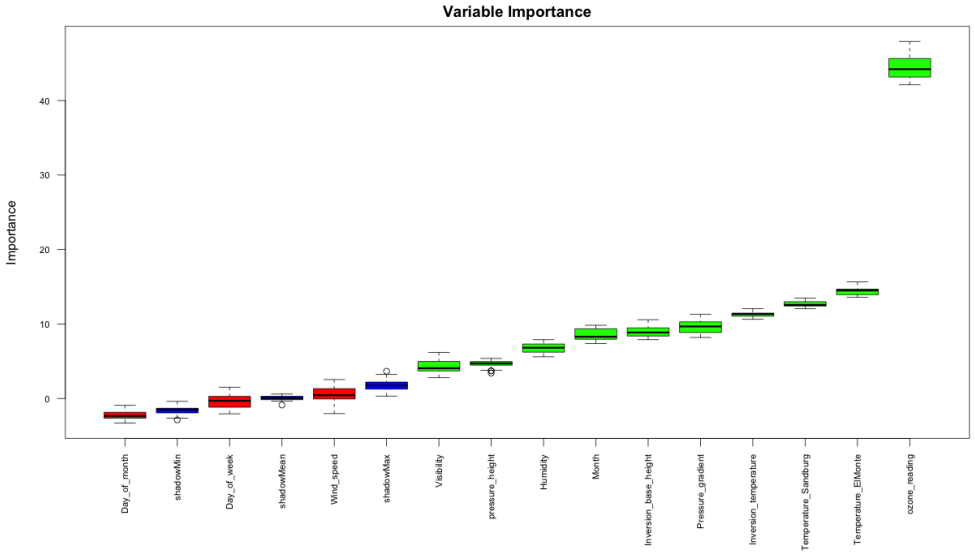
Attribute Subset Selection

- Another way to reduce dimensionality of data
- Redundant attributes
  - Duplicate much or all of the information contained in one or more other attributes
    - E.g., purchase price of a product and the amount of sales tax paid
- Irrelevant attributes
  - Contain no information that is useful for the data mining task at hand
    - Ex. A student's ID is often irrelevant to the task of predicting his/her GPA



Attribute Subset Selection

- **Overview**
  - **Goal**: Reduce data dimensions by removing irrelevant or redundant attributes.
  - **Benefit**: Makes data mining more focused, efficient, and easier to understand.
- **Challenges**
  - Picking the "best" attributes is tough and time-consuming, especially when you're not sure what the data's behavior is.
  - Using wrong attributes can lead to poor quality patterns and slow down the mining process.
- **How to Find a "Good" Subset?**
  - With (d) attributes, there are (2^d) possible subsets. Searching all is impractical.
  - Heuristic methods, often greedy, are commonly used. They make the best local choice in the hope it will lead to a globally good solution.

Decision Tree Induction for Attribute Subset Selection

- **What is a Decision Tree?**
  - A decision tree is like a flowchart.
  - **Internal Nodes**: Represent a test on an attribute (e.g., Is age > 25?).
  - **Branches**: The outcomes of the test (e.g., Yes or No).
  - **Leaf Nodes**: Final class prediction (e.g., Will Buy, Won't Buy).

How Does it Work for Classification?

- Start with all data at the root node.
- Pick the "best" attribute to split the data into subsets. "Best" usually means that the attribute does a good job separating the different classes.
- Repeat the process for each subset until you reach a conclusion for all paths in the tree.

Using for Attribute Subset Selection

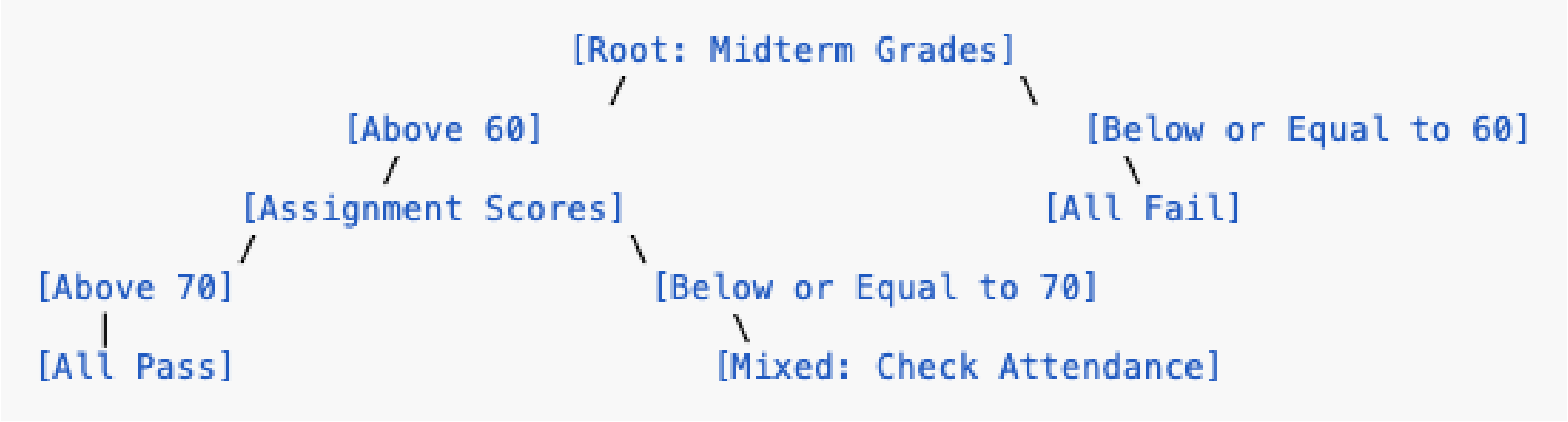
- Build a decision tree using your data.
- Any attribute that appears in the tree is considered relevant for your task.
- Attributes that don't appear are considered irrelevant.
- Why is it Useful?
  - It automatically identifies the most important attributes for you
  - The process is efficient and can handle large datasets

Decision Tree Example

- Imagine you are trying to predict whether a student will pass or fail a course. Data looks like this:

Attendance	Assignment	Midterm Grades	Participation in Online Forums	Pass/Fail
90	85	70	40	Pass
80	75	65	35	Pass
85	40	50	20	Fail
70	30	30	15	Fail
60	45	35	25	Fail
95	90	80	60	Pass
88	70	60	30	Pass
75	60	55	20	Fail

How the Decision Tree Algorithm Would Work:



This simplified tree uses "Midterm Grades" and "Assignment Scores" for making decisions, suggesting these might be the most relevant attributes for predicting "Pass/Fail"

Next, checkout the Jypyter notebook

Feature Subset Selection Using Decision Tree on Abalone Dataset

Steps:

1. **Load the Data**: Read the CSV file to understand its structure.
2. **Preprocessing**: Handle missing values, if any, and encode categorical features.
3. **Decision Tree**: Train a decision tree model on the dataset.
4. **Feature Importance**: Extract feature importances from the decision tree model.
5. **Subset Selection**: Select a subset of features based on their importance.

Let's start with the first step and load the data.

The dataset consists of the following columns:

- `Sex` : Gender of the abalone (M, F, I)
- `Length` : Length measurement
- `Diameter` : Diameter measurement
- `Height` : Height measurement
- `Whole weight` : Weight of the whole abalone
- `Shucked weight` : Weight after removing the shell
- `Viscera weight` : Weight of the gut
- `Shell weight` : Weight of the shell
- `Rings` : Age of the abalone (target variable)

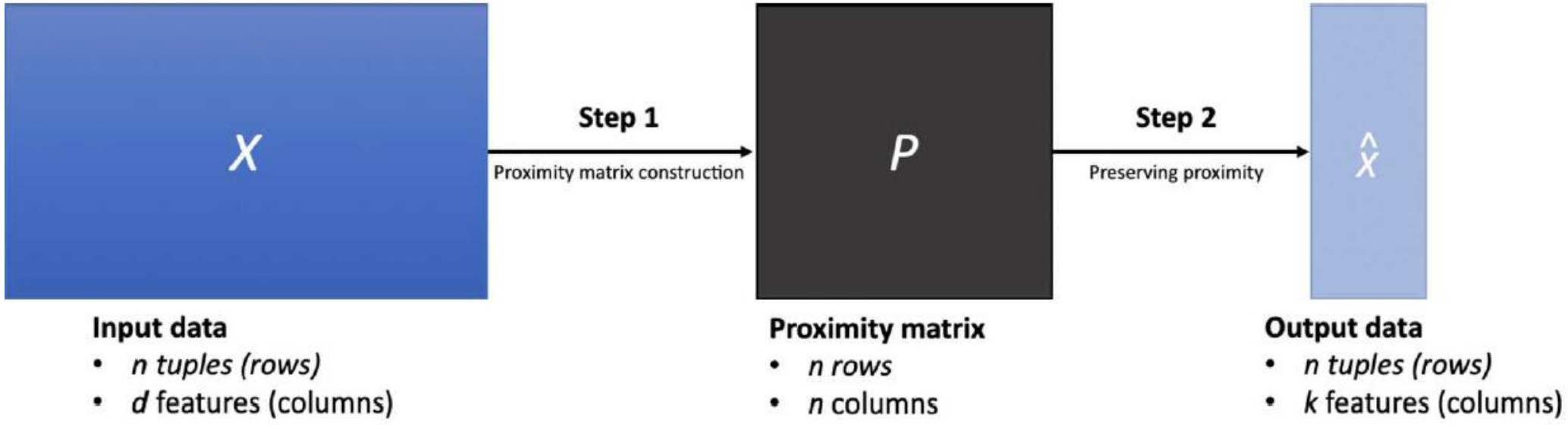
[1]:
```
import pandas as pd

# Load the data
file_path = 'abalone.csv'
df = pd.read_csv(file_path)
```

2.6.3 Nonlinear Dimensionality Reduction Methods

- PCA is a linear method for dimensionality reduction
  - Each principal component is a linear combination of the original input attributes
  - It works well if the input data approximately follows a Gaussian distribution or forms a few linearly separable clusters
  - When the input data are linearly inseparable, however, PCA becomes ineffective. Luckily, there are many nonlinear methods we can resort to in this case.

General Steps for Nonlinear Methods:

- **Constructing Proximity Matrix:** Create a matrix that tells us how similar each data point is to every other data point. This is called the proximity matrix 'P'.
- **Preserving Proximity**: Use this proximity information to create a new, lower-dimensional representation of the data. This new representation should keep the relationships (proximity) between data points as close as possible to the original.

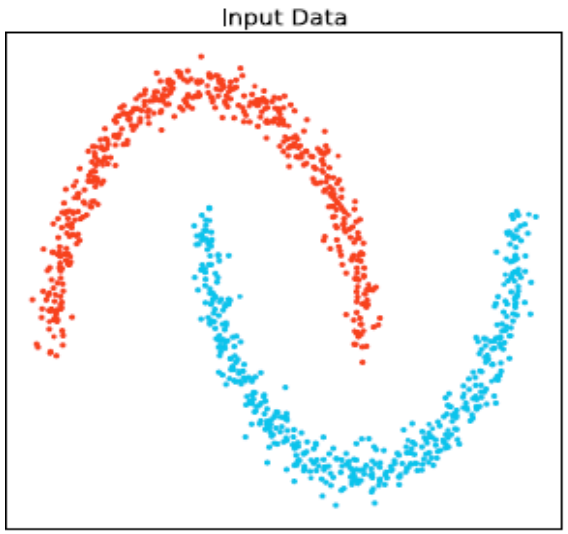


Two Methods: KPCA and SNE

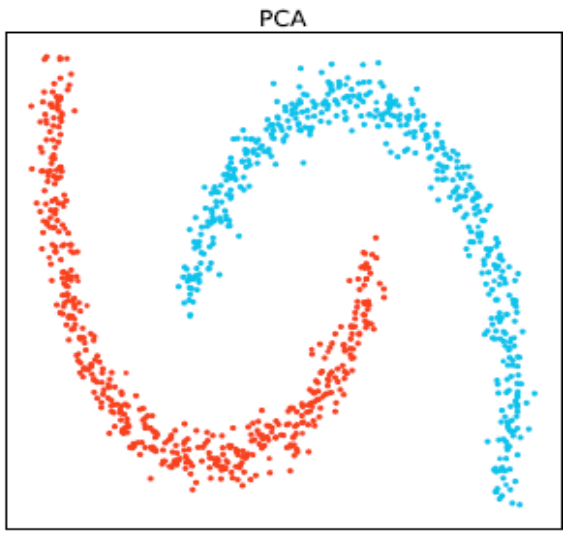
- KPCA (Kernel PCA):
  - Step 1: Uses a function (kernel function) to calculate the affinity or similarity between each pair of points.
  - Step 2: After computing the affinity, KPCA tries to maintain these in the lower-dimensional space.
- SNE (Stochastic Neighbor Embedding):
  - Step 1: A different approach to creating the proximity matrix
  - Step 2: Similar to KPCA, SNE also tries to preserve these relationships in the lower-dimensional space.

Example: Comparison on Nonlinear Data Points: Linear vs. Nonlinear Dimensional Reduction Methods

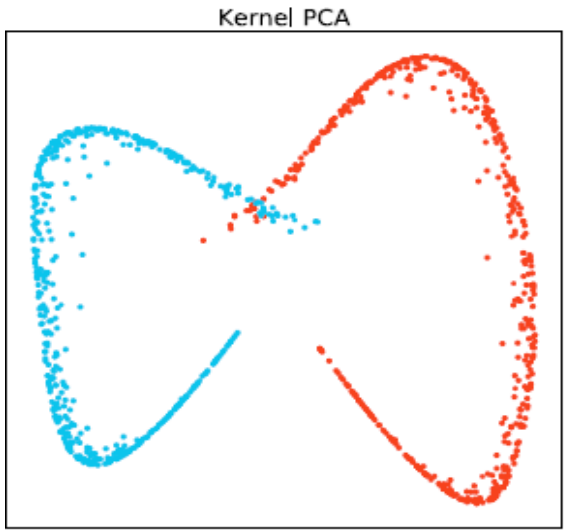
- Visualization: An example of linear vs. nonlinear dimensionality reduction methods
  - Given a collection of input data in 2-D space (Fig. (a)): Red and blue data points are not linearly separable
  - PCA transformation can not make it linearly separable
  - KPCA can make the points linearly separable
  - t-SNE (t-distributional NSE) can make them linearly separable



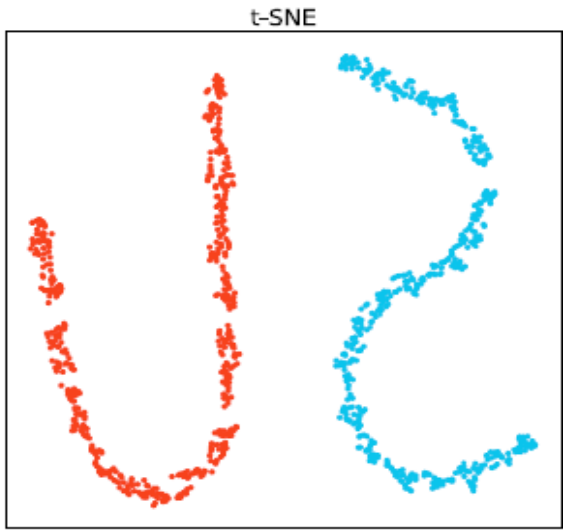
(a) Input data



(b) PCA



(c) KPCA



(d) t-SNE