1.Compare the clustering results obtained from the original dataset and PCA-transformed data:

To compare the clustering results, you can visually inspect the clusters formed in both cases and evaluate how similar or different they are. Additionally, you can compare clustering evaluation metrics like silhouette score or Davies–Bouldin index for both cases.

2.Discuss any similarities or differences observed in the clustering results:

After comparing the clustering results, discuss any patterns or differences you observe. Are the clusters formed in both cases similar or do they differ significantly? Are there any clusters present in one representation but not in the other? Analyze the reasons behind these similarities or differences.

3.Reflect on the impact of dimensionality reduction on clustering performance:

Consider how dimensionality reduction using PCA impacted the clustering performance. Did PCA improve or degrade the clustering results compared to using the original dataset? Reflect on how reducing the dimensionality affected the clustering algorithm's ability to capture the underlying structure of the data.

4.Analyze the trade-offs between using PCA and clustering directly on the original dataset:

Analyze the trade-offs between using PCA and clustering directly on the original dataset. Consider factors such as computational efficiency, interpretability of results, and clustering performance. Discuss the advantages and disadvantages of each approach and under what circumstances one might be preferred over the other.

Conclusion:

Summary of Key Findings and Insights:

Provide a brief summary of the key findings from the assignment, including observations from data exploration, clustering results, and comparison between original and PCA-transformed data.

Highlight any significant patterns, similarities, or differences observed during the analysis.

Practical Implications of Using PCA and Clustering in Data Analysis:

Discuss the practical implications of using PCA and clustering in real-world data analysis scenarios.

Explain how PCA helps in reducing the dimensionality of high-dimensional datasets, thereby simplifying the analysis and potentially improving computational efficiency.

Highlight the role of clustering in identifying meaningful patterns or groups within the data, which can be valuable for tasks like customer segmentation, anomaly detection, or recommendation systems.

Recommendations for When to Use Each Technique:

Based on the analysis conducted, provide recommendations for when to use PCA and clustering in data analysis.

Suggest using PCA when dealing with high-dimensional data to reduce dimensionality and remove redundant information while preserving important patterns.

Recommend clustering techniques when there is a need to identify natural groupings or clusters within the data, which can provide valuable insights for decision-making.

Discuss scenarios where using PCA in combination with clustering may yield the best results, such as preprocessing data with PCA before applying clustering algorithms.

pca

May 16, 2024

[28]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
import warnings
warnings.filterwarnings('ignore')
```

[5]:
```
df=pd.read_csv('/content/wine.csv')
```

[6]:
```
df.head()
```

[6]:
```
   Type  Alcohol  Malic   Ash  Alcalinity  Magnesium  Phenols  Flavanoids  \
0     1    14.23   1.71  2.43        15.6        127     2.80        3.06
1     1    13.20   1.78  2.14        11.2        100     2.65        2.76
2     1    13.16   2.36  2.67        18.6        101     2.80        3.24
3     1    14.37   1.95  2.50        16.8        113     3.85        3.49
4     1    13.24   2.59  2.87        21.0        118     2.80        2.69

   Nonflavanoids  Proanthocyanins  Color   Hue  Dilution  Proline
0           0.28             2.29   5.64  1.04      3.92     1065
1           0.26             1.28   4.38  1.05      3.40     1050
2           0.30             2.81   5.68  1.03      3.17     1185
3           0.24             2.18   7.80  0.86      3.45     1480
4           0.39             1.82   4.32  1.04      2.93      735
```

[7]:
```
df.shape
```

[7]:
```
(178, 14)
```

[8]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 178 entries, 0 to 177
Data columns (total 14 columns):
```

```
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Type             178 non-null    int64
 1   Alcohol          178 non-null    float64
 2   Malic            178 non-null    float64
 3   Ash              178 non-null    float64
 4   Alcalinity       178 non-null    float64
 5   Magnesium        178 non-null    int64
 6   Phenols          178 non-null    float64
 7   Flavanoids       178 non-null    float64
 8   Nonflavanoids    178 non-null    float64
 9   Proanthocyanins  178 non-null    float64
 10  Color            178 non-null    float64
 11  Hue              178 non-null    float64
 12  Dilution         178 non-null    float64
 13  Proline          178 non-null    int64
dtypes: float64(11), int64(3)
memory usage: 19.6 KB
```

[9]:
```
df.describe()
```

[9]:

	Type	Alcohol	Malic	Ash	Alcalinity	Magnesium	\
count	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	
mean	1.938202	13.000618	2.336348	2.366517	19.494944	99.741573	
std	0.775035	0.811827	1.117146	0.274344	3.339564	14.282484	
min	1.000000	11.030000	0.740000	1.360000	10.600000	70.000000	
25%	1.000000	12.362500	1.602500	2.210000	17.200000	88.000000	
50%	2.000000	13.050000	1.865000	2.360000	19.500000	98.000000	
75%	3.000000	13.677500	3.082500	2.557500	21.500000	107.000000	
max	3.000000	14.830000	5.800000	3.230000	30.000000	162.000000	

	Phenols	Flavanoids	Nonflavanoids	Proanthocyanins	Color	\
count	178.000000	178.000000	178.000000	178.000000	178.000000	
mean	2.295112	2.029270	0.361854	1.590899	5.058090	
std	0.625851	0.998859	0.124453	0.572359	2.318286	
min	0.980000	0.340000	0.130000	0.410000	1.280000	
25%	1.742500	1.205000	0.270000	1.250000	3.220000	
50%	2.355000	2.135000	0.340000	1.555000	4.690000	
75%	2.800000	2.875000	0.437500	1.950000	6.200000	
max	3.880000	5.080000	0.660000	3.580000	13.000000	

	Hue	Dilution	Proline
count	178.000000	178.000000	178.000000
mean	0.957449	2.611685	746.893258
std	0.228572	0.709990	314.907474
min	0.480000	1.270000	278.000000
25%	0.782500	1.937500	500.500000

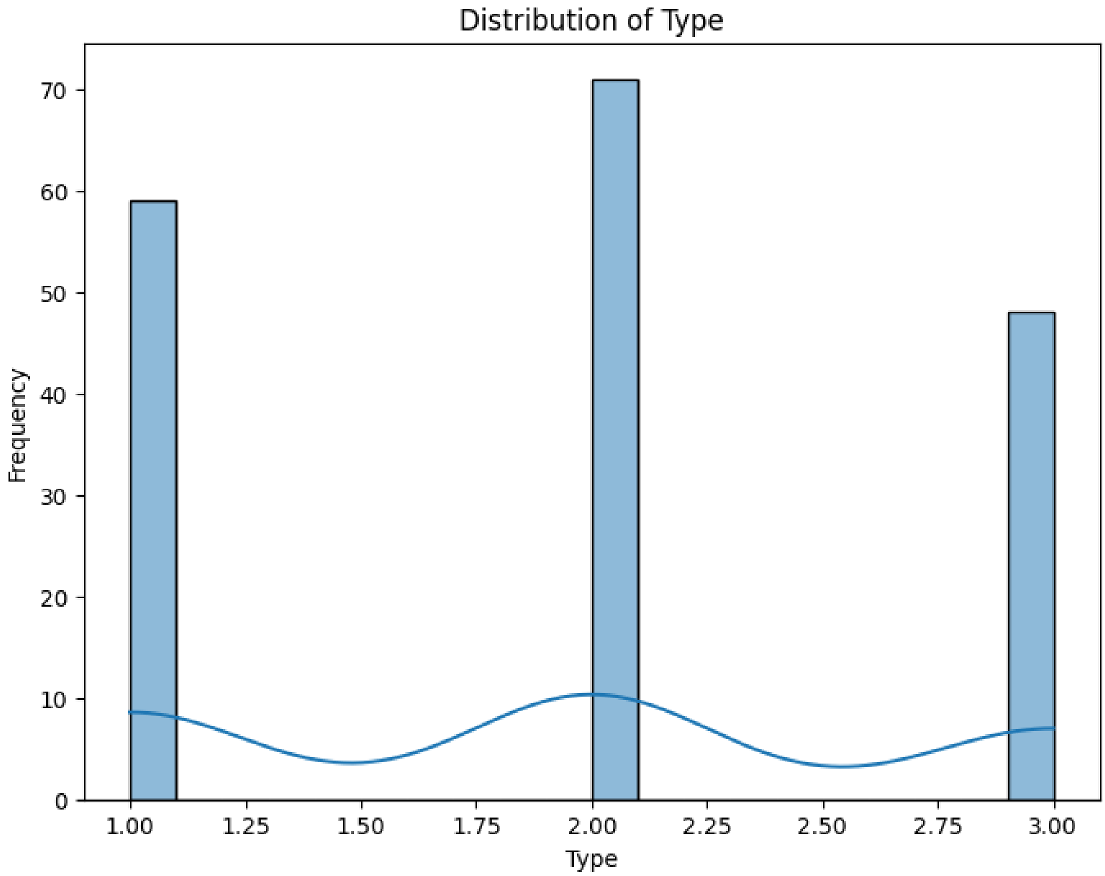
```
50%      0.965000    2.780000    673.500000
75%      1.120000    3.170000    985.000000
max      1.710000    4.000000   1680.000000
```

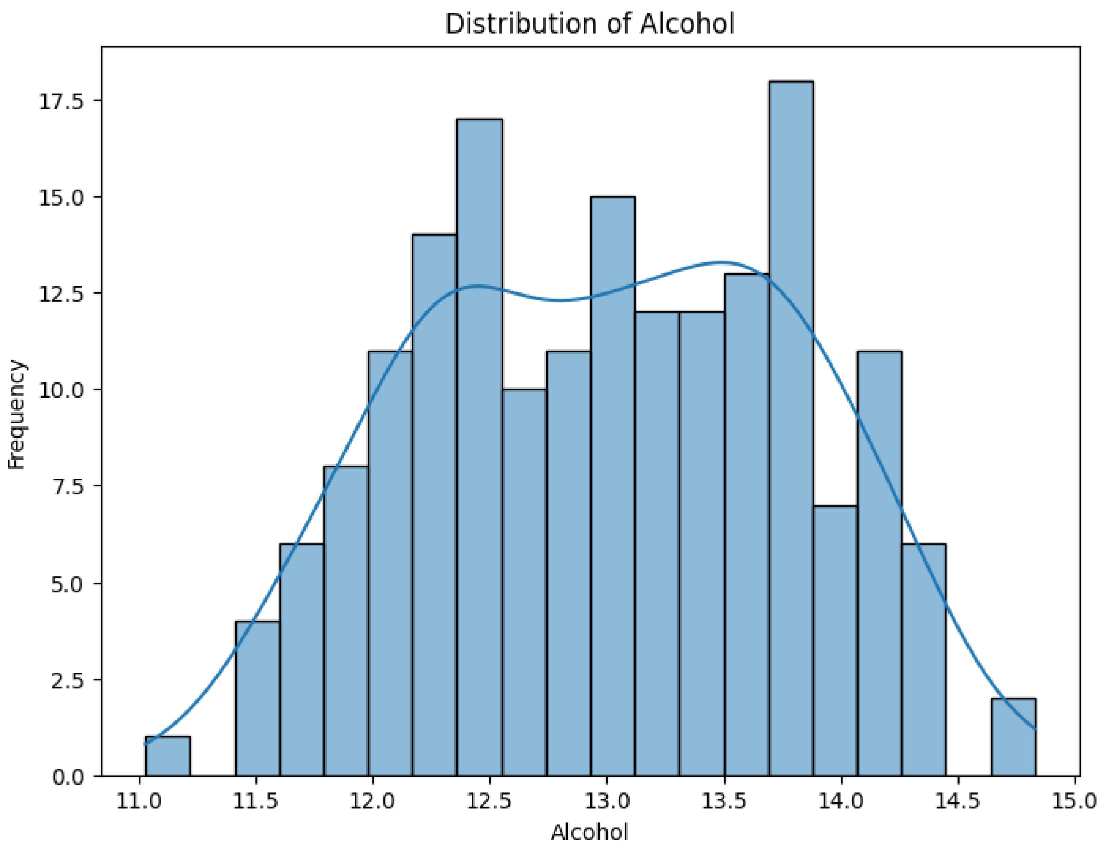
[10]:
```
df.isnull().sum()
```

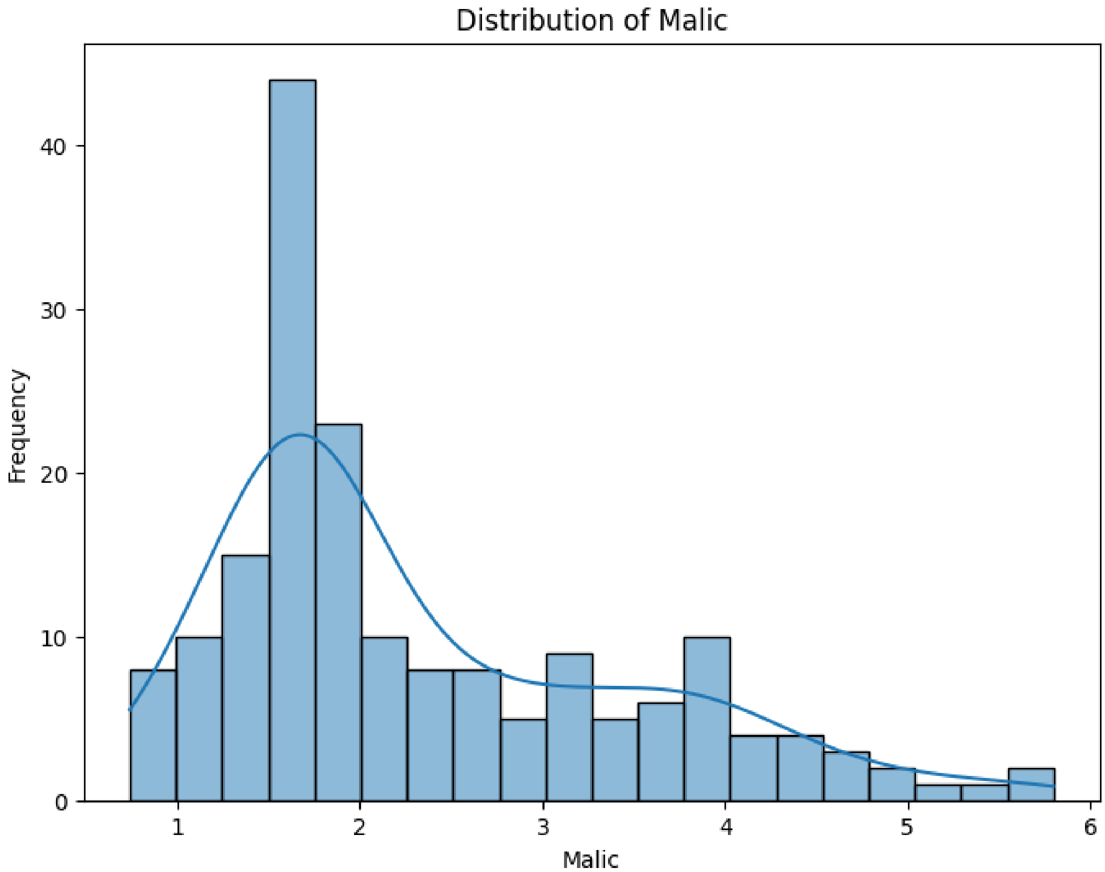
[10]:
```
Type               0
Alcohol            0
Malic              0
Ash                0
Alcalinity         0
Magnesium          0
Phenols            0
Flavanoids         0
Nonflavanoids      0
Proanthocyanins    0
Color              0
Hue                0
Dilution           0
Proline            0
dtype: int64
```

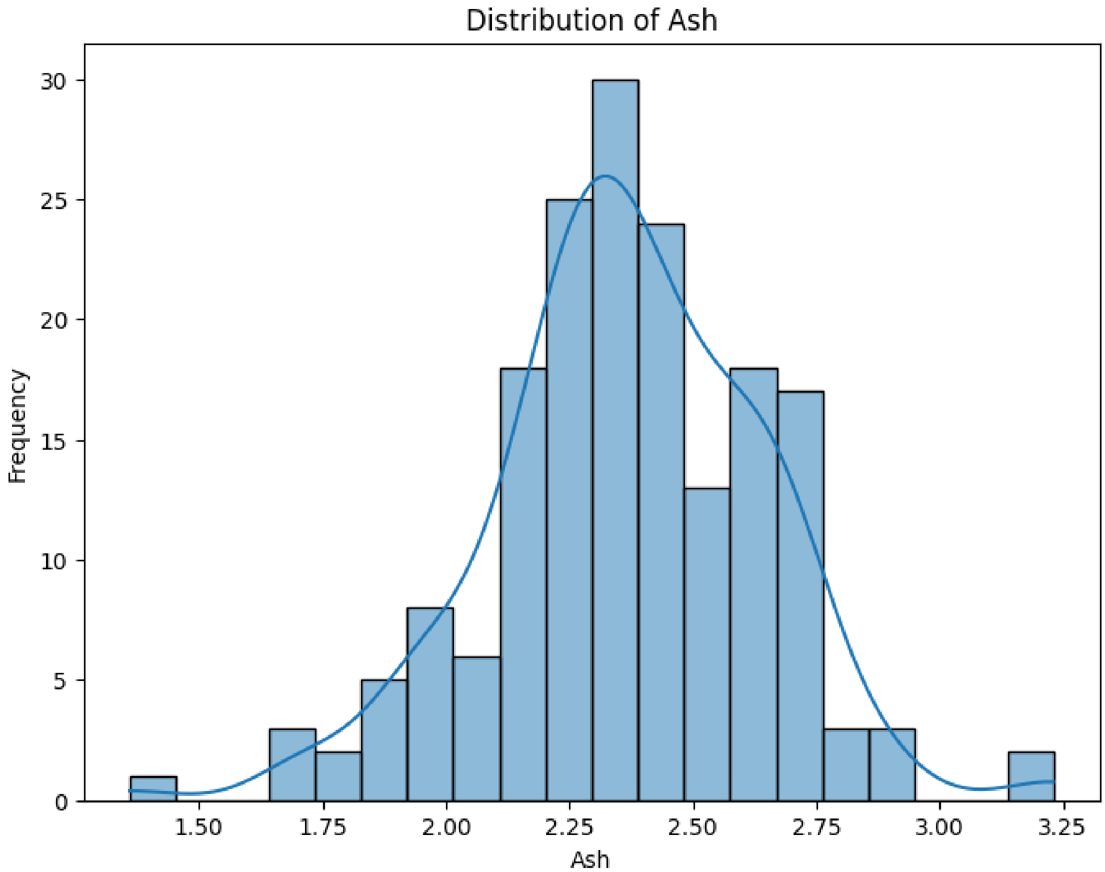
2. Examine the distribution of features using histograms, box plots, or density plots

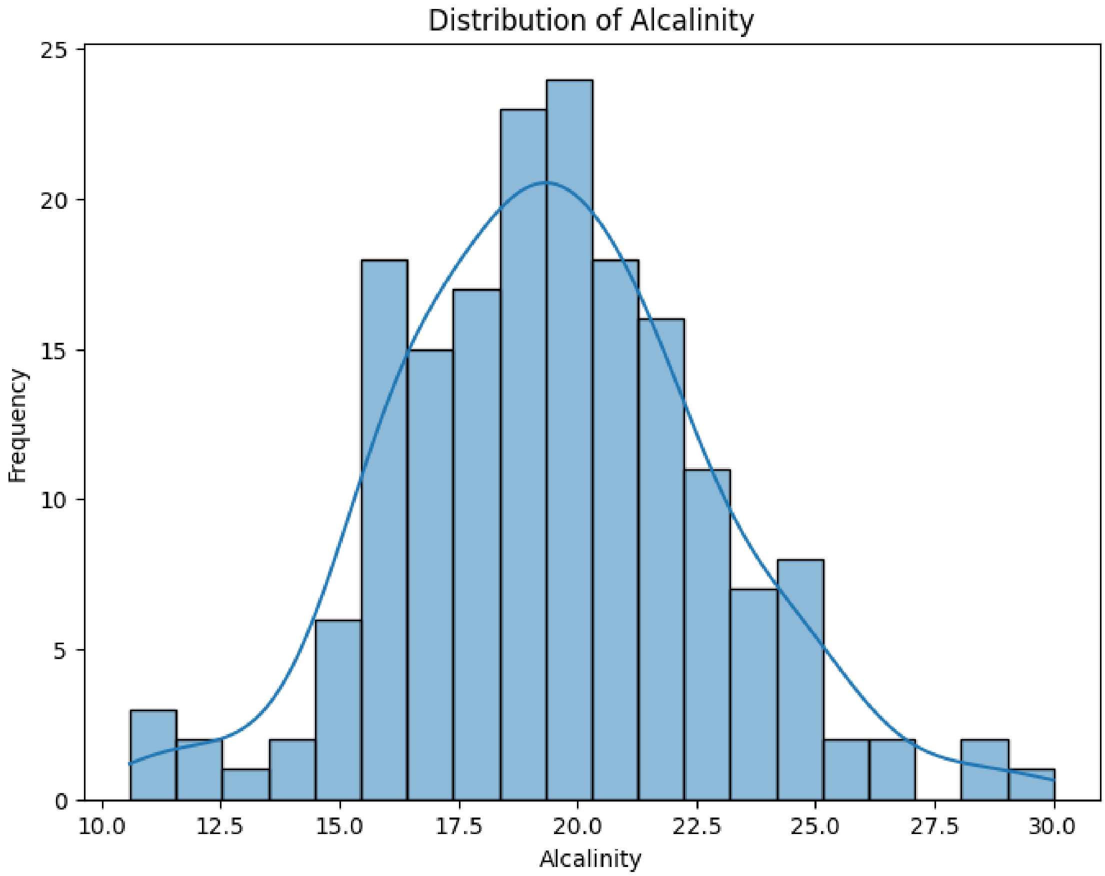
[14]:
```
# Examine the distribution of numerical features
numerical_features = df.select_dtypes(include=['float64', 'int64'])
for column in numerical_features.columns:
    plt.figure(figsize=(8, 6))
    sns.histplot(df[column], bins=20, kde=True)
    plt.title(f'Distribution of {column}')
    plt.xlabel(column)
    plt.ylabel('Frequency')
    plt.show()

# Examine the distribution of categorical features
categorical_features = df.select_dtypes(include=['object'])
for column in categorical_features.columns:
    plt.figure(figsize=(8, 6))
    sns.countplot(data=df, x=column)
    plt.title(f'Count of {column}')
    plt.xlabel(column)
    plt.ylabel('Count')
    plt.xticks(rotation=45)
    plt.show()
```


Distribution of Type
Frequency
Type
0
10
20
30
40
50
60
70
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00

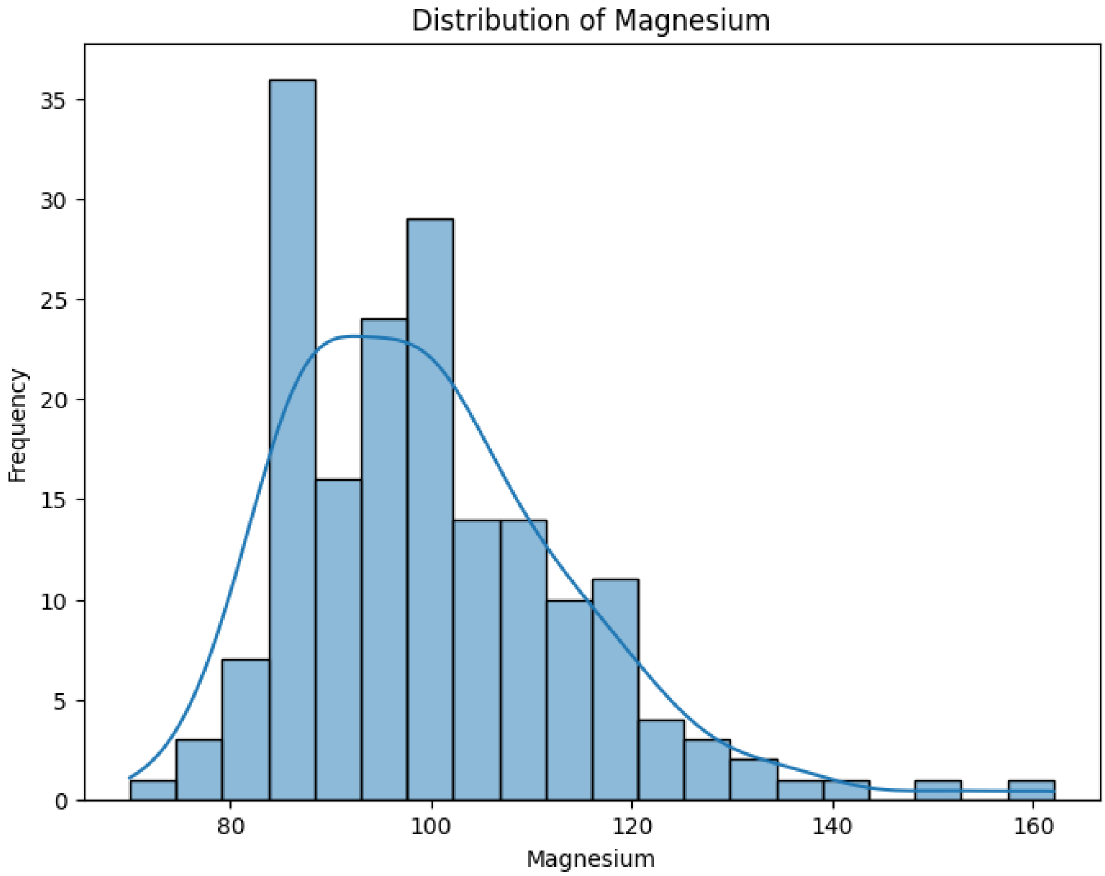

Distribution of Alcohol
Frequency
Alcohol
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
11.0
11.5
12.0
12.5
13.0
13.5
14.0
14.5
15.0

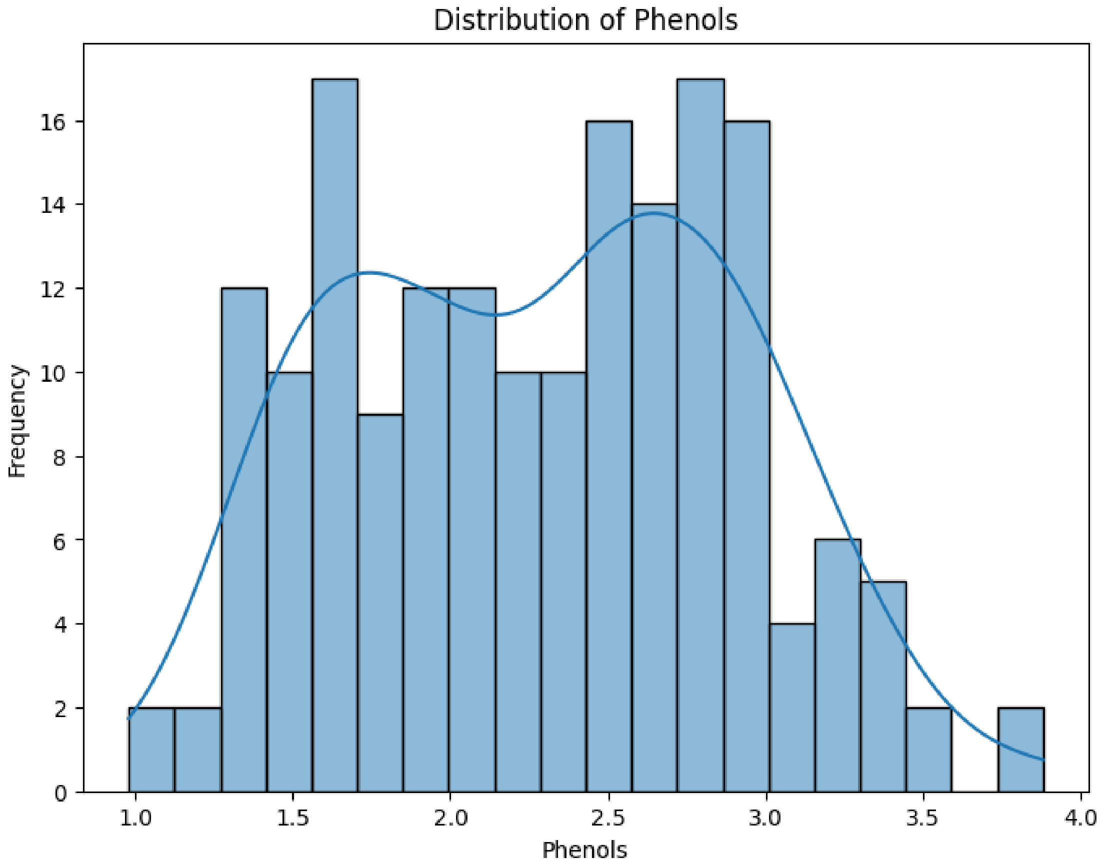

Distribution of Malic
Frequency
Malic
0
10
20
30
40
1
2
3
4
5
6

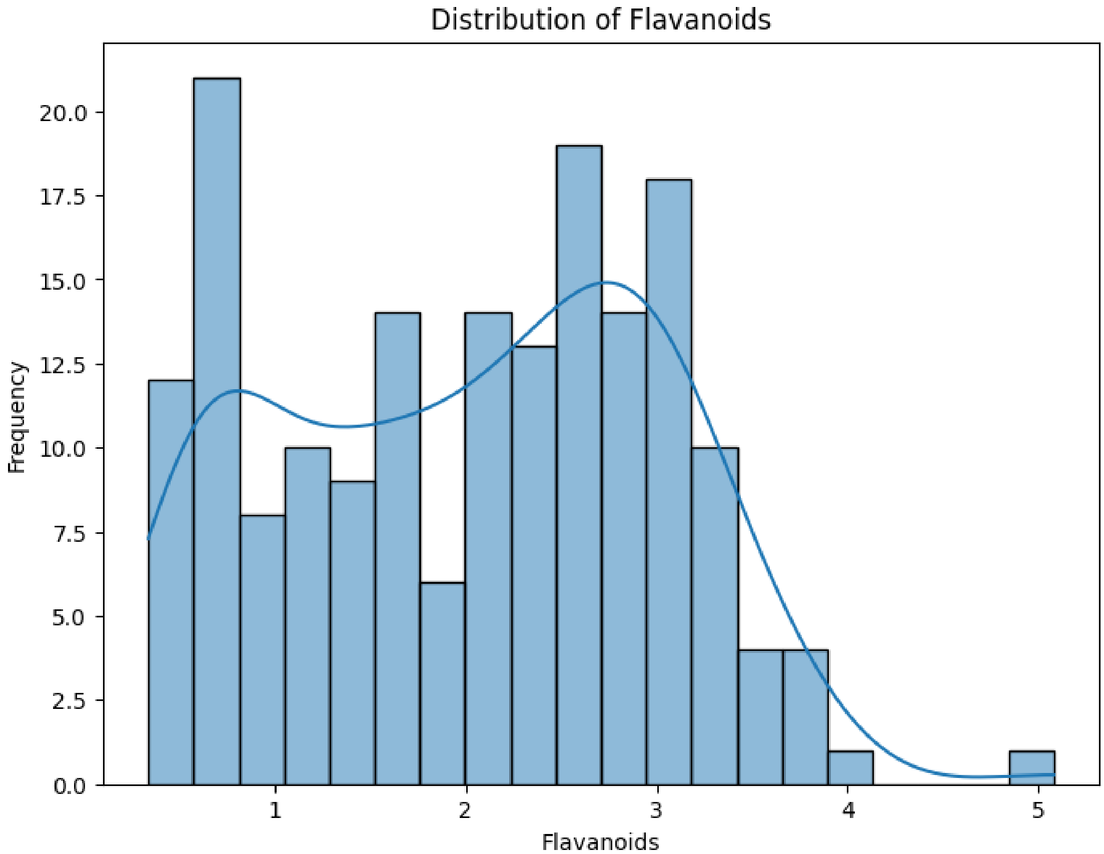

Distribution of Ash
Frequency
Ash
0
5
10
15
20
25
30
1.50
1.75
2.00
2.25
2.50
2.75
3.00
3.25

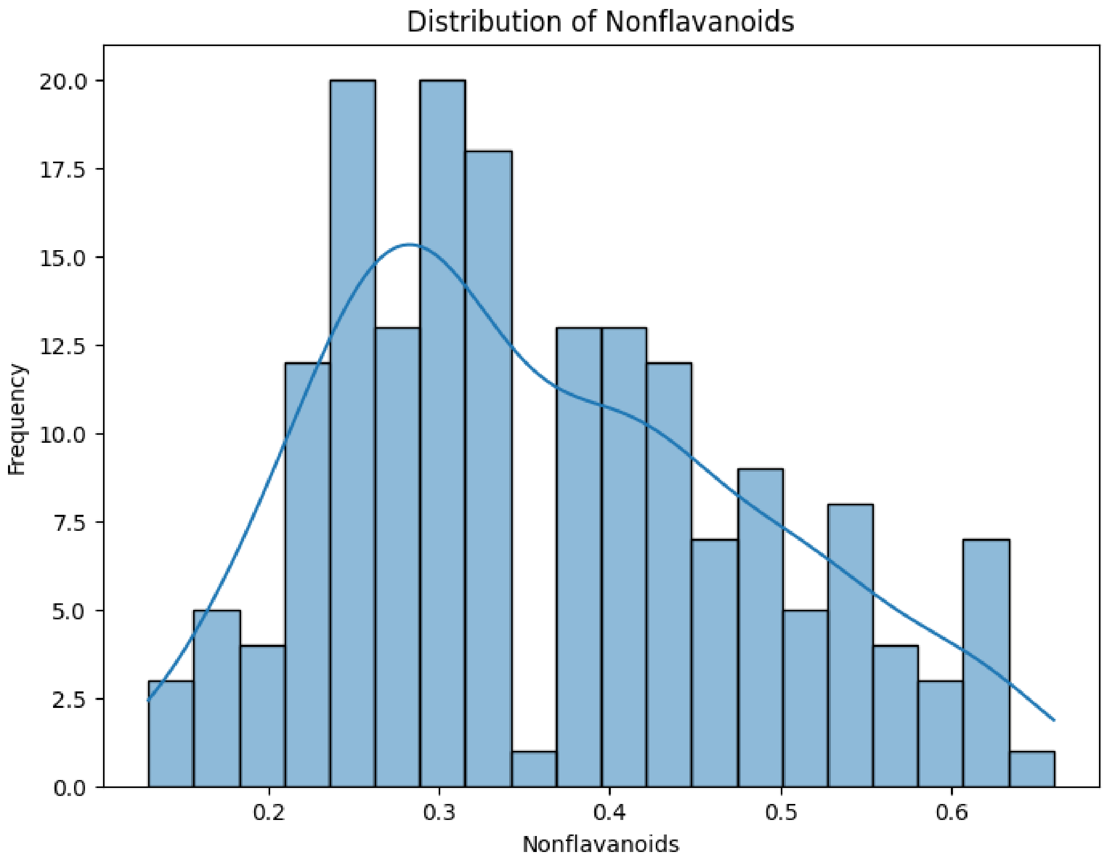


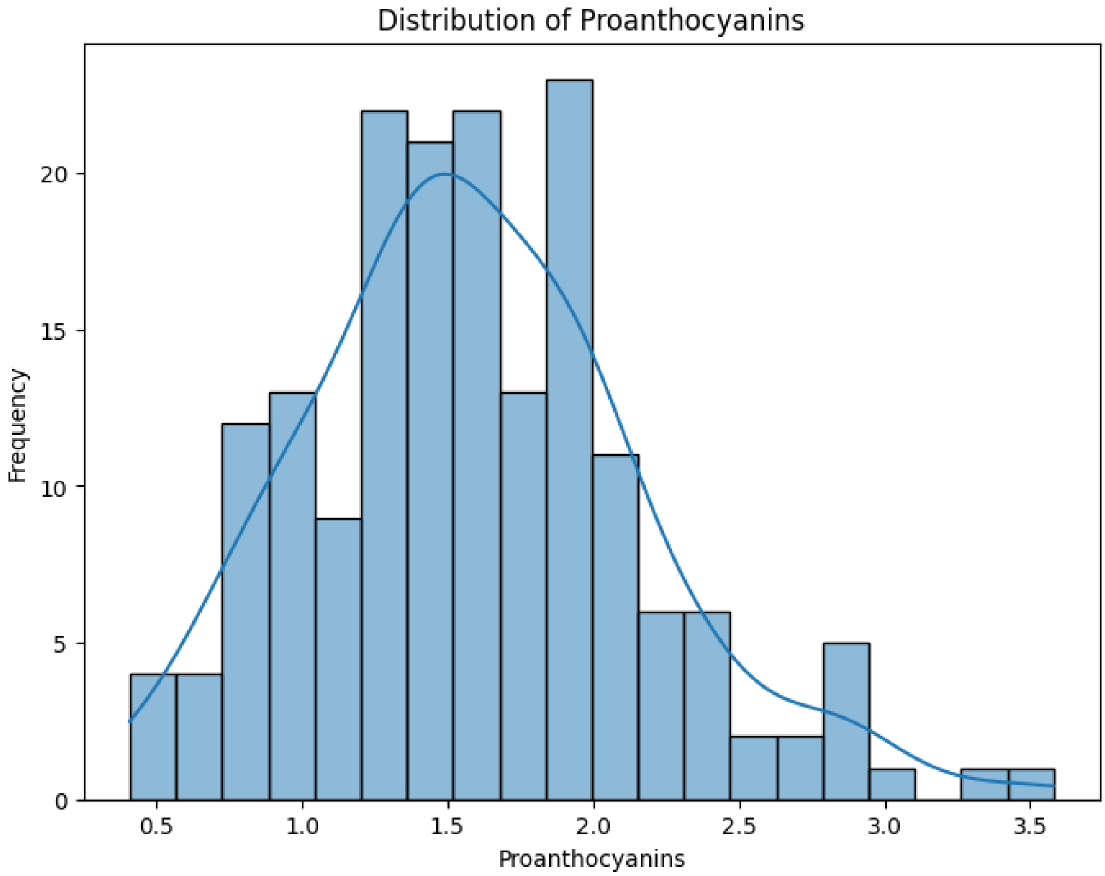
Distribution of Alcalinity
Frequency
Alcalinity
0
5
10
15
20
25
10.0
12.5
15.0
17.5
20.0
22.5
25.0
27.5
30.0


Distribution of Magnesium
Frequency
Magnesium
0
5
10
15
20
25
30
35
80
100
120
140
160

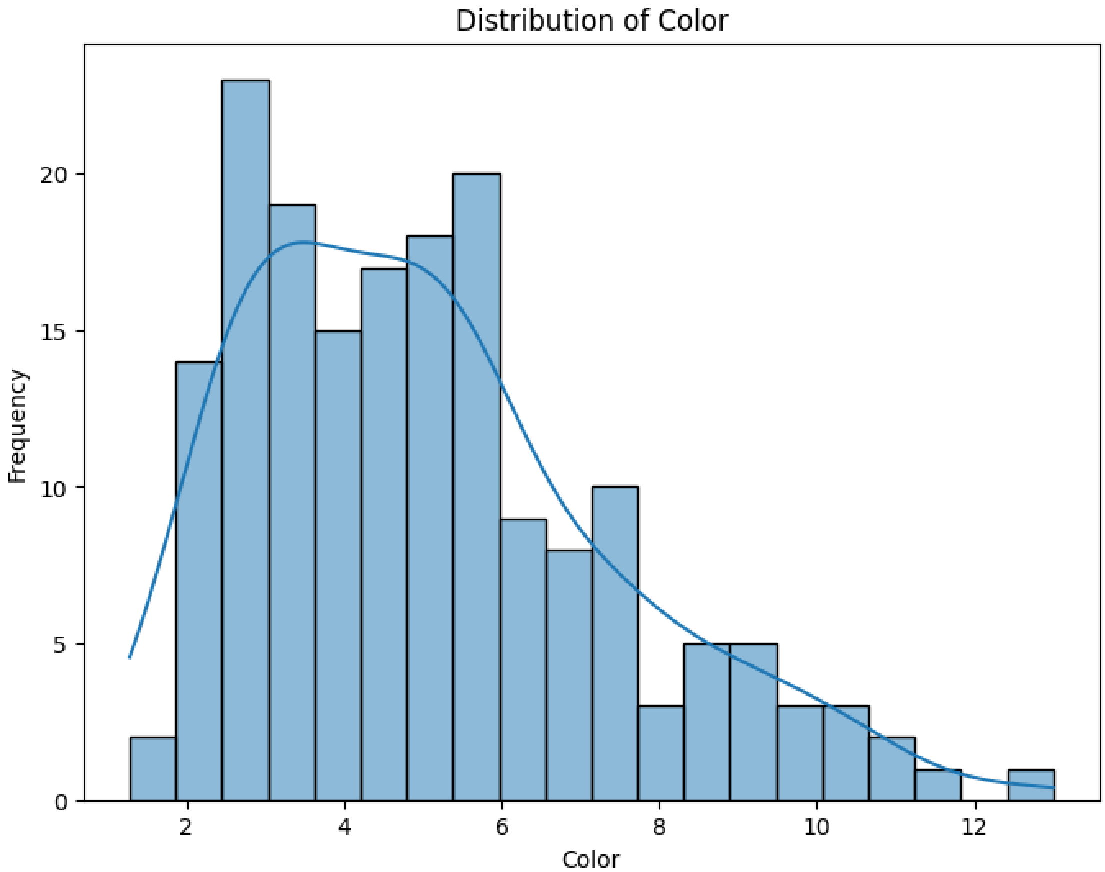

Distribution of Phenols
Frequency
Phenols
0
2
4
6
8
10
12
14
16
1.0
1.5
2.0
2.5
3.0
3.5
4.0

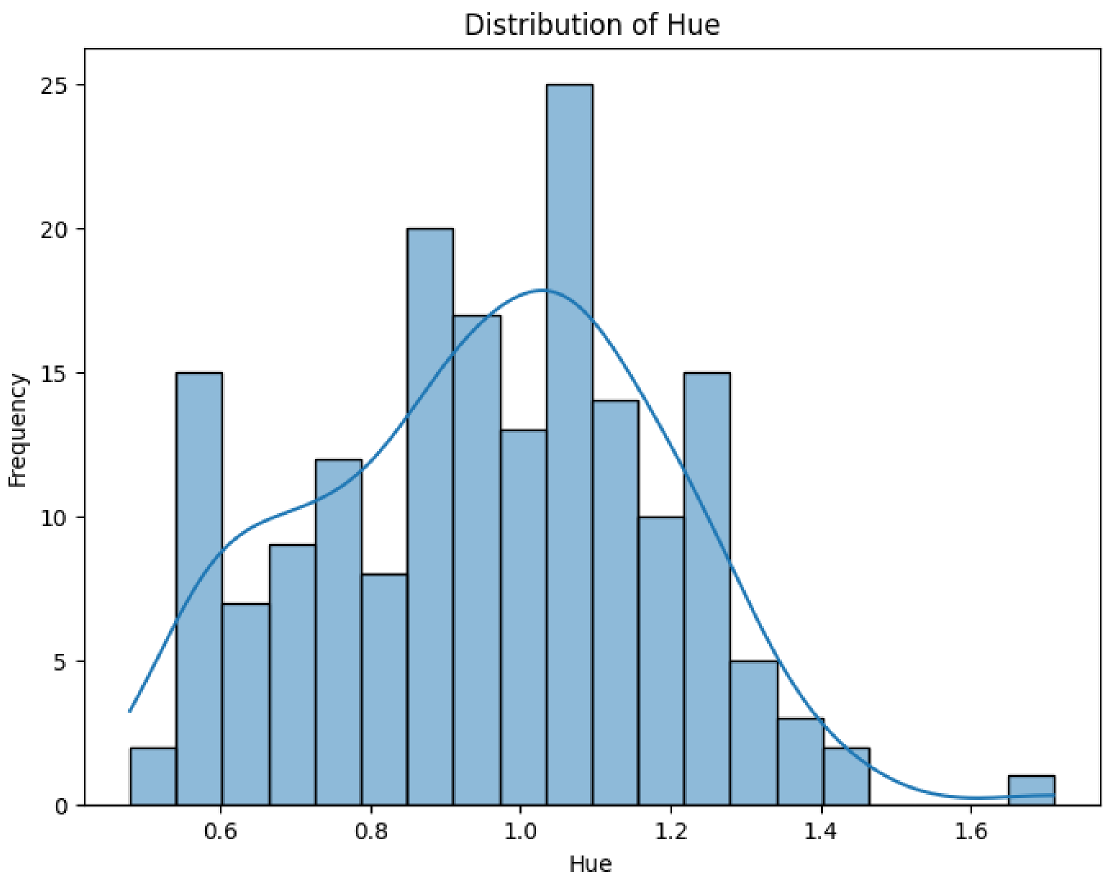

Distribution of Flavanoids
Frequency
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
1
2
3
4
5
Flavanoids

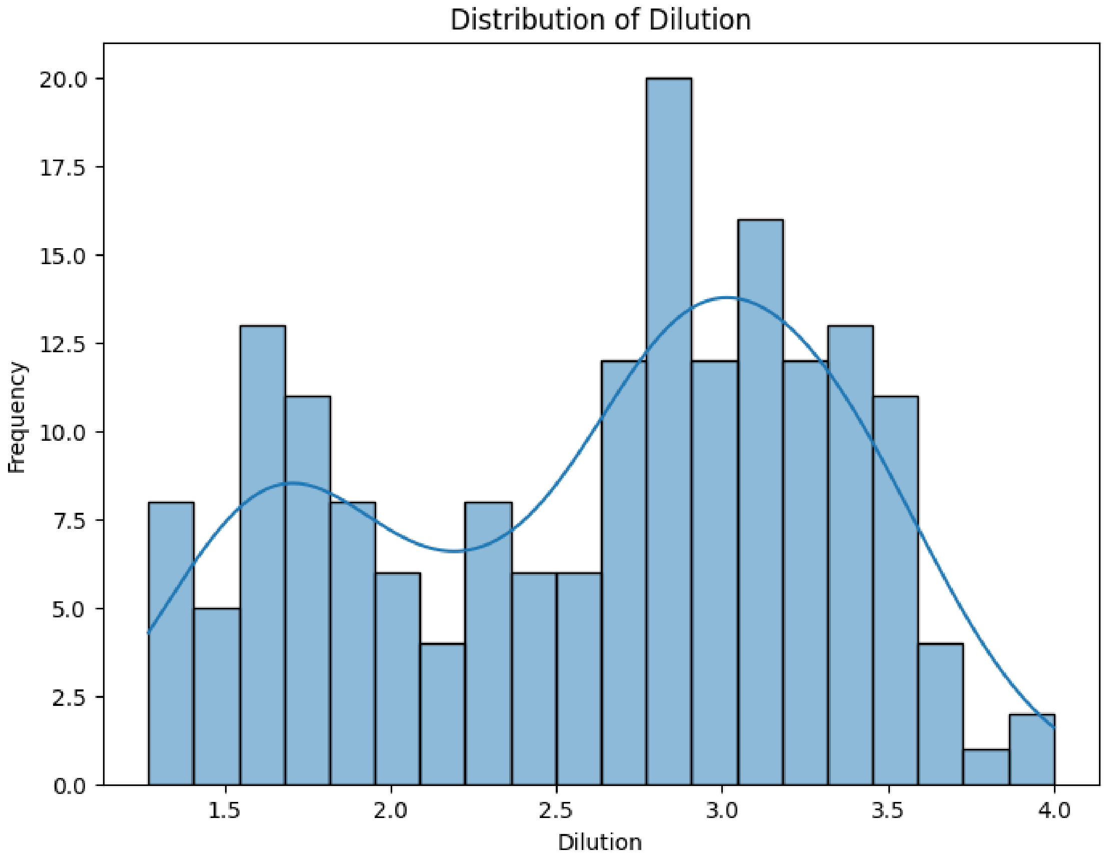

Distribution of Nonflavanoids
Frequency
Nonflavanoids
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
0.2
0.3
0.4
0.5
0.6



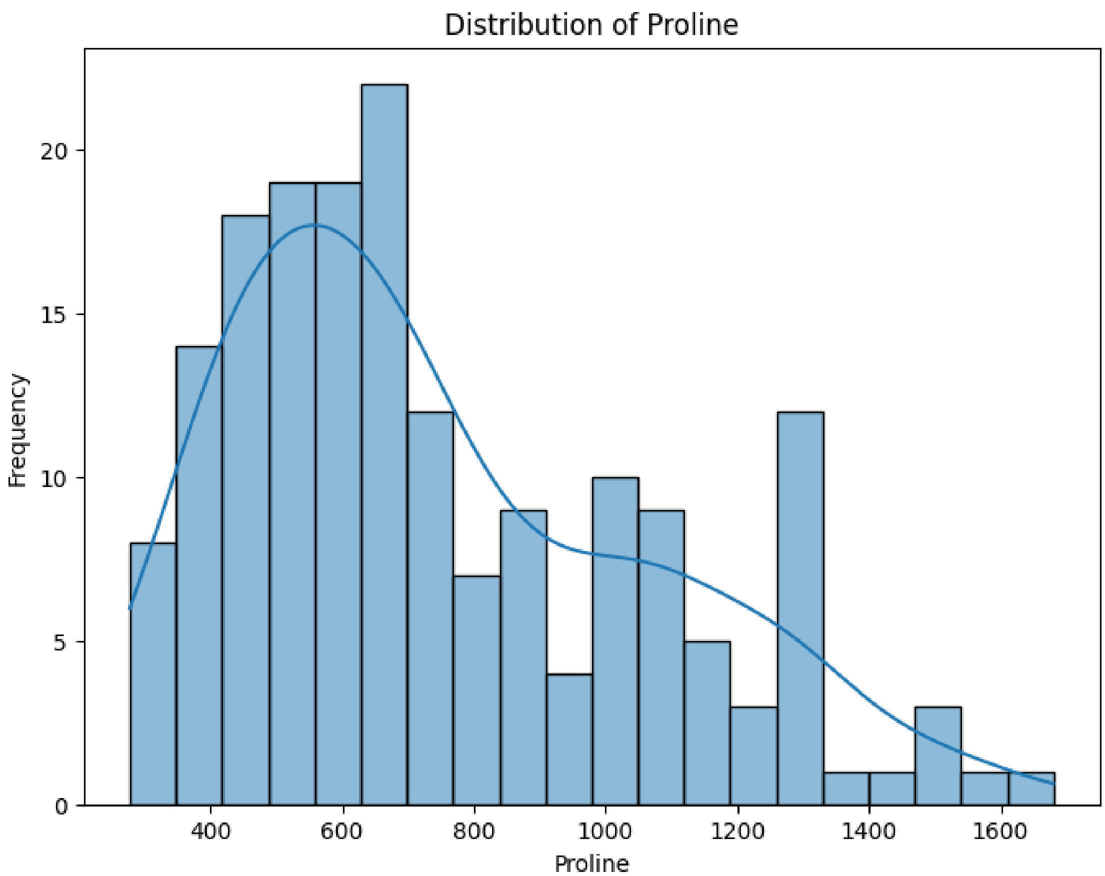
Distribution of Proanthocyanins
Frequency
0
5
10
15
20
0.5
1.0
1.5
2.0
2.5
3.0
3.5
Proanthocyanins


Distribution of Color
Frequency
Color
0
5
10
15
20
2
4
6
8
10
12


Distribution of Hue
Frequency
Hue
0
5
10
15
20
25
0.6
0.8
1.0
1.2
1.4
1.6

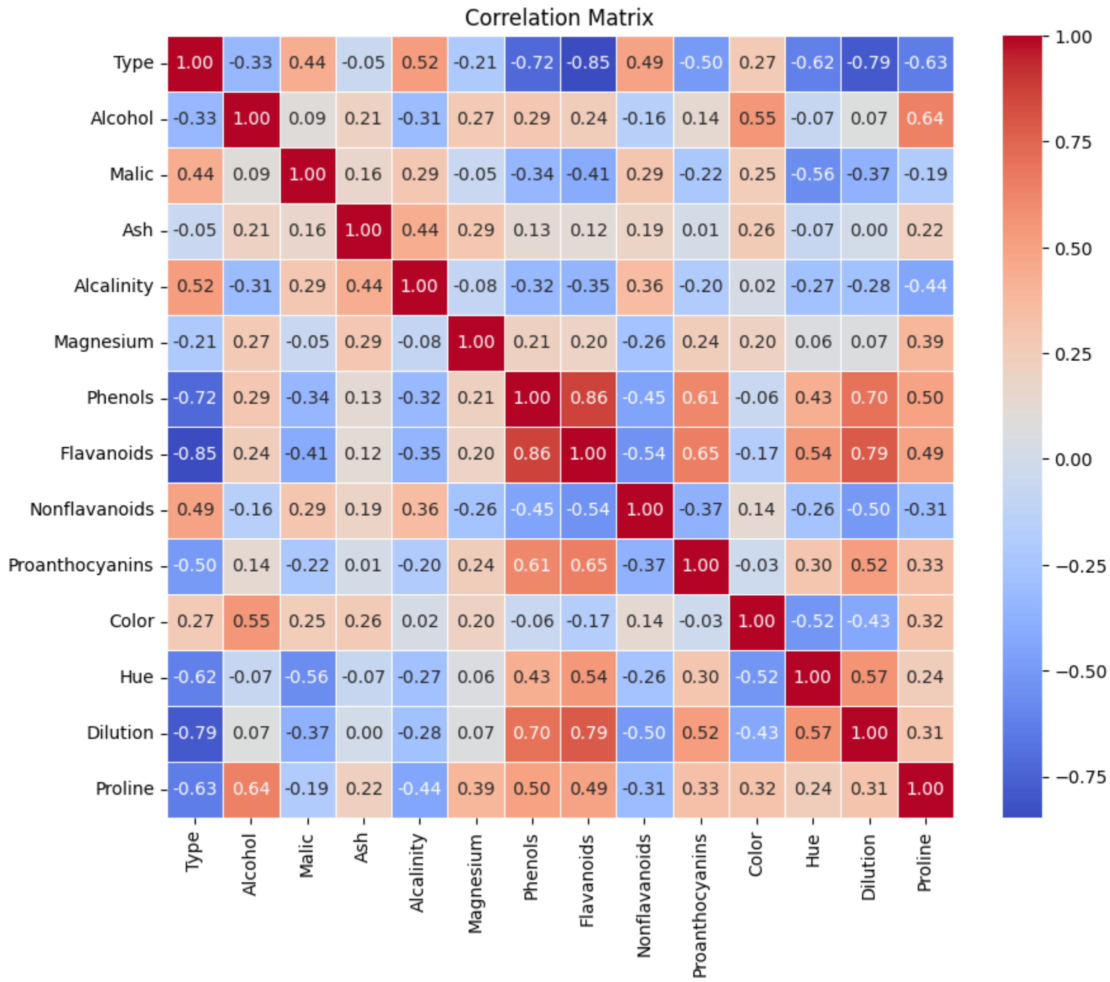


Distribution of Dilution
Frequency
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
1.5
2.0
2.5
3.0
3.5
4.0
Dilution



3.Investigate correlations between features

[15]:
```
correlation_matrix = df.corr()

# Plot correlation matrix heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f",␣
↪linewidths=.5)
plt.title('Correlation Matrix')
plt.show()
```



TASK2. 1.Standardize the features

[17]:
```
#Standardize the numerical features
scaler = StandardScaler()
standardized_data = scaler.fit_transform(df[numerical_features.columns])
```

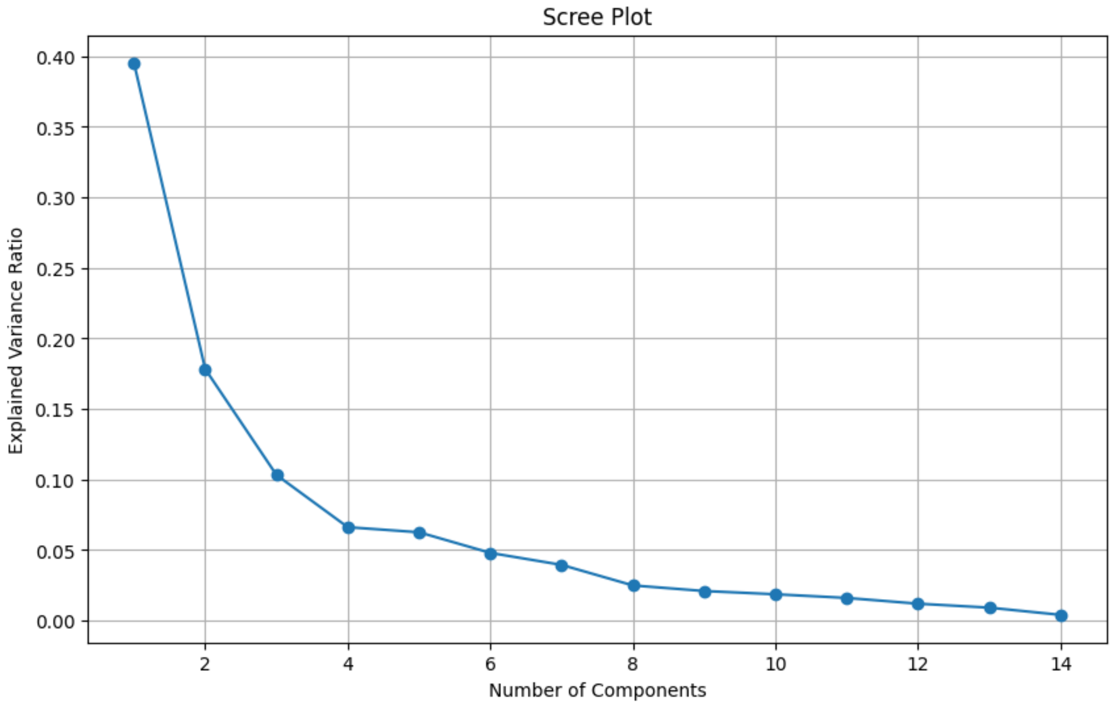
2.Initialize PCA with the desired number of components

[19]:
```
pca = PCA(n_components=None)  # You can specify the number of components or␣
↪leave it as None

# Fit PCA to the standardized data
pca.fit(standardized_data)
```

[19]: PCA()

Determine the optimal number of principal components:

[20]:
```
plt.figure(figsize=(10, 6))
plt.plot(range(1, len(pca.explained_variance_ratio_) + 1), pca.
 ↪explained_variance_ratio_, marker='o', linestyle='-')
plt.title('Scree Plot')
plt.xlabel('Number of Components')
plt.ylabel('Explained Variance Ratio')
plt.grid(True)
plt.show()
```



Transform the original dataset into the principal components:

[21]:
```
principal_components = pca.transform(standardized_data)
```

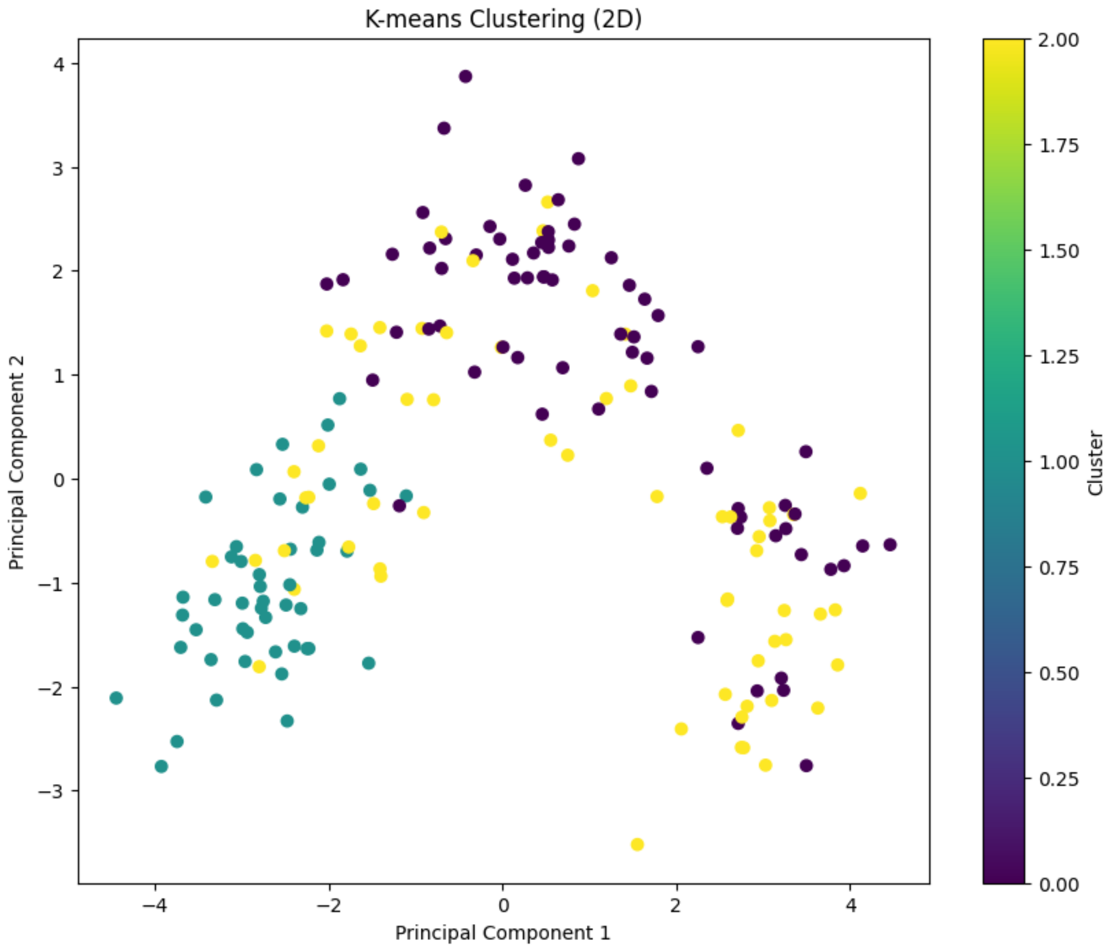
TASK 3:Apply a clustering algorithm (e.g., K-means)

[25]:
```
num_clusters = 3  # You can adjust this number based on your dataset
kmeans = KMeans(n_clusters=num_clusters, random_state=42)
```

[26]:
```
# Fit K-means to the original dataset
kmeans.fit(df[numerical_features.columns])
```

[26]:
```
KMeans(n_clusters=3, random_state=42)
```

[27]:
```
# Transform the data into principal components for visualization
principal_components_for_vis = pca.transform(standardized_data)  # Assuming you␣
↪already have fitted PCA

# Plot the clusters in 2D using the first two principal components
plt.figure(figsize=(10, 8))
plt.scatter(principal_components_for_vis[:, 0], principal_components_for_vis[:,␣
↪1], c=kmeans.labels_, cmap='viridis')
plt.title('K-means Clustering (2D)')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.colorbar(label='Cluster')
plt.show()
```



Evaluate the clustering performance

[29]:
```
# Calculate the silhouette score
silhouette_avg = silhouette_score(df[numerical_features.columns], kmeans.
    ↪labels_)
print("Silhouette Score:", silhouette_avg)
```

```
Silhouette Score: 0.5711220218931753
```

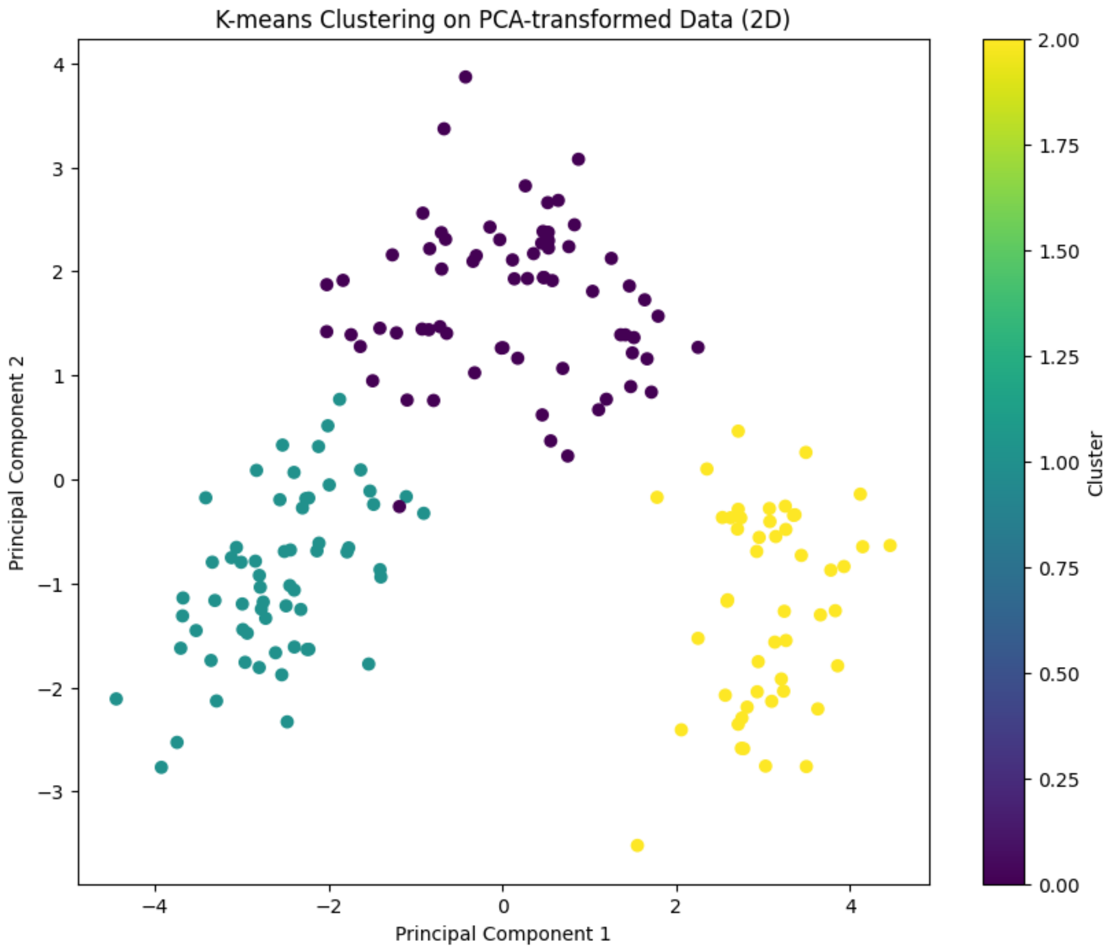
Task 4: Clustering with PCA Data:

[31]:
```
# Initialize K-means with the desired number of clusters
kmeans_pca = KMeans(n_clusters=num_clusters, random_state=42)

# Fit K-means to the PCA-transformed dataset
kmeans_pca.fit(principal_components)
```

[31]:
```
KMeans(n_clusters=3, random_state=42)
```

Visualize the clustering results obtained from PCA-transformed data

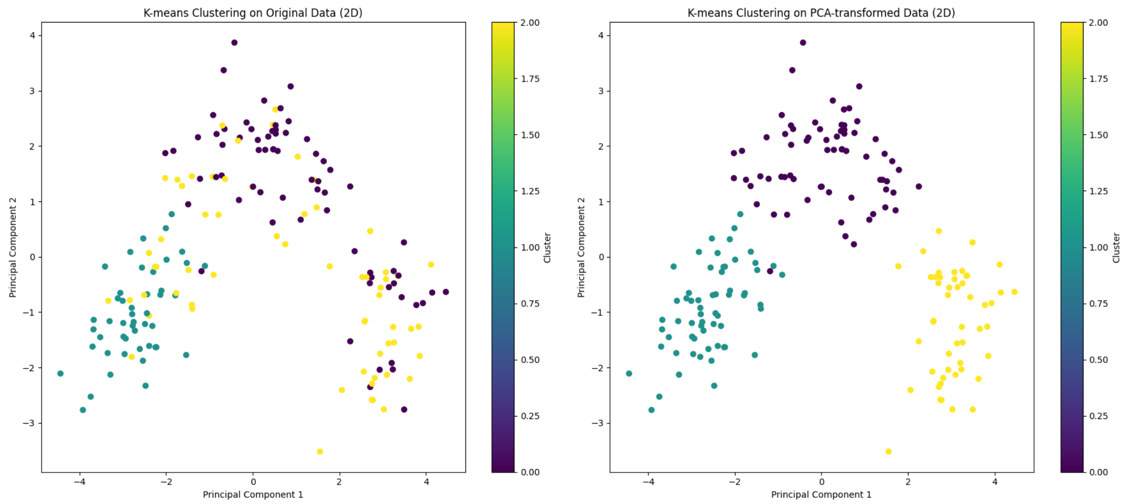
[32]:
```
# Plot the clusters in 2D using the first two principal components
plt.figure(figsize=(10, 8))
plt.scatter(principal_components[:, 0], principal_components[:, 1],␣
    ↪c=kmeans_pca.labels_, cmap='viridis')
plt.title('K-means Clustering on PCA-transformed Data (2D)')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.colorbar(label='Cluster')
plt.show()
```



Compare the clustering results from PCA-transformed data with those from the original dataset

[33]:
```
# Plot the clusters in 2D using the first two principal components for both␣
↪datasets
plt.figure(figsize=(18, 8))

plt.subplot(1, 2, 1)
plt.scatter(principal_components_for_vis[:, 0], principal_components_for_vis[:,␣
↪1], c=kmeans.labels_, cmap='viridis')
plt.title('K-means Clustering on Original Data (2D)')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.colorbar(label='Cluster')

plt.subplot(1, 2, 2)
plt.scatter(principal_components[:, 0], principal_components[:, 1],␣
↪c=kmeans_pca.labels_, cmap='viridis')
```

```
plt.title('K-means Clustering on PCA-transformed Data (2D)')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.colorbar(label='Cluster')

plt.tight_layout()
plt.show()
```



TASK5