```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import seaborn as sns

# Load your dataset
data = pd.read_csv('irisdata.csv')

# Display the first few rows of the dataset
print(data.head())

# Check for missing values
print(data.isnull().sum())

# Separate the features and the target (categorical column)
features = data.drop('species', axis=1)
target = data['species']

# Standardize the features
scaler = StandardScaler()
features_scaled = scaler.fit_transform(features)

# Display the first few rows of the standardized data
print(features_scaled[:5])

# Use the Elbow Method to find the optimal number of clusters
sse = []
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(features_scaled)
    sse.append(kmeans.inertia_)

# Plot the SSE for each number of clusters
plt.figure(figsize=(10, 6))
plt.plot(range(1, 11), sse, marker='o')
plt.xlabel('Number of clusters')
plt.ylabel('Sum of squared distances (SSE)')
plt.title('Elbow Method for Optimal Number of Clusters')
plt.show()

# Assuming the optimal number of clusters is 3 based on the Elbow
Method
kmeans = KMeans(n_clusters=3, random_state=42)
kmeans.fit(features_scaled)

# Get the cluster labels
```

```
labels = kmeans.labels_

# Add the cluster labels to the original dataframe
data['Cluster'] = labels

# Visualize the clusters
plt.figure(figsize=(10, 6))
sns.scatterplot(x=features_scaled[:, 0], y=features_scaled[:, 1],
hue=labels, palette='viridis')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.title('Clusters Visualization')
plt.show()

# Optionally, visualize the clusters with the original species labels
plt.figure(figsize=(10, 6))
sns.scatterplot(x=features_scaled[:, 0], y=features_scaled[:, 1],
hue=target, palette='viridis')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.title('Original Species Visualization')
plt.show()
```