**HW 7 Report**

Brian Barbu (brb9da)

**Introduction:**

The task of this assignment was to analyze different data sets using some popular machine learning algorithms to see how these algorithms can be used to identify significant patterns in data. In this assignment, final performance measurements for 2, 5, 10, 20, and 50 folds of each data set was recorded using the J48, JRip, and PART machine learning algorithms. From this data, this report will be able to make significant conclusions regarding the efficacy of the J48, JRip, and PART algorithms by comparing their accuracy in the different tests taken. This assignment is meant to introduce machine learning data evaluation and demonstrates it with the different types of tests used for each data set.

**Algorithms used:**

The three algorithms used in this assignment is J48, JRip, and PART. These are machine learning algorithms that use a sequence of rules to identify significant data. Each algorithm evaluates data differently and can create unique results that represent the data in different but significant ways. These algorithms are performed by a chosen amount of folds or split cross validations that effect how many iterations of the algorithm with the data set is performed. All of the resulting values from the algorithm’s iterations are averaged into the performance measurements. The performance measurements that these algorithms produce are useful for analyzing data since the estimated results are computed with a statistic of accuracy. These algorithms can also be expressed alternatively in decisions trees that outline the sequence of possible events.

**Program Structure:**

This program is made in the WekaMachineLearning.java file. First the program prompts the user to choose a data file to process by invoking the getInput() method and saving the user’s input as an int. This int choice correlates to a data file that the user chooses. Then the program asks the user the amount of k-fold cross validations he wants to do using the getK() method. Next, the program asks the user to choose an algorithm to use with the selectAlgo() method and this selection is giving in an int that correlates to a different jar file being initiated. An Evaluation class called validation is made which uses the classify() method to apply the algorithm to the data set. The rest of the output given by the algorithm is then displayed by printing the String obtained from algo.toString(). The accuracy is also printed from the calculatedAccuracy() method to describe the algorithm with k folds result with the data set. Lastly, the program asks the user if he would like to rerun the program and do another test. The user chooses 1 for yes or 2 for no. If yes, the choice variable is set to 1 which resets the program to the beginning stages for more user input. If not, the program exits. This is the general program structure of this java file.

**Data set 1 – Iris:**  Description-

Number of Instances: 150

The domain: Botanical

Data Attributes:

-petalwidth: measure of flower petal width; double

-petallength: measure of flower petal length; double

-sepallength: measure of sepal length; double

-sepalwidth: measure of sepal width; double

Class Attribute:

-class: gives flower classification as Iris-setosa, Iris-versicolor, or Iris-virginica

**Rules:**

**J48:**

petalwidth <= 0.6: Iris-setosa (50.0)

petalwidth > 0.6: Iris-versicolor (25.0)

classifies flower as setosa or versicolor on whether petal width is less than or greater than .6

**JRip:**

(petallength >= 5) => class=Iris-virginica (32.0/2.0)

(petalwidth >= 1.7) => class=Iris-virginica (6.0/1.0)

(petallength >= 3) => class=Iris-versicolor (47.0/0.0)

=> class=Iris-setosa (50.0/0.0)

Classifies Iris as virignica or versicolor based on whether petal length is first greater than or equal to 5, than width greater than or equal to 1.7, and then for petal length greater or equal to 3. Else it is a Iris-setosa

**PART:**

petalwidth <= 0.6: Iris-setosa (50.0)

: Iris-versicolor (25.0)

If petal width less greater than .6 its Iris-setosa otherwise its versicolor

**Data set 2 – Labor: Description-**

**Number of Instances**: 57

**The domain**: Civilian

**Data Attributes:**

-wage-increase-first-year: measure of wage increase from first year; double

-statutory-holidays: measure of amount of statutory holidays; int

-vacation: measure of vacation habits; String

-working-hours: measure of working hours; int

-wage-increase-second-year: measure of wage increase from second year; double

-pension: measure of whether person has pension or not; boolean

-education-allowance: whether person has an education allowance; boolean

-cost-of-living-adjustment: measure of whether person has a cost of living adjustment or not; boolean

-longterm-disability-assistance: measure of whether person has long term disability assistance or not; boolean

**Class Attribute:**

**-**class: gives classification measure of laborer as good or bad

**Rules:**

**J48:**

wage-increase-first-year <= 2.5: bad (5.18/1.18)

wage-increase-first-year > 2.5: good (23.82/1.0)

if wage increase in first year less or equal to 2.5 it gets a bad class rating, if greater than 2.5 it gets a good class rating

**JRip:**

(wage-increase-first-year <= 2.5) => class=bad (5.0/1.0)

=> class=good (24.0/1.0)

If wage increase in first year is less than or equal to 2.5 it gets good bad class rating, else gets good class rating

**PART:**

wage-increase-first-year > 2.5: good (23.82/1.0)

: bad (5.18/1.18)

If wage increase in first year greater than 2.5, get good class rating, else gives bad class rating.

**Data set 3 – Nursery: Description-**

**Number of Instances**: 12960

**The domain:**

**Data Attributes:**

-health:

-has\_nurs:

-parents:

-social:

-housing:

-finance:

-children:

**-**form:

**Class Attribute:**

-class: gives measure of priority level as either very\_recom, spec\_prior, not\_recom, and priority

**Rules:**

**J48:**

health = recommended

| has\_nurs = proper

| | social = nonprob

| | | housing = convenient

| | | | finance = convenient: very\_recom (32.0/1.0)

| | | | finance = inconv

Check for health, having a nurse, social condition, housing, and finance as some of the variables

**JRip:**

(children = 1) and (has\_nurs = proper) and (form = complete) and (housing = convenient) and (health = recommended) and (finance = convenient) and (parents = usual) => class=recommend (3.0/1.0)

Checks for whether has one child and has a proper nurse, a complete or, convenient housing, health, finance, and parets, then gives recommended class

**PART:**

health = not\_recom: not\_recom (2160.0)

has\_nurs = less\_proper AND

health = priority: priority (576.0)

**if not recommended health, gives not recommende class, if has proper nurse and priority health, gives priority class.**

**Results:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| J48 Algorithm | | Number of Folds (“k”) | | | | |
| Data Sets | Percent  Accuracy  % | 2 | 5 | 10 | 20 | 50 |
| Iris | 32.67 | 89.33 | 94.00 | 92.67 | 94.00 |
| Labor | 75.44 | 80.70 | 82.46 | 78.95 | 77.19 |
| Nursery | 70.65 | 71.87 |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| JRip Algorithm | | Number of Folds (“k”) | | | | |
| Data Sets | Percent  Accuracy  % | 2 | 5 | 10 | 20 | 50 |
| Iris | 30.67 | 94.00 | 93.33 | 92.67 | 94.00 |
| Labor | 71.93 | 87.72 | 85.96 | 87.72 | 87.72 |
| Nursery | 70.10 | 73.56 | 76.71 | 89.48 | 94.61 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| PART Algorithm | | Number of Folds (“k”) | | | | |
| Data Sets | Percent  Accuracy  % | 2 | 5 | 10 | 20 | 50 |
| Iris | 32.67 | 89.33 | 90.67 | 90.67 | 94.00 |
| Labor | 75.44 | 84.21 | 75.44 | 71.93 | 70.18 |
| Nursery | 72.69 | 68.81 | 77.01 | 92.47 | 96.98 |

**Conclusion:**

This assignment was important because it demonstrated data mining techniques and showed how different machine learning algorithms can derive significant data. The tables obtained in the results sections depict percent accuracy measurements found with the program that rate the efficacy of the algorithms in represented data and its possibilities. When algorithms are able to produce higher percent accuracy data, those algorithms offer a more significant representation of the data. The amount of folds used altered the amount of iterations of the program and therefore affected the amount of data values averaged into the results. Computers use the same random seed when performing programs which is why algorithms of the same data set and amount of folds will produce the same results. Adding more folds should increase the validity of data because more trials means a larger population of data and therefore more possibilities can be represented. This relationship can be see in the tables as the percent accuracy found with 2 folds was usually less than accuracy measurements at greater k values. Looking at the data, it can be seen that some algorithms performed unusually better with certain data sets than others. For example, the JRip algorithm produced percent accuracy measurements in the high 80s for folds 5 and greater while the other algorithms all performed in the low 70 range. An explanation for this performance would lie in the coding of the individual algorithms. The JRip algorithm must have more rules that apply and represent data in the labor data set when compared to other algorithms. The J48 algorithm is more simple when compared to the other algorithms as most of its rules are few factored and only a few lines. JRip and PART rules are clearly more elaborate and connected. J48 did not clearly outperform either of the other algorithms under any of the experiments. It can be expected that more complex algorithms that try to compare more types of data will produce better results. These complex algorithms involve more factors and therefore can represent the data more accurately.