ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Διαχείριση Μεγάλων Δεδομένων από το Twitter και εξόρυξη δεδομένων

Συγγραφέας: Κουλιανός Νικόλαος
ice18390113

Επιβλέπων: Τρούσσας Χρήστος

Αθήνα, Οκτώβριος 2023

UNIVERSITY OF WEST ATTICA

FACULTY OF ENGINEERING DEPARTMENT OF INFORMATICS AND COMPUTER ENGINEERING

DIPLOMA THESIS

Big Data Management from Twitter and Data Mining

Author: Coulianos Nicholas
ice18390113

Supervisor: Troussas Christos

Athens, October 2023

Big Data Management from Twitter and Data Mining

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Διαχείριση Μεγάλων Δεδομένων από τοTwitter και εξόρυξη δεδομένων

Big Data Management from Twitter and Data Mining

Νίκος Κουλιανός

18390113

Γλώσσα Συγγραφής: Αγγλικά

Μέλη Εξεταστικής Επιτροπής συμπεριλαμβανομένου και του Εισηγητή

Α/α	ΟΝΟΜΑ ΕΠΩΝΥΜΟ	ΒΑΘΜΙΔΑ/ΙΔΙΟΤΗΤΑ	ΥΠΟΓΡΑΦΗ
1	ΧΡΗΣΤΟΣ ΤΡΟΥΣΣΑΣ	ΕΠ. ΚΑΘΗΓΗΤΗΣ	Christos Troussas Digitally signed by Christos Troussas Date: 2023.10.19 17:23:46 +03'00'
2	ΑΚΡΙΒΗ ΚΡΟΥΣΚΑ	ΕΔΙΠ	Akrivi Krouska Digitally signed by Akrivi Krouska Date: 2023.10.19 17:41:47 +03'00'
3	ΠΑΝΑΓΙΩΤΑ ΤΣΕΛΕΝΤΗ	ΕΔΙΠ	Panagiota Tselenti Digitally signed by Panagiota Tselenti Date: 2023.10.20 17:15:52 +03'00'

Ημερομηνία Εξέτασης 6/10/23

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Ο κάτωθι υπογεγραμμένος **Κουλιανός Νικόλαος** του **Θεόδωρου-Νομικού**, με αριθμό μητρώου **ice18390113** φοιτητής του **Πανεπιστημίου Δυτικής Αττικής** της **Σχολής Μηχανικών** του **Τμήματος Μηχανικών Πληροφορικής και Υπολογιστών**, δηλώνω υπεύθυνα ότι:

«Είμαι ο συγγραφέας αυτής της Διπλωματικής Εργασίας και κάθε βοήθεια την οποία είχα για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφή από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος. Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του διπλώματός μου».

Επιθυμώ την απαγόρευση πρόσβασης στο πλήρες κείμενο της εργασίας μου μέχρι 6 μήνες (06/05/2024) και έπειτα από αίτηση μου στη Βιβλιοθήκη και έγκριση του επιβλέποντα καθηγητή.

Ο Δηλών

Νίκος Ν. Κουλιανός

Επιβλέπων Καθηγητής

Christos Troussas

Digitally signed by Christos Troussas
Date: 2023.10.19 17:24:05 +03'00'

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή, κύριο Τρούσσα για την εμπιστοσύνη που μου έδειξε από την αρχή υποστηρίζοντας τις ιδέες που πρότεινα που σχημάτισαν το παρόν θέμα της εργασίας αυτής. Επίσης, θα ήθελα να τον ευχαριστήσω για τις συμβουλές και την καθοδήγηση που μου παρείχε κατά την διάρκεια εκπόνησης της διπλωματικής εργασίας μου.

Θα ήθελα να ευχαριστήσω επίσης την οικογένεια μου που με στηρίζει και με έχει βοηθήσει να εξελιχτώ και να φτάσω στο σημείο που βρίσκομαι σήμερα. Επιπλέον, θα ήθελα να ευχαριστήσω τους φίλους που μου στάθηκαν όλο αυτό το καιρό.

Τέλος, θα ήθελα να ευχαριστήσω όλους τους καθηγητές του ιδρύματος που ήταν πρόθυμοι να με βοηθήσουν να κατανοήσω καλύτερα το πεδίο επιστήμης των υπολογιστών αλλά και προσπαθούσαν να με βοηθήσουν να εξελιχτώ.

Περίληψη

Η παρούσα διπλωματική εξετάζει την ενσωμάτωση του ισχυρού οικοσυστήματος του Hadoop με την επεξεργασία δεδομένων από το Twitter σε πραγματικό χρόνο, που στη συνέχεια διαχειρίζονται από το πρότυπο του MapReduce. Ο πρωταρχικός στόχος αυτής της έρευνας είναι να εντοπίσει σημαντικές πληροφορίες που κρύβονται μέσα σε μεγάλες και διαρκώς μεταβαλλόμενες πηγές δεδομένων. Στο δεύτερο κεφάλαιο παραθέτονται τα προβλήματα των Μεγάλων Δεδομένων (Big Data), τονίζοντας τους κρίσιμους ρόλους του HDFS και του YARN. Η μελέτη επικεντρώνεται στη δυναμική φύση των δεδομένων του Twitter, με ιδιαίτερη έμφαση στον στρατηγικό ρόλο του Kafka στον συντονισμό των δεδομένων. Εντός αυτού του πλαισίου, το τρίτο κεφάλαιο τονίζει τη θεμελιώδη σημασία του MapReduce, το οποίο έχει επαινεθεί για τις εξαιρετικές του ικανότητες στην υποστήριξη της παράλληλης επεξεργασίας. Επίσης, η έρευνα διευρύνει το πεδίο της ανάλυσης δεδομένων εξερευνώντας πολύπλοκους αλγορίθμους ομαδοποίησης (clustering) και ταξινόμησης (classification), οι οποίοι είναι κρίσιμα εργαλεία για τη συλλογή ωφέλιμων δεδομένων από τον καταιγισμό των πληροφοριών και δεδομένων. Τα επόμενα κεφάλαια εξετάζουν την πραγματική υλοποίηση της μελέτης. Στο τέταρτο και πέμπτο Κεφάλαιο, αποκαλύπτεται το αρχιτεκτονικό σχέδιο, παρέχοντας την απαραίτητη ανάλυση πάνω στο σχεδιασμό. Στο Κεφάλαιο έκτο, αναλύονται προσεκτικά τα δεδομένα του Twitter και τα αποτελέσματα που έχουν εξαχθεί. Το τελευταίο και έβδομο κεφάλαιο ασχολούμαστε με την πρακτική εφαρμογή, διευρύνοντας το πεδίο για την καθιέρωση ενός συστήματος ολοκληρωμένης αξιολόγησης.

ΕΠΙΣΤΗΜΟΝΙΚΗ ΠΕΡΙΟΧΗ: Αποθήκευση Δεδομένων, Εξόρυξη Δεδομένων

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Apache Hadoop, HDFS, YARN, MapReduce, Kafka, Twitter, Streaming Data, ML, Clustering, Classification

Abstract

This thesis investigates the seamless integration of Hadoop's strong ecosystem with real-time Twitter data processing, all managed by the powerful MapReduce paradigm. The primary goal of this research is to find significant insights hidden inside huge and ever-changing data sources. The second chapter sets the scene against the backdrop of Big Data issues, stressing the critical roles of HDFS and YARN. The study focuses on the dynamic nature of Twitter data, with a special emphasis on Kafka's strategic role in data management orchestration. Within this dynamic area, Chapter three emphasizes the fundamental importance of MapReduce, which has been praised for its exceptional powers in aiding parallel processing. The research broadens its reach by delving into complex clustering and classification algorithms, which are critical tools for collecting actionable insight from the vast sea of data. The next chapters dive into the study's actual implementation. In Chapters four and five, the architectural blueprint is unveiled, providing insight into the complex design. In Chapter six, we carefully analyze the intricate Twitter data and share the achieved results. Chapter seven transitions to practical investigation, expanding the scope to establish a thorough evaluation system.

SCIENTIFIC FIELD: Data Warehousing, Data Mining

KEYWORDS: Apache Hadoop, HDFS, YARN, MapReduce, Kafka, Twitter, Streaming Data, ML, Clustering, Classification

Table of Contents

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ i
ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ..... iii
ΕΥΧΑΡΙΣΤΙΕΣ iv
Περίληψη..... v
Abstract..... vi
List of Figures..... xi
List of Tables..... xii
List of Abbreviations xiii
1. Introduction 1
1.1 Problem Statement 1
1.2 Research Objectives..... 1
1.3 Thesis Organization 1
2. Theoretical Background..... 3
2.1 Big Data Management and Challenges..... 3
2.1.1 Addressing V’s of Big Data 3
2.1.2 The Big Data Problem and it’s Adaptation Strategies..... 4
2.1.3 Introducing Hadoop 5
2.2 Hadoop Distributed File System (HDFS) 5
2.2.1 What is HDFS..... 5
2.2.2 Large Files and Scalability in HDFS 6
2.2.3 Hardware Failure 7
2.2.4 HDFS Backbone: Understanding NameNode and DataNodes 7
2.2.5 Ensuring Data Integrity: The Power of Replication..... 8
2.2.6 Unraveling Replica Placement 9
2.2.7 State: Safemode 10
2.2.8 File System Metadata 10
2.2.9 Data Block Organization..... 11
2.3 Yet Another Resource Negotiator (YARN) 11
2.3.1 What is YARN..... 11
2.3.2 ResourceManager’s Two Major Components..... 13
Scheduler..... 13
ApplicationsManager 13
2.4 Streaming Data and Twitter 13
2.4.1 What are Streaming Data..... 13

2.4.2 Twitter 14
2.5 Kafka as a Streaming Platform........ 15
2.5.1 What is Kafka 15
2.5.2 Decoding Kafka: Simplifying Data Streaming 15
2.5.3 Kafka's Architecture........ 16
2.6 Serialization Techniques for Streaming Data 17
2.6.1 What is Serialization........ 18
2.6.2 Data Serialization on Distributed Systems........ 18
2.6.3 The importance of Data Serialization on Big Data 19
2.6.4 Deserialization........ 19
2.6.5 Deserialization on Distributed Systems........ 20
2.6.6 Avro........ 20
3. MapReduce and Data Analysis 22
3.1 Introduction to MapReduce 22
3.1.1 The Benefits of MapReduce 22
3.1.2 MapReduce in a nutshell 22
3.1.3 The Input-Output and Key-Value Pair Relationship 23
3.1.4 Deconstructing MapReduce........ 24
The Mapper 24
Map Count: Enhancing Simplicity and Efficiency........ 25
The Reducer 25
Reduce Counts: Enhancing Simplicity and Efficiency........ 26
Partitioner........ 26
3.2 Clustering Techniques for Result Extraction........ 27
3.2.1 K-Means........ 27
3.2.2 Gaussian mixture model (GMM) 29
3.3 Classification Techniques for Result Extraction........ 30
3.3.1 Preprocessing Techniques........ 30
3.3.2 K-Nearest Neighbor (KNN) 31
3.3.3 Support Vector Machines (SVM)........ 32
3.3.4 Performance Evaluation 32
4. Methodology........ 34
4.1 Problem Definition........ 34
4.1.1 Data Management Challenges........ 34
4.1.2 Processing Complexity 34

4.1.3 Scalability and Real-Time Analysis 35
4.2 Data Collection 35
4.2.1 Kafka Topic and Partition Configuration 35
Topic 35
Partition 35
4.2.2 Broker Configuration 35
Data Ingestion Process 35
4.3 Dataset Presentation 36
4.3.1 Data Collection 36
4.3.2 Dataset Characteristics 36
4.4 Data Preprocessing 36
4.5 Exploratory Data Analysis 37
4.6 Feature Engineering 37
4.7 Data Analysis 38
4.7.1 HDFS and MapReduce Analysis 38
4.7.2 Clustering and Classification Analysis 38
4.8 Results Interpretation 38
4.9 Validation 39
4.9.1 Data Validation 39
4.9.2 Validation of Analysis Techniques 39
4.9.3 Visualization 40
5. Implementation 41
5.1 System Architecture and Design 41
5.1.1 Application Ambitions: Impact and Purpose 41
5.1.2 Applications Architecture 42
5.2 Tools and Technologies Used 43
5.3 Data Collection and Preprocessing 43
5.3.1 Streamlining Data Collection with Kafka 44
Twitter Data Streaming 44
5.3.2 Data Preprocessing 45
5.5 Clustering and Classification Implementation 47
5.6 Dataflow Management 48
6. Experimental Evaluation 50
6.1 Experimental Setup and Data Description 50
6.2 Results and Analysis 50
6.2.1 Kafka Analysis 50

6.2.2 MapReduce Analysis 51
6.2.3 HDFS Analysis 52
6.2.4 Clustering Analysis 54
K-Means 54
Gaussian Mixture Model 55
6.2.5 Classification Analysis 56
6.3 Discussion 59
6.3.1 Interpretation of Results 59
Kafka as a Data Collector 59
Exploiting MapReduce for Data Analysis 60
Harnessing HDFS for Data Storage and Accessibility 60
K-Means VS Gaussian Mixture Model 60
K-Nearest Neighbors VS Support Vector Machines 61
Total results for clustering and classification methods 61
Final thoughts on results 61
7. Conclusion 62
7.1 Summary of Contributions 62
7.2 Limitations and Future Work 63
8. Appendices 65
Appendix A: United States Indexing 65
Appendix B: HDFS Logs 66
9. References 67

List of Figures

Figure 1: The 5V's of Big Data Characteristics [7]. 4
Figure 2: HDFS Architecture[12]. 8
Figure 3: Data Replication [12]. 9
Figure 4: The YARN Architecture [12]. 12
Figure 5: The Kafka Architecture [37]. 17
Figure 6: Serialization and Deserialization process [38]. 19
Figure 7: The MapReduce Architecture [3]. 24
Figure 8: The Elbow Method [52]. 28
Figure 9: Application's Infrastructure Diagram. 42
Figure 10: Data Collection. 51
Figure 11: HDFS File Browser. 53
Figure 12: HDFS File Information. 53
Figure 13: Elbow Point. 54
Figure 14: K-Means Clustering. 55
Figure 15: GMM Clustering. 56
Figure 16: KNN Classification over K-Means Cluster. 57
Figure 17: KNN Classification over GMM Cluster. 57
Figure 18: SVM Classification over K-Means Cluster. 58
Figure 19: SVM Classification over GMM Cluster. 58
Figure 20: NameNode log when creating directory and storing data. 66
Figure 21: DataNode log when creating directory and storing data. 66

List of Tables

Table 1: Part of the Dataset. 50
Table 2: Input Data for MapReduce Paradigm. 51
Table 3: Mapper Output. 51
Table 4: Shuffle and Sort phase. 52
Table 5: MapReduce output. 52
Table 6: Performance Metrics. 59

List of Abbreviations

GNU	Gnu's Not Unix
UNIX	Uniplexed Information Computing System
IBM	Internation Business Machines Corporation
IoT	Internet of Things
HDFS	Hadoop Distributed File System
OS	Operating System
YARN	Yet Another Resource Negotiator
TCP	Transmission Control Protocol
RM	ResourceManager
AM	ApplicationMaster
DAG	Directed Acyclic Graph
CPU	Central Processing Unit
REST	Representational state transfer
API	Application Programming Interface
RPC	Remote Procedure Calls
KB	Kilobyte
MB	Megabyte
GB	Gigabyte
TB	Terabyte
PB	Petabyte
EB	Exabyte
ZB	Zettabyte
int	Integer
SSE	Sum of Squared Errors
GMM	Gaussian Mixture Model
KNN	K Nearest Neighbour
SVM	Support Vector Machines

TF - Term Frequency

IDF - Inverse Document Frequency

EDA - Exploratory Data Analysis

CSV - comma-separated values

1. Introduction

1.1 Problem Statement

In the era of big data, organizations face significant challenges in managing and deriving value from vast and diverse datasets. The challenges of data storage, processing, scalability, and real-time analysis necessitate creative solutions. Traditional data management systems fail to keep up with the volume and velocity of data created, especially when dealing with streaming data sources such as social media platforms. In addition, effective analytic approaches that can extract valuable insights from big datasets are critical.

1.2 Research Objectives

The primary goals are to analyze the challenges caused by big data management, to investigate effective data processing technologies, and to generate relevant insights from varied datasets. These goals are broken down as follows:

- **Objective 1:** Understanding and addressing obstacles in managing large and diverse datasets; including as data storage, processing complexity, scalability, and real-time analysis.
- **Objective 2:** Thoroughly look into the underlying operational processes of HDFS, diving into its architectural complexities in conjunction with MapReduce integration, and analyze the potential of YARN architecture. This examination intends to improve the overall performance of the system, increase data processing speed, and align with the study's general objectives.
- **Objective 3:** To extract timely insights from real-time streaming data sources, with a special focus on platforms such as Twitter. This goal also includes an evaluation of Kafka's usability and efficiency as a reliable streaming platform.
- **Objective 4:** To design a simple yet sturdy architecture that clearly illustrates the integrated system's entire range of capabilities. This design will seamlessly integrate HDFS, MapReduce, Kafka, clustering, classification and sophisticated analytics, providing a "developers-friendly" representation of the system's ability to handle and analyze real-world streaming data.

These study objectives together guide the investigation toward a thorough knowledge of the difficulties and opportunities in big data management and analysis, providing a road map for navigating the intricacies inherent in this dynamic ecosystem.

1.3 Thesis Organization

This thesis dives deep into big data management and analysis, exploring difficulties and solutions throughout chapters. The first chapter presents the research, emphasizing its relevance, aims, and organization. In Chapter 2, a comprehensive review of literature is presented, covering topics such as big data challenges, the role of Hadoop Distributed File System (HDFS), the role of Yet Another Resource Negotiator (YARN), and the use of streaming data from platforms such as Twitter, as well as the Kafka streaming framework and serialization techniques. Chapter 3 focuses on the MapReduce paradigm, explaining its principles and presenting data analysis clustering and classification algorithms. In Chapter 4, implementation methodologies are explored in depth, as are system architecture and design options, tools and technologies, data collecting and preprocessing stages, and the integration of MapReduce, clustering, and classification approaches. Chapter 5 is characterized by empirical validation, which defines the experimental setup, performance measures, outcomes, and analysis. Finally, Chapter 6 summarizes the thesis with a review of contributions, admits limits, and offers prospective future study possibilities. In addition, Chapter 7 has the appendices, which share the vital source code that underlies this research. Finally, in Chapter 8, there is the references section.

2. Theoretical Background

2.1 Big Data Management and Challenges

When one enters the world of data, the phrase "Big Data" becomes a confusing mystery that is cloaked in a lack of widespread agreement [1]. Based on the findings of [2], a cautious definition emerges that covers data amounts that boldly reach petabytes (10^{15} bytes) and beyond.

The 'big data' notion is based on an ongoing debate over data that, by virtue of its size and complexity, elegantly beyond the capabilities of individual devices. We are living in a time where data is proliferating at an unprecedented rate, growing at an exponential rate that is unmatched in human history [3]. Globally, the data generating environment has experienced a seismic upheaval, driving us from a modest 2 zettabytes (2×10^{21} bytes) in 2010 to an astounding apex of 64 zettabytes in 2020. The forecasts point us toward an incredible 181 zettabytes by the year 2025, therefore this trajectory is not a conclusion but rather a prologue to that future [4].

2.1.1 Addressing V’s of Big Data

It is interesting to note that the Big Data tale is anchored by three crucial characteristics known as the "Three V's of Big Data". These three triad V’s capture crucial elements that provide light on the complex symbiosis inside the vast tapestry of data, with each dimension tracing a unique path through its complexities [5]:

i. **Volume:** The first dimension, "Volume", is concerned with the sheer volume of data produced and gathered. The staggering amount of data is highlighted by the exponential rise from 2 zettabytes in 2010 to 64 zettabytes in 2020 [3]. Such massive numbers require sophisticated storage and analytical methods for management and value extraction.

ii. **Velocity:** The second dimension, "Velocity", describes how quickly data is produced and must be processed. Real-time analysis is essential for timely decision-making and getting insights from quickly evolving trends as a result of the unparalleled pace at which data is coming in from multiple sources.

iii. **Variety:** The third dimension, “Variety”, covers the many types of data, such as structured, unstructured, and semi-structured data. This aspect highlights the difficulties in combining and evaluating data from multiple sources and formats to get actionable insights.

Although the three V’s of big data volume, velocity, and variety are the core of this concept, the three V’s were later expanded to the five V’s of big data. Big data was defined by IBM as having the following characteristics: volume, velocity, variety, veracity and value [6].

iv. **Veracity:** The fourth dimension, “Veracity”, brings the discussion of data integrity and dependability. The pursuit of data accuracy takes center stage in the face of the explosion of data from many, occasionally unreliable sources. The importance of data validation, cleaning, and the guarantee of precision to negotiate the complex web of data authenticity is amplified by this dimension.

v. **Value:** The fifth dimension, “Value”, represents the completion of the Big Data journey, is where insights drawn from sizable, constantly changing, and complex datasets interact to provide tangible value. It represents the pinnacle of the Big Data adventure, when raw data is transformed into useful knowledge, sparking innovation and operational improvement. This aspect symbolizes the core of Big Data's goal, which is to provide priceless insights that spur development and lead paradigmatic transformations.

Figure 1 depicts a brief yet all-encompassing graphic depiction of the five V’s that underpin the framework of big data.

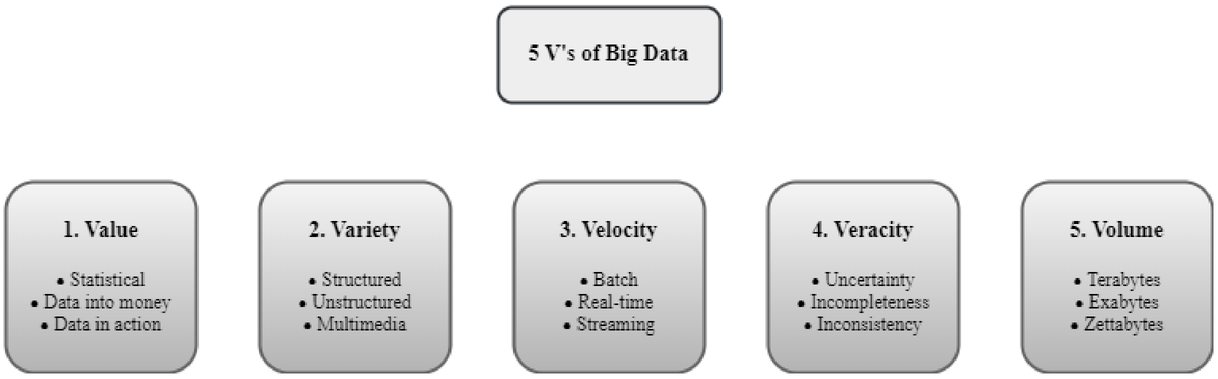


Figure 1: The 5V’s of Big Data Characteristics [7].

As time develops, the idea of data experiences the emergence of more dimensions, indicated by the “V’s”. These extra dimensions assist to a more complete understanding of the issue by depicting the changing nature of data in our current society [8].

2.1.2 The Big Data Problem and it’s Adaptation Strategies

Building upon the fundamental investigation of the five V’s that enshroud the world of big data, we are attracted to tackle the complicated issues that emerge in the aftermath of massive and heterogeneous information. Traditional approaches, illustrated by relational databases, adhere to organized data with rigid schemas. Yet, this environment undergoes a seismic upheaval as data expansion, driven largely by unstructured sources, accelerates at an astounding factor of 20:1 [9]. Simultaneously, the increasing scale of datasets throws a shadow on conventional data warehouses, leaving them incapable of processing full datasets. As a result, the integrity of the information held inside these repositories weakens, undermining its statistical accuracy and consequently, the trustworthiness of analytical insights gained from it. The clarion call of the big data era demands novel architectures designed to embrace scalability, resilience, and efficient parallel processing as essential attributes [10].

In response, the spotlight sharpens on the realm of big data processing and management; a pursuit that encompasses the intricate art of capturing and storing colossal data. A notable instance of this surge in interest is the emergence of the MapReduce paradigm. Nevertheless, it is incumbent to recognize that the pursuit extends beyond mere technology adoption. It beckons the development of platforms that harness these technologies to unravel meaningful insights and empower prudent business decisions. This evolution births the domain of big data analytics, a sphere wherein data mining and machine learning techniques seize pivotal roles across industries. From the corridors of retail and banking to the bastions of insurance and healthcare, these techniques illuminate the path towards operational optimization, risk mitigation, and enhanced profitability. It is within these sectors that the synergy of data mining and machine learning crystalize, propelling organizations to chart a course towards success amidst the complexities of the data-rich landscape [10].

To meet the escalating demands for higher throughput in the realm of big data, the adoption of distributed computing architectures is experiencing an unprecedented surge. Leading industry titans like Facebook, Amazon, and Yahoo! have embraced this trend, operating data centers comprising numerous nodes that collectively store massive petabytes of data [11].

2.1.3 Introducing Hadoop

Apache Hadoop stands as one of the most widely adopted distributed computing frameworks designed for the storage and processing of extensive datasets. Embraced not only by internet behemoths but also by small and medium-sized enterprises, often via outsourced solutions, Hadoop has heralded transformative shifts within the business landscape. By 2015, an estimated 76% of Fortune companies had integrated this technology into their operations, underscoring its pervasive influence [11].

Hadoop, a widely used framework for handling large datasets, tackles the scalability issue by providing solutions for distributed storage and processing. Although Hadoop is essentially a data processing platform, it provides a highly flexible base for various machine learning programs and applications [3], [12]. Given the lack of a single tool or framework that handles the full, or even a significant fraction, of common activities, a careful review of the trade-offs between usability, performance, and algorithmic selection becomes critical when evaluating various solutions. Despite their extensive incorporation into business processes, there is a noticeable paucity of detailed study on several such solutions, and the lack of a defined industry standard adds to the landscape's complexity [3].

At the moment, the Hadoop project consists of four main parts [3], [12], [13]:

i. **Hadoop Common**: The common utilities that serve as the foundation for the remaining Hadoop components.
ii. **Hadoop Distributed File System (HDFS):** A distributed file system designed to provide fast and efficient access to application data.
iii. **Hadoop YARN:** A platform dedicated to job scheduling orchestration and smart cluster resource management.
iv. **Hadoop MapReduce:** A system built on the YARN architecture that allows for simultaneous processing of huge and complicated datasets.

2.2 Hadoop Distributed File System (HDFS)

2.2.1 What is HDFS

The Hadoop Distributed File System (HDFS) is a popular file system widely used by enterprises involved in big data applications, is largely used for batch data processing. HDFS is well-known for its ability to provide high-throughput data access with low latency which is crucial in managing large datasets [14]. HDFS was meticulously engineered within the Apache Software

Foundation framework to serve a dual mandate: first, to provide a distributed repository capable of efficiently managing voluminous datasets while also bolstering fault tolerance against hardware failures, and second, to provide a robust infrastructure for the seamless execution of MapReduce operations critical for advanced data analytics [15]. It is important to note that the architectural design of HDFS is chiefly responsible for its modern fame. Similar to the UNIX file system design, HDFS distinguishes itself by spreading data across several hard drives while relying on cheap technology. This strategic approach not only makes its use cost-effective, but it also simplifies maintenance operations. Notably, the open-source Apache Hadoop framework runs easily on top of the HDFS infrastructure, enhancing its capabilities and emphasizing its importance [14].

Hadoop is well suited for large-scale data analysis, notably for handling and processing large files and storing large data units. Furthermore, in the context of streaming data, Hadoop has capabilities for real-time data processing and analysis. This versatility applies to a wide range of data sources, including social network data, where Hadoop's value remains significant [16]. It is worth noting that the majority of data used in social networks is unstructured, accounting for roughly 79% of all data used in social media situations. As a result, data processing becomes significantly more complex within the context of big data analytics. In this context, Apache Hadoop appears as a solid and practical framework that is well suited to analyzing material collected from social media networks [16 - 18].

As previously stated, the HDFS design is critical in organizing and analyzing this diverse and enormous data. Taking a high-level view HDFS is carefully designed to support large data volumes across a dispersed network of commodity hardware nodes. It employs a master-slave design, with data nodes serving as the foundation, holding discrete data blocks, transmitting data on demand, and providing periodic status updates to the NameNode. As the central repository, the NameNode catalogs this dynamic inventory, which includes file references and metadata, and successfully orchestrates data node interactions to satisfy client requests. This architecture is fortified by inherent fault tolerance mechanisms, which keep three or more clones of each data block to protect against probable disk failures. Furthermore, NameNode interruption eventualities are expertly managed, either by the availability of a supplementary NameNode or by establishing metadata backups across various file systems [3], [11 - 13].

2.2.2 Large Files and Scalability in HDFS

Prior to digging further into the complexities of HDFS architectures, it is important to note that HDFS applications are designed for large datasets, with a normal HDFS file ranging in size from gigabytes to terabytes. This design philosophy guarantees that HDFS is fine-tuned to handle large files. Its performance goals include providing large aggregate data bandwidth and the capacity to easily scale to hundreds of nodes within a single cluster. Furthermore, HDFS is designed to manage a substantial volume of files, numbering in the tens of millions, within a solitary instance. This thorough design approach extends to HDFS's portability across several systems. This deliberate versatility promotes HDFS's widespread acceptance as a preferred basis for a wide range of applications [3], [14], [19].

2.2.3 Hardware Failure

In the realm of data systems, hardware failures are a prevailing reality rather than an occasional anomaly. An HDFS environment encompasses a multitude of server machines, often numbering in the hundreds or thousands, each entrusted with housing specific segments of the file system's extensive data repository. Given the sheer abundance of individual components and the inherent vulnerability of each to potential malfunctions, the existence of non-operational components within HDFS is an inherent likelihood. Consequently, the aspiration to promptly detect faults and facilitate seamless automated recovery assumes a central position within the architectural blueprint of HDFS [12 - 13].

2.2.4 HDFS Backbone: Understanding NameNode and DataNodes

HDFS operates inside a master/slave architectural framework, which shapes its operational structure differently. An HDFS cluster is made up of a single NameNode, the authoritative master server in charge of managing the file system namespace and overseeing client access to files. Concurrently, a cohort of DataNodes is added to the cluster, generally equal to the number of nodes in the setup. These DataNodes play a critical role in administering the storage associated with their respective hosting nodes [3], [11 - 12], [20 - 21].

At it’s heart, HDFS provides an interface for the file system namespace, allowing for the storing of user-generated data within files. This underlying method divides files into discrete chunks, which are painstakingly stored over an array of DataNodes. The NameNode's critical functions include a wide range of file system namespace activities, including file manipulation, directory administration, and the strategic allocation of blocks to their appropriate DataNodes [12 - 13].

The DataNodes work together to answer read and write requests started by the file system's clients. Aside from that, the DataNodes actively participate in complex processes like as the creation, removal, and replication of data blocks, painstakingly according to the commands provided by the authoritative NameNode [11 - 12].

The NameNode and DataNode components are both precisely developed software modules designed to run smoothly on commodity hardware. These workstations are often outfitted with a GNU/Linux operating system (OS), which provides a solid foundation for their operation. Because HDFS's fundamental design is based on the Java programming language, the NameNode and DataNode software are globally compatible with any system that supports Java. This purposeful adoption of Java's strong portability provides HDFS with the flexibility to be smoothly deployed over a wide range of hardware [11 - 13].

A dedicated workstation hosts the NameNode software exclusively in a conventional deployment setup. Concurrently, each of the cluster's constituent machines houses an instance of the DataNode software. While the architectural design allows for the operation of numerous DataNodes on a single system at the same time, actual deployments frequently choose for a separate instance per machine to ensure best performance [12].

The design notion of a single NameNode inside a cluster assists to reduce the architectural complexity of the system. The NameNode serves as the central node, essentially arbitrating and holding all HDFS metadata. Importantly, this architectural framework is designed in such a way that user data is not routed through the NameNode, maintaining its integrity and efficient functioning [13].

Figure 2 shows a simplified visual representation of the HDFS architecture.

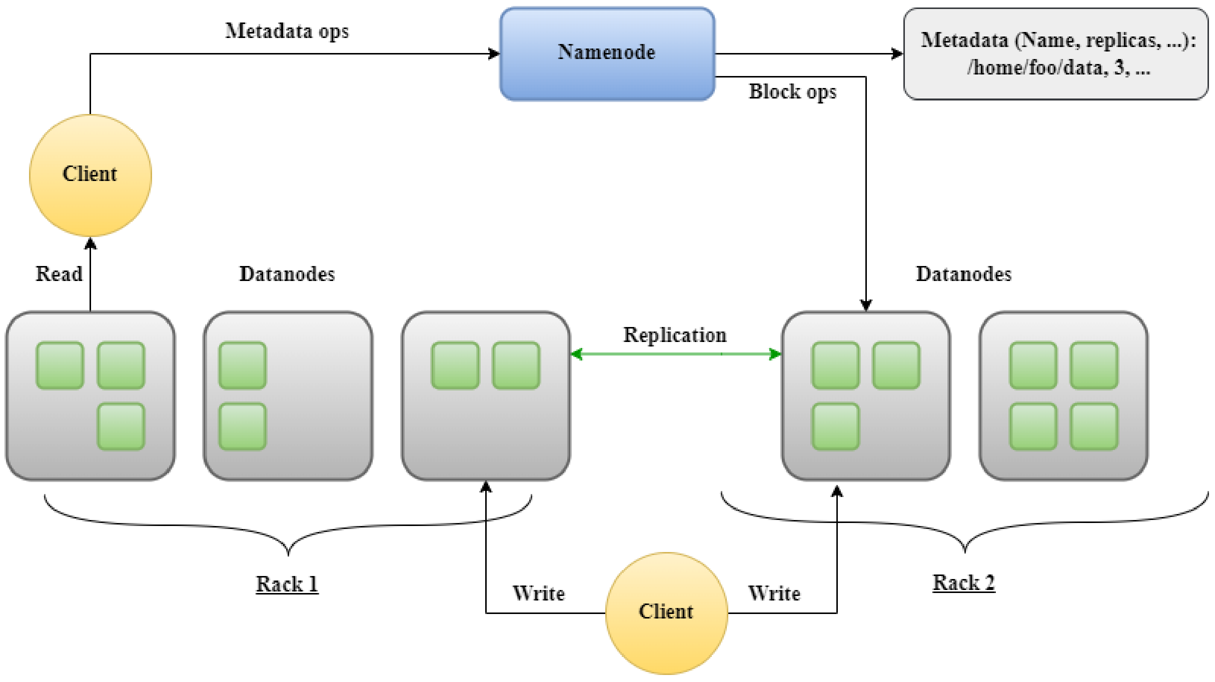


Figure 2: HDFS Architecture[12].

2.2.5 Ensuring Data Integrity: The Power of Replication

HDFS has a strong architecture that is geared to ensure the dependable storage of large files over a large cluster of servers. This is accomplished by segmenting each file into a number of blocks, resulting in a cohesive and understandable data arrangement. In order to provide fault tolerance, the system leverages block replication, creating several copies of a block throughout the cluster [13].

The specifications of block size and replication factor are flexible, allowing for per-file customization. Notably, with the exception of the last block, the blocks are all the same size, supporting a streamlined and unified structure. Furthermore, the framework's versatility was highlighted by the addition of variable-length blocks to ease append and hsync operations [12].

Applications may set the appropriate number of file copies under the HDFS umbrella, giving them actual control over data redundancy. This replication factor can be specified both when the file is created and when it is modified later. HDFS requires a write-once concept for files (excluding appends and truncates) to ensure data integrity, with a single writer assigned at any given time [3], [11 - 12].

The NameNode, an authoritative entity that navigates the distribution of block copies, is important to the orchestration of replication processes. Regular interactions with DataNodes influence its operational decisions. These DataNodes broadcast Heartbeats to demonstrate their operational status, and periodic Blockreports reveal the whole set of blocks on a DataNode, supporting a dynamic and intelligent replication strategy [12 - 13].

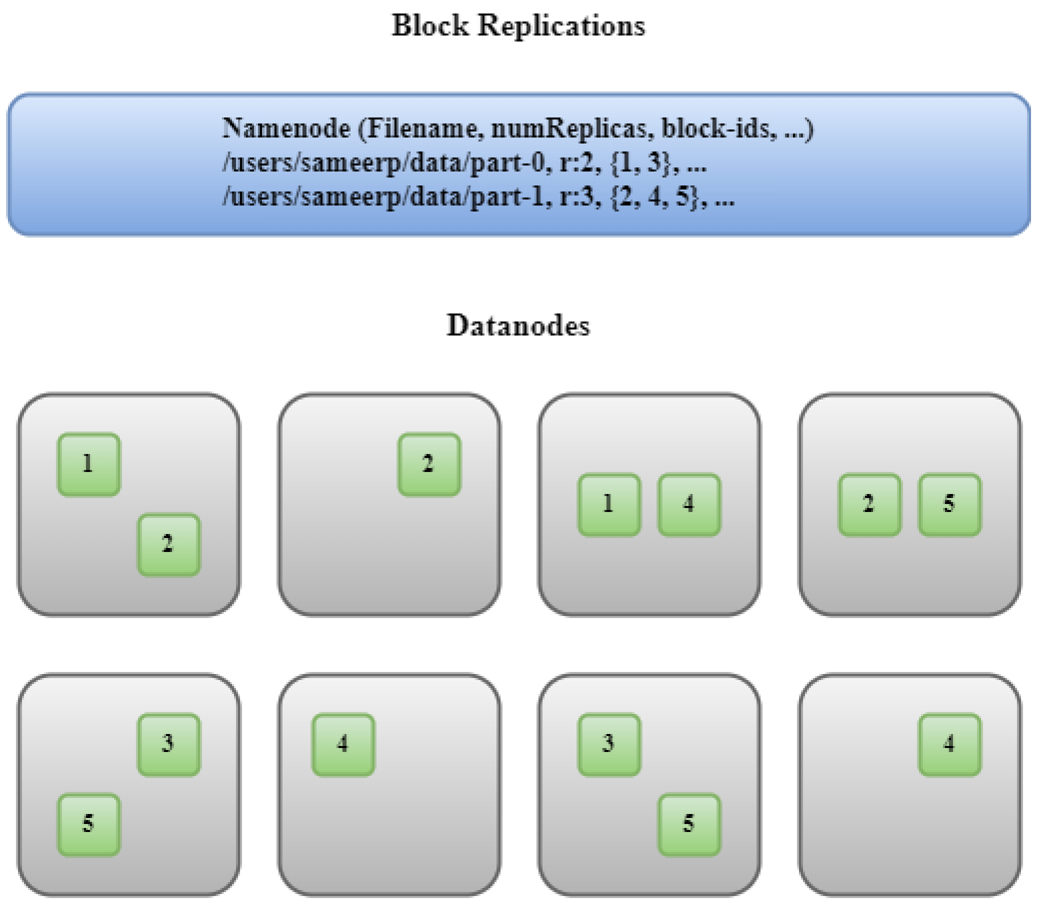

Block Replications
Namenode (Filename, numReplicas, block-ids, ...)
/users/sameerp/data/part-0, r:2, {1, 3}, ...
/users/sameerp/data/part-1, r:3, {2, 4, 5}, ...
Datanodes
1
2
2
1
4
2
5
3
5
4
3
5
4

Figure 3: Data Replication [12].

2.2.6 Unraveling Replica Placement

HDFS's resiliency and efficiency are dependent on replica placement. In contrast to other distributed file systems, HDFS promotes replica optimization, which necessitates rigorous calibration and practical knowledge. To improve data dependability, availability, and network consumption, a rack-aware replica placement strategy has been implemented. While the present policy is in its early stages, it has the short-term goals of validation, behavior comprehension, and laying the groundwork for more advanced techniques [12 - 13].

Large-scale HDFS deployments need clusters spanning many racks, with inter-node communication frequently traversing switches. Notably, network bandwidth differs across nodes in the same rack and between racks. The NameNode is critical in assigning rack IDs to DataNodes in accordance with the Hadoop Rack Awareness protocol. An initial strategy seeks to distribute replicas across distinct racks in order to avoid data loss during rack failures and enhance data reading bandwidth. In the event of component failure, this strategy assures equal allocation and efficient load balancing. However, because of block transfers across racks, the approach imposes additional write operation costs. This delicate balance emphasizes the significance of careful replica placement inside HDFS [3], [12 - 13].

For a replication factor of three, HDFS implements a strategic placement approach. If the writer is on a DataNode, one duplicate is placed on the local computer; otherwise, a replica is placed on a random DataNode in the same rack as the writer. Another copy is assigned to a node in a separate distant rack, while the last replica is assigned to a specific node inside the same remote rack. This strategy maintains data dependability and read performance while improving write performance by decreasing inter-rack write traffic. When the replication factor exceeds three, further copies are deployed at random while keeping a per-rack upper limit in mind. With the inclusion of HDFS Storage Types and Storage Policies, replica placement now takes into account both rack awareness and storage policy. The NameNode first utilizes rack awareness to validate the storage type of the candidate node against the file's policy. If a suitable node cannot be discovered, the NameNode searches for nodes that provide fallback storage types. This refined method enables accurate replica placement within HDFS [12 - 13].

2.2.7 State: Safemode

During startup, the NameNode enters to a Safemode state, which stops data block replication. The NameNode receives TCP Heartbeat signals and Blockreport messages from DataNodes, which provide listings of hosted data blocks with low replica counts. A data block is considered safely replicated when the required minimum replicas check in. Safemode is terminated after a specific proportion of securely replicated blocks report (plus time). After entering Safemode, the NameNode detects blocks that require more copies and begins replication to other DataNodes [12].

2.2.8 File System Metadata

The NameNode manages the HDFS namespace, and it uses the EditLog as a transaction log to record any change made to file system information. For example, adding a new file to HDFS causes the NameNode to record the change in the EditLog. Similarly, changes like changing the replication factor result in new entries in the EditLog. The EditLog is saved as a file in the host operating system's file system. This includes the whole file system namespace, as well as the mapping of blocks to files and other file system properties, which are all preserved in a file called the FsImage. This FsImage is also saved locally on the FileSystem of the NameNode [11 - 13].

The NameNode keeps an in-memory representation of the whole file system namespace as well as the file Blockmap. A checkpoint is done upon startup or when a specified threshold is met. This entails reading the FsImage and EditLog from disk, applying all of the EditLog transactions to the in-memory FsImage representation, and then storing the revised FsImage to disk. Because it’s modifications have been applied to the permanent FsImage, the previous EditLog can be shortened. This checkpoint technique guarantees that the HDFS has a consistent view of the file system information by capturing a snapshot of the metadata in the FsImage. Due to performance considerations, incremental changes are not directly applied to the FsImage, thus the EditLog is utilized to capture these changes until they are merged into the FsImage during a checkpoint [11 - 13].

Meanwhile, HDFS data is stored in files on the DataNode's native file system. Each HDFS block is associated with a single file on the local system. The DataNode does not store all files in the same directory to minimize inefficiencies; instead, it determines the ideal amount of files per directory and organizes subdirectories accordingly. When a DataNode boots up, it searches its local file system for HDFS data blocks associated with local files and provides this Blockreport to the NameNode. This report allows the NameNode to have a consistent picture of dispersed data blocks among DataNodes [11 - 13].

2.2.9 Data Block Organization

HDFS is architected to effectively manage and support the storage of large files. It is best suited for applications that handle large datasets. These applications follow a write-once, read-many paradigm, highlighting the importance of efficient streaming-speed retrieval. This concept is enforced on HDFS files, with a common block size of 128 MB. As a result, HDFS divides its files into 128 MB chunks, with the goal of distributing these chunks among multiple DataNodes wherever possible. This method guarantees that resources are used optimally inside the HDFS framework [11 - 12].

2.3 Yet Another Resource Negotiator (YARN)

We shift our focus from the Hadoop Distributed File System (HDFS) to the delicate interaction it has with YARN (Yet Another Resource Negotiator), a sophisticated resource management framework. This symbiotic link between HDFS and YARN is the foundation of Hadoop's unique ability to manage large datasets methodically and effectively. The harmonization of data storage and processing takes precedence within this collaborative framework, ushering in a new era of creative data-driven solutions. The strategic integration of HDFS with YARN enables organizations to manage the intricacies of big data by orchestrating smooth interactions that enable optimal performance and decision-making.

2.3.1 What is YARN

To put it simply, YARN, or Yet Another Resource Negotiator, is a ground-breaking framework within the Hadoop ecosystem that brings a new approach to resource management and task scheduling. It divides resource allocation and task execution monitoring into two distinct entities: the central ResourceManager (RM) and specific ApplicationMaster (AM) instances for each application. This separation improves efficiency and flexibility by allowing for equal resource distribution, thorough resource consumption monitoring, and effective job execution throughout the Hadoop cluster.

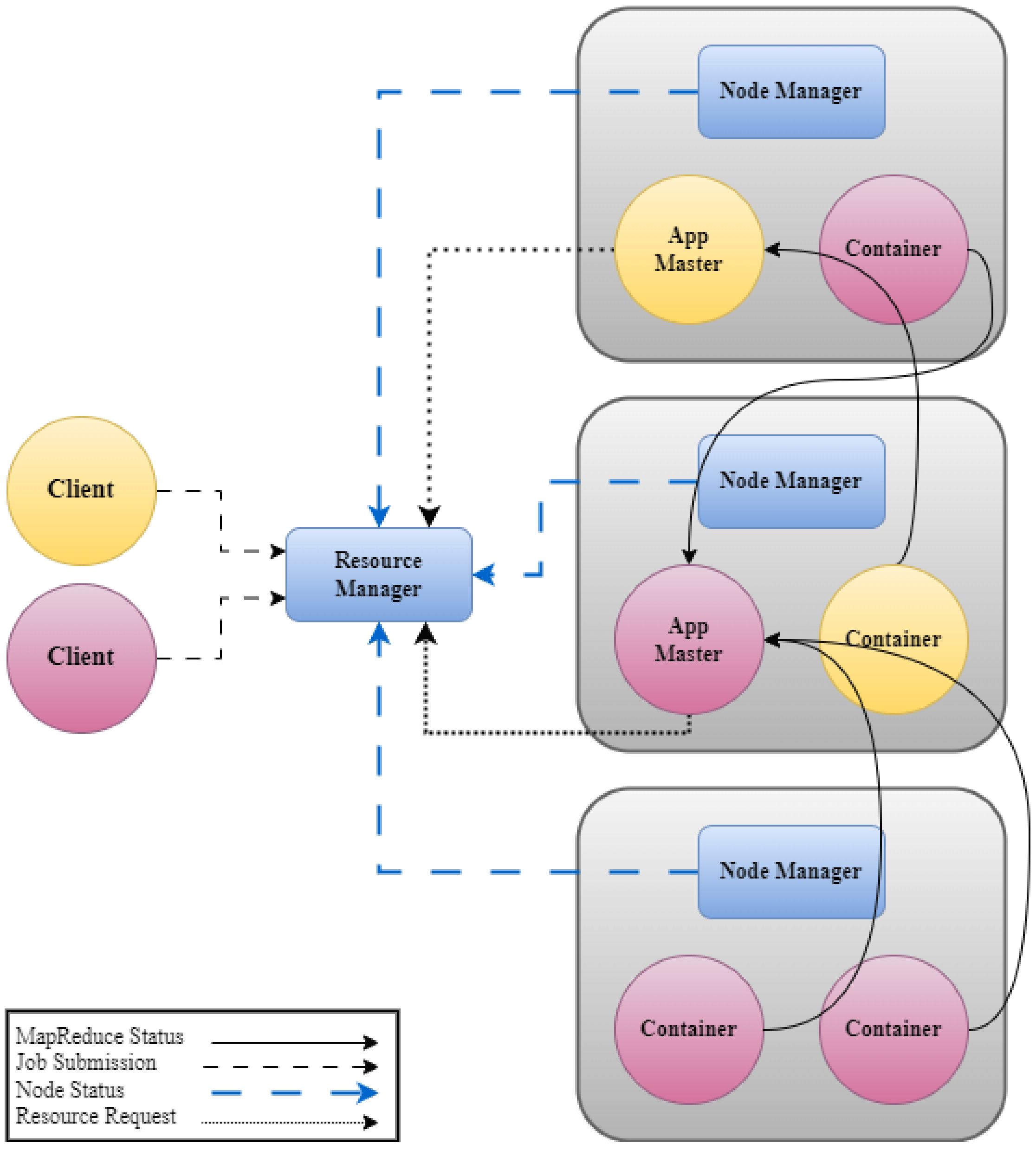


Figure 4: The YARN Architecture [12].

Let us now conduct an in-depth research, digging attentively into the complexity inherent in this subject while adhering to a rigorous and scientific approach. YARN, ushers in a major paradigm shift in Hadoop's resource management and task scheduling. The fundamental premise is to separate the functions of resource allocation and task monitoring into different entities. This novel architecture includes a centralized ResourceManager (RM) and separate ApplicationMaster (AM) instances for each application, whether it is a single task or a complicated job DAG [3], [12 - 13].

The interaction between the ResourceManager and the NodeManager is the foundation of the data-computation framework. The ResourceManager serves as the ultimate authority in the system, managing the distribution of resources across all apps. The NodeManager, on the other hand, acts as a per-machine agent, managing containers and monitoring resource use (CPU, memory, disk, network) and transmitting this information to the ResourceManager and Scheduler [3], [12].

The ApplicationMaster, which is specifically suited to the demands of each application, lies at the heart of YARN's architecture. This specific framework component is in charge of negotiating resources with the ResourceManager and coordinating on task execution and supervision with the NodeManager(s). The varied responsibilities allocated inside YARN result in an optimal environment for various applications, resulting in improved efficiency and performance throughout the Hadoop ecosystem [12 - 13].

2.3.2 ResourceManager’s Two Major Components

Scheduler

The Scheduler, a fundamental component of YARN, manages resource allocation across concurrently running applications while adhering to preset limits such as capacity and queues. It functions as a distinct scheduler, completely focused on allocation and devoid of application state monitoring. This component also does not guarantee the resume of unsuccessful tasks, whether due to application or hardware faults. The Scheduler orchestrates its scheduling activities based on the abstract idea of a "Container", which encapsulates features such as memory, CPU, storage, and network resources. A flexible strategy inside the Scheduler is in charge of dividing cluster resources among various queues and applications. This policy is customizable, and alternative implementations can be used to meet unique needs. Examples of these plug-ins are the CapacityScheduler and the FairScheduler, which both provide unique methods of distributing resources and managing workloads [12 - 13].

ApplicationsManager

The ApplicationsManager plays a pivotal role in the YARN framework. Its responsibilities encompass receiving job submissions, orchestrating the initial container for the application's dedicated ApplicationMaster, and ensuring the recovery of the ApplicationMaster container in the event of failure. The ApplicationMaster, which is unique to each application, is tasked with responsibilities such as getting appropriate resource containers from the Scheduler, checking their status, and vigilantly monitoring the program's overall progress [12 - 13].

2.4 Streaming Data and Twitter

Streaming data has developed as an important route in modern data analysis, providing real-time access to continually developing information. Among the noteworthy providers of such streaming data, the social media site Twitter stands out. Twitter, with its continual stream of tweets, is a wonderful resource for analyzing dynamic trends and ongoing debates. This surge of real-time data creates both obstacles and possibilities, leading enterprises to capitalize on it. In today's digital world, the integration of streaming data, notably from Twitter, has become a critical component in the hunt for up to the minute insights that guide decision-making and create strategy.

2.4.1 What are Streaming Data

Streaming data is the continuous creation of data from a variety of sources, in which data records are transferred concurrently and in relatively tiny amounts, frequently on the kilobyte scale.

Log files generated by user activities in mobile or web applications, online transactions, real-time gaming interactions, social media updates, financial market data, geospatial information, and telemetry from connected devices or data center equipment are all examples of this type of data. This dynamic data flow is critical in current contexts, acting as a useful resource for real-time insights and informed decision-making across a wide range of businesses [22].

2.4.2 Twitter

The concept of streaming data is well shown in the area of social media, particularly Twitter. Twitter demonstrates the continual flow of real-time data, with its enormous user base creating constant streams of brief messages. The dynamic nature of streaming data corresponds nicely with the quick updates and immediate interactions found on sites such as Twitter [23]. Moreover, Twitter originated as a text-based SMS service and for that reason Tweets where restricted to 140 characters [24 - 26]. As Twitter grew, the maximum Tweet length expanded to 280 characters, allowing individuals to share glimpses of their everyday lives and express their thoughts on a wide range of issues, including the economics [24], [26].

Due to a number of compelling elements, Twitter emerged as the best focus point for sentiment analysis study. Twitter, with its large user base [24], provides a rich supply of opinionated textual data that may be easily acquired via its API [27 - 29]. The variety of its user profiles, which include everyday people, celebrities, and politicians, adds to a diverse data landscape. Furthermore, the accumulated corpus includes a vast range of resources across many fields, permitting the analysis of textual content across multiple languages. Because of this convergence of characteristics, Twitter is an excellent target for in-depth sentiment analysis research [16], [24], [29].

In modern times, Twitter is a renowned public network that gathers a large number of tweets from a variety of sources [16]. As communication tools, social media sites such as Twitter carry significant power. The regular entry of thousands of tweets into these digital spaces contributes to the dynamic character of these social networks [30]. It provides a variety of statistics that include tweets, retweets, hashtags, demographic information, and news many times related to hashtags [27].

According to references [27 - 28], the following enumeration highlights Twitter's essential characteristics.

i. **Tweet:** The content encompasses a spectrum of emotions, opinions on events, multimedia elements like photos, videos, and links, among others. This content is readily shareable among the user’s connections [16], [24], [26].

ii. **Retwet:** Stands as a prominent information-sharing method on Twitter, enabling users to repost tweets of interest. This technique usually leaves the original tweets alone, accompanied with a shortened version of the original author's username [26], [31].

iii. **Hashtag:** Hashtags are essential for indexing keywords or subjects on Twitter. This one-of-a-kind tool was pioneered on the platform, allowing users to easily track and engage with subjects of interest [26].

iv. **Search:** Twitter's powerful search tool allows users to explore keywords and phrases, providing real-time access to newly updated tweets related to their interests [32]. This function attracts new users by promoting content discovery and the seamless exchange of relevant information [29].

v. **Follow:** Following is an activity that registered users perform to keep up with people and friends, businesses, or organizations of interest, allowing them to get real-time updates. Twitter's fundamental power resides in its ability to distribute information and ease the tracking of varied accounts, rather than simply interacting with friends and sharing personal experiences [29].

vi. **Handle:** This entails sending updates or public messages to other Twitter users with the syntax "@username." The "@" mark is used to identify a specific individual or entity mentioned in a tweet [31].

As a result of the interweaving of streaming data and platforms such as Twitter, a new era of dynamic communication and information sharing has begun. Twitter's constant supply of real-time data allows users to easily express their ideas, experiences, and updates. As the world of streaming data evolves, the impact on social media platforms like Twitter highlights the critical role these platforms play in defining modern communication and society at large.

2.5 Kafka as a Streaming Platform

Moving on from the complexities of streaming data on networks like Twitter, we get at Kafka, a sophisticated streaming platform that expands on the notion of real-time data interchange. While Twitter is a microcosm of immediate information exchange, Kafka takes this concept to the next level by offering a comprehensive framework for properly organizing and analyzing large amounts of data. Our investigation of Kafka reveals a new level of complication in data processing, increasing our grasp of modern data dynamics and their many applications.

2.5.1 What is Kafka

Apache Kafka stands out as a cornerstone in the complex world of streaming data processing, famous for its ability to effortlessly integrate diverse systems and applications. Kafka has risen to prominence in the data processing industry as a unified, high-throughput, low-latency platform [33 - 36]. It is written in Java and Scala and reflects the open-source framework ethos, processing streams of records in real-time applications. This dual approach comprises building real-time data pipelines across various systems as well as developing applications that dynamically adapt to the constant flow of streaming records. Working in clustered server setups, Kafka collects these records into classified topics, each having a key, value, and timestamp. This architecture promotes good data organization while enabling applications to react quickly to the ever-changing data landscape, firmly establishing Apache Kafka as a cornerstone of the new data processing paradigm [33 - 34].

2.5.2 Decoding Kafka: Simplifying Data Streaming

As it was previously said, Kafka operates as a distributed system, with networked servers and clients communicating through a strong TCP network protocol. Its deployment flexibility spans several platforms, including bare-metal hardware, virtual machines, and containers in both on-premise and cloud environments [34 - 35].

- **Servers:** Kafka operates as a server cluster that spans data centers or cloud regions. This cluster's servers act as brokers for data storage, while others run Kafka Connect for data integration with other systems. Relational databases and other Kafka clusters are included. The architecture is designed for scalability and fault tolerance, making it easy to adapt to essential use cases. In the case of a server failure, the remaining servers take over responsibilities to guarantee that operations continue uninterrupted and that data is not lost [35].
- **Clients:** Kafka provides clients that allow for the creation of distributed applications and microservices capable of handling event streams in parallel and at scale. These clients make it easier to read, write, and process events, guaranteeing fault tolerance even during network outages or machine problems. The Kafka ecosystem offers a wide range of clients, both from the Kafka community and from Kafka itself. These clients support Java, Scala, Go, Python, C/C++, and other languages, as well as REST APIs and the sophisticated Kafka Streams library [35].

2.5.3 Kafka’s Architecture

An event is an occurrence or activity that occurred in the real world or in a business environment. Events, also known as records or messages, are the fundamental units of data in Kafka. Data is read from or written to in the form of events when communicating with Kafka. Each event has a key, a value, a timestamp, and optional metadata headers [33], [35].

Producers and consumers have different and important roles in the Kafka ecosystem. Producers are client applications in charge of writing events to Kafka, whereas consumers read and process these events. Kafka has a clear distinction between the two, which contributes to its famed scalability. In contrast to closely connected systems, producers and consumers work separately, ensuring smooth operation. Kafka also includes strong assurances, such as the ability to process events exactly-once, which improves its dependability and overall speed [33], [35].

Topics provide a hierarchical grouping for events inside Kafka. Topics are analogous to directories in a filesystem, with events functioning as files within those folders. Kafka's uniqueness lies in its multi-producer and multi-subscriber nature for topics. Kafka may accept a variety of scenarios, whether there are zero, one, or many producers producing events or consumers subscribing to them. Unlike traditional messaging systems, Kafka stores events for future access rather than deleting them immediately after consumption. This lifespan is governed by per-topic parameters that specify the period of event retention. Kafka's consistent performance regardless of data size enables event storage to be increased without sacrificing efficiency [33], [35 - 36].

Topics in Kafka undergo partitioning, a process that distributes a topic across distinct "buckets" spread across multiple Kafka brokers. This partitioned distribution is critical for scalability, allowing client applications to read and write data across several brokers at the same time. Each new event that is added to a subject becomes an addition to one of its divisions. Events with similar keys, such as customer or vehicle ID’s, are routed to the same partition, and Kafka assures that consumers of a single topic-partition access events in the same order they were written [33], [35].

Kafka orchestrates its publish and subscribe messaging structure using four unique APIs when deployed as a clustered system on numerous servers: Admin API, Producer API, Consumer API, Kafka Streams API, and Kafka Connect API [33], [35]. In addition,

i. **Admin API:** To supervise and investigate topics, brokers, and other Kafka entities.

ii. **Producer API:** To produce and transmit a continuous stream of events to one or more Kafka topics.

iii. **Consumer API:** To actively participate in the process of subscribing to and reading from one or more topics while digesting the stream of events created by those subjects.

iv. **Kafka Streams API:** Efficiently handling input streams leads to the generation of output streams directed to multiple output topics.

v. **Kafka Connector API:** Using reusable consumers and producers makes it easier to link Kafka topics to existing applications or data systems, which contributes to more efficient operations.

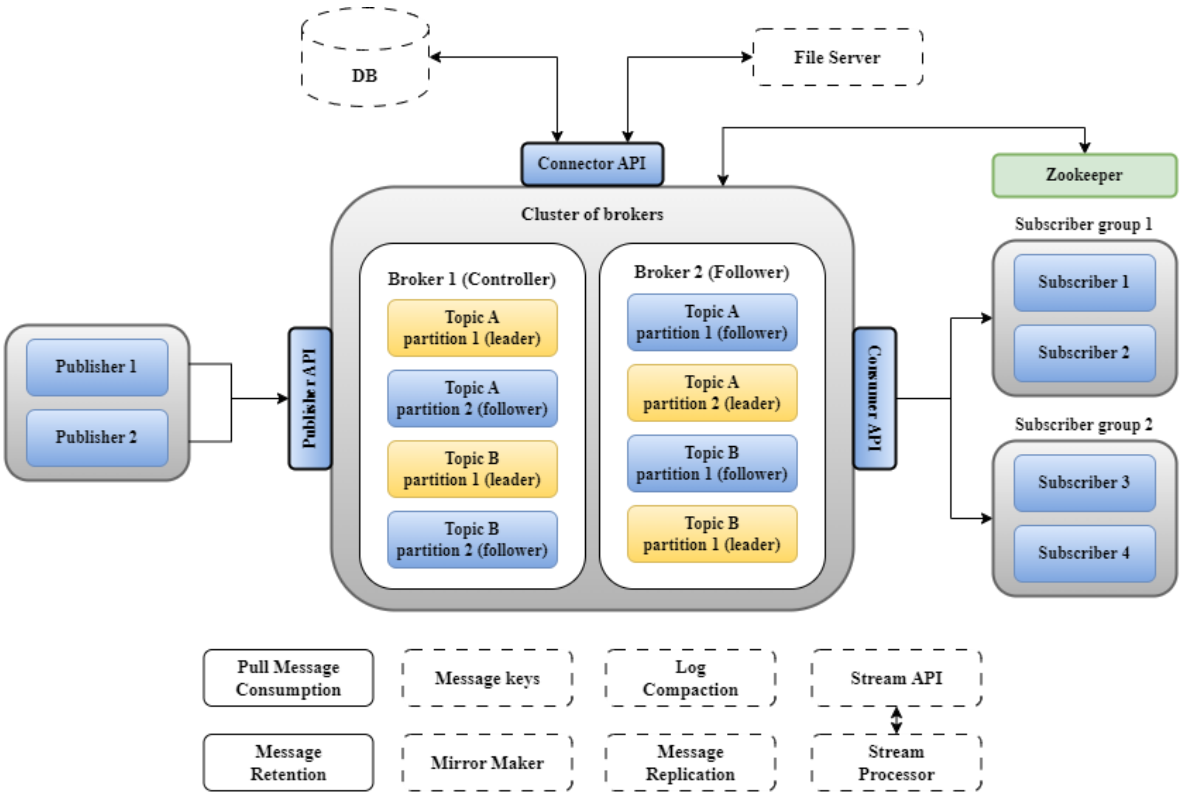


Figure 5: The Kafka Architecture [37].

In conclusion, Apache Kafka is a powerful distributed streaming technology that excels at real-time data handling and event processing at scale. Kafka is essential in high-throughput, fault-tolerant data processing due to its efficient data gathering, storage, and distribution capabilities. Its seamless integration, partitioned topics, and various APIs make it a modern data-engineering cornerstone, allowing different applications to thrive in the dynamic world of continuous data.

2.6 Serialization Techniques for Streaming Data

With our grasp of Kafka's capabilities in place, we can now move on to another key part of streaming data management, which is Serialization Techniques. As data moves across the complex environment of modern information systems, the requirement for efficient packaging and transmission becomes critical. Serialization techniques enable data to be encoded in a way that assures its integrity, compatibility, and efficient transmission between numerous components. These strategies enable for smooth data sharing and effective communication by bridging the gap between various data sources and processing engines. Join us as we explore the topic of serialization and its critical role in streaming data management.

2.6.1 What is Serialization

Serialization is a critical procedure that effortlessly converts complex data structures into a compact sequence of bytes that contains the object's state [13], [38 - 39]. This transformation enables efficient transmission and storage, allowing data to flow smoothly between systems and applications. Serialized data acts as a link between multiple data repositories, in-memory platforms, applications, and other systems, assuring the integrity and accuracy of information as it travels [38].

Serialization serves as a crucial mechanism for storing the state of items and simplifying their reproduction in a variety of settings. This procedure includes both storage and data interchange, effectively encapsulating the numerous components that comprise an object [13], [38]. Handling the complex process of preserving and transferring these components can be difficult, making serialization a popular approach for encapsulating things into a shared format. The smooth transfer of items is made possible via serialization, allowing for [38]:

i. Objects can seamlessly traverse from one application to another across domains thanks to web services such as REST APIs.

ii. Allowing for seamless data transfer and interaction between domains.

iii. During this procedure, security rules are followed, and individual user information is kept across many apps.

iv. Data flow is possible even when firewalls are crossed.

v. Data sharing through network connections is facilitated, particularly for messaging reasons.

2.6.2 Data Serialization on Distributed Systems

Serialization is critical in distributed systems where data and replicas are dispersed across several cluster nodes in diverse partitions. When data is not available locally, systems obtain it from other nodes [13], making Serialization's advantages are particularly valuable in the context of sche-maless big data systems. Such systems store data without strict structural definitions, embracing a more flexible approach. Serialization aids these environments by enabling data to be represented in a portable and transferrable format, facilitating seamless data exchange, even in the absence of predefined schemas serialization critical in situations such as [38]:

i. Transmitting messages to a designated topic.

ii. Inserting elements into a queue, set, or list.

iii. Appending key/value pairs to a map.

iv. Performing operations on map entries

2.6.3 The importance of Data Serialization on Big Data

The benefits of serialization are essential in the context of schemaless big data systems. Such systems use a more flexible approach to data storage, avoiding tight structural definitions. Serialization facilitates data transmission in these situations by allowing data to be represented in a portable and transferable manner, allowing for easy data interchange even in the absence of preset schemas [13], [38 - 39]. Serialization has specific advantages in the context of Big Data systems [38]:

- Serialization helps to structure data by imposing schema or criteria throughout the reading process. This eliminates the possibility of data missing necessary fields, improper classifications, or other quality control standards.
- Serialization provides portability benefits in the world of big data, which comes from numerous systems and is frequently written in multiple computer languages. Data may be serialized and standardized for smooth transfer to other corporate systems or apps.
- Serialization encourages versioning in the ever-changing large data world. It allows for the assignment of version numbers to objects, facilitating effective lifecycle management as data evolves.

2.6.4 Deserialization

Deserialization is the counterpart of serialization, involving the rebuilding of data structures from bytes or strings in order to create objects for consumption. It facilitates the retrieval of object states from storage or transmission, which is critical for data access across devices [13], [39 - 40]. The following figure (figure 6) shows a visual representation of Serialization and Deserialization.

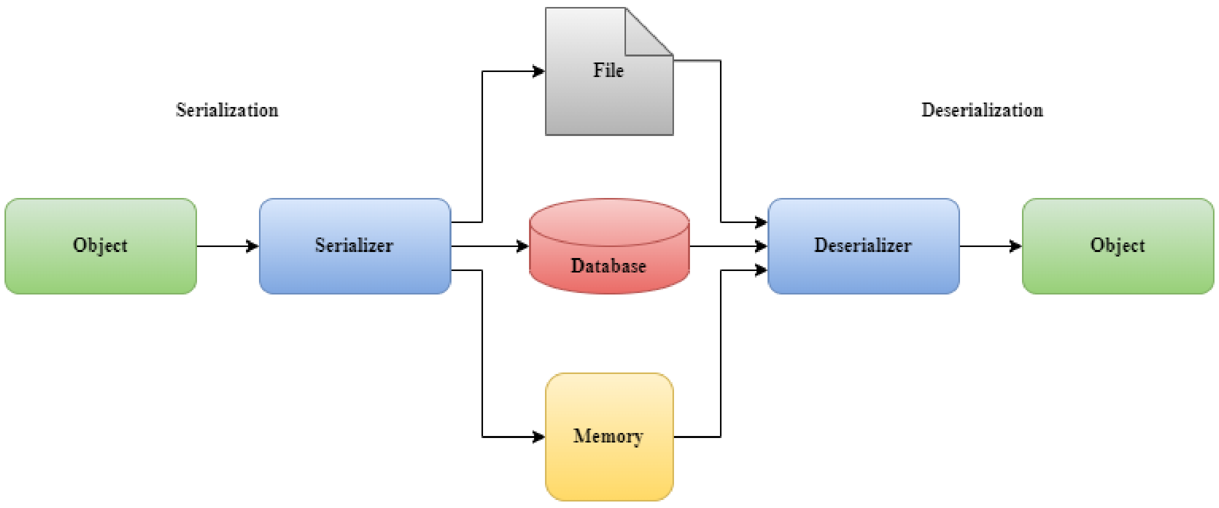


Figure 6: Serialization and Deserialization process [38].

2.6.5 Deserialization on Distributed Systems

Data transmission is required in distributed systems for smooth object sharing among nodes. Serialization streamlines the delivery process by transforming item states into a consistent format. Deserialization then reconstructs items from the serialized form, allowing them to be reconstructed after transmission, even across applications, firewalls, and other network obstacles [13], [38 - 40].

2.6.6 Avro

Moving our focus to Avro, a prime illustration of efficient data serialization. This technology not only simplifies the complex procedures of serialization and deserialization, but it also has the extraordinary capacity to ease schema evolution. Avro is a language independent, schema driven serialization solution that has established its position as the main tool for data serialization inside the Hadoop ecosystem [13], [39], [41]. The key feature is its seamless compatibility with language independent schemas for read and write operations. This orchestration allows Avro to compress data into a small yet elaborately organized binary format, one that unfolds gracefully when intercepted by data-consuming applications [13].

Notably, Avro uses the extensible JSON schema format for hosting schema structures, a format that has been well-received across a variety of languages such as C, C++, Python, Ruby, and others [13], [39], [41]. In addition to its standing, the binary structured composition provided by the Avro toolkit has the combined qualities of compression and "splittability", making it well suited to Hadoop's MapReduce attempts [13], [39], [41]. Avro emerges as a vital option for negotiating the complexities of data transfer and harmonic interoperability within the confines of distributed systems in this vast landscape of adaptability and efficiency [39].

Avro's base is built on the concept of schemas. Every Avro data instance has the schema that was utilized when it was created, eliminating the need for extra per-value expenses. This property leads to quick and efficient serialization, as well as low storage needs. Notably, this property works well with dynamic scripting languages since each piece of data is totally self-descriptive, together with its matching schema [39]. Furthermore, when Avro data is put in a file, the schema that goes with it is integrated, ensuring interoperability with multiple tools for later processing. When a data-reading software predicts a certain schema, flawless resolution is achieved since both schemas are available [39].

Avro provides an efficient schema exchange method during the first connection handshake between the client and server in the area of remote procedure calls (RPC). This complicated method typically eliminates the need for direct schema transfer in the majority of encounters through a meticulously polished procedure boosted by smart optimizations. Issues such as identically named fields, nonexistent characteristics, or more attributes may be easily handled through the exchange of detailed schemas at both endpoints, highlighting the efficiency of this technique [13], [39].

Notably, Avro schemas are thoroughly described in JSON, which not only speeds up implementation but also interacts easily with languages that already have JSON libraries [13], [39], [41]. This method emphasizes Avro's adaptability and compatibility with many programming environments. Avro offers a comprehensive range of features and benefits such as [39]:

- Avro is an optimized binary format that provides data efficiency and fast processing capabilities.

- Avro's capabilities include diverse and complicated data structures.
- Avro provides a specialized container file solution for long-term data storage.
- Avro works smoothly with dynamic languages, removing the requirement for required code generation for data file interactions and RPC protocols, with code generation functioning as an optional optimization best suited for statically typed languages.

3. MapReduce and Data Analysis

We now enter the realm of data analysis after having traversed the intricacies of Big Data Management, the challenges it entails, the architecture of Hadoop Distributed File System (HDFS), the optimization plugins enhancing its efficiency, the realm of streaming data associated with the power of Twitter, and the transformative capabilities of Kafka as a streaming platform. Chapter 3 introduces us to the complex area of MapReduce, revealing its underpinnings and function in processing and extracting useful insights from large datasets. As we go, we will look at the sophisticated processes of clustering and classification, which are critical in obtaining meaningful findings from a massive sea of data. This chapter represents the start of an in-depth look into the area of data analysis.

3.1 Introduction to MapReduce

3.1.1 The Benefits of MapReduce

We continue our investigation by delving into the world of MapReduce; a complicated framework that facilitates the detailed separation of complex data processing operations into simpler, parallelizable components. Through this approach, MapReduce revolutionizes data analysis by providing a scalable and effective alternative for solving sophisticated computations across enormous datasets.

MapReduce emerges as a powerful tool primarily designed for software developers, giving a programming style that effectively navigates the world of large data management. MapReduce, rather than being an independent programming language, functions as a unique paradigm, gaining popularity owing to its user-friendly and effective nature. It seamlessly handles complex activities such as search, indexing, text analysis, graph analysis, data transformation, and machine learning areas where relational databases like basic SQL frequently falls short [3], [10], [13], [20]. The procedural nature of MapReduce promotes comprehension among experienced programmers in these disciplines. Furthermore, it allows developers to avoid the complexities of parallel computing because the system manages this complexity transparently. Notably, non-programmers may use pre-built MapReduce applications and function libraries, demonstrating how the technology has evolved beyond its original programmer-centric design [13], [20].

The MapReduce paradigm is widely used in big data processing, with applications ranging from search indexing to distribution sorting, log analysis, and machine learning. MapReduce's impact has grown beyond its humble beginnings, thanks to Google's pioneering efforts. In the current scenario, industry giants like Microsoft, Yahoo, and Facebook have embraced MapReduce, embedding it into their clusters to power a slew of apps [42]. Furthermore, MapReduce has proven its worth by producing substantial performance improvements, cutting processing times by roughly 30% when compared to alternative methodologies used in data mining and processing [43].

3.1.2 MapReduce in a nutshell

Hadoop MapReduce is a powerful software framework developed to deal with the issues of processing massive volumes of data in parallel across large clusters of cheap hardware. The fundamental principle of MapReduce is to divide input data into self-contained chunks that can be processed simultaneously by individual map tasks. The framework controls the collection of map outputs effectively, which are then integrated into reduce tasks. This technique maximizes data storage and accessibility by storing both input and output in a file system [3], [10], [13], [44 - 45].

One of MapReduce's distinguishing characteristics is its inherent capacity to manage job scheduling, real-time monitoring, and the automated recovery of failed processes. The framework enables continuous processing by re-executing tasks that may fail during execution. This proactive approach to fault tolerance helps to MapReduce's dependability for large-scale data processing [13], [44].

The shared use of computation nodes and storage nodes is an intriguing aspect of MapReduce's design. In this configuration, the MapReduce framework and the HDFS coexist on the same set of nodes [12]. This strategic alignment enhances data locality and results in exceptional aggregate bandwidth throughout the whole cluster. As a result, MapReduce takes use of the architecture to effectively schedule activities on nodes where the relevant data is already stored [13], [44]. The MapReduce framework is made up of various components, including a master ResourceManager, one NodeManager for each cluster node, and an MRAppMaster for each application [46]. These components work together to organize MapReduce job execution, providing efficient task allocation, effective scheduling, and consistent monitoring [13], [44].

Developers interact with the MapReduce framework by specifying input/output locations, implementing map and reduce functions within encapsulated interfaces and classes.The MapReduce process begins when a job, containing configurations, is submitted to the ResourceManager. The ResourceManager facilitates effective data processing by coordinating software distribution, task scheduling, monitoring, and status updates. Finally, Hadoop MapReduce is a solid solution for Big Data issues, using parallel processing, fault tolerance, and data locality for efficient distributed processing [13], [44]. MapReduce is a powerful solution for modern data analytics and processing requirements because to its architecture's competent job allocation, seamless coordination, and resource management. Figure 7 show the MapReduce architecture.

3.1.3 The Input-Output and Key-Value Pair Relationship

The MapReduce framework is built on processing <key, value> pairs, with the input and output of a job represented as sets of these pairs. This design underscores the significance of serialization, demanding that the key and value classes implement the Writable interface to maintain framework compatibility. Furthermore, key classes should implement the WritableComparable interface in order for the framework to sort them efficiently [13], [44]. Mapping Input and Output Dynamics in MapReduce Jobs [44]:

(input) <k1, v1> → map → <k2, v2> → combine → <k2, v2> → reduce → <k3, v3> (output)

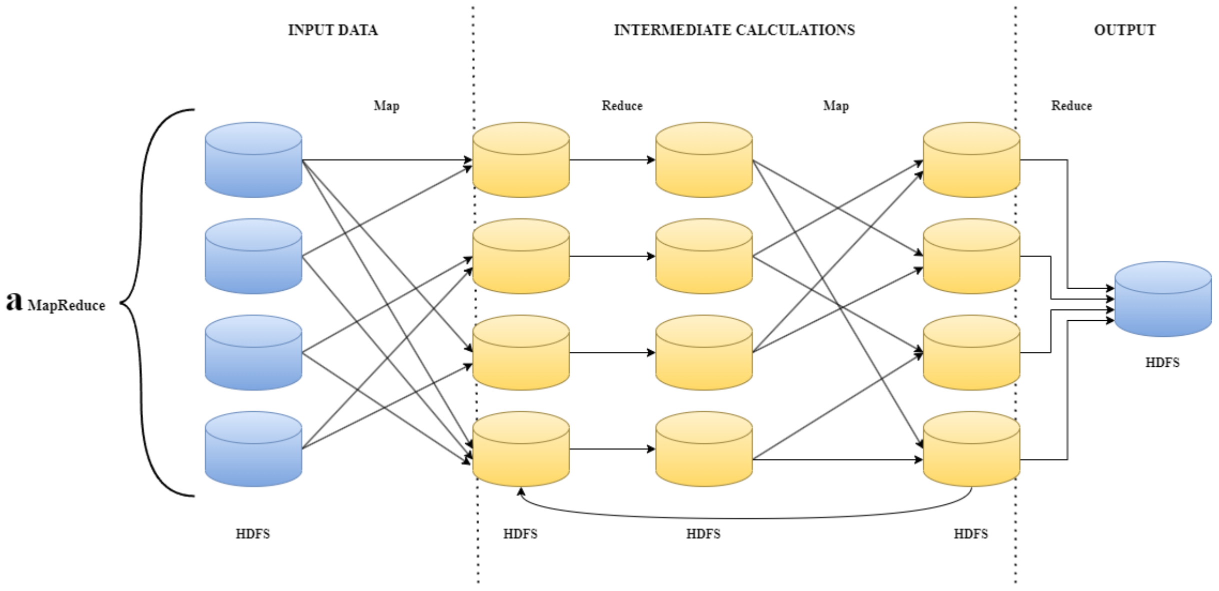


Figure 7: The MapReduce Architecture [3].

3.1.4 Deconstructing MapReduce

Applications in the MapReduce domain regularly interact with the Mapper and Reducer interfaces to methodically create and integrate the necessary map and reduce methods. These approaches represent the critical basis of the task, driving the complicated procedures that enable precise and efficient data translation and aggregation.

The Mapper

The Mapper plays an important role in Hadoop's MapReduce framework, managing the transformation of input <key, value> pairs into a succession of intermediate <key, value> pairs. Individual map jobs perform these transformational procedures, which change input data into intermediary records that are unique from their initial counterparts [13], [44].

Following the production of InputSplit by the specified InputFormat, map jobs are assigned. The Job.setMapperClass(Class) function seamlessly integrates mapper implementations into jobs. Following that, for each <key, value> combination in the InputSplit, the framework calls the map(WritableComparable, Writable, Context) function [13], [44].

Context.write(WritableComparable, Writable) calls are used to gather output pairs that may differ in structure from their input counterparts. In the meanwhile, counters may be used to report application data, offering insight into the execution process and assisting with performance evaluation [13], [44].

As the outputs of Mappers are sorted and partitioned, the role of Reducers comes into play. They process the aggregated intermediate values and help to produce the final product. The intermediate findings are rigorously saved in a standardized (key-len, key, value-len, value) format to simplify further processing steps. Through the Configuration, applications can retain control over compression and CompressionCodec use, maximizing storage and data transport efficiency [13], [44].

Additionally, the use of custom Partitioners and Combiners improves control over intermediate data handling. Custom Partitioners control which keys are given to which Reducers, allowing for more efficient aggregation and less data transmission. Furthermore, the integration of Combiners was made possible by Job.setCombinerClass(Class), which enables local aggregation of intermediate outputs. This intermediate aggregation minimizes the amount of data transmitted from the Mapper to the Reducer, resulting in faster processing and better resource efficiency [13], [44].

Map Count: Enhancing Simplicity and Efficiency

The quantity of map tasks is typically influenced by the cumulative size of the inputs, corresponding to the total number of blocks constituting the input files. Striking the optimal balance of parallelism for map tasks appears to fall within the range of 10 to 100 maps per node; however, certain cases need up to 300 maps for tasks with low CPU burden. An adequate execution duration for each map job is required for efficient task distribution. Map jobs should ideally take at least a minute to finish, taking into account the setup time [13], [44].

Consider the following situation, based on information from Apache's official documentation for MapReduce [44]:

- Intended input data size is 10TB.
- Blocksize is 128MB (see unit 2.2.9).

As a result of this arrangement, about 82,000 maps are required for proper processing. However, developers can use Configuration.set(MRJobConfig.NUM_MAPS, int) to offer the framework with a particular optimization proposal for cases requiring more parallelism.

The Reducer

The Reducer plays an essential part in the complex architecture of MapReduce. It performs the critical role of compressing an array of intermediate values, which are naturally connected by a shared key, into a dataset with increased compactness and efficiency [13], [44].

By using the Job.setNumReduceTasks(int) function, the number of reducers assigned to a certain job can be adjusted to the work by the user. Job.SetReducerClass(Class) allows developers to control the behavior of the Reducer within the task, laying the groundwork for its execution. Then, for each <key, (list of values)> pair in the grouped inputs, the framework handles the execution of the reduce(WritableComparable, Iterable<Writable>, Context) function. The cleanup(Context) function can be configured to do any necessary post-processing tasks [13], [44].

The Reducer's job entails three important stages [13], [44]:

i. **Shuffle:** As input, the Reducer is given the sorted results produced by the mappers. During this step, the framework uses HTTP to fetch the specific partition of the mappers output that is relevant to the Reducer's task.

ii. **Sort:** During this stage, the framework organizes the Reducer inputs by their respective keys, accounting for the possibility of different mappers producing identical keys. The shuffling and sorting phases run in parallel, elegantly combining the map outputs as they are fetched.

- **Secondary Sort:** The framework provides a solution in circumstances where different equivalence rules are required for grouping intermediate keys than grouping keys before reduction. This can be accomplished by utilizing the Job.setSortComparatorClass(Class) function to provide a Comparator. This is supplemented by the Job.setGroupingComparatorClass(Class) method, which controls how intermediate keys are grouped. These mechanisms can be used together to effectively simulate a secondary sort based on values.

iii. **Reduce:** The reduce(WritableComparable, Iterable<Writable>, Context) method is executed for every <key, (list of values)> pair inside the grouped inputs at this critical phase. Context.write(WritableComparable, Writable) is commonly used to record the output of the reduction job in the FileSystem. Applications that use Counters can efficiently report useful statistics for analysis. It is vital to note that the Reducer's output is not sorted.

Reduce Counts: Enhancing Simplicity and Efficiency

The number of reduces can be achieved by considering factors such as 0.95 or 1.75 multiplied by the product of the number of nodes and the maximum containers per node. While using 0.95 all reduces begin immediately, accelerating the transmission of map outputs as they finish. Alternatively, selecting 1.75 causes faster nodes to finish their initial reduces and then launch a second wave of reduces, considerably improving load balancing [44].

The scaling factors offered are purposefully set slightly below whole numbers in order to assign a reserved fraction of the framework's reduction slots. This allocation serves to accommodate speculative tasks and mitigate the impact of potential failures [44].

When no reduction is intended, the count of reduce jobs can be set to zero. In such cases, map task outputs are immediately directed to the FileSystem, following the output path defined by FileOutputFormatsetOutputPath(Job, Path). Notably, the framework skips sorting map outputs before storing them in the FileSystem [44].

Partitioner

The Partitioner is crucial in segmenting the key space, controlling the distribution of intermediate map-output keys. It uses the key, or a piece of it, to compute the partition, which is often achieved using a hash function. This operation results in the generation of partitions equivalent in number to the reduce tasks assigned to the job. As a result, the Partitioner manages the allocation of intermediate keys (and their associated records) to the appropriate reduce jobs for reduction. By default, the HashPartitioner takes up this role [44].

In conclusion, the inspection of the MapReduce architecture emphasizes its critical significance in current data processing and analysis. MapReduce has transformed the way big data is handled and insights are extracted through its complicated orchestration of mapping and reduction processes, along with its parallel processing capabilities. It has become a cornerstone of Big Data technologies due to its capacity to distribute workloads across clusters of nodes while maintaining fault tolerance and effective resource use. MapReduce enables organizations to discover significant patterns, trends, and correlations among enormous datasets by offering a standardized framework for developers to use, boosting innovations across multiple industries.

3.2 Clustering Techniques for Result Extraction

Following our extensive review of the MapReduce structure, the following chapter turns its attention to the difficult topic of Clustering Techniques for Result Extraction. This section delves into the challenges of applying clustering algorithms in order to extract significant insights and patterns from processed data. This chapter explains the advanced tactics used to uncover relevant information for a variety of analytical activities by illuminating the subtleties of data classification and grouping.

Clustering Analysis entails splitting a dataset into numerous groups, where internal data similarity within a group outweighs external data similarity [47]. This approach has the potential to uncover underlying data structures, with applications ranging from classification to image processing to pattern recognition [48]. Clustering is an unsupervised data mining strategy that eliminates the requirement for training or preset output objectives. It has the ability to reveal cohesive data subsets with comparable qualities, allowing for a plethora of analytical options [49].

In preparation for digging into preprocessing strategies for best clustering results, it is crucial to recognize its critical role in the process. However, for a more in-depth investigation, please see Unit 3.3.1.

3.2.1 K-Means

K-means clustering has emerged as a cornerstone of data mining, as evidenced by a number of convincing considerations. The K-means method orchestrates the partitioning of a dataset into K-clusters, with the twin aim of increasing compactness within each cluster and minimizing separation among the K-clusters [50]. This approach aims to divide the dataset into K-clusters, with each observation allocated to the cluster with the closest mean. K-means outperforms hierarchical clustering in processing large datasets. It is notable as a non-probabilistic clustering technique with a time complexity of O(n), giving it computational supremacy over Gaussian mixtures. This computing capability places K-means as a superior alternative for a wide range of applications, as proven by recent cryptocurrency analyses [51].

The clustering technique begins with the identification of the dataset to be clustered, indicated as $X_{ij}(i = 1,..., n; j = 1,..., m)$, where n demonstrates the amount of data to be clustered and m represents the number of variables. The iterative procedure begins with the formation of initial centroids for each cluster, denoted as $C_{kj}(k = 1,..., k; j = 1,..., m)$, which are often picked at random. The correlation between each data point and each cluster centroid is then determined [48]. This establishes the framework for the K-means algorithm, as detailed in the following essential steps [48 - 49], [51 - 52]:

(1) K centroids are chosen by picking K rows at random from the dataset. To illustrate, academics frequently use three clusters to characterize visitor intake into three unique categories: tranquil, moderate, and crowded [48].

(2) Choose K objects at random from the dataset to serve as the first cluster centers or means.

(3) Allocate each data point to its nearest centroid, based on the Euclidean distance between the item and the centroid.

$$d(i,k) = \sqrt{\sum_i^m \left(C_{ij} - C_{kj}\right)^2}$$

(4) The cluster centroid for each of the K-clusters is updated by computing fresh mean values taken from all data points within the cluster. The centroid of the K'th cluster is a vector of length p that contains the mean values of all variables related to observations inside that cluster. In this case, p specifies the number of variables under consideration.

(5) Continue iterating through steps 3 and 4 until either the cluster assignments are stable or the maximum number of iterations is reached.

In order to avoid the arbitrary selection of the number of clusters (k) in the K-means clustering process, various methods provide more systematic approaches. One such technique is the "Elbow Method", which offers a strategic way to determine the optimal value of k for achieving meaningful and effective clustering results.

This approach makes use of distance metrics, calculating the squared distance between data points and their centroids for a variety of "k" values. A series of K values are assessed through iterative assessment utilizing the Sum of Squared Errors (SSE), which acts as an efficacy metric. By visualizing the K-SSE curve and locating the "elbow point", where the decline rate changes, the best "k" is established [52]. The following is an illustration of the K-SSE curve:

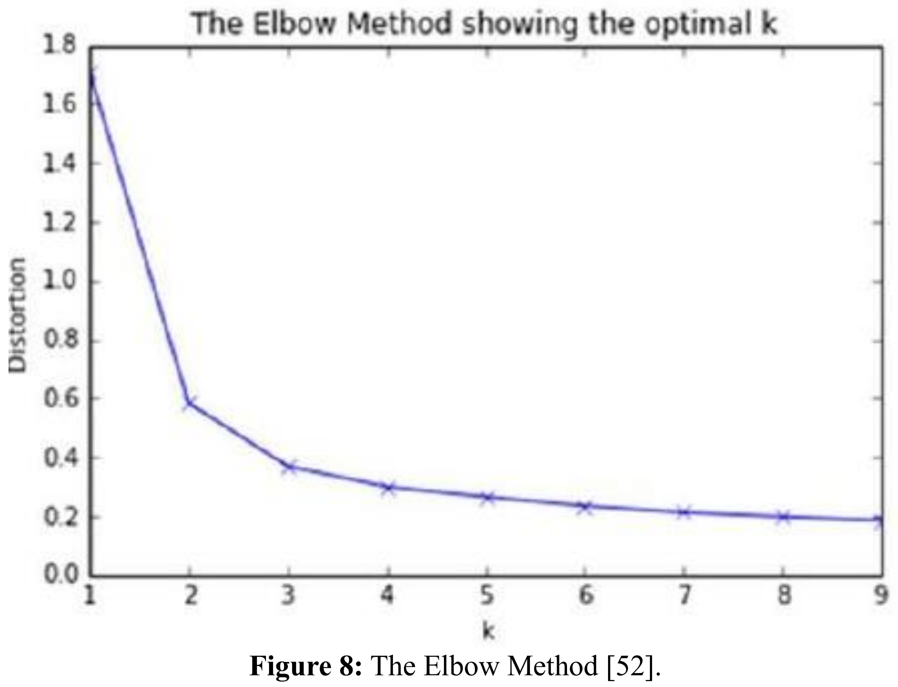


Figure 8: The Elbow Method [52].

3.2.2 Gaussian mixture model (GMM)

The Gaussian Mixture Model (GMM) has various benefits over the traditional k-means clustering approach. GMM prioritizes data covariance, allowing it to successfully accept datasets with various distribution shapes. Unlike K-means, which uses rigorous classification, GMM uses soft classification. While K-means assigns definite categories in this situation, GMM assigns probabilities to data points across specified categories. As a consequence, GMM not only gives classifications but also evaluates the likelihood of data points belonging to specific categories. This characteristic enables GMM to assess the degree of uncertainty associated with the relationship between data points and certain categories [53].

GMM makes use of a composite or fusion of many Gaussian distributions. Given the Gaussian distribution's general application to a wide range of datasets, it is plausible to infer that clusters have different Gaussian distributions [53 - 54]. Within this distribution type, the probability density function for one dimension is given as follows:

$$f(X|\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(X-\mu)^2}{2\sigma^2}}$$

In this context, μ and σ represent the distribution's mean and standard deviation, respectively [38, 44]. The probability density function for a multidimensional d-variate dimension is stated as follows:

$$f(X|\mu,\sigma) = \frac{1}{\sqrt{2\pi}|\Sigma|} e^{-\left(\frac{1}{2}(X-\mu)^T\Sigma^{-1}(X-\mu)\right)}$$

The parameter μ represents the mean of the distribution as a d-dimensional array. The symbol Σ represents X's covariance matrix, whereas T represents vector transposition and -1 indicates matrix inverse. In cases containing k clusters, μ and Σ are estimated for each individual cluster, indicated by k. In situations when there is just one distribution, the maximum-likelihood approach is used to estimate, but when numerous clusters are involved, the probability density becomes a linear function formed from the distribution densities of all clusters [53 - 54].

$$p(X) = \sum_{k=1}^{K} \pi_k\, f(X|\mu_k, \Sigma_k)$$

π_k is the mixing coefficient for the distribution of cluster k. We may compute the following using the maximum log-likelihood method [53 - 54]:

$$p(X|\mu,\Sigma,\pi) = \sum_{i=1}^{N} ln\left(\sum_{k=1}^{K} \pi_k\, f(X_i|\mu_k, \Sigma_k)\right)$$

Following that, the next stage includes the creation of a random variable β_k(X) = p(k|X), followed by the application of Bayes' theorem [53 - 54]:

$$\beta_k(X) = \frac{p(X|k)\pi_k}{\sum_{k=1}^{K} \pi_k\, p(X|k)}$$

The log-likelihood function reaches its maximum value when it's derivative with regard to μ and π, p(X|μ, Σ, π), equals 0. As a result, we may get parameter estimates using the following formulations [53 - 54]:

$$\mu_k = \frac{\sum_{n=1}^{N} \beta_k(x_n)x_n}{\sum_{n=1}^{N} \beta_k(x_n)},$$

$$\Sigma_k = \frac{\sum_{n=1}^{N} \beta_k(x_n-\mu_n)(x_n-\mu_n)^T}{\sum_{n=1}^{N} \beta_k(x_n)},$$

$$\pi_n = \frac{1}{N}\sum_{n=1}^{N} \beta_k\,(x_n)$$

The summation $\sum_{n=1}^{N} \beta_k\,(x_n)$, represents the total count of items within cluster k, while N signifies the total number of samples x_i. To effectively cluster data, it is necessary to estimate the Gaussian parameters, and identifying sample allocations to certain clusters becomes critical for this purpose. As a result, using the expectation-maximization approach becomes mandatory. This involves assigning arbitrary values to μ_k, Σ_k, and k, followed by the estimate of latent variables (β_k) and subsequent parameter updates [53 - 54].

3.3 Classification Techniques for Result Extraction

As we continue our research into data analysis, we come across another crucial aspect; classification. Classification is a crucial part in the field of data mining. This supervised method utilizes labeled training data to detect patterns and properties, with the goal of classifying objects while optimizing accuracy gains.

The procedure involves developing correlations between input and output properties, resulting in the creation of a model during training [55]. Data is meticulously classified using categorization based on evaluated similarities and differences. As it tackles the problem of identifying class labels for ambiguous objects, this approach becomes vital [56]. As a result, classification is a critical tool in data analysis, allowing for the rough evaluation of unknown object classifications, the categorization and grouping of data into multiple ideas or groups [49], [57].

Based on that, classification is essential for activities such as sentiment and opinion analysis [24]. Classification aids in categorizing text data in order to comprehend thoughts and ideas stated in various kinds of text, such as reviews or social media postings. This helps firms understand client preferences and trends, resulting in better decisions and more successful communication methods. In brief, classification makes it easier to extract significant insights from text, allowing for a better comprehension of people's thoughts and points of view [10], [24], [58].

Thus, as we evolve around sentiment and opinion analysis, it is evident why classification is so important. Classification techniques are particularly useful in activities such as interpreting client attitudes, product evaluations, and comments on platforms such as Twitter [24].

3.3.1 Preprocessing Techniques

Analyzing Twitter data has its own set of difficulties. Before it can be utilized for analysis, the data must be thoroughly prepared. This procedure, known as preprocessing, employs a variety of approaches to prepare the data for analysis. On a greater scale, this has resulted in the development of numerous methods for processing data and developing effective algorithms for sentiment analysis. Researchers have also looked at new methods of improving the outcomes, making the analysis more efficient and dependable. All of this emphasizes the importance of preprocessing in making sense of Twitter data and gaining relevant insights. Several of these techniques include [16], [24]:

i. **Tokenization:** The documents are separated into words or phrases, resulting in a word vector referred as a "bag-of-words".

ii. **Stemming:** Stemming algorithms work through eliminating word suffixes that comply with grammatical standards.

iii. **Punctuation Removal:** Is a text analysis preparation technique that includes removing punctuation symbols from words since they do not add relevant information.

iv. **Stop Words Removal:** This technique involves the removal of commonly used words that hold little meaning and significance in text classification. This method significantly decreases the size of the corpus while keeping critical information.

v. **TF-IDF:** TF-IDF, which stands for "term frequency-inverse document frequency," is a common approach for building feature vectors. This numerical measure evaluates the importance of a word in a document in respect to a corpus.

3.3.2 K-Nearest Neighbor (KNN)

To identify objects based on their closest distances, K-Nearest Neighbor (K-NN) is a critical step that requires the assistance of training data. The KNN technique predicts outcomes by finding the nearest neighbors using distance measurement [24], [59]. The smallest distance between the assessed information and the values of K neighbors within the training data is calculated using this method [49], [60].

The following steps are involved in implementing the K-NN method [49], [57]:

(1) Establish the parameter 'k' denoting the count of nearest neighbors.

(2) Calculate the object's squared Euclidean distance from the given training data. The formula is given below:

$$d_i = \sqrt{\sum_{i=1}^{n} (x_{ij} - p_j)^2}, \text{ where:}$$

- x_{ij} = Training data
- p_j = Testing data
- n = Number of Samples

(3) In descending order, arrange the squared distances determined in step 2.

(4) Obtaining the classification groups of the nearest neighbors depending on the value of k selected on step 1.

(5) Anticipate object classifications by using the nearest neighbor's category with the greatest corresponding value.

3.3.3 Support Vector Machines (SVM)

The Support Vector Machine (SVM) is a well-known two-classification model with a long past. When using SVM to segment gathered data, the main goal is to find an acceptable hyperplane. This segmentation seeks to optimize the interval, which includes both soft and hard intervals, and is then converted into a unique quadratic programming problem for resolution [20], [24], [61].

Let's assume that T denotes the training dataset, which comprises of pairings $(x_1 y_1), (x_1 y_1), \ldots, (x_N, y_N)$ where $x_i \in R^n$ and $y_i \in \{+1, -1\}, i = 1,2,3, \ldots, N$. In this case, x_i represents the feature vector of the i-th sample spectrum, whereas y_i corresponds to the task-related labels. Support Vector Machines (SVM) are designed to categorize the hyperplane, which is represented as $wx^T + b = 0$. This requires dealing with the optimization issue for the Lagrangian multiplier using Lagrangian duality theory. A penalty factor C is established to allow for mistake tolerance. When the data cannot be separated linearly, a kernel function K(x, z) is used to map the sample space into a higher-dimensional realm. The following equation depicts the unique objective function [61].

$$\min_{a} \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \, \alpha_j y_i y_j (x_i x_j) - \sum_{i=1}^{N} \alpha_i \, s.t. \sum_{i=1}^{N} \alpha_i \, y_i = 0, C \geq \alpha_i \geq 0, i = 1,2,3, \ldots, N$$

The sequential minimum optimization procedure is used to establish the decision plane, which is as follows:

$$f(x) = sign\left(\sum_{i=1}^{N} \alpha_i^T \, y_i (x x_i) + b \right)$$

3.3.4 Performance Evaluation

When reviewing and measuring the performance of a classification model, focusing exclusively on a single metric such as classification accuracy is insufficient to offer a complete picture of classification efficacy. To provide a more comprehensive perspective, evaluation measures must include accuracy, recall rate, and F1 score [62]. These four metrics are represented by the following formulae. The following formulas reflect these four measures:

i. $Accurancy = \frac{TP+TN}{TP+FP+FN+TN}$

ii. $Precision = \frac{TP}{TP+FP}$

iii. $RecallRate = \frac{TP}{TP+FN}$

iv. $F1_{score} = \frac{2 \times Precision \times Recall}{Precision+Recall}$

where,

- TP → True Positives
- FP → False Positives
- TN → True Negatives
- FN → False Negatives

4. Methodology

This chapter examines the core of this study, where preparation and execution pave the way for the accomplishment of our goals. This methodology chapter outlines the way we employed to address the numerous issues given by managing big datasets, improving data processing efficiency, and extracting useful insights from real-time streaming data sources.

We begin by acknowledging the challenges we come across related to Big Data Management at unit 4.1. Following that, we examine how we collected data (Unit 4.2). We will additionally demonstrate the dataset structure we chose (Unit 4.3) along with the critical data preprocessing work, we performed (Unit 4.4) to ensure our analysis is accurate and meaningful.

We will examine Exploratory Data Analysis (Unit 4.5), Feature Engineering (Unit 4.6), and Data Analysis (Unit 4.7) in the sections that follow. In this part, we will look over the steps we used to discover patterns and insights in our dataset. Our goal is to provide a complete overview of the strategies used to transform raw data into useful knowledge. In Unit 4.8, we shift our focus to Results Interpretation, where we present the results of our data analysis and evaluate how they relate in the context of our study objectives. This step serves as a connection between data analysis and validation (Unit 4.9), confirming the reliability of our results. Our methodology was developed to be transparent while also closely aligned with the study's objectives. Each step in our research process is documented and explained to guarantee transparency and consistency.

4.1 Problem Definition

Every research project begins by defining the problems that need to be solved, which should be related to the study objectives. In this section, we will discuss the challenges that inspired this study. These challenges include data management, dealing with complicated data processing, providing scalability, and performing real-time analysis. These challenges have an unbreakable connection to our research's goals and objectives, and they serve as the foundation for our methodical analysis.

4.1.1 Data Management Challenges

Our main challenge lies in managing of large datasets, which is associated with Objective 1: Understanding and addressing obstacles in the management of large and diverse datasets. These datasets include a wide range of data kinds, including structured, semi-structured, and unstructured data. The complexities of dealing with such data require a thoughtful approach that takes into account all aspects of data storage, management, and retrieval.

4.1.2 Processing Complexity

Aligned with Objective 2, we address the complexities of data processing in the big data landscape by thoroughly examining the fundamental operational procedures of HDFS, delving into its architectural complexity in conjunction with MapReduce integration. The enormous volume of data needs efficient processing algorithms that make use of distributed computing.

4.1.3 Scalability and Real-Time Analysis

In line with Objective 3, we aim to collect timely insights from real-time streaming data sources, with a particular emphasis on platforms like Twitter. This objective focuses our attention on the issues of scalability and real-time analysis. Our research focuses on streaming data sources and the important part they play in guiding decision-making processes as we aim for seamless scalability and real-time insights.

4.2 Data Collection

To gain access to real-time streaming data, we used the Kafka streaming platform, an extraordinarily robust and scalable solution known for its ability to manage high-throughput data ingestion. We adjusted a Kafka topic and partition approach that was ideally fit with the complexities of our study aims.

4.2.1 Kafka Topic and Partition Configuration

Topic

We employed a single Kafka topic to serve as an interface for incoming data streams during our data collecting process. The topic is created to correspond with the exact data sources we wanted to monitor, creating an efficient collection process.

Partition

We used a single partition within the Kafka topic for our data collection. We adopted this approach to ensure that the data volume within the scope of our study remained manageable. We wanted to simplify data processing and analysis by reducing the number of partitions to one, lowering complexity without compromising speed.

4.2.2 Broker Configuration

We used a single Kafka broker for easy data input. This setup ensured efficient data transfer while reducing resource overhead. It was consistent with the size of data creation from our chosen source, Twitter, utilizing gathering data for the goals of our study.

Data Ingestion Process

Data were ingested into the Kafka topic using producer clients, which were configured to connect to the designated broker. The producer clients were created to retrieve data from our chosen source, Twitter, turn it into a suitable format, and then feed it into the Kafka topic. We were able to keep our data collection architecture simple by using a single topic, a single partition, and a single broker. This simplicity was extremely helpful for our research since it decreased potential causes of failure, simplified troubleshooting, and provided a smooth data flow. In the following sections, we will look over the data preprocessing and analysis steps, demonstrating how this streamlined data collection approach seamlessly integrates with our overall research methodology.

4.3 Dataset Presentation

In this section, we present an extensive overview of the dataset that was collected for this study. The collection is made up of Twitter data, with special focus on tweet locations both within and outside the United States. The objective of data collection is to track regions in the United States that are following cryptocurrency trends.

4.3.1 Data Collection

To assure both the depth and breadth of our dataset, we used a variety of data collecting methodologies:

i. **Twitter API:** We used the Twitter API to capture tweets containing particular cryptocurrency terms like "ADA", "BTC", "ETH" and "CRYPTO." This enabled us to capture talks and debates about these digital assets.
ii. **Location-Based Filtering:** We established filters to gather tweets with user locations in order to restrict our emphasis on geographical data. Tweets from within the United States and tweets from outside the United States were classified individually.
iii. **Location Categories:** Tweets with non-US origins were grouped into a single register to reflect their distinct geographical origins. Tweets in the United States were further classified by state, resulting in 50 different state-based records.

4.3.2 Dataset Characteristics

To provide insight into the dataset's characteristics, we include the following details:

i. **Size:** Our dataset consists of 2,500 registries, each of which represents a unique tweet. We found 51 distinct values reflecting unique places in this dataset, accounting for the 50 states in the US as well as an extra category for tweets originating outside the US.
ii. **Location Data:** Specifically, we collect and preserve the location of the tweet as a single-line item.

The way how we sorted and processed this information serves as the foundation for the rest of our study. It assists us in analyzing cryptocurrency talks in various locations. In the sections that follow, we will discuss how we processed and examined the data, which is critical for identifying patterns and insights.

4.4 Data Preprocessing

To facilitate our research, we systematically characterized tweet locations retrieved from Twitter throughout the data preprocessing phase. Our strategy was as follows:

i. Locations were categorized by comparing them to a predetermined list of US state names and postal codes.
ii. Index numbers ranging from 1 to 50 were assigned to locations that matched US states or postal codes.

iii. Locations outside the United States were grouped together and given the index number of 51, denoting "outside of US".

4.5 Exploratory Data Analysis

In this section, we are going to examine the location data obtained for our crypto-currency trend study. The main objective of this step is to obtain an in-depth understanding of the data's characteristics and to discover any useful insights that could supply our upcoming clustering and classification work. The dataset has been modified after the data preprocessing steps and is now prepared for its next processing stages.

Our exploratory data analysis (EDA) approach starts with an in-depth examination of the distribution of location data as well as the number of occurrences for each numerical identity. We are also looking for any notable patterns or trends that emerge from this first investigation. Furthermore, we check for any unexpected observations or anomalies that may have an influence on the quality of our study. While our EDA is primarily concerned with summarizing the dataset and comprehending its fundamental properties, it serves as a basis for the more complex phases of clustering and classification. We want to make use of EDA to discover any early ideas that could impact the rest methods of analysis. This step is important for us to better understand how cryptocurrency trends differ depending on where they are occur in the world.

4.6 Feature Engineering

In the realm of feature engineering, we improve the quality of our dataset in order to best prepare it for further examination. While several significant components were covered in prior sections (Units 4.2 and 4.3) and are mentioned again here for context, this section focuses on additional engineering steps.

i. **Categorization of US States (See Unit 4.2):** We had previously categorized the data based on US states, using number labels to identify patterns within the US.
ii. **Kafka Data Collection (Refer to Unit 4.3):** As mentioned in Section 4.3, Kafka effectively gathers cryptocurrencies trend data from Twitter.

Building upon these foundational aspects, we now detail the specific feature engineering steps we used to enhance and expand our dataset for analysis:

iii. **MapReduce job and HDFS Storage (see Unit 4.7.1):** Our MapReduce step processes the data after it has been collected, and the results are safely stored in HDFS. This step guarantees that the data is set up for further investigation.
iv. **Classification and clustering (see Unit 4.7.2):** The feature engineering process extends into the clustering and classification phases, where we use algorithms like K-Means, GMM, K-Nearest Neighbor (KNN), and Support Vector Machines (SVM) to enhance our dataset in order to obtain new insights.

4.7 Data Analysis

In this section, we take an in-depth look at the data analysis stage, where we employ approaches to obtain useful information from the acquired location data. This analysis step is strongly tied to our research aims, which include tackling difficulties related to data processing complexity and real-time analysis.

4.7.1 HDFS and MapReduce Analysis

We conducted a study on the dataset using the Hadoop Distributed File System (HDFS) and the MapReduce framework to acquire an in-depth understanding of the data's underlying structure and potential scalability. This work is an important first step in resolving the issues related with data storage and processing complexity.

- **Data Processing in HDFS:** We look at how data is stored and maintained in the HDFS environment, with special focus on its unique characteristics like data replication and fault tolerance. This investigation gives information on how HDFS solves data storage and reliability concerns (Objective 2).
- **MapReduce Framework:** We go into the MapReduce paradigm, analyzing its key components such as the Mapper and the Reducer. This study helps us in understanding how MapReduce can be utilized to effectively manage massive datasets, which aligns with our objective of increasing data processing speed and efficiency (Objective 2).

4.7.2 Clustering and Classification Analysis

We proceed with elegant data analysis techniques, specifically clustering and classification, after completing our basic study. This phase aligns with our greater goal of developing a robust infrastructure capable of managing real-world streaming data, which is stated in our research objecttives.

- **Clustering Analysis:** We apply K-Means and Gaussian Mixture Model (GMM) clustering techniques to identify inherent patterns within our data. The findings of this study are shown using scatter plots, which provide insights into data segmentation and grouping.
- **Classification Analysis:** We use K-Nearest Neighbor (KNN) and Support Vector Machines (SVM) to do classification on the clustered data. The classification results are presented to demonstrate the importance of various strategies in classifying data points.

4.8 Results Interpretation

In this part, we evaluate the results of our research work in order to shed light on their significance in the context of our research objectives. We will discuss the importance of our findings for the challenges of managing big datasets, improving data processing efficiency, and obtaining insights from real-time streaming data.

- **Data Collection and Preprocessing:** Our research started with the collection and preprocessing of location data, which included the assignment of numerical labels. These methods confirmed that our dataset was consistent and fit for further examination.
- **Frequency Analysis:** We evaluated the frequency of each numerical label using the MapReduce framework, which indicates the regional distribution of cryptocurrency trends. This study provides information on the amount of cryptocurrency related activities in different regions.
- **Clustering Analysis:** K-Means and GMM clustering algorithms were used to reveal underlying patterns in the data. Visualizations of these clusters give useful information on the geographical distributions of cryptocurrency trends.
- **Classification Analysis:** Additional research includes the use of classification algorithms, notably KNN and SVM, to previously created clusters. The visual results highlight each model's classification performance.
- **Performance Metrics:** We calculated precision, accuracy, recall, and F1 score to evaluate the performance of our classification models. These measurements provide a thorough assessment of each model's performance and practical utility.

4.9 Validation

The validation process is essential for confirming the accuracy and robustness of our study findings. In this part, we describe the steps we followed to confirm our results and give openness about the accuracy of our research.

4.9.1 Data Validation

We visually evaluated our data after the MapReduce phase to confirm its integrity. Although we did not utilize formal validation procedures, this visual inspection allowed us to evaluate if the data appeared to be in good shape in terms of size and structure. While this provided us with a basic understanding of data quality, we acknowledge that more robust validation methods may be utilized in future study to ensure data integrity.

4.9.2 Validation of Analysis Techniques

Each analysis approach employed in this study was validated to ensure that it was appropriate for meeting our research aims. This validation includes the following:

- **Cross-Validation:** We employed cross-validation to assess the reliability of our classification models. This step, which was executed during classification, allowed us to evaluate how well our models performed on different areas of the data. Cross-validation guaranteed that our conclusions were not excessively impacted by a single dataset, which increased the validity of our results.
- **Model Comparison:** Following classification and cross-validation, we compared the performance of our clustering and classification models. This evaluation provided useful information on which approaches were more efficient in achieving

our study targets. It helped us to make educated choices about the best techniques for our specific dataset and study aims.

- **Performance Metrics:** As we discussed on Unit 4.8 after classification we calculated precision, accuracy, recall, and F1 score to evaluate the performance of our classification models.

4.9.3 Visualization

Visualization was an essential part of our validation strategy. It helped us in better understanding of the data and identifying potential problems. We utilized scatter plots to visualize the results of clustering and classification procedures. An elbow point diagram was also utilized to determine the optimal number of clusters for the K-Means clustering technique.

5. Implementation

Considering the examination of theoretical frameworks and substantial research that has served as the foundation of our research thus far, we are now at the point where we proceed on to the practical section of our work. We seek to combine the theoretical foundations with real-world application in a seamless way. Here, we will begin the process of transforming these abstract concepts and gathered information into an operational and beneficial system. Within this framework, we want to create a solution capable of addressing the various challenges that have been thoroughly examined and discussed in previous sections.

This chapter focuses on "Implementation". That involves taking what we have learnt and turning it into a working system. We will utilize the tools and approaches we mentioned before to build a system that can manage large amounts of Twitter data. Our implementation route is separated into stages, each with a distinct task. First, in Section 4.1, we will define the systematic design of our system's architecture, similar to constructing a project's foundation. Section 4.2, "Tools and Technologies Used", will discuss the software and technologies we used to make our system work properly. To make sure that the job is done correctly, we use reliable and up-to-date tools.

Due to the fact that data is such a vital aspect of our work, in Section 4.3, "Data Collection and Preprocessing", we'll discuss how we obtained and processed the Twitter data for analysis. Section 4.4 is where the biggest part of our work occurs. In "MapReduce Implementation for Twitter Data," we will demonstrate how we process and modify Twitter data using the MapReduce framework. Next, in Section 4.5, we will discuss "Clustering and Classification Implementation". This is where we utilize different approaches to identify patterns in the data and categorize it. Finally, in Section 4.6, we will tie it all together. We will demonstrate how MapReduce, clustering, and classification work together to accomplish our solution.

It is critical to remember that our primary goals when we move into the pragmatic component of our study are to gain knowledge, promote creativity, and make our research results usable and helpful in real-life scenarios. In the following chapters, we will describe the impact of our efforts, casting light on the achievements that we achieved and demonstrating how these accomplishments connect with our study's main objectives.

5.1 System Architecture and Design

5.1.1 Application Ambitions: Impact and Purpose

This application is designed with the primary goal of navigating through extensive datasets sourced from Twitter, allowing users to get insights into emerging trends in current technologies. This program offers to a wide user base worldwide, although it focuses primarily on users in the United States. Its main purpose is to analyze live Twitter data connected to trending investment technologies like Crypto, NFTs (those one-of-a-kind digital assets), and the Metaverse.

It is important to note that the application is versatile and not restricted to this theme topic. Users have the ability to choose from a variety of requests. Because of its adaptability, the program may give preliminary evaluations of how newly passed domestic laws or international developments may effect trends within specific areas or even on a global scale. As a result, the application provides users with the resources they need to make educated decisions and forecast the prospective trajectory of objects of interest.

In conclusion, this program is an effective tool for exploiting Twitter's real-time data stream to identify patterns, forecast market trends, and analyze the potential development of subjects under analysis. Its range of applications and thorough analytical skills make it a vital tool for those looking to keep a competitive advantage in the ever-changing technological world.

In short, the application can be summarized as follows:

- **Global Reach:** It serves users all around the world, with a particular emphasis on the United States.
- **Real-time Twitter Analysis:** Its major job is to analyze Twitter data in real time.
- **Diverse Monitoring:** Used to keep track of advances in a variety of fields of study.
- **Informed Decision-Making:** Users can stay up-to-date with developments, especially within the United States, while considering the global landscape.

5.1.2 Applications Architecture

As we analyze the architecture of this application, we uncover a well-organized framework. Each component of this architecture has a distinct purpose, similar to how individual pieces contribute to the overall picture. These components are dedicated to keeping track of many elements within the evolving technological landscape, which includes a diverse range of emerging concepts and trends. This architectural structure ensures that even those without prior knowledge can navigate the application effortlessly.

In order to understand how these components interact, take a look at the figure below. It demonstrates how everything fits together to provide a clear picture of shifting trends in investing technology.

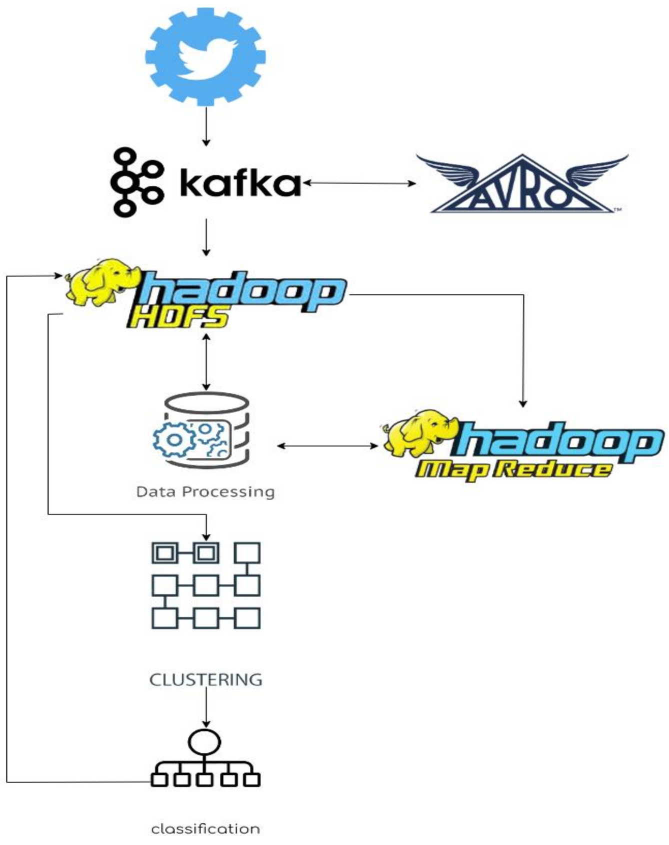


Figure 9: Application's Infrastructure Diagram.

The process of gathering data from Twitter involves bringing in Kafka, a distributed streaming infrastructure famous for its effectiveness in managing large volume real-time data streams. However, Avro plays an important role in streamlining management of data and avoiding the need for direct application manipulation of these data files. Avro acts as a strategic counterpoint to Kafka, providing incoming data streams with a structured format as they pass through the pipeline. This structured data not only prepares itself for further processing, but it also ensures its spot in HDFS. Avro, which plays an important part in this process, guarantees that the serialized data will be deserialized, making it easily available for in depth analysis and insights. As a result, we now have a large collection of Twitter derived data stored in HDFS, ready for full analysis and insights.

After successfully storing the data in a readable formal within HDFS, our study moves on to the first phase: the start of MapReduce operations and next storage the results back into HDFS. However, in order to acquire a good understanding of the application's architecture, it is necessary to dig into the level of detail of certain components with increasing complexity. These components include Kafka, which is in charge of bringing in data, the MapReduce framework, and the complex area of data analysis. It is critical to understand these components in order to fully comprehend how our program works.

5.2 Tools and Technologies Used

In our application's core framework, we have selected vital packages and frameworks to perform certain tasks. Our implementation languages are Java and Python. This decision is based on the fact that Hadoop is mostly based on Java, however data analysis, clustering, and classification benefit from Python's solid architecture. More precisely:

- **Kafka:** Controls real-time data streams to provide maximum throughput and fault tolerance.
- **Hadoop HDFS and YARN:** Offers a reliable and scalable structure to support distributed cluster data storage (HDFS) and resource management (YARN).
- **MapReduce:** Handles large-scale data processing effectively, converting raw data into meaningful insights.
- **Avro:** Optimizes data serialization and storage.
- **Scikit-Learn:** Provides various data analysis tools for clustering and classification.

These programming languages and technologies serve as the backbone of our application, allowing us to analyze and process data in depth.

5.3 Data Collection and Preprocessing

As we progress, our focus shifts to the critical step of data collection and preprocessing. In this part, we will discuss the process of acquiring and refining data for our study. We intend to provide useful insights on the data sources we used, their inherent features, and the essential preprocessing measures we took to ensure the quality and relevancy of the data used in our research.

5.3.1 Streamlining Data Collection with Kafka

Our Kafka solution is an elegant and deliberate method designed to effectively capture data from the Twitter API. Our architecture is built on a Kafka producer that has been developed to interface with the Twitter API and gather real-time data. We chose a simplified solution inside this environment, leveraging a single Kafka broker. This design choice not only simplifies the architecture, but also consolidates the data flow into a single Kafka topic and partition, assuring administrative convenience and excellent data organization.

While the complexities of ZooKeeper's work are beyond the scope of this research, its presence must be acknowledged. ZooKeeper works in the background, handling critical operations such as broker coordination, leader election, and general cluster management to ensure the smooth operation of our Kafka context. It's also worth noting that the data collected from Twitter is deserialized using Avro. This serialization technique optimizes data for storage while simultaneously laying the groundwork for effective downstream processing. This strategic application of Avro improves the overall efficiency of our data pipeline, allowing us to extract relevant insights from the Twitter data stream.

In summary, our Kafka implementation is a system in which each component plays a critical role in effectively receiving and processing Twitter data, therefore contributing to the development of analytics and insights. We have vital elements in our Kafka producer that deserve special attention. Some of those include connecting to additional data sources like the Twitter API and strategically selecting data characteristics such as keywords. These factors all contribute to the producer's ability to manage data input.

Twitter Data Streaming

The following Java code snippet is a real-time data retrieval and processing application that primarily monitors digital currencies movements on Twitter. It uses OAuth1 authentication to build a secure connection to Twitter's data stream, which is secured by API keys and access tokens. It also monitors certain phrases like "ADA", "BTC", "ETH", and "CRYPTO" to filter and record tweets on cryptocurrencies. As previously stated, these keywords can be substituted in order to evaluate another field of study.

```
String[] twitterKeys = new String[] {
    "API Key",
    "API Key Secret",
    "Access Token",
    "Access Token Secret"
 };

 String[] keywords = {
    "ADA",
    "BTC",
    "ETH",
    "CRYPTO"
 };

BlockingQueue<String> queue = new LinkedBlockingQueue<String>(10000);
StatusesFilterEndpoint endpoint = new StatusesFilterEndpoint();
endpoint.trackTerms(Lists.newArrayList(keywords[0], keywords[1], keywords[2]));
Authentication auth = new OAuth1(twitterKeys[0], twitterKeys[1], twitterKeys[2],
twitterKeys[3]);

Client client = new ClientBuilder()
    .hosts(Constants.STREAM_HOST)
    .endpoint(endpoint)
    .authentication(auth)
    .processor(new StringDelimitedProcessor(queue))
    .build();

client.connect();
```

5.3.2 Data Preprocessing

Transforming data is another crucial feature of our Kafka producer. The producer handles data as it enters from Twitter API, including gathering location data coupled with each tweet. Such modifications are critical for preparing data for subsequent examination.

```
String tweetText = ((String) user.get("location")).toUpperCase();
String data = "51";
for (int j = 0; j < states.length; j++) {
    if (tweetText.contains(states[i].toUpperCase()) ||
            tweetText.contains(postal[i].toUpperCase())) {
        data = tweetText.replace(tweetText, String.valueOf(i + 1));
        break;
    }
}
```

Twitter data is prepared for analysis during the code's preprocessing step. Initially, the location information in the tweet is converted to uppercase format. As specified in the appendix (Appendix A), the default number code of "51" is used to signify a location outside of the United States. The code then searches through a list of US states and postal abbreviations to see whether the tweet's location matches any of them. If a match is found, the numerical code is updated to correspond to the state or abbreviation. This numerical code is inserted in an Avro record and serialized for storage or future processing within a Kafka topic. Additionally, the code saves this code to a local text file for future use. During this preprocessing phase, we include geographic information in the tweet data. Taking into consideration the numerical mapping of the United States, as stated in Appendix A, data analysis is further simplified.

5.4 MapReduce Implementation for Twitter Data

Switching from Kafka to MapReduce is similar to switching from processing individual emails to summarizing data from various sources. Switching from Kafka to MapReduce is similar to switching from processing individual emails to summarizing data from various sources. You process one message at a time in the Kafka world, much like reading and replying to emails as they arrive. In the MapReduce world, you deal with large amounts of data collectively, like analyzing and summarizing data from various emails all at once. Three distinct classes play critical roles in organizing and performing data processing jobs under Hadoop's MapReduce architecture:

i. **Mapper:** Its purpose is to receive input data, process it, and create <key, value> pairs. In this case, it tokenizes the input text data, treating each line as a token, and produces each line as a key with a value of one. This indicates that during the MapReduce mapping phase, the Mapper creates a <key, value> pair for each line in the incoming data. The key in this pair corresponds to the line of text, and the value is always set to one. This is the first stage of the MapReduce process, in which data is organized and prepared for parallel processing.
ii. **Reducer:** This component receives the key-value pairs crafted by the Mapper. The main objective is to collect and analyze these pairings in order to create the decisive result. In this case, it utilizes the emitted <key, value> pairs, which represent lines of text alongside the value one, and then counts the occurrences of each line. Following that, it creates new <key, value> pairs, where the key represents a unique line of text and the value indicates the count of how frequently that specific line appeared in the input data. It basically organizes the process of summarizing and calculating data that yields the desired results.
iii. **Runner:** The Runner or Driver class orchestrates the entire MapReduce job. It sets job parameters, designates input and output paths, specifies Mapper and Reducer classes, and configures job-specific settings. The Driver class serves as the entry point, ensuring that job configurations are correctly set and managing job execution on the Hadoop cluster.

The following code demonstrates the mapping process, which involves tokenizing lines of text into words and emitting key-value pairs with word counts.

```
public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable> output,
            Reporter reporter) throws IOException {
    String line = value.toString();
    StringTokenizer token = new StringTokenizer(line, "\n");

    while (token.hasMoreTokens()) {
        word.set(token.nextToken());
        output.collect(word, one);
    }
}
```

The following snippet demonstrates the Reducer's critical function in Hadoop MapReduce. The following snippet demonstrates the Reducer's critical function in Hadoop MapReduce. It effectively collects and summarizes items with the same key, which is critical for activities like counting and analyzing scattered data, allowing raw outputs to be transformed into important insights.

```
public void reduce(Text key, Iterator<IntWritable> values, OutputCollector<Text,
IntWritable> output, Reporter reporter) throws IOException {
    int sum = 0;
    while (values.hasNext())
        sum += values.next().get();
    output.collect(key, new IntWritable(sum));
}
```

5.5 Clustering and Classification Implementation

Following an effective data transformation with MapReduce, a well-organized dataset ideal for academic data research becomes available. It offers features for both unsupervised clustering and supervised classification, making it an adaptable data analysis toolset. The Gaussian Mixture Model (GMM) clustering function uses the scikit-learn package to sort data into groups and displays the results in the form of a scatter plot. Similarly, the K-Means clustering function accomplishes the same goal but using the K-Means technique for clustering. The code includes a "Elbow Method" function that plots the Sum of Squared Errors (K-SSE) for different cluster counts to discover the ideal number of clusters for K-Means.

In terms of classification, our toolkit serves two purposes. The K-Nearest Neighbors (KNN) function performs classification tasks through the use of closest neighbor techniques. It assesses performance by computing important measures like accuracy, precision, recall, and F1-score, all while offering visual insights via scatter plots. Furthermore, the Support Vector Machine (SVM) classification algorithm uses GridSearchCV to fine-tune hyperparameters, enhancing classification performance. Additionally, it evaluates the effectiveness of classification using several metrics like KNN. It also helps understanding the results by offering visual representations in the form of scatter plots.

The primary goal of this research project is to perform complex clustering and classification tasks utilizing a dataset obtained from the MapReduce output. The first step in this complex procedure is to import the dataset and dissect it into key elements, which include but are not limited to state numbers and their related frequencies. These core aspects serve as the primary building blocks for our research's succeeding phases.

We employ two distinct clustering approaches within the data analysis workflow. The first, that is the classic K-Means algorithm, which is well-known for its ability to group data points based on similarity. In addition to this, we also use the Gaussian Mixture Model (GMM), a probabilistic clustering technique noted for its versatility and ability to detect complicated data structures. These clustering algorithms form the foundation of our research, allowing us to find relevant patterns and insights within the dataset and thereby advance the study's broader aims.

K-Means divides the dataset into clusters based on similarity, while the Elbow Method determines the best number of clusters. Visualizations are used to help people recognize the structure of data. Similarly, the GMM method is used to cluster the data, and visualizations are created to highlight data distribution and grouping patterns. Following that, the workflow enters the classification phase, where the K-Nearest Neighbors (KNN) and Support Vector Machine (SVM) algorithms are used. Both cover data segmentation, classifier fitting, and prediction production, with performance evaluated through graphs.

Our system is organized around many crucial steps, with a significant emphasis on data loading, clustering, and classification. These crucial factors operate in conjunction to reveal hidden structures and patterns in the dataset. This method not only improves our understanding of the data, but it also acts as an effective tool for carrying out machine learning tasks successfully. Furthermore, it is critical in enhancing data-driven decision-making processes, providing a strong platform for making informed and strategic decisions with precision.

5.6 Dataflow Management

This particular component is vital because it orchestrates flawless communication among a plethora of related pieces, including critical components such as Kafka, MapReduce, and the data analysis phase. Its primary goal is to provide seamless transfer of data and effective data storage within this ecosystem. Aside from its vital function in "data transportation", this component also acts as guard for data security. Its purpose is to keep data safe within a secure enclave inside the ecosystem.

The operation effectively establishes a connection between the Kafka stream and HDFS, ensuring a continuous flow of data from Twitter. After MapReduce processing, data is stored once more in HDFS for additional analysis. This approach involves capturing data in an organized and timestamped manner, assuring its accessibility and traceability for future usage. A vital aspect of this system is its ability to create dependable connections between diverse components. This system's capacity to establish stable connections between many components is critical. This guarantees that data created at each phase is efficiently stored, allowing for further analysis and contributing to a smooth and robust process.

From a source code perspective, it appears that it performs a decent job of handling data utilizing Hadoop Distributed File System (HDFS). It adheres to best practices for working with HDFS files, such as creating, reading, and writing data effectively utilizing built-in Hadoop tools. Furthermore, it handles errors well by managing exceptions and correctly closing streams. This configuration has some interesting code. It generates file names with timestamps for data in HDFS, moves data between local and HDFS storage effectively, and provides a secure connection to the HDFS cluster. These sections demonstrate a solid understanding of HDFS fundamentals and make the code more reliable and adaptable.

In conclusion, this component is essential for ensuring data integrity, connectivity, and accessibility within a complicated data processing and analysis workflow. It is a critical connection that adds to the systems overall reliability and performance.

The following code snippet is vital and its worth observing because it initializes the configuration and file system connection to HDFS, allowing the application to communicate with all of its components, including Kafka, MapReduce, and data analysis modules, in real time. Furthermore, it guarantees that data is safely saved in HDFS, creating a dependable and centralized repository for vital data.

```
private Configuration conf;
private FileSystem fs;

public svHDFS() throws IOException {
    conf = new Configuration();
    conf.set("fs.defaultFS", "hdfs://localhost:9000");
    conf.addResource(new Path("/HADOOP_HOME/conf/core-site.xml"));
    conf.addResource(new Path("/HADOOP_HOME/conf/hdfs-site.xml"));
    fs = FileSystem.get(conf);
}
```

6. Experimental Evaluation

In the following sections, we will go through our experimental setup, as well as the results of our data processing. This study breaks down each critical component of our system, making it easier to understand. We begin with Kafka, where we initially collect data. Then we dig into MapReduce, which is in charge of the majority of our data processing. Following that, we will look at HDFS, our data storage choice. After that, we will look at how clustering and classification techniques might help us extract important insights from our data. We want to offer an in-depth overview of our data processing pipeline and its functionality in these parts.

6.1 Experimental Setup and Data Description

Moving forward, we will now look at the dataset's layout. Understanding this structure is crucial for the analysis that follows. This phase serves as the revealing of the dataset's complexities, successfully establishing the groundwork for additional research.

The original dataset, collected by Kafka, has a distinct structure. Within this dataset, numerical values span the range from 1 to 50 and each of these numbers corresponds to a state within the United States. When the number is "51," it indicates a location outside of the United States. Notably, this dataset employs a newline character as a delimiter, which was purposefully intended to improve visibility and distinguishability. This deviation from usual delimiters such as semicolons or commas results in a dataset saved as a text file rather than a standard CSV file. For example, here is a sample of how the dataset looks:

9
36
31
34
50

Table 1: Part of the Dataset.

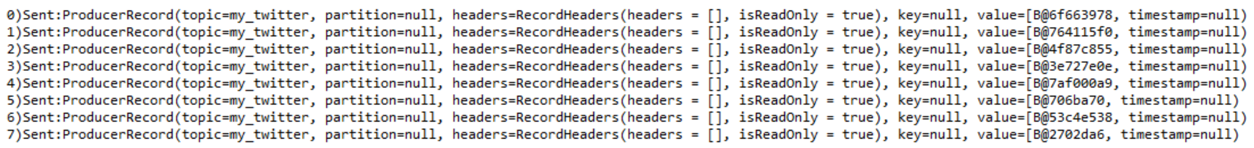
As previously stated, it is important to understand that the data originates in the form of text strings indicating geographical places before they are added in our collection. However, in order to properly prepare the data for future analysis, we must go through a preprocessing procedure. In the scope of our study, we are focused on a dataset that is of crucial importance, with 51 different entries and a grand total of 2500 records, establishing itself as an important and necessary resource for our research.

6.2 Results and Analysis

We now move our focus to the results and analysis in section 5.2, where we dive into the findings and insights generated from our study.

6.2.1 Kafka Analysis

In this chapter, our focus is on understanding the complexities of the data collection process supplied by Kafka.. We begin by launching a Kafka producer, which is significant since it marks the beginning of our data collecting journey. We begin collecting tweets at this time in order to do our analysis later on. It is important to note that we are simultaneously creating a dedicated file while collecting data. This file is required before we can migrate the data to HDFS, which we will go through in detail in the next sections. A figure below is attached showcasing how tweets are collected by the Kafka producer.



```
0)Sent:ProducerRecord(topic=my_twitter, partition=null, headers=RecordHeaders(headers = [], isReadOnly = true), key=null, value=[B@6f663978, timestamp=null)
1)Sent:ProducerRecord(topic=my_twitter, partition=null, headers=RecordHeaders(headers = [], isReadOnly = true), key=null, value=[B@764115f0, timestamp=null)
2)Sent:ProducerRecord(topic=my_twitter, partition=null, headers=RecordHeaders(headers = [], isReadOnly = true), key=null, value=[B@4f87c855, timestamp=null)
3)Sent:ProducerRecord(topic=my_twitter, partition=null, headers=RecordHeaders(headers = [], isReadOnly = true), key=null, value=[B@3e727e0e, timestamp=null)
4)Sent:ProducerRecord(topic=my_twitter, partition=null, headers=RecordHeaders(headers = [], isReadOnly = true), key=null, value=[B@7af000a9, timestamp=null)
5)Sent:ProducerRecord(topic=my_twitter, partition=null, headers=RecordHeaders(headers = [], isReadOnly = true), key=null, value=[B@706ba70, timestamp=null)
6)Sent:ProducerRecord(topic=my_twitter, partition=null, headers=RecordHeaders(headers = [], isReadOnly = true), key=null, value=[B@53c4e538, timestamp=null)
7)Sent:ProducerRecord(topic=my_twitter, partition=null, headers=RecordHeaders(headers = [], isReadOnly = true), key=null, value=[B@2702da6, timestamp=null)
```

Figure 10: Data Collection.

6.2.2 MapReduce Analysis

Following that is MapReduce Analysis, which is an important aspect of our research. It is the framework that assists us in processing and making sense of the material we have gathered. In this part, we will look at how MapReduce works and why it's vital for our research.

Records are initially expressed as <key, value> pairs in the MapReduce job, with each record related with its own state number. The framework organizes records based on these state numbers throughout the shuffling phase. Following that, during the reducing phase, the system computes record frequencies for each state. This procedure provides a simplified dataset with record frequencies classified by state. This streamlined representation simplifies our research and allows us to derive valuable insights from the data. The process previously stated is illustrated in the following example. Assume that input data is as follows:

12
45
30
12
30
12
40

Table 2: Input Data for MapReduce Paradigm.

Now, let us go through the MapReduce process step by step. The Mapper scans each line, to-kenizes it, and emits key-value pairs with the value "1" for each integer detected. As an example:

12	1
45	1
30	1
12	1
30	1

Table 3: Mapper Output.

In Shuffle and Sort phase, the intermediate <key, value> pairs are shuffled and sorted by key, resulting in a grouping of all values (1s) for each unique number (key).

12	1, 1, 1
30	1, 1
40	1
45	1

Table 4: Shuffle and Sort phase.

The Reducer gets grouped values for each number and sums them to get the total count for each number.

12	3
30	2
40	1
45	1

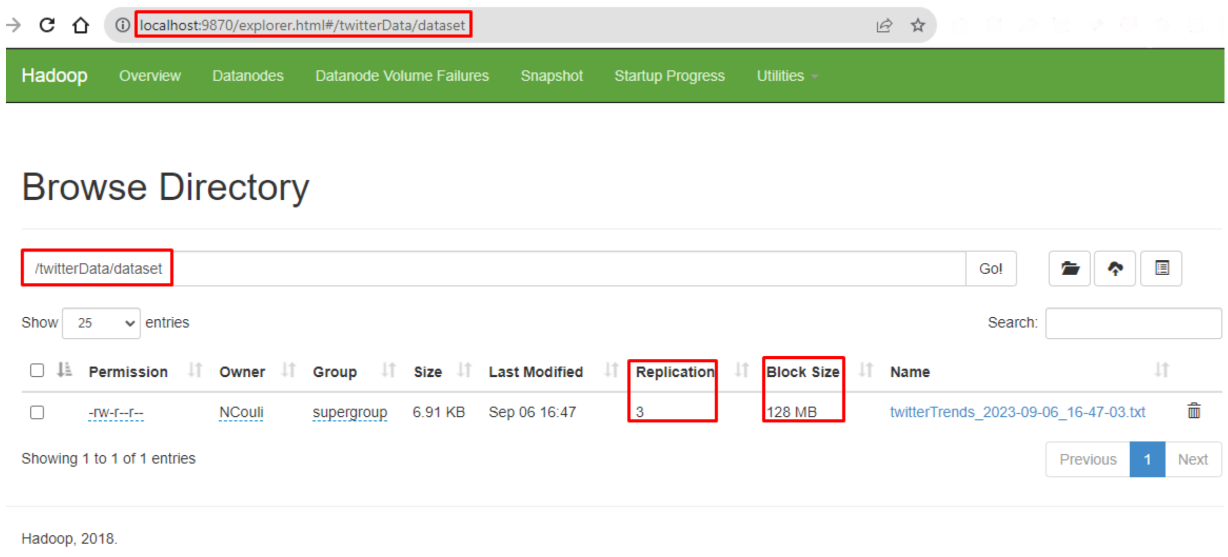
Table 5: MapReduce output.

After executing the MapReduce task, here are the final results. In the input data, each number is associated with its total count. This is a simplified illustration of how MapReduce works to count the number of occurrences in input data. MapReduce can handle far larger and more complicated datasets in practice.

6.2.3 HDFS Analysis

In this section, we will look at the Hadoop Distributed File System (HDFS) and its role in data management and storage in the context of Big Data. We will look at two critical stages of the data lifecycle: data collection and data processing. The seamless transfer of data into the HDFS from Kafka is referred to, as data collection in the context of our investigation. When data is stored into HDFS, it is immediately available for future actions and processes. HDFS is a critical component for the effective storage and administration of big datasets, ensuring that they are well organized and ready for further analysis and operations.

Moving on from data collection, we enter the data processing phase, where MapReduce, a critical component of Hadoop, takes the spotlight. MapReduce allow us to process data across multiple machines, letting us to extract important insights from HDFS data. This movement from data collection to data processing emphasizes HDFS's critical role in storing and effectively managing data for analysis in the Big Data environment. As we move forward, the figures below provide a visual representation of how HDFS successfully stores files.



Browse Directory

/twitterData/dataset Go!

Show 25 entries Search:

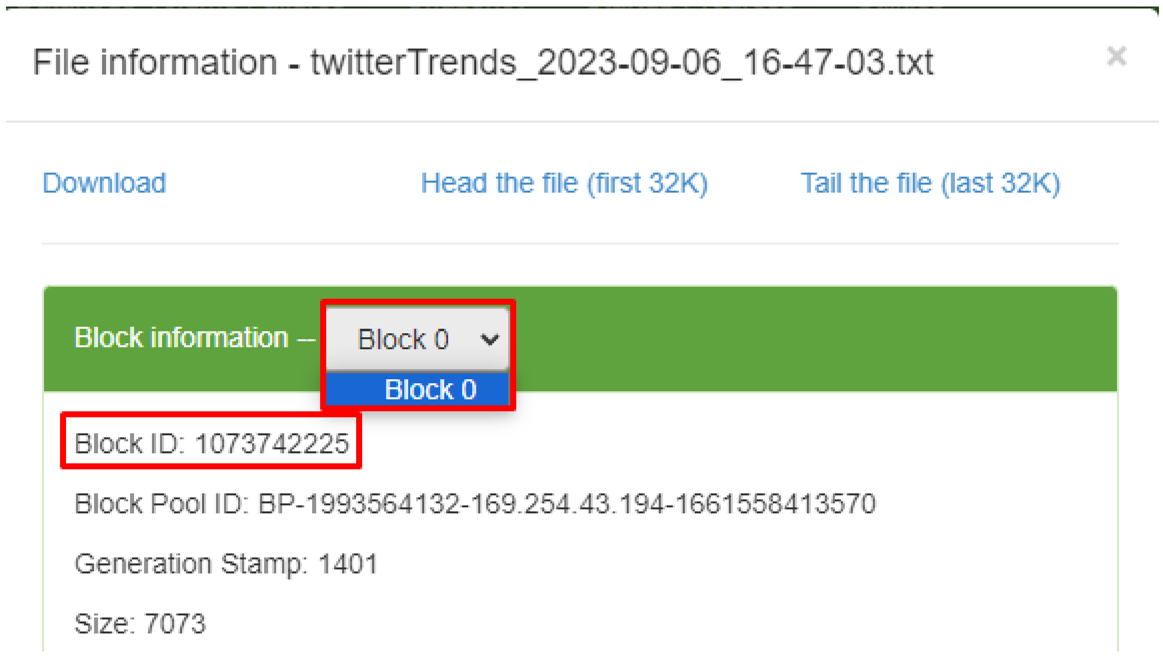
Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	NCouli	supergroup	6.91 KB	Sep 06 16:47	3	128 MB	twitterTrends_2023-09-06_16-47-03.txt

Showing 1 to 1 of 1 entries

Previous 1 Next

Hadoop, 2018.

Figure 11: HDFS File Browser.


File information - twitterTrends_2023-09-06_16-47-03.txt

Block information -- Block 0

Block ID: 1073742225

Block Pool ID: BP-1993564132-169.254.43.194-1661558413570

Generation Stamp: 1401

Size: 7073

Figure 12: HDFS File Information.

A more detailed examination of the figures above reveals some of HDFS’s inherent and significant characteristics. Notably, one of the most striking aspects is our data's alignment with the predetermined block size of 128MB. This alignment emphasizes our data's flawless integration inside the HDFS architecture, as well as the system's ability to handle it efficiently, which is mostly related to the data's size. Furthermore, it is essential to acknowledge HDFS's comprehensive security mech-anisms, which include the preservation of three data replicas. This replication procedure offers a layer of resilience and security to our stored data, protecting it from any data loss or system failures. In addition, the numbering of blocks inside HDFS helps to improve the general structure and management of our data. This numbering system demonstrates HDFS's efficiency in its attempt to preserve and maintain our data, highlighting its capabilities as a reliable and resilient file storage solution.

6.2.4 Clustering Analysis

After successfully processing and storing all of our data in the Hadoop Distributed File System (HDFS) through the efficient utilization of the MapReduce framework, we shift our focus to clustering analysis. In this section, we will look at hidden patterns, groups, and insights in our dataset. Using K-Means and the Gaussian Mixture Model (GMM), we hope to uncover connections between data points that conventional research may have missed. This step of our research promises to be very instructive since it has the potential to provide previously unnoticed patterns in our dataset. These insights will help us make more educated decisions and have better understanding of underlying concepts.

K-Means

In the initial stages of the K-Means clustering research, we proceeded on an important step: identifying the ideal number of clusters, commonly referred to as the Elbow point. Figure 13 illustrates the Elbow point, which marks the optimal number of clusters for our dataset. Our study pinpointed a single point on the graph that represented the ideal cluster count for our dataset. This discovery serves as a base for a more extensive and informative clustering study, allowing us to examine complicated data patterns and connections.

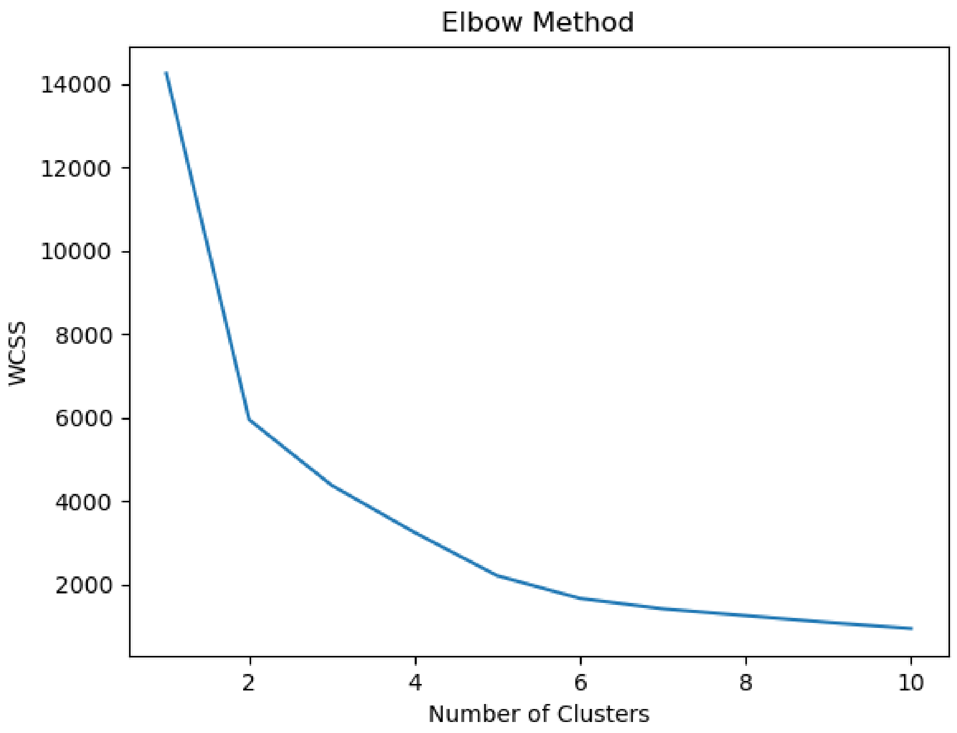


Figure 13: Elbow Point.

Close inspection of the graph showed that the reduction in the sum of squared distances between data points and their assigned cluster centroids slowed dramatically at a certain elbow point. This critical juncture shows that the best number of clusters, indicated as 'k', is 5. Our goal is to find a balance between having accurate clusters and providing useful insights.

After identifying the elbow point in the K-Means clustering, we perform K-Means clustering using the optimal number of clusters found by this analysis. This phase allows us to cluster similar data points for future study and evaluation.

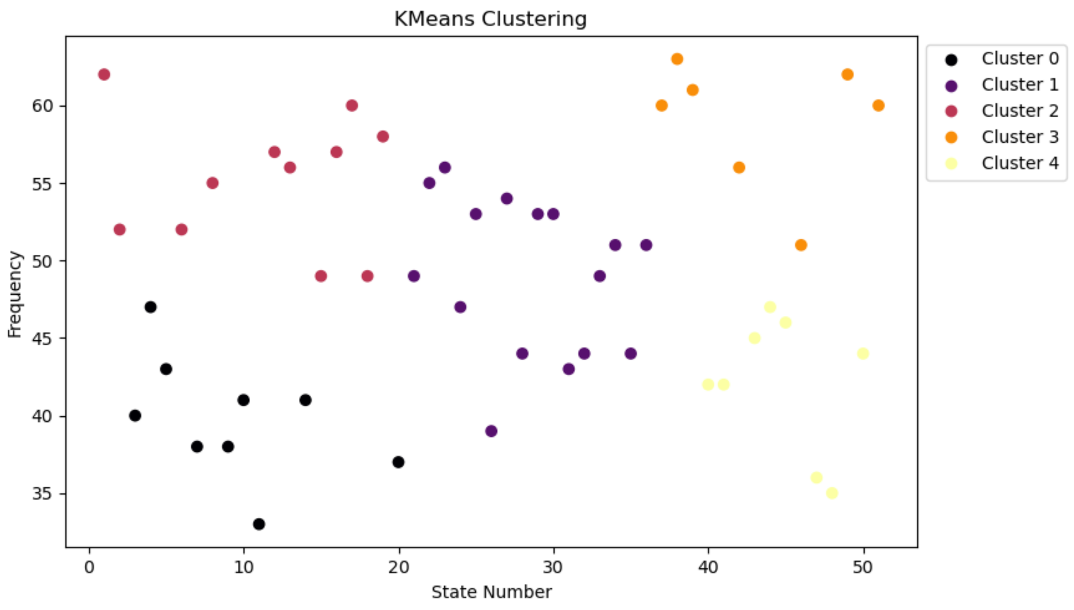


Figure 14: K-Means Clustering.

The K-Means clustering results for our dataset are shown in Figure 14. Each scatter plot point represents a data point, and the colors indicate the clusters to which these data points belong. The purpose of K-Means clustering is to group together similar data points, and this visual depiction shows how the algorithm categorized our set of data into various clusters. The graphic representation gives us a clear picture of the cluster assignments, allowing us to see patterns and correlations between data points. It provides a first look at the structure of our dataset and assists in recognizing the groups with-in our data.

Gaussian Mixture Model

Gaussian Mixture Model is a powerful clustering technique we have employed to gain deeper in-sights from our dataset. GMM, in contrast to standard clustering approaches such as K-Means, uses a probabilistic approach. It permits data points to be members of numerous clusters at the same time, each with a different level of membership probability. This adaptability is especially useful when working with big datasets with overlapping patterns. Similarly to K-Means, the Gaussian Mixture Model analysis seeks to find important patterns in our data.

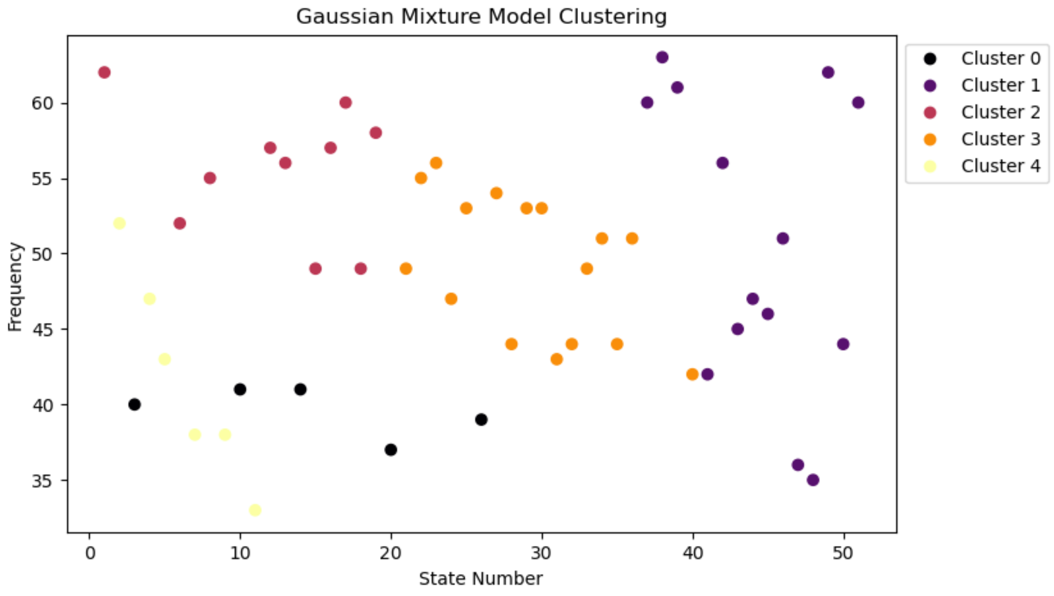


Figure 15: GMM Clustering.

Clusters in K-Means and GMM do not match since these two clustering methods work on different principles and assumptions. Moreover, K-Means allocates each data point to a single, well-defined cluster, with the goal of minimizing the distance between each data point and the centroid of its corresponding cluster. This method frequently yields compact, spherical clusters with distinct borders. GMM, on the other hand, represents data points as probabilistic mixes of Gaussian distributions. It permits data points to be members of many clusters at the same time, with varied degrees of membership likelihood. This inherent flexibility leads to the formation of more diffuse and overlapping clusters.

In essence, K-Means aims to produce non-overlapping, separate clusters, whereas GMM allows for mixed memberships. Because of these basic differences in modeling methodologies, the cluster topologies in K-Means and GMM diverge, as seen in the corresponding charts.

6.2.5 Classification Analysis

In the classification aspect of our analysis, we go beyond clustering to uncover finer details within the data. We use two separate classification approaches, k-Nearest Neighbors (KNN) and Support Vector Machines (SVM), to build on the clusters formed by K-Means and GMM. This multifaceted approach allows us to dive deeper into the hidden patterns inside each cluster. We use this approach in order to increase classification performance by exploiting the unique properties of each cluster. Each cluster represents a distinct segment of the data, complete with its own characteristics. We want to extract finer characteristics that may be obscured in a wider classification study by applying KNN and SVM within these subgroups.

KNN, is a robust and intuitive algorithm, classifies data points based on the majority "opinion" of their neighbors within each cluster. This method takes use of the intrinsic similarities between data points inside the same cluster, allowing for precise and context-aware classification. In contrast, SVM uses a complex approach to create an ideal hyperplane that successfully separates various classes within each cluster. This allows for advanced separation of data points, resulting in classification limits that fit with the data's detailed patterns.

Figure 16 illustrates the KNN classification results for the K-Means cluster. Similarly, figure 17 showcases the SVM classification over the GMM cluster.

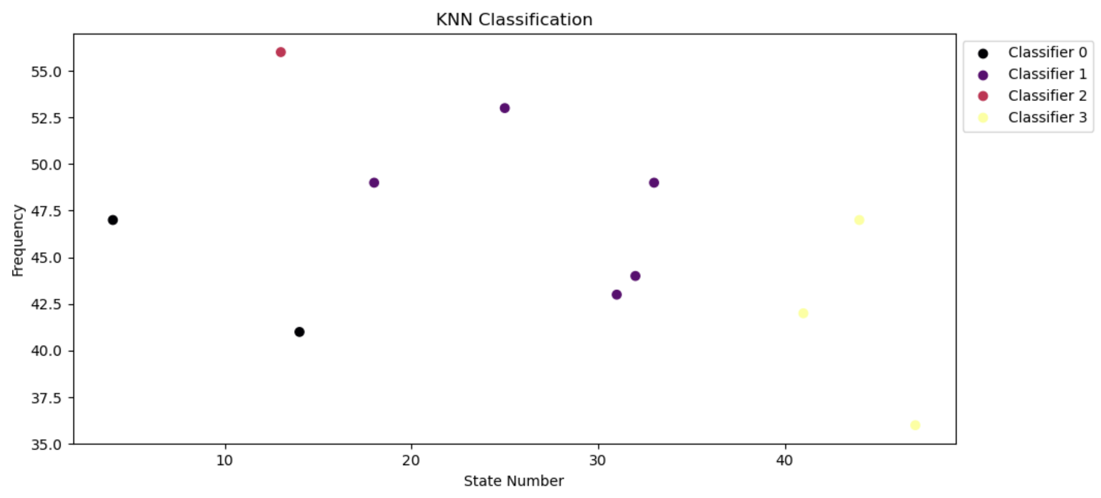


Figure 16: KNN Classification over K-Means Cluster.

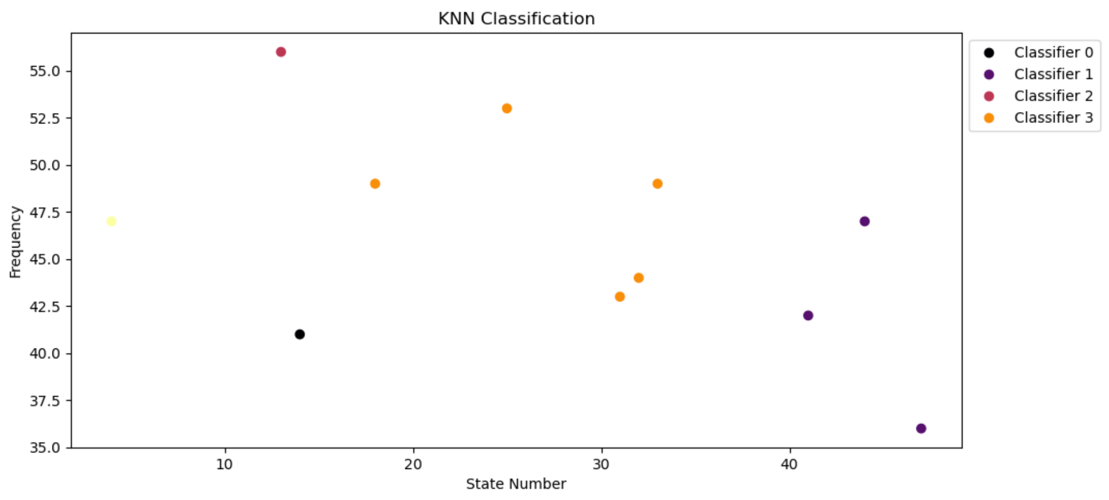


Figure 17: KNN Classification over GMM Cluster.

The main reason why figures are not identical is because of the clustering techniques that have occurred before applying KNN. As a result, when applied to these various clusterings, KNN modifies its classification boundaries accordingly resulting in differences in the output figures. This difference demonstrates the KNN algorithm's sensitivity to the features of the underlying clusters and emphasizes the need of using a suitable clustering approach in alongside classification.

Following that, the following figures 18, 19 zooms in on SVM classification withing K-Means and GMM clustering accordingly.

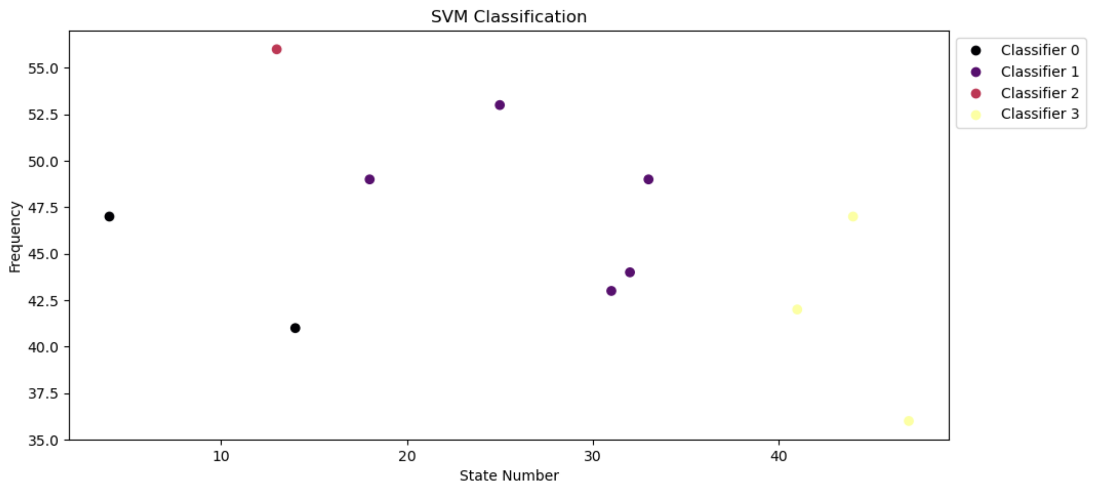


Figure 18: SVM Classification over K-Means Cluster.

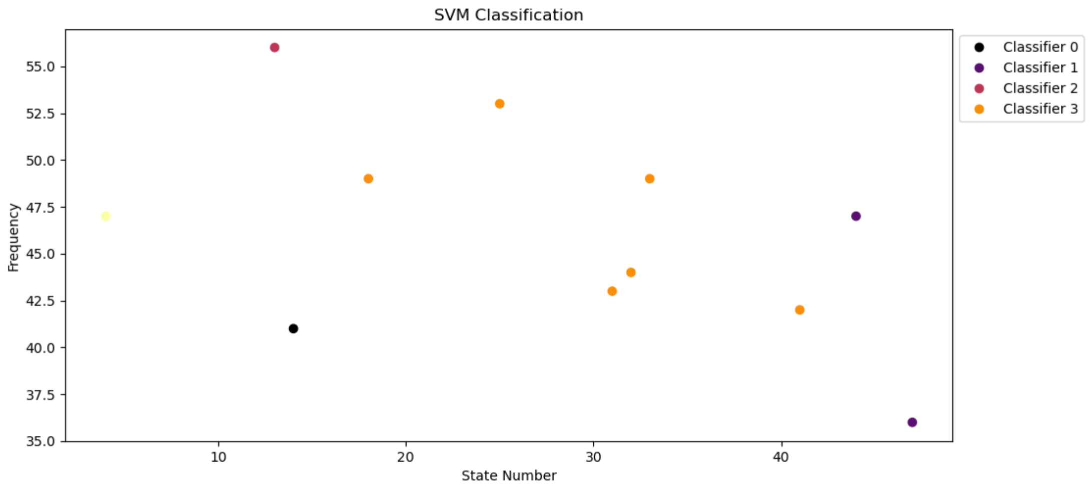


Figure 19: SVM Classification over GMM Cluster.

Just as with KNN classification, the variations observed in the figures for SVM classification can be attributed to SVM's high sensitivity to the specific characteristics and details within the input data. The difference in the amount of data points between the classification and clustering figures can be due to a variety of factors, including the distinct nature of the classification and clustering procedures.

- **Data Reduction:** The dimensionality of data during clustering might have lowered, which may result in fewer data points in the display.
- **Data Distribution:** The kind of data, as well as how it is distributed, can have an influence on the number of data points. Dot density might vary if data are extremely dense in certain locations and sparse in others.
- **Classification vs. Clustering:** Clustering brings together comparable data points to form clusters, which can lead to bigger groups with more data points. In classification, you are assigning data points to specific classes or labels, and some classes may contain fewer data points than others.

Now, let us look at performance metrics in greater detail.

	KNN		**SVM**	
Evaluation Metrics	K-Means	GMM	K-Means	GMM
Accuracy	0.90909	0.9091	0.90909	0.818
Precision	0.92727	0.9273	0.92727	0.879
Recall	0.90909	0.9091	0.90909	0.818
F1 Score	0.89899	0.899	0.89899	0.812

Table 6: Performance Metrics.

Based on the results of the four different models, which combine K-Means and GMM clustering with KNN and SVM classification algorithms, we can draw several conclusions:

- **Accuracy and Precision are constantly high:** All four models Accuracy and Precision metrics are consistently high, hovering around 0.90909. This indicates that these models are doing a good job of accurately identifying data pieces.
- **Recall and F1 Score Variation:** While Accuracy and Precision tends to be outstanding there is a slight shift in Recall and F1 Score. The model that combines GMM clustering and SVM classification has a slightly lower recall and F1 Score than the rest. This indicates that this model may struggle to recognize every case of the positive class properly, resulting in a slightly lower recall score. The F1 Score, which balances accuracy and recall, remains quite high at 0.81212.
- **Model Selection:** The four models are chosen based on individual criteria. If achieving a balance between precision and recall is a priority, the model that combines GMM clustering and SVM classification is a viable choice. On the other hand, if high precision and accuracy are important and a slightly lower recall is acceptable, any of the other three models may be suitable.

6.3 Discussion

This chapter attempts to broaden our level of comprehension, scope for thought and reflection, as well as implications. In this section of this article, we evaluate the relevance of our findings, closely examining how they relate to our research objectives, and explore the broader context in which our research contributes to knowledge. The "Discussion" unit serves as a link between the empirical and the conceptual, transforming information into insight and comprehension.

6.3.1 Interpretation of Results

Kafka as a Data Collector

Our data collection process commenced by initiating a Kafka producer, setting in motion the essential task of gathering data (in this case, tweets) from Twitter, while simultaneously creating a file used as a repository for these tweets. This file had to be firstly produced, before moving forward with the data migration to HDFS and set the stage for our subsequent analytical work.

Exploiting MapReduce for Data Analysis

We used the powerful MapReduce framework to help us navigate the vast amount of data that Kafka has allowed us to gather. MapReduce consists of three distinct phases, each one of them somehow contributed to the overall data processing pipeline.

- **Mapping Phase:** The Mapper thoroughly examined each line, tokenized it, and then generated <key, value> pairs.
- **Shuffling and Sorting Phase:** The pairs were meticulously shuffled and sorted, primarily according to their keys, following their assignment to each record in accordance with its unique state number. The framework then arranged all of the records according to their associated state numbers.
- **Reducing Phase:** This is where insights were finally synthesized. By computing record frequencies for each state, we constructed a condensed, simplified dataset which divided record frequencies into state categories. This simplification process improved the effectiveness of our study while allowing us to easily gain important insights from the data.

Harnessing HDFS for Data Storage and Accessibility

HDFS's role was proven to be crucial in the latter stages of our data analysis journey. It consists of two unique phases.

- **Data Collection Phase:** HDFS effortlessly took control as data got transferred from Kafka, facilitating a seamless transition, thus making data instantly accessible for our following actions and processes.
- **Data Processing Phase:** This is where MapReduce came to the forefront once again, allowing data processing across machines, supporting the extraction of significant insights from HDFS data.

K-Means VS Gaussian Mixture Model

A fundamental distinction arises in the cluster topologies produced by K-Means and GMM. While K-Means strives to produce distinct, non-overlapping clusters, GMM allows for mixed memberships, leading to various cluster arrangements.

On the one hand, GMM, as a potent clustering technique, offers a probabilistic strategy that takes into account data points being members belonging to numerous clusters simultaneously, each with a different likelihood of membership. As already indicated, dealing with huge datasets defined by overlapping patterns lends itself particularly well to this adaptation and flexibility. On the other hand, K-Means aims to reduce the distance between each data point and the centroid of its related cluster by assigning each data point to a single, clearly defined cluster.

Both clustering techniques used, provided visual depiction, however the K-Means method, successfully grouped similar data points into separate, distinct clusters, providing us with a visual representation of these groups created. This visual illustration assisted us in determining the optimum number of clusters, referred to as the Elbow point (which turned out to be 5, in this case). This discovery indicated the technique's significance in optimizing the clustering process, while also laying the groundwork for a larger-scale clustering investigation, enabling us to explore and study intricate data patterns and relationships.

K-Nearest Neighbors VS Support Vector Machines

In the “classification” part of our analysis, we looked further into the dataset beyond clustering. Utilizing KNN and SVM, we built upon the clusters formed by K-Means and GMM to extract more precise and detailed information within each cluster. Thanks to this comprehensive method, we were able to identify subtle traits that were hidden in a larger scale classification analysis. In order to define precisely the classification borders, SVM seem to be using a more complex and intricate method than KNN, which depends on neighbor-based classification within each cluster while considering the inherent similarities between data points.

The sensitivity of KNN and SVM to underlying cluster properties led to disparities in classification results between the two algorithms. KNN relies on neighbor-based consensus within clusters, while SVM establishes optimal hyperplanes to separate classes, leading to the apparent differences in the classification results.

Total results for clustering and classification methods

Combining the results of our four models, which integrate K-Means and GMM clustering with KNN and SVM classification algorithms, we draw several noteworthy conclusions:

- Accuracy and Precision consistently display high values across all models, demonstrating their effectiveness in precisely identifying data patterns.
- Recall and F1 Score vary slightly among the models, with the GMM clustering and SVM classification model showing a slightly lower recall and F1 Score. This implies that this model might not be able to identify all instances of the positive class.
- Model selection should be based on predetermined standards. The GMM clustering and SVM classification approach could be appropriate if striking a balance between precision and recall is important. However, any of the other three models can be taken into consideration if great precision and accuracy are essential and a somewhat lower recall is acceptable.

Final thoughts on results

Summarizing, it can be argued that our thorough investigation, experimentation and analysis have illuminated how the synergy between Kafka, MapReduce, and HDFS was the driving force behind our analysis, as well as the efficacy of various clustering and classification algorithms, providing insightful in-formation for decision-making and emphasizing the trade-offs between different performance indicators.

These results contribute to a deeper understanding of our dataset, thus providing suggestions for efficient data collection and insightful extraction, along with choosing the most suitable modeling approach based on particular goals and priorities.

7. Conclusion

This article has undertaken a comprehensive exploration of a wide range of Big Data management and analytics-related topics, providing insights into the practical, real-world implementation of technologies such as Hadoop, Kafka, and MapReduce. Through this study, we have presented a data processing pipeline that includes data collection, transformation, and analysis, demonstrating the potential for uncovering and extracting useful patterns and knowledge from huge datasets using clustering and classification algorithms.

This study's route has included a detailed literature analysis, clarifying the difficulties and adaptation tactics in handling massive amounts of data, diving into the complexity of HDFS, illustrating its design, scalability, and data integrity procedures. Furthermore, this paper has discussed the importance of streaming data and its practical applications, particularly in the context of Twitter data, while introducing the powerful streaming platform of Kafka. Emphasis was given on the serialization techniques for streaming data as well as on their significance in Big Data contexts. The MapReduce framework was also, thoroughly examined, focusing on its key features and advantages. Techniques for result extraction using clustering and classification were additionally covered.

Data collection, preparation, and dataflow management, along with the system architecture and tools and technologies employed have all been covered in detail in the practical implementation sections. Experimental results and analysis related to several areas of the study, including clustering, classification, HDFS, MapReduce, Kafka, and many more were also included and demonstrated.

Our implementation takes a step-by-step approach, starting with Kafka collecting data and moving through the steps of MapReduce for data aggregation, as well as employing clustering techniques like K-Means and Gaussian Mixture Models (GMM) to discover hidden patterns in our data. Following that, we move into the world of classification, employing K-Nearest Neighbors (KNN) and Support Vector Machines (SVM) on clustered data to acquire a better understanding of deeper patterns. Notably, HDFS was critical in maintaining the integrity of our data and facilitating rapid access to our key datasets.

Last but not least, while recognizing the valuable contributions made by the current re-search, the importance of also recognizing and acknowledging the inherent limitations as well as identifying prospective areas for future investigation and advancement within this ever-evolving and constantly expanding subject is highlighted.

7.1 Summary of Contributions

Our purpose in this section is to clarify the key contributions from each chapter. We intend to make efficient big data management easier to understand by conducting extensive study and defining each component. We conducted extensive research in our study, which provided useful insights that enhance both theoretical understanding and real world applications. These contributions are in line with the main goals of our academic journey:

i. **Revealing Challenges in Big Data Management:** In line with Objective 1, Chapter 2 goes into the topic of dealing with large and diverse datasets. We explain the complexities involved, from complex data storage to real-time analysis. These ideas set the stage for the approach we take in the following steps.

ii. **Navigating Hadoop's Domain:** In line with Objective 2, Chapter 2.2 navigates the landscape of Hadoop Distributed File System (HDFS), combining on Chapter 3.1 the core of MapReduce. Our investigation extends to the YARN architecture, with the goal of improving system performance and accelerating data processing.

iii. **Navigating Real Time Streams through Kafka:** Aligned with Objective 3, Chapter 2.4 breaks down the area of real time data processing, with a focus on platforms such as Twitter. Our research dives into Kafka's strength (Chapter 2.5) as a strong stream processing platform, which is directly related to the goal of collecting real time data.

iv. **Designing a complete system structure:** Following Objective 4, Unit 4, 5 reveals the blueprint for the proposed integrated system. This design combines HDFS, Map-Reduce, Kafka, avro, clustering, classification, and advanced analytics. It demonstrates how the system can manage and analyze real-world streaming data to benefit users.

All the contributions from each chapter have been linked in this section, resulting in a unified image that simplifies the world of big data and resonates in both academic and real-world situations. As we bring the key ideas from these chapters together, they illustrate the journey we have traveled to understand the difficulties of big data management.

7.2 Limitations and Future Work

While this research aims to shed light on several aspects of big data management and analysis, it is crucial to recognize some limits that present opportunities for future research and development.

Limitations:

i. **Considerations regarding Scope:** The combination of HDFS, MapReduce, and specialized analytical approaches is the focus of this research. While this targeted approach allows for in-depth investigation, it is crucial to note that the larger landscape of big data management comprises a diverse variety of developing technologies and frameworks.

ii. **Simplified Model Implementation:** The research's implementation includes a simplified model that demonstrates the integration of chosen components. While this approach improves clarity and understanding, it's important to remember that real-world scenarios may involve more complex and multidimensional systems. This constraint arises from the purposeful reduction of our study concepts in order to promote a clear presentation of our findings.

iii. **Scalability Evaluation:** Despite optimizing system performance, further examination of our architecture's scalability, particularly for larger and diverse datasets, would be valuable.

iv. **Real Time Processing Challenges:** Handling real-time streaming data, particularly from platforms like as Twitter, introduces challenges relating to data quality, noise, and content volatility, which may affect the accuracy of derived insights.

v. **Limited Comparison situations:** During the assessment process, the proposed architecture's performance and efficacy may have been examined in specified situations. A broader range of comparison situations might offer a more complete picture of the architecture's strengths and weaknesses.

vi. **Implications for Ethics and Law:** While the research focuses mostly on technological factors, the larger deployment of big data solutions also entails ethical and legal questions of data privacy, security, and compliance. These issues are not explicitly addressed in this study, but they are important concerns in real world applications.

The results of our analysis pave the foundation for future study and development in the field of big data management and analysis:

Future Work:

i. **Resource Management:** Investigating ways for dynamically distributing resources inside the HDFS and MapReduce frameworks might improve system performance even more, especially under variable workloads.

ii. **Improving Streaming Platforms:** We could discover more effective techniques to real-time data processing and analysis by investigating enhancements to streaming systems such as Kafka, therefore boosting the efficiency of dynamic data stream integration.

iii. **Exploration of Data Security:** Addressing security and privacy of data problems in the context of real time data streams provides a new environment for in depth research and discovery.

iv. **Better Integration of AI:** Advanced artificial intelligence approaches, such as machine learning and deep learning, may improve data processing accuracy and generate more in depth insights.

8. Appendices

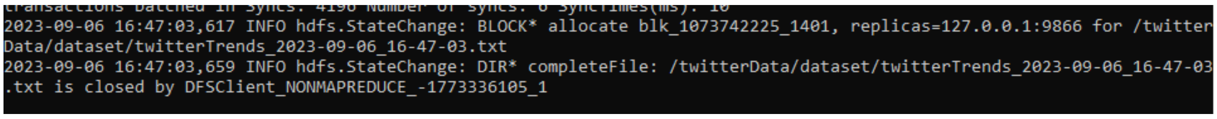
Appendix A: United States Indexing

Number	State	Abbreviation
1	Delaware	DE
2	Pennsylvania	PA
3	New Jersey	NJ
4	Georgia	GA
5	Connecticut	CT
6	Massachusetts	MA
7	Maryland	MD
8	South Carolina	SC
9	New Hampshire	NH
10	Virginia	VA
11	New York	NY
12	North Carolina	NC
13	Rhode Island	RI
14	Vermont	VT
15	Kentucky	KY
16	Tennessee	TN
17	Ohio	OH
18	Louisiana	LA
19	Indiana	IN
20	Mississippi	MS
21	Illinois	IL
22	Alabama	AL
23	Maine	ME
24	Missouri	MO
25	Arkansas	AR
26	Michigan	MI
27	Florida	FL
28	Texas	TX
29	Iowa	IA
30	Wisconsin	WI
31	California	CA
32	Minnesota	MN
33	Oregon	OR
34	Kansas	KS
35	West Virginia	WV
36	Nevada	NV

37	Nebraska	NE
38	Colorado	CO
39	North Dakota	ND
40	South Dakota	SD
41	Montana	MT
42	Washington	WA
43	Idaho	ID
44	Wyoming	WY
45	Utah	UT
46	Oklahoma	OK
47	New Mexico	NM
48	Arizona	AZ
49	Alaska	AK
50	Hawaii	HI
51	-	-

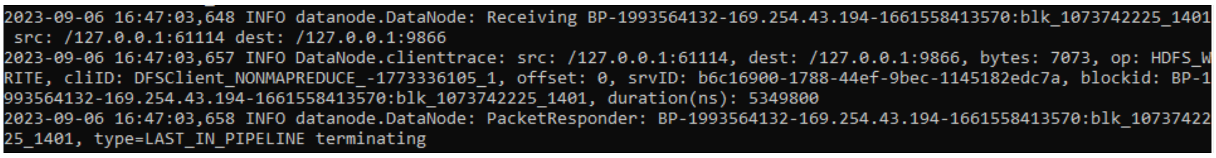
Table 7: United States indexing[1,2]

Appendix B: HDFS Logs



```
transactions batched in syncs: 4196 Number of syncs: 6 SyncTimes(ms): 10
2023-09-06 16:47:03,617 INFO hdfs.StateChange: BLOCK* allocate blk_1073742225_1401, replicas=127.0.0.1:9866 for /twitter
Data/dataset/twitterTrends_2023-09-06_16-47-03.txt
2023-09-06 16:47:03,659 INFO hdfs.StateChange: DIR* completeFile: /twitterData/dataset/twitterTrends_2023-09-06_16-47-03
.txt is closed by DFSClient_NONMAPREDUCE_-1773336105_1
```

Figure 20: NameNode log when creating directory and storing data.



```
2023-09-06 16:47:03,648 INFO datanode.DataNode: Receiving BP-1993564132-169.254.43.194-1661558413570:blk_1073742225_1401
 src: /127.0.0.1:61114 dest: /127.0.0.1:9866
2023-09-06 16:47:03,657 INFO DataNode.clienttrace: src: /127.0.0.1:61114, dest: /127.0.0.1:9866, bytes: 7073, op: HDFS_W
RITE, cliID: DFSClient_NONMAPREDUCE_-1773336105_1, offset: 0, srvID: b6c16900-1788-44ef-9bec-1145182edc7a, blockid: BP-1
993564132-169.254.43.194-1661558413570:blk_1073742225_1401, duration(ns): 5349800
2023-09-06 16:47:03,658 INFO datanode.DataNode: PacketResponder: BP-1993564132-169.254.43.194-1661558413570:blk_10737422
25_1401, type=LAST_IN_PIPELINE terminating
```

Figure 21: DataNode log when creating directory and storing data.

[1] https://www.sos.arkansas.gov/education/arkansas-history/history-of-the-flag/order-of-states-admission

[2] https://www.scouting.org/resources/los/states/

9. References

[1] Herland, M., Khoshgoftaar, T.M. & Wald, R. A review of data mining using big data in health informatics. Journal of Big Data 1, 2 (2014). https://doi.org/10.1186/2196-1115-1-2.

[2] Martin M (2013) Big Cdata/social media combo poised to advance healthcare. HPC Source: 33–35. http://www.scientificcomputing.com/digital-editions/2013/04/hpc-source-big-data-beyond.

[3] Landset, S., Khoshgoftaar, T.M., Richter, A.N. et al. A survey of open source tools for machine learning with big data in the Hadoop ecosystem. Journal of Big Data 2, 24 (2015). https://doi.org/10.1186/s40537-015-0032-1.

[4] Albi Nani, Valuing big data: An analysis of current regulations and proposal of frameworks, International Journal of Accounting Information Systems, Volume 51, 2023, 100637, ISSN 1467-0895, https://doi.org/10.1016/j.accinf.2023.100637.

[5] Laney D. 3D data management: controlling data volume, velocity and variety. META Group; 2001.

[6] Guido L. Geerts, Daniel E. O'Leary, V-Matrix: A wave theory of value creation for big data, International Journal of Accounting Information Systems, Volume 47, 2022, 100575, ISSN 1467-0895, https://doi.org/10.1016/j.accinf.2022.100575.

[7] Hariri, R.H., Fredericks, E.M. & Bowers, K.M. Uncertainty in big data analytics: survey, opportunities, and challenges. J Big Data 6, 44 (2019). https://doi.org/10.1186/s40537-019-0206-3.

[8] Niculescu, Virginia. (2020). On the Impact of High Performance Computing in Big Data Analytics for Medicine. Applied Medical Informatics. 42. 9-18.

[9] Karmasphere (2011) Understanding the Elements of Big Data: More than a Hadoop Distribution.

[10] Duque Barrachina, A., O’Driscoll, A. A big data methodology for categorising technical support requests using Hadoop and Mahout. Journal of Big Data 1, 1 (2014). https://doi.org/10.1186/2196-1115-1-1.

[11] Petra Leimich, Josh Harrison, William J. Buchanan. A RAM triage methodology for Hadoop HDFS forensics. Digital Investigation,Volume 18,2016,Pages 96-109,ISSN 1742-2876, https://doi.org/10.1016/j.diin.2016.07.003.

[12] https://hadoop.apache.org/.

[13] Tom White. 2015. Hadoop: The Definitive Guide (4th. ed.). O'Reilly Media, Inc.

[14] Viraaji Mothukuri, Sai S. Cheerla, Reza M. Parizi, Qi Zhang, Kim-Kwang Raymond Choo, BlockHDFS: Blockchain-integrated Hadoop distributed file system for secure provenance traceability, Blockchain: Research and Applications, Volume 2, Issue 4,2021,100032,ISSN 2096-7209, https://doi.org/10.1016/j.bcra.2021.100032.

[15] J. Dean, S. Ghemawat, Mapreduce: simplified data processing on large clusters, Commun. ACM 51 (1) (2008) 107–113.

[16] Senthilkumar, Anitha & Metilda, Mary. (2022). Apache Hadoop based Effective Sentiment Analysis on Demonetization and Covid-19 Tweets. Global Transitions Proceedings. 3. 10.1016/j.gltp.2022.03.021.

[17] Parameshachari, B.D. Big Data Analytics on Weather Data: Predictive Analysis Using Multi Node Cluster Architecture. Int. J. Comput. Appl., 0975-8887.

[18] G. Mertens , L. Gerritsen, S. Duijndam, E. Salemink, I.M. Engelhard, Fear of the coronavirus (COVID-19): Predictors in an online study conducted in March 2020, J. Anxiety Disord. 74 (2020) 102258.

[19] Shaaopeng Guan, Conghui Zhang, Yilin Wang, Wenqing Liu, Hadoop-based secure storage solution for big data in cloud computing environment, Digital Communications and Networks, 2023,, ISSN 2352-8648, https://doi.org/10.1016/j.dcan.2023.01.014.

[20] Shengying Yang, Wuyin Jin, Yunxiang Yu, and Kamarul Faizal Hashim. 2023. Optimized hadoop map reduce system for strong analytics of cloud big product data on amazon web service. Inf. Process. Manage. 60, 3 (May 2023). https://doi.org/10.1016/j.ipm.2023.103271.

[21] Mohamed Haggag, Mohsen M. Tantawy, Magdy M.S. El-Soudani, Token-based authentication for Hadoop platform, Ain Shams Engineering Journal, Volume 14, Issue 4, 2023,101921,ISSN 2090-4479, https://doi.org/10.1016/j.asej.2022.101921.

[22] https://aws.amazon.com/streaming-data/.

[23] http://www.alexa.com/siteinfo/twitter.com – http://mywptips.com/top-microblogging-sites-list/.

[24] Troussas, Christos & Krouska, Akrivi & Virvou, Maria. (2019). Trends on Sentiment Analysis over Social Networks: Pre-processing Ramifications, Stand-Alone Classifiers and Ensemble Averaging. 10.1007/978-3-319-94030-4_7.

[25] Swathi, T.; Kasiviswanath, N.; Rao, A.A. An optimal deep learning-based LSTM for stock price prediction using twitter sentiment analysis. Appl. Intell. 2022, 52, 13675–13688.

[26] https://help.twitter.com/en.

[27] Pirim, H., Nagahi, M., Larif, O. et al. Integrated twitter analysis to distinguish systems thinkers at various levels: a case study of COVID-19. Appl Netw Sci 8, 12 (2023). https://doi.org/10.1007/s41109-022-00520-9.

[28] Anwar Hridoy, S.A., Ekram, M., Islam, M.S. et al. Localized twitter opinion mining using sentiment analysis. Decis. Anal. 2, 8 (2015). https://doi.org/10.1186/s40165-015-0016-4.

[29] Wang, Y.; Guo, J.; Yuan, C.; Li, B. Sentiment Analysis of Twitter Data. Appl. Sci. 2022, 12, 11775. https://doi.org/10.3390/app122211775.

[30] N. Chintalapudi, G. Battineni, M. Di Canio, G.G. Sagaro, F. Amenta, Text mining with sentiment analysis on seafarers' medical documents, Int. J. Inf. Manag. Data Insights 1 (2021) 100005 Wilson.

[31] Giachanou, A.; Crestani, F. Like it or not: A survey of Twitter sentiment analysis methods. ACM Comput. Surv. (CSUR) 2016, 49, 28.

[32] A Step-By-Step Guide to Getting Started on Twitter. Available online: http://img.constantcontact.com/docs/pdf/getting-started-on-twitter.pdf.

[33] Khan, Faisal. (2021). Apache kafka with real-time data streaming.

[34] Kreps, J., Narkhede, N. and Rao, J., 2011, June. Kafka: A distributed messaging system for log processing. In Proceedings of the NetDB (Vol. 11, No. 2011, pp. 1-7).

[35] https://kafka.apache.org/documentation/.

[36] Cristian Martín, Peter Langendoerfer, Pouya Soltani Zarrin, Manuel Díaz, Bartolomé Rubio, Kafka-ML: Connecting the data stream with ML/AI frameworks, Future Generation Computer Systems,Volume 126,2022,Pages 15-33, ISSN 0167-739X, https://doi.org/10.1016/j.future.2021.07.037.

[37] Lazidis, Apostolos & Tsakos, Konstantinos & Petrakis, Euripides. (2022). Publish-Subscribe approaches for the IoT and the cloud: Functional and performance evaluation of open-source systems. Internet of Things. 19. 100538. 10.1016/j.iot.2022.100538.

[38] https://hazelcast.com/glossary/deserialization/.

[39] https://avro.apache.org/docs/1.11.1/specification/.

[40] https://hazelcast.com/glossary/serialization/.

[41] Shivayogappa, Ashok & Shivashankar, Supreeth. (2020). A Comparison of HDFS File Formats: Avro, Parquet and ORC. International Journal of Advanced Science and Technology. 29. 4665-4675. 10.5281/zenodo.7027910.

[42] Chen, C., Xu, Y., Zhu, Y. et al. Online MapReduce scheduling problem of minimizing the makespan. J Comb Optim 33, 590–608 (2017). https://doi.org/10.1007/s10878-015-9982-7.

[43] Dayalan, Muthu. (2018). MapReduce: Simplified Data Processing on Large Cluster. International Journal of Research and Engineering. 5. 399-403. 10.21276/ijre.2018.5.5.4.

[44] https://hadoop.apache.org/docs/stable/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html.

[45] Hossein Azgomi, Mohammad Karim Sohrabi, MR-MVPP: A map-reduce-based approach for creating MVPP in data warehouses for big data applications, Information Sciences, Volume 570, 2021, Pages 200-224, ISSN 0020-0255, https://doi.org/10.1016/j.ins.2021.04.004.

[46] https://hadoop.apache.org/docs/stable/hadoop-yarn/hadoop-yarn-site/YARN.html.

[47] A. P. Windarto, “Implementation of Data Mining on Rice Imports by Major Country of Origin Using Algorithm Using K-Means Clustering Method,” Int. J. Artif. Intell. Res., vol. 1, no. 2, p. 26, 2017.

[48] Rohman, Eka & Rachmad, Aeri. (2020). Clustering Tourist Destinations Based on Number of Visitors Using the K-Mean Method. 10.2991/assehr.k.200303.075.

[49] Fitri, Esmi & Ariansyah, M. & Winarno, Sri & Budiman, Fikri & Rohmani, Asih & Zeniarja, Junta & Sugiarto, Edi. (2023). PERFORMANCE OF K-MEANS CLUSTERING AND KNN CLASSIFIER IN FISH FEED SELLER DETERMINATION MODELS. Jurnal Teknik Informatika (Jutif). 4. 485-491. 10.52436/1.jutif.2023.4.3.725.

[50] J. MacQueen, Some methods for classification and analysis of multivariate observations, Presented at the Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics, The Regents of the University of California, 1967.

[51] Debasmita Das, Parthajit Kayal, Moinak Maiti,A K-means clustering model for analyzing the Bitcoin extreme value returns, Decision Analytics Journal, Volume 6, 2023, 100152, ISSN 2772-6622, https://doi.org/10.1016/j.dajour.2022.100152.

[52] Afolabi, Abiodun. (2023). Implementing K-Means Clustering. 10.13140/RG.2.2.19019.57121.

[53] Zefeng Huang, Zhonghua Gou,Gaussian Mixture Model based pattern recognition for understanding the long term impact of COVID-19 on energy consumption of public buildings, Journal of Building Engineering, Volume 72, 2023, 106653,ISSN 2352-7102, https://doi.org/10.1016/j.jobe.2023.106653.

[54] Smaranda Belciug, Dominic Gabriel Iliescu,Deep learning and Gaussian Mixture Modelling clustering mix. A new approach for fetal morphology view plane differentiation, Journal of Biomedical Informatics, Volume 143, 2023, 104402, ISSN 1532-0464,https://doi.org/10.1016/j.jbi.2023.104402.

[55] S. Faisal, "Klasifikasi data minning menggunakan algoritma c4. 5 terhadap kepuasan pelanggan sewa kamera cikarang,"Techno Xplore: Jurnal Ilmu Komputer dan Teknologi Informasi, vol. 4, no.1, pp. 38-45, 2019, doi:https://doi.org/10.36805/technoxplore.v4i1.541.

[56] A. Yazdinejad, A. Dehghantanha, R. M. Parizi, and G. Epiphaniou, "An optimized fuzzy deep learning model for data classification based on NSGA-II," Neurocomputing, vol. 522, pp. 116–128, Feb. 2023, doi: 10.1016/J.NEUCOM.2022.12.027.

[57] (2023). Naive Bayes and KNN for Airline Passenger Satisfaction Classification: Comparative Analysis. Journal of Information System Exploration and Research. 1. 10.52465/joiser.v1i2.167.

[58] Adadi, A. A survey on data-efficient algorithms in big data era. J Big Data 8, 24 (2021). https://doi.org/10.1186/s40537-021-00419-9.

[59] S. Adhikary and S. Banerjee, “Introduction to Distributed Nearest Hash: On Further Optimizing Cloud Based Distributed kNN Variant,” Procedia Comput Sci, vol. 218, pp. 1571–1580, 2023, doi: 10.1016/j.procs.2023.01.135.

[60] A. Ali, M. Hamraz, N. Gul, D. M. Khan, S. Aldahmani, and Z. Khan, "A k nearest neighbour ensemble via extended neighbourhood rule and feature subsets," Pattern Recognit, vol. 142, p. 109641, Oct. 2023, doi: 10.1016/J.PATCOG.2023.109641.

[61] Hongyi Ge, Xiaodi Ji, Xuejing Lu, Ming Lv, Yuying Jiang, Zhiyuan Jia, Yuan Zhang, Identification of heavy metal pollutants in wheat by THz spectroscopy and deep support vector machine, Spectrochimica Acta Part A: Molecular and Biomolecular Spectroscopy, Volume 303, 2023, 123206, ISSN 1386-1425, https://doi.org/10.1016/j.saa.2023.123206.

[62] S. Faisal, "Klasifikasi data minning menggunakan algoritma c4. 5 terhadap kepuasan pelanggan sewa kamera cikarang,"Techno Xplore: Jurnal Ilmu Komputer dan Teknologi Informasi, vol. 4, no.1, pp. 38-45, 2019, doi:https://doi.org/10.36805/technoxplore.v4i1.541.