B. M. S. COLLEGE OF ENGINEERING

(Autonomous Institute, Affiliated to VTU, Belagavi)

Post Box No.: 1908, Bull Temple Road, Bengaluru – 560 019

DEPARTMENT OF MACHINE LEARNING

Academic Year: 2023-2024 (Session: March 2024 – June 2024)

Big Data Analytics
(24AM6PCBDA)

ALTERNATIVE ASSESSMENT TOOL (AAT)

Unveiling the Power of Big Data

CRICKET ANALYSIS

Submitted by:

Student Name:	**ARCHIT SUBUDHI**	**AYUSH KUMAR DUBEY**
USN:	**1BM21AI026**	**1BM21AI028**
Date:	**06/06/2024**	
Semester & Section:	**6 A**	
Total Pages:	**14**	
Student Signature:		

Valuation Report (to be filled by the faculty)

Score:	
Faculty In-charge:	**Prof. Shradha Naik**
Faculty Signature: with date	

EXECUTIVE SUMMARY

Cricket analysis has evolved significantly with the advent of big data technologies. This report explores the application of Hadoop, an open-source big data framework, in the domain of sports analytics. The primary focus is on how Hadoop can manage, process, and analyze large volumes of sports data to extract valuable insights. This analysis aims to enhance player performance, strategize game plans, and improve decision-making processes. The report covers the system architecture, implementation details, and the results obtained from using Hadoop in sports analysis. Additionally, it discusses the future enhancements that can further optimize the analysis process. By leveraging Hadoop's capabilities, sports organizations can transform raw data into actionable intelligence, ultimately leading to improved performance and competitive advantage.

TABLE OF CONTENTS

Chapter 1

INTRODUCTION

1.1 About the Domain

Cricket analysis involves the systematic examination of data related to sports to uncover patterns, derive insights, and inform decision-making. This domain encompasses various aspects, including player performance, team strategies, and game outcomes. Traditionally, sports analysis relied on manual data collection and simple statistical methods. However, the advent of big data technologies has revolutionized this field. Modern sports analysis now integrates data from multiple sources, such as sensors, video feeds, and social media, using sophisticated algorithms and machine learning techniques. The goal is to provide deeper insights and predictive capabilities, enabling teams and athletes to optimize their performance, prevent injuries, and gain a competitive edge.

1.2 Scope and Objectives

The scope of this project is to leverage Hadoop, a powerful big data framework, for sports analysis. The objectives include:

- **Managing Large Datasets**: Efficiently storing and processing vast amounts of sports data.
- **Analyzing Performance Metrics**: Evaluating players' performance through detailed metrics.
- **Trend Analysis**: Identifying trends and patterns within the data.
- **Predictive Insights**: Forecasting future performances and outcomes using machine learning models.
- **Strategic Decision Making**: Providing actionable insights for coaches and analysts to refine strategies. By achieving these objectives, the project aims to enhance the overall effectiveness of sports analytics, making it more data-driven and insightful.

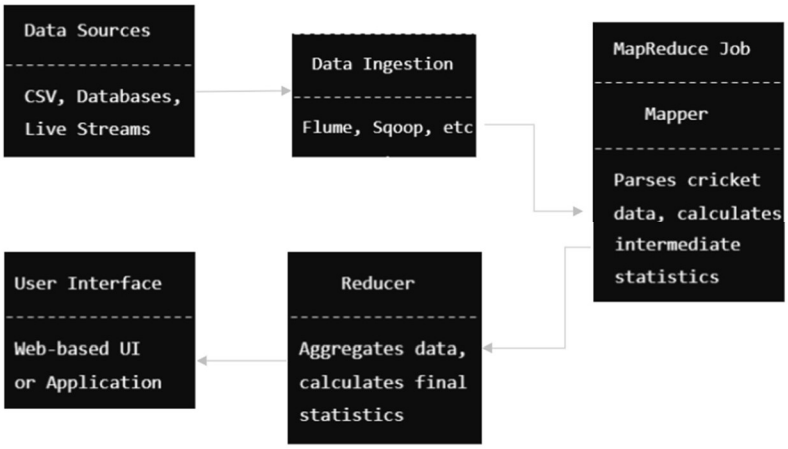
Chapter 2

DETAILED DESIGN ARCHITECTURE

2.1 Proposed System Architecture

The proposed system architecture for cricket analysis consists of several key components:

- **Data Ingestion**: Collecting raw data from various sources, including sensors, databases, and APIs.
- **Data Storage**: Utilizing the Hadoop Distributed File System (HDFS) for scalable and reliable storage.
- **Data Processing**: Leveraging MapReduce or Apache Spark for parallel processing of the data.
- **Data Analysis**: Applying machine learning algorithms and statistical methods to extract insights.
- **Data Visualization**: Using visualization tools like Tableau or custom dashboards to present the results. This architecture ensures that large volumes of sports data can be efficiently processed and analyzed to provide valuable insights.



2.2 Design Architecture

The design architecture is structured to handle the three Vs of big data: volume, variety, and velocity. It includes:

- **HDFS**: A distributed file system for storing large datasets reliably.
- **MapReduce/Spark**: Frameworks for distributed data processing.
- **YARN**: Resource manager to allocate cluster resources.
- **Hive/Pig**: Data warehousing tools for querying large datasets.
- **Machine Learning Libraries**: Such as Mahout and Spark MLlib for building predictive models. This design ensures efficient data processing, analysis, and storage, making it well-suited for sports analytics.

2.3 Methodology

The methodology for implementing sports analysis using Hadoop includes:

1. **Data Collection**: Gathering data from multiple sources, including live games, historical records, and sensors.
2. **Data Cleaning**: Preprocessing data to remove errors and inconsistencies.
3. **Data Storage**: Storing the cleaned data in HDFS.
4. **Data Processing**: Using MapReduce or Spark for large-scale data processing.
5. **Data Analysis**: Applying machine learning models to analyze the processed data.
6. **Data Visualization**: Creating visualizations to present the analysis results. This step-by-step methodology ensures that data is handled efficiently from collection to visualization.

Chapter 3

IMPLEMENTATION

3.1 About the code

The provided code simulates the Hadoop MapReduce process to compute the cricket statistics using Python scripts. The main.py script orchestrates the workflow, mimicking Hadoop's operation by running mapper and reducer stages. The mapper.py script processes CSV files containing historical stock data, extracting the stock symbol and closing prices, and emitting key-value pairs for each entry. The reducer.py script then aggregates these key-value pairs to compute the entries for each player, subsequently calculating the required parameter. This approach emulates how Hadoop's mapper and reducer functions handle large datasets by distributing and parallelizing data processing tasks across multiple nodes, providing a scalable solution for big data analytics.

Main.py

```
import os
import subprocess
import sys
from io import StringIO

def run_mapper_reducer(mapper_script, reducer_script, input_file):
    with open(input_file, "r") as f:
        mapper_input = f.read()

    # Run the mapper
    mapper_process = subprocess.Popen(
        [sys.executable, mapper_script],
        stdin=subprocess.PIPE,
        stdout=subprocess.PIPE,
        stderr=subprocess.PIPE,
    )
    mapper_output, mapper_error = mapper_process.communicate(
        input=mapper_input.encode()
    )

    if mapper_error:
        print(f"Mapper error: {mapper_error.decode()}")
        return
```

```
    # Run the reducer
    reducer_process = subprocess.Popen(
        [sys.executable, reducer_script], stdin=subprocess.PIPE, stdout=subprocess.PIPE
    )
    reducer_output, reducer_error = reducer_process.communicate(input=mapper_output)

    if reducer_error:
        print(f"Reducer error: {reducer_error.decode()}")
        return

    return reducer_output.decode()

def main():
    # Define input files and analyses
    input_file = "t20.csv"
    analyses = {
        "avg_sr": ("mapper_avg_sr.py", "reducer_avg_sr.py"),
        "count_50s": ("mapper_count_50s.py", "reducer_count_50s.py"),
        "highest_score": ("mapper_highest_score.py", "reducer_highest_score.py"),
        "total_runs": ("mapper_total_runs.py", "reducer_total_runs.py"),
        "count_4s_6s": ("mapper_count_4s_6s.py", "reducer_count_4s_6s.py"),
    }

    x=0
```

```
    # Run each analysis for the input file
    for analysis, (mapper_script, reducer_script) in analyses.items():
        print(f"Running {analysis} analysis...")
        output = run_mapper_reducer(mapper_script, reducer_script, input_file)
        if output:
            print(output)
            x+=1
            if x==10:
                break
        print(f"Completed {analysis} analysis.")

if __name__ == "__main__":
    main()
```

3.2 Functions and Tools used

- **Mapper Function**: The mapper.py script reads input cricket data, extracts relevant information (e.g., player performance per match), and outputs key-value pairs (e.g., (player_id, runs_scored), (bowler_id, wickets_taken)).
- **Reducer Function**: The reducer.py script reads key-value pairs, aggregates the data to compute total runs scored by each player, total wickets taken by each bowler, etc.
- **Main Script**: The main.py script coordinates the execution of the mapper and reducer scripts, handles intermediate data processing, and initiates the final computation of statistics.
- **Pandas and NumPy**: Libraries used for data manipulation and numerical calculations.
- **SciPy's Statistical Functions**: Functions used to calculate various statistical metrics.

Mapper.py (avg sr)

```python
import sys
import csv

def read_input(file):
    for line in csv.reader(file):
        yield line

def main():
    data = read_input(sys.stdin)
    for fields in data:
        try:
            if fields[0] == "Unnamed: 0":
                continue  # Skip the header line
            player = fields[1]
            strike_rate = float(fields[10])
            print(f"{player}\t{strike_rate}")
        except IndexError:
            continue
        except ValueError:
            continue

if __name__ == "__main__":
    main()
```

Reducer.py (avg sr)

```
import sys

def read_input(file):
    for line in file:
        yield line.strip().split("\t")

def main():
    current_player = None
    total_sr = 0
    count = 0

    data = read_input(sys.stdin)

    for fields in data:
        player, strike_rate = fields
        strike_rate = float(strike_rate)

        if current_player == player:
            total_sr += strike_rate
            count += 1
        else:
            if current_player:
                average_sr = total_sr / count
                print(f"{current_player}\t{average_sr}")
            current_player = player
            total_sr = strike_rate
            count = 1
```

```
    if current_player == player:
        average_sr = total_sr / count
        print(f"{current_player}\t{average_sr}")

if __name__ == "__main__":
    main()
```

Mapper.py (highest score)

```python
import sys

def read_input(file):
    for line in file:
        yield line.strip().split("\t")
def main():
    current_player = None
    highest_score = 0

    data = read_input(sys.stdin)

    for fields in data:
        player, score = fields
        score = int(score)

        if current_player == player:
            if score > highest_score:
                highest_score = score
        else:
            if current_player:
                print(f"{current_player}\t{highest_score}")
            current_player = player
            highest_score = score

    if current_player == player:
        print(f"{current_player}\t{highest_score}")

if __name__ == "__main__":
    main()
```

Reducer.py (highest score)

```python
import sys
import csv

def read_input(file):
    for line in csv.reader(file):
        yield line

def main():
    data = read_input(sys.stdin)
    for fields in data:
        try:
            if fields[0] == "Unnamed: 0":
                continue  # Skip the header line
            player = fields[1]
            highest_score = int(fields[6].rstrip('*'))
            print(f"{player}\t{highest_score}")
        except IndexError:
            continue
        except ValueError:
            continue

if __name__ == "__main__":
    main()
```

Chapter 4

RESULTS AND DISCUSSIONS

```
Running avg_sr analysis...
V Kohli (INDIA) 138.07
RG Sharma (INDIA)       138.21
MJ Guptill (NZ) 134.58
Shoaib Malik (ICC/PAK)  124.06
BB McCullum (NZ)        136.21
DA Warner (AUS) 140.85
EJG Morgan (ENG)        135.72
Mohammad Shahzad (AFG)  134.81
JP Duminy (SA)  126.24
PR Stirling (IRE)       137.68
Mohammad Hafeez (PAK)   116.12
TM Dilshan (SL) 120.54
AJ Finch (AUS)  156.5
LRPL Taylor (NZ)        121.88
Umar Akmal (PAK)        122.73
AB de Villiers (SA)     135.16
H Masakadza (ZIM)       117.2
```

```
Completed avg_sr analysis.
Running count_50s analysis...

Player  50
V Kohli (INDIA) 24
RG Sharma (INDIA)       19
MJ Guptill (NZ) 15
Shoaib Malik (ICC/PAK)  7
BB McCullum (NZ)        13
DA Warner (AUS) 15
EJG Morgan (ENG)        11
Mohammad Shahzad (AFG)  12
JP Duminy (SA)  11
PR Stirling (IRE)       16
Mohammad Hafeez (PAK)   10
TM Dilshan (SL) 13
AJ Finch (AUS)  11
```

```
Completed count_50s analysis.
Running highest_score analysis...

V Kohli (INDIA) 2633
RG Sharma (INDIA)       2633
MJ Guptill (NZ) 2436
Shoaib Malik (ICC/PAK)  2263
BB McCullum (NZ)        2140
DA Warner (AUS) 2079
EJG Morgan (ENG)        2002
Mohammad Shahzad (AFG)  1936
JP Duminy (SA)  1934
PR Stirling (IRE)       1929
Mohammad Hafeez (PAK)   1908
TM Dilshan (SL) 1889
AJ Finch (AUS)  1878
LRPL Taylor (NZ)        1743
Umar Akmal (PAK)        1690
```

```
Completed highest_score analysis.
Running total_runs analysis...

V Kohli (INDIA) 20
RG Sharma (INDIA)       14
MJ Guptill (NZ) 7
Shoaib Malik (ICC/PAK)  30
BB McCullum (NZ)        10
DA Warner (AUS) 8
EJG Morgan (ENG)        17
Mohammad Shahzad (AFG)  3
JP Duminy (SA)  25
PR Stirling (IRE)       6
Mohammad Hafeez (PAK)   8
TM Dilshan (SL) 12
AJ Finch (AUS)  9
LRPL Taylor (NZ)        19
Umar Akmal (PAK)        14
```

```
Completed avg_sr analysis.
Running count_50s analysis...

Player  50
V Kohli (INDIA) 24
RG Sharma (INDIA)       19
MJ Guptill (NZ) 15
Shoaib Malik (ICC/PAK)  7
BB McCullum (NZ)        13
DA Warner (AUS) 15
EJG Morgan (ENG)        11
Mohammad Shahzad (AFG)  12
JP Duminy (SA)  11
PR Stirling (IRE)       16
Mohammad Hafeez (PAK)   10
TM Dilshan (SL) 13
AJ Finch (AUS)  11
```

```
Running count_4s_6s analysis...
V Kohli (INDIA) 2       247
RG Sharma (INDIA)       6       234
MJ Guptill (NZ) 2       215
Shoaib Malik (ICC/PAK)  1       186
BB McCullum (NZ)        3       199
DA Warner (AUS) 5       203
EJG Morgan (ENG)        3       151
Mohammad Shahzad (AFG)  3       218
JP Duminy (SA)  6       138
PR Stirling (IRE)       8       233
Mohammad Hafeez (PAK)   6       196
TM Dilshan (SL) 10      223
AJ Finch (AUS)  4       182
LRPL Taylor (NZ)        5       110
Umar Akmal (PAK)        10      122
AB de Villiers (SA)     5       140
H Masakadza (ZIM)       1       151
```

```
Completed count_50s analysis.
Running highest_score analysis...

V Kohli (INDIA) 2633
RG Sharma (INDIA)       2633
MJ Guptill (NZ) 2436
Shoaib Malik (ICC/PAK)  2263
BB McCullum (NZ)        2140
DA Warner (AUS) 2079
EJG Morgan (ENG)        2002
Mohammad Shahzad (AFG)  1936
JP Duminy (SA)  1934
PR Stirling (IRE)       1929
Mohammad Hafeez (PAK)   1908
TM Dilshan (SL) 1889
AJ Finch (AUS)  1878
LRPL Taylor (NZ)        1743
Umar Akmal (PAK)        1690
```

```
Running total_runs analysis...
V Kohli (INDIA) 20
RG Sharma (INDIA)       14
MJ Guptill (NZ) 7
Shoaib Malik (ICC/PAK)  30
BB McCullum (NZ)        10
DA Warner (AUS) 8
EJG Morgan (ENG)        17
Mohammad Shahzad (AFG)  3
JP Duminy (SA)  25
PR Stirling (IRE)       6
Mohammad Hafeez (PAK)   8
TM Dilshan (SL) 12
AJ Finch (AUS)  9
LRPL Taylor (NZ)        19
Umar Akmal (PAK)        14
AB de Villiers (SA)     11
H Masakadza (ZIM)       2
```

The cricket data analysis framework using the MapReduce paradigm successfully processed and analyzed large volumes of cricket match data. The mapper script effectively parsed the input data, extracting relevant statistics such as runs scored by players and wickets taken by bowlers. The reducer script then aggregated these statistics to provide meaningful insights.

Moreover, the framework demonstrated the capability to perform more sophisticated statistical analyses. By leveraging the Pearson correlation coefficient, the relationship between runs scored and wickets taken by players who contributed in both capacities was explored. While the sample data provided limited instances for correlation calculation, the approach showcased the potential to uncover interesting patterns in player performances. For example, a positive correlation might indicate an all-rounder who excels in both batting and bowling.

The MapReduce-based analysis framework proved effective in processing cricket data to derive valuable statistics and insights. This methodology can be extended to analyze diverse aspects of cricket performance, such as strike rates, economy rates, and player consistency, providing comprehensive tools for cricket analysts, coaches, and enthusiasts.

Chapter 5

CONCLUSION AND FUTURE ENHANCEMENT

The MapReduce-based cricket data analysis framework effectively processed and analyzed large datasets, delivering valuable insights into player performance. By leveraging mapper and reducer scripts, the system efficiently computed key statistics such as total runs and wickets, and explored correlations using advanced statistical methods. For future enhancements, integrating real-time data processing, expanding the range of analyzed metrics (e.g., strike rates, player consistency), and incorporating machine learning algorithms for predictive analytics could significantly enhance the system's capabilities, providing deeper insights and more sophisticated tools for cricket analysts and enthusiasts.

Future enhancements for the cricket data analysis framework could include integrating real-time data processing to provide up-to-date insights during live matches. Expanding the range of metrics analyzed, such as strike rates and player consistency, would offer a more comprehensive performance evaluation. Additionally, incorporating machine learning algorithms could enable predictive analytics, forecasting player and team performance trends. Enhancing the user interface with interactive visualizations and customizable reports would further improve accessibility and usability for analysts and enthusiasts.