loan prediction model

November 15, 2024

[2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib_inline
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

[3]:
```
df=pd.read_csv('loan.csv')
df.head()
```

[3]:
```
    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
```

[4]:
```
df.shape
```

[4]:
```
(614, 13)
```

[5]:
```
df.describe()
```

```
[5]:        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
```

```
[6]: df['LoanAmount_log']=np.log(df['LoanAmount'])
```

```
[7]: df.isnull().sum()
```

```
[7]: 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
     LoanAmount_log       22
     dtype: int64
```

```
[8]: df['Gender'].fillna(df['Gender'].mode()[0], inplace=True)
     df['Married'].fillna(df['Married'].mode()[0], inplace=True)
     df['Dependents'].fillna(df['Dependents'].mode()[0],inplace=True)
     df['Self_Employed'].fillna(df['Self_Employed'].mode()[0],inplace=True)
     df['LoanAmount']=df['LoanAmount'].fillna(df['LoanAmount'].mean())
     df['LoanAmount_log']=df['LoanAmount_log'].fillna(df['LoanAmount_log'].mean())
     df['Loan_Amount_Term'].fillna(df['Loan_Amount_Term'].mode()[0], inplace=True)
```

```
df['Credit_History'].fillna(df['Credit_History'].mode()[0], inplace=True)
```

[9]:

```
df.isnull().sum()
```

[9]:

```
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
LoanAmount_log       0
dtype: int64
```

[10]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 14 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Loan_ID            614 non-null    object
 1   Gender             614 non-null    object
 2   Married            614 non-null    object
 3   Dependents         614 non-null    object
 4   Education          614 non-null    object
 5   Self_Employed      614 non-null    object
 6   ApplicantIncome    614 non-null    int64
 7   CoapplicantIncome  614 non-null    float64
 8   LoanAmount         614 non-null    float64
 9   Loan_Amount_Term   614 non-null    float64
 10  Credit_History     614 non-null    float64
 11  Property_Area      614 non-null    object
 12  Loan_Status        614 non-null    object
 13  LoanAmount_log     614 non-null    float64
dtypes: float64(5), int64(1), object(8)
memory usage: 67.3+ KB
```

[11]:

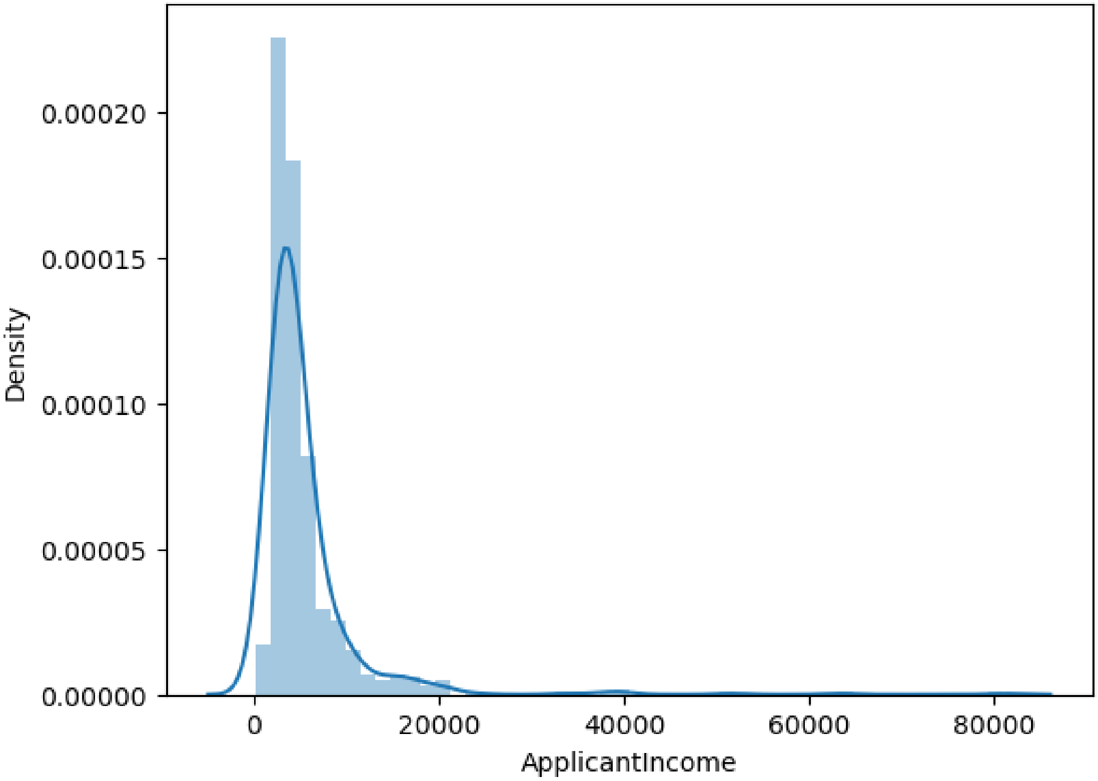
```
df['TotalIncome']=df['ApplicantIncome'] + df['CoapplicantIncome']
df['TotalIncome_log']=np.log(df['TotalIncome'])
```

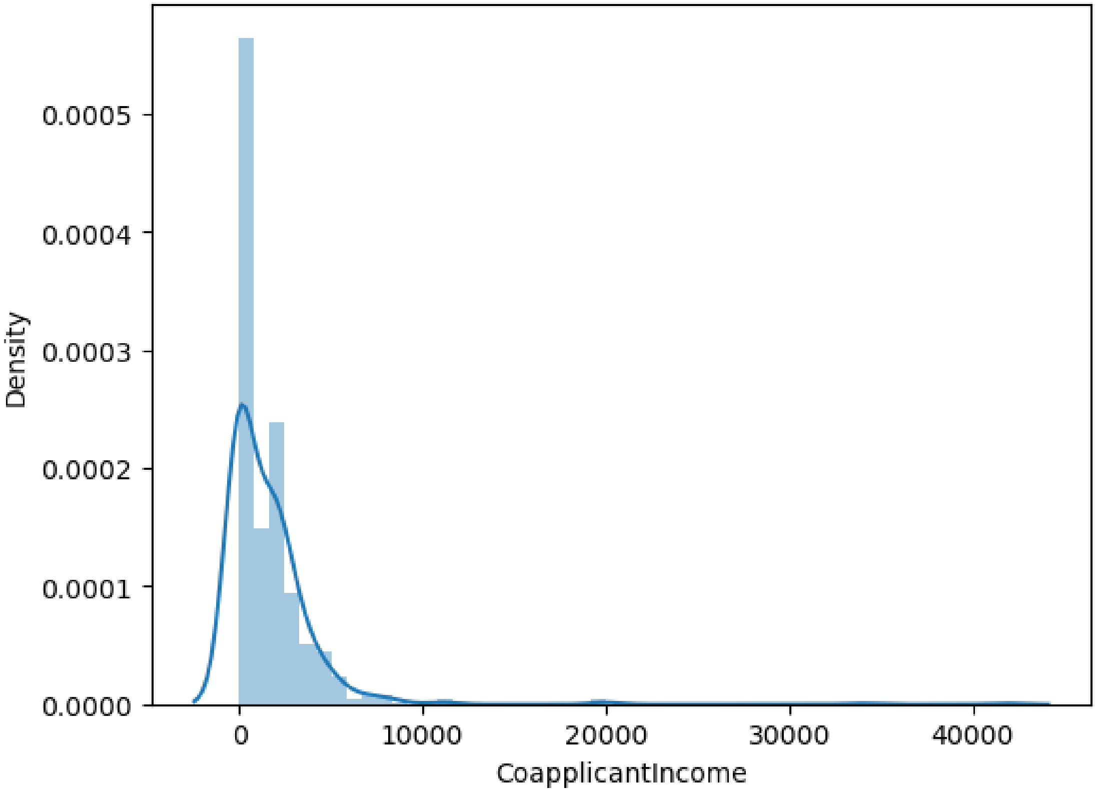
```
[12]: df.head()
```

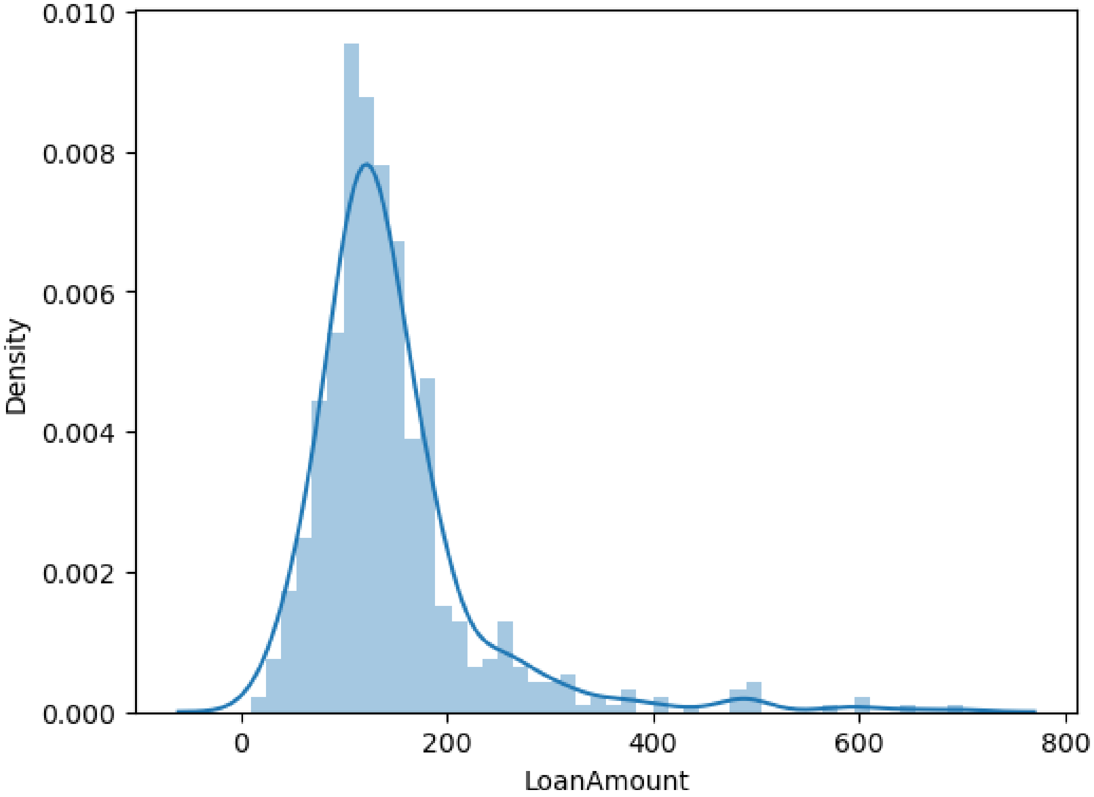
```
[12]:     Loan_ID Gender Married Dependents     Education Self_Employed  \
      0  LP001002   Male      No          0      Graduate            No
      1  LP001003   Male     Yes          1      Graduate            No
      2  LP001005   Male     Yes          0      Graduate           Yes
      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  146.412162             360.0
      1             4583             1508.0  128.000000             360.0
      2             3000                0.0   66.000000             360.0
      3             2583             2358.0  120.000000             360.0
      4             6000                0.0  141.000000             360.0

         Credit_History Property_Area Loan_Status  LoanAmount_log  TotalIncome  \
      0             1.0         Urban           Y        4.857444       5849.0
      1             1.0         Rural           N        4.852030       6091.0
      2             1.0         Urban           Y        4.189655       3000.0
      3             1.0         Urban           Y        4.787492       4941.0
      4             1.0         Urban           Y        4.948760       6000.0

         TotalIncome_log
      0         8.674026
      1         8.714568
      2         8.006368
      3         8.505323
      4         8.699515
```

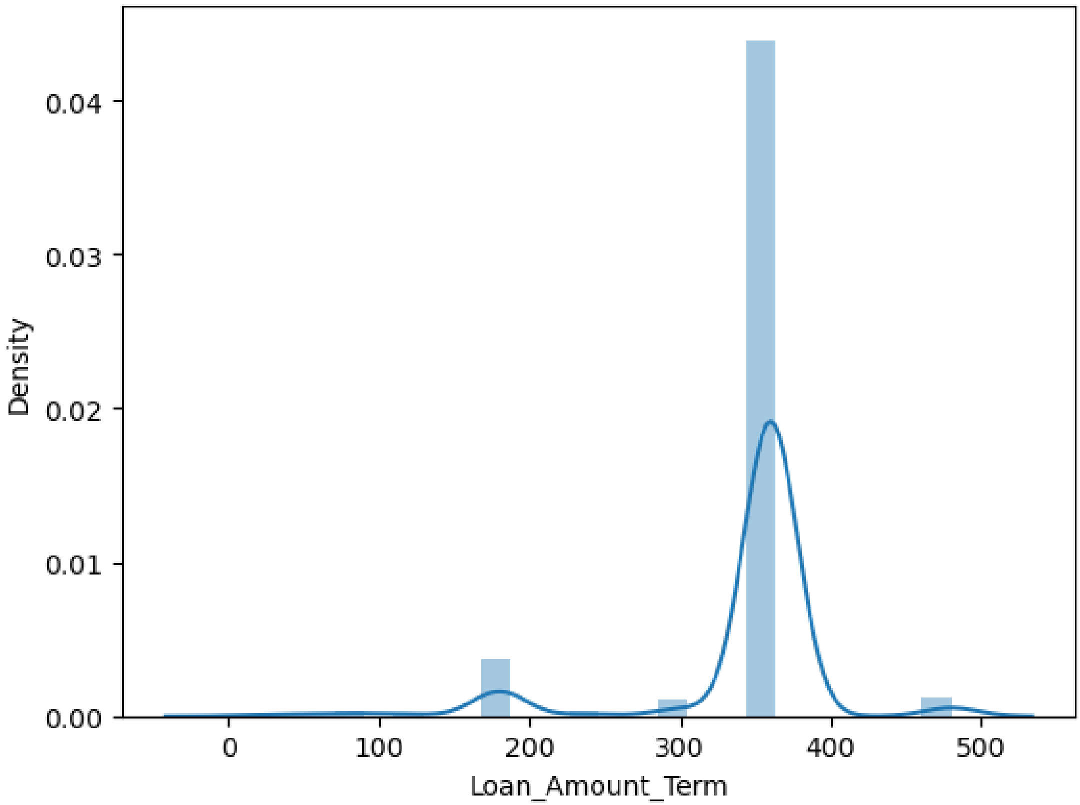
```
[13]: def distplots(col):
          sns.distplot(df[col])
          plt.show()
```

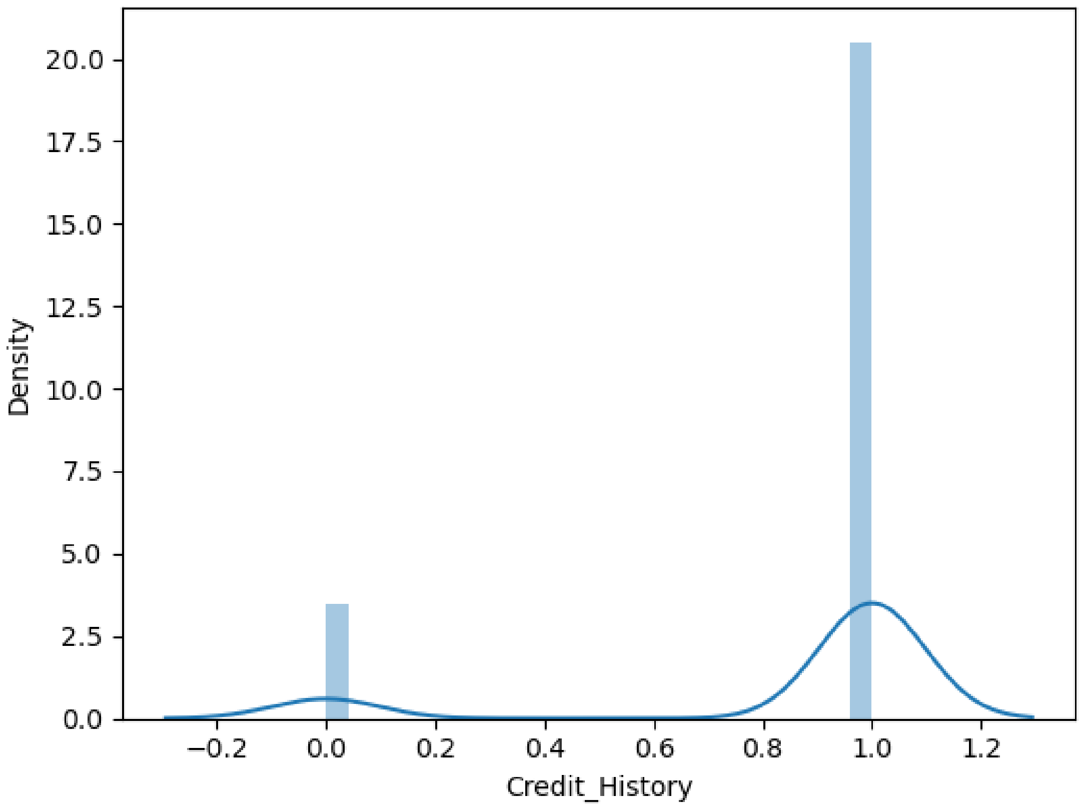
```
[14]: for i in list(df.select_dtypes(exclude=['object']).columns):
          distplots(i)
```

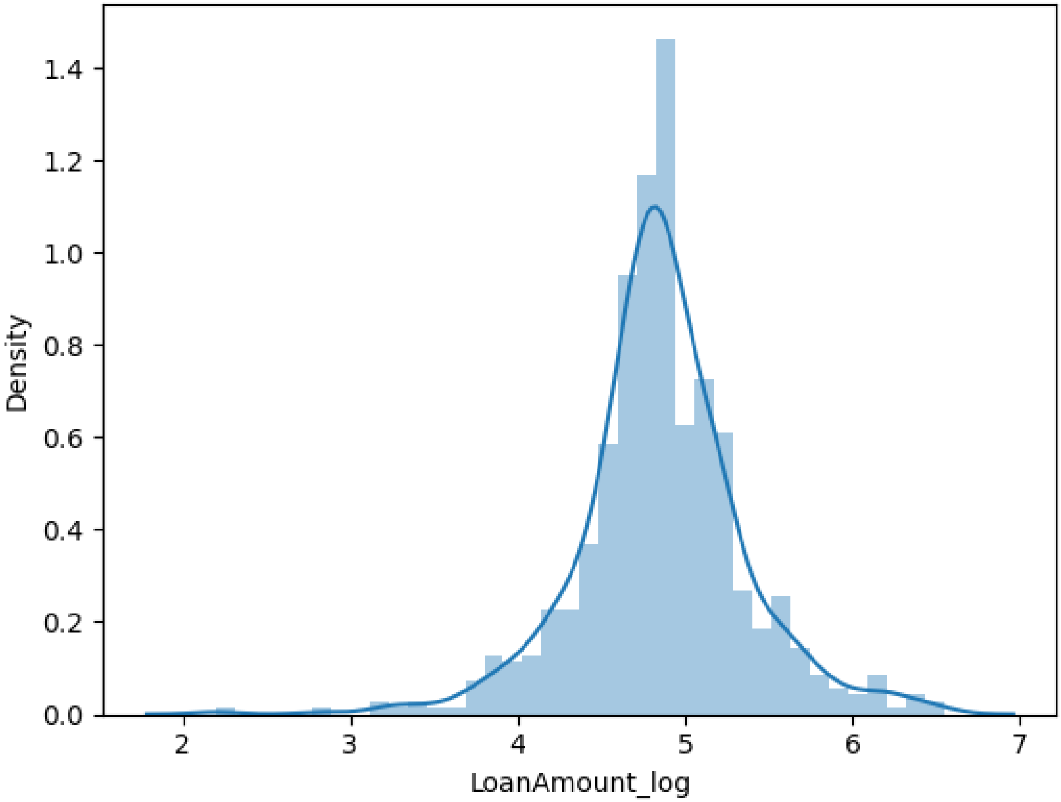

0.00020
0.00015
0.00010
0.00005
0.00000
Density
0
20000
40000
60000
80000
ApplicantIncome

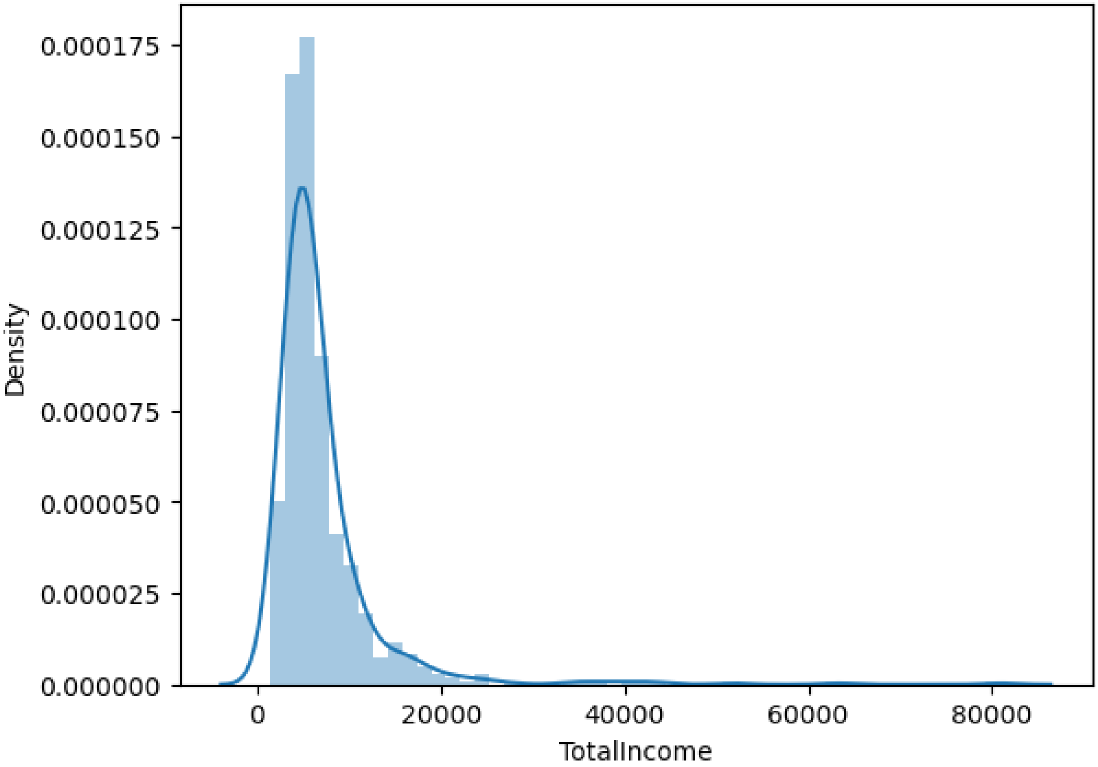

0.0005
0.0004
0.0003
0.0002
0.0001
0.0000
Density
0
10000
20000
30000
40000
CoapplicantIncome

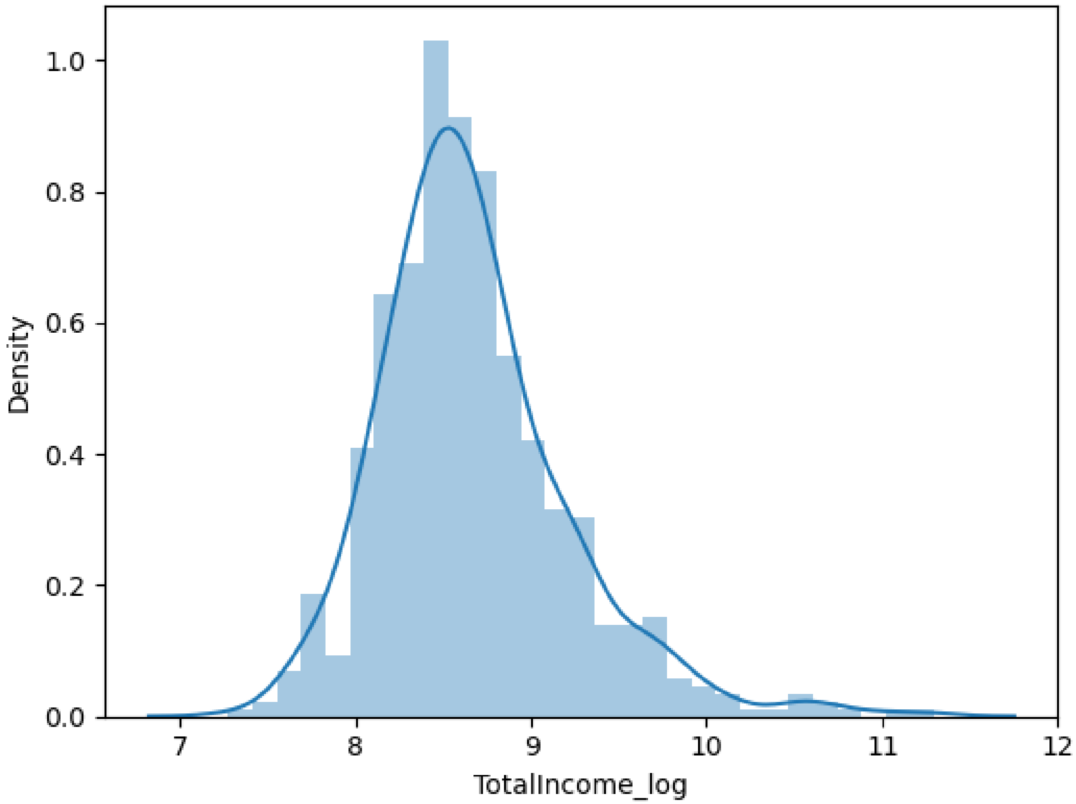

0.010
0.008
0.006
0.004
0.002
0.000
Density
0
200
400
600
800
LoanAmount


0.04
0.03
0.02
0.01
0.00
Density
0
100
200
300
400
500
Loan_Amount_Term

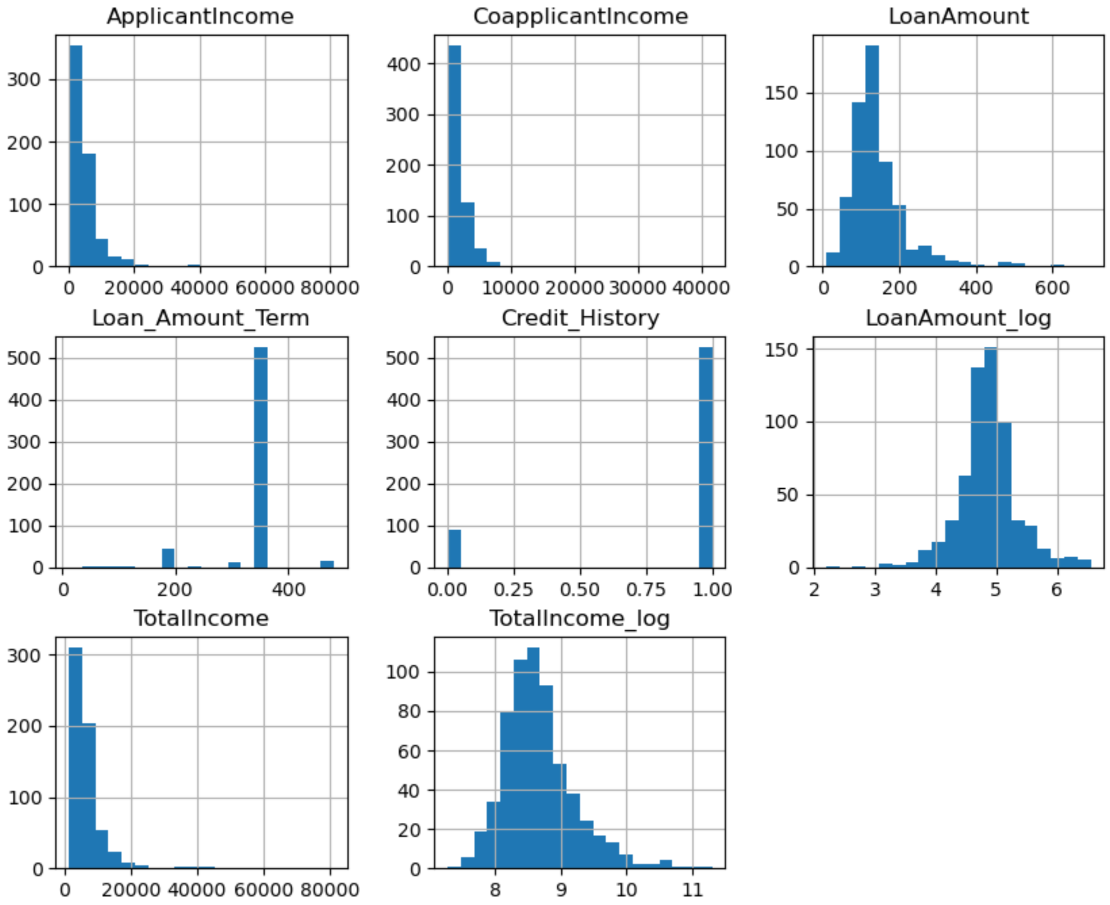

20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
Density
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2
Credit_History


1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Density
2
3
4
5
6
7
LoanAmount_log


0.000175
0.000150
0.000125
0.000100
0.000075
0.000050
0.000025
0.000000
Density
0
20000
40000
60000
80000
TotalIncome



[15]:
```
df.hist(bins=20, figsize=(10,8))
plt.show()
```



```
[16]: x=df.iloc[:,np.r_[1:5,9:11,13:15]].values
      y=df.iloc[:,12].values
```

```
[17]: 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=1)
```

```
[18]: from sklearn.preprocessing import LabelEncoder
      enco=LabelEncoder()
```

```
[19]: for i in range(0,5):
          x_train[:, i]=enco.fit_transform(x_train[:, i])
```

```
[20]: y_train=enco.fit_transform(y_train)
```

```
[21]: for i in range(0,5):
          x_test[:, i]=enco.fit_transform(x_test[:, i])
```

[22]:
```
x_test[:, 7]=enco.fit_transform(x_test[:, 7])
```

[23]:
```
y_test=enco.fit_transform(y_test)
```

[24]:
```
from sklearn.preprocessing import StandardScaler
sca=StandardScaler()
sca.fit_transform(x_train, x_test)
```

[24]:
```
array([[ 0.45429969,  0.70171306,  1.2229778 , …, -2.41235247,
        -0.03635731, -0.40050182],
       [ 0.45429969, -1.42508393, -0.75881131, …,  0.41453312,
        -0.58541747, -0.52100416],
       [-2.20119015,  0.70171306, -0.75881131, …,  0.41453312,
        -0.31831824, -0.04887686],
       …,
       [ 0.45429969, -1.42508393, -0.75881131, …,  0.41453312,
        -0.91876924, -0.53324614],
       [ 0.45429969,  0.70171306,  0.23208325, …,  0.41453312,
         0.53580108, -0.05241671],
       [-2.20119015,  0.70171306, -0.75881131, …,  0.41453312,
         0.21012861, -0.2934214 ]])
```

[25]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier,AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
```

[26]:
```
models={
    'Support Vendor Machine': SVC(),
    'Logistic Regression': LogisticRegression(),
    'Random Forest': RandomForestClassifier(),
    'Decision Tree': DecisionTreeClassifier()
}
```

[27]:
```
model_names = []
accuracies = []


for name, clf in models.items():
    clf.fit(x_train, y_train)
    score = clf.score(x_test, y_test)
    model_names.append(name)
    accuracies.append(score)
    print(f"{name} accuracy: {score:.2f}")


df_models = pd.DataFrame({'Model': model_names, 'Accuracy': accuracies})
```

```
Support Vendor Machine accuracy: 0.68
Logistic Regression accuracy: 0.80
Random Forest accuracy: 0.32
Decision Tree accuracy: 0.40
```

[ ]:

[ ]:

[ ]:

[ ]:

[ ]: