4-6-2024

Preparing and Teaching Data Science Courses

Dale E. Parson

PREPARING AND TEACHING DATA SCIENCE COURSES

Dale E. Parson
Kutztown University of Pennsylvania
parson@kutztown.edu

ABSTRACT

Preparing and teaching realistic data science courses requires labor-intensive preparation and course delivery. It is not enough to download data and push buttons on machine learning tools. First, there must be a human expert available in the problem domain to supply data and evaluate work. Without a human expert to provide information that is missing or incorrect in archived data, the tendency is to take the output of machine learning algorithms using potentially faulty data on faith. Second, any real-world data requires custom scripts for correcting invalid values, creating derived attributes, and formatting data for analysis. Then comes the analysis, which is usually iterative because of incremental discoveries, often requiring additional data, expertise, data preparation, and analysis. This case study outlines four domains of data analysis that have been very useful in teaching and student-oriented research: 1) analyzing Java programming student performance as a function of work habits; 2) analyzing physical and chemical relationships in Pennsylvania stream flow data; 3) analyzing audio files for waveform type and noise levels; and 4) analyzing raptor migration counts in Pennsylvania as a function of climate change.

KEY WORDS

analytics, classification, data cleaning, data science, machine learning, regression

1. Introduction

This report is a case study on the steps taken to perform research, prepare materials and projects, teach, and evaluate student work in upper-level undergraduate and masters-level graduate data science courses at Kutztown University. Research in this context refers to incorporating novel data domains and datasets into course materials that in some cases leads to publishable results. This approach does not use toy or well-established example datasets. The approach uses at least partially new data for each course offering, giving the instructor and students the opportunity to uncover data relationships not previous established. The author relates data science to archaeology as excavation and analysis of deep patterns in data. If object-oriented architecture is plumbing, data science is the iterative application of digging and examining.

The following sections explore topics in a chronological manner. First, it is essential to identify and enlist human domain experts in guiding and reviewing results of automated or semi-automated analysis. Applying algorithms to data can help create useful analysis, but algorithms can also mislead by missing relationships or inferring incorrect relationships in data, for example inferring causation where there is only correlation. Applying algorithms can provide new insights to domain experts, but algorithms can also suggest false conclusions that human expertise can identify.

Second, it is necessary to locate and obtain data in the expert's domain. The expert or third parties may provide data repositories.

Third, it is usually necessary to clean, organize, and store the data. Datasets may contain omissions, errors, and redundant entries. Detecting errors and imputing values for missing fields [1] may be straightforward using basic statistical techniques or may require complicated, domain-specific scripting.

Analysis starts with identification of so-called *non-target attributes* – the control variables of these algorithmic experiments – and *target attributes* – the experimental variables. The latter are attribute values we are attempting to infer. They are tagged onto the dataset in supervised learning or are missing, to be inferred, in unsupervised learning. Using tagged datasets is the norm in teaching data science because the main constructs of interest are the algorithmically extracted mathematical models that relate non-target attributes to target attributes. Students and course projects need complete, accurate, stable datasets on which to build relationship models. Unsupervised learning, i.e., predicting the future from past examples and past learning, first requires learning in a stable environment.

Analysis is the final stage in this report. Analysis is an iterative process. Attribute elimination, augmentation, and derivation may be suggested by examining intermediate results. Data visualization may suggest next steps in this iterative process.

The sections that follow take up each of these stages in turn in terms of the author's teaching experience. Rather than

Proceedings of the 39th Annual Pennsylvania Association of Computer and Information Science Educators Conference (PACISE 2024), Kutztown University of PA, April 5 and 6, 2024.

presenting a sequence of projects, this report presents a sequence of stages, with the same 4 projects per stage.

2. Enlisting a Domain Expert and Their Data

2.1 Correlating student programming habits to grades

Our first case study relates programming student behavior in a sophomore-to-senior Java programming course to project letter grades (classification) and project numeric grades (regression) [2-4]. These grades served as target attributes to be predicted from other attributes. The author served as the primary domain expert in this and in two of the remaining three case studies, with faculty peers and graduate students contributing expertise where needed in this one. It was important to build competence in using tools, libraries, and in performing analysis before venturing outside the author's domains of expertise.

Most of the data were collected automatically using makefiles in the Unix programming environment when students performed compile, test, and project submit steps. A few survey questions about other courses, projects, and exams that competed for time and attention accompanied each project. The proposed project went through the university's Institutional Review Board (IRB) approval process, and students signed IRB-approved forms granting or denying participation. An optional makefile target could turn off data collection. All 31 students in the two spring 2013 sections and 39 students in the two spring 2014 sections of Java Programming participated.

There were 90 attributes in the data collected in 2013, including student time of work per session, number of sessions, code lines added / deleted / changed, success or failure of compile & test operations, timestamps on files, and others detailed in [2]. A set of Python scripts integrated project grade data and other data discussed in Section 3.

In spring 2013 the author also taught a graduate course intended for study of the internals of database management systems, but with permission of the department chair, borrowed over half of the semester to explore analysis of data from PA stream flow sensors outlined in Section 2.2 using the Weka toolkit [5,6]. This graduate course offering was the genesis of the data science courses at Kutztown University. The graduate students served as auxiliary domain experts in suggesting Java student behavior data to collect. No data outside project edit, compile, test, and submit actions, survey answers, project grades, and incoming computer science GPAs (grade point averages) were collected. Student IDs were obfuscated, and the author did not include data such as gender and race that might lead to reverse identification of students.

2.2 Correlating PA stream flow properties

The author served as the domain expert for this study, based on two years of experience with his high school son volunteering as monthly water samplers and data collectors for the Maiden Creek Watershed Association several years earlier [7]. We acquired additional expertise in stream heights, flow rates, and turbidity by kayaking flooded PA streams from 2004 through 2007. Expertise in this and other environmental studies was augmented by suggestions from the author's nephew, who holds a Ph.D. in environmental engineering and is a lead engineer for a contract company working with the Federal Emergency Management Agency (FEMA) and other agencies on flood plain and related data analysis.

Course data came from on-line databases supplied by the US Geological Survey (USGS) [8]. The author limited the scope of the project to PA streams and rivers. The following are the primary attributes used from 46 automated sampling sites with over 490,000 stream sample records. Sites were selected based mostly on available measured attributes.

timestamp	measured
pH	measured
TempCelsius	measured
Conductance	measured
GageHt	measured
DischargeRate	measured
TimeOfYear	derived attribute
Month	derived attribute
MinuteOfDay	derived attribute
MinuteOfYear	derived attribute
OxygenMgPerLiter	measured

USGS stream data are available as text reports requiring scripted conversion to comma separated value (CSV) files. After some preliminary analysis, OxygenMgPerLiter served as the tagged, target attribute to be predicted from other attributes. We continue to extend and use PA stream data in more advanced analysis.

2.3 Correlating audio signal properties

The author again served as the domain expert, having worked in industry for over a decade as a software architect and developer for tools used to debug embedded audio signal processing applications in telephony [9].

The data used from spring 2020 through fall 2023 for two projects per semester were generated using statistical sampling techniques for standard audio reference properties via the ChucK audio programming dataflow language [10]. The tagged target classification attribute is waveform type from the standard reference set {sine, triangle, square, sawtooth, pulse}, and the tagged numeric regression attribute is white noise level, where white noise is uniformly distributed random signal levels across the audible frequency spectrum. We have used 10,005 instances so far; 5 are noiseless training instances for the standard {sine, triangle, square, sawtooth, pulse} set. One derived dataset has as non-target attributes the first 32 strongest signal frequencies and amplitudes for classification of waveform class, and another for white noise regression has statistical distributions of mean, median, min, and max for signal levels across the full audible spectrum. The raw data are .wav audio files. Feature extraction is discussed in the next section. Figure 1 shows a single cycle of reference audio waveform types generated in the author's fall 2023 undergraduate Python scripting course. The author has tentative plans to replace ChucK signal generation with Python libraries. The signal range of [-32768, 32767] in Figure 1 corresponds to the standard 16-bit representation in .wav files.

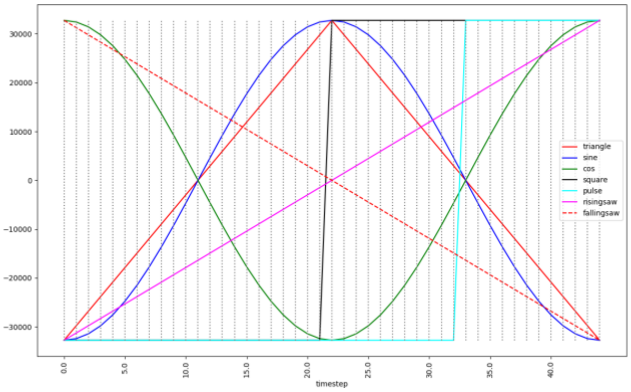


Figure 1: Visualizing reference audio waveforms

2.4 Correlating climate properties to raptor counts

The domain expert for this fourth case study is Dr. Laurie Goodrich, lead researcher at the Hawk Mountain Sanctuary (HM) in eastern Pennsylvania [11]. The author has been a member of the sanctuary for many years. One of the premiere activities of the sanctuary has been the annual autumn counting of 23 raptor species during fall migration. There are 88 years of data from 1934 through 2021, with integration of data from 2022 and 2023 in the author's work queue.

HM volunteers began recording climate data such as temperature, wind speed, wind direction, visibility, and cloud cover consistently starting in 1976. 1976 through 2021 gives us 46 years X approximately 100 days during the migration season X 12 median observations per day, yielding 55,200 discrete observation records (exact current count is 55,826), usually of a 60-minute duration. Volunteers perform the observations and Dr. Goodrich deploys the data in Excel spreadsheet format that the author converts to CSV for Python data cleaning.

The primary target attributes are the individual species' raptor counts and their long-term trends. This project is especially oriented towards time-series trend analysis of correlations between climate changes and raptor counts. The author and students began working with Dr. Goodrich in fall 2019, and incremental data deployment and analysis is ongoing. The author received a grant from the Kutztown University Research Committee for a summer 2022 stipend and funding for several student workers. The author has used these data extensively in data science courses.

Figure 2 shows Hawk Mountain's primary observation and data collection point at North Lookout. The ridge leading away is the Kittatinny Mountain, going northeast into northern New Jersey. Behind the camera the ridge falls to the valley, with the remainder of the Kittatinny Mountain continuing about a mile to the south on a southwest ridge to central PA, disconnected from North Lookout by the interposed valley of the Eckville Fault. Updrafts on the northern side of the ridge, to the left in the photo, provide lift and energy for locomotion for migrating raptors.

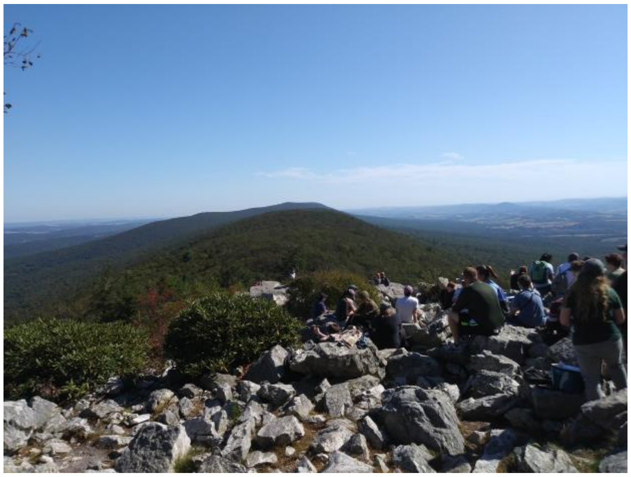


Figure 2: North Lookout at Hawk Mountain Sanctuary

The author prefers having students get physical contact with their data sources when possible. Figure 2 shows students at North Lookout after Dr. Goodrich gave us an expert presentation in September 2019. Unfortunately, COVID protocols made an in-person visit impossible in 2020-2021. The Kutztown Women in STEM student club and the author have a similar visit planned for April 27, 2024. One graduate student has completed a master's thesis and another is performing supervised research in this domain.

2.5 Summary: data domain expertise

An instructor had best start with problem domains for which they have some level of expertise. They are learning algorithms, tools, techniques, and pedagogical approaches, activities that constitute a serious workload, without acquiring new domain knowledge or engaging external experts at the same time. Only after several course offerings is it time to find a domain with an external expert.

Figure 3 illustrates one concrete example of why working with an expert in analyzing a domain that is new to the faculty member is essential. The lines in Figure 3 illustrate total annual counts for dominant wind directions in the range N, NNE, NE … WNW, NW, NNW, and UNK, with the latter representing no dominant wind direction during observation periods. Notice the green NW line that plummets going from 1994 to 1995. The author took that plummet at face value during initial analysis, looking to correlate it with declining raptor counts. Subsequent discussion with Dr. Goodrich revealed that, "1995 is the year that observers started using three-letter wind direction designations such as WNW." In fact, close examination of the red WNW line buried in Figure 3 revealed that this count rose at the same time that NW plummeted. Subsequent analysis was necessary to find how to integrate new WNW and NNW counts back into NW to establish data consistency. The result is that WNW alone integrates back into NW in order to synchronize with preceding years. This fact is significant because WNW and NW winds create updrafts that raptors use to coast along the northern side of the Kittatinny Ridge.

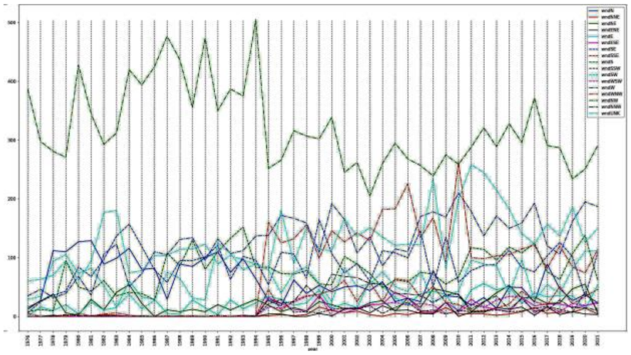


Figure 3: Apparent precipitous drop in NW wind in 1995

3. Cleaning, Organizing, Correcting Data

3.1 Java programming student data preparation

The author wrote 3525 non-blank lines of Python code in 13 scripts and libraries to integrate data from automated makefile collection of student build / test / submit actions, surveys, grade information, and other course roster data into a CSV file for analysis of the spring 2013 dataset. These numbers jumped to 4247 non-blank Python lines in 14 scripts and libraries because of an extension to perform new within-students analysis discussed in Section 4.1.

A potential problem presented by any data preparation stage with a lot of custom scripts is that of injecting errors into the dataset thanks to buggy code. For this project the author used grant funds to employ a student who had earned a grade of A in the 2013 course [2,3]. She had access only to her own raw data generated by the makefiles, the surveys, and the auxiliary data inputs. She used Excel on her own data to duplicate data preparation steps outlined by the author. When we compared her CSV file created within Excel to the author's created by Python scripts, there were some substantial differences. It turned out that the student was using the output of her first edit / build / test session as the starting data point in the project sequence, while the author was using the unchanged state of the handout code. When she updated her Excel calculations to use the handout code as the starting point, our CSV files lined up exactly to within trivial rounding differences. In current projects we compare numeric results using the Python math library's *isclose*() function that implements almost-equals testing within default or explicit tolerances [12]. Ignoring minor rounding differences is essential in derived data verification.

Having independent means for verifying data cleaned, aggregated, and structured by scripts that may contain bugs is essential for establishing veracity of integrated data and its analysis.

3.2 PA stream flow data acquisition and preparation

In 2017 the author wrote 504 non-blank lines of Python code in 2 scripts to parse textual data downloaded manually from the USGS web site [8] to create CSV files with the attributes enumerated in Section 2.2. This Python code relies on Python's regular expression library *re* in coding a parser for the text data [13]. Manual comparison of samples of the 8-attribute automated observations of Section 2.2 to CSV records was adequate for verification, augmented with a calculator to check the four derived attributes.

Some student assignments include familiarization with the *re* library in parsing and structuring textual data. In 2017 a graduate student introduced the class to the interactive *pythex* utility for testing regular expressions [14].

If analyzing water data were our primary project, it would be possible to write a web scraper to download data only for select sites containing the desired measurement attributes and other attributes such as location. Manual use of the USGS site requires specifying certain data constraints such as required attributes and location. While the instructor continues to use one USGS water project in some courses, the number of data collection sites needed does not justify spending time writing a web scraper.

3.3 Generating and extracting audio signal properties

A Python script generates a set of executable ChucK scripts [10] – ChucK is an application specific dataflow language for generating sound and music – one per .wav file, with signal gains uniformly distributed between 0.5 and 0.75, and white noise gains between 0.1 and 0.25, where 0.0 is no signal and 1.0 is the normalized maximum signal strength. The generator creates equal numbers of {sine, triangle, square, sawtooth, pulse} waveforms, one per generated ChucK script. A shell script then executes these ChucK scripts, one at a time, to create .wav files of one second duration.

Listing 1 shows the ChucK dataflow code for generating a triangle waveform with a fundamental frequency of 999 Hertz (cycles per second), a signal gain of approximately 0.675, and a white noise gain of 0.201into a .wav file of 1 second duration. The unique serial number of this .wav file is 183046. Tagged parameter values are embedded in the .wav file names. In Listing 1 the “dac” is the digital-to-analog converter that drives a loudspeaker. The dataflow sends the dac’s output to the WvOut object __w__ that stores the .wav file. These projects have generated 10,005 such .wav files, 2001 per waveform class. Figure 4 gives a schematic dataflow view of the code in Listing 1. Arrowed lines show generated signal flow direction.

```
TriOsc generator => Gain mixer => dac ;
Noise noisey => Gain noisegain => mixer ;
999 => generator.freq ;
0.675189571038 => generator.gain ;
0.201025367482 => noisegain.gain ;
dac => Gain __g__ => WvOut __w__ => blackhole;
"lazy1_TriOsc_999_0.675189571038_0.2010253674
82_183046.wav" => __w__.wavFilename ;
1::second => now ;
```

Listing 1: A ChucK dataflow script that generates a .wav file

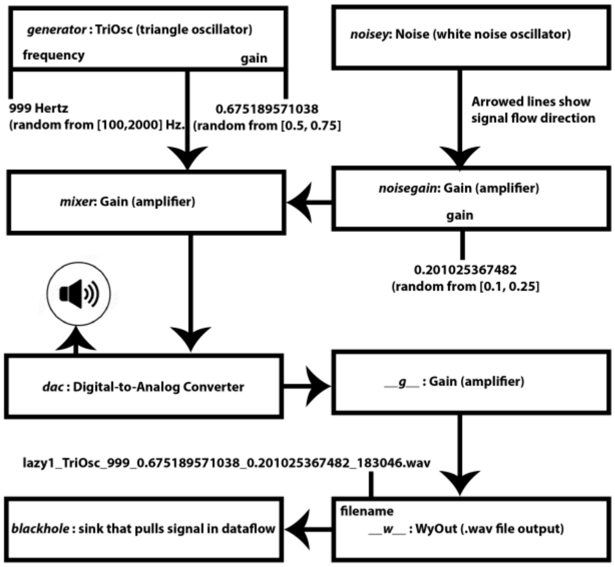

generator : TriOsc (triangle oscillator)
frequency
gain
noisey: Noise (white noise oscillator)
999 Hertz
(random from [100,2000] Hz.)
0.675189571038
(random from [0.5, 0.75]
Arrowed lines show
signal flow direction
mixer: Gain (amplifier)
noisegain: Gain (amplifier)
gain
0.201025367482
(random from [0.1, 0.25]
dac : Digital-to-Analog Converter
__g__ : Gain (amplifier)
lazy1_TriOsc_999_0.675189571038_0.201025367482_183046.wav
filename
blackhole : sink that pulls signal in dataflow
__w__ : WvOut (.wav file output)

Figure 4: The signal dataflow of Listing 1 ChucK code

These audio projects used a cryptic Chuck signal analysis script for reading a .wav file and extracting frequency and signal strength measurements from it as non-target attributes for machine learning tools outlined in Section 4.3. This script saves these non-target attribute values in a text file whose name contains the tagged, target attribute values including waveform class, signal fundamental frequency, signal gain level, and white noise level.

The point here is not to learn this audio problem domain or to decode ChucK dataflow scripts for signal generation and analysis. The point is that domain expertise and one or more domain experts are required to guide student projects in data analysis. This is not just a matter of pushing machine learning tool buttons.

In a fall 2023 course the author had some success, both in terms of simplifying coding by sticking to Python, and in terms of flexibility of analysis, by using the SciPy libraries [15] for reading .wav files and extracting non-target attribute values from them. Tagged target attributes are extracted from the .wav file names as discussed. Moving signal generation and extraction to Python opens up these stages for student coding in a language they know.

3.4 Climate and raptor data cleaning and structuring

Hawk Mountain climate and raptor observation data are collected by volunteers on an isolated lookout that is often cold and windy. Even though Hawk Mountain deploys their records as Excel files, the presence of several classes of errors required the most substantial Python data cleaning of these four case studies [16,17].

The first problem encountered is that the data from 1934 through 1975 are missing most climate attributes such as air temperature, wind speed, and wind direction. A master’s thesis student and the author integrated weather data from the Allentown, PA Airport downloaded from a National Oceanic and Atmospheric Administration (NOAA) website [18] into our dataset. Analysis uncovered the fact that the Lehigh Valley in which the airport resides suffers from the *Heat Island Effect* [19], creating climate conditions that do not correlate well with the weather on North Lookout, which is more subject to winds from the northwest and which cools thoroughly at night. The author has discarded this airport data in ongoing research, and has limited trend analysis to 1976 through the present, when weather conditions are in the deployed data.

The next problem is that some missing HM data, especially temperature records that should have been recorded as missing, were recorded in a substantial number of cases as a series of 0 Celsius numbers. It was necessary to write Python code that attempts to distinguish legitimate 0 records from missing data. The Python script inspects adjacent observation periods for fluctuation of temperatures that may make 0 crossings but that also have some non-0 values for nearby observations on the same day. Python converts a string of all zeroes for a day into missing, unknown value entries. In a related data problem, early years did not record cloud cover percentages, but they were sometimes entered as zeroes. These also needed to be marked as unknown.

A related problem was invalid temperature records that would have boiled or evaporated volunteer observers. For example, what appears to have been intended as a record of 24 degrees C (75.2F) was entered as 2424 C (4395.2 F). The volunteer simply wrote “24” twice. The Python script checked for temperature values outside of 2.5 standard deviations for the day, marking them as unknown.

Some raptor counts for an “all” category, for example the sum of immature and adult individuals of a species, did not match the actual sum of the sub-category counts. Our analyses use only the “all” counts. When they do not match the sum of the sub-counts, we use the larger of the recorded “all” entries and the sum of the sub-categories. The sub-categories vary by species.

Hawk Mountain began using 3-letter wind direction entries such as WNW for west-northwest in 1995. Volunteers occasionally rearranged these letters, for example recording NWW for WNW. Python cleaning was a matter of permuting the letters and picking the correct, canonical arrangement. As previously noted, prior to 1995 a wind direction of WNW was just recorded as NW. It became necessary to combine NW and WNW counts into an aggregate value from 1995 onward, to align later recordings with the lower resolution recordings before 1995. Note that WNW could have been just as readily recorded as W, since WNW lies between W and NW on the compass. Determining that WNW and NW should be combined starting in 1995, instead of WNW and W, required linear trend analysis in these and other 3-letter wind directions introduced in 1995.

3.5 Summary: cleaning, organizing, & correcting data

A general observation is in order. The more people involved in data generation, e.g., Java programming students in Section 3.1 or Hawk Mountain volunteers in the current Section 3.4, the more necessity for inspecting and cleaning data. The stream sensors of Section 3.2 are generally reliable, although they may fail, data transmission packets may fail to deliver, and extremely cold weather may affect sensor accuracy, but the sensors and automated data collection tend to be reliable. Stochastically generated audio of Section 3.3 in these studies is the most reliable. Collection of real-world audio recordings and data streams may suffer from some transmission, storage, and lossy signal problems, but even they should be pretty reliable.

Manual entry of large datasets, and datasets that span large intervals in time during which data collection procedures change, are the ones most likely to need preliminary human analysis to determine the need for cleaning at an early stage just after collection. Hawk Mountain manual data collection, and addition of attributes such as temperature, later cloud cover, and change from observing treetops to using ground-level wind gages to measure wind speed, are examples of potential sources of data errors and changes in processes that need to be massaged.

Human coding of more complex data integration or cleaning scripts such as those for Java programming students in Section 3.1 require validation that does not use the scripts to validate themselves. Both data entry and data manipulation via scripts require careful inspection for patterns of errors such as those discussed in Section 3.

4. Iterative Analysis

This section outlines approaches and results of analysis of the four case studies in very general, summary terms. The reader is directed to published papers enumerated in the Reference section for process details and results of these data analyses.

4.1 Analyzing Java programming student activities

The analysis of the 31 Java programming students in the two spring 2013 sections took the form of regression modeling in Weka [6] of non-target attributes such as starting time on projects, length of work sessions, time of day of work sessions, and magnitude of code changes per session, to numeric project grades [2]. Linear models unearthed the most important behavioral attributes, two of which appear here.

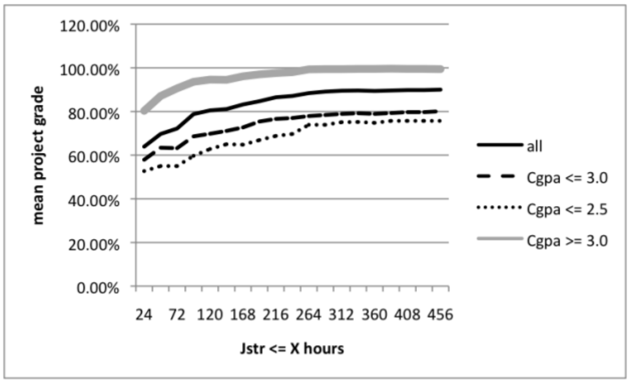


Figure 5: Project grade as a function of starting time

Figure 5 shows mean project grade as a function of mean student starting time before the due date (Jstr) on a span of 456 hours (2.7 weeks) to 24 hours in bands according to incoming computer science grade point average (Cgpa). In roughly the last 12 days of a project schedule, project grades slope downward; starting a project within 24 hours of its due date costs 20 points, 2 full letter grades, compared to starting it at least 12 days before its due date.

Figure 6 shows mean project grade as a function of mean work session length in minutes. A work session consists of consecutive makefile build and / or test actions with no gaps greater than 15 minutes between such actions. Grades for all Cgpa bins less than or equal to 3.0 drop off for average work sessions that are less than 60 minutes.

After adding 39 students in the two spring 2014 sections of Java Programming to the 31 students of the previous year, we performed additional analytical steps [3]. Data analysis is usually an iterative process in which one cycle of investigation uncovers the need for additional derived data attributes and subsequent analysis. An added analysis considers project data only for students with a project grade Gprj spread of at least 20% between their best and worst project grades. Different students occupy different overall grading bands, but each shares the fact that the difference between their best and worst Gprj is at least 20%.

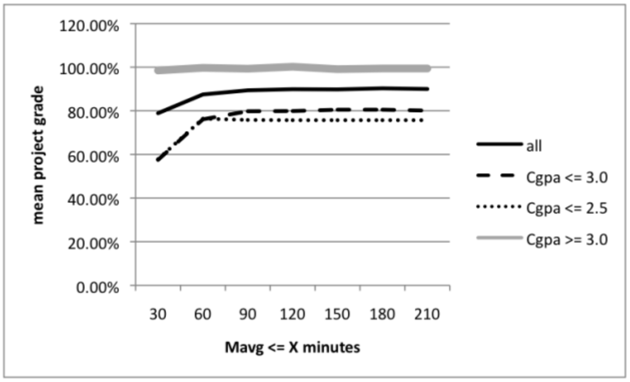


Figure 6: Project grade as a function of work session length

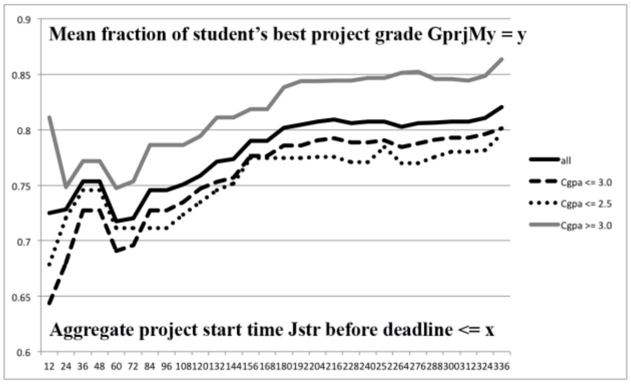


Figure 7: Within-student grade as a function of starting time

Figure 7 shows the mean within-student performance on project grade as a function of starting time before the deadline. Project grades scaled as the mean fraction of each student's best grade plummeted by 10% to 15% by procrastinating from the 2-week handout of an assignment to 24 or even 12 hours before it was due. We also analyzed skipping some days after an early start and found that on average only half of such days required actual work to be done. Presumably, an early start gives a student a sample of how difficult the project will be, and therefore how much leeway there is in working every day.

The uptick for Cgpa >= 3.0 at 24 hours led us to investigate the concept of *active procrastination*. [20,21]. Active procrastinators tend to have many projects engaged in the early part of an assignment period. They have adopted an optimization strategy of deferring work until it is necessary. Some active procrastinators also increase their flow – their focus – by waiting until work is necessary. Note that the uptick in Figure 7 at 24 hours occurs only for students with high Cgpa, and only for a very small fraction of those high-Cgpa students. Passive procrastinators, in contrast, defer work out of avoidance or poor time management skills. They do not do well. Figure 8 shows that within-student performance also drops off when mean work sessions are less than an hour long.

The author has used these analyses in scheduling classes. All classes take place in computer lab classrooms, all classes are at least 75 minutes long, and the author gives students at least a 60-minute work session at the start of each project cycle, after they have had a weekend to read the assignment materials. Thus, both the Jstr lead time and the Mavg work time constraints for grades are met.

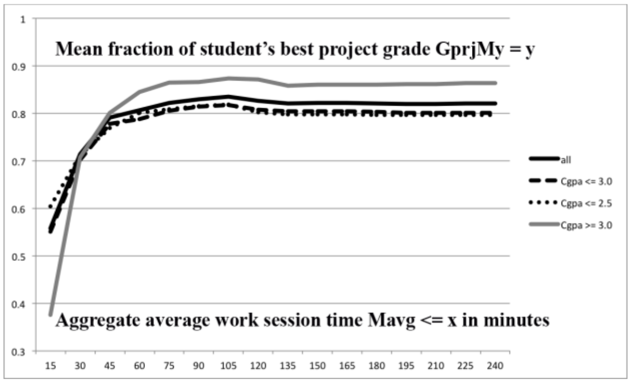


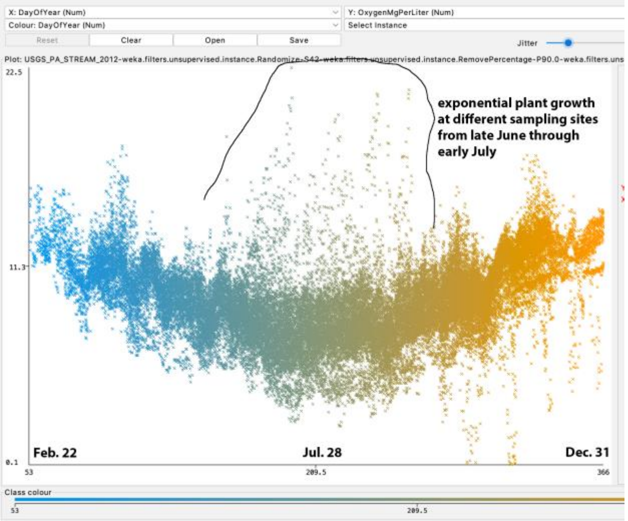
Figure 8: Within-student grade as a function of work session

4.2 Analyzing USGS stream flow data relationships

Analysis of the USGS PA stream flow data [8] was the first of the author's data science projects used by students. Two essential points are that 1) the USGS PA stream flow data analysis was not a novel research project and, 2) verifying results relied on research papers that had already been published. Having a stable target analysis in the form of established scientific facts was essential for giving our first data science students reliable projects.

The author and students used Weka regression models to establish the relationships of the non-target attributes enumerated in Section 2.2 to OxygenMgPerLiter (dissolved oxygen in milligrams per liter). Figures 9 and 10 use Weka scatterplots to graph the main points. Each data record in one year's recording of 46 automated sampling sites, with over 490,000 records in all, appears as a point in these scatterplots. Figure 9 shows the level of dissolved oxygen lowest during the summer, caused by the inverse relationship of water temperature and the ability of water to hold dissolved oxygen [22], accounting for the overall U shape of Figure 9, with the low point in midsummer.

In the third course in which we used PA stream flow data the author had the students "crowd source" the selection of water sampling sites to use as training data. There are a lot of potential training sites and a good number of students. Ideally, training data should be distinct from testing data, so as not to "load the dice," so to speak. Training data should be a representative cross section of overall data. It should not bias trained machine models towards special cases, a condition called "over-fitting". Students were searching the space of potential training data sites in parallel. The most inappropriate sites were those missing several months of data for the year.



X: DayOfYear (Num)

Y: OxygenMgPerLiter (Num)

Colour: DayOfYear (Num)

Select Instance

Plot: USGS_PA_STREAM_2012-weka.filters.unsupervised.instance.Randomize-S42-weka.filters.unsupervised.instance.RemovePercentage-P90.0-weka.filters.uns

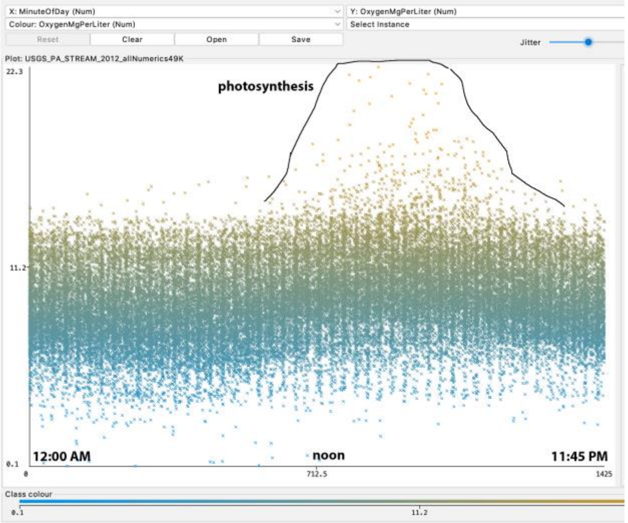
Reset
Clear
Open
Save
Jitter
22.5
11.3
0.1
exponential plant growth at different sampling sites from late June through early July
Feb. 22
Jul. 28
Dec. 31
53
209.5
366
Class colour
53
209.5

Figure 9: Dissolved O_2 as a function of day of year

One of the students came to office hours with graphed data from one of their assigned sampling sites on the Schuylkill River in Philadelphia. Their data showed an upward spike in dissolved oxygen during a few days in the first week of July, followed by a rapid decline back to typical numbers. We could find nothing special in searching for the date. Later that week the author inspected data from two sites upstream from that site, one near Norristown and another further upstream near Royersford. What the author found was that the upstream sites' O_2 levels spiked a few days earlier, and that upstream spikes were a little lower. The author did a literature search and found that an exponential growth in stream plant life in late June or early July, followed by a fallback in plant density, was an established scientific fact. Photosynthesis from the exponential plant growth accounts for the spikes. Different spikes at different times from different sites appear in the highlighted region of Figure 9.

Figure 10 shows these same records, this time graphing dissolved oxygen as a function of the minute of the day. Since there are the same minutes of the day in all 4 seasons, most of Figure 10 just averages out any pattern. However, note the highlighted outlier instances near the top right of Figure 10, from late morning through the evening. That region shows the existence of increases in O_2 levels in a diurnal pattern from late morning through evening, thanks to sunlight-induced photosynthesis. Plotting or modeling only the months of the growing season show daily rises and falls in O_2 levels thanks to diurnal photosynthesis.

Using data that reveals established scientific facts, and using student crowd sourcing to investigate potential training data sources, are two pedagogical findings from this PA stream flow case study.



X: MinuteOfDay (Num)

Y: OxygenMgPerLiter (Num)

Colour: OxygenMgPerLiter (Num)

Select Instance

Plot: USGS_PA_STREAM_2012_allNumerics49K

Figure 10: Dissolved O_2 as a function of minute of day

4.3 Analyzing .wav files for classification & regression

Unlike the Java programming student data, the stream flow data, and the climate-to-raptor count data, the author had complete control over the creation of the .wav file audio dataset and its anticipated analysis.

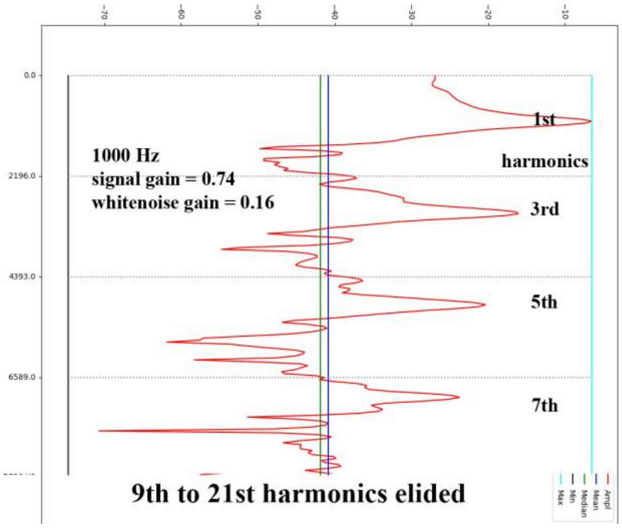


Figure 11: Frequency domain histogram of a square wave

Figure 11 shows a frequency-domain plot of a 1000 Hz. (cycles per second) square wave with a signal gain of 0.74 on a scale of 0.0 to 1.0, and a white noise gain of 0.16. This plot is essentially a histogram of signal strengths at frequencies across the audible range of 0 to 22,050 Hz., with the upper harmonics elided to reduce figure size. The fundamental frequency of 1000 Hz. is labeled **1st** at the top. A square wave consists of only odd harmonics – 3000 Hz., 5000 Hz., etc. for a 1000 Hz. fundamental frequency – with those odd harmonics decaying in level at an established rate. Triangle waves also populate only odd harmonics, but at a different decay rate than square waves. Sawtooth waves and pulse waves populate even and odd harmonics but at differing decay rates. Sine waves populate only the fundamental (1st) frequency. The harmonics and their decay rates act as signatures for waveform type classification.

After several course offerings using these waveforms, the author determined a data representation approach that yields the most accurate classification of waveform types. Normalize the strongest, fundamental frequency to 1.0 and its amplitude in the histogram of Figure 11 to 1.0. These are attributes freq1 and ampl1. The second strongest frequency in the data is freq2 as a multiple of freq1, and its amplitude ampl2 is a fraction of ampl1. Continue through the 32 strongest amplitude peaks, normalizing the frequencies as multiples of freq1 and their amplitudes as fractions of ampl1. This data representation approach provides perfect results for classification of waveform type. Below is the Weka OneR rule that considers only a single non-target attribute and yields a perfect classification for 10,005 .wav files. The decayed amplitude of the first multiple of the fundamental frequency, relative to the fundamental amplitude, is enough of a “finger print” to distinguish the waveform type.

```
ampl2:
    < 0.056232   -> SinOsc, < 0.223464     -> TriOsc
    < 0.415814   -> SqrOsc, < 0.7182685    -> SawOsc
    >= 0.7182685 -> PulseOsc
```

The point for this case study is that the most effective data representation format for analysis may not be the most obvious or the simplest mapping of the raw data. It took several iterations of analysis of this dataset to arrive at its most effective structure for classification of wave type.

An even later discovery of the best data representation format for the white noise level occurred in a course in fall 2023. The white noise gain of 0.16 accounts for the non-harmonic squiggles in Figure 11. While white noise is in principle uniformly distributed across the frequency spectrum of Figure 11, true uniform distribution is approached only as the number of sample values in the histogram approach infinity. There are 22,050 points in Figure 11’s histogram, so non-uniformity in levels induced by white noise appears.

In fall 2023 we took the statistical min, max, mean, and median of all values in each .wav file frequency domain histogram, and correlated those statistical measures to the white noise gain. What we found is that the median signal level correlates very closely with white noise gain. Min and max by their nature are extremely non-uniform, leaving mean and median. Max in particular is affected by the non-noise signal gain. Moreover, extremes of either min or max can weigh on the mean more than on the median. In the full dataset of over 490,000 audio samples, median has a correlation coefficient of 0.964257 with the tagged, target white noise gain level, where a correlation coefficient of 1.0 is perfect correlation and of 0.0 is no correlation. The mean of values in each histogram, in contrast, has a correlation coefficient of only 0.063363. Mean is pulled away from the white noise level by the signal peaks.

Again, the point for this case study is that the most effective data representation format for analysis may not be the most obvious or the simplest mapping of the raw data. It took even more iterations of analysis of this dataset to arrive at its most effective structure for regression of white noise gain than it did for classification of wave type.

4.4 Analyzing climate change to raptor correlations

The page limit and prospective length of this discussion prohibit doing it here. The reader is directed to an extensive write-up from the summer of 2022 and the summer of 2023 [16,17]. Dr. Laurie Goodrich’s expertise, the Hawk Mountain datasets, and the time to investigate them have been invaluable in upper-level and graduate data science courses and in one master’s thesis.

The current state of this research is summarized at the bottom of reference [17]: “For the prime observation months of October and November … wind speed measures that correlate strongly with declining raptor species counts are consistently declining during observation periods. There are three potential hypotheses about the declining raptor counts. A) Diminishing updrafts on the north-northwest side of the Kittatinny Ridge are leading the raptors to cross the mountain at more widespread locations instead of funneling them past North Lookout and across the Eckville Fault. B) Raptors are wintering further north, perhaps due to increasing temperatures. C) Raptor populations are declining in numbers. The next step in this investigation is to look for trends in the raptor counts during the spring, northerly migration. If there has been no significant change in the last quarter century, that would indicate alternative (A).” The author and a graduate student will continue to explore this data in summer 2024.

5. Conclusions

Having access to domain experts and their data, learning to identify data deficiencies and to code scripts to clean, augment, and structure them, and acquiring the knowledge and skills to apply machine learning models for data relationships are all essential, non-trivial aspects of preparing and teaching a substantial data science curriculum. Sections 2.5 and 3.5 provide conclusions for data domain expertise, data acquisition, and data preparation. Like data analysis, teaching data science is an iterative, deepening process.

While the data domains and projects surveyed in this report are custom and unique to the author’s courses, students are welcome to use off-the-shelf data sources such as Kaggle in doing individual projects in several courses [23].

6. Acknowledgements

Dr. Laurie Goodrich of Hawk Mountain Sanctuary has supported that research project in ways that deserve the author's and students' thanks. Dr. Lisa Frye, the author's department chair, has been very supportive of this adventurous curricular exploration.

References:

[1] Bilogur, Aleksey, "Simple techniques for missing data imputation". Kaggle Notebook, 2018, https://www.kaggle.com/code/residentmario/simple-techniques-for-missing-data-imputation/notebook.

[2] D. Parson and A. Seidel, "Mining Student Time Management Patterns in Programming Projects," *Proceedings of FECS'14: 2014 Intl. Conf. on Frontiers in CS & CE Education*, Las Vegas, NV, July 21 - 24, 2014.

[3] D. Parson, "Using Weka to Mine Temporal Work Patterns of Programming Students," Tutorial at 2014 Intl. Conf. on Frontiers in CS & CE Education, Las Vegas, NV.

[4] D. Parson, L. Bogumil & A. Seidel, "Data Mining Temporal Work Patterns of Programming Student Populations," *Proceedings of the 30th Annual Spring Conference of the Pennsylvania Computer and Information Science Educators* (PACISE) Edinboro University of PA, Edinboro, PA, April 10-11, 2015.

[5] Witten, Frank, Hall, Pal, *Data mining: practical machine learning tools and techniques*, fourth edition, (San Francisco, CA: Morgan Kaufmann / Elsevier, 2016).

[6] Witten, et. al. "Weka 3: Machine learning software in Java," https://www.cs.waikato.ac.nz/ml/weka/index.html, 1993-2024.

[7] Maiden Creek Watershed Association, https://berksnature.org/water/maiden-creek-watershed-association/ .

[8] US Geological Survey, "USGS Water Data for the Nation," historical data, https://waterdata.usgs.gov/nwis.

[9] D. Parson, P. Beatty and B. Schlieder, "A Tcl-based Self-configuring Embedded System Debugger," *Proceedings of Fifth Tcl/Tk Workshop*, USENIX, July, 1997.

[10] ChucK Team, "ChucK Music Programming Language", https://chuck.stanford.edu/ , 2003-2024.

[11] Laurie Goodrich, Ph.D., "Sarkis Acopian Director of Conservation Science," Hawk Mountain Sanctuary, https://www.hawkmountain.org/about/community/staff/laurie-goodrich.

[12] Python *math* library for the *isclose*(), almost-equals function, https://docs.python.org/3/library/math.html.

[13] Python *re* library for parsing text via regular expressions, https://docs.python.org/3/library/re.html.

[14] The *pythex* utility for testing *re* regular expressions, https://pythex.org/.

[15] Documentation for the SciPy scientific computing library, https://docs.scipy.org/doc/scipy/.

[16] D. Parson, "Analysis of Hawk Mountain Sanctuary Observation Data from 1976 through 2021," white paper, https://research.library.kutztown.edu/cisfaculty/20/ , 2022.

[17] D. Parson, "Analysis of Hawk Mountain Wind Speed to Raptor Count Trends from 1976 through 2021," https://research.library.kutztown.edu/cisfaculty/19/ , 2023.

[18] National Oceanic and Atmospheric Administration (NOAA), National Centers for Environmental Information, Allentown Lehigh Valley International Airport data, 1948 through 2021. https://www.ncdc.noaa.gov/cdo-web/datasets/GHCND/stations/GHCND:USW00014737/detail

[19] U.S. Environmental Protection Agency, "Heat Island Effect". https://www.epa.gov/heatislands.

[20] Angela H. C. Chu and Jin N. Choi, "Rethinking Procrastination: Positive Effects of 'Active' Procrastination Behavior on Attitudes and Performance," *The Journal of Social Psychology*, 2005, 145(3), p. 245-264.

[21] E. Kim and E. H. Seo, "The Relationship of Flow and Self-regulated Learning to Active Procrastination," *Social Behavior and Personality*, 2013, 41(7), p. 1099-1114.

[22] USGS, "Dissolved Oxygen and Water," June 5, 2018, https://www.usgs.gov/special-topics/water-science-school/science/dissolved-oxygen-and-water.

[23] "Kaggle: Your Machine Learning and Data Science Community," https://www.kaggle.com/.