Using a Dense Neural Network to Identify the Ideal Pitcher from the New York Yankees' Roster against 2021 MLB Teams

Anthony Williams and Tyler Eischens
Stanford University
450 Serra Mall, Stanford, CA 94305
tonyw5@stanford.edu, tyeisch@stanford.edu

Abstract

Our project for CS230 Spring 2022 focused around baseball. Specifically, we took all of the game data from the 2021 season and converted it into a format readable by a Dense Neural Network. We then trained multiple iterations of a Dense Neural Network on that modified data. The goal was to create a network that could take as input a pitcher's pitching statistics with the team they were playing against, and return how that team would perform against them. Our inputs to the network were specific pitchers frameworks, including the following stats: fastball percentage, fastball velocity, slider percentage, cutter percentage, curveball percentage, curveball velocity, changeup percentage, changeup velocity, sinker percentage, knuckleball percentage, junk pitch percentage, age, ERA, and hits, as well as the current team that pitcher is playing against. The output data was the offensive stats of the team. There were 15 outputs, with 9 of them being the runs scored each inning, and the remaining 6 being at-bats, hits, home runs, walks, team earned runs, and individual earned runs. After we were satisfied with the model we created, we then proceeded to utilize that model with the 2022 New York Yankees bullpen. For any given member of their pitching team, we could plug in their stats as well as the team they would play against, and in return receive an estimate of how the opposing team would perform. This allowed us to simulate which pitcher the New York Yankees should start against any given team for the best outcome.

Introduction:

The New York Yankees are one of the most storied baseball franchises to exist in the United States. In the midst of their 120th season, they currently have 27 MLB World Championships [1]. While the average viewer might say that the chances any team is going to win a baseball game on any given day is a 50/50 split, analysts use vast amounts of data from previous games to determine which team has the better shot of winning. This plays into things like sports betting. We decided that we wanted to base our project in this area. However, the aim with our project is not to identify the team that has the best odds of winning that day in order to help out gamblers. Using techniques discussed in this class, namely a Dense Neural Network, the goal of our project was to evaluate the games of every MLB team from the 2021 season to find out the type of pitcher that they struggled against the most. Then, using the model we trained with all of the 2021 season data from each team in the MLB, we found the ideal New York Yankee pitcher against any given team.

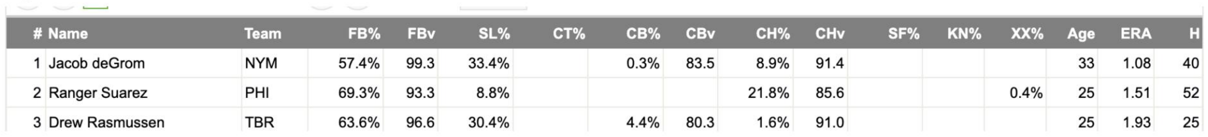
Related Work:

There are a respectable number of works published in the baseball sphere that have to do with Deep/Machine Learning. This is due to the vast and well kept data, from score cards, going back to the late 19th century. As seen on [6], there is no short supply of subjects to analyze in Baseball. Take for example this article by John Pette [5]. He took data from the 2015-2020 seasons, and using various regression techniques, predicted offensive stats of individual players such as runs, homeruns, RBIs, and steals [5]. This article from 2018 by Micah

Melling talks about the use of defensive shift heat maps in order to predict which player is at bat [7]. This article is interesting because in opposition to ours, where we are studying the pitcher, this one studies every other player on defense. In an area extremely similar to our proposed project, this article from 2018, also by Micah Melling, discusses the probability that a pitcher will surrender a run in a given situation [8]. After inputting multiple fields of data, the model will output the probability of the pitcher giving up a run to the batter. This model works in a similar way to ours, by not putting importance on the name of the pitcher, but simply their stats, and the team they are pitching against [8]. Other works seen in [6] include World Series predictions, Player Decline predictions, and even Cy Young Award Voting predictions. The majority of currently published works in the area take all of the available data and use it to predict wins and losses. We hope that our process and model can evolve into a useful tool for coaches nationwide to utilize their best weapons(pitchers).

Dataset and Features:

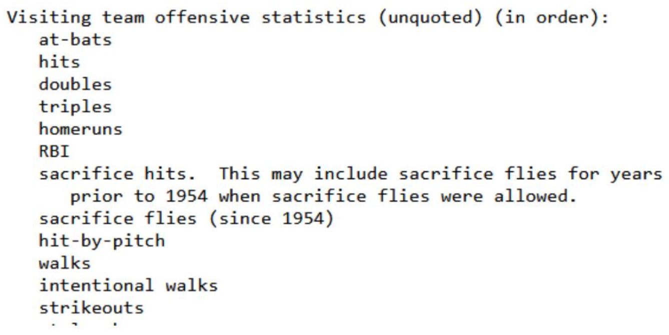
Our approach to using our dataset is to combine two separate datasets into our inputs X and our outputs Y. The input X of our dataset will be the starting pitcher framework - the percentage breakdown and velocity of his pitches, as well as useful pitching metrics and age - as well as their opponent for a respective game. We pulled our pitching data from [9]. Specifically, the data features included are: fastball percentage, fastball velocity, slider percentage, cutter percentage, curveball percentage, curveball velocity, changeup percentage, changeup velocity, sinker percentage, knuckleball percentage, junk pitch percentage, age, ERA, and hits/9in. Below, you can see an example of how some of those statistics look before we modify them to fit inside our model. For the team name being played against, we created a dictionary that mapped the team name to a number, as well as forgoing the current pitchers' name and team, as it is not important in training our model. For all missing values, originally string 'nan', they were replaced with 0.0 and all percentages, originally strings , to their respective decimal values.



#	Name	Team	FB%	FBv	SL%	CT%	CB%	CBv	CH%	CHv	SF%	KN%	XX%	Age	ERA	H
1	Jacob deGrom	NYM	57.4%	99.3	33.4%		0.3%	83.5	8.9%	91.4				33	1.08	40
2	Ranger Suarez	PHI	69.3%	93.3	8.8%				21.8%	85.6			0.4%	25	1.51	52
3	Drew Rasmussen	TBR	63.6%	96.6	30.4%		4.4%	80.3	1.6%	91.0				25	1.93	25

Figure 1: The original representation of the pitcher stats

The output Y consists of some notable offensive statistics from that game. The output Y contains offensive performance stats such as hits, runs, walks, and homeruns.



```
Visiting team offensive statistics (unquoted) (in order):
   at-bats
   hits
   doubles
   triples
   homeruns
   RBI
   sacrifice hits.  This may include sacrifice flies for years
      prior to 1954 when sacrifice flies were allowed.
   sacrifice flies (since 1954)
   hit-by-pitch
   walks
   intentional walks
   strikeouts
```

Figure 2: The original representation of offensive stats of the teams in the given game

The dataset that we are gathering the game statistics from comes from Retrosheet.org, and, as visible in the picture above, contains a vastly larger amount of data than we need for our model to be correctly trained [2]. The unfiltered dataset contains over 140 categories, but for our output Y we will have 15 features. 9 of those features are the by inning box scores, with the remaining 6 being at-bats, runs, homeruns, walks, team earned runs, and individual earned runs. We decided to limit the number of outputs so drastically from the statistics given because a lot of the offensive results are not a reflection of the pitcher, but of how the rest of the team operates.

Methods:

The approach to our project is not super complicated, considering we do not need to use images in training our model, just data. However, our project is going to be implemented in two different stages, as noted above. The first stage in our project was training a model to identify how different teams would perform against certain types of pitchers. The model we implemented for this part of the project was a Dense Neural Network. A Dense Neural Network is one of the more simple models we learned in class this year, but is extremely effective. It operates by having multiple layers of neurons that, over lots of training iterations, begin to learn how to map the input to the output. Each of the neurons learns to identify a particular feature in the input and how it pertains to the other features, respectively. We believe that this is the most efficient network for this particular task due to the fact that our data does not involve images, it is just pure statistics. The loss function that we plan on using for our project is a MSLE loss function. This works here because the data we are trying to map from X to Y is not binary and instead is finding the right balance of a large vector output. The Dense Neural Network we plan on implementing will vary from the ones we worked on in class because it is not going to be used as a classifier. Our output Y will have 15 values to be estimated instead of a single binary value. After training our model we implement the second stage of our project. Once we trained the model using the data from the 2021 MLB season, we then took the current roster of the 2022 New York Yankees and used each pitcher in their bullpen as an input for the model. The output from this will be discussed in more detail in the next section.

Experiments/Results/Discussion:

The initial plan was to create a multi layer Dense Neural Network with our still up in the air. After multiple iterations of multi layer Dense Neural Networks, we ran in to two main problems. The first being that, depending on the loss function, our model either performed horribly, and the second being that the model seemed to think it had 100% accuracy, as shown in the graph below.

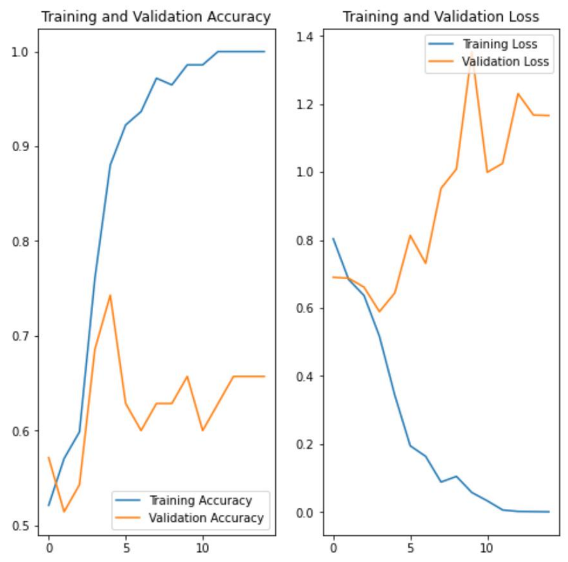


Figure 3: These graphs show highly improbable results.

This was with a 2-layer network with a Mean Square Error loss function. We spent a lot of time optimizing the batch size as well, and finally settled on 40. In the end, the model we settled on was simply a single layer Dense Neural Network, directly mapping the inputs to 15 output neurons, the same number of features in our output. We received on average close to 86% accuracy on both training and validation sets. The 'mean squared error' loss function was used since our values were not uniformly situated between any two numbers. The only thing that mattered was they were positive. The optimizer we used was an 'Adam' optimizer. You can see the results from our final model below.

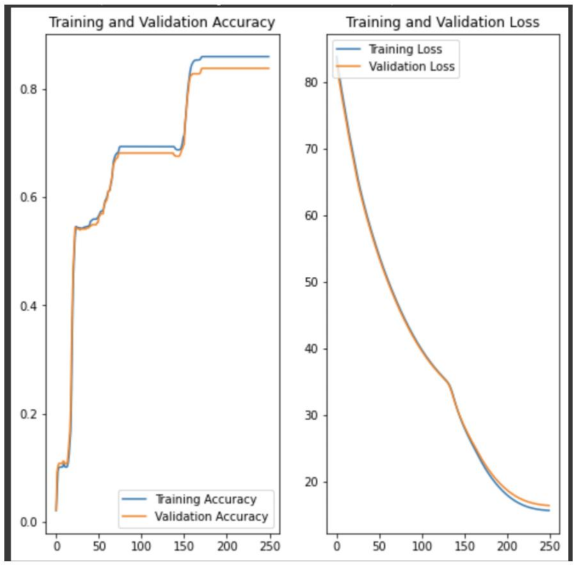
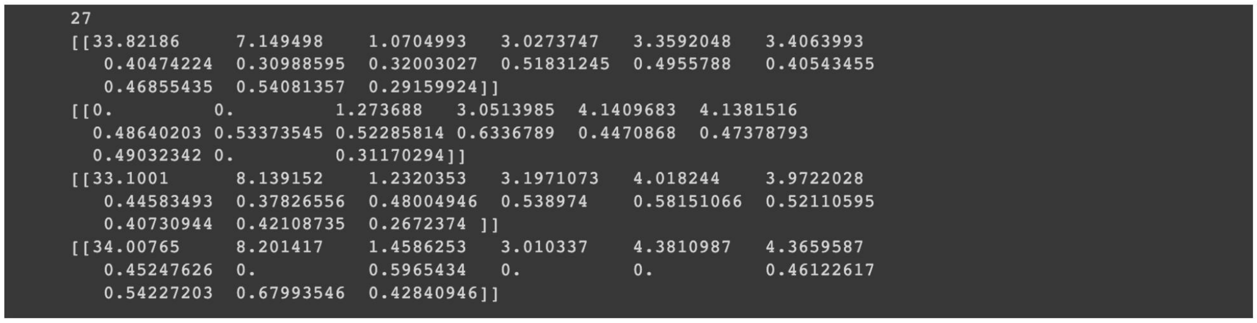


Figure 2: The training and validation sets going up against our model with 250 epochs.

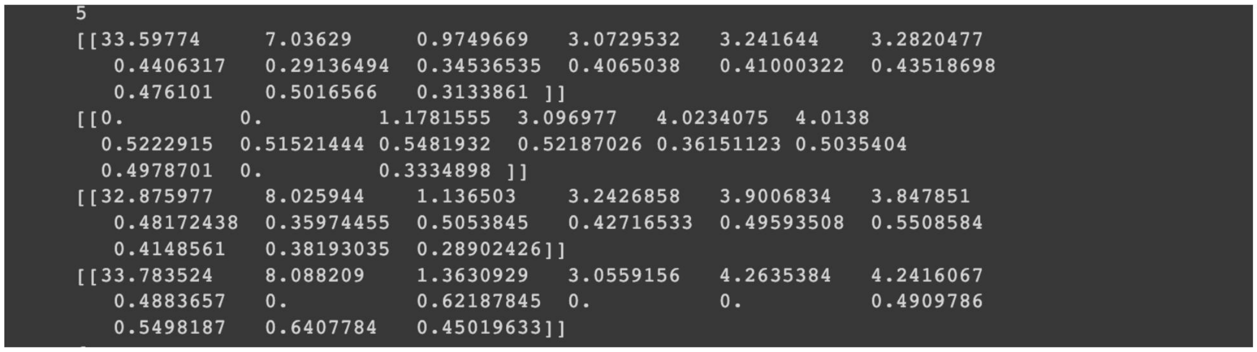
One notable occurrence is with the box score output. Instead of being whole numbers, the box scores were arranged such that a higher number in the location meant a higher chance of a run being scored that inning. This was an outcome we did not foresee but was extremely interesting to observe.

Using our model to predict the game's outcome from pitcher's statistics allows us to use said model to find out how the Yankees current pitchers would hold up against these teams comparatively. We specifically use our model to enter the yankees current top 4 pitchers against multiple teams. Some results can be seen below:



```
27
[[33.82186     7.149498    1.0704993   3.0273747   3.3592048   3.4063993
   0.40474224  0.30988595  0.32003027  0.51831245  0.4955788   0.40543455
   0.46855435  0.54081357  0.29159924]]
[[0.         0.         1.273688   3.0513985  4.1409683  4.1381516
  0.48640203 0.53373545 0.52285814 0.6336789  0.4470868  0.47378793
  0.49032342 0.         0.31170294]]
[[33.1001      8.139152    1.2320353   3.1971073   4.018244    3.9722028
   0.44583493  0.37826556  0.48004946  0.538974    0.58151066  0.52110595
   0.40730944  0.42108735  0.2672374 ]]
[[34.00765     8.201417    1.4586253   3.010337    4.3810987   4.3659587
   0.45247626  0.          0.5965434   0.          0.          0.46122617
   0.54227203  0.67993546  0.42840946]]
```

The four arrays represent the Yankees' top three pitchers: Gerrit Cole, Jordan Mongomery, Nestor Cortes, and Jameson Taillon. The array containing the smallest numbers is predicted to give that team the hardest time. Team 27 represents Pittsburg Pirates and according to these arrays Cole is the best pitching option..



```
5
[[33.59774     7.03629     0.9749669   3.0729532   3.241644    3.2820477
   0.4406317   0.29136494  0.34536535  0.4065038   0.41000322  0.43518698
   0.476101    0.5016566   0.3133861 ]]
[[0.         0.         1.1781555  3.096977   4.0234075  4.0138
  0.5222915  0.51521444 0.5481932  0.52187026 0.36151123 0.5035404
  0.4978701  0.         0.3334898 ]]
[[32.875977    8.025944    1.136503    3.2426858   3.9006834   3.847851
   0.48172438  0.35974455  0.5053845   0.42716533  0.49593508  0.5508584
   0.4148561   0.38193035  0.28902426]]
[[33.783524    8.088209    1.3630929   3.0559156   4.2635384   4.2416067
   0.4883657   0.          0.62187845  0.          0.          0.4909786
   0.5498187   0.6407784   0.45019633]]
```

Here Cole has the best chance against Team 5, the San Diego Padres.

Conclusion/Future Work:

At the conclusion of our project, we were satisfied with the outcome of both our testing period, final model and the predictions created by plugging in the New York Yankees roster to the final model. The only issue detectable with our model was the output being unable to properly account for box scores. We believe that the work we have done here could be greatly expanded upon. We hope to bring in more applicable data while also finding a better comparative method for the predictor. Considering the fact that our model used quite limited amounts of game data, some of which did not accurately represent the actions of a particular pitcher in the game, we believe that we could further break down the data so that we only observe the stats that the offensive team garnered while playing against the starting pitcher.

Contributions:

The contributions to the project were very evenly split. The original idea for the project came from Tony, who is from New York and wanted to integrate Deep Learning and his favorite baseball team. The specifics about the model we used came from Tyler, who had done something similar in the previous quarter in CS231A. As for the coding contribution, it was probably close to a 60/40 split Tony/Tyler, as Tony did the majority of the work to parse the data, which was realistically the most time demanding part of the coding. For the writing, it was close to a 60/40 split Tyler/Tony, to even out the work done by Tony with the coding. We both contributed equally to the video.

References

[1] *New York Yankees Team History & Encyclopedia | Baseball-Reference.Com*. https://www.baseball-reference.com/teams/NYY/index.shtml. Accessed 13 May 2022.

[2] *Retrosheet Game Logs*. https://www.retrosheet.org/gamelogs/index.html. Accessed 13 May 2022.

[3] Garg, Rohit. "A Beginners Guide To Regression Techniques." *Analytics India Magazine*, 13 Jan. 2020, https://analyticsindiamag.com/a-beginners-guide-to-regression-techniques/.

[4] Brownlee, Jason. "Regression Tutorial with the Keras Deep Learning Library in Python." *Machine Learning Mastery*, 8 June 2016, https://machinelearningmastery.com/regression-tutorial-keras-deep-learning-library-python/.

[5] Pette, John. "Baseball and Machine Learning: A Data Science Approach to 2021 Hitting Projections." *Medium*, 17 May 2021, https://towardsdatascience.com/baseball-and-machine-learning-a-data-science-approach-to-2021-hitting-projections-4d6eeed01ede.

[6] *Machine Learning – Baseball Data Science*. https://www.baseballdatascience.com/category/machine-learning/. Accessed 31 May 2022.

[7] Melling, Micah. *Deep Learning on MLB Shift Images – Baseball Data Science*. 12 Oct. 2018, https://www.baseballdatascience.com/deep-learning-on-mlb-shift-images/.

[8] Melling, Micah. When Will a Pitcher Surrender a Run? – Baseball Data Science. 1 Jan. 2018, https://www.baseballdatascience.com/when-will-a-pitcher-surrender-a-run/.

[9] *Major League Leaderboards » 2022 » Pitchers » Dashboard | FanGraphs Baseball*. https://www.fangraphs.com/leaders.aspx?pos=all&stats=pit&lg=all&qual=0&type=8&season=2022&month=0&season1=2022&ind=0&team=0&rost=0&age=0&filter=&players=0&startdate=&enddate=. Accessed 31 May 2022.

Coding Libraries Used:
Numpy, TensorFlow, PIL, Panda, os, matplotlib, google.colab