In [1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics
from sklearn.feature_selection import SelectFromModel
from sklearn import svm

data=pd.read_csv("train.csv")
data.head()
```

Out[1]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loa
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	

In [2]:
```python
data.shape
```

Out[2]:
```
(614, 13)
```

In [3]:
```python
data.isnull().sum()
```

Out[3]:
```
Loan_ID               0
Gender               13
Married               3
Dependents           15
Education             0
Self_Employed        32
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           22
Loan_Amount_Term     14
Credit_History       50
Property_Area         0
Loan_Status           0
dtype: int64
```

In [4]:
```python
data['Gender'].value_counts()
```

Out[4]:
```
Male      489
Female    112
Name: Gender, dtype: int64
```

In [5]:
```python
data['Married'].value_counts()
```

Out[5]:
```
Yes    398
No     213
Name: Married, dtype: int64
```

In [6]:
```python
data['Dependents'].value_counts()
```

Out[6]:
```
0     345
1     102
2     101
3+     51
Name: Dependents, dtype: int64
```

In [7]:
```python
data['Self_Employed'].value_counts()
```

Out[7]:
```
No     500
Yes     82
Name: Self_Employed, dtype: int64
```

In [8]:
```python
data['Credit_History'].value_counts()
```

Out[8]:
```
1.0    475
0.0     89
Name: Credit_History, dtype: int64
```

In [9]:
```python
data['Property_Area'].value_counts()
```

Out[9]:
```
Semiurban    233
Urban        202
Rural        179
Name: Property_Area, dtype: int64
```

In [10]:
```python
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 13 columns):
Loan_ID              614 non-null object
Gender               601 non-null object
Married              611 non-null object
Dependents           599 non-null object
Education            614 non-null object
Self_Employed        582 non-null object
ApplicantIncome      614 non-null int64
CoapplicantIncome    614 non-null float64
LoanAmount           592 non-null float64
Loan_Amount_Term     600 non-null float64
Credit_History       564 non-null float64
Property_Area        614 non-null object
Loan_Status          614 non-null object
dtypes: float64(4), int64(1), object(8)
memory usage: 62.4+ KB
```

In [11]:
```python
data.describe()
```

Out[11]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
count	614.000000	614.000000	592.000000	600.00000	564.000000
mean	5403.459283	1621.245798	146.412162	342.00000	0.842199
std	6109.041673	2926.248369	85.587325	65.12041	0.364878
min	150.000000	0.000000	9.000000	12.00000	0.000000
25%	2877.500000	0.000000	100.000000	360.00000	1.000000
50%	3812.500000	1188.500000	128.000000	360.00000	1.000000
75%	5795.000000	2297.250000	168.000000	360.00000	1.000000
max	81000.000000	41667.000000	700.000000	480.00000	1.000000

In [12]:
```python
data['Gender'].fillna(data['Gender'].mode()[0],inplace=True)
data['Married'].fillna(data['Married'].mode()[0],inplace=True)
data['Dependents'].fillna(data['Dependents'].mode()[0],inplace=True)
data['Self_Employed'].fillna(data['Self_Employed'].mode()[0],inplace=True)
data['LoanAmount'].fillna(data['LoanAmount'].median(),inplace=True)
data['Loan_Amount_Term'].fillna(data['Loan_Amount_Term'].median(),inplace=True)
data['Credit_History'].fillna(data['Credit_History'].mode()[0],inplace=True)
```

In [13]:
```python
data.isnull().sum()
```

Out[13]:
```
Loan_ID              0
Gender               0
Married              0
Dependents           0
Education            0
Self_Employed        0
ApplicantIncome      0
CoapplicantIncome    0
LoanAmount           0
Loan_Amount_Term     0
Credit_History       0
Property_Area        0
Loan_Status          0
dtype: int64
```

In [14]:
```python
data['Loan_Status'].value_counts()
```

Out[14]:
```
Y    422
N    192
Name: Loan_Status, dtype: int64
```

In [15]:
```python
data['Combined_Income']=data['ApplicantIncome']+data['CoapplicantIncome']
data.drop(['ApplicantIncome','CoapplicantIncome'],axis=1,inplace=True)
```

In [16]:
```python
data['Income_Loan_Ratio']=(data['Loan_Amount_Term']*data['Combined_Income'])/data['LoanAmoun
t']
mean=np.mean(data['Income_Loan_Ratio'])
std=np.std(data['Income_Loan_Ratio'])
mean=float(mean)
std=float(std)
data['Income_Loan_Ratio']=(data['Income_Loan_Ratio']-mean)/std
data=data[data['Income_Loan_Ratio']<=2]
```

In [17]:
```python
data
```

Out[17]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	LoanAmount	Loan_Amount_Term	Credit_History	Pr
0	LP001002	Male	No	0	Graduate	No	128.0	360.0	1.0	
1	LP001003	Male	Yes	1	Graduate	No	128.0	360.0	1.0	
2	LP001005	Male	Yes	0	Graduate	Yes	66.0	360.0	1.0	
3	LP001006	Male	Yes	0	Not Graduate	No	120.0	360.0	1.0	
4	LP001008	Male	No	0	Graduate	No	141.0	360.0	1.0	
5	LP001011	Male	Yes	2	Graduate	Yes	267.0	360.0	1.0	
6	LP001013	Male	Yes	0	Not Graduate	No	95.0	360.0	1.0	
7	LP001014	Male	Yes	3+	Graduate	No	158.0	360.0	0.0	
8	LP001018	Male	Yes	2	Graduate	No	168.0	360.0	1.0	
9	LP001020	Male	Yes	1	Graduate	No	349.0	360.0	1.0	
10	LP001024	Male	Yes	2	Graduate	No	70.0	360.0	1.0	
11	LP001027	Male	Yes	2	Graduate	No	109.0	360.0	1.0	
12	LP001028	Male	Yes	2	Graduate	No	200.0	360.0	1.0	
13	LP001029	Male	No	0	Graduate	No	114.0	360.0	1.0	
14	LP001030	Male	Yes	2	Graduate	No	17.0	120.0	1.0	
15	LP001032	Male	No	0	Graduate	No	125.0	360.0	1.0	
16	LP001034	Male	No	1	Not Graduate	No	100.0	240.0	1.0	
17	LP001036	Female	No	0	Graduate	No	76.0	360.0	0.0	
18	LP001038	Male	Yes	0	Not Graduate	No	133.0	360.0	1.0	
19	LP001041	Male	Yes	0	Graduate	No	115.0	360.0	1.0	
20	LP001043	Male	Yes	0	Not Graduate	No	104.0	360.0	0.0	
21	LP001046	Male	Yes	1	Graduate	No	315.0	360.0	1.0	
22	LP001047	Male	Yes	0	Not Graduate	No	116.0	360.0	0.0	
23	LP001050	Male	Yes	2	Not Graduate	No	112.0	360.0	0.0	
24	LP001052	Male	Yes	1	Graduate	No	151.0	360.0	1.0	
25	LP001066	Male	Yes	0	Graduate	Yes	191.0	360.0	1.0	
26	LP001068	Male	Yes	0	Graduate	No	122.0	360.0	1.0	
27	LP001073	Male	Yes	2	Not Graduate	No	110.0	360.0	1.0	
28	LP001086	Male	No	0	Not Graduate	No	35.0	360.0	1.0	
29	LP001087	Female	No	2	Graduate	No	120.0	360.0	1.0	
...	...	...	...	...	...	...	...	...	...	
584	LP002911	Male	Yes	1	Graduate	No	146.0	360.0	0.0	
585	LP002912	Male	Yes	1	Graduate	No	172.0	84.0	1.0	
586	LP002916	Male	Yes	0	Graduate	No	104.0	360.0	1.0	
587	LP002917	Female	No	0	Not Graduate	No	70.0	360.0	1.0	
588	LP002925	Male	No	0	Graduate	No	94.0	360.0	1.0	
589	LP002926	Male	Yes	2	Graduate	Yes	106.0	360.0	0.0	
590	LP002928	Male	Yes	0	Graduate	No	56.0	180.0	1.0	
591	LP002931	Male	Yes	2	Graduate	Yes	205.0	240.0	1.0	
592	LP002933	Male	No	3+	Graduate	Yes	292.0	360.0	1.0	
593	LP002936	Male	Yes	0	Graduate	No	142.0	180.0	1.0	
594	LP002938	Male	Yes	0	Graduate	Yes	260.0	360.0	1.0	
595	LP002940	Male	No	0	Not Graduate	No	110.0	360.0	1.0	
596	LP002941	Male	Yes	2	Not Graduate	Yes	187.0	360.0	1.0	
597	LP002943	Male	No	0	Graduate	No	88.0	360.0	0.0	
598	LP002945	Male	Yes	0	Graduate	Yes	180.0	360.0	1.0	
599	LP002948	Male	Yes	2	Graduate	No	192.0	360.0	1.0	
600	LP002949	Female	No	3+	Graduate	No	350.0	180.0	1.0	
601	LP002950	Male	Yes	0	Not Graduate	No	155.0	360.0	1.0	
602	LP002953	Male	Yes	3+	Graduate	No	128.0	360.0	1.0	
603	LP002958	Male	No	0	Graduate	No	172.0	360.0	1.0	
604	LP002959	Female	Yes	1	Graduate	No	496.0	360.0	1.0	
605	LP002960	Male	Yes	0	Not Graduate	No	128.0	180.0	1.0	
606	LP002961	Male	Yes	1	Graduate	No	173.0	360.0	1.0	
607	LP002964	Male	Yes	2	Not Graduate	No	157.0	360.0	1.0	
608	LP002974	Male	Yes	0	Graduate	No	108.0	360.0	1.0	
609	LP002978	Female	No	0	Graduate	No	71.0	360.0	1.0	
610	LP002979	Male	Yes	3+	Graduate	No	40.0	180.0	1.0	
611	LP002983	Male	Yes	1	Graduate	No	253.0	360.0	1.0	
612	LP002984	Male	Yes	2	Graduate	No	187.0	360.0	1.0	
613	LP002990	Female	No	0	Graduate	Yes	133.0	360.0	0.0	

595 rows × 13 columns

In [18]:
```python
from sklearn.model_selection import train_test_split
train,test=train_test_split(data,test_size=0.2,random_state=4)
```

In [19]:
```python
train.shape
```

Out[19]:
```
(476, 13)
```

In [20]:
```python
train['Loan_Status'].value_counts()
```

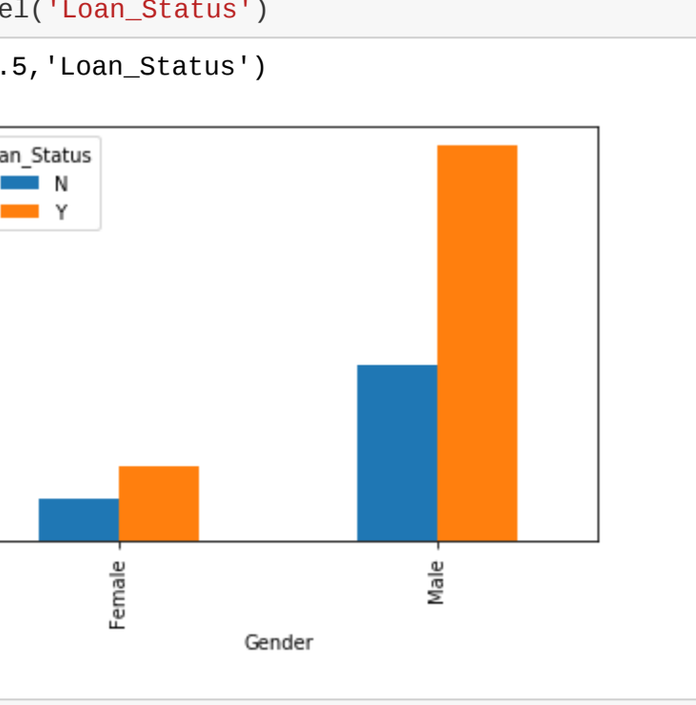
Out[20]:
```
Y    325
N    151
Name: Loan_Status, dtype: int64
```

In [21]:
```python
test.shape
```

Out[21]:
```
(119, 13)
```

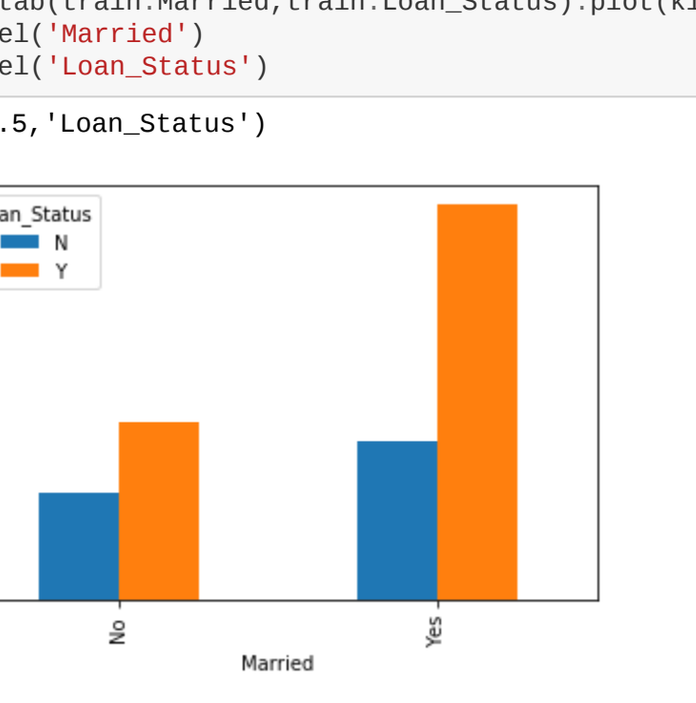
In [22]:
```python
pd.crosstab(train.Gender,train.Loan_Status).plot(kind='bar')
plt.xlabel('Gender')
plt.ylabel('Loan_Status')
```

Out[22]:
```
Text(0,0.5,'Loan_Status')
```



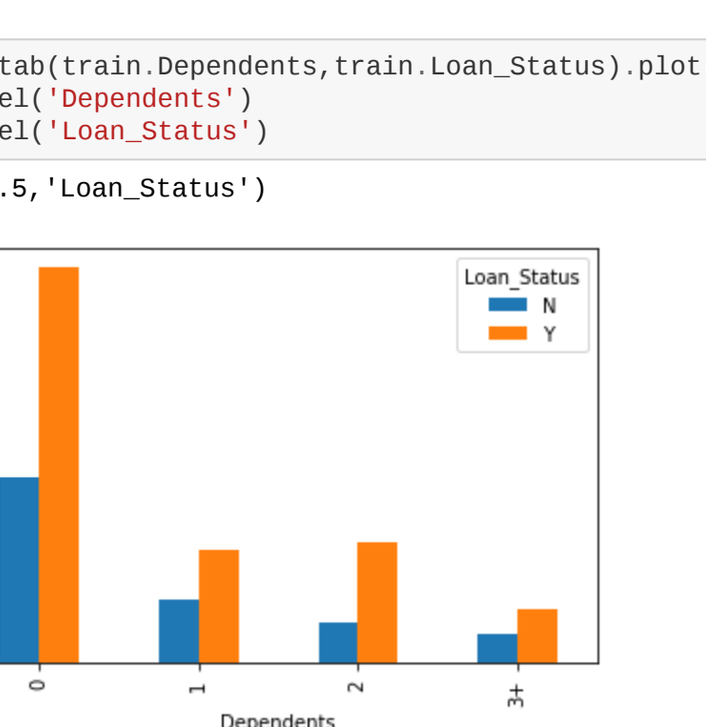
In [23]:
```python
pd.crosstab(train.Married,train.Loan_Status).plot(kind='bar')
plt.xlabel('Married')
plt.ylabel('Loan_Status')
```

Out[23]:
```
Text(0,0.5,'Loan_Status')
```



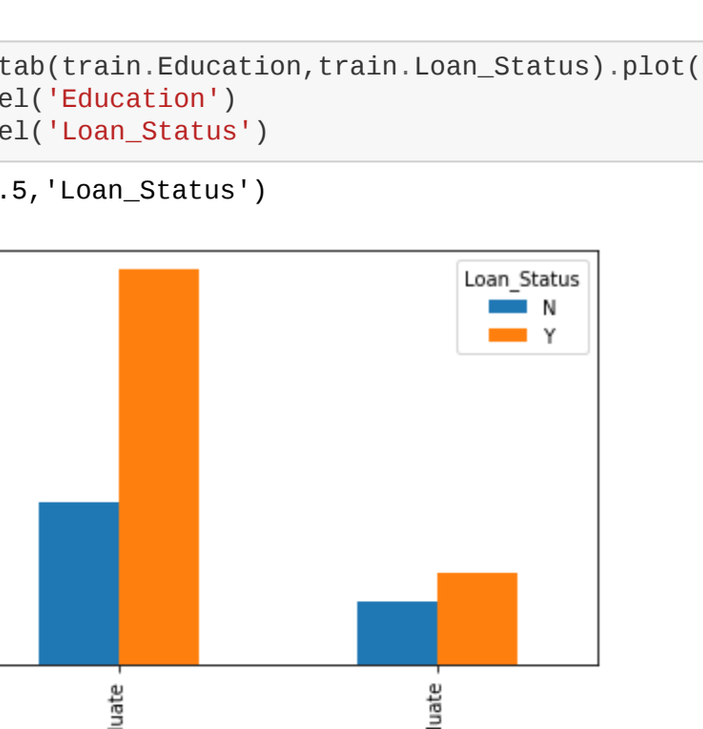
In [24]:
```python
pd.crosstab(train.Dependents,train.Loan_Status).plot(kind='bar')
plt.xlabel('Dependents')
plt.ylabel('Loan_Status')
```

Out[24]:
```
Text(0,0.5,'Loan_Status')
```



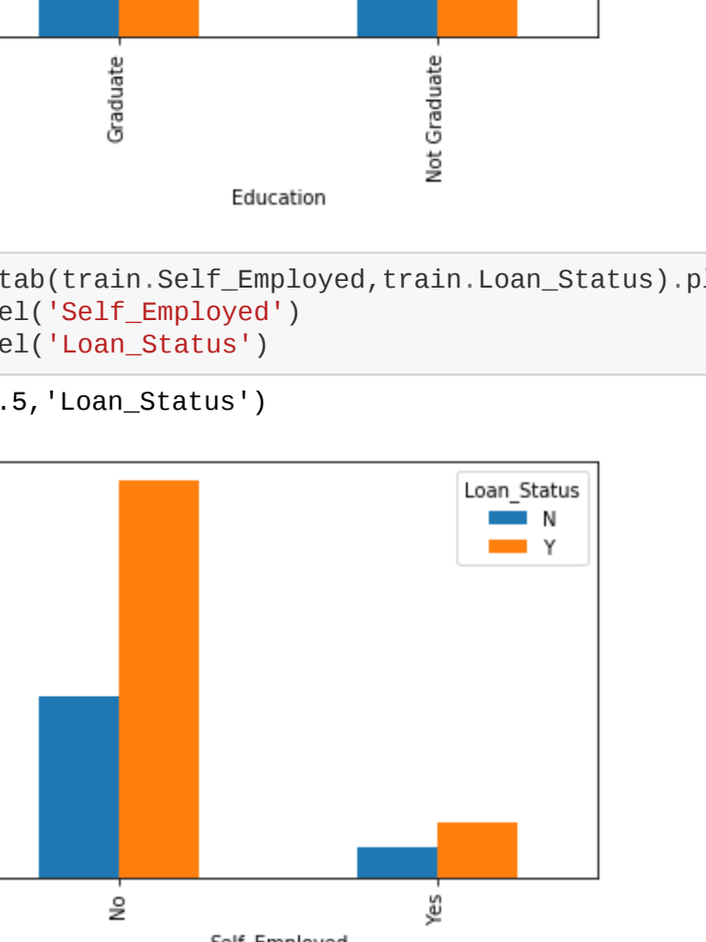
In [25]:
```python
pd.crosstab(train.Education,train.Loan_Status).plot(kind='bar')
plt.xlabel('Education')
plt.ylabel('Loan_Status')
```

Out[25]:
```
Text(0,0.5,'Loan_Status')
```



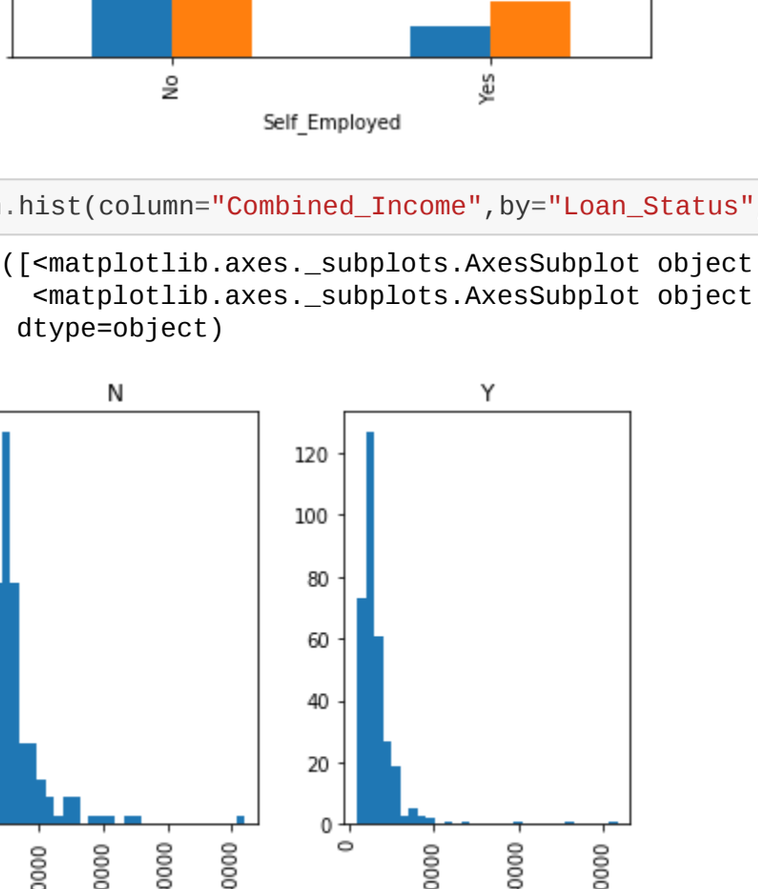
In [26]:
```python
pd.crosstab(train.Self_Employed,train.Loan_Status).plot(kind='bar')
plt.xlabel('Self_Employed')
plt.ylabel('Loan_Status')
```

Out[26]:
```
Text(0,0.5,'Loan_Status')
```



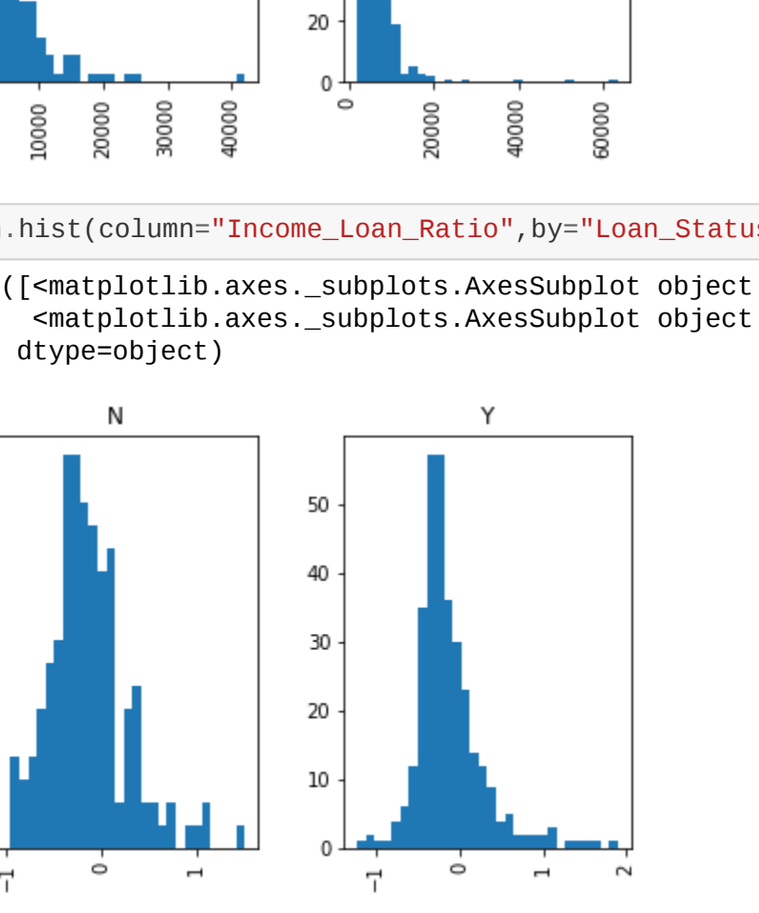
In [27]:
```python
train.hist(column="Combined_Income",by="Loan_Status",bins=30)
```

Out[27]:
```
array([<matplotlib.axes._subplots.AxesSubplot object at 0x00000137F1032630>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x00000137F1081CF8>],
      dtype=object)
```



In [28]:
```python
train.hist(column="Income_Loan_Ratio",by="Loan_Status",bins=30)
```

Out[28]:
```
array([<matplotlib.axes._subplots.AxesSubplot object at 0x00000137F11200F0>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x00000137F11BC390>],
      dtype=object)
```



In [29]:
```python
cols = list(train.columns.values)
cols
```

Out[29]:
```
['Loan_ID',
 'Gender',
 'Married',
 'Dependents',
 'Education',
 'Self_Employed',
 'LoanAmount',
 'Loan_Amount_Term',
 'Credit_History',
 'Property_Area',
 'Loan_Status',
 'Combined_Income',
 'Income_Loan_Ratio']
```

In [30]:
```python
train = train[['Loan_ID', 'Gender', 'Married','Dependents','Education','Self_Employed','Loan
Amount','Loan_Amount_Term','Credit_History','Property_Area','Combined_Income','Income_Loan_R
atio','Loan_Status']]
test = test[['Loan_ID', 'Gender', 'Married','Dependents','Education','Self_Employed','LoanAm
ount','Loan_Amount_Term','Credit_History','Property_Area','Combined_Income','Income_Loan_Rat
io','Loan_Status']]
```

In [31]:
```python
train.head()
```

Out[31]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	LoanAmount	Loan_Amount_Term	Credit_History	Pr
161	LP001562	Male	Yes	0	Graduate	No	275.0	360.0	1.0	
14	LP001030	Male	Yes	2	Graduate	No	17.0	120.0	1.0	
216	LP001722	Male	Yes	0	Graduate	No	135.0	360.0	1.0	
201	LP001677	Male	No	2	Graduate	No	166.0	360.0	0.0	
532	LP002723	Male	No	2	Graduate	No	110.0	360.0	0.0	

In [32]:
```python
train_copy=train.copy()
train_copy['Gender'] = train_copy['Gender'].map({'Male':1,'Female':0})
train_copy['Married'] = train_copy['Married'].map({'Yes':1,'No':0})
train_copy['Dependents'] = train_copy['Dependents'].map({'3+':3,'0':0, '1':1, '2':2})
train_copy['Education'] = train_copy['Education'].map({'Graduate':1,'Not Graduate':0})
train_copy['Self_Employed'] = train_copy['Self_Employed'].map({'Yes':1,'No':0})
train_copy['Property_Area'] = train_copy['Property_Area'].map({'Rural':0,'Semiurban':1,'Urba
n':2})
train_copy['Loan_Status'] = train_copy['Loan_Status'].map({'Y':1,'N':0})
train_copy.head()
```

Out[32]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	LoanAmount	Loan_Amount_Term	Credit_History	Pr
161	LP001562	1	1	0	1	0	275.0	360.0	1.0	
14	LP001030	1	1	2	1	0	17.0	120.0	1.0	
216	LP001722	1	1	0	1	0	135.0	360.0	1.0	
201	LP001677	1	0	2	1	0	166.0	360.0	0.0	
532	LP002723	1	0	2	1	0	110.0	360.0	0.0	

In [33]:
```python
test_copy=test.copy()
test_copy['Gender'] = test_copy['Gender'].map({'Male':1,'Female':0})
test_copy['Married'] = test_copy['Married'].map({'Yes':1,'No':0})
test_copy['Dependents'] = test_copy['Dependents'].map({'3+':3,'0':0, '1':1, '2':2})
test_copy['Education'] = test_copy['Education'].map({'Graduate':1,'Not Graduate':0})
test_copy['Self_Employed'] = test_copy['Self_Employed'].map({'Yes':1,'No':0})
test_copy['Property_Area'] = test_copy['Property_Area'].map({'Rural':0,'Semiurban':1,'Urban'
:2})
test_copy['Loan_Status'] = test_copy['Loan_Status'].map({'Y':1,'N':0})
test_copy.head()
```

Out[33]:

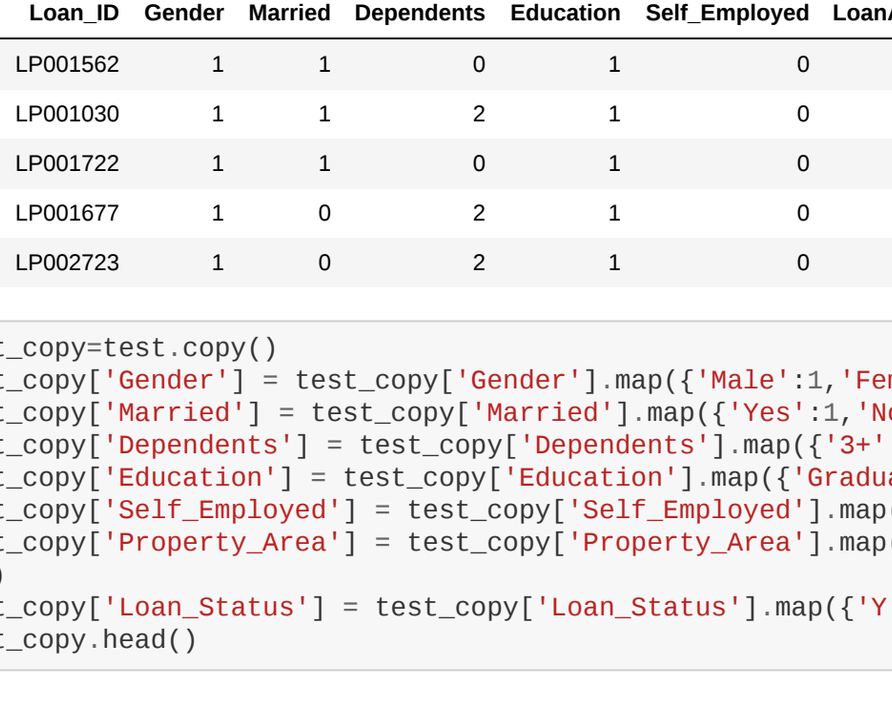
	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	LoanAmount	Loan_Amount_Term	Credit_History	Pr
233	LP001776	0	0	0	1	0	280.0	360.0	1.0	
159	LP001552	1	1	0	1	0	255.0	360.0	1.0	
559	LP002804	0	1	0	1	0	182.0	360.0	1.0	
557	LP002795	1	1	3	1	1	260.0	360.0	1.0	
165	LP001574	1	1	0	1	0	182.0	360.0	1.0	

In [34]:
```python
clf = DecisionTreeClassifier()
clf = clf.fit(train_copy.iloc[:,1:12],train_copy.iloc[:,-1])
```

In [35]:
```python
labels=['Gender','Married','Dependents','Education','Self_Employed','LoanAmount','Loan_Amoun
t_Term','Credit_History','Property_Area','Combined_Income','Income_Loan_Ratio'];
for feature in zip(labels,clf.feature_importances_):
    print(feature)
```

```
('Gender', 0.010738665308201733)
('Married', 0.004849719816607234)
('Dependents', 0.03288559564056252)
('Education', 0.01907556461198845)
('Self_Employed', 0.015138586935224739)
('LoanAmount', 0.15825332996005442)
('Loan_Amount_Term', 0.017646550757809184)
('Credit_History', 0.2975073339540112)
('Property_Area', 0.035992528442859564)
('Combined_Income', 0.21773790360542972)
('Income_Loan_Ratio', 0.19017422096725123)
```

In [36]:
```python
features = labels
importances = clf.feature_importances_
indices = np.argsort(importances)
plt.title('Feature Importances')
plt.barh(range(len(indices)), importances[indices], color='b', align='center')
plt.yticks(range(len(indices)), [features[i] for i in indices])
plt.xlabel('Relative Importance')
plt.show()
```



In [37]:
```python
important_train=train_copy[['Credit_History','Income_Loan_Ratio','Dependents','Property_Are
a']]
important_test=test_copy[['Credit_History','Income_Loan_Ratio','Dependents','Property_Area'
]]
new_clf = DecisionTreeClassifier()
new_clf.fit(important_train, train_copy.iloc[:,-1])
y_pred = new_clf.predict(important_test)
score=metrics.accuracy_score(test_copy.iloc[:,-1], y_pred)
score
```

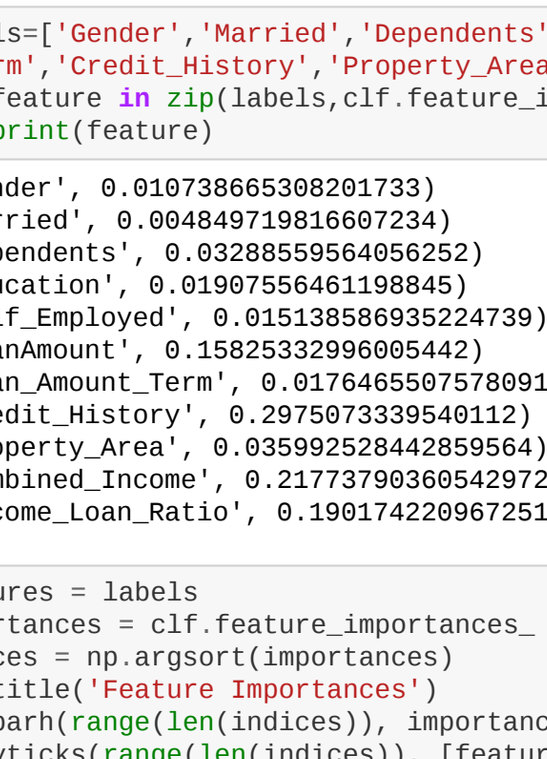
Out[37]:
```
0.6722689075630253
```

In [38]:
```python
results=metrics.confusion_matrix(test_copy.iloc[:,-1],y_pred)
results
```

Out[38]:
```
array([[15, 20],
       [19, 65]], dtype=int64)
```

In [39]:
```python
plt.matshow(results)
plt.colorbar()
```

Out[39]:
```
<matplotlib.colorbar.Colorbar at 0x137f12b50f0>
```

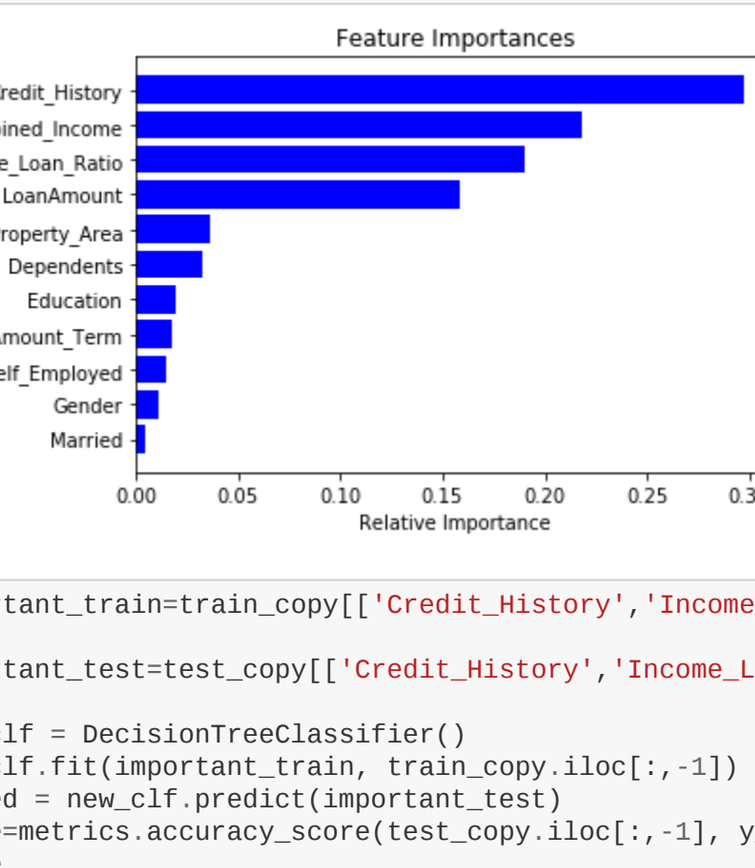


In [40]:
```python
report=metrics.precision_score(test_copy.iloc[:,-1],y_pred)
print(report)
```

```
0.7647058823529411
```

In [41]:
```python
y_true=test_copy.iloc[:,-1]
```

In [42]:
```python
fpr, tpr, threshold = metrics.roc_curve(y_true, y_pred)
auc = metrics.roc_auc_score(y_true, y_pred)
plt.plot(fpr,tpr,label="data , auc="+str(auc))
plt.legend(loc=4)
plt.show()
```

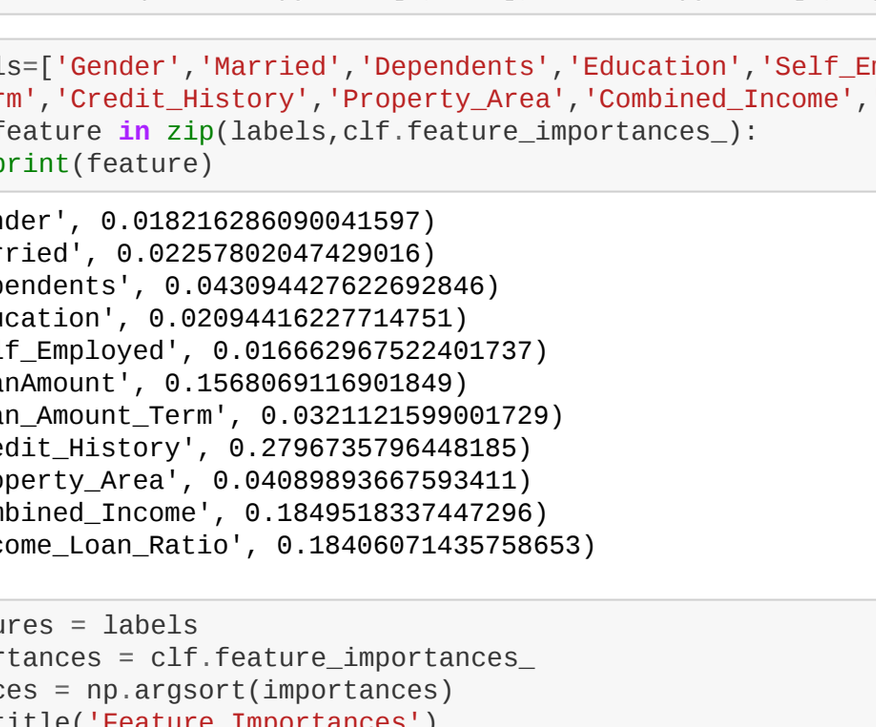


In [43]:
```python
clf = RandomForestClassifier(n_estimators=300, random_state=5, min_samples_split=4, class_we
ight={0:0.6, 1:0.4})
clf = clf.fit(train_copy.iloc[:,1:12], train_copy.iloc[:,-1])
```

In [44]:
```python
labels=['Gender','Married','Dependents','Education','Self_Employed','LoanAmount','Loan_Amoun
t_Term','Credit_History','Property_Area','Combined_Income','Income_Loan_Ratio'];
for feature in zip(labels,clf.feature_importances_):
    print(feature)
```

```
('Gender', 0.018216286090041597)
('Married', 0.02257802047429016)
('Dependents', 0.043094427622692846)
('Education', 0.02094416227714751)
('Self_Employed', 0.016662967522401737)
('LoanAmount', 0.1568069116901849)
('Loan_Amount_Term', 0.0321121559001729)
('Credit_History', 0.2796735796448185)
('Property_Area', 0.04089893667593411)
('Combined_Income', 0.1849518337447296)
('Income_Loan_Ratio', 0.18406071435758653)
```

In [45]:
```python
features = labels
importances = clf.feature_importances_
indices = np.argsort(importances)
plt.title('Feature Importances')
plt.barh(range(len(indices)), importances[indices], color='b', align='center')
plt.yticks(range(len(indices)), [features[i] for i in indices])
plt.xlabel('Relative Importance')
plt.show()
```



In [46]:
```python
important_train = train_copy[['Credit_History','Income_Loan_Ratio','Dependents','Property_Ar
ea']]
important_test = test_copy[['Credit_History','Income_Loan_Ratio','Dependents','Property_Are
a']]
```

In [47]:
```python
new_clf = RandomForestClassifier(n_estimators=300, random_state=5, min_samples_split=4, clas
s_weight={0:0.6, 1:0.4})
new_clf.fit(important_train, train_copy.iloc[:,-1])
y_pred = new_clf.predict(important_test)
score=metrics.accuracy_score(test_copy.iloc[:,-1], y_pred)
score
```

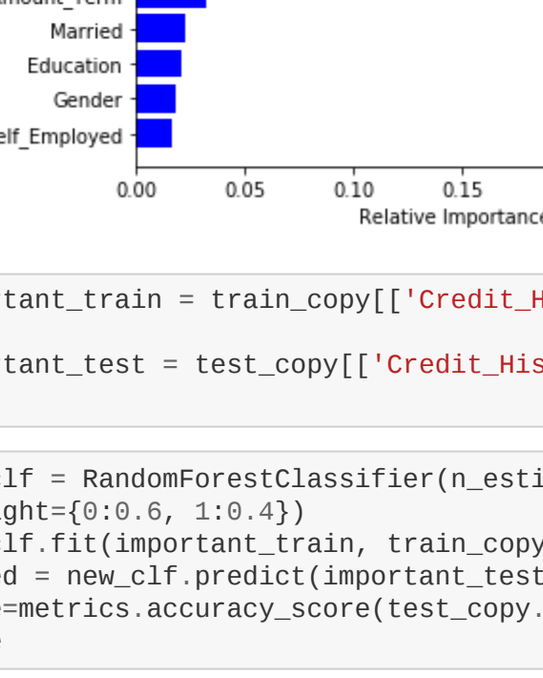
Out[47]:
```
0.7563025210084033
```

In [48]:
```python
results=metrics.confusion_matrix(test_copy.iloc[:,-1],y_pred)
results
```

Out[48]:
```
array([[19, 16],
       [13, 71]], dtype=int64)
```

In [49]:
```python
plt.matshow(results)
plt.colorbar()
```

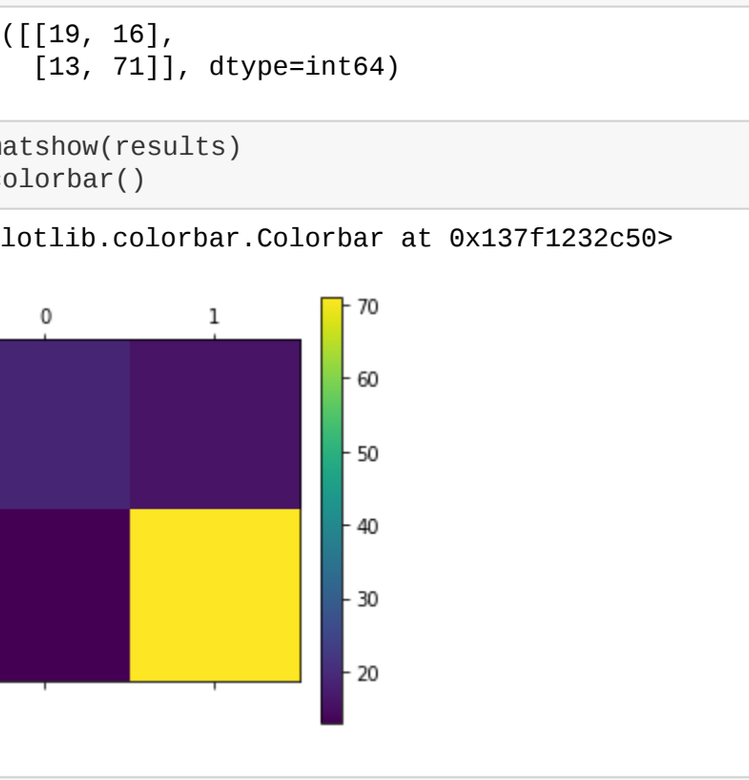
Out[49]:
```
<matplotlib.colorbar.Colorbar at 0x137f1232c50>
```



In [50]:
```python
report=metrics.precision_score(test_copy.iloc[:,-1],y_pred)
print(report)
```

```
0.8160919540229885
```

In [51]:
```python
y_true = test_copy.iloc[:,-1]
fpr, tpr, threshold = metrics.roc_curve(y_true, y_pred)
auc = metrics.roc_auc_score(y_true, y_pred)
plt.plot(fpr,tpr,label="data , auc="+str(auc))
plt.legend(loc=4)
plt.show()
```



In [52]:
```python
clf = svm.SVC(kernel='linear')
clf.fit(train_copy.iloc[:,1:12],train_copy.iloc[:,-1])
y_pred = clf.predict(test_copy.iloc[:,1:12])
```

In [53]:
```python
score=metrics.accuracy_score(test_copy.iloc[:,-1], y_pred)
score
```

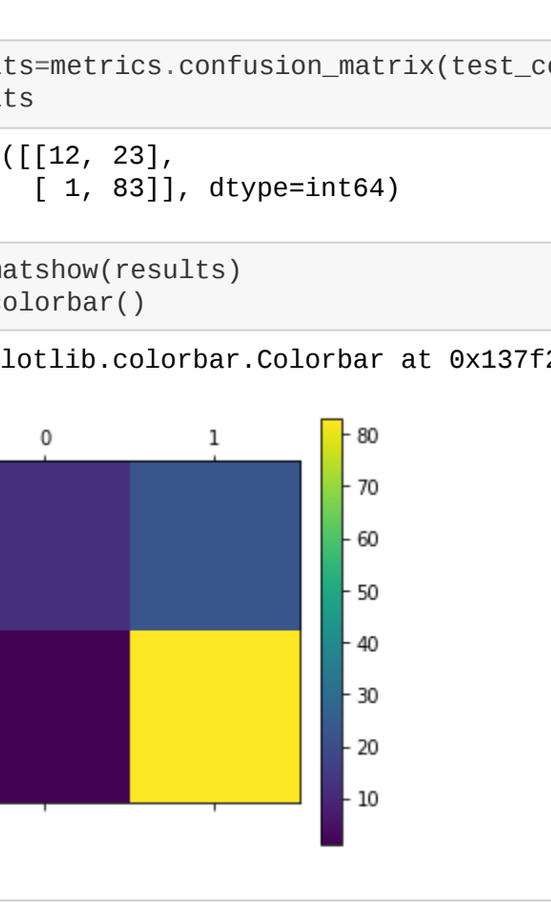
Out[53]:
```
0.7983193277310925
```

In [54]:
```python
results=metrics.confusion_matrix(test_copy.iloc[:,-1],y_pred)
results
```

Out[54]:
```
array([[12, 23],
       [ 1, 83]], dtype=int64)
```

In [55]:
```python
plt.matshow(results)
plt.colorbar()
```

Out[55]:
```
<matplotlib.colorbar.Colorbar at 0x137f2546a20>
```



In [56]:
```python
report=metrics.precision_score(test_copy.iloc[:,-1],y_pred)
print(report)
```

```
0.7830188679245284
```

In [57]:
```python
y_true = test_copy.iloc[:,-1]
fpr, tpr, threshold = metrics.roc_curve(y_true, y_pred)
auc = metrics.roc_auc_score(y_true, y_pred)
plt.plot(fpr,tpr,label="data , auc="+str(auc))
plt.legend(loc=4)
plt.show()
```