Final Report

Name: Alex Hoffer, Chongxian Chen, Jacob Smith

Group Number: 59

Project Title: Machine Learn Your Way to March Madness Glory!

Abstract

The application of machine learning to Biochemistry and Biophysics has enabled researchers in this eld to make remarkable discoveries, such as the generation of new DNA sequences. However, students of Biochemistry and Biophysics do not get the opportunity to learn machine learning. Dr. Victor Hsu of the Oregon State University Biochemistry and Biophysics department has commissioned the Stat Champs to produce an instructional module to give his students the chance to familiarize themselves with machine learning. The software product the Stats Champs have agreed to develop is a web page that allows students to train a machine learning model based on the college basketball statistics and machine learning algorithm of their choosing in order to produce a March Madness bracket. This will help students understand how machine learning algorithms produce models and how inclusion or exclusion of certain data can inuence such models. Over the course of Fall term 2016, the Stat Champs developed materials such as design documents and technology reviews in order to prepare for the engineering of the module. Then, in Winter term 2017, the Stat Champs began the software development phase of this project. In Spring term 2017, the project was finished. This report comprehensively describes the lifecycle of producing the project.

CONTENTS

1 Introduction

Biochemistry and Biophysics are two fields that are ripe with many exciting breakthroughs. Machine learning is used by these disciplines to aid their research by providing the ability to generate new sequences of DNA. Our client, Dr. Victor Hsu, is a professor in the Biochemistry and Biophysics department at OSU who recognized that his students should understand machine learning in order to prepare them for their careers. He noticed that the department does not encourage learning machine learning, and that even if they were to, machine learning is a difficult topic for people who haven't been trained in computer science. Additionally, learning machine learning through its application to these fields can be confusing, since machine learned models of DNA can be tough to interpret. Therefore, he commissioned us to produce an online instructional module where his students could grasp machine learning fundamentals in a fun and straightforward manner. To achieve this, he asked us to provide an interface where users could select from men's college basketball statistics and generate a March Madness bracket. In doing so, budding scientists would be able to familiarize themselves with basic machine learning algorithms and also witness how the inclusion or exclusion of data influences the models trained on them. Alex Hoffer developed the GUI, Chongxian Chen implemented machine learning algorithms, and Jacob Smith gathered data for use. Dr. Hsu's role in the project was supervision only. This report chronicles the lifecycle of the project through Fall, Winter, and Spring terms in 2017.

2 Original Requirements Document

Oregon State University Computer Science Senior Design 2016

Requirements Document

By Alex Hoffer, Jake Smith, and Chen Chongxian

Team Name: Stat Champs

Abstract

The Biochemistry and Biophysics department at Oregon State University will have a server that allows for students to select statistical categories from college basketball games and generate March Madness brackets from these selections.

Table of Contents

1 Introduction

1.1 Purpose

The purpose of this software requirements specification is to outline what is necessary for this software product to include. That is, this document will describe what inputs the software will expect and utilize and what ways it will transform these inputs into meaningful outputs. The intended audience for this document is Dr. Victor Hsu of the Oregon State Biochemistry and Biophysics department.

1.2 Scope

The Biochemistry and Biophysics department at Oregon State University needs to train their students on basic machine learning concepts. This is because machine learning is a highly useful tool in Biochemistry and Biophysics. For example, finding and understanding biologically functional DNA sequences is made possible by machine learning. However, machine learning is not a cornerstone of these students' education. There needs to be an instructional tool that provides these students with the opportunity to understand basic machine learning principles. Grasping machine learning concepts from its applications to biochemistry/biophysics is difficult because these models are generally hard to interpret. Therefore, this tool should utilize results that are simple to interpret in order for students to understand how training data on certain statistics can either damage or improve the accuracy of their generated model. The use of March Madness brackets makes this possible because the output of the students' effort will be straightforward: a team can either win or lose in each round, and whichever basketball statistics the user chose to train their data on will be reflected in how far each team goes in their model.

1.3 Definitions, acronyms, and abbreviations

NCAA: National Collegiate Athletic Association, the organization that holds the March Madness tournament each year.
Scikit: A machine learning API supported by Python that provides us with the algorithms we will allow users to select from to use to generate brackets. The user guide for this API can be found in the References section at (1).
Webix: A graphical user interface API supported by JavaScript that provides us with the ability to present instructions and the generated brackets in a way that is appealing to the user. The user guide for this API can be found in the References section at (2).

1.4 References

[1] "User guide: Contents - scikit-learn 0.18 documentation," in *Scikit-Learn*, 2010. [Online].
Available: `http://scikit-learn.org/stable/user_guide.html`. Accessed: Nov. 8, 2016.
[2] X.S. Ltd, "Guides Webix Docs" [Online].
Available: `http://docs.webix.com/desktop_basic_task.html`. Accessed: Nov. 8, 2016.

1.5 Overview

The remaining contents of this document will consist of more specific descriptions of the software itself such as the product perspective, the product functions, user characteristics, constraints, assumptions/dependencies, as well as technical challenges and functional requirements. There will also be a Gantt chart to outline our project's lifespan.

2 Overall Description

2.1 Product Perspective

This product is independent and completely self-contained. As of right now, there are no plans to implement a larger product which this product must interact with. There are various APIs the product must interact with in order to function, and these are outlined in other sections of this document.

2.2 Product Functions

Without logging into a specially created account, the user will be able to access a web page where they can select a) basketball statistics and b) a machine learning algorithm. They will then be presented with a March Madness bracket that reflects the machine learning algorithm they chose that has been trained on the basketball statistics they chose. The user can run this program as many times as they like to generate more brackets.

2.3 User Characteristics

We do not place any restrictions on who can access this web page. Anyone with the link can utilize our service. However, this product is designed specifically for undergraduate or graduate students of Biochemistry and Biophysics at Oregon State University. It will be designed with this in mind by allowing for them to select machine learning algorithms which may be useful when applied to Biochemistry and Biophysics.

2.4 Constraints

This product will be using several libraries and languages. Thus, we must be aware of the limitations of these technologies. We believe that webix will be sufficient for developing a user-friendly experience, and scikit will be sufficient for implementing the machine learning algorithms we allow the user to select from. However, there are considerations, such as how scikit will communicate its results to JavaScript, that we must develop solutions for before the module can be fully functional. A more in-depth exploration of these technologies and the constraints they may impose can be found in our technology review.

2.5 Assumptions and dependencies

The browser that the user is viewing the web page with must have JavaScript enabled.

2.6 Technical Challenges/Issues

We would like this instructional tool to be hosted online. This means that we must consider the different browsers students may use. We want our tool to be compatible with Internet Explorer, Google Chrome, and Mozilla Firefox. We must be careful in developing our GUI so that it appears the same in each of these browsers. To ensure the appearance of the module will be identical between these three options, our GUI will be written in the JavaScript library webix, which provides support for these three browsers. Another technical challenge we must solve is allowing users to select from a myriad of statistical categories and transferring this input to our machine learning submodule. The statistical categories we will be including are individual player statistics such as points, rebounds, and assists per game, as well as team statistics, such as record against their opponent and points, rebounds, and assists per game. These statistics will be held in a .csv file and will be accessed using Python. We also must be sure that our machine learning submodule will function properly in each of these three browsers. That is, users should be able to select from which machine learning algorithm they want to use after selecting their statistical categories and the output of such a decision should accurately reflect both of these factors. We will include a number of machine learning algorithms for the user to choose from, such as generalized linear models, linear and quadratic discriminant analyses, and support vector machines. The description for how these algorithms are implemented can be found in the scikit documentation [1]. In order to achieve this, we will use the Python scikit library, which is stable for our purposes.

3 Specific requirements

3.1 Functional Requirements

1. Users start out by seeing instructions on how to use the tool presented to them using the JavaScript library webix.
2. Users then see a compilation of basketball statistics consisting of the categories such as points, assists, and rebounds per game (for each player and team) and select which statistics they want to train their model on.
3. The user is asked which machine learning algorithm out of a set including supervised and unsupervised learning algorithms we provide that they want to use.
4. The statistics are passed to the corresponding .machine learning algorithm which is computed using scikit. 5. The resulting bracket is generated and presented to the user.
5. The user is given an option to create a new bracket.

3.2 Technical Requirements

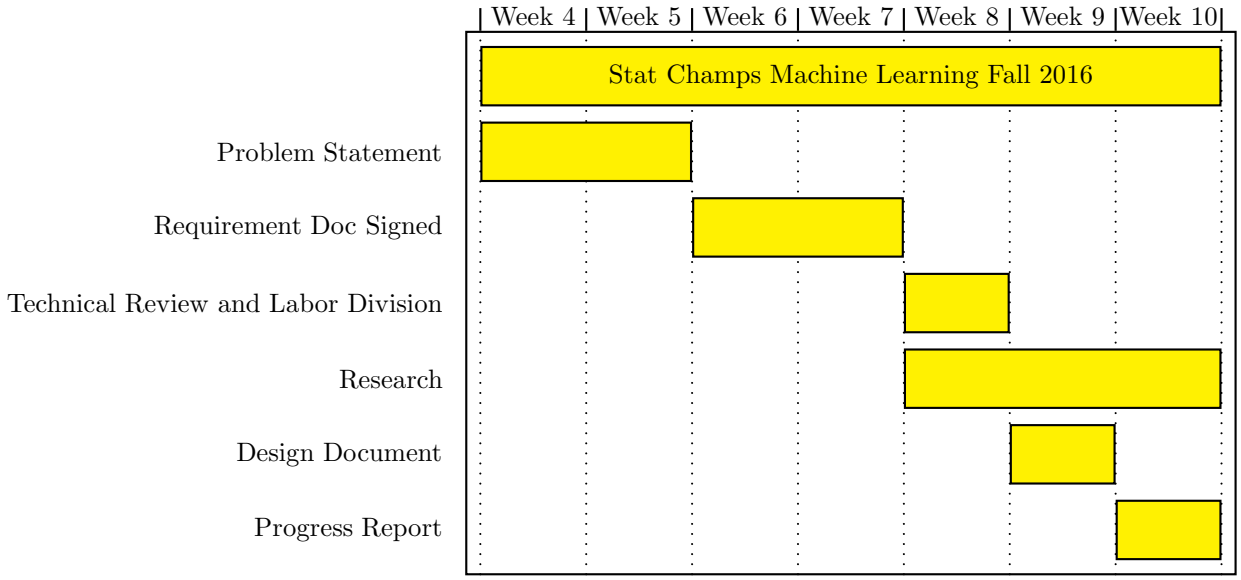
1. Cross-browser support (IE, Firefox, Chrome) for both GUI (using webix) and machine learning submodule (using scikit).
2. API with acceptable level of documentation

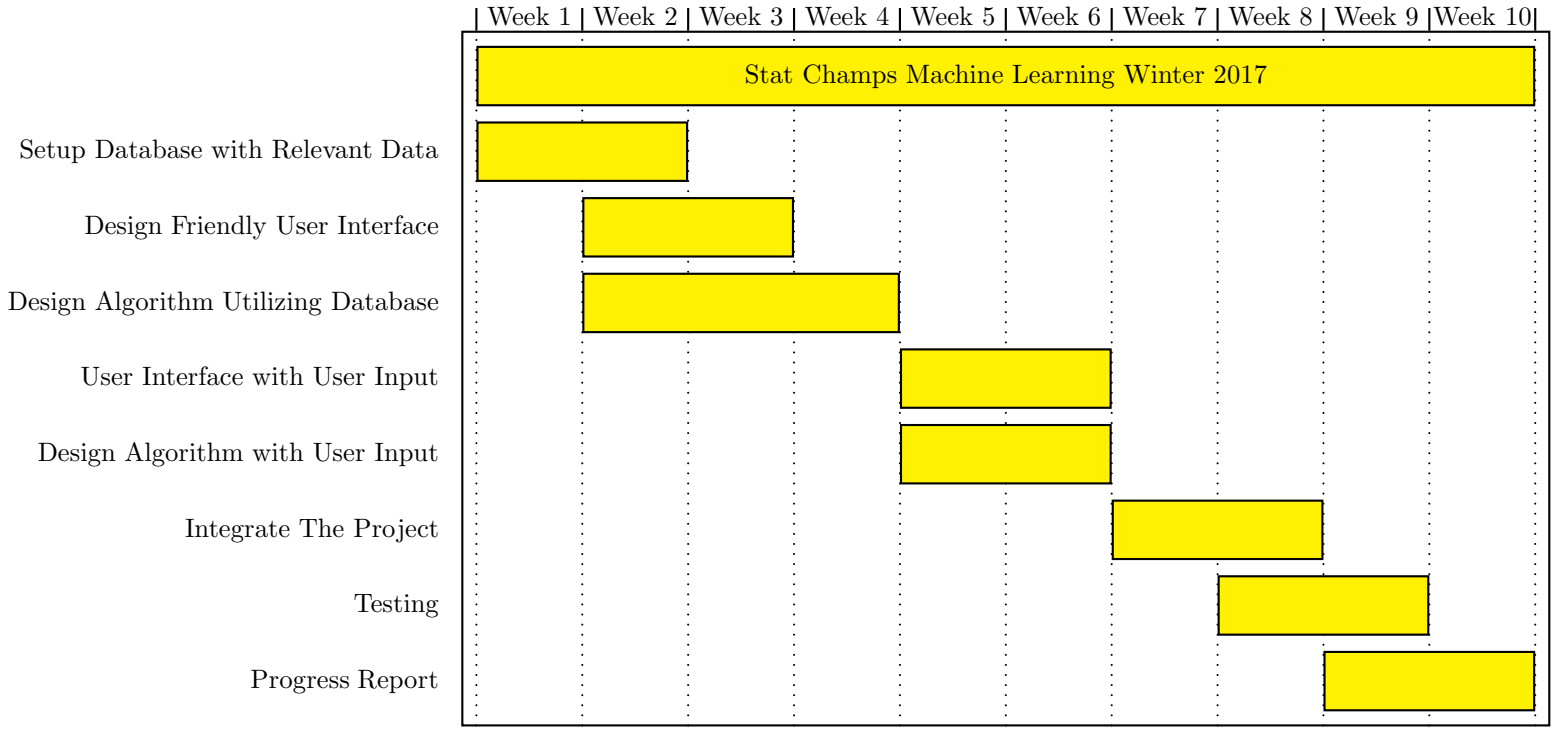
3.3 Usability Requirements

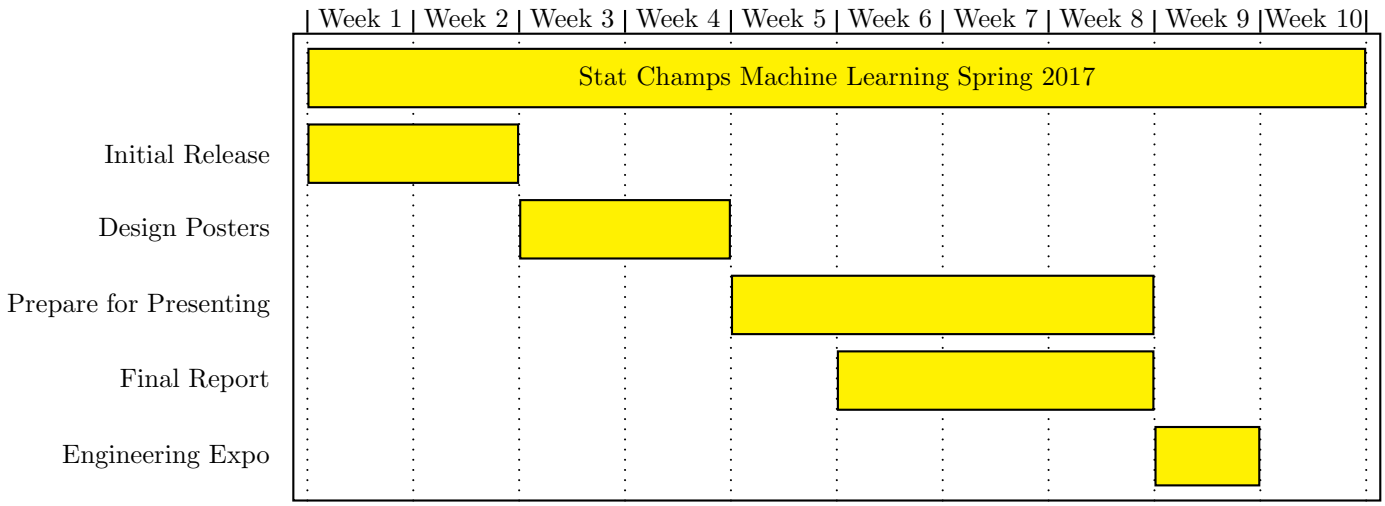
1. The system will look and act the same in all major browsers (outlined above).

4 Appendix

4.1 Gantt Chart






Week 1
Week 2
Week 3
Week 4
Week 5
Week 6
Week 7
Week 8
Week 9
Week 10
Stat Champs Machine Learning Spring 2017
Initial Release
Design Posters
Prepare for Presenting
Final Report
Engineering Expo

5 Agreement

Client

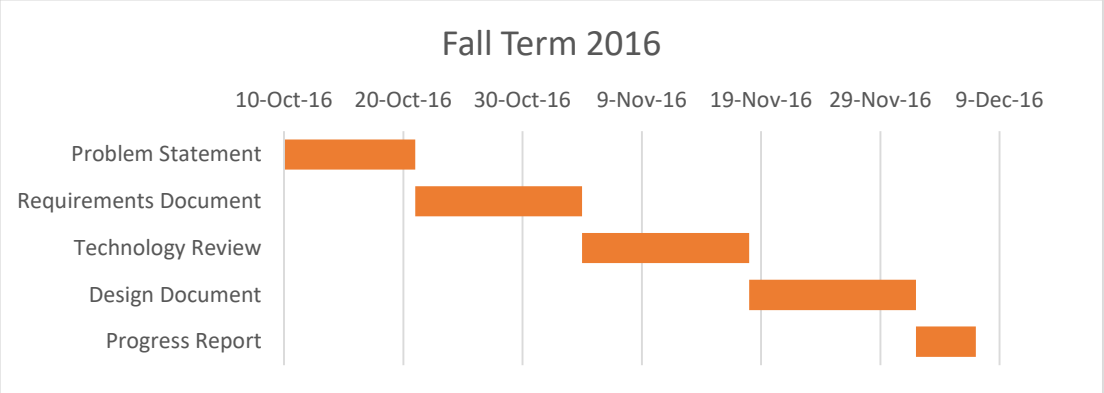
Developer

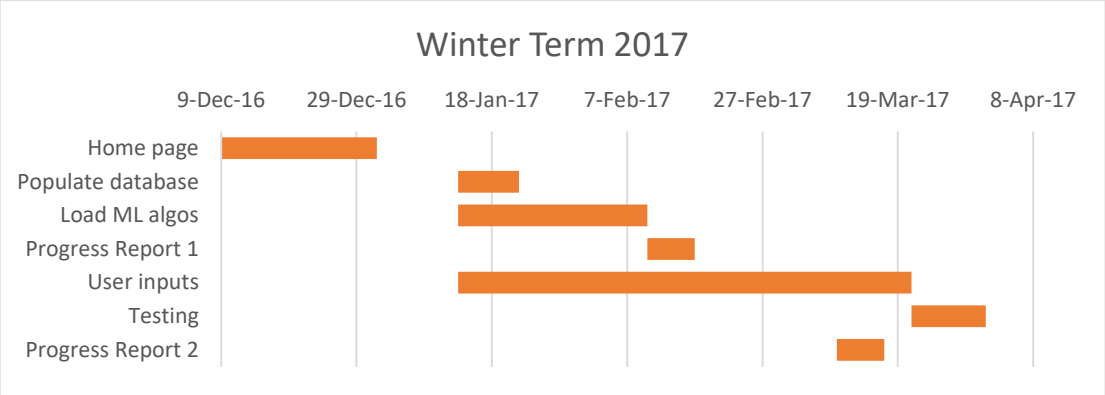
Developer

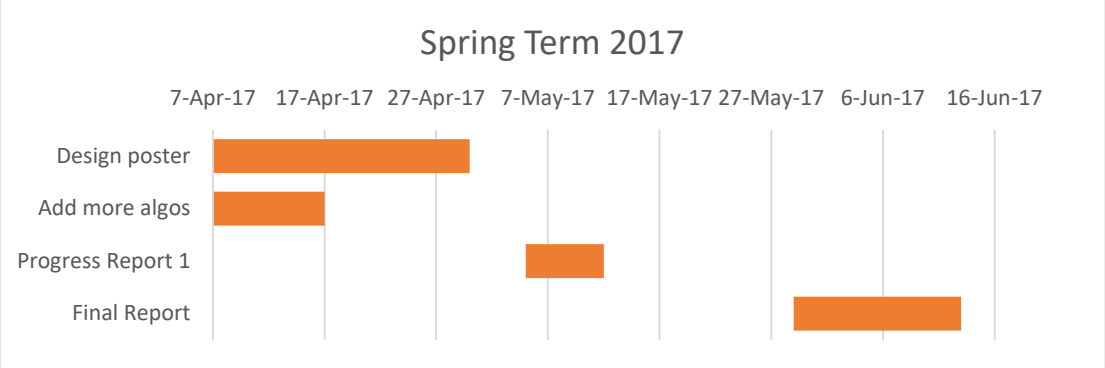
Developer

2.1 Changes from Original Requirements Document

2.2 Revised Gantt Charts


Fall Term 2016
10-Oct-16
20-Oct-16
30-Oct-16
9-Nov-16
19-Nov-16
29-Nov-16
9-Dec-16
Problem Statement
Requirements Document
Technology Review
Design Document
Progress Report


Winter Term 2017
9-Dec-16
29-Dec-16
18-Jan-17
7-Feb-17
27-Feb-17
19-Mar-17
8-Apr-17
Home page
Populate database
Load ML algos
Progress Report 1
User inputs
Testing
Progress Report 2


Spring Term 2017
7-Apr-17
17-Apr-17
27-Apr-17
7-May-17
17-May-17
27-May-17
6-Jun-17
16-Jun-17
Design poster
Add more algos
Progress Report 1
Final Report

3 Original Design Document

Design Document for Machine Learning for March Madness

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

TABLE OF CONTENTS

I. Introduction

Stat Champs
December 2, 2016

AFTER making reasoned decisions about which technologies to use to implement this module, we must now consider how these technologies will work together. This design document intends to explain how the three technologies each of the Stat Champs have selected to use will complete their assigned task. It will do so through the use of diagrams followed by paragraphs describing what the technology must do and why it must do it. The first three designs were produced by Alex Hoffer. These first three designs 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 display of the machine learning bracket that is generated. In technologies 4 and 5 Jake Smith will talk about how we will obtain and store the sports statistics for easy use by our machine learning algorithms. 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. Chongxian Chen will be responsible for designing and implementing machine learning algorithm. Technologies 6, 7 and 8 will be discussing technology involved in designing machine learning algorithm, namely algorithm library, statistics model and cloud server for hosting the project

II. Glossary

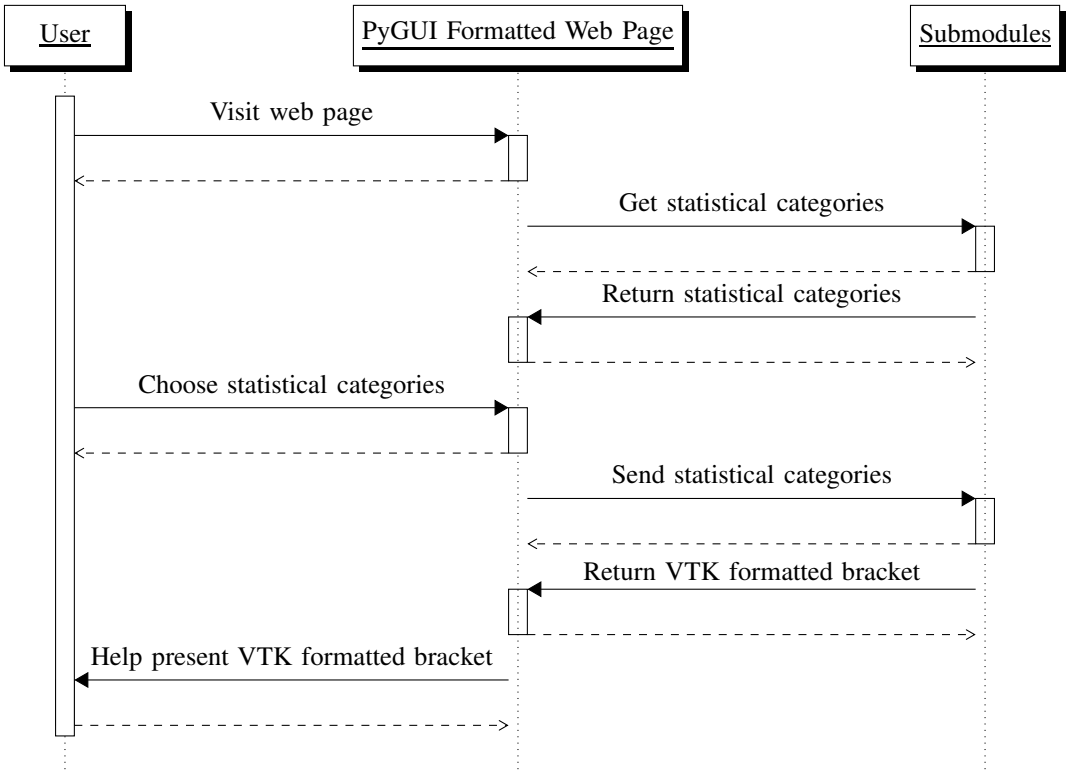
PyGUI: Graphical user interface API designed specifically for use with the Python programming language.
VTK: Data visualization API that will be used with the Python programming language.
Client-server architecture: Website format that consists of a user sending and receiving data to a server which also sends and receives data.
Webix: JavaScript API that is designed to support graphical user interfaces.
AWS: Amazon Web Service.
EC2: Amazon Elastic Compute Cloud.
SSH: Secure Shell.
RDS: Amazon Relational Database Service.
2P = 2-Point Field Goals
2P Percentage = 2-Point Field Goal Percentage
2PA = 2-Point Field Goal Attempts
3P = 3-Point Field Goals
3P Percentage = 3-Point Field Goal Percentage
3PA = 3-Point Field Goal Attempts
AST = Assists
AST Percentage = Assist percentage is an estimate of the percentage of teammate field goals a player assisted while he was on the floor.
Award Share = The formula is (award points) / (maximum number of award points).
BLK = Blocks
BLK Percentage = Block percentage is an estimate of the percentage of opponent two-point field goal attempts blocked by the player while he was on the floor.
BPM = Box Plus/Minus is a box score estimate of the points per 100 possessions that a player contributed above a league-average player, translated to an average team.
DRB = Defensive Rebounds
DRB Percentage = Defensive rebound percentage is an estimate of the percentage of available defensive rebounds a player grabbed while he was on the floor.
DRtg = Defensive Rating for players and teams it is points allowed per 100 possessions.
DWS = Defensive Win Shares
eFG Percentage = Effective Field Goal Percentage; the formula is (FG + 0.5 * 3P) / FGA. This statistic adjusts for the fact that a 3-point field goal is worth one more point than a 2-point field goal.
FG = Field Goals (includes both 2-point field goals and 3-point field goals)
FG Percentage = Field Goal Percentage; the formula is FG / FGA.
FGA = Field Goal Attempts (includes both 2-point field goal attempts and 3-point field goal attempts)
FT = Free Throws
FT Percentage = Free Throw Percentage; the formula is FT / FTA.
FTA = Free Throw Attempts
Four Factors = Dean Oliver's "Four Factors of Basketball Success
G = Games
GB = Games Behind
GmSc = Game Score: was created to give a rough measure of a player's productivity for a single game. The scale is similar

to that of points scored, (40 is an outstanding performance, 10 is an average performance, etc.)
GS = Games Started
MP = Minutes Played
MOV = Margin of Victory
ORtg = Offensive Rating for players it is points produced per 100 posessions, while for teams it is points scored per 100 possessions.
ORB = Offensive Rebounds
ORB Percentage = Offensive rebound percentage is an estimate of the percentage of available offensive rebounds a player grabbed while he was on the floor.
Pace = Pace factor is an estimate of the number of possessions per 48 minutes by a team. (Note: 40 minutes is used in the calculation for the WNBA.)
PER = Player Efficiency Rating The PER sums up all a player's positive accomplishments, subtracts the negative accomplishments, and returns a per-minute rating of a player's performance
PF = Personal Fouls
Poss = This formula estimates possessions based on both the team's statistics and their opponent's statistics, then averages them to provide a more stable estimate.
PProd = Points Produced
SOS = Strength of Schedule; a rating of strength of schedule. The rating is denominated in points above/below average, where zero is average
SRS = Simple Rating System; a rating that takes into account average point differential and strength of schedule.
STL = Steals
STL Percentage = Steal Percentage is an estimate of the percentage of opponent possessions that end with a steal by the player while he was on the floor.
Stops = Measure of individual defensive stops
TOV = Turnovers
TOV Percentage = Turnover percentage is an estimate of turnovers per 100 plays.
TRB = Total Rebounds
TRB Percentage = Total rebound percentage is an estimate of the percentage of available rebounds a player grabbed while he was on the floor.
TS Percentage = True shooting percentage is a measure of shooting efficiency that takes into account field goals, 3-point field goals, and free throws.
TSA = True Shooting Attempts
Usg Percentage = Usage percentage is an estimate of the percentage of team plays used by a player while he was on the floor.
Win Probability = The estimated probability that Team A will defeat Team B in a given matchup.

III. Designs

Design viewpoint 1: Using PyGUI for GUI of webpage

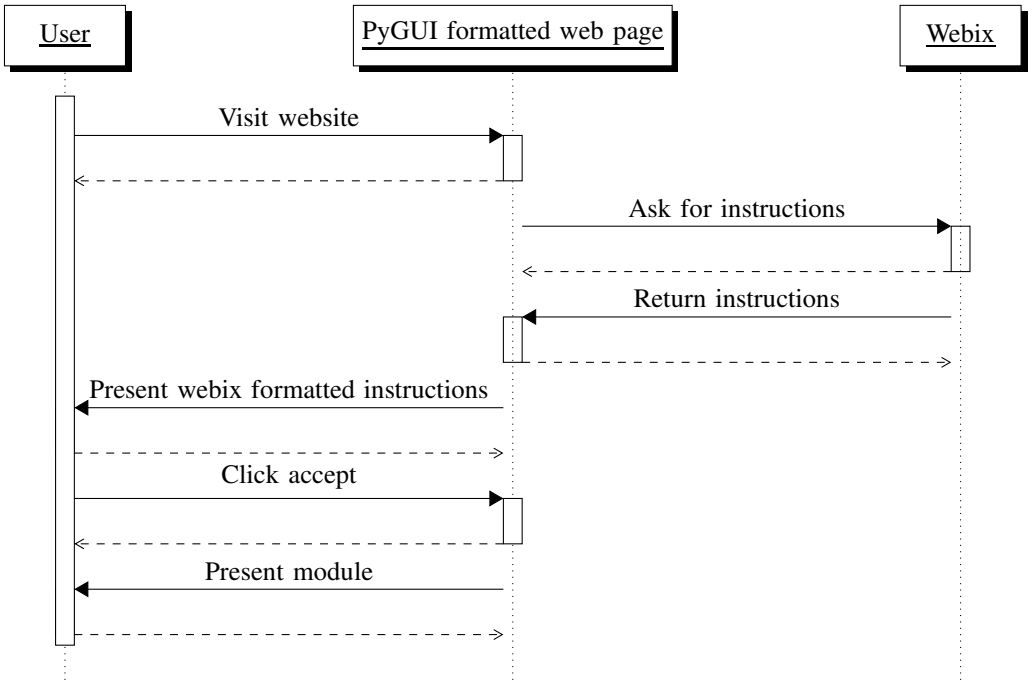
Design view 1:



Design Rationale: The module needs a web page to operate. This web page needs to be presented in a way that is pleasing to the eye and easy to use, because our intended users are biochemists, not computer scientists. This means the page needs to be as non-esoteric as possible. To achieve this, we will be using PyGUI to format our web page. The precondition of this sequence diagram is that the user has a browser. The postcondition of this diagram are that the user is presented with a bracket that resulted from collaboration between PyGUI and VTK. The flow of events from the perspective of PyGUI is quite simplistic. While a typical client-server architecture will allow us to transfer data back and forth between our submodules, PyGUI must make this data transmission appear as convenient as possible to the user.

Design viewpoint 2: Using Webix to present instructions for the module

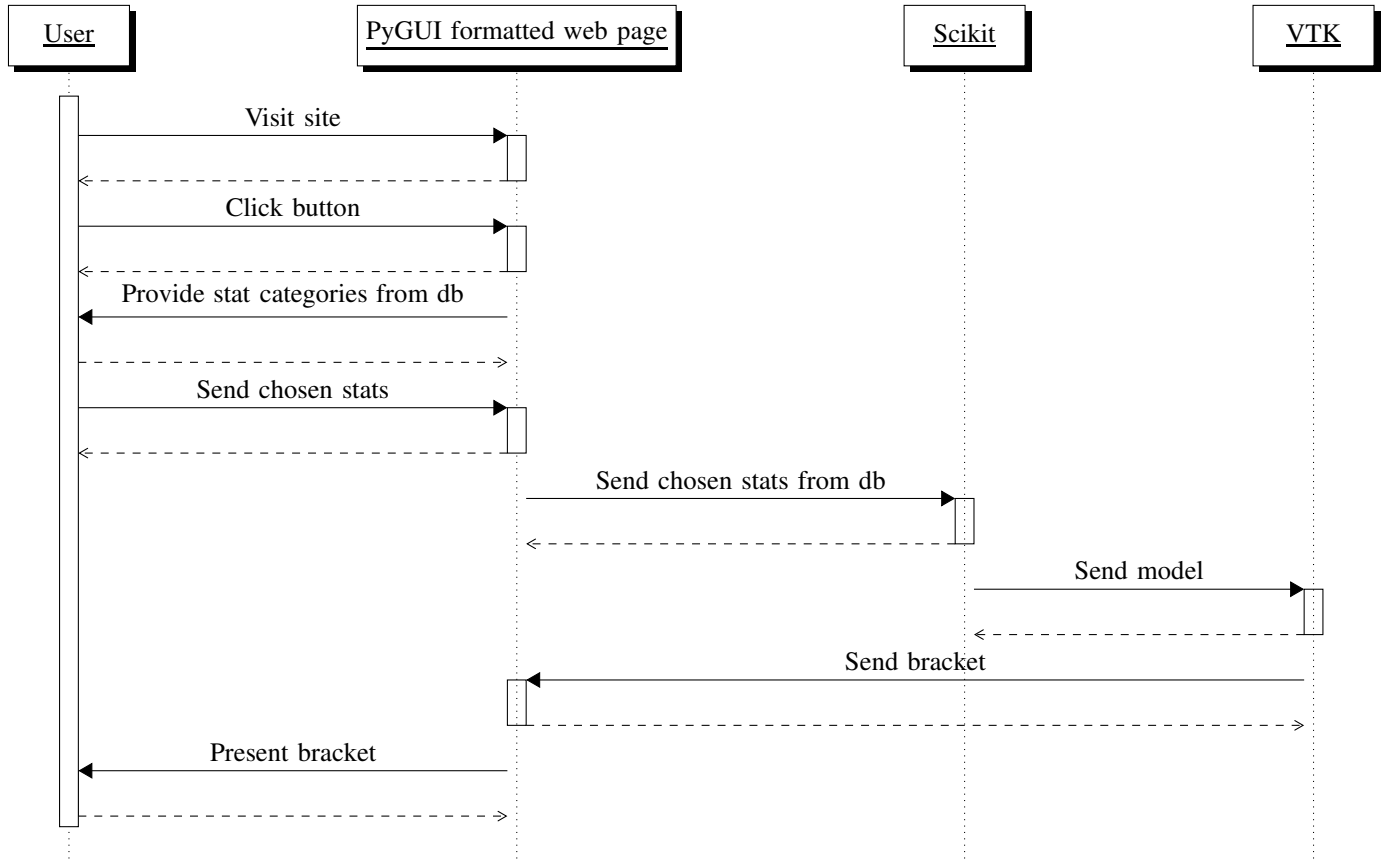
Design view 2:



Design Rationale: The module needs to present instructions to the user which informs them how to use the service. These instructions should appear seamlessly and should be visually inoffensive. Our main aim is reducing the chance of these instructions appearing unclear and thus maximizing utility. To achieve this, we will be using Webix, which will interact with our client/server code as well as with PyGUI. The precondition of this sequence diagram is that the user has a browser and has loaded the web page. The postconditions are that the user has seen the instructions (hopefully having read them) and has clicked accept to begin the service. The flow of events of the average use case are as follows: 1) the user visits the web page 2) the user is presented with instructions on properly using the module 3) the user reads the instructions 4) the user clicks accept 5) the module is presented. Note that the PyGUI formatted page in this diagram is a bit vague. PyGUI, of course, cannot handle the entire module. The web page will be calling a number of other technologies to present the module, but these technologies are not necessary when only considering how Webix will be used.

Design viewpoint 3: Generating bracket using VTK

Design view 3:



Design Rationale: The most essential function of the entire module is presenting a machine learned March Madness bracket. However, the process of machine learning is completely separate from the design of the bracket which will be displayed. Once Scikit generates a machine learned model that reflects the user's chosen statistics, it must communicate to VTK what this data is, and VTK will handle the bracket generation. In order for VTK to produce a bracket for display, a number of preconditions must occur. First, the user must have read and accepted the instructions. Then, the user must have selected data produced by a database. This data must be passed to Scikit, which generates a learned model. The postconditions of this sequence diagram is that the user will have seen the March Madness bracket that results from their chosen statistics. The flow of events for the average use case is as follows: 1) The user reads and accepts the instructions 2) The user selects statistics 3) The server sends these statistics to a machine learning submodule 4) The submodule generates a model 5) The submodule passes the model to VTK 6) VTK transforms the model into a bracket 7) A collaboration of VTK, PyGUI, and the server produce this bracket for the user to view.

Design viewpoint 4: Collection of data for the database

Design Rationale: The Machine Learning algorithms we want to work with get more accurate the more examples and stats you feed into it. Therefore, for the data collection we need to include as many possible data points as possible. We will do this by collecting stats from every major stat website and also pull more advanced data from the hoop-math which breaks down each players moves into more specialized data points. For example, hoop-math with break down a players shots from just saying he shot a 2 pointer to telling you where the shot was and how close a defender was to him at that moment. The design to grab the stats is to manually download all the csv files I can from Sports-Reference and upload them to the database then once I am caught up with the current season I will create a python script that takes the stats from all the daily games and uploads it into the database.

Design viewpoint 5: Database Design

Design view 4: Couldnt get UML database design to post correctly. Design Rationale:

For designing the database I have decided to have multiple connected databases. The fist will be a running tally for all the players stats, everything from FG percentage to player efficiency and more. The second will be all the team game logs with their opponents stats for that game as well. That way we can do look ups for past matchups and analyze how both teams and players did against each other and apply that for future matchups. Then for the final database that will be connected to the game logs will be the advanced game player stats for example it will have play by play stats like who passed to who, when and who shot the ball, from where, and who was guarding him. The user will be able to choose between either all three databases in there bracket analysis or just two or even one. Each bracket should be able to bring in stats that should change

the outcome of the machine learning predictions.

Design viewpoint 6: SciKit Learn Machine Learning Model Design Rationale:

The project will be implementing the prediction model using python machine learning library SciKit-Learn. More specifically, we will be using the supervised learning regression model.

The model will be starting with simple and straightforward input first, and then gradually add more complex and hard-to-predict input. The benefit of starting with straightforward input is that we can easily verify the prediction accuracy of our model. For instance, the most straightforward input could be the match results between two teams in recent years. In most situation the team that wins significantly more in the recent matches should have a higher chance of winning current match. After confirming that our model could predict well on basic inputs, we can then add more complex inputs that may also affect the result of the game like home team status, weather and rest status.

After verifying that our model?s prediction accuracy is satisfying, we will be working on enable and disable categories of inputs to meet our project?s education purpose. When enabling all or most categories of inputs, we will be expecting our model to be more accurate. With trivial category of input, our model may show very inaccurate prediction that may even contradict to recent match results. The education purpose of our project will allow users to see a difference when they choose different category of data.

When the model is basically complete, we will be testing it with actual matches, particularly the march madness event. We will be continuing modifying our algorithm to enhance the prediction accuracy with live matches.

Timeline: Training model with basic inputs: Winter Week 1 to Week 3. Training model with more categories of inputs: Winter Week 4 to Week 6. Enable and disable category of inputs: Winter Week 7 to Week 9. Testing and improving the model with March Madness: Spring Week 2 to Spring Week 5.

Design viewpoint 7: Amazon Web Service(AWS) to host the database and computing power Design Rationale:

We will be hosting our database and computing machine on Amazon Web Service. AWS is free for the first 12 months of registration with some limitation of use. After researching on AWS website, we will be able to host one linux instance free with Amazon EC2. We will also be able to hosting a free MySQL database with limited I/O from Amazon RDS. Although the I/O times are limited, but it is large enough for our project(10,000,000 I/Os). Hosting our computing power and database on the cloud is a good idea because we will be able to easily resize our computing power and database with AWS infrastructure when our project grows to a bigger size.

To start with AWS free tier for 12 months , we will be creating a group account. First we will be going to EC2 and host a Linux instance. We will have to choose a region among Amazon?s global servers. The one that is most convenient to us will be US West(Oregon). The operating system we will be using will be Ubuntu Server. Ubuntu is very popular among developers and have a large supporting community if we have trouble. Python can run seamlessly on Ubuntu terminal. For the instance type, we have only one option. T2.micro with one core cpu, one Gigabyte Ram and 8 Gigabyte Storage. The Ram is not very big so we may need to be careful with our ram usage or upgrade it in the future.

After creating the EC2 instance, we will moving forward to create the RDS database instance. We will be using relational database MySQL to store our data. The free tier also provide us with one core cpu, one Gigabyte Ram and 8 Gigabyte Storage. That should be enough for the teams in NCAA. We can also easily upgrade it when our project grows. When completing the setup of the database on AWS, we need to manually connect it to a database management tools so we can create tables and manage data. A convenient cross-platform tool we can use will be MySQL workbench. It is open-sourced, free and works very well on most platforms including Windows, Mac and Linux. We will be using SSH to connect the database on AWS to MySQL workbench. After that we can manage it like we have learned in class. Running SQL queries or create table using MySQL workbench?s graphical interface.

Timeline: Creating EC2 and RDS instance on AWS: Winter week 1 to week 2. Connecting to EC2 and RDS instance: Winter Week 3 to Week 4.

Design viewpoint 8: Statistics Model Design Rationale:

With the Statistics Libraries in Python like Py-Statiscs and a large amount of input data in our library, our statistics model is expected to give a good estimate about the weight of each factor in our prediction model. This factor will vary for different teams, but we will have a general statistics equation. After providing it with a large amount of data for each teams, the equation is expected to give respective weight for each team. And then we can supply these weight to our machine learning model to get the final result.

First we will be implementing the equation with basic inputs matching results. With only matching results, the equation may doesn?t make too much sense to estimate the weight since there is nothing to compare with. Then we should add a trivial data to the equation so we can tell the difference. We will be expecting the trivial data to have a significantly less weight than recent match results. For the trivial data, the candidates are weather, team colors, etc. We will try the team colors first since that is an easy to collect data category for the beginning of our project.

After the basic statistics model makes sense with the data category we provide, we will be adding more category of data like player info, home game status, coach info etc. With a large amount of data, we should be able to a weight of each of these categories. Then we can apply these weights in our prediction model. We will be expecting to see our model become more

accurate. With more important categories of data added, our prediction should be more confident and we will be testing it with the march madness results next Spring. With the new data, we will be testing the accuracy of our predictions and improving it.

Timeline: Implement statistics model with basic inputs: Winter Week 1 to Week 3. Implement statistics model with more categories of inputs: Winter Week 4 to Week 6. Enable and disable category of inputs and generate perspective weight: Winter Week 7 to Week 9. Testing and improving the statistics model with March Madness: Spring Week 2 to Spring Week 5.

IV. AGREEMENT

Client

Developer

Developer

Developer

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].

[9] Amazon Machine Learning, *Amazon*. [Online]. Available: https://aws.amazon.com/machine-learning/. [Accessed: 14-Nov-2016].

[10] "SciKit Learn,Machine Learning in Python, *scikit-learn*. [Online]. Available: http://scikit-learn.org/stable/. [Accessed: 14-Nov-2016].

[11] "Pylearn2 devdocumentation, *Pylearn2*. [Online]. Available: http://deeplearning.net/software/pylearn2/. [Accessed: 14-Nov-2016].

[12] "ONID - OSU Network ID, *Oregon State University*. [Online]. Available: http://onid.oregonstate.edu. [Accessed: 14-Nov-2016].

[13] "Python statistics Mathematical statistics functions, *Python*. [Online]. Available: https://docs.python.org/3/library/statistics.html. [Accessed: 14-Nov-2016].

[14] "Statistical functions (scipy.stats), *scipy.stats*. [Online]. Available: https://docs.scipy.org/doc/scipy/reference/stats.html. [Accessed: 14-Nov-2016].

[15] "Python Data Analysis Library, *Pandas*. [Online]. Available: http://pandas.pydata.org. [Accessed: 14-Nov-2016].

3.1 Changes from Original Design Document

4 Original Technology Review

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.

TABLE OF CONTENTS

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 three possible technologies were written by Alex Hoffer. These first three 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. In technologies 4 and 5 Jake Smith will talk about how we will obtain and store the sports statistics for easy use by our machine learning algorithms. 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. Chongxian Chen will be responsible for designing and implementing machine learning algorithm. Technologies 6, 7 and 8 will be discussing technology involved in designing machine learning algorithm, namely algorithm library, statistics model and cloud server for hosting the project

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.

D. Technology 4: Where the data will come from

1) Options 1, 2, and 3: Ncaa.com, sports-reference.com, and kenpom.com

2) Goals for use in design: The Goal is to determine where to get the statistics from that can provide an easy way to take the data off the website as well as have all the necessary statistical categories were looking for.

3) Criteria being evaluated: Cost is a factor because there are websites such as kenpom that offer advanced statistics for mens ncaa basketball but you must pay to access the data. Websites such as ncaa.com and sports-reference are free to use and browse the statistics. The amount of stats available is important because the more data points you have to work with the better the machine learning algorithms will do. Kenpom has an advantage in this area, it offers more statistical categories than both ncaa and sports-reference.com. Some websites even offer statistics of where each player on the court was at all times during the game. The ease of use is another important factor both kenpom and sports-reference provide downloadable csv files that can easily be dealt with to put in a database while ncaa.com does not. Correctness of the data is important because the data must be accurate for the machine learning algorithms to do their best. Statistics for both ncaa and sports-reference are backed up by the ncaa or a third party like espn. Kenpom and cites you must pay for your data there is no certainty that the advanced data will be accurate.

4) Table comparing options:

Website	Cost	Amount	Ease of use	Correctness	S
NCAA.com	Free	All normal statistics	Need to write script to get data	Confirmed	S
Sports-reference.com	Free	All normal statistics	Downloadable csv file	Confirmed	
Kenpom.com	Costs money	All normal statistics plus more advanced	Downloadable csv file	Not validated	

5) Discussion: Sports-reference is a great option because it has most of the statistics we will need and it is easy to download and use in the csv files unlike the other free option ncaa.com which has roughly the same statistics but without the easy to use downloadable csv. For ncaa.com we would need to write a piece of code that scrapes statistics that we need from the website to gather the data. But if we did go that route we could scrape all the sites (ex. Espn, yahoo) and compare them against each other to see if they all are correct. Or we could pay for the data using a site like kenpom.com and possibly get data points we would not get otherwise using the two free options should as player movement and touches.

6) Selection of best option: Sports-reference.com is the best option because its free, easy to deal with and has enough data to go on. More data can always be scraped off other free websites. It also provides a deep history of the teams statistics so we could incorporate those data points as well.

E. Technology 5: Storage of the data

1) Options 1, 2, and 3: Excel, Python SQLite, and phpmyadmin

2) Goals for use in design: Goal is to figure out the best database option to use that will be the easiest to integrate with the website and machine learning algorithms.

3) Criteria being evaluated: Cost isnt a factor is this one because they are all free or offered through OSU. The availability and security of the system arent a factor because they are all available and the data in the database is not sensitive so it does not need to be encrypted or protected in any extra fashion. Ease of use is a factor because the database needs to be able to integrate with the website nicely as well as be able to add and delete data easily. Speed wont be a factor because they all offer around the same speeds.

4) Table comparing options:

Technology	Cost	Ease of use	Security	Speed
Excel	Free	Easy	Protected	Fast
Python SQLite	Free	Easy	Protected	Fast
Phpmyadmin	Free	Easy	Protected	fast

5) Discussion: They are all viable options to use for this project each do about the same thing except excel and phpmyadmin would be external from the machine learning python code. Both excel and phpmyadmin are both nice to use because you can search on your dataset easily. They are all very easy to interact with via python code but phpmyadmin has the big bonus of having a nice user interface for the database as well as the ability to add data very easily via csv files which are the form most of the statistics would come in. The big downside of SQLite, a database included with Python, is it creates a single file for all data per database. Other databases such as MySQL, phpmyadmin, and Oracle and Microsoft SQL Server have more complicated persistence schemes while offering additional advanced features that are useful for web application data storage.

6) Selection of best option: Given that phpmyadmin has the user interface and it is very csv file friendly I am going to have to choose it for our database storage.

F. Technology 6: Machine Learning Libraries/Platforms to Train the Model

1) Options 1, 2, and 3: SciKit, Amazon Machine Learning(AML), Pylearn2.

2) Goals for use in design: Our Algorithm should be as accurate as possible. It should be highly scalable considering we may have a lot of basketball match data. We also want full control of our algorithm, i.e. have access to the source code if we are using some library in case we may need to modify them for our project.

3) Criteria being evaluated: The cost is definitely one important aspect to be considered in our project. We want to make our model available to most people so they can learn that machine learning is a power tool to use in many areas. Providing access to as many people as possible at a low rate or free is our goal. The availability is also considered. Some service may only be available in some countries. We should try to avoid those that are only accessible in a small area. Speed is also an important aspect of our algorithm. Making our algorithm efficient really enhance the user experience a lot. Finally security is something to be considered in the process but not too important since we dont have private user information. But keep in mind that a major flow in our project could cause danger to the system.

4) Table comparing options:

Technology	Economy	Readability	Availability	Speed	Notes
SciKit	Extremely	very	Open Source	Very	The most popular ML library
AML	very	very	Open Source	Normal	Easy to start with but hard to modify
Pylearn2	Extremely	very	Open Source	Very	Popular Library but no longer actively developing

5) Discussion: All three options are very reliable. Amazon Machine Learning(AML) is a fast developing platform that provides easy Machine Learning model to many customers with different background to start with[9]. The advantage of Amazon Machine Learning model is that it has great GUI that makes the process easier. And according to its introduction, AML is very closed to the purpose of our model. With data training the model, it will give a prediction based on the data. Users can also narrow down the searching areas. But because AML is not open sourced I will first dismiss this option. Our project is likely to modify a lot of algorithm in order to better meet the potentially changing needs. SciKit is currently the most popular open source library on Python and has active developers developing them and fixing bugs[10]. While Pylearn2 is also a famous and a widely used library on Python, there is no developers responsible for developing them. Although Pylearn2 claims that they will continue reviewing pull request on Github, there is no active developer for the project[11].

6) Selection of best option: SciLearn is the best option for the need of our project. It is open-sourced, reliable and has active developers developing it. It is also widely popular which means we will be more likely to get community support.

G. Technology 7:Server for computational power and hosting our database

1) Options 1, 2, and 3: Amazon Web Service(AWS), Google Cloud Platform(GCP), Oregon State University Student Engineering Server(OSU Server).

2) Goals for use in design: Our project is likely to require strong computational power because our prediction model needs to evaluate a large amount of data. And if we want to access our model from different devices and different locations, it is important that we train our model on the cloud. A reliable database server on the cloud is also needed for the users to easily use our model worldwide and cross-platform.

3) Criteria being evaluated: Choosing the server for our project should consider the cost, availability, speed, reliability. All of these factors are extremely important. The reliability is the most important factor of them. Our data is precious and the machine-learning-algorithm-trained model is peculiar. Thus we can suffer any data loss. The speed is important as well. The AlphaGo from DeepMind trains itself by playing GO with itself millions of times every day. With fast speed, our model could be more accurate. Availability is also to be considered. For example, Google Cloud Server is not accessible while OSU server and Amazon Web Service are accessible in China.

4) Table comparing options:

Technology	Economy	Readability	Availability	Speed	Notes
AWS	very	very	Worldwide	Very	TA very popular service by Amazon
GCP	very	very	Most Countries	Very	Not Accessible from China
OSU Server	Extremely	Normal	Worldwide	Varies	Speed slows down when off campus

5) Discussion: When considering the server we use to host our model and data, it is extremely important to consider the factors mentioned above carefully. OSU Student Engineering Server is functionally complete. You can host Linux project there and OSU also provides database server. It is easy and familiar to use. If we have difficulty, we can directly access to tech support on campus. But it is not very fast outside campus. And OSU engineering server is not very flexible if Windows server is to be used. Also, student usually only have access to MySQL database[12]. Google Cloud Platform and Amazon Web Service are very similar. They both are hosted by giant companies in the US. The major difference between them when considering starting a new project is that Google is not accessible in China[13].

6) Selection of best option: Overall, after carefully comparing important aspects of them, I think Amazon Web Service is the most reliable and appropriate solution for our project.

H. Technology 8:Statistics Model to determine the importance of different categories of data

1) Options 1, 2, and 3: py-statistics, scipy.stats, Pandas

2) Goals for use in design: An important aspect of our project is a statistical model to determine the importance of data. With good python library and combination of machine learning training, we will get a pretty good estimate of how these data factors affect our result. Choosing a functional and fast statistical library will make our data analyzing easier and make our prediction more accurate.

3) Criteria being evaluated: The cost are all free for the three libraries. The availability is important to be considered. The speed is important to consider when choosing the libraries. We are going to use the functions a couple times, even likely in a loop that cause higher complexity. So the basic running time of functions in these libraries are important. Security is not too important since we dont collect sensitive data from user but should be considered for the security and reliability of the system.

4) Table comparing options:

Technology	Economy	Readability	Availability	Speed	Notes
py-statistics	Extremely	Extremely	Cross Platform	Not Sure	Official statistics library by Python Foundation
scipy.stats	Extremely	very	Cross Platform	Not Sure	Widely popular and sponsored Open Source Project
Pandas	Extremely	very	Worldwide	Cross Platform	Sponsored Open Source Project

5) Discussion: Py-statistics is a powerful library developed by python foundation. It is reliable and widely used. Most importantly, Python Foundation will keep updating it to ensure its performance and security[13]. Scipy is a widely used library that may provide some functions that others don't. Scipy also highlights its probability functions which will suit our project for predicting basketball result[14]. Scipy is sponsored open source project by ENTHOUGHT. It is very reliable and popular. Pandas is a Python data analysis Library sponsored by NUMFOCUS. The Pandas library highlights its high-performance, easy-to-use data structures and data analysis[15]. It is also very popular, widely used and actively maintained by developers.

6) Selection of best option: All three libraries are very powerful. Our project will use py-statistics first. But the performance will need to be tested when in developing. The other two options may also be used in different context.

II. Conclusion

There are many tools available which can be used to achieve the exact software product we desire. With careful consideration, we have located what we believe to be the correct tools for the job. For developing a graphical user interface which will allow students to easily and accurately interact with the module, we will be using PyGUI to develop our basic interface, Webix to provide the user with instructions on how to use the tool, and VTK to present the machine learned bracket in a way that is visually appealing. For getting the basketball data we will use sports-reference.com because of their downloadable csv files. We will store all the data using phpmyadmin which offers a nice user interface for us to work with. We will be mainly using SciKit Learn library in Python for our machine learning algorithm. Amazon Web Service is the best choice to host our project on cloud.

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].

[9] Amazon Machine Learning, *Amazon*. [Online]. Available: https://aws.amazon.com/machine-learning/. [Accessed: 14-Nov-2016].

[10] "SciKit Learn,Machine Learning in Python, *scikit-learn*. [Online]. Available: http://scikit-learn.org/stable/. [Accessed: 14-Nov-2016].

[11] "Pylearn2 devdocumentation, *Pylearn2*. [Online]. Available: http://deeplearning.net/software/pylearn2/. [Accessed: 14-Nov-2016].

[12] "ONID - OSU Network ID, *Oregon State University*. [Online]. Available: http://onid.oregonstate.edu. [Accessed: 14-Nov-2016].

[13] "Python statistics Mathematical statistics functions, *Python*. [Online]. Available: https://docs.python.org/3/library/statistics.html. [Accessed: 14-Nov-2016].

[14] "Statistical functions (scipy.stats), *scipy.stats*. [Online]. Available: https://docs.scipy.org/doc/scipy/reference/stats.html. [Accessed: 14-Nov-2016].

[15] "Python Data Analysis Library, *Pandas*. [Online]. Available: http://pandas.pydata.org. [Accessed: 14-Nov-2016].

4.1 Changes from Original Technology Review

5 Weekly Blog Posts

5.1 Alex

5.1.1 Fall Term

5.1.1.1 *Week 3*: This week we collaborated with Dr. Hsu and wrote our Project Statement, which he reviewed and signed off on. We submitted the Project Statement at 11 am on 10/14/2016. We did not really encounter any problems, except for maybe some LaTek formatting issues. I am sure the use of LaTek will become easier with more practice. Next week we will be submitting our resumes for peer review and figuring out how to proceed with this project now that we have a clear, working vision to follow.

5.1.1.2 *Week 4*: This week we finalized our project statement by revising it to reflect our instructors' suggestions. We also all produced resumes and gave them to classmates for feedback. Next week we will work on our project requirements document and attend Career Fair.

5.1.1.3 *Week 5*: This week we attended career fair and developed our project requirements document. Next week we will continue developing this document and will consult the client to get his approval.

5.1.1.4 *Week 6*: This week we revised the rough draft of our project requirements document, formatted it using LaTek, and added a Gantt chart. We submitted this requirements document to our client, but as of 3:20 pm have not heard back from him. By the end of the weekend, we hope to have the document signed and submitted. Next week, we will individually work on our tech reviews.

5.1.1.5 *Week 7*: This week we revised our project requirements document and began working on our tech review document. Next week we will finish our tech review document.

5.1.1.6 *Week 8*: This week we developed our technology review document. We ran into some issues coming up with 3 responsibilities for everybody. We also had some difficulty identifying potential technologies for each of these responsibilities. Next week we will complete our design document.

5.1.1.7 *Week 9*: This week we talked about how we would make our design document, and discussed how we would approach recording ourselves for the progress report. We are having some difficulties with design document formatting. We will probably rent out a microphone for use with a computer to record ourselves for the progress report. Next week we will turn in our design document.

5.1.1.8 *Week 10*: This week we finished our design document. We will send it to our client and get a signature as soon as possible. We faced difficulty in getting LaTex to properly generate designs like message sequence diagrams. Next week we will submit our progress report and conclude the term.

5.1.2 Winter Term

5.1.2.1 *Week 1*: This week we came back from winter break and re-calibrated. We voted on a meeting time (Tuesdays) and attended the first class. We look forward to the term.

5.1.2.2 *Week 2*: This week we had our first meeting back with our TA. Our meeting consisted of planning out the term for our team. This meant we re-established the responsibilities we set for ourselves individually last term and verbally sketched out an idea of what our Beta release would look like. My own contributions this week were setting up the web page the module will be hosted on and doing some GUI work. The web page can be found here: http://web.engr.oregonstate.edu/ hoffera/CapstoneProject/MachineLearnYourWayToMarchMadnessGlory.html. More

to come on the GUI work. Chongxian has set up some machine learning algorithms we can use so next week we will have Jake provide the data to them to see how they operate. After we get a handle on these algorithms, we will begin setting up the module.

5.1.2.3 *Week 3*: This week Chongxian arranged the machine learning algorithms, Jake compiled the statistics we'd use in a .csv file, and I worked on the GUI of our web page. Next week we have class on Thursday and wee should have the algorithms being allowed to accept stats as input.

5.1.2.4 *Week 4*: This week I continued to polish the GUI for the webpage. Chongxian has selected our machine learning algorithms and Jake has helped him find examples of how to implement them. Jake also gathered a lot of basketball statistics for use in the module. We need to do the OneNote portfolio, a progress report, and a voice-over update by late February.

5.1.2.5 *Week 5*: This week we attended class, Chongxian continued developing our machine learning algorithms, Jake continued to gather data, and I continued to develop our GUI. Next week we plan to release an alpha version of our module and we need to create a OneNote, edit our documents, make a status report, and submit these to the OneNote.

5.1.2.6 *Week 6*: This week we completed our progress report, both written and presentation versions. I made our OneNote and uploaded all of our documents to it. We had a bit of a hard time filling up all of the required time for the presentation. Next week we will continue development.

5.1.2.7 *Week 7*: This week we continued coding. I re-submitted my OneNote to Dr. Winters because it didn't go through the first time. I also met with Dr. Winters to modify my OneNote a bit. We need to finish our coding to be at a beta level release.

5.1.2.8 *Week 8*: This week I waited for Jake and Chongxian to make progress. Next week we need to setup a meeting with Dr. Hsu. We also are supposed to be presenting a beta release of our project.

5.1.2.9 *Week 9*: This week we did elevator pitches. I have made a draft of our poster using a LaTex conference poster template, and I added a signature page to our progress report and sent it to our client. My two partners should have made progress on integration. Next week I need to polish the poster more and write my new progress report, which I will use the IEEEtran format for.

5.1.3 Spring Term

5.1.3.1 *Week 1*: This week I went to class to get accustomed to what the term would entail. I updated some CSS on our page. Then, I emailed our client with a link to our project. Hope to hear from him next week and hope to update our poster for Expo.

5.1.3.2 *Week 2*: In week 2, Chongxian implemented different algorithms to choose from. We all updated the poster and submitted our new draft. We also emailed our client again with our finished product and our draft for him to sign off on. We await his response. In the coming weeks, we need his sign-off on the poster and then we have Expo.

5.1.3.3 *Week 3*: This week we attended class. I went to Dr. Hsu's office to talk to him about signing off on our posters. We also pushed our code to the Github repo. Our poster final is due May 1.

5.1.3.4 *Week 4*: This week I modified our poster to match our client's suggestions, then McGrath's suggestions. Then, I submitted the poster for printing. Next week I need to do the WIRED assignment.

5.1.3.5 *Week 5*: This week I interviewed Brandon Chatham for the WIRED assignment. We are preparing for Expo.

5.1.3.6 *Week 6*: This week we got our spring term progress report ready. Expo is next week and we're preparing.

5.1.3.7 *Week 7*: This week we had Expo. Term's almost over. In the remaining three weeks we need to edit our original docs, finish 3 small writing assignments, and do a final progress report.

5.2 Chongxian

5.3 Jake

6 Final Poster

COLLEGE OF ENGINEERING

Electrical Engineering & Computer Science

MACHINE LEARN YOUR WAY TO MARCH MADNESS GLORY!

Teaching Biochemists and Biophysicists Machine Learning

Introduction/Background

IMPORTANCE OF MACHINE LEARNING TO BIOCHEMISTRY AND BIOPHYSICS

Biochemistry and biophysics are two fields that are ripe with many exciting breakthroughs. Machine learning, a type of artificial intelligence where computer programs adapt to new data, is used by biochemists and biophysicists to do things like analyze genomic DNA sequences. Our client, a professor in the department of Biochemistry and Biophysics at OSU, recognized there was a need for his budding scientists to understand machine learning so they could be better prepared for their careers.

NEED FOR A MACHINE LEARNING INSTRUCTIONAL TOOL

Our client noticed that the Biochemistry and Biophysics curriculum at OSU did not encourage undergraduate students to learn machine learning. Even if machine learning classes were to become a cornerstone of their coursework, the content would be difficult for people without a Computer Science background. To make matters worse, teaching machine learning to these students through its application to biochemistry is particularly challenging, since biochemical results are non-definitive in that DNA sequences often do not need to be exact and can be unclear. Meanwhile, college basketball results are win-lose and therefore it is more straightforward to interpret the differences in results based on changing the inputs.

WHAT WE WERE COMMISSIONED TO DO

We were enlisted to produce an online instructional module where these students could grasp machine learning fundamentals in a clear manner. Our client wanted us to develop this module so that students could generate machine learned NCAA March Madness brackets. Since a fundamental aspect of learning machine learning is recognizing how the inclusion or exclusion of data influences resulting models, this module would satisfy the need by producing models (brackets) that were distinguishable from each other based on the college basketball statistics a user chose for training.

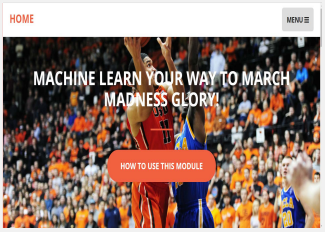


Fig. 1: Home page of website that includes project information and a link to our module.

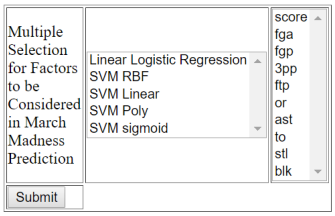


Fig. 2: Menu where the user chooses a machine learning algorithm and stats to generate a bracket with.

PROJECT INFORMATION

Class: CS Senior Capstone, 2016-2017

Developers:

- **Alex Hoffer** (hoffera@oregonstate.edu)
- **Jacob Smith** (smitjaco@oregonstate.edu)
- **Chongxian Chen** (chencho@oregonstate.edu)

Client: Dr. Victor Hsu, Oregon State University, Department of Biochemistry and Biophysics

PROJECT DESCRIPTION

To implement the module, we needed to complete the following five steps:

1. Develop a Graphical User Interface (GUI)
2. Aggregate/select college basketball statistics
3. Feed statistics to machine learner
4. Train a model using an algorithm of the user's choosing
5. Generate March Madness bracket that represents the model

The following headings are technical descriptions of the five steps:

1. GUI

Alex used HTML, CSS, and JavaScript to produce the GUI for our web page. HTML was used to split the page into logical sections such as Home (found in *Fig. 1*), Instructions, Module, Purpose, and About. We utilized CSS to make these sections look clean and usable. Finally, JavaScript was used to enhance the user experience by making the page interactive, such as turning certain buttons different colors upon clicking in order to notify the user of the action they had just performed.

2. AGGREGATE/SELECT STATISTICS

Jacob gathered college basketball statistics from 1985 to the current season from the website Kaggle.com in the CSV file format. Since the regular season didn't conclude until March, Jacob manually updated the database to reflect the current standings frequently until the final game was played. Then, he added stats from the tournament for future use in algorithms and analysis. We used a Python script to allow users to choose from a wide variety of stats including categories like field goals attempted per game to train a model on, as demonstrated by *Fig. 2*.

3. FEED STATISTICS TO MACHINE LEARNER

Using the Python SciKit-Learn library, Chongxian read the CSV files of the user selected statistics into Numpy arrays.

4. TRAIN A MODEL USING AN ALGORITHM

Along with their choice of statistics, users are also able to choose between different machine learning estimators such as Linear Regression and SVM Polynomial. By using a basketball ELO rating system, the supervised machine learning model is able to fit on the statistics and predict new matches. A CSV file of the match results between two teams with the probability is generated as a result. The bracket results effectively present how the users choice affects the machine learning prediction.

5. GENERATE BRACKET OF RESULTS

While a machine learned module is being generated, the user is presented with a screen that includes the command line arguments given to SciKit and informs the user on which steps are necessary to complete their request. The prediction CSV file generated from the machine learning model was then transferred into bracket form by Jake using a Python script.

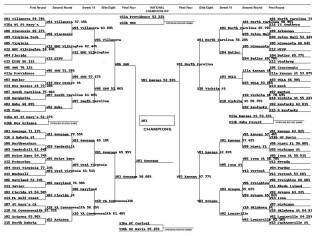


Fig. 3: A March Madness bracket predicted by the SVM RBF algorithm.

CONCLUSION

Alex Hoffer, Jacob Smith, Chongxian Chen

FEATURES PROVIDED BY THE MODULE:

- A Graphical User Interface (GUI) including a "Home" page, "Instructions" page, "About The Developers" page, and a "Purpose" page.
- The ability to select from a set of college basketball statistics.
- The ability to select from a set of popular machine learning algorithms.
- A machine learned bracket that corresponds to the specific statistics and algorithm the user requested the model to be trained on.

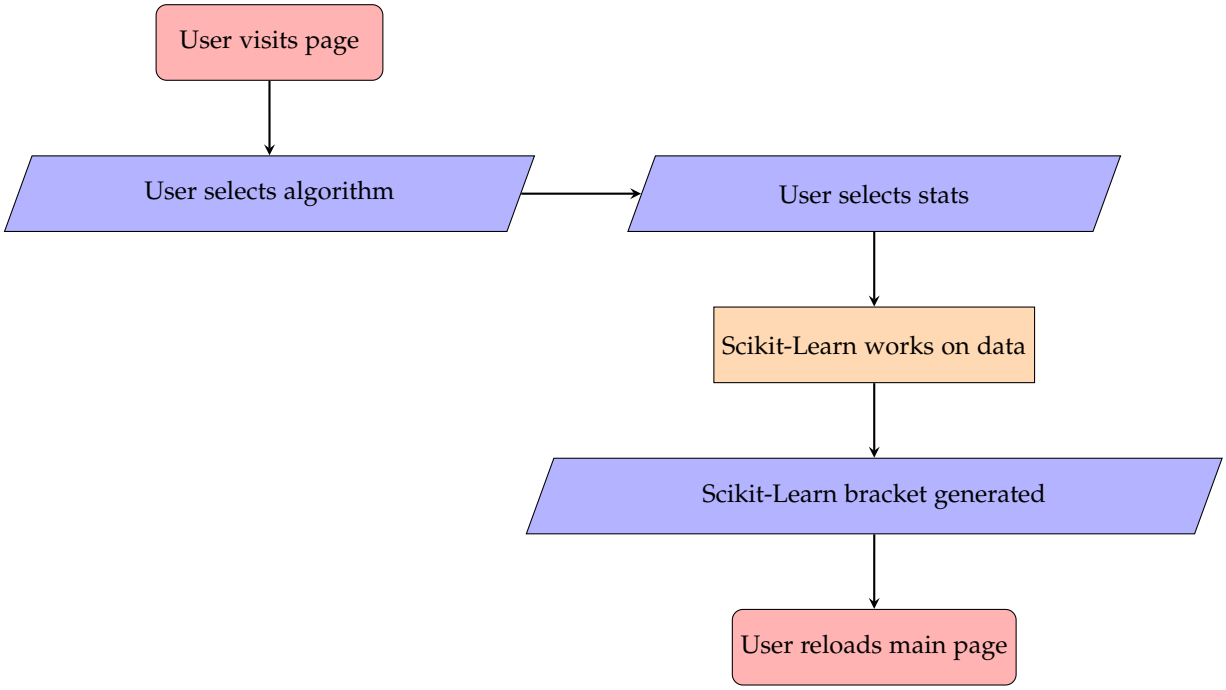
The project was completed in early April. All functional requirements as outlined by our client were completed. Future improvements to our module could include more machine learning algorithms, a wider variety of statistical categories, a more elegant looking outputted bracket, and finding a way to increase the speed at which the machine learning algorithms generate results. Additionally, the developers of such modules may wish to have prior machine learning experience, more proper modes of communication, and a more specific work schedule established before development in order to allow for time after completion to polish each component of the module separately.

Oregon State UNIVERSITY

7 Project Documentation

We were both lucky and unlucky to be assigned a project that did not make well-detailed documentation a necessary responsibility. We were lucky in the sense that this shaved off a lot of work we would have to do, and unlucky in the sense that this project did not expose us to how to properly write documentation.

7.1 How the project works



7.2 How to use the software

No installation is necessary. To use our software, all a user has to do is go to this website: http://web.engr.oregonstate.edu/ hoffera/CapstoneProject/MachineLearnYourWayToMarchMadnessGlory.html. On this site, a link to the machine learning module is posted. The module is compatible across all browsers. There are instructions on how to use the module on this site, as well as on the page where the module is hosted itself. To use the module, the user clicks on one machine learning algorithm and any number of the provided statistical categories and submits their choices. Then, a waiting period ranging from two minutes for simple machine learning algorithms to many hours for more precise algorithms is required. A message displays on the screen to inform the user they must wait. Then, the user is informed when the bracket has been generated. Finally, the user returns to the module page to see their generated bracket presented. The user can repeat this process an arbitrary number of times to generate an arbitrary number of brackets, but of course, the amount of time it takes to utilize certain machine learning algorithms is an important constraint the user must consider.

7.3 Requirements for usage

No special hardware, operating system, or runtime requirements dictate the usage of this module. As stated previously, any browser on any operating system with internet access can utilize the module.

7.4 API Documentation

We were not asked to document any API, and do not have any user guides. Such documents would not be particularly useful to any programmers because our module uses Scikit-Learn. This means that the API has already been documented for us by the fine people at Scikit-Learn. This documentation can be found here: http://scikit-learn.org/stable/documentation.html.

8 How We Learned New Technology

8.1 Alex

8.1.0.8 *Websites*:

- https://stackoverflow.com/
- https://www.w3schools.com/js/
- http://scikit-learn.org/stable/documentation.html

Stack Overflow was most valuable because it provided useful information on debugging small issues with web development. W3 Schools was second to this, because it gave me a lot of insight on how to use JavaScript to make a website *pop*. JavaScript is one of my weaker languages, so I needed it to make sure I was doing the right things. Finally, the documentation for Scikit-Learn was of course valuable for the inclusion of machine learning algorithms, but is listed last because implementation of machine learning algorithms was primarily Chongxian's responsibility.

8.1.0.9 *People*: Several people on campus were tremendously helpful in providing information that we needed. We owe a debt to our teaching assistant Xinze Guan for recommending Scikit-Learn for machine learning algorithm implementation. Our client, Dr. Victor Hsu, was also useful in this regard. We also relied on our instructor Kevin McGrath's LaTeX templates and Makefiles and Dr. Kirsten Winters' writing advice. Finally, my teammates helped me learn a lot of new and exciting technologies, with Jake demonstrating to me proper usage of LaTeX and Chongxian for exploring Scikit-Learn and elucidating a lot of its mysteries to me.

9 What We Learned

9.1 Alex

9.1.0.10 *Technical Information*: In terms of web development, I learned more JavaScript and polished my HTML/CSS abilities. I became acquainted with how Amazon Web Services works and what their limits on their free service are. I also learned a little on how to write embedded Python, how to use Scikit-Learn in Python, and how to use Python to scrape data from websites. Since these were not my primary responsibilities, though, I didn't learn them as well as I would've liked. Finally, I learned how to use LaTeX, proper usage of the IEEETran format, and the mathematical underpinnings of various machine learning algorithms.

9.1.0.11 *Non-Technical Information*: I learned how to interact with a software client. I also learned how to take an idea and move it through the necessary phases of development, including requirements, planning, and design phases all the way to implementation, maintenance, and presentation. Essentially, I learned the engineering practices necessary to develop a project from embryo to adulthood, and learned how to sell or pitch an idea.

9.1.0.12 *Project Work*: I learned that a project must run on a well-defined schedule. If each milestone of a project builds on a previous one, this is especially important. Without a rigid schedule, you fall behind sooner rather than later and the versions of your project you will release will be fraught with bugs.

9.1.0.13 *Project Management*: I learned that in a team setting, somebody has to be the team manager (whether de facto or not), otherwise milestones won't be reached and the project will be delivered late.

9.1.0.14 *Working in Teams*: I discovered certain methods for overcoming differences in work style. Some people procrastinate, others get work done early. There's nothing wrong with either approach, but if you're in a team where there are conflicting philosophies, you need to find out early how to bridge the gap between you and your group mates.

9.1.0.15 *What I would do differently*: I would choose different responsibilities for this project. I wish I could've done more of the machine learning aspects, rather than simply oversee Chongxian's progress. I would've considered using something besides Scikit-Learn because I would've liked to have used more exotic machine learning algorithms, or at the very least neural networks, which are not supported in Scikit-Learn.

9.2 Chongxian

9.3 Jacob

APPENDIX A

ESSENTIAL CODE LISTINGS

```
<!DOCTYPE html>
<html lang="en">

<!-- Page Title, CSS, JavaScript files to load in, etc. -->
<head>

    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta name="description" content="">
    <meta name="author" content="">

    <title>Machine Learn Your Way to March Madness Glory!</title>

    <!-- Bootstrap Core CSS -->
    <link href="vendor/bootstrap/css/bootstrap.min.css" rel="stylesheet">

    <!-- Custom Fonts -->
    <link href="vendor/font-awesome/css/font-awesome.min.css" rel="stylesheet" type="text/css">
    <link href='https://fonts.googleapis.com/css?family=Open+Sans:300
italic,400italic,600italic,700italic,800italic,400,300,600,
700,800' rel='stylesheet' type='text/css'>
    <link href='https://fonts.googleapis.com/css?family=Merriweather:400,
300,300italic,400italic,700,700italic,900,900italic'
rel='stylesheet' type='text/css'>

    <!-- Plugin CSS -->
    <link href="vendor/magnific-popup/magnific-popup.css" rel="stylesheet">

    <!-- Theme CSS -->
    <link href="css/creative.min.css" rel="stylesheet">

```

```
    <!-- HTML5 Shim and Respond.js IE8 support of HTML5 elements and media queries -->
    <!-- WARNING: Respond.js doesn't work if you view the page via file:// -->
    <!--[if lt IE 9]>
        <script src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv.js"></script>
        <script src="https://oss.maxcdn.com/libs/respond.js/1.4.2/respond.min.js"></script>
    <![endif]-->

</head>

<!-- Header stuff. Navigation buttons. -->

<body id="page-top">

    <nav id="mainNav" class="navbar navbar-default navbar-fixed-top">
        <div class="container-fluid">
            <!-- Brand and toggle get grouped for better mobile display -->
            <div class="navbar-header">
                <button type="button" class="navbar-toggle collapsed"
data-toggle="collapse" data-target="#bs-example-navbar-collapse-1">
                    <span class="sr-only">Toggle navigation</span> Menu <i class="fa fa-bars"></i>
                </button>
                <a class="navbar-brand page-scroll" href="#page-top">Home</a>
            </div>

            <!-- Collect the nav links, forms, and other content for toggling -->
            <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">
                <ul class="nav navbar-nav navbar-right">
                    <li>
                        <a class="page-scroll" href="#about">Instructions</a>
                    </li>
                    <li>
                        <a class="page-scroll" href="#services">Module</a>
                    </li>

        <li>
      <a class="page-scroll" href="#portfolio">Purpose</a>
        </li>

                    <li>
                        <a class="page-scroll" href="#contact">About</a>
                    </li>
                </ul>
            </div>
            <!-- /.navbar-collapse -->
        </div>
        <!-- /.container-fluid -->
    </nav>

  <!-- Title and content for first page presented. -->

```

```
    <header>
        <div class="header-content">
            <div class="header-content-inner">
                <h1 id="homeHeading">Machine Learn Your Way to March Madness Glory!</h1>
                <hr>
                <a href="#about" class="btn btn-primary btn-xl page-scroll">How to Use This Module</a>
            </div>
        </div>
    </header>

  <!-- Instructions Section -->

    <section class="bg-primary" id="about">
        <div class="container">
            <div class="row">
                <div class="col-lg-8 col-lg-offset-2 text-center">
                    <h2 class="section-heading">How to Use This Module</h2>
                    <hr class="light">
                    <p class="text-faded">To generate a machine learned March Madness bracket, click on the Load
the Module button below. You will be asked to
click on a link that will take you to the module.
From there, select one or more statistical
categories to train the bracket on from the
box at the top left of the page.
Allow the module two minutes
to generate a machine learned bracket.</p>
                    <a href="#services" class="page-scroll btn btn-default btn-xl sr-button">Load the Module</a>
                </div>
            </div>
        </div>
    </section>

  <!-- Module Section -->

    <section id="services">
        <div class="container">
            <div class="row">
                <div class="col-lg-12 text-center">
                    <h2 class="section-heading">Module</h2>
      <a href="http://crtour.xyz/CapstoneProject/V2/select.php">Click here to load the module.</a>
                    <hr class="primary">
        <a href="#portfolio" class="btn btn-primary btn-xl page-scroll">Purpose of module</a>
                </div>
            </div>
        </div>
  </section>

  <!-- Purpose Section -->

  <section class="no-padding" id="portfolio">
```

```
133         <aside class="bg-dark">
134         <div class="container text-center">
135             <div class="call-to-action">
136                 <h2>Purpose of Module</h2>
137     <p> This is an instructional tool intended to teach
138 students of Biochemistry and Biophysics at Oregon State
139 University machine learning concepts. Since machine learning
140 leads to remarkable discoveries in the fields of biochemistry
141 and biophysics, it is important students in this discipline
142 are able to understand how the inclusion or exclusion of
143 specific data sets influences a resulting model.
144 Rather than learn machine learning through its application
145 to biochemistry/biophysics (which would be especially
146 challenging), this module teaches students machine learning
147 through something that is simple.</p>
148                 <a href="#contact" class="page-scroll btn btn-default btn-xl
149 sr-button">About the developers</a>
150             </div>
151         </div>
152     </aside>
153     </section>
154
155   <!-- About Section -->
156
157
158       <section id="contact">
159        <div class="container">
160             <div class="row">
161                 <div class="col-lg-8 col-lg-offset-2 text-center">
162                     <h2 class="section-heading">About the Developers</h2>
163                     <hr class="primary">
164                     <p>The developers of this module are Alex Hoffer,
165 Chongxian Chen, and Jacob Smith. We developed this
166 project for our Computer Science Senior Capstone course,
167 in the 2016-2017 year. Our instructor was D. Kevin McGrath.
168 If you need to get in touch with us, please click the link below.</p>
169                 </div>
170
171     <!-- Email subsection -->
172
173                 <div class="col-lg-4 text-center">
174                     <i class="fa fa-envelope-o fa-3x sr-contact"></i>
175                     <p><a href="mailto:hoffera@oregonstate.edu">Contact Alex Hoffer</a></p>
176                 </div>
177             </div>
178         </div>
179     </section>
180
181     <!-- jQuery -->
182     <script src="vendor/jquery/jquery.min.js"></script>
```

```
183
184     <!-- Bootstrap Core JavaScript -->
185     <script src="vendor/bootstrap/js/bootstrap.min.js"></script>
186
187     <!-- Plugin JavaScript -->
188     <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery-easing/1.3/jquery.easing.min.js"></script>
189     <script src="vendor/scrollreveal/scrollreveal.min.js"></script>
190     <script src="vendor/magnific-popup/jquery.magnific-popup.min.js"></script>
191
192     <!-- Theme JavaScript -->
193     <script src="js/creative.min.js"></script>
194
195 </body>
196
197 </html>
```

Listing 1: MachineLearnYourWayToMarchMadnessGlory.html. The graphical user interface that provides users information on how to engage with the module, a link to the module, and information about our project.

```
1  """
2  This tool
3  """
4  import pandas as pd
5  import math
6  from sklearn import model_selection, linear_model, svm, naive_bayes
7  from sklearn.naive_bayes import GaussianNB, MultinomialNB
8  import csv
9  import random
10 import sys
11 sys.path.append('./predicted_bracket_generator')
12 from bracket_builder import make_bracket
13
14 base_elo = 1600
15 team_elos = {}  # Reset each year.
16 team_stats = {}
17 X = []
18 y = []
19 submission_data = []
20 folder = 'input'
21 prediction_year = 2017
22 start_year = 2016
23
24 def calc_elo(win_team, lose_team, season):
25     winner_rank = get_elo(season, win_team)
26     loser_rank = get_elo(season, lose_team)
27
28     """
29     This is originally from from:
30     http://zurb.com/forrst/posts/An_Elo_Rating_function_in_Python_written_for_foo-hQl
31     """
32     rank_diff = winner_rank - loser_rank
```

```
    exp = (rank_diff * -1) / 400
    odds = 1 / (1 + math.pow(10, exp))
    if winner_rank < 2100:
        k = 32
    elif winner_rank >= 2100 and winner_rank < 2400:
        k = 24
    else:
        k = 16
    new_winner_rank = round(winner_rank + (k * (1 - odds)))
    new_rank_diff = new_winner_rank - winner_rank
    new_loser_rank = loser_rank - new_rank_diff

    return new_winner_rank, new_loser_rank


def initialize_data():
    for i in range(start_year, prediction_year+1):
        team_elos[i] = {}
        team_stats[i] = {}


def get_elo(season, team):
    try:
        return team_elos[season][team]
    except:
        try:
            # Get the previous season's ending value.
            team_elos[season][team] = team_elos[season-1][team]
            return team_elos[season][team]
        except:
            # Get the starter elo.
            team_elos[season][team] = base_elo
            return team_elos[season][team]


def predict_winner(team_1, team_2, model, season, stat_fields):
    features = []

    # Team 1
    features.append(get_elo(season, team_1))
    for stat in stat_fields:
        features.append(get_stat(season, team_1, stat))

    # Team 2
    features.append(get_elo(season, team_2))
    for stat in stat_fields:
        features.append(get_stat(season, team_2, stat))

    return model.predict_proba([features])

```

```
83
84  def update_stats(season, team, fields):
85      """
86  ␣␣␣␣This␣accepts␣some␣stats␣for␣a␣team␣and␣udpates␣the␣averages.
87  ␣␣␣␣First,␣we␣check␣if␣the␣team␣is␣in␣the␣dict␣yet.␣If␣it's␣not,␣we␣add␣it.
88  ␣␣␣␣Then,␣we␣try␣to␣check␣if␣the␣key␣has␣more␣than␣5␣values␣in␣it.
89  ␣␣␣␣␣␣␣␣If␣it␣does,␣we␣remove␣the␣first␣one
90  ␣␣␣␣␣␣␣␣Either␣way,␣we␣append␣the␣new␣one.
91  ␣␣␣␣If␣we␣can't␣check,␣then␣it␣doesn't␣exist,␣so␣we␣just␣add␣this.
92  ␣␣␣␣Later,␣we'll␣get␣the␣average␣of␣these␣items.
93  ␣␣␣␣"""
94      if team not in team_stats[season]:
95          team_stats[season][team] = {}
96
97      for key, value in fields.items():
98          # Make sure we have the field.
99          if key not in team_stats[season][team]:
100             team_stats[season][team][key] = []
101
102         if len(team_stats[season][team][key]) >= 9:
103             team_stats[season][team][key].pop()
104         team_stats[season][team][key].append(value)
105
106
107 def get_stat(season, team, field):
108     try:
109         l = team_stats[season][team][field]
110         return sum(l) / float(len(l))
111     except:
112         return 0
113
114
115 def build_team_dict():
116     team_ids = pd.read_csv(folder + '/Teams.csv')
117     team_id_map = {}
118     for index, row in team_ids.iterrows():
119         team_id_map[row['Team_Id']] = row['Team_Name']
120     return team_id_map
121
122
123 def build_season_data(all_data):
124     # Calculate the elo for every game for every team, each season.
125     # Store the elo per season so we can retrieve their end elo
126     # later in order to predict the tournaments without having to
127     # inject the prediction into this loop.
128     print("Building␣season␣data.")
129     for index, row in all_data.iterrows():
130         # Used to skip matchups where we don't have usable stats yet.
131         skip = 0
132
```

```
        # Get starter or previous elos.
        team_1_elo = get_elo(row['Season'], row['Wteam'])
        team_2_elo = get_elo(row['Season'], row['Lteam'])

        # Add 100 to the home team (# taken from Nate Silver analysis.)
        if row['Wloc'] == 'H':
            team_1_elo += 100
        elif row['Wloc'] == 'A':
            team_2_elo += 100

        # We'll create some arrays to use later.
        team_1_features = [team_1_elo]
        team_2_features = [team_2_elo]

        # Build arrays out of the stats we're tracking..
        for field in stat_fields:
            team_1_stat = get_stat(row['Season'], row['Wteam'], field)
            team_2_stat = get_stat(row['Season'], row['Lteam'], field)
            if team_1_stat is not 0 and team_2_stat is not 0:
                team_1_features.append(team_1_stat)
                team_2_features.append(team_2_stat)
            else:
                skip = 1

        if skip == 0:  # Make sure we have stats.
            # Randomly select left and right and 0 or 1 so we can train
            # for multiple classes.
            if random.random() > 0.5:
                X.append(team_1_features + team_2_features)
                y.append(0)
            else:
                X.append(team_2_features + team_1_features)
                y.append(1)

        # AFTER we add the current stuff to the prediction, update for
        # next time. Order here is key so we don't fit on data from the
        # same game we're trying to predict.
        if row['Wfta'] != 0 and row['Lfta'] != 0:
            stat_1_fields = {
                'score': row['Wscore'],
                'fgp': row['Wfgm'] / row['Wfga'] * 100,
                'fga': row['Wfga'],
                'fga3': row['Wfga3'],
                '3pp': row['Wfgm3'] / row['Wfga3'] * 100,
                'ftp': row['Wftm'] / row['Wfta'] * 100,
                'or': row['Wor'],
                'dr': row['Wdr'],
                'ast': row['Wast'],
                'to': row['Wto'],
                'stl': row['Wstl'],
```

```
                'blk': row['Wblk'],
                'pf': row['Wpf']
            }
            stat_2_fields = {
                'score': row['Lscore'],
                'fgp': row['Lfgm'] / row['Lfga'] * 100,
                'fga': row['Lfga'],
                'fga3': row['Lfga3'],
                '3pp': row['Lfgm3'] / row['Lfga3'] * 100,
                'ftp': row['Lftm'] / row['Lfta'] * 100,
                'or': row['Lor'],
                'dr': row['Ldr'],
                'ast': row['Last'],
                'to': row['Lto'],
                'stl': row['Lstl'],
                'blk': row['Lblk'],
                'pf': row['Lpf']
            }
            update_stats(row['Season'], row['Wteam'], stat_1_fields)
            update_stats(row['Season'], row['Lteam'], stat_2_fields)

        # Now that we've added them, calc the new elo.
        new_winner_rank, new_loser_rank = calc_elo(
            row['Wteam'], row['Lteam'], row['Season'])
        team_elos[row['Season']][row['Wteam']] = new_winner_rank
        team_elos[row['Season']][row['Lteam']] = new_loser_rank

    return X, y


if __name__ == "__main__":
  #choose model
    if len(sys.argv) > 1:
      if sys.argv[1] == '1':
        print("You choose linear logistic regression to be the model")
        model = linear_model.LogisticRegression()
      elif sys.argv[1] == '2':
        print("You choose svm rbf to be the model")
        model = svm.SVC(gamma=0.001, C=100., probability=True, kernel='rbf')
      elif sys.argv[1] == '3':
        print("You choose svm linear to be the model")
        model = svm.SVC(gamma=0.001, C=100., probability=True, kernel='linear')
      elif sys.argv[1] == '4':
        print("You choose svm poly to be the model")
        model = svm.SVC(gamma=0.001, C=100., probability=True, kernel='poly')
      elif sys.argv[1] == '5':
        print("You choose svm sigmoid to be the model")
        model = svm.SVC(gamma=0.001, C=100., probability=True, kernel='sigmoid')
        elif sys.argv[1] == '6':
            print("You choose Niave Bayes Gaussian to be the model")
```

```
            model = GaussianNB()
        elif sys.argv[1] == '7':
            print("You choose Niave Bayes multinomial models to be the model")
            model = MultinomialNB()
        else:
            model = linear_model.LogisticRegression()
            print("You choose linear logistic regression to be the model")
    else:
      model = linear_model.LogisticRegression()
      print("You choose linear logistic regression to be the model")

    #choose fields
    if len(sys.argv) > 2:
      stat_fields = sys.argv[2:]
      print("You choose ", str(sys.argv[2:]), " as the feature to be considered")
    else:
      stat_fields = ['score', 'fga', 'fgp', 'fga3', '3pp', 'stl']
      print("You choose 'score', 'fga', 'fgp', 'fga3', '3pp', 'stl' as the feature to be considered")

      #stat_fields = ['score', 'fga', 'fgp', 'fga3', '3pp', 'ftp', 'or', 'dr',
      #'ast', 'to', 'stl', 'blk', 'pf']

    initialize_data()
    season_data = pd.read_csv(folder + '/RegularSeasonDetailedResults.csv')
    season_data = season_data[season_data['Season']>=start_year]
    tourney_data = pd.read_csv(folder + '/TourneyDetailedResults.csv')
    tourney_data = tourney_data[tourney_data['Season']>=start_year]
    frames = [season_data, tourney_data]
    all_data = pd.concat(frames)

    # Build the working data.
    X, y = build_season_data(all_data)

    # Fit the model.
    print("Fitting on %d samples." % len(X))

    # Check accuracy.
    print("Doing model selection.")
    print(model_selection.cross_val_score(
        model, X, y, cv=10, scoring='accuracy', n_jobs=-1
    ).mean())

    model.fit(X, y)

    # Now predict tournament matchups.
    print("Getting teams.")
    seeds = pd.read_csv(folder + '/TourneySeeds.csv')
    # for i in range(2016, 2017):
    tourney_teams = []
    for index, row in seeds.iterrows():
```

```
283         if row['Season'] == prediction_year:
284             tourney_teams.append(row['Team'])
285 
286     # Build our prediction of every matchup.
287     print("Predicting matchups.")
288     tourney_teams.sort()
289     for team_1 in tourney_teams:
290         for team_2 in tourney_teams:
291             if team_1 < team_2:
292                 prediction = predict_winner(
293                     team_1, team_2, model, prediction_year, stat_fields)
294                 label = str(prediction_year) + '_' + str(team_1) + '_' + \
295                     str(team_2)
296                 submission_data.append([label, prediction[0][0]])
297 
298     # Write the results.
299     print("Writing %d results." % len(submission_data))
300     with open(folder + '/submission.csv', 'w') as f:
301         writer = csv.writer(f)
302         writer.writerow(['id', 'pred'])
303         writer.writerows(submission_data)
304 
305     # Now so that we can use this to fill out a bracket, create a readable
306     # version.
307     print("Outputting readable results.")
308     team_id_map = build_team_dict()
309     readable = []
310     less_readable = []  # A version that's easy to look up.
311     for pred in submission_data:
312         parts = pred[0].split('_')
313         less_readable.append(
314             [team_id_map[int(parts[1])], team_id_map[int(parts[2])], pred[1]])
315         # Order them properly.
316         if pred[1] > 0.5:
317             winning = int(parts[1])
318             losing = int(parts[2])
319             proba = pred[1]
320         else:
321             winning = int(parts[2])
322             losing = int(parts[1])
323             proba = 1 - pred[1]
324         readable.append(
325             [
326                 '%s beats %s: %f' %
327                 (team_id_map[winning], team_id_map[losing], proba)
328             ]
329         )
330     with open(folder + '/readable-predictions.csv', 'w') as f:
331         writer = csv.writer(f)
332         writer.writerows(readable)
```

```
333     with open(folder + '/less-readable-predictions.csv', 'w') as f:
334         writer = csv.writer(f)
335         writer.writerows(less_readable)
336 
337     print("Generating pictures.")
338     generatorPath = "./predicted_bracket_generator/"
339     DATAPATH = generatorPath+"data/"
340     submissionPath = folder+'/submission.csv'
341     emptyBracketPath = generatorPath+'/empty_bracket.jpg'
342     outputFilePath = 'predicted_bracket.jpg'
343 
344     make_bracket(DATAPATH, submissionPath, emptyBracketPath, outputFilePath)
```

Listing 2: mm.py, where basketball statistics are read and fed into Scikit-Learn machine learning algorithms. Then, the resulting March Madness bracket is generated.