In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [3]:

```
fc= pd.read_csv('Fraud_check (1).csv')
fc
```

Out[3]:

	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 [4]:

```
# initial analysis
```

In [5]:

```
fc.shape
```

Out[5]:

```
(600, 6)
```

In [7]:

```
fc.dtypes
```

Out[7]:

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

In [8]:

```
fc.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 [9]:

```
fc.describe()
```

Out[9]:

	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 [10]:

```
fc.corr()
```

Out[10]:

	Taxable.Income	City.Population	Work.Experience
Taxable.Income	1.000000	-0.064387	-0.001818
City.Population	-0.064387	1.000000	0.013135
Work.Experience	-0.001818	0.013135	1.000000

In [11]:

```
fc.isna().sum()
```

Out[11]:

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

In [12]:

```
# Converting Taxable.Income <=30000 as "Risky" and others are "Good".
fc['Taxable.Income']=pd.cut(x=fc['Taxable.Income'], bins=[10002,30000,99620], labels=['Risk
fc
```

Out[12]:

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

600 rows × 6 columns

In [13]:

```
fc['Taxable.Income'].value_counts()
```

Out[13]:

```
Good     476
Risky    124
Name: Taxable.Income, dtype: int64
```

In [14]:

```
# Encoding categorial data
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
```

In [15]:

```
fc['Undergrad']=le.fit_transform(fc['Undergrad'])
fc['Marital.Status']=le.fit_transform(fc['Marital.Status'])
fc['Taxable.Income']=le.fit_transform(fc['Taxable.Income'])
fc['Work.Experience']=le.fit_transform(fc['Work.Experience'])
fc['Urban']=le.fit_transform(fc['Urban'])
fc['City.Population']=le.fit_transform(fc['City.Population'])

fc
```

Out[15]:

	Undergrad	Marital.Status	Taxable.Income	City.Population	Work.Experience	Urban
0	0	2	0	84	10	1
1	1	0	0	398	18	1
2	0	1	0	481	30	1
3	1	2	0	574	15	1
4	0	1	0	4	28	0
...	...	...	...	...	...	...
595	1	0	0	55	7	1
596	1	0	0	107	2	1
597	0	0	0	459	0	1
598	1	1	0	533	17	0
599	0	0	0	477	16	0

600 rows × 6 columns

In [16]:

```
X=fc.drop(labels='Taxable.Income',axis=1)
y=fc[['Taxable.Income']]
```

In [19]:

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.20,random_state=12)
```

In [20]:

```
X_train.shape,y_train.shape
```

Out[20]:

```
((480, 5), (480, 1))
```

In [21]:

```
X_test.shape,y_test.shape
```

Out[21]:

```
((120, 5), (120, 1))
```

In [23]:

```
import warnings
warnings.filterwarnings('ignore')
```

In [24]:

```
from sklearn.ensemble import RandomForestClassifier
rf_model = RandomForestClassifier(n_estimators=100, max_depth=3)
rf_model.fit(X_train,y_train)
```

Out[24]:

```
RandomForestClassifier(max_depth=3)
```

In [26]:

```
y_test_pred=rf_model.predict(X_test)
y_test_pred
```

Out[26]:

```
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])
```

In [27]:

```
rf_model.score(X_test,y_test)
```

Out[27]:

```
0.8916666666666667
```

In [28]:

```
from sklearn.metrics import accuracy_score, confusion_matrix
```

In [29]:

```
accuracy_score(y_test,y_test_pred)
```

Out[29]:

```
0.8916666666666667
```

In [30]:

```
confusion_matrix(y_test,y_test_pred)
```

Out[30]:

```
array([[107,   0],
       [ 13,   0]], dtype=int64)
```

In [ ]:

```
```