目 录

[Python绘图代码 2](#_Toc516494396)

[【matPlotTest.py】 2](#_Toc516494397)

[【nihe.py】 6](#_Toc516494398)

[C++数据清洗代码 9](#_Toc516494399)

[【DataClear.cpp】 9](#_Toc516494400)

[【GearData.cpp】 12](#_Toc516494401)

[Java 逻辑、计算、界面设计代码 14](#_Toc516494402)

[【Mysql\_Connectjava】 14](#_Toc516494403)

[【ReadData.java】 16](#_Toc516494404)

[【SVMReadData.java】 20](#_Toc516494405)

[【TestData.java】 23](#_Toc516494406)

[【Parameter.java】 25](#_Toc516494407)

[【WriterData.java】 31](#_Toc516494408)

[【WriterGearData.java】 34](#_Toc516494409)

[【ZZB\_SVM.java】 37](#_Toc516494410)

[【ZZB\_JCS.java】 39](#_Toc516494411)

[【GUI.java】 52](#_Toc516494412)

[Java 基于Hadoop分布式计算代码 68](#_Toc516494413)

[【hadoopClear.java】 68](#_Toc516494414)

[【ZZP\_Mapper.java】 69](#_Toc516494415)

[【ZZB\_Reducer.java】 70](#_Toc516494416)

# Python绘图代码

## 【matPlotTest.py】

import numpy as np   
import matplotlib.pyplot as plt   
  
##########################三条曲线对照图##########################  
# evenly sampled time at 200ms intervals   
# x = [200,400,800,1000,2000,3000,4000,5000,6000,8000,10000,12000,15000,17500,20000,22500,25000,27500,30000,32500,35000,37500,40000,42500,45000,47500,50000,52500,55000,57500,60000,62500,65000,67500,70000,72500,75000,77500,80000,82500,85000,87500,90000,92500,95000,97500,100000,110000,120000,125000,130000,140000,150000,160000,175000,]  
# y1 = [33.0,30.0,36.0,37.2,31.7,34.0,35.5,37.6,36.1,37.8,40.3,42.4,35.7,40.0,46.1,42.1,42.0,45.1,45.6,48.3,45.6,48.3,48.7,46.8,48.0,48.0,46.8,46.9,48.3,49.6,49.2,48.9,49.0,50.0,51.9,48.9,50.4,53.2,53.3,50.7,50.9,51.4,50.1,51.9,51.6,53.2,52.1,51.2,53.6,51.7,54.7,54.4,53.5,54.4,56.3,]  
# y2 = [61.1,58.5,51.1,54.3,53.0,56.5,54.3,57.1,52.2,57.3,54.6,54.0,54.5,57.0,56.4,56.6,56.2,56.3,56.2,56.7,55.3,56.3,57.5,55.7,57.0,57.0,56.3,55.2,56.1,57.7,56.7,55.8,58.1,57.6,57.4,57.7,57.8,58.4,57.8,57.4,58.2,58.4,56.5,58.2,57.7,59.7,57.5,57.0,58.0,56.0,58.0,57.7,57.7,59.4,58.3,]  
# y3 = [32.4,57.4,49.6,59.5,51.3,51.8,52.7,54.0,51.0,52.7,48.9,49.1,48.6,49.7,49.0,49.6,49.7,50.7,48.6,48.6,48.5,48.7,47.6,48.3,48.5,47.2,47.4,47.2,48.7,46.7,47.6,48.0,46.4,48.0,46.1,47.2,47.1,47.3,48.6,46.2,46.3,45.7,46.1,46.5,46.7,44.9,46.3,47.4,47.2,47.6,47.1,47.8,46.6,45.3,48.5,]   
# plt.plot(x,y1,color = 'red',linestyle = '-',label = 'Accuracy')  
# plt.plot(x,y2,color = 'blue',linestyle = '-.',label = 'Precision')  
# plt.plot(x,y3,color = 'green',linestyle = ':',label = 'Recall')  
# plt.xlabel('Data Amount')  
# plt.ylabel('Percentage (%)')  
# plt.legend(loc='lowwer right')   
# plt.show()   
##########################三条曲线对照图##########################  
  
  
  
##########################查全率、查准率精度变化图##########################  
# x = [3000,4000,5000,6000,8000,10000,12000,15000,17500,20000,22500,25000,27500,30000,32500,35000,37500,40000,42500,45000,47500,50000,52500,55000,57500,60000,62500,65000,67500,70000,72500,75000,77500,80000,82500,85000,87500,90000,92500,95000,97500,100000,110000,120000,125000,130000,140000,150000,160000,175000,]  
# y2 = [56.5,54.3,57.1,52.2,57.3,54.6,54.0,54.5,57.0,56.4,56.6,56.2,56.3,56.2,56.7,55.3,56.3,57.5,55.7,57.0,57.0,56.3,55.2,56.1,57.7,56.7,55.8,58.1,57.6,57.4,57.7,57.8,58.4,57.8,57.4,58.2,58.4,56.5,58.2,57.7,59.7,57.5,57.0,58.0,56.0,58.0,57.7,57.7,59.4,58.3,]  
# y3 = [51.8,52.7,54.0,51.0,52.7,48.9,49.1,48.6,49.7,49.0,49.6,49.7,50.7,48.6,48.6,48.5,48.7,47.6,48.3,48.5,47.2,47.4,47.2,48.7,46.7,47.6,48.0,46.4,48.0,46.1,47.2,47.1,47.3,48.6,46.2,46.3,45.7,46.1,46.5,46.7,44.9,46.3,47.4,47.2,47.6,47.1,47.8,46.6,45.3,48.5,]   
# plt.plot(x,y2,color = 'blue',linestyle = '-.',label = 'Precision')  
# plt.plot(x,y3,color = 'green',linestyle = ':',label = 'Recall')  
# plt.xlabel('Data Amount')  
# plt.ylabel('Percentage (%)')  
# plt.legend(loc='upper left')   
# plt.title('Comparative Analysis',color='black')   
# plt.annotate('Precision', xy=(63000, 55.5), xytext=(70000, 53), arrowprops=dict(facecolor='blue', shrink=0.01), )   
# plt.annotate('Recall', xy=(55000, 49), xytext=(70000, 51), arrowprops=dict(facecolor='green', shrink=0.01), )   
# plt.show()  
##########################查全率、查准率精度变化图##########################  
  
  
  
  
# ##########################P-R图##########################  
# y2 = [61.1,58.5,51.1,54.3,53.0,56.5,54.3,57.1,52.2,57.3,54.6,54.0,54.5,57.0,56.4,56.6,56.2,56.3,56.2,56.7,55.3,56.3,57.5,55.7,57.0,57.0,56.3,55.2,56.1,57.7,56.7,55.8,58.1,57.6,57.4,57.7,57.8,58.4,57.8,57.4,58.2,58.4,56.5,58.2,57.7,59.7,57.5,57.0,58.0,56.0,58.0,57.7,57.7,59.4,58.3,]  
# y3 = [32.4,57.4,49.6,59.5,51.3,51.8,52.7,54.0,51.0,52.7,48.9,49.1,48.6,49.7,49.0,49.6,49.7,50.7,48.6,48.6,48.5,48.7,47.6,48.3,48.5,47.2,47.4,47.2,48.7,46.7,47.6,48.0,46.4,48.0,46.1,47.2,47.1,47.3,48.6,46.2,46.3,45.7,46.1,46.5,46.7,44.9,46.3,47.4,47.2,47.6,47.1,47.8,46.6,45.3,48.5,]   
# y2=y2[20:]  
# y3=y3[20:]  
# plt.plot(y2,y3,color = 'blue',linestyle = ':')  
# plt.xlabel('Precision')  
# plt.ylabel('Recall')  
# plt.legend(loc='lowwer right')   
# plt.show()   
##########################P-R图##########################  
  
  
##########################决策树精度变化图##########################  
# x = [200,400,800,1000,2000,3000,4000,5000,6000,8000,10000,12000,15000,17500,20000,22500,25000,27500,30000,32500,35000,37500,40000,42500,45000,47500,50000,52500,55000,57500,60000,62500,65000,67500,70000,72500,75000,77500,80000,82500,85000,87500,90000,92500,95000,97500,100000,110000,120000,125000,130000,140000,150000,160000,175000,]  
# y1 = [33.0,30.0,36.0,37.2,31.7,34.0,35.5,37.6,36.1,37.8,40.3,42.4,35.7,40.0,46.1,42.1,42.0,45.1,45.6,48.3,45.6,48.3,48.7,46.8,48.0,48.0,46.8,46.9,48.3,49.6,49.2,48.9,49.0,50.0,51.9,48.9,50.4,53.2,53.3,50.7,50.9,51.4,50.1,51.9,51.6,53.2,52.1,51.2,53.6,51.7,54.7,54.4,53.5,54.4,56.3,]  
# y2 = [61.1,58.5,51.1,54.3,53.0,56.5,54.3,57.1,52.2,57.3,54.6,54.0,54.5,57.0,56.4,56.6,56.2,56.3,56.2,56.7,55.3,56.3,57.5,55.7,57.0,57.0,56.3,55.2,56.1,57.7,56.7,55.8,58.1,57.6,57.4,57.7,57.8,58.4,57.8,57.4,58.2,58.4,56.5,58.2,57.7,59.7,57.5,57.0,58.0,56.0,58.0,57.7,57.7,59.4,58.3,]  
# y3 = [32.4,57.4,49.6,59.5,51.3,51.8,52.7,54.0,51.0,52.7,48.9,49.1,48.6,49.7,49.0,49.6,49.7,50.7,48.6,48.6,48.5,48.7,47.6,48.3,48.5,47.2,47.4,47.2,48.7,46.7,47.6,48.0,46.4,48.0,46.1,47.2,47.1,47.3,48.6,46.2,46.3,45.7,46.1,46.5,46.7,44.9,46.3,47.4,47.2,47.6,47.1,47.8,46.6,45.3,48.5,]   
# plt.plot(x,y1,'rs')  
# plt.title('DecisionTree',color='black')  
# plt.xlabel('Data Amount')  
# plt.ylabel('Accuracy (%)')  
# plt.show()   
##########################决策树精度变化图##########################  
  
  
##########################SVM精度变化图##########################  
# x = [200,400,800,1000,2000,3000,4000,5000,6000,8000,10000,12000,15000,17500,20000,22500,25000,27500,30000,32500,35000,37500,40000,42500,45000,]  
# y1 = [55.0,49.5,51.0,56.0,54.3,53.9,54.2,55.4,56.8,55.9,56.1,55.2,56.1,56.8,56.4,57.4,56.8,56.8,57.4,57.6,56.7,58.1,57.6,57.2,58.4,]  
# plt.plot(x,y1,'rs')  
# plt.title('Support Vector Machine',color='black')  
# plt.xlabel('Data Amount')  
# plt.ylabel('Accuracy (%)')  
# # plt.plot(x,y1,color = 'red',linestyle = '-',label = 'Accuracy')  
# plt.show()   
##########################SVM精度变化图##########################  
  
  
  
##########################精度变化对比分析图##########################  
# x = [200,400,800,1000,2000,3000,4000,5000,6000,8000,10000,12000,15000,17500,20000,22500,25000,27500,30000,32500,35000,37500,40000,42500,45000,]  
# y1 = [55.0,49.5,51.0,56.0,54.3,53.9,54.2,55.4,56.8,55.9,56.1,55.2,56.1,56.8,56.4,57.4,56.8,56.8,57.4,57.6,56.7,58.1,57.6,57.2,58.4,]  
# x1 = [200,400,800,1000,2000,3000,4000,5000,6000,8000,10000,12000,15000,17500,20000,22500,25000,27500,30000,32500,35000,37500,40000,42500,45000,47500,50000,52500,55000,57500,60000,62500,65000,67500,70000,72500,75000,77500,80000,82500,85000,87500,90000,92500,95000,97500,100000,110000,120000,125000,130000,140000,150000,160000,175000,]  
# y2 = [33.0,30.0,36.0,37.2,31.7,34.0,35.5,37.6,36.1,37.8,40.3,42.4,35.7,40.0,46.1,42.1,42.0,45.1,45.6,48.3,45.6,48.3,48.7,46.8,48.0,48.0,46.8,46.9,48.3,49.6,49.2,48.9,49.0,50.0,51.9,48.9,50.4,53.2,53.3,50.7,50.9,51.4,50.1,51.9,51.6,53.2,52.1,51.2,53.6,51.7,54.7,54.4,53.5,54.4,56.3,]  
# s1 = []  
# count = 0  
# sy1 = []  
# while count<len(x):  
# s1.append(x1[count])  
# sy1.append(y2[count])  
# count+=1  
# plt.plot(x,y1,'r',s1,sy1,'b')  
# plt.title('Comparative Analysis',color='black')   
# plt.annotate('SVM', xy=(12000, 54.5), xytext=(15000, 50), arrowprops=dict(facecolor='red', shrink=0.01), )   
# plt.annotate('DecisionTree', xy=(17000, 38), xytext=(20000, 33), arrowprops=dict(facecolor='blue', shrink=0.01), )   
# plt.xlabel('Data Amount')  
# plt.ylabel('Accuracy (%)')  
# # plt.plot(x,y1,color = 'red',linestyle = '-',label = 'Accuracy')  
# plt.show()   
##########################精度变化对比分析图##########################

## 【nihe.py】

#coding:utf-8  
import numpy  
from scipy import log  
from scipy.optimize import curve\_fit  
import matplotlib.pyplot as plt  
import time   
import matplotlib.font\_manager as fm  
  
myfont = fm.FontProperties(fname='zui.ttf')  
  
  
def func(x, a, b):  
 y = a \* log(x) + b  
 return y  
  
def polyfit(x, y, degree):  
 results = {}  
 popt, pcov = curve\_fit(func, x, y)  
 results['polynomial'] = popt  
  
 # r-squared  
 yhat = func(x ,popt[0] ,popt[1] ) # or [p(z) for z in x]  
 ybar = numpy.sum(y)/len(y) # or sum(y)/len(y)  
 ssreg = numpy.sum((yhat-ybar)\*\*2) # or sum([ (yihat - ybar)\*\*2 for yihat in yhat])  
 sstot = numpy.sum((y - ybar)\*\*2) # or sum([ (yi - ybar)\*\*2 for yi in y])  
 results['determination'] = ssreg / sstot  
 return results  
  
  
x = [200,400,800,1000,2000,3000,4000,5000,6000,8000,10000,12000,15000,17500,20000,22500,25000,27500,30000,32500,35000,37500,40000,42500,45000,47500,50000,52500,55000,57500,60000,62500,65000,67500,70000,72500,75000,77500,80000,82500,85000,87500,90000,92500,95000,97500,100000,110000,120000,125000,130000,140000,150000,160000,175000,]  
y = [33.0,30.0,36.0,37.2,31.7,34.0,35.5,37.6,36.1,37.8,40.3,42.4,35.7,40.0,46.1,42.1,42.0,45.1,45.6,48.3,45.6,48.3,48.7,46.8,48.0,48.0,46.8,46.9,48.3,49.6,49.2,48.9,49.0,50.0,51.9,48.9,50.4,53.2,53.3,50.7,50.9,51.4,50.1,51.9,51.6,53.2,52.1,51.2,53.6,51.7,54.7,54.4,53.5,54.4,56.3,]  
z0 = polyfit(x, y, 2)  
  
  
  
x1 = [200,400,800,1000,2000,3000,4000,5000,6000,8000,10000,12000,15000,17500,20000,22500,25000,27500,30000,32500,35000,37500,40000,42500,45000,]  
y1 = [55.0,49.5,51.0,56.0,54.3,53.9,54.2,55.4,56.8,55.9,56.1,55.2,56.1,56.8,56.4,57.4,56.8,56.8,57.4,57.6,56.7,58.1,57.6,57.2,58.4,]  
z1 = polyfit(x1, y1, 2)  
def showDynamic(dx,dy):  
 s = []  
 for i in x:  
 s.append(i)  
 for i in range(100):  
 s.append(x[-1]+i\*dx)  
 yvals=func(s,z0['polynomial'][0],z0['polynomial'][1])  
  
 s1 = []  
 for i in x1:  
 s1.append(i)  
 for i in range(100):  
 s1.append(x1[-1]+i\*dy)  
 yvals1 = func(s1,z1['polynomial'][0],z1['polynomial'][1])  
  
  
 plot1=plt.plot(x, y, '\*',label='Initial data point')  
 plot2=plt.plot(s, yvals, 'r',label='Fitting curve')  
 plt.legend(loc='upper left', prop = myfont)   
  
 plot3=plt.plot(x1, y1, '\*')  
 plot4=plt.plot(s1, yvals1, 'b')  
  
 plt.xlabel('Amount')  
 plt.ylabel('Accuracy (%)')  
 plt.legend(loc=4) #指定legend的位置,读者可以自己help它的用法  
 plt.title('曲线拟合',fontproperties = myfont)  
 plt.annotate('SVM', xy=(12000, 54.5), xytext=(24000, 50), arrowprops=dict(facecolor='blue', shrink=0.01), )   
 plt.annotate('DecisionTree', xy=(17000, 38), xytext=(30000, 33), arrowprops=dict(facecolor='red', shrink=0.01), )   
 plt.show()  
 plt.close()  
  
for i in range(10):  
 dx = pow(2,i+5)  
 dy = dx + (x[-1] - x1[-1])//100;  
 showDynamic(dx,dy)  
 time.sleep(0.1)

# C++数据清洗代码

## 【DataClear.cpp】

**#include**<iostream>  
**#include**<string>  
**#include**<fstream>  
**using namespace** std;  
  
**int** main()  
{  
 **int** count=0;  
 **float** attr[34];  
 ifstream in(**"/Users/zhangzhaobo/Documents/Graduation-Design/Mydata.txt"**);  
 ofstream out(**"/Users/zhangzhaobo/Documents/Graduation-Design/Data/New\_Data.txt"**);  
 string line[7]={**"Diff\_X"**,**"Diff\_Y"**,**"Pixels\_Areas"**,**"Diff\_Luminosity"**,**"TypeOfSteel"**,**"Steel\_Plate\_Thickness"**,**"Fault"**};  
 **for**(**int** i=0;i<7;i++)  
 {  
 out<<line[i]<<**"\t\t"**;  
 }  
 out<<endl;  
 **for** (**int** i = 0; i < 34; ++i)  
 {  
 in>>line[1];  
 */\* code \*/* }  
 **while**(count<1941)  
 {  
 **for** (**int** i = 0; i < 34; ++i)  
 {  
 in>>attr[i];  
 }  
 **int** X\_dis=attr[1]-attr[0];  
 **int** Y\_dis=attr[3]-attr[2];  
 **int** Area\_Range=attr[4];  
 **int** Luminosity\_dis=attr[9]-attr[8];  
 **int** TypeOfSteel=attr[11];  
 **int** Steel\_Plate\_Thickness=attr[13];  
 **if** (X\_dis<100)  
 {  
 X\_dis=X\_dis/10;  
 }  
 **else**{  
 X\_dis=10+X\_dis/100;  
 }  
 **if** (Y\_dis<100)  
 {  
 Y\_dis=Y\_dis/10;  
 }  
 **else**{  
 **if** (Y\_dis<1000)  
 {  
 Y\_dis=10+Y\_dis/100;  
 }  
 **else** {  
 Y\_dis=20+Y\_dis/1000;  
 }  
 }  
 Luminosity\_dis=Luminosity\_dis/5;  
 **if** (Area\_Range<100)  
 {  
 Area\_Range=Area\_Range/10;  
 }  
 **else** {  
 **if** (Area\_Range<1000)  
 {  
 Area\_Range=10+Area\_Range/100;  
 }  
 **else** {  
 Area\_Range=20+Area\_Range/1000;  
 }  
 }  
 out<<X\_dis<<**"\t\t"**<<Y\_dis<<**"\t\t"**<<Area\_Range<<**"\t\t"**<<Luminosity\_dis<<**"\t\t"**<<TypeOfSteel<<**"\t\t"**<<Steel\_Plate\_Thickness<<**"\t\t"**;  
 **int** Fault=0;  
 **for** (**int** i = 0; i < 7; ++i)  
 {  
 **if** (attr[i+27]!=0)  
 {  
 Fault=i;  
 }  
  
 }  
 out<<Fault<<endl;  
 count++;  
 }  
 in.close();  
 **return** 0;  
}

## 【GearData.cpp】

**#include**<iostream>  
**#include**<string>  
**#include**<fstream>  
**#include**<string>  
**#include** <iomanip>  
**using namespace** std;  
  
**int** main()  
{  
 **float** data[4];  
 string hz[10]={**"0"**,**"10"**,**"20"**,**"30"**,**"40"**,**"50"**,**"60"**,**"70"**,**"80"**,**"90"**};  
 string file;  
 ofstream out(**"/Users/zhangzhaobo/Documents/Graduation-Design/Data/BrokenTooth Data/New\_b30hz.txt"**);  
 **for** (**int** i = 0; i < 10; ++i)  
 {  
 file=**"/Users/zhangzhaobo/Documents/Graduation-Design/Data/BrokenTooth Data/b30hz"**+hz[i]+**".txt"**;  
 ifstream in(file);  
 **while**(in>>data[0])  
 {  
 out<<setprecision(2)<<data[0]<<**"\t\t"**;  
 **for** (**int** i = 1; i < 4; ++i)  
 {  
 in>>data[i];  
 out<<setprecision(2)<<data[i]<<**"\t\t"**;  
 }  
 out<<endl;  
 }  
 cout<<file<<**" is done!"**<<endl;  
 in.close();  
 }  
 out.close();  
 ofstream out1(**"/Users/zhangzhaobo/Documents/Graduation-Design/Data/Healthy Data/New\_h30hz.txt"**);  
 **for** (**int** i = 0; i < 10; ++i)  
 {  
 file=**"/Users/zhangzhaobo/Documents/Graduation-Design/Data/Healthy Data/h30hz"**+hz[i]+**".txt"**;  
 ifstream in(file);  
 **while**(in>>data[0])  
 {  
 out1<<setprecision(2)<<data[0]<<**"\t\t"**;  
 **for** (**int** i = 1; i < 4; ++i)  
 {  
 in>>data[i];  
 out1<<setprecision(2)<<data[i]<<**"\t\t"**;  
 }  
 out1<<endl;  
 }  
 cout<<file<<**" is done!"**<<endl;  
 in.close();  
 }  
 out1.close();  
 **return** 0;  
}

# Java 逻辑、计算、界面设计代码

## 【Mysql\_Connectjava】

*/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 \* Author : HustWolf --- 张照博  
  
 \* Time : 2018.1-2018.5  
  
 \* Address : HUST  
  
 \* Version : 1.0  
  
 \* 封装数据库连接，会话，断开三个功能  
 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/***import** java.sql.Connection;  
**import** java.sql.DriverManager;  
**import** java.sql.SQLException;  
**import** java.sql.Statement;  
  
**public class** Mysql\_Connect {  
  
 *//此处查看网络才知道。要求SSL，所以就酱紫咯：https://zhidao.baidu.com/question/2056521203295428667.html* **private static** String url = **"jdbc:mysql://127.0.0.1:3306/Graduation\_Design?useUnicode=true&characterEncoding=GBK&useSSL=true"**;  
*// private static String url = "jdbc:mysql://159.203.250.111:3306/GD?useUnicode=true&characterEncoding=GBK&useSSL=true";* **private static** String user = **"root"**;  
  
 **private static** String password = **"zzb1184827350"**;  
*// private static String password = "zzb162122";* **private** Statement statement;  
  
 **private** Connection conn;  
  
 **public** Connection getConnection() { **return this**.conn;}  
  
 **public** Statement getStatement() {  
 **try** {  
 **this**.statement = **this**.conn.createStatement();;  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 }  
 **return this**.statement;  
 }  
  
 **public void** Connect() {  
 **try** {  
 String driver = **"com.mysql.jdbc.Driver"**;  
 Class.forName(driver);  
 conn = DriverManager.getConnection(url, user, password);  
  
 **if** (!conn.isClosed()){  
*// System.out.println("Connect Success!");* }  
 **else** {  
 System.out.println(**"\n\nFailed to connect to the Database!"**);  
 }  
*// this.statement = conn.createStatement();* } **catch** (ClassNotFoundException e) {  
 System.out.println(**"Sorry,can`t find the Driver!"**);  
 e.printStackTrace();  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 }  
 }  
 **public void** Dis\_Connect() **throws** SQLException {  
 **try** {  
 conn.close();  
 } **catch** (Exception e) {  
  
 e.printStackTrace();  
  
 }  
 }  
}

## 【ReadData.java】

*/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 \* Author : HustWolf --- 张照博  
  
 \* Time : 2018.1-2018.5  
  
 \* Address : HUST  
  
 \* Version : 6.1  
  
 \* 从数据库读取数据，并且从ReadData这个函数传出去！5.0加了训练集和验证集的划分  
  
 \* 第六代提供了离散化方式并且增加了属性划分区间这一变量  
  
 \* 在原有基础上，增添了可接受Parmeter设定这一条  
 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/  
  
  
/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\* 这是从数据库或者是文本文件读取数据的时候用的  
\* 其实我觉得如果可以每一次读一条数据，然后处理一条会比较好  
\* 但是算了，数据量不大的话，这个样子也不会增加太多时间的！  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/***import** java.sql.ResultSet;  
**import** java.sql.SQLException;  
**import** java.sql.Statement;  
**import** java.text.NumberFormat;  
**import** java.util.ArrayList;  
**import** java.util.List;  
  
**public class** ReadData {  
 **private** Object[] Name;  
 **private** Mysql\_Connect mysql = **new** Mysql\_Connect();  
 ArrayList<List<Float>> range = **new** ArrayList<>();  
 **public** NumberFormat nf = NumberFormat.getNumberInstance();  
  
 ReadData() {  
 nf.setMaximumFractionDigits(1);  
 Name = **new** Object[]{**"Sensor1"**, **"Sensor2"**, **"Sensor3"**, **"Sensor4"**, **"Load"**, **"category"**};  
 }  
  
 **public static** String getSelectQuery(Object[] Name, String table, **int** id) {  
 String select = **"SELECT "**;  
 **for** (**int** i = 0; i < Name.length - 1; ++i) {  
 select += (**"`"** + (Name[i] + **"`"** + **","**));  
 }  
 select += (**"`"** + Name[Name.length - 1] + **"`"**);  
 select += **" from "** + table + **" where id = "** + id;  
*// System.out.println(select);* **return** select;  
 }  
  
 **public** Object[][] readTrainData(Parameter par) {  
 **try** {  
 mysql.Connect();  
 Statement statement = mysql.getStatement();  
 **int** columnCount = par.getTrainNum();  
 **float**[][] dataToTrain;  
 dataToTrain = **new float**[columnCount][Name.length];  
 **for** (**int** i = 0; i < columnCount; ++i) {  
 String getDataQuery = getSelectQuery(Name, **"gear"**, i \* par.getTrainDistance());  
 ResultSet select\_ok;  
 select\_ok = statement.executeQuery(getDataQuery);  
 select\_ok.next();  
 **for** (**int** j = 0; j < Name.length; ++j) {  
 dataToTrain[i][j] = Float.parseFloat(nf.format(select\_ok.getFloat((String) Name[j])));  
 }  
 }  
 **if** (!range.isEmpty()){  
 range.clear();  
 }  
 range = par.EADC(dataToTrain);  
 Object[][] re = **new** Object[columnCount][Name.length];  
 **for** (**int** valueindex = 0; valueindex < Name.length - 1; ++valueindex) {  
 **for** (**int** i = 0; i < dataToTrain.length; ++i) {  
 **for** (**int** x = 0; x < range.get(valueindex).size(); ++x) {  
 **if** (dataToTrain[i][valueindex] > range.get(valueindex).get(x) && dataToTrain[i][valueindex] <= range.get(valueindex).get(x + 1)) {  
 re[i][valueindex] = (**"|"** + range.get(valueindex).get(x) + **"<X≤"** + range.get(valueindex).get(x + 1) + **"|"**);  
 **break**;  
 }  
 }  
 }  
 }  
 **for** (**int** i = 0; i < dataToTrain.length; ++i) {  
 re[i][Name.length - 1] = dataToTrain[i][Name.length - 1];  
 }  
 statement.close();  
 mysql.Dis\_Connect();  
 **return** re;  
 } **catch** (SQLException e) {  
 e.printStackTrace();  
 }  
 **return new** Object[1][1];  
 }  
  
 **public** Object[][] readTestData(Parameter par) {  
 **try** {  
 mysql.Connect();  
 Statement statement = mysql.getStatement();  
 **int** columnCount = par.getTestNum();  
 **float**[][] dataToTest;  
 dataToTest = **new float**[columnCount][Name.length];  
 **for** (**int** i = 0; i < columnCount; ++i) {  
 String getDataQuery = getSelectQuery(Name, **"gear"**, i \* par.getTestDistance() + 1);  
 ResultSet select\_ok;  
 select\_ok = statement.executeQuery(getDataQuery);  
 select\_ok.next();  
 **for** (**int** j = 0; j < Name.length; ++j) {  
 dataToTest[i][j] = Float.parseFloat(nf.format(select\_ok.getFloat((String) Name[j])));  
 }  
 }  
*// for (List<Float> x:range) {  
// System.out.println(x.size());  
// for (float s:x){  
// System.out.print(s+"-->");  
// }  
// }* **if** (range.size() != 0) {  
 Object[][] re = **new** Object[columnCount][range.size() + 1];  
 **for** (**int** valueindex = 0; valueindex < range.size(); ++valueindex) {  
 **for** (**int** i = 0; i < dataToTest.length; ++i) {  
 **for** (**int** x = 0; x < range.get(valueindex).size(); ++x) {  
 **if** (dataToTest[i][valueindex] > range.get(valueindex).get(x) && dataToTest[i][valueindex] <= range.get(valueindex).get(x + 1)) {  
 re[i][valueindex] = (**"|"** + range.get(valueindex).get(x) + **"<X≤"** + range.get(valueindex).get(x + 1) + **"|"**);  
 **break**;  
 }  
 }  
 }  
 }  
 **for** (**int** i = 0; i < dataToTest.length; ++i) {  
 re[i][Name.length - 1] = dataToTest[i][Name.length - 1];  
 *//便利旧集合没有就添加到新集合* }  
 statement.close();  
 mysql.Dis\_Connect();  
 **return** re;  
 }  
  
 } **catch** (SQLException e) {  
 e.printStackTrace();  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 }  
 **return new** Object[1][1];  
 }  
}

## 【SVMReadData.java】

*/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 \* Author : HustWolf --- 张照博  
  
 \* Time : 2018.1-2018.5  
  
 \* Address : HUST  
  
 \* Version : 2.5  
  
 \* 从数据库中读取符合SVM格式的数据，增加精度至0.001  
  
 \* 2.5 版本新增了Parmeter，可以自由调节！  
  
 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/***import** java.io.\*;  
**import** java.sql.ResultSet;  
**import** java.sql.SQLException;  
**import** java.sql.Statement;  
**import** java.text.NumberFormat;  
  
**public class** SVMReadData {  
 **private** Object[] Name;  
 **protected** Mysql\_Connect mysql=**new** Mysql\_Connect();  
 **private static int** atoi(String s) {  
 **return** Integer.parseInt(s);  
 }  
 SVMReadData() {  
 Name = **new** Object[]{**"Sensor1"**,**"Sensor2"**,**"Sensor3"**,**"Sensor4"**, **"Load"**, **"category"**};  
 }  
  
 **public** String readTrainData(Parameter par) {  
 **try** {  
 mysql.Connect();  
 Statement statement=mysql.getStatement();  
 NumberFormat nf = NumberFormat.getNumberInstance();  
 nf.setMaximumFractionDigits(3);  
 FileWriter svmTrainData = **new** FileWriter(**"svmTrainData.txt"**);  
 **int** columnCount = par.getTrainNum();  
 Object[][] DataTrain;  
 DataTrain = **new** Object[columnCount][Name.length];  
 **for** (**int** i = 0;i<columnCount;++i) {  
 String getDataQuery = ReadData.getSelectQuery(Name,**"gear"**,i \* par.getTrainDistance());  
 ResultSet select\_ok;  
 select\_ok = statement.executeQuery(getDataQuery);  
 select\_ok.next();  
 **for** (**int** j = 0; j < Name.length; ++j){  
*// DataTrain[i][j]=Float.parseFloat(nf.format(select\_ok.getFloat((String) Name[j])));* DataTrain[i][j]=select\_ok.getFloat((String) Name[j]);  
  
 }  
 **if**((Float)DataTrain[i][DataTrain[0].length-1] == 1.0)  
 svmTrainData.write(**"1 "**);  
 **else** svmTrainData.write(**"0 "**);  
 **for** (**int** j = 0; j<DataTrain[0].length-1; ++j ){  
 svmTrainData.write(j+**":"**+DataTrain[i][j]+**" "**);  
 }  
 svmTrainData.write(**"\n"**);  
 }  
 statement.close();  
 mysql.Dis\_Connect();  
 svmTrainData.close();  
 **return "svmTrainData.txt"**;  
 } **catch** (SQLException e) {  
 e.printStackTrace();  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 }  
 **return "svmTrainData.txt"**;  
 }  
  
 **public** String readTestData(Parameter par) {  
 **try** {  
 mysql.Connect();  
 Statement statement=mysql.getStatement();  
 NumberFormat nf = NumberFormat.getNumberInstance();  
 nf.setMaximumFractionDigits(3);  
 FileWriter svmTestData = **new** FileWriter(**"svmTestData.txt"**);  
 **int** columnCount = par.getTestNum();  
 Object[][] DataTest;  
 DataTest = **new** Object[columnCount][Name.length];  
 **for** (**int** i = 0;i<columnCount;++i) {  
 String getDataQuery = ReadData.getSelectQuery(Name,**"gear"**,i \* par.getTestDistance()+1);  
 ResultSet select\_ok;  
 select\_ok = statement.executeQuery(getDataQuery);  
 select\_ok.next();  
 **for** (**int** j = 0; j < Name.length; ++j){  
*// DataTest[i][j]=Float.parseFloat(nf.format(select\_ok.getFloat((String) Name[j])));* DataTest[i][j]=select\_ok.getFloat((String) Name[j]);  
 }  
 **if**((Float)DataTest[i][DataTest[0].length-1] == 1)  
 svmTestData.write(**"1 "**);  
 **else** svmTestData.write(**"0 "**);  
 **for** (**int** j = 0; j<DataTest[0].length-1; ++j ){  
 svmTestData.write(j+**":"**+DataTest[i][j]+**" "**);  
 }  
 svmTestData.write(**"\n"**);  
 }  
 statement.close();  
 mysql.Dis\_Connect();  
 svmTestData.close();  
 **return "svmTestData.txt"**;  
 } **catch** (SQLException e) {  
 e.printStackTrace();  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 }  
 **return "svmTestData.txt"**;  
 }  
}

## 【TestData.java】

*/\*  
 \* Author : HustWolf --- 张照博  
  
 \* Time : 2018.1-2018.5  
  
 \* Address : HUST  
  
 \* Version : 2.5  
  
 \* 输入测试数据得到故障模型的预测结果  
  
 \* @param decisionTree 决策树  
  
 \* @param Attr\_Name 特征列表  
  
 \* @param TestData 测试数据  
 \*/***public class** TestData{  
 **public static** String getCategory(Object number){  
 **if** ((Float)number == 0){  
 **return "0"**;  
 }  
 **else return "1"**;  
 }  
  
 **public static** String TestData(Object obj, Object[] Attr\_Name, Object[] TestData,String line) {  
 **if** (obj **instanceof** ZZB\_JCS.Tree && Attr\_Name.length>0){  
 ZZB\_JCS.Tree tree = (ZZB\_JCS.Tree) obj;  
 String attribute\_Name = tree.getAttribute();  
 Object[] Attr\_Not\_Used = **new** Object[Attr\_Name.length-1];  
 Object[] Data\_Not\_Used = **new** Object[Attr\_Name.length-1];  
 Object testvalue = TestData[0];  
 **for** (**int** i=0,j=0;i<Attr\_Name.length ;++i ) {  
 **if** ( attribute\_Name != Attr\_Name[i] ) {  
 Data\_Not\_Used[j] = TestData[i];  
 Attr\_Not\_Used[j++] = Attr\_Name[i];  
 }  
 **else** {  
 testvalue = TestData[i];  
 }  
 }  
 **boolean** flag=**false**;  
 **for** (Object attrValue : tree.getAttributeValues()){  
 **if** ((testvalue.equals(attrValue))) {  
 Object child = tree.getChild(attrValue);  
 flag=**true**;  
 line=TestData(child,Attr\_Not\_Used,Data\_Not\_Used,line);  
 }  
 }  
 **if** (!flag){  
 line=**"Sorry, we don't find this data, maybe it's OK!"**;  
*// System.out.println("Sorry, we don't find this data, maybe it's OK!");* }  
 }**else** {  
 line=**"The Category of this Data is:"**+getCategory(obj);  
*// System.out.println("The Category of this Data is:"+getCategory(obj));* }  
 **return** line;  
 }  
}

## 【Parameter.java】

*/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 \* Author : HustWolf --- 张照博  
  
 \* Time : 2018.1-2018.5  
  
 \* Address : HUST  
  
 \* Version : 1.0  
  
 \* 定义一些静态的数值，并且提供getter  
 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/***import** java.text.NumberFormat;  
**import** java.util.\*;  
  
**class** Alone\_Value\_Category **implements** Comparable<Alone\_Value\_Category>{  
 **private float** sensor;  
 **private float** category;  
 *// private float[] range = new float[2];* Alone\_Value\_Category(**float** a, **float** b){  
 **super**();  
 **this**.sensor = a;  
 **this**.category = b;  
 }  
  
 **float** getSensor(){  
 **return** sensor;  
 }  
 **float** getCategory(){  
 **return** category;  
 }  
 *// void setRange(float a, float b){  
// range[0] = a;  
// range[1] =b;  
// }* @Override  
 **public** String toString() {  
 **return "\n[Sensor:"** + sensor + **", category="** + category + **"]"**;  
 }  
 @Override  
 **public int** compareTo(Alone\_Value\_Category o) {  
 **return** Float.compare(**this**.sensor,o.sensor);  
 }  
}  
  
**class** Interval{  
 **private float** top;  
 **private float** bottom;  
 **public** Map<Float,List<Alone\_Value\_Category> > sample = **new** HashMap<Float, List<Alone\_Value\_Category>>();  
 Interval(){};  
 Interval(Interval b){  
 top = b.top;  
 bottom = b.bottom;  
 sample = b.sample;  
 }  
 Interval(**float** a, **float** b, **float** c, List<Alone\_Value\_Category> d){  
 **this**.top = a;  
 **this**.bottom = b;  
 sample.put(c,d);  
 }  
  
 **public float** getTop() {  
 **return** top;  
 }  
  
 **public float** getBottom() {  
 **return** bottom;  
 }  
  
 **public void** setTop(**float** top) {  
 **this**.top = top;  
 }  
  
 **public void** setBottom(**float** bottom) {  
 **this**.bottom = bottom;  
 }  
  
 **public void** setSample(Map<Float, List<Alone\_Value\_Category>> sample) {  
 **this**.sample = sample;  
 }  
  
 **public** Interval addTmp(Interval b){  
 Interval re = **new** Interval(b);  
 **if** (top>b.top) re.setTop(top);  
 **else** re.setTop(b.top);  
 **if** (bottom<b.bottom) re.setBottom(bottom);  
 **else** re.setBottom(b.bottom);  
 re.sample.putAll(sample);  
 **return** re;  
 }  
 **public void** merge(Interval b){  
 **if** (top<b.top)  
 top = b.top;  
 **if** (bottom>b.bottom)  
 bottom = b.bottom;  
 sample.putAll(b.sample);  
 }  
 **public int** getCount(){  
 **int** count = 0;  
 **for**(List<Alone\_Value\_Category> s:sample.values()){  
 count+=s.size();  
 }  
 **return** count;  
 }  
 @Override  
 **public** String toString() {  
 **return "bottom:"**+bottom+**" top:"**+top+**" size:"**+getCount();  
 }  
}  
**public class** Parameter {  
 **private static int** rate;  
 **private static int** trainNum;  
 **private static int** testNum;  
 Parameter(){  
 rate = 2;  
 trainNum = 10000;  
 testNum = trainNum/rate;  
 }  
 **public int** getTrainNum(){  
 **return** trainNum;  
 }  
 **public int** getTestNum(){  
 **return** testNum;  
 }  
 **public int** getTestDistance(){  
 **return** 2000000/testNum;  
 }  
 **public int** getTrainDistance(){  
 **return** 2000000/trainNum;  
 }  
 **public void** setTrainNum(**int** t){  
 trainNum = t;  
 testNum = trainNum / rate;  
 }  
 **public void** setTestNum(**int** t){  
 testNum = t;  
 trainNum = testNum \* rate;  
 }  
  
  
  
 **void** Clear(ArrayList<Interval> allInterval){  
 ArrayList<Interval> del = **new** ArrayList<>();  
 **for** (**int** s = 0;s<allInterval.size();++s) {  
 **if** (allInterval.get(s).getCount() == 0){  
 **if** (s>0) {  
 allInterval.get(s - 1).merge(allInterval.get(s));  
 del.add(allInterval.get(s));  
 }  
 **continue**;  
 }  
 }  
 allInterval.removeAll(del);  
 }  
 **double** Entropy(ArrayList<Interval> set, **int** size){  
 **double** shang = 0;  
 NumberFormat nf = NumberFormat.getNumberInstance();  
 nf.setMaximumFractionDigits(4);  
 **for** (Interval x:set){  
 **double** p =(**double**)x.getCount()/(**double**)size;  
 shang -= p\*(Math.log(p)/Math.log(2));  
 }  
 **return** Double.parseDouble(nf.format(shang));  
 }  
  
 **public** ArrayList<List<Float>> EADC(**float**[][] dat) {  
 ArrayList<List<Float>> re = **new** ArrayList<>();  
 **for** (**int** valueindex = 0; valueindex< dat[0].length-1;++valueindex) {  
 ArrayList<Alone\_Value\_Category> LIST = **new** ArrayList<>();  
 **for** (**int** i = 0; i < dat.length; ++i) {  
 LIST.add(**new** Alone\_Value\_Category(dat[i][valueindex], dat[i][dat[valueindex].length - 1]));  
 *//便利旧集合没有就添加到新集合* }  
 Collections.sort(LIST);  
 **float** len = LIST.get(LIST.size() - 1).getSensor() - LIST.get(0).getSensor();  
 **int** k = 40;  
 **float** gap = (len + 1) / k;  
 **float** Lowest = LIST.get(0).getSensor() - 0.50f;  
 **float** Highest = LIST.get(LIST.size()-1).getSensor() + 0.50f;  
 NumberFormat nf = NumberFormat.getNumberInstance();  
 nf.setMaximumFractionDigits(1);  
 List<Float> range = **new** LinkedList<>();  
 **for** (**int** x = 0; x <= k; ++x) {  
 range.add(Float.parseFloat(nf.format(Lowest + x \* gap)));  
 }  
 ArrayList<Interval> allInterval = **new** ArrayList<>();  
 **for** (**int** i = 0; i < k; ++i) {  
 Interval newarea = **new** Interval();  
 newarea.setBottom(range.get(i));  
 newarea.setTop(range.get(i + 1));  
 **for** (Alone\_Value\_Category s : LIST) {  
 **if** (s.getSensor() > range.get(i) && s.getSensor() < range.get(i + 1)) {  
 **if** (!newarea.sample.containsKey(s.getCategory())) {  
 newarea.sample.put(s.getCategory(), **new** LinkedList<>());  
 }  
 newarea.sample.get(s.getCategory()).add(s);  
 }  
 }  
 allInterval.add(newarea);  
 }  
 **int** size = 0;  
 Clear(allInterval);  
 **for** (Interval s : allInterval) {  
 size += s.getCount();  
 }  
 k = allInterval.size();  
 **int** k0 = k;  
 **double** Ck0 = 0;  
 **boolean** Loop = **true**;  
 **double** Hpk\_1 = 0;  
 **while** (Loop && k >= 10) {  
 **double** minD = 1000;  
 **int** mergePoint = 0;  
 **double** Hp0 = Entropy(allInterval, size);  
 **double** Hpk;  
 ArrayList<Interval> newA = **new** ArrayList<>();  
 **for** (**int** i = 0; i < allInterval.size() - 1; ++i) {  
 newA.addAll(allInterval);  
 newA.get(i).merge(newA.get(i + 1));  
 newA.remove(i + 1);  
 Hpk = Entropy(newA, size);  
 **if** (Hpk - Hp0 < minD) {  
 Hpk\_1 = Hpk;  
 minD = Hpk - Hp0;  
 mergePoint = i;  
 }  
 newA.clear();  
 }  
 allInterval.get(mergePoint).merge(allInterval.get(mergePoint + 1));  
 allInterval.remove(mergePoint + 1);  
 **double** Ck\_1 = (k0 - 1) \* Hpk\_1 - Hp0 \* (k - 2);  
 **if** (Ck\_1 > Ck0) {  
 --k;  
 } **else** {  
 Loop = **false**;  
 --k;  
 }  
 Ck0 = Ck\_1;  
 }  
 range.clear();  
 range.add(-100f);  
 **for** (Interval s:allInterval) {  
 range.add(s.getTop());  
 }  
 range.add(100f);  
 re.add(range);  
*// long endTime=System.currentTimeMillis(); //获取结束时间  
// System.out.println("\n程序运行时间： "+(endTime-startTime)+"ms");* }  
 **return** re;  
 }  
}

## 【WriterData.java】

*/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 \* Author : HustWolf --- 张照博  
  
 \* Time : 2018.1-2018.5  
  
 \* Address : HUST  
  
 \* Version : 2.0  
  
 \* 写数据到数据库中去，经过清洗的数据！  
 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/***import** java.io.\*;  
**import** java.sql.SQLException;  
**import** java.sql.Statement;  
  
**public class** WriteData {  
  
 **protected** Mysql\_Connect mysql=**new** Mysql\_Connect();  
  
 **public** String getInsertQuery(**int** id,Object[] name,Object[] value){  
 String insert=**"INSERT INTO steelplate(id"**;  
 **for** (**int** i=0;i<name.length;++i){  
 insert+=(**","**+name[i]);  
 }  
 insert+=**") values("**+id;  
 **for** (**int** i=0;i<value.length;++i){  
 insert+=(**",'"**+value[i]+**"'"**);  
 }  
 insert+=**");"**;  
 **return** insert;  
 }  
  
 **public void** WriteData() **throws** IOException {  
 **try** {  
 mysql.Connect();  
 Statement statement = mysql.getStatement();  
 String DELETE = **"delete from steelplate where id<100000"**;  
 **boolean** delete\_ok = statement.execute(DELETE);  
 **if** (delete\_ok) {  
 System.out.println(**"Have Fun! Boys!\n\n"**);  
 }  
 statement.close();  
 mysql.Dis\_Connect();  
 } **catch** (SQLException e) {  
 e.printStackTrace();  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 }  
 File file = **new** File(**"/Users/zhangzhaobo/IdeaProjects/Graduation\_Design/src/New\_Data.txt"**);  
 BufferedReader in = **new** BufferedReader(**new** FileReader(file));  
 String line; *//一行数据  
 //逐行读取，并将每个数组放入到数组中* line=in.readLine();  
 **int** id=0;  
 Object[] Name = line.split(**"\t\t"**);  
 **try** {  
 mysql.Connect();  
 Statement statement=mysql.getStatement();  
*// Object[] name = new Object[] {"X\_Minimum","X\_Maximum","Y\_Minimum","Y\_Maximum","Pixels\_Areas","X\_Perimeter","Y\_Perimeter","Sum\_of\_Luminosity","Minimum\_of\_Luminosity","Maximum\_of\_Luminosity","Length\_of\_Conveyer","TypeOfSteel\_A300","TypeOfSteel\_A400","Steel\_Plate\_Thickness","Edges\_Index","Empty\_Index","Square\_Index","Outside\_X\_Index","Edges\_X\_Index","Edges\_Y\_Index","Outside\_Global\_Index","LogOfAreas","Log\_X\_Index","Log\_Y\_Index","Orientation\_Index","Luminosity\_Index","SigmoidOfAreas","Pastry","Z\_Scratch","K\_Scatch","Stains","Dirtiness","Bumps","Other\_Faults"};  
// Object[] value = new Object[]{"42","50","270900","270944","267","17","44","24220","76","108","1687","1","0","80","0.0498","0.2415","0.1818","0.0047","0.4706","1","1","2.4265","0.9031","1.6435","0.8182","-0.2913","0.5822","1","0","0","0","0","0","0"};  
// System.out.println(getInsertQuery(0,name,value));* **while**((line = in.readLine()) != **null**) {  
 Object[] Data\_Array = line.split(**"\t\t"**);  
 String INSERT = getInsertQuery(id, Name, Data\_Array);  
*// System.out.println(INSERT);* **boolean** insert\_ok = statement.execute(INSERT);  
 **if** (insert\_ok) {  
 System.out.println(**"Insert Failed!"**);  
 }  
 id++;  
 }  
 statement.close();  
 mysql.Dis\_Connect();  
 } **catch** (SQLException e) {  
 e.printStackTrace();  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 }  
 in.close();  
 }  
}

## 【WriterGearData.java】

*/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 \* Author : HustWolf --- 张照博  
  
 \* Time : 2018.1-2018.5  
  
 \* Address : HUST  
  
 \* Version : 1.1  
  
 \* 将两百万条记录清洗后写入到数据库中  
  
 \* 1.1 版本更正了属性名字 HZ-->Load  
 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/***import** java.io.\*;  
**import** java.text.NumberFormat;  
**import** java.io.IOException;  
**import** java.sql.SQLException;  
**import** java.sql.Statement;  
**import** java.util.Scanner;  
  
  
**public class** WriteGearData {  
 **protected** Mysql\_Connect mysql=**new** Mysql\_Connect();  
 **private** Object[] Name = **new** String[]{**"Sensor1"**,**"Sensor2"**,**"Sensor3"**,**"Sensor4"**,**"Load"**,**"category"**};  
  
  
 **public void** Delete(**int** max){  
 **try** {  
 mysql.Connect();  
 Statement statement = mysql.getStatement();  
 String DELETE = **"delete from gear where id<"**+max;  
 **boolean** delete\_ok = statement.execute(DELETE);  
 **if** (delete\_ok) {  
 System.out.println(**"Have Fun! Boys!\n\n"**);  
 }  
 statement.close();  
 mysql.Dis\_Connect();  
 } **catch** (SQLException e) {  
 e.printStackTrace();  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 }  
 }  
 **public** String getInsertQuery(**int** id,Object[] name,Object[] value){  
 String insert=**"INSERT INTO gear(id"**;  
 **for** (**int** i=0;i<name.length;++i){  
 insert+= (**",`"**+(name[i]+**"`"**));  
 }  
 insert+=**") values("**+id;  
 **for** (**int** i=0;i<value.length;++i){  
 insert+=(**","**+value[i]+**""**);  
 }  
 insert+=**");"**;  
 **return** insert;  
 }  
  
 **public void** WriteData() **throws** IOException {  
 Delete(3000000);  
 **int** load[] = **new int** [] {0,10,20,30,40,50,60,70,80,90};  
*// int id = 1005311;* **int** id = 0;  
 String [] FileNames = **new** String[]{**"/Users/zhangzhaobo/Documents/Graduation-Design/Data/BrokenTooth Data/b30hz"**,**"/Users/zhangzhaobo/Documents/Graduation-Design/Data/Healthy Data/h30hz"**};  
 **for**(**int** fileindex = 0;fileindex <FileNames.length;++fileindex ) {  
 **for** (**int** x : load) {  
 String filename = FileNames[fileindex] + x + **".txt"**;  
 Scanner scanner = **new** Scanner(**new** FileInputStream(filename));  
 **int** count = 0;  
 **try** {  
 mysql.Connect();  
 Statement statement = mysql.getStatement();  
 **do** {  
 *//连续添加多条静态SQL* Object[] line = **new** Object[6];  
 NumberFormat nf = NumberFormat.getNumberInstance();  
 nf.setMaximumFractionDigits(4);  
 **for** (**int** i = 0; i < 4; ++i) {  
 line[i] = Float.parseFloat(nf.format(scanner.nextFloat()));  
 }  
 line[4] = x;  
 line[5] = fileindex;  
 String INSERT = getInsertQuery(id, Name, line);  
 statement.addBatch(INSERT);  
 id++;  
 count++;  
 *//执行批量执行* **if** (count > 40000) {  
 statement.executeBatch();  
 count = 0;  
 System.out.println(**"写入了一次40000条"**);  
 }  
 } **while** (scanner.hasNext());  
 statement.executeBatch();  
 statement.close();  
 mysql.Dis\_Connect();  
 } **catch** (SQLException e) {  
 e.printStackTrace();  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 }  
 }  
 }  
 }  
}

## 【ZZB\_SVM.java】

**import** java.io.IOException;  
**import** java.text.NumberFormat;  
  
**public class** ZZB\_SVM {  
 **public static** Float main() **throws** IOException {  
 SVMReadData sr = **new** SVMReadData();  
 Parameter par = **new** Parameter();  
 String trainFileName = sr.readTrainData(par);  
 String testFileName = sr.readTestData(par);  
 *//训练使用的数据以及训练得出生成的模型文件名。* String[] trainFile = { trainFileName, **"model.txt"** };  
 *//测试数据文件，模型文件，结果存放文件* String[] predictFile = { testFileName, **"model.txt"**,**"predict.txt"** };  
 System.out.println(**"........SVM Start.........."**);  
 **long** start=System.currentTimeMillis();  
 svm\_train.main(trainFile); *//训练* System.out.println(**"Usage of Time : "**+(System.currentTimeMillis()-start));  
 *//预测* **float** x = svm\_predict.main(predictFile);  
 **return** x;  
 }  
 **public static void** DataToPlot() **throws** IOException {  
 NumberFormat nf = NumberFormat.getNumberInstance();  
 nf.setMaximumFractionDigits(1);  
 **int**[] numOfTrain = **new int**[]{  
 200, 400, 800, 1000, 2000,  
 3000, 4000, 5000, 6000, 8000,  
 10000, 12000, 15000, 17500, 20000,  
 22500, 25000, 27500, 30000, 32500,  
 35000, 37500, 40000, 42500, 45000,  
 };  
 **float**[] ACC = **new float**[numOfTrain.length];  
 SVMReadData sr = **new** SVMReadData();  
 Parameter par = **new** Parameter();  
 **for** (**int** number = 0; number < numOfTrain.length; ++number) {  
 par.setTrainNum(numOfTrain[number]);  
 String trainFileName = sr.readTrainData(par);  
 String testFileName = sr.readTestData(par);  
 String[] trainFile = {trainFileName, **"model.txt"**};  
 *//测试数据文件，模型文件，结果存放文件* String[] predictFile = {testFileName, **"model.txt"**, **"predict.txt"**};  
 **long** start = System.currentTimeMillis();  
 svm\_train.main(trainFile); *//训练* System.out.println(**"Usage of Time : "** + (System.currentTimeMillis() - start));  
 *//预测* ACC[number] = Float.parseFloat(nf.format(svm\_predict.main(predictFile)));  
 }  
 System.out.print(**"x = ["**);  
 **for** (**int** num:numOfTrain) System.out.print(num + **","**);  
 System.out.println(**"]"**);  
  
 System.out.print(**"y1 = ["**);  
 **for** (**float** num:ACC) System.out.print(num + **","**);  
 System.out.println(**"]"**);  
 }  
}

## 【ZZB\_JCS.java】

*/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 \* Author : HustWolf --- 张照博  
  
 \* Time : 2018.1-2018.5  
  
 \* Address : HUST  
  
 \* Version : 1.5  
  
 \* 决策树主体，从读取到生成  
 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/***import** java.io.\*;  
**import** java.text.NumberFormat;  
**import** java.util.HashMap;  
**import** java.util.LinkedList;  
**import** java.util.List;  
**import** java.util.Map;  
**import** java.util.Map.Entry;  
**import** java.util.Set;  
  
*//最外层类名***public class** ZZB\_JCS{  
 **private** NumberFormat nf = NumberFormat.getNumberInstance();  
  
 */\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 \* Define the Class of Sample  
  
 \* it is about its nature and function  
 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/* **static class** Sample{  
 *//attributes means 属性* **private** Map<String,Object> attributes = **new** HashMap<String,Object>();  
 *//category means 类别* **private** Object category;  
  
 **public** Object getAttribute(String name){  
 **return** attributes.get(name);  
 }  
  
 **public void** setAttribute(String name,Object value){  
 attributes.put(name,value);  
 }  
  
 **public void** setCategory(Object category){  
 **this**.category=category;  
 }  
  
 **public** String toString(){  
 **return** attributes.toString();  
 }  
 }  
 */\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 \* this is the function to read the sample  
  
 \* just like decoding the data  
 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/* **static void** DataToTest(Parameter par, ReadData data) **throws** IOException{  
 FileWriter out1 = **new** FileWriter(**"DataToTest.txt"**);  
 Object[][] DataToTest = data.readTestData(par);  
 **for** (**int** i = 0; i < DataToTest.length; ++i) {  
 **for** (**int** j = 0; j < DataToTest[i].length; ++j) {  
 out1.write(DataToTest[i][j] + **" "**);  
 }  
 out1.write(**"\n"**);  
 }  
 out1.close();  
 }  
*// 此处需要改造为读取外部数据！并且能够进行分解，改造为可读取的形式* **static** Map<Object,List<Sample>> getSample(ReadData data, String[] attribute\_Names, Parameter par, **boolean** WriteToTXT){  
 *//样本属性及其分类，暂时先在代码里面写了。后面需要数据库或者是文件读取* Object[][] rawData = data.readTrainData(par);  
 *//最终组合出一个包含所有的样本的Map* Map<Object,List<Sample>> sample\_set = **new** HashMap<Object,List<Sample>>();  
  
 **if** (WriteToTXT) {  
 **try**{  
 DataToTest(par,data);  
 }**catch** (IOException e){  
 System.out.println(e);  
 }  
 }  
 *//读取每一排的数据  
 //分解后读取样本属性及其分类，然后利用这些数据构造一个Sample对象  
 //然后按照样本最后的分类划分样本集，* **for** (Object[] row:rawData) {  
 *//新建一个Sample对象，没处理一次加入Map中，最后一起返回* Sample sample = **new** Sample();  
 **int** i=0;  
 *//每次处理一排数据，构成一个样本中各项属性的值* **for** (**int** n=row.length-1; i<n; ++i) {  
 sample.setAttribute(attribute\_Names[i],row[i]);  
 }  
 *//为处理完的一个样本进行分类，根据0，1来，此时i已经在最后一位* sample.setCategory(row[i]);  
 *//将解析出来的一排加入整体分类后的样本中，row[i]此刻是指分类后的集合* List<Sample> samples = sample\_set.get(row[i]);  
 *//现在整体样本集中查询，有的话就返回value，而如果这个类别还没有样本，那么就添加一下* **if**(samples == **null**){  
 samples = **new** LinkedList<Sample>();  
 sample\_set.put(row[i],samples);  
 }  
 *//不管是当前分类的样本集中是否为空，都要加上把现在分离出来的样本丢进去。  
 //此处基本只有前几次分类没有完毕的时候才会进入if，后面各个分类都有了样本就不会为空了。* samples.add(sample);  
 }  
 *//最后返回的是一个每一个类别一个链表的Map，串着该类别的所有样本 (类别 --> 此类样本)* **return** sample\_set;  
 }  
  
 */\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 \* this is the class of the decision-tree  
  
 \* 决策树（非叶结点），决策树中的每个非叶结点都引导了一棵决策树  
  
 \* 每个非叶结点包含一个分支属性和多个分支，分支属性的每个值对应一个分支，该分支引导了一棵子决策树  
 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/* **static class** Tree{  
  
 **private** String attribute;  
 **private** Map<Object,Object> children = **new** HashMap<Object,Object>();  
 **public** Tree(String attribute){  
 **this**.attribute=attribute;  
 }  
  
 **public** String getAttribute(){  
 **return** attribute;  
 }  
  
 **public** Object getChild(Object attrValue){  
 **return** children.get(attrValue);  
 }  
  
 **public void** setChild(Object attrValue,Object child){  
 children.put(attrValue,child);  
 }  
  
 **public** Set<Object> getAttributeValues(){  
 **return** children.keySet();  
 }  
 }  
  
 */\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 \* this is the function to choose the Best Test Attribute ID3 Algorithm  
  
 \* it will be used in the generateDecisionTree()  
  
 \* 选取最优测试属性。最优是指如果根据选取的测试属性分支，则从各分支确定新样本  
  
 \* 的分类需要的信息熵之和最小，这等价于确定新样本的测试属性获得的信息增益最大  
  
 \* 返回数组：选取的属性下标、信息熵之和、Map(属性值->(分类->样本列表))  
 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/* **static** Object[] ID3(Map<Object,List<Sample>> categoryToSamples,String[] attribute\_Names){  
 *//最优的属性的下标！* **int** minIndex = -1;  
 *//最小的信息熵* **double** minValue = Double.MAX\_VALUE;  
 *//最优的分支方案！* Map<Object,Map<Object,List<Sample>>> minSplit = **null**;  
  
 *//对每一个属性，都要计算信息熵，选区最小的为最优，Ent(D)* **for** (**int** attrIndex = 0;attrIndex<attribute\_Names.length;++attrIndex) {  
 *//统计样本总数的计数器* **int** allCount = 0;  
 *//按照当前属性构建Map，这个Map的层级关系根据下面的层次划分：属性值[Key]->(分类[Key]->样本列表[Value]) [Value]  
 // curSplits就是一个某一个在当前属性下某一种选择值 所对应的所有样本集！ 所有的Dv的集合是也？？待定！* Map<Object,Map<Object,List<Sample>>> curSplits = **new** HashMap<Object,Map<Object,List<Sample>>>();  
  
 */\* 这儿的整个流程画个图哈~下面是对某一个属性进行信息增益的计算了！  
  
 拿到一个数据对，【所属类别-->样本集】  
 |  
 V  
 解析数据对，分解出key和value  
 其中key为类别，value为此类别所有的样本  
 |  
 V  
 对于Value里边读出来的每个样本，分别：  
 读取当前属性下的值，然后建立起来当前属性值相同的所有样本的样本集；  
 |  
 V  
 此处还要将每个样本集拆分为分类样本集！  
 |  
 V  
 这一轮下来，就得到关于这个属性的不同属性值对应的样本集合  
 而在这些集合集合中又有分类样本集！  
 就好比，这一轮对年龄下手，最终得到了40岁以上的好人、坏人  
 30-40岁之间的好人、坏人集合  
 30岁以下的好人、坏人的集合  
 最后一共得到了6个样本集？  
 只不过是已Map中键值对的形式存在，二层包装而已！  
 --正分类--> 一个Map  
 ---属性值1，比如是学生 ->分类两类 |  
 某一属性（这个就是curSplits这个Map的本体） --负分类--> 一个Map（此处画图方便重用了！）  
 ---属性值2，比如不是学生->分类两类 |  
 --正分类--> 一个Map  
 \*/  
 /\*  
 \* Set<Map.Entry<K,V>> entrySet​()  
 \* Returns: A set view of the mappings contained in this map  
 \* Entry 这个数据类型大致等于C++中的pair，也就是数据打包的意思  
 \*/* **for** (Entry<Object,List<Sample>> entry : categoryToSamples.entrySet()) {  
 *//先拿到数据的分类的名称，我们这儿就0，1* Object category = entry.getKey();  
 *//再拿到这个类别！注意是类别，不是属性值！类别所对应的所有样本！* List<Sample> samples = entry.getValue();  
 *//然后再慢慢的对每个样本进行操作，将其分为按照属性值划分的各种Dv，然后返回到curSplits* **for** (Sample sample : samples ) {  
 *// 根据当前要计算的属性，得到当前样本的关于这个属性的值* Object attrValue = sample.getAttribute(attribute\_Names[attrIndex]);  
 *// 根据前面当前样本getAttribute()所获得的属性值，来获取这个属性的值相同的所有的样本的样本集* Map<Object,List<Sample>> split = curSplits.get(attrValue);  
 *// 考虑到一开始肯定没法得到一个完整的Map,所以需要从无到有建立起来！* **if** (split == **null**) {  
 *//建立一个关于这个属性值的Map，层次关系为：属性值->(All Sample) 见Line156* split = **new** HashMap<Object,List<Sample>>();  
 curSplits.put(attrValue,split);  
 }  
 *//建立起来之后，就可以读取这个属性值等于某个值时对应的分类样本集合了。* List<Sample> splitSamples = split.get(category);  
 *// 如果读不到当前属性对应这个值的分类的话，那就要建立一个属性值等于当前样本的属性值，且分类相同的样本集。* **if** (splitSamples == **null**) {  
 splitSamples = **new** LinkedList<Sample>();  
 *// 结合当前这个属性值，组成一个集合，放到Map--split里面去。* split.put(category,splitSamples);  
 }  
 *// 最后再把当前的这个样本放到这个样本集中？？？！！可以直接这么搞的？%%%%%% 难道是引用传递？  
 // 是的！没有用new自然就是一个引用传递！卧槽！这都给忘了！？* splitSamples.add(sample);  
 }  
 *//统计样本总数的计数器需要对当前属性下的样本的数量进行统计。* allCount += samples.size();  
 }  
  
 *// 当前属性值的信息增益寄存器* **double** curValue = 0.0;  
 *//读取当前属性下的每一种属性值对应的样本集* **for** (Map<Object,List<Sample>> splits : curSplits.values()) {  
 **double** perSplitCount = 0;  
 *//读取每个属性值的样本集Dv的size，得到所有该属性为此值的样本总数，不论类别如何* **for** (List<Sample> list : splits.values()) {  
 *//累计当前样本的分支总数* perSplitCount += list.size();  
 }  
 *//计数器，当前分支的信息熵和信息增益，这儿是按出现频率在算呢！* **double** perSplitValue = 0.0;  
 *//计算每个属性值对应的信息熵* **for** (List<Sample> list : splits.values() ) {  
 *//此处完全就是ID3算法的信息熵的计算公式！也就是ENT(D) = -Sum(Pk\*log2(Pk))见《机器学习》 P75* **double** p = list.size() / perSplitCount;  
 *//貌似是因为p无论如何都是小于1的，所以采用p -= 实际上是加了？* perSplitValue -= p\*(Math.log(p)/Math.log(2));  
 }  
 *//这应该还算不上不是信息增益吧！只能算是信息熵之和了。* curValue += (perSplitCount / allCount) \* perSplitValue;  
 }  
 *//选择最小的信息熵为最优！？* **if** (minValue > curValue){  
 minIndex = attrIndex;  
 minValue = curValue;  
 minSplit = curSplits;  
 }  
 }  
 *//所以最终返回的就是一个信息熵之和 最小的属性的列表索引 + 最小的信息熵之和 + 最小的信息熵之和所对应的子树！  
// System.out.println(attribute\_Names[minIndex]);* **return new** Object[] {minIndex,minValue,minSplit};  
 }  
  
  
 */\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 \* this is the function to output the Decision Tree to the Dashboard  
 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/* **static void** outputDecisionTree(FileWriter out,Object obj, **int** level, Object from) **throws** IOException {  
 *//这个到后面决定输出多少个|----- 也就是说是决定层级的* **for** (**int** i=0; i < level ;++i){  
*// System.out.print("|---->");* out.write(**"|---->"**);  
 }  
 *// 所有子节点专用？除了根节点都要吧！* **if** (from != **null**){  
*// System.out.printf("(%s):",from);* out.write(**"("**+from+**"):"**);  
 }  
 *//大概是说，如果这个东西还有子节点，那就继续递归* **if** (obj **instanceof** Tree){  
 Tree tree = (Tree) obj;  
 String attribute\_Name = tree.getAttribute();  
*// System.out.printf("[%s = ?]\n",attribute\_Name);* out.write(**"["**+attribute\_Name+**" = ?]\n"**);  
 **for** (Object attrValue : tree.getAttributeValues()){  
 Object child =tree.getChild(attrValue);  
 outputDecisionTree(out,child,level+1,attribute\_Name + **" = "** + attrValue);  
 }  
 }**else** {  
*// System.out.printf("【\* CATEGORY = %s \*】\n", TestData.getCategory(obj));* out.write(**"【\* CATEGORY = "**+TestData.getCategory(obj)+**" \*】\n"**);  
 }  
 }  
  
  
 */\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 \* this is the function to generate the DecisionTree  
  
 \* use the data which read from the files to get the Decisiontree  
  
 \* the most important part I think!  
 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/* **static** Object generateDecisionTree(Map<Object,List<Sample>> categoryToSamples,String[] attribute\_Names){  
 *//如果只有一个分类，那么该样本所属分类作为新样本的分类* **if**(categoryToSamples.size() == 1) {  
 **return** categoryToSamples.keySet().iterator().next();  
 }  
 *//如果没有提供决策的属性（也就是没有给你属性名字清单），那么样本集中具有最多样本的分类作为新样本的分类，也就是投票选举出新的分类* **if** (attribute\_Names.length == 0) {  
 **int** max = 0;  
 Object maxCategory = **null**;  
 *// 如果没有属性列表的话，那就直接按照分类作为K个样本集，取数量较大的那个样本集的类别作为本分类。* **for** (Entry<Object,List<Sample>> entry : categoryToSamples.entrySet() ) {  
 **int** cur = entry.getValue().size();  
 **if** (cur > max) {  
 max = cur;  
 maxCategory = entry.getKey();  
 }  
 }  
 **return** maxCategory;  
 }  
 *//如果有属性清单的话，那么就选择测试所用的属性了。* Object[] rst = ID3(categoryToSamples,attribute\_Names);  
*// System.out.println(attribute\_Names[(Integer)rst[0]]);  
 //决策树的根节点选取，分支的属性为选取的测试属性* Tree tree = **new** Tree(attribute\_Names[(Integer)rst[0]]);  
 *//已用过的测试属性不能再次被选择为测试属性* String[] Attr\_Not\_Used = **new** String[attribute\_Names.length-1];  
 **for** (**int** i=0,j=0;i<attribute\_Names.length ;++i ) {  
 **if** (i != (Integer)rst[0]) {  
 Attr\_Not\_Used[j++] = attribute\_Names[i];  
 }  
 }  
 *//根据分支的属性生成新的分支* @SuppressWarnings(**"unchecked"**)  
 Map<Object,Map<Object,List<Sample>>> splits = (Map<Object,Map<Object,List<Sample>>>) rst[2];  
 **for** (Entry<Object,Map<Object,List<Sample>>> entry : splits.entrySet()) {  
 Object attrValue = entry.getKey();  
 Map<Object,List<Sample>> split = entry.getValue();  
 *//又是递归调用？那我岂不是玩完？层数不能超过二十层！这是底线！* Object child = generateDecisionTree(split,Attr\_Not\_Used);  
 tree.setChild(attrValue,child);  
 }  
 **return** tree;  
 }  
  
 **public static void** readTXT(GUI gui,Object decisionTree){  
 **try** {  
 File file = **new** File(**"GUIDATA.txt"**);  
 BufferedReader in = **new** BufferedReader(**new** FileReader(file));  
 **int** linecount = 0;  
 **while** (in.readLine() != **null**) {  
 ++linecount;  
 }  
 in.close();  
*// System.out.println(linecount);* in = **new** BufferedReader(**new** FileReader(file));  
 String[] LINES = **new** String[linecount];  
 **for** (**int** i = 0; i < linecount; ++i) {  
 LINES[i] = in.readLine();  
 }  
 in.close();  
 GUI.updateTEXT(gui, LINES, decisionTree);  
 }**catch** (IOException e){  
 System.out.println(**"Nothing! Continue!"**);  
 }  
 }  
  
 **public void** DataToPlot() **throws** Exception {  
 ReadData data = **new** ReadData();  
 Parameter par = **new** Parameter();  
 nf.setMaximumFractionDigits(1);  
 String[] attribute = **new** String[]{**"Sensor1"**, **"Sensor2"**, **"Sensor3"**, **"Sensor4"**, **"Load"**};  
 String[] attribute\_Names = **new** String[]{**"Sensor1"**, **"Sensor2"**, **"Sensor3"**, **"Sensor4"**, **"Load"**, **"category"**};  
 **int**[] numOfTrain = **new int**[]{  
 200, 400, 800, 1000, 2000,  
 3000, 4000, 5000, 6000, 8000,  
 10000, 12000, 15000, 17500, 20000,  
 22500, 25000, 27500, 30000, 32500,  
 35000, 37500, 40000, 42500, 45000,  
 47500, 50000, 52500, 55000, 57500,  
 60000, 62500, 65000, 67500, 70000,  
 72500, 75000, 77500, 80000, 82500,  
 85000, 87500, 90000, 92500, 95000,  
 97500, 100000, 110000, 120000, 125000,  
 130000, 140000, 150000, 160000, 175000,  
 };  
 **float**[] ACC = **new float**[numOfTrain.length];  
 **float**[] Precision = **new float**[numOfTrain.length];  
 **float**[] Recall = **new float**[numOfTrain.length];  
 *//读取样本集* **for** (**int** number = 0; number < numOfTrain.length; ++number) {  
 **long** startTime=System.currentTimeMillis(); *//获取开始时间* par.setTrainNum(numOfTrain[number]);  
 Map<Object, List<Sample>> samples = getSample(data, attribute\_Names, par, **false**);  
 *//生成决策树* Object decisionTree = generateDecisionTree(samples, attribute);  
 *//输出决策树  
// File file = new File("GUIDATA.txt");  
// FileWriter out = new FileWriter(file);  
// outputDecisionTree(out,decisionTree,0,null);  
// out.close();* Object[][] testD = data.readTestData(par);  
 **float** RightCount = 0;  
 **float** FaultCount = 0;  
 **float** TP = 0;  
 **float** FN = 0;  
 **float** FP = 0;  
 **float** TN = 0;  
 **for** (Object[] test : testD) {  
 String res = **""**;  
 res = TestData.TestData(decisionTree, attribute, test, res);  
 **if** (res.contains(**":"**)) {  
 **if** (res.contains(**"1"**)) {  
 **if** ((**float**) test[test.length - 1] == 1) {  
 RightCount += 1;  
 TP += 1;  
 } **else** {  
 FaultCount += 1;  
 FP += 1;  
 }  
 } **else if** (res.contains(**"0"**)) {  
 **if** ((**float**) test[test.length - 1] == 0) {  
 RightCount += 1;  
 TN += 1;  
 } **else** {  
 FaultCount += 1;  
 FN += 1;  
 }  
 }  
 } **else** {  
 FaultCount += 1;  
 }  
 }  
 **float** acc = Float.parseFloat(nf.format(RightCount / (RightCount + FaultCount) \* 100));  
 **float** precision = Float.parseFloat(nf.format(TP / (TP + FP)\*100));  
 **float** recall = Float.parseFloat(nf.format(TP / (TP + FN)\*100));  
 System.out.println(**"数量级为："**+par.getTrainNum()+**",精度为："** + acc + **"%，查准率为："** + precision + **"，%查全率为："** + recall+**"%"**);  
 **long** endTime=System.currentTimeMillis(); *//获取结束时间* System.out.println(**"程序运行时间： "**+(endTime-startTime)+**"ms"**);  
 ACC[number] = acc;  
 Precision[number] = precision;  
 Recall[number] = recall;  
 }  
 System.out.print(**"x = ["**);  
 **for** (**int** num:numOfTrain) System.out.print(num + **","**);  
 System.out.println(**"]"**);  
  
 System.out.print(**"y1 = ["**);  
 **for** (**float** num:ACC) System.out.print(num + **","**);  
 System.out.println(**"]"**);  
  
 System.out.print(**"y2 = ["**);  
 **for** (**float** num:Precision) System.out.print(num + **","**);  
 System.out.println(**"]"**);  
  
 System.out.print(**"y3 = ["**);  
 **for** (**float** num:Recall) System.out.print(num + **","**);  
 System.out.println(**"]"**);  
  
 System.out.println(**"plt.plot(x, y1, 'r--', x, y2, 'bs', x, y3, 'g^')"**);  
 System.out.println(**"plt.axis([0, 180000, 20, 70])"**);  
  
 }  
 **public static** Object JCS(Parameter par) **throws** Exception{  
 String[] attribute = **new** String[] {**"Sensor1"**,**"Sensor2"**,**"Sensor3"**, **"Sensor4"**, **"Load"**};  
 String[] attribute\_Names = **new** String[] {**"Sensor1"**,**"Sensor2"**,**"Sensor3"**,**"Sensor4"**,**"Load"**, **"category"**};  
 *//读取样本集* ReadData data = **new** ReadData();  
 Map<Object,List<Sample>> samples = getSample(data,attribute\_Names,par,**true**);  
 *//生成决策树* Object decisionTree = generateDecisionTree(samples,attribute);  
 *//输出决策树* File file = **new** File(**"GUIDATA.txt"**);  
 FileWriter out = **new** FileWriter(file);  
 outputDecisionTree(out,decisionTree,0,**null**);  
 out.close();  
 **return** decisionTree;  
 }  
  
 **public static void** main(String[] args) **throws** Exception{  
 **long** startTime=System.currentTimeMillis(); *//获取开始时间  
// ####################################################  
// 下面是主体的，生成决策树，GUI部分的内容  
//####################################################* GUI gui = **new** GUI();  
*//####################################################  
// 下面是输出决策树画图用的数据的内容  
//####################################################  
// ZZB\_JCS zzb = new ZZB\_JCS();  
// zzb.DataToPlot();  
  
// ####################################################  
// 下面是输出SVM的画图数据的内容  
//####################################################  
// ZZB\_SVM.DataToPlot();* **long** endTime=System.currentTimeMillis(); *//获取结束时间* System.out.println(**"本次运行时间："**+(endTime-startTime)+**"ms"**);  
 }  
}

## 【GUI.java】

*/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 \* Author : HustWolf --- 张照博  
  
 \* Time : 2018.1-2018.5  
  
 \* Address : HUST  
  
 \* Version : 2.0  
  
 \* 人机交互界面  
 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/***import** javax.swing.\*;  
**import** java.awt.\*;  
**import** java.awt.event.\*;  
**import** java.io.\*;  
**import** java.text.NumberFormat;  
**import** java.util.Vector;  
**import** java.util.HashMap;  
**import** java.util.Map;  
  
*/\*  
 8 \* GUI(图形用户界面)  
 9 \* Graphical User Interface(图形用户接口)  
 10 \* 用图形的方式,来显示计算机操作的界面,这样更方便更直观.  
 11 \*  
 12 \* CLI  
 13 \* Command Line User Interface(命令行用户接口)  
 14 \* 就是常用的Dos命令行操作.  
 15 \* 需要记忆一些常用的命令.操作更直观.  
 16 \*  
 17 \* 举例:  
 18 \* 比如:创建文件夹,或者删除文件夹等  
 19 \* md haha del haha  
 20 \*  
 21 \*  
 22 \* Java的GUI提供的对象都存在 java.Awt 和 javax.Swing 两个包中.  
 23 \*  
 24 \* java.Awt:Abstract Window ToolKit(抽象 窗口工具包)  
 25 \* 需要调用本地系统方法实现功能.属重量级控件 (跨平台不够强)  
 26 \*  
 27 \* java.Swing:在AWT的基础上,建立的一套图形界面系统,其中提供了更多的组件,  
 28 \* 而且完全由java实现,增强了移植性,属于轻量级控件.(跨平台很好)  
 29 \*  
 30 \* java.swt: IBM 公司开发 Eclipse 用的组件工具 可以Eclipse网站下载后就可以使用了.  
 31 \*  
 32 \*  
 33 \* 布局管理器  
 34 \* 1)容器中的组件的排放方式,就是布局.  
 35 \* 2)常见的布局管理器  
 36 \* FlowLayout(流式布局管理器)  
 37 \* 从左到右的顺序排列  
 38 \* Panel默认的布局管理器  
 39 \* BorderLayout(便捷布局管理器)  
 40 \* 东 南 西 北 中  
 41 \* Frame 默认的布局管理器  
 42 \* 不指定布局方式,默认 满屏覆盖,在添加一个 也是 满屏覆盖  
 43 \* GridLayout (网格布局管理器)  
 44 \* 规则的矩阵  
 45 \* CardLayout (卡片布局管理器)  
 46 \* 选项卡  
 47 \* GridBagLayout(网格包布局管理器)  
 48 \* 非规则的矩阵  
 49 \*  
 50 \* 事件监听机制组成  
 51 \* 事件源:  
 52 \* 事件:Event  
 53 \* 监听器:Listener  
 54 \* 时间处理:(引发事件后处理方式)  
 55 \*  
 56 \* 事件源:就是awt包或者swing包中的那些图像界面组件.  
 57 \* 事件:每个事件源都有自己特定的对应时间和共性时间.  
 58 \* 监听器:可以出发某一个事件的动作都已经封装到监听器中.  
 59   
\*/***class** MyWin **extends** WindowAdapter{  
 @Override  
 **public void** windowClosing(WindowEvent e) {  
 System.out.println(**"Bye Bye!"**);  
 JOptionPane.showMessageDialog(**null**,**" Welcome for Your Next Time!"**,**"MESSAGE FROM ZZB"**,JOptionPane.WARNING\_MESSAGE);  
 System.exit(0);  
 }  
 @Override  
 **public void** windowActivated(WindowEvent e) {  
 *//每次获得焦点 就会触发* System.out.println(**"Welcome Back!"**);  
 }  
 @Override  
 **public void** windowOpened(WindowEvent e) {  
 *// TODO Auto-generated method stub* System.out.println(**"Now It is Working!"**);  
 JOptionPane.showMessageDialog(**null**,**"Welcome To Here!\n【init】：Load Model！\n【clear】: Clear the Screen!\n【next 】：The Next Line！\n【exit 】： Exit the System!\nFor More Option, input \"HELP\""**,**"MESSAGE FROM ZZB"**,JOptionPane.WARNING\_MESSAGE);  
 }  
}  
  
**public class** GUI{  
 **private boolean** isInit = **false**;  
 **public** Parameter par = **new** Parameter();  
 **private float** RightCount = 0;  
 **private float** FaultCount = 0;  
 **private** Object tree;  
 **private** Frame f;  
 **private int** nextTimes=0;  
 **private** String[] Test\_Names = **new** String[] {**"Sensor1"**,**"Sensor2"**,**"Sensor3"**,**"Sensor4"**, **"Load"**};  
  
 **private** Button but,but1,but2;  
 **private** TextField ta;  
 **private** MenuBar mb;  
 **private** Menu m,subm,Run;  
 **private** MenuItem closeItem,openItem,saveItem,subItem1,subItem,subItem2,subItem3,subItem4,subItem5;  
 **private** FileDialog openDialog,saveDialog;  
 **private** File file;  
 **private** JPanel jp1,jp2,jp3,jp4,jp5,jp6,jp7,jp8,jp9,jp10,jp11,jp12,jp13,jp14;  
 **private** JLabel jl1,jl2,jl3,jl4,jl5,jl6,jl7,jl8,jl9,jl10,jl11,jl12;  
 **private** String[] TEXT = **new** String[11];  
 **private static** String[] LINES;  
 **public static int** line=0;  
 **private** Vector<String > TData = **new** Vector<String>();  
 **private static** String Space = **"\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t"**;  
  
 GUI() {  
 init();  
 par.setTrainNum(400);  
 }  
  
 **private void** saveCommand(){  
 **if**(file==**null**){  
 saveDialog.setVisible(**true**);  
 String dirPath=saveDialog.getDirectory();  
 String fileName=saveDialog.getFile();  
 **if**(dirPath==**null** || fileName==**null**)  
 **return**;  
 file=**new** File(dirPath,fileName);  
 }  
 **try** {  
 BufferedWriter bufw = **new** BufferedWriter(**new** FileWriter(file));  
 **int** i = 0;  
 **while** (i < LINES.length){  
 **if**(LINES[i]!=**null**)  
 bufw.write(LINES[i]+**"\n"**);  
 i++;  
 }  
 bufw.close();  
 } **catch** (IOException e2) {  
 **throw new** RuntimeException(**"Failed to Save !"**);  
 }  
 }  
 **private void** addLine(){  
 **if** (line<=LINES.length/10)  
 line++;  
 **else**{  
 line = 0;  
 }  
 }  
 **private void** subLine(){  
 **if** (line>0)  
 line--;  
 **else**{  
 line = 0;  
 }  
 }  
 **private void** autoTest(){  
 **for** (**int** i=0;i<TData.size();++i) {  
 Object[] test = TData.get(i).split(**" "**);  
 String res=**""**;  
 res=TestData.TestData(tree, Test\_Names,test,res);  
 **if** (res.contains(**":"**)){  
 **if**((res.contains(**"1"**) && (((String)test[test.length-1]).contains(**"1"**))) || ((res.contains(**"0"**) && (((String)test[test.length-1]).equals(**"0.0"**))))){  
 RightCount++;  
 }  
 **else** {  
 FaultCount++;  
 }  
 *//一套方案！\*\*\*\*\*\*\*\*\*\*\*\** }  
 **else** {  
 FaultCount++;  
 }  
 }  
 }  
 **public static void** updateTEXT(GUI obj,String[] txt,Object tree1){  
 LINES = **new** String[txt.length];  
 **for** (**int** i=0;i<txt.length;++i){  
 obj.LINES[i]=txt[i];  
 }  
 obj.tree=tree1;  
 }  
  
 **public void** updateDisplay(){  
 **if**(line<LINES.length/10) {  
 **for** (**int** i = line \* 10, j = 1; i < line \* 10 + 10; i++) {  
 TEXT[j++] =Space+**""**+ LINES[i];  
 }  
 }**else**{  
 **for**(**int** i=1;i<LINES.length-(LINES.length/10)\*10;++i){  
 TEXT[i] = Space+**""**+LINES[(LINES.length/10)\*10+i];  
 }  
 **for** (**int** i=LINES.length-(LINES.length/10)\*10;i<11;++i){  
 TEXT[i]=Space+**"|||||||||||||||||===========》》》》》》DONE!"**;  
 }  
 }  
 System.out.println(line);  
 jl2.setText(TEXT[1]);  
 jl3.setText(TEXT[2]);  
 jl4.setText(TEXT[3]);  
 jl5.setText(TEXT[4]);  
 jl6.setText(TEXT[5]);  
 jl7.setText(TEXT[6]);  
 jl8.setText(TEXT[7]);  
 jl9.setText(TEXT[8]);  
 jl10.setText(TEXT[9]);  
 jl11.setText(TEXT[10]);  
 }  
 **private void** openFile(){  
 openDialog.setVisible(**true**);  
 String dirPath=openDialog.getDirectory();  
 String fileName=openDialog.getFile();  
 System.out.println(dirPath+**"...."**+fileName);  
 **if**(dirPath==**null** || fileName==**null**)  
 **return**;  
 file=**new** File(dirPath,fileName);  
 **try** {  
 BufferedReader bufr=**new** BufferedReader(**new** FileReader(file));  
 String line=**null**;  
 **while**((line=bufr.readLine())!=**null**){  
 TData.add(line);  
 }  
 bufr.close();  
 } **catch** (IOException e2) {  
 **throw new** RuntimeException(**"open the Exception"**);  
 }  
 }  
 **private void** dealCommand(String command){  
 **if** (command.isEmpty()) {  
 JOptionPane.showMessageDialog(**null**, **"未检测到命令"**, **"输入错误"**, JOptionPane.WARNING\_MESSAGE);  
 }  
 **if** (command.toLowerCase().equals(**"exit"**)){  
 System.exit(0);  
 }  
 **else if** (command.toLowerCase().equals(**"help"**)){  
 JOptionPane.showMessageDialog(**null**,**"【init】：Load Model！\n【clear】: Clear the Screen!\n【test】: Test your DATA!\n【load】:Load the File Choosed\n【autoload】：Load the default testdata\n【autotest】：test all Test\_Data\n【last】：The Last Line\n【next】：The Next Line\n【help】：Show All Options\n【save】：Save the DecisionTree as '.txt' File\n【showinfo】：Show Info of the User\n【exit】： Exit the System\n【setTrainNum】：Reset the TrainData Num\n【setTestNum】：Reset the TestData Num"**,**"\*\*\*\* Command Option \*\*\*\*"**,JOptionPane.WARNING\_MESSAGE);  
 **return**;  
 }  
  
 String [] comm = command.split(**" "**);  
 **if**(comm[0].toLowerCase().equals(**"settrainnum"**)){  
 **try**{  
 par.setTrainNum(Integer.valueOf(comm[1]));  
 jl12.setText(Space+**"设置完毕，当前训练数据为："**+par.getTrainNum()+**"，请初始化！"**);  
 isInit = **false**;  
 }**catch** (Exception e){  
 System.out.println(e);  
 JOptionPane.showMessageDialog(**null**,**"你输入的数据有误，标准为：\n\t SetTrainNum 10000"**,**"Type Error"**,JOptionPane.WARNING\_MESSAGE);  
 }  
 **return**;  
 }  
 **else if**(comm[0].toLowerCase().equals(**"settestnum"**)){  
 **try**{  
 par.setTestNum(Integer.valueOf(comm[1]));  
 jl12.setText(Space+**"设置完毕，当前测试数据为："**+par.getTestNum()+**"，请初始化！"**);  
 isInit = **false**;  
 }**catch** (Exception e){  
 JOptionPane.showMessageDialog(**null**,**"你输入的数据有误，标准为：\n\t SetTestNum 10000"**,**"Type Error"**,JOptionPane.WARNING\_MESSAGE);  
 }  
 **return**;  
 }  
 **if** (command.toLowerCase().equals(**"init"**)){  
 **try** {  
 ZZB\_JCS.readTXT(**this**, ZZB\_JCS.JCS(par));  
 }**catch** (Exception e){  
 System.out.println(**"初始化失败！"**);  
 }  
 isInit = **true**;  
 line=0;  
 jl12.setText(Space+**"初始化完成，请继续操作（可键入\"help\"查询）"**);  
 **return**;  
 }  
 **if** (!isInit){  
 jl12.setText(Space+**"请先输入init进行初始化操作！"**);  
 **return**;  
 }  
 **else if**(command.toLowerCase().equals(**"save"**)){  
 saveCommand();  
 }  
 **else if**(command.toLowerCase().equals(**"showinfo"**)){  
 jl2.setText(Space+**"当前页面 "**+Space+line);  
 jl3.setText(Space+**"总页数 "**+Space+LINES.length/10);  
 jl4.setText(Space+**"总行数 "**+Space+LINES.length);  
 jl5.setText(Space+**"训练集样本数"**+Space+par.getTrainNum());  
 jl6.setText(Space+**"验证集样本数"**+Space+par.getTestNum());  
 jl7.setText(Space+**"当前工作目录"**+Space+System.getProperty(**"user.dir"**));  
 jl8.setText(Space+**"操作系统 "**+Space+System.getProperty(**"os.name"**));  
 jl9.setText(Space+**"使用者 "**+Space+System.getProperty(**"user.name"**));  
 jl10.setText(Space+**""**);  
 jl11.setText(Space+**""**);  
 jl12.setText(Space+**""**);  
 }  
 **else if**(command.toLowerCase().equals(**"clear"**)){  
 System.out.println(**"Down, Clear ALL!"**);  
 line=0;  
 jl2.setText(Space+**"Line 1"**);  
 jl3.setText(Space+**"Line 2"**);  
 jl4.setText(Space+**"Line 3"**);  
 jl5.setText(Space+**"Line 4"**);  
 jl6.setText(Space+**"Line 5"**);  
 jl7.setText(Space+**"Line 6"**);  
 jl8.setText(Space+**"Line 7"**);  
 jl9.setText(Space+**"Line 8"**);  
 jl10.setText(Space+**"Line 9"**);  
 jl11.setText(Space+**"Line 10"**);  
 jl12.setText(Space+**"Page："**+Space+ **"0"**);  
 }  
 **else if**(command.toLowerCase().equals(**"next"**)){  
 addLine();  
 jl12.setText(Space+**"Page:"**+ Space + line);  
 updateDisplay();  
 }  
 **else if**(command.toLowerCase().equals(**"last"**)){  
 subLine();  
 jl12.setText(Space+**"Page:"**+ Space + line);  
 updateDisplay();  
 }  
 **else if**(command.toLowerCase().equals(**"load"**)){  
 openFile();  
 }  
 **else if** (command.toLowerCase().equals(**"autoload"**)){  
 file=**new** File(**"DataToTest.txt"**);  
 **try** {  
 BufferedReader tdata=**new** BufferedReader(**new** FileReader(file));  
 String line=**null**;  
 **while**((line=tdata.readLine())!=**null**){  
 TData.add(line);  
 }  
 tdata.close();  
 jl12.setText(Space+**"Auto Load Done! Now you can test data to check the model!"**);  
 } **catch** (IOException e2) {  
 System.out.println(e2);  
 }  
 }  
 **else if**(command.toLowerCase().equals(**"autotest"**)){  
 **if** (TData.isEmpty()){  
 jl12.setText(Space+**"Please Open the Test File to load the Data! Or input autoload to load the data！"**);  
 **return**;  
 }  
 **else** {  
 autoTest();  
 NumberFormat nf = NumberFormat.getNumberInstance();  
 nf.setMaximumFractionDigits(1);  
 System.out.println(RightCount+**" "**+FaultCount);  
 **try**{  
 **float** acc1 = RightCount/(RightCount + FaultCount)\*100;  
 jl12.setText(Space+Space+**"Please Wait For Calculate!"**);  
 jl12.setText(Space+**"决策树(Descision Tree) Model 准确率： "**+Float.parseFloat(nf.format(acc1)) + **"% 支持向量机(SVM) Model 准确率为： "** + Float.parseFloat(nf.format(ZZB\_SVM.main()))+**"%"**);  
 }**catch** (IOException e){  
 System.out.println(**"这他么都能给报错？我不信！!"**);  
 }  
 RightCount = 0;  
 FaultCount = 0;  
 }  
 }  
 **else**{  
 comm = command.split(**" "**);  
 **if** (comm[0].toLowerCase().equals(**"test"**) && comm.length<2) {  
 System.out.println(**"Test Ready NOW!"**);  
 Object[] test;  
 **if**(TData.isEmpty()) {  
 test = **new** Object[]{**"-3.0"**, **"2.0"**, **"2.0"**, **"1.0"**, **"0.0"** };  
 }  
 **else** {  
 **if** (nextTimes<TData.size()) {  
 test = TData.get(nextTimes).split(**" "**);  
 nextTimes++;  
 }  
 **else** {  
 test = TData.get(TData.size()-1).split(**" "**);  
 }  
 }  
 String res=**""**;  
 res=TestData.TestData(tree, Test\_Names,test,res);  
 String tdata = **""**;  
 **for** (**int** i=0;i<test.length;++i){  
 tdata =tdata + test[i] + **" | "**;  
 }  
 jl12.setText(Space+tdata+**" "**+res);  
 }  
  
 **else if**(comm[0].toLowerCase().equals(**"test"**) && comm.length-1==Test\_Names.length){  
 Object[] test = **new** Object[Test\_Names.length];  
 **for** (**int** i=0;i<test.length;++i){  
 test[i]=comm[i+1];  
 }  
 String res=**""**;  
 res=TestData.TestData(tree, Test\_Names,test,res);  
 jl12.setText(Space+res);  
 }  
 }  
 }  
 **private void** init(){  
  
 f=**new** Frame(**"The Graduation Design Windows form Zhang Zhaobo for Teachers!"**);  
 f.setBounds(300, 100, 800, 600);  
 f.setLayout(**new** GridLayout(14,1));  
 ta=**new** TextField(50);  
 mb=**new** MenuBar();  
 m=**new** Menu(**"File"**);  
 closeItem=**new** MenuItem(**"Exit"**);  
 openItem=**new** MenuItem(**"Open"**);  
 saveItem=**new** MenuItem(**"Save"**);  
 subm=**new** Menu(**"Show"**);  
 subItem1=**new** MenuItem(**"Next"**);  
 subItem=**new** MenuItem(**"Last"**);  
 subItem4=**new** MenuItem(**"Clear"**);  
 subm.add(subItem);  
 subm.add(subItem1);  
 subm.add(subItem4);  
 m.add(subm);  
 m.add(openItem);  
 m.add(saveItem);  
 m.add(closeItem);  
 Run = **new** Menu(**"Run"**);  
 subItem2 = **new** MenuItem(**"Auto Test"**);  
 subItem3 = **new** MenuItem(**"Help"**);  
 Run.add(subItem2);  
 Run.add(subItem3);  
 mb.add(m);  
 mb.add(Run);  
 but = **new** Button(**"Execute !"**);  
 openDialog=**new** FileDialog(f,**"I wanna to open"**,FileDialog.LOAD);  
 saveDialog=**new** FileDialog(f,**"I wanna to save"**,FileDialog.SAVE);  
 f.setMenuBar(mb);  
 jp1 = **new** JPanel();  
 jl1 = **new** JLabel(**"This is the Code Line for Command!"**);  
 jl1.setSize(300,40);  
 jp1.add(jl1);  
 f.add(jp1);  
  
 jp2 = **new** JPanel();  
 jp2.add(ta);  
 jp2.add(but);  
 f.add(jp2);  
  
 jp3 = **new** JPanel();  
 jl2 = **new** JLabel(Space+**"Line 1"**,SwingConstants.LEFT);  
 jp3.add(jl2);  
 jp3.setLayout(**new** FlowLayout(FlowLayout.LEFT));  
 f.add(jp3);  
  
 jp4 = **new** JPanel();  
 jl3 = **new** JLabel(Space+**"Line 2"**,SwingConstants.LEFT);  
 jp4.add(jl3);  
 jp4.setLayout(**new** FlowLayout(FlowLayout.LEFT));  
 f.add(jp4);  
  
 jp5 = **new** JPanel();  
 jl4 = **new** JLabel(Space+**"Line 3"**,SwingConstants.LEFT);  
 jp5.add(jl4);  
 jp5.setLayout(**new** FlowLayout(FlowLayout.LEFT));  
 f.add(jp5);  
  
 jp6 = **new** JPanel();  
 jl5 = **new** JLabel(Space+**"Line 4"**,SwingConstants.LEFT);  
 jp6.add(jl5);  
 jp6.setLayout(**new** FlowLayout(FlowLayout.LEFT));  
 f.add(jp6);  
  
 jp7 = **new** JPanel();  
 jl6 = **new** JLabel(Space+**"Line 5"**,SwingConstants.LEFT);  
 jp7.add(jl6);  
 jp7.setLayout(**new** FlowLayout(FlowLayout.LEFT));  
 f.add(jp7);  
  
 jp8 = **new** JPanel();  
 jl7 = **new** JLabel(Space+**"Line 6"**,SwingConstants.LEFT);  
 jp8.add(jl7);  
 jp8.setLayout(**new** FlowLayout(FlowLayout.LEFT));  
 f.add(jp8);  
  
 jp9 = **new** JPanel();  
 jl8 = **new** JLabel(Space+**"Line 7"**,SwingConstants.LEFT);  
 jp9.add(jl8);  
 jp9.setLayout(**new** FlowLayout(FlowLayout.LEFT));  
 f.add(jp9);  
  
 jp10 = **new** JPanel();  
 jl9 = **new** JLabel(Space+**"Line 8"**,SwingConstants.LEFT);  
 jp10.add(jl9);  
 jp10.setLayout(**new** FlowLayout(FlowLayout.LEFT));  
 f.add(jp10);  
  
 jp11 = **new** JPanel();  
 jl10 = **new** JLabel(Space+**"Line 9"**,SwingConstants.LEFT);  
 jp11.add(jl10);  
 jp11.setLayout(**new** FlowLayout(FlowLayout.LEFT));  
 f.add(jp11);  
  
 jp12 = **new** JPanel();  
 jl11 = **new** JLabel(Space+**"Line 10"**,SwingConstants.LEFT);  
 jp12.add(jl11);  
 jp12.setLayout(**new** FlowLayout(FlowLayout.LEFT));  
 f.add(jp12);  
  
 jp13 = **new** JPanel();  
 jl12 = **new** JLabel(**""**);  
 jp13.add(jl12);  
 jp13.setLayout(**new** FlowLayout(FlowLayout.LEFT));  
 f.add(jp13);  
  
 jp14 = **new** JPanel();  
 but1 = **new** Button(**"CLEAR"**);  
 but2 = **new** Button(**"NEXT"**);  
 jp14.add(but1);  
 jp14.add(but2);  
 f.add(jp14);  
  
 f.addWindowListener(**new** MyWin());  
 event();  
 f.setVisible(**true**);  
 f.setResizable(**true**);  
 }  
  
 **private void** event(){  
 f.addWindowListener(**new** WindowAdapter() {  
 @Override  
 **public void** windowClosing(WindowEvent e) {  
 *// TODO Auto-generated method stub* System.exit(0);  
  
 }  
 });  
 saveItem.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 *// TODO Auto-generated method stub* saveCommand();  
 }  
 });  
 openItem.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 *// TODO Auto-generated method stub* openFile();  
 }  
 });  
 closeItem.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 *// TODO Auto-generated method stub* System.exit(0);  
 }  
 });  
  
 f.addWindowListener(**new** WindowAdapter() {  
 **public void** windowClosing(WindowEvent e){  
 System.exit(0);  
 }  
  
 });  
  
 but.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 *// TODO Auto-generated method stub* System.out.println(**"Put Down the Button to Execute the Command!"**);  
 dealCommand(ta.getText());  
 }  
 });  
  
 but.addMouseListener(**new** MouseAdapter() {  
 **private int** count=0;  
 **private int** clickCount=1;  
 **public void** mouseEntered(MouseEvent e){  
 System.out.println(**"Entered The EXECUTE BUTTON!"**+count++);  
 }  
 **public void** mouseClicked(MouseEvent e){  
 **if**(e.getClickCount()==2){  
 System.out.println(**"Double Click!"**);  
 }**else** System.out.println(**"Click : "**+clickCount++);  
 }  
 });  
 ta.addKeyListener(**new** KeyAdapter() {  
 **public void** keyPressed(KeyEvent e){  
 **if**(e.getKeyCode()==KeyEvent.VK\_ENTER){  
 System.out.println(**"Put Down the Enter to Execute the Command!"**);  
 dealCommand(ta.getText());  
 }  
 }  
 });  
 but1.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 *// TODO Auto-generated method stub* System.out.println(**"Put Down the Button1 to Clear the Data!"**);  
 line=0;  
 jl2.setText(Space+**"Line 1"**);  
 jl3.setText(Space+**"Line 2"**);  
 jl4.setText(Space+**"Line 3"**);  
 jl5.setText(Space+**"Line 4"**);  
 jl6.setText(Space+**"Line 5"**);  
 jl7.setText(Space+**"Line 6"**);  
 jl8.setText(Space+**"Line 7"**);  
 jl9.setText(Space+**"Line 8"**);  
 jl10.setText(Space+**"Line 9"**);  
 jl11.setText(Space+**"Line 10"**);  
 }  
 });  
 but2.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 *// TODO Auto-generated method stub* dealCommand(**"next"**);  
 }  
 });  
 but2.addKeyListener(**new** KeyAdapter() {  
 **public void** keyPressed(KeyEvent e){  
 System.out.println(KeyEvent.getKeyText(e.getKeyCode())+**" \*\*\*\*\*\*>>>> "**+e.getKeyCode());  
 **if**(e.getKeyCode()==KeyEvent.VK\_ENTER){  
 System.out.println(**"Put Down the Enter to Change the Data!"**);  
 updateDisplay();  
 }  
 }  
 });  
 subItem2.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 *// TODO Auto-generated method stub* dealCommand(**"autotest"**);  
 }  
 });  
 subItem.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 *// TODO Auto-generated method stub* dealCommand(**"next"**);  
 }  
 });  
 subItem1.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 *// TODO Auto-generated method stub* dealCommand(**"last"**);  
 }  
 });  
 subItem4.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 *// TODO Auto-generated method stub* dealCommand(**"clear"**);  
 }  
 });  
 subItem4.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 *// TODO Auto-generated method stub* dealCommand(**"help"**);  
 }  
 });  
 }  
}

# Java 基于Hadoop分布式计算代码

## 【hadoopClear.java】

**import** org.apache.hadoop.conf.Configuration;  
 **import** org.apache.hadoop.fs.Path;  
 **import** org.apache.hadoop.io.IntWritable;  
 **import** org.apache.hadoop.io.Text;  
 **import** org.apache.hadoop.mapreduce.Job;  
 **import** org.apache.hadoop.mapreduce.lib.input.FileInputFormat;  
 **import** org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;  
*/\*\*  
 \* Created by ZZB on 2018/6/10.  
 \*/***public class** hadoopClear {  
 **public static void** main(String[] args)**throws** Exception{  
 *//创建配置对象* Configuration conf = **new** Configuration();  
 *//创建job对象* Job job = Job.getInstance(conf,**"hadoopClear"**);  
 *//设置运行job的类* job.setJarByClass(hadoopClear.**class**);  
 *//设置mapper 类* job.setMapperClass(ZZB\_Mapper.**class**);  
 *//设置reduce 类* job.setReducerClass(ZZB\_Reducer.**class**);  
 *//设置map输出的key value* job.setMapOutputKeyClass(Text.**class**);  
 job.setMapOutputValueClass(IntWritable.**class**);  
 *//设置reduce 输出的 key value* job.setOutputKeyClass(Text.**class**);  
 job.setOutputValueClass(IntWritable.**class**);  
 *//设置输入输出的路径* FileInputFormat.setInputPaths(job, **new** Path(args[1]));  
 FileOutputFormat.setOutputPath(job, **new** Path(args[2]));  
 *//提交job* **boolean** b = job.waitForCompletion(**true**);  
 **if**(!b){  
 System.***out***.println(**"wordcount task fail!"**);  
 }  
 }  
}

## 【ZZP\_Mapper.java】

**import** org.apache.hadoop.io.IntWritable;  
 **import** org.apache.hadoop.io.LongWritable;  
 **import** org.apache.hadoop.io.Text;  
 **import** org.apache.hadoop.mapreduce.Mapper;  
  
 **import** java.io.IOException;  
  
*/\*\*  
 \* Created by ZZB on 2018/6/10.  
 \*/***public class** ZZB\_Mapper **extends** Mapper<LongWritable, Text,Text,IntWritable>{  
 **protected void** map(LongWritable key, Text value, Context context) **throws** IOException, InterruptedException{  
 *//得到输入的每一行数据* String line = value.toString();  
 *//通过空格分隔* String[] values = line.split(**" "**);  
 line = **""**;  
 **for**(**int** i=0; i<values.**length**-1 ;++i)  
 {  
 line += (values[i]+**" "**);  
 }  
 context.write(**new** Text(line), **new** IntWritable((**int**)Float.*parseFloat*(values[values.**length**-1])));  
 }  
}

## 【ZZB\_Reducer.java】

**import** org.apache.hadoop.io.IntWritable;  
 **import** org.apache.hadoop.io.Text;  
 **import** org.apache.hadoop.mapreduce.Reducer;  
  
 **import** java.io.IOException;  
 **import** java.util.HashMap;  
 **import** java.util.Map;  
 **import** java.util.Iterator;  
  
*/\*\*  
 \* Created by ZZB on 2018/6/10.  
 \*/***public class** ZZB\_Reducer **extends** Reducer<Text, IntWritable, Text, IntWritable> {  
 **protected void** reduce(Text key, Iterable<IntWritable> values, Context context) **throws** IOException, InterruptedException {  
 Integer count=0;  
 **int** times = 0;  
 **for** (IntWritable value : values) {  
 count+=value.get();  
 times++;  
 }  
 **if** (times==1){  
 context.write(**new** Text(key),**new** IntWritable(count));  
 }  
 **else if** (count%times==0){  
 context.write(**new** Text(key),**new** IntWritable(count/times));  
 }  
 }

}