bank-personal-loan-modelling

February 3, 2024

1 Import required Libraries

[1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

[2]:

```
df = pd.read_csv('Bank_Personal_Loan_Modelling.csv')
df.head()
```

[2]:

```
   ID  Age  Experience  Income  ZIP Code  Family  CCAvg  Education  Mortgage  \
0   1   25           1      49     91107       4    1.6          1         0
1   2   45          19      34     90089       3    1.5          1         0
2   3   39          15      11     94720       1    1.0          1         0
3   4   35           9     100     94112       1    2.7          2         0
4   5   35           8      45     91330       4    1.0          2         0

   Personal Loan  Securities Account  CD Account  Online  CreditCard
0              0                   1           0       0           0
1              0                   1           0       0           0
2              0                   0           0       0           0
3              0                   0           0       0           0
4              0                   0           0       0           1
```

[3]:

```
df.shape
```

[3]:

```
(5000, 14)
```

[4]:

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

[4]:

```
ID                    0
Age                   0
Experience            0
Income                0
ZIP Code              0
Family                0
CCAvg                 0
```

```
Education             0
Mortgage              0
Personal Loan         0
Securities Account    0
CD Account            0
Online                0
CreditCard            0
dtype: int64
```

```
[5]: df.columns
```

```
[5]: Index(['ID', 'Age', 'Experience', 'Income', 'ZIP Code', 'Family', 'CCAvg',
            'Education', 'Mortgage', 'Personal Loan', 'Securities Account',
            'CD Account', 'Online', 'CreditCard'],
           dtype='object')
```

```
[6]: df.drop(columns=['ID','ZIP Code'], axis=1, inplace = True)
```

```
[7]: df.describe().T
```

```
[7]:                      count       mean         std   min   25%   50%    75%  \
     Age                 5000.0  45.338400   11.463166  23.0  35.0  45.0   55.0
     Experience          5000.0  20.104600   11.467954  -3.0  10.0  20.0   30.0
     Income              5000.0  73.774200   46.033729   8.0  39.0  64.0   98.0
     Family              5000.0   2.396400    1.147663   1.0   1.0   2.0    3.0
     CCAvg               5000.0   1.937938    1.747659   0.0   0.7   1.5    2.5
     Education           5000.0   1.881000    0.839869   1.0   1.0   2.0    3.0
     Mortgage            5000.0  56.498800  101.713802   0.0   0.0   0.0  101.0
     Personal Loan       5000.0   0.096000    0.294621   0.0   0.0   0.0    0.0
     Securities Account  5000.0   0.104400    0.305809   0.0   0.0   0.0    0.0
     CD Account          5000.0   0.060400    0.238250   0.0   0.0   0.0    0.0
     Online              5000.0   0.596800    0.490589   0.0   0.0   1.0    1.0
     CreditCard          5000.0   0.294000    0.455637   0.0   0.0   0.0    1.0

                           max
     Age                  67.0
     Experience           43.0
     Income              224.0
     Family                4.0
     CCAvg                10.0
     Education             3.0
     Mortgage            635.0
     Personal Loan         1.0
     Securities Account    1.0
     CD Account            1.0
     Online                1.0
     CreditCard            1.0
```

[8]:

```
import plotly.express as ps
```

[9]:

```
fig = ps.box(df, y = ['Age', 'Experience','Income','Family','Education'])
fig.show()
```

[10]:

```
df.dtypes
```

[10]:

```
Age                      int64
Experience               int64
Income                   int64
Family                   int64
CCAvg                  float64
Education                int64
Mortgage                 int64
Personal Loan            int64
Securities Account       int64
CD Account               int64
Online                   int64
CreditCard               int64
dtype: object
```

[11]:

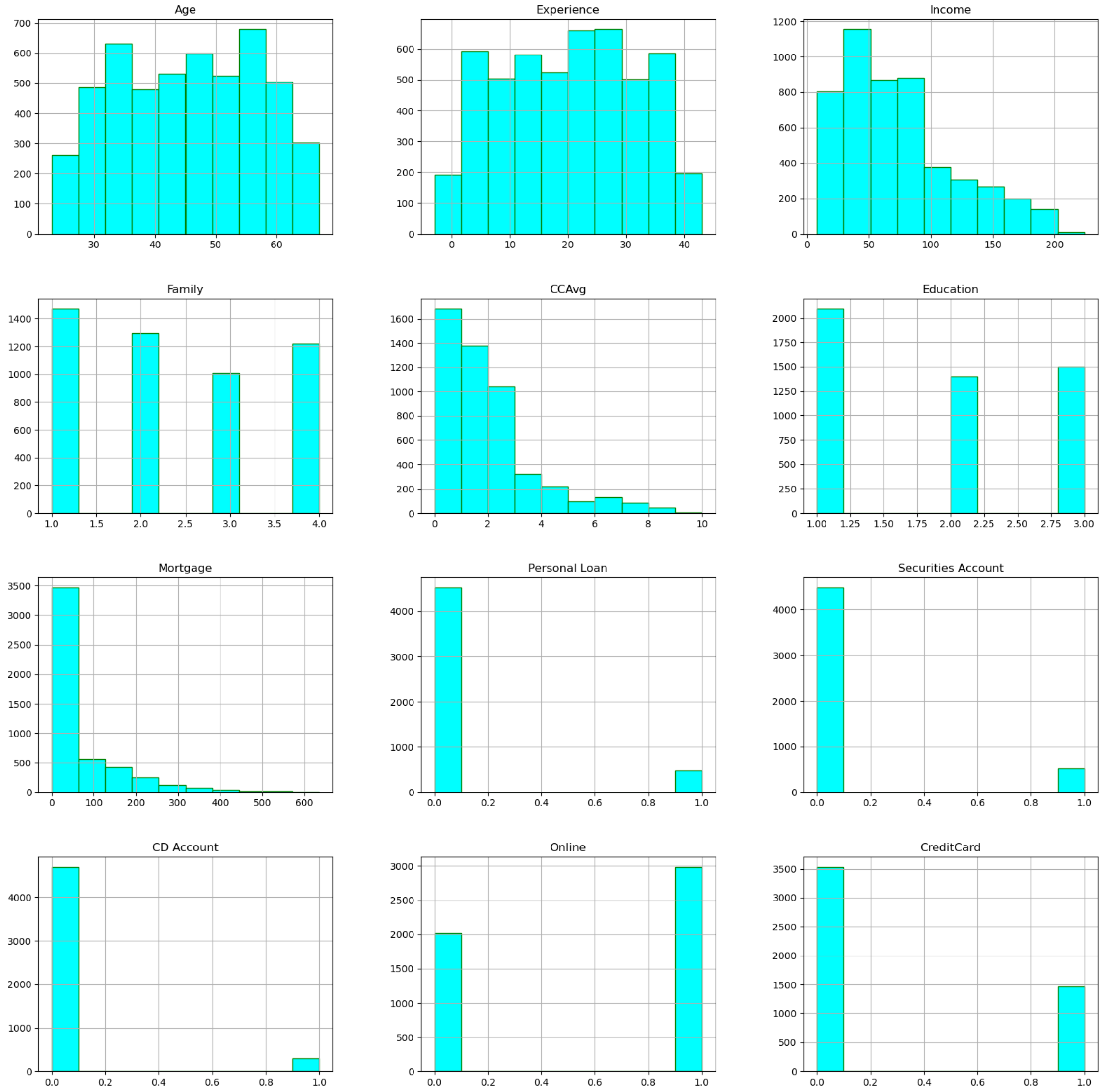
```
df.skew()
```

[11]:

```
Age                  -0.029341
Experience           -0.026325
Income                0.841339
Family                0.155221
CCAvg                 1.598443
Education             0.227093
Mortgage              2.104002
Personal Loan         2.743607
Securities Account    2.588268
CD Account            3.691714
Online               -0.394785
CreditCard            0.904589
dtype: float64
```

[12]:

```
df.hist(figsize=(20,20), color='cyan', edgecolor='green')
plt.show()
```

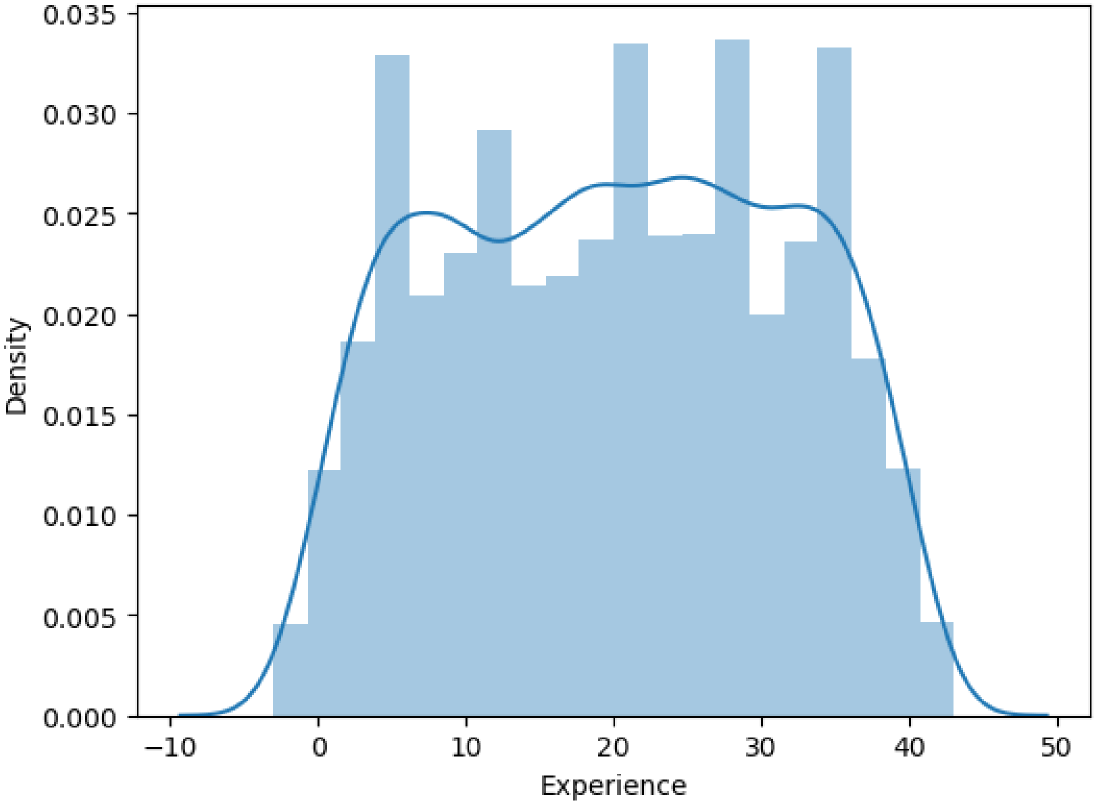


[13]:

```
sns.distplot(df['Experience'])
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_12060\4088753809.py:1: UserWarning:



`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751
```

```
[13]: <Axes: xlabel='Experience', ylabel='Density'>
```



```
[14]: df['Experience'].mean()
```

```
[14]: 20.1046
```

```
[15]: negative_exp = df[df['Experience']<0]
      negative_exp.head()
```

```
[15]:      Age  Experience  Income  Family  CCAvg  Education  Mortgage  \
      89    25          -1     113       4   2.30          3         0
      226   24          -1      39       2   1.70          2         0
      315   24          -2      51       3   0.30          3         0
      451   28          -2      48       2   1.75          3        89
      524   24          -1      75       4   0.20          1         0

           Personal Loan  Securities Account  CD Account  Online  CreditCard
      89               0                   0           0       0           1
```

```
226                0                   0             0         0             0
315                0                   0             0         1             0
451                0                   0             0         1             0
524                0                   0             0         1             0
```

[16]:

```
negative_exp.shape
```

[16]:

```
(52, 12)
```

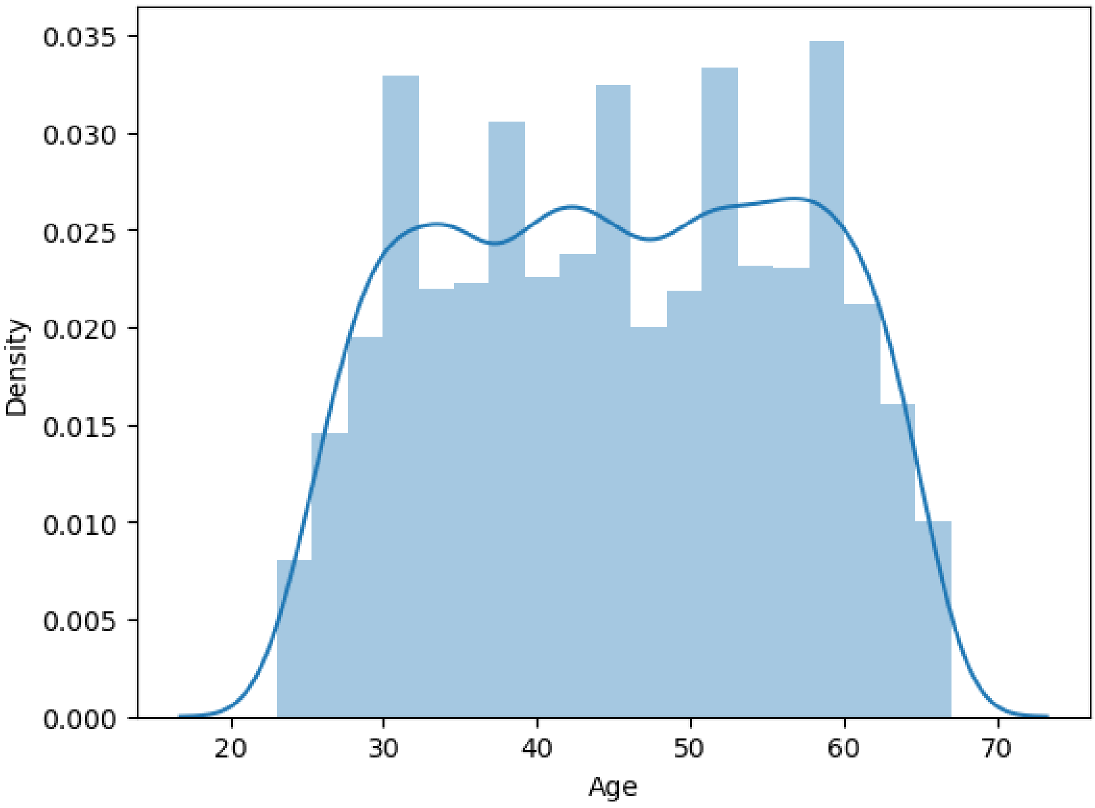
[17]:

```
sns.distplot(df['Age'])
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_12060\3255828239.py:1: UserWarning:



`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751
```

[17]:

```
<Axes: xlabel='Age', ylabel='Density'>
```



```
[18]: negative_exp['Experience'].mean()
```

```
[18]: -1.4423076923076923
```

```
[19]: negative_exp.size
```

```
[19]: 624
```

print('There are {} records which has negative values for experience, approx {} %'. format(negative_exp.((negative_exp.size/df.size) * 100)))

```
[20]: percentage = (negative_exp.size/df.size) * 100
      percentage
```

```
[20]: 1.04
```

```
[21]: data = df.copy()
```

```
[22]: data.head()
```

[22]:
```
   Age  Experience  Income  Family  CCAvg  Education  Mortgage  Personal Loan  \
0   25           1      49       4    1.6          1         0              0
1   45          19      34       3    1.5          1         0              0
2   39          15      11       1    1.0          1         0              0
3   35           9     100       1    2.7          2         0              0
4   35           8      45       4    1.0          2         0              0

   Securities Account  CD Account  Online  CreditCard
0                   1           0       0           0
1                   1           0       0           0
2                   0           0       0           0
3                   0           0       0           0
4                   0           0       0           1
```

[23]:
```
data.shape
```

[23]:
```
(5000, 12)
```

[24]:
```
data['Experience'] = np.where(data['Experience']<0,#check negative data
                             data['Experience'].mean(),#calculate mean
                             data['Experience'])#fill mean in negative data
```

[25]:
```
data[data['Experience']<0]
```

[25]:
```
Empty DataFrame
Columns: [Age, Experience, Income, Family, CCAvg, Education, Mortgage, Personal
Loan, Securities Account, CD Account, Online, CreditCard]
Index: []
```

[26]:
```
data.corr()
```

[26]:
```
                         Age  Experience    Income    Family     CCAvg  \
Age                 1.000000    0.977008 -0.055269 -0.046418 -0.052012
Experience          0.977008    1.000000 -0.049054 -0.045488 -0.048708
Income             -0.055269   -0.049054  1.000000 -0.157501  0.645984
Family             -0.046418   -0.045488 -0.157501  1.000000 -0.109275
CCAvg              -0.052012   -0.048708  0.645984 -0.109275  1.000000
Education           0.041334    0.018097 -0.187524  0.064929 -0.136124
Mortgage           -0.012539   -0.013378  0.206806 -0.020445  0.109905
Personal Loan      -0.007726   -0.014045  0.502462  0.061367  0.366889
Securities Account -0.000436   -0.000462 -0.002616  0.019994  0.015086
CD Account          0.008043    0.005502  0.169738  0.014110  0.136534
Online              0.013702    0.013455  0.014206  0.010354 -0.003611
CreditCard          0.007681    0.008833 -0.002385  0.011588 -0.006689

                    Education  Mortgage  Personal Loan  Securities Account  \
Age                  0.041334 -0.012539      -0.007726           -0.000436
```

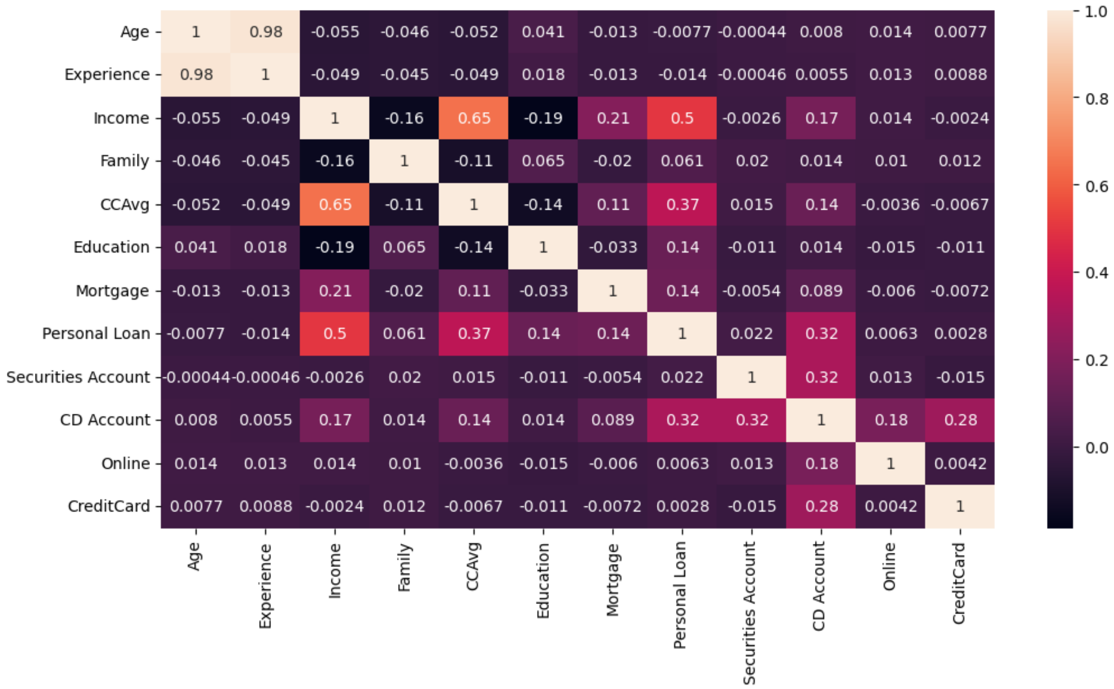
```
Experience           0.018097 -0.013378      -0.014045           -0.000462
Income              -0.187524  0.206806       0.502462           -0.002616
Family               0.064929 -0.020445       0.061367            0.019994
CCAvg               -0.136124  0.109905       0.366889            0.015086
Education            1.000000 -0.033327       0.136722           -0.010812
Mortgage            -0.033327  1.000000       0.142095           -0.005411
Personal Loan        0.136722  0.142095       1.000000            0.021954
Securities Account  -0.010812 -0.005411       0.021954            1.000000
CD Account           0.013934  0.089311       0.316355            0.317034
Online              -0.015004 -0.005995       0.006278            0.012627
CreditCard          -0.011014 -0.007231       0.002802           -0.015028

                    CD Account    Online  CreditCard
Age                   0.008043  0.013702    0.007681
Experience            0.005502  0.013455    0.008833
Income                0.169738  0.014206   -0.002385
Family                0.014110  0.010354    0.011588
CCAvg                 0.136534 -0.003611   -0.006689
Education             0.013934 -0.015004   -0.011014
Mortgage              0.089311 -0.005995   -0.007231
Personal Loan         0.316355  0.006278    0.002802
Securities Account    0.317034  0.012627   -0.015028
CD Account            1.000000  0.175880    0.278644
Online                0.175880  1.000000    0.004210
CreditCard            0.278644  0.004210    1.000000
```

[27]:

```
plt.figure(figsize=(12,6))
sns.heatmap(data.corr(), annot=True)
```

[27]:

```
<Axes: >
```



[28]:
```
data = data.drop(['Experience'], axis=1)
```

[29]:
```
data.head()
```

[29]:
```
   Age  Income  Family  CCAvg  Education  Mortgage  Personal Loan  \
0   25      49       4    1.6          1         0              0
1   45      34       3    1.5          1         0              0
2   39      11       1    1.0          1         0              0
3   35     100       1    2.7          2         0              0
4   35      45       4    1.0          2         0              0

   Securities Account  CD Account  Online  CreditCard
0                   1           0       0           0
1                   1           0       0           0
2                   0           0       0           0
3                   0           0       0           0
4                   0           0       0           1
```

[30]:
```
#Education
data['Education'].unique()
```

[30]:
```
array([1, 2, 3], dtype=int64)
```

[31]:
```
def experience(x):
    if x == 1:
        return "UnderGrade"
    if x == 2:
        return 'Graduate'
    if x == 3:
        return 'Professional Person'
```

[32]:
```
data['EDU'] = data['Education'].apply(experience)
```

[33]:
```
data.head()
```

[33]:
```
   Age  Income  Family  CCAvg  Education  Mortgage  Personal Loan  \
0   25      49       4    1.6          1         0              0
1   45      34       3    1.5          1         0              0
2   39      11       1    1.0          1         0              0
3   35     100       1    2.7          2         0              0
4   35      45       4    1.0          2         0              0

   Securities Account  CD Account  Online  CreditCard         EDU
0                   1           0       0           0  UnderGrade
1                   1           0       0           0  UnderGrade
2                   0           0       0           0  UnderGrade
3                   0           0       0           0    Graduate
4                   0           0       0           1    Graduate
```

[34]:
```
data['EDU'].unique()
```

[34]:
```
array(['UnderGrade', 'Graduate', 'Professional Person'], dtype=object)
```

[35]:
```
education_did = data.groupby('EDU')['Age'].count()
```

[36]:
```
education_did
```

[36]:
```
EDU
Graduate               1403
Professional Person    1501
UnderGrade             2096
Name: Age, dtype: int64
```

[37]:
```
fig = ps.pie(data, values=education_did, names = education_did.index,␣
 ↪title='Pie Chart')
fig.show()
```

[38]:
```
data.columns
```

```
[38]: Index(['Age', 'Income', 'Family', 'CCAvg', 'Education', 'Mortgage',
       'Personal Loan', 'Securities Account', 'CD Account', 'Online',
       'CreditCard', 'EDU'],
      dtype='object')
```

```
[39]: data['Income'].unique()
```

```
[39]: array([ 49,  34,  11, 100,  45,  29,  72,  22,  81, 180, 105, 114,  40,
       112, 130, 193,  21,  25,  63,  62,  43, 152,  83, 158,  48, 119,
        35,  41,  18,  50, 121,  71, 141,  80,  84,  60, 132, 104,  52,
       194,   8, 131, 190,  44, 139,  93, 188,  39, 125,  32,  20, 115,
        69,  85, 135,  12, 133,  19,  82, 109,  42,  78,  51, 113, 118,
        64, 161,  94,  15,  74,  30,  38,   9,  92,  61,  73,  70, 149,
        98, 128,  31,  58,  54, 124, 163,  24,  79, 134,  23,  13, 138,
       171, 168,  65,  10, 148, 159, 169, 144, 165,  59,  68,  91, 172,
        55, 155,  53,  89,  28,  75, 170, 120,  99, 111,  33, 129, 122,
       150, 195, 110, 101, 191, 140, 153, 173, 174,  90, 179, 145, 200,
       183, 182,  88, 160, 205, 164,  14, 175, 103, 108, 185, 204, 154,
       102, 192, 202, 162, 142,  95, 184, 181, 143, 123, 178, 198, 201,
       203, 189, 151, 199, 224, 218], dtype=int64)
```

```
[40]: data['Securities Account'].value_counts()
```

```
[40]: 0    4478
      1     522
      Name: Securities Account, dtype: int64
```

```
[41]: data['CD Account'].value_counts()
```

```
[41]: 0    4698
      1     302
      Name: CD Account, dtype: int64
```

```
[42]: def security(y):
          if (y['Securities Account'] == 1) & (y['CD Account'] ==1):
              return 'Holds Securities and Deposirt'
          if (y['Securities Account'] == 0) & (y['CD Account'] ==0):
              return 'Does not hold Securities and Deposite account'
          if (y['Securities Account'] == 1) & (y['CD Account'] ==0):
              return 'Holds only Securities accounts'
          if (y['Securities Account'] == 0) & (y['CD Account'] ==1):
              return 'Holds only Deposite accounts '
```

```
[43]: data['Account_holder_category'] = data.apply(security, axis=1)
```

```
[44]: data.head()
```

```
[44]:    Age  Income  Family  CCAvg  Education  Mortgage  Personal Loan  \
      0   25      49       4    1.6          1         0              0
      1   45      34       3    1.5          1         0              0
      2   39      11       1    1.0          1         0              0
      3   35     100       1    2.7          2         0              0
      4   35      45       4    1.0          2         0              0

         Securities Account  CD Account  Online  CreditCard         EDU  \
      0                   1           0       0           0  UnderGrade
      1                   1           0       0           0  UnderGrade
      2                   0           0       0           0  UnderGrade
      3                   0           0       0           0    Graduate
      4                   0           0       0           1    Graduate

                            Account_holder_category
      0                  Holds only Securities accounts
      1                  Holds only Securities accounts
      2  Does not hold Securities and Deposite account
      3  Does not hold Securities and Deposite account
      4  Does not hold Securities and Deposite account
```

```
[45]: values = data['Account_holder_category'].value_counts()
      values.index
```

```
[45]: Index(['Does not hold Securities and Deposite account',
             'Holds only Securities accounts', 'Holds only Deposite accounts ',
             'Holds Securities and Deposirt'],
            dtype='object')
```

```
[46]: fig = ps.pie(data, values=values, names = values.index, title = 'Pie Chart')
      fig.show()
```

```
[47]: ps.box(data, x='Education', y='Income', facet_col='Personal Loan')
```

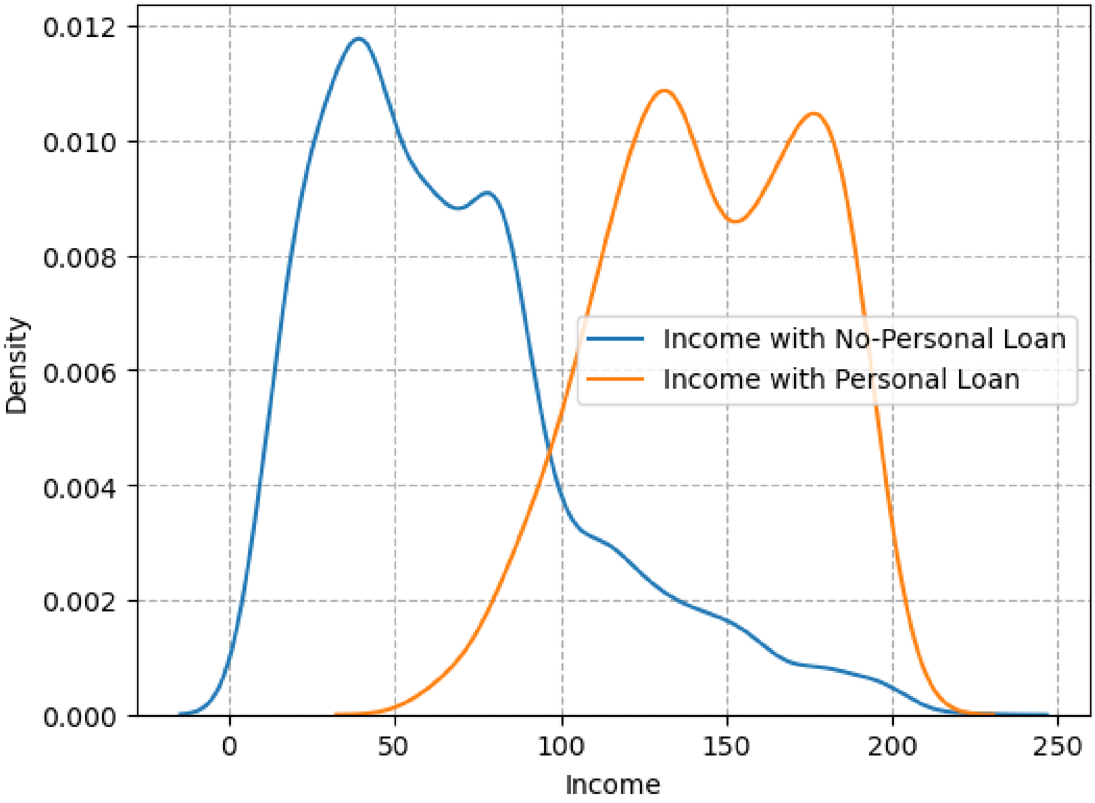
```
[48]: data.columns
```

```
[48]: Index(['Age', 'Income', 'Family', 'CCAvg', 'Education', 'Mortgage',
             'Personal Loan', 'Securities Account', 'CD Account', 'Online',
             'CreditCard', 'EDU', 'Account_holder_category'],
            dtype='object')
```

```
[49]: sns.distplot(data[data['Personal Loan'] == 0]['Income'], hist=False,
                   label= 'Income with No-Personal Loan')
      sns.distplot(data[data['Personal Loan'] == 1]['Income'], hist=False,
                   label= 'Income with Personal Loan')
      plt.grid(linestyle='--')
      plt.legend()
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_12060\1876980550.py:1: UserWarning:



`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `kdeplot` (an axes-level function for kernel density
plots).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751


C:\Users\Admin\AppData\Local\Temp\ipykernel_12060\1876980550.py:3: UserWarning:



`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `kdeplot` (an axes-level function for kernel density
plots).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751
```

```
[49]: <matplotlib.legend.Legend at 0x184a79f5610>
```

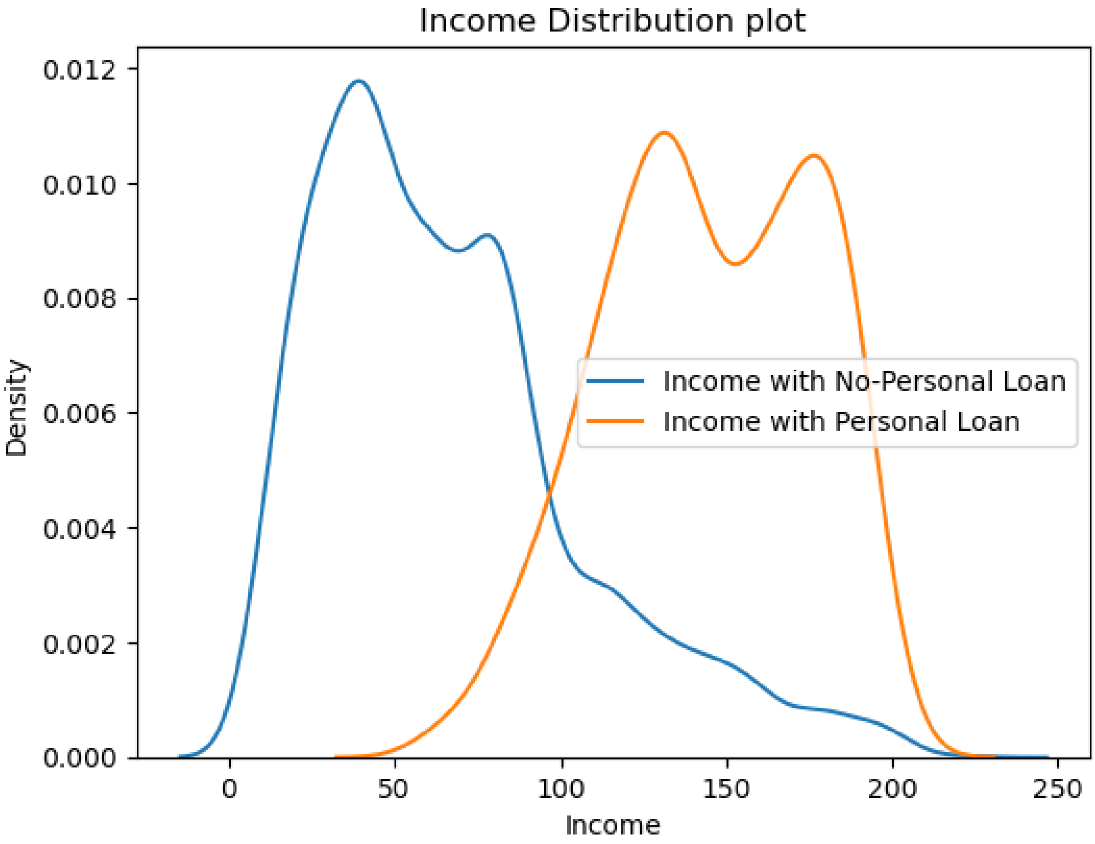


[50]:
```
def plot(col1,col2, label1, label2, title):
    sns.distplot(data[data[col2] == 0][col1], hist=False,
            label= label1)
    sns.distplot(data[data[col2] == 1][col1], hist=False,
            label= label2)
    plt.legend()
    plt.title(title)
```

[51]:
```
plot('Income','Personal Loan','Income with No-Personal Loan','Income with␣
 ↪Personal Loan','Income Distribution plot')
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_12060\1758143882.py:2: UserWarning:



`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `kdeplot` (an axes-level function for kernel density
plots).
```

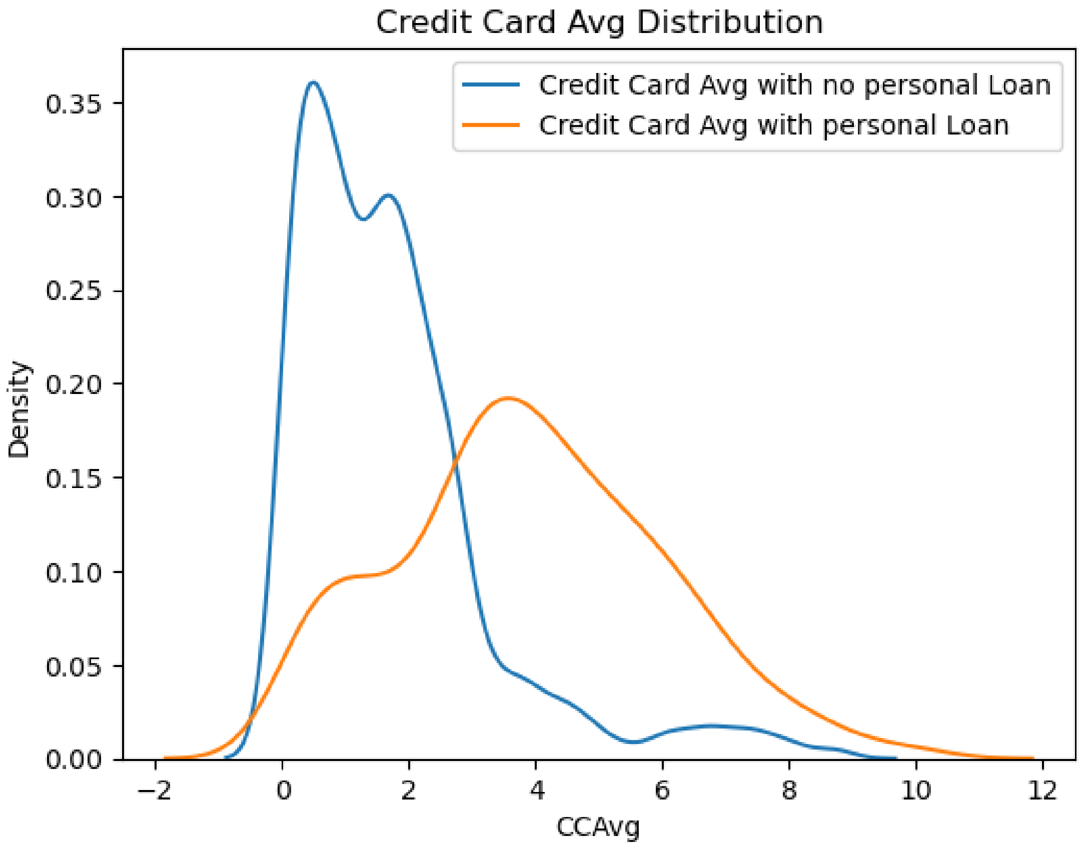
```
For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751


C:\Users\Admin\AppData\Local\Temp\ipykernel_12060\1758143882.py:4: UserWarning:



`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `kdeplot` (an axes-level function for kernel density
plots).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751
```



[52]:

```
plot('CCAvg','Personal Loan',
    'Credit Card Avg with no personal Loan',
    'Credit Card Avg with personal Loan',
     'Credit Card Avg Distribution'
    )
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_12060\1758143882.py:2: UserWarning:



`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `kdeplot` (an axes-level function for kernel density
plots).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751


C:\Users\Admin\AppData\Local\Temp\ipykernel_12060\1758143882.py:4: UserWarning:



`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `kdeplot` (an axes-level function for kernel density
plots).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751
```

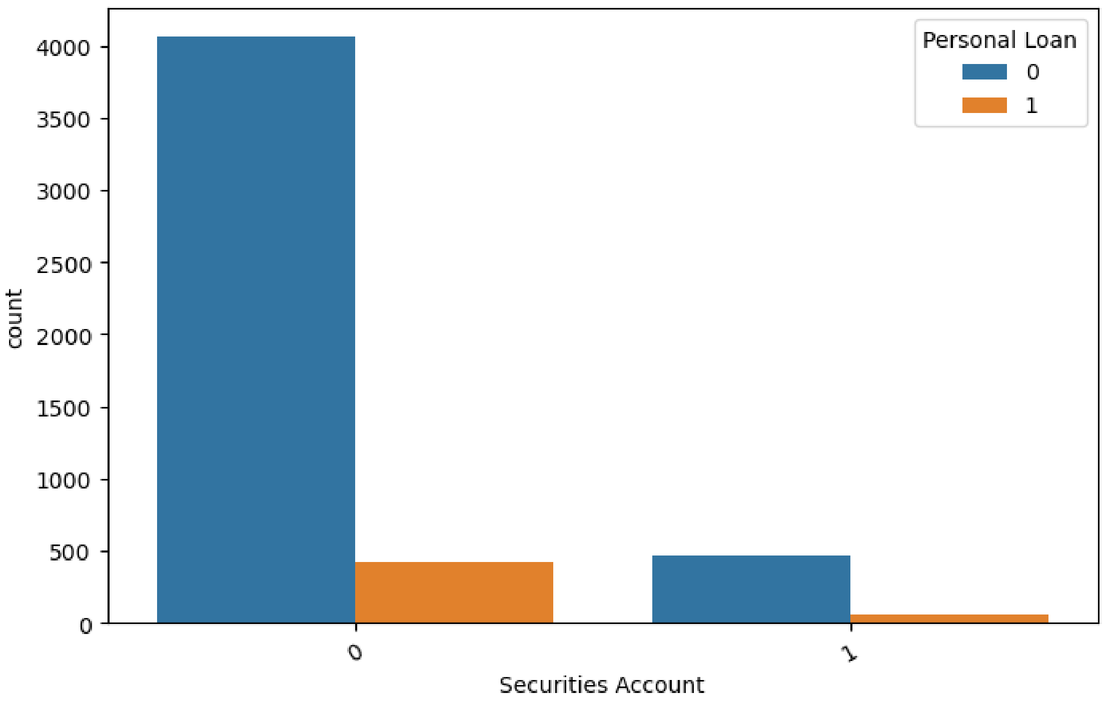


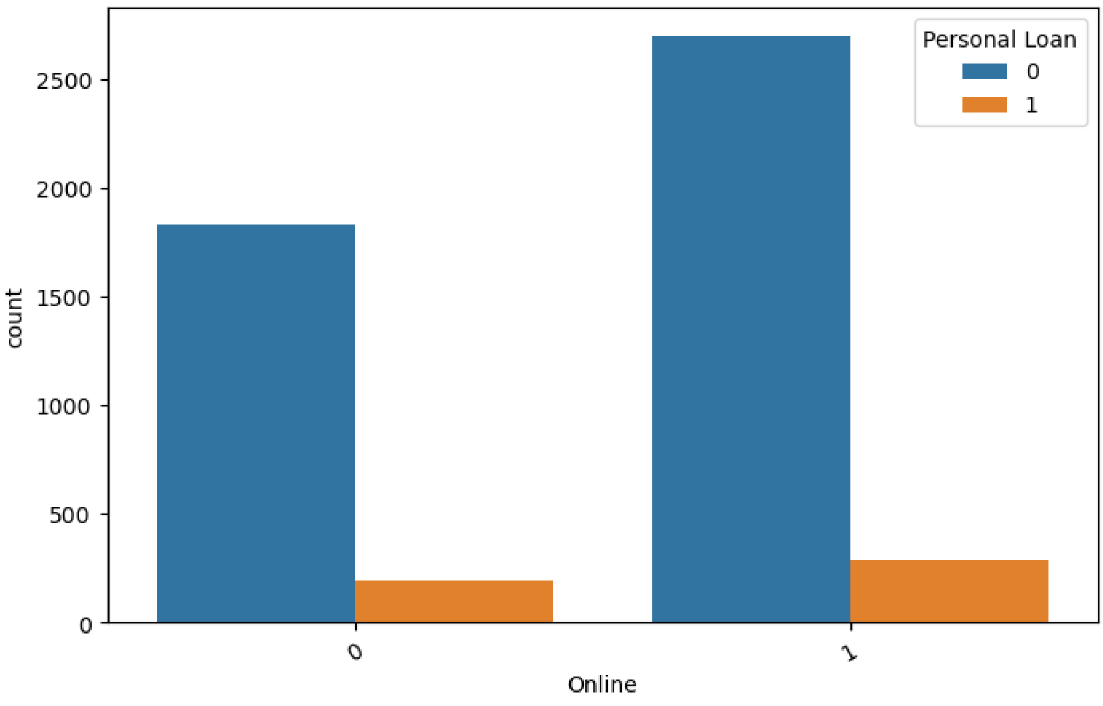
```
[53]: data.columns
```

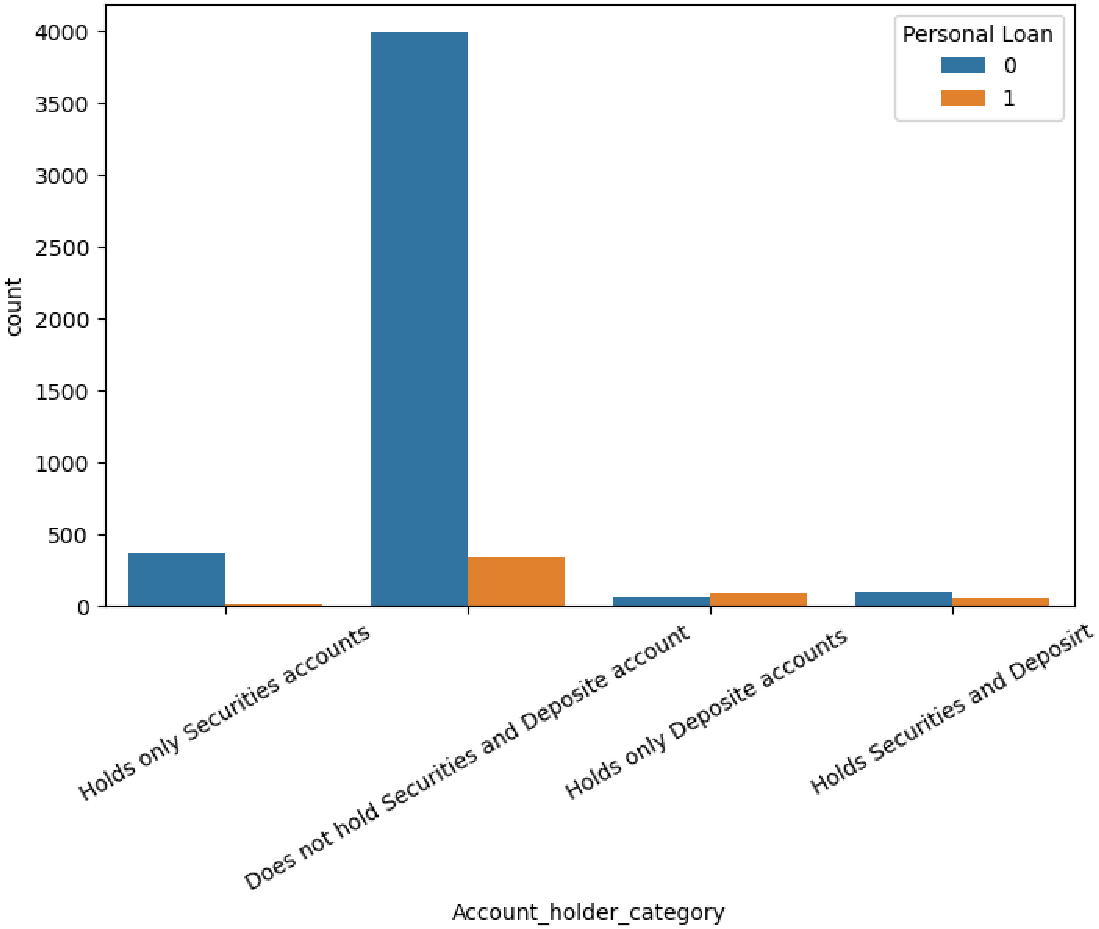
```
[53]: Index(['Age', 'Income', 'Family', 'CCAvg', 'Education', 'Mortgage',
              'Personal Loan', 'Securities Account', 'CD Account', 'Online',
              'CreditCard', 'EDU', 'Account_holder_category'],
             dtype='object')
```

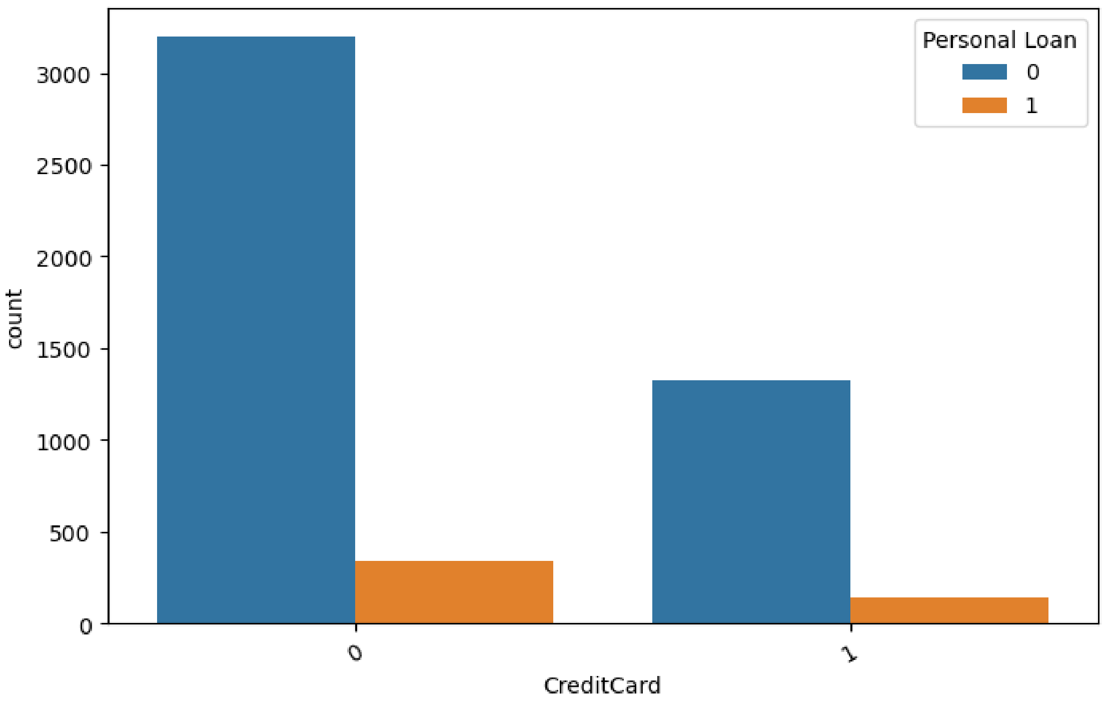
```
[54]: col =['Securities Account',
           'Online',
           'Account_holder_category',
           'CreditCard']
```

```
[55]: for I in col:
          plt.figure(figsize=(8,5))
          sns.countplot(x=I, data=data, hue='Personal Loan')
          plt.xticks(rotation=30)
```


Personal Loan
0
1
count
4000
3500
3000
2500
2000
1500
1000
500
0
0
1
Securities Account


Personal Loan
0
1
count
2500
2000
1500
1000
500
0
0
1
Online


4000
3500
3000
2500
2000
1500
1000
500
0
count
Personal Loan
0
1
Holds only Securities accounts
Does not hold Securities and Deposite account
Holds only Deposite accounts
Holds Securities and Deposirt
Account_holder_category



[56]:

```
from scipy.stats import zscore
```

[57]:

```
q1 = data.quantile(0.25)
q3 = data.quantile(0.75)

IQR = q3 - q1
print(IQR)
```

```
Age                    20.0
Income                 59.0
Family                  2.0
CCAvg                   1.8
Education               2.0
Mortgage              101.0
Personal Loan           0.0
Securities Account      0.0
CD Account              0.0
Online                  1.0
CreditCard              1.0
dtype: float64
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_12060\722697894.py:1: FutureWarning:

The default value of numeric_only in DataFrame.quantile is deprecated. In a
future version, it will default to False. Select only valid columns or specify
```

```
the value of numeric_only to silence this warning.

C:\Users\Admin\AppData\Local\Temp\ipykernel_12060\722697894.py:2: FutureWarning:

The default value of numeric_only in DataFrame.quantile is deprecated. In a
future version, it will default to False. Select only valid columns or specify
the value of numeric_only to silence this warning.
```

[58]:

```
# Log Normal Transformation

data_1 = data[['Income','CCAvg']]
data_1 = np.log(data_1 +1)
data_1
```

[58]:

```
        Income     CCAvg
0     3.912023  0.955511
1     3.555348  0.916291
2     2.484907  0.693147
3     4.615121  1.308333
4     3.828641  0.693147
…          …         …
4995  3.713572  1.064711
4996  2.772589  0.336472
4997  3.218876  0.262364
4998  3.912023  0.405465
4999  4.430817  0.587787

[5000 rows x 2 columns]
```

[59]:

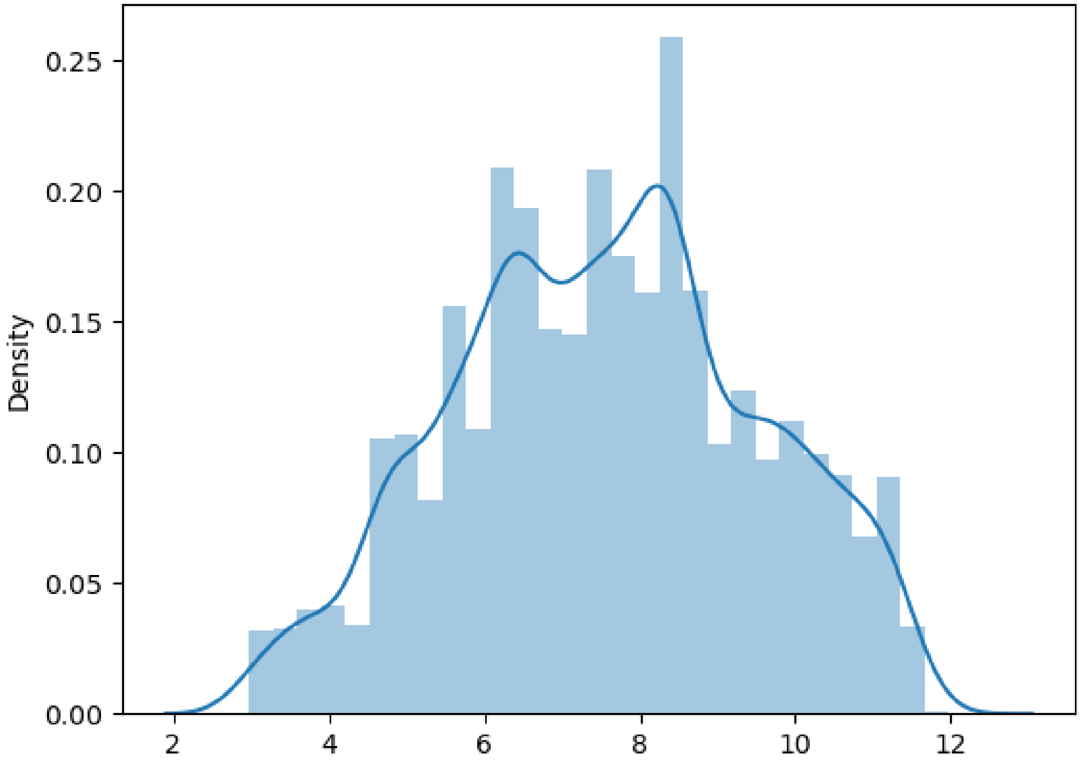
```
#Power Transforer
from sklearn.preprocessing import PowerTransformer
```

[60]:

```
pt = PowerTransformer(method='yeo-johnson', standardize=False)
pt.fit(data['Income'].values.reshape(-1,1))
Income = pt.transform(data['Income'].values.reshape(-1,1))
sns.distplot(Income)
plt.show()
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_12060\4180918905.py:4: UserWarning:



`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).
```

```
For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751
```



[61]:
```
#handle outliers
#convert categorical data into numerical
#model building
#Logistic, svm
```

[62]:
```
df.head()
```

[62]:
```
   Age  Experience  Income  Family  CCAvg  Education  Mortgage  Personal Loan  \
0   25           1      49       4    1.6          1         0              0
1   45          19      34       3    1.5          1         0              0
2   39          15      11       1    1.0          1         0              0
3   35           9     100       1    2.7          2         0              0
4   35           8      45       4    1.0          2         0              0

   Securities Account  CD Account  Online  CreditCard
0                   1           0       0           0
1                   1           0       0           0
```

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

2 Logistics regression

[63]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix,␣
  ↪classification_report
x = df.drop('Personal Loan', axis=1)
y = df['Personal Loan']
```

[64]:

```
x.head()
```

[64]:

```
   Age  Experience  Income  Family  CCAvg  Education  Mortgage  \
0   25           1      49       4    1.6          1         0
1   45          19      34       3    1.5          1         0
2   39          15      11       1    1.0          1         0
3   35           9     100       1    2.7          2         0
4   35           8      45       4    1.0          2         0

   Securities Account  CD Account  Online  CreditCard
0                   1           0       0           0
1                   1           0       0           0
2                   0           0       0           0
3                   0           0       0           0
4                   0           0       0           1
```

[65]:

```
y.head()
```

[65]:

```
0    0
1    0
2    0
3    0
4    0
Name: Personal Loan, dtype: int64
```

[66]:

```
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.1,␣
  ↪random_state=42)
```

[67]:

```
print('x_train', x_train.shape)
print('x_test', x_test.shape)
print('y_train', y_train.shape)
print('y_test', y_test.shape)
```

```
x_train (4500, 11)
x_test (500, 11)
y_train (4500,)
y_test (500,)
```

[68]:

```
model = LogisticRegression()
model.fit(x_train,y_train)
```

```
C:\Users\Admin\anaconda3\Lib\site-
packages\sklearn\linear_model\_logistic.py:460: ConvergenceWarning:

lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
```

[68]:

```
LogisticRegression()
```

[69]:

```
y_pred = model.predict(x_test)
```

[70]:

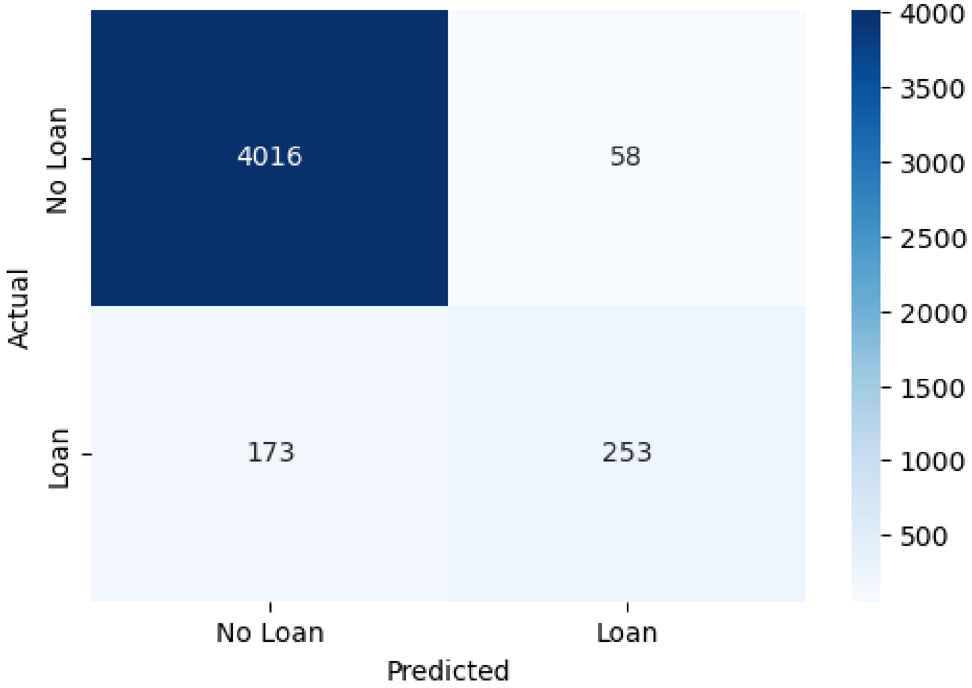
```
y_pred = model.predict(x_train)
accuracy = accuracy_score(y_train,y_pred)
conf_matrix = confusion_matrix(y_train,y_pred)
classi_report = classification_report(y_train,y_pred)
print('Accuracy of training data : ', accuracy)
print('\n\nConfusion matrix : \n',conf_matrix)
print('classifiction_report:\n\n',classi_report)
```

```
Accuracy of training data :  0.9486666666666667


Confusion matrix :
 [[4016   58]
 [ 173  253]]
classifiction_report:

               precision    recall  f1-score   support

           0       0.96      0.99      0.97      4074
           1       0.81      0.59      0.69       426

    accuracy                           0.95      4500
   macro avg       0.89      0.79      0.83      4500
```

```
weighted avg       0.94      0.95      0.95      4500
```

[71]:

```
plt.figure(figsize=(6,4))
sns.heatmap(conf_matrix, annot=True, fmt='g', cmap='Blues',xticklabels=['No␣
 ↪Loan','Loan'],
            yticklabels=['No Loan','Loan'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```



[72]:

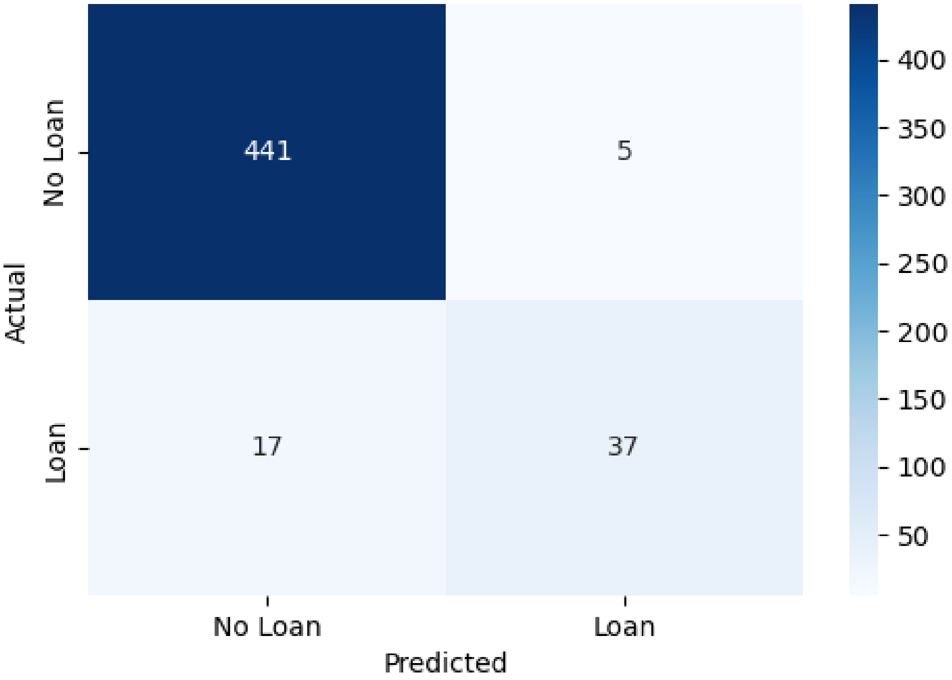
```
y_pred = model.predict(x_test)
accuracy = accuracy_score(y_test,y_pred)
conf_matrix = confusion_matrix(y_test,y_pred)
classi_report = classification_report(y_test,y_pred)
print('Accuracy of testing data : ', accuracy)
print('\n\nConfusion matrix : \n',conf_matrix)
print('classifiction_report:\n\n',classi_report)
```

```
Accuracy of testing data :  0.956


Confusion matrix :
 [[441   5]
 [ 17  37]]
```

```
classifiction_report:

              precision    recall  f1-score   support

           0       0.96      0.99      0.98       446
           1       0.88      0.69      0.77        54

    accuracy                           0.96       500
   macro avg       0.92      0.84      0.87       500
weighted avg       0.95      0.96      0.95       500
```

[73]:

```
plt.figure(figsize=(6,4))
sns.heatmap(conf_matrix, annot=True, fmt='g', cmap='Blues',xticklabels=['No␣
 ↪Loan','Loan'],
            yticklabels=['No Loan','Loan'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```



[ ]: