```
import pandas as pd
import numpy as np
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import plotly.express as px
import plotly.graph_objects as go
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
import scipy.cluster.hierarchy as sch
from sklearn.metrics import pairwise_distances
from sklearn.cluster import AgglomerativeClustering
```

```
data= pd.read_csv('cyber_crime_2020.csv')
data
```

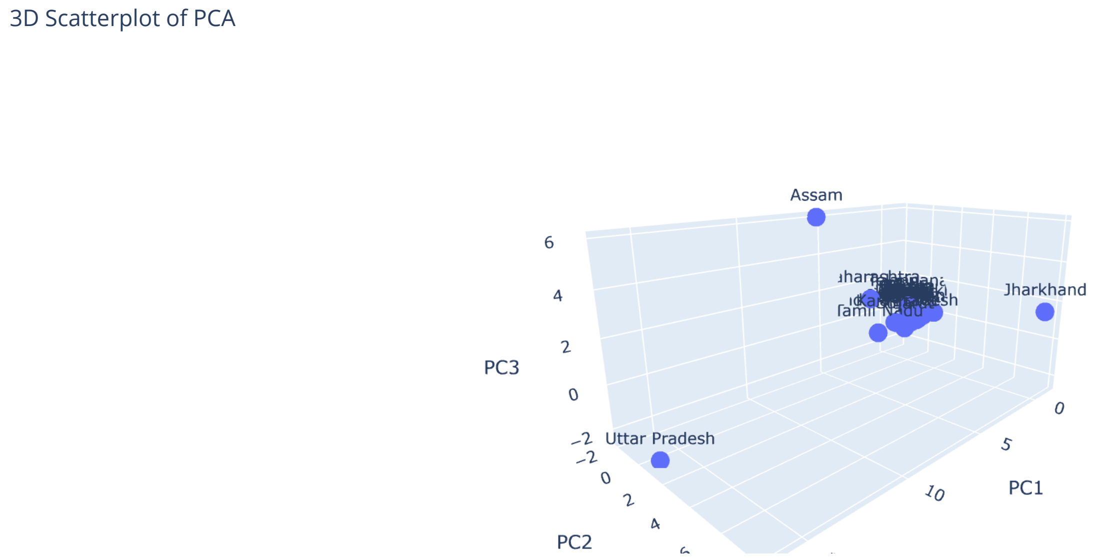
	State/UT	Personal Revenge	Anger	Fraud	Extortion	Causing Disrepute	Prank	Sexual Exploitation	Political Motives	Terrorist Activities	Inciting Hate against Country	Disrupt Public Service
0	Andhra Pradesh	83	39	1149	56	15	2	169	67	0	0	2
1	Arunachal Pradesh	1	0	26	0	0	0	0	0	0	0	0
2	Assam	654	164	242	447	85	35	483	24	0	58	12
3	Bihar	84	34	1218	102	19	12	32	7	0	0	1
4	Chhattisgarh	0	1	75	7	41	0	35	0	2	0	0
5	Goa	0	0	25	0	10	0	4	0	0	0	0
6	Gujarat	6	31	875	26	203	43	37	3	0	6	8
7	Haryana	14	3	157	17	9	1	70	1	0	0	2
8	Himachal Pradesh	2	1	19	9	15	0	34	3	0	0	1
9	Jharkhand	4	4	1069	14	2	0	13	0	7	0	7
10	Karnataka	147	13	9680	74	368	0	191	18	0	3	1
11	Kerala	44	34	96	21	58	10	138	10	0	0	0
12	Madhya Pradesh	7	6	292	13	66	2	119	3	0	0	0
13	Maharashtra	36	105	3413	45	76	32	612	9	0	3	2
14	Manipur	0	2	40	0	3	0	10	10	0	0	0
15	Meghalaya	6	10	81	7	9	0	9	1	1	0	0
16	Mizoram	0	0	3	0	2	3	1	1	3	0	0
17	Nagaland	0	0	5	0	1	1	0	0	0	0	0
18	Odisha	1	33	1380	175	0	0	239	0	0	0	0
19	Punjab	4	19	164	29	19	3	58	2	0	7	2
20	Rajasthan	22	10	641	42	73	11	67	4	0	4	0
21	Sikkim	0	0	0	0	0	0	0	0	0	0	0
22	Tamil Nadu	83	57	134	112	43	7	192	108	0	0	16
23	Telangana	96	24	4436	115	3	0	85	8	0	0	1
24	Tripura	14	1	11	0	2	0	3	1	0	0	0
25	Uttar Pradesh	78	210	4674	1055	547	87	560	73	96	82	35
26	Uttarakhand	11	5	98	33	6	0	44	1	0	0	0
27	West Bengal	66	8	72	12	3	3	44	1	0	0	0
28	A & N Islands	0	0	0	1	0	0	2	0	0	0	0
29	Chandigarh	0	0	7	1	0	0	7	0	0	0	0
30	D & N Haveli and Daman & Diu	0	0	0	0	0	0	3	0	0	0	0
31	Delhi	2	4	23	15	0	0	20	0	0	0	0
32	Jammu & Kashmir	3	4	33	9	28	2	12	1	4	2	1
33	Ladakh	0	0	0	0	0	0	0	0	0	0	1
34	Lakshadweep	2	0	0	0	0	0	0	0	0	0	0
35	Puducherry	0	0	4	3	0	0	0	0	0	0	0

Next steps: Generate code with data | View recommended plots | New interactive sheet

```
y = data['State/UT']
X = data.drop('State/UT', axis=1)


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

```
# Apply PCA
pca = PCA()
x_transformed = pca.fit_transform(X_scaled)
x_transformed_df = pd.DataFrame(x_transformed, columns=[f'PC{i+1}' for i in range(x_transformed.shape[1])])
x_transformed_df['State/UT'] = y
# 3D Scatter plot of first three principal components
fig = px.scatter_3d(x_transformed_df, x='PC1', y='PC2', z='PC3', text=y, title="3D Scatterplot of PCA",
                    labels={'PC1': 'PC1', 'PC2': 'PC2', 'PC3': 'PC3'},hover_data= {'State/UT':True})
fig.show()
```



from the plot clearly- Uttar Pradesh, Assam and Jharkhand are outliers

Generate a slider using jupyter widgets Close

```
explained_variance = pca.explained_variance_ratio_
explained_variance
```

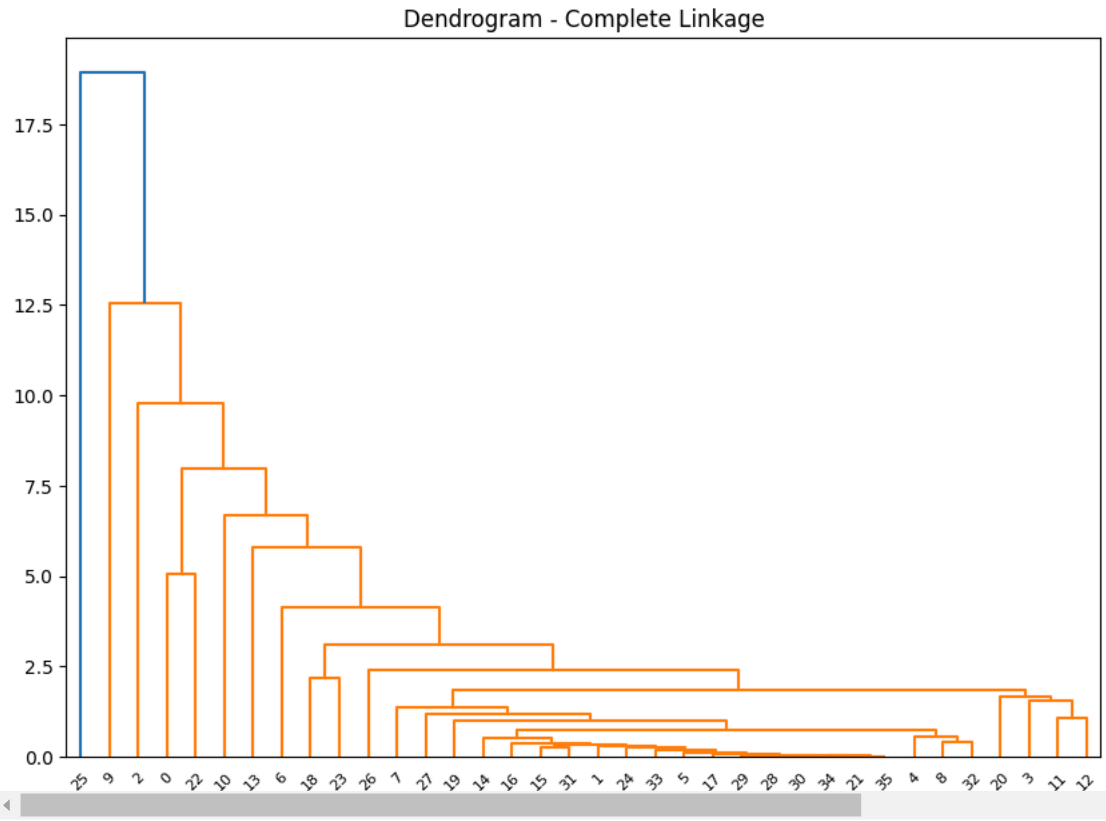
```
array([5.98067022e-01, 1.25306684e-01, 8.76169786e-02, 6.72793176e-02,
       5.21212991e-02, 2.63189147e-02, 1.89093034e-02, 1.24659164e-02,
       3.96578148e-03, 2.97843395e-03, 2.40069242e-03, 1.10879109e-03,
       6.89102087e-04, 4.80445815e-04, 2.13744345e-04, 7.75732983e-05])
```

```
plt.plot(range(1, len(explained_variance)+1), explained_variance, marker='o')
plt.title('Scree Plot')
plt.xlabel('Number of Principal Components')
plt.ylabel('Explained Variance')
plt.show()
```

Scree Plot

From the above Scree plot, elbow point seems to be at n=6, therefore number of principal components for efficient data dimensional reduction is 6

```
# Plot dendrogram for complete linkage
plt.figure(figsize=(10, 7))
plt.title("Dendrogram - Complete Linkage")
dendrogram = sch.dendrogram(sch.linkage(X_scaled, method='complete'))
plt.show()
```



Start coding or generate with AI.

```
# Apply hierarchical clustering with 4 clusters and complete linkage
agg_clustering = AgglomerativeClustering(n_clusters=4, linkage='complete')
y_agg = agg_clustering.fit_predict(X_scaled)

# Add cluster labels to the data
data['Agglomerative_Cluster'] = y_agg

# Get the states/UTs in each cluster
for i in range(4):
  cluster_states = data[data['Agglomerative_Cluster'] == i]['State/UT'].tolist()
  print(f"Cluster {i+1}: {cluster_states}")
```

```
Cluster 1: ['Andhra Pradesh', 'Arunachal Pradesh', 'Bihar', 'Chhattisgarh', 'Goa', 'Gujarat', 'Haryana', 'Himachal Pradesh', 'Karnat
Cluster 2: ['Uttar Pradesh']
Cluster 3: ['Jharkhand']
Cluster 4: ['Assam']
```

```
# Apply K-Means clustering with K=4
kmeans = KMeans(n_clusters=4, random_state=42)
y_kmeans = kmeans.fit_predict(X_scaled)

# Add cluster labels to the data
data['KMeans_Cluster'] = y_kmeans
```