In [6]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import OrdinalEncoder
from sklearn.model_selection import train_test_split
from scipy.stats import zscore
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
%matplotlib inline
sns.set(color_codes = True)
```

In [9]:

```
data=pd.read_csv('C:/Users/Anas Khanooni/Desktop/Supervised Learning/Bank_Personal_L
```

In [10]:

```
data.shape
```

Out[10]:

```
(5000, 14)
```

In [12]:

```
data.head()
```

Out[12]:

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

In [13]:

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

In [14]:

```
data.head()
```

Out[14]:

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

In [15]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 12 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Age                 5000 non-null   int64
 1   Experience          5000 non-null   int64  
 2   Income              5000 non-null   int64  
 3   Family              5000 non-null   int64  
 4   CCAvg               5000 non-null   float64
 5   Education           5000 non-null   int64  
 6   Mortgage            5000 non-null   int64  
 7   Personal Loan       5000 non-null   int64  
 8   Securities Account  5000 non-null   int64  
 9   CD Account          5000 non-null   int64  
 10  Online              5000 non-null   int64  
 11  CreditCard          5000 non-null   int64  
dtypes: float64(1), int64(11)
memory usage: 468.9 KB
```

In [16]:
```
target_feat = data['Personal Loan']
```

In [17]:
```
data.drop('Personal Loan',axis = 1, inplace = True)
```

In [18]:
```
target_feat.head(10)
```

Out[18]:
```
0    0
1    0
2    0
3    0
4    0
5    0
6    0
7    0
8    0
9    1
Name: Personal Loan, dtype: int64
```

In [20]:
```
data = pd.concat([data,target_feat],axis=1)
```

In [21]:
```
data.head()
```

Out[21]:

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

In [24]:
```
data.describe()
```

Out[24]:

	Age	Experience	Income	Family	CCAvg	Education	Mortgage
count	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000
mean	45.338400	20.104600	73.774200	2.396400	1.937938	1.881000	56.498800
std	11.463166	11.467954	46.033729	1.147663	1.747659	0.839869	101.713802
min	23.000000	-3.000000	8.000000	1.000000	0.000000	1.000000	0.000000
25%	35.000000	10.000000	39.000000	1.000000	0.700000	1.000000	0.000000
50%	45.000000	20.000000	64.000000	2.000000	1.500000	2.000000	0.000000

	Age	Experience	Income	Family	CCAvg	Education	Mortgage
75%	55.000000	30.000000	98.000000	3.000000	2.500000	3.000000	101.000000
max	67.000000	43.000000	224.000000	4.000000	10.000000	3.000000	635.000000

All the columns need inspection for dirty values such as '-3' in experience.

In [23]:

```
data.isnull().values.any()
```

Out[23]:

```
False
```

Task 2

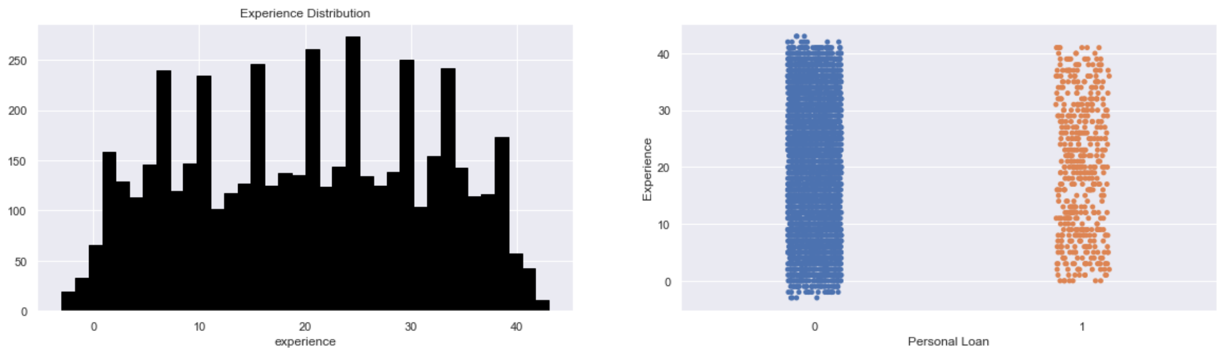
In [31]:

```
plt.figure(figsize=(20,5))
plt.subplot(1,2,1)
plt.hist(data['Experience'],color = 'black', edgecolor = 'black', bins = int(180/5))
plt.title('Experience Distribution');
plt.xlabel('experience');
plt.subplot(1,2,2)
sns.stripplot(data['Personal Loan'], data['Experience'])
```

```
C:\Users\Anas Khanooni\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Future
Warning: Pass the following variables as keyword args: x, y. From version 0.12, the
only valid positional argument will be `data`, and passing other arguments without a
n explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[31]:

```
<AxesSubplot:xlabel='Personal Loan', ylabel='Experience'>
```



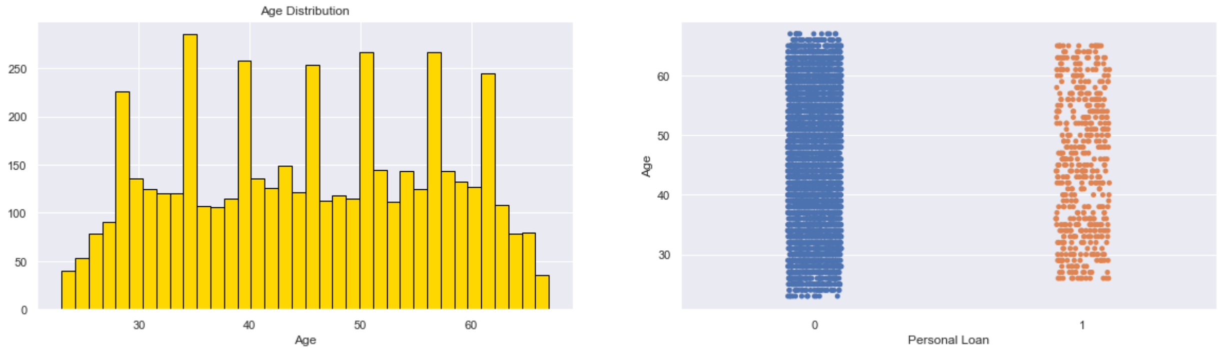
In [34]:

```
plt.figure(figsize=(20,5))
plt.subplot(1,2,1)
plt.hist(data['Age'],color = 'gold', edgecolor = 'black', bins = int(180/5));
plt.title('Age Distribution');
plt.xlabel('Age');
plt.subplot(1,2,2)
sns.stripplot(data['Personal Loan'], data['Age'])
```

```
C:\Users\Anas Khanooni\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Future
Warning: Pass the following variables as keyword args: x, y. From version 0.12, the
only valid positional argument will be `data`, and passing other arguments without a
n explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[34]:

```
<AxesSubplot:xlabel='Personal Loan', ylabel='Age'>
```



In [ ]:

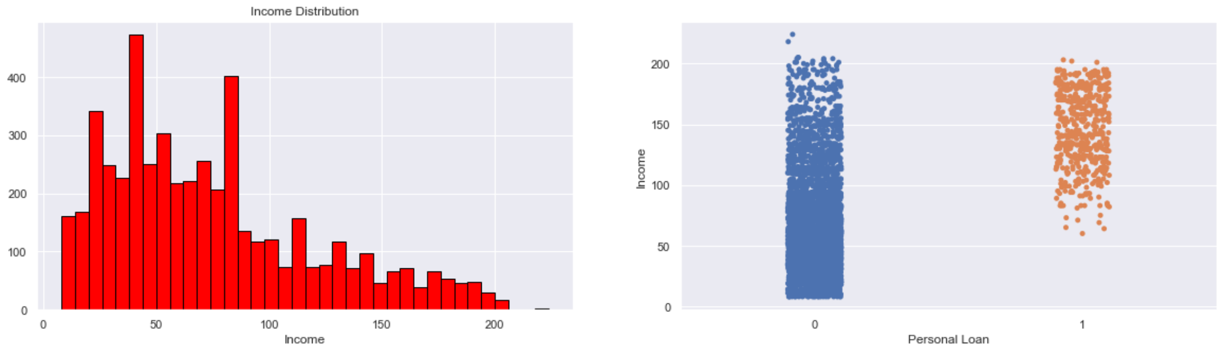
In [39]:

```
plt.figure(figsize=(20,5))
plt.subplot(1,2,1)
plt.hist(data['Income'],color = 'red', edgecolor = 'black', bins = int(180/5));
plt.title('Income Distribution');
plt.xlabel('Income');
plt.subplot(1,2,2)
sns.stripplot(data['Personal Loan'], data['Income'])
```

```
C:\Users\Anas Khanooni\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Future
Warning: Pass the following variables as keyword args: x, y. From version 0.12, the
only valid positional argument will be `data`, and passing other arguments without a
n explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[39]:

```
<AxesSubplot:xlabel='Personal Loan', ylabel='Income'>
```



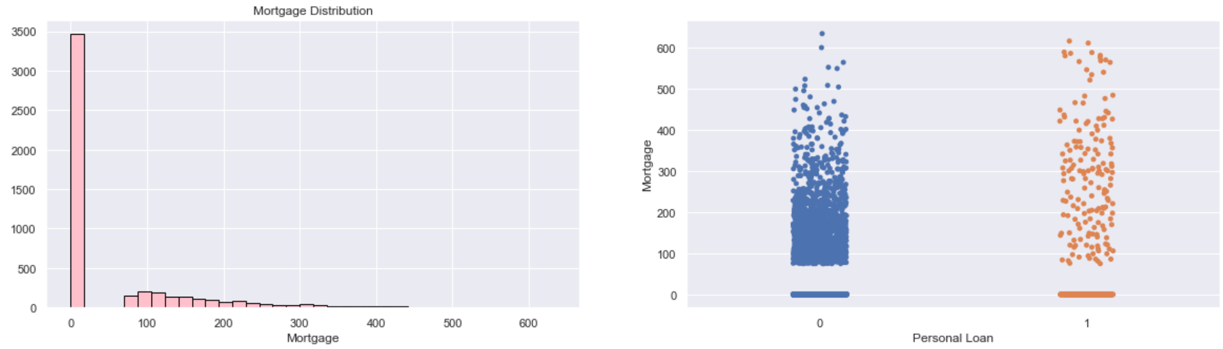
In [41]:

```
plt.figure(figsize=(20,5))
plt.subplot(1,2,1)
plt.hist(data['Mortgage'],color = 'pink', edgecolor = 'black', bins = int(180/5));
plt.title('Mortgage Distribution');
plt.xlabel('Mortgage');
plt.subplot(1,2,2)
sns.stripplot(data['Personal Loan'], data['Mortgage'])
```

```
C:\Users\Anas Khanooni\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Future
Warning: Pass the following variables as keyword args: x, y. From version 0.12, the
only valid positional argument will be `data`, and passing other arguments without a
n explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[41]:

```
<AxesSubplot:xlabel='Personal Loan', ylabel='Mortgage'>
```



The mortgage has a heavy left skew. House mortgage is shown to be majorly zero.

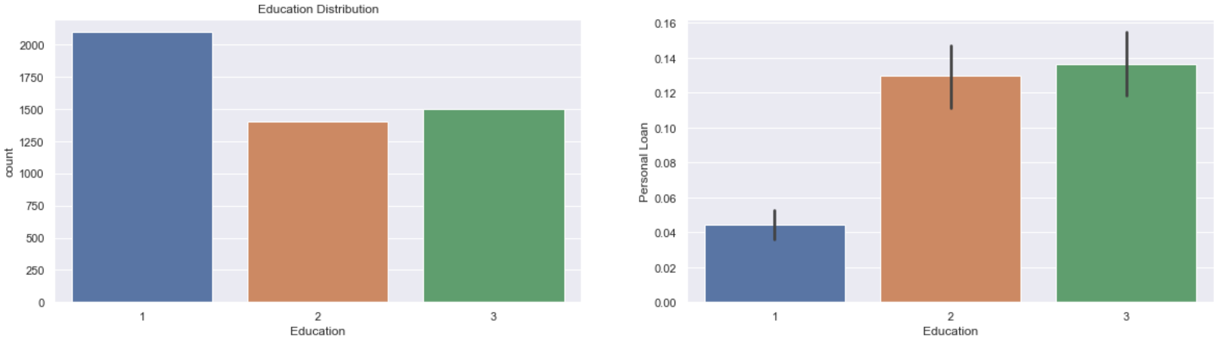
In [42]:

```
plt.figure(figsize=(20,5))
plt.subplot(1,2,1)
sns.countplot(data['Education']);
plt.title('Education Distribution');
plt.xlabel('Education');
plt.subplot(1,2,2)
sns.barplot(data['Education'],data['Personal Loan'])
```

```
C:\Users\Anas Khanooni\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Future
Warning: Pass the following variable as a keyword arg: x. From version 0.12, the onl
y valid positional argument will be `data`, and passing other arguments without an e
xplicit keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\Anas Khanooni\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Future
Warning: Pass the following variables as keyword args: x, y. From version 0.12, the
only valid positional argument will be `data`, and passing other arguments without a
n explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[42]:

```
<AxesSubplot:xlabel='Education', ylabel='Personal Loan'>
```



It is observed that under graduates are higher in number than graduates and professionals who seem to be equal in numbers.

In [43]:

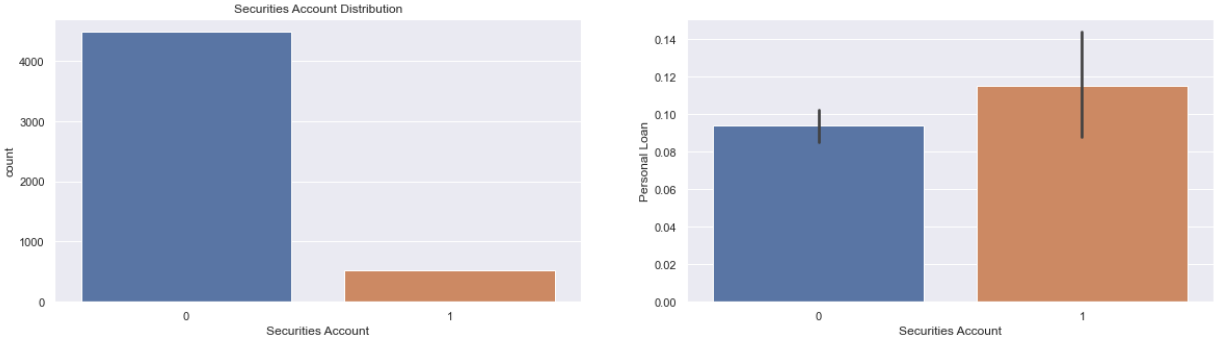
```
plt.figure(figsize=(20,5))
plt.subplot(1,2,1)
sns.countplot(data['Securities Account']);
plt.title('Securities Account Distribution');
plt.xlabel('Securities Account');
plt.subplot(1,2,2)
sns.barplot(data['Securities Account'],data['Personal Loan'])
```

```
C:\Users\Anas Khanooni\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Future
Warning: Pass the following variable as a keyword arg: x. From version 0.12, the onl
y valid positional argument will be `data`, and passing other arguments without an e
xplicit keyword will result in an error or misinterpretation.
```

```
  warnings.warn(
C:\Users\Anas Khanooni\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Future
Warning: Pass the following variables as keyword args: x, y. From version 0.12, the
only valid positional argument will be `data`, and passing other arguments without a
n explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[43]:

```
<AxesSubplot:xlabel='Securities Account', ylabel='Personal Loan'>
```



Very less people have securities account than not.

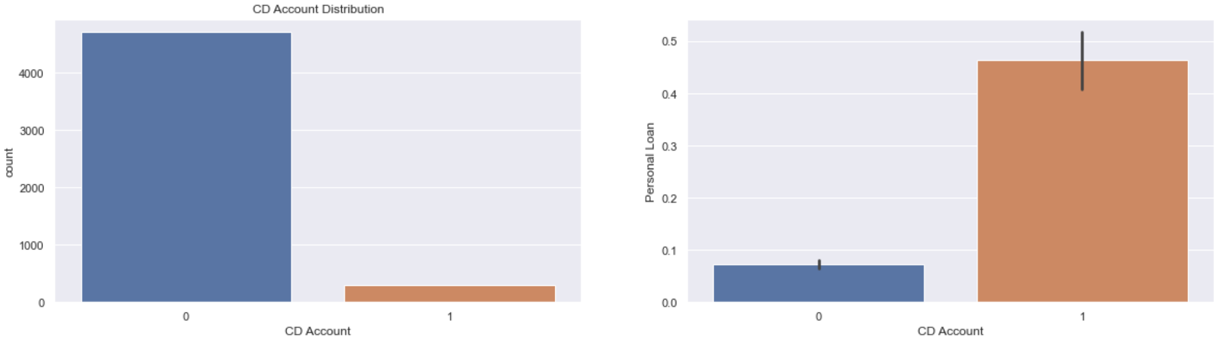
In [44]:

```
plt.figure(figsize=(20,5))
plt.subplot(1,2,1)
sns.countplot(data['CD Account']);
plt.title('CD Account Distribution');
plt.xlabel('CD Account');
plt.subplot(1,2,2)
sns.barplot(data['CD Account'],data['Personal Loan'])
```

```
C:\Users\Anas Khanooni\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Future
Warning: Pass the following variable as a keyword arg: x. From version 0.12, the onl
y valid positional argument will be `data`, and passing other arguments without an e
xplicit keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\Anas Khanooni\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Future
Warning: Pass the following variables as keyword args: x, y. From version 0.12, the
only valid positional argument will be `data`, and passing other arguments without a
n explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[44]:

```
<AxesSubplot:xlabel='CD Account', ylabel='Personal Loan'>
```



In [45]:

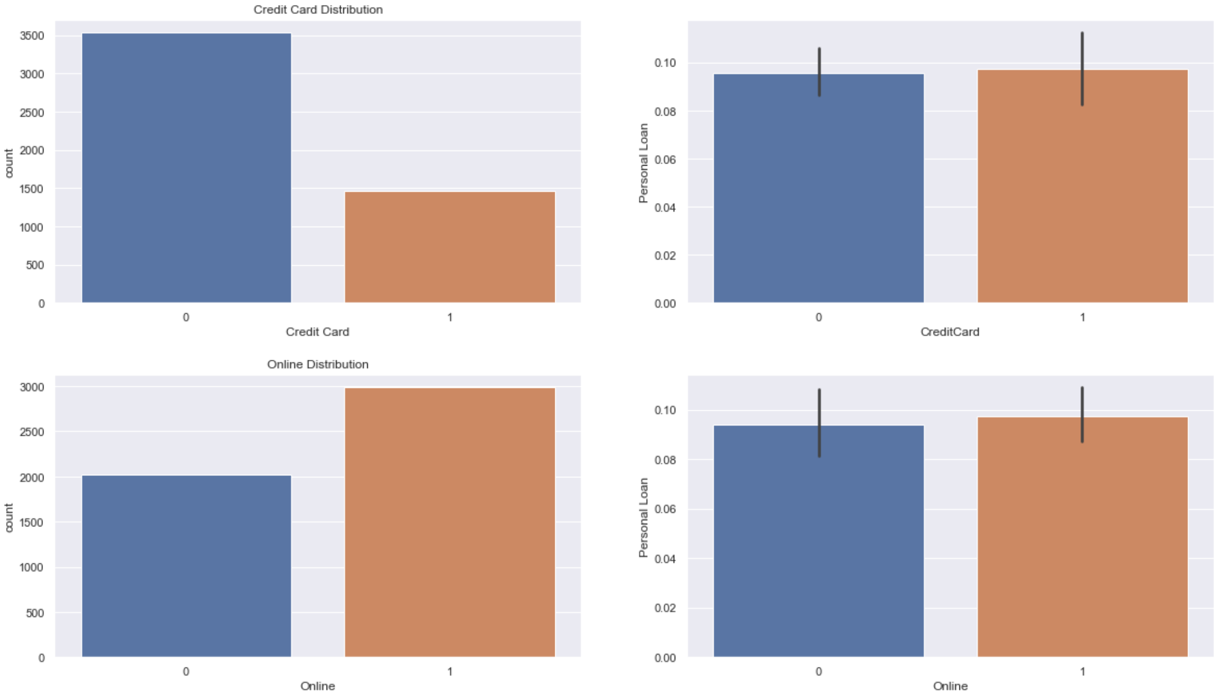
```
plt.figure(figsize=(20,5))
plt.subplot(1,2,1)
sns.countplot(data['CreditCard']);
plt.title('Credit Card Distribution');
plt.xlabel('Credit Card');
plt.subplot(1,2,2)
sns.barplot(data['CreditCard'],data['Personal Loan'])

plt.figure(figsize=(20,5))
plt.subplot(1,2,1)
sns.countplot(data['Online']);
plt.title('Online Distribution');
plt.xlabel('Online');
plt.subplot(1,2,2)
sns.barplot(data['Online'],data['Personal Loan'])
```

```
C:\Users\Anas Khanooni\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Future
Warning: Pass the following variable as a keyword arg: x. From version 0.12, the onl
y valid positional argument will be `data`, and passing other arguments without an e
xplicit keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\Anas Khanooni\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Future
Warning: Pass the following variables as keyword args: x, y. From version 0.12, the
only valid positional argument will be `data`, and passing other arguments without a
n explicit keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\Anas Khanooni\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Future
Warning: Pass the following variable as a keyword arg: x. From version 0.12, the onl
y valid positional argument will be `data`, and passing other arguments without an e
xplicit keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\Anas Khanooni\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Future
Warning: Pass the following variables as keyword args: x, y. From version 0.12, the
only valid positional argument will be `data`, and passing other arguments without a
n explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[45]:

```
<AxesSubplot:xlabel='Online', ylabel='Personal Loan'>
```



From the countplot, it is seen that more people are online than not.

Task 3

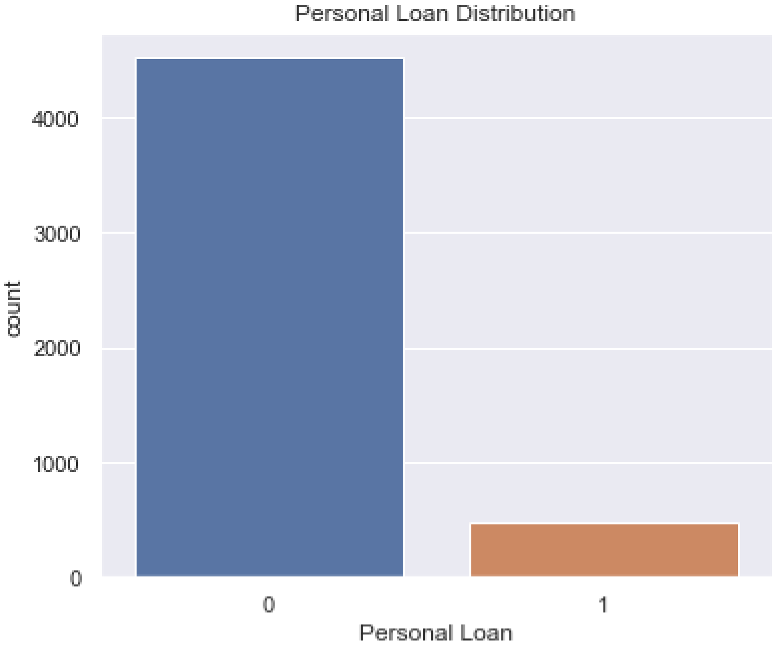
In [46]:

```
plt.figure(figsize=(6,5))
sns.countplot(data['Personal Loan'])
plt.title('Personal Loan Distribution')
```

```
C:\Users\Anas Khanooni\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Future
Warning: Pass the following variable as a keyword arg: x. From version 0.12, the onl
y valid positional argument will be `data`, and passing other arguments without an e
xplicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[46]:

```
Text(0.5, 1.0, 'Personal Loan Distribution')
```



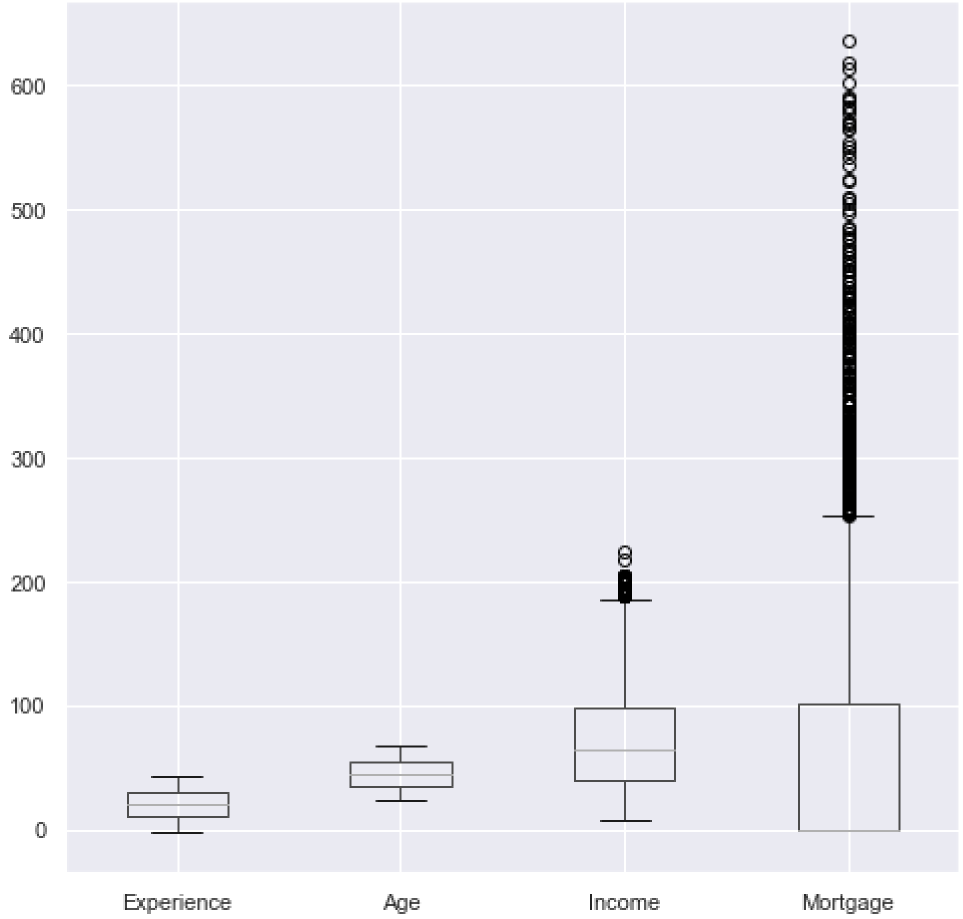
Data Processing

In [47]:

```
data.boxplot(column = ['Experience','Age','Income','Mortgage'], return_type = 'axes'
```

Out[47]:

```
<AxesSubplot:>
```



In [48]:

```
print(data.var())
```

```
Age                    131.404166
Experience             131.513962
```

```
Income                 2119.104235
Family                    1.317130
CCAvg                     3.054312
Education                 0.705380
Mortgage              10345.697538
Securities Account        0.093519
CD Account                0.056763
Online                    0.240678
CreditCard                0.207606
Personal Loan             0.086801
dtype: float64
```

In [49]:

```
data.skew()
```

Out[49]:

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

Removing Negative Values from "EXPERIENCE"

In [50]:

```
data['Experience'].describe()
```

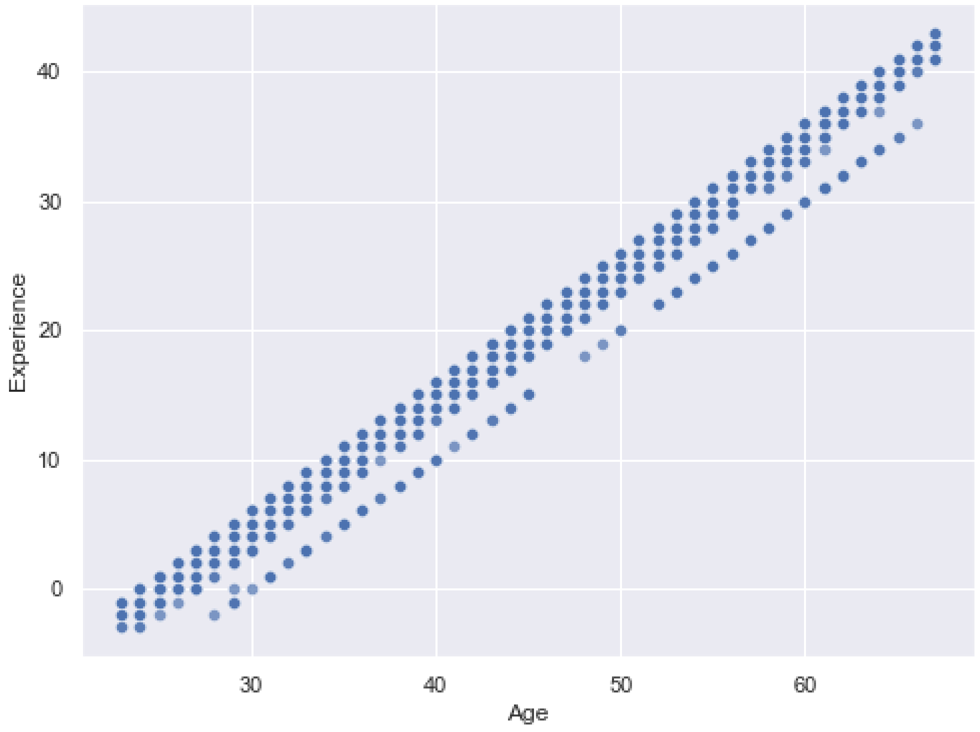
Out[50]:

```
count    5000.000000
mean       20.104600
std        11.467954
min        -3.000000
25%        10.000000
50%        20.000000
75%        30.000000
max        43.000000
Name: Experience, dtype: float64
```

In [51]:

```
plt.figure(figsize=(8,6))
sns.scatterplot(data['Age'],data['Experience'],alpha=0.7)
plt.show()
```

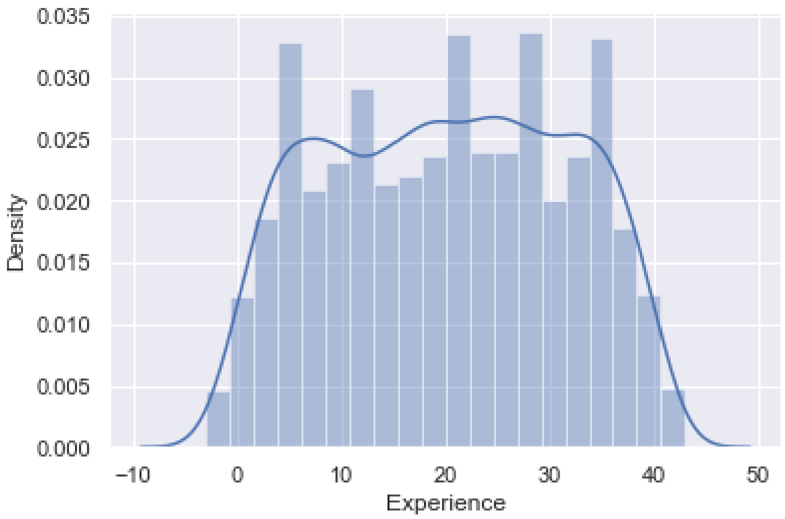
```
C:\Users\Anas Khanooni\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Future
Warning: Pass the following variables as keyword args: x, y. From version 0.12, the
only valid positional argument will be `data`, and passing other arguments without a
n explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



In [52]:
```
sns.distplot(data['Experience'])
```

```
C:\Users\Anas Khanooni\anaconda3\lib\site-packages\seaborn\distributions.py:2551: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a future ver
sion. Please adapt your code to use either `displot` (a figure-level function with s
imilar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[52]:
```
<AxesSubplot:xlabel='Experience', ylabel='Density'>
```



It appears that there's no reliable mode to replace negatives with

In [53]:
```
data['Experience'][data['Age']<30].describe(include='all')
```

Out[53]:
```
count    488.000000
mean       1.969262
std        1.868654
min       -3.000000
25%        1.000000
50%        2.000000
75%        3.000000
max        5.000000
Name: Experience, dtype: float64
```

In [55]:

```
target_labels = data['Experience'][data['Age']<30].mode().value_counts()
target_mode = data['Experience'][data['Age']<30].mode()

df = pd.concat([target_mode,target_labels],axis=0)
```

In [56]:

```
print(df)
```

```
0    3
3    1
dtype: int64
```

The mean would be a better replacement for negatives.

In [57]:

```
data['Experience'][data['Experience']<0] = 1.969262
```

```
<ipython-input-57-60c41ecfd7d5>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  data['Experience'][data['Experience']<0] = 1.969262
```

In [58]:

```
data.describe()
```

Out[58]:

	Age	Experience	Income	Family	CCAvg	Education	Mortgage
count	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000
mean	45.338400	20.140080	73.774200	2.396400	1.937938	1.881000	56.498800
std	11.463166	11.406153	46.033729	1.147663	1.747659	0.839869	101.713802
min	23.000000	0.000000	8.000000	1.000000	0.000000	1.000000	0.000000
25%	35.000000	10.000000	39.000000	1.000000	0.700000	1.000000	0.000000
50%	45.000000	20.000000	64.000000	2.000000	1.500000	2.000000	0.000000
75%	55.000000	30.000000	98.000000	3.000000	2.500000	3.000000	101.000000
max	67.000000	43.000000	224.000000	4.000000	10.000000	3.000000	635.000000

Outlier Identification

In [61]:

```
feat_desc  = data.describe()
feat_desc
```

Out[61]:

	Age	Experience	Income	Family	CCAvg	Education	Mortgage
count	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000
mean	45.338400	20.140080	73.774200	2.396400	1.937938	1.881000	56.498800
std	11.463166	11.406153	46.033729	1.147663	1.747659	0.839869	101.713802
min	23.000000	0.000000	8.000000	1.000000	0.000000	1.000000	0.000000
25%	35.000000	10.000000	39.000000	1.000000	0.700000	1.000000	0.000000
50%	45.000000	20.000000	64.000000	2.000000	1.500000	2.000000	0.000000
75%	55.000000	30.000000	98.000000	3.000000	2.500000	3.000000	101.000000

	Age	Experience	Income	Family	CCAvg	Education	Mortgage
max	67.000000	43.000000	224.000000	4.000000	10.000000	3.000000	635.000000

In [63]:

```python
outlier_list = []
for i in list(feat_desc):
    q1 = feat_desc[i]['25%']
    print(f'the q1 of {i} is {q1}')
    q3 = feat_desc[i]['75%']
    print(f'the q3 of {i} is {q3}')
    iqr =  abs(q1 - q3)
    print('.')
    print('.')
    print(f'the interquartile range of {i} is {iqr}')
    mx = feat_desc[i]['max']
    mn = feat_desc[i]['min']
    print('are there any outliers?')
    up_lim = q3 + 1.5*iqr
    dwn_lim = q1 - 1.5*iqr
    print('.')
    print('.')
    if ((mx<=up_lim) & (mn>=dwn_lim)):
        print(f'there are no outliers in {i}')

    else:
        print('outliers!')
        outlier_list.append(i)

    print('.')
    print('.')
    print('.')
```

```
the q1 of Age is 35.0
the q3 of Age is 55.0
.
.
the interquartile range of Age is 20.0
are there any outliers?
.
.
there are no outliers in Age
.
.
.
the q1 of Experience is 10.0
the q3 of Experience is 30.0
.
.
the interquartile range of Experience is 20.0
are there any outliers?
.
.
there are no outliers in Experience
.
.
.
the q1 of Income is 39.0
the q3 of Income is 98.0
.
.
the interquartile range of Income is 59.0
```

```
are there any outliers?
.
.
outliers!
.
.
.
the q1 of Family is 1.0
the q3 of Family is 3.0
.
.
the interquartile range of Family is 2.0
are there any outliers?
.
.
there are no outliers in Family
.
.
.
the q1 of CCAvg is 0.7
the q3 of CCAvg is 2.5
.
.
the interquartile range of CCAvg is 1.8
are there any outliers?
.
.
outliers!
.
.
.
the q1 of Education is 1.0
the q3 of Education is 3.0
.
.
the interquartile range of Education is 2.0
are there any outliers?
.
.
there are no outliers in Education
.
.
.
the q1 of Mortgage is 0.0
the q3 of Mortgage is 101.0
.
.
the interquartile range of Mortgage is 101.0
are there any outliers?
.
.
outliers!
.
.
.
the q1 of Securities Account is 0.0
the q3 of Securities Account is 0.0
.
.
the interquartile range of Securities Account is 0.0
are there any outliers?
.
.
outliers!
.
.
.
the q1 of CD Account is 0.0
the q3 of CD Account is 0.0
```

```
.
.
the interquartile range of CD Account is 0.0
are there any outliers?
.
.
outliers!
.
.
.
the q1 of Online is 0.0
the q3 of Online is 1.0
.
.
the interquartile range of Online is 1.0
are there any outliers?
.
.
there are no outliers in Online
.
.
.
the q1 of CreditCard is 0.0
the q3 of CreditCard is 1.0
.
.
the interquartile range of CreditCard is 1.0
are there any outliers?
.
.
there are no outliers in CreditCard
.
.
.
the q1 of Personal Loan is 0.0
the q3 of Personal Loan is 0.0
.
.
the interquartile range of Personal Loan is 0.0
are there any outliers?
.
.
outliers!
.
.
.
```

Identifies columns from a dataset with outliers and collects them in a separate list.

In [65]:

```
feat_desc[outlier_list]
```

Out[65]:

	Income	CCAvg	Mortgage	Securities Account	CD Account	Personal Loan
count	5000.000000	5000.000000	5000.000000	5000.000000	5000.00000	5000.000000
mean	73.774200	1.937938	56.498800	0.104400	0.06040	0.096000
std	46.033729	1.747659	101.713802	0.305809	0.23825	0.294621
min	8.000000	0.000000	0.000000	0.000000	0.00000	0.000000
25%	39.000000	0.700000	0.000000	0.000000	0.00000	0.000000
50%	64.000000	1.500000	0.000000	0.000000	0.00000	0.000000
75%	98.000000	2.500000	101.000000	0.000000	0.00000	0.000000
max	224.000000	10.000000	635.000000	1.000000	1.00000	1.000000

In [66]:

```
outlier_list = outlier_list[0:3]
```

In [67]:

```
outlier_list
```

Out[67]:

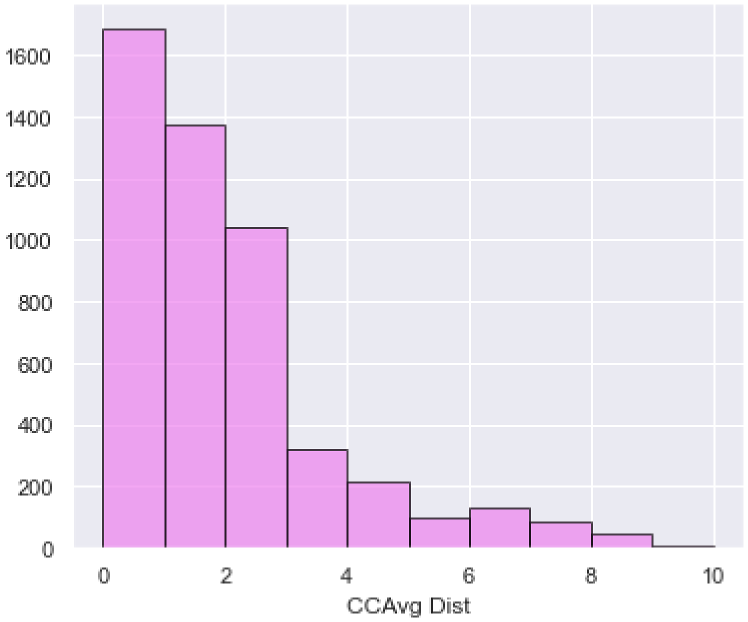
```
['Income', 'CCAvg', 'Mortgage']
```

In [69]:

```
plt.figure(figsize = (6,5))
plt.hist(data['CCAvg'],color='violet',edgecolor='black',alpha = 0.7)
plt.xlabel('CCAvg Dist')
```

Out[69]:

```
Text(0.5, 0, 'CCAvg Dist')
```



CCAvg has a left skew

In [71]:

```
def remove_outliers(i):
    outliers=[]   #a fresh outlier list to accomodate outliers for each category col
    print('')
    print(f'Calculated outliers for {i}:')
    print('')
    q1 = feat_desc[i]['25%']
    q3 = feat_desc[i]['75%']
    iqr =  abs(q1 - q3)
    mx = feat_desc[i]['max']
    mn = feat_desc[i]['min']
    up_lim = q3 + 1.5*iqr
    dwn_lim = q1 - 1.5*iqr
    filterate = data[i][~((data[i]<dwn_lim)|(data[i]>up_lim))] #wipes outlier values
                                                                                #be tr


    return filterate
```

Collects outliers for any column, filters them out and returns the column

In [72]:

```
filterate = remove_outliers('Income')
```

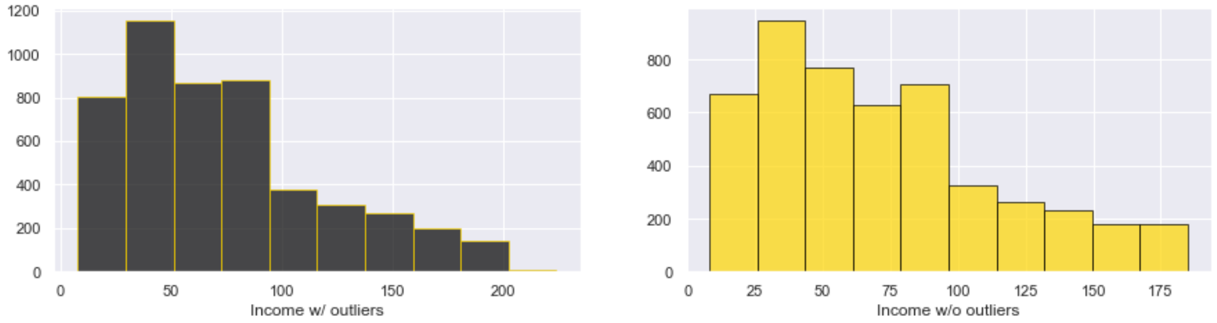
```
Calculated outliers for Income:
```

In [73]:

```
plt.figure(figsize=(15,3.5))
plt.subplot(1,2,1)
plt.hist(data['Income'],color = 'black', edgecolor = 'gold', alpha = 0.7)
plt.xlabel('Income w/ outliers')
plt.subplot(1,2,2)
plt.hist(filterate, color = 'gold', edgecolor = 'black',alpha = 0.7)
plt.xlabel('Income w/o outliers')
```

Out[73]:

```
Text(0.5, 0, 'Income w/o outliers')
```



In [74]:

```
data['Income'] = filterate #income columns is replaced with refined data
```

In [75]:

```
outlier_list
```

Out[75]:

```
['Income', 'CCAvg', 'Mortgage']
```

In [76]:

```
filterate = remove_outliers('CCAvg')
```

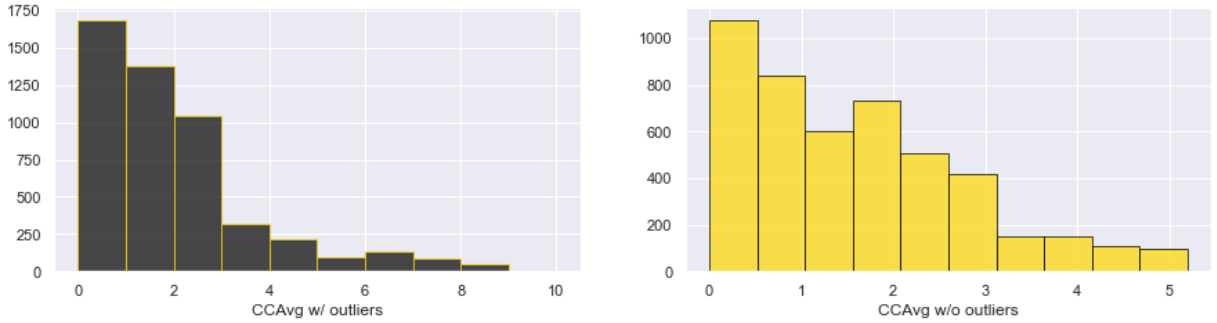
```
Calculated outliers for CCAvg:
```

In [77]:

```
plt.figure(figsize = (15,3.5))
plt.subplot(1,2,1)
plt.hist(data['CCAvg'], color = 'black',edgecolor = 'gold', alpha = 0.7)
plt.xlabel('CCAvg w/ outliers')
plt.subplot(1,2,2)
plt.hist(filterate,color = 'gold', edgecolor = 'black', alpha = 0.7)
plt.xlabel('CCAvg w/o outliers')
```

Out[77]:

```
Text(0.5, 0, 'CCAvg w/o outliers')
```



In [78]:

```
data['CCAvg'] = filterate
```

In [79]:

```
outlier_list
```

Out[79]:

```
['Income', 'CCAvg', 'Mortgage']
```

In [80]:

```
filterate = remove_outliers('Mortgage')
```

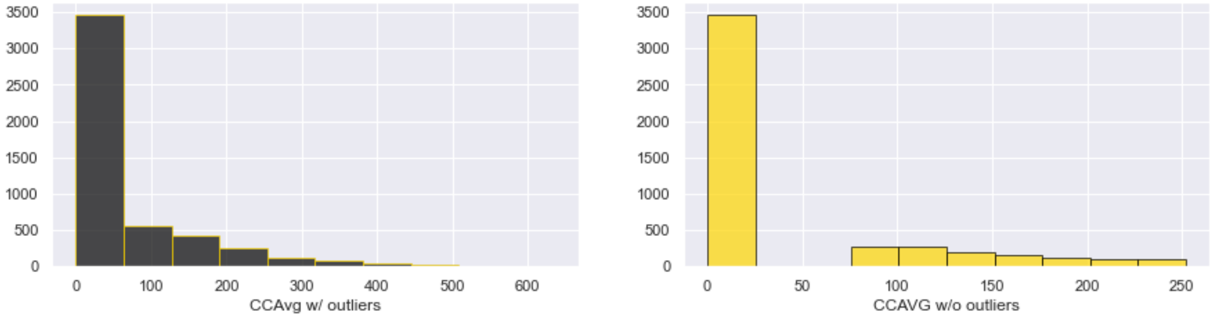
```
Calculated outliers for Mortgage:
```

In [81]:

```
plt.figure(figsize=(15,3.5))
plt.subplot(1,2,1)
plt.hist(data['Mortgage'],color = 'black',edgecolor='gold',alpha=0.7)
plt.xlabel('CCAvg w/ outliers')
plt.subplot(1,2,2)
plt.hist(filterate,color='gold',edgecolor='black',alpha=0.7)
plt.xlabel('CCAVG w/o outliers')
```

Out[81]:

```
Text(0.5, 0, 'CCAVG w/o outliers')
```



In [82]:

```
data['Mortgage'] = filterate
```

In [83]:

```
data.describe()
```

Out[83]:

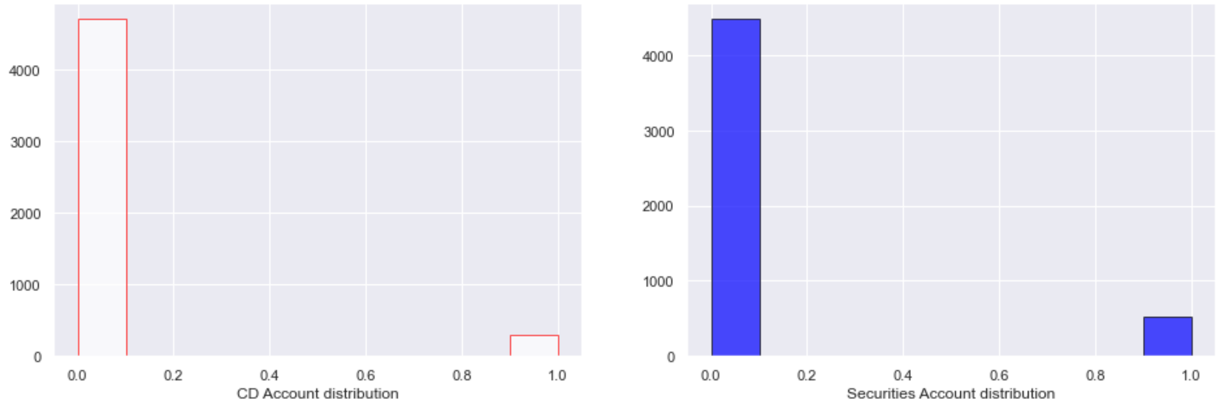
	Age	Experience	Income	Family	CCAvg	Education	Mortgage
count	5000.000000	5000.000000	4904.000000	5000.000000	4676.000000	5000.000000	4709.000000
mean	45.338400	20.140080	71.407626	2.396400	1.597923	1.881000	38.011467
std	11.463166	11.406153	43.221791	1.147663	1.189172	0.839869	68.100514
min	23.000000	0.000000	8.000000	1.000000	0.000000	1.000000	0.000000
25%	35.000000	10.000000	38.000000	1.000000	0.600000	1.000000	0.000000
50%	45.000000	20.000000	63.000000	2.000000	1.500000	2.000000	0.000000
75%	55.000000	30.000000	94.000000	3.000000	2.300000	3.000000	81.000000
max	67.000000	43.000000	185.000000	4.000000	5.200000	3.000000	252.000000

In [85]:

```
plt.figure(figsize=(16,5))
plt.subplot(1,2,1)
plt.hist(data['CD Account'], color = 'white',edgecolor = 'red',alpha = 0.7)
plt.xlabel('CD Account distribution')
plt.subplot(1,2,2)
plt.hist(data['Securities Account'], color = 'blue',edgecolor = 'black',alpha = 0.7)
plt.xlabel('Securities Account distribution')
```

Out[85]:

```
Text(0.5, 0, 'Securities Account distribution')
```



REMOVING MISSING VALUES

In [86]:

```
data.isna().any()
```

Out[86]:

```
Age                   False
Experience            False
Income                 True
Family                False
CCAvg                  True
Education             False
Mortgage               True
Securities Account    False
CD Account            False
Online                False
CreditCard            False
Personal Loan         False
dtype: bool
```

In [88]:

```
inc_missing = pd.DataFrame(data.Income.isnull())
data[inc_missing['Income'] == True]
```

Out[88]:

	Age	Experience	Income	Family	CCAvg	Education	Mortgage	Securities Account	CD Account	Online
18	46	21.0	NaN	2	NaN	3	0.0	0	0	0
47	37	12.0	NaN	4	0.2	3	211.0	1	1	1
53	50	26.0	NaN	3	2.1	3	240.0	0	0	1
59	31	5.0	NaN	2	4.5	1	NaN	0	0	0
303	49	25.0	NaN	4	3.0	1	NaN	0	0	0
...	...	...	...	...	...	...	...	...	...	...
4659	28	4.0	NaN	1	NaN	1	0.0	0	0	0
4670	52	26.0	NaN	1	1.7	1	0.0	0	0	1
4895	45	20.0	NaN	2	2.8	1	0.0	0	0	1
4981	34	9.0	NaN	2	3.0	1	122.0	0	0	1
4993	45	21.0	NaN	2	NaN	1	0.0	0	0	1

96 rows × 12 columns

In [89]:

```
mort_missing = pd.DataFrame(data.Mortgage.isna())
data[mort_missing['Mortgage'] == True]
```

Out[89]:

	Age	Experience	Income	Family	CCAvg	Education	Mortgage	Securities Account	CD Account	Online
22	29	5.0	62.0	1	1.2	1	NaN	0	0	1
39	38	13.0	80.0	4	0.7	3	NaN	0	0	1
42	32	7.0	132.0	4	1.1	2	NaN	0	0	1
59	31	5.0	NaN	2	4.5	1	NaN	0	0	0
66	62	36.0	105.0	2	2.8	1	NaN	0	0	0
...	...	...	...	...	...	...	...	...	...	...
4859	34	8.0	165.0	1	NaN	3	NaN	0	0	0
4865	50	24.0	133.0	4	1.4	2	NaN	0	0	0
4899	54	29.0	85.0	4	1.3	3	NaN	0	0	1
4942	52	26.0	109.0	1	2.4	1	NaN	0	1	1
4963	32	6.0	98.0	2	4.5	3	NaN	0	0	0

291 rows × 12 columns

In [90]:

```
ccavg_missing = pd.DataFrame(data.CCAvg.isna())
data[ccavg_missing['CCAvg'] == True]
```

Out[90]:

	Age	Experience	Income	Family	CCAvg	Education	Mortgage	Securities Account	CD Account	Online
9	34	9.0	180.0	1	NaN	3	0.0	0	0	0
18	46	21.0	NaN	2	NaN	3	0.0	0	0	0
44	46	20.0	104.0	1	NaN	1	0.0	0	0	1
55	41	17.0	139.0	2	NaN	1	0.0	0	0	1
61	47	21.0	125.0	1	NaN	1	112.0	1	0	0
...	...	...	...	...	...	...	...	...	...	...
4908	40	16.0	138.0	2	NaN	1	0.0	0	0	1
4911	46	22.0	153.0	2	NaN	1	0.0	0	0	0
4937	33	8.0	162.0	1	NaN	1	0.0	0	1	1
4980	29	5.0	135.0	3	NaN	1	0.0	0	1	1
4993	45	21.0	NaN	2	NaN	1	0.0	0	0	1

324 rows × 12 columns

In [91]:

```
print(inc_missing.shape)
print(mort_missing.shape)
print(ccavg_missing.shape)
```

```
(5000, 1)
(5000, 1)
(5000, 1)
```

In [92]:

```
median_filler = lambda x:x.fillna(x.median())
data = data.apply(median_filler, axis =0)
```

Run to replace NaN values with median in the respective columns.

In [93]:

```
data.isna().any()
```

Out[93]:

```
Age                   False
Experience            False
Income                False
Family                False
CCAvg                 False
Education             False
Mortgage              False
Securities Account    False
CD Account            False
Online                False
CreditCard            False
Personal Loan         False
dtype: bool
```

Therefore, all the missing values have been filled with median.

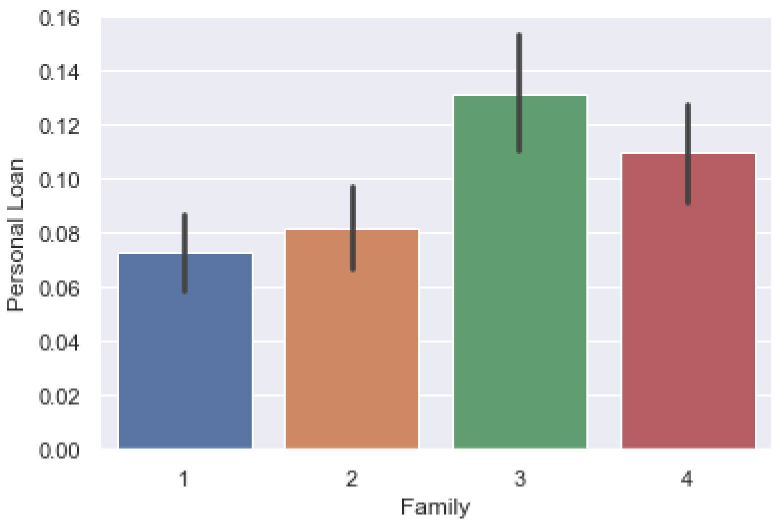
In [94]:

```
sns.barplot(data['Family'],data['Personal Loan'])
```

```
C:\Users\Anas Khanooni\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Future
Warning: Pass the following variables as keyword args: x, y. From version 0.12, the
only valid positional argument will be `data`, and passing other arguments without a
n explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[94]:

```
<AxesSubplot:xlabel='Family', ylabel='Personal Loan'>
```



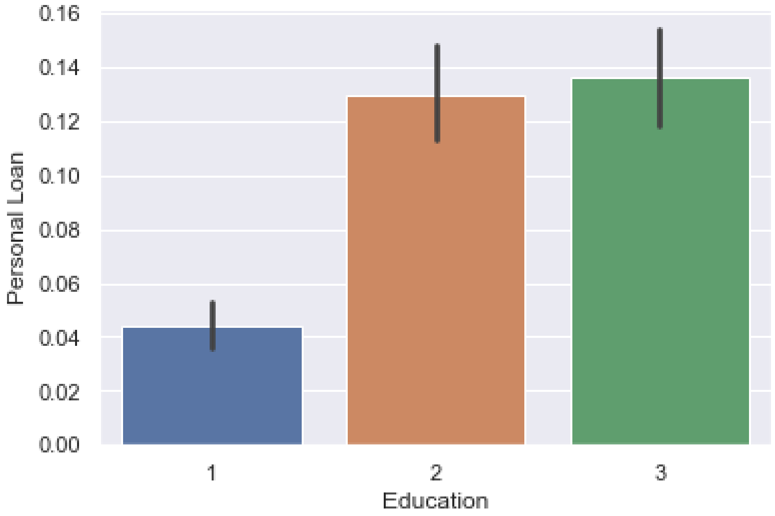
In [95]:

```
sns.barplot(data['Education'],data['Personal Loan'])
```

```
C:\Users\Anas Khanooni\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Future
Warning: Pass the following variables as keyword args: x, y. From version 0.12, the
only valid positional argument will be `data`, and passing other arguments without a
n explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[95]:

```
<AxesSubplot:xlabel='Education', ylabel='Personal Loan'>
```



In [96]:
```
data['Family'] = data['Family'].astype(str)
```

In [97]:
```
data['Family'].dtype
```

Out[97]:
```
dtype('O')
```

In [100…
```
data['Education'] = data['Education'].astype(str)
```

In [101…
```
data['Education'].dtype
```

Out[101…
```
dtype('O')
```

In [102…
```
data.dtypes
```

Out[102…
```
Age                     int64
Experience            float64
Income                float64
Family                 object
CCAvg                 float64
Education              object
Mortgage              float64
Securities Account      int64
CD Account              int64
Online                  int64
CreditCard              int64
Personal Loan           int64
dtype: object
```

In [103…
```
ordinal_encoder= OrdinalEncoder()
```

In [104…
```
ordinal_cat = ordinal_encoder.fit_transform(data[['Education','Family']])
```

In [105…
```
data[['Education','Family']] = ordinal_cat
```

In [106…
```
data.dtypes
```

Out[106…
```
Age                     int64
Experience            float64
Income                float64
Family                float64
CCAvg                 float64
Education             float64
Mortgage              float64
Securities Account      int64
CD Account              int64
```

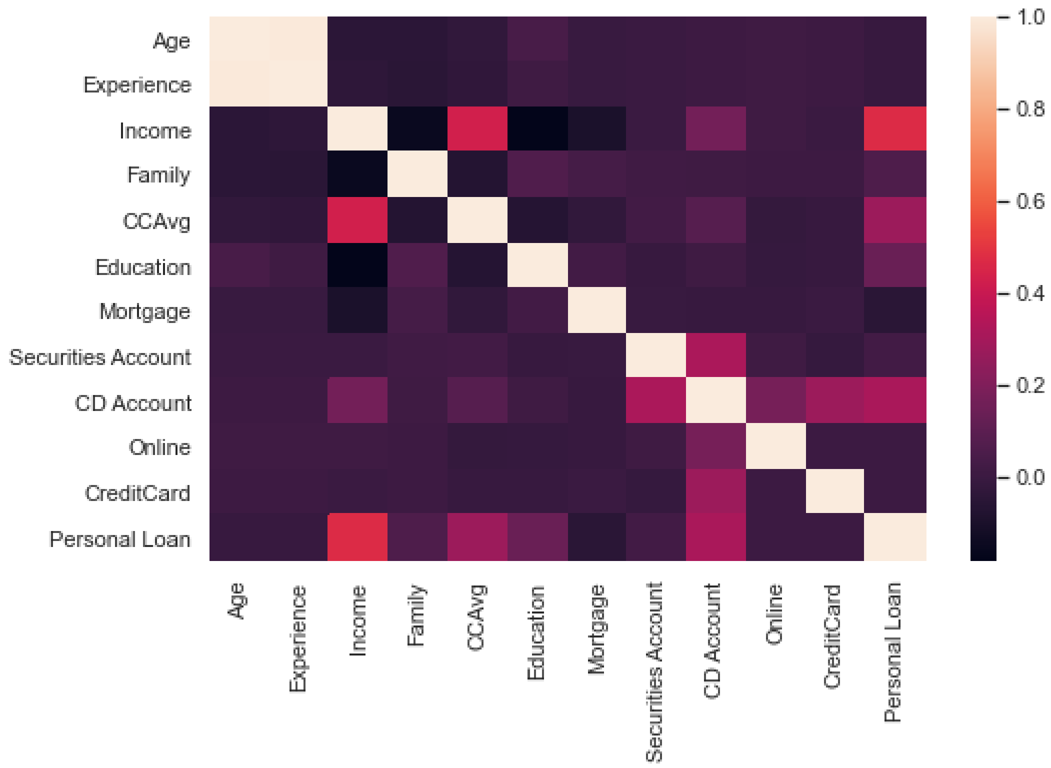
```
Online                 int64
CreditCard             int64
Personal Loan          int64
dtype: object
```

In [107…

```
plt.figure(figsize=(8,5))
ht_mp = data.corr()
sns.heatmap(ht_mp)
```

Out[107…

```
<AxesSubplot:>
```



TASK 4

In [108…

```
data.head()
```

Out[108…

	Age	Experience	Income	Family	CCAvg	Education	Mortgage	Securities Account	CD Account	Online	Cr
0	25	1.0	49.0	3.0	1.6	0.0	0.0	1	0	0	
1	45	19.0	34.0	2.0	1.5	0.0	0.0	1	0	0	
2	39	15.0	11.0	0.0	1.0	0.0	0.0	0	0	0	
3	35	9.0	100.0	0.0	2.7	1.0	0.0	0	0	0	
4	35	8.0	45.0	3.0	1.0	1.0	0.0	0	0	0	

In [109…

```
X = data.drop('Personal Loan',axis=1)
y = data['Personal Loan']
```

In [110…

```
X
```

Out[110…

	Age	Experience	Income	Family	CCAvg	Education	Mortgage	Securities Account	CD Account	Online

	Age	Experience	Income	Family	CCAvg	Education	Mortgage	Securities Account	CD Account	Online
0	25	1.0	49.0	3.0	1.6	0.0	0.0	1	0	0
1	45	19.0	34.0	2.0	1.5	0.0	0.0	1	0	0
2	39	15.0	11.0	0.0	1.0	0.0	0.0	0	0	0
3	35	9.0	100.0	0.0	2.7	1.0	0.0	0	0	0
4	35	8.0	45.0	3.0	1.0	1.0	0.0	0	0	0
...	...	...	...	...	...	...	...	...	...	...
4995	29	3.0	40.0	0.0	1.9	2.0	0.0	0	0	1
4996	30	4.0	15.0	3.0	0.4	0.0	85.0	0	0	1
4997	63	39.0	24.0	1.0	0.3	2.0	0.0	0	0	0
4998	65	40.0	49.0	2.0	0.5	1.0	0.0	0	0	1
4999	28	4.0	83.0	2.0	0.8	0.0	0.0	0	0	1

5000 rows × 11 columns

In [111…

```
y
```

Out[111…

```
0       0
1       0
2       0
3       0
4       0
       ..
4995    0
4996    0
4997    0
4998    0
4999    0
Name: Personal Loan, Length: 5000, dtype: int64
```

In [112…

```
X_scaled = X.apply(zscore)
```

In [113…

```
x_train, x_test, y_train, y_test = train_test_split(X_scaled,y,test_size=0.30,random
```

In [114…

```
x_train
```

Out[114…

	Age	Experience	Income	Family	CCAvg	Education	Mortgage	Securities Account	Acco
1366	1.279146	1.215248	-0.893268	-0.345432	-1.122987	-1.049078	0.977837	-0.341423	-0.25
3452	1.366391	1.478290	-1.126826	0.525991	-1.036040	0.141703	-0.536892	-0.341423	-0.25
2252	1.104657	1.039887	-0.706422	0.525991	-0.166571	-1.049078	-0.536892	-0.341423	-0.25
2758	1.453636	1.390610	-0.846557	-0.345432	-0.775199	1.332484	-0.536892	2.928915	-0.25
2436	0.668434	0.776844	-0.753134	0.525991	-0.079624	-1.049078	-0.536892	-0.341423	-0.25
...	...	...	...	...	...	...	...	...	
3603	0.493944	0.426121	-0.612999	1.397414	-1.296881	-1.049078	-0.536892	-0.341423	-0.25
4722	-0.465747	-0.363005	-0.192596	-1.216855	-0.079624	1.332484	-0.536892	-0.341423	-0.25

	Age	Experience	Income	Family	CCAvg	Education	Mortgage	Securities Account	Acco
3340	-1.425438	-1.502855	-0.402798	1.397414	0.181216	1.332484	-0.536892	-0.341423	-0.25
3064	1.191902	1.127567	0.274519	0.525991	2.441836	-1.049078	-0.536892	-0.341423	-0.25
3398	-0.465747	-0.538367	-0.215952	-0.345432	0.702898	0.141703	-0.536892	-0.341423	-0.25

3500 rows × 11 columns

In [115…

```
y_train
```

Out[115…

```
1366    0
3452    0
2252    0
2758    0
2436    0
       ..
3603    0
4722    0
3340    0
3064    0
3398    0
Name: Personal Loan, Length: 3500, dtype: int64
```

In [116…

```
loanees = data.loc[data['Personal Loan']==1]
nolonees = data.loc[data['Personal Loan']==0]
```

In [117…

```
print(f'loanees number : {len(loanees)}, percentage: {round(len(loanees)/len(y),3)}'
print(f'nonlonees number : {len(nolonees)}, percentage: {round(len(nolonees)/len(y),
```

```
loanees number : 480, percentage: 0.096
nonlonees number : 4520, percentage: 0.904
```

In [118…

```
print(f'train loanees number :{len(y_train[y_train == 1])}, percentage : {round(len(
print(f'train non loanees number :{len(y_train[y_train == 0])}, percentage : {round(
```

```
train loanees number :343, percentage : 0.0686
train non loanees number :3157, percentage : 0.6314
```

In [119…

```
print(f'test loanees number :{len(y_test[y_test == 1])}, percentage : {round(len(y_t
print(f'test non loanees number :{len(y_test[y_test == 0])}, percentage : {round(len
```

```
test loanees number :137, percentage : 0.0274
test non loanees number :1363, percentage : 0.2726
```

LOGISTIC, KNN & NAIVE BAYES

In [120…

```
    x_train.describe()
```

Out[120…

	Age	Experience	Income	Family	CCAvg	Education	Mortgage
count	3500.000000	3500.000000	3500.000000	3500.000000	3500.000000	3500.000000	3500.000000
mean	0.000115	-0.001662	0.007896	0.022557	-0.000806	0.000510	-0.003403
std	1.001143	1.001970	1.009462	0.998713	1.000837	1.002865	0.991865
min	-1.948906	-1.765897	-1.477162	-1.216855	-1.383828	-1.049078	-0.536892
25%	-0.901970	-0.889090	-0.753134	-1.216855	-0.775199	-1.049078	-0.536892

	Age	Experience	Income	Family	CCAvg	Education	Mortgage
50%	-0.029524	-0.012282	-0.192596	-0.345432	-0.079624	0.141703	-0.536892
75%	0.842923	0.864525	0.508076	1.397414	0.529004	1.332484	-0.536892
max	1.889859	2.004375	2.656804	1.397414	3.137411	1.332484	3.242431

Logistic Regression

In [121...
```
model = LogisticRegression(solver="liblinear")
```

In [122...
```
model.fit(x_train,y_train)
y_pred = model.predict(x_test)
```

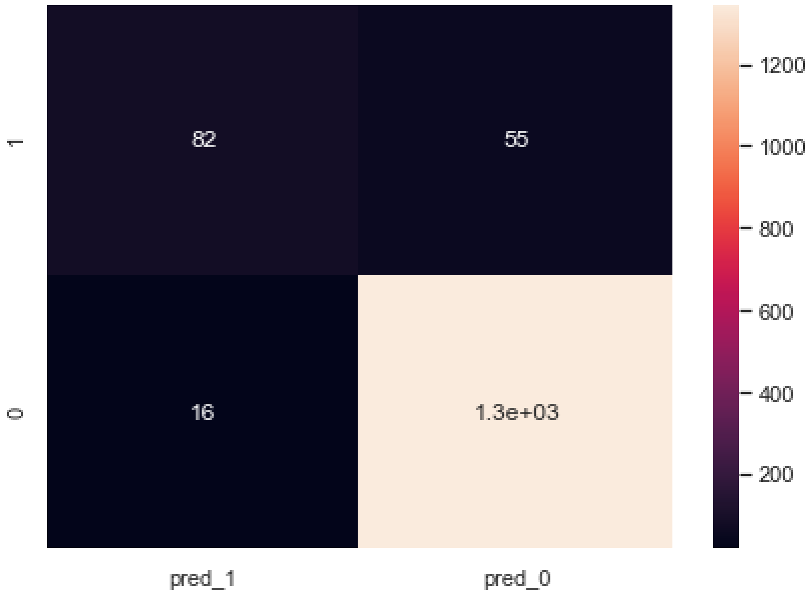
In [123...
```
mod_score = model.score(x_test,y_test)
print(mod_score)
```

```
0.9526666666666667
```

In [124...
```
cm_log = metrics.confusion_matrix(y_test,y_pred,labels=[1,0])

df_cmlog = pd.DataFrame(cm_log, index = [i for i in['1','0']],columns = [i for i in
plt.figure(figsize = (7,5))
sns.heatmap(df_cmlog, annot = True)
```

Out[124...
```
<AxesSubplot:>
```



In [125...
```
log_report = metrics.classification_report(y_test,y_pred,labels=[1,0])
print(metrics.classification_report(y_test,y_pred,labels=[1,0]))
```

```
              precision    recall  f1-score   support

           1       0.84      0.60      0.70       137
           0       0.96      0.99      0.97      1363

    accuracy                           0.95      1500
   macro avg       0.90      0.79      0.84      1500
weighted avg       0.95      0.95      0.95      1500
```

G. Naive Bayes

In [126…
```
nb_model = GaussianNB()
nb_model.fit(x_train,y_train.ravel())
```

Out[126…
```
GaussianNB()
```

In [127…
```
train_pred = nb_model.predict(x_train)

print(f'train accuracy: {round(metrics.accuracy_score(y_train,train_pred),4)}')
```

```
train accuracy: 0.9057
```

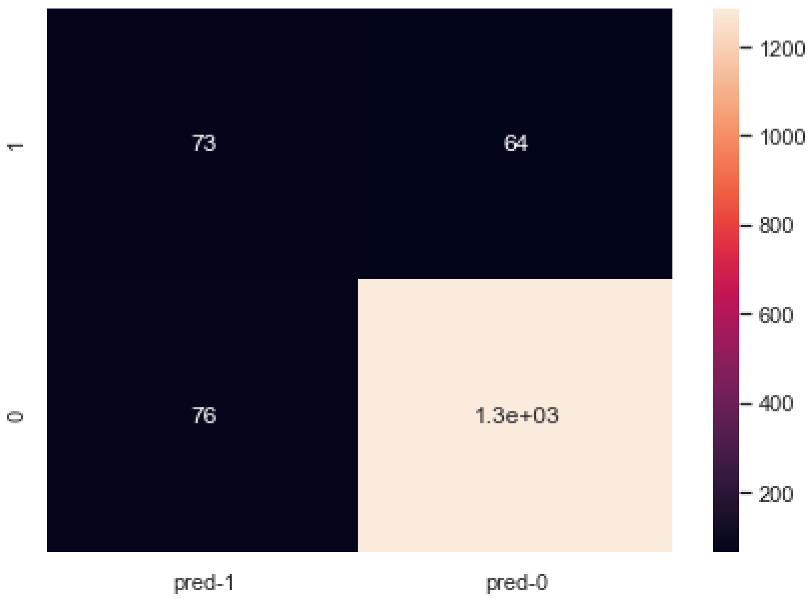
In [128…
```
y_pred = nb_model.predict(x_test)
print(f'test accuracy: {round(metrics.accuracy_score(y_test,y_pred),4)}')
```

```
test accuracy: 0.9067
```

Task 6

In [129…
```
cm_nb = metrics.confusion_matrix(y_test,y_pred,labels=[1,0])
df_cmnb = pd.DataFrame(cm_nb, index = [i for i in ['1','0']], columns = [i for i in
plt.figure(figsize=(7,5))
sns.heatmap(df_cmnb, annot = True)
```

Out[129…
```
<AxesSubplot:>
```



In [130…
```
knn_report = metrics.classification_report(y_test,y_pred,labels=[1,0])
print(metrics.classification_report(y_test,y_pred,labels=[1,0]))
```

```
              precision    recall  f1-score   support

           1       0.49      0.53      0.51       137
           0       0.95      0.94      0.95      1363

    accuracy                           0.91      1500
   macro avg       0.72      0.74      0.73      1500
weighted avg       0.91      0.91      0.91      1500
```

The KNN model has commited the least "false negative" misclassifications proving higher accuracy among the models.

Task 7

In [133…

```
print('logistic reg report')
print('')
print(log_report)
print('')
print('naive bayes report')
print('')
print(naive_report)
print('')
print('knn report')
print('')
print(knn_report)
```

```
logistic reg report

              precision    recall  f1-score   support

           1       0.84      0.60      0.70       137
           0       0.96      0.99      0.97      1363

    accuracy                           0.95      1500
   macro avg       0.90      0.79      0.84      1500
weighted avg       0.95      0.95      0.95      1500


naive bayes report

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-133-61aaad863d60> in <module>
      5 print('naive bayes report')
      6 print('')
----> 7 print(naive_report)
      8 print('')
      9 print('knn report')

NameError: name 'naive_report' is not defined
```

NAIVE BAYES COMPARE TO THE OTHER TWO

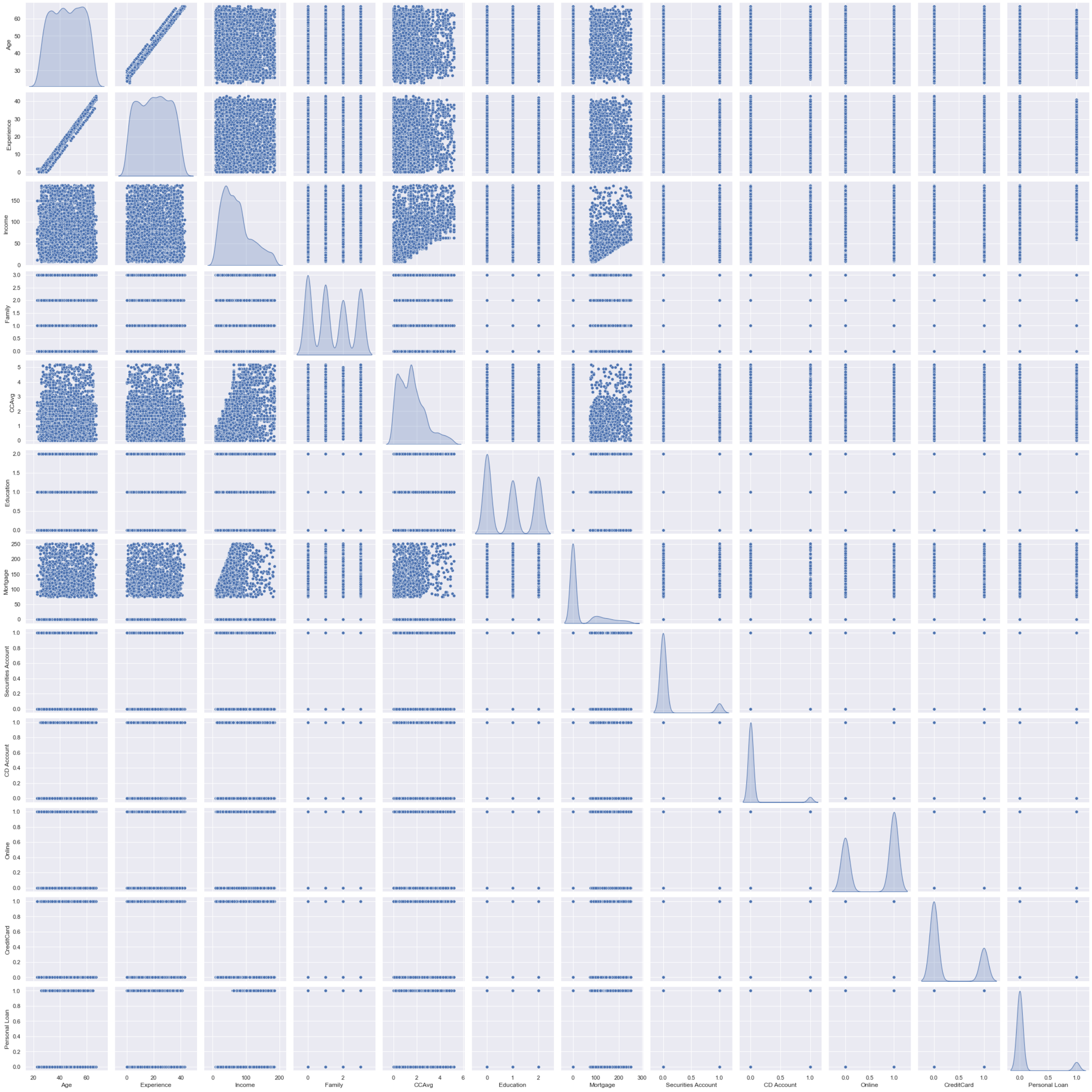
In [134…

```
plt.figure(figsize=(9,9))
sns.pairplot(data, diag_kind = 'kde')
```

Out[134…

```
<seaborn.axisgrid.PairGrid at 0x239be2299d0>
```

```
<Figure size 648x648 with 0 Axes>
```

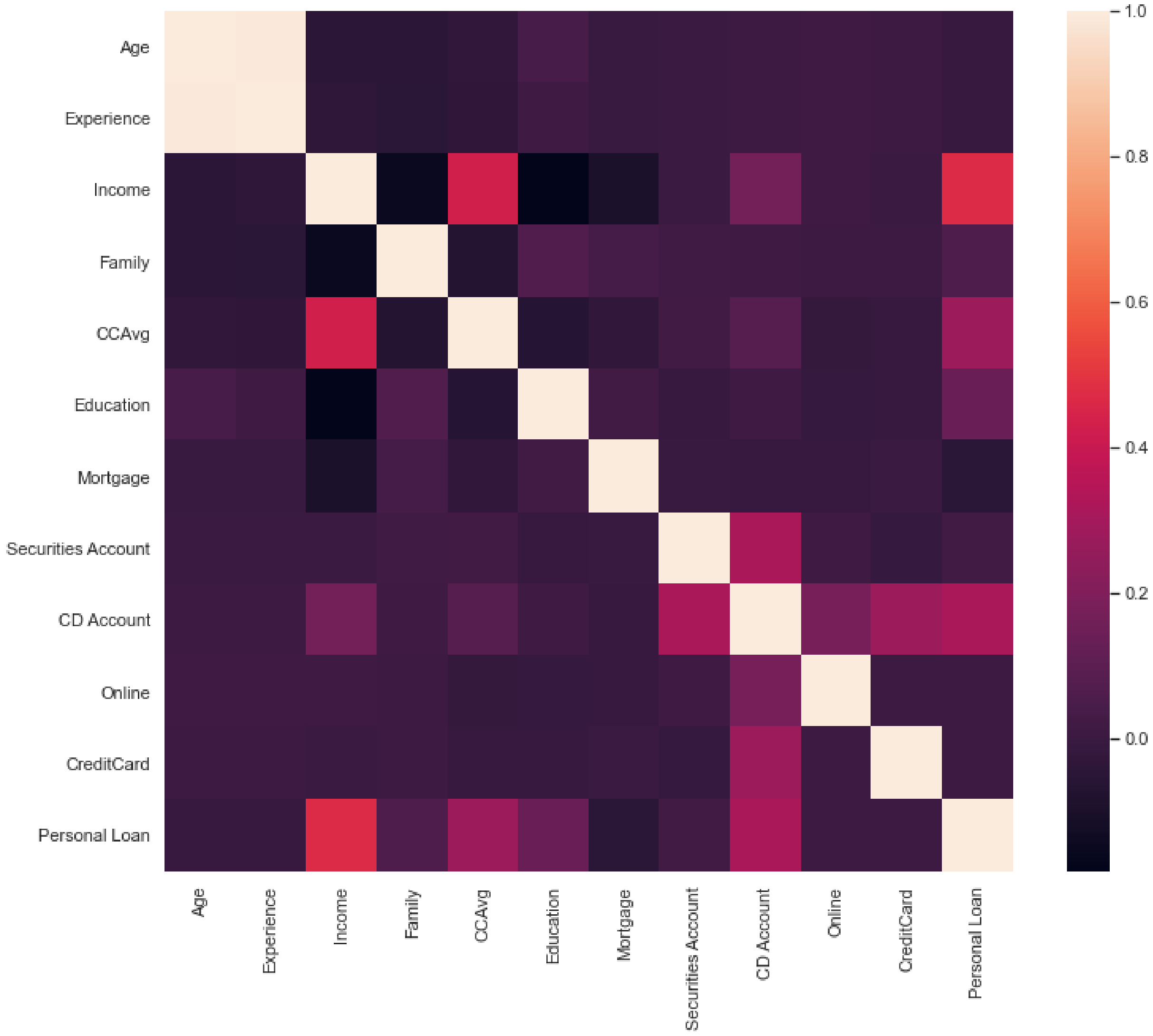


In [135…

```
plt.figure(figsize=(12,10))
sns.heatmap(data.corr())
```

Out[135…

```
<AxesSubplot:>
```



Conclusion

The classification goal is to predict the likelihood of a liability customer buying personal loans.

A bank wants a new marketing campaign; so that they need information about the correlation between the variables given in the dataset.

Here I used 4 classification models to study

But we can use SVM also as all the Kernels have good accuracy as well.