Sania David

In [57]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, accuracy_score
%matplotlib inline
warnings.filterwarnings('ignore')
sns.set()
```

In [58]:
```
df = pd.read_csv('wine.csv')
df
```

0	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28
1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26
2	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30
3	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24
4	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39
...	...	...	...	...	...	...	...	...
173	13.71	5.65	2.45	20.5	95	1.68	0.61	0.52
174	13.40	3.91	2.48	23.0	102	1.80	0.75	0.43
175	13.27	4.28	2.26	20.0	120	1.59	0.69	0.43
176	13.17	2.59	2.37	20.0	120	1.65	0.68	0.53
177	14.13	4.10	2.74	24.5	96	2.05	0.76	0.56

178 rows × 14 columns

In [59]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 178 entries, 0 to 177
Data columns (total 14 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   Alcohol               178 non-null    float64
 1   Malic_Acid            178 non-null    float64
 2   Ash                   178 non-null    float64
 3   Ash_Alcanity          178 non-null    float64
 4   Magnesium             178 non-null    int64  
 5   Total_Phenols         178 non-null    float64
 6   Flavanoids            178 non-null    float64
 7   Nonflavanoid_Phenols  178 non-null    float64
 8   Proanthocyanins       178 non-null    float64
 9   Color_Intensity       178 non-null    float64
 10  Hue                   178 non-null    float64
 11  OD280                 178 non-null    float64
 12  Proline               178 non-null    int64  
 13  Customer_Segment      178 non-null    int64  
dtypes: float64(11), int64(3)
memory usage: 19.6 KB
```

In [60]:
```
df.describe().round(2)
```

Out[60]:

	Alcohol	Malic_Acid	Ash	Ash_Alcanity	Magnesium	Total_Phenols	Flavanoids	Nonflavanoid_Phenols
count	178.00	178.00	178.00	178.00	178.00	178.00	178.00	178.00
mean	13.00	2.34	2.37	19.49	99.74	2.30	2.03	0.36
std	0.81	1.12	0.27	3.34	14.28	0.63	1.00	0.12
min	11.03	0.74	1.36	10.60	70.00	0.98	0.34	0.13
25%	12.36	1.60	2.21	17.20	88.00	1.74	1.20	0.27
50%	13.05	1.87	2.36	19.50	98.00	2.36	2.13	0.34
75%	13.68	3.08	2.56	21.50	107.00	2.80	2.88	0.44
max	14.83	5.80	3.23	30.00	162.00	3.88	5.08	0.66

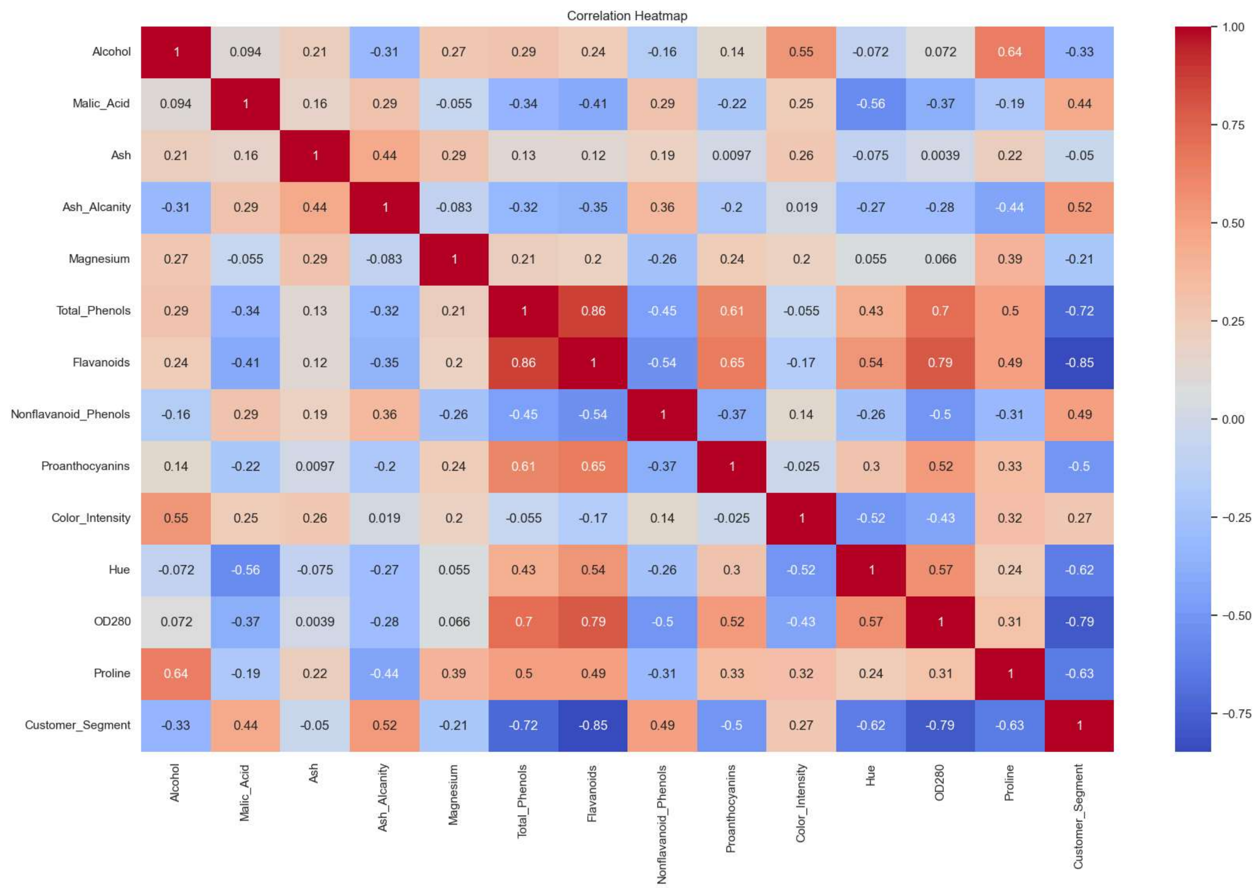
In [61]:
```
df.nunique()
```

Out[61]:
```
Alcohol                 126
Malic_Acid              133
Ash                      79
Ash_Alcanity             63
Magnesium                53
Total_Phenols            97
Flavanoids              132
Nonflavanoid_Phenols     39
Proanthocyanins         101
Color_Intensity         132
Hue                      78
OD280                   122
Proline                 121
Customer_Segment          3
dtype: int64
```

In [62]:
```
df.isnull().sum()
```

Out[62]:
```
Alcohol                 0
Malic_Acid              0
Ash                     0
Ash_Alcanity            0
Magnesium               0
Total_Phenols           0
Flavanoids              0
Nonflavanoid_Phenols    0
Proanthocyanins         0
Color_Intensity         0
Hue                     0
OD280                   0
Proline                 0
Customer_Segment        0
dtype: int64
```

In [63]:
```
correlation_matrix = df.corr()
plt.figure(figsize=(20,12))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```



In [64]:
```
df
```

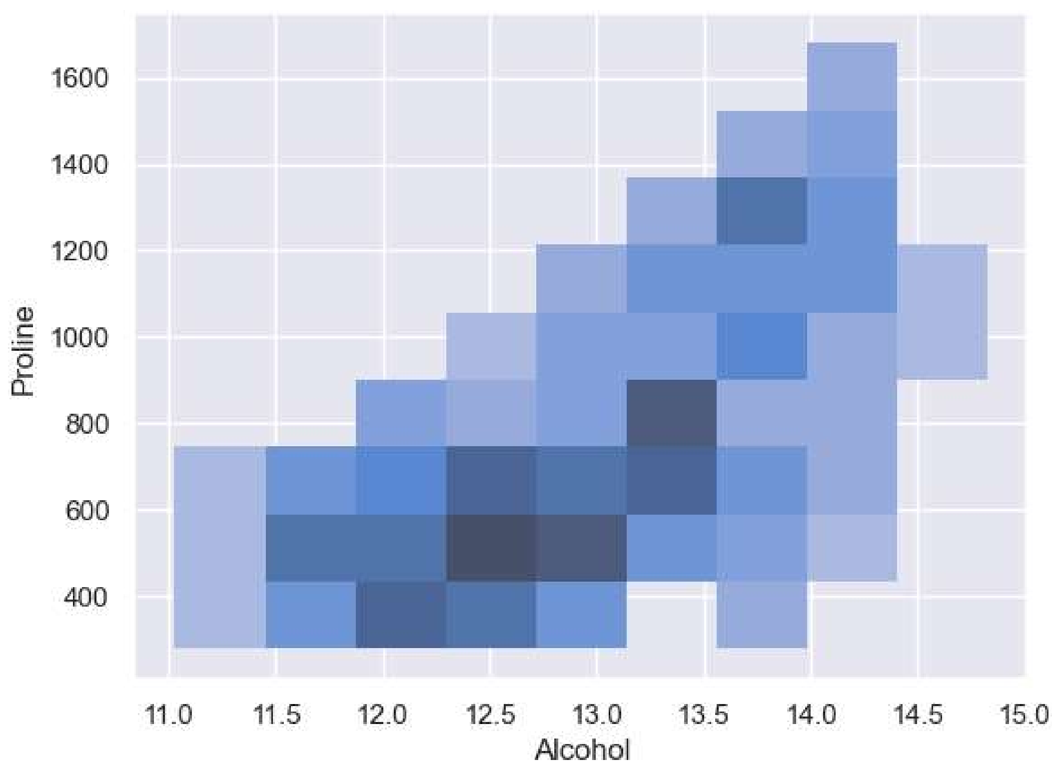
Out[64]:

	Alcohol	Malic_Acid	Ash	Ash_Alcanity	Magnesium	Total_Phenols	Flavanoids	Nonflavanoid_Phenols	Pro:
0	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	
1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	
2	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	
3	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	
4	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	
...	...	...	...	...	...	...	...	...	
173	13.71	5.65	2.45	20.5	95	1.68	0.61	0.52	
174	13.40	3.91	2.48	23.0	102	1.80	0.75	0.43	
175	13.27	4.28	2.26	20.0	120	1.59	0.69	0.43	
176	13.17	2.59	2.37	20.0	120	1.65	0.68	0.53	
177	14.13	4.10	2.74	24.5	96	2.05	0.76	0.56	

178 rows × 14 columns

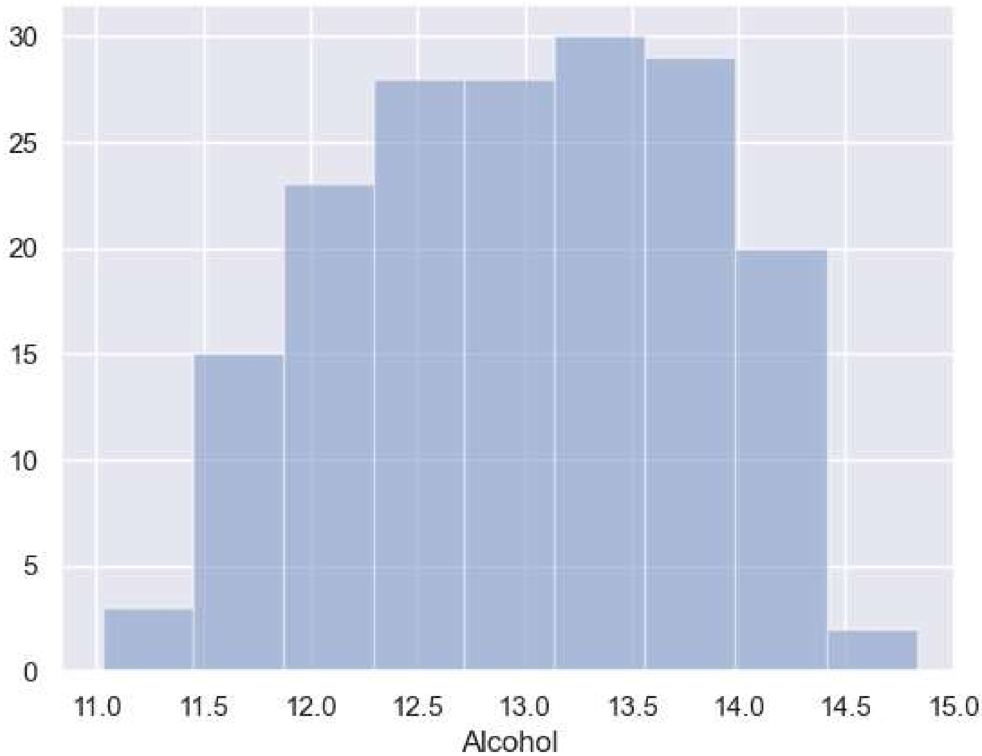
In [65]:
```
sns.histplot(data=df, x = "Alcohol",y="Proline")
```

Out[65]: <AxesSubplot: xlabel='Alcohol', ylabel='Proline'>



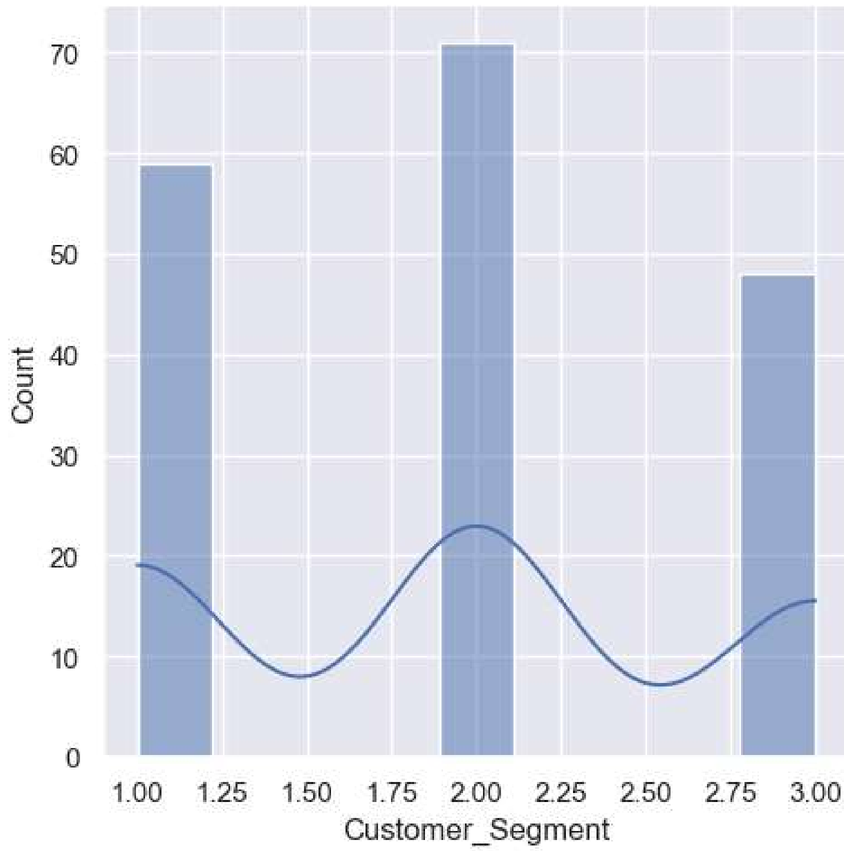
In [66]:
```
sns.distplot(a=df["Alcohol"],hist=True,kde=False,rug=False)
```

Out[66]:
```
<AxesSubplot: xlabel='Alcohol'>
```



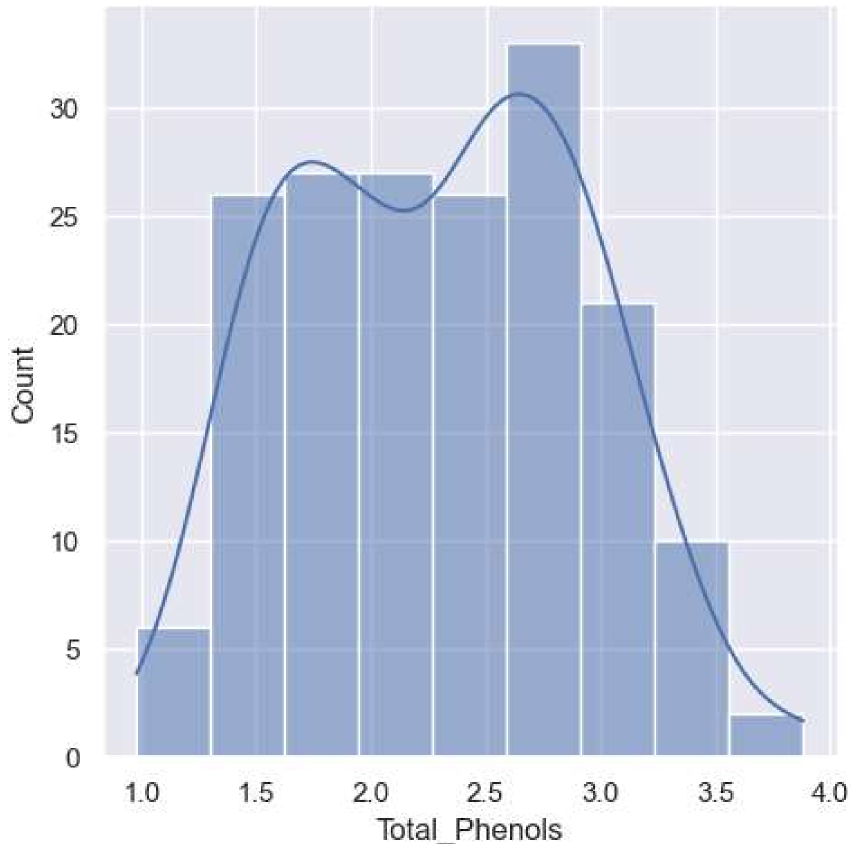
In [67]:
```
sns.displot(data=df["Customer_Segment"],kde=True)
```

Out[67]:
```
<seaborn.axisgrid.FacetGrid at 0x1dc9af23650>
```

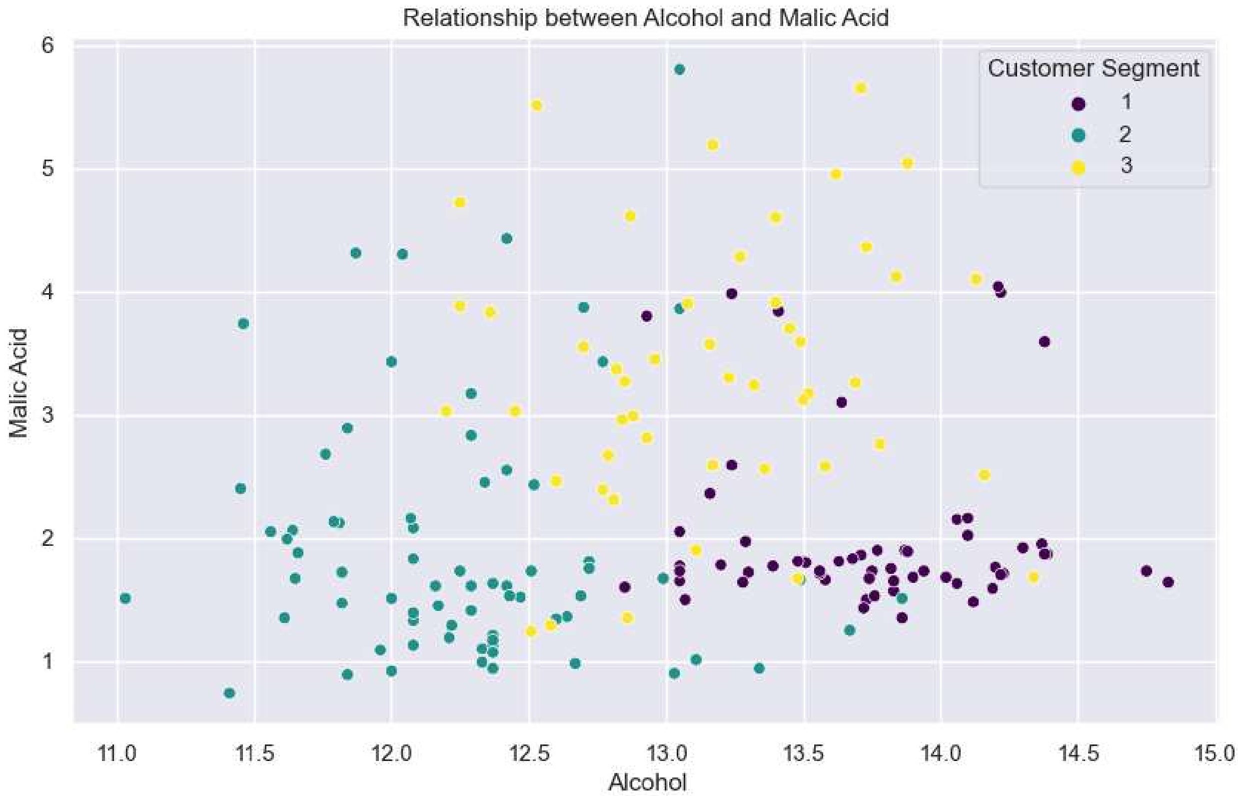


In [68]:
```
sns.displot(data=df["Total_Phenols"], kde=True)
```

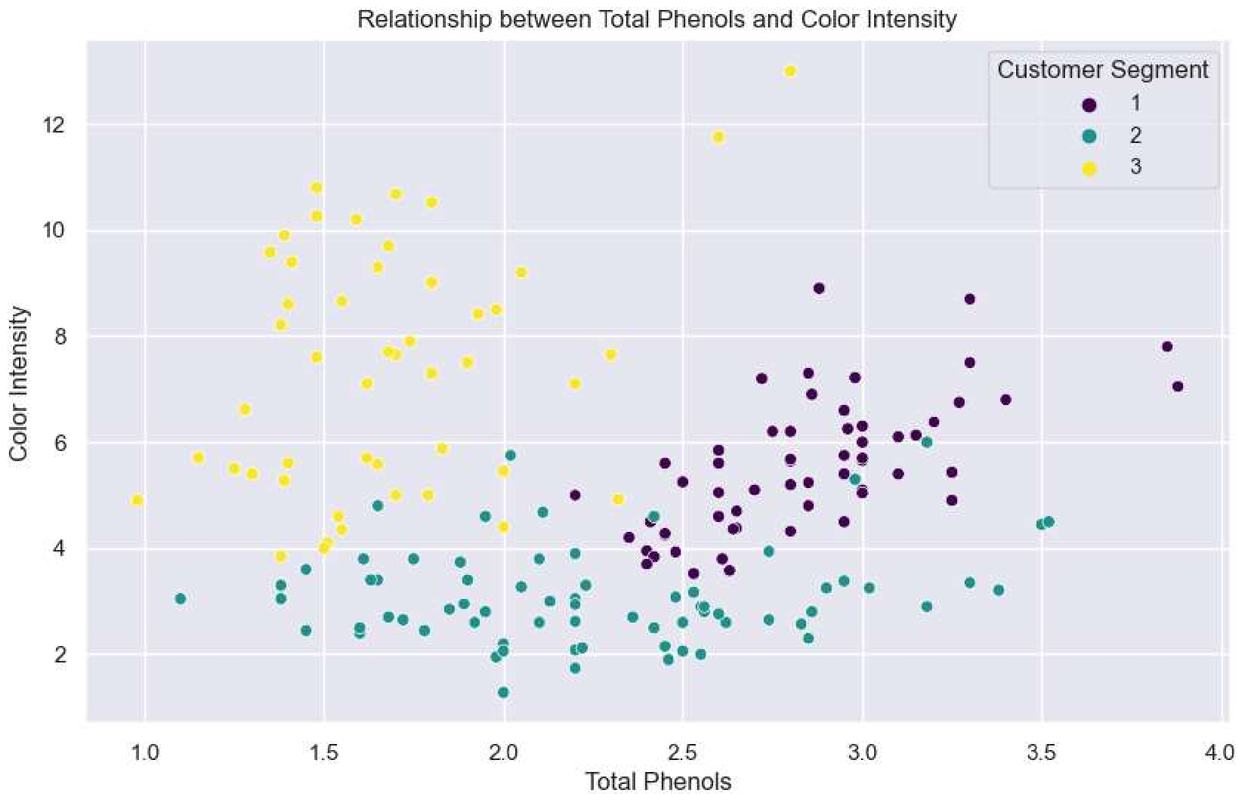
Out[68]:
```
<seaborn.axisgrid.FacetGrid at 0x1dc9e48ed10>
```



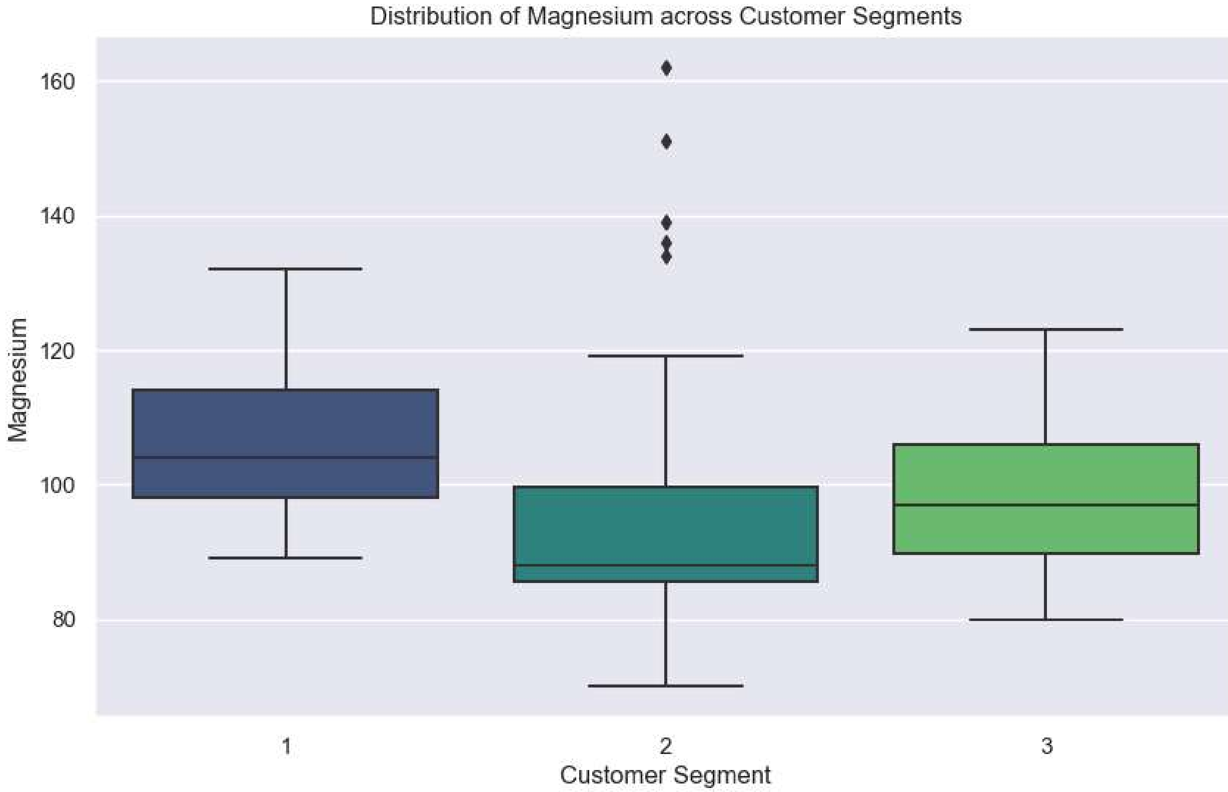
In [69]:
```
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df, x='Alcohol', y='Malic_Acid', hue='Customer_Segment', palette='viri
plt.title('Relationship between Alcohol and Malic Acid')
plt.xlabel('Alcohol')
plt.ylabel('Malic Acid')
plt.legend(title='Customer Segment')
plt.show()
```



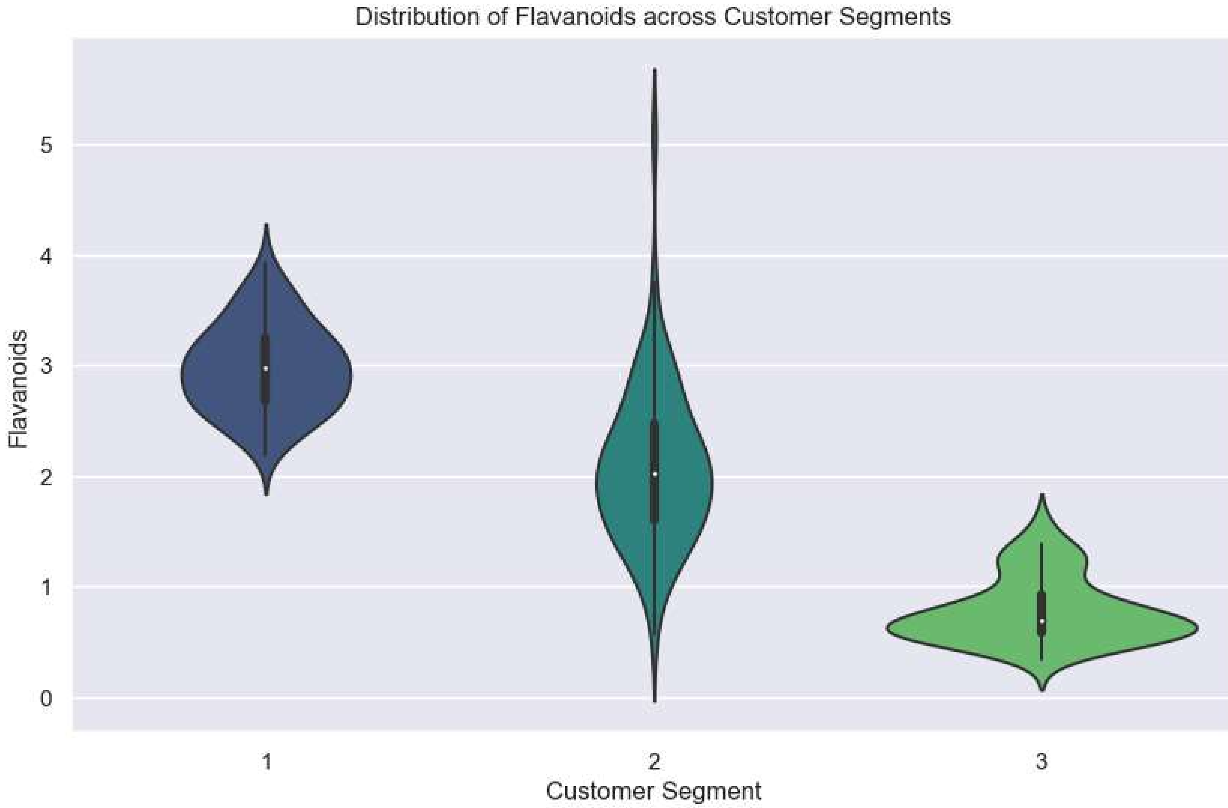
In [70]:
```
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df, x='Total_Phenols', y='Color_Intensity', hue='Customer_Segment', pa
plt.title('Relationship between Total Phenols and Color Intensity')
plt.xlabel('Total Phenols')
plt.ylabel('Color Intensity')
plt.legend(title='Customer Segment')
plt.show()
```



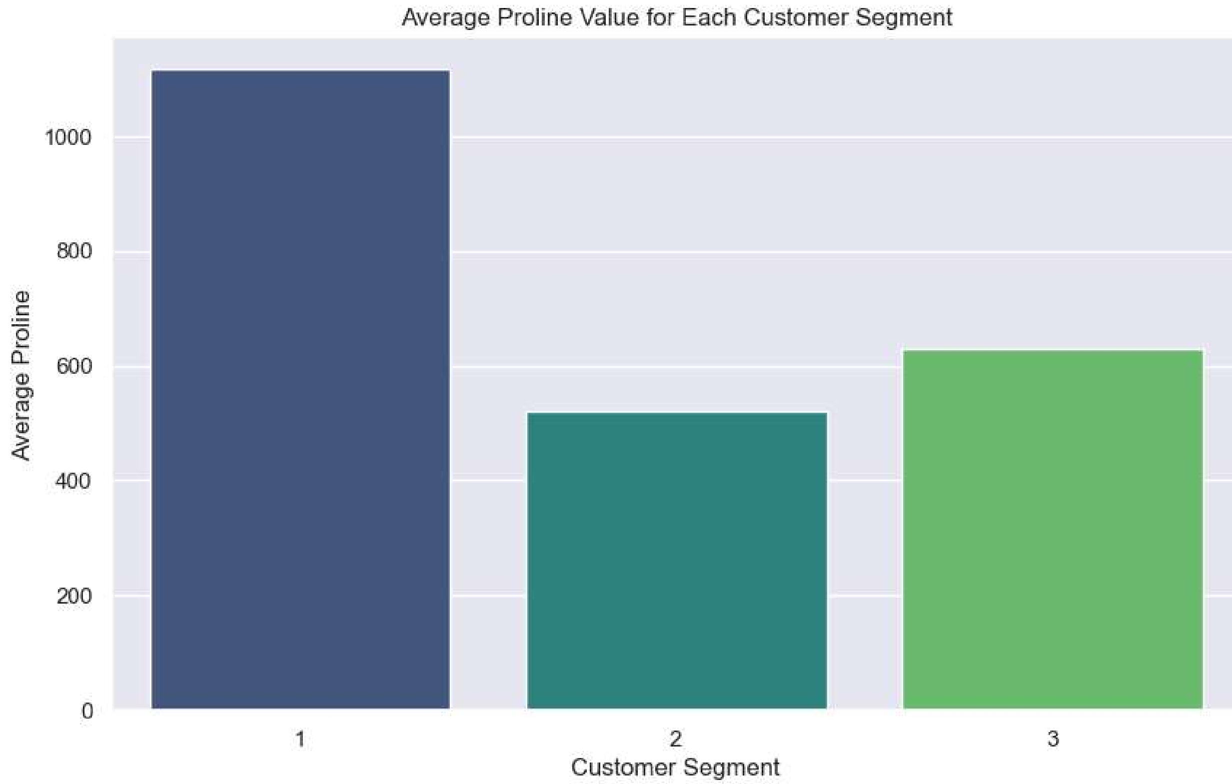
In [71]:
```
plt.figure(figsize=(10, 6))
sns.boxplot(data=df, x='Customer_Segment', y='Magnesium', palette='viridis')
plt.title('Distribution of Magnesium across Customer Segments')
plt.xlabel('Customer Segment')
plt.ylabel('Magnesium')
plt.show()
```



In [72]:
```
plt.figure(figsize=(10, 6))
sns.violinplot(data=df, x='Customer_Segment', y='Flavanoids', palette='viridis')
plt.title('Distribution of Flavanoids across Customer Segments')
plt.xlabel('Customer Segment')
plt.ylabel('Flavanoids')
plt.show()
```



In [73]:
```
average_proline = df.groupby('Customer_Segment')['Proline'].mean().reset_index()
plt.figure(figsize=(10, 6))
sns.barplot(data=average_proline, x='Customer_Segment', y='Proline', palette='viridis')
plt.title('Average Proline Value for Each Customer Segment')
plt.xlabel('Customer Segment')
plt.ylabel('Average Proline')
plt.show()
```



In [74]:
```
df
```

Out[74]:

	Alcohol	Malic_Acid	Ash	Ash_Alcanity	Magnesium	Total_Phenols	Flavanoids	Nonflavanoid_Phenols	Proa
0	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	
1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	
2	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	
3	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	
4	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	
...	...	...	...	...	...	...	...	...	
173	13.71	5.65	2.45	20.5	95	1.68	0.61	0.52	
174	13.40	3.91	2.48	23.0	102	1.80	0.75	0.43	
175	13.27	4.28	2.26	20.0	120	1.59	0.69	0.43	
176	13.17	2.59	2.37	20.0	120	1.65	0.68	0.53	
177	14.13	4.10	2.74	24.5	96	2.05	0.76	0.56	

178 rows × 14 columns

In [75]:
```
x = df.iloc[:,:-1].values
x
```

Out[75]:
```
array([[1.423e+01, 1.710e+00, 2.430e+00, ..., 1.040e+00, 3.920e+00,
        1.065e+03],
       [1.320e+01, 1.780e+00, 2.140e+00, ..., 1.050e+00, 3.400e+00,
        1.050e+03],
       [1.316e+01, 2.360e+00, 2.670e+00, ..., 1.030e+00, 3.170e+00,
        1.185e+03],
       ...,
       [1.327e+01, 4.280e+00, 2.260e+00, ..., 5.900e-01, 1.560e+00,
        8.350e+02],
       [1.317e+01, 2.590e+00, 2.370e+00, ..., 6.000e-01, 1.620e+00,
        8.400e+02],
       [1.413e+01, 4.100e+00, 2.740e+00, ..., 6.100e-01, 1.600e+00,
        5.600e+02]])
```

In [76]:
```
y = df.iloc[:,-1].values
y
```

Out[76]:
```
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, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
       3, 3], dtype=int64)
```

In [77]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state = 0
```

In [78]:
```
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
```

In [79]:
```
x_train.shape
```

Out[79]:
```
(142, 13)
```

In [80]:
```
x_test.shape
```

Out[80]:
```
(36, 13)
```

In [81]:
```
pca = PCA(n_components =2)
```

In [82]:
```
x_train = pca.fit_transform(x_train)
```

In [83]:
```
x_test = pca.transform(x_test)
```

In [84]:
```
x_train.shape
```

Out[84]:
```
(142, 2)
```

In [85]:
```
x_test.shape
```

Out[85]:
```
(36, 2)
```

In [86]:
```
pca.components_
```

Out[86]:
```
array([[ 0.12959991, -0.24464064, -0.01018912, -0.24051579,  0.12649451,
         0.38944115,  0.42757808, -0.30505669,  0.30775255, -0.11027186,
         0.30710508,  0.37636185,  0.2811085 ],
       [-0.49807323, -0.23168482, -0.31496874,  0.02321825, -0.25841951,
        -0.1006849 , -0.02097952, -0.0399057 , -0.06746036, -0.53087111,
         0.27161729,  0.16071181, -0.36547344]])
```

In [87]:
```
pca.explained_variance_ratio_
```

Out[87]:
```
array([0.36884109, 0.19318394])
```

In [88]:
```
classifier = LogisticRegression(random_state = 0)
```

In [89]:
```
classifier = LogisticRegression(random_state = 0)
```

In [90]:
```
classifier.fit(x_train , y_train)
```

Out[90]:
```
▾         LogisticRegression
LogisticRegression(random_state=0)
```

In [91]:
```
y_pred = classifier.predict(x_test)
y_pred
```

Out[91]:
```
array([1, 3, 2, 1, 2, 1, 1, 3, 2, 2, 3, 3, 1, 2, 3, 2, 1, 1, 2, 1, 2, 1,
       1, 2, 2, 2, 2, 2, 2, 3, 1, 1, 2, 1, 1, 1], dtype=int64)
```

In [92]:
```
y_test
```

Out[92]:
```
array([1, 3, 2, 1, 2, 2, 1, 3, 2, 2, 3, 3, 1, 2, 3, 2, 1, 1, 2, 1, 2, 1,
       1, 2, 2, 2, 2, 2, 2, 3, 1, 1, 2, 1, 1, 1], dtype=int64)
```

In [93]:
```
cm = confusion_matrix(y_test,y_pred)
print(cm)
```

```
[[14  0  0]
 [ 1 15  0]
 [ 0  0  6]]
```

In [95]:
```
accuracy_score(y_test, y_pred)
```

Out[95]:
```
0.9722222222222222
```

Thank You