K L University
(Koneru Lakshmaiah Education Foundation)
Deemed to be University, Estd. u/s 3 of UGC Act, 1956
Accredited by NAAC as 'A' Grade University ✧ Approved by AICTE ✧ ISO 9001-2008 Certified
Campus: Greenfields, Vaddeswarm -522502, Guntur District, Andhra Pradesh, INDIA.
Phone: +91-863-2399999 Fax: +91-863- 2388999
Admin Off: 29-36-38, Museum Road, Governorpet, Vijayawada - 520 002, Ph: +91-866-2577715, Fax: +91-866-2577717.

Department of Electronics and Communication Engineering

18TP3101
TP&T-1
MINOR PROJECT-I

A Project Based Lab Report
On
"LOAN PREDICTION ANALYSIS USING LOGISTIC REGRESSION"

SUBMITTED BY:
N.Satish
180040132
SEC NO:4

K L University
(Koneru Lakshmaiah Education Foundation)
Deemed to be University, Estd. u/s 3 of UGC Act, 1956
Accredited by NAAC as 'A' Grade University ✧ Approved by AICTE ✧ ISO 9001-2008 Certified
Campus: Greenfields, Vaddeswarm -522502, Guntur District, Andhra Pradesh, INDIA.
Phone: +91-863-2399999 Fax: +91-863- 2388999
Admin Off: 29-36-38, Museum Road, Governorpet, Vijayawada - 520 002, Ph: +91-866-2577715, Fax: +91-866-2577717.

Department of Electronics and Communication Engineering

BONAFIDE CERTIFICATE

This is to certify that the project-based laboratory Report Title "LOAN PREDICTION ANALYSIS " Submitted by: N.SAI KIRAN id number 180040096. In the Department of Electronics and Communication Engineering, KL University in partial fulfillment of the requirements for the completion of a project-based Laboratory in course in III B Tech V Semester, is a bonafide record of the work carried out by her under during the academic year 2020 – 2021.

PROJECT SUPERVISOR **HEAD OF THE DEPARTMENT**

Loan Prediction Data:

We have the loan application information like the applicant's name, personal details, financial information and requested loan amount and related details and the outcome (whether the application was approved or rejected). Based on this we are going to train a model and predict if a loan will get approved or not.

Problem:

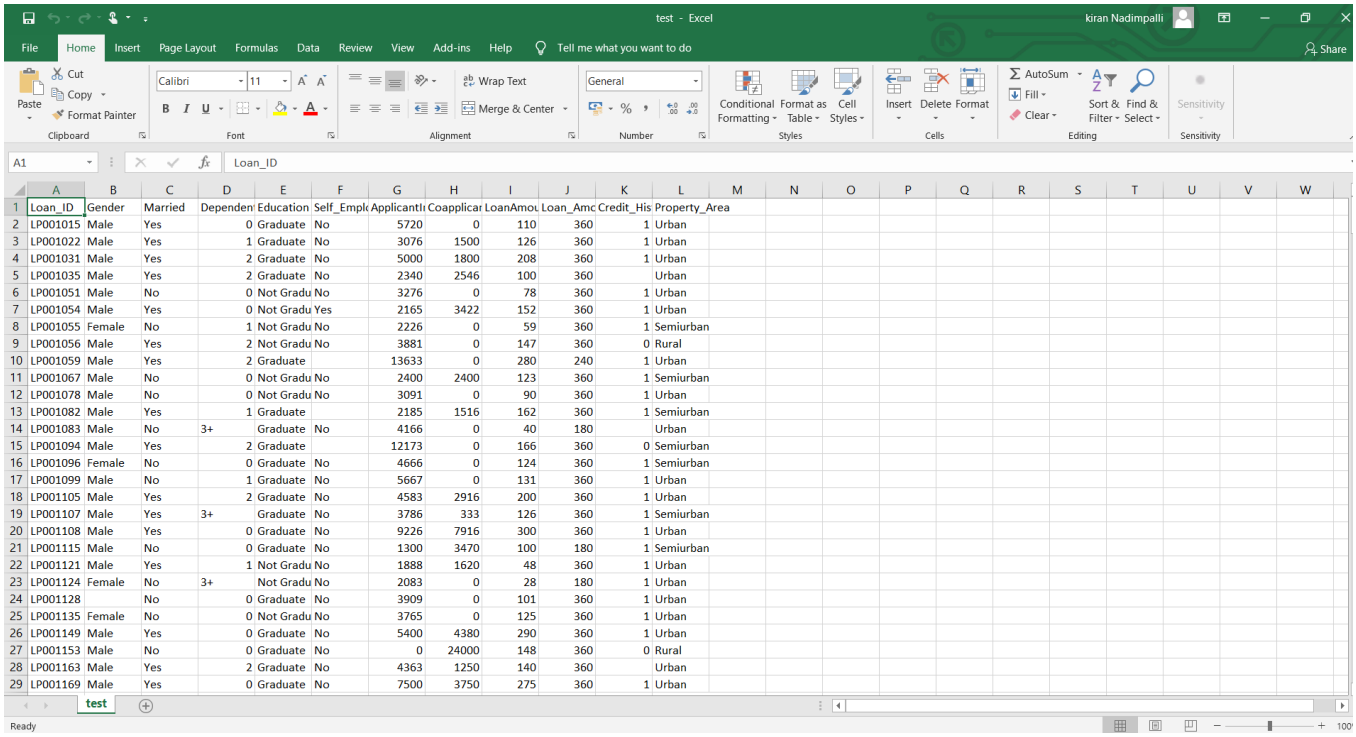
Problem

A Company wants to automate the loan eligibility process (real time) based on customer detail provided while filling online application form. These details are Gender, Marital Status, Education, Number of Dependents, Income, Loan Amount, Credit History and others. To automate this process, they have given a problem to identify the customers segments, those are eligible for loan amount so that they can specifically target these customers. Here they have provided a data set.

Loading The Loan Application Data:

1. Open command prompt and type “jupyter notebook”
2. Press enter.
3. Then jupyter window will be opened and you can load the data set using the command read_csv(“filename.csv”).
4. Then the data is loaded.

Data Set Used For Loan Prediction:



Loan_ID	Gender	Married	Dependen	Education	Self_Empl	ApplicantI	Coapplicar	LoanAmou	Loan_Amc	Credit_His	Property_Area
LP001015	Male	Yes	0	Graduate	No	5720	0	110	360	1	Urban
LP001022	Male	Yes	1	Graduate	No	3076	1500	126	360	1	Urban
LP001031	Male	Yes	2	Graduate	No	5000	1800	208	360	1	Urban
LP001035	Male	Yes	2	Graduate	No	2340	2546	100	360		Urban
LP001051	Male	No	0	Not Gradu	No	3276	0	78	360	1	Urban
LP001054	Male	Yes	0	Not Gradu	Yes	2165	3422	152	360	1	Urban
LP001055	Female	No	1	Not Gradu	No	2226	0	59	360	1	Semiurban
LP001056	Male	Yes	2	Not Gradu	No	3881	0	147	360	0	Rural
LP001059	Male	Yes	2	Graduate		13633	0	280	240	1	Urban
LP001067	Male	No	0	Not Gradu	No	2400	2400	123	360	1	Semiurban
LP001078	Male	No	0	Not Gradu	No	3091	0	90	360	1	Urban
LP001082	Male	Yes	1	Graduate		2185	1516	162	360	1	Semiurban
LP001083	Male	No	3+	Graduate	No	4166	0	40	180		Urban
LP001094	Male	Yes	2	Graduate		12173	0	166	360	0	Semiurban
LP001096	Female	No	0	Graduate	No	4666	0	124	360	1	Semiurban
LP001099	Male	No	1	Graduate	No	5667	0	131	360	1	Urban
LP001105	Male	Yes	2	Graduate	No	4583	2916	200	360	1	Urban
LP001107	Male	Yes	3+	Graduate	No	3786	333	126	360	1	Semiurban
LP001108	Male	Yes	0	Graduate	No	9226	7916	300	360	1	Urban
LP001115	Male	No	0	Graduate	No	1300	3470	100	180	1	Semiurban
LP001121	Male	Yes	1	Not Gradu	No	1888	1620	48	360	1	Urban
LP001124	Female	No	3+	Not Gradu	No	2083	0	28	180	1	Urban
LP001128		No	0	Graduate	No	3909	0	101	360	1	Urban
LP001135	Female	No	0	Not Gradu	No	3765	0	125	360	1	Urban
LP001149	Male	Yes	0	Graduate	No	5400	4380	290	360	1	Urban
LP001153	Male	No	0	Graduate	No	0	24000	148	360	0	Rural
LP001163	Male	Yes	2	Graduate	No	4363	1250	140	360		Urban
LP001169	Male	Yes	0	Graduate	No	7500	3750	275	360	1	Urban

Logistic Regression Basic Command:

```
model = LogisticRegression()
model.fit(x_train,y_train)
predictions = model.predict(x_test)
print(accuracy_score(y_test, predictions))
```

Code:-

```
import pandas as pd
import numpy as np
import seaborn as sns
import math
import matplotlib.pyplot as plt
% matplotlib inline
data=pd.read_csv("/content/sample_data/Loan Prediction.csv")
data.head(8)
data.isnull().sum()
data.Gender = data.Gender.fillna('Male')
data.Married = data.Married.fillna('Yes')
data.Self_Employed = data.Self_Employed.fillna('No')
data.Dependents = data.Dependents.fillna('0')
data.LoanAmount = data.LoanAmount.fillna(data.LoanAmount.mean())
data.Credit_History = data.Credit_History.fillna(1.0)
data.Loan_Amount_Term = data.Loan_Amount_Term.fillna(data.Loan_Amount_Term.me
an())
data.isnull().sum()
data.info()
data.shape
sns.countplot(x="ApplicantIncome" , data=data)
sns.countplot(x="Education" , data=data)
sns.countplot(x="Loan_Status" , hue = "Education" , data=data )
sns.countplot(x="Loan_Status" , hue = "Dependents" , data=data )
sns.countplot(x="Loan_Status" , hue = "Self_Employed" , data=data )
data["LoanAmount"].plot.hist()
# data.drop("Gender", axis=2 , inplace = true)  to remove certain cells
# data.dropna("inplact = true") to remove null cells

#To create dummy variables for categorical features .
Gender=pd.get_dummies(data['Gender'],drop_first=True)
#Changing the title .
Gender=Gender.rename(columns={'Male':'Gender'})
Gender.head()
Married=pd.get_dummies(data['Married'],drop_first=True)
Married.head()
Dependents=pd.get_dummies(data['Dependents'])
Dependents.head()
Education=pd.get_dummies(data['Education'],drop_first=True)
Education.head()
Self_Employed=pd.get_dummies(data['Self_Employed'],drop_first=True)
Self_Employed=Self_Employed.rename(columns={'Yes':'Self_Employed'})
Self_Employed.head()
Credit_History=pd.get_dummies(data['Credit_History'],drop_first=True)
Credit_History=Credit_History.rename(columns={1.0:'Credit_History'})
Credit_History.head()
Property_Area=pd.get_dummies(data['Property_Area'],drop_first=False)
```

```
Property_Area.head()
Loan_Status=pd.get_dummies(data['Loan_Status'],drop_first=True)
Loan_Status.head()
data.head()
#We are trying to concatinate the different coloumns to the single dataset .
datanew=pd.concat([data,Gender,Married,Dependents,Education,Self_Employed,Cre
dit_History,Loan_Status],axis=1)
#Checking that weather the dataset is updated or not .
datanew.head()
#Here we removed the unwanted coloumns (Parameters/inputs) .
datanew.drop(['Loan_ID','Gender','Married','Education','Dependents','Self_Emp
loyed','Property_Area','Loan_Status'],axis=1,inplace=True)
#Checking that weather the dataset is updated or not .
datanew.head()
#Changing the coloumn name .
datanew=datanew.rename(columns={'Y':'Loan_Status','Not Graduate':'Education',
'Yes':'Marry'})
#Checking that weather the dataset is updated or not .
datanew.head()
#Removing the target and assigning the remaining to X .
X=datanew.drop(["Loan_Status"],axis=1)
#Assigning the target to y
y=datanew["Loan_Status"]

#X=datanew.drop("Loan_Status",axis=1)
#y=datanew["Loan_Status"] # data need to predict
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, rand
om_state=20)
from sklearn.linear_model import LogisticRegression
logmodel=LogisticRegression()
#Fitting the model according to the given training data .
logmodel.fit(X_train,y_train)
pred=logmodel.predict(X_test)
from sklearn.metrics import classification_report
classification_report(y_test,pred)
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test,pred)
from sklearn.metrics import accuracy_score
accuracy_score(y_test,pred)
```

Outputs:

complete_loan_prediction.ipynb

File Edit View Insert Runtime Tools Help Last edited on August 31

```
import math
import matplotlib.pyplot as plt
% matplotlib inline
```

to run matplot lib we use matplotlib in line

```
data=pd.read_csv("/content/sample_data/Loan Prediction.csv")
data.head(8)
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Are
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0	Urba
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rur
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urba
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urba
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urba
5	LP001011	Male	Yes	2	Graduate	Yes	5417	4196.0	267.0	360.0	1.0	Urba
6	LP001013	Male	Yes	0	Not Graduate	No	2333	1516.0	95.0	360.0	1.0	Urba
7	LP001014	Male	Yes	3+	Graduate	No	3036	2504.0	158.0	360.0	0.0	Semiurba

complete_loan_prediction.ipynb

File Edit View Insert Runtime Tools Help Last edited on August 31

```
data.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
```

```
data.Gender = data.Gender.fillna('Male')
```

```
data.Married = data.Married.fillna('Yes')
data.Self_Employed = data.Self_Employed.fillna('No')
```

```
data.Dependents = data.Dependents.fillna('0')
data.LoanAmount = data.LoanAmount.fillna(data.LoanAmount.mean())
```

```
data.Credit_History = data.Credit_History.fillna(1.0)
data.Loan_Amount_Term = data.Loan_Amount_Term.fillna(data.Loan_Amount_Term.mean())
```

complete_loan_prediction.ipynb

File Edit View Insert Runtime Tools Help Last edited on August 31

```
data.Credit_History = data.Credit_History.fillna(1.0)
data.Loan_Amount_Term = data.Loan_Amount_Term.fillna(data.Loan_Amount_Term.mean())
```

```
data.isnull().sum()
```

```
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
Loan_Status          0
dtype: int64
```

```
data.info()
data.shape
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Loan_ID            614 non-null    object
```

complete_loan_prediction.ipynb

File Edit View Insert Runtime Tools Help Last edited on August 31

```
Loan_Status          0
dtype: int64
```


```
data.info()
data.shape
```


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

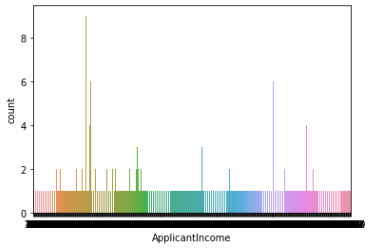
analysing of data // staticial graphs b/w different parameters to known the relation

complete_loan_prediction.ipynb

File Edit View Insert Runtime Tools Help Last edited on August 31

```
sns.countplot(x="ApplicantIncome" , data=data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9dc7c6b4a8>
```

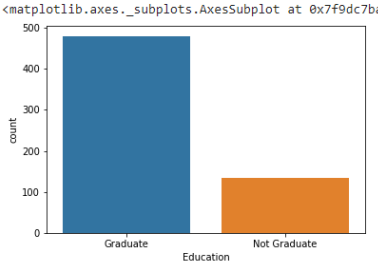


applicaant income and amount frequency

Double-click (or enter) to edit

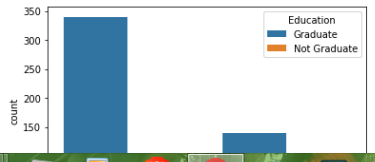
```
sns.countplot(x="Education" , data=data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9dc7ba90f0>
```




```
sns.countplot(x="Loan_Status" , hue = "Education" , data=data )
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9e0657af60>
```

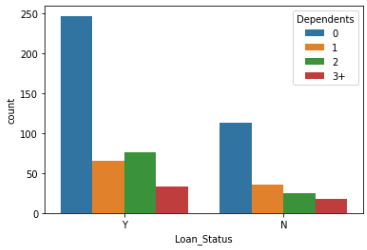


complete_loan_prediction.ipynb

File Edit View Insert Runtime Tools Help Last edited on August 31

```
sns.countplot(x="Loan_Status" , hue = "Dependents" , data=data )
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9dc775ab70>
```



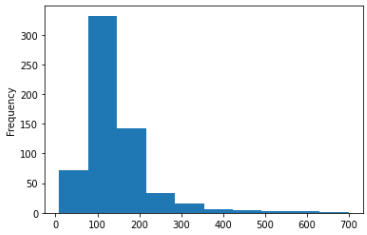
from this we can conclude no.of dependents are independent of loan approval

```
sns.countplot(x="Loan_Status" , hue = "Self_Employed" , data=data )
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9e065d5978>
```


```
data["LoanAmount"].plot.hist()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9e060238d0>
```



applicants vs frequency of income

```
# data.drop("Gender", axis=2 , inplace = true)  to remove certain cells
# data.dropna("inplact = true") to remove null cells
```

complete_loan_prediction.ipynb

File Edit View Insert Runtime Tools Help Last edited on August 31

```
# data.drop("Gender", axis=2 , inplace = true)  to remove certain cells
# data.dropna("inplact = true") to remove null cells
```

```
#To create dummy variables for categorical features .
Gender=pd.get_dummies(data['Gender'],drop_first=True)
#Changing the title .
Gender=Gender.rename(columns={'Male':'Gender'})
Gender.head()
```

	Gender
0	1
1	1
2	1
3	1
4	1

```
Married=pd.get_dummies(data['Married'],drop_first=True)
Married.head()
```

	Yes
0	0
1	1

complete_loan_prediction.ipynb

File Edit View Insert Runtime Tools Help Last edited on August 31

```
Dependents=pd.get_dummies(data['Dependents'])
Dependents.head()
```

	0	1	2	3+
0	1	0	0	0
1	0	1	0	0
2	1	0	0	0
3	1	0	0	0
4	1	0	0	0

```
Education=pd.get_dummies(data['Education'],drop_first=True)
Education.head()
```

	Not Graduate
0	0
1	0
2	0
3	1
4	0

complete_loan_prediction.ipynb

File Edit View Insert Runtime Tools Help Last edited on August 31

```
4    0
```

```
Self_Employed=pd.get_dummies(data['Self_Employed'],drop_first=True)
Self_Employed=Self_Employed.rename(columns={'Yes':'Self_Employed'})
Self_Employed.head()
```

	Self_Employed
0	0
1	0
2	1
3	0
4	0

```
Credit_History=pd.get_dummies(data['Credit_History'],drop_first=True)
Credit_History=Credit_History.rename(columns={1.0:'Credit_History'})
Credit_History.head()
```

	Credit_History
0	1
1	1

complete_loan_prediction.ipynb

File Edit View Insert Runtime Tools Help Last edited on August 31

```
Property_Area=pd.get_dummies(data['Property_Area'],drop_first=False)
Property_Area.head()
```

	Rural	Semiurban	Urban
0	0	0	1
1	1	0	0
2	0	0	1
3	0	0	1
4	0	0	1

```
Loan_Status=pd.get_dummies(data['Loan_Status'],drop_first=True)
Loan_Status.head()
```

	Y
0	1
1	0
2	1
3	1
4	1

complete_loan_prediction.ipynb

File Edit View Insert Runtime Tools Help Last edited on August 31

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

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Are
0	LP001002	Male	No	0	Graduate	No	5849	0.0	146.412162	360.0	1.0	Urba
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.000000	360.0	1.0	Rur
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.000000	360.0	1.0	Urba
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.000000	360.0	1.0	Urba
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.000000	360.0	1.0	Urba

```
[ ] #We are trying to concatinate the different coloumns to the single dataset .
datanew=pd.concat([data,Gender,Married,Dependents,Education,Self_Employed,Credit_History,Loan_Status],axis=1)
```

```
#Checking that weather the dataset is updated or not .
datanew.head()
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Are
0	LP001002	Male	No	0	Graduate	No	5849	0.0	146.412162	360.0	1.0	Urba

complete_loan_prediction.ipynb

File Edit View Insert Runtime Tools Help Last edited on August 31

```
[ ] #Here we removed the unwanted coloumns (Parameters/inputs) .
datanew.drop(['Loan_ID','Gender','Married','Education','Dependents','Self_Employed','Property_Area','Loan_Status'],axis=1,inplace=True)
```

```
#Checking that weather the dataset is updated or not .
datanew.head()
```

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Yes	0	1	2	3+	Not Graduate	Credit_History	Y
0	5849	0.0	146.412162	360.0	1.0	0	1	0	0	0	0	1	1
1	4583	1508.0	128.000000	360.0	1.0	1	0	1	0	0	0	1	0
2	3000	0.0	66.000000	360.0	1.0	1	1	0	0	0	0	1	1
3	2583	2358.0	120.000000	360.0	1.0	1	1	0	0	0	1	1	1
4	6000	0.0	141.000000	360.0	1.0	0	1	0	0	0	0	1	1

```
[ ] #Changing the coloumn name .
datanew=datanew.rename(columns={'Y':'Loan_Status','Not Graduate':'Education','Yes':'Marry'})
```

```
[ ] #Checking that weather the dataset is updated or not .
datanew.head()
```

complete_loan_prediction.ipynb

File Edit View Insert Runtime Tools Help Last edited on August 31

3	2583	2358.0	120.000000	360.0	1.0	1	1	0	0	0	1	1	1
4	6000	0.0	141.000000	360.0	1.0	0	1	0	0	0	0	1	1

```
#Changing the coloumn name .
datanew=datanew.rename(columns={'Y':'Loan_Status','Not Graduate':'Education','Yes':'Marry'})
```

```
#Checking that weather the dataset is updated or not .
datanew.head()
```

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Marry	0	1	2	3+	Education	Credit_History	Loan_Status
0	5849	0.0	146.412162	360.0	1.0	0	1	0	0	0	0	1	1
1	4583	1508.0	128.000000	360.0	1.0	1	0	1	0	0	0	1	0
2	3000	0.0	66.000000	360.0	1.0	1	1	0	0	0	0	1	1
3	2583	2358.0	120.000000	360.0	1.0	1	1	0	0	0	1	1	1
4	6000	0.0	141.000000	360.0	1.0	0	1	0	0	0	0	1	1

train data

```
#Removing the target and assigning the remaining to X .
X=datanew.drop(["Loan_Status"],axis=1)
#Assigning the target to y
```

complete_loan_prediction.ipynb

File Edit View Insert Runtime Tools Help Last edited on August 31

```
X=datanew.drop(["Loan_Status"],axis=1)
#Assigning the target to y
y=datanew["Loan_Status"]

#X=datanew.drop("Loan_Status",axis=1)
#y=datanew["Loan_Status"] # data need to predict
```

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=20)
```

```
from sklearn.linear_model import LogisticRegression
```

```
logmodel=LogisticRegression()
```

```
#Fitting the model according to the given training data .
logmodel.fit(X_train,y_train)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

```
pred=logmodel.predict(X_test)
```

complete_loan_prediction.ipynb

File Edit View Insert Runtime Tools Help Last edited on August 31

```
[ ] #Fitting the model according to the given training data .
    logmodel.fit(X_train,y_train)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

```
[ ] pred=logmodel.predict(X_test)
```

```
[ ] from sklearn.metrics import classification_report
```

```
[ ] classification_report(y_test,pred)
```

```
'              precision    recall  f1-score   support\n\n           0       0.93      0.41      0.57        69\n           1       0.74      0.98      0.84       116\n\n    accuracy                           0.77       185\n   macro avg       0.83      0.69      0.70       185\nweighted avg       0.81      0.77      0.74       185\n'
```

```
[ ] from sklearn.metrics import confusion_matrix
```

```
confusion_matrix(y_test,pred)
```

```
array([[ 28,  41],
       [  2, 114]])
```

complete_loan_prediction.ipynb

File Edit View Insert Runtime Tools Help Last edited on August 31

+ Code + Text

```
[ ] '              precision    recall  f1-score   support\n\n           0       0.93      0.41      0.57        69\n           1       0.74      0.98      0.84
    116\n\n    accuracy                           0.77       185\n   macro avg       0.83      0.69      0.70       185\nweighted avg       0.81      0.77      0.74
    185\n'
```

```
[ ] from sklearn.metrics import confusion_matrix
```

```
[ ] confusion_matrix(y_test,pred)
```

```
array([[ 28,  41],
       [  2, 114]])
```

```
[ ] from sklearn.metrics import accuracy_score
```

```
[ ] accuracy_score(y_test,pred)
```

```
0.7675675675675676
```