**BDA Assignment 3**

Team Members

|  |  |
| --- | --- |
| Name | Registration No. |
|  |  |
|  |  |

**Selected Dataset**:

Letter Recognition Data Set

Data Set Information:

The objective is to identify each of a large number of black-and-white rectangular pixel displays as one of the 26 capital letters in the English alphabet. The character images were based on 20 different fonts and each letter within these 20 fonts was randomly distorted to produce a file of 20,000 unique stimuli. Each stimulus was converted into 16 primitive numerical attributes (statistical moments and edge counts) which were then scaled to fit into a range of integer values from 0 through 15. We typically train on the first 16000 items and then use the resulting model to predict the letter category for the remaining 4000. See the article cited above for more details.

**Explain what fields are present in csv file used to import data to Pig**:

1. lettr capital letter (26 values from A to Z)   
2. x-box horizontal position of box (integer)   
3. y-box vertical position of box (integer)   
4. width width of box (integer)   
5. high height of box (integer)   
6. onpix total # on pixels (integer)   
7. x-bar mean x of on pixels in box (integer)   
8. y-bar mean y of on pixels in box (integer)   
9. x2bar mean x variance (integer)   
10. y2bar mean y variance (integer)   
11. xybar mean x y correlation (integer)   
12. x2ybr mean of x \* x \* y (integer)   
13. xy2br mean of x \* y \* y (integer)   
14. x-ege mean edge count left to right (integer)   
15. xegvy correlation of x-ege with y (integer)   
16. y-ege mean edge count bottom to top (integer)   
17. yegvx correlation of y-ege with x (integer)

**Explain what problem is solved using Pig**:

json file was first loaded and then pig UDF function was applied on one of the variable to check they fit in the range into a range of integer values from 0 through 15.

In pig json file was loaded using load command and attribute list where specified by using JsonLoader function. then grouping was done using llettr attribute and MAX function is used to find max of all x-box attribute in the group. dump to display result. min is used to find minimum of x-box attribute in the following statement and dump to display result. In following statement count function is used to count the number of x-box values and dump to display result. The statement which follows uses sum function find the sum of all values in x-box attribute and dump to display the result. The statement after that finds the average of all values in x-box attribute and dump to display result.

In the statement which uses filter operation find the tuples of those letters where the x-box is greater than average using avg function.

In the next statement join is used to join two relations letter relation with another relation containing the frequency of usage of a letter.

**Pig commands:**

Jar filename: myudfs.jar

Filename: ASSIGN.java

package myudfs;

import java.io.IOException;

import org.apache.pig.EvalFunc;

import org.apache.pig.data.Tuple;

import org.apache.pig.impl.util.WrappedIOException;

public class ASSIGN extends EvalFunc<String>{

public String exec(Tuple input) throws IOException {

if(input == null || input.size() == 0)

return null;

try{

int i = (Integer)input.get(0);

if(i > 15){

i = 15;

}

else if(i < 0){

i = 0;

}

return i;

}catch(Exception e){

throws WrappedIOException.wrap("Caught exception processing input row ",e);

}

}

}

register /root/pigdemo/myudfs.jar;

letter = LOAD 'letter.json' USING JsonLoader('llettr:chararray, x-box:int, y-box:int, width:int, high:int, onpix:int, x-bar:int, y-bar:int, x2bar:int, y2bar:int, xybar:int, x2ybr:int, xy2br:int, x-ege:int, xegvy:int, y-ege:int, yegvx:int');

letter1 = FOREACH letter GENERATE myudfs.ASSIGN(x-box);

DUMP letter1;

letter = LOAD 'letter.json' USING JsonLoader('llettr:chararray, x-box:int, y-box:int, width:int, high:int, onpix:int, x-bar:int, y-bar:int, x2bar:int, y2bar:int, xybar:int, x2ybr:int, xy2br:int, x-ege:int, xegvy:int, y-ege:int, yegvx:int');

letter2 = group letter by llettr;

letter3 = foreach letter2 generate group as grp, MAX(letter.x-box);

dump letter3;

letter4 = foreach letter2 generate group as grp, MIN(letter.x-box);

dump letter4;

letter5 = foreach letter2 generate group as grp, COUNT(letter.x-box);

dump letter5;

letter6 = foreach letter2 generate group as grp, SUM(letter.x-box);

dump letter6;

letter7 = foreach letter2 generate group as grp, AVG(letter.x-box);

dump letter7;

average = FOREACH grouped\_count\_bag GENERATE AVG(letter.x-box) AS avg;

letter\_average = CROSS letter, average;

filtered\_letter\_average = FILTER letter\_average BY (error\_count>avg);

A = LOAD 'letter.json' USING JsonLoader('llettr:chararray, x-box:int, y-box:int, width:int, high:int, onpix:int, x-bar:int, y-bar:int, x2bar:int, y2bar:int, xybar:int, x2ybr:int, xy2br:int, x-ege:int, xegvy:int, y-ege:int, yegvx:int');

B = LOAD 'Relative.json' USING JsonLoader('Letter:chararray, frequency:float');

C = JOIN A BY llettr, B BY Letter;

dump C;

dump B;

**R script**:

data <- read.csv("C:\dataset.csv")

plot(new$Category,new$max,main="Comparison Maximum of x.box and Letter", xlab="Letter", ylab="Value")

plot(new$Category,new$min,main="Comparison Minimum of x.box and Letter", xlab="Letter", ylab="Value")

plot(new$Category,new$sum,main="Comparison Sum of x.box and Letter", xlab="Letter", ylab="Value")

plot(new$Category,new$avd,main="Comparison Average of x.box and Letter", xlab="Letter", ylab="Value")

plot(new$Category,new$x,type = "o",xlab="Letter",ylab="Count",main="Comparision of Count and Letter")

hist(new3$x.box,xlab="X Box avg",main="Histogram of average of x.box")

plot(newtemp$llettr,newtemp$x.box,xlab="Letter",ylab="Count",main="Comparsion between mean and letter")

plot(new$x,data$Relative.frequency.in.the.English.language,xmap="x.map",xlab="X map",ylab="Relative frequency in the English language",main="Comparison after of join operation")

plot(new$x,data$Letter,xmap="x.map",xlab="X map",ylab="Letter",main="Comparison between letter and X map")

**Screenshot of visualization in R**:







