CS5100 - Weather Predictor User Manual

Team Information: Neethu Prasad

Nidhi Gupta

Description:

Predict the occurrences of various weather phenomenon like Rain, Fog, Snow, Thunderstorm using a decision tree classifier implemented using C4.5 algorithm suggested by Quinlan

1. Files

The following files are included with the submission

External Library

Weka.jar - External library for J48 classifier.

Java Files

DecisionTree.java - main file.
Evaluation.java - computes the accuracy of given result.
GainRatio.java - Computes the gain ratio of given feature, threshold value and left and right subset data at threshold position.
GrowTree.java - creates a decision tree by recursively selecting the feature with best gain ratio.
J48Decision.java - trains and classifies test data using Weka J48 implementation.
PostTreePruning.java - prunes the inital decision tree if the pruned tree gives better accuracy.
SelectFeature.java - calls GainRatio.java to get the maximum gain ratio for given feature and returns a feature with maximum gain ratio.
ValidateWithPruning.java -classifies the given test and random data with the help of C4.5 decision tree.
Feature.java - data structure for storing a feature name value pair.
Node.java - data structure for a node in the decision tree.
WeatherData.java - data structure for storing input data instance.

Input file(s)

Inputs to C4.5 algorithm
WeatherDataTest.csv - stores 2015 weather data from weatherundergrounds.com
WeatherDataTrain.csv - sttores 2014 weather data from weatherundergrounds.com
randomData.csv - stores random data to be classified
Input to J48 (same train and test data in arff format)
classnameTest.arff stores 2015 weather data and classnameTrain.arff stores 2014 weather data.
fogTest.arff, fogTrain.arff, snowTest.arff, snowTrain.arff, rainTest.arff, rainTrain.arff, thunderTest.arff, thunderTrain.arff

Output file(s)

Output.txt stores the classification result for randomData.csv

2. Software and Hardware Requirements

IDE	Eclipse Mars Release (4.5.0)
JRE	JavaSE-1.8
External library	weka.jar
Processor	Intel(R)Core(™)I5-5200U CPU@2.20GHz
RAM	8GB
OS	64 bit Windows 10

3.Software Installation Instructions

a) Weka.jar - installation instruction and jar file are available at http://www.cs.waikato.ac.nz/ml/weka/downloading.html.
We have also included weka.jar in our zip file
Make sure to use a stable version.

b) JRE - can be downloaded from http://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html

c) Eclipse IDE - Latest Eclipse IDE could be downloaded from https://eclipse.org/downloads/

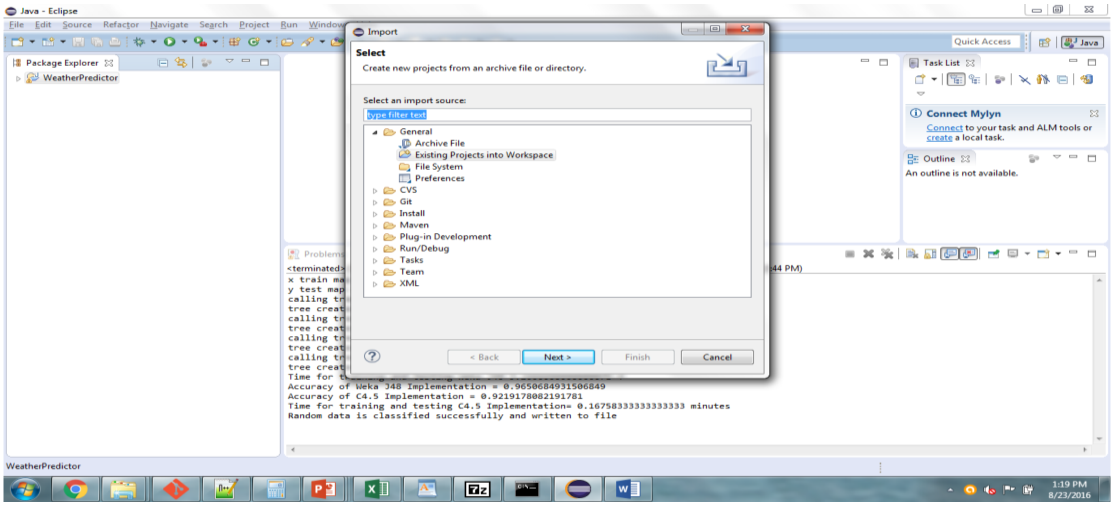
4. Setting Up the Environment

Download the project folder from
https://drive.google.com/drive/folders/0B9m5qbswJ6giMHJZLVp3dER0Vk0

Extract the source folder (Weather-Predictor that contains the source Java and input files) from the zipped folder we just downloaded and save it to a convenient location on computer.

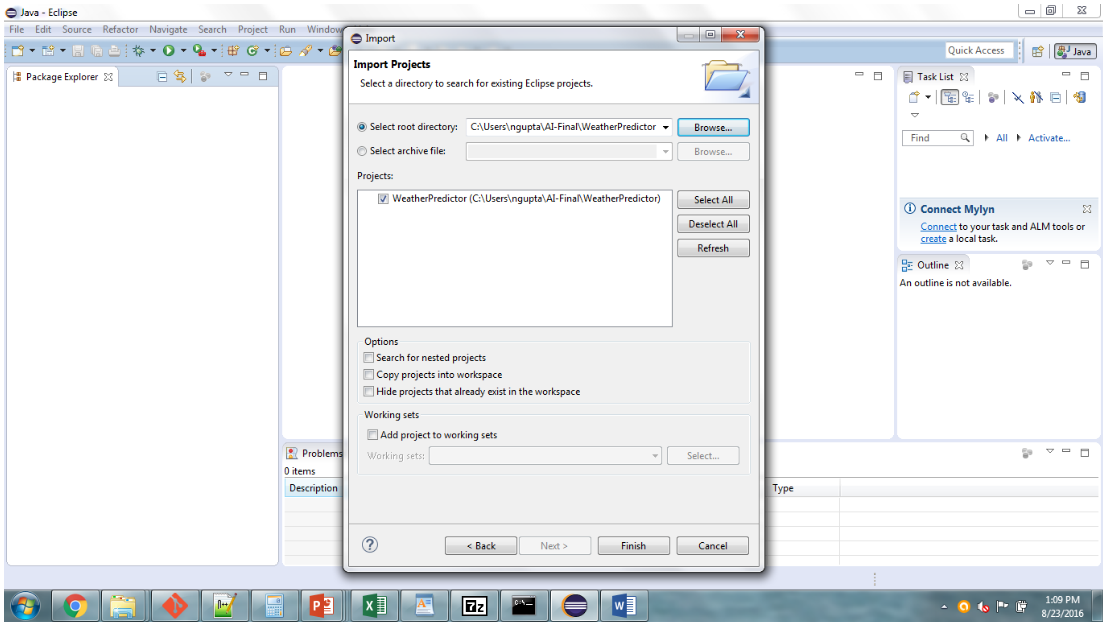
Steps to execute the Weather Predictor Project from Eclipse

1) After opening Eclipse IDE, go to **Imports** in **File** Menu and select **Existing projects into Workspace** inside **General folder**



```
<terminated>
x train ma
y test map
calling tr
tree creat
calling tr
tree creat
calling tr
tree creat
calling tr
tree creat
Time for t
Accuracy of Weka J48 Implementation = 0.9650684931506849
Accuracy of C4.5 Implementation = 0.9219178082191781
Time for training and testing C4.5 Implementation= 0.16758333333333333 minutes
Random data is classified successfully and written to file
```

2) **Import Dialog box** is displayed. Select the **root directory** and browse to the folder **WeatherPredictor** we just extracted.



● Select root directory: C:\Users\ngupta\AI-Final\WeatherPredictor

○ Select archive file:

Projects:

☑ WeatherPredictor (C:\Users\ngupta\AI-Final\WeatherPredictor)

Options

□ Search for nested projects

□ Copy projects into workspace

□ Hide projects that already exist in the workspace

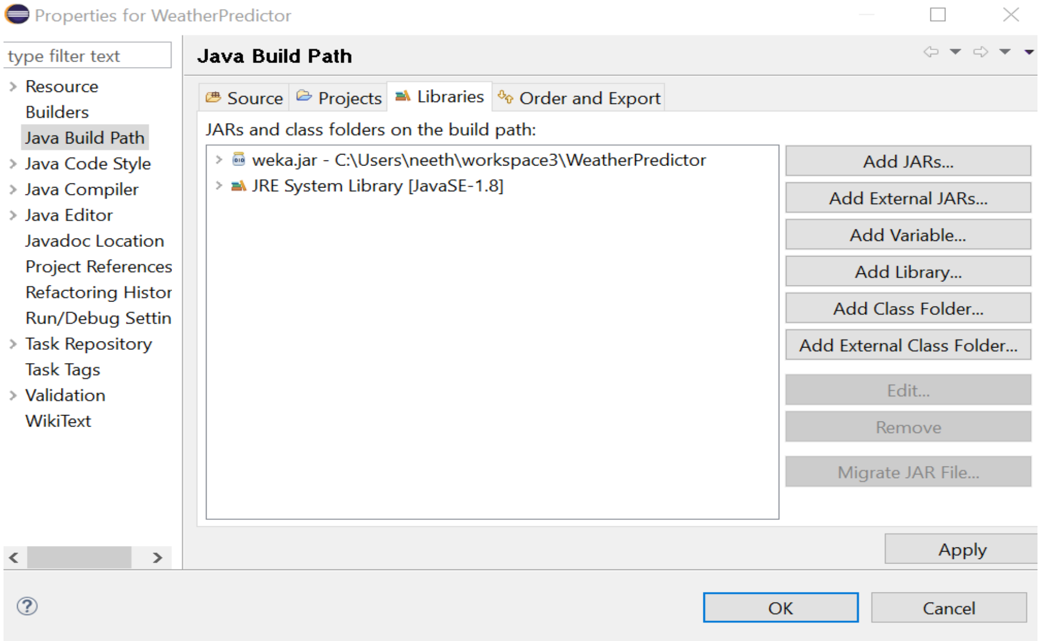
Working sets

□ Add project to working sets

Working sets:

Click on **Finish**

3) Right click the **WeatherPredictor** Project that gets created on **Package Explorer** and select properties and select Java Build Path -> Libraries -> Add External JARs -> weka.jar



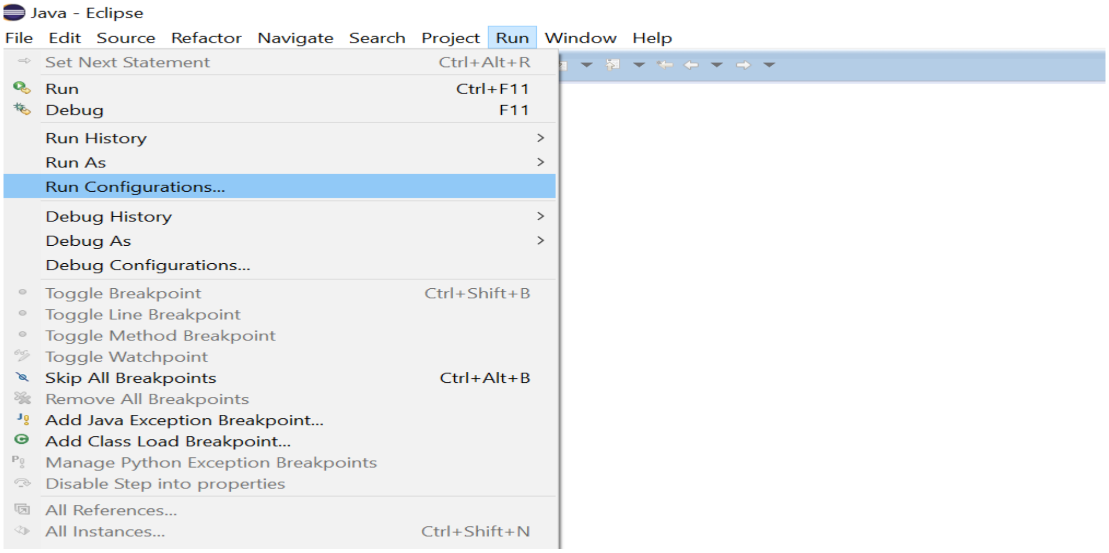
JARs and class folders on the build path:

weka.jar - C:\Users\neeth\workspace3\WeatherPredictor

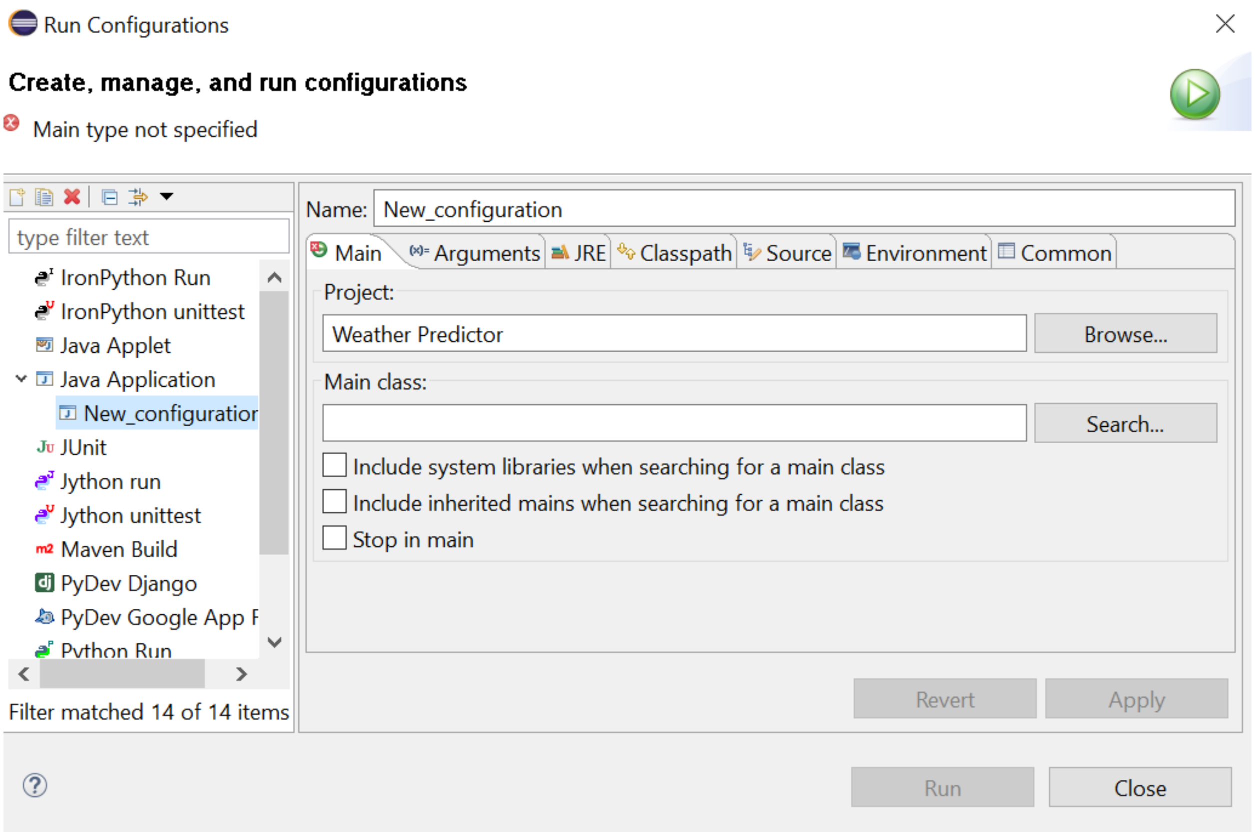
JRE System Library [JavaSE-1.8]

4) Select the **WeatherPredictor** Project that gets created on **Package Explorer** and click on **Run**.

Select **Java ->Run Configurations**



5) Select **Java Application** from left pane



Run Configurations

Create, manage, and run configurations

Main type not specified

Name: New_configuration

Project: Weather Predictor

Main class:

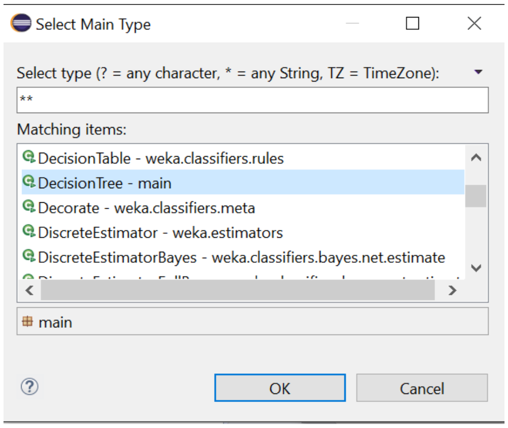
□ Include system libraries when searching for a main class

□ Include inherited mains when searching for a main class

□ Stop in main

Filter matched 14 of 14 items

6) Click on search button beside Main class. A dialog box appears, now select **DecisionTree-main** from the drop down menu, as this Java file holds the main method of our application.



Select Main Type

Select type (? = any character, * = any String, TZ = TimeZone):

**

Matching items:

- DecisionTable - weka.classifiers.rules
- DecisionTree - main
- Decorate - weka.classifiers.meta
- DiscreteEstimator - weka.estimators
- DiscreteEstimatorBayes - weka.classifiers.bayes.net.estimate

main

Hit **Ok** to proceed

7) Click on **Arguments** tab and input the file names in the following order

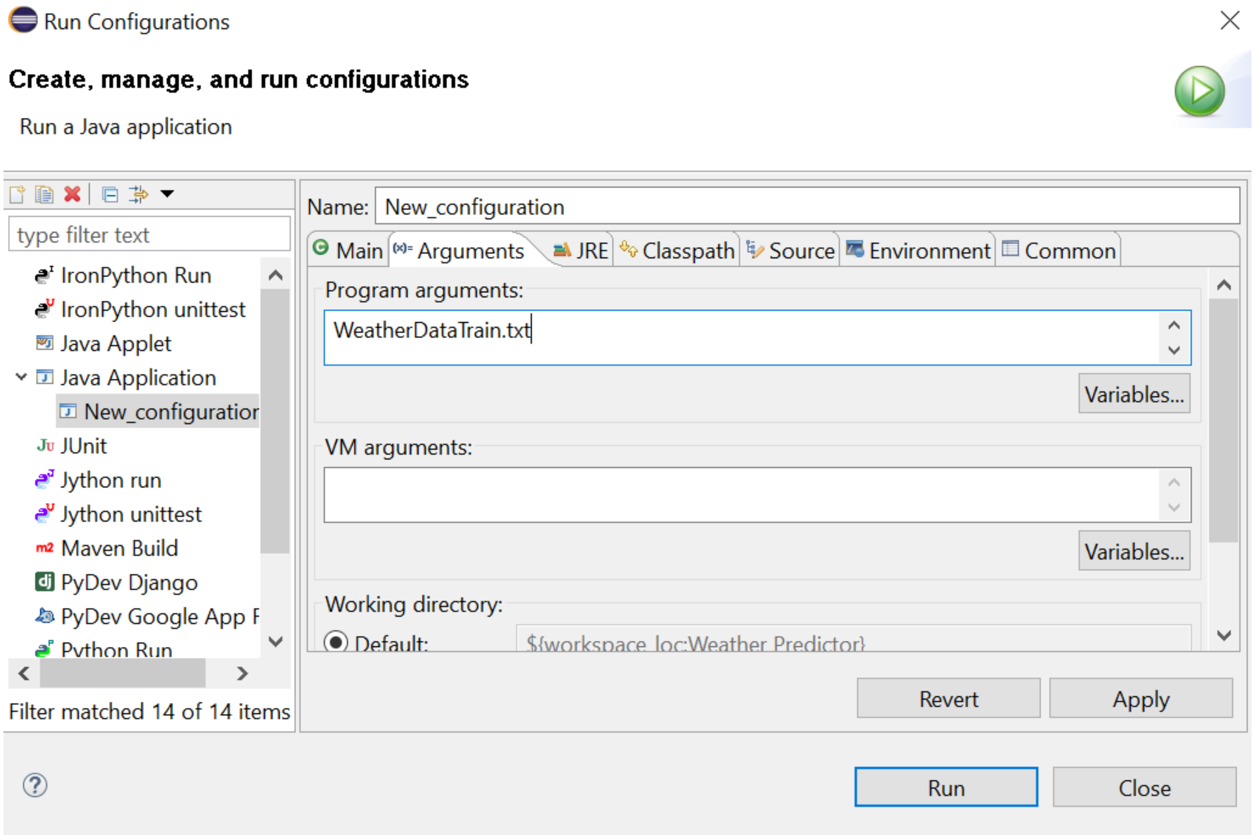
WeatherDataTrain.csv

WeatherDataTest.csv

randomData.csv

output.txt

Click on **Apply**



Run Configurations

Create, manage, and run configurations

Run a Java application

Name: New_configuration

Program arguments: WeatherDataTrain.txt

VM arguments:

Working directory:

● Default: ${workspace_loc:Weather Predictor}

Filter matched 14 of 14 items

8) Click on **Run**. The Java project is then executed and prints the output (performance parameters like Execution time, Prediction accuracy) on console. Additionally, an output.txt file gets created inside the java project that predicts the weather phenomenon for random data sent as input to the program