Big Data: Analyzing Apache Spark And Kafka For Processing Iterative Algorithms

Akanksha Pandey[1], Astitwa Bhargava[2]
[1, 2] Rgnclc, Nliu, Bhopal
National Law Institute University

Abstract- *With the advancement in the technology, management and processing of data in real-time has become the need of the hour. Whatever data is being received needs to be stabilizing very frequently, in order to get the maximum value out of it. The term best defines this is the 'Big Data', which deals with Volume, Velocity and Variety. Managing such a large amount of data is not only expensive but also time consuming. The primary purpose of this research paper is to address this issue by considering some iterative data like 'Product Structure'. Product Structures are unavoidable and the maintenance of its completeness, consistency and correctness requires a lot of effort.*

Further, in this research paper, Hadoop technology is proposed by the researchers for cost effectiveness. There is an issue of time consumption for the incoming data in real time, so to address this problem 'Kafka' and 'Spark' are used. Researchers used 'Spark' and 'Apache Kafka' to resolve the issue of data processing and data streaming respectively and further 'Sqoop' is used for local database.

Keywords- Big Data, Hadoop, HDFS, Product Structure, Kafka, Spark, Hive, Sqoop.

I. INTRODUCTION

Big Data is a term that describes the large volume of data- structured, unstructured and semi-structured – that overwhelm a business on a day-to-day basis. It's been seen that nowadays large amount of data are being generated with the blink of an eye. But it is not the amount of data that is important; it's what the organization does with the data that matters. Big Data can be analyzed for insights that lead to better decisions and strategic business moves. Since the amount of data that is created and stored on a global level is almost unbelievable, and it is increasing regularly, so we can say that the potential of Big Data is very high. That means there is even more potential to obtain key insights from business information – yet only a small percentage of data is actually analyzed. One can take data from any source and analyze it to find answers that would make possible the following- 1) Cost reduction, 2) New product development and optimized offerings, 3) Time reduction and 4) Smart decision making. The way organizations manage and derive insights from it is changing the way the world uses business information. There are various areas where we can see its effects including 1) Banking, 2) Education, 3) Government, 4) Health Care, 5) Manufacturing, 6) Retails. After processing and analyzing the data and insights, products and services that emerge from analysis, we get the primary value from the big data. The sweeping changes in big data technologies and management approaches needed to be accompanied by similar dramatic shifts in how data supports decisions and product/service innovation. The sources of Big Data are required to be understood first rather than directly moving towards how Big Data works for the business. The sources for big data generally fall into one of the three categories: 1) Streaming data, 2) Social media data, 3) Publicly available sources. These are considered as the basic sources for fetching of data that can be further converted into information. After identifying all the potential sources for data, consider the decision one will have to make once the process of harnessing information begins. These include: 1) How to store and manage it, 2) How much to analyze, 3) How to use any insights you uncover. Now, as per the use of the information one can move ahead with the well-furnished information gained from a large amount of data.

Since unstructured data is very large and complex, for analyzing such kind of data Hadoop is used as it has distributed storage and distributed processing framework. This is an open source and is freely available. Hadoop is designed to support Big Data – Data that is too big for any traditional database technologies to accommodate. Since it is not usually possible for traditional technologies to carry Big Data, Hadoop came into picture. Unstructured data is BIG – really BIG in maximum cases. In HDFS data is being stored as files. Hadoop does not have any schema or a structure for the data that has to get stored. Hadoop uses applications like Sqoop, HIVE, HBASE etc in order to import and export data from traditional and non-traditional databases. Hadoop will import the unstructured data for converting it into a structured form and then after getting structured or semi-structured data it will export the data into traditional databases for further analysis. Hadoop is a very powerful tool for writing customized codes. Typically complex algorithms are being involved for the analysis of unstructured data. Hadoop framework could be used for exploiting the benefits and achieving the efficiency and reliability, since, algorithms of any complexity can be resolved by the programmers. User needs to understand the data at a crude level and appropriately program any algorithm they want to use. Hadoop gives this kind of flexibility to the users.

Terabytes of data and sometimes more can be processed that makes to run different applications on systems that involve thousands of nodes and due to the use of distributed File System in Hadoop, it is made possible. During cases when there is node failure it can be continued with its operations. Catastrophic system failure would not occur due to a single point of failure. For large clusters of data Hadoop, an open-source MapReduce implementation is being built up. JobTracker is a single master node and it contains different slave nodes called TaskTrackers. Hadoop runs at its best in Ubuntu. As the data that is being collected from the social sites (say Twitter or Facebook) is random in nature, hence the processing of the data is to be done in such a way that the randomness in the data should get removed. According to the types of data its arrangement needs to be done and then sorting of the arranged data is to be done. This sorting is done for better understanding. Sorting of username can be according to the expression they use or the departments they belong to, or any other criteria. MapReduce technique with suitable algorithm could be used for the processing of the data. As per the user specifications, MapReduce processes the data in order to arrange it. There are two types of HDFS nodes: DataNode and NameNode. The DataNode stores the data blocks of the files in HDFS and NameNode contains the metadata, with the enumeration of blocks of HDFS and a list of DataNode in the cluster.

MapReduce technique is mainly used for parallel processing of data set across various clusters known as filtering, performed by the map function and generating computation result by aggregation, which is the reduce function. Mapper, Combiner and Partitioner are the 3 sub tasks that are needed to be performed by the Map Job. Mapper involves the mapping of data, Combiner combines the mapped data and Partitioner splits the data into small clusters, after which the shuffling key/value of map job to unique reduce job is done. Mapper, Combiner and Partitioner all three of them perform their tasks separately but each of its performance is depended upon each other. Joiner and reducer are 2 subtasks involved in reduce job. The joiner holds the joining of the intermediate results from the map jobs and reducer subtask is used for performing aggregation. After the map and reduce jobs, the end result is stored in Hadoop Distributed File System (HDFS). The Hadoop distributed File System output for a MapReduce job can be used to store the final results of map reduce process and the output can be viewed by browsing the file system in the name node log. The processing of the MapReduce process is given by job details log and then the job is considered to be completed. NameNode log also contains information about the cluster summary, capacity of the file system, distributed file system used and remaining and also the number of live nodes and dead nodes. The NameNode and JobTracker Details are obtained as the result of the execution of Map Reduce process. NameNode log can be used to locate the output directory of the file system and the output of the map reduce job.

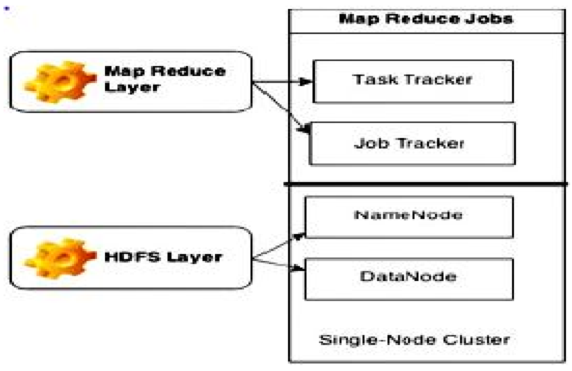

Fig 1: Map Reduce Jobs

MapReduce programming model is used for the processing of Hadoop in an efficient manner, but when it comes to iterative algorithms in such kind of scenarios, MapReduce is not that much efficient. In case of MapReduce, after every Map Phase, the resultant is taken from memory and then it is exported into disk. Importing and exporting of data in an iterative algorithm needs to be done every time, which will culminate into a very tedious and time consuming process. Apache Spark performs all the computation in its own memory and persist the data in memory whenever necessary and that is the solution to the problem about which we were talking about.

II. PRODUCT STRUCTURE

For hierarchical decomposition of a product, we use a term called Product Structure, typically known as the bill of materials (BOM). As business becomes more responsive to unique consumer tastes and derivatives products grow to meet the unique configuration, BOM management can become unmanageable. It contains all the necessary information starting from a smallest component to the most complex one. Following activities takes place within product structure modeling: 1) Define product components, 2) Define product assortment, 3) Product structuring, 4) Create master structure, 5) Documenting, 6) Define product structure views.

In this paper the Product Structure is considered as an example in order to show what all information is required for manufacturing a product. And how well the information is made available when that is required.

III. DATA CONSUMPTION & PROCESSING

In the data consumption phase, we need to gather the Product Structure data. Kafka is used to consume that data and that is what we are doing in this research which is considered to be the first step. This is followed by processing phase where the consumed Product Structure data is compared with the existing Product Structure data already stored in HDFS or HIVE. For real-time data consumption and data processing, here in this paper, we are using two tools namely Kafka and Spark. In data consumption phase data is extracted from the data source and consumed via Kafka. Spark is configured to pull data from Kafka as and when it is available and that too in the data processing phase.

A. Apache Kafka

Regarding large amount of data, which is actually what the Big Data talks about, we have two main challenges. The first challenge is the collection of large volume of data and the second challenge is analyzing that collected large amount of data. A messaging system is required in order to overcome those challenges. Kafka is designed for distributed high throughput systems. For replacement of more traditional message broker Kafka tends to work very efficiently. On comparing Kafka with other different messaging systems, it provides us with better built-in partitioning, throughput, inherent fault-tolerance and replication, so if one wants to go for large scale message processing applications, Apache Kafka is the right choice. Though an application needs to be focused more on data, they don’t have to worry about how one is going to share it because the main responsibility of a messaging system is transferring data from one application to another. Reliable messaging queuing is the concept on which the Distributed messaging is based. Between client applications and messaging system, queuing of messages is done asynchronously. Apache Kafka enables us to pass messages from one end-point to another. It also handles high volume of data through robust queuing, making it a distributed publish-subscribe messaging system. For both online and offline message consumption Kafka is suitable. For preventing data loss, Kafka messages are persisted on the disk and also replicated within the cluster. For Zookeeper synchronization service Kafka is on the top. For real time streaming data analysis Apache Kafka integrates very well with Apache Storm and Spark. The following are the components of Kafka:

(i) Kakfa Broker: Multiple brokers are typically required in a Kafka cluster in order to maintain load balance. Zookeeper is used for maintaining Kafka cluster state since Kafka broker are stateless. TB of messages are handled by each broker without performance impact and hundreds of thousands of read and writes per seconds are handled at an instance by one Kafka broker. Zookeeper has the privilege for doing election of Kafka leader.

(ii) ZooKeeper: Kafka broker are managed and coordinated by the using Zookeeper. If there is a presence of any new broker or if any incident of failure of broker occurs in the Kafka system, Zookeeper service is mostly used in such scenario to notify about such incidents to producer and consumer. Producer and consumer will get the information about the presence or failure of broker from the Zookeeper and then they will start taking quick decisions and also start coordinating their task with some other broker.

(iii) Kafka Producer: Brokers get data that is being pushed by the producers. All the producers search for a new broker and automatically start sending messages to that new broker. Kafka producer sends messages as fast as they can and this sending also depends on how fast broker handles the messages. By that we mean to say that Kafka producer doesn’t wait for acknowledgements from broker side.

(iv) Kafka Consumer: Kafka consumer needs to maintain how many messages have to be consumed by using partition offset, because Kafka brokers are stateless. If the consumer has consumed all prior messages, in such cases consumer will acknowledge a particular message offset. In order that a consumer wants to have a buffer of bytes ready to consume, the consumer will issue an asynchronous pull request. By supplying an offset value the consumer can skip or rewind to any point in a partition. Zookeeper is responsible for notifying about the consumer offset value.

B. Apache Spark:

For fast computation, Apache Spark is suggested to be used as it provides lightning-fast cluster computing technology. For more types of computation like interactive queries and stream processing, one can extend MapReduce, since Spark is based upon Hadoop MapReduce. Apache Spark owns in-memory cluster computing that increases the processing speed of an application and this feature is considered the main feature of Spark. Different kinds of workloads such as batch applications, iterative algorithms, interactive queries and streaming are all covered under spark. We can say that spark is designed in a way to cover all of the above-mentioned workloads. By maintaining separate tools, it reduces management burden, apart from supporting all these workloads in a respective system. Spark uses Hadoop in two ways – One is storage and second is processing. Since Spark has its own cluster management computation, it uses Hadoop for storage purpose only. Apache Spark provides three main features that are 1) Speed, 2) Supports multiple languages, 3) Advanced Analytics. Following are the components of Apache Spark:

(i) Apache Spark Core: For spark platform, the underlying general engine on which all other functionality is built upon is Spark Core. Referencing datasets in external storage systems and In-memory computing all are being provided under it.
(ii) Spark SOL: Introduction of a new data abstraction called SchemaRDD is through Spark SQL. It is a component on top of Spark Core and support to structured and semi-structured is being provided under this.
(iii) Spark Streaming: Spark streaming leverages Spark Core's fast scheduling capability to perform streaming analytics. It consumes data in mini-batches and performs RDD (Resilient Distributed Datasets) transformation on those mini-batches of data.
(iv) MLlib (Machine Learning Library): Because of the distributed memory-based Spark architecture above Spark, there is MLlib which is a distributed machine learning framework. When a benchmark is to be done MLlib developers compare it against Alternating Least Squares (ALS) implementation. As compared to Hadoop disk-based version of Apache Mahout (before Mahout gained a Spark interface), Spark Mllib is found to be nine times faster than Hadoop disk-based version of Apache Mahout.
(v) GraphX: On the top of spark, GraphX is placed which is distributed graph-processing framework.

C. Sqoop

Apache Sqoop is used for efficiently transferring bulk data between Apache Hadoop and structure data stores such as relational databases. For an efficient execution of tasks at a much lower cost, Sqoop helps to offload certain tasks (such as ETL processing) from the EDW to Hadoop. For extracting data from Hadoop and exporting it into external structures data stores Sqoop can be used. Some relational databases such as Teradata, Netezza, Oracle, MySQL, Postgres and HSOLDB for working with the following relational databases Sqoop can be used. Apache Sqoop does the following to integrate bulk data movement between Hadoop and Structures data stores:

(i) For importing sequential datasets from mainframe: It satisfies the growing need to move data from mainframe to HDFS
(ii) Import direct to ORCFiles: Improved compression and light-weight indexing for improved query performance.
(iii) For importing data: Moves certain data from external stores and EDWs into Hadoop to optimize cost-effectiveness of combined data storage and processing
(iv) Parallel data transfer: For faster performance and optimal system utilization
(v) Fast data copies: From external systems into Hadoop
(vi) Efficient data analysis: Improves efficiency of data analysis by combining structures data with unstructured data in a schema-on-read data lake
(vii) Managing load balancing: Mitigates excessive storage and processing loads to other systems.

YARN coordinates data ingest from Apache Sqoop and other services that delivers data into the Enterprise Hadoop cluster.

D. HIVE

For processing structured data in Hadoop one can use Hive, as Hive is a data warehouse infrastructure tool. In order to make querying and analyzing easy, it resides on the top of Hadoop to summarize Big Data. Facebook has started Hive initially and then later Apache Software Foundation took it up for doing further development as an open source and named it as Apache Hive and from then onwards many companies started using it. For example, Amazon uses it in Amazon Elastic MapReduce. Hive is not 1) A relational database, 2) A design for OnLine Transaction Processing (OLTP), 3) A language for real-time queries and low-level updates. Features of Hive 1) It stores schema in a database and processed data into HDFS, 2) It is designed for OLAP, 3) It provides SQL type language for querying called HiveQL or HQL 4) It is familiar, fast, scalable and extensible.

IV. INTEGRATION OF KAFKA & SPARK

In this paper, Product Structure data is streamed using Kafka and it is further processed using Spark. Both the data consumption and data processing operations can be performed in real-time. Extracting the product structure data as and when it is available in the data source (data source can be a data server) and then immediately streaming it via Kafka makes the product structure data available for real time processing.

In the data processing phase, the resultant value after comparing the product structure data streamed into Spark with the Product Structure data already stored in Hive or HDFS is exported using Sqoop. To carry out this process, it is essential to check the correctness of the product structure data initially, because after the consumption phase the product, structure data is directly compared with other existing product structure data stored in HDFS. Hence any defects in product structure will lead to improper productions. Fig 2 shows the flow chart for data consumption, processing and then exporting the resultant delta value.

Hence the product structure data is extracted from the data source and it is consumed via Kafka into Spark. In Kafka when the message is published to the Kafka topic zookeeper will be updated. This message will be consumed by the consumer based on its requests. The consumer sends the request along with an offset. This offset specifies the position of the message from where it wants to read. These offsets are maintained by the Zookeeper every time there is communication between the Kafka producers, Kafka consumer and the Zookeeper. This communication ensures that the message is consumed into Spark only once and hence it avoids redundancy. Since Zookeeper keeps Kafka topic, producer and consumer coordinate with each other. Kafka consumer can be accurate about which messages are successfully consumed into Spark. This is done by keeping track of the message offsets. Hence it ensures that no message is delivered to spark multiple times.

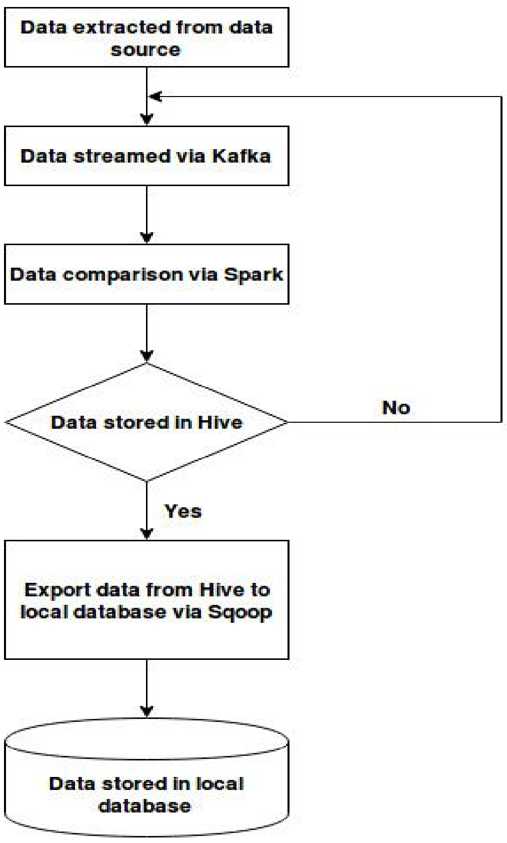
Data extracted from data source
Data streamed via Kafka
Data comparison via Spark
Data stored in Hive
No
Yes
Export data from Hive to local database via Sqoop
Data stored in local database

Fig 2: Flow chart for consumption, processing and exporting Product Structure

Product Structure data consumed via Kafka is in xml format. Spark puts this xml data into HDFS or Hive so it can be further compared. During the comparison, if the product structure data is consumed for the first time, it is directly stored without any comparison. Next time when the product structure data is consumed, it is compared with the already existing product structure data. After the comparison, the old product structure data is deleted and it is replaced with new product structure data. Finally, the change between these two product structure data is stored in HDFS. This computed value can then be exported into local database via Sqoop so that it can be queried and accessed from local database.

V. RESULT & ANALYSIS

After every map and reduce task, Hadoop uses the map reduce model in-order to dump the data into disk. If we talk about timing, then fetching the data from disk after every map and reduce task, takes a lot of time. And in cases where we hold some kind of iterative algorithms this time consumption would add to a disadvantage. Spark and Hadoop performance measurement is shown in figure 3. The time taken by Spark and Hadoop for executing an iterative algorithm is shown in figure 4.

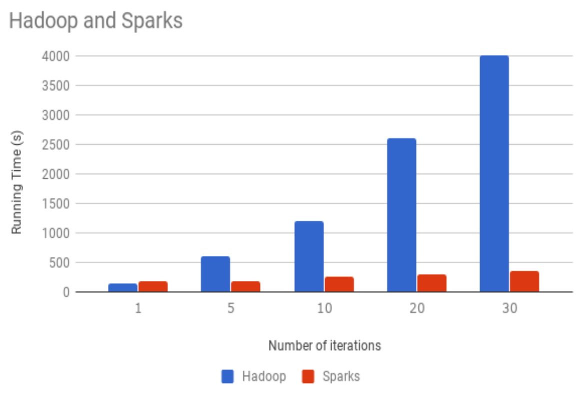

Fig 3: Performance measurement for Spark and Hadoop

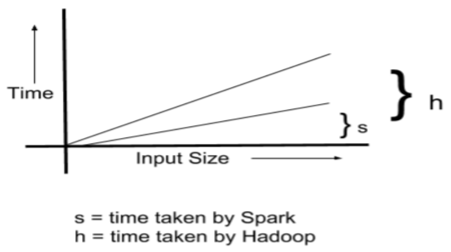

Fig 4: Time taken to execute an iterative algorithm in Spark and Hadoop

The input data size increases in an iterative algorithm and the time taken for executing it also increases in Hadoop as well as in Spark that could be seen in figure 3 and figure 4. But on comparing both the Hadoop and Spark time of execution of an iterative algorithm we can see that time taken by Spark in order to complete the execution is approximately 10 times lesser than that of the time taken by Hadoop. We can notice an exponential increase with respect to the size of data and also its time in case of Hadoop when Hadoop needs to execute an iterative algorithm.

But, in same scenario when an iterative algorithm needs to be executed in Spark, it varies linearly with respect to the size of data.

So for processing of data in real-time we can use Spark so that we can overcome with the above mentioned problem. If we compare Hadoop and Spark, we can see that Hadoop's processing time of an iterative algorithm is 100 times more than that of Spark. Spark set a net world record in 100TB sorting, beating the previous record held by Hadoop MapReduce by three times, using only one-tenth of the resources; it received a new SQL query engine with a state-of-the-art optimizer; and many of its built-in algorithms became five times faster.

Processing of data can be 100x times faster than that of hours of time taken by Hadoop, and Spark on the same can process it 100x times faster can say hours will be changed into few seconds if we use Spark for processing iterative algorithm. For real time consumption and processing of tasks we need to integrate together both Kafka and Spark. Real-time data analysis means processing data generated by the real-time events streams coming in at the rate of millions of events per seconds.

VI. CONCLUSION

For efficiently managing and processing of data, Hadoop is an ideal platform. In order to process data in parallel, batch-processing mode Hadoop is considered an ideal and efficient solution. There can be a case when one need to process iterative algorithms but then, it comes up with several drawbacks. As in Hadoop after every map reduce phase, the fetching of data needs to be done from the disk. But, in case of Spark the data persist within the memory.

Supporting streaming of data along with distributed processing is the ability that gives strength to Spark. Therefore, if we want to talk about processing of data in real time this would give a useful combination that delivers in near real-time. If we talk about MapReduce, it was basically designed to perform batch and distributed processing on large amount of data and thus it would not fulfill the requirement that is being fulfilled by Spark. If MapReduce wants to process data in real time it can do but if we consider the speed or time taken by it, it would be nowhere close to that of Spark.

REFERENCES

[1] Mayer-Schönberger, V.; Cukier, K. Big Data: A Revolution that Will Transform How We Live, Work, and Think; Houghton Mifflin Harcourt: Boston, MA, USA, 2013.

[2] Sagiroglu, S.; Sinanc, D. Big data: A review. In Proceedings of the 2013 International Conference on Collaboration Technologies and Systems (CTS), San Diego, CA, USA, 20–24 May 2013; pp. 42–47.

[3] Hashem, I.A.T.; Yaqoob, I.; Anuar, N.B.; Mokhtar, S.; Gani, A.; Ullah Khan, S. The rise of "big data" on cloud computing: Review and open research issues. Inf. Syst. 2015, 47, 98–115.

[4] Sharma, S. Rise of Big Data and related issues. In Proceedings of the 2015 Annual IEEE India Conference (INDICON), New Delhi, India, 17–20 December 2015; pp. 1–6.

[5] Eynon, R. The rise of Big Data: What does it mean for education, technology, and media research Learn. Media Technol. 2013, 38, 237–240.

[6] Wang, H.; Jiang, X.; Kambourakis, G. Special issue on Security, Privacy and Trust in network-based Big Data. Inf. Sci. Int. J. 2015, 318, 48–50.

[7] Thuraisingham, B. Big data security and privacy. In Proceedings of the 5th ACM Conference on Data and Application Security and Privacy, San Antonio, TX, USA, 2–4 March 2015; pp. 279–280.

[8] Rijmenam, V. Think Bigger: Developing a Successful Big Data Strategy for Your Business; Amacom: New York, NY, USA, 2014.

[9] Big Data Working Group; Cloud Security Alliance (CSA). Expanded Top Ten Big Data Security and Privacy. April 2013. Available online: https://downloads.cloudsecurityalliance.org/initiatives/bdwg/Expanded_

[10] Top_Ten_Big_Data_Security_and_Privacy_Challenges.pdf (accessed on 9 December 2015).

[11] Meng, X.; Ci, X. Big data management: Concepts, techniques and challenges. Comput. Res. Dev. **2013**, 50, 146–169.

[12] Chen, M.; Mao, S.; Liu, Y. Big data: A survey. Mob. Netw. Appl. 2014, 19, 171–209.

[13] Khan, M.A.-U.-D.; Uddin, M.F.; Gupta, N. Seven V's of Big Data understanding Big Data to extract value.

[14] In Proceedings of the 2014 Zone 1 Conference of the American Society for Engineering (ASEE Zone 1 2014), Bridgeport, CT, USA, 3–5 April 2014.

[15] Cumbley, R.; Church, P. Is “Big Data” creepy? Comput. Law Secur. Rev. 2013, 29, 601–609.

[16] Dijcks, J.P. Oracle: Big data for the enterprise. In Oracle White Paper; Oracle Corporation: Redwood City, CA, USA, 2012.

[17] Minelli, M.; Chambers, M.; Dhiraj, A. Big Data, Big Analytics: Emerging Business Intelligence and Analytic Trends for Today’s Businesses; JohnWiley & Sons: New York, NY, USA, 2013.

[18] F. Provost, and T. Fawcett,“Robust Classification for Imprecise Environments”, Machine Learning, 42/3, 203–231, 2001.

[19] D. Lewis, and J. Catlett, “Heterogeneous Uncertainity Sampling for Supervised Learning”, In Proceedings of the Eleventh International Conference of Machine Learning, pp.148–156 San Francisco, CA. Morgan Kaufmann, 1994.

[20] S. Dumais, J. Platt, D. Heckerman, and M. Sahami, “Inductive Learning Algorithms and Representations for Text Categorization”. In Proceedings of the Seventh International Conference on Information and Knowledge Management, pp. 148–155, 1998.

[21] D Mladeni´c, and M. Grobelnik, “Feature Selection for Unbalanced Class Distribution and Naive Bayes”. In Proceedings of the 16th International Conference on Machine Learning, pp. 258–267. Morgan Kaufmann, 1999.

[22] K, Woods, C, Doss, K. Bowyer,J. Solka, C. Priebe, and P. Kegelmeyer, “Comparative Evaluation of Pattern Recognition Techniques for Detection of Microcalcifications in mammography”, International Journal of Pattern Recognition and Artificial Intelligence, 7(6), 1417–1436, 1993.

[23] T. Fawcett, and F. Provost, “Combining Data Mining and Machine Learning for Effective User Profile”, In Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining, pp. 8–13 Portland, OR. AAAI, 1996.