Supervised Learning Classification Project: AllLife Bank Personal Loan Campaign

Problem Statement

Context

AllLife Bank is a US bank that has a growing customer base. The majority of these customers are liability customers (depositors) with varying sizes of deposits. The number of customers who are also borrowers (asset customers) is quite small, and the bank is interested in expanding this base rapidly to bring in more loan business and in the process, earn more through the interest on loans. In particular, the management wants to explore ways of converting its liability customers to personal loan customers (while retaining them as depositors).

A campaign that the bank ran last year for liability customers showed a healthy conversion rate of over 9% success. This has encouraged the retail marketing department to devise campaigns with better target marketing to increase the success ratio.

You as a Data scientist at AllLife bank have to build a model that will help the marketing department to identify the potential customers who have a higher probability of purchasing the loan.

Objective

To predict whether a liability customer will buy personal loans, to understand which customer attributes are most significant in driving purchases, and identify which segment of customers to target more.

Data Dictionary

- `ID` : Customer ID
- `Age` : Customer’s age in completed years
- `Experience` : #years of professional experience
- `Income` : Annual income of the customer (in thousand dollars)
- `ZIP Code` : Home Address ZIP code.
- `Family` : the Family size of the customer
- `CCAvg` : Average spending on credit cards per month (in thousand dollars)
- `Education` : Education Level. 1: Undergrad; 2: Graduate;3: Advanced/Professional
- `Mortgage` : Value of house mortgage if any. (in thousand dollars)
- `Personal_Loan` : Did this customer accept the personal loan offered in the last campaign? (0: No, 1: Yes)
- `Securities_Account` : Does the customer have securities account with the bank? (0: No, 1: Yes)
- `CD_Account` : Does the customer have a certificate of deposit (CD) account with the bank? (0: No, 1: Yes)
- `Online` : Do customers use internet banking facilities? (0: No, 1: Yes)
- `CreditCard` : Does the customer use a credit card issued by any other Bank (excluding All life Bank)? (0: No, 1: Yes)

Importing necessary libraries

In [40]:

```
# import libraries for data manipulation
import numpy as np
import pandas as pd

import warnings
warnings.filterwarnings("ignore")  # ignore warnings

# import libraries for data visualization
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats
from sklearn import metrics, tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import (confusion_matrix, classification_report,
                             accuracy_score, precision_score, recall_score, f1_score)
import warnings
warnings.filterwarnings("ignore")  # ignore warnings

%matplotlib inline
sns.set()

%matplotlib inline
```

Loading the dataset

Loading the dataset

In [41]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", for
ce_remount=True).
```

Data Overview

- Observations
- Sanity checks

In [42]:
```
# read the data
df = pd.read_csv('/content/drive/MyDrive/Loan_Modelling.csv')
# returns the first 5 rows
df.head()
```

Out[42]:

	ID	Age	Experience	Income	ZIPCode	Family	CCAvg	Education	Mortgage	Personal_Loan	Securities_Account	CD_Account	Online	C
0	1	25	1	49	91107	4	1.6	1	0	0	1	0	0	
1	2	45	19	34	90089	3	1.5	1	0	0	1	0	0	
2	3	39	15	11	94720	1	1.0	1	0	0	0	0	0	
3	4	35	9	100	94112	1	2.7	2	0	0	0	0	0	
4	5	35	8	45	91330	4	1.0	2	0	0	0	0	0	

In [5]:
```
df.shape
```

Out[5]:
```
(5000, 14)
```

In [6]:
```
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   ZIPCode             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
```

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

Out[7]:
```
ID                    0
Age                   0
Experience            0
Income                0
ZIPCode               0
Family                0
CCAvg                 0
Education             0
Mortgage              0
Personal_Loan         0
Securities_Account    0
CD_Account            0
Online                0
CreditCard            0
dtype: int64
```

In [8]:
```
df.describe(include = 'all').T
```

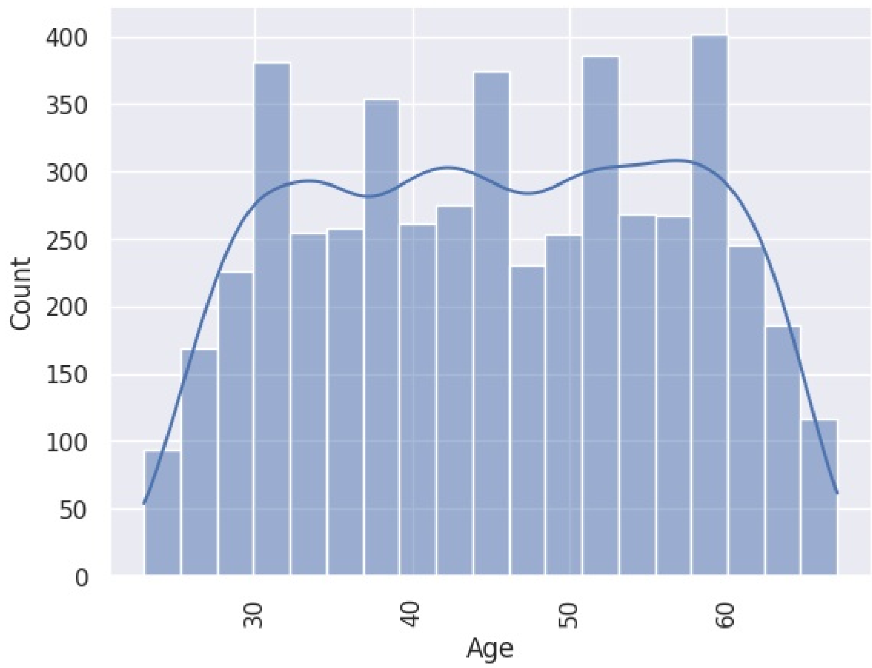
Out[8]:

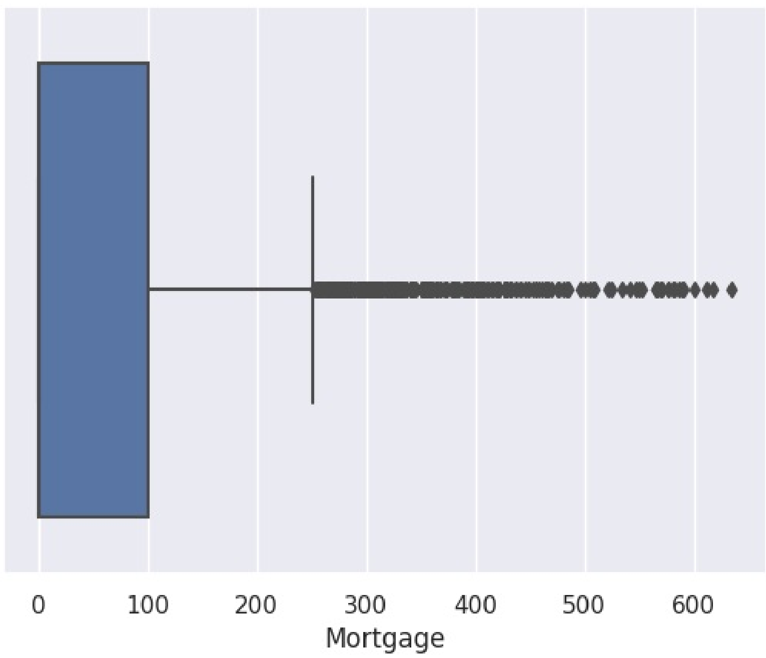
	count	mean	std	min	25%	50%	75%	max
ID	5000.0	2500.500000	1443.520003	1.0	1250.75	2500.5	3750.25	5000.0
Age	5000.0	45.338400	11.463166	23.0	35.00	45.0	55.00	67.0
Experience	5000.0	20.104600	11.467954	-3.0	10.00	20.0	30.00	43.0
Income	5000.0	73.774200	46.033729	8.0	39.00	64.0	98.00	224.0
ZIPCode	5000.0	93169.257000	1759.455086	90005.0	91911.00	93437.0	94608.00	96651.0
Family	5000.0	2.396400	1.147663	1.0	1.00	2.0	3.00	4.0
CCAvg	5000.0	1.937938	1.747659	0.0	0.70	1.5	2.50	10.0
Education	5000.0	1.881000	0.839869	1.0	1.00	2.0	3.00	3.0
Mortgage	5000.0	56.498800	101.713802	0.0	0.00	0.0	101.00	635.0
Personal_Loan	5000.0	0.096000	0.294621	0.0	0.00	0.0	0.00	1.0
Securities_Account	5000.0	0.104400	0.305809	0.0	0.00	0.0	0.00	1.0
CD_Account	5000.0	0.060400	0.238250	0.0	0.00	0.0	0.00	1.0
Online	5000.0	0.596800	0.490589	0.0	0.00	1.0	1.00	1.0
CreditCard	5000.0	0.294000	0.455637	0.0	0.00	0.0	1.00	1.0

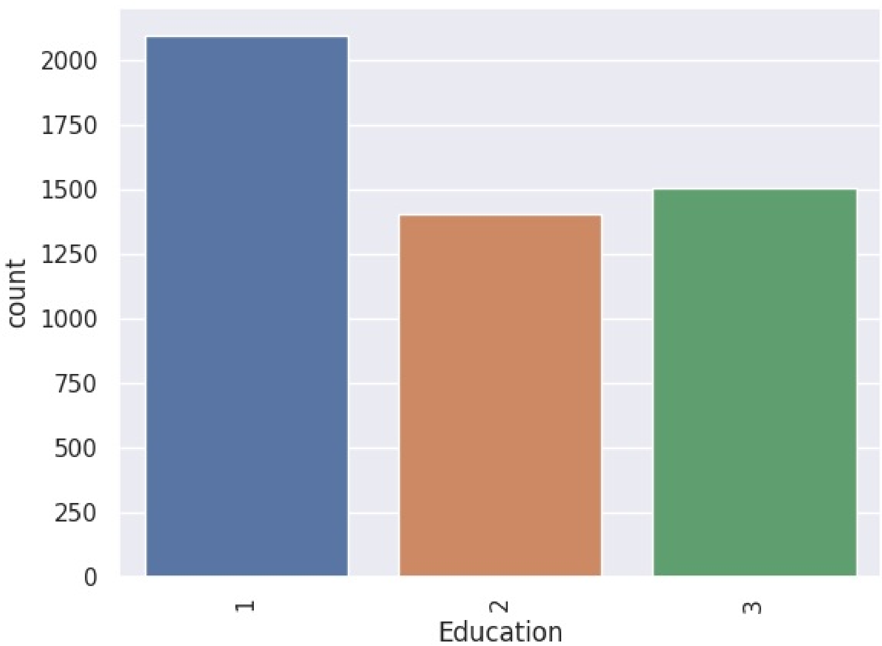
Exploratory Data Analysis.

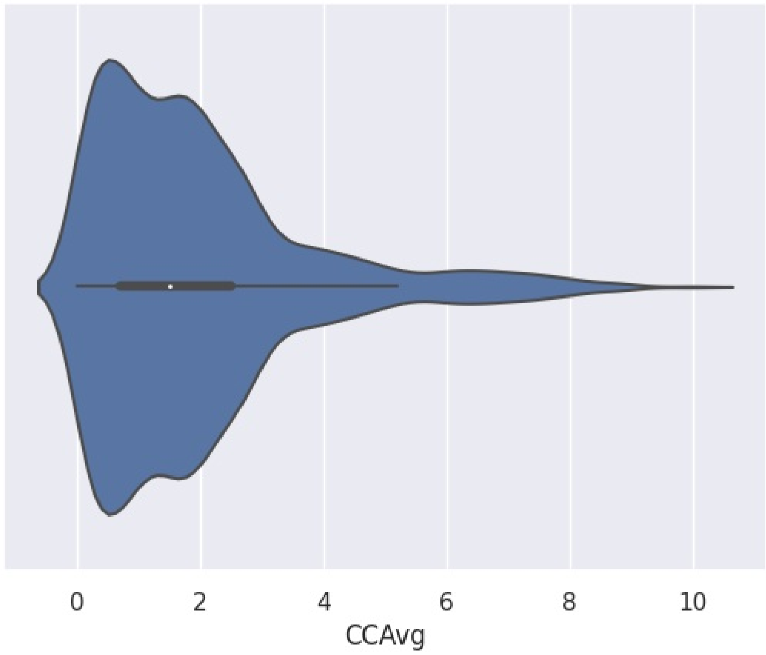
In [9]:

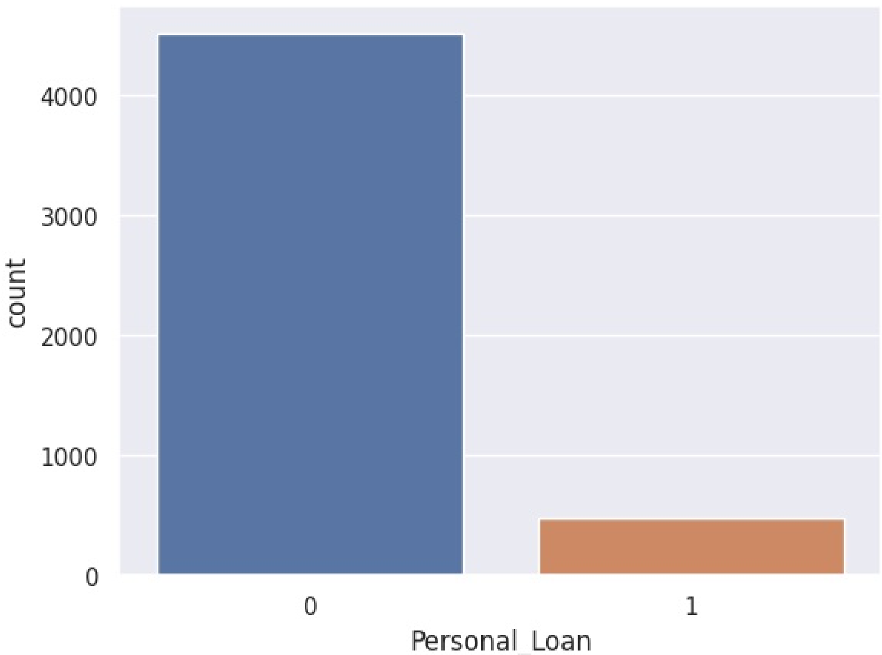
```
sns.histplot(data = df, x = 'Age', kde = True)
plt.xticks(rotation = 90);
plt.show()
sns.boxplot(df, x = 'Mortgage')
plt.show()
sns.countplot(data = df, x = 'Education')
plt.xticks(rotation = 90);
plt.show()
sns.violinplot(data = df, x = 'CCAvg')
plt.show()
sns.countplot(data = df, x = 'Personal_Loan')
plt.show()
plt.show()
sns.countplot(data = df, x = 'Securities_Account')
plt.show()
sns.histplot(data = df, x = 'Family', kde = True)
plt.show()
plt.figure(figsize = (20,7))
```

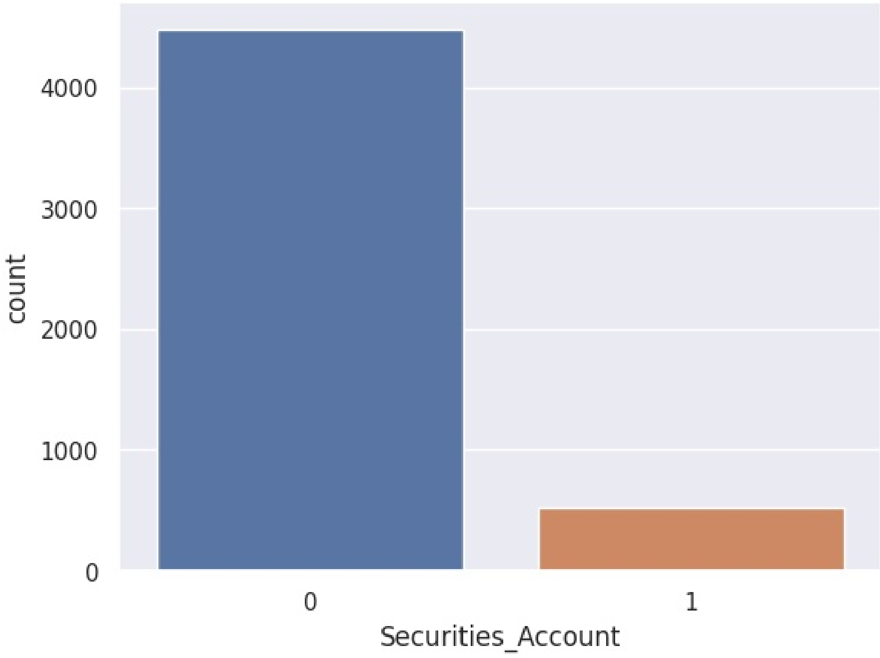


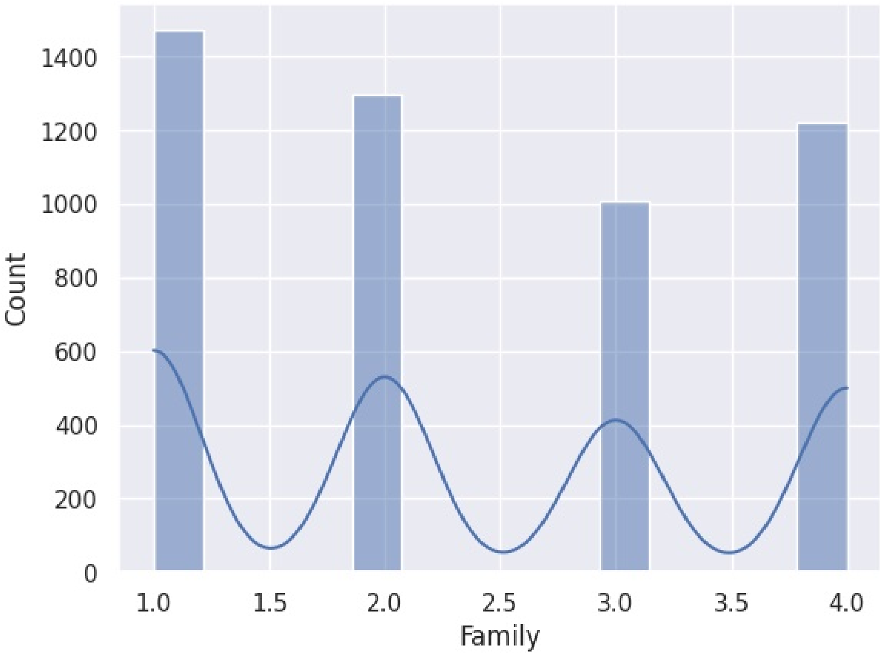

0
100
200
300
400
500
600
Mortgage


2000
1750
1500
1250
1000
750
500
250
0
count
1
2
3
Education


0
2
4
6
8
10
CCAvg


count
0
1000
2000
3000
4000
0
1
Personal_Loan

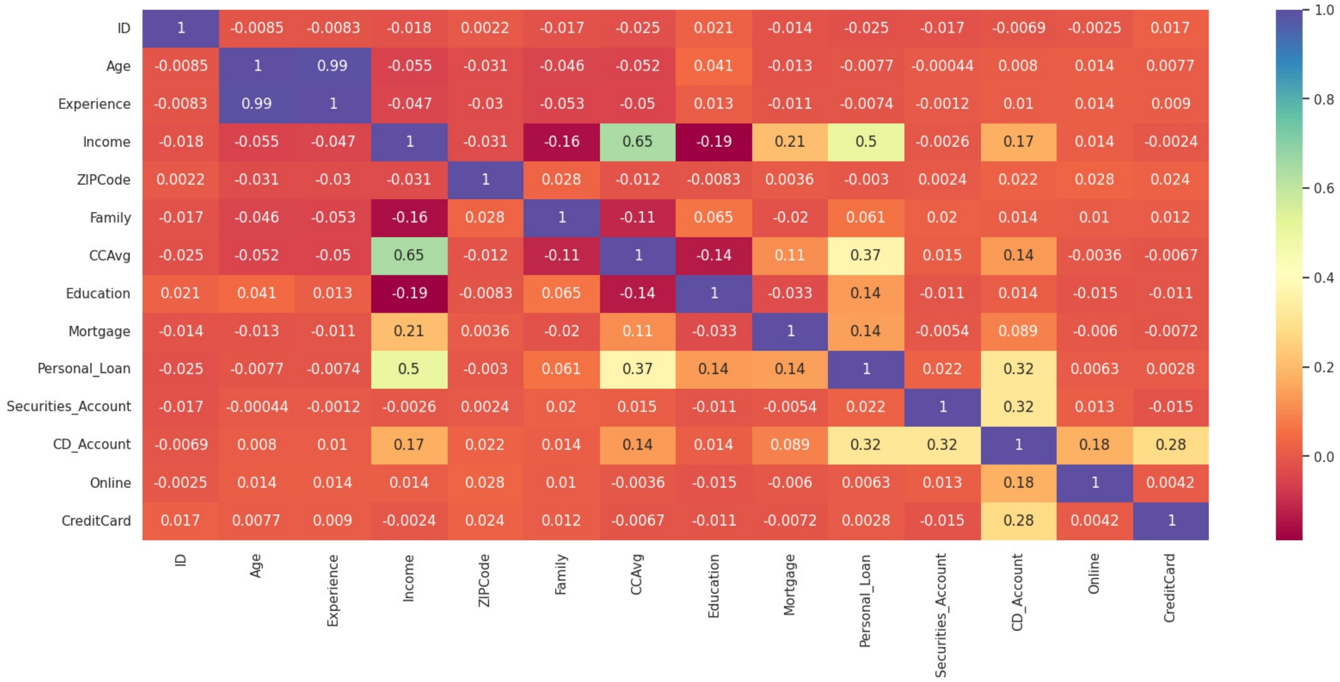




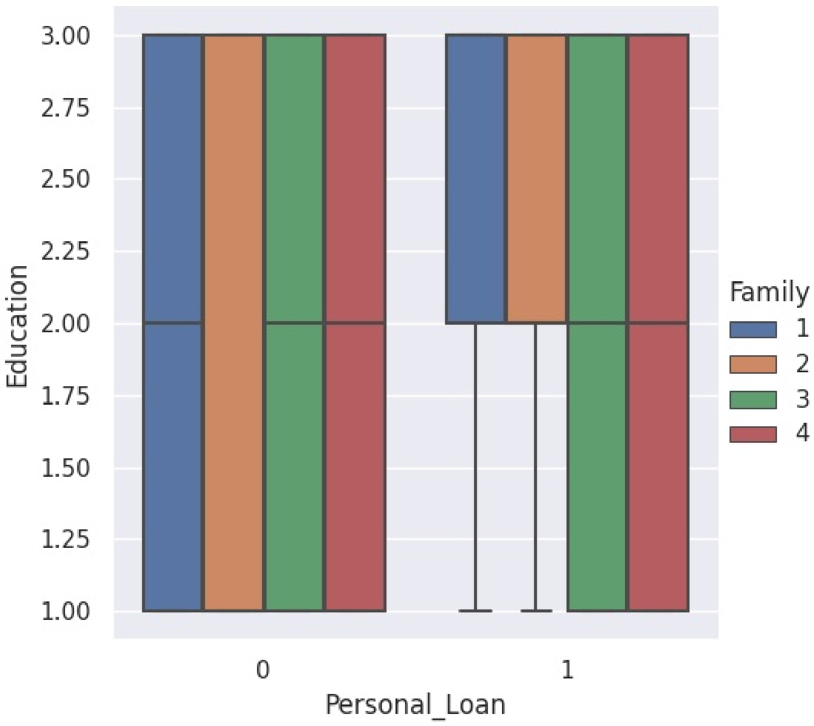
Out[9]: <Figure size 2000x700 with 0 Axes>

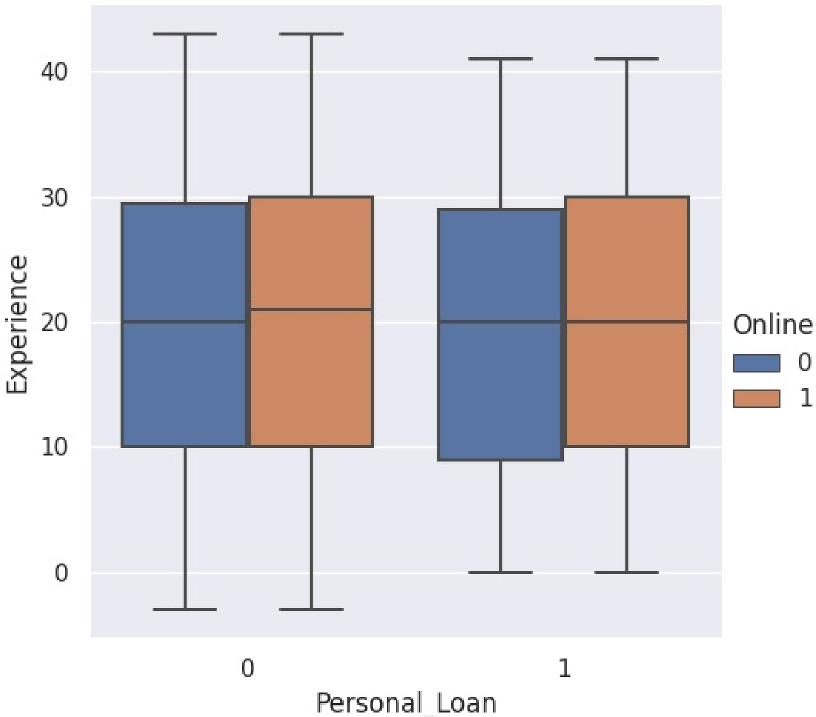
<Figure size 2000x700 with 0 Axes>

In [10]:
```
plt.figure(figsize=(20, 8))
sns.heatmap(df.corr(), annot=True, cmap="Spectral");
```



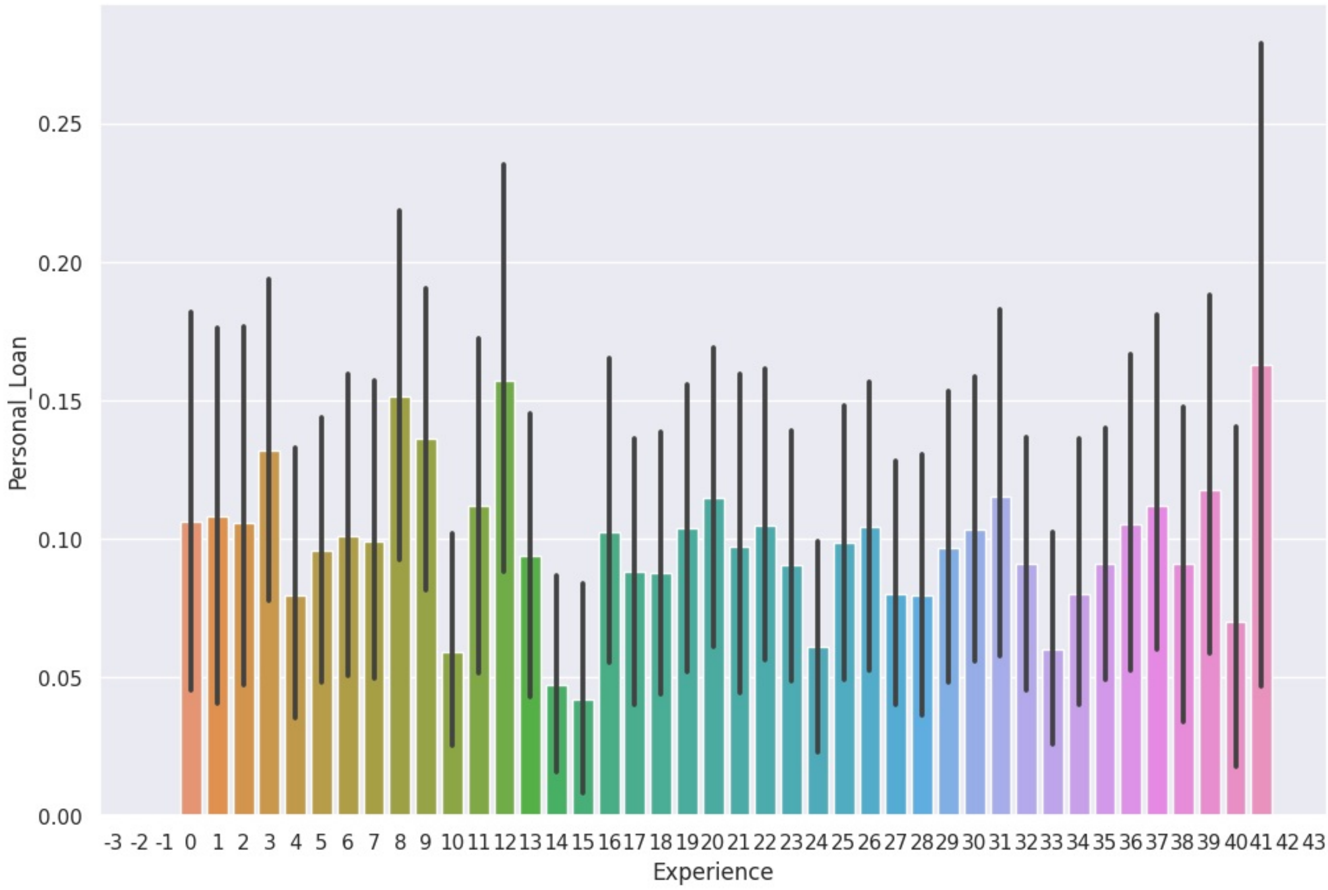
In [12]:
```
sns.catplot(kind="box",data=df, y="Education",x = "Personal_Loan", hue="Family");
sns.catplot(data=df, kind="box", y="Experience", x= "Personal_Loan", hue="Online");
```





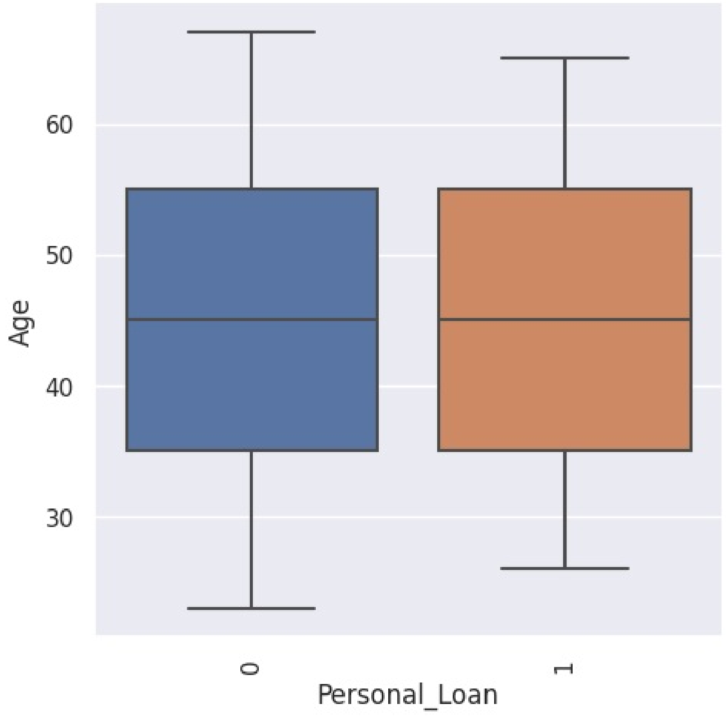
In [83]:
```
plt.figure(figsize=(12, 8))
sns.barplot( data = df, x="Experience", y = 'Personal_Loan')
```

Out[83]:
```
<Axes: xlabel='Experience', ylabel='Personal_Loan'>
```



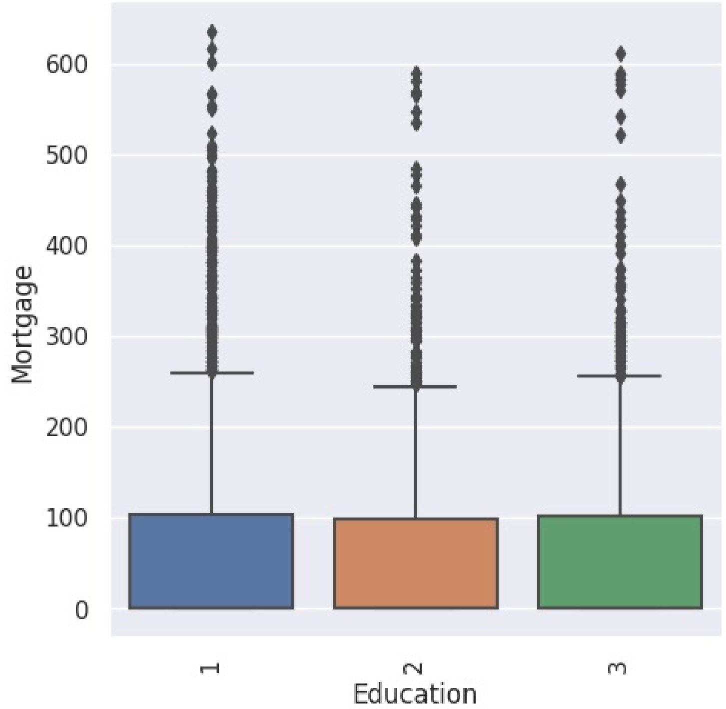
In [13]:

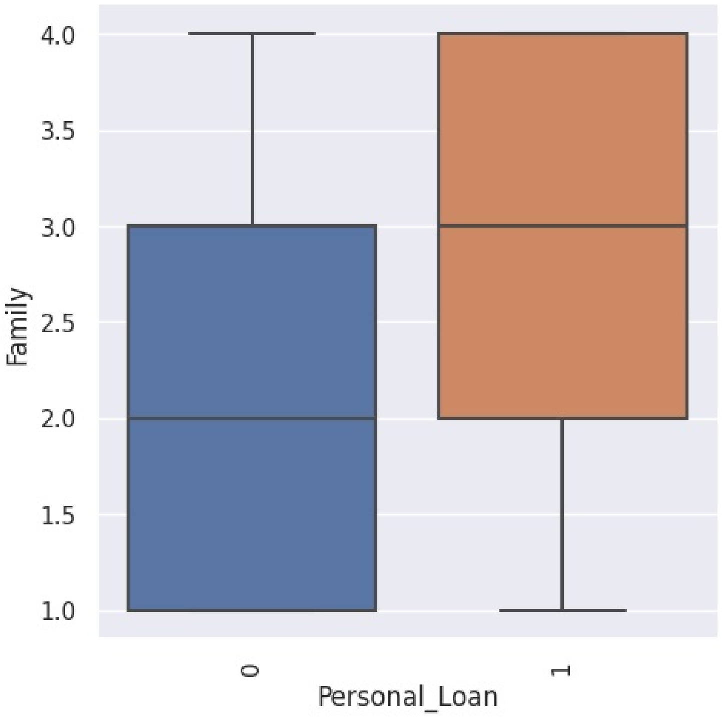
```
sns.catplot(data=df, kind="box", x="Personal_Loan", y = "Age");
plt.xticks(rotation=90);
```



In [14]:
```
credit_customer = df [["CreditCard"]].copy()
print (credit_customer["CreditCard"].value_counts())
sns.catplot(data=df, kind="box", x="Education", y = "Mortgage");
plt.xticks(rotation=90);
sns.catplot(data=df, kind="box", x="Personal_Loan", y = "Family");
plt.xticks(rotation=90);
```

```
0    3530
1    1470
Name: CreditCard, dtype: int64
```





The number of credit card holders is 1470.

People with undergrads have more mortgages.

People with education tend to have more personal Loan

Income and CCAge have the highest correlation which is 0.65.

Personal loan and income have the next highest correlation of 0.5.

Data Preprocessing

- Missing value treatment
- Feature engineering (if needed)
- Outlier detection and treatment (if needed)
- Preparing data for modeling
- Any other preprocessing steps (if needed)

In [15]:
```
df.isnull().values.any()
```

Out[15]:
```
False
```

In [16]:
```
def featuress(feature1: str, feature2: str, data=df):
    crosstab = pd.crosstab(data[feature1], data[feature2])
    chi, p_value, dof, expected =  stats.chi2_contingency(crosstab)
    Ho = f"{feature1} -> no effect on {feature2}"
    Ha = f"{feature1}-> an effect on {feature2}"
```

```
    if p_value < 0.05:
        print(f'{Ha.upper()} as the p_value ({p_value.round(3)}) < 0.05')
    else:
        print(f'{Ho} as the p_value ({p_value.round(3)}) > 0.05')

def feature_chart(features: list, data=df):
    for feature in features:
        print("="*30, feature, "="*(50-len(feature)))
        for col in list(data.columns):
            if col != feature: featuress(col , feature)

feature_chart(['Personal_Loan', 'CD_Account'])
```

```
============================== Personal_Loan =====================================
ID -> no effect on Personal_Loan as the p_value (0.493) > 0.05
Age -> no effect on Personal_Loan as the p_value (0.12) > 0.05
Experience -> no effect on Personal_Loan as the p_value (0.704) > 0.05
INCOME-> AN EFFECT ON PERSONAL_LOAN as the p_value (0.0) < 0.05
ZIPCode -> no effect on Personal_Loan as the p_value (0.76) > 0.05
FAMILY-> AN EFFECT ON PERSONAL_LOAN as the p_value (0.0) < 0.05
CCAVG-> AN EFFECT ON PERSONAL_LOAN as the p_value (0.0) < 0.05
EDUCATION-> AN EFFECT ON PERSONAL_LOAN as the p_value (0.0) < 0.05
MORTGAGE-> AN EFFECT ON PERSONAL_LOAN as the p_value (0.0) < 0.05
Securities_Account -> no effect on Personal_Loan as the p_value (0.141) > 0.05
CD_ACCOUNT-> AN EFFECT ON PERSONAL_LOAN as the p_value (0.0) < 0.05
Online -> no effect on Personal_Loan as the p_value (0.693) > 0.05
CreditCard -> no effect on Personal_Loan as the p_value (0.884) > 0.05
============================== CD_Account ========================================
ID -> no effect on CD_Account as the p_value (0.493) > 0.05
AGE-> AN EFFECT ON CD_ACCOUNT as the p_value (0.027) < 0.05
Experience -> no effect on CD_Account as the p_value (0.086) > 0.05
INCOME-> AN EFFECT ON CD_ACCOUNT as the p_value (0.0) < 0.05
ZIPCode -> no effect on CD_Account as the p_value (0.675) > 0.05
FAMILY-> AN EFFECT ON CD_ACCOUNT as the p_value (0.018) < 0.05
CCAVG-> AN EFFECT ON CD_ACCOUNT as the p_value (0.0) < 0.05
Education -> no effect on CD_Account as the p_value (0.58) > 0.05
MORTGAGE-> AN EFFECT ON CD_ACCOUNT as the p_value (0.0) < 0.05
PERSONAL_LOAN-> AN EFFECT ON CD_ACCOUNT as the p_value (0.0) < 0.05
SECURITIES_ACCOUNT-> AN EFFECT ON CD_ACCOUNT as the p_value (0.0) < 0.05
ONLINE-> AN EFFECT ON CD_ACCOUNT as the p_value (0.0) < 0.05
CREDITCARD-> AN EFFECT ON CD_ACCOUNT as the p_value (0.0) < 0.05
```

Model Building

In [17]:
```
df_dummies = pd.get_dummies(df, columns=['Education', 'Family'], drop_first=True)
df_dummies.head()
```

Out[17]:

	ID	Age	Experience	Income	ZIPCode	CCAvg	Mortgage	Personal_Loan	Securities_Account	CD_Account	Online	CreditCard	Educatior
0	1	25	1	49	91107	1.6	0	0	1	0	0	0	
1	2	45	19	34	90089	1.5	0	0	1	0	0	0	
2	3	39	15	11	94720	1.0	0	0	0	0	0	0	
3	4	35	9	100	94112	2.7	0	0	0	0	0	0	
4	5	35	8	45	91330	1.0	0	0	0	0	0	1	

In [18]:
```
df_dummies.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 17 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   ZIPCode             5000 non-null   int64
 5   CCAvg               5000 non-null   float64
 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
 12  Education_2         5000 non-null   uint8
 13  Education_3         5000 non-null   uint8
 14  Family_2            5000 non-null   uint8
 15  Family_3            5000 non-null   uint8
 16  Family_4            5000 non-null   uint8
dtypes: float64(1), int64(11), uint8(5)
memory usage: 493.3 KB
```

In [36]:
```
X = df_dummies.drop(['Personal_Loan'], axis=1)
```

```
X.head(5)
```

Out[36]:

	ID	Age	Experience	Income	ZIPCode	CCAvg	Mortgage	Securities_Account	CD_Account	Online	CreditCard	Education_2	Education_3
0	1	25	1	49	91107	1.6	0	1	0	0	0	0	0
1	2	45	19	34	90089	1.5	0	1	0	0	0	0	0
2	3	39	15	11	94720	1.0	0	0	0	0	0	0	0
3	4	35	9	100	94112	2.7	0	0	0	0	0	1	0
4	5	35	8	45	91330	1.0	0	0	0	0	1	1	0

In [37]:
```
y = df_dummies['Personal_Loan']
y.head(5)
```

Out[37]:
```
0    0
1    0
2    0
3    0
4    0
Name: Personal_Loan, dtype: int64
```

In [38]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1)
```

INITIAL MODEL

In [39]:
```
mod = DecisionTreeClassifier(criterion='gini',
                             class_weight={0:0.15, 1:0.85},
                             random_state=1)
mod.fit(X_train, y_train)
```

Out[39]:
```
DecisionTreeClassifier
DecisionTreeClassifier(class_weight={0: 0.15, 1: 0.85}, random_state=1)
```

In [31]:
```
print(tree.export_text(mod,feature_names=feature_names,show_weights=True))
```

```
|--- Income <= 98.50
|   |--- CCAvg <= 2.95
|   |   |--- weights: [374.10, 0.00] class: 0
|   |--- CCAvg >  2.95
|   |   |--- CD_Account <= 0.50
|   |   |   |--- CCAvg <= 3.95
|   |   |   |   |--- Income <= 81.50
|   |   |   |   |   |--- Age <= 36.50
|   |   |   |   |   |   |--- Family_4 <= 0.50
|   |   |   |   |   |   |   |--- CCAvg <= 3.50
|   |   |   |   |   |   |   |   |--- Family_3 <= 0.50
|   |   |   |   |   |   |   |   |   |--- weights: [0.00, 0.85] class: 1
|   |   |   |   |   |   |   |   |--- Family_3 >  0.50
|   |   |   |   |   |   |   |   |   |--- weights: [0.00, 0.85] class: 1
|   |   |   |   |   |   |   |--- CCAvg >  3.50
|   |   |   |   |   |   |   |   |--- weights: [0.15, 0.00] class: 0
|   |   |   |   |   |   |--- Family_4 >  0.50
|   |   |   |   |   |   |   |--- weights: [0.60, 0.00] class: 0
|   |   |   |   |   |--- Age >  36.50
|   |   |   |   |   |   |--- ZIPCode <= 91269.00
|   |   |   |   |   |   |   |--- ID <= 1184.50
|   |   |   |   |   |   |   |   |--- weights: [0.00, 0.85] class: 1
|   |   |   |   |   |   |   |--- ID >  1184.50
|   |   |   |   |   |   |   |   |--- weights: [1.05, 0.00] class: 0
|   |   |   |   |   |   |--- ZIPCode >  91269.00
|   |   |   |   |   |   |   |--- weights: [5.55, 0.00] class: 0
|   |   |   |   |--- Income >  81.50
|   |   |   |   |   |--- ID <= 934.50
|   |   |   |   |   |   |--- weights: [1.35, 0.00] class: 0
|   |   |   |   |   |--- ID >  934.50
|   |   |   |   |   |   |--- ZIPCode <= 95084.00
|   |   |   |   |   |   |   |--- CCAvg <= 3.05
|   |   |   |   |   |   |   |   |--- weights: [0.60, 0.00] class: 0
|   |   |   |   |   |   |   |--- CCAvg >  3.05
|   |   |   |   |   |   |   |   |--- Experience <= 38.50
|   |   |   |   |   |   |   |   |   |--- Family_4 <= 0.50
|   |   |   |   |   |   |   |   |   |   |--- ZIPCode <= 90692.00
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [0.15, 0.00] class: 0
|   |   |   |   |   |   |   |   |   |   |--- ZIPCode >  90692.00
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 4
|   |   |   |   |   |   |   |   |   |--- Family_4 >  0.50
|   |   |   |   |   |   |   |   |   |   |--- Education_3 <= 0.50
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [0.60, 0.00] class: 0
|   |   |   |   |   |   |   |   |   |   |--- Education_3 >  0.50
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [0.00, 0.85] class: 1
|   |   |   |   |   |   |   |   |--- Experience >  38.50
|   |   |   |   |   |   |   |   |   |--- Online <= 0.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [0.15, 0.00] class: 0
```

```
|   |   |   |   |   |   |   |   |   |--- Online >  0.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [0.15, 0.00] class: 0
|   |   |   |   |   |   |--- ZIPCode >  95084.00
|   |   |   |   |   |   |   |--- Income <= 82.50
|   |   |   |   |   |   |   |   |--- weights: [0.15, 0.00] class: 0
|   |   |   |   |   |   |   |--- Income >  82.50
|   |   |   |   |   |   |   |   |--- weights: [0.75, 0.00] class: 0
|   |   |   |--- CCAvg >  3.95
|   |   |   |   |--- weights: [6.75, 0.00] class: 0
|   |   |--- CD_Account >  0.50
|   |   |   |--- ID <= 766.50
|   |   |   |   |--- weights: [0.15, 0.00] class: 0
|   |   |   |--- ID >  766.50
|   |   |   |   |--- weights: [0.00, 6.80] class: 1
|--- Income >  98.50
|   |--- Education_3 <= 0.50
|   |   |--- Education_2 <= 0.50
|   |   |   |--- Family_3 <= 0.50
|   |   |   |   |--- Family_4 <= 0.50
|   |   |   |   |   |--- Income <= 100.00
|   |   |   |   |   |   |--- CCAvg <= 4.20
|   |   |   |   |   |   |   |--- weights: [0.45, 0.00] class: 0
|   |   |   |   |   |   |--- CCAvg >  4.20
|   |   |   |   |   |   |   |--- Securities_Account <= 0.50
|   |   |   |   |   |   |   |   |--- weights: [0.00, 0.85] class: 1
|   |   |   |   |   |   |   |--- Securities_Account >  0.50
|   |   |   |   |   |   |   |   |--- weights: [0.00, 0.85] class: 1
|   |   |   |   |   |--- Income >  100.00
|   |   |   |   |   |   |--- Income <= 103.50
|   |   |   |   |   |   |   |--- CCAvg <= 3.06
|   |   |   |   |   |   |   |   |--- weights: [2.10, 0.00] class: 0
|   |   |   |   |   |   |   |--- CCAvg >  3.06
|   |   |   |   |   |   |   |   |--- Experience <= 14.50
|   |   |   |   |   |   |   |   |   |--- weights: [0.15, 0.00] class: 0
|   |   |   |   |   |   |   |   |--- Experience >  14.50
|   |   |   |   |   |   |   |   |   |--- weights: [0.00, 0.85] class: 1
|   |   |   |   |   |   |--- Income >  103.50
|   |   |   |   |   |   |   |--- ID <= 20.00
|   |   |   |   |   |   |   |   |--- weights: [0.15, 0.00] class: 0
|   |   |   |   |   |   |   |--- ID >  20.00
|   |   |   |   |   |   |   |   |--- weights: [64.80, 0.00] class: 0
|   |   |   |   |--- Family_4 >  0.50
|   |   |   |   |   |--- Income <= 102.00
|   |   |   |   |   |   |--- weights: [0.15, 0.00] class: 0
|   |   |   |   |   |--- Income >  102.00
|   |   |   |   |   |   |--- Age <= 34.00
|   |   |   |   |   |   |   |--- weights: [0.00, 0.85] class: 1
|   |   |   |   |   |   |--- Age >  34.00
|   |   |   |   |   |   |   |--- weights: [0.00, 15.30] class: 1
|   |   |   |--- Family_3 >  0.50
|   |   |   |   |--- Income <= 108.50
|   |   |   |   |   |--- weights: [1.05, 0.00] class: 0
|   |   |   |   |--- Income >  108.50
|   |   |   |   |   |--- Age <= 26.00
|   |   |   |   |   |   |--- weights: [0.15, 0.00] class: 0
|   |   |   |   |   |--- Age >  26.00
|   |   |   |   |   |   |--- ZIPCode <= 90019.50
|   |   |   |   |   |   |   |--- weights: [0.15, 0.00] class: 0
|   |   |   |   |   |   |--- ZIPCode >  90019.50
|   |   |   |   |   |   |   |--- Income <= 118.00
|   |   |   |   |   |   |   |   |--- ID <= 2808.00
|   |   |   |   |   |   |   |   |   |--- CCAvg <= 3.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [0.00, 0.85] class: 1
|   |   |   |   |   |   |   |   |   |--- CCAvg >  3.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [0.00, 0.85] class: 1
|   |   |   |   |   |   |   |   |--- ID >  2808.00
|   |   |   |   |   |   |   |   |   |--- weights: [0.15, 0.00] class: 0
|   |   |   |   |   |   |   |--- Income >  118.00
|   |   |   |   |   |   |   |   |--- weights: [0.00, 28.05] class: 1
|   |   |--- Education_2 >  0.50
|   |   |   |--- Income <= 110.50
|   |   |   |   |--- CCAvg <= 3.54
|   |   |   |   |   |--- Income <= 106.50
|   |   |   |   |   |   |--- Family_2 <= 0.50
|   |   |   |   |   |   |   |--- weights: [3.15, 0.00] class: 0
|   |   |   |   |   |   |--- Family_2 >  0.50
|   |   |   |   |   |   |   |--- weights: [0.75, 0.00] class: 0
|   |   |   |   |   |--- Income >  106.50
|   |   |   |   |   |   |--- Family_4 <= 0.50
|   |   |   |   |   |   |   |--- weights: [0.75, 0.00] class: 0
|   |   |   |   |   |   |--- Family_4 >  0.50
|   |   |   |   |   |   |   |--- Age <= 42.00
|   |   |   |   |   |   |   |   |--- weights: [0.30, 0.00] class: 0
|   |   |   |   |   |   |   |--- Age >  42.00
|   |   |   |   |   |   |   |   |--- weights: [0.00, 0.85] class: 1
|   |   |   |   |--- CCAvg >  3.54
|   |   |   |   |   |--- weights: [0.00, 2.55] class: 1
|   |   |   |--- Income >  110.50
|   |   |   |   |--- Income <= 116.50
```

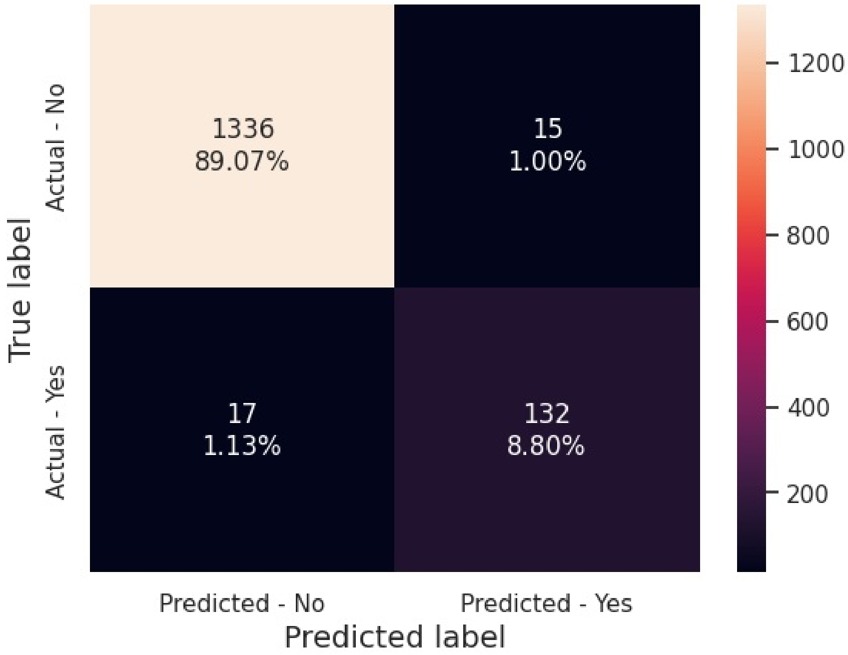
```
|   |   |   |   |   |--- Mortgage <= 141.50
|   |   |   |   |   |   |--- Age <= 60.50
|   |   |   |   |   |   |   |--- CCAvg <= 1.20
|   |   |   |   |   |   |   |   |--- weights: [0.30, 0.00] class: 0
|   |   |   |   |   |   |   |--- CCAvg >  1.20
|   |   |   |   |   |   |   |   |--- ZIPCode <= 94887.00
|   |   |   |   |   |   |   |   |   |--- CCAvg <= 2.65
|   |   |   |   |   |   |   |   |   |   |--- Income <= 113.50
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [0.00, 1.70] class: 1
|   |   |   |   |   |   |   |   |   |   |--- Income >  113.50
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 2
|   |   |   |   |   |   |   |   |   |--- CCAvg >  2.65
|   |   |   |   |   |   |   |   |   |   |--- Age <= 31.50
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [0.00, 0.85] class: 1
|   |   |   |   |   |   |   |   |   |   |--- Age >  31.50
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [0.00, 3.40] class: 1
|   |   |   |   |   |   |   |   |--- ZIPCode >  94887.00
|   |   |   |   |   |   |   |   |   |--- weights: [0.15, 0.00] class: 0
|   |   |   |   |   |   |--- Age >  60.50
|   |   |   |   |   |   |   |--- Income <= 114.50
|   |   |   |   |   |   |   |   |--- weights: [0.15, 0.00] class: 0
|   |   |   |   |   |   |   |--- Income >  114.50
|   |   |   |   |   |   |   |   |--- weights: [0.30, 0.00] class: 0
|   |   |   |   |   |--- Mortgage >  141.50
|   |   |   |   |   |   |--- Income <= 112.50
|   |   |   |   |   |   |   |--- weights: [0.30, 0.00] class: 0
|   |   |   |   |   |   |--- Income >  112.50
|   |   |   |   |   |   |   |--- weights: [0.30, 0.00] class: 0
|   |   |   |   |--- Income >  116.50
|   |   |   |   |   |--- weights: [0.00, 91.80] class: 1
|   |--- Education_3 >  0.50
|   |   |--- Income <= 116.50
|   |   |   |--- CCAvg <= 2.45
|   |   |   |   |--- Age <= 41.50
|   |   |   |   |   |--- weights: [3.60, 0.00] class: 0
|   |   |   |   |--- Age >  41.50
|   |   |   |   |   |--- Experience <= 31.50
|   |   |   |   |   |   |--- Online <= 0.50
|   |   |   |   |   |   |   |--- weights: [0.45, 0.00] class: 0
|   |   |   |   |   |   |--- Online >  0.50
|   |   |   |   |   |   |   |--- ZIPCode <= 93596.00
|   |   |   |   |   |   |   |   |--- weights: [0.00, 2.55] class: 1
|   |   |   |   |   |   |   |--- ZIPCode >  93596.00
|   |   |   |   |   |   |   |   |--- weights: [0.15, 0.00] class: 0
|   |   |   |   |   |--- Experience >  31.50
|   |   |   |   |   |   |--- weights: [1.50, 0.00] class: 0
|   |   |   |--- CCAvg >  2.45
|   |   |   |   |--- ZIPCode <= 90389.50
|   |   |   |   |   |--- weights: [0.15, 0.00] class: 0
|   |   |   |   |--- ZIPCode >  90389.50
|   |   |   |   |   |--- ID <= 4852.50
|   |   |   |   |   |   |--- CD_Account <= 0.50
|   |   |   |   |   |   |   |--- Income <= 99.50
|   |   |   |   |   |   |   |   |--- ZIPCode <= 93882.50
|   |   |   |   |   |   |   |   |   |--- weights: [0.30, 0.00] class: 0
|   |   |   |   |   |   |   |   |--- ZIPCode >  93882.50
|   |   |   |   |   |   |   |   |   |--- weights: [0.00, 1.70] class: 1
|   |   |   |   |   |   |   |--- Income >  99.50
|   |   |   |   |   |   |   |   |--- weights: [0.00, 11.05] class: 1
|   |   |   |   |   |   |--- CD_Account >  0.50
|   |   |   |   |   |   |   |--- Experience <= 21.00
|   |   |   |   |   |   |   |   |--- weights: [0.30, 0.00] class: 0
|   |   |   |   |   |   |   |--- Experience >  21.00
|   |   |   |   |   |   |   |   |--- weights: [0.00, 0.85] class: 1
|   |   |   |   |   |--- ID >  4852.50
|   |   |   |   |   |   |--- weights: [0.15, 0.00] class: 0
|   |   |--- Income >  116.50
|   |   |   |--- ID <= 13.50
|   |   |   |   |--- weights: [0.00, 0.85] class: 1
|   |   |   |--- ID >  13.50
|   |   |   |   |--- weights: [0.00, 96.05] class: 1
```

In [75]:

```
def confusion_matrix( model, y_actual, labels=[1, 0], xtest=X_test):
    y_predict = model.predict(xtest)
    cm = metrics.confusion_matrix(y_actual, y_predict, labels=[0, 1])
    df_cm = pd.DataFrame(cm, index=["Actual - No","Actual - Yes"],
                         columns=['Predicted - No','Predicted - Yes'])


    group_counts = [f"{value:0.0f}" for value in cm.flatten()]
    group_percentages = [f"{value:.2%}" for value in cm.flatten()/np.sum(cm)]

    labels = [f"{gc}\n{gp}" for gc, gp in zip(group_counts, group_percentages)]
    labels = np.asarray(labels).reshape(2,2)


    sns.heatmap(df_cm, annot=labels, fmt='')
    plt.ylabel('True label', fontsize=14)
```

```
    plt.xlabel('Predicted label', fontsize=14);
```

In [76]:
```
confusion_matrix(model, y_test)
```



In [26]:
```
y_train.value_counts(normalize=True)
```

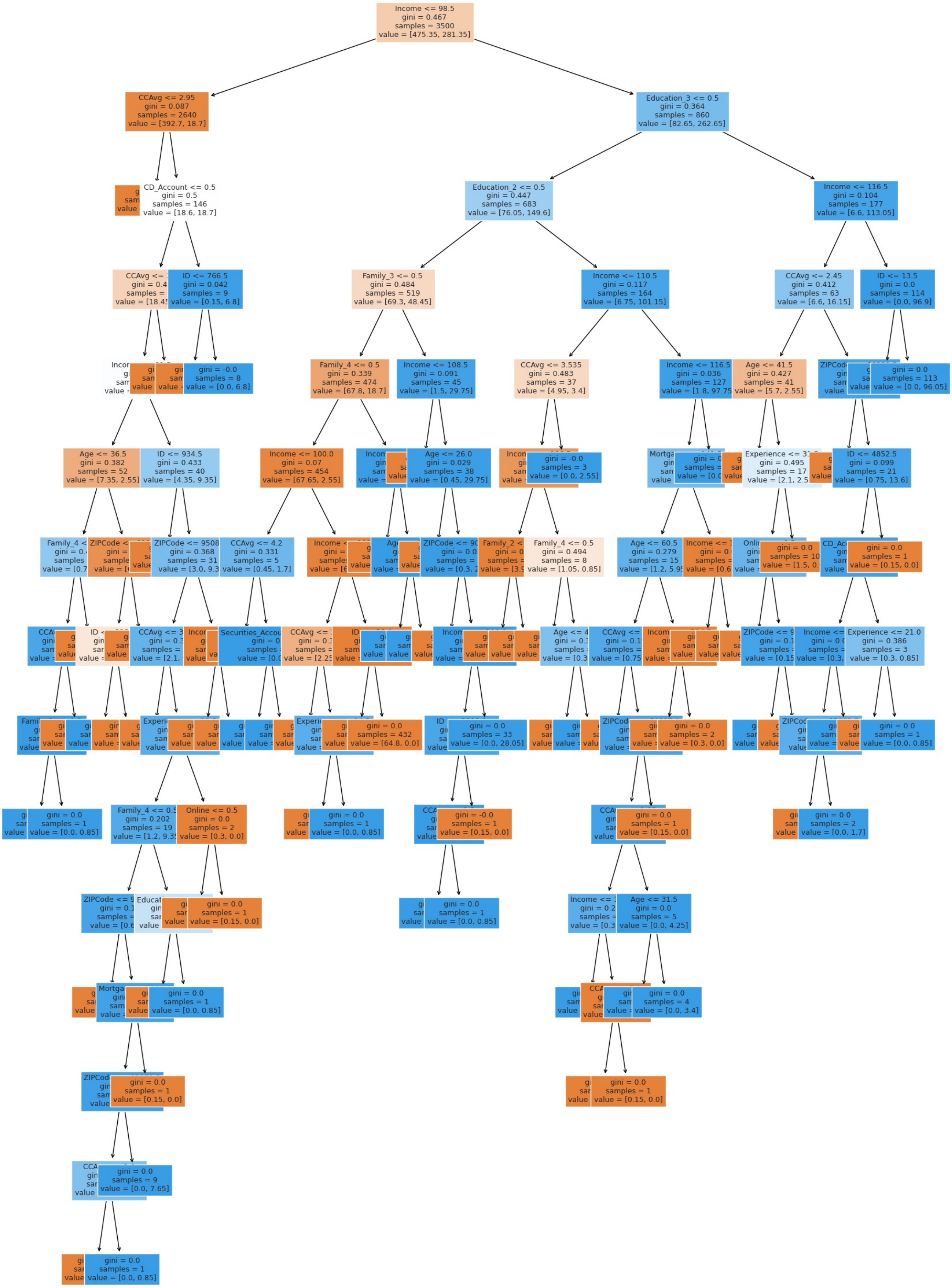
Out[26]:
```
0    0.905429
1    0.094571
Name: Personal_Loan, dtype: float64
```

BASELINE MODEL DECISION TREE

In [27]:
```
feature_names = list(X.columns)
print(feature_names)
```

```
['ID', 'Age', 'Experience', 'Income', 'ZIPCode', 'CCAvg', 'Mortgage', 'Securities_Account', 'CD_Account', 'Onli
ne', 'CreditCard', 'Education_2', 'Education_3', 'Family_2', 'Family_3', 'Family_4']
```

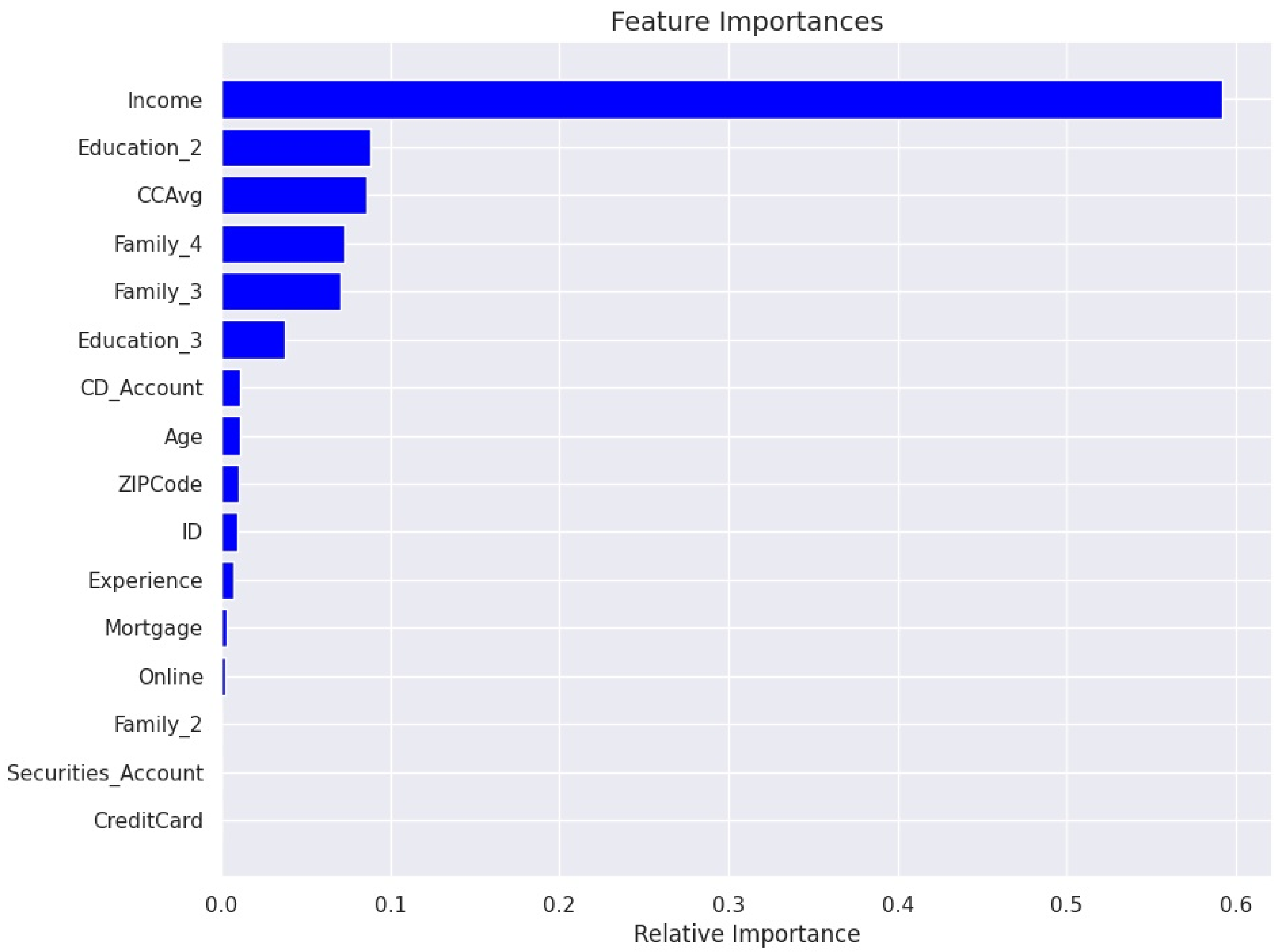
In [47]:
```
plt.figure(figsize=(20, 30))
out = tree.plot_tree(mod,
                     feature_names=feature_names,
                     filled=True,
                     fontsize=9,
                     node_ids=False,
                     class_names=None,)
for i in out:
     arrow = i.arrow_patch
     if arrow is not None:
        arrow.set_edgecolor('black')
        arrow.set_linewidth(1)
plt.show()
```



Model Performance Improvement

In [49]:
```python
def imp_plot(model):

    importances = model.feature_importances_
    indices = np.argsort(importances)
    size = len(indices)//2

    plt.figure(figsize=(10, size))
```

```
    plt.title("Feature Importances", fontsize=14)
    plt.barh(range(len(indices)), importances[indices], color='blue', align='center')
    plt.yticks(range(len(indices)), [feature_names[i] for i in indices])
    plt.xlabel("Relative Importance", fontsize=12);
imp_plot(mod)
```



In [51]:
```
pd.DataFrame(mod.feature_importances_,
                    columns=["Imp"],
                    index=X_train.columns).sort_values(by='Imp', ascending=False)
```

Out[51]:

	Imp
Income	5.925077e-01
Education_2	8.813411e-02
CCAvg	8.563040e-02
Family_4	7.261065e-02
Family_3	7.032437e-02
Education_3	3.713789e-02
CD_Account	1.133793e-02
Age	1.092303e-02
ZIPCode	1.047472e-02
ID	8.985089e-03
Experience	7.040014e-03
Mortgage	2.947456e-03
Online	1.946623e-03
Family_2	9.799370e-18
Securities_Account	1.601820e-18
CreditCard	0.000000e+00

USING GRIDSEARCH

In [52]:
```
estimator = DecisionTreeClassifier(random_state=1, class_weight={0:.15,1:.85})
parameters = {'max_depth': np.arange(1,10),
              'criterion': ['entropy','gini'],
              'splitter': ['best','random'],
              'min_impurity_decrease': [0.000001,0.00001,0.0001],
```

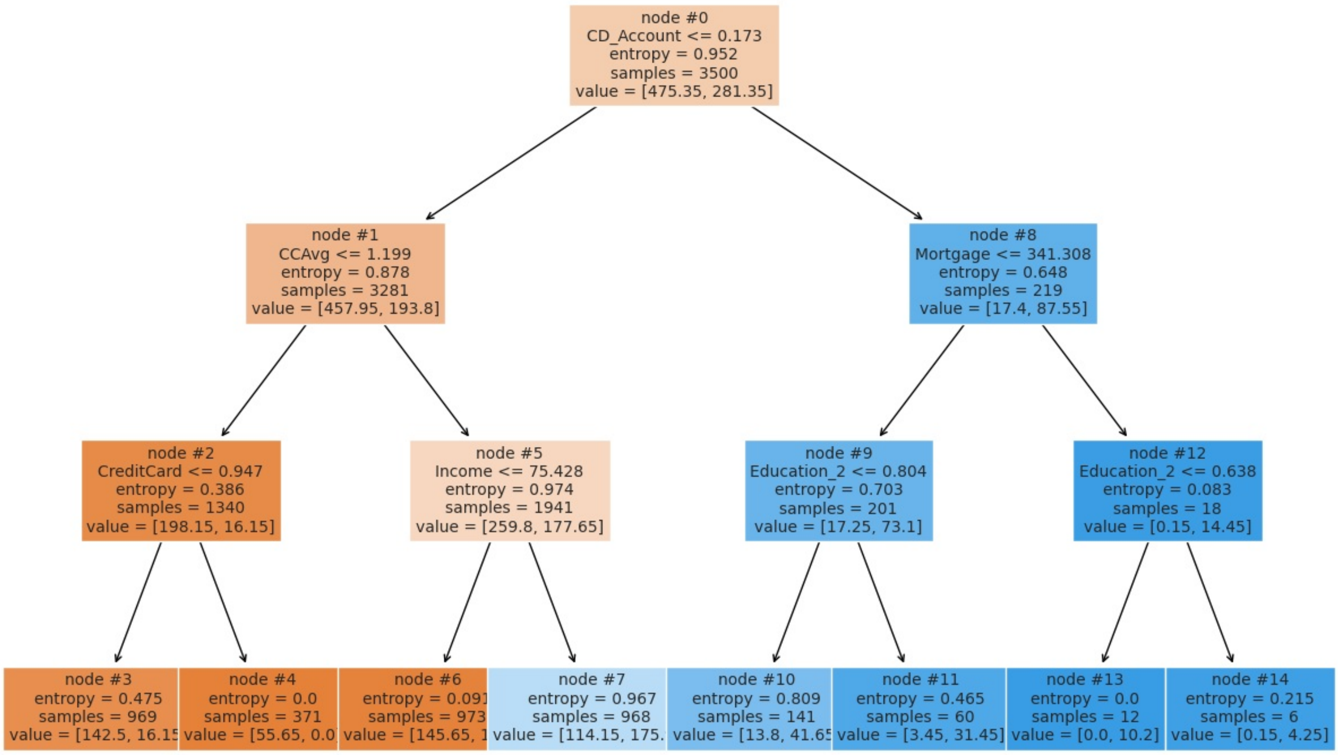
```python
              'max_features': ['log2','sqrt']}

scorer = metrics.make_scorer(metrics.recall_score)
grid_obj = GridSearchCV(estimator, param_grid=parameters, scoring=scorer, cv=5)
grid_obj = grid_obj.fit(X_train, y_train)

estimator = grid_obj.best_estimator_

estimator.fit(X_train, y_train)
```

Out[52]:

```
DecisionTreeClassifier

DecisionTreeClassifier(class_weight={0: 0.15, 1: 0.85}, criterion='entropy',
                       max_depth=3, max_features='log2',
                       min_impurity_decrease=1e-06, random_state=1,
                       splitter='random')
```

In [53]:

```python
plt.figure(figsize=(15, 10))
out = tree.plot_tree(estimator,
                     feature_names=feature_names,
                     filled=True,
                     fontsize=10,
                     node_ids=True,
                     class_names=None)
for i in out:
    arrow = i.arrow_patch
    if arrow is not None:
        arrow.set_edgecolor('black')
        arrow.set_linewidth(1)
plt.show()
```



In [54]:

```python
print(tree.export_text(estimator,
                       feature_names=feature_names,
                       show_weights=True))
```

```
|--- CD_Account <= 0.17
|   |--- CCAvg <= 1.20
|   |   |--- CreditCard <= 0.95
|   |   |   |--- weights: [142.50, 16.15] class: 0
|   |   |--- CreditCard >  0.95
|   |   |   |--- weights: [55.65, 0.00] class: 0
|   |--- CCAvg >  1.20
|   |   |--- Income <= 75.43
|   |   |   |--- weights: [145.65, 1.70] class: 0
|   |   |--- Income >  75.43
|   |   |   |--- weights: [114.15, 175.95] class: 1
|--- CD_Account >  0.17
|   |--- Mortgage <= 341.31
|   |   |--- Education_2 <= 0.80
|   |   |   |--- weights: [13.80, 41.65] class: 1
|   |   |--- Education_2 >  0.80
|   |   |   |--- weights: [3.45, 31.45] class: 1
|   |--- Mortgage >  341.31
|   |   |--- Education_2 <= 0.64
|   |   |   |--- weights: [0.00, 10.20] class: 1
|   |   |--- Education_2 >  0.64
|   |   |   |--- weights: [0.15, 4.25] class: 1
```

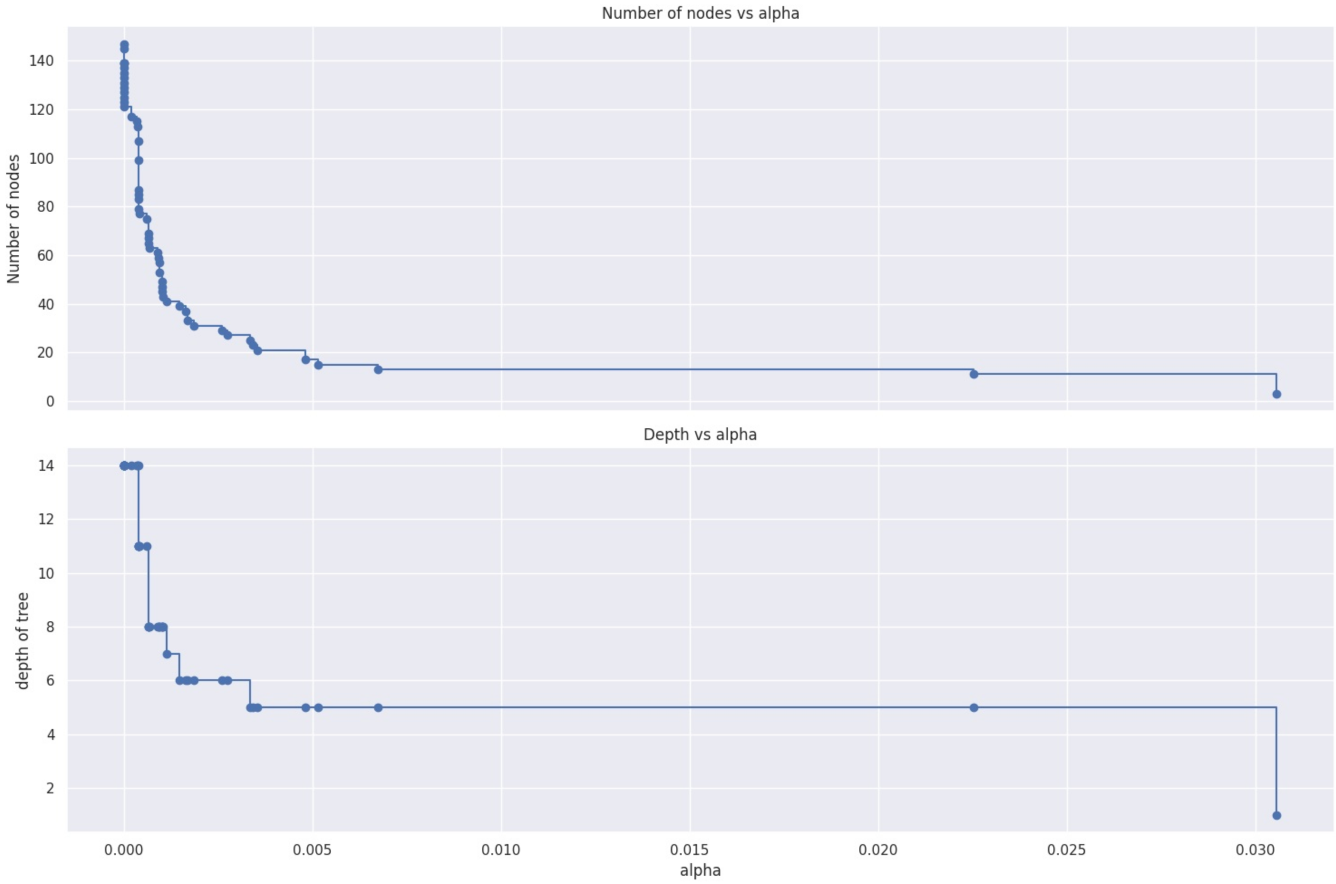
Total impurity of leaves vs effective alphas of pruned tree

In [55]:
```
cf = DecisionTreeClassifier(random_state=1, class_weight = {0:0.15, 1:0.85})
path = cf.cost_complexity_pruning_path(X_train, y_train)
ccp_alphas, impurities = path.ccp_alphas, path.impurities
```

In [56]:
```
cfs = []
for ccp_alpha in ccp_alphas:
    cf = DecisionTreeClassifier(random_state=1,
                                 ccp_alpha=ccp_alpha,
                                 class_weight = {0:0.15,1:0.85})
    cf.fit(X_train, y_train)
    cfs.append(cf)

print(f"Last tree has number of nodes: {cfs[-1].tree_.node_count} with ccp_alpha: {ccp_alphas[-1]}")
```

```
Last tree has number of nodes: 1 with ccp_alpha: 0.25379571489480973
```

In [57]:
```
cfs = cfs[:-1]
ccp_alphas = ccp_alphas[:-1]

node_counts = [cf.tree_.node_count for cf in cfs]
depth = [cf.tree_.max_depth for cf in cfs]
fig, ax = plt.subplots(2, 1, figsize=(15, 10), sharex=True)
ax[0].plot(ccp_alphas, node_counts, marker='o', drawstyle="steps-post")
ax[0].set_ylabel("Number of nodes")
ax[0].set_title("Number of nodes vs alpha")
ax[1].plot(ccp_alphas, depth, marker='o', drawstyle="steps-post")
ax[1].set_xlabel("alpha")
ax[1].set_ylabel("depth of tree")
ax[1].set_title("Depth vs alpha")
fig.tight_layout()
```



Model Comparison and Final Model Selection

In [58]:
```
def rrecall_score(model):

    pred_train = model.predict(X_train)
    pred_test = model.predict(X_test)
    print("Recall for training set : ", metrics.recall_score(y_train, pred_train))
    print("Recall for test set : ", metrics.recall_score(y_test, pred_test))
```

In [59]:
```
model = DecisionTreeClassifier(criterion='gini',
                               class_weight={0:0.15, 1:0.85},
                               random_state=1)
model.fit(X_train, y_train)
rrecall_score(model)
```

```
Recall for training set :  1.0
Recall for test set :  0.8859060402684564
```

In [60]:
```
rrecall_score(estimator)
```

```
Recall for training set :  0.9365558912386707
Recall for test set :  0.8523489932885906
```

In [69]:
```
best_model2 = DecisionTreeClassifier(ccp_alpha=0.01,
                                     class_weight={0: 0.15, 1: 0.85},
                                     random_state=1)
best_model2.fit(X_train, y_train)
```

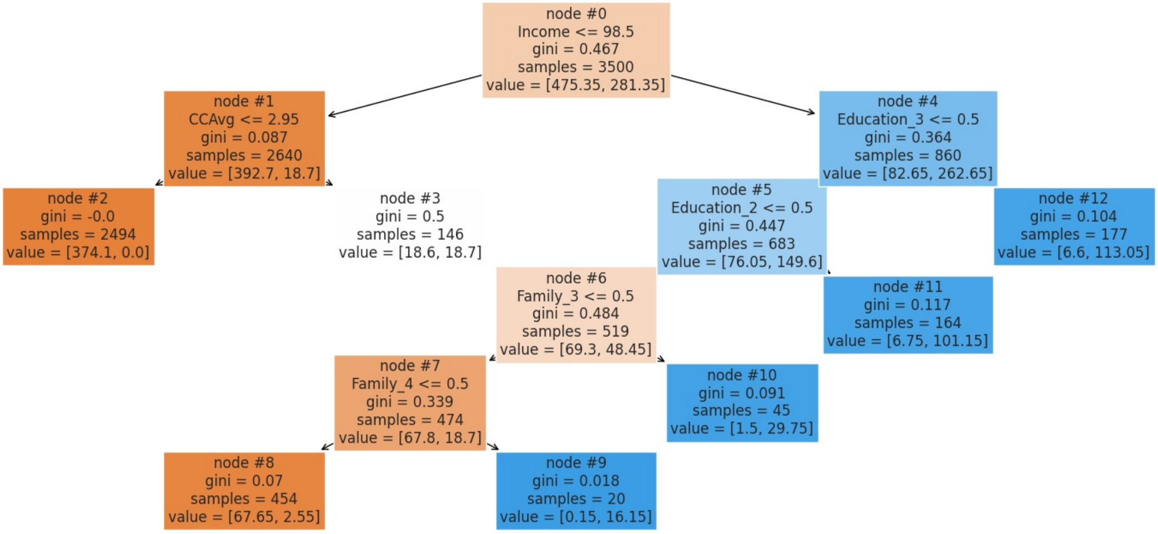
Out[69]:
```
▾                  DecisionTreeClassifier
DecisionTreeClassifier(ccp_alpha=0.01, class_weight={0: 0.15, 1: 0.85},
                       random_state=1)
```

In [65]:
```
rrecall_score(best_model2)
```

```
Recall for training set :  0.9909365558912386
Recall for test set :  0.9865771812080537
```

In [66]:
```
plt.figure(figsize=(20, 8))

out = tree.plot_tree(best_model2,
                     feature_names=feature_names,
                     filled=True,
                     fontsize=12,
                     node_ids=True,
                     class_names=None)
for o in out:
    arrow = o.arrow_patch
    if arrow is not None:
```

```
        arrow.set_edgecolor('black')
        arrow.set_linewidth(1)
plt.show()
```



In [181…

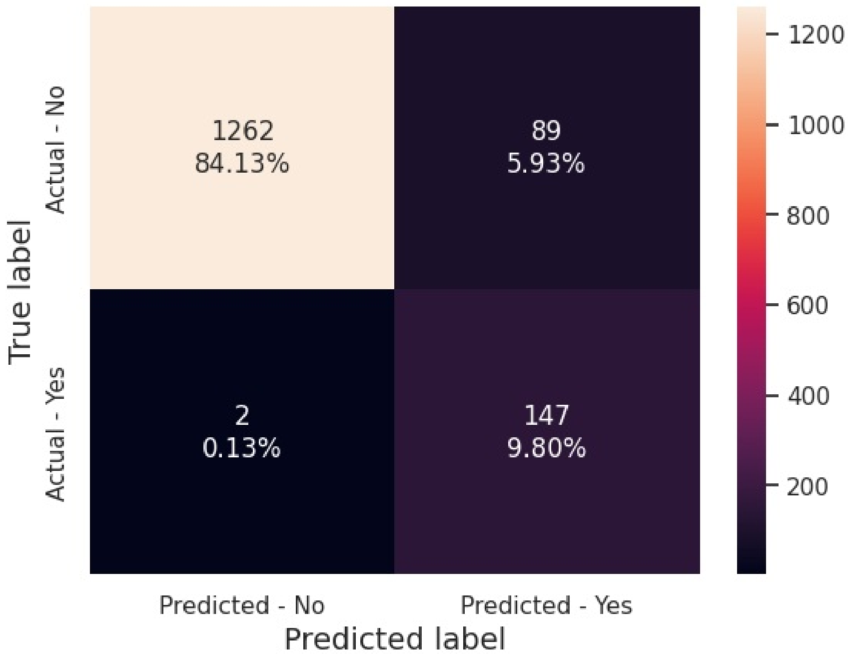
```
comparison_frame = pd.DataFrame({'Model':['Initial decision tree model','Decision treee with hyperparameter tun
                                          'Decision tree with post-pruning'],
                                'Train_Recall':[1, 0.95, 0.99],
                                'Test_Recall':[0.91, 0.91, 0.98]})
comparison_frame
```

Out[181]:

	Model	Train_Recall	Test_Recall
0	Initial decision tree model	1.00	0.91
1	Decision treee with hyperparameter tuning	0.95	0.91
2	Decision tree with post-pruning	0.99	0.98

In [79]:

```
confusion_matrix(best_model2, y_test)
```



Actionable Insights and Business Recommendations

- Customers having an income of 98,000 dollars tend to take personal loans. Bank can offer pre qualifying offer for personal loan for those whose income is 98,000 dollars.
- Customers with online facilities also tend to have personal loans. The online bank portal can be improved and various loan advertisements can be added.
- Prequalifying for a Loan can also attract customers of all kind.
- People with graduate degrees and advanced education tend to take personal loans. The bank can send different loan offers to people with graduate degrees.
- People with big family members also tend to take personal loans,