**Practical: 8**

**Aim: Write a program to implement k-means clustering on iris dataset**

**Code:**

from sklearn.cluster import KMeans

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.cluster import KMeans

from sklearn.metrics import silhouette\_score

from sklearn.preprocessing import MinMaxScaler

from sklearn.datasets import load\_iris

iris = pd.read\_csv("iris.csv")

x = iris.iloc[:, [0, 1, 2, 3]].values

print(iris.info())

print(iris[0:10])

# Frequency distribution of species"

iris\_outcome = pd.crosstab(index=iris["species"], # Make a crosstab

columns="count") # Name the count column

print(iris\_outcome)

iris\_setosa = iris.loc[iris["species"] == "Iris-setosa"]

iris\_virginica = iris.loc[iris["species"] == "Iris-virginica"]

iris\_versicolor = iris.loc[iris["species"] == "Iris-versicolor"]

sns.FacetGrid(iris, hue="species", size=3).map(

sns.distplot, "petal\_length").add\_legend()

sns.FacetGrid(iris, hue="species", size=3).map(

sns.distplot, "petal\_width").add\_legend()

sns.FacetGrid(iris, hue="species", size=3).map(

sns.distplot, "sepal\_length").add\_legend()

plt.show()

sns.set\_style("whitegrid")

sns.pairplot(iris, hue="species", size=3)

plt.show()

# Finding the optimum number of clusters for k-means classification

wcss = []

for i in range(1, 11):

kmeans = KMeans(n\_clusters=i, init='k-means++',

max\_iter=300, n\_init=10, random\_state=0)

kmeans.fit(x)

wcss.append(kmeans.inertia\_)

kmeans = KMeans(n\_clusters=3, init='k-means++',

max\_iter=300, n\_init=10, random\_state=0)

y\_kmeans = kmeans.fit\_predict(x)

# Visualising the clusters

plt.scatter(x[y\_kmeans == 0, 0], x[y\_kmeans == 0, 1],

s=100, c='r', label='Iris-setosa')

plt.scatter(x[y\_kmeans == 1, 0], x[y\_kmeans == 1, 1],

s=100, c='g', label='Iris-versicolour')

plt.scatter(x[y\_kmeans == 2, 0], x[y\_kmeans == 2, 1],

s=100, c='b', label='Iris-virginica')

# Plotting the centroids of the clusters

plt.scatter(kmeans.cluster\_centers\_[:, 0], kmeans.cluster\_centers\_[

:, 1], s=100, c='black', label='Centroids')

plt.legend()

fig = plt.figure(figsize=(15, 15))

ax = fig.add\_subplot(111, projection='3d')

plt.scatter(x[y\_kmeans == 0, 0], x[y\_kmeans == 0, 1],

s=100, c='r', label='Iris-setosa')

plt.scatter(x[y\_kmeans == 1, 0], x[y\_kmeans == 1, 1],

s=100, c='g', label='Iris-versicolour')

plt.scatter(x[y\_kmeans == 2, 0], x[y\_kmeans == 2, 1],

s=100, c='b', label='Iris-virginica')

# Plotting the centroids of the clusters

plt.scatter(kmeans.cluster\_centers\_[:, 0], kmeans.cluster\_centers\_[

:, 1], s=100, c='black', label='Centroids')

plt.show()

Iris dataframe information

Text, letter

Description automatically generated

Iris data Head with 10 rows

A screenshot of a computer

Description automatically generated with medium confidence

Frequency distribution of each class

Text, chat or text message

Description automatically generated

Chart, histogram

Description automatically generated

Chart, histogram

Description automatically generated

Chart, histogram

Description automatically generated

Chart, scatter chart

Description automatically generated

Chart, scatter chart

Description automatically generated

Chart, scatter chart

Description automatically generated