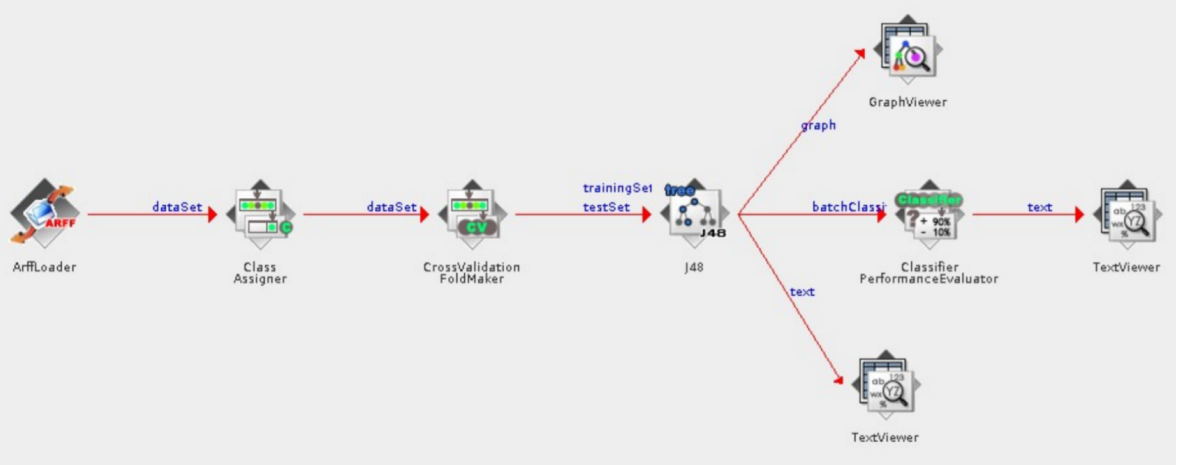
TUTORIAL 3 - MACHINE LEARNING



DONE BY:

ALBERTO CASTELLANO MACIAS with NIA: 100414242

ALEX OSWALD with NIA: 100479018

NARIM WOO with NIA: 100478751

TUTORIAL 3 - MACHINE LEARNING

EXERCISE 1 : KNOWLEDGE FLOW

11. Execute the KnowledgeFlow. To do so, you can either (1) press the start button placed in the top right part of the interface; or (2) click the Start loading option from the Arff Loader node. After that, select the option Show Results from the TextViewer nodes. What information is shown on each of them? What is the percentage of correctly classified instances?
We generated 2 different TextViewer nodes:
The first one was linked to node J48 and shows how each instance is classified by means of a decision tree that classifies each instance based on intervals created for each attribute. The size of the tree is 730, with 584 leaves.
The second one was linked to node ClassifierPerformanceEvaluator and shows a summary for the results. We obtained 28071 correctly classified instances (86.21%) from a total of 32561 instances. We also see that there were 2890 instances misclassified as class <=50k and 1600 instances misclassified as >=50k.

13. What is the utility of creating knowledge flows with this Weka interface?
Knowledge flows like this one we just created are very useful in order to develop machine learning algorithms more easily as the results are shown much more clear and they are easier to understand.

EXERCISE 2: EXPERIMENTER

2.1 Pac-Man data generation and pre-processing

1. Modify the printLineData() function you programmed in Tutorial 1, so it generates the .arff files1 Weka receives as input. It should contain the proper header, and each line of data should contain an instance of the game state (with the attributes you consider relevant) plus the direction Pac-Man has just taken.

2. Generate a training data file called “all data pacman.arff” with more than 500 instances. To do so, execute the game with the following parameters: python busters.py -g RandomGhost -l openHunt. Note that since we want you to collect the instances with the KeyBoardAgent (the one by default), you should write the printLineData() function for that particular agent.

3. Generate two new files from the one you just have created. On each of them you should select a subset of different attributes. Save these files as “filter data pacman manual1.arff” and “filter data pacman manual2.arff”. By doing this, you should have 3 different .arff files with more than 500 instances each.

TUTORIAL 3 - MACHINE LEARNING

2.2 Design and execution of the experiment Now we proceed to design and execute the experiment using the Experimenter interface:

1. Launch Weka.

2. Open the Experimenter.

3. Click on New to create a new experiment.

4. Select Classification as the type of experiment.

5. Select the three previous datasets.

6. Select the algorithms J48, IbK (with k=1,3,5), PART, ZeroR and NaiveBayes.

7. On Results Destination select ARFF and click on Browse to select the file. This file will contain the results and data from the experiment, and you will be able to open it as a spreadsheet when Experimenter finishes.

8. Save the experiment clicking on Save, choosing the name you want for the experiment.

9. Click the Run tan and press Start.

2.3 Results analysis

1. Click on the Analyse tab to analyze the results.

2. Click on Experimenter to select the results of the current experiment.

3. Select Percent correct on the Comparison field, and then select Perform test.

4. The v and * characters indicates if the result is statiscally better (v), worse (*) or equal ( ) than the base scheme, with the specified significance value (0.05 by default). The numbers between brackets (v/ /*) appearing below each scheme indicate the number of times a scheme is better, equal or worse than the base scheme.

5. Is there a data set that seems more appropriate?

TUTORIAL 3 - MACHINE LEARNING

The data set that seems more appropriate is the data set containing every attribute without any other additions as it returned the highest mean accuracy of the tested algorithms.

6. Which algorithm do you think is the best?
We think the best algorithm is IBk where k=1 due to the obtained accuracy results.

7. Are the results of the best algorithm much better than the others?
The results of the best algorithm (IBk, k=1) take a significant lead over the zeroR algorithm in terms of accuracy, and also maintain a margin over the PART, Naive Bayes, and other k-values for the IBk algorithm. However, the IBk k=1 algorithm had an accuracy most closely yielded by the J48 algorithm.

8. Save in a file both the configuration and the results of the analysis.

9. What do you think the Weka Experimenter is suitable for?
Based on this tutorial's experimentation, the Weka Experimenter is most suitable for comparing the efficacy of different machine learning algorithms over the same group of data sets. By way of this comparison the Weka Experimenter can help one decide which machine learning model to employ for this needs.

CONCLUSIONS

Through the trials of Tutorial 3 we were able to use the Weka KnowledgeFlow to analyze our data and graphically design experiments. By using the Weka Experimenter we were able to compare the efficacy of different machine learning algorithms for a group of sample data sets. Using these Weka implementations together with Python added a layer of versatility to both our python program and the graphical tool, Weka.

As a combined exercise, we were able to use Weka and Python together for both the design and testing stages of implementing a machine learning algorithm. Instead of using a python-only implementation of machine learning algorithms such as pytorch we were able to use a more guided approach that clearly and intuitively demonstrated the benefits and ease of A/B testing different models and machine learning algorithms in a real scenario.