Assignment 5
Applications of Machine Learning

By Srilalith Nampally

Question 1:

By using the silhouette scores metric, I found the optimal number of clusters to be 3. Thus, the number of species identified is also 3.

Question 2:

Implementation:

Read_csv: using pandas dataframes

Datapre-processing: if numeric standard scaler and if categorical onehotencoding

Reducing Dimensionality: Used principal component analysis to reduce dimensions to 2.

Finding optimal K value: Used silhouette score metrics for clusters between 2 to 10.

Clustering: K-means clustering, using optimal K value found previously (3).

Visualization: matplotlib.pyplot

Question 3:

Code:

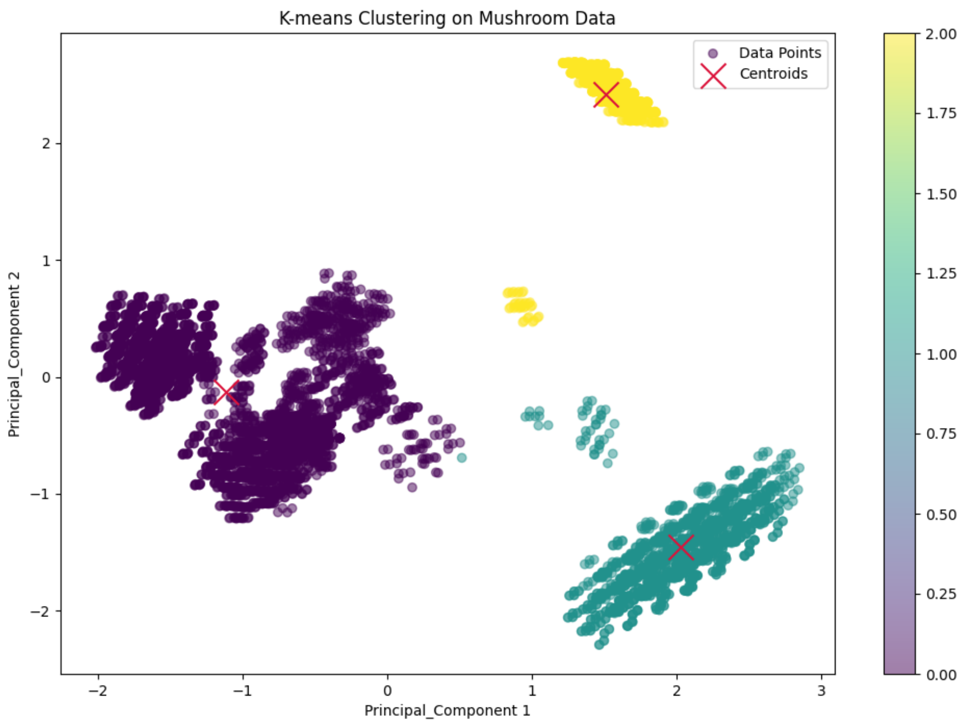
```
import pandas as pd

import numpy as np

import sklearn.cluster as cluster

from sklearn.compose import ColumnTransformer

from sklearn.preprocessing import StandardScaler, OneHotEncoder

from sklearn.decomposition import PCA

from sklearn.metrics import silhouette_score
```

```
import matplotlib.pyplot as plt


def readCSV(filepath):
    df = pd.read_csv(filepath, engine='pyarrow')
    return df


def preprocessData(data_frame):
    numeric_columns = data_frame.select_dtypes(include=[np.number]).columns.tolist()
    categorical_columns = data_frame.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(data_frame)
    one_hot_features = transformer.named_transformers_['one_hot'].get_feature_names_out(categorical_columns)
    features = np.append(numeric_columns, one_hot_features)
    transformed_df = pd.DataFrame(transformed_data, columns=features)
    return transformed_df


def applyPCA(data_frame, n_components=2):
    pca = PCA(n_components=n_components)
    principal_components = pca.fit_transform(data_frame)
    principal_df = pd.DataFrame(data=principal_components, columns=[f'PC{i}' for i in range(1, n_components+1)])
    return principal_df
```

```
def findOptimalClusters(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 = cluster.KMeans(n_clusters=k, random_state=42)
    labels = kmeans.fit_predict(data_frame)
    inertia.append(kmeans.inertia_)
    silhouette_scores.append(silhouette_score(data_frame, labels))

  # plt.subplot(1, 2, 2)
  plt.plot(K, silhouette_scores, 'bx-')
  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 KmeansClustering(data_frame, n_clusters=5):
  kmeans = cluster.KMeans(n_clusters=n_clusters)
  kmeans.fit(data_frame)
  return kmeans.labels_, kmeans.cluster_centers_
```

```
if __name__ == "__main__":
    filepath = 'mushroom.csv'
    DF = readCSV(filepath)
    DF = preprocessData(DF)
    PCA_DF = applyPCA(DF)
    optimal_k = findOptimalClusters(PCA_DF)
    labels, centers = KmeansClustering(PCA_DF, optimal_k)


    plt.figure(figsize=(12, 8))
    scatter = plt.scatter(PCA_DF['PC1'], PCA_DF['PC2'], c=labels, cmap='viridis', alpha=0.5,
marker='o', label='Data Points')
    plt.scatter(centers[:, 0], centers[:, 1], s=300, c='crimson', marker='x', label='Centroids')
    plt.xlabel('Principal_Component 1')
    plt.ylabel('Principal_Component 2')
    plt.title('K-means Clustering on Mushroom Data')
    plt.legend()
    plt.colorbar(scatter)
    plt.show()
```

Question 4:



From this figure it is clear that there are 3 major groups of data. Based on this it is safe to say that there are actually 3 species even though we have some outliers.

Question 5:

Plot for Optimal K:

K-Means Scatter-Plot: