What is data mining

It is computation to facilitate knowledge discovery in databases. Its goal is to provide more tools and techniques for KDD. Data Mining is a process of sorting large datasets to discover valuable information and use it for analysis to increase efficiency in business operations.

What is KDD

KDD stands for **Knowledge Discovery in Databases**. It refers to the overall process of discovering useful knowledge from data, which includes various steps such as data preprocessing, data mining, and interpretation/evaluation of the discovered patterns.

Steps of KDD

1. Selection: Choosing appropriate data
2. Preprocessing: Cleaning the relevant data
3. Transformation: Convert data into usable common formats
4. Data Science: Apply algorithms and tools
5. Interpretation: Evaluate the results found

Parts of DM

1. Model: is to be fit on data
2. Search: technique to evaluate data point
3. Preference: criteria to select one model over other

Tasks of Data Mining (Also called models of Data Mining)

1. Predictive Tasks

These tasks aim to **predict unknown or future values** of other variables.

- **Classification**
    - Maps data into **predefined classes or labels**.
    - **Example**: Email classification (Spam or Not Spam), disease diagnosis (Diabetic or Non-Diabetic).
- **Regression**
    - Predicts a **continuous numeric value**.
    - **Example**: Predicting house prices, stock market trends.
- **Prediction**
    - A generalization of both classification and regression.
    - Focused on **forecasting future data** using historical patterns.
    - **Example**: Sales prediction for next quarter.
- **Time Series Analysis**
    - Analyzes data that arrives **in sequence over time**.

    - Useful for identifying **trends, seasonality, and cyclic behavior**.
    - **Example**: Weather forecasting, sensor data analysis

2. Descriptive Tasks

These tasks aim to **understand the underlying structure** of the data without prior knowledge.

- **Clustering**
    - Groups data into clusters of similar records **without predefined labels**.
    - **Example**: Customer segmentation in marketing.
- **Summarization**
    - Provides a **compact description** of the dataset (often statistical or visual).
    - **Example**: Reporting average income by region.
- **Association Rule Mining**
    - Discovers interesting **relations between variables** in large datasets.
    - **Example**: Market Basket Analysis (If a customer buys bread, they are likely to buy butter).
- **Sequence Pattern Mining / Sequential Pattern Discovery**
    - Finds **regular sequences or patterns** in ordered data.
    - **Example**: Web clickstream analysis, DNA sequence analysis.

KDD Issues

- Human interaction
- Overfitting, Outliers
- Large dataset
- High dimension
- Multimedia data
- Missing data
- Irrelevant data
- Noisy data
- Changing data

Data preprocessing for KDD

- Data cleaning
- Data integration
- Data reduction
- Data transformation
- Data discretization

Correlation analysis

- chi square test → sum ((observed-expected)^2 /expected)
- Pearsons product moment coefficient

Wavelet transformation compresses data by capturing both frequency and location information, making it ideal for multi-resolution analysis. It is space-efficient as it represents signals with fewer coefficients while preserving important features.

Principal Component Analysis (PCA) is a dimensionality reduction technique that transforms correlated features into a set of uncorrelated principal components.It is space-efficient and improves performance by retaining the most significant variance in fewer dimensions.

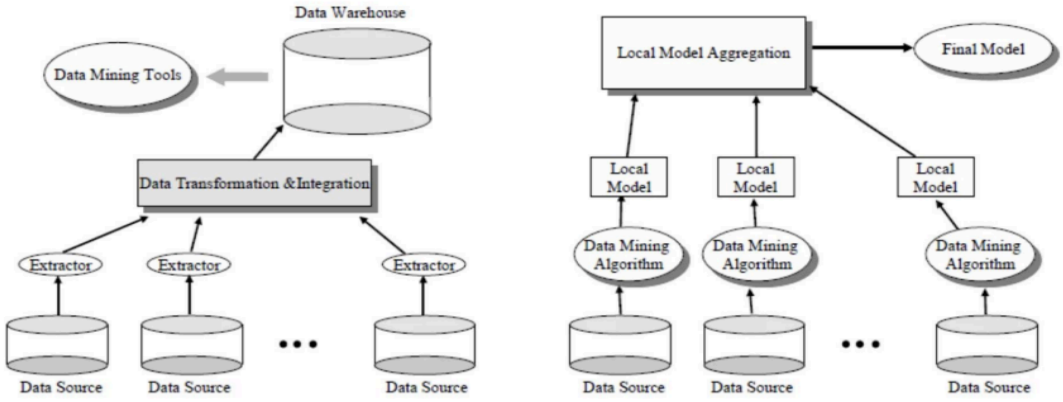
Sampling -in data preprocessing involves selecting a representative subset of data to reduce processing time while preserving patterns.

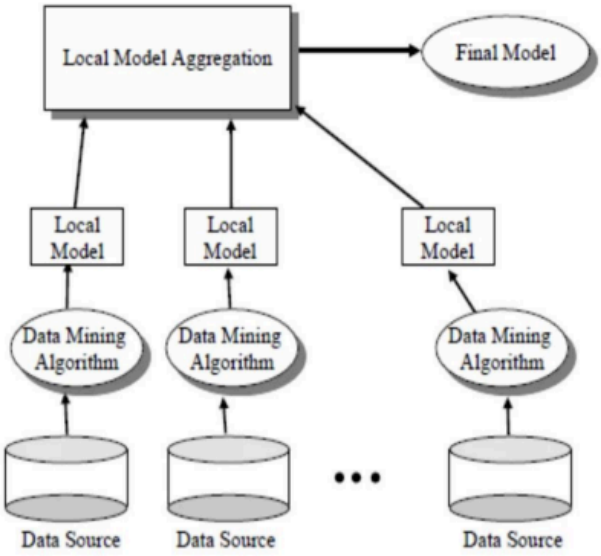
- **Min-Max Normalization**: scales values to a fixed range $[0, 1]$ using $x' = \frac{x-min}{max-min}$.
- **Z-score Normalization**: standardizes data to zero mean and unit variance using $x' = \frac{x-\mu}{\sigma}$.
- **Decimal Scaling**: moves the decimal point of values based on the maximum absolute value using $x' = \frac{x}{10^j}$, where j ensures all values are <1.

What is Distributed Data Mining(DDM)

Distributed Data Mining process involves the mining of distributed datasets stored in multiple local Databases. With the help of Distributed Data Mining, admins can perform data analysis and mining operations in a distributed manner to discover knowledge and use it efficiently for business operations.

Architecture of Distributed Data Mining (DDM)





Steps in DDM (Distributed Data Mining) – CIRTMPR

1. **Data Cleaning** – Removes noisy, inconsistent, or incomplete data.
2. **Data Integration** – Combines data from multiple sources into a unified dataset.
3. **Data Reduction** – Reduces data volume and dimensionality while preserving integrity.
4. **Data Transformation** – Converts and aggregates data into suitable formats for mining.
5. **Data Mining** – Extracts useful patterns or knowledge from large datasets.
6. **Pattern Evaluation** – Identifies interesting patterns based on defined measures.
7. **Knowledge Representation** – Presents mined knowledge using visualization or reporting tools.

3 Types of DMM - **MMG**

- **Multi-Agent System** – Uses identical agents that collaborate across distributed nodes to analyze and solve problems.
- **Meta-learning** – Builds local models on distributed datasets and combines them into a global classifier.
- **Grid** – Distributes computation to remote resources, enabling data mining where the data resides.

Advantages

1. **Efficient for Distributed Data** – Avoids costly centralization by analyzing data locally across multiple sites.
2. **Scalable** – Handles large datasets efficiently by parallelizing tasks across nodes.
3. **Improved Performance** – Enables simultaneous query execution at different locations for faster processing.

4. **Better Decision-Making** – Generates analytical models and reports to support strategic business decisions.

Types of Data to be mined

• Database data (extended-relational, object-oriented, heterogeneous,legacy) data warehouse, transactional data, stream, spatiotemporal, time-series, sequence, text and web, multi-media, graphs & social and information networks

Text
Set
Matrix
Graphs
Images
Time Series
Sequence

4 key applications of Data Mining:

1. **Market Basket Analysis** – Identifies products frequently bought together to optimize product placement and cross-selling.
2. **Fraud Detection** – Detects unusual patterns in transactions to identify and prevent fraudulent activities.
3. **Customer Segmentation** – Groups customers based on behavior or demographics for targeted marketing.
4. **Healthcare Diagnosis** – Analyzes patient data to assist in disease prediction, diagnosis, and treatment planning.

How to handle Missing Data

Deletion

- **Listwise Deletion**: Remove entire rows with missing values.
- **Pairwise Deletion**: Use only available values for analysis without removing entire rows.

Imputation

- **Mean/Median/Mode Imputation**: Replace missing values with the mean (for continuous), median, or mode (for categorical) of the column.
- **Regression Imputation**: Predict missing values using regression based on other variables.
- **K-Nearest Neighbors (KNN) Imputation**: Fill in missing values based on the nearest neighbors.
- **Multiple Imputation**: Create multiple datasets with imputed values and combine results for more robust estimates.

Using Algorithms That Handle Missing Data

- Some models (like **decision trees** or **XGBoost**) can inherently handle missing values without preprocessing.

Using Placeholder or Indicator Variables

- Replace missing values with a constant (e.g., “Unknown” or -999) and optionally add a binary flag column indicating missingness.

Hadoop Introduction:

Hadoop is an **open-source distributed computing framework** developed by the **Apache Software Foundation**. It is designed to **store and process large-scale datasets** across clusters of computers using simple programming models. It provides **scalability, fault-tolerance, and high availability**, making it ideal for big data processing.

Core Components of Hadoop:

1. **HDFS (Hadoop Distributed File System)**
    - A distributed storage system that splits large files into blocks and stores them across multiple nodes for redundancy and fault tolerance.
2. **YARN (Yet Another Resource Negotiator)**
    - Manages and schedules computing resources in the cluster; separates resource management from job scheduling.
3. **MapReduce**
    - A programming model for processing large datasets in parallel across distributed nodes using two functions: **Map** (filter/sort) and **Reduce** (aggregate results).
4. **Hadoop Common**
    - A set of shared utilities and libraries needed by all Hadoop modules; includes Java libraries and essential files

- Hive: A data warehouse tool that allows users to query data using SQL-like queries (HiveQL). It simplifies the process of querying data stored in Hadoop.
- Pig: A scripting platform that allows users to write complex data processing tasks in a simpler, high-level language called Pig Latin.
- HBase: A distributed NoSQL database built on top of HDFS, suitable for real-time access to large datasets.
- ZooKeeper: A coordination service that helps manage distributed systems, ensuring reliability and synchronization.
- Oozie: A workflow scheduler that allows you to automate Hadoop jobs.
- Spark: Although not part of the core Hadoop project, Apache Spark is a fast, in-memory data processing engine that is often used alongside Hadoop for quicker data processing than traditional MapReduce.
- Flume: A tool for efficiently collecting, aggregating, and moving large amounts of log data into Hadoop.

Problems with Apriori

Problem	Solution / Improvement
1. Generates a large number of candidate sets	Use **hash-based techniques** or **reduce candidate generation** by improving pruning strategies.
2. Multiple database scans	Use algorithms like **FP-Growth** that scan the database fewer times by building compact data structures.
3. High computational cost and time-consuming	Employ **sampling** or **partitioning** methods to reduce dataset size, or use **parallel and distributed mining** techniques.
4. Inefficient with dense datasets	Use algorithms like **Eclat** or **FP-Growth** which handle dense data better by avoiding candidate generation.
5. Memory consumption due to large candidate sets	Use **vertical data format** or **bitset representations** to optimize memory usage.

1. Multilevel Association Rules

- These rules are derived from data at **different levels of abstraction or hierarchy** within a dataset.
- For example, items in a supermarket can be categorized hierarchically:
    - Level 1: *Beverage*
    - Level 2: *Soft Drink*
    - Level 3: *Coca-Cola*
- **Purpose**: To discover patterns not only at a detailed level but also at higher conceptual levels.
- **Example**:
    - Level 1 Rule: *Beverages → Snacks*
    - Level 2 Rule: *Soft Drinks → Chips*
    - Level 3 Rule: *Coca-Cola → Lays Chips*

- **Use case**: Useful for businesses wanting insights across different product categories, not just individual items.

---

2. Multidimensional Association Rules

- These rules involve **more than one dimension (attribute)** or table in the database, not just items.
- **Types**:
  - **Intra-dimensional**: Rules within the same dimension.
    - Example: `buys(X, "Milk") → buys(X, "Bread")`
  - **Inter-dimensional**: Rules across different dimensions.
    - Example: `age(X, "20-30") ∧ income(X, "High") → buys(X, "Laptop")`
- **Purpose**: To find associations that span across different attributes (like age, location, income, product).
- **Use case**: Targeted marketing, personalized recommendations, demographic-based analysis.

A **Bloom filter** is a space-efficient probabilistic data structure used to **test whether an element is possibly in a set or definitely not**. It uses multiple hash functions to map elements to bits in a bit array. If all the bits for an element are set, the element *might* be in the set (with some false positive chance), but if any bit is not set, the element is definitely not in the set. Bloom filters are fast and memory-efficient but do not store the actual elements and can have false positives.

Mining Social Network Graphs – An Overview

Mining social network graphs refers to the process of analyzing and extracting meaningful patterns, insights, and knowledge from **social network data** represented as a **graph** of **nodes** (users, entities) and **edges** (relationships or interactions).

1. What is a Social Network Graph?

A **social network graph** is a **graph-based representation** of social relationships:

- **Nodes (Vertices):** Individuals, users, or entities (e.g., people, organizations).
- **Edges (Links):** Relationships or interactions (e.g., friendship, communication, followers).

Graphs can be:

- **Directed** (e.g., Twitter: A follows B)
- **Undirected** (e.g., Facebook friends)
- **Weighted** (e.g., number of messages sent)

2. Objectives of Mining Social Network Graphs

- **Community Detection**: Identify groups of closely connected nodes (e.g., friend groups, interest clusters).
- **Influence Analysis**: Find influential users (e.g., opinion leaders, influencers).
- **Link Prediction**: Predict future relationships (e.g., friend suggestions).
- **Anomaly Detection**: Detect unusual patterns (e.g., fake accounts, fraud).
- **Information Diffusion**: Analyze how information spreads (e.g., viral content, rumor spread).
- **Recommendation Systems**: Suggest content, friends, or products based on network behavior.

Dynamic Time Warping (DTW) is an algorithm for measuring similarity between two time-dependent sequences that may vary in speed or length. Rather than comparing points strictly one-to-one (as in Euclidean distance), DTW "warps" the time axis of each series to find the best possible alignment: it builds an M×NM\times NM×N cost matrix of all pairwise distances between points in series A(length M) and B(length N), then uses dynamic programming to trace a minimal-cost warping path from (1,1) to (M,N) under boundary (must start and end aligned), monotonicity (can't go backwards in time), and step-size (moves can be diagonal, vertical, or horizontal) constraints.

PageRank is an algorithm developed by Google founders Larry Page and Sergey Brin to rank web pages in search engine results. It works on the principle that a page is important if it is linked to by other important pages. Instead of simply counting the number of incoming links, PageRank considers the quality of those links, assigning higher importance to pages that are linked by other high-ranking pages. The algorithm models a "random surfer" who randomly clicks links on the web; the likelihood of the surfer landing on a particular page determines its PageRank. A damping factor (usually set to 0.85) simulates the probability that the surfer will continue clicking, while the remaining probability accounts for randomly jumping to any page. This recursive and probabilistic approach allows PageRank to measure the relative importance of each page in a network, making it a foundational concept in web search and graph-based ranking systems.

Significance in Time Series Analysis

1. **Handles Temporal Distortions**
   Real-world signals—speech, ECG, financial data—often exhibit local speedups or slowdowns. DTW's flexibility lets you align similar patterns even if one series runs faster or slower, making it far more robust than direct point-wise comparisons.
2. **Improves Classification & Clustering**
   By providing a distance measure that respects temporal dynamics, DTW enables more accurate nearest-neighbor classification, clustering, and anomaly detection on time-series datasets where timing matters.
3. **Wide Applicability**
   From speech and gesture recognition to bioinformatics (e.g., DNA/protein sequences) and finance, DTW underpins many pattern-matching systems that must accommodate non-linear time shifts.

Recommendation system

A Recommendation System refers to a system that is capable of predicting the future preference of a set of items for a user, and recommend the top items.

1. Rule-Based System

- **Principle:** Recommends items based on predefined business logic or filters.
- **Advantage:** Easy to implement and control.
- **Limitation:** Lacks personalization and adaptability.

2. Market Basket Analysis

- **Principle:** Finds associations between frequently co-purchased items.
- **Advantage:** Increases cross-selling and combo opportunities.
- **Limitation:** Ignores user preferences and context.

3. Content-Based Filtering

- **Principle:** Recommends items similar to those the user has interacted with.
- **Advantage:** Personalized and works well with new users.
- **Limitation:** Can't suggest diverse or novel items.

4. Collaborative Filtering

- **Principle:** Uses behavior of similar users/items to generate recommendations.
- **Advantage:** Learns from user patterns without item metadata.
- **Limitation:** Suffers from cold-start and data sparsity.

5. Hybrid / Ensemble

- **Principle:** Combines multiple recommendation techniques for improved performance.
- **Advantage:** Balances strengths of different methods.
- **Limitation:** More complex to design and maintain.

6. Unsupervised Clustering-Based

- **Principle:** Groups users/items with similar behavior for segment-based recommendations.
- **Advantage:** Simple and scalable for segmentation.
- **Limitation:** Lacks personalization within clusters.

7. Classification-Based (Buying Propensity)

- **Principle:** Predicts likelihood of purchase using supervised learning.
- **Advantage:** Good for targeting high-intent users.
- **Limitation:** Not suitable for ranking or discovery.

8. Deep Learning-Based

- **Principle:** Learns complex patterns using neural networks from diverse inputs.
- **Advantage:** Highly accurate and adaptable to unstructured data.
- **Limitation:** Requires large data and is hard to interpret.

9. Graph-Based

- **Principle:** Models user-item relationships using graph structures.
- **Advantage:** Captures multi-level and contextual relationships.
- **Limitation:** Computationally expensive at scale.

10. Image-Based (Fashion Domain)

- **Principle:** Uses visual features to recommend visually similar items.
- **Advantage:** Effective for fashion and style-driven categories.
- **Limitation:** Needs quality images and visual relevance.

Incremental Data Mining Algorithms for Mining Frequent Patterns

- **Definition**: Incremental data mining focuses on updating the results of a mining algorithm when new data is added, without having to re-run the entire algorithm on the entire dataset.
- **Key Concepts**:
    - **Frequent Pattern Mining**: Identifying patterns that appear frequently in a dataset, such as associations between items in market basket analysis.
    - **Incremental Algorithms**: These algorithms are designed to efficiently update frequent patterns as new data arrives without needing to process the entire data again. Examples include:
        - **Apriori Algorithm** (Modified for incremental updates).
        - **FP-growth Algorithm** (Modified for incremental updates).

Streaming Data

- **Definition**: Streaming data refers to continuously generated data that is processed in real-time.
- **Characteristics**:
    - **High Volume**: Large amounts of data generated constantly (e.g., social media feeds).
    - **High Velocity**: Data arrives at high speeds (e.g., sensor data from IoT devices).
    - **Time-sensitive**: Immediate processing and analysis are often required.
    - **Continuous**: It’s a real-time, ongoing flow of data that doesn’t have a natural endpoint.
    - **Unbounded Size**: The amount of data is essentially infinite or very large, and you can't store all the data forever.

Issues and Challenges in Streaming Data Mining

- **Data Loss**: Due to the continuous nature of data, there is a risk of losing data if it cannot be processed fast enough.
- **Concept Drift**: The underlying data patterns may change over time (e.g., a sudden change in customer behavior).
- **Limited Memory**: Since streaming data can't be stored indefinitely, algorithms need to work with limited memory.
- **Real-Time Processing**: Efficient algorithms need to handle large volumes of data in real-time, often requiring low-latency processing.
- **Scalability**: Managing and processing large amounts of streaming data requires scalable solutions.

Streaming Data Mining Algorithms

- **Sliding Window Algorithms**: These algorithms use a window of the most recent data points and discard older ones. Commonly used for time-series data.
- **Hoeffding Tree**: A decision tree algorithm for data streams that uses a "Hoeffding bound" to decide when to split nodes in real-time.
- **Stream Clustering**: Clustering techniques (like K-means) that adapt to streaming data, updating the cluster centroids incrementally.
- **Count-Min Sketch**: A probabilistic data structure used to approximate the frequency of items in data streams, useful for counting frequent items in real-time.

Vector Space Model (VSM)

- **Definition**: A mathematical model for representing text documents as vectors in a multi-dimensional space.
- **Key Concepts**:
    1. **Term Frequency (TF)**: How often a term appears in a document.
    2. **Inverse Document Frequency (IDF)**: Measures how important a term is within the entire document collection.
    3. **TF-IDF**: A weighted measure of how important a word is in a document relative to the entire corpus.
- **Steps**:
    1. **Tokenization**: Split the text into words or tokens.
    2. **Weighting**: Assign weights (e.g., TF-IDF) to each term.
    3. **Vector Representation**: Represent each document as a vector of term weights in a high-dimensional space.
- **Cosine Similarity**: A measure of similarity between two vectors (documents). Cosine similarity is often used to compare document vectors.

Clustering

- **Definition**: Clustering is the process of grouping similar objects together based on certain features. In text mining, clustering is used to group similar documents or texts.

Flat Clustering (Non-hierarchical):

- **Definition**: All objects (documents) are grouped into a predefined number of clusters.
- **Popular Algorithms**:
    - **K-Means**: Partitioning method that divides the dataset into 'k' clusters by minimizing the variance within each cluster.
    - **K-Medoids**: Similar to K-means but uses actual data points as cluster centroids (medoids).
- **Advantages**:
    - Simple to implement.
    - Suitable for large datasets.
- **Disadvantages**:
    - Requires specifying the number of clusters (k) beforehand.
    - Sensitive to initial cluster centroids.

Hierarchical Clustering:

- **Definition**: Creates a tree-like structure (dendrogram) that shows the hierarchy of clusters.
    - **Agglomerative (Bottom-up approach)**: Starts with each document as a cluster, then merges the closest clusters iteratively.
    - **Divisive (Top-down approach)**: Starts with all data points in a single cluster and recursively splits them into smaller clusters.
- **Distance Measures**:
    - **Single Linkage**: Distance between the closest members of two clusters.
    - **Complete Linkage**: Distance between the farthest members of two clusters.
    - **Average Linkage**: Average distance between all pairs of members in the two clusters.
- **Advantages**:
    - Does not require the number of clusters to be specified in advance.
    - Produces a tree structure that can be visualized.
- **Disadvantages**:
    - Computationally expensive for large datasets.
    - Sensitive to noise and outliers.

Fast Update (FUP) is an incremental algorithm designed to efficiently update frequent itemsets when new transactions are added, avoiding the need to rerun the Apriori algorithm from scratch.

Key Concepts:

- Efficiently updates support counts for existing itemsets using only the new data.
- Detects newly frequent itemsets and revalidates old infrequent ones.
- Maintains the consistency of the association rules as data grows.

Steps:

1. **Scan New Data**: Count support for previously known frequent itemsets in the new transactions.
2. **Update Counts**: Add these counts to the existing support values.
3. **Identify Emerging Itemsets**: Check if any new itemsets now exceed the minimum support threshold.

4. **Re-evaluate Old Infrequent Itemsets**: Some may now qualify as frequent with the updated data.

Advantages:

- Reduces computation by reusing previously mined results.
- Suitable for applications with dynamic or streaming data.

Crawling

- **Definition**: Web crawling is the process of systematically browsing the web to collect web pages (documents) for indexing.
- **Web Crawlers** (also known as bots or spiders): Automated programs that visit websites, download content, and follow links to other pages.
- **Challenges**:
    - **Scalability**: Crawlers need to manage large-scale data.
    - **Content Duplication**: Avoid collecting duplicate content.
    - **Robots.txt**: Crawlers must respect the instructions in this file (which tells them which parts of a site to crawl and which to ignore).

Indexing

- **Definition**: Indexing is the process of organizing and storing the crawled web content in a way that makes it easy to retrieve during search queries.
- **Inverted Index**: A common indexing structure where a mapping is created between terms and the documents they appear in. This allows fast retrieval of documents based on the search terms.

Jaccard Coefficient is a similarity measure used to compare the similarity and diversity of two sets.

- $\text{Jaccard}(A,B) = |A \cup B| \ / \ |A \cap B|$