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 [6]:
```
# Installing the libraries with the specified version.
!pip install numpy==1.25.2 pandas==1.5.3 matplotlib==3.7.1 seaborn==0.13.1 sci
kit-learn==1.2.2 sklearn-pandas==2.2.0 -q --user
```

Note:

1. After running the above cell, kindly restart the notebook kernel (for Jupyter Notebook) or runtime (for Google Colab), write the relevant code for the project from the next cell, and run all cells sequentially from the next cell.
2. On executing the above line of code, you might see a warning regarding package dependencies. This error message can be ignored as the above code ensures that all necessary libraries and their dependencies are maintained to successfully execute the code in this notebook.

In [11]:
```
# Libraries to help with reading and manipulating data
import pandas as pd
import numpy as np
# libaries to help with data visualization
import matplotlib.pyplot as plt
import seaborn as sns
# Removes the limit for the number of displayed columns
pd.set_option("display.max_columns", None)
# Sets the limit for the number of displayed rows
pd.set_option("display.max_rows", 200)
# Library to split data
from sklearn.model_selection import train_test_split
# To build model for prediction
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
# To tune different models
from sklearn.model_selection import GridSearchCV
# To get diferent metric scores
from sklearn.metrics import (
    f1_score,
    accuracy_score,
    recall_score,
    precision_score,
    confusion_matrix,
    make_scorer,
)
# Ignore waringings as they do not affect the code
import warnings
warnings.filterwarnings("ignore")
# Set defaults for seaborn Library
sns.set()
```

Loading the dataset

In [20]:
```
# Define the path to the dataset
file_path = 'Loan_Modelling.csv'
# Open and read the data
df = pd.read_csv(file_path)
# Create a copy of the data to avoid changes to the original data
data = df.copy()
```

In [ ]:
```
# Define the path to the dataset
# Assuming the file is in the current working directory or a subdirectory.
# Adjust './data/Loan_Modelling.csv' if your file is in a 'data' subfolder.
file_path = 'Loan_Modelling.csv'
# or
file_path = './data/Loan_Modelling.csv'  # If data is in a subfolder named 'da
ta'

# Open and read the
```

In [18]:
```
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", force_remount=True).
```

Data Overview

- Observations
- Sanity checks

In [21]:
```
# Viewing the first five rows of the data
data.head()
```

Out[21]:

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

In [22]:
```
# Viewing the last five rows of data
data.tail()
```

Out[22]:

	ID	Age	Experience	Income	ZIPCode	Family	CCAvg	Education	Mortgage	Personal_
4995	4996	29	3	40	92697	1	1.9	3	0	
4996	4997	30	4	15	92037	4	0.4	1	85	
4997	4998	63	39	24	93023	2	0.3	3	0	
4998	4999	65	40	49	90034	3	0.5	2	0	
4999	5000	28	4	83	92612	3	0.8	1	0	

In [23]:
```
# Checking the shape of the data (number of rows and columns)
data.shape
rows, columns = data.shape
print(f"Number of rows: {rows}")
print(f"Number of columns: {columns}")
```

```
Number of rows: 5000
Number of columns: 14
```

In [24]:
```
# Checkng the data types of the columns in the dataset
data.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 [25]:
```
# Checking for missing values
data.isnull().sum()
```

Out[25]:

	0
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

Observations

- There is no missing values in the dataset

In [26]:
```
# print duplicate values
print(data[data.duplicated(keep=False)])
```

```
Empty DataFrame
Columns: [ID, Age, Experience, Income, ZIPCode, Family, CCAvg, Education, Mor
tgage, Personal_Loan, Securities_Account, CD_Account, Online, CreditCard]
Index: []
```

In [27]:
```
# Check for duplicate values in ID column
data['ID'].duplicated().sum()
```

Out[27]: 0

Observations

- There are no duplicate values in ID (Customer ID). Therefore, there are no duplicate values in this column.
- Duplicate values are expected in the other columns

In [40]:
```
data.drop(columns=['ID'],inplace=True)
```

In [28]:
```
# Checking the statistical summary of the data
data.describe().T
```

Out[28]:

	count	mean	std	min	25%	50%	75%	
ID	5000.0	2500.500000	1443.520003	1.0	1250.75	2500.5	3750.25	50
Age	5000.0	45.338400	11.463166	23.0	35.00	45.0	55.00	
Experience	5000.0	20.104600	11.467954	-3.0	10.00	20.0	30.00	
Income	5000.0	73.774200	46.033729	8.0	39.00	64.0	98.00	2
ZIPCode	5000.0	93169.257000	1759.455086	90005.0	91911.00	93437.0	94608.00	966
Family	5000.0	2.396400	1.147663	1.0	1.00	2.0	3.00	
CCAvg	5000.0	1.937938	1.747659	0.0	0.70	1.5	2.50	
Education	5000.0	1.881000	0.839869	1.0	1.00	2.0	3.00	
Mortgage	5000.0	56.498800	101.713802	0.0	0.00	0.0	101.00	6
Personal_Loan	5000.0	0.096000	0.294621	0.0	0.00	0.0	0.00	
Securities_Account	5000.0	0.104400	0.305809	0.0	0.00	0.0	0.00	
CD_Account	5000.0	0.060400	0.238250	0.0	0.00	0.0	0.00	
Online	5000.0	0.596800	0.490589	0.0	0.00	1.0	1.00	
CreditCard	5000.0	0.294000	0.455637	0.0	0.00	0.0	1.00	

Observations

- Age is evenly distributed, ranging from 23 to 67.
- Experience is evenly distributed, ranging from -3 to 43.
- Income is right-skewed with large values pulling and raising the mean
- Zipcode will be treated as a categorical variable as it will not have meaningful numerical relationships
- Family is normally distributed, ranging from 1 to 4.
- CCAvg is slightly right-skewed, ranging from 0 to 10.
- Education is slightly left-skewed, ranging from 1 to 3
- Mortgage is heavily right-skewed, ranging from 0 to 635
- *Personal_Loan is a binary value, with 9.6% of customers having a personal loan.
- Securities_Account is a binary value, with 10% of customers having a securities account.
- CD_Account is a binary value, with 6% of customers having a CD Account.
- Online is a binary value, with 59% of customers banking online.
- CreditCard is a binary value, with 29% of users having a credit card.

Exploratory Data Analysis.

- EDA is an important part of any project involving data.
- It is important to investigate and understand the data better before building a model with it.
- A few questions have been mentioned below which will help you approach the analysis in the right manner and generate insights from the data.
- A thorough analysis of the data, in addition to the questions mentioned below, should be done.

Questions:

1. What is the distribution of mortgage attribute? Are there any noticeable patterns or outliers in the distribution?

- Heavily right-skewed with a large number of outliers above the 3rd quartile

1. How many customers have credit cards?

- 3,500

1. What are the attributes that have a strong correlation with the target attribute (personal loan)?

- customers with education level 3 are about three times more likely to have a personal loan than those with level 1.

1. How does a customer's interest in purchasing a loan vary with their age?

- Most are between 35 and 55

1. How does a customer's interest in purchasing a loan vary with their education?

- Customers with education level 3 are about three times more likely to have a personal loan than those with level 1.

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)

Model Building

In [29]:
```
# Find the 25th percentile and 75th percentile.
Q1 = data.quantile(0.25)
Q3 = data.quantile(0.75)
 # Inter Quantile Range (75th percentile - 25th percentile)
IQR = Q3 - Q1
# Finding lower and upper bounds for all values. All values outside these boun
ds are outliers
lower = (
 Q1 - 1.5 * IQR
 )
upper = Q3 + 1.5 * IQR
```

In [30]:
```
# determine which values are below the lower bound
 # and above the upper bound and convert this into a percentage
(
    (data.select_dtypes(include=["float64", "int64"]) < lower)
    | (data.select_dtypes(include=["float64", "int64"]) > upper)
 ).sum() / len(data) * 100
```

Out[30]:

	0
ID	0.00
Age	0.00
Experience	0.00
Income	1.92
ZIPCode	0.00
Family	0.00
CCAvg	6.48
Education	0.00
Mortgage	5.82
Personal_Loan	9.60
Securities_Account	10.44
CD_Account	6.04
Online	0.00
CreditCard	0.00

dtype: float64

- Age and experience are highly correlated, so we can drop experience.
- Income, CCAvg, and Mortgage have a significant number of values above the 3rd quartile considered outliers. However, these values are legitimate and do not require outlier treatment

Model Evaluation Criterion

Data preperation for Modeling The heatmap shows that Age and Experience are very closely related (0.99), so Experience can be dropped

In [31]:
```
# Dropping Experience as it is closely correlated with Age
# Dropping Personal_Loan as it is the target variable we are predicting
X = data.drop(["Personal_Loan", "Experience"], axis=1)
Y = data["Personal_Loan"]
# One hot encoding categorical variables
X = pd.get_dummies(X, columns=["ZIPCode", "Education"], drop_first=True)
# Splitting data in train and test sets
X_train, X_test, y_train, y_test = train_test_split(
    X, Y, test_size=0.30, random_state=1
)
# Display data ab out the test and training sets
print("Shape of Training set : ", X_train.shape)
print("Shape of test set : ", X_test.shape)
print("Percentage of classes in training set:")
print(y_train.value_counts(normalize=True))
print("Percentage of classes in test set:")
print(y_test.value_counts(normalize=True))
```

```
Shape of Training set :  (3500, 478)
Shape of test set :  (1500, 478)
Percentage of classes in training set:
Personal_Loan
0    0.905429
1    0.094571
Name: proportion, dtype: float64
Percentage of classes in test set:
Personal_Loan
0    0.900667
1    0.099333
Name: proportion, dtype: float64
```

- The class distribution is very similar between the training and test sets, which is good. It indicates that the split was likely done with stratification, preserving the overall class distribution in both sets.
- The dataset is imbalanced, with the majority class (0) representing about 90% of the data and the minority class (1) only about 10%. This imbalance can affect the performance of the decision tree, potentially biasing it towards the majority class.

Model Building

In [32]:
```
# Define a function to compute different metrics for evaluating the performanc
e of a classification model built using sklearn
def model_performance_classification_sklearn(model, predictors, target):
    """
    Compute various metrics to evaluate classification model performance.
    Parameters:
    model (sklearn classifier): The trained classification model
    predictors (array-like): Independent variables used for prediction
    target (array-like): True labels for the target variable
    Returns:
    pandas.DataFrame: A single-row dataframe containing Accuracy, Recall, Prec
ision, and F1 score
    """
    # Make predictions using the independent variables
    pred = model.predict(predictors)
    # Compute Accuracy: the proportion of correct predictions (both true posit
ives and true negatives) among the total number
    acc = accuracy_score(target, pred)
    # Compute Recall: the proportion of actual positive cases that were correc
tly identified
    recall = recall_score(target, pred)
    # Compute Precision: the proportion of predicted positive cases that are a
ctually positive
    precision = precision_score(target, pred)
    # Compute F1-score: the harmonic mean of precision and recall, providing a
single score that balances both metrics
    f1 = f1_score(target, pred)
    # Create a dataframe to store and return all computed metrics
    df_perf = pd.DataFrame(
        {"Accuracy": acc, "Recall": recall, "Precision": precision, "F1": f
1,},
        index=[0],
    )
    # Return the  dataframe of performance metrics
    return df_perf
```

In [33]:
```
# Define a function to create and display a confusion matrix
def confusion_matrix_sklearn(model, predictors, target):
    """
    Plot the confusion matrix with percentages for a classification model.
    Parameters:
    model: classifier object
    predictors: independent variables (features)
    target: dependent variable (true labels)
    """
    # Generate predictions using the model
    y_pred = model.predict(predictors)
    # Compute the confusion matrix
    cm = confusion_matrix(target, y_pred)
    # Create labels for the confusion matrix
    # Each cell will display count and percentage
    labels = np.asarray(
        [
            ["{0:0.0f}".format(item) + "\n{0:.2%}".format(item / cm.flatten().
sum())]
            for item in cm.flatten()
        ]
         ).reshape(2, 2)
    # Set up the matplotlib figure
    plt.figure(figsize=(6, 4))
    # Create a heatmap of the confusion matrix
    # annot=labels adds the custom labels to each cell
    sns.heatmap(cm, annot=labels, fmt="") # cm is now defined within the funct
ion's scope
    # Set labels for the axes
    plt.ylabel("True label")
    plt.xlabel("Predicted label")
    # Return the confusion matrix for potential later use if needed
    return cm # Returning cm to make it accessible outside the function if nee
ded
```

Build Decision Tree Model

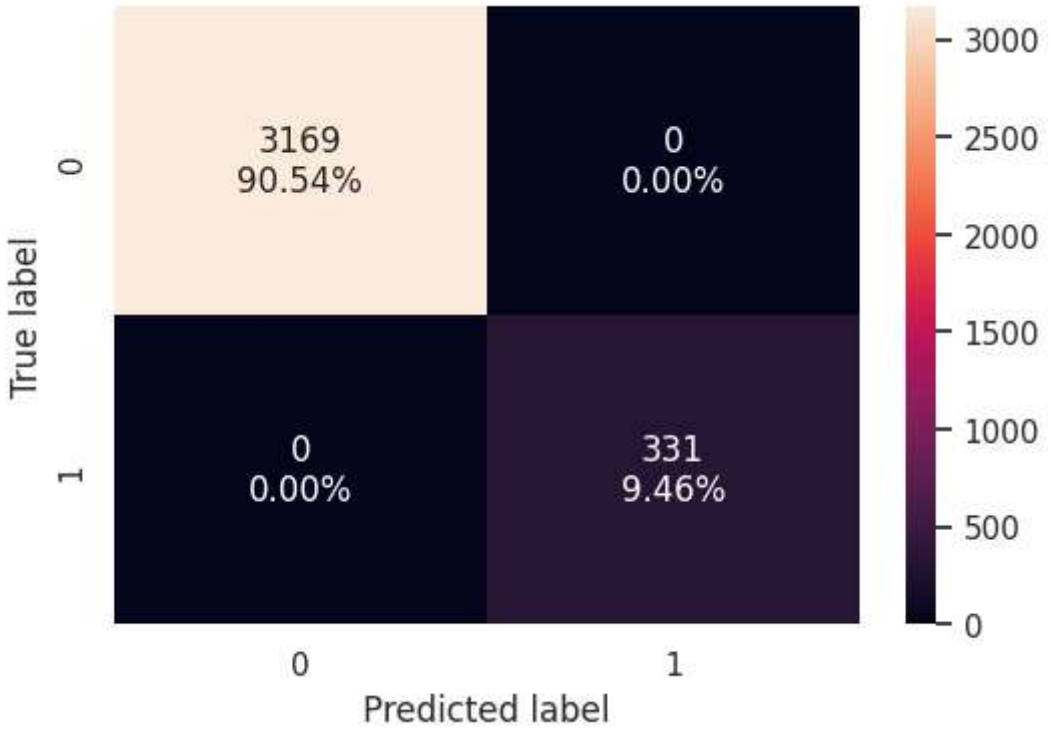
In [34]:
```
# Building a decision tree model
model = DecisionTreeClassifier(criterion="gini", random_state=1)
model.fit(X_train, y_train)
```

Out[34]:
```
DecisionTreeClassifier(random_state=1)
```

Checking model performance on training data

In [35]:
```
confusion_matrix_sklearn(model, X_train, y_train)
```

Out[35]:
```
array([[3169,    0],
       [   0,  331]])
```



In [36]:
```
# Compute and display the performance metrics for the training set
decision_tree_perf_train = model_performance_classification_sklearn(
 model, X_train, y_train
 )
decision_tree_perf_train
```

Out[36]:

	Accuracy	Recall	Precision	F1
0	1.0	1.0	1.0	1.0

- The classification model has achieved perfect performance metrics across the board, with an Accuracy, Recall, Precision, and F1-score all equal to 1.0. This indicates that the model correctly classified all instances, with no false positives or false negatives.
- While this is an ideal result, it is important to consider whether this performance is realistic or if it might indicate overfitting, especially if the model was evaluated on the training data or a non-representative test set

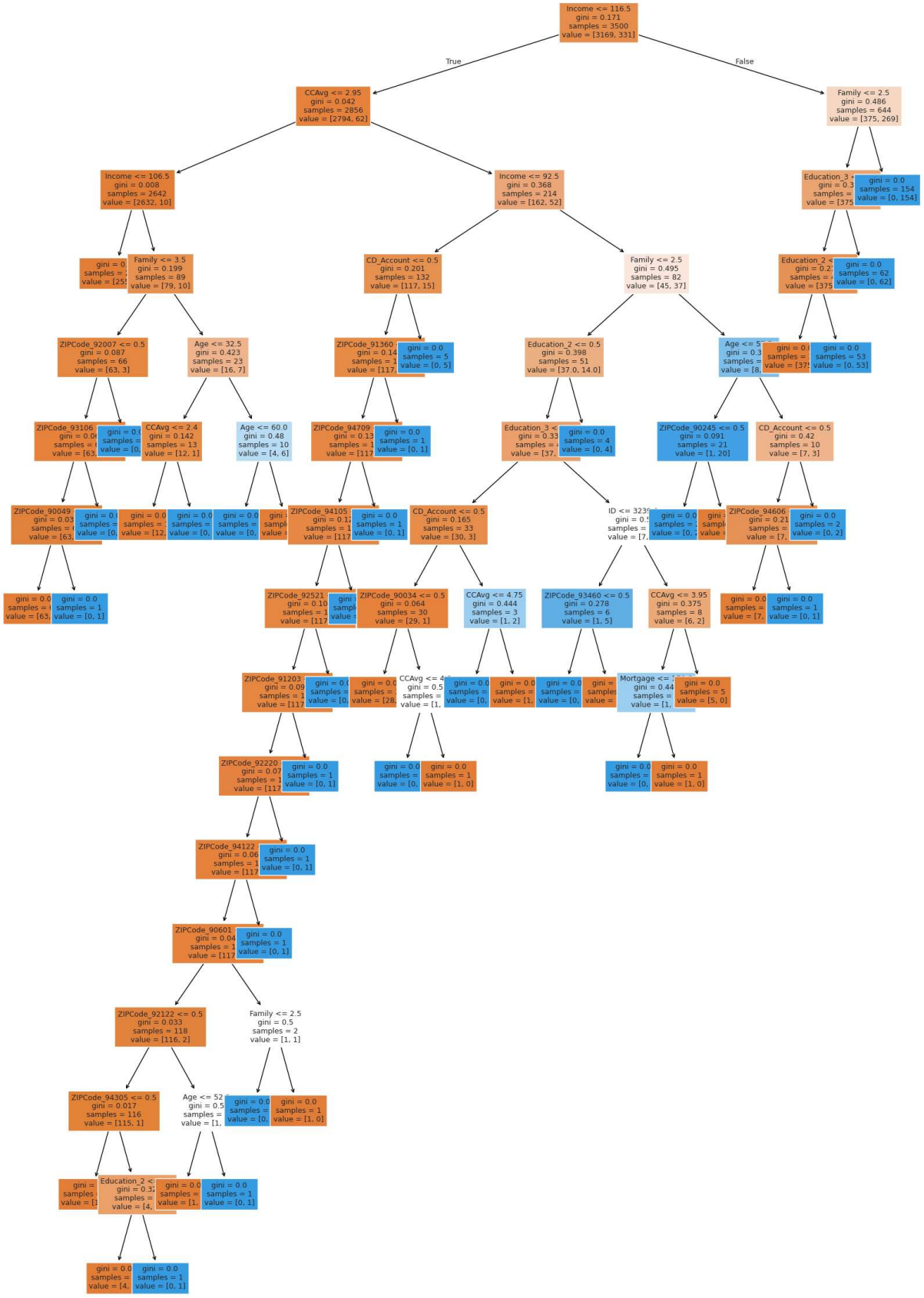
Visualizing the Decision Tree for the training data

In [37]:
```
# Extract and print feature names from the training dataset
feature_names = list(X_train.columns)
print(feature_names)
```

```
['ID', 'Age', 'Income', 'Family', 'CCAvg', 'Mortgage', 'Securities_Account',
'CD_Account', 'Online', 'CreditCard', 'ZIPCode_90007', 'ZIPCode_90009', 'ZIPC
ode_90011', 'ZIPCode_90016', 'ZIPCode_90018', 'ZIPCode_90019', 'ZIPCode_9002
4', 'ZIPCode_90025', 'ZIPCode_90027', 'ZIPCode_90028', 'ZIPCode_90029', 'ZIPC
ode_90032', 'ZIPCode_90033', 'ZIPCode_90034', 'ZIPCode_90035', 'ZIPCode_9003
6', 'ZIPCode_90037', 'ZIPCode_90041', 'ZIPCode_90044', 'ZIPCode_90045', 'ZIPC
ode_90048', 'ZIPCode_90049', 'ZIPCode_90057', 'ZIPCode_90058', 'ZIPCode_9005
9', 'ZIPCode_90064', 'ZIPCode_90065', 'ZIPCode_90066', 'ZIPCode_90068', 'ZIPC
ode_90071', 'ZIPCode_90073', 'ZIPCode_90086', 'ZIPCode_90089', 'ZIPCode_9009
5', 'ZIPCode_90210', 'ZIPCode_90212', 'ZIPCode_90230', 'ZIPCode_90232', 'ZIPC
ode_90245', 'ZIPCode_90250', 'ZIPCode_90254', 'ZIPCode_90266', 'ZIPCode_9027
2', 'ZIPCode_90274', 'ZIPCode_90275', 'ZIPCode_90277', 'ZIPCode_90280', 'ZIPC
ode_90291', 'ZIPCode_90304', 'ZIPCode_90401', 'ZIPCode_90404', 'ZIPCode_9040
5', 'ZIPCode_90502', 'ZIPCode_90503', 'ZIPCode_90504', 'ZIPCode_90505', 'ZIPC
ode_90509', 'ZIPCode_90601', 'ZIPCode_90623', 'ZIPCode_90630', 'ZIPCode_9063
8', 'ZIPCode_90639', 'ZIPCode_90640', 'ZIPCode_90650', 'ZIPCode_90717', 'ZIPC
ode_90720', 'ZIPCode_90740', 'ZIPCode_90745', 'ZIPCode_90747', 'ZIPCode_9075
5', 'ZIPCode_90813', 'ZIPCode_90840', 'ZIPCode_91006', 'ZIPCode_91007', 'ZIPC
ode_91016', 'ZIPCode_91024', 'ZIPCode_91030', 'ZIPCode_91040', 'ZIPCode_9110
1', 'ZIPCode_91103', 'ZIPCode_91105', 'ZIPCode_91107', 'ZIPCode_91109', 'ZIPC
ode_91116', 'ZIPCode_91125', 'ZIPCode_91129', 'ZIPCode_91203', 'ZIPCode_9120
7', 'ZIPCode_91301', 'ZIPCode_91302', 'ZIPCode_91304', 'ZIPCode_91311', 'ZIPC
ode_91320', 'ZIPCode_91326', 'ZIPCode_91330', 'ZIPCode_91335', 'ZIPCode_9134
2', 'ZIPCode_91343', 'ZIPCode_91345', 'ZIPCode_91355', 'ZIPCode_91360', 'ZIPC
ode_91361', 'ZIPCode_91365', 'ZIPCode_91367', 'ZIPCode_91380', 'ZIPCode_9140
1', 'ZIPCode_91423', 'ZIPCode_91604', 'ZIPCode_91605', 'ZIPCode_91614', 'ZIPC
ode_91706', 'ZIPCode_91709', 'ZIPCode_91710', 'ZIPCode_91711', 'ZIPCode_9173
0', 'ZIPCode_91741', 'ZIPCode_91745', 'ZIPCode_91754', 'ZIPCode_91763', 'ZIPC
ode_91765', 'ZIPCode_91768', 'ZIPCode_91770', 'ZIPCode_91773', 'ZIPCode_9177
5', 'ZIPCode_91784', 'ZIPCode_91791', 'ZIPCode_91801', 'ZIPCode_91902', 'ZIPC
ode_91910', 'ZIPCode_91911', 'ZIPCode_91941', 'ZIPCode_91942', 'ZIPCode_9195
0', 'ZIPCode_92007', 'ZIPCode_92008', 'ZIPCode_92009', 'ZIPCode_92024', 'ZIPC
ode_92028', 'ZIPCode_92029', 'ZIPCode_92037', 'ZIPCode_92038', 'ZIPCode_9205
4', 'ZIPCode_92056', 'ZIPCode_92064', 'ZIPCode_92068', 'ZIPCode_92069', 'ZIPC
ode_92084', 'ZIPCode_92093', 'ZIPCode_92096', 'ZIPCode_92101', 'ZIPCode_9210
3', 'ZIPCode_92104', 'ZIPCode_92106', 'ZIPCode_92109', 'ZIPCode_92110', 'ZIPC
ode_92115', 'ZIPCode_92116', 'ZIPCode_92120', 'ZIPCode_92121', 'ZIPCode_9212
2', 'ZIPCode_92123', 'ZIPCode_92124', 'ZIPCode_92126', 'ZIPCode_92129', 'ZIPC
ode_92130', 'ZIPCode_92131', 'ZIPCode_92152', 'ZIPCode_92154', 'ZIPCode_9216
1', 'ZIPCode_92173', 'ZIPCode_92177', 'ZIPCode_92182', 'ZIPCode_92192', 'ZIPC
ode_92220', 'ZIPCode_92251', 'ZIPCode_92325', 'ZIPCode_92333', 'ZIPCode_9234
6', 'ZIPCode_92350', 'ZIPCode_92354', 'ZIPCode_92373', 'ZIPCode_92374', 'ZIPC
ode_92399', 'ZIPCode_92407', 'ZIPCode_92507', 'ZIPCode_92518', 'ZIPCode_9252
1', 'ZIPCode_92606', 'ZIPCode_92612', 'ZIPCode_92614', 'ZIPCode_92624', 'ZIPC
ode_92626', 'ZIPCode_92630', 'ZIPCode_92634', 'ZIPCode_92646', 'ZIPCode_9264
7', 'ZIPCode_92648', 'ZIPCode_92653', 'ZIPCode_92660', 'ZIPCode_92661', 'ZIPC
ode_92672', 'ZIPCode_92673', 'ZIPCode_92675', 'ZIPCode_92677', 'ZIPCode_9269
1', 'ZIPCode_92692', 'ZIPCode_92694', 'ZIPCode_92697', 'ZIPCode_92703', 'ZIPC
ode_92704', 'ZIPCode_92705', 'ZIPCode_92709', 'ZIPCode_92717', 'ZIPCode_9273
5', 'ZIPCode_92780', 'ZIPCode_92806', 'ZIPCode_92807', 'ZIPCode_92821', 'ZIPC
ode_92831', 'ZIPCode_92833', 'ZIPCode_92834', 'ZIPCode_92835', 'ZIPCode_9284
3', 'ZIPCode_92866', 'ZIPCode_92867', 'ZIPCode_92868', 'ZIPCode_92870', 'ZIPC
ode_92886', 'ZIPCode_93003', 'ZIPCode_93009', 'ZIPCode_93010', 'ZIPCode_9301
4', 'ZIPCode_93022', 'ZIPCode_93023', 'ZIPCode_93033', 'ZIPCode_93063', 'ZIPC
ode_93065', 'ZIPCode_93077', 'ZIPCode_93101', 'ZIPCode_93105', 'ZIPCode_9310
6', 'ZIPCode_93107', 'ZIPCode_93108', 'ZIPCode_93109', 'ZIPCode_93111', 'ZIPC
ode_93117', 'ZIPCode_93118', 'ZIPCode_93302', 'ZIPCode_93305', 'ZIPCode_9331
```

```
1', 'ZIPCode_93401', 'ZIPCode_93403', 'ZIPCode_93407', 'ZIPCode_93437', 'ZIPC
ode_93460', 'ZIPCode_93524', 'ZIPCode_93555', 'ZIPCode_93561', 'ZIPCode_9361
1', 'ZIPCode_93657', 'ZIPCode_93711', 'ZIPCode_93720', 'ZIPCode_93727', 'ZIPC
ode_93907', 'ZIPCode_93933', 'ZIPCode_93940', 'ZIPCode_93943', 'ZIPCode_9395
0', 'ZIPCode_93955', 'ZIPCode_94002', 'ZIPCode_94005', 'ZIPCode_94010', 'ZIPC
ode_94015', 'ZIPCode_94019', 'ZIPCode_94022', 'ZIPCode_94024', 'ZIPCode_9402
5', 'ZIPCode_94028', 'ZIPCode_94035', 'ZIPCode_94040', 'ZIPCode_94043', 'ZIPC
ode_94061', 'ZIPCode_94063', 'ZIPCode_94065', 'ZIPCode_94066', 'ZIPCode_9408
0', 'ZIPCode_94085', 'ZIPCode_94086', 'ZIPCode_94087', 'ZIPCode_94102', 'ZIPC
ode_94104', 'ZIPCode_94105', 'ZIPCode_94107', 'ZIPCode_94108', 'ZIPCode_9410
9', 'ZIPCode_94110', 'ZIPCode_94111', 'ZIPCode_94112', 'ZIPCode_94114', 'ZIPC
ode_94115', 'ZIPCode_94116', 'ZIPCode_94117', 'ZIPCode_94118', 'ZIPCode_9412
2', 'ZIPCode_94123', 'ZIPCode_94124', 'ZIPCode_94126', 'ZIPCode_94131', 'ZIPC
ode_94132', 'ZIPCode_94143', 'ZIPCode_94234', 'ZIPCode_94301', 'ZIPCode_9430
2', 'ZIPCode_94303', 'ZIPCode_94304', 'ZIPCode_94305', 'ZIPCode_94306', 'ZIPC
ode_94309', 'ZIPCode_94402', 'ZIPCode_94404', 'ZIPCode_94501', 'ZIPCode_9450
7', 'ZIPCode_94509', 'ZIPCode_94521', 'ZIPCode_94523', 'ZIPCode_94526', 'ZIPC
ode_94534', 'ZIPCode_94536', 'ZIPCode_94538', 'ZIPCode_94539', 'ZIPCode_9454
2', 'ZIPCode_94545', 'ZIPCode_94546', 'ZIPCode_94550', 'ZIPCode_94551', 'ZIPC
ode_94553', 'ZIPCode_94555', 'ZIPCode_94558', 'ZIPCode_94566', 'ZIPCode_9457
1', 'ZIPCode_94575', 'ZIPCode_94577', 'ZIPCode_94583', 'ZIPCode_94588', 'ZIPC
ode_94590', 'ZIPCode_94591', 'ZIPCode_94596', 'ZIPCode_94598', 'ZIPCode_9460
4', 'ZIPCode_94606', 'ZIPCode_94607', 'ZIPCode_94608', 'ZIPCode_94609', 'ZIPC
ode_94610', 'ZIPCode_94611', 'ZIPCode_94612', 'ZIPCode_94618', 'ZIPCode_9470
1', 'ZIPCode_94703', 'ZIPCode_94704', 'ZIPCode_94705', 'ZIPCode_94706', 'ZIPC
ode_94707', 'ZIPCode_94708', 'ZIPCode_94709', 'ZIPCode_94710', 'ZIPCode_9472
0', 'ZIPCode_94801', 'ZIPCode_94803', 'ZIPCode_94806', 'ZIPCode_94901', 'ZIPC
ode_94904', 'ZIPCode_94920', 'ZIPCode_94923', 'ZIPCode_94928', 'ZIPCode_9493
9', 'ZIPCode_94949', 'ZIPCode_94960', 'ZIPCode_94965', 'ZIPCode_94970', 'ZIPC
ode_94998', 'ZIPCode_95003', 'ZIPCode_95005', 'ZIPCode_95006', 'ZIPCode_9500
8', 'ZIPCode_95010', 'ZIPCode_95014', 'ZIPCode_95020', 'ZIPCode_95023', 'ZIPC
ode_95032', 'ZIPCode_95035', 'ZIPCode_95037', 'ZIPCode_95039', 'ZIPCode_9504
5', 'ZIPCode_95051', 'ZIPCode_95053', 'ZIPCode_95054', 'ZIPCode_95060', 'ZIPC
ode_95064', 'ZIPCode_95070', 'ZIPCode_95112', 'ZIPCode_95120', 'ZIPCode_9512
3', 'ZIPCode_95125', 'ZIPCode_95126', 'ZIPCode_95131', 'ZIPCode_95133', 'ZIPC
ode_95134', 'ZIPCode_95135', 'ZIPCode_95136', 'ZIPCode_95138', 'ZIPCode_9519
2', 'ZIPCode_95193', 'ZIPCode_95207', 'ZIPCode_95211', 'ZIPCode_95307', 'ZIPC
ode_95348', 'ZIPCode_95351', 'ZIPCode_95354', 'ZIPCode_95370', 'ZIPCode_9540
3', 'ZIPCode_95405', 'ZIPCode_95422', 'ZIPCode_95449', 'ZIPCode_95482', 'ZIPC
ode_95503', 'ZIPCode_95518', 'ZIPCode_95521', 'ZIPCode_95605', 'ZIPCode_9561
6', 'ZIPCode_95617', 'ZIPCode_95621', 'ZIPCode_95630', 'ZIPCode_95670', 'ZIPC
ode_95678', 'ZIPCode_95741', 'ZIPCode_95747', 'ZIPCode_95758', 'ZIPCode_9576
2', 'ZIPCode_95812', 'ZIPCode_95814', 'ZIPCode_95816', 'ZIPCode_95817', 'ZIPC
ode_95818', 'ZIPCode_95819', 'ZIPCode_95820', 'ZIPCode_95821', 'ZIPCode_9582
2', 'ZIPCode_95825', 'ZIPCode_95827', 'ZIPCode_95828', 'ZIPCode_95831', 'ZIPC
ode_95833', 'ZIPCode_95841', 'ZIPCode_95842', 'ZIPCode_95929', 'ZIPCode_9597
3', 'ZIPCode_96001', 'ZIPCode_96003', 'ZIPCode_96008', 'ZIPCode_96064', 'ZIPC
ode_96091', 'ZIPCode_96094', 'ZIPCode_96145', 'ZIPCode_96150', 'ZIPCode_9665
1', 'Education_2', 'Education_3']
```

In [38]:
```
# Display the Decision Tree for this training data
plt.figure(figsize=(20, 30))
out = tree.plot_tree(
    model,
    feature_names=feature_names,
    filled=True,
    fontsize=9,
    node_ids=False,
    class_names=None,
 )
 # Add arrows to the decision tree split if they are missing
for o in out:
    arrow = o.arrow_patch
    if arrow is not None:
        arrow.set_edgecolor("black")
        arrow.set_linewidth(1)
plt.show()
```


Income <= 116.5
gini = 0.171
samples = 3500
value = [3169, 331]
True
False
CCAvg <= 2.95
gini = 0.042
samples = 2856
value = [2794, 62]
Family <= 2.5
gini = 0.486
samples = 644
value = [375, 269]
Income <= 106.5
gini = 0.008
samples = 2642
value = [2632, 10]
Income <= 92.5
gini = 0.368
samples = 214
value = [162, 52]
gini = 0.0
samples = 154
value = [0, 154]
Family <= 3.5
gini = 0.199
samples = 89
value = [79, 10]
CD_Account <= 0.5
gini = 0.201
samples = 132
value = [117, 15]
Family <= 2.5
gini = 0.495
samples = 82
value = [45, 37]
gini = 0.0
samples = 62
value = [0, 62]
ZIPCode_92007 <= 0.5
gini = 0.087
samples = 66
value = [63, 3]
Age <= 32.5
gini = 0.423
samples = 23
value = [16, 7]
ZIPCode_91360
gini = 0.14
gini = 0.0
samples = 5
value = [0, 5]
Education_2 <= 0.5
gini = 0.398
samples = 51
value = [37.0, 14.0]
gini = 0.0
samples = 53
value = [0, 53]
CCAvg <= 2.4
gini = 0.142
samples = 13
value = [12, 1]
Age <= 60.0
gini = 0.48
samples = 10
value = [4, 6]
gini = 0.0
samples = 1
value = [0, 1]
Education_3 <
gini = 0.33
gini = 0.0
samples = 4
value = [0, 4]
ZIPCode_90245 <= 0.5
gini = 0.091
samples = 21
value = [1, 20]
CD_Account <= 0.5
gini = 0.42
samples = 10
value = [7, 3]
ZIPCode_94105
gini = 0.1
gini = 0.0
samples = 1
value = [0, 1]
CD_Account <= 0.5
gini = 0.165
samples = 33
value = [30, 3]
gini = 0.0
samples = 2
value = [0, 2]
ZIPCode_92521
gini = 0.1
ZIPCode_90034 <= 0.5
gini = 0.064
samples = 30
value = [29, 1]
CCAvg <= 4.75
gini = 0.444
samples = 3
value = [1, 2]
ZIPCode_93460 <= 0.5
gini = 0.278
samples = 6
value = [1, 5]
CCAvg <= 3.95
gini = 0.375
samples = 8
value = [6, 2]
gini = 0.0
samples = 1
value = [0, 1]
ZIPCode_91203
gini = 0.09
gini = 0.0
samples = 1
value = [0, 1]
ZIPCode_92220
gini = 0.07
gini = 0.0
samples = 1
value = [0, 1]
ZIPCode_94122
gini = 0.06
gini = 0.0
samples = 1
value = [0, 1]
ZIPCode_90601
gini = 0.04
gini = 0.0
samples = 1
value = [0, 1]
ZIPCode_92122 <= 0.5
gini = 0.033
samples = 118
value = [116, 2]
Family <= 2.5
gini = 0.5
samples = 2
value = [1, 1]
ZIPCode_94305 <= 0.5
gini = 0.017
samples = 116
value = [115, 1]
gini = 0.0
samples = 1
value = [1, 0]
gini = 0.0
samples = 1
value = [0, 1]

In [ ]:

Model Performance Improvement

In [39]:
```
# Text report showing the rules of a decision tree
print(tree.export_text(model, feature_names=feature_names, show_weights=True))
```

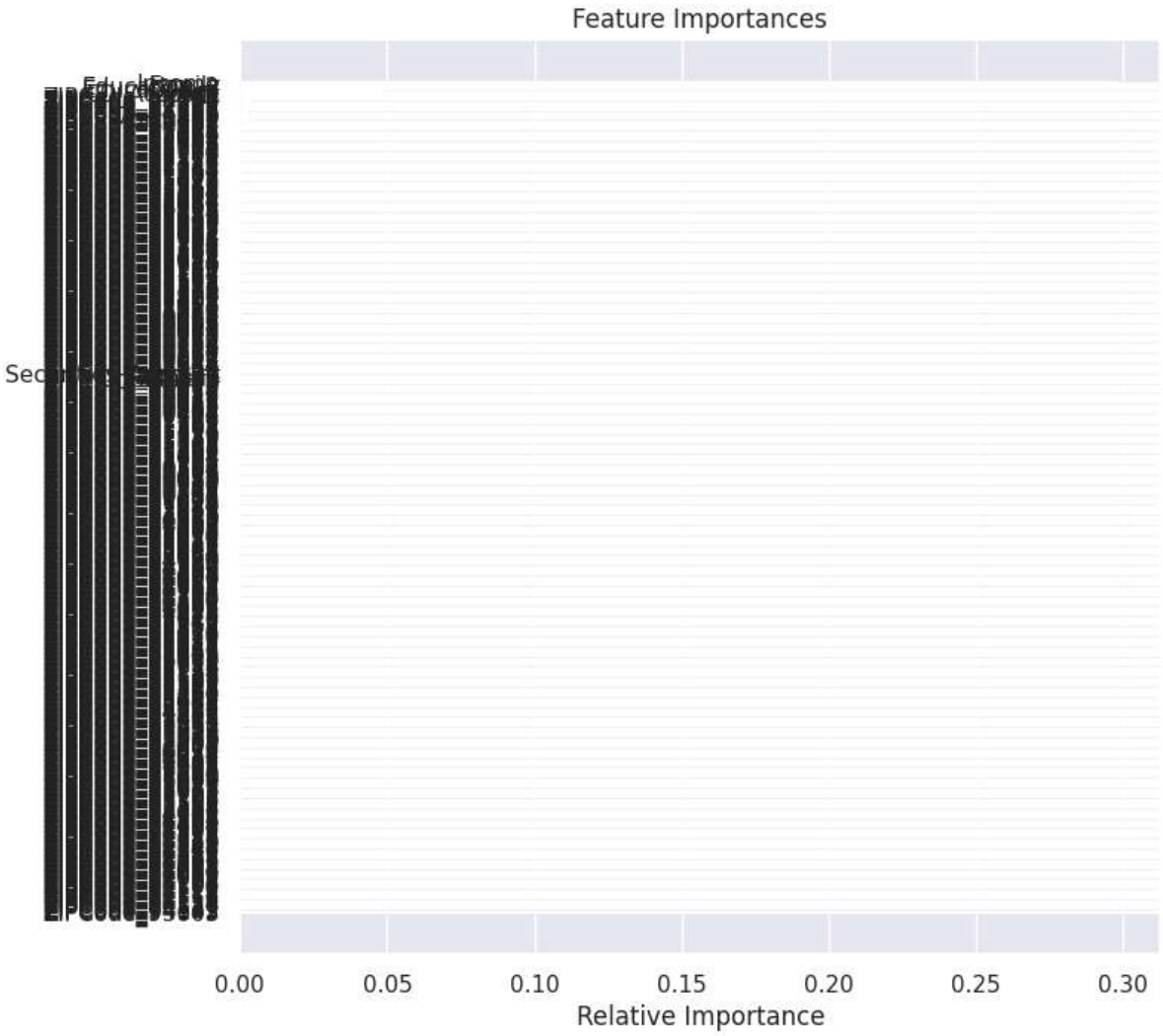
```
|--- Income <= 116.50
|   |--- CCAvg <= 2.95
|   |   |--- Income <= 106.50
|   |   |   |--- weights: [2553.00, 0.00] class: 0
|   |   |--- Income >  106.50
|   |   |   |--- Family <= 3.50
|   |   |   |   |--- ZIPCode_92007 <= 0.50
|   |   |   |   |   |--- ZIPCode_93106 <= 0.50
|   |   |   |   |   |   |--- ZIPCode_90049 <= 0.50
|   |   |   |   |   |   |   |--- weights: [63.00, 0.00] class: 0
|   |   |   |   |   |   |--- ZIPCode_90049 >  0.50
|   |   |   |   |   |   |   |--- weights: [0.00, 1.00] class: 1
|   |   |   |   |   |--- ZIPCode_93106 >  0.50
|   |   |   |   |   |   |--- weights: [0.00, 1.00] class: 1
|   |   |   |   |--- ZIPCode_92007 >  0.50
|   |   |   |   |   |--- weights: [0.00, 1.00] class: 1
|   |   |   |--- Family >  3.50
|   |   |   |   |--- Age <= 32.50
|   |   |   |   |   |--- CCAvg <= 2.40
|   |   |   |   |   |   |--- weights: [12.00, 0.00] class: 0
|   |   |   |   |   |--- CCAvg >  2.40
|   |   |   |   |   |   |--- weights: [0.00, 1.00] class: 1
|   |   |   |   |--- Age >  32.50
|   |   |   |   |   |--- Age <= 60.00
|   |   |   |   |   |   |--- weights: [0.00, 6.00] class: 1
|   |   |   |   |   |--- Age >  60.00
|   |   |   |   |   |   |--- weights: [4.00, 0.00] class: 0
|   |--- CCAvg >  2.95
|   |   |--- Income <= 92.50
|   |   |   |--- CD_Account <= 0.50
|   |   |   |   |--- ZIPCode_91360 <= 0.50
|   |   |   |   |   |--- ZIPCode_94709 <= 0.50
|   |   |   |   |   |   |--- ZIPCode_94105 <= 0.50
|   |   |   |   |   |   |   |--- ZIPCode_92521 <= 0.50
|   |   |   |   |   |   |   |   |--- ZIPCode_91203 <= 0.50
|   |   |   |   |   |   |   |   |   |--- ZIPCode_92220 <= 0.50
|   |   |   |   |   |   |   |   |   |   |--- ZIPCode_94122 <= 0.50
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 5
|   |   |   |   |   |   |   |   |   |   |--- ZIPCode_94122 >  0.50
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [0.00, 1.00] class:
1
|   |   |   |   |   |   |   |   |   |--- ZIPCode_92220 >  0.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [0.00, 1.00] class: 1
|   |   |   |   |   |   |   |   |--- ZIPCode_91203 >  0.50
|   |   |   |   |   |   |   |   |   |--- weights: [0.00, 1.00] class: 1
|   |   |   |   |   |   |   |--- ZIPCode_92521 >  0.50
|   |   |   |   |   |   |   |   |--- weights: [0.00, 1.00] class: 1
|   |   |   |   |   |   |--- ZIPCode_94105 >  0.50
|   |   |   |   |   |   |   |--- weights: [0.00, 1.00] class: 1
|   |   |   |   |   |--- ZIPCode_94709 >  0.50
|   |   |   |   |   |   |--- weights: [0.00, 1.00] class: 1
|   |   |   |   |--- ZIPCode_91360 >  0.50
|   |   |   |   |   |--- weights: [0.00, 1.00] class: 1
|   |   |   |--- CD_Account >  0.50
|   |   |   |   |--- weights: [0.00, 5.00] class: 1
|   |   |--- Income >  92.50
|   |   |   |--- Family <= 2.50
```

```
|   |   |   |   |--- Education_2 <= 0.50
|   |   |   |   |   |--- Education_3 <= 0.50
|   |   |   |   |   |   |--- CD_Account <= 0.50
|   |   |   |   |   |   |   |--- ZIPCode_90034 <= 0.50
|   |   |   |   |   |   |   |   |--- weights: [28.00, 0.00] class: 0
|   |   |   |   |   |   |   |--- ZIPCode_90034 >  0.50
|   |   |   |   |   |   |   |   |--- CCAvg <= 4.80
|   |   |   |   |   |   |   |   |   |--- weights: [0.00, 1.00] class: 1
|   |   |   |   |   |   |   |   |--- CCAvg >  4.80
|   |   |   |   |   |   |   |   |   |--- weights: [1.00, 0.00] class: 0
|   |   |   |   |   |   |--- CD_Account >  0.50
|   |   |   |   |   |   |   |--- CCAvg <= 4.75
|   |   |   |   |   |   |   |   |--- weights: [0.00, 2.00] class: 1
|   |   |   |   |   |   |   |--- CCAvg >  4.75
|   |   |   |   |   |   |   |   |--- weights: [1.00, 0.00] class: 0
|   |   |   |   |   |--- Education_3 >  0.50
|   |   |   |   |   |   |--- ID <= 3239.00
|   |   |   |   |   |   |   |--- ZIPCode_93460 <= 0.50
|   |   |   |   |   |   |   |   |--- weights: [0.00, 5.00] class: 1
|   |   |   |   |   |   |   |--- ZIPCode_93460 >  0.50
|   |   |   |   |   |   |   |   |--- weights: [1.00, 0.00] class: 0
|   |   |   |   |   |   |--- ID >  3239.00
|   |   |   |   |   |   |   |--- CCAvg <= 3.95
|   |   |   |   |   |   |   |   |--- Mortgage <= 170.00
|   |   |   |   |   |   |   |   |   |--- weights: [0.00, 2.00] class: 1
|   |   |   |   |   |   |   |   |--- Mortgage >  170.00
|   |   |   |   |   |   |   |   |   |--- weights: [1.00, 0.00] class: 0
|   |   |   |   |   |   |   |--- CCAvg >  3.95
|   |   |   |   |   |   |   |   |--- weights: [5.00, 0.00] class: 0
|   |   |   |   |   |--- Education_2 >  0.50
|   |   |   |   |   |   |--- weights: [0.00, 4.00] class: 1
|   |   |   |   |--- Family >  2.50
|   |   |   |   |   |--- Age <= 57.50
|   |   |   |   |   |   |--- ZIPCode_90245 <= 0.50
|   |   |   |   |   |   |   |--- weights: [0.00, 20.00] class: 1
|   |   |   |   |   |   |--- ZIPCode_90245 >  0.50
|   |   |   |   |   |   |   |--- weights: [1.00, 0.00] class: 0
|   |   |   |   |   |--- Age >  57.50
|   |   |   |   |   |   |--- CD_Account <= 0.50
|   |   |   |   |   |   |   |--- ZIPCode_94606 <= 0.50
|   |   |   |   |   |   |   |   |--- weights: [7.00, 0.00] class: 0
|   |   |   |   |   |   |   |--- ZIPCode_94606 >  0.50
|   |   |   |   |   |   |   |   |--- weights: [0.00, 1.00] class: 1
|   |   |   |   |   |   |--- CD_Account >  0.50
|   |   |   |   |   |   |   |--- weights: [0.00, 2.00] class: 1
|   |--- Income >  116.50
|   |   |--- Family <= 2.50
|   |   |   |--- Education_3 <= 0.50
|   |   |   |   |--- Education_2 <= 0.50
|   |   |   |   |   |--- weights: [375.00, 0.00] class: 0
|   |   |   |   |--- Education_2 >  0.50
|   |   |   |   |   |--- weights: [0.00, 53.00] class: 1
|   |   |   |--- Education_3 >  0.50
|   |   |   |   |--- weights: [0.00, 62.00] class: 1
|   |   |--- Family >  2.50
|   |   |   |--- weights: [0.00, 154.00] class: 1
```

In [40]:
```
# Create and print a DataFrame of feature importances
 # 1. Extract feature importances from the model
 # 2. Create a DataFrame with importances and feature names computed as the  Gini importance
 # 3. Display the result
print(
 pd.DataFrame(
 model.feature_importances_, columns=["Imp"], index=X_train.columns
 ).sort_values(by="Imp", ascending=False)
 )
```

```
                    Imp
Income         0.297816
Family         0.248530
Education_2    0.165238
Education_3    0.144207
CCAvg          0.047550
...                 ...
ZIPCode_92110  0.000000
ZIPCode_92109  0.000000
ZIPCode_92106  0.000000
ZIPCode_92104  0.000000
ZIPCode_93009  0.000000

[478 rows x 1 columns]
```

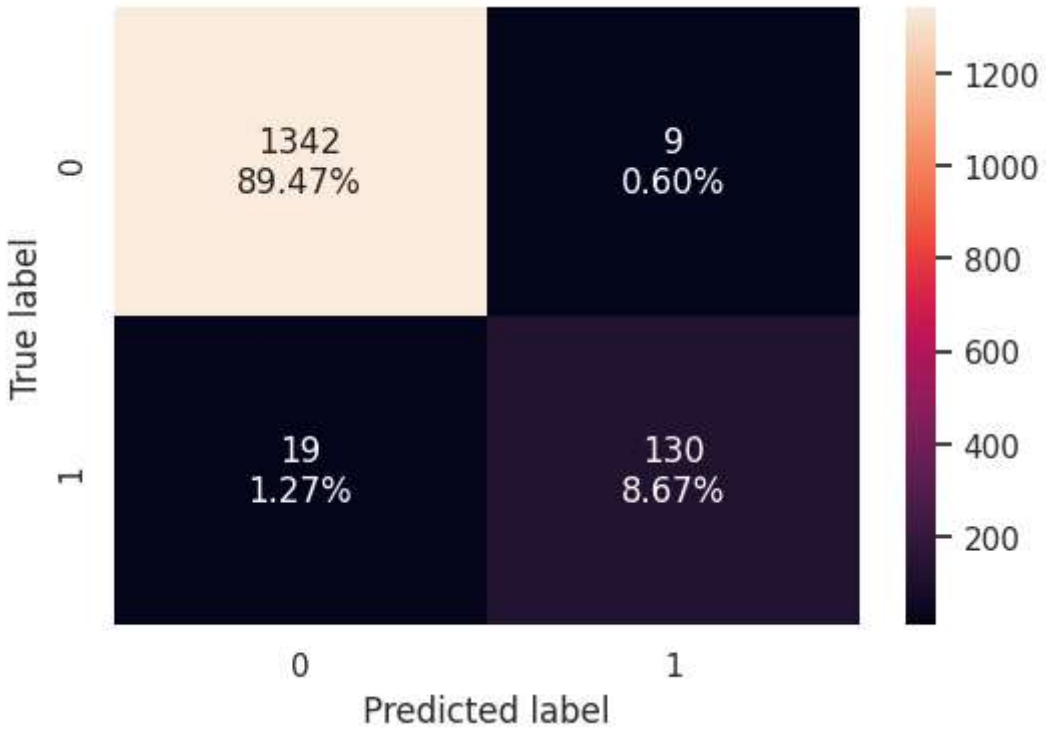
In [41]:
```
# Visualize feature importances of the model
importances = model.feature_importances_
indices = np.argsort(importances)
# Create a bar plot of feature importances
plt.figure(figsize=(8, 8))
plt.title("Feature Importances")
plt.barh(range(len(indices)), importances[indices], color="violet", align="cen
ter")
# Customize y-axis with feature names
plt.yticks(range(len(indices)), [feature_names[i] for i in indices])
# Display the plot
plt.xlabel("Relative Importance")
plt.show()
```



Checking model performance on test data

In [42]:
```
# Compute and display the confusion matrix for the test set
confusion_matrix_sklearn(model, X_test, y_test)
```

Out[42]:
```
array([[1342,    9],
       [  19,  130]])
```



Model Performance Comparison and Final Model Selection

Pre-pruning

In [43]:
```
# Choose the type of classifier.
estimator = DecisionTreeClassifier(random_state=1)
# Grid of parameters to choose from
parameters = {
    "max_depth": np.arange(6, 15),
    "min_samples_leaf": [1, 2, 5, 7, 10],
    "max_leaf_nodes": [2, 3, 5, 10],
}
# Type of scoring used to compare parameter combinations
acc_scorer = make_scorer(recall_score)
# Run the grid search
grid_obj = GridSearchCV(estimator, parameters, scoring=acc_scorer, cv=5)
grid_obj = grid_obj.fit(X_train, y_train)
# Set the clf to the best combination of parameters
estimator = grid_obj.best_estimator_
# Fit the best algorithm to the data.
estimator.fit(X_train, y_train)
```

Out[43]:

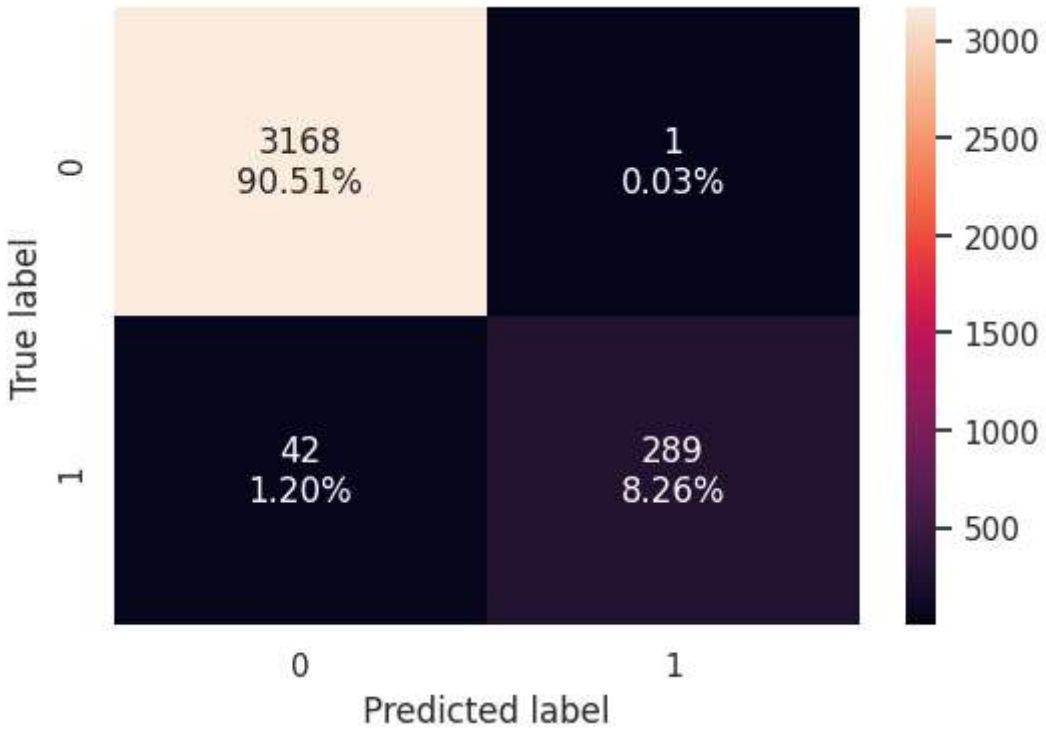
DecisionTreeClassifier (https learn.

```
DecisionTreeClassifier(max_depth=6, max_leaf_nodes=10, min_samples_leaf=10,
                       random_state=1)
```

Checking performance on the training data

In [44]:
```
# Compute and display the confusion matrix for the training set
confusion_matrix_sklearn(estimator, X_train, y_train)
```

Out[44]:
```
array([[3168,    1],
       [  42,  289]])
```



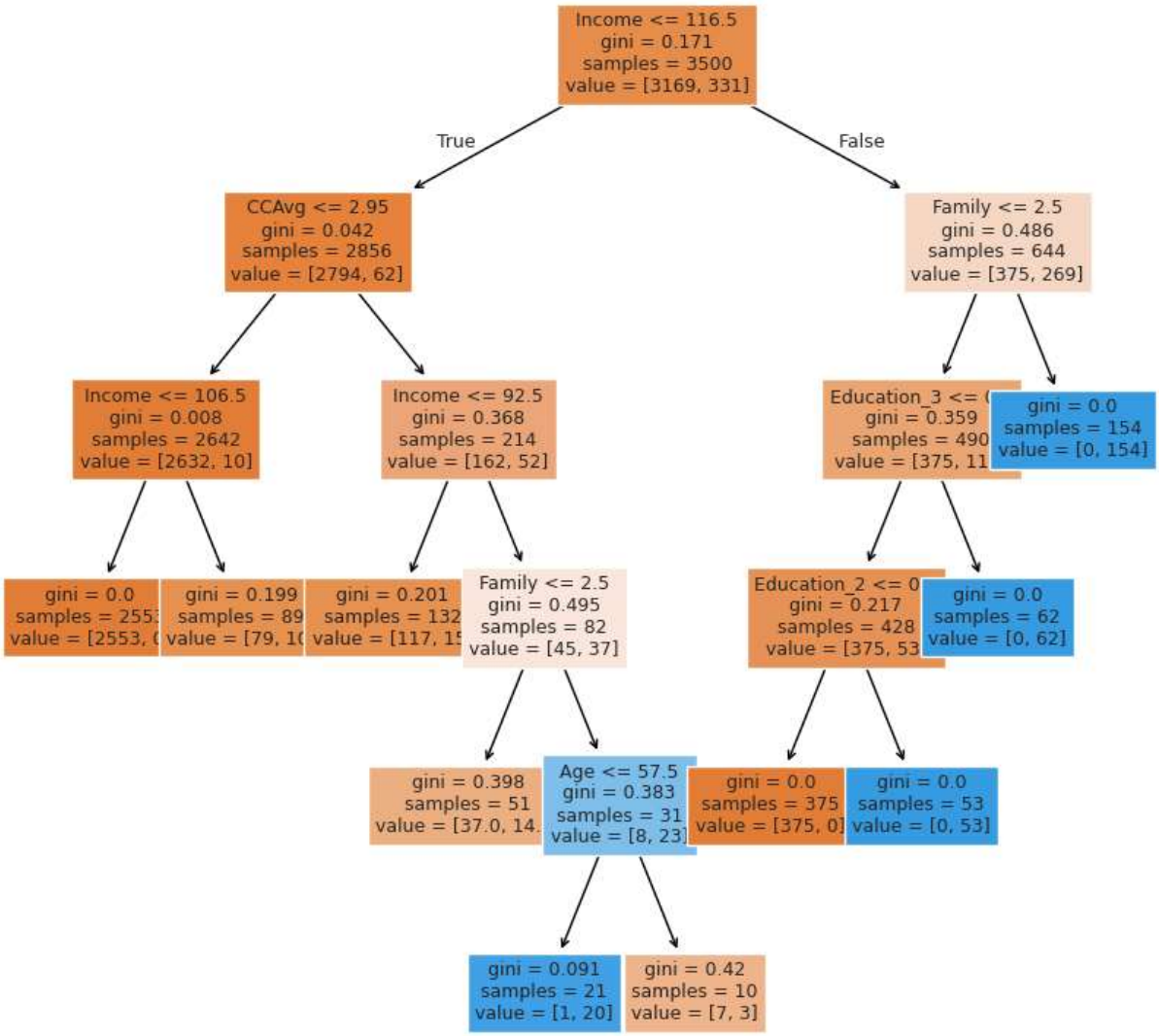
In [45]:
```
# Evaluate the performance of the tuned decision tree on the training data
decision_tree_tune_perf_train = model_performance_classification_sklearn(estim
ator, X_train, y_train)
# Display the performance metrics
decision_tree_tune_perf_train
```

Out[45]:

	Accuracy	Recall	Precision	F1
0	0.987714	0.873112	0.996552	0.930757

Visualizing the Decision Tree

In [46]:
```
# Display the Decision Tree for this training data
plt.figure(figsize=(10, 10))
out = tree.plot_tree(
    estimator,
    feature_names=feature_names,
    filled=True,
    fontsize=9,
    node_ids=False,
    class_names=None,
)
# Add arrows to the decision tree split if they are missing
for o in out:
    arrow = o.arrow_patch
    if arrow is not None:
        arrow.set_edgecolor("black")
        arrow.set_linewidth(1)
plt.show()
```



In [47]:
```
# Text report showing the rules of a decision tree
print(tree.export_text(estimator, feature_names=feature_names, show_weights=True))
```

```
|--- Income <= 116.50
|   |--- CCAvg <= 2.95
|   |   |--- Income <= 106.50
|   |   |   |--- weights: [2553.00, 0.00] class: 0
|   |   |--- Income >  106.50
|   |   |   |--- weights: [79.00, 10.00] class: 0
|   |--- CCAvg >  2.95
|   |   |--- Income <= 92.50
|   |   |   |--- weights: [117.00, 15.00] class: 0
|   |   |--- Income >  92.50
|   |   |   |--- Family <= 2.50
|   |   |   |   |--- weights: [37.00, 14.00] class: 0
|   |   |   |--- Family >  2.50
|   |   |   |   |--- Age <= 57.50
|   |   |   |   |   |--- weights: [1.00, 20.00] class: 1
|   |   |   |   |--- Age >  57.50
|   |   |   |   |   |--- weights: [7.00, 3.00] class: 0
|--- Income >  116.50
|   |--- Family <= 2.50
|   |   |--- Education_3 <= 0.50
|   |   |   |--- Education_2 <= 0.50
|   |   |   |   |--- weights: [375.00, 0.00] class: 0
|   |   |   |--- Education_2 >  0.50
|   |   |   |   |--- weights: [0.00, 53.00] class: 1
|   |   |--- Education_3 >  0.50
|   |   |   |--- weights: [0.00, 62.00] class: 1
|   |--- Family >  2.50
|   |   |--- weights: [0.00, 154.00] class: 1
```

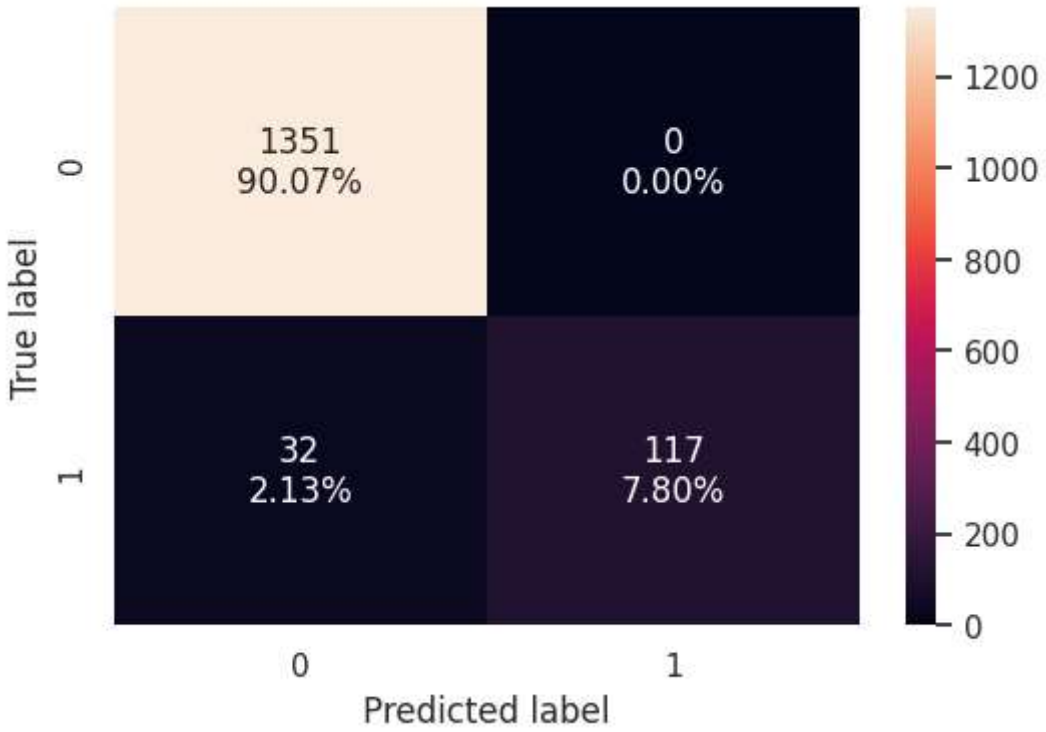
In [48]:
```
# Create and print a DataFrame of feature importances
 # 1. Extract feature importances from the model
 # 2. Create a DataFrame with importances and feature names computed as the  Gini importance
 # 3. Display the result
print(
    pd.DataFrame(
        estimator.feature_importances_, columns=["Imp"], index=X_train.columns
    ).sort_values(by="Imp", ascending=False)
)
```

```
                     Imp
Income          0.337681
Family          0.275581
Education_2     0.175687
Education_3     0.157286
CCAvg           0.042856
...                  ...
ZIPCode_92093   0.000000
ZIPCode_92084   0.000000
ZIPCode_92069   0.000000
ZIPCode_92068   0.000000
ZIPCode_93009   0.000000

[478 rows x 1 columns]
```

Checking the performance on the test data

In [49]:
```
# Compute and display the confusion matrix for the test set
confusion_matrix_sklearn(estimator, X_test, y_test)
```

Out[49]:
```
array([[1351,    0],
       [  32,  117]])
```



In [50]:
```
# Evaluate the performance of the tuned decision tree on the test data
decision_tree_tune_perf_test = model_performance_classification_sklearn(estimator, X_test, y_test)
# Display the performance metrics
decision_tree_tune_perf_test
```

Out[50]:

	Accuracy	Recall	Precision	F1
0	0.978667	0.785235	1.0	0.879699

Cost-complexity Pruning

- Cost complexity pruning helps to find the optimal tree size by balancing the trade-off between tree complexity and its accuracy on the training data.
- The ccp_alpha parameter determines the strength of the pruning

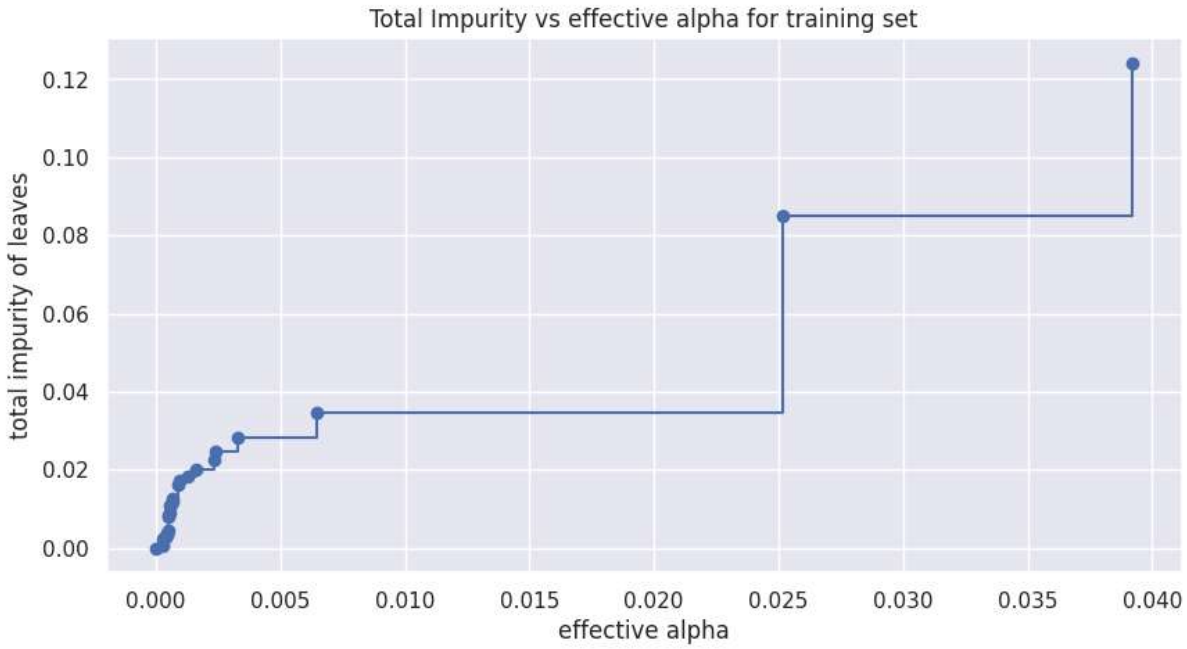
In [51]:
```
# Perform cost complexity pruning on the decision tree
clf = DecisionTreeClassifier(random_state=1)
path = clf.cost_complexity_pruning_path(X_train, y_train)
ccp_alphas, impurities = path.ccp_alphas, path.impurities
```

In [52]:
```
# Display the cost complexity pruning path as a DataFrame
pd.DataFrame(path)
```

Out[52]:

	ccp_alphas	impurities
0	0.000000	0.000000
1	0.000276	0.000552
2	0.000279	0.002224
3	0.000381	0.002605
4	0.000381	0.002986
5	0.000476	0.003462
6	0.000476	0.003938
7	0.000500	0.004438
8	0.000513	0.008031
9	0.000527	0.008558
10	0.000544	0.009103
11	0.000545	0.010739
12	0.000625	0.011364
13	0.000667	0.012031
14	0.000700	0.012731
15	0.000882	0.016260
16	0.000940	0.017200
17	0.001305	0.018505
18	0.001647	0.020153
19	0.002333	0.022486
20	0.002407	0.024893
21	0.003294	0.028187
22	0.006473	0.034659
23	0.025146	0.084951
24	0.039216	0.124167
25	0.047088	0.171255

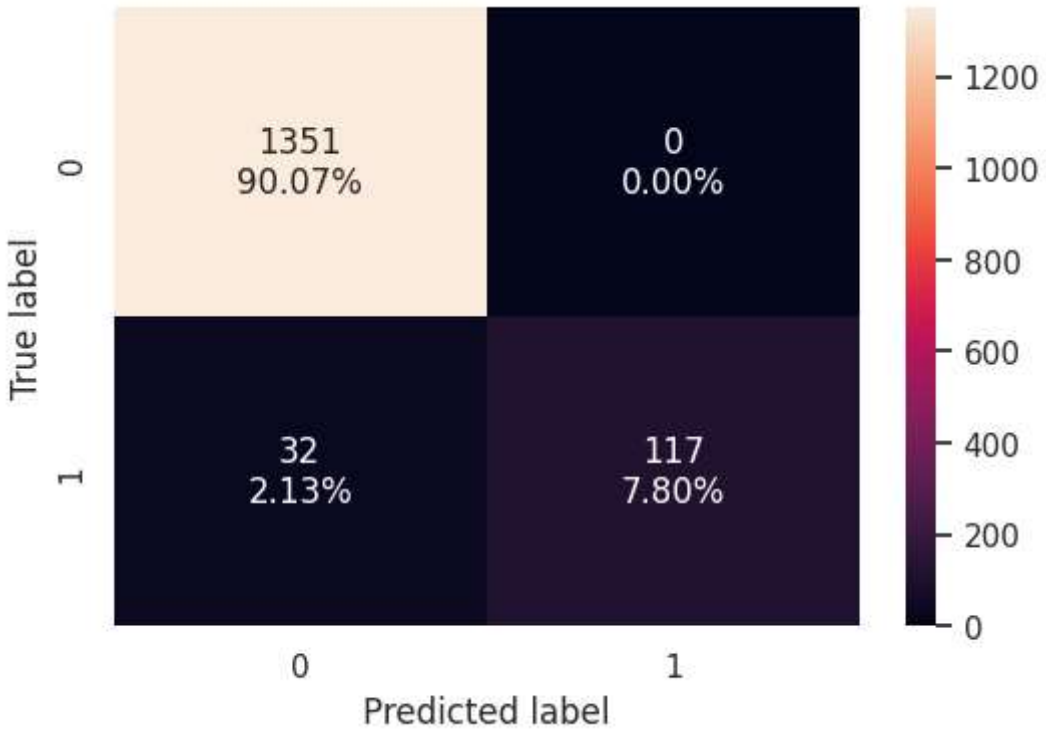
In [53]:
```
# Create a figure and axis object with specified size
fig, ax = plt.subplots(figsize=(10, 5))
# Plot the relationship between ccp_alphas and impurities
# Use markers for data points and step-style for the line
ax.plot(ccp_alphas[:-1], impurities[:-1], marker="o", drawstyle="steps-post")
# Set the x-axis label
ax.set_xlabel("effective alpha")
# Set the y-axis label
ax.set_ylabel("total impurity of leaves")
# Set the title of the plot
ax.set_title("Total Impurity vs effective alpha for training set")
# Display the plot
plt.show()
```



In [66]:
```
# Initialize an empty list to store recall scores for the training data
recall_train = []
# Loop through each classifier in the list of classifiers
for clf in clfs:
  # Predict the training data using the current classifier
  pred_train = clf.predict(X_train)

# Calculate the recall score for the training data predictions
  values_train = recall_score(y_train, pred_train)

# Append the recall score to the recall_train list
  recall_train.append(values_train)
# Initialize an empty list to store recall scores for the test data
recall_test = []
 # Loop through each classifier in the list of classifiers
for clf in clfs:
  # Predict the test data using the current classifier
  pred_test = clf.predict(X_test)

# Calculate the recall score for the test data predictions
  values_test = recall_score(y_test, pred_test)

# Append the recall score to the recall_test list
  recall_test.append(values_test)
```

In [68]:
```
# Compute and display the confusion matrix for the test set
confusion_matrix_sklearn(estimator, X_test, y_test) # Assuming you want to eva
luate 'estimator' instead of 'estimator_2'
```

Out[68]:
```
array([[1351,    0],
       [  32,  117]])
```



In [73]:
```
# Initialize an empty list to store recall scores for the training data
recall_train = []
# Loop through each classifier in the list of classifiers
for clf in clfs:
  # Predict the training data using the current classifier
  pred_train = clf.predict(X_train)
  # Calculate the recall score for the training data predictions
  values_train = recall_score(y_train, pred_train)

# Append the recall score to the recall_train list
recall_train.append(values_train)
# Initialize an empty list to store recall scores for the test data
recall_test = []
# Loop through each classifier in the list of classifiers
for clf in clfs:
  # Predict the test data using the current classifier
  pred_test = clf.predict(X_test)

# Calculate the recall score for the test data predictions
values_test = recall_score(y_test, pred_test)

# Append the recall score to the recall_test list
recall_test.append(values_test)
```

In [80]:
```
# Find the index of the classifier with the highest recall score on the test d
ata
index_best_model = np.argmax(recall_test)
# Select the classifier with the highest recall score on the test data
best_model = clfs[index_best_model]
# Print the details of the best classifier
print("Best model index:", index_best_model)
print("Best model type:", type(best_model))
print("Best model parameters:")
for param, value in best_model.get_params().items():
  print(f"  {param}: {value}")
# Print feature importances
if hasattr(best_model, 'feature_importances_'):
  print("\nFeature Importances:")
for feature, importance in zip(X_train.columns, best_model.feature_importances
_):
  print(f"  {feature}: {importance}")
```

```
Best model index: 0
Best model type: <class 'sklearn.tree._classes.DecisionTreeClassifier'>
Best model parameters:
  ccp_alpha: 0.0
  class_weight: None
  criterion: gini
  max_depth: None
  max_features: None
  max_leaf_nodes: None
  min_impurity_decrease: 0.0
  min_samples_leaf: 1
  min_samples_split: 2
  min_weight_fraction_leaf: 0.0
  monotonic_cst: None
  random_state: 1
  splitter: best

Feature Importances:
  ID: 0.003892822493332149
  Age: 0.02445668360615259
  Income: 0.29781592102929316
  Family: 0.24853023062132423
  CCAvg: 0.04754972346331871
  Mortgage: 0.0022244699961897997
  Securities_Account: 0.0
  CD_Account: 0.021360903064610828
  Online: 0.0
  CreditCard: 0.0
  ZIPCode_90007: 0.0
  ZIPCode_90009: 0.0
  ZIPCode_90011: 0.0
  ZIPCode_90016: 0.0
  ZIPCode_90018: 0.0
  ZIPCode_90019: 0.0
  ZIPCode_90024: 0.0
  ZIPCode_90025: 0.0
  ZIPCode_90027: 0.0
  ZIPCode_90028: 0.0
  ZIPCode_90029: 0.0
  ZIPCode_90032: 0.0
  ZIPCode_90033: 0.0
  ZIPCode_90034: 0.0015571289973328585
  ZIPCode_90035: 0.0
  ZIPCode_90036: 0.0
  ZIPCode_90037: 0.0
  ZIPCode_90041: 0.0
  ZIPCode_90044: 0.0
  ZIPCode_90045: 0.0
  ZIPCode_90048: 0.0
  ZIPCode_90049: 0.0032845689787490014
  ZIPCode_90057: 0.0
  ZIPCode_90058: 0.0
  ZIPCode_90059: 0.0
  ZIPCode_90064: 0.0
  ZIPCode_90065: 0.0
  ZIPCode_90066: 0.0
  ZIPCode_90068: 0.0
```

```
ZIPCode_90071: 0.0
ZIPCode_90073: 0.0
ZIPCode_90086: 0.0
ZIPCode_90089: 0.0
ZIPCode_90095: 0.0
ZIPCode_90210: 0.0
ZIPCode_90212: 0.0
ZIPCode_90230: 0.0
ZIPCode_90232: 0.0
ZIPCode_90245: 0.003177814280271144
ZIPCode_90250: 0.0
ZIPCode_90254: 0.0
ZIPCode_90266: 0.0
ZIPCode_90272: 0.0
ZIPCode_90274: 0.0
ZIPCode_90275: 0.0
ZIPCode_90277: 0.0
ZIPCode_90280: 0.0
ZIPCode_90291: 0.0
ZIPCode_90304: 0.0
ZIPCode_90401: 0.0
ZIPCode_90404: 0.0
ZIPCode_90405: 0.0
ZIPCode_90502: 0.0
ZIPCode_90503: 0.0
ZIPCode_90504: 0.0
ZIPCode_90505: 0.0
ZIPCode_90509: 0.0
ZIPCode_90601: 0.0015312082664450438
ZIPCode_90623: 0.0
ZIPCode_90630: 0.0
ZIPCode_90638: 0.0
ZIPCode_90639: 0.0
ZIPCode_90640: 0.0
ZIPCode_90650: 0.0
ZIPCode_90717: 0.0
ZIPCode_90720: 0.0
ZIPCode_90740: 0.0
ZIPCode_90745: 0.0
ZIPCode_90747: 0.0
ZIPCode_90755: 0.0
ZIPCode_90813: 0.0
ZIPCode_90840: 0.0
ZIPCode_91006: 0.0
ZIPCode_91007: 0.0
ZIPCode_91016: 0.0
ZIPCode_91024: 0.0
ZIPCode_91030: 0.0
ZIPCode_91040: 0.0
ZIPCode_91101: 0.0
ZIPCode_91103: 0.0
ZIPCode_91105: 0.0
ZIPCode_91107: 0.0
ZIPCode_91109: 0.0
ZIPCode_91116: 0.0
ZIPCode_91125: 0.0
ZIPCode_91129: 0.0
```

```
ZIPCode_91203: 0.0030438594340106095
ZIPCode_91207: 0.0
ZIPCode_91301: 0.0
ZIPCode_91302: 0.0
ZIPCode_91304: 0.0
ZIPCode_91311: 0.0
ZIPCode_91320: 0.0
ZIPCode_91326: 0.0
ZIPCode_91330: 0.0
ZIPCode_91335: 0.0
ZIPCode_91342: 0.0
ZIPCode_91343: 0.0
ZIPCode_91345: 0.0
ZIPCode_91355: 0.0
ZIPCode_91360: 0.002854402866314406
ZIPCode_91361: 0.0
ZIPCode_91365: 0.0
ZIPCode_91367: 0.0
ZIPCode_91380: 0.0
ZIPCode_91401: 0.0
ZIPCode_91423: 0.0
ZIPCode_91604: 0.0
ZIPCode_91605: 0.0
ZIPCode_91614: 0.0
ZIPCode_91706: 0.0
ZIPCode_91709: 0.0
ZIPCode_91710: 0.0
ZIPCode_91711: 0.0
ZIPCode_91730: 0.0
ZIPCode_91741: 0.0
ZIPCode_91745: 0.0
ZIPCode_91754: 0.0
ZIPCode_91763: 0.0
ZIPCode_91765: 0.0
ZIPCode_91768: 0.0
ZIPCode_91770: 0.0
ZIPCode_91773: 0.0
ZIPCode_91775: 0.0
ZIPCode_91784: 0.0
ZIPCode_91791: 0.0
ZIPCode_91801: 0.0
ZIPCode_91902: 0.0
ZIPCode_91910: 0.0
ZIPCode_91911: 0.0
ZIPCode_91941: 0.0
ZIPCode_91942: 0.0
ZIPCode_91950: 0.0
ZIPCode_92007: 0.0030870354597473232
ZIPCode_92008: 0.0
ZIPCode_92009: 0.0
ZIPCode_92024: 0.0
ZIPCode_92028: 0.0
ZIPCode_92029: 0.0
ZIPCode_92037: 0.0
ZIPCode_92038: 0.0
ZIPCode_92054: 0.0
ZIPCode_92056: 0.0
```

```
ZIPCode_92064: 0.0
ZIPCode_92068: 0.0
ZIPCode_92069: 0.0
ZIPCode_92084: 0.0
ZIPCode_92093: 0.0
ZIPCode_92096: 0.0
ZIPCode_92101: 0.0
ZIPCode_92103: 0.0
ZIPCode_92104: 0.0
ZIPCode_92106: 0.0
ZIPCode_92109: 0.0
ZIPCode_92110: 0.0
ZIPCode_92115: 0.0
ZIPCode_92116: 0.0
ZIPCode_92120: 0.0
ZIPCode_92121: 0.0
ZIPCode_92122: 0.0015840085514948932
ZIPCode_92123: 0.0
ZIPCode_92124: 0.0
ZIPCode_92126: 0.0
ZIPCode_92129: 0.0
ZIPCode_92130: 0.0
ZIPCode_92131: 0.0
ZIPCode_92152: 0.0
ZIPCode_92154: 0.0
ZIPCode_92161: 0.0
ZIPCode_92173: 0.0
ZIPCode_92177: 0.0
ZIPCode_92182: 0.0
ZIPCode_92192: 0.0
ZIPCode_92220: 0.0030941711601925955
ZIPCode_92251: 0.0
ZIPCode_92325: 0.0
ZIPCode_92333: 0.0
ZIPCode_92346: 0.0
ZIPCode_92350: 0.0
ZIPCode_92354: 0.0
ZIPCode_92373: 0.0
ZIPCode_92374: 0.0
ZIPCode_92399: 0.0
ZIPCode_92407: 0.0
ZIPCode_92507: 0.0
ZIPCode_92518: 0.0
ZIPCode_92521: 0.002994764927010445
ZIPCode_92606: 0.0
ZIPCode_92612: 0.0
ZIPCode_92614: 0.0
ZIPCode_92624: 0.0
ZIPCode_92626: 0.0
ZIPCode_92630: 0.0
ZIPCode_92634: 0.0
ZIPCode_92646: 0.0
ZIPCode_92647: 0.0
ZIPCode_92648: 0.0
ZIPCode_92653: 0.0
ZIPCode_92660: 0.0
ZIPCode_92661: 0.0
```

```
ZIPCode_92672: 0.0
ZIPCode_92673: 0.0
ZIPCode_92675: 0.0
ZIPCode_92677: 0.0
ZIPCode_92691: 0.0
ZIPCode_92692: 0.0
ZIPCode_92694: 0.0
ZIPCode_92697: 0.0
ZIPCode_92703: 0.0
ZIPCode_92704: 0.0
ZIPCode_92705: 0.0
ZIPCode_92709: 0.0
ZIPCode_92717: 0.0
ZIPCode_92735: 0.0
ZIPCode_92780: 0.0
ZIPCode_92806: 0.0
ZIPCode_92807: 0.0
ZIPCode_92821: 0.0
ZIPCode_92831: 0.0
ZIPCode_92833: 0.0
ZIPCode_92834: 0.0
ZIPCode_92835: 0.0
ZIPCode_92843: 0.0
ZIPCode_92866: 0.0
ZIPCode_92867: 0.0
ZIPCode_92868: 0.0
ZIPCode_92870: 0.0
ZIPCode_92886: 0.0
ZIPCode_93003: 0.0
ZIPCode_93009: 0.0
ZIPCode_93010: 0.0
ZIPCode_93014: 0.0
ZIPCode_93022: 0.0
ZIPCode_93023: 0.0
ZIPCode_93033: 0.0
ZIPCode_93063: 0.0
ZIPCode_93065: 0.0
ZIPCode_93077: 0.0
ZIPCode_93101: 0.0
ZIPCode_93105: 0.0
ZIPCode_93106: 0.0031835053178644115
ZIPCode_93107: 0.0
ZIPCode_93108: 0.0
ZIPCode_93109: 0.0
ZIPCode_93111: 0.0
ZIPCode_93117: 0.0
ZIPCode_93118: 0.0
ZIPCode_93302: 0.0
ZIPCode_93305: 0.0
ZIPCode_93311: 0.0
ZIPCode_93401: 0.0
ZIPCode_93403: 0.0
ZIPCode_93407: 0.0
ZIPCode_93437: 0.0
ZIPCode_93460: 0.0027805874952372502
ZIPCode_93524: 0.0
ZIPCode_93555: 0.0
```

```
ZIPCode_93561: 0.0
ZIPCode_93611: 0.0
ZIPCode_93657: 0.0
ZIPCode_93711: 0.0
ZIPCode_93720: 0.0
ZIPCode_93727: 0.0
ZIPCode_93907: 0.0
ZIPCode_93933: 0.0
ZIPCode_93940: 0.0
ZIPCode_93943: 0.0
ZIPCode_93950: 0.0
ZIPCode_93955: 0.0
ZIPCode_94002: 0.0
ZIPCode_94005: 0.0
ZIPCode_94010: 0.0
ZIPCode_94015: 0.0
ZIPCode_94019: 0.0
ZIPCode_94022: 0.0
ZIPCode_94024: 0.0
ZIPCode_94025: 0.0
ZIPCode_94028: 0.0
ZIPCode_94035: 0.0
ZIPCode_94040: 0.0
ZIPCode_94043: 0.0
ZIPCode_94061: 0.0
ZIPCode_94063: 0.0
ZIPCode_94065: 0.0
ZIPCode_94066: 0.0
ZIPCode_94080: 0.0
ZIPCode_94085: 0.0
ZIPCode_94086: 0.0
ZIPCode_94087: 0.0
ZIPCode_94102: 0.0
ZIPCode_94104: 0.0
ZIPCode_94105: 0.00294684868817828
ZIPCode_94107: 0.0
ZIPCode_94108: 0.0
ZIPCode_94109: 0.0
ZIPCode_94110: 0.0
ZIPCode_94111: 0.0
ZIPCode_94112: 0.0
ZIPCode_94114: 0.0
ZIPCode_94115: 0.0
ZIPCode_94116: 0.0
ZIPCode_94117: 0.0
ZIPCode_94118: 0.0
ZIPCode_94122: 0.0031457406795291688
ZIPCode_94123: 0.0
ZIPCode_94124: 0.0
ZIPCode_94126: 0.0
ZIPCode_94131: 0.0
ZIPCode_94132: 0.0
ZIPCode_94143: 0.0
ZIPCode_94234: 0.0
ZIPCode_94301: 0.0
ZIPCode_94302: 0.0
ZIPCode_94303: 0.0
```

```
ZIPCode_94304: 0.0
ZIPCode_94305: 0.0006385763006303433
ZIPCode_94306: 0.0
ZIPCode_94309: 0.0
ZIPCode_94402: 0.0
ZIPCode_94404: 0.0
ZIPCode_94501: 0.0
ZIPCode_94507: 0.0
ZIPCode_94509: 0.0
ZIPCode_94521: 0.0
ZIPCode_94523: 0.0
ZIPCode_94526: 0.0
ZIPCode_94534: 0.0
ZIPCode_94536: 0.0
ZIPCode_94538: 0.0
ZIPCode_94539: 0.0
ZIPCode_94542: 0.0
ZIPCode_94545: 0.0
ZIPCode_94546: 0.0
ZIPCode_94550: 0.0
ZIPCode_94551: 0.0
ZIPCode_94553: 0.0
ZIPCode_94555: 0.0
ZIPCode_94558: 0.0
ZIPCode_94566: 0.0
ZIPCode_94571: 0.0
ZIPCode_94575: 0.0
ZIPCode_94577: 0.0
ZIPCode_94583: 0.0
ZIPCode_94588: 0.0
ZIPCode_94590: 0.0
ZIPCode_94591: 0.0
ZIPCode_94596: 0.0
ZIPCode_94598: 0.0
ZIPCode_94604: 0.0
ZIPCode_94606: 0.0029196168699991126
ZIPCode_94607: 0.0
ZIPCode_94608: 0.0
ZIPCode_94609: 0.0
ZIPCode_94610: 0.0
ZIPCode_94611: 0.0
ZIPCode_94612: 0.0
ZIPCode_94618: 0.0
ZIPCode_94701: 0.0
ZIPCode_94703: 0.0
ZIPCode_94704: 0.0
ZIPCode_94705: 0.0
ZIPCode_94706: 0.0
ZIPCode_94707: 0.0
ZIPCode_94708: 0.0
ZIPCode_94709: 0.0029000733121754482
ZIPCode_94710: 0.0
ZIPCode_94720: 0.0
ZIPCode_94801: 0.0
ZIPCode_94803: 0.0
ZIPCode_94806: 0.0
ZIPCode_94901: 0.0
```

```
ZIPCode_94904: 0.0
ZIPCode_94920: 0.0
ZIPCode_94923: 0.0
ZIPCode_94928: 0.0
ZIPCode_94939: 0.0
ZIPCode_94949: 0.0
ZIPCode_94960: 0.0
ZIPCode_94965: 0.0
ZIPCode_94970: 0.0
ZIPCode_94998: 0.0
ZIPCode_95003: 0.0
ZIPCode_95005: 0.0
ZIPCode_95006: 0.0
ZIPCode_95008: 0.0
ZIPCode_95010: 0.0
ZIPCode_95014: 0.0
ZIPCode_95020: 0.0
ZIPCode_95023: 0.0
ZIPCode_95032: 0.0
ZIPCode_95035: 0.0
ZIPCode_95037: 0.0
ZIPCode_95039: 0.0
ZIPCode_95045: 0.0
ZIPCode_95051: 0.0
ZIPCode_95053: 0.0
ZIPCode_95054: 0.0
ZIPCode_95060: 0.0
ZIPCode_95064: 0.0
ZIPCode_95070: 0.0
ZIPCode_95112: 0.0
ZIPCode_95120: 0.0
ZIPCode_95123: 0.0
ZIPCode_95125: 0.0
ZIPCode_95126: 0.0
ZIPCode_95131: 0.0
ZIPCode_95133: 0.0
ZIPCode_95134: 0.0
ZIPCode_95135: 0.0
ZIPCode_95136: 0.0
ZIPCode_95138: 0.0
ZIPCode_95192: 0.0
ZIPCode_95193: 0.0
ZIPCode_95207: 0.0
ZIPCode_95211: 0.0
ZIPCode_95307: 0.0
ZIPCode_95348: 0.0
ZIPCode_95351: 0.0
ZIPCode_95354: 0.0
ZIPCode_95370: 0.0
ZIPCode_95403: 0.0
ZIPCode_95405: 0.0
ZIPCode_95422: 0.0
ZIPCode_95449: 0.0
ZIPCode_95482: 0.0
ZIPCode_95503: 0.0
ZIPCode_95518: 0.0
ZIPCode_95521: 0.0
```

```
ZIPCode_95605: 0.0
ZIPCode_95616: 0.0
ZIPCode_95617: 0.0
ZIPCode_95621: 0.0
ZIPCode_95630: 0.0
ZIPCode_95670: 0.0
ZIPCode_95678: 0.0
ZIPCode_95741: 0.0
ZIPCode_95747: 0.0
ZIPCode_95758: 0.0
ZIPCode_95762: 0.0
ZIPCode_95812: 0.0
ZIPCode_95814: 0.0
ZIPCode_95816: 0.0
ZIPCode_95817: 0.0
ZIPCode_95818: 0.0
ZIPCode_95819: 0.0
ZIPCode_95820: 0.0
ZIPCode_95821: 0.0
ZIPCode_95822: 0.0
ZIPCode_95825: 0.0
ZIPCode_95827: 0.0
ZIPCode_95828: 0.0
ZIPCode_95831: 0.0
ZIPCode_95833: 0.0
ZIPCode_95841: 0.0
ZIPCode_95842: 0.0
ZIPCode_95929: 0.0
ZIPCode_95973: 0.0
ZIPCode_96001: 0.0
ZIPCode_96003: 0.0
ZIPCode_96008: 0.0
ZIPCode_96064: 0.0
ZIPCode_96091: 0.0
ZIPCode_96094: 0.0
ZIPCode_96145: 0.0
ZIPCode_96150: 0.0
ZIPCode_96651: 0.0
Education_2: 0.16523843019084813
Education_3: 0.14420690394974808
```

Post Pruning

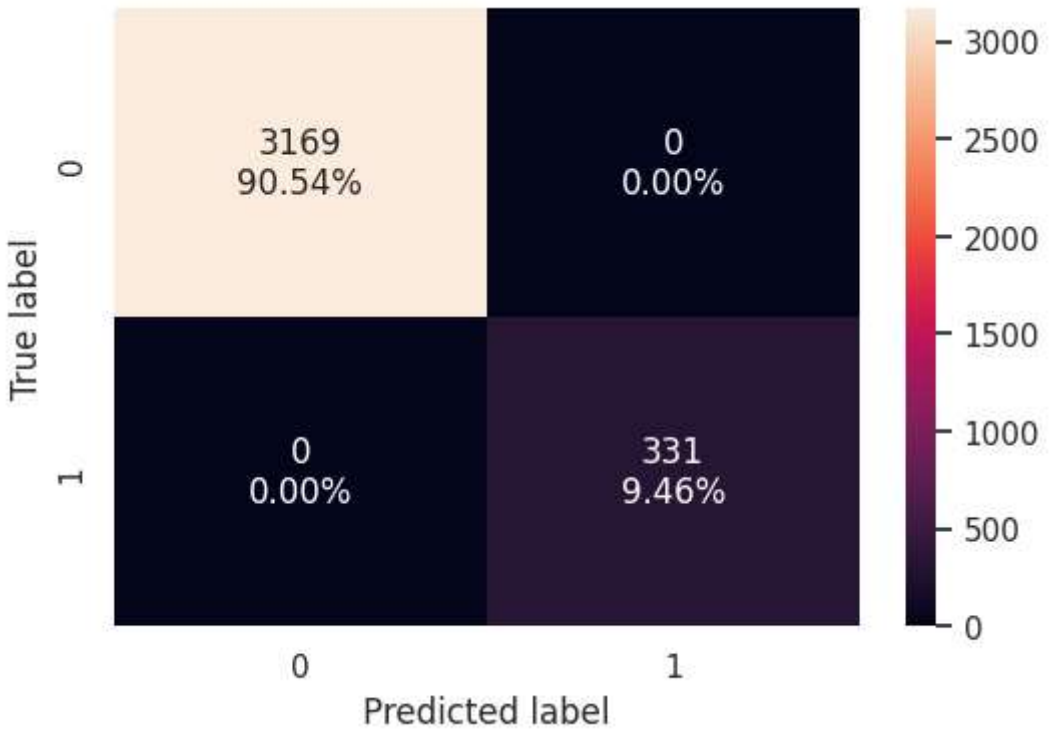
In [81]:
```
# Create a new DecisionTreeClassifier with the ccp_alpha value from the best model identified earlier,
# specified class weights, and a fixed random state for reproducibility
estimator_2 = DecisionTreeClassifier(
 ccp_alpha=best_model.ccp_alpha,  # Use the ccp_alpha value from the best model
 class_weight={0: 0.15, 1: 0.85},  # Assign class weights to handle class imbalance
 random_state=1  # Set random state for reproducibility
 )
# Fit the new classifier to the training data (X_train and y_train)
estimator_2.fit(X_train, y_train)
```

Out[81]:
```
DecisionTreeClassifier(class_weight={0: 0.15, 1: 0.85}, random_state=1)
```

Check the performance on training data

In [82]:
```
# Compute and display the confusion matrix for the training set
confusion_matrix_sklearn(estimator_2, X_train, y_train)
```

Out[82]:
```
array([[3169,    0],
       [   0,  331]])
```



In [83]:
```
# Compute and display the performance metrics for the training set
decision_tree_perf_train = model_performance_classification_sklearn(
estimator_2, X_train, y_train
)
# Display the performance metrics
decision_tree_perf_train
```

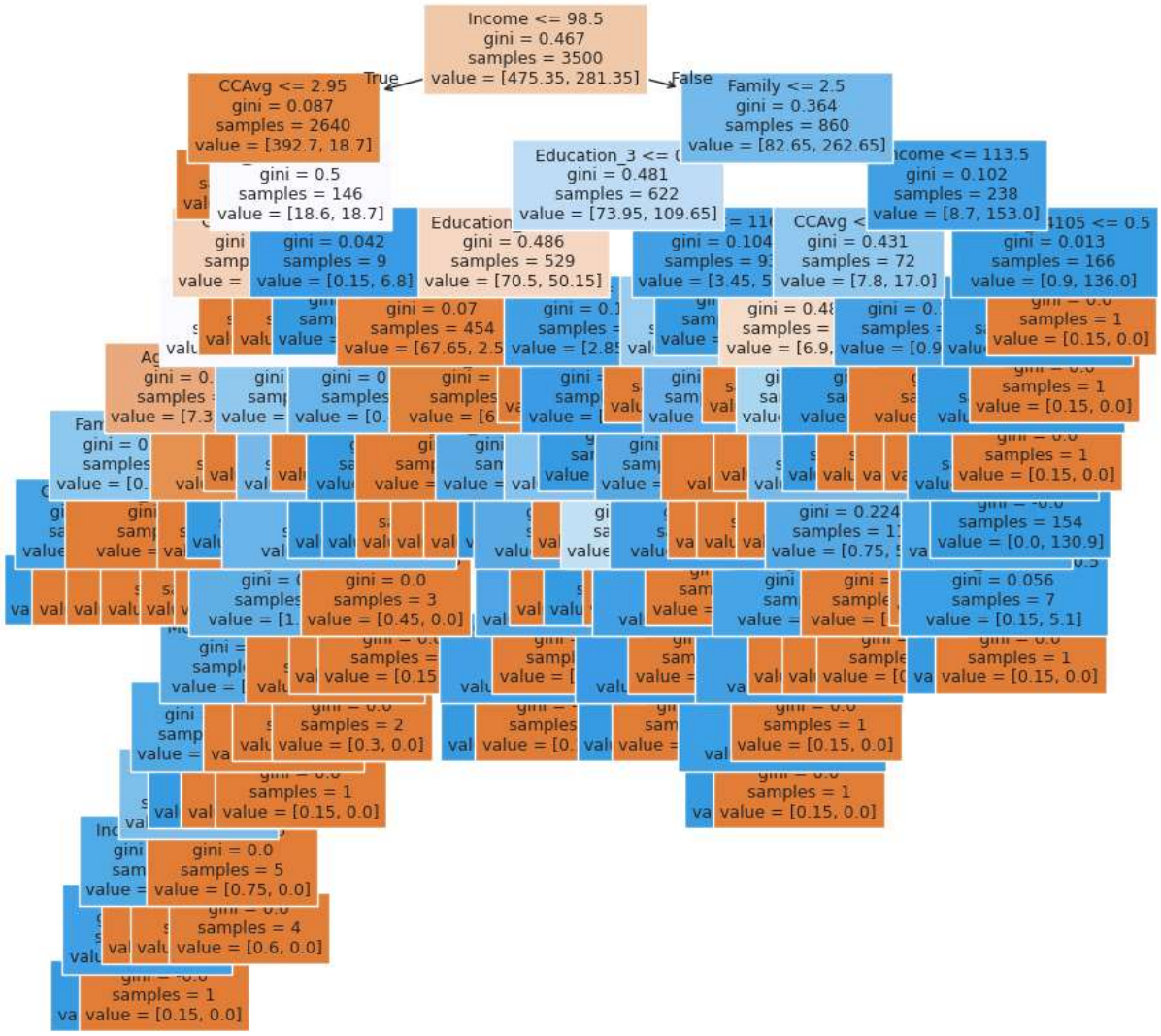
Out[83]:

	Accuracy	Recall	Precision	F1
0	1.0	1.0	1.0	1.0

- The classification model has achieved perfect performance metrics across the board, with an Accuracy, Recall, Precision, and F1-score all equal to 1.0. This indicates that the model correctly classified all instances with no false positives or false negatives **bold text**
- While this is an ideal result, it is important to consider whether this performance is realistic or if it might indicate overfitting, especially if the model was evaluated on the training data or a non-representative test set

Visualizing the Decision Tree

In [84]:
```
# Set up the figure size for the plot
plt.figure(figsize=(10, 10))
# Plot the decision tree using the specified estimator and configuration
out = tree.plot_tree(
estimator_2,
# The decision tree classifier to plot
feature_names=feature_names,  # Names of the features used in the decision tre
e
filled=True,
# Color the nodes to reflect the majority class
fontsize=9,
node_ids=False,
class_names=None
)
# Set the font size for the text in the plot
# Do not display node IDs
# Do not display class names
# Loop through the plotted tree elements to add arrows to the decision tree sp
lits if they are missing
for o in out:
  arrow = o.arrow_patch
if arrow is not None:
  arrow.set_edgecolor("black")  # Set the edge color of the arrows to black
  arrow.set_linewidth(1)
# Set the line width of the arrows to 1
# Display the plot
plt.show()
```


Income <= 98.5
gini = 0.467
samples = 3500
value = [475.35, 281.35]
True
False
CCAvg <= 2.95
gini = 0.087
samples = 2640
value = [392.7, 18.7]
Family <= 2.5
gini = 0.364
samples = 860
value = [82.65, 262.65]
gini = 0.5
samples = 146
value = [18.6, 18.7]
Education_3 <= 0
gini = 0.481
samples = 622
value = [73.95, 109.65]
Income <= 113.5
gini = 0.102
samples = 238
value = [8.7, 153.0]
gini = 0.042
samples = 9
value = [0.15, 6.8]
Education
gini = 0.486
samples = 529
value = [70.5, 50.15]
gini = 0.104
CCAvg <
gini = 0.431
samples = 72
value = [7.8, 17.0]
1105 <= 0.5
gini = 0.013
samples = 166
value = [0.9, 136.0]
gini = 0.07
samples = 454
value = [67.65, 2.5
gini = 0.48
samples = 1
value = [0.15, 0.0]
gini = 0.056
samples = 7
value = [0.15, 5.1]
gini = 0.0
samples = 3
value = [0.45, 0.0]
gini = 0.224
samples = 154
value = [0.0, 130.9]
samples = 2
value = [0.3, 0.0]
gini = 0.0
samples = 5
value = [0.75, 0.0]
samples = 4
value = [0.6, 0.0]

Actionable Insights and Business Recommendations

In [85]:
```
# Text report showing the rules of a decision tree
print(tree.export_text(estimator_2, 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 <= 3.50
|   |   |   |   |   |   |   |--- CCAvg <= 3.50
|   |   |   |   |   |   |   |   |--- weights: [0.00, 1.70] class: 1
|   |   |   |   |   |   |   |--- CCAvg >  3.50
|   |   |   |   |   |   |   |   |--- weights: [0.15, 0.00] class: 0
|   |   |   |   |   |   |--- Family >  3.50
|   |   |   |   |   |   |   |--- ZIPCode_92507 <= 0.50
|   |   |   |   |   |   |   |   |--- weights: [0.45, 0.00] class: 0
|   |   |   |   |   |   |   |--- ZIPCode_92507 >  0.50
|   |   |   |   |   |   |   |   |--- weights: [0.15, 0.00] class: 0
|   |   |   |   |   |--- Age >  36.50
|   |   |   |   |   |   |--- ZIPCode_91203 <= 0.50
|   |   |   |   |   |   |   |--- Online <= 0.50
|   |   |   |   |   |   |   |   |--- weights: [2.70, 0.00] class: 0
|   |   |   |   |   |   |   |--- Online >  0.50
|   |   |   |   |   |   |   |   |--- weights: [3.90, 0.00] class: 0
|   |   |   |   |   |   |--- ZIPCode_91203 >  0.50
|   |   |   |   |   |   |   |--- weights: [0.00, 0.85] class: 1
|   |   |   |   |--- Income >  81.50
|   |   |   |   |   |--- ID <= 934.50
|   |   |   |   |   |   |--- weights: [1.35, 0.00] class: 0
|   |   |   |   |   |--- ID >  934.50
|   |   |   |   |   |   |--- CCAvg <= 3.05
|   |   |   |   |   |   |   |--- weights: [0.60, 0.00] class: 0
|   |   |   |   |   |   |--- CCAvg >  3.05
|   |   |   |   |   |   |   |--- Securities_Account <= 0.50
|   |   |   |   |   |   |   |   |--- ZIPCode_95616 <= 0.50
|   |   |   |   |   |   |   |   |   |--- Mortgage <= 173.00
|   |   |   |   |   |   |   |   |   |   |--- ID <= 3334.00
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 4
|   |   |   |   |   |   |   |   |   |   |--- ID >  3334.00
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [0.00, 5.95] class:
1
|   |   |   |   |   |   |   |   |   |--- Mortgage >  173.00
|   |   |   |   |   |   |   |   |   |   |--- Age <= 53.00
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [0.15, 0.00] class:
0
|   |   |   |   |   |   |   |   |   |   |--- Age >  53.00
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [0.15, 0.00] class:
0
|   |   |   |   |   |   |   |   |--- ZIPCode_95616 >  0.50
|   |   |   |   |   |   |   |   |   |--- Education_3 <= 0.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [0.15, 0.00] class: 0
|   |   |   |   |   |   |   |   |   |--- Education_3 >  0.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [0.30, 0.00] class: 0
|   |   |   |   |   |   |   |--- Securities_Account >  0.50
|   |   |   |   |   |   |   |   |--- CreditCard <= 0.50
|   |   |   |   |   |   |   |   |   |--- weights: [0.30, 0.00] class: 0
|   |   |   |   |   |   |   |   |--- CreditCard >  0.50
```

```
|   |   |   |   |   |   |   |   |   |--- weights: [0.15, 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
|   |--- Family <= 2.50
|   |   |--- Education_3 <= 0.50
|   |   |   |--- Education_2 <= 0.50
|   |   |   |   |--- Income <= 100.00
|   |   |   |   |   |--- CCAvg <= 4.20
|   |   |   |   |   |   |--- weights: [0.45, 0.00] class: 0
|   |   |   |   |   |--- CCAvg >  4.20
|   |   |   |   |   |   |--- CreditCard <= 0.50
|   |   |   |   |   |   |   |--- weights: [0.00, 0.85] class: 1
|   |   |   |   |   |   |--- CreditCard >  0.50
|   |   |   |   |   |   |   |--- weights: [0.00, 0.85] class: 1
|   |   |   |   |--- Income >  100.00
|   |   |   |   |   |--- ZIPCode_91367 <= 0.50
|   |   |   |   |   |   |--- ZIPCode_95138 <= 0.50
|   |   |   |   |   |   |   |--- weights: [66.90, 0.00] class: 0
|   |   |   |   |   |   |--- ZIPCode_95138 >  0.50
|   |   |   |   |   |   |   |--- weights: [0.15, 0.00] class: 0
|   |   |   |   |   |--- ZIPCode_91367 >  0.50
|   |   |   |   |   |   |--- CD_Account <= 0.50
|   |   |   |   |   |   |   |--- weights: [0.15, 0.00] class: 0
|   |   |   |   |   |   |--- CD_Account >  0.50
|   |   |   |   |   |   |   |--- weights: [0.00, 0.85] class: 1
|   |   |   |--- Education_2 >  0.50
|   |   |   |   |--- Income <= 110.00
|   |   |   |   |   |--- weights: [1.80, 0.00] class: 0
|   |   |   |   |--- Income >  110.00
|   |   |   |   |   |--- Income <= 116.50
|   |   |   |   |   |   |--- Mortgage <= 141.50
|   |   |   |   |   |   |   |--- ZIPCode_90025 <= 0.50
|   |   |   |   |   |   |   |   |--- Age <= 48.50
|   |   |   |   |   |   |   |   |   |--- ZIPCode_91125 <= 0.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [0.00, 2.55] class: 1
|   |   |   |   |   |   |   |   |   |--- ZIPCode_91125 >  0.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [0.15, 0.00] class: 0
|   |   |   |   |   |   |   |   |--- Age >  48.50
|   |   |   |   |   |   |   |   |   |--- weights: [0.15, 0.00] class: 0
|   |   |   |   |   |   |   |--- ZIPCode_90025 >  0.50
|   |   |   |   |   |   |   |   |--- weights: [0.15, 0.00] class: 0
|   |   |   |   |   |   |--- Mortgage >  141.50
|   |   |   |   |   |   |   |--- weights: [0.60, 0.00] class: 0
|   |   |   |   |   |--- Income >  116.50
|   |   |   |   |   |   |--- weights: [0.00, 45.05] class: 1
|   |   |--- Education_3 >  0.50
|   |   |   |--- Income <= 116.50
|   |   |   |   |--- CCAvg <= 1.10
|   |   |   |   |   |--- weights: [1.95, 0.00] class: 0
|   |   |   |   |--- CCAvg >  1.10
|   |   |   |   |   |--- ID <= 4505.50
```

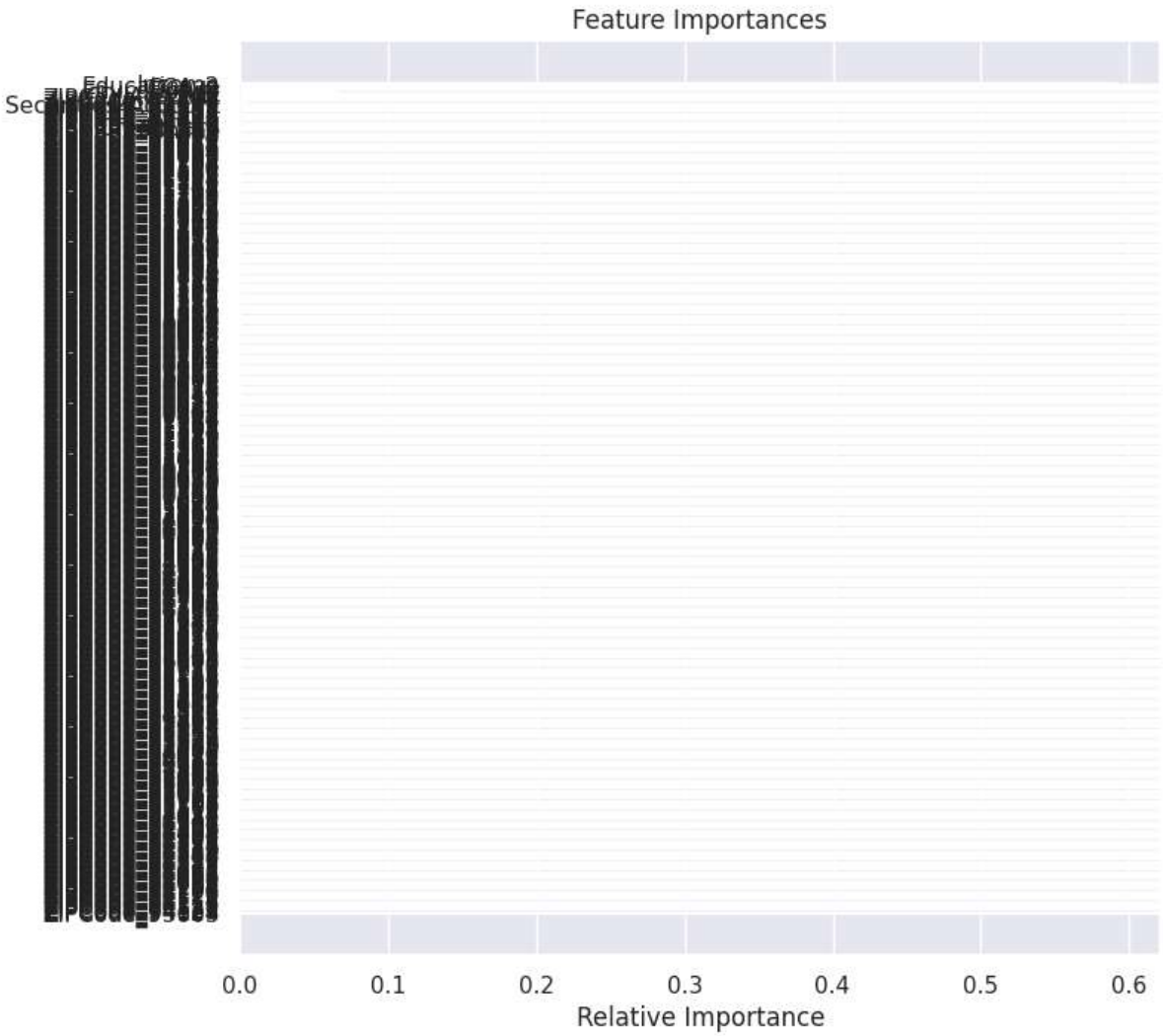
```
|   |   |   |   |   |   |--- CCAvg <= 1.95
|   |   |   |   |   |   |   |--- ID <= 346.50
|   |   |   |   |   |   |   |   |--- weights: [0.00, 0.85] class: 1
|   |   |   |   |   |   |   |--- ID >  346.50
|   |   |   |   |   |   |   |   |--- weights: [0.60, 0.00] class: 0
|   |   |   |   |   |   |--- CCAvg >  1.95
|   |   |   |   |   |   |   |--- ZIPCode_93460 <= 0.50
|   |   |   |   |   |   |   |   |--- ZIPCode_92008 <= 0.50
|   |   |   |   |   |   |   |   |   |--- ZIPCode_90277 <= 0.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [0.00, 5.95] class: 1
|   |   |   |   |   |   |   |   |   |--- ZIPCode_90277 >  0.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [0.15, 0.00] class: 0
|   |   |   |   |   |   |   |   |--- ZIPCode_92008 >  0.50
|   |   |   |   |   |   |   |   |   |--- weights: [0.15, 0.00] class: 0
|   |   |   |   |   |   |   |--- ZIPCode_93460 >  0.50
|   |   |   |   |   |   |   |   |--- weights: [0.15, 0.00] class: 0
|   |   |   |   |   |--- ID >  4505.50
|   |   |   |   |   |   |--- ZIPCode_95035 <= 0.50
|   |   |   |   |   |   |   |--- weights: [0.30, 0.00] class: 0
|   |   |   |   |   |   |--- ZIPCode_95035 >  0.50
|   |   |   |   |   |   |   |--- weights: [0.15, 0.00] class: 0
|   |   |   |--- Income >  116.50
|   |   |   |   |--- weights: [0.00, 52.70] class: 1
|   |--- Family >  2.50
|   |   |--- Income <= 113.50
|   |   |   |--- CCAvg <= 2.75
|   |   |   |   |--- Income <= 106.50
|   |   |   |   |   |--- weights: [3.90, 0.00] class: 0
|   |   |   |   |--- Income >  106.50
|   |   |   |   |   |--- Age <= 28.50
|   |   |   |   |   |   |--- weights: [1.35, 0.00] class: 0
|   |   |   |   |   |--- Age >  28.50
|   |   |   |   |   |   |--- Family <= 3.50
|   |   |   |   |   |   |   |--- weights: [0.90, 0.00] class: 0
|   |   |   |   |   |   |--- Family >  3.50
|   |   |   |   |   |   |   |--- Age <= 60.00
|   |   |   |   |   |   |   |   |--- ZIPCode_95054 <= 0.50
|   |   |   |   |   |   |   |   |   |--- ZIPCode_94305 <= 0.50
|   |   |   |   |   |   |   |   |   |   |--- ZIPCode_94304 <= 0.50
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [0.00, 5.10] class:
1
|   |   |   |   |   |   |   |   |   |   |--- ZIPCode_94304 >  0.50
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [0.15, 0.00] class:
0
|   |   |   |   |   |   |   |   |   |--- ZIPCode_94305 >  0.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [0.15, 0.00] class: 0
|   |   |   |   |   |   |   |   |--- ZIPCode_95054 >  0.50
|   |   |   |   |   |   |   |   |   |--- weights: [0.15, 0.00] class: 0
|   |   |   |   |   |   |   |--- Age >  60.00
|   |   |   |   |   |   |   |   |--- CreditCard <= 0.50
|   |   |   |   |   |   |   |   |   |--- weights: [0.15, 0.00] class: 0
|   |   |   |   |   |   |   |   |--- CreditCard >  0.50
|   |   |   |   |   |   |   |   |   |--- weights: [0.15, 0.00] class: 0
|   |   |   |--- CCAvg >  2.75
|   |   |   |   |--- Age <= 57.00
|   |   |   |   |   |--- ZIPCode_90245 <= 0.50
|   |   |   |   |   |   |--- weights: [0.00, 11.90] class: 1
```

```
|   |   |   |   |   |--- ZIPCode_90245 >  0.50
|   |   |   |   |   |   |--- weights: [0.15, 0.00] class: 0
|   |   |   |   |--- Age >  57.00
|   |   |   |   |   |--- ZIPCode_95123 <= 0.50
|   |   |   |   |   |   |--- weights: [0.60, 0.00] class: 0
|   |   |   |   |   |--- ZIPCode_95123 >  0.50
|   |   |   |   |   |   |--- weights: [0.15, 0.00] class: 0
|   |   |--- Income >  113.50
|   |   |   |--- ZIPCode_94105 <= 0.50
|   |   |   |   |--- Age <= 66.00
|   |   |   |   |   |--- ZIPCode_94608 <= 0.50
|   |   |   |   |   |   |--- Income <= 116.50
|   |   |   |   |   |   |   |--- CCAvg <= 2.50
|   |   |   |   |   |   |   |   |--- weights: [0.30, 0.00] class: 0
|   |   |   |   |   |   |   |--- CCAvg >  2.50
|   |   |   |   |   |   |   |   |--- ZIPCode_90245 <= 0.50
|   |   |   |   |   |   |   |   |   |--- weights: [0.00, 5.10] class: 1
|   |   |   |   |   |   |   |   |--- ZIPCode_90245 >  0.50
|   |   |   |   |   |   |   |   |   |--- weights: [0.15, 0.00] class: 0
|   |   |   |   |   |   |--- Income >  116.50
|   |   |   |   |   |   |   |--- weights: [0.00, 130.90] class: 1
|   |   |   |   |   |--- ZIPCode_94608 >  0.50
|   |   |   |   |   |   |--- weights: [0.15, 0.00] class: 0
|   |   |   |   |--- Age >  66.00
|   |   |   |   |   |--- weights: [0.15, 0.00] class: 0
|   |   |   |--- ZIPCode_94105 >  0.50
|   |   |   |   |--- weights: [0.15, 0.00] class: 0
```

In [86]:
```
# #Create and print a DataFrame of feature importances
# 1. Extract feature importances from the model
# 2. Create a DataFrame with importances and feature names computed as the  Gi
ni importance
# 3. Display the result
print(
pd.DataFrame(
estimator_2.feature_importances_, columns=["Imp"], index=X_train.columns
).sort_values(by="Imp", ascending=False)
)
```

```
                    Imp
Income         0.591625
Education_2    0.136686
CCAvg          0.077851
Education_3    0.065629
Family         0.065575
...                 ...
ZIPCode_92110  0.000000
ZIPCode_92109  0.000000
ZIPCode_92106  0.000000
ZIPCode_92104  0.000000
ZIPCode_93009  0.000000

[478 rows x 1 columns]
```

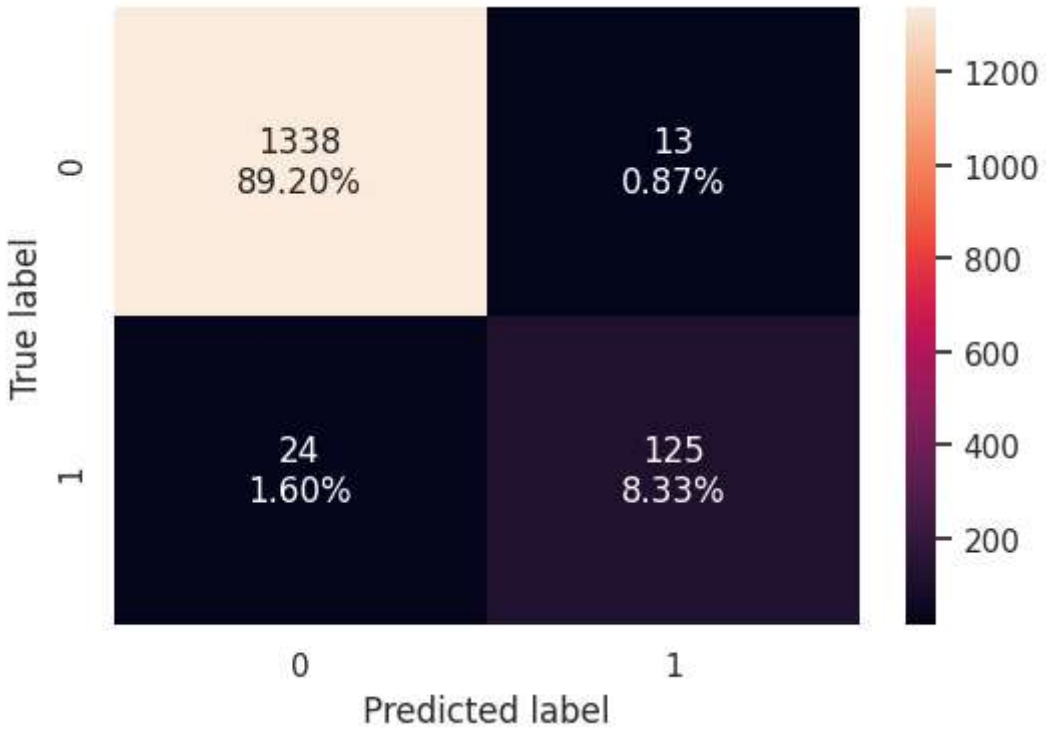
In [87]:
```
# Calculate feature importances from the trained estimator
importances = estimator_2.feature_importances_
# Sort feature indices based on their importances
indices = np.argsort(importances)
# Create a figure for plotting with a specific size
plt.figure(figsize=(8, 8))
# Set the title of the plot
plt.title("Feature Importances")
# Create a horizontal bar plot to visualize feature importances
plt.barh(range(len(indices)), importances[indices], color="violet", align="cen
ter")
# Set y-axis tick labels to display feature names corresponding to their indic
es
plt.yticks(range(len(indices)), [feature_names[i] for i in indices])
# Set the label for the x-axis
plt.xlabel("Relative Importance")
# Display the plot
plt.show()
```



Checking performance on the test data

In [88]:
```
# Compute and display the confusion matrix for the test set
confusion_matrix_sklearn(estimator_2, X_test, y_test)
```

Out[88]:
```
array([[1338,   13],
       [  24,  125]])
```



In [93]:
```
# Compute the performance metrics for the test set
decision_tree_perf_test = model_performance_classification_sklearn(estimator_
2, X_test, y_test)
# Display the performance metrics
decision_tree_perf_test
```

Out[93]:

	Accuracy	Recall	Precision	F1
0	0.975333	0.838926	0.905797	0.87108

Model Performance Comparison and Final Model Selection

In [94]:
```
# Training performance comparison
# Combine the performance metrics of the original decision tree and the tuned
(pre-pruned) decision tree
models_train_comp_df = pd.concat(
[decision_tree_perf_train.T, decision_tree_tune_perf_train.T], axis=1,
)
# Set column names for the combined dataframe
models_train_comp_df.columns = ["Decision Tree sklearn", "Decision Tree (Pre-P
runing)"]
# Print a header for the comparison
print("Training performance comparison:")
# Display the comparison dataframe
models_train_comp_df
```

```
Training performance comparison:
```

Out[94]:

	Decision Tree sklearn	Decision Tree (Pre-Pruning)
Accuracy	1.0	0.987714
Recall	1.0	0.873112
Precision	1.0	0.996552
F1	1.0	0.930757

In [96]:
```
# Compare the testing performance of the original decision tree and the tuned
(pre-pruned) decision tree
models_test_comp_df = pd.concat(
[decision_tree_perf_test.T, decision_tree_tune_perf_test.T], axis=1,)
# Set column names for the combined dataframe
models_test_comp_df.columns = ["Decision Tree sklearn", "Decision Tree (Pre-Pr
uning)"]
# Print a header for the comparison
print("Test performance comparison:")
# Display the comparison dataframe
models_test_comp_df
```

```
Test performance comparison:
```

Out[96]:

	Decision Tree sklearn	Decision Tree (Pre-Pruning)
Accuracy	0.975333	0.978667
Recall	0.838926	0.785235
Precision	0.905797	1.000000
F1	0.871080	0.879699

Expolatory Data Analysis observations

- Income is right-skewed and has many outliers above the 3rd quartile
- Income is right-skewed and has many outliers above the 3rd quartile.
- CCAvg is right-skewed and has many outliers above the 3rd quartile
- Mortgage is heavily right-skewed and there are many outliers above the 3rd quartile in this variable.
- 29.4% of customers have one child, and only 20.2% of customers have three children.
- The majority (42%) of customers have an undergraduate level of education.
- 89.6% of customers do not have a securities account
- 94% of customers do not have a CD account
- 59% of users do their banking online
- 70% of customers do not have a credit card
- The majority of customers (29.4%) live in area 94
- Very few customers (0.8%) live in area 96
- There is an extremely strong correlation (0.99) between age and experience.
- There is a strong correlation (.65) between income and credit card average.
- 90% of customers do not have a personal loan
- Customers with advanced/professional degrees are about three times more likely to have a personal loan than those with undergraduate degrees.
- Education of advanced/professional and graduates have similar rates of personal loans, both significantly higher than undergraduate.
- 90.4% of families do not have a personal loan
- Larger families (3 -4 persons) seem to have a higher percentage of personal loans compared to smaller families (1- 2 persons)
- 89.6% of customers do not have a securities account
- There is a slightly higher percentage of securities account holders among those with personal loans (12.5%) than those without (10.22%).
- 93% of customers do not have a CD account.
- There seems to be a higher proportion of individuals with personal loans among those with a CD account than those who do not have a CD account.
- 59.68% of customers use online banking
- Online banking is more popular than traditional banking in this dataset, regardless of loan status
- 70.6% of customers do not have a credit card.
- Credit card ownership rates are similar between those customers with personal loans and those without.
- ZIP code 94 has the highest population (1472, 29.44% of the sample).
- ZIP code 96 has the lowest population (40, 0.8% of the sample).
- Personal loan rates are relatively consistent across most ZIP codes, ranging from 9.37% to 10.31% for ZIP codes 90-95.
- Income, CCAvg, and Mortgage have a significant number of values above the 3rd quartile that are considered outliers. However, these values are legitimate and do not require outlier treatment.
- There is a drop of in personal loans as the customers have more experience.