CSC 458 Data Mining and Predictive Analytics I, Spring 2021

Dr. Dale E. Parson, Assignment 5, Using Ensemble Classifiers with the basic classifiers and data of Assignment 4. Due by 11:59 PM on Sunday May 9 via D2L Assignment 5. I cannot award points after 9 AM on Monday May 10. I need to turn in grades that day. Do not wait until the last minute to work on this. I am trying to avoid pressure for you on final exam week. It is strictly interpreting results from my 2.5-hour batch run of Weka. You do not need to run Weka again. You need to interpret its output.

From the top of README.txt:
CSC458 Spring 2021 Assignment 5. This is due end of Sunday May 9.
I will not accept solutions after 9 AM on Monday May 10 because I
need to get grading done and into MyKU. **I will answer questions only in class and during the final exam period because this assignment serves in place of the final exam. Questions refer to concepts learned through earlier assignments. Our final exam period is Thursday, May 6, 2 p.m. – 4 p.m.**

The goals of this assignment include comparisons of machine models used without & with Ensemble Learning and without and with external TEST data as in Assignment 4. The Assignment uses the RANDomly down-sampled 468,015 water sampling instances from Assignments 3 and 4, with 49,066 training instances and 418,949 disjoint testing instances, and the seven attributes of Figure 2 of Assignment 4 (repeated below).

Perform the following steps to set up for this semester's projects and to get my handout. Start out in your login directory on csit (a.k.a. acad).

```
cd $HOME
mkdir DataMine # This should already be there from previous assignments.
cd ./DataMine
cp ~parson/DataMine/csc458ensemble5sp2021.problem.zip csc458ensemble5sp2021.problem.zip
unzip csc458ensemble5sp2021.problem.zip
cd ./csc458ensemble5sp2021
```

TO GET THE FILES FROM ACAD TO YOUR Mac OR Linux machine use this command line:

```
scp YOURLOGIN@acad.kutztown.edu://home/kutztown.edu/parson/DataMine/csc458ensemble5sp2021.problem.zip
csc458ensemble5sp2021.problem.zip
```

You can use the reverse command line to copy files from your machine into your acad account.

WINDOWS USERS should log into https://download.kutztown.edu/ , log in, go to Computer Science, and download WinSCP. It appears you can just go to http://winscp.net/eng/download.php To get it.

EDIT THE SUPPLIED README.txt which contains questions Q1 through Q10. Keep with the supplied format, and do not turn in a Word or PDF or other file format. I will deduct 20% for other file formats, because with this many varying assignments being turned in, I need a way to grade these in

reasonable time, which for me is a batch edit run on the **vim** editor. Please turn in your final files README.txt by the deadline using the D2L Assignment 5.

Running Weka – You do not need to run Weka for this assignment. I will go over the batch file that I coded and ran for 2.5 hours per run to configure and execute the Weka models.

4/29/2021 Q4 in the README.txt had an extra line. It should look like this:

```
------------------------------------------------------------------------
Q4: What accounts for the model in your answer to Q3 having almost an
    identical kappa for 10FoldCrossValidation and ExternalTestFile?

------------------------------------------------------------------------
********************************************************************************
```

We will go over **csc458ensemble5sp2021.py** that runs the model in class.

Here are the files in the handout directory.

USGS_PA_STREAM_2012_CLASSIFICATION_RAND_TRAIN_7attrs.arff
USGS_PA_STREAM_2012_CLASSIFICATION_RAND_TEST_7attrs.arff

The training and testing datasets **from Assignment 4 with the attributes of Assignment 4's Figure 2**.

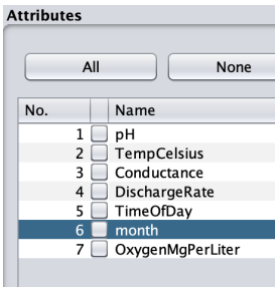

Figure 2: Coarse-grain temporal values TimeOfDay (nominal) and month (discrete integers)

I needed to have identical attributes in the TEST data in order to automate testing.

README.txt is the only file you must edit. Turn in via D2L Assignment 5.

csc458ensemble5sp2021.py is my batch Python script that generates the following Weka output. We will go over it in class. You do not need to change or run it.

csc458ensemble5sp2021.summary.ref.csv is a comma-separated value file that you need to consult to answer questions in README.txt. We will go over in class, and there are some useful Unix **sort** command line docs below for those of you who do not use Excel. Using Excel is easier.

csc458ensemble5sp2021.summary.ref is a more readable version of the results in the CSV file. It contains the accuracy / error measures from Weka and the confusion matrices. You may want to consult it when answering README.txt questions.

csc458ensemble5sp2021.all.ref is all classifier output from Weka. It is BIG. You may want to consult it when answering README.txt questions.

The **makefile** and **makelib** assisted me in running the batch **csc458ensemble5sp2021.py**.

README.txt recommends sorting values within **csc458ensemble5sp2021.summary.ref.csv** on various attributes. You could do all your searching manually, but that is time-consuming, tedious, and error prone. I will give a short overview of using Excel for sorting in class, but if you are not an Excel user, you can use Unix command lines as follows.

Here are the first 5 lines of **csc458ensemble5sp2021.summary.ref.csv** which indicate the order of model runs in my Python script:

$ **head -5 csc458ensemble5sp2021.summary.ref.csv**
testkey,testdatatype,kappa,MAE,RMSE,Instances,runtime,cputime
ZeroR,10FoldCrossValidation,0.0,0.1522,0.2759,49066,1.7,0.001378
OneR,10FoldCrossValidation,0.3975,0.089,0.2983,49066,2.14,0.001218
NaiveBayes,10FoldCrossValidation,0.4138,0.1192,0.2438,49066,3.99,0.00103
BayesNetmaxNrParents,10FoldCrossValidation,0.674,0.0611,0.1873,49066,6.34,0.000897

$ **grep 10Iterations csc458ensemble5sp2021.summary.ref.csv | head -5**
Bagging10IterationsNaiveBayes,10FoldCrossValidation,0.4152,0.1192,0.2436,49066,19.2,0.00122
Adaboost10IterationsNaiveBayes,10FoldCrossValidation,0.4138,0.1767,0.2946,49066,73.56,0.001496
Bagging10IterationsBayesNetmaxNrParents,10FoldCrossValidation,0.712,0.061,0.1749,49066,44.29,0.0
01417
Adaboost10IterationsBayesNetmaxNrParents,10FoldCrossValidation,0.7197,0.0572,0.1788,49066,98.61,
0.001369
Bagging10IterationsJ48,10FoldCrossValidation,0.7478,0.0611,0.1674,49066,354.04,0.0014

The fields follow.

testkey gives the name of the classifier being run. More complicated testkey values like Bagging10IterationsJ48 indicate Bagging for 10 Iterations with J48 as the base classifier. BayesNetmaxNrParents is BayesNet with the plateaued value for config parameter maxNrParents in Assignment 4 Q7.

testdatatype is either 10FoldCrossValidation (testing using 10-fold cross validation against the training set) or ExternalTestFile (train with USGS_PA_STREAM_2012_CLASSIFICATION_RAND_TRAIN_7attrs.arff and test with USGS_PA_STREAM_2012_CLASSIFICATION_RAND_TEST_7attrs.arff as in Assignment 4).

kappa is the Kappa value for that test.

MAE is the mean absolute error for that test.

RMSE is the root mean squared error for that test.
Instances gives the number of instances in the test data.
runtime gives the real time in seconds that it took to run this test. Sum of runtimes is about 2.5 hours.
cputime gives the processor time that it took to run this test.

Here is an example entry from **csc458ensemble5sp2021.summary.ref**.

```
START TEST ZeroR 10-fold cross validation
**********************************
 END TEST ZeroR 10-fold cross validation
Correctly Classified Instances       15739               32.0772 %
Incorrectly Classified Instances     33327               67.9228 %
Kappa statistic                          0
Mean absolute error                      0.1522
Root mean squared error                  0.2759
Relative absolute error                100      %
Root relative squared error            100      %
Total Number of Instances            49066

=== Confusion Matrix ===

     a     b     c     d     e     f     g     h     i     j   <-- classified as
     0     0     0     0     5     0     0     0     0     0 |     a = '(-inf-3.29]'
     0     0     0     0   732     0     0     0     0     0 |     b = '(3.29-5.08]'
     0     0     0     0  4574     0     0     0     0     0 |     c = '(5.08-6.87]'
     0     0     0     0 13781     0     0     0     0     0 |     d = '(6.87-8.66]'
     0     0     0     0 15739     0     0     0     0     0 |     e = '(8.66-10.45]'
     0     0     0     0 10119     0     0     0     0     0 |     f = '(10.45-12.24]'
     0     0     0     0  3727     0     0     0     0     0 |     g = '(12.24-14.03]'
     0     0     0     0   334     0     0     0     0     0 |     h = '(14.03-15.82]'
     0     0     0     0    50     0     0     0     0     0 |     i = '(15.82-17.61]'
     0     0     0     0     5     0     0     0     0     0 |     j = '(17.61-inf)'
```

Finally, here are some guidelines on running the Unix sort utility and other shell commands with the data in **csc458ensemble5sp2021.summary.ref.csv**.

$ **head -1 csc458ensemble5sp2021.summary.ref.csv**
testkey,testdatatype,kappa,MAE,RMSE,Instances,runtime,cputime
$ **head -1 csc458ensemble5sp2021.summary.ref.csv | sed -e 's/,/ /g'**
testkey testdatatype kappa MAE RMSE Instances runtime cputime

Piping output to this **sed** (stream editor) command replaces each comma with three consecutive spaces. Use plain single quote characters around the sed command string.

Run sort using **comma** as the delimiter (-t, for “tab”), -k key column 4 (MAE), -n for numeric (otherwise string-based sort), -r when you wany reverse (descending order), -s to preserve original order for ties.
$ **sort -t, -k 4 -n -r -s csc458ensemble5sp2021.summary.ref.csv**
Adaboost10IterationsNaiveBayes,10FoldCrossValidation,0.4138,0.1767,0.2946,49066,73.56,0.001496
Adaboost100IterationsNaiveBayes,10FoldCrossValidation,0.4138,0.1767,0.2946,49066,76.31,0.001567
Adaboost10IterationsNaiveBayes,ExternalTestFile,0.412,0.1767,0.2947,418949,40.02,0.001482

Adaboost100IterationsNaiveBayes,ExternalTestFile,0.412,0.1767,0.2947,418949,39.69,0.001778
ZeroR,ExternalTestFile,0.0,0.1523,0.276,418949,4.35,0.00162
ZeroR,10FoldCrossValidation,0.0,0.1522,0.2759,49066,1.7,0.001378
Bagging10IterationsNaiveBayes,ExternalTestFile,0.4091,0.1197,0.2444,418949,61.03,0.001074
ETC.

Sort on **testdatatype** as the primary key (field 2).

$ **sort -t, -k 2 -s csc458ensemble5sp2021.summary.ref.csv**
ZeroR,10FoldCrossValidation,0.0,0.1522,0.2759,49066,1.7,0.001378
OneR,10FoldCrossValidation,0.3975,0.089,0.2983,49066,2.14,0.001218
NaiveBayes,10FoldCrossValidation,0.4138,0.1192,0.2438,49066,3.99,0.00103
Adaboost10IterationsNaiveBayes,10FoldCrossValidation,0.4138,0.1767,0.2946,49066,73.56,0.001496
Adaboost100IterationsNaiveBayes,10FoldCrossValidation,0.4138,0.1767,0.2946,49066,76.31,0.001567
Bagging100IterationsNaiveBayes,10FoldCrossValidation,0.4143,0.1192,0.2435,49066,145.55,0.001407
Bagging10IterationsNaiveBayes,10FoldCrossValidation,0.4152,0.1192,0.2436,49066,19.2,0.00122
ETC.

FOR A NUMERIC SORT USE -n in the Unix sort command.

Despite documentation to the contrary, it appears that there is no command line option that makes Linux **sort** utility use a **stable sort**, which is a sort that maintains the original order on a tie. With a stable sort we could achieve primary and secondary keys by sorting on the secondary key and piping that via "|" to a sort on the primary key. In Excel you sort first on the secondary key and then on the primary key. Excel's sort is stable.

Another option is to use python or preferably ipython in interactive mode:

$ **ipython**

```
In [1]: import csv
In [2]: file = open('csc458ensemble5sp2021.summary.ref.csv', 'r')
In [3]: csvin = csv.reader(file)
In [4]: datalist = list(csvin)
In [8]: header = datalist[0]
In [9]: instances = datalist[1:]
In [10]: print(header)
['testkey', 'testdatatype', 'kappa', 'MAE', 'RMSE', 'Instances',
'runtime', 'cputime']

In [15]: print([inst for inst in instances[0:3]])
[['ZeroR', '10FoldCrossValidation', '0.0', '0.1522', '0.2759', '49066', '1.7', '0.001378'],
['OneR', '10FoldCrossValidation', '0.3975', '0.089', '0.2983', '49066', '2.14', '0.001218'],
['NaiveBayes', '10FoldCrossValidation', '0.4138', '0.1192', '0.2438', '49066', '3.99',
'0.00103']]
```

FOR A NUMERIC SORT USE float(…) in the Python list sort function.

```
In [21]: instances.sort(key=lambda inst : float(inst[6]))
```

Above sorts on real execution time.

```
In [22]: print(instances)
[['ZeroR',   '10FoldCrossValidation',   '0.0',   '0.1522',   '0.2759',
'49066', '1.7', '0.001378'],
['OneR',   '10FoldCrossValidation',   '0.3975',   '0.089',   '0.2983',
'49066', '2.14', '0.001218'],
['OneR',  'ExternalTestFile',  '0.3974',  '0.089',  '0.2984',  '418949',
'3.93', '0.001501'], ['NaiveBayes', '10FoldCrossValidation', '0.4138',
'0.1192', '0.2438', '49066', '3.99', '0.00103'],
ETC.
```

For a primary & secondary sort key, do it like this:

```
# inst[2] is STRING testdatatype, inst[6] is float runtime
In [23]: instances.sort(key=lambda inst : (inst[1], float(inst[6]))) #
inst[1] primary, inst[6] secondary key

In [24]: print(instances)
[['ZeroR',   '10FoldCrossValidation',   '0.0',   '0.1522',   '0.2759',
'49066', '1.7', '0.001378'],
['OneR',   '10FoldCrossValidation',   '0.3975',   '0.089',   '0.2983',
'49066', '2.14', '0.001218'],
['NaiveBayes', '10FoldCrossValidation', '0.4138', '0.1192', '0.2438',
'49066',       '3.99',       '0.00103'],       ['BayesNetmaxNrParents',
'10FoldCrossValidation', '0.674', '0.0611', '0.1873', '49066', '6.34',
'0.000897'],
ETC.
['Bagging100IterationsNaiveBayes',     'ExternalTestFile',     '0.4122',
'0.1195', '0.244', '418949', '516.42', '0.001309'],
['Bagging100IterationsJ48',  'ExternalTestFile',  '0.7661',  '0.0599',
'0.162', '418949', '662.97', '0.001768'],
['Bagging100IterationsBayesNetmaxNrParents',        'ExternalTestFile',
'0.7246', '0.0599', '0.1711', '418949', '737.49', '0.001659']]
```