

School of Information Technology & Engineering



BIG DATA ANALYTICS

SWE2011

A PROJECT REPORT

ANALYSIS BASED ON WEB LOG FILES

by

BANDARU S S PRIYANKA (15MIS0424)

Under the guidance of

Nirmala M

TABLE OF COMPONENTS-

S.No	Components	Page No
1.	Problem Statement	4
2.	Approach	5
3.	Dataset	5
4.	Modules	6
5.	Proposed Implementation Framework(Diagram)	7
6.	Implementation Status	10
7.	Conclusion	28
8.	References	29

INTRODUCTION

A significant development in the field of technology in sectors such as business, public and private has been observed leading to accumulation of large data over the web. Information acquired from the web are used to describe the exponential growth and availability of data, both structured and unstructured. As data over the web is heterogeneous in nature, analyzing such data is necessary in order to gain acquaintance wherein log file analysis is an effective solution. Log files are the files that list the actions that have been occurred and reside in web server. There prevails a need to process and store log files using traditional techniques however in the enterprise scenario the data from these log files is outsized due to which processing capacity of conventional approaches becomes incompetent for gaining information for processing.

PROBLEM STATEMENT

A significant development in the field of technology in sectors such as business, public and private has been observed leading to accumulation of large data over the web. Information acquired from the web are used to describe the exponential growth and availability of data, both structured and unstructured. As data over the web is heterogeneous in nature, analysing such data is necessary in order to gain acquaintance wherein log file analysis is an effective solution. Log files are the files that list the actions that have been occurred and reside in web server. There prevails a need to process and store log files using traditional techniques however in the enterprise scenario the data from these log files is outsized due to which processing capacity of conventional approaches becomes incompetent for gaining information for processing.

So, we have the dataset of the weblogs that contains the attributes -

- ☐ ip
- ☐ date
- ☐ time
- ☐ cik
- ☐ accession
- ☐ extension
- ☐ size
- ☐ norefer
- ☐ browser
- ☐ find
- ☐ code

APPROACH

For the implementation of our problem statement, we have taken a set of about 500 transactions of web logs that give us data about the IP address, date, time, access file, its size, browser type, etc. The data taken has been taken for a single day itself as the data was too huge to be processed properly. The data contained missing and incorrect values. These transactions were ignored to give a correct result. After this pre-processing stage, the data was analyzed using the WEKA tool. We have used classification technique in our project.

This technique allows much better and accurate results along with a better understanding of the situation.

We are applying 3 techniques for implementing our project-

- 1.Data pre processing using weka tool
- 2.Data classification using python IDE and Hadoop
- 3.Data Clustering using pythonIDE and Hadoop

DATASET

The dataset we have collected from UCI Repository. The data set includes the 10 attributes and around 600 tuples. Firstly the dataset contains the missing value, irrelevant data, redundant data, but then we applied pre-processing techniques to the dataset and the data cleaning is done. so, the final dataset we got is cleaned and accurate. It contain all the details of the weblogs that can be used to know that which ip is more occurring and hence which site is more trending.

The dataset is benchmark as we are applying the python code also, such that we are getting the result that which ip is more trending.

Size of Dataset-The size of dataset is very large as there are 10 attributes and 600 tuples that gives the idea about the variety of inputs.

MODULES

1) Pre-processing Phase

Data pre-processing is an important step in the data mining process. The phrase "garbage in, garbage out" is particularly applicable to data mining and machine learning projects. Data- gathering methods are often loosely controlled, resulting in out-of-range values (e.g., Income:-100), impossible data combinations (e.g., Sex: Male, Pregnant: Yes), missing values, etc.

Pre –Processing techniques are-

a) Data Cleaning-

i) Missing values:

- (1) Ignore the tuple
- (2) Fill in the missing value manually
- (3) Use a global constant to fill in the missing value
- (4) Use the attribute mean to fill in the missing value
- (5) Use the attribute mean for all samples belonging to the same class.
- (6) Use the most probable value to fill in the missing value

ii) Noisy data:

- (1) Binning
- (2) Clustering
- (3) Regression

iii) Inconsistent data

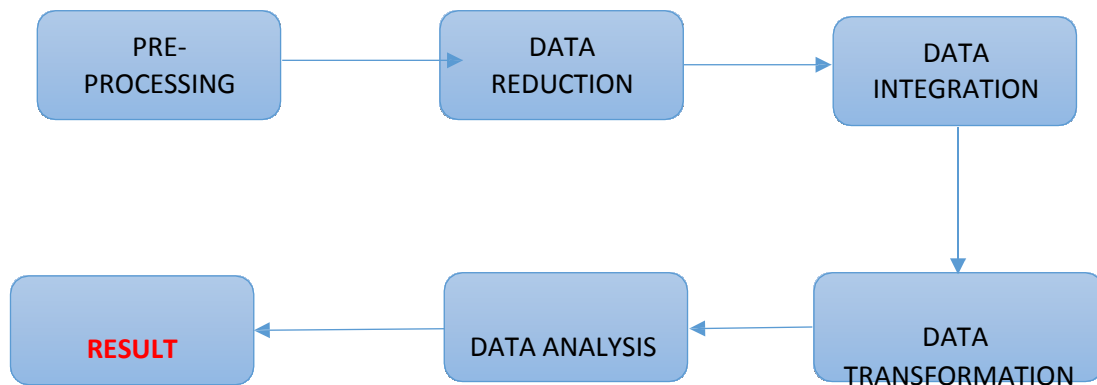
b) Data Integration

- i) Correlation Analysis
- ii) Normalization

- c) Data Transformation
 - i) Smoothing
 - ii) Aggregation
 - iii) Generalization
 - iv) Normalization
 - v) Attribute construction
- d) Data reduction
 - i) Data cube aggregation
 - ii) Attribute subset selection
 - iii) Dimensional reduction
- e) Data Sampling
 - i) Numerosity reduction
 - ii) Discretization and concept hierarchy generation

PROPOSED IMPLEMENTATION FRAMEWORK

□ Diagram



2. Using Weka

We used the Weka tool for preprocessing of the data taken. **Waikato Environment for Knowledge Analysis (Weka)** is a suite of machine learning software written in Java. Weka contains a collection of visualization tools and algorithms for data analysis and predictive modeling, together with graphical user interfaces for easy access to these functions.

The original non-Java version of Weka was a Tcl/Tk front-end to (mostly third-party) modeling algorithms implemented in other programming languages, plus data preprocessing utilities in C, and a Makefile-based system for running machine learning experiments.

3. Using Python-

Python is a widely used high-level programming language for general-purpose programming, created by Guido van Rossum and first released in 1991. An interpreted language, Python has a design philosophy that emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer lines of code than might be used in languages such as C++ or Java. The language provides constructs intended to enable writing clear programs on both a small and large scale. Python features a dynamic type system and automatic memory management and supports multiple programming paradigms, including object-oriented, imperative, functional programming, and procedural styles. It has a large and comprehensive standard library.

4. Using Hadoop-

Apache Hadoop is an open-source software framework used for distributed storage and processing of dataset of big data using the MapReduce programming model. It consists of computer clusters built from commodity hardware. All the modules in Hadoop are

designed with a fundamental assumption that hardware failures are common occurrences and should be automatically handled by the framework.

The core of Apache Hadoop consists of a storage part, known as Hadoop Distributed File System (HDFS), and a processing part which is a MapReduce programming model. Hadoop splits files into large blocks and distributes them across nodes in a cluster. It then transfers packaged code into nodes to process the data in parallel. This approach takes advantage of data locality where nodes manipulate the data they have access to. This allows the dataset to be processed faster and more efficiently than it would be in a more conventional supercomputer architecture that relies on a parallel file system where computation and data are distributed via high-speed networking.

The base Apache Hadoop framework is composed of the following modules:

- a. *Hadoop Common* – contains libraries and utilities needed by other Hadoop modules;
- b. *Hadoop Distributed File System (HDFS)* – a distributed file-system that stores data on commodity machines, providing very high aggregate bandwidth across the cluster;
- c. *Hadoop YARN* – a platform responsible for managing computing resources in clusters and using them for scheduling users' applications; and
- d. *Hadoop MapReduce* – an implementation of the MapReduce programming model for large-scale data processing

IMPLEMENTATION STATUS

1) Data Pre-Processing

- a. **Data Reduction**-The dataset we have contains the redundant data, missing values. So, we are doing the data reduction in this by ignoring the tuples. In this way, we have the reduced and cleaned dataset.
- b. **Data Integration**-For the preprocessing technique, we implemented the_chi-square analysis-Using the weka tool, we did chi-sq. analysis of the dataset that results in the ranking of the attributes.
- c. **Normalization**-Using the weka tool, we normalized the data.
- d. **Data transformation**-Discretization-We did discretization using the weka tool in the dataset.

2) Data Analysis

- a. **Naïve Bayes**-We are taking the dataset that are the weblogs, then we are applying the Naïve Bayes using the weka tool. In this the confusion matrix is also made.
- b. **Decision tree**-Using the weka tool, we are analyzing our dataset as the decision tree we are getting.

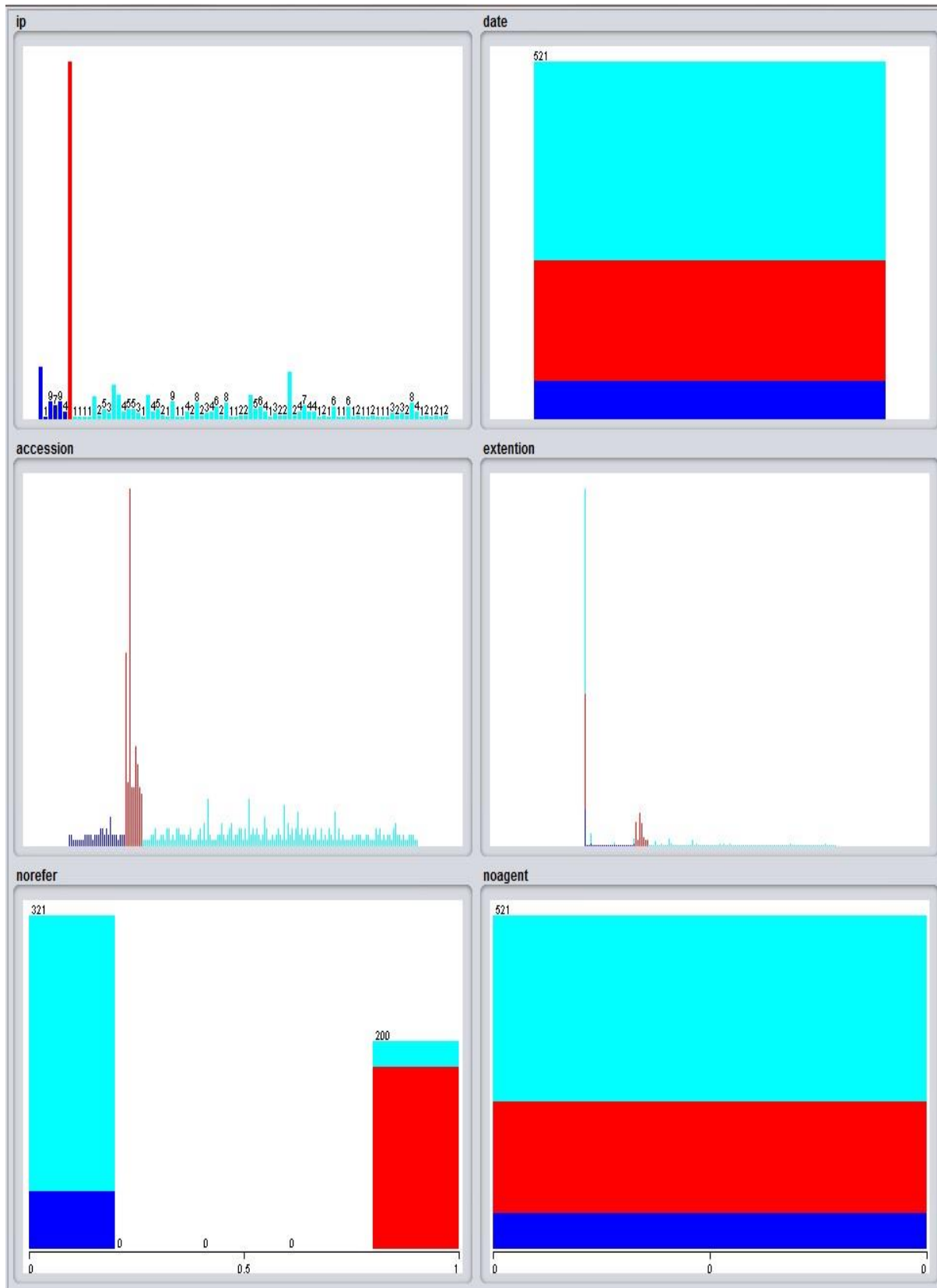
3) Classification-

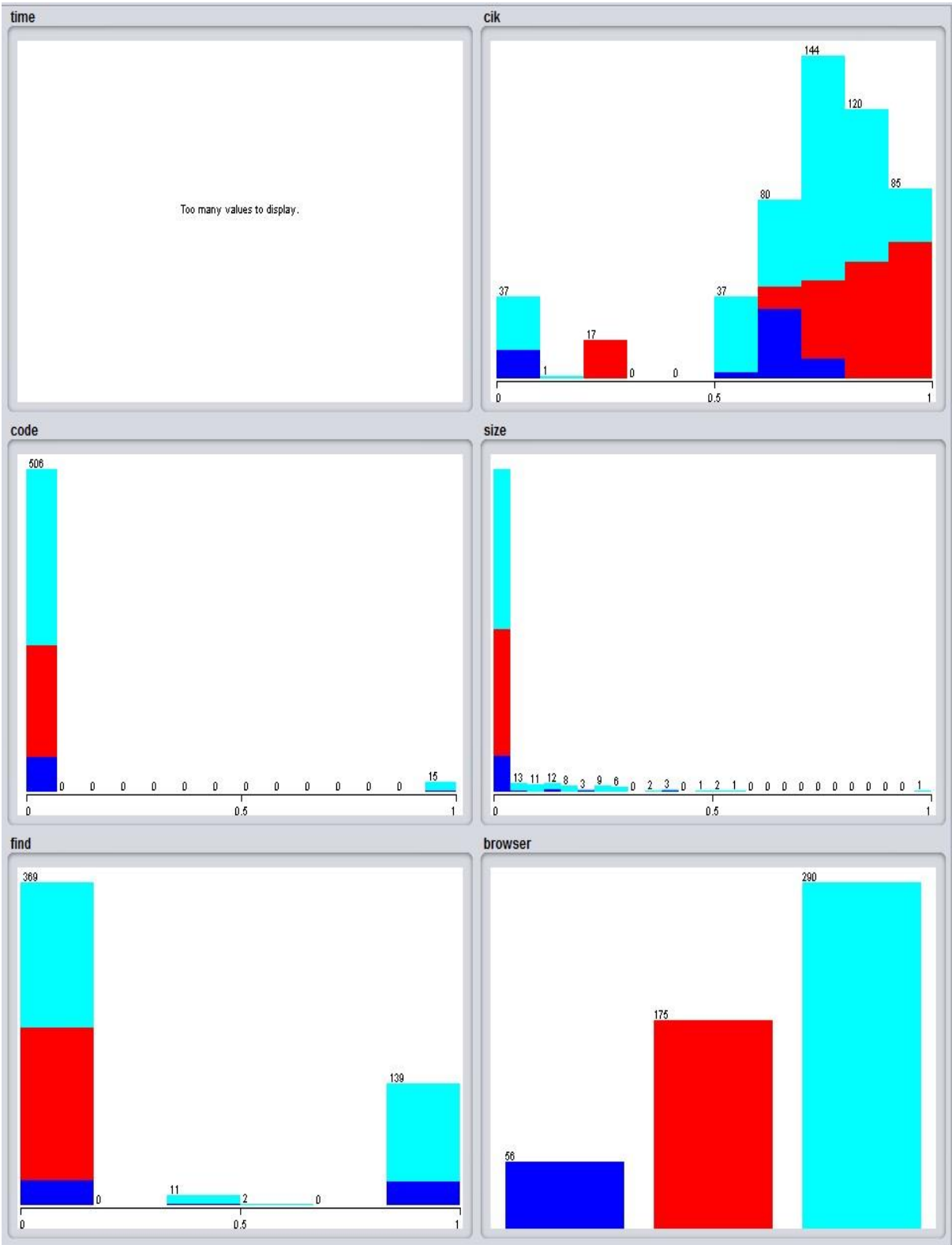
We have done the classification techniques rather than the clustering because-

- a. For the labeled data, the clustering results are quite similar to the cross-validation performance of the semi-supervised learning.
- b. For the unlabeled data, the clustering results are not as good as the further prediction of the trained SVM classifier (using labeled data as aforementioned) according to some qualitative checking (visually check the classified images).

We have implemented our project using two platforms-

1) Weka





- 2) **Python IDE**-The following dataset is analyzed and transferred into python. We have written the python code that includes the mapper code and the reducer code.

□ **Word Count**

Mapper.py-

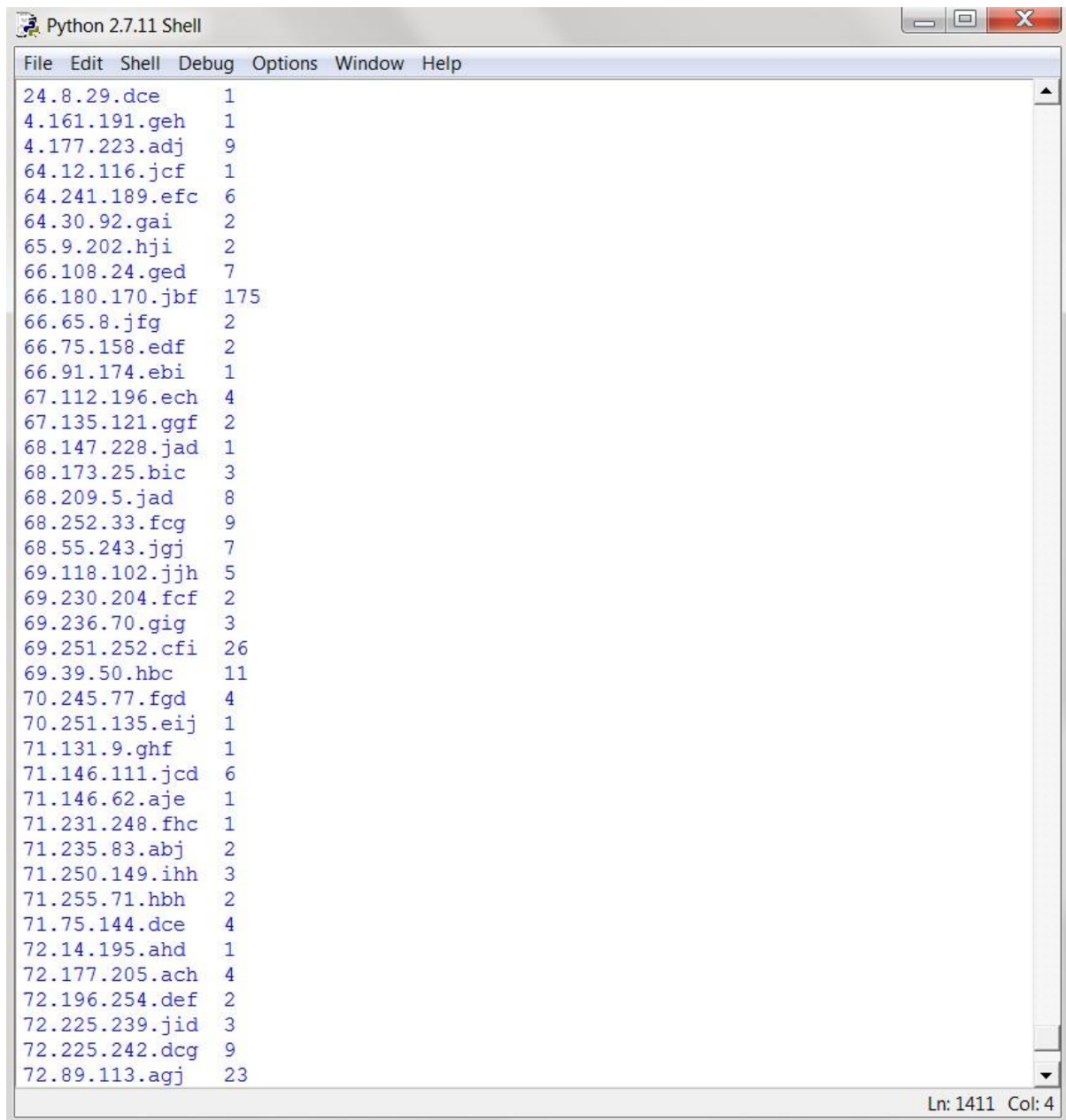
```
import sys, xlrd, csv
book= xlrd.open_workbook("C:\Users\Girisha\Desktop\Code\Book1.xlsx")
sheet = book.sheets()[0]
sheet = book.sheet_by_name("Book1")
sheet = book.sheet_by_index(0)
r = sheet.row(0); c = sheet.col_values(0)
data = []
for i in xrange(sheet.nrows):
    data.append(sheet.row_values(i))
num_cols = sheet.ncols
cr= csv.reader(open("C:\Users\Girisha\Desktop\Code\Book1.csv","rb"))
arr = range(522)
x = 0
for row in cr:
    arr[x] = row[0]
    x += 1
word2count = {}
words = arr
for word in words:
    print '%s\t%s' % (word, 1)
```

Reducer.py-

```
from operator import itemgetter; import sys
word2count = {}
for line in sys.stdin:
    line=line.strip()
    try:
        word, count = line.split()
        count = int(count)
        word2count[word] = word2count.get(word,0)+count
    except ValueError:
        break
sorted_word2count = sorted(word2count.items(),key=itemgetter(0))
for word, count in sorted_word2count:
    print '%s\t%s'% (word,count)
```

Implementation Screenshots-

Python 2.7.11 Shell		Python 2.7.11 Shell	
File	Edit Shell Debug Options Window Help	File	Edit Shell Debug Options V
ip	1	66.180.170.jbf	1
69.251.252.cfi	1	66.180.170.jbf	1
69.251.252.cfi	1	66.180.170.jbf	1
69.251.252.cfi	1	66.180.170.jbf	1
69.251.252.cfi	1	66.180.170.jbf	1
68.147.228.jad	1	66.180.170.jbf	1
68.252.33.fcg	1	66.180.170.jbf	1
68.252.33.fcg	1	71.146.62.aje	1
68.252.33.fcg	1	214.13.209.efb	1
68.252.33.fcg	1	71.131.9.ghf	1
68.252.33.fcg	1	24.8.29.dce	1
68.252.33.fcg	1	69.39.50.hbc	1
68.252.33.fcg	1	152.163.101.jdj	1
68.252.33.fcg	1	76.177.188.jfg	1
68.252.33.fcg	1	72.225.239.jid	1
68.252.33.fcg	1	75.21.81.djf	1
69.251.252.cfi	1	76.177.188.jfg	1
69.251.252.cfi	1	72.225.239.jid	1
66.108.24.ged	1	72.225.239.jid	1
66.108.24.ged	1	75.80.31.gjc	1
66.108.24.ged	1	151.191.175.bbi	1
66.108.24.ged	1	76.177.188.jfg	1
66.108.24.ged	1	75.80.31.gjc	1
66.108.24.ged	1	69.39.50.hbc	1
69.251.252.cfi	1	89.24.6.fdh	1
69.251.252.cfi	1	75.24.199.jde	1
69.251.252.cfi	1	75.24.199.jde	1
69.251.252.cfi	1	89.24.6.fdh	1
69.251.252.cfi	1	24.15.67.acc	1
69.251.252.cfi	1	76.177.188.jfg	1
69.251.252.cfi	1	89.24.6.fdh	1
69.251.252.cfi	1	66.91.174.ebi	1
69.251.252.cfi	1	192.193.221.gge	1
69.251.252.cfi	1	76.177.188.jfg	1
69.251.252.cfi	1	192.193.221.gge	1
69.251.252.cfi	1	75.24.199.jde	1
69.251.252.cfi	1	192.193.221.gge	1
69.251.252.cfi	1	75.24.199.jde	1
69.251.252.cfi	1	75.24.199.jde	1
69.251.252.cfi	1	70.245.77.fgd	1
69.251.252.cfi	1	24.15.67.acc	1
69.251.252.cfi	1	70.245.77.fgd	1
69.251.252.cfi	1	24.15.67.acc	1
69.251.252.cfi	1	89.24.6.fdh	1
69.251.252.cfi	1	69.118.102.jjh	1
4.177.223.adj	1	70.245.77.fgd	1
4.177.223.adj	1	69.118.102.jjh	1
4.177.223.adj	1	89.24.6.fdh	1
4.177.223.adj	1		



The screenshot shows a Python 2.7.11 Shell window with a menu bar (File, Edit, Shell, Debug, Options, Window, Help) and a text area containing a list of IP addresses and their corresponding counts. The list is as follows:

24.8.29.dce	1
4.161.191.geh	1
4.177.223.adj	9
64.12.116.jcf	1
64.241.189.efc	6
64.30.92.gai	2
65.9.202.hji	2
66.108.24.ged	7
66.180.170.jbf	175
66.65.8.jfg	2
66.75.158.edf	2
66.91.174.ebi	1
67.112.196.ech	4
67.135.121.ggf	2
68.147.228.jad	1
68.173.25.bic	3
68.209.5.jad	8
68.252.33.fcq	9
68.55.243.jgj	7
69.118.102.jjh	5
69.230.204.fcf	2
69.236.70.gig	3
69.251.252.cfi	26
69.39.50.hbc	11
70.245.77.fgd	4
70.251.135.eij	1
71.131.9.ghf	1
71.146.111.jcd	6
71.146.62.aje	1
71.231.248.fhc	1
71.235.83.abj	2
71.250.149.ihh	3
71.255.71.hbh	2
71.75.144.dce	4
72.14.195.ahd	1
72.177.205.ach	4
72.196.254.def	2
72.225.239.jid	3
72.225.242.dcg	9
72.89.113.agj	23

The status bar at the bottom right indicates "Ln: 1411 Col: 4".

Once we have cleaned the data for our implementation, we have written the code for the Mapper and Reducer. In the mapper program, the data is taken from the excel file and the IP addresses are only taken and each is assigned a value of 1. In the reducer program, the IP addresses thus taken will be counted and the final count is displayed along with the IP. The final conclusion can be determined from this.

□ K-Means Algorithm

```
import numpy as np
import os

def compute_euclidean_distance(point, centroid):
    return np.sqrt(np.sum((point - centroid)**2))

def assign_label_cluster(distance, data_point, centroids):
    index_of_minimum = min(distance, key=distance.get)
    return [index_of_minimum, data_point, centroids[index_of_minimum]]

def compute_new_centroids(cluster_label, centroids):
    return np.array(cluster_label + centroids)/2

def iterate_k_means(data_points, centroids, total_iteration):
    label = []
    cluster_label = []
    total_points = len(data_points)
    k = len(centroids)
    for iteration in range(0, total_iteration):
        for index_point in range(0, total_points):
            distance = {}
            for index_centroid in range(0, k): distance[index_centroid]=
compute_euclidean_distance
(data_points[index_point],centroids[index_centroid])
label=assign_label_cluster(distance,data_points[index_point], centroids)
centroids[label[0]]=compute_new_centroids(label[1],centroids[label[0]])
            if iteration == (total_iteration - 1):
                cluster_label.append(label)
    return [cluster_label, centroids]

def print_label_data(result):
    print("Result of k-Means Clustering: \n")
    for data in result[0]:
        print("data point: {}".format(data[1]))
        print("cluster number: {} \n".format(data[0]))
    print("Last centroids position: \n {}".format(result[1]))

def create_centroids(): centroids
= [] centroids.append([5.0,
0.0]) centroids.append([45.0,
70.0]) centroids.append([50.0,
90.0])
return np.array(centroids)

if __name__ == "__main__":
```



```

filename = os.path.dirname(__file__) + "\\data.csv"
data_points = np.genfromtxt(filename, delimiter=",")
centroids = create_centroids()
total_iteration = 100
[cluster_label, new_centroids] = iterate_k_means(data_points, centroids,
total_iteration)
print_label_data([cluster_label, new_centroids])
print()

```

```

Python 2.7.11 Shell
File Edit Shell Debug Options Window Help

data point: 500.0
cluster number: 2

data point: 500.0
cluster number: 2

data point: 500.0
cluster number: 2

data point: 500.0
cluster number: 2

data point: 500.0
cluster number: 2

data point: 500.0
cluster number: 2

data point: 500.0
cluster number: 2

data point: 500.0
cluster number: 2

data point: 500.0
cluster number: 2

data point: 500.0
cluster number: 2

data point: 500.0
cluster number: 2

data point: 500.0
cluster number: 2

```

```

Python 2.7.11 Shell
File Edit Shell Debug Options Window Help

data point: 100.0
cluster number: 1

data point: 100.0
cluster number: 1

data point: 100.0
cluster number: 1

data point: 100.0
cluster number: 1

data point: 100.0
cluster number: 1

data point: 100.0
cluster number: 1

data point: 100.0
cluster number: 1

data point: 100.0
cluster number: 1

data point: 100.0
cluster number: 1

data point: 100.0
cluster number: 1

data point: 100.0
cluster number: 1

Last centroids position:
[[ 1.  1.]
 [100. 100.]
 [500. 500.]]
()
>>> |

```

2. Using Hadoop-

HDFS-Hadoop Distributed File System Hadoop Distributed File System seizes an outsized log files in a superfluous manner across numerous machines in order to accomplish an elevation in terms of accessibility for parallel processing as well as resilience and recovery on occurrence failures. There is a provision of high throughput access to the log files. It is considered to be a block structured file system as it fragments the log files into small blocks which are of fixed size depending upon the percept of the user. Blocks are replicated by the replication factor over a period of time across multiple machines within the Hadoop cluster due to which on occurrences of failure the loss of data can be recuperated

MapReduce-MapReduce is an effortless programming model which is essential for parallel processing of large dimension of data which can be of a structured or unstructured format list of data. Elementary conception of MapReduce is to renovate lists of effort data to lists of productive data. On certain occurrences when data is not in the best of its format and difficult to decipher an outline is required to make the change in the input data. This adaptation is done by MapReduce in order to make the input data comprehensible and this is done in two variant phases namely: Map and Reduce phase simply by segregating the entire work load into fragmented tasks and dispensing them over the numerous machines within the Hadoop cluster.

Implementation –We have used the Cloudera VM software for implementing the map-reduce code in Hadoop. In this system, firstly the file is transferred to HDFS and then it is executed. The input file is taken in the csv format and then we have written the python code and the commands are given to run the code.

The result we are getting is the output of ip that which ip is most occurring-

Procedure-

Step-1: The input file is saved in the folder trend_training >> data named as Book1.txt

This input file has to be put into the Hadoop File System Using the command:

```
hdfs dfs -put /home/training/trend_training/data/Book1.txt /user /training/book_input
```

Step-2: The mapper written in python has to be saved in trend_training >> code folder.

The reducer code also has to be saved in the same folder as the mapper that is, trend_training >> code

Step-3: In the terminal, being in the trend_training >> code directory we have to execute the following script to get the required output.

```
hs mapper.py reducer.py /user /training/book_input /user /training/book_output
```

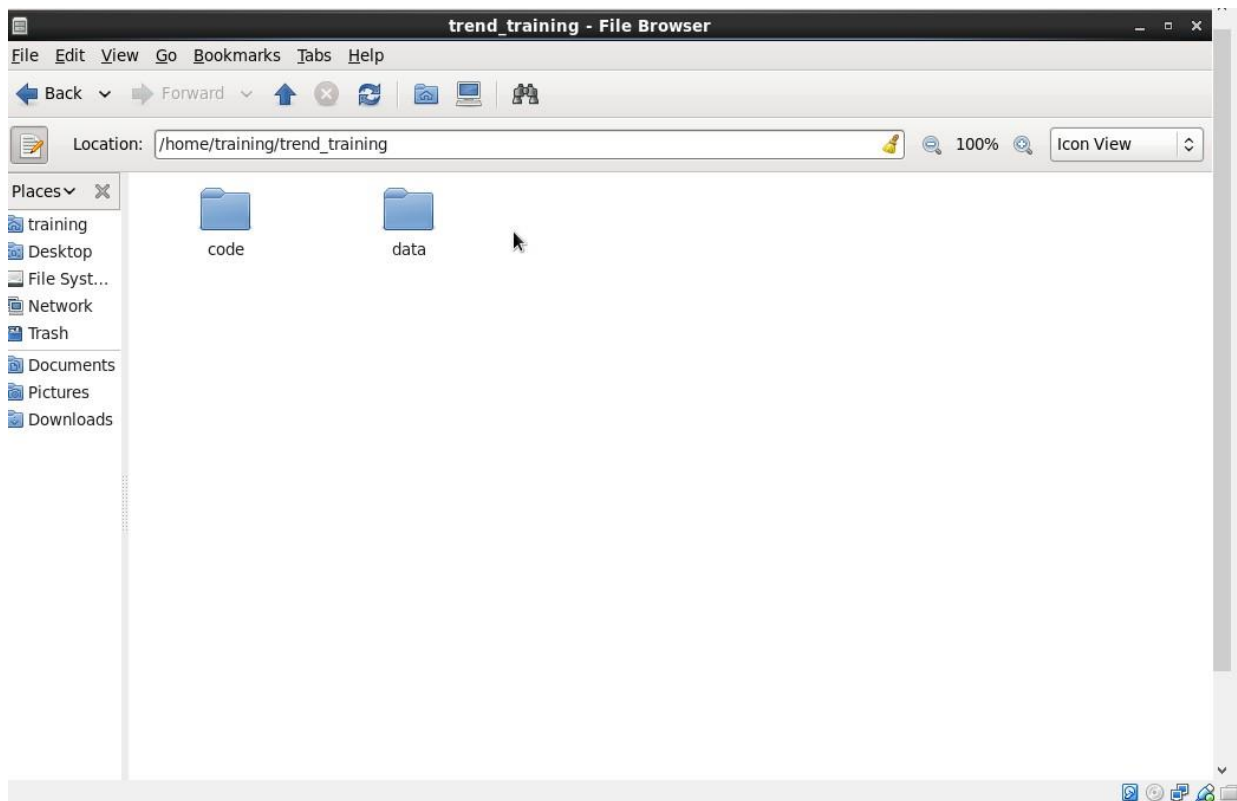
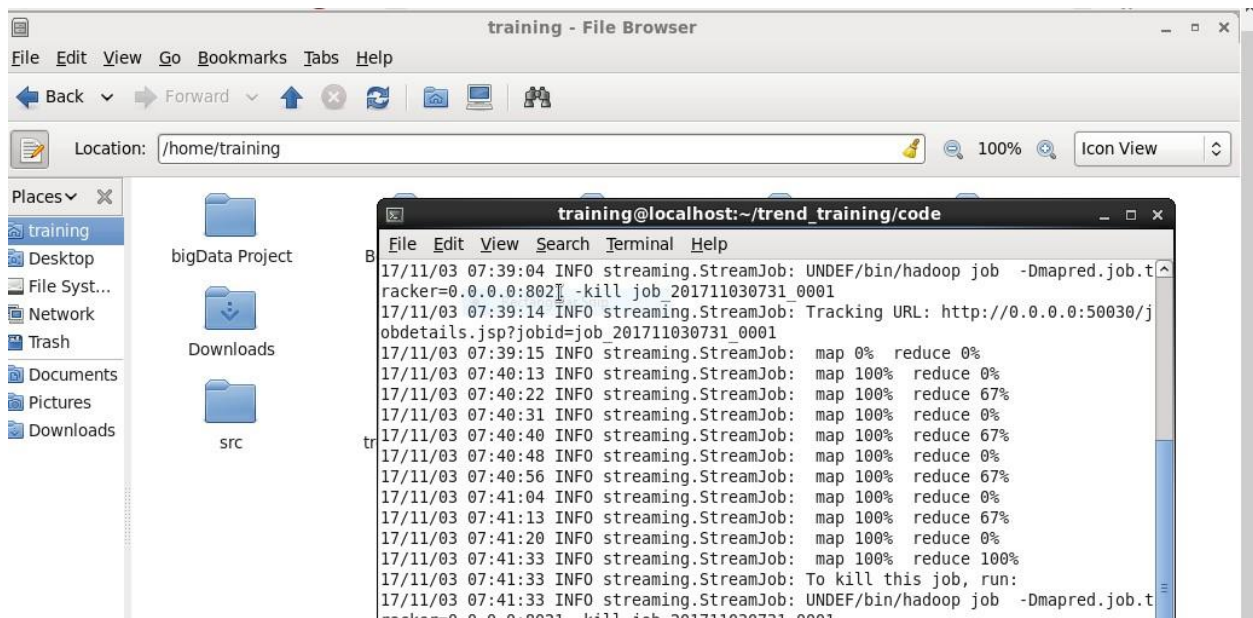
Step-4: The output obtained from the mapper-reducer script is stored in the HDFS.

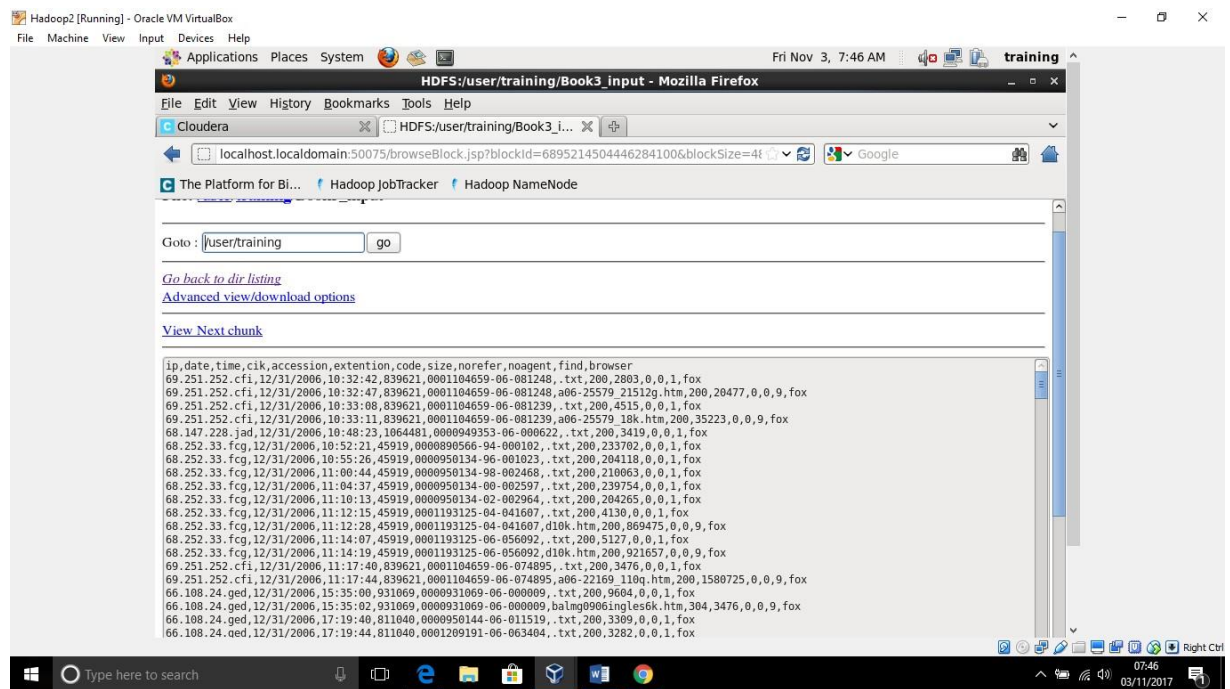
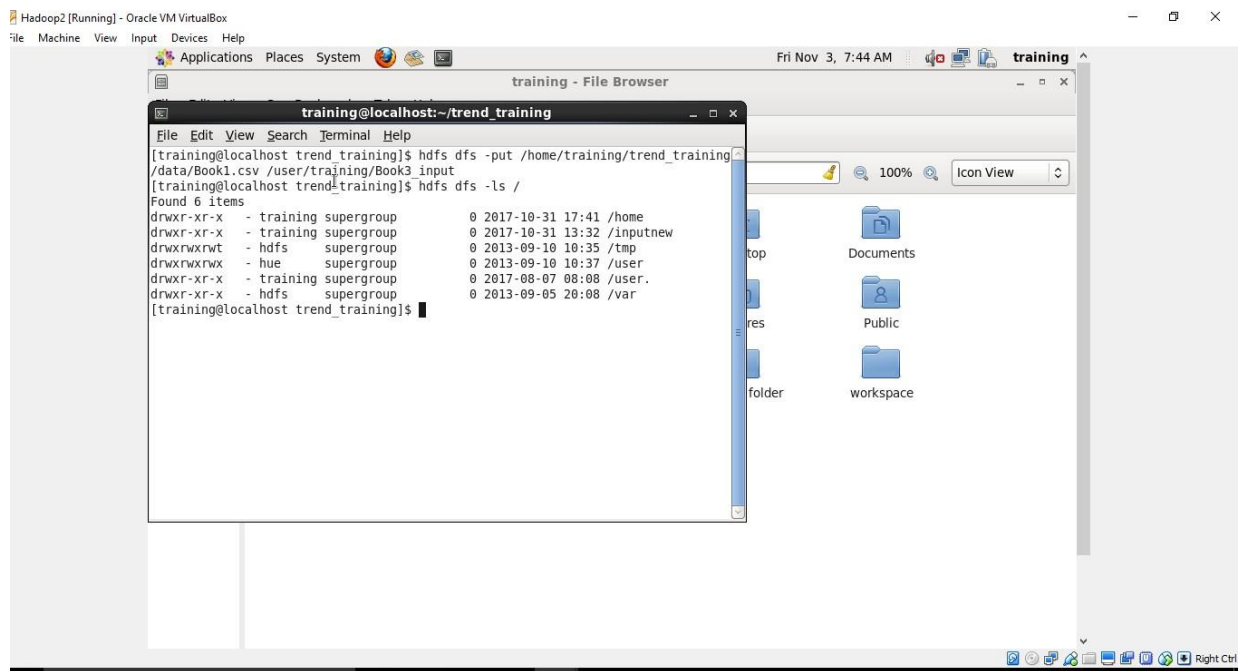
To view the output on the terminal we use the following commands:-

```
hdfs dfs -ls /home/training/book_output
```

```
hdfs dfs -cat /home/training/book_output/part-00000
```

Screenshots-

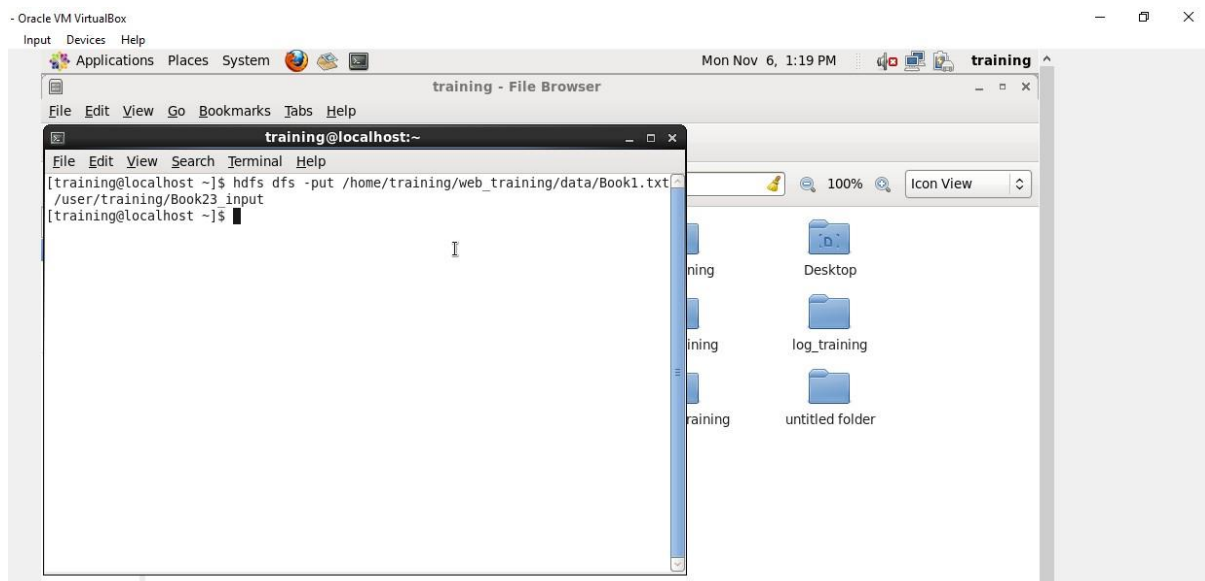
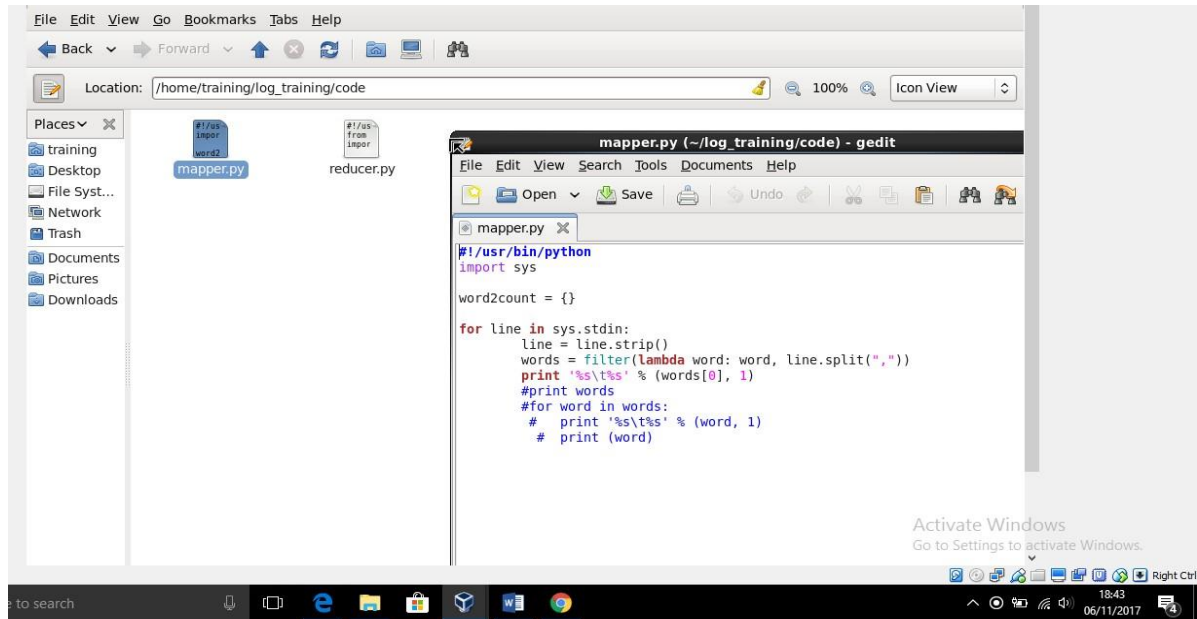




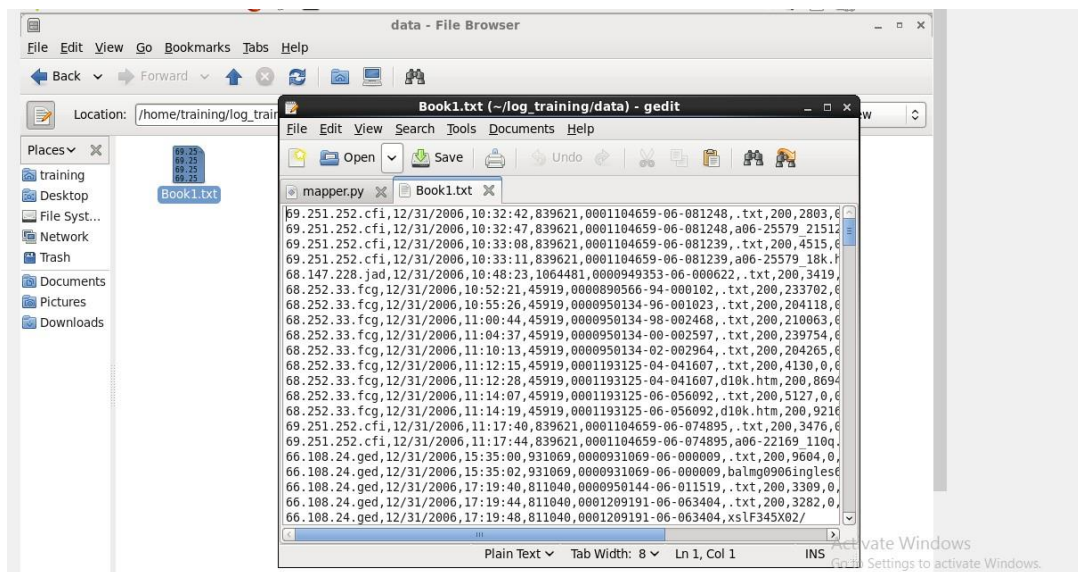
1.Implementing the word count algorithm-

Screenshot of running wordcount algorithm to estimate the count of ip,cik,accession code.

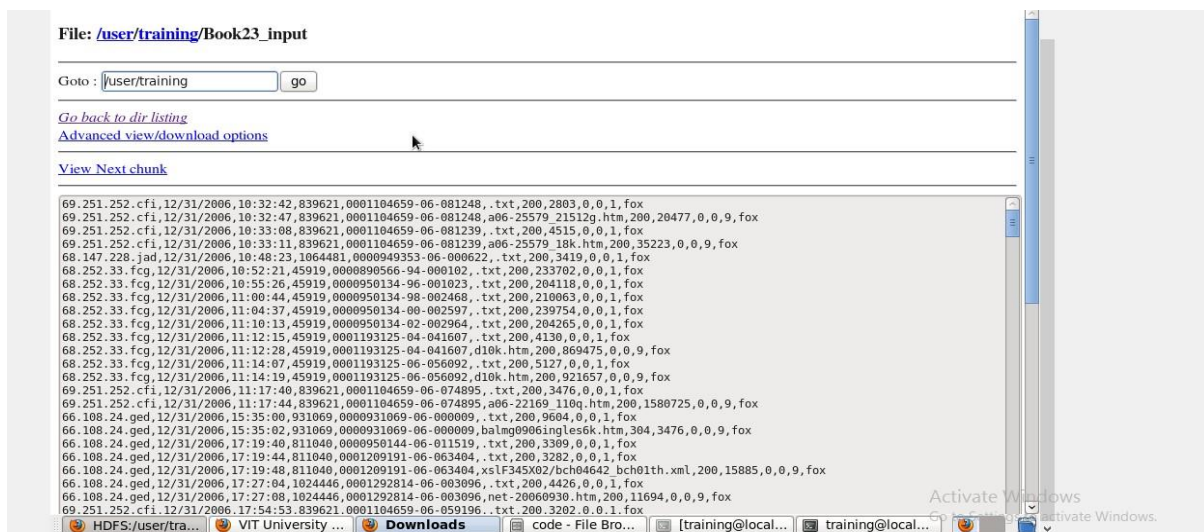
Step-1:Here is the mapper and the reducer code-



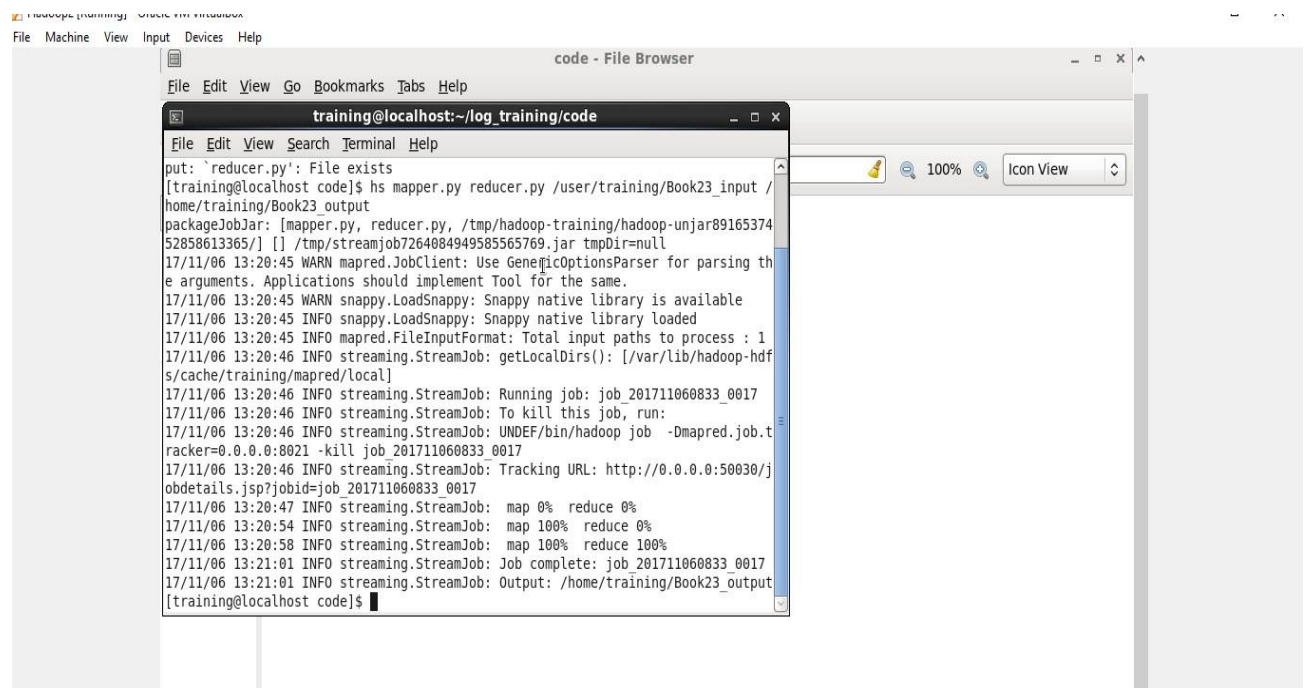
Step-2: The dataset in the txt format



Dataset in the hdf5-



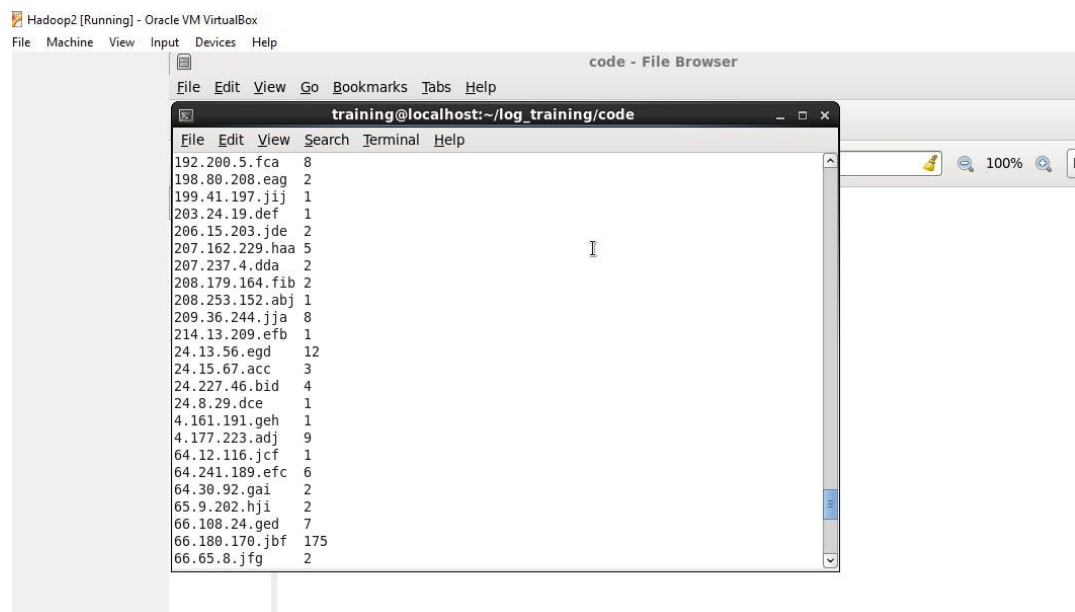
Step-3: Implementing commands to run the mapreduce code-



The screenshot shows a terminal window titled "training@localhost:~/log_training/code". The user has executed the command `put: 'reducer.py': File exists` and then `[training@localhost code]$ hs mapper.py reducer.py /user/training/Book23_input /home/training/Book23_output`. The terminal displays various Hadoop logs, including warnings about the GenicOptionsParser and snappy native library, and information about the job's progress. The job is identified as `job_201711060833_0017`. The logs show the job is running, with map and reduce percentages increasing to 100%. The job is completed, and the output is written to `/home/training/Book23_output`.

Step-4: Viewing the output-

The count is coming of the ip's in front of each ip, representing that how many times the particular ip made request-



The screenshot shows a terminal window titled "training@localhost:~/log_training/code". The terminal displays the output of the Hadoop MapReduce job, which is a list of IP addresses and their corresponding counts. The output is as follows:

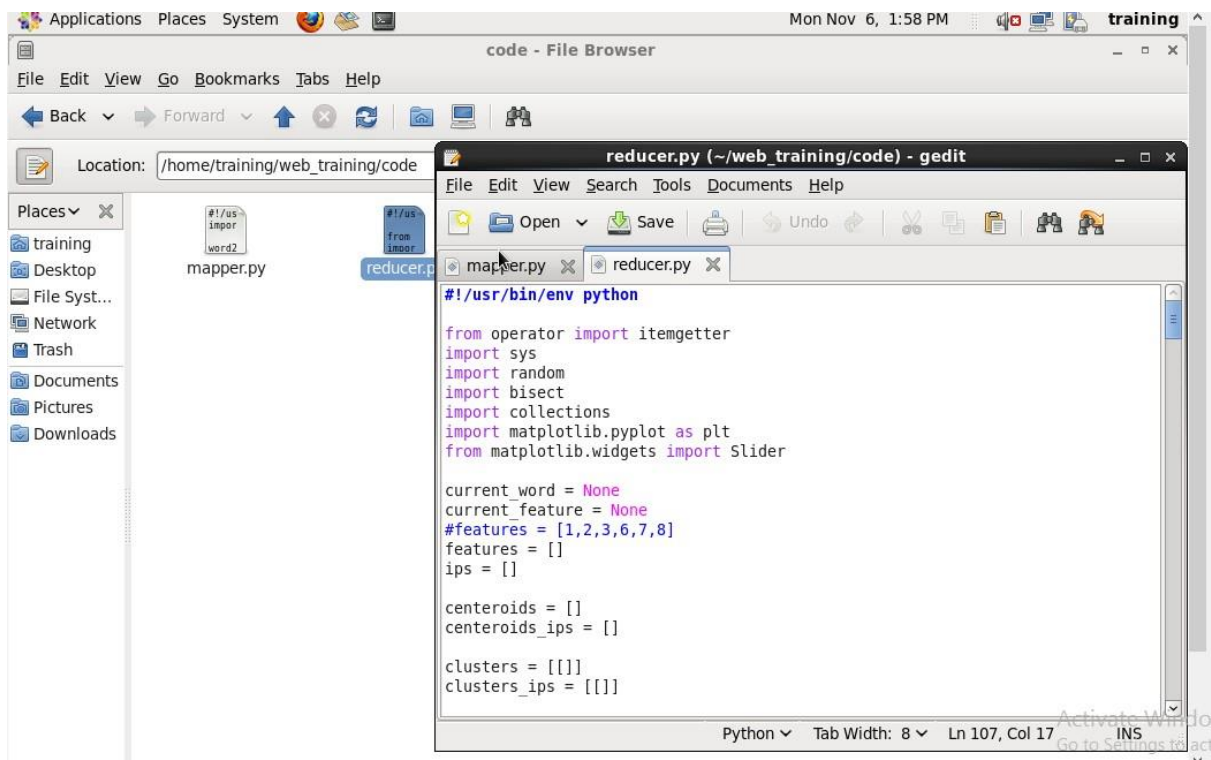
192.200.5.fca	8
198.80.208.eag	2
199.41.197.jij	1
203.24.19.def	1
206.15.203.jde	2
207.162.229.haa	5
207.237.4.dda	2
208.179.164.fib	2
208.253.152.abj	1
209.36.244.jja	8
214.13.209.efb	1
24.13.56.egd	12
24.15.67.acc	3
24.227.46.bid	4
24.8.29.dce	1
4.161.191.geh	1
4.177.223.adj	9
64.12.116.jcf	1
64.241.189.efc	6
64.30.92.gai	2
65.9.202.hji	2
66.108.24.ged	7
66.180.170.jbf	175
66.65.8.jfg	2

2.Implementing K-means clustering algorithm-

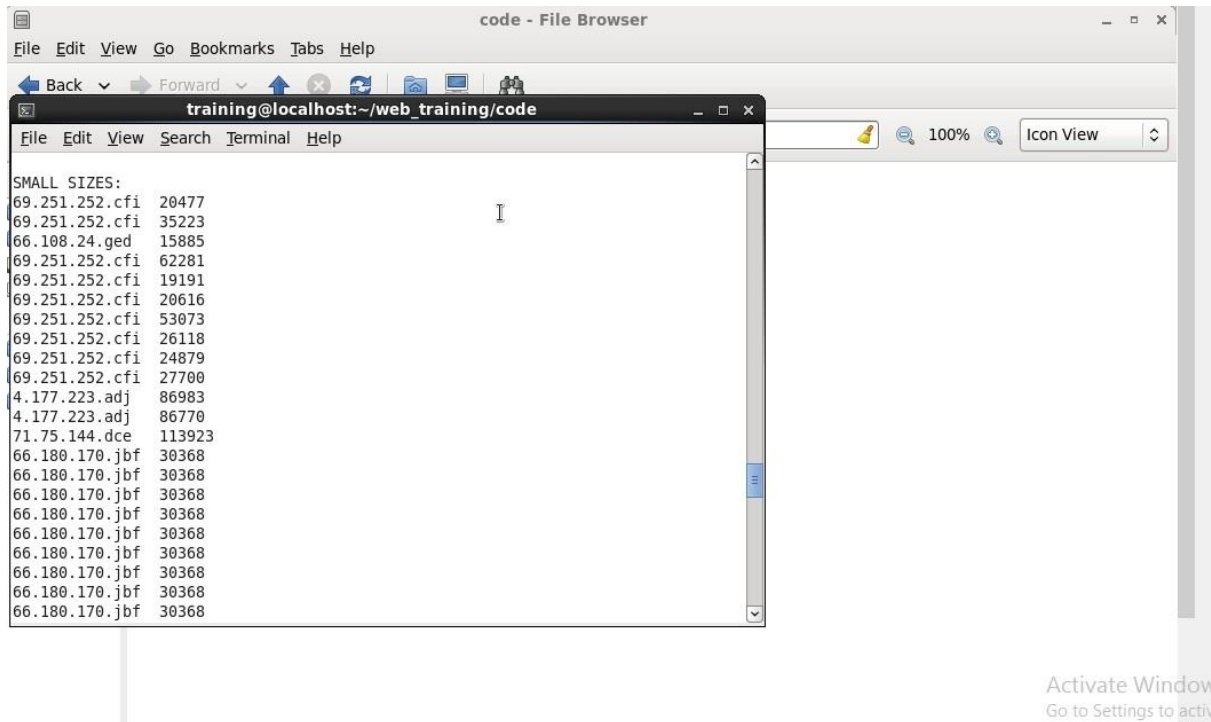
Screenshots of k means clustering algorithm ,such that we are giving our dataset as the input and we are making clusters for the size attribute that will give the 5 different clusters-

- 1.very small
- 2.small
- 3.medium
- 4.large
- 5.very large

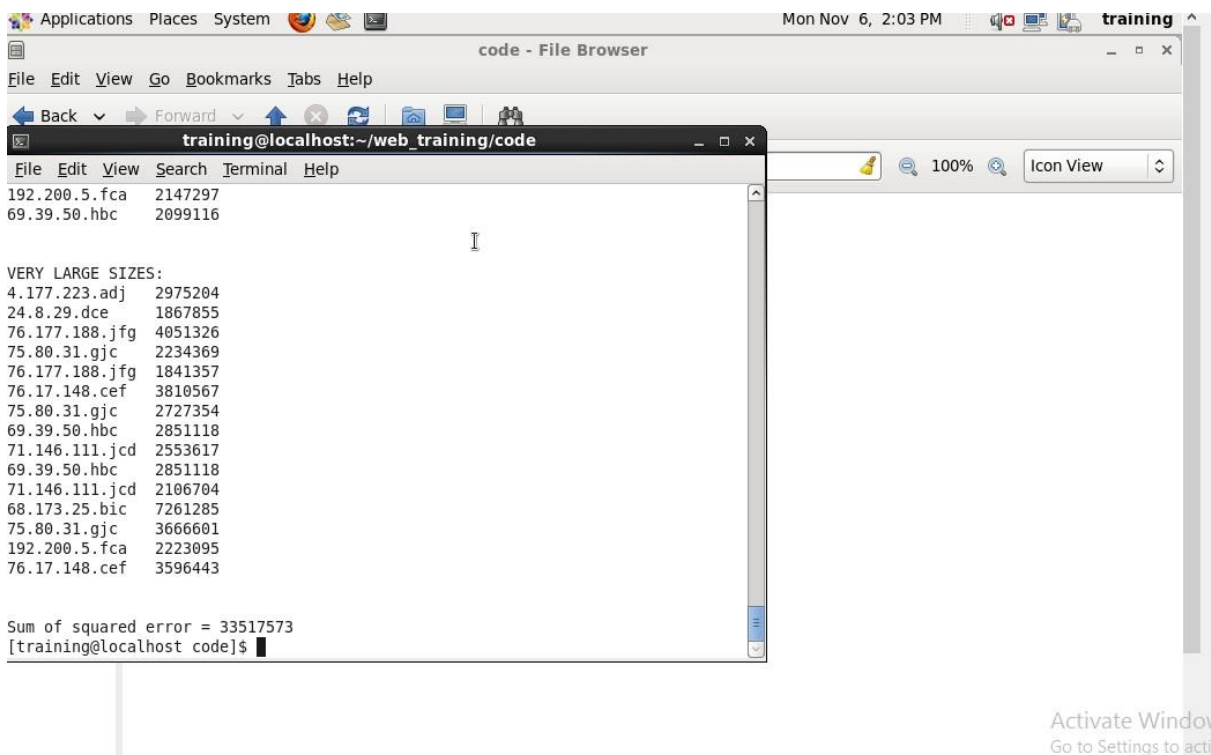
Step-1:The mapper and reducer code is saved in the code folder of the training
The data set is stored in the text format in the data folder of the training.



Step-2:Running the code and the output come



```
code - File Browser
File Edit View Go Bookmarks Tabs Help
Back Forward
training@localhost:~/web_training/code
File Edit View Search Terminal Help
SMALL SIZES:
69.251.252.cfi 20477
69.251.252.cfi 35223
66.108.24.ged 15885
69.251.252.cfi 62281
69.251.252.cfi 19191
69.251.252.cfi 20616
69.251.252.cfi 53073
69.251.252.cfi 26118
69.251.252.cfi 24879
69.251.252.cfi 27700
4.177.223.adj 86983
4.177.223.adj 86770
71.75.144.dce 113923
66.180.170.jbf 30368
66.180.170.jbf 30368
66.180.170.jbf 30368
66.180.170.jbf 30368
66.180.170.jbf 30368
66.180.170.jbf 30368
66.180.170.jbf 30368
66.180.170.jbf 30368
```



```
code - File Browser
File Edit View Go Bookmarks Tabs Help
Back Forward
training@localhost:~/web_training/code
File Edit View Search Terminal Help
192.200.5.fca 2147297
69.39.50.hbc 2099116
VERY LARGE SIZES:
4.177.223.adj 2975204
24.8.29.dce 1867855
76.177.188.jfg 4051326
75.80.31.gjc 2234369
76.177.188.jfg 1841357
76.17.148.cef 3810567
75.80.31.gjc 2727354
69.39.50.hbc 2851118
71.146.111.jcd 2553617
69.39.50.hbc 2851118
71.146.111.jcd 2106704
68.173.25.bic 7261285
75.80.31.gjc 3666601
192.200.5.fca 2223095
76.17.148.cef 3596443
Sum of squared error = 33517573
[training@localhost code]$
```

Thus the 5 different clusters are made.

Sample code-

```
import sys
```

```
word2count = { }
```

```
for line in sys.stdin:
```

```
    line = line.strip()
```

```
    words = filter(lambda word: word, line.split(","))
```

```
        print '%s\t%s' % (words[0], words[7])
```

```
        #print words
```

```
    #for word in words:
```

```
        # print '%s\t%s' % (word, 1)
```

```
        # print (word)
```

CONCLUSION

Trend analysis portrays the users browsing pattern and summarizes the outcome into a graphical report which depicts most visited web pages, browsing session and trending keywords. Hadoop MapReduce framework provides parallel distributed processing and reliable data storage for large volumes of log files. In order to manage such log files, Hadoop MapReduce plays a key role by proficient management of data and decreases the response time. The proposed system with the help of Hadoop MapReduce analyzes the log files and segregates the fields of the log files using regular expression mechanism. Regex not only reduces the code length but also reduces the overhead of usage of string functions. The segregated and structured fields are stored in the database in accordance with Hadoop thereby enabling ease of data retrieval.

REFERENCES

Sayalee Narkhede and Tripti Baraskar, “hmr log analyzer: analyze web application logs over hadoop mapreduce”, International Journal of UbiComp (IJU), Vol.4, No.3, July 2013

Milind Bhandare, Prof. Kuntal Barua, Vikas Nagare, Dynaneshwar Ekhande, Rahul Pawar, “Generic Log Analyzer Using Hadoop Mapreduce Framework”, International Journal of Emerging Technology and Advanced Engineering, Volume 3, Issue 9, September 2013

Navin Kumar Tyagi, A. K. Solanki and Manoj Wadhwa, “Analysis of Server Log by Web Usage Mining for Website Improvement”, IJCSI International Journal of Computer Science Issues, Vol. 7, Issue 4, No 8, July 2010

L.K .Joshila Grace, V.Maheswari, Dhinaharan Nagamalai, “ANALYSIS OF WEB LOGS AND WEB USER IN WEB MINING”, International Journal of Network Security & Its Applications (IJNSA), Vol.3, No.1, January 2011

Praveen Kumar, Dr Vijay Singh Rathore, “Efficient Capabilities of Processing of Big Data using Hadoop Map Reduce”, International Journal of Advanced Research in Computer and Communication Engineering Vol. 3, Issue 6, June 2014

Anuja Pandit, Amruta Deshpande, Prajakta Karmarkar, “Log Mining Based on Hadoop’s Map and Reduce Technique”, International Journal on Computer Science and Engineering (IJCSE) Vol. 5 No. 04 Apr 2013

De Roure D, Jennings NR, Shadbolt NR. The semantic grid: past, present, and future. *Proc IEEE* 2005;93:669–681.

Berners-Lee T, Hendler J, Lassila O. *The Semantic Web*. New York: Scientific American; 2001.

De Roure D, Jennings NR, Shadbolt N. The semantic grid: a future e-Science infrastructure. In: Berman F, Hey AJG, Fox G, eds. *Grid Computing: Making The Global Infrastructure a Reality*. John Wiley & Sons;2003, 437–470.

Ahmed M, Chowdhury ASMR, Ahmed M, Rafee MMH. An advanced survey on cloud computing and state-of-the-art research issues. *Int J Comp Sci* 2012, 9:201–207.

Foster I, Zhao Y, Raicu I, Lu S. Cloud computing and grid computing 360-degree compared. *Grid Comput Environ Workshop*; 2008, 1–10.

Mell P, Grance T. The NIST definition of cloud computing. Recommendations of the National Institute of Standards and Technology; 2011.

Schwiegelshohn U, Badia RM, Bubak M, Danelutto M, Dustdar S, Gagliardi F, Geiger A,

Ladislav Hluchy L, Kranzlmuller D, Laure E, et al. Perspectives on grid computing. *Future Gen Comp Syst* 2010, 26:1104–1115.

Cannataro M, Congiusta A, Pugliese A, Talia D, Trunfio P. Distributed data mining on grids: services, tools, and applications. *IEEE Trans SystMan Cybern B* 2004, 34:2451–2465.

Ellisman M, Brady M, Hart D, Lin FP, Müller M, Smarr L. The emerging role of biogrids. *Commun ACM* 2004, 47:52–57.

Cannataro M, Guzzi PH, Lobosco M, Weber dos Santos R. GridSnake: a Grid-based Implementation of the Snake Segmentation Algorithm. *22nd IEEE International Symposium on Computer-Based Medical Systems, 2009. CBMS 2009*. Albuquerque, NM; 2009, 1–6.

Teixeira GM, Pommeranzembaum IR, de Oliveira BL, Lobosco M, dos Santos RW. Automatic segmentation of cardiac mri using snakes and genetic algorithms. In: Bubak M, van Albada GD, Dongarra J, Sloot PMA, eds. *ICCS (3), volume 5103 of Lecture Notes in Computer Science*. Berlin, Deutschland: Springer; 2008, 168–177.