In [1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

from sklearn.metrics import confusion_matrix, classification_report, accuracy_
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```

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

In [3]:
```python
df
```

Out[3]:

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0
...	...	...	...	...	...
395	15691863	Female	46	41000	1
396	15706071	Male	51	23000	1
397	15654296	Female	50	20000	1
398	15755018	Male	36	33000	0
399	15594041	Female	49	36000	1

400 rows × 5 columns

In [4]:
```python
df.head()
```

Out[4]:

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0

In [5]:
```
df.head(10)
```

Out[5]:

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0
5	15728773	Male	27	58000	0
6	15598044	Female	27	84000	0
7	15694829	Female	32	150000	1
8	15600575	Male	25	33000	0
9	15727311	Female	35	65000	0

In [6]:
```
df.tail()
```

Out[6]:

	User ID	Gender	Age	EstimatedSalary	Purchased
395	15691863	Female	46	41000	1
396	15706071	Male	51	23000	1
397	15654296	Female	50	20000	1
398	15755018	Male	36	33000	0
399	15594041	Female	49	36000	1

In [7]:
```
df.tail(10)
```

Out[7]:

	User ID	Gender	Age	EstimatedSalary	Purchased
390	15807837	Male	48	33000	1
391	15592570	Male	47	23000	1
392	15748589	Female	45	45000	1
393	15635893	Male	60	42000	1
394	15757632	Female	39	59000	0
395	15691863	Female	46	41000	1
396	15706071	Male	51	23000	1
397	15654296	Female	50	20000	1
398	15755018	Male	36	33000	0
399	15594041	Female	49	36000	1

Basic Stats

In [8]:
```
df.shape
```

Out[8]:
```
(400, 5)
```

In [9]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   User ID          400 non-null    int64 
 1   Gender           400 non-null    object
 2   Age              400 non-null    int64 
 3   EstimatedSalary  400 non-null    int64 
 4   Purchased        400 non-null    int64 
dtypes: int64(4), object(1)
memory usage: 15.8+ KB
```

In [10]:
```
df.describe()
```

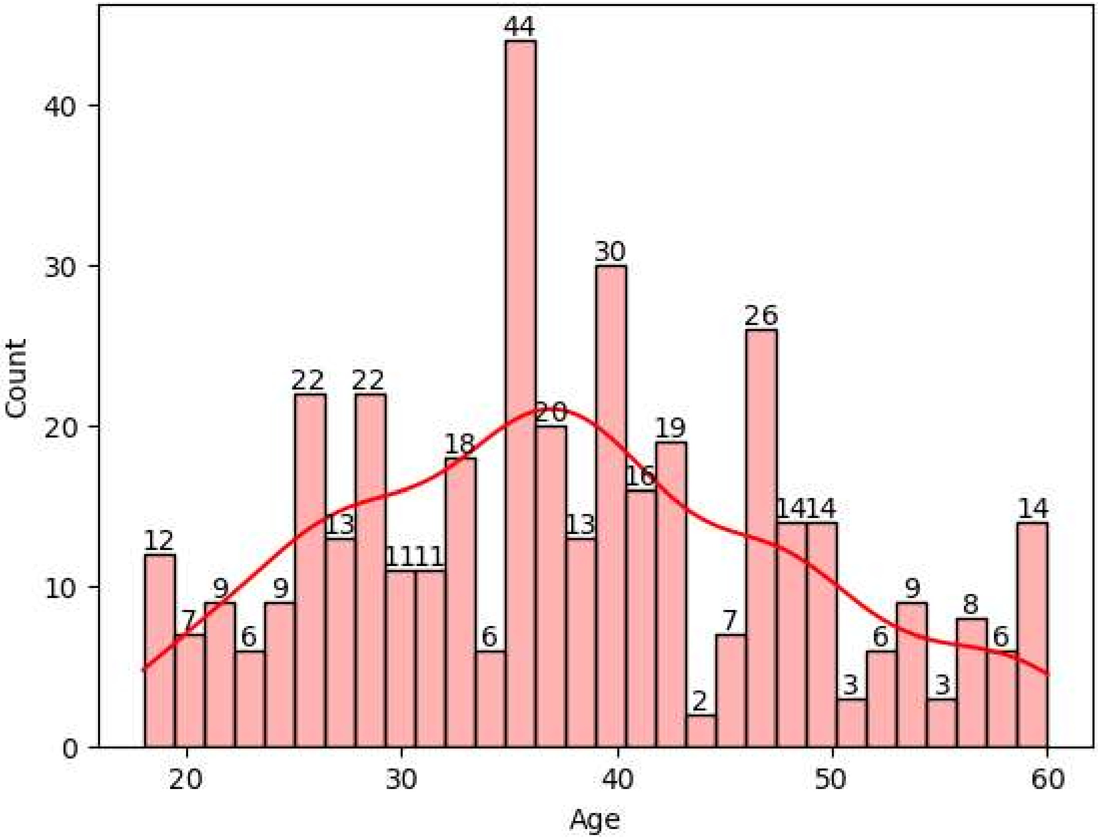
Out[10]:

	User ID	Age	EstimatedSalary	Purchased
count	4.000000e+02	400.000000	400.000000	400.000000
mean	1.569154e+07	37.655000	69742.500000	0.357500
std	7.165832e+04	10.482877	34096.960282	0.479864
min	1.556669e+07	18.000000	15000.000000	0.000000
25%	1.562676e+07	29.750000	43000.000000	0.000000
50%	1.569434e+07	37.000000	70000.000000	0.000000
75%	1.575036e+07	46.000000	88000.000000	1.000000
max	1.581524e+07	60.000000	150000.000000	1.000000

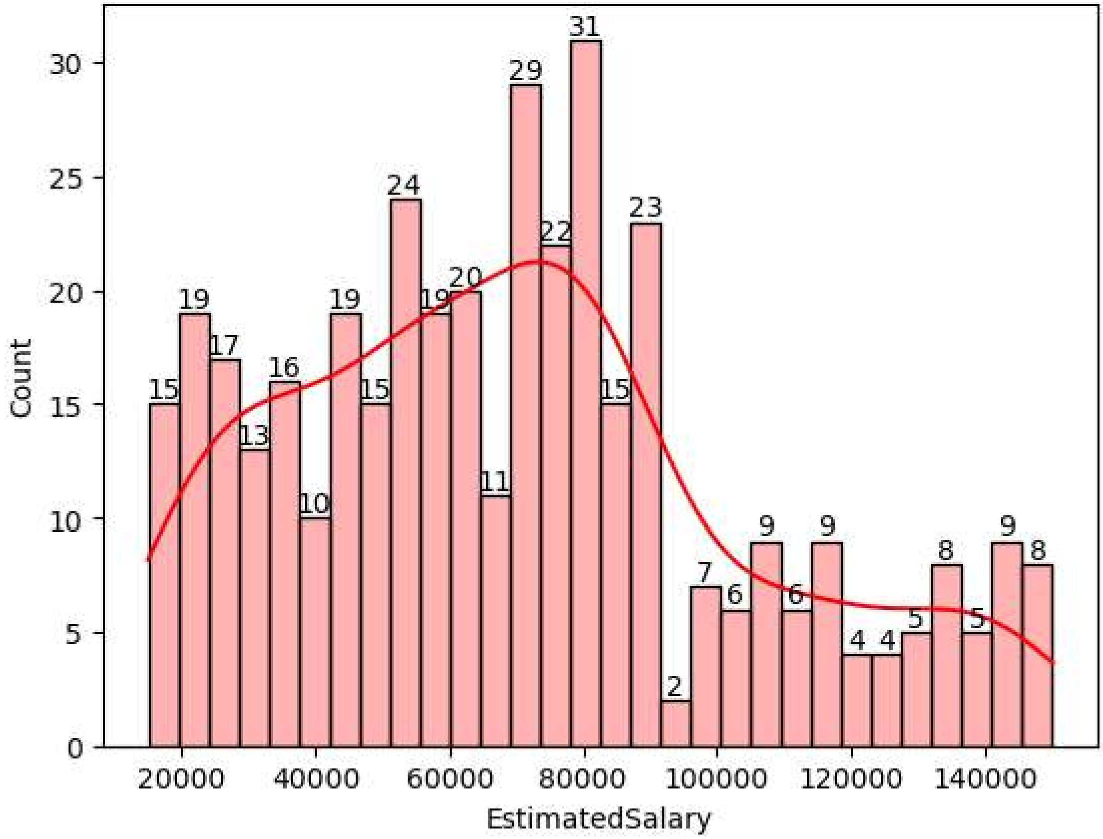
In [11]:
```
df.isnull().sum()
```

Out[11]:
```
User ID            0
Gender             0
Age                0
EstimatedSalary    0
Purchased          0
dtype: int64
```

In [12]:
```
histplot = sns.histplot(df['Age'], kde=True, bins=30, color='red', alpha=0.3)
for i in histplot.containers:
    histplot.bar_label(i,)
plt.show()
```



In [13]:
```
histplot = sns.histplot(df['EstimatedSalary'], kde=True, bins=30, color='red',
for i in histplot.containers:
    histplot.bar_label(i,)
plt.show()
```



In [14]:
```
df["Gender"].value_counts()
```

Out[14]:
```
Female    204
Male      196
Name: Gender, dtype: int64
```

In [15]:
```
def gender_encoder(value):
    if (value == "Male"):
        return 1
    elif (value == "Female"):
        return 0
    else:
        return -1
```

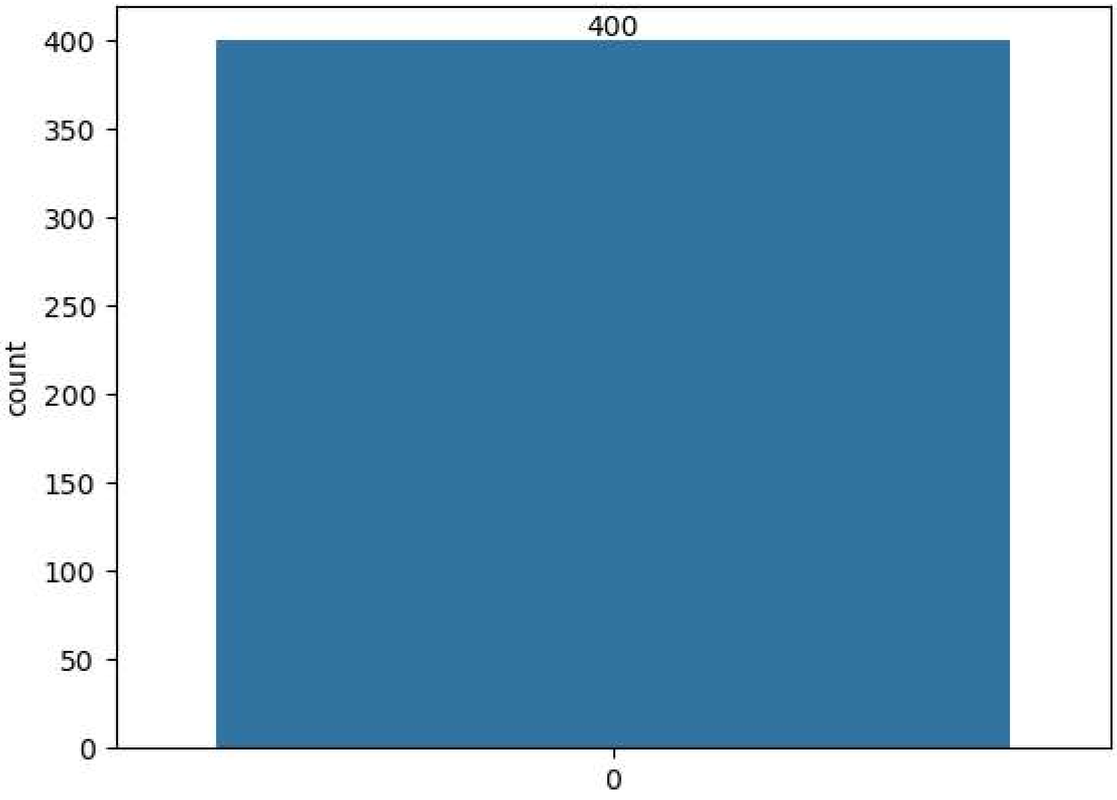
In [16]:
```
df["Gender"] = df["Gender"].apply(gender_encoder)
```

In [17]:
```
df["Purchased"].value_counts()
```

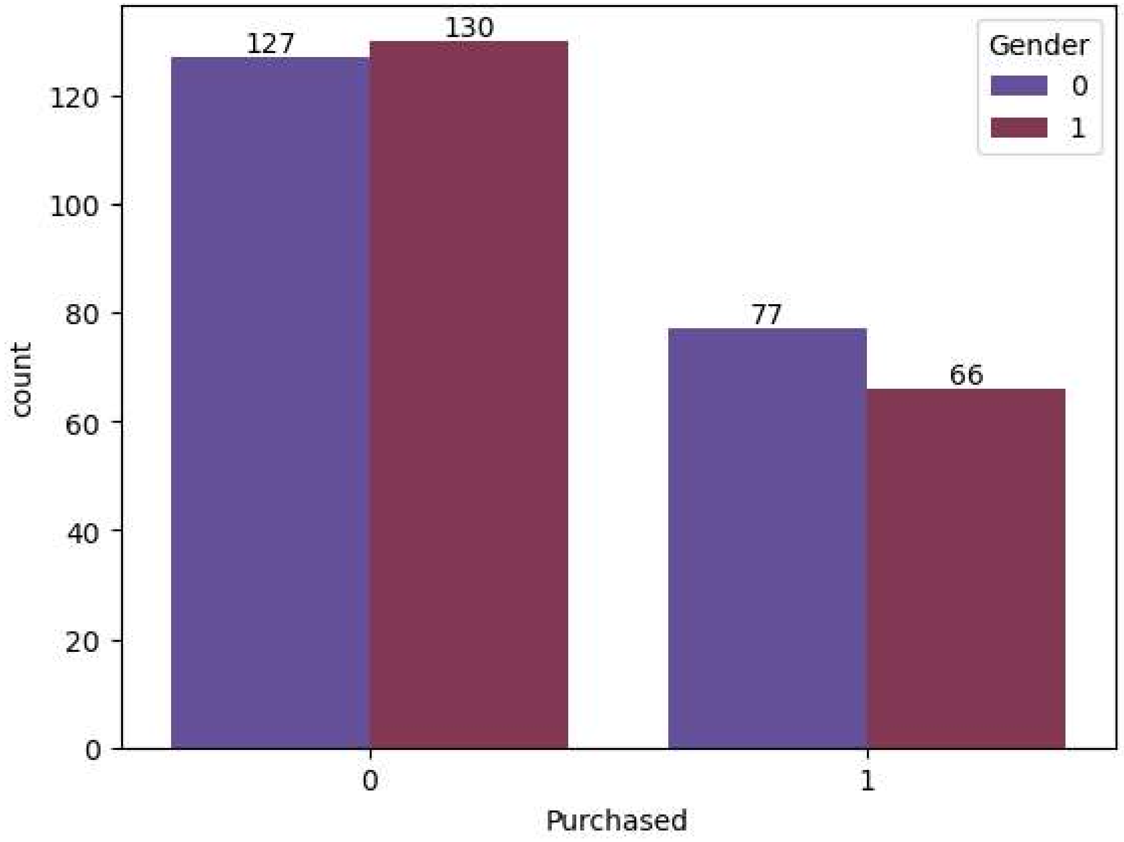
Out[17]:
```
0    257
1    143
Name: Purchased, dtype: int64
```

In [36]:

```
countplot = sns.countplot(df["Purchased"])
for i in countplot.containers:
    countplot.bar_label(i,)
plt.show()
```

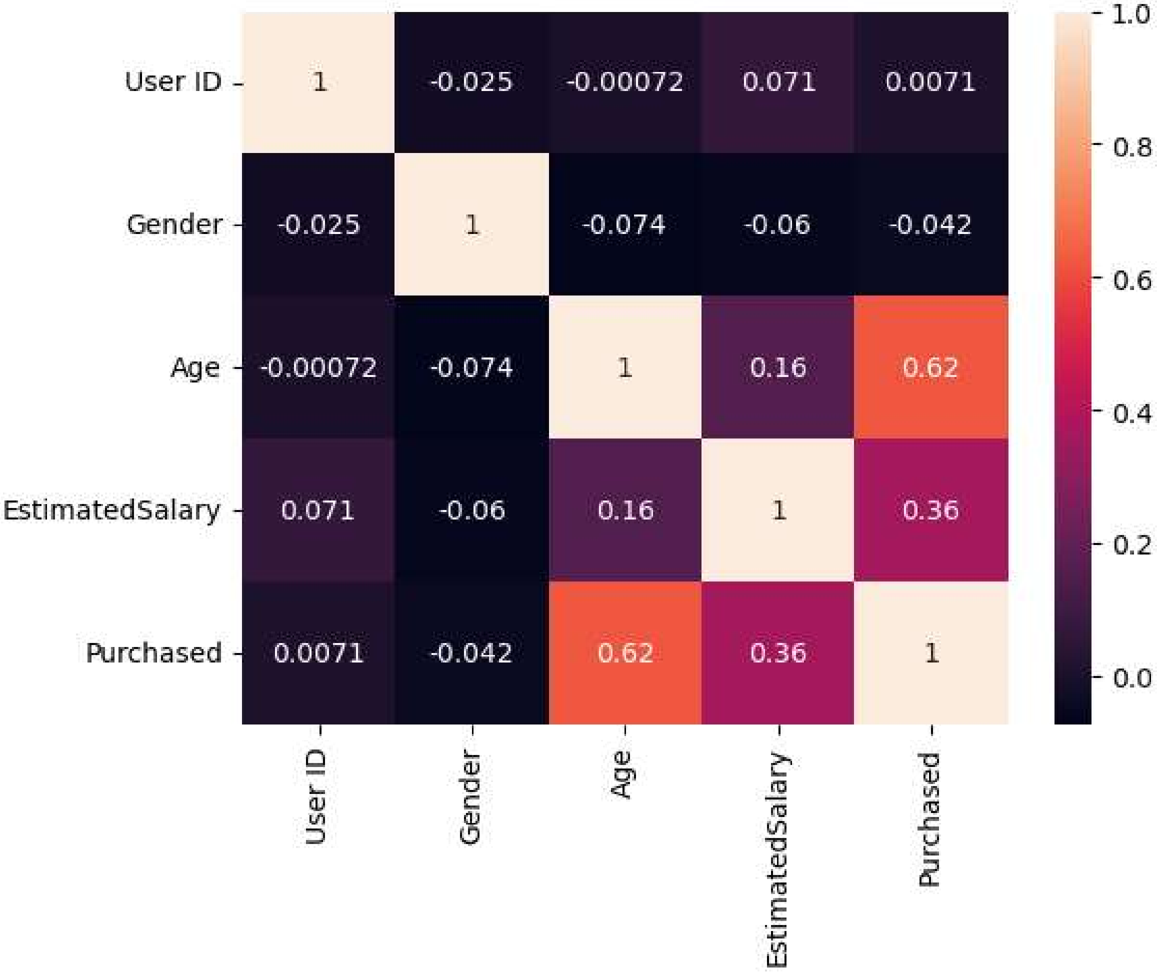


```
In [19]: # Create the countplot with hue
         countplot = sns.countplot(x=df["Purchased"], hue=df["Gender"], palette="twilig

         # Add labels to the bars
         for i in countplot.containers:
             countplot.bar_label(i)

         # Display the plot
         plt.show()
```



In [20]:

```
sns.heatmap(df.corr(), annot=True)
plt.show()
```



Data Preperation

In [21]:

```
x = df[["Age", "EstimatedSalary"]]
y = df["Purchased"]
```

In [22]:

```
scaler = StandardScaler()
x = scaler.fit_transform(x)
```

In [23]:

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, rando
```

In [24]:

```
x_train.shape, x_test.shape, y_train.shape, y_test.shape
```

Out[24]:

```
((320, 2), (80, 2), (320,), (80,))
```

Model Building

In [25]:

```
model = LogisticRegression(n_jobs=-1)
```

In [26]:

```
model.fit(x_train, y_train)
```

Out[26]:

```
LogisticRegression(n_jobs=-1)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [27]:

```
y_pred = model.predict(x_test)
```

Evaluation

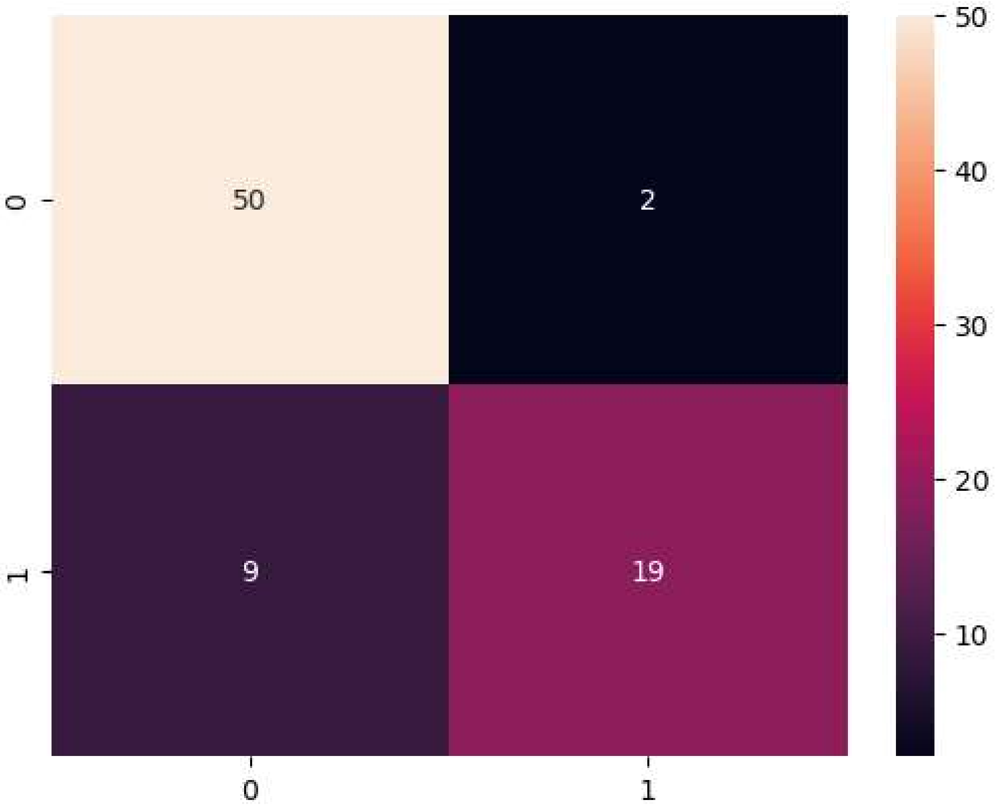
In [28]:

```
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
[[50  2]
 [ 9 19]]
```

In [29]:

```
sns.heatmap(confusion_matrix(y_test, y_pred), annot= True)
plt.show()
```



In [30]:

```
print(f"TN value is {cm[0][0]}")
print(f"FP value is {cm[0][1]}")
print(f"FN value is {cm[1][0]}")
print(f"TP value is {cm[1][1]}")
```

```
TN value is 50
FP value is 2
FN value is 9
TP value is 19
```

In [31]:

```
print(f"Accuracy score is {accuracy_score(y_test, y_pred)}")
```

```
Accuracy score is 0.8625
```

In [32]:

```
print(f"Error rate is {1-accuracy_score(y_test, y_pred)}")
```

```
Error rate is 0.13749999999999996
```

In [33]:

```
print(f"Precision score is {precision_score(y_test, y_pred)}")
```

```
Precision score is 0.9047619047619048
```

In [34]:

```
print(f"Recall score is {recall_score(y_test, y_pred)}")
```

```
Recall score is 0.6785714285714286
```

In [35]:

```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.85      0.96      0.90        52
           1       0.90      0.68      0.78        28

    accuracy                           0.86        80
   macro avg       0.88      0.82      0.84        80
weighted avg       0.87      0.86      0.86        80
```

In [ ]:

In [ ]:

In [ ]: