Loan Prediction1

January 30, 2022

1 Loan Eligibility Prediction

Problem

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, identify the customers segments, those are eligible for loan amount so that they can specifically target these customers.

Data:

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 loan in months - Credit_History: credit history meets guidelines - Property_Area: Urban/ Semi Urban/ Rural - Loan_Status: Loan approved (Y/N)

1.1 1. Load Libraries

[1]:
```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
%matplotlib inline
import seaborn as sns
```

1.2 2. Load Data

[2]:
```
data_train = pd.read_csv('train.csv')
data_test = pd.read_csv('test.csv')
```

[3]:
```
# check the shape of data
data_train.shape
```

[3]:
```
(614, 13)
```

[4]:
```
# training data infotmation
data_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Loan_ID            614 non-null    object 
 1   Gender             601 non-null    object 
 2   Married            611 non-null    object 
 3   Dependents         599 non-null    object 
 4   Education          614 non-null    object 
 5   Self_Employed      582 non-null    object 
 6   ApplicantIncome    614 non-null    int64  
 7   CoapplicantIncome  614 non-null    float64
 8   LoanAmount         592 non-null    float64
 9   Loan_Amount_Term   600 non-null    float64
 10  Credit_History     564 non-null    float64
 11  Property_Area      614 non-null    object 
 12  Loan_Status        614 non-null    object 
dtypes: float64(4), int64(1), object(8)
memory usage: 62.5+ KB
```

[5]:
```
# check test data shape
data_test.shape
```

[5]:
```
(367, 12)
```

[6]:
```
# displaying first few rows
data_train.head()
```

[6]:
```
    Loan_ID Gender Married Dependents     Education Self_Employed  \
0  LP001002   Male      No          0      Graduate            No   
1  LP001003   Male     Yes          1      Graduate            No   
2  LP001005   Male     Yes          0      Graduate           Yes   
3  LP001006   Male     Yes          0  Not Graduate            No   
4  LP001008   Male      No          0      Graduate            No   

   ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
0             5849                0.0         NaN             360.0   
1             4583             1508.0       128.0             360.0   
2             3000                0.0        66.0             360.0   
3             2583             2358.0       120.0             360.0   
4             6000                0.0       141.0             360.0   

   Credit_History Property_Area Loan_Status  
0             1.0         Urban           Y  
1             1.0         Rural           N  
2             1.0         Urban           Y  
```

```
3             1.0        Urban           Y
4             1.0        Urban           Y
```

[7]:
```python
data_test.head()
```

[7]:
```
    Loan_ID Gender Married Dependents     Education Self_Employed  \
0  LP001015   Male     Yes          0      Graduate            No
1  LP001022   Male     Yes          1      Graduate            No
2  LP001031   Male     Yes          2      Graduate            No
3  LP001035   Male     Yes          2      Graduate            No
4  LP001051   Male      No          0  Not Graduate            No

   ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
0             5720                  0       110.0             360.0
1             3076               1500       126.0             360.0
2             5000               1800       208.0             360.0
3             2340               2546       100.0             360.0
4             3276                  0        78.0             360.0

   Credit_History Property_Area
0             1.0         Urban
1             1.0         Urban
2             1.0         Urban
3             NaN         Urban
4             1.0         Urban
```

1.3 3. Descriptive Statistics

[8]:
```python
data_train.describe() #descriptive statistics
```

[8]:
```
       ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
count       614.000000         614.000000  592.000000         600.00000
mean       5403.459283        1621.245798  146.412162         342.00000
std        6109.041673        2926.248369   85.587325          65.12041
min         150.000000           0.000000    9.000000          12.00000
25%        2877.500000           0.000000  100.000000         360.00000
50%        3812.500000        1188.500000  128.000000         360.00000
75%        5795.000000        2297.250000  168.000000         360.00000
max       81000.000000       41667.000000  700.000000         480.00000

       Credit_History
count      564.000000
mean         0.842199
std          0.364878
min          0.000000
25%          1.000000
50%          1.000000
```

```
75%          1.000000
max          1.000000
```

[9]:
```
data_train.describe(include=['object']) # categorical data
```

[9]:
```
         Loan_ID Gender Married Dependents Education Self_Employed  \
count        614    601     611        599       614           582
unique       614      2       2          4         2             2
top     LP001002   Male     Yes          0  Graduate            No
freq           1    489     398        345       480           500

       Property_Area Loan_Status
count            614         614
unique             3           2
top        Semiurban           Y
freq             233         422
```

[10]:
```
data_test.describe()
```

[10]:
```
       ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
count       367.000000         367.000000  362.000000        361.000000
mean       4805.599455        1569.577657  136.132597        342.537396
std        4910.685399        2334.232099   61.366652         65.156643
min           0.000000           0.000000   28.000000          6.000000
25%        2864.000000           0.000000  100.250000        360.000000
50%        3786.000000        1025.000000  125.000000        360.000000
75%        5060.000000        2430.500000  158.000000        360.000000
max       72529.000000       24000.000000  550.000000        480.000000

       Credit_History
count      338.000000
mean         0.825444
std          0.380150
min          0.000000
25%          1.000000
50%          1.000000
75%          1.000000
max          1.000000
```

1.4 4. Finding Missing Values

[11]:
```
data_train.isnull().sum()
```

[11]:
```
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
```

[12]:
```
data_test.isnull().sum()
```

[12]:
```
Loan_ID               0
Gender               11
Married               0
Dependents           10
Education             0
Self_Employed        23
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount            5
Loan_Amount_Term      6
Credit_History       29
Property_Area         0
dtype: int64
```

[13]:
```
def get_combined_data():
    train = pd.read_csv('train.csv')
    test = pd.read_csv('test.csv')
    target = train.Loan_Status
    train.drop('Loan_Status', 1, inplace=True)
    combined = train.append(test)
    combined.reset_index(inplace=True)
    combined.drop(['index', 'Loan_ID'], inplace=True, axis=1)
    return combined
```

[14]:
```
combined = get_combined_data()
combined.head()
```

```
C:\Users\Dileep\AppData\Local\Temp/ipykernel_14944/3992989479.py:5:
FutureWarning: In a future version of pandas all arguments of DataFrame.drop
except for the argument 'labels' will be keyword-only
  train.drop('Loan_Status', 1, inplace=True)
```

[14]:
```
  Gender Married Dependents     Education Self_Employed  ApplicantIncome  \
0   Male      No          0      Graduate            No             5849
```

```
1   Male      Yes            1      Graduate             No             4583
2   Male      Yes            0      Graduate            Yes             3000
3   Male      Yes            0  Not Graduate             No             2583
4   Male       No            0      Graduate             No             6000

   CoapplicantIncome  LoanAmount  Loan_Amount_Term  Credit_History  \
0                0.0         NaN             360.0             1.0
1             1508.0       128.0             360.0             1.0
2                0.0        66.0             360.0             1.0
3             2358.0       120.0             360.0             1.0
4                0.0       141.0             360.0             1.0

  Property_Area
0         Urban
1         Rural
2         Urban
3         Urban
4         Urban
```

[15]:
```
combined.shape
```

[15]:
```
(981, 11)
```

[16]:
```
combined.columns.values
```

[16]:
```
array(['Gender', 'Married', 'Dependents', 'Education', 'Self_Employed',
       'ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
       'Loan_Amount_Term', 'Credit_History', 'Property_Area'],
      dtype=object)
```

[17]:
```
combined.describe()
```

[17]:
```
       ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
count       981.000000         981.000000  954.000000        961.000000
mean       5179.795107        1601.916330  142.511530        342.201873
std        5695.104533        2718.772806   77.421743         65.100602
min           0.000000           0.000000    9.000000          6.000000
25%        2875.000000           0.000000  100.000000        360.000000
50%        3800.000000        1110.000000  126.000000        360.000000
75%        5516.000000        2365.000000  162.000000        360.000000
max       81000.000000       41667.000000  700.000000        480.000000

       Credit_History
count      902.000000
mean         0.835920
std          0.370553
min          0.000000
```

```
25%          1.000000
50%          1.000000
75%          1.000000
max          1.000000
```

[18]:
```
combined.isnull().sum()
```

[18]:
```
Gender                24
Married                3
Dependents            25
Education              0
Self_Employed         55
ApplicantIncome        0
CoapplicantIncome      0
LoanAmount            27
Loan_Amount_Term      20
Credit_History        79
Property_Area          0
dtype: int64
```

1.5 5. Missing Data Imputation

[19]:
```
combined['Gender'].value_counts()
```

[19]:
```
Male      775
Female    182
Name: Gender, dtype: int64
```

[20]:
```
def impute_gender():
    global combined
    combined['Gender'].fillna('Male', inplace=True)
```

[21]:
```
combined['Married'].value_counts()
```

[21]:
```
Yes    631
No     347
Name: Married, dtype: int64
```

[22]:
```
def impute_marital_status():
    global combined
    combined['Married'].fillna('Yes', inplace=True)
```

[23]:
```
combined['Self_Employed'].value_counts()
```

[23]:
```
No     807
Yes    119
Name: Self_Employed, dtype: int64
```

[24]:
```
def impute_employment():
    global combined
    combined['Self_Employed'].fillna('No',inplace=True)
```

[25]:
```
combined['LoanAmount'].skew()
```

[25]:
```
2.714035799071379
```

[26]:
```
def impute_loan_amount():
    global combined
    combined['LoanAmount'].fillna(combined['LoanAmount'].median(), inplace=True)
```

[27]:
```
combined['Credit_History'].value_counts()
```

[27]:
```
1.0    754
0.0    148
Name: Credit_History, dtype: int64
```

[28]:
```
def impute_credit_history():
    global combined
    combined['Credit_History'].fillna(2, inplace=True)
```

[29]:
```
impute_gender() # calling function
```

[30]:
```
impute_marital_status()
```

[31]:
```
impute_employment()
```

[32]:
```
impute_loan_amount()
```

[33]:
```
impute_credit_history()
```

[34]:
```
combined.isnull().sum()
```

[34]:
```
Gender                0
Married               0
Dependents           25
Education             0
Self_Employed         0
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount            0
Loan_Amount_Term     20
Credit_History        0
Property_Area         0
dtype: int64
```

[35]:
```
combined.head()
```

[35]:
```
  Gender Married Dependents     Education Self_Employed  ApplicantIncome  \
0   Male      No          0      Graduate            No             5849
1   Male     Yes          1      Graduate            No             4583
2   Male     Yes          0      Graduate           Yes             3000
3   Male     Yes          0  Not Graduate            No             2583
4   Male      No          0      Graduate            No             6000

   CoapplicantIncome  LoanAmount  Loan_Amount_Term  Credit_History  \
0                0.0       126.0             360.0             1.0
1             1508.0       128.0             360.0             1.0
2                0.0        66.0             360.0             1.0
3             2358.0       120.0             360.0             1.0
4                0.0       141.0             360.0             1.0

  Property_Area
0         Urban
1         Rural
2         Urban
3         Urban
4         Urban
```

[36]:
```
def process_gender():
    global combined
    combined['Gender'] = combined['Gender'].map({'Male':1,'Female':0})
```

[37]:
```
def process_marital_status():
    global combined
    combined['Married'] = combined['Married'].map({'Yes':1,'No':0})
```

[38]:
```
def process_dependents():
    global combined
    combined['Singleton'] = combined['Dependents'].map(lambda d: 1 if d=='1'␣
↪else 0)
    combined['Small_Family'] = combined['Dependents'].map(lambda d: 1 if d=='2'␣
↪else 0)
    combined['Large_Family'] = combined['Dependents'].map(lambda d: 1 if␣
↪d=='3+' else 0)
    combined.drop(['Dependents'], axis=1, inplace=True)
```

[39]:
```
# 1-> Graduate, 0-> Not graduate
def process_education():
    global combined
    combined['Education'] = combined['Education'].map({'Graduate':1,'Not␣
↪Graduate':0})
```

[40]:
```python
# 1-> Self Employed, 0-> Not Employed
def process_employment():
    global combined
    combined['Self_Employed'] = combined['Self_Employed'].map({'Yes':1,'No':0})
```

[41]:
```python
def process_income():
    global combined
    combined['Total_Income'] = combined['ApplicantIncome'] +␣
↪combined['CoapplicantIncome']
    combined.drop(['ApplicantIncome','CoapplicantIncome'], axis=1, inplace=True)
```

[42]:
```python
def process_loan_amount():
    global combined
    combined['Debt_Income_Ratio'] = combined['Total_Income']/
↪combined['LoanAmount']
```

[43]:
```python
combined['Loan_Amount_Term'].value_counts()
```

[43]:
```
360.0    823
180.0     66
480.0     23
300.0     20
240.0      8
84.0       7
120.0      4
60.0       3
36.0       3
12.0       2
350.0      1
6.0        1
Name: Loan_Amount_Term, dtype: int64
```

[44]:
```python
approved_term = data_train[data_train['Loan_Status']=='Y']['Loan_Amount_Term'].
↪value_counts()
unapproved_term =␣
↪data_train[data_train['Loan_Status']=='N']['Loan_Amount_Term'].value_counts()
df = pd.DataFrame([approved_term,unapproved_term])
df.index = ['Approved','Unapproved']
```

[45]:
```python
df
```

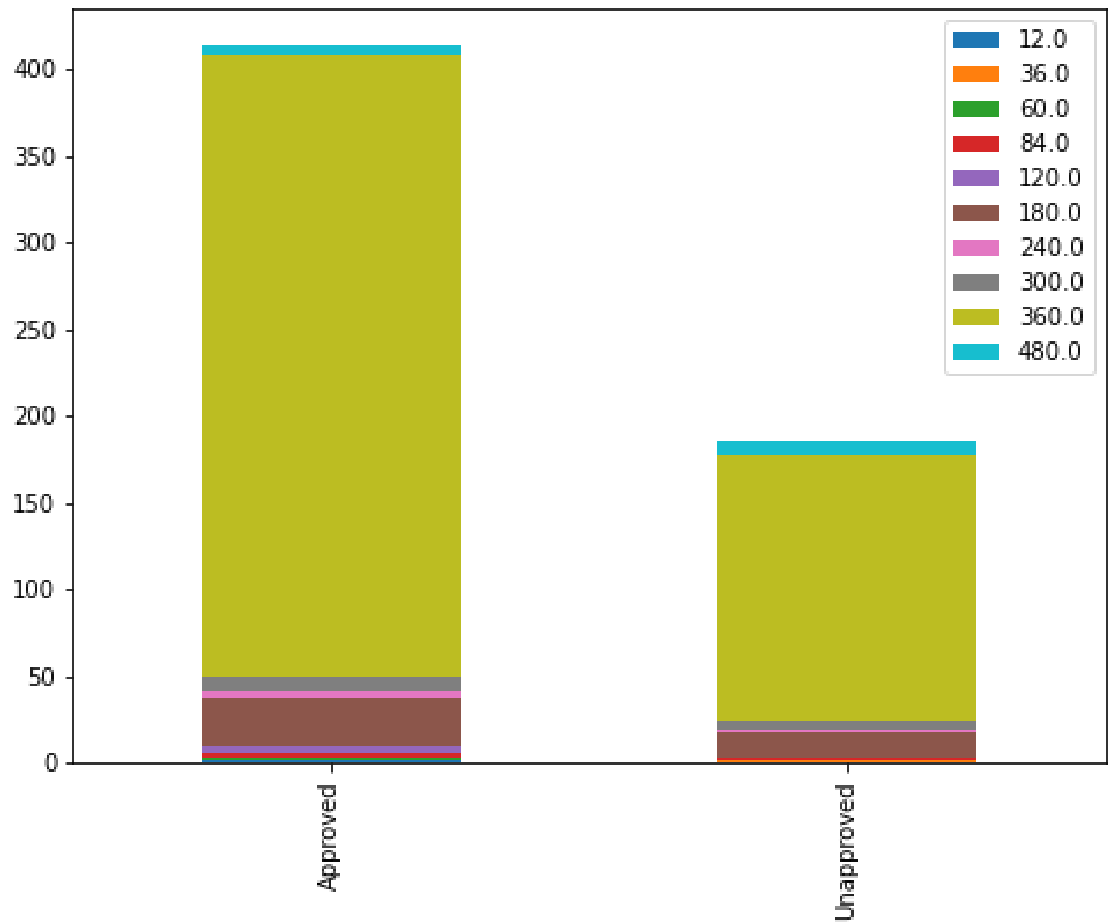
[45]:
```
            12.0  36.0  60.0  84.0  120.0  180.0  240.0  300.0  360.0  \
Approved     1.0   NaN   2.0   3.0    3.0   29.0    3.0    8.0  359.0
Unapproved   NaN   2.0   NaN   1.0    NaN   15.0    1.0    5.0  153.0

            480.0
Approved      6.0
```

```
Unapproved    9.0
```

[46]:
```
df.plot(kind='bar', stacked=True, figsize=(8,6))
```

[46]:
```
<AxesSubplot:>
```



[47]:
```
def process_loan_term():
    global combined
    combined['Very_Short_Term'] = combined['Loan_Amount_Term'].map(lambda t: 1␣
↪if t<=60 else 0)
    combined['Short_Term'] = combined['Loan_Amount_Term'].map(lambda t: 1 if␣
↪t>60 and t<180 else 0)
    combined['Long_Term'] = combined['Loan_Amount_Term'].map(lambda t: 1 if␣
↪t>=180 and t<=300  else 0)
    combined['Very_Long_Term'] = combined['Loan_Amount_Term'].map(lambda t: 1␣
↪if t>300 else 0)
    combined.drop('Loan_Amount_Term', axis=1, inplace=True)
```

[48]:
```
def process_credit_history():
    global combined
    combined['Credit_History_Bad'] = combined['Credit_History'].map(lambda c: 1␣
↪if c==0 else 0)
    combined['Credit_History_Good'] = combined['Credit_History'].map(lambda c:␣
↪1 if c==1 else 0)
    combined['Credit_History_Unknown'] = combined['Credit_History'].map(lambda␣
↪c: 1 if c==2 else 0)
    combined.drop('Credit_History', axis=1, inplace=True)
```

[49]:
```
def process_property():
    global combined
    property_dummies = pd.get_dummies(combined['Property_Area'],
                                      prefix='Property')
    combined = pd.concat([combined, property_dummies], axis=1)
    combined.drop('Property_Area', axis=1, inplace=True)
```

[50]:
```
process_gender() # calling method
```

[51]:
```
process_marital_status()
```

[52]:
```
process_dependents()
```

[53]:
```
process_education()
```

[54]:
```
process_employment()
```

[55]:
```
process_income()
```

[56]:
```
process_loan_amount()
```

[57]:
```
process_loan_term()
```

[58]:
```
process_credit_history()
```

[59]:
```
process_property()
```

[60]:
```
combined.isnull().sum()
```

[60]:
```
Gender                      0
Married                     0
Education                   0
Self_Employed               0
LoanAmount                  0
Singleton                   0
Small_Family                0
Large_Family                0
```

```
Total_Income              0
Debt_Income_Ratio         0
Very_Short_Term           0
Short_Term                0
Long_Term                 0
Very_Long_Term            0
Credit_History_Bad        0
Credit_History_Good       0
Credit_History_Unknown    0
Property_Rural            0
Property_Semiurban        0
Property_Urban            0
dtype: int64
```

[61]:

```
combined.head()
```

[61]:

```
   Gender  Married  Education  Self_Employed  LoanAmount  Singleton  \
0       1        0          1              0       126.0          0
1       1        1          1              0       128.0          1
2       1        1          1              1        66.0          0
3       1        1          0              0       120.0          0
4       1        0          1              0       141.0          0

   Small_Family  Large_Family  Total_Income  Debt_Income_Ratio  \
0             0             0        5849.0          46.420635
1             0             0        6091.0          47.585938
2             0             0        3000.0          45.454545
3             0             0        4941.0          41.175000
4             0             0        6000.0          42.553191

   Very_Short_Term  Short_Term  Long_Term  Very_Long_Term  Credit_History_Bad  \
0                0           0          0               1                   0
1                0           0          0               1                   0
2                0           0          0               1                   0
3                0           0          0               1                   0
4                0           0          0               1                   0

   Credit_History_Good  Credit_History_Unknown  Property_Rural  \
0                    1                       0               0
1                    1                       0               1
2                    1                       0               0
3                    1                       0               0
4                    1                       0               0

   Property_Semiurban  Property_Urban
0                   0               1
1                   0               0
```

```
2                  0               1
3                  0               1
4                  0               1
```

[62]:
```python
def feature_scaling(df):
    df -= df.min()
    df /= df.max()
    return df
```

[63]:
```python
combined['LoanAmount'] = feature_scaling(combined['LoanAmount'])
combined['Total_Income'] = feature_scaling(combined['Total_Income'])
combined['Debt_Income_Ratio'] = feature_scaling(combined['Debt_Income_Ratio'])
```

[64]:
```python
combined[60:70]
```

[64]:
```
    Gender  Married  Education  Self_Employed  LoanAmount  Singleton  \
60       1        1          1              0    0.160637          0
61       1        1          1              0    0.130246          0
62       1        1          0              1    0.225760          0
63       1        1          1              0    0.169320          1
64       0        0          1              0    0.154848          0
65       1        1          1              0    0.360347          0
66       1        0          0              0    0.169320          0
67       1        1          1              0    0.438495          1
68       1        1          0              1    0.167873          0
69       0        0          1              0    0.183792          0

    Small_Family  Large_Family  Total_Income  Debt_Income_Ratio  \
60             0             0      0.061012           0.082855
61             0             1      0.019948           0.040406
62             0             0      0.058021           0.052283
63             0             0      0.044031           0.057195
64             0             0      0.034239           0.050728
65             0             0      0.111604           0.058666
66             0             0      0.050429           0.065036
67             0             0      0.116996           0.047897
68             0             1      0.071118           0.091266
69             0             0      0.035923           0.042389

    Very_Short_Term  Short_Term  Long_Term  Very_Long_Term  \
60                0           0          0               1
61                0           0          0               1
62                0           0          1               0
63                0           0          0               1
64                0           0          0               1
65                0           0          0               1
66                0           0          1               0
```

```
67                   0             0            0                 1
68                   1             0            0                 0
69                   0             0            0                 1

    Credit_History_Bad  Credit_History_Good  Credit_History_Unknown  \
60                   0                    1                       0
61                   0                    1                       0
62                   1                    0                       0
63                   1                    0                       0
64                   1                    0                       0
65                   0                    1                       0
66                   1                    0                       0
67                   0                    1                       0
68                   0                    1                       0
69                   1                    0                       0

    Property_Rural  Property_Semiurban  Property_Urban
60               0                   0               1
61               0                   0               1
62               1                   0               0
63               1                   0               0
64               0                   1               0
65               0                   1               0
66               0                   0               1
67               0                   0               1
68               0                   0               1
69               0                   1               0
```

2 Model Building

2.1 1. Random Forest

[65]:
```python
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.feature_selection import SelectFromModel
```

[66]:
```python
def compute_score(clf, X, y, scoring='accuracy'):
    xval = cross_val_score(clf, X, y, cv = 5, scoring=scoring)
    return np.mean(xval)
```

[67]:
```python
def recover_train_test_target():
    global combined, data_train
    targets = data_train['Loan_Status'].map({'Y':1,'N':0})
    train = combined.head(614)
    test = combined.iloc[614:]
    return train, test, targets
```

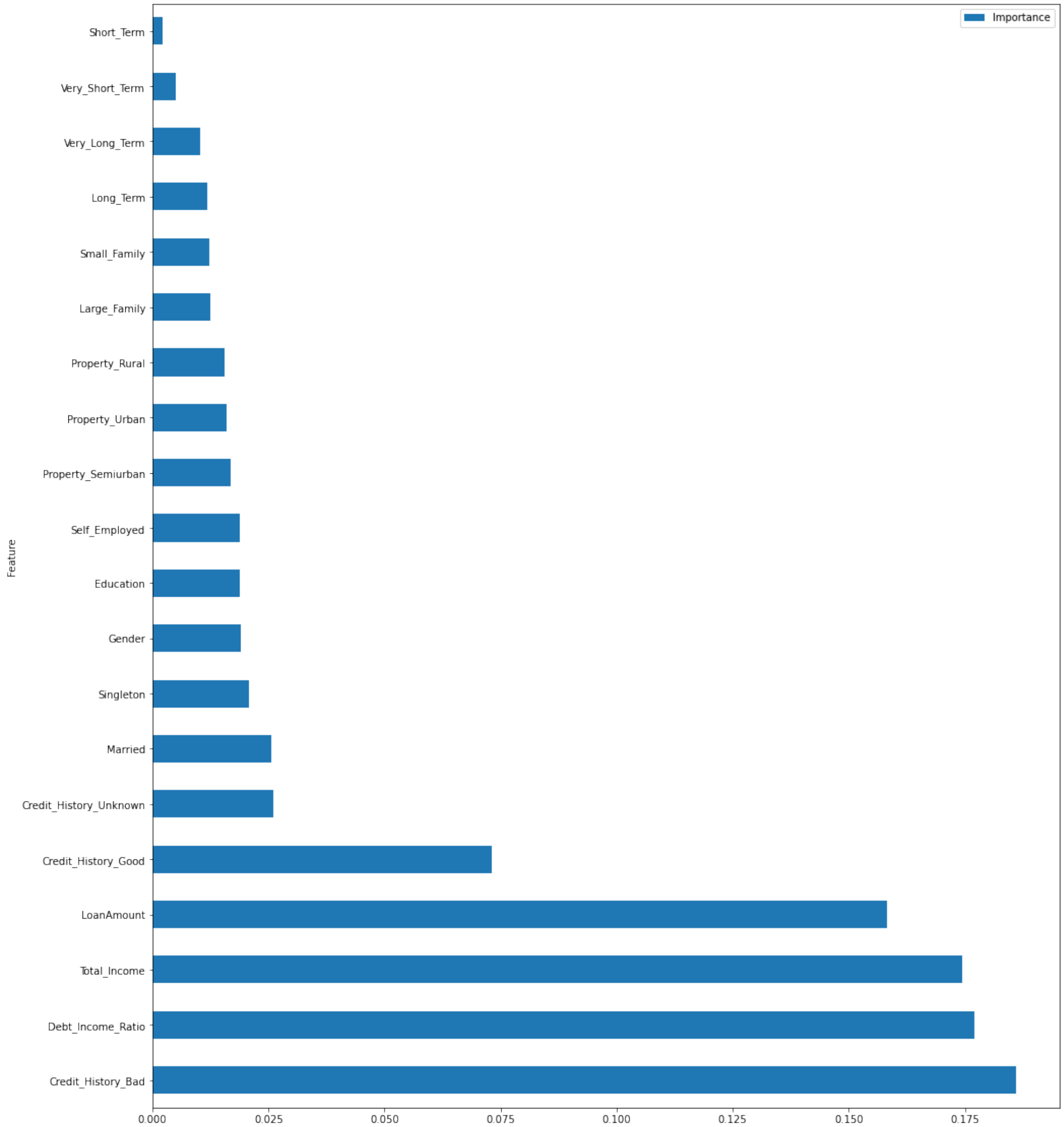
[68]:
```
train, test, targets = recover_train_test_target()
```

[69]:
```
clf = RandomForestClassifier(n_estimators=50, max_features='sqrt')
clf = clf.fit(train, targets)
```

[70]:
```
features = pd.DataFrame()
features['Feature'] = train.columns
features['Importance'] = clf.feature_importances_
features.sort_values(by=['Importance'], ascending=False, inplace=True)
features.set_index('Feature', inplace=True)
```

[71]:
```
features.plot(kind='barh', figsize=(16, 20))
```

[71]:
```
<AxesSubplot:ylabel='Feature'>
```



[72]:
```
model = SelectFromModel(clf, prefit=True)
train_reduced = model.transform(train)
train_reduced.shape
```

```
C:\Users\Dileep\anaconda3\lib\site-packages\sklearn\base.py:438: UserWarning: X
has feature names, but SelectFromModel was fitted without feature names
  warnings.warn(
```

[72]: (614, 5)

[73]:
```
test_reduced = model.transform(test)
test_reduced.shape
```

```
C:\Users\Dileep\anaconda3\lib\site-packages\sklearn\base.py:438: UserWarning: X
has feature names, but SelectFromModel was fitted without feature names
  warnings.warn(
```

[73]:
```
(367, 5)
```

[74]:
```
parameters = {'bootstrap': False,
              'min_samples_leaf': 3,
              'n_estimators':100,
              'min_samples_split': 10,
              'max_features': 'auto',
              'max_depth': 5}

model = RandomForestClassifier(**parameters)
model.fit(train, targets)
```

[74]:
```
RandomForestClassifier(bootstrap=False, max_depth=5, min_samples_leaf=3,
                       min_samples_split=10)
```

[75]:
```
print("Accuracy:{:.2f}".format(compute_score(model, train, targets,␣
 ↪scoring='accuracy')))
```

```
Accuracy:0.80
```

2.2 2. Gradient Boosting

[76]:
```
from sklearn import ensemble
gb = ensemble.GradientBoostingClassifier(n_estimators = 100, max_depth = 5,␣
 ↪min_samples_split = 2,
          learning_rate = 0.1)
```

[77]:
```
gb.fit(train, targets)
```

[77]:
```
GradientBoostingClassifier(max_depth=5)
```

[78]:
```
print("Accuracy:",clf.score(train, targets))
```

```
Accuracy: 0.998371335504886
```

2.3 3. Logistic Regression

[79]:
```
from sklearn.linear_model import LogisticRegression
```

[80]:
```
logreg = LogisticRegression()
logreg.fit(train, targets)
```

[80]:
```
LogisticRegression()
```

[81]:
```
print("Accuracy:",logreg.score(train, targets))
```

```
Accuracy: 0.8094462540716613
```

2.4 Submission

[82]:
```
output = gb.predict(test).astype(int)
df_output = pd.DataFrame()
aux = pd.read_csv('test.csv')
df_output['Loan_ID'] = aux['Loan_ID']
df_output['Loan_Status'] = np.vectorize(lambda s: 'Y' if s==1 else 'N')(output)
df_output[['Loan_ID','Loan_Status']].to_csv('submission.csv',index=False)
```

[83]:
```
df_output.head()
```

[83]:
```
    Loan_ID Loan_Status
0  LP001015           Y
1  LP001022           Y
2  LP001031           Y
3  LP001035           Y
4  LP001051           Y
```

[84]:
```
df_output['Loan_Status'].value_counts()
```

[84]:
```
Y    270
N     97
Name: Loan_Status, dtype: int64
```

[85]:
```
df_output.shape
```

[85]:
```
(367, 2)
```

2.5 Findings from the data

1. Applicants who are male and married tends to have more applicant income whereas applicant who are female and married have least applicant income
2. Applicants who are male and are graduated have more applicant income over the applicants who have not graduated.
3. Again the applicants who are married and graduated have the more applicant income.
4. Applicants who are not self employed have more applicant income than the applicants who are self employed.

5. Applicants who have more dependents have least applicant income whereas applicants which have no dependents have maximum applicant income.
6. Applicants who have property in urban and have credit history have maximum applicant income
7. Applicants who are graduate and have credit history have more applicant income.
8. Loan Amount is linearly dependent on Applicant income
9. From heatmaps, applicant income and loan amount are highly positively correlated.
10. Male applicants are more than female applicants.
11. No of applicants who are married are more than no of applicants who are not married.
12. Applicants with no dependents are maximum.
13. Applicants with graduation are more than applicants whith no graduation.
14. Property area is to be find more in semi urban areas and minimum in rural areas.