Project 2: Personal loan Campaign AIML- ML- Project-low_code

Module 2 - by Habiba Mohamed

19/05/2024

Contents / Agenda

Executive Summary

Great Learning
POWER AHEAD

- In my initiative to identify liability customers who are likely to purchase personal loans, I developed a Decision Tree model. Initially, my model exhibited near-perfect accuracy and recall on both training and testing datasets, indicating significant overfitting. To address this, I applied pruning techniques and post-pruning processes, effectively reducing the complexity of the model by limiting the maximum depth and minimum leaf size. This refinement helped improve the model's generalizability, ensuring more reliable predictions on unseen data. The final model strikes a balance between accuracy and robustness, making it a valuable tool for the bank to create targeted marketing campaigns and increase loan conversion rates among depositors.
- After pruning, a drop in training accuracy and recall is expected, but it should improve the model's ability to generalize, leading to better performance on actual unseen data.

Business Problem Overview and Solution Approach

- Defining the problem:

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. Me 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.

Business Info Overview:

Data head info:

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

Data Tail info:

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

Business Info Overview:

Data Info and Data Types before pre-processing some needed features to 'category'

```
<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
```

Business Info Overview:

Statistical summary of data:

	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

Business Solution Approach

- To help AllLife Bank identify liability customers who are likely to purchase personal loans, we propose a data-driven approach using Decision Tree models. First, gather and preprocess customer data, ensuring to handle missing values and perform feature engineering. Conduct exploratory data analysis to understand data distributions and relationships. Split the data into training and testing sets, then train the Decision Tree model. Evaluate model performance using metrics like accuracy, precision, recall, F1 score. Deploy the best-performing model and integrate it into the bank's system for real-time predictions, continuously monitoring its performance. Finally, use model insights to create targeted marketing campaigns aimed at high-probability customers, personalizing offers to increase conversion rates.

Data Preprocessing

Experience Column unique values:

```
array([ 1, 19, 15,  9,  8, 13, 27, 24, 10, 39,  5, 23, 32, 41, 30, 14, 18,
       21, 28, 31, 11, 16, 20, 35,  6, 25,  7, 12, 26, 37, 17,  2, 36, 29,
        3, 22, -1, 34,  0, 38, 40, 33,  4, -2, 42, -3, 43])
```

Experience Column values <0:

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

Then replaced the values of Experience Column values <0:

-1 with 1, -2 with 2, -3 with 3

Education Column unique values:

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

Data Preprocessing, Feature Engineering

- ZIPCode column has 467 unique numbers, Number of unique values if we take first two digits of ZIPCode: 7, then I changed to String [0:2] and type to Category.

And did convert dtypes of the following columns categorical features to 'category':

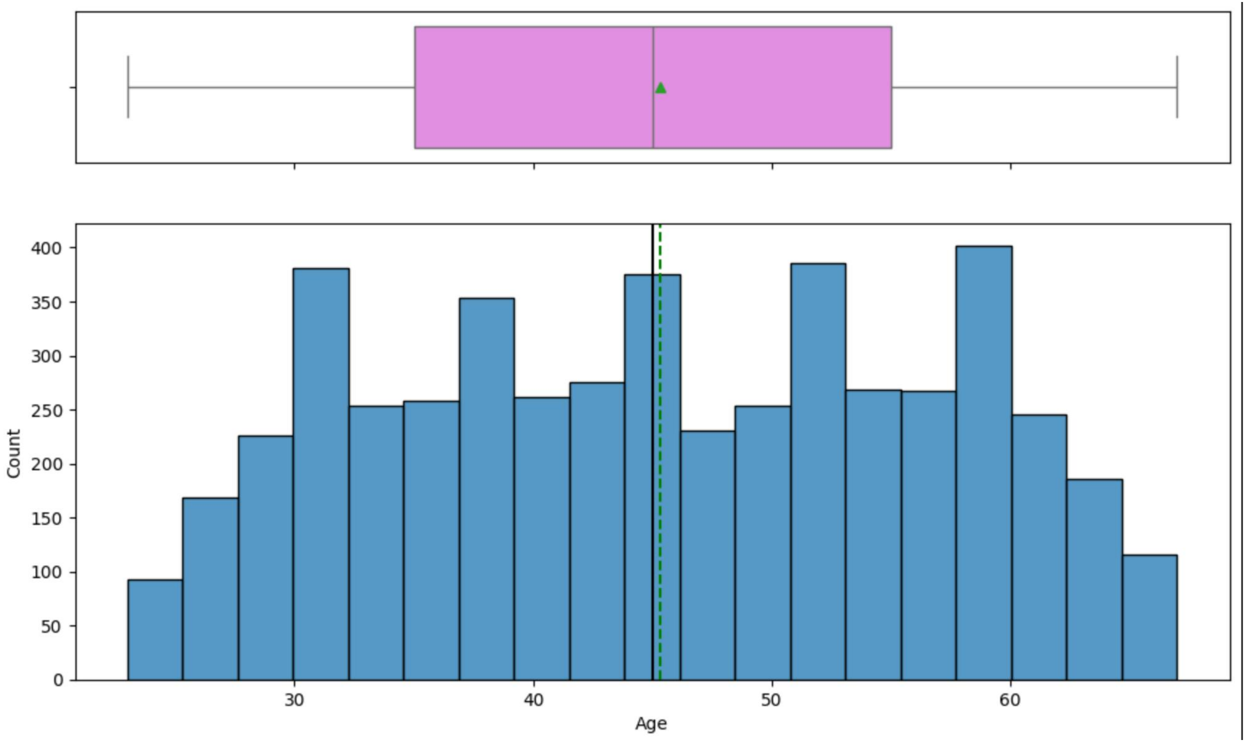
Education, Personal_Loan, Securities_Account, CD_Account, Online, CreditCard, ZIPCode.

-Making the following as the updated Dtypes, with No Missing values along all the data

```
<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   category
 5   Family              5000 non-null   int64
 6   CCAvg               5000 non-null   float64
 7   Education           5000 non-null   category
 8   Mortgage            5000 non-null   int64
 9   Personal_Loan       5000 non-null   category
 10  Securities_Account  5000 non-null   category
 11  CD_Account          5000 non-null   category
 12  Online              5000 non-null   category
 13  CreditCard          5000 non-null   category
dtypes: category(7), float64(1), int64(6)
memory usage: 308.8 KB
```

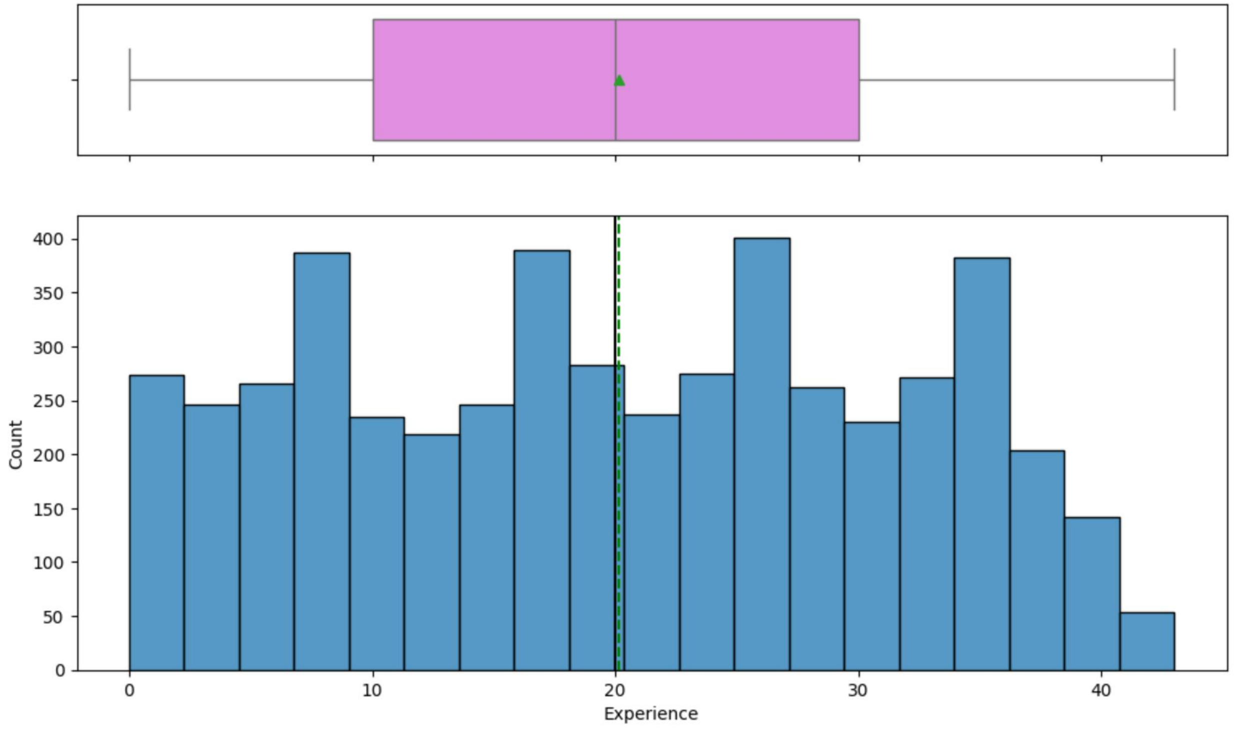
EDA Results

- Age Histogram and boxplot:



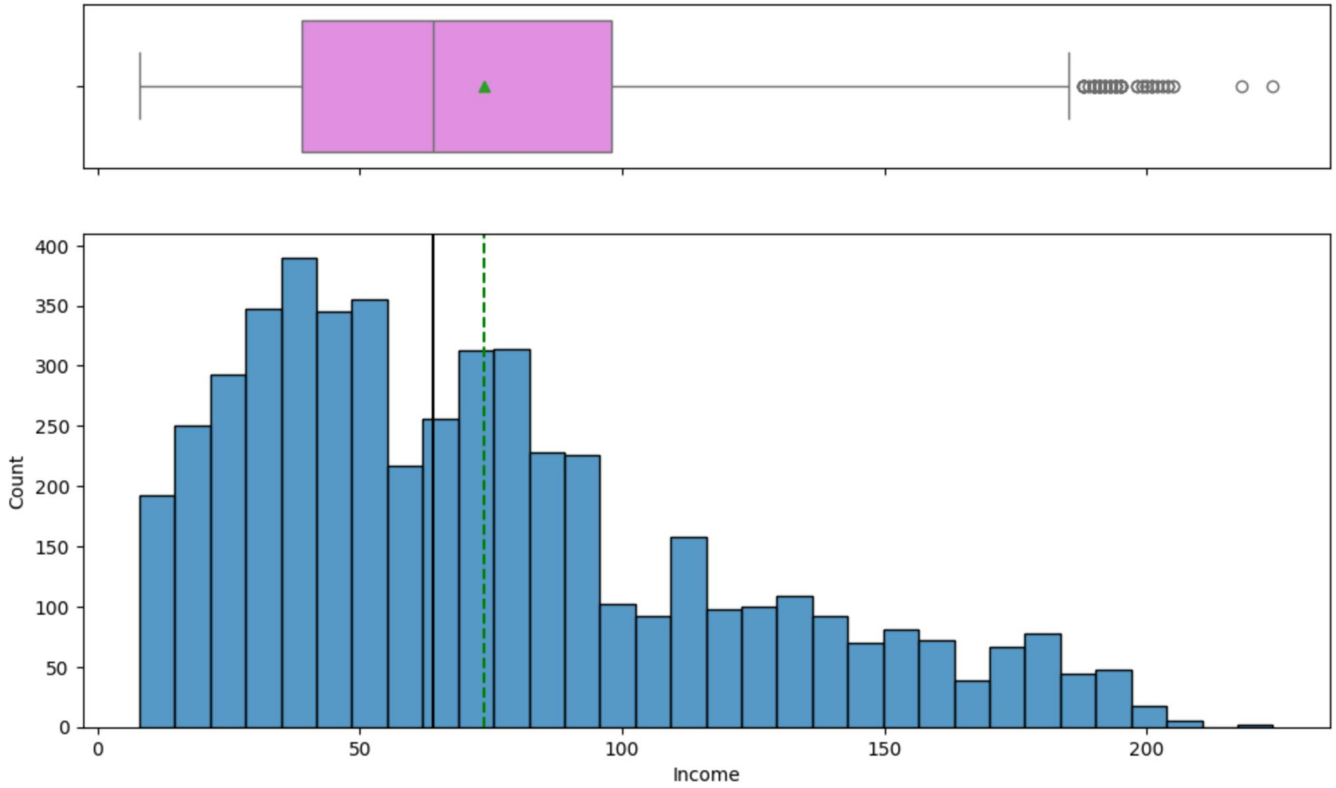
EDA Results

- Experience Histogram and boxplot:



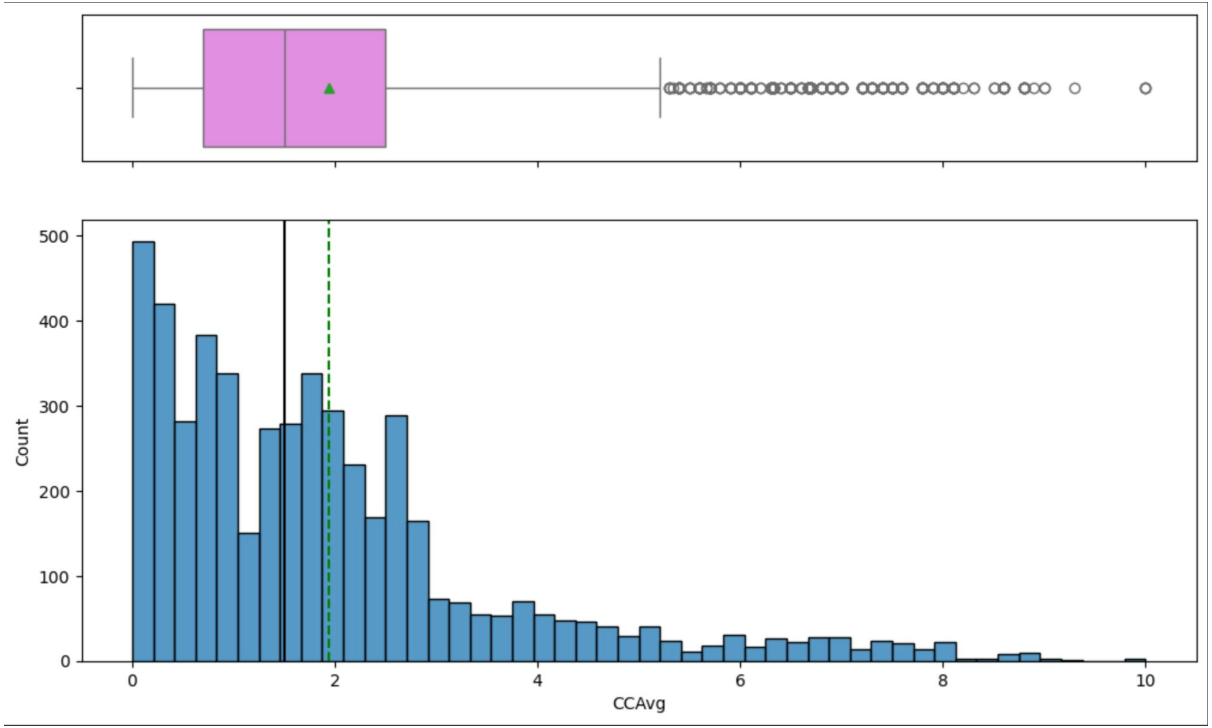
EDA Results

- Income Histogram and Boxplot, having outliers



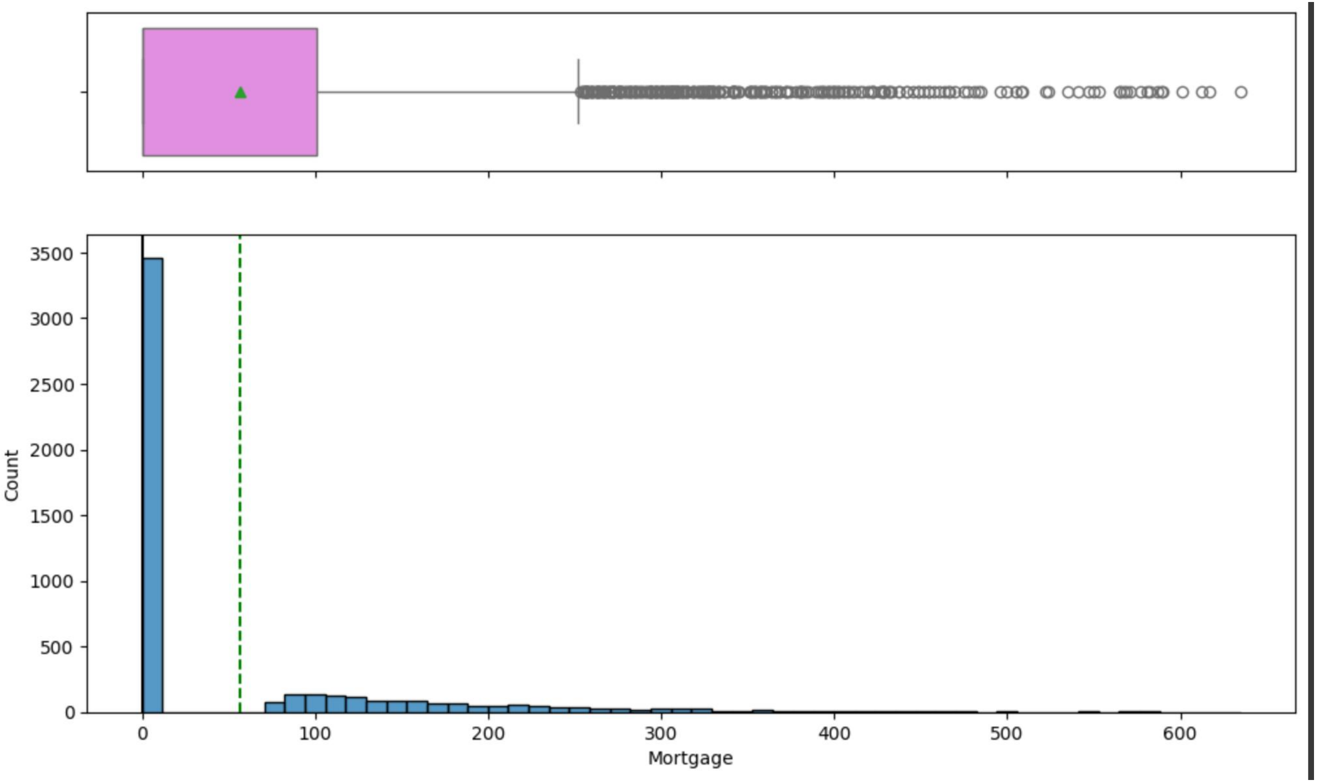
EDA Results

- CCAvg Histogram and boxplot, having Outliers



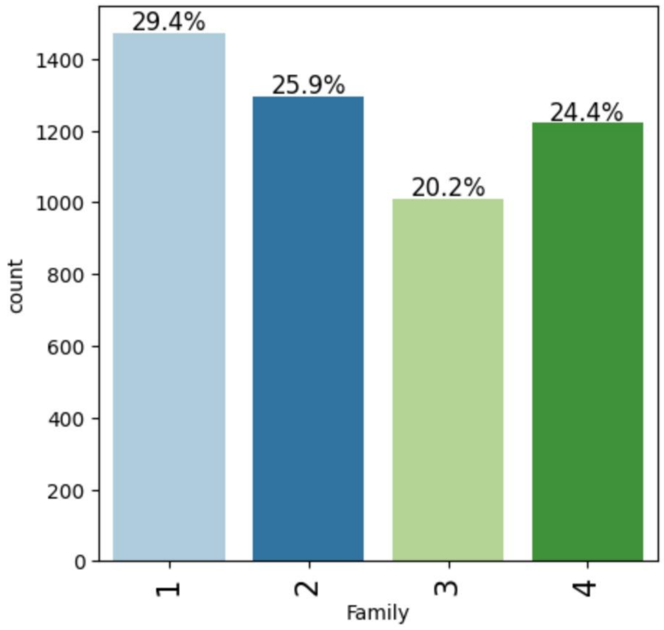
EDA Results

- Mortgage Histogram and boxplot, having Outliers

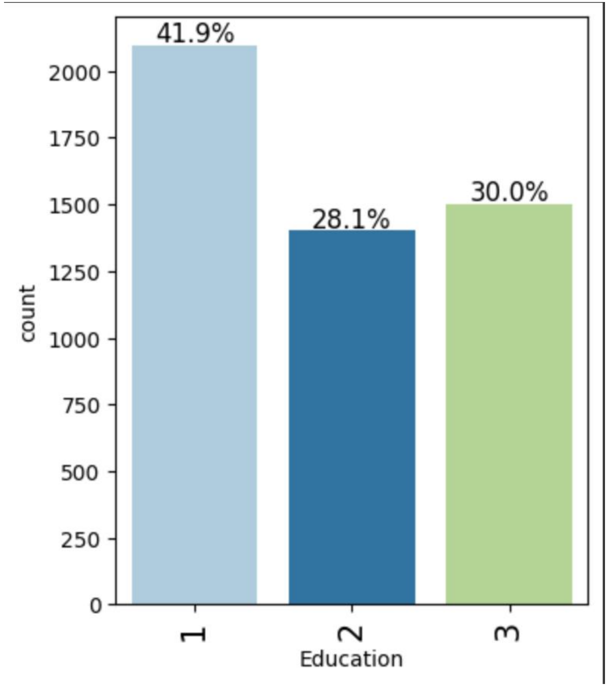


EDA Results

- Family Barplot

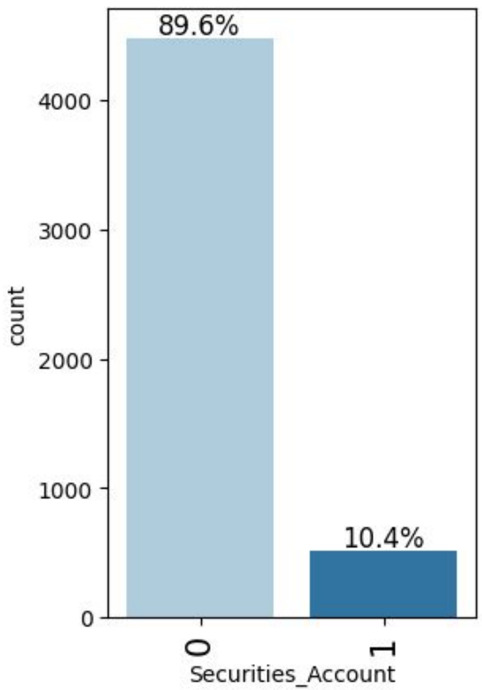


- Education Barplot

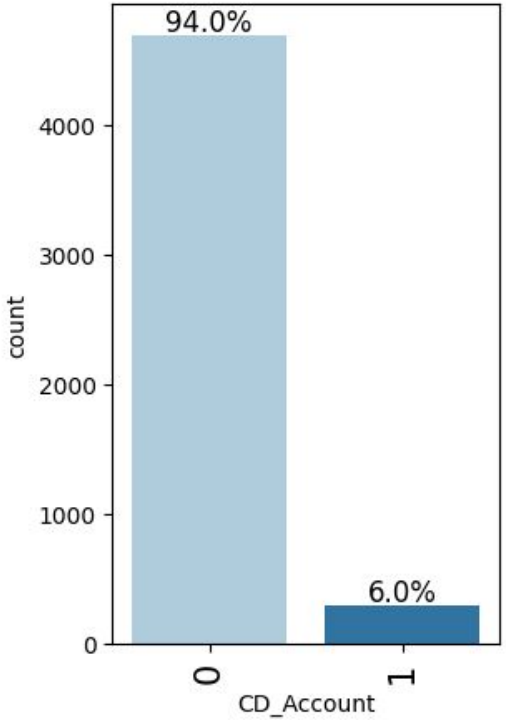


EDA Results

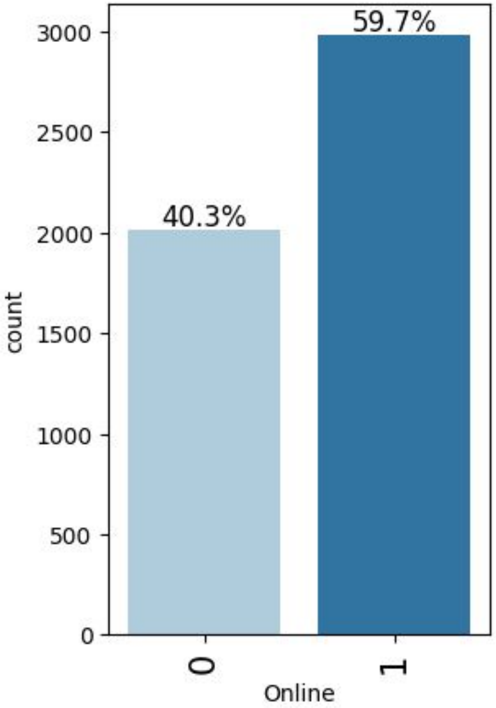
- Securities_Account Barplot



- CD_Account Barplot

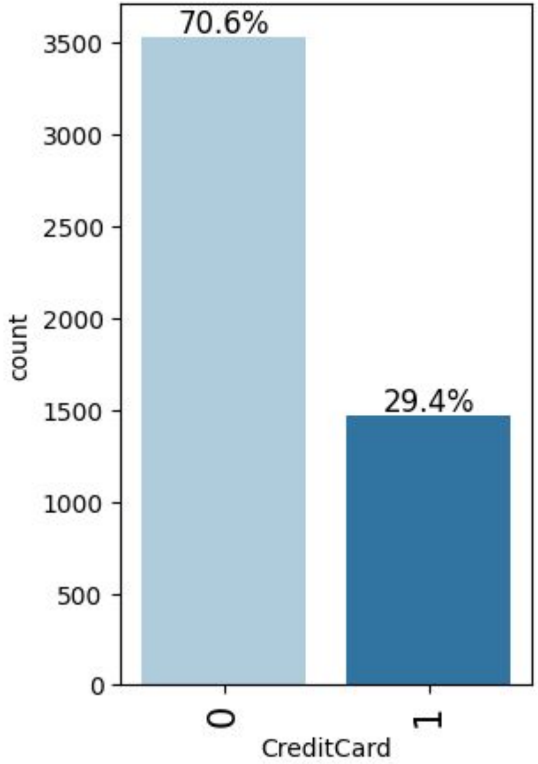


- Online Barplot

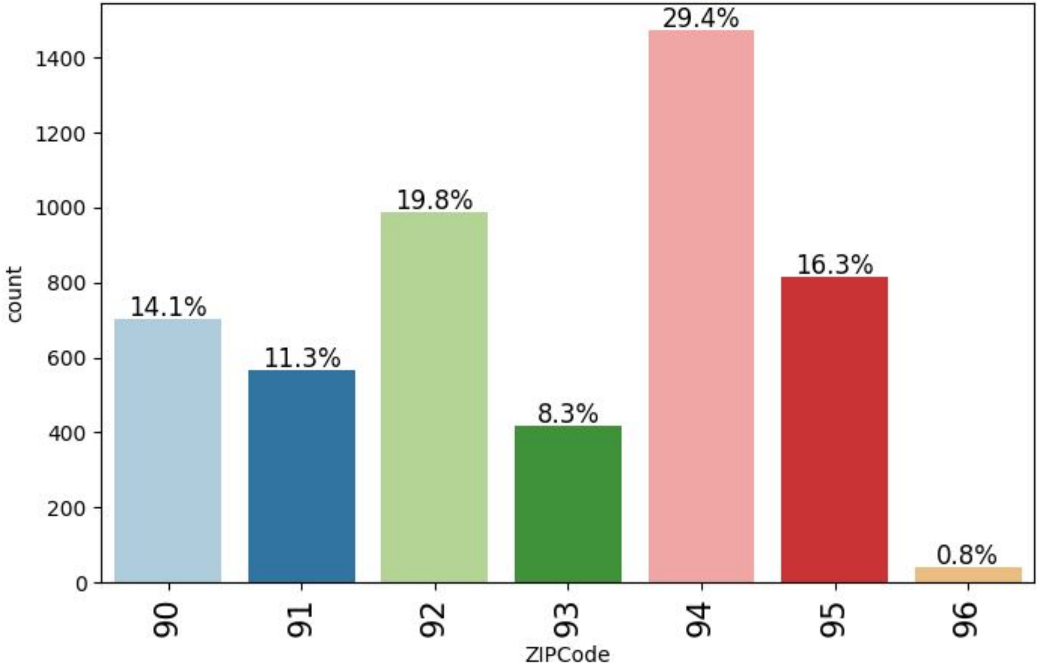


EDA Results

- CreditCard Barplot

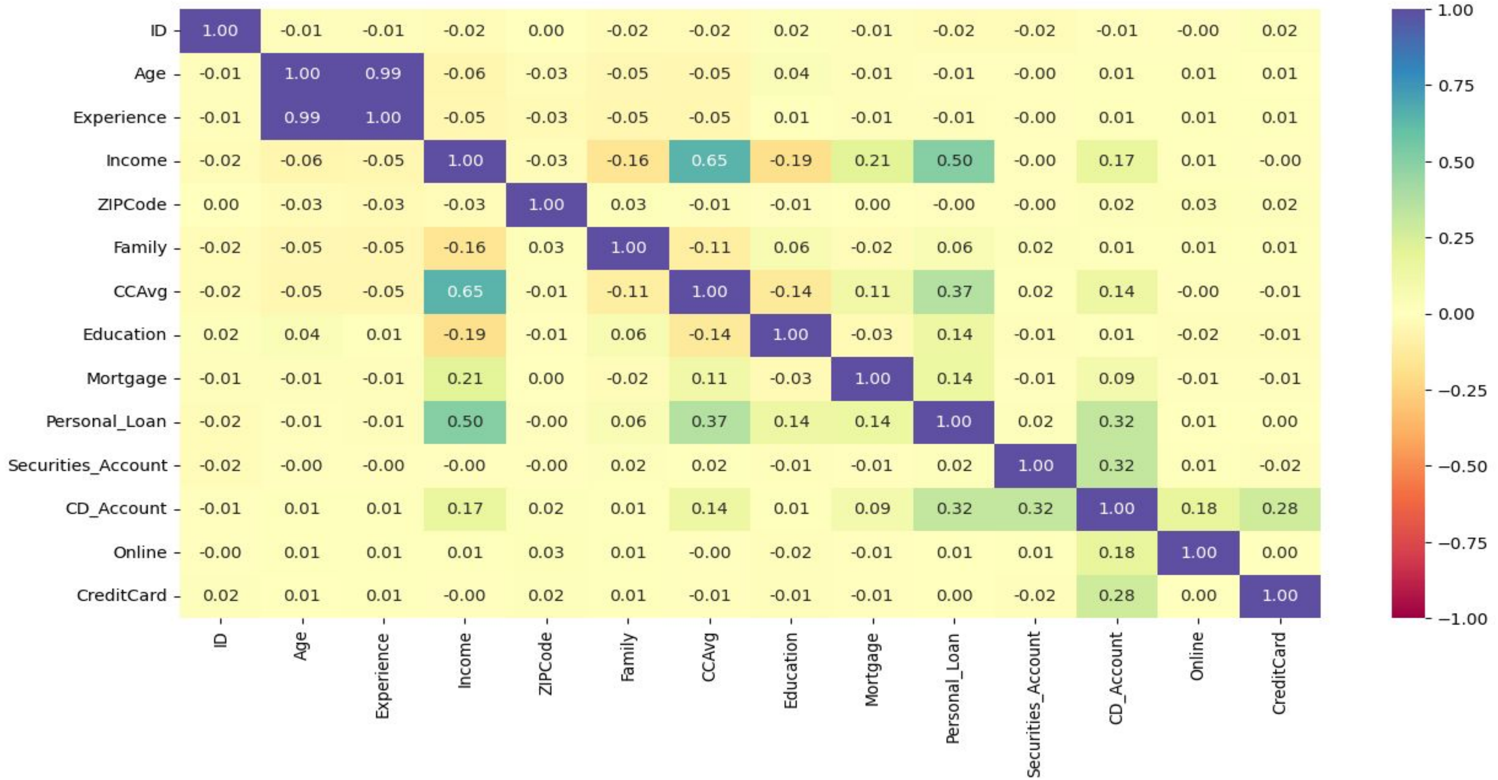


- ZIPCode Barplot



EDA Results, Bivariate analysis

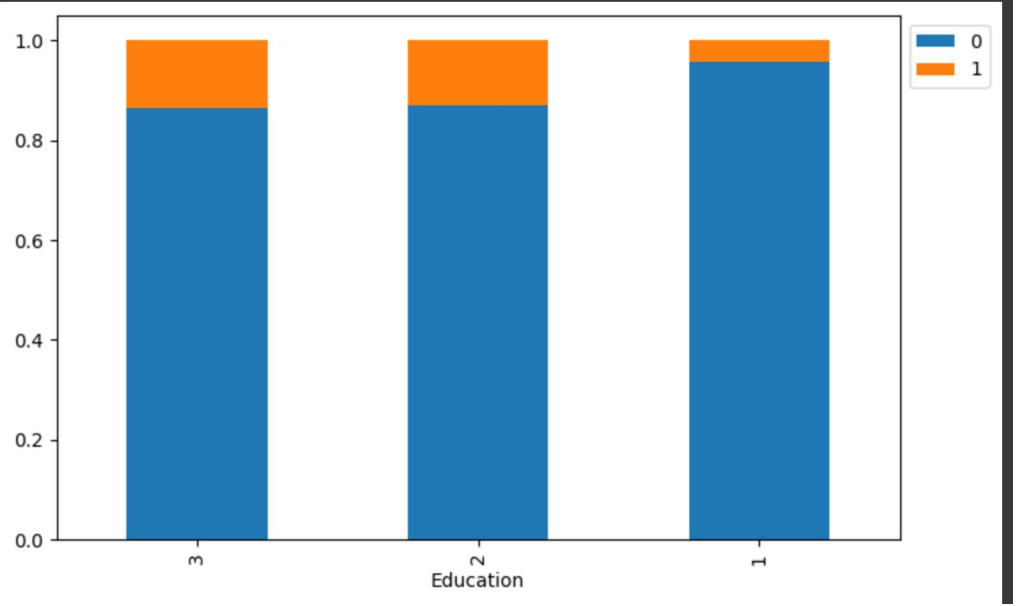
- Heatmap of all the data



EDA Results, Bivariate analysis

- Education vs Personal Loan Barplot:

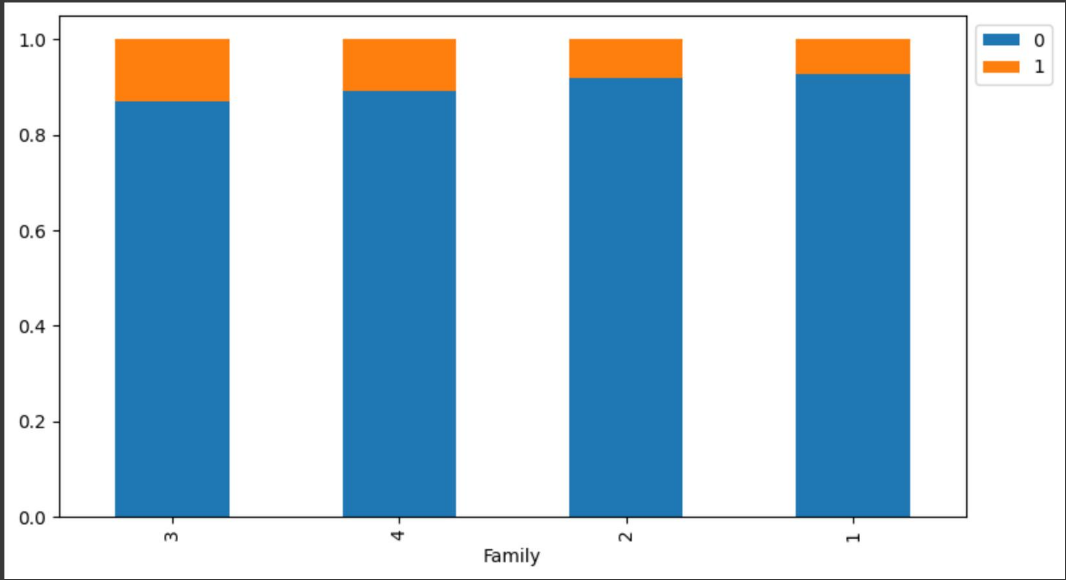
Personal_Loan	0	1	All
Education			
All	4520	480	5000
3	1296	205	1501
2	1221	182	1403
1	2003	93	2096



EDA Results, Bivariate analysis

- Personal Loan vs. Family

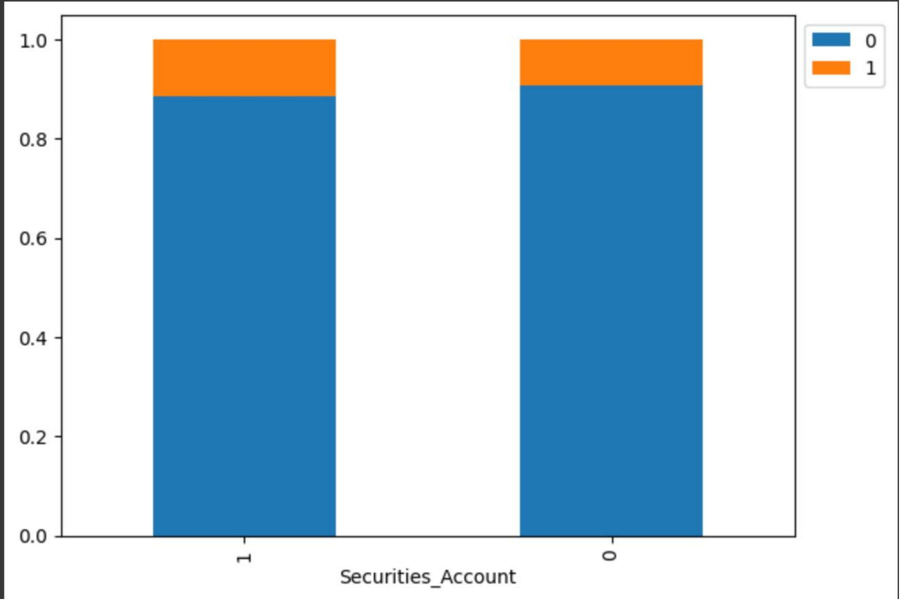
Personal_Loan	0	1	All
Family			
All	4520	480	5000
4	1088	134	1222
3	877	133	1010
1	1365	107	1472
2	1190	106	1296



EDA Results, Bivariate analysis

- Securities Account v s Personal Loan Barplot

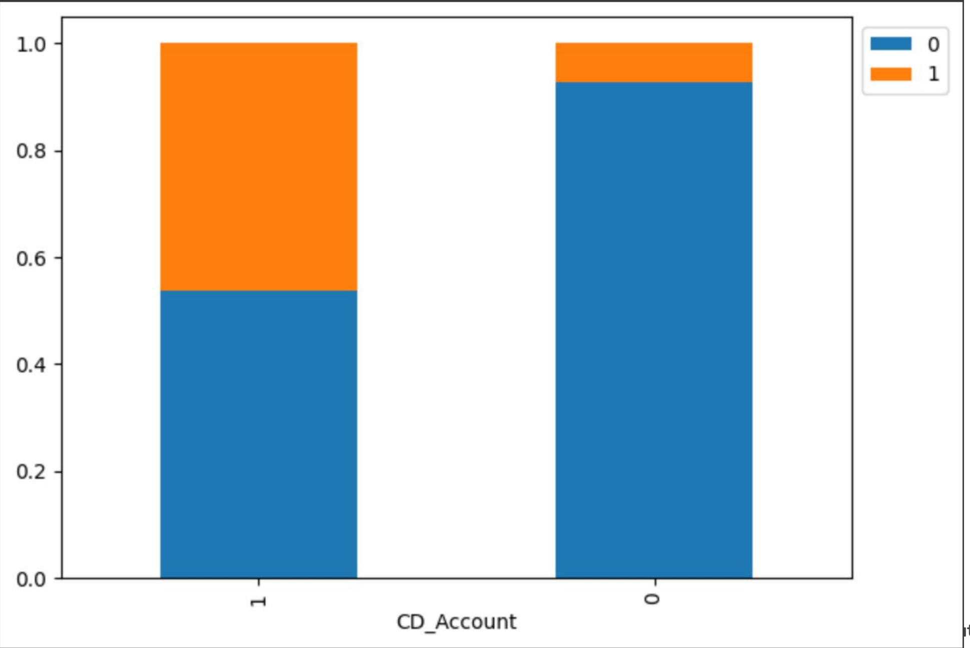
```
Personal_Loan          0    1   All
Securities_Account
All                 4520  480  5000
0                   4058  420  4478
1                    462   60   522
------------------------------------------------------------------------------------------
```



EDA Results, Bivariate analysis

- CD Account Vs Personal Loan Barplot

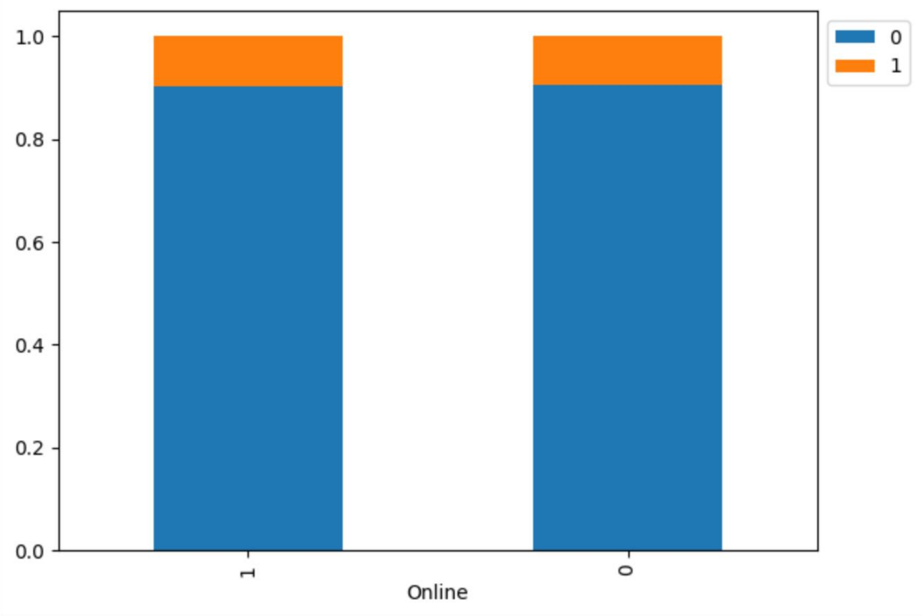
```
Personal_Loan     0     1   All
CD_Account
All            4520   480  5000
0              4358   340  4698
1               162   140   302
------------------------------------------------------------------------------------------------------------------------
```



EDA Results, Bivariate analysis

- Online Vs Personal Loan Barplot

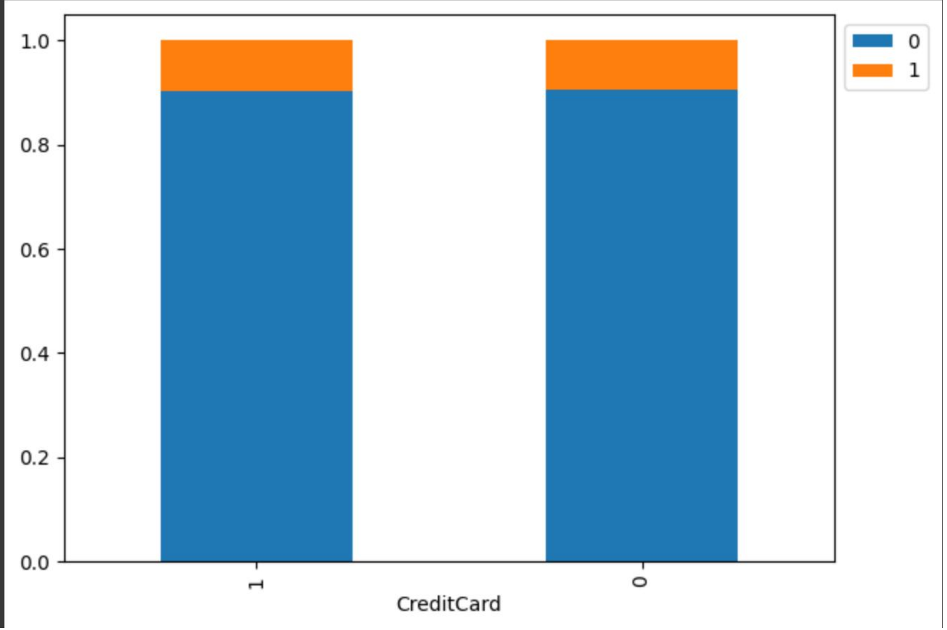
Personal_Loan	0	1	All
Online			
All	4520	480	5000
1	2693	291	2984
0	1827	189	2016



EDA Results, Bivariate analysis

- Credit Card Vs Personal_Loan Barplot

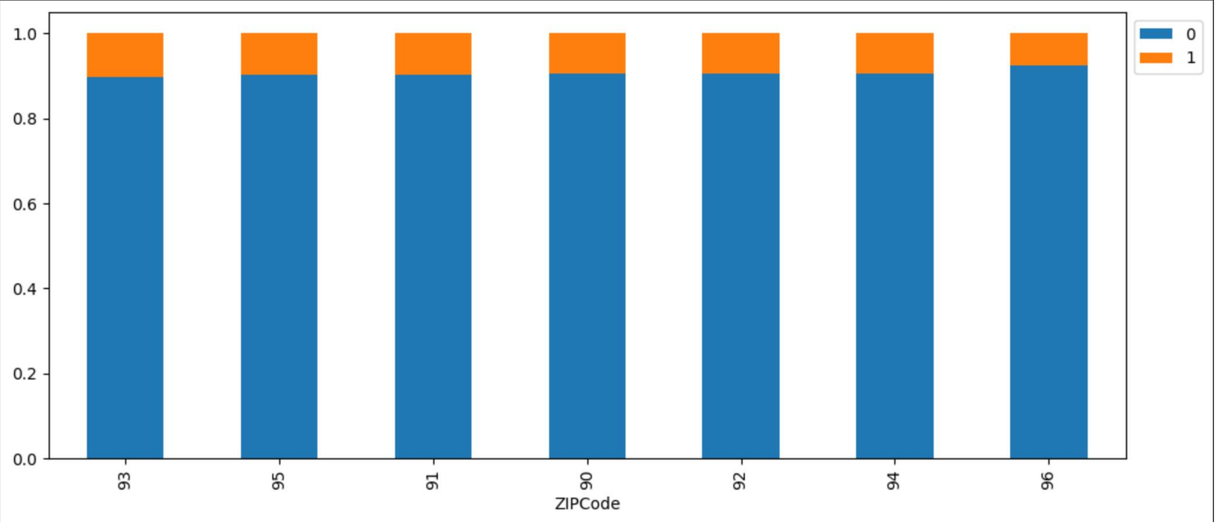
Personal_Loan	0	1	All
CreditCard			
All	4520	480	5000
0	3193	337	3530
1	1327	143	1470



EDA Results, Bivariate analysis

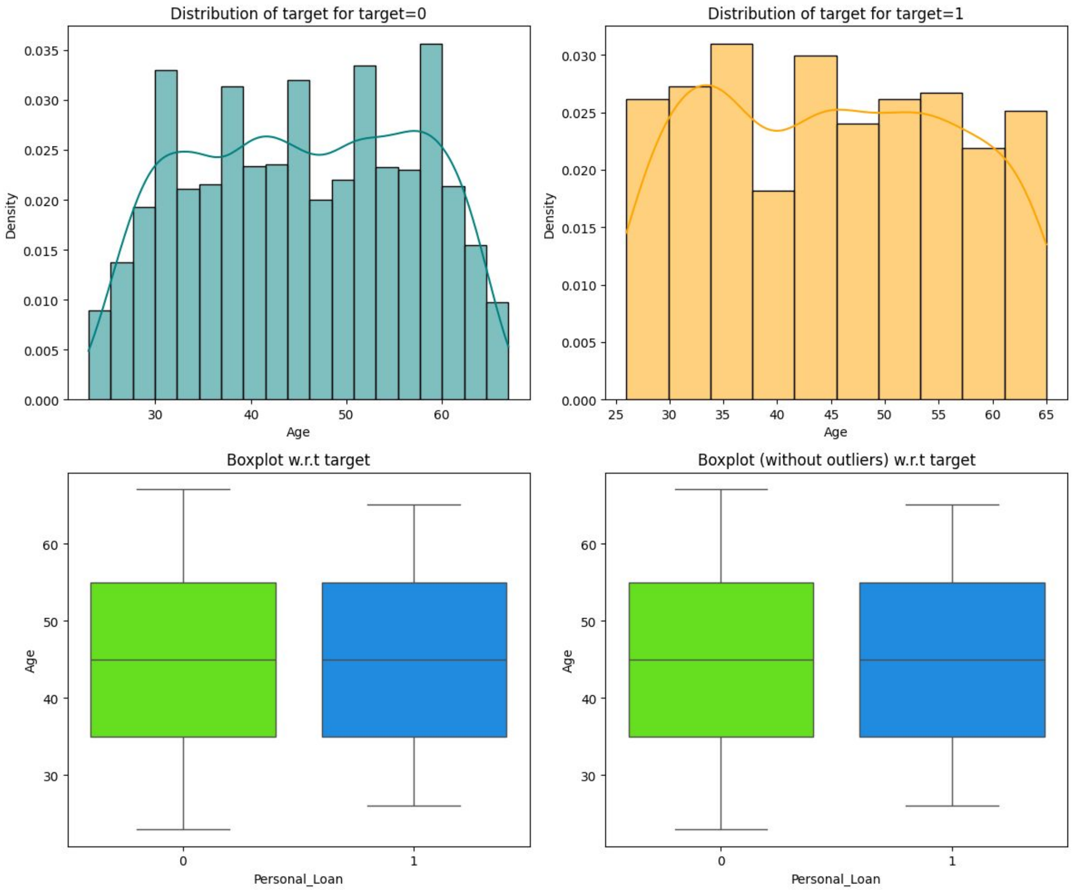
- ZIPCode Vs Personal_Loan Barplot

Personal_Loan	0	1	All
ZIPCode			
All	4520	480	5000
94	1334	138	1472
92	894	94	988
95	735	80	815
90	636	67	703
91	510	55	565
93	374	43	417
96	37	3	40



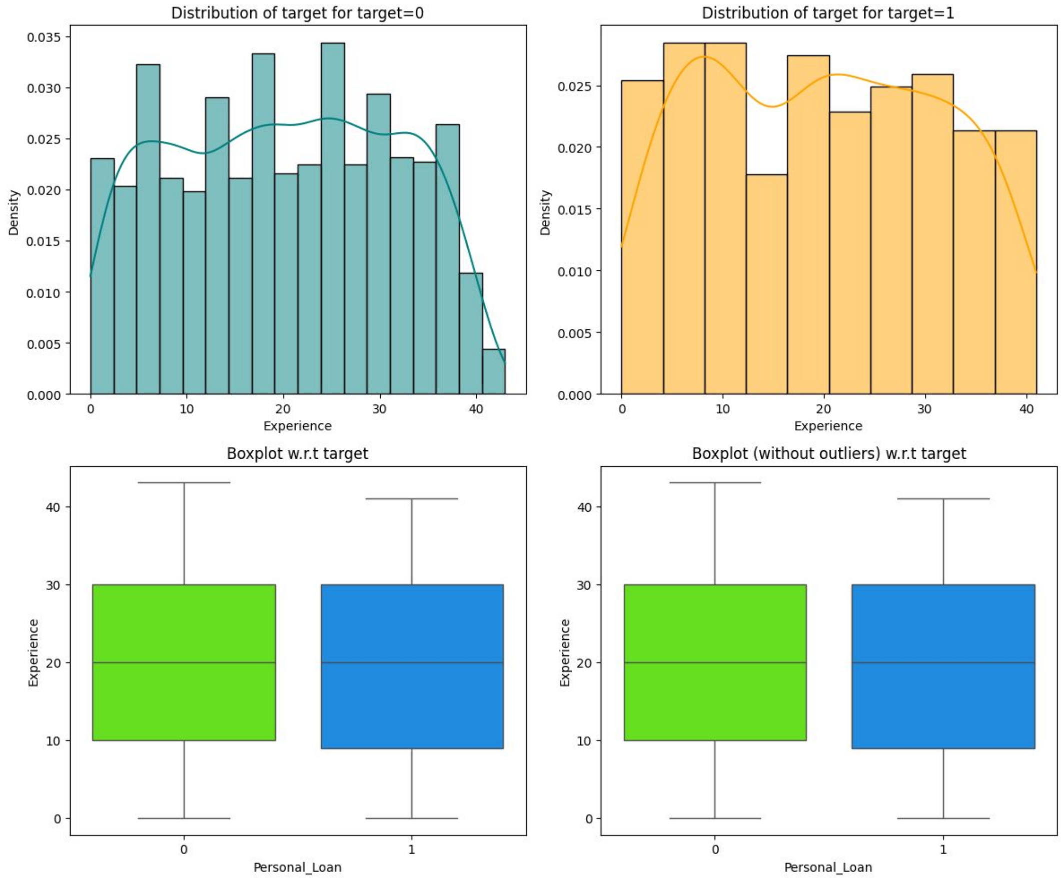
EDA Results, Bivariate analysis

- Distribution Plot + Boxplot of Age Vs. Personal Loan:



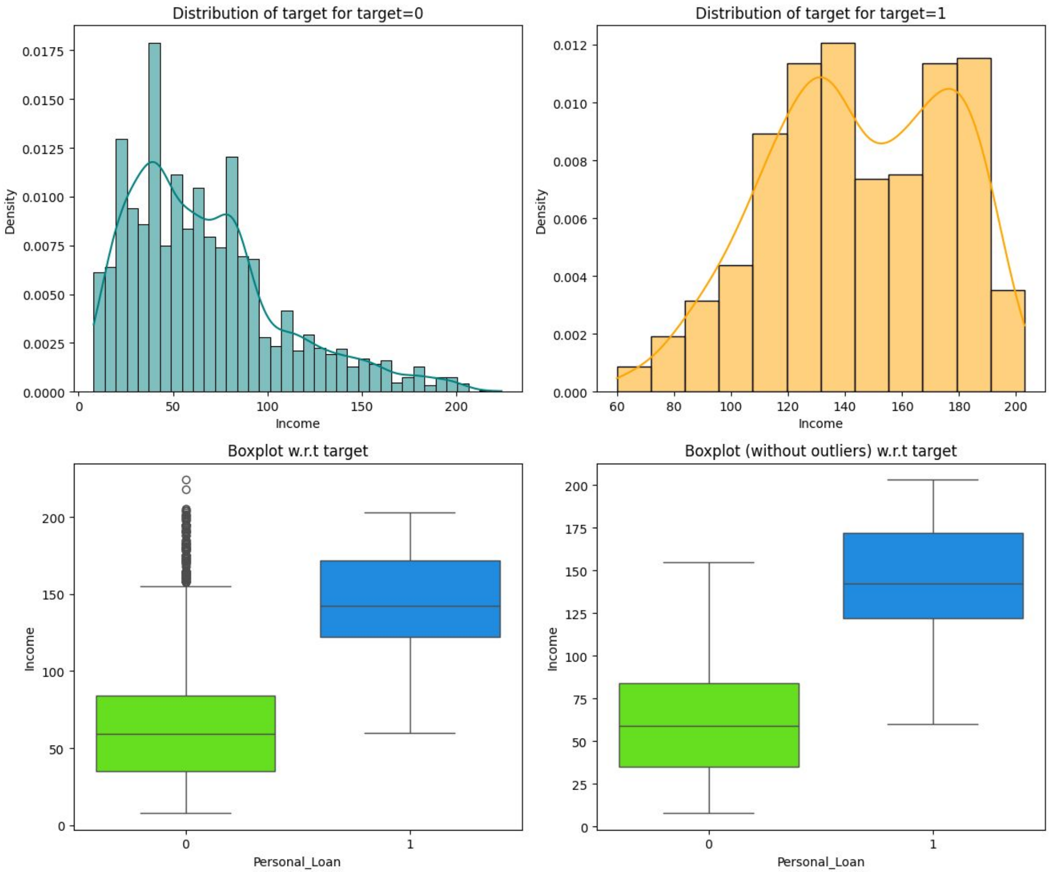
EDA Results, Bivariate analysis

- Distribution Plot + Boxplot of Experience Vs. Personal Loan:



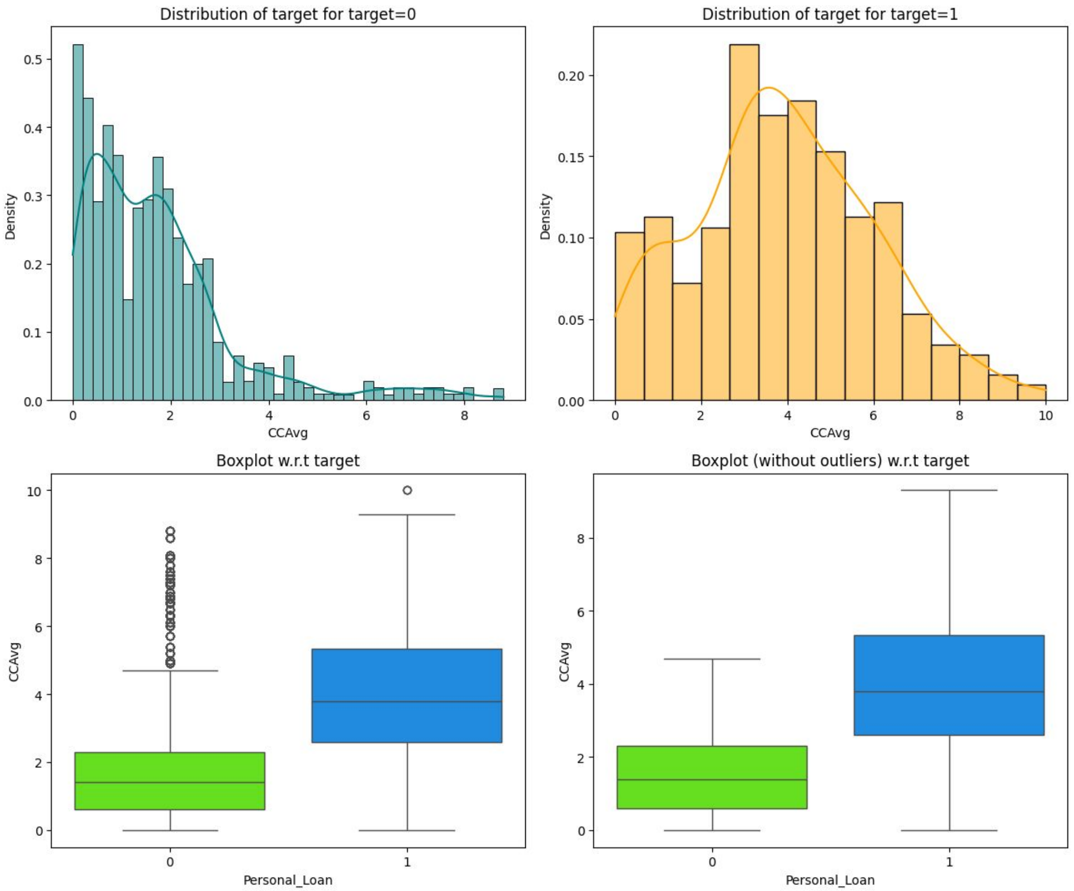
EDA Results, Bivariate analysis

- Distribution Plot + Boxplot of Income Vs. Personal Loan:



EDA Results, Bivariate analysis

- Distribution Plot + Boxplot of CCAvg Vs Personal Loan:



Data Preprocessing for Modelling

- The result of calculating outliers in the data:

```
ID              99.90
Age            100.00
Experience      86.70
Income         100.00
Family           0.00
CCAvg            6.48
Mortgage        30.76
dtype: float64
```

Data Preprocessing for Modelling

- Dropping Experience as it is perfectly correlated to Age.
- Created Dummies for ZIPCode, Education
- Splitted the data in test and training, test size= 0.30, Random State

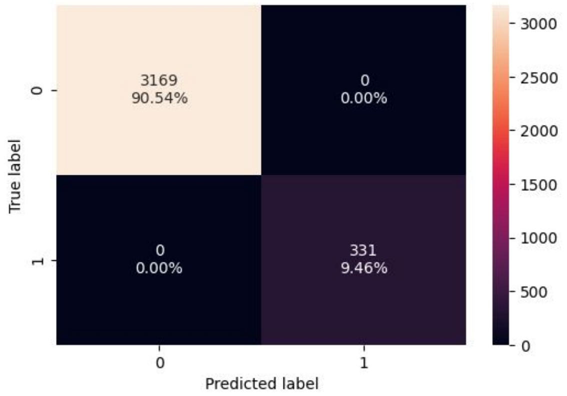
```
Shape of Training set :  (3500, 18)
Shape of test set :  (1500, 18)
Percentage of classes in training set:
Personal_Loan
0    0.905429
1    0.094571
Name: proportion, dtype: float64
Percentage of classes in test set:
Personal_Loan
0    0.900667
1    0.099333
Name: proportion, dtype: float64
```

Model Building

- Created functions to calculate different metrics and confusion matrix so that we don't have to use the same code repeatedly for each model.
- The model_performance_classification_sklearn function will be used to check the model performance of models.
- The confusion_matrix_sklearnfunction will be used to plot confusion matrix.
- Selecting the Gini as Decision Tree Classifier, on a random state

Model Performance Summary

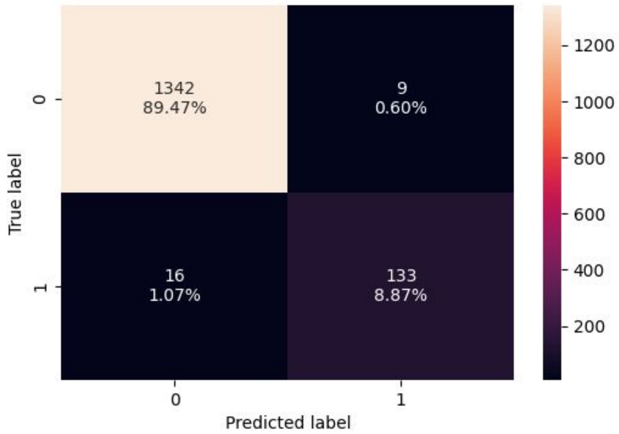
- Checking on model performance in Training data, Confusion Matrix:



- Model evaluation criterion

	Accuracy	Recall	Precision	F1
0	1.0	1.0	1.0	1.0

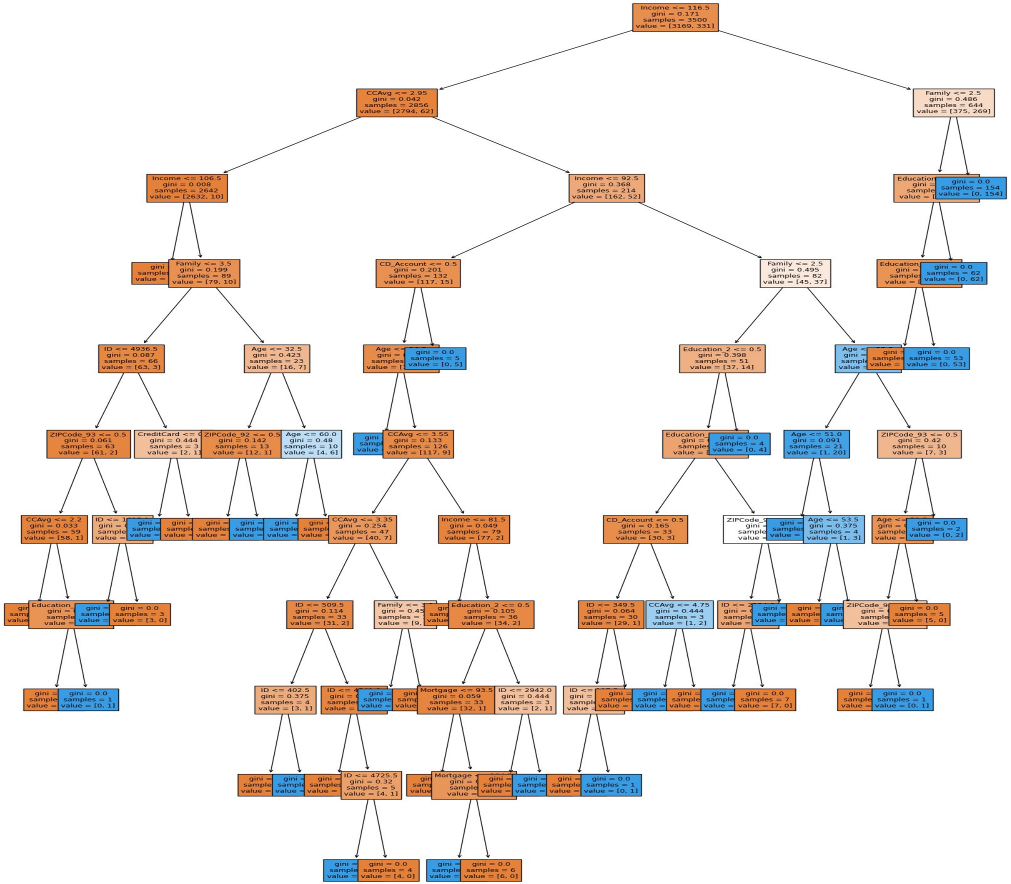
- Checking on model performance in Test data, Confusion Matrix:



- Model evaluation criterion

	Accuracy	Recall	Precision	F1
0	0.983333	0.892617	0.93662	0.914089

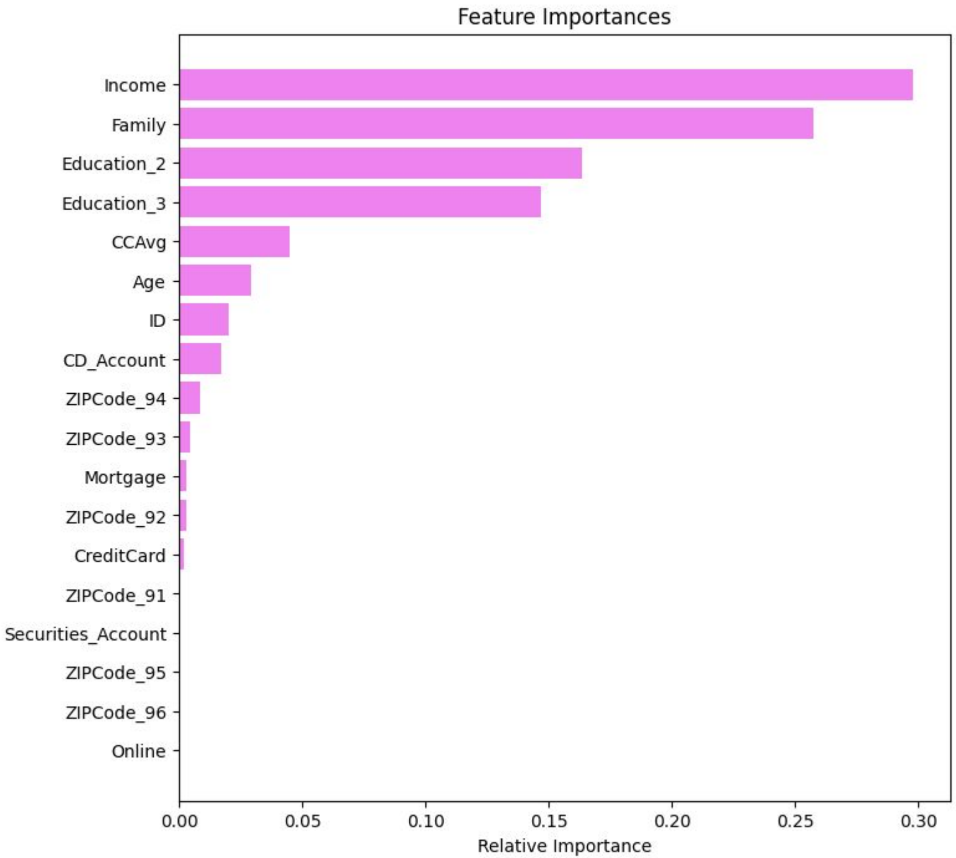
Model Performance Summary: Visualizing Decision Tree



Model Performance Summary:

Model Feature Importance:

	Imp
Income	0.298018
Family	0.257587
Education_2	0.163412
Education_3	0.147127
CCAvg	0.044768
Age	0.029516
ID	0.020281
CD_Account	0.017273
ZIPCode_94	0.008713
ZIPCode_93	0.004766
Mortgage	0.003236
ZIPCode_92	0.003080
CreditCard	0.002224
Online	0.000000
Securities_Account	0.000000
ZIPCode_91	0.000000
ZIPCode_95	0.000000
ZIPCode_96	0.000000

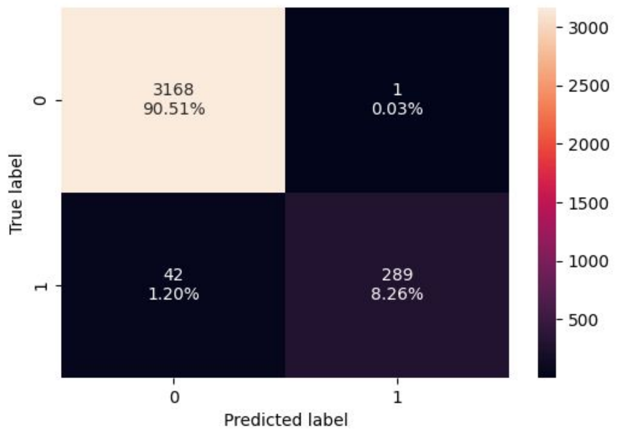


Model Performance Improvement

Pre-Pruning:Pre-Pruning:

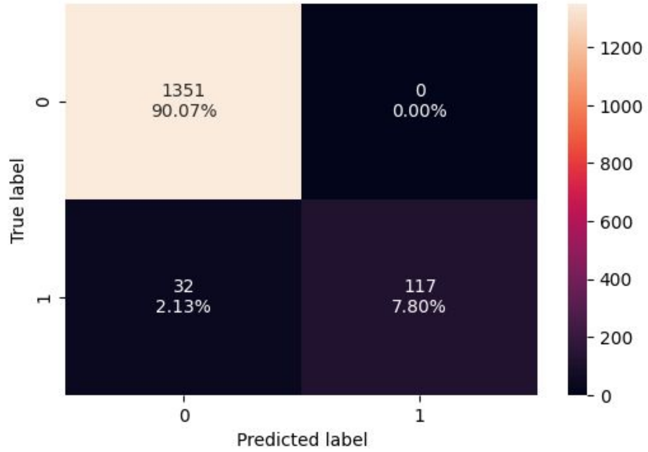
```
▾ DecisionTreeClassifier
DecisionTreeClassifier(max_depth=6, max_leaf_nodes=10, min_samples_leaf=10,
                       random_state=1)
```

- Checking performance on training data



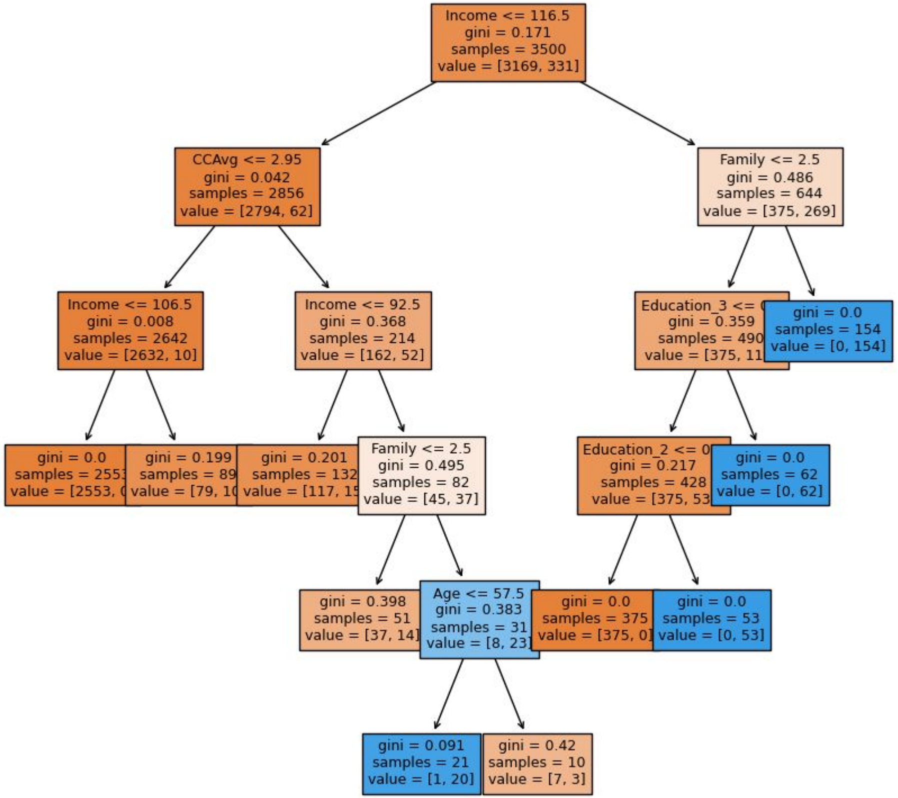
	Accuracy	Recall	Precision	F1
0	0.987714	0.873112	0.996552	0.930757

- Checking performance on test data:



	Accuracy	Recall	Precision	F1
0	0.978667	0.785235	1.0	0.879699

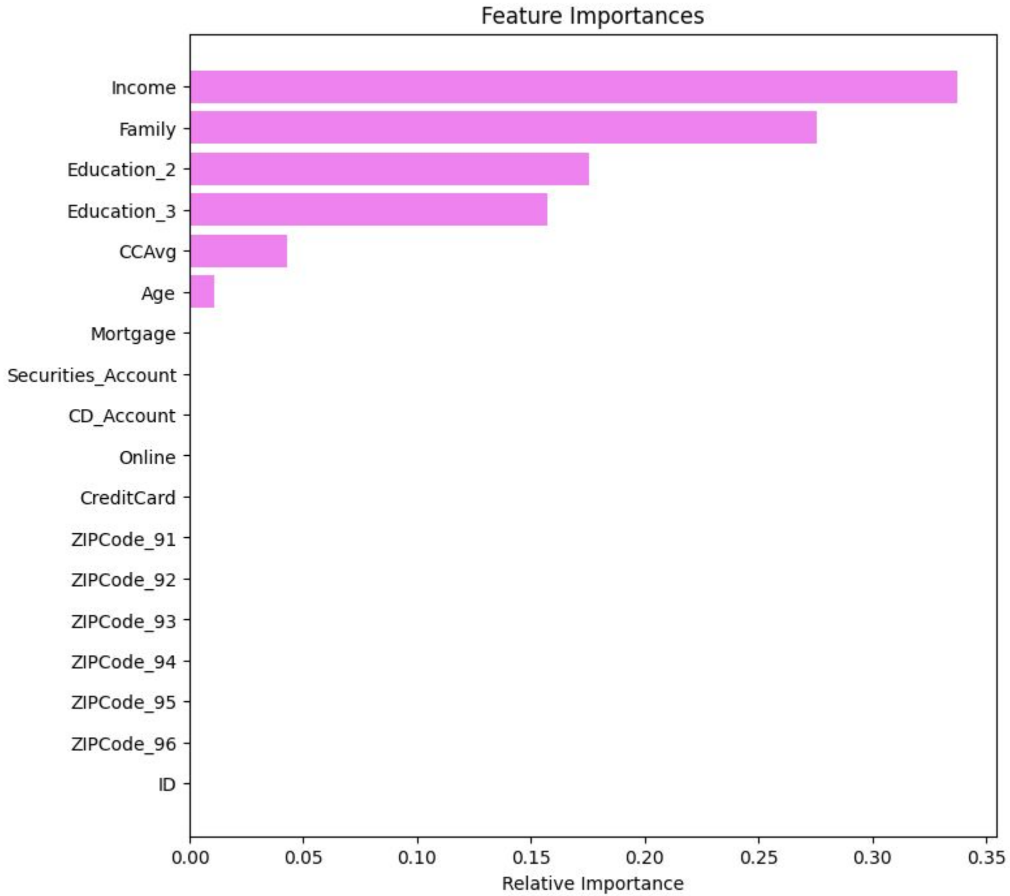
Model Performance Improvement, Visualizing Decision Tree



Model Performance Improvement

Model Feature Importance:

	Imp
Income	0.337681
Family	0.275581
Education_2	0.175687
Education_3	0.157286
CCAvg	0.042856
Age	0.010908
ZIPCode_92	0.000000
ZIPCode_96	0.000000
ZIPCode_95	0.000000
ZIPCode_94	0.000000
ZIPCode_93	0.000000
ID	0.000000
ZIPCode_91	0.000000
Online	0.000000
CD_Account	0.000000
Securities_Account	0.000000
Mortgage	0.000000
CreditCard	0.000000

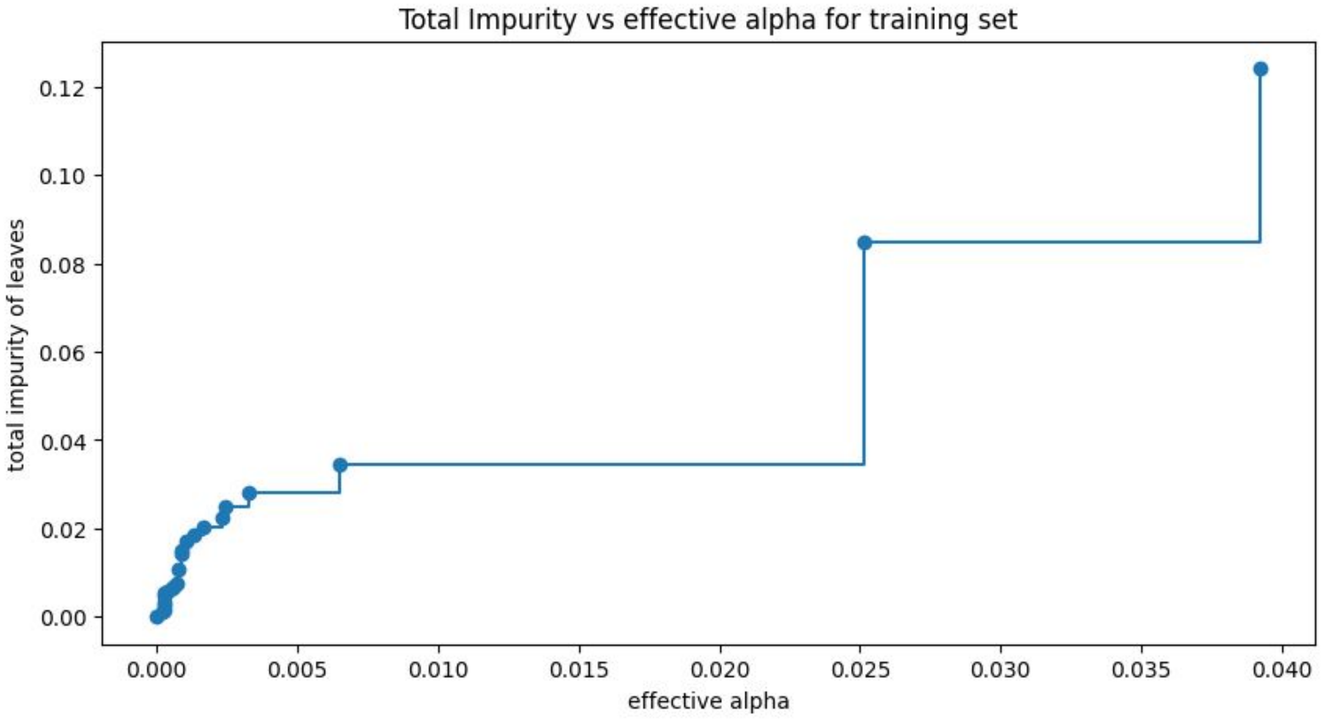


Model Performance Improvement

Cost-Complexity Pruning:

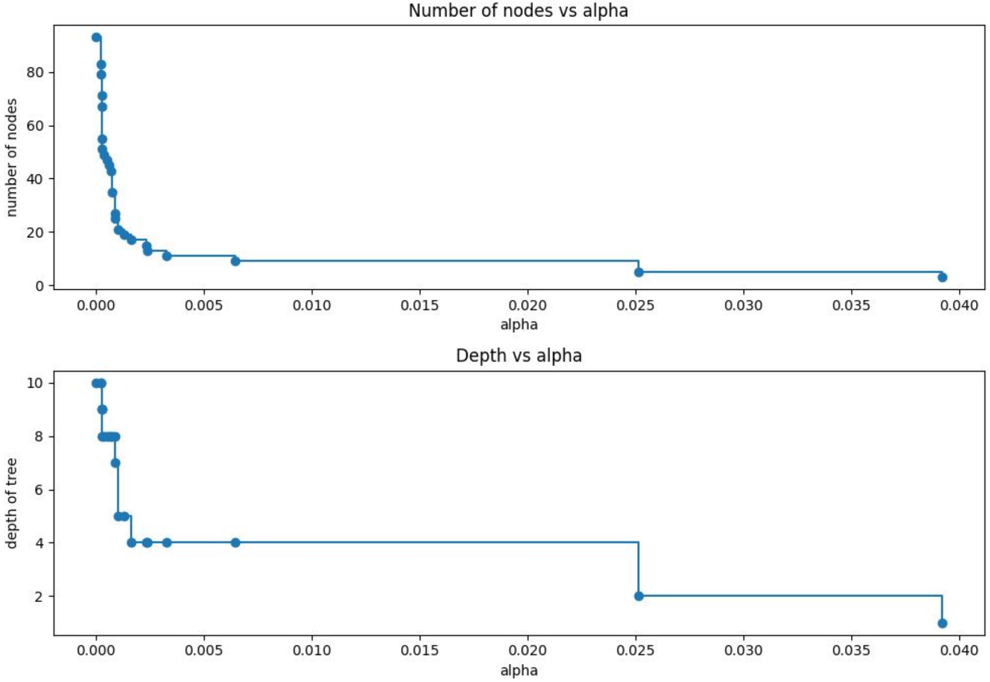
Dataframe Path

	ccp_alphas	impurities
0	0.000000	0.000000
1	0.000223	0.001114
2	0.000250	0.001614
3	0.000268	0.002688
4	0.000272	0.003232
5	0.000273	0.004868
6	0.000276	0.005420
7	0.000381	0.005801
8	0.000527	0.006329
9	0.000625	0.006954
10	0.000700	0.007654
11	0.000769	0.010731
12	0.000882	0.014260
13	0.000889	0.015149
14	0.001026	0.017200
15	0.001305	0.018505
16	0.001647	0.020153
17	0.002333	0.022486
18	0.002407	0.024893
19	0.003294	0.028187
20	0.006473	0.034659
21	0.025146	0.084951
22	0.039216	0.124167
23	0.047088	0.171255

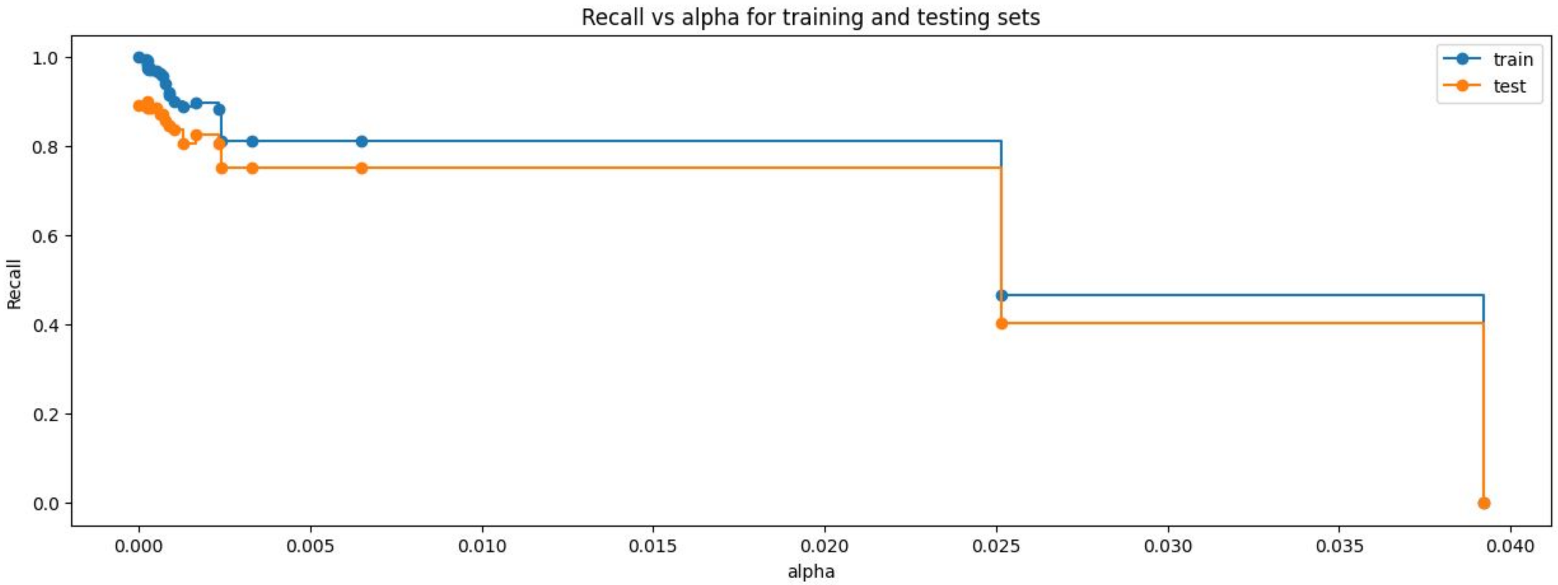


Model Performance Improvement

- Next, we train a decision tree using effective alphas. The last value in ccp_alphas is the alpha value that prunes the whole tree, leaving the tree, clfs[-1], with one node.
- Number of nodes in the last tree is: 1 with ccp_alpha: 0.04708834100596766



Model Performance Improvement

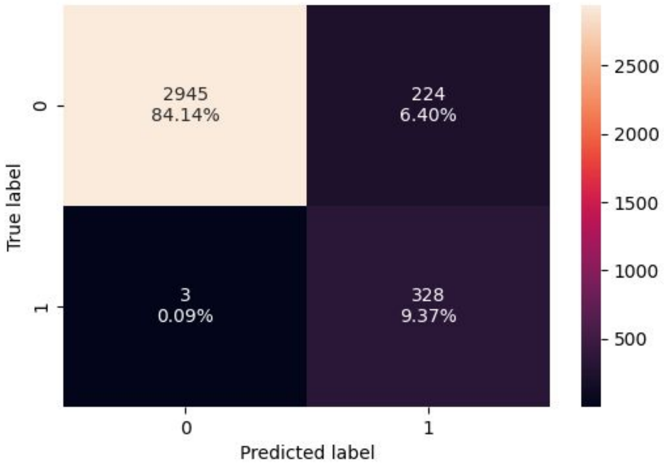


- DecisionTreeClassifier(ccp_alpha=0.00027210884353741507, random_state=1)

Model Performance Improvement, Post-Pruning

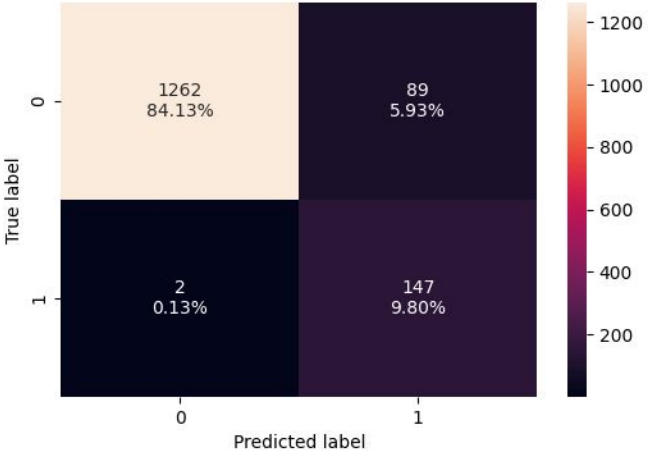
```
DecisionTreeClassifier
DecisionTreeClassifier(ccp_alpha=0.01, class_weight={0: 0.15, 1: 0.85},
                       random_state=1)
```

- Checking performance on training data



	Accuracy	Recall	Precision	F1
0	0.935143	0.990937	0.594203	0.742922

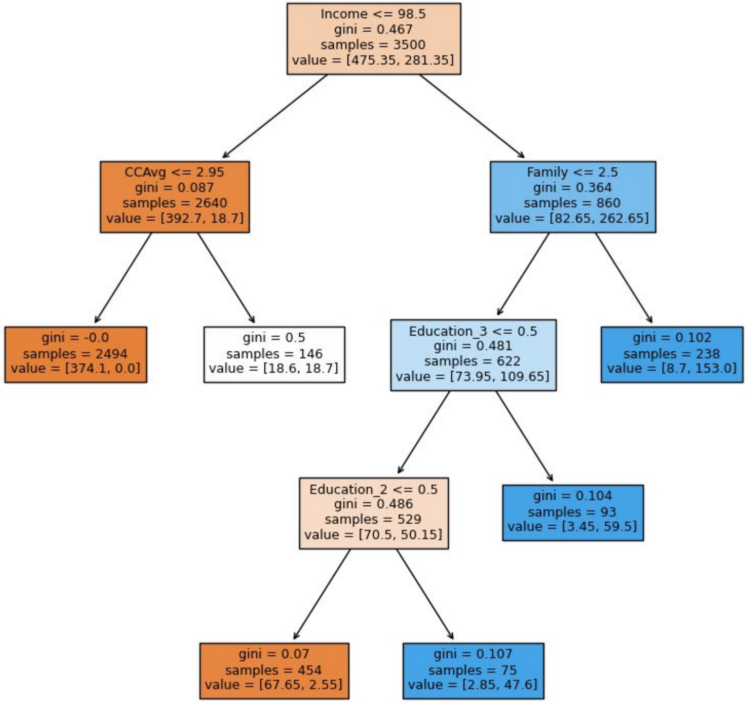
- Checking performance on testing data



	Accuracy	Recall	Precision	F1
0	0.939333	0.986577	0.622881	0.763636

Model Performance Improvement, Post-Pruning

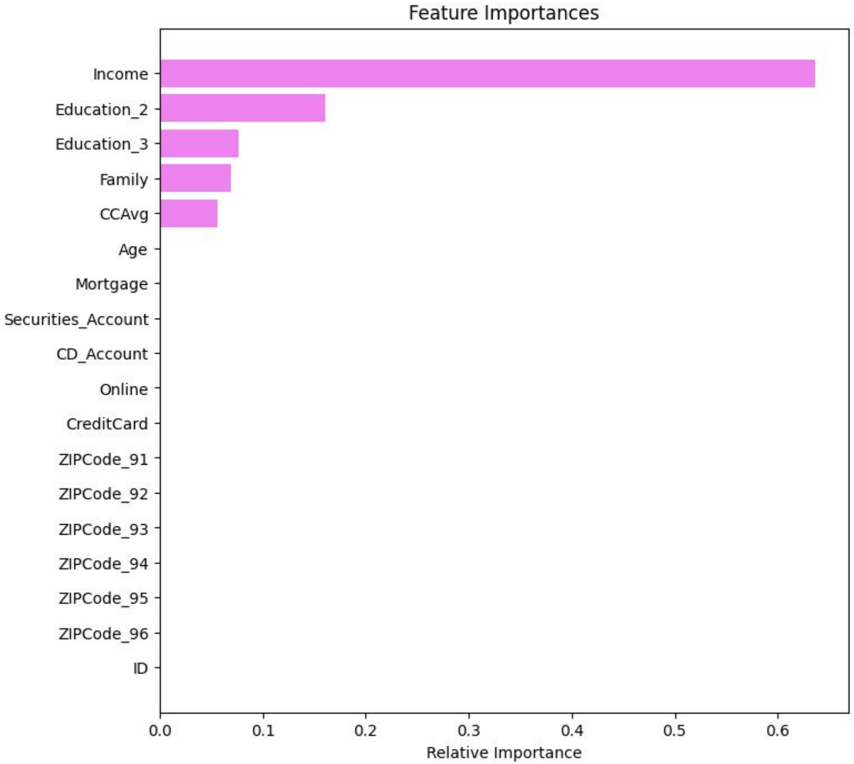
Visualizing Decision Tree



Model Performance Improvement, Post-Pruning

-Feature Importance Data after post-Pruning

	Imp
Income	0.636860
Education_2	0.160224
Education_3	0.076930
Family	0.069445
CCAvg	0.056541
ZIPCode_92	0.000000
ZIPCode_96	0.000000
ZIPCode_95	0.000000
ZIPCode_94	0.000000
ZIPCode_93	0.000000
ID	0.000000
ZIPCode_91	0.000000
Age	0.000000
Online	0.000000
CD_Account	0.000000
Securities_Account	0.000000
Mortgage	0.000000
CreditCard	0.000000



Model Performance Improvement, Post-Pruning

- Model Performance Comparison and Final Model Selection:

Training Performance Comparison:

Training performance comparison:

	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

Test Performance Comparison:

Test performance comparison:

	Decision Tree sklearn	Decision Tree (Pre-Pruning)
Accuracy	0.983333	0.978667
Recall	0.892617	0.785235
Precision	0.936620	1.000000
F1	0.914089	0.879699

Great Learning

Happy Learning !