K-MEANS CLUSTERING: IRIS DATASET

In [5]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.cluster import KMeans
```

In [6]:
```
data=load_iris()
X=data.data
```

In [7]:
```
k=3

kmeans = KMeans(n_clusters=k, random_state=42)
kmeans.fit(X)
```

```
C:\ProgramData\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
C:\ProgramData\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

Out[7]:
```
KMeans(n_clusters=3, random_state=42)
```

In [8]:
```
cluster_labels = kmeans.labels_
```

In [9]:
```
x_coordinates = X[:, 0]  # Sepal Length (cm)
y_coordinates = X[:, 1]  # Sepal Width (cm)
colors = cluster_labels
colormap = 'viridis'

plt.scatter(x_coordinates, y_coordinates, c=colors, cmap=colormap)
plt.title('K-Means Clustering on Iris Dataset')
plt.xlabel('Sepal Length (cm)')
plt.ylabel('Sepal Width (cm)')
plt.show()
```

In [13]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_wine
from sklearn.cluster import KMeans
```

In [14]:
```
data=load_wine()
X=data.data
```

In [15]:
```
k=3

kmeans = KMeans(n_clusters=k, random_state=42)
kmeans.fit(X)
```

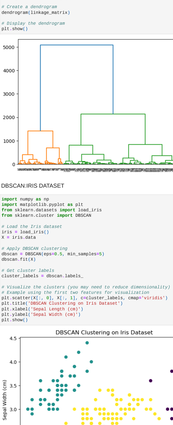
```
C:\ProgramData\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
C:\ProgramData\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

Out[15]:
```
KMeans(n_clusters=5, random_state=42)
```

In [21]:
```
cluster_labels = kmeans.labels_
```

In [22]:
```
x_coordinates = X[:, 0]  # Sepal Length (cm)
y_coordinates = X[:, 1]  # Sepal Width (cm)
colors = cluster_labels
colormap = 'viridis'

plt.scatter(x_coordinates, y_coordinates, c=colors, cmap=colormap)
plt.title('K-Means Clustering on Iris Dataset')
plt.xlabel('Sepal Length (cm)')
plt.ylabel('Sepal Width (cm)')
plt.show()
```



Hirachical :numbers

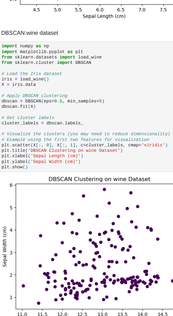
In [23]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.cluster.hierarchy import dendrogram, linkage

# Generate sample data (you can replace this with your own data)
np.random.seed(0)
data = np.random.rand(10, 2)

# Perform hierarchical clustering
linkage_matrix = linkage(data, method='ward')

# Create a dendrogram
dendrogram(linkage_matrix)

# Display the dendrogram
plt.show()
```



HIERACHICAL DATASETS:WINE

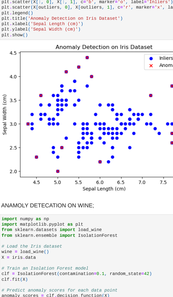
In [27]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.cluster.hierarchy import dendrogram, linkage
from sklearn.datasets import load_wine

# Load the wine dataset
wine = load_wine()
X = wine.data

# Perform hierarchical clustering
linkage_matrix = linkage(X, method='ward')

# Create a dendrogram
dendrogram(linkage_matrix)

# Display the dendrogram
plt.show()
```



DBSCAN:IRIS DATASET

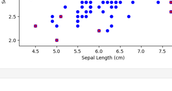
In [22]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.cluster import DBSCAN

# Load the Iris dataset
iris = load_iris()
X = iris.data

# Apply DBSCAN clustering
dbscan = DBSCAN(eps=0.5, min_samples=5)
dbscan.fit(X)

# Get cluster labels
cluster_labels = dbscan.labels_

# Visualize the clusters (you may need to reduce dimensionality)
# Example using the first two features for visualization
plt.scatter(X[:, 0], X[:, 1], c=cluster_labels, cmap='viridis')
plt.title('DBSCAN Clustering on Iris Dataset')
plt.xlabel('Sepal Length (cm)')
plt.ylabel('Sepal Width (cm)')
plt.show()
```



DBSCAN:wine dataset

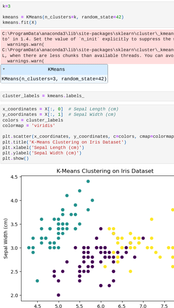
In [24]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_wine
from sklearn.cluster import DBSCAN

# Load the Iris dataset
iris = load_wine()
X = iris.data

# Apply DBSCAN clustering
dbscan = DBSCAN(eps=0.5, min_samples=5)
dbscan.fit(X)

# Get cluster labels
cluster_labels = dbscan.labels_

# Visualize the clusters (you may need to reduce dimensionality)
# Example using the first two features for visualization
plt.scatter(X[:, 0], X[:, 1], c=cluster_labels, cmap='viridis')
plt.title('DBSCAN Clustering on wine Dataset')
plt.xlabel('Sepal Length (cm)')
plt.ylabel('Sepal Width (cm)')
plt.show()
```



ANAMOLY DETECATION ON IRIS

In [25]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.ensemble import IsolationForest

# Load the Iris dataset
iris = load_iris()
X = iris.data

# Train an Isolation Forest model
clf = IsolationForest(contamination=0.1, random_state=42)
clf.fit(X)

# Predict anomaly scores for each data point
anomaly_scores = clf.decision_function(X)

# Identify anomalies (outliers)
outliers = np.where(clf.predict(X) == -1)

# Visualize the anomalies
plt.scatter(X[:, 0], X[:, 1], c='b', marker='o', label='Inliers')
plt.scatter(X[outliers, 0], X[outliers, 1], c='r', marker='x', label='Anomalies')
plt.legend()
plt.title('Anomaly Detection on Iris Dataset')
plt.xlabel('Sepal Length (cm)')
plt.ylabel('Sepal Width (cm)')
plt.show()
```


ANAMOLY DETECATION ON WINE:

In [27]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_wine
from sklearn.ensemble import IsolationForest

# Load the Iris dataset
wine = load_wine()
X = iris.data

# Train an Isolation Forest model
clf = IsolationForest(contamination=0.1, random_state=42)
clf.fit(X)

# Predict anomaly scores for each data point
anomaly_scores = clf.decision_function(X)

# Identify anomalies (outliers)
outliers = np.where(clf.predict(X) == -1)

# Visualize the anomalies
plt.scatter(X[:, 0], X[:, 1], c='b', marker='o', label='Inliers')
plt.scatter(X[outliers, 0], X[outliers, 1], c='r', marker='x', label='Anomalies')
plt.legend()
plt.title('Anomaly Detection on wine Dataset')
plt.xlabel('Sepal Length (cm)')
plt.ylabel('Sepal Width (cm)')
plt.show()
```


In [ ]: