In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.tree import  DecisionTreeClassifier
from sklearn import tree
from sklearn.metrics import classification_report
from sklearn import preprocessing
```

In [2]:
```
data=pd.read_csv('Fraud_check.csv')
data
```

Out[2]:

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

600 rows × 6 columns

In [3]:
```
data.shape
```

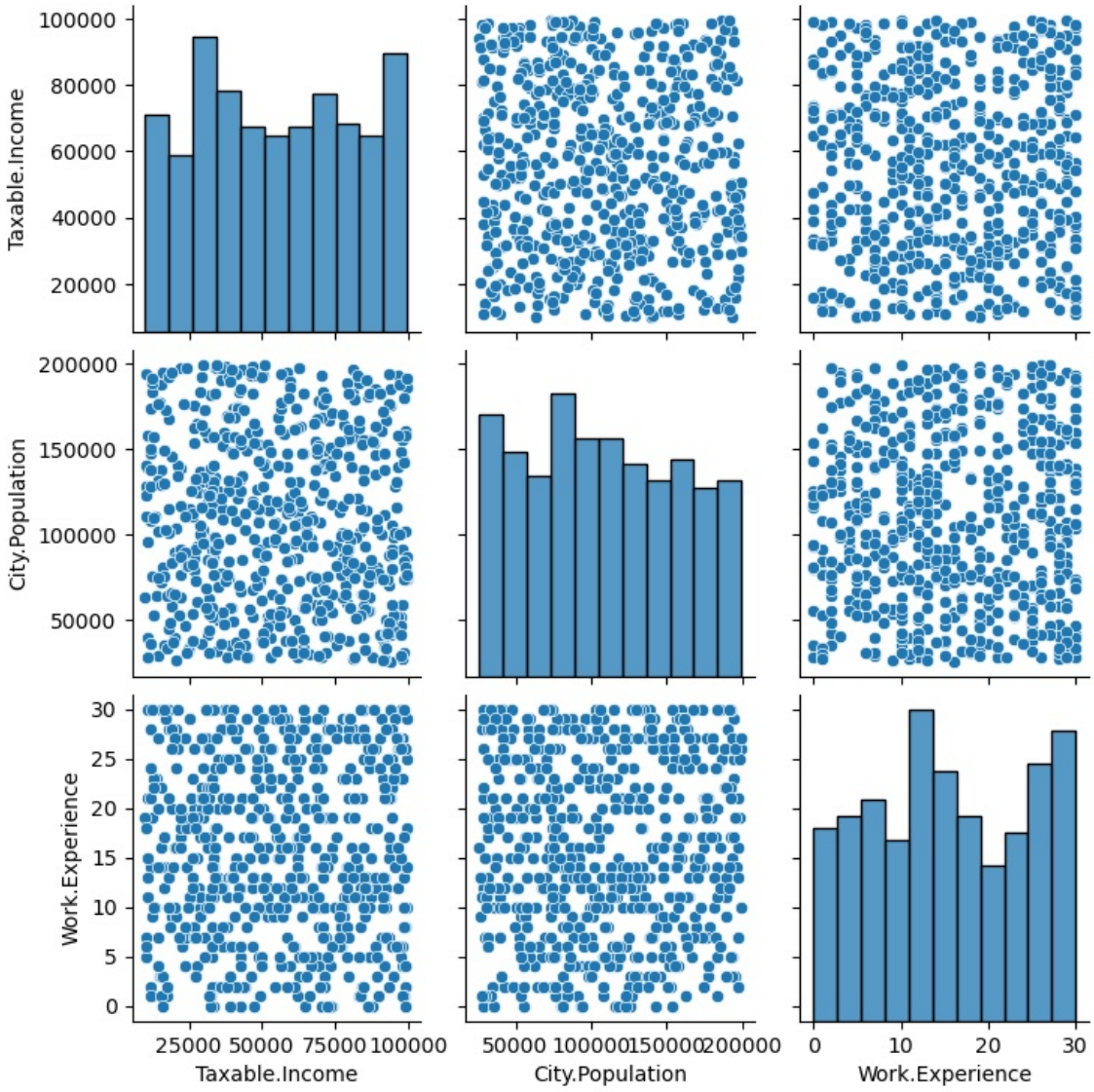
Out[3]:
```
(600, 6)
```

In [5]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 600 entries, 0 to 599
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   Undergrad        600 non-null    object
 1   Marital.Status   600 non-null    object
 2   Taxable.Income   600 non-null    int64 
 3   City.Population  600 non-null    int64 
 4   Work.Experience  600 non-null    int64 
 5   Urban            600 non-null    object
dtypes: int64(3), object(3)
memory usage: 28.2+ KB
```

In [6]:
```
sns.pairplot(data)
```

Out[6]:
```
<seaborn.axisgrid.PairGrid at 0x28628abc250>
```



In [7]:
```
plt.figure(figsize=(20,10))
sns.heatmap(data.corr(),annot=True)
```

Out[7]:
```
<AxesSubplot:>
```

In [8]:
```
label_encoder = preprocessing.LabelEncoder()
data['Undergrad']= label_encoder.fit_transform(data['Undergrad'])
data['Urban']= label_encoder.fit_transform(data['Urban'])
data['Marital.Status']= label_encoder.fit_transform(data['Marital.Status'])
```

In [9]:
```
data
```

Out[9]:

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

600 rows × 6 columns

In [10]:
```
data['Status'] = data['Taxable.Income'].apply(lambda Income: 'Risky' if Income <= 30000 else 'Good')
```

In [11]:
```
data
```

Out[11]:

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

600 rows × 7 columns

In [12]:
```
data['Status'].unique()
```

Out[12]:
```
array(['Good', 'Risky'], dtype=object)
```

In [13]:
```
label_encoder = preprocessing.LabelEncoder()
data['Status']= label_encoder.fit_transform(data['Status'])
data
```

Out[13]:

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

600 rows × 7 columns

In [14]:
```
x=data.iloc[:,0:4]
y=data['Status']
```

In [15]:
```
data['Status'].unique()
```

Out[15]:
```
array([0, 1])
```

In [16]:
```
data.Status.value_counts()
```

Out[16]:
```
0    476
1    124
Name: Status, dtype: int64
```

In [17]:
```
colnames = list(data.columns)
colnames
```

Out[17]:
```
['Undergrad',
 'Marital.Status',
 'Taxable.Income',
 'City.Population',
 'Work.Experience',
 'Urban',
 'Status']
```

In [18]:
```
x_train, x_test,y_train,y_test = train_test_split(x,y, test_size=0.2,random_state=40)
```

In [19]:
```
model = DecisionTreeClassifier(criterion = 'entropy',max_depth=3)
model.fit(x_train,y_train)
```

Out[19]:
```
DecisionTreeClassifier(criterion='entropy', max_depth=3)
```

In [20]:
```
tree.plot_tree(model);
```

In [21]:
```
fn=['Undergrad' ,'Marital.Status','Taxable.Income','City.Population']
cn=['Good', 'Risky']
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);
```

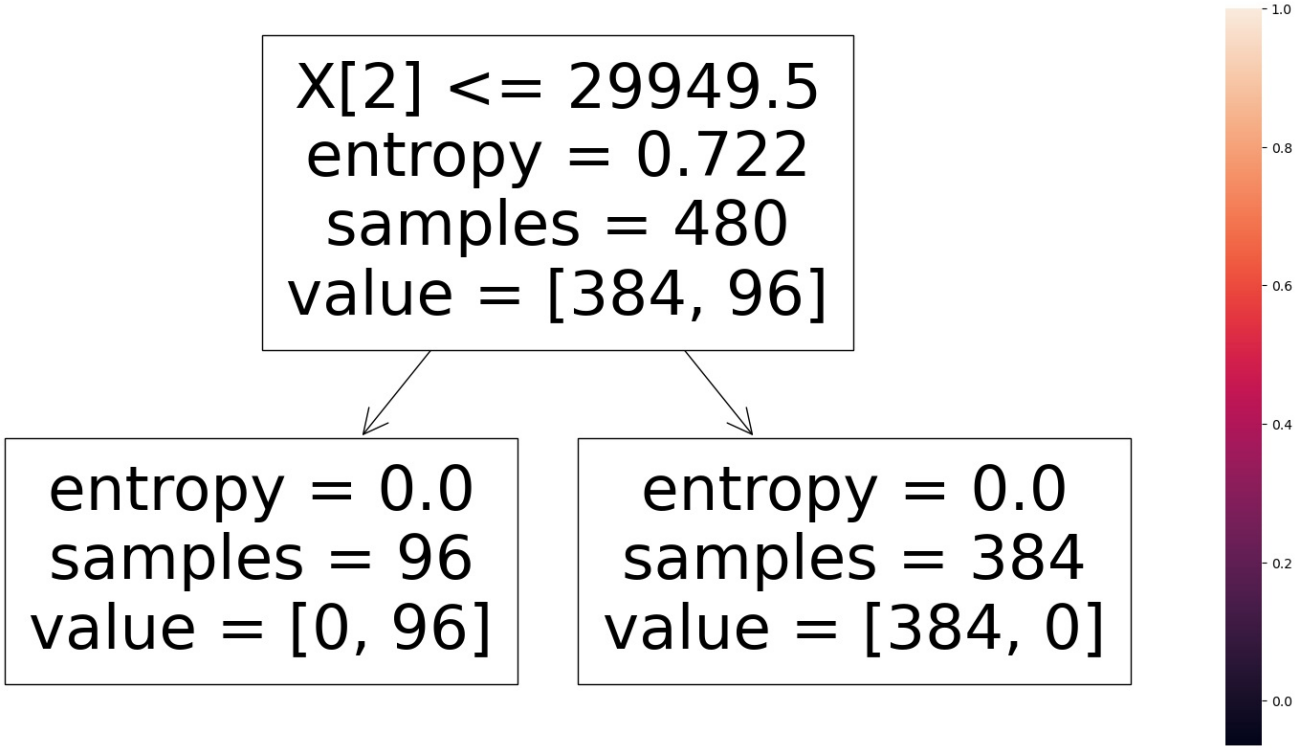
In [22]:
```
model.feature_importances_
```

Out[22]:
```
array([0., 0., 1., 0.])
```

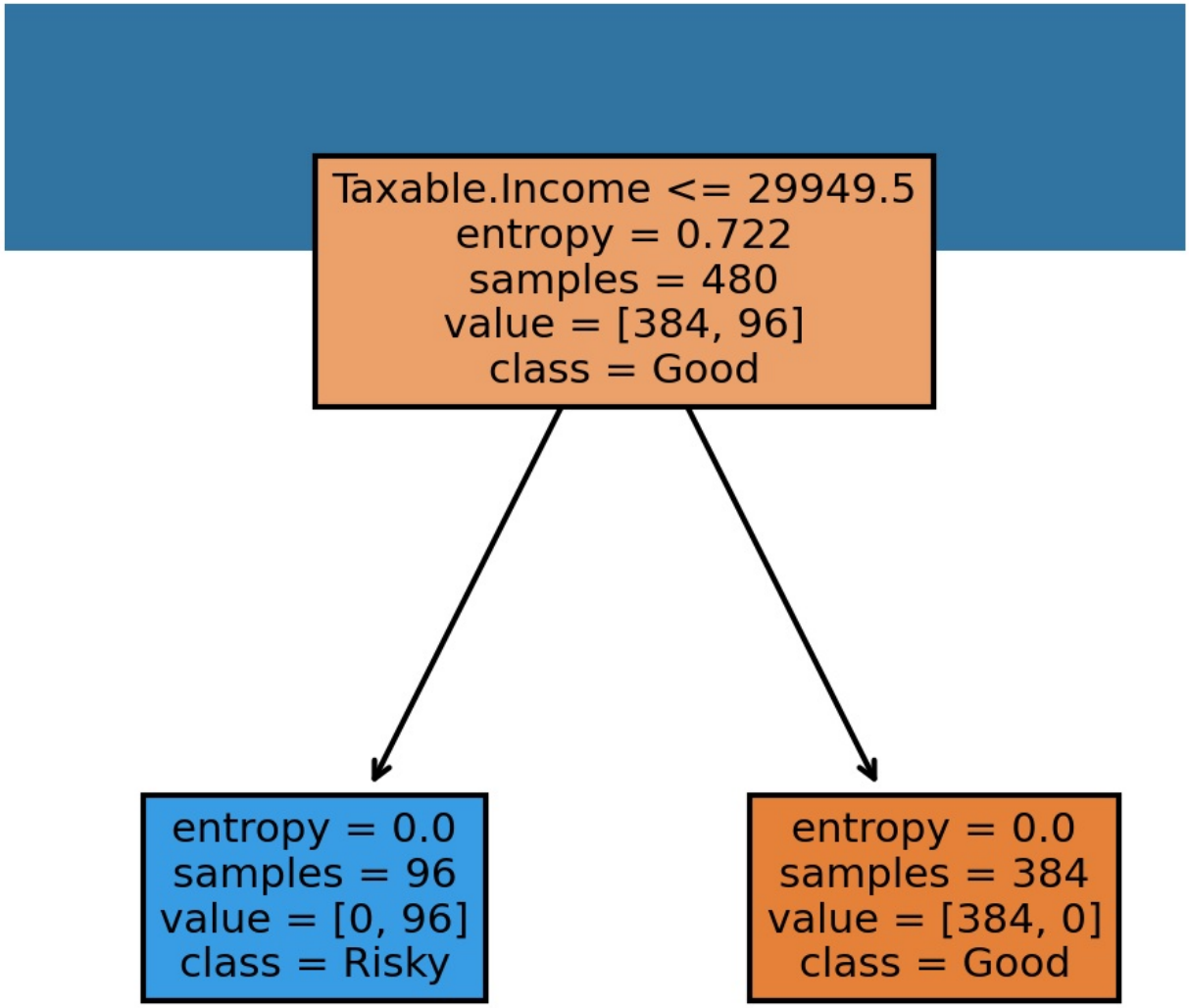
In [23]:
```
feature_imp = pd.Series(model.feature_importances_,index=fn).sort_values(ascending=False)
feature_imp
```

Out[23]:
```
Taxable.Income     1.0
Undergrad          0.0
Marital.Status     0.0
City.Population    0.0
dtype: float64
```

In [24]:
```
sns.barplot(x=feature_imp, y=feature_imp.index)
plt.xlabel('Feature Importance Score')
plt.ylabel('Features')
plt.title("Visualizing Important Features")
plt.show()
```



Visualizing Important Features



In [25]:
```
preds = model.predict(x_test)
pd.Series(preds).value_counts()
```

Out[25]:
```
0    92
1    28
dtype: int64
```

In [26]:
```
preds
```

Out[26]:
```
array([0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0,
       0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0])
```

In [27]:
```
pd.crosstab(y_test,preds)
```

Out[27]:

Status	col_0	0	1
0		92	0
1		0	28

In [28]:
```
np.mean(preds==y_test)
```

In [28]:
```
np.mean(preds==y_test)
```

Out[28]:
```
1.0
```

In [29]:
```
model_gini = DecisionTreeClassifier(criterion='gini', max_depth=3)
```

In [30]:
```
model_gini.fit(x_train, y_train)
```

Out[30]:
```
DecisionTreeClassifier(max_depth=3)
```

In [31]:
```
pred=model.predict(x_test)
np.mean(preds==y_test)
```

Out[31]:
```
1.0
```

In [32]:
```
model.feature_importances_
```

Out[32]:
```
array([0., 0., 1., 0.])
```

In [33]:
```
from sklearn.tree import DecisionTreeRegressor
```

In [34]:
```
array = data.values
X = array[:,0:4]
y = array[:,3]
```

In [35]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=1)
```

In [36]:
```
model = DecisionTreeRegressor()
model.fit(X_train, y_train)
```

Out[36]:
```
DecisionTreeRegressor()
```

In [37]:
```
model.score(X_test,y_test)
```

Out[37]:
```
0.9998959550879601
```

In [ ]:

In [ ]: