**DMBI PROJECT**

**(Association)**

PROJECT MEMBERS

Taufiq Monghal

Shubhangi Shinde

Videet Singhai

**Aim:** SELECTING STUDENTS FOR REMEDIAL CLASSES USING SCORING BASED ON ASSOCIATION RULE METHOD

**a) Problem definition, Identifying which data mining task is needed**

Problem Statement:

1. In Educational institutions the existing system of identifying weak students is based on the traditional grade point methods which select too many students for remedial classes.
2. The existing system measures the student’s performance by an aggregate score computed from individual subject scores using a complex formula.
3. This is burden to teachers as they have to take classes for more number of students.

Identifying which data mining task is needed:

We use data mining techniques as tools for efficiently identifying weak students from the student database. We use SBA (scoring based on association) method for identifying weak students and selecting them for remedial classes. We used the prefix tree data structure to represent this information in main memory. This prefix tree will be mined to find the most frequent sequential pattern(s) using **FP growth algorithm**. This result will then be used to score the present student database, thereby identifying the weak students and providing them remedial classes.

**b) Identify and use a standard data mining dataset available for the problem. Some links for data mining datasets are:** [**WEKA**](http://moodle.spit.ac.in/mod/resource/view.php?id=3804) **site, UCI Machine Learning Repository, KDD site, KDD Cup etc.**

Since the dataset we found online were not up to our expectations and ideologies of implementation we have generated our own dataset for fulfilling the results. For generation of database the following is the code we devised in java:

GenerateDB.java(2 lakh Record)

import java.io.\*;

class GenerateDB

{

PrintWriter pw;

Generatedb(PrintWriter pw)

{

this.pw=pw;

}

void generate(int num)

{

char c;

int r,count;

for(int i=0;i<num;i++)

{

count=0;

for(int j=0;j<FPAlgo.numsub;j++)

{

r=(int)(Math.random()\*100);

if(r>=90)

{

c='A';

count+=10;

}

else if(r>=70)

{

c='B';

count+=9;

}

else if(r>=40)

{

c='C';

count+=8;

}

else if(r>=20)

{

c='D';

count+=7;

}

else if(r>=10)

{

c='E';

count+=6;

}

else

{

c='F';

count+=5;

}

pw.print(c);

}

if(count/FPAlgo.numsub>=7)

{

r=(int)(Math.random()\*10);

if(r>=2)

pw.println(" P");

else

pw.println(" F");

}

else

{

r=(int)(Math.random()\*10);

if(r>=2)

pw.println(" F");

else

pw.println(" P");

}

}

}

public static void main(String arg[])throws Exception

{

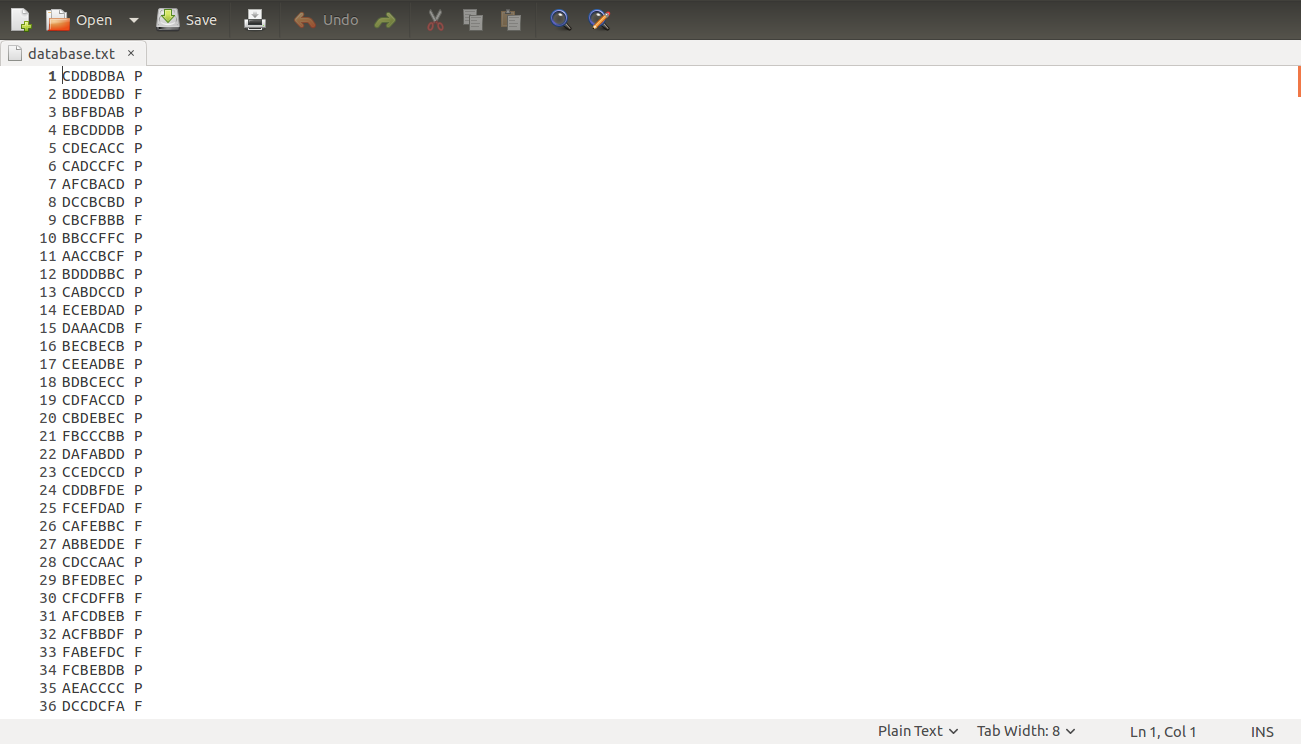
PrintWriter pw=new PrintWriter(new OutputStreamWriter(new FileOutputStream("database.txt")),true);

new Generatedb(pw).generate(200000);

}

}

**OUTPUT:**



**Generatepdb.java (200 Records)**

import java.io.\*;

class Generatepdb

{

PrintWriter pw;

Generatepdb(PrintWriter pw)

{

this.pw=pw;

}

void generate(int num)

{

char c;

int r,count;

for(int i=0;i<num;i++)

{

count=0;

for(int j=0;j<FPAlgo.numsub;j++)

{

r=(int)(Math.random()\*100);

c=(char)((int)(Math.random()\*6)+'A');

pw.print(c);

}

pw.println();

}

}

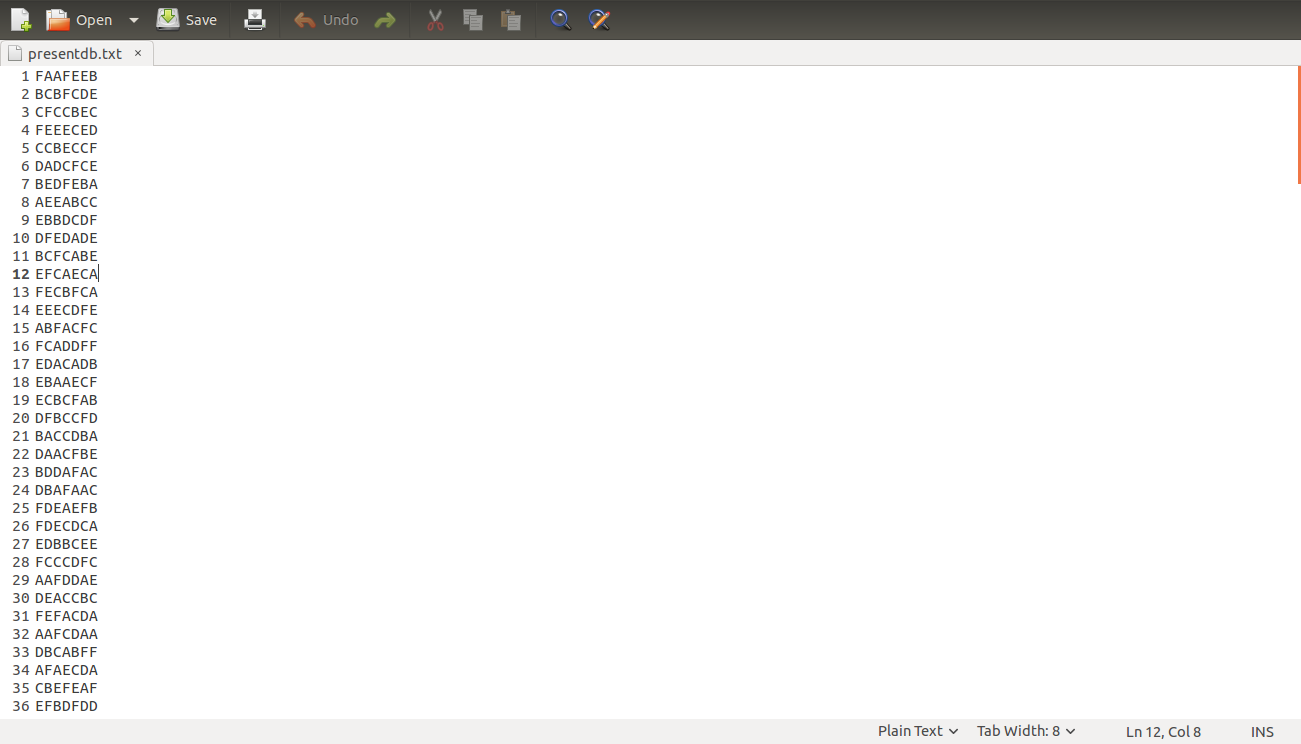
public static void main(String arg[])throws Exception

{

PrintWriter pw=new PrintWriter(new OutputStreamWriter(new FileOutputStream("presentdb.txt")),true);

new Generatepdb(pw).generate(200);

}

}

**OUTPUT:**

**FPGrowth.java**

import java.io.\*;

class FPAlgo

{

final static int numsub=7,numgrade=6;

final static float tsup=2f;

int supp,supf;

float conp,conf;

PrintWriter pw;

BufferedReader br;

FPAlgo()

{

try

{

pw=new PrintWriter(new OutputStreamWriter(new FileOutputStream("itemset.txt")),true);

br=new BufferedReader(new FileReader("database.txt"));

}

catch(Exception e)

{

System.out.println(e);

}

}

String reverse(String st)

{

String r="";

int len=st.length()-1;

while(len>=0)

{

r+=st.charAt(len);

len--;

}

return r;

}

void produce(FPTree tree,String suf)

{

String suf1="";

Node li,f;

float cp,cf;

for(int i=0;i<numsub;i++)

for(int j=0;j<numgrade;j++)

{

FPTree nt=new FPTree();

suf1=(char)(i+'1')+""+(char)(j+'A')+suf;

cp=tree.arr[i][j][0].count+tree.arr[i][j][1].count;

if(cp>0)

cp=tree.arr[i][j][0].count\*100/cp;

cf=100-cp;

if((tree.arr[i][j][0].count>=supp&&cp>=conp)||(tree.arr[i][j][1].count>=supf&&cf>=conf))

{

li=tree.arr[i][j][0].next;

while(li!=null)

{

String st="";

f=li.fat;

while(f!=tree.root)

{

st+=f.data;

f=f.fat;

}

if(!st.equals(""))

{

st=reverse(st);

for(int x=0;x<li.count;x++)

nt.insert(st,true);

}

li=li.next;

}

li=tree.arr[i][j][1].next;

while(li!=null)

{

String st="";

f=li.fat;

while(f!=tree.root)

{

st+=f.data;

f=f.fat;

}

if(!st.equals(""))

{

st=reverse(st);

for(int x=0;x<li.count;x++)

nt.insert(st,false);

}

li=li.next;

}

if(tree.arr[i][j][0].count>=supp&&cp>=conp)

pw.println(suf1+"P"+" "+tree.arr[i][j][0].count+" "+cp);

else

pw.println(suf1+"F"+" "+tree.arr[i][j][1].count+" "+cf);

produce(nt,suf1);

}

}

}

void mine(FPTree tree)

{

String suf="";

Node li,f;

float cp,cf;

for(int i=0;i<numsub;i++)

for(int j=0;j<numgrade;j++)

{

FPTree nt=new FPTree();

suf=(char)(i+'1')+""+(char)(j+'A');

cp=tree.arr[i][j][0].count+tree.arr[i][j][1].count;

if(cp>0)

cp=tree.arr[i][j][0].count\*100/cp;

cf=100-cp;

if((tree.arr[i][j][0].count>=supp&&cp>=conp)||(tree.arr[i][j][1].count>=supf&&cf>=conf))

{

li=tree.arr[i][j][0].next;

while(li!=null)

{

String st="";

f=li.fat;

while(f!=tree.root)

{

st+=f.data;

f=f.fat;

}

if(!st.equals(""))

{

st=reverse(st);

for(int x=0;x<li.count;x++)

nt.insert(st,true);

}

li=li.next;

}

li=tree.arr[i][j][1].next;

while(li!=null)

{

String st="";

f=li.fat;

while(f!=tree.root)

{

st+=f.data;

f=f.fat;

}

if(!st.equals(""))

{

st=reverse(st);

for(int x=0;x<li.count;x++)

nt.insert(st,false);

}

li=li.next;

}

if(tree.arr[i][j][0].count>=supp&&cp>=conp)

pw.println(suf+"P"+" "+tree.arr[i][j][0].count+" "+cp);

else

pw.println(suf+"F"+" "+tree.arr[i][j][1].count+" "+cf);

produce(nt,suf);

}

}

}

FPTree formFPTree()

{

int p=0,f=0;

FPTree fp=new FPTree();

try

{

String s,sa[];

while((s=br.readLine())!=null)

{

sa=s.split(" ");

if(sa[1].charAt(0)=='P')

{

p++;

fp.insert(sa[0],true);

}

else

{

f++;

fp.insert(sa[0],false);

}

}

conf=p+f;

conp=p\*100/conf;

conf=100-conp;

supp=(int)(p\*tsup/100);

supf=(int)(f\*tsup/100);;

//System.out.println(supp+" "+supf+" "+conp+" "+conf);

}

catch(Exception e)

{

System.out.println(e);

}

return fp;

}

public static void main(String[] arg)

{

FPAlgo fpa=new FPAlgo();

//System.out.println(fpa.formFPTree().arr[1][0][1].count);

fpa.mine(fpa.formFPTree());

Scorer sr=new Scorer();

sr.filldata();

sr.score();

sr.display(sr.head);

}

}

class FPTree

{

Node root;

Node arr[][][],end[][][];

FPTree()

{

root=new Node();

arr=new Node[FPAlgo.numsub][FPAlgo.numgrade][2];

for(int i=0;i<FPAlgo.numsub;i++)

for(int j=0;j<FPAlgo.numgrade;j++)

{

arr[i][j][0]=new Node();

arr[i][j][1]=new Node();

}

}

void traverse(Node tr,int lev)

{

Node s=tr.son;

while(s!=null)

{

traverse(s,lev+1);

s=s.sib;

}

System.out.println(lev+""+tr.data+" "+tr.count);

}

void insert(String str,boolean p)

{

int len=str.length();

Node k=root,s,r;

char c;

int in=p==true?0:1;

for(int i=0;i<len;i++)

{

c=str.charAt(i);

s=k.son;

while(s!=null&&(s.data!=c||s.pass!=p))

s=s.sib;

if(s==null)

{

s=new Node(c,p);

s.next=arr[i][c-'A'][in].next;

arr[i][c-'A'][in].next=s;

s.fat=k;

s.sib=k.son;

k.son=s;

}

arr[i][c-'A'][in].count++;

s.count++;

k=s;

}

}

}

class Node

{

Node son,sib,fat,next;

char data;

int count;

boolean pass;

Node()

{

}

Node(char d,boolean p)

{

pass=p;

data=d;

}

}

class Scorer

{

Nod head;

PrintWriter pw;

Scorer()

{

try

{

pw=new PrintWriter(new OutputStreamWriter(new FileOutputStream("result.txt")),true);

}

catch(Exception e)

{

e.printStackTrace();

}

}

void filldata()

{

try

{

BufferedReader br=new BufferedReader(new FileReader("presentdb.txt"));

String st;

head=new Nod();

while((st=br.readLine())!=null)

insert(st);

}

catch(Exception e)

{

System.out.println(e);

}

}

void insert(String str)

{

int len=str.length();

Nod s=head.son,f=head;

for(int i=0;i<len;i++)

{

while(s!=null)

{

if(s.data==str.charAt(i))

break;

s=s.sibling;

}

if(s==null)

{

Nod n;

for(int j=i;j<len;j++)

{

n=new Nod(str.charAt(j));

n.sibling=f.son;

f.son=n;

f=n;

}

break;

}

f=s;

s=f.son;

}

f.lf=new Leaf();

}

void scoreitem(int lev,String st,Nod tr,float con,int sup)//lev=1

{

char c=st.charAt(0);

if(tr.lf!=null)

{

if(c=='P')

{

tr.lf.WN+=con\*sup/3;

tr.lf.N+=con\*sup\*(100-con)/3;

}

else

{

tr.lf.WP+=con\*sup;

tr.lf.P+=con\*sup\*con;

}

return;

}

if(lev!=c-'0')

{

Nod s=tr.son;

while(s!=null)

{

scoreitem(lev+1,st,s,con,sup);

s=s.sibling;

}

}

else

{

c=st.charAt(1);

Nod s=tr.son;

while(s!=null&&s.data!=c)

s=s.sibling;

if(s!=null)

scoreitem(lev+1,st.substring(2),s,con,sup);

}

}

char arr[]=new char[15];

int in;

void display(Nod tr)

{

int count;

if(tr.lf!=null)

{

count=0;

for(int j=0;j<in;j++)

{

pw.print(arr[j]);

count+=10-(arr[j]-'A');

}

if((tr.lf.WP+tr.lf.WN)!=0)

pw.println(" "+count/in+" "+((tr.lf.P+tr.lf.N)/(tr.lf.WP+tr.lf.WN)));

else

pw.println(" 0");

return;

}

Nod s=tr.son;

while(s!=null)

{

arr[in++]=s.data;

display(s);

in--;

s=s.sibling;

}

}

void score()

{

try

{

BufferedReader br=new BufferedReader(new FileReader("itemset.txt"));

String st,sa[];

while((st=br.readLine())!=null)

{

sa=st.split(" ");

scoreitem(1,sa[0],head,Float.parseFloat(sa[2]),Integer.parseInt(sa[1]));

}

}

catch(Exception e)

{

System.out.println(e);

}

}

}

class Nod

{

Nod(char c)

{

data=c;

}

Nod()

{

}

char data;

Nod son,sibling;

Leaf lf;

}

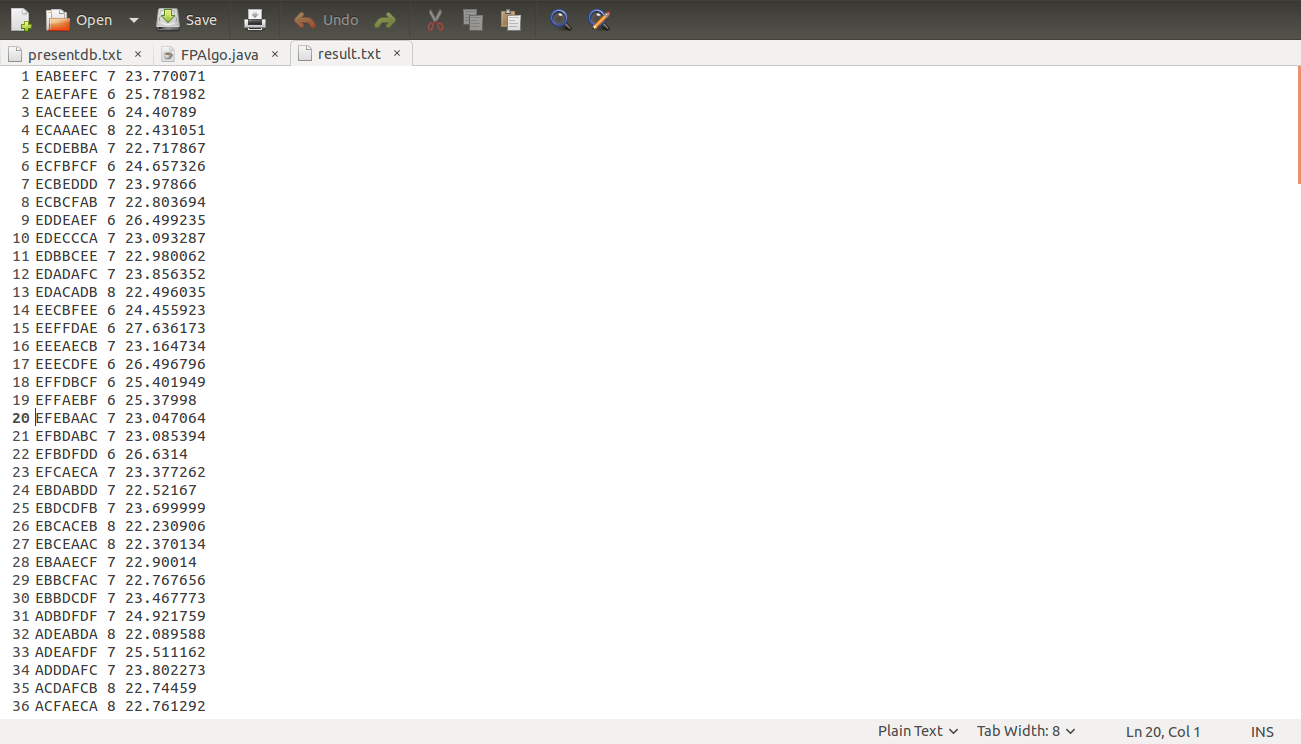
class Leaf

{

float P,N,WP,WN;

}

result.txt



**c) Provide clearly the BI decision that is to be taken as a result of mining.**

BI decision to be taken is whether a student will be selected for the remedial class or not based on the score devised using association rules. The score if more than 23 is selected for remedial class.

**d) Interpret and visualize the results**

**PYTHON CODE:**

import matplotlib.pyplot as plt

import numpy as np

list\_conf =[]

CGPA = []

min\_conf=raw\_input("Give the minimum conf\n")

f = open("result.txt")

for line in f:

conf = float(line.split()[2])

CGPA.append(int(line.split()[1]))

min\_conf = float(min\_conf)

if conf > min\_conf:

list\_conf.append(conf)

#print list

f.close()

labels = 'Remedial', 'Non-Remedial'

sizes = [len(list\_conf),200]

explode = (0.1, 0) # only "explode" the 2nd slice (i.e. 'Hogs')

fig1, ax1 = plt.subplots()

ax1.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%',

shadow=True, startangle=90)

ax1.axis('equal') # Equal aspect ratio ensures that pie is drawn as a circle.

plt.show()

c\_six=0

c\_sev=0

c\_eig=0

c\_nine=0

for number in CGPA:

if number == 6:

c\_six+=1

elif number == 7:

c\_sev+=1

elif number == 8:

c\_eig+=1

else:

c\_nine+=1

n\_groups = 4

cgpa = (c\_six, c\_sev, c\_eig, c\_nine)

std\_men = (2, 7, 4, 1)

fig, ax = plt.subplots()

index = np.arange(n\_groups)

bar\_width = 0.35

opacity = 0.4

error\_config = {'ecolor': '0.3'}

rects1 = plt.bar(index, cgpa, bar\_width,

alpha=opacity,

color='b',

yerr=std\_men,

error\_kw=error\_config,

label='Count of students')

plt.xlabel('CGPA')

plt.ylabel('Count')

plt.title('CGPA Graph')

plt.xticks(index + bar\_width / 2, ('6', '7', '8', '9'))

plt.legend()

plt.tight\_layout()

def autolabel(rects):

for rect in rects:

height = rect.get\_height()

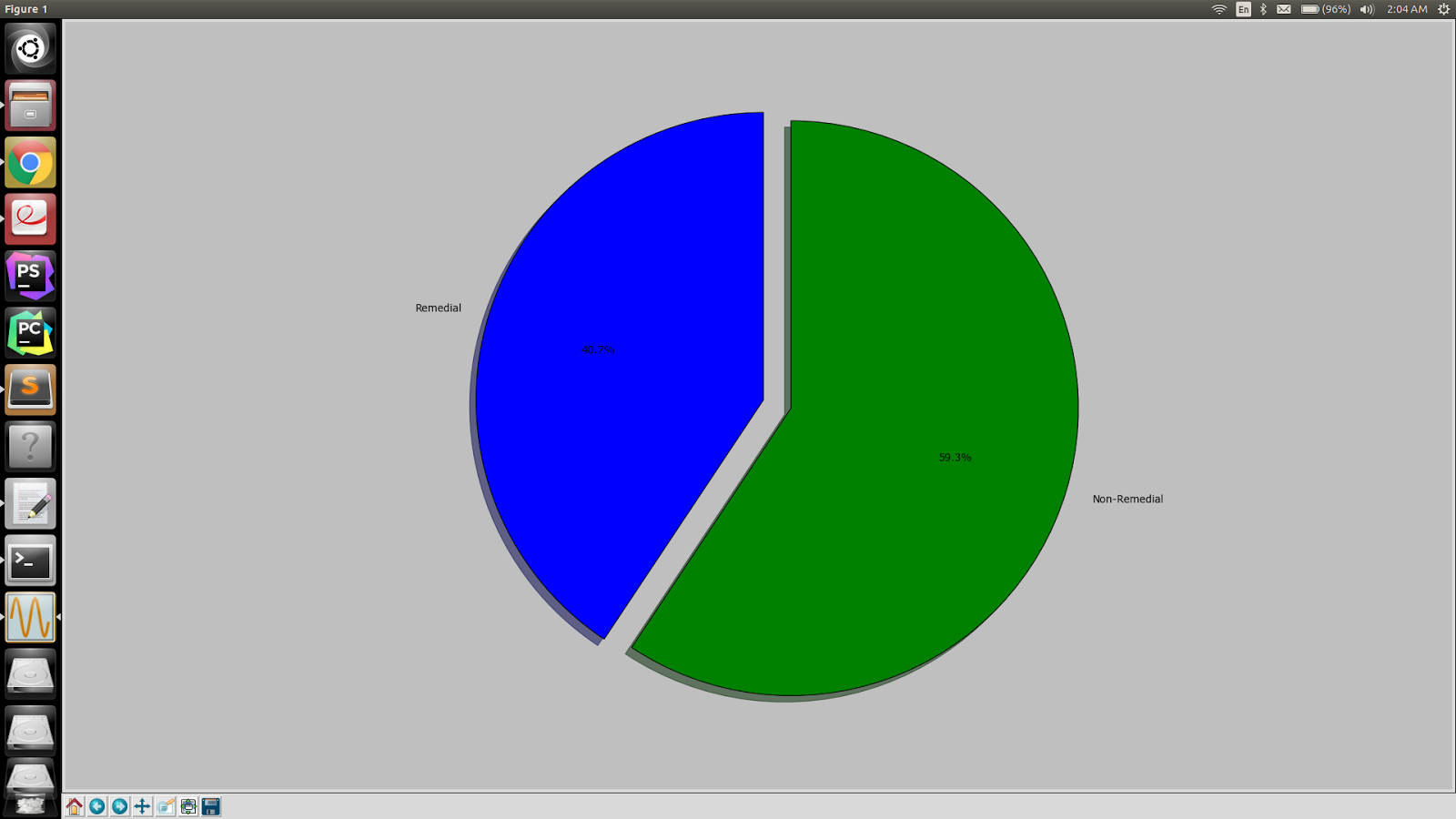
ax.text(rect.get\_x() + rect.get\_width()/2., 1.07\*height,

'%d' % int(height),

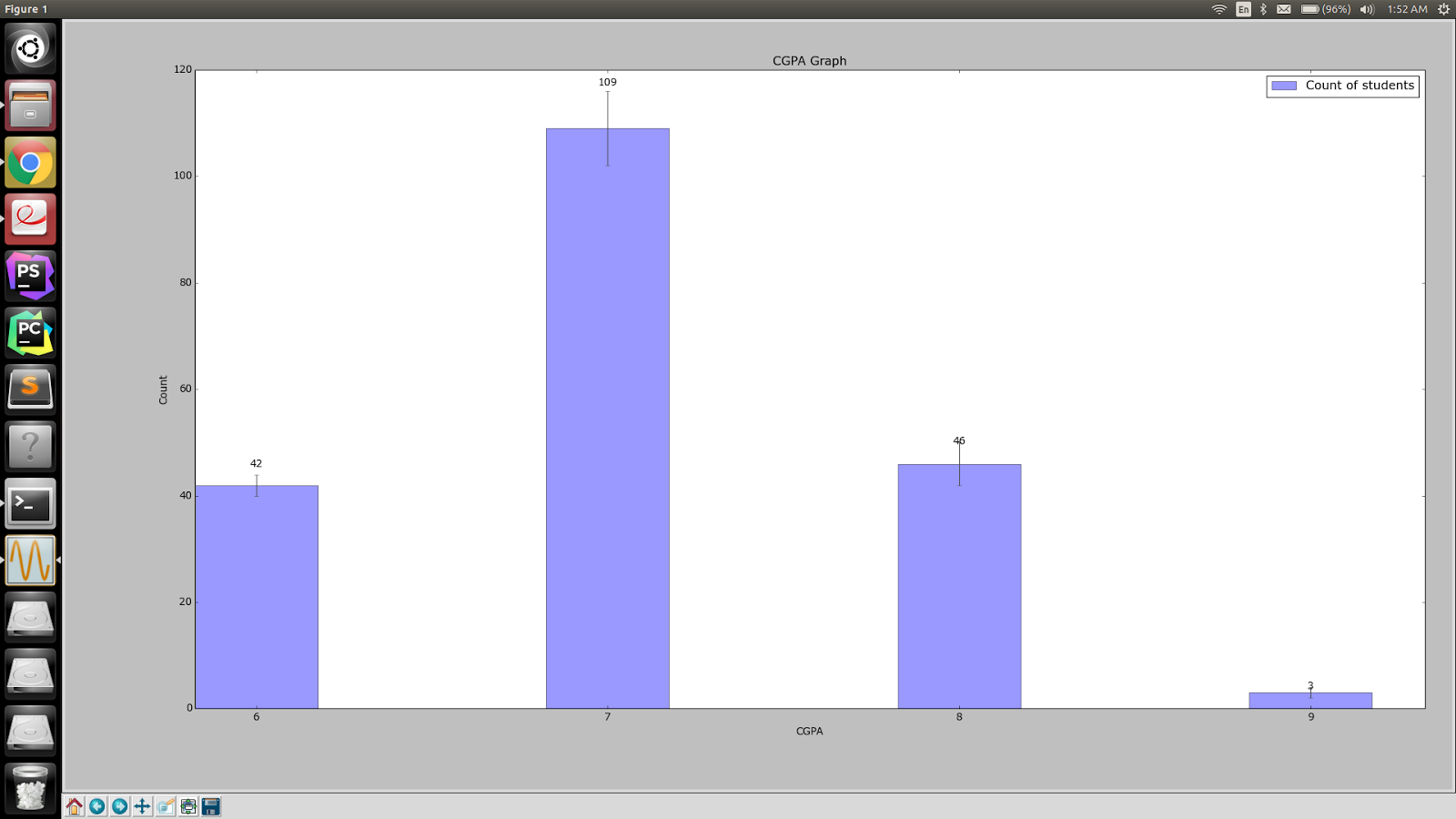
ha='center', va='bottom')

autolabel(rects1)

plt.show()



The above is the visualisation for the students that have been selected for Remedial and the ones for Non Remedial after calculating the score based on association rules.



The above Bar Graph gives the count of the students depending on the grade point depending on the present year database generated using traditional approach.