Customer Satisfication Prediction

In [1]:
```
# importing all the libraries
```

In [54]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler,LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
import warnings
warnings.filterwarnings("ignore")
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score,precision_score,recall_score,f1_score,confusion_matrix,
```

In [3]:
```
#import dataset
```

In [4]:
```
df=pd.read_csv("C:/Users/user/Downloads/customer_support_tickets.csv")
df
```

Out[4]:

	Ticket ID	Customer Name	Customer Email	Customer Age	Customer Gender	Product Purchased	Date of Purchase	Ticket Type	Ticket Subject	
0	1	Marisa Obrien	carrollallison@example.com	32	Other	GoPro Hero	2021-03-22	Technical issue	Product setup	{
1	2	Jessica Rios	clarkeashley@example.com	42	Female	LG Smart TV	2021-05-22	Technical issue	Peripheral compatibility	{
2	3	Christopher Robbins	gonzalestracy@example.com	48	Other	Dell XPS	2020-07-14	Technical issue	Network problem	I {
3	4	Christina Dillon	bradleyolson@example.org	27	Female	Microsoft Office	2020-11-13	Billing inquiry	Account access	{
4	5	Alexander Carroll	bradleymark@example.com	67	Female	Autodesk AutoCAD	2020-02-04	Billing inquiry	Data loss	{
...	...	...	...	...	...	...	...	...	...	
8464	8465	David Todd	adam28@example.net	22	Female	LG OLED	2021-12-08	Product inquiry	Installation support	{I
8465	8466	Lori Davis	russell68@example.com	27	Female	Bose SoundLink Speaker	2020-02-22	Technical issue	Refund request	{
8466	8467	Michelle Kelley	ashley83@example.org	57	Female	GoPro Action Camera	2021-08-17	Technical issue	Account access	{
8467	8468	Steven Rodriguez	fpowell@example.org	54	Male	PlayStation	2021-10-16	Product inquiry	Payment issue	{
8468	8469	Steven Davis MD	lori20@example.net	53	Other	Philips Hue Lights	2020-06-01	Billing inquiry	Hardware issue	T

8469 rows × 17 columns

Exploratory Data Analysis

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

Out[5]:

	Ticket ID	Customer Name	Customer Email	Customer Age	Customer Gender	Product Purchased	Date of Purchase	Ticket Type	Ticket Subject	Ticl
0	1	Marisa Obrien	carrollallison@example.com	32	Other	GoPro Hero	2021-03-22	Technical issue	Product setup	I'm {proc
1	2	Jessica Rios	clarkeashley@example.com	42	Female	LG Smart TV	2021-05-22	Technical issue	Peripheral compatibility	I'm {proc
2	3	Christopher Robbins	gonzalestracy@example.com	48	Other	Dell XPS	2020-07-14	Technical issue	Network problem	I'm f {proc
3	4	Christina Dillon	bradleyolson@example.org	27	Female	Microsoft Office	2020-11-13	Billing inquiry	Account access	I'm {proc
4	5	Alexander Carroll	bradleymark@example.com	67	Female	Autodesk AutoCAD	2020-02-04	Billing inquiry	Data loss	I'm {proc

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

Out[6]:

	Ticket ID	Customer Name	Customer Email	Customer Age	Customer Gender	Product Purchased	Date of Purchase	Ticket Type	Ticket Subject	Ticket
8464	8465	David Todd	adam28@example.net	22	Female	LG OLED	2021-12-08	Product inquiry	Installation support	{product_ is ma
8465	8466	Lori Davis	russell68@example.com	27	Female	Bose SoundLink Speaker	2020-02-22	Technical issue	Refund request	I'm hav {product_
8466	8467	Michelle Kelley	ashley83@example.org	57	Female	GoPro Action Camera	2021-08-17	Technical issue	Account access	I'm hav {product_
8467	8468	Steven Rodriguez	fpowell@example.org	54	Male	PlayStation	2021-10-16	Product inquiry	Payment issue	I'm hav {product_
8468	8469	Steven Davis MD	lori20@example.net	53	Other	Philips Hue Lights	2020-06-01	Billing inquiry	Hardware issue	There se hardw

In [7]:
```
df.shape
```

Out[7]:
```
(8469, 17)
```

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

Out[8]:
```
Index(['Ticket ID', 'Customer Name', 'Customer Email', 'Customer Age',
       'Customer Gender', 'Product Purchased', 'Date of Purchase',
       'Ticket Type', 'Ticket Subject', 'Ticket Description', 'Ticket Status',
       'Resolution', 'Ticket Priority', 'Ticket Channel',
       'First Response Time', 'Time to Resolution',
       'Customer Satisfaction Rating'],
      dtype='object')
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8469 entries, 0 to 8468
Data columns (total 17 columns):
 #   Column                        Non-Null Count  Dtype  
---  ------                        --------------  -----  
 0   Ticket ID                     8469 non-null   int64  
 1   Customer Name                 8469 non-null   object 
 2   Customer Email                8469 non-null   object 
 3   Customer Age                  8469 non-null   int64  
 4   Customer Gender               8469 non-null   object 
 5   Product Purchased             8469 non-null   object 
 6   Date of Purchase              8469 non-null   object 
 7   Ticket Type                   8469 non-null   object 
 8   Ticket Subject                8469 non-null   object 
 9   Ticket Description            8469 non-null   object 
 10  Ticket Status                 8469 non-null   object 
 11  Resolution                    2769 non-null   object 
 12  Ticket Priority               8469 non-null   object 
 13  Ticket Channel                8469 non-null   object 
 14  First Response Time           5650 non-null   object 
 15  Time to Resolution            2769 non-null   object 
 16  Customer Satisfaction Rating  2769 non-null   float64
dtypes: float64(1), int64(2), object(14)
memory usage: 1.1+ MB
```

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

Out[10]:

	count	mean	std	min	25%	50%	75%	max
Ticket ID	8469.0	4235.000000	2444.934048	1.0	2118.0	4235.0	6352.0	8469.0
Customer Age	8469.0	44.026804	15.296112	18.0	31.0	44.0	57.0	70.0
Customer Satisfaction Rating	2769.0	2.991333	1.407016	1.0	2.0	3.0	4.0	5.0

In [11]:
```
df.drop(columns=["Ticket ID","Customer Name","Customer Email","Ticket Description"],axis=1,inplace
```

In [12]:
```
df
```

Out[12]:

	Customer Age	Customer Gender	Product Purchased	Date of Purchase	Ticket Type	Ticket Subject	Ticket Status	Resolution	Ticket Priority	Ticket Channel	Fi Respor Ti
0	32	Other	GoPro Hero	2021-03-22	Technical issue	Product setup	Pending Customer Response	NaN	Critical	Social media	2023- 12:15
1	42	Female	LG Smart TV	2021-05-22	Technical issue	Peripheral compatibility	Pending Customer Response	NaN	Critical	Chat	2023- 16:45
2	48	Other	Dell XPS	2020-07-14	Technical issue	Network problem	Closed	Case maybe show recently my computer follow.	Low	Social media	2023- 11:14
3	27	Female	Microsoft Office	2020-11-13	Billing inquiry	Account access	Closed	Try capital clearly never color toward story.	Low	Social media	2023- 07:29
4	67	Female	Autodesk AutoCAD	2020-02-04	Billing inquiry	Data loss	Closed	West decision evidence bit.	Low	Email	2023- 00:12
...	...	...	...	...	...	...	...	...	...	...	
8464	22	Female	LG OLED	2021-12-08	Product inquiry	Installation support	Open	NaN	Low	Phone	N
8465	27	Female	Bose SoundLink Speaker	2020-02-22	Technical issue	Refund request	Open	NaN	Critical	Email	N
8466	57	Female	GoPro Action Camera	2021-08-17	Technical issue	Account access	Closed	Eight account century nature kitchen.	High	Social media	2023- 09:44
8467	54	Male	PlayStation	2021-10-16	Product inquiry	Payment issue	Closed	We seat culture plan.	Medium	Email	2023- 18:28
8468	53	Other	Philips Hue Lights	2020-06-01	Billing inquiry	Hardware issue	Open	NaN	High	Phone	N

8469 rows × 13 columns

In [13]:
```
df.columns
```

Out[13]:
```
Index(['Customer Age', 'Customer Gender', 'Product Purchased',
       'Date of Purchase', 'Ticket Type', 'Ticket Subject', 'Ticket Status',
       'Resolution', 'Ticket Priority', 'Ticket Channel',
       'First Response Time', 'Time to Resolution',
       'Customer Satisfaction Rating'],
      dtype='object')
```

In [14]:
```
df.drop('Resolution',axis=1,inplace=True)
```

In [15]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8469 entries, 0 to 8468
Data columns (total 12 columns):
 #   Column                        Non-Null Count  Dtype  
---  ------                        --------------  -----  
 0   Customer Age                  8469 non-null   int64  
 1   Customer Gender               8469 non-null   object 
 2   Product Purchased             8469 non-null   object 
 3   Date of Purchase              8469 non-null   object 
 4   Ticket Type                   8469 non-null   object 
 5   Ticket Subject                8469 non-null   object 
 6   Ticket Status                 8469 non-null   object 
 7   Ticket Priority               8469 non-null   object 
 8   Ticket Channel                8469 non-null   object 
 9   First Response Time           5650 non-null   object 
 10  Time to Resolution            2769 non-null   object 
 11  Customer Satisfaction Rating  2769 non-null   float64
dtypes: float64(1), int64(1), object(10)
memory usage: 794.1+ KB
```

In [16]:
```
df['Date of Purchase']=pd.to_datetime(df['Date of Purchase'],errors="coerce")
df['First Response Time']=pd.to_datetime(df['First Response Time'],errors='coerce')
df['Time to Resolution']=pd.to_datetime(df['Time to Resolution'],errors='coerce')
```

In [17]:
```
df["total_time_taken"]=(df["Time to Resolution"]-df["First Response Time"]).dt.total_seconds()/360
```

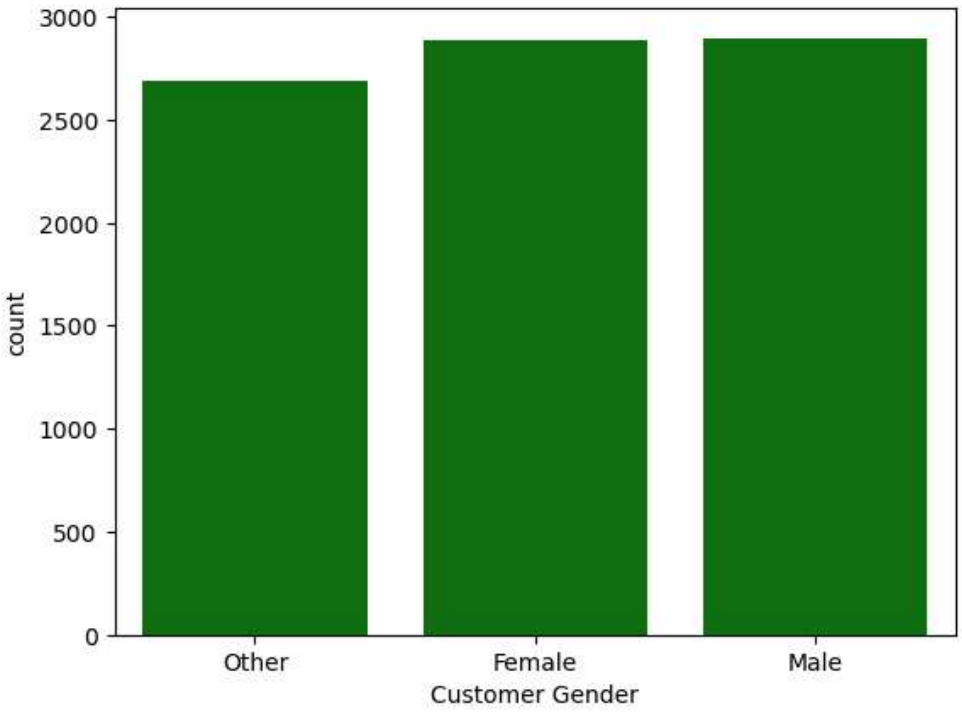
In [18]:
```
df["total_time_taken"]=df["total_time_taken"].fillna(df["total_time_taken"].median())
df["First Response Time"]=df["First Response Time"].fillna(df["First Response Time"].median())
df["Time to Resolution"]=df["Time to Resolution"].fillna(df["Time to Resolution"].median())
df["Customer Satisfaction Rating"]=df["Customer Satisfaction Rating"].fillna(df["Customer Satisfac
```

In [19]:
```
df["Customer Satisfaction Rating"].value_counts()
```

Out[19]:
```
Customer Satisfaction Rating
3.0    6280
1.0     553
2.0     549
5.0     544
4.0     543
Name: count, dtype: int64
```

In [20]:
```
sns.countplot(x="Customer Gender",data=df,color="Green")
```

Out[20]:
```
<Axes: xlabel='Customer Gender', ylabel='count'>
```

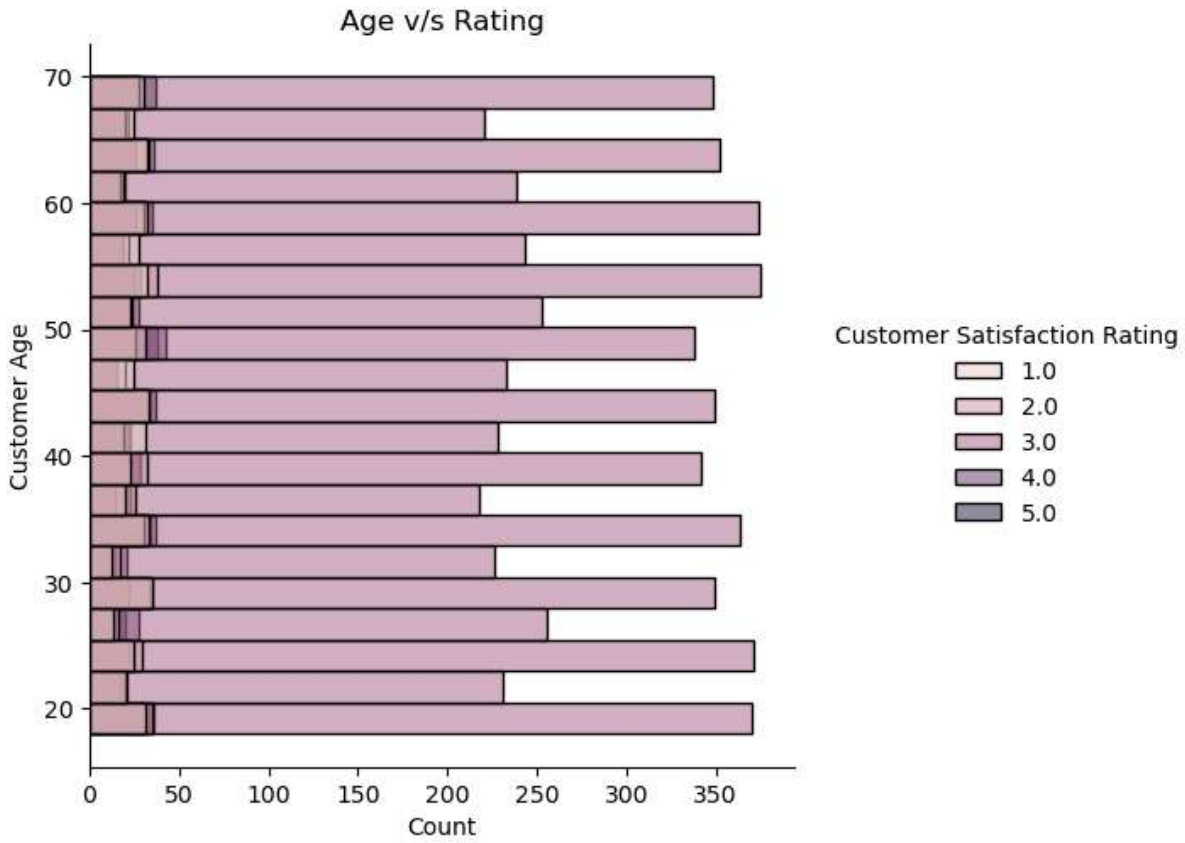


In [21]:
```
plt.figure(figsize=(15,4))
sns.displot(y="Customer Age",hue="Customer Satisfaction Rating",data=df,color="Blues")
plt.title("Age v/s Rating")
```

```
C:\Users\user\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure layout
has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

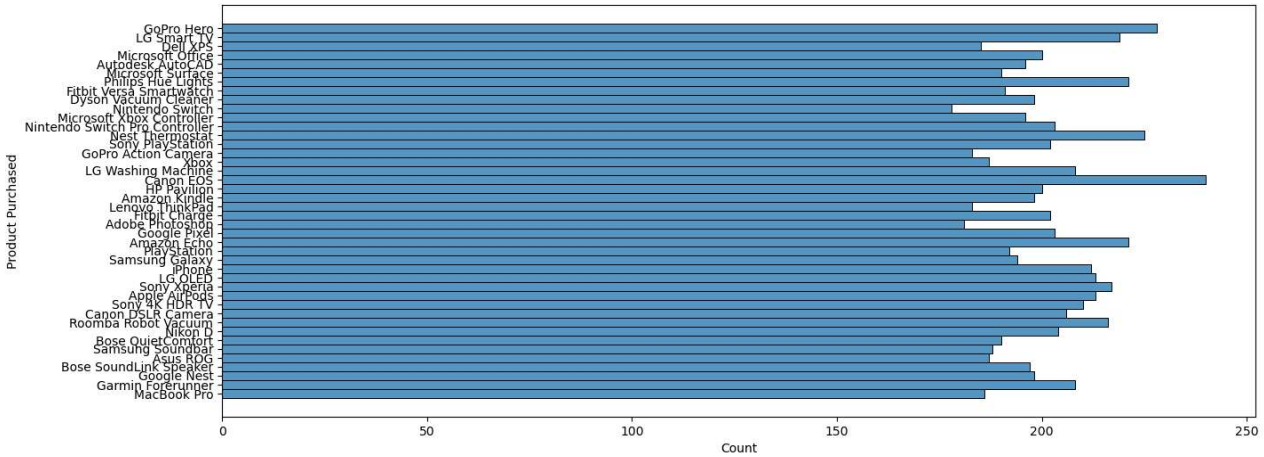
Out[21]:
```
Text(0.5, 1.0, 'Age v/s Rating')

<Figure size 1500x400 with 0 Axes>
```



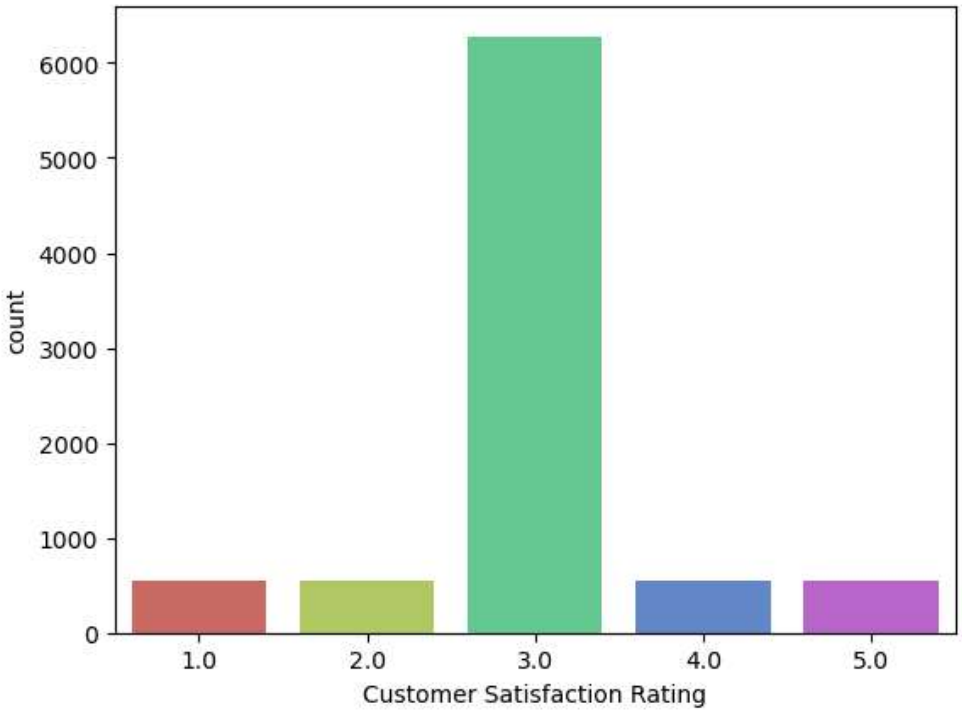
In [22]:
```
plt.figure(figsize=(15,6))
sns.histplot(y="Product Purchased",data=df)
```

Out[22]:
```
<Axes: xlabel='Count', ylabel='Product Purchased'>
```

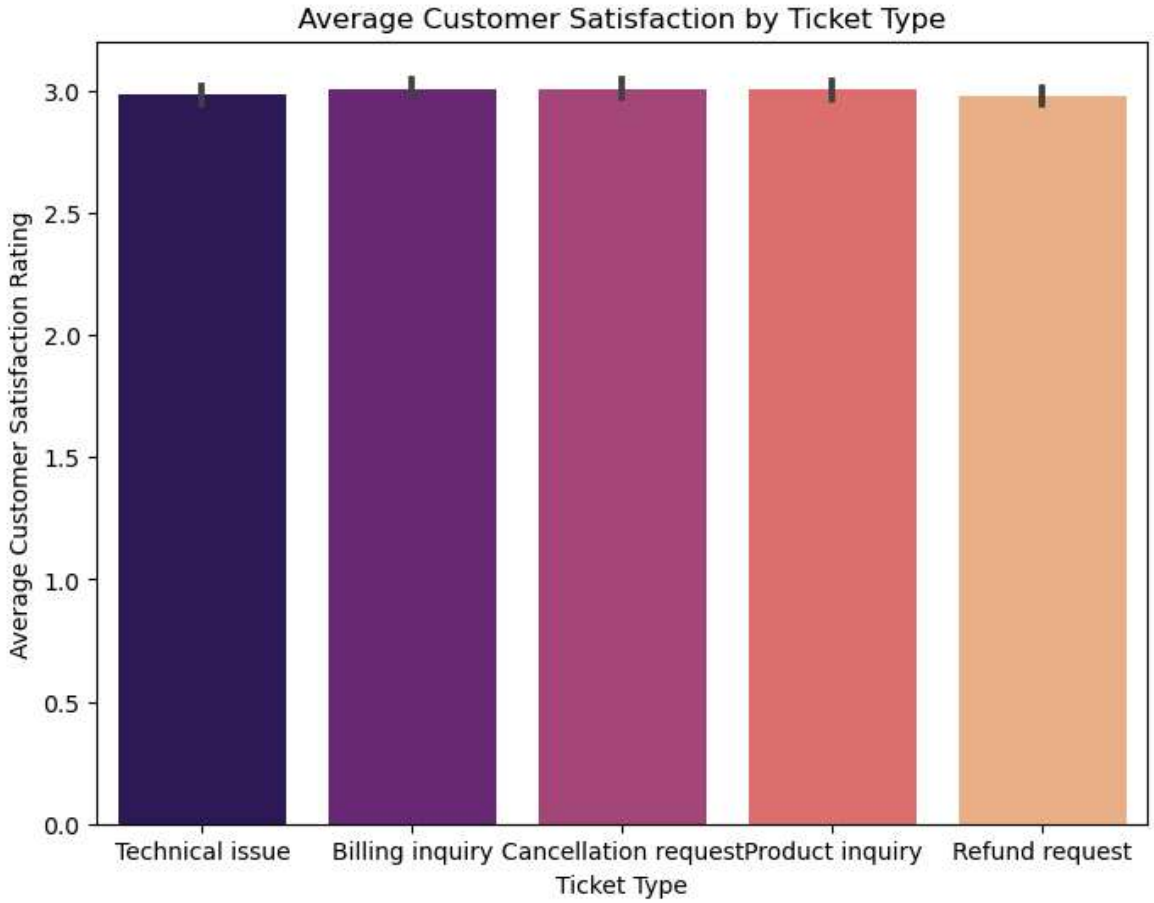


In [23]:
```
sns.countplot(x="Customer Satisfaction Rating",data=df,palette="hls")
```

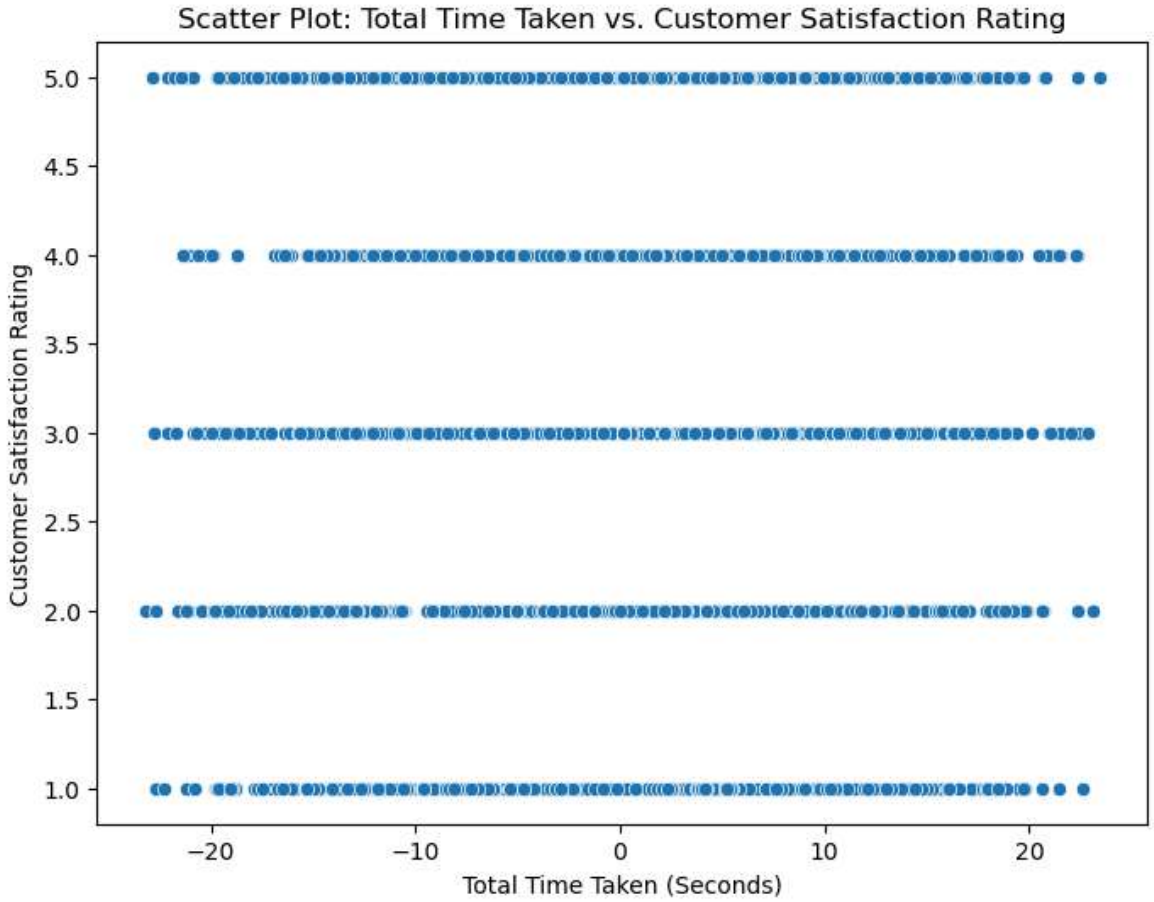
Out[23]:
```
<Axes: xlabel='Customer Satisfaction Rating', ylabel='count'>
```



In [24]:
```
plt.figure(figsize=(8, 6))
sns.barplot(x='Ticket Type', y='Customer Satisfaction Rating',data=df,palette="magma")
plt.title('Average Customer Satisfaction by Ticket Type')
plt.xlabel('Ticket Type')
plt.ylabel('Average Customer Satisfaction Rating')
plt.show()
```

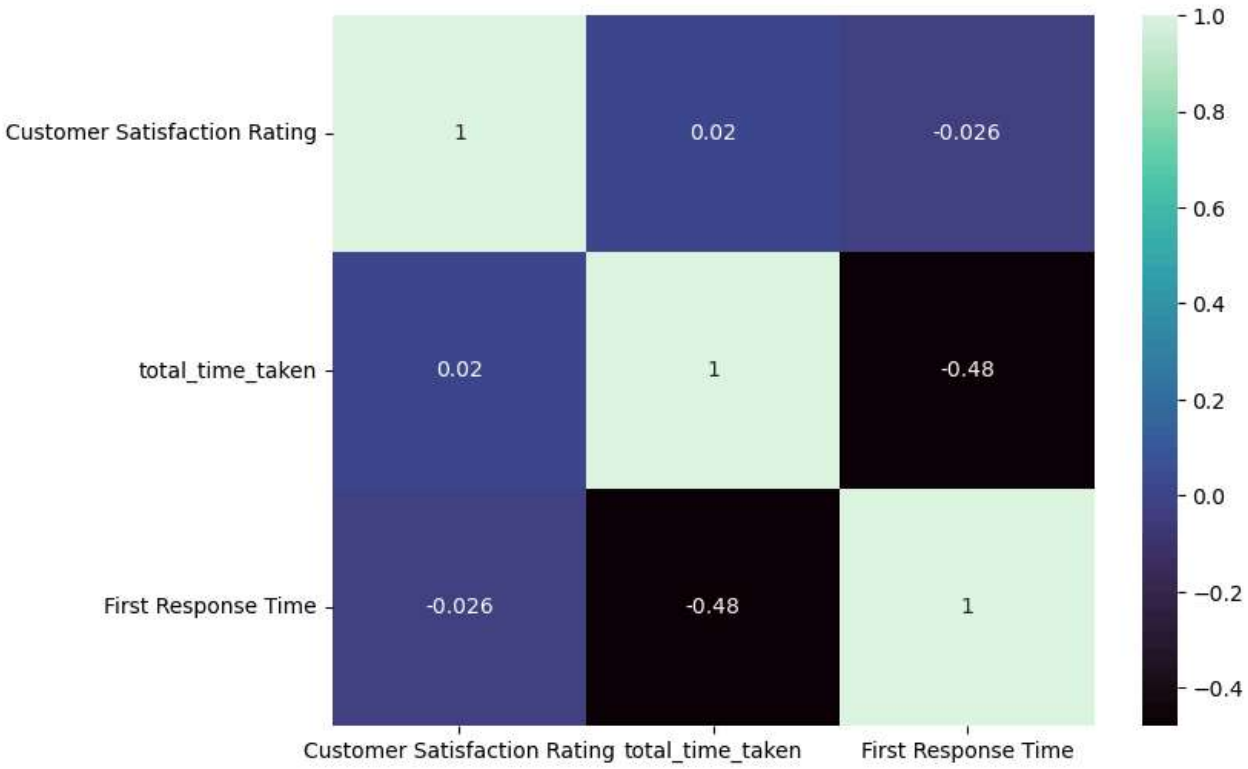


In [25]:
```
plt.figure(figsize=(8, 6))
sns.scatterplot(x='total_time_taken', y='Customer Satisfaction Rating', data=df)
plt.title('Scatter Plot: Total Time Taken vs. Customer Satisfaction Rating')
plt.xlabel('Total Time Taken (Seconds)')
plt.ylabel('Customer Satisfaction Rating')
plt.show()
```



In [26]:
```
correlation_matrix = df[['Customer Satisfaction Rating', 'total_time_taken', 'First Response Time'
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix,annot=True,cmap="mako")
```

Out[26]: <Axes: >



In [27]:
```
df["purchase_day"]=df["Date of Purchase"].dt.day
df['Purchase_Month'] = df['Date of Purchase'].dt.month
df['Purchase_year'] = df['Date of Purchase'].dt.year
```

In [28]:
```
df['First Response Year'] = df['First Response Time'].dt.year
df['First Response Month'] = df['First Response Time'].dt.month
df['First Response Day'] = df['First Response Time'].dt.day
df['First Response Hour'] = df['First Response Time'].dt.hour
df['First Response Minute'] = df['First Response Time'].dt.minute
df['Resolution Year'] = df['Time to Resolution'].dt.year
df['Resolution Month'] = df['Time to Resolution'].dt.month
df['Resolution Day'] = df['Time to Resolution'].dt.day
df['Resolution Hour'] = df['Time to Resolution'].dt.hour
df['Resolution Minute'] = df['Time to Resolution'].dt.minute
```

In [29]:
```
df.drop(columns=["Date of Purchase"],inplace=True,errors="ignore")
```

In [30]:
```
df.drop(columns=["First Response Time","Time to Resolution"],inplace=True,errors="ignore")
```

In [31]:
```
df
```

Out[31]:

	Customer Age	Customer Gender	Product Purchased	Ticket Type	Ticket Subject	Ticket Status	Ticket Priority	Ticket Channel	Customer Satisfaction Rating	total_time_taken
0	32	Other	GoPro Hero	Technical issue	Product setup	Pending Customer Response	Critical	Social media	3.0	0.166667
1	42	Female	LG Smart TV	Technical issue	Peripheral compatibility	Pending Customer Response	Critical	Chat	3.0	0.166667
2	48	Other	Dell XPS	Technical issue	Network problem	Closed	Low	Social media	3.0	6.850000
3	27	Female	Microsoft Office	Billing inquiry	Account access	Closed	Low	Social media	3.0	-5.533333
4	67	Female	Autodesk AutoCAD	Billing inquiry	Data loss	Closed	Low	Email	1.0	19.683333
...	...	...	...	...	...	...	...	...	...	...
8464	22	Female	LG OLED	Product inquiry	Installation support	Open	Low	Phone	3.0	0.166667
8465	27	Female	Bose SoundLink Speaker	Technical issue	Refund request	Open	Critical	Email	3.0	0.166667
8466	57	Female	GoPro Action Camera	Technical issue	Account access	Closed	High	Social media	3.0	-5.216667
8467	54	Male	PlayStation	Product inquiry	Payment issue	Closed	Medium	Email	3.0	-12.933333
8468	53	Other	Philips Hue Lights	Billing inquiry	Hardware issue	Open	High	Phone	3.0	0.166667

8469 rows × 23 columns

In [32]:
```
df.select_dtypes(include="object")
```

Out[32]:

	Customer Gender	Product Purchased	Ticket Type	Ticket Subject	Ticket Status	Ticket Priority	Ticket Channel
0	Other	GoPro Hero	Technical issue	Product setup	Pending Customer Response	Critical	Social media
1	Female	LG Smart TV	Technical issue	Peripheral compatibility	Pending Customer Response	Critical	Chat
2	Other	Dell XPS	Technical issue	Network problem	Closed	Low	Social media
3	Female	Microsoft Office	Billing inquiry	Account access	Closed	Low	Social media
4	Female	Autodesk AutoCAD	Billing inquiry	Data loss	Closed	Low	Email
...	...	...	...	...	...	...	...
8464	Female	LG OLED	Product inquiry	Installation support	Open	Low	Phone
8465	Female	Bose SoundLink Speaker	Technical issue	Refund request	Open	Critical	Email
8466	Female	GoPro Action Camera	Technical issue	Account access	Closed	High	Social media
8467	Male	PlayStation	Product inquiry	Payment issue	Closed	Medium	Email
8468	Other	Philips Hue Lights	Billing inquiry	Hardware issue	Open	High	Phone

8469 rows × 7 columns

In [33]:
```
label_encoder=LabelEncoder()
df['Customer Gender'] = label_encoder.fit_transform(df['Customer Gender'])
df['Ticket Priority'] = label_encoder.fit_transform(df['Ticket Priority'])
df['Ticket Status'] = label_encoder.fit_transform(df['Ticket Status'])
```

In [34]:
```
df=pd.get_dummies(df,["Product Purchased","Ticket Type","Ticket Subject","Ticket Channel"])
```

In [35]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8469 entries, 0 to 8468
Data columns (total 86 columns):
 #   Column                                          Non-Null Count  Dtype  
---  ------                                          --------------  -----  
 0   Customer Age                                    8469 non-null   int64  
 1   Customer Gender                                 8469 non-null   int32  
 2   Ticket Status                                   8469 non-null   int32  
 3   Ticket Priority                                 8469 non-null   int32  
 4   Customer Satisfaction Rating                    8469 non-null   float64
 5   total_time_taken                                8469 non-null   float64
 6   purchase_day                                    8469 non-null   int32  
 7   Purchase_Month                                  8469 non-null   int32  
 8   Purchase_year                                   8469 non-null   int32  
 9   First Response Year                             8469 non-null   int32  
 10  First Response Month                            8469 non-null   int32  
 11  First Response Day                              8469 non-null   int32  
 12  First Response Hour                             8469 non-null   int32  
 13  First Response Minute                           8469 non-null   int32  
 14  Resolution Year                                 8469 non-null   int32  
 15  Resolution Month                                8469 non-null   int32  
 16  Resolution Day                                  8469 non-null   int32  
 17  Resolution Hour                                 8469 non-null   int32  
 18  Resolution Minute                               8469 non-null   int32  
 19  Product Purchased_Adobe Photoshop               8469 non-null   bool   
 20  Product Purchased_Amazon Echo                   8469 non-null   bool   
 21  Product Purchased_Amazon Kindle                 8469 non-null   bool   
 22  Product Purchased_Apple AirPods                 8469 non-null   bool   
 23  Product Purchased_Asus ROG                      8469 non-null   bool   
 24  Product Purchased_Autodesk AutoCAD              8469 non-null   bool   
 25  Product Purchased_Bose QuietComfort             8469 non-null   bool   
 26  Product Purchased_Bose SoundLink Speaker        8469 non-null   bool   
 27  Product Purchased_Canon DSLR Camera             8469 non-null   bool   
 28  Product Purchased_Canon EOS                     8469 non-null   bool   
 29  Product Purchased_Dell XPS                      8469 non-null   bool   
 30  Product Purchased_Dyson Vacuum Cleaner          8469 non-null   bool   
 31  Product Purchased_Fitbit Charge                 8469 non-null   bool   
 32  Product Purchased_Fitbit Versa Smartwatch       8469 non-null   bool   
 33  Product Purchased_Garmin Forerunner             8469 non-null   bool   
 34  Product Purchased_GoPro Action Camera           8469 non-null   bool   
 35  Product Purchased_GoPro Hero                    8469 non-null   bool   
 36  Product Purchased_Google Nest                   8469 non-null   bool   
 37  Product Purchased_Google Pixel                  8469 non-null   bool   
 38  Product Purchased_HP Pavilion                   8469 non-null   bool   
 39  Product Purchased_LG OLED                       8469 non-null   bool   
 40  Product Purchased_LG Smart TV                   8469 non-null   bool   
 41  Product Purchased_LG Washing Machine            8469 non-null   bool   
 42  Product Purchased_Lenovo ThinkPad               8469 non-null   bool   
 43  Product Purchased_MacBook Pro                   8469 non-null   bool   
 44  Product Purchased_Microsoft Office              8469 non-null   bool   
 45  Product Purchased_Microsoft Surface             8469 non-null   bool   
 46  Product Purchased_Microsoft Xbox Controller     8469 non-null   bool   
 47  Product Purchased_Nest Thermostat               8469 non-null   bool   
 48  Product Purchased_Nikon D                       8469 non-null   bool   
 49  Product Purchased_Nintendo Switch               8469 non-null   bool   
 50  Product Purchased_Nintendo Switch Pro Controller  8469 non-null   bool   
 51  Product Purchased_Philips Hue Lights            8469 non-null   bool   
 52  Product Purchased_PlayStation                   8469 non-null   bool   
 53  Product Purchased_Roomba Robot Vacuum           8469 non-null   bool   
 54  Product Purchased_Samsung Galaxy                8469 non-null   bool   
 55  Product Purchased_Samsung Soundbar              8469 non-null   bool   
 56  Product Purchased_Sony 4K HDR TV                8469 non-null   bool   
 57  Product Purchased_Sony PlayStation              8469 non-null   bool   
 58  Product Purchased_Sony Xperia                   8469 non-null   bool   
 59  Product Purchased_Xbox                          8469 non-null   bool   
 60  Product Purchased_iPhone                        8469 non-null   bool   
 61  Ticket Type_Billing inquiry                     8469 non-null   bool   
```

```
 62  Ticket Type_Cancellation request                8469 non-null   bool
 63  Ticket Type_Product inquiry                     8469 non-null   bool
 64  Ticket Type_Refund request                      8469 non-null   bool
 65  Ticket Type_Technical issue                     8469 non-null   bool
 66  Ticket Subject_Account access                   8469 non-null   bool
 67  Ticket Subject_Battery life                     8469 non-null   bool
 68  Ticket Subject_Cancellation request             8469 non-null   bool
 69  Ticket Subject_Data loss                        8469 non-null   bool
 70  Ticket Subject_Delivery problem                 8469 non-null   bool
 71  Ticket Subject_Display issue                    8469 non-null   bool
 72  Ticket Subject_Hardware issue                   8469 non-null   bool
 73  Ticket Subject_Installation support             8469 non-null   bool
 74  Ticket Subject_Network problem                  8469 non-null   bool
 75  Ticket Subject_Payment issue                    8469 non-null   bool
 76  Ticket Subject_Peripheral compatibility         8469 non-null   bool
 77  Ticket Subject_Product compatibility            8469 non-null   bool
 78  Ticket Subject_Product recommendation           8469 non-null   bool
 79  Ticket Subject_Product setup                    8469 non-null   bool
 80  Ticket Subject_Refund request                   8469 non-null   bool
 81  Ticket Subject_Software bug                     8469 non-null   bool
 82  Ticket Channel_Chat                             8469 non-null   bool
 83  Ticket Channel_Email                            8469 non-null   bool
 84  Ticket Channel_Phone                            8469 non-null   bool
 85  Ticket Channel_Social media                     8469 non-null   bool
dtypes: bool(67), float64(2), int32(16), int64(1)
memory usage: 1.3 MB
```

In [36]:
```
df[df.select_dtypes(include='bool').columns] = df.select_dtypes(include='bool').astype(int)
```

In [37]:
```
df
```

Out[37]:

	Customer Age	Customer Gender	Ticket Status	Ticket Priority	Customer Satisfaction Rating	total_time_taken	purchase_day	Purchase_Month	Purchase_year
0	32	2	2	0	3.0	0.166667	22	3	2021
1	42	0	2	0	3.0	0.166667	22	5	2021
2	48	2	0	2	3.0	6.850000	14	7	2020
3	27	0	0	2	3.0	-5.533333	13	11	2020
4	67	0	0	2	1.0	19.683333	4	2	2020
...	...	...	...	...	...	...	...	...	...
8464	22	0	1	2	3.0	0.166667	8	12	2021
8465	27	0	1	0	3.0	0.166667	22	2	2020
8466	57	0	0	1	3.0	-5.216667	17	8	2021
8467	54	1	0	3	3.0	-12.933333	16	10	2021
8468	53	2	1	1	3.0	0.166667	1	6	2020

8469 rows × 86 columns

In [38]:
```
x=df.drop(columns=["Customer Satisfaction Rating"],axis=1)
y=df['Customer Satisfaction Rating']
```

In [39]:
```
scaler=StandardScaler()
scaler.fit_transform(x)
```

Out[39]:
```
array([[-0.7863118 ,  1.26253802,  1.20827849, ..., -0.5820315 ,
        -0.58003172,  1.73000805],
       [-0.13251232, -1.2039982 ,  1.20827849, ..., -0.5820315 ,
        -0.58003172, -0.57803199],
       [ 0.25976737,  1.26253802, -1.24066504, ..., -0.5820315 ,
        -0.58003172,  1.73000805],
       ...,
       [ 0.84818691, -1.2039982 , -1.24066504, ..., -0.5820315 ,
        -0.58003172,  1.73000805],
       [ 0.65204706,  0.02926991, -1.24066504, ...,  1.71812006,
        -0.58003172, -0.57803199],
       [ 0.58666711,  1.26253802, -0.01619327, ..., -0.5820315 ,
         1.72404363, -0.57803199]])
```

In [40]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=42)
```

In [41]:
```
model=LogisticRegression()
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
accuracy_lr=accuracy_score(y_test,y_pred)
accuracy_lr
```

```
C:\Users\user\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.py:465: ConvergenceWarni
ng: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-learn.org/stable/m
odules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (https://scikit
-learn.org/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

Out[41]:
```
0.731995277449823
```

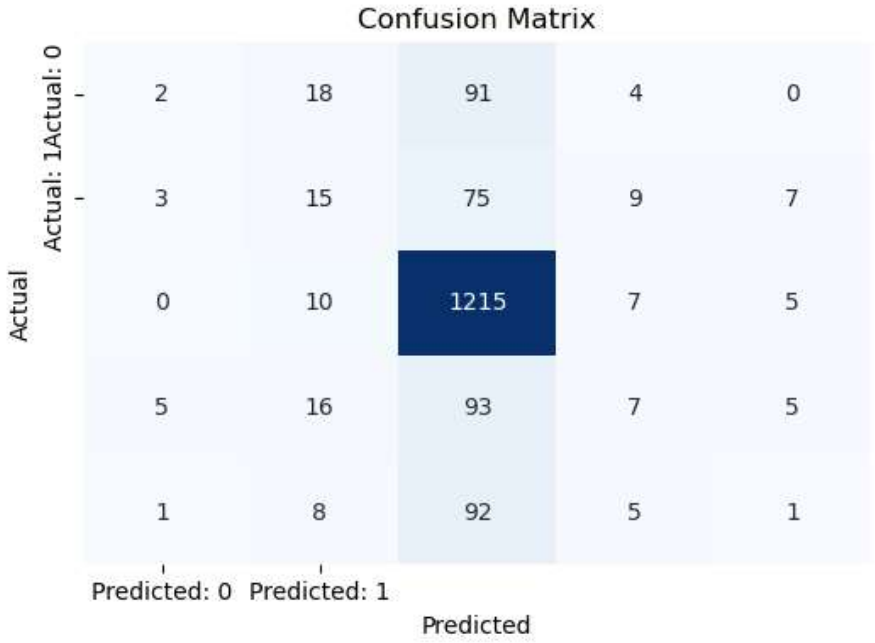
In [42]:
```
print("confusion metrix of Logistic Regression")
cm=confusion_matrix(y_test,y_pred)
cm
```

```
confusion metrix of Logistic Regression
```

Out[42]:
```
array([[   2,   18,   91,    4,    0],
       [   3,   15,   75,    9,    7],
       [   0,   10, 1215,    7,    5],
       [   5,   16,   93,    7,    5],
       [   1,    8,   92,    5,    1]], dtype=int64)
```

In [50]:
```
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", cbar=False,
            xticklabels=["Predicted: 0", "Predicted: 1"],
            yticklabels=["Actual: 0", "Actual: 1"])

plt.title("Confusion Matrix")
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.show()
```



In [49]:
```
model_2=DecisionTreeClassifier()
model_2.fit(x_train,y_train)
y_pred_dt=model_2.predict(x_test)
accuracy_dt=accuracy_score(y_test,y_pred_dt)
accuracy_dt
```

Out[49]:
```
0.718417945690673
```

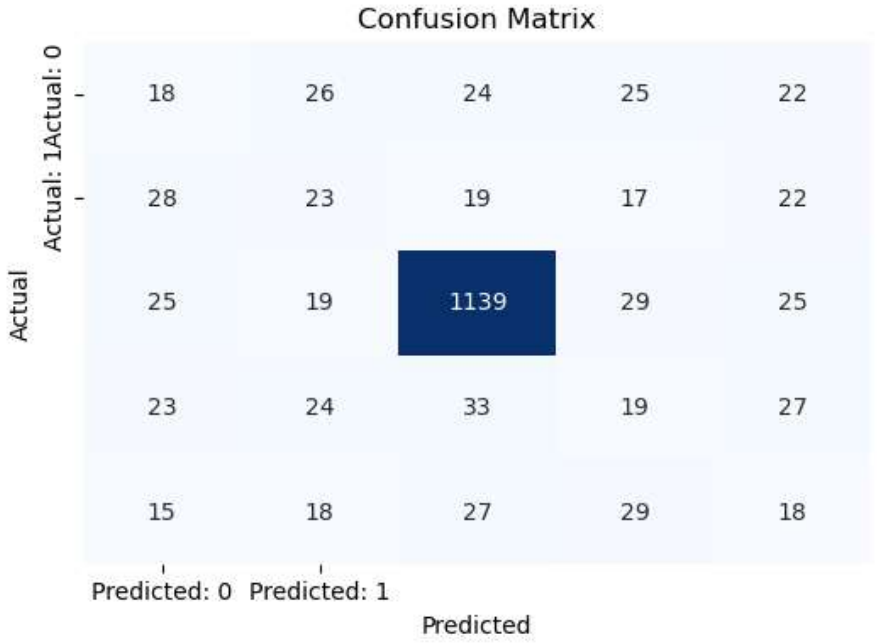
In [51]:
```
print("confusion metrix of Logistic Regression")
cm_dt=confusion_matrix(y_test,y_pred_dt)
cm_dt
```

```
confusion metrix of Logistic Regression
```

Out[51]:
```
array([[  18,   26,   24,   25,   22],
       [  28,   23,   19,   17,   22],
       [  25,   19, 1139,   29,   25],
       [  23,   24,   33,   19,   27],
       [  15,   18,   27,   29,   18]], dtype=int64)
```

In [52]:
```
plt.figure(figsize=(6, 4))
sns.heatmap(cm_dt, annot=True, fmt="d", cmap="Blues", cbar=False,
            xticklabels=["Predicted: 0", "Predicted: 1"],
            yticklabels=["Actual: 0", "Actual: 1"])

plt.title("Confusion Matrix")
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.show()
```



In [57]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
y_pred_rfc=rfc.predict(x_test)
accuracy_rfc=accuracy_score(y_test,y_pred_rfc)
accuracy_rfc
```

Out[57]:
```
0.7378984651711924
```

In [59]:
```
print("confusion metrix of Logistic Regression")
cm_rfc=confusion_matrix(y_test,y_pred_rfc)
cm_rfc
```

```
confusion metrix of Logistic Regression
```

Out[59]:
```
array([[  22,   27,   35,   10,   21],
       [  20,   20,   35,   16,   18],
       [  22,   24, 1154,   17,   20],
       [  17,   19,   31,   31,   28],
       [  14,   15,   40,   15,   23]], dtype=int64)
```

In [58]:
```
svc=SVC()
svc.fit(x_train,y_train)
y_pred_svc=svc.predict(x_test)
accuracy_svc=accuracy_score(y_test,y_pred_svc)
accuracy_svc
```

Out[58]:
```
0.7302243211334121
```

In [60]:
```
print("confusion metrix of Logistic Regression")
cm_svc=confusion_matrix(y_test,y_pred_svc)
cm_svc
```

```
confusion metrix of Logistic Regression
```

Out[60]:
```
array([[   0,    0,  115,    0,    0],
       [   0,    0,  109,    0,    0],
       [   0,    0, 1237,    0,    0],
       [   0,    0,  126,    0,    0],
       [   0,    0,  107,    0,    0]], dtype=int64)
```

CONCLUSION

Random Forest outperforms the other models, achieving the highest accuracy of 0.7379, followed closely by Logistic Regression (0.7320) and Support Vector Classifier (SVC) (0.7302). The Decision Tree model shows the lowest accuracy at 0.7184. This suggests that, at least for this dataset, ensemble models like Random Forest tend to deliver better generalization and predictive performance than individual models like Decision Tree or Logistic Regression.

Hyperparameter Tunning

In [62]:
```
from sklearn.model_selection import GridSearchCV
```

In [66]:
```
rfc=RandomForestClassifier()
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'max_features': ['auto', 'sqrt', 'log2'],
}
grid_search = GridSearchCV(estimator=rfc, param_grid=param_grid, cv=5, n_jobs=-1, verbose=2)
grid_search.fit(x_train, y_train)
print("Best parameters found: ", grid_search.best_params_)
```

```
Fitting 5 folds for each of 324 candidates, totalling 1620 fits
Best parameters found:  {'max_depth': 30, 'max_features': 'log2', 'min_samples_leaf': 1, 'min_sam
ples_split': 2, 'n_estimators': 50}
```

In [68]:
```
best_rf = grid_search.best_estimator_
print("Best Random Forest model accuracy: ", best_rf.score(x_test, y_test))
```

```
Best Random Forest model accuracy:  0.7331759149940968
```

In [ ]:
```
```