In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as npm
import pylab as pl
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error,classifi
```

In [ ]:
```
d
```

In [2]:
```
df=pd.read_csv('Social_Network_Ads.csv')
```

In [20]:
```
df.info()
df.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 3 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Age              400 non-null    int64
 1   EstimatedSalary  400 non-null    int64
 2   Purchased        400 non-null    int64
dtypes: int64(3)
memory usage: 9.5 KB
```

Out[20]:

	Age	EstimatedSalary	Purchased
0	19	19000	0
1	35	20000	0
2	26	43000	0
3	27	57000	0
4	19	76000	0

In [4]:
```
df.shape
```

Out[4]:
```
(400, 3)
```

In [5]:
```
X = df.iloc[:, [0, 2]].values
y = df.iloc[:, 2].values
```

In [6]:
```
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,rando
y_train
y_test
```

```
Out[6]: array([0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0,
               1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0,
               0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1,
               0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1])
```

```
In [7]: sc=StandardScaler()
        X_train = sc.fit_transform(X_train)
        X_test = sc.transform(X_test)
```

```
In [8]: model=LogisticRegression()
        model.fit(X_train,y_train)
```

```
Out[8]: ▾ LogisticRegression

        LogisticRegression()
```

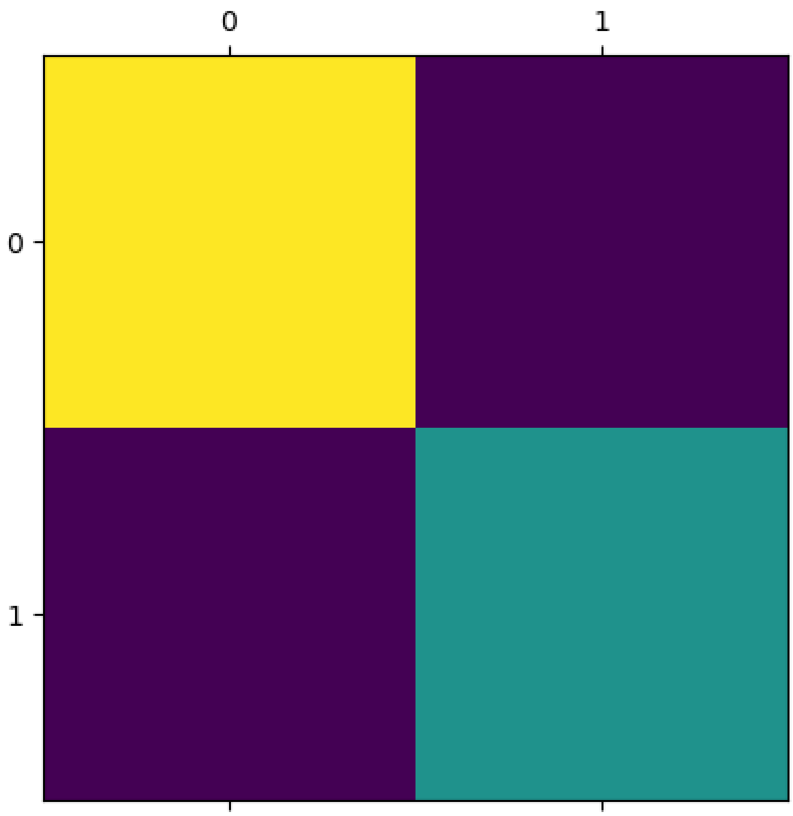
```
In [9]: y_pred = model.predict(X_test)
```

```
In [10]: y_pred
```

```
Out[10]: array([0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0,
                1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0,
                0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1,
                0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1])
```

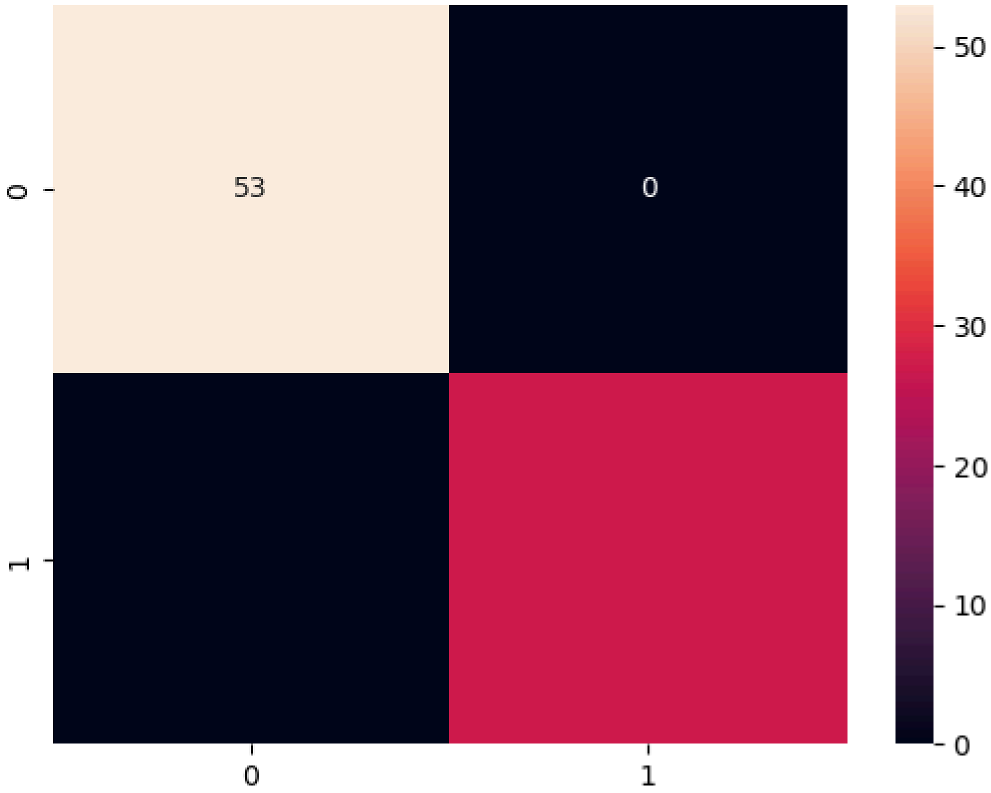
```
In [11]: from sklearn.metrics import confusion_matrix
         cm=confusion_matrix(y_test, y_pred)
         pl.matshow(cm)
```

```
Out[11]: <matplotlib.image.AxesImage at 0x13689ad90>
```



```
In [12]: sns.heatmap(cm,annot=True)
```

```
Out[12]: <Axes: >
```



```
In [13]: print(cm)
```

```
[[53  0]
 [ 0 27]]
```

In [14]:

```
from sklearn.metrics import accuracy_score
print("Accuracy: ",accuracy_score(y_test,y_pred))
```

```
Accuracy:  1.0
```

In [19]:

```
TN = cm[0][0]
FN = cm[1][0]
TP = cm[1][1]
FP = cm[0][1]
accuracy = (TN + TP)/(TN+FN+TP+FP)
error_rate = 1 - accuracy
precision = TP / (TP+FP)
recall = TP / (TP+FN)

print(TN)
print(FN)
print(FP)
print(TP)
print('Confusion matrix:\n', cm)
print('Accuracy:', accuracy)
print('Error rate:', error_rate)
print('Precision:', precision)
print('Recall:', recall)
```

```
53
0
0
27
Confusion matrix:
 [[53  0]
 [ 0 27]]
Accuracy: 1.0
Error rate: 0.0
Precision: 1.0
Recall: 1.0
```

In [16]:

```
report = classification_report(y_test, y_pred)
print("Classification Report:")
print(report)
```

```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        53
           1       1.00      1.00      1.00        27

    accuracy                           1.00        80
   macro avg       1.00      1.00      1.00        80
weighted avg       1.00      1.00      1.00        80
```

In [17]:

```
f1_score=(2*precision*recall)/(precision+recall)
print("F1 score is:",f1_score)
```

```
F1 score is: 1.0
```

```
In [ ]:
```