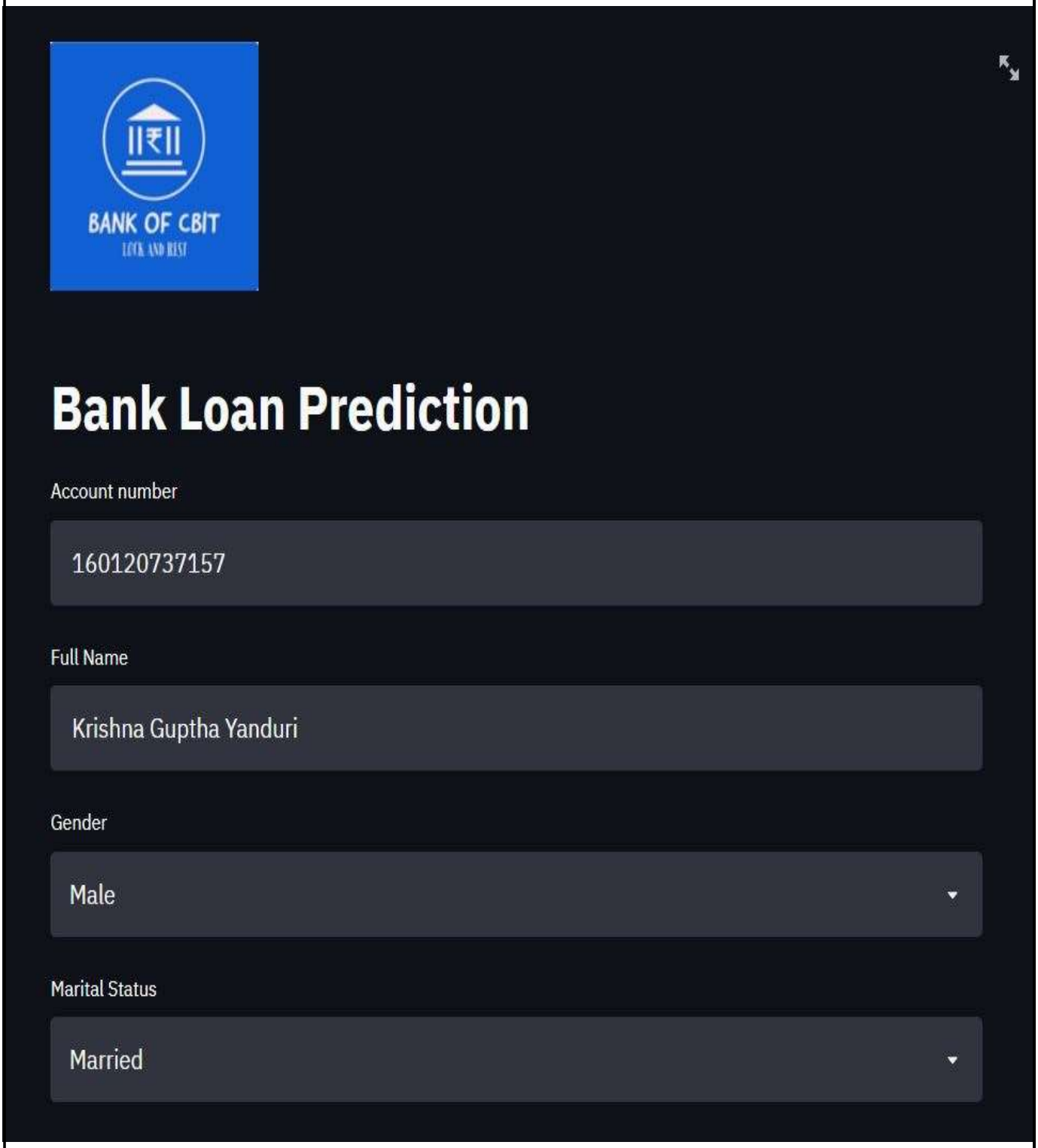
2 Month

Submit

Hello: Manoj Kumar Paliviri || Account number: 160120737155 || Congratulations!! you will get the loan from Bank

Fig (5.2) NEGATIVE TEST CASE

The person with a low salary, living in a rural/semi-urban area, who is a graduate and has a low credit score is most likely not eligible for the loan.



Dependents

One

Education

Graduate

Employment Status

Job

Property Area

Semi-Urban

Credit Score

Between 300 to 500

Applicant's Monthly Income($)

2000 − +

Co-Applicant's Monthly Income($)

1000 − +

Loan Amount

499 − +

Loan Duration

1 Year

Submit

Hello: Krishna Guptha Yanduri || Account number: 160120737157 || According to our Calculations, you will not get the loan from Bank

6. CONCLUSION & FUTURE SCOPE

CONCLUSION:

The interaction of expectation begins from cleaning and handling of information, attribution of missing qualities, and afterward model structure to an assessment of model and testing on test information. On the Data set, the best-case precision acquired from logistic regression is 0.8311. The accompanying ends are reached after examination that those candidates whose credit score rating was most noticeably awful will neglect to get approval because of a higher likelihood of not repaying the credit sum. More often than not, those candidates who have top-level salaries are bound to take care of their credits. Some other trademarks like gender and marital status appear not to be much affected over by the predictor.

FUTURE SCOPE:

Hyper Parameter Tuning can be implemented on the algorithm to retrieve optimal accuracy and precision. Few advanced algorithms may produce better accuracy in the upcoming future. MySQL can be implemented on this project to access real-time data from the database. In real-life scenarios, the confidential data from the bank can be retrieved and used for the prediction of loan eligibility of customers.

7. BIBLIOGRAPHY

[1] https://www.kaggle.com/altruistdelhite04/loan-prediction-problem-dataset

[2] https://medium.com/devcareers/loan-prediction-using-selected-machine-learning-algorithms-1bdc00717631

[3] https://towardsdatascience.com/predict-loan-eligibility-using-machine-learning-models-7a14ef904057

[4] https://www.ijert.org/predict-loan-approval-in-banking-system-machine-learning-approach-for-cooperative-banks-loan-approval

[5] https://github.com/Spidy20/Streamlit_Bank_Loan_Prediction