NCAA Bracket Predictor

Sprint 1

Team Members:

Kevin Brosam, Nate Lang, John Hattas, Alex Berkhout, Matt Petter

**User Stories:**

Kevin wants to win one billion dollars. Kevin has some picks that he is not sure about and needs something to do a more detailed analysis. He really values three-point percentage over points in the paint. By using this program Kevin can get a more detailed analysis of who will win the game.

Nate goes to the website and wants to generate a bracket to win a lot money. Nate goes to the website and generates a bracket. The website informs the user of all stats of each team and compares them to their opponent. Nate thinks the generated bracket looks fine, but he believes that one team that the website has winning will lose. Nate changes that one team around in his bracket and runs the generation again, except he manually picks the one victory. He then uses the bracket that was generated and wins a billion dollars.

John knows nothing about basketball but is looking for something to help guide his picks. He wants indicators to show which team is more likely to win based on which is picked. It generates a bracket based on the indicators. John is confident about some games and wants to override the picks. The bracket will regenerate the bracket based on the new picks. John wants to win a billion dollars.

Alex wants to compare his bracket with predictions based off pure statistics. He would like to check specific match ups to see which team the system thinks are more likely to win and use it to help him help him decide his picks. Upon looking at the computer-generated bracket, he wants to see the predicted results for each round and be able to change some settings to potentially see different outcomes.

Kevin is trying to get an idea of who will win each game in the tournament. He prefers near home teams but does not want to look up how far each team is from each site and compare. He wants a bracket picker that factors in how far each team is from each game site.

Nate wants to compare a multitude of different options and wants an easy way to compare them. He is curious about how what he prefers stacks up against other ways of predicting the bracket. Nate also wants it to display the data in easy to read graphes.

**Task Cards:**

* Convert data into usable CSV files
* Create a git library and get the whole team able to pull and commit
* Get whole team using Jupyter Notebooks and Anaconda
* Write a program that uses the CSV files correctly
* Randomly pick a bracket, no indicators or bias
* Display the bracket in an easy to understand way
* Modify existing code to have a more precise method of picking winners
* Back test with different methods to hopefully find one that can predict the winners from previous years
* Add point system and algorithm evaluation through number of games correct, %, etc.

**Tests:**

* Have everyone commit to the git
* Have everyone open code using Jupyter Notebooks
* CSV file data is accurately read in, team names match ranks
* Bracket has correct teams playing each other
* Bracket displays in the correct format
* Bracket picks winners based off an indicator rather than randomly

**Sprint Backlog:**

|  |  |  |
| --- | --- | --- |
| Task | Priority [1-10 (1 being lowest)] | Completed(Y/N) |
| Randomly Generated Bracket | 8 | Y |
| Display Bracket to User (In bracket form) | 3 | N |
| Generate bracket based on basic algorithm | 5 | Y |
| Collect Statistics for every team into CSV files | 9 | Y |
| Add point counting system / success rate prediction | 4 | Y |

**Product Backlog:**

|  |  |  |
| --- | --- | --- |
| Task | Priority [1-10 (1 being lowest)] | Completed(Y/N) |
| Develop an algorithm that predicts previous tournaments results | 1 | N |
| Integrate more advanced statistics | 7 | N |
| Create picture of the bracket with appropriate teams | 7 | N |
| Collect Data | 10 | Y |
| Have a basic working model | 10 | Y |
| Back test for better prediction results | 4 | N |
| Potentially display through HTML | 1 | N |
| Update for 2018 tournament | 3 | N |
| Compare different basic algorithms to find the easiest while not losing accuracy | 8 | N |
| Display data in charts and tables, potentially using R | 5 | N |
| Add location as one of the indicator | 2 (if reasonably possible) | N |
| Create User Interface | 4 | N |