EXPERIMENT – 10

AIM: To Demonstrate and Analyze the performance of three Clustering Algorithms: k-Means, Agglomerative Clustering, and DBSCAN, on the standard dataset.

DESCRIPTION:

Clustering is an unsupervised learning technique that groups data points based on their similarities. Here, we use the Iris dataset, which consists of features of three species of iris flowers (Setosa, Versicolor, and Virginica). The goal is to apply k-Means, Agglomerative Clustering, and DBSCAN to classify these flowers into clusters and evaluate their clustering performance.

1. **k-Means**: Partitions the dataset into k clusters by minimizing the distance of data points from the centroid of each cluster.
   k-Means is a popular partition-based clustering algorithm that works by dividing a dataset into a predefined number of clusters (denoted as **k**). This method works well for datasets with spherical clusters and assumes clusters are roughly the same size.

2. **Agglomerative Clustering**: A hierarchical clustering method that starts by treating each data point as a separate cluster, then repeatedly merges the closest clusters until reaching the desired number of clusters.
   Agglomerative Clustering is a type of **hierarchical clustering**, which creates a hierarchy of clusters rather than dividing the data into a fixed number of clusters. It's a **bottom-up** approach, starting with each data point as its own cluster and merging clusters until only a single cluster remains or until reaching the desired number of clusters.

3. **DBSCAN**: A density-based clustering method that groups points closely packed together, marking points in low-density regions as outliers.
   DBSCAN is a **density-based clustering** algorithm that groups data points based on regions of high density and identifies outliers as points in low-density regions. Unlike k-Means or Agglomerative Clustering, DBSCAN does not require the number of clusters to be specified in advance, making it particularly useful for datasets with irregular cluster shapes or noise.

CODE:

```
import pandas as pd # import pandas for data manipulation
from sklearn import datasets
from sklearn.cluster import KMeans, AgglomerativeClustering, DBSCAN
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import silhouette_score, adjusted_rand_score, accuracy_score
from scipy.stats import mode
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Load the iris dataset
iris = datasets.load_iris()

df = pd.DataFrame(iris.data, columns=iris.feature_names)
df.head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

```
X = iris.data
X
```

```
array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2],
       [5.4, 3.9, 1.7, 0.4],
       [4.6, 3.4, 1.4, 0.3],
       [5. , 3.4, 1.5, 0.2],
       [4.4, 2.9, 1.4, 0.2],
```

```
y_true = iris.target
y_true
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

```
# Setting up clustering models
kmeans = KMeans(n_clusters=3, random_state=42)
agg_clustering = AgglomerativeClustering(n_clusters=3)
dbscan = DBSCAN(eps=0.5, min_samples=5)
```

```
# Fitting the models and predicting clusters
kmeans_labels = kmeans.fit_predict(X_scaled)
agg_labels = agg_clustering.fit_predict(X_scaled)
dbscan_labels = dbscan.fit_predict(X_scaled)
```

```
#Define a function to calculate accuracy by mapping cluster labels to true labels
def calculate_cluster_accuracy(y_true, cluster_labels):
    labels = np.zeros_like(cluster_labels)
    for i in np.unique(cluster_labels):
        mask = cluster_labels == i
        labels[mask] = mode(y_true[mask])[0]
    return accuracy_score(y_true, labels)
```

```
kmeans_accuracy = calculate_cluster_accuracy(y_true, kmeans_labels)
agg_accuracy = calculate_cluster_accuracy(y_true, agg_labels)
dbscan_accuracy = calculate_cluster_accuracy(y_true, dbscan_labels)
```

```
print("Silhouette Score for k-Means:", silhouette_score(X_scaled, kmeans_labels))
print("Accuracy for k-Means:", kmeans_accuracy)

print("\nSilhouette Score for Agglomerative Clustering:", silhouette_score(X_scaled, agg_labels
print("Accuracy for Agglomerative Clustering:", agg_accuracy)

# DBSCAN often has outliers labeled as -1, so we need to filter both data and labels
core_samples_mask = np.zeros_like(dbscan_labels, dtype=bool)
core_samples_mask[dbscan_labels != -1] = True
dbscan_silhouette = silhouette_score(X_scaled[core_samples_mask], dbscan_labels[core_samples_ma

print("\nSilhouette Score for DBSCAN (excluding noise):", dbscan_silhouette)
print("Accuracy for DBSCAN:", dbscan_accuracy)
```

```
Silhouette Score for k-Means: 0.4798814508199817
Accuracy for k-Means: 0.6666666666666666

Silhouette Score for Agglomerative Clustering: 0.4466890410285909
Accuracy for Agglomerative Clustering: 0.8266666666666667

Silhouette Score for DBSCAN (excluding noise): 0.6558885287002016
Accuracy for DBSCAN: 0.68
```

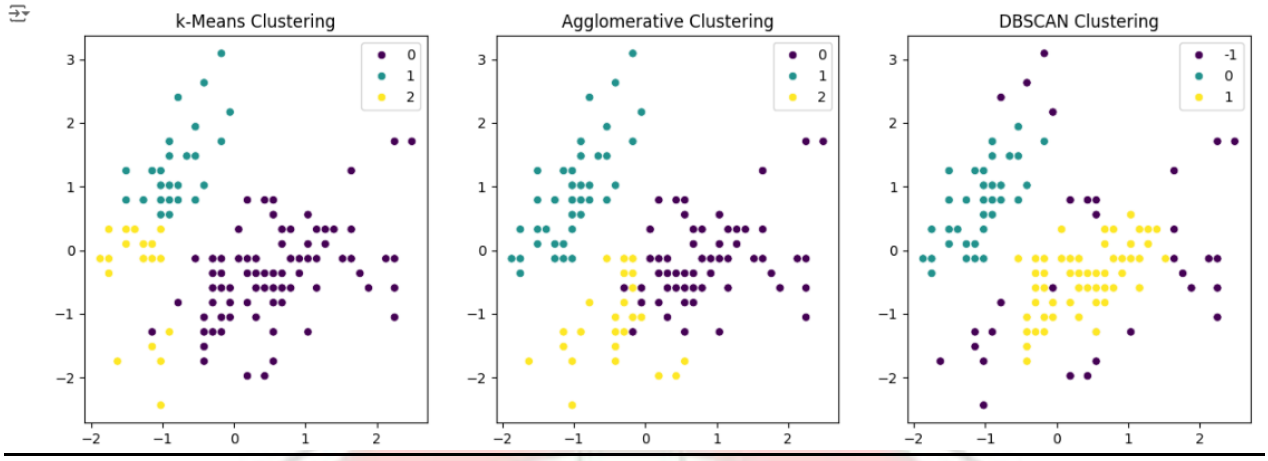
```
fig, axs = plt.subplots(1, 3, figsize=(15, 5))

# k-Means plot
sns.scatterplot(x=X_scaled[:, 0], y=X_scaled[:, 1], hue=kmeans_labels, palette="viridis", ax=axs[0])
axs[0].set_title("k-Means Clustering")

# Agglomerative Clustering plot
sns.scatterplot(x=X_scaled[:, 0], y=X_scaled[:, 1], hue=agg_labels, palette="viridis", ax=axs[1])
axs[1].set_title("Agglomerative Clustering")

# DBSCAN plot
sns.scatterplot(x=X_scaled[:, 0], y=X_scaled[:, 1], hue=dbscan_labels, palette="viridis", ax=axs[2])
axs[2].set_title("DBSCAN Clustering")

plt.show()
```



OUTPUT ANALYSIS:

k-Means Clustering:

- **Silhouette Score**: 0.4799 – This indicates a moderate level of separation between clusters. While the score is not extremely high, it suggests that k-Means has formed distinguishable, although not highly compact, clusters.
- **Accuracy**: 66.67% – This accuracy reflects a moderate alignment between the k-Means clusters and the true Iris species labels. It shows that while k-Means has successfully identified some clusters, it does not fully align with the actual classes, which could be due to the spherical assumption k-Means makes.

Agglomerative Clustering:

- **Silhouette Score**: 0.4467 – Similar to k-Means, this score suggests moderate cluster separation but not optimal compactness, which is expected as Agglomerative Clustering does not explicitly optimize for compact clusters.
- **Accuracy**: 82.67% – This higher accuracy compared to k-Means suggests that Agglomerative Clustering aligns better with the true Iris species. It effectively captures the relationships between data points, reflecting the dataset's underlying hierarchical structure.

DBSCAN:

- **Silhouette Score (excluding noise)**: 0.6559 – This high score indicates that DBSCAN has successfully identified well-separated clusters for the majority of the points, as the algorithm is adept at finding clusters with varying shapes and densities.
- **Accuracy**: 68% – DBSCAN shows a modest alignment with the true labels, with an accuracy higher than k-Means but lower than Agglomerative Clustering. This is likely due to DBSCAN's identification of outliers as noise, which can impact the alignment with true labels, especially in datasets where all points belong to clusters.