CAPSTONE PROJECT REPORT

Distributed Machine Learning for Banking Data Analytics

1. Project Overview

In the modern banking sector, organizations generate and manage massive volumes of data related to customers, transactions, campaigns, and financial behavior. Efficient processing and analysis of this data is essential for improving customer engagement, managing risk, and making data-driven decisions.

This capstone project demonstrates how **distributed computing and machine learning technologies**—including **Hadoop, Hive, Apache Spark, Spark ML, and Spark Streaming**—can be used to analyze large-scale banking data and extract actionable insights. The project simulates a real-world banking analytics environment using the **bank.csv** dataset.

2. Dataset Description

The dataset used in this project is **bank.csv**, which contains customer and marketing campaign information.

Key Columns:

- **age** – Age of the client
- **job** – Job category
- **marital** – Marital status
- **education** – Education level
- **default** – Credit default status
- **balance** – Account balance
- **housing** – Housing loan status
- **loan** – Personal loan status
- **contact** – Contact communication type
- **day, month** – Last contact date
- **duration** – Contact duration (seconds)

- **campaign** – Number of contacts in current campaign
- **pdays, previous, poutcome** – Previous campaign details
- **y** – Target variable (term deposit subscription)

3. Tools and Technologies Used

Technology	Purpose
Hadoop HDFS	Distributed data storage
MapReduce	Batch processing & aggregation
Apache Hive	SQL-based querying on HDFS
Apache Spark	Distributed data processing
Spark SQL	Structured analytics
Spark ML	Machine learning modeling
Spark Streaming	Real-time data processing (simulated)
Google Colab	Spark execution environment
Ubuntu (WSL)	Hadoop & Hive setup

4. Hadoop: Data Ingestion and MapReduce Analysis

4.1 Data Ingestion into HDFS

- Created directories in HDFS for banking data
- Uploaded **bank.csv** into HDFS
- Verified file storage using HDFS commands

4.2 MapReduce Jobs Implemented

Job 1: Average Account Balance by Job

- Calculated average balance for each job category
- Output stored in HDFS

Job 2: Housing Loan Distribution by Education

- Counted number of customers with and without housing loans
- Grouped by education level

Job 3: Monthly Campaign Contacts and Subscription Status

- Analyzed number of contacts per month
- Evaluated subscription outcomes (y)

Job 4: Average Contact Duration by Campaign Outcome

- Calculated average call duration for each previous campaign result

Job 5: Relationship Between Age and Balance

- Summarized age-wise balance trends

✓ These jobs demonstrate **parallel batch processing using MapReduce**.

5. Hive: Data Warehousing and SQL Analytics

5.1 Database and Table Creation

- Created Hive database: banking_data
- Created external table client_info matching dataset schema
- Loaded data from HDFS into Hive table

5.2 Basic Exploration

- Counted total number of clients
- Displayed sample records

5.3 Filtering and Sorting

- Retrieved married clients with personal loans
- Identified top 10 clients with highest balances

5.4 Aggregation and Grouping

- Average age by job category
- Default counts by education level

5.5 Advanced Analytics

- Top job categories by average balance
- Campaign success rate by month
- Correlation between age and balance
- Subscription rate by previous campaign outcome
- Comparison of contact duration for subscribers vs non-subscribers

✓ Hive enabled **SQL-based distributed analytics on large datasets**.

6. Apache Spark: Data Processing and Analysis

6.1 Data Loading and Inspection

- Loaded bank.csv into Spark DataFrame
- Inspected schema, records, and summary statistics

6.2 Data Filtering and Transformation

- Filtered high-balance clients
- Created derived columns such as age groups
- Used Spark UDFs for categorization

6.3 Aggregation and Analysis

- Average balance by job
- Subscription counts by marital status
- Loan default rate analysis
- Correlation between age and balance

✓ Spark enabled **fast in-memory distributed processing**.

7. Spark ML: Predictive Modeling

7.1 Objective

Predict whether a client will subscribe to a **term deposit** (y) using machine learning.

7.2 Data Preprocessing

- Handled categorical variables using StringIndexer
- Selected relevant numerical features
- Created feature vectors using VectorAssembler

7.3 Model Selection

- **Logistic Regression** chosen due to:
    - Binary classification nature
    - Interpretability
    - Efficiency in distributed environments

7.4 Model Training

- Split data into training (80%) and testing (20%)
- Trained model using Spark ML pipeline

7.5 Model Evaluation

- Evaluated using **Area Under ROC Curve (AUC)**
- Achieved strong predictive performance

✓ Spark ML demonstrated **scalable machine learning on distributed data**.

8. Spark Streaming: Real-Time Data Processing

8.1 Stream Simulation

- Simulated real-time banking transactions using existing dataset
- Added timestamp columns to mimic streaming events

8.2 Real-Time Aggregation

- Calculated average balance and duration by job category
- Displayed continuous updates

8.3 Window Operations

- Applied time-based window aggregations
- Analyzed trends across defined time intervals

8.4 Late and Out-of-Order Data Handling

- Explained use of **watermarking** conceptually
- Described how Spark handles delayed events in real-time systems

✓ Streaming component fulfilled **real-time analytics requirement**.

9. Data Parallelism and Resource Management

- Data partitioning performed automatically by Spark
- Parallel execution used for aggregations and ML training
- Observed efficient CPU and memory utilization
- Spark managed task scheduling and fault tolerance internally

10. Challenges Faced and Learnings

Challenges:

- Local Spark installation issues due to network constraints
- Managing multiple distributed frameworks simultaneously
- Understanding coordination between Hadoop, Hive, and Spark

Learnings:

- Practical experience with distributed ecosystems
- Understanding batch vs real-time analytics
- Building scalable ML pipelines
- Importance of resource and task management

11. Conclusion

This project successfully demonstrates how **distributed computing and machine learning** can be applied to real-world banking data. By integrating **Hadoop, Hive, Spark, Spark ML, and Spark Streaming**, the project simulates a complete end-to-end analytics pipeline used in modern financial institutions.

The insights generated from this system can support:

- Customer targeting
- Campaign optimization
- Risk assessment
- Strategic decision-making

12. Submission Summary

Files Submitted:

- Hadoop MapReduce scripts and outputs
- Hive queries and results
- Spark notebooks (Core, SQL, ML, Streaming)
- Documentation (this report)
- Video explanation (15+ minutes)

Execution Environment:

- Hadoop & Hive: Local Ubuntu (WSL)
- Spark & Spark ML: Google Colab