



Faculty of Engineering and Technology

Computer Science Department

COMP338 – ARTIFICIAL INTELLIGENCE

	Students Name	Students Number
1	Yousef Sharbi	1202057
2	Anas karakra	1200467

Project 3: Machine Learning (Regression)

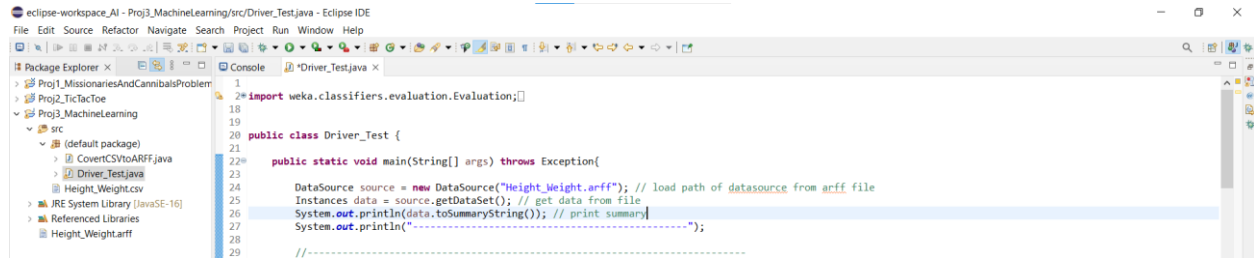
This class convert .csv file to .arff file:

```
1 // Yousef Sharbi 1202057
2 // Anas karakra 1200467
3
4 import weka.core.Instances;
5 import weka.core.converters.ArffSaver;
6 import weka.core.converters.CSVLoader;
7 import java.io.File;
8
9 public class ConvertCSVtoARFF {
10     public static void main(String[] args) {
11
12         try {
13             CSVLoader loader = new CSVLoader();
14             loader.setSource(new File("Height_Weight.csv")); // load csv file
15             Instances data = loader.getDataSet(); // get all dataset from csv file
16
17             ArffSaver saver = new ArffSaver();
18             saver.setInstances(data); // set the data inside arff file
19             saver.setFile(new File("Height_Weight.arff"));
20             saver.writeToFile();
21
22             System.out.println("the file converted from csv to arff successfully");
23         } catch (Exception e) {
24             System.out.println("error in file !!!!!");
25         }
26     }
27 }
28
29
30
31
```

Output:

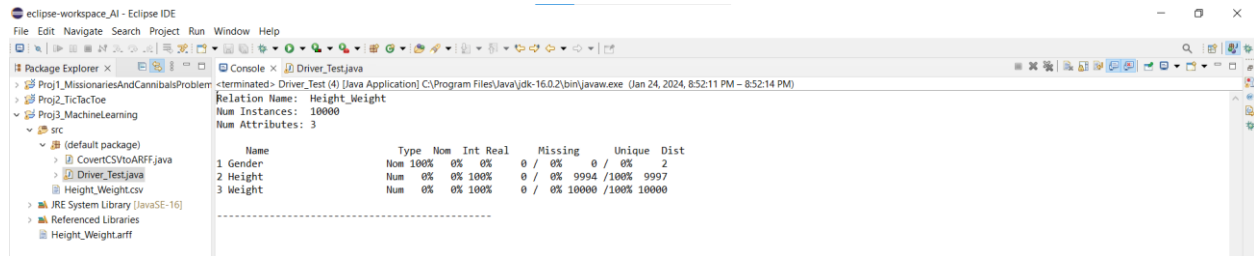
```
the file converted from csv to arff successfully
```

print data summary from dataset:



```
1  import weka.classifiers.evaluation.Evaluation;
18
19
20 public class Driver_Test {
21
22     public static void main(String[] args) throws Exception{
23
24         DataSource source = new DataSource("Height_Weight.arff"); // load path of datasource from arff file
25         Instances data = source.getDataSet(); // get data from file
26         System.out.println(data.toSummaryString()); // print summary
27         System.out.println("-----");
28
29         //-----
```

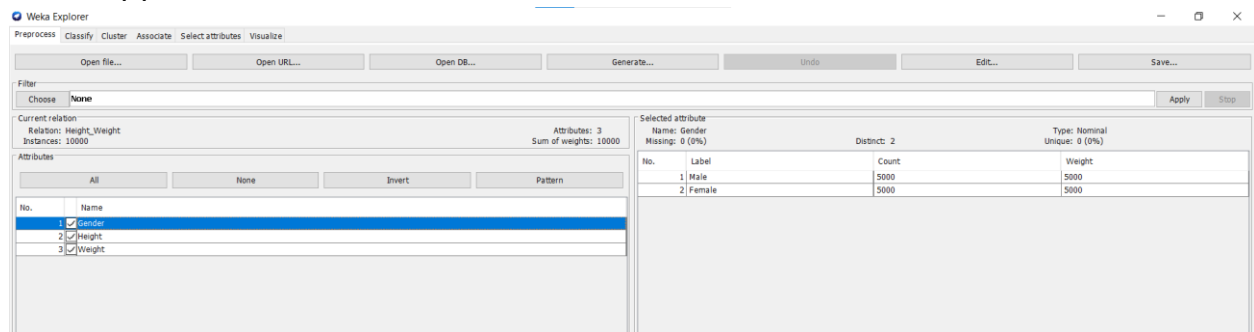
Output:



```
terminated> Driver_Test (4) [Java Application] C:\Program Files\Java\jdk-16.0.2\bin\javaw.exe (Jan 24, 2024, 8:52:11 PM - 8:52:14 PM)
Relation Name: Height_Weight
Num Instances: 10000
Num Attributes: 3

  Name      Type  Num  Int  Real  Missing  Unique  Dist
1 Gender    Nom 100%  0%  0%  0 / 0%  0 / 0%  2
2 Height    Num  0%  0% 100% 0 / 0% 9994 /100% 9997
3 Weight    Num  0%  0% 100% 0 / 0% 10000 /100% 10000
-----
```

Weka application:



The Weka Explorer application is shown with the 'Select attributes' tab selected. The 'Current relation' is 'Height_Weight' with 10000 instances and 3 attributes. The 'Selected attribute' is 'Gender' with 2 distinct values and a nominal type. The 'Attributes' list shows 'Gender', 'Height', and 'Weight' selected. The 'Selected attribute' table shows the distribution of 'Gender'.

No.	Label	Count	Weight
1	Male	5000	5000
2	Female	5000	5000

1) convert the height from inches to cms and the weight from pounds to kilograms:

```
Height_Weight.arff
32 //-----
33 // 1)
34 // * convert the height from inches to cms
35 Attribute heightAttribute = data.attribute("Height"); // set heightAttribute
36
37 for (int i = 0; i < data.size(); i++) {
38     double heightInches = data.instance(i).value(heightAttribute); // get height instance value
39
40     double heightCms = heightInches * 2.54; // convert it to cms
41     data.instance(i).setValue(heightAttribute, heightCms); // set the new value of cms
42 }
43
44 // * convert the weight from pounds to kilograms
45 Attribute weightAttribute = data.attribute("Weight"); // set weightAttribute
46
47 for (int i = 0; i < data.size(); i++) {
48     double weightPounds = data.instance(i).value(weightAttribute); // get weight instance value
49
50     double weightKgs = weightPounds * 0.453592; // convert it to kilograms
51     data.instance(i).setValue(weightAttribute, weightKgs); // set the new value of kilograms
52 }
53
54
55 //-----
```

2) Print the main statistics of the features (i.e., mean, median, standard deviation, min, and max values) in a proper table:

```

56 //-----
57 // 2)
58
59 int numOfAttributes = data.numAttributes(); // number of attributes
60
61 for (int i = 0; i < numOfAttributes; i++) {
62     // for height
63     if (data.attribute(i).equals(heightAttribute)) {
64         System.out.println(" - Attribute " + (i+1) + ": " + data.attribute(i).name());
65         AttributeStats as = data.attributeStats(i); // use attributeStats to get stats
66         Stats s = as.numericStats; // set numericStats to get the main statistics
67
68         double[] values = data.attributeToDoubleArray(i); // to get Median
69         Arrays.sort(values);
70
71         System.out.println(" Mean: " + s.mean + ", Median: " + calculateMedian(values) + ", Standard Deviation: " + s.stdDev +
72             ", Min value: " + s.min + ", Max value: " + s.max);
73     }
74     // for weight
75     else if (data.attribute(i).equals(weightAttribute)) {
76         System.out.println(" - Attribute " + (i+1) + ": " + data.attribute(i).name());
77         AttributeStats as = data.attributeStats(i); // use attributeStats to get stats
78         Stats s = as.numericStats; // set numericStats to get the main statistics
79
80         double[] values = data.attributeToDoubleArray(i); // to get Median
81         Arrays.sort(values);
82
83         System.out.println(" Mean: " + s.mean + ", Median: " + calculateMedian(values) + ", Standard Deviation: " + s.stdDev +
84             ", Min value: " + s.min + ", Max value: " + s.max);
85     }
86 }
87
88 System.out.println("\n" + "-----");
89
90 //-----
91
92
```

```

237
238
239
240 public static double calculateMedian(double[] values) {
241     int middle = values.length / 2;
242     if (values.length % 2 == 0) {
243         return (values[middle - 1] + values[middle]) / 2.0;
244     } else {
245         return values[middle];
246     }
247 }
248
249
250
251
```

Output:

```

> JRE System Library [JavaSE-16]
> Referenced Libraries
> Height_Weight.arff
-----
- Attribute 2: Height
Mean: 168.5736817759217, Median: 168.4478978, Standard Deviation: 9.77272143421192, Min value: 137.82835782, Max value: 200.65680468
- Attribute 3: Weight
Mean: 73.22805433663937, Median: 73.12489443737599, Standard Deviation: 14.56413106614006, Min value: 29.347460006183997, Max value: 122.46516754880798
-----
```

3) For each of the following models, you have to split the data into 70% training and 30% test:

```
93 // 3)
94
95 data.setClassIndex(data.numAttributes()-1); // or you can use this code: data.setClass(data.attribute("Weight"));
96
97 int trainSize = (int)(data.numInstances() * 0.7); // size of training data
98 int testSize = data.numInstances() - trainSize; // size of test data
99
100 Instances trainingData = new Instances(data, 0, trainSize); //trainingData: 0 start ... till train size 70
101 Instances testData = new Instances(data, trainSize, testSize); //testData: 70 start ... till 100
102
103
```

4) Select a subset of 100 instances from randomly selected from the dataset and generate the first model (called M1) and test this models performance using appropriate regression metrics:

```
103 //-----
104 // 4)
105
106 data.randomize(new java.util.Random()); // make random to the data
107 Instances dataM1 = new Instances(trainingData, 0, 100); // data of Model 1 (Select a subset of 100 instances)
108
109
110 // Generate the linear regression model (M1)
111 LinearRegression m1 = new LinearRegression();
112 m1.buildClassifier(dataM1); // build the relation
113 System.out.println("Relation M1: ");
114 System.out.print(m1+"\n");
115
116 Evaluation e_m1 = new Evaluation(trainingData); // set the training data on evaluation model
117 e_m1.evaluateModel(m1, testData); // evaluate the model on the test set
118
119 // Print regression metrics
120 System.out.println("\n" + "Regression Metrics for Model (M1):" + "\n");
121 System.out.println("Mean Absolute Error (MAE): " + e_m1.meanAbsoluteError());
122 System.out.println("Root Mean Squared Error (RMSE): " + e_m1.rootMeanSquaredError());
123 System.out.println("Relative Absolute Error (RAE): " + e_m1.relativeAbsoluteError());
124 System.out.println("Correlation coefficient: " + e_m1.correlationCoefficient());
125 System.out.println("\n" + "-----");
126
```

Output:

```
> JRE System Library [JavaSE-16]
> Referenced Libraries
  Height_Weight.arff

Relation M1:

Linear Regression Model

Weight =

      1.045 * Height +
     -98.5208

Regression Metrics for Model (M1):

Mean Absolute Error (MAE): 8.975152480831019
Root Mean Squared Error (RMSE): 9.98000623355699
Relative Absolute Error (RAE): 54.37054813872104
Correlation coefficient: 0.8502130121504533

-----
```

- 5) Select a subset of 1000 instances from randomly selected from the dataset and generate the first model (called M2) and test this models performance using appropriate regression metrics:

```
127 //-----
128 // 5)
129
130 data.randomize(new java.util.Random()); // make random to the data
131 Instances dataM2 = new Instances(trainingData, 0, 1000); // data of Model 2 (Select a subset of 1000 instances)
132
133 // Generate the linear regression model (M2)
134 LinearRegression m2 = new LinearRegression();
135 m2.buildClassifier(dataM2); // build the relation
136 System.out.println("Relation M2: ");
137 System.out.print(m2+"\n");
138
139 Evaluation e_m2 = new Evaluation(trainingData); // set the training data on evaluation model
140 e_m2.evaluateModel(m2, testData); // evaluate the model on the test set
141
142 // Print regression metrics
143 System.out.println("\n" + "Regression Metrics for Model (M2):" + "\n");
144 System.out.println("Mean Absolute Error (MAE): " + e_m2.meanAbsoluteError());
145 System.out.println("Root Mean Squared Error (RMSE): " + e_m2.rootMeanSquaredError());
146 System.out.println("Relative Absolute Error (RAE): " + e_m2.relativeAbsoluteError());
147 System.out.println("Correlation coefficient: " + e_m2.correlationCoefficient());
148 System.out.println("\n" + "-----");
```

Output:

```
> JRE System Library [JavaSE-16]
> Referenced Libraries
  Height_Weight.arff

-----
Relation M2:

Linear Regression Model

Weight =

    1.0437 * Height +
    -97.9889

Regression Metrics for Model (M2):

Mean Absolute Error (MAE): 9.28496854321847
Root Mean Squared Error (RMSE): 10.269561955403626
Relative Absolute Error (RAE): 56.24738189393165
Correlation coefficient: 0.8502130121502433

-----
```

- 6) Select a subset of 5000 instances from randomly selected from the dataset and generate the first model (called M3) and test this models performance using appropriate regression metrics:

```
149 //-----
150 // 6)
151
152 data.randomize(new java.util.Random()); // make random to the data
153 Instances dataM3 = new Instances(trainingData, 0, 5000); // data of Model 3 (Select a subset of 5000 instances)
154
155 // Generate the linear regression model (M3)
156 LinearRegression m3 = new LinearRegression();
157 m3.buildClassifier(dataM3); // build the relation
158 System.out.println("Relation M3: ");
159 System.out.print(m3+"\n");
160
161 Evaluation e_m3 = new Evaluation(trainingData); // set the training data on evaluation model
162 e_m3.evaluateModel(m3, testData); // evaluate the model on the test set
163
164 // Print regression metrics
165 System.out.println("\n" + "Regression Metrics for Model (M3):" + "\n");
166 System.out.println("Mean Absolute Error (MAE): " + e_m3.meanAbsoluteError());
167 System.out.println("Root Mean Squared Error (RMSE): " + e_m3.rootMeanSquaredError());
168 System.out.println("Relative Absolute Error (RAE): " + e_m3.relativeAbsoluteError());
169 System.out.println("Correlation coefficient: " + e_m3.correlationCoefficient());
170 System.out.println("\n" + "-----");
171
172 //-----
```

Output:

```
> Height_Weight.csv
> JRE System Library [JavaSE-16]
> Referenced Libraries
  Height_Weight.arff

-----
Relation M3:

Linear Regression Model

Weight =

    1.0647 * Height +
    -101.8309

Regression Metrics for Model (M3):

Mean Absolute Error (MAE): 8.860976969821895
Root Mean Squared Error (RMSE): 9.871383250933473
Relative Absolute Error (RAE): 53.67888466772786
Correlation coefficient: 0.8502130121504335

-----
```

- 7) Use the entire dataset and generate the first model (called M4) and test this models performance using appropriate regression metrics:

```
173 //7)
174 LinearRegression m4 = new LinearRegression();
175 m4.buildClassifier(trainingData); // build the relation
176 System.out.println("Relation M4: ");
177 System.out.print(m4+"\n");
178
179 // Evaluate the model on the test set
180 Evaluation e_m4 = new Evaluation(trainingData); // set the training data on evaluation model
181 e_m4.evaluateModel(m4, testData); // evaluate the model on the test set
182
183 // Print regression metrics for Model M4
184 System.out.println("\nRegression Metrics for Model (M4):\n");
185 System.out.println("Mean Absolute Error (MAE): " + e_m4.meanAbsoluteError());
186 System.out.println("Root Mean Squared Error (RMSE): " + e_m4.rootMeanSquaredError());
187 System.out.println("Relative Absolute Error (RAE): " + e_m4.relativeAbsoluteError());
188 System.out.println("Correlation coefficient: " + e_m4.correlationCoefficient());
189 System.out.println("\n-----");
190 //
191 //-----
```

Output:

> JRE System Library [JavaSE-16]
> Referenced Libraries
Height_Weight.arff

Relation M4:

Linear Regression Model

Weight =

8.9048 * Gender=Male +
1.0649 * Height +
-110.7855

Regression Metrics for Model (M4):

Mean Absolute Error (MAE): 3.5972860073961304
Root Mean Squared Error (RMSE): 4.531020812938216
Relative Absolute Error (RAE): 21.79198765164259
Correlation coefficient: 0.8502130121505559

8) Print the appropriate performance metrics for Regression and compare the performance of the generated models in your own words:

```
192 //8)
193
194 System.out.println("compare the performance of the generated models:"+ "\n");
195 System.out.println("- Mean Absolute Error(MAE): "+ "\n");
196 System.out.println("M1: "+e_m1.meanAbsoluteError());
197 System.out.println("M2: "+e_m2.meanAbsoluteError());
198 System.out.println("M3: "+e_m3.meanAbsoluteError());
199 System.out.println("M4: "+e_m4.meanAbsoluteError());
200
201 System.out.println("\n" + "Model M4 has the lowest Mean Absolute Error, which M4 accuracy best than the other." + "\n");
202
203 System.out.println("-----" + "\n");
204 System.out.println("- Root Mean Squared Error(RMSE): "+ "\n");
205 System.out.println("M1: "+e_m1.rootMeanSquaredError());
206 System.out.println("M2: "+e_m2.rootMeanSquaredError());
207 System.out.println("M3: "+e_m3.rootMeanSquaredError());
208 System.out.println("M4: "+e_m4.rootMeanSquaredError());
209
210 System.out.println("\n" + "Model M4 has the lowest Root Mean Squared Error, which M4 better precision in predictions than the other." + "\n");
211
212 System.out.println("-----" + "\n");
213 System.out.println("- Relative Absolute Error (RAE): "+ "\n");
214 System.out.println("M1: "+e_m1.relativeAbsoluteError());
215 System.out.println("M2: "+e_m2.relativeAbsoluteError());
216 System.out.println("M3: "+e_m3.relativeAbsoluteError());
217 System.out.println("M4: "+e_m4.relativeAbsoluteError());
218
219 System.out.println("\n" + "Model M4 has the lowest Relative Absolute Error, which M4 better performance in terms of relative accuracy than the other." + "\n");
220
221 System.out.println("-----" + "\n");
222 System.out.println("- Correlation coefficient: "+ "\n");
223 System.out.println("M1: "+e_m1.correlationCoefficient());
224 System.out.println("M2: "+e_m2.correlationCoefficient());
225 System.out.println("M3: "+e_m3.correlationCoefficient());
226 System.out.println("M4: "+e_m4.correlationCoefficient());
227
228 System.out.println("\n" + "All models have the same correlation coefficient, which there is a strong linear relationship between the predicted and actual values.");
229 }
```

Output:

```
compare the performance of the generated models:
- Mean Absolute Error(MAE):
M1: 8.975152488831019
M2: 9.28496854321847
M3: 8.868976969821895
M4: 3.5972860873961304
Model M4 has the lowest Mean Absolute Error, which M4 accuracy best than the other.
-----
- Root Mean Squared Error(RMSE):
M1: 9.98000623355699
M2: 10.269561955483626
M3: 9.871383258933473
M4: 4.531820812938216
Model M4 has the lowest Root Mean Squared Error, which M4 better precision in predictions than the other.
-----
- Relative Absolute Error (RAE):
M1: 54.37054813872104
M2: 56.24738189393165
M3: 53.67888466772786
M4: 21.79198765164259
Model M4 has the lowest Relative Absolute Error, which M4 better performance in terms of relative accuracy than the other.
-----
- Correlation coefficient:
M1: 0.8502130121504533
M2: 0.8502130121502433
M3: 0.8502130121504335
M4: 0.8502130121505559
All models have the same correlation coefficient, which there is a strong linear relationship between the predicted and actual values.
```