7-24-2014

Using Weka to Mine Temporal Work Patterns of Programming Students

Dale E. Parson
Kutztown University, parson@kutztown.edu

Recommended Citation

Tutorial notes presented at The 2014 International Conference on Frontiers in Education: Computer Science and Computer Engineering (FECS’14), Las Vegas, July 21-24.

Using Weka to Mine Temporal Work Patterns of Programming Students

Dale E. Parson, Kutztown University of PA, http://faculty.kutztown.edu/parson

Follow-up to “Mining Student Time Management Patterns in Programming Projects”

Dale E. Parson and Allison Seidel, FECS’14 (#FEC2189)
http://faculty.kutztown.edu/parson/FECS2014ParsonTutorial.zip
http://www.cs.waikato.ac.nz/ml/weka/
http://www.cs.waikato.ac.nz/ml/weka/book.html

1. Examine a typical programming project directory, makefile, and logdata.sh script.

   See FECS2014ParsonTutorial/FillWord4/ makefile and logdata.sh.

2. Run **CLASSPATH=.. make build test** and examine the zipfile & datamine/.

   Note that the **mv** command moves a zip file to the instructor’s inbox every time a student runs **make** [**build** | **test** | **turnitin**]. Much of the later data extraction works with multiple zipfiles from multiple make actions, e.g., constructing a picture of a “work session” from multiple make invocations. (A “session” consists of one or more make invocations with no gaps >= 60 minutes.)

   Alternative run ./**buildunix.sh && ./rununix.sh** or **buildwindows.bat**, **runwindows.bat** to build without a makefile, used for build & test on a student machine. This step creates and appends data to data_do_not_lose_this_file.txt.

3. Run **python worktimeToARFF.py 1 prjdata.csv fakegrades.csv surveys.csv emaildata.csv fakearff.arff ./fakemine FillWord4/**

   Use Python 2.7.X. Here are the contents of the demo CSV files:

   prjdata.csv

   ```
   ##cour,seme,prjn,start,end
   csc243,sp2014,1,2014-02-10 00:01,2014-02-28 23:59
   csc243,sp2014,2,2014-02-27 00:01,2014-03-16 23:59
   csc243,sp2014,3,2014-03-13 00:01,2014-04-05 23:59
   csc243,sp2014,4,2014-04-10 00:01,2014-04-19 23:59
   csc243,sp2014,5,2014-04-20 00:01,2014-05-03 23:59
   ```

   Course, semester, project number, start datetime, end datetime

   fakegrades.csv

   ```
   ##suem,suid,Gprj1,Gprj2,Gprj3,ignore,ignore,Gprj4,Gprj5,ignore,ignore,Gcrs,Gle
   t,yea,trk,Cumg,Crdg,Cumm,Crdm
   ```

```
parson,c243s14id1,1.02,1,0.97,0,0.97,1.02,1.05,0.9,0.85,0.9572,A,Sophomore,UGRD Liberal Arts & Science - BS CSC/INFO TECHNOLOGY,3.82,45,3.67,21
```

See schema_STUDENT_PRJ_WORK.txt. Fake data out of grading spreadsheet.

surveys.csv

```
##suem,prjn,Xasn,Xdue,Xams
parson,1,1,2,3
```

Project number, count of competing CS assignments handed out, due, and any competing exam.

emaildata.csv

```
##suem,prjn,clue,count
parson,4,0,1
parson,4,1,2
```

Email to instructor. The clue field is 0 for clueless emails, 1 for emails with good student understanding.

fakearff.arff is the output ARFF file.

./fakemine contains the mined ZIP files.

FillWord4/is the initial handout directory.

Notes from **worktimeToARFF.py**:

Mac/OSX datetime strings are incompatible. Linux & Solaris are OK.

```
__seconds_between_sessions__ = 3600 # Set to interval separating sessions.
__mode_session_time_minutes_quantum__ = 15
__mode_session_bytes_quantum__ = 1000
__mode_session_lines_quantum__ = 20
__diff_quantum__ = 20

# Next pattern depends on the course's source language.
__src_re__ = re.compile(r'^.*\.java$')
# Next pattern cracks apart fields in 'ls -l' while maintaining compatibility
# with both Solaris & Linux, as far as I can tell. Assumes strip() off of ends.
# Group 1 is bytes, 2 is month, 3 is day, 4 is time, 5 is filename.
__ls_re__ = re.compile(r'^\S+\s+\d+\s+\S+\s+\S+\s+(\d+)\s+([A-Z][a-z]+)\s+(\d+)\s+(\d+:\d+)\s+(\S+)$')
# MAY 14, 2014 change __ls_re__ to account for a platform where a student
# got an ls with a year instead of hours:minute. Assume we can get either.
```

```
__ls_re__ = re.compile(r'^\S+\s+\d+\s+\S+\s+\S+\s+(\d+)\s+([A-Z][a-z]+)\s+(\d+)\s+(\d+(:\d+)?)\s+(\S+)$')
```

4. Run **python addrank.py fakearff.arff** to get the centile rankings.

5. Run **./wekacmd.sh fakearff.arff.tmp.arff** to inspect the fake data.

6. Open demoStudentDB.arff to inspect some data prepared for the tutorial.

   This dataset is a fake dataset prepped for the demo.

7. Discuss **Preprocessing** (StringToNominal and date removal), **Select attributes**, redundant attributes, analyses for a **numeric target attribute** (Simple K-means, M5Rule and M5P tree), discretizing and analyses for **nominal target attributes** (OneR, J48, naiveBayes).

schema_STUDENT_PRJ_WORK.txt

```
 1      studentid                           suid
 2      student-year                        syea    (Sophomore, Junior, Senior)
 3      student-track                       strk    (SD or IT or OT for other)
 4      course                              cour
 5      semester                            seme
 6      project-number                      prjn
 7      project-start-datetime              prjs
 8      project-end-datetime                prje
 9      assigned-until-started-hours        Hstr    (round to nearest hour)
10      completed-until-due-hours           Hend    (round to nearest hour)
11      started-until-due-hours             Jstr    (round to nearest hour)
12      Jstr - hours lost to skipped days   Jfst    (Jstr - 24 * each skip)
13      assigned-until-completed-hours      Jend    (round to nearest hour)
14      started-until-completed-hours       Jall    (round to nearest hour)
15      min-session-time-minutes            Mmin    (session gap of >= 60 mins)
16      max-session-time-minutes            Mmax
17      mean-session-time-minutes           Mavg
18      stddev-session-time-minutes         Mdev
19      median-session-time-minutes         Mmed
20      mode-session-time-minutes           Mmod    (round to nearest 15)
21      mean-time-between-sessions-hours    Havg
22      stddev-time-between-sessions        Hdev
23      min-session-files                   Fmin
24      max-session-files                   Fmax
25      mean-session-files                  Favg
26      stddev-session-files                Fdev
27      median-session-files                Fmed
28      mode-session-files                  Fmod
29      min-session-bytes                   Ymin
30      max-session-bytes                   Ymax
31      mean-session-bytes                  Yavg
32      stddev-session-bytes                Ydev
33      median-session-bytes                Ymed
34      mode-session-bytes                  Ymod    (round to nearest 1000)
35      min-session-lines                   Lmin    (may need to use ?)
36      max-session-lines                   Lmax    (may need to use ?)
37      mean-session-lines                  Lavg    (may need to use ?)
38      stddev-session-lines                Ldev    (may need to use ?)
39      median-session-lines                Lmed    (may need to use ?)
40      mode-session-lines                  Lmod    (round 20, need to use ?)
41      min-session-added                   Amin    (may need to use ?)
42      max-session-added                   Amax    (may need to use ?)
43      mean-session-added                  Aavg    (may need to use ?)
44      stddev-session-added                Adev    (may need to use ?)
45      median-session-added                Amed    (may need to use ?)
46      mode-session-added                  Amod    (round 20, need to use ?)
47      min-session-deleted                 Dmin    (may need to use ?)
48      max-session-deleted                 Dmax    (may need to use ?)
49      mean-session-deleted                Davg    (may need to use ?)
```

50	stddev-session-deleted	Ddev	(may need to use ?)
51	median-session-deleted	Dmed	(may need to use ?)
52	mode-session-deleted	Dmod	(round 20, need to use ?)
53	min-session-changed	Cmin	(may need to use ?)
54	max-session-changed	Cmax	(may need to use ?)
55	mean-session-changed	Cavg	(may need to use ?)
56	stddev-session-changed	Cdev	(may need to use ?)
57	median-session-changed	Cmed	(may need to use ?)
58	mode-session-changed	Cmod	(round 120, to use ?)
59	number-sessions	Snum	
60	total-session-time-minutes	Mtot	
61	number-sessions-centered-hour0-3	S0003	
62	number-sessions-centered-hour4-7	S0407	
63	number-sessions-centered-hour8-11	S0811	
64	number-sessions-centered-hour12-15	S1215	
65	number-sessions-centered-hour16-19	S1619	
66	number-sessions-centered-hour20-23	S2023	
67	mean-compete-csc-projects-assign	Xasn	
68	mean-compete-csc-projects-due	Xdue	
69	mean-compete-exams	Xams	
70	number-builds-started	Bsta	
71	number-builds-completed	Bend	
72	number-tests-unix-started	Tstx	
73	number-tests-unix-completed	Tenx	
74	number-tests-pc-started	Tstp	(Tests on student's machine.)
75	number-tests-pc-completed	Tenp	(Tests on student's machine.)
76	total-tests-started	Tstb	(Both Unix & PC test starts.)
77	total-tests-completed	Tenb	(Both Unix & PC test end.)
78	post-turnitin-make-actions	Ptis	
79	clued-emails	Eyes	
80	clueless-emails	Enot	
81	total-emails	Etot	
82	grade point average at start	Cumg	
83	number credits at start semester	Crdg	
84	grade point average in csc >= 125	Cumm	
85	number credits in csc >= 125	Crdm	
86	course-numeric-grade	Gcrs	
87	course-letter-grade	Glet	
88	project-numeric-grade	Gprj	
89	project-letter-bin	Gplt	(3 bands per grade)
90	course-percentile-grade	GcrsRank	
91	project-percentile-grade	GprjRank	

NOTES:

1. Any attribute containing ? as a value in this dataset can and probably should be discarded on initial analysis. Find the grey cells in Weka's EDIT window. That includes mode attributes, because there is not always an unambiguous mode. It includes line data (lines changed/added/deleted), and surveys (because of survey data collection errors), and probably others.

2. Of the string data, studentid should be removed, and the others should be nominalized using filter StringToNominal.

3. Attributes Gcrs, Glet and GcrsRank are redundant with each other, giving different views of the same data. You can keep at most one at a time, or the algorithms will infer one from the others. Gprj, Gplt and GprjRank are the same for the project. GcrsRank and GprjRank are numeric centile ranks for the course and project respectively. They may be the very useful since they expand clumped grade concentrations, and can be Discretized into (10?) bins for J48, NaiveBayes and other classifiers requiring nominal targets.

4. Looking back through the spring csc243 dataset with Weka in September, I am surprised to see OneR outperforming J48 in various basic investigations. Apparently, J48 is being confused by ambiguous data. I don't remember that from my quick look this summer.

5. One approach is to use OneR to the find the most use predictive attribute, remove that attribute, then see what the second-most predictive attribute is, then remove that. This approach will give you a set of perhaps up to 10 of the most predictive attributes. Then you can throw out all the others, keep those 10, and use more powerful algorithms such as J48, NaiveBayes or M5P / M5Rules on those attributes to see how they fare. The number 10 is just a guess. Too few means throwing away too much data; too many become hard to interpret.

6. My final suggestion for now is to see what you can use to predict Gplt, and Gprj, GprjRank, and a Discretized GprjRank, one at a time. Gplt and a Discretized GprjRank are nominal and therefore amenable to OneR, J48, NaiveBayes and RandomTree. Gprj and GprjRank are numeric and therefore amenable to M5P, M5Rules, and SimpleKMeans clustering (among others). Creating enough clusters to show at least 4 different grade levels in the target attribute actually looks like it might be useful.

7. May 16, 2014 added Jfst which is Jstr - 24 hours * number of days skipped
work between the start and the final turnitin.