In [3]:
```
import pandas as pd
df=pd.read_csv("ml.csv")
df.head()
```

Out[3]:

	Items	Order No.	Order Type	Grand Total (?)	Invoice No.	Payment Type	Ratings
0	Aloo Cheese Burger	159.0	Dine In	210.0	670369.0	Cash	5.0
1	Laziz Special Pizza (Large)	158.0	Delivery	370.0	764617.0	Online	7.0
2	Carnival Pizza (Medium), Cheese Garlic Bread, ...	157.0	Dine In	2320.0	993592.0	Cash	6.0
3	Corn Delight Pizza (Medium), Tomato & Corn Piz...	156.0	Delivery	310.0	358473.0	Online	9.0
4	Laziz Special Pizza (Small)	155.0	Dine In	160.0	849054.0	Cash	8.0

In [4]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score,classification_report,confusion_matr
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

In [5]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score,classification_report,confusion_matr
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
import plotly.express as px
import seaborn as sns
import plotly.graph_objects as go
```

In [21]:
```
x=df.iloc[:,[1,3,4]].values
y=df.iloc[:,[5]]
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=1/3,random_state=0
```

In [7]:
```
sns.heatmap(df.corr(),annot=True,linecolor="red")
```

Out[7]: <AxesSubplot:>

In [8]:
```
px.density_heatmap(df,x="Order No.",y="Order Type")
```

In [19]:
```
from sklearn.preprocessing import LabelEncoder
labenc=LabelEncoder()
df['Items']=labenc.fit_transform(df['Items'])
df['Order Type']=labenc.fit_transform(df['Order Type'])
df['Grand Total (?)']=labenc.fit_transform(df['Grand Total (?)'])
df['Payment Type']=labenc.fit_transform(df['Payment Type'])
df['Ratings']=labenc.fit_transform(df['Ratings'])
```

In [20]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=1/3,random_state=0
```

In [14]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 103 entries, 0 to 102
Data columns (total 7 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Items            103 non-null    int64
 1   Order No.        103 non-null    float64
 2   Order Type       103 non-null    int64
 3   Grand Total (?)  103 non-null    int64
 4   Invoice No.      103 non-null    float64
 5   Payment Type     103 non-null    int64
 6   Ratings          103 non-null    int64
dtypes: float64(2), int64(5)
memory usage: 5.8 KB
```

In [22]:
```
#using Logistic Regression
from sklearn.linear_model import LogisticRegression
model_1=LogisticRegression()
#fitting of model
model_1.fit(x_train,y_train)
#prediction
y_pred_1=model_1.predict(x_test)
#accuracy
accuracy_1=accuracy_score(y_test,y_pred_1)
print("The accuracy score for Logistic Regression is : ",format(accuracy_1*100)
```

```
The accuracy score for Logistic Regression is :  65.71428571428571
```

In [23]:
```
#using RandomForestClassifier
from sklearn.ensemble import RandomForestClassifier
model_3=RandomForestClassifier()
#fitting of model
model_3.fit(x_train,y_train)
#prediction
y_pred_3=model_3.predict(x_test)
#accuracy
accuracy_3=accuracy_score(y_test,y_pred_3)
print("The accuracy score for Reandom Forest is : ",format(accuracy_3*100))
```

```
The accuracy score for Reandom Forest is :  65.71428571428571
```

In [24]:
```
#using Naive Bayes Classifier
from sklearn.naive_bayes import BernoulliNB
model_6=BernoulliNB()
#fitting of model
model_6.fit(x_train,y_train)
#prediction
y_pred_6=model_6.predict(x_test)
#accuracy
accuracy_6=accuracy_score(y_test,y_pred_6)
print("The accuracy score for Naive Bayes Classifier is : ",format(accuracy_6*1
```

```
The accuracy score for Naive Bayes Classifier is :  65.71428571428571
```

In [25]:
```
from sklearn.svm import SVC #'Support vector classifier'
model=SVC(kernel='linear',random_state=1)
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred))
from sklearn.metrics import accuracy_score,confusion_matrix,classification_repo
acc=accuracy_score(y_pred,y_test)*100
acc
```

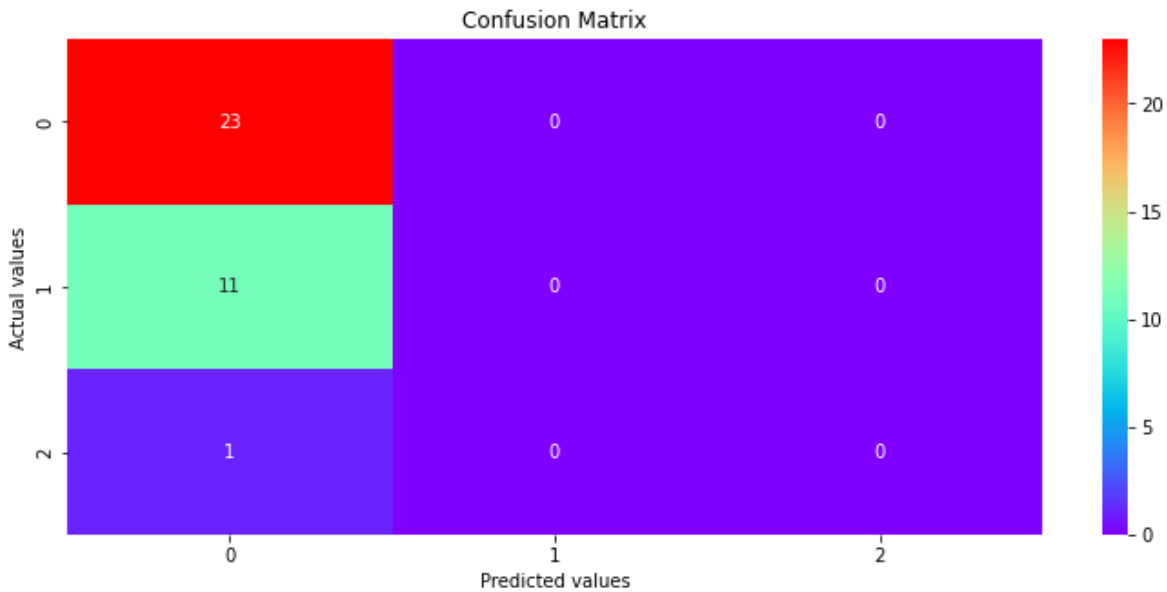
```
              precision    recall  f1-score   support

           0       0.68      1.00      0.81        23
           1       1.00      0.09      0.17        11
           2       0.00      0.00      0.00         1

    accuracy                           0.69        35
   macro avg       0.56      0.36      0.32        35
weighted avg       0.76      0.69      0.58        35
```

Out[25]: 68.57142857142857

In [26]:
```
cm=confusion_matrix(y_test,y_pred_1)
print(cm)
```

```
[[23  0  0]
 [11  0  0]
 [ 1  0  0]]
```

In [27]:
```
plt.figure(figsize=(12,5))
plt.title("Confusion Matrix")
sns.heatmap(cm, annot=True, fmt='d', cmap="rainbow")
plt.ylabel("Actual values")
plt.xlabel("Predicted values")
plt.savefig('Confusion_matrix.png')
```



In [28]:
```
new_x=pd.read_csv("nml.csv")
new_x
type(new_x)
df1=np.array(new_x)
```

In [34]:
```
#Logistic Regression
new_x=pd.read_csv("nml.csv")
new_x
type(new_x)
df1=np.array(new_x)
y_pred_1=model_1.predict(df1)
y_pred_1
```

Out[34]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0], dtype=int64)
```

In [36]:
```
#Random Forest
new_x=pd.read_csv("nml.csv")
new_x
type(new_x)
df1=np.array(new_x)
y_pred_3=model_3.predict(df1)
y_pred_3
```

Out[36]:
```
array([1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1], dtype=int64)
```

In [38]:
```
#SVM
new_x=pd.read_csv("nml.csv")
new_x
type(new_x)
df1=np.array(new_x)
y_pred=model.predict(df1)
y_pred
```

Out[38]:
```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1], dtype=int64)
```

In [ ]: