Final Project Proposal

Year: 2022 **Semester:** Spring **Project Name:** March Madness Perfect Bracket

Creation Date: 11/1/2021 **Last Modified:** 1/19/2022

Team Members:

Tycho Halpern Email: thalper@purdue.edu

Luca Rivera Email: river172@purdue.edu

Rylee Benes Email: rbenes@purdue.edu

Repository:

https://github.com/thalper/March-Madness

Description of Problem:

Warren Buffet advertised a $1,000,000,000 prize for completing a perfect bracket during March Madness[1], and since then, every year millions of people have attempted to win the prize. With 63 games ahead (plus play in games) there is a less than $1 / 2^{63}$ chance of a random bracket being perfect. Apart from creating the perfect bracket, millions more people compete against their friends each year to see who can create the best bracket of their friends. Other attempts have been made to analyze matchups and predict games, but we have found that none of the existing open-source models are able to make predictions better than average.

Proposed Solution:

We want to create an AI algorithm that creates a better and consistent win percentage over any of its "bracket-picking" competitors. We plan on creating an open-source Python project to develop a neural network that looks at very broad to very specific March Madness statistics, creating training and testing data to develop the most accurate algorithm as a combination of those stats. Some examples of these statistics include a team's win percentage, conference standings, tournament distance from a university, and fan involvement. By finding the best combination of these statistics, the grouping of the ones with the greatest win percentages, we should be able to create an AI that does better than its competitors. We plan on breaking our project up into multiple sections to arrive at a final deliverable. We will start by gathering relevant data from https://www.kaggle.com/andrewsundberg/college-basketball-dataset as .csv files. We will then manually calculate a few more data points for each team, such as proximity of a school's city or fan involvement. We will read in the data in our main program. From there, we plan on breaking up the data into odd and even years, using half for training data and half for testing data. We will create our final algorithm by taking the best results from this method. Finally, we plan on presenting our final bracket in a readable graphic. We will be using the open-source library TensorFlow[3] for the machine learning aspect of the neural network, and h5py[4] to load and store weights for the model. For organizing data in the project, we will be using the NumPy library.

After the model is complete, we will create an interactive program that displays the predicted bracket as well as showing more details about each game.

Success Criteria:

Data Collection and Storage:

Specific: Gather complete data for each of the 68 teams in the 2021-22 tournament, as well as complete data for every team that participated in the tournament in previous years.
Measurable: The data will match the season data from https://barttorvik.com/.
Assignable: The data will exist in a folder at the root level.
Realistic: We have data for previous years and need to add data for the current year.
Trackable: The data will exist in /Previous.

Data Pipelining:

Specific: We will be able to read data from .csv files into NumPy arrays in the main program.
Measurable: View portions of the array to verify it matches with the .csv data files.
Assignable: Print portions of the array as a separate module.
Realistic: Importing data from .csv to NumPy arrays has been done before.
Trackable: We will use a PrintTable() function to view the data in the main program.

Neural Network:

Specific: Create a usable, modifiable model that can take the collected data to accurately predict game outcomes better than competing software.
Measurable: The weights added by the neural network to different game statistics help in predicting the appropriate winner of a matchup.
Assignable: The data will exist in a folder at the root level.
Realistic: Most of the models we have found are quite simple and not overly complex. We hope that our model will consider much more data, making it better than competing software.
Trackable: The weights created by our neural network functions impact each matchup in a beneficial way.

Deliverable Model:

Specific: Show a decipherable diagram that presents the final bracket similar to how brackets are filled out on paper or online. A complete bracket will be a possible outcome of the tournament.
Measurable: The bracket diagram is comparable to any March Madness bracket filled out on paper or online.
Assignable: The data will exist in /Final.
Realistic: We have examples of what a completed bracket should look like.
Trackable: This criterion will be met if our functions used to visualize a bracket look similar to brackets created on paper or online.

Bracket Success A:

Specific: We are setting specific thresholds for the generate brackets and the outcome we expect them to have: 60% of generated brackets to perform above 50^{th} percentile
Measurable: We can identify how close or how far off we were from the threshold we set.

Assignable: We can compare the score of our brackets with the rest of the entries.
Realistic: We researched how common it was to be on the percentiles specified and we believe that this.
Trackable: This criterion will be met if we manage to get all the brackets in the percentile we have specified.

Bracket Success B:

Specific: We are setting specific thresholds for the generate brackets and the outcome we expect them to have: 30% of generated brackets to perform above 75th percentile
Measurable: We can identify how close or how far off we were from the threshold we set.
Assignable: We can compare the score of our brackets with the rest of the entries.
Realistic: We researched how common it was to be on the percentiles specified and we believe that this.
Trackable: This criterion will be met if we manage to get all the brackets in the percentile we have specified.

Bracket Success C:

Specific: We are setting specific thresholds for the generate brackets and the outcome we expect them to have: 10% of generated brackets to perform above 90th percentile
Measurable: We can identify how close or how far off we were from the threshold we set.
Assignable: We can compare the score of our brackets with the rest of the entries.
Realistic: We researched how common it was to be on the percentiles specified and we believe that this.
Trackable: This criterion will be met if we manage to get all the brackets in the percentile we have specified.

ECE49595O Course Requirements Satisfaction and Specifications:

We plan to create an open-source Python project that uses a neural network for developing our bracket. We will be using our prior knowledge from ECE 473: Introduction to Artificial Intelligence as a basis for our algorithm. We want to advance our knowledge by utilizing TensorFlow to create a more involved and efficient neural network. All of this will be done from scratch, using past March Madness data, in hopes of testing our algorithm in real-time in March 2022.

Market Analysis:

The market for our project is people who are interested in taking part in the competition of the $1,000,000,000 prize of March Madness. Besides the people who take part in the march madness bracket prediction, thousands of people bet on college basketball games. What this means is that not only the people interested in the March Madness prediction are our target but also everyday college basketball fans.

Competitive Analysis:

We looked at other open-source attempts to predict March Madness outcomes and have found that none of them are satisfactory. The neural networks they use take very few inputs, and usually only look

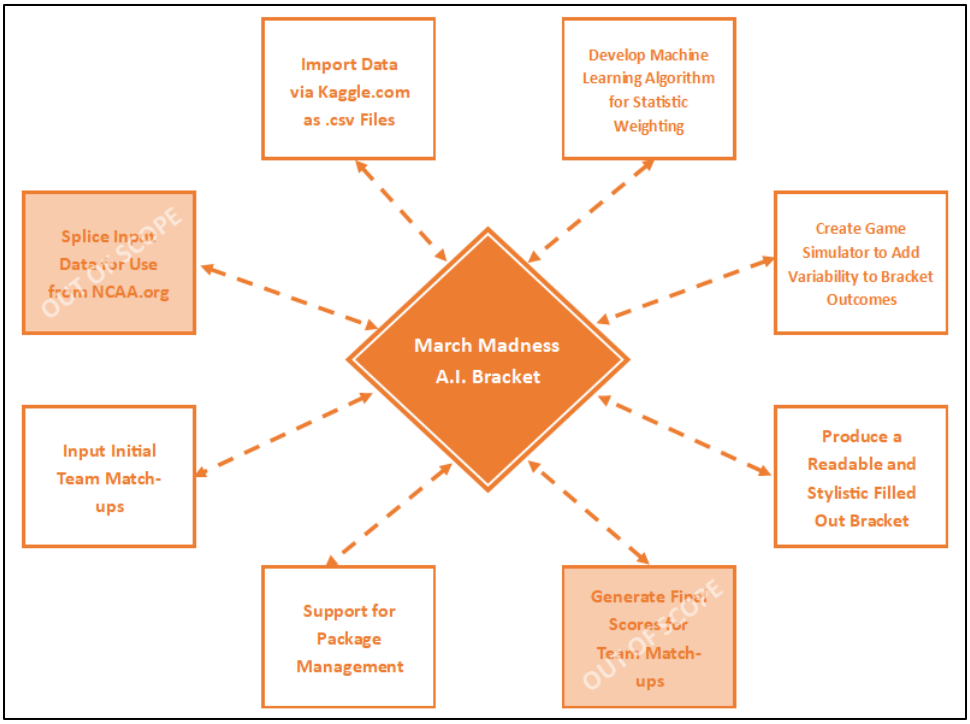
at win-loss percentage, seed numbers, and maybe average points per game and average points against per game. For example, one open-source predictor uses Dungeons & Dragons battle logic[5] to determine which team will win each game. Other, better software exists for predicting brackets, but it does not exist in an open source form[6].

Open-Source Components:

The open-source software we will be using includes:

TensorFlow for neural network training,
h5py to store neural network data,
importing data from kaggle.com,
NumPy for data storage and manipulation,
pipenv for package management,
and an open-source bracket generator to generate the final output of the bracket.

Component Diagram:



References:

Billion-dollar bracket challenge:

1. https://bleacherreport.com/articles/1931210-warren-buffet-will-pay-1-billion-to-fan-with-perfect-march-madness-bracket

Data Collection:

2. https://www.kaggle.com/andrewsundberg/college-basketball-dataset

Neural Network:

3. https://www.tensorflow.org/
4. https://www.h5py.org/

Competing software:

5. https://github.com/btelle/dnd-march-madness
6. https://unanimous.ai/sports-march-madness