

# Department of Computer Science and Engineering

## Bigdata and Analytics (BCS714D)

### Module-05

#### Chapter 1: Spark and Big Data Analytics

#### Chapter 2: Text, Web Content, and Link Analytics

### 5.1 SPARK

1. Apache Spark is a tool used to process and analyze big data very fast.
2. It can be up to 100 times faster than older systems like Hadoop.
3. It works with HDFS and other storage systems.
4. Spark is easy to use because of its simple programming style.

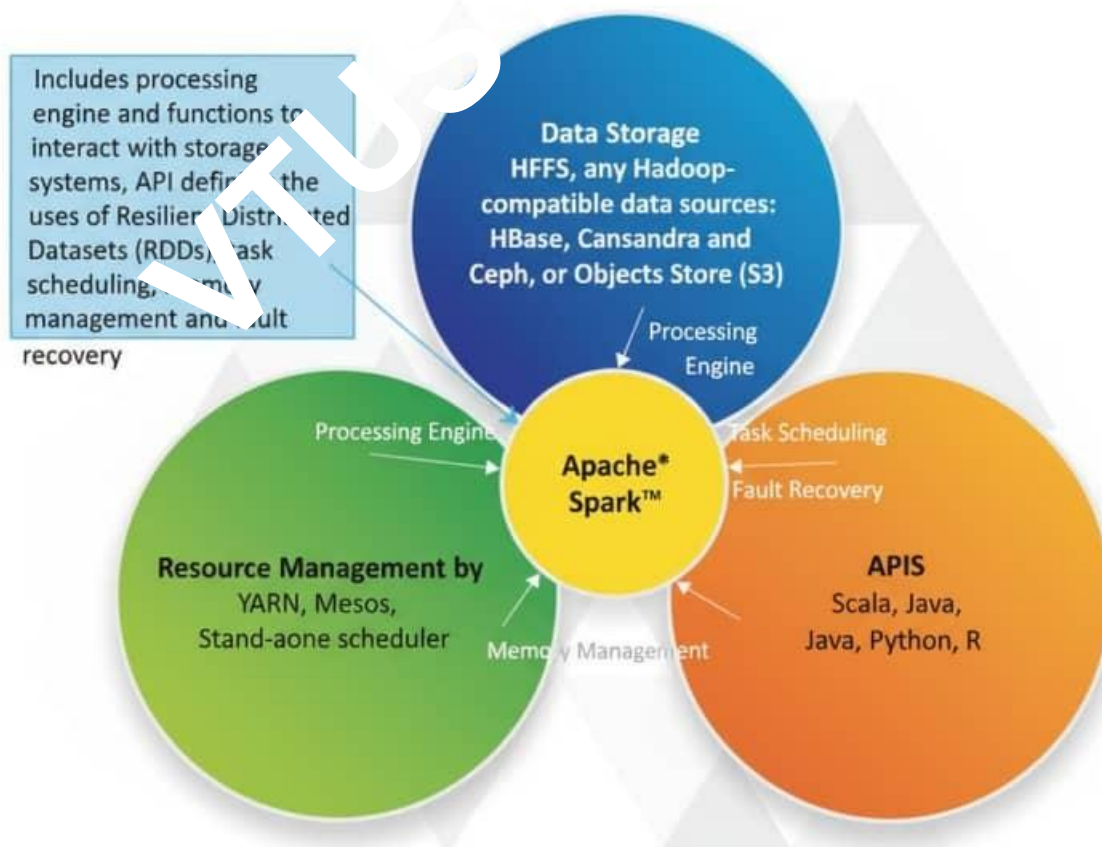
#### Main Features

- Spark is fast, scalable, and easy to use. □ It has different parts for different tasks:
  - Spark SQL – for working with data using SQL.
  - Spark Streaming – for live (real-time) data.
  - MLlib – for machine learning.
  - GraphX – for graph data and relationships.
- It is easy for both programmers and computers to understand.
- Used for:
  - Data analysis
  - Machine learning
  - Stream processing
  - Graph processing
- Spark runs on Windows, Linux, and Mac OS.

#### 5.1.1 Introduction to Big Data Tool-Spark Apache

Spark has three main parts:

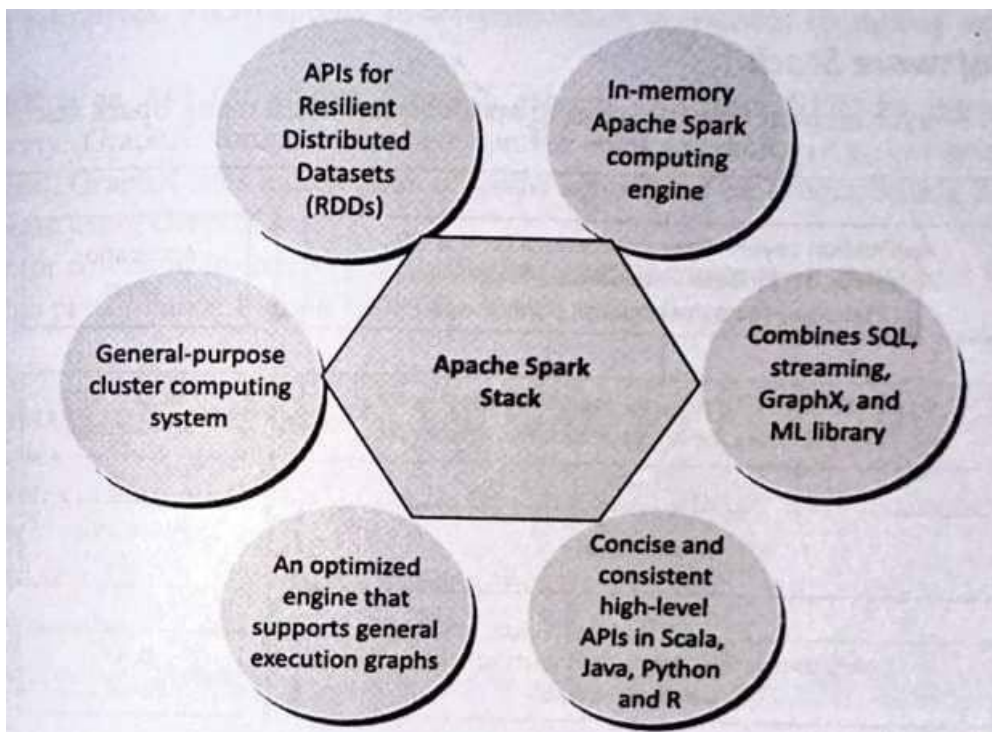
1. **Data Storage:** Uses HDFS or other compatible systems like HBase, Cassandra, Ceph, or cloud storage (Amazon S3) to store large amounts of data.
  2. **Standard APIs:** Provides easy-to-use interfaces in Scala, Java, Python, and R for building Spark applications for analytics, machine learning, and stream processing.
  3. **Resource Management:** Manages computing resources either on a single server (standalone mode) or across multiple machines using systems like YARN or Apache Mesos.
- **Ceph** is an open-source system that stores data as objects, blocks, or files and supports automatic replication.
  - Apache Mesos is a cluster manager that shares CPU, memory, and other resources efficiently across different applications.
  - Amazon S3 is a cloud storage service used to store big data files online.



#### 5.1.1.1 Features of Spark

- **RDD (Resilient Distributed Dataset):**
  - Core concept in Spark.

- Represents distributed data across many computers.
  - Supports parallel processing.
  - Fault tolerant (can recover lost data automatically). **Data Handling:**
  - Can process structured, semi-structured, and unstructured data.
  - Works with data from various sources.
- Includes Map and Reduce, plus many additional functions for flexible data processing.
- Performance Optimization:**
- Uses lazy evaluation to optimize query performance.
- APIs and Languages:**
- Provides easy-to-use APIs in Scala, Java, and Python.
  - Spark is written in Scala and runs on the JVM (Java Virtual Machine).
- Interactive Tools:**
- Includes interactive shells for Scala and Python.
  - Useful for testing, learning, and exploring data quickly.



- **Functions Supported:**

### 5.1.1.1 Spark Software Stack

Five-Layer Architecture for Running Applications Using Spark Stack

#### 1. Layers in Spark Architecture

- **Application Support Layer:**  
Contains Spark **SQL**, **Streaming**, **R**, **GraphX**, **MLlib**, and **Arrow**.
- **Processing Layer:**  
**Spark Core** – the main processing engine that performs computation using RDDs.
- **Data Store Layer:**  
Provides data to Spark (e.g., **HDFS**, **Hive**, **Cassandra**, etc.).
- **Cluster Management Layer:**  
Managed by **Hadoop YARN**, **Mesos**, or **Standalone Spark** for task scheduling and resource management.
- **Underlying Infrastructure:**  
Physical servers or cloud systems where Spark runs.

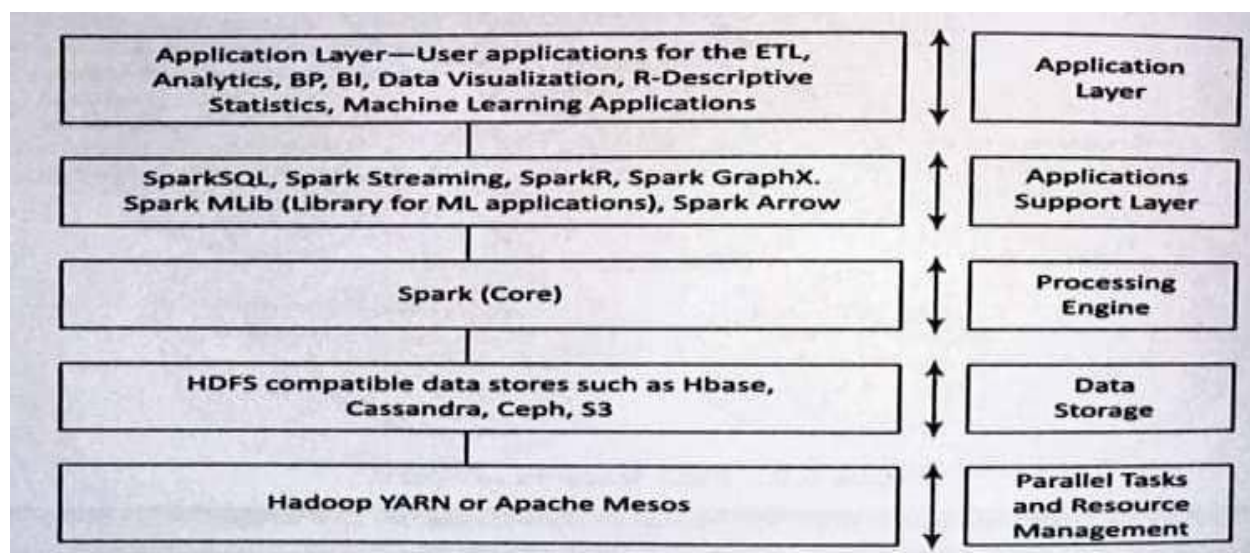
#### 2. Main Components of the Spark Stack

| Component              | Purpose / Function                                                                                                                                                |
|------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Spark SQL</b>       | For <b>structured data</b> processing. Runs SQL queries using JDBC/ODBC, supports HiveQL, UDFs, DataFrames, Parquet, Hive, and Cassandra.                         |
| <b>Spark Streaming</b> | For <b>real-time data</b> processing using <b>micro-batches</b> . Uses <b>DStreams</b> (series of RDDs).                                                          |
| <b>SparkR</b>          | <b>R interface</b> to Spark. Allows users to run Spark jobs interactively from <b>R shell</b> using RDD API.                                                      |
| <b>Spark MLlib</b>     | <b>Machine Learning library</b> for scalable learning algorithms — supports classification, regression, clustering, recommendation, and dimensionality reduction. |
| <b>Spark GraphX</b>    | For <b>graph processing and analytics</b> . Extends RDD with graph properties. Supports operators like subgraph, joinVertices, and aggregateMessages.             |

|                    |                                                                                                                              |
|--------------------|------------------------------------------------------------------------------------------------------------------------------|
| <b>Spark Arrow</b> | For <b>in-memory columnar analytics</b> and <b>vectorized UDFs (VUDFs)</b> . Improves performance for Python data pipelines. |
|--------------------|------------------------------------------------------------------------------------------------------------------------------|

### 3. Key Features of Apache Spark

- Fast and general-purpose **compute engine**.
- **RDDs** for distributed and fault-tolerant data processing.
- Supports **batch**, **streaming**, **machine learning**, and **graph** computations.
- Works with **structured**, **semi-structured**, and **unstructured** data.
- Offers **APIs in Scala, Java, and Python**.
- Provides **interactive shells** for learning and testing.
- Integrates with **Hadoop**, **Hive**, **Cassandra**, etc.



## 5.2 INTRODUCTION TO DATA ANALYSIS WITH SPARK

**Data Analysis** is the process of **inspecting**, **cleaning**, **transforming**, and **modeling** data to **discover useful information**, **draw conclusions**, and **support decision-making**.

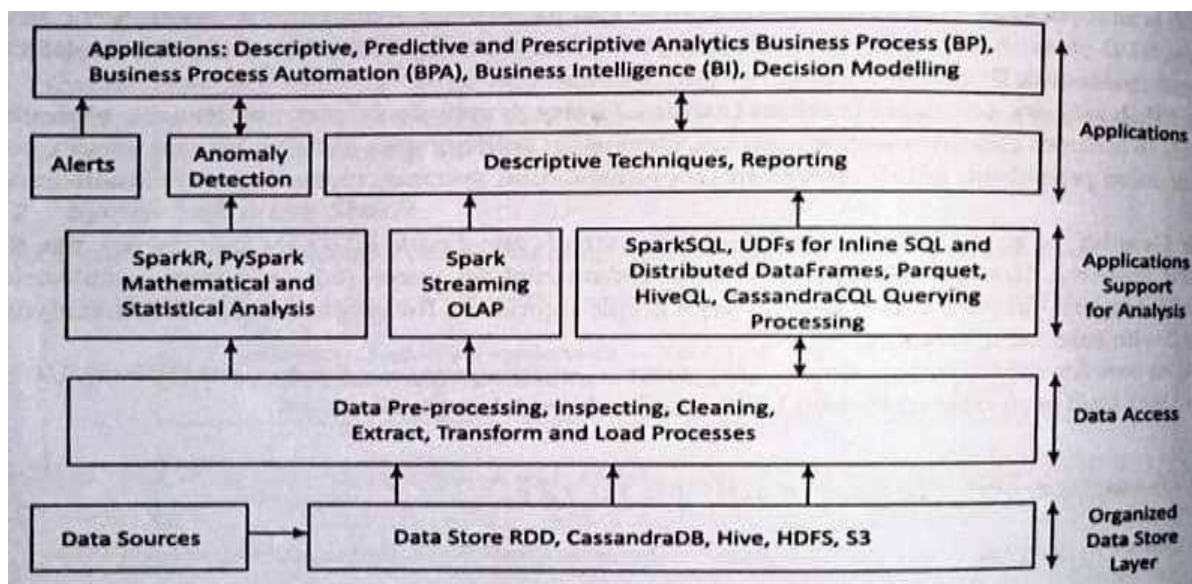
**Spark for Data Analysis**

- Spark supports **data analysis** using advanced features like:
  - **SQL and DataFrames** ◦ **UDFs (User Defined Functions)** ◦ **Vectorized UDFs (VUDFs)** ◦ **Group Vectorized UDFs**
  - **Pandas and Python libraries** for analytics and visualization.

### Example: Car Company

- A car company sells **5 models**, each in **16 colors**.
- Sales happen through **many showrooms** across the country.
- The company analyzes **annual sales and profits**:
  - **Model-wise** ◦ **Color-wise** ◦ **Region-wise** ◦ **Showroom-wise**
- This analysis helps the company:
  - Discover useful trends and insights. ◦ Perform **predictive analytics**. ◦ Plan **future manufacturing, marketing, and sales strategies**. **Steps in Data Analysis (Figure 5.4)**

| Step                                     | Description                                                                                                                                                                                                                                                                                                     |
|------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>1. Data Storage</b>                   | Collect and store data from multiple sources — e.g., <b>HDFS, Hive, Cassandra, or Amazon S3</b> .                                                                                                                                                                                                               |
| <b>2. Data Pre-processing</b>            | Clean and prepare the data before analysis: <ul style="list-style-type: none"> <li>• Remove outliers and inconsistent data</li> <li>• Filter irrelevant/redundant info</li> <li>• Perform cleaning, editing, reduction, or wrangling</li> <li>• Validate and transform data.</li> </ul>                         |
| <b>3. ETL (Extract, Transform, Load)</b> | Extract data, transform it into usable format, and load it for analysis.                                                                                                                                                                                                                                        |
| <b>4. Analysis / Processing</b>          | Perform <b>mathematical</b> and <b>statistical</b> analysis on the queried data. Use <b>OLAP</b> or other analysis tools.                                                                                                                                                                                       |
| <b>5. Applications of Analyzed Data</b>  | Use analyzed results for: <ul style="list-style-type: none"> <li>• <b>Descriptive, Predictive, Prescriptive analytics</b></li> <li>• <b>Business Process Automation (BPA)</b></li> <li>• <b>Business Intelligence (BI)</b></li> <li>• <b>Decision modeling</b></li> <li>• <b>Knowledge discovery</b></li> </ul> |



### 5.2.1 Spark SQL

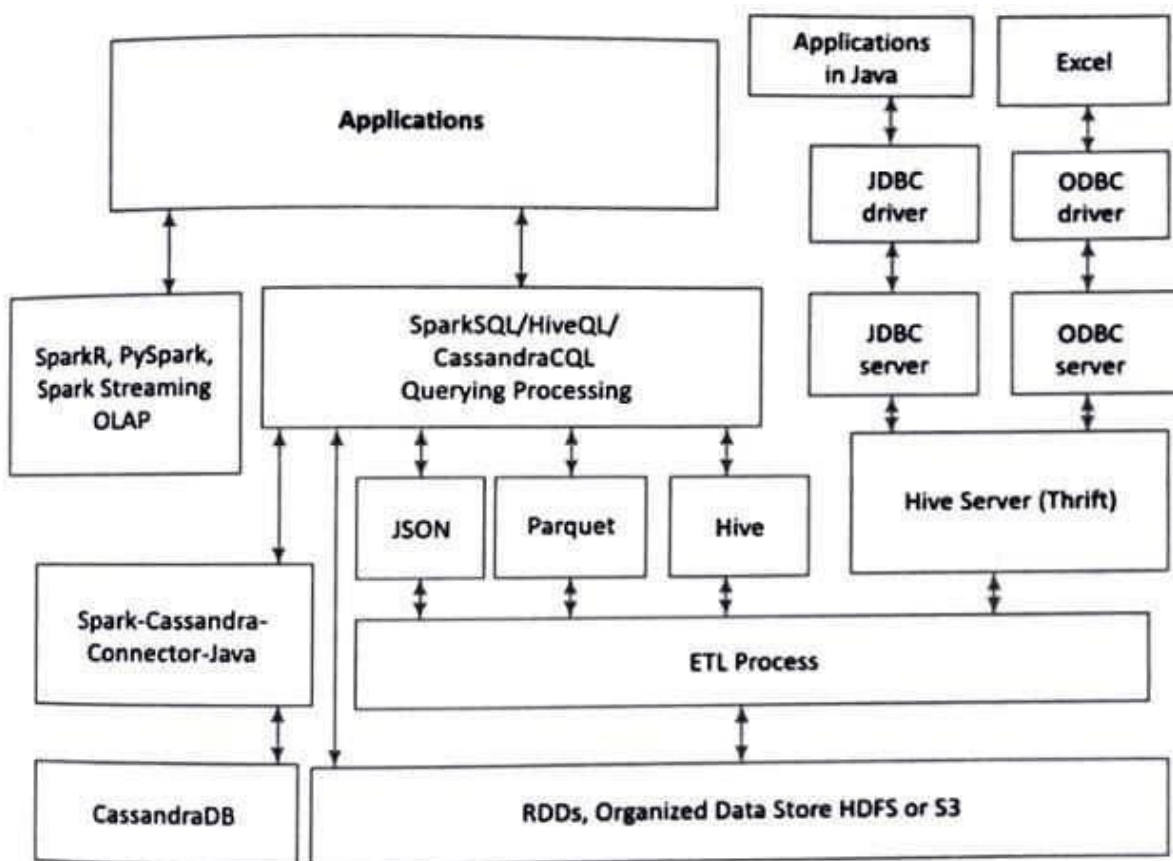
#### Functions of Spark SQL

1. Runs **SQL-like queries** using **Catalyst optimizer** and **Tungsten execution engine**.
2. Processes **structured data** efficiently.
3. Provides **flexible APIs** to connect with different **data sources**.
4. Performs **ETL operations** (Extract, Transform, Load) on data from **JSON, Parquet, Hive, Cassandra**, etc.
5. Supports **ad-hoc queries** for quick, on-demand data analysis.

#### Features of Spark SQL

1. Supports multiple languages – **SparkR, PySpark, Python, Java**, etc.
2. Provides **JDBC and ODBC APIs** for connecting external apps like **Excel** and **Java programs**.

3. Extracts and transforms data from various **file formats** for flexible querying.
4. Includes **Hive integration** – supports **Hive tables, databases, warehouses, UDFs, and SerDe.**
5. Allows **querying with HiveQL and Cassandra CQL.**
6. Supports **Spark Streaming** for **real-time (OLTP)** and **structured streaming** processing.



### A. DBC Server

1. JDBC (Java Database Connectivity) allows applications to **read data tables from RDBMS** using a **JDBC client**.
2. Many **Java applications** use **JDBC drivers and servers** to connect to databases.
3. **Spark SQL** provides **JDBC connectivity** for external applications to access Spark data.
4. **Command to start JDBC server:**
5. `./sbin/start-thriftserver.sh --master sparkMaster`

6. **Hive Thrift Server** allows remote clients (via JDBC) to send SQL queries to Hive and receive results.
7. Supports multiple languages — **Scala, Java, Python, and R**.

## B. JSON, Hive, Parquet, and DataFrames 1.

### JSON Objects:

- Store semi-structured data in key-value format.
- Commonly used for web and data interchange.

### 2. Hive Objects:

- Provide **database, tables, columns, partitions, and views**. ◦ Uses **HiveQL** for defining and querying data.
- Stores data in **HDFS** (reliable but slower in I/O).

### 3. Columnar Storage:

- Faster I/O performance than row storage.
- Loads only required columns during processing.
- Supports **compression, encoding, and parallel execution**.

### 4. File Formats:

- **RC (Record Columnar) File:** Data stored in columns, partitioned into row groups.
- **ORC (Optimized Row Columnar) File:** Stores data in “**stripes**”; supports **concurrent reads** and efficient metadata handling.
- **Parquet File:**
  - ✦ **Columnar storage** format optimized for Big Data queries.
  - ✦ Consists of **metadata and data blocks**.
  - ✦ Enables **fast retrieval** of specific columns.
  - ✦ Supported by three projects:
    - i. **parquet-format** – defines file structure and metadata.
    - ii. **parquet-mr** – handles reading/writing of nested column data.
    - iii. **parquet-compatibility** – ensures cross-language read/write support.

### 5. Spark DataFrame (SchemaRDD):

- A **distributed collection of data** organized into **named columns** (like database tables).
- Supports transformations such as **filter, join, and groupBy**. ◦ Previously known as **SchemaRDD**.
- Can be converted to and from **RDDs (Resilient Distributed Datasets)**.

### 6. Creating DataFrames from Various Sources:

- From **JSON datasets, Hive tables, Parquet files, external databases, or existing RDDs**.

- Commonly used in **PySpark** or **Hive programs** for merging and joining large datasets.

## A. Spark DataFrame Operations (Example 5.1)

1. **Table and DataFrames** ○ Table: toyPuzzleTypeCostTbl with **4 columns**: puzzleType, puzzleCode, numPieces, puzzleCost.
  - DataFrame1: toyTypeCodes → columns **puzzleType, puzzleCode**.
  - DataFrame2: toyCodesCost → columns **puzzleCode, puzzleCost**.
2. **Creating SQLContext** sqlContext =  
org.apache.spark.sql.SQLContext(sc)
3. **Creating DataFrame from Parquet file** toyTypeCost =  
sqlContext.parquetFile("toyPuzzleTypeCostTbl")
4. **Creating DataFrame from JSON file** toyPuzzleTypeCostTbl =  
sqlContext.read.json("toyPuzzleTypeCostTbl.json")
5. **Selecting columns for DataFrames** toyTypeCodes =  
toyPuzzleTypeCostTbl.select("puzzleType", "puzzleCode")  
toyCodesCost = toyPuzzleTypeCostTbl.select("puzzleCode",  
"puzzleCost")
6. **Joining DataFrames on puzzleCode** toyTypeCodesCost =  
toyTypeCodes.join( toyCodesCost, toyTypeCodes["puzzleCode"] ==  
toyCodesCost["puzzleCode"]  
)  
# Resulting DataFrame has columns: puzzleType, puzzleCode, puzzleCost
7. **SQLContext vs HiveContext** ○ **SQLContext**: basic Spark SQL; no  
HiveServer required.
  - **HiveContext**: full Hive support, access to HiveServer, HiveQL commands recommended.

8. **Aggregation & Statistical Functions** ○ Examples: count(), sum(), avg(), min(), max() ○ Statistical: stdev(), variance() ○ Useful for summarizing and analyzing DataFrames.

## B. Aggregation Example (Example 5.2 – Car Sales)

1. **Objective** ○ Find **total annual Jaguar Land Rover (JLR) sales** across all showrooms.
  - Identify **showroom with maximum JLR sales**.
2. **Query for total JLR sales**  

```
SELECT SUM(JLRDS)
FROM CarShowroomsCumulativeYearlySales;
```
3. **Query for showroom with max JLR sales**  

```
SELECT csID, MAX(JLRDS)
FROM CarShowroomsCumulativeYearlySales;
```
4. **Notes** ○ Aggregate functions (SUM, MAX) help in **business analytics**.
  - Can be executed using **Spark SQL** with SQLContext or HiveContext.

## 5.2.2 Using Python Advanced Features with Spark SQL

### 1. Python Features

- General-purpose, **interpreted, interactive, object-oriented** language.
- Supports **basic types, lists, dictionaries, sets, tuples, functions, classes**.
- Extensive **standard library**: regex, testing, web, databases, email, image processing, etc.
- Combines **performance and expressive coding**.
- Works easily with **Spark SQL** to build **Big Data applications**.

### 2. PySpark Example (JSON File Query) 1. Import Spark SQL and

#### HiveContext

```
from pyspark.sql import HiveContext, Row
hiveCtx = HiveContext(sc)
```

## 2. Use a Row Object

- Row objects store **individual records**.
- Example: toyPuzzleProduct with fields ProductCategory, ProductID, ProductName.

3. **Load a JSON File**

```
input = hiveCtx.jsonFile("toyTypeProductTbl") input.registerTempTable("toyPuzzleProduct")
```

## 4. Run a Query

```
result = hiveCtx.sql("SELECT ProductID, ProductName FROM toyPuzzleProduct ORDER BY ProductCategory")  
result.show()
```

- **HiveContext** connects Spark SQL to Hive.
- **Row objects** represent data records.
- JSON and other formats can be loaded and queried with PySpark.

### 5.2.1.1 Python Libraries for Analysis

#### 1. NumPy (Numerical Python)

- Provides **N-dimensional arrays** and **vector/matrix operations**.
- Supports **linear algebra**, **Fourier transforms**, **random numbers**, and **broadcasting functions**.
- Integrates with **C/C++/Fortran** for high-performance computation.
- Common operations:
  - np.loadtxt('file.txt') – load text file
  - np.genfromtxt('file.csv', delimiter=',') – load CSV
  - np.mean(arr, axis=0) – mean along axis
  - np.sum(arr), np.min(arr), np.max(arr) – aggregation
  - np.var(arr), np.std(arr, axis=0) – variance and standard deviation
  - np.add(arr1, arr2) – vector addition
  - np.corr() – correlation coefficient

#### 2. SciPy (Scientific Python)

- Built on top of NumPy.
- Provides **special functions**, **numerical integration**, **optimization**, **distance computations**.
- Key features:
  - Dense and sparse matrices
  - Polynomial and vector operations
  - Real and imaginary number functions
  - Matrix creation and linear algebra routines
- Example functions:
  - linalg.det(A) – determinant
  - np.cast['f'] – cast to type
  - b.flatten() – flatten array

- `np.vsplit(c, 2)`, `np.hsplit(d, 2)` – split arrays

### 3. Pandas (Python Data Analysis)

- Provides **Series**, **DataFrame**, and **Panel** structures.
  - **Series**: 1D labeled array
  - **DataFrame**: 2D table of Series
  - **Panel**: 3D container of DataFrames (4D possible with labels)

**Merge, join, and concatenate** objects like databases.

**RPy** interface for R functions.

Provides **statistics**, **machine learning**, **IDE**, and **API** support.

**SQL-like operations**: SELECT, WHERE, GROUPBY, JOIN, UNION, UPDATE, DELETE.

**GroupBy** feature: split-apply-combine workflow.

**Size mutability**: insert/delete columns in DataFrames and Panels.

**Slicing and dicing**: access lower-dimensional objects easily; axes can have **arbitrary names** (e.g., year, car



- **Key features:**

1.

- 2.
- 3.
- 4.
- 5.
- 6.
7. model,  
sales).

## 4. Spark Integration

- Spark 2.3 supports **Python UDFs (User Defined Functions)** and **Vectorized UDFs (VUDFs)** using **Apache Arrow** for **high-performance data pipelines**.

### 5.1.1.1 User-Defined Functions (UDFs)

#### 1. Python UDFs (User Defined Functions)

- UDFs let you **write custom functions** in Python for Spark SQL.
- Useful when **built-in functions are not enough**.

- **Example:**

Increase puzzle cost by 10%  
 from pyspark.sql.functions  
 import udf

```
df.withColumn('newCost', plusTenPercent(df.cost))
```

Works **row by row** → slower for big data

Can be used for **aggregations**, e.g. 'max sale' in a showroom.

```
@udf('float') def  
plusTenPercent(v):  
    return v + 0.1 * v
```

- 
- 

## 2. Vectorized UDFs (VUDFs)

- Faster version of UDFs using **column-wise (batch) operations**.
- Uses **pandas.Series** as input/output.
- Requires **Apache Arrow** for high performance. **Example:** Increase puzzle cost by 10% (vectorized)

```
from pyspark.sql.functions import pandas_udf
```

```
@pandas_udf('float') def
vectorized_plusTenPercent(v):
    return v + 0.1 * v

df.withColumn('newCost', vectorized_plusTenPercent(df.cost)) •
```

Works **on columns/batches** → much faster than regular UDFs.

- Ideal for **big datasets** and **ML pipelines**.

### 5.2.1.2 Grouped Vectorized UDFS (GVUDFS)

- GVUDFs use the **split-apply-combine** pattern from Pandas.
- Operates **on groups of data**, e.g., compute yearly sales **for each showroom**.

#### Steps

1. **Split** the Spark DataFrame into groups using `groupBy`.
2. **Apply** a vectorized UDF (`pandas.DataFrame` → `pandas.DataFrame`) to each group.
3. **Combine** results into a new DataFrame.
4. **Return** a new Spark DataFrame with the results.

#### Key Features

Input/output: `pandas.DataFrame`.

Grouping defined using `groupBy`.

Output size can differ from input.

Returns **StructType** columns (same type).

- 
- 
- 
- 

**Example: Add 10% to costs in group of toy products**

```
from pyspark.sql.functions import pandas_udf
```

```
# Define GVUDF
```

```
@pandas_udf(df.schema)
```

```
def costTenPercentPlus(pdf):
```

```
    return pdf.assign(v = pdf.v + 0.1 * pdf.v)
```

```
# Apply GVUDF to groups by 'id' df.groupby('id').apply(costTenPercentPlus)
```

- `df.v` → original cost column
- `v` → new column with increased cost
- Works **group by group**, faster than row-wise UDFs

### 5.3.3 Data Analysis Operations

#### Common Operations in Data Analysis

1. **Filter data** – select rows/columns based on conditions.
2. **Top-ten lists** – find top values or percentages.
3. **Sub-totals** – calculate totals for groups of data.
4. **Multi-field filters** – filter using multiple criteria.
5. **Unique lists** – extract unique values from repeating data.
6. **Remove duplicates/outliers** – detect and clean unusual or repeated data.
7. **Multi-key sorting** – sort data by more than one column.
8. **Count unique items** – find the number of distinct elements in a list.
9. **SUMIF/COUNTIF** – conditional sum and count functions.
10. **Database functions** – e.g., DSUM (sum), DMAX (maximum).
11. **Convert lists to tables** – organize data for analysis.

#### 5.3.3.1 Removing Outliers for Data Quality Improvement for Analysis

- **Outliers** are values that do not fit with the rest of the data.
- They can happen due to:

- Data-entry mistakes (e.g., missing decimals)
- Programming errors
- Temporary effects or delays in data stabilization
- **Why remove them:**
  - Outliers can **skew results**, making averages and analysis misleading.
- **Basic method using Python / PySpark:**
  1. Convert data to **numeric type** (if not already).
  2. Calculate **mean** and **standard deviation** of the data.
  3. Keep only values that are **within 3 standard deviations** of the mean.

# Example in PySpark

```
distanceNumerics = distances.map(lambda x: float(x))
stats = distanceNumerics.stats() # mean, stdev
mean = stats.mean()
stdev = stats.stdev()
```

```
reasonableDistances = distanceNumerics.filter(
    lambda x: abs(x - mean) < 3 * stdev
)
print(reasonableDistances.collect())
```

- Only the **reasonable values** (not outliers) are kept for analysis.

## Text, Web Content, Link, and Social Network Analytics Text

### analytics

- Text analytics means studying text to find patterns, meanings, and useful information.
- It works with unstructured data like messages and documents.
- Text data changes over time and can be grouped by topics or sentiments.
- Machine learning helps systems learn automatically from text.
- Example: When you book a flight, your phone can learn from SMS details and send reminders automatically.
- A vector is a list of related items, like letters in a word.
- A feature is a property of something, like color, shape, or size.
- A category is a group made using common features, like fruits or cars.
- A label is the name given to a category, like “sports news.”

- A dimension is how many features or variables describe something.
- Graph data uses nodes (things) and edges (connections) to show relationships.
- Social networks like Facebook or LinkedIn can be shown as graphs of people connected to each other.
- Web content mining means finding useful information from websites.
- Hyperlinks are links that connect one web page to another.
- Link analytics studies how web pages are connected to find important or related pages.
- The Count Triangles Algorithm finds groups of three connected points in a network.
- Graph node centrality measures how important a node is in a network.
- Degree centrality counts direct links a node has.
- Betweenness centrality shows how often a node connects others — meaning it's influential.
- Closeness centrality shows how near a node is to all others — meaning it can reach others quickly.
- The chapter explains text mining using machine learning methods like Naive Bayes and SVM.
- It also covers web mining, PageRank, and social network analysis to find clusters, hubs, and communities.
- Algorithms like SimRank and triangle counting help discover relationships and groups in networks.

## Text mining

- A lot of text data is created every day from news, social media, chats, emails, blogs, and reports.
- Text mining helps to analyze and find useful information from this large amount of text.
- It uses machine learning methods like Naive Bayes classifier and Support Vector Machines (SVM).
- Text data is often unstructured, changes with time, and contains different topics and sentiments.
- Text mining techniques include clustering, classification, event detection, and trend analysis.
- Text mining means finding hidden patterns, relationships, and trends from text that are not easily visible.
- It combines AI, databases, NLP, and data mining to analyze text data.

- Applications of text mining include:
  - Predicting stock movements from company reports
  - Product and business decision-making
  - Contextual advertising
  - Spam (email) filtering
  - Drug action or fraud detection
  - Knowledge management
  - Social media analysis
- Text mining helps answer questions like:
  - Which words or terms often occur together?
  - How is one piece of information linked to another?
  - What categories can be formed from the data? □ What can be predicted from the text?
- Natural Language Processing (NLP) helps computers understand human language.
- NLP uses ML algorithms that learn from examples and find patterns in sentences.
- Common NLP tasks are:
  - Text summarization
  - Sentiment analysis
  - Topic and concept extraction
  - Named entity recognition
  - Parts-of-speech tagging
  - Question answering and translation
- Information Retrieval (IR) means finding the right documents or data from large collections — like how search engines work.
- Information Extraction (IE) pulls structured data (like names, places, dates) from unstructured text.



- IE uses patterns (like “Citizen of <x>”) and semantic dictionaries to find meaning.
- Document Clustering groups similar documents together for easy searching and topic organization.
- Document Classification assigns documents into categories (like sports, news, or business).
- Web Mining finds patterns in website data to understand user behavior and improve online services.
- Concept Extraction groups words and phrases that have similar meanings into common concepts.

### **Text Mining Process**

- Text is the main way people exchange information, but it is unstructured and difficult to process.
- Text mining means analyzing text to find useful information for a specific purpose.
- A text document is made of characters → words → phrases → sentences.
- Text mining involves both word search and semantic (meaning-based) pattern search.
- The text mining process works like a pipeline with five main phases that run step-by-step.

### **Five Phases of Text Mining Process (Simple Summary)**

#### **1. Text Pre-processing – cleaning and preparing text**

- Removes unwanted or extra information (like comments, special symbols, or typos).
- Tokenization: splits text into words or tokens.
- POS tagging: labels words as noun, verb, adjective, etc.
- Word sense disambiguation: finds the correct meaning of a word (e.g., *bank* = riverbank or money bank).
- Parsing: builds a tree of how words relate grammatically in a sentence.

#### **2. Feature Generation – converting text into measurable data**

- Bag of Words: represents a document by the words it contains and how often they appear.

- Stemming: reduces words to their root form (e.g., *speaking, speaks, speaker* → *speak*).
- Stop words removal: removes common words like *a, the, in, for*.
- Vector Space Model (VSM): represents each document as a vector of word frequencies using TF-IDF (Term Frequency – Inverse Document Frequency).
- TF-IDF gives importance to words that appear often in one document but not in many others.

**Example:**

If “toys” appears 16 times in a 1000-word document →  $TF = 0.016$

If “toys” appears in 1000 out of 10 million documents →  $IDF = \log(10,000,000 / 1,000)$  = 4

Then  $TF-IDF = 0.016 \times 4 = 0.064$

### 3. Feature Selection – choosing the most useful data

- Removes irrelevant or repeated features to reduce complexity.
- Improves processing speed and accuracy.
- Uses methods like PCA (Principal Component Analysis) and LDA (Linear Discriminant Analysis).
- N-grams: identifies phrases with 2 or more consecutive words (e.g., “tasty food” = 2-gram).
- Detects noise or outliers (unusual data) and removes them.

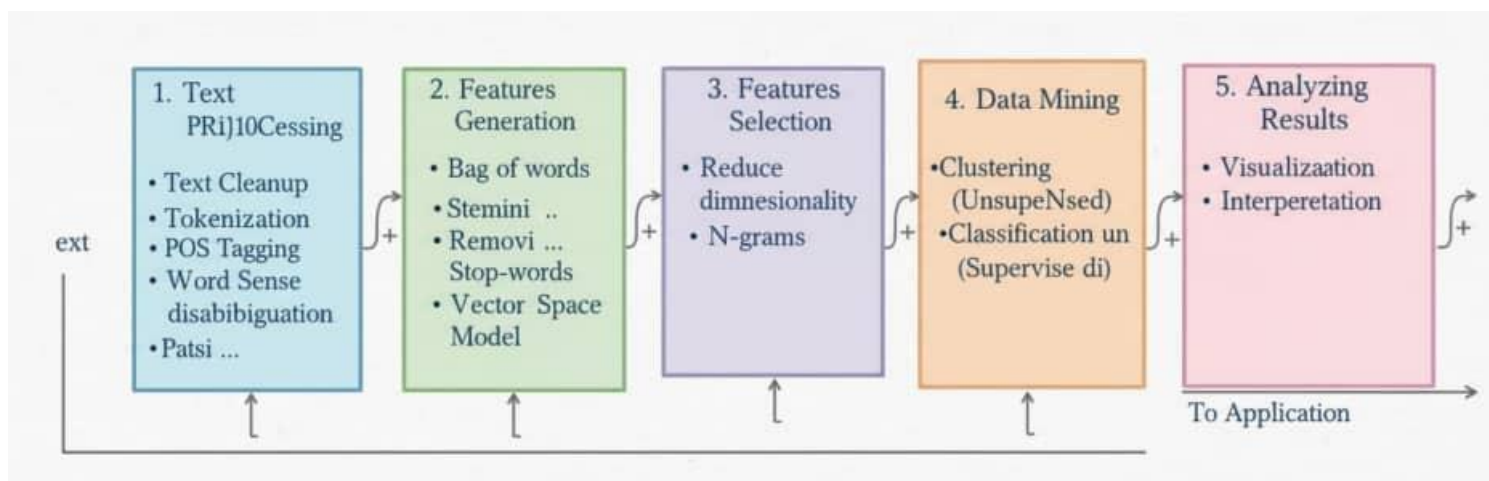
### 4. Data Mining Techniques – finding patterns or predictions

- Unsupervised Learning (Clustering): groups documents without predefined labels.
  - **Example:** finding topic groups in blogs or articles.
- Supervised Learning (Classification): trains on labeled data to classify new data.

- **Example:** spam detection or automatic news categorization. ◦ Common algorithms: Naive Bayes and SVM (Support Vector Machine).
- Evolutionary pattern analysis: tracks how topics or trends change over time (e.g., research trends or news events).

## 5. Result Analysis and Visualization

- Check and interpret results of the mining process.
- If results are good → use them; if not → review and improve the process.
- Create visuals or graphs to show findings.
- Apply results to improve business or institutional processes.



## Text Mining Challenges

1. **NLP (Natural Language Processing) Issues**
  - POS tagging (assigning word types like noun, verb).
  - Ambiguity (words with multiple meanings).
  - Tokenization (splitting text into words).
  - Parsing (finding grammatical relationships).
  - Stemming (reducing words to root form).
  - Synonymy and polysemy (same meaning or multiple meanings of words).
2. **Mining Technique Challenges**
  - Finding the **right algorithm** for text analysis. ◦ Handling **large volumes of data**.
  - Extracting **concepts and semantic relationships**.

- Working **without training data**.
- 3. **Variety of Data** ○ Different data sources need **different methods**.
  - Texts are often **unstructured** and **language-independent**.
- 4. **Information Visualization** – Showing results clearly using visuals or graphs.
- 5. **Real-time Processing** – Text streams (like social media) need **fast and efficient processing**.
- 6. **Scalability** – Systems must handle **growing amounts of text data** smoothly.

## Supervised Text Classification

1. **Supervised learning** means training models using **labeled data** (documents already categorized).
2. It uses **Information Retrieval (IR)**, **Machine Learning (ML)**, and **NLP** techniques.
3. The algorithm learns from examples (training data) to **classify new documents** into the right categories.
4. Each document is:
  - A **positive example** for its labeled categories.
  - A **negative example** for others.
5. The goal is to find a **weight vector** that best classifies new unseen documents.

## Common Methods for Supervised Text Classification

1. **K-Nearest Neighbour (KNN):** ○ Uses past labeled documents for training. ○ Finds the  $k$  most similar documents to a new one.
  - Classifies by **majority vote** (for categories) or **average** (for numbers).
2. **Support Vector Machine (SVM):** ○ Separates data into categories with the best dividing line or boundary.
3. **Naive Bayes Method:** ○ Uses probability to predict which category a document belongs to.
4. **Decision Tree:** ○ Uses a series of **if-then rules** to classify text based on word features.
5. **Decision Rule:** ○ Similar to decision trees but written directly as **logical conditions**.

## Naive Bayes Analysis

- ❑ **Naive Bayes** is a **probabilistic and statistical** text classification method.
- ❑ It is based on **Bayes' Theorem** and assumes **independence between words (features)** — this is called the *naive assumption*.
- ❑ It is also known as the **multivariate Bernoulli method**.

- The classifier calculates the **probability of a document belonging to each class** and assigns it to the one with the **highest probability**.
- For a given document (as a bag of words), the probability of belonging to class  $k$  is:  $P(k|D)$

where  $w_i$  are the words in the document.

- The **Maximum a Posteriori (MAP)** rule is used to find the most likely class:  $P(k) \prod_{i=1}^n P(w_i|k)$

?

(You drop the denominator because it is the same for all classes.)

- **Bag-of-words assumption:** the position of words in the document does not matter.
- **Conditional independence:** each word's probability is independent of others, given the class.
- To avoid **underflow** (tiny decimal values turning into zero), we use **logarithms** instead of multiplying probabilities:  $\log P(k) + \sum_{i=1}^n \log P(w_i|k)$

?



(Choosing the class with the highest log-score gives the same result.)

- Simple and fast to train.
- Works well even with a small training set.
- Needs less CPU and memory compared to methods like SVM or Random Forest.
- Not sensitive to irrelevant features.

- Sentiment analysis (positive/negative opinions).

- **Advantages:**

- **Applications:**

- Email spam filtering.
- Document classification.
- Language detection.
- Authorship identification.
- Age or gender detection.

## Support Vector Machines

1. **SVM** is a **supervised learning method**, meaning it needs **training data** (labeled examples).
2. It is used for **classification**, **pattern recognition**, **handwriting recognition**, **image classification**, and **bioinformatics**.

3. Each data point is represented as a **vector** in an **n-dimensional space** (for example, each word in text analysis can be one dimension).
4. The goal of SVM is to find a **hyperplane** (a line, plane, or surface) that **separates data into classes**.
  - In 2D → hyperplane is a **line** ○ In 3D → hyperplane is a **plane**
5. The **best hyperplane** is the one that has the **maximum margin** — i.e., the largest distance between the closest data points of different classes.
6. The **closest points** to the hyperplane are called **support vectors**.
  - These points are the **most important** because they determine the position and orientation of the hyperplane.
  - Each data point belongs to **Class +1** or **Class -1**.
  - The classifier tries to ensure:

$$\frac{1}{2} \sum_{i=1}^n \left( \frac{y_i}{\|w\|} - 1 \right)^2 \quad \text{Class +1} \quad \frac{1}{2} \sum_{i=1}^n \left( \frac{y_i}{\|w\|} + 1 \right)^2 \quad \text{Class -1}$$

$$\frac{1}{2} \sum_{i=1}^n \left( \frac{y_i}{\|w\|} - 1 \right)^2 + \frac{1}{2} \sum_{i=1}^n \left( \frac{y_i}{\|w\|} + 1 \right)^2$$

**Kernel functions** are used when data **is not linearly separable** — they map data into higher-dimensional space so a hyperplane can separate it.

7. The **maximum-margin hyperplane** (middle line between two classes) gives the **best classification** and is less likely to make mistakes.
8. **Binary classification:**

9. The **decision function** is:

where **w** = weight vector and **b** = bias.

10.

11. **Applications of SVM:**

- Text and document classification
- Handwriting recognition
- Image recognition
- Pattern recognition
- Outlier (anomaly) detection
- Bioinformatics (gene and protein classification)

## **WEB MINING, WEB CONTENT AND WEB USAGE ANALYTICS**

### **Web Data**

1. The **Web** is a collection of **interconnected files** stored on web servers.
2. **Web data** includes three main parts:
  - **Web content:** text, images, videos, records.
  - **Web structure:** hyperlinks, tags, website connections.
  - **Web usage:** server logs, browsing history, and click data.
3. **Features of web data:**
  - **Large volume:** huge amount of information available easily.
  - **Heterogeneous:** contains different types of data (text, images, videos).
  - **Variety and diversity:** information on almost every topic in different formats (text, tables, lists, audio, video).

- **Semi-structured:** web pages are created using **HTML**, which has a nested structure.
- **Hyperlinks:** connect pages within a website or across different websites.
- **Redundancy:** same or similar information can appear on many pages.
- **Multiple sections per page:** main content, ads, menus, navigation bars, and copyright info.
- **Web forms:** allow users to enter data (like login or search forms) that is sent to the server.
- **Dynamic content:** web pages change frequently, with fast-growing information (e.g., social media updates).

## Web Mining

❑ **Data mining** means finding patterns and useful knowledge from large datasets.

→ Process: **Raw Data** → **Patterns** → **Knowledge**

❑ **Web data mining** applies data mining techniques to web data.

- It combines **data mining**, **machine learning**, **natural language processing**, **statistics**, **databases**, **information retrieval**, and **visualization**.

❑ **Definition:**

Web mining is the process of **extracting useful knowledge** from **web data**, such as web documents, hyperlinks, and logs.

❑ **Applications of web mining:**

- Extracting important parts from web documents
- Finding web graph patterns (like **PageRank**)
- Identifying users, sessions, and detecting malicious activity
- Finding usage path patterns

❑ **Types of Web Mining (Web Mining Taxonomy):** Web mining is divided into **three main categories**:

### (i) Web Content Mining

- Extracts information from the content of web pages.

- Content includes **text, images, audio, video**, and **structured data** (tables, lists).

## (ii) Web Structure Mining

- Finds structural information (how pages are linked).
- Two main structures:
  - **Hyperlinks:** links within a page (intra-document) or between pages (interdocument).
  - **Document structure:** web pages = nodes; hyperlinks = edges (web graph).

## (iii) Web Usage Mining

- Analyzes user behavior and browsing patterns using logs.
- Types of logs:
  - **Web Server Logs:** record IP address, page viewed, access time.
  - **Application Server Logs:** help in finding service-related issues.
  - **Application Level Logs:** record system or user activities for auditing or troubleshooting.



## Web Content Mining

### Definition:

Web Content Mining is the process of **discovering information or resources from the content of web documents** across the World Wide Web.

- Can be done via:
  1. **Direct mining** of web documents
  2. **Mining through search engines** (faster than direct mining)

### Relationship with other fields:

- **Data mining:** Web content is similar to structured data from databases/files; traditional data mining techniques can be applied.
- **Text mining:** Much of web content is textual, requiring text-mining techniques.
- **Web data structure:** Mostly **semi-structured** or **unstructured**, unlike fully structured data.

### Applications of Web Content Mining

1. Classifying web documents into categories

2. Identifying topics of web documents
  3. Finding similar web pages across servers
  4. Relevance-related applications:
    - **Recommendations:** Top “n” relevant documents
    - **Filters:** Show/hide documents based on criteria
    - **Queries:** Enhance query results using user/role/task-based relevance
- 

## Common Web Content Mining Techniques

### Pre-processing Steps

1. **Text Extraction:** Extract text from HTML.
2. **Data Cleaning:** Fill missing values, smooth noisy data.
3. **Tokenizing:** Convert text into word tokens.
4. **Stemming:** Reduce words to their root form (e.g., “closed” → “close”)
  - Porter algorithm commonly used.
5. **Stop Words Removal:** Remove common words like *a, an, the, in, for*.
6. **Collection Frequency (CF):** Count occurrences of a term in the collection.
7. **Document Frequency (DF):** Number of documents containing a specific term.
8. **Term Frequency (TF):** Count of a term’s occurrences in a single document.
9. **Bag of Words:** Represent documents as a set of words with their frequencies.

**Note:** CF and DF can differ: a word can appear many times in few documents (high CF, low DF).

---

## Mining Tasks for Web Content Analytics

1. **Classification (Supervised)** ◦ Assigns a new web document to a predefined category. ◦ Uses **term vectors** and algorithms trained on existing data.
2. **Clustering (Unsupervised)** ◦ Groups documents with similar features without pre-defined categories.
  - Similarity often measured by **dot product** of document vectors.
3. **Association Analysis** ◦ Identifies correlations between web documents that often appear together.
4. **Topic Identification, Tracking, and Drift Analysis** ◦ Organizes web pages into topics using text similarity or co-citation.
  - Helps track new topics, view counts, and topic evolution.
5. **Concept Hierarchy Creation** ◦ Organizes categories and subcategories to capture relationships. ◦ Factors: category structure (flat, tree, network), max categories per document, category dimensions (domain, time, location).
6. **Relevance Assessment** ◦ **Document Relevance:** Usefulness of a document in a situation. ◦ **Query-based Relevance:** Similarity between search keywords and documents. ◦ **User-based Relevance:** Personalized relevance based on user profile.
  - **Role/Task-based Relevance:** Relevance based on role or task profiles, not individual users. [Web Usage Mining](#)

**Definition:**

Web Usage Mining is the process of **discovering and analyzing patterns in click streams** and other data generated as a result of user interactions with web resources.

**Purpose:**

- Understand user behavior
- Improve website structure, personalization, and recommendations

- Detect anomalies or malicious activity

## Phases of Web Usage Mining 1. Pre-processing

Converts raw usage data from various sources into a structured form suitable for pattern discovery. Key steps include:

- **Data Cleaning:** Remove irrelevant or noisy data (e.g., image requests, failed requests).
- **Feature Extraction:** Extract relevant features like URL, timestamp, session ID, IP address, etc.
- **Feature Reduction:** Reduce dimensionality by removing redundant or insignificant features.
- **User/Session/Page Identification:** Identify unique users, sessions, and pages visited.
- **Formatting & Data Summarization:** Transform the data into structured tables for analysis.

## 2. Pattern Discovery

Identifies meaningful patterns in pre-processed data using techniques from statistics, data mining, machine learning, and pattern recognition:

- **Statistical Analysis:** Discover usage trends and distributions.
- **Association Rules:** Find frequently co-occurring pages or sequences.
- **Clustering:** Group users or sessions with similar behaviors.
- **Classification:** Categorize users or sessions based on predefined patterns.
- **Sequential Patterns:** Discover frequent navigation paths.

## 3. Pattern Analysis

Filters and interprets the discovered patterns to extract actionable knowledge:

- Remove **uninteresting or trivial patterns**
- Apply **query-based analysis**, **OLAP**, and **visualization** tools for deeper insights

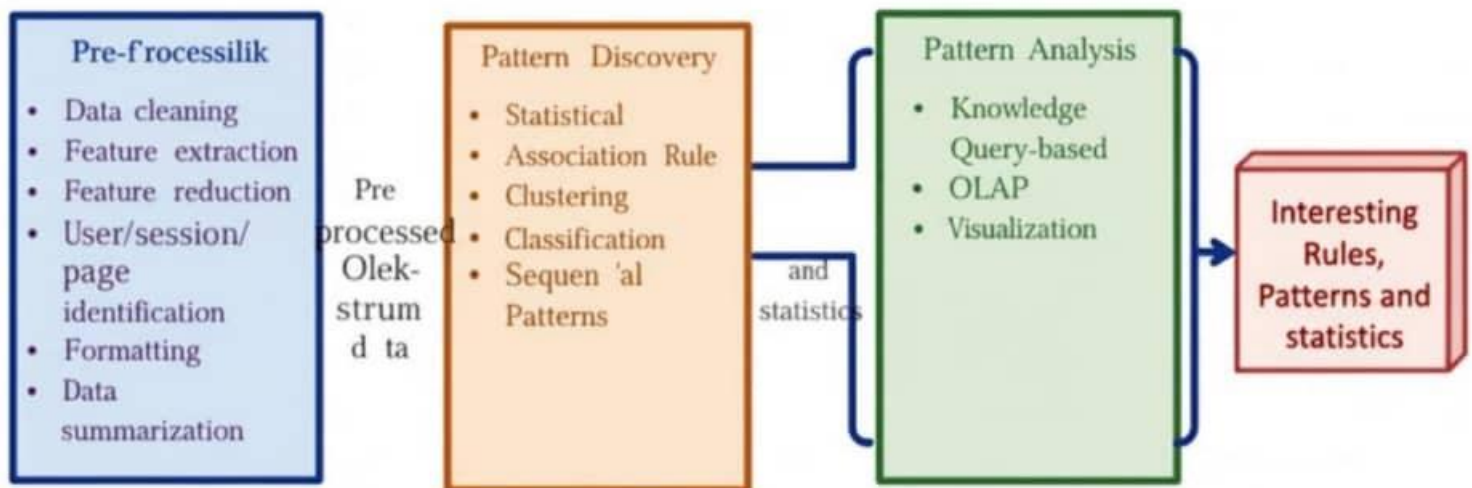
## Sources of Web Usage Data

1. **Server-level logs:** HTTP requests and responses collected at the web server.

2. **Client-level logs:** Data from the user's browser or device.
3. **Proxy-level logs:** Data collected by intermediate proxy servers.

### Usage patterns captured:

- Single-user access
- Multi-user access
- Single-site or multi-site navigation



## Pre-processing

### Definition:

Pre-processing is the **data preparation phase** in web usage mining. It transforms raw usage data into a structured form suitable for applying common data mining techniques (e.g., using the **vector space model**).

### Objectives of Pre-processing

Identify and structure the following:

1. **Users:**
  - Through cookies, logins, or URL information.

## 2. **Sessions:**

- A sequence of pages accessed by a single user during one visit.

## 3. **Content:**

- Extract state variables from server logs for each active session.

## 4. **Page References:**

- Record which pages were accessed in each session.

### **Tasks in Pre-processing**

#### 1. **Data Extraction:**

- Collect usage data from multiple sources (server logs, client logs, proxy logs).

#### 2. **Data Cleaning:**

- Remove irrelevant or noisy data (e.g., images, failed requests).

#### 3. **Feature Extraction:**

- Identify relevant features such as URLs, timestamps, session IDs, user IDs.

#### 4. **Feature Reduction:**

- Eliminate redundant or insignificant features to reduce dimensionality.

#### 5. **User Identification:**

- Detect unique users based on cookies, IP addresses, or login info.

#### 6. **Session Identification:**

- Define the boundaries of user sessions.

#### 7. **Page Identification:**

- Determine the specific pages accessed by users in sessions.

#### 8. **Data Formatting:**

- Transform data into a structured form suitable for pattern discovery algorithms.

#### 9. **Data Summarization:**

- Aggregate information to generate meaningful statistics for analysis.

### **Pattern Discovery**

After **pre-processing**, web usage data is ready for **knowledge extraction** using methods from statistics, machine learning, and data mining. The choice of technique depends on the analyst's goals.

#### 1. **Statistical Techniques**

- Extract descriptive knowledge about users.
- Common metrics: page views, viewing time, path length.
- Applications:

1. Identify **most frequently accessed pages**.
2. Compute **average page view time** or **average path length**.
3. Support **marketing and site optimization decisions**.

## 2. Association Rule Mining

- Discovers relationships between pages accessed together in a session.
- Applications:
  1. Identify correlations between users visiting similar content (e.g., course info and eBooks).
  2. Recommend products (e.g., books on related topics).
  3. Help web designers **restructure websites**.
  4. Preload documents to **reduce page access time**.

## 3. Clustering

- Groups items with **similar features**.
- Types of clustering in WUM:
  - a. User Clustering:**
    - Groups users with similar browsing behaviors.
    - Applications: Personalized content, e-commerce user segmentation.
  - b. Web Page Clustering:**
    - Groups pages with related content.
    - Applications: Search engines, content recommendation.
- Techniques:
  - **Model-based clustering:** Uses theoretical models (e.g., Expectation Maximization).
  - **Distance-based clustering:** Measures distance/similarity between users or pages; includes **partitional** and **hierarchical** clustering.

## 4. Classification

- Assigns data items to **predefined classes**.
- Applications:
  1. Develop **user profiles** based on categories.
  2. Discover rules from server logs (e.g., demographic patterns).
- Common algorithms: Decision Trees, Naive Bayes, k-NN, Support Vector Machines.

## 5. Sequential Pattern Discovery

- Analyzes **user navigation sequences** on a website.
- Models navigation as **trails** or sequences of pages visited.
- Markov Model Representation:
  - Each page = **state**.
  - Transition probability = likelihood of navigating from one page to another.
- Applications:
  - Compute **user behavior metrics**.
  - Discover **frequent navigation patterns** and rules.

## Pattern Analysis

The objective of pattern analysis is to filter out uninteresting rules or patterns from the rules, patterns or statistics obtained in the pattern discovery phase.

The most common form of pattern analysis consists of:

- (i) A knowledge query mechanism such as SQL
- (ii) Another method is to load usage data into a data cube in order to perform Online Analytical Processing (OLAP) operations
- (iii) Visualization techniques, such as graphing patterns or assigning the colors to different values, can often highlight overall patterns or trends in the data
- (iv) Content and structure information can filter out patterns containing pages of a certain usage type, content type or pages that match a certain hyperlink structure.

- Data cube enables visualizing data from different angles. For example, toys data visualization using category, colour and children
- preferences. Another example, news from category, such as sports, success stories, films or targeted readers (children, college students, etc).

## PAGE RANK, STRUCTURE OF WEB AND ANALYZING A WEB GRAPH

### Page Rank Definition

- ❑ In-degree (visibility) is the number of links pointing **to** a page.
- ❑ Out-degree (luminosity) is the number of links pointing **from** a page to others.
- ❑ Earlier PageRank approach:
  - A page with more in-links than out-links has higher authority.
  - PageRank measured a page's importance by its in-degree relative to outdegree.
- ❑ Limitation: Earlier method does not consider the **importance of linking pages** (parents).
- ❑ New PageRank approach (Page et al., 1998):
  - Considers the **whole web**, not just local links.
  - Takes into account the **authority of parent pages** linking to the page.
  - Pages linked by important pages get higher rank.

### Web Structure

- ❑ Web can be represented as a **directed graph** where pages are vertices and hyperlinks are edges.
- ❑ Page U has n hyperlinks represented as a vector  $u = [u_1, u_2, \dots, u_n]$ .

- ❑ Page V has m hyperlinks represented as a vector  $v = [v_1, v_2, \dots, v_m]$ .
- ❑ A hyperlink from Page U points to another page, e.g., Page V.
- ❑ Hyperlinks can have **two meanings**:
  - The text of a hyperlink describes the **destination page**.
  - A hyperlink conveys some **authority** from the linking page to the destination page.
- ❑ **Example**: Page U has hyperlinks to Page V (1), Page W (2), Page X (2), Page U' (1), and Page Y (2).
- ❑ The structure shows how pages and hyperlinks **interconnect** and influence authority flow.

## Hubs and Authorities

- ❑ A hub is a page that links to many content pages.
- ❑ A content page or authority has useful, reliable, and important information.
- ❑ In-degrees (number of links pointing to a page) are not always a good measure of authority, because a few links from highly authoritative pages may be more valuable than many links from low-authority pages.
- ❑ HITS (Hypertext-Induced Topic Selection) algorithm computes hub scores and authority scores for pages related to a specific topic t.
- ❑ HITS works on a sub-graph of the web relevant to the topic t.
- ❑ Steps of HITS algorithm:
  - Use a search engine to find a root set R of pages (e.g., top 200 pages for topic t).
  - Build a sub-graph S by including pages pointed to by R and pages pointing to R (limit to ~50 pages per “reverse link” query).
  - Remove purely navigational links and links between pages on the same host.
  - Consider only pages with 4–8 out-links from a given hyperlink.
- ❑ HITS uses a bipartite graph:
  - Hubs link to authorities.
  - Authorities are linked by hubs.

- ❑ Hub score of a page increases if it links to many high-authority pages.
- ❑ Authority score of a page increases if it is linked by many high-hub pages.
- ❑ The process is iterative: hub and authority scores are updated repeatedly until they converge.

## Web Communities

- ❑ Web communities are websites or collections of websites that **restrict content and links to members**.
- ❑ Examples include **LinkedIn, SlideShare, Twitter, Facebook**, DIY sites, blogs, and bulletin boards.
- ❑ Main issues in web communities are **privacy** and **reliability of information**.
- ❑ Analysis of web communities uses **web graph metrics**, such as:
  - **Triangle count** – number of triangles formed by three connected nodes.
  - **Clustering coefficient** – measures how tightly nodes are connected.
  - **K-neighbourhood** – number of 1st, 2nd, 3rd... neighbor nodes ( $K = 1, 2, 3...$ ).
  - **K-core analysis** – identifies cores (dense connected groups of nodes) which can be triangles, rectangles, or groups of interconnected cores. ❑ **Spark GraphX** functions can
- ❑ analyze:
  - Degree centrality and distribution
  - Betweenness and closeness centrality
  - Neighborhoods and strongly connected components
  - Triangle counts and PageRank
  - Shortest path and Breadth First Search (BFS)
  - Minimum spanning tree or forest
  - Spectral clustering and cluster coefficient

## Limitations of Link, Rank and Web Graph Analysis

- ❑ Search engines rely on metadata of pages; if metadata is biased, the page rank can be unfairly high.
- ❑ Search engines may favor advertising clients, causing biased rankings.
- ❑ A top authority page might act as a hub for a different topic, increasing its rank incorrectly.
- ❑ Topic drift and content changes can affect ranks; off-topic pages may appear as authorities.
- ❑ Affiliated pages or sites can boost each other's ranks artificially.

- ❑ Ranks can be unstable, as adding new pages or links may significantly change existing ranks.