# 1. OBJECTIVE OF THE PROGRAM

The objective of this code is to perform dimensionality reduction using t-SNE (t-Distributed Stochastic Neighbor Embedding) on the Iris dataset and visualize the dataset in 2D space.
The dataset originally contains 4 features per sample:

1. Sepal length
2. Sepal width
3. Petal length
4. Petal width

Since humans cannot directly visualize 4D data, t-SNE maps the dataset into 2 dimensions, preserving similarity between points as much as possible.
The code also evaluates the embedding quality using:

1. Trustworthiness score
2. Silhouette score

# 2. KEY TERMS

**2.1 Dimensionality Reduction**
Definition: Process of reducing the number of features (dimensions) while preserving important structure.
Goal:
Convert high-dimensional dataset $X \in \mathbb{R}^d$ into low-dimensional dataset $Y \in \mathbb{R}^k$, where $k < d$.

**2.2 t-SNE (t-Distributed Stochastic Neighbor Embedding)**
Definition: A nonlinear dimensionality reduction technique mainly used for visualization.
It converts distances between points into probabilities, then tries to preserve those probabilities in low-dimensional space.

t-SNE converts distances into probabilities.
Similarity [High-dimensional] probability: For two points $x_i$and $x_j$, t-SNE defines conditional probability:

$$p_{(j|i)} = \frac{exp\left(-\frac{|x_i - x_j|^2}{2\sigma_i^2}\right)}{\sum_{k \neq i} exp\left(-\frac{|x_i - x_k|^2}{(2\sigma_i^2)}\right)}$$

Where:

- $|| x_i - x_j ||^2$= squared Euclidean distance
- $\sigma_i$= variance controlling neighborhood size

Then symmetric joint probability:

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$$

Similarity [Low-dimensional] probability: In low dimension, similarity is computed using Student t-distribution

$$q_{ij} = \frac{\left(1 + |y_i - y_j|^2\right)^{-1}}{\sum_{k \neq l}(1 + |y_k - y_l|^2)^{-1}}$$

This heavy-tailed distribution reduces the crowding problem.

t-SNE minimizes KL divergence (Loss Function): t-SNE minimizes the difference between $P$and $Q$using Kullback-Leibler divergence

$$KL(P||Q) = \sum_i \sum_j p_{ij} \, log\left(\frac{p_{ij}}{q_{ij}}\right)$$

This ensures that points close in original space remain close in embedding.

**2.3 Perplexity (t-SNE Parameter)**
Definition: Perplexity controls how many neighbours t-SNE considers important.
Controls the effective number of neighbors.

$$Perplexity(P_i) = 2^{H(P_i)}$$

Small perplexity is appropriate because the corpus vocabulary is small.
Where entropy is:

$$H(P_i) = -\sum_j p_{j|i} \log_2(p_{j|i})$$

- Higher perplexity → considers more global structure.
- Lower perplexity → focuses on local neighborhoods.

**2.4 StandardScaler (Standardization)**
Definition: Rescales features so that each feature has mean 0 and variance 1.
Formula:

$$z = \frac{x - \mu}{\sigma}$$

Where:

- $\mu$= mean
- $\sigma$= standard deviation
- This is required because t-SNE is sensitive to feature scaling.

**2.5 Trustworthiness Score**
Definition: Measures how well the low-dimensional embedding preserves local neighbors.
Range: 0 to 1 [Higher is better.]
Formula:

$$T(k) = 1 - \frac{2}{nk(2n - 3k - 1)} \sum_{i=1}^{n} \sum_{j \in U_k(i)} (r(i,j) - k)$$

Where:

- ✦ $U_k(i)$are points that appear in k-nearest neighbors in embedding but not in original space
- ✦ $r(i,j)$is rank in original space

**2.6 Silhouette Score**

Definition: Measures clustering separation.

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$$

Where:

- ✦ $a(i)$= average distance to points in same cluster
- ✦ $b(i)$= average distance to nearest different cluster
- ✦ Range: **-1 to 1 [**Higher is better.]

# 3. CODE EXPLANATION

**3.1 Import Libraries**

```
import numpy as np
import matplotlib.pyplot as plt
```

`numpy` is imported for numerical operations.
`matplotlib.pyplot` is imported for visualization.

```
from sklearn.datasets import load_iris
from sklearn.preprocessing import StandardScaler
from sklearn.manifold import TSNE, trustworthiness
from sklearn.metrics import silhouette_score
```

`load_iris` loads the Iris dataset.
`StandardScaler` standardizes features.
`TSNE` runs the t-SNE algorithm.
`trustworthiness` evaluates neighborhood preservation.
`silhouette_score` evaluates clustering quality.

**3.2 Load Dataset**

```
X, y = load_iris(return_X_y=True)
```

`X` stores features (150 samples × 4 features).
`y` stores class labels (0, 1, 2).
This is a supervised dataset, but t-SNE itself is unsupervised.

**3.3 Standardize Data**

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

`scaler = StandardScaler()` initializes scaling object.
`fit_transform()` computes mean/std and transforms values.
*Purpose:* Ensures all features contribute equally to distance computations.

**3.5 Initialize t-SNE**

```
tsne = TSNE(
    n_components=2,
```

```
        perplexity=30,
        learning_rate=200,
        n_iter=1000,
        metric="mahalanobis",
        random_state=42
    )
```

`n_components=2`: output will be 2D embedding.
`perplexity=30`: considers ~30 nearest neighbors.
`learning_rate=200`: step size for gradient descent updates.
`n_iter=1000`: number of optimization iterations.
`metric="mahalanobis"`: uses Mahalanobis distance instead of Euclidean.
`random_state=42`: ensures reproducible results.

**3.6 Mahalanobis distance formula:**

$$d(x, y) = \sqrt{(x - y)^T S^{-1} (x - y)}$$

Where $S$is covariance matrix.

**3.7 Fit and Transform Dataset**

```
X_tsne = tsne.fit_transform(X_scaled)
```

Fits t-SNE model on standardized data.
Returns 2D embedding of shape (150 × 2).
This is the final reduced representation.

**3.8 Compute Trustworthiness Score**

```
trust = trustworthiness(X_scaled, X_tsne, n_neighbors=5)
print(f"Trustworthiness score: {trust:.4f}")
```

Measures whether nearest neighbors in original space remain neighbors in embedded space.
`n_neighbors=5` means check neighborhood consistency among top 5 nearest points.

**3.9 Compute Silhouette Score**

```
sil_score = silhouette_score(X_tsne, y)
print(f"Silhouette score: {sil_score:.4f}")
```

Measures how well the 3 iris classes separate in embedding.
Uses `y` labels to check if points of same class are grouped.

**3.10 Visualization**

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

Creates a plotting canvas of size 8×6.

```
scatter = plt.scatter(
    X_tsne[:, 0],
    X_tsne[:, 1],
    c=y,
    cmap="viridis",
    s=60,
    alpha=0.8
)
```

Plots t-SNE output points.
`X_tsne[:,0]` is x-axis coordinate.

`X_tsne[:,1]` is y-axis coordinate.
`c=y` assigns color by class label.
`cmap="viridis"` sets colormap.
`s=60` controls point size.
`alpha=0.8` makes points slightly transparent.

**3.11 Add Colorbar and Labels**

```
plt.colorbar(scatter, label="Class Label")
```

Adds legend bar mapping colors to class IDs (0,1,2).

```
plt.title("t-SNE Visualization (Iris Dataset)")
plt.xlabel("t-SNE Dimension 1")
plt.ylabel("t-SNE Dimension 2")
plt.grid(True)
```

Adds title and axis labels.
`grid(True)` improves readability.

**3.12 Display Output**

```
plt.show()
```

Renders the scatter plot.

**3.13 For 2nd and 3rd part**

```
import pandas as pd
url = "https://raw.githubusercontent.com/pandas-
dev/pandas/main/pandas/tests/io/data/csv/tips.csv"
# Corrected URL to an existing raw CSV file
data = pd.read_csv(url)
print(data.head())
data.info()
data.describe()

import pandas as pd
url = "https://raw.githubusercontent.com/pandas-
dev/pandas/main/pandas/tests/io/data/csv/iris.csv"
data = pd.read_csv(url)
print(data.head())
data.info()
data.describe()
```

```
.head()
```

Shows sample rows so you can confirm:
dataset is correctly loaded
column names are correct
data types appear meaningful

```
.info()
```

Gives:

- number of rows and columns
- datatypes

- ✦ memory usage
- ✦ missing values status

This helps decide preprocessing steps like:

- ✦ encoding categorical features
- ✦ scaling numeric features
- ✦ handling missing values

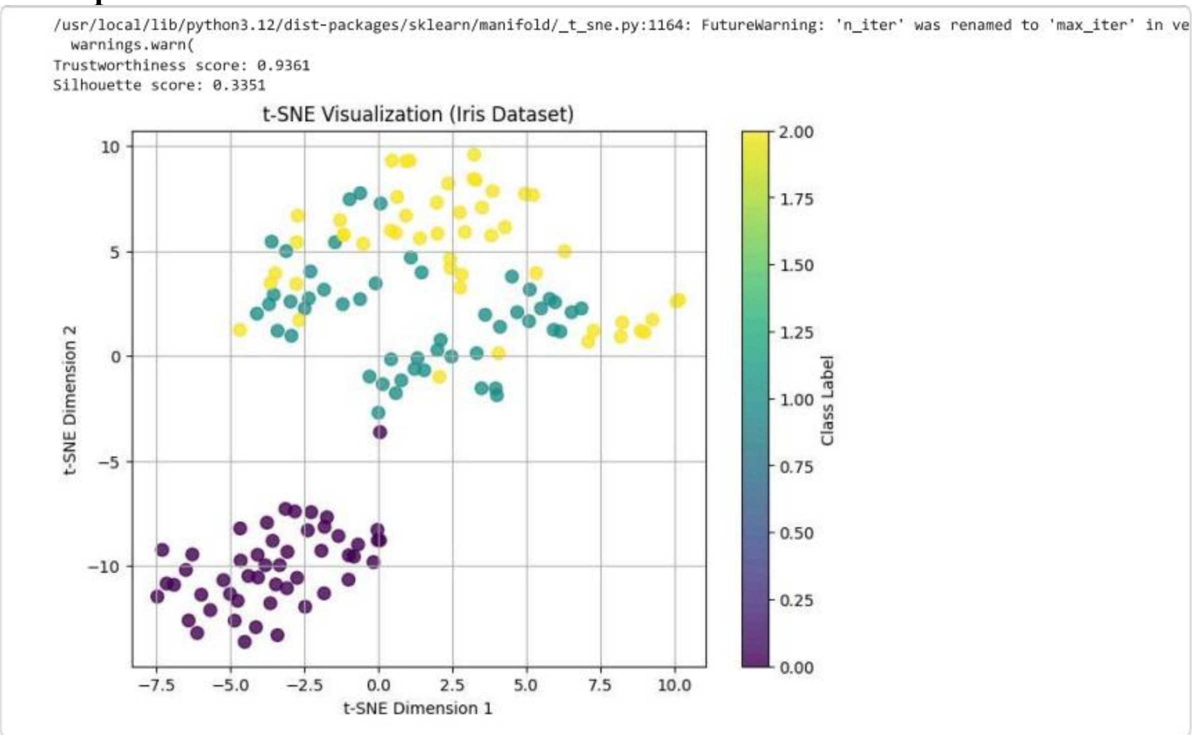
```
.describe()
```

Gives statistical distribution summary.
Used for:

- ✦ detecting outliers
- ✦ understanding data spread
- ✦ checking feature ranges

# 4. Output Interpretation

### 4.1 Output




The output includes:
**Trustworthiness score: 0.9361**
**Silhouette score: 0.3351**
Scatter plot titled: *t-SNE Visualization (Iris Dataset)*

**Trustworthiness Score Interpretation (0.9361)**

- ✦ This value is close to 1.
- ✦ It indicates that local neighborhood structure is strongly preserved.
- ✦ Meaning: nearest neighbors in original 4D space remain neighbors in 2D.

This is considered a strong embedding.

**Silhouette Score Interpretation (0.3351)**

- Value is moderately positive.
- This means clusters exist but are not perfectly separated.
- For Iris dataset, one class (Setosa) is usually well separated, while the other two overlap.

So, this score is logically consistent.

**Scatter Plot Explanation**

- The purple cluster (label 0) forms a clear separated group at the bottom-left region. This corresponds to Iris Setosa, which is naturally separable.
- The green and yellow points (labels 1 and 2) overlap heavily in the upper region. These represent Iris Versicolor and Iris Virginica, which share similar feature patterns.
- The axes represent t-SNE coordinates, not original features.
- Their numeric scale has no direct physical meaning.
- The main interpretation is based on relative closeness, not absolute axis values.

t-SNE pipeline:
preprocessing → reduction → evaluation → visualization.

**4.2 Output**

```
   total_bill   tip     sex smoker  day    time  size
0       16.99  1.01  Female     No  Sun  Dinner     2
1       10.34  1.66    Male     No  Sun  Dinner     3
2       21.01  3.50    Male     No  Sun  Dinner     3
3       23.68  3.31    Male     No  Sun  Dinner     2
4       24.59  3.61  Female     No  Sun  Dinner     4
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244 entries, 0 to 243
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   total_bill  244 non-null    float64
 1   tip         244 non-null    float64
 2   sex         244 non-null    object 
 3   smoker      244 non-null    object 
 4   day         244 non-null    object 
 5   time        244 non-null    object 
 6   size        244 non-null    int64  
dtypes: float64(2), int64(1), object(4)
memory usage: 13.5+ KB
```

| | total_bill | tip | size |
|---|---|---|---|
| count | 244.000000 | 244.000000 | 244.000000 |
| mean | 19.785943 | 2.998279 | 2.569672 |
| std | 8.902412 | 1.383638 | 0.951100 |
| min | 3.070000 | 1.000000 | 1.000000 |
| 25% | 13.347500 | 2.000000 | 2.000000 |
| 50% | 17.795000 | 2.900000 | 2.000000 |
| 75% | 24.127500 | 3.562500 | 3.000000 |
| max | 50.810000 | 10.000000 | 6.000000 |

*This code does not create graphs, but it produces tabular outputs*

total_bill

count = 244
mean ≈ 19.7859
std ≈ 8.9024
min = 3.07
max = 50.81
Meaning: Bills range from 3.07 to 50.81.

tip

mean ≈ 2.9983
max = 10.00

Meaning: Average tip is ~3.

size

mean ≈ 2.5697
max = 6
Meaning: Average group size is ~2.6 people.

**4.3 Output**

```
   SepalLength  SepalWidth  PetalLength  PetalWidth         Name
0          5.1         3.5          1.4         0.2  Iris-setosa
1          4.9         3.0          1.4         0.2  Iris-setosa
2          4.7         3.2          1.3         0.2  Iris-setosa
3          4.6         3.1          1.5         0.2  Iris-setosa
4          5.0         3.6          1.4         0.2  Iris-setosa
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   SepalLength  150 non-null    float64
 1   SepalWidth   150 non-null    float64
 2   PetalLength  150 non-null    float64
 3   PetalWidth   150 non-null    float64
 4   Name         150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

| | SepalLength | SepalWidth | PetalLength | PetalWidth |
|---|---|---|---|---|
| **count** | 150.000000 | 150.000000 | 150.000000 | 150.000000 |
| **mean** | 5.843333 | 3.054000 | 3.758667 | 1.198667 |
| **std** | 0.828066 | 0.433594 | 1.764420 | 0.763161 |
| **min** | 4.300000 | 2.000000 | 1.000000 | 0.100000 |
| **25%** | 5.100000 | 2.800000 | 1.600000 | 0.300000 |
| **50%** | 5.800000 | 3.000000 | 4.350000 | 1.300000 |
| **75%** | 6.400000 | 3.300000 | 5.100000 | 1.800000 |
| **max** | 7.900000 | 4.400000 | 6.900000 | 2.500000 |

*This code does not create graphs, but it produces tabular outputs*

SepalLength

mean ≈ 5.8433
min = 4.3
max = 7.9

SepalWidth

mean ≈ 3.0540
min = 2.0
max = 4.4

PetalLength

mean ≈ 3.7587
min = 1.0
max = 6.9

PetalWidth

mean ≈ 1.1987
min = 0.1
max = 2.5

These values represent physical measurements of iris flower parts.

# 5. Analytical Recapitulation

This code integrates two fully developed workflows: (1) loading the dataset and performing exploratory data analysis (EDA) with Pandas on both tips.csv and iris.csv, and (2) applying dimensionality reduction and visualizing the Iris dataset using t-SNE. To start, the program imports essential Python libraries, including Pandas for data manipulation, NumPy for handling numerical operations, Matplotlib for graphical representation, and various scikit-learn modules for dataset loading, feature scaling, t-SNE application, and assessment metrics. The initial workflow retrieves the tips.csv dataset from a raw GitHub URL via pd.read_csv(url) and saves it into a Pandas DataFrame. The code subsequently displays the first few rows with .head() to confirm successful loading and examine sample values. Following this, .info() provides an overview of the dataset's structure, revealing a total of 244 rows and 7 columns, along with data types, non-null counts, and memory information. The dataset comprises numeric columns like total_bill, tip, and size, alongside categorical columns such as sex, smoker, day, and time. Lastly, .describe() generates summary statistics for the numeric columns, including count, mean, standard deviation, minimum, maximum, and quartiles. These statistics are based on mathematical formulas such as mean $\mu = \frac{1}{n}\sum x_i$and standard deviation $\sigma = \sqrt{\frac{1}{n-1}\sum(x_i - \mu)^2}$, helping identify data distribution and possible outliers. The second workflow involves loading the Iris dataset, which can be done through either scikit-learn's load_iris(return_X_y=True) function or by importing the iris.csv file from GitHub's raw URL into a DataFrame. This process generates a feature matrix \(X\) that comprises measurements of the flowers and a label vector \(y\) that holds the class labels. The code utilizes .head(), .info(), and .describe() to analyze the dataset's structure, verifying that there are 150 entries and 5 columns in the CSV version. Among these, four columns represent numerical measurements of the flowers (SepalLength, SepalWidth, PetalLength, PetalWidth), while one column indicates the categorical label Name. After inspection, the t-SNE pipeline begins by standardizing the Iris feature matrix using `StandardScaler`, applying normalization $z = (x - \mu)/\sigma$so that each feature has mean 0 and standard deviation 1, which is necessary because t-SNE is highly sensitive to feature scaling. The t-SNE algorithm is initialized with various parameters: n_components=2 for a 2D embedding, perplexity=30 to manage the size of neighborhoods, learning_rate=200 to dictate the size of the gradient descent steps, and n_iter=1000 for the number of optimization iterations. It uses Mahalanobis distance as the metric for similarity measurement and sets random_state=42 to ensure reproducibility. The t-SNE process transforms high-dimensional pairwise distances into probability distributions \( p_{ij} \), calculates low-dimensional similarity probabilities \( q_{ij} \) using a Student t-distribution to alleviate the crowding issue, and aims to minimize the difference between \( P \) and \( Q \) through the KL divergence \( KL(P \| Q) \) via gradient descent optimization. Once the final 2D embedding is produced, the code assesses the quality of the dimensionality reduction by calculating a trustworthiness score of 0.9361, which indicates a strong retention of local neighborhood relationships from the original space, along with a silhouette score of 0.3351, suggesting moderate separation among the three iris classes. Finally, the reduced 2D points are visualized using a scatter plot where each point represents a flower sample and colors represent class labels. The visualization shows one clearly separated cluster (Setosa) and overlapping clusters for the other two classes (Versicolor and Virginica), which aligns with known Iris dataset behavior. Overall, the combined code demonstrates a complete workflow starting from online dataset loading and statistical inspection, followed by preprocessing, dimensionality reduction using t-SNE, quantitative embedding evaluation, and final 2D visualization.