Front-End Development of a Learn2Mine, a Gamified Learning System

An essay submitted in partial fulfillment of

the requirements for graduation from the

Honors College at the College of Charleston

with a Bachelor of Science in

Computer Science

Jacob Dierksheide

May 2014

Advisor: Paul Anderson

Abstract

Education through the web has become an increasingly popular phenomenon in recent years. Online applications like Codecademy, Khan Academy, and Rosalind allow their users to practice a variety of programming languages, core academic subjects, or other personal skills through hands-on lessons and examples. These websites offer an attractive new ideology for the learning process. They each fill a specific need for accessible guidance for newcomers to these different fields that are becoming more and more necessary in today's connected world. There is, however, a void in the pantheon of these educational tools that need to be filled. None of these sites teach true data science.

Data science is a rapidly growing subject in academics, with many universities recently beginning to offer major courses of study in the field (Anderson, Argamon, Cheshire, Dubois). Data science combines the needs for the application knowledge of computer scientists and mathematicians with the specific domain expertise of the other scientists in order to gain knowledge from existing data. One popular realm of data science, for example, is bioinformatics. Those practicing bioinformatics may use their familiarity with gene sequences and their programming capabilities to perform algorithms that may expose differences in expressed gene traits between treatment groups. The practice of data science therefore requires a person to be well versed in these many different fields. Attempts have been made to establish a web presence for this sort of merging between a domain and informatics system through applications like Weka (Hall et. al. 2009), RapidMiner (Mierswa et. al, 2006) and Taverna (Woltenscroft et. al.), however there does not yet exist a platform that gives users exposure to all of these skills in a way that is efficient. Learn2Mine has been introduced in an effort to fill that void. Learn2Mine is an application consisting on 3 main parts: a Google App Engine web application as a front-end, a Galaxy Project server, and an RStudio server. Each of these plays an important role in the makeup of Learn2Mine, and interacts with the others to give users an extremely interactive gamified platform through which to learn data science.

I. Introduction

A. Emergence of Data Science

As the analysis of large datasets becomes a more sophisticated and necessary field of study, many major universities have begun offering degrees in Data Science or related topics like Business Analytics or Data Mining. University of Washington, New York University and the University of Southern California are just a few of the major institutions that offer a Data Science major by name (Swanstrom 2012). For a long time data scientists had no individualized training, but rather were people pulled from the math, computer science, or domain science professions who were forced to adapt the skills they had learned from their respective fields. Figure 1 shows the sources of education for people who end up practicing data science. Data scienctists can come from a variety of tangentially related or unrelated fields, however no single one of these backgrounds on its own could completely cover the skill set needed to truly practice data science. A pure computer scientist has the experience in programming and creating that is required to automate the analysis of big data. Mathematicians contain the knowledge of theory and statistics necessary to create the algorithms for performing that manipulation. And finally domain experts are familiar with the specific sciences and businesses needed to interpret the results of an analysis. Each of these represent an important piece of the data science puzzle, but none of them in themselves are able to optimally practice data science, therefore a specific program is needed to equip data scientists with the skills from each of these fields. For this reason, major schools of higher education offer specific data science classes, however there is little presence of proper education in the field on the web. Most other domains have websites tailored toward effectively teaching the skills in an individual setting, Codecademy teaches Python and JavaScript, Duolingo teaches foreign languages, but no major educational website has come forth offering courses in data science. This is why Learn2Mine was created: to give users a place to experience and practice all the different skills necessary to being successful in data science.

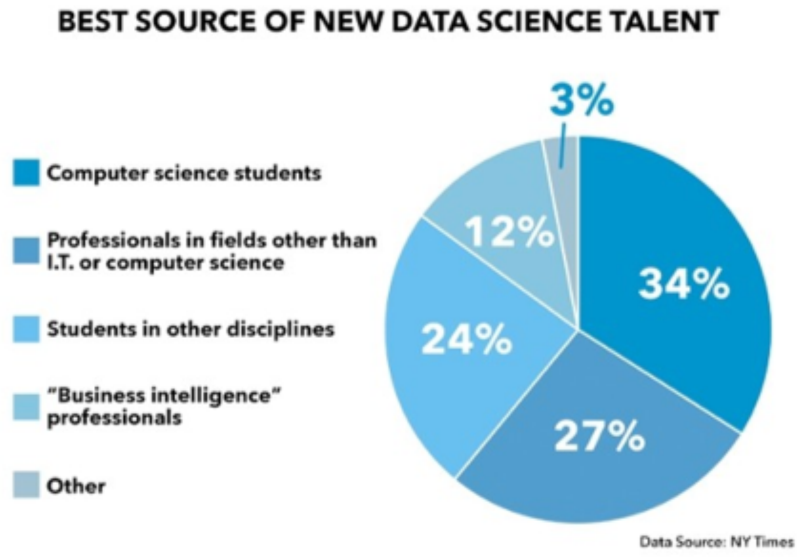


Figure 1: Where Data Scientists Come From

B. The 3 Pillars of Learn2Mine

Compositionally, Learn2Mine is a web application that consists of the marriage of three major parts. A Google App Engine front end is where Learn2Mine's personality lies and where the user is persisted. Much of the heavy lifting, however, is done through two additional project servers: RStudio and Galaxy. Together, these three allow for a very comprehensive experience in data science education.

Google App Engine is a Platform as a Service that lets people host their web applications on Google's infrastructure. Users can upload entire application directories, written in Python, Java, or PHP to be served by Google through their Appspot website. This comes complete with a Google Query Language (GQL) database that allows the persistence any necessary application data. This branch of Learn2Mine is where users can look at their profile, find new lessons to practice, and track their progress.

While Google App Engine is the face of Learn2Mine, and is where the User's profile as they know it exists, the meat of application lies in its integration with a Galaxy Project server. The Galaxy Project is an open-sourced platform for running data intensive algorithms (Blankberg et. al. 2001, Giardine et. al. 2005, Goecks et. al. 2010). Here is where the user's responses to the lessons are evaluated, and feedback given accordingly. For Learn2Mine's purposes, Galaxy has been extended with set of additional tools for the various lessons Learn2Mine offers. Some of these tools are to be run directly on the Galaxy Project server, while others are accessed through interfaces built directly in to the lessons pages located on Google

App Engine. The outputs of these specialized lessons are sent to a single Grade tool, which compares the results to the expected output and if earned, awards the user a badge. This process will be described in greater detail later.

Learn2Mine also makes use of an RStudio server to give users a place to write and run code written in R, a powerful language for data manipulation that Learn2Mine offers lessons in using. RStudio is a cloud-based Integrated Development Environment (IDE) that allows its users to write, run, and save R scripts, as well as interact with a console for more dynamic R coding. This facet of Learn2Mine serves as a useful tool for the practice of the exercises that it provides. Users can then copy their code and/or output (depending on the problem specifications) for submission to Galaxy. RStudio also allows personalized directories through users on the machine on which it is being run, therefore each person who logs into Learn2Mine is given their own personal workspace here they can save, modify, and upload any files they might want to interact with through the IDE.

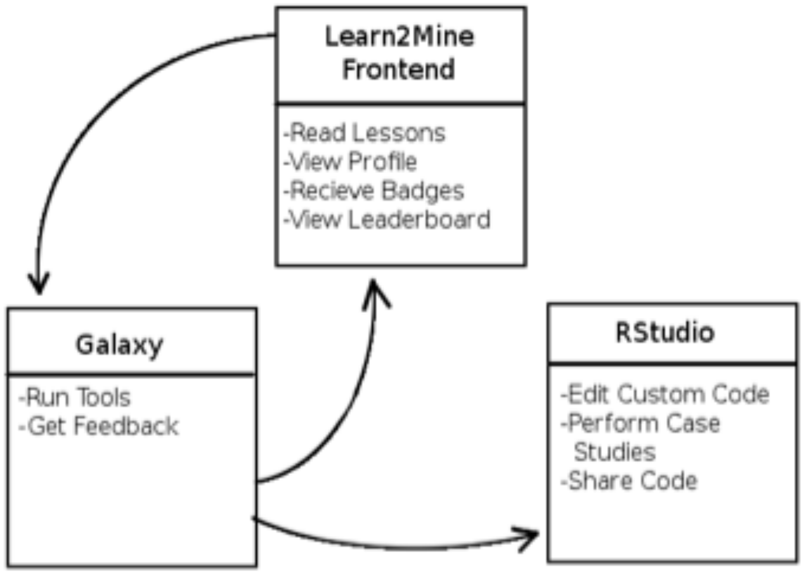


Figure 2: Learn2Mine Components and their Interactions

C. Gamification

The ideology that drives the development of Learn2Mine is the use of gamification in the learning process. Gamification's aim is to bring the positive aspects of games together with education in order to make the task of learning more engaging and rewarding. It is important to note that gamification is not simply learning through educational games (also known as "edu-tainment"), but is the implementation of popular game concepts in a non-game setting (Deterding et. al. 2011, Gerber 2012). These concepts include things like reward schedules, feedback, public reputation, advanced user paths, and content unlocks (Direkova 2012, Glover 2003, Muntean 2011, Nah et. al. 2013). Learn2Mine attempts to make use of these features in order to make its users' experience highly goal-oriented and absorbing, as gamification has been shown to increase a user's engagement with an application (Thom et. al. 2012). Through an unlimited allowance of lesson submissions, Learn2Mine grants the permission to fail, and gives instantaneous feedback, which is a much more efficient reward system than the delayed feedback of a more traditional classroom education setting.

II. Methods of Design

A. User Identity

An important facet of gamification is a sense of user identity. When a user first accesses the Learn2Mine they are asked to log in using a Google account. This log in is authenticated through OAuth 2.0, which is a standard and open protocol for application authorization on the web (Oauth, 2014). The use of this Google account makes Learn2Mine easily accessible to newcomers, and means that there is no additional memorization of any additional username or passwords. Once the login has been accepted, a query is run to see if a user with that email already exists in the datastore. If the user is found, they are simply redirected to the Home page. If the user does not exist, a new object of the User class is created and saved into the datastore using a put() statement. This User class contains 5 important fields: EMAIL, BADGESEARNED (a list of the badges that the user has completed so far), BADGESDUE (a list of the badges the user is owed, but have not been awarded yet), SKILLSEARNED (a list of the broad skill categories that have been completed, that do not correspond to a lesson), and SESSION (a unique 21-digit integer used to identify the user).

B. Badges

The goal of users on Learn2Mine is to complete lessons on the various algorithms and skills it provides in order to earn badges. These badges serve as the primary reward system of the website, and supply tangible proof of the fact the user has mastered that skill. The implementation of these badges is done through another increasingly popular mechanism: Mozilla OpenBages. OpenBadges is an online standard hosted by Mozilla that allows the free presentment of badges by any organization, which can then be added to a "Mozilla Backpack" and displayed on various online profiles to serve as a sort of "virtual resume" for the user (Mozilla, 2014). These badges are not proprietary, and are simply held and managed by Mozilla. The authenticity of these badges therefore must be provided by the awarding organization through JSON files kept by that organization. When a user submits a lesson for grading to Galaxy and the lesson is deemed "mastered" by the grader, a new JSON file for that user is written on the server side. This JSON file includes the user's email, the lesson they have completed, the time it was completed, an image representation of the badge, and links to additional JSON files that authorize both the specific badge and the organization granting it. The link to this publicly available file is then sent through an interface to Mozilla, which verifies the different component fields of the JSON object and, if all are acceptable, passes the link to the user's backpack where it is then saved. An example of what these badges look like can be seen in Figure 3.

Figure 3: Example Learn2Mine Badge

C. The Skill Tree

While users can employ their Mozilla backpack to hold their badges and display them on their own personal website, Learn2Mine uses it's own method of displaying them in the form of an interactive skill tree. The skill tree serves as the user's "Profile" on the Learn2Mine front-end website. Though users are free to do the lessons in whatever order they please, the flow of the skill tree and logical progression of the subjects within it satisfy several important aspects of gamification like increasing complexity and content unlocks. By giving users a suggested, but not absolute path through which to traverse the sites lessons, Learn2Mine allows them to learn at their own pace whether that means moving through lessons one at time, taking a breadth-first approach to the tree, or jumping right into the more advanced lessons. The skill tree begins completely grayed out, and becomes colored in as the user completes each skill, giving another layer of visual feedback to their accomplishments.

The actual skill tree is rendered by use of a jquery tree object that is outlined on the page Profile.html through JavaScript. The actual content of the graph is defined through a JSON variable called networ_json. This variable contains a list "nodes", filled with representations of graph nodes with an id for the node and a link that the node should point to (the lesson's page), as well as a list of edges, which define a source and a target node id that should connect 2 nodes on the graph. Another variable, style, has a list of image links paired with the node ids that fill the skill tree with the badge images. The links to these images, rather than be directly hard-coded into the Profile.html file, are generated dynamically through a function makeTemplateValues() which is called from the file skillTree.py, and passed through template values when the page is rendered by Google App Engine's main. Adding new lessons to the skill tree is therefore as easy as defining a new "node" and a new "edge" for the desired skill to the networ_json variable, and adding the lesson identifier to makeTemplateValues()

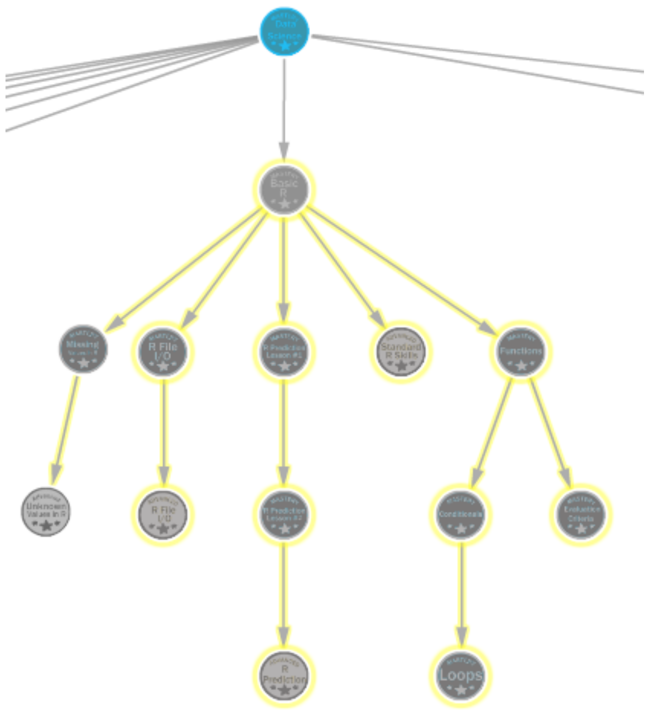
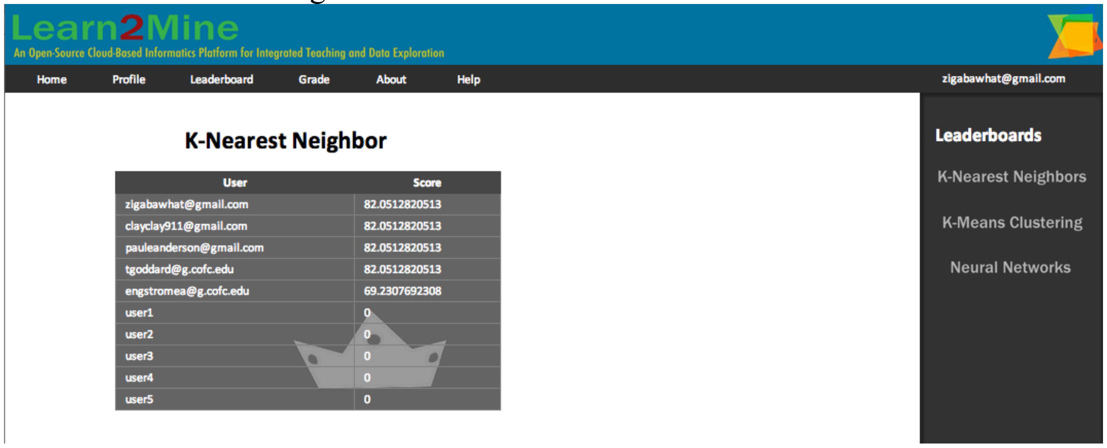


Figure 4: The R branch of the Skill Tree

D. Leaderboards

Another common theme in the implementation of gamification in anything is the development of a sense of community between the people engaging in it. This allows for competition and collaboration to drive the educational process. Learn2Mine seeks to provide this in the form of Leaderboards that shows the top scoring users along with their record-setting scores publicly on the site. The ability of a lesson to have a leaderboard of course depends upon the content of that lesson, however a few prototype leaderboards have been made for lessons in which they make sense. The k-Nearest Neighbor lesson, for example, asks users to successfully classify as many test cases as they can of a famous diabetes training set as they can using either their own algorithm (the advanced track) or one that is provided for them (the standard lesson). Though the standard lesson can only produce so many different levels of success due to its finite number of inputs, by writing their own k-Nearest Neighbor function using sophisticated processes like genetic algorithms the rate of success can be drastically improved. When using the kNN tool, a special "score" variable representing the percentage of correctly classified cases is passed along to Learn2Mine. This value is then checked against a python list stored in the Google Datastore that contains the current top 10 scores for that lesson. If this score beats any of the ones in question, it is placed within the list and the rest of the top 10 are shifted down 1 slot each. Users can therefore measure their success in a lesson against the rest of the Learn2Mine community, and hopefully use their place on the leaderboard (or lack thereof) as motivation to rework and improve their algorithm. When the user accesses a leaderboard page, it pulls this top10 list for the appropriate lesson, and generates an HTML table by iterating through it. The result of this can be seen in Figure 5.



K-Nearest Neighbor

User	Score
zigabawhat@gmail.com	82.0512820513
clayclay911@gmail.com	82.0512820513
pauleanderson@gmail.com	82.0512820513
tgoddard@g.cofc.edu	82.0512820513
engstromea@g.cofc.edu	69.2307692308
user1	0
user2	0
user3	0
user4	0
user5	0

Figure 5: k-Nearest Neighbor Leaderboard page

E. Google App Engine Application Structure

Hosting an application on Google App Engine is as simple as uploading a directory with a few necessary components to Google Servers using their SDK for the appropriate language. The first is a YAML file that needs to be called app.yaml. This file defines several important pieces of information about the application, including name, language specifics, and version. Any handlers for scripts or static directories that are to be interacted with in the application code are to be declared in this file as well. For example, Learn2Mine is able to access images in the /images directory because of the following code snippet found in its app.yaml

```
handlers:
        - url: /images
```

```
    static_dir: images
```

Most important in this under this list of handlers is that of url: ./*, this must point to a WSGIApplication object (WSGIApplication is a class that can be found in the webapp2 framework. Here Learn2Mine has:

```
- url: ./*
  script: main.app
```

This tells Google App Engine that when a user requests to learn2mine.appspot.com, the application is to look to the app variable in the top directory's main.py file to decide how to handle that request. The WSGIApplication is instantiated with a list of strings/class pairs to tell it what how this redirection should work. For example, one such pair is ("/Home", HomeHandler). Therefore when a user requests the page learn2mine.appspot.com/Home, the HomeHandler class is called to render the page.

One advantage to building a web application as opposed to simply a series of static HTML pages is the ability to use templates to generate content on each page. A template allows for variables, or even stretches of code to be written into the HTML files, which then are filled in or executed by the application upon rendering the page. Learn2Mine uses the JINJA template engine to perform all of its rendering. A common example of this is the display of the currently logged in user on each page of Learn2Mine. Within the div block that holds the navigation bar on each page of HTML is the text {{user}}. The handler for each page makes a python dictionary for with an entry for each piece of the template that that page needs filled in. In this case, the datastore is queried for the currently logged in user, and the EMAIL field of that user is stored in a variable called "email" and passed into the dictionary as {'user':email}. This dictionary, called template_values is then given to the HTML through the statement:

```
self.response.write(template.render(template_values))
```

This statement replaces all elements that match the key of any item in the dictionary, surrounded by double braces (in this case {{user}}) with the matching dictionary value (email). It also evaluates any other code statements that may be present in the page.

F. Learn2Mine's Layout

The bulk of the user's experience with Learn2Mine is decided by their interaction with the Google App Engine front end of the application, therefore a lot of time was spent trying to make this face of the operation as clean and friendly as we could. Nearly every page on Learn2Mine features the same minimalistic layout. A thin banner, reading the title of the site Learn2Mine spans the top of the page horizontally and links back to the Home page. Floating to the right of this is the logo for the Anderson lab, the umbrella under which the project resides, these can be seen at the top of Figure 6. The color scheme for the Learn2Mine logo is blue, red, yellow, and green all in deep, understated shades. These mirror the 4 colors that represent the different branches of the Anderson lab. The interplay between these two logos creates a visually interesting but not overwhelming header to the page. Directly under this heading is a thin black bar that provides the necessary links for navigation around the main areas of the site (Home, Profile, Leaderboards, Grade, About, and Help). In the early stages of Learn2Mine these links were more colorful, also reflecting the core 4 colors, but were just hyperlinks. These links were later redesigned to be colored buttons, with hand-drawn icons signifying each of the link's locations. These buttons later proved to be a bit overpowering and busy, and the hover image would often take a fraction of a second to load, that would make the button seem to disappear, which did not look very good to interact with. For these reasons this navigation bar was later redesigned to a simpler charcoal and white colored button, with clean type centered text. The hover interaction was made to be simply a lighter shade of the background color, so the application no longer needed to retrieve the image on the user's hover over the link, giving the buttons a more responsive feel. Floating right of these buttons is the current user's email, as was described earlier.

These two components: the banner and the navigation bar, make up a uniform heading across all of Learn2Mine. Beneath that is then the main content of the page in question. For most pages, this merely consists of any relevant information/links about the page (a Lessons Page has the text for the appropriate lesson, the About page has the Frequently Asked Questions, etc.), however the exception to this is the Home page which features and additional component: the Pageslide. This pageslide comes from the jquery library and is a stationary tab on the right hand side of the page that displays the user's earned badges. When the user loads the Home page, the current User is queried from the database, and their BADGESEARNED field is retrieved. This field is a list of lesson identifiers in the form of string, which can be linked to their appropriate badge images by appending "Mastery.png" to each. Each item in the list is therefore placed into a string representing an HTML image tag like this:

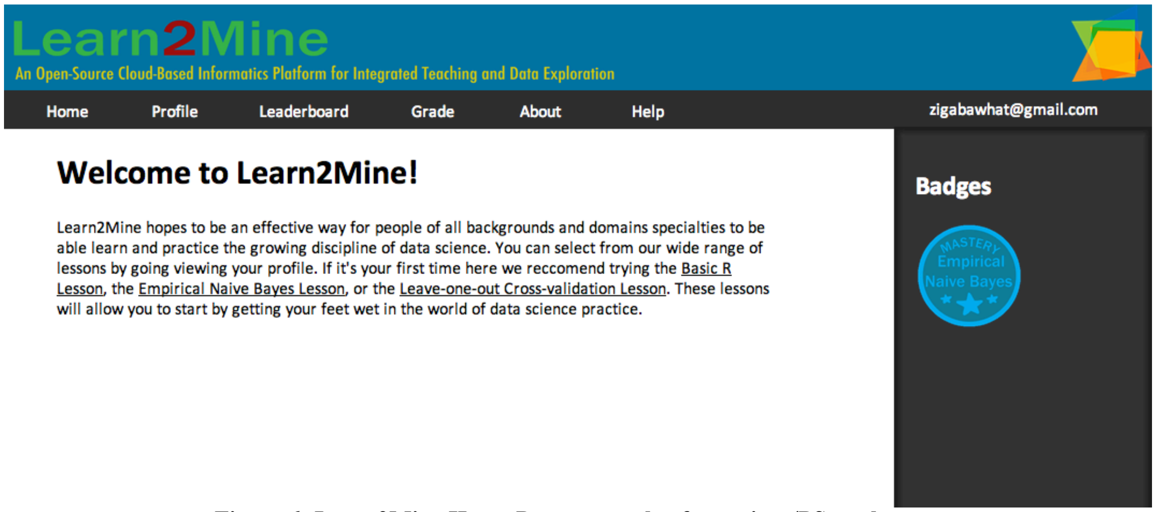
```
    "<img src=" + user.BADGESEARNED[i] + "/>"
```

all strings are joined together resulting in a single long string of html image tags, which is passed into the pageslide using a template, which will be discussed in more detail later. The outcome of this can be seen one the right hand side of Figure 6 below.

Learn2Mine has to account for the fact the people of all different screen orientations and resolutions will be potentially viewing the website. The content of the site looks best when given a block (in html terms, a "div") of fixed width that is centered within the page. This way, a narrow screen will not force the text to be squished horizontally, making the page very long and causing unreadable breaks in the words, but at the same time a very wide screen will not make the text appear too sparse. 960 pixels was chosen as the standard width for the bloc, since on most laptop resolutions this will fill the majority of the page space while still leaving things tight enough to not appear stretched. Centering the block serves to break up the expanse of white space off to the right that would appear in overly wide screens with a fixed width content area as well. Making this stylistic "rule" is not excessively difficult with the user of a Cascading Style Sheet (CSS), which is a handy tool in formatting HTML. The centering, which is done uniformly across all pages, is signified by a defining class in the main.css file, which looks like this:

```
    .container{
            margin-left: auto;
            margin-right: auto;
    }
```

This means that any div class in the HTML that is given the attribute "class=container" will have its horizontal margins automatically sized according to the width of the page, effectively centering it. The only significant factor that affects how these containers should be formatted are the existence of a pageslide. Most pages, which do not contain the pageslide, are marked by the id "NPStext" (for No Pageslide Text) given to the div of the container class which specifies that left and right padding should be 50 pixels, pulling it away from the edges a little bit and making it more easily readable by the user. Otherwise, on the Home page, the same class is marked by "PStext" and is given a left padding of 50, but a right of 350, pulling it away from the pageslide (which is of width 300 pixels), and making sure no text gets hidden underneath it.


Learn2Mine
An Open-Source Cloud-Based Informatics Platform for Integrated Teaching and Data Exploration
Home
Profile
Leaderboard
Grade
About
Help
zigabawhat@gmail.com
Badges
MASTERY Empirical Naive Bayes

Welcome to Learn2Mine!

Learn2Mine hopes to be an effective way for people of all backgrounds and domains specialties to be able learn and practice the growing discipline of data science. You can select from our wide range of lessons by going viewing your profile. If it's your first time here we reccomend trying the Basic R Lesson, the Empirical Naive Bayes Lesson, or the Leave-one-out Cross-validation Lesson. These lessons will allow you to start by getting your feet wet in the world of data science practice.

Figure 6: Learn2Mine Home Page, example of container/PStext layout

Clicking on any of the badges in the skill tree will bring up the analogous lesson for that skill. Each Learn2Mine lesson walks the user through the necessary information they need to understand the algorithm to assignment that they will be attempting to successfully complete in order to earn their badge. Because each of the lessons require several pages of explanation, it would be inefficient to write up to six or seven separate HTML pages for each individual lesson. Therefore, a single HTML template page was written to house all of the generic lesson information, and is filled in by the LessonsHandler class in the main.py that drives the websites services. When the user navigates to a lesson from the skill tree, they are sent to an address like the following: http://learn2mine.appspot.com/Lessons?lesson=knn1. This directs them to the Lessons template page. The portion of the URL after the "?", known as a query string, provides a key value pair that is used to identify which lesson needs to be rendered on the page. This information is retrieved by the main() function, and then to another python function that takes that value as an input and returns all of the necessary text to fill out the page with the fill lesson. Take, for instance, in the URL example mentioned previously. The lessonWriter() function would accept the term "knn1", and return the body of the first page of the k-Nearest Neighbor lesson, as well as other specifics to that page. Using this method for page creation also serves to make the addition of new lessons very straightforward, which is a feature that Learn2Mine strives to allow. A user with access to the source code of Learn2Mine need only to add an elif statement to the lessonWriter() function, and specify the necessary text about the lesson to be filled in by the template. By writing directly in HTML here they are also able to use additional HTML tags and features not directly listed in the template, if their lesson should require it.

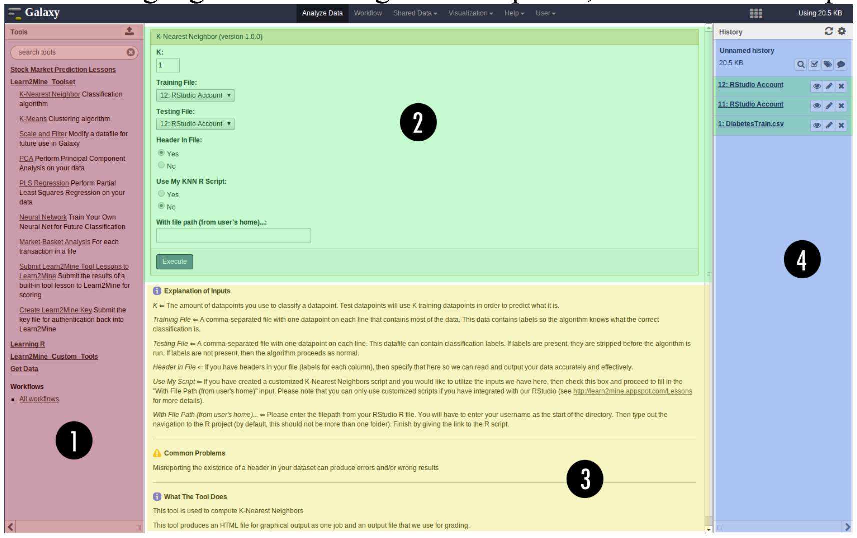
G. Types of Lessons

There are two different types of lessons that appear on Learn2Mine, reflecting different levels of difficulty. The first of these henceforth referred to as "tool lessons" interact with predefined Galaxy Tools. These lessons provide an explanation of a specific algorithm, geared toward anyone with no prior knowledge of data science practices. Once the user has been carefully exposed to the algorithm and its uses, they are presented with a specific dataset on which they are expected to perform it, often with a specified goal. For example, the lesson in the k-Means algorithm provided a set of data recorded from various earthquakes, and asks them to use k-Means to classify the earthquakes in such a way that minimizes within-cluster and maximizes between-cluster variation. Users then interface with a pre-written Galaxy tool in order to perform this operation. Because in these cases the code is all pre-written for them, the user's interaction with the algorithm is fairly minimal in these cases. The k-Means tool for example only really allows the user to set the k parameter to influence their success with it. Though minimal in the users influence over their outcome, the tool lessons allow for a great introductory understanding of how these algorithms work by letting them see how the manipulation of different inputs or methods affects the outcome. These tool lessons also all have specifically formatted outputs with dynamic visual representations of the results, which further aid understanding.

Alternately, there are lessons that require users to write code themselves in order to complete them. These lessons all require the use of a single Galaxy tool (called Grade Code), in order to be graded. Rather than run an algorithm to evaluate output, these R lessons take the output of code the user is expected to write themselves in RStudio as an input. These input results are then parsed into individual problems (as will be explained in greater detail later) and matched to expected values in order to determine whether or not the lesson has been passed. These lessons offer far less handholding than the tool lessons, and are designed to teach the users how to design, write, and practice data science algorithms for themselves, rather than just gain a familiarity with what they do. These R lessons were developed and piloted for the College of Charleston's Data Science 101 class in the fall of 2013.

H. Galaxy Tools

Fitting the Galaxy Project to the needs of Learn2Mine required the extension of it through the addition of new "tools." Tools are the basic driving force behind a users interaction with Galaxy, upon arriving at the home page; they may select an individual tool from a variety of categories ranging from simple Text Manipulation to more complex sets like Multiple Regression Analysis and Metagenomic Analysis, these tools are chosen from the toolbox, which can be seen as the red-shaded area marked 1 in Figure 7 below. Users can then provide inputs to the tool's interface, which could be anything from simple parameter options to entire uploaded datasets. Once these inputs have been set, executing the tool fires off a script with those inputs as command line arguments. This script can be written in any language installed on the serving machine, provided that language is run through an interpreter, rather than a compiled language.



Stock Market Prediction Lessons
Learn2Mine_Toolset
K-Nearest Neighbor Classification algorithm
K-Means Clustering algorithm
Scale and Filter Modify a datafile for future use in Galaxy
PCA Perform Principal Component Analysis on your data
PLS Regression Perform Partial Least Squares Regression on your data
Neural Network Train Your Own Neural Net for Future Classification
Market-Basket Analysis For each transaction in a file
Submit Learn2Mine Tool Lessons to Learn2Mine Submit the results of a built-in tool lesson to Learn2Mine for scoring
Create Learn2Mine Key Submit the key file for authentication back into Learn2Mine
Learning R
Learn2Mine_Custom_Tools
Get Data
Workflows
- All workflows

K-Nearest Neighbor (version 1.0.0)
K: 1
Training File: 12: RStudio Account
Testing File: 12: RStudio Account
Header In File: [✓ Yes] No
Use My KNN R Script: Yes [✓ No]
With file path (from user's home)...:
Execute

Explanation of Inputs

K ⇐ The amount of datapoints you use to classify a datapoint. Test datapoints will use K training datapoints in order to predict what it is.

Training File ⇐ A comma-separated file with one datapoint on each line that contains most of the data. This data contains labels so the algorithm knows what the correct classification is.

Testing File ⇐ A comma-separated file with one datapoint on each line. This datafile can contain classification labels. If labels are present, they are stripped before the algorithm is run. If labels are not present, then the algorithm proceeds as normal.

Header In File ⇐ If you have headers in your file (labels for each column), then specify that here so we can read and output your data accurately and effectively.

Use My Script ⇐ If you have created a customized K-Nearest Neighbors script and you would like to utilize the inputs we have here, then check this box and proceed to fill in the "With File Path (from user's home)" input. Please note that you can only use customized scripts if you have integrated with our RStudio (see http://learn2mine.appspot.com/Lessons for more details).

With File Path (from user's home)... ⇐ Please enter the filepath from your RStudio R file. You will have to enter your username as the start of the directory. Then type out the navigation to the R project (by default, this should not be more than one folder). Finish by giving the link to the R script.

Common Problems

Misreporting the existence of a header in your dataset can produce errors and/or wrong results

What The Tool Does

This tool is used to compute K-Nearest Neighbors

This tool produces an HTML file for graphical output as one job and an output file that we use for grading.

Unnamed history
20.5 KB
12: RStudio Account
11: RStudio Account
1: DiabetesTrain.csv

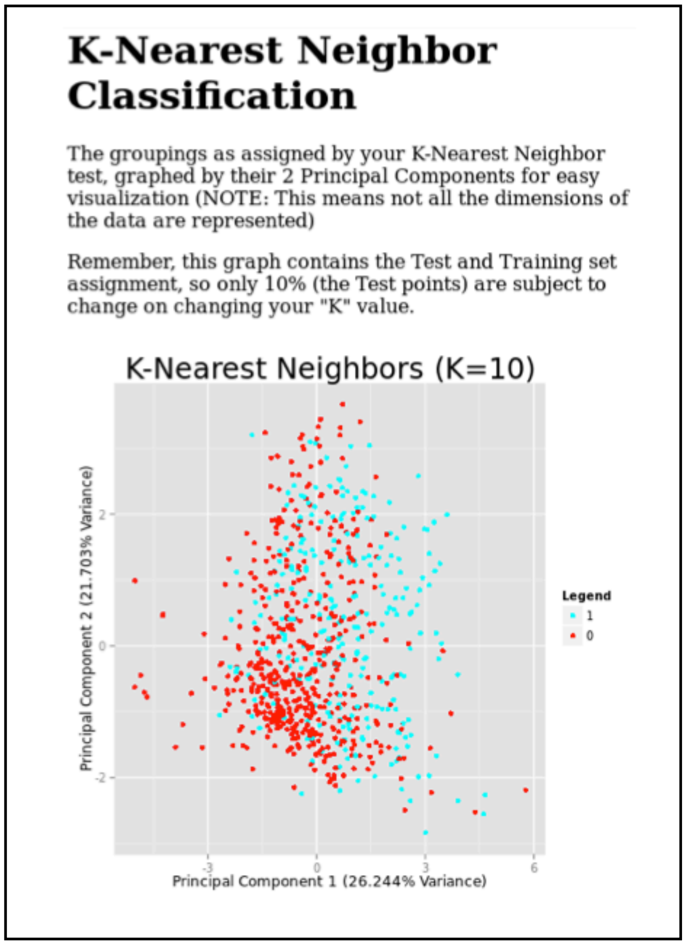
Figure 7: Galaxy Tool Interface

Therefore, each Galaxy tool is made up of 2 primary files: an XML file that defines the tool and necessary parameters, and the script that actually does the work on the outputs. The XML file defines the front-end interface of the tool as it will be presented on to the user. A set of specific XML tags are documented and described on the projects website to help with the development of new tools. Within this file you can define the inputs, outputs, tests, and any sort of description about the tools, which will then be parsed by Galaxy and formatted accordingly. The areas marked 2 and 3 on Figure 7 above show the results of the tags used in the XML file. The green area, marked 2, contains everything declared within the <inputs>. The file knn.xml declares 6 <param> tags within its <input>, which translate to the interface as it appears, these are:

1. An integer (max 3 digits)
2. A Training data file
3. A Testing Data file
4. Exclusive radio buttons for whether or not the data in the files have headings
5. Exclusive radio buttons for whether or not the user wants the tool to evaluate using their personal script
6. A text area for the path to their script (called if input 5 is marked yes)

Below this, the yellow area marked 3, is any information about the tool that may be needed for its use. This, too is defined in the XML within a <help> tag. This information includes common things like information bout each of the inputs, or about frequent errors the tool may run into and how to handle them. Within the help tag are special codes for symbols that may help direct the user's attention toward them, for example typing ..class:: warningmark makes the yellow warning mark that can be seen before "Common Problems" in Figure 7.

The tools added for Learn2Mine's purposes call a Python script upon execution. The number and types of the inputs are dependent on the specific algorithm the tool is made for, however each of these tools generates 2 outputs: the first is simple comma-separated-value representation of the results, while the second is an HTML representation. The CSV output is uniquely and predictably formatted so that the algorithm's outcome may be graded for completion by Learn2Mine, while the HTML is a much more user-friendly embodiment of these results that can be easily understood and provide dynamic feedback about the user's inputs. For the actual performance of the algorithms, the tools added for Learn2Mine's purposes first call a Python script upon execution. This Python script grabs the inputs passed


K-Nearest Neighbor Classification

The groupings as assigned by your K-Nearest Neighbor test, graphed by their 2 Principal Components for easy visualization (NOTE: This means not all the dimensions of the data are represented)

Remember, this graph contains the Test and Training set assignment, so only 10% (the Test points) are subject to change on changing your "K" value.

K-Nearest Neighbors (K=10)
Principal Component 2 (21.703% Variance)
Principal Component 1 (26.244% Variance)
Legend
1
0
-3
0
3
6
-2
0
2

Figure 8: Formatted HTML Tool Output

into the tool, and then uses them as inputs for a second script, which is written in R. Here is where the actual algorithms are performed on the data. While R is very good at this sort of intensive calculation and data management, it is not as straightforward when it comes to presenting it. Therefore, once the algorithm has been finished the necessary outputs are written to a temporary directory (which is initiated for each run of any tool). The location of these outputs are then captured by the initial Python script, which writes a nicely formatted HTML file to display and explain the results to the user, an example of which can be see in Figure 8.

Once the tool has finished running, the two outputs become available in the "History" panel, the blue-shaded area marked 4 in Figure 7. One very advantageous feature of Galaxy is that it supports the output of one tool to be used as input for a next one. Learn2Mine utilizes this in order to perform its grading. This is all done through a specialized tool called the "Submit Data Tool". Users simply select their CSV output from the tool they ran, then pick from a drop-down list which lesson standards they want it graded using. The Submit Data Tool then passes the CSV results along to a grader() function, which parses the data according to the chosen lesson and evaluates it against predetermined thresholds for each lesson in order to determine if the user has passed the lesson. If the lesson is indeed passed (or, in Learn2Mine's terms, if "Mastery" is achieved) then the application on Google App Engine is notified through a POST request, which sends to a specified link (/Grade) the email (which is retrieved as the user is logged into Galaxy), the lesson, and, if applicable for leaderboard purposes, the score of the user's submission. /Grade then responds by appending to the User class associated with that email's BADGESDUE field the name of the lesson that was submitted.

I. Changes Over Time

Since the beginning of its development in the summer of 2013, Learn2Mine's marriage with Galaxy has become more and more streamlined and is further approaching a seamless connection for the user. Initially, Learn2Mine required users to link over to Galaxy for them to perform all of their lessons. Here, the users would input various parameters into the preset tools to learn how the different factors would affect their algorithm's results. This was a good introduction to the algorithms, however it was fairly rudimentary by design, so the concept of "advanced lessons" was added. These advanced lessons asked the users to write their own algorithm in RStudio to try to create a more optimal version of the algorithm. For example, the lesson written for k-Nearest Neighbor challenged them to use a genetic algorithm result in a better classification ratio than the one we prepackage for them is able to achieve with a provided Diabetes dataset. This allowed for users to get a more meaningful learning experience out of the application by requiring them to get their hands dirty and manipulate code for themselves. This was an effective step up in the difficulty of the system, however could be quite a challenging feat for those users who did were not familiar with code, particularly since R is such a unique language. To create an effectively gamified system, it was important to strike an appropriate balance between these levels of freedom. Holding the user's hand too much through the process would devalue their experience and the associated reward system or, alternatively, making it too open ended would rob them of concrete goals to work toward to advance in the "game" (Asaj et. al. 2012).

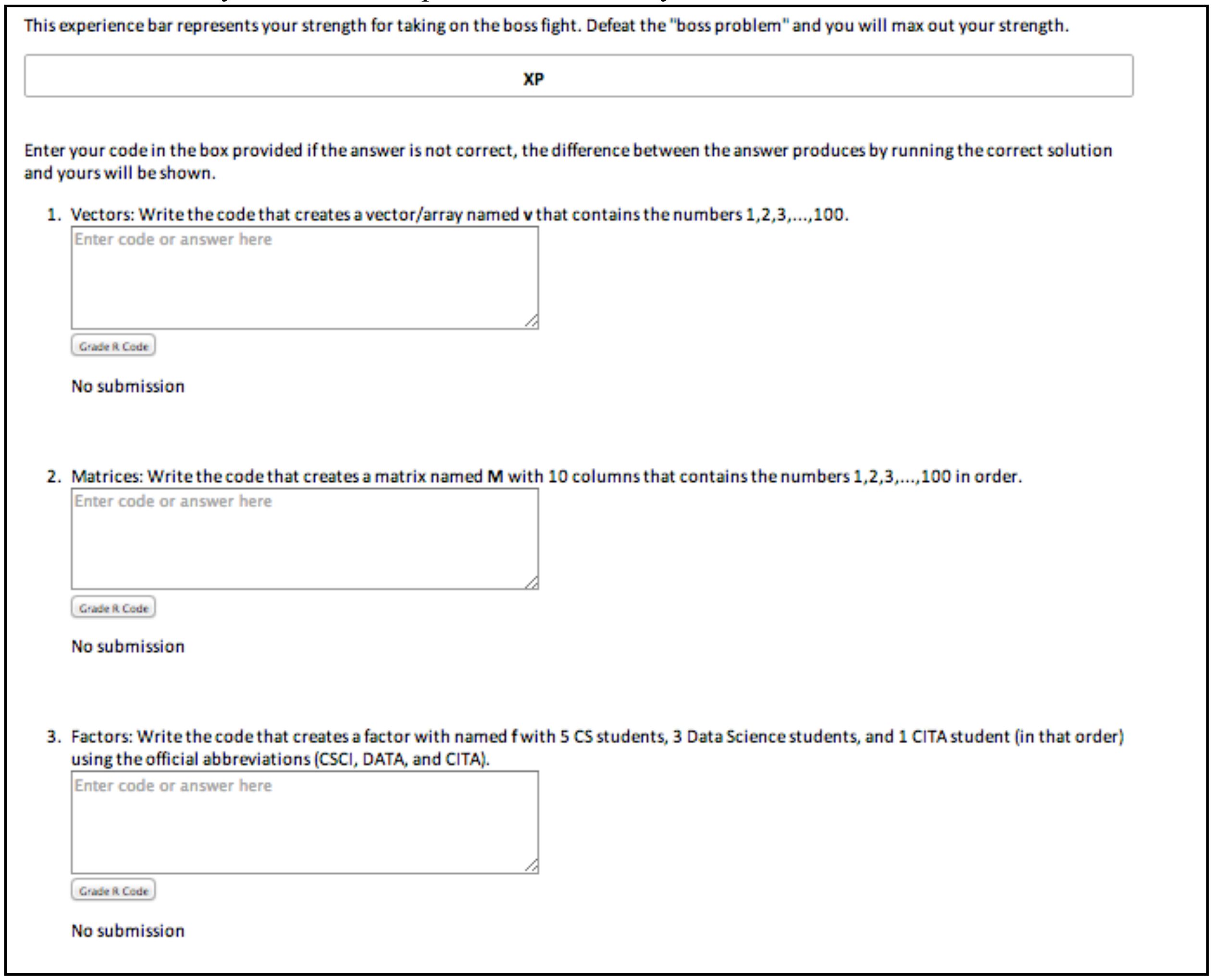
The addition of more basic lessons in R programming came about when Learn2Mine was introduced to the Data Science 101 class in the fall of 2013. These students piloted the program, which led to the creation of several new lessons for their purposes. These students were coming at R from an area of little to no programming background, so the lessons had to be able to build their knowledge from the ground up. An immediate solution to this was the creation of a new tool to encompass all of the lessons tailored to the class. This tool was very simplistic, and featured only a text area, and a drop down box that contained the potential lessons they could be submitting it as. Lessons asked that students write code to solve a series of posed problems in RStudio, then copy and paste their responses into the text are for submission in the following format:

```
1)
Problem 1 solution
2)
Problem 2 solution.
```

This way, a multitude of these introductory data science and R coding lessons could be created without having to make a completely new tool for each one; which would clutter up the toolbox. However, though some level of understanding can be measured through the answers to these questions, to really test a person's programming understanding we needed a way for them to submit their code for grading directly. To do this, an R file was saved into the myTools directory, which contained code for the correct solutions to the problems. For example, one such problem asked the user to create a function to sum 3 inputs, to be called my.sum(). A file to grade this lesson (called Grade.R) contained its own function to compute this called solution.my.sum(). When the user passes in their my.sum() function in, it is imported into the solution file, and both the my.sum() and solution.my.sum() are read with the same inputs. The Grade.R file then checks that the return value of both functions are equivalent; if this is the case, the solution is marked correct, and if all solutions for that lesson are correct the lesson is considered "Mastered".

This redesign of the manner in which lessons are created made things significantly easier for both the user to engage in and the developer to create new content for Learn2Mine. Condensing it all into one tool made the interface much cleaner, and by plugging the Submit Data tool's code into this Grade R Code tool allowed for the removal of the extra step of using the Submit Data tool on it's own. The text area input also greatly diversifies the types of inputs students can give, and therefore of questions that can be asked of them. However, though an improvement this system was still not as effective as it could be. The self-separating of answers was not an intuitive system for newcomers, requiring direct and frequent explanation, as well as granting the potential for a large number of syntax errors or discrepancies that could claim incorrect results due to ill-formatted responses despite the user performing the algorithms correctly. Additionally, the back-and-forth navigation between the Google App Engine and Galaxy faces of Learn2Mine became cumbersome, as users would have to log in to 2 or 3 different systems just to complete a lesson. Therefore the user's experience was once again re-evaluated and the site was redesigned to its current state.

The goal became to make the Learn2Mine experience as singular as possible. This was achieved by using the publicly available Galaxy Application Programming Interface (API). Rather than having the users link to Galaxy directly, and interact with that interface, all of their activity could be confined to the Google App Engine front by use of the methods defined in the Galaxy API. Now, instead of simply explanations of the lessons they are to go and accomplish elsewhere, the page for a specific lesson includes an individual TextArea for each question that is poses. Users then have clearly defined places to respond to each question. Figure 10 shows an example of a page that utilizes this layout, the Basic R lesson. The Basic R lesson contains 10 different questions, the first 3 of which are shown in the figure. Once they have an answer for a question and have entered it into the appropriate TextArea, they may click a button beneath the TextArea to submit that answer for grading. At this point, a JavaScript function is fired off, which collects the answer and some necessary values (keys in the form of strings known as workflow ids to Galaxy) are delivered via a POST request to the /OnSiteGrader link on Learn2Mine. This then tosses the information to a post() function in the class GradingHandler in the main.py of the Google App Engine. This main.py file imports the galaxy_api library, which gives it access to a function called workflow_execute_parameters(), this function allows the remote execution of a galaxy tool, and retrieval of its response. By passing this function the code and email of the user, as well as some of the aforementioned specific key values, Galaxy has everything it needs to run its grader code to evaluate the user's response. The process of executing a Galaxy tool can often take a few seconds, so a refresh button below the TextArea allows them to keep requesting their results until they are ready. The user is then notified of whether or not they answered the question successfully.


This experience bar represents your strength for taking on the boss fight. Defeat the "boss problem" and you will max out your strength.

Enter your code in the box provided if the answer is not correct, the difference between the answer produces by running the correct solution and yours will be shown.

1. Vectors: Write the code that creates a vector/array named **v** that contains the numbers 1,2,3,...,100.

No submission

2. Matrices: Write the code that creates a matrix named **M** with 10 columns that contains the numbers 1,2,3,...,100 in order.

No submission

3. Factors: Write the code that creates a factor with named f with 5 CS students, 3 Data Science students, and 1 CITA student (in that order) using the official abbreviations (CSCI, DATA, and CITA).

No submission

Figure 9: Basic R Lesson

This ability to separate questions also set the groundwork for a progress bar on each lesson page. This progress bar is built according to the number of questions the page poses, and fills according to how many have been answered correctly. Once the all questions have been completed and the progress bar fills, the user is presented with an image of the badge they have just earned. This new system is much more dynamic and friendly to those unfamiliar with the Learn2Mine system, or those who are not being taught in a classroom setting. Additionally the progress bar and displaying of the badge are further areas Learn2Mine is able to implement gamification to enhance the learning experience. Completion of individual questions filling the progress bar offer further opportunities for feedback, serving as little "checkpoints" of their progression through the lesson. The immediate displaying of the badge image is a similarly rewarding additional point of feedback.

III. Results

Learn2Mine was piloted in the College of Charleston's Data Science 101 class in the fall of 2013. Throughout the semester, students interacted regularly with the website both inside and out of class on a regular basis in order to complete assignments that counted toward their grade. At the end of the semester, students of the class were asked to complete an anonymous survey posing a series of statements meant to reflect their experiences with Learn2Mine and their opinions of it. These statements were to be responded to on a 1-5 scale, 1 meaning "Strongly Disagree", 2 meaning "Disagree", 3 meaning "Neutral", 4 meaning "Agree", and 5 meaning "Strong Agree" and were as follows:

1. The automatic feedback on correctness increased my motivation to complete the homework assignments.
2. The visual representation of accumulating badges and gaining skills moved the focus on learning rather than obtaining high marks from the instructor.
3. The ability to retry assignments improved my ability to learn and understand the material.
4. A student said "You have managed to create something that is so frustrating, yet so rewarding when finished. There is no greater feeling than getting the badge."
5. I would consider continuing to use Learn2Mine in other courses or in future studies of data science.

Table 1: Results of Anonymous Survey of Pilot Class

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	Median	Mean
Statement 1	2	0	3	17	10	4.11	4
Statement 2	1	6	7	12	7	3.6	4
Statement 3	0	0	5	10	18	4.4	5
Statement 4	2	4	3	14	10	3.89	4
Statement 5	2	4	3	15	9	3.8	4

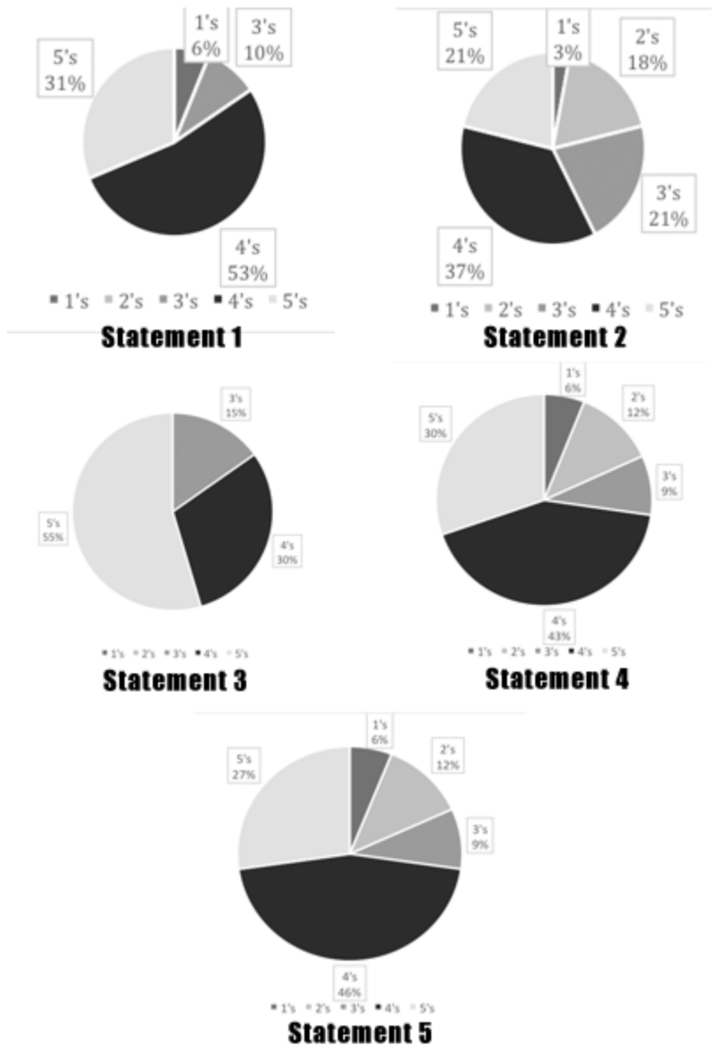


Figure 10: Results from each statement

Table 1 shows the number of observations of each response for each of the 5 statements, as well as the mean and median of the 1-5 values for each of those statements. As the Figure 11 and Table 1 show, attitude toward Learn2Mine was generally positive. Both mean and median for all statements was above the "Neutral" mark of 3, with most resting around the "Agree" at 4. The least-agreeable statement was #2, concerning whether or not the badges shifted the focus to the actual educational process rather than grades. When given in a classroom setting like this, where ultimately grades are an important factor this outcome makes sense, and it was striking that this statement was even able to get the generally positive response that it did. The two questions that got the best response, each with a mean falling between the "Agree" and "Strongly Agree" fields were questions numbers 1 and 3, both of which dealt with the instantaneous nature of grading through Learn2Mine. Students tended to greatly enjoy both the automatic feedback that could help them understand their mistakes shortly after making them (as opposed to the system of waiting several days that traditional grading often employs) and the ability to resubmit their work multiple times. This forgiving submission system allowed them to take greater risks with their practice of the lessons, and to learn through trial-and-error. This survey also allowed the students to lend any general comments they had about the system. A common problem this pilot class commented on as far as the interface is concerned was the multiple logins between the three parts of the system ("too many steps to submit", "having to go to Galaxy, Learn2Mine, and RStudio was a nuisance") a problem that has since been addressed with the updated integrated Lessons.

IV. Future Work

Learn2Mine is a project that will continue to grow and evolve over time, ever seeking to be the go-to source for Data Science education on the web. The next big step for the project will be exposing it to a community of potential developers by making it open source. By making the source code free and accessible to all, it is the hope that Learn2Mine will undergo rapid improvement as newly interested parties with different ideas and different programming capabilities able lend a cooperative hand to its development. Moving the project to this state will also allow people and organizations to not only expand its main code base, but also use the code to expand it for their own specialized purposes, similar to what we have done with the Galaxy Project's open source code. Through this specialization Learn2Mine could have the potential to take a variety of different forms in order to reach all manners of domains.

Getting the project to this point will require a few different adjustments. First is the addition of documentation. Comments are plentiful throughout the Learn2Mine code base in order to explain statements as they happen, however it lacks any wider-view explanations of its workings on a grander scheme. Instructions for how to develop code for Google App Engine according to our trends, and importantly how to add lessons to Galaxy will significantly cut down on the time it takes for any developers to get started doing meaningful work fixing and expanding the project. On top of this addition, a few changes will need to be made to Learn2Mine's existing code base for it to become generalized enough for the world at large to potentially contribute. Specifically, there are several places in the code base where passwords of the learn2mine-server virtual machine are hard-coded in, in order to automate things like user creation. These will need to be scrubbed and replaced with placeholders, which those who plan to host their own servers can replace with their personal passwords. This, too, will need to be explained in documentation.

V. Acknowledgements

Learn2Mine was the brainchild of Dr. Paul Anderson, head of the Data Science department at the College of Charleston. The project was developed under his guidance by Clayton Turner and myself; both of us undergraduate students at the college. Clayton was responsible for the back-end development of the application's processes, while my own development work dealt with the front-end interface and its integration. On behalf of all of us on the Learn2Mine team we would like to thank the Experimental Program to Stimulate Competitive Research, Institutional Development Award (EPSCoR/IDE) for funding the beginning of the project, and the College of Charleston for the use of the facilities.

VI. References

Paul Anderson. Data Science Program. http://datascience.cofc.edu/.

Shiomo Argamon. Data Science @ IIT.http://www.iit.edu/csl/cs/ programs/undergrad/datascience undergrad.shtml.

Naim Asaj, K Bastian, Mark Poguntke, Florian Schaub, Michael Weber,Daniel Ritter, Dominik Deuter, and Fabian Groh. Gamification: Stateof the Art Definition and Utilization. In Research Trends in MediaInformatics, pages 39–45, Ulm, Germany, 2012.

Daniel Blankenberg, Gregory Von Kuster, Nathaniel Coraor, GuruprasadAnanda, Ross Lazarus, Mary Mangan, Anton Nekrutenko, and JamesTaylor. Galaxy: A web-based genome analysis tool for experimentalists.Current protocols in molecular biology edited by Frederick M Ausubelet al, Chapter 19, 2001.

Coye Cheshire. UC Berkeley School of Information: Data Science.http://www.ischool.berkeley.edu/research/datascience.

Sebastian Deterding, Miguel Sicart, Lennart Nacke, Kenton O'Hara, and Dan Dixon. Gamification. using game-design elements in non-gamingcontexts. In Proceedings of the 2011 annual conference extendedabstracts on Human factors in computing systems - CHI EA '11, pages2425–2428, New York, New York, USA, 2011. ACM Press.

Nadya Direkova. Game On: 16 Game Mechanics for User Engagement,2012. http://www.slideshare.net/gzicherm/nadya-direkova-game-on-16-design-patterns-for-user-engagement.

A. Dom´ınguez, J. Saenz-de Navarrete, L. De-Marcos, L. Fern´andez-Sanz, C. Pag´es, and J.J. Mart´ınez-Herr´aiz. Gamifying learning experiences: Practical implications and outcomes. Computers & Education,(63):380–392, 2012.

Philip L. Dubois. UNC Charlotte: Data Science and Business AnalyticsInitiative. https://dsba.uncc.edu/academic-programs.

M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, andI. Witten. The WEKA Data Mining Software. SIGKDD Explorations,11(1):10–18, 2009.

Hannah Gerber. Can Education be Gamified?: Examining Gamification, Education, and the Future.http://shsu.academia.edu/HannahGerber/White-Papers, 2012.

Belinda Giardine, Cathy Riemer, Ross C Hardison, Richard Burhans,Laura Elnitski, Prachi Shah, Yi Zhang, Daniel Blankenberg, Istvan Albert,James Taylor, Webb Miller, W James Kent, and Anton Nekrutenko.Galaxy: A platform for interactive large-scale genome analysis. GenomeResearch, 15:1451–1455, 2005.

Jeremy Goecks, Anton Nekrutenko, and James Taylor. Galaxy: acomprehensive approach for supporting accessible, reproducible, andtransparent computational research in the life sciences. GenomeBiology, 11:R86, 2010.

Ian Glover. Play As You Learn : Gamification as a Technique forMotivating Learners. http://www.editlib.org/p/112246?nl, 2003.

Karl Kapp. The Gamification of Learning and Instruction. Pfeiffer, SanFrancisco, CA, 2012.

Ingo Mierswa, Michael Wurst, Ralf Klinkenberg, Martin Scholz, andTimm Euler. YALE: Rapid Prototyping for Complex Data MiningTasks. In Proceedings of the 12th ACM SIGKDD international conferenceon Knowledge discovery and data mining, volume 2006, pages935–940, 2006.

"Mozilla Open Badges." About. Mozilla, n.d. Web. 23 Apr. 2014.

Cristina Ioana Muntean. Raising engagement in e-learning throughgamification. In The 6th International Conference on Virtual LearningICVL 2011, pages 323–329, 2011.

Fiona Fui-hoon Nah, Venkata Rajasekhar Telaprolu, Shashank Rallapalli, and Pavani Venkata. Gamification of Education Using ComputerGames Background : Gamification and Its Application toEducation. http://link.springer.com/chapter/10.1007/978-3-642-39226-9 12#page-1, 2013.

"OAuth." Community Site. Oauth, n.d. Web. 23 Apr. 2014.

Ryan Swanstrom. Colleges With Data Science Degrees. Wordpress. April 9, 2012. Accessed: April 23, 2014.

Jennifer Thom, David R Millen, Joan Dimicco, and Rogers Street.Removing Gamification from an Enterprise SNS. In Incentives, pages1067–1070, Seattle, WA, 2012.

Nikolay Vyahhi, Phillip Compeau, Andrey Balandin, Aleksey Kladov,Ekaterina Sosa, Mikhail Dvorkin, Sonya Alexandrova, and Mike Rayko.Rosalind. http://rosalind.info/.

Katherine Wolstencroft, Robert Haines, Donal Fellows, Alan Williams,DavidWithers, Stuart Owne, Stian Soiland-Reyes, Ian Dunlop, AleksandraNenadic, Paul Fisher, Jiten Bhagat, Khalid Belhajjame, Finn Bacall,Alex Hardisty, Abraham Nieva de la Hidalga, Marie P. Blacazar Vargas,Shoaib Sufi, and Carole Goble. The Taverna workflow suite: designingand executing workflows of Web Services on the desktop, web or inthe cloud. Nucleic Acids Research.