In [45]:

```
import pandas as pd
import warnings
warnings.filterwarnings('ignore')
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix,accuracy_score
from sklearn.model_selection import GridSearchCV
from sklearn.feature_selection import SelectFromModel,RFE
```

In [3]:

```
fraud_data=pd.read_csv("Fraud_check (1).csv")
fraud_data
```

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

Initial Investigation

In [4]:

```
fraud_data.dtypes
```

Out[4]:

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

In [5]:

```
fraud_data.shape
```

Out[5]:

```
(600, 6)
```

In [6]:

```
fraud_data.describe(include='all')
```

Out[6]:

	Undergrad	Marital.Status	Taxable.Income	City.Population	Work.Experience	Urban
count	600	600	600.000000	600.000000	600.000000	600
unique	2	3	NaN	NaN	NaN	2
top	YES	Single	NaN	NaN	NaN	YES
freq	312	217	NaN	NaN	NaN	302
mean	NaN	NaN	55208.375000	108747.368333	15.558333	NaN
std	NaN	NaN	26204.827597	49850.075134	8.842147	NaN
min	NaN	NaN	10003.000000	25779.000000	0.000000	NaN
25%	NaN	NaN	32871.500000	66966.750000	8.000000	NaN
50%	NaN	NaN	55074.500000	106493.500000	15.000000	NaN
75%	NaN	NaN	78611.750000	150114.250000	24.000000	NaN
max	NaN	NaN	99619.000000	199778.000000	30.000000	NaN

In [7]:

```
fraud_data.isna().sum()
```

Out[7]:

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

In [8]:

```
#Converted numerical into categorical as per the problem statement
fraud_data.loc[fraud_data['Taxable.Income']<= 30000, '<= 30000']='Risky'
fraud_data.loc[fraud_data['Taxable.Income']>30000, '> 30000']='Good'
```

In [9]:

```
fraud_data.head(12)
```

Out[9]:

	Undergrad	Marital.Status	Taxable.Income	City.Population	Work.Experience	Urban	<= 30000
0	NO	Single	68833	50047	10	YES	NaN
1	YES	Divorced	33700	134075	18	YES	NaN
2	NO	Married	36925	160205	30	YES	NaN
3	YES	Single	50190	193264	15	YES	NaN
4	NO	Married	81002	27533	28	NO	NaN
5	NO	Divorced	33329	116382	0	NO	NaN
6	NO	Divorced	83357	80890	8	YES	NaN
7	YES	Single	62774	131253	3	YES	NaN
8	NO	Single	83519	102481	12	YES	NaN
9	YES	Divorced	98152	155482	4	YES	NaN
10	NO	Single	29732	102602	19	YES	Risky
11	NO	Single	61063	94875	6	YES	NaN

In [11]:

```
del fraud_data['Taxable.Income']
```

In [12]:

```
del fraud_data['<= 30000']
```

In [13]:

```
fraud_data.head(10)
```

Out[13]:

	Undergrad	Marital.Status	City.Population	Work.Experience	Urban	> 30000
0	NO	Single	50047	10	YES	Good
1	YES	Divorced	134075	18	YES	Good
2	NO	Married	160205	30	YES	Good
3	YES	Single	193264	15	YES	Good
4	NO	Married	27533	28	NO	Good
5	NO	Divorced	116382	0	NO	Good
6	NO	Divorced	80890	8	YES	Good
7	YES	Single	131253	3	YES	Good
8	NO	Single	102481	12	YES	Good
9	YES	Divorced	155482	4	YES	Good

In [14]:

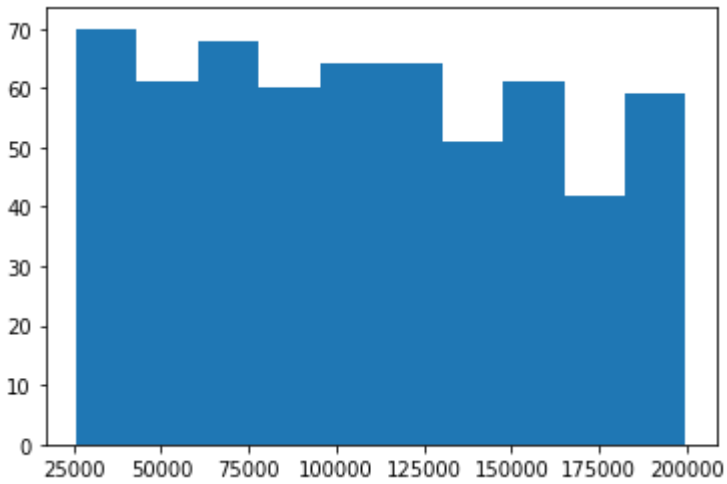
```
fraud_data.fillna(value=0,axis=0,inplace=True)
```

In [15]:

```
plt.hist(x="City.Population",data=fraud_data)
```

Out[15]:

```
(array([70., 61., 68., 60., 64., 64., 51., 61., 42., 59.]),
 array([ 25779. ,  43178.9,  60578.8,  77978.7,  95378.6, 112778.5,
        130178.4, 147578.3, 164978.2, 182378.1, 199778. ]),
 <a list of 10 Patch objects>)
```

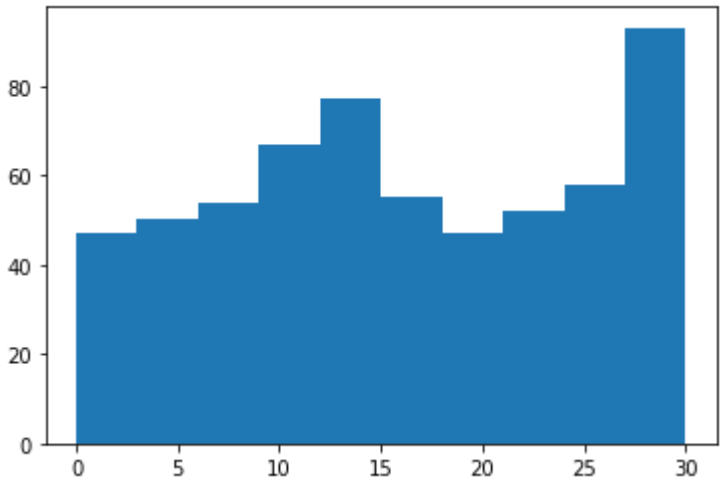


In [16]:

```
plt.hist(x="Work.Experience",data=fraud_data)
```

Out[16]:

```
(array([47., 50., 54., 67., 77., 55., 47., 52., 58., 93.]),
 array([ 0.,  3.,  6.,  9., 12., 15., 18., 21., 24., 27., 30.]),
 <a list of 10 Patch objects>)
```



In [17]:

```
fraud_data[['Undergrad','Urban']]=pd.get_dummies(fraud_data[['Undergrad','Urban']],drop_fir
```

In [19]:

```
fraud_data['Marital.Status']=pd.get_dummies(fraud_data['Marital.Status'],drop_first=True)
```

In [22]:

```
fraud_data.head(10)
```

Out[22]:

	Undergrad	Marital.Status	City.Population	Work.Experience	Urban	> 30000	Tax.Income
0	0	0	50047	10	1	Good	1
1	1	0	134075	18	1	Good	1
2	0	1	160205	30	1	Good	1
3	1	0	193264	15	1	Good	1
4	0	1	27533	28	0	Good	1
5	0	0	116382	0	0	Good	1
6	0	0	80890	8	1	Good	1
7	1	0	131253	3	1	Good	1
8	0	0	102481	12	1	Good	1
9	1	0	155482	4	1	Good	1

In [21]:

```
fraud_data['Tax.Income']=pd.get_dummies(fraud_data['> 30000'],drop_first=True)
```

In [24]:

```
scaled_data=StandardScaler()
x_scale=scaled_data.fit_transform(fraud_data.iloc[:, :-2])
```

Model Building

In [25]:

```
X=fraud_data.iloc[:, :-2]
y=fraud_data[['Tax.Income']]
```

In [26]:

```
X_train,X_test,y_train,y_test=train_test_split(x_scale,y,test_size=0.20,random_state=21)
```

In [27]:

```
X_train
```

Out[27]:

```
array([[ 0.96076892, -0.6912543 ,  0.25615362,  1.29507153, -1.00668904],
       [-1.040833  ,  1.4466456 ,  1.34458195,  0.61593715,  0.99335541],
       [-1.040833  ,  1.4466456 , -1.60360867,  0.38955903, -1.00668904],
       ...,
       [ 0.96076892, -0.6912543 ,  0.39512584, -0.17638629, -1.00668904],
       [-1.040833  ,  1.4466456 , -1.47098075, -0.40276442, -1.00668904],
       [ 0.96076892,  1.4466456 ,  0.58947011,  1.4082606 , -1.00668904]])
```

In [28]:

```
rf_model=RandomForestClassifier(n_estimators=120,max_depth=5,random_state=0)
rf_model.fit(X_train,y_train)
```

Out[28]:

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                       max_depth=5, max_features='auto', max_leaf_nodes=Non
e,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=120,
                       n_jobs=None, oob_score=False, random_state=0, verbose
=0,
                       warm_start=False)
```

In [29]:

```
y_pred_train=rf_model.predict(X_train)
y_pred_test=rf_model.predict(X_test)
```

Model Training and Testing Accuracy

In [31]:

```
print(classification_report(y_test,y_pred_test))
print(confusion_matrix(y_test,y_pred_test))
print(accuracy_score(y_test,y_pred_test))
```

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        21
           1       0.82      1.00      0.90        99

    accuracy                           0.82       120
   macro avg       0.41      0.50      0.45       120
weighted avg       0.68      0.82      0.75       120

[[ 0 21]
 [ 0 99]]
0.825
```

In [35]:

```
#True postiverate is rate is decreased,it is not going to a good model
```

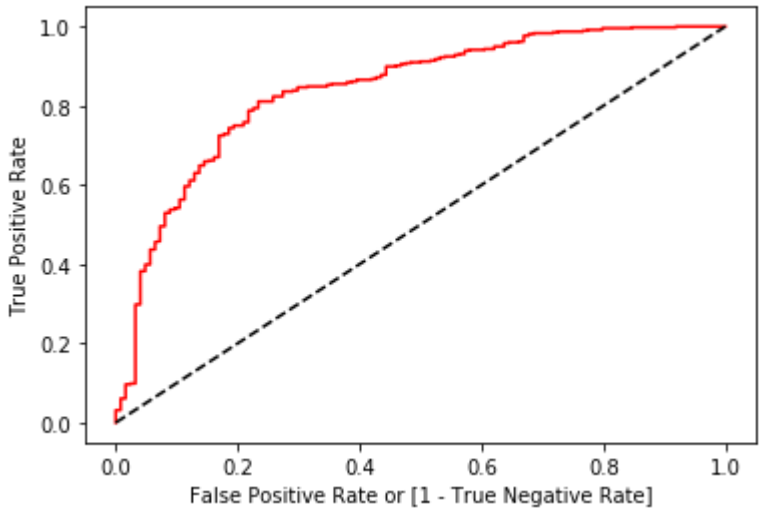
In [34]:

```
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score

fpr, tpr, thresholds = roc_curve(y, rf_model.predict_proba (x_scale)[:,1])

auc = roc_auc_score(y_test, y_pred_test)
print(auc)

import matplotlib.pyplot as plt
plt.plot(fpr, tpr, color='red', label='rf_model ( area  = %0.2f)'%auc)
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate or [1 - True Negative Rate]')
plt.ylabel('True Positive Rate')
```

```
0.5
```

Out[34]:

```
Text(0, 0.5, 'True Positive Rate')
```



In [50]:

```
#Hence,True positive value is getting 0, it means fraudelent is getting low
```