CSC 458 Data Mining and Predictive Analytics I, Fall 2019

Dr. Dale E. Parson, Assignment 1, Using Python scripting constructs to read and parse structured textual data (a comma-separated value or CSV file) and to write an ARFF (attribute-relation file format) table of data for later analysis. Due by 11:59 PM on Friday September 27 via make turnitin.

You should work on **acad** to run **make test** and **make turnitin**. Running **make test** uses a 2.x version of Python. Copying the handout directory to another Linux or Unix machine (including Mac/OSX) should work for **make test**, but you need to be on acad for **make turnitin**.

The goals of this assignment are to practice using Python programming constructs and data types to crack apart a textual CSV data file and create an ARFF file amenable to analysis with the Weka data mining tool. This is the only programming (scripting) assignment this semester. Note that writing data cleaning scripts can account for as much as 50% of a data analyst's workload.

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

```
cd  $HOME
mkdir  DataMine
cp  ~parson/DataMine/csc458fall2019assn1.problem.zip DataMine/csc458fall2019assn1.problem.zip
cd   ./DataMine
unzip  csc458fall2019assn1.problem.zip
cd  ./csc458fall2019assn1
make test
```

Running **make test** fails initially because you must complete the definition of file weatherToARFF2019.py that I have started. That file has your detailed instructions for this assignment in the form of Python comments. Script weatherToARFF2019.py when completed will analyze handout file KPAHAMBU4.parson.csv and file KPAHAMBU4.STUDENTID.csv (where STUDENTID is your KU student login ID) downloaded from the Weather Underground, and will create output file weatherToARFF2019.arff. Please look for STUDENT comments in file weatherToARFF2019.py and follow those instructions. Script file weatherToARFF2019.py was originally a working solution from which I removed code that you must now complete, starting with your name near the top. **Make sure to indent Python using only spaces (no tabs). My handout code uses 4 spaces per indentation level. Use that**.

Run **make turnitin** on acad by the due date. The late penalty is 10% per day, and I will not accept solutions after I go over an assignment. Plan to attend all classes, either in person or via Zoom, and ask questions. Running **make turnitin** does not send you email. I will send project grades via KU email, typically before the next class after each due date. A successful run looks roughly like the following. It prints an error message and aborts when it does not work. Your output .arff file must have identical formatting to mine, including spacing.

```
$ make test
/usr/bin/python ./weatherToARFF2019.py weatherToARFF2019.arff KPAHAMBU4.parson.csv
diff weatherToARFF2019.arff weatherToARFF2019.arff.ref > weatherToARFF2019.arff.dif
/usr/bin/python ./weatherToARFF2019.py weatherToARFF2019.arff KPAHAMBU4.parson.csv KPAHAMBU4.parson.csv
# Convert station, student, and winddir strings to nominals
# The following runs Weka in command-line mode.
java -cp /home/KUTZTOWN/parson/weka/weka-3-8-1/weka.jar "weka.filters.unsupervised.attribute.StringToNominal" -R 2,3,7 -i weatherToARFF2019.arff -o weatherToARFF2019nominals.arff
```

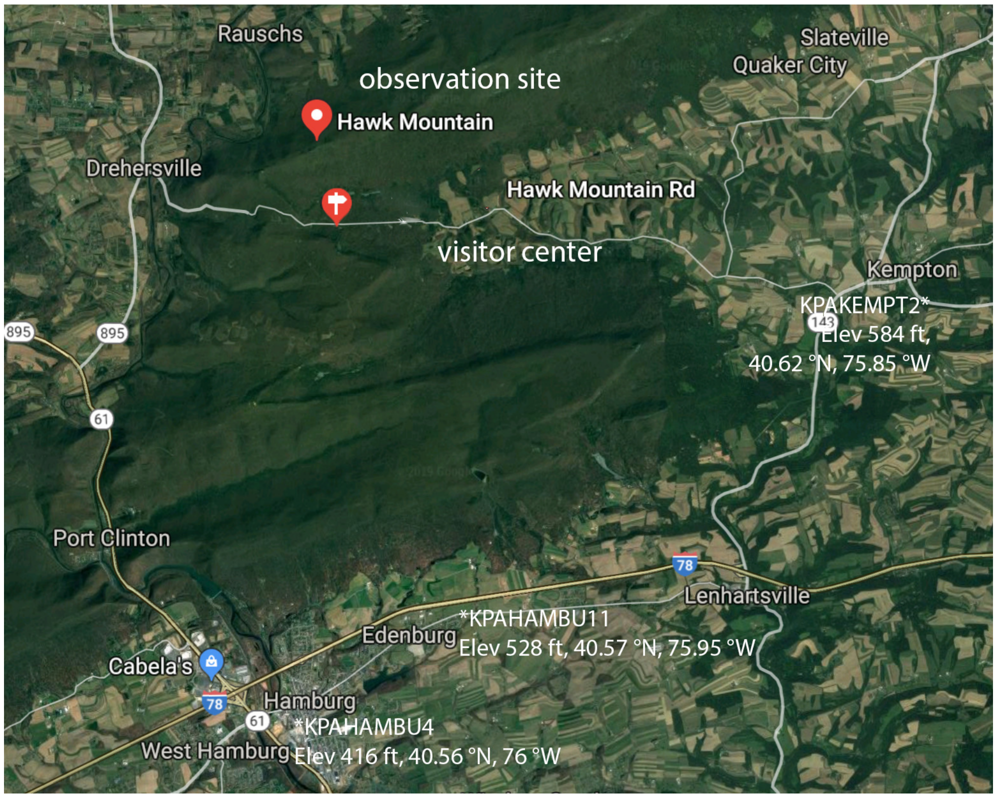
When running **make test**, if your program creates format differences from mine, then data1fall2019chlorophyllassn1.dif in your project directory details the differences between your data1fall2019chlorophyllassn1.out (left-arrowed lines in the diff file) and my data1fall2019chlorophyllassn1.arff.ref (right-arrowed). Your output file format must match mine exactly.

```
$ make turnitin
/bin/rm -f *.o *.class .jar core *.exe *.obj *.pyc
/bin/rm -f *.out *.o *.arff *.dif *.out ./weatherToARFF2019.arff
/bin/rm -f ./weatherToARFF2019nominals.arff

Do you really want to send csc458fall2018assn1 to Professor Parson?
Hit Enter to continue, control-C to abort.



/bin/bash -c "cd .. ; /bin/chmod 700 .                ; \
                    /bin/tar cvf ./csc458fall2018assn1_parson.tar csc458fall2019assn1      ; \
                    /bin/gzip ./csc458fall2018assn1_parson.tar                 ; \
                    /bin/chmod 666 ./csc458fall2018assn1_parson.tar.gz           ; \
                    /bin/mv ./csc458fall2018assn1_parson.tar.gz ~parson/incoming"
csc458fall2019assn1/
csc458fall2019assn1/makelib
csc458fall2019assn1/makefile
csc458fall2019assn1/studentsWeather.py
csc458fall2019assn1/weatherToARFF2019.py
csc458fall2019assn1/KPAHAMBU4.parson.csv
csc458fall2019assn1/KPAHAMBU4.STUDENTID.csv
csc458fall2019assn1/weatherToARFF2019.arff.ref
```

In addition to **make test**, you can run **make testparson** to test your weatherToARFF2019.py using my supplied KPAHAMBU4.parson.csv, or **make teststudent** to test your weatherToARFF2019.py using my supplied KPAHAMBU4.parson.csv and your manually captured KPAHAMBU4.STUDENTID.csv. Running **make test** runs both of these tests.

NATURE OF THIS PROJECT

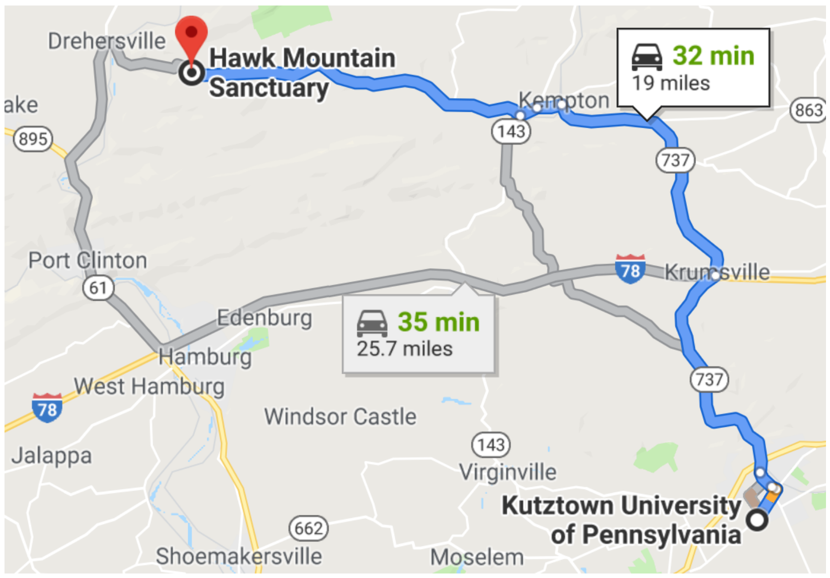
This is the first year we are using raptor count data from the Hawk Mountain Sanctuary[1] to look for migration patterns. They have supplied us raptor count observation data for 2017[2] and 2018[3]. In assignment 1 we are going to augment that data with weather data recordings from a weather station in Hamburg, PA. Our steps for capturing, cleaning, formatting, and storing this weather data appear below. I will merge the Hawk Mountain data with our weather data for analyses in assignment 2 and subsequent assignments. Credit goes to Dr. Michael Davis in Geography for helping me find the Hamburg site data, and to Dr. Laurie Goodrich of Hawk Mountain Sanctuary for supplying their data. Dr. Goodrich has offered us an on-site presentation and tour of their facilities this semester.



Hawk Mountain Sanctuary and nearby weather recording stations via Weather Underground

Above is a map showing the location of the Hawk Mountain observation site and the KPAHAMBU4 weather station supplying our weather data for assignment 1[4]. The '*' asterisks show the approximate locations of the GPS coordinates for the weather stations. We are not using the Kempton weather station because it does not record precipitation, nor the Edenburg station because it does not have hourly data for 2017 and 2018.

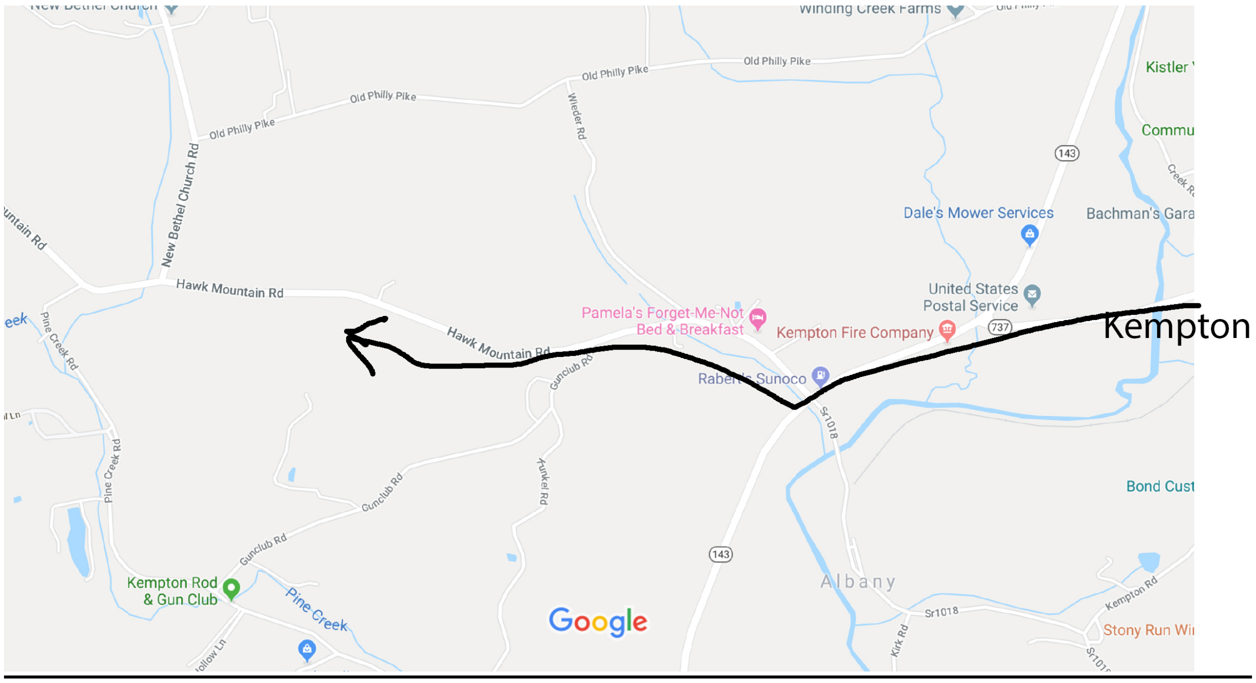
[1] https://www.hawkmountain.org/
[2] https://faculty.kutztown.edu/parson/fall2019/HMS_F2017.txt
[3] https://faculty.kutztown.edu/parson/fall2019/hawkcount_data2018hourly.txt
[4] There is a detailed public hiking trail map that I have been capturing over the last 5 years, and hiking for 50 years, at https://acad.kutztown.edu/~parson/HikeyUSGS.jpg



Driving map from KU to Hawk Mtn. for our planned field trip. PA143N is closed N of Virginville.

Above is a driving map for our invited tour and presentation of Hawk Mountain. PA143N has been closed N of Virginville, and that grayed NW road is winding and hilly, so we should take PA737N when the time comes. Directions follow. **We will meet at the Hawk Mountain Visitor's Center on left side of Hawk Mountain Road at the top of the mountain on Saturday September 21 at 9 AM. Make sure to be on time. Plan for an hour's presentation + two hours optional hike to North Lookout. With drive it's ~4.5 hours.**

Hawk Mountain Sanctuary, Visitor Center
40.635, -75.99 (Use PA-737N out of Town, road closure on PA-143 N.)
40.635N, 75.99W
Go to Turkey Hill on Main Street and turn left onto PA-737 N, continue until it ends after Kempton.
Updated 9/1: PA-737N turns left in Kempton. Turn left and stay on PA-737N until it ends at PA-143. Turn left onto PA-143S for 0.3 miles, turn RIGHT at Sunoco gas station onto Hawk Mtn Rd to Eckville.
Continue onto Hawk Mountain Road.
Follow Hawk Mountain Rd up the ridge to the **Visitor Center** (not the Educational Building that you'll see first) on the left.
(18.9 mi) (**Give yourself at least 45 minutes' drive time from KU. Meet at Hawk Mtn Visitor Center.**)



Send me email at parson@kutztown.edu by September 8 whether you commit to going, and, if so, whether you could drive in a car pool. Round trip driving time will be over an hour. Leave an hour for presentation at their Visitor Center, and at least 2 hours for an optional hike out to the North Lookout Observation Site. That's 4.5 hours, and we could run over. You can skip the somewhat rigorous hike. Also, let me know if you are willing to drive in a car pool, and the number of students you can take, including yourself. This field trip is not required for your grade.

Part I. Crowd Sourcing for Interactive Capture of the Data (20% of the assignment grade)

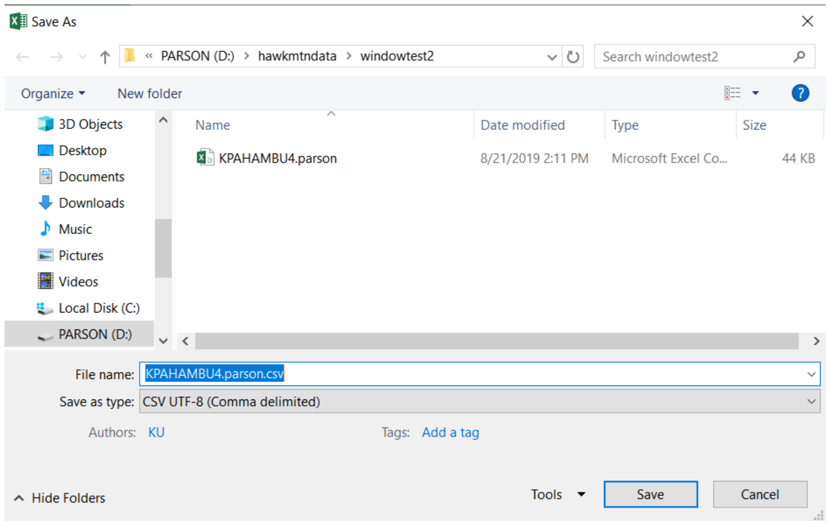
We need to capture, clean, format, and store weather data to augment the somewhat limited weather data supplied by Hawk Mountain. They supply mostly wind direction and speed. On the morning of our first class I will update http://faculty.kutztown.edu/parson/fall2019/csc458fall2019urls.txt to give you each a distinct set of 9 or 10 URLs at Weather Underground for manual data collection. If you can write a web scraper to collect this data in the CSV format required by the assignment, fine. It was going to cost me more time to attack writing a scraper than it is worth, so I am crowd sourcing interactive data collection from 278 web pages across 27 students + myself. I am waiting until the first day of class to hand out URLs because students may add or drop the course between now and then. This Part I takes around 30 minutes or less to complete.

I have saved a 25-minute Zoom video[5] from my data capture session on a Mac that gives detailed instructions for Part I. I will demo doing this on Windows in the first class. Here is an outline of those detailed steps:

1. Go to your next URL from http://faculty.kutztown.edu/parson/fall2019/csc458fall2019urls.txt in a browser.

[5] https://kutztown.zoom.us/recording/share/muojwSb90KC9b2O46W8Bz2kDy5SC-z63y2_fawH_pkqwIumekTziMw

2. Sweep with the mouse & copy the table at the bottom, starting with the date, and ending with the last row and last column of data. Avoid copying the ads at the bottom.
3. Paste the table into an initially empty Excel Workbook (spreadsheet). Paste each subsequent table on your individual assignment list below the previous table. Keep columns aligned, and make sure the dates are included and consecutive. The student who has a URL range jumping from December 2017 to August 2018 will have one non-consecutive table entry.
4. After each paste, Save your Excel file. The first time this step requires Save As. Make the file name KPAHAMBU4.STUDENTID.xlsx (let Excel pick the file extension). Thereafter a simple Save will do.
5. After the final Save, do a Save As using the CSV UTF-8 format. Make sure to use the file name KPAHAMBU4.STUDENTID.csv with your login ID for STUDENTID. In one test Excel did not change the xlsx extension to csv. It must be csv, and the format must be CSV UTF-8. You can ignore the warning about losing some Excel features such as graphs. Here is what this Save As looks like on Windows.


Save As

« PARSON (D:) > hawkmtndata > windowtest2

Search windowtest2

Organize ▾ New folder

3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Local Disk (C:)
PARSON (D:)

Name	Date modified	Type	Size
KPAHAMBU4.parson	8/21/2019 2:11 PM	Microsoft Excel Co...	44 KB

File name: KPAHAMBU4.parson.csv

Save as type: CSV UTF-8 (Comma delimited)

Authors: KU Tags: Add a tag

6. Use Filezilla or a similar utility to copy both your KPAHAMBU4.STUDENTID.xlsx and your KPAHAMBU4.STUDENTID.csv files into the **csc458fall2019assn1** directory on acad that you created from instructions on the first page of this handout.

Because having accurate data is essential to later projects, I will spot check CSV files for all students to ensure accurate measures for a sample of the dates. The first inaccurate table capture will cost half of Part I's 20%, and any additional inaccurate table capture will cost the remainder of the 20%. If the dates come across in the copy & paste steps, and if the final row of data (usually after 11 PM) comes across, and if the columns for all pasted tables are aligned, you will be fine. Make sure to follow the saving instructions.

Part II. Completing CSV-to-ARFF translator weatherToARFF2019.py (80% of the assignment grade)

I have coded this translator and then taken out the parts that you must complete. There are detailed instructions in the code comments in weatherToARFF2019.py. Look for the **STUDENT** tags for your part.

Here is what the makefile and my code do:

Open output file weatherToARFF2019.arff and one or more input files KPAHAMBU4.STUDENTID.csv specified on the command line in that order.

Scan, format, and store in variables these lines from KPAHAMBU4.STUDENTID.csv as strings:

```
13-Aug-17,,,,,,,,,,
,,,,,,,,,,
Time,Temperature,Dew Point,Humidity,Wind,Speed,Gust,Pressure,Precip. Rate.,Precip. Accum.,UV,Solar
,,,,,,,,,,
```

Except for the date lines, my scanning ignores the other lines and all-whitespace lines. It uses only the input KPAHAMBU4.STUDENTID.csv file name and the date lines in that file to set the following variables. We will go over this code in class.

YEAR, MONTH, DAY, STATION, STUDENT

You will need to use my values. Do not change them. There are three additional variables that you need to use.

ISERROR – My comments in the code say, "Check to make sure units of measure are AM or PM, F or C (for celsius), %, mph, in (for inches). If not, report and record an error in ISERROR." There are examples of error reporting in my code. This ensures that you have captured your KPAHAMBU4.STUDENTID.csv file correctly.

FILENAME and LINENO are for reading the next input file. You can use them when reporting an error per my examples in the code. Don't change them. You can make up any other temporary variables that you need.

Each line your code reads and verifies looks like this (see handout code).

```
12:04 AM,67.8 F,67.1 F,98 %,West,0.0 mph,0.0 mph,29.87 in,0.00 in,0.00 in,,0 w/m²
```

The above line was preceded by the "13-Aug-17,,,,,,,,,," line. The corresponding line in my weatherToARFF2019.arff.ref output file looks like this:

```
"2017-08-13 00:04",KPAHAMBU4,parson,19.89,19.50,0.98,None,0.0,0.0,29.87,0.00,0.00
```

My code splits apart the input fields, using the commas as separators, and then your code below STUDENT comments does its work. Things to note about what your code must do.

1. It uses string.split() and string.strip() to separate numeric data like "12:04 AM" from unit of measure.
2. It converts time to a 24-hour format.

3. It converts Fahrenheit to Celsius for F values, does not change values with C as the unit of measure. It converts either to strings with 2 fractional digits of precision. The formula is in the comments.
4. It convert humidity from % to a fraction.
5. When the wind speed is 0, it sets direction to None.
6. Otherwise, any blank field in the input becomes a ? character in the ARFF file.
7. It uses the wind speed, gust, precipitation, and barometric pressure using the strings in the CSV file.
8. It ignores the last 2 fields from the CSV file.
9. It merges STATION, STUDENT, hour, and minute data into each line's datetime, surrounded by double quotes.

Here is a line with a wind speed but no direction from the CSV file, followed by its resulting ARFF line.

```
2:17 AM,65.4 F,65.1 F,99 %,,1.0 mph,2.0 mph,29.87 in,0.00 in,0.00 in,,0 w/m²
"2017-08-13 02:17",KPAHAMBU4,parson,18.56,18.39,0.99,?,1.0,2.0,29.87,0.00,0.00
```

My code also writes the header for the arff file:

```
% D. Parson, csc458 fall 2019 assignment 1
@relation weather
@attribute datetime date "yyyy-MM-dd HH:mm"
@attribute station string
@attribute student string
% tempc and dewpc are in Celsius
@attribute tempc numeric
@attribute dewpc numeric
% humidity is a fraction in the range [0.0, 1.0]
@attribute humidity numeric
% winddir is a string Like West or None when there is no wind.
@attribute winddir string
% windspd and windgust are in mph
@attribute windspd numeric
@attribute windgust numeric
% barometric pressure is inch of mercury
% see https://en.wikipedia.org/wiki/Inch_of_mercury
% https://en.wikipedia.org/wiki/Atmospheric_pressure
@attribute barompres numeric
% precipitation/hour and accumulated precipitation/day are in inches
@attribute preciprt numeric
@attribute precipaccum numeric
@data
```

Detailed coding instructions are in the STUDENT comments for weatherToARFF2019.py. We will go over the details, along with **make test** actions, in class. Please attend. Grading rubrics for Part II (80%):

Correctly processing the parts of the fields list already split by my code (20%), correct conversion and formatting of F values to C (10%), correct verification of units of measure (10%) ("Check to make sure units of measure are AM or PM, F or C (for celsius), %, mph, in (for inches). If not, report and record an error in ISERROR."), correct formatting of output strings (10%), correct output (write) code (10%), no diffs (10%), no ARFF file format errors (10%). Each bug costs 10%, and each day it is turned in late is a 10% penalty. I will not accept assignments after I go over them in class; those earn 0%.

That sums to 100%.

Run **make test** after any code change that might break a working program, and run **make turnitin** as described above by the due date to avoid losing points. If you make subsequent changes after submitting, just run **make test** and then **make turnitin** again, thereby over-writing the previous submission.