

January 2018

Teaching With Jupyter In-Class Activities: Lessons Learned and Next Steps

David Anastasiu
San Jose State University, danastasiu@scu.edu

Follow this and additional works at: https://scholarworks.sjsu.edu/computer_eng_pub



Part of the [Digital Communications and Networking Commons](#)

Recommended Citation

David Anastasiu. "Teaching With Jupyter In-Class Activities: Lessons Learned and Next Steps" *The 20th CSU Symposium on University Teaching* (2018).

This Presentation is brought to you for free and open access by the Computer Engineering at SJSU ScholarWorks. It has been accepted for inclusion in Faculty Publications by an authorized administrator of SJSU ScholarWorks. For more information, please contact scholarworks@sjsu.edu.



What is Jupyter Notebook?

- Interactive browser-based document that enables mixing **rich text** with **mathematical equations**, live **data visualizations**, and interactive **execution of code**.
- Popular with students and professionals alike from fields as diverse as Data Science, Sociology, Political Science, Physics, and Journalism.
- Supports more than 40 programming languages:



- The benefits of using notebooks include:
 - ✓ Interactivity.
 - ✓ Analytical reproducibility.
 - ✓ Collaboration.
 - ✓ Ease of access to computing resources.
- Used by dozens of major companies and part of curriculum at many universities.



Active Learning using Jupyter Notebook

In-Class Activities

- Designed to aid presentation of theoretical concepts, helping students learn through practice.
- Plan to include 1-2 Activities per class.
- Beginning of notebook introduces topic and ties in with lecture.
- Description of concepts is intermingled with demonstrations and short practical exercises.
- Students work in groups and are given 5-10 minutes to complete exercises.
- Each exercise is followed with in-class discussion analyzing proposed solutions by students in the class.

Take-Home Activities

- Some notebooks are assigned as homework assignments and provide additional opportunity for practice.
- Activities are not graded on correctness. Students are encouraged to work through activity problems.

Example Jupyter Notebook Activities

CMPE 139: Database Systems I

```
File Edit View Insert Cell Kernel Widgets Help
Trusted Python (default)

Exercise 1: Optimizing the IO Cost
Can you find a logically equivalent form that uses fewer total reads?

In [9]: 1 y = RJoin(Project(["R"], R), Project(["R"], S))
2 render_markdown()
3 print_compare_results(s,y)
4 cost_markdown()

(I10(R(A,B)) & M4(I10(S(B,C)))

True
Total Reads: 180

• M4 tuples read in: 30 out: 5
• I10 tuples read in: 25 out: 5
• R(A,B) has 25 tuples
• I10 tuples read in: 125 out: 25
• S(B,C) has 25 tuples

Exercise 2: Comparing costs
Let's explore how the cost of the two expressions compare as the data size increases, and with different numbers of distinct values in the data. Given that R has N values, S has M values, you can assume that N = M to simplify, and that they will both have the same number of unique B values.
You can do this any way you choose, but we outline one way below.
We'll start with a function for each cost that will take as input:
• The number of tuples in R, N
• The number of distinct B values in R, N_B
• The number of tuples in R M4, S, O1
• The number of tuples in I10(R M4, S, O2)
Your function should return the total number of reads as in the cost_markdown function

In [10]: 1 def cost_simple_n3(n, m):
2
3     Cost to perform a simple M4 join
4     Assuming 1 tuple / page
5
6     return n + m*n
```

```
File Edit View Insert Cell Kernel Widgets Help
Trusted Python (default)

Activity 3-2
The goal for this activity will be to compute some BCNF decompositions, using the tools from last lecture.
First we'll load those tools, and some sample data.

In [1]: 1 from closure import compute_closure, display_side_by_side, print_setup
2
3 load_ext sql
4 sql setup

Out[2]: 'Connected: None@None'

In [3]: 1 WAG DROP TABLE IF EXISTS T;
2 CREATE TABLE T(Course VARCHAR(255), Classroom INT, Time INT);
3 INSERT INTO T VALUES ('CS 304', 122, 900);
4 INSERT INTO T VALUES ('CS 304', 122, 900);
5 INSERT INTO T VALUES ('EE 101', 218, 900);

Done.
1 rows affected.
1 rows affected.
1 rows affected.

Out[3]: []

Exercise 1
First, let's decompose T into BCNF. Explicitly go through the steps of the BCNF algorithm using the compute_closure function, then decompose the following table (i.e. by creating new SQL tables) into BCNF.
We've also made a function, display_side_by_side, for nicer display!
```

CMPE 255: Data Mining

```
File Edit View Insert Cell Kernel Widgets Help
Trusted Python (default)

Another way to rank pages
The ranking approaches just discussed are easily exploited.
• With n degree ranking, someone might set up a thousand pages that link to one page, thereby inflating its rank.
• With out-degree ranking, someone might include a mountain of links on every page they make to inflate its rank.
We know there is a deeper meaning in our internet graph. Some pages are more important than others. Degree ranking misses a key component of website importance: the number of sites linking to a particular site matters less than the importance of those sites.
Our third approach, pagerank, captures this underlying meaning by focusing on probability rather than degree. If one visits site A, what is the probability they will visit site B next? We can infer these probabilities from the adjacency matrix of the graph. Given these probabilities, we can find the page rank of each node by an iterative procedure.

PR(p_i) = \alpha \sum_{p_j \in \text{In}(p_i)} \left( PR^{(t-1)}(p_j) \times \frac{1}{|\text{Out}(p_j)|} \right) + (1 - \alpha) p_i

where n is the number of nodes in the graph, PR^{(t)}(p_i) and PR^{(t-1)}(p_i) are the current and last iteration pagerank scores for node p_i, In(p_i) and Out(p_i) are the in- and out-degrees of the node, respectively, (1 - \alpha) is a damping factor with 0 < \alpha < 1, and p_i is the teleportation probability (i.e. the probability of jumping to a random node instead of following an outgoing link, usually 1/n). Generally, \alpha is close to 1, e.g. 0.8 or 0.9. Note that \frac{1}{|\text{Out}(p_j)|} is in fact the probability of transition from p_j to p_i.
If we put all pagerank scores in a vector, we can write the above formula as:
p^t = \alpha M p^{t-1} + (1 - \alpha) 1/n,
where p^t and p^{t-1} are the pagerank vectors from the current and previous iterations, M is the transition probability matrix derived from the graph adjacency matrix, and 1 is a vector of all 1's of the appropriate size. We set the initial transition probabilities for all nodes to 1/n, i.e. p^0 = 1/n.
Given enough iterations, the series converges to the leading eigenvector of the matrix:
\alpha M + (1 - \alpha) 1/n.
```

```
File Edit View Insert Cell Kernel Widgets Help
Trusted Python (default)

Percentage of variance explained for each component

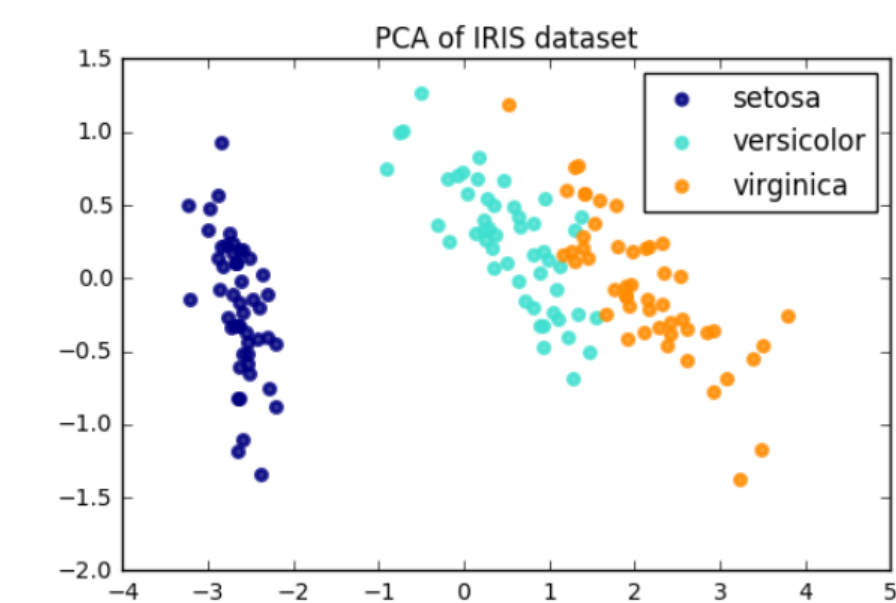
In [4]: 1 print("Explained variance ratio (first two components):")
2 % string explained_variance_ratio

Explained variance ratio (first two components): [ 0.92461621  0.05301557]

In [5]: 1 plt.figure()
2 colors = ['navy', 'turquoise', 'darkorange']
3 linewidth = 2

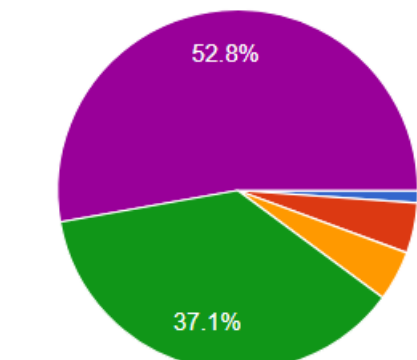
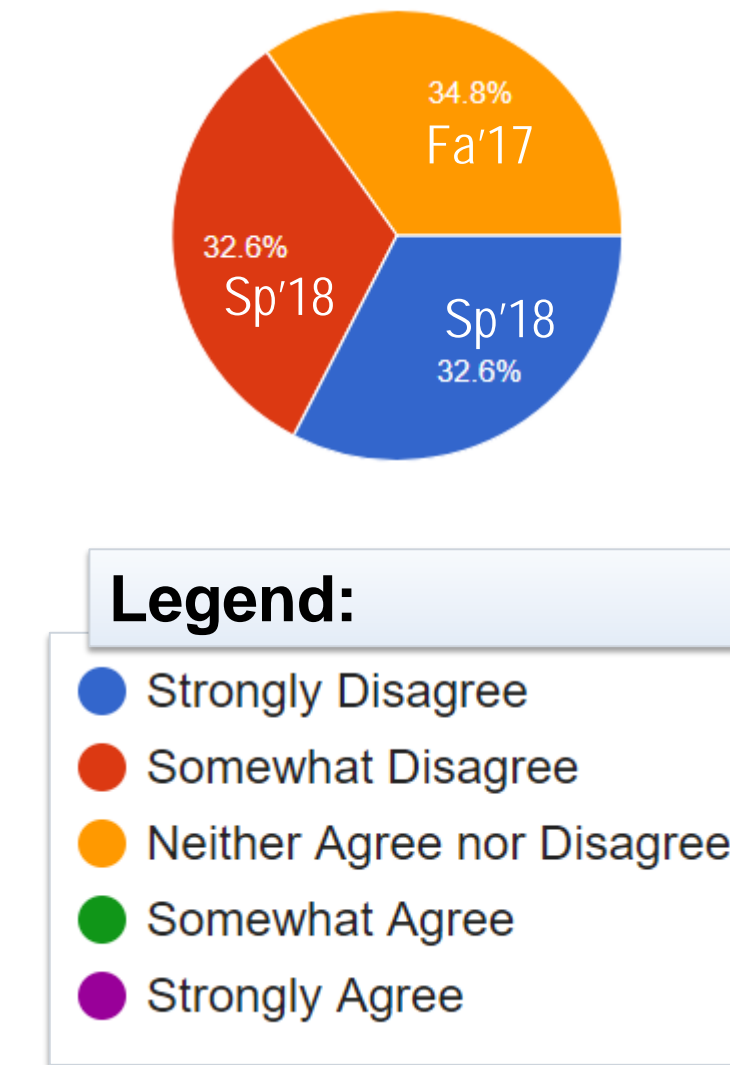
Scatter Plots for PCA and LDA

In [6]: 1 for color, i, target_name in zip(colors, [0, 1, 2], target_names):
2     plt.scatter(X[:, 0] == i, X[:, 1] == i, color=color, alpha=.8, linewidth=2,
3               label=target_name)
4     plt.legend(loc='best', shadow=False, scatterpoints=1)
5     plt.title('PCA of IRIS dataset')
6
7 plt.figure()
8 for color, i, target_name in zip(colors, [0, 1, 2], target_names):
9     plt.scatter(X[:, 0] == i, X[:, 1] == i, color=color,
10               label=target_name)
11 plt.legend(loc='best', shadow=False, scatterpoints=1)
12 plt.title('LDA of IRIS dataset')
13
14 plt.show()
```



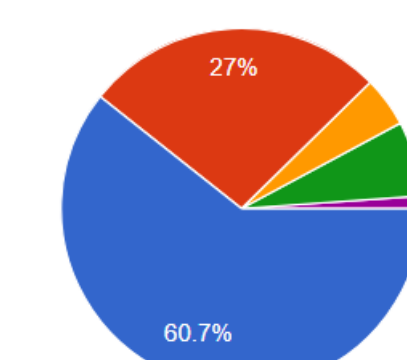
Lessons Learned from Student Feedback

- End of semester survey given to students in 3 sections of CMPE 255 (Data Mining), over 2 semesters.
- Likert-scale survey with 10 categorical and 4 open answer questions.
- Avoids response style bias by testing both positive and negative responses.
- Positive questions coded 1-5 and negative ones 5-1.
- 89 responses, evenly distributed between sections.

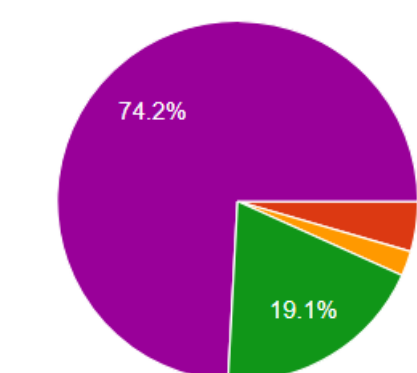


I understood material taught in class better because of the Jupyter Notebook activities

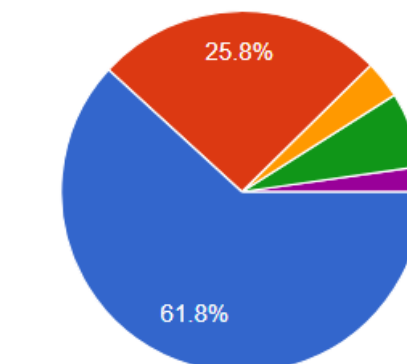
The Jupyter Notebook activities did not help clarify concepts in the lectures.



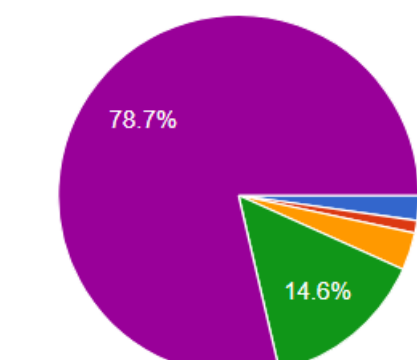
It was helpful to have rich text descriptions of problems and concepts in the same page I was coding in.



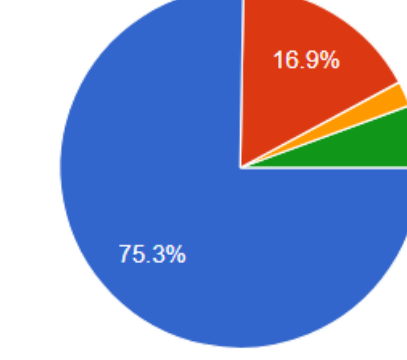
I found the ability to intersperse textual information with executable code in Jupyter Notebooks odd and useless.



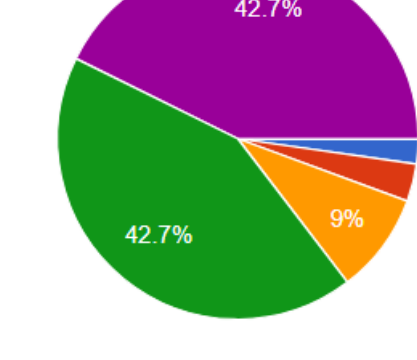
Once installed, I found the Jupyter notebooks easy to use and intuitive.



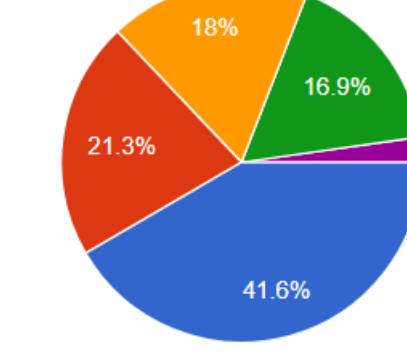
I found it difficult to execute programs in the Jupyter Notebook environment.



Activities had a good degree of difficulty that kept me engaged.



The activities were too difficult and I could not finish them even if I spent the whole class time on them.

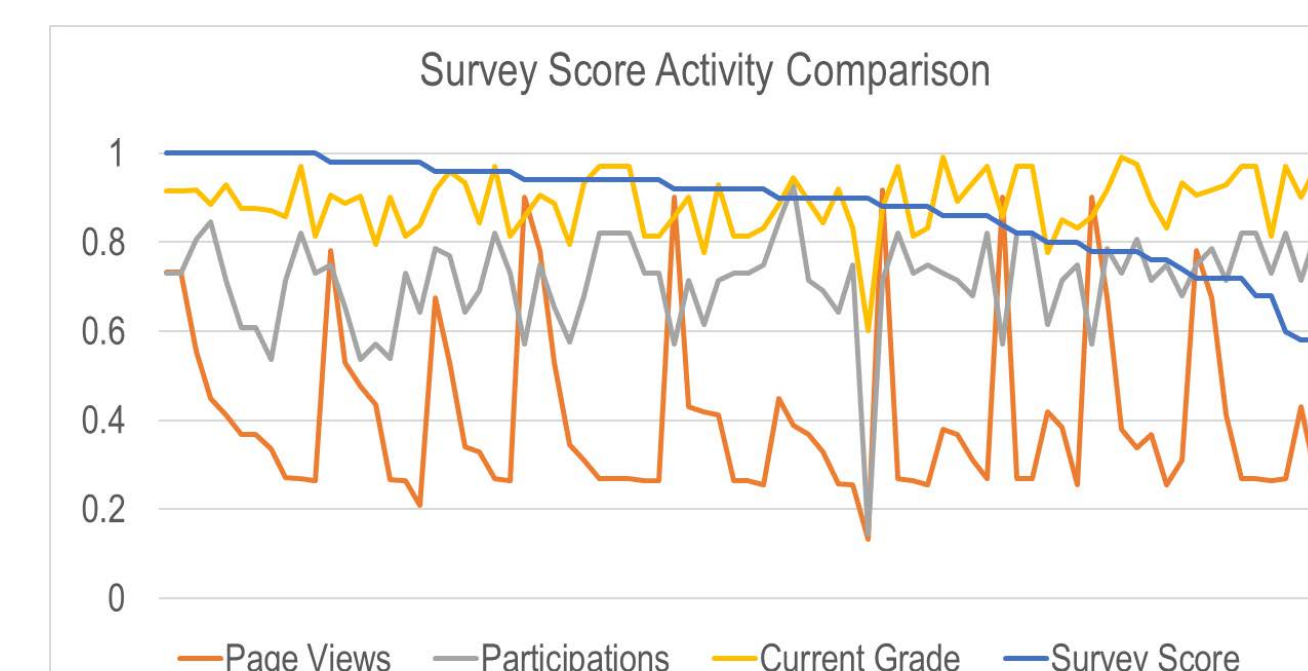


- Results primarily positive: 94% over 3.5 and 78% over 4.0.
- Results not found to be correlated with student in-class activity (measured by Canvas page views and interactions) or current grade.

$$r_{\text{views}} = -0.069$$

$$r_{\text{part}} = -0.133$$

$$r_{\text{grade}} = -0.171$$



Open Answer Questions and Example Answers

What were the most useful features of the Jupyter Notebook activities? Why?

- It helped us learn concepts better. Was a very good learning tool and very easy to use.*
- [Executing] pieces of programs in real time [helps] break down complex material into understandable chunks.*
- Concept followed by activity. Faster learning.*
- It helped me understand the methods and algorithms mentioned in the slides in a practical way.*

What were the downsides of programming in Jupyter Notebooks? Why?

- I can not recall any downside.*
- Syntax highlighting or syntax help like in other editors.*

How, if at all, did you approach solving homework assignments for the class (or even other classes) in a different way after being exposed to Jupyter Notebook?

- Running step by step programs to make sure each part works well.*
- [Getting] into the habit of writing descriptions along with the program.*
- I would dissect the problems into a set of small problems, implement each of them instead of trying to solve the big problem as a whole.*

What is one thing that could be improved in the use of Jupyter Notebook and/or in-class activities for this class?

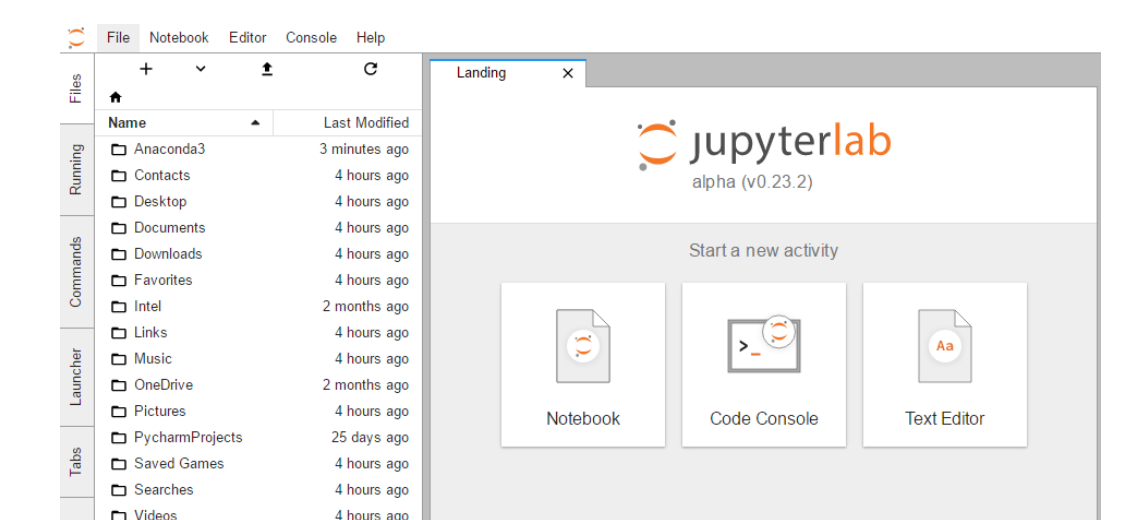
- Should be more in-class with a little more time.*
- Have a complete solution posted after the in-class activities are due.*

Next Steps: Jupyter Hub, HPC, and JupyterLab

- Alleviate initial setup troubles + ensure identical setup.



- JupyterLab will soon replace Jupyter Notebook.
- Continue to add and improve activities.



Acknowledgements and Links

All logos are property of their respective companies and registered products.

- [1] Project Jupyter: <https://jupyter.org>
- [2] Jupyter Lab: <https://github.com/jupyterlab/jupyterlab>
- [3] Teaching with Jupyter (Google Group): <https://groups.google.com/d/forum/jupyter-education>
- [4] JupyterHub for Teaching: <http://jupyterhub-deploy-teaching.readthedocs.io>