In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets
import numpy as np
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 [6]:
```
df.head()
```

Out[6]:

	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

In [5]:
```
df=pd.read_csv("C:\\Users\\Admin\\Downloads\\Assignment 6\\Fraud_check.csv")
df.tail()
```

Out[5]:

	Undergrad	Marital.Status	Taxable.Income	City.Population	Work.Experience	Urban
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

In [7]:
```
df=pd.get_dummies(df,columns=['Undergrad','Marital.Status','Urban'], drop_first=True)
```

In [8]:
```
df["TaxInc"] = pd.cut(df["Taxable.Income"], bins = [10002,30000,99620], labels = ["Risk
```

In [9]:
```
print(df)
```

```
     Taxable.Income  City.Population  Work.Experience  Undergrad_YES  \
0             68833            50047               10              0
1             33700           134075               18              1
2             36925           160205               30              0
3             50190           193264               15              1
4             81002            27533               28              0
..              ...              ...              ...            ...
```

```
595           76340           39492                7             1
596           69967           55369                2             1
597           47334          154058                0             0
598           98592          180083               17             1
599           96519          158137               16             0

     Marital.Status_Married  Marital.Status_Single  Urban_YES TaxInc
0                         0                      1          1   Good
1                         0                      0          1   Good
2                         1                      0          1   Good
3                         0                      1          1   Good
4                         1                      0          0   Good
..                      ...                    ...        ...    ...
595                       0                      0          1   Good
596                       0                      0          1   Good
597                       0                      0          1   Good
598                       1                      0          0   Good
599                       0                      0          0   Good

[600 rows x 8 columns]
```

In [10]:
```
df = pd.get_dummies(df,columns = ["TaxInc"],drop_first=True)
```

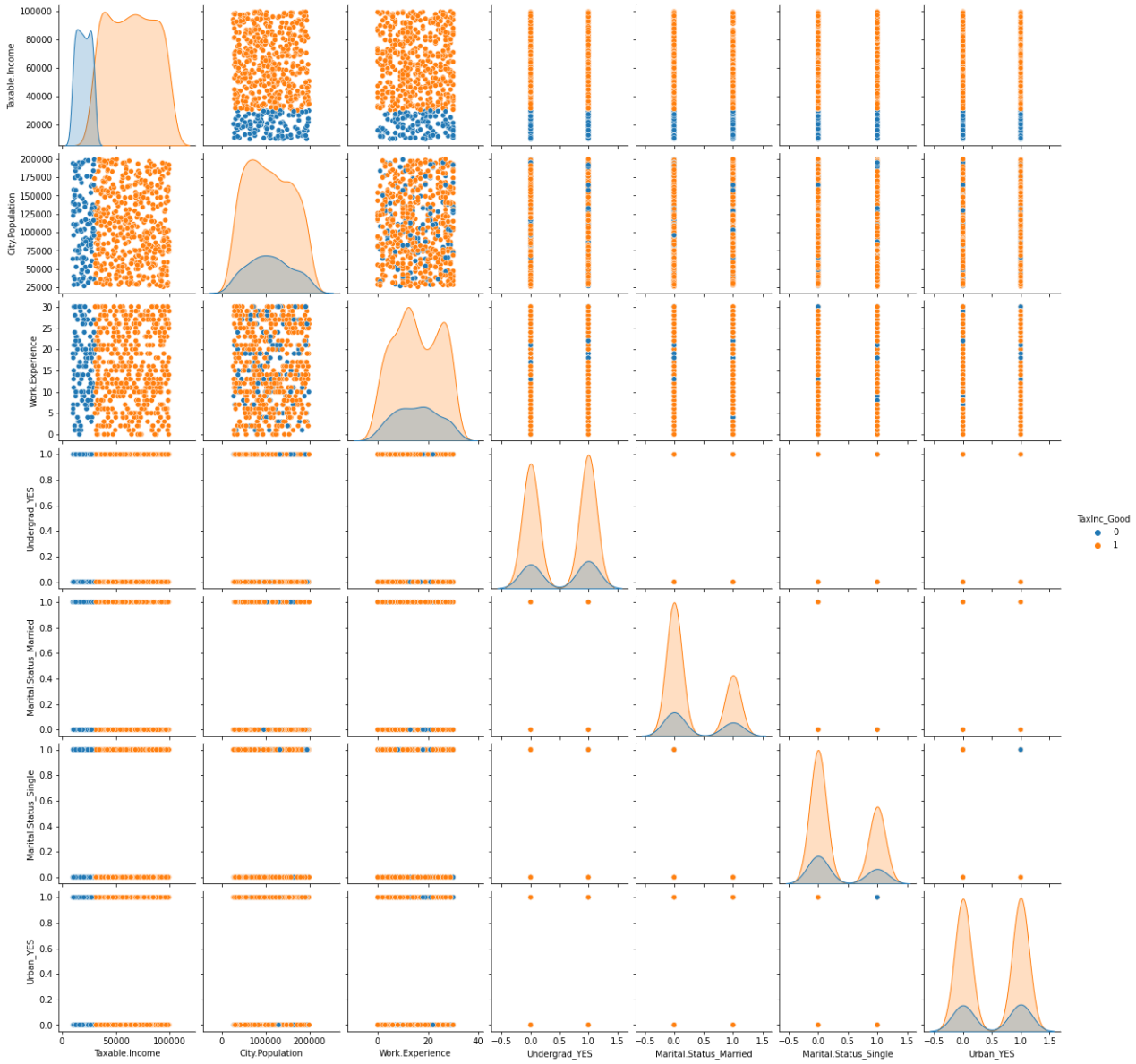
In [11]:
```
df.tail(10)
```

Out[11]:

	Taxable.Income	City.Population	Work.Experience	Undergrad_YES	Marital.Status_Married	Marital.S
590	43018	85195	14	0	1	
591	27394	132859	18	1	0	
592	68152	75143	16	1	0	
593	84775	131963	10	0	0	
594	47364	97526	9	0	1	
595	76340	39492	7	1	0	
596	69967	55369	2	1	0	
597	47334	154058	0	0	0	
598	98592	180083	17	1	1	
599	96519	158137	16	0	0	

In [12]:
```
import seaborn as sns
sns.pairplot(data=df, hue = 'TaxInc_Good')
```

Out[12]:
```
<seaborn.axisgrid.PairGrid at 0x1cdfa370340>
```



In [13]:

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

In [14]:

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

Out[14]:

	City.Population	Work.Experience	Undergrad_YES	Marital.Status_Married	Marital.Status_Single	Urb
590	0.341473	0.466667	0.0	1.0	0.0	
591	0.615406	0.600000	1.0	0.0	1.0	
592	0.283703	0.533333	1.0	0.0	1.0	
593	0.610256	0.333333	0.0	0.0	0.0	
594	0.412341	0.300000	0.0	1.0	0.0	
595	0.078811	0.233333	1.0	0.0	0.0	

	City.Population	Work.Experience	Undergrad_YES	Marital.Status_Married	Marital.Status_Single	Urb
596	0.170058	0.066667	1.0	0.0	0.0	
597	0.737240	0.000000	0.0	0.0	0.0	
598	0.886810	0.566667	1.0	1.0	0.0	
599	0.760683	0.533333	0.0	0.0	0.0	

In [15]:
```
X = df_norm.drop(['TaxInc_Good'], axis=1)
y = df_norm['TaxInc_Good']
```

In [16]:
```
from sklearn.model_selection import train_test_split
```

In [17]:
```
Xtrain, Xtest, ytrain, ytest = train_test_split(X, y, test_size=0.2, random_state=0)
```

In [18]:
```
df_norm["income"]="<=30000"
df_norm.loc[df["Taxable.Income"]>=30000,"income"]="Good"
df_norm.loc[df["Taxable.Income"]<=30000,"income"]="Risky"
```

In [19]:
```
df.drop(["Taxable.Income"],axis=1,inplace=True)
```

In [27]:
```
df.rename(columns={"Undergrad":"undergrad","Marital.Status":"marital","City.Population"
```

In [28]:
```
from sklearn import preprocessing
le=preprocessing.LabelEncoder()
for column_name in df.columns:
    if df[column_name].dtype == object:
        df[column_name] = le.fit_transform(df[column_name])
    else:
        pass
```

In [29]:
```
features = df.iloc[:,0:5]
labels = df.iloc[:,5]
```

In [30]:
```
colnames = list(df.columns)
predictors = colnames[0:5]
target = colnames[5]
```

In [31]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(features,labels,test_size = 0.2,strati
```

In [32]:
```
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)
```

Out[32]:
```
RandomForestClassifier(criterion='entropy', n_estimators=15, n_jobs=3,
                       oob_score=True)
```

In [33]:
```
model.estimators_
model.classes_
model.n_features_
model.n_classes_
```

Out[33]:
```
2
```

In [34]:
```
model.n_outputs_
```

Out[34]:
```
1
```

In [35]:
```
model.oob_score_
```

Out[35]:
```
0.5375
```

In [36]:
```
prediction = model.predict(x_train)
```

In [37]:
```
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_train,prediction)
```

In [38]:
```
np.mean(prediction == y_train)
```

Out[38]:
```
0.9895833333333334
```

In [39]:
```
from sklearn.metrics import confusion_matrix
confusion = confusion_matrix(y_train,prediction)
```

In [40]:
```
pred_test = model.predict(x_test)
```

In [41]:
```
acc_test =accuracy_score(y_test,pred_test)
```

In [43]:
```
pip install pydotplus
```

```
Collecting pydotplus
  Downloading pydotplus-2.0.2.tar.gz (278 kB)
Requirement already satisfied: pyparsing>=2.0.1 in c:\users\admin\anaconda3\lib\site-pac
kages (from pydotplus) (3.0.4)
Building wheels for collected packages: pydotplus
  Building wheel for pydotplus (setup.py): started
  Building wheel for pydotplus (setup.py): finished with status 'done'
  Created wheel for pydotplus: filename=pydotplus-2.0.2-py3-none-any.whl size=24575 sha2
```

```
56=2c7a485e379e0248e2f534bfa81e927f290cad1bda9ecd360539d184103dfd2f
  Stored in directory: c:\users\admin\appdata\local\pip\cache\wheels\89\e5\de\6966007cf2
23872eedfbebbe0e074534e72e9128c8fd4b55eb
Successfully built pydotplus
Installing collected packages: pydotplus
Successfully installed pydotplus-2.0.2
Note: you may need to restart the kernel to use updated packages.
```

In [44]:
```
from sklearn.tree import export_graphviz
import pydotplus
from six import StringIO
```

In [45]:
```
tree = model.estimators_[5]
```

In [47]:
```
dot_data = StringIO()
export_graphviz(tree,out_file = dot_data, filled = True,rounded = True, feature_names =
```

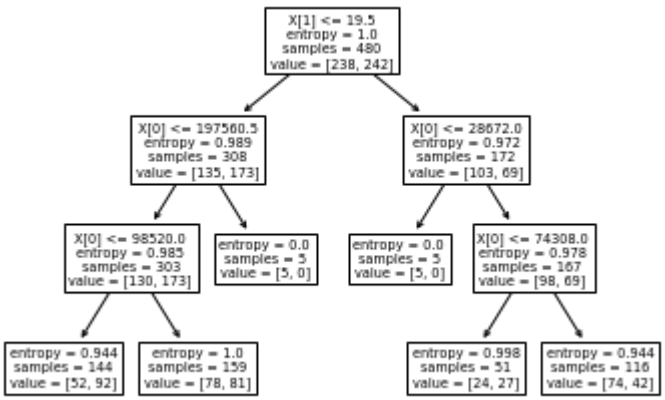
In [48]:
```
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
```

In [49]:
```
model = DecisionTreeClassifier(criterion = 'entropy',max_depth=3)
model.fit(x_train,y_train)
```

Out[49]:
```
DecisionTreeClassifier(criterion='entropy', max_depth=3)
```

In [50]:
```
from sklearn import tree
```

In [51]:
```
tree.plot_tree(model);
```



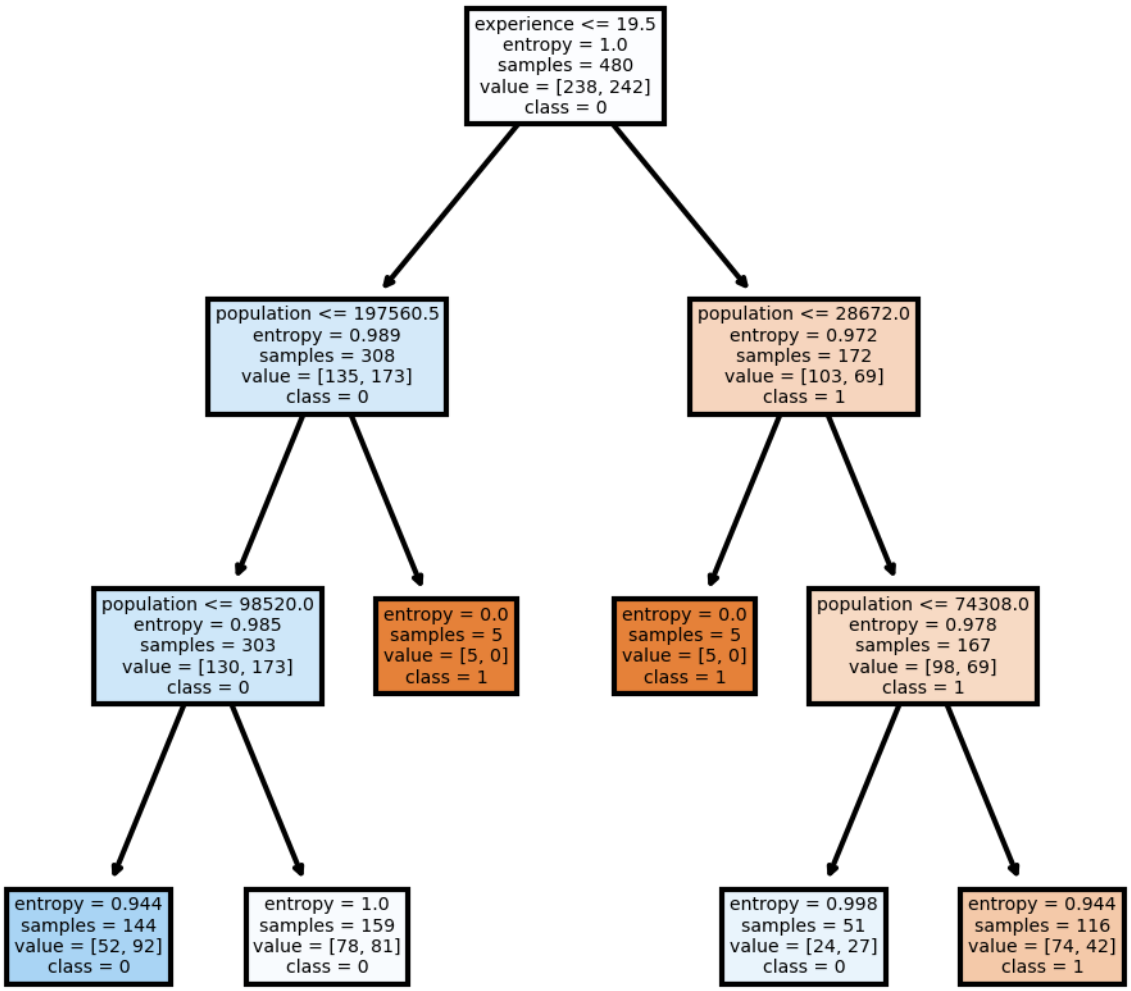
In [52]:
```
colnames = list(df.columns)
colnames
```

Out[52]:
```
['population',
 'experience',
 'Undergrad_YES',
 'Marital.Status_Married',
```

```
'Marital.Status_Single',
'Urban_YES',
'TaxInc_Good']
```

In [53]:

```
fn=['population','experience','Undergrad_YES','Marital.Status_Married','Marital.Status_
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);
```



In [54]:

```
preds = model.predict(x_test) # predicting on test data set
pd.Series(preds).value_counts() # getting the count of each category
```

Out[54]:

```
1    85
0    35
dtype: int64
```

In [55]:
```
preds
```

Out[55]:
```
array([0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1,
       0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1,
       0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0,
       1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1,
       1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       0, 0, 1, 0, 1, 1, 1, 1, 0, 1], dtype=uint8)
```

In [56]:
```
pd.crosstab(y_test,preds)
```

Out[56]:

col_0	0	1
Urban_YES		
0	16	44
1	19	41

In [57]:
```
np.mean(preds==y_test)
```

Out[57]:
```
0.475
```

In [58]:
```
from sklearn.tree import DecisionTreeClassifier
model_gini = DecisionTreeClassifier(criterion='gini', max_depth=3)
```

In [59]:
```
model_gini.fit(x_train, y_train)
```

Out[59]:
```
DecisionTreeClassifier(max_depth=3)
```

In [60]:
```
pred=model.predict(x_test)
np.mean(preds==y_test)
```

Out[60]:
```
0.475
```

In [61]:
```
from sklearn.tree import DecisionTreeRegressor
```

In [62]:
```
array = df.values
X = array[:,0:3]
y = array[:,3]
```

In [63]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=
```

In [64]:
```
model = DecisionTreeRegressor()
model.fit(X_train, y_train)
```

Out[64]:
```
DecisionTreeRegressor()
```

In [65]:
```
model.score(X_test,y_test)
```

Out[65]:
```
-0.8931902985074629
```

In [ ]:
```

```