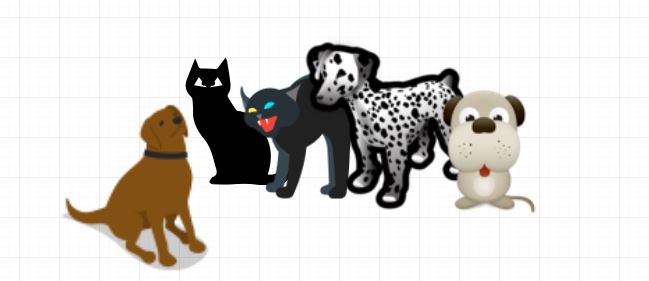
**Unit – IV**

**Unsupervised learning**

Unsupervised learning is the training of a machine using information that is neither classified nor labeled and allowing the algorithm to act on that information without guidance. Here the task of the machine is to group unsorted information according to similarities, patterns, and differences without any prior training of data.

Unlike supervised learning, no teacher is provided that means no training will be given to the machine. Therefore the machine is restricted to find the hidden structure in unlabeled data by itself.   
**For instance**, suppose it is given an image having both dogs and cats which it has never seen. 



Thus the machine has no idea about the features of dogs and cats so we can’t categorize it as ‘dogs and cats ‘. But it can categorize them according to their similarities, patterns, and differences, i.e., we can easily categorize the above picture into two parts. The first may contain all pics having **dogs** in them and the second part may contain all pics having **cats** in them. Here you didn’t learn anything before, which means no training data or examples.

It allows the model to work on its own to discover patterns and information that was previously undetected. It mainly deals with unlabelled data.

Unsupervised learning is classified into two categories of algorithms:

1.clustering

2.association

**Clustering**:

Clustering or cluster analysis is a machine learning technique, which groups the unlabelled dataset. It can be defined as **"A way of grouping the data points into different clusters, consisting of similar data points. The objects with the possible similarities remain in a group that has less or no similarities with another group."**

It does it by finding some similar patterns in the unlabelled dataset such as shape, size, color, behavior, etc., and divides them as per the presence and absence of those similar patterns.

It is an [unsupervised learning](https://www.javatpoint.com/unsupervised-machine-learning) method, hence no supervision is provided to the algorithm, and it deals with the unlabeled dataset.

After applying this clustering technique, each cluster or group is provided with a cluster-ID. ML system can use this id to simplify the processing of large and complex datasets.

The clustering technique can be widely used in various tasks. Some most common uses of this technique are:

* Market Segmentation
* Statistical data analysis
* Social network analysis
* Image segmentation
* Anomaly detection, etc.

**K-Means:**

K-Means Clustering is an [Unsupervised Learning algorithm](https://www.javatpoint.com/unsupervised-machine-learning), which groups the unlabeled dataset into different clusters. Here K defines the number of pre-defined clusters that need to be created in the process, as if K=2, there will be two clusters, and for K=3, there will be three clusters, and so on.

It is an iterative algorithm that divides the unlabeled dataset into k different clusters in such a way that each dataset belongs only one group that has similar properties.

It allows us to cluster the data into different groups and a convenient way to discover the categories of groups in the unlabeled dataset on its own without the need for any training.

It is a centroid-based algorithm, where each cluster is associated with a centroid. The main aim of this algorithm is to minimize the sum of distances between the data point and their corresponding clusters.

The algorithm takes the unlabeled dataset as input, divides the dataset into k-number of clusters, and repeats the process until it does not find the best clusters. The value of k should be predetermined in this algorithm.

The k-means [clustering](https://www.javatpoint.com/clustering-in-machine-learning) algorithm mainly performs two tasks:

* Determines the best value for K center points or centroids by an iterative process.
* Assigns each data point to its closest k-center. Those data points which are near to the particular k-center, create a cluster.

Hence each cluster has datapoints with some commonalities, and it is away from other clusters.

The below diagram explains the working of the K-means Clustering Algorithm:



How does the K-Means Algorithm Work?

The working of the K-Means algorithm is explained in the below steps:

**Step-1:** Select the number K to decide the number of clusters.

**Step-2:** Select random K points or centroids. (It can be other from the input dataset).

**Step-3:** Assign each data point to their closest centroid, which will form the predefined K clusters.

**Step-4:** Calculate the variance and place a new centroid of each cluster.

**Step-5:** Repeat the third steps, which means reassign each datapoint to the new closest centroid of each cluster.

**Step-6:** If any reassignment occurs, then go to step-4 else go to FINISH.

**Step-7**: The model is ready.

**Image Segmentation By Clustering**

 Image Segmentation: In computer vision, image segmentation is the process of partitioning an image into multiple segments. The goal of segmenting an image is to change the representation of an image into something that is more meaningful and easier to analyze. It is usually used for locating objects and creating boundaries.

It is not a great idea to process an entire image because many parts in an image may not contain any useful information. Therefore, by segmenting the image, we can make use of only the important segments for processing.

An image is basically a set of given pixels. In image segmentation, pixels which have similar attributes are grouped together. Image segmentation creates a pixel-wise mask for objects in an image which gives us a more comprehensive and granular understanding of the object.

Uses:

1. Used in self-driving cars. Autonomous driving is not possible without object detection which involves segmentation.
2. Used in the healthcare industry. Helpful in segmenting cancer cells and tumours using which their severity can be gauged.

There are many more uses of image segmentation.

In this article, we will perform segmentation on an image of the monarch butterfly using a clustering method called K Means Clustering.

**Stages of Data preprocessing for K-means Clustering**

1. Data Cleaning

* Removing duplicates
* Removing irrelevant observations and errors
* Removing unnecessary columns
* Handling inconsistent data
* Handling outliers and noise

2. Handling missing data

3. Data Integration

4. Data Transformation

* Feature Construction
* Handling skewness
* Data Scaling

5. Data Reduction

* Removing dependent (highly correlated) variables
* Feature selection
* PCA

**Using Clustering for Semi-Supervised Learning.**

Semi-supervised clustering is a method that partitions unlabeled data by creating the use of domain knowledge. It is generally expressed as pair wise constraints between instances or just as an additional set of labeled instances.

The quality of unsupervised clustering can be essentially improved using some weak structure of supervision, for instance, in the form of pair wise constraints (i.e., pairs of objects labeled as belonging to similar or different clusters). Such a clustering procedure that depends on user feedback or guidance constraints is known as semi supervised clustering.

There are several methods for semi-supervised clustering that can be divided into two classes which are as follows −

**Constraint-based semi-supervised clustering** − It can be used based on user-provided labels or constraints to support the algorithm toward a more appropriate data partitioning. This contains modifying the objective function depending on constraints or initializing and constraining the clustering process depending on the labeled objects.

**Distance-based semi-supervised clustering** − It can be used to employ an adaptive distance measure that is trained to satisfy the labels or constraints in the supervised data. Multiple adaptive distance measures have been utilized, including string-edit distance trained using Expectation-Maximization (EM), and Euclidean distance changed by the shortest distance algorithm.

An interesting clustering method, known as CLTree (CLustering based on decision TREEs). It integrates unsupervised clustering with the concept of supervised classification. It is an instance of constraint-based semi-supervised clustering. It changes a clustering task into a classification task by considering the set of points to be clustered as belonging to one class, labeled as “Y,” and inserts a set of relatively uniformly distributed, “nonexistence points” with a multiple class label, “N.”

The problem of partitioning the data area into data (dense) regions and empty (sparse) regions can then be changed into a classification problem. These points can be considered as a set of “Y” points. It shows the addition of a collection of uniformly distributed “N” points, defined by the “o” points.

The original clustering problem is thus changed into a classification problem, which works out a design that distinguishes “Y” and “N” points. A decision tree induction method can be used to partition the two-dimensional space. Two clusters are recognized, which are from the “Y” points only.

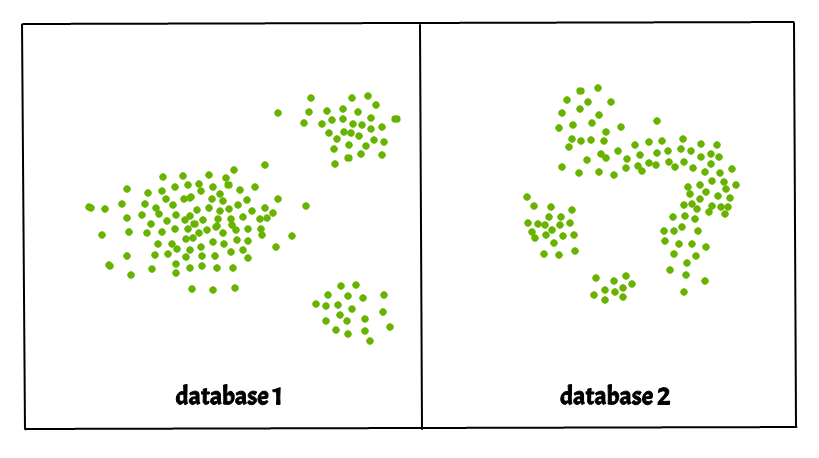
It can be used to insert a large number of “N” points to the original data can introduce unnecessary overhead in the calculation. Moreover, it is unlikely that some points added would truly be uniformly distributed in a very high-dimensional space as this can need an exponential number of points.

**DBSCAN Clustering**

Clustering analysis or simply Clustering is basically an Unsupervised learning method that divides the data points into a number of specific batches or groups, such that the data points in the same groups have similar properties and data points in different groups have different properties in some sense. It comprises many different methods based on differential evolution.   
E.g. K-Means (distance between points), Affinity propagation (graph distance), Mean-shift (distance between points), DBSCAN (distance between nearest points), Gaussian mixtures (Mahalanobis distance to centers), Spectral clustering (graph distance) etc.

Fundamentally, all clustering methods use the same approach i.e. first we calculate similarities and then we use it to cluster the data points into groups or batches. Here we will focus on **Density-based spatial clustering of applications with noise** (DBSCAN) clustering method.

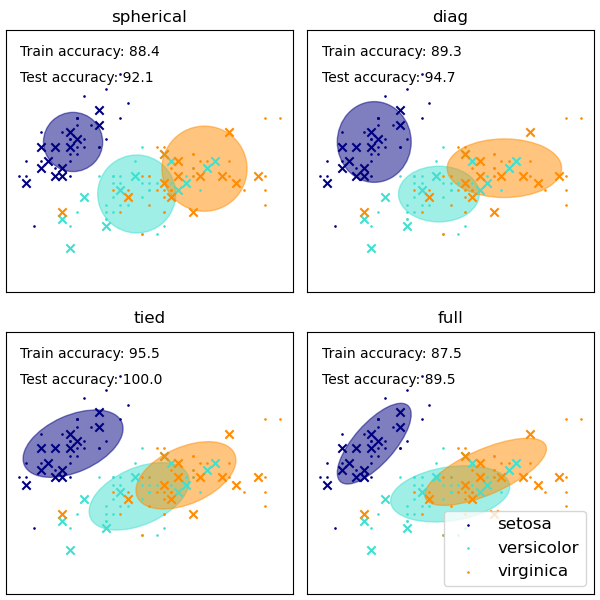
Clusters are dense regions in the data space, separated by regions of the lower density of points. The ***DBSCAN algorithm*** is based on this intuitive notion of “clusters” and “noise”. The key idea is that for each point of a cluster, the neighborhood of a given radius has to contain at least a minimum number of points.



[**Gaussian Mixture**](https://scikit-learn.org/stable/modules/generated/sklearn.mixture.GaussianMixture.html#sklearn.mixture.GaussianMixture)

The [**Gaussian Mixture**](https://scikit-learn.org/stable/modules/generated/sklearn.mixture.GaussianMixture.html#sklearn.mixture.GaussianMixture) object implements the [expectation-maximization](https://scikit-learn.org/stable/modules/mixture.html#expectation-maximization) (EM) algorithm for fitting mixture-of-Gaussian models. It can also draw confidence ellipsoids for multivariate models, and compute the Bayesian Information Criterion to assess the number of clusters in the data. A [**Gaussian Mixture. Fit**](https://scikit-learn.org/stable/modules/generated/sklearn.mixture.GaussianMixture.html#sklearn.mixture.GaussianMixture.fit) method is provided that learns a Gaussian Mixture Model from train data. Given test data, it can assign to each sample the Gaussian it most probably belongs to using the [**Gaussian Mixture. Predict**](https://scikit-learn.org/stable/modules/generated/sklearn.mixture.GaussianMixture.html#sklearn.mixture.GaussianMixture.predict) method.

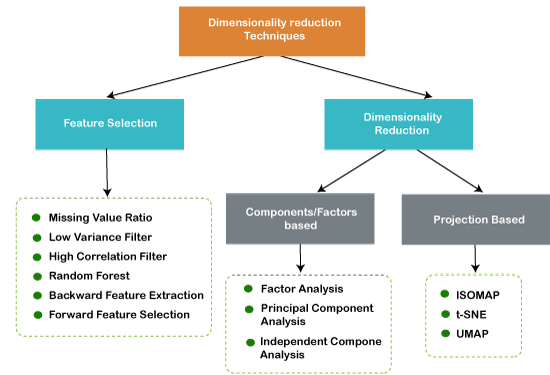
The **[GaussianMixture](https://scikit-learn.org/stable/modules/generated/sklearn.mixture.GaussianMixture.html" \l "sklearn.mixture.GaussianMixture" \o "sklearn.mixture.GaussianMixture)** comes with different options to constrain the covariance of the difference classes estimated: spherical, diagonal, tied or full covariance.



**Dimensionality reduction**

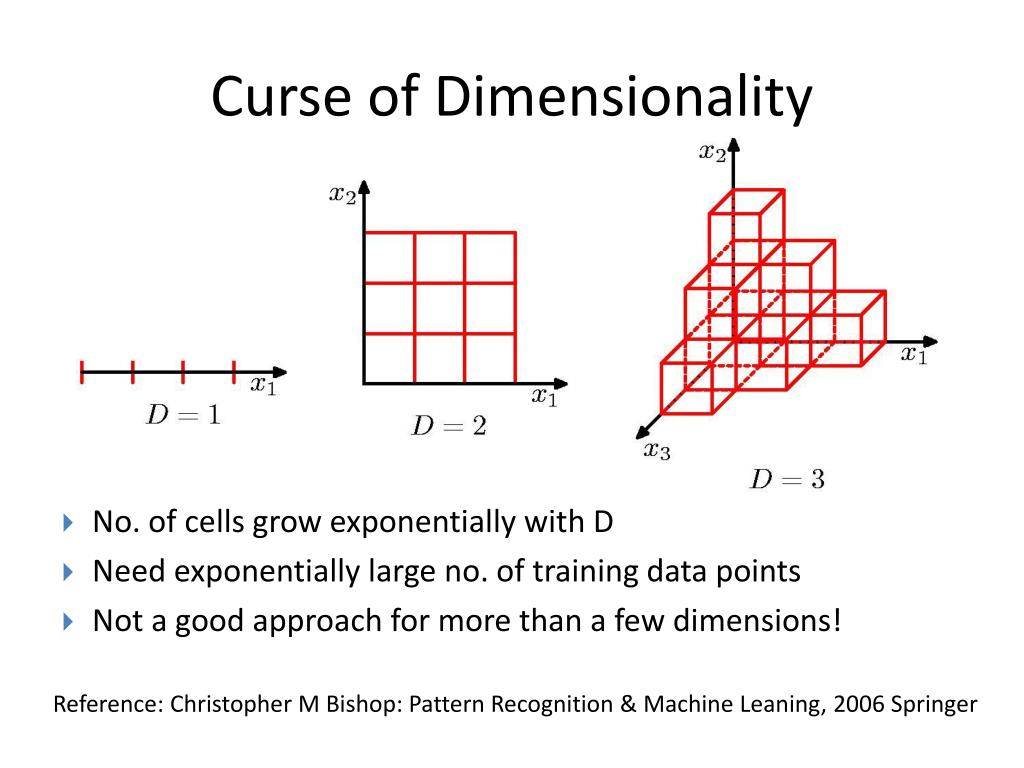
Dimensionality reduction technique can be defined as, ***"It is a way of converting the higher dimensions dataset into lesser dimensions dataset ensuring that it provides similar information."*** These techniques are widely used in [machine learning](https://www.javatpoint.com/machine-learning) for obtaining a better fit predictive model while solving the classification and regression problems.

It is commonly used in the fields that deal with high-dimensional data, such as **speech recognition, signal processing, bioinformatics, etc. It can also be used for data visualization, noise reduction, cluster analysis**, etc.



**Curse of Dimensionality**

Curse of Dimensionality refers to a set of problems that arise when working with high-dimensional data. The dimension of a dataset corresponds to the number of attributes/features that exist in a dataset. A dataset with a large number of attributes, generally of the order of a hundred or more, is referred to as high dimensional data. Some of the difficulties that come with high dimensional data manifest during analyzing or visualizing the data to identify patterns, and some manifest while training machine learning models. The difficulties related to training machine learning models due to high dimensional data is referred to as ‘Curse of Dimensionality’. The popular aspects of the curse of dimensionality; ‘data sparsity’ and ‘distance concentration’ are discussed in the following sections.



**Main Approaches for Dimensionality Reduction**

1. Principal Component Analysis (PCA)

Principal Component Analysis is one of the leading linear techniques of dimensionality reduction. This method performs a direct mapping of the data to a lesser dimensional space in a way that maximizes the variance of the data in the low-dimensional representation.

Essentially, it is a statistical procedure that orthogonally converts the ‘*n’* coordinates of a dataset into a new set of *n* coordinates, known as the principal components. This conversion results in the creation of the first principal component having the maximum variance. Each succeeding principal component bears the highest possible variance, under the condition that it is orthogonal (not correlated) to the preceding components.

The PCA conversion is sensitive to the relative scaling of the original variables. Thus, the data column ranges must first be normalized before implementing the PCA method. Another thing to remember is that using the PCA approach will make your dataset lose its interpretability. So, if interpretability is crucial to your analysis, PCA is not the right dimensionality reduction method for your project.

2. Non-negative matrix factorization (NMF)

NMF breaks down a non-negative matrix into the product of two non-negative ones. This is what makes the NMF method a valuable tool in areas that are primarily concerned with non-negative signals (for instance, astronomy). The multiplicative update rule by Lee & Seung improved the NMF technique by – including uncertainties, considering missing data and parallel computation, and sequential construction.

These inclusions contributed to making the NMF approach stable and linear. Unlike PCA, NMF does not eliminate the mean of the matrices, thereby creating unphysical non-negative fluxes. Thus, NMF can preserve more information than the PCA method.

Sequential NMF is characterized by a stable component base during construction and a linear modeling process. This makes it the perfect tool in astronomy. Sequential NMF can preserve the flux in the direct imaging of circumstellar structures in astronomy, such as detecting exoplanets and direct imaging of circumstellar disks.

3. Linear discriminant analysis (LDA)

The linear discriminant analysis is a generalization of Fisher’s linear discriminant method that is widely applied in statistics, pattern recognition, and machine learning. The LDA technique aims to find a linear combination of features that can characterize or differentiate between two or more classes of objects. LDA represents data in a way that maximizes class separability. While objects belonging to the same class are juxtaposed via projection, objects from different classes are arranged far apart.

4. Generalized discriminant analysis (GDA)

The generalized discriminant analysis is a nonlinear discriminant analysis that leverages the kernel function operator. Its underlying theory matches very closely to that of support vector machines (SVM), such that the GDA technique helps to map the input vectors into high-dimensional feature space. Just like the LDA approach, GDA also seeks to find a projection for variables in a lower-dimensional space by maximizing the ratio of between-class scatters to within-class scatter.

5. Missing Values Ratio

When you explore a given dataset, you might find that there are some missing values in the dataset. The first step in dealing with missing values is to identify the reason behind them. Accordingly, you can then impute the missing values or drop them altogether by using the befitting methods. This approach is perfect for situations when there are a few missing values.

However, what to do when there are too many missing values, say, over 50%? In such situations, you can set a threshold value and use the missing values ratio method. The higher the threshold value, the more aggressive will be the dimensionality reduction. If the percentage of missing values in a variable exceeds the threshold, you can drop the variable.

Generally, data columns having numerous missing values hardly contain useful information. So, you can remove all the data columns having missing values higher than the set threshold.

6. Low Variance Filter

Just as you use the missing values ratio method for missing variables, so for constant variables, there’s the low variance filter technique. When a dataset has constant variables, it is not possible to improve the model’s performance. Why? Because it has zero variance.

In this method also, you can set a threshold value to wean out all the constant variables. So, all the data columns with variance lower than the threshold value will be eliminated. However, one thing you must remember about the low variance filter method is that variance is range dependent. Thus, normalization is a must before implementing this dimensionality reduction technique.

7. High Correlation Filter

If a dataset consists of data columns having a lot of similar patterns/trends, these data columns are highly likely to contain identical information. Also, dimensions that depict a higher correlation can adversely impact the model’s performance. In such an instance, one of those variables is enough to feed the ML model.

For such situations, it’s best to use the Pearson correlation matrix to identify the variables showing a high correlation. Once they are identified, you can select one of them using VIF (Variance Inflation Factor). You can remove all the variables having a higher value ( VIF > 5 ). In this approach, you have to calculate the correlation coefficient between numerical columns ([Pearson’s Product Moment Coefficient](http://en.wikipedia.org/wiki/Pearson_product-moment_correlation_coefficient)) and between nominal columns ([Pearson’s chi-square value](http://en.wikipedia.org/wiki/Pearson%27s_chi-squared_test)). Here, all the pairs of columns having a correlation coefficient higher than the set threshold will be reduced to 1.

Since correlation is scale-sensitive, you must perform column normalization.

8. Backward Feature Elimination

In the backward feature elimination technique, you have to begin with all ‘n’ dimensions. Thus, at a given iteration, you can train a specific classification algorithm is trained on *n* input features. Now, you have to remove one input feature at a time and train the same model on *n-1* input variables *n* times. Then you remove the input variable whose elimination generates the smallest increase in the error rate, which leaves behind *n-1* input features. Further, you repeat the classification using *n-2* features, and this continues till no other variable can be removed.

Each iteration (*k)*creates a model trained on n-k features having an error rate of *e(k)*. Following this, you must select the maximum bearable error rate to define the smallest number of features needed to reach that classification performance with the given ML algorithm.

9. Forward Feature Construction

The forward feature construction is the opposite of the backward feature elimination method. In the forward feature construction method, you begin with one feature and continue to progress by adding one feature at a time (this is the variable that results in the greatest boost in performance).

Both forward feature construction and backward feature elimination are time and computation-intensive. These methods are best suited for datasets that already have a low number of input columns.

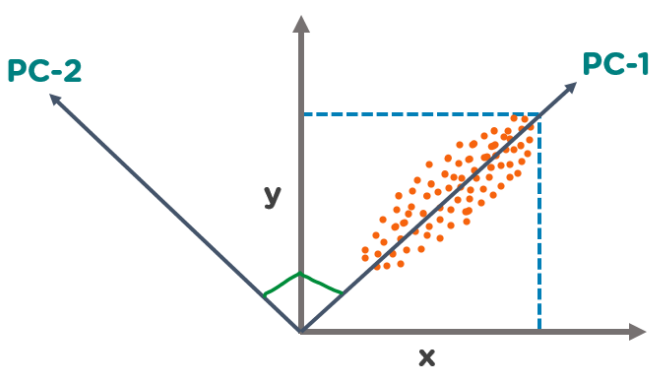
10. Random Forests

Random forests are not only excellent classifiers but are also extremely useful for feature selection. In this dimensionality reduction approach, you have to carefully construct an extensive network of trees against a target attribute. For instance, you can create a large set (say, 2000) of shallow trees (say, having two levels), where each tree is trained on a minor fraction (3) of the total number of attributes.

The aim is to use each attribute’s usage statistics to identify the most informative subset of features. If an attribute is found to be the best split, it usually contains an informative feature that is worthy of consideration. When you calculate the score of an attribute’s usage statistics in the random forest in relation to other attributes, it gives you the most predictive attributes.

**Principal Component Analysis**

The Principal Component Analysis is a popular unsupervised learning technique for reducing the dimensionality of data. It increases interpretability yet, at the same time, it minimizes information loss. It helps to find the most significant features in a dataset and makes the data easy for plotting in 2D and 3D. PCA helps in finding a sequence of linear combinations of variables.



In the above figure, we have several points plotted on a 2-D plane. There are two principal components. PC1 is the primary principal component that explains the maximum variance in the data. PC2 is another principal component that is orthogonal to PC1.

**What is Scikit-Learn (Sklearn)**

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon **NumPy, SciPy** and **Matplotlib**.

**Installation**

If you already installed NumPy and Scipy, following are the two easiest ways to install scikit-learn −

Using pip

Following command can be used to install scikit-learn via pip −

pip install -U scikit-learn

Using conda

Following command can be used to install scikit-learn via conda −

conda install scikit-learn

On the other hand, if NumPy and Scipy is not yet installed on your Python workstation then, you can install them by using either **pip** or **conda**.

Another option to use scikit-learn is to use Python distributions like ***Canopy*** and ***Anaconda*** because they both ship the latest version of scikit-learn.

**Features**

Rather than focusing on loading, manipulating and summarising data, Scikit-learn library is focused on modeling the data. Some of the most popular groups of models provided by Sklearn are as follows −

**Supervised Learning algorithms** − Almost all the popular supervised learning algorithms, like Linear Regression, Support Vector Machine (SVM), Decision Tree etc., are the part of scikit-learn.

**Unsupervised Learning algorithms** − On the other hand, it also has all the popular unsupervised learning algorithms from clustering, factor analysis, PCA (Principal Component Analysis) to unsupervised neural networks.

**Clustering** − This model is used for grouping unlabeled data.

**Cross Validation** − It is used to check the accuracy of supervised models on unseen data.

**Dimensionality Reduction** − It is used for reducing the number of attributes in data which can be further used for summarisation, visualisation and feature selection.

**Ensemble methods** − As name suggest, it is used for combining the predictions of multiple supervised models.

**Feature extraction** − It is used to extract the features from data to define the attributes in image and text data.

**Feature selection** − It is used to identify useful attributes to create supervised models.

**Open Source** − It is open source library and also commercially usable under BSD license.

**KERNEL PCA:**

PCA is a linear method. That is it can only be applied to datasets which are linearly separable. It does an excellent job for datasets, which are linearly separable. But, if we use it to non-linear datasets, we might get a result which may not be the optimal dimensionality reduction. Kernel PCA uses a kernel function to project dataset into a higher dimensional feature space, where it is linearly separable. It is similar to the idea of Support Vector Machines. There are various kernel methods like linear, polynomial, and gaussian.

In the kernel space the two classes are linearly separable. Kernel PCA uses a kernel function to project the dataset into a higher-dimensional space, where it is linearly separable. Finally, we applied the kernel PCA to a non-linear dataset using scikit-learn.

Kernel Principal Component Analysis (PCA) is a technique for dimensionality reduction in machine learning that uses the concept of kernel functions to transform the data into a high-dimensional feature space. In traditional PCA, the data is transformed into a lower-dimensional space by finding the principal components of the covariance matrix of the data. In kernel PCA, the data is transformed into a high-dimensional feature space using a non-linear mapping function, called a kernel function, and then the principal components are found in this high-dimensional space.

**Advantages of Kernel PCA:**

1. Non-linearity: Kernel PCA can capture non-linear patterns in the data that are not possible with traditional linear PCA.
2. Robustness: Kernel PCA can be more robust to outliers and noise in the data, as it considers the global structure of the data, rather than just local distances between data points.
3. Versatility: Different types of kernel functions can be used in kernel PCA to suit different types of data and different objectives.

**Disadvantages of Kernel PCA:**

1. Complexity: Kernel PCA can be computationally expensive, especially for large datasets, as it requires the calculation of eigenvectors and eigenvalues.
2. Model selection: Choosing the right kernel function and the right number of components can be challenging and may require expert knowledge or trial and error.

