In [1]:
```
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 [2]:
```
df=pd.read_csv('wine.csv')
df
```

Out[2]:

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

178 rows × 14 columns

In [3]:
```
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 [4]:
```
df.describe().round(2)
```

Out[4]:

	Alcohol	Malic_Acid	Ash	Ash_Alcanity	Magnesium	Total_Phenols	Flavanoids	Nonfla
count	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	
std	0.81	1.12	0.27	3.34	14.28	0.63	1.00	
min	11.03	0.74	1.36	10.60	70.00	0.98	0.34	
25%	12.36	1.60	2.21	17.20	88.00	1.74	1.20	
50%	13.05	1.87	2.36	19.50	98.00	2.36	2.13	
75%	13.68	3.08	2.56	21.50	107.00	2.80	2.88	
max	14.83	5.80	3.23	30.00	162.00	3.88	5.08	

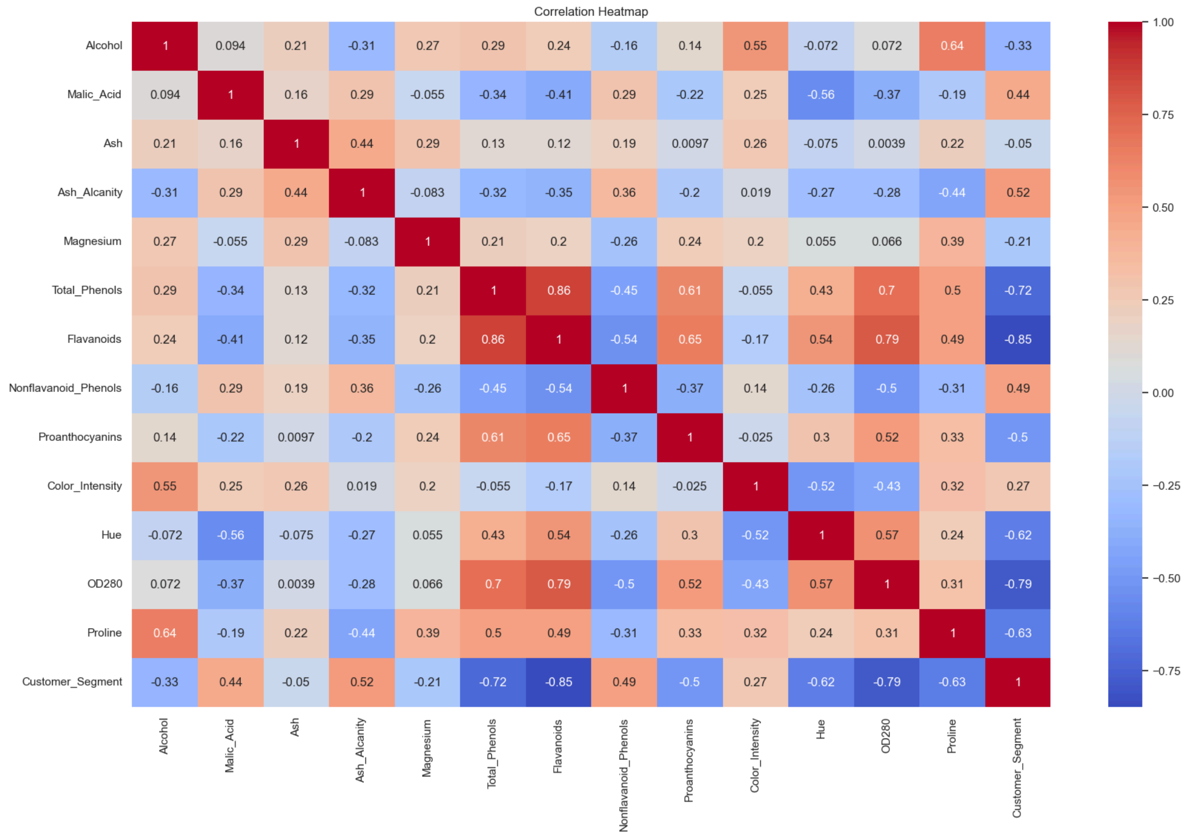
In [5]:
```
df.nunique()
```

Out[5]:
```
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 [6]:
```
df.isnull().sum()
```

Out[6]:
```
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 [7]:
```
correlation_matrix = df.corr()
plt.figure(figsize=(20, 12))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```



In [8]:
```
df
```

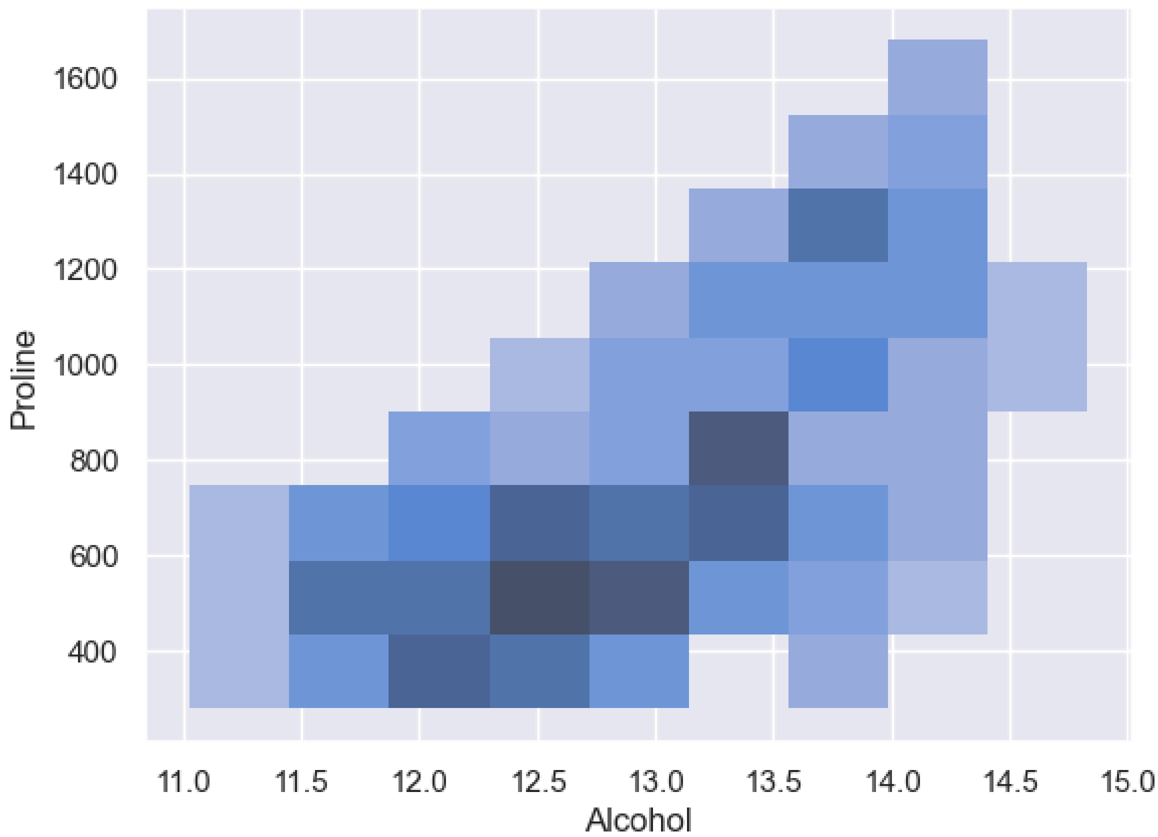
Out[8]:

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

178 rows × 14 columns

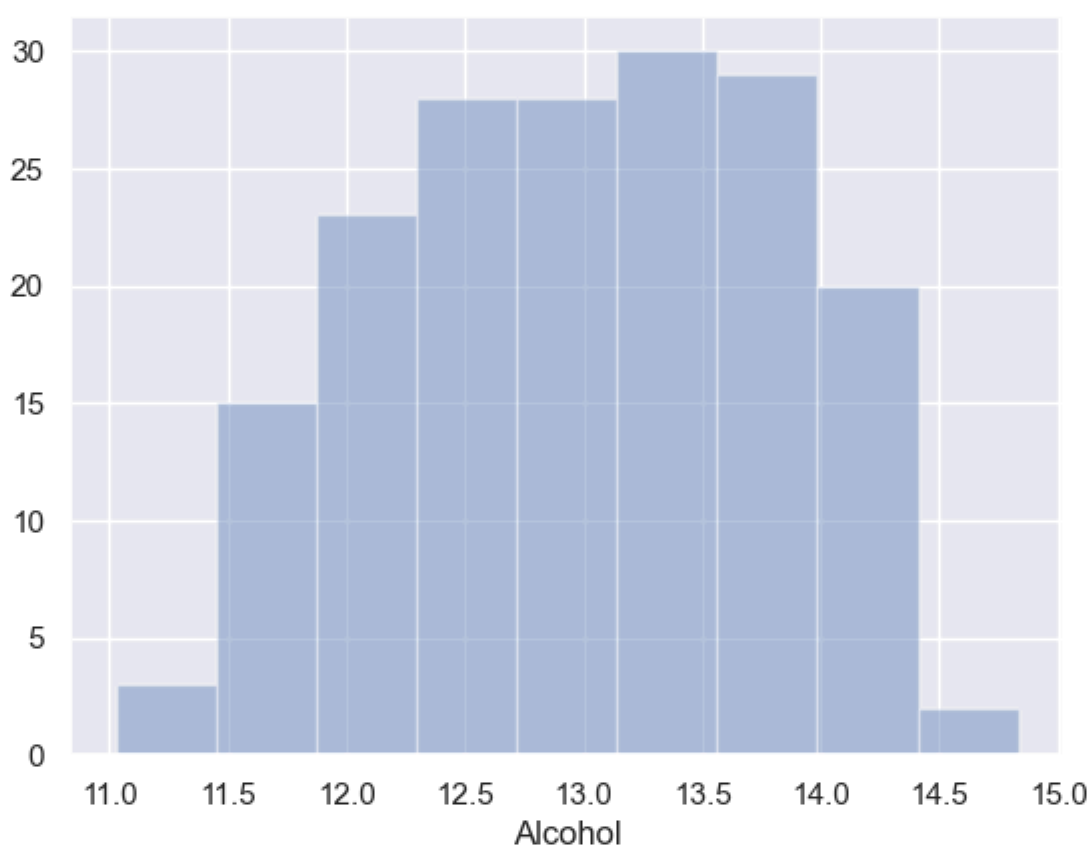
In [9]:
```
sns.histplot(
    data=df,
    x="Alcohol",
    y="Proline")
```

Out[9]: <Axes: xlabel='Alcohol', ylabel='Proline'>



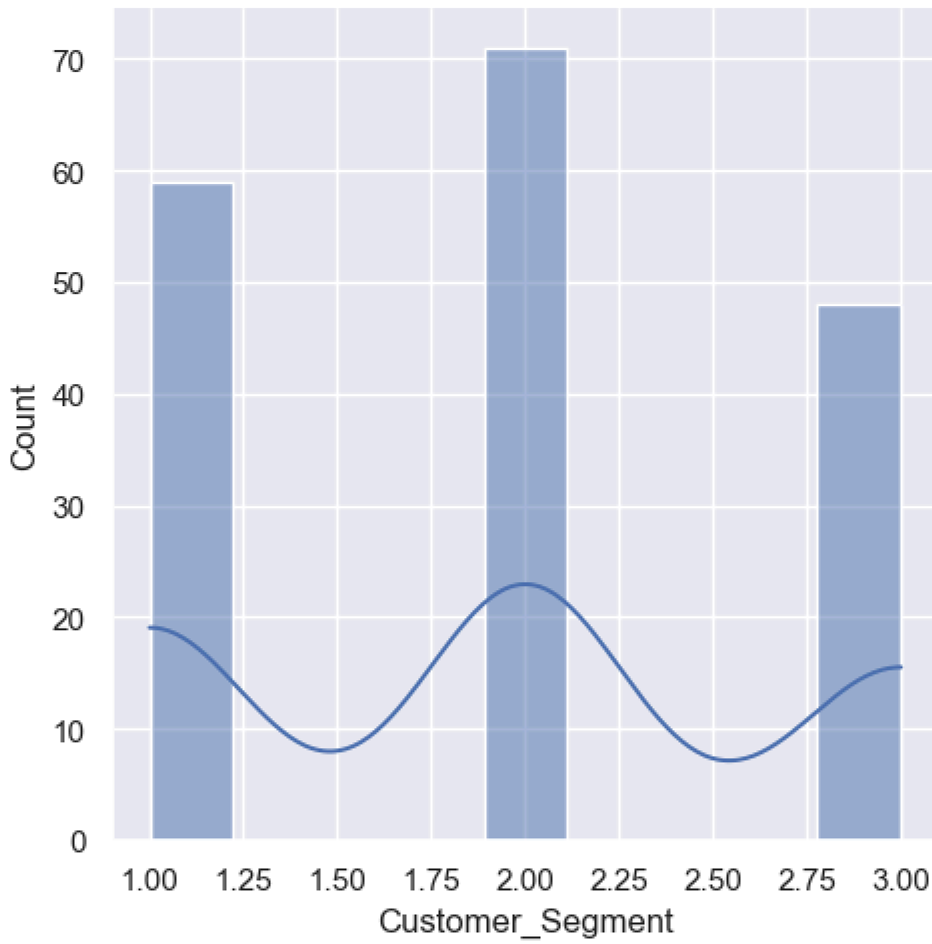
In [10]:
```
sns.distplot( a=df["Alcohol"],hist=True, kde=False, rug=False )
```

Out[10]: <Axes: xlabel='Alcohol'>



In [11]:
```
sns.displot( data=df["Customer_Segment"], kde=True )
```

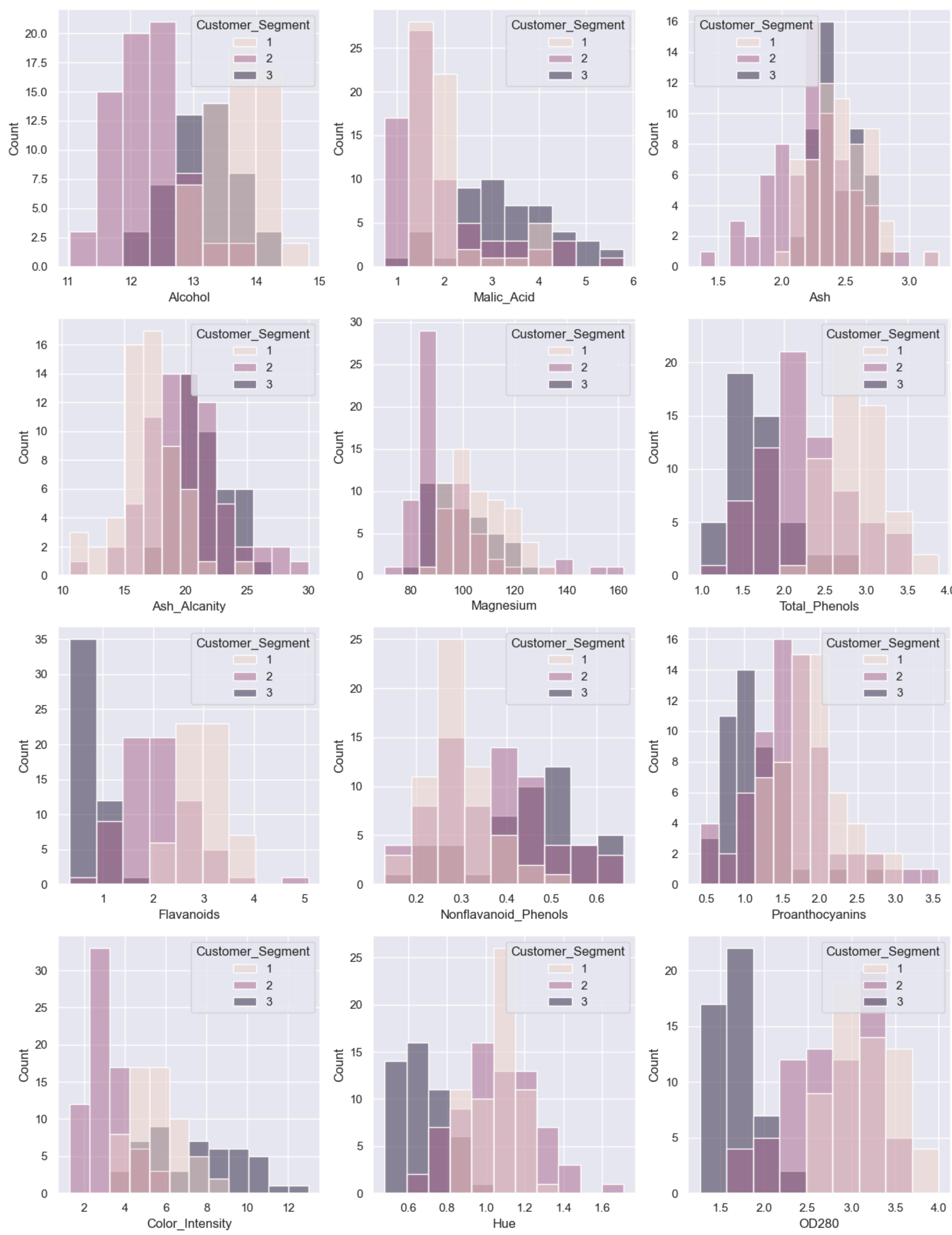
Out[11]:
```
<seaborn.axisgrid.FacetGrid at 0x1ed6627e710>
```



In [13]:
```
fig, axs = plt.subplots(nrows=4, ncols=3, figsize=(15,20))
axs=axs.flat
for i in range(len(df.columns)-1):
    sns.histplot(data=df, x=df.columns[i],hue="Customer_Segment",ax=axs[i])
```

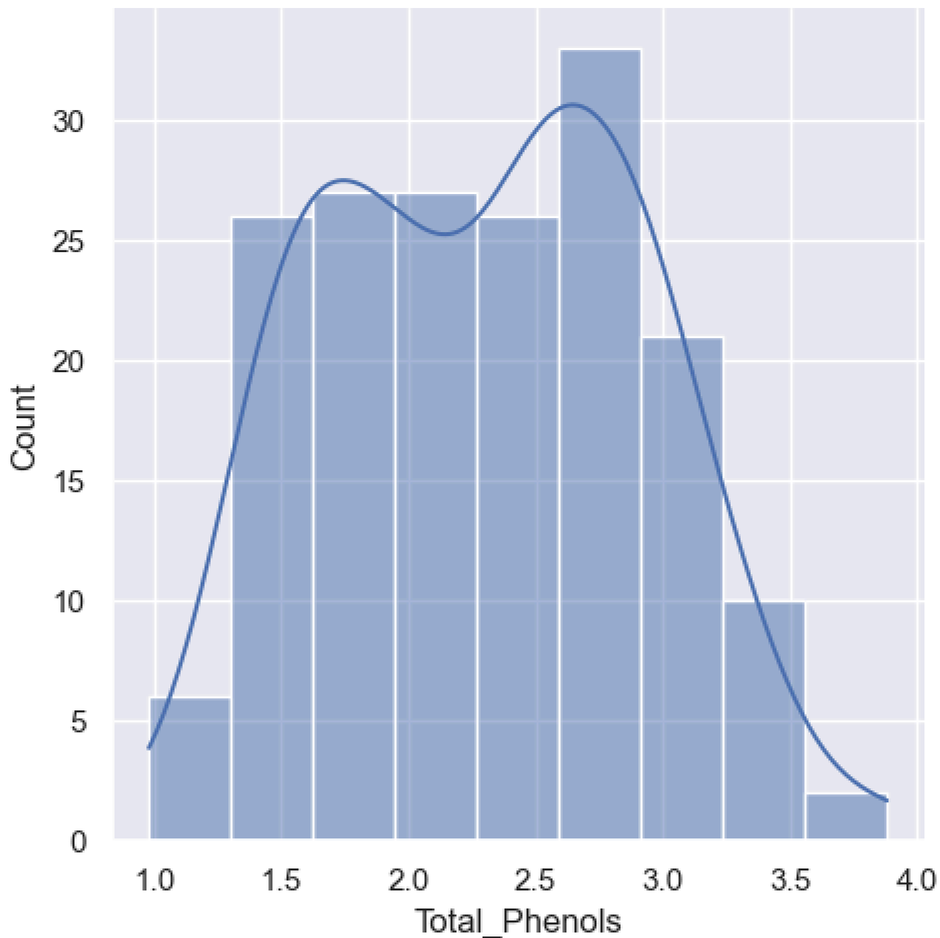
```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In[13], line 6
      4 # Iterate over columns except the last one ('Customer_Segment')
      5 for i, column in enumerate(df.columns[:-1]):  # Exclude the last colu
mn
----> 6     sns.histplot(data=df, x=column, hue="Customer_Segment", ax=axs
[i])
      8 # Remove unused subplots
      9 for ax in axs[len(df.columns)-1:]:

IndexError: index 12 is out of bounds for axis 0 with size 12
```

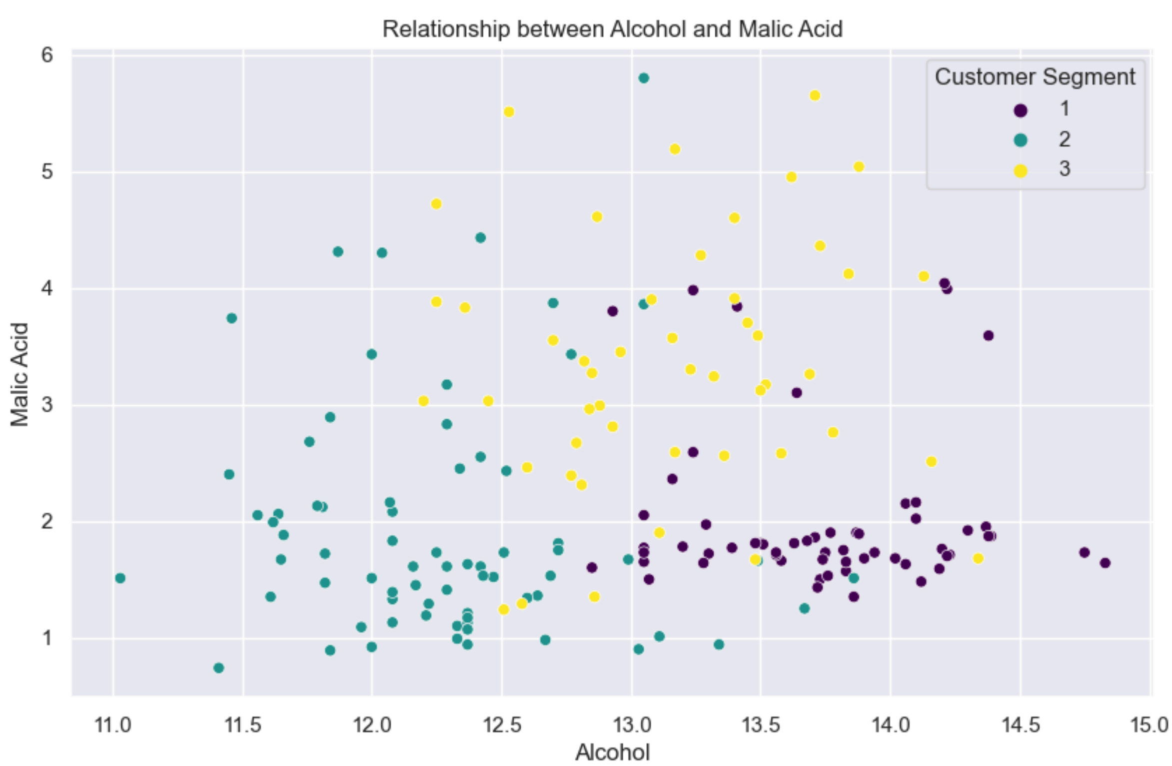


In [14]:
```
sns.displot( data=df["Total_Phenols"], kde=True )
```

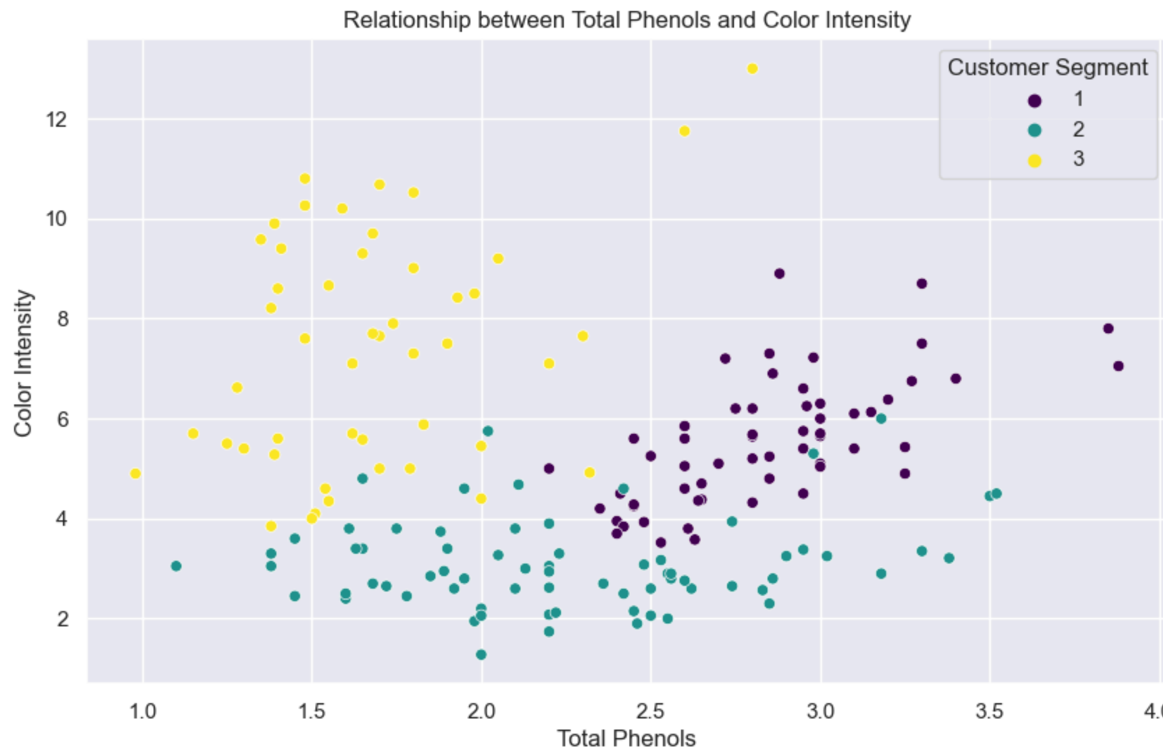
Out[14]: <seaborn.axisgrid.FacetGrid at 0x1ed697eda10>



In [15]:
```
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df, x='Alcohol', y='Malic_Acid', hue='Customer_Segment',
plt.title('Relationship between Alcohol and Malic Acid')
plt.xlabel('Alcohol')
plt.ylabel('Malic Acid')
plt.legend(title='Customer Segment')
plt.show()
```

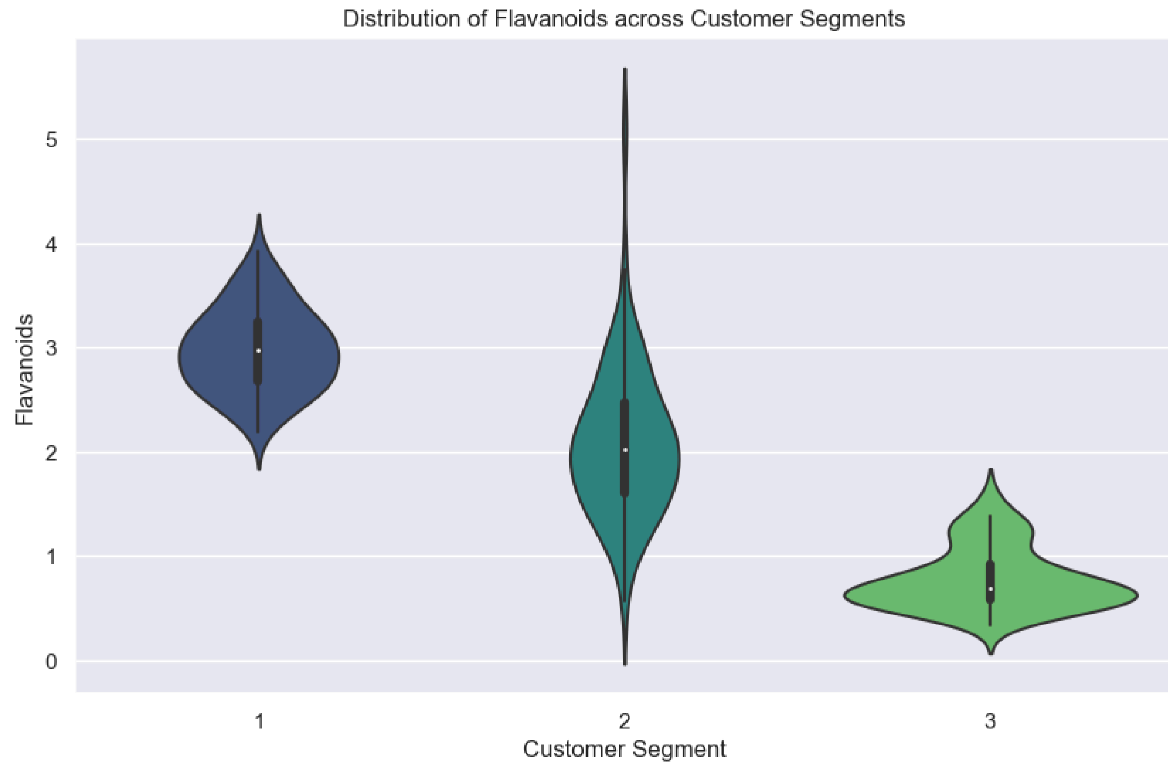


In [16]:
```
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df, x='Total_Phenols', y='Color_Intensity', hue='Customer
plt.title('Relationship between Total Phenols and Color Intensity')
plt.xlabel('Total Phenols')
plt.ylabel('Color Intensity')
plt.legend(title='Customer Segment')
plt.show()
```



In [17]:
```
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 [18]:
```
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 [19]:
```
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='
plt.title('Average Proline Value for Each Customer Segment')
plt.xlabel('Customer Segment')
plt.ylabel('Average Proline')
plt.show()
```

In [20]:
```
df
```

Out[20]:

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

178 rows × 14 columns

In [21]:
```
X = df.iloc[:, :-1].values
X
```

Out[21]:
```
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 [22]:
```
y = df.iloc[:, -1].values
y
```

Out[22]:
```
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 [23]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, ran
```

In [24]:
```
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

In [25]:
```
X_train.shape
```

Out[25]:
```
(142, 13)
```

In [26]:
```
X_test.shape
```

Out[26]:
```
(36, 13)
```

In [27]:
```
pca = PCA(n_components = 2)
```

In [28]:
```
X_train = pca.fit_transform(X_train)
```

In [29]:
```
X_test = pca.transform(X_test)
```

In [30]:
```
X_train.shape
```

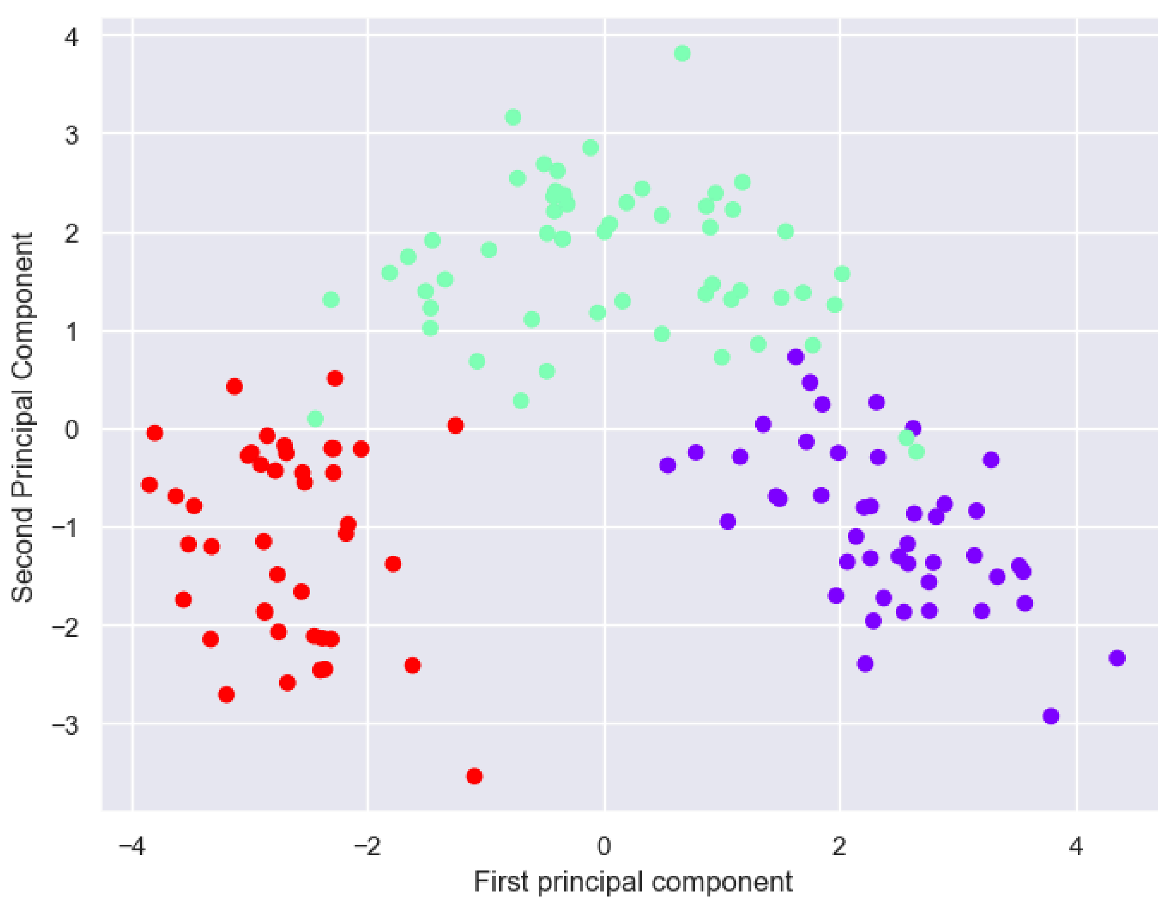
Out[30]: (142, 2)

In [31]:
```
X_test.shape
```

Out[31]: (36, 2)

In [32]:
```
plt.figure(figsize=(8,6))
plt.scatter(X_train[:,0],X_train[:,1],c=y_train,cmap='rainbow')
plt.xlabel('First principal component')
plt.ylabel('Second Principal Component')
```

Out[32]: Text(0, 0.5, 'Second Principal Component')



In [33]:
```
pca.components_
```

Out[33]:
```
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 [34]:
```
pca.explained_variance_ratio_
```

Out[34]: array([0.36884109, 0.19318394])

In [35]:
```
classifier = LogisticRegression(random_state = 0)
```

In [36]:
```
classifier.fit(X_train, y_train)
```

Out[36]:
```
▾ LogisticRegression
LogisticRegression(random_state=0)
```
(https://scikit-learn.org/1.4/modules/generated/sklearn.linear_model.Lo

In [37]:
```
y_pred = classifier.predict(X_test)
y_pred
```

Out[37]:
```
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 [38]:
```
y_test
```

Out[38]:
```
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 [39]:
```
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

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

In [40]:
```
accuracy_score(y_test, y_pred)
```

Out[40]:
```
0.9722222222222222
```

In [ ]:
```

```