



# BANK LOAN PREDICTION

26.07.2019

---

SOUMYOJIT SINGHO ROY

HENRY HARVIN

C-102 SECTOR 2

NOIDA 201301

## Overview

A Company wants to automate the loan eligibility process (real-time) based on customer detail provided while filling the 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 customer's segments, those are eligible for loan amounts so that they can specifically target these customers. Here they have provided a data set.

## Goals

### 1. Bank-loan prediction

In this project, we are going to predict if the given applicant is capable of getting a loan from the organization after reading their data and we will also come out with different possibilities of getting a loan from the company

## Dataset

We have imported the train(consist all variable) and test(consist all variable from train dataset except Loan\_status) dataset from the given repository and it shows as below:

	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

The dataset consist of a given set of columns:

Variable	Description
Loan_ID	Unique Loan ID
Gender	Male/ Female
Married	Applicant married (Y/N)
Dependents	Number of dependents
Education	Applicant Education (Graduate/ Under Graduate)
Self_Employed	Self-employed (Y/N)
ApplicantIncome	Applicant income
CoapplicantIncome	Coapplicant income
LoanAmount	Loan amount in thousands
Loan_Amount_Term	Term of the loan in months
Credit_History	credit history meets guidelines
Property_Area	Urban/ Semi-Urban/ Rural
Loan_Status	Loan approved (Y/N)

## Data Cleaning and learning the Data

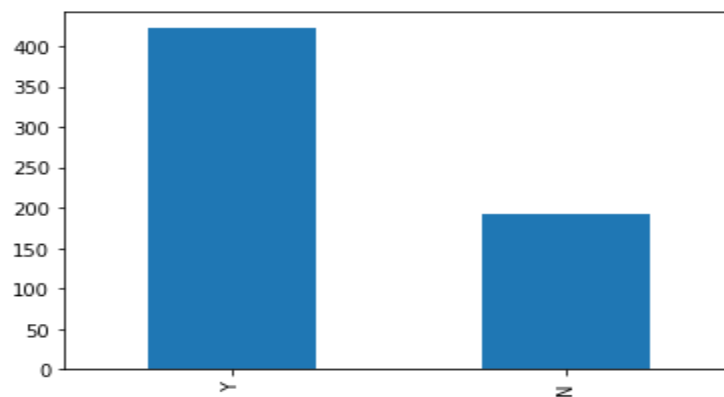
1. Setting Loan\_Id as index---> `train.set_index('Loan_ID')`
2. Creating a copy of train and test----> `train_original = train.copy()`  
`test_original = test.copy()`
3. shape of the data set ---> `print(train.shape)` : 614, 13  
`print(test.shape)` : 367, 12
4. count the loan status(no.of approve and not approve loan) ---->  
`train['Loan_Status'].value_counts()` : Y 422  
N 192
5. Removing the null value in the dataset

## 6. Describing the dataset-----> train.describe()

	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	12.000000	0.000000
25%	2877.500000	0.000000	100.000000	360.000000	1.000000
50%	3812.500000	1188.500000	128.000000	360.000000	1.000000
75%	5795.000000	2297.250000	168.000000	360.000000	1.000000
max	81000.000000	41667.000000	700.000000	480.000000	1.000000

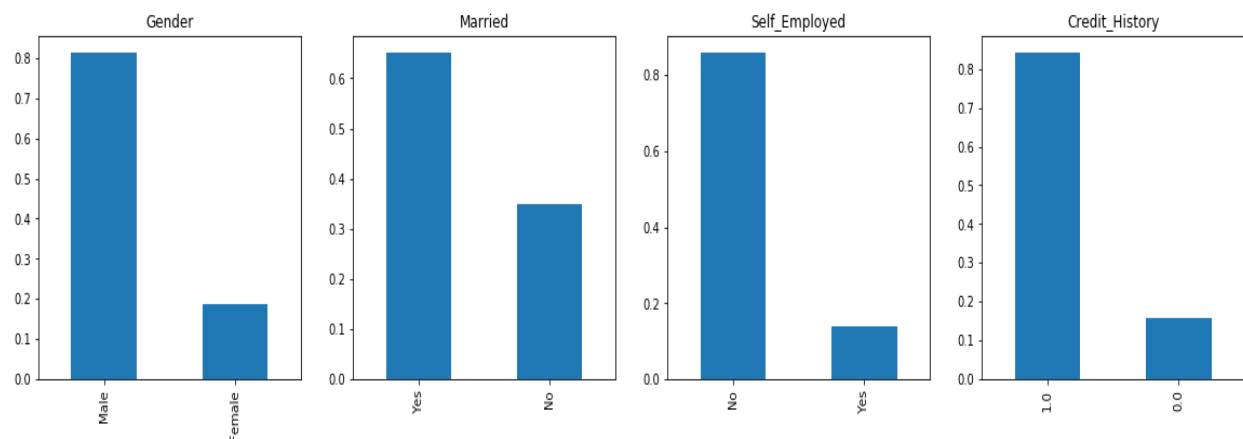
## Visualization

### 1. plotting no. of loan approved → train['Loan\_Status'].value\_counts().plot.bar()



### 2. Visualizing Categorical variable

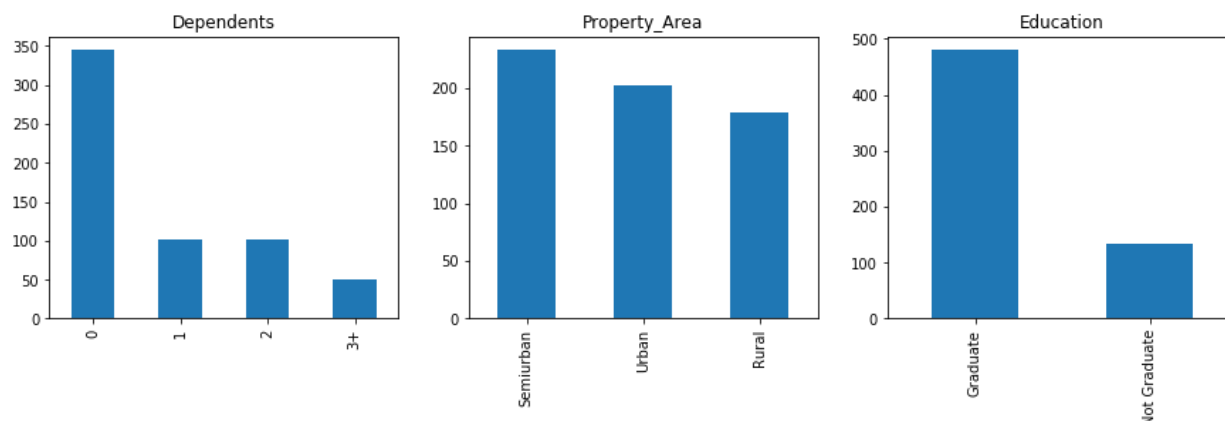
#### 1. plotting gender variation married variation employed variation and credit variation



So from the given plot, we can interpret that 80% of applicants in the dataset are male. Around 65% of the applicants in the dataset are married. Around 15% of applicants in the dataset are self-employed. Around 85% of applicants have repaid their debts.

### 3. Visualizing Ordinal variable

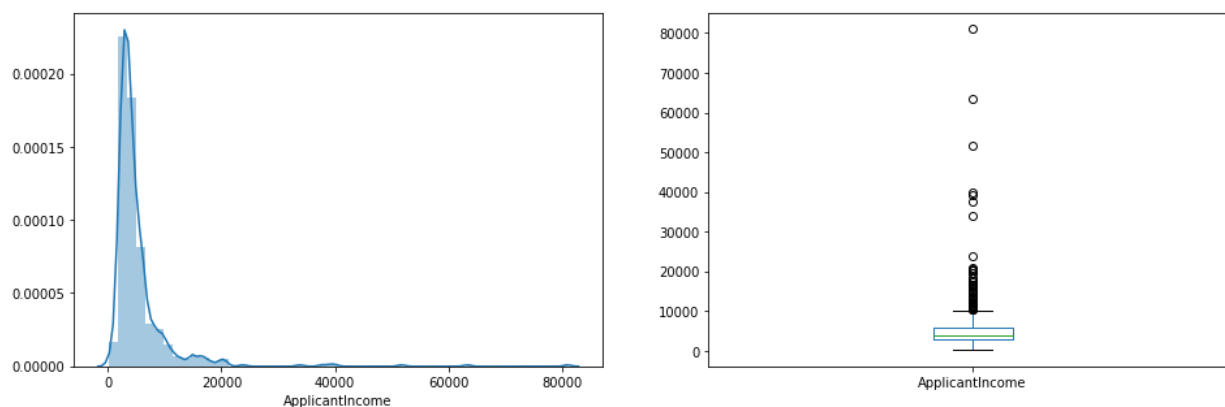
#### 1. plotting dependent, property area and education variation



Following inferences can be made from the above bar plots:

Most of the applicants don't have any dependents. Around 80% of the applicants are Graduate. Most of the applicants are from Semiurban area.

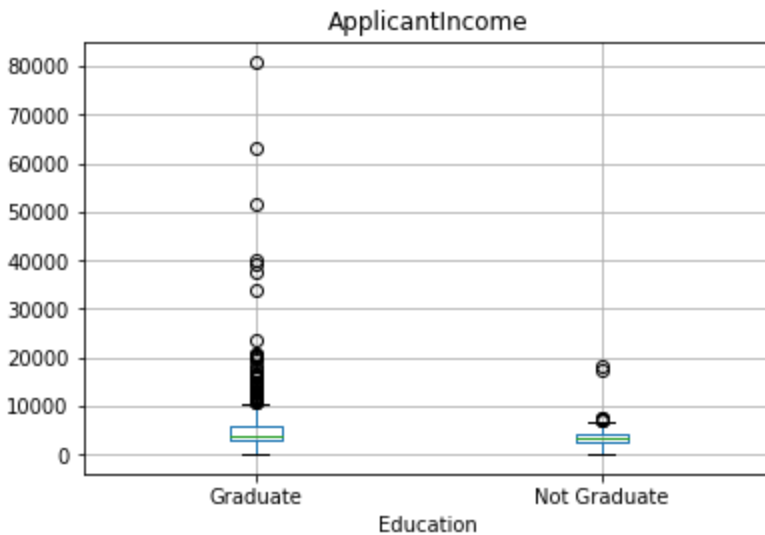
#### 2. Plotting applicant income:



It can be inferred that most of the data in the distribution of applicant income is towards left which means it is not normally distributed. We will try to make it normal in later sections as algorithms works better if the data is normally distributed.

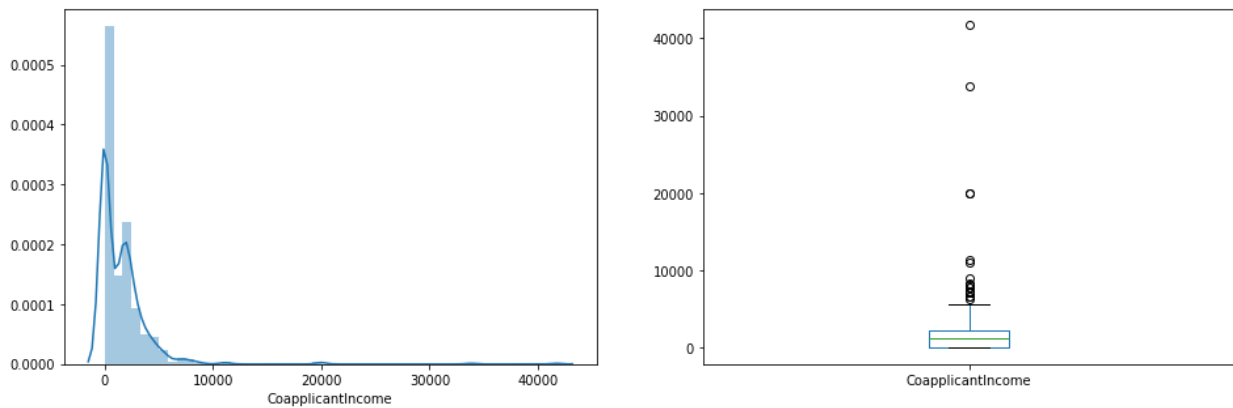
The boxplot confirms the presence of a lot of outliers/extreme values. This can be attributed to the income disparity in the society. Part of this can be driven by the fact that we are looking at people with different education levels.

### 3. plotting applicant income vs education



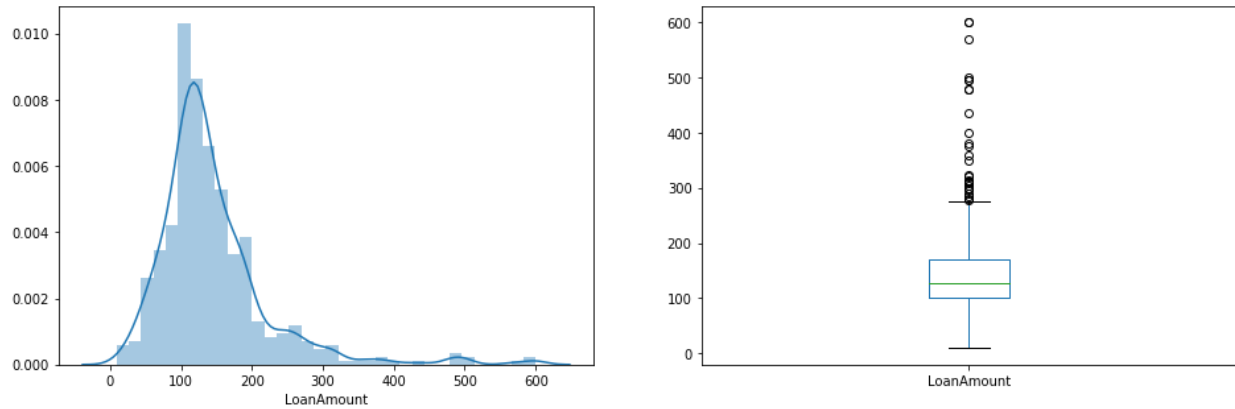
We can see that there are a higher number of graduates with very high incomes, which are appearing to be the outliers.

### 4. Coapplicant Income



We see a similar distribution as that of the applicant income. Majority of coapplicant's income ranges from 0 to 5000. We also see a lot of outliers in the coapplicant income and it is not normally distributed.

## 5. Loan Amount



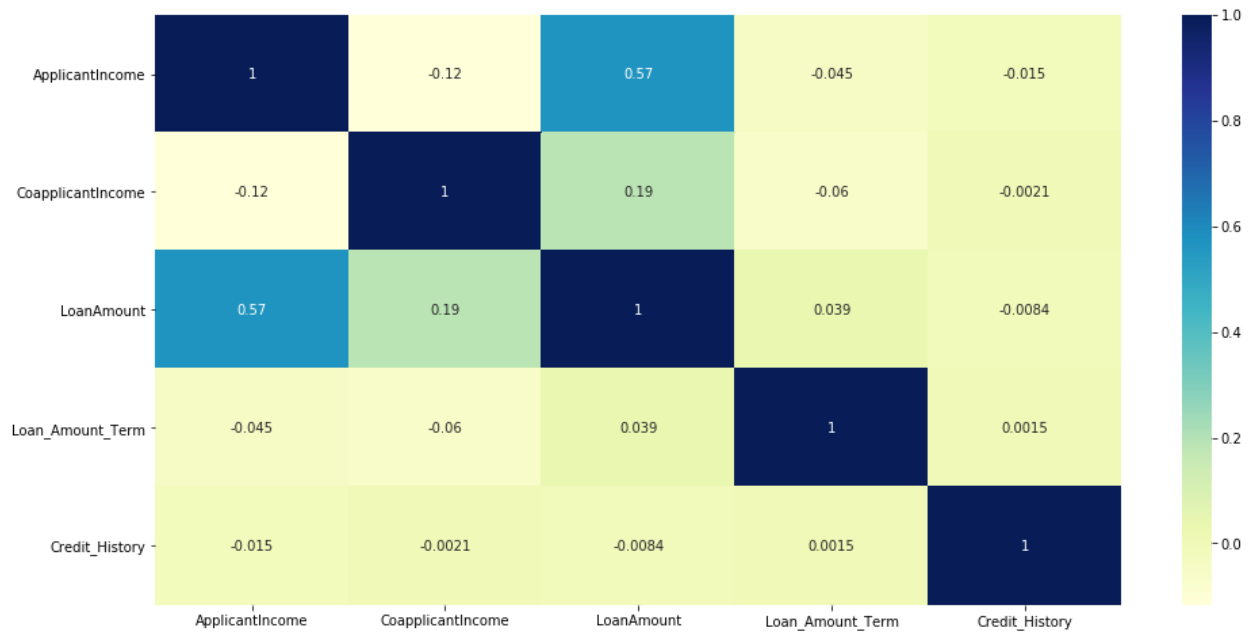
We see a lot of outliers in this variable and the distribution is fairly normal.

## Bivariate Analysis

1 checking out the correlation among the variables --> `cor = train_original.corr()`

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
ApplicantIncome	1.000000	-0.116605	0.570909	-0.045306	-0.014715
CoapplicantIncome	-0.116605	1.000000	0.188619	-0.059878	-0.002056
LoanAmount	0.570909	0.188619	1.000000	0.039447	-0.008433
Loan_Amount_Term	-0.045306	-0.059878	0.039447	1.000000	0.001470
Credit_History	-0.014715	-0.002056	-0.008433	0.001470	1.000000

## 2. Plotting the Heat map



so we can conclude from the given heat map that credit history is strongly related to loan amount and applicant income but it is negatively related with loan amount term and co-applicant income loan\_amount\_term is strongly related with credit history and loan amount and is negatively related with applicant and co-applicant income loan amount is strongly related with the loan amount term as applicant income and very strongly with applicant income but it is negatively related with credit history

## Hypothesis

1. let's check our first hypothesis which states Applicants with high income should have more chances of loan approval

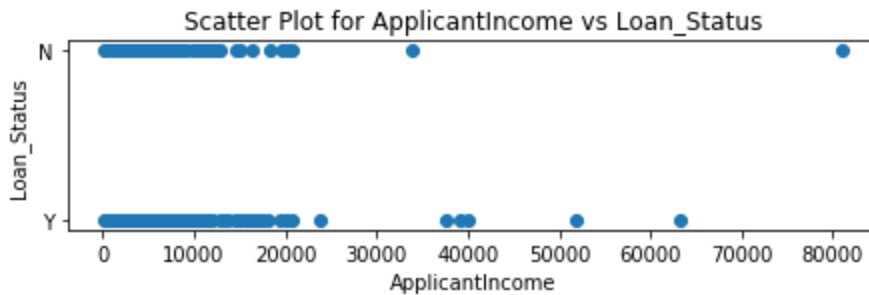
Code

```
def scatterplot(df,var):
```

```
    plt.scatter(df[var],train['Loan_Status'])
    plt.xlabel(var); plt.ylabel('Loan_Status')
    plt.title('Scatter Plot for '+var+' vs Loan_Status')
    plt.figure(figsize=(16,8))
    plt.subplot(4,2,1)
```



```
scatterplot(train,'ApplicantIncome')
plt.show()
plt.tight_layout()
```



so we can conclude from the plot the the applicant having higher income has more chance to get the loan as the density of the scatter is high towards the higher income side

2. lets check our second hypothesis Applicants who have repaid their previous debts should have higher chances of loan approval.

```
train.groupby(['Credit_History','Loan_Status']).size().unstack()
```

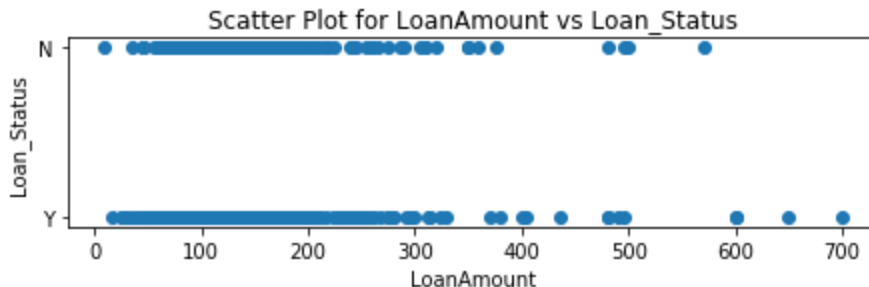
	Loan_Status	
	N	Y
Credit_History		
0.0	82	7
1.0	97	378

so it is clear from the above table that the customer who repaid their loan has the higher chance to to get their loan approved

3. lets check our third hypothesis that Loan approval should also depend on the loan amount

Code

```
plt.figure(figsize=(16,8))
plt.subplot(4,2,1)
scatterplot(train,'LoanAmount')
plt.show()
plt.tight_layout()
```



so we can conclude from the plot the the applicant with less loan amount has more chance to get the loan as the density of the scatter is high towards the higher loan side

## Creating dummy variable for all catagorical data

in_ID	Education	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Status	1.0	36.0	60.0	84.0	...	360.0	480.0	Male	1	2	3+	Yes	Yes	Semiurban
11002	Graduate	5849	0.0	146.0	Y	1	0	0	0	...	1	0	1	0	0	0	0	0	0
11003	Graduate	4583	1508.0	128.0	N	1	0	0	0	...	1	0	1	1	0	0	1	0	0
11005	Graduate	3000	0.0	66.0	Y	1	0	0	0	...	1	0	1	0	0	0	1	1	0
11006	Not Graduate	2583	2358.0	120.0	Y	1	0	0	0	...	1	0	1	0	0	0	1	0	0
11008	Graduate	6000	0.0	141.0	Y	1	0	0	0	...	1	0	1	0	0	0	0	0	0

24 columns

## Model Building

Now we need to predict weather the candidate is going to get the loan or not so for that we are going to build a logistic regression model and a random forest model by splitting the data into train and test.

Code

Logistic regression

```
x=train[num_vars]
```

```
y=train['Loan_Status']
```


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

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=0)
```

```
from sklearn import metrics
```

```
logreg = LogisticRegression()
```

```
logreg.fit(x_train, y_train)
```



```

y_pred=logreg.predict(x_test)
y_pred=pd.DataFrame(y_pred, columns=['Loan_Status'])
y_pred['Loan_Status'].value_counts()
from sklearn.metrics import accuracy_score
print('Logistic regression accuracy: ',(accuracy_score(y_test,y_pred )*100).round(3),'%')
#accuracy score of the model

```

Output   Logistic regression accuracy: 82.927 %

So we are getting a accuracy of 82.97% in our model

Random forest

```

from sklearn.ensemble import RandomForestClassifier
rf=RandomForestClassifier()
rf.fit(x_train,y_train)

rf_pred=pd.DataFrame(rf.predict(x_test), columns=['Loan_Status'])
acc=accuracy_score(y_test,rf_pred)*100
print('Random Forest accuracy: ',acc.round(3),'%') #accuracy score of the model

```

Output -- Random Forest accuracy: 85.366 %

So we getting a accuracy of 85.366% from our random forest model

Plotting the ROC curve

Code

```

from sklearn.metrics import roc_curve, roc_auc_score
fpr,tpr,threshold=roc_curve(y_test,y_pred)
fpr_rf,tpr_rf,threshold_rf=roc_curve(y_test,rf_pred)

roc_auc=roc_auc_score(y_test,y_pred)

```

```
roc_auc_rf=roc_auc_score(y_test,rf_pred)

plt.title('Receiver Operating Characteristic')
plt.plot(fpr,tpr,'b',label='Logistic Regression (area=%0.2f)%roc_auc')
plt.plot(fpr_rf,tpr_rf,'g',label='Random Forest (area=%0.2f)%roc_auc_rf')
plt.legend(loc='lower right')
plt.plot([0,1],[0,1],'r--')
plt.xlim([0,1])
plt.ylim([0,1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```

