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.Import Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.cluster import KMeans, DBSCAN, AgglomerativeClustering
from sklearn.decomposition import PCA
from scipy.cluster.hierarchy import dendrogram, linkage
```

2.Loading Dataset

```
# Load dataset
df = pd.read_csv("/content/Loan_default.csv")
df = pd.DataFrame(df)
df.head()
```

	LoanID	Age	Income	LoanAmount	CreditScore	MonthsEmployed	NumCre
0	I38PQUQS96	56	85994	50587	520	80	
1	HPSK72WA7R	69	50432	124440	458	15	
2	C1OZ6DPJ8Y	46	84208	129188	451	26	
3	V2KKSFM3UN	32	31713	44799	743	0	
4	EY08JDHTZP	60	20437	9139	633	8	

3.Transformation

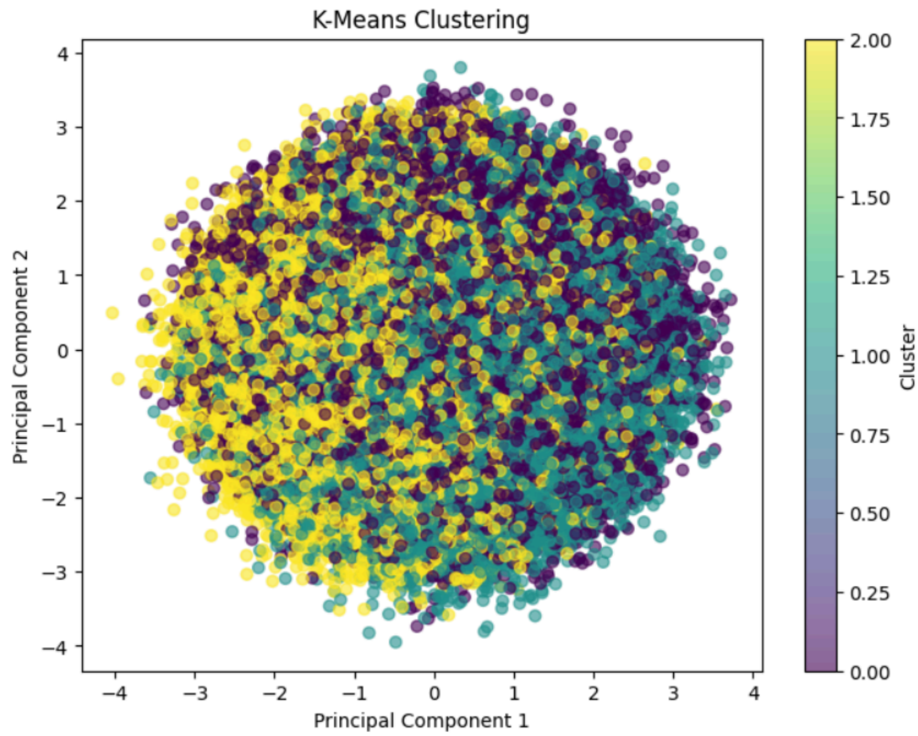
```
from sklearn.preprocessing import LabelEncoder, StandardScaler

label_encoder = LabelEncoder()


df['Education_encoded'] = label_encoder.fit_transform(df['Education'])
df['EmploymentType_encoded'] = label_encoder.fit_transform(df['EmploymentType'])
df['MaritalStatus_encoded'] = label_encoder.fit_transform(df['MaritalStatus'])
df['LoanPurpose_encoded'] = label_encoder.fit_transform(df['LoanPurpose'])


features = [
    'Age', 'Income', 'LoanAmount', 'CreditScore', 'MonthsEmployed', 'NumCreditLines',
    'InterestRate', 'LoanTerm', 'DTIRatio', 'Education_encoded',
    'EmploymentType_encoded', 'MaritalStatus_encoded', 'LoanPurpose_encoded'
]


scaler = StandardScaler()
df_scaled = scaler.fit_transform(df[features])
df_scaled = pd.DataFrame(df_scaled, columns=features)
df_scaled.head()
```

	Age	Income	LoanAmount	CreditScore	MonthsEmployed	NumCreditLines	InterestRate	LoanTerm	DTIRatio	Education_encoded	Empl
0	0.833990	0.089693	-1.086833	-0.341492	0.590533	1.341937	0.261771	-0.001526	-0.260753	-1.335708	
1	1.701221	-0.823021	-0.044309	-0.731666	-1.285731	-1.343791	-1.308350	1.412793	0.778585	0.451884	
2	0.166888	0.043854	0.022715	-0.775718	-0.968209	0.446694	1.156831	-0.708685	-0.823728	0.451884	
3	-0.767053	-1.303452	-1.168538	1.061875	-1.718715	0.446694	-0.967805	-0.708685	-1.170174	-0.441912	
4	1.100830	-1.592855	-1.671921	0.369631	-1.487790	1.341937	-1.052188	0.705634	0.995114	-1.335708	

K-Means:

K-Means is 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 for **well-separated, spherical clusters** but may struggle with irregularly shaped distributions.

```python
kmeans = KMeans(n_clusters=3, random_state=42, n_init=10)
df['kmeans_cluster'] = kmeans.fit_predict(df_scaled)


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


plt.figure(figsize=(8, 6))
plt.scatter(df_pca[:, 0], df_pca[:, 1], c=df['kmeans_cluster'], cmap='viridis', alpha=0.6)
plt.title("K-Means Clustering")
plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.colorbar(label="Cluster")
plt.show()
```



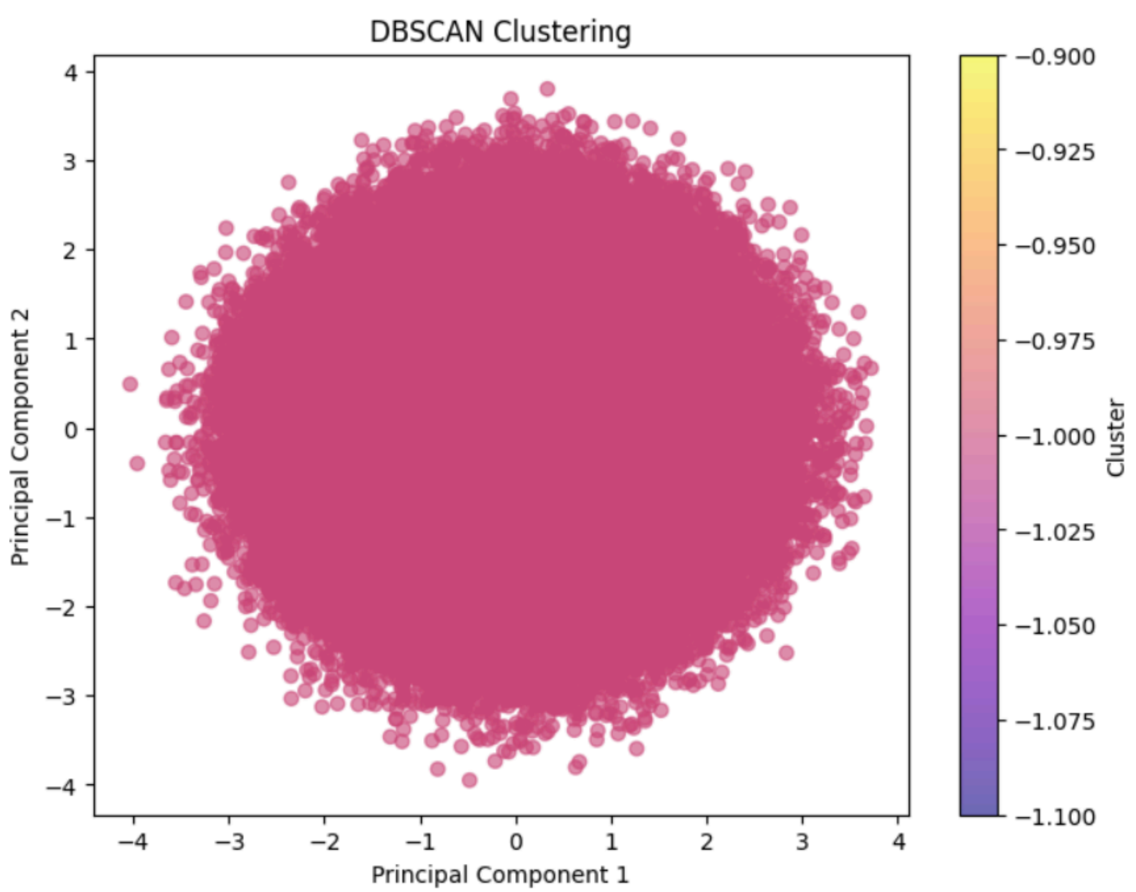
1. **Clusters Overlap** – The K-Means algorithm did not form well-separated clusters, indicating overlapping data distributions.
2. **K-Means Limitation** – The assumption of spherical clusters may not fit the dataset's structure.
3. **Alternative Methods Needed** – DBSCAN or Hierarchical Clustering might work better for improved separation.

DBSCAN:

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a clustering algorithm that groups points based on **density** rather than distance to a centroid. It can **identify**

arbitrarily shaped clusters and detect **noise (outliers)**, making it more robust than K-Means for complex datasets.

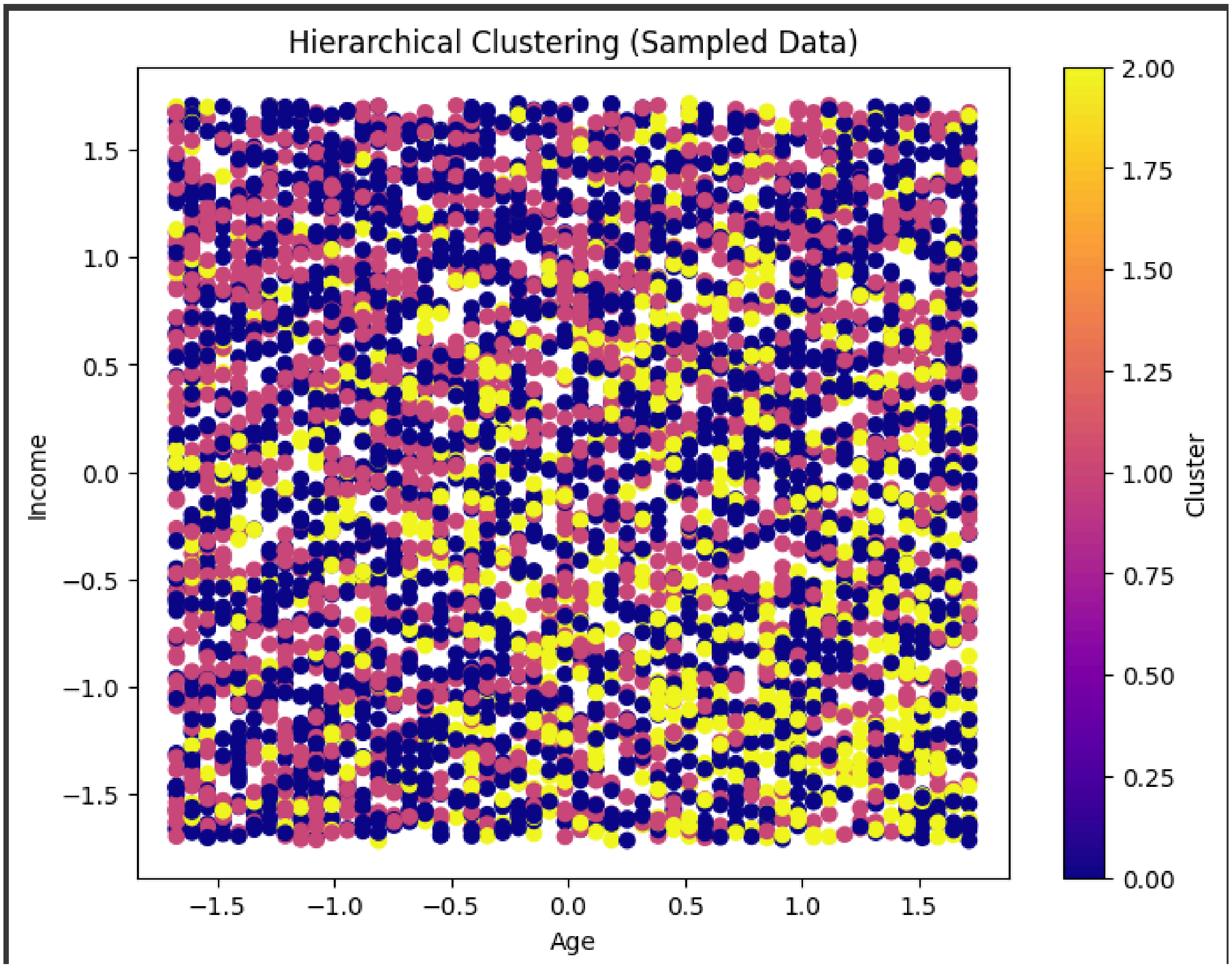
```
from sklearn.cluster import DBSCAN
import matplotlib.pyplot as plt

# Apply DBSCAN clustering
dbscan = DBSCAN(eps=0.5, min_samples=5)
df['dbscan_cluster'] = dbscan.fit_predict(df_scaled)


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


plt.figure(figsize=(8, 6))
plt.scatter(df_pca[:, 0], df_pca[:, 1], c=df['dbscan_cluster'], cmap='plasma', alpha=0.6)
plt.title("DBSCAN Clustering")
plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.colorbar(label="Cluster")
plt.show()
```



1. **No Clear Clusters** – DBSCAN failed to identify distinct groups, classifying most data points into a single cluster.
2. **Parameter Issue** – The chosen eps and min_samples may be inappropriate, leading to poor clustering results.
3. **Possible Fixes** – Adjust eps and min_samples, check data scaling, or try alternative clustering methods like K-Means or Hierarchical Clustering.

3.Hierarchical Clustering

```python
# Encode categorical columns
le = LabelEncoder()
df['Education_encoded'] = le.fit_transform(df['Education'])
df['EmploymentType_encoded'] = le.fit_transform(df['EmploymentType'])
df['MaritalStatus_encoded'] = le.fit_transform(df['MaritalStatus'])

# Select relevant features (numeric + encoded)
features = [
    'Age', 'Income', 'LoanAmount', 'CreditScore',
    'Education_encoded', 'EmploymentType_encoded', 'MaritalStatus_encoded'
]

# Drop rows with missing values in selected features
df_clean = df.dropna(subset=features)

# Standardize the features
scaler = StandardScaler()
df_scaled = scaler.fit_transform(df_clean[features])

# Take a sample (optional, for speed)
sample_size = 5000
df_sample = df_clean[features].sample(n=sample_size, random_state=42)
df_sample_scaled = scaler.fit_transform(df_sample)

# Apply Hierarchical Clustering
hc = AgglomerativeClustering(n_clusters=3, linkage='ward')
labels = hc.fit_predict(df_sample_scaled)

# Assign cluster labels to sample
df_sample['hc_cluster'] = labels

# Plot clusters using first two features
plt.figure(figsize=(8, 6))
plt.scatter(df_sample_scaled[:, 0], df_sample_scaled[:, 1], c=labels, cmap='plasma')
plt.xlabel(features[0])
plt.ylabel(features[1])
plt.title("Hierarchical Clustering (Sampled Data)")
plt.colorbar(label='Cluster')
plt.show()
```



Conclusion:

K-Means Clustering successfully grouped the data but assumed spherical clusters, which may not always represent the true structure of the dataset.

DBSCAN Clustering struggled to form meaningful clusters, likely due to improper parameter selection or the dataset's characteristics.