Technology Review for Machine Learning for March Madness

Alex Hoffer, Chongxian Chen, and Jacob Smith
Team Name: Stat Champs

Abstract

Students of Biochemistry and Biophysics at Oregon State University need to have the opportunity to learn machine learning algorithms that will be useful in their careers. However, grasping the nuances of these algorithms is difficult when they are taught through their application to Biochemistry and Biophysics. One way that may facilitate this learning process is introducing these concepts through their application to something fun and simple, like college basketball statistics. This project will develop an online module where these students can select a machine learning algorithm, choose which college basketball statistics they want to train this algorithm with, and generate a NCAA March Madness bracket.

I. Introduction

THERE are many important technological considerations when developing this instructional tool. The page must be reasonably fast, responsive, and usable, it must provide statistics for the user to choose from, and it must provide these statistics as input to machine learning algorithms. Finally, the bracket must be generated and presented to the user. Each one of these necessary components has several potential technologies that could be used satisfactorily. It is the purpose of this document to identify the possible technologies and make reasoned decisions as to what technologies we will be using. The first nine possible technologies were written by Alex Hoffer. These first nine technologies are chiefly concerned with the usability of the service, specifically the design of the graphical user interface, the instructions that will be presented to the user, and the presentation of the machine learning bracket that is generated. It is our belief that in order for the tool to be effective in educating students it must be well designed and easy to understand.
insert Chongxian and Jake's info
insert table of contents

Stat Champs
November 14, 2016

A. Technology 1: Graphical user interface of web page

1) Options 1, 2, and 3: Kivy, PyQt, PyGUI. All three options are libraries to be used with Python.

2) Goals for use in design: The page must be visually pleasing, interactive, and usable. This tool will be used to demonstrate machine learning processes to programming neophytes so the logic and design of the module has to be easy to understand to ensure that the user will not be confused.

3) Criteria being evaluated: Since there are not major concerns with security for this module because of the lack of sensitive information being used, security is not an important criterion. Cost is also not a very crucial factor because there are plenty of free technologies available that can generate visually pleasing and usable web pages. Availability is an important factor because we want the module to work the same on all major browsers, specifically Chrome, Firefox, and Internet Explorer. Speed is also a valuable consideration because we want the machine learning component of this service to be quick. Therefore, the technology that we use to design the web page itself should be reasonably economical so the page loads quickly and doesnt absorb too many resources that should be allocated for the machine learning processes. Readability is also important- we would like the technology to be easy to understand. Of similar importance as speed is economy. We want this technology to be able to do a lot with only a few lines of code.

4) Table comparing options:

Technology	Economy	Readability	Availability	Speed	Notes
Kivy	Extremely	Extremely	Cross-platform	Very fast, built on OpenGL ES 2	Focus is on complex UIs
PyQt	Extremely	Somewhat	Cross-platform	Fast, written in C++	Focus is on mobile development
PyGUI	Extremely	Extremely	Cross-platform	Not conclusive	Smallest and simplest

5) Discussion: Each of these three technologies are viable options. Since they are all Python libraries, they should be easy to learn, which is important because the GUI is the first thing to be developed in our project and the other requirements build from it. Each of the three are powerful enough for our purposes, and each have different strengths. Kivy is sophisticated and elegant, but it might be too vast for our uses. Kivy is used in complex operations like touch screen animation [1], and our project doesn't call for an extremely sophisticated GUI. In fact, our GUI should be simple and unobtrusive so as not to draw attention away from the machine learning components. PyQt's strength appears to be in mobile development [2]. Since our module doesn't have to be usable from a mobile device, it may be best to go with a technology with a focus in computer based web development. This brings us to PyGUI, which is the best of the three options for our purposes. PyGUI is the oldest of the three and was developed to be simple to implement [2]. Its focus matches with ours: we need a simple GUI, and PyGUI is the least flashy and most reasonable of these technologies.

6) Selection of best option: PyGUI is the technology that seems to most closely capture the needs of our GUI without adding fancy features that may slow our module down and make maintenance and testing difficult.

B. Technology 2: Instructions for the user to interact with the module before the module itself is presented

1) Options 1, 2, and 3: Webix, UKI, MochaUI. All three options are freely available JavaScript libraries. Use of JavaScript is important here because we want the instructions to seamlessly lead into the module itself without requiring the user to refresh the page.

2) Goals for use in design: The module must have instructions for the user to follow so they can effectively use the tool. These instructions should be easy to read and flow seamlessly into the tool. The machine learning component of this project should be the central focus, and so the instructions that we provide to the user have to be designed in such a way that they are not visually offensive or disorganized.

3) Criteria being evaluated: Security is again not particularly important due to the innocuousness of the data being transferred from user to server. Cost is not concerning because of the wide availability of open source technologies that can properly satisfy this requirement. Speed is important because we want our server's processing resources to be saved for the machine learning algorithms and we want the page to be quick. Availability is important because we want this tool to be usable on each of the major browsers. Readability and economy are also important.

4) Table comparing options:

Technology	Economy	Readability	Availability	Speed	Notes
Webix	Extremely	Very	Cross-platform	Not conclusive, no metrics	Easy to learn, 128KB
UKI	Very	Somewhat	Cross-platform	Fast (progressive rendering)	34 KB, meant for desktop web apps
Angular JS	Very	Below average	Cross-platform	Somewhat fast	Easy to start, hard to maintain

5) Discussion: Webix is a popular and reliable option. What makes it so attractive is how feature rich it is, its ease of use, and its readability. The example program listed on [3] demonstrates how economical it is, too. With only several lines of code, Webix is capable of generating a clean and usable user interface. One drawback of Webix is that it's difficult to tell how fast it is. Its documentation merely makes note that is fast, but provides no usable metrics. It also isn't as lightweight as UKI (128 KB for Webix [3] and 34 KB for UKI [4]), but it is still quite small, so its size won't be slowing down the server. UKI is also a good option and its focus appears to directly parallel ours. It is meant for desktop applications [4], which is the medium of our project. It is also quite fast due to its progressive rendering and can produce 30,000 tables almost instantaneously [4]. Angular JS seems to be the worst of these three options, as it is hard to learn [5] and its programs grow rapidly in complexity. It may be too ambitious for the requirements of our project to learn Angular JS.

6) Selection of best option: Webix is the best option because it is easy to use, reliable, and offers the features that we need without slowing down the server. UKI is another good option but is harder to use. Angular JS is not necessary for our project.

C. Technology 3: Presentation of the machine learned bracket that is generated by scikit

1) Options 1, 2, and 3: TkInter, VTK, Wax. All three options are freely available Python libraries.

2) Goals for use in design: The module is functionally useless without the presentation of the bracket that was generated by the machine learning algorithm. This technology must present the bracket in a way that is visually pleasing and easy to understand.

3) Criteria being evaluated: Cost and security are again not important for the reasons outlined earlier. Availability is crucial because we want the bracket to appear the same in all major browsers. Speed is important too because we want the user to see their bracket promptly. Readability and economy are also useful considerations.

4) Table comparing options:

Technology	Economy	Readability	Availability	Speed	Notes
TkInter	Extremely	Somewhat	Cross-platform	Slow	Most popular Python GUI library
VTK	Extremely	Extremely	Cross-platform	Fast, written in C++	Focus is on data display
Wax	Very	Somewhat	Cross-platform	Somewhat fast	Good for quick development of small project

5) Discussion: TkInter is the standard for Python GUIs. However, it isn't very fast [6] and offers many features that aren't necessary for our needs. It also isn't particularly easy to read, which has implications for both testing and maintenance. We need a framework that can accept data generated by a machine learning algorithm and then present the data in bracket form. This is, in essence, data visualization. This makes VTK a compelling option. VTK is economical (you can do a lot with a small amount of code), easy to read, very fast, and was developed with data visualization in mind [7]. Wax is the least compelling option. It is difficult to read, only somewhat fast, and appears to be in many ways flawed. [8] suggests that Wax is a good option if you want to quickly hack something together, but the presentation of the bracket is perhaps the most important part of this project, and as such, it should not be hacked together. We require an elegant solution to this problem and VTK appears to be the tool that will be most poised to fit our needs.

6) Selection of best option: VTK is the option that is most likely to give us the results we want. It is designed with data visualization in mind, which is what we need it for. The other two frameworks are flawed.

II. Conclusion

The conclusion goes here.

Appendix A
Proof of the First Zonklar Equation

Appendix one text goes here.

APPENDIX B

Appendix two text goes here.

REFERENCES

[1] Kivy: Cross-platform Python Framework for NUI, *Kivy*. [Online]. Available: https://kivy.org/. [Accessed: 14-Nov-2016].
[2] D. Bolton, 5 Top Python GUI Frameworks for 2015 - Dice Insights, *Dice*, Feb-2016. [Online]. Available: http://insights.dice.com/2014/11/26/5-top-python-guis-for-2015/. [Accessed: 14-Nov-2016].
[3] JavaScript Framework and HTML5 UI Library for Web App Development-Webix, *Webix*. [Online]. Available: https://webix.com/. [Accessed: 14-Nov-2016].
[4] UKI, *Best Web Frameworks*. [Online]. Available: http://www.bestwebframeworks.com/web-framework-review/javascript/117/uki/. [Accessed: 14-Nov-2016].
[5] J. Shore, An Unconventional Review of AngularJS, *Let's Code Javascript*, 14-Jan-2015. [Online]. Available: http://www.letscodejavascript.com/v3/blog/2015/01/angular_review. [Accessed: 14-Nov-2016].
[6] F. Lungh, Notes on Tkinter Performance, *Effbot*, 14-Jul-2002. [Online]. Available: http://effbot.org/zone/tkinter-performance.htm. [Accessed: 14-Nov-2016].
[7] VTK-Enabled Applications, *VTK*. [Online]. Available: http://www.vtk.org/. [Accessed: 14-Nov-2016].
[8] Wax GUI Toolkit, *Python*. [Online]. Available: https://wiki.python.org/moin/wax. [Accessed: 14-Nov-2016].