Import Necessary Libraries

In [1]:

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from io import StringIO
import pydotplus
import seaborn as sns
import matplotlib.image as mpimg
from sklearn.tree import export_graphviz
from sklearn.tree import DecisionTreeClassifier as DT
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn import tree
from IPython.display import Image
```

Data Collection

In [2]:

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

In [4]:

```
fraud=df.copy()
fraud.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]:

```
fraud.describe().T
```

Out[5]:

	count	mean	std	min	25%	50%	75%	m
Taxable.Income	600.0	55208.375000	26204.827597	10003.0	32871.50	55074.5	78611.75	9961
City.Population	600.0	108747.368333	49850.075134	25779.0	66966.75	106493.5	150114.25	19977
Work.Experience	600.0	15.558333	8.842147	0.0	8.00	15.0	24.00	3

In [6]:

```
fraud.isna().sum()
```

Out[6]:

```
Undergrad          0
Marital.Status     0
Taxable.Income     0
City.Population    0
Work.Experience    0
Urban              0
dtype: int64
```

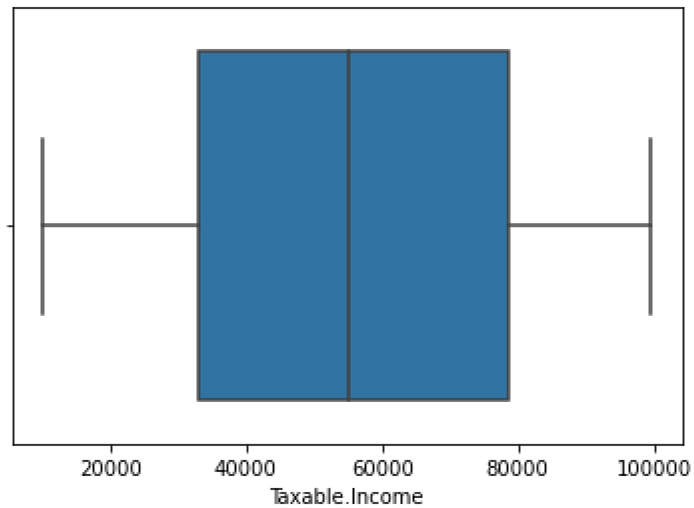
In [7]:

```
Out[7]: Undergrad          object
        Marital.Status     object
        Taxable.Income      int64
        City.Population     int64
        Work.Experience     int64
        Urban              object
        dtype: object
```

```
In [9]: import warnings
        warnings.filterwarnings('ignore')
```

Outlier Check

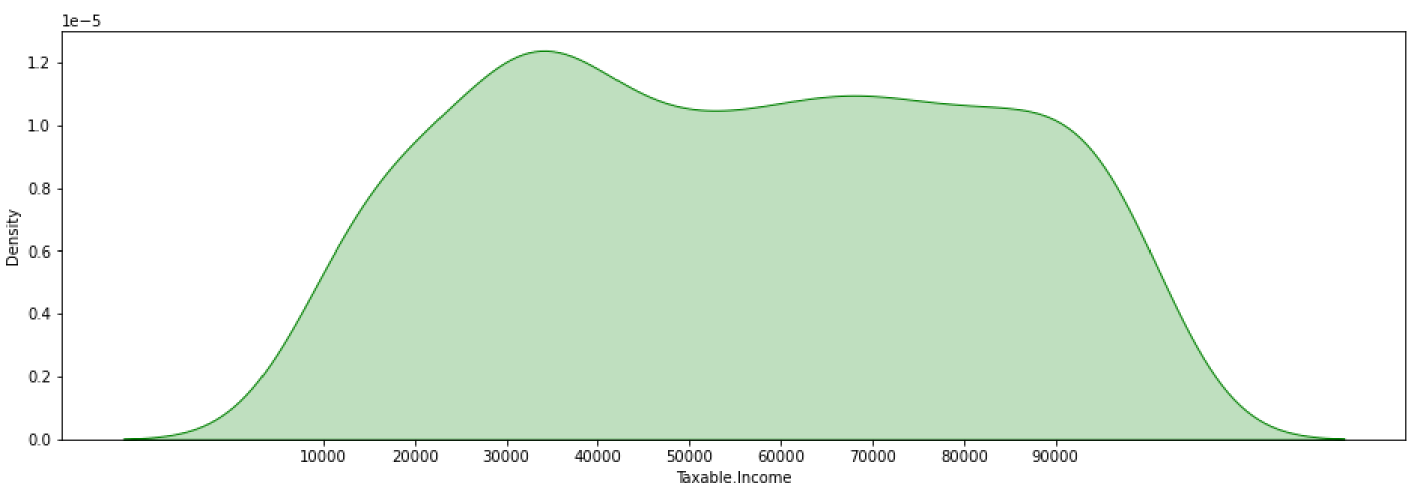
```
In [10]: ax = sns.boxplot(fraud['Taxable.Income'])
```




```
In [11]: plt.rcParams["figure.figsize"] = 9,5
```

```
In [12]: plt.figure(figsize=(16,5))
         print("Skew: {}".format(fraud['Taxable.Income'].skew()))
         print("Kurtosis: {}".format(fraud['Taxable.Income'].kurtosis()))
         ax = sns.kdeplot(fraud['Taxable.Income'],shade=True,color='g')
         plt.xticks([i for i in range(10000,100000,10000)])
         plt.show()
```

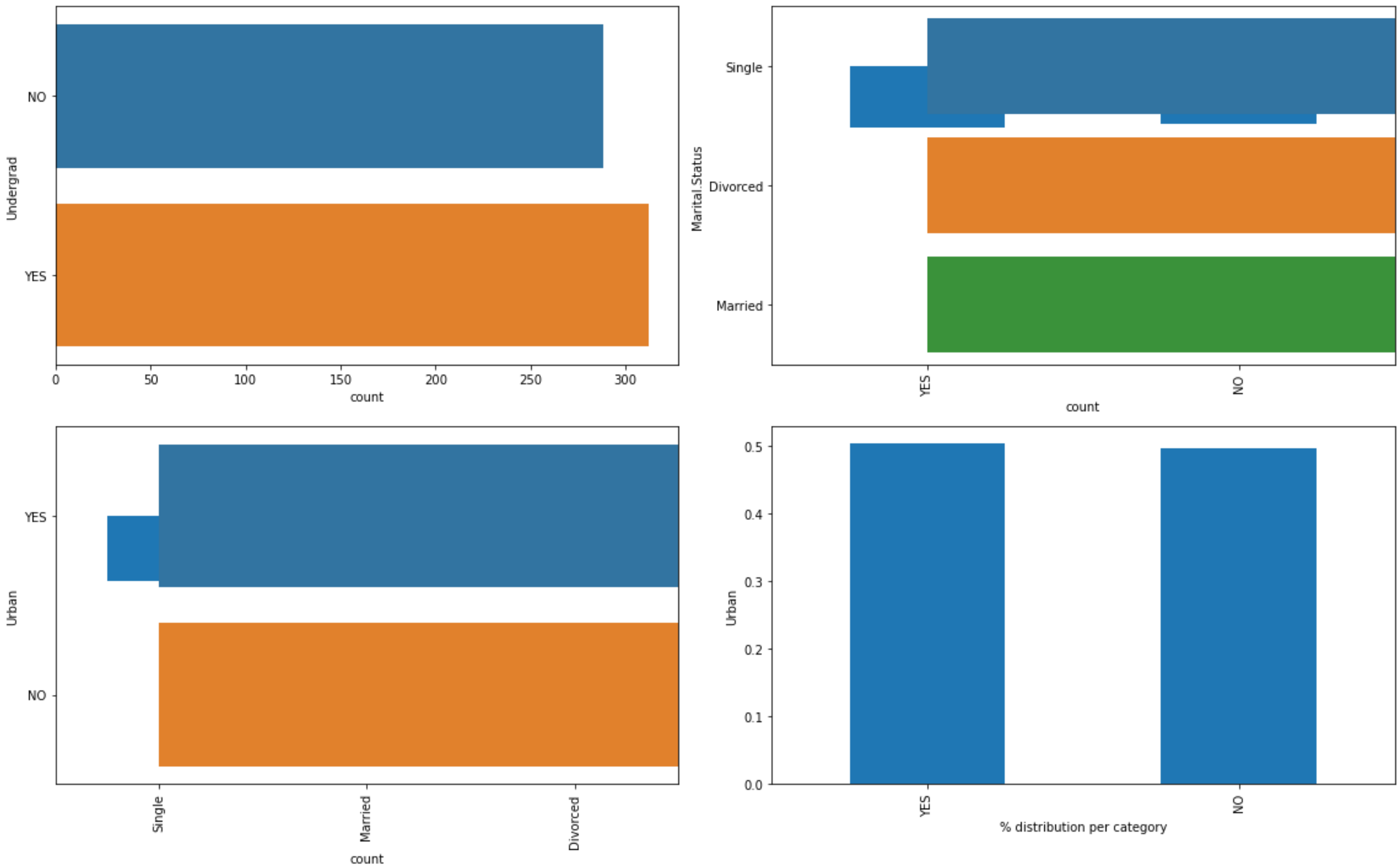
```
Skew: 0.030014788906377175
Kurtosis: -1.1997824607083138
```



Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

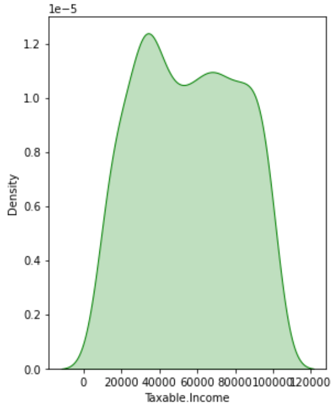
In [13]:
```python
obj_colum = fraud.select_dtypes(include='object').columns.tolist()
```

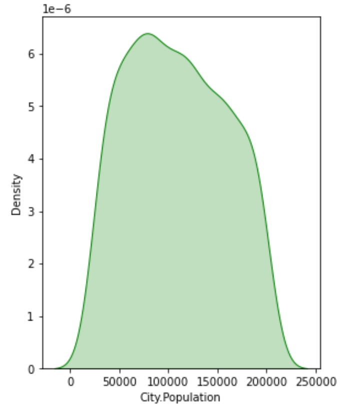
In [14]:
```python
plt.figure(figsize=(16,10))
for i,col in enumerate(obj_colum,1):
    plt.subplot(2,2,i)
    sns.countplot(data=fraud,y=col)
    plt.subplot(2,2,i+1)
    fraud[col].value_counts(normalize=True).plot.bar()
    plt.ylabel(col)
    plt.xlabel('% distribution per category')
plt.tight_layout()
plt.show()
```

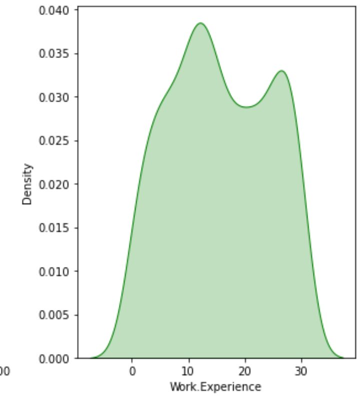


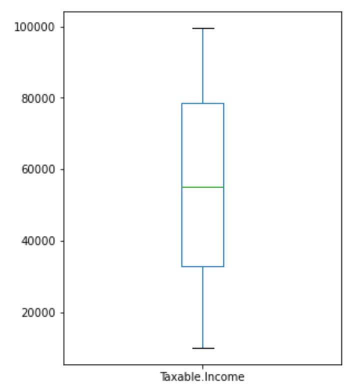
In [15]:
```python
num_columns = fraud.select_dtypes(exclude='object').columns.tolist()
```

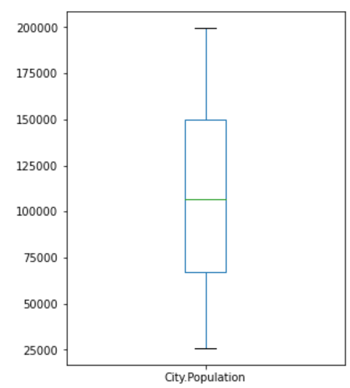
In [17]:
```python
plt.figure(figsize=(18,40))
for i,col in enumerate(num_columns,1):
    plt.subplot(8,4,i)
    sns.kdeplot(df[col],color='g',shade=True)
    plt.subplot(8,4,i+10)
    df[col].plot.box()
plt.tight_layout()
plt.show()
num_data = df[num_columns]
pd.DataFrame(data=[num_data.skew(),num_data.kurtosis()],index=['skewness','kurtosis'])
```

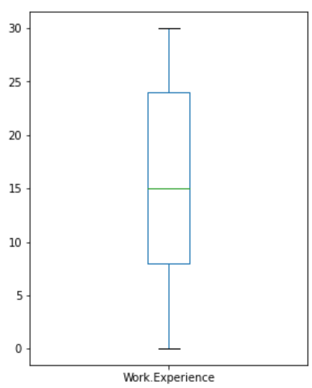












Out[17]:

	Taxable.Income	City.Population	Work.Experience
skewness	0.030015	0.125009	0.018529
kurtosis	-1.199782	-1.120154	-1.167524

In [18]:

```
fraud = pd.get_dummies(fraud, columns = ['Undergrad','Marital.Status','Urban'])
```

In [19]:

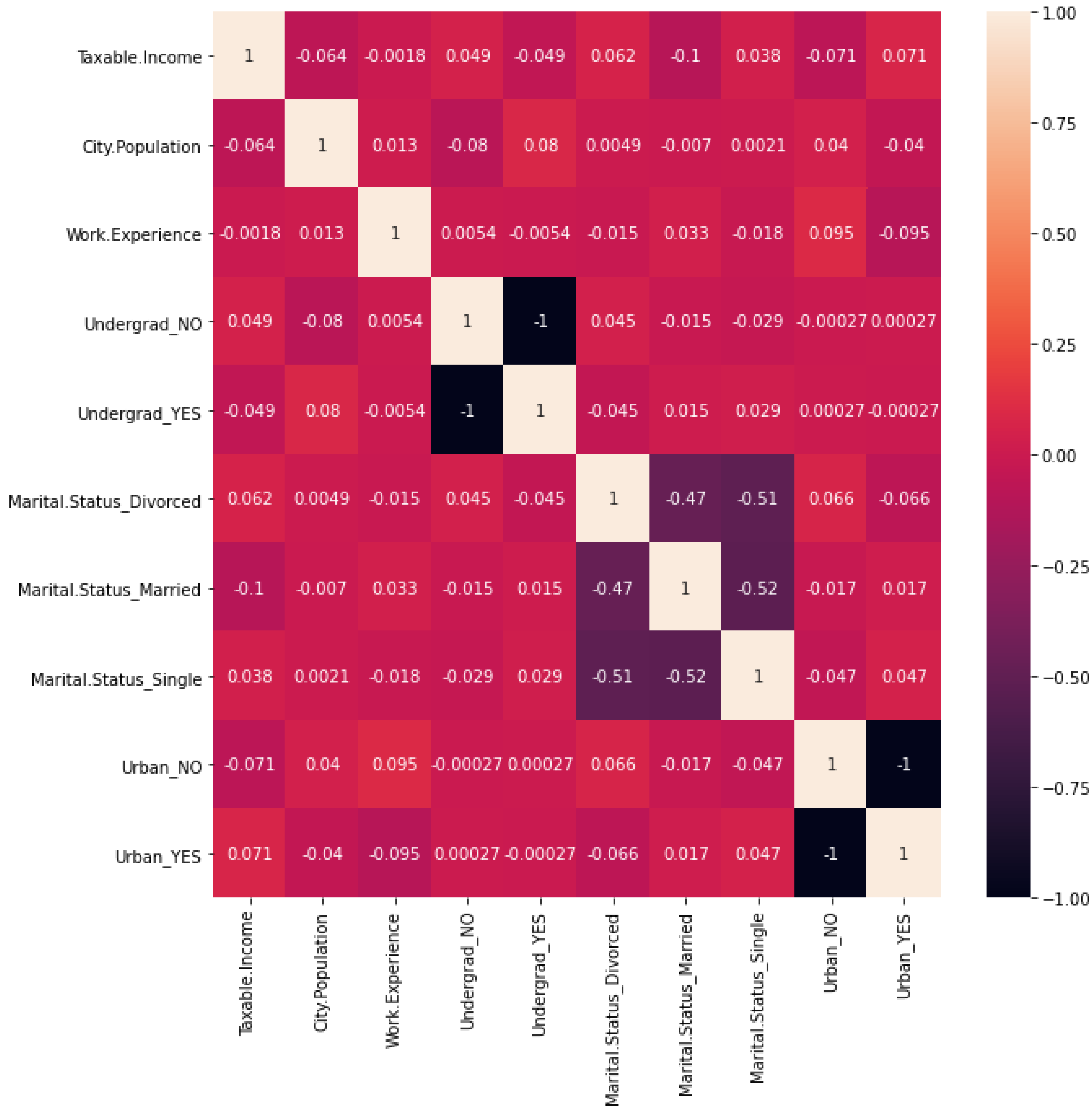
```
corr = fraud.corr()
```

In [20]:

```
plt.figure(figsize=(10,10))
```

```python
sns.heatmap(corr,annot=True)
```

Out[20]: <AxesSubplot:>



Decision Tree Model

In [22]:
```python
fraud['Taxable.Income']=pd.cut(fraud['Taxable.Income'],bins=[0,30000,100000],labels=['ris
```

In [23]:
```python
list(fraud.columns)
```

Out[23]:
```
['Taxable.Income',
 'City.Population',
 'Work.Experience',
 'Undergrad_NO',
 'Undergrad_YES',
 'Marital.Status_Divorced',
 'Marital.Status_Married',
 'Marital.Status_Single',
```

```
 'Urban_NO',
 'Urban_YES']
```

In [24]:
```
X = fraud.iloc[:,1:10]
y = fraud.iloc[:,0]
```

In [25]:
```
x_train,x_test,y_train,y_test = train_test_split(X,y,test_size = 0.2)
```

In [26]:
```
y_train.value_counts()
```

Out[26]:
```
good     375
risky    105
Name: Taxable.Income, dtype: int64
```

In [27]:
```
model = DT(criterion='entropy')
model.fit(x_train,y_train)
```

Out[27]:
```
DecisionTreeClassifier(criterion='entropy')
```

In [28]:
```
pred_train = model.predict(x_train)
```

In [29]:
```
accuracy_score(y_train,pred_train)
```

Out[29]:
```
1.0
```

In [30]:
```
confusion_matrix(y_train,pred_train)
```

Out[30]:
```
array([[375,   0],
       [  0, 105]], dtype=int64)
```

In [31]:
```
pred_test = model.predict(x_test)
accuracy_score(y_test,pred_test)
```

Out[31]:
```
0.6583333333333333
```

In [32]:
```
confusion_matrix(y_test,pred_test)
```

Out[32]:
```
array([[76, 25],
       [16,  3]], dtype=int64)
```

In [33]:
```
df_t=pd.DataFrame({'Actual':y_test, 'Predicted':pred_test})
```

In [34]:
```
df_t
```

Out[34]:

	Actual	Predicted
356	risky	good
172	risky	good
7	good	risky
87	risky	good

	Actual	Predicted
259	risky	good
...	...	...
102	good	risky
173	good	risky
553	good	good
502	good	good
60	good	good

120 rows × 2 columns

In [36]:
```
cols = list(fraud.columns)
```

In [37]:
```
predictors = cols[1:10]
target = cols[0]
```

In [39]:
```
dot_data = StringIO()
```

In [40]:
```
export_graphviz(model, out_file = dot_data ,filled = True,rounded =True,feature_names = p
```

In [41]:
```
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
```

In [42]:
```
graph.write_png('fraud_full.png')
```

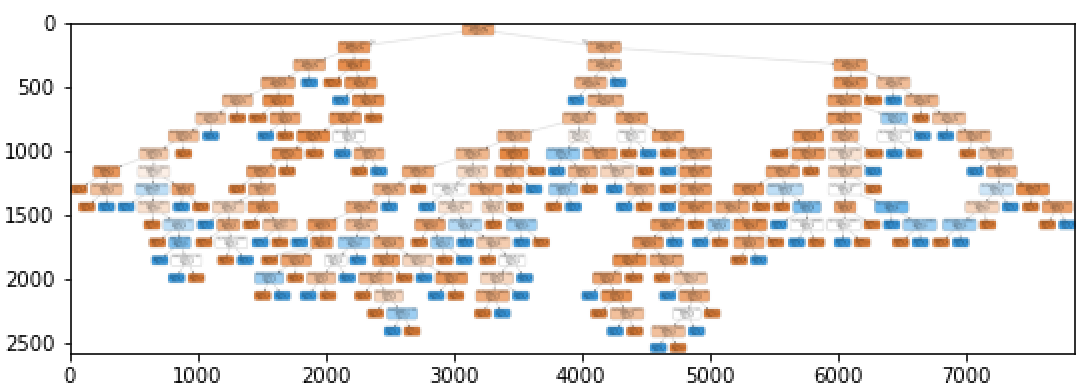
Out[42]: True

Conclusion

In [43]:
```
img = mpimg.imread('fraud_full.png')
```

In [44]:
```
plt.imshow(img)
```

Out[44]: <matplotlib.image.AxesImage at 0x1f248641340>



Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

In [45]:
```
model.feature_importances_
```

Out[45]:
```
array([0.53209575, 0.27991161, 0.03494171, 0.02333879, 0.00957776,
       0.03243444, 0.02260328, 0.0366056 , 0.02849107])
```

In [47]:
```
fi = pd.DataFrame({'feature': list(x_train.columns),
                   'importance': model.feature_importances_}).\
                    sort_values('importance', ascending = False)
```

In [48]:
```
fi
```

Out[48]:

	feature	importance
0	City.Population	0.532096
1	Work.Experience	0.279912
7	Urban_NO	0.036606
2	Undergrad_NO	0.034942
5	Marital.Status_Married	0.032434
8	Urban_YES	0.028491
3	Undergrad_YES	0.023339
6	Marital.Status_Single	0.022603
4	Marital.Status_Divorced	0.009578

In [ ]:

In [ ]: