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

```
df  = pd.read_csv("salary.csv")
```

```
df.head()
```

amed: 0	age	Workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours per wee
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	4
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	1
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	4
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	4
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	4

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 1032 entries, 0 to 1031
Data columns (total 16 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Unnamed: 0      1032 non-null   int64
 1   age             1032 non-null   int64
 2   Workclass       1032 non-null   object
 3   fnlwgt          1032 non-null   int64
 4   education       1032 non-null   object
 5   education-num   1032 non-null   int64
 6   marital-status  1032 non-null   object
 7   occupation      1032 non-null   object
 8   relationship    1032 non-null   object
 9   race            1032 non-null   object
 10  sex             1032 non-null   object
 11  capital-gain    1032 non-null   int64
 12  capital-loss    1032 non-null   int64
 13  hours-per-week  1032 non-null   int64
 14  native-country  1032 non-null   object
 15  Income          1032 non-null   int64
dtypes: int64(8), object(8)
memory usage: 129.1+ KB
```

```
#occupation,native-country hours-per-week capital-loss capital-gain education-num education Workclass age
#object = Workclass education occupation native-country
df['occupation'].unique()
```

```
array([' Adm-clerical', ' Exec-managerial', ' Handlers-cleaners',
       ' Prof-specialty', ' Other-service', ' Sales', ' Transport-moving',
       ' Farming-fishing', ' Machine-op-inspct', ' Tech-support',
       ' Craft-repair', ' Protective-serv', ' Armed-Forces',
       ' Priv-house-serv'], dtype=object)
```

```
df['Workclass'].unique()
```

```
array([' State-gov', ' Self-emp-not-inc', ' Private', ' Federal-gov',
       ' Local-gov', ' Self-emp-inc'], dtype=object)
```

```
df['education'].unique()
```

```
array([' Bachelors', ' HS-grad', ' 11th', ' Masters', ' 9th',
       ' Some-college', ' Assoc-acdm', ' 7th-8th', ' Doctorate',
       ' Assoc-voc', ' Prof-school', ' 5th-6th', ' 10th', ' Preschool',
       ' 12th', ' 1st-4th'], dtype=object)
```

```
df['native-country'].unique()
```

```
array([' United-States', ' Cuba', ' Jamaica', ' India', ' Mexico',
       ' Puerto-Rico', ' Honduras', ' England', ' Canada', ' Germany',
       ' Iran', ' Philippines', ' Poland', ' Columbia', ' Cambodia',
       ' Thailand', ' Ecuador', ' Laos', ' Taiwan', ' Haiti', ' Portugal',
       ' Dominican-Republic', ' El-Salvador', ' France', ' Guatemala',
       ' Italy', ' China', ' South', ' Japan', ' Yugoslavia'],
      dtype=object)
```

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['native-country'] = le.fit_transform(df['native-country'])
df['native-country'] = df['native-country'].astype(float)
```

```
df['native-country']
```

```
0       28.0
1       28.0
2       28.0
3       28.0
4        4.0
        ...
1027    28.0
1028    28.0
1029    28.0
1030    28.0
1031    18.0
Name: native-country, Length: 1032, dtype: float64
```

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['education'] = le.fit_transform(df['education'])
df['education'] = df['education'].astype(float)
```

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['Workclass'] = le.fit_transform(df['Workclass'])
df['Workclass'] = df['Workclass'].astype(float)
```

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['occupation'] = le.fit_transform(df['occupation'])
df['occupation'] = df['occupation'].astype(float)
```

```
df.isnull().count()
```

```
Unnamed: 0        1032
age               1032
Workclass         1032
fnlwgt            1032
education         1032
education-num     1032
marital-status    1032
occupation        1032
relationship      1032
race              1032
sex               1032
capital-gain      1032
capital-loss      1032
hours-per-week    1032
native-country    1032
Income            1032
dtype: int64
```

```
from sklearn.model_selection import train_test_split
x = df[['occupation','native-country', 'hours-per-week','capital-loss', 'capital-gain', 'education-num', 'education', 'Workclass', 'a
```

```
y = df["Income"]
```

```
x.head()
```

	occupation	native-country	hours-per-week	capital-loss	capital-gain	education-num	education	Workclass	age
0	0.0	28.0	40	0	2174	13	9.0	5.0	39
1	3.0	28.0	13	0	0	13	9.0	4.0	50
2	5.0	28.0	40	0	0	9	11.0	2.0	38
3	5.0	28.0	40	0	0	7	1.0	2.0	53

```
y.head()
```

```
0    0
1    0
2    0
3    0
4    0
Name: Income, dtype: int64
```

```
x_train, x_test,y_train,y_test = train_test_split(x , y , test_size =0.3)
```

```
from sklearn.linear_model import LogisticRegression
import warnings
warnings.filterwarnings('ignore')
```

```
log = LogisticRegression()
log.fit(x_train, y_train)
```

```
y_hat = log.predict(x_test)
```

```
from sklearn.metrics import confusion_matrix
```

```
confusion_matrix(y_test,y_hat)
```

```
array([[224,  21],
       [ 43,  22]])
```

```
from sklearn.metrics import accuracy_score,precision_score,recall_score , f1_score
```

```
accuracy_score(y_test, y_hat)
```

```
0.7935483870967742
```

```
precision_score(y_test,y_hat)
```

```
0.5116279069767442
```

```
recall_score(y_test,y_hat)
```

```
0.3384615384615385
```

```
f1_score(y_test,y_hat)
```

```
0.40740740740740744
```

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

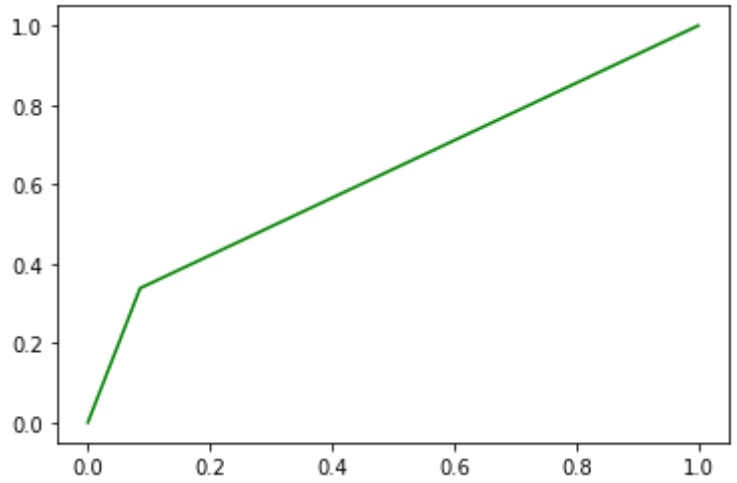
```
roc_curve(y_test,y_hat)
```

```
(array([0.        , 0.08571429, 1.        ]),
 array([0.        , 0.33846154, 1.        ]),
```

```
      array([2, 1, 0]))
```

```
fpr,tpr,thres = roc_curve(y_test,y_hat)
```

```
plt.plot(fpr,tpr,"g-",label = "current")
plt.show()
```




```
roc_auc_score(y_test,y_hat)
```

```
     0.6263736263736265
```