Ojas Patil

2019130048

TE COMPS

Batch C

Experiment 5

Aim: To train and test a machine learning model using K-Means algorithm

Theory:

K-Means Clustering is an unsupervised learning algorithm that is used to solve the clustering problems in machine learning or data science. It groups the unlabeled dataset into different clusters. Here K defines the number of pre-defined clusters that need to be created in the process, as if K=2, there will be two clusters, and for K=3, there will be three clusters, and so on.

How does K-Means algorithm work?

The working of the K-Means algorithm is explained in the below steps:

Step-1: Select the number K to decide the number of clusters.

Step-2: Select random K points or centroids. (It can be other from the input dataset).

Step-3: Assign each data point to their closest centroid, which will form the predefined K clusters.

Step-4: Calculate the variance and place a new centroid of each cluster.

Step-5: Repeat the third steps, which means reassign each datapoint to the new closest centroid of each cluster.

Step-6: If any reassignment occurs, then go to step-4 else go to FINISH.

Step-7: The model is ready.

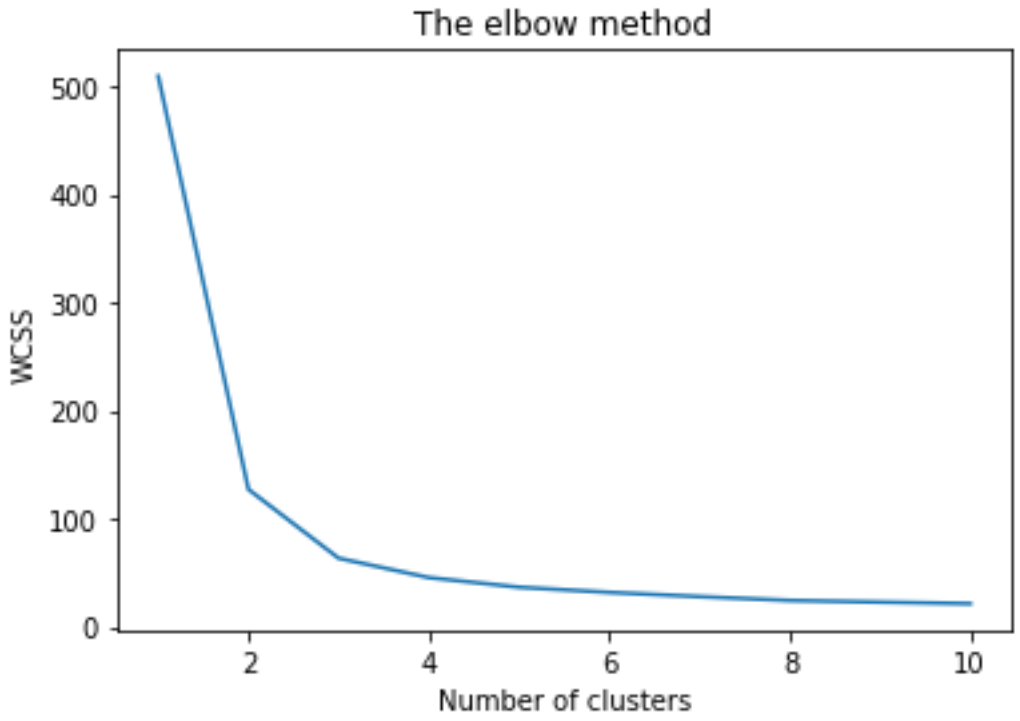
Code:

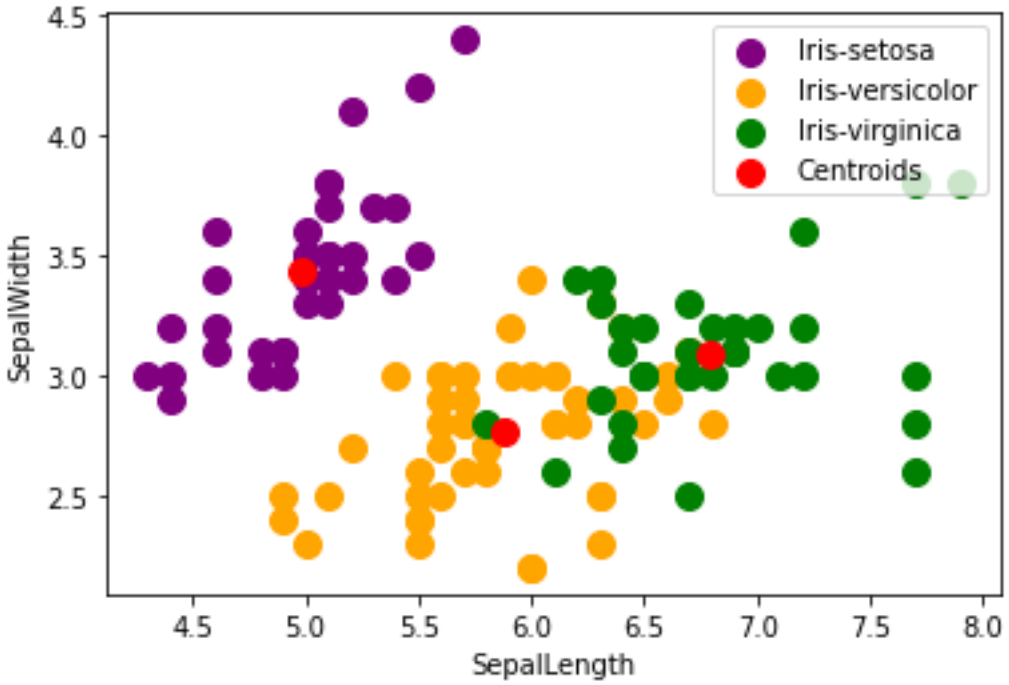
```
# %%
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.model_selection import train_test_split

# %%
df = pd.read_csv('Iris.csv')
df.head()
df.shape
df = df.drop(columns=['Id'], axis=1)
species = {'Iris-setosa': 0, 'Iris-versicolor': 1, 'Iris-virginica': 2}
df['SpeciesNo'] = df['Species']
df['SpeciesNo'] = [species[i] for i in df.SpeciesNo]
df.sample(5)

# %%
print(df.isnull().sum())
print(df.duplicated().sum())
x_train, x_test, y_train, y_test = train_test_split(df.iloc[:,:4],
df['SpeciesNo'], train_size=0.8, random_state=4)
df = pd.DataFrame(x_train)

# %%
kmeans = KMeans(n_clusters=3, random_state=0).fit(df)
kmeans.labels_
kmeans.cluster_centers_
print(kmeans.inertia_)
testPredict = kmeans.predict(x_test)
from sklearn.metrics import accuracy_score
accuracy_score(y_test, testPredict) * 100


# %%
x = df.values
y_kmeans = kmeans.fit_predict(x)
plt.scatter(x[y_kmeans == 0, 0], x[y_kmeans == 0, 1], s = 100, c = 'purple',
label = 'Iris-setosa')
plt.scatter(x[y_kmeans == 1, 0], x[y_kmeans == 1, 1], s = 100, c = 'orange',
label = 'Iris-versicolor')
plt.scatter(x[y_kmeans == 2, 0], x[y_kmeans == 2, 1], s = 100, c = 'green',
label = 'Iris-virginica')

#Plotting the centroids of the clusters
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:,1], s =
100, c = 'red', label = 'Centroids')
```

```
plt.legend()

# %%
lst = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, random_state=0).fit(df)
    kmeans.labels_
    kmeans.cluster_centers_
    testPredict = kmeans.predict(x_test)
    from sklearn.metrics import accuracy_score
    acc = accuracy_score(y_test, testPredict) * 100
    lst.append([i, acc, kmeans])

abc = max(lst, key=lambda x: x[1])
print(f"Accuracy is maximum when number of clusters is {abc[0]} which is
{abc[1]}")
kmeans = abc[2]
x = df.values
y_kmeans = kmeans.fit_predict(x)
plt.scatter(x[y_kmeans == 0, 0], x[y_kmeans == 0, 1], s = 100, c = 'purple',
label = 'Iris-setosa')
plt.scatter(x[y_kmeans == 1, 0], x[y_kmeans == 1, 1], s = 100, c = 'orange',
label = 'Iris-versicolor')
plt.scatter(x[y_kmeans == 2, 0], x[y_kmeans == 2, 1], s = 100, c = 'green',
label = 'Iris-virginica')

plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:,1], s =
100, c = 'red', label = 'Centroids')

plt.legend()

# %%
petallength = float(input("Enter petal length in cm: "))
sepallength = float(input("Enter sepal length in cm: "))
petalwidth = float(input("Enter petal width in cm: "))
sepalwidth = float(input("Enter sepal width in cm: "))

print(f"""
petalLength: {petallength},
sepallength: {sepallength},
petalwidth: {petalwidth},
sepalwidth: {sepalwidth}
""")

test = [[sepallength, sepalwidth, petallength, petalwidth]]
y = kmeans.predict(test)
if y[0] == 0:
    print('Iris-setosa')
```

```
elif y[0] == 1:
    print('Iris-versicolor')
elif y[0] == 2:
    print('Iris-virginica')
```

Output:






```
petalLength: 5.6,
sepallength: 6.4,
petalwidth: 2.2,
sepalwidth: 2.8


iris-virginica
```

Conclusion:

From above experiment, I learned about basics of K-Means algorithm. It is a centroid-based algorithm, where each cluster is associated with a centroid. The main aim of this algorithm is to minimize the sum of distances between the data point and their corresponding clusters.

It determines the best value for K center points or centroids by an iterative process and assigns each data point to its closest k-center. Those data points which are near to the particular k-center, create a cluster.

The algorithm takes the unlabeled dataset as input, divides the dataset into k-number of clusters, and repeats the process until it does not find the

best clusters. The value of k should be predetermined in this algorithm. Accuracy of algorithm varies with number of clusters selected.

GitHub: https://github.com/PatilOjas/AIML-Lab/tree/main/Exp%205