Unit - II

Data Mining Query Languages(DMQL)

In data mining, a specialized Data Mining Query Language (DMQL) is developed to facilitate the manipulation, querying, and analysis of data. It plays a pivotal role in uncovering hidden relationships, trends, and patterns within large datasets.

Unlike traditional SQL, which is primarily designed for managing relational databases, DMQL is specifically tailored for data mining tasks. This specialization makes it significantly easier to perform complex data mining operations and integrate with databases and data mining tools seamlessly.

DMQL empowers analysts and data scientists by allowing them to define precise actions for their data analysis tasks. Its capability to extract valuable insights and knowledge from vast datasets makes it an indispensable tool in the field of data mining.

Importance of Data Mining Query Language

The significance of the Data Mining Query Language (DMQL) lies in its ability to streamline and improve the data mining process. Here are the key reasons why DMQL is essential in the field of data mining:

Importance of Data Mining Query Language

The Data Mining Query Language (DMQL) is vital for optimizing the data mining process, offering several advantages:

1. **Data Access and Retrieval**
   DMQL provides a structured and efficient approach to accessing and retrieving data from large, complex datasets. Since data mining tasks require working with substantial and intricate data sources, effective data access and retrieval are essential.
2. **Data Manipulation**
   DMQL enables users to preprocess, clean, and modify data before applying data mining algorithms. This step is crucial for preparing datasets for meaningful analysis and ensuring accurate results.

3. **Flexible Querying**
   DMQL supports flexible querying, allowing data scientists and analysts to tailor queries to meet specific requirements. This adaptability is critical for exploring and refining datasets to uncover valuable insights.
4. **Efficient Data Analysis**
   By using DMQL, users can efficiently analyze large datasets to extract relevant patterns and insights. It streamlines tasks such as computation, summarization, and analysis, making the process more effective.
5. **Automation**
   DMQL facilitates the automation of data extraction and transformation, reducing manual effort and minimizing errors. This improves the efficiency of repetitive data mining tasks, saving time and resources.
6. **Decision Support**
   DMQL plays a crucial role in decision support systems by enabling the extraction and analysis of valuable information from massive datasets. This helps organizations make informed decisions based on data-driven insights.
7. **Knowledge Discovery**
   DMQL is a key component in knowledge discovery processes, empowering analysts to uncover new patterns, trends, and insights from large datasets. It aids in identifying previously hidden relationships within the data.

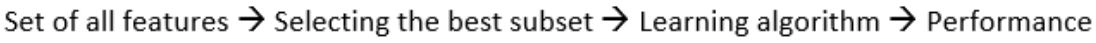
Feature selection

In data science many times we encounter vast of features present in a dataset. But it is not necessary all features contribute equally in prediction that's where feature selection comes. It involves selecting a subset of relevant features from the original feature set to reduce the feature space while improving the model’s performance by reducing computational power. When a dataset has too many features, some may be irrelevant or add noise which can slow down training process and reduce accuracy but it helps in building simpler, faster and more accurate models and also helps in reducing overfitting.

1. Filter Methods

Filter methods evaluate each feature independently with target variable. Feature with high correlation with target variable are selected as it means this feature has some relation and can

help us in making predictions. These methods are used in the preprocessing phase to remove irrelevant or redundant features based on statistical tests (correlation) or other criteria.



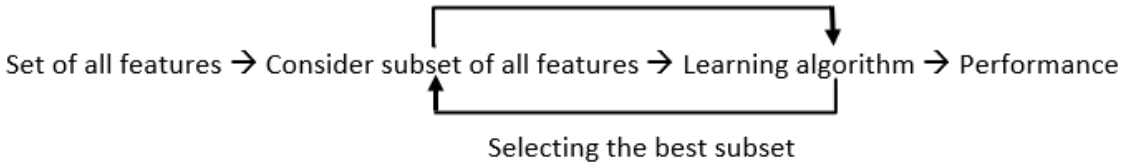
Filter Methods Implementation

Advantages:

- Quickly evaluate features without training the model.
- Good for removing redundant or correlated features**.**

2. Wrapper methods

Wrapper methods are also referred as greedy algorithms that train algorithm. They use different combination of features and compute relation between these subset features and target variable and based on conclusion addition and removal of features are done. Stopping criteria for selecting the best subset are usually pre-defined by the person training the model such as when the performance of the model decreases or a specific number of features are achieved.



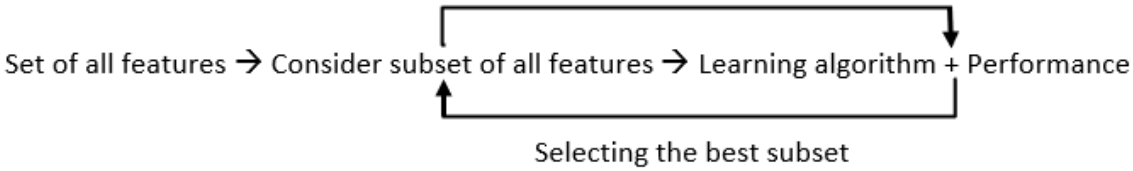
Wrapper Methods Implementation

Advantages:

- Can lead to better model performance since they evaluate feature subsets in the context of the model.
- They can capture feature dependencies and interactions.

3. Embedded methods

Embedded methods perform feature selection during the model training process. They combine the benefits of both filter and wrapper methods. Feature selection is integrated into the model training allowing the model to select the most relevant features based on the training process dynamically.



Embedded Methods Implementation

Advantages:

- More efficient than wrapper methods because the feature selection process is embedded within model training.
- Often more scalable than wrapper methods.

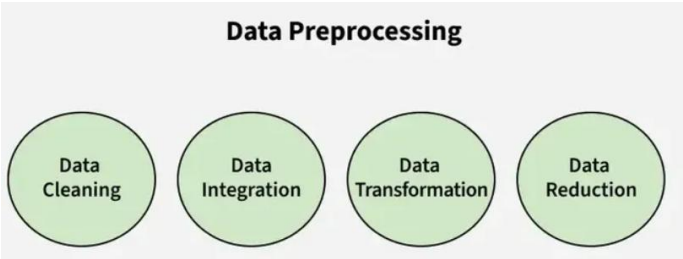
Data pre-processing

Data preprocessing is the process of preparing raw data for analysis by cleaning and transforming it into a usable format. In data mining it refers to preparing raw data for mining by performing tasks like cleaning, transforming, and organizing it into a format suitable for mining algorithms.

- Goal is to improve the quality of the data.
- Helps in handling missing values, removing duplicates, and normalizing data.
- Ensures the accuracy and consistency of the dataset.

Steps in Data Preprocessing

Some key steps in data preprocessing are Data Cleaning, Data Integration, Data Transformation, and Data Reduction.



1. Data Cleaning: It is the process of identifying and correcting errors or inconsistencies in the dataset. It involves handling missing values, removing duplicates, and correcting incorrect or outlier data to ensure the dataset is accurate and reliable. Clean data is essential for effective analysis, as it improves the quality of results and enhances the performance of data models.

- **Missing Values:** This occur when data is absent from a dataset. You can either ignore the rows with missing data or fill the gaps manually, with the attribute mean, or by using the most probable value. This ensures the dataset remains accurate and complete for analysis.
- **Noisy Data:** It refers to irrelevant or incorrect data that is difficult for machines to interpret, often caused by errors in data collection or entry. It can be handled in several ways:
    - **Binning Method:** The data is sorted into equal segments, and each segment is smoothed by replacing values with the mean or boundary values.
    - **Regression:** Data can be smoothed by fitting it to a regression function, either linear or multiple, to predict values.
    - **Clustering:** This method groups similar data points together, with outliers either being undetected or falling outside the clusters. These techniques help remove noise and improve data quality.
- **Removing Duplicates:** It involves identifying and eliminating repeated data entries to ensure accuracy and consistency in the dataset. This process prevents errors and ensures reliable analysis by keeping only unique records.

2. Data Integration: It involves merging data from various sources into a single, unified dataset. It can be challenging due to differences in data formats, structures, and meanings. Techniques like record linkage and data fusion help in combining data efficiently, ensuring consistency and accuracy.

- **Record Linkage** is the process of identifying and matching records from different datasets that refer to the same entity, even if they are represented differently. It helps in combining data from various sources by finding corresponding records based on common identifiers or attributes.
- **Data Fusion** involves combining data from multiple sources to create a more comprehensive and accurate dataset. It integrates information that may be inconsistent or incomplete from different sources, ensuring a unified and richer dataset for analysis.

3. Data Transformation: It involves converting data into a format suitable for analysis. Common techniques include normalization, which scales data to a common range; standardization, which adjusts data to have zero mean and unit variance; and discretization, which converts continuous data into discrete categories. These techniques help prepare the data for more accurate analysis.

- **Data Normalization**: The process of scaling data to a common range to ensure consistency across variables.
- **Discretization**: Converting continuous data into discrete categories for easier analysis.

- **Data Aggregation**: Combining multiple data points into a summary form, such as averages or totals, to simplify analysis.
- **Concept Hierarchy Generation**: Organizing data into a hierarchy of concepts to provide a higher-level view for better understanding and analysis.

4. Data Reduction: It reduces the dataset's size while maintaining key information. This can be done through feature selection, which chooses the most relevant features, and feature extraction, which transforms the data into a lower-dimensional space while preserving important details. It uses various reduction techniques such as,

- **Dimensionality Reduction (e.g., Principal Component Analysis)**: A technique that reduces the number of variables in a dataset while retaining its essential information.
- **Numerosity Reduction**: Reducing the number of data points by methods like sampling to simplify the dataset without losing critical patterns.
- **Data Compression**: Reducing the size of data by encoding it in a more compact form, making it easier to store and process.

Uses of Data Preprocessing

Data preprocessing is utilized across various fields to ensure that raw data is transformed into a usable format for analysis and decision-making. Here are some key areas where data preprocessing is applied:

1. **Data Warehousing**: In data warehousing, preprocessing is essential for cleaning, integrating, and structuring data before it is stored in a centralized repository. This ensures the data is consistent and reliable for future queries and reporting.

2. **Data Mining**: Data preprocessing in data mining involves cleaning and transforming raw data to make it suitable for analysis. This step is crucial for identifying patterns and extracting insights from large datasets.

3. **Machine Learning**: In machine learning, preprocessing prepares raw data for model training. This includes handling missing values, normalizing features, encoding categorical variables, and splitting datasets into training and testing sets to improve model performance and accuracy.

4. **Data Science**: Data preprocessing is a fundamental step in data science projects, ensuring that the data used for analysis or building predictive models is clean, structured, and relevant. It enhances the overall quality of insights derived from the data.

5. **Web Mining**: In web mining, preprocessing helps analyze web usage logs to extract meaningful user behavior patterns. This can inform marketing strategies and improve user experience through personalized recommendations.

6. **Business Intelligence (BI):** Preprocessing supports BI by organizing and cleaning data to create dashboards and reports that provide actionable insights for decision-makers.

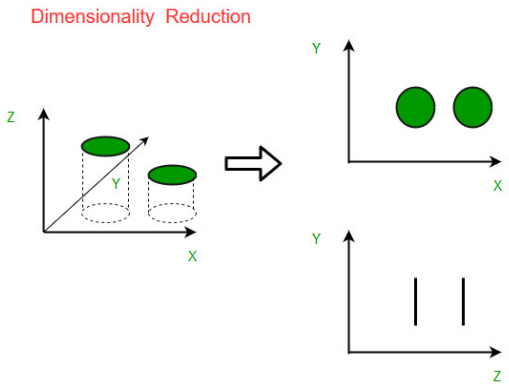
7. **Deep Learning Purpose**: Similar to machine learning, deep learning applications require preprocessing to normalize or enhance features of the input data, optimizing model training processes.

Dimensionality reduction

Dimensionality reduction is the process of reducing the number of input variables (features) in a data set while preserving as much important information as possible.

How Dimensionality Reduction Works?

Lets understand how dimensionality Reduction is used with the help of example. Imagine a dataset where each data point exists in a 3D space defined by axes X, Y and Z. If most of the data variance occurs along X and Y then the Z-dimension may contribute very little to understanding the structure of the data.



- Before Reduction You can see that Data exist in 3D (X,Y,Z). It has high redundancy and Z contributes little meaningful information
- On the right after reducing the dimensionality the data is represented in **lower-dimensional spaces**. The top plot (X-Y) maintains the meaningful structure while the bottom plot (Z-Y) shows that the Z-dimension contributed little useful information.

Dimensionality Reduction Techniques

Dimensionality reduction techniques can be broadly divided into two categories:

1. Feature Selection

Feature selection chooses the most relevant features from the dataset without altering them. It helps remove redundant or irrelevant features, improving model efficiency. Some common methods are:

- **Filter methods** rank the features based on their relevance to the target variable.
- **Wrapper methods** use the model performance as the criteria for selecting features.
- **Embedded methods** combine feature selection with the model training process.

2. Feature Extraction

Feature extraction involves creating new features by combining or transforming the original features. These new features retain most of the dataset's important information in fewer dimensions. Common feature extraction methods are:

1. **Principal Component Analysis (PCA):** Converts correlated variables into uncorrelated 'principal components, reducing dimensionality while maintaining as much variance as possible enabling more efficient analysis.
2. **Missing Value Ratio:** Variables with missing data beyond a set threshold are removed, improving dataset reliability.
3. **Backward Feature Elimination**: Starts with all features and removes the least significant ones in each iteration. The process continues until only the most impactful features remain, optimizing model performance.
4. **Forward Feature Selection:** Forward Feature Selection Begins with one feature, adds others incrementally and keeps those improving model performance.
5. **Random Forest**: Random forest Uses decision trees to evaluate feature importance, automatically selecting the most relevant features without the need for manual coding, enhancing model accuracy.
6. **Factor Analysis**: Groups variables by correlation and keeps the most relevant ones for further analysis.
7. **Independent Component Analysis (ICA)**: Identifies statistically independent components, ideal for applications like 'blind source separation' where traditional correlation-based methods fall short.

Advantages of Dimensionality Reduction

As seen earlier high dimensionality makes models inefficient. Let's now summarize the key advantages of reducing dimensionality.

- **Faster Computation**: With fewer features machine learning algorithms can process data more quickly. This results in faster model training and testing which is particularly useful when working with large datasets.

- **Better Visualization**: As we saw in the earlier figure reducing dimensions makes it easier to visualize data and reveal hidden patterns.

- **Prevent Overfitting**: With few features models are less likely to memorize the training data and overfit. This helps the model generalize better to new, unseen data improve its ability to make accurate predictions.

Disadvantages of Dimensionality Reduction

- **Data Loss & Reduced Accuracy:** Some important information may be lost during dimensionality reduction and affect model performance.

- **Choosing the Right Components:** Deciding how many dimensions to keep is difficult as keeping too few may lose valuable information while keeping too many can led to overfitting.

Unit - III

1. Concept Description:

The simplest kind of descriptive data mining is concept description. A concept usually refers to a collection of data such as frequent_buyers, graduate_students, and so on. As a data mining task, concept description is not a simple enumeration of the data. Instead, concept description generates descriptions for characterization and comparison of the data. It is sometimes called class description, when the concept to be described refers to a class of objects

Purpose:

It goes beyond simple enumeration of data points. It aims to generate concise and informative descriptions that capture the core characteristics of the concept.

2. Characterization: Characterization provides a concise summary of the general characteristics of a target class of data. It answers questions like, "What are the defining features of this group?"

- **Example:**

If the concept is "frequent buyers," characterization might involve summarizing their average purchase frequency, preferred product categories, or typical spending patterns.

- **Methods:**

Data characterization can utilize various techniques like data generalization, attribute-oriented induction, and analytical characterization.

3. Comparison (or Discrimination): Comparison, also known as discrimination, focuses on highlighting the differences between two or more sets of data or concepts.

- **Example:**

Comparing "frequent buyers" with "occasional buyers" might reveal differences in their demographics, purchase behavior, or preferred products.

- **Methods:**

Comparison techniques can involve analyzing attribute relevance, applying different generalization levels, or using descriptive statistical measures.

Data Generalization

Data generalization is a process that abstracts a large set of task-relevant data in a database from a relatively low conceptual level to higher conceptual levels. Methods for the efficient and flexible generalization of large data sets can be categorized according to two approaches :(1) the data cube (or OLAP) approach and (2) the attribute –oriented induction approach. In this section, we describe the attribute-oriented induction approach.

Attribute-Oriented Induction

The attribute oriented induction approach to data generalization and summarization based characterization was first proposed in 1989, a few years prior to the introduction of the data cube approach. The data cube approach can be considered as a data warehouse-based pre-computation oriented materialized-view approach. It performs off-line aggregation before an OLAP or data mining query is submitted for processing. On the other hand, the attribute-oriented induction approach, atleast in its initial proposal, is a relational database query-oriented, generalization-based, on-line data analysis technique. However, there is no inherent barrier distinguishing the two approaches based on on-line aggregation versus off-line pre computation. Some aggregations in the data cube can be computed on-line, while off-line pre computation of multidimensional space can speed of attribute-oriented induction as well.

AOI for Data Characterization

Data Characterization is a **descriptive data mining task** that summarizes the general features of a target class of data. It provides a high-level overview or profile of the data, usually in the form of **summaries, statistical measures, or visualizations**.

Target Class Identification:

- Defines the specific subset of data to be characterized (e.g., "customers who bought product X").
- Usually specified through **query conditions** or data labels.

Data Collection & Selection:

- Involves retrieving relevant data from a data warehouse or database.
- Uses OLAP (Online Analytical Processing) operations like **roll-up**, **drill-down**, **slice**, and **dice**.

Attribute Relevance Analysis:

- Selecting **important attributes** (features) that contribute to the meaningful characterization.
- Uses techniques like correlation, entropy, or attribute ranking.

Summary Statistics Computation:

- Descriptive measures: **mean, median, mode, min, max, standard deviation, frequency**.
- Aggregation operations to generalize data over dimensions.

Visualization & Presentation:

- Results are often presented via:
  - **Tables** (containing statistical summaries)
  - **Graphs/Charts** (e.g., histograms, boxplots)
  - **OLAP cubes** or **multi-dimensional views**

Comparison of Classes (Optional):

- Sometimes combined with **discrimination** to compare two or more classes (e.g., "high-value vs low-value customers").

Efficient Implementation of AOI

In data mining, **AOI (Area of Interest)** refers to the **specific focus area or scope** within a dataset for analysis. In the context of **data characterization**, implementing AOI efficiently is critical for reducing computation time, improving accuracy, and generating meaningful summaries.

1. Data Preprocessing

- **Cleaning**: Remove noise, duplicates, and missing values.
- **Integration**: Combine data from multiple sources.
- **Transformation**: Normalize or discretize data.
- **Reduction**: Use dimensionality reduction techniques (e.g., PCA) to focus on key features.

2. Attribute-Oriented Induction (AOI) Technique

- Generalizes data using **concept hierarchies**.
- Replaces low-level values with higher-level concepts.
- Steps:
  - Select target class.
  - Remove irrelevant attributes.

    - Apply concept generalization.
    - Aggregate and summarize.

Benefits:

- Reduces volume of data.
- Speeds up computation.
- Improves clarity of patterns.

3. OLAP Integration

- Use **OLAP operations** (Online Analytical Processing) such as:
    - **Roll-up** (data abstraction)
    - **Drill-down** (detailed view)
    - **Slice** (select one dimension)
    - **Dice** (select a sub-cube)
- Enables **interactive and multi-dimensional exploration** of AOI.

4. Efficient Query Mechanisms

- Use **SQL-based query languages** to define and extract AOI.
- Apply **indexing** and **partitioning** on data warehouses to speed up access.
- Use **materialized views** for precomputed summaries.

5. Parallel and Distributed Processing

- Leverage **parallel computing** for large datasets.
- Use frameworks like **Hadoop, Spark** to handle AOI operations efficiently at scale.

6. Automated Feature Selection

- Apply machine learning techniques to **automatically identify relevant attributes**.
    - Examples: Information gain, Chi-square, ReliefF
- Reduces manual effort and improves AOI relevance.

Mining Frequent Patterns, Associations and Correlations

Basic Concepts:

1. Frequent Patterns

A **frequent pattern** is a set of items, subsequences, or substructures that appear **frequently** in a dataset.

Example:

- In a supermarket: {milk, bread} bought together by many customers.

- In web logs: sequences of pages frequently visited together.

Types:

- **Frequent itemsets**: Sets of items that appear together in transactions.
- **Frequent subsequences**: Patterns that appear in a sequence (e.g., shopping over time).
- **Frequent substructures**: Common structures in graphs (e.g., molecular compounds).

Key Metric:

- **Support**: The proportion of transactions in which the itemset occurs.

Frequent pattern mining finds all patterns whose **support ≥ minimum support threshold**.

2. Association Rules

An **association rule** is an implication of the form: $X \rightarrow Y$

where **X** and **Y** are itemsets, and the rule suggests that **if X occurs, Y is likely to occur** as well.

Key Measures:

- **Support (s)**: Frequency of $X \cup Y$ in the dataset.
- **Confidence (c)**: Likelihood of Y given $X \rightarrow$ confidence = support($X \cup Y$) / support(X)
- **Lift**: Measures how much more likely Y is given X than by chance.
    - lift = confidence / support(Y)

Example:

If 80% of customers who buy diapers also buy beer:

- Rule: {diaper} → {beer}
- Confidence: 80%
- Support: Suppose 30% of all transactions contain both.

3. Correlation Analysis

Purpose:

To **measure the strength and direction of a relationship** between itemsets beyond what is revealed by support and confidence.

Problem with Confidence:

- High confidence does **not always** mean a strong correlation.
- Example: If item Y is very common, it might appear in many transactions **regardless** of X.

Solutions:

- Use statistical **correlation measures** like:
  - **Lift** (as above)
  - **Chi-square**
  - **Conviction**
  - **All-confidence / Kulczynski measure**

Interpretation:

- **Lift > 1**: Positive correlation between X and Y
- **Lift = 1**: X and Y are independent
- **Lift < 1**: Negative correlation

Apriori method

Apriori Algorithm is a basic method used in data analysis to find groups of items that often appear together in large sets of data. It helps to discover useful patterns or rules about how items are related which is particularly valuable in market basket analysis.

he Apriori Algorithm operates through a systematic process that involves several key steps:

1. Identifying Frequent Itemsets

- The Apriori algorithm starts by looking through all the data to count how many times each single item appears. These single items are called 1-itemsets.

- Next it uses a rule called minimum support this is a number that tells us how often an item or group of items needs to appear to be important. If an item appears often enough meaning its count is above this minimum support it is called a frequent itemset.

2. Creating Possible item group

- After finding the single items that appear often enough (frequent 1-item groups) the algorithm combines them to create pairs of items (2-item groups). Then it checks which pairs are frequent by seeing if they appear enough times in the data.

- This process keeps going step by step making groups of 3 items, then 4 items and so on. The algorithm stops when it can't find any bigger groups that happen often enough.

3. Removing Infrequent Item groups

- The Apriori algorithm uses a helpful rule to save time. This rule says: if a group of items does not appear often enough then any larger group that incl2 udes these items will also not appear often.

- Because of this, the algorithm does not check those larger groups. This way it avoids wasting time looking at groups that won't be important make the whole process faster.

4. Generating Association Rules

- The algorithm makes rules to show how items are related.
- It checks these rules using support, confidence and lift to find the strongest ones.

Key Metrics of Apriori Algorithm

- **Support**: This metric measures how frequently an item appears in the dataset relative to the total number of transactions. A higher support indicates a more significant presence of the itemset in the dataset. Support tells us how often a particular item or combination of items appears in all the transactions ("Bread is bought in 20% of all transactions.")
- **Confidence**: Confidence assesses the likelihood that an item Y is purchased when item X is purchased. It provides insight into the strength of the association between two items. Confidence tells us how often items go together. ("If bread is bought, butter is bought 75% of the time.")
- **Lift**: Lift evaluates how much more likely two items are to be purchased together compared to being purchased independently. A lift greater than 1 suggests a strong positive association. Lift shows how strong the connection is between items. ("Bread and butter are much more likely to be bought together than by chance.")

Transaction ID	Items Bought
T1	Bread, Butter, Milk
T2	Bread, Butter
T3	Bread, Milk
T4	Butter, Milk
T5	Bread, Milk

Step 1 : Setting the parameters

- **Minimum Support Threshold:** 50% (item must appear in at least 3/5 transactions). This threshold is formulated from this formula:

Support(A)=Number of transactions containing itemset ATotal number of transactionsSuppor t(A)=Total number of transactionsNumber of transactions containing itemset A

- **Minimum Confidence Threshold:** 70% (You can change the value of parameters as per the use case and problem statement). This threshold is formulated from this formula:

Confidence(X→Y)=Support(X∪Y)Support(X)Confidence($X \rightarrow Y$)=Support(X)Support($X \cup Y$)

Step 2: Find Frequent 1-Itemsets

Lets count how many transactions include each item in the dataset (calculating the frequency of each item).

Item	Support Count	Support %
Bread	4	80%
Butter	3	60%
Milk	4	80%

All items have support% ≥ 50%, so they qualify as frequent 1-itemsets. if any item has support% < 50%, It will be omitted out from the frequent 1- itemsets.

Step 3: Generate Candidate 2-Itemsets

Combine the frequent 1-itemsets into pairs and calculate their support. For this use case we will get 3 item pairs (bread,butter) , (bread,ilk) and (butter,milk) and will calculate the support similar to step 2

Item Pair	Support Count	Support %
Bread, Butter	2	40%
Bread, Milk	3	60%
Butter, Milk	2	40%

Frequent 2-itemsets: {Bread, Milk} meet the 50% threshold but {Butter, Milk} and {Bread ,Butter} doesn't meet the threshold, so will be committed out.

Step 4: Generate Candidate 3-Itemsets

Combine the frequent 2-itemsets into groups of 3 and calculate their support. for the triplet we have only got one case i.e {bread,butter,milk} and we will calculate the support.

Item Triplet	Support Count	Support %
Bread, Butter, Milk	1	40%

Since this does not meet the 50% threshold, there are no frequent 3-itemsets.

Step 5: Generate Association Rules

Now we generate rules from the frequent itemsets and calculate confidence.

Rule 1: If Bread → Butter (if customer buys bread, the customer will buy butter also)

- Support of {Bread, Butter} = 2.
- Support of {Bread} = 4.
- Confidence = 2/4 = 50% (Failed threshold).

Rule 2: If Butter → Bread (if customer buys butter, the customer will buy bread also)

- Support of {Bread, Butter} = 3.
- Support of {Butter} = 3.
- Confidence = 3/3 = 100% (Passes threshold).

Rule 3: If Bread → Milk (if customer buys bread, the customer will buy milk also)

- Support of {Bread, Milk} = 3.
- Support of {Bread} = 4.
- Confidence = 3/4 = 75% (Passes threshold).

Association Rule Mining

Association Mining searches for frequent items in the data set. In frequent mining usually, interesting associations and correlations between item sets in transactional and relational databases are found. In short, Frequent Mining shows which items appear together in a transaction or relationship.

Need of Association Mining: Frequent mining is the generation of association rules from a Transactional Dataset. If there are 2 items X and Y purchased frequently then it's good to put them together in stores or provide some discount offer on one item on purchase of another item. This can really increase sales. For example, it is likely to find that if a customer buys **Milk** and **bread** he/she also buys **Butter**. So the association rule is **['milk]^['bread']=>['butter']**. So the seller can suggest the customer buy butter if he/she buys Milk and Bread.

Important Definitions :

- **Support :** It is one of the measures of interestingness. This tells about the usefulness and certainty of rules. **5% Support** means total 5% of transactions in the database follow the rule.

$$\text{Support}(A \rightarrow B) = \text{Support\_count}(A \cup B)$$

- **Confidence:** A confidence of 60% means that 60% of the customers who purchased a milk and bread also bought butter.

$$\text{Confidence}(A \rightarrow B) = \text{Support\_count}(A \cup B) / \text{Support\_count}(A)$$

If a rule satisfies both minimum support and minimum confidence, it is a strong rule.

- **Support_count(X)**: Number of transactions in which X appears. If X is A **union** B then it is the number of transactions in which A and B both are present.
- **Maximal Itemset:** An itemset is maximal frequent if none of its supersets are frequent.
- **Closed Itemset:** An itemset is closed if none of its immediate supersets have same support count same as Itemset.
- **K- Itemset:** Itemset which contains K items is a K-itemset. So it can be said that an itemset is frequent if the corresponding support count is greater than the minimum support count.

Example On finding Frequent Itemsets - Consider the given dataset with given transactions.

TransactionId	Items
1	{A,C,D}
2	{B,C,D}
3	{A,B,C,D}
4	{B,D}
5	{A,B,C,D}

- Lets say minimum support count is 3
- Relation hold is maximal frequent => closed => frequent

Improving the Efficiency of Apriori Algorithm

The **Apriori algorithm** is a classic method for mining frequent itemsets and association rules.
However, it has **major drawbacks**, mainly due to:

- **Multiple database scans**
- **High candidate generation cost**
- **Wasted computation on infrequent itemsets**

Techniques to Improve Apriori Efficiency

Hash-Based Itemset Counting

- Use **hash tables** to reduce the size of candidate itemsets (especially for 2-itemsets).
- Hash candidate itemsets into **buckets** during the database scan.
- **Infrequent buckets** are pruned early — reduces total candidates.

Benefit: Cuts down the number of itemsets considered.

Transaction Reduction

- After each iteration, **remove transactions** that **do not contain any frequent itemsets**.

- Such transactions will not contribute to the next level of candidate generation.

Benefit: Reduces the dataset size as you go deeper.

Partitioning

- Divide the database into **smaller partitions**.
- Find **locally frequent itemsets** in each partition.
- Only **globally frequent** itemsets are tested in full database.

Benefit: Reduces the number of database scans.

Prune Infrequent Candidates Early

- Use the **Apriori property** aggressively:
  "All subsets of a frequent itemset must also be frequent."
- Before counting support, **eliminate any candidate** with an infrequent subset.

Benefit: Prevents wasteful support counting.

Vertical Data Format (like in ECLAT)

- Store data as item → list of transaction IDs (TID-list).
- Use **set intersections** to compute support rather than scanning transactions.

Benefit: Faster support counting, especially for dense datasets.

Using Sampling

- Run Apriori on a **sample** of the database.
- Use results to estimate likely frequent itemsets.
- Confirm frequent itemsets with a full scan if needed.

Benefit: Faster with minimal loss of accuracy.

Dynamic Itemset Counting (DIC)

- Start counting support for new itemsets **before** the end of the pass.
- Unlike traditional Apriori, it doesn't wait for a full pass to evaluate next level itemsets.

Benefit: Reduces number of full database scans.

Pattern-Growth Approach for mining Frequent Item sets

The FP-Growth (Frequent Pattern Growth) algorithm efficiently mines frequent itemsets from large transactional datasets. Unlike the Apriori algorithm which suffers from high computational cost due to candidate generation and multiple database scans. FP-Growth avoids these inefficiencies by compressing the data into an FP-Tree (Frequent Pattern Tree) and extracts patterns directly from it.

How FP-Growth Works

Here's how it works in simple terms:

1. **Data Compression**: First FP-Growth compresses the dataset into a smaller structure called the **Frequent Pattern Tree (FP-Tree)**. This tree stores information about item sets (collections of items) and their frequencies without need to generate candidate sets like Apriori does.

2. **Mining the Tree**: The algorithm then examines this tree to identify patterns that appear frequently based on a minimum support threshold. It does this by breaking the tree down into smaller "conditional" trees for each item making the process more efficient.

3. **Generating Patterns**: Once the tree is built and analyzed the algorithm generates the frequent patterns (itemsets) and the rules that describe relationships between items.

Working of FP- Growth Algorithm

Lets jump to the usage of FP- Growth Algorithm and how it works with reallife data.
Consider the following data:

Transaction ID	Items
T1	{E,K,M,N,O,Y}
T2	{D,E,K,N,O,Y}
T3	{A,E,K,M}
T4	{K,M,Y}
T5	{C,E,I,K,O,O}

Unit – IV

Basic Concept of Classification

Data Mining: Data mining in general terms means mining or digging deep into data that is in different forms to gain patterns, and to gain knowledge on that pattern. In the process of data mining, large data sets are first sorted, then patterns are identified and relationships are established to perform data analysis and solve problems.

Classification is a task in data mining that involves assigning a class label to each instance in a dataset based on its features. The goal of classification is to build a model that accurately predicts the class labels of new instances based on their features.

There are two main types of classification: binary classification and multi-class classification. Binary classification involves classifying instances into two classes, such as "spam" or "not spam", while multi-class classification involves classifying instances into more than two classes.

The process of building a classification model typically involves the following steps:

Data Collection:
The first step in building a classification model is data collection. In this step, the data relevant to the problem at hand is collected. The data should be representative of the problem and should contain all the necessary attributes and labels needed for classification. The data can be collected from various sources, such as surveys, questionnaires, websites, and databases.

Data Preprocessing:
The second step in building a classification model is data preprocessing. The collected data needs to be preprocessed to ensure its quality. This involves handling missing values, dealing with outliers, and transforming the data into a format suitable for analysis. Data preprocessing also involves converting the data into numerical form, as most classification algorithms require numerical input.

Handling Missing Values: Missing values in the dataset can be handled by replacing them with the mean, median, or mode of the corresponding feature or by removing the entire record.

Dealing with Outliers: Outliers in the dataset can be detected using various statistical techniques such as z-score analysis, boxplots, and scatterplots. Outliers can be removed from the dataset or replaced with the mean, median, or mode of the corresponding feature.

Data Transformation: Data transformation involves scaling or normalizing the data to bring it into a common scale. This is done to ensure that all features have the same level of importance in the analysis.

Feature Selection:

The third step in building a classification model is feature selection. Feature selection involves identifying the most relevant attributes in the dataset for classification. This can be done using various techniques, such as correlation analysis, information gain, and principal component analysis.

Correlation Analysis: Correlation analysis involves identifying the correlation between the features in the dataset. Features that are highly correlated with each other can be removed as they do not provide additional information for classification.

Information Gain: Information gain is a measure of the amount of information that a feature provides for classification. Features with high information gain are selected for classification.

Principal Component Analysis:

Principal Component Analysis (PCA) is a technique used to reduce the dimensionality of the dataset. PCA identifies the most important features in the dataset and removes the redundant ones.

Model Selection:
The fourth step in building a classification model is model selection. Model selection involves selecting the appropriate classification algorithm for the problem at hand. There are several algorithms available, such as decision trees, support vector machines, and neural networks.
Decision Trees: Decision trees are a simple yet powerful classification algorithm. They divide the dataset into smaller subsets based on the values of the features and construct a tree-like model that can be used for classification.

Support Vector Machines: Support Vector Machines (SVMs) are a popular classification algorithm used for both linear and nonlinear classification problems. SVMs are based on the concept of maximum margin, which involves finding the hyperplane that maximizes the distance between the two classes.

Neural Networks:

Neural Networks are a powerful classification algorithm that can learn complex patterns in the data. They are inspired by the structure of the human brain and consist of multiple layers of interconnected nodes.

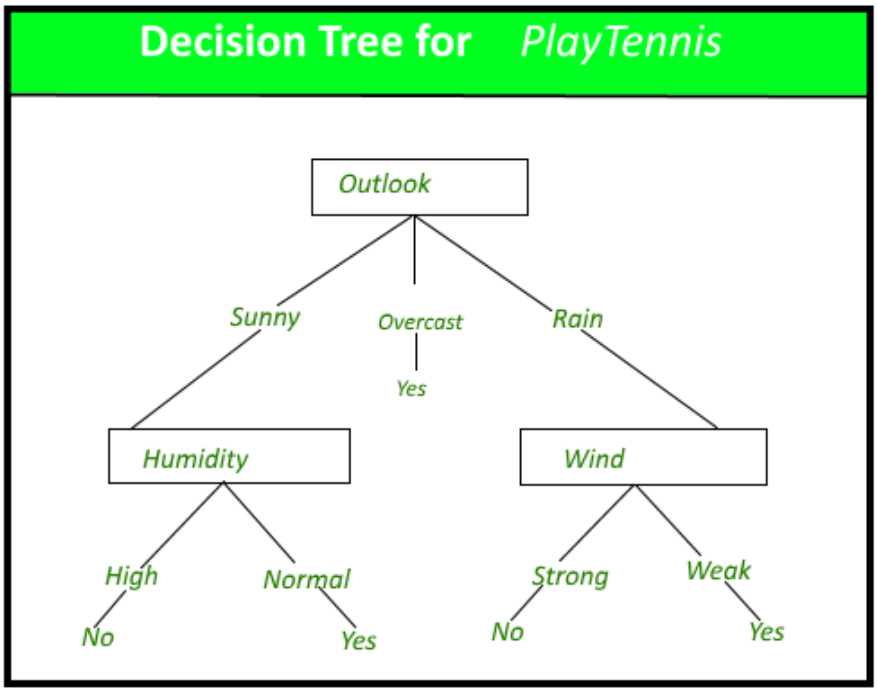
Model Training:
The fifth step in building a classification model is model training. Model training involves using the selected classification algorithm to learn the patterns in the data. The data is divided into a training set and a validation set. The model is trained using the training set, and its performance is evaluated on the validation set.

Model Evaluation:
The sixth step in building a classification model is model evaluation. Model evaluation involves assessing the performance of the trained model on a test set. This is done to ensure that the model generalizes well

Decision Tree Induction

- Decision tree induction is a common technique in data mining that is used to generate a predictive model from a dataset. This technique involves constructing a tree-like structure, where each internal node represents a test on an attribute, each branch represents the outcome of the test, and each leaf node represents a prediction. The goal of decision tree induction is to build a model that can accurately predict the outcome of a given event, based on the values of the attributes in the dataset.

- To build a decision tree, the algorithm first selects the attribute that best splits the data into distinct classes. This is typically done using a measure of impurity, such as entropy or the Gini index, which measures the degree of disorder in the data. The algorithm then repeats this process for each branch of the tree, splitting the data into smaller and smaller subsets until all of the data is classified.
- Decision tree induction is a popular technique in data mining because it is easy to understand and interpret, and it can handle both numerical and categorical data. Additionally, decision trees can handle large amounts of data, and they can be updated with new data as it becomes available. However, decision trees can be prone to overfitting, where the model becomes too complex and does not generalize well to new data. As a result, data scientists often use techniques such as pruning to simplify the tree and improve its performance.



Attribute Selection Measures

Attribute subset Selection is a technique which is used for data reduction in data mining process. Data reduction reduces the size of data so that it can be used for analysis purposes more efficiently.

Need of Attribute Subset Selection

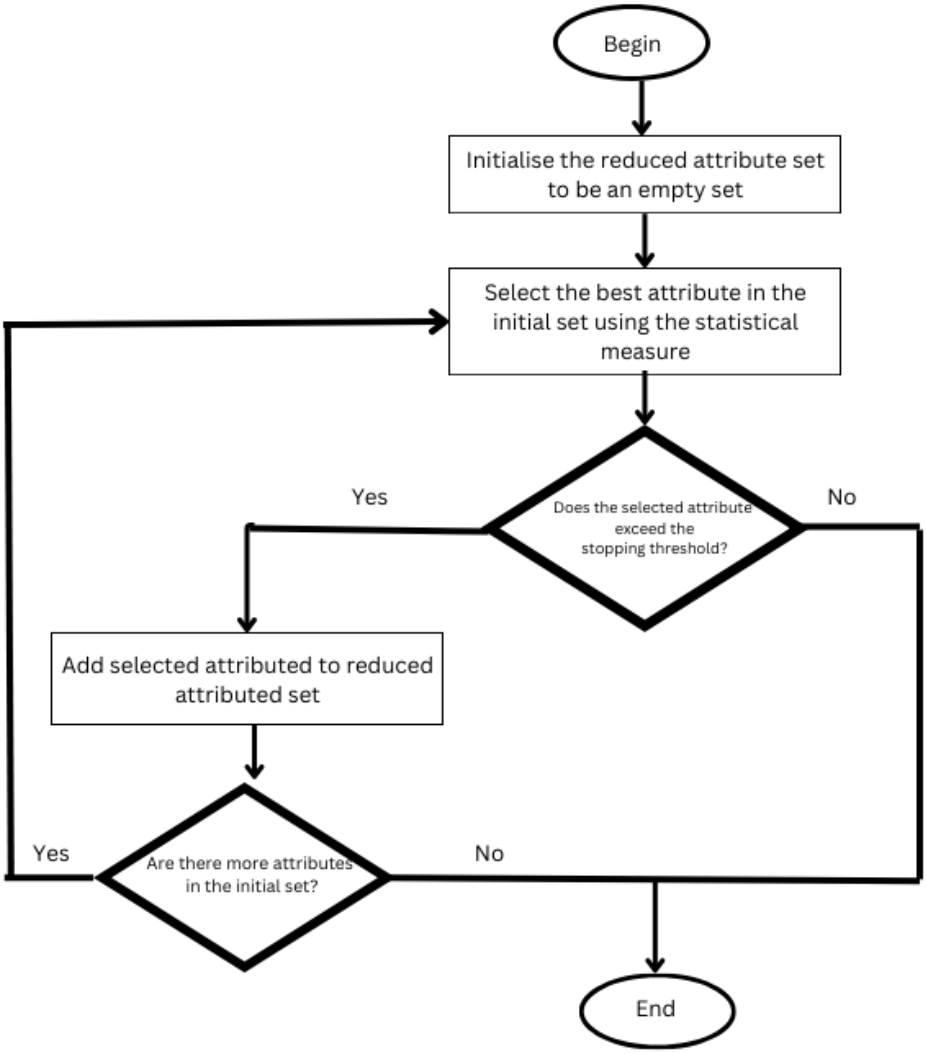
The data set may have a large number of attributes. But some of those attributes can be irrelevant or redundant. The goal of attribute subset selection is to find a minimum set of

attributes such that dropping of those irrelevant attributes does not much affect the utility of data and the cost of data analysis could be reduced. Mining on a reduced data set also makes the discovered pattern easier to understand.

Methods of Attribute Subset Selection

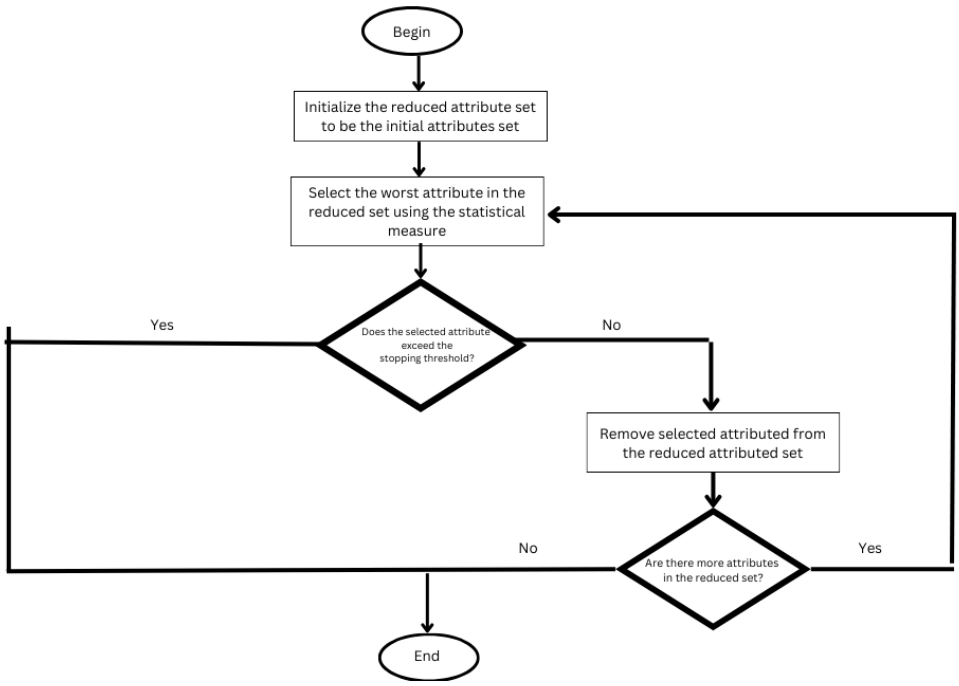
1. Stepwise Forward Selection. 2. Stepwise Backward Elimination. 3. Combination of Forward Selection and Backward Elimination. 4. Decision Tree Induction. All the above methods are greedy approaches for attribute subset selection.

Stepwise Forward Selection: This procedure start with an empty set of attributes as the minimal set. The most relevant attributes are chosen (having minimum p-value) and are added to the minimal set. In each iteration, one attribute is added to a reduced set.



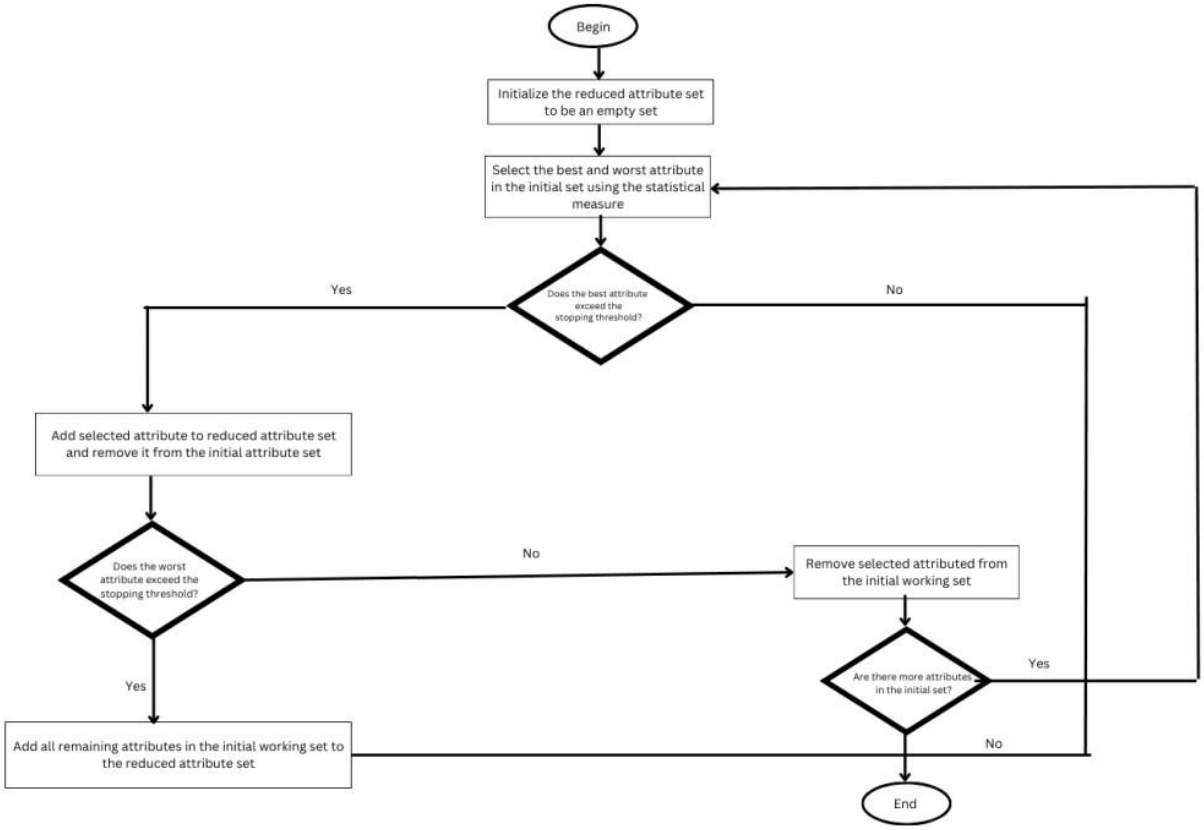
Stepwise Backward Elimination

Here all the attributes are considered in the initial set of attributes. In each iteration, one attribute is eliminated from the set of attributes whose p-value is higher than significance level.



Combination of Forward Selection and Backward Elimination

The stepwise forward selection and backward elimination are combined so as to select the relevant attributes most efficiently. This is the most common technique which is generally used for attribute selection.



Decision Tree Induction

This approach uses decision tree for attribute selection. It constructs a flow chart like structure having nodes denoting a test on an attribute. Each branch corresponds to the outcome of test and leaf nodes is a class prediction. The attribute that is not the part of tree is considered irrelevant and hence discarded.

Tree Pruning

Decision tree pruning is a critical technique in machine learning used to optimize decision tree models by reducing overfitting and improving generalization to new data. **In this guide, we'll explore the importance of decision tree pruning, its types, implementation, and its significance in machine learning model optimization.**

What is Decision Tree Pruning?

Decision tree pruning is a technique used to prevent decision trees from overfitting the training data. Pruning aims to simplify the decision tree by removing parts of it that do not provide significant predictive power, thus improving its ability to generalize to new data.

Decision Tree Pruning removes unwanted nodes from the overfitted decision tree to make it smaller in size which results in more fast, more accurate and more effective predictions.

Types Of Decision Tree Pruning

There are two main types of decision tree pruning: **Pre-Pruning** and **Post-Pruning**.

Pre-Pruning (Early Stopping)

Sometimes, the growth of the decision tree can be stopped before it gets too complex, this is called pre-pruning. It is important to prevent the overfitting of the training data, which results in a poor performance when exposed to new data.

Some common pre-pruning techniques include:

- **Maximum Depth**: It limits the maximum level of depth in a decision tree.
- **Minimum Samples per Leaf**: Set a minimum threshold for the number of samples in each leaf node.
- **Minimum Samples per Split**: Specify the minimal number of samples needed to break up a node.
- **Maximum Features:** Restrict the quantity of features considered for splitting.

Post-Pruning (Reducing Nodes)

After the tree is fully grown, post-pruning involves removing branches or nodes to improve the model's ability to generalize. Some common post-pruning techniques include:

- **Cost-Complexity Pruning (CCP)**: This method assigns a price to each subtree primarily based on its accuracy and complexity, then selects the subtree with the lowest fee.

- **Reduced Error Pruning**: Removes branches that do not significantly affect the overall accuracy.
- **Minimum Impurity Decrease**: Prunes nodes if the decrease in impurity (Gini impurity or entropy) is beneath a certain threshold.
- **Minimum Leaf Size**: Removes leaf nodes with fewer samples than a specified threshold.

Bayes Classification Methods

Bayes' Theorem describes the probability of an event, based on precedent knowledge of conditions which might be related to the event. In other words, Bayes' Theorem is the add-on of Conditional Probability.

With the help of Conditional Probability, one can find out the probability of **X** given **H, and** it is denoted by **P(X | H)**. Now Bayes' Theorem states that if we know Conditional Probability (**P(X | H))** then we can find out **P(H | X)**, given the condition that **P(X)** and **P(H)** are already known to us.

Bayes' Theorem is named after Thomas Bayes. He first makes use of conditional probability to provide an algorithm which uses evidence to calculate limits on an unknown parameter. Bayes' Theorem has two types of probabilities:

1. Prior Probability [P(H)]
2. Posterior Probability [P(H/X)]

Where,

- **X** - X is a data tuple.
- **H** - H is some Hypothesis.

1. Prior Probability

Prior Probability is the probability of occurring an event before the collection of new data. It is the best logical evaluation of the probability of an outcome which is based on the present knowledge of the event before the inspection is performed.

2. Posterior Probability

When new data or information is collected then the Prior Probability of an event will be revised to produce a more accurate measure of a possible outcome. This revised probability becomes the Posterior Probability and is calculated using Bayes' theorem. So, the Posterior Probability is the probability of an event **X** occurring given that event **H** has occurred.

For example

Suppose, three bags have the labels A, B, and C. One bag has a red ball in it, while the other two do not. The prior probability of red ball found in bag B is one-third or 0.333. But when bag C is seen, and the result shows that there is no red ball in that bag, then the posterior

probability of red ball found in bag A and B becomes 0.5, as each bag has one out of two chances.

Formula

Bayes' Theorem, can be mathematically represented by the equation given below :

P(H/X)=P(X/H)P(H)/P(X)$P(H/X) = P(X/H)P(H)/P(X)$

Where,

- **H** and **X** are the events and,
- **P (X) ≠ 0**
- **P(H/X) -** Conditional probability of H.

  Given that X occurs.
- **P(X/H) -** Conditional probability of X.

  Given that H occurs.
- **P(H) and P(X) -** Prior Probabilities of occurring H and X independent of each other.

Applications of Bayes' Theorem

In the real world, there are plenty of applications of the Bayes' Theorem. Some applications are given below :

- It can also be used as a building block and starting point for more complex methodologies, For example, The popular Bayesian networks.
- Used in classification problems and other probability-related questions.
- Bayesian inference, a particular approach to statistical inference.
- In genetics, Bayes' theorem can be used to calculate the probability of an individual having a specific genotype.