Machine Learning (using Python)

PREDICT LOAN APPLICATION STATUS

Done By :

Sk Samim Islam

Techno India Batanagar

Table of Contents

ACKNOWLEDGEMENT

I would like to express my special thanks of gratitude to my teacher **Professor Arnab Chakraborty** who gave me the golden opportunity to do this wonderful project on the topic **Predicting Loan Application Status**, which also helped me in doing a lot of Research and I came to know about so many new things I am really thankful to them.

I am obliged to my project team members for the valuable information provided by them in their respective fields. I am grateful for their cooperation during the period of my assignment.

SK SAMIM ISLAM

Project Objective

While there have been several delinquencies in the responds of loan applicants, The traditional methods of making loan decision is gradually fading out as predictive analytics (application scoring, behavioural scoring and collection scoring) are gradually replacing traditional way of scoring loan applicants.

In light of this, I want to demonstrate how to predict whether a loan applicant will be accepted or not using Machine Learning.

Project Scope

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 option 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.

For the analysis of high-dimensional and loan application data, machine learning offers a worthy approach for making classy and automatic algorithms. It brings attention towards the suite of machine learning algorithms and tools that are used for the analysis of diseases and decision-making process accordingly.

Dataset Design

- ## *Train.csv*

Attributes

- *Application_ID* : Contain the application ID of the applicant
- *Gender* : Contain the gender of the applicant
- *Dependents* : Contains the dependency of applicant
- *Education* : Contain the educational qualification of the applicant
- *Self_Employed* : Whether the applicant is self employed or not
- *ApplicantIncome* : Contain the applicant income
- *CoapplicantIncome* : Contain the coapplicant income
- *LoanAmount* : Contain the amount the applicant applied for
- *Loan_Amount_Term*: Contain the term of the loan amount
- *Credit_History* : Contain the credit history of the applicant
- *Property_Area* : Contain the information about the property area
- *Loan_Status* : Whether the loan is granted or not

Application_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area
1345	M	Yes	2	Not Gradu	No	4288	3263	133	180	1	Urban
1349	M	No	0	Graduate	No	4843	3806	151	360	1	Semiurban
1350	M	Yes		Graduate	No	13650	0		360	1	Urban
1356	M	Yes	0	Graduate	No	4652	3583		360	1	Semiurban
1357	M			Graduate	No	3816	754	160	360	1	Urban
1367	M	Yes	1	Graduate	No	3052	1030	100	360	1	Urban
1369	M	Yes	2	Graduate	No	11417	1126	225	360	1	Urban
1370	M	No	0	Not Graduate		7333	0	120	360	1	Rural
1379	M	Yes	2	Graduate	No	3800	3600	216	360	0	Urban
1384	M	Yes	3+	Not Gradu	No	2071	754	94	480	1	Semiurban
1385	M	No	0	Graduate	No	5316	0	136	360	1	Urban
1387	F	Yes	0	Graduate		2929	2333	139	360	1	Semiurban
1391	M	Yes	0	Not Gradu	No	3572	4114	152		0	Rural
1392	F	No	1	Graduate	Yes	7451	0		360	1	Semiurban
1398	M	No	0	Graduate		5050	0	118	360	1	Semiurban
1401	M	Yes	1	Graduate	No	14583	0	185	180	1	Rural
1404	F	Yes	0	Graduate	No	3167	2283	154	360	1	Semiurban
1405	M	Yes	1	Graduate	No	2214	1398	85	360		Urban
1421	M	Yes	0	Graduate	No	5568	2142	175	360	1	Rural
1422	F	No	0	Graduate	No	10408	0	259	360	1	Urban
1426	M	Yes		Graduate	No	5667	2667	180	360	1	Rural
1430	F	No	0	Graduate	No	4166	0	44	360	1	Semiurban
1431	F	No	0	Graduate	No	2137	8980	137	360	0	Semiurban
1432	M	Yes	2	Graduate	No	2957	0	81	360	1	Semiurban
1439	M	Yes	0	Not Gradu	No	4300	2014	194	360	1	Rural
1443	F	No	0	Graduate	No	3692	0	93	360		Rural
1448		Yes	3+	Graduate	No	23803	0	370	360	1	Rural
1449	M	No	0	Graduate	No	3865	1640		360	1	Rural
1451	M	Yes	1	Graduate	Yes	10513	3850	160	180	0	Urban
1465	M	Yes	0	Graduate	No	6080	2569	182	360		Rural

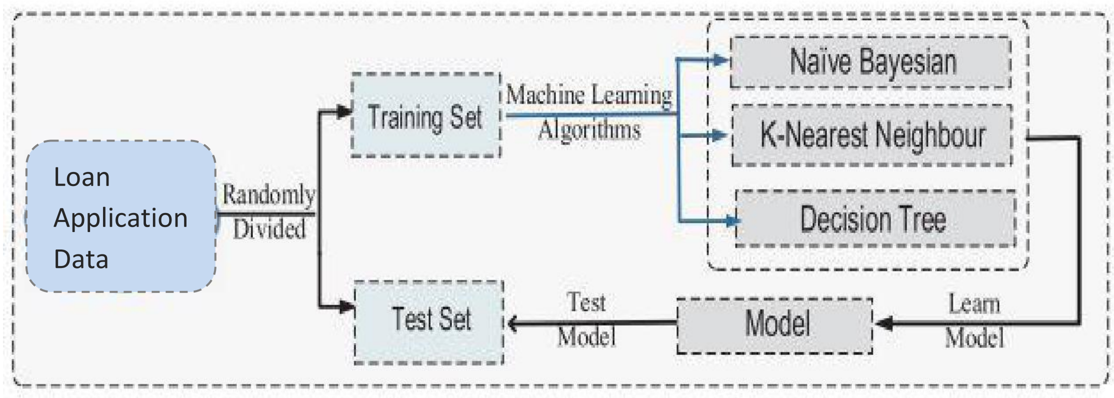
➢ *Test.csv*

Attributes

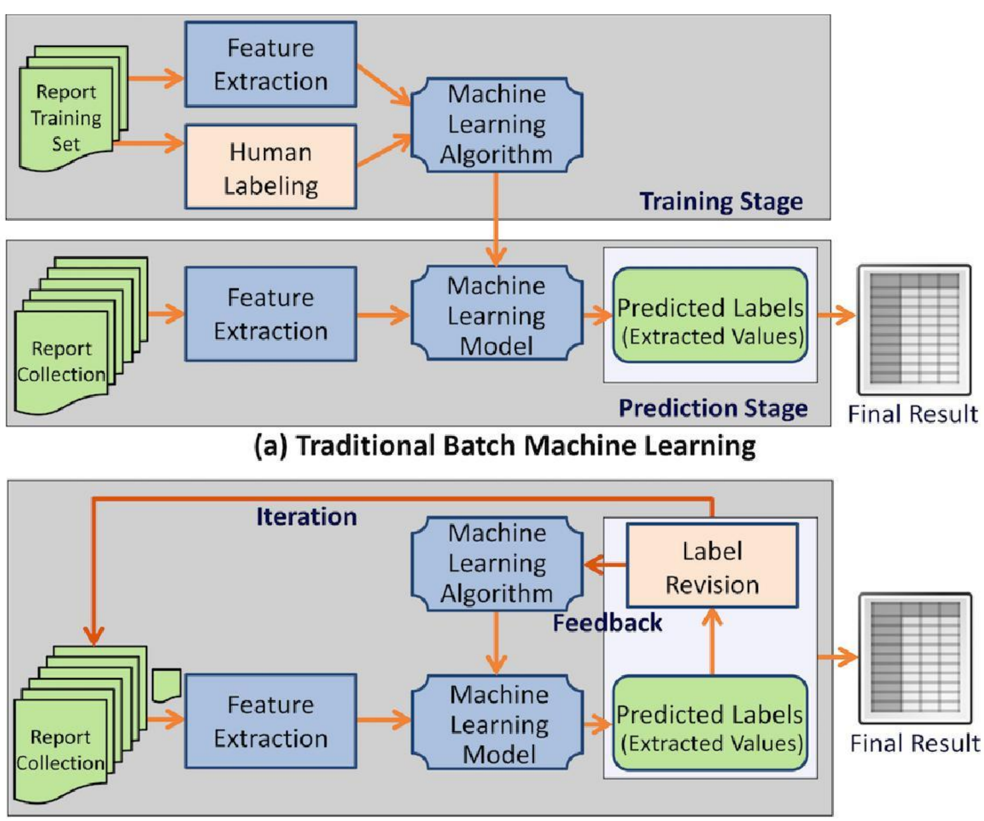
- *Application_ID* : Contain the application ID of the applicant
- *Gender* : Contain the gender of the applicant
- *Dependents* : Contains the dependency of applicant
- *Education* : Contain the educational qualification of the applicant
- *Self_Employed* : Whether the applicant is self employed or not
- *ApplicantIncome* : Contain the applicant income
- *CoapplicantIncome* : Contain the coapplicant income
- *LoanAmount* : Contain the amount the applicant applied for
- *Loan_Amount_Term*: Contain the term of the loan amount
- *Credit_History* : Contain the credit history of the applicant
- *Property_Area* : Contain the information about the property area

Applicatio	Gender	Married	Dependen	Education	Self_Empl	ApplicantI	Coapplicar	LoanAmou	Loan_Amc	Credit_His	Property_	Loan_Status
1002	M	No	0	Graduate	No	5849	0		360	1	Urban	Y
1003	M	Yes	1	Graduate	No	4583	1508	128	360	1	Rural	N
1005	M	Yes	0	Graduate	Yes	3000	0	66	360	1	Urban	Y
1006	M	Yes	0	Not Gradu	No	2583	2358	120	360	1	Urban	Y
1008	M	No	0	Graduate	No	6000	0	141	360	1	Urban	Y
1011	M	Yes	2	Graduate	Yes	5417	4196	267	360	1	Urban	Y
1013	M	Yes	0	Not Gradu	No	2333	1516	95	360	1	Urban	Y
1014	M	Yes	3+	Graduate	No	3036	2504	158	360	0	Semiurban	N
1018	M	Yes	2	Graduate	No	4006	1526	168	360	1	Urban	Y
1020	M	Yes	1	Graduate	No	12841	10968	349	360	1	Semiurban	N
1024	M	Yes	2	Graduate	No	3200	700	70	360	1	Urban	Y
1027	M	Yes	2	Graduate		2500	1840	109	360	1	Urban	Y
1028	M	Yes	2	Graduate	No	3073	8106	200	360	1	Urban	Y
1029	M	No	0	Graduate	No	1853	2840	114	360	1	Rural	N
1030	M	Yes	2	Graduate	No	1299	1086	17	120	1	Urban	Y
1032	M	No	0	Graduate	No	4950	0	125	360	1	Urban	Y
1034	M	No	1	Not Gradu	No	3596	0	100	240		Urban	Y
1036	F	No	0	Graduate	No	3510	0	76	360	0	Urban	N
1038	M	Yes	0	Not Gradu	No	4887	0	133	360	1	Rural	N
1041	M	Yes	0	Graduate		2600	3500	115		1	Urban	Y
1043	M	Yes	0	Not Gradu	No	7660	0	104	360	0	Urban	N
1046	M	Yes	1	Graduate	No	5955	5625	315	360	1	Urban	Y
1047	M	Yes	0	Not Gradu	No	2600	1911	116	360	0	Semiurban	N
1050		Yes	2	Not Gradu	No	3365	1917	112	360	0	Rural	N
1052	M	Yes	1	Graduate		3717	2925	151	360		Semiurban	N
1066	M	Yes	0	Graduate	Yes	9560	0	191	360	1	Semiurban	Y
1068	M	Yes	0	Graduate	No	2799	2253	122	360	1	Semiurban	Y
1073	M	Yes	2	Not Gradu	No	4226	1040	110	360	1	Urban	Y

Application Work Flow



The Three Machine Learning Algorithms Used in Predicting Loan Application Status

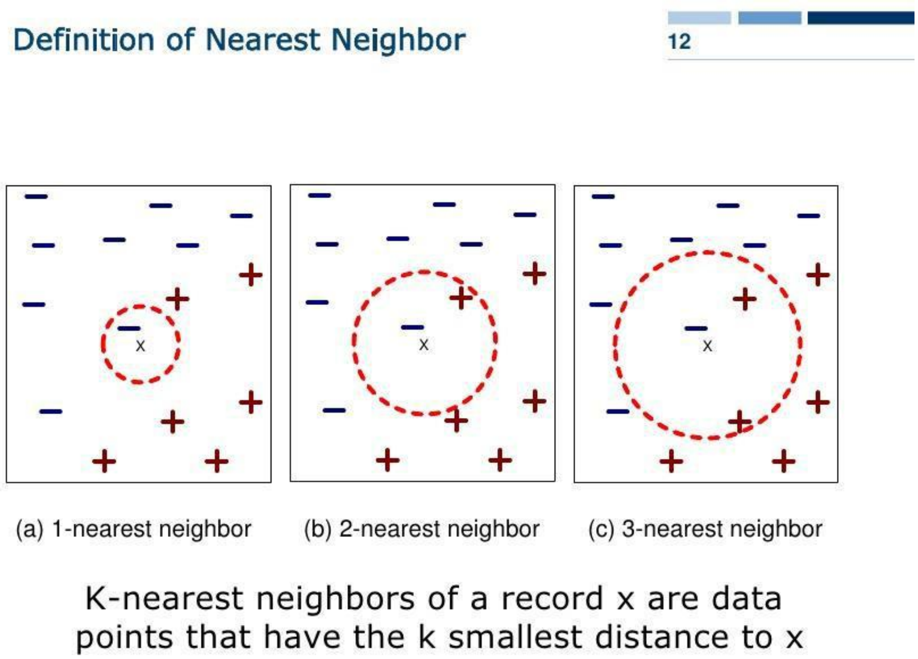


(a) Traditional Batch Machine Learning

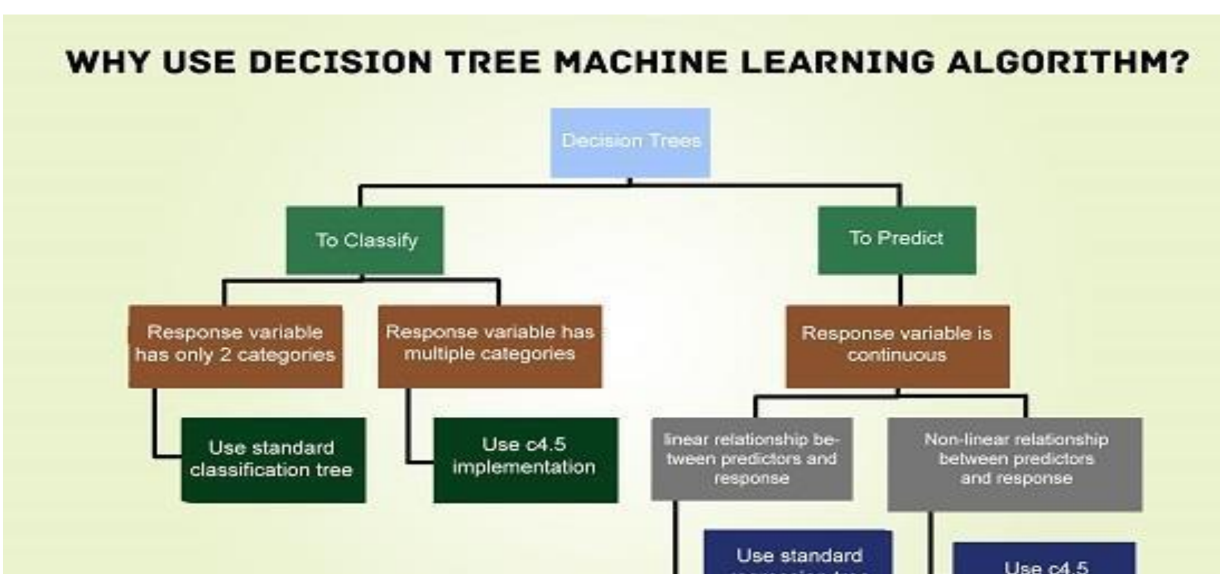
(b) Online Machine Learning

Different Modules

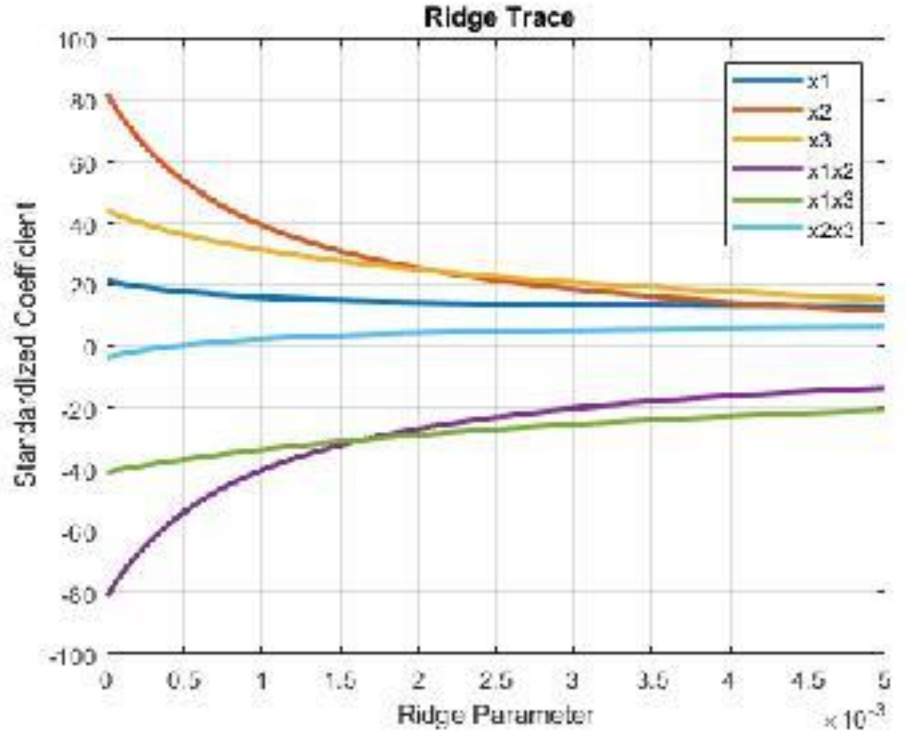
➢ KNN(K-NEAREST NEIGHBOUR)



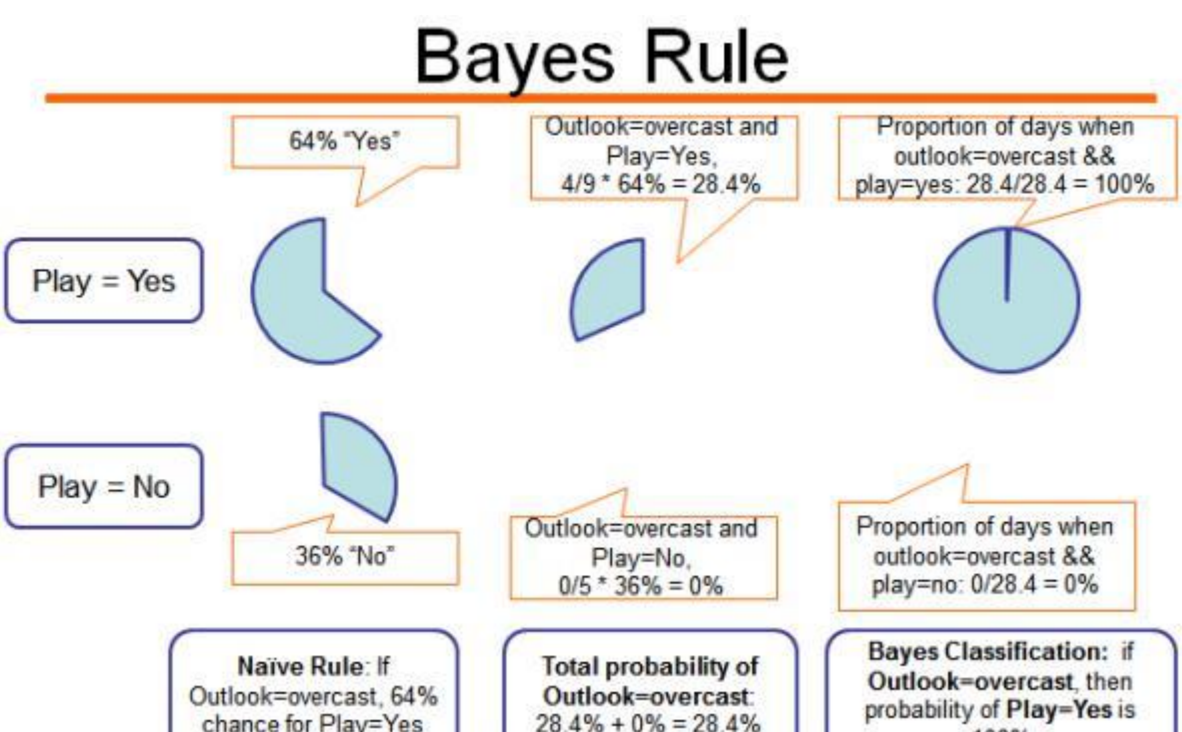
➢ DECISION TREE



LINEAR REGREESION(RIDGE)



NAÏVE BAYES



Screenshots

In [2]:
```
import pandas as pd
from sklearn import svm
from sklearn import cross_validation
from sklearn.linear_model import LinearRegression
from sklearn.feature_selection import SelectKBest, f_classif
import numpy as np
import matplotlib.pyplot as plt
```

In [3]:
```
loan = pd.read_csv('d:/Train.csv')
loan_test = pd.read_csv('d:/Test.csv')
loan.describe()
```

Out[3]:

	Application_ID	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
count	100.000000	100.00000	100.000000	95.000000	95.000000	92.000000
mean	1160.470000	4122.83000	1700.550000	134.221053	341.684211	0.836957
std	104.622212	2258.89434	1947.668891	63.456163	61.309342	0.371429
min	1002.000000	1000.00000	0.000000	17.000000	60.000000	0.000000
25%	1062.500000	2636.00000	0.000000	99.500000	360.000000	1.000000
50%	1153.000000	3598.00000	1558.500000	120.000000	360.000000	1.000000
75%	1253.500000	4710.00000	2394.500000	154.500000	360.000000	1.000000
max	1343.000000	12841.00000	10968.000000	349.000000	480.000000	1.000000

In [4]:
```
loan.count()
```

Out[4]:
```
Application_ID       100
Gender                99
Married              100
Dependents           100
Education            100
Self_Employed         94
ApplicantIncome      100
CoapplicantIncome    100
LoanAmount            95
Loan_Amount_Term      95
Credit_History        92
Property_Area        100
Loan_Status          100
```

In [5]:
```
loan.loc[loan['Education'] == 'Graduate', 'Education'] = 1
loan.loc[loan['Education'] == 'Not Graduate', 'Education'] = 0
loan.loc[loan['Property_Area'] == 'Rural', 'Property_Area'] = 0
loan.loc[loan['Property_Area'] == 'Semiurban', 'Property_Area'] = 1
loan.loc[loan['Property_Area'] == 'Urban', 'Property_Area'] = 2
loan2 = loan
loan3 = loan
loan4 = loan
```

In [6]:
```
predictors = ['Education', 'ApplicantIncome', 'CoapplicantIncome', 'Property_Area']
model = svm.SVC(probability=True)
scores = cross_validation.cross_val_score(model, loan[predictors], loan['Loan_Status'], cv=12)
print(scores)
print(scores.mean())
```

```
[0.66666667 0.66666667 0.66666667 0.66666667 0.625      0.625
 0.625      0.625      0.625      0.625      0.625      0.625     ]
0.6388888888888888
```

In [7]:
```
loan3['LoanAmount'] = loan['LoanAmount'].fillna(loan['LoanAmount'].mean())
loan3['Loan_Amount_Term'] = loan['Loan_Amount_Term'].fillna(loan['Loan_Amount_Term'].mean())
#print(loan['Credit_History'].mode())
#loan3['Credit_History'] = loan['Credit_History'].fillna(loan['Credit_History'].mode())
loan3['Credit_History'] = loan3['Credit_History'].fillna(loan3['Credit_History'].median())
```

In [8]:
```
predictors = ['Credit_History']
```

In [9]:
```
model = svm.SVC(probability=True)
scores = cross_validation.cross_val_score(model, loan3[predictors], loan3['Loan_Status'], cv=3)
print(scores)
print(scores.mean())
model.fit(loan3[predictors], loan3['Loan_Status'])
```

```
[0.79411765 0.78787879 0.78787879]
0.7899584076054665
```

Out[9]:
```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
  max_iter=-1, probability=True, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
```

In [10]:
```
loan_test.describe()
```

Out[10]:

	Application_ID	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
count	514.000000	514.000000	514.000000	497.000000	505.000000	472.000000
mean	2163.075875	5652.608949	1605.816965	148.742455	342.059406	0.843220
std	467.056621	6574.855143	3081.978215	89.057125	65.870517	0.363979
min	1345.000000	150.000000	0.000000	9.000000	12.000000	0.000000
25%	1760.250000	2894.250000	0.000000	100.000000	360.000000	1.000000
50%	2150.000000	3862.000000	1031.000000	128.000000	360.000000	1.000000
75%	2543.750000	6000.000000	2229.750000	170.000000	360.000000	1.000000
max	2990.000000	81000.000000	41667.000000	700.000000	480.000000	1.000000

```
In [11]: print(loan_test.count())
loan_test.loc[loan_test['Education'] == 'Graduate', 'Education'] = 1
loan_test.loc[loan_test['Education'] == 'Not Graduate', 'Education'] = 0
loan_test.loc[loan_test['Property_Area'] == 'Rural', 'Property_Area'] = 0
loan_test.loc[loan_test['Property_Area'] == 'Semiurban', 'Property_Area'] = 1
loan_test.loc[loan_test['Property_Area'] == 'Urban', 'Property_Area'] = 2
```

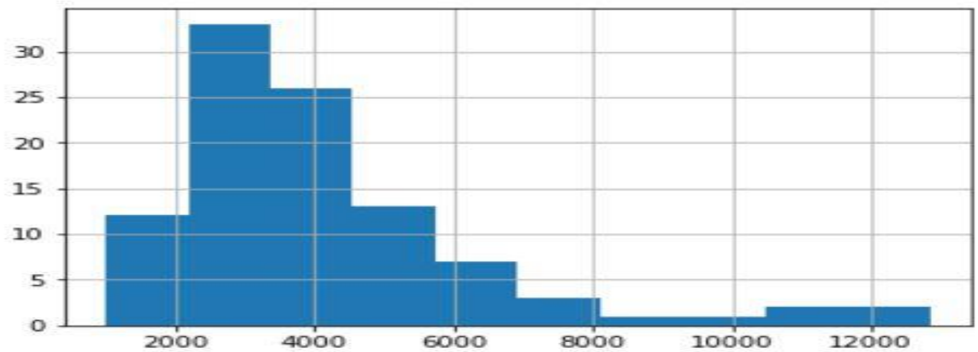
```
Application_ID        514
Gender                502
Married               511
Dependents            499
Education             514
Self_Employed         488
ApplicantIncome       514
CoapplicantIncome     514
LoanAmount            497
Loan_Amount_Term      505
Credit_History        472
Property_Area         514
dtype: int64
```

```
In [12]: loan_test2 = loan_test
loan_test2['Credit_History'] = loan_test['Credit_History'].fillna(loan_test['Credit_History'].mode())
loan_test2 = loan_test2[np.logical_not(np.isnan(loan_test2.Credit_History))]
loan_test2 = loan_test2.reset_index(drop=True)
print(loan_test2[predictors])
sattu = (model.predict(loan_test2[predictors]))
print(sattu)
```

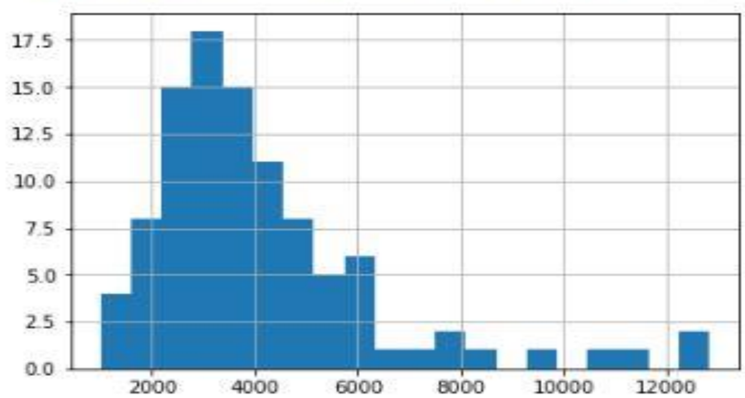
```
     Credit_History
0               1.0
1               1.0
2               1.0
3               1.0
4               1.0
5               1.0
6               1.0
23              1.0
24              1.0
25              1.0
26              0.0
27              1.0
28              1.0
29              1.0
..              ...
442             1.0
443             0.0
444             1.0
445             1.0
446             1.0
447             1.0
448             0.0
449             1.0
450             1.0
451             1.0
452             1.0
453             1.0
454             1.0
455             1.0
456             0.0
457             1.0
```

```
458          1.0
459          1.0
460          1.0
461          1.0
462          1.0
463          1.0
464          1.0
465          1.0
466          1.0
467          1.0
468          1.0
469          1.0
470          1.0
471          0.0

[472 rows x 1 columns]
['Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y'
 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y'
 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y'
 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y'
 'N' 'Y' 'N' 'N' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y'
 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y'
 'Y' 'Y' 'N' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y'
 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'N'
 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y'
 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'N'
 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y'
 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y'
 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'N'
 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y'
 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y'
 'Y' 'Y' 'N' 'Y' 'Y' 'N' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'N' 'Y' 'N'
 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y'
 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'N'
 'N' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'N' 'Y' 'Y' 'Y' 'Y'
 'N' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y'
 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y'
 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y'
 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'N' 'Y'
 'Y' 'N' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'N' 'Y' 'N' 'Y' 'N'
 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'N' 'Y'
 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y'
 'Y' 'Y' 'Y' 'N']
```

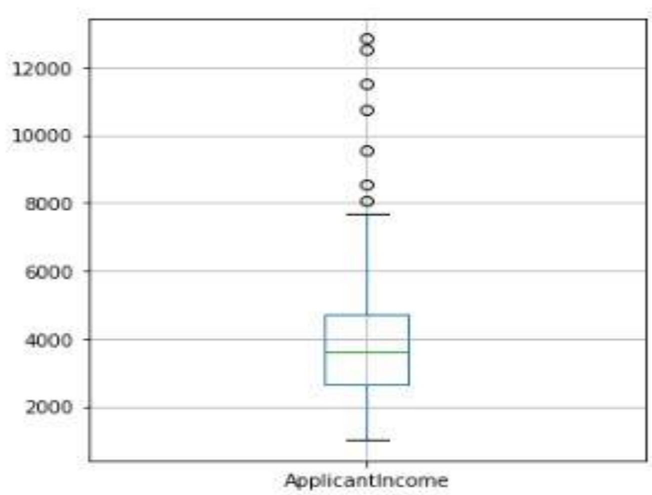
```
In [13]:  #For Non Numeric counts.
          loan['Property_Area'].value_counts()

Out[13]:  2    50
          1    41
          0     9
          Name: Property_Area, dtype: int64

In [14]:  loan['ApplicantIncome'].hist()

Out[14]:  <matplotlib.axes._subplots.AxesSubplot at 0x849e3b0>
```



```
In [15]:  loan['ApplicantIncome'].hist(bins=20)

Out[15]:  <matplotlib.axes._subplots.AxesSubplot at 0x882ca50>
```



```
In [16]:  loan.boxplot(column ='ApplicantIncome', figsize=(5,5), grid=True)

Out[16]:  <matplotlib.axes._subplots.AxesSubplot at 0x8893b30>
```



```
In [17]: # NATIVE BAYES.....!!!!
```

```
In [20]: try:
             mydata = pd.read_csv("d:/Train.csv")
         except:
             print("File open Error....!!!!")
         df = mydata
         df
         data = pd.DataFrame()
```

```
In [21]: df.Property_Area.unique()

Out[21]: array(['Urban', 'Rural', 'Semiurban'], dtype=object)
```

```
In [22]: # number of Yes
         print("No of people :-")
         n = df["Loan_Status"][df["Loan_Status"] == "Y"].count()
         print("Yes = ",n)

         # number of No
         n = df["Loan_Status"][df["Loan_Status"] == "N"].count()
         print("Rural = ",n)

         No of people :-
         Yes =  64
         Rural =  36
```

```
In [23]: # number of Graduate
         print("No of people :-\n")
         n = df["Education"][df["Education"] == "Graduate"].count()
         print("Graduate = ",n)
         # number of Not Graduate
         n = df["Education"][df["Education"] == "Not Graduate"].count()
         print("Not Graduate = ",n)

         n = df["Married"][df["Married"] == "Yes"].count()
         print("\nMarried = ",n)

         n = df["Married"][df["Married"] == "No"].count()
         print("Un Married = ",n)

         n = df["Gender"][df["Gender"] == "M"].count()
         print("\nMale = ",n)

         n = df["Gender"][df["Gender"] == "F"].count()
         print("Female = ",n)

         No of people :-

         Graduate =  77
         Not Graduate =  23

         Married =  69
         Un Married =  31

         Male =  84
         Female =  15
```

In [24]:
```
# calculate likelywood
# group the data by gender and calculate the of means of each feature
data_means = df.groupby("Loan_Status").mean()

# view the values
data_means

# MEAN
# print("Mean in terms of loan w.r.t Property area::")
# rural_mean = df['LoanAmount'].groupby(df['Rural']).mean()
```

Out[24]:

Loan_Status	Application_ID	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
N	1149.611111	4481.444444	1780.916667	148.558824	351.176471	0.53125
Y	1166.578125	3921.109375	1655.343750	126.229508	336.393443	1.00000

In [25]:
```
# group the data by gender and calculate the varieince of each feature
data_variance = df.groupby("Loan_Status").var()

# view the values
data_variance
```

Out[25]:

Loan_Status	Application_ID	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
N	8887.158730	7.769226e+06	4.908219e+06	5088.375223	2428.877005	0.257056
Y	12157.962054	3.587315e+06	3.228524e+06	3328.446448	4473.442623	0.000000

In [26]:
```
# MEANS wrt Property_Area
print("MEAN :-")

yes_mean = data_means["LoanAmount"][data_means.index == "Y"].values[0]
print("Yes = ",yes_mean)

no_mean = data_means["LoanAmount"][data_means.index == "N"].values[0]
print("No = ",no_mean)
```

```
MEAN :-
Yes =  126.22950819672131
No =  148.55882352941177
```

In [27]:
```
# VARIENCE wrt. Yes / No
print("VARIANCE :-")
yes_variance = data_variance["LoanAmount"][data_variance.index == "Y"].values[0]
print("Yes = ",yes_variance)

no_variance = data_variance["LoanAmount"][data_variance.index == "N"].values[0]
print("No = ",no_variance)
```

```
VARIANCE :-
Yes =  3328.446448087431
No =  5088.375222816398
```

In [28]:
```
# calculation of prior for both the classes Y / N
n_yes = df["Loan_Status"][df["Loan_Status"] == "Y"].count()
n_no = df["Loan_Status"][df["Loan_Status"] == "N"].count()
total_yn =n_yes + n_no

# number of rural divided by total no. people
p_yes = n_yes/total_yn

# number of urban divided by total people
p_no = n_no/total_yn

print("Probability of :-\n")
print("P(\"Yes\") = {} \nP(\"No\") = {}".format(p_yes,p_no))
```

```
Probability of :-

P("Yes") = 0.64
P("No") = 0.36
```

In [29]:
```
# Create a function  that calculates p(x/y)
import numpy as np
def p_x_given(x,mean_y,variance_y):
    # input the arguments into a probability density function
    p = 1/(np.sqrt(2 * np.pi * variance_y)) * np.exp((-(x - mean_y) ** 2)/(2 * variance_y))

    return p
```

In [30]:
```
# numerator of the posterior if the unclassified observation is a YES
num_posterior_yes = p_yes * \
p_x_given(loan_test["CoapplicantIncome"][6],yes_mean,yes_variance) * \
p_x_given(loan_test["LoanAmount"][6],yes_mean,yes_variance) *\
p_x_given(loan_test["Loan_Amount_Term"][6],yes_mean,yes_variance)

print("num_posterior_yes = ",num_posterior_yes)
```

```
num_posterior_yes =  8.202583074537033e-77
```

In [31]:
```
# numerator of the posterior if the unclassified observation is a NO
num_posterior_no = p_no * \
p_x_given(loan_test["CoapplicantIncome"][6],no_mean,no_variance) * \
p_x_given(loan_test["LoanAmount"][6],no_mean,no_variance) *\
p_x_given(loan_test["Loan_Amount_Term"][6],no_mean,no_variance)

print("num_posterior_no = ",num_posterior_no)
```

```
num_posterior_no =  7.420456200592003e-51
```

In [32]:
```
if (num_posterior_yes > num_posterior_no):
    print("Loan Approved")
elif (num_posterior_no > num_posterior_yes):
    print("Loan Rejected")
else:
    print("Under Review....!!!!")
```

```
Loan Rejected
```

In [33]:
```
# KNN Graph
```

In [34]:
```
from sklearn import neighbors
%matplotlib inline
import seaborn
```

In [37]:
```
training_data = pd.DataFrame()
import pandas as pd
try:
    training_data  = pd.read_csv("d:/Train.csv")
except:
    print("error")



training_data=training_data.dropna()
training_data
```

Out[37]:

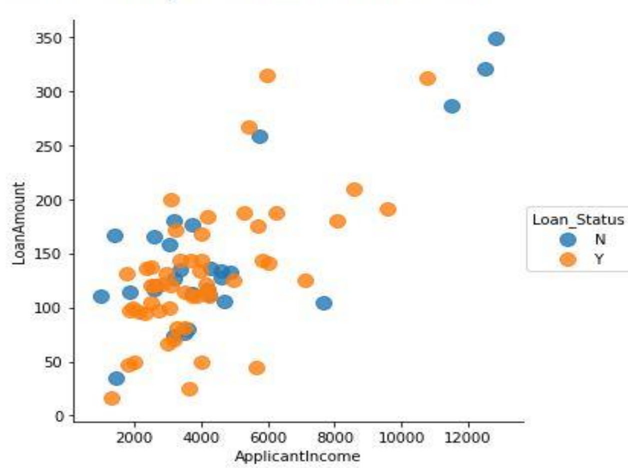
	Application_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
1	1003	M	Yes	1	Graduate	No	4583	1508	128.0	360.0	1.0	Rural	N
2	1005	M	Yes	0	Graduate	Yes	3000	0	66.0	360.0	1.0	Urban	Y
3	1006	M	Yes	0	Not Graduate	No	2583	2358	120.0	360.0	1.0	Urban	Y
4	1008	M	No	0	Graduate	No	6000	0	141.0	360.0	1.0	Urban	Y
5	1011	M	Yes	2	Graduate	Yes	5417	4196	267.0	360.0	1.0	Urban	Y
6	1013	M	Yes	0	Not Graduate	No	2333	1516	95.0	360.0	1.0	Urban	Y
7	1014	M	Yes	3+	Graduate	No	3036	2504	158.0	360.0	0.0	Semiurban	N
8	1018	M	Yes	2	Graduate	No	4006	1526	168.0	360.0	1.0	Urban	Y
9	1020	M	Yes	1	Graduate	No	12841	10968	349.0	360.0	1.0	Semiurban	N
10	1024	M	Yes	2	Graduate	No	3200	700	70.0	360.0	1.0	Urban	Y
12	1028	M	Yes	2	Graduate	No	3073	8106	200.0	360.0	1.0	Urban	Y
13	1029	M	No	0	Graduate	No	1853	2840	114.0	360.0	1.0	Rural	N
14	1030	M	Yes	2	Graduate	No	1299	1086	17.0	120.0	1.0	Urban	Y
15	1032	M	No	0	Graduate	No	4950	0	125.0	360.0	1.0	Urban	Y
17	1036	F	No	0	Graduate	No	3510	0	76.0	360.0	0.0	Urban	N
18	1038	M	Yes	0	Not Graduate	No	4887	0	133.0	360.0	1.0	Rural	N
20	1043	M	Yes	0	Not Graduate	No	7660	0	104.0	360.0	0.0	Urban	N
21	1046	M	Yes	1	Graduate	No	5955	5625	315.0	360.0	1.0	Urban	Y
22	1047	M	Yes	0	Not Graduate	No	2600	1911	116.0	360.0	0.0	Semiurban	N
25	1066	M	Yes	0	Graduate	Yes	9560	0	191.0	360.0	1.0	Semiurban	Y
26	1068	M	Yes	0	Graduate	No	2799	2253	122.0	360.0	1.0	Semiurban	Y
27	1073	M	Yes	2	Not Graduate	No	4226	1040	110.0	360.0	1.0	Urban	Y
28	1086	M	No	0	Not Graduate	No	1442	0	35.0	360.0	1.0	Urban	N

28	1086	M	No	0	Not Graduate	No	1442	0	35.0	360.0	1.0	Urban	N
31	1095	M	No	0	Graduate	No	3167	0	74.0	360.0	1.0	Urban	N
32	1097	M	No	1	Graduate	Yes	4692	0	106.0	360.0	1.0	Rural	N
33	1098	M	Yes	0	Graduate	No	3500	1667	114.0	360.0	1.0	Semiurban	Y
34	1100	M	No	3+	Graduate	No	12500	3000	320.0	360.0	1.0	Rural	N
37	1112	F	Yes	0	Graduate	No	3667	1459	144.0	360.0	1.0	Semiurban	Y
38	1114	M	No	0	Graduate	No	4166	7210	184.0	360.0	1.0	Urban	Y
39	1116	M	No	0	Not Graduate	No	3748	1668	110.0	360.0	1.0	Semiurban	Y
...	...	...	...	...	...	...	...	...	...	...	...	...	...
64	1222	F	No	0	Graduate	No	4166	0	116.0	360.0	0.0	Semiurban	N
65	1225	M	Yes	0	Graduate	No	5726	4595	258.0	360.0	1.0	Semiurban	N
66	1228	M	No	0	Not Graduate	No	3200	2254	126.0	180.0	0.0	Urban	N
67	1233	M	Yes	1	Graduate	No	10750	0	312.0	360.0	1.0	Urban	Y
68	1238	M	Yes	3+	Not Graduate	Yes	7100	0	125.0	60.0	1.0	Urban	Y
69	1241	F	No	0	Graduate	No	4300	0	136.0	360.0	0.0	Semiurban	N
70	1243	M	Yes	0	Graduate	No	3208	3066	172.0	360.0	1.0	Urban	Y
71	1245	M	Yes	2	Not Graduate	Yes	1875	1875	97.0	360.0	1.0	Semiurban	Y
72	1248	M	No	0	Graduate	No	3500	0	81.0	300.0	1.0	Semiurban	Y
74	1253	M	Yes	3+	Graduate	Yes	5266	1774	187.0	360.0	1.0	Semiurban	Y
75	1255	M	No	0	Graduate	No	3750	0	113.0	480.0	1.0	Urban	N
76	1256	M	No	0	Graduate	No	3750	4750	176.0	360.0	1.0	Urban	N
77	1259	M	Yes	1	Graduate	Yes	1000	3022	110.0	360.0	1.0	Urban	N
78	1263	M	Yes	3+	Graduate	No	3167	4000	180.0	300.0	0.0	Semiurban	N
80	1265	F	No	0	Graduate	No	3846	0	111.0	360.0	1.0	Semiurban	Y
82	1267	F	Yes	2	Graduate	No	1378	1881	167.0	360.0	1.0	Urban	N
84	1275	M	Yes	1	Graduate	No	3988	0	50.0	240.0	1.0	Urban	Y
85	1279	M	No	0	Graduate	No	2366	2531	136.0	360.0	1.0	Semiurban	Y
87	1282	M	Yes	0	Graduate	No	2500	2118	104.0	360.0	1.0	Semiurban	Y
88	1289	M	No	0	Graduate	No	8566	0	210.0	360.0	1.0	Urban	Y
89	1310	M	Yes	0	Graduate	No	5695	4167	175.0	360.0	1.0	Semiurban	Y
90	1316	M	Yes	0	Graduate	No	2958	2900	131.0	360.0	1.0	Semiurban	Y
91	1318	M	Yes	2	Graduate	No	6250	5654	188.0	180.0	1.0	Semiurban	Y
92	1319	M	Yes	2	Not Graduate	No	3273	1820	81.0	360.0	1.0	Urban	Y
93	1322	M	No	0	Graduate	No	4133	0	122.0	360.0	1.0	Semiurban	Y
94	1325	M	No	0	Not Graduate	No	3620	0	25.0	120.0	1.0	Semiurban	Y
96	1327	F	Yes	0	Graduate	No	2484	2302	137.0	360.0	1.0	Semiurban	Y
97	1333	M	Yes	0	Graduate	No	1977	997	50.0	360.0	1.0	Semiurban	Y
98	1334	M	Yes	0	Not Graduate	No	4188	0	115.0	180.0	1.0	Semiurban	Y
99	1343	M	Yes	0	Graduate	No	1759	3541	131.0	360.0	1.0	Semiurban	Y

80 rows × 13 columns

In [38]:
```
seaborn.lmplot('ApplicantIncome', 'LoanAmount', data=training_data, fit_reg=False,hue="Loan_Status", \
               scatter_kws={"marker": "D","s": 100})
```

Out[38]: <seaborn.axisgrid.FacetGrid at 0x4e85c10>



In [39]:
```
x = training_data.as_matrix(columns=['ApplicantIncome', 'LoanAmount'])
y = np.array(training_data['Loan_Status'])
print ("X: ",x)
print ("y: ",y)
```

```
X:   [[ 4583.    128.]
 [ 3000.     66.]
 [ 2583.    120.]
 [ 6000.    141.]
 [ 5417.    267.]
 [ 2333.     95.]
 [ 3036.    158.]
 [ 4006.    168.]
 [12841.    349.]
 [ 3200.     70.]
 [ 3073.    200.]
 [ 1853.    114.]
 [ 1299.     17.]
 [ 4950.    125.]
 [ 3510.     76.]
 [ 4887.    133.]
 [ 7660.    104.]
 [ 5955.    315.]
 [ 2600.    116.]
 [ 9560.    191.]
 [ 2799.    122.]
 [ 4226.    110.]
 [ 1442.     35.]
 [ 3167.     74.]
 [ 4692.    106.]
 [ 3500.    114.]
 [12500.    320.]
 [ 3667.    144.]
 [ 4166.    184.]
 [ 3748.    110.]
 [ 3600.     80.]
 [ 1800.     47.]
 [ 3941.    134.]
 [ 5649.     44.]
 [ 5821.    144.]
 [ 2645.    120.]
 [ 4000.    144.]
 [ 1928.    100.]
 [ 3086.    120.]
 [ 4230.    112.]
 [ 4616.    134.]
 [11500.    286.]
 [ 2708.     97.]
 [ 2132.     96.]
 [ 3366.    135.]
 [ 8080.    180.]
 [ 3357.    144.]
 [ 2500.    120.]
 [ 3029.     99.]
 [ 2609.    165.]
 [ 4166.    116.]
 [ 5726.    258.]
 [ 3200.    126.]
 [10750.    312.]
 [ 7100.    125.]
 [ 4300.    136.]
 [ 3208.    172.]
 [ 1875.     97.]
 [ 3500.     81.]
 [ 5266.    187.]
 [ 3750.    113.]
 [ 3750.    176.]
 [ 1000.    110.]
 [ 3167.    180.]
 [ 3846.    111.]
 [ 1378.    167.]
 [ 3988.     50.]
 [ 2366.    136.]
 [ 2500.    104.]
 [ 8566.    210.]
 [ 5695.    175.]
 [ 2958.    131.]
 [ 6250.    188.]
 [ 3273.     81.]
 [ 4133.    122.]
 [ 3620.     25.]
 [ 2484.    137.]
 [ 1977.     50.]
 [ 4188.    115.]
 [ 1759.    131.]]
y:  ['N' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'N' 'Y' 'Y' 'N' 'Y' 'Y' 'N' 'N' 'N' 'Y'
 'N' 'Y' 'Y' 'Y' 'N' 'N' 'N' 'Y' 'N' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'N'
 'Y' 'Y' 'Y' 'N' 'N' 'N' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'N' 'N' 'N' 'N' 'Y'
 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'N' 'N' 'N' 'N' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y'
 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y']
```

In [40]:
```
clf = neighbors.KNeighborsClassifier( weights = 'uniform')
trained_model = clf.fit(x,y)
print ("trained_model: ",trained_model)
```

```
trained_model:  KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=5, p=2,
           weights='uniform')
```

In [41]:
```
trained_model.score(x,y)
```

Out[41]:
```
0.7375
```

In [42]:
```
x_test = np.array([[.4,.6]])
```

In [43]:
```
trained_model.predict(x_test)
```

Out[43]:
```
array(['N'], dtype=object)
```

In [44]:
```
# We can even look at the probabilities the learner assigned
# to each class:
trained_model.predict_proba(x_test)
```

Out[44]:
```
array([[0.6, 0.4]])
```

In [50]:
```
# R E G R R E S S I O N ' S
```

In [51]:
```
    # - LINEAR REGRESSION
```

In [53]:
```
%matplotlib inline
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Reading train Data
data = pd.read_csv('d:/Train.csv')
df = data
print(data.shape)
data.tail()
```

```
(100, 13)
```

Out[53]:

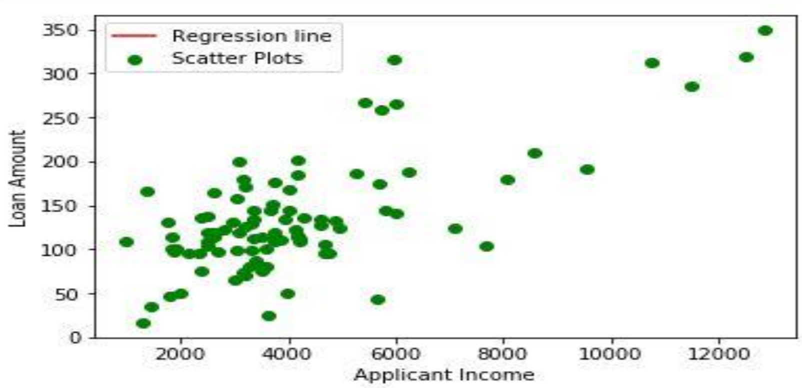
	Application_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
95	1326	M	No	0	Graduate	NaN	6782	0	NaN	360.0	NaN	Urban	N
96	1327	F	Yes	0	Graduate	No	2484	2302	137.0	360.0	1.0	Semiurban	Y
97	1333	M	Yes	0	Graduate	No	1977	997	50.0	360.0	1.0	Semiurban	Y
98	1334	M	Yes	0	Not Graduate	No	4188	0	115.0	180.0	1.0	Semiurban	Y
99	1343	M	Yes	0	Graduate	No	1759	3541	131.0	360.0	1.0	Semiurban	Y

```
In [54]: X = data['ApplicantIncome'].values
         Y = data['LoanAmount'].values
```

```
In [55]: # mean of x and y
         mean_x =np.mean(X)
         mean_y =np.mean(Y)

         # total no of values
         n=len(X)

         # using formula to calculate b0 and b1 values:-
         numer = 0
         denom = 0

         for i in range(n):
             numer+= (X[i] - mean_x)*(Y[i] - mean_y)
             denom+= (X[i] - mean_x)**2
         b1 = numer/denom
         b0 = mean_y - (b1 * mean_x)

         # Print coefficients
         print("b1 = {} and b0 = {}".format(b1,b0))

         b1 = nan and b0 = nan
```

```
In [56]: # Plotting values and regression line
         max_x = np.max(x) #+ 100
         min_x = np.min(x) #- 100

         # Calculating line values x and y
         x = np.linspace(min_x,max_x,1000)
         y = b0 * b1 * x
```

```
# Plotting line
plt.plot(x,y,color='red',label = 'Regression line')
# Plotting scatter Plots
plt.scatter(X,Y,color='green',label = 'Scatter Plots')

plt.xlabel('Applicant Income')
plt.ylabel('Loan Amount')
plt.legend()
plt.show()
```




```
In [57]: # r square method to determine effeciency

         ss_t = 0
         ss_r = 0
         for i in range(n):
             y_pred = b0 *  b1 * X[i]
             ss_t +=(Y[i] - mean_y) **2
             ss_r +=(Y[i] - y_pred) **2
         r2 = 1 - (ss_r/ss_t)
         print(r2)

         nan
```

Code

#importing packages

```
import pandas as pd
from sklearn import svm
from sklearn import cross_validation
from sklearn.linear_model import LinearRegression
from sklearn.feature_selection import SelectKBest, f_classif
import numpy as np
import matplotlib.pyplot as plt
```

#reading csv files

```
loan = pd.read_csv('d:/traindata.csv')
loan_test = pd.read_csv('d:/testdata.csv')
loan.describe()

loan.count()

loan.loc[loan['Education'] == 'Graduate', 'Education'] = 1
loan.loc[loan['Education'] == 'Not Graduate', 'Education'] = 0
loan.loc[loan['Property_Area'] == 'Rural', 'Property_Area'] = 0
loan.loc[loan['Property_Area'] == 'Semiurban', 'Property_Area'] = 1
loan.loc[loan['Property_Area'] == 'Urban', 'Property_Area'] = 2
loan2 = loan
loan3 = loan
loan4 = loan

predictors = ['Education', 'ApplicantIncome', 'CoapplicantIncome', 'Property_Area']
model = svm.SVC(probability=True)
scores = cross_validation.cross_val_score(model, loan[predictors], loan['Loan_Status'], cv=12)
print(scores)
print(scores.mean())

loan3['LoanAmount'] = loan['LoanAmount'].fillna(loan['LoanAmount'].mean())
loan3['Loan_Amount_Term'] =
loan['Loan_Amount_Term'].fillna(loan['Loan_Amount_Term'].mean())
#print(loan['Credit_History'].mode())
#loan3['Credit_History'] = loan['Credit_History'].fillna(loan['Credit_History'].mode())
loan3['Credit_History'] = loan3['Credit_History'].fillna(loan3['Credit_History'].median())
```

```
predictors = ['Credit_History']

model = svm.SVC(probability=True)
scores = cross_validation.cross_val_score(model, loan3[predictors], loan3['Loan_Status'], cv=3)
print(scores)
print(scores.mean())
model.fit(loan3[predictors], loan3['Loan_Status'])

loan_test.describe()

print(loan_test.count())
loan_test.loc[loan_test['Education'] == 'Graduate', 'Education'] = 1
loan_test.loc[loan_test['Education'] == 'Not Graduate', 'Education'] = 0
loan_test.loc[loan_test['Property_Area'] == 'Rural', 'Property_Area'] = 0
loan_test.loc[loan_test['Property_Area'] == 'Semiurban', 'Property_Area'] = 1
loan_test.loc[loan_test['Property_Area'] == 'Urban', 'Property_Area'] = 2


loan_test2 = loan_test
loan_test2['Credit_History'] = loan_test['Credit_History'].fillna(loan_test['Credit_History'].mode())
loan_test2 = loan_test2[np.logical_not(np.isnan(loan_test2.Credit_History))]
loan_test2 = loan_test2.reset_index(drop=True)
print(loan_test2[predictors])
sattu = (model.predict(loan_test2[predictors]))
print(sattu)
```

#For Non Numeric counts.

```
loan['Property_Area'].value_counts()

loan['ApplicantIncome'].hist()

loan['ApplicantIncome'].hist(bins=20)

loan.boxplot(column ='ApplicantIncome', figsize=(5,5), grid=True)
```

 # NATIVE BAYES.....!!!!

```
try:
   mydata = pd.read_csv("D:/traindata.csv")
except:
   print("File open Error....!!!!")
df = mydata
df
data = pd.DataFrame()
```

```
df.Property_Area.unique()

# number of Yes
print("No of people :-")
n = df["Loan_Status"][df["Loan_Status"] == "Y"].count()
print("Yes = ",n)

# number of No

n = df["Loan_Status"][df["Loan_Status"] == "N"].count()
print("Rural = ",n)

# number of Graduate
print("No of people :-\n")
n = df["Education"][df["Education"] == "Graduate"].count()
print("Graduate = ",n)
# number of Not Graduate
n = df["Education"][df["Education"] == "Not Graduate"].count()
print("Not Graduate = ",n)

n = df["Married"][df["Married"] == "Yes"].count()
print("\nMarried = ",n)

n = df["Married"][df["Married"] == "No"].count()
print("Un Married = ",n)

n = df["Gender"][df["Gender"] == "M"].count()
print("\nMale = ",n)

n = df["Gender"][df["Gender"] == "F"].count()
print("Female = ",n)


# calculate likelywood
# group the data by gender and calculate the of means of each feature
data_means = df.groupby("Loan_Status").mean()

# view the values
data_means

# MEAN
# print("Mean in terms of loan w.r.t Property area::")
# rural_mean = df['LoanAmount'].groupby(df['Rural']).mean()
```

```
# group the data by gender and calculate the varieince of each feature
data_variance = df.groupby("Loan_Status").var()

# view the values
data_variance

# MEANS wrt Property_Area
print("MEAN :-")

yes_mean = data_means["LoanAmount"][data_means.index == "Y"].values[0]
print("Yes = ",yes_mean)

no_mean = data_means["LoanAmount"][data_means.index == "N"].values[0]
print("No = ",no_mean)

# VARIENCE wrt. Yes / No
print("VARIANCE :-")
yes_variance = data_variance["LoanAmount"][data_variance.index == "Y"].values[0]
print("Yes = ",yes_variance)

no_variance = data_variance["LoanAmount"][data_variance.index == "N"].values[0]
print("No = ",no_variance)

# calculation of prior for both the classes Y / N
n_yes = df["Loan_Status"][df["Loan_Status"] == "Y"].count()
n_no = df["Loan_Status"][df["Loan_Status"] == "N"].count()
total_yn =n_yes + n_no

# number of rural divided by total no. people
p_yes = n_yes/total_yn

# number of urban divided by total people
p_no = n_no/total_yn

print("Probability of :-\n")
print("P(\"Yes\") = {} \nP(\"No\") = {}".format(p_yes,p_no))


# Create a function  that calculates p(x/y)
import numpy as np
def p_x_given(x,mean_y,variance_y):
    # input the arguments into a probability density function
    p = 1/(np.sqrt(2 * np.pi * variance_y)) * np.exp((-(x - mean_y) ** 2)/(2 * variance_y))

    return p
```

```
# numerator of the posterior if the unclassified observation is a YES
num_posterior_yes = p_yes * \
p_x_given(loan_test["CoapplicantIncome"][6],yes_mean,yes_variance) * \
p_x_given(loan_test["LoanAmount"][6],yes_mean,yes_variance) *\
p_x_given(loan_test["Loan_Amount_Term"][6],yes_mean,yes_variance)

print("num_posterior_yes = ",num_posterior_yes)

# numerator of the posterior if the unclassified observation is a NO
num_posterior_no = p_no * \
p_x_given(loan_test["CoapplicantIncome"][6],no_mean,no_variance) * \
p_x_given(loan_test["LoanAmount"][6],no_mean,no_variance) *\
p_x_given(loan_test["Loan_Amount_Term"][6],no_mean,no_variance)

print("num_posterior_no = ",num_posterior_no)


if (num_posterior_yes > num_posterior_no):
   print("Loan Approved")
elif (num_posterior_no > num_posterior_yes):
   print("Loan Rejected")
else:
   print("Under Review....!!!!")

# NOW PROCEEDING TOWARDS KNN

from sklearn import neighbors
%matplotlib inline
import seaborn

training_data = pd.DataFrame()
import pandas as pd
try:
   training_data  = pd.read_csv("D:/traindata.csv")
except:
   print("error")



training_data=training_data.dropna()
training_data.head()

seaborn.lmplot('ApplicantIncome', 'LoanAmount', data=training_data,
```

```
fit_reg=False,hue="Loan_Status", \
            scatter_kws={"marker": "D","s": 100})

x = training_data.as_matrix(columns=['ApplicantIncome', 'LoanAmount'])
y = np.array(training_data['Loan_Status'])
print ("X: ",x)
print ("y: ",y)

clf = neighbors.KNeighborsClassifier( weights = 'uniform')
trained_model = clf.fit(x,y)
print ("trained_model: ",trained_model)

trained_model.score(x,y)

x_test = np.array([[.4,.6]])

trained_model.predict(x_test)
```

 # We can even look at the probabilities the learner assigned
 # to each class:

```
trained_model.predict_proba(x_test)
```

 # - LINEAR REGRESSION

```
%matplotlib inline
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Reading train Data
data = pd.read_csv('d:/Train.csv')
df = data
print(data.shape)
data.tail()

X = data['ApplicantIncome'].values
Y = data['LoanAmount'].values

# mean of x and y
mean_x =np.mean(X)
mean_y =np.mean(Y)

# total no of values
n=len(X)
```

```
# using formula to calculate b0 and b1 values:-
numer = 0
denom = 0

for i in range(n):
    numer+= (X[i] - mean_x)*(Y[i] - mean_y)
    denom+= (X[i] - mean_x)**2
b1 = numer/denom
b0 = mean_y - (b1 * mean_x)

# Print coefficients
print("b1 = {} and b0 = {}".format(b1,b0))

# Plotting values and regression line
max_x = np.max(x) #+ 100
min_x = np.min(x) #- 100

# Calculating line values x and y
x = np.linspace(min_x,max_x,1000)
y = b0 * b1 * x

# Plotting line
plt.plot(x,y,color='red',label = 'Regression line')
# Plotting scatter Plots
plt.scatter(X,Y,color='green',label = 'Scatter Plots')

plt.xlabel('Applicant Income')
plt.ylabel('Loan Amount')
plt.legend()
plt.show()

# r square method to determine effeciency

ss_t = 0
ss_r = 0
for i in range(n):
    y_pred = b0 *  b1 * X[i]
    ss_t +=(Y[i] - mean_y) **2
    ss_r +=(Y[i] - y_pred) **2
r2 = 1 - (ss_r/ss_t)
print(r2)
```

Certificate

This is to certify that *Mr. Sk Samim Islam* of *Techno India Batanagar*, registration number: *153320110038*, has successfully completed a project on *Predict Loan Application Status* using *Machine Learning with Python* under the guidance of *Prof. Arnab Chakraborty*.

____

[ARNAB CHAKRABORTY]

Didactics IT Solutions