**Bank Loan Decision Tree**

**Importing the packages**

import pandas as pd

import numpy as np

from sklearn import tree

from sklearn import preprocessing

### Loading the dataset

data = pd.read\_excel('Bank\_Personal\_Loan\_Modelling.xlsx',sheet\_name=1)

data.head()

Out[6]:

ID Age Experience ... CD Account Online CreditCard

0 1 25 1 ... 0 0 0

1 2 45 19 ... 0 0 0

2 3 39 15 ... 0 0 0

3 4 35 9 ... 0 0 0

4 5 35 8 ... 0 0 1

[5 rows x 14 columns]

### Checking for null values

data.isna().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

### Random Forest

from sklearn.ensemble import RandomForestClassifier

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

rf\_model=RandomForestClassifier(n\_estimators=1000,max\_features=2,oob\_score=True)

rf\_model.fit(X=data[features],y=data['Personal Loan'])

Out[11]:

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)

rf\_model.oob\_score\_

Out[12]: 0.9882

for features,imp in zip(features,rf\_model.feature\_importances\_):

print(features,imp)

Age 0.04488356692163336

Experience 0.04427410560355725

Income 0.34199877112301474

Family 0.09992398343037842

CCAvg 0.1804361815501903

Education 0.1657718399686197

Mortgage 0.04444453885325242

Securities Account 0.005625749547089936

CD Account 0.054448576345085556

Online 0.008136754300067332

CreditCard 0.010055932357111054

#### Inference : This shows that the important features are Income,CCAvg and Education

### Decision Tree

tree\_model = tree.DecisionTreeClassifier(max\_depth=3)

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

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

Out[16]:

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

max\_depth=3, 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',

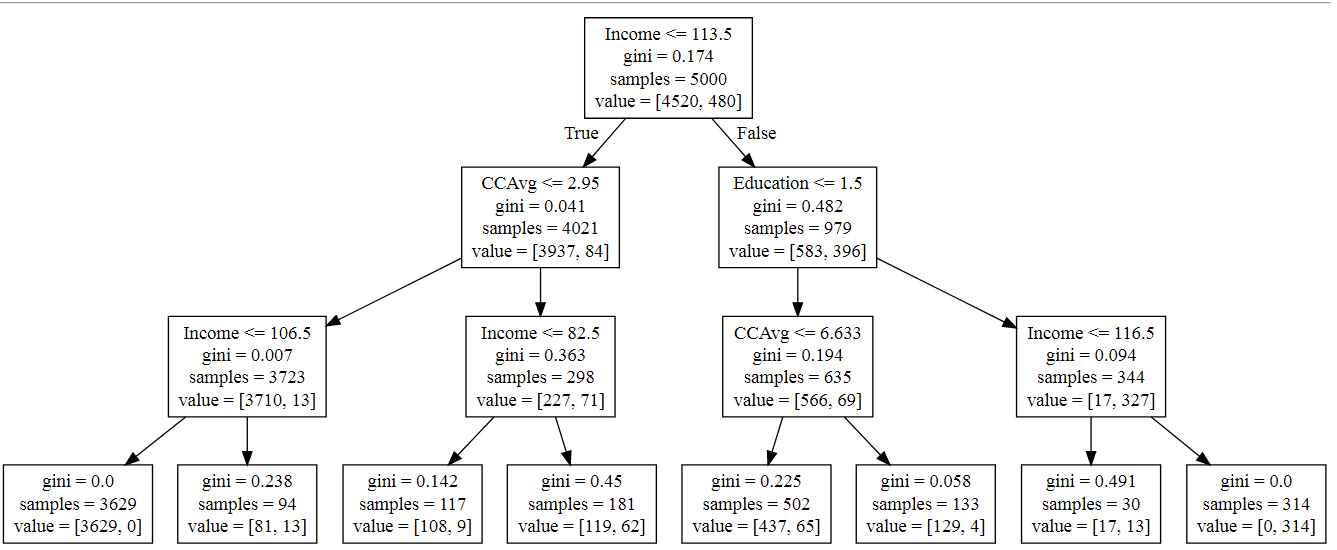
random\_state=None, splitter='best')

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

Out[17]: 0.9668

with open('Dtree4.dot','w') as f:

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



**Rules corresponding to given tree:**

1. **From total 5000 customers, 4520 doesn’t accept the loan offered from campaign and 480 customers has accepted.**
2. **The customer whose Avg. spending on credit cards per month ($000)(CCAvg) is <=2.95 is 4021 from which 3937 doesn’t accept the loan and 84 has accepted the loan.**
3. **The customer whose income <=106.5 is 3723, from which 3710 doesn’t accept the loan and 13 has accepted the loan. From which 3629 doesn’t accepted the loan and 0 accepted the loan.**
4. **If education is undergraduate and CCAvg <=6.663 such 635 customer present from which 556 accept the loan and 69 has accepted the loan.**
5. **If education is Graduate or advance professional and income <=116.5,from total 344 customers 17 doesn’t accept the loan and 327 has accepted the loan.**

**Further from 314 customers, all have accepted the loan.**