Summer Internship Programme

Henry Harvin Education India LLP
Sector-2, Noida, U.P.-201306

Project Title – **LOAN PREDICTION**

Mentor Name: Ms. Pooja Gupta (Senior Consultant)

Name: JUJARAY PRASANTH

Course: Summer Internship Programme (SIP) Python

Batch: Jun-Jul 2019

Job: Business Analyst Associate (Intern)

Institution: Lovely professional university,phagwara

DECLARATION

I here by declare that the project report entitled “**Loan prediction**” submitted by me to **HENRY HARVIN EDUCATION INDIA** is a record of bonafide 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.

DATE: 28 Jul 2019
JUJARAY PRASANTH
SIP – Python

problem statement:-

Some anonymous person is approaching for the bank loan . now the manager of the bank is calling data scientist and saying him if he want to sanction loan to him as a data scientist what were your predictions on him to sanction the loan.

data scientist has consider many variables of the person on these basis he use to predict

1)loan id .	represents the person id
2)gender	represents the gender of the person
3)married	represents the married status of the person
4)education	represents the qualification of the person
5)self employed	represents is he employed or not
6)applicant income	represents the income of the person
7)loan amount	represents any any previous loan
8)credit history	represents is their any credit history previouster
9)dependents	represents the dependent of the person
10)coapplicants income	represents the income of co applicant
11)loan amount term	represent the term of the loan
12)property area	represents the property area of the person
13)loan status	represents the status of the loan

Here iam applying the logistic regression method for prediction because of we have to predict yes or no situation. At same time we are applying random forest for loan prediction. depending on the accuaracy levels of both method we are choosing the best method for the prediction.

In [6]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import sklearn
```

import numpy as np

iam importing numerical python library making np as the object

import pandas as pd

iam importing pandas library for doing operations on colums

import seaborn as sns

iam importing seaborn library for the predictions of the graphs

import matplotlib.pyplot as plt

iam importing matplot library for doing statistical operations

import sklearn

iam using for random forest

```
In [8]: train = pd.read_csv("E:\\SIP\\PY DATA\\train.csv")
        test = pd.read_csv("E:\\SIP\\PY DATA\\test.csv")
```

train = pd.read_csv("E:\\SIP\\PY DATA\\train.csv")

test = pd.read_csv("E:\\SIP\\PY DATA\\test.csv")

Here we are reading data from the particular location by using function pd.read()

train.describe()

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

Out[10]:

	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

Pandas describe() is used to view some basic statistical details like percentile, mean, std etc. of a data frame or a series of numeric values. When this method is applied to a series of string

train.head()

here head is the function used for giving first top 5 values by defalut

```
In [11]: train.head()
```

Out[11]:

ender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	L
Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0	Urban	
Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	
Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	
Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	
Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	

train_original = train.copy()

test_original = test.copy()

```
In [12]: print(train.shape)
         print(test.shape)

         ### Univariate Analysis

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

here iam training the data we are storing the variable in train data frame

through which we are converting the categorical values in to numerical values by using test function and printing the values of test and train.

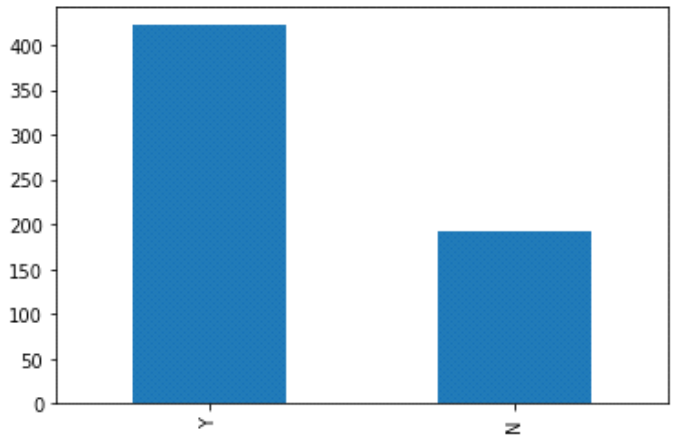
```
In [13]: train['Loan_Status'].value_counts()

Out[13]: Y    422
         N    192
         Name: Loan_Status, dtype: int64
```

we are how many variants are applied for loan and of integer type

```
In [14]: train['Loan_Status'].value_counts().plot.bar()

Out[14]: <matplotlib.axes._subplots.AxesSubplot at 0xb9966d8>
```



here we are plotting the bar plot on loan status of the variants

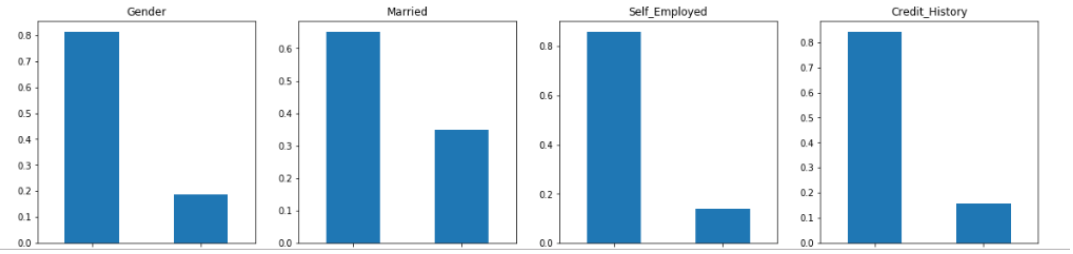
```
plt.subplot(241)
train['Gender'].value_counts(normalize=True).plot.bar(figsize=(20,10), title= 'Gender')

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()
```



here we are plotting the bar plot on gender,married,self employed,credit history by using the function

plt.subplot(242)

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

and so on ...

In [16]:
```
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[16]: `<matplotlib.axes._subplots.AxesSubplot at 0xbe76e10>`

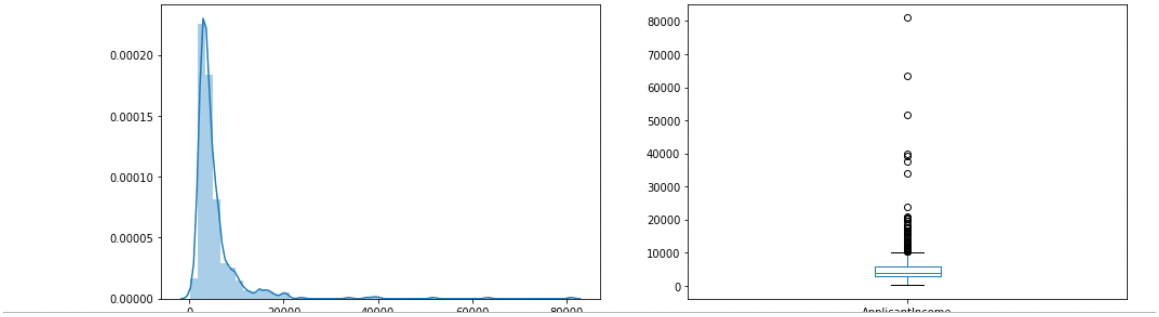
same thing done on the dependendts,property_area,education

```
In [17]: plt.figure(3)

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

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

Out[17]: <matplotlib.axes._subplots.AxesSubplot at 0xbc3ed68>
```



Here we are plotting the blot plot of the applicants income of the figure

size (16,5) by using

plt.subplot(121)

sns.distplot(train['ApplicantIncome'])

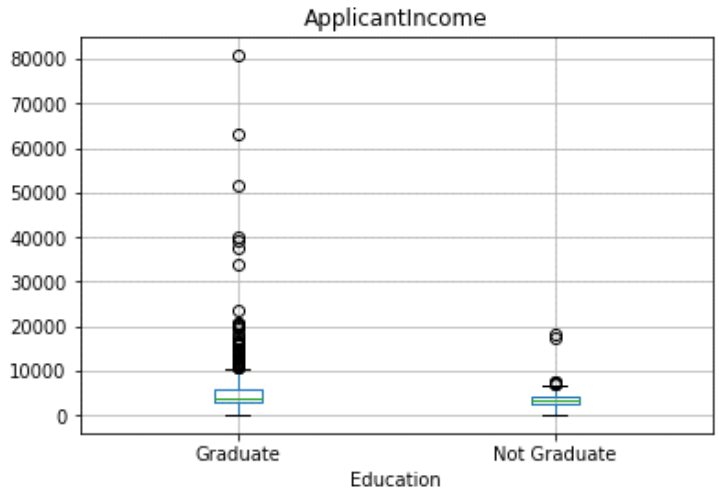
plt.subplot(122)

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

```
In [18]: plt.figure(4)
         train.boxplot(column = 'ApplicantIncome', by = 'Education')
         plt.suptitle("")

Out[18]: Text(0.5, 0.98, '')

         <Figure size 432x288 with 0 Axes>
```



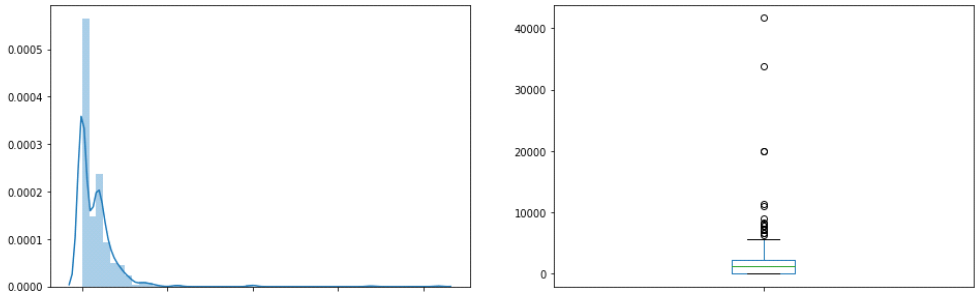
in the above figure we are plotting the box plot based on the application income and education .

```
In [19]: plt.figure(5)

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

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

Out[19]: <matplotlib.axes._subplots.AxesSubplot at 0xccf9668>
```



as same as above figure we are plotting box plot on the applicants co applicant income

we are plotting the box plot on loan amount which is of figure size (16,5)

```
In [21]: train.columns

Out[21]: Index(['Loan_ID', 'Gender', 'Married', 'Dependents', 'Education',
                'Self_Employed', 'ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
                'Loan_Amount_Term', 'Credit_History', 'Property_Area', 'Loan_Status'],
               dtype='object')
```

here it represents particular columns which are stored in train data set and gives all the cloumns we are importing Label Encoder() function from the sklearn It can also be used to transform non-numerical labels (as long as they are hashable and comparable) to numerical labels.ibrary This is a categorical feature with numeric values. If I give it to the model as it is, the model will treat it as continuous variable,

```
In [23]: train['Gender'].unique()

Out[23]: array(['Male', 'Female', nan], dtype=object)
```

unique() function is used to know the unique values of the varible .in gender we are having male ,female,nan

```
In [24]: train.isna().sum()

Out[24]: 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
```

Pandas dataframe.isna() function is used to detect missing values.

```
In [25]: train = train[~train['Gender'].isna()]
```

```
In [26]: train = train[~train['Dependents'].isna()]
         train = train[~train['Self_Employed'].isna()]
         train = train[~train['LoanAmount'].isna()]
         train = train[~train['Credit_History'].isna()]
         train = train[~train['Loan_Amount_Term'].isna()]
```

```
In [27]: train.isna().sum()

Out[27]: 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
```

here we isna function ()returns the null values

```
In [28]: from sklearn import preprocessing
         le = preprocessing.LabelEncoder()
         le.fit(train['Gender'])
         x=le.transform(train['Gender'])
         train['Gender'] = x

In [29]: le.fit(train['Married'])
         x=le.transform(train['Married'])
         train['Married'] = x

In [30]: le.fit(train['Dependents'])
         x=le.transform(train['Dependents'])
         train['Dependents'] = x

In [31]: le.fit(train['Education'])
         x=le.transform(train['Education'])
         train['Education'] = x

In [32]: le.fit(train['Self_Employed'])
         x=le.transform(train['Self_Employed'])
         train['Self_Employed'] = x

In [33]: le.fit(train['Property_Area'])
         x=le.transform(train['Property_Area'])
         train['Property_Area'] = x
```

Here we are fitting the data in train data set of various riables of gender,married,dependents,education,self employed by using the lebel encoder we are converting the caterical values in to the numerical value by using the function

le.fit(train['Married'])

x=le.transform(train['Married'])

train['Married'] = x

```
In [55]: train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 480 entries, 1 to 613
Data columns (total 13 columns):
Loan_ID              480 non-null object
Gender               480 non-null int32
Married              480 non-null int32
Dependents           480 non-null int32
Education            480 non-null int32
Self_Employed        480 non-null int32
ApplicantIncome      480 non-null int64
CoapplicantIncome    480 non-null int64
LoanAmount           480 non-null int64
Loan_Amount_Term     480 non-null int64
Credit_History       480 non-null int64
Property_Area        480 non-null int32
Loan_Status          480 non-null int32
dtypes: int32(7), int64(5), object(1)
memory usage: 39.4+ KB
```

We are getting entire information by using the above command train.info()

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

Out[37]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
1	LP001003	1	1	1	0	0	4583	1508.0	128.0	360.0	1.0
2	LP001005	1	1	0	0	1	3000	0.0	66.0	360.0	1.0
3	LP001006	1	1	0	1	0	2583	2358.0	120.0	360.0	1.0
4	LP001008	1	0	0	0	0	6000	0.0	141.0	360.0	1.0
5	LP001011	1	1	2	0	1	5417	4196.0	267.0	360.0	1.0

By using train.head() we are getting first five values of variables by defaults

```
In [39]: train.corr()
```

Out[39]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Cred
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	

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 [40]: x = train[[ 'Gender', 'Married', 'Dependents', 'Education','Self_Employed', 'ApplicantIncome', 'CoapplicantIncome', 'LoanAmount'
```

```
In [41]: y = train[['Loan_Status']]
```

Here y is the target variable that is "loan status" which we want to predict and x is variables which we have to store

```
In [42]: from sklearn.model_selection import train_test_split
         xtrain,xtest,ytrain,ytest = train_test_split(x,y,test_size = 0.3)

In [43]: from sklearn.linear_model import LogisticRegression
         model = LogisticRegression(random_state = 0)

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

         C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning: Default solver will be changed
         to 'lbfgs' in 0.22. Specify a solver to silence this warning.
           FutureWarning)
```

we are applying logistic regression test on the model and we are fitting the data in

x train ,y train and for this we are importing import test_split from the sklearn.model

```
In [45]: model.score(xtest,ytest)

Out[45]: 0.8125

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

Out[47]: 0.8125

In [51]: from sklearn.ensemble import RandomForestClassifier
```

when we are applying logistic regression and we are getting accuracy rate as 0.8125 .

```
Out[52]: 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 [53]: model_random.score(xtrain,ytrain)

Out[53]: 0.8065476190476191

In [54]: model_random.score(xtest,ytest)

Out[54]: 0.8125
```

we are using random forest classfier for the prediction we are getting accuarcy rate as 0.8125 by using the functions

from sklearn.ensemble import RandomForestClassifier

model_random = RandomForestClassifier(n_estimators = 60, max_depth =1,random_state = 0,max_features=7)

model_random.score(xtrain,ytrain)

model_random.score(xtrain,ytrain)