DS210 Final Project

Anna Rozinsky

Introduction

I am fascinated about the intersection between sports and data science. I have played the EA Sports NHL video game since I was little, and growing up I was always curious about how each player got their rating out of 100 for every season the video game was released. This project gave me the opportunity to research their method, and apply what I learned in class to create my own very basic rating system. My goal was to construct a system that would process a player, and based on what position(s) they played give them an overall rating out of 100.

I had to do some research prior to building this model, as I wanted to keep this model fairly simple and decided to choose the top five traits that each position should have. I split players into three positions - wingers, centers, and defensemen. Wingers play the most offense, I chose their traits to showcase their offensive talent, and typical physical style of play. Wingers are typically large and powerful players, so the following traits best numerically describe those attributes: Goals, SH% (which is percentages of shots taken that result in a goal), Rush attempts (entering the offensive zone), Total points, and Hits. Centers dictate the flow of the game, and are important both offensively and defensively. Therefore, I chose the following traits to showcase playmaking ability: Faceoff %, Total Points, Takeaways, First Assists, and IPP (which is a measurement of how often a team scores when a player is present on the ice, even if not directly involved in the goal). Finally, the attributes for a defenseman rested heavily on their ability to defend their own net, and also their play making from their opposing blue line: Hits, Shots blocked, Takeaways, Total Points, and Rush attempts.

I decided that my approach to the model would be to normalize all of the values and use a sigmoid function to calculate the ratings. This way outliers would hopefully not skew the data, and each attribute of the different positions could be weighted depending on how relevant each trait is to each position.

Code Construction

I split my code into two modules to ensure efficiency. My first module is called cleaning.rs. It begins with any necessary imports, and then creates public structs and enums. The struct creates a hashmap that represents a player with their name, their position(s) and any metrics needed to measure their performance. The enum defines each player's position (Center, Wing, and/or Defense). The first function in the cleaning.rs opens the imported csv file, parses through each line, and extracts the values for each player. It skips over the header line which contains all of the column labels. Each player is then interested in the “players” map. The next function is called clean fields. This function searches through each line of the csv file and checks the validity of the data. I added this as there were some issues with players not having certain variables accounted for, and if any data is missing it skips over that line and reports it when the code is run. This function also assigns each player into a hashmap based on their position and necessary metrics. The next function is default metric is a helper function that returns 0.0 as a default value for

when a metric cannot be parsed from the data. This was included after I realized that a decent amount of the players were missing values. Because I wanted them to be included, I created this function so that it would fill in any empty “cells” in the imported csv file. This allowed all players to be accounted for when calculating their individual ratings. The final function in this module is to normalize metrics. In order to make sure that there were no outliers, I decided to normalize each of the metrics on a scale of 0 to 1. This finds the maximum value for each metric, and divides each player’s metric by that value with the assumption that all metrics are finite.

I then created the main.rs module. In this model, I imported several standard library modules like HashMap, File and I/O related modules to help read the file that I was importing. It also imports my cleaning.rs file so that it can be used in the main.rs module. The first function is called calculate score. It computes a player's score based on their position, and depending on the metric there is a weight. If a player is categorized as a center, then the weights are applied to the normalized values for each of the categories of the position. For Centers, the metrics are Faceoffs, Total Points, Takeaways, First Assists, and IPP. Their weights in order on a scale of 0 to 1 are 0.25, 0.3, 0.15, 0.2, 0.1. For Wingers, the metrics are Goals, SH%, Rush Attempts, Total Points, and Hits. Their weights are 0.35, 0.25, 0.15, 0.2, 0.05. For Defense, the metrics are Hits, Shots Blocked, Takeaways, Totals Points, and Rush Attempts. Their weights are 0.15, 0.3, 0.2, 0.2, 0.15. These weights were calculated based on how valuable I personally felt each of the metrics were to players' performance, and I took into account how valuable these metrics were in the research I conducted prior to beginning the project when I was selecting the metrics themselves for each individual position. The final rating of each of the players is scaled using a logistic sigmoid function, which maps the score to a value between 0 and 100. The reason I chose the sigmoid function to model the data was because its an ideal model for showcases probabilities and proportions. Sigmoid functions are often used in neural networks and other nonlinear classification models, as it represents the probability of a sample belonging to a certain class in a nonlinear fashion, which helps stabilize predictions for large outputs and takes into account depth for certain models. In this context, this sigmoid function ensures that the ratings of each of the players can be interpreted as a percentage, and ensures that the extreme weighted sums dont yield extreme outputs. It also ensures that players with different metrics can still be compared on a similar scale.
The main function of the program first reads the csv file that I have imported from Natural Stat Trick, which includes data for every skater (so the data excludes goalies) that played in an NHL game between the 2021-22 season and the 2023-24 season. Statistics include 5v5 and penalty scenarios (power play or penalty kill). It then applies the clean fields to extract all of the necessary player data and will print out if it missed any rows to see if any corrections needed to be made. It then applies the normalize metrics method to all of the hashmaps to normalize all of the metrics for each of the players. It then groups each player based on their position, and then sorts them based on their score within their position. The code is then prompted to print out the top ten players for each position. The code also provides an interactive loop that will prompt the

user to input a players name, regardless of capitalization, and outputs the players name, position, and their normalized metrics. The loop will continue to prompt the user until enter is pressed with an empty string as the input, in which case it will exit out of the loop.

I also conducted four tests for the code to ensure the main functions implemented throughout the code were operating correctly. The first made sure that the code was cleaning and sorting the data correctly. The second ensured that the code was properly normalizing metrics with two players so that the code would take the max between the two players and properly normalize them. The next test ran the calculate score function to test if the scores are being properly calculated for a certain position, and that the output is between 0 and 100. The final test ensures that the header is being excluded from the processing of the csv, as it is optimal for the code to skip over the header instead of trying to parse through all of the column titles when they are not useful. All of the tests in the code pass, which affirms the correctness of the code from those functions. This ensures that all of the data cleaning and normalization is running as it should.

Results

The results of this code were compared against the EA sports data that I was referring to in the beginning of this report. Of all the positions, I would argue that centers were the best predicted as there are clear markers as to what makes a center a good center. For the other two positions, I would say I either did not select the correct metrics to measure or the position is more nuanced and there is some work to be done to investigate further. Defensmen have the hardest measurement, as I am only selecting five metrics to measure their abilities. I chose defensive statistics, however defensive players are also responsible for being productive on offense as well. That was not very clearly displayed when choosing only five metrics to measure. I also think that the scores are a little inflated, however overall I think they are reflective of overall performance and rating compared to other players. This project was an interesting investigation into player ratings, and I would further investigate and attempt to improve this algorithm further.

Sample Output:
Top Players in Center Position:
Connor McDavid: 96.31%
Leon Draisaitl: 94.45%
Claude Giroux: 93.51%
Tim Stützle: 92.54%
Tage Thompson: 92.34%
Nathan MacKinnon: 92.18%
Mitchell Marner: 92.08%
Auston Matthews: 91.84%
Phillip Danault: 91.56%
Pierre-Luc Dubois: 91.40%

Top Players in Wing Position:
Alex DeBrincat: 97.69%
Rickard Rakell: 96.51%
Viktor Arvidsson: 91.28%
David Pastrnak: 76.84%
Nikita Kucherov: 76.41%
Matthew Tkachuk: 75.78%
Mikko Rantanen: 75.55%
Brady Tkachuk: 74.99%
Artemi Panarin: 74.87%
Kirill Kaprizov: 74.58%

Top Players in Defense Position:
MacKenzie Weegar: 96.90%
Moritz Seider: 96.65%
Alex Pietrangelo: 96.41%
Cale Makar: 96.39%
Adam Fox: 96.21%
Darnell Nurse: 96.14%
Jacob Trouba: 95.95%
Colton Parayko: 95.66%
Noah Dobson: 95.60%
Kris Letang: 95.52%

Enter a player name to get their score (or press Enter to exit):
Travis Konecny
Player: Travis Konecny

Stats for Travis Konecny at Center:
Faceoffs %: 0.06
Total Points: 0.07

Takeaways: 0.34
First Assists: 0.53
IPP: 0.64

Current Rating: 78.40%

Enter a player name to get their score (or press Enter to exit):
Trevor Zegras
Player: Trevor Zegras

Stats for Trevor Zegras at Center:
Faceoffs %: 0.02
Total Points: 0.35
Takeaways: 0.35
First Assists: 0.26
IPP: 0.26

Current Rating: 76.80%

Enter a player name to get their score (or press Enter to exit):

Exiting...