Sales of a supermarket

- Dataset info : Sales data of supermarket

Importing Libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings as warning
warning.filterwarnings("ignore")
```

Read the data

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

EDA - Exploratory Data Analysis

In [3]:
```
df.head()
```

Out[3]:

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	Date	Time	Payment	cogs	gross margin percentage	gross income	Rati
0	750-67-8428	A	Yangon	Member	Female	Health and beauty	74.69	7	26.1415	548.9715	1/5/2019	13:08	Ewallet	522.83	4.761905	26.1415	
1	226-31-3081	C	Naypyitaw	Normal	Female	Electronic accessories	15.28	5	3.8200	80.2200	3/8/2019	10:29	Cash	76.40	4.761905	3.8200	
2	631-41-3108	A	Yangon	Normal	Male	Home and lifestyle	46.33	7	16.2155	340.5255	3/3/2019	13:23	Credit card	324.31	4.761905	16.2155	
3	123-19-1176	A	Yangon	Member	Male	Health and beauty	58.22	8	23.2880	489.0480	1/27/2019	20:33	Ewallet	465.76	4.761905	23.2880	
4	373-73-7910	A	Yangon	Normal	Male	Sports and travel	86.31	7	30.2085	634.3785	2/8/2019	10:37	Ewallet	604.17	4.761905	30.2085	

In [4]:
```
df.tail()
```

Out[4]:

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	Date	Time	Payment	cogs	gross margin percentage	gross income	R
995	233-67-5758	C	Naypyitaw	Normal	Male	Health and beauty	40.35	1	2.0175	42.3675	1/29/2019	13:46	Ewallet	40.35	4.761905	2.0175	
996	303-96-2227	B	Mandalay	Normal	Female	Home and lifestyle	97.38	10	48.6900	1022.4900	3/2/2019	17:16	Ewallet	973.80	4.761905	48.6900	
997	727-02-1313	A	Yangon	Member	Male	Food and beverages	31.84	1	1.5920	33.4320	2/9/2019	13:22	Cash	31.84	4.761905	1.5920	
998	347-56-2442	A	Yangon	Normal	Male	Home and lifestyle	65.82	1	3.2910	69.1110	2/22/2019	15:33	Cash	65.82	4.761905	3.2910	
999	849-09-3807	A	Yangon	Member	Female	Fashion accessories	88.34	7	30.9190	649.2990	2/18/2019	13:28	Cash	618.38	4.761905	30.9190	

Checking various attributes of Data

In [5]:
```
df.shape
```

Out[5]:
```
(1000, 17)
```

In [6]:
```
df.columns
```

Out[6]:
```
Index(['Invoice ID', 'Branch', 'City', 'Customer type', 'Gender',
       'Product line', 'Unit price', 'Quantity', 'Tax 5%', 'Total', 'Date',
       'Time', 'Payment', 'cogs', 'gross margin percentage', 'gross income',
       'Rating'],
      dtype='object')
```

In [7]:
```
df.columns.value_counts()
```

Out[7]:
```
Invoice ID                1
Total                     1
gross income              1
gross margin percentage   1
cogs                      1
Payment                   1
Time                      1
Date                      1
Tax 5%                    1
Branch                    1
Quantity                  1
Unit price                1
Product line              1
Gender                    1
Customer type             1
City                      1
Rating                    1
Name: count, dtype: int64
```

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

Out[8]:
```
Invoice ID                 object
Branch                     object
City                       object
Customer type              object
Gender                     object
Product line               object
Unit price                float64
Quantity                    int64
Tax 5%                    float64
Total                     float64
Date                       object
Time                       object
Payment                    object
cogs                      float64
gross margin percentage   float64
gross income              float64
Rating                    float64
dtype: object
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 17 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Invoice ID              1000 non-null   object
 1   Branch                  1000 non-null   object
 2   City                    1000 non-null   object
 3   Customer type           1000 non-null   object
 4   Gender                  1000 non-null   object
 5   Product line            1000 non-null   object
 6   Unit price              1000 non-null   float64
 7   Quantity                1000 non-null   int64
 8   Tax 5%                  1000 non-null   float64
 9   Total                   1000 non-null   float64
 10  Date                    1000 non-null   object
 11  Time                    1000 non-null   object
 12  Payment                 1000 non-null   object
 13  cogs                    1000 non-null   float64
 14  gross margin percentage 1000 non-null   float64
 15  gross income            1000 non-null   float64
 16  Rating                  1000 non-null   float64
dtypes: float64(7), int64(1), object(9)
memory usage: 132.9+ KB
```

- From the above info, we can see that we do not have any null or missing values in our data and our data is clean

In [10]:
```
df.isnull().sum()
```

Out[10]:
```
Invoice ID                 0
Branch                     0
City                       0
Customer type              0
Gender                     0
Product line               0
Unit price                 0
Quantity                   0
Tax 5%                     0
Total                      0
Date                       0
Time                       0
Payment                    0
cogs                       0
gross margin percentage    0
gross income               0
Rating                     0
dtype: int64
```

- This confirms that there are no null values present in our data

In [11]:
```
df.describe()
```

Out[11]:

	Unit price	Quantity	Tax 5%	Total	cogs	gross margin percentage	gross income	Rating
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.00000	1.000000e+03	1000.000000	1000.00000
mean	55.672130	5.510000	15.379369	322.966749	307.58738	4.761905e+00	15.379369	6.97270
std	26.494628	2.923431	11.708825	245.885335	234.17651	6.131498e-14	11.708825	1.71858
min	10.080000	1.000000	0.508500	10.678500	10.17000	4.761905e+00	0.508500	4.00000
25%	32.875000	3.000000	5.924875	124.422375	118.49750	4.761905e+00	5.924875	5.50000
50%	55.230000	5.000000	12.088000	253.848000	241.76000	4.761905e+00	12.088000	7.00000
75%	77.935000	8.000000	22.445250	471.350250	448.90500	4.761905e+00	22.445250	8.50000
max	99.960000	10.000000	49.650000	1042.650000	993.00000	4.761905e+00	49.650000	10.00000

In [12]:
```
categoricals = ["Branch", "City", "Customer type", "Gender", "Product line",
                "Payment"]
numericals = ["Unit price", "Quantity", "Tax 5%", "gross margin percentage", "gross income",
              "Total", "Rating"]
```

- By doing this we can know that which columns are categorical and which are numericals

In [13]:
```
month = ["January", "February", "March",
         "April", "May", "June", "July",
         "August", "September", "October", "November",
         "December"]

weekday = ["Monday", "Tuesday", "Wednesday", "Thursday",
           "Friday", "Saturday", "Sunday"]
def convert_date(x):
    date = datetime.strptime(x, "%m/%d/%Y")
    return [month[date.month-1], date.day, weekday[date.isoweekday()-1]]
```

- We have differentiated months and weekdays from the date so that we can perform better analysis and we have made a function for it so we can simply call it later when we have to use it

In [14]:
```
def plot(df, name, num, axes):
    grouped = df.groupby(name)
    mean = grouped[num].mean()
    sns.barplot(x=mean.index, y=mean, ax=axes)
    for container in axes.containers:
        axes.bar_label(container, rotation=90, label_type="center")
    axes.set_xticklabels(axes.get_xticklabels(), rotation=90)
```

- Here we have created a function so that while plotting we can simply call this function and do not have to mention those things all over again

Converting Dates

In [15]:
```
import datetime
```

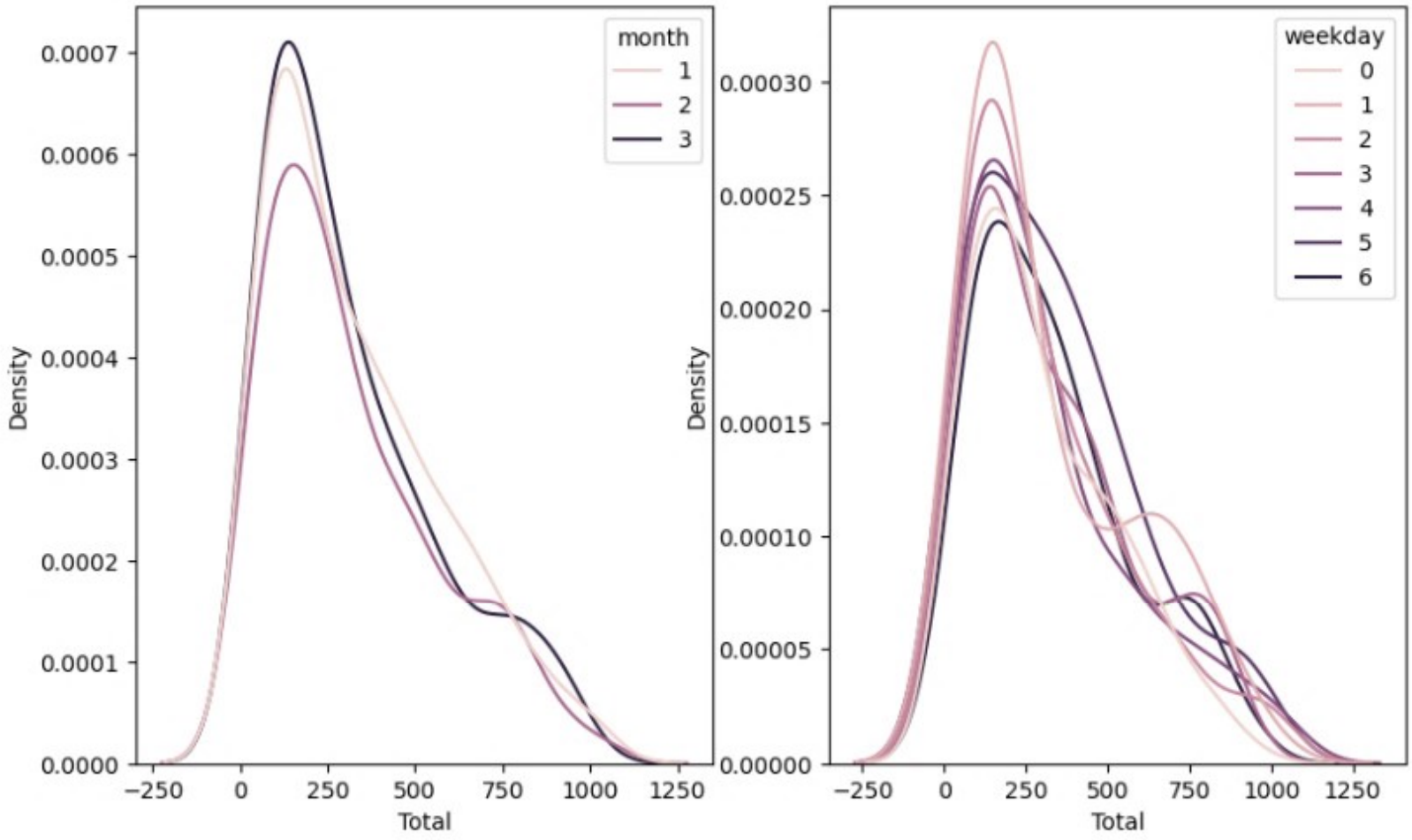
In [16]:
```
df["Date"] = pd.to_datetime(df["Date"])

df["month"] = df["Date"].dt.month
df["day"] = df["Date"].dt.day
df["weekday"] = df["Date"].dt.weekday
df["Hour"] = pd.to_datetime(df["Time"]).dt.hour
```

- Here we have imported datetime and then converted our 'Date' column to datetime datatype as it was object datatype before
- Next we have created four new columns which are month,day,weekday,and hour so that we can perform our analysis accordingly

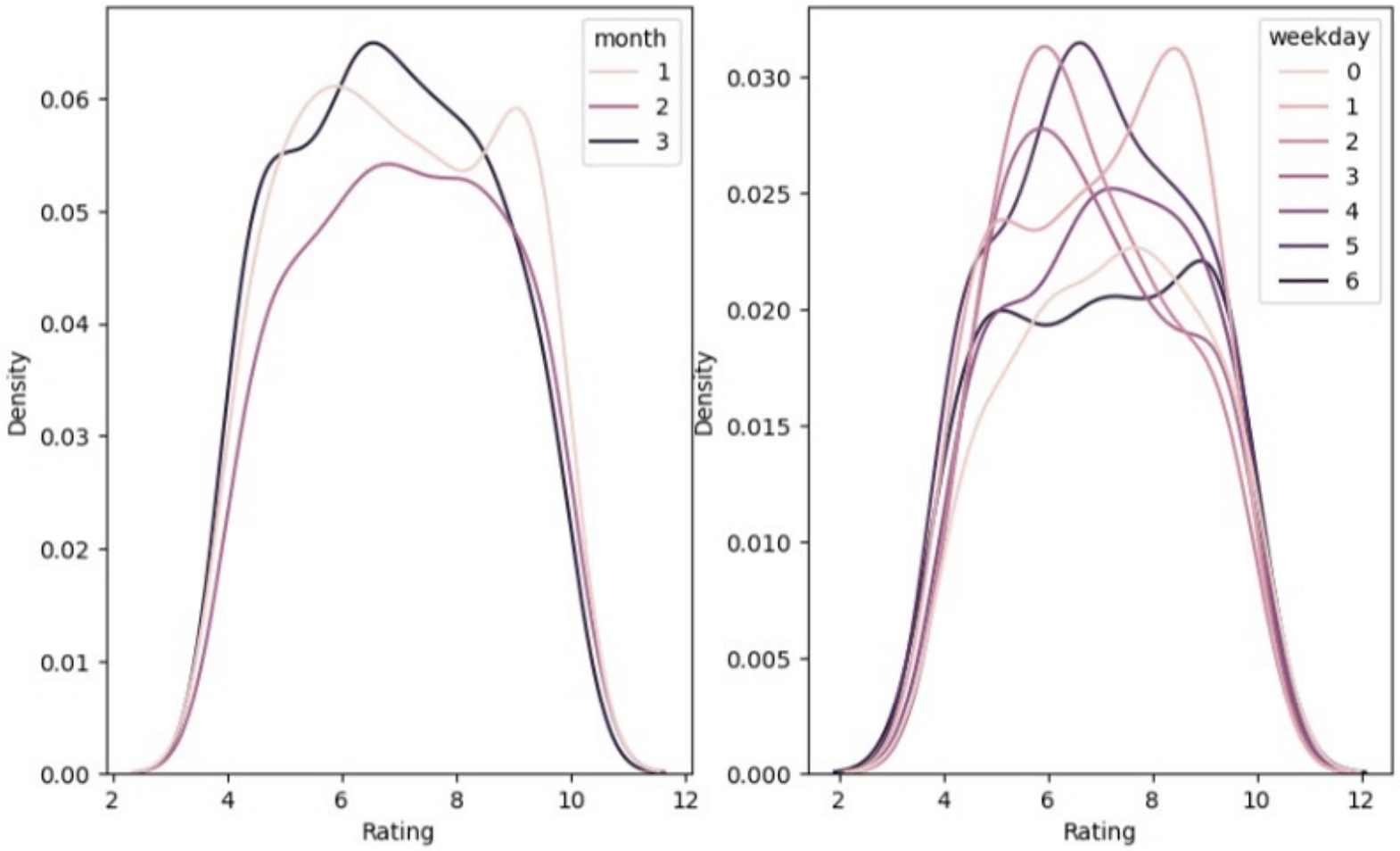
Total paid and ratings distributed by dates

In [17]:
```
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(10, 6))
sns.kdeplot(df, x="Total", hue="month", ax=axes[0])
sns.kdeplot(df, x="Total", hue="weekday", ax=axes[1])
plt.show()
```



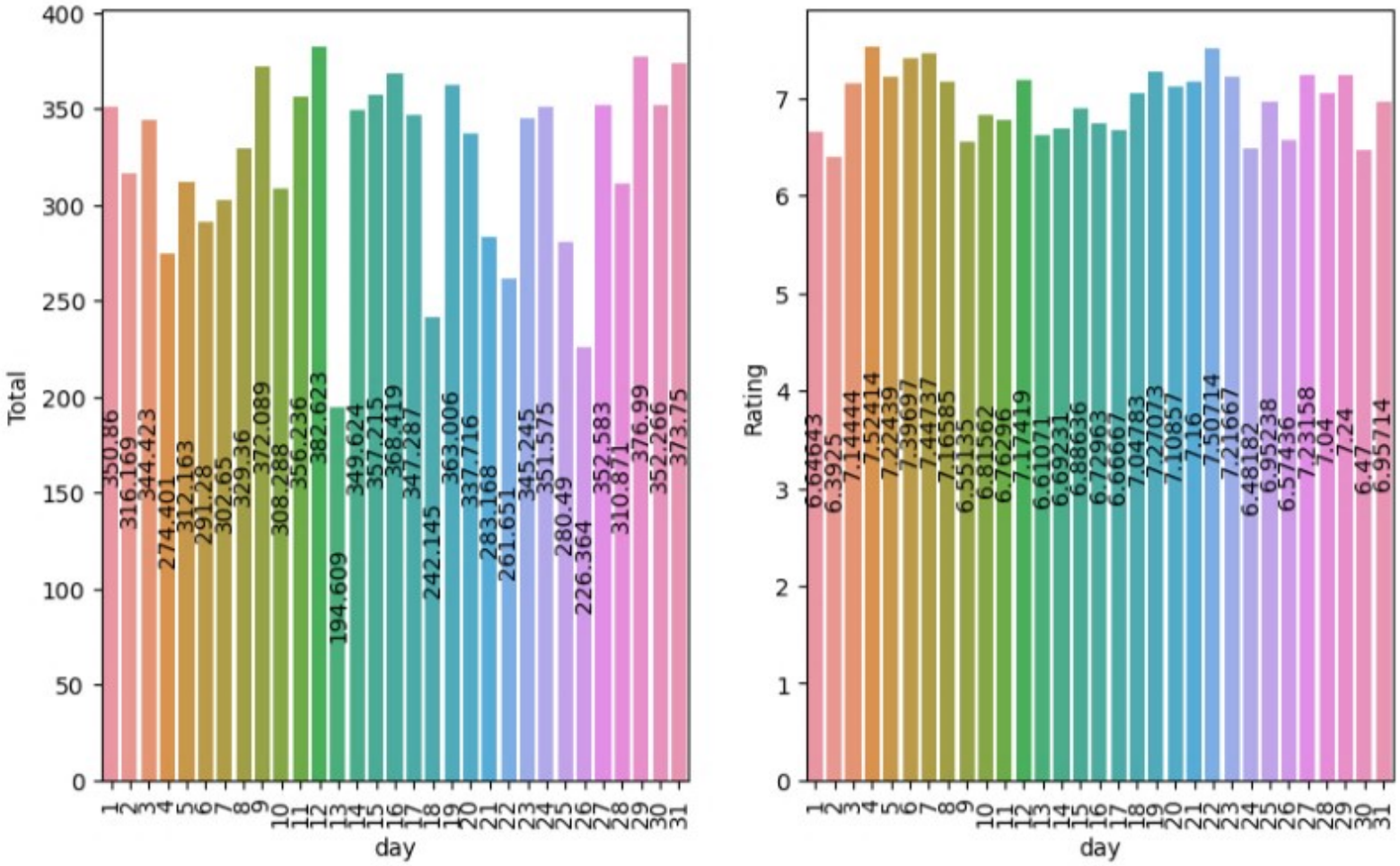
- From the above plot we can understand that how much is the total density according to month and weekday

In [18]:
```
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(10, 6))
sns.kdeplot(df, x="Rating", hue="month", ax=axes[0])
sns.kdeplot(df, x="Rating", hue="weekday", ax=axes[1])
plt.show()
```



- Here we can see the ratings according to the month and weekday
- We can see that the 3rd month rating is highest at rating 7
- In weekday, we can observe that 5th weekday is highest at rating 7

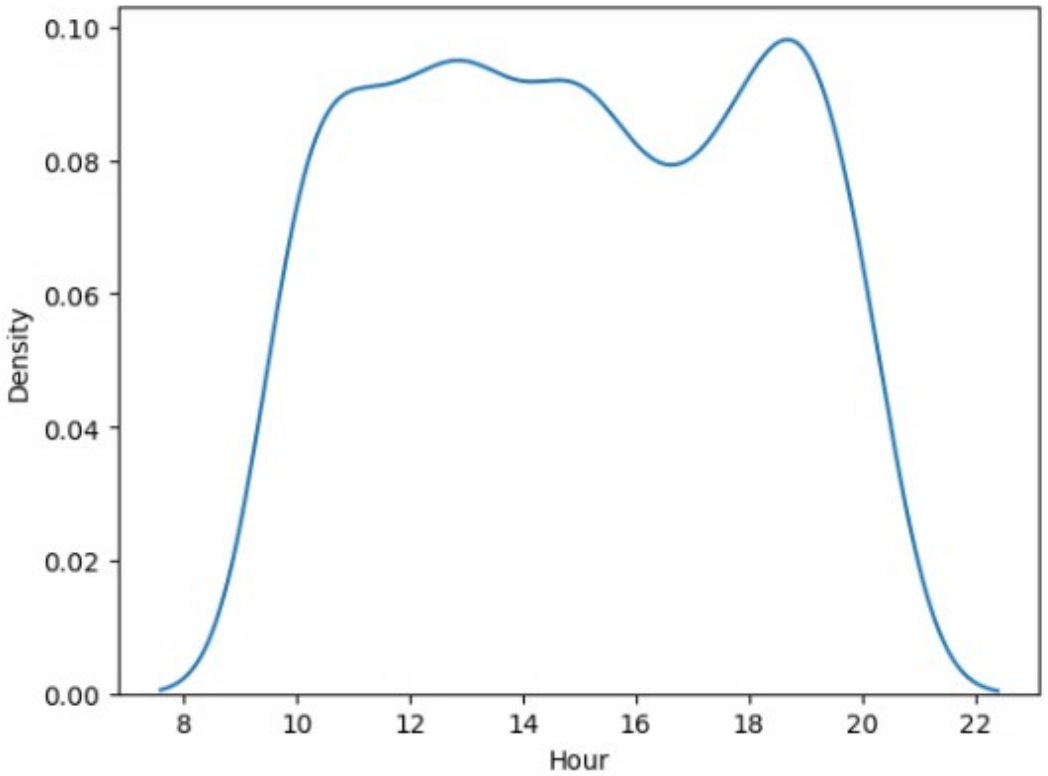
In [19]:
```
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(10, 6))
plot(df, "day", "Total", axes[0])
plot(df, "day", "Rating", axes[1])
plt.show()
```



- In this we have a comparison between total by day and rating by day which gives us an idea on the fluctuations

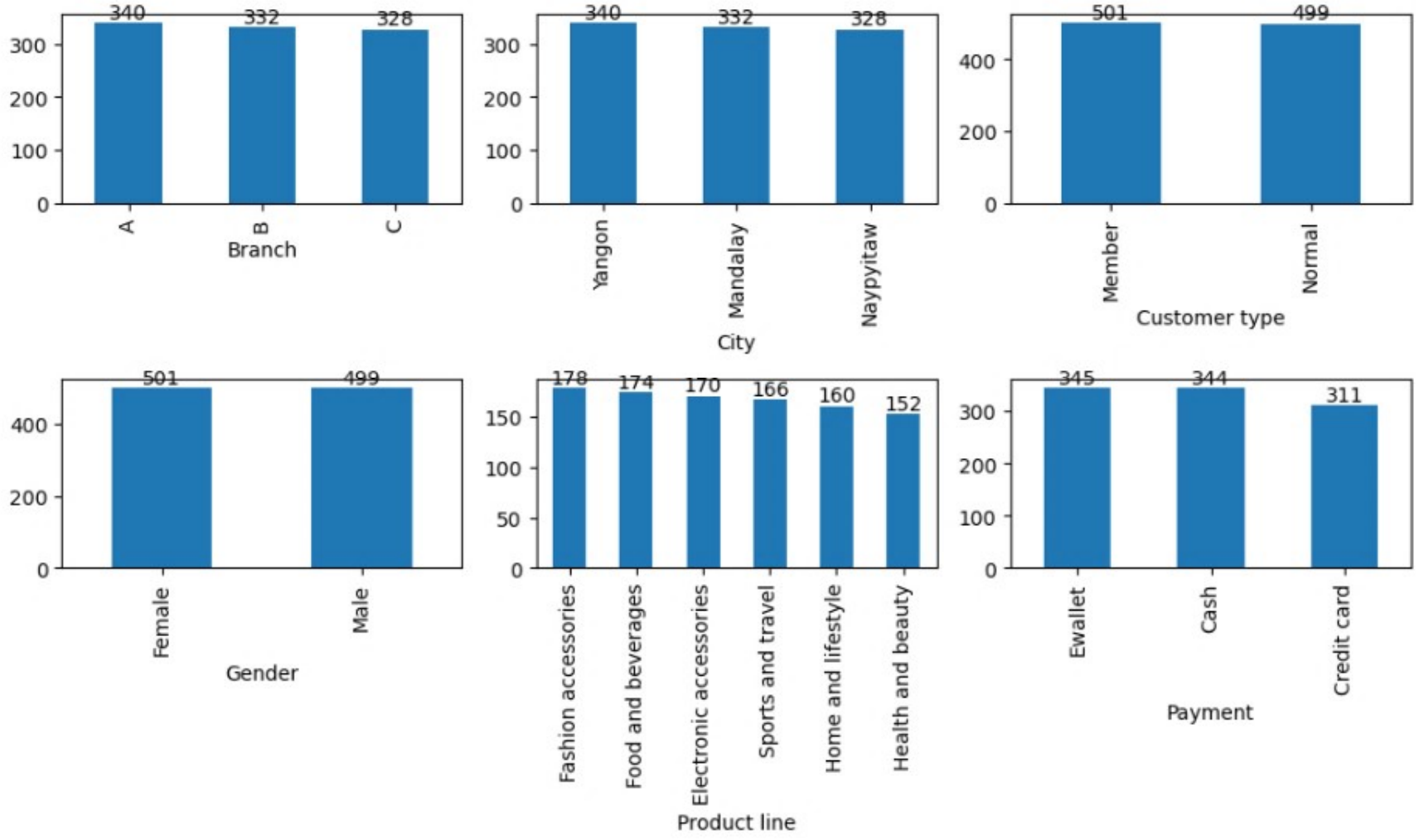
Different Purchase Hours density

In [20]:
```
sns.kdeplot(df, x="Hour")
plt.show()
```



Categoricals Count

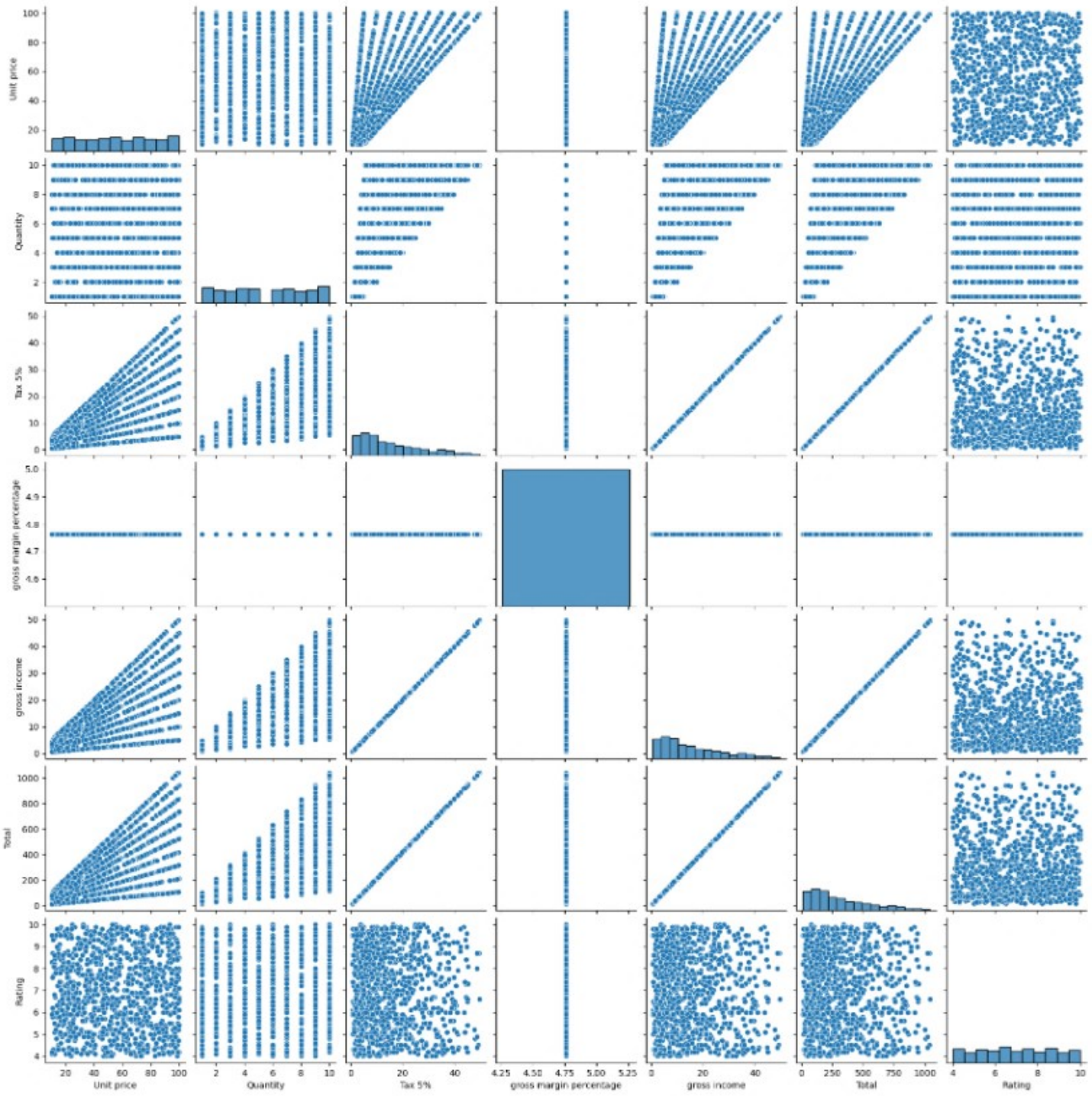
In [21]:
```
fig, axes = plt.subplots(nrows=2, ncols=3, figsize=(10, 6))
index = 0
for i in range(2):
    for j in range(3):
        df[categoricals[index]].value_counts().plot(kind="bar", ax=axes[i][j])
        index += 1
        for container in axes[i][j].containers:
            axes[i][j].bar_label(container)
plt.tight_layout()
plt.show()
```



- In the above charts we can see the counts according to the different categories which helps us majorly to understand the data easily

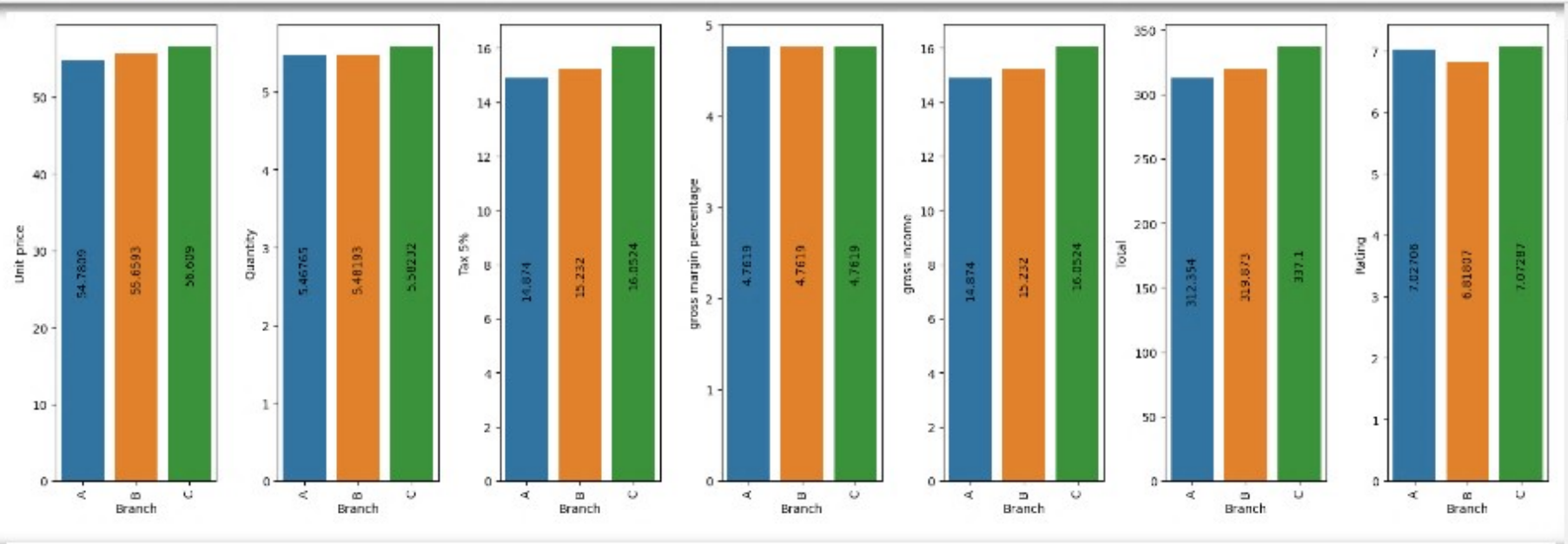
Numerical Data distribution

In [22]:
```
sns.pairplot(df, vars=numericals)
plt.show()
```



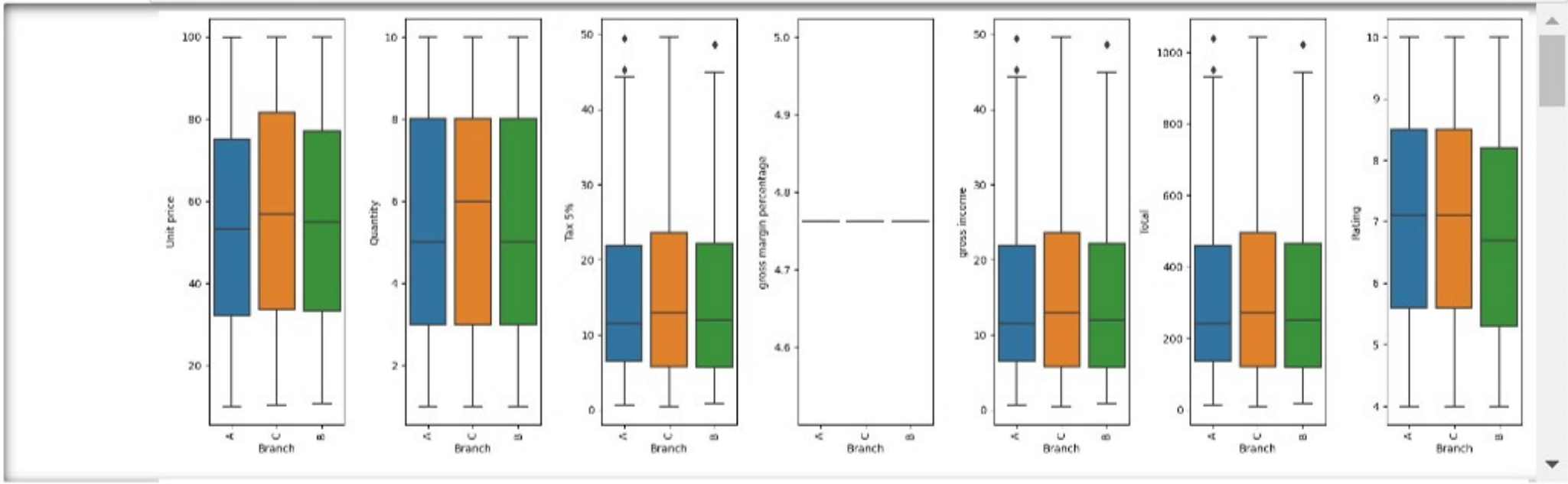
Mean values for each category

In [23]:
```
for k in categoricals:
    fig, axes = plt.subplots(nrows=1, ncols=7, figsize=(18, 6))
    for i, j in enumerate(numericals):
        plot(df, k, j, axes[i])
    plt.tight_layout()
    plt.show()
```



Numerical outliers on boxplots distributed among categories

In [24]:
```
for k in categoricals:
    fig, axes = plt.subplots(nrows=1, ncols=7, figsize=(18, 6))
    for i, j in enumerate(numericals):
        sns.boxplot(df, y=j, x=k, ax=axes[i])
        axes[i].set_xticklabels(axes[i].get_xticklabels(), rotation=90)
    plt.tight_layout()
    plt.show()
```



Model Building

In [25]:
```
import pandas as pd
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.metrics import recall_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.tree import DecisionTreeClassifier
```

In [26]:
```
df
```

Out[26]:

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	...	Time	Payment	cogs	gross margin percentage	gross income	Rating
0	750-67-8428	A	Yangon	Member	Female	Health and beauty	74.69	7	26.1415	548.9715	...	13:08	Ewallet	522.83	4.761905	26.1415	9.1
1	226-31-3081	C	Naypyitaw	Normal	Female	Electronic accessories	15.28	5	3.8200	80.2200	...	10:29	Cash	76.40	4.761905	3.8200	9.6
2	631-41-3108	A	Yangon	Normal	Male	Home and lifestyle	46.33	7	16.2155	340.5255	...	13:23	Credit card	324.31	4.761905	16.2155	7.4
3	123-19-1176	A	Yangon	Member	Male	Health and beauty	58.22	8	23.2880	489.0480	...	20:33	Ewallet	465.76	4.761905	23.2880	8.4
4	373-73-7910	A	Yangon	Normal	Male	Sports and travel	86.31	7	30.2085	634.3785	...	10:37	Ewallet	604.17	4.761905	30.2085	5.3
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
995	233-67-5758	C	Naypyitaw	Normal	Male	Health and beauty	40.35	1	2.0175	42.3675	...	13:46	Ewallet	40.35	4.761905	2.0175	6.2
996	303-96-2227	B	Mandalay	Normal	Female	Home and lifestyle	97.38	10	48.6900	1022.4900	...	17:16	Ewallet	973.80	4.761905	48.6900	4.4
997	727-02-1313	A	Yangon	Member	Male	Food and beverages	31.84	1	1.5920	33.4320	...	13:22	Cash	31.84	4.761905	1.5920	7.7
998	347-56-2442	A	Yangon	Normal	Male	Home and lifestyle	65.82	1	3.2910	69.1110	...	15:33	Cash	65.82	4.761905	3.2910	4.1
999	849-09-3807	A	Yangon	Member	Female	Fashion accessories	88.34	7	30.9190	649.2990	...	13:28	Cash	618.38	4.761905	30.9190	6.6

1000 rows × 21 columns

In [27]:
```
df = df.drop(categoricals, axis=1)
df
```

Out[27]:

	Invoice ID	Unit price	Quantity	Tax 5%	Total	Date	Time	cogs	gross margin percentage	gross income	Rating	month	day	weekday	Hour
0	750-67-8428	74.69	7	26.1415	548.9715	2019-01-05	13:08	522.83	4.761905	26.1415	9.1	1	5	5	13
1	226-31-3081	15.28	5	3.8200	80.2200	2019-03-08	10:29	76.40	4.761905	3.8200	9.6	3	8	4	10
2	631-41-3108	46.33	7	16.2155	340.5255	2019-03-03	13:23	324.31	4.761905	16.2155	7.4	3	3	6	13
3	123-19-1176	58.22	8	23.2880	489.0480	2019-01-27	20:33	465.76	4.761905	23.2880	8.4	1	27	6	20
4	373-73-7910	86.31	7	30.2085	634.3785	2019-02-08	10:37	604.17	4.761905	30.2085	5.3	2	8	4	10
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
995	233-67-5758	40.35	1	2.0175	42.3675	2019-01-29	13:46	40.35	4.761905	2.0175	6.2	1	29	1	13
996	303-96-2227	97.38	10	48.6900	1022.4900	2019-03-02	17:16	973.80	4.761905	48.6900	4.4	3	2	5	17
997	727-02-1313	31.84	1	1.5920	33.4320	2019-02-09	13:22	31.84	4.761905	1.5920	7.7	2	9	5	13
998	347-56-2442	65.82	1	3.2910	69.1110	2019-02-22	15:33	65.82	4.761905	3.2910	4.1	2	22	4	15
999	849-09-3807	88.34	7	30.9190	649.2990	2019-02-18	13:28	618.38	4.761905	30.9190	6.6	2	18	0	13

1000 rows × 15 columns

In [28]:
```
df = df.drop(columns = ['Invoice ID','Date','Time'])
df
```

Out[28]:

	Unit price	Quantity	Tax 5%	Total	cogs	gross margin percentage	gross income	Rating	month	day	weekday	Hour
0	74.69	7	26.1415	548.9715	522.83	4.761905	26.1415	9.1	1	5	5	13
1	15.28	5	3.8200	80.2200	76.40	4.761905	3.8200	9.6	3	8	4	10
2	46.33	7	16.2155	340.5255	324.31	4.761905	16.2155	7.4	3	3	6	13
3	58.22	8	23.2880	489.0480	465.76	4.761905	23.2880	8.4	1	27	6	20
4	86.31	7	30.2085	634.3785	604.17	4.761905	30.2085	5.3	2	8	4	10
...	...	...	...	...	...	...	...	...	...	...	...	...
995	40.35	1	2.0175	42.3675	40.35	4.761905	2.0175	6.2	1	29	1	13
996	97.38	10	48.6900	1022.4900	973.80	4.761905	48.6900	4.4	3	2	5	17
997	31.84	1	1.5920	33.4320	31.84	4.761905	1.5920	7.7	2	9	5	13
998	65.82	1	3.2910	69.1110	65.82	4.761905	3.2910	4.1	2	22	4	15
999	88.34	7	30.9190	649.2990	618.38	4.761905	30.9190	6.6	2	18	0	13

1000 rows × 12 columns

In [29]:
```
df['Recommended'] = np.random.randint(2, size=len(df))
df
```

Out[29]:

	Unit price	Quantity	Tax 5%	Total	cogs	gross margin percentage	gross income	Rating	month	day	weekday	Hour	Recommended
0	74.69	7	26.1415	548.9715	522.83	4.761905	26.1415	9.1	1	5	5	13	0
1	15.28	5	3.8200	80.2200	76.40	4.761905	3.8200	9.6	3	8	4	10	1
2	46.33	7	16.2155	340.5255	324.31	4.761905	16.2155	7.4	3	3	6	13	1
3	58.22	8	23.2880	489.0480	465.76	4.761905	23.2880	8.4	1	27	6	20	0
4	86.31	7	30.2085	634.3785	604.17	4.761905	30.2085	5.3	2	8	4	10	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...
995	40.35	1	2.0175	42.3675	40.35	4.761905	2.0175	6.2	1	29	1	13	1
996	97.38	10	48.6900	1022.4900	973.80	4.761905	48.6900	4.4	3	2	5	17	1
997	31.84	1	1.5920	33.4320	31.84	4.761905	1.5920	7.7	2	9	5	13	0
998	65.82	1	3.2910	69.1110	65.82	4.761905	3.2910	4.1	2	22	4	15	1
999	88.34	7	30.9190	649.2990	618.38	4.761905	30.9190	6.6	2	18	0	13	1

1000 rows × 13 columns

In [30]:
```
x = df.drop('Recommended', axis = 1)
x
```

Out[30]:

	Unit price	Quantity	Tax 5%	Total	cogs	gross margin percentage	gross income	Rating	month	day	weekday	Hour
0	74.69	7	26.1415	548.9715	522.83	4.761905	26.1415	9.1	1	5	5	13
1	15.28	5	3.8200	80.2200	76.40	4.761905	3.8200	9.6	3	8	4	10
2	46.33	7	16.2155	340.5255	324.31	4.761905	16.2155	7.4	3	3	6	13
3	58.22	8	23.2880	489.0480	465.76	4.761905	23.2880	8.4	1	27	6	20
4	86.31	7	30.2085	634.3785	604.17	4.761905	30.2085	5.3	2	8	4	10
...	...	...	...	...	...	...	...	...	...	...	...	...
995	40.35	1	2.0175	42.3675	40.35	4.761905	2.0175	6.2	1	29	1	13
996	97.38	10	48.6900	1022.4900	973.80	4.761905	48.6900	4.4	3	2	5	17
997	31.84	1	1.5920	33.4320	31.84	4.761905	1.5920	7.7	2	9	5	13
998	65.82	1	3.2910	69.1110	65.82	4.761905	3.2910	4.1	2	22	4	15
999	88.34	7	30.9190	649.2990	618.38	4.761905	30.9190	6.6	2	18	0	13

1000 rows × 12 columns

In [31]:
```
y = df['Recommended']
y
```

Out[31]:
```
0      0
1      1
2      1
3      0
4      1
      ..
995    1
996    1
997    0
998    1
999    1
Name: Recommended, Length: 1000, dtype: int32
```

- Train Test Split

In [32]:
```
xtrain,xtest,ytrain,ytest = train_test_split (x,y, test_size = 0.25)
```

- **Decision Tree Classifier**

In [33]:
```
dt_clf = DecisionTreeClassifier(criterion = 'gini', random_state = 100, max_depth = 6, min_samples_leaf = 8)
```

In [34]:
```
dt_clf.fit(xtrain,ytrain)
```

Out[34]:
```
DecisionTreeClassifier
DecisionTreeClassifier(max_depth=6, min_samples_leaf=8, random_state=100)
```

In [35]:
```
ypred = dt_clf.predict(xtest)
ypred
```

Out[35]:
```
array([1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0,
       1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0,
       0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1,
       0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1,
       1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0,
       1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1,
       0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0,
       0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1,
       1, 1, 1, 1, 1, 1, 0, 1])
```

In [36]:
```
dt_clf.score(xtest,ytest)
```

Out[36]:
```
0.492
```

In [37]:
```
print(classification_report(ytest,ypred, labels =[0,1]))
```

```
              precision    recall  f1-score   support

           0       0.48      0.31      0.37       124
           1       0.50      0.67      0.57       126

    accuracy                           0.49       250
   macro avg       0.49      0.49      0.47       250
weighted avg       0.49      0.49      0.47       250
```

In [38]:
```
!pip uninstall scikit-learn --yes
```

```
Found existing installation: scikit-learn 1.2.2
Uninstalling scikit-learn-1.2.2:
  Successfully uninstalled scikit-learn-1.2.2
```

```
ERROR: Exception:
Traceback (most recent call last):
  File "C:\ProgramData\anaconda3\Lib\site-packages\pip\_internal\cli\base_command.py", line 180, in exc_logging_wrapper
    status = run_func(*args)
             ^^^^^^^^^^^^^^^
  File "C:\ProgramData\anaconda3\Lib\site-packages\pip\_internal\commands\uninstall.py", line 110, in run
    uninstall_pathset.commit()
  File "C:\ProgramData\anaconda3\Lib\site-packages\pip\_internal\req\req_uninstall.py", line 432, in commit
    self._moved_paths.commit()
  File "C:\ProgramData\anaconda3\Lib\site-packages\pip\_internal\req\req_uninstall.py", line 278, in commit
    save_dir.cleanup()
  File "C:\ProgramData\anaconda3\Lib\site-packages\pip\_internal\utils\temp_dir.py", line 173, in cleanup
    rmtree(self._path)
  File "C:\ProgramData\anaconda3\Lib\site-packages\pip\_vendor\tenacity\__init__.py", line 291, in wrapped_f
    return self(f, *args, **kw)
           ^^^^^^^^^^^^^^^^^^^^
  File "C:\ProgramData\anaconda3\Lib\site-packages\pip\_vendor\tenacity\__init__.py", line 381, in __call__
    do = self.iter(retry_state=retry_state)
         ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\ProgramData\anaconda3\Lib\site-packages\pip\_vendor\tenacity\__init__.py", line 327, in iter
    raise retry_exc.reraise()
          ^^^^^^^^^^^^^^^^^^^
  File "C:\ProgramData\anaconda3\Lib\site-packages\pip\_vendor\tenacity\__init__.py", line 160, in reraise
    raise self.last_attempt.result()
          ^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\ProgramData\anaconda3\Lib\concurrent\futures\_base.py", line 449, in result
    return self.__get_result()
           ^^^^^^^^^^^^^^^^^^^
  File "C:\ProgramData\anaconda3\Lib\concurrent\futures\_base.py", line 401, in __get_result
    raise self._exception
  File "C:\ProgramData\anaconda3\Lib\site-packages\pip\_vendor\tenacity\__init__.py", line 384, in __call__
    result = fn(*args, **kwargs)
             ^^^^^^^^^^^^^^^^^^^
  File "C:\ProgramData\anaconda3\Lib\site-packages\pip\_internal\utils\misc.py", line 130, in rmtree
    shutil.rmtree(dir, ignore_errors=ignore_errors, onerror=rmtree_errorhandler)
  File "C:\ProgramData\anaconda3\Lib\shutil.py", line 759, in rmtree
    return _rmtree_unsafe(path, onerror)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\ProgramData\anaconda3\Lib\shutil.py", line 617, in _rmtree_unsafe
    _rmtree_unsafe(fullname, onerror)
  File "C:\ProgramData\anaconda3\Lib\shutil.py", line 622, in _rmtree_unsafe
    onerror(os.unlink, fullname, sys.exc_info())
  File "C:\ProgramData\anaconda3\Lib\shutil.py", line 620, in _rmtree_unsafe
    os.unlink(fullname)
PermissionError: [WinError 5] Access is denied: 'C:\\Users\\husai\\AppData\\Roaming\\Python\\Python311\\site-packages\\~1learn
\\.libs\\msvcp140.dll'
```

In [39]:
```
!pip uninstall imblearn --yes
```

```
Found existing installation: imblearn 0.0
Uninstalling imblearn-0.0:
  Successfully uninstalled imblearn-0.0
```

In [40]:
```
!pip install scikit-learn==1.2.2
```

```
Defaulting to user installation because normal site-packages is not writeable
Collecting scikit-learn==1.2.2
  Obtaining dependency information for scikit-learn==1.2.2 from https://files.pythonhosted.org/packages/db/98/169b46a84b48f92df
2b5e163fce75d471f4df933f8b3d925a61133210776/scikit_learn-1.2.2-cp311-cp311-win_amd64.whl.metadata
  Using cached scikit_learn-1.2.2-cp311-cp311-win_amd64.whl.metadata (11 kB)
Requirement already satisfied: numpy>=1.17.3 in c:\programdata\anaconda3\lib\site-packages (from scikit-learn==1.2.2) (1.24.3)
Requirement already satisfied: scipy>=1.3.2 in c:\programdata\anaconda3\lib\site-packages (from scikit-learn==1.2.2) (1.11.1)
Requirement already satisfied: joblib>=1.1.1 in c:\programdata\anaconda3\lib\site-packages (from scikit-learn==1.2.2) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\programdata\anaconda3\lib\site-packages (from scikit-learn==1.2.2)
(2.2.0)
Using cached scikit_learn-1.2.2-cp311-cp311-win_amd64.whl (8.3 MB)
Installing collected packages: scikit-learn
Successfully installed scikit-learn-1.2.2
```

In [41]:
```
!pip install imblearn
```

```
Defaulting to user installation because normal site-packages is not writeable
Collecting imblearn
  Obtaining dependency information for imblearn from https://files.pythonhosted.org/packages/81/a7/4179e6ebfd654bd0eac0b9c06125
b8b4c96a9d0a8ff9e9507eb2a26d2d7e/imblearn-0.0-py2.py3-none-any.whl.metadata
  Using cached imblearn-0.0-py2.py3-none-any.whl.metadata (355 bytes)
Requirement already satisfied: imbalanced-learn in c:\programdata\anaconda3\lib\site-packages (from imblearn) (0.10.1)
Requirement already satisfied: numpy>=1.17.3 in c:\programdata\anaconda3\lib\site-packages (from imbalanced-learn->imblearn)
(1.24.3)
Requirement already satisfied: scipy>=1.3.2 in c:\programdata\anaconda3\lib\site-packages (from imbalanced-learn->imblearn) (1.
11.1)
Requirement already satisfied: scikit-learn>=1.0.2 in c:\users\husai\appdata\roaming\python\python311\site-packages (from imbal
anced-learn->imblearn) (1.2.2)
Requirement already satisfied: joblib>=1.1.1 in c:\programdata\anaconda3\lib\site-packages (from imbalanced-learn->imblearn)
(1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\programdata\anaconda3\lib\site-packages (from imbalanced-learn->imble
arn) (2.2.0)
Using cached imblearn-0.0-py2.py3-none-any.whl (1.9 kB)
Installing collected packages: imblearn
Successfully installed imblearn-0.0
```

In [42]:
```
from imblearn.combine import SMOTEENN
```

In [43]:
```
sm = SMOTEENN(random_state = 42)
x_resampled, y_resampled = sm.fit_resample(x,y)
```

In [44]:
```
xr_train,xr_test,yr_train,yr_test = train_test_split(x_resampled,y_resampled, test_size = 0.25)
```

In [45]:
```
dt_sm = DecisionTreeClassifier(criterion = 'gini', random_state = 100, max_depth = 6, min_samples_leaf = 8)
```

In [46]:
```
dt_sm.fit(xr_train,yr_train)
yr_predict = dt_sm.predict(xr_test)
score_r = dt_sm.score(xr_test,yr_test)
print(score_r)
print(metrics.classification_report(yr_test, yr_predict))
```

```
0.6875
              precision    recall  f1-score   support

           0       0.73      0.70      0.72        27
           1       0.64      0.67      0.65        21

    accuracy                           0.69        48
   macro avg       0.68      0.69      0.68        48
weighted avg       0.69      0.69      0.69        48
```

In [47]:
```
print(metrics.confusion_matrix(yr_test, yr_predict))
```

```
[[19  8]
 [ 7 14]]
```

- **Random Forest Classifier**

In [48]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [49]:
```
rf_clf = RandomForestClassifier(n_estimators = 100, criterion = 'gini', random_state = 100, max_depth = 6, min_samples_leaf = 8)
```

In [50]:
```
rf_clf.fit(xtrain,ytrain)
```

Out[50]:
```
RandomForestClassifier(max_depth=6, min_samples_leaf=8, random_state=100)
```

In [51]:
```
ypred = rf_clf.predict(xtest)
```

In [52]:
```
rf_clf.score(xtest,ytest)
```

Out[52]:
```
0.54
```

In [53]:
```
print(classification_report(ytest,ypred,labels = [0,1]))
```

```
              precision    recall  f1-score   support

           0       0.55      0.39      0.45       124
           1       0.53      0.69      0.60       126

    accuracy                           0.54       250
   macro avg       0.54      0.54      0.53       250
weighted avg       0.54      0.54      0.53       250
```

In [54]:
```
sm1 = SMOTEENN()
x_resampled1, y_resampled1 = sm1.fit_resample(x,y)
```

In [55]:
```
xr_train1,xr_test1,yr_train1,yr_test1 = train_test_split(x_resampled1,y_resampled1,test_size = 0.25)
```

In [56]:
```
rf_sm = RandomForestClassifier(n_estimators = 100, criterion = 'gini', random_state = 100, max_depth = 6, min_samples_leaf = 8)
```

In [57]:
```
rf_sm.fit(xr_train1,yr_train1)
```

Out[57]:
```
RandomForestClassifier(max_depth=6, min_samples_leaf=8, random_state=100)
```

In [58]:
```
yr_pred1 = rf_sm.predict(xr_test1)
```

In [59]:
```
score_r1 = rf_sm.score(xr_test1,yr_test1)
```

In [60]:
```
print(score_r1)
print(metrics.classification_report(yr_test1,yr_pred1))
```

```
0.7021276595744681
              precision    recall  f1-score   support

           0       0.68      0.74      0.71        23
           1       0.73      0.67      0.70        24

    accuracy                           0.70        47
   macro avg       0.70      0.70      0.70        47
weighted avg       0.70      0.70      0.70        47
```

In [61]:
```
print(metrics.confusion_matrix(yr_test1,yr_pred1))
```

```
[[17  6]
 [ 8 16]]
```

- **Logistic Regression**

In [62]:
```
from sklearn.linear_model import LogisticRegression
```

In [63]:
```
lg = LogisticRegression()
```

In [64]:
```
lg.fit(xtrain,ytrain)
```

Out[64]:
```
▾ LogisticRegression
LogisticRegression()
```

In [65]:
```
ypred = lg.predict(xtest)
```

In [66]:
```
lg.score(xtest,ytest)
```

Out[66]:
```
0.544
```

In [67]:
```
print(classification_report(ytest,ypred,labels = [0,1]))
```

```
              precision    recall  f1-score   support

           0       0.57      0.35      0.43       124
           1       0.53      0.74      0.62       126

    accuracy                           0.54       250
   macro avg       0.55      0.54      0.53       250
weighted avg       0.55      0.54      0.53       250
```

In [68]:
```
sm2 = SMOTEENN()
x_resampled2, y_resampled2 = sm2.fit_resample(x,y)
```

In [69]:
```
xr_train2,xr_test2,yr_train2,yr_test2 = train_test_split(x_resampled2,y_resampled2,test_size = 0.25)
```

In [70]:
```
lg_sm = LogisticRegression()
```

In [71]:
```
lg_sm.fit(xr_train2,yr_train2)
```

Out[71]:
```
▾ LogisticRegression
LogisticRegression()
```

In [72]:
```
yr_pred2 = lg_sm.predict(xr_test2)
```

In [73]:
```
score_r2 = lg_sm.score(xr_test2,yr_test2)
```

In [74]:
```
print(score_r2)
print(metrics.classification_report(yr_test2,yr_pred2))
```

```
0.5306122448979592
              precision    recall  f1-score   support

           0       0.67      0.52      0.58        31
           1       0.40      0.56      0.47        18

    accuracy                           0.53        49
   macro avg       0.53      0.54      0.52        49
weighted avg       0.57      0.53      0.54        49
```

In [75]:
```
print(metrics.confusion_matrix(yr_test2,yr_pred2))
```

```
[[16 15]
 [ 8 10]]
```

- **SVM**

In [76]:
```
from sklearn.svm import SVC
```

In [77]:
```
sv = SVC()
```

In [78]:
```
sv.fit(xtrain,ytrain)
```

Out[78]:
```
SVC()
```

In [79]:
```
ypred = sv.predict(xtest)
```

In [80]:
```
sv.score(xtest,ytest)
```

Out[80]:
```
0.552
```

In [81]:
```
print(classification_report(ytest,ypred,labels = [0,1]))
```

```
              precision    recall  f1-score   support

           0       0.68      0.19      0.29       124
           1       0.53      0.91      0.67       126

    accuracy                           0.55       250
   macro avg       0.60      0.55      0.48       250
weighted avg       0.60      0.55      0.48       250
```

In [82]:
```
sm3 = SMOTEENN()
x_resampled3, y_resampled3 = sm3.fit_resample(x,y)
```

In [83]:
```
xr_train3,xr_test3,yr_train3,yr_test3 = train_test_split(x_resampled3,y_resampled3,test_size = 0.25)
```

In [84]:
```
sv_sm = SVC()
```

In [85]:
```
sv_sm.fit(xr_train3,yr_train3)
```

Out[85]:
```
SVC()
```

In [86]:
```
yr_pred3 =sv_sm.predict(xr_test3)
```

In [87]:
```
score_r3 = lg_sm.score(xr_test3,yr_test3)
```

In [88]:
```
print(score_r3)
print(metrics.classification_report(yr_test3,yr_pred3))
```

```
0.62
              precision    recall  f1-score   support

           0       0.68      0.65      0.67        26
           1       0.64      0.67      0.65        24

    accuracy                           0.66        50
   macro avg       0.66      0.66      0.66        50
weighted avg       0.66      0.66      0.66        50
```

In [89]:
```
print(metrics.confusion_matrix(yr_test3,yr_pred3))
```

```
[[17  9]
 [ 8 16]]
```

Creating a Neural Network

ANN

In [90]:
```
pip install --upgrade tensorflow --user
```

```
Requirement already satisfied: tensorflow in c:\users\husai\appdata\roaming\python\python311\site-packages (2.16.1)
Requirement already satisfied: tensorflow-intel==2.16.1 in c:\users\husai\appdata\roaming\python\python311\site-packages (from
tensorflow) (2.16.1)
Requirement already satisfied: absl-py>=1.0.0 in c:\users\husai\appdata\roaming\python\python311\site-packages (from tensorflow
-intel==2.16.1->tensorflow) (2.1.0)
Requirement already satisfied: astunparse>=1.6.0 in c:\users\husai\appdata\roaming\python\python311\site-packages (from tensorf
low-intel==2.16.1->tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=23.5.26 in c:\users\husai\appdata\roaming\python\python311\site-packages (from tens
orflow-intel==2.16.1->tensorflow) (24.3.25)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in c:\users\husai\appdata\roaming\python\python311\site-pack
ages (from tensorflow-intel==2.16.1->tensorflow) (0.5.4)
Requirement already satisfied: google-pasta>=0.1.1 in c:\users\husai\appdata\roaming\python\python311\site-packages (from tenso
rflow-intel==2.16.1->tensorflow) (0.2.0)
Requirement already satisfied: h5py>=3.10.0 in c:\users\husai\appdata\roaming\python\python311\site-packages (from tensorflow-i
ntel==2.16.1->tensorflow) (3.11.0)
Requirement already satisfied: libclang>=13.0.0 in c:\users\husai\appdata\roaming\python\python311\site-packages (from tensorfl
ow-intel==2.16.1->tensorflow) (18.1.1)
Requirement already satisfied: ml-dtypes~=0.3.1 in c:\users\husai\appdata\roaming\python\python311\site-packages (from tensorfl
ow-intel==2.16.1->tensorflow) (0.3.2)
Requirement already satisfied: opt-einsum>=2.3.2 in c:\users\husai\appdata\roaming\python\python311\site-packages (from tensorf
low-intel==2.16.1->tensorflow) (3.3.0)
Requirement already satisfied: packaging in c:\programdata\anaconda3\lib\site-packages (from tensorflow-intel==2.16.1->tensorfl
ow) (23.1)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3 in c:\users\hus
ai\appdata\roaming\python\python311\site-packages (from tensorflow-intel==2.16.1->tensorflow) (4.25.3)
Requirement already satisfied: requests<3,>=2.21.0 in c:\programdata\anaconda3\lib\site-packages (from tensorflow-intel==2.16.1
->tensorflow) (2.31.0)
Requirement already satisfied: setuptools in c:\programdata\anaconda3\lib\site-packages (from tensorflow-intel==2.16.1->tensorf
low) (68.0.0)
Requirement already satisfied: six>=1.12.0 in c:\programdata\anaconda3\lib\site-packages (from tensorflow-intel==2.16.1->tensor
flow) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in c:\users\husai\appdata\roaming\python\python311\site-packages (from tensorfl
ow-intel==2.16.1->tensorflow) (2.4.0)
Requirement already satisfied: typing-extensions>=3.6.6 in c:\programdata\anaconda3\lib\site-packages (from tensorflow-intel==
2.16.1->tensorflow) (4.7.1)
Requirement already satisfied: wrapt>=1.11.0 in c:\programdata\anaconda3\lib\site-packages (from tensorflow-intel==2.16.1->tens
orflow) (1.14.1)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in c:\users\husai\appdata\roaming\python\python311\site-packages (from tenso
rflow-intel==2.16.1->tensorflow) (1.63.0)
Requirement already satisfied: tensorboard<2.17,>=2.16 in c:\users\husai\appdata\roaming\python\python311\site-packages (from t
ensorflow-intel==2.16.1->tensorflow) (2.16.2)
Requirement already satisfied: keras>=3.0.0 in c:\users\husai\appdata\roaming\python\python311\site-packages (from tensorflow-i
ntel==2.16.1->tensorflow) (3.3.2)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in c:\users\husai\appdata\roaming\python\python311\site-pac
kages (from tensorflow-intel==2.16.1->tensorflow) (0.31.0)
Requirement already satisfied: numpy<2.0.0,>=1.23.5 in c:\programdata\anaconda3\lib\site-packages (from tensorflow-intel==2.16.
1->tensorflow) (1.24.3)
Requirement already satisfied: wheel<1.0,>=0.23.0 in c:\programdata\anaconda3\lib\site-packages (from astunparse>=1.6.0->tensor
flow-intel==2.16.1->tensorflow) (0.38.4)
Requirement already satisfied: rich in c:\users\husai\appdata\roaming\python\python311\site-packages (from keras>=3.0.0->tensor
flow-intel==2.16.1->tensorflow) (13.7.1)
```

```
Note: you may need to restart the kernel to use updated packages.
```

In [91]:
```
x.isnull().sum()
```

Out[91]:
```
Unit price                0
Quantity                  0
Tax 5%                    0
Total                     0
cogs                      0
gross margin percentage   0
gross income              0
Rating                    0
month                     0
day                       0
weekday                   0
Hour                      0
dtype: int64
```

In [92]:
```
y.isnull().sum()
```

Out[92]:
```
0
```

In [93]:
```
from sklearn.model_selection import train_test_split
```

In [94]:
```
xtrain,xtest,ytrain,ytest = train_test_split(x,y, test_size = 0.25, random_state = 25)
```

In [95]:
```
from sklearn.preprocessing import StandardScaler
```

In [96]:
```
Scaler = StandardScaler()
xtrain = pd.DataFrame(Scaler.fit_transform(xtrain),columns = xtrain.columns)
xtest = pd.DataFrame(Scaler.transform(xtest),columns = xtest.columns)
```

In [97]:
```
xtrain.head()
```

Out[97]:

	Unit price	Quantity	Tax 5%	Total	cogs	gross margin percentage	gross income	Rating	month	day	weekday	Hour
0	-1.092828	-1.250408	-1.095232	-1.095232	-1.095232	0.0	-1.095232	0.118655	1.200075	-1.499464	1.004120	1.290036
1	1.047375	-0.556764	0.107029	0.107029	0.107029	0.0	0.107029	1.532146	-1.206492	0.568023	1.516775	0.345493
2	-0.347360	-0.903586	-0.729464	-0.729464	-0.729464	0.0	-0.729464	0.118655	-0.003209	0.223442	1.516775	-0.599050
3	-0.335102	-0.903586	-0.725230	-0.725230	-0.725230	0.0	-0.725230	0.413132	1.200075	1.257185	-1.046499	-1.543593
4	0.065675	-1.597229	-1.070084	-1.070084	-1.070084	0.0	-1.070084	1.532146	1.200075	-0.236000	-0.533845	0.030645

In [98]:
```
xtest.head()
```

Out[98]:

	Unit price	Quantity	Tax 5%	Total	cogs	gross margin percentage	gross income	Rating	month	day	weekday	Hour
0	0.674084	-1.250408	-0.688320	-0.688320	-0.688320	0.0	-0.688320	-0.529195	1.200075	-0.236000	-0.533845	-0.913898
1	1.036232	1.524165	2.222343	2.222343	2.222343	0.0	2.222343	0.530923	1.200075	0.453162	-1.046499	0.660341
2	-0.231101	0.483701	0.140604	0.140604	0.140604	0.0	0.140604	-1.059255	-1.206492	1.142324	0.491465	-0.599050
3	0.551511	1.177344	1.366603	1.366603	1.366603	0.0	1.366603	0.472028	-0.003209	0.453162	-1.046499	1.290036
4	-0.960969	-0.209943	-0.694564	-0.694564	-0.694564	0.0	-0.694564	0.589818	-1.206492	0.797743	-1.046499	0.975188

In [99]:
```
xtrain.shape
```

Out[99]: (750, 12)

In [100]:
```
xtest.shape
```

Out[100]: (250, 12)

Building a Neural Network with keras

In [101]:
```
from keras.models import Sequential
from keras.layers import Dense
```

In [102]:
```
NN = Sequential()
NN.add(Dense(64, input_dim = xtrain.shape[1], activation = "relu"))
NN.add(Dense(32, activation = "relu"))
NN.add(Dense(1, activation = "linear"))
```

In [103]:
```
NN.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 64)	832
dense_1 (Dense)	(None, 32)	2,080
dense_2 (Dense)	(None, 1)	33

Total params: 2,945 (11.50 KB)

Trainable params: 2,945 (11.50 KB)

Non-trainable params: 0 (0.00 B)

Loss Function

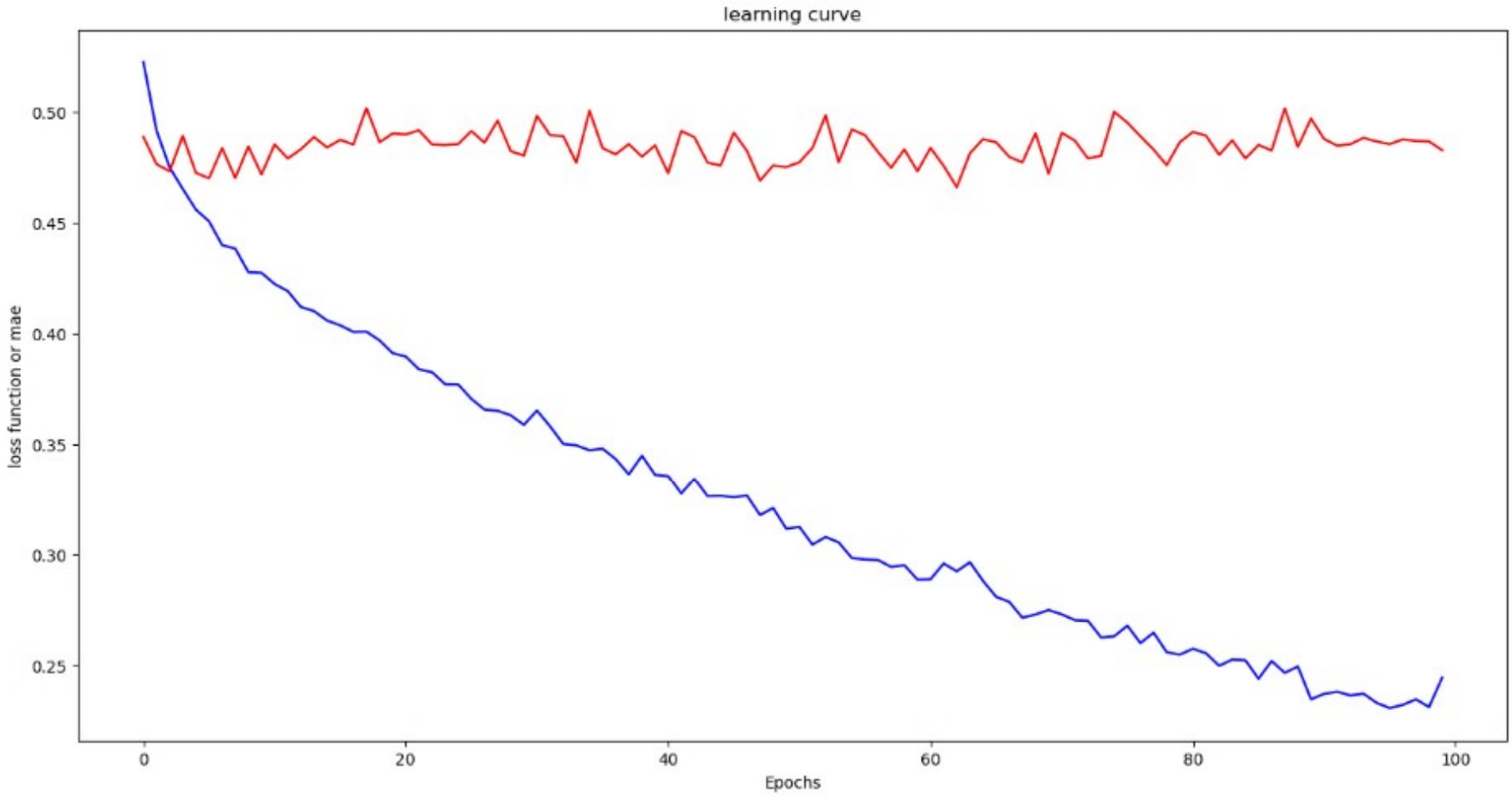
In [104]:
```
NN.compile(loss = "mae", optimizer = "adam")
```

In [105]:
```
hist = NN.fit(xtrain,ytrain, validation_split = 0.2, epochs = 100)
```

```
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.2470 - val_loss: 0.4878
Epoch 92/100
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.2278 - val_loss: 0.4848
Epoch 93/100
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.2268 - val_loss: 0.4855
Epoch 94/100
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.2403 - val_loss: 0.4884
Epoch 95/100
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.2247 - val_loss: 0.4867
Epoch 96/100
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.2282 - val_loss: 0.4855
Epoch 97/100
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.2157 - val_loss: 0.4877
Epoch 98/100
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.2390 - val_loss: 0.4870
Epoch 99/100
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.2097 - val_loss: 0.4867
Epoch 100/100
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.2498 - val_loss: 0.4828
```

Visualization of model training

In [106]:
```
import matplotlib.pyplot as plt
plt.figure(figsize = (16,8))
plt.plot(hist.history["loss"],c = "blue")
plt.plot(hist.history["val_loss"],c = "red")
plt.xlabel("Epochs")
plt.ylabel("loss function or mae")
plt.title("learning curve")
plt.show()
```



Creating Prediction

In [107]:
```
tr_pred = NN.predict(xtrain)
ts_pred = NN.predict(xtest)
```

```
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
8/8 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step
```

In [108]:
```
tr_pred[0:5]
```

Out[108]:
```
array([[0.32074603],
       [1.1318403 ],
       [0.27733427],
       [1.102524  ],
       [1.0392272 ]], dtype=float32)
```

In [109]:
```
ts_pred[0:5]
```

Out[109]:
```
array([[0.86919427],
       [0.23813106],
       [0.58450055],
       [0.24976726],
       [0.69719464]], dtype=float32)
```

In [110]:
```
NN.evaluate(xtrain,ytrain)
```

```
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2354
```

Out[110]:
```
0.27205267548561096
```

MSE for test

In [111]:
```
NN.evaluate(xtest,ytest)
```

```
8/8 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.5276
```

Out[111]:
```
0.5160898566246033
```

In [112]:
```
from sklearn.metrics import mean_absolute_error , mean_squared_error
```

In [113]:
```
tr_mae = mean_absolute_error(ytrain,tr_pred)
tr_mse = mean_squared_error(ytrain,tr_pred)
print("training_mae:",round(tr_mae,2))
print("training_mse:",round(tr_mse,2))
```

```
training_mae: 0.27
training_mse: 0.14
```

In [114]:
```
ts_mae = mean_absolute_error(ytest, ts_pred)
ts_mse = mean_squared_error(ytest, ts_pred)
print("testing mae:",round(ts_mae,2))
print("testing_mse:",round(ts_mse,2))
```

```
testing mae: 0.52
testing_mse: 0.37
```