CONSIDER INCORPORATING DIMENSIONALITY REDUCTION TECHNIQUES LIKE PCA OR T-SNE TO VISUALIZE HIGH-DIMENSIONAL CUSTOMER DATA AND DISCOVER UNDERLYING PATTERNS.

What is t-SNE ?

t-SNE (t-distributed Stochastic Neighbor Embedding) is an unsupervised non-linear dimensionality reduction technique for data exploration and visualizing high-dimensional data. Non-linear dimensionality reduction means that the algorithm allows us to separate data that cannot be separated by a straight line.

t-SNE gives you a feel and intuition on how data is arranged in higher dimensions. It is often used to visualize complex datasets into two and three dimensions, allowing us to understand more about underlying patterns and relationships in the data.

Take our **Dimensionality Reduction in Python** course to learn about exploring high-dimensional data, feature selection, and feature extraction.

How t-SNE works

The t-SNE algorithm finds the similarity measure between pairs of instances in higher and lower dimensional space. After that, it tries to optimize two similarity measures. It does all of that in three steps.

1. t-SNE models a point being selected as a neighbor of another point in both higher and lower dimensions. It starts by calculating a pairwise similarity between all data points in the high-dimensional space using a Gaussian kernel. The points that are far apart have a lower probability of being picked than the points that are close together.
2. Then, the algorithm tries to map higher dimensional data points onto lower dimensional space while preserving the pairwise similarities.
3. It is achieved by minimizing the divergence between the probability distribution of the original high-dimensional and lower-dimensional. The algorithm uses gradient descent to minimize the divergence. The lower-dimensional embedding is optimized to a stable state.

t-SNE Python Example

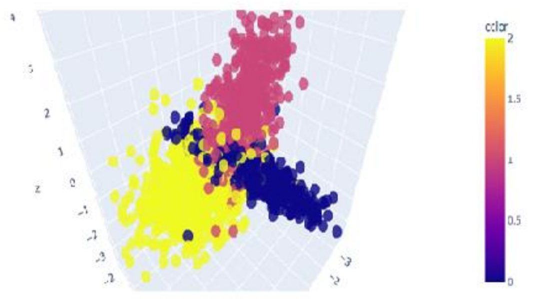
```
In[1]:import plotly.express as px


      from sklearn.datasets import make_classification

      X, y=make_classification(n_features=6,n_classes=3,n_samples=1500,n_informative=2,
      random_state=5,n_clusters_per_class=1,)fig = px.scatter_3d(x=X[:, 0], y=X[:, 1],
  z=X[:, 2], color=y, opacity=0.8)

      fig.show()
```

Out[1]:



Fitting and Transforming t-SNE

```
In[2]:from sklearn.manifold import TSNE

        tsne = TSNE(n_components=2, random_state=42)

        X_tsne = tsne.fit_transform(X)

        tsne.kl_divergence_

Out[2]:1.1169137954711914
```

t-SNE Visualization Python

Similar to PCA, we will visualize two t-SNE components on a scatter plot.

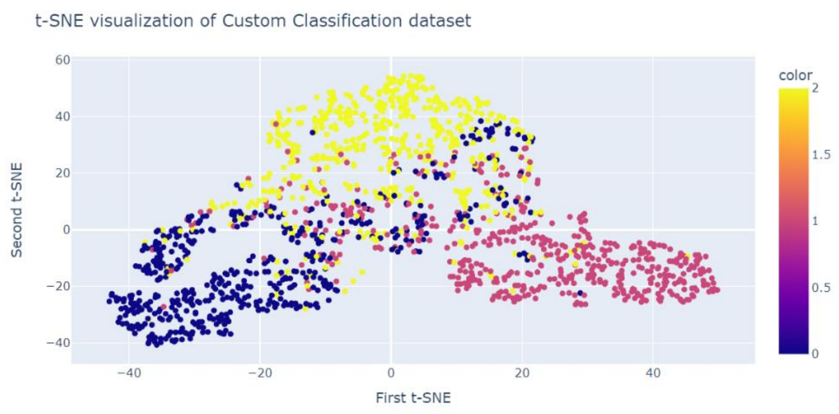
```
In[3]:  fig = px.scatter(x=X_tsne[:, 0], y=X_tsne[:, 1], color=y)

        fig.update_layout(title="t-SNE visualization of Custom Classification dataset",

xaxis_title="First t-SNE",yaxis_title="Second t-SNE")

        fig.show()
```

Out[3]:



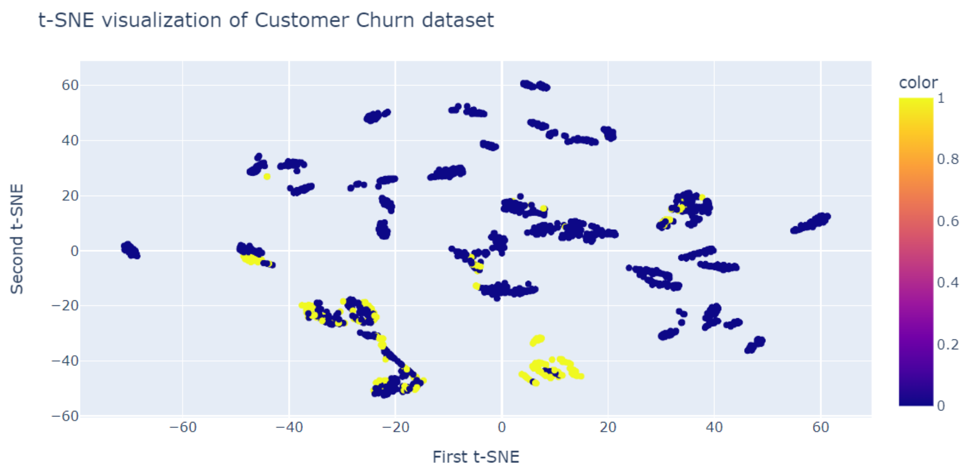
Visualizing t-SNE

```
In[4]:fig = px.scatter(x=X_train_tsne[:, 0], y=X_train_tsne[:, 1], color=y_train)

            fig.update_layout(title="t-SNE visualization of Customer Churn dataset",

xaxis_title="First t-SNE",yaxis_title="Second t-SNE",)

            fig.show()
```



Importing Mall_CustomerDataset:

```
In[6]:import pandas as pd
        df = pd.read_csv("Mall_Customer.csv")
        df.head(3)
```

Out[6]:

	CustomerID	Genre	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6