ECE 5464 HW4 – Prof. Jones – HW 5

Ramya Garapati
906650137

1. The answer – how many different species are represented here?

Through silhouette score, I found out that there are three clusters. Therefore, there are three species.

2. A written description of your method.

In data processing for Encoding Categorical Features the Categorical variables are transformed using OneHotEncoder, which converts each categorical feature with m possible values into m binary features. The Column Transformer from sklearn. compose is used to apply these transformations appropriately to different columns of the data frame.

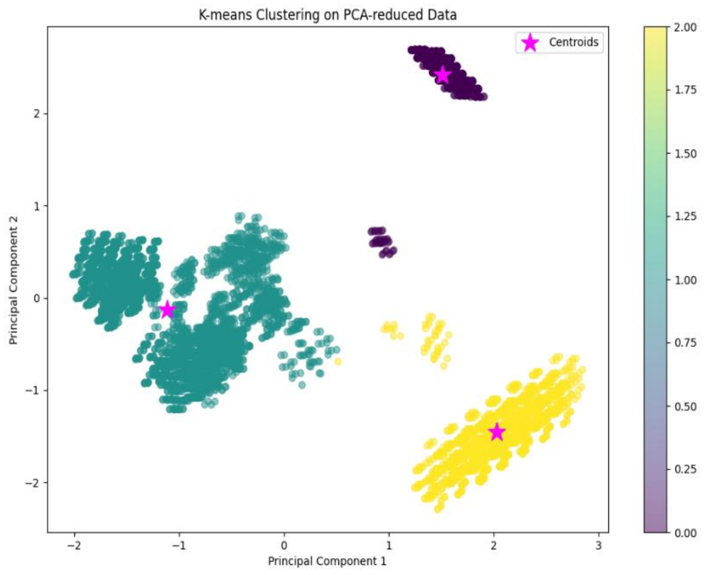
For the Dimensionality Reduction I used PCA. The Principal Component Analysis (PCA)is applied to the preprocessed data to reduce its dimensions while capturing most of the variance in the data. This step helps in visualizing the data in a lower-dimensional space and can improve clustering performance by reducing noise and redundancy in the data.

Finding the Optimal Number of Clusters With Elbow Method and Silhouette Score, I calculated both the inertia (sum of squared distances of samples to their closest cluster center) and silhouette scores (measure of how similar an object is to its own cluster compared to other clusters) for a range of cluster numbers. By plotting the silhouette scores against the number of clusters, I visually determine the optimal number of clusters. The optimal number is chosen based on the highest silhouette score, which indicates a good separation between the clusters.

Performing K-means Clustering With the optimal number of clusters determined from the previous step, I applied K-means clustering to the PCA-reduced data. K-means is run with the number of clusters set to this optimal value, ensuring the data is grouped into meaningful clusters.

The final step involves visualizing the clustered data in the PCA-reduced space. Each data point is colored based on its cluster label, and the centroids of the clusters are marked distinctly. This visualization helps in interpreting the clustering results and assessing the distribution and separation of clusters in the reduced space.

3. Some analysis that justifies your answer.



From the plot we can clearly see that there are 3 concentrations of data. Therefore it is safe to assume that there are 3 kinds of species.

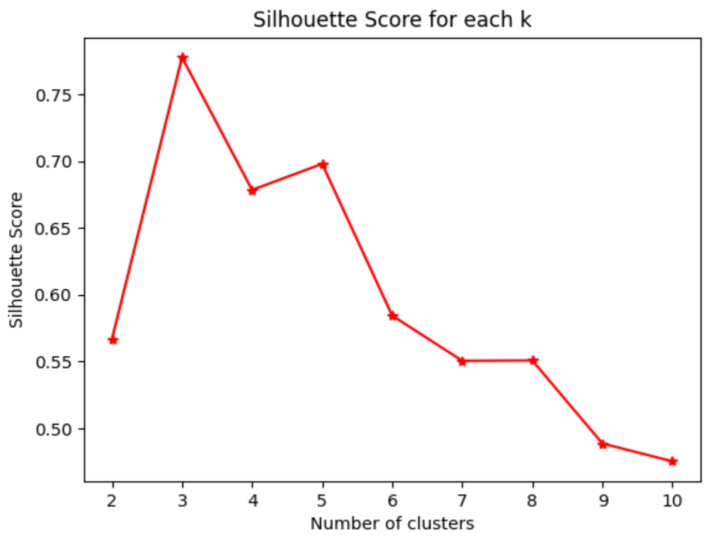
4. The source code for any program that you wrote to generate this analysis, using the techniques of unsupervised learning.

Code:

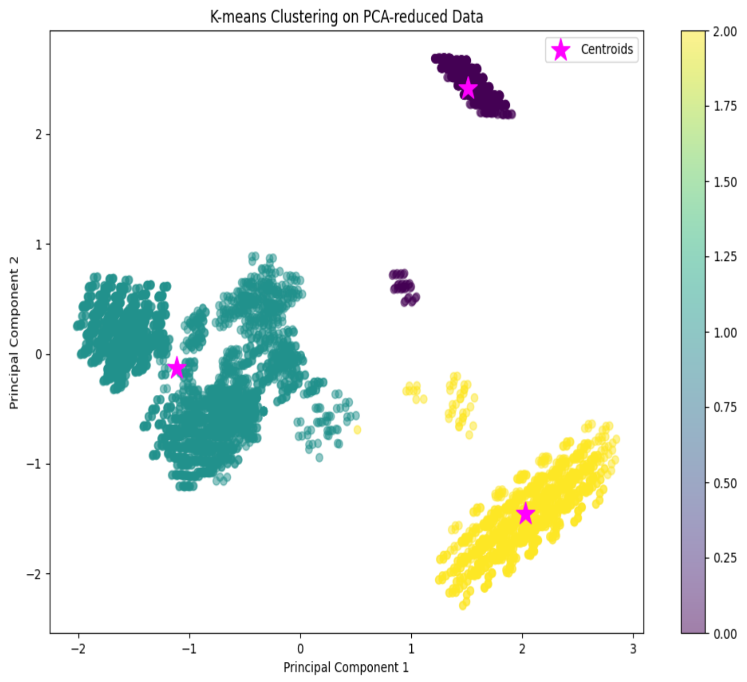
```
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt


def preprocess_data(df):
    """Preprocess the data by scaling numeric features and encoding categorical features."""
    numeric_columns = df.select_dtypes(include=[np.number]).columns.tolist()
    categorical_columns = df.select_dtypes(exclude=[np.number]).columns.tolist()

    transformer = ColumnTransformer([
        ('scale', StandardScaler(), numeric_columns),
        ('one_hot', OneHotEncoder(), categorical_columns)
    ], remainder='passthrough', sparse_threshold=0)

    transformed_data = transformer.fit_transform(df)
    feature_names = np.concatenate([
        numeric_columns,
        transformer.named_transformers_['one_hot'].get_feature_names_out(categorical_columns)
    ])
    return pd.DataFrame(transformed_data, columns=feature_names)


def apply_pca(df, n_components=2):
    """Apply PCA to reduce dimensions of the dataset."""
    pca = PCA(n_components=n_components)
    principal_components = pca.fit_transform(df)
    return pd.DataFrame(principal_components, columns=[f'PC{i + 1}' for i in
range(n_components)])


def findOptimalK(data_frame, max_clusters=10):
    inertia = []
    silhouette_scores = []
    K = range(2, max_clusters + 1)  # Starting from 2 clusters to compute silhouette score
    for k in K:
        kmeans = KMeans(n_clusters=k, random_state=77)
        labels = kmeans.fit_predict(data_frame)
        inertia.append(kmeans.inertia_)
        silhouette_scores.append(silhouette_score(data_frame, labels))

    plt.plot(K, silhouette_scores, 'r*-')
    plt.xlabel('Number of clusters')
    plt.ylabel('Silhouette Score')
    plt.title('Silhouette Score for each k')
    plt.show()

    # Choose the number of clusters based on highest silhouette score
    optimal_k = K[np.argmax(silhouette_scores)]
    return optimal_k
def perform_kmeans(df, n_clusters=5):
    """Apply KMeans clustering to the dataset."""
    kmeans = KMeans(n_clusters=n_clusters)
    kmeans.fit(df)
    return kmeans.labels_, kmeans.cluster_centers_


def plot_clusters(df, labels, centers):
    """Plot the clustered data with PCA components."""
    plt.figure(figsize=(12, 8))
    scatter = plt.scatter(df['PC1'], df['PC2'], c=labels, cmap='viridis', alpha=0.5)
    plt.scatter(centers[:, 0], centers[:, 1], s=300, c='magenta', marker='*', label='Centroids')
    plt.xlabel('Principal Component 1')
    plt.ylabel('Principal Component 2')
    plt.title('K-means Clustering on PCA-reduced Data')
    plt.legend()
    plt.colorbar(scatter)
    plt.show()


if __name__ == "__main__":
    df = pd.read_csv('mushroom.csv', engine='pyarrow')
    processed_df = preprocess_data(df)
    pca_df = apply_pca(processed_df)
    k = findOptimalK(pca_df)
    labels, centers = perform_kmeans(pca_df, k)
    plot_clusters(pca_df, labels, centers)
```

5. Any useful charts and graphs needed to support your answer.

Plot of silhouette scores:



Scatter plot of mushroom data and centroids:


K-means Clustering on PCA-reduced Data
Centroids
Principal Component 1
Principal Component 2