HENRY HARVIN®
EDUCATION

Project Title – **Loan Prediction**

Mentor Name: Ms. Pooja Gupta (Designation)

Name: Tusar kumar samal

Course: Summer Internship Programme (SIP) Python

Batch: 12th Jun to 28th july 2019

Job: Business Analyst Associate (Intern)

Institute: Lovely professional university.

DECLARATION

I hereby declare that the project report entitled “**Loan Prediction**” submitted by me to **HENRY HARVIN EDUCATION INDIA** is a record of bonafied project work carried out by me under the guidance of MS. POOJA GUPTA. This project is an original report with references taken from websites and help from mentors and teachers.

TABLE OF CONTENTS

Abstract:

The main income earning assets for a bank are loans. In a first, personal loans are set to account for most of incremental growth in non-food bank credit during 2017-18. According to the Reserve Bank of India (RBI) data on sectoral deployment of bank credit, personal loans, which include home, vehicle and education loans, accounted for a record 96 per cent of incremental non-food credit in the last financial year. In this project a data scientist wants to know the eligible person to the right to give loan from the bank. There are so many categories to predict the loan, these are categorical variables and ordinal variables. In categorical variables 'gender, marital status, self-employed and credit history and in ordinal variable 'dependents, property area and education'. So those are the main categories to find out the eligible person for taking loan from bank. Here also I use logistic regression and random forest for choosing the best method for loan prediction.

INTRODUCTION

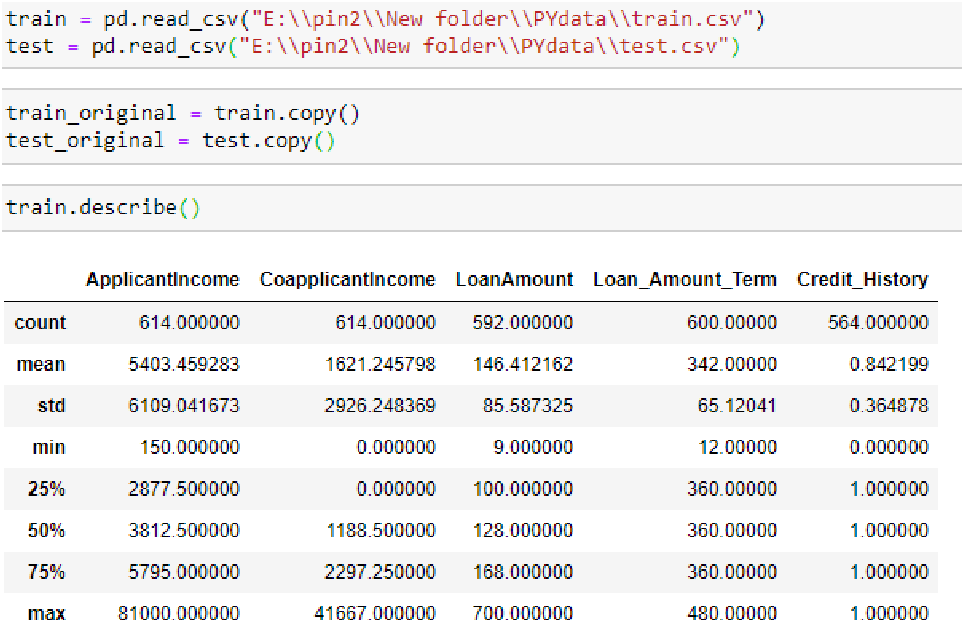
A Company wants to automate the loan eligibility process based on customer detail provided while filling online application form. These details are Gender, Marital Status, Education, Number of Dependents, Income, Loan Amount, Credit History and others. To automate this process, they have given a problem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

DATA

- **Pandas**
- **Numpy**
- **Matplotlib**
- **Seaborn**
- **Sklearn**

Importing Data with read_csv()

The first step to any data science project is to import your data. Often, you'll work with data in Comma Separated Value (CSV) files and run into problems at the very start of your workflow.



```
train = pd.read_csv("E:\\pin2\\New folder\\PYdata\\train.csv")
test = pd.read_csv("E:\\pin2\\New folder\\PYdata\\test.csv")
```

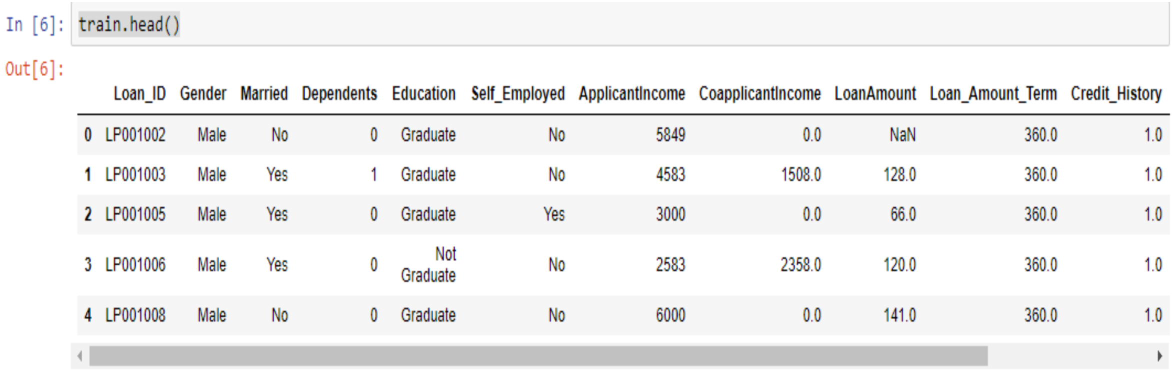
```
train_original = train.copy()
test_original = test.copy()
```

```
train.describe()
```

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
count	614.000000	614.000000	592.000000	600.00000	564.000000
mean	5403.459283	1621.245798	146.412162	342.00000	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

Train.head()

As shown in the output image, it can be seen that the index of returned rows is ranging from 0 to 4. Hence, top 5 rows were returned.

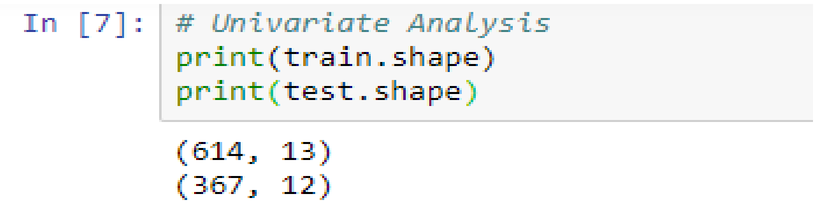


```
In [6]: train.head()
Out[6]:
```

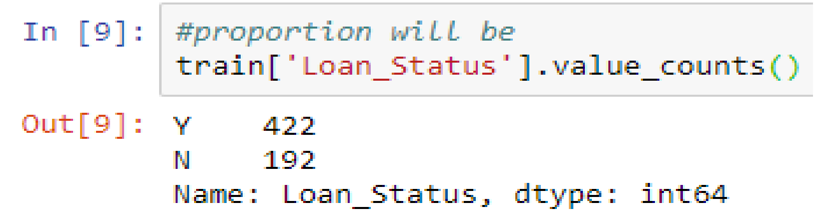
	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	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

Univariate Analysis:

Univariate analysis is the simplest form of analyzing data. “Uni” means “one”, so in other words your data has only one variable. It doesn’t deal with causes or relationships (unlike regression) and it’s major purpose is to describe; it takes data, summarizes that data and finds patterns in the data.



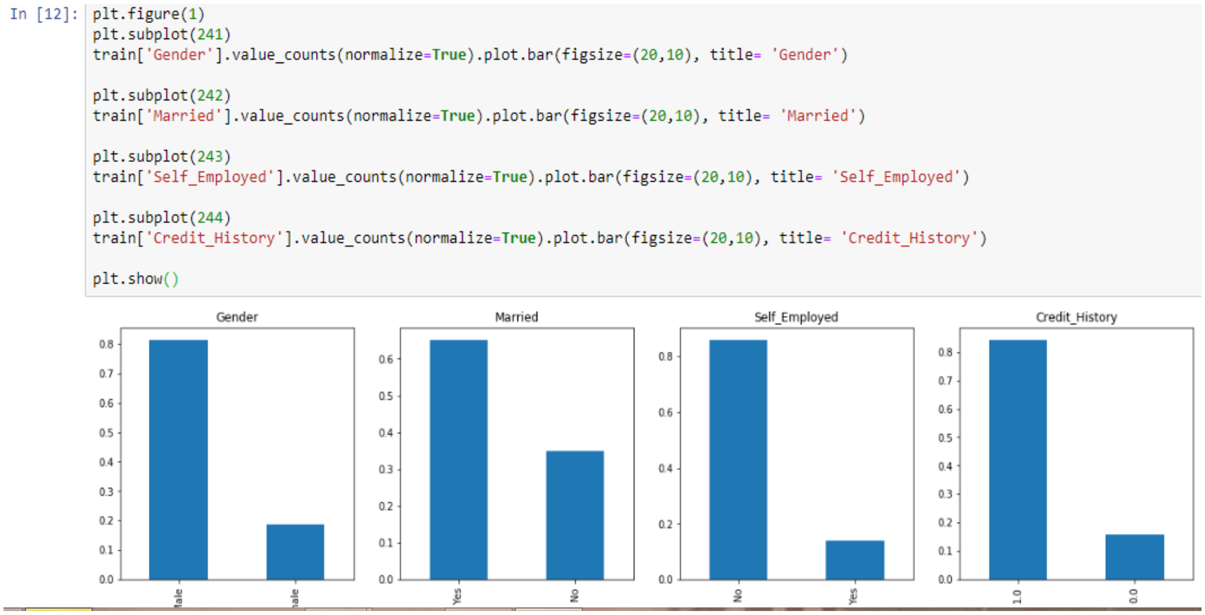
```
In [7]: # Univariate Analysis
        print(train.shape)
        print(test.shape)

        (614, 13)
        (367, 12)
```



```
In [9]: #proportion will be
        train['Loan_Status'].value_counts()

Out[9]: Y    422
        N    192
        Name: Loan_Status, dtype: int64
```

Visualizing Categorical variable:

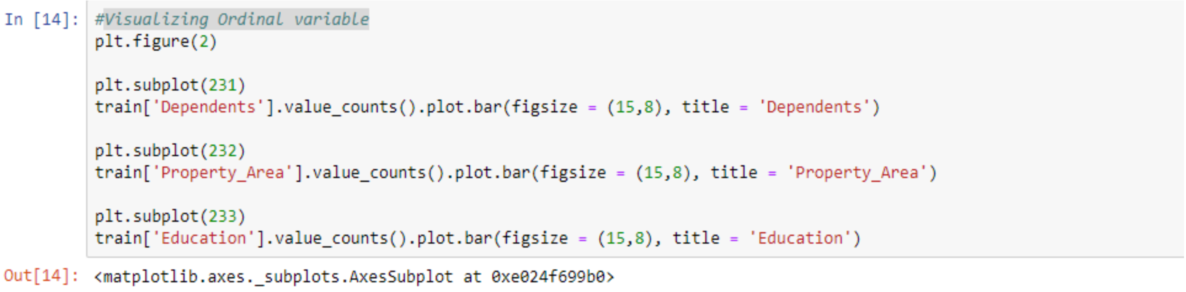
A categorical variable is one that has two or more categories, but there is no intrinsic ordering to the categories. For example, gender is a categorical variable having two categories (male and female) and there is no intrinsic ordering to the categories.



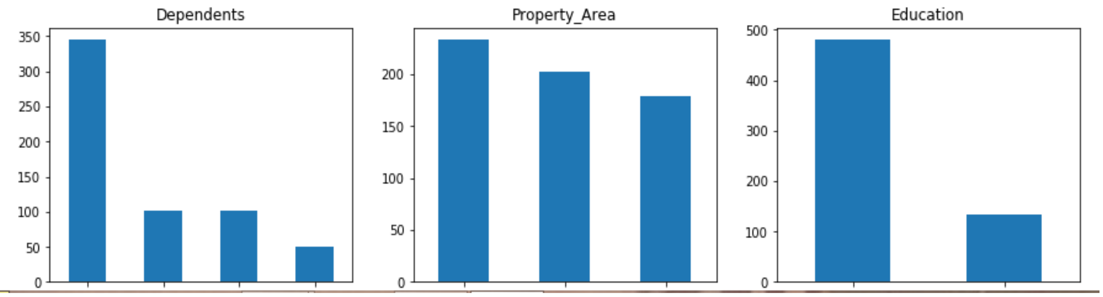
```
In [12]: plt.figure(1)
         plt.subplot(241)
         train['Gender'].value_counts(normalize=True).plot.bar(figsize=(20,10), title= 'Gender')

         plt.subplot(242)
         train['Married'].value_counts(normalize=True).plot.bar(figsize=(20,10), title= 'Married')

         plt.subplot(243)
         train['Self_Employed'].value_counts(normalize=True).plot.bar(figsize=(20,10), title= 'Self_Employed')

         plt.subplot(244)
         train['Credit_History'].value_counts(normalize=True).plot.bar(figsize=(20,10), title= 'Credit_History')

         plt.show()
```

Visualizing Ordinal variable:

ordinal values represent discrete and ordered units. It is therefore nearly the same as nominal data, except that it’s ordering matters.

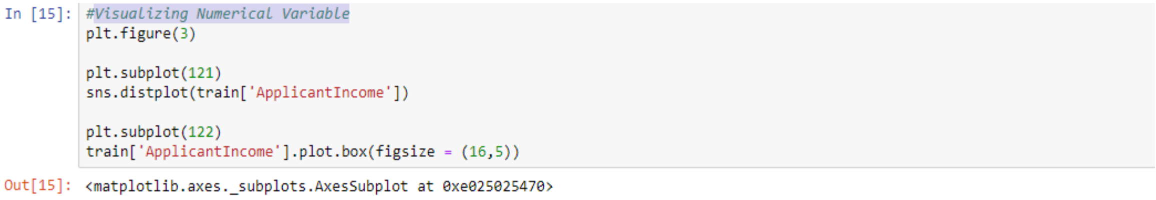


```
In [14]: #Visualizing Ordinal variable
         plt.figure(2)

         plt.subplot(231)
         train['Dependents'].value_counts().plot.bar(figsize = (15,8), title = 'Dependents')

         plt.subplot(232)
         train['Property_Area'].value_counts().plot.bar(figsize = (15,8), title = 'Property_Area')

         plt.subplot(233)
         train['Education'].value_counts().plot.bar(figsize = (15,8), title = 'Education')

Out[14]: <matplotlib.axes._subplots.AxesSubplot at 0xe024f699b0>
```

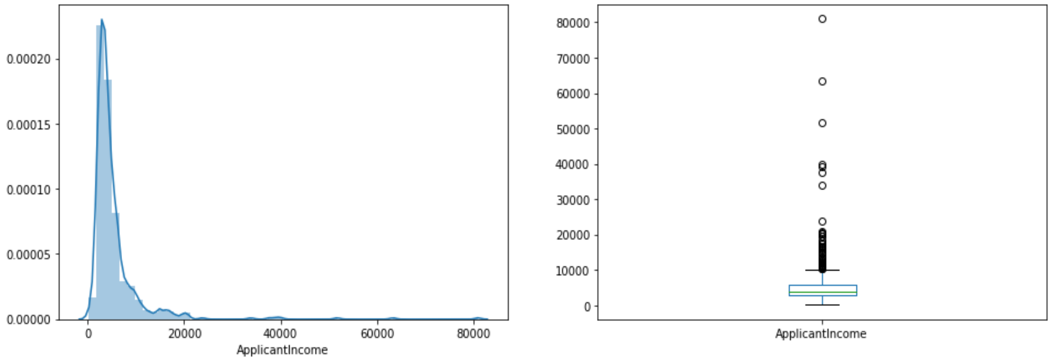


Visualizing Numerical Variable:

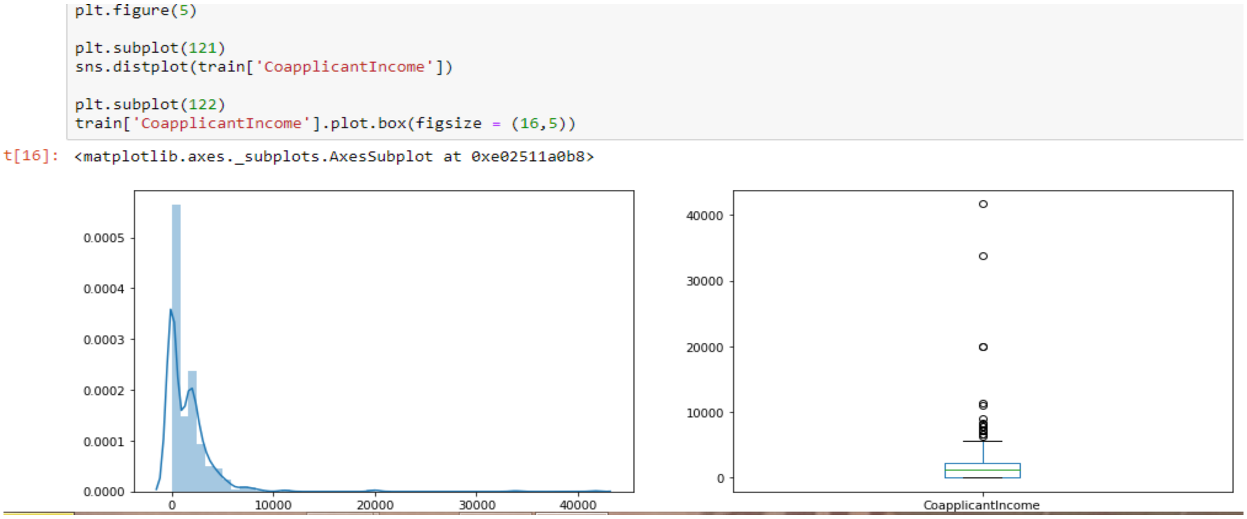
This type of data **can’t be measured but it can be counted**. It basically represents information that can be categorized into a classification.



```
In [15]: #Visualizing Numerical Variable
         plt.figure(3)

         plt.subplot(121)
         sns.distplot(train['ApplicantIncome'])

         plt.subplot(122)
         train['ApplicantIncome'].plot.box(figsize = (16,5))

Out[15]: <matplotlib.axes._subplots.AxesSubplot at 0xe025025470>
```



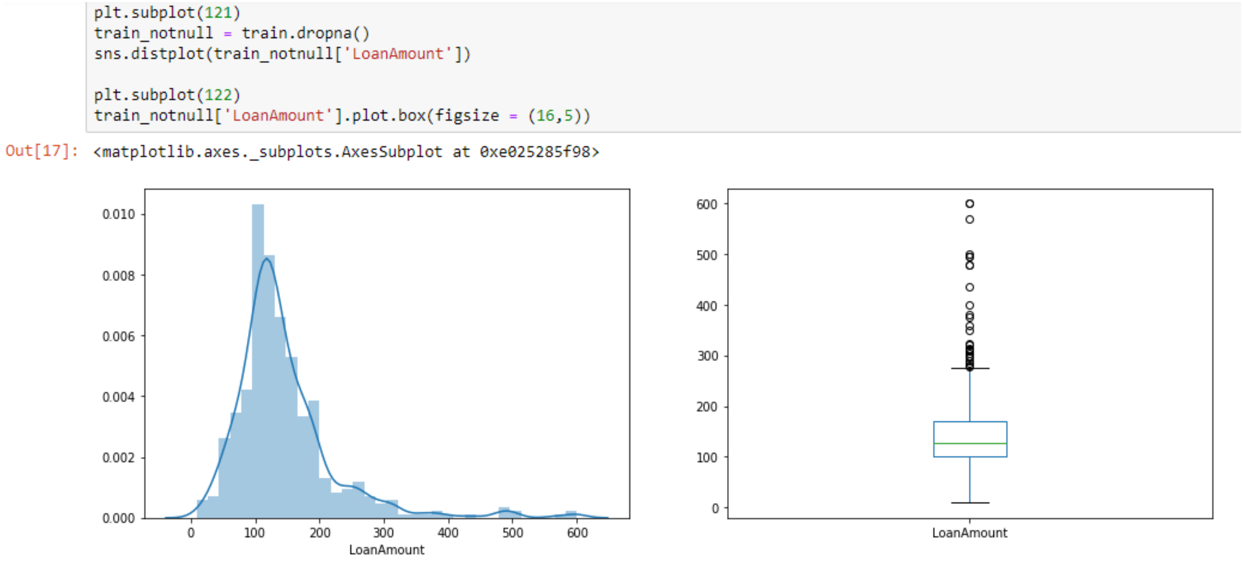
Co applicant Income:



```
plt.figure(5)

plt.subplot(121)
sns.distplot(train['CoapplicantIncome'])

plt.subplot(122)
train['CoapplicantIncome'].plot.box(figsize = (16,5))
```

```
t[16]: <matplotlib.axes._subplots.AxesSubplot at 0xe02511a0b8>
```

Loan Amount:

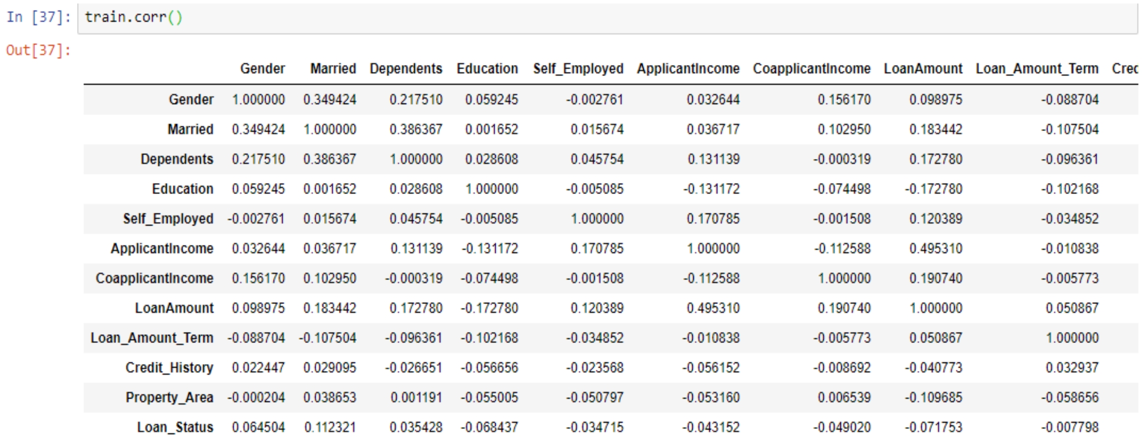


```
plt.subplot(121)
train_notnull = train.dropna()
sns.distplot(train_notnull['LoanAmount'])

plt.subplot(122)
train_notnull['LoanAmount'].plot.box(figsize = (16,5))
```

```
Out[17]: <matplotlib.axes._subplots.AxesSubplot at 0xe025285f98>
```

Train.corr():

Pandas `dataframe.corr()` is used to find the pairwise correlation of all columns in the dataframe. Any `na` values are automatically excluded. For any non-numeric data type columns in the dataframe it is ignored.



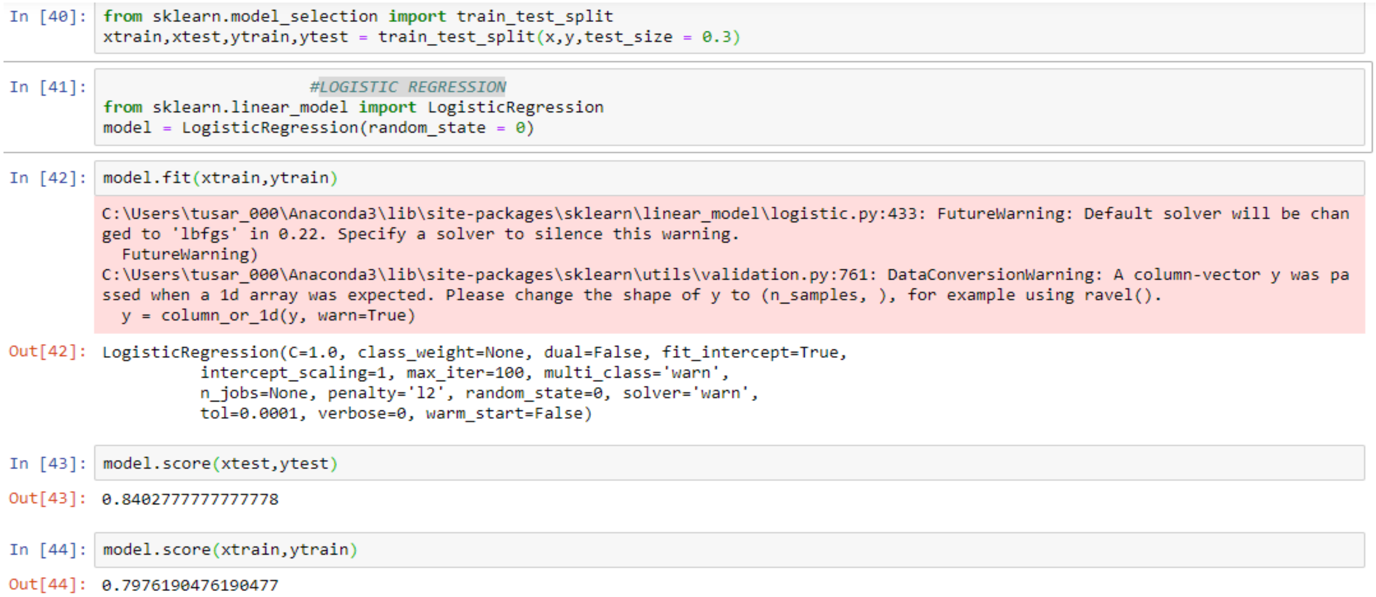
```
In [37]: train.corr()
```

Out[37]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Crec
Gender	1.000000	0.349424	0.217510	0.059245	-0.002761	0.032644	0.156170	0.098975	-0.088704	
Married	0.349424	1.000000	0.386367	0.001652	0.015674	0.036717	0.102950	0.183442	-0.107504	
Dependents	0.217510	0.386367	1.000000	0.028608	0.045754	0.131139	-0.000319	0.172780	-0.096361	
Education	0.059245	0.001652	0.028608	1.000000	-0.005085	-0.131172	-0.074498	-0.172780	-0.102168	
Self_Employed	-0.002761	0.015674	0.045754	-0.005085	1.000000	0.170785	-0.001508	0.120389	-0.034852	
ApplicantIncome	0.032644	0.036717	0.131139	-0.131172	0.170785	1.000000	-0.112588	0.495310	-0.010838	
CoapplicantIncome	0.156170	0.102950	-0.000319	-0.074498	-0.001508	-0.112588	1.000000	0.190740	-0.005773	
LoanAmount	0.098975	0.183442	0.172780	-0.172780	0.120389	0.495310	0.190740	1.000000	0.050867	
Loan_Amount_Term	-0.088704	-0.107504	-0.096361	-0.102168	-0.034852	-0.010838	-0.005773	0.050867	1.000000	
Credit_History	0.022447	0.029095	-0.026651	-0.056656	-0.023568	-0.056152	-0.008692	-0.040773	0.032937	
Property_Area	-0.000204	0.038653	0.001191	-0.055005	-0.050797	-0.053160	0.006539	-0.109685	-0.058656	
Loan_Status	0.064504	0.112321	0.035428	-0.068437	-0.034715	-0.043152	-0.049020	-0.071753	-0.007798	

Logistic regression:

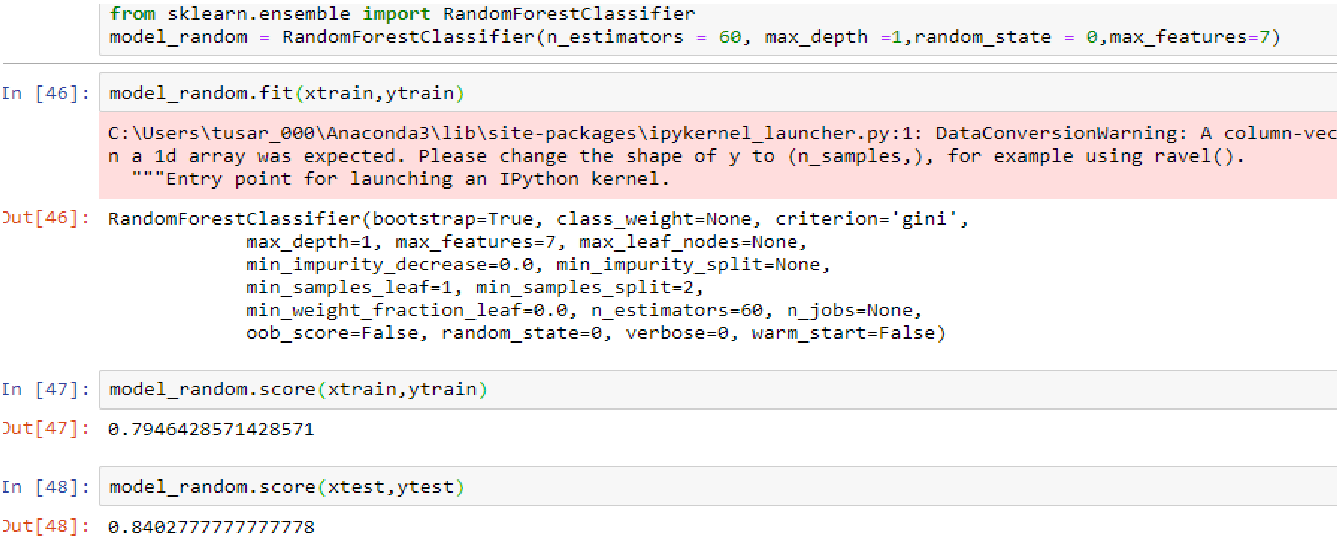
Logistic regression is a statistical method for analyzing a dataset in which there are one or more independent variables that determine an outcome. The outcome is measured with a dichotomous variable (in which there are only two possible outcomes).



```
In [40]: from sklearn.model_selection import train_test_split
         xtrain,xtest,ytrain,ytest = train_test_split(x,y,test_size = 0.3)

In [41]:                      #LOGISTIC REGRESSION
         from sklearn.linear_model import LogisticRegression
         model = LogisticRegression(random_state = 0)

In [42]: model.fit(xtrain,ytrain)

         C:\Users\tusar_000\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning: Default solver will be chan
         ged to 'lbfgs' in 0.22. Specify a solver to silence this warning.
           FutureWarning)
         C:\Users\tusar_000\Anaconda3\lib\site-packages\sklearn\utils\validation.py:761: DataConversionWarning: A column-vector y was pa
         ssed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
           y = column_or_1d(y, warn=True)

Out[42]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, max_iter=100, multi_class='warn',
                   n_jobs=None, penalty='l2', random_state=0, solver='warn',
                   tol=0.0001, verbose=0, warm_start=False)

In [43]: model.score(xtest,ytest)

Out[43]: 0.8402777777777778

In [44]: model.score(xtrain,ytrain)

Out[44]: 0.7976190476190477
```

Random Forest:

A random forest is a data construct applied to machine learning that develops large numbers of random decision trees analyzing sets of variables. This type of algorithm helps to enhance the ways that technologies analyze complex data.



```
         from sklearn.ensemble import RandomForestClassifier
         model_random = RandomForestClassifier(n_estimators = 60, max_depth =1,random_state = 0,max_features=7)

In [46]: model_random.fit(xtrain,ytrain)

         C:\Users\tusar_000\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: DataConversionWarning: A column-vec
         n a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
           """Entry point for launching an IPython kernel.

Out[46]: RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                     max_depth=1, max_features=7, 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, n_estimators=60, n_jobs=None,
                     oob_score=False, random_state=0, verbose=0, warm_start=False)

In [47]: model_random.score(xtrain,ytrain)

Out[47]: 0.7946428571428571

In [48]: model_random.score(xtest,ytest)

Out[48]: 0.8402777777777778
```

References:

[1] Ogawa, Ms Sumiko, et al. Financial Interconnectedness and Financial Sector Reforms in the Caribbean. No. 13-175. International Monetary Fund, 2013.

[2] Strahan, Philip E. "Borrower risk and the price and nonprice terms of bank loans." FRB of New York Staff Report 90 (1999).

[3] Tomar, Divya, and Sonali Agarwal. "A survey on Data Mining approaches for Healthcare." International Journal of Bio-Science and Bio-Technology 5.5 (2013): 241-266.

[4] Pulakkazhy, Sreekumar, and R. V. S. Balan. "Data mining in banking and its applications-a review." Journal of computer science 9.10 (2013): 1252.

[5] Tomar, Divya, and Sonali Agarwal. "A survey on Data Mining approaches for Healthcare." International Journal of Bio-Science and Bio-Technology 5.5 (2013): 241-266.

[6] Sharma, Poonam, and Gudla Balakrishna. "PrefixSpan: Mining Sequential Patterns by PrefixProjected Pattern." International Journal of Computer Science and Engineering Survey 2.4 (2011): 111.

[7] Chitra, K., and B. Subashini. "Data Mining Techniques and its Applications in Banking Sector." International Journal of Emerging Technology and Advanced Engineering 3.8 (2013): 219-226.

[8] Zurada, Jozef, and Martin Zurada. "How Secure Are “Good Loans”: Validating Loan-Granting Decisions And Predicting Default Rates On Consumer Loans."Review of Business Information Systems (RBIS) 6.3 (2011): 65-84.

[9] İkizler, Nazlı, and H. Altay Guvenir. "Mining interesting rules in bank loans data." Proceedings of the Tenth Turkish Symposium on Artificial Intelligence and Neural Networks. 2001.

[10] Abhijit A. Sawant and P. M. Chawan, "Comparison of Data Mining Techniques used for Financial Data Analysis," International Journal of Emerging Technology and Advanced Engineering, june 2013.

[11] Sudhakar M and Dr. C. V. Krishna Reddy, "CREDIT EVALUATION MODEL OF LOAN PROPOSALS FOR BANKS USING DATA MINING," International Journal of Latest Research in Science and Technology, pp. 126-131, july 2014.

[12] R.Mahammad Shafi, A Tool for Enhancing Business Process in Banking Sector, 3rd ed.: International Journal of Scientific & Engineering Research, 2012.