In [1]:

```
import pandas as pd
import matplotlib.pyplot as plt
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 [2]:

```
data = pd.read_csv('Fraud_check.csv')
```

In [3]:

```
# EDA and Data Preprocessing
```

In [4]:

```
data.head()
```

Out[4]:

	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]:

```
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]:

```
data.columns
```

Out[6]:

```
Index(['Undergrad', 'Marital.Status', 'Taxable.Income', 'City.Population',
       'Work.Experience', 'Urban'],
      dtype='object')
```

In [7]:

```
# Renaming columns
data = data.rename({'Undergrad':'under_grad', 'Marital.Status':'marital_status', 'Taxable.I
                    'City.Population':'city_population', 'Work.Experience':'work_experience
data.head()
```

Out[7]:

	under_grad	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 [8]:

```
data.describe()
```

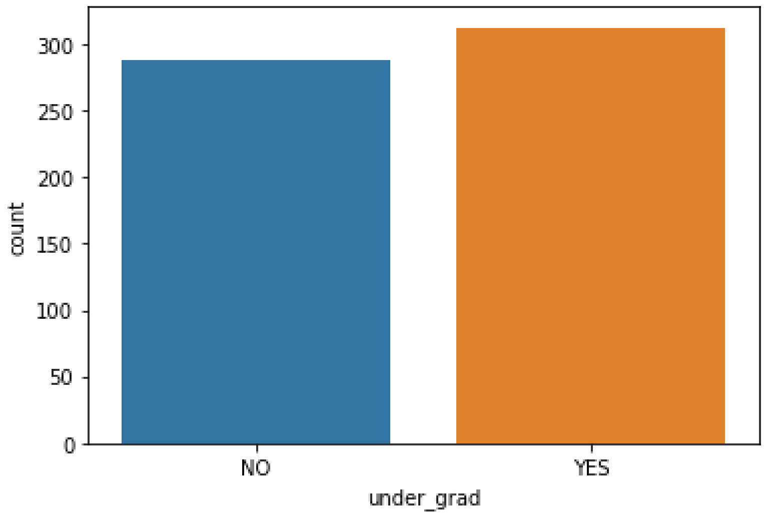
Out[8]:

	taxable_income	city_population	work_experience
count	600.000000	600.000000	600.000000
mean	55208.375000	108747.368333	15.558333
std	26204.827597	49850.075134	8.842147
min	10003.000000	25779.000000	0.000000
25%	32871.500000	66966.750000	8.000000
50%	55074.500000	106493.500000	15.000000
75%	78611.750000	150114.250000	24.000000
max	99619.000000	199778.000000	30.000000

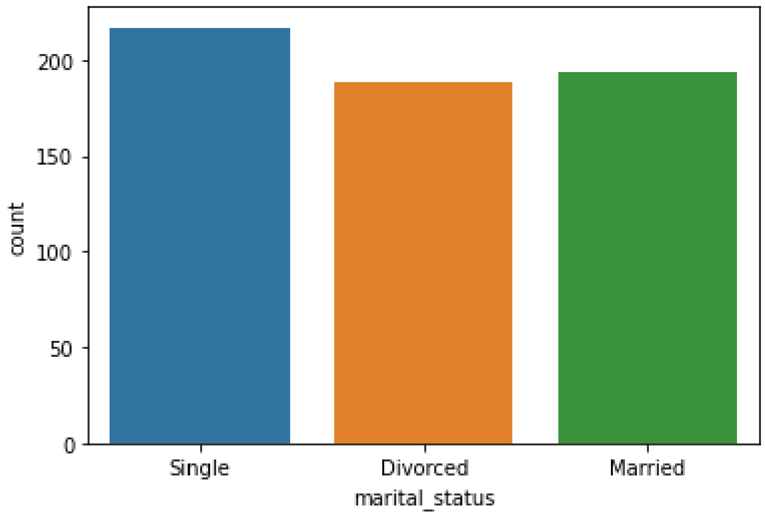
In [9]:

```
# checking count of categories for categorical columns colums
import seaborn as sns

sns.countplot(data['under_grad'])
plt.show()

sns.countplot(data['marital_status'])
plt.show()

sns.countplot(data['urban'])
plt.show()
```

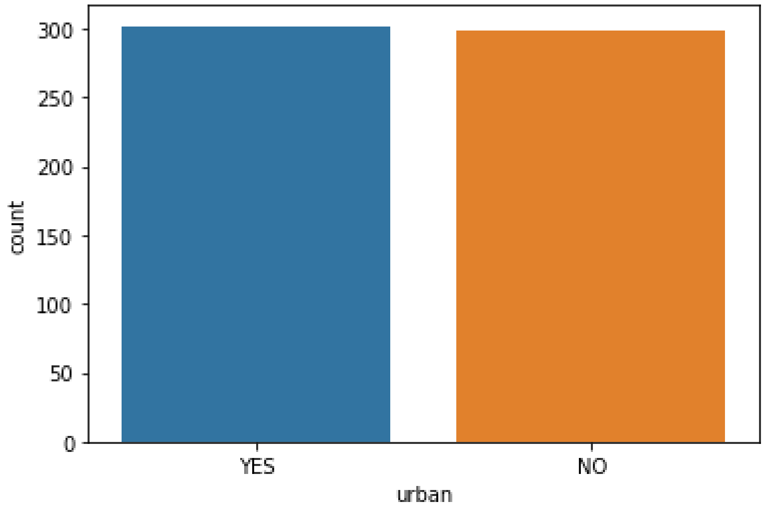
```
C:\Users\sowmya sandeep\anaconda3\lib\site-packages\seaborn\_decorators.py:3
6: FutureWarning: Pass the following variable as a keyword arg: x. From vers
ion 0.12, the only valid positional argument will be `data`, and passing oth
er arguments without an explicit keyword will result in an error or misinter
pretation.
  warnings.warn(
```



```
C:\Users\sowmya sandeep\anaconda3\lib\site-packages\seaborn\_decorators.py:3
6: FutureWarning: Pass the following variable as a keyword arg: x. From vers
ion 0.12, the only valid positional argument will be `data`, and passing oth
er arguments without an explicit keyword will result in an error or misinter
pretation.
  warnings.warn(
```



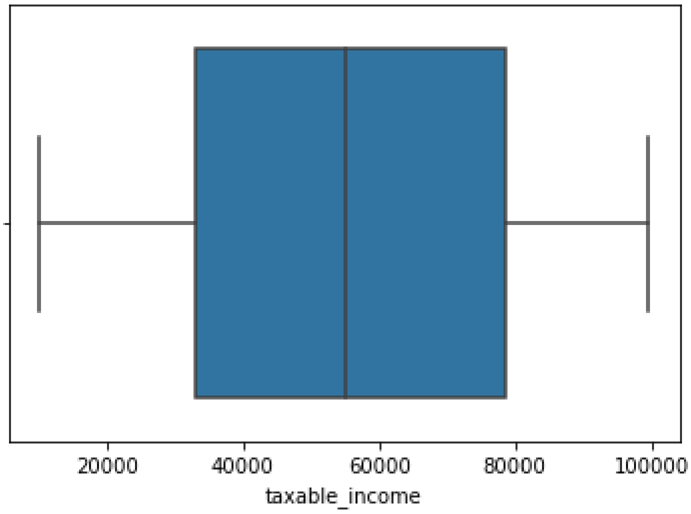
```
C:\Users\sowmya sandeep\anaconda3\lib\site-packages\seaborn\_decorators.py:3
6: FutureWarning: Pass the following variable as a keyword arg: x. From vers
ion 0.12, the only valid positional argument will be `data`, and passing oth
er arguments without an explicit keyword will result in an error or misinter
pretation.
  warnings.warn(
```



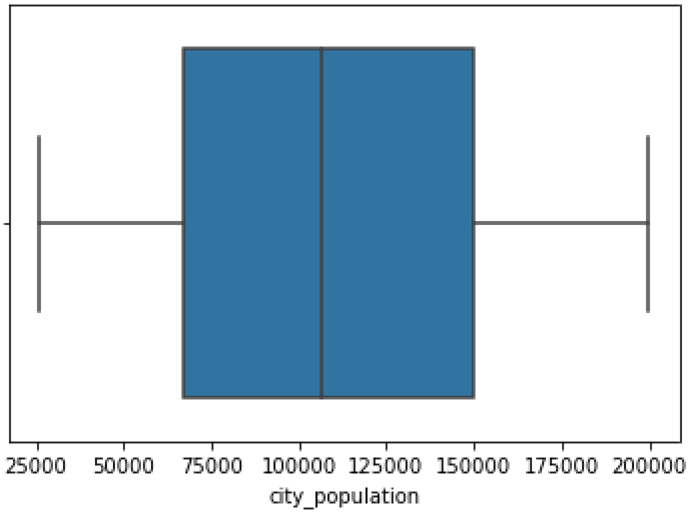
In [10]:

```
# Checking for outliers in numerical data
sns.boxplot(data['taxable_income'])
plt.show()

sns.boxplot(data['city_population'])
plt.show()

sns.boxplot(data['work_experience'])
plt.show()
```

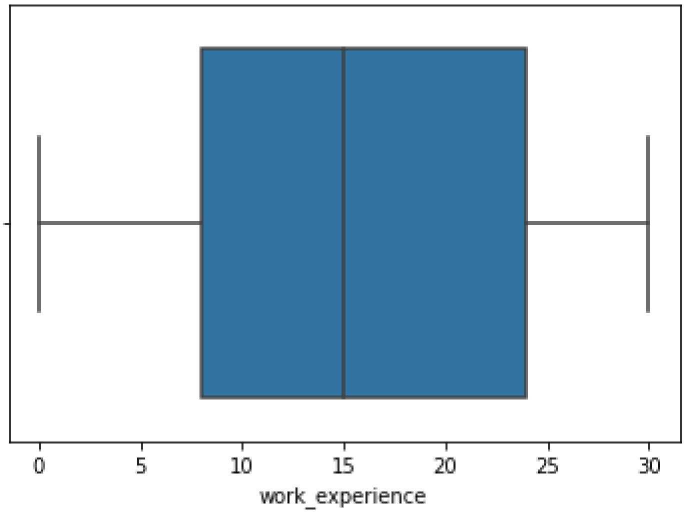
```
C:\Users\sowmya sandeep\anaconda3\lib\site-packages\seaborn\_decorators.py:3
6: FutureWarning: Pass the following variable as a keyword arg: x. From vers
ion 0.12, the only valid positional argument will be `data`, and passing oth
er arguments without an explicit keyword will result in an error or misinter
pretation.
  warnings.warn(
```



```
C:\Users\sowmya sandeep\anaconda3\lib\site-packages\seaborn\_decorators.py:3
6: FutureWarning: Pass the following variable as a keyword arg: x. From vers
ion 0.12, the only valid positional argument will be `data`, and passing oth
er arguments without an explicit keyword will result in an error or misinter
pretation.
  warnings.warn(
```

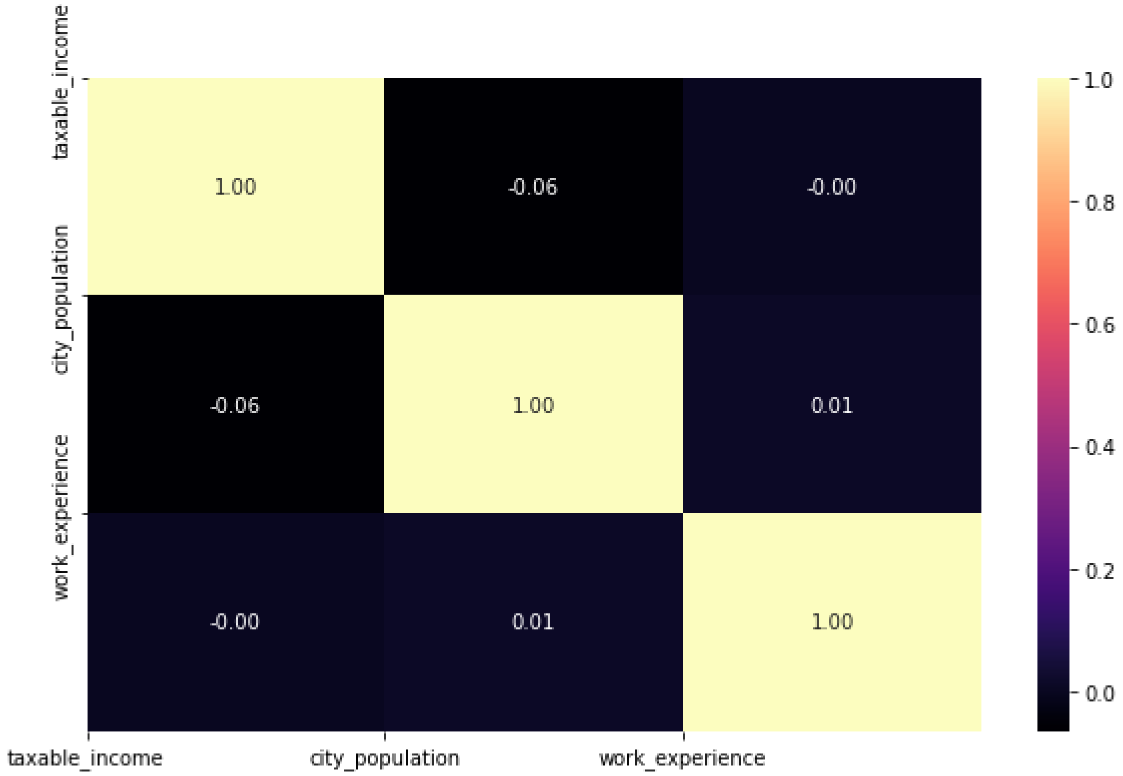


```
C:\Users\sowmya sandeep\anaconda3\lib\site-packages\seaborn\_decorators.py:3
6: FutureWarning: Pass the following variable as a keyword arg: x. From vers
ion 0.12, the only valid positional argument will be `data`, and passing oth
er arguments without an explicit keyword will result in an error or misinter
pretation.
  warnings.warn(
```



In [11]:

```
# Correlation analysis for data
corr = data.corr()
#Plot figsize
fig, ax = plt.subplots(figsize=(10, 6))
#Generate Heat Map, allow annotations and place floats in map
sns.heatmap(corr, cmap='magma', annot=True, fmt=".2f")
#Apply xticks
plt.xticks(range(len(corr.columns)), corr.columns);
#Apply yticks
plt.yticks(range(len(corr.columns)), corr.columns)
#show plot
plt.show()
```



In [12]:

```
# Converting taxable_income <= 30000 as "Risky" and others are "Good"
data['taxable_category'] = pd.cut(x = data['taxable_income'], bins = [10002,30000,99620], l
data
```

Out[12]:

	under_grad	marital_status	taxable_income	city_population	work_experience	urban	taxa
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 × 7 columns

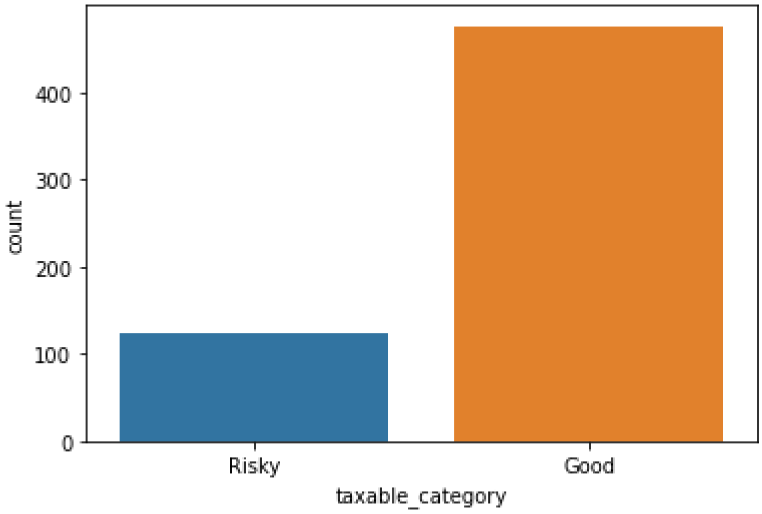
In [13]:

```
sns.countplot(data['taxable_category'])
```

```
C:\Users\sowmya sandeep\anaconda3\lib\site-packages\seaborn\_decorators.py:3
6: FutureWarning: Pass the following variable as a keyword arg: x. From vers
ion 0.12, the only valid positional argument will be `data`, and passing oth
er arguments without an explicit keyword will result in an error or misinter
pretation.
  warnings.warn(
```

Out[13]:

```
<AxesSubplot:xlabel='taxable_category', ylabel='count'>
```



In [14]:

```
data['taxable_category'].value_counts()
```

Out[14]:

```
Good     476
Risky    124
Name: taxable_category, dtype: int64
```

In [15]:

```
#encoding categorical data
label_encoder = preprocessing.LabelEncoder()

data['under_grad'] = label_encoder.fit_transform(data['under_grad'])
data['marital_status'] = label_encoder.fit_transform(data['marital_status'])
data['urban'] = label_encoder.fit_transform(data['urban'])
data['taxable_category'] = label_encoder.fit_transform(data['taxable_category'])
data.sample(10)
```

Out[15]:

	under_grad	marital_status	taxable_income	city_population	work_experience	urban	taxa
247	1	0	11865	108245	1	0	
310	1	0	59615	189435	22	1	
529	0	0	33116	83388	14	0	
10	0	2	29732	102602	19	1	
29	1	0	94033	41863	30	1	
5	0	0	33329	116382	0	0	
236	1	1	46070	193193	3	0	
332	1	0	98240	84132	1	0	
581	1	0	31085	57473	10	1	
419	0	1	68269	138074	20	0	

In [16]:

```
# dropping column taxable_income
data1 = data.drop('taxable_income', axis = 1)
data1
```

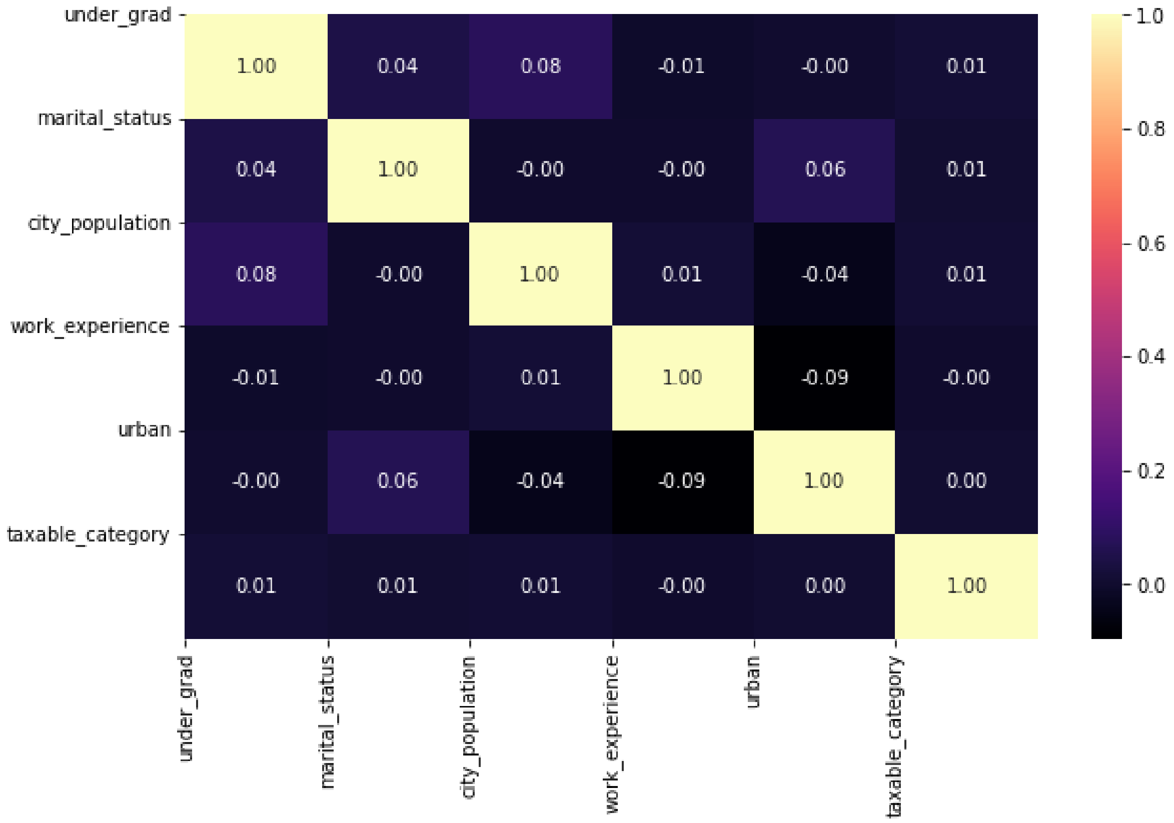
Out[16]:

	under_grad	marital_status	city_population	work_experience	urban	taxable_category
0	0	2	50047	10	1	0
1	1	0	134075	18	1	0
2	0	1	160205	30	1	0
3	1	2	193264	15	1	0
4	0	1	27533	28	0	0
...	...	...	...	...	...	...
595	1	0	39492	7	1	0
596	1	0	55369	2	1	0
597	0	0	154058	0	1	0
598	1	1	180083	17	0	0
599	0	0	158137	16	0	0

600 rows × 6 columns

In [17]:

```
# Correlation analysis for data11
corr = data1.corr()
#Plot figsize
fig, ax = plt.subplots(figsize=(10, 6))
#Generate Heat Map, allow annotations and place floats in map
sns.heatmap(corr, cmap='magma', annot=True, fmt=".2f")
#Apply xticks
plt.xticks(range(len(corr.columns)), corr.columns);
#Apply yticks
plt.yticks(range(len(corr.columns)), corr.columns)
#show plot
plt.show()
```



In [18]:

```
# Dividing data into independent variables and dependent variable
X = data1.drop('taxable_category', axis = 1)
y = data1['taxable_category']
```

In [19]:

```
X
```

Out[19]:

	under_grad	marital_status	city_population	work_experience	urban
0	0	2	50047	10	1
1	1	0	134075	18	1
2	0	1	160205	30	1
3	1	2	193264	15	1
4	0	1	27533	28	0
...	...	...	...	...	...
595	1	0	39492	7	1
596	1	0	55369	2	1
597	0	0	154058	0	1
598	1	1	180083	17	0
599	0	0	158137	16	0

600 rows × 5 columns

In [20]:

```
y
```

Out[20]:

```
0      0
1      0
2      0
3      0
4      0
      ..
595    0
596    0
597    0
598    0
599    0
Name: taxable_category, Length: 600, dtype: int32
```

In [21]:

```
# Splitting data into training and testing data
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size= 0.33, random_state= 42
```

In [22]:

```
x_train
```

Out[22]:

	under_grad	marital_status	city_population	work_experience	urban
509	0	1	65531	27	1
149	0	2	49505	25	0
124	1	0	139324	13	0
428	1	1	128266	24	1
465	0	0	116282	21	0
...	...	...	...	...	...
71	0	2	105680	22	0
106	1	2	58535	20	1
270	0	1	130680	5	0
435	0	0	111774	4	1
102	1	0	91488	23	0

402 rows × 5 columns

In [23]:

```
x_test
```

Out[23]:

	under_grad	marital_status	city_population	work_experience	urban
110	0	2	32450	19	1
419	0	1	138074	20	0
565	0	0	31064	28	0
77	1	1	118344	26	0
181	0	0	36116	20	0
...	...	...	...	...	...
231	1	2	153147	2	0
403	0	0	130912	27	1
278	0	1	114823	11	0
472	0	1	151963	11	1
350	0	1	89949	25	0

198 rows × 5 columns

In [25]:

```
y_train
```

Out[25]:

```
509    1
149    0
124    0
428    1
465    1
      ..
71     0
106    1
270    0
435    0
102    0
Name: taxable_category, Length: 402, dtype: int32
```

In [26]:

```
y_test
```

Out[26]:

```
110    1
419    0
565    0
77     0
181    1
      ..
231    0
403    0
278    1
472    0
350    0
Name: taxable_category, Length: 198, dtype: int32
```

In [27]:

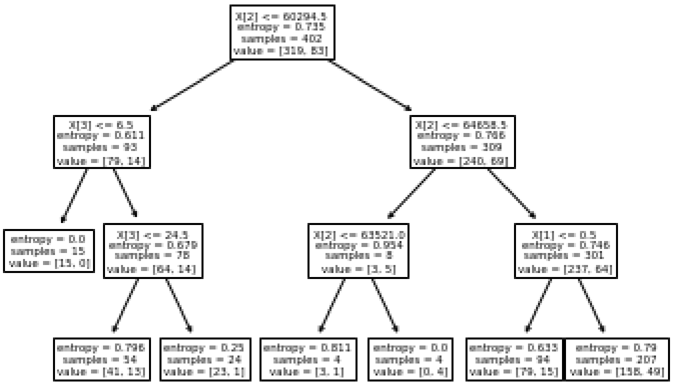
```
# Building model based on C5.0 Algorithm
model_c5 = DecisionTreeClassifier(criterion = 'entropy', max_depth= 3)
model_c5.fit(x_train, y_train)
```

Out[27]:

```
▾                  DecisionTreeClassifier

DecisionTreeClassifier(criterion='entropy', max_depth=3)
```

In [28]:

```
# Plotting Decision tree
tree.plot_tree(model_c5);
```



In [29]:

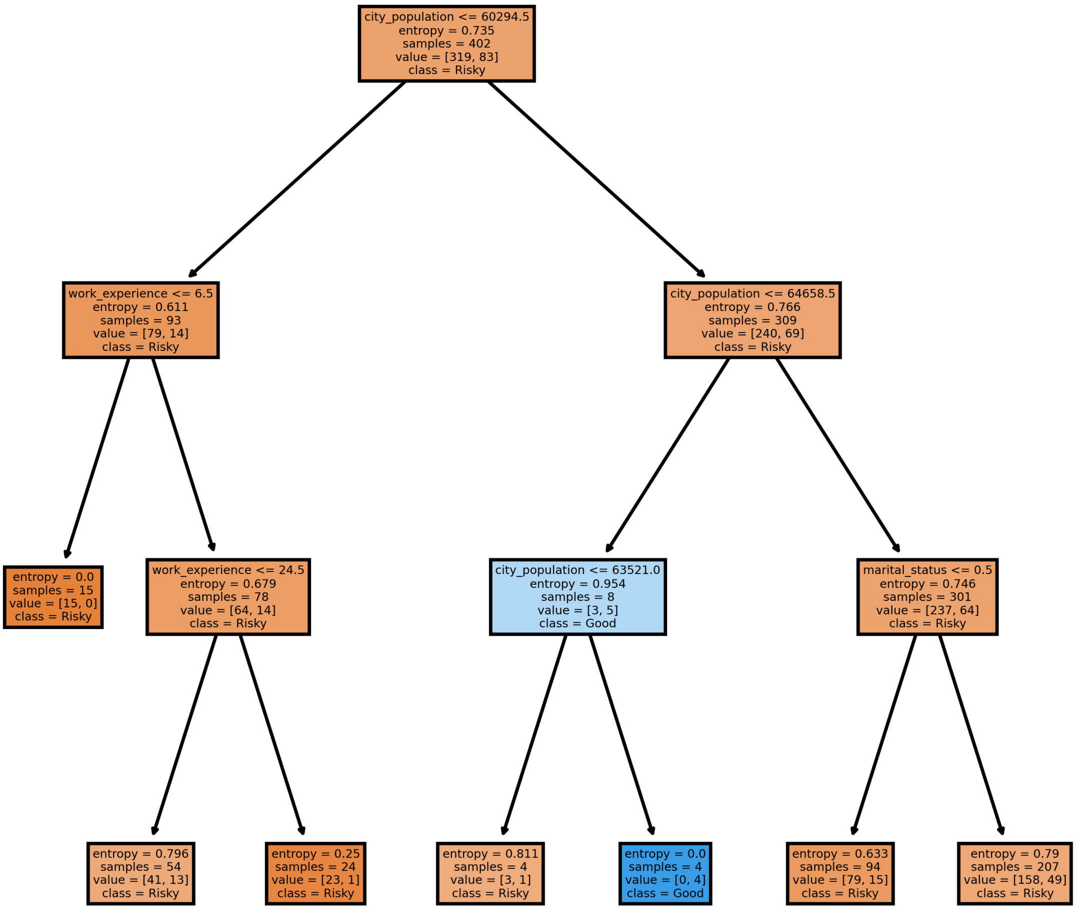
```
data1.columns
```

Out[29]:

```
Index(['under_grad', 'marital_status', 'city_population', 'work_experience',
       'urban', 'taxable_category'],
      dtype='object')
```

In [30]:

```
fn=['under_grad', 'marital_status', 'city_population', 'work_experience',
       'urban', 'taxable_category']
cn=['Risky', 'Good']
fig, axes = plt.subplots(nrows = 1,ncols = 1,figsize = (6,6), dpi=600)
tree.plot_tree(model_c5,
               feature_names = fn,
               class_names=cn,
               filled = True);
```



In [31]:

```
# Predicting Data
preds = model_c5.predict(x_test)
pd.Series(preds).value_counts()
```

Out[31]:

```
0    197
1      1
dtype: int64
```

In [32]:

```
preds
```

Out[32]:

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

In [33]:

```
# Creating cross tables for checking model
pd.crosstab(y_test, preds)
```

Out[33]:

col_0	0	1
taxable_category		
0	156	1
1	41	0

In [34]:

```
# Checking accuracy of model
model_c5.score(x_test, y_test)
```

Out[34]:

```
0.7878787878787878
```

In [35]:

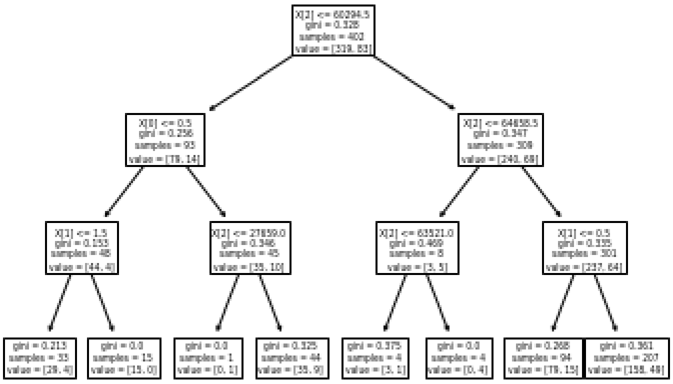
```
# Building model based on CART Algorithm
model_CART = DecisionTreeClassifier(criterion = 'gini', max_depth= 3)
model_CART.fit(x_train, y_train)
```

Out[35]:

```
▾ DecisionTreeClassifier
DecisionTreeClassifier(max_depth=3)
```

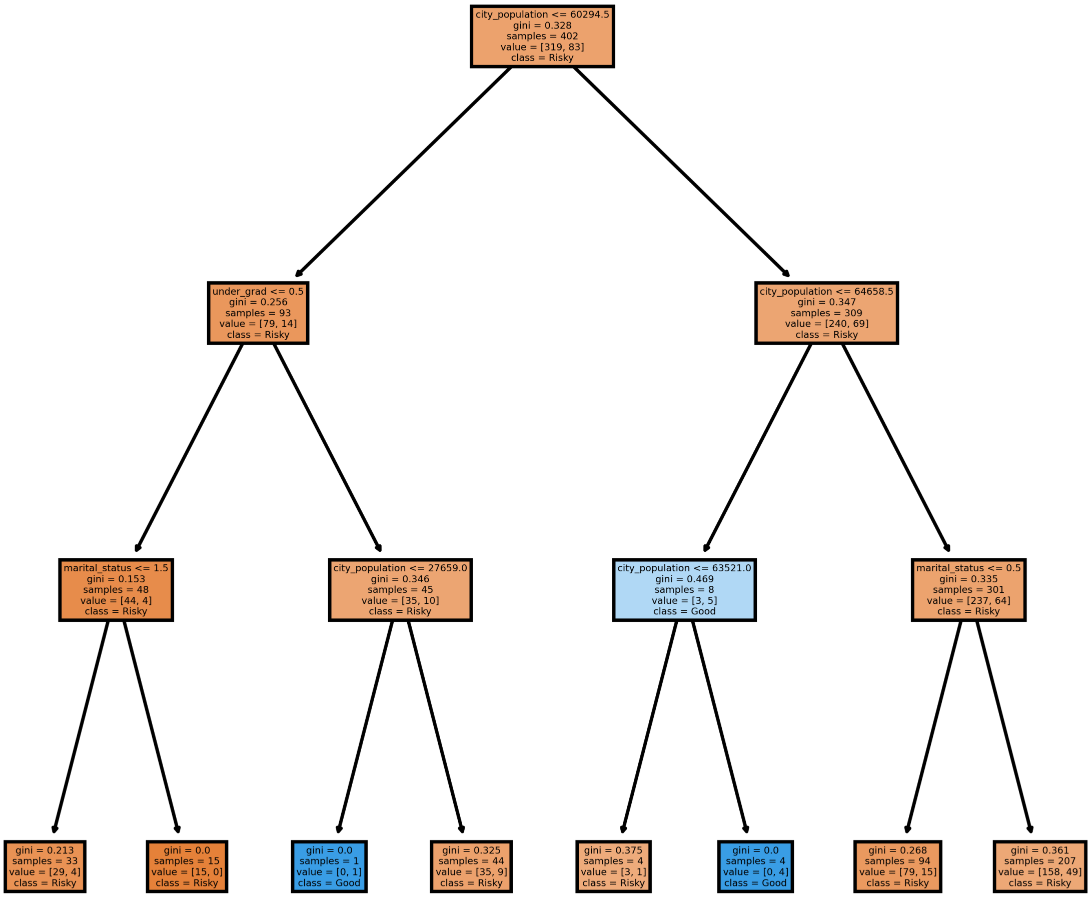
In [36]:

```
# Plotting Decision tree
tree.plot_tree(model_CART);
```



In [37]:

```
fn=['under_grad', 'marital_status', 'city_population', 'work_experience',
       'urban', 'taxable_category']
cn=['Risky', 'Good']
fig, axes = plt.subplots(nrows = 1,ncols = 1,figsize = (6,6), dpi=600)
tree.plot_tree(model_CART,
               feature_names = fn,
               class_names=cn,
               filled = True);
```



In [38]:

```
# Predicting Data
preds = model_CART.predict(x_test)
pd.Series(preds).value_counts()
```

Out[38]:

```
0    197
1      1
dtype: int64
```

In [39]:

```
preds
```

Out[39]:

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

In [40]:

```
# Creating cross tables for checking model
pd.crosstab(y_test, preds)
```

Out[40]:

col_0	0	1
taxable_category		
0	156	1
1	41	0

In [41]:

```
# Checking accuracy of model
model_CART.score(x_test, y_test)
```

Out[41]:

```
0.7878787878787878
```

In [ ]:

```

```