```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
```

```
df=pd.read_csv("/content/Bank_Personal_Loan_Modelling.csv")
```

```
df
```

	ID	Age	Experience	Income	ZIP Code	Family	CCAvg	Education	Mortgage	Personal Loan	Securities Account	CD Account	Online	CreditCard
0	1	25	1	49	91107	4	1.6	1	0	0	1	0	0	0
1	2	45	19	34	90089	3	1.5	1	0	0	1	0	0	0
2	3	39	15	11	94720	1	1.0	1	0	0	0	0	0	0
3	4	35	9	100	94112	1	2.7	2	0	0	0	0	0	0
4	5	35	8	45	91330	4	1.0	2	0	0	0	0	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4995	4996	29	3	40	92697	1	1.9	3	0	0	0	0	1	0
4996	4997	30	4	15	92037	4	0.4	1	85	0	0	0	1	0
4997	4998	63	39	24	93023	2	0.3	3	0	0	0	0	0	0
4998	4999	65	40	49	90034	3	0.5	2	0	0	0	0	1	0
4999	5000	28	4	83	92612	3	0.8	1	0	0	0	0	1	1

5000 rows × 14 columns

```
df.shape
```

```
(5000, 14)
```

```
df.columns
```

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

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 14 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   ID                  5000 non-null   int64  
 1   Age                 5000 non-null   int64  
 2   Experience          5000 non-null   int64  
 3   Income              5000 non-null   int64  
 4   ZIP Code            5000 non-null   int64  
 5   Family              5000 non-null   int64  
 6   CCAvg               5000 non-null   float64
 7   Education           5000 non-null   int64  
 8   Mortgage            5000 non-null   int64  
 9   Personal Loan       5000 non-null   int64  
 10  Securities Account  5000 non-null   int64  
 11  CD Account          5000 non-null   int64  
 12  Online              5000 non-null   int64  
 13  CreditCard          5000 non-null   int64  
dtypes: float64(1), int64(13)
memory usage: 547.0 KB
```

```
df.describe()
```

	ID	Age	Experience	Income	ZIP Code	Family	CCAvg	Education	Mortgage	Personal Loan
count	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000
mean	2500.500000	45.338400	20.104600	73.774200	93152.503000	2.396400	1.937938	1.881000	56.498800	0.096000
std	1443.520003	11.463166	11.467954	46.033729	2121.852197	1.147663	1.747659	0.839869	101.713802	0.294621
min	1.000000	23.000000	-3.000000	8.000000	9307.000000	1.000000	0.000000	1.000000	0.000000	0.000000
25%	1250.750000	35.000000	10.000000	39.000000	91911.000000	1.000000	0.700000	1.000000	0.000000	0.000000

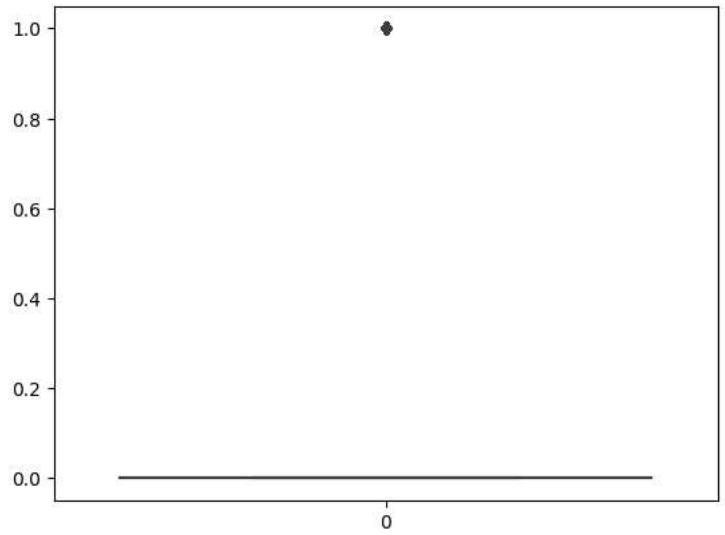
```
df.isnull().sum()
```

```
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
```

```
df.drop('ID',axis=1,inplace=True)
```

```
sns.boxplot(df['Personal Loan']);
plt.show
```

```
<function matplotlib.pyplot.show(close=None, block=None)>
```




```
fig, axis = plt.subplots (2, 2, figsize=(10, 10), sharex=False)
sns.distplot(df['Age'], bins=10,ax=axis[0,0]);
sns.distplot(df['Experience'], ax=axis [0,1],color='orange');
sns.distplot(df['CCAvg'], ax=axis[1,0], color='gray');
sns.distplot(df['Family'], ax=axis[1,1], color='yellow');
plt.show()
```

```
<ipython-input-12-908094a8f162>: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 `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

  sns.distplot(df['Age'], bins=10,ax=axis[0,0]);
<ipython-input-12-908094a8f162>: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 `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

  sns.distplot(df['Experience'], ax=axis [0,1],color='orange');
<ipython-input-12-908094a8f162>: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

  sns.distplot(df['CCAvg'], ax=axis[1,0], color='gray');
<ipython-input-12-908094a8f162>:5: 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).
```

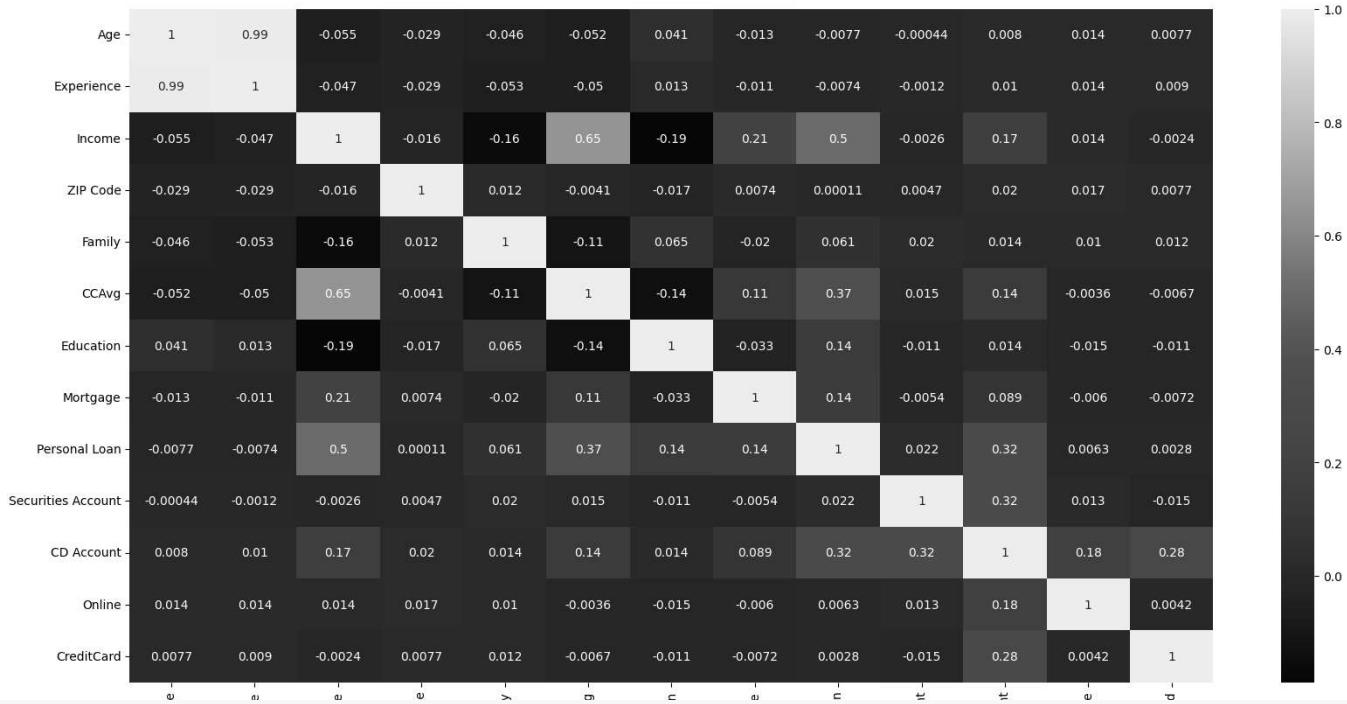
```
df['Income']=df['Income']/12
df['Mortgage']=df['Mortgage']/10
```

```
fig, axis = plt.subplots(1,2, figsize=(6,4), sharex=False)
sns.distplot(df['Income'], ax=axis[0], color='green');
sns.distplot(df['Mortgage'], ax=axis[1], color='red');
plt.show()
```

```
<ipython-input-15-18afdb615a6a>:2: UserWarning:

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

```
plt.figure(figsize=(20,10))
sns.heatmap(df.corr(),annot=True);
plt.show()
```




```
x = df.drop(['Personal Loan'], axis=1)
y = df['Personal Loan']
```

```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3,random_state=100)
```

```
from sklearn.linear_model import LogisticRegression
```

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

```
▾ LogisticRegression
LogisticRegression()
```

```
logiR_test = logiR.predict(x_test)
```

```
print("Classification Report")
print(classification_report(y_test, logiR_test))
```

```
Classification Report
              precision    recall  f1-score   support

           0       0.92      0.97      0.95      1342
           1       0.57      0.32      0.41       158

    accuracy                           0.90      1500
   macro avg       0.75      0.64      0.68      1500
weighted avg       0.89      0.90      0.89      1500
```

```
logiR_predict_train=logiR.predict_proba(x_train)[:,1] > 0.8
logiR_predict_test=logiR.predict_proba(x_test) [:,1]> 0.8
```

```
print("Classification Report")
cm =classification_report(y_test,logiR_predict_test, labels=[1,0])
print(cm)
```

```
Classification Report
              precision    recall  f1-score   support

           1       0.33      0.01      0.02       158
           0       0.90      1.00      0.94      1342
```

```
          accuracy                           0.89      1500
         macro avg       0.61      0.50      0.48      1500
      weighted avg       0.84      0.89      0.85      1500
```

```
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(x_train,y_train)
```

```
GaussianNB()
```

```
gnb_predict_test=gnb.predict_proba(x_test) [:,1] > 0.8
print(classification_report(y_test,gnb_predict_test, labels=[1,0]))
```

```
                  precision    recall  f1-score   support

               1       0.50      0.55      0.53       158
               0       0.95      0.94      0.94      1342

          accuracy                           0.90      1500
         macro avg       0.72      0.74      0.73      1500
      weighted avg       0.90      0.90      0.90      1500
```

```
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split

data = load_breast_cancer()

label_names = data['target_names']
labels = data['target']
feature_names = data['feature_names']
features = data['data']

print(label_names)
print(labels[0])
print(feature_names[0])
print(features[0])
```

```
['malignant' 'benign']
0
mean radius
[1.799e+01 1.038e+01 1.228e+02 1.001e+03 1.184e-01 2.776e-01 3.001e-01
 1.471e-01 2.419e-01 7.871e-02 1.095e+00 9.053e-01 8.589e+00 1.534e+02
 6.399e-03 4.904e-02 5.373e-02 1.587e-02 3.003e-02 6.193e-03 2.538e+01
 1.733e+01 1.846e+02 2.019e+03 1.622e-01 6.656e-01 7.119e-01 2.654e-01
 4.601e-01 1.189e-01]
```

```
X_train, X_test, y_train, y_test = train_test_split(features,labels,random_state=42)
```