International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-9 Issue-2, December, 2019

Indian Premier League Dataset Analytics using Hadoop-Hive

Sapna S., Sandhya S.

Abstract: ***Big Data is a term used to represent huge volume of both unstructured and structured data which cannot be processed by the traditional data processing techniques. This data is too huge, grows exponentially and doesn't fit into the structure of the traditional database systems. Analyzing Big Data is a very challenging task since it involves the processing of huge amount of data. As the industry or its business grows, the data related to the industries also tend to grow on a larger scale. Prominent data analysis tools are required to analyze the data in order to gain value out of it. Hadoop is a sought-after open source framework that uses MapReduce techniques to store and process huge datasets. However, the programs written using MapReduce techniques are not flexible and also require maintenance. This problem is overcome by making use of HiveQL. In order to execute queries in HiveQL, the platform required is Hive. It is an open-source data warehousing set-up built on Hadoop. HiveQL queries are compiled into MapReduce jobs that are executed utilizing Hadoop. In this paper we have analyzed the Indian Premier League dataset using HiveQL and compared its execution time with that of traditional SQL queries. It was found that the HiveQL provided better performance with larger dataset while SQL performed better with smaller datasets.***

Keywords: Big Data, Hadoop, Hive, IPL

I. INTRODUCTION

Big data is a well known term used to portray the exponential development, openness and availability of data, both unstructured and structured. This huge amount of data originates from various sources: climatic information from the sensors, various social media sites, computerized pictures and recordings, transaction records, GPS signals and many more. This data can be generally termed as big data. The five unique characteristics of Big Data are Volume, Veracity, Velocity, Value and Variety [1]. Volume refers to the humungous amount of data coming from various sources. Veracity refers to the exactness or the reliability of the information. Velocity refers to the speed at which the information in produced and further broken down. Value refers to the information or meaning gained from the existing data. Variety refers to the various types of data. As and when the data is generated, it has to be processed and analyzed in order to extract meaningful information from the data, else the data generated is useless. Since more than 80% of the data produced is unstructured, traditional database systems fail to analyze such kind of data. Big Data analytics overcomes this disadvantage and helps in processing and analyzing such unstructured or semi-structured data in a short amount of time. Built-in tools are used for this purpose. Big Data Analytics internally uses Hadoop framework for processing the data. Hadoop framework consists of MapReduce and Hadoop Distributed File System (HDFS) [2, 14]. MapReduce framework provides a way of processing the data in a distributed manner and HDFS provides a way to store the huge amount of data. Hadoop framework also provides various tools like Hive, Pig, Cassandra etc for Big Data analytics. Pig is used to analyze huge data sets. Pig enables the programmers to concentrate more on analyzing the data rather than writing MapReduce programs. PigLatin is the language used to write pig scripts. Hive is a data warehousing tool used to manage or organize the data. Hive tool provides a way to write Structured Query Language (SQL) like scripts to perform MapReduce operations. The programming language is termed as HiveQL. Cassandra is a tool used to store structured data. In this paper, we have used Hive tool with the Hadoop Framework in order to process the Indian Premier League (IPL) dataset [12]. HiveQL is used to write SQL like queries to process the data. These queries are internally converted to MapReduce operations to process and analyze the data. SQL is mostly suited for smaller datasets whereas HiveQL is most suited for larger datasets. This paper is organized as follows: Section II presents the Literature Review of the related works in the field of Big Data, Section III deals with the architecture of the proposed system, Section IV shows the results and Section V depicts the performance analysis. Section VI provides the conclusion.

II. LITERATURE SURVEY

J. Mehta et al. [3] proposed a system which used Hadoop-Hive to analyze the stock data of 14 years in order to identify the companies whose volumes are traded high in past years. For this purpose, NYSE Historical Dataset was used. The stock and shares data analysis could give the users a view on where to invest based on its performance in the market in the past years. This data analysis could be useful in the field of Business Intelligence.

Revised Manuscript Received on December 30, 2019.
* Correspondence Author
Sapna S.*, Department of Information Science and Engineering, NMAM Institute of Technology, Nitte, Karkala (Karnataka) India. E-mail: sapna_s@nitte.edu.in
Sandhya S., Department of Information Science and Engineering, NMAM Institute of Technology, Nitte, Karkala (Karnataka) India. E-mail: sandhyas@nitte.edu.in

D. Lakshmi et al. [4] proposed a system that used Hadoop Hive to analyze the Movie Lens Dataset in order to understand the relationship between user profiles and ratings. HiveQL is preferred over Map-Reduce since the latter is time consuming and complex by nature. The ratings was obtained based on many factors which included occupation, age and gender which would give a view on the best target population for any given movie. Also, T. Mehta et al. [7] analyzed the Movie Lens Dataset using Hive and R to check how the age of an individual affects the ratings given to a movie. The study concluded that Hadoop with R can be used to solve different types of problems with huge datasets efficiently.

S.S. Mazumdar et al. [5] proposed a framework in which multiple datasets like movie dataset, weather dataset and healthcare dataset were used for the analysis purpose. Apache Hive was used for faster query processing. Also, machine learning techniques were used on the result sets for finding out the analytics of the dataset. It also provided a way to use machine learning with big data to analyze data in a simpler way.

S. K. Pushpa et al. [6] studied the Airport Dataset using Hadoop-Hive. This study focused on the common necessities of an airport like finding the flights that fly without any intermediate stops, different actively operative airports from a particular country, country which supports maximum flights and so on. The processing was done in a distributed environment and thus resulted in smaller response time for huge dataset.

P. Beri et al. [8] analyzed the data generated from social networking sites like Twitter and Facebook on how the data is generated by these systems and how the storage of such kind of data is managed to support faster access to data with low response time and high accessibility.

A. U. Abdullahi et al. [9] analyzed the Oceanographic data and Meteorological data on Hive. The analysis was carried out on the performance of the system with indexing and no indexing for queries involving ‘select’ and ‘where’ clauses, ‘select’ and ‘join’ clauses, and ‘select’ and ‘group by’ clauses. The comparison was made for Hive Execution with respect to traditional SQL based systems. It justified the efficiency of the Hive based systems in query processing for large datasets.

B. Jain et al. [10] proposed a method for optimizing the queries written in HiveQL on the Hive framework. The optimization was brought about by using Join ordering and Indexing on the existing large dataset. The analysis was performed on the Movie Lens Dataset and provided future scope for optimization of queries in Hive.

P. Jain et al. [11] proposed a system for analysis of customer related statistics like customer complaints and reviews using Pig and Hive to provide them with the appropriate service based on the issue raised. The comparison was made between the response times for Hive and Pig and the author concluded that, for the given dataset, Hive worked better than Pig.

III. PROPOSED SYSTEM

A. Architecture

Fig 1 presents the architecture of the proposed system. The IPL Dataset is given as input to the system having the Hive Component. Hive Component internally consists of Driver, Metastore, Query Compiler, Thrift Server and ODBC/JDBC Driver [13]. Driver is used to execute the Hive Queries. Metastore contains the details on the metadata of the tables along with the Hadoop Distributed File System Mappings. Query Compiler is used to test the correctness of the queries written using HiveQL. Hive internally converts these queries into Map-Reduce Jobs and they are executed on the interpreter.

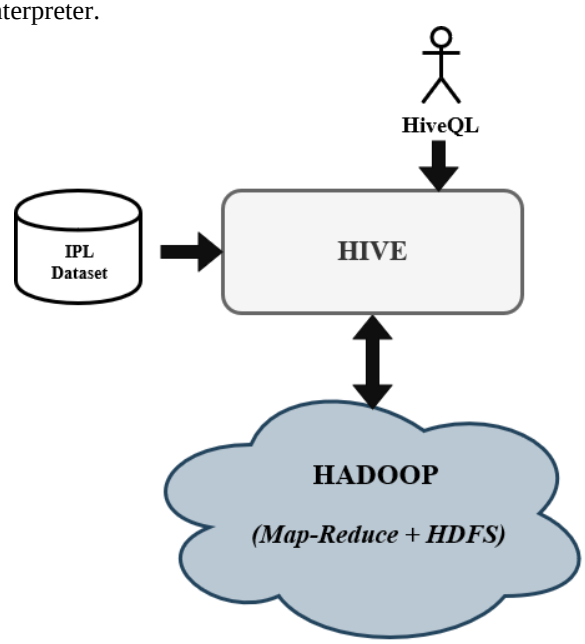

Fig. 1. Architecture of the proposed system

The communication between Hive and Hadoop Framework takes place using the following steps:

1) The user uses the Hive Web Interface or Command Line Interface to send query to the Driver for execution.
2) The driver uses the compiler which parses the query to check for any errors in the syntax and the query plan.
3) The compiler forwards the metadata request to the Metastore.
4) Metastore sends the metadata in response to the Metadata request.
5) The compiler verifies if the metadata is as requested and in case of success, it resends the plan to driver for execution. This completes the parsing phase of the compiler.
6) On receiving the query plan, the driver forwards it to the execution engine.
7) The query processing is done internally as a Map-Reduce Task. In the meantime, the execution engine also performs metadata operations on the Metastore.
8) Once the result of execution is made available by the execution engine, the result is sent to the driver.
9) The Driver sends the result to the Hive Interface.

B. Dataset

Indian Premier League (IPL) is a Twenty20-Cricket League in India held during the month of March to May every year where 8 different teams representing 8 different cities of India contest with each other for the title. The IPL dataset [12] was taken from Kaggle.

It consists of information related to every match and every delivery in a given match for all the seasons from 2008 to 2019. In this paper, we have considered two csv files namely matches.csv and deliveries.csv representing the match details and delivery details respectively. Deliveries consist of about 1,79,079 rows and matches consist of 757 rows. Table I and Table II show the description of the attributes of Matches and Deliveries Table respectively.

Table- I: Attributes of Matches Table

Attribute	Description
Id	Distinct ID assigned to a Match
Season	Year in which the IPL Match was conducted
City	The city in which the match was held
Date	Date of the Match in dd/mm/yyyy format
Team 1	First Contesting Team
Team 2	Second Contesting Team
Toss Winner	Winner of the Toss
Toss Decision	Decision taken on whether to bat or field first by the winner of Toss
Result	Normal/Draw
Winner	Winning team for the Match
Win by Runs	Number of runs by which the team won
Win by Wickets	Number of wickets by which the team won
Player of the Match	The best performer of the Match
Venue	Location of the Match
Umpire 1	First Umpire for the match
Umpire 2	Second Umpire for the match
Umpire 3	Third Umpire for the match

IV. RESULTS AND DISCUSSION

This section briefly describes the queries written using HiveQL for creation of tables and extraction of meaningful information from the tables.

A. Table Creation and Population

Size of the IPL dataset on the local system is around 30 MB. In order to extract the meaning out of the data, first we need to create the tables and load the data onto the tables. MATCHES table and DELIVERIES table is created using HiveQL. Fig 2 shows the execution of the HiveQL Queries for creation and population of the MATCHES table. Fig 3 shows the execution of the HiveQL Queries for creation and population of the DELIVERIES Table.

Table- II: Attributes of Deliveries Table

Attribute	Description
Match_id	Distinct ID assigned to a Match (referred from Matches Table)
Innings	1 – First Innings, 2 – Second Innings
Batting Team	Team Batting currently
Bowling Team	Team Bowling currently
Over	Current Over of the Innings
Ball	Current Ball of the Over
Batsman	Name of the Batsman on Strike
Non-Striker	Name of the Batsman at the non-striker end
Bowler	Name of the Bowler for the Over in progress
Is_Super_Over	Indicates whether the over is super over or not
Wide Runs	Runs granted as wide
Bye Runs	Runs granted as bye
Leg Bye Runs	Runs granted as Leg-Bye
No Ball Runs	Runs granted for No Ball
Penalty Runs	Runs granted for Penalty
Batsman Runs	Runs scored by the Batsman
Extra Runs	Combined Extra Runs
Total Runs	Extra Runs + Runs scored by the Batsman
Player Dismissed	Name of the dismissed player (if any)
Dismissal Kind	Reason for the dismissal of the player (if any)
Fielder	Name of the Fielder responsible for dismissal (if any)

```
hive> CREATE TABLE MATCHES ( Id INT, Season STRING, City STRING,  Date_Of_Match DATE,  Team1 STRING, Team2 STRING, Toss_Winner STRING, Toss_Dec
ision STRING, Result STRING , Dl_Applied INT, Winner STRING, Win_By_Runs INT, Win_By_Wickets INT, Player_Of_Match STRING, Venue STRING, Umpire1
 STRING, Umpire2 STRING, Umpire3 STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LINES TERMINATED BY '\n' ;
OK

hive> LOAD DATA LOCAL INPATH 'matches.dat'  OVERWRITE INTO TABLE MATCHES;
Loading data to table default.matches
OK
```

Fig. 2. Creation of Table MATCHES and loading of data into it from the matches.dat file

B. Data Analysis

The IPL Data Analysis includes the following queries:

- The IPL Season with most number of matches played
- The most successful team in the IPL history from 2008-2019
- The cricketer to receive Man of the Match maximum number of times in every team given a season
- Highest Run Scorer from every team given a season

• Most successful bowler in the history of IPL from 2008-2019

All of the above queries are executed by the creation of views since views tend to minimize the processing time involved in the query execution.

```
hive> CREATE TABLE DELIVERIES ( Match_Id INT, Inning INT, Batting_Team STRING, Bowling_Team STRING, Overs INT, Ball INT, Batsman STRING, Non_St
riker STRING, Bowler STRING, Is_Super_Over INT, Wide_Runs INT, Bye_Runs INT, Leg_Bye_Runs INT, No_Ball_Runs INT, Penalty_Runs INT, Batsman_Runs
 INT, Extra_Runs INT, Total_Runs INT, Player_Dismissed STRING, Dismissal_Kind STRING, Fielder STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY
 ',' LINES TERMINATED BY '\n' ;
OK

hive> LOAD DATA LOCAL INPATH 'deliveries.dat'  OVERWRITE INTO TABLE DELIVERIES;
Loading data to table default.deliveries
OK
```

Fig. 3. Creation of Table DELIVERIES and loading data into it from deliveries.dat file.

a) The IPL Season with most number of matches played

This query determines the season in which the maximum number of matches were played. The Table MATCHES and the attributes Match ID, and Season were utilized. A view was created to store the results of count of the numbers of matches played in every season by aggregating over the 'season' attribute. Finally, the season with maximum matches was fetched. Fig 4 gives the result obtained by the execution of this query.

number_of_matches	season
76	2013

Fig. 4. IPL Season with the maximum number of matches played

b) The most successful team in the IPL history from 2008-2019

This query determines the most successful team in the IPL history from 2008-2019. This is determined by counting the number of times a team has won the IPL trophy. First, a view was created to store the details on the final match of every season. Finally, aggregation was performed over the 'winner' attribute to obtain the Name of the most successful IPL team from 2008-2019. Fig 5 gives the results obtained by the execution of this query.

win_count	team
4	Mumbai Indians

Fig. 5. Most successful team in the history of IPL from 2008-2019

c) The cricketer to receive Man of the Match maximum number of times in every team given a season

This query determines the top player of every team depending on the number of times the player was awarded Man of the Match for a given season. First, a view is created to store the details on the number of times each player in a particular team has been awarded Man of the Match by aggregating over the attributes 'player_of_match' and 'winner' attribute for a given season. The resulting view has three attributes: 'maxcount', 'player_of_match' and 'winner'. Next, another view is created to store the details on the maximum count of Man of the Match winner for every team by using the first view created. This view has two attributes: 'maxcount' and 'winner'. Finally, both the views are joined based on attributes 'maxcount' and 'winner' to determine the player from every team with the maximum Man of the Match award. Fig 6 shows the result of execution of the query for the season 2017.

f.maximum_wins	g.player_of_match	f.winner
1	CH Gayle	Royal Challengers Bangalore
1	SV Samson	Delhi Daredevils
1	SS Iyer	Delhi Daredevils
1	RR Pant	Delhi Daredevils
1	KK Nair	Delhi Daredevils
1	KM Jadhav	Royal Challengers Bangalore
1	HV Patel	Royal Challengers Bangalore
1	Mohammed Shami	Delhi Daredevils
1	CJ Anderson	Delhi Daredevils
2	Rashid Khan	Sunrisers Hyderabad
2	N Rana	Mumbai Indians
2	KH Pandya	Mumbai Indians
2	AJ Tye	Gujarat Lions
2	Sandeep Sharma	Kings XI Punjab
3	BA Stokes	Rising Pune Supergiant
3	NM Coulter-Nile	Kolkata Knight Riders

Fig. 6. The list of cricketers from every IPL Team to receive Man of the Match maximum number of times for the year 2017.

d) Highest Run Scorer from every team given a season

This query determines the batsman who scored the highest runs for a given season of IPL. First, a view is created to store the sum of runs scored by every player from every team by aggregating over the attributes 'batsman' and 'batting_team' for a given season. This view has 3 attributes: 'max_runs', 'batsman' and 'batting_team'. Next, another view is created using the first view to store the details on maximum value of attribute 'max_runs' from every team. This view has two attributes: max_runs and 'batting_team'. Finally, both the views are joined based on attributes 'max_runs' and 'batting_team' to determine the player from every team with the maximum runs. Fig 7 shows the result of execution of the query for the season 2008.

b.max_runs	a.batsman	b.batting_team	b.season
616.0	SE Marsh	Kings XI Punjab	2008
472.0	SR Watson	Rajasthan Royals	2008
349.0	SC Ganguly	Kolkata Knight Riders	2008
514.0	ST Jayasuriya	Mumbai Indians	2008
534.0	G Gambhir	Delhi Daredevils	2008
436.0	AC Gilchrist	Deccan Chargers	2008
421.0	SK Raina	Chennai Super Kings	2008
371.0	R Dravid	Royal Challengers Bangalore	2008

Fig. 7. The list of batsman from every IPL Team to score maximum runs for the year 2008

e) Most successful bowler in the history of IPL from 2008-2019

This query retrieves the name of the bowler with the maximum wickets in the history of IPL from 2008-2019. First, a view is created to store the sum of wickets taken by every bowler from every team by aggregating over the attributes 'bowler' and 'bowling_team' combining all the seasons of IPL. This view has 3 attributes: 'max_wickets', 'bowler' and 'bowling_team'. Finally, the selection of one row from the view is done on the basis of a wicket count matching the maximum wicket count from the view. Fig 8 shows the details of the most successful bowler in the IPL history from 2008-2019.

max_wickets	bowler	bowling_team
170	SL Malinga	Mumbai Indians

Fig. 8. The most successful bowler in the history of IPL from 2008-2019

V. PERFORMANCE ANALYSIS

In this work, IPL Dataset was analyzed and the results were visualized using HiveQL. Hive Queries were executed on Hadoop-Hive platform and the execution time was noted down. The same set of queries was executed using SQL and the execution time of SQL query was compared with the HiveQL execution time. The execution time using SQL was higher compared to that on Hadoop-Hive. Table III shows the comparative analysis of the response time of SQL and HiveQL queries and Fig 9 depicts a graph for the response time comparison.

Table- III: Comparative Analysis of the Response time for HiveQL and MySQL

Query	SQL Response Time (in sec)	Hive Response Time (in sec)
1	42.09	25.41
2	76.43	43.87
3	87.91	56.32
4	101.36	74.57
5	43.99	29.86

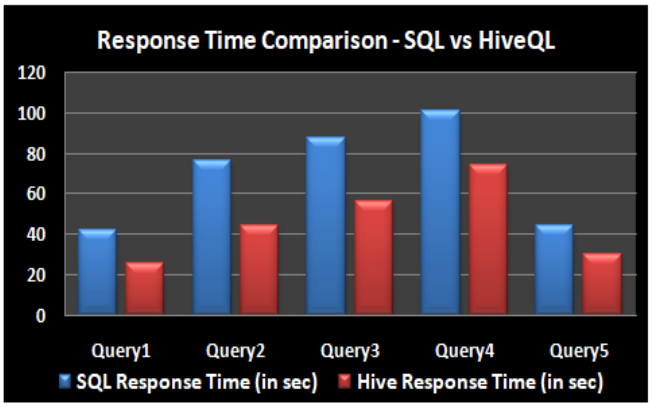

Fig. 9. Graph comparing the response time for HiveQL and SQL

VI. CONCLUSION

Analyzing Big Data is a very challenging task since it involves the processing of huge amount of data. Hadoop-Hive is well suited for the analysis of huge datasets. The performance of Hadoop-Hive in case of smaller datasets is low but it increases as the size of the dataset grows. However, SQL is well suited for smaller datasets.

This work analyzed IPL dataset using both SQL and HiveQL for five different cases. The first case determined the IPL season with most number of matches played. The second case determined the most successful team in the history of IPL. The third case determined the cricketer from every team to receive Man of Match title maximum number of times. The fourth case determined the best batsman from every team and the last case determined the best bowler in the history of IPL. For each of these cases, it was found that the response time given by HiveQL was lesser than that of SQL. This proves that Hive works the best for larger datasets.

As a part of future work, further queries can be written to retrieve other meaningful information like team statistics, player consistency, strike rate and other parameters which can be of use to the bidders in careful selection of their IPL teams.

REFERENCES

1. A. A. Prakash, and A. Aloysius, "Architecture Design for Hadoop No-SQL and Hive", in *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, vol. 3, no. 1, 2018, pp. 1069–1077.
2. J. K. Basha, and M. Balamurugan, "A Review on Hive and Pig", in *International Journal of Advanced Research in Basic Engineering Sciences and Technology*, vol. 3, no. 39, ISSN 2456-5717, 2017, pp. 53-58.
3. J. Mehta, and J. Woo, "Big Data Analysis of Historical Data using HIVE", in *APRN Journal of Systems and Software*, vol. 5, no. 2, ISSN 2222-9833, 2015, pp.40-43.
4. D. Lakshmi, I. Kim, and J. Woo, "Analysis of Movie Lens Data Set using Hive", in *ARPN Journal of Science and Technology*, vol. 3, no. 12, ISSN 2225-7217, 2013, pp. 1195-1198.
5. S. S. Mazumdar, and J. Mazumdar, "Big Data Analytics Framework using Machine Learning on Multiple Datasets", in *International Journal of Science and Research (IJSR)*, vol. 4, no. 8, ISSN 2319-7064, 2015, pp. 414-418.

6. S. K. Pushpa, T. N. Manjunath, and Srividhya, "Analysis of Airport Data using Hadoop-Hive: A Case Study", in *International Journal of Computer Applications*, ISSN 0975-8887, 2016, pp. 23-28.
7. T. Mehta and S. Hooda, "Analysis and Visualization of Movie Len Data Set using Hive and R", in *International Journal of Multidisciplinary Allied Research Review and Practices*, vol. 3, no. 7, ISSN 2455-1570, 2016.
8. P. Beri, and S. Ojha, "Comparative Analysis of Big Data Management for Social Networking Sites", in *International Conference on Computing for Sustainable Global Development*, 2016, pp. 1196-1200.
9. A. U. Abdullahi, R. Ahmad, and N. M. Zakaria, "Big Data: Performance Profiling of Meteorological and Oceanographic Data on Hive", in *3rd International Conference on Computer and Information Sciences*, 2016, pp. 203-208.
10. B. Jain and M. K. Kakhani, "Query Optimization in Hive for Large Datasets", in *Advances in Computer Science and Information Technology (ACSIT)*, vol. 2, no. 4, ISSN 2393-9915, 2015, pp. 321-325.
11. P. Jain and J. P. Maurya, "Comparative Analysis using Hive and Pig on Consumer Data", in *International Journal of Computer Science and Information Technologies*, vol. 8, no. 2, ISSN 0975-9646, 2017, pp. 285-291.
12. Indian Premier League 2008-2019 Dataset. Available: https://www.kaggle.com/nowke9/ipldata
13. Apache Hive TM. Available: https://hive.apache.org/
14. Apache Hadoop, Available: https://hadoop.apache.org/

AUTHORS PROFILE

Sapna S., obtained her B.E. degree in Computer Science and Engineering from NMAM Institute of Technology, Nitte, India and M.Tech degree in Computer Science and Engineering from Manipal Institute of Technology, Manipal, India. She has 1.5 years of teaching experience and 1 year of industry experience. She is currently working as an Assistant Professor in the Department of Information Science and Engineering, NMAMIT, Nitte. Her areas of interest include Image Processing, Machine Learning and Big Data Analytics.

Sandhya S., obtained her B.E. and M.Tech degree in Computer Science and Engineering from NMAM Institute of Technology, Nitte, India. She is currently working as an Assistant Professor in the Department of Information Science and Engineering, NMAMIT, Nitte. She has 6 months of teaching experience and 2 years of industry experience. Her areas of interest include Image Processing, Machine Learning, Big Data Analytics and Database Management Systems.