Unsupervised Learning

Unsupervised learning is a type of machine learning where an algorithm learns patterns and structures in data **without labeled outputs**. Unlike supervised learning, where the model learns from input-output pairs, unsupervised learning identifies hidden patterns, relationships, or groupings in the data.

Types of Unsupervised Learning

1. **Clustering** – Grouping similar data points together.
2. **Dimensionality Reduction** – Reducing the number of features while retaining key information.
3. **Anomaly Detection** – Identifying unusual patterns or outliers.
4. **Association Rules** – Discovering relationships between variables in large datasets.

Examples of Unsupervised Learning

1. Clustering

- **Example: Customer Segmentation**
  - E-commerce platforms use clustering algorithms like **K-Means** to group customers based on their purchasing behavior (e.g., frequent buyers, occasional shoppers, or bargain hunters).
- **Example: Image Segmentation**
  - In medical imaging, clustering helps segment different tissues in MRI scans.

2. Dimensionality Reduction

- **Example: PCA (Principal Component Analysis) for Face Recognition**
  - PCA helps in reducing high-dimensional image data for better processing in facial recognition systems.
- **Example: t-SNE for Data Visualization**
  - t-SNE (t-distributed Stochastic Neighbor Embedding) helps visualize high-dimensional data in 2D/3D.

3. Anomaly Detection

- **Example: Fraud Detection in Banks**
  - Algorithms like **Isolation Forest** or **DBSCAN** detect unusual credit card transactions that could indicate fraud.
- **Example: Network Intrusion Detection**
  - Unsupervised learning helps identify suspicious activities in cybersecurity logs.

4. Association Rule Learning

- **Example: Market Basket Analysis**
  - Retailers use **Apriori Algorithm** to find patterns like "people who buy bread often buy butter."
- **Example: Recommendation Systems**
  - Streaming platforms like Netflix use **unsupervised learning** to suggest similar shows/movies based on viewing history.

Association Rules

Association Rule Mining is a part of the data mining process which is performed to find the interesting association rules from a large sample of datasets.

An association rule is a pattern/relationship which portrays the frequent association of two itemsets.

It is represented as: A => B (where, A, B are itemsets (one/more))

Which means that if a customer bought the items of (A) set, then the customer is likely to also buy the items of (B) set.

This Mining is performed on the Transactional databases of large no.of E.com platforms to increase the sales and maximize profits.

Ex: if a customer buys milk, then he is more likely to buy bread or biscuits.

I.e., {milk} => {bread, biscuits}

Ex: if a customer buys a toothbrush and toothpaste, then he is more likely

to also buy the mouth-washer.

I.e, {toothbrush, toothpaste} => {mouth-washer}

Hence, the process of analyzing the large transactional datasets and finding the interesting association rules, is known as Association Rule mining.

A transactional dataset is a dataset containing information about the previously performed transactions (items bought) by the customers.

It only contains the transaction_ID column and itemsets column.

Mining Frequent Patterns:

Frequent Patterns are nothing but the interesting patterns or associations which occur frequently/redundantly in the given dataset.

That is, the patterns that appear frequently in the given large sample of dataset, are known as frequent patterns.

A frequent pattern can be a frequent dataitem, frequent sequence (sequence of data items) or frequent substructures (such as, graph, tree, etc).

Ex: ‘Milk’ dataitem occurs frequently.

Ex: {Milk, Bread, Biscuits} occurs frequently.

Market Basket Analysis (MBA) is a technique which aims for frequent patterns mining. It analyses the customers' buying habits by finding associations between different items that customers place in their shopping baskets.

Frequent patterns are represented by association rules.

Association Rule is represented as: A => B (where, A,B are items sets)

Here, set(A) (LHS) is called Antecedent, and set(B) (RHS) is called Consequent.

The metrics used in association mining for measuring interestingness of patterns are: support and confidence.

-> Support: The proportion of transactions that contain both the antecedent(A) and the consequent(B).

I.e., How frequently the determined association rule is applicable/appears in the given dataset.

Support(A=>B) =

For a single data item or an itemset:

Support(A) =

-> Confidence: It measures the strength/reliability of the association rule. The frequent occurrence of (B) itemset containing the items of (A). (or) The Support of the rule itself (A=>B) (antecedent and consequent (AUB) over the support of antecedent (A).

Confidence(A=>B) =

Mining Methods:

The are two methods (algorithms) for mining the interesting association rules from the given large sample of transaction dataset, viz.:

(1) Apriori Algorithm

(2) FP Growth Algorithm

The aim of both these algorithms is determining the interesting frequent itemsets by finding strong association rules.

In these algorithms we use minimum support (min_sup) and minimum confidence (min_con) to eliminate weak association rules and have only strong association rules (known as threshold values).

I.e., if only some of the customers bought a set of things, doesn't mean that we must consider it as a new pattern. We only consider those patterns which occur most frequently (I.e., the majority of the customers must buy the set of things). Hence, to eliminate those weak patterns (non-frequent), we set a minimum-limit for the value of support and confidence.

Hence, the strong association rules are those rules whose support and confidence values are greater than or equal to the minimum support and confidence values, respectively.

(1) Apriori Algorithm:

The two primary drawbacks of the Apriori Algorithm are:

At each step, candidate sets have to be built.

To build the candidate sets, the algorithm has to repeatedly scan the database.

These two properties inevitably make the algorithm slower.

To overcome these redundant steps, a new association-rule mining algorithm was developed named Frequent Pattern Growth Algorithm.

It overcomes the disadvantages of the Apriori algorithm by storing all the transactions in a Trie Data Structure.

Ex: (notes)

Cluster Analysis

- Clustering is the process of grouping the similar data points in the large dataset.
- Grouping is performed based on the similar characteristics.
- The final group of similar data points formed at the end is known as a cluster.
- The goal of clustering is to maximize the similarity within clusters and minimize the similarity between clusters.
- These clusters are then labeled and fed to machine learning models as training data.
- Hence, the process of analyzing a dataset to form clusters, is known as cluster analysis.
- And the algorithms used for cluster analysis are usually unsupervised algorithms.(non labeled data is provided)

Data Structures used in Cluster Analysis

- The working of clustering algorithms is simple:
    + It takes a matrix which represents the dissimilarities b/w every pair of data points (known as dissimilarity matrix) and a data matrix which represents the data points with its features, as the inputs, and groups the pairs with less dissimilarity values to form a cluster.
    + The dissimilarity matrix is built from a data matrix using a proximity measure(distance metric).
    + Hence, Data Matrix and Dissimilarity Matrix are considered as the data structures used for clustering.

1) Data Matrix:
    - It is a two dimensional matrix, whose rows represent the data points, and the columns represent a variable/feature/attribute of a data point.

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \cdots & x_{Np} \end{bmatrix}$$

2) Dissimilarity Matrix:

+ It is a square matrix in which each element d(i, j) represents the dissimilarity/distance b/w data points (i) and (j) from the data matrix.
+ The diagonal elements d(i, i) or d(j, j) are 0, because it represents the same data point.
+ A dissimilarity value d(i, j) is calculated using a proximity measure(distance metric), such as, Euclidean Distance, Manhattan Distance, etc.
+ The Dissimilarity matrix formed at the end, and the data matrix, are fed to the clustering algorithm as input, which then forms clusters with the pairs having less dissimilarity values.

- The proximity measure(metric/formula to calculate the dissimilarity b/w two data points), is not the same for every data point. It depends on the type of attributes of a data point.
- Based on the type of value that can be stored in a column of a data matrix, the attribute types are classified into (4), they are:

1. Nominal Attributes
2. Binary Attributes
3. Ordinal Attributes
4. Numeric Attributes

1) Nominal Attributes:

- The attributes whose values represent names/categories/states, without any specific order/ranking, are nominal attributes.

Ex:

<u>Attribute</u>	<u>Values</u>
Colors	Black, Brown, Yellow
Birds	Peacock, Pigeon, Crow

- Proximity measure for nominal attributes is:

$$d(i, j) = \frac{p - m}{p},$$

2) Binary Attributes:

- The Attributes which have only two possible values/states, are binary attributes.
- These are of two types:
- Symmetric Attribute: When those two values are equally important.

Ex:

Attribute	Values
Gender	Male, Female

- Asymmetric Attribute: When those two values are not equally important(i.e., opposite values).

Ex:

Attribute	Values
Cancer	Yes, No
Result	pass, fail
Is employee	true, false

- Proximity Measure for Symmetric attributes is:

$$d(i, j) = \frac{r+s}{q+r+s+t}.$$

- Proximity Measure for Asymmetric attributes is:

$$d(i, j) = \frac{r+s}{q+r+s}.$$

3) Ordinal Attributes:
- Attributes whose values have a meaningful ranking/order/sequence, but the difference/intervals b/w them are not consistent/meaningful.

Ex:

Attributes	Values
Grade	A, B, C, D
Qualification	high school, inter, degree
Ratings	bad, good, satisfactory, best
Temp levels	low, medium, high

- Proximity measure, for ordinal attributes is:

$$z_{if} = \frac{r_{if} - 1}{M_f - 1}.$$

4) Numeric Attributes:

- The attributes which can hold numeric data (numbers), are numeric attributes.
- The values can be discrete or continuous.

Ex:

Attributes	Values
Brightness	25, 50, 30
Saturation	5.4, 26.7, 8.1

- Since the values are of numeric type, the dissimilarity value is termed as “distance”.
- Proximity measure for Numeric attributes are many, two of them are:

=>Euclidean Distance:

$$d(i,j) = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \cdots + (x_{ip} - x_{jp})^2}.$$

=>Manhattan Distance:

$$d(i,j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \cdots + |x_{ip} - x_{jp}|.$$

Clustering Methods:

- Partitioning Method
- Hierarchical Method
- Density-based Method
- Grid-Based Method

1) Partitioning Method:

- The clustering method which forms clusters by initially assuming the partitioning of a given dataset into a specified no.of clusters and then iteratively assigning the data points to the nearest cluster (cluster from which the distance is minimum).

I.e, Initially, hypothetical random partitions/clusters are formed, and then the distances b/w every data point and every assumed cluster is calculated. Then, the data point which has the minimum distance to an assumed cluster is the new and actual assigned cluster for that data point. This goes on iteratively until a certain condition is met.

- For every iteration, the distance b/w data points and the cluster which the data point is supposed to be, keeps on decreasing.
- The major benefit of these methods is that they form non-overlapping clusters.
- There are many Algorithms for partitioning methods. Most used one is K-Means Algorithm.

K-Means Clustering Algorithm:

- A Clustering algorithm which follows the Partitioning method with the Euclidean Distance to create non-overlapping clusters from the given dataset.
- This Algorithm works by considering the centroid points of clusters.
- For every iteration the centroid keeps on changing and refining. And the algorithm terminates when centroids of previous iteration and current iteration are equal.

I.e, iteratively assigns the clusters to every datapoint, based on the distance from the centroids of clusters.

- Steps:
  - Decide the no.of clusters to be created (k).
  - Initialize any of (k) no.of data points as the centroids of (k) clusters.
  - Assign each data point to the cluster with the nearest centroid. (use Euclidean Distance formula)
  - Recalculate the centroids based on the mean of the points in each cluster.
  - Repeat the steps (3-4) until the convergence of centroids.(centroids no longer change in iteration / previous iteration centroids and current iteration centroids are equal.)
- Hence, for every iteration, the centroid keeps on changing by averaging, and the distance b/w every data point and its suitable cluster keeps on decreasing.

Ex: (notes)

K-Medoids Clustering

K-Medoids is a **clustering algorithm** similar to **K-Means**, but it is **more robust to noise and outliers** because it chooses actual data points (medoids) as cluster centers instead of computing centroids.

- Steps:

1. Initialize: Select k random data points as the initial medoids (representative cluster centers).
2. Assign Data Points: Assign each data point to the nearest medoid based on a distance metric (e.g., Euclidean distance).
3. Update Medoids:

    - For each cluster, select a new medoid that minimizes the sum of distances between itself and other points in the cluster.
4. Repeat steps 2-3 until the medoids no longer change or a stopping criterion is met.

2) Hierarchical method:

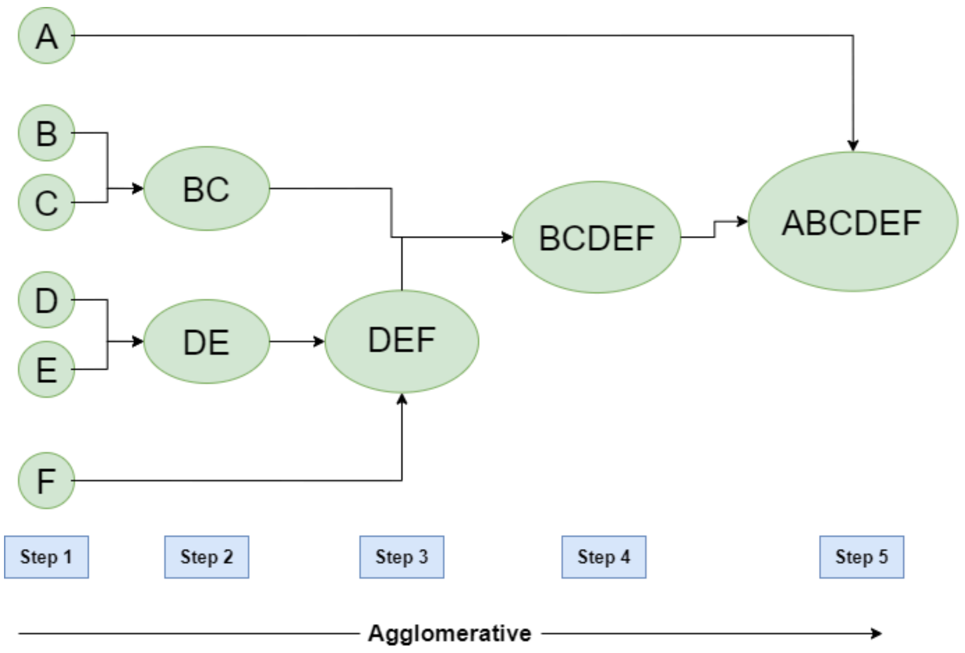
This method of clustering builds a hierarchy of clusters.

Clustering is performed by merging/dividing clusters and forming a tree-like structure, known as a dendrogram.

There are two approaches:

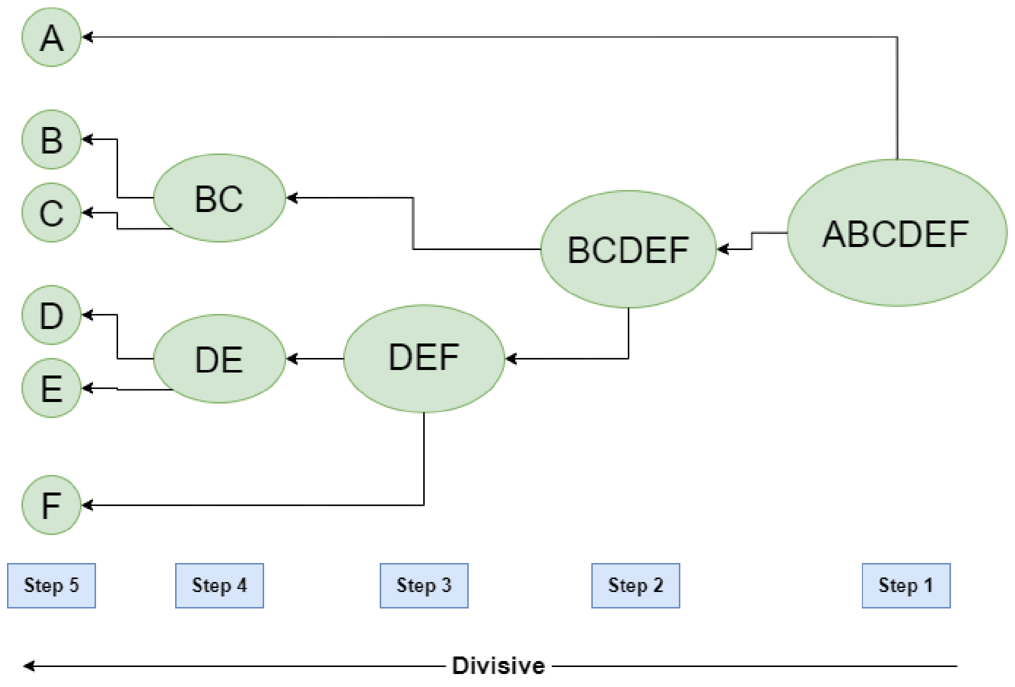
(a) Agglomerative (bottom-up)(merging):

- It starts with each data point as a single cluster and then merges the closest cluster/cluster with similar characteristics iteratively until only one cluster remains.
- Hence, this forms a single cluster for a given dataset.
- The root of the tree is the final cluster.



(b) Divisive (top-down)(splitting):

- It starts with all data points in a single cluster(dataset as a cluster) and then splits the cluster recursively until each data point is in its own cluster.
- Hence, this forms multiple separate clusters for a given data set.
- The terminal/leaf nodes of the tree are the final clusters.



Random Forests

Random Forests is a powerful and versatile ensemble learning algorithm used for both **classification** and **regression** tasks. It builds multiple decision trees and combines their results to produce more accurate and stable predictions.

How Random Forests Work

1. **Bootstrap Sampling (Bagging)**:
    - From the training data, multiple subsets are randomly selected **with replacement** (i.e., some data points may be repeated in each subset).
2. **Building Multiple Decision Trees**:
    - For each subset, a **decision tree** is trained. The trees are allowed to grow deep, making them complex and able to capture more detailed patterns.
3. **Random Feature Selection**:
    - When splitting a node in the decision tree, instead of considering all features, **a random subset of features** is chosen. This helps in making trees less correlated with each other.
4. **Voting/Averaging**:
    - **For classification**: Each tree casts a "vote" for a class, and the class with the most votes is chosen as the final prediction.
    - **For regression**: The average of all tree predictions is taken as the final prediction.

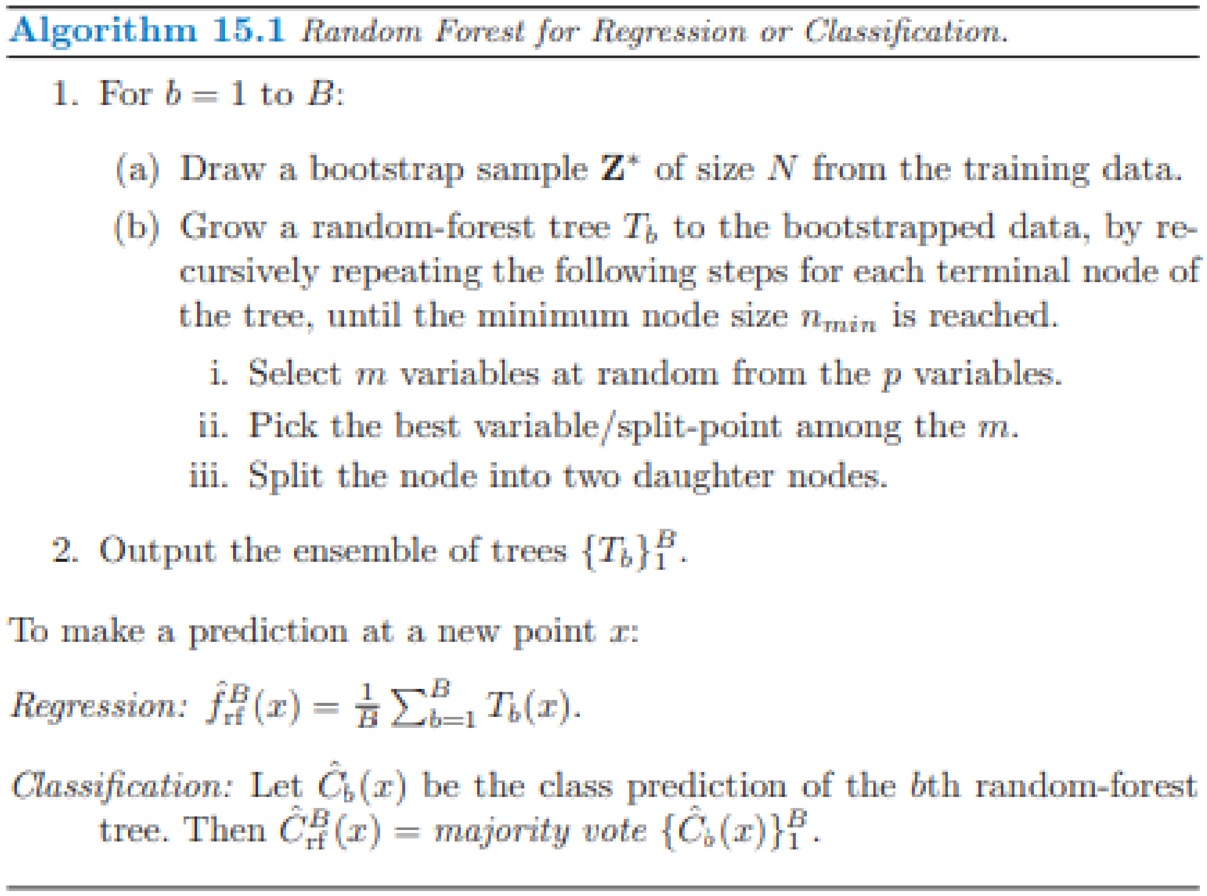
Key Features of Random Forests

- **Ensemble Learning**: Combines multiple models (trees) to improve accuracy and reduce overfitting.

- **Bagging**: Reduces variance by averaging predictions across many trees.
- **Robustness**: Performs well even with missing or noisy data.
- **Non-Linear**: Can handle non-linear relationships between features.
- **Out-of-Bag Error**: A built-in method for estimating the model's error by using data points not included in the bootstrap samples.

Advantages of Random Forests

1. **High Accuracy**: Often outperforms individual decision trees or other classifiers.
2. **Handles Overfitting**: By averaging multiple trees, it reduces the likelihood of overfitting.
3. **Works Well with Unstructured Data**: Can handle both structured (tabular) and unstructured data (images, text).
4. **Feature Importance**: Provides insight into which features are most important in making predictions.



Algorithm 15.1 *Random Forest for Regression or Classification.*

```
1. For $b = 1$ to $B$:
   (a) Draw a bootstrap sample $\mathbf{Z}^*$ of size $N$ from the training data.
   (b) Grow a random-forest tree $T_b$ to the bootstrapped data, by re-
       cursively repeating the following steps for each terminal node of
       the tree, until the minimum node size $n_{min}$ is reached.
         i. Select $m$ variables at random from the $p$ variables.
        ii. Pick the best variable/split-point among the $m$.
       iii. Split the node into two daughter nodes.
2. Output the ensemble of trees $\{T_b\}_1^B$.
```

To make a prediction at a new point x:

Regression: $\hat{f}_{\rm rf}^B(x) = \frac{1}{B}\sum_{b=1}^B T_b(x)$.

Classification: Let $\hat{C}_b(x)$ be the class prediction of the bth random-forest tree. Then $\hat{C}_{\rm rf}^B(x) = \textit{majority vote}\ \{\hat{C}_b(x)\}_1^B$.