

**0**

1

25

1

49

91107

4

1.6

1

0

0

1

0

0

0

**2**

3

39

15

11

94720

1

1.0

1

0

0

0

0

0

0

**4**

5

35

8

45

91330

4

1.0

2

0

0

0

0

0

1

**6**

7

53

27

72

91711

2

1.5

2

0

0

0

0

1

0

**8**

9

35

10

81

90089

3

0.6

2

104

0

0

0

1

0

**0**

1

25

1

49

91107

4

1

1

0

0

1

0

0

0

**2**

3

39

15

11

94720

1

1

1

0

0

0

0

0

0

**4**

5

35

8

45

91330

4

1

2

0

0

0

0

0

1

**Decision Tree of Bank Loan dataset**

In [1]:

**import numpy as np**

**import pandas as pd**

In [2]:

s=pd.read\_excel("Bank\_Personal\_Loan\_Modelling.xlsx",sheet\_name=1) s.head(10)

Out[2]:

**ID Age Experience Income**

**ZIP**

**Code**

**Family CCAvg Education Mortgage**

**Personal**

**Loan**

**Securities**

**Account**

**CD**

**Account**

**Online CreditCard**

**1** 2 45 19 34 90089 3 1.5 1 0 0 1 0 0 0

**3** 4 35 9 100 94112 1 2.7 2 0 0 0 0 0 0

**5** 6 37 13 29 92121 4 0.4 2 155 0 0 0 1 0

**7** 8 50 24 22 93943 1 0.3 3 0 0 0 0 0 1

**9** 10 34 9 180 93023 1 8.9 3 0 1 0 0 0 0 **convert float values to int values**

In [3]:

df=pd.DataFrame(s)

df.CCAvg = df.CCAvg.astype(int)

In [4]:

s.head()

Out[4]:

**ID Age Experience Income**

**ZIP**

**Code**

**Family CCAvg Education Mortgage**

**Personal**

**Loan**

**Securities**

**Account**

**CD**

**Account**

**Online CreditCard**

**1** 2 45 19 34 90089 3 1 1 0 0 1 0 0 0

**3** 4 35 9 100 94112 1 2 2 0 0 0 0 0 0

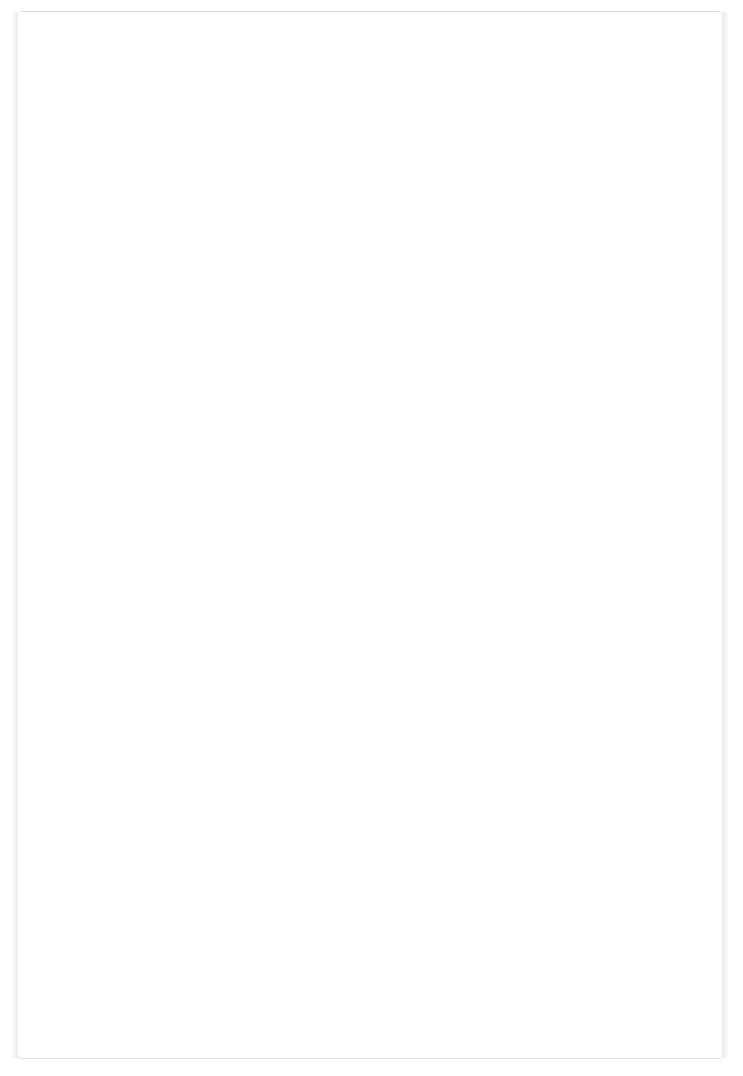
**Implementing Random Forest**

In [5]:

**from sklearn.ensemble import** RandomForestClassifier

model=RandomForestClassifier(n\_estimators=1000 ,max\_features =2, oob\_score =**True** )

features=['Age','Experience', 'Income','Family','CCAvg','Education','Mortgage','Securities



features=['Age','Experience', 'Income','Family','CCAvg','Education','Mortgage','Securities Account','CD Account' ,'Online','CreditCard']

model.fit(X=s[features],y=s['Personal Loan'])

Out[5]:

RandomForestClassifier(bootstrap=True, ccp\_alpha=0.0, class\_weight=None,

criterion='gini', max\_depth=None, max\_features=2,

max\_leaf\_nodes=None, max\_samples=None,

min\_impurity\_decrease=0.0, min\_impurity\_split=None,

min\_samples\_leaf=1, min\_samples\_split=2,

min\_weight\_fraction\_leaf=0.0, n\_estimators=1000,

n\_jobs=None, oob\_score=True, random\_state=None,

verbose=0, warm\_start=False)

In [6]:

**for** feature,imp **in** zip(features,model.feature\_importances\_):

print(feature,imp)

Age 0.049755163126708454

Experience 0.05038110950031682

Income 0.3528233225094181

Family 0.0989525880299077

CCAvg 0.15437440608042036

Education 0.1664612405399596

Mortgage 0.04707380736552793

Securities Account 0.00638514569069101

CD Account 0.05392871547773416

Online 0.00915833569068792

CreditCard 0.01070616598862781

**attributes -Income, CCavg and Education have higher accuracy than other attributes. Hence these three attributes will be selected.**

**Implementing Decision Tree**

**\*checking for null values**

In [7]:

s.isnull().sum()

Out[7]:

ID 0

Age 0

Experience 0

Income 0

ZIP Code 0

Family 0

CCAvg 0

Education 0

Mortgage 0

Personal Loan 0

Securities Account 0

CD Account 0

Online 0

CreditCard 0

dtype: int64

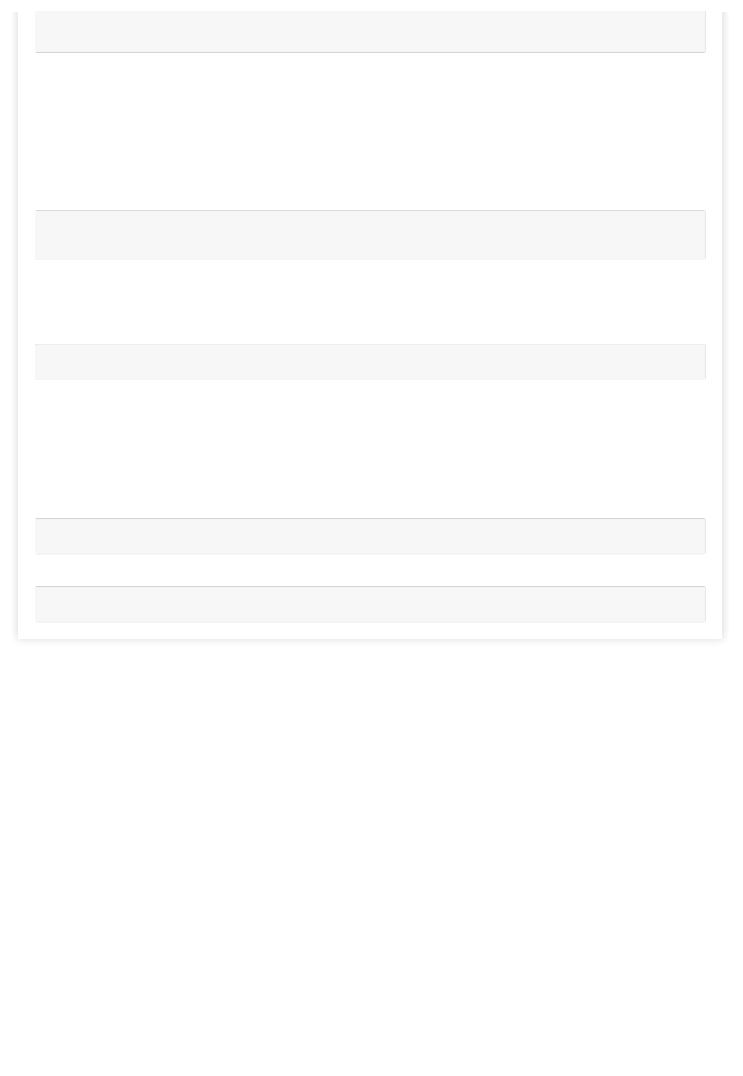
**there are no null values**

In [8]:

**from sklearn import** tree

tree\_model=tree .DecisionTreeClassifier(max\_depth=6)

predictors=pd.DataFrame([s['Income'],s["CCAvg"],s['Education']]).T



predictors=pd.DataFrame([s['Income'],s["CCAvg"],s['Education']]).T

tree\_model.fit(X=predictors,y=s['Personal Loan'])

Out[8]:

DecisionTreeClassifier(ccp\_alpha=0.0, class\_weight=None, criterion='gini',

max\_depth=6, max\_features=None, max\_leaf\_nodes=None,

min\_impurity\_decrease=0.0, min\_impurity\_split=None,

min\_samples\_leaf=1, min\_samples\_split=2,

min\_weight\_fraction\_leaf=0.0, presort='deprecated',

random\_state=None, splitter='best')

In [9]:

**with** open ("DT\_Bank1",'w') **as** f:

f=tree .export\_graphviz(tree\_model,feature\_names=['Income','CCAvg','Education'],out\_file=f); **model accuracy**

In [10]:

tree\_model.score(X=predictors,y=s['Personal Loan'])

Out[10]:

0.9732

**The model accuracy is 97.32%**

**Inference**

* If income<=113.5 and CCAvg<=2.5 and income<=106.5 then loan sanctioned.
* If income<=106.5 and education<=1.5 and CCavg<=1.5 then loan sanctioned.
* If income>111.5 and CCAvg<=1.5 and Education<=1.5 then loan sanctioned.
* If income<=110.5 and CCAvg<=0.5 and Education<=1.5 then loan not sanctioned.
* If income>110.5 and CCAvg<=0.5 and Education<=1.5 then loan sanctioned.
* If income>108.5 and CCAvg<=0.5 and Education<=1.5 then loan sanctioned.
* If income>116.5 and Education>=1.5 then loan not sanctioned.
* If income<=116.5 and CCAvg>=3.5 and Education<=1.5 then loan sanctioned.
* If income<=114.5 and CCAvg<=1.5 and Education<=1.5 then loan sanctioned.
* If income<=114.5 and CCAvg>=1.5 and Education<=1.5 then loan sanctioned.