

Loan Prediction Problem

A project report submitted for the partial fulfillment of the
Bachelor of Technology Degree
in
IT under
Maulana Abul Kalam Azad University of Technology
BY
AMIT KUMAR JHA
ROLL NO: 10400217129

Under the Guidance of:

Prof Swagatam Basu

Department of Information Technology

For the Academic Year 2020-2021



Institute of Engineering & Management

Management House, D-1, Sector-V, Salt Lake Electronics Complex,
Kolkata-700091 Affiliated To:



MAULANA ABUL KALAM AZAD UNIVERSITY OF TECHNOLOGY, WEST BENGAL
(FORMERLY KNOWN AS WEST BENGAL UNIVERSITY OF TECHNOLOGY)
Main Campus : Haringhata, Nadia, Pin-741249
Kolkata Campus : BF-142, SECTOR-I, SALT LAKE CITY, KOLKATA-700 064, (INDIA)
Website : www.wbut.ac.in

Maulana Abul Kalam Azad University of Technology

BF-142, Salt Lake, Sector I, Kolkata-700064

JUNE 2021

1



**INSTITUTE
OF ENGINEERING & MANAGEMENT**

Salt Lake Electronics Complex, Kolkata - 700091, WB, INDIA

Phone : (033) 2357 2969/2059/2995

(033) 2357 8189/8908/5389

Fax : 91 33 2357 8302

E-mail : director@iemcal.com

Website : www.iemcal.com

CERTIFICATE

TO WHOM IT MAY CONCERN

This is to certify that the project report entitled “**Loan Prediction Problem**”, submitted by

AMIT KUMAR JHA Roll no-10400217129.

Students of **INSTITUTE OF ENGINEERING & MANAGEMENT**, in partial fulfilment of requirements for the award of the degree of **Bachelor of Technology in Information Technology**, is a bonafide work carried out under the supervision and guidance of **Prof. Swagatam Basu** during the final year of the academic session of 2017-2021. The content of this report has not been submitted to any other University or Institute for the award of any other degree. It is further certified that work is entirely original and its performance has been found to be quite satisfactory.

Prof. Swagatam Basu

Project Mentor

Dept. of Information Technology

Institute of Engineering & Management

Prof. Indraneel Mukhopadhyay

H.O.D (IT)

Dept. of Information Technology

Institute of Engineering & Management

ACKNOWLEDGEMENT

We should like to take this opportunity to extend our gratitude to the following revered persons without whose immense support, completion of this project wouldn't have been possible.

We are sincerely grateful to our advisor and mentor **Prof. Swagatam Basu, Information Technology**, IEM Kolkata, for his constant support, significant insights and for generating in us a profound interest for this subject that kept us motivated during the entire duration of this project.

We would also like to express our sincere gratitude to **Prof. Indraneel Mukhopadhyay**, HOD of Information Technology , **Satyajit Chakrabarti** (Director, IEM) and other faculties of Institute of Engineering & Management, for their assistance and encouragement.

Last but not the least, we would like to extend our warm regards to our families and peers who have kept supporting us and always had faith in our work.

Amit Kumar Jha
Roll No -10400217129.
Dept. of IT
Institute of Engineering & Management, Kolkata

ABSTRACT

With the enhancement in the banking sector lots of people are applying for bank loans but the bank has its limited assets which it has to grant to limited people only, so finding out to whom the loan can be granted which will be a safer operation for the bank is a typical process. So in this paper we try to reduce this risk factor behind selecting the safe person so as to save lots of bank efforts and assets. This is done by mining the Big Data of the previous records of the people to whom the loan was granted before and on the basis of these records/experiences the machine was trained using the machine learning model which give the most accurate result. The main objective of this paper is to predict whether assigning the loan to particular person will be safe or not. This paper is divided into four sections (i)Data Collection (ii) Comparison of machine learning models on collected data (iii) Training of system on most promising model (iv) Testing

Keywords:

Loan, Machine Learning, Training, Testing, Prediction

TABLE OF CONTENTS

TOPIC	PAGE NUMBER
CHAPTER 1. INTRODUCTION	
1.1. Motivation	6
1.2. Need for this Project	7
1.3.Scope of Project	7
CHAPTER 2. Project Review & Formulation	
2.1.Review.....	8
CHAPTER 3. Proposed METHODOLOGY	
3.1. Collection of Dataset.....	8
3.2Classification of different variable.....	8

3.3 Graph Plotting.....	9-14
3.4. Different Algorithm.....	13-14
3.5. Result Analysis.....	15

CHAPTER 4.

4.1 Source Code15-29
-----------------------	---------

CHAPTER 5. RESULTS

5.1 Logistic Regression Curve	30-32
5.2 Decision tree	33
5.3	
5.4	

CHAPTER 6.CONCLUSION	34
-----------------------------------	-----------

CHAPTER 7.REFERENCES	34
-----------------------------------	-----------

Chapter-1: Introduction

Motivation

Predicting the outcome of a loan is a recurrent, crucial and difficult issue in insurance and banking.

The objective of our project is to predict whether a loan will default or not based on past information of the people.

We used a dataset provided by one of the nationalised banks, concerning almost 1000 loans issued.

Using a very structured pipeline to load and test algorithm, we reviewed two of the classification Machine Learning strategies to extract information from the data provided by banks.

Need for this Project

Distribution of the loans is the core business part of almost every banks. The main portion of the bank's assets is directly came from the profit earned from the loans distributed by the banks. Today many banks/financial companies approves loan after a rigorous process of verification and validation but still there is no surety whether the chosen applicant is the deserving right applicant out of all applicants. Through this system we can predict whether that particular applicant is safe or not and the whole process of validation of features is automated by machine learning technique. The disadvantage of this model is that it emphasise different weights to each factor but in real life some time loan can be approved on the basis of single strong factor only, which is not possible through this system.

Scope of Project

Loan prediction can provide special advantages to the bank. The Loan Prediction System can automatically calculate the weight of each features taking part in loan processing and on new test data same features are processed with respect to their associated weight. A some limit can be set for the applicant to check whether his/her loan can be sanctioned or not. Loan Prediction System allows jumping to specific application so that it can be check on priority basis. This Paper is exclusively for the managing authority of Bank/finance company, whole process

of prediction is done privately no stakeholders would be able to alter the processing. Result against particular Loan Id can be send to various department of banks so that they can take appropriate action on application. This helps all others department too carried out other formalities. In near future this module of prediction can be integrate with the automated processing system. Whenever you will fill the details of applicant the application will automatically predict the guarantee of Loan returning.

Chapter 2: Problem review & Formulation

Loan Prediction is very helpful for employee of banks as well as for the applicant also. The aim of this Paper is to provide quick, immediate and easy way to choose the deserving applicants. The Loan Prediction System can automatically calculate the weight of each features taking part in loan processing and on new test data same features are processed with respect to their associated weight. A some limit can be set for the applicant to check whether his/her loan can be sanctioned or not. Loan Prediction System allows jumping to specific application so that it can be check on priority basis. This Paper is exclusively for the managing authority of Bank/finance company, whole process of prediction is done privately no stakeholders would be able to alter the processing. In near future this module of prediction can be integrate with the automated processing system. Whenever you will fill the details of applicant the application will automatically predict the guarantee of Loan returning.

Chapter 3: Proposed Methodology

3.1 Collection of Data Sets

It consists into approximately 10,000 samples of loans granted by the bank, with the full set of informations about the borrower, the history of payments and the outcome of the loan. The dataset is quite clean and the figures can be considered as ground truth, but lots of columns are either irrelevant, very sparse or non informative. Moreover, the dataset is very unbalanced, with approximately 17% of loans considered as defaulted. Since the objective is to predict the outcome from the informations gathered at the signature of the loan, we cannot use the data concerning the history of payments or the current situation of a loan. Excluding features for which the information is incomplete, or uninformative, we get a total of 19 features, that cover personal information (credit grade, income, housing status, ...) and credit information (amount, interest rates, ...). Accuracy is not well-suited for our problem. The unbalance of the classes would lead an algorithm to never predict a default. Prediction score allows us to quantify a good prediction on both precision and recall.[4]

3.2 Classification of Different Variables

Loan_ID	614 non-null object
Gender	601 non-null object
Married	611 non-null object

Dependents 599 non-null object
 Education 614 non-null object
 Self_Employed 582 non-null object
 ApplicantIncome 614 non-null int64
 CoapplicantIncome 614 non-null float64
 LoanAmount 592 non-null float64

 Loan_Amount_Term 600 non-null float64

 Credit_History 564 non-null float64

 Property_Area 614 non-null object

 Loan_Status 614 non-null object

 dtypes: float64(4), int64(1), object(8)

 memory usage: 62.5+ KB

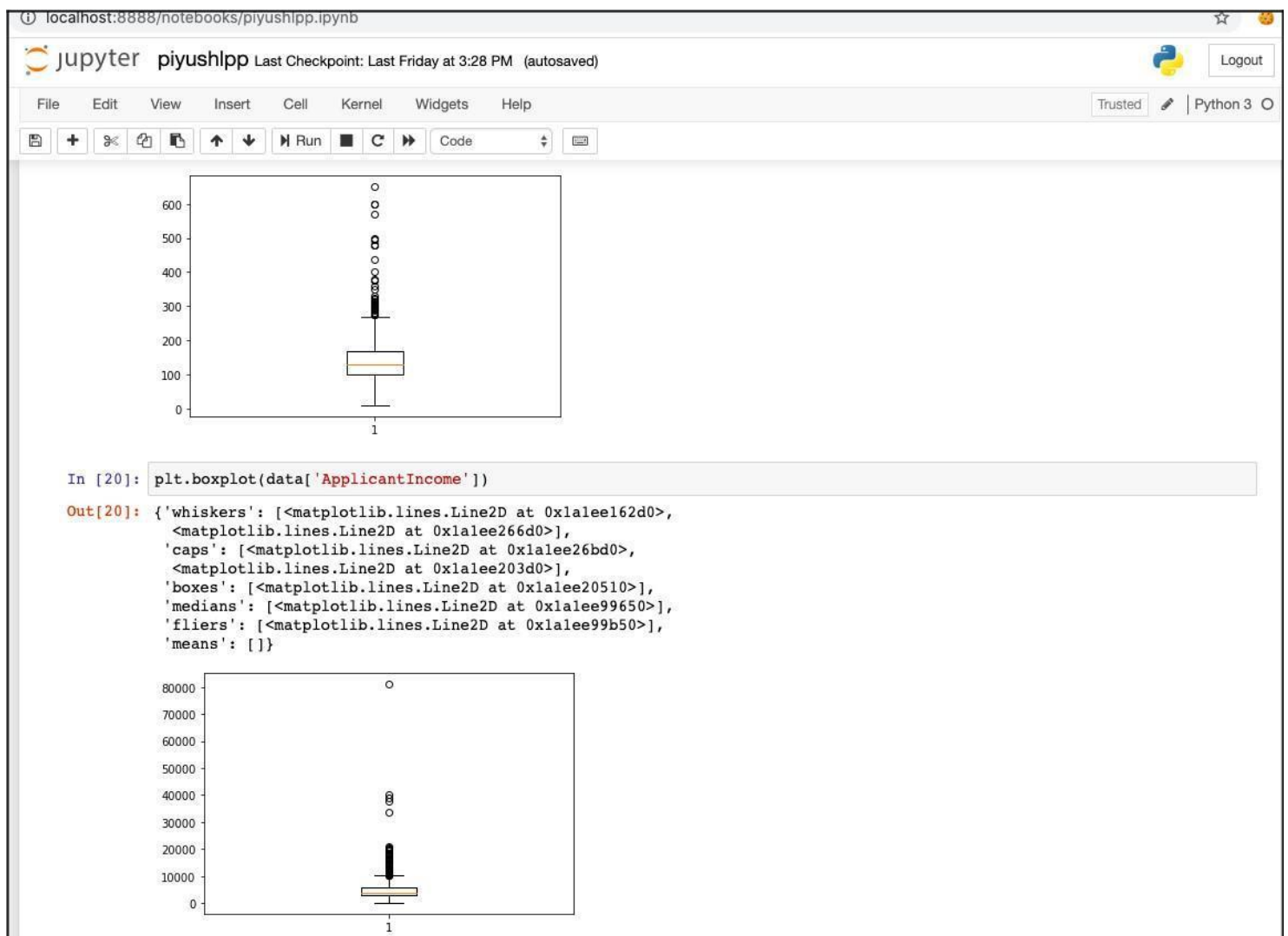
3.3 Graphs plotting

3.3.1.Univariate Graph

Use of one variable to plot a graph(Box-plot).

Applicant income

Fig.1.Box Plot Applicant Income



Loan Amount Term

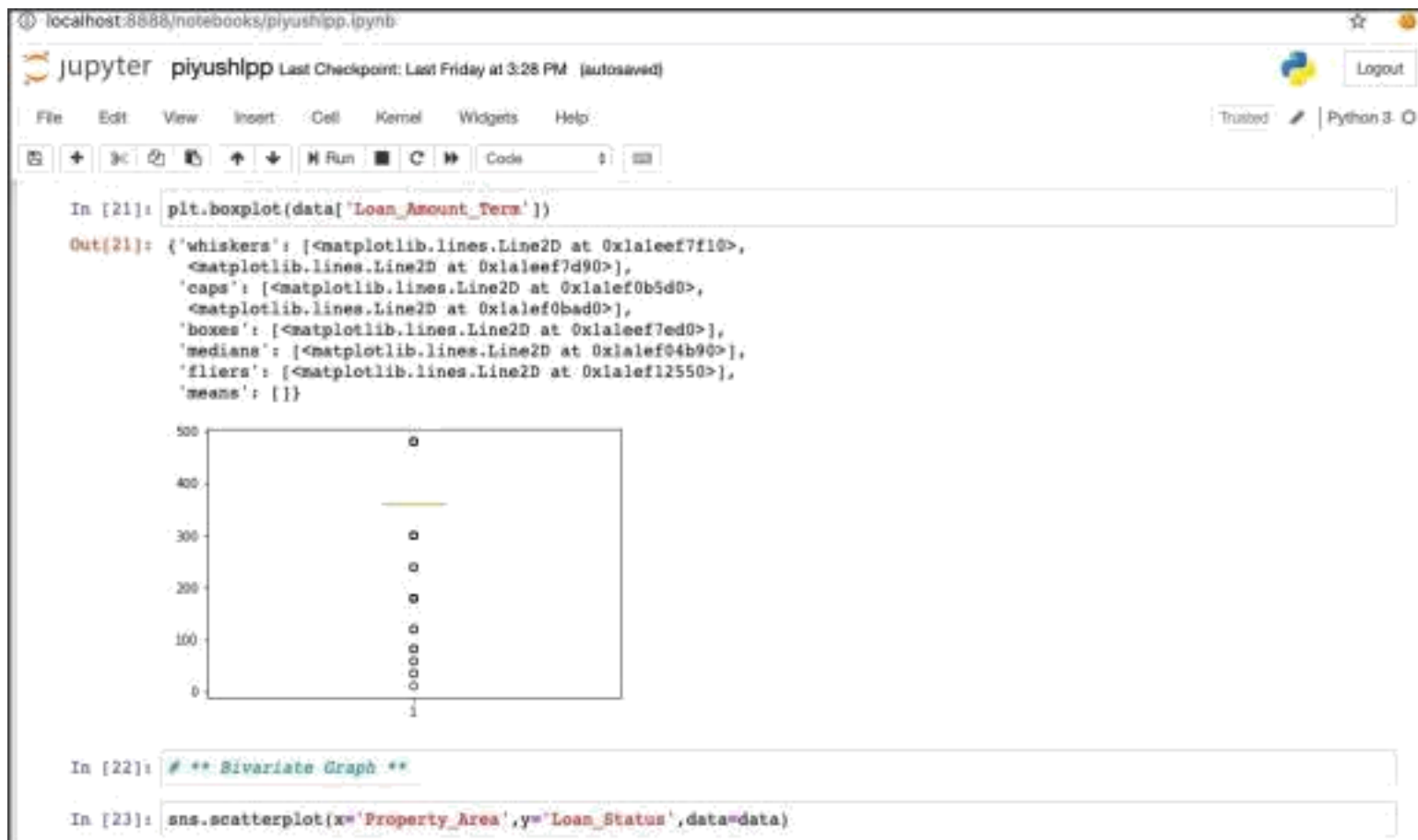


Fig.2.Loan Amount Term(Box Plot)

3.3.2. Bivariate Graph

1.Property Area Vs Loan Status(Box Plot) 2.Self Employed vs Loan Status(Box Plot)

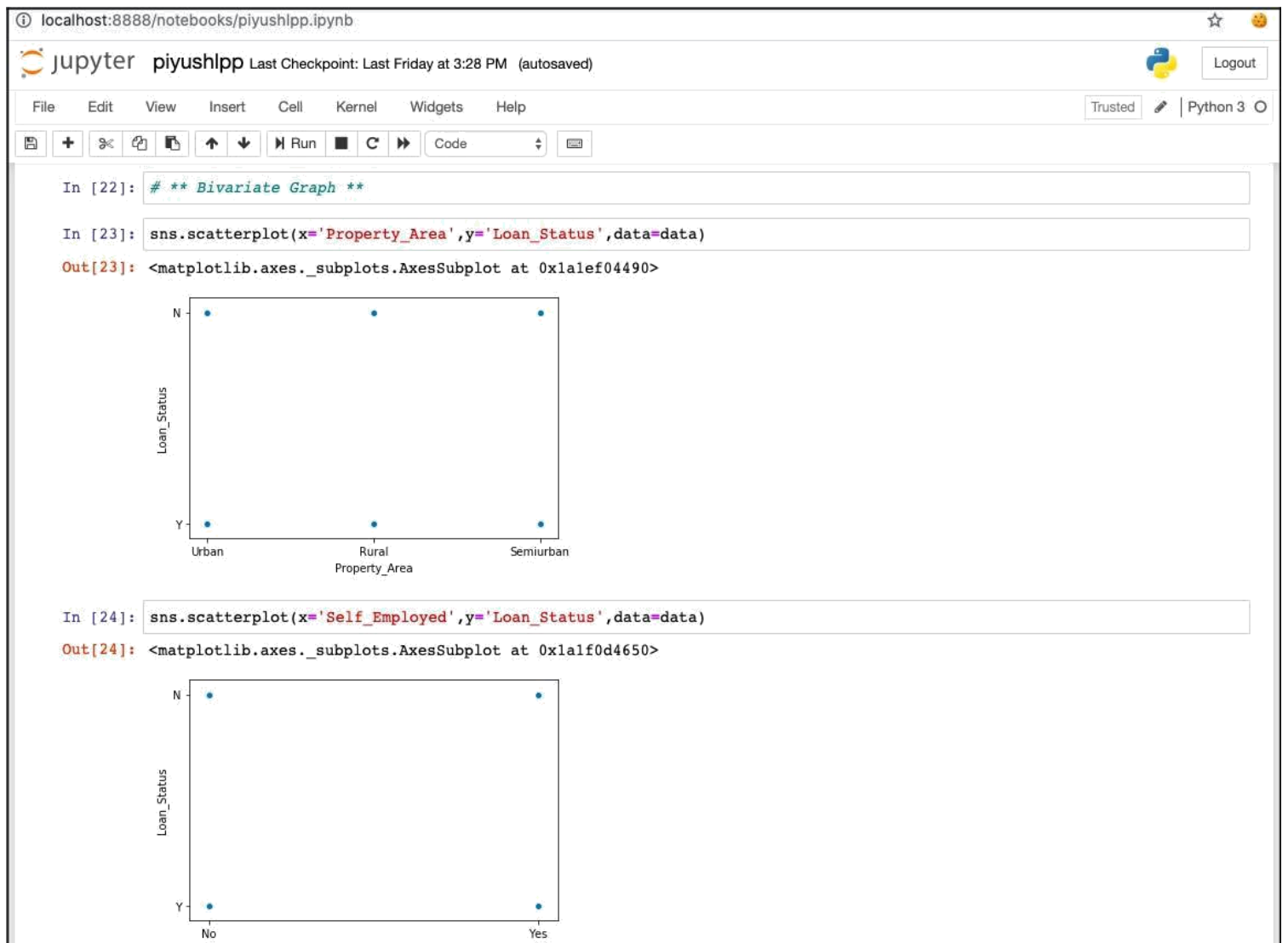


Fig.3.Property Area Vs Loan Status(Box Plot)

Property Area Vs Loan Status(Box Graph)

1.Urban

2.Rural

3.Semiurban

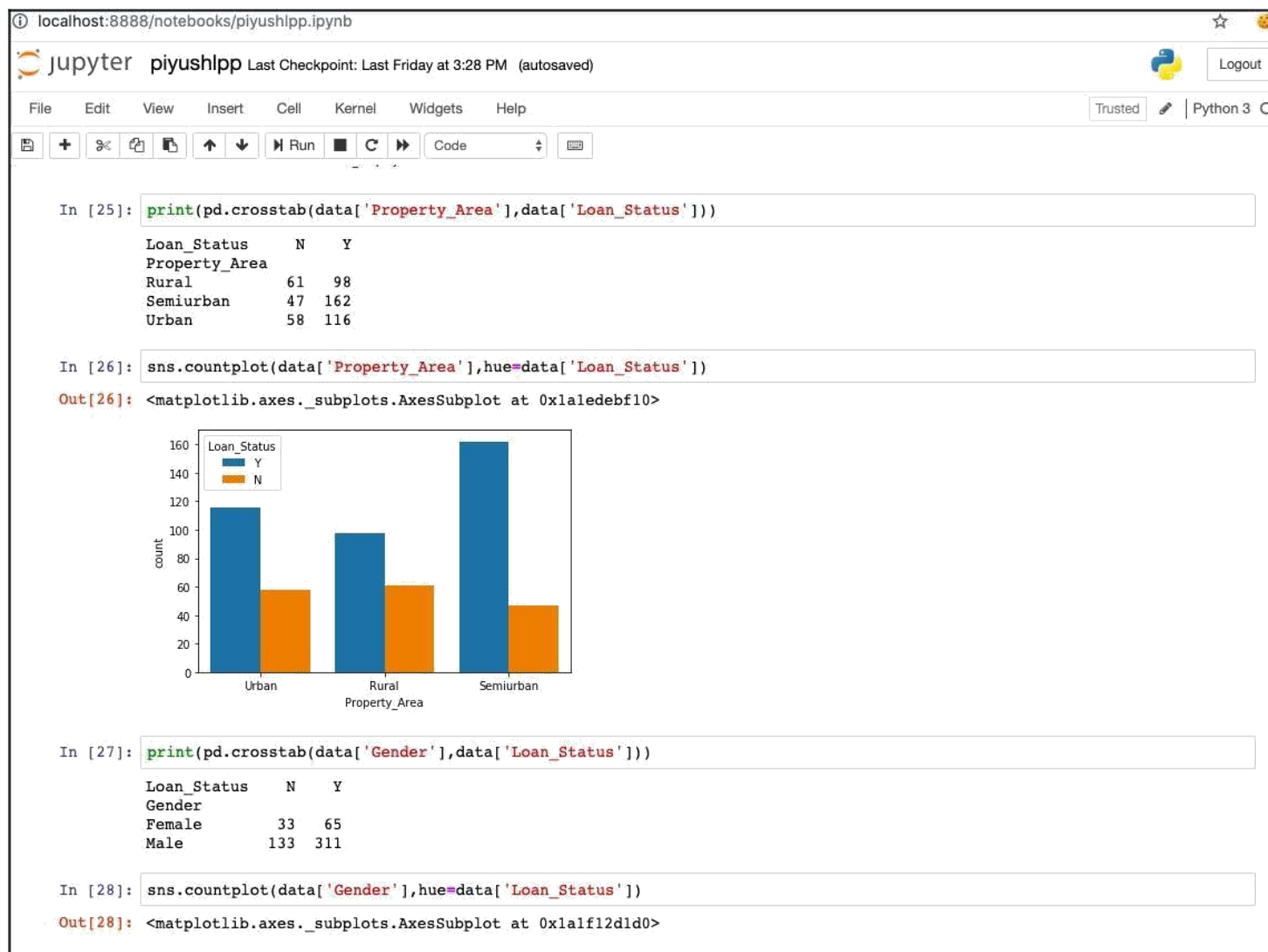


Fig.4.Property Area Vs Loan Status(Bar Graph)

Gender Vs Loan Status(Bar Graph)

1.Male

2.Female

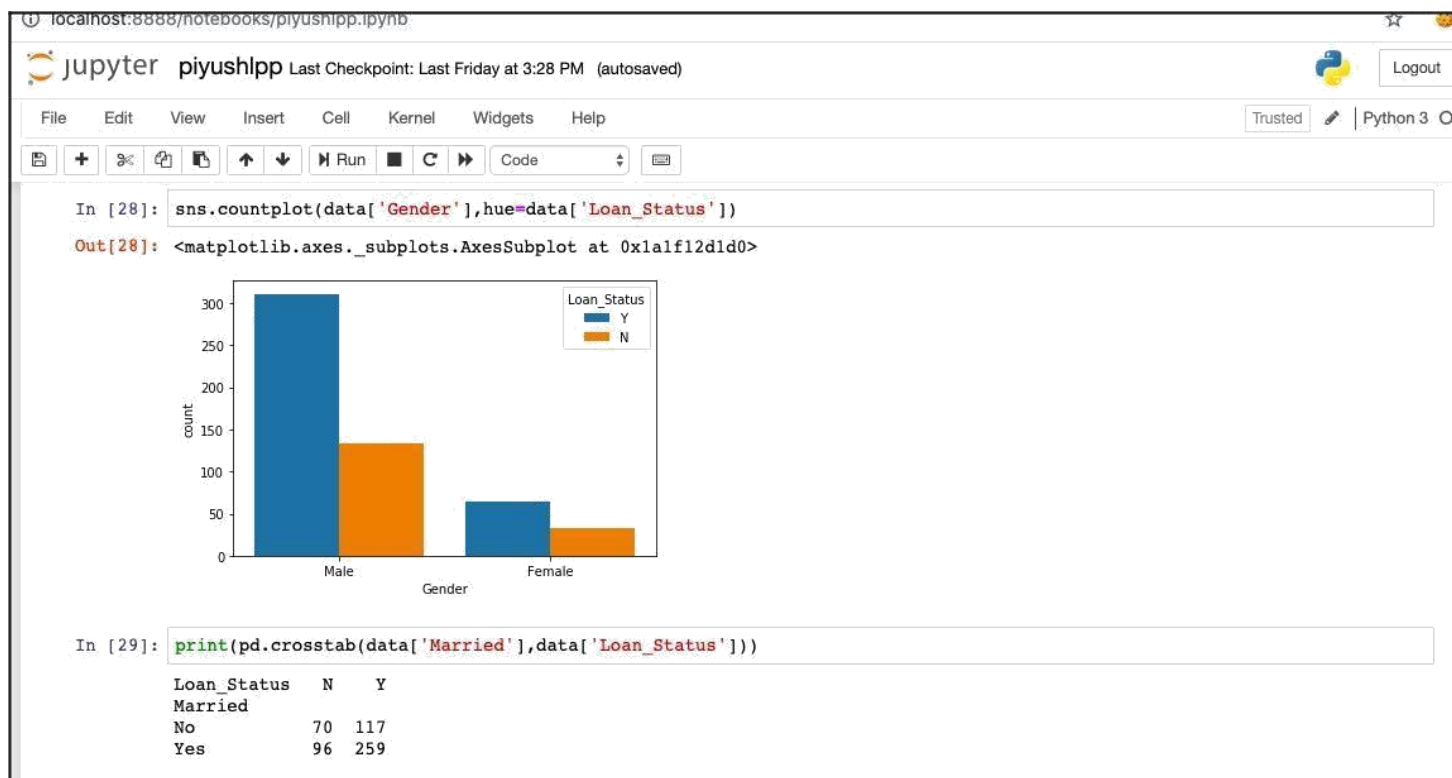


Fig.5.Gender Vs Loan Status(Box Plot)

Married Vs Loan Status(Bar Graph)

1.Yes(Married)

2.No(Not Married)

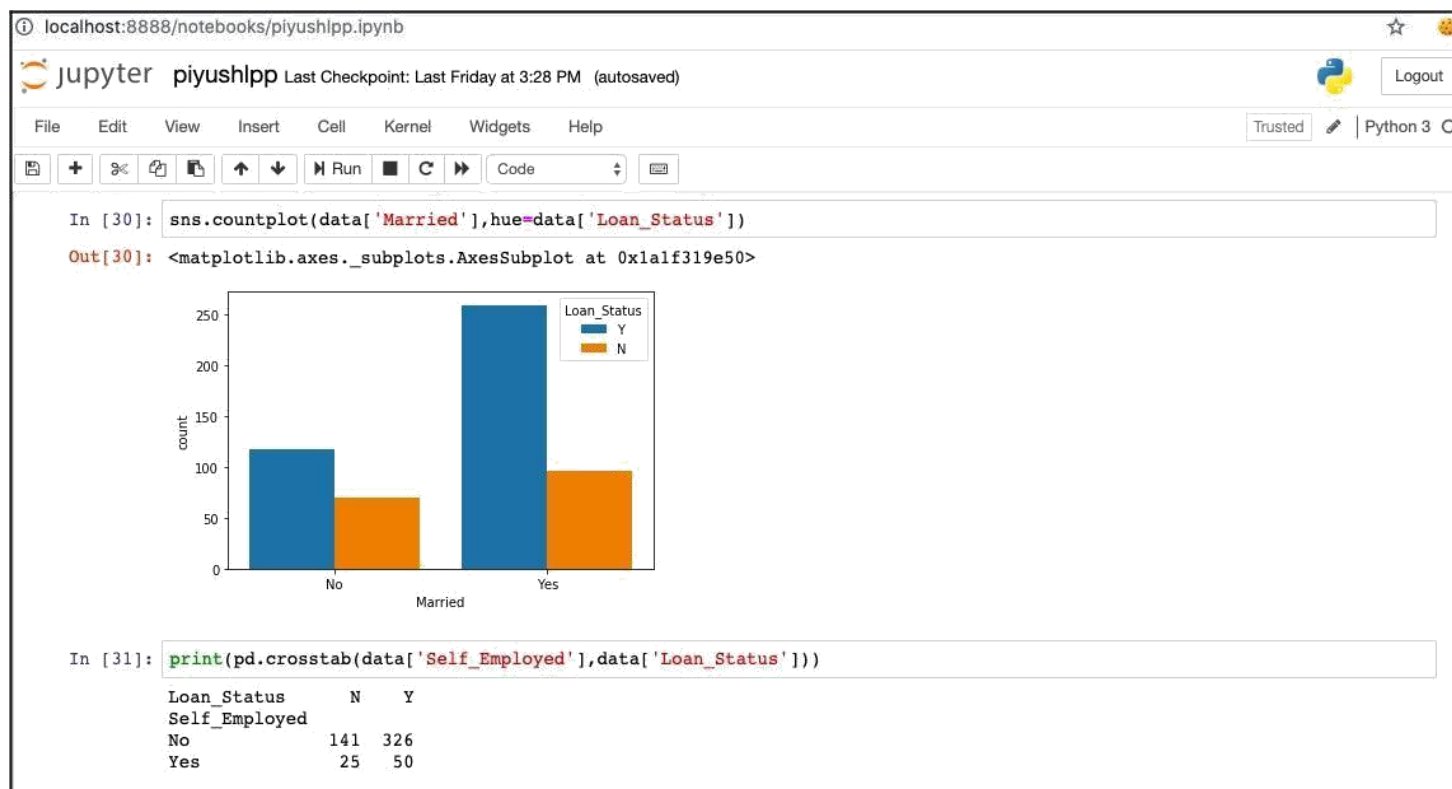


Fig.6.Marital Status Vs Loan Status(Bar Graph)

Education Vs Loan Status(Bar Graph)

1.Graduate

2.Not Graduate

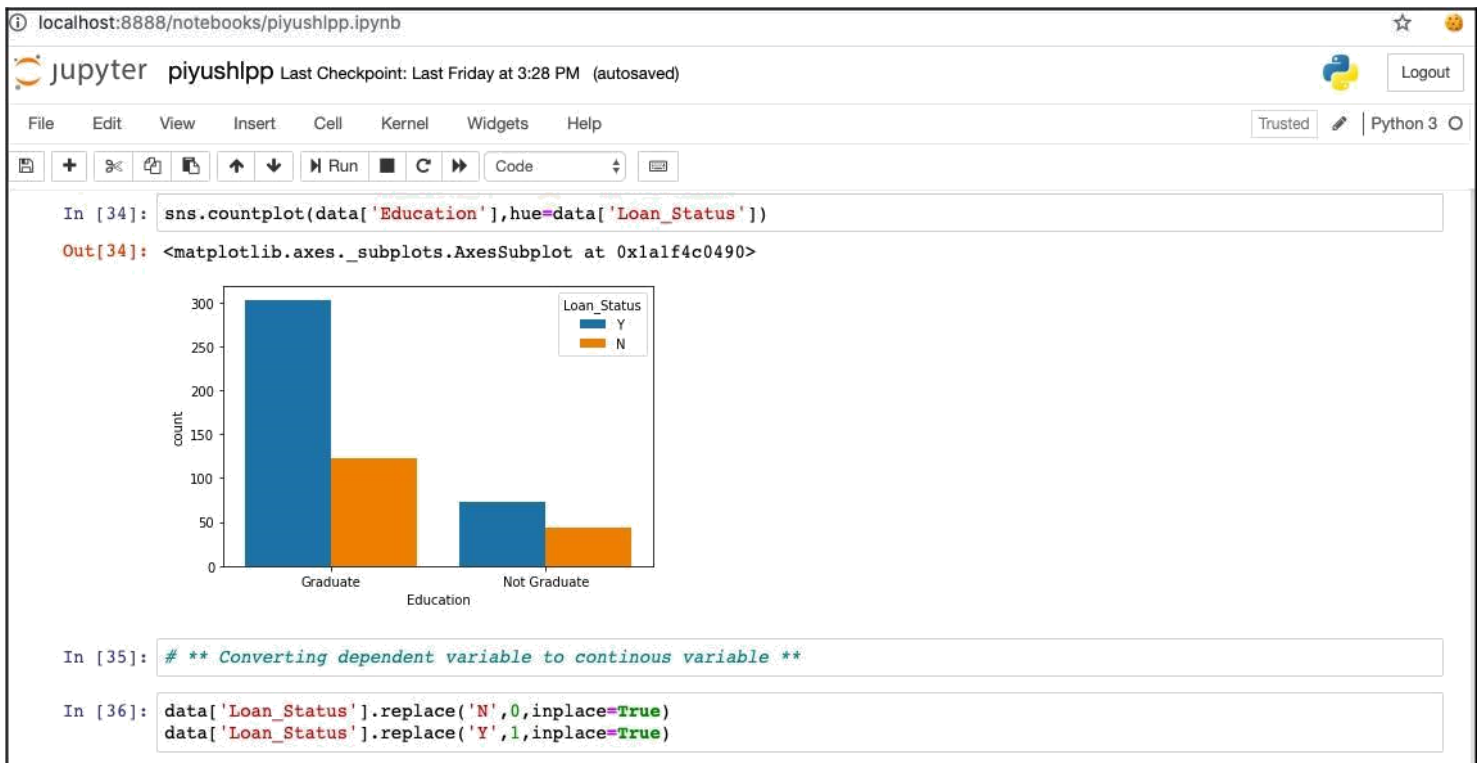


Fig.7.Education Vs Loan Status(Bar Graph)

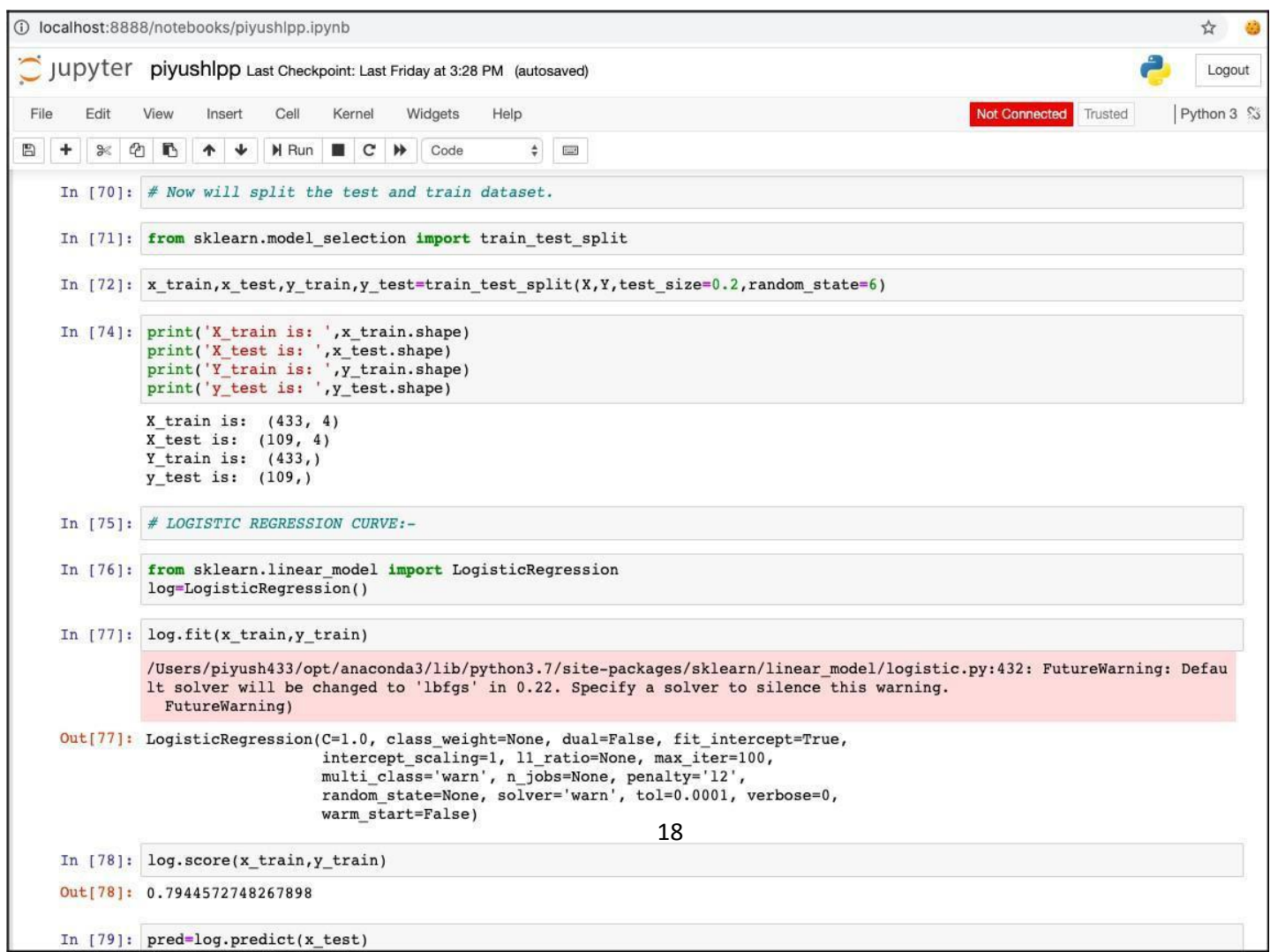
3.4 Test the model on Algorithm

3.4.1. Logistic Regression Curve

Logistic Regression is a Machine Learning algorithm which is used for the classification problems, it is a predictive analysis algorithm and based on the

concept of probability. The hypothesis of logistic regression tends to limit the cost function between 0 and 1. [1]

Fig.8. Splitting Test and Train Dataset



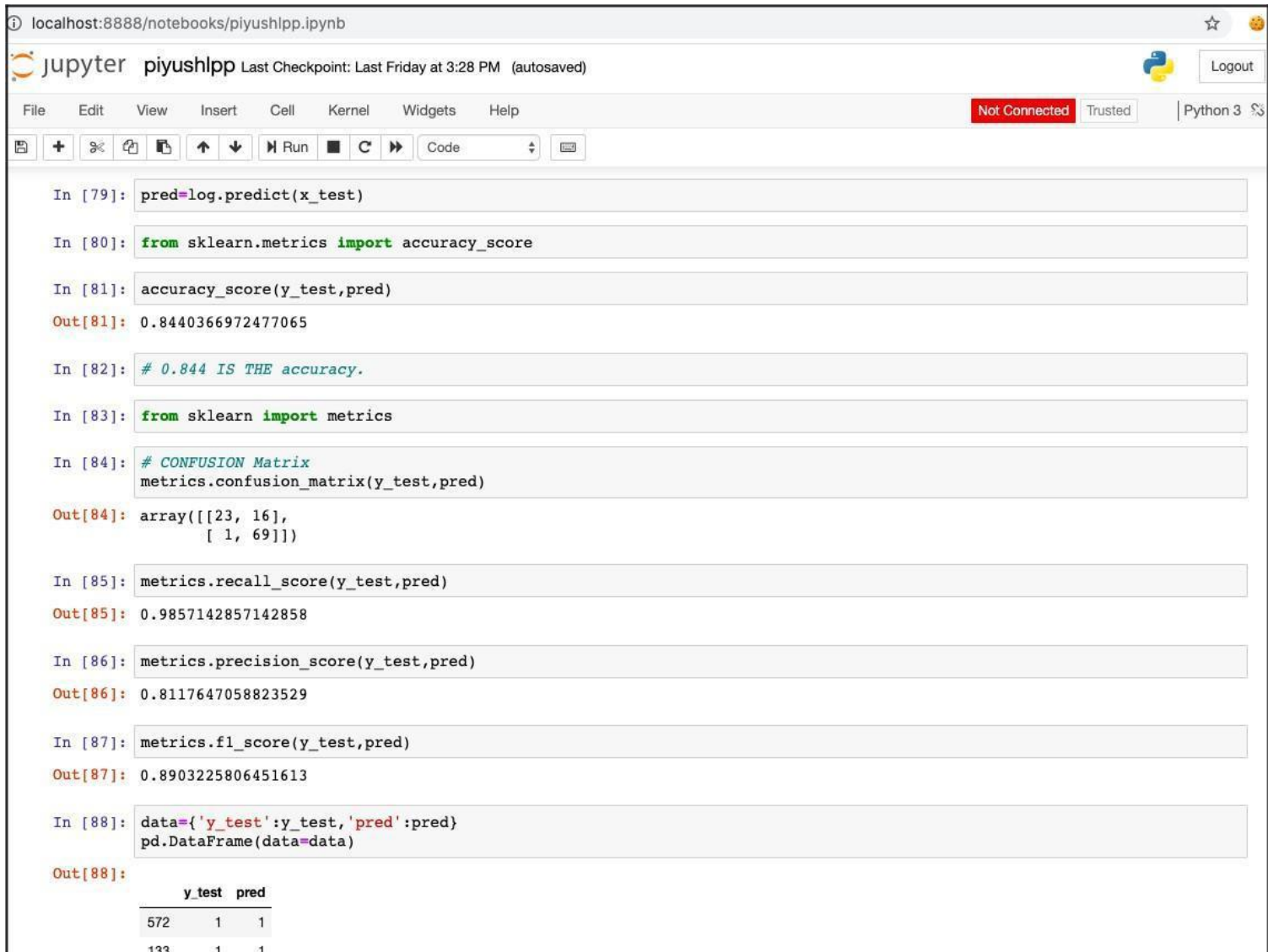
The screenshot shows a Jupyter Notebook window titled 'piyushlpp'. The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, running cells, and code execution. The notebook content consists of several code cells:

- In [70]:** A comment: `# Now will split the test and train dataset.`
- In [71]:** `from sklearn.model_selection import train_test_split`
- In [72]:** `x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=6)`
- In [74]:** A series of print statements to check the shapes of the split data:

```
print('X_train is: ', x_train.shape)
print('X_test is: ', x_test.shape)
print('Y_train is: ', y_train.shape)
print('y_test is: ', y_test.shape)
```

The output shows: `X_train is: (433, 4)`, `X_test is: (109, 4)`, `Y_train is: (433,)`, and `y_test is: (109,)`.
- In [75]:** A comment: `# LOGISTIC REGRESSION CURVE:-`
- In [76]:** `from sklearn.linear_model import LogisticRegression`
`log = LogisticRegression()`
- In [77]:** `log.fit(x_train, y_train)`
A warning message is displayed: `/Users/piyush433/opt/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py:432: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.`
- Out[77]:** The output of the fit method: `LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True, intercept_scaling=1, l1_ratio=None, max_iter=100, multi_class='warn', n_jobs=None, penalty='l2', random_state=None, solver='warn', tol=0.0001, verbose=0, warm_start=False)`
- In [78]:** `log.score(x_train, y_train)`
Out[78]: `0.7944572748267898`
- In [79]:** `pred = log.predict(x_test)`

Logis>c Regression Algorithm



The image shows a Jupyter Notebook interface for a Logistic Regression algorithm. The notebook is titled 'piyushlpp' and is running on 'localhost:8888/notebooks/piyushlpp.ipynb'. The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, running code, and other functions. The notebook content consists of several code cells and their corresponding outputs.

```
In [79]: pred=log.predict(x_test)
```

```
In [80]: from sklearn.metrics import accuracy_score
```

```
In [81]: accuracy_score(y_test,pred)
```

```
Out[81]: 0.8440366972477065
```

```
In [82]: # 0.844 IS THE accuracy.
```

```
In [83]: from sklearn import metrics
```

```
In [84]: # CONFUSION Matrix
metrics.confusion_matrix(y_test,pred)
```

```
Out[84]: array([[23, 16],
               [ 1, 69]])
```

```
In [85]: metrics.recall_score(y_test,pred)
```

```
Out[85]: 0.9857142857142858
```

```
In [86]: metrics.precision_score(y_test,pred)
```

```
Out[86]: 0.8117647058823529
```

```
In [87]: metrics.f1_score(y_test,pred)
```

```
Out[87]: 0.8903225806451613
```

```
In [88]: data={'y_test':y_test,'pred':pred}
pd.DataFrame(data=data)
```

```
Out[88]:
```

	y_test	pred
572	1	1
133	1	1

Fig.9.Logistic Regression Algorithm

3.4.2. Decision Tree

The basic algorithm of decision tree requires all attributes or features should be discretised. Feature selection is based on greatest information gain of features. The knowledge depicted in decision tree can be represented in the form of IF-THEN rules. This model is an extension of C4.5 classification algorithms described by Quinlan.[2]

```

localhost:8888/notebooks/piyushlpp.ipynb
jupyter piyushlpp Last Checkpoint: Last Friday at 3:28 PM (autosaved)
File Edit View Insert Cell Kernel Widgets Help Not Connected Trusted Python 3

In [89]: # ** USING DECISION TREE **

In [90]: from sklearn.tree import DecisionTreeClassifier
         clf=DecisionTreeClassifier()

In [91]: clf.fit(x_train,y_train)
Out[91]: DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
                                max_features=None, max_leaf_nodes=None,
                                min_impurity_decrease=0.0, min_impurity_split=None,
                                min_samples_leaf=1, min_samples_split=2,
                                min_weight_fraction_leaf=0.0, presort=False,
                                random_state=None, splitter='best')

In [92]: pred1=clf.predict(x_test)

In [93]: accuracy_score(y_test,pred1)
Out[93]: 0.8440366972477065

In [100]: # This will give the number of values true+,true- etc.
          metrics.confusion_matrix(y_test,pred1)
Out[100]: array([[23, 16],
                 [ 1, 69]])

In [95]: metrics.f1_score(y_test,pred1)
Out[95]: 0.8903225806451613

In [96]: metrics.recall_score(y_test,pred1)
Out[96]: 0.9857142857142858

In [97]: metrics.precision_score(y_test,pred1)
Out[97]: 0.8117647058823529

In [97]: metrics.precision_score(y_test,pred1)
Out[97]: 0.8117647058823529

In [98]: # Through Decision tree have a accuray of 0.811

In [ ]:

```

Fig.10.Decision Tree Algorithm

3. 5.Result analysis

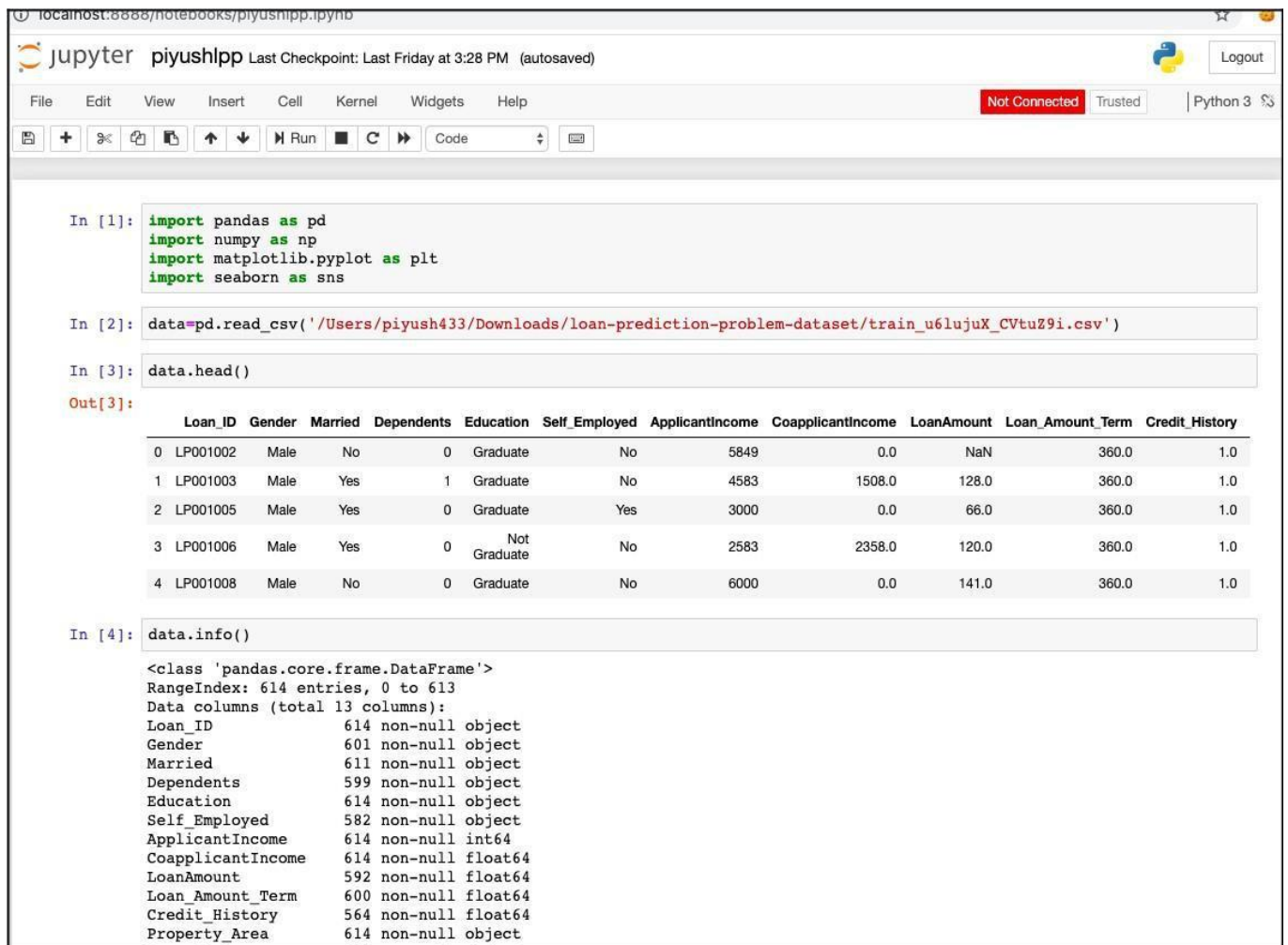
We will check for the accuracy of project with both the algorithm.As the dataset is less the accuracy of the project more than expected from the algorithm.

Chapter 4: Project Implementation

Platform used

1. Python 3
2. Anaconda Jupyter notebook
3. Web browser Chrome

Fetching Dataset



The screenshot shows a Jupyter Notebook interface with the following content:

```
In [1]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
In [2]: data=pd.read_csv('/Users/piyush433/Downloads/loan-prediction-problem-dataset/train_u6lujuX_CVtuZ9i.csv')
```

```
In [3]: data.head()
```

Out[3]:

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

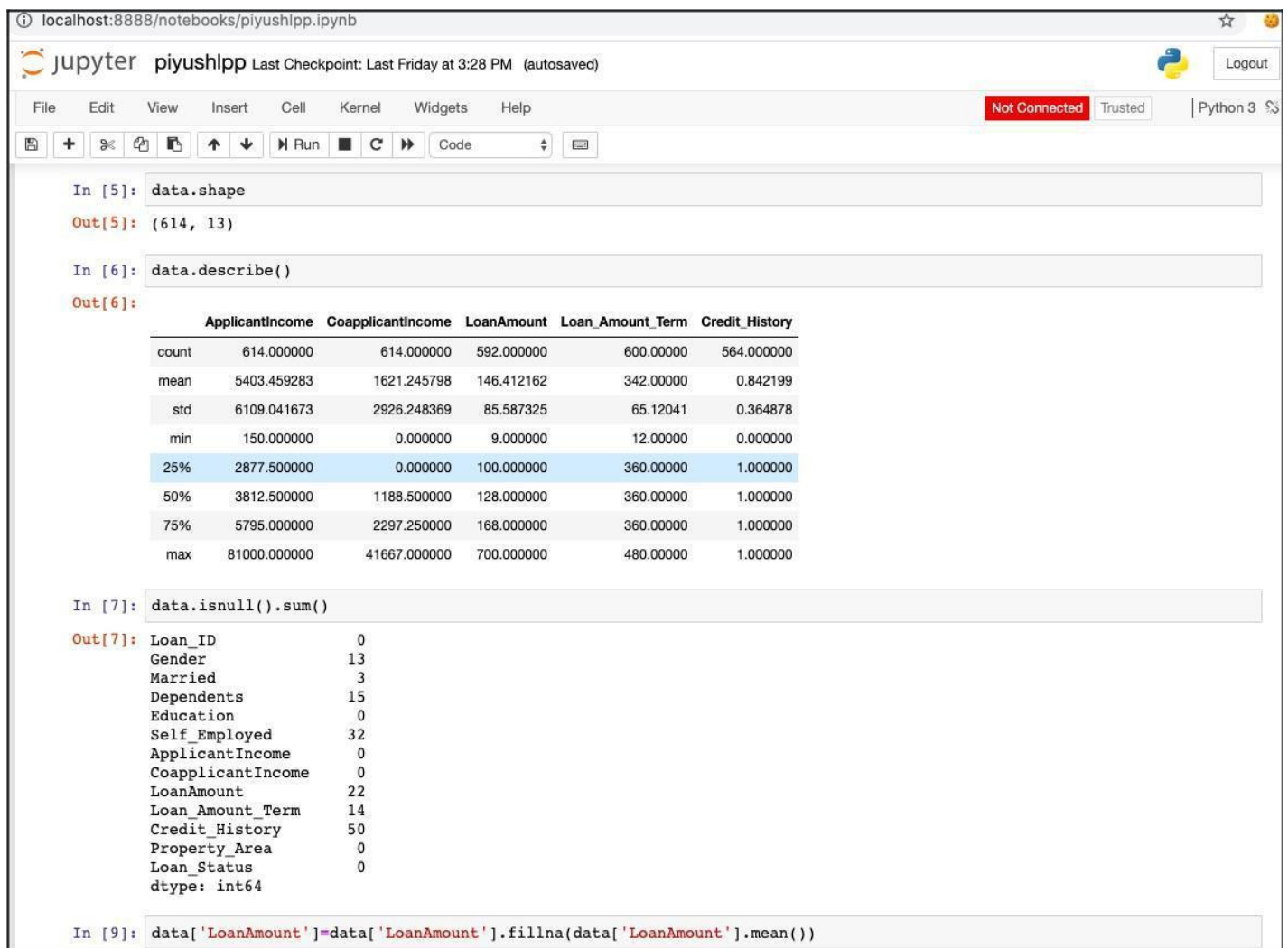
```
In [4]: data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 13 columns):
Loan_ID                614 non-null object
Gender                 601 non-null object
Married                611 non-null object
Dependents              599 non-null object
Education              614 non-null object
Self_Employed          582 non-null object
ApplicantIncome        614 non-null int64
CoapplicantIncome      614 non-null float64
LoanAmount             592 non-null float64
Loan_Amount_Term       600 non-null float64
Credit_History         564 non-null float64
Property_Area          614 non-null object
```

Fig.11.Dataset Fetching

Dividing the different Variables

Fig.12.Differentiating Independent and Dependent Variable



The screenshot shows a Jupyter Notebook window with the following content:

```
In [5]: data.shape
Out[5]: (614, 13)
```

```
In [6]: data.describe()
```

```
Out[6]:
```

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
count	614.000000	614.000000	592.000000	600.000000	564.000000
mean	5403.459283	1621.245798	146.412162	342.000000	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

```
In [7]: data.isnull().sum()
Out[7]: 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
```

```
In [9]: data['LoanAmount']=data['LoanAmount'].fillna(data['LoanAmount'].mean())
```

Correlation Matrix

For finding the variable dependent on Loan Status



Fig.13.Correlation Matrix

Splitting the Dataset for Getting training dataset

It is done by dropping the variable on which Loan Status doesn't depend.

localhost:8888/notebooks/piyushpp.ipynb

jupyter piyushpp Last Checkpoint: Last Friday at 3:28 PM (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Not Connected Trusted Python 3

In [40]: `data2=data.drop(labels=['ApplicantIncome'],axis=1)`

In [41]: `data2=data2.drop(labels=['CoapplicantIncome'],axis=1)`

In [42]: `data2=data2.drop(labels=['LoanAmount'],axis=1)`

In [43]: `data2=data2.drop(labels=['Loan_Amount_Term'],axis=1)`

In [44]: `data2=data2.drop(labels=['Loan_ID'],axis=1)`

In [45]: `data2.head()`

Out[45]:

	Gender	Married	Dependents	Education	Self_Employed	Credit_History	Property_Area	Loan_Status
0	Male	No	0	Graduate	No	1.0	Urban	1
1	Male	Yes	1	Graduate	No	1.0	Rural	0
2	Male	Yes	0	Graduate	Yes	1.0	Urban	1
3	Male	Yes	0	Not Graduate	No	1.0	Urban	1
4	Male	No	0	Graduate	No	1.0	Urban	1

In [46]: `# Changing to continuous Variable`

In [47]: `from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder
le=LabelEncoder()
ohe=OneHotEncoder()`

In [49]: `data2['Property_Area']=le.fit_transform(data2['Property_Area'])`

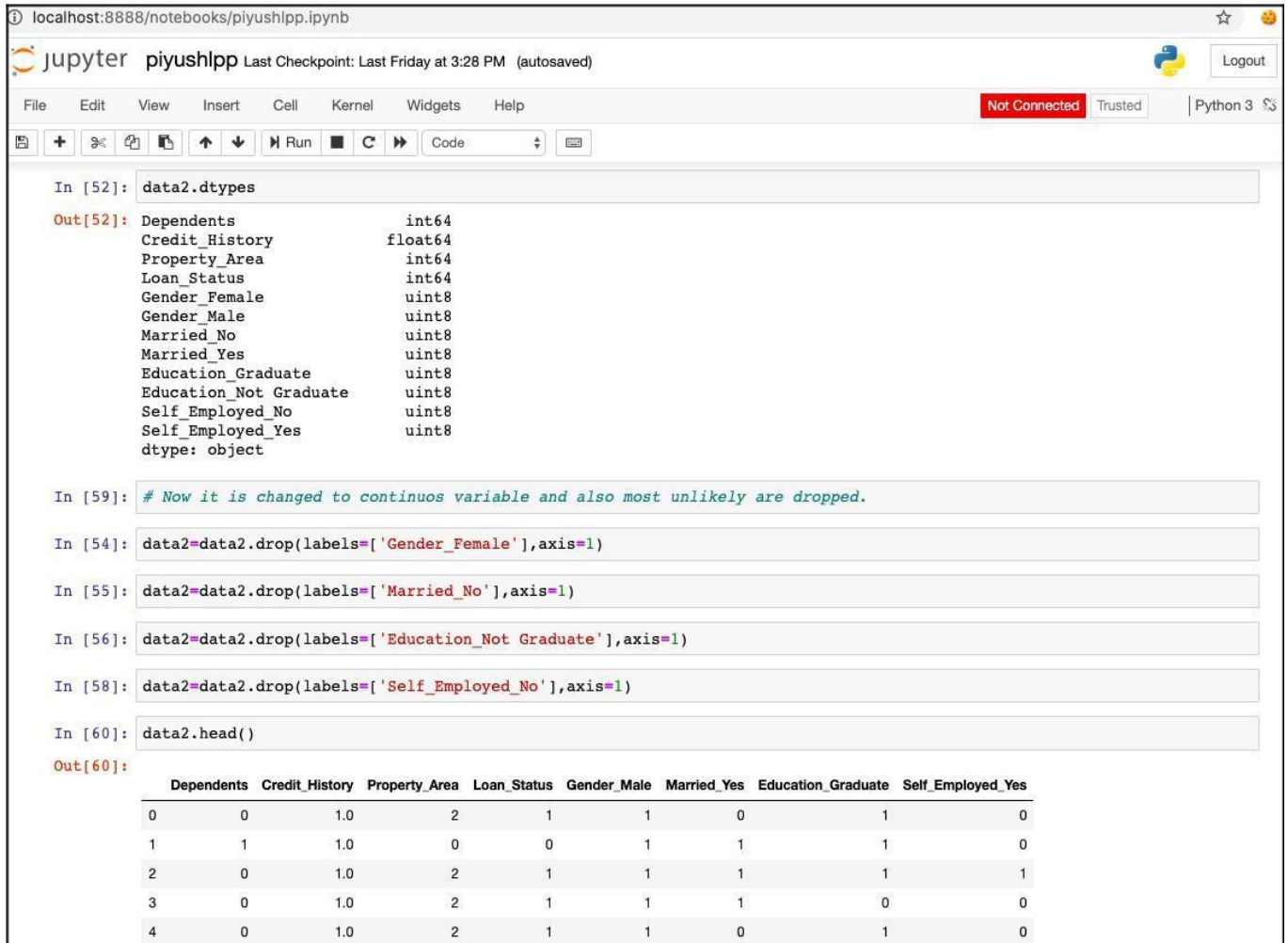
In [50]: `data2['Dependents']=le.fit_transform(data2['Dependents'])`

In [51]: `data2=pd.get_dummies(data2)`

Fig.14.Variable Dropping

Dropping the Variable and making continuous the dataset

Fig.15.Making Continuos Dataset Dropping Null Set



The screenshot shows a Jupyter Notebook interface with the following content:

```
In [52]: data2.dtypes
```

```
Out[52]: Dependents          int64
Credit_History      float64
Property_Area        int64
Loan_Status          int64
Gender_Female        uint8
Gender_Male          uint8
Married_No           uint8
Married_Yes          uint8
Education_Graduate   uint8
Education_Not Graduate uint8
Self_Employed_No     uint8
Self_Employed_Yes    uint8
dtype: object
```

```
In [59]: # Now it is changed to continuous variable and also most unlikely are dropped.
```

```
In [54]: data2=data2.drop(labels=['Gender_Female'],axis=1)
```

```
In [55]: data2=data2.drop(labels=['Married_No'],axis=1)
```

```
In [56]: data2=data2.drop(labels=['Education_Not Graduate'],axis=1)
```

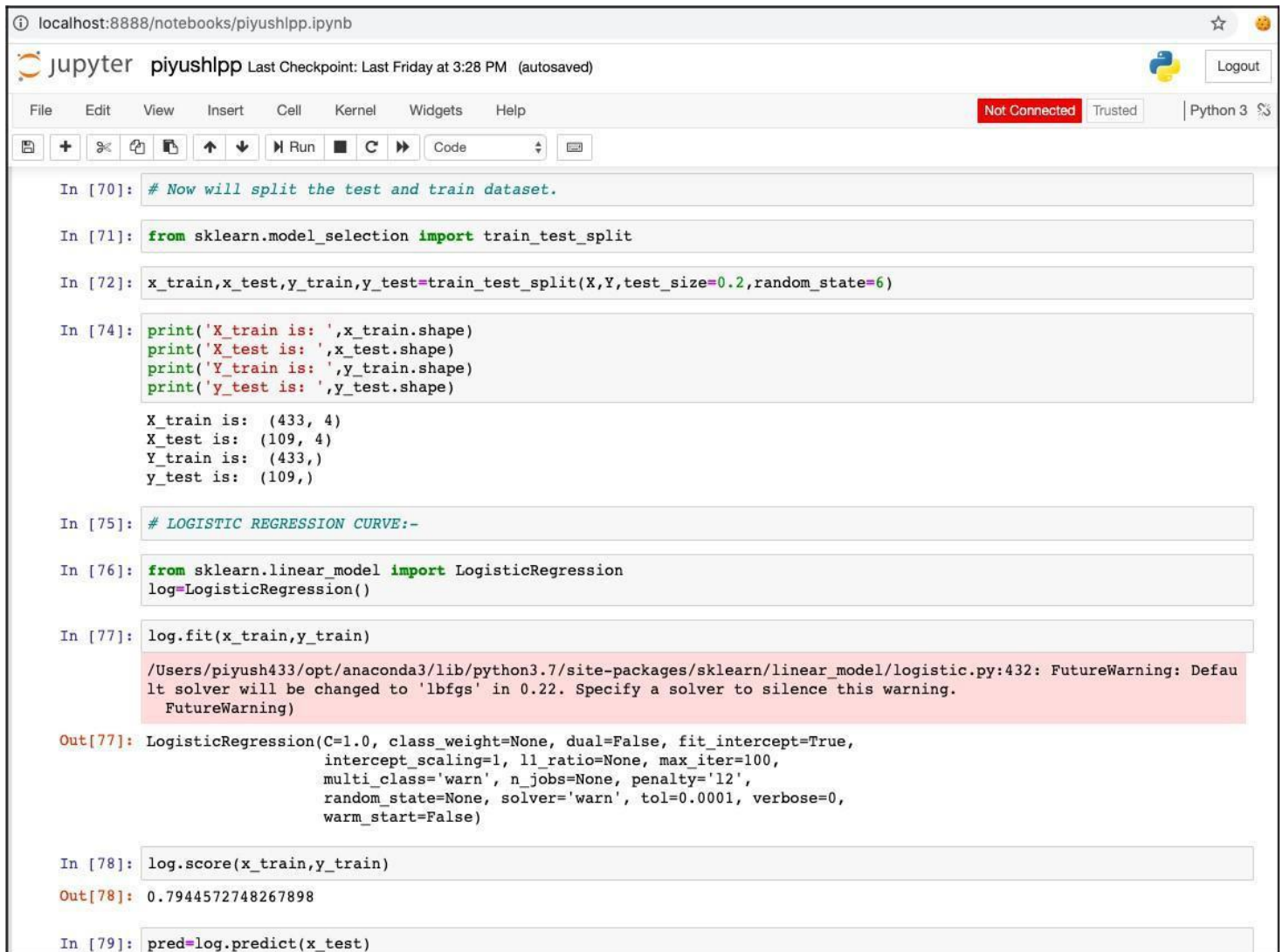
```
In [58]: data2=data2.drop(labels=['Self_Employed_No'],axis=1)
```

```
In [60]: data2.head()
```

```
Out[60]:
```

	Dependents	Credit_History	Property_Area	Loan_Status	Gender_Male	Married_Yes	Education_Graduate	Self_Employed_Yes
0	0	1.0	2	1	1	0	1	0
1	1	1.0	0	0	1	1	1	0
2	0	1.0	2	1	1	1	1	1
3	0	1.0	2	1	1	1	0	0
4	0	1.0	2	1	1	0	1	0

Dividing the test and trained dataset and seeing the output of it



The screenshot shows a Jupyter Notebook interface with the following content:

```
In [70]: # Now will split the test and train dataset.

In [71]: from sklearn.model_selection import train_test_split

In [72]: x_train,x_test,y_train,y_test=train_test_split(X,Y,test_size=0.2,random_state=6)

In [74]: print('X_train is: ',x_train.shape)
          print('X_test is: ',x_test.shape)
          print('Y_train is: ',y_train.shape)
          print('y_test is: ',y_test.shape)

          X_train is: (433, 4)
          X_test is: (109, 4)
          Y_train is: (433,)
          y_test is: (109,)

In [75]: # LOGISTIC REGRESSION CURVE:-

In [76]: from sklearn.linear_model import LogisticRegression
          log=LogisticRegression()

In [77]: log.fit(x_train,y_train)

/Users/piyush433/opt/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py:432: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)

Out[77]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                             intercept_scaling=1, l1_ratio=None, max_iter=100,
                             multi_class='warn', n_jobs=None, penalty='l2',
                             random_state=None, solver='warn', tol=0.0001, verbose=0,
                             warm_start=False)

In [78]: log.score(x_train,y_train)

Out[78]: 0.7944572748267898

In [79]: pred=log.predict(x_test)
```

Fig.16.Split test and Train Dataset

Logistic Regression Curve

Confusion Matrix is used for finding the probability.

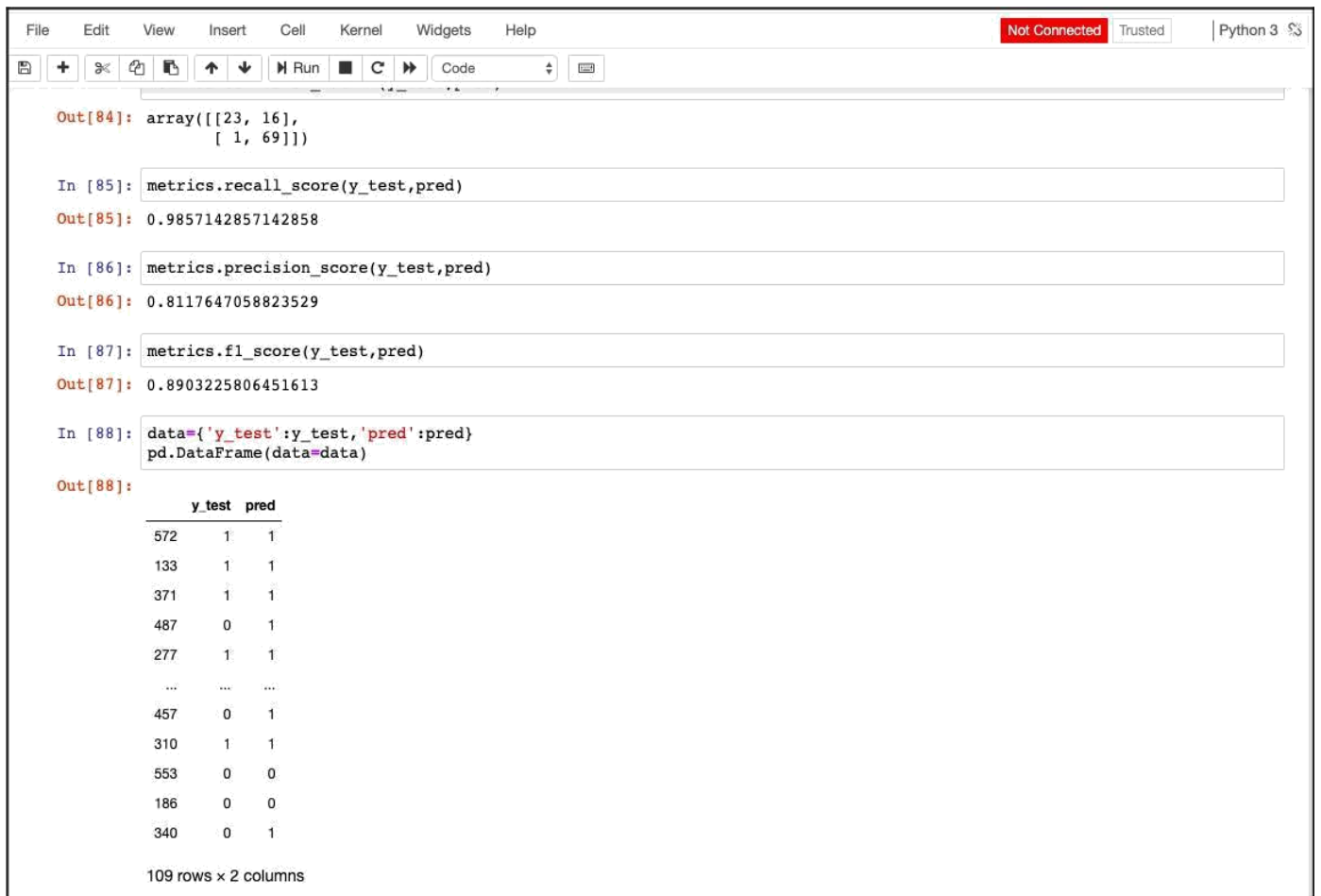
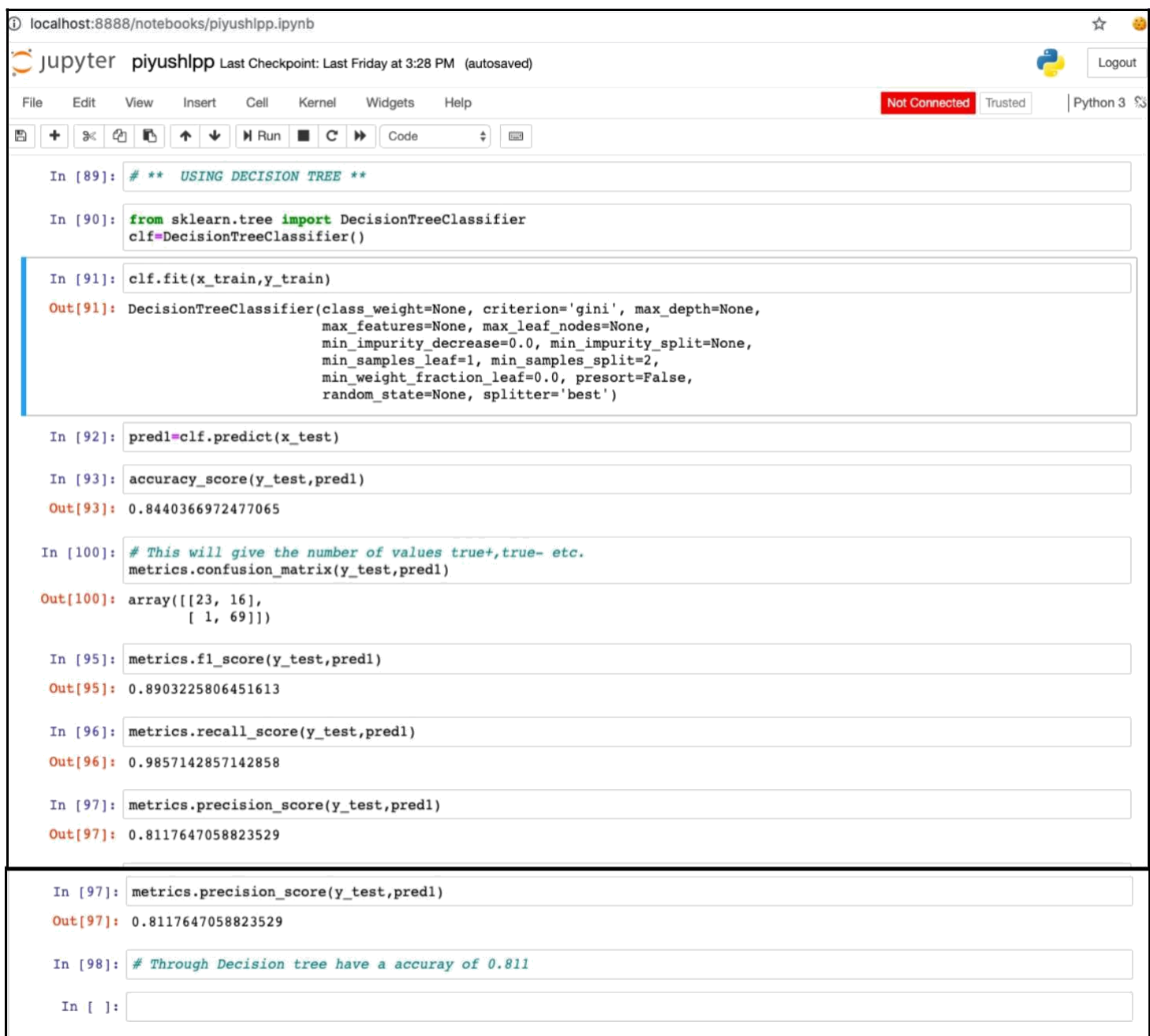


Fig.17.Confusion Matrix

Decision Tree

Using weighted variable for plotting the graph and finding the probability



```
In [89]: # ** USING DECISION TREE **

In [90]: from sklearn.tree import DecisionTreeClassifier
         clf=DecisionTreeClassifier()

In [91]: clf.fit(x_train,y_train)
Out[91]: DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
                                max_features=None, max_leaf_nodes=None,
                                min_impurity_decrease=0.0, min_impurity_split=None,
                                min_samples_leaf=1, min_samples_split=2,
                                min_weight_fraction_leaf=0.0, presort=False,
                                random_state=None, splitter='best')

In [92]: pred1=clf.predict(x_test)

In [93]: accuracy_score(y_test,pred1)
Out[93]: 0.8440366972477065

In [100]: # This will give the number of values true+,true- etc.
          metrics.confusion_matrix(y_test,pred1)
Out[100]: array([[23, 16],
                 [ 1, 69]])

In [95]: metrics.f1_score(y_test,pred1)
Out[95]: 0.8903225806451613

In [96]: metrics.recall_score(y_test,pred1)
Out[96]: 0.9857142857142858

In [97]: metrics.precision_score(y_test,pred1)
Out[97]: 0.8117647058823529

In [97]: metrics.precision_score(y_test,pred1)
Out[97]: 0.8117647058823529

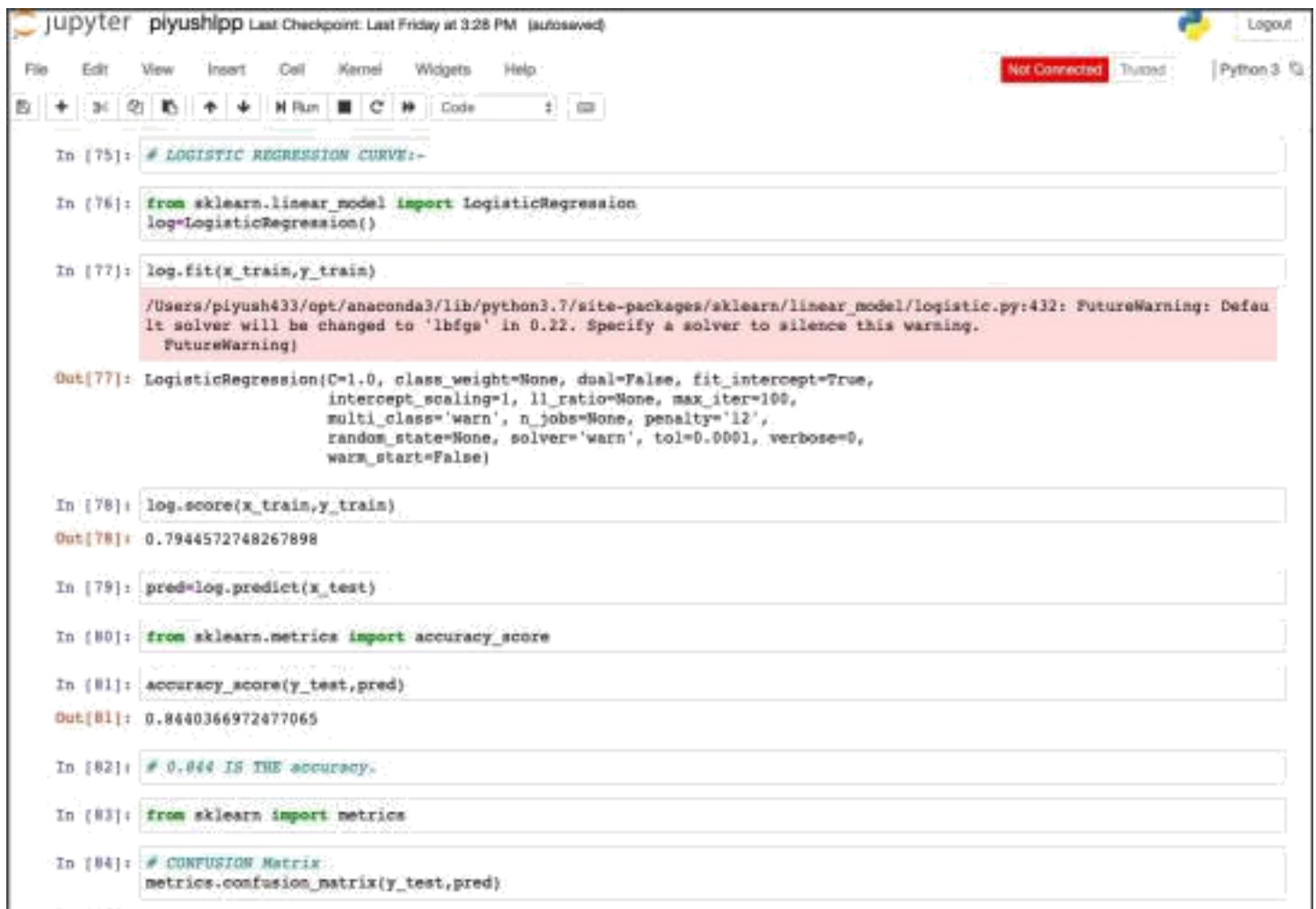
In [98]: # Through Decision tree have a accuray of 0.811

In [ ]:
```

Fig.18.Decision Tree

Chapter 5: Results

From Logistic Curve We get



The image shows a Jupyter Notebook interface with the username 'piyushipp' and a last checkpoint from 'Last Friday at 3:28 PM (autosaved)'. The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, running, and code execution. The notebook contains several code cells:

- In [75]:** `# LOGISTIC REGRESSION CURVE:-`
- In [76]:** `from sklearn.linear_model import LogisticRegression
log=LogisticRegression()`
- In [77]:** `log.fit(x_train,y_train)`
Output: `/Users/piyush433/opt/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py:432: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
FutureWarning]`
- Out[77]:** `LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True, intercept_scaling=1, l1_ratio=None, max_iter=100, multi_class='warn', n_jobs=None, penalty='l2', random_state=None, solver='warn', tol=0.0001, verbose=0, warm_start=False)`
- In [78]:** `log.score(x_train,y_train)`
Out[78]: `0.7944572748267898`
- In [79]:** `pred=log.predict(x_test)`
- In [80]:** `from sklearn.metrics import accuracy_score`
- In [81]:** `accuracy_score(y_test,pred)`
Out[81]: `0.8440366972477065`
- In [82]:** `# 0.844 IS THE accuracy.`
- In [83]:** `from sklearn import metrics`
- In [84]:** `# CONFUSION Matrix
metrics.confusion_matrix(y_test,pred)`

Fig.19.Logistic Regression Accuracy

Confusion Matrix

```
File Edit View Insert Cell Kernel Widgets Help Not Connected Trusted Python 3 3.8
Out[84]: array([[23, 16],
               [ 1, 69]])

In [85]: metrics.recall_score(y_test,pred)
Out[85]: 0.9857142857142858

In [86]: metrics.precision_score(y_test,pred)
Out[86]: 0.8117647058823529

In [87]: metrics.f1_score(y_test,pred)
Out[87]: 0.8903225806451613

In [88]: data={'y_test':y_test,'pred':pred}
pd.DataFrame(data=data)
Out[88]:
```

	y_test	pred
572	1	1
133	1	1
371	1	1
487	0	1
277	1	1
...
457	0	1
310	1	1
553	0	0
186	0	0
340	0	1

109 rows x 2 columns

Fig.20.Logistic Regression(Confusion Matrix)

We compared the training and testing dataset and also through confusion matrix we compared and get an accuracy of 0.84.

Using Decision Tree:-

```
localhost:8888/notebooks/piyushlpp.ipynb
jupyter piyushlpp Last Checkpoint: Last Friday at 3:28 PM (autosaved)
File Edit View Insert Cell Kernel Widgets Help Not Connected Trusted Python 3

In [89]: # ** USING DECISION TREE **

In [90]: from sklearn.tree import DecisionTreeClassifier
         clf=DecisionTreeClassifier()

In [91]: clf.fit(x_train,y_train)
Out[91]: DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
                                max_features=None, max_leaf_nodes=None,
                                min_impurity_decrease=0.0, min_impurity_split=None,
                                min_samples_leaf=1, min_samples_split=2,
                                min_weight_fraction_leaf=0.0, presort=False,
                                random_state=None, splitter='best')

In [92]: pred1=clf.predict(x_test)

In [93]: accuracy_score(y_test,pred1)
Out[93]: 0.8440366972477065

In [100]: # This will give the number of values true+,true- etc.
          metrics.confusion_matrix(y_test,pred1)
Out[100]: array([[23, 16],
                 [ 1, 69]])

In [95]: metrics.f1_score(y_test,pred1)
Out[95]: 0.8903225806451613

In [96]: metrics.recall_score(y_test,pred1)
Out[96]: 0.9857142857142858

In [97]: metrics.precision_score(y_test,pred1)
Out[97]: 0.8117647058823529

In [97]: metrics.precision_score(y_test,pred1)
Out[97]: 0.8117647058823529

In [98]: # Through Decision tree have a accuray of 0.811

In [ ]:
```

Fig.21.Decision Tree Accuracy

We used weighted variable to compare with the testing and training dataset and got an accuracy of 0.811.[3]

Chapter 6: Conclusion

From a proper analysis of positive points and constraints on the component, it can be safely concluded that the product is a highly efficient component. This application is working properly and meeting to all Banker requirements. This component can be easily plugged in many other systems.

There have been numbers cases of computer glitches, errors in content and most important weight of features is fixed in automated prediction system, So in the near future the so –called software could be made more secure, reliable and dynamic weight adjustment .In near future this module of prediction can be integrate with the module of automated processing system. the system is trained on old training dataset in future software can be made such that new testing data should also take part in training data after some fix time.

Chapter 7: References

[1]Machine Learning Algorithms: International Journal of Computer Science and Telecommunications (Volume2, Issue3, June 2011).

[2] J. R. Quinlan. *Induction of Decision Tree. Machine Learning, Vol. 1, No. 1. pp. 81-106., 1086.* [3] J.R. Quinlan. *Induction of decision trees. MachinelearningSpringer, 1(1):81 –106, 1086.*

[4] [kaggle.com](https://www.kaggle.com) for dataset.