Sairam Konar D15C 27 2024-2025

Experiment 9

Aim: To perform Exploratory data analysis using Apache Spark and Pandas

Theory:

1. What is Apache Spark and How Does It Work?

Answer:

- **Introduction to Apache Spark**
  Apache Spark is an open-source, distributed computing framework designed for fast and scalable big data processing. Developed at UC Berkeley's AMPLab, Spark provides an optimized engine for large-scale data analytics, machine learning, and real-time processing. Unlike traditional single-node tools (like Pandas or Excel), Spark efficiently handles massive datasets by distributing computations across clusters.

- **Key Features of Apache Spark**
  - **Unified Analytics Engine:** Supports batch processing, real-time streaming, machine learning (MLlib), and graph processing (GraphX).
  - **In-Memory Processing:** Retains intermediate data in RAM, significantly speeding up iterative algorithms.
  - **Fault Tolerance:** Recovers lost data partitions automatically using lineage information.
  - **Multi-Language Support**: APIs available in Python (PySpark), Scala, Java, and R.
  - **Lazy Evaluation:** Optimizes execution plans by delaying computation until necessary.

- **How Apache Spark Works**
  Spark operates in a master-slave architecture, where:

  - **Driver Node (Master)**: Coordinates tasks, schedules jobs, and manages execution.
  - **Worker Nodes (Slaves)**: Perform distributed computations in parallel.

  Core Components:

  - **Resilient Distributed Dataset (RDD)**: Fundamental data structure in Spark.
    Immutable, partitioned collections processed in parallel.
    Supports transformations (e.g., map, filter) and actions (e.g., count, collect).
  - **DataFrames & Datasets:** Higher-level abstractions built on RDDs, optimized for structured/semi-structured data.
    Support SQL-like operations (e.g., groupBy, join).
  - **Spark SQL**: Enables querying structured data using SQL syntax.
  - **Spark Streaming:** Processes real-time data streams in micro-batches.
  - **MLlib & GraphX:** Libraries for machine learning and graph processing.

- **Use Cases of Apache Spark:**
  - **Real-Time Analytics**: Fraud detection, IoT sensor monitoring.
  - **Large-Scale ETL:** Processing terabytes of log files.
  - **Machine Learning:** Training models on distributed datasets.
  - **Social Media Analysis:** Sentiment analysis, trend detection.
  - **Financial Data Processing**: Risk modeling, transaction analysis.

2) How is Data Exploration Done in Apache Spark?
Answer:

- **Initializing Spark Session**
  Before performing any operations in Spark, we need to create a SparkSession, which acts as the entry point for interacting with Spark's functionalities. This session configures the application name, cluster settings, and other parameters required for distributed computing.

- **Loading Data into Spark**
  Spark supports reading data from various sources (CSV, JSON, Parquet, databases, etc.). When loading data:
  - **Header:** Specifies if the first row contains column names.
  - **Infer Schema:** Automatically detects data types (e.g., integers, strings) or enforces a predefined schema.
  - **Partitioning:** Large datasets are split into chunks (partitions) for parallel processing.

- **Inspecting Data Structure**
  After loading, we examine:
  - **Schema:** Lists column names and their data types (e.g., StringType, IntegerType).
  - **Sample Records:** Displays the first few rows to understand the data layout.
  - **Summary Statistics:** Computes basic metrics (count, mean, min/max, standard deviation) for numerical columns.

- **Handling Missing Data**
  Missing values (null/NaN) can distort analysis. Common strategies:
  - **Dropping Rows**: Remove records with missing values (if they're insignificant).
  - **Imputation:** Replace nulls with mean/median (for numerical data) or mode (for categorical data).
  - **Flagging:** Mark missing values for further investigation.
- **Aggregations and Grouping**
  Spark allows:
  - **GroupBy:** Segregate data by categories (e.g., sales by region).
  - **Aggregate Functions:** Compute summaries like sum(), avg(), countDistinct().

- **Pivoting:** Reshape data for cross-tabulation (e.g., sales per product per month).

- **Filtering and Sorting**
  - **Filtering:** Subset data based on conditions (e.g., transactions > $100).
  - **Sorting:** Order records by specific columns (ascending/descending).

- **Data Visualization (via Pandas Conversion)**
  Spark DataFrames lack built-in plotting, so we:

  - **Limit Data Size:** Convert a subset (e.g., 1000 rows) to a Pandas DataFrame.
  - **Use Matplotlib/Seaborn**: Generate histograms, scatter plots, or box plots for trends/outliers.

Conclusion:
This experiment aimed to understand how Exploratory Data Analysis (EDA) can be performed using both Apache Spark and Pandas. Spark, with its distributed and in-memory processing, is ideal for analyzing large-scale datasets, while Pandas is better suited for smaller, quick analysis tasks. We explored the step-by-step EDA process—loading, inspecting, cleaning, summarizing, and analyzing data. Each tool serves a specific purpose, and together they offer flexibility and efficiency across data sizes. Understanding how these tools work prepares data professionals to handle diverse analytical challenges with the right approach.