[Year]

PRDECTING PERSONAL LOAN APPROVAL USING MACHINE LEARNING

DONE BY
T. MUTHULAKSHMI
S. LAKSHMI
M.DHIVYADHARSHINI
P. RADHIKA

veerakumar
[Type the company name]
[Pick the date]

Predicting Personal Loan Approaval Using Machine Learning

1.Introduction

1.1 Overview

In this project, we are going to solve the Loan Approval Prediction Hackathon hosted by Analytics Vidhya. This is a classification problem in which we need to classify whether the loan will be approved or not. classification refers to a predictive modeling problem where a class label is predicted for a given example of input data. A few examples of classification problems are Spam Email detection, Cancer detection, Sentiment Analysis, etc.

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

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

Predicting personal loan approval using machine learning analyses a borrower's financial data and credit history to determine the likelihood of loan approval. This can help financial

institutions to make more informed decisions about which loan applications to approve and which to deny.

Impact

As the data is increasing daily due to in the banking sector, people want to apply for loans through the internet. Machine Learning (ML), as a typical method for, has go more consideration increasingly. Individuals of various businesses are ML calculations to take care of the issues dependent on their industry information. Banks are facing a significant problem in the approval of the loan. Daily there are so many applications that are challenging to manage by the bank employees, and also the chances of some mistakes are high most banks earn profit from the loan, but it is risky to choose deserving customers from the number of applications. There are various algorithms that have been used with varying levels of success. regression, decision tree, random forest, and neural networks have all been used and have been able to accurately predict loan defaults. Commonly used features in these studies include credit score, income, and employment history, some ones also other features like age, occupation, and education level.

Data collection

In this project we have used .CSV data. This data is downloaded from kaggle.com. Please refer to the link given below to download the dataset.

Link: https://www.kaggle.com/datasets/altruistdelhite04/loan-prediction-problem-dataset

As the dataset is downloaded. Let us read and understand the data properly with the help of some techniques and some techniques. & preparation. Read the Dataset Our dataset format might be in .CSV, excel files, .txt etc. We can read the dataset with the help of pandas. In pandas we have a function called read.CSV() to read the dataset. As a parameter we have to give the directory of the CSV file.

Read the Dataset

Our dataset format might be in .CSV excel files, .txt, etc. We can read the dataset with the help of pandas. In pandas we have a function called read.CSV() to read the dataset. As a parameter we have to give the directory of the CSV file

```
#importing the dataset which is in csv file
data = pd.read_csv('loan_prediction.csv')
data
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	L
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	3
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	3
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	3
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	3
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	3
...	...	...	...	...	...	...	...	...	...	...
609	LP002978	Female	No	0	Graduate	No	2900	0.0	71.0	3
610	LP002979	Male	Yes	3+	Graduate	No	4106	0.0	40.0	1
611	LP002983	Male	Yes	1	Graduate	No	8072	240.0	253.0	3
612	LP002984	Male	Yes	2	Graduate	No	7583	0.0	187.0	3
613	LP002990	Female	No	0	Graduate	Yes	4583	0.0	133.0	3

614 rows × 13 columns

Data Preparation

As we have understood how the data is, let's pre-process the collected data. The download data set is not suitable for training the machine learning model as it might have so much randomness so we need to clean the dataset properly in order to fetch good results. This activity includes the following steps.

● Handling missing values

● Handling categorical data

● Handling Imbalance Data Note: These are the general steps of pre-processing the data before using it for machine learning. Depending on the CONDITION of your dataset, you may or may not have to go through all these steps

```
#finding the sum of null values in each column
data.isnull().sum()
```

```
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
```

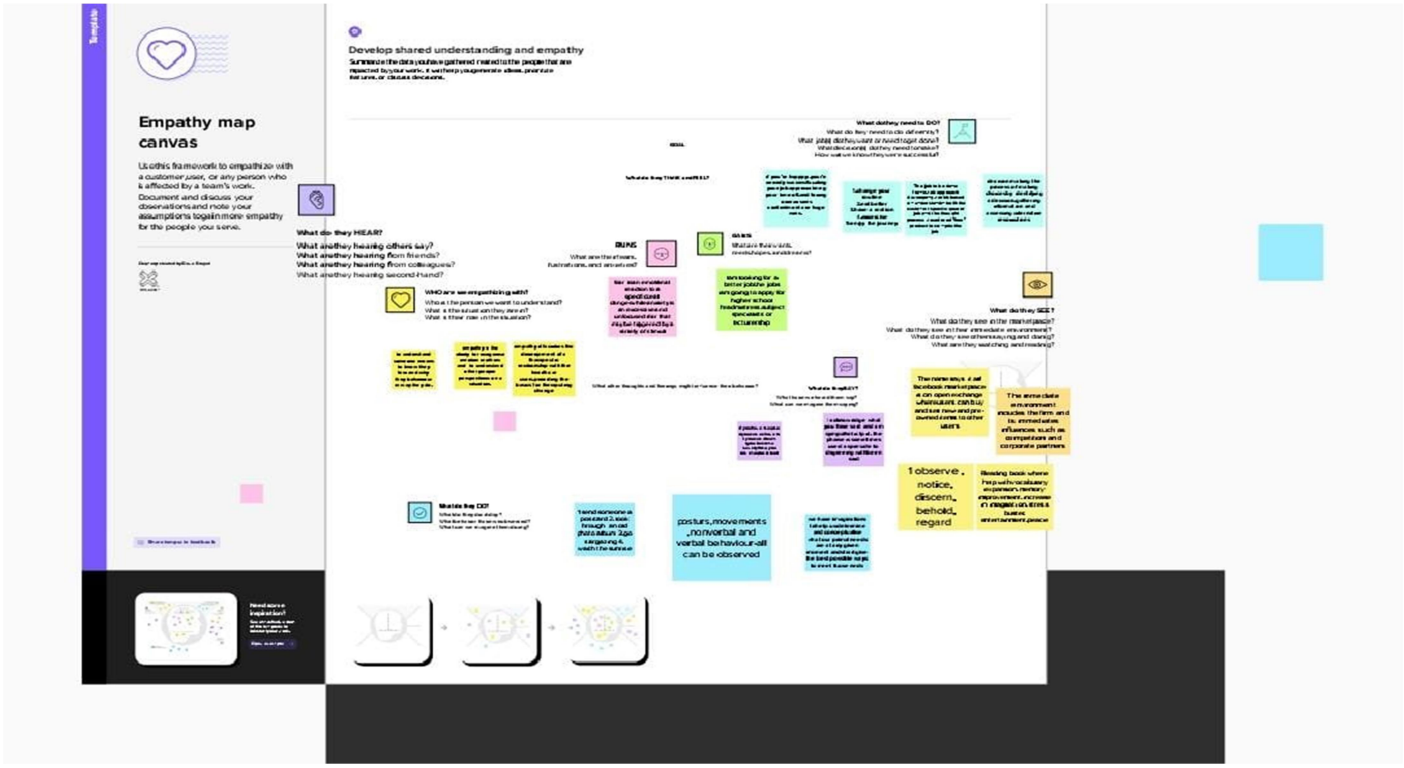
1.2 Purpose

Dream Housing Finance company deals in all kinds of home loans. They have a presence across all urban, semi-urban and rural areas. The customer first applies for a home loan and after that, the company validates the customer eligibility for the loan.

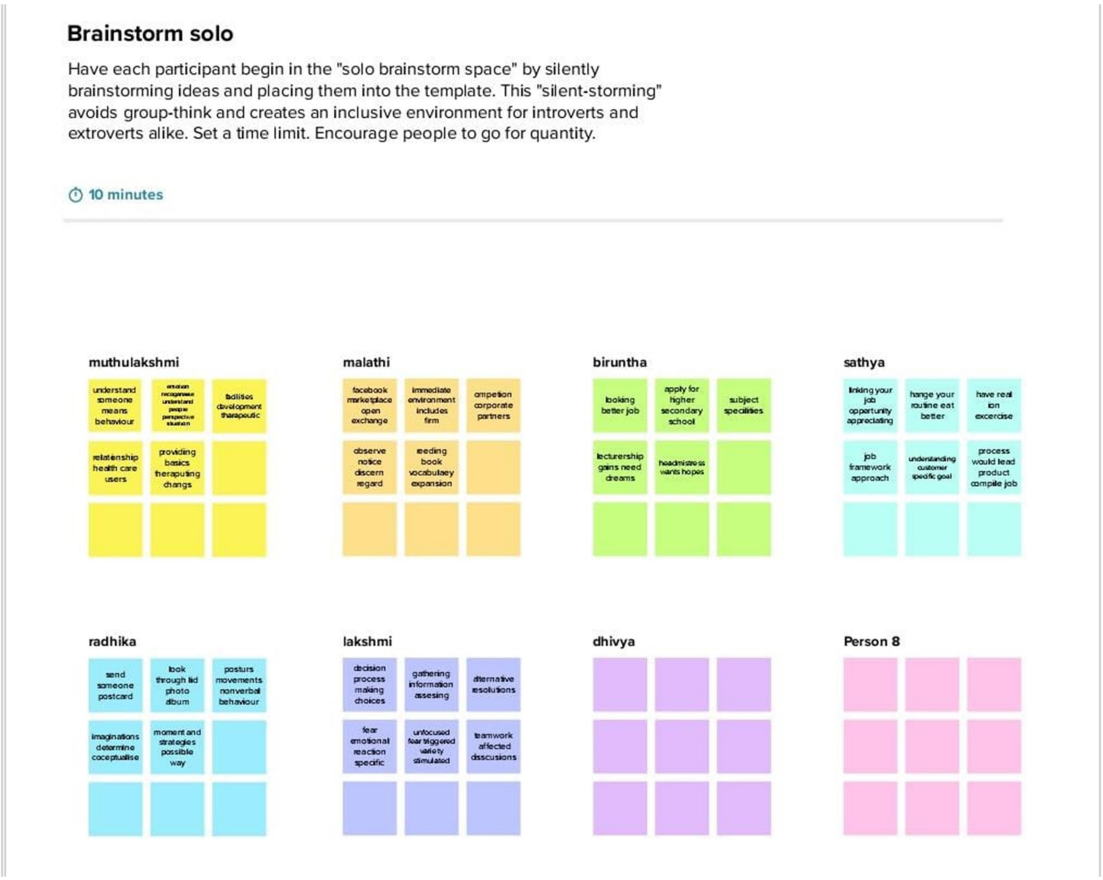
The company wants to automate the loan eligibility process (real-time) based on customer detail provided while filling out online application forms. These details are Gender, Marital Status, Education, number of Dependents, Income, Loan Amount, Credit History, and others.

1. Problem Definition And Design Thinking

1.1 Empathy map

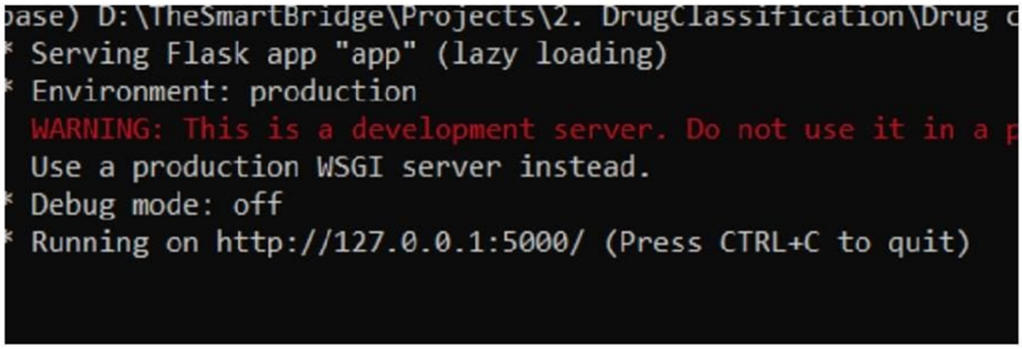


2.2 Braighnstorming Map

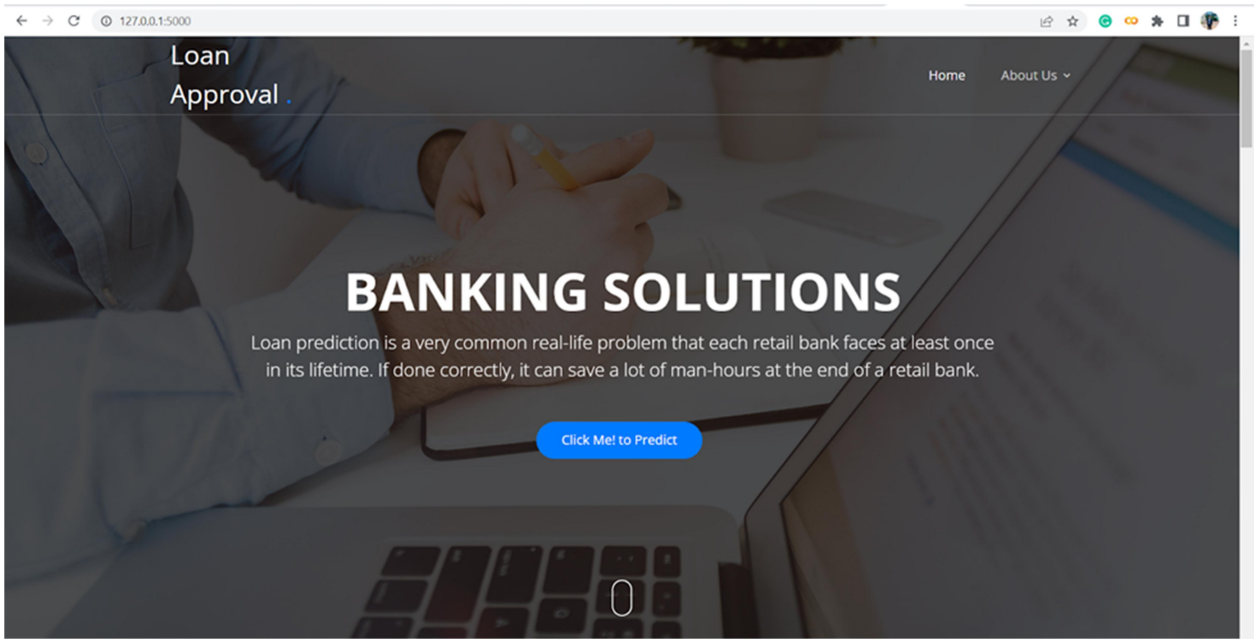


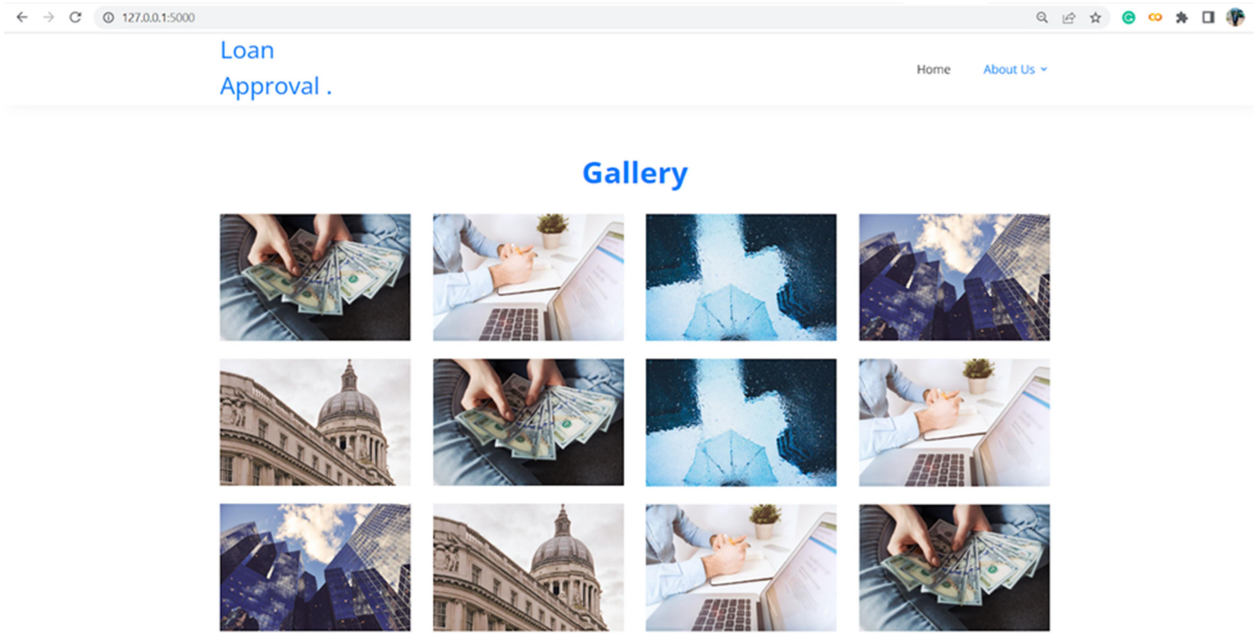
3.**RESULT** .

- Click on the predict button from the top left Open anaconda prompt from the start menu
- navigate to the folder where your python script is.
- Now type “python app.py” commands
-



-


127.0.0.1:5000
Loan
Approval .
Home
About Us
BANKING SOLUTIONS
Loan prediction is a very common real-life problem that each retail bank faces at least once in its lifetime. If done correctly, it can save a lot of man-hours at the end of a retail bank.
Click Me! to Predict


127.0.0.1:5000
Loan
Approval .
Home
About Us
Gallery

127.0.0.1:5000

Loan Approval .

Home About Us

Loan Approval How it works ?

Credit Information Bureau India Limited (CIBIL) score plays a critical role in the loan approval process for Indian banking industry. An individual customer's credit score provides loan providers with an indication of how likely it is that they will pay back a loan based on their respective credit history. This article is an attempt to discuss basics Loan Approval Process and working principles of CIBIL score in Indian finance industry keeping a view of individual customer benefits.

Learn More

127.0.0.1:5000/predict

Loan Approval .

Home About Us Contact

Loan Approval Predcition Form

Fill the Form for Prediction

Gender

-- select gender --

Married Status

select married status

Dependents

-- select dependents --

Education

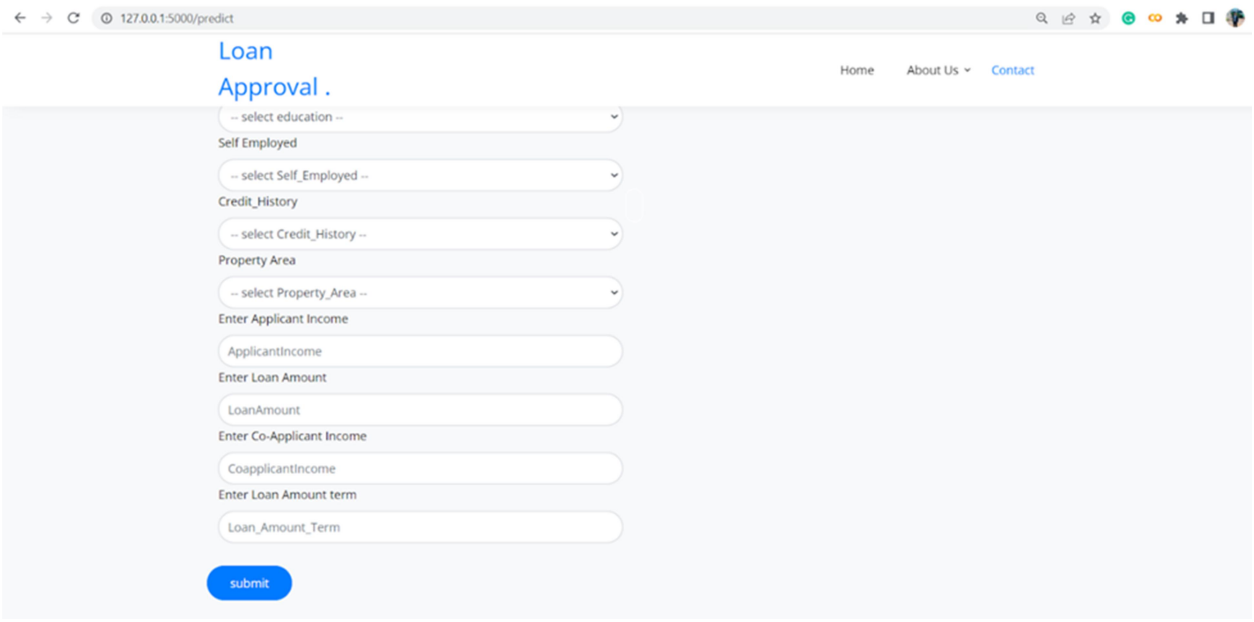
-- select education --

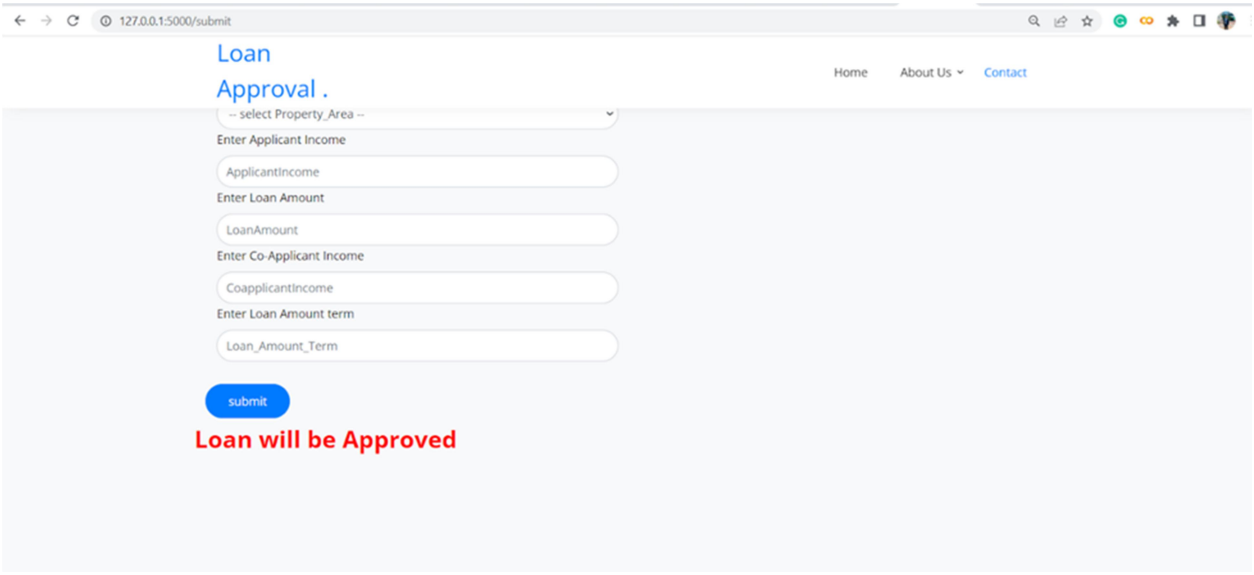
Self Employed

-- select Self_Employed --

Credit_History

-- select Credit_History --





4.ADVANTAGE

- Advantages of personal loans
- Spread the cost of a significant purchase safely
- Can help you manage your personal finances
- Ideal if you have struggled to save in the past
- Unsecured loans are not tied to assets

DISADVANTAGE

- Disadvantage of personal loans
- Long-term commitment
- Good product requires a good credit score
- Certain loan types are riskier than others
- Will never get 0% interest - unlike a credit card or finance deal

5.CONCLUSION

- From the proper view of analysis this system can be used perfect for detection of clients who are eligible for approval of loan. The software is working perfect and can be used for all banking requirements. This system can be easily uploaded in any operating system. Since the technology is moving towards online, this system has more scope for the upcoming days. This system is more secure and reliable. Since we have used Random Forest Algorithm the system returns very accurate results. There is no issue if there are many no of customers applying for loan. This system accepts data for N no. of customers. In future we can add more algorithms to this system for getting more accurate results.

Then comes the fact that today's lifestyle is more inclined towards making our lives easier and luxurious. Thus, this lifestyle often requires expensive appliances or services. This has led to a sharp rise in the expenses of an average man. Thus, the requirement for extra money which they can repay in installments later.

6.APPLICATION

Traditional processes determine the risk by manually looking at the applicant's income, credit history, and several other dynamic parameters and creating a data-driven risk model. Despite using data science in this process, there is still a large amount of manual work involved. Researchers have recently explored the possibility of using deep learning in various aspects of this process. For example, credit score and credit history are essential parameters for assessing the applicant's lending risk. DL-based approaches such as Embedding Transactional Recurrent Neural Network (E.T.-RNN) compute the credit scores of applicants by looking at the history of their credit and debit card transactions. Such an approach eliminates the high dependency on manual…

7.FUTURE SCOPE

After the personal loan officer receives a positive report from the verification team, the loan agreement is drafted once the interest rate and tenure are agreed upon by the lender

and the borrower. A loan agreement will have all the information a borrower needs to know before signing for a personal loan. Personal loans can stimulate economic growth by providing individuals with the funds they need to make major purchases, start businesses, or invest in their education.

8.APPENDIX

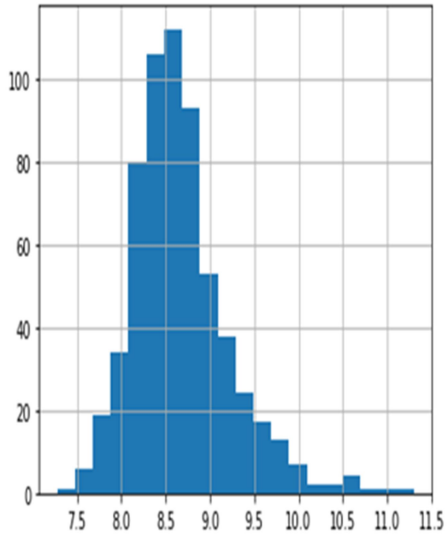
A.source code

Visual analysis is the process of using visual representations, such as charts, plots, and graphs, to explore and understand data. It is a way to quickly identify patterns, trends, and outliers in the data, which can help to gain insights and make informed decisions In simple words, univariate analysis is understanding the data with a single feature. Here we have displayed two different graphs such as distplot and countplot.

The Seaborn package provides a wonderful function distplot. With the help of we can find the distribution of the feature. To make multiple graphs in a single plot, we use subplot.

In [10]:
```
df['TotalIncome']= df['ApplicantIncome']+ df['CoapplicantIncome']
df['TotalIncome_log']= np.log(df['TotalIncome'])
df['TotalIncome_log'].hist(bins=20)
```

Out[10]: <AxesSubplot:>



Now our data is cleaned and it's time to build the model. We can train our data on different algorithms. For this project we are applying four classification algorithms. The best model is saved based on its performance.

A function named decision Tree is created and train and test data are passed as the parameters. Inside the function, Decision Tree Classifier algorithm is initialised and training data is passed to the model with the .fit() function. Test data is predicted with .predict() function and saved in a new variable. For evaluating the model, a confusion matrix and classification report is done.

```
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))
```

Building and training an Artificial Neural Network (ANN) using the library with Tensor Flow as the backend. The ANN is initialized as an instance of the Sequential class, which is a linear stack of layers. Then, the input layer and two hidden layers are added to the model using the Dense class, where the number of units and activation function are specified. The output layer is also added using the Dense class with a sigmoid activation function. The model is then compiled with the Adam optimizer, binary cross-entropy loss function, and accuracy metric. Finally, the model is fit to the training data with a batch size of 100, 20% validation split, and 100 epoch

ANN

```
# Importing the Keras libraries and packages
import tensorflow
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
```

```
[226] # Initialising the ANN
      classifier = Sequential()
```

```
[227] # Adding the input layer and the first hidden layer
      classifier.add(Dense(units=100, activation='relu', input_dim=11))
```

```
      # Adding the second hidden layer
      classifier.add(Dense(units=50, activation='relu'))
```

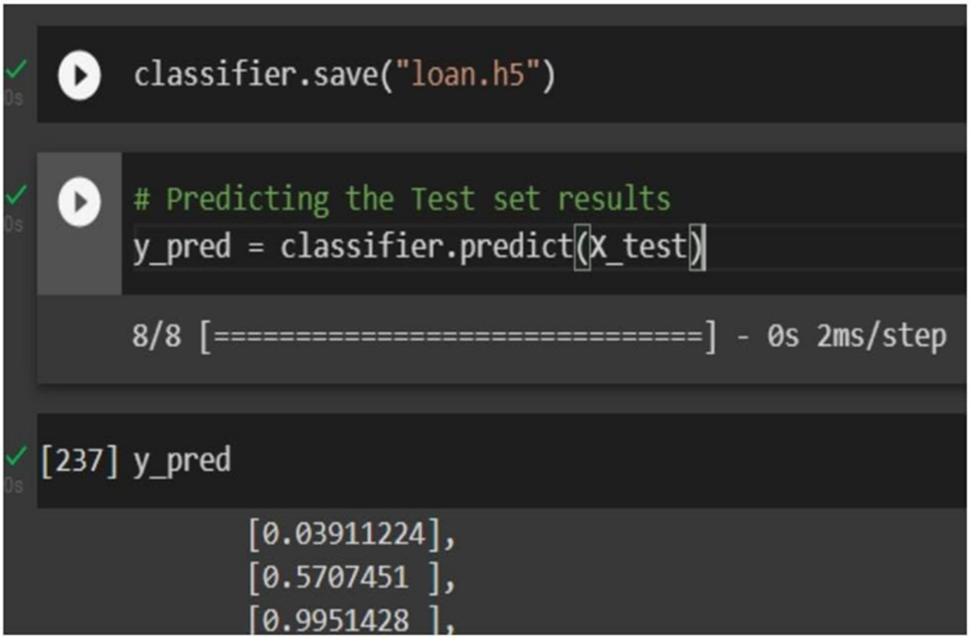
```
[229] # Adding the output layer
      classifier.add(Dense(units=1, activation='sigmoid'))
```

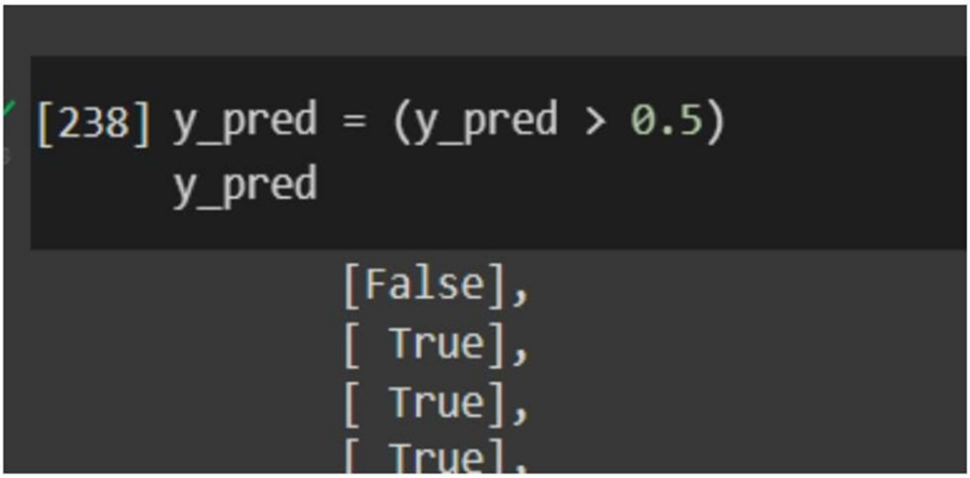
```
[230] # Compiling the ANN
      classifier.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

```
# Fitting the ANN to the Training set
model_history = classifier.fit(X_train, y_train, batch_size=100, validation_split=0.2, epochs=100)
```

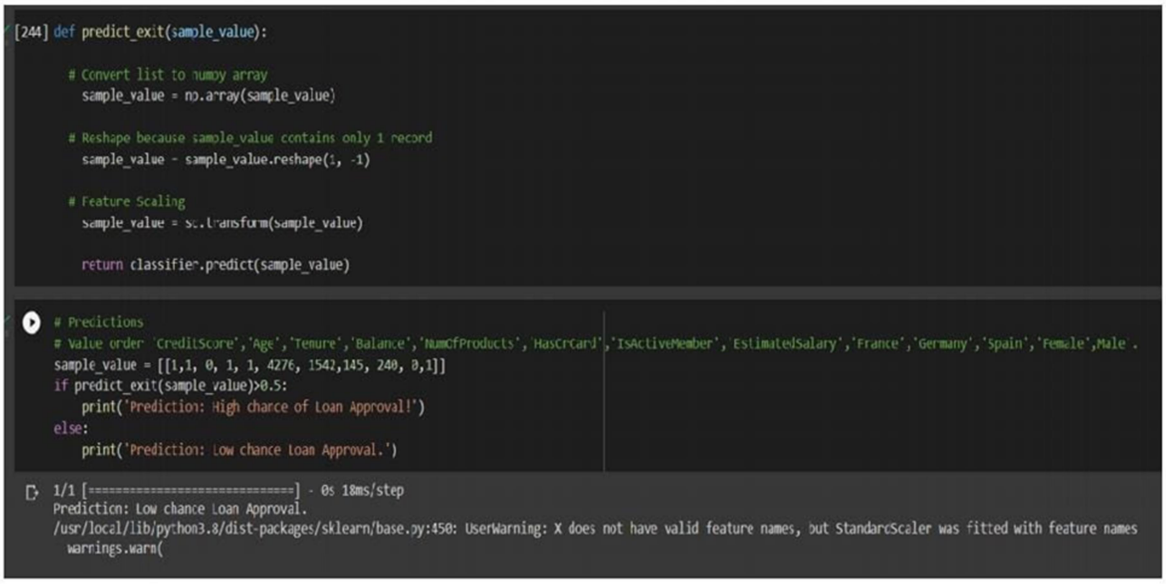
```
Epoch 72/100
4/4 [==============================] - 0s 11ms/step - loss: 0.4286 - accuracy: 0.7824 - val_loss: 0.7493 - val_accuracy: 0.6703
Epoch 73/100
4/4 [==============================] - 0s 12ms/step - loss: 0.4252 - accuracy: 0.8017 - val_loss: 0.7592 - val_accuracy: 0.6703
Epoch 74/100
4/4 [==============================] - 0s 12ms/step - loss: 0.4244 - accuracy: 0.8017 - val_loss: 0.7638 - val_accuracy: 0.6703
Epoch 75/100
4/4 [==============================] - 0s 11ms/step - loss: 0.4222 - accuracy: 0.7989 - val_loss: 0.7577 - val_accuracy: 0.6703
Epoch 76/100
4/4 [==============================] - 0s 14ms/step - loss: 0.4200 - accuracy: 0.7934 - val_loss: 0.7586 - val_accuracy: 0.6703
Epoch 77/100
4/4 [==============================] - 0s 11ms/step - loss: 0.4181 - accuracy: 0.7989 - val_loss: 0.7657 - val_accuracy: 0.6703
```

```
Epoch 95/100
4/4 [==============================] - 0s 17ms/step - loss: 0.3877 - accuracy: 0.8292 - val_loss: 0.8256 - val_accuracy: 0.6593
Epoch 96/100
4/4 [==============================] - 0s 13ms/step - loss: 0.3858 - accuracy: 0.8292 - val_loss: 0.8253 - val_accuracy: 0.6593
Epoch 97/100
4/4 [==============================] - 0s 13ms/step - loss: 0.3858 - accuracy: 0.8347 - val_loss: 0.8260 - val_accuracy: 0.6593
Epoch 98/100
4/4 [==============================] - 0s 12ms/step - loss: 0.3841 - accuracy: 0.8430 - val_loss: 0.8382 - val_accuracy: 0.6593
Epoch 99/100
4/4 [==============================] - 0s 12ms/step - loss: 0.3817 - accuracy: 0.8347 - val_loss: 0.8357 - val_accuracy: 0.6593
Epoch 100/100
4/4 [==============================] - 0s 11ms/step - loss: 0.3805 - accuracy: 0.8430 - val_loss: 0.8368 - val_accuracy: 0.6593
```





This code defines a function named "predict_exit" which takes in a sample_value as an input. The function then converts the input sample_value from a list to a numpy array. It reshapes the sample_value array as it contains only one record. Then, it applies feature scaling to the reshaped sample_value array using a scaler object 'sc' that should have been previously defined and fitted. Finally, the function returns the prediction of the classifier on the scaled sample_value.

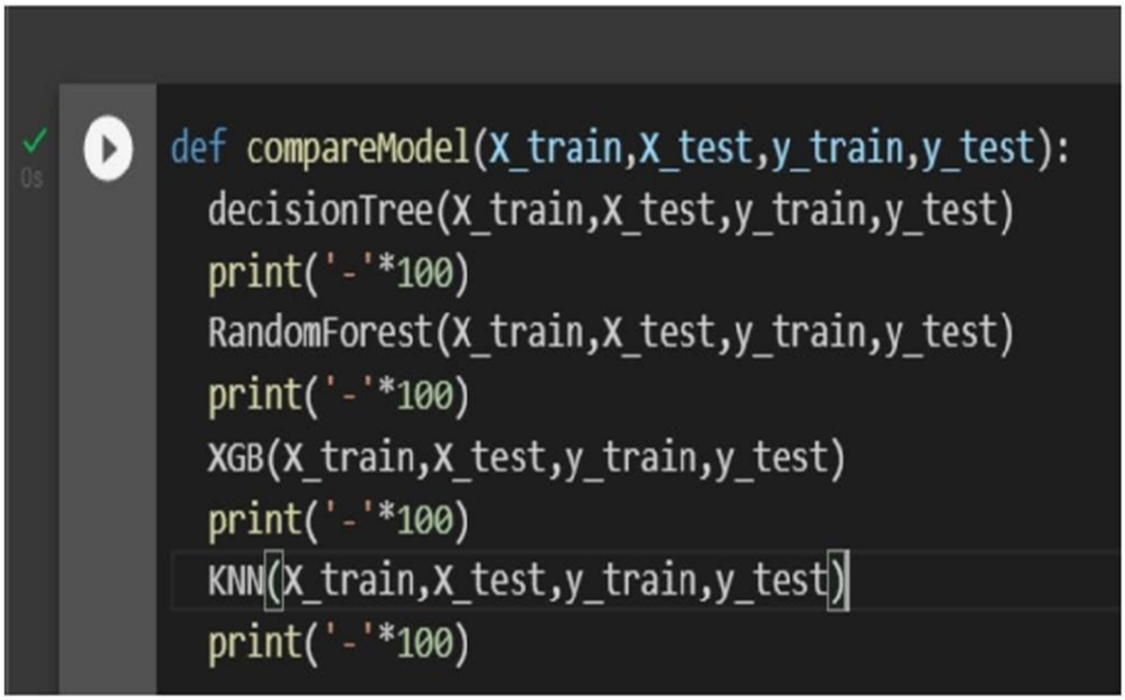


Multiple evaluation metrics means evaluating the model's performance on a test set using different performance measures. This can provide a more comprehensive understanding of the model's strengths and weaknesses. We are using evaluation metrics for classification tasks including accuracy, precision, recall, support and F1-score.

```
compareModel(X_train,X_test,y_train,y_test)
```

```
1.0
0.7822222222222223
Decision Tree
Confusion_Matrix
[[83 24]
 [25 93]]
Classification Report
              precision    recall  f1-score   support

           0       0.77      0.78      0.77       107
           1       0.79      0.79      0.79       118

    accuracy                           0.78       225
   macro avg       0.78      0.78      0.78       225
weighted avg       0.78      0.78      0.78       225

------------------------------------------------------------
```

```
------------------------------------------------------------
1.0
0.8088888888888889
Random Forest
Confusion_Matrix
[[ 78  29]
 [ 14 104]]
Classification Report
              precision    recall  f1-score   support

           0       0.85      0.73      0.78       107
           1       0.78      0.88      0.83       118

    accuracy                           0.81       225
   macro avg       0.81      0.81      0.81       225
weighted avg       0.81      0.81      0.81       225

------------------------------------------------------------
```


0s
def compareModel(X_train,X_test,y_train,y_test):
decisionTree(X_train,X_test,y_train,y_test)
print('-'*100)
RandomForest(X_train,X_test,y_train,y_test)
print('-'*100)
XGB(X_train,X_test,y_train,y_test)
print('-'*100)
KNN(X_train,X_test,y_train,y_test)
print('-'*100)

```
0.933920704845815
0.8222222222222222
XGBoost
Confusion_Matrix
[[ 78  29]
 [ 11 107]]
Classification Report
              precision    recall  f1-score   support

           0       0.88      0.73      0.80       107
           1       0.79      0.91      0.84       118

    accuracy                           0.82       225
   macro avg       0.83      0.82      0.82       225
weighted avg       0.83      0.82      0.82       225
```

```
0.7665198237885462
0.6666666666666666
KNN
Confusion_Matrix
[[60 47]
 [28 90]]
Classification Report
              precision    recall  f1-score   support

           0       0.68      0.56      0.62       107
           1       0.66      0.76      0.71       118

    accuracy                           0.67       225
   macro avg       0.67      0.66      0.66       225
weighted avg       0.67      0.67      0.66       225
```

```
0.933920704845815
0.8222222222222222
XGBoost
Confusion_Matrix
[[ 78  29]
 [ 11 107]]
Classification Report
              precision    recall  f1-score   support

           0       0.88      0.73      0.80       107
           1       0.79      0.91      0.84       118

    accuracy                           0.82       225
   macro avg       0.83      0.82      0.82       225
weighted avg       0.83      0.82      0.82       225
```

```
0.7665198237885462
0.6666666666666666
KNN
Confusion_Matrix
[[60 47]
 [28 90]]
Classification Report
              precision    recall  f1-score   support

           0       0.68      0.56      0.62       107
           1       0.66      0.76      0.71       118

    accuracy                           0.67       225
   macro avg       0.67      0.66      0.66       225
weighted avg       0.67      0.67      0.66       225
```

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

```
8/8 [==============================] - 0s 4ms/step
0.6844444444444444
ANN Model
Confusion_Matrix
[[63 44]
 [27 91]]
Classification Report
              precision    recall  f1-score   support

           0       0.70      0.59      0.64       107
           1       0.67      0.77      0.72       118

    accuracy                           0.68       225
   macro avg       0.69      0.68      0.68       225
weighted avg       0.69      0.68      0.68       225
```

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

```
8/8 [==============================] - 0s 4ms/step
0.6844444444444444
ANN Model
Confusion_Matrix
[[63 44]
 [27 91]]
Classification Report
              precision    recall  f1-score   support

           0       0.70      0.59      0.64       107
           1       0.67      0.77      0.72       118

    accuracy                           0.68       225
   macro avg       0.69      0.68      0.68       225
weighted avg       0.69      0.68      0.68       225
```

```
from sklearn.model_selection import cross_val_score

# Random forest model is selected

rf = RandomForestClassifier()
rf.fit(x_train,y_train)
yPred = rf.predict(x_test)

f1_score(yPred,y_test,average='weighted')

0.9679166666666668

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

np.mean(cv)

0.985
```

```
0.9691629955947136
0.8222222222222222
Random Forest
Confusion_Matrix
[[ 77  30]
 [ 10 108]]
Classification Report
              precision    recall  f1-score   support

           0       0.89      0.72      0.79       107
           1       0.78      0.92      0.84       118

    accuracy                           0.82       225
   macro avg       0.83      0.82      0.82       225
weighted avg       0.83      0.82      0.82       225

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   2 out of   2 | elapsed:    0.0s remaining:    0.0s
```

```
#saviung the model by using pickle function
pickle.dump(model,open('rdf.pkl','wb'))
```

```
@app.route('/submit',methods=["POST","GET"])# route to show the predictions in a web UI
def submit():
    #  reading the inputs given by the user
    input_feature=[int(x) for x in request.form.values() ]
    #input_feature = np.transpose(input_feature)
    input_feature=[np.array(input_feature)]
    print(input_feature)
    names = ['Gender', 'Married', 'Dependents', 'Education', 'Self_Employed', 'ApplicantIncome',
       'CoapplicantIncome','LoanAmount','Loan_Amount_Term','Credit_History','Property_Area']
    data = pandas.DataFrame(input_feature,columns=names)
    print(data)

    #data_scaled = scale.fit_transform(data)
    #data = pandas.DataFrame(,columns=names)



     # predictions using the loaded model file
    prediction=model.predict(data)
    print(prediction)
    prediction = int(prediction)
    print(type(prediction))

    if (prediction == 0):
       return render_template("output.html",result ="Loan wiil Not be Approved")
    else:
       return render_template("output.html",result = "Loan will be Approved")
     # showing the prediction results in a UI
```

In the above example, '/' URL is bound with the home.html function. Hence, when the home page of the web server is opened in the browser, the html page will be rendered. Whenever you enter the values from the html page the values can be retrieved using POST Method.

we are routing our app to predict() function. This function retrieves all the values from the HTML page using Post request. That is stored in an array. This array is passed to the model.predict() function. This function returns the prediction. And this prediction value will be rendered to the text that we have mentioned in the submit.html page earlier.