Importing the required libraries

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
```

Reading the dataset

In [2]:
```
walmart_data = pd.read_csv('/Users/ashleshad/Downloads/Walmart Sales.csv')
```

In [3]:
```
walmart_data.head(6)
```

Out[3]:

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Date	Time	Payment	Rating
0	750-67-8428	A	Yangon	Member	Female	Health and beauty	74.69	7	1/5/2019	13:08	Ewallet	9.1
1	226-31-3081	A	Naypyitaw	Normal	Female	Electronic accessories	15.28	5	3/8/2019	10:29	Cash	9.6
2	631-41-3108	A	Yangon	Normal	Male	Home and lifestyle	46.33	7	3/3/2019	13:23	Credit card	7.4
3	123-19-1176	B	Yangon	Member	Male	Health and beauty	58.22	8	1/27/2019	20:33	Ewallet	8.4
4	373-73-7910	C	Yangon	Normal	Male	Sports and travel	86.31	7	2/8/2019	10:37	Ewallet	5.3
5	699-14-3026	B	Naypyitaw	Normal	Male	Electronic accessories	85.39	7	3/25/2019	18:30	Ewallet	4.1

Summary Statistics

In [4]:
```
walmart_data.describe()
```

Out[4]:

	Unit price	Quantity	Rating
count	1000.000000	1000.000000	1000.00000
mean	55.672130	5.510000	6.97270
std	26.494628	2.923431	1.71858
min	10.080000	1.000000	4.00000
25%	32.875000	3.000000	5.50000
50%	55.230000	5.000000	7.00000
75%	77.935000	8.000000	8.50000
max	99.960000	10.000000	10.00000

In [5]:
```
walmart_data.skew()
```

```
/var/folders/_j/tzw6wdvd1fv_1s_66tcw6my40000gn/T/ipykernel_1194/3942749770.py:1: FutureWarning: The default
value of numeric_only in DataFrame.skew is deprecated. In a future version, it will default to False. In ad
dition, specifying 'numeric_only=None' is deprecated. Select only valid columns or specify the value of num
eric_only to silence this warning.
  walmart_data.skew()
```

Out[5]:
```
Unit price    0.007077
Quantity      0.012941
Rating        0.009010
dtype: float64
```

Checking missing values

In [6]:
```
walmart_data.isnull().sum()
```

Out[6]:
```
Invoice ID       0
Branch           0
City             0
Customer type    0
Gender           0
Product line     0
Unit price       0
Quantity         0
Date             0
Time             0
Payment          0
Rating           0
dtype: int64
```

Categorical Columns to Numerical

In [7]:
```
label_encoder = LabelEncoder()
walmart_data['Customer type'] = label_encoder.fit_transform(walmart_data['Customer type'])
walmart_data['Gender'] = label_encoder.fit_transform(walmart_data['Gender'])
walmart_data['Product line'] = label_encoder.fit_transform(walmart_data['Product line'])
walmart_data['Payment'] = label_encoder.fit_transform(walmart_data['Payment'])
```

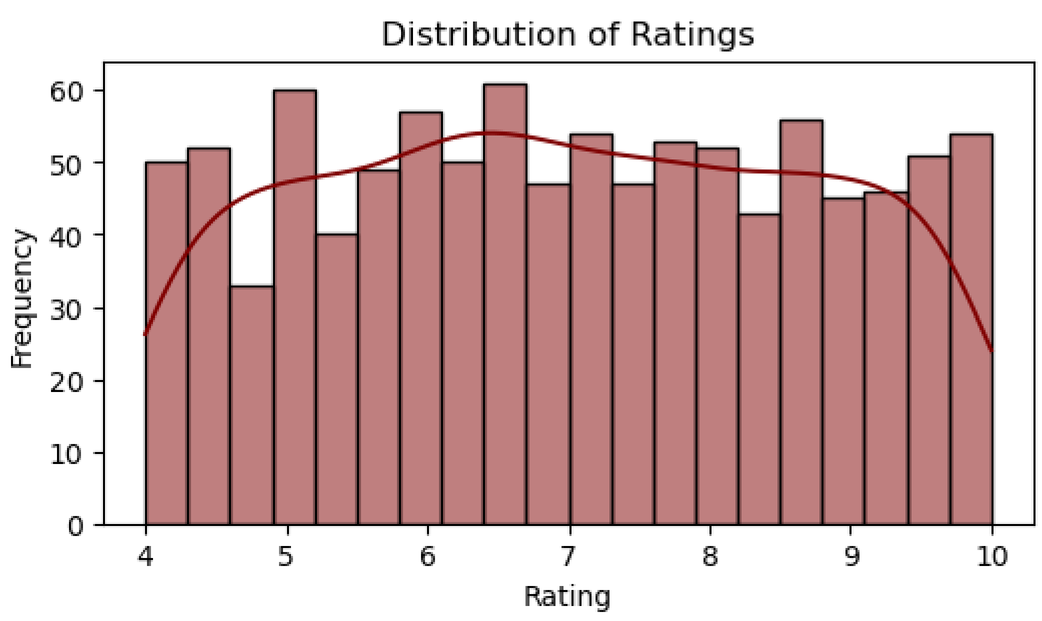
Checking the Data Types

In [8]:
```
walmart_data.dtypes
```

Out[8]:
```
Invoice ID        object
Branch            object
City              object
Customer type      int64
Gender             int64
Product line       int64
Unit price       float64
Quantity           int64
Date              object
Time              object
Payment            int64
Rating           float64
dtype: object
```

Exploratory Data Analysis

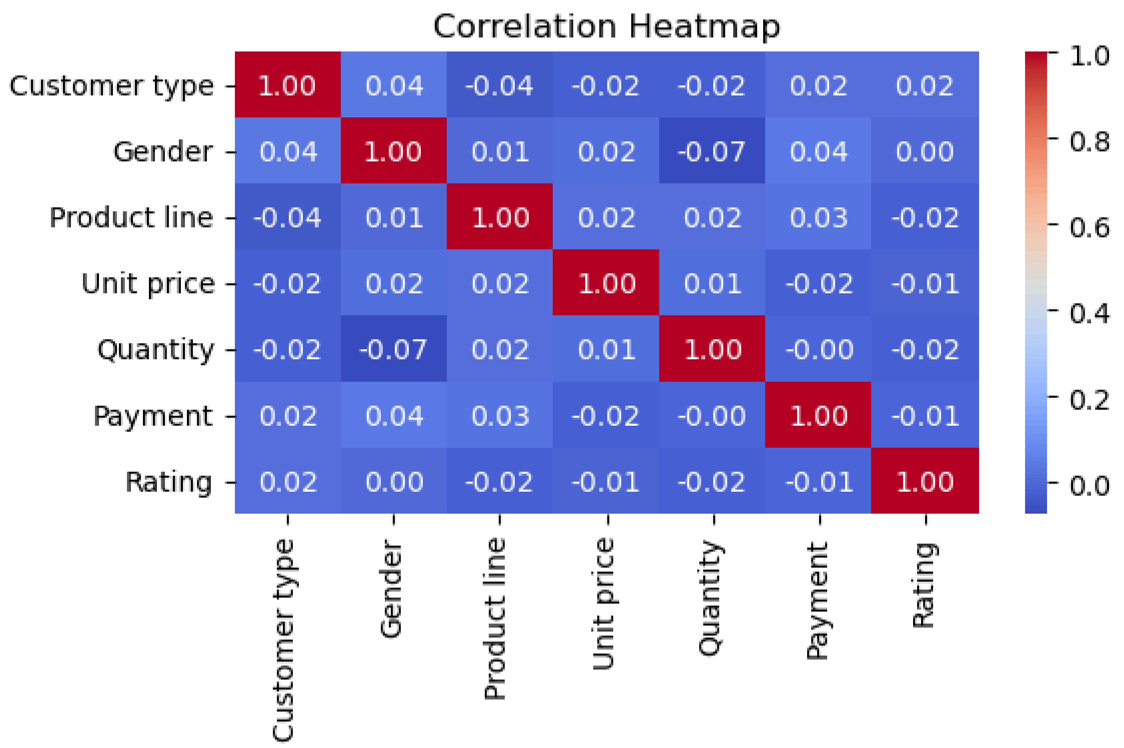
In [9]:
```
plt.figure(figsize=(6, 3))
sns.histplot(walmart_data['Rating'], bins=20, kde=True, color='maroon')
plt.title('Distribution of Ratings')
plt.xlabel('Rating')
plt.ylabel('Frequency')
plt.show()
```



Correlation Heatmap

In [10]:
```
plt.figure(figsize=(6, 3))
corr = walmart_data.corr()
sns.heatmap(corr, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Heatmap')
plt.show()
```

```
/var/folders/_j/tzw6wdvd1fv_1s_66tcw6my40000gn/T/ipykernel_1194/2436359786.py:2: FutureWarning: The default
value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Selec
t only valid columns or specify the value of numeric_only to silence this warning.
  corr = walmart_data.corr()
```



In [11]:
```
# Convert 'Invoice ID' to string type
walmart_data['Invoice ID'] = walmart_data['Invoice ID'].astype(str)

# Remove '-' and convert to integer
walmart_data['Invoice ID'] = walmart_data['Invoice ID'].str.replace('-', '').astype(int)
```

Dropping unnecessary columns

In [12]:
```
walmart_data = walmart_data.drop(['Branch', 'Date', 'Time', 'City'], axis=1)
```

In [13]:
```
walmart_data
```

Out[13]:

	Invoice ID	Customer type	Gender	Product line	Unit price	Quantity	Payment	Rating
0	750678428	0	0	3	74.69	7	2	9.1
1	226313081	1	0	0	15.28	5	0	9.6
2	631413108	1	1	4	46.33	7	1	7.4
3	123191176	0	1	3	58.22	8	2	8.4
4	373737910	1	1	5	86.31	7	2	5.3
...	...	...	...	...	...	...	...	...
995	233675758	1	1	3	40.35	1	2	6.2
996	303962227	1	0	4	97.38	10	2	4.4
997	727021313	0	1	2	31.84	1	0	7.7
998	347562442	1	1	4	65.82	1	0	4.1
999	849093807	0	0	1	88.34	7	0	6.6

1000 rows × 8 columns

Defining Target Variable and dependent features

In [14]:
```
X = walmart_data.drop('Rating', axis=1)
y = walmart_data['Rating']
```

Splitting dataset into training and testing

In [15]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=44)
```

Applying Machine Learning Algorithms

RANDOM FOREST REGRESSOR MODEL

In [16]:
```
RFR = RandomForestRegressor(n_estimators=100, random_state=42)
RFR.fit(X_train, y_train)
```

Out[16]:
```
▾ RandomForestRegressor
RandomForestRegressor(random_state=42)
```

In [17]:
```
y_pred = RFR.predict(X_test)
```

In [18]:
```
mse = mean_squared_error(y_test, y_pred)
```

In [19]:
```
mse
```

Out[19]:
```
3.164777029999999
```

In [20]:
```
mae = mean_absolute_error(y_test, y_pred)
```

In [21]:
```
mae
```

Out[21]:
```
1.5031999999999994
```

In [22]:
```
r2 = r2_score(y_test, y_pred)
```

In [23]:
```
r2
```

Out[23]:
```
-0.14289053226338222
```

LINEAR REGRESSION MODEL

In [24]:
```
LR = LinearRegression()
```

In [25]:
```
LR.fit(X_train, y_train)
```

Out[25]:
```
▾ LinearRegression
LinearRegression()
```

In [26]:
```
y_pred = LR.predict(X_test)
```

In [27]:
```
lr_mse = mean_squared_error(y_test, y_pred)
```

In [28]:
```
lr_mse
```

Out[28]:
```
2.878762209157371
```

In [29]:
```
lr_mae = mean_absolute_error(y_test, y_pred)
```

In [30]:
```
lr_mae
```

Out[30]:
```
1.438864703666802
```

In [31]:
```
lr_r2 = r2_score(y_test, y_pred)
```

In [32]:
```
lr_r2
```

Out[32]:
```
-0.039602487725202806
```

Decision Tree Regressor

In [33]:
```
DTR = DecisionTreeRegressor()
```

In [34]:
```
DTR.fit(X_train, y_train)
```
Out[34]:
```
▾ DecisionTreeRegressor
DecisionTreeRegressor()
```
In [35]:
```
y_pred = DTR.predict(X_test)
```
In [36]:
```
dtr_mse = mean_squared_error(y_test, y_pred)
```
In [37]:
```
dtr_mse
```
Out[37]:
```
6.087249999999999
```
In [38]:
```
dtr_mae = mean_absolute_error(y_test, y_pred)
```
In [39]:
```
dtr_mae
```
Out[39]:
```
2.0075
```
In [40]:
```
dtr_r2 = r2_score(y_test, y_pred)
```
In [41]:
```
dtr_r2
```
Out[41]:
```
-1.1982782125160565
```

Summary

LOGISTIC REGRESSION SEEMS TO BE THE BEST AMONGST THE OTHER TWO FOR PREDICTING THE CUSTOMER RATINGS.

In [ ]:
```

```