```java
package classes;

import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.io.UnsupportedEncodingException;
import java.util.ArrayList;
import java.util.List;
import java.util.Random;
import java.util.concurrent.TimeUnit;

/**
 * created by lgcaobianco on 2018-04-14
 */

public class Adaline {
    List<Double[]> inputs;
    List<Double[]> conjuntoOperacao;
    List<Double> coeficientesIniciais = new ArrayList<Double>();
    List<Double> coeficientesFinais = new ArrayList<Double>();
    List<Double> u = new ArrayList<Double>(35);
    int contadorEpocas = 0;
    double taxaAprendizagem;
    double erroQuadraticoAtual = 0;
    double erroQuadraticoAnterior = 0;


    public Adaline(String nomeArquivo, String tipoArquivo) {
        LeitorPontosEntrada leitor = new LeitorPontosEntrada(nomeArquivo, tipoArquivo);
        this.inputs = leitor.extrairPontos();
        for (int i = 0; i < (inputs.get(0).length - 1); i++) {
            Random random = new Random();
            double randomDouble = random.nextDouble();
            coeficientesIniciais.add(randomDouble);
            coeficientesFinais.add(randomDouble);
            taxaAprendizagem = 0.0025;
        }
    }

    public void construirConjuntoOperacao(String nomeArquivo, String tipoArquivo) {
        LeitorPontosEntrada leitor = new LeitorPontosEntrada(nomeArquivo, tipoArquivo);
        this.conjuntoOperacao = leitor.extrairPontos();
    }

```

```
44      public void imprimirMatrizInputs() {
45          for (int i = 0; i < inputs.size(); i++) {
46              for (int j = 0; j < inputs.get(i).length; j++)
     {
47                  System.out.print(inputs.get(i)[j] + " ");
48              }
49              System.out.println();
50          }
51      }
52
53      public void imprimirMatrizCoeficientes(String
    qualMatriz) {
54          if (qualMatriz.equals("inicial")) {
55              for (int i = 0; i < coeficientesIniciais.size(
    ); i++) {
56                  System.out.print(coeficientesIniciais.get(
    i) + " ");
57              }
58              System.out.println();
59          }
60          if (qualMatriz.equals("final")) {
61              for (int i = 0; i < coeficientesFinais.size();
     i++) {
62                  System.out.print(coeficientesFinais.get(i)
     + " ");
63              }
64              System.out.println();
65          }
66      }
67
68      public double combinador(int linhaMatriz) {
69          double somatorio = 0;
70          for (int i = 0; i < coeficientesFinais.size(); i++
    ) {
71              somatorio += coeficientesFinais.get(i) *
    inputs.get(linhaMatriz)[i];
72          }
73          return somatorio;
74      }
75
76      public void treinarAdaline() {
77          double novoElemento = 0, u = 0;
78          for (int i = 0; i < inputs.size(); i++) {
79              u = combinador(i);
80              for (int j = 0; j < coeficientesFinais.size();
     j++) {
81                  novoElemento = coeficientesFinais.get(j)
82                          + (taxaAprendizagem * (inputs.get(
    i)[5] - u) * inputs.get(i)[j]);
83                  coeficientesFinais.set(j, novoElemento);
```

```
 84              }
 85          }
 86      }
 87
 88      public double calcularErroQuadratico() {
 89          double eqm = 0, u = 0;
 90          for (int i = 0; i < inputs.size(); i++) {
 91              u = combinador(i);
 92              eqm += Math.pow((inputs.get(i)[5] - u), 2);
 93          }
 94
 95          return eqm / inputs.size();
 96      }
 97
 98      public double combinadorFinal(int linhaMatriz) {
 99          double somatorio = 0;
100          for (int i = 0; i < coeficientesFinais.size(); i
    ++) {
101              somatorio += coeficientesFinais.get(i) *
    conjuntoOperacao.get(linhaMatriz)[i];
102          }
103          return somatorio;
104      }
105
106      public int classificarAmostra(double u) {
107          if (u >= 0) {
108              return 1;
109          } else {
110              return -1;
111          }
112      }
113
114      public static void main(String[] args)
115              throws InterruptedException,
    FileNotFoundException, UnsupportedEncodingException {
116          double epsilon = Math.pow(10, -6);
117          List<Adaline> adalines = new ArrayList<Adaline>(5
    );
118          PrintWriter writer = new PrintWriter("/home/
    lgcaobianco/repositorios/epc-rna/epc2/erroqm.csv", "UTF-8
    ");
119          for (int i = 0; i < 5; i++) {
120              Adaline aux = new Adaline("/home/lgcaobianco/
    repositorios/epc-rna/epc2/src/classes/base/inputs", ".csv
    ");
121              aux.construirConjuntoOperacao("/home/
    lgcaobianco/repositorios/epc-rna/epc2/src/classes/base/
    inputs",
122                      ".csv");
123              adalines.add(aux);
```

```
124             System.out.println("Adaline instanciada.");
125             TimeUnit.MINUTES.sleep(1 / 5);
126 
127         }
128 
129         for (Adaline adaline : adalines) {
130             System.out.println("======================
    Adaline ======================");
131             writer.println("======================
    Adaline ======================");
132             // fase de treino
133             System.out.println("
    Treinamento                 ");
134             adaline.erroQuadraticoAtual = adaline.
    calcularErroQuadratico();
135             while (Math.abs(adaline.erroQuadraticoAtual -
     adaline.erroQuadraticoAnterior) > epsilon) {
136                 adaline.treinarAdaline();
137                 adaline.erroQuadraticoAnterior = adaline.
    erroQuadraticoAtual;
138                 adaline.erroQuadraticoAtual = adaline.
    calcularErroQuadratico();
139                 adaline.contadorEpocas++;
140                 writer.write(adaline.contadorEpocas +
    ", " + adaline.erroQuadraticoAtual+"\n");
141             }
142             System.out.print("Os coeficientes inciais são
    : ");
143             adaline.imprimirMatrizCoeficientes("inicial")
    ;
144 
145             System.out.print("Os coeficientes finais são
    : ");
146             adaline.imprimirMatrizCoeficientes("final");
147             System.out.println(
148                     "O erro quadratico terminou em: " + (
    adaline.erroQuadraticoAtual - adaline.
    erroQuadraticoAnterior)
149                             + " com: " + adaline.
    contadorEpocas + " epocas.");
150             System.out.println(
    "                                        ");
151             System.out.println();
152 
153             // fase de operacao
154             System.out.println("                Operação
                ");
155 
156             for (int i = 0; i < adaline.conjuntoOperacao.
    size(); i++) {
```

```
157                 int resultado = adaline.
    classificarAmostra(adaline.combinadorFinal(i));
158                 System.out.println((i + 1) + ", " +
    resultado);
159             }
160             System.out.println(
    "                                                  ");
161             System.out.println(
    "====================================================");
162 
163         }
164 
165         writer.close();
166     }
167 
168 }
169 
```