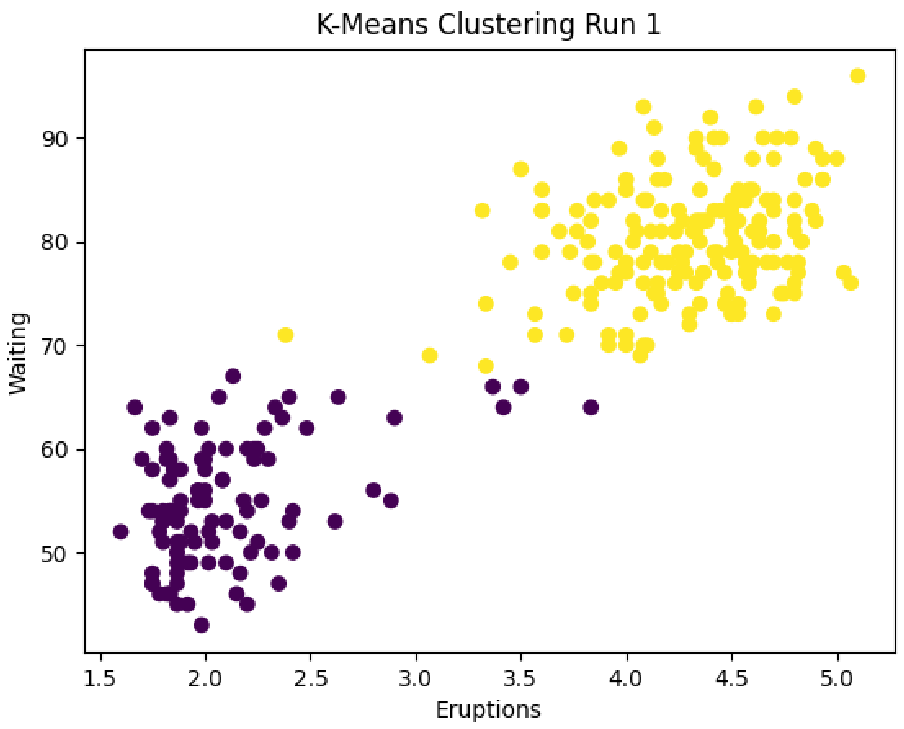
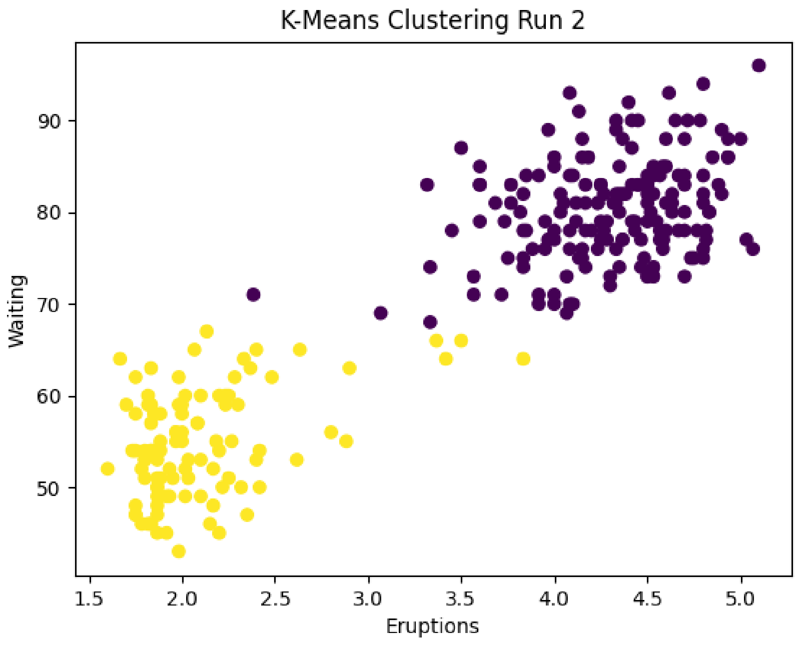
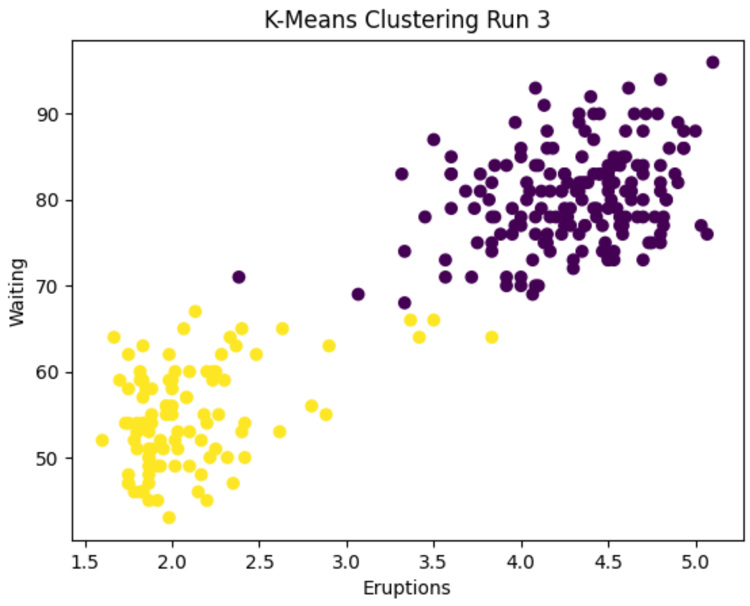
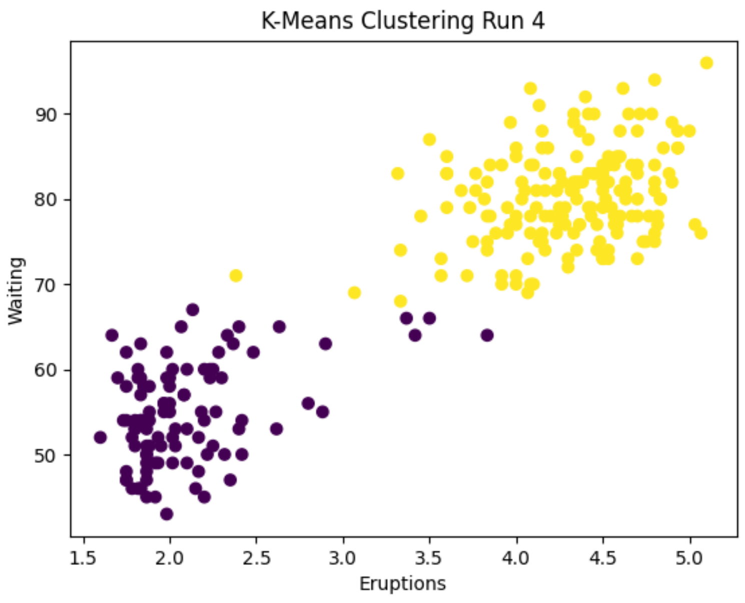
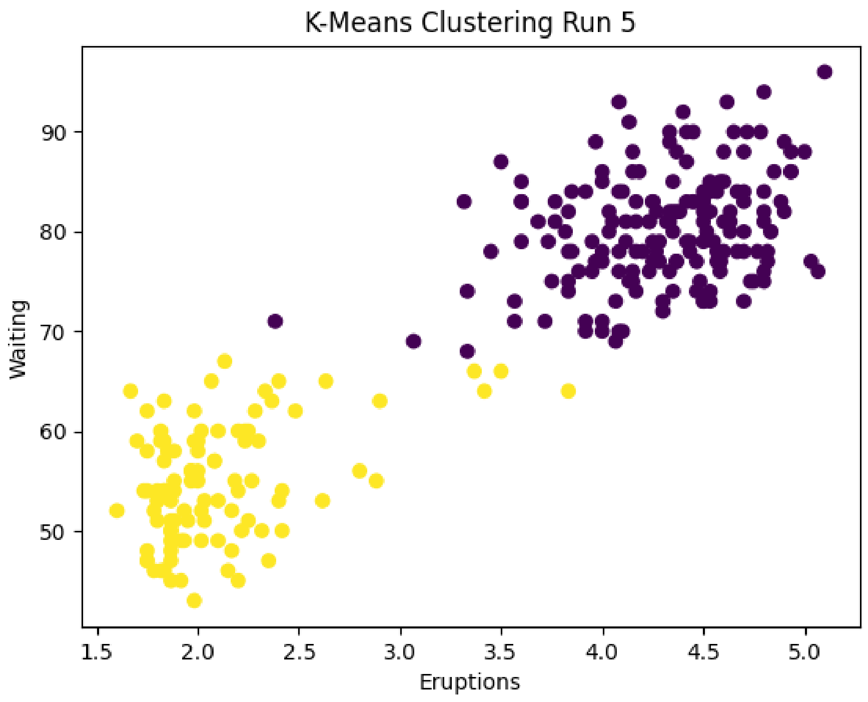
```python
##############################
## Fourth Assignment ##
##############################


# 1.) Old Faithful

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

geyser_data = pd.read_csv('/Users/aidanlynde/ECON491/Assignment4/data/geyser.csv')

# a.)
   # Remove the empty column
geyser_data = geyser_data[['eruptions', 'waiting']]

   # Check for NaN values and handle them
geyser_data.dropna(inplace=True)
geyser_data.reset_index(drop=True, inplace=True)

   # Perform k-means clustering with 5 different initializations
for i in range(5):
       # Randomly select two indices
   initial_indices = np.random.choice(geyser_data.index, 2, replace=False)
   initial_centers = geyser_data.iloc[initial_indices]

       # Run k-means
   kmeans = KMeans(n_clusters=2, init=initial_centers.values, n_init=1)
   kmeans.fit(geyser_data)

       # Scatter plot
   plt.figure()
   plt.scatter(geyser_data['eruptions'], geyser_data['waiting'], c=kmeans.labels_)
   plt.title(f'K-Means Clustering Run {i+1}')
   plt.xlabel('Eruptions')
   plt.ylabel('Waiting')
   plt.show()

# b.) The k-means clustering of the Old Faithful geyser data, with the number of
clusters set to two,
# reveals distinct patterns in geyser eruption behavior. The two clusters can be
interpreted as representing
# two different types of eruptions. The first cluster, possibly consisting of shorter
waiting times, could
# correspond to eruptions that are more frequent but less forceful or of shorter
duration. This cluster
# represents a quick cycle of the geyser's activity, where the energy is released more
regularly but with less
# intensity. In contrast, the second cluster, likely characterized by longer waiting
times, suggests eruptions
# that are less frequent but more powerful or longer-lasting. This cluster indicates a
slower cycle where the geyser
# accumulates more energy over a longer period, resulting in a more dramatic eruption.
The scatterplots
# generated from the k-means clustering illustrate these two distinct behaviors in the
Old Faithful's eruptions,
# with each cluster capturing a unique aspect of the geyser's natural rhythm. The
clarity of this separation in
# the data highlights the effectiveness of k-means in identifying and categorizing
these eruption patterns,
# offering valuable insights into the geothermal dynamics of the Old Faithful geyser.
```




K-Means Clustering Run 2
Waiting
Eruptions
90
80
70
60
50
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0


K-Means Clustering Run 3
Waiting
Eruptions
90
80
70
60
50
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0

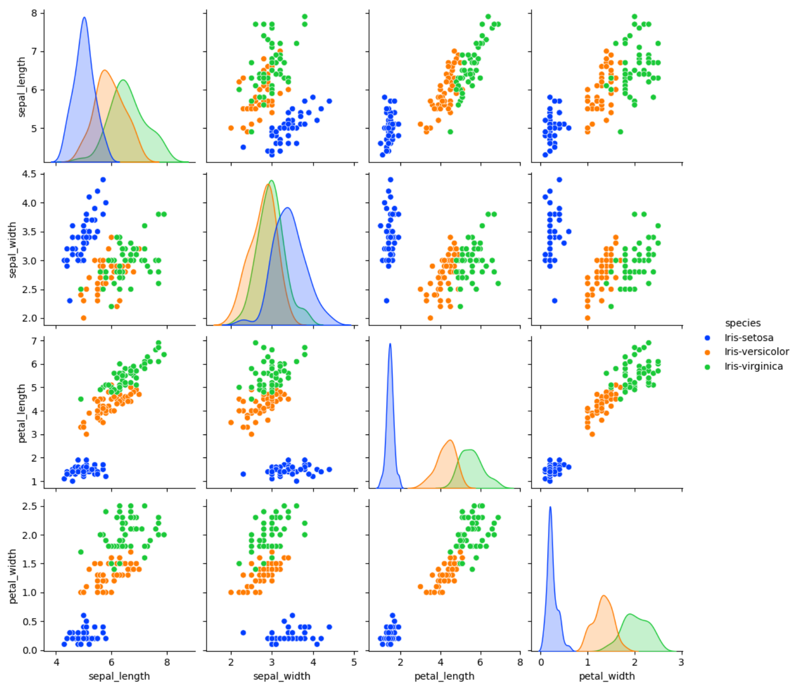


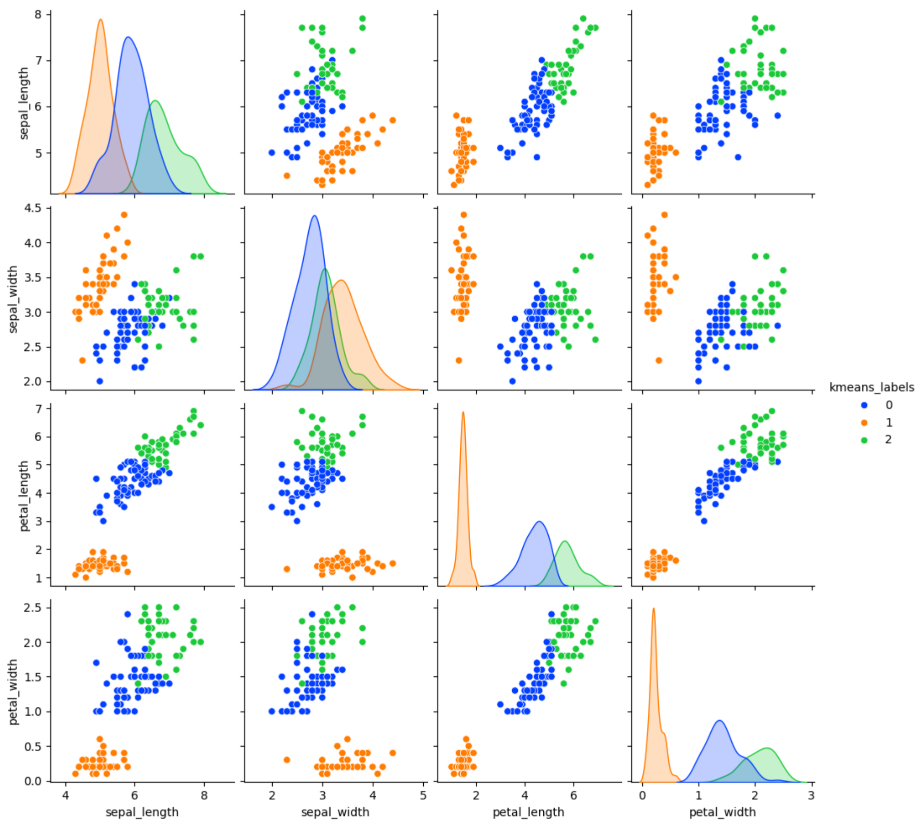


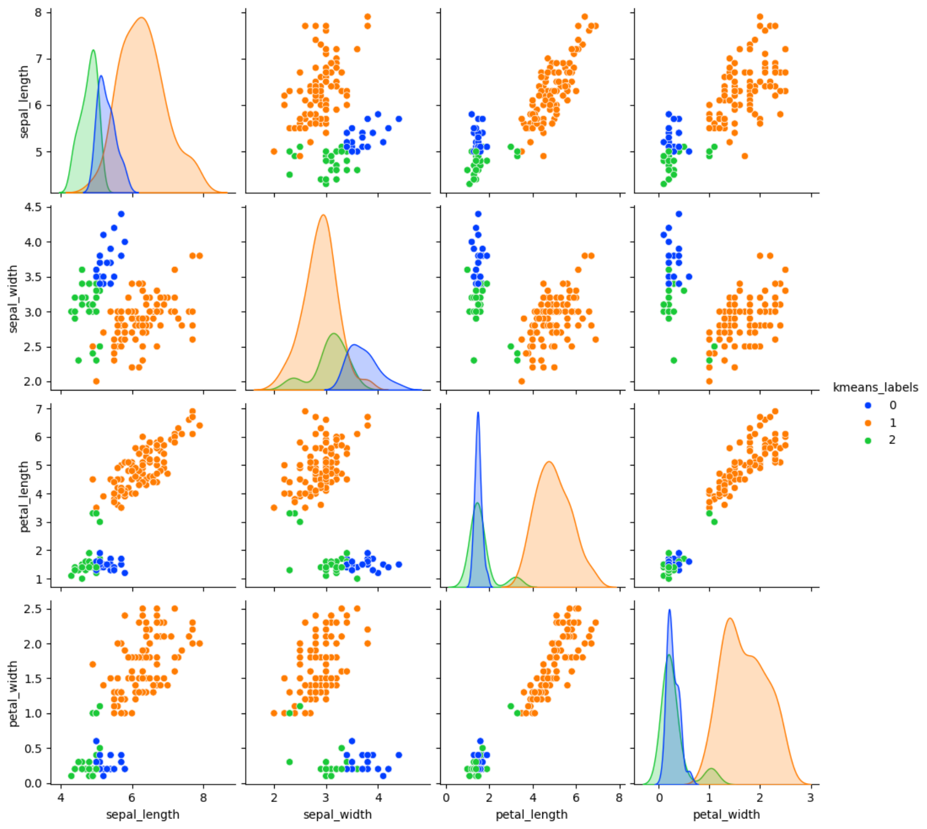
```
# 2.) Iris Flower Data Set
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
import numpy as np
```

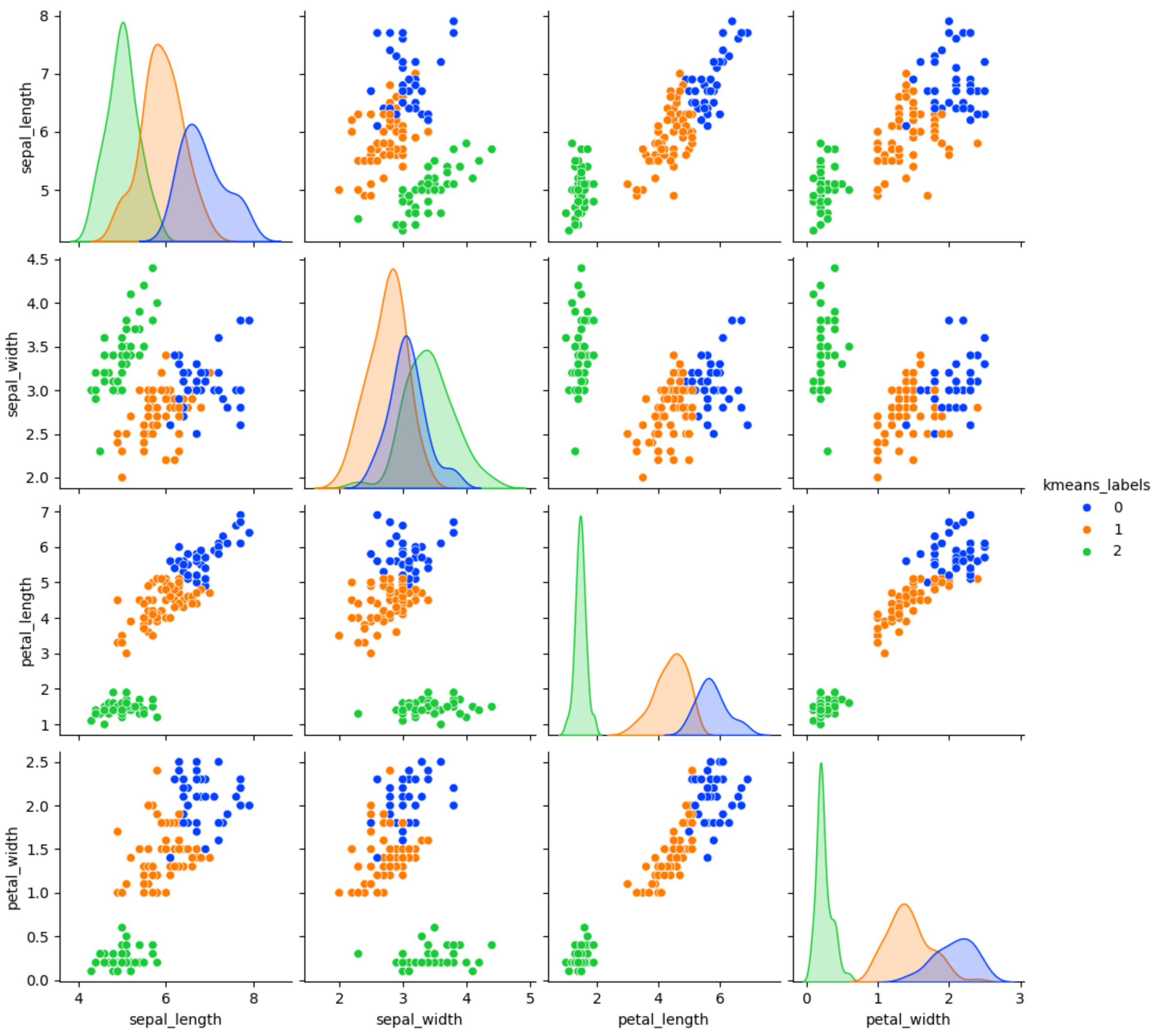
```
# a.)

   # Load the Iris dataset
iris_file_path = '/Users/aidanlynde/ECON491/Assignment4/data/iris.csv'
iris_data = pd.read_csv(iris_file_path, header=None)

   # Add column names
iris_data.columns = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width',
'species']

   # Create scatterplots for each pair of variables
sns.pairplot(iris_data, hue='species', palette='bright')
plt.suptitle("Scatterplots for Each Pair of Variables in the Iris Dataset", y=1.02)
plt.show()

# b.)

   # Prepare for k-means clustering with k=3 and 5 different initializations
n_init = 5
k = 3

   # Scatterplot function
def plot_kmeans_scatterplots(data, kmeans_labels, title):
   sns.pairplot(data, hue=kmeans_labels, palette='bright')
   plt.suptitle(title, y=1.02)
   plt.show()

   # Running k-means and plotting results
for i in range(n_init):
       # Randomly select three data points as initial centers
   initial_indices = np.random.choice(iris_data.index, k, replace=False)
   initial_centers = iris_data.iloc[initial_indices, :-1]

       # Run k-means
   kmeans = KMeans(n_clusters=k, init=initial_centers.values, n_init=1)
   kmeans.fit(iris_data.iloc[:, :-1])

       # Add KMeans labels to the DataFrame for plotting
   iris_data_with_labels = iris_data.copy()
   iris_data_with_labels['kmeans_labels'] = kmeans.labels_
```

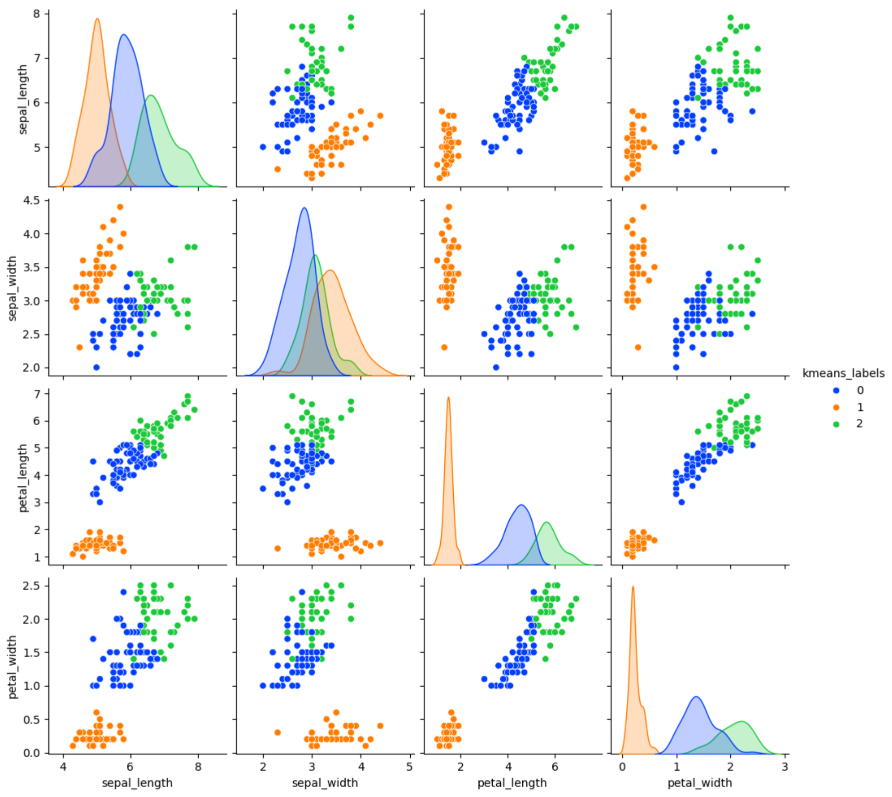
```
        # Create scatter plots
   plot_title = f'K-Means Clustering Run {i+1} with k=3'
   plot_kmeans_scatterplots(iris_data_with_labels, 'kmeans_labels', plot_title)

# c.)

from scipy.cluster.hierarchy import dendrogram, linkage
from matplotlib.colors import ListedColormap

   # Prepare data (excluding the species column)
X = iris_data.iloc[:, :-1].values

   # Perform hierarchical clustering using complete linkage
Z = linkage(X, method='complete', metric='euclidean')

   # Function to label and color dendrogram
def label_color_func(id):
   if id < len(iris_data):
       species = iris_data.iloc[id]['species']
       if species == 'Iris-setosa':
           return 'blue'
       elif species == 'Iris-versicolor':
           return 'green'
       else:
           return 'red'
   else:
       return 'black'

   # Plotting the dendrogram
plt.figure(figsize=(12, 8))
dendrogram(Z, labels=iris_data['species'].values, leaf_rotation=90, leaf_font_size=8,
          color_threshold=0, above_threshold_color='gray',
link_color_func=label_color_func)
plt.title("Hierarchical Clustering Dendrogram of the Iris Dataset")
plt.xlabel("Sample index or (cluster size)")
plt.ylabel("Distance")
plt.show()

# d.) In comparing the k-means and hierarchical clustering approaches applied to the
Iris dataset,
# distinct differences in cluster formation and data segmentation are evident. K-means
clustering,
# known for its simplicity and efficiency, demonstrated a clear partitioning of the
dataset into three groups
# based on the specified number of clusters (k=3). The scatterplots generated from
k-means exhibit a
# relatively clean separation, particularly between the Iris setosa species and the
other two. However,
# k-means clustering is sensitive to the initial placement of centroids and may
produce varying results upon
# different initializations. In contrast, hierarchical clustering, employing a
complete linkage strategy,
# offered a more nuanced view of data relationships through its dendrogram. This
method illustrated the hierarchical
# structure of data groupings and provided insights into the natural clustering of the
dataset, revealing a more
# gradual merging of clusters. The dendrogram's color coding of species further
highlighted how closely or distantly
# the species are related in terms of their features. While hierarchical clustering
excelled in revealing the data's
# inherent structure, it is computationally more intensive than k-means and can be
less straightforward to interpret
# for cluster assignment. Overall, each method has its strengths and limitations, with
k-means excelling in clear-cut
# cluster separation and efficiency, and hierarchical clustering providing deeper
insights into the dataset's underlying structure.
```

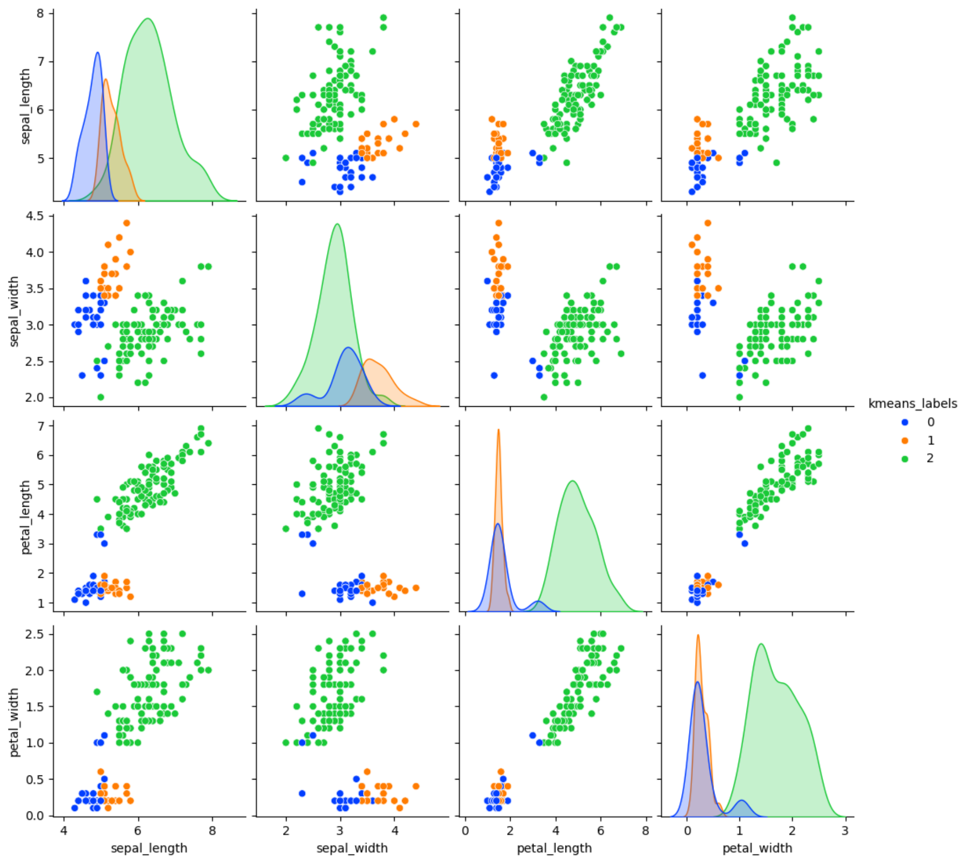


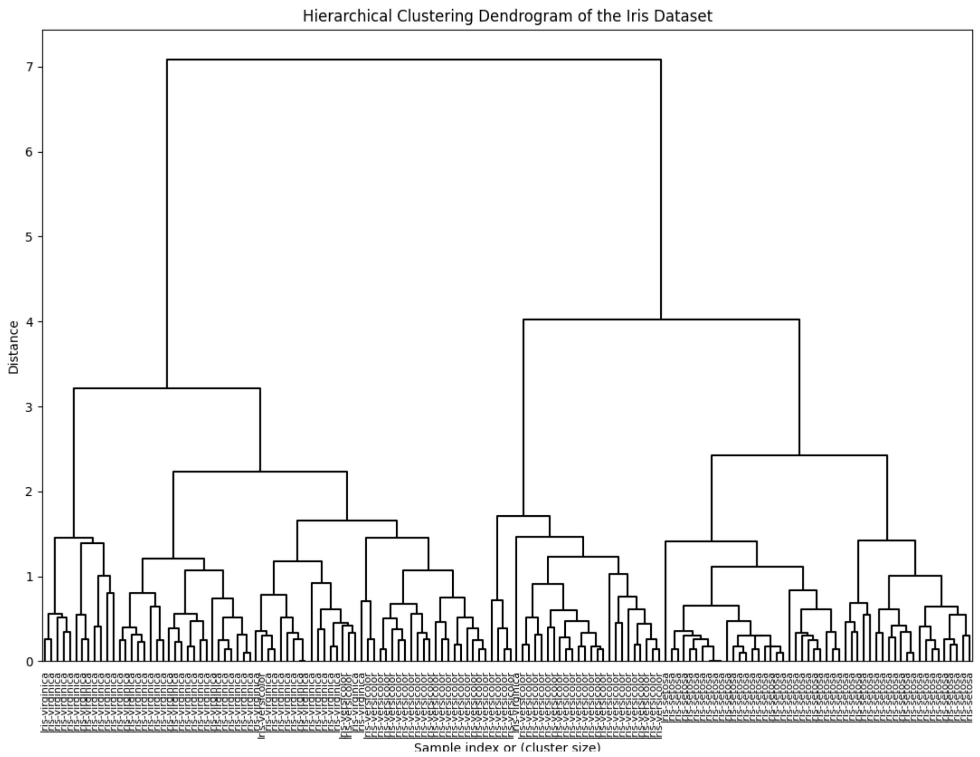

sepal_length
sepal_width
petal_length
petal_width
kmeans_labels
0
1
2


sepal_length
sepal_width
petal_length
petal_width
kmeans_labels
0
1
2


sepal_length
sepal_width
petal_length
petal_width
kmeans_labels
0
1
2

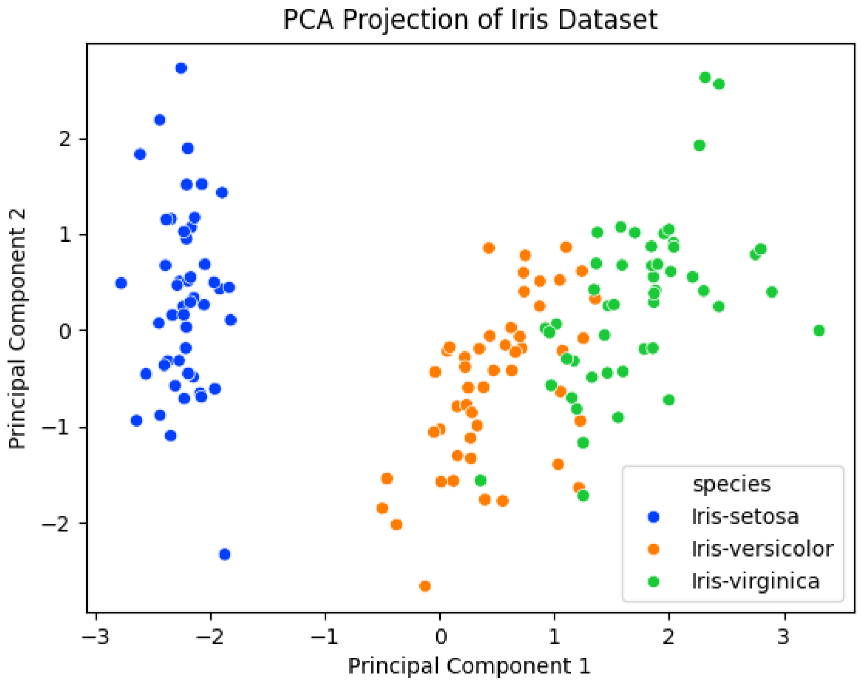

sepal_length
sepal_width
petal_length
petal_width
kmeans_labels
0
1
2

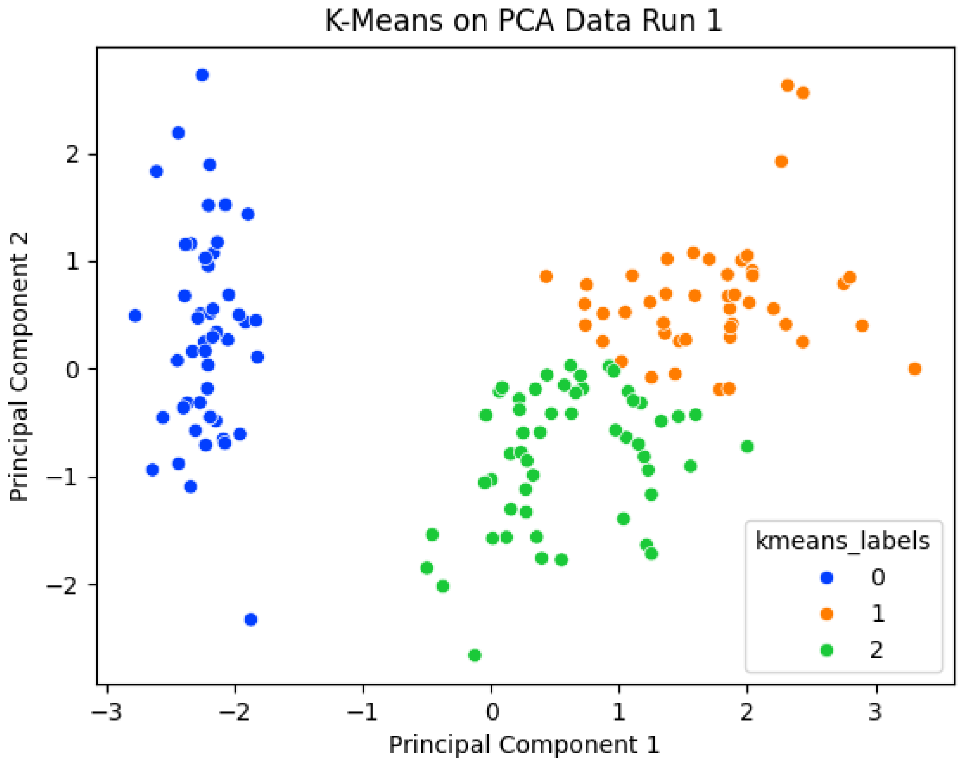

sepal_length
sepal_width
petal_length
petal_width
kmeans_labels
0
1
2



```
# 3.) Iris Flower Data set revised

from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
# a.)
   # Normalizing the data
scaler = StandardScaler()
X_normalized = scaler.fit_transform(iris_data.iloc[:, :-1])  # Excluding the species column

   # Applying PCA
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_normalized)

   # Creating a DataFrame for the PCA results
pca_df = pd.DataFrame(data=X_pca, columns=['PC1', 'PC2'])
pca_df['species'] = iris_data['species']

   # Plotting the PCA-transformed data
sns.scatterplot(x='PC1', y='PC2', hue='species', data=pca_df, palette='bright')
plt.title("PCA Projection of Iris Dataset")
plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.show()

# b.)

   # K-means clustering on PCA-transformed data
for i in range(n_init):
       # Randomly select three data points as initial centers
   initial_indices = np.random.choice(pca_df.index, k, replace=False)
   initial_centers = pca_df.iloc[initial_indices, :-1]

       # Run k-means
   kmeans_pca = KMeans(n_clusters=k, init=initial_centers.values, n_init=1)
   kmeans_pca.fit(pca_df.iloc[:, :-1])

       # Scatter plot with k-means results
   pca_df['kmeans_labels'] = kmeans_pca.labels_
   sns.scatterplot(x='PC1', y='PC2', hue='kmeans_labels', data=pca_df,
palette='bright')
   plt.title(f'K-Means on PCA Data Run {i+1}')
   plt.xlabel("Principal Component 1")
   plt.ylabel("Principal Component 2")
   plt.show()

# Applying k-means clustering to the PCA-transformed Iris dataset yields intriguing
insights into
# the effectiveness of dimensionality reduction in clustering tasks. The PCA
transformation,
# by condensing the dataset into its two principal components, captures the majority
of the variance
# in the data while reducing its complexity. This simplification facilitates a more
pronounced separation
# between clusters, as evidenced in the scatter plots of the PCA-transformed data.
When comparing the
# k-means results on the PCA data with those obtained from the original
four-dimensional space, a notable
# improvement in cluster delineation is observed. The clusters in the PCA space align
more closely with the
# natural species divisions, suggesting that PCA aids in accentuating the inherent
differences between
```

```
# the Iris species. This improvement is particularly evident for the Iris-setosa
species, which appears
# almost completely isolated from the other two species in the PCA space. However, the
overlap between
# Iris-versicolor and Iris-virginica, although reduced, still presents a challenge,
indicating that some
# complex relationships in the original data cannot be fully captured by the first two
principal components
# alone. Overall, the application of k-means to the PCA-transformed data demonstrates
the utility of PCA
# in enhancing clustering performance, particularly in scenarios where reducing data
dimensionality while
# retaining key variance is crucial.
```


PCA Projection of Iris Dataset
Principal Component 1
Principal Component 2
species
Iris-setosa
Iris-versicolor
Iris-virginica
−3
−2
−1
0
1
2
3


K-Means on PCA Data Run 1
Principal Component 1
Principal Component 2
kmeans_labels
0
1
2
−3
−2
−1
0
1
2
3

```
# 4.) k-means on the plane

# a.) In this scenerio, we have two sets of points, P and Q, on a two-dimensional
plane. The k-means algorithm is
# initialized with one point from P and one from Q as cluster centers. Given the
conditions: 1.)Each within-cluster
# distance is bounded above by D/2 2.)The distance between any point in P and any
point in Q is greater than 2D
# We can conclude that if k-means is initialized as specified, it will correctly
classify all points in the first
# iteration and terminate. This is because the initial cluster centers are closer to
every point in their respective
#  clusters than to any point in the other cluster, given the distance constraints.
Thus, no point will change its
# cluster assignment in subsequent iterations, leading to immediate termination.

# b.) For an example where k-means converges in one step to the correct partition
after an incorrect initial partition:
# Consider a simple case with n = m = 2.
# Let P1 = (1,1), P2 = (1,2), and Q1 = (10,10), Q2 = (10,11)
# initialize k-means with one center at P1 and the other at Q2
# After the first iteration, k-means will move the cluster centers to the mean of the
points closest to them. The new
# centers will likely be at the mean of P1 and P2, and Q1 and Q2, respectively. All
subsequent assignments
# will be correct and the algorithm will converge

# c.) For an example where k-means converges in one step to an incorrect partition:
# Again consider a case with n = m = 2.
# Let P1 = (1,1), P2 = (2,2), and Q1 = (1,2), Q2 = (2,1)
# initialize k-means with centers at P1 and Q1
# The points are positioned such that, after the first iteration, the new centers may
be incorrectly positioned
# to cluster P1 with Q1, and P2 with Q2, due to their proximity. This results in an
incorrect partition that the
# algorithm will maintain in subsequent iterations
```