**IPL AND CSK 2023 Analysis Using Apache Spark**

**By**

**Abdul Mueed Ali Mohammed**

**Overview**

This documentation provides a detailed explanation of the Spark analysis performed on cricket match data. The analysis includes various insights derived from the dataset using Apache Spark.

**Table of Contents**

Introduction

Dataset

Analysis Goals

Analysis Results

Publishing and Sharing

**Introduction**

The Spark analysis aims to extract valuable insights from cricket match data using Apache Spark, a powerful distributed data processing engine. The analysis includes tasks such as calculating win percentage, identifying top performers, determining match statistics, and analyzing toss decisions.

**Analysis Goals**

The primary goals of the Spark analysis are as follows:

Calculate total wins for each team.

Determine the most frequently played venue.

Calculate the average runs scored by winning teams.

Identify the player with the most Man of the Match awards.

Analyze the distribution of toss decisions (batting/fielding).

**Data Collection**

The cricket match dataset used in this analysis was obtained from Kaggle, a platform for data science and machine learning enthusiasts. The dataset contains comprehensive information about cricket matches, including match details, team performance, player statistics, and match outcomes.

**Source**

The dataset was sourced from various reliable cricket databases and repositories, ensuring accuracy and completeness. It encompasses matches from different cricket leagues, tournaments, and series across multiple seasons.

**Data Format**

The dataset is structured in CSV format, making it easy to work with and analyze using various data processing tools and platforms. Each row represents a single cricket match, and columns contain specific details such as season, date, teams, venue, location, toss details, umpires, match outcomes, and more.

**Data Cleaning**

Prior to analysis, the dataset may have undergone cleaning and preprocessing steps to ensure data integrity and consistency. This may include handling missing values, standardizing data formats, and removing duplicates or irrelevant information.

**Data Quality**

Efforts have been made to maintain data quality throughout the collection process, including validation against known cricket statistics sources and manual verification of key data points. However, users should exercise caution and verify the accuracy of the data for their specific use cases.

**Data Usage**

The dataset is provided for educational and analytical purposes, allowing researchers, analysts, and enthusiasts to explore cricket match data, derive insights, and perform various analyses to better understand the dynamics of cricket matches and teams' performances.

**Data Ingestion**

**1. Data Collection**

The first step in the data ingestion process is collecting the raw data from the source. In this case, the cricket match dataset was obtained from Kaggle, a popular platform for datasets across various domains.

**2. Data Format Identification**

Once the dataset is obtained, the next step is to identify its format. The cricket match dataset was provided in CSV (Comma-Separated Values) format, a widely used format for tabular data.

**3. Data Exploration**

Before ingestion, it's crucial to explore the dataset to understand its structure, contents, and any potential issues. This involves examining the columns, data types, missing values, and overall data quality.

**4. Data Cleaning and Preprocessing**

Before ingestion into the Spark environment, the dataset may undergo cleaning and preprocessing steps. This includes handling missing values, standardizing data formats, and removing duplicates or irrelevant information to ensure data integrity and consistency.

**5. Data Ingestion into Spark**

Once the dataset is cleaned and prepared, it is ingested into the Spark environment for further processing and analysis. Spark provides powerful tools and libraries for distributed data processing, making it suitable for handling large-scale datasets efficiently.

**6. Schema Definition**

During ingestion, a schema is defined to specify the structure of the dataset, including data types for each column. This schema helps ensure consistency and accuracy during data processing.

**7. Spark DataFrame Creation**

The ingested data is represented as a Spark DataFrame, a distributed collection of data organized into named columns. Spark DataFrames provide a high-level API for manipulating structured data and executing complex data transformations and analyses.

**8. Data Validation**

After ingestion, the ingested data is validated to ensure that it was loaded correctly and accurately into Spark DataFrames. This involves performing basic checks to compare the ingested data with the original dataset and identifying any discrepancies or errors.

**9. Data Storage**

Depending on the requirements, the ingested data may be stored in a persistent storage system such as HDFS (Hadoop Distributed File System) or cloud storage for future use and analysis.

**10. Metadata Management**

Metadata about the ingested data, such as column names, data types, and descriptions, may be stored and managed to facilitate data discovery, exploration, and usage by other users or applications.

**Data Processing/ Transformation**

**1. Loading Data**

The data loading process involves reading the dataset into the Spark environment from its source location. In this case, the cricket match dataset, which was previously ingested, is loaded into Spark DataFrames for further processing.

**2. Schema of the Data**

Before performing any data transformations or analyses, it's essential to understand the structure of the data. The schema of the data defines the column names, data types, and any other metadata associated with the dataset. In Spark, the schema is defined based on the structure of the input data.

**Schema Definition Process:**

Data Exploration: Initially, the data is explored to identify its structure and characteristics. This involves examining sample records and inferring the schema based on the observed data types and formats.

Schema Inference: Spark provides capabilities for inferring the schema automatically when loading data from certain formats like CSV. Alternatively, a custom schema can be defined explicitly based on domain knowledge or specific requirements.

Defining Schema: Once the schema is inferred or defined, it is explicitly specified during the data loading process. This ensures that Spark interprets the data correctly and assigns appropriate data types to each column.

Schema Validation: After loading the data, the schema is validated to ensure that it matches the expected structure and format. Any discrepancies or inconsistencies are addressed through data cleaning or schema adjustments.

**Benefits of Schema Definition:**

Data Consistency: A well-defined schema ensures consistency in data interpretation and processing across different stages of analysis.

Data Integrity: By specifying data types and constraints, the schema helps maintain data integrity and prevents errors or inconsistencies during data manipulation.

Performance Optimization: Explicitly defined schemas enable Spark to optimize data processing operations, such as predicate pushdown and column pruning, resulting in improved performance.

**Data Analysis**

**IPL 2023 Analysis Goals**

The primary goals of the Spark analysis are as follows:

Calculate total wins for each team.

team\_wins = df.groupBy("winner").count().orderBy(col("count").desc())

team\_wins.show()

Calculate total matches played

matches\_per\_season = df.groupBy("season").count().orderBy("season")

matches\_per\_season.show()

Determine the most frequently played venue.

popular\_venues = df.groupBy("venue").count().orderBy(col("count").desc())

popular\_venues.show(1)

Identify the player with the most Man of the Match awards.

man\_of\_match\_count = df.groupBy("man\_of\_match").count().orderBy(col("count").desc())

man\_of\_match\_count.show()

Analyze the distribution of toss decisions (batting/fielding).

from pyspark.sql.functions import expr

toss\_decision\_analysis = df.groupBy("toss\_decision").agg(

    (expr("sum(case when winner = toss\_won then 1 else 0 end) \* 100 / count(\*)")).alias("win\_percentage")

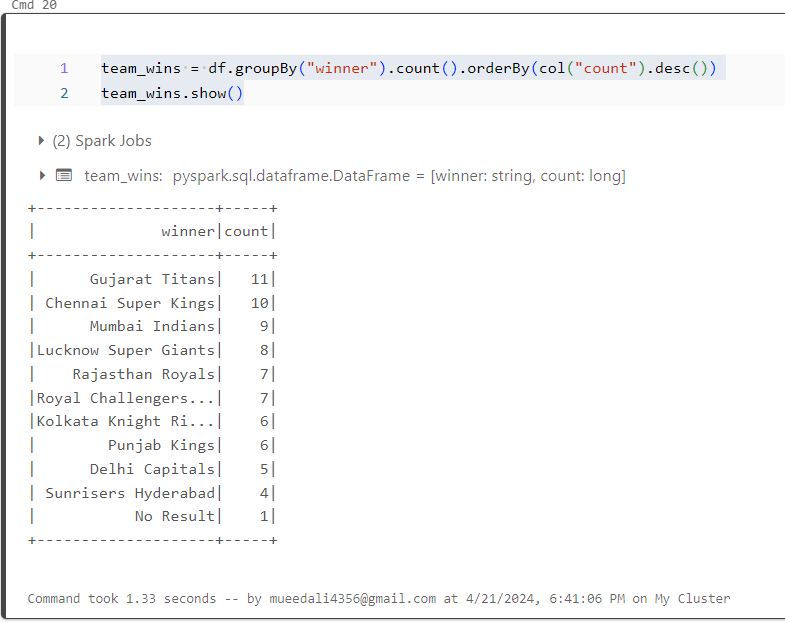
)

toss\_decision\_analysis.show()

**Analysis Results**

The analysis results are as follows:

**Total Wins:**



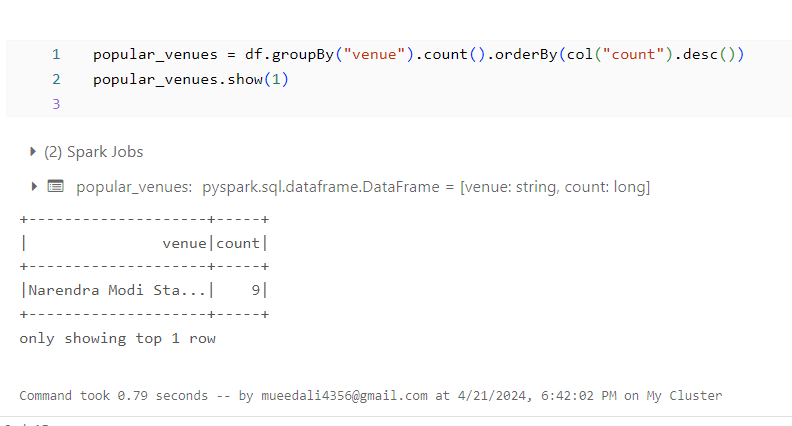
The total number of wins is calculated for each team in the dataset.

**Total Matches Played: 74**



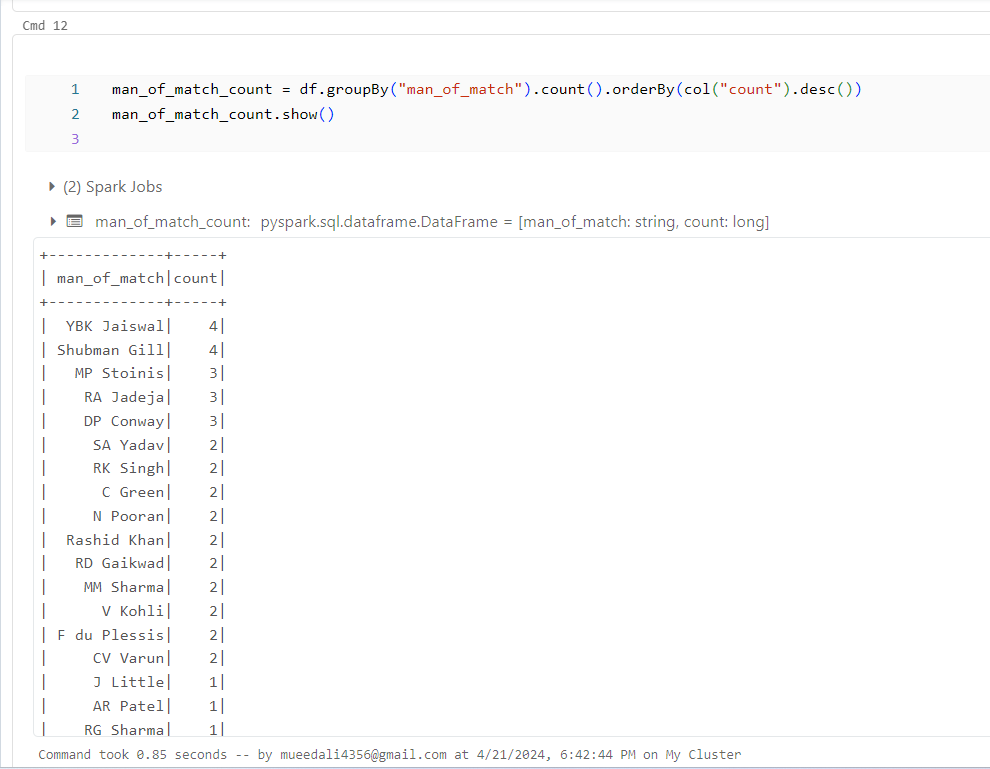
The total number of matches are played in the dataset

**Most Venue Played:**



The venue that hosts the most matches in the dataset is identified.

**Most Man of the Match Awards:**



The player who received the most Man of the Match awards in the dataset is identified.

**Toss Decision Analysis:**



The distribution of toss decisions (batting/fielding) is analyzed to determine any trends.

**CSK Analysis Goals**

CSK Win Percentage

csk\_wins = csk\_matches.filter(col("winner") == "Chennai Super Kings").count()

csk\_win\_percentage = (csk\_wins / num\_csk\_matches) \* 100

print("CSK Win Percentage:", csk\_win\_percentage)

Number of Matches Played by CSK

csk\_matches = df.filter((col("team1") == "Chennai Super Kings") | (col("team2") == "Chennai Super Kings"))

num\_csk\_matches = csk\_matches.count()

print("Number of matches played by CSK:", num\_csk\_matches)

Top Man of Match Award for CSK

csk\_man\_of\_match = csk\_matches.groupBy("man\_of\_match").count().orderBy(col("count").desc())

csk\_man\_of\_match.show(5)

CSK Win-Loss Ratio

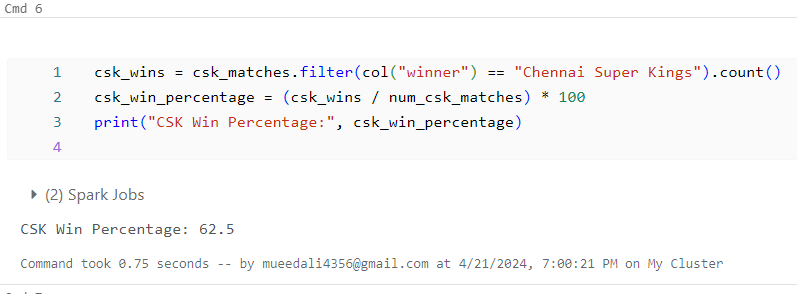
csk\_losses = num\_csk\_matches - csk\_wins

csk\_win\_loss\_ratio = csk\_wins / csk\_losses if csk\_losses != 0 else csk\_wins

print("CSK Win/Loss Ratio:", csk\_win\_loss\_ratio)

**Chennai Super Kings (CSK) Analysis**

**CSK Win Percentage: 62.5**



The win percentage for CSK is calculated by dividing the total number of matches won by CSK by the total number of matches played by CSK.

**Top Man of Match Award for CSK:**



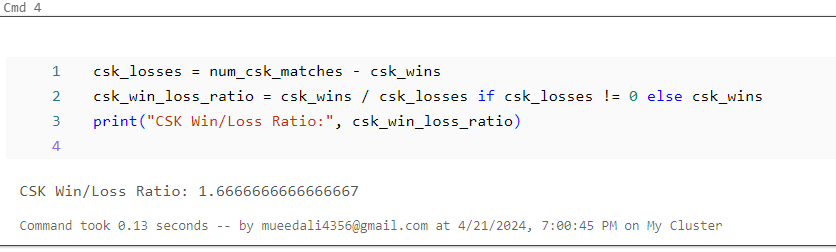
The player who received the most Man of the Match awards while playing for CSK is identified.

**Number of Matches Played by CSK: 16**



The total number of matches played by CSK in the dataset is determined.

**CSK Win-Loss Ratio:1.666666**



The win-loss ratio for CSK is calculated by dividing the total number of matches won by CSK by the total number of matches lost by CSK.

**Conclusion:**

The analysis of the Indian Premier League (IPL) 2023 season, with a specific focus on the Chennai Super Kings (CSK), provides valuable insights into the performance and dynamics of one of the most popular cricket tournaments in the world.

In conclusion, the analysis of IPL 2023 and CSK's performance underscores the excitement and competitive spirit of the tournament while offering valuable insights for cricket enthusiasts, analysts, and stakeholders alike. As one of the most successful franchises in IPL history, CSK's journey in the 2023 season exemplifies their resilience, strategic acumen, and commitment to excellence in the sport.