Unsupervised Learning on Mice Genetics Dataset

Objective

The objective of this project is to apply unsupervised learning algorithms to the Mice Genetics dataset to predict clusters and compare them with the target labels. The goal is to identify the best clustering model that aligns well with the known categories.

Description of the Data

The Mice Genetics dataset consists of 80 columns, with 3 non-numerical columns. The dataset includes various protein expression levels measured in mice, which are used to identify different genetic backgrounds or conditions.

Data Exploration and Cleaning

1. **Initial Exploration**:
    - The dataset was initially explored to understand the distribution and types of data.
    - Non-numerical columns were identified and excluded from the analysis.
2. **Handling Missing Values**:
    - All rows with missing values (NA) were dropped to simplify the analysis.
3. **Principal Component Analysis (PCA)**:
    - PCA was performed to reduce the dimensionality of the data to 2 components for easier visualization and clustering.
    - The PCA-transformed data was used for fitting the clustering models.

Summary of Different Models Trained

Three unsupervised learning algorithms were applied to the PCA-transformed data:

1. **DBSCAN (Density-Based Spatial Clustering of Applications with Noise)**:
    - **Parameter Search**: A for loop was used to search for the best parameters (eps and min_samples).
    - **Performance**: DBSCAN performed the best among the three models, effectively identifying clusters and handling noise points.
2. **Agglomerative Clustering**:
    - **Parameter Search**: Different linkage methods and the number of clusters were tested.
    - **Performance**: Agglomerative clustering provided reasonable results but was less effective than DBSCAN.
3. **K-Means Clustering**:

- **Parameter Search**: The number of clusters (K) was plotted using the elbow method to find the optimal value which was 4.
- **Performance**: K-Means clustering was straightforward but did not perform as well as DBSCAN in capturing the underlying structure of the data.

Best Model: DBSCAN

- **Best Parameters**: The optimal parameters for DBSCAN were found to be eps=0.7 and min_samples=2.
- **Evaluation Metrics**:
  - **Silhouette Score**: 0.60

Final Results

- **Clusters Identified**: DBSCAN successfully identified meaningful clusters in the PCA-transformed data.
- **Comparison with Target Labels**: The clusters were compared with the target labels using Silhouette Score, showing a strong alignment.
- **Visualization**: The clusters were visualized using scatter plots, highlighting the separation and structure of the clusters

```
!pip install ucimlrepo

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

from ucimlrepo import fetch_ucirepo


# fetch dataset

mice_protein_expression = fetch_ucirepo(id=342)


# data (as pandas dataframes)

X = mice_protein_expression.data.features

y = mice_protein_expression.data.targets


# metadata

print(mice_protein_expression.metadata)


# variable information

print(mice_protein_expression.variables)
```

{'uci_id': 342, 'name': 'Mice Protein Expression', 'repository_url': 'https://archive.ics.uci.edu/dataset/342/mice+protein+expression', 'data_url': 'https://archive.ics.uci.edu/static/public/342/data.csv', 'abstract': 'Expression levels of 77 proteins measured in the cerebral cortex of 8 classes of control and Down syndrome mice exposed to context fear conditioning, a task used to assess associative learning.', 'area': 'Biology', 'tasks': ['Classification', 'Clustering'], 'characteristics': ['Multivariate'], 'num_instances': 1080, 'num_features': 80, 'feature_types': ['Real'], 'demographics': [], 'target_col': ['class'], 'index_col': ['MouseID'], 'has_missing_values': 'yes', 'missing_values_symbol': 'NaN', 'year_of_dataset_creation': 2015, 'last_updated': 'Tue Apr 16 2024', 'dataset_doi': '10.24432/C50S3Z', 'creators': ['Clara Higuera', 'Katheleen Gardiner', 'Krzysztof Cios'], 'intro_paper': {'title': 'Self-Organizing Feature Maps Identify Proteins Critical to Learning in a Mouse Model of Down Syndrome', 'authors': 'C. Higuera, K. Gardiner, K. Cios', 'published_in': 'PLoS ONE', 'year': 2015, 'url': 'https://www.semanticscholar.org/paper/5c5754b02a4f2f36ccf8cdda78059cdb19860532', 'doi':

'10.1371/journal.pone.0129126'}, 'additional_info': {'summary': 'The data set consists of the expression levels of 77 proteins/protein modifications that produced detectable signals in the nuclear fraction of cortex. There are 38 control mice and 34 trisomic mice (Down syndrome), for a total of 72 mice. In the experiments, 15 measurements were registered of each protein per sample/mouse. Therefore, for control mice, there are 38x15, or 570 measurements, and for trisomic mice, there are 34x15, or 510 measurements. The dataset contains a total of 1080 measurements per protein. Each measurement can be considered as an independent sample/mouse.\r\n\r\nThe eight classes of mice are described based on features such as genotype, behavior and treatment. According to genotype, mice can be control or trisomic. According to behavior, some mice have been stimulated to learn (context-shock) and others have not (shock-context) and in order to assess the effect of the drug memantine in recovering the ability to learn in trisomic mice, some mice have been injected with the drug and others have not.\r\n\r\nClasses:\r\nc-CS-s: control mice, stimulated to learn, injected with saline (9 mice)\r\nc-CS-m: control mice, stimulated to learn, injected with memantine (10 mice)\r\nc-SC-s: control mice, not stimulated to learn, injected with saline (9 mice)\r\nc-SC-m: control mice, not stimulated to learn, injected with memantine (10 mice)\r\n\r\nt-CS-s: trisomy mice, stimulated to learn, injected with saline (7 mice)\r\nt-CS-m: trisomy mice, stimulated to learn, injected with memantine (9 mice)\r\nt-SC-s: trisomy mice, not stimulated to learn, injected with saline (9 mice)\r\nt-SC-m: trisomy mice, not stimulated to learn, injected with memantine (9 mice)\r\n\r\nThe aim is to identify subsets of proteins that are discriminant between the classes.\r\n', 'purpose': None, 'funded_by': None, 'instances_represent': None, 'recommended_data_splits': None, 'sensitive_data': None, 'preprocessing_description': None, 'variable_info': '1 Mouse ID \r\n2..78 Values of expression levels of 77 proteins; the names of proteins are followed by â€œ_nâ€\x9d indicating that they were measured in the nuclear fraction. For example: DYRK1A_n\r\n79 Genotype: control (c) or trisomy (t)\r\n80 Treatment type: memantine (m) or saline (s)\r\n81 Behavior: context-shock (CS) or shock-context (SC)\r\n82 Class: c-CS-s, c-CS-m, c-SC-s, c-SC-m, t-CS-s, t-CS-m, t-SC-s, t-SC-m \r\n', 'citation': None}}

	name	role	type	...	description	units	missing_values
0	MouseID	ID	Categorical	...	None	None	no
1	DYRK1A_N	Feature	Continuous	...	None	None	yes
2	ITSN1_N	Feature	Continuous	...	None	None	no
3	BDNF_N	Feature	Continuous	...	None	None	yes
4	NR1_N	Feature	Continuous	...	None	None	no
..	...	...	...	...	...	...	...
77	CaNA_N	Feature	Continuous	...	None	None	no
78	Genotype	Feature	Categorical	...	None	None	no
79	Treatment	Feature	Categorical	...	None	None	no
80	Behavior	Feature	Categorical	...	None	None	no

81 class Target Categorical ... None None no

[82 rows x 7 columns]

X.head()

y

	class
0	c-CS-m
1	c-CS-m
2	c-CS-m
3	c-CS-m
4	c-CS-m
...	...
1075	t-SC-s
1076	t-SC-s
1077	t-SC-s
1078	t-SC-s
1079	t-SC-s

1080 rows × 1 columns

X.describe()

X.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 1080 entries, 0 to 1079

Data columns (total 80 columns):

#	Column	Non-Null Count	Dtype
0	DYRK1A_N	1077 non-null	float64
1	ITSN1_N	1077 non-null	float64
2	BDNF_N	1077 non-null	float64
3	NR1_N	1077 non-null	float64
4	NR2A_N	1077 non-null	float64
5	pAKT_N	1077 non-null	float64
6	pBRAF_N	1077 non-null	float64
7	pCAMKII_N	1077 non-null	float64
8	pCREB_N	1077 non-null	float64
9	pELK_N	1077 non-null	float64
10	pERK_N	1077 non-null	float64
11	pJNK_N	1077 non-null	float64
12	PKCA_N	1077 non-null	float64
13	pMEK_N	1077 non-null	float64
14	pNR1_N	1077 non-null	float64
15	pNR2A_N	1077 non-null	float64
16	pNR2B_N	1077 non-null	float64
17	pPKCAB_N	1077 non-null	float64
18	pRSK_N	1077 non-null	float64
19	AKT_N	1077 non-null	float64
20	BRAF_N	1077 non-null	float64
21	CAMKII_N	1077 non-null	float64
22	CREB_N	1077 non-null	float64
23	ELK_N	1062 non-null	float64

24 ERK_N 1077 non-null float64

25 GSK3B_N 1077 non-null float64

26 JNK_N 1077 non-null float64

27 MEK_N 1073 non-null float64

28 TRKA_N 1077 non-null float64

29 RSK_N 1077 non-null float64

30 APP_N 1077 non-null float64

31 Bcatenin_N 1062 non-null float64

32 SOD1_N 1077 non-null float64

33 MTOR_N 1077 non-null float64

34 P38_N 1077 non-null float64

35 pMTOR_N 1077 non-null float64

36 DSCR1_N 1077 non-null float64

37 AMPKA_N 1077 non-null float64

38 NR2B_N 1077 non-null float64

39 pNUMB_N 1077 non-null float64

40 RAPTOR_N 1077 non-null float64

41 TIAM1_N 1077 non-null float64

42 pP70S6_N 1077 non-null float64

43 NUMB_N 1080 non-null float64

44 P70S6_N 1080 non-null float64

45 pGSK3B_N 1080 non-null float64

46 pPKCG_N 1080 non-null float64

47 CDK5_N 1080 non-null float64

48 S6_N 1080 non-null float64

49 ADARB1_N 1080 non-null float64

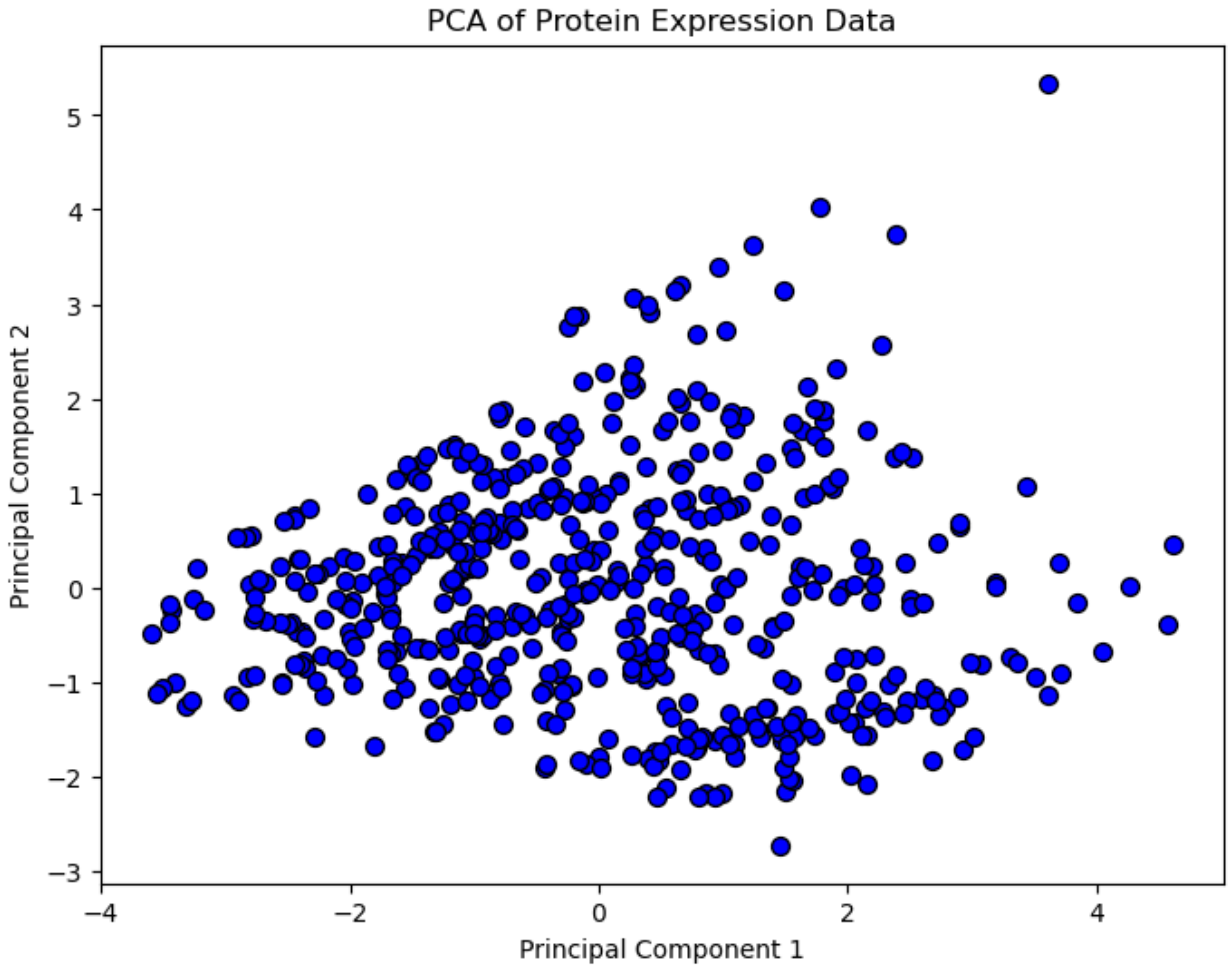
50 AcetylH3K9_N 1080 non-null float64

51 RRP1_N 1080 non-null float64

52 BAX_N 1080 non-null float64

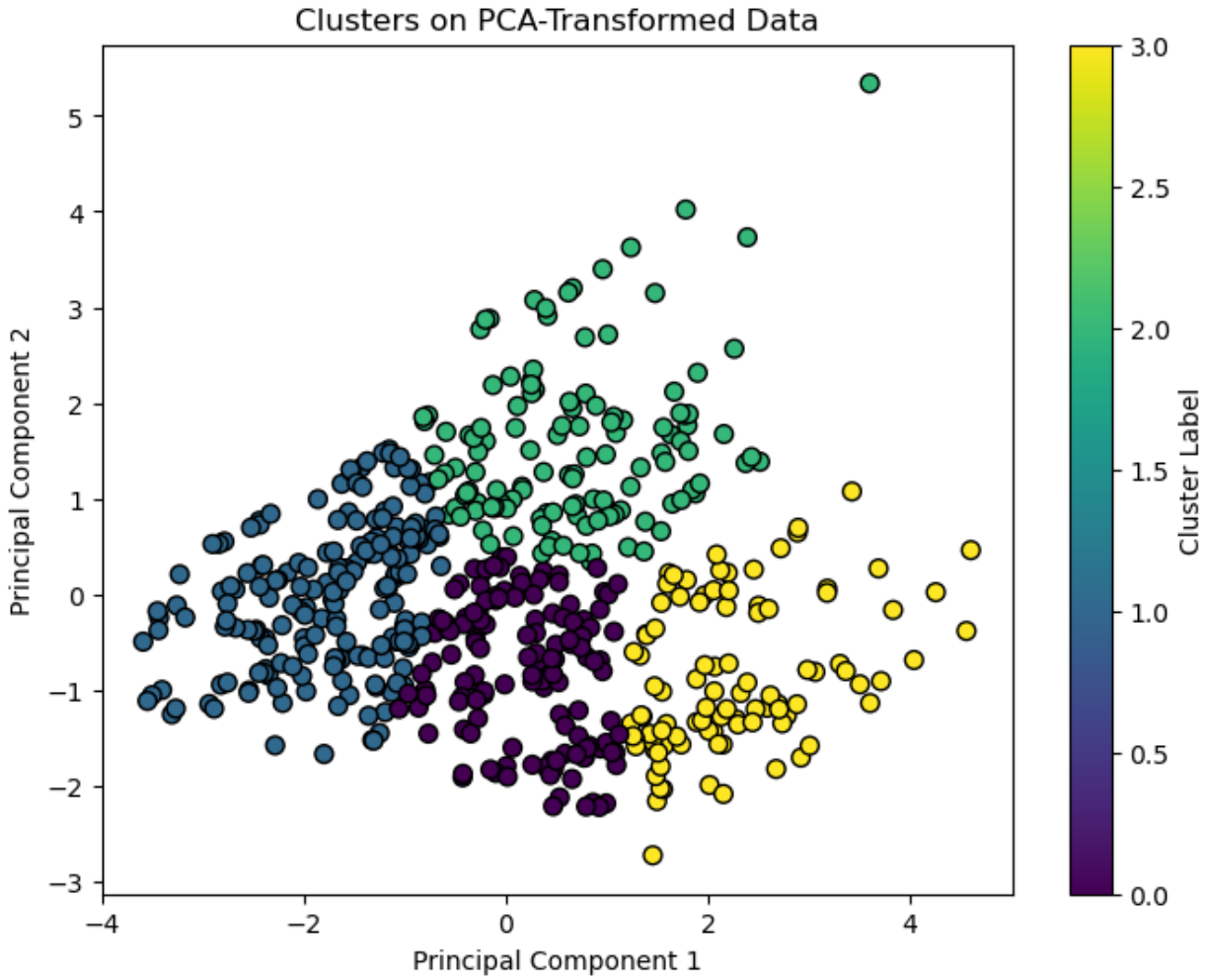
```
 53  ARC_N          1080 non-null   float64
 54  ERBB4_N        1080 non-null   float64
 55  nNOS_N         1080 non-null   float64
 56  Tau_N          1080 non-null   float64
 57  GFAP_N         1080 non-null   float64
 58  GluR3_N        1080 non-null   float64
 59  GluR4_N        1080 non-null   float64
 60  IL1B_N         1080 non-null   float64
 61  P3525_N        1080 non-null   float64
 62  pCASP9_N       1080 non-null   float64
 63  PSD95_N        1080 non-null   float64
 64  SNCA_N         1080 non-null   float64
 65  Ubiquitin_N    1080 non-null   float64
 66  pGSK3B_Tyr216_N  1080 non-null   float64
 67  SHH_N          1080 non-null   float64
 68  BAD_N          867 non-null    float64
 69  BCL2_N         795 non-null    float64
 70  pS6_N          1080 non-null   float64
 71  pCFOS_N        1005 non-null   float64
 72  SYP_N          1080 non-null   float64
 73  H3AcK18_N      900 non-null    float64
 74  EGR1_N         870 non-null    float64
 75  H3MeK4_N       810 non-null    float64
 76  CaNA_N         1080 non-null   float64
 77  Genotype       1080 non-null   object
 78  Treatment      1080 non-null   object
 79  Behavior       1080 non-null   object
dtypes: float64(77), object(3)
memory usage: 675.1+ KB
```

```
X = X.drop(['Genotype','Treatment','Behavior'],axis=1)
X.head()
X = X.dropna()
X.isna().sum()
Out[12]:
DYRK1A_N    0
ITSN1_N     0
BDNF_N      0
NR1_N       0
NR2A_N      0
        ..
SYP_N       0
H3AcK18_N   0
EGR1_N      0
H3MeK4_N    0
CaNA_N      0
Length: 77, dtype: int64
from sklearn.decomposition import PCA
from sklearn.preprocessing import OneHotEncoder
pca = PCA(n_components=2)
pca_result = pca.fit_transform(X)

plt.figure(figsize=(8, 6))
plt.scatter(pca_result[:, 0], pca_result[:, 1], c='blue', edgecolor='k', s=50)
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('PCA of Protein Expression Data')
plt.show()
```



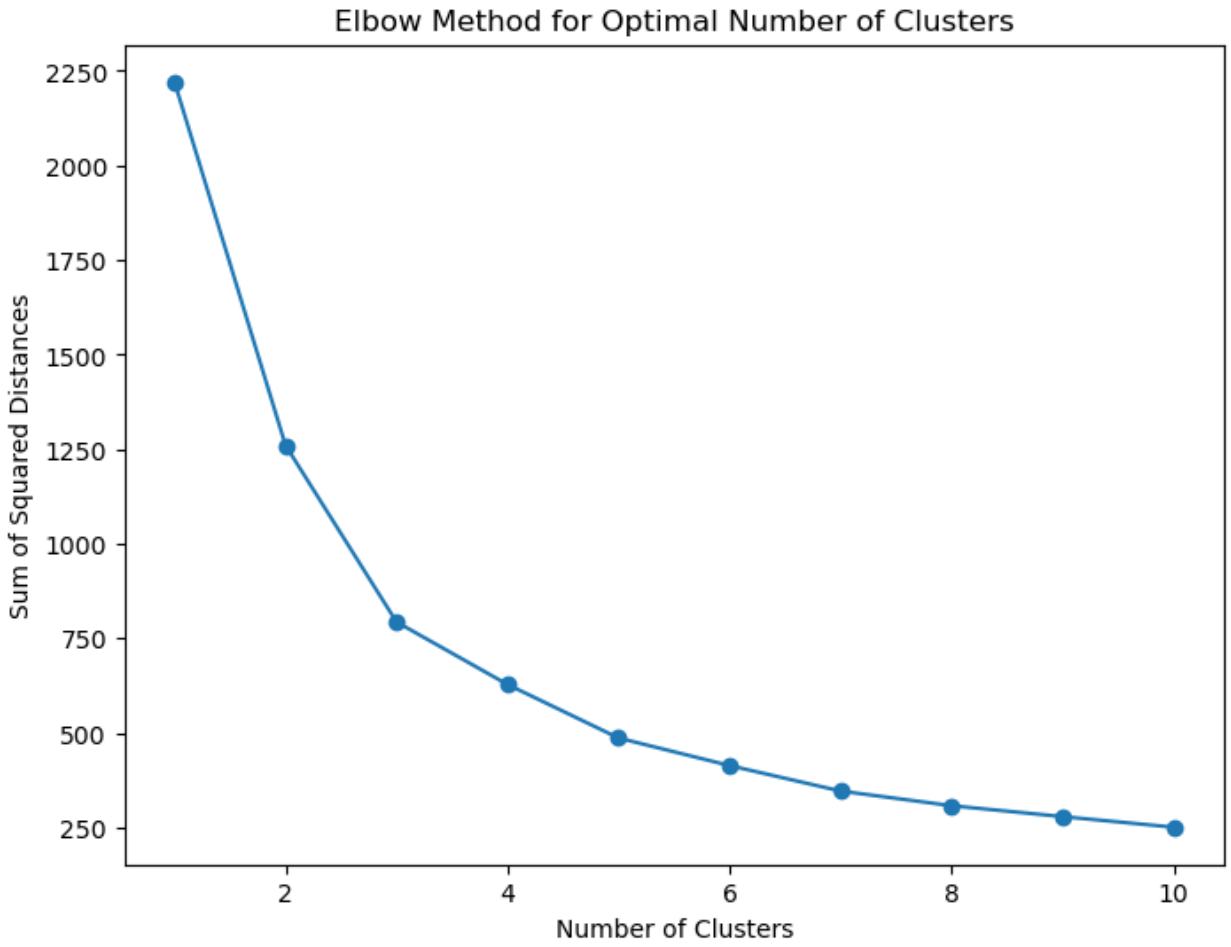

```
from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters=4)

kmeans.fit(pca_result)

cluster_labels = kmeans.labels_

plt.figure(figsize=(8, 6))

plt.scatter(pca_result[:, 0], pca_result[:, 1], c=cluster_labels, cmap='viridis', edgecolor='k', s=50)

plt.xlabel('Principal Component 1')

plt.ylabel('Principal Component 2')

plt.title('Clusters on PCA-Transformed Data')

plt.colorbar(label='Cluster Label')
```

```
plt.show()
```



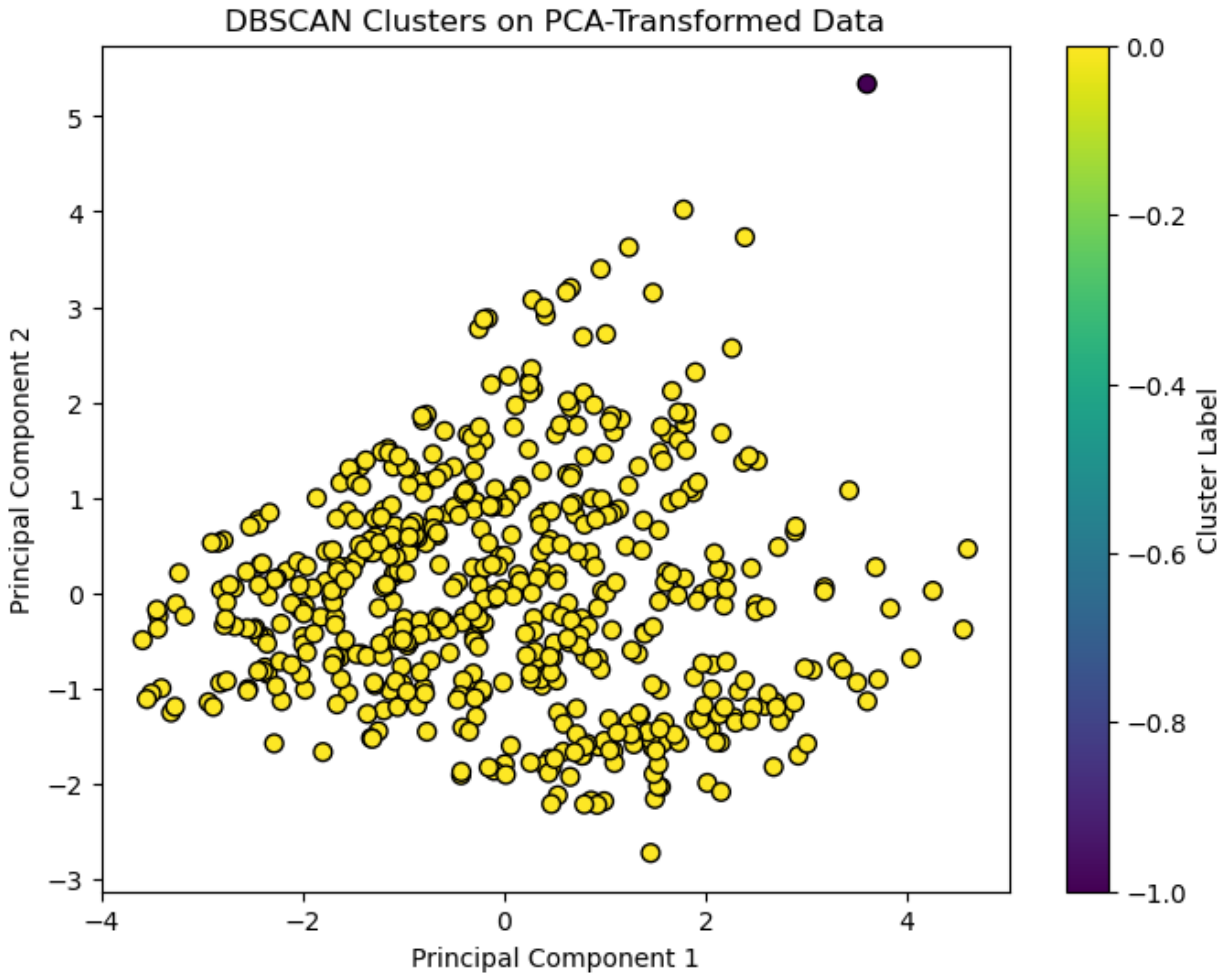

```
sse = []
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k)
    kmeans.fit(pca_result)
    sse.append(kmeans.inertia_)


plt.figure(figsize=(8, 6))
plt.plot(range(1, 11), sse, marker='o')
plt.xlabel('Number of Clusters')
plt.ylabel('Sum of Squared Distances')
plt.title('Elbow Method for Optimal Number of Clusters')
```

```
plt.show()
```




```
from sklearn.cluster import DBSCAN
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import silhouette_score
from sklearn.model_selection import KFold,ParameterGrid
from sklearn.metrics import adjusted_rand_score, normalized_mutual_info_score
param_grid = {
   'eps': np.arange(0.1, 1.0, 0.1),
   'min_samples': range(2, 10)
}
best_params = None
best_score = -1
```

```
for params in ParameterGrid(param_grid):
    dbscan = DBSCAN(eps=params['eps'], min_samples=params['min_samples'])
    labels = dbscan.fit_predict(pca_result)
    if len(set(labels)) > 1:
        score = silhouette_score(pca_result, labels)
        if score > best_score:
            best_score = score
            best_params = params
print(f'Best Parameters: {best_params}')
print(f'Best Silhouette Score: {best_score}')
dbscan = DBSCAN(eps=best_params['eps'], min_samples=best_params['min_samples'])
labels = dbscan.fit_predict(pca_result)
Best Parameters: {'eps': 0.7000000000000001, 'min_samples': 2}
Best Silhouette Score: 0.6000363590678445
plt.figure(figsize=(8, 6))
plt.scatter(pca_result[:, 0], pca_result[:, 1], c=labels, cmap='viridis', edgecolor='k', s=50)
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('DBSCAN Clusters on PCA-Transformed Data')
plt.colorbar(label='Cluster Label')
plt.show()
```




```
from sklearn.cluster import AgglomerativeClustering
from scipy.cluster.hierarchy import dendrogram, linkage

hierarchical = AgglomerativeClustering(n_clusters=3,linkage='ward')
hierarchical.fit(pca_result)
hierarchical_labels = hierarchical.labels_
from sklearn.metrics import silhouette_score, davies_bouldin_score
silhouette_avg = silhouette_score(pca_result, hierarchical_labels)
db_index = davies_bouldin_score(pca_result, hierarchical_labels)

print(f'Silhouette Score: {silhouette_avg}')
print(f'Davies-Bouldin Index: {db_index}')
```

Silhouette Score: 0.3728050389923807

Davies-Bouldin Index: 0.8671802038921039

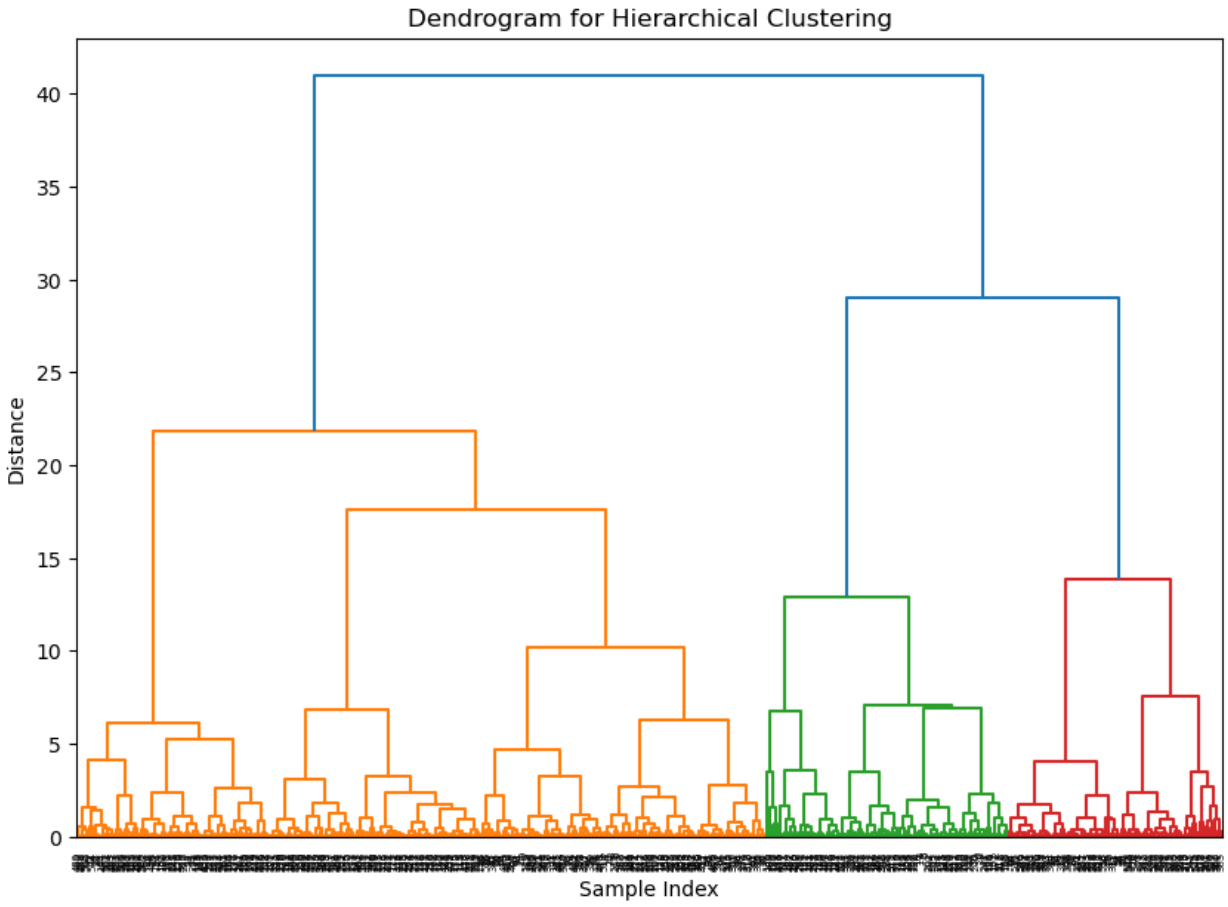
```
param_grid = {
   'n_clusters': [2, 3, 4, 5],
   'linkage': ['ward', 'complete', 'average', 'single']
}
best_params = None
best_score = -1
kf = KFold(n_splits=5, shuffle=True, random_state=42)
for params in ParameterGrid(param_grid):
   scores = []
   for train_index, val_index in kf.split(pca_result):
      train_data, val_data = pca_result[train_index], pca_result[val_index]
      hierarchical = AgglomerativeClustering(n_clusters=params['n_clusters'],
linkage=params['linkage'])
      hierarchical.fit(train_data)
      labels = hierarchical.fit_predict(val_data)
      if len(set(labels)) > 1:
         score = silhouette_score(val_data, labels)
         scores.append(score)
   avg_score = np.mean(scores)
   if avg_score > best_score:
      best_score = avg_score
      best_params = params
print(f'Best Parameters: {best_params}')
print(f'Best Silhouette Score: {best_score}')
```

Best Parameters: {'linkage': 'single', 'n_clusters': 2}

Best Silhouette Score: 0.4155256983433387

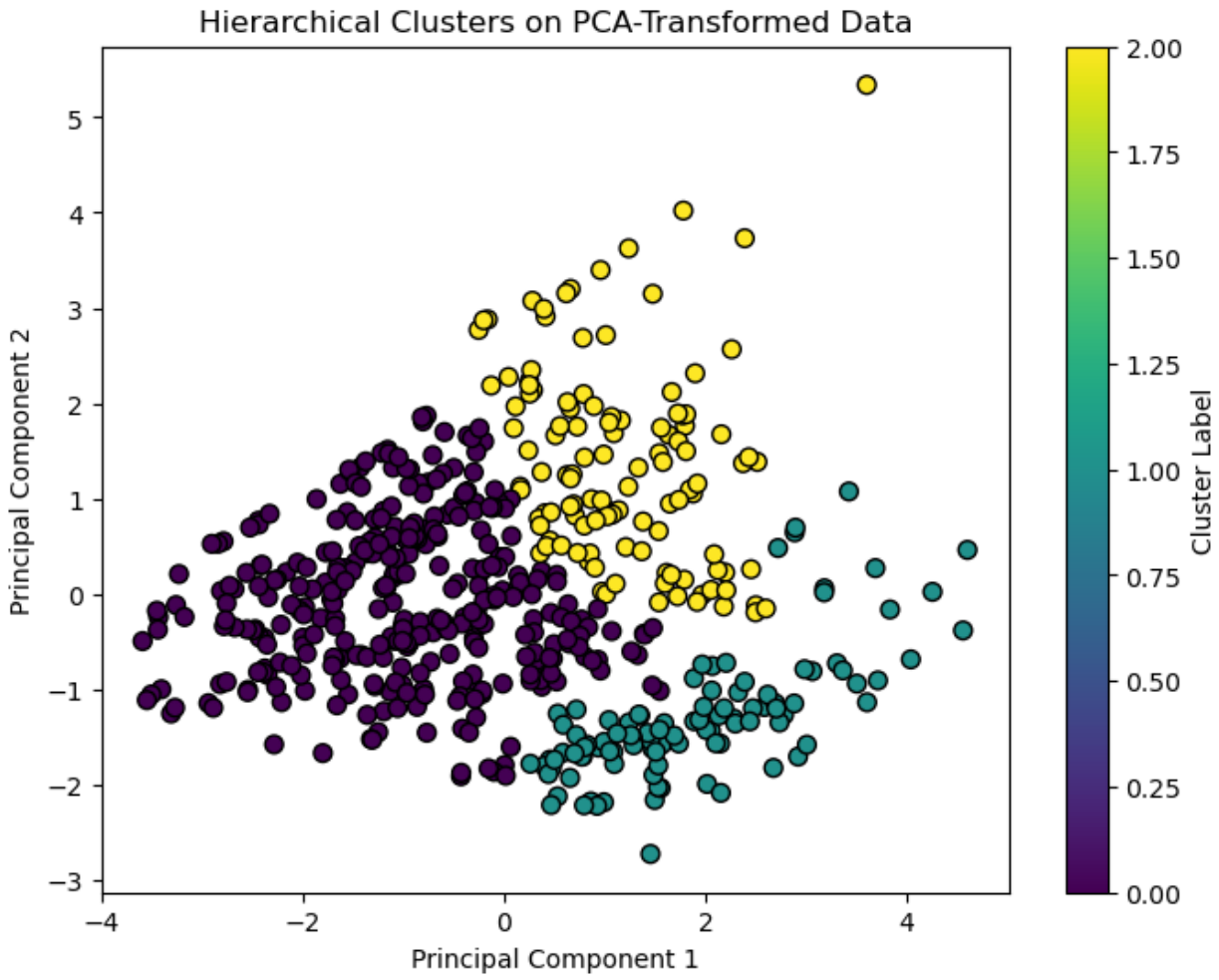
```
Z = linkage(pca_result, method='ward')
```

```
plt.figure(figsize=(10, 7))
dendrogram(Z)
plt.title('Dendrogram for Hierarchical Clustering')
plt.xlabel('Sample Index')
plt.ylabel('Distance')
plt.show()
```




```
plt.figure(figsize=(8, 6))
plt.scatter(pca_result[:, 0], pca_result[:, 1], c=hierarchical_labels, cmap='viridis', edgecolor='k', s=50)
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('Hierarchical Clusters on PCA-Transformed Data')
plt.colorbar(label='Cluster Label')
```

```
plt.show()
```




```
for cluster in range(3):
    cluster_data = X[hierarchical_labels == cluster]
    print(f'Cluster {cluster} Summary:')
    print(cluster_data.describe())
```

Cluster 0 Summary:

	DYRK1A_N	ITSN1_N	BDNF_N	...	EGR1_N	H3MeK4_N	CaNA_N
count	332.000000	332.000000	332.000000	...	332.000000	332.000000	332.000000
mean	0.391108	0.581345	0.287977	...	0.180608	0.197657	1.367432
std	0.158436	0.165190	0.038011	...	0.034452	0.044843	0.324345
min	0.145327	0.245359	0.115181	...	0.121468	0.101787	0.788462
25%	0.273292	0.456648	0.263937	...	0.156212	0.164728	1.108069

50% 0.356666 0.558752 0.290665 ... 0.173757 0.191075 1.357109

75% 0.452284 0.677435 0.311170 ... 0.200960 0.219010 1.612432

max 0.992220 1.151506 0.417221 ... 0.326143 0.333271 2.129791

[8 rows x 77 columns]

Cluster 1 Summary:

	DYRK1A_N	ITSN1_N	BDNF_N	...	EGR1_N	H3MeK4_N	CaNA_N
count	103.000000	103.000000	103.000000	...	103.000000	103.000000	103.000000
mean	0.421010	0.627181	0.334053	...	0.196199	0.217768	1.232963
std	0.164599	0.180628	0.045435	...	0.041969	0.044110	0.238048
min	0.194417	0.348331	0.232139	...	0.142407	0.146804	0.872258
25%	0.290289	0.502910	0.301343	...	0.168336	0.186181	1.035062
50%	0.365907	0.577851	0.324419	...	0.186439	0.210056	1.171313
75%	0.539343	0.715363	0.361587	...	0.214329	0.238219	1.455418
max	0.940956	1.085552	0.443358	...	0.360692	0.372005	1.696719

[8 rows x 77 columns]

Cluster 2 Summary:

	DYRK1A_N	ITSN1_N	BDNF_N	...	EGR1_N	H3MeK4_N	CaNA_N
count	117.000000	117.000000	117.000000	...	117.000000	117.000000	117.000000
mean	0.478623	0.738057	0.374891	...	0.156796	0.162574	1.537491
std	0.160074	0.183585	0.039257	...	0.019589	0.029317	0.330924
min	0.256346	0.477769	0.303897	...	0.120911	0.115270	0.795637
25%	0.334410	0.592269	0.344816	...	0.142573	0.141726	1.294631
50%	0.456786	0.719069	0.373247	...	0.155666	0.159211	1.562096
75%	0.573992	0.857961	0.397923	...	0.167615	0.179059	1.806884
max	0.945885	1.336398	0.497160	...	0.226396	0.284145	2.115555

[8 rows x 77 columns]

```
eps_values = [0.3, 0.5, 0.7]
min_samples_values = [3, 5, 7]

best_ari = -1
best_params = None

for eps in eps_values:
    for min_samples in min_samples_values:
        # Perform DBSCAN clustering
        dbscan = DBSCAN(eps=eps, min_samples=min_samples)
        dbscan_labels = dbscan.fit_predict(pca_result)

        # Convert labels to a DataFrame
        labels_df = pd.DataFrame({'Cluster': dbscan_labels})

        # Merge clustering results with target labels
        results = pd.concat([labels_df, y.reset_index(drop=True)], axis=1)

        # Handle noise points (label -1)
        results['Cluster'] = results['Cluster'].astype(str)

        # Evaluate clustering performance
        ari = adjusted_rand_score(y['class'], results['Cluster'])
        nmi = normalized_mutual_info_score(y['class'], results['Cluster'])
        silhouette_avg = silhouette_score(pca_result, dbscan_labels)

        print(f'eps: {eps}, min_samples: {min_samples}, ARI: {ari}, NMI: {nmi}, Silhouette Score: {silhouette_avg}')
```

```
    if ari > best_ari:
        best_ari = ari
        best_params = (eps, min_samples)

print(f'Best Parameters: eps={best_params[0]}, min_samples={best_params[1]}')
print(f'Best Adjusted Rand Index: {best_ari}')
```

eps: 0.3, min_samples: 3, ARI: 0.22479396609453056, NMI: 0.4371835273504064, Silhouette Score: -0.08980357358168613

eps: 0.3, min_samples: 5, ARI: 0.2190201810957853, NMI: 0.4207028158768853, Silhouette Score: -0.027080951387885178

eps: 0.3, min_samples: 7, ARI: 0.21832257727878246, NMI: 0.4138516040332791, Silhouette Score: 0.001829309894662775

eps: 0.5, min_samples: 3, ARI: 0.23384951320847377, NMI: 0.45367233055238393, Silhouette Score: 0.17924611069784274

eps: 0.5, min_samples: 5, ARI: 0.23336832624574125, NMI: 0.45027120460913, Silhouette Score: 0.23457737596562436

eps: 0.5, min_samples: 7, ARI: 0.23306053996224788, NMI: 0.45001363774247477, Silhouette Score: 0.23647298710010312

eps: 0.7, min_samples: 3, ARI: 0.23620823173264943, NMI: 0.4650033580452026, Silhouette Score: 0.6000363590678445

eps: 0.7, min_samples: 5, ARI: 0.235810802852644, NMI: 0.4649984220868739, Silhouette Score: 0.5027773495899465

eps: 0.7, min_samples: 7, ARI: 0.23582809114982572, NMI: 0.46330664825986206, Silhouette Score: 0.47893138355358683

Best Parameters: eps=0.7, min_samples=3

Best Adjusted Rand Index: 0.23620823173264943

```
dbscan = DBSCAN(eps=0.7, min_samples=2)
dbscan_labels = dbscan.fit_predict(pca_result)
```

In [36]:

```
plt.figure(figsize=(8, 6))
plt.scatter(pca_result[:, 0], pca_result[:, 1], c=dbscan_labels, cmap='viridis', edgecolor='k', s=50)
plt.xlabel('Principal Component 1')
```

```
plt.ylabel('Principal Component 2')
plt.title('DBSCAN Clusters on PCA-Transformed Data')
plt.colorbar(label='Cluster Label')
plt.show()
```