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:

A screenshot of a computer screen

Description automatically generated

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:

A graph with blue lines

Description automatically generated

## K-Means Scatter-Plot:

A screenshot of a computer screen

Description automatically generated