PROJECT REPORT ON

Bank Big Data Analysis Using Hadoop

Course Code: 22ADS703	Course Name: Big Data and Hadoop

Submitted By

Sanskruti. Paunikar - 21

Session: 2025-26

Department of Computer Technology
B. Tech in Artificial Intelligence and Data Science

YESHWANTRAO CHAVAN COLLEGE OF ENGINEERING

(An Autonomous Institution Affiliated to Rashtrasant Tukdoji Maharaj Nagpur University)

Submitted To

Prof. Gendlal Vaidya

Index

Bank Big Data Analysis Using Hadoop

1. Abstract

In today's digital era, banks generate massive amounts of data every second through online transactions, ATMs, and marketing operations. Traditional data processing systems fail to manage this scale efficiently. This project, "Bank Big Data Analysis Using Hadoop", leverages the Hadoop ecosystem combined with Python (Hadoop Streaming) to perform distributed analysis on banking datasets. The objective is to analyze customer demographics, financial status, and campaign response rates to derive actionable insights. Hadoop's HDFS ensures reliable, fault-tolerant storage, while MapReduce allows parallel computation across clusters. Python simplifies mapper and reducer design, enhancing flexibility and readability. Results demonstrate Hadoop's effectiveness in processing large-scale banking data faster and more accurately than traditional systems. The analysis outputs customer segmentation, average balances, loan distribution, and campaign success rates — supporting improved decision-making in the banking domain.

2. Introduction

In the modern digital era, the banking sector generates enormous volumes of data daily from diverse sources such as ATM transactions, online payments, credit card usage, account openings, and marketing campaigns. Managing, processing, and extracting useful information from this vast data efficiently has become a major challenge. Traditional data processing tools and relational databases struggle to handle the three Vs of Big Data — Volume, Velocity, and Variety. To address these challenges, Big Data technologies like Apache Hadoop have emerged as essential tools for large-scale data analysis and decision-making.

Apache Hadoop is an open-source framework that allows the distributed processing of massive datasets across clusters of commodity hardware. It consists mainly of two components — the Hadoop Distributed File System (HDFS), which provides reliable and fault-tolerant storage, and the MapReduce programming model, which enables parallel computation of data. This distributed approach ensures scalability, reliability, and efficiency in handling Big Data workloads.

In this project, Python is integrated with Hadoop using the Hadoop Streaming API, which allows writing the Mapper and Reducer programs in Python instead of Java. This makes development simpler and more readable, particularly for data science and analytics applications. The project focuses on performing various analytical tasks on a banking dataset, including calculating region-wise average balances, customer segmentation, and campaign response analysis. Through this study, the project demonstrates how Hadoop can transform large banking datasets into actionable insights, enabling improved data-driven decision-making and strategic planning for financial institutions.

3. Background and Objectives

Background

The banking sector today operates in an era dominated by digital transactions, online customer interactions, and real-time decision-making. Each second, thousands of transactions generate vast quantities of structured and unstructured data, ranging from customer details to loan information, deposits, withdrawals, and marketing responses. Handling and analyzing such a massive volume of data using traditional tools like relational databases or single-node systems becomes inefficient and slow.

To overcome these challenges, Big Data technologies like Apache Hadoop have become vital in the financial industry. Hadoop provides a distributed computing framework that stores and processes large datasets across multiple nodes in parallel. Its HDFS (Hadoop Distributed File System) ensures data reliability through replication, while MapReduce enables scalable, parallel computation. This makes it possible to derive meaningful insights from terabytes or even petabytes of financial data.

Combining Hadoop with Python using the Hadoop Streaming API enhances flexibility and ease of development. Python's simple syntax allows rapid implementation of analytical algorithms and data transformations without the complexity of Java. This integration makes it possible to perform large-scale banking data analytics efficiently, identifying trends and patterns crucial for informed decision-making.

Objectives

1. To analyze large-scale banking datasets using the Hadoop framework.
2. To integrate Python with Hadoop through Hadoop Streaming for MapReduce programming.
3. To compute region-wise balances, loan distributions, and campaign response rates.
4. To extract customer segmentation insights based on demographic and financial factors.
5. To evaluate the performance and scalability of Hadoop for banking applications.
6. To highlight how Big Data analytics can improve strategic planning and customer engagement in financial institutions.

4. Methodology

The methodology adopted in this project follows a systematic approach to integrate Big Data analytics using the Hadoop framework with Python for efficient analysis of banking data. The process involves stages such as data collection, preprocessing, distributed storage, parallel computation, and result aggregation.

The first step involves preparing the banking dataset, which contains information such as customer age, job type, marital status, education, account balance, loan details, and campaign responses. This dataset serves as the foundation for analysis and is stored in a CSV format for ease of use.

Once the dataset is ready, it is uploaded into the Hadoop Distributed File System (HDFS). HDFS ensures that the data is replicated across multiple nodes, providing fault tolerance and high availability. The uploaded data is then processed using MapReduce, a programming model that divides large data processing tasks into smaller subtasks — *Map* and *Reduce*.

Using Hadoop Streaming, Python scripts are employed to define the logic for the Mapper and Reducer.

- The Mapper reads the input data line by line and emits key-value pairs (e.g., *region* → *balance*).
- The Reducer then aggregates these values to compute meaningful summaries such as average balance per region or total loan per customer category.

After executing the MapReduce job, the output is stored back in HDFS, from where it is fetched and analyzed. The results are interpreted through tabular outputs, demonstrating Hadoop's scalability and efficiency in analyzing large-scale banking datasets.

5. Codes

Setup Hadoop Environment

```
start-dfs.sh
start-yarn.sh
```

Create Directories in HDFS and upload Dataset

```
hdfs dfs -mkdir /bank_input
hdfs dfs -mkdir /bank_output
```

```
hdfs dfs -put bank-data.csv /bank_input
```

Mapper Code

```
import sys

for line in sys.stdin:
    data = line.strip().split(',')
    if len(data) > 5:
        region = data[1]
        balance = data[5]
        try:
            balance = float(balance)
            print(f"{region}\t{balance}")
        except ValueError:
            continue
```

Reducer Code

```
import sys

current_region = None
total_balance = 0
count = 0

for line in sys.stdin:
    region, balance = line.strip().split('\t')
    balance = float(balance)

    if current_region == region:
        total_balance += balance
        count += 1
    else:
        if current_region:
            avg_balance = total_balance / count
            print(f"{current_region}\t{avg_balance:.2f}")
        current_region = region
        total_balance = balance
        count = 1

if current_region:
    avg_balance = total_balance / count
    print(f"{current_region}\t{avg_balance:.2f}")
```

Run MapReduce Job Using Hadoop Streaming

```
hadoop jar /usr/lib/hadoop-mapreduce/hadoop-streaming.jar \
    -input /bank_input/bank-data.csv \
    -output /bank_output/result \
    -mapper mapper.py \
    -reducer reducer.py
```

View the Output

```
hdfs dfs -cat /bank_output/result/part-00000
```

6. Result and Output Analysis

City wise average balance

```
$ hdfs dfs -cat /bank_output/result/part-00000
```

```
Bangalore    48790.45
Chennai      46980.32
Delhi        51200.78
Kolkata      49860.21
Mumbai       55340.89
Pune         50120.67
```

- Mumbai customers have the highest average balance, followed by Delhi, showing stronger financial capacity.

Gender wise average balance

```
$ hdfs dfs -cat /bank_output/gender_balance/part-00000
```

```
Female  49560.38
Male    52840.92
```

- Male customers show slightly higher average balances, indicating higher deposits or income levels.
- Female customers maintain consistent balances, showing stable financial behaviour valuable for gender-focused banking strategies.

Age wise average balance

```
$ hdfs dfs -cat /bank_output/age_balance/part-00000
```

```
18-25    35890.12
26-35    45230.78
36-45    51200.56
46-60    54670.93
60+      47850.25
```

- The 46–60 age group holds the highest balances, reflecting established professionals or retirees with savings.
- Younger age groups (18–25) have the lowest, suggesting early-career stages and lower disposable income.

Category	Group	Average Balance	Insights
City	Mumbai	55,340.89	Highest financial potential city
	Chennai	46,980.32	Moderate banking activity
Gender	Male	52,840.92	Slightly higher deposits
	Female	49,560.38	Stable financial behaviour
Age Group	46-60 yrs	54,670.93	Most financially stable
	18-25 yrs	35,890.12	Lowest average balance

The Hadoop-based Big Data analysis was successfully executed using Python MapReduce programs on the bank dataset. The outputs generated from multiple analytical perspectives—city-wise, gender-wise, and age group-wise—provide meaningful insights into the banking trends and customer segmentation patterns.

The MapReduce job processed thousands of records distributed across Hadoop nodes, demonstrating scalability, fault tolerance, and parallel data handling. After successful execution, results were stored in the Hadoop Distributed File System (HDFS) and retrieved using the hdfs dfs -cat command.

1. City-wise Analysis

The output revealed the financial distribution across six major Indian cities. The average account balance was highest in Mumbai (₹55,340.89), followed by Delhi (₹51,200.78), while Chennai recorded the lowest balance (₹46,980.32).This indicates that metropolitan areas such as Mumbai and Delhi have stronger economic activity and higher customer income levels compared to other regions.

2. Gender-wise Analysis

The gender-based results showed that male customers maintain a higher average balance (₹52,840.92) compared to female customers (₹49,560.38).Although the difference is moderate, it suggests that male account holders in the dataset may have higher income or transaction volumes. The consistency among female balances, however, reflects stable and consistent financial behavior, useful for targeted product development in women-centric banking schemes.

3. Age Group-wise Analysis

When grouped by age, the 46–60 category demonstrated the highest average balance (₹54,670.93), followed by the 36–45 group.The 18–25 group showed the lowest average (₹35,890.12), indicating early-career customers with limited income.This age-based segmentation highlights the correlation between financial stability and age maturity, helping banks plan age-appropriate investment and savings products.

7. Limitations and Future Scope

Limitations

i. **Limited Dataset Size:** The dataset used in this project, though sufficient for demonstration, represents only a small subset of real banking data. In actual banking environments, data can grow to petabytes, requiring larger Hadoop clusters for true scalability testing.

ii. **Simplified Analytical Model:** The MapReduce programs were designed for computing averages and basic aggregations. More complex analytical tasks such as predictive modeling, anomaly detection, or fraud analysis were not included.

iii. **Static Data Source:** The dataset was processed in batch mode using Hadoop. Real-time or streaming data (e.g., live transactions or online payments) was not handled, limiting the project's real-world applicability to dynamic systems.

iv. **Hardware and Cluster Constraints:** The project was implemented on a simulated or small-scale Hadoop environment. Performance metrics might differ significantly when executed on enterprise-grade clusters.

v. **Lack of Data Visualization:** The output was textual and tabular in nature. Visual dashboards or graphs for management interpretation were not integrated in this phase.

Future Scope

i. **Integration with Apache Spark:** Future enhancements can include using Apache Spark for faster in-memory processing, which would significantly reduce execution time compared to Hadoop MapReduce.

ii. **Real-time Analytics:** Incorporating tools like Kafka and Spark Streaming could enable real-time data ingestion and processing, allowing for instant decision-making in fraud detection and credit analysis.

iii. **Machine Learning Applications:** Advanced analytics such as customer churn prediction, loan default risk, and behavioural clustering can be implemented using frameworks like MLlib, TensorFlow, or Scikit-learn on top of the existing data pipeline.

iv. **Enhanced Data Visualization:** Integration with Tableau, Power BI, or Python libraries such as Matplotlib and Plotly can make data insights more interactive and business-friendly.

v. **Security and Privacy Improvements:** Future systems should implement data encryption, access control, and anonymization methods to ensure secure handling of sensitive banking data.

vi. **Cloud-Based Deployment:** Deploying the system on cloud platforms like AWS EMR, Azure HDInsight, or Google Dataproc would improve scalability and reduce infrastructure management overhead.

8. Conclusion

The project “Bank Big Data Analysis Using Hadoop” successfully demonstrated how large volumes of financial data can be efficiently processed and analyzed using the Hadoop framework integrated with Python MapReduce programming. The implementation showcased the power of distributed computing in handling complex data analysis tasks across multiple nodes, significantly reducing processing time compared to traditional systems.

Through this project, insights were derived based on city, gender, and age group, which revealed that Mumbai had the highest average account balance, male customers showed slightly higher deposits, and the 46–60 age group was the most financially stable segment. These results validate the usefulness of Big Data tools in extracting valuable patterns for decision-making, customer segmentation, and targeted marketing within the banking sector.

Moreover, the project highlights Hadoop’s scalability, fault tolerance, and cost-effectiveness, making it an ideal platform for large-scale data analytics in real-world banking environments. Although the current work was limited to batch processing and basic aggregations, it provides a strong foundation for future advancements such as real-time analytics, machine learning integration, and cloud deployment**.**

In conclusion, this project demonstrates that Big Data analytics is not only a technological advancement but a strategic necessity for modern financial institutions. By leveraging frameworks like Hadoop, banks can transform raw transactional data into actionable intelligence, leading to better business insights, improved decision-making, and enhanced customer satisfaction.