```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
pip install umap-learn
```

```
Requirement already satisfied: umap-learn in /usr/local/lib/python3.10/dist-packages (0.5.6)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-packages (from umap-learn) (1.25.2)
Requirement already satisfied: scipy>=1.3.1 in /usr/local/lib/python3.10/dist-packages (from umap-learn) (1.11.4)
Requirement already satisfied: scikit-learn>=0.22 in /usr/local/lib/python3.10/dist-packages (from umap-learn) (1.2.2)
Requirement already satisfied: numba>=0.51.2 in /usr/local/lib/python3.10/dist-packages (from umap-learn) (0.58.1)
Requirement already satisfied: pynndescent>=0.5 in /usr/local/lib/python3.10/dist-packages (from umap-learn) (0.5.12)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from umap-learn) (4.66.2)
Requirement already satisfied: llvmlite<0.42,>=0.41.0dev0 in /usr/local/lib/python3.10/dist-packages (from numba>=0.51.2->umap-learn) (0
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.10/dist-packages (from pynndescent>=0.5->umap-learn) (1.4.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.22->umap-learn) (3.
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans, DBSCAN
from sklearn.decomposition import PCA
import plotly.graph_objs as go
import plotly.figure_factory as ff
import umap # use 'pip install umap-learn' or 'conda install -c conda-forge umap-learn'
```

```
# Importing dataset and examining it
dataset = pd.read_csv("/content/drive/MyDrive/Colab Notebooks/Data/tripadvisor_review.csv")
print(dataset.head())
print(dataset.shape)
print(dataset.info())
print(dataset.describe())
```

```
   User ID  Category 1  Category 2  Category 3  Category 4  Category 5  \
0  User 1        0.93         1.8        2.29        0.62        0.80
1  User 2        1.02         2.2        2.66        0.64        1.42
2  User 3        1.22         0.8        0.54        0.53        0.24
3  User 4        0.45         1.8        0.29        0.57        0.46
4  User 5        0.51         1.2        1.18        0.57        1.54

   Category 6  Category 7  Category 8  Category 9  Category 10
0        2.42        3.19        2.79        1.82         2.42
1        3.18        3.21        2.63        1.86         2.32
2        1.54        3.18        2.80        1.31         2.50
3        1.52        3.18        2.96        1.57         2.86
4        2.02        3.18        2.78        1.18         2.54
(980, 11)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 980 entries, 0 to 979
Data columns (total 11 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   User ID      980 non-null    object
 1   Category 1   980 non-null    float64
 2   Category 2   980 non-null    float64
 3   Category 3   980 non-null    float64
 4   Category 4   980 non-null    float64
 5   Category 5   980 non-null    float64
 6   Category 6   980 non-null    float64
 7   Category 7   980 non-null    float64
 8   Category 8   980 non-null    float64
 9   Category 9   980 non-null    float64
 10  Category 10  980 non-null    float64
dtypes: float64(10), object(1)
memory usage: 84.3+ KB
None
       Category 1  Category 2  Category 3  Category 4  Category 5  Category 6  \
count  980.000000  980.000000  980.000000  980.000000  980.000000  980.000000
mean     0.893194    1.352612    1.013306    0.532500    0.939735    1.842898
std      0.326912    0.478280    0.788607    0.279731    0.437430    0.539538
min      0.340000    0.000000    0.130000    0.150000    0.060000    0.140000
25%      0.670000    1.080000    0.270000    0.410000    0.640000    1.460000
50%      0.830000    1.280000    0.820000    0.500000    0.900000    1.800000
75%      1.020000    1.560000    1.572500    0.580000    1.200000    2.200000
max      3.220000    3.640000    3.620000    3.440000    3.300000    3.760000
```

```
       Category 7  Category 8  Category 9  Category 10
count  980.000000  980.000000  980.000000   980.000000
mean     3.180939    2.835061    1.569439     2.799224
std      0.007824    0.137505    0.364629     0.321380
min      3.160000    2.420000    0.740000     2.140000
25%      3.180000    2.740000    1.310000     2.540000
50%      3.180000    2.820000    1.540000     2.780000
75%      3.180000    2.910000    1.760000     3.040000
max      3.210000    3.390000    3.170000     3.660000
```

```
# Defining feature set
X = dataset.drop(['User ID'], axis = 1) # Features
print(type(X))
print(X.shape)
```

```
<class 'pandas.core.frame.DataFrame'>
(980, 10)
```

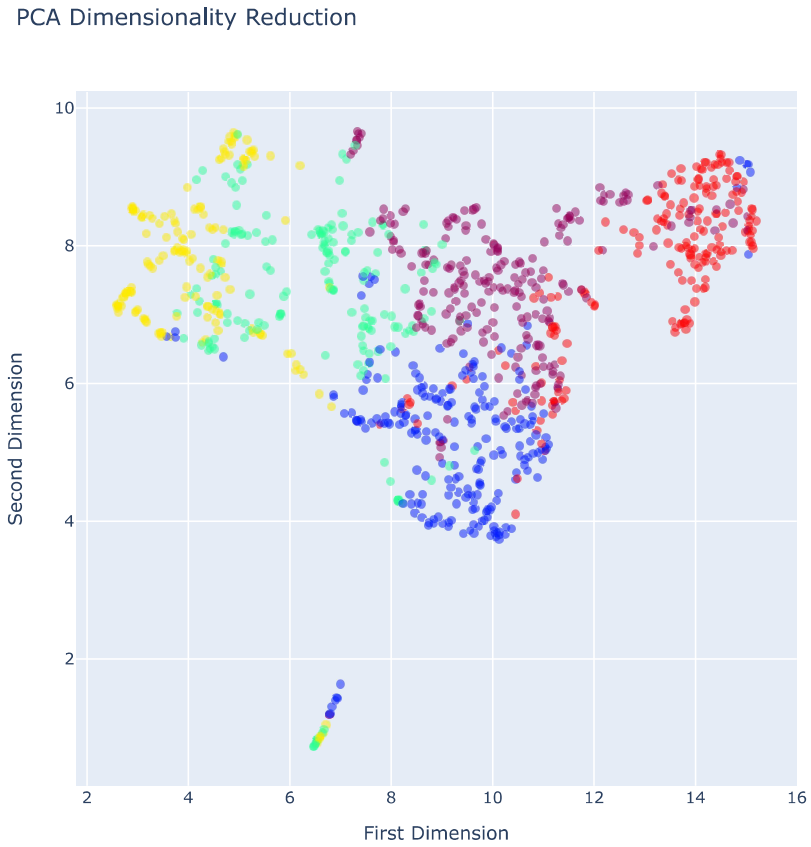
```
# Normalizing numerical features so that each feature has mean 0 and variance 1
feature_scaler = StandardScaler()
X_scaled = feature_scaler.fit_transform(X)
```

```
# Implementing PCA to visualize dataset
pca = PCA(n_components = 2)
pca.fit(X_scaled)
x_pca = pca.transform(X_scaled)
print("Variance explained by each of the n_components: ",pca.explained_variance_ratio_)
print("Total variance explained by the n_components: ",sum(pca.explained_variance_ratio_))

products=list(dataset['User ID'])

data = [go.Scatter(x=x_pca[:,0], y=x_pca[:,1], mode='markers',
                   marker = dict(color=None, colorscale='Rainbow', opacity=0.5),
                               text=[f'Product: {a}' for a in products],
                               hoverinfo='text')]

layout = go.Layout(title = 'PCA Dimensionality Reduction', width = 700, height = 700,
                   xaxis = dict(title='First Principal Component'),
                   yaxis = dict(title='Second Principal Component'))
fig = go.Figure(data=data, layout=layout)
fig.show()
```

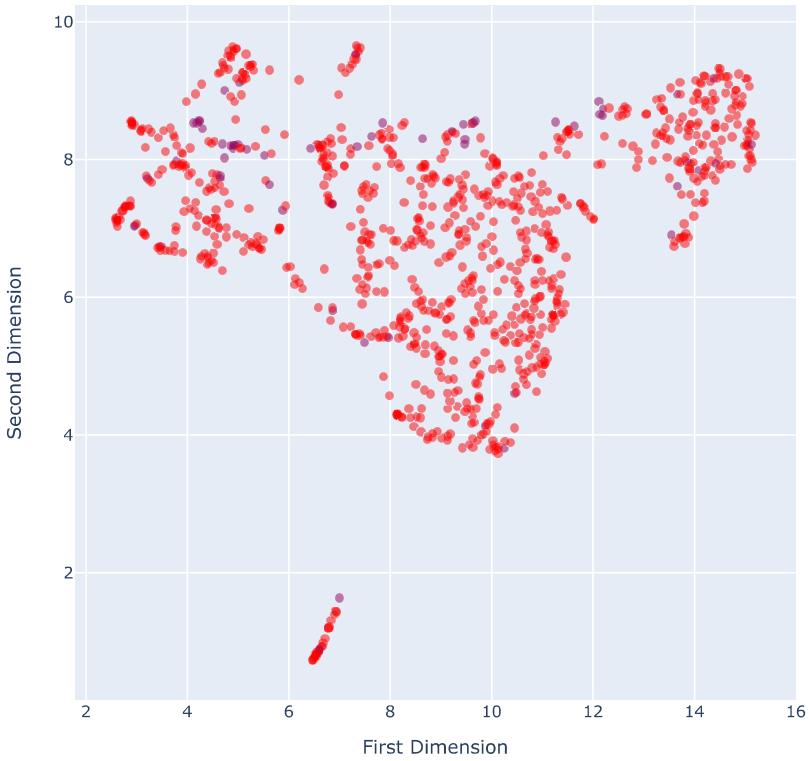
```
Variance explained by each of the n_components:  [0.29775044 0.1262794 ]
Total variance explained by the n_components:  0.42402983746016903
```


```
# Labelling clusters using KMeans for PCA
kmeans = KMeans(n_clusters = 5)
kmeans.fit(x_pca)

labels = list(kmeans.labels_)
data = [go.Scatter(x=x_umap[:,0], y=x_umap[:,1], mode='markers',
                   marker = dict(color=kmeans.labels_, colorscale='Rainbow', opacity=0.5),
                               text=[f'Product: {a}<br>Label: {b}' for a,b in list(zip(products,labels))],
                               hoverinfo='text')]

layout = go.Layout(title = 'PCA Dimensionality Reduction', width = 700, height = 700,
                   xaxis = dict(title='First Dimension'),
                   yaxis = dict(title='Second Dimension'))
fig = go.Figure(data=data, layout=layout)
fig.show()
```

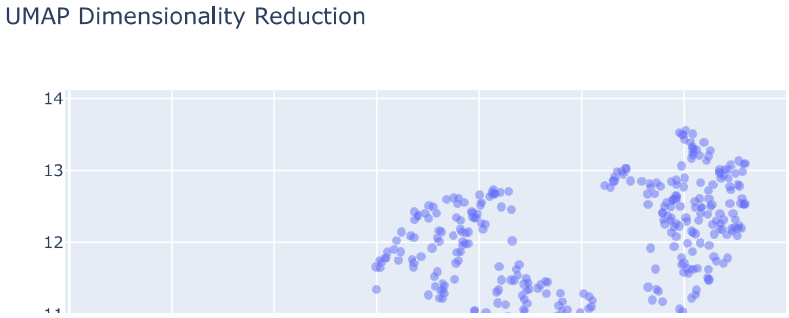
```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
```




```
# Implementing DBSCAN for PCA
dbscan = DBSCAN(eps=0.5, min_samples=12)
dbscan.fit(x_pca)
labels = list(dbscan.labels_)
data = [go.Scatter(x=x_umap[:,0], y=x_umap[:,1], mode='markers',
                   marker = dict(color=dbscan.labels_, colorscale='Rainbow', opacity=0.5),
                                 text=[f'Product: {a}<br>Label: {b}' for a,b in list(zip(products,labels))],
                                 hoverinfo='text')]

layout = go.Layout(title = 'PCA Dimensionality Reduction', width = 700, height = 700,
                   xaxis = dict(title='First Dimension'),
                   yaxis = dict(title='Second Dimension'))
fig = go.Figure(data=data, layout=layout)
fig.show()
```




```
# Implementing UMAP to visualize dataset
u = umap.UMAP(n_components = 2, n_neighbors=15, min_dist=0.1)
x_umap = u.fit_transform(X_scaled)

data = [go.Scatter(x=x_umap[:,0], y=x_umap[:,1], mode='markers',
                   marker = dict(color=None, colorscale='Rainbow', opacity=0.5),
                               text=[f'Product: {a}' for a in products],
                               hoverinfo='text')]

layout = go.Layout(title = 'UMAP Dimensionality Reduction', width = 700, height = 700,
                   xaxis = dict(title='First Dimension'),
                   yaxis = dict(title='Second Dimension'))
fig = go.Figure(data=data, layout=layout)
fig.show()
```




```
# Labelling clusters using KMeans for UMAP
kmeans = KMeans(n_clusters = 5)
kmeans.fit(x_umap)

labels = list(kmeans.labels_)
data = [go.Scatter(x=x_umap[:,0], y=x_umap[:,1], mode='markers',
                   marker = dict(color=kmeans.labels_, colorscale='Rainbow', opacity=0.5),
                               text=[f'Product: {a}<br>Label: {b}' for a,b in list(zip(products,labels))],
                               hoverinfo='text')]

layout = go.Layout(title = 'UMAP Dimensionality Reduction', width = 700, height = 700,
                   xaxis = dict(title='First Dimension'),
                   yaxis = dict(title='Second Dimension'))
fig = go.Figure(data=data, layout=layout)
fig.show()
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
```