Import necessary modules : Numpy, Pandas(Data wrangling), Seaborn & Matplotlib(Data Visualisation)

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

```
df= pd.read_excel("bank.xlsx" , sheet_name=1)
```

```
df.head(10)
```

Show Top 10 Records

```
   ID  Age  Experience  Income  ZIP Code  Family  CCAvg     Education  \
0   1   25           1      49     91107       4    1.6     Undergrad
1   2   45          19      34     90089       3    1.5     Undergrad
2   3   39          15      11     94720       1    1.0     Undergrad
3   4   35           9     100     94112       1    2.7      Graduate
4   5   35           8      45     91330       4    1.0      Graduate
5   6   37          13      29     92121       4    0.4      Graduate
6   7   53          27      72     91711       2    1.5      Graduate
7   8   50          24      22     93943       1    0.3  Professional
8   9   35          10      81     90089       3    0.6      Graduate
9  10   34           9     180     93023       1    8.9  Professional

   Mortgage Personal Loan Securities Account CD Account Online CreditCard
0         0            No                Yes         No     No         No
1         0            No                Yes         No     No         No
2         0            No                 No         No     No         No
3         0            No                 No         No     No         No
4         0            No                 No         No     No        Yes
5       155            No                 No         No    Yes         No
6         0            No                 No         No    Yes         No
7         0            No                 No         No     No        Yes
8       104            No                 No         No    Yes         No
9         0           Yes                 No         No     No         No
```

Show Columns of DataFrame

```
df.columns
```

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

Show Data Types

```
df.dtypes
```

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

Show information about columns

There are no null values.

7 variables are continuous of int64 type , 6 variables are categorical of object type.

```
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   object
 8   Mortgage            5000 non-null   int64
 9   Personal Loan       5000 non-null   object
 10  Securities Account  5000 non-null   object
 11  CD Account          5000 non-null   object
 12  Online              5000 non-null   object
 13  CreditCard          5000 non-null   object
```

```
dtypes: float64(1), int64(7), object(6)
memory usage: 547.0+ KB
```

Drop all the duplicates from DataFrame.

```
df.drop_duplicates()
```

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

      Mortgage Personal Loan Securities Account CD Account Online CreditCard
0            0            No                Yes         No     No         No
1            0            No                Yes         No     No         No
2            0            No                 No         No     No         No
3            0            No                 No         No     No         No
4            0            No                 No         No     No        Yes
...        ...           ...                ...        ...    ...        ...
4995         0            No                 No         No    Yes         No
4996        85            No                 No         No    Yes         No
4997         0            No                 No         No     No         No
4998         0            No                 No         No    Yes         No
4999         0            No                 No         No    Yes        Yes

[5000 rows x 14 columns]
```

Describe Function.

There is high variation between mean of ID , ZiP code and rest variables so we need to do data preprocessing.

```
df.describe()
```

```
                ID          Age   Experience       Income      ZIP Code  \
count  5000.000000  5000.000000  5000.000000  5000.000000   5000.000000
```

```
mean    2500.500000    45.338400    20.104600    73.774200  93152.503000
std     1443.520003    11.463166    11.467954    46.033729   2121.852197
min        1.000000    23.000000    -3.000000     8.000000   9307.000000
25%     1250.750000    35.000000    10.000000    39.000000  91911.000000
50%     2500.500000    45.000000    20.000000    64.000000  93437.000000
75%     3750.250000    55.000000    30.000000    98.000000  94608.000000
max     5000.000000    67.000000    43.000000   224.000000  96651.000000

            Family        CCAvg     Mortgage
count  5000.000000  5000.000000  5000.000000
mean      2.396400     1.937913    56.498800
std       1.147663     1.747666   101.713802
min       1.000000     0.000000     0.000000
25%       1.000000     0.700000     0.000000
50%       2.000000     1.500000     0.000000
75%       3.000000     2.500000   101.000000
max       4.000000    10.000000   635.000000
```

Personal loan have two values: Yes/No.

```
df['Personal Loan'].unique()
```

```
array(['No', 'Yes'], dtype=object)
```

CD Account have two values Yes/No.

```
df['CD Account'].unique()
```

```
array(['No', 'Yes'], dtype=object)
```

```
df.columns
```

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

Experience have total unique values of 47.

```
df['Experience'].nunique()
```

```
47
```

```
df['Income'].nunique()
```

```
162
```

Income have total unique values of 162.

```
df['ZIP Code'].nunique()
```

```
467
```

ZIP Code contains 467 unique values.

```
df['Family'].unique()
```

```
array([4, 3, 1, 2])
```

Family contain total unique values 4,3,2,1 showing 4 members, 3 members, 2 members, 1 member.

```
df['CCAvg'].nunique()
```

```
108
```

CCAvg contain 108 unique values.

```
df['Education'].unique()
```

```
array(['Undergrad', 'Graduate', 'Professional'], dtype=object)
```

Education contain three values : Undergraduate,Graduate,Professional.

```
df['Mortgage'].nunique()
```

```
347
```

Mortgage contain unique value of 347.

```
df['Personal Loan'].unique()
```

```
array(['No', 'Yes'], dtype=object)
```

Personal Loan contain unique values Yes/No.

```
df['Securities Account'].unique()
```

```
array(['Yes', 'No'], dtype=object)
```

Securities account contains unique values of Yes/No.

```
df['CD Account'].unique()
```

```
array(['No', 'Yes'], dtype=object)
```

CD Account contain unique values of Yes/No.

```
df['Online'].unique()
```

```
array(['No', 'Yes'], dtype=object)
```

Online contain unique values of Yes/No.

```
df['CreditCard'].unique()
```

```
array(['No', 'Yes'], dtype=object)
```

Credit Card contain unique values of Yes/No.

```
df['Age'].nunique()
```

```
45
```

Age contains unique values of 45.

```
df.dtypes
```

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

```
table = pd.crosstab(df['Online'], df['Personal Loan'])
print(table)
```

```
Personal Loan    No  Yes
Online
No             1827  189
Yes            2693  291
```

People who use internet banking more are less opted for Personal loan in previous campaign.

Only 291 people who uses online opted for Loan out of 5000.

```
table = pd. crosstab(df['Securities Account'], df['Personal Loan'])
print(table)
```

```
Personal Loan         No  Yes
Securities Account
No                  4058  420
Yes                  462   60
```

People who have more security account deposit are less likely to opted loan according to previous year data.
Only 60 people who have security account opted for loan out of 5000.

And 420 who don’t have security account opted for loan out of 5000.

```
table = pd.crosstab(df['CD Account'], df['Personal Loan'])
print(table)
```

```
Personal Loan    No  Yes
CD Account
No             4358  340
Yes             162  140
```

People who have more CD Account are less likely to opted for loan .

Only 140 People who have CD Account opted for loan out of 5000 and 162 who don't have CD Account opted for loan.

```
table = pd.crosstab(df['Online'], df['Personal Loan'])
print(table)
```

```
Personal Loan    No  Yes
Online
No             1827  189
Yes            2693  291
```

```
table = pd.crosstab(df['CreditCard'], df['Personal Loan'])
print(table)
```

```
Personal Loan    No  Yes
CreditCard
No             3193  337
Yes            1327  143
```

People who uses Credit Card more are less likely to opted for loan.

Only 143 who uses credit card are opted for loan and 1327 who don't use credit card opted for loan out of 5000.

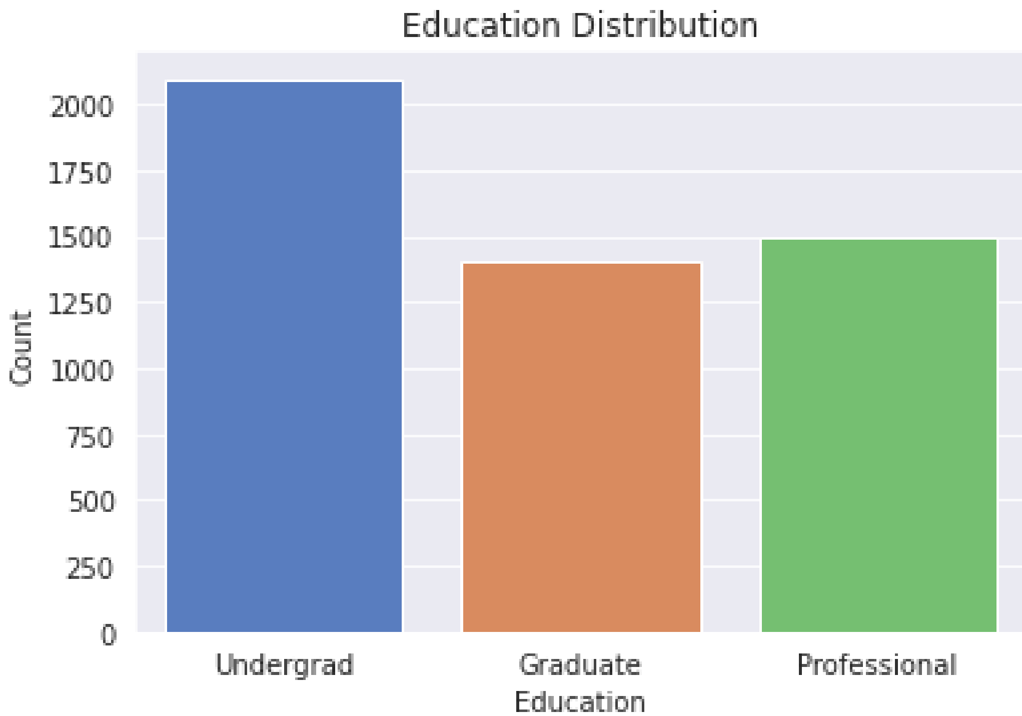
```
table = pd.crosstab(df['Education'], df['Personal Loan'])
print(table)
```

```
Personal Loan    No  Yes
Education
Graduate       1221  182
Professional   1296  205
Undergrad      2003   93
```

Professionals are more likely to opted for loan compare to graduate and undergraduate .After professionals Graduate are more likely to opt for loan .

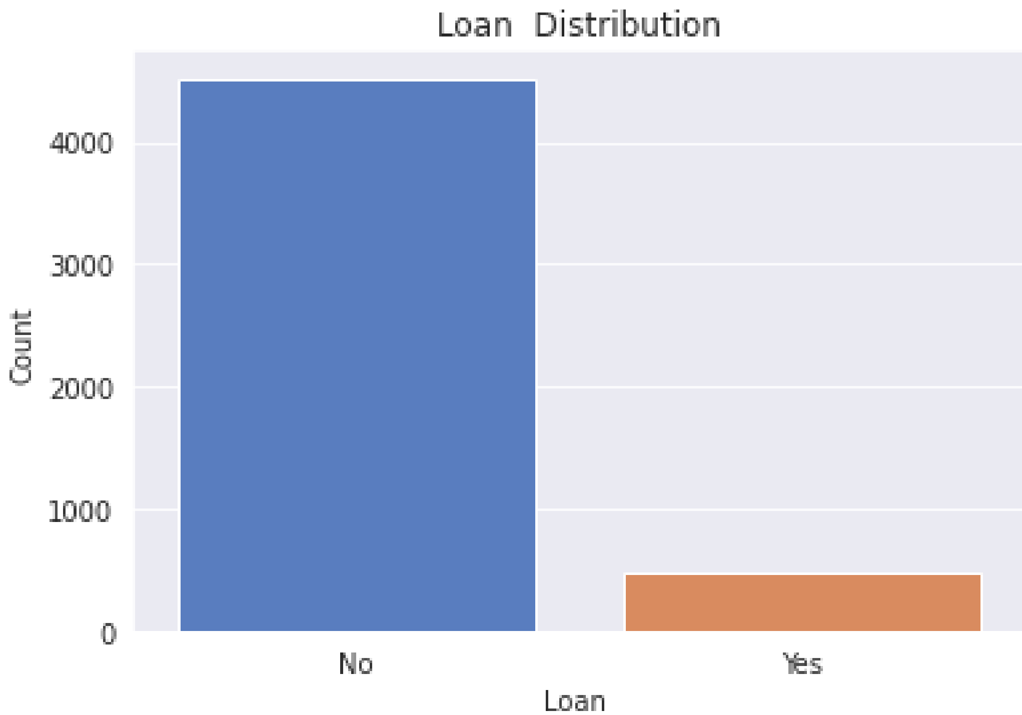
```
sns.set_style("darkgrid") # Set the style of the plot
sns.countplot(x="Education", data=df, palette="muted") # Create the countplot
```

```
plt.title("Education Distribution") # Set the title of the plot
plt.xlabel("Education") # Set the x-axis label
plt.ylabel("Count") # Set the y-axis label
plt.show() # Display the plot
```



Education distributions counts maximum Undergraduate then Professionals then Graduate .

```
sns.set_style("darkgrid") # Set the style of the plot
sns.countplot(x="Personal Loan", data=df, palette="muted") # Create the countplot
plt.title("Loan  Distribution") # Set the title of the plot
plt.xlabel("Loan ") # Set the x-axis label
plt.ylabel("Count") # Set the y-axis label
plt.show() # Display the plot
```

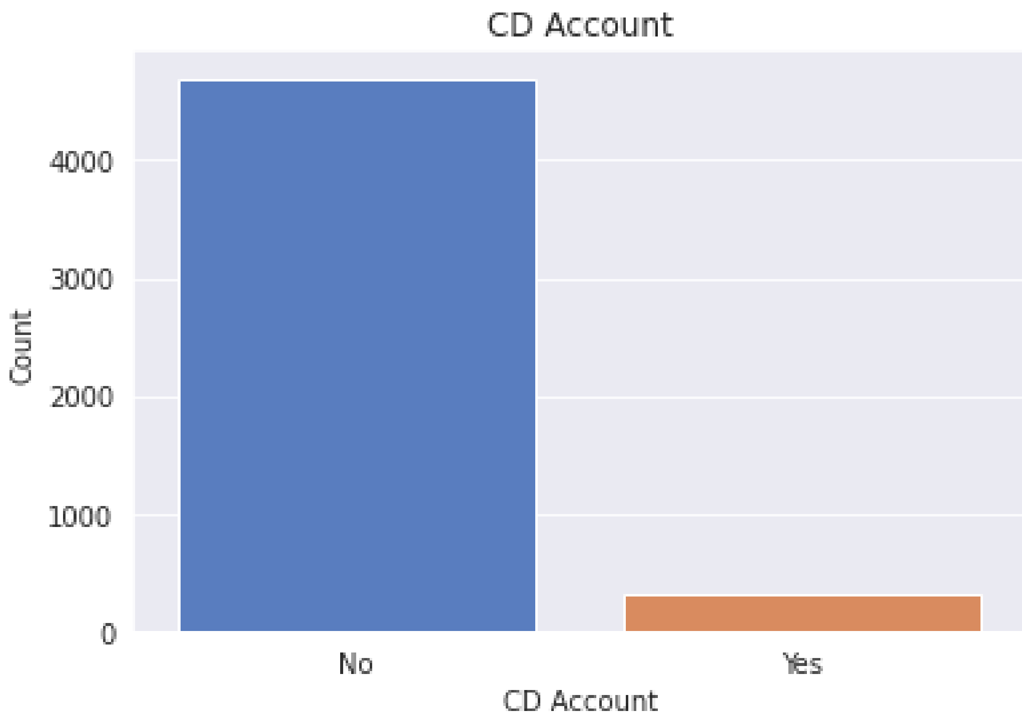


In Loan Distribution graph there are more than 4000 counts who don’t opted for loan only few less than 1000 opted for loan out of 5000.

```
sns.set_style("darkgrid") # Set the style of the plot
sns.countplot(x="Securities Account", data=df, palette="muted") # Create the countplot
plt.title("Securities Account") # Set the title of the plot
plt.xlabel("Securities Accouunt") # Set the x-axis label
plt.ylabel("Count") # Set the y-axis label
plt.show() # Display the plot
```

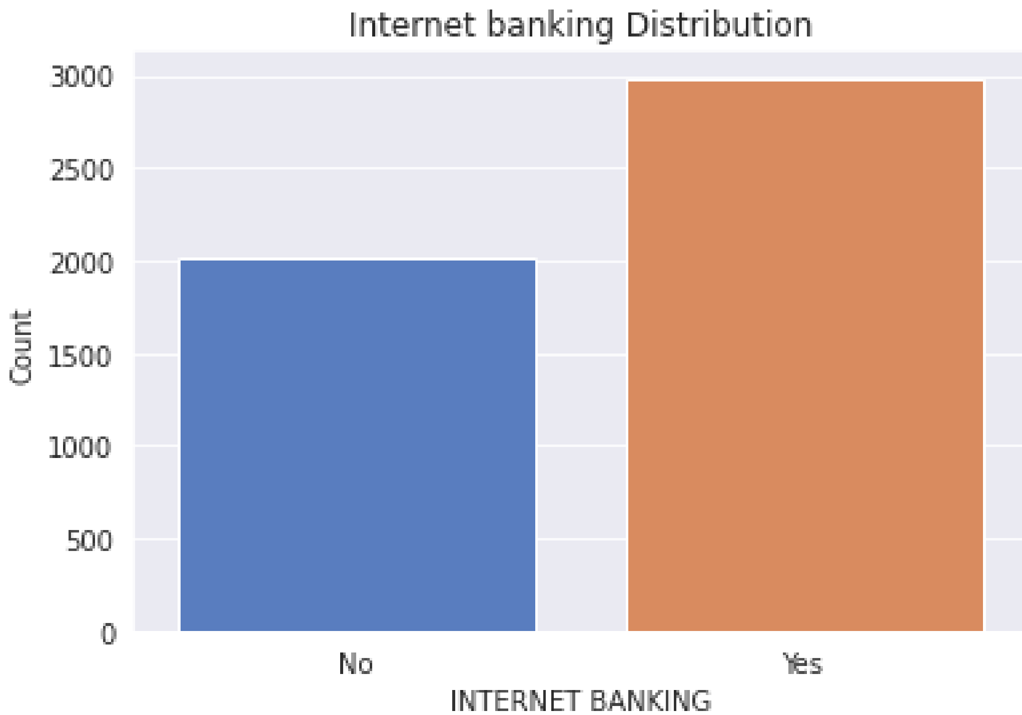
Securities account count plot show much more number of users who don’t have securities account and very few less 1000 who have securities account .

```
sns.set_style("darkgrid") # Set the style of the plot
sns.countplot(x="CD Account", data=df, palette="muted") # Create the
countplot
plt.title("CD Account") # Set the title of the plot
plt.xlabel("CD Account") # Set the x-axis label
plt.ylabel("Count") # Set the y-axis label
plt.show() # Display the plot
```



CD Account countplot shows than more than 4000 users don't have CD Account and very few less than 1000 have CD Account .

```
sns.set_style("darkgrid") # Set the style of the plot
sns.countplot(x="Online", data=df, palette="muted") # Create the countplot
plt.title("Internet banking Distribution") # Set the title of the plot
plt.xlabel("INTERNET BANKING") # Set the x-axis label
plt.ylabel("Count") # Set the y-axis label
plt.show() # Display the plot
```



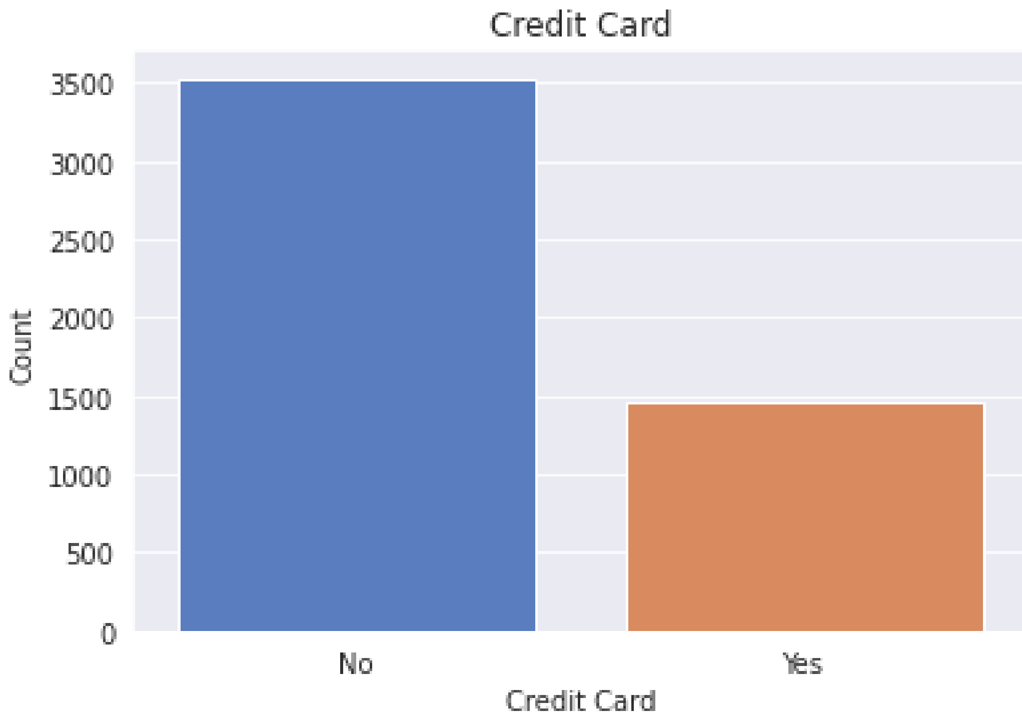
Internet banking count plot show that 3000 users uses internet banking out of 5000 and 2000 don’t use internet banking out of 5000.

```
sns.set_style("darkgrid") # Set the style of the plot
sns.countplot(x="Family", data=df, palette="muted") # Create the countplot
plt.title("Family members") # Set the title of the plot
plt.xlabel("Family Members") # Set the x-axis label
plt.ylabel("Count") # Set the y-axis label
plt.show() # Display the plot
```

There are 4 types of family members .

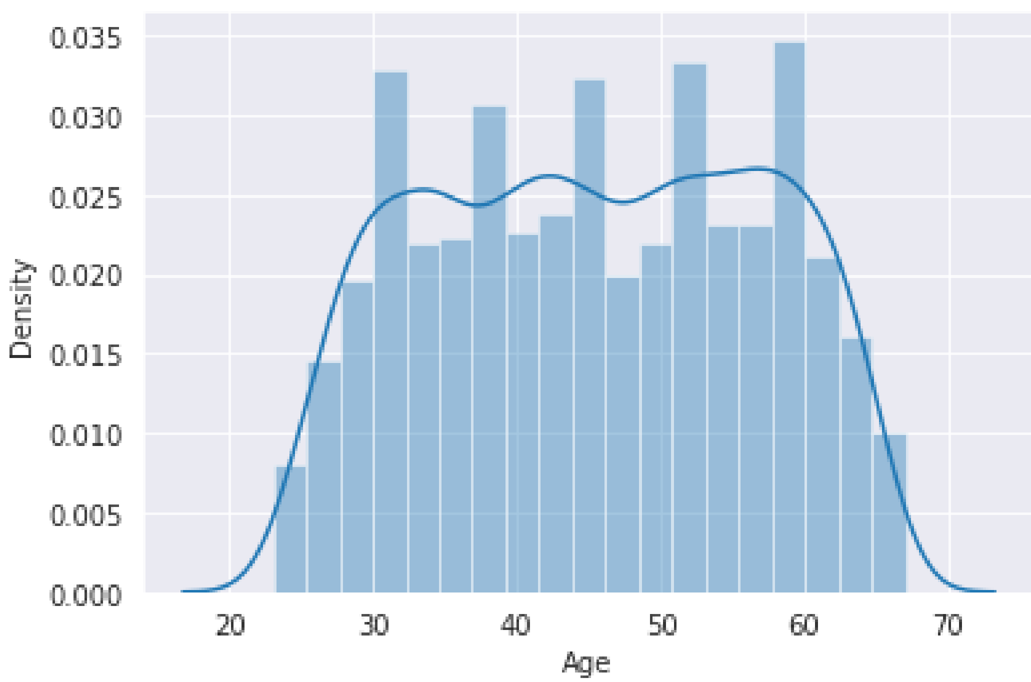
More than 1400 samples are 1 member , then 2 member family , then 4 member family then least 3 member family .

```
sns.set_style("darkgrid") # Set the style of the plot
sns.countplot(x="CreditCard", data=df, palette="muted") # Create the countplot
plt.title("Credit Card") # Set the title of the plot
plt.xlabel("Credit Card") # Set the x-axis label
plt.ylabel("Count") # Set the y-axis label
plt.show() # Display the plot
```



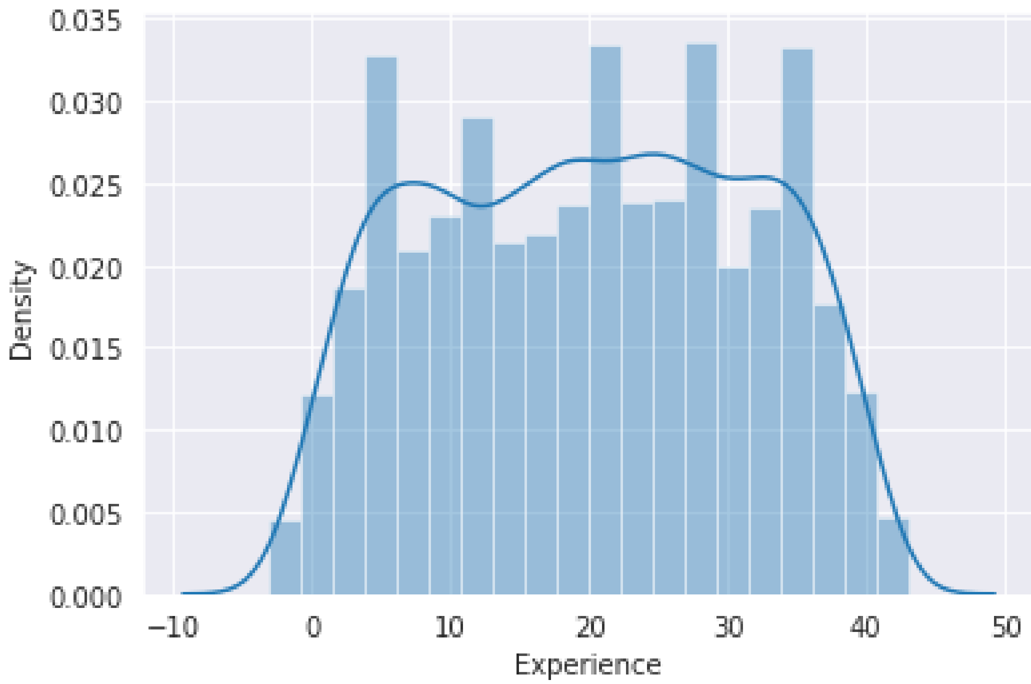
In credit card count plot around 3500 users don’t use credit card and around 1500 users uses credit card out of 5000.

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



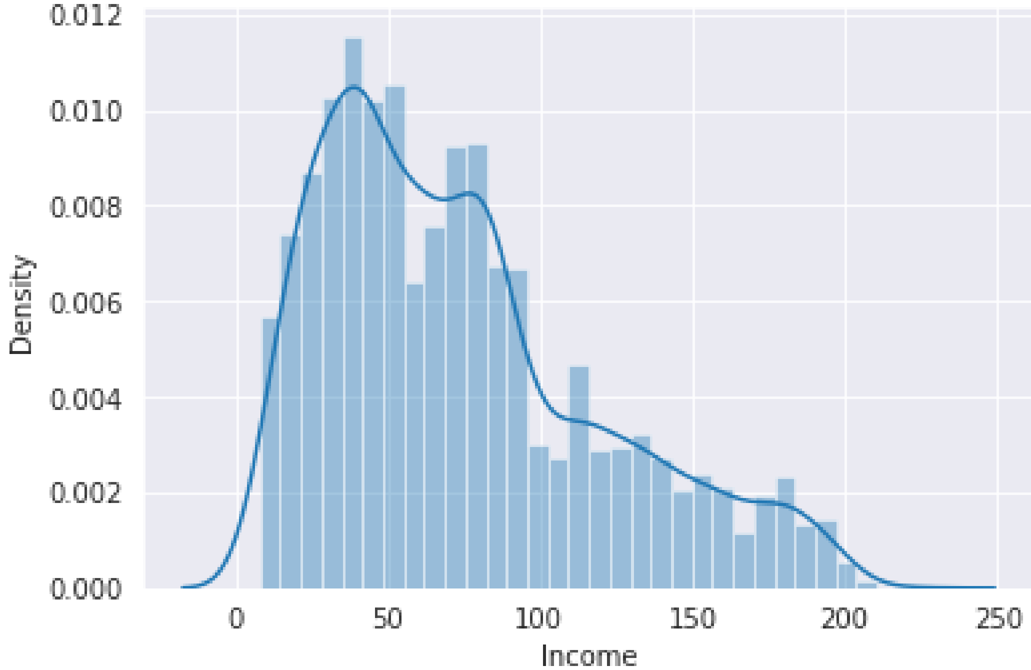
In Age distribution plot data is normally distributed .

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



In Experience distribution plot data is normally distributed .

```
sns.distplot(df["Income"])
```

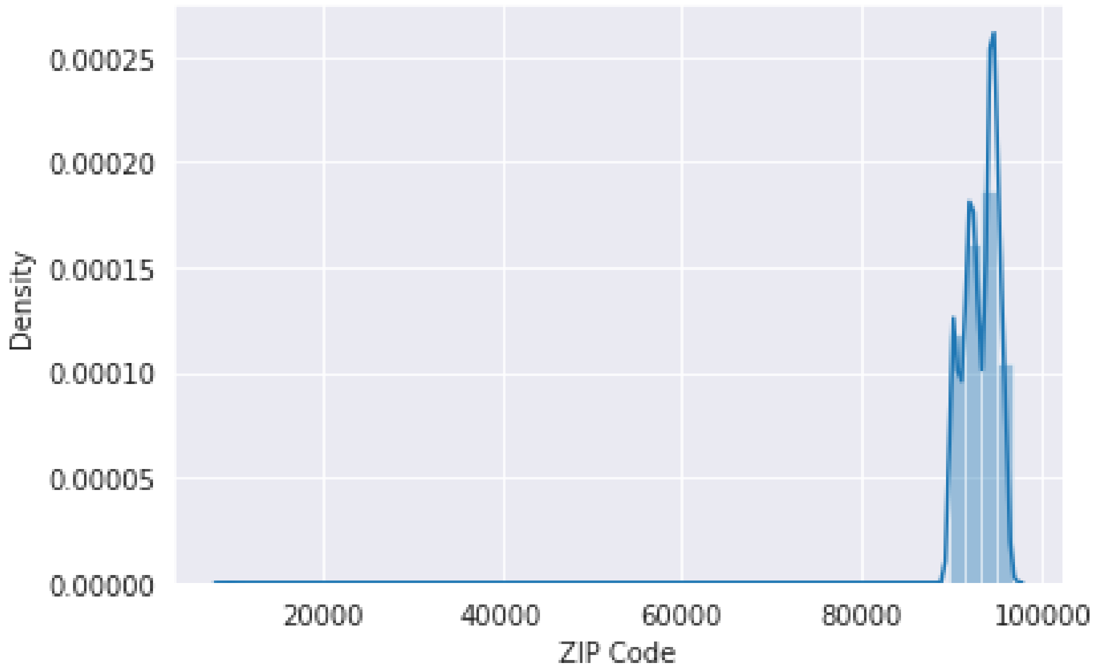


In Income distribution plot data is positive skewed .Mean is greater than mean is greater than mode.

```
df.columns
```

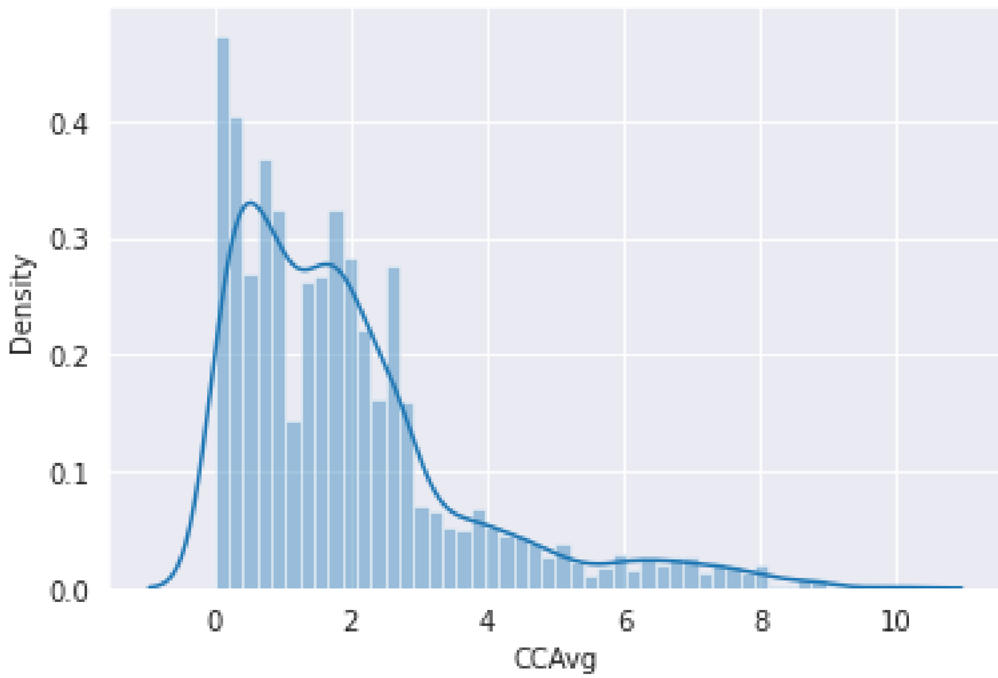
```
Index(['ID', 'Age', 'Experience', 'Income', 'ZIP Code', 'Family', 'CCAvg',
       'Education', 'Mortgage', 'Personal Loan', 'Securities Account',
       'CD Account', 'Online', 'CreditCard'],
      dtype='object')
```

```
sns.distplot(df["ZIP Code"])
```



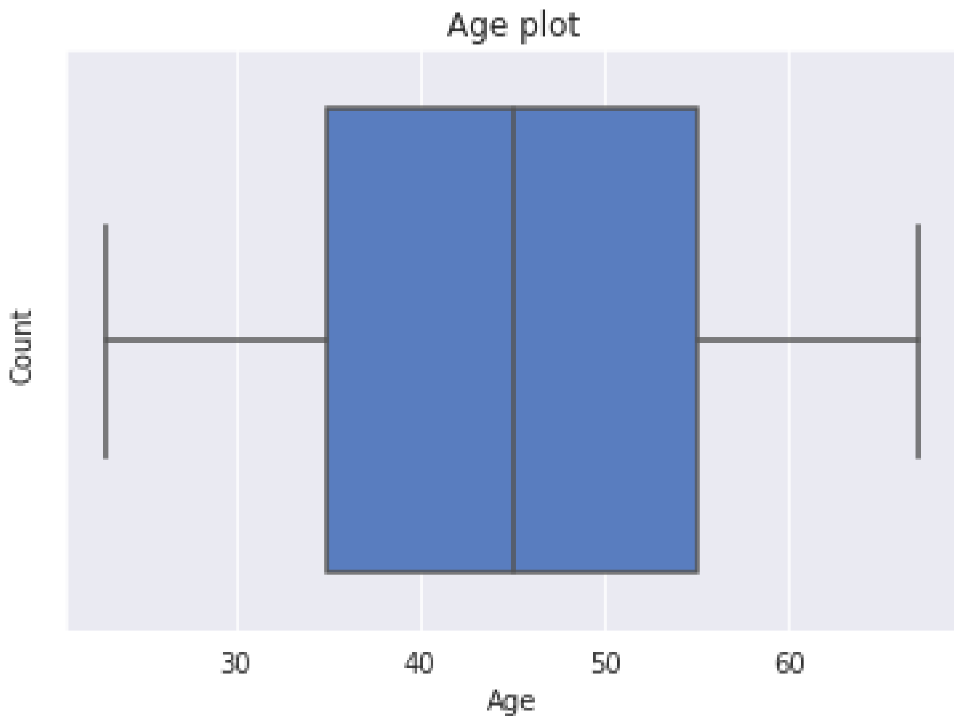
ZIP Code distribution plot is not normally distributed and left skewed.

```
sns.distplot(df["CCAvg"])
```



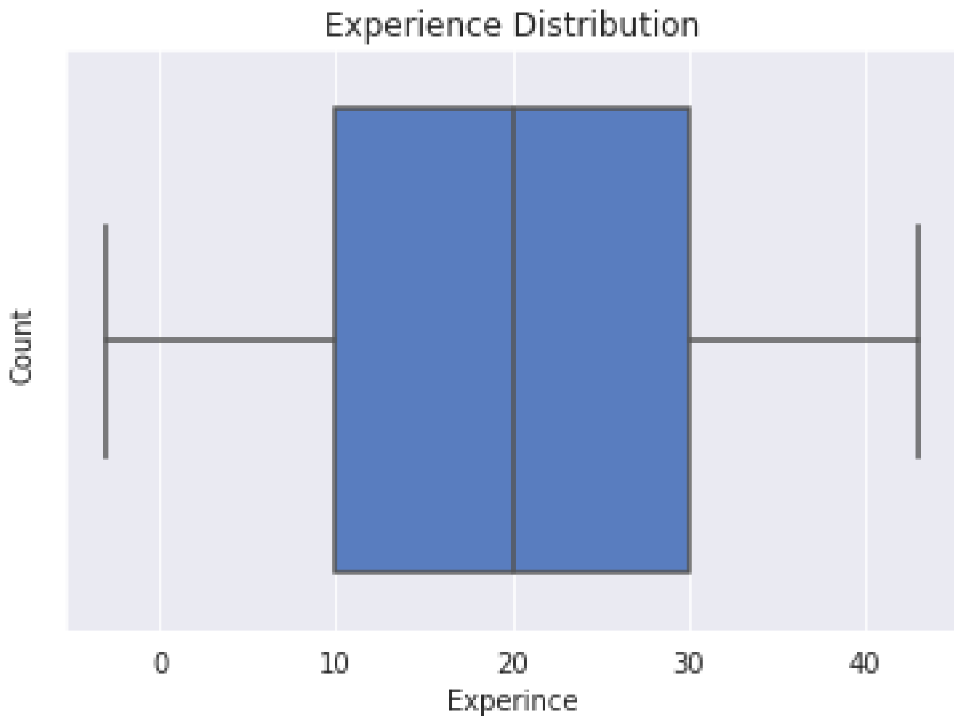
Distribution plot of CCAvg is not normally distributed and right skewed.

```
sns.set_style("darkgrid") # Set the style of the plot
sns.boxplot(x="Age", data=df, palette="muted") # Create the countplot
plt.title("Age plot") # Set the title of the plot
plt.xlabel("Age") # Set the x-axis label
plt.ylabel("Count") # Set the y-axis label
plt.show() # Display the plot
```

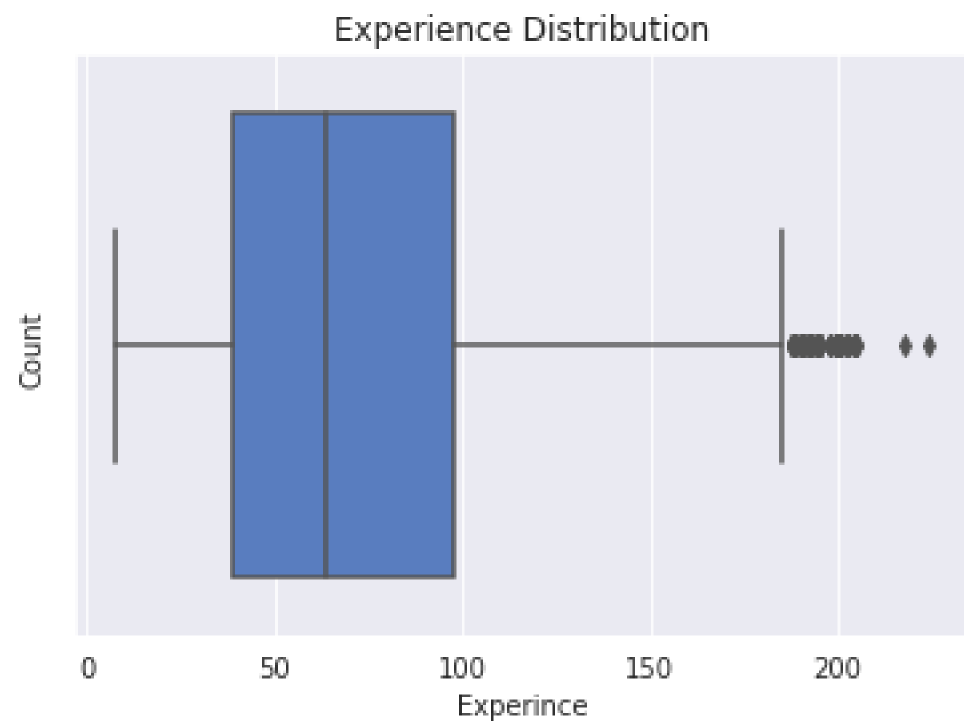


Age boxplot contains no outliers.

```
sns.set_style("darkgrid") # Set the style of the plot
sns.boxplot(x="Experience", data=df, palette="muted") # Create the countplot
plt.title("Experience Distribution") # Set the title of the plot
plt.xlabel("Experince") # Set the x-axis label
plt.ylabel("Count") # Set the y-axis label
plt.show() # Display the plot
```




```
sns.set_style("darkgrid") # Set the style of the plot
sns.boxplot(x="Income", data=df, palette="muted") # Create the countplot
plt.title("Experience Distribution") # Set the title of the plot
plt.xlabel("Experince") # Set the x-axis label
plt.ylabel("Count") # Set the y-axis label
plt.show() # Display the plot
```



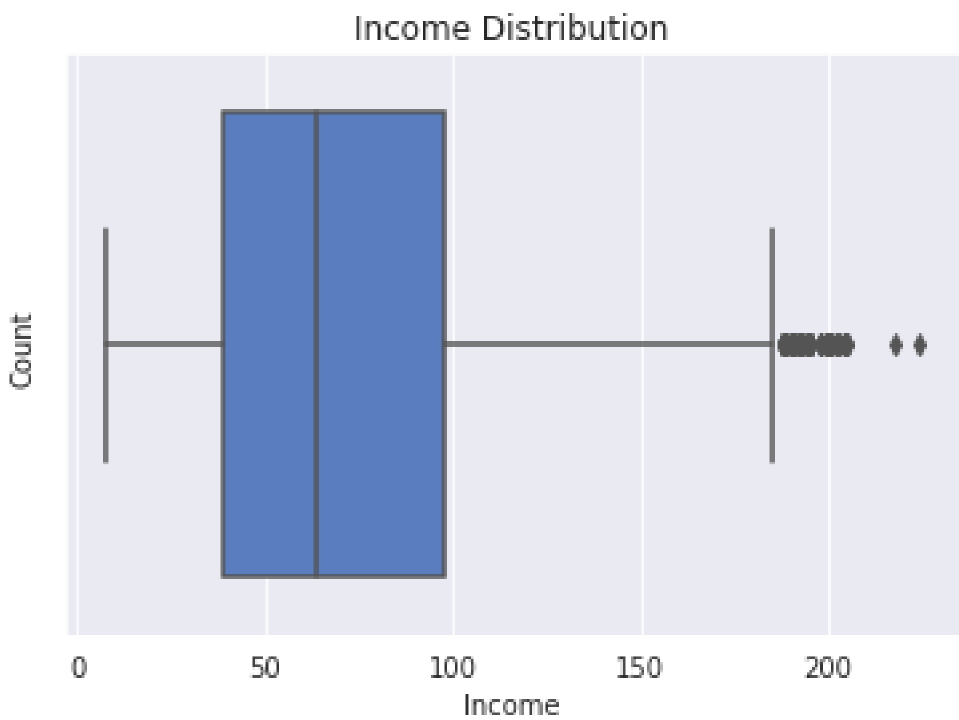
Experience boxplot is not normally distributed and contains outliers so did data treatment of outliers.

```
q = df['Experience'].quantile(0.95)

# filter the DataFrame to exclude values above the 95th percentile
df = df[df['Experience'] < q]

# print the filtered DataFrame
print(df)
```

```
        ID  Age  Experience  Income  ZIP Code  Family     CCAvg     Education  \
0        1   25           1      49     91107       4  1.600000     Undergrad
1        2   45          19      34     90089       3  1.500000     Undergrad
2        3   39          15      11     94720       1  1.000000     Undergrad
3        4   35           9     100     94112       1  2.700000      Graduate
4        5   35           8      45     91330       4  1.000000      Graduate
...    ...  ...         ...     ...       ...     ...       ...           ...
4992  4993   30           5      13     90037       4  0.500000  Professional
4993  4994   45          21     218     91801       2  6.666667     Undergrad
4995  4996   29           3      40     92697       1  1.900000  Professional
4996  4997   30           4      15     92037       4  0.400000     Undergrad
4999  5000   28           4      83     92612       3  0.800000     Undergrad

      Mortgage Personal Loan Securities Account CD Account Online CreditCard
0            0            No                Yes         No     No        No
```

```
1              0            No                 Yes          No      No          No
2              0            No                  No          No      No          No
3              0            No                  No          No      No          No
4              0            No                  No          No      No         Yes
...          ...           ...                 ...         ...     ...         ...
4992           0            No                  No          No      No          No
4993           0            No                  No          No     Yes          No
4995           0            No                  No          No     Yes          No
4996          85            No                  No          No     Yes          No
4999           0            No                  No          No     Yes         Yes

[4716 rows x 14 columns]
```

```python
sns.set_style("darkgrid") # Set the style of the plot
sns.boxplot(x="Income", data=df, palette="muted") # Create the countplot
plt.title("Income Distribution") # Set the title of the plot
plt.xlabel("Income") # Set the x-axis label
plt.ylabel("Count") # Set the y-axis label
plt.show() # Display the plot
```



Income boxplot is not normally distributed and contains outliers so we will do data treatment of outliers.

```python
q = df['Income'].quantile(0.95)

# filter the DataFrame to exclude values above the 95th percentile
df = df[df['Income'] < q]
```

```python
# print the filtered DataFrame
print(df)
```

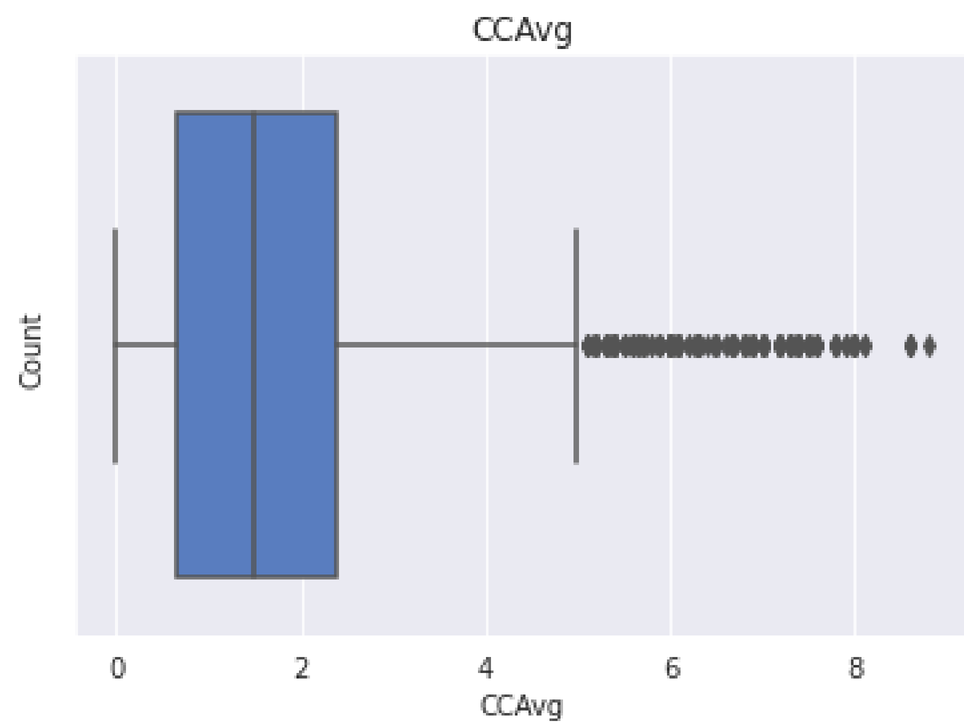
```
        ID  Age  Experience  Income  ZIP Code  Family  CCAvg     Education  \
0        1   25           1      49     91107       4    1.6     Undergrad   
1        2   45          19      34     90089       3    1.5     Undergrad   
2        3   39          15      11     94720       1    1.0     Undergrad   
3        4   35           9     100     94112       1    2.7      Graduate   
4        5   35           8      45     91330       4    1.0      Graduate   
...    ...  ...         ...     ...       ...     ...    ...           ...   
4991  4992   51          25      92     91330       1    1.9      Graduate   
4992  4993   30           5      13     90037       4    0.5  Professional   
4995  4996   29           3      40     92697       1    1.9  Professional   
4996  4997   30           4      15     92037       4    0.4     Undergrad   
4999  5000   28           4      83     92612       3    0.8     Undergrad   

      Mortgage Personal Loan Securities Account CD Account Online CreditCard  
0            0            No                Yes         No     No         No  
1            0            No                Yes         No     No         No  
2            0            No                 No         No     No         No  
3            0            No                 No         No     No         No  
4            0            No                 No         No     No        Yes  
...        ...           ...                ...        ...    ...        ...  
4991       100            No                 No         No     No        Yes  
4992         0            No                 No         No     No         No  
4995         0            No                 No         No    Yes         No  
4996        85            No                 No         No    Yes         No  
4999         0            No                 No         No    Yes        Yes  

[4476 rows x 14 columns]
```

```python
sns.set_style("darkgrid") # Set the style of the plot
sns.boxplot(x="ZIP Code", data=df, palette="muted") # Create the countplot
plt.title("Zip code") # Set the title of the plot
plt.xlabel("Zip code") # Set the x-axis label
plt.ylabel("Count") # Set the y-axis label
plt.show() # Display the plot
```


```
df.columns
```

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

```
sns.set_style("darkgrid") # Set the style of the plot
sns.boxplot(x="CCAvg", data=df, palette="muted") # Create the countplot
plt.title("CCAvg") # Set the title of the plot
plt.xlabel("CCAvg") # Set the x-axis label
plt.ylabel("Count") # Set the y-axis label
plt.show() # Display the plot
```



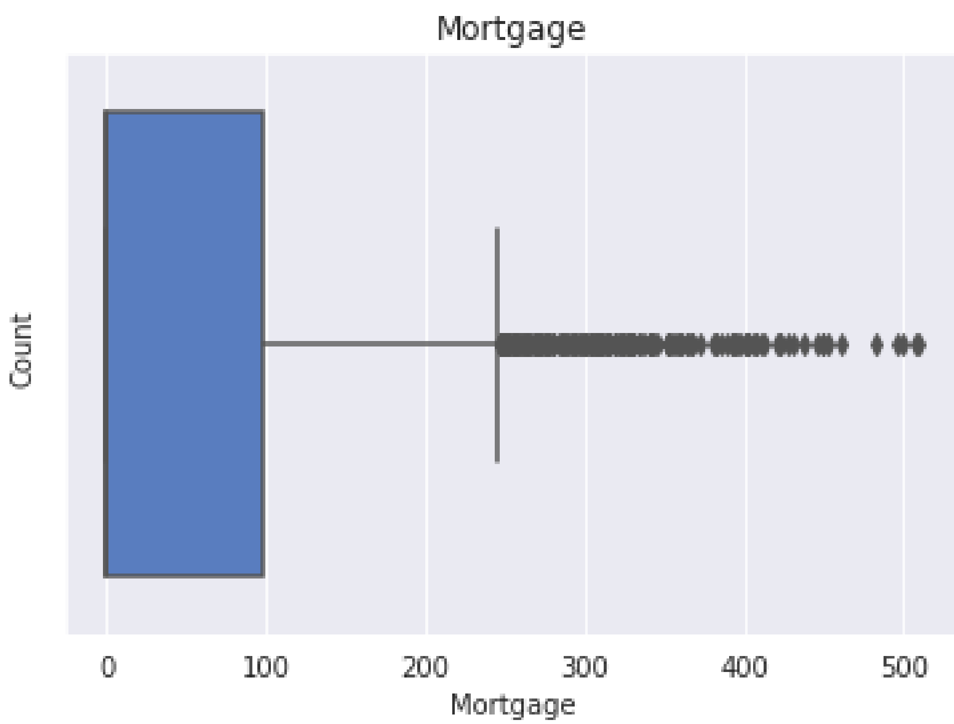
CCAvg boxplot contains outliers so we did data treatment to outliers.

```
q = df['CCAvg'].quantile(0.95)

# filter the DataFrame to exclude values above the 95th percentile
df = df[df['CCAvg'] < q]

# print the filtered DataFrame
print(df)
```

```
        ID  Age  Experience  Income  ZIP Code  Family  CCAvg     Education  \
0        1   25           1      49     91107       4    1.6     Undergrad
1        2   45          19      34     90089       3    1.5     Undergrad
2        3   39          15      11     94720       1    1.0     Undergrad
3        4   35           9     100     94112       1    2.7      Graduate
4        5   35           8      45     91330       4    1.0      Graduate
...    ...  ...         ...     ...       ...     ...    ...           ...
4991  4992   51          25      92     91330       1    1.9      Graduate
4992  4993   30           5      13     90037       4    0.5  Professional
4995  4996   29           3      40     92697       1    1.9  Professional
4996  4997   30           4      15     92037       4    0.4     Undergrad
4999  5000   28           4      83     92612       3    0.8     Undergrad

      Mortgage Personal Loan Securities Account CD Account Online CreditCard
0            0            No                Yes         No     No         No
```

```
1               0               No                    Yes          No      No          No
2               0               No                     No          No      No          No
3               0               No                     No          No      No          No
4               0               No                     No          No      No         Yes
...           ...              ...                    ...         ...     ...         ...
4991          100               No                     No          No      No         Yes
4992            0               No                     No          No      No          No
4995            0               No                     No          No     Yes          No
4996           85               No                     No          No     Yes          No
4999            0               No                     No          No     Yes         Yes

[4244 rows x 14 columns]
```

```python
sns.set_style("darkgrid") # Set the style of the plot
sns.boxplot(x="Mortgage", data=df, palette="muted") # Create the countplot
plt.title("Mortgage") # Set the title of the plot
plt.xlabel("Mortgage") # Set the x-axis label
plt.ylabel("Count") # Set the y-axis label
plt.show() # Display the plot
```



Mortgage boxplot contains outliers so we will do data treatment to outliers.

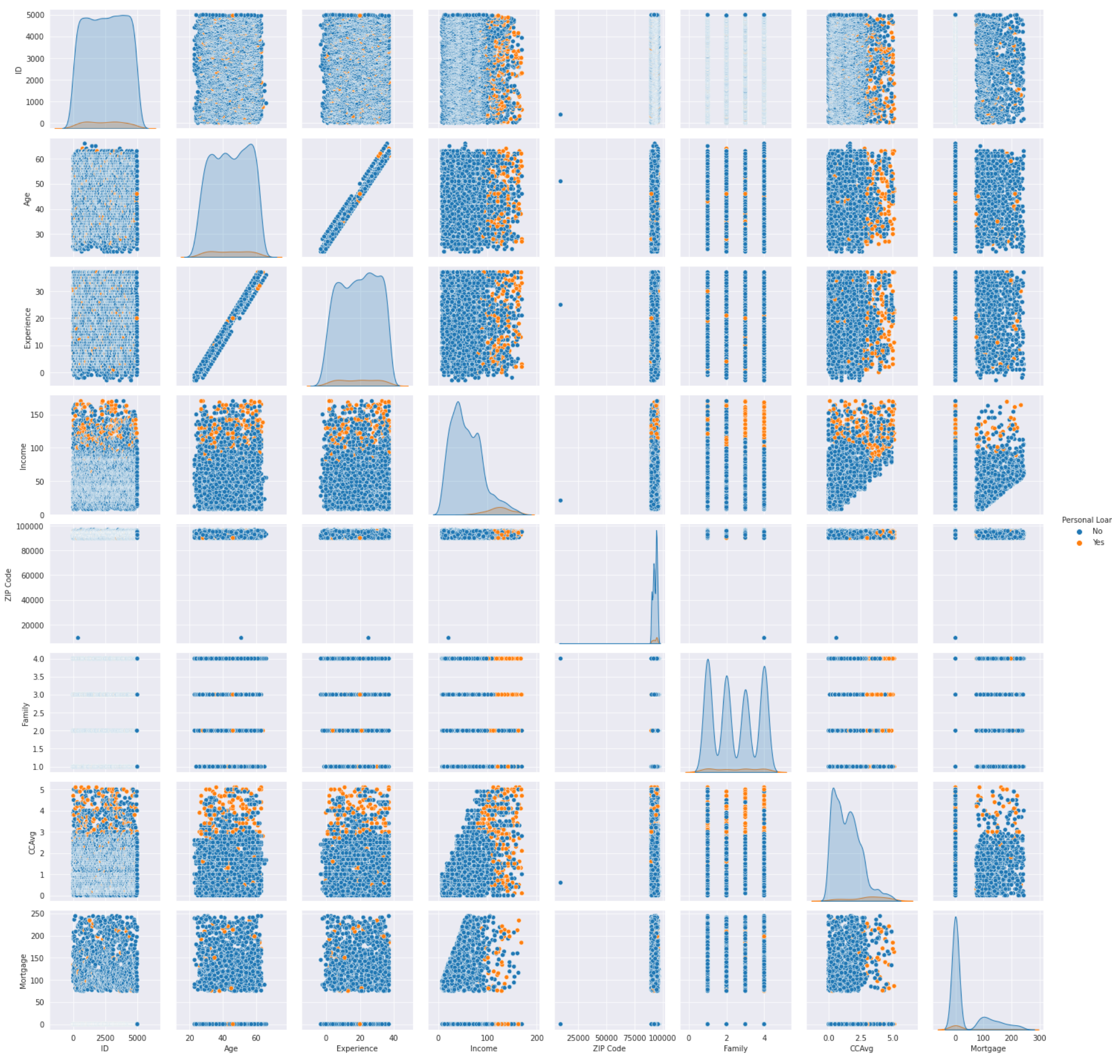
```python
q = df['Mortgage'].quantile(0.95)

# filter the DataFrame to exclude values above the 95th percentile
df = df[df['Mortgage'] < q]
```

```
# print the filtered DataFrame
print(df)
```

```
        ID  Age  Experience  Income  ZIP Code  Family  CCAvg    Education  \
0        1   25           1      49     91107       4    1.6    Undergrad   
1        2   45          19      34     90089       3    1.5    Undergrad   
2        3   39          15      11     94720       1    1.0    Undergrad   
3        4   35           9     100     94112       1    2.7     Graduate   
4        5   35           8      45     91330       4    1.0     Graduate   
...    ...  ...         ...     ...       ...     ...    ...          ...   
4991  4992   51          25      92     91330       1    1.9     Graduate   
4992  4993   30           5      13     90037       4    0.5  Professional   
4995  4996   29           3      40     92697       1    1.9  Professional   
4996  4997   30           4      15     92037       4    0.4    Undergrad   
4999  5000   28           4      83     92612       3    0.8    Undergrad   

      Mortgage Personal Loan Securities Account CD Account Online CreditCard  
0            0            No                Yes         No     No         No  
1            0            No                Yes         No     No         No  
2            0            No                 No         No     No         No  
3            0            No                 No         No     No         No  
4            0            No                 No         No     No        Yes  
...        ...           ...                ...        ...    ...        ...  
4991       100            No                 No         No     No        Yes  
4992         0            No                 No         No     No         No  
4995         0            No                 No         No    Yes         No  
4996        85            No                 No         No    Yes         No  
4999         0            No                 No         No    Yes        Yes  

[4031 rows x 14 columns]
```

```
sns.pairplot(data=df , hue="Personal Loan")
```

```
<seaborn.axisgrid.PairGrid at 0x7f66153f5ee0>
```

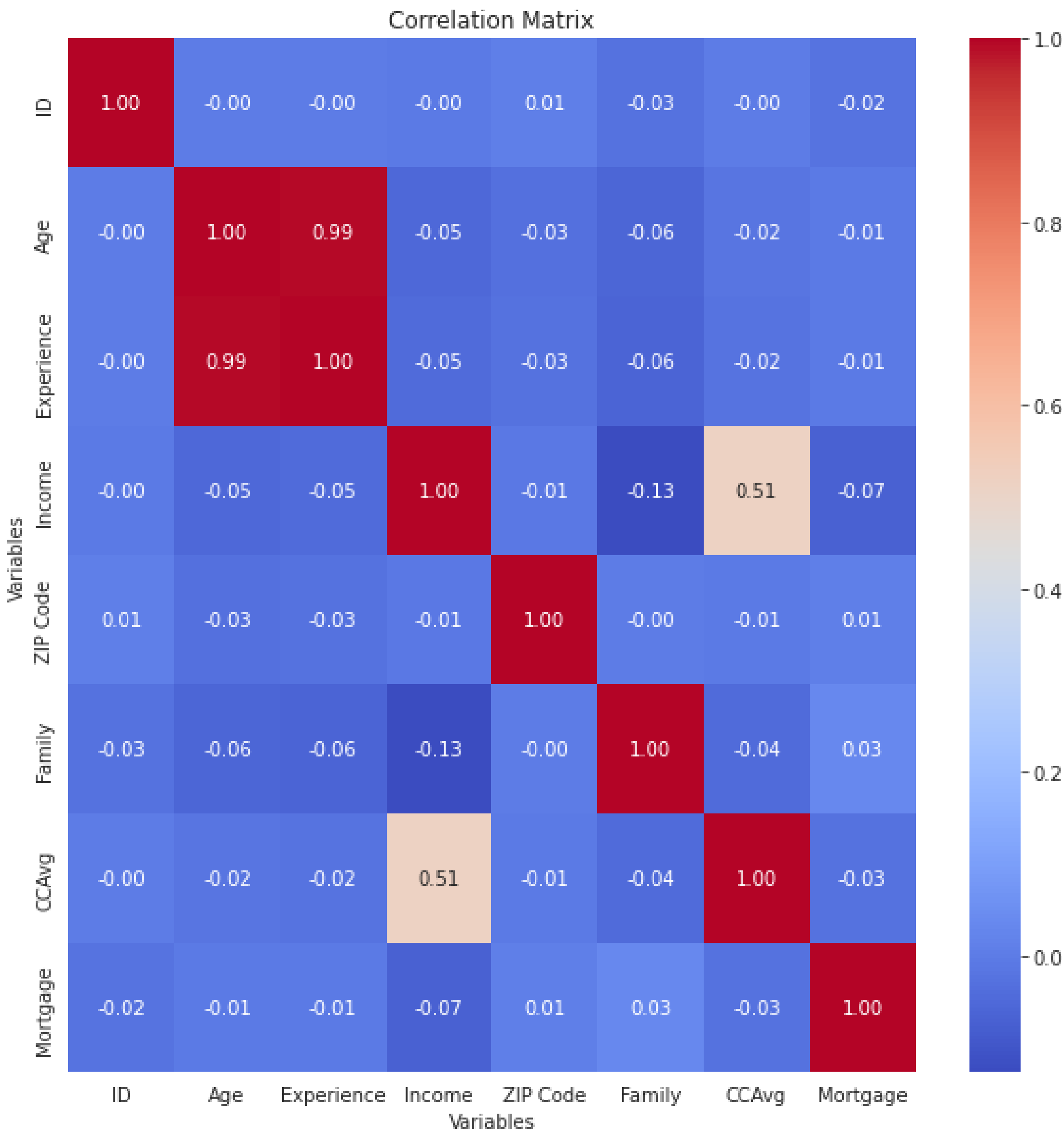


1)Experience and Age are highly correlated.

```
corr=df.corr()
```

```
plt.figure(figsize=(10, 10))
sns.heatmap(corr, cmap='coolwarm', annot=True, fmt='.2f')
```

```
plt.title('Correlation Matrix')
plt.xlabel('Variables')
plt.ylabel('Variables')

plt.show()
```



Heartmap show correlation matrix. Experience is highly correlated with Age so we will drop one of them while model building . Rest parameter are normally correlated.

```
skewness=df[['ID','Age','Experience','Income','ZIP Code','CCAvg','Mortgage']]
```

```
print(skewness.skew())
```

```
ID             -0.014335
Age            -0.060093
Experience     -0.075585
Income          0.772612
ZIP Code      -13.832771
CCAvg           0.844064
Mortgage        1.498761
dtype: float64
```

ID,Age,Experience,Zip code are negatively skewed.

Income , CCavg, Mortgage are positively skewed.

```
kutrosis=df[['ID','Age','Experience','Income','ZIP Code','CCAvg','Mortgage']]

print(kutrosis.kurt())
```

```
ID              -1.208904
Age             -1.185368
Experience      -1.171609
Income           0.084178
ZIP Code       519.860788
CCAvg            0.298434
Mortgage         0.876522
dtype: float64
```

```
summary = df.describe()

iqr = summary.loc['75%'] - summary.loc['25%']


print(iqr)
```

```
ID            2523.5
Age             19.0
Experience      18.0
Income          48.0
ZIP Code      2667.0
Family           3.0
CCAvg            1.6
Mortgage        82.0
dtype: float64
```

```
Age_Mean=df["Age"].mean()
print(Age_Mean)
```

```
44.39345075663607
```

One sample t-test.

```
from scipy.stats import ttest_1samp

t_statistic, p_value = ttest_1samp(df['Age'], 43)

# print the test results
print('One-sample t-test statistic:', t_statistic)
print('p-value:', p_value)
```

```
One-sample t-test statistic: 8.15159825526504
p-value: 4.747481740445591e-16
```

Here p-value < 0.05 so we will reject null hypothesis and select alternate hypothesis that is The population mean is not equal to the hypothesized value.

```
df.describe()
```

```
                ID          Age   Experience       Income      ZIP Code  \
count  4031.000000  4031.000000  4031.000000  4031.000000   4031.000000
mean   2513.076656    44.393451    19.106921    61.531878  93156.963036
std    1449.964554    10.853136    10.819554    34.913618   2202.431292
min       1.000000    23.000000    -3.000000     8.000000   9307.000000
25%    1245.000000    35.000000    10.000000    34.000000  91942.000000
50%    2528.000000    45.000000    19.000000    55.000000  93524.000000
75%    3768.500000    54.000000    28.000000    82.000000  94609.000000
max    5000.000000    66.000000    37.000000   170.000000  96651.000000

            Family        CCAvg     Mortgage
count  4031.000000  4031.000000  4031.000000
mean      2.457703     1.524237    37.295956
std       1.160617     1.128664    66.112683
min       1.000000     0.000000     0.000000
25%       1.000000     0.600000     0.000000
50%       2.000000     1.400000     0.000000
75%       4.000000     2.200000    82.000000
max       4.000000     5.100000   244.000000
```

```
from scipy.stats import ttest_1samp

t_statistic, p_value = ttest_1samp(df['Experience'], 17.66)


print('One-sample t-test statistic:', t_statistic)
print('p-value:', p_value)
```

```
One-sample t-test statistic: 8.490669766463846
p-value: 2.850378134215185e-17
```

Here p-value < 0.05 so we will reject null hypothesis and select alternate hypothesis that is The population mean is not equal to the hypothesized value.

```
df
```

```
        ID  Age  Experience  Income  ZIP Code  Family  CCAvg    Education  \
0        1   25           1      49     91107       4    1.6    Undergrad
1        2   45          19      34     90089       3    1.5    Undergrad
2        3   39          15      11     94720       1    1.0    Undergrad
3        4   35           9     100     94112       1    2.7     Graduate
4        5   35           8      45     91330       4    1.0     Graduate
...    ...  ...         ...     ...       ...     ...    ...          ...
4991  4992   51          25      92     91330       1    1.9     Graduate
4992  4993   30           5      13     90037       4    0.5  Professional
4995  4996   29           3      40     92697       1    1.9  Professional
4996  4997   30           4      15     92037       4    0.4    Undergrad
4999  5000   28           4      83     92612       3    0.8    Undergrad

      Mortgage Personal Loan Securities Account CD Account Online CreditCard
0            0            No                Yes         No     No         No
1            0            No                Yes         No     No         No
2            0            No                 No         No     No         No
3            0            No                 No         No     No         No
4            0            No                 No         No     No        Yes
...        ...           ...                ...        ...    ...        ...
4991       100            No                 No         No     No        Yes
4992         0            No                 No         No     No         No
4995         0            No                 No         No    Yes         No
4996        85            No                 No         No    Yes         No
4999         0            No                 No         No    Yes        Yes

[4031 rows x 14 columns]
```

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()

df_encoded = df.apply(le.fit_transform)
```

```
# print the encoded DataFrame
print(df_encoded)

        ID  Age  Experience  Income  ZIP Code  Family  CCAvg  Education  \
0        0    2           4      33        82       3     19          2
1        1   22          22      22        34       2     18          2
2        2   16          18       3       365       0     12          2
3        3   12          12      74       296       0     34          0
4        4   12          11      31        94       3     12          0
...    ...  ...         ...     ...       ...     ...    ...        ...
4991  4026   28          28      68        94       0     24          0
4992  4027    7           8       5        18       3      5          1
4995  4028    6           6      26       207       0     24          1
4996  4029    7           7       7       139       3      4          2
4999  4030    5           7      61       188       2     10          2

      Mortgage  Personal Loan  Securities Account  CD Account  Online  \
0            0              0                   1           0       0
1            0              0                   1           0       0
2            0              0                   0           0       0
3            0              0                   0           0       0
4            0              0                   0           0       0
...        ...            ...                 ...         ...     ...
4991        26              0                   0           0       0
4992         0              0                   0           0       0
4995         0              0                   0           0       1
4996        11              0                   0           0       1
4999         0              0                   0           0       1

      CreditCard
0              0
1              0
2              0
3              0
4              1
...          ...
4991           1
4992           0
4995           0
4996           0
4999           1

[4031 rows x 14 columns]

cat_columns = ['Family','Personal Loan','Education','Securities Account','CD 
Account','Online','CreditCard']

# Apply label encoding to categorical columns
for col in cat_columns:
```

```
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
```

Converted categorical columns into continuous using label encoder for model building.

```
# Print the encoded dataframe
print(df)
```

```
        ID  Age  Experience  Income  ZIP Code  Family  CCAvg  Education  \
0        1   25           1      49     91107       3    1.6          2
1        2   45          19      34     90089       2    1.5          2
2        3   39          15      11     94720       0    1.0          2
3        4   35           9     100     94112       0    2.7          0
4        5   35           8      45     91330       3    1.0          0
...    ...  ...         ...     ...       ...     ...    ...        ...
4991  4992   51          25      92     91330       0    1.9          0
4992  4993   30           5      13     90037       3    0.5          1
4995  4996   29           3      40     92697       0    1.9          1
4996  4997   30           4      15     92037       3    0.4          2
4999  5000   28           4      83     92612       2    0.8          2

      Mortgage  Personal Loan  Securities Account  CD Account  Online  \
0            0              0                   1           0       0
1            0              0                   1           0       0
2            0              0                   0           0       0
3            0              0                   0           0       0
4            0              0                   0           0       0
...        ...            ...                 ...         ...     ...
4991       100              0                   0           0       0
4992         0              0                   0           0       0
4995         0              0                   0           0       1
4996        85              0                   0           0       1
4999         0              0                   0           0       1

      CreditCard
0              0
1              0
2              0
3              0
4              1
...          ...
4991           1
4992           0
4995           0
4996           0
4999           1
```

```
[4031 rows x 14 columns]
```

```
# Import libraries
from sklearn.model_selection import train_test_split

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

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Print the shape of the training and testing sets
print("Training set shape:", X_train.shape, y_train.shape)
print("Testing set shape:", X_test.shape, y_test.shape)
```

```
Training set shape: (3224, 13) (3224,)
Testing set shape: (807, 13) (807,)
```

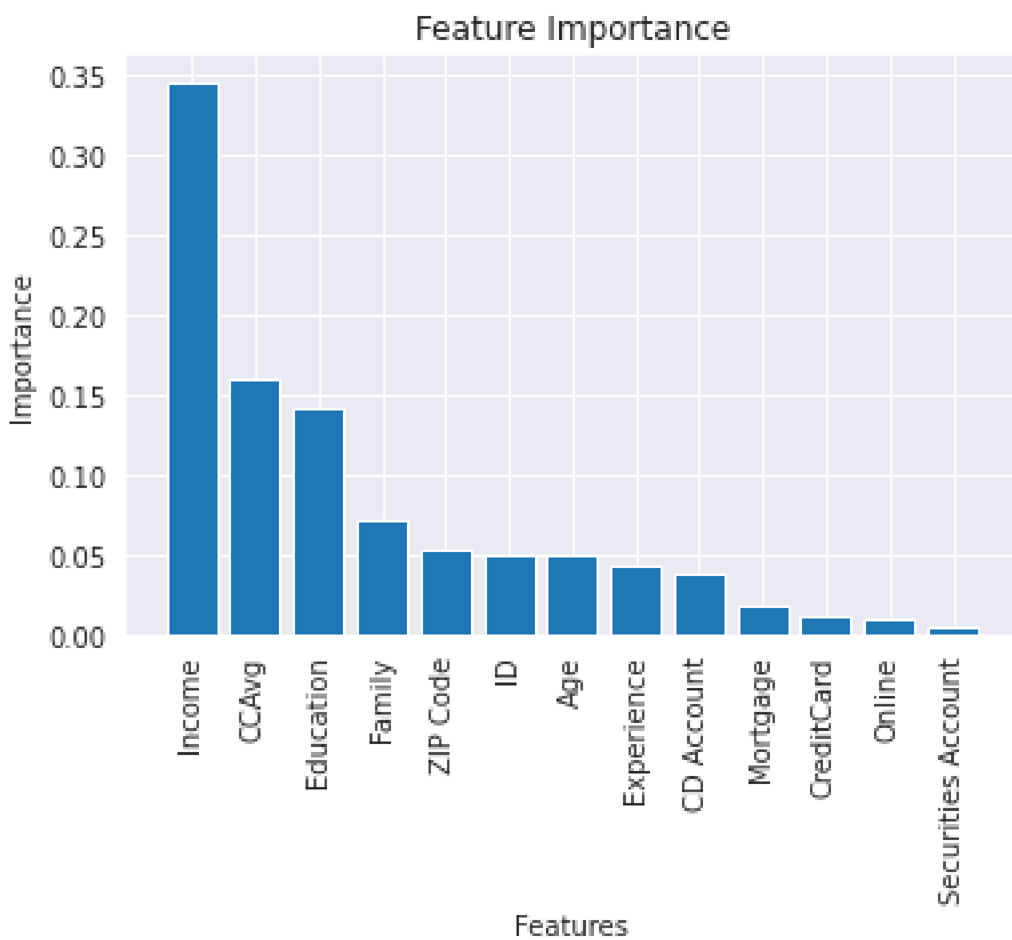
```
df['Education']
```

```
0       2
1       2
2       2
3       0
4       0
       ..
4991    0
4992    1
4995    1
4996    2
4999    2
Name: Education, Length: 4031, dtype: int64
```

```
from sklearn.ensemble import RandomForestClassifier
import matplotlib.pyplot as plt



rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X, y)


importances = rf_model.feature_importances_


sorted_indices = importances.argsort()[::-1]
sorted_importances = importances[sorted_indices]
```

```
print("Feature ranking:")
for i in range(X.shape[1]):
    print("%d. %s (%f)" % (i + 1, X.columns[sorted_indices[i]],
sorted_importances[i]))


plt.bar(range(X.shape[1]), sorted_importances)
plt.xticks(range(X.shape[1]), X.columns[sorted_indices], rotation=90)
plt.xlabel('Features')
plt.ylabel('Importance')
plt.title('Feature Importance')
plt.show()
```

```
Feature ranking:
1. Income (0.345087)
2. CCAvg (0.160593)
3. Education (0.140986)
4. Family (0.071978)
5. ZIP Code (0.053130)
6. ID (0.050608)
7. Age (0.049673)
8. Experience (0.043969)
9. CD Account (0.038619)
10. Mortgage (0.018386)
11. CreditCard (0.011868)
12. Online (0.010108)
13. Securities Account (0.004996)
```



Performed Random Forest feature importance to check which features are showing high variability in model.

Income , CCAvg, Education,Family are most important features.

```
df.head()
```

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

Performed Random Forest Classifier.

```python
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

X=df.drop(['Personal Loan' , 'ID' , 'Age' ,'Experience','CD Account'
,'Mortgage' ,'CreditCard' ,'Online', 'Securities Account'] , axis=1)
y=df["Personal Loan"]



X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)


clf = RandomForestClassifier(n_estimators=100, random_state=42)
clf.fit(X_train, y_train)


y_pred = clf.predict(X_test)

# Compute the accuracy score of the classifier
accuracy = accuracy_score(y_test, y_pred)

print("Accuracy: {:.2f}%".format(accuracy*100))
```

```
Accuracy: 98.51%
```

Performed Naïve Bayes.

```python
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score

X=df.drop(['Personal Loan' , 'ID' , 'Age' ,'Experience','CD Account'
,'Mortgage' ,'CreditCard' ,'Online', 'Securities Account'] , axis=1)
y=df["Personal Loan"]


# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Train a Gaussian Naive Bayes classifier
```

```
gnb = GaussianNB()
gnb.fit(X_train, y_train)

# Make predictions on the test set
y_pred = gnb.predict(X_test)

# Calculate the accuracy score
acc = accuracy_score(y_test, y_pred)
print("Accuracy score:", acc)
```

```
Accuracy score: 0.9541511771995044
```