EXPERIMENT NO:7

Aim: To implement different clustering algorithms.
Problem Statement: a) Clustering algorithm for unsupervised classification (K-means, density based (DBSCAN), Hierarchical clustering)
b) Plot the cluster data and show mathematical steps.

Theory:

1. **Importing Libraries**

```
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.preprocessing import LabelEncoder
from sklearn.cluster import DBSCAN
import scipy.cluster.hierarchy as sch
import matplotlib.pyplot as plt
import seaborn as sns
```

2. **Importing dataset**

```
df=pd.read_csv('diabetes_dataset_with_notes.csv')
df = pd.DataFrame(df)
```

3. **Transformation**

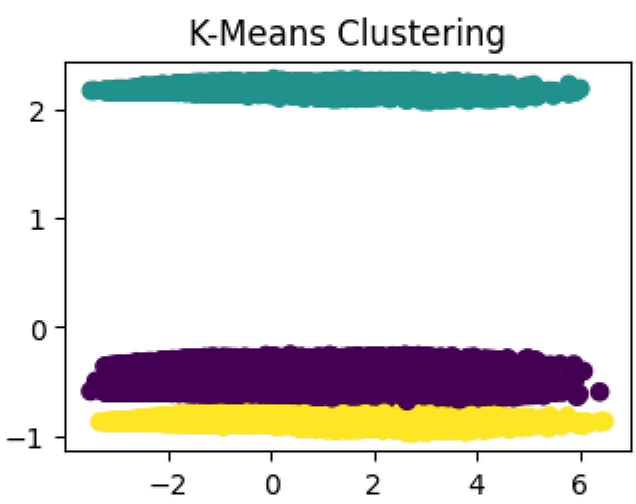
```
label_encoder = LabelEncoder()
df['gender_encoded'] = label_encoder.fit_transform(df['gender'])
df['smoking_history_encoded'] = label_encoder.fit_transform(df['smoking_history'])

features = ['age', 'race:AfricanAmerican', 'race:Asian', 'race:Caucasian', 'race:Hispanic',
            'race:Other', 'hypertension', 'heart_disease', 'smoking_history_encoded', 'bmi',
            'hbA1c_level', 'blood_glucose_level']

scaler = StandardScaler()
df_scaled = scaler.fit_transform(df[features])
```

4. **K Means** :-K-Means Clustering: A centroid-based clustering algorithm that partitions data into k clusters by minimizing the distance between data points and their respective cluster centers. It works well with well-separated clusters but assumes a spherical shape.

```
kmeans = KMeans(n_clusters=3, random_state=42)
df['kmeans_cluster'] = kmeans.fit_predict(df_scaled)

pca = PCA(n_components=2)
df_pca = pca.fit_transform(df_scaled)

plt.subplot(2, 2, 1)
plt.scatter(df_pca[:, 0], df_pca[:, 1], c=df['kmeans_cluster'], cmap='viridis')
plt.title("K-Means Clustering")

plt.tight_layout()
plt.show()
```

This code performs K-Means clustering on a dataset with 3 clusters, assigns cluster labels, and visualizes the results using PCA (Principal Component Analysis) for dimensionality reduction.

- **K-Means Clustering:** It fits the scaled dataset and assigns each data point a cluster label (0, 1, or 2), which is stored in df['kmeans_cluster'].
- **PCA Transformation:** The dataset is reduced to 2D for visualization.
- **Plotting:** A scatter plot is created using the first two PCA components, with points colored based on their assigned clusters.



- Successfully identified three distinct clusters.
- The clusters are horizontally aligned, suggesting that one or two dominant features are driving the segmentation.

- Works well for structured, well-separated data but may struggle with complex patterns.

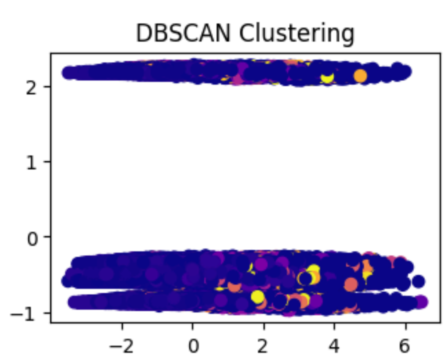
5. **DBSCAN Clustering:-**DBSCAN (Density-Based Spatial Clustering of Applications with Noise): A clustering algorithm that groups data points based on density rather than predefined clusters. It can detect arbitrarily shaped clusters and outliers but is sensitive to parameter selection (eps, min_samples).

```
dbscan = DBSCAN(eps=0.5, min_samples=2)
df['dbscan_cluster'] = dbscan.fit_predict(df_scaled)

plt.subplot(2, 2, 2)
plt.scatter(df_pca[:, 0], df_pca[:, 1], c=df['dbscan_cluster'], cmap='plasma')
plt.title("DBSCAN Clustering")

plt.tight_layout()
plt.show()
```

This code applies DBSCAN clustering on a dataset and visualizes the results using PCA for dimensionality reduction.

Steps:

1. **DBSCAN Clustering:**
   - DBSCAN(eps=0.5, min_samples=2): Groups points based on density.
   - Noise points (outliers) are labeled as -1.
   - Cluster labels are stored in df['dbscan_cluster'].
2. **PCA for Visualization:**
   - Reduces data to 2D for plotting.
3. **Scatter Plot:**
   - Points are colored by their DBSCAN cluster labels using the 'plasma' colormap.
   - Outliers appear as separate scattered points.



- Identified dense regions, but the separation is not as clear as K-Means.
- Some points were classified as noise (outliers).
- Suitable for non-linear and arbitrarily shaped clusters, but parameter tuning (eps, min_samples) is crucial for better results.

Conclusion:

- K-Means performed better in this case, successfully grouping data into well-defined clusters.
- DBSCAN struggled with clear separation and produced noisy clusters, likely due to dataset structure or parameter settings.
- The choice of clustering algorithm depends on data distribution – K-Means is ideal for structured clusters, while DBSCAN is useful for detecting complex patterns and outliers.