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: Saksham Sood
Course: Summer Internship Programme (SIP) Python
Batch: Jun-Jul 2019
Job: Business Analyst Associate (Intern)
Institution: Lovely Professional University, Jalandhar, Punjab.

<u>DECLARATION</u>

I hereby 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
Saksham Sood
SIP – Python

Acknowledgements

In the accomplishment of this project successfully, many people have best owned upon me their blessings and the heart pledged support, this time I am utilizing to thank all the people who have been concerned with this project. I would like to thank my teachers MR. DHIRAJ UPADHYAYA and MR. ANIL JADON whose valuable guidance has been the ones that helped me patch this project and make it full proof success.

Their suggestions and instructions have served as the major contributor towards the completion of the project. I would like to thank my mentor MS. POOJA GUPTA for giving me this golden opportunity.

Then I would like to thank my parents and friends who have helped me with their valuable suggestions and guidance has been very helpful in various phases of the completion of the project. Last but not the least I would like to thank my batchmates who have helped me a lot.

SAKSHAM SOOD

SIP-Python

Introduction

Predicting the outcome of a loan is a recurrent, crucial and difficult issue in insurance and banking. The objective of our project is to predict whether a loan will default or not based on objective financial data only.

This is a Python-based Project. This project was created via Spyder 3.3.5. IDE (Integrated Development Environment) using Python 3.7.3 and Ipython Console 7.4.0. The final outcome of this project is saved in a Jupyter Notebook v7.8.0. The libraries of python used in this project are:

1. NumPy
2. Pandas
3. Matplotlib
4. Seaborn
5. Stats models
6. Sci-kit Learn

This project is based on a data set provided by the teachers via GITHUB.

Since the objective is to predict the outcome from the information gathered at the signature of the loan, we cannot use the data concerning the history of payments or the current situation of a loan.

Excluding features for which the information is incomplete, or uninformative, we get a total of 19 features, that cover personal information (credit grade, income, housing status, ...) and credit information (amount, interest rates, ...). Accuracy is not well-suited for our problem. The unbalance of the classes would lead an algorithm to never predict a default. F1-score allows us to quantify a good prediction on both precision and recall.

Loan Prediction

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.

Data

1. Variable Description
2. Loan ID - Unique Loan ID
3. Gender - Male/ Female
4. Married - Applicant married (Y/N)
5. Dependents - Number of dependents
6. Education - Applicant Education (Graduate/ Undergraduate)
7. Self Employed - Self-employed (Y/N)
8. Applicant Income - Applicant income
9. Co-applicant Income – Co-applicant income
10. Loan Amount - Loan amount in thousands
11. Loan Amount Term - Term of loan in months
12. Credit History - credit history meets guidelines
13. Property Area - Urban/ Semi Urban/ Rural
14. Loan Status - Loan approved (Y/N)

CODE: -

In [1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```python
test_url = 'C:/Users/Saksham/Documents/Study Material/Internship/Finalprojects_DS-master/Loan_Prediction/test.csv'
train_url = 'C:/Users/Saksham/Documents/Study Material/Internship/Finalprojects_DS-master/Loan_Prediction/train.csv'
```

In [3]:
```python
train = pd.read_csv(train_url)
test = pd.read_csv(test_url)
```

In [4]:
```python
train.head()
```

Out[4]:

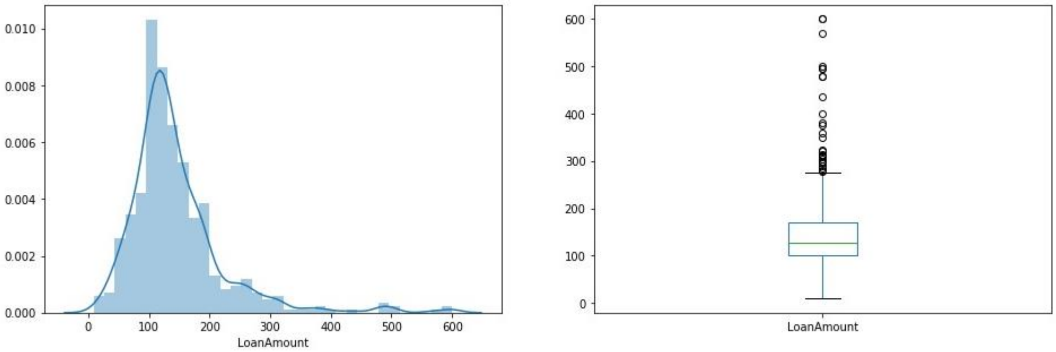
	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_Histor
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.(
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.(
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.(
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.(
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.(

In [5]:
```python
plt.figure(6)

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

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

Out[5]: <matplotlib.axes._subplots.AxesSubplot at 0x20c8971f978>

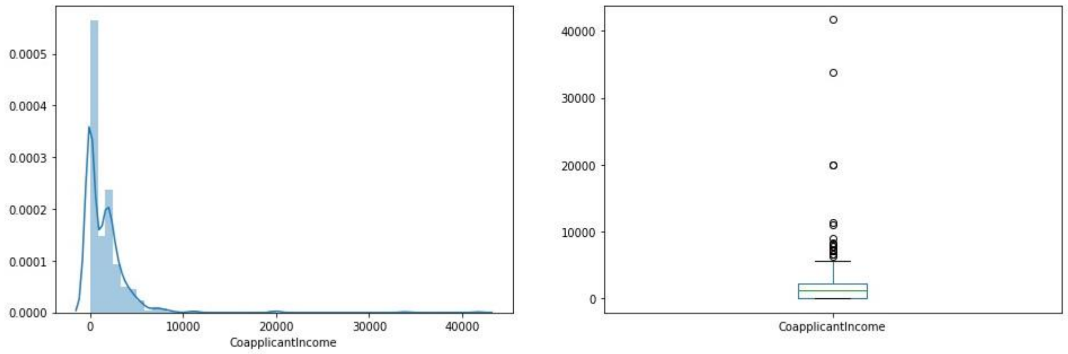


In [6]:
```python
plt.figure(5)

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

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

Out[6]: <matplotlib.axes._subplots.AxesSubplot at 0x20c8caa3eb8>



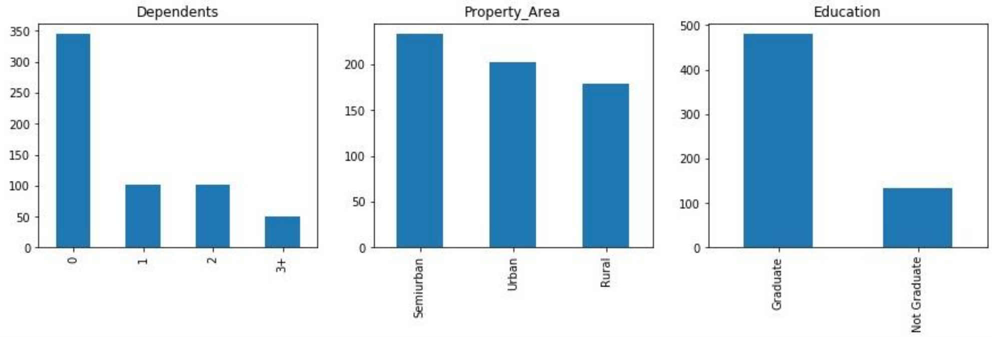
In [7]:
```python
plt.figure(2)

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[7]: <matplotlib.axes._subplots.AxesSubplot at 0x20c8c9fed30>



In [8]:
```python
train.corr()
```

Out[8]:

	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

In [9]:
```python
train.columns
```

Out[9]:
```
Index(['Loan_ID', 'Gender', 'Married', 'Dependents', 'Education',
       'Self_Employed', 'ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
       'Loan_Amount_Term', 'Credit_History', 'Property_Area', 'Loan_Status'],
      dtype='object')
```

In [10]:
```python
from sklearn import preprocessing
le = preprocessing.LabelEncoder()
```

In [11]:
```python
train['Gender'].unique()
```

Out[11]: array(['Male', 'Female', nan], dtype=object)

In [12]:
```
train.isna().sum()
```
Out[12]:
```
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
```
In [13]:
```
train = train[~train['Gender'].isna()]
```
In [14]:
```
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 [15]:
```
train.isna().sum()
```
Out[15]:
```
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
```
In [16]:
```
from sklearn import preprocessing
le = preprocessing.LabelEncoder()
le.fit(train['Gender'])
x=le.transform(train['Gender'])
train['Gender'] = x
```
In [17]:
```
le.fit(train['Married'])
x=le.transform(train['Married'])
train['Married'] = x
```
In [18]:
```
le.fit(train['Dependents'])
x=le.transform(train['Dependents'])
train['Dependents'] = x
```
In [19]:
```
le.fit(train['Education'])
x=le.transform(train['Education'])
train['Education'] = x
```
In [20]:
```
le.fit(train['Self_Employed'])
x=le.transform(train['Self_Employed'])
train['Self_Employed'] = x
```
In [21]:
```
le.fit(train['Property_Area'])
x=le.transform(train['Property_Area'])
train['Property_Area'] = x
```
In [22]:
```
le.fit(train['Loan_Status'])
x=le.transform(train['Loan_Status'])
train['Loan_Status'] = x
```

In [21]:
```
le.fit(train['Property_Area'])
x=le.transform(train['Property_Area'])
train['Property_Area'] = x
```

In [22]:
```
le.fit(train['Loan_Status'])
x=le.transform(train['Loan_Status'])
train['Loan_Status'] = x
```

In [23]:
```
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 float64
LoanAmount           480 non-null float64
Loan_Amount_Term     480 non-null float64
Credit_History       480 non-null float64
Property_Area        480 non-null int32
Loan_Status          480 non-null int32
dtypes: float64(4), int32(7), int64(1), object(1)
memory usage: 39.4+ KB
```

In [24]:
```
train.head()
```

Out[24]:

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

In [25]:
```
cols = ['ApplicantIncome','CoapplicantIncome','LoanAmount','Loan_Amount_Term','Credit_History']
train[cols] = train[cols].applymap(np.int64)
```

In [26]:
```
train.corr()
```

Out[26]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Cr
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	

In [27]:
```
x = train[[ 'Gender', 'Married', 'Dependents', 'Education','Self_Employed', 'ApplicantIncome', 'CoapplicantIncome', 'LoanAmount','Loan_Amount_T
```

In [28]:
```
y = train[['Loan_Status']]
```

In [29]:
```
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest = train_test_split(x,y,test_size = 0.3)
```

In [30]:
```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression(random_state = 0)
```

In [31]:
```
model.fit(xtrain,ytrain)
```

```
C:\Users\Saksham\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)
C:\Users\Saksham\Anaconda3\lib\site-packages\sklearn\utils\validation.py:761: DataConversionWarning: A column-vector y was passed 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[31]:
```
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 [32]:
```
model.score(xtest,ytest)
```

Out[32]:
```
0.8263888888888888
```

In [33]:
```
model.score(xtrain,ytrain)
```

Out[33]:
```
0.7976190476190477
```

In [34]:
```
from sklearn.ensemble import RandomForestClassifier
model_random = RandomForestClassifier(n_estimators = 60, max_depth =1,random_state = 0,max_features=7)
```

In [35]:
```
model_random.fit(xtrain,ytrain)
```

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

Out[35]:
```
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 [36]:
```
model_random.score(xtrain,ytrain)
```

Out[36]:
```
0.7946428571428571
```

In [37]:
```
model_random.score(xtest,ytest)
```

Out[37]:
```
0.8402777777777778
```

Accuracy: - 79.46%

Cross-Validation Score: - 84.02%