—--------Import Important libraries—----------
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import preprocessing
from sklearn import datasets
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
```

—---------Read Dataset—----------
```
df = pd.read_csv('Downloads/Fraud_check.csv')
df
```

	Undergrad	Marital.Status	Taxable.Income	City.Population	Work.Experience	Ur
0	NO	Single	68833	50047	10	
1	YES	Divorced	33700	134075	18	
2	NO	Married	36925	160205	30	
3	YES	Single	50190	193264	15	
4	NO	Married	81002	27533	28	
...	...	...	...	...	...	
595	YES	Divorced	76340	39492	7	
596	YES	Divorced	69967	55369	2	
597	NO	Divorced	47334	154058	0	
598	YES	Married	98592	180083	17	
599	NO	Divorced	96519	158137	16	

600 rows × 6 columns

—----------Get dummies—----------
```
df = pd.get_dummies(df,columns=['Undergrad','Marital.Status','Urban'],drop_first=True)
df
```

	Taxable.Income	City.Population	Work.Experience	Undergrad_YES	Marital.Status_Married	Marital.Status_Single	Urban_YES
0	68833	50047	10	0	0	1	1
1	33700	134075	18	1	0	0	1
2	36925	160205	30	0	1	0	1
3	50190	193264	15	1	0	1	1
4	81002	27533	28	0	1	0	0
...	...	...	...	...	...	...	...
595	76340	39492	7	1	0	0	1

—----------Cut—----------
```
df['TC'] = pd.cut(df['Taxable.Income'],bins=[10002,30000,99620],labels=['Risky','Good'])
df
```

	Taxable.Income	City.Population	Work.Experience	Undergrad_YES	Marital.Status_Married	Marital.Status_Single	Urban_YES	TC
0	68833	50047	10	0	0	1	1	Good
1	33700	134075	18	1	0	0	1	Good
2	36925	160205	30	0	1	0	1	Good
3	50190	193264	15	1	0	1	1	Good
4	81002	27533	28	0	1	0	0	Good
...	...	...	...	...	...	...	...	...
595	76340	39492	7	1	0	0	1	Good
596	69967	55369	2	1	0	0	1	Good
597	47334	154058	0	0	0	0	1	Good
598	98592	180083	17	1	1	0	0	Good
599	96519	158137	16	0	0	0	0	Good

600 rows × 8 columns

—----------Get dummies—----------

```
df = pd.get_dummies(df,columns=['TC'],drop_first=True)
df.tail(10)
```

	Taxable.Income	City.Population	Work.Experience	Undergrad_YES	Marital.Status_Married	Marital.Status_Single	Urban_YES	TC_Good
590	43018	85195	14	0	1	0	1	1
591	27394	132859	18	1	0	1	1	0
592	68152	75143	16	1	0	1	0	1
593	84775	131963	10	0	0	0	1	1
594	47364	97526	9	0	1	0	1	1
595	76340	39492	7	1	0	0	1	1

—----------Normalize data—----------

```
def norm_func(i):
    x=(i-i.min())/(i.max()-i.min())
    return(x)
```

—----------Divide data—----------

```
divide = norm_func(df.iloc[:,1:])
divide.tail(10)
```

	City.Population	Work.Experience	Undergrad_YES	Marital.Status_Married	Marital.Status_Single	Urban_YES	TC_Good
590	0.341473	0.466667	0.0	1.0	0.0	1.0	1.0
591	0.615406	0.600000	1.0	0.0	1.0	1.0	0.0
592	0.283703	0.533333	1.0	0.0	1.0	0.0	1.0
593	0.610256	0.333333	0.0	0.0	0.0	1.0	1.0
594	0.412341	0.300000	0.0	1.0	0.0	1.0	1.0
595	0.078811	0.233333	1.0	0.0	0.0	1.0	1.0
596	0.170058	0.066667	1.0	0.0	0.0	1.0	1.0
597	0.737240	0.000000	0.0	0.0	0.0	1.0	1.0
598	0.886810	0.566667	1.0	1.0	0.0	0.0	1.0
599	0.760683	0.533333	0.0	0.0	0.0	0.0	1.0

—----------Rename—---------------------
df =
divide.rename({'City.Population':'c','Work.Experience':'w','Undergrad_YES':'u','Marital.Status_Married':'m','Marital.Status_Single':'ms','Urban_YES':'u','TC_Good':'t'},axis=1)
df

	c	w	u	m	ms	u	t	income
0	0.139472	0.333333	0.0	0.0	1.0	1.0	1.0	Good
1	0.622394	0.600000	1.0	0.0	0.0	1.0	1.0	Good
2	0.772568	1.000000	0.0	1.0	0.0	1.0	1.0	Good
3	0.962563	0.500000	1.0	0.0	1.0	1.0	1.0	Good
4	0.010081	0.933333	0.0	1.0	0.0	0.0	1.0	Good
...	...	...	...	...	...	...	...	...
595	0.078811	0.233333	1.0	0.0	0.0	1.0	1.0	Good
596	0.170058	0.066667	1.0	0.0	0.0	1.0	1.0	Good
597	0.737240	0.000000	0.0	0.0	0.0	1.0	1.0	Good
598	0.886810	0.566667	1.0	1.0	0.0	0.0	1.0	Good

—----------Divide data—----------
x = divide.drop(['TC_Good'],axis=1)
y = divide['TC_Good']

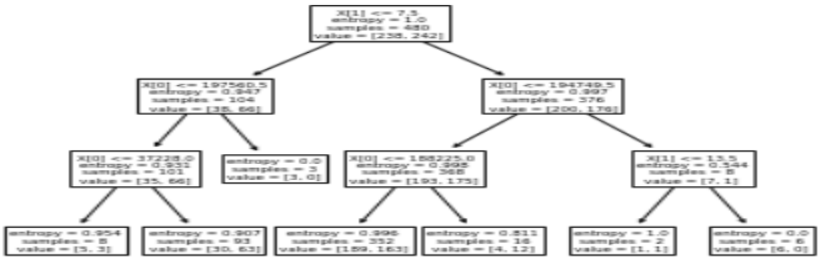
—----------Train Test—----------
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3,random_state=30)

divide['income']='<=30000'
divide.loc[df['Taxable.Income']>=30000,'income']='Good'
divide.loc[df['Taxable.Income']<=30000,'income']='Risky'

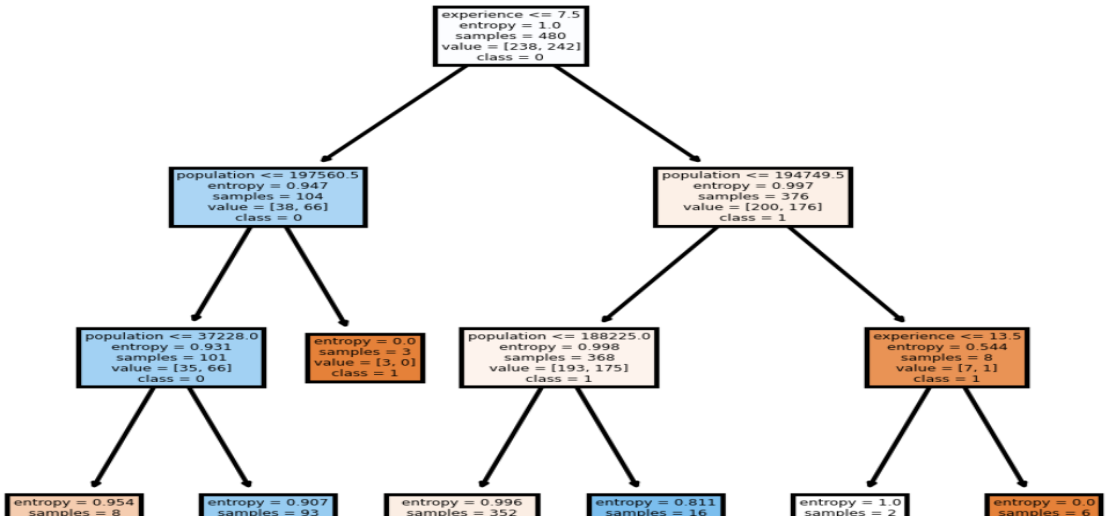
df.drop(['Taxable.Income'],axis=1,inplace=True)

model = DecisionTreeClassifier(criterion='entropy',max_depth=30)
model.fit(x_train,y_train)

tree.plot_tree(model)



```
fn = ['c','w','u','m','ms','u']
cn = ['1','0']
fig,axes=plt.subplots(nrows=1,ncols=1,figsize=(4,4),dpi=300)
tree.plot_tree(model,feature_names=fn,class_names=cn,filled=True)
```




```
—---------Prediction—---------------
predict = model.predict(x_test)
predict
```

```
array([1., 1., 1., 0., 1., 1., 0., 1., 1., 1., 1., 1., 1., 1., 1., 0., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 0., 1., 1., 0., 1., 0.,
       0., 1., 1., 1., 0., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       0., 1., 1., 1., 0., 1., 1., 1., 1., 0., 1., 0., 1., 1., 1., 0., 1.,
       1., 1., 1., 0., 0., 1., 1., 0., 1., 1., 1., 1., 0., 1., 0., 1., 1.,
       0., 1., 1., 1., 0., 1., 0., 0., 1., 0., 1., 1., 1., 0., 1., 1., 1.,
       0., 0., 1., 0., 1., 1., 1., 1., 1., 1., 0., 1., 1., 1., 0., 0., 1.,
       1., 0., 1., 1., 1., 1., 1., 1., 1., 0., 1., 1., 1., 1., 0., 0., 1.,
       1., 0., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       0., 1., 1., 0., 1., 0., 1., 1., 1., 1., 1., 1., 1., 1., 1., 0., 1.,
       1., 0., 1., 0., 1., 0., 1., 1., 1., 1.])
```

```
pd.crosstab(y_test,predict)
```

col_0	0.0	1.0
TC_Good		
0.0	6	32
1.0	36	106

```
np.mean(y_test==predict)
```

```
0.6222222222222222
```

—————Gini plot——————————
model = DecisionTreeClassifier(criterion='gini',max_depth=30)
model.fit(x_train,y_train)
predict = model.predict(x_test)
predict

```
array([1., 1., 1., 0., 1., 1., 0., 1., 1., 0., 1., 0., 1., 1., 1., 1., 1.,
       1., 1., 1., 0., 1., 1., 1., 1., 1., 1., 1., 1., 0., 1., 1., 1., 1.,
       0., 0., 1., 1., 1., 1., 0., 1., 1., 0., 1., 1., 1., 1., 1., 1., 1.,
       0., 1., 1., 1., 1., 0., 1., 1., 0., 1., 1., 0., 1., 1., 0., 0., 1.,
       1., 0., 1., 0., 1., 1., 1., 0., 1., 1., 1., 0., 0., 1., 0., 1., 1.,
       0., 1., 1., 1., 0., 0., 0., 0., 1., 1., 1., 1., 1., 0., 1., 1., 1.,
       1., 1., 0., 0., 1., 1., 1., 1., 1., 1., 0., 1., 1., 1., 1., 0., 1.,
       1., 0., 1., 0., 1., 1., 1., 1., 1., 0., 1., 1., 1., 1., 1., 0., 1.,
       1., 0., 1., 1., 1., 1., 1., 0., 0., 1., 1., 1., 1., 1., 1., 1., 1.,
       0., 1., 1., 0., 1., 1., 1., 1., 1., 0., 1., 1., 0., 1., 1., 1., 1.,
       1., 1., 1., 0., 1., 0., 1., 1., 1., 1.])
```

```
pd.crosstab(y_test,predict)
```

col_0	0.0	1.0
TC_Good		
0.0	6	32
1.0	39	103

```
np.mean(predict==y_test)
```

```
0.6055555555555555
```

—————Random forest——————————
from sklearn.ensemble **import** RandomForestClassifier **as** RF
model **=** RF(n_jobs **=** 3,n_estimators **=** 15, oob_score **= True**, criterion **=** "entropy")
model.fit(x_train,y_train)

```
RandomForestClassifier(criterion='entropy', n_estimators=15, n_jobs=3,
                       oob_score=True)
```

—————Accuracy——————————
from sklearn.metrics **import** accuracy_score

```
accuracy = accuracy_score(y_train,prediction)
##98.33%

—---------Mean—----------------
np.mean(prediction == y_train)
##98.33%

—--------confusion matrix—-----------
from sklearn.metrics import confusion_matrix
confusion = confusion_matrix(y_train,prediction)

pred_test = model.predict(x_test)

acc_test =accuracy_score(y_test,pred_test)
##78.333%
```