l5

August 27, 2024

0.1 ML_LAB_5

Name: Imaad Hajwane
SRN: 202101132
Roll No: 21
Program: Computer Engineering
Year: Last year
Div: A
Subject: ML

Q. Write a program to implement K-Nearest Mean clustering Algorithm and find the value of k using WCSS

import libraries

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.metrics import classification_report, roc_auc_score, roc_curve
from sklearn.datasets import make_blobs
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
from sklearn.metrics import auc
from sklearn.metrics import precision_recall_curve
from scipy.spatial import distance
from sklearn.metrics import silhouette_score
from sklearn.metrics import davies_bouldin_score
from sklearn.metrics import calinski_harabasz_score
from sklearn.metrics import adjusted_rand_score
```

read csv file

[2]:
```
# Read the CSV file
df = pd.read_csv("Almond.csv")
```

basic informations

[3]:
```
# Display the basic information about the dataset
print("Dataset Info:")
print(df.info())

# Display the first few rows of the dataset
print("\nDataset Head:")
print(df.head())

# Display the statistical summary of the dataset
print("\nDataset Description:")
print(df.describe())
```

```
Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2803 entries, 0 to 2802
Data columns (total 14 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Unnamed: 0                2803 non-null   int64
 1   Length (major axis)       1946 non-null   float64
 2   Width (minor axis)        1861 non-null   float64
 3   Thickness (depth)         1799 non-null   float64
 4   Area                      2803 non-null   float64
 5   Perimeter                 2803 non-null   float64
 6   Roundness                 1946 non-null   float64
 7   Solidity                  2803 non-null   float64
 8   Compactness               2803 non-null   float64
 9   Aspect Ratio              1004 non-null   float64
 10  Eccentricity              1004 non-null   float64
 11  Extent                    2803 non-null   float64
 12  Convex hull(convex area)  2803 non-null   float64
 13  Type                      2803 non-null   object
dtypes: float64(12), int64(1), object(1)
memory usage: 306.7+ KB
None

Dataset Head:
   Unnamed: 0  Length (major axis)  Width (minor axis)  Thickness (depth)  \
0           0                  NaN          227.940628         127.759132
1           1                  NaN          234.188126         128.199509
2           2                  NaN          229.418610         125.796547
3           3                  NaN          232.763153         125.918808
4           4                  NaN          230.150742         107.253448

      Area   Perimeter  Roundness  Solidity  Compactness  Aspect Ratio  \
0  22619.0  643.813269        NaN  0.973384     1.458265           NaN
```

```
1  23038.0  680.984841        NaN  0.957304     1.601844           NaN
2  22386.5  646.943212        NaN  0.967270     1.487772           NaN
3  22578.5  661.227483        NaN  0.965512     1.540979           NaN
4  19068.0  624.842706        NaN  0.951450     1.629395           NaN

   Eccentricity    Extent  Convex hull(convex area)   Type
0           NaN  0.681193                   23237.5  MAMRA
1           NaN  0.656353                   24065.5  MAMRA
2           NaN  0.683620                   23144.0  MAMRA
3           NaN  0.685360                   23385.0  MAMRA
4           NaN  0.714800                   20041.0  MAMRA

Dataset Description:
        Unnamed: 0  Length (major axis)  Width (minor axis)  \
count  2803.000000          1946.000000         1861.000000
mean   1401.000000           290.609274          171.025915
std     809.300727            62.719433           29.916529
min       0.000000           151.335266           88.050529
25%     700.500000           245.966293          149.453659
50%    1401.000000           279.879883          170.168365
75%    2101.500000           330.508575          190.640427
max    2802.000000           515.352478          258.569794

       Thickness (depth)          Area    Perimeter    Roundness     Solidity  \
count        1799.000000   2803.000000  2803.000000  1946.000000  2803.000000
mean          109.705378  26511.117374   743.863770     0.470466     0.955828
std            18.940597  13782.561344   230.632076     0.118673     0.039596
min            59.494278   6037.000000   311.563489     0.173748     0.718772
25%            97.091682  16211.500000   571.730009     0.384810     0.944579
50%           110.280136  23440.500000   707.487369     0.472718     0.970422
75%           121.392773  33451.000000   878.896530     0.577553     0.981484
max           181.845200  89282.000000  1864.947387     0.697293     0.992889

       Compactness  Aspect Ratio  Eccentricity       Extent  \
count  2803.000000   1004.000000   1004.000000  2803.000000
mean      1.825233      1.753216      0.813114     0.724587
std       0.794058      0.206616      0.041312     0.047474
min       1.164469      1.400082      0.699897     0.454538
25%       1.357398      1.612490      0.784476     0.701673
50%       1.576412      1.705716      0.810120     0.733720
75%       1.965953      1.833339      0.838141     0.757551
max       9.660057      2.731251      0.930563     0.845813

       Convex hull(convex area)
count               2803.000000
mean               27696.218159
std                14237.347610
min                 6355.000000
```

```
25%                 17088.500000
50%                 24589.000000
75%                 34863.250000
max                 90642.500000
```

identification of missing values and filling the missing values

[4]:
```
df.isnull()
```

[4]:
```
      Unnamed: 0  Length (major axis)  Width (minor axis)  Thickness (depth)  \
0          False                 True               False              False
1          False                 True               False              False
2          False                 True               False              False
3          False                 True               False              False
4          False                 True               False              False
…            …                    …                   …                  …
2798       False                 True               False              False
2799       False                 True               False              False
2800       False                 True               False              False
2801       False                 True               False              False
2802       False                False               False               True

       Area  Perimeter  Roundness  Solidity  Compactness  Aspect Ratio  \
0     False      False       True     False        False          True
1     False      False       True     False        False          True
2     False      False       True     False        False          True
3     False      False       True     False        False          True
4     False      False       True     False        False          True
…       …          …          …        …            …              …
2798  False      False       True     False        False          True
2799  False      False       True     False        False          True
2800  False      False       True     False        False          True
2801  False      False       True     False        False          True
2802  False      False      False     False        False         False

      Eccentricity  Extent  Convex hull(convex area)   Type
0             True   False                     False  False
1             True   False                     False  False
2             True   False                     False  False
3             True   False                     False  False
4             True   False                     False  False
…              …      …                         …      …
2798          True   False                     False  False
2799          True   False                     False  False
2800          True   False                     False  False
2801          True   False                     False  False
2802         False   False                     False  False
```

```
[2803 rows x 14 columns]
```

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

[5]:
```
Unnamed: 0                    0
Length (major axis)         857
Width (minor axis)          942
Thickness (depth)          1004
Area                          0
Perimeter                     0
Roundness                   857
Solidity                      0
Compactness                   0
Aspect Ratio               1799
Eccentricity               1799
Extent                        0
Convex hull(convex area)      0
Type                          0
dtype: int64
```

[6]:
```
# Replace missing values with 0
df.fillna(0, inplace=True)
```

[7]:
```
df.isnull().sum()
```

[7]:
```
Unnamed: 0                  0
Length (major axis)         0
Width (minor axis)          0
Thickness (depth)           0
Area                        0
Perimeter                   0
Roundness                   0
Solidity                    0
Compactness                 0
Aspect Ratio                0
Eccentricity                0
Extent                      0
Convex hull(convex area)    0
Type                        0
dtype: int64
```

selection of relevantfeatures in the dataset

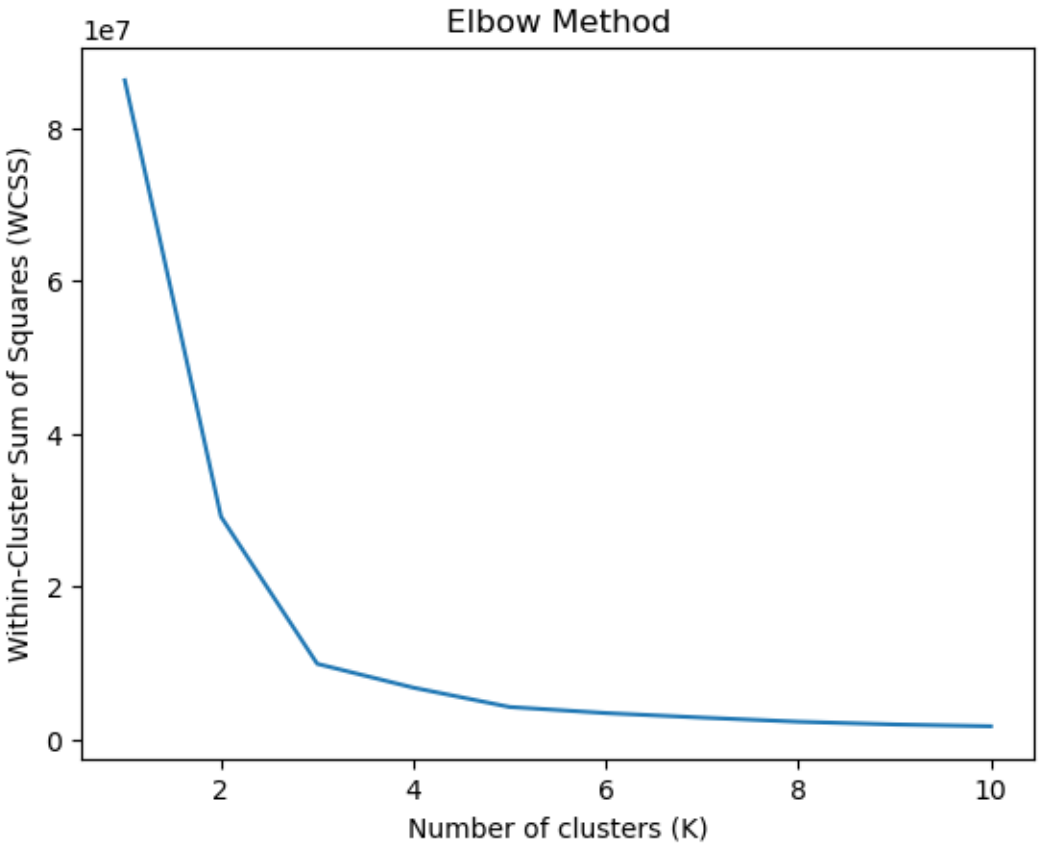
[8]:
```
# Select relevant features for clustering
X = df[['Length (major axis)', 'Width (minor axis)', 'Thickness (depth)']]
```

Implementation of K-Means clustering using WCSS

[9]:
```
# Implement K-Means clustering and find the optimal value of K using WCSS
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10,␣
 ↪random_state=0)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)
```

Elbow Plot

[10]:
```
# Plot the Elbow curve to find the optimal value of K
plt.plot(range(1, 11), wcss)
plt.title('Elbow Method')
plt.xlabel('Number of clusters (K)')
plt.ylabel('Within-Cluster Sum of Squares (WCSS)')
plt.show()
```



Identification of Elbow Point

[11]:
```
from kneed import KneeLocator

k_range = list(range(1, 11))

# Use kneedle to find the elbow
kneedle = KneeLocator(k_range, wcss, curve='convex', direction='decreasing')

# Get the optimal k value
optimal_k = kneedle.elbow
print(f"The optimal number of clusters is: {optimal_k}")
```

```
The optimal number of clusters is: 3
```

[12]:
```
# Implementing K-Means clustering with the optimal value of K
k = 3
kmeans = KMeans(n_clusters=k, init='k-means++', max_iter=300, n_init=10,␣
 ↪random_state=0)
pred_y = kmeans.fit_predict(X)
```

Evaluation Score (Silhouette score)

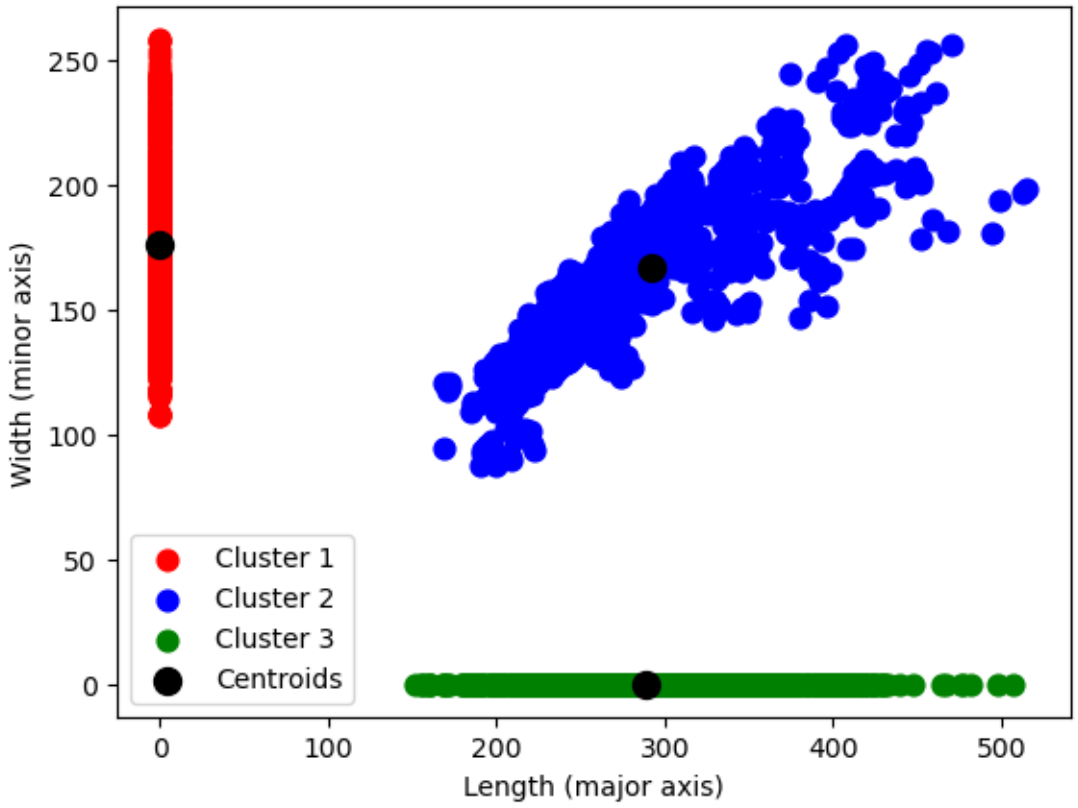
[13]:
```
from sklearn.metrics import silhouette_score

# Calculate the Silhouette Score
silhouette_avg = silhouette_score(X, pred_y)
print(f"Silhouette Score: {silhouette_avg}")
```

```
Silhouette Score: 0.7121232447493592
```

Cluster Plot K=3

[14]:
```
# Convert DataFrame to NumPy array
X_array = X.to_numpy()

# Plotting the clusters and centroids
plt.scatter(X_array[pred_y == 0, 0], X_array[pred_y == 0, 1], s=60, c='red',␣
 ↪label='Cluster 1')
plt.scatter(X_array[pred_y == 1, 0], X_array[pred_y == 1, 1], s=60, c='blue',␣
 ↪label='Cluster 2')
plt.scatter(X_array[pred_y == 2, 0], X_array[pred_y == 2, 1], s=60, c='green',␣
 ↪label='Cluster 3')
# Add more scatter plots for additional clusters if needed

plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1],␣
 ↪s=100, c='black', label='Centroids')
plt.xlabel('Length (major axis)')
plt.ylabel('Width (minor axis)')
plt.legend()
```

```
plt.show()
```

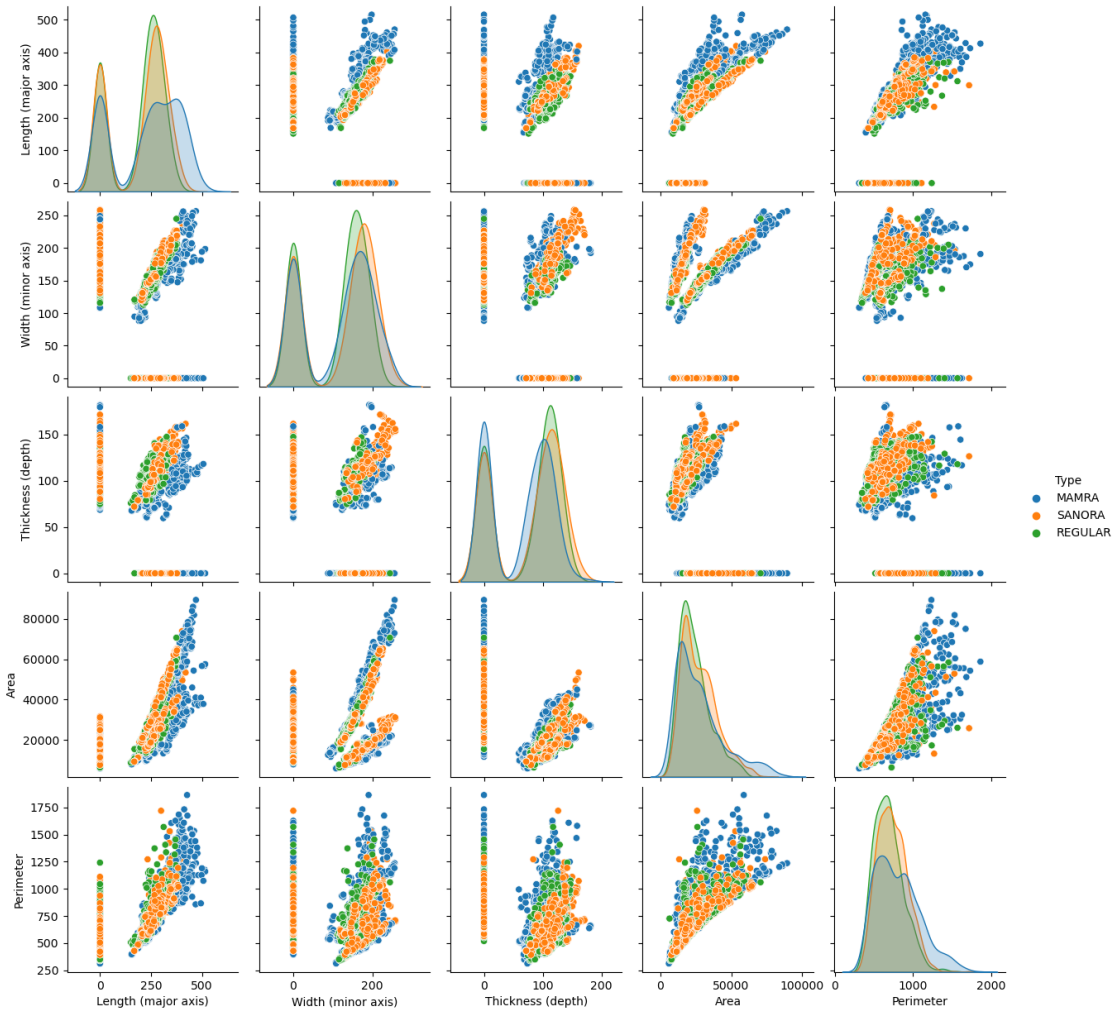


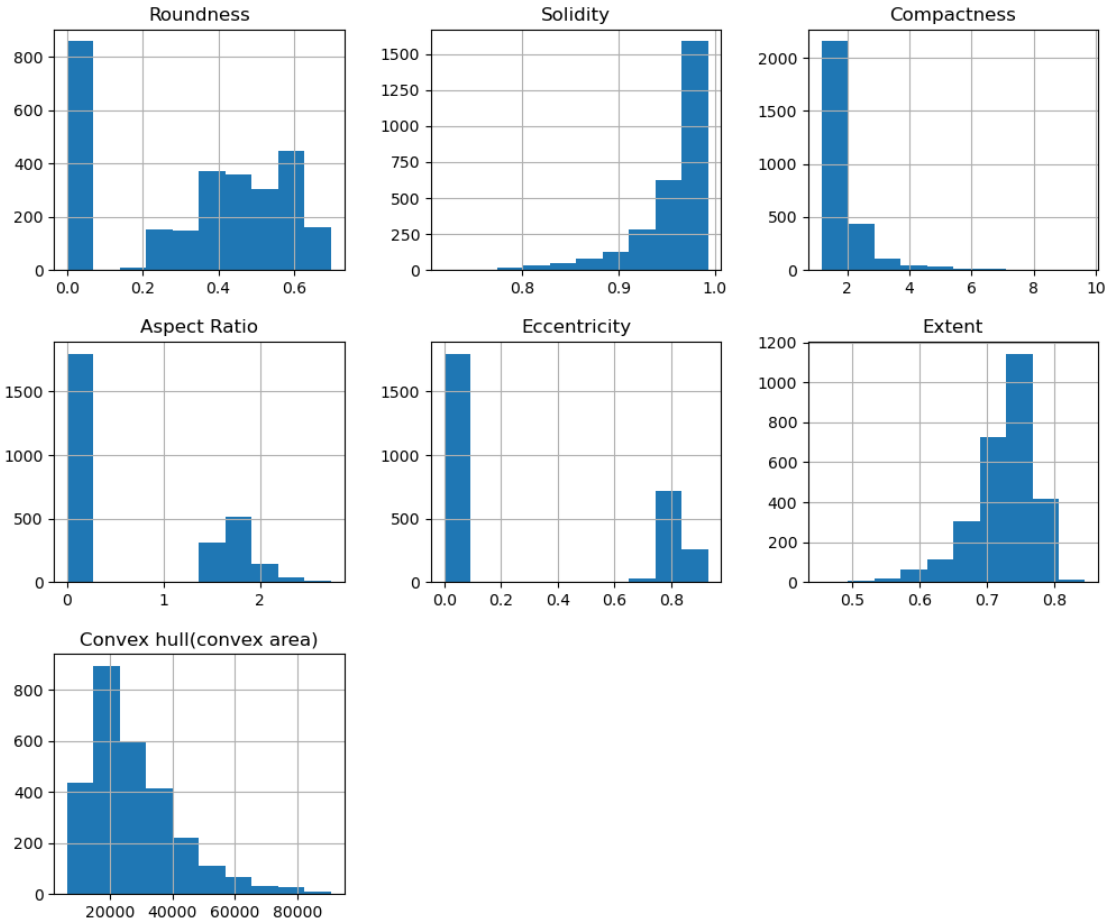
PLOTS

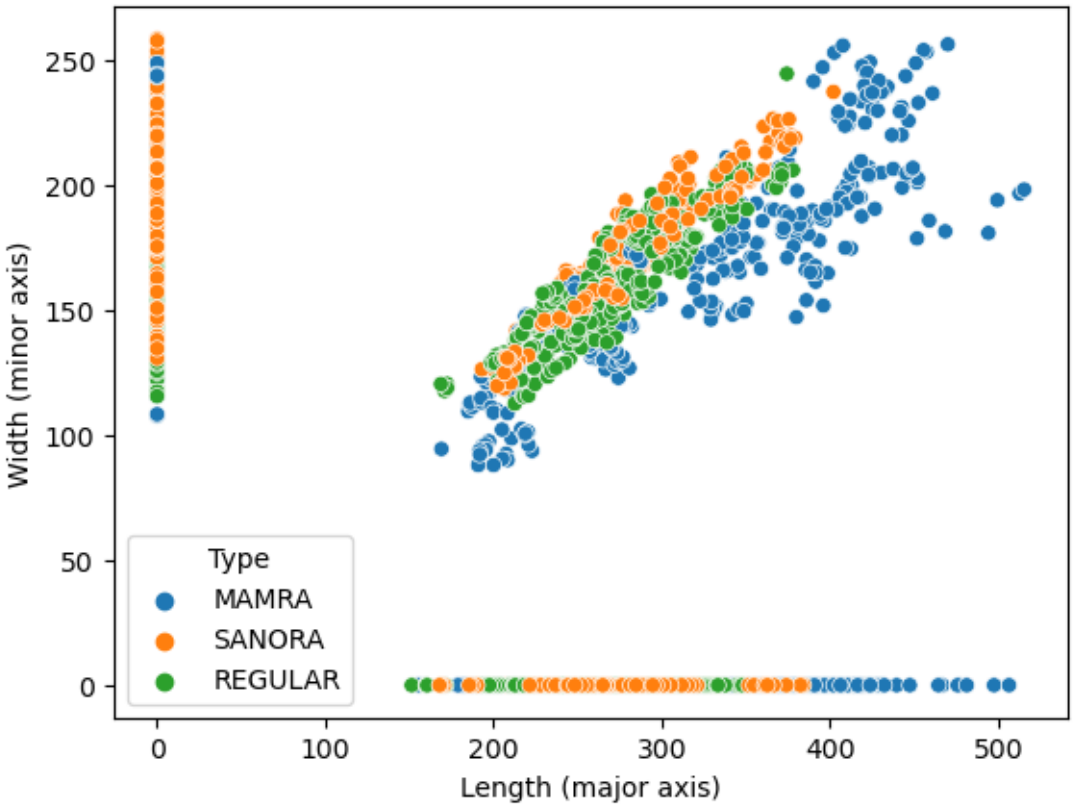
[15]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Pairplot
sns.pairplot(df, vars=["Length (major axis)", "Width (minor axis)", "Thickness␣
 ↪(depth)", "Area", "Perimeter"], hue='Type')
plt.show()

# Histograms
df.hist(column=["Roundness", "Solidity", "Compactness", "Aspect Ratio",␣
 ↪"Eccentricity", "Extent", "Convex hull(convex area)"], figsize=(12, 10))
plt.show()

# Scatter Plots
sns.scatterplot(data=df, x="Length (major axis)", y="Width (minor axis)",␣
 ↪hue="Type")
```

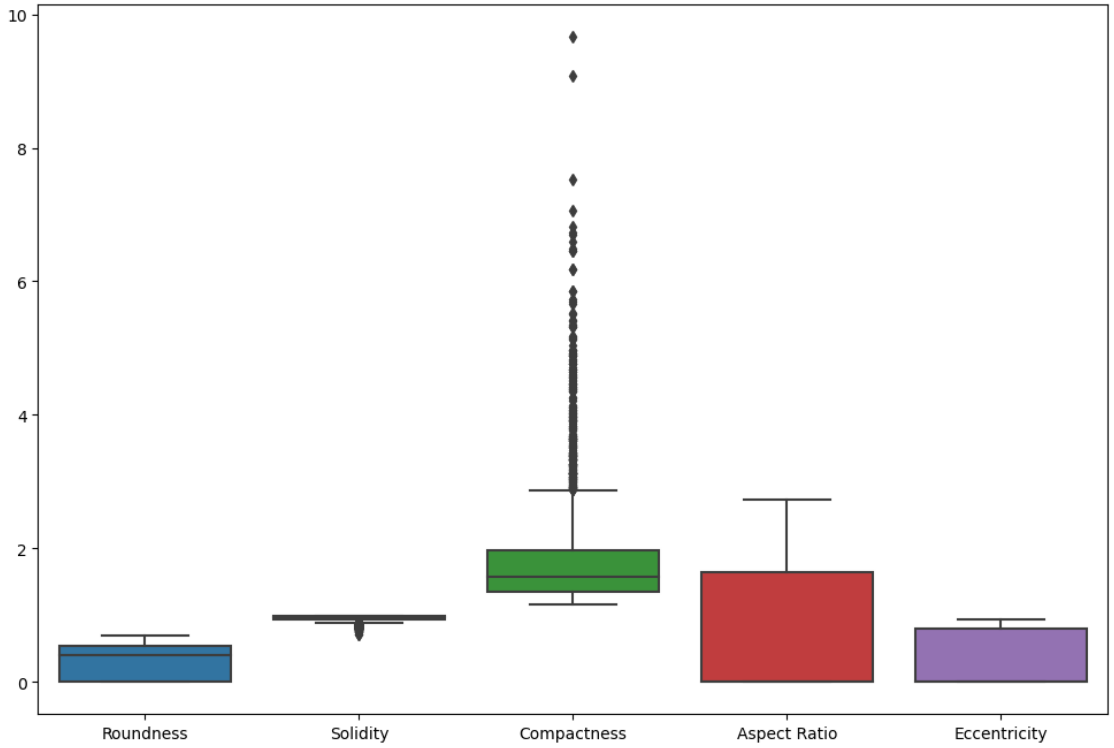
```
plt.show()

# Box Plots
plt.figure(figsize=(12, 8))
sns.boxplot(data=df[['Roundness', 'Solidity', 'Compactness', 'Aspect Ratio',␣
 ↪'Eccentricity']])
plt.show()

# Count Plot
plt.figure(figsize=(8, 6))
sns.countplot(data=df, x='Type')
plt.show()
```

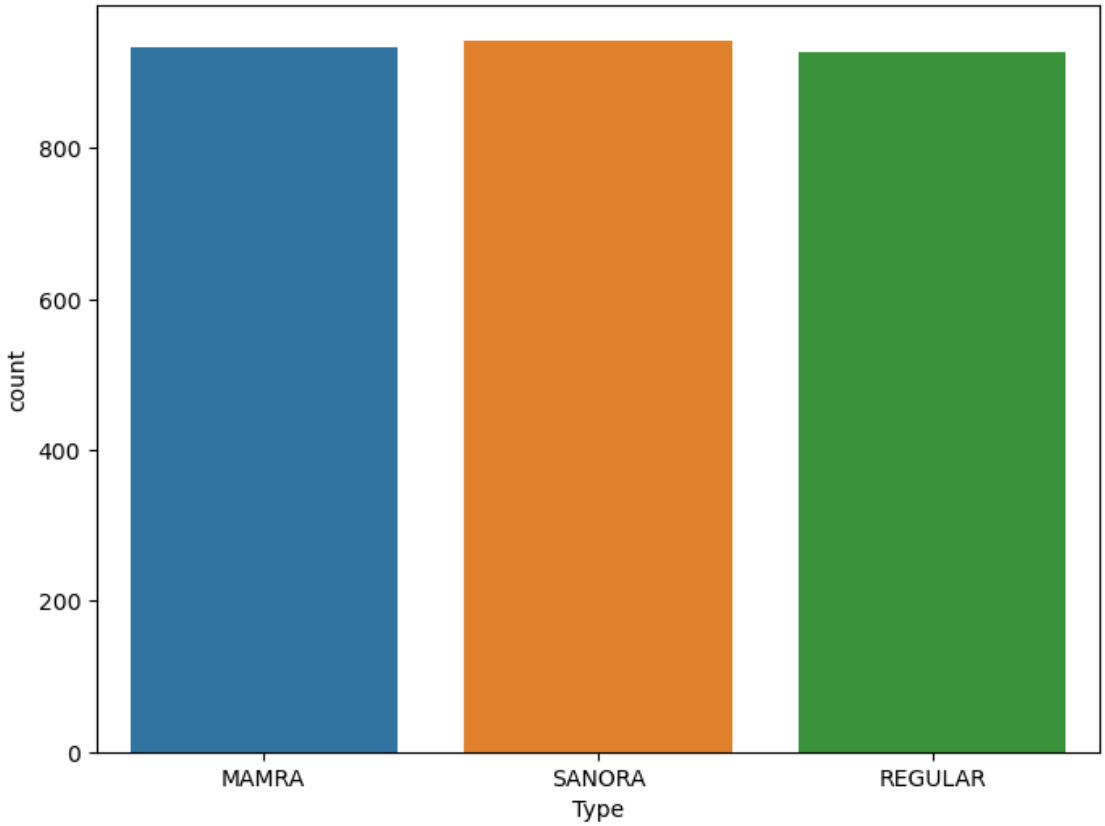
```
c:\Users\iamim\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning:
The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```




Roundness
Solidity
Compactness
Aspect Ratio
Eccentricity
Extent
Convex hull(convex area)


Width (minor axis)
Length (major axis)
Type
MAMRA
SANORA
REGULAR
0
50
100
150
200
250
0
100
200
300
400
500


0
2
4
6
8
10
Roundness
Solidity
Compactness
Aspect Ratio
Eccentricity



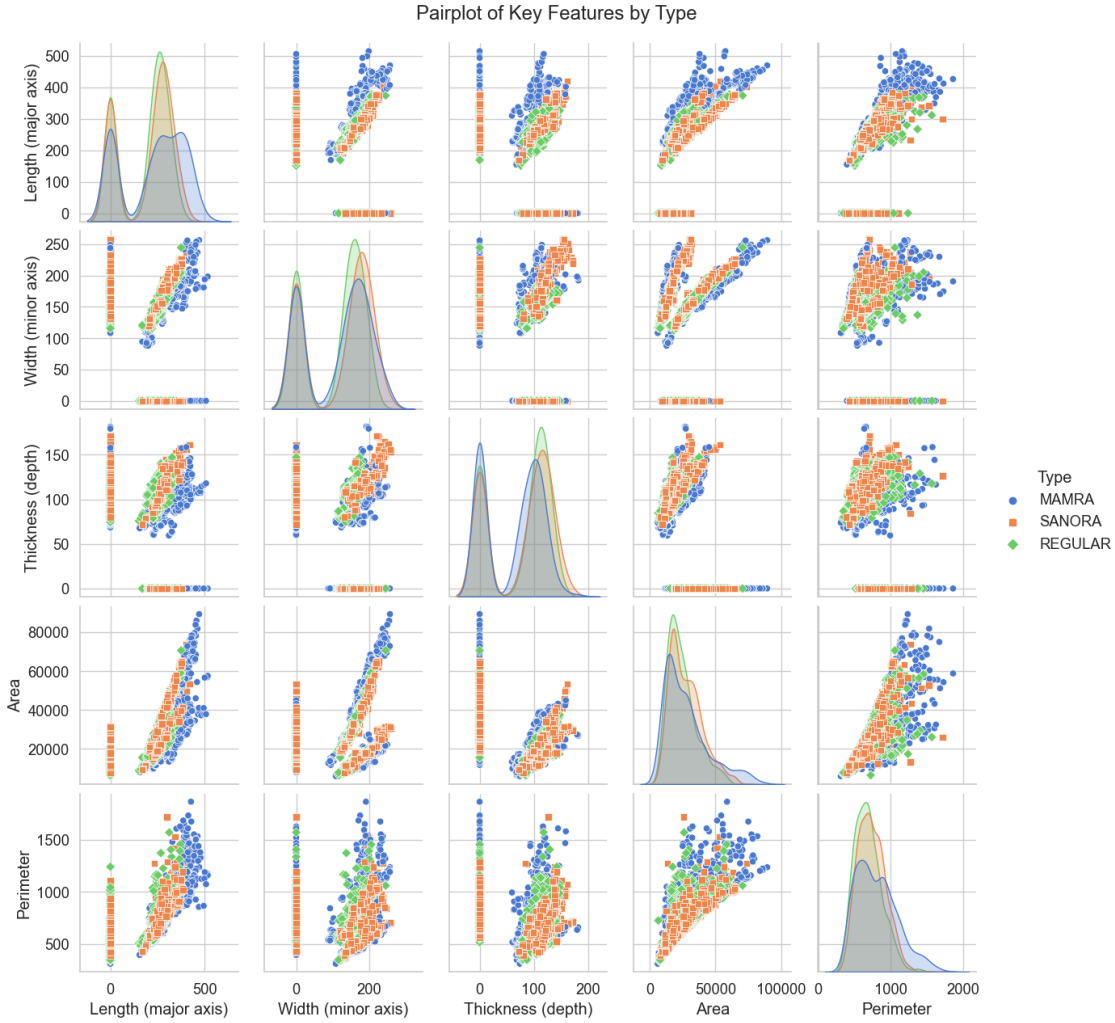
PLOTS (Imporved Interactions)

[16]:
```
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
import plotly.io as pio
import pandas as pd

# Seaborn style
sns.set(style="whitegrid", palette="muted", font_scale=1.2)
```

[17]:
```
# Pairplot
pairplot = sns.pairplot(df,
                        vars=["Length (major axis)", "Width (minor axis)",␣
 ↪"Thickness (depth)", "Area", "Perimeter"],
                        hue='Type',
                        diag_kind='kde',
                        markers=["o", "s", "D"])
```

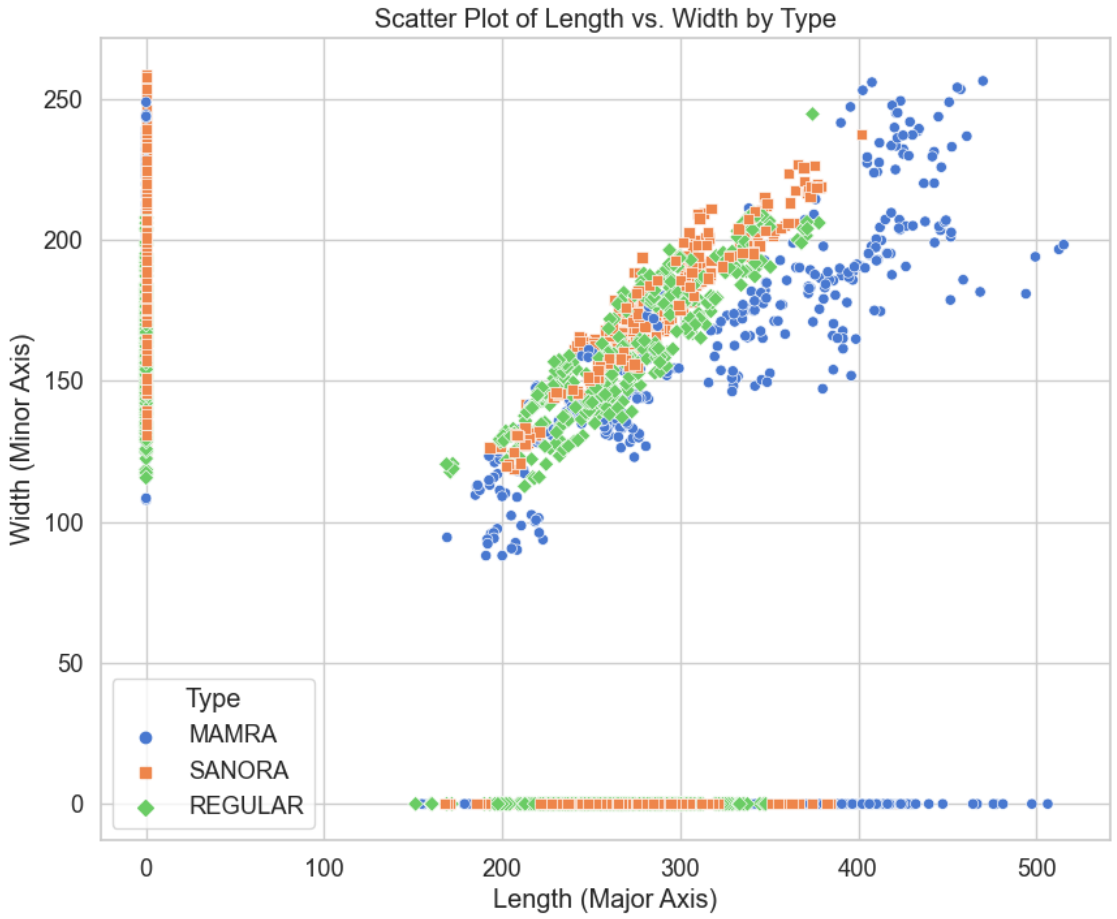
```
pairplot.fig.suptitle("Pairplot of Key Features by Type", y=1.02)
pairplot.savefig("pairplot.png")
plt.show()
```

```
c:\Users\iamim\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning:
The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

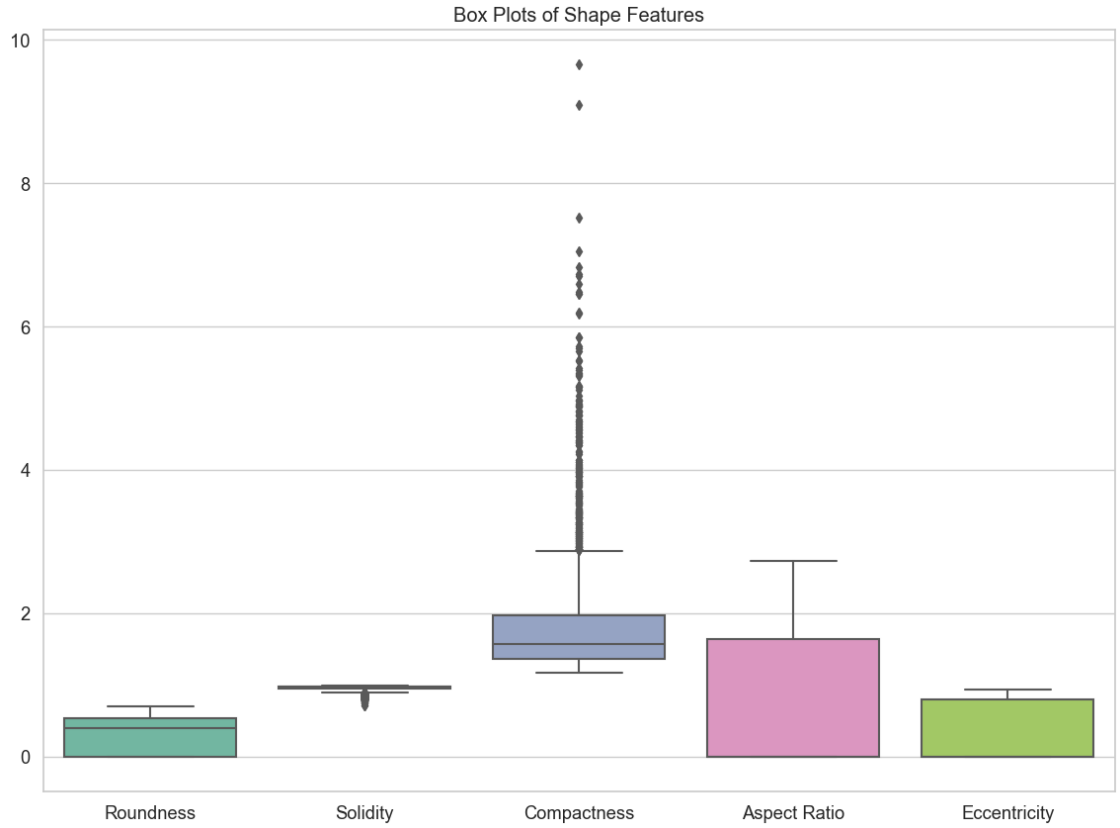


[18]:
```
# Scatter Plot with Seaborn
plt.figure(figsize=(10, 8))
scatter = sns.scatterplot(data=df, x="Length (major axis)", y="Width (minor␣
 ↪axis)", hue="Type", style="Type", markers=["o", "s", "D"])
plt.title("Scatter Plot of Length vs. Width by Type")
plt.xlabel("Length (Major Axis)")
plt.ylabel("Width (Minor Axis)")
plt.legend(title='Type')
```

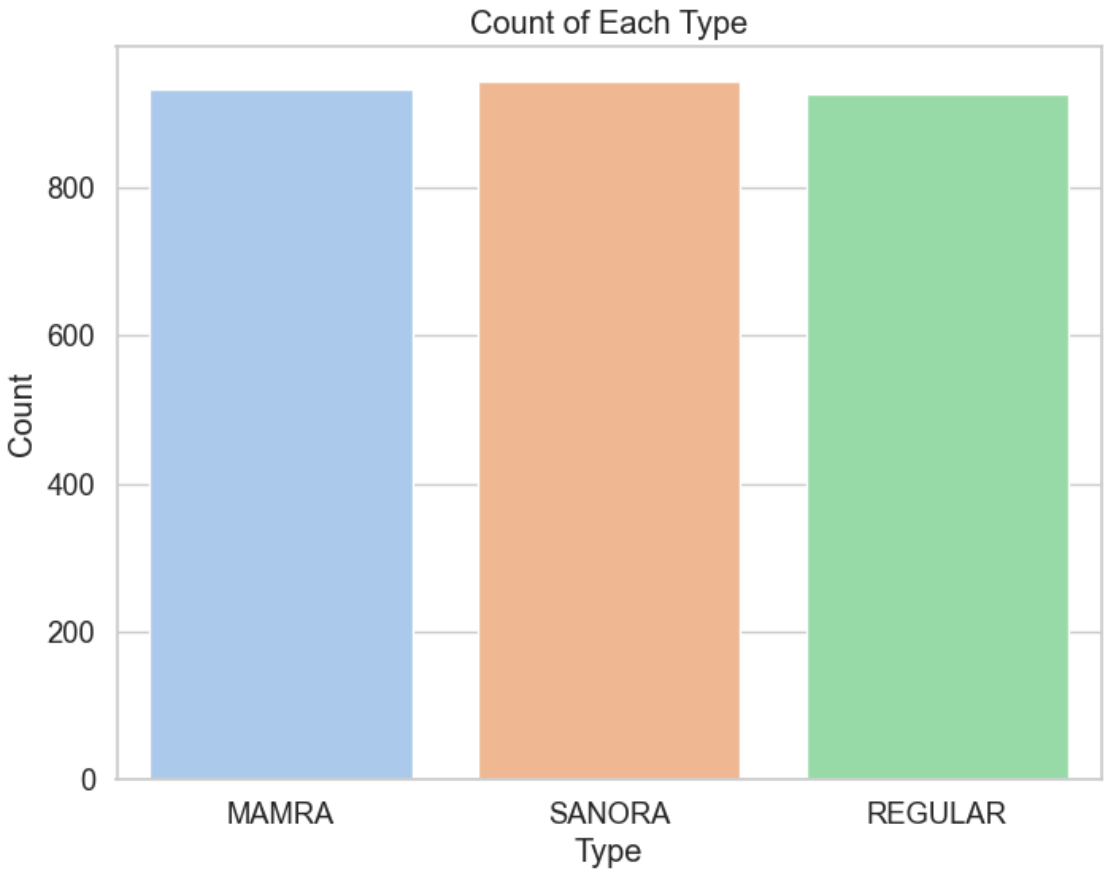
```
plt.savefig("scatter_plot.png")
plt.show()
```



[19]:
```
# Box Plots with enhanced layout and color
plt.figure(figsize=(14, 10))
boxplot = sns.boxplot(data=df[['Roundness', 'Solidity', 'Compactness', 'Aspect␣
↪Ratio', 'Eccentricity']], palette="Set2")
plt.title("Box Plots of Shape Features")
plt.savefig("box_plots.png")
plt.show()
```

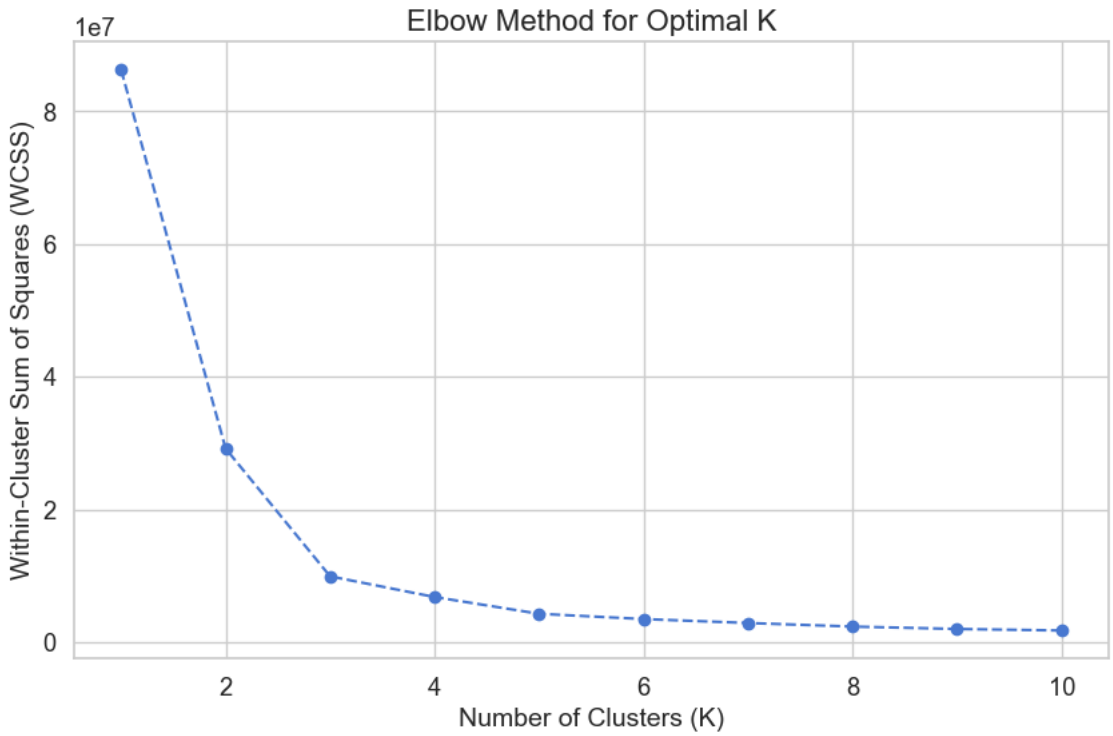


[20]:
```
# Count Plot with Seaborn
plt.figure(figsize=(8, 6))
countplot = sns.countplot(data=df, x='Type', palette="pastel")
plt.title("Count of Each Type")
plt.xlabel("Type")
plt.ylabel("Count")
plt.savefig("count_plot.png")
plt.show()
```



[21]:
```
import matplotlib.pyplot as plt
import plotly.graph_objects as go
import plotly.io as pio

# Matplotlib version
plt.figure(figsize=(10, 6))
plt.plot(range(1, 11), wcss, marker='o', linestyle='--', color='b')
plt.title('Elbow Method for Optimal K', fontsize=16)
plt.xlabel('Number of Clusters (K)', fontsize=14)
plt.ylabel('Within-Cluster Sum of Squares (WCSS)', fontsize=14)
plt.grid(True)
plt.savefig("elbow_curve.png")
plt.show()

# # Plotly interactive version
# elbow_fig = go.Figure()

# elbow_fig.add_trace(go.Scatter(
#     x=list(range(1, 11)),
```

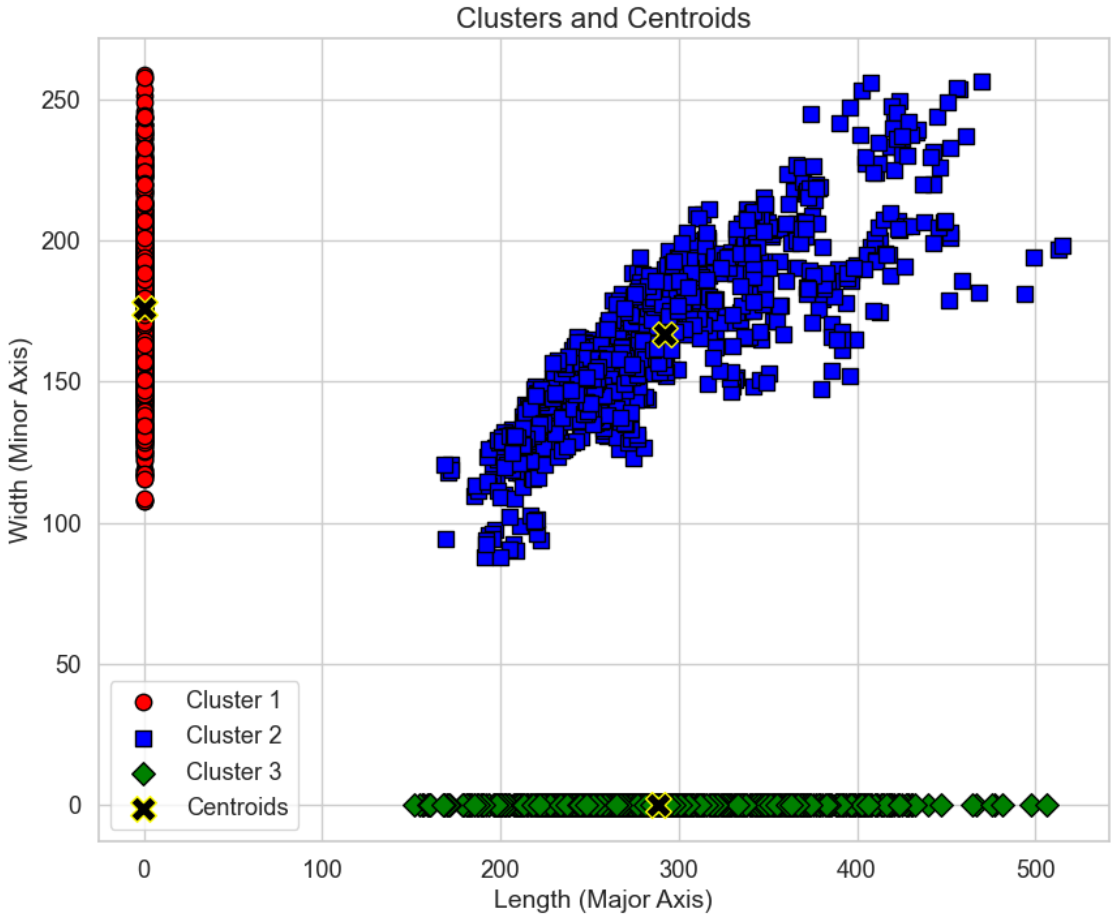
```
#     y=wcss,
#     mode='lines+markers',
#     line=dict(dash='dash', color='blue'),
#     marker=dict(symbol='circle', size=10, color='blue'),
#     name='WCSS'
# ))

# elbow_fig.update_layout(
#     title='Elbow Method for Optimal K',
#     xaxis_title='Number of Clusters (K)',
#     yaxis_title='Within-Cluster Sum of Squares (WCSS)',
#     template='plotly_white'
# )

# elbow_fig.write_image("elbow_curve_interactive.png")
# # pio.write_html(elbow_fig, file="elbow_curve_interactive.html",␣
 ↪auto_open=True)
```



[22]:
```
import matplotlib.pyplot as plt
import plotly.express as px
import plotly.graph_objects as go
import numpy as np
```

```
# Matplotlib Version
plt.figure(figsize=(10, 8))

# Plotting each cluster with customized markers and colors
plt.scatter(X_array[pred_y == 0, 0], X_array[pred_y == 0, 1], s=80, c='red',␣
 ↪marker='o', edgecolors='black', label='Cluster 1')
plt.scatter(X_array[pred_y == 1, 0], X_array[pred_y == 1, 1], s=80, c='blue',␣
 ↪marker='s', edgecolors='black', label='Cluster 2')
plt.scatter(X_array[pred_y == 2, 0], X_array[pred_y == 2, 1], s=80, c='green',␣
 ↪marker='D', edgecolors='black', label='Cluster 3')
# Add more scatter plots for additional clusters if needed

# Plotting centroids with larger, distinct markers
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1],␣
 ↪s=200, c='black', marker='X', edgecolor='yellow', label='Centroids')

# Adding labels, title, and legend
plt.xlabel('Length (Major Axis)', fontsize=14)
plt.ylabel('Width (Minor Axis)', fontsize=14)
plt.title('Clusters and Centroids', fontsize=16)
plt.legend()
plt.grid(True)
plt.savefig("clusters_and_centroids.png")
plt.show()

# # Plotly Interactive Version
# plotly_fig = go.Figure()

# # Adding clusters
# plotly_fig.add_trace(go.Scatter(
#     x=X_array[pred_y == 0, 0],
#     y=X_array[pred_y == 0, 1],
#     mode='markers',
#     marker=dict(color='red', size=10, symbol='circle', line=dict(width=2,␣
 ↪color='black')),
#     name='Cluster 1'
# ))

# plotly_fig.add_trace(go.Scatter(
#     x=X_array[pred_y == 1, 0],
#     y=X_array[pred_y == 1, 1],
#     mode='markers',
#     marker=dict(color='blue', size=10, symbol='square', line=dict(width=2,␣
 ↪color='black')),
#     name='Cluster 2'
# ))
```

```
# plotly_fig.add_trace(go.Scatter(
#     x=X_array[pred_y == 2, 0],
#     y=X_array[pred_y == 2, 1],
#     mode='markers',
#     marker=dict(color='green', size=10, symbol='diamond', line=dict(width=2,␣
 ↪color='black')),
#     name='Cluster 3'
# ))

# # Adding centroids
# plotly_fig.add_trace(go.Scatter(
#     x=kmeans.cluster_centers_[:, 0],
#     y=kmeans.cluster_centers_[:, 1],
#     mode='markers',
#     marker=dict(color='black', size=15, symbol='x', line=dict(width=2,␣
 ↪color='yellow')),
#     name='Centroids'
# ))

# # Customizing layout
# plotly_fig.update_layout(
#     title='Clusters and Centroids',
#     xaxis_title='Length (Major Axis)',
#     yaxis_title='Width (Minor Axis)',
#     template='plotly_white'
# )

# # Saving the Plotly figure
# plotly_fig.write_image("clusters_and_centroids_interactive.png")
# pio.write_html(plotly_fig, file="clusters_and_centroids_interactive.html",␣
 ↪auto_open=True)
```


Clusters and Centroids
Width (Minor Axis)
Length (Major Axis)
0
50
100
150
200
250
0
100
200
300
400
500
Cluster 1
Cluster 2
Cluster 3
Centroids