International Journal of Research (IJR)
p-ISSN: 2348-6848
e-ISSN: 2348-795X
Vol. 8 Issue 6
June 2021

Big Data Analytics: Predicting Academic Course Preference Using HADOOP Inspired Map Reduce

K. Manasa[1], Md. Asim[2]
[1]P.G. Scholar, [2]Guide, Assistant Professor
[1,2] Department : CSE
[1,2] Dr. K.V.Subba Reddy College Of Engineering For Women
Email: [1]k.manasa538@gmail.com, [2] shaikh.mdasim@gmail.com

ABSTRACT

With the rapid development of the new technologies, new academic courses introduced to educational system which results in large data which is unregulated data and it is also challenging for the students to prefer those courses in order to increase their career prospects and another challenge is to convert the unregulated data into structured and meaningful information there is a need of Data Mining Tools. Hadoop Distributed File System is used to hold large amount of data and these files are stored in redundant fashion across multiple machines. The process of extracting information is more complex and it is difficult to handle at a shorter duration, this is because the data is unstructured. To handle large amounts of data, the data mining systems uses file systems for decision-making. Knowledge extracted using Map Reduce will be helpful in decision making for students to determine courses chosen for industrial trainings. Map Reduce jobs run over Hadoop clusters by splitting the big data into small chunks and process the data by running it parallel on distributed clusters. The current work believes that only large volumes of data can be evaluated for the efficiency of HDFS tools in data handling and extraction, but not for systems where data is minimal, just like our project where the information is too small for the students to choose their courses. It is also observed that HDFS systems are not as effective in batch processing and for real-time applications. In order to overcome these two problems, we proposed to work on UIDAI Aadhaar real-time dataset to perform data analysis using Apache Spark Ecosystems.

Keywords: Data mining, Big Data, Task analysis, File systems, Industrial training, Distributed databases

INTRODUCTION

1.1 Description

Data mining is one of the most popular technologies for extracting useful information. The term Big Data refers to huge amount of datasets that are collected from different sources that may be through mobiles, CCTV, hospital, banking, government sectors, colleges, etc... Big Data is the rising field of data mining. Big Data is a concept used to characterize a data set that is immense in scale and still exponentially increasing over time. In short, such data is so huge and dynamic that it can't be saved or handled effectively by any of the conventional data processing tools. Big Data is mainly used for storage and analysis purpose. Hadoop is one major data system and the solution to unstructured and gigantic data processing problems.

Big Data incorporates social event of data for capacity and investigation reason which oversee tasks like looking, sharing, perception of data, inquiry preparing, updating and keep up

security of data. It manages unstructured data which may incorporate MS Office documents, PDF, Text and so on though organized data might be the social data. Hadoop is one big data approach that refers to problems found with unstructured data treatment. Hadoop is a framework of open source computing that performs parallel processing of cluster programmers. The approach to big data will allow schools, organizations, colleges to get a holistic aspect of the pupils. It helps address concerns about learning habits, deeper comprehension and programmer patterns, and potential collection of courses for students, which helps to build dazzling undergraduate learning experiences.

The expression "Big Data" has as of late been applied to datasets that develop so enormous that they become unbalanced to work with utilizing conventional database the executive's frameworks. They are data sets whose size exceeds the ability of widely available programming methods and capability systems to collect, store, supervise, just as the data is stored in the middle of the lane. Big Data sizes are continuously expanding from a few hundred terabytes (TB) to numerous Peta bytes (PB) of data, as of now.

LITERATURE SURVEY

2.1 Literature Survey

In educational institutions and one of the most important areas of discovery, data mining or machine learning is a very important field of study and plays an invaluable responsibility in order to uncover specific information taken from historical data stored in a large dataset. Data mining for college, i.e. The discipline that uses data mining techniques in the educational setting is Educational Data Mining (EDM). It is a very important field of study that aims to predict valuable knowledge from instructional datasets in order to enhance educational outcomes and to better determine the learning experience of students. Educational Data Mining may be used as the best choice for studying science and as a data mining branch.

When constructing a model of consumer perception, behaviour and research, Educational Data Mining can be helpful. Data Mining or the exploration of knowledge has gained attention in such a way that it has become extremely important because it is very helpful in examining and shortening the divergent method of data type into practical facts. Educational data mining is based on many techniques in data mining, such as k-nearest neighbour, neural networks, decision trees, vector support machines, naive bays, and many more. There are several open source tools, such as WEKA, rapid miner, orange, and kineme, SSDT, developed for data investigation and to gain a comprehensible framework for potential use for easy review of data using data mining techniques.

In [1] authors used Waikato Environment for Knowledge Analysis which is best suited for the analysis of data and to built a model to get predictive outcome. WEKA was used to predict final-year student marks and these were based on two distinct parameters of the dataset. In each sample, there was one common knowledge, i.e. a variety of students in the last four semesters could be taken from one college course.

The authors used Decision Tree Classification Methods to assess student success and used artificial neural networks to construct classifier models. The outcome generated was
Focused on different characteristics to predict the outcome of the students. Study of student vulnerability and power, which can be beneficial in optimising potential results. The authors used the Naive Bayes classification algorithm in this article, which demonstrates the greatest precision relative to other classification algorithms. It is primarily cynosure in selecting the highest priority decision tree algorithm and describing the detailed significance of each one of them.

Researchers also ended with an idea about the better use of data mining strategies in the student's prophecy prediction and provided a good understanding that the two primary methods that researchers highly suggest for the student's prophecy prediction are data mining prediction algorithms, Decision Tree and Neural Network. In order to classify and analyse future outcomes and variables influencing them, the author applied data mining techniques and also addressed the k-Nearest Neighbour (k-NN) algorithm that plays an effective role in the classifier's accuracy.

PROPOSED WORK

3.1 Proposed work

In the HDFS for map reduction, the input dataset obtained from students is processed. The job is divided by Hadoop into Map and Minimize Tasks. The Map Reduce software translates input data element lists into a set of output data elements and Map and Reduce can use it twice. The major problem of using map reduce is that it is used only for batch processing and time consuming. So, in order to overcome this issue here we used new technology called spark. Spark is a technology which is used for real-time processing and batch processing too. Instead of Map Reduce, we use spark here since it is a polyglot and has its own Mlib and SQL, etc. Aadhaar datasets can be analyzed via spark to find the number of cards produced and rejected in India. We will evaluate the dataset here in either standalone mode or pseudo mode. Once the data set is convenient, we first look at the SQL spark queries to count the number of aadhaar cards produced and denied, followed by state wise or district wise. Our next step is to visualize the data by using tableau where tableau is one of the famous and powerful tools in BI tools.

3.2 Modules

Admin Module

- PATH settings
- Administrator as winutils to give permission
- “spark-shell” command Running the aadhaar datasets program on spark-shell
- Running the aadhaar datasets program on spark-shell
- Aadhaar Output

User Module

- Starting HDFS
- Retrieving data from HDFS
- Entering into spark-shell
- Creating Case Class
- Data frame from RDD
- Output of district Wise Average Rejected Records

3.3 Implementation of Proposed System

Creating Spark Context

To execute any operation in spark, you must first create object of Spark Context class. A Spark Context class represents the connection to our existing Spark cluster and provides the entry point for interacting with Spark. We need to create a Spark Context instance so that we can interact with Spark and distribute our jobs. If you are executing spark commands from spark-shell like this, you can see Spark Context is already created for you by just typing “sc” on console.

```
scala> sc
res0: org.apache.spark.SparkContext = org.apache.spark.SparkContext@170272a8
```

Fig 3.1 “sc” command in scala

The following are the packages required for creating RDDs and executing SQL queries on data frames.

importorg.apache.spark.{SparkConf, SparkContext
importorg.apache.spark.sql.SQLContext
importorg.apache.spark.sql.functions

Now, the next step is to create the case class called Record which is going to infer the schema and used for modeling the immutable data.

case class Record(state: String, district: String,generated: Int,rejected: Int)

Next, we are going to save all the input files as RDD by using val keyword means immutable, sc.textfile is the scala API in that we are going to define the file path of csv files. If we are including one file we define as follows

val records = sc.textFile("aadhaar.csv")

If we are including multiple files from a directory we defined as follows

//val records = sc.textFile("D://aadhaar*.csv")

We are creating RDD called actualrecords and performing the transformations called filter.mapi, here we are excluding the header and splitting the lines.

valactualrecords = records.filter(row => row !=header).map(line=>line.split(",").map(_.trim))

whatever the data we are going to convert the RDD as dataframe on which we can perform SQL queries and assigning the name as Aadhaar records.

Valdf=actualrecords.map(record=>Record(record(0), record(1),record(2).toInt,record(3).toInt)).toDF() df.createTempView("aadhaarrecords")

Next we are going to display all the data stored in the table by using the following command. we can also get the count of all generated and rejected cards by using the following command

spark.sqlContext.sql("SELECT*FROMaadhaarrecords").collect().foreach(print ln)
spark.sqlContext.sql
("SELECT COUNT(*) FROM aadhaarrecords").collect().foreach(println)

In the following steps we will get all the generated and rejected records followed by states

Valstate GenRecords=spark.sqlContext.sql("SELECT state,SUM(generated)FROM aadhaarrecords Group By state") stateGenRecords.map(row=>row(0)).collect
valstateRejRecords = spark.sqlContext.sql("SELECT state,SUM(rejected) FROM aadhaarrecords Group By state") stateRejRecords.map(row=>row(0)).collect

By using he following queries we are going to get the maximum rejected record list and maximum succeed record list .

valstateWiseRecords = spark.sqlContext.sql("SELECT state,SUM(generated + rejected) FROM aadhaarrecords Group By state") stateWiseRecords.collect
valstateWiseMaxRecords = spark.sqlContext.sql("SELECT state,SUM(generated + rejected) As Total FROM aadhaarrecords Group By state ORDER BY Total DESC LIMIT 3") stateWiseMaxRecords.collectvaldistrictWiseAvgRejectedRecords=spark.sqlContext.sql(" SELECT district,AVG(rejected) As Average FROM aadhaarrecords Group By district ORDER BY Average DESC LIMIT 3")
districtWiseAvgRejectedRecords.collect

3.4 Implementation procedure

Implementation of aadhaar card can be done two ways i.e…..

1. Standalone mode
2. Pseudo mode

3.4.1 Execution of Aadhar Datasets on Windows (standalone mode)

Where as in standalone mode that is on windows we are going to use the local file system for storing the datasets, at first, we will create an RDD by loading the data from local file system to spark by giving location of aadhaar data sets. Next we will perform the

transformations, actions like map and filter operations and first, collect and then the next step is to convert a given RDD in to Data Frame (table view) on which we can pass queries what we want to find like total no of rejected records and accepted records, we will find the top three districts and bottom three districts where there is a maximum success and rejection ratio.

Step 1

First, check the java version and winutils folder where it points to hadoop binaries and check whether all the environment variables are properly set or not as shown in the Fig 3.2 (JAVA_PATH, HADOOP_PATH, SPARK_PATH).

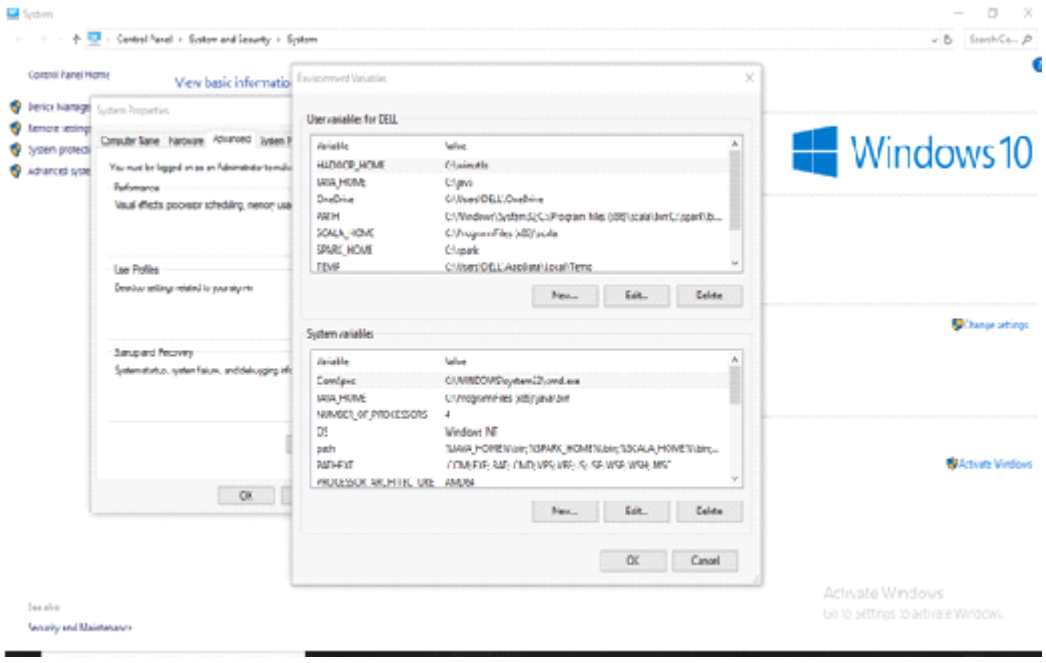

Fig: 3.2 PATH Settings

Step 2

Now open the command prompt as administrator and move to the winutils directory and give the permissions (read, write, execute permissions) to hive directory as shown in the Fig 3.3

```
C:\winutils\bin>winutils.exe chmod -R 777 C:\tmp\hive

C:\winutils\bin>winutils.exe ls -F \tmp\hive
drwxrwxrwx|1|DESKTOP-6H1JBD3\DELL|DESKTOP-6H1JBD3\None|0|Feb|12|2018|\tmp\hive

C:\winutils\bin>
```

Fig: 3.3 Administrator as winutils to give permission

Step 3

Next move into to the spark directory and type as spark-shell where spark-shell is going to get started and we will directly enter into the scala prompt as shown in the Fig 3.4

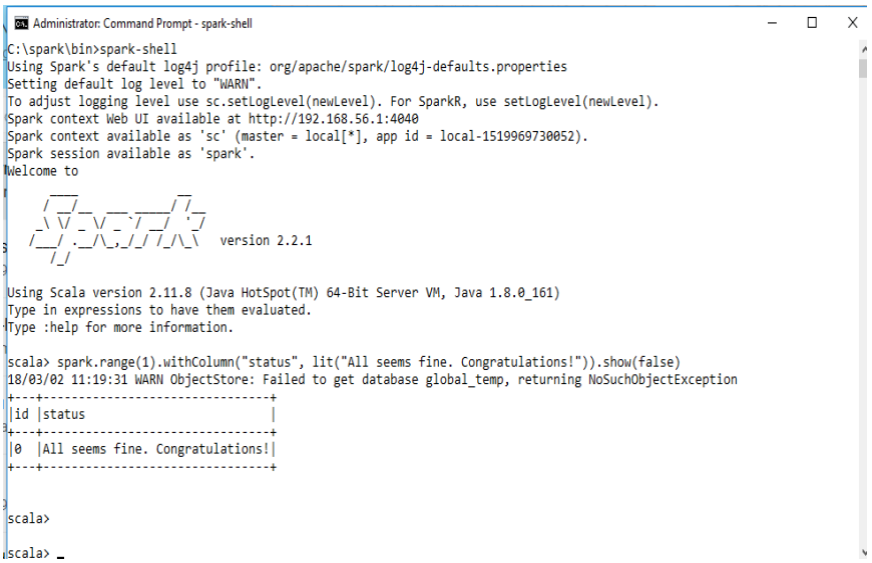

Fig: 3.4 "spark-shell" command

Step 4
Then run the above-mentioned program as follows in spark shell

```
scala> val header = records.first()
header: String = State,District,Aadhaar generated,Enrolment Rejected

scala> val actualrecords = records.filter(row => row != header).map(line => line.split(",").map(_.trim))
actualrecords: org.apache.spark.rdd.RDD[Array[String]] = MapPartitionsRDD[3] at map at <console>:33

scala> val df = actualrecords.map(record => Record(record(0), record(1),record(2).toInt,record(3).toInt)).toDF()
18/02/12 18:05:51 WARN ObjectStore: Failed to get database global_temp, returning NoSuchObjectException
df: org.apache.spark.sql.DataFrame = [state: string, district: string ... 2 more fields]

scala> val df = actualrecords.map(record => Record(record(0), record(1),record(2).toInt,record(3).toInt)).toDF()
df: org.apache.spark.sql.DataFrame = [state: string, district: string ... 2 more fields]

scala> df.registerTempTable("aadhaarrecords")
warning: there was one deprecation warning; re-run with -deprecation for details

scala> :warnings
<console>:40: warning: method registerTempTable in class Dataset is deprecated: Use createOrReplaceTempView(viewName) instead.
      df.registerTempTable("aadhaarrecords")
         ^
```

Fig: 3.5 running the Aadhaar datasets program on spark-shell (1)

Step 5
By performing the queries on data frames, Fig 3.7 shows the output of state wise maximum records (success, rejected records)

```
scala> val stateGenRecords = spark.sqlContext.sql("SELECT state,SUM(generated) FROM aadhaarrecords Group By state" )
stateGenRecords: org.apache.spark.sql.DataFrame = [state: string, sum(generated): bigint]

scala> stateGenRecords.collect
res7: Array[org.apache.spark.sql.Row] = Array([Nagaland,6925], [Karnataka,102822], [Odisha,154865], [Kerala,14008], [Tamil Nadu,268130], [Chhattisgarh,27370], [Andhra

scala> val stateRejRecords = spark.sqlContext.sql("SELECT state,SUM(rejected) FROM aadhaarrecords Group By state" )
stateRejRecords: org.apache.spark.sql.DataFrame = [state: string, sum(rejected): bigint]

scala> stateRejRecords.collect
res9: Array[org.apache.spark.sql.Row] = Array([Nagaland,379], [Karnataka,24472], [Odisha,36341], [Kerala,1301], [Tamil Nadu,52396], [Chhattisgarh,5890], [Andhra Prades

scala> val stateWiseMaxRecords = spark.sqlContext.sql("SELECT state,SUM(generated + rejected) As Total FROM aadhaarrecords Group By state ORDER BY Total DESC LIMIT 3")
stateWiseMaxRecords: org.apache.spark.sql.DataFrame = [state: string, Total: bigint]

scala> stateWiseMaxRecords.collect
res13: Array[org.apache.spark.sql.Row] = Array([Uttar Pradesh,771415], [Bihar,747988], [Gujarat,404538])

scala> districtWiseAvgRejectedRecords.collect
res15: Array[org.apache.spark.sql.Row] = Array([West Champaran,915.4285714285714], [Sitapur,904.5714285714286], [Barddhaman,837.7142857142857])
```

Fig: 3.6 running the aadhaar datasets program on spark-shell (2)

```
scala> spark.sqlContext.sql("SELECT * FROM aadhaarrecords").collect().foreach(println)
[Andaman and Nicobar Islands,South Andaman,2,0]
[Andhra Pradesh,Ananthapuramu,239,27]
[Andhra Pradesh,Chittoor,394,34]
[Andhra Pradesh,Cuddapah,189,13]
[Andhra Pradesh,East Godavari,290,20]
[Andhra Pradesh,Guntur,386,16]
[Andhra Pradesh,Krishna,279,25]
[Andhra Pradesh,Kurnool,249,6]
..
..
..
..
[Uttar Pradesh,Amroha,575,82]
[Uttar Pradesh,Auraiya,466,55]
[Uttar Pradesh,Azamgarh,3467,121]
[Uttar Pradesh,Baghpat,581,46]
[Uttar Pradesh,Bahraich,2684,239]
[Uttar Pradesh,Ballia,1727,135]
[Uttar Pradesh,Balrampur,1726,141]
```

Fig: 3.7 Aadhaar Output

3.4.2 Execution of Aadhar Datasets on Ubuntu (pseudo mode)

Where as in pseudo mode (Ubuntu), the single machine is going to act as name node and data node, name node is nothing but the node where we can store the addresses of all the data that are stored in data nodes, and data nodes are the one where the data is going to reside.

In this mode, we are going to run spark on top of HDFS (Hadoop distributed file system). The first step is to start all hadoop daemons. In the second step create a directory in HDFS and load the data from local file system in to the respective directory of HDFS. After these start the spark shell and import the packages which are needed then create an RDD by loading the specific location where the data sets stored in HDFS file system on which we can perform

transformations and actions and converting the RDD in to Data Frame, passing Spark SQL queries which is similar to what we have done in standalone mode.

Step 1

Initially start the hadoop dfs daemons, the Data node and the name node by using the command as shown in Fig 3.8 (start-dfs.sh)

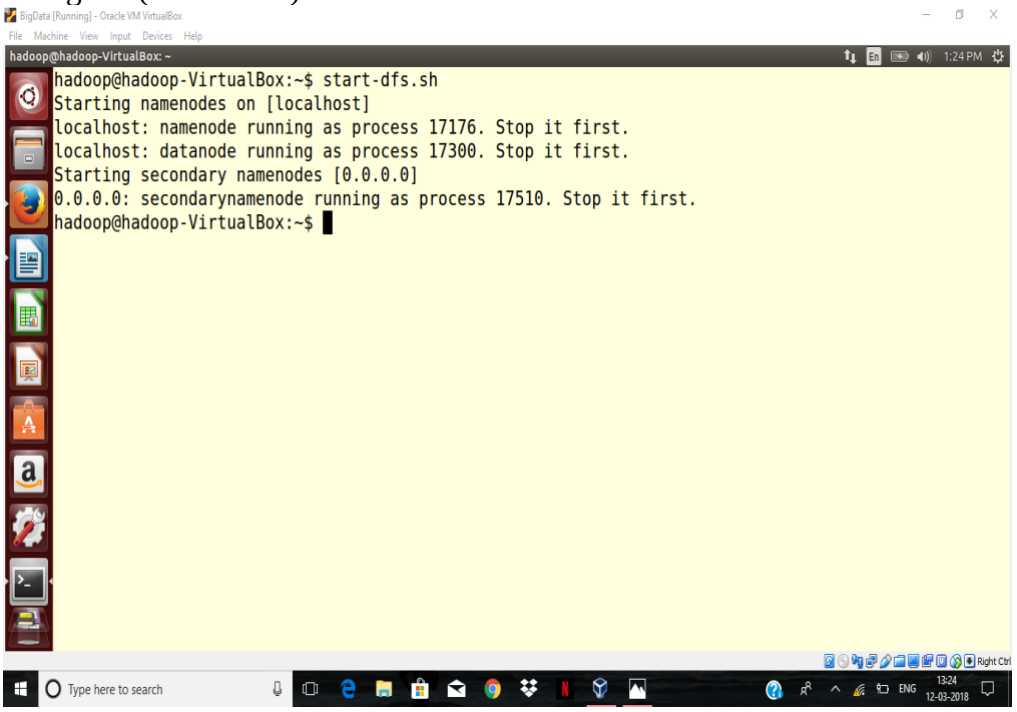

Fig: 3.8 Starting HDFS

Step 2

Now load tha data into the hdfs from local system because for spark we are using the hdfs filesystem, i.e. spark is running on the hdfs.

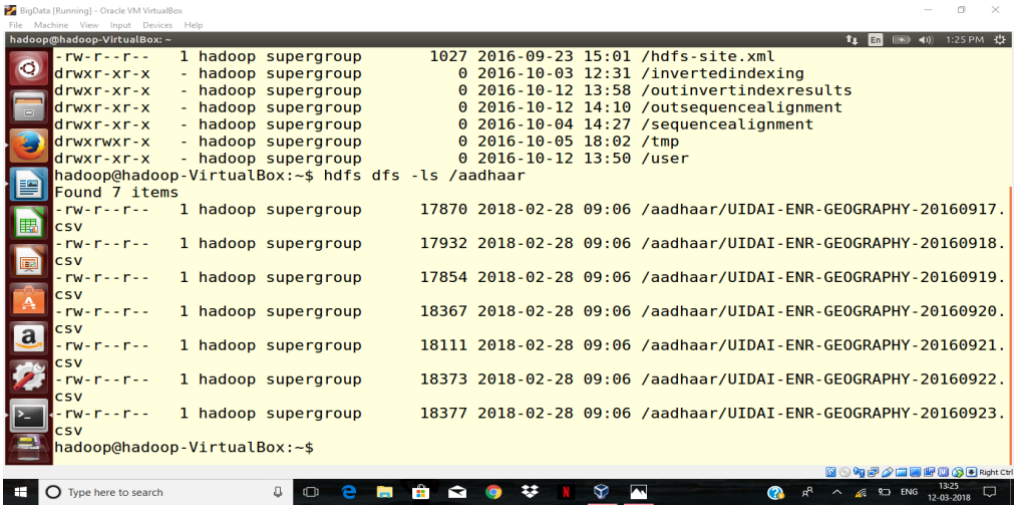

Fig: 3.9

Retrieving data from HDFS

Step 3 Now start the spark-shell in separate shell by using the spark-shell command, automatically you will prompt with the scala command

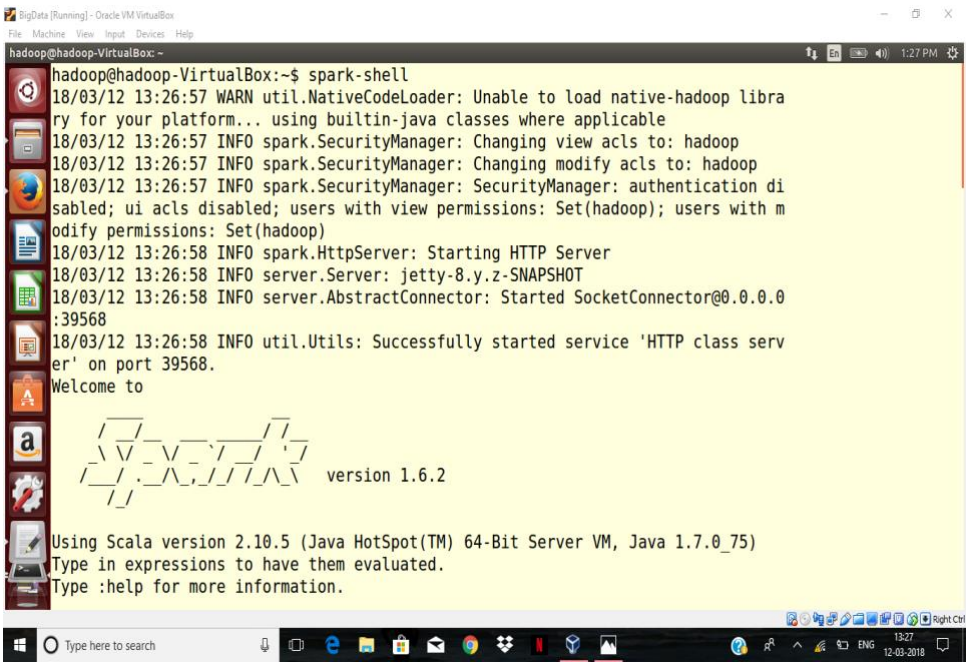

Fig: 3.10 entering into spark-shell

Step 4 Let's implement the queries on aadhaar datasets by loading the data from HDFS spark as shown in Fig 3.11

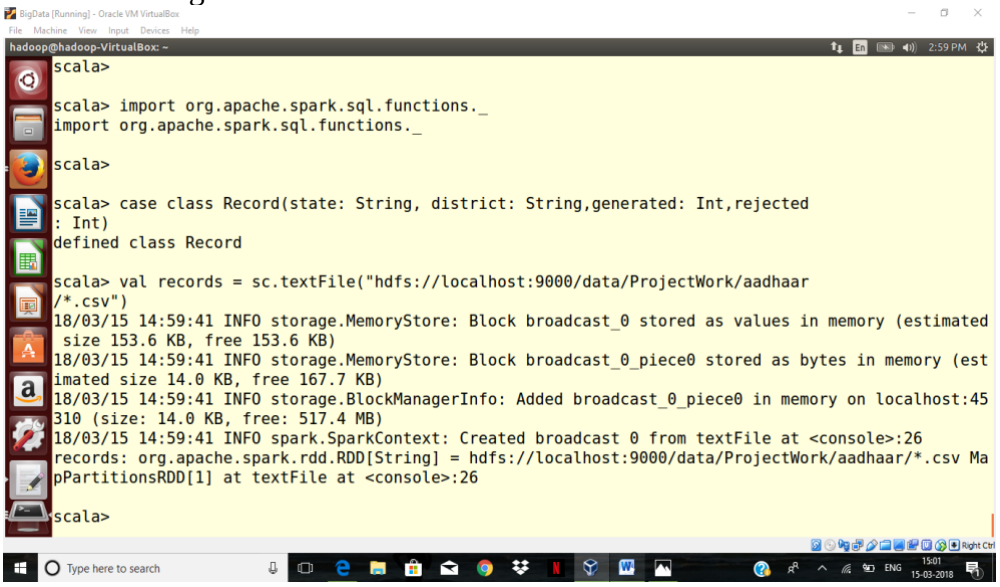

Fig: 3.11 Creating Case Class

Step 5

Fig 3.12 showing data stored in data frame after converting the RDD into data frame

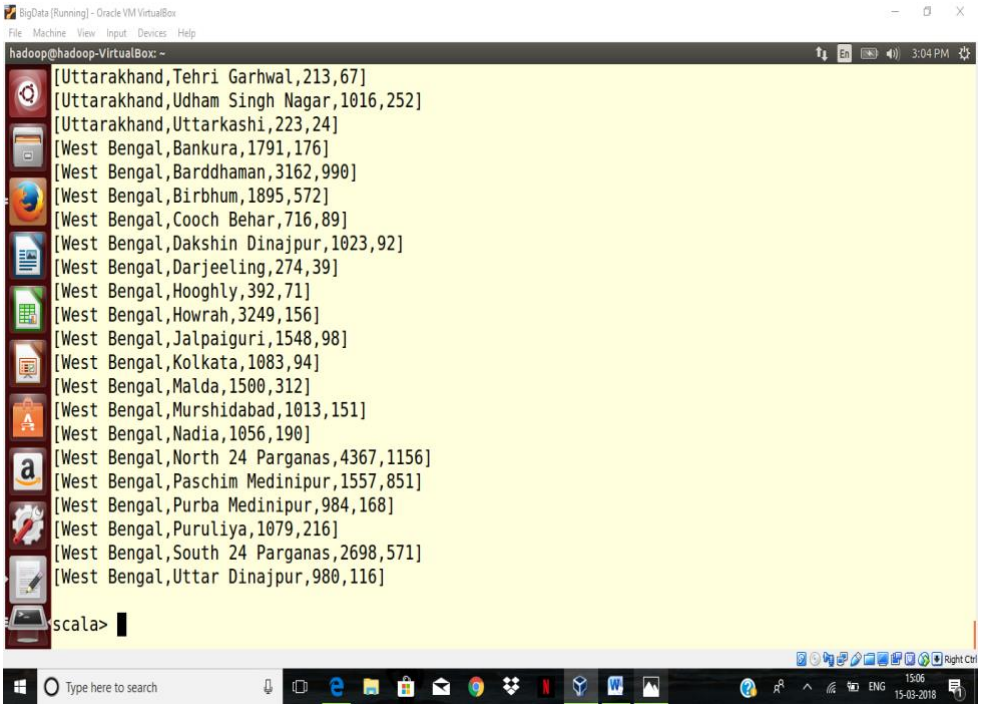

Fig: 3.12 Data frame from RDD

RESULT ANALYSIS OF PROPOSED WORK

4.1 Output Visualization

The next step is to visualize the data after extracting the results from data sets by using tableau where tableau is one of the popular and strong tools in BI tools. In the form of graphs and maps, users can create and circulate an immersive and collaborative dashboard that illustrates the patterns, variations, and density of the data. Tableau can connect to files, relational and Big Data sources to acquire and process data. The software allows data blending and real-time collaboration, which makes it very unique. It is used by businesses, academic researchers, and many government organizations for visual data analysis. In this project we will load the output data which we got by passing Spark SQL queries on aadhaar datasets, after loading the data into tableau go to the work sheet where we can create various visualizations by drag and dropping the measures and dependencies in to columns and rows.

Step1

Open Tableau and click on text file from there we can load CSV/JSON/Microsoft Excel/Text Files etc like shown in Fig 4.2.

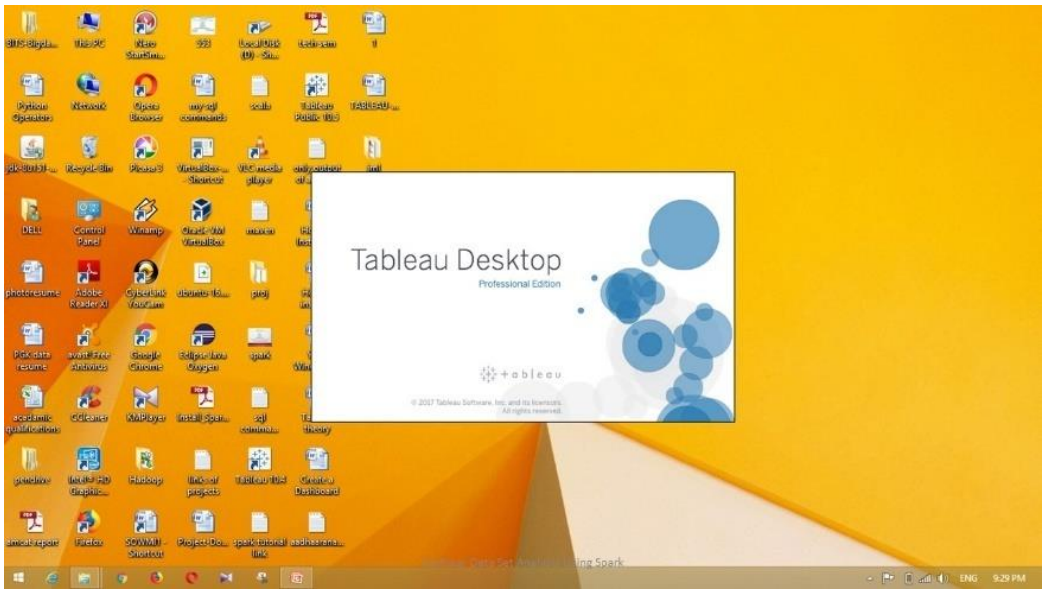

Fig 4.1 Launching of the Tableau Desktop application

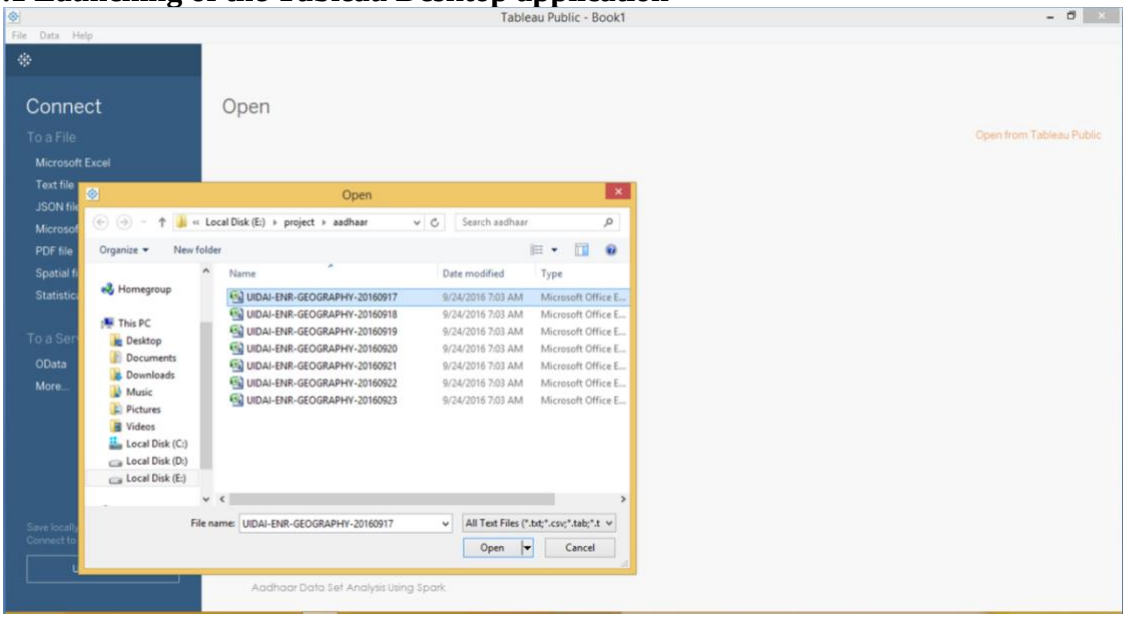

Fig: 4.2 importing ".csv" file

Step 2

After that load the aadhaar dataset output. After loading the datasets, it will be shown as Fig 4.3

International Journal of Research (IJR)
p-ISSN: 2348-6848
e-ISSN: 2348-795X
Vol. 8 Issue 6
June 2021

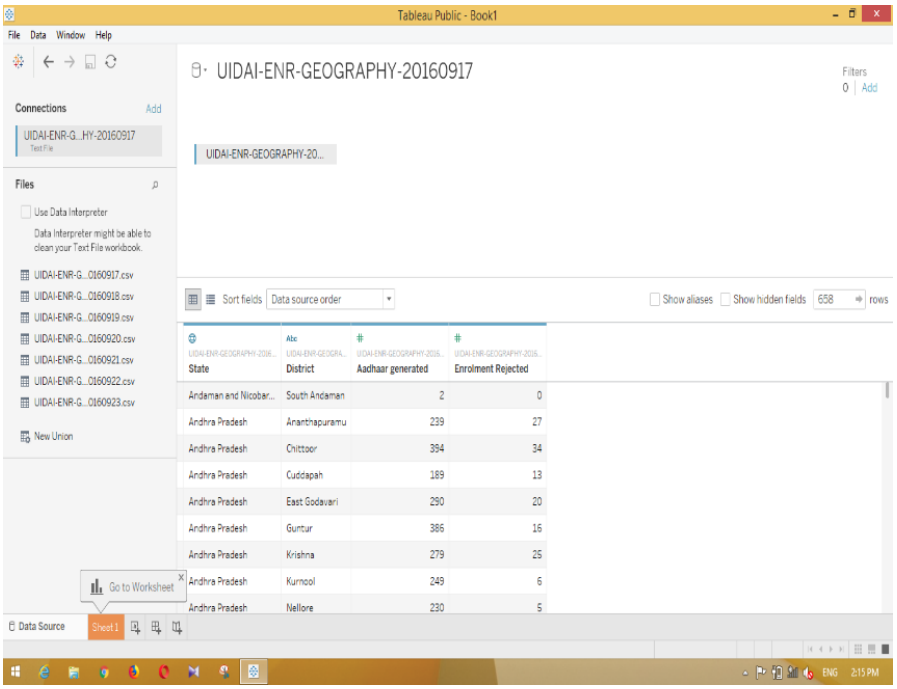

Fig: 4.3 Data loaded to Tableau

Step 3 Open the Dashboard by clicking on the sheet1 as shown in the Fig 4.4 where we can visualize the data.

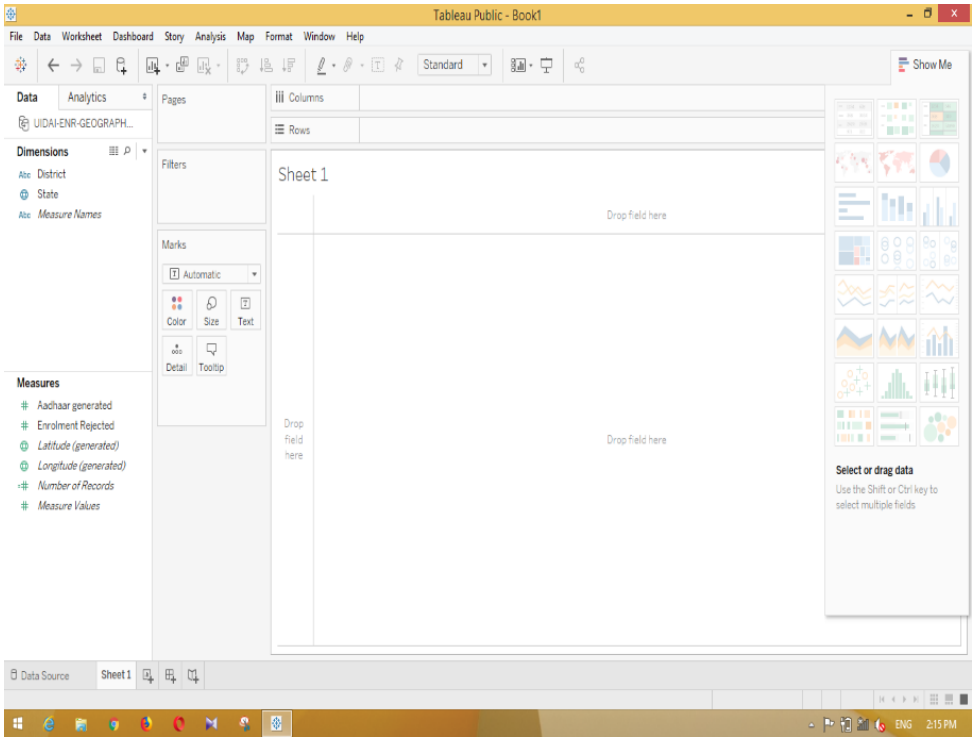

Fig: 4.4 Dash board

Step 4 Next drag and drop the attributes in their specified columns and rows, then visualize the data in different formats as shown in the Fig 4.5

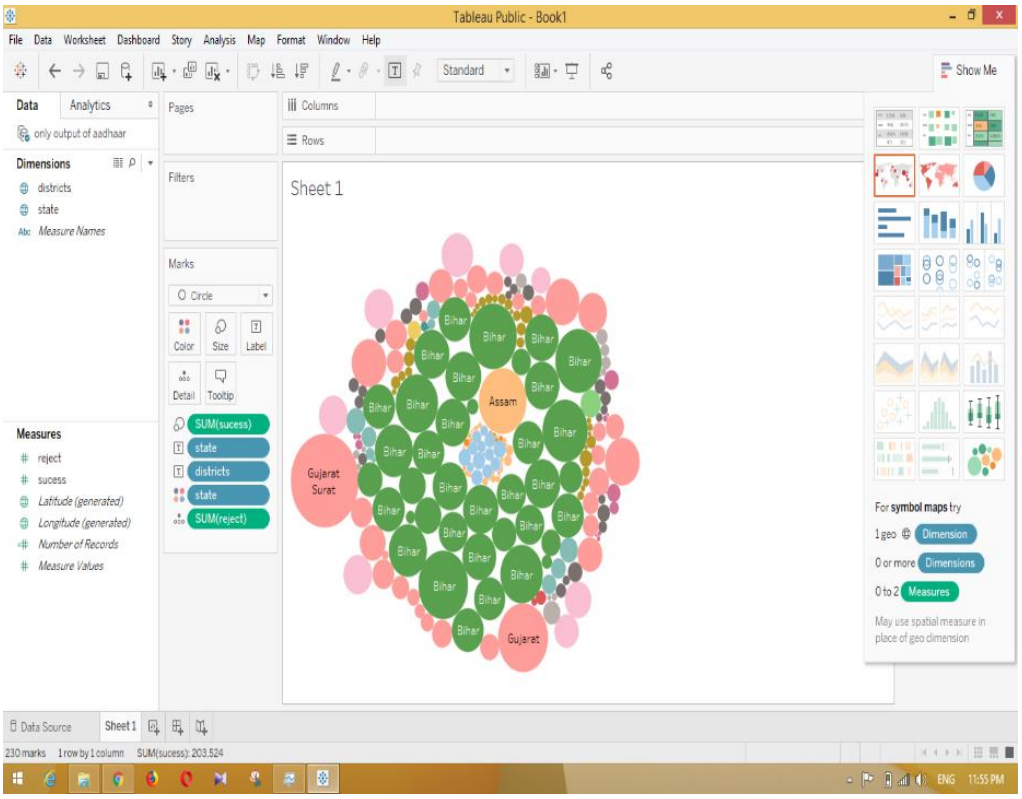

Fig: 4.5 Packed Bubbles Map view

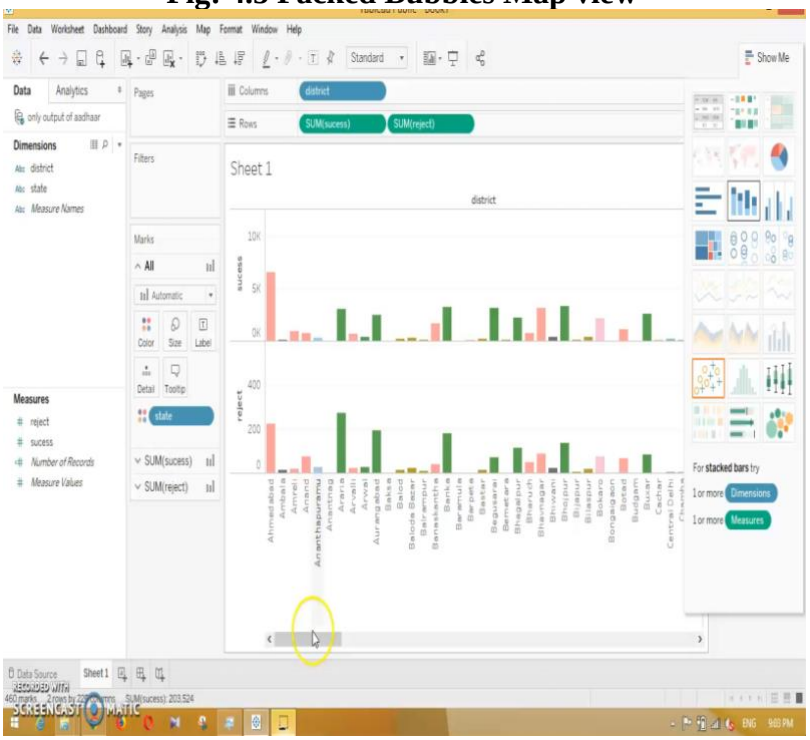

Fig: 4.6 Stacked Bars Map View

In Fig 4.6 we can see the number of success and rejected aadhaar cards. Here X-axis represents names of the states and Y-axis represents success or rejected of aadhaar cards. Instant of displaying complete number of success and rejected cards in a single line, here we have divided into two in which the top graph represents the number of aadhaar cards which are successful and bottom graph represents the number of rejected cards in a state-wise.

In Fig 4.7 we have shown the represents of every state and district wise number of success and rejected of aadhaar cards in the form of boxes. In bar map view we can't see the complete details of cards but when coming to tree map view we can see the complete details of success and rejected of aadhaar cards in a box format. When we place cursor on the box we can see the name of the state, district of that state, number of aadhaar cards and number of rejected cards.

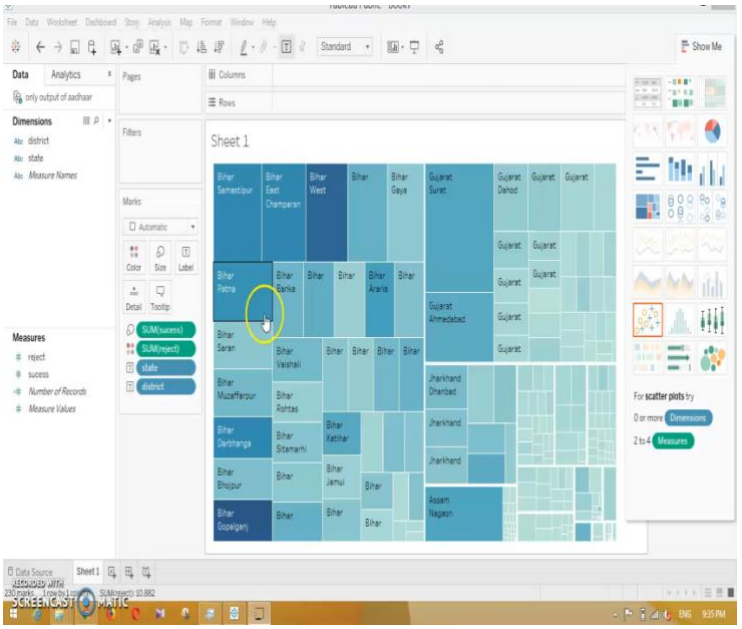

Fig: 4.7 Tree Map View

CONCLUSION AND FUTURE WORK

5.1 Conclusion

The aadhaar data collection is evaluated for different queries. The Adobe Spark Eco system is used to store, process, analyzes and extracts the necessary aadhaar data. This work highlights the contradictions and fluctuations in enrolment in populations, timelines, age, and reactions of state governments and state people. Data analysis can be used for very specific purposes that can implement thousands of read or write per second and can be used directly from the enrollment desk to upload on to the HDFS storage for data enrollment. Semi-structured or unstructured data can be preserved. As part of potential project work, the proposed work could be further expanded to include analyze or handle aadhaar dataset for fraud, mistake and data duplication.

5.2 Future work

In this project, we evaluated the data set against multiple queries using the Hadoop environment for large data processing and storage. The government must make people more conscious of the advantages of the aadhaar card. Some people who are illiteracy they don't know how to save the aadhaar number in a digital way, so that the reason we came with a new idea like displaying the Aadhaar Number using the registered phone number. Here it is not mandatory for those people to remember 12 digits number of aadhaar which is rarely used, instead of that they can use their phone number which is commonly used in daily life for knowing or displaying the aadhaar number.

REFERENCES

[1] Chitra Jalota, Rashmi Agrawal," Analysis of Educational Data Mining using Classification" 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing.

[2] Wei Zhang, Shiming Q (2018), "A Brief Review of the Core Technologies and Applications of Online Learning System Educational Data Mining," IEEE 3rd International Conference on Big Data Analysis.

[3] B. Manjulatha, Ambica Venna, K.Soumya,"Implementation of Hadoop Operations for Big Data Processing in Educational Institutions", International Journal of Innovative Research in Computer and Communication Engineering,ISSN(Online) : 2320-9801,Vol. 4, Issue 4, April 2016.

[4] N. Ankita, R. Anjali, "Analysis of Student Performance Using Data Mining Technique", International Journal of Innovative Research in Computer and Communication Engineering, Vol. 5, Issue 1, January 2017.

[5] P. Shruthi, B. Chaitra, "Student Performance Prediction in Education Sector Using Data Mining", International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 6, Issue 3, March 2016.

[6] K. Kohli and S. Birla, "Data Mining on Student Database to Improve Future Performance", International Journal of Computer Applications, Vol.146 No.15, pp. 0975 – 8887, July 2016.

[7] Udeni Jayasinghe, Anuja Dharmaratne, Ajantha Atukorale, "Students' Performance Evaluation in Online Education System Vs Traditional Education System", IEEE 2015 12th International Conference on Remote Engineering and Virtual Instrumentation (REV).

[8] Rifki Sadikin, Andria Arisal, Rofithah Omar, Nur Hidayah Mazni, " Processing Next Generation Sequencing Data in Map- Reduce Framework using Hadoop-BAM in a Computer Cluster" 2017 2nd International Conferences on Information Technology, Information Systems and Electrical Engineering (ICITISEE)

[9] Katrina Sin,Loganathan Muthu,"Application of Big Data in Education Data Mining and Learning Analytics – A Literature Review" , ICTACT Journal on Soft Computing,ISSN: 2229-6956 (online), Vol 5, Issue 4,July 2015.

[10] Peter Brusilovsky, Sibel Somyürek, Julio Guerra, Roya Hosseini, Vladimir Zadorozhny , Paula J. Durlach,"Open Social Student Modeling for Personalized Learning," IEEE Transactions on Emerging Topics in Computing, Volume: 4, Issue: 3, July-Sept. 2016.

[11] Bhushan Jadhav, Archana B. Patankar, Sonali B. Jadhav, "A Practical approach for integrating Big data Analytics into E-governance using hadoop", Proceedings of the 2nd International Conference on Inventive Communication and Computational Technologies (ICICCT 2018)

[12] Suyash Mishra, Dr Anuranjan Misra, "Structured and Unstructured Big Data Analytics", International Conference on Current Trends in Computer, Electrical, Electronics and Communication (ICCTCEEC-2017)

[13] Jisha S Manjaly, Dr.T.Subbulakshmi, "Various approches to improve MapReduce performance in Hadoop", Proceedings of the International Conference on Inventive Computation Technologies (ICICT-2018)

[14] Nandita Yambem, Nandakumar A N, "AMPO: Algorithm for MapReduce Performance Optimization for Enhancing Big Data Analytics", 2017 International Conference on Electrical, Electronics, Communication, Computer and Optimization Techniques (ICEECCOT)

[15] Payal M. Bante, Dr. K. Rajeswari, "Big Data Analytics using Hadoop Map Reduce

Framework and Data Migration Process", **2017 International Conference on Computing, Communication, Control and Automation (ICCUBEA)**

[16] BhartiThakur,ManishMann, "Data Mining for Big Data: AReview", International Journal of Advanced Research in Computer Science and Software Engineering,Vol. 4,May 2015J.