```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

# import os
# for dirname, _, filenames in os.walk('/kaggle/input'):
#     for filename in filenames:
#         print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

Hour 2: Practical k-Means with the Wine Dataset 🍷

1. Load the Wine Dataset

We will load the Wine dataset using scikit-learn. This dataset contains the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines.

```
from sklearn.datasets import load_wine
import pandas as pd

# Load the Wine dataset
wine = load_wine()
wine_data = pd.DataFrame(data=wine.data, columns=wine.feature_names)
wine_target = pd.Series(wine.target)

print("Wine Dataset Shape:", wine_data.shape)
print("\nFirst 5 rows of the dataset:")
print(wine_data.head())
print("\nTarget Variable Distribution:")
print(wine_target.value_counts())
```

```
Wine Dataset Shape: (178, 13)

First 5 rows of the dataset:
   alcohol  malic_acid   ash  alcalinity_of_ash  magnesium  total_phenols  \
0    14.23        1.71  2.43               15.6      127.0           2.80
1    13.20        1.78  2.14               11.2      100.0           2.65
2    13.16        2.36  2.67               18.6      101.0           2.80
3    14.37        1.95  2.50               16.8      113.0           3.85
4    13.24        2.59  2.87               21.0      118.0           2.80

   flavanoids  nonflavanoid_phenols  proanthocyanins  color_intensity   hue  \
0        3.06                  0.28             2.29             5.64  1.04
1        2.76                  0.26             1.28             4.38  1.05
2        3.24                  0.30             2.81             5.68  1.03
3        3.49                  0.24             2.18             7.80  0.86
4        2.69                  0.39             1.82             4.32  1.04

   od280/od315_of_diluted_wines  proline
0                          3.92   1065.0
1                          3.40   1050.0
2                          3.17   1185.0
3                          3.45   1480.0
4                          2.93    735.0

Target Variable Distribution:
1    71
0    59
2    48
Name: count, dtype: int64
```

2. Explore and Preprocess the Data

We will explore the dataset's features and target variable. Then, we will check for missing values and scale the features using `StandardScaler`.

```
from sklearn.preprocessing import StandardScaler

# Check for missing values
print("\nMissing values per column:")
print(wine_data.isnull().sum())

# Scale the features
scaler = StandardScaler()
scaled_data = scaler.fit_transform(wine_data)
scaled_wine_data = pd.DataFrame(scaled_data, columns=wine_data.columns)

print("\nScaled data shape:")
print(scaled_wine_data.shape)
print("\nFirst 5 rows of scaled data:")
print(scaled_wine_data.head())
```

```
Missing values per column:
alcohol                         0
malic_acid                      0
ash                             0
alcalinity_of_ash               0
magnesium                       0
total_phenols                   0
flavanoids                      0
nonflavanoid_phenols            0
proanthocyanins                 0
color_intensity                 0
hue                             0
od280/od315_of_diluted_wines    0
proline                         0
dtype: int64
```

```
Scaled data shape:
(178, 13)

First 5 rows of scaled data:
    alcohol  malic_acid       ash  alcalinity_of_ash  magnesium  \
0  1.518613   -0.562250  0.232053          -1.169593   1.913905
1  0.246290   -0.499413 -0.827996          -2.490847   0.018145
2  0.196879    0.021231  1.109334          -0.268738   0.088358
3  1.691550   -0.346811  0.487926          -0.809251   0.930918
4  0.295700    0.227694  1.840403           0.451946   1.281985

   total_phenols  flavanoids  nonflavanoid_phenols  proanthocyanins  \
0       0.808997    1.034819             -0.659563         1.224884
1       0.568648    0.733629             -0.820719        -0.544721
2       0.808997    1.215533             -0.498407         2.135968
3       2.491446    1.466525             -0.981875         1.032155
4       0.808997    0.663351              0.226796         0.401404

   color_intensity       hue  od280/od315_of_diluted_wines   proline
0         0.251717  0.362177                      1.847920  1.013009
1        -0.293321  0.406051                      1.113449  0.965242
2         0.269020  0.318304                      0.788587  1.395148
3         1.186068 -0.427544                      1.184071  2.334574
4        -0.319276  0.362177                      0.449601 -0.037874
```

3. Apply k-Means Clustering

We will apply the k-Means algorithm to the scaled data.

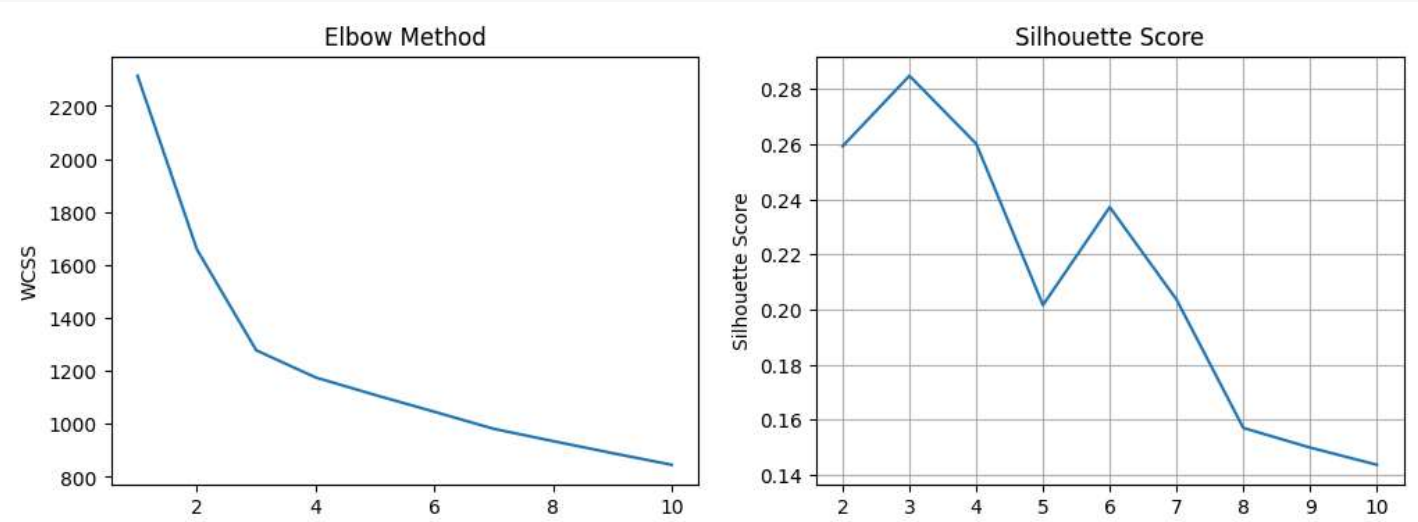
```
from sklearn.cluster import KMeans

# Apply k-Means
kmeans = KMeans(n_clusters=3, random_state=42) #initial assumption: 3 clusters
kmeans.fit(scaled_wine_data)
labels = kmeans.labels_

print("\nCluster labels:")
print(labels)
```

```
Cluster labels:
[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 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2
 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 1 0 0 2 0 0 0 0 0 0 0 0 0 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 1]
```

4. Determine the Optimal Number of Clusters (k)

We will use the elbow method and silhouette score to find the best 'k'.

```
import matplotlib.pyplot as plt
from sklearn.metrics import silhouette_score

fig, axes = plt.subplots(1,2, figsize = (12,4))

# Elbow Method
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, random_state=42, n_init=10)
    kmeans.fit(scaled_wine_data)
    wcss.append(kmeans.inertia_)

axes[0].plot(range(1, 11), wcss)
axes[0].set_title('Elbow Method')
axes[0].set_xlabel('Number of clusters')
axes[0].set_ylabel('WCSS')

# Silhouette Score
silhouette_scores = []
for i in range(2, 11):
    kmeans = KMeans(n_clusters=i, random_state=42, n_init=10)
    labels = kmeans.fit_predict(scaled_wine_data)
    silhouette_avg = silhouette_score(scaled_wine_data, labels)
    silhouette_scores.append(silhouette_avg)

axes[1].plot(range(2, 11), silhouette_scores)
axes[1].set_title('Silhouette Score')
axes[1].set_xlabel('Number of clusters')
axes[1].set_ylabel('Silhouette Score')

plt.grid(True)
plt.show()
```



5. Analyze the Clusters

We will analyze the characteristics of the clusters by examining the mean values of each feature within each cluster.

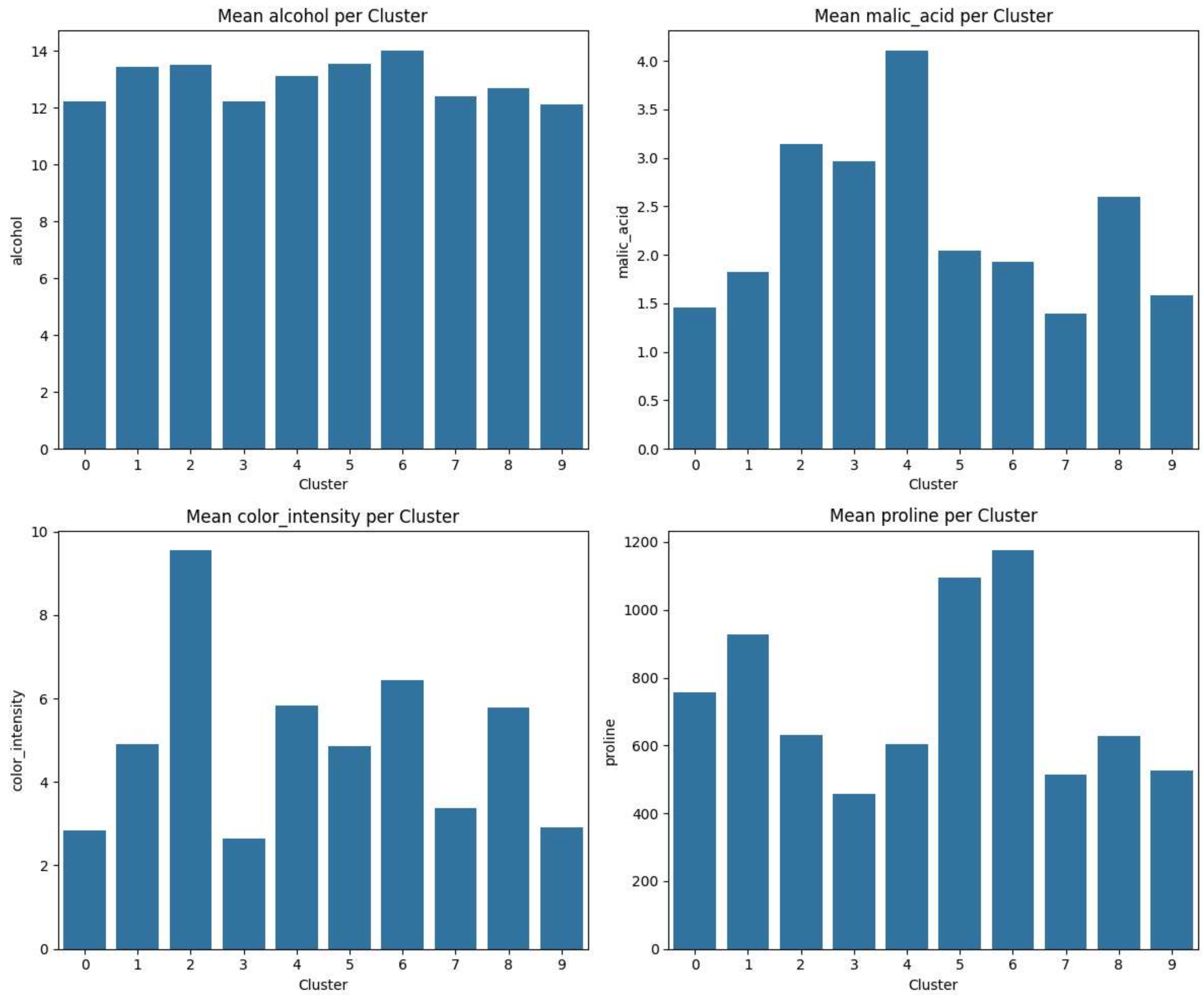
```
# Analyze cluster characteristics
wine_data['Cluster'] = labels
cluster_means = wine_data.groupby('Cluster').mean()
```

```
print("\nCluster Means:")
print(cluster_means)
```

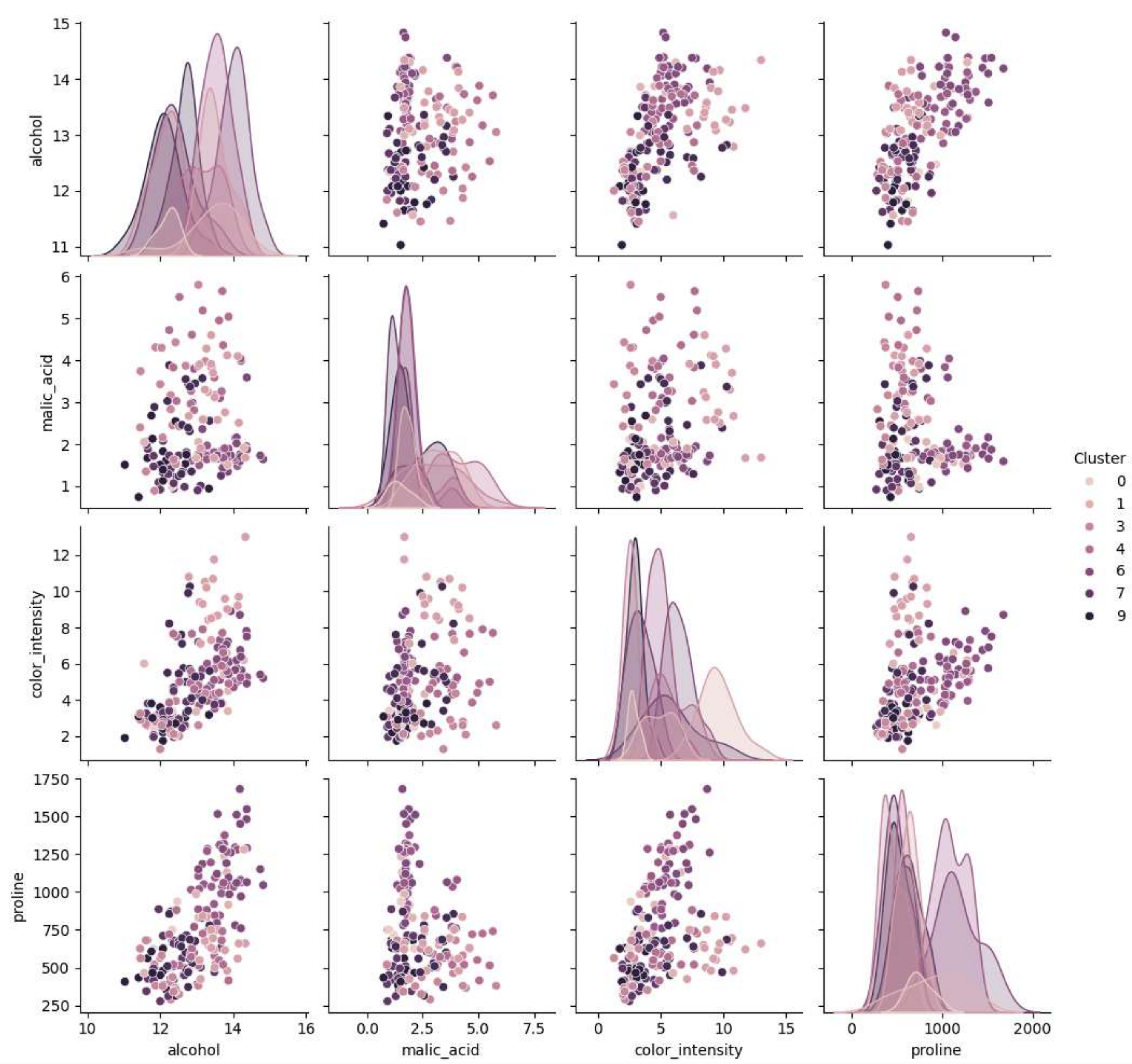
```
Cluster Means:
           alcohol  malic_acid       ash  alcalinity_of_ash   magnesium  \
Cluster
0        12.205000    1.455000  2.160000          18.025000  145.750000
1        13.419000    1.829000  2.813000          23.090000  116.100000
2        13.501667    3.146111  2.520556          22.333333  104.277778
3        12.232105    2.963684  2.307895          20.815789   92.684211
4        13.114118    4.106471  2.467647          21.941176   93.176471
5        13.550370    2.045556  2.447407          16.985185  102.333333
6        14.002500    1.931667  2.370000          15.904167  107.958333
7        12.409091    1.389545  1.979091          18.286364   88.727273
8        12.666667    2.596111  2.245556          19.261111   99.222222
9        12.094737    1.585263  2.441053          21.436842   89.263158

         total_phenols  flavanoids  nonflavanoid_phenols  proanthocyanins  \
Cluster
0             1.962500    1.597500              0.237500         2.525000
1             2.951000    3.170000              0.364000         1.862000
2             1.820556    0.967222              0.418889         1.490556
3             2.538947    2.465263              0.334211         1.982632
4             1.700588    0.642353              0.537059         0.986471
5             2.603704    2.697778              0.284444         1.634444
6             3.106250    3.317500              0.265000         2.222917
7             2.353182    2.100909              0.280000         1.465909
8             1.501111    0.876111              0.398889         0.958333
9             1.979474    1.763158              0.495789         1.379474

         color_intensity       hue  od280/od315_of_diluted_wines      proline
Cluster
0               2.837500  1.112500                      2.567500   757.500000
1               4.903000  1.166000                      3.081000   927.000000
2               9.540000  0.619444                      1.608333   631.944444
3               2.645789  0.905263                      3.068421   456.578947
4               5.823529  0.757647                      1.809412   603.823529
5               4.847407  1.066667                      3.195556  1094.444444
6               6.442917  1.040833                      3.190417  1174.458333
7               3.363636  1.110455                      2.885909   514.136364
8               5.775000  0.729222                      1.731111   627.888889
9               2.921579  1.144737                      2.541579   525.368421
```

Start coding or generate with AI.

```
# Visualization of cluster means for selected features
selected_features = ['alcohol', 'malic_acid', 'color_intensity', 'proline']

# Selected features for visualization
selected_features = ['alcohol', 'malic_acid', 'color_intensity', 'proline']

# Create a 2x2 grid of subplots
fig, axes = plt.subplots(2, 2, figsize=(12, 10))

# Flatten the axes array for easier iteration
axes = axes.flatten()

# Loop through features and plot in each subplot
for idx, feature in enumerate(selected_features):
    sns.barplot(x=cluster_means.index, y=cluster_means[feature], ax=axes[idx])
    axes[idx].set_title(f'Mean {feature} per Cluster')
    axes[idx].set_xlabel('Cluster')
    axes[idx].set_ylabel(feature)

# Adjust layout for better spacing
plt.tight_layout()
plt.show()
```




```
# Pair plot of selected features with cluster labels
sns.pairplot(wine_data, vars=selected_features, hue='Cluster')
plt.show()
```



Double-click (or enter) to edit

6. Interpret the Results and Draw Conclusions

We will interpret the clustering results in the context of the wine dataset and draw conclusions about the different groups of wine samples based on the feature means.

Step 1: Examine the Cluster Means

Let's look at each feature and compare the mean values across the three clusters:

- **alcohol:** Cluster 0: 13.74, Cluster 1: 12.26, Cluster 2: 13.15
- **malic_acid:** Cluster 0: 2.01, Cluster 1: 1.94, Cluster 2: 3.33
- **ash:** Cluster 0: 2.46, Cluster 1: 2.24, Cluster 2: 2.44
- **alcalinity_of_ash:** Cluster 0: 17.03, Cluster 1: 20.24, Cluster 2: 21.10
- **magnesium:** Cluster 0: 106.33, Cluster 1: 92.77, Cluster 2: 98.81
- **total_phenols:** Cluster 0: 2.84, Cluster 1: 2.07, Cluster 2: 1.68