Globetrotters DATA 477 Final Paper

Arjun Bhan
School of Computer Science & Mathematics
Marist College
arjun.bhan1@marist.edu

Nicholas Blaskey
School of Computer Science & Mathematics
Marist College
nicholas.blaskey1@marist.edu

Mark Mantone
School of Computer Science & Mathematics
Marist College
mark.mantone1@marist.edu

***Abstract*—In the modern day Basketball data analytics is the key to winning. Teams from all different types of sports are basing their strategies and structure around data science. A popular example is with the team the Warriors. For years, the Warriors were ranked amongst the worst teams in the league. This all changed when the Warriors noticed a correlation between the number of three pointers shot by their team and the probability of winning. Historically 3 pointers were viewed as inefficient but the Warriors data suggest otherwise. The Warrior built their team around the plan for 3 pointers to be the focal point of their offence. This plan worked to great success resulting in the Warriors winning 3 rings and making NBA teams all around the league realize the usefulness of the 3 pointer.**

Over this semester our group has created multiple models that help indicate important NBA data. We have designed clusters to group players together based on their stats, used neural networks to check the probability that a team will win and have constructed a betting bot that identifies whether or not to bet on a certain team to make money.

We have based our models on primarily on the NBA games data from Kaggle. It is a Kaggle database containing comprehensive data on all NBA games from 2004 to the end of 2020. The dataset contains a lot of information on each player in every game including stats like field goal percentage, three pointers attempted and, personal pouls.

The actual dataset has the methodology used to collect the data in a linked github repo. This is valuable both to give confidence to the dataset being valid and reliable but also for allowing us to collect further data if needed. The dataset comes from scraped data from the official NBA statistics website.

We have used HTML, CSS and JavaScript to design a user interface allowing the user to use our models and look and interact with our data. The user interface is interactive for each model. The user can apply different models to games in the database.

I. Introduction

Our project goes in depth to analyze different NBA data. We have named our team the Globetrotters to represent ourselves as basketball fans. There are three members on our team. Our first member is Nicholas Blaskey who is a Senior Computer Science and Data Science major from Chatham, NJ. The next member is Mark Mantone who is a Senior Data Science and Analytics major from Long Island. Lastly we have Arjun Bhan who is a Senior Data Science and Analytics major from Sommers, NY.

The main goal of our project is to find the best NBA data sources that we can where we can then analyze and process that data. We also want to create predictive and informative models. We then want to create a user interface to create a platform allowing access to both models and data.

II. Value

A. Sports betting

In recent times the sport gambling industry has begun to flourish. Many states are loosening restrictions previously put on the industry. This has allowed the industry to move into new frontier such as apps and websites. This has resulted in increased participation in sport betting. Currently there are only 20 States permitting sports gambling however this number will likely increase. Analysts predict this number to go up to cover about 80% of Americans in just a few years. [1]

The sports gambling industry is very profitable. The industry currently has a market of size $150 billion. In just October of 2020 over 3 billion dollars of bets were placed. Companies like Simplebet have used machine learning and sport analytics to predict outcomes of games. Simplebet has already over 15 million in profits from its machine learning system. Its CEO, Chris Bevilacqua, believes that the future of sports betting is through machine learning. [4]

Alexandre Bucquet and Vishnu Sarukkai, two Stanford students, have designed multiple different machine learning systems for sports betting. These include Gausian, LSTM and neural network models to predict Football, Basketball and Soccer games. Most of these models had positive prediction rates. Given the success of these sports prediction softwares it is very likely that machine learning applications with sports gambling will continue to grow. [5]

For our project we have used reinforcement learning and neural networks to identify which teams will likely win a game and whether or not you should bet on them. Our program is able to give an estimation on a game and determine how much money you will make from it. With the growth of sports gambling applications like these are more needed than ever. There is potential for users to consider our models prediction alongside with what their own prediction is. The user doesn't

need to blindly follow the predictions we provide. The user can use it as more of a guide or something to consider.

B. Sports analytics

Sport analytics is a growing field that is currently valued at 1.05 billion and is expected to reach 5.11 billion by 2026. Sports analytics access team and players to gain useful information for game performance. Player data is more accessible than ever. Many players are made to ware devices that are used to gain insight on their performance and fitness. These devices track things such as the player's speed, acceleration, and heart rate. [7]

Many companies are outsourcing their sports analytics to prominent tech companies. For instance, in 2019 the Local Organizing Committee (LOC) had made a deal with Special Olympics World Games Abu Dhabi to provide its AI and machine learning system to help improve the performances of its athletes. [9]

The NBA has made major attempts to improve its own sports analytics system. Each team has a group of dedicated analysts who report on player performance and make suggestions on how a team can perform better. Coaches in particular use analytics to develop plays for their teams and find weakness in their rivals.

For our project we clustered players in multiple categories based on their basketball skills such as playmaking, scoring and defensive abilities. This is very useful for sports analytics as it allows one to see the strength and weakness of certain players. Our cluster can also compare players to themselves from another time. This can be useful in showing how player games have changed over time and whether they are regressing or progressing as players. We also have multiple data sources that we have filtered down containing important NBA data such as games, players

C. What Makes Our System Different

Sports analytical models are very common. For instance an Emory Professor named Micheal Lewis created multiple linear regression models to predict the performance and role in the team. He utilized information such as whether the player graduated, how many college wins they had and their height. [11]

Although Micheal Lewis did a great job at designing his linear regression model I felt that it was lacking in multiple aspects. Firstly the linear models he creates usually only use four different predictors. For our model we utilized multiple data tables and categories to make sure as much data as possible was being used for our models.

He utilized linear regression. We took a different approach utilizing much more complex models together. Another major differentiator is that many people create models, or ways to view NBA data. There aren't many sites which display both models and predictors. We feel this is a real advantage since it helps the explainability of the use of models. It allows the user to be able to dig into data without leaving the site to reason about a model's prediction. It also allows for the data to be viewed manually and then consider what prediction a model will be made about. We have more than just a site to view data or a site to run predictions; we have a platform allowing both to be done.

III. Investment

The current cost to the project is essentially free due to aws's free tier. There is however the cost of the domain name that was already purchased for unrelated reasons (nicholas-blaskey.com). In addition there is the cost to run DNS to the domain name with aws's route 53 ($0.50 each month). The domain name is also not necessary at this stage.

These costs however cannot stay this low because arguably even now we are limited by the low compute power of the free tier of EC2. The API has difficulty now responding to large amounts of requests at the same time. It makes actions like looking up a list of game ids to get the information about this infeasible. The design of the API however was influenced to try and prevent cases where many requests would need to be made.

In addition to the low computing power limiting development choices, it also limits the amount of data we can transfer. We have had to limit things like how many rows the game table can show due to the low compute power. This doesn't affect the minimal viable product but in the future it will be essential for having hardware to give the users all the data they could ever want in a timely fashion.

To provide the level of data access potential customers are going to expect we are going to need to shift off of aws's EC2 for the database server. A dedicated database server will provide more reliability and performance and prevent cases which we dealt with in development like having a bad query from the API bring down the whole system. We are going to utilize the Amazon Aurora PostgreSQL-Compatible DB.

We have made use of aws's cost calculator to calculate the cost of a dedicated database server. [12] We have intentionally gone for a beefier server than what we could probably make out with. This is because our business model is based on having good data reliability and speeds. A slow database would effectively kill our business. However we are realistic in that our application can make effective use of caching due to the data not changing very often so we don't need to overdo it. We have chosen to go with 2 DB instances replicas of each other to ensure data integrity. We went with the db.r5.large which has 2 vCPUs and 16 GiB of RAM. We choose 50 GB of storage along with an estimated 1 write per second and 5 reads per second. We additionally have an extra 100 GB of backup storage.

We have chosen to go with the on demand pricing model. This will allow us to scale up or down depending on how the business and technical needs of the project change over time. This works out to an estimated $433.65 a month or $5,203 yearly. The on demand pricing model costs us more but gives us the ability to better suit the needs of the customers.

In addition to the database server we will require both an API server and a frontend server. For this we are going to use

Amazon EC2. We are going to use the EC2 Instance Saving Plan pricing strategy to get a good deal. We are going to go with two separate instances. The frontend server will be less demanding since it will only need to basically serve HTML files so for this we will go with 2 vCPUs and 8 GiB of RAM. We will also have 30 GB of SSD space. This will cost $30.73 monthly for the instance and $3.00 monthly for the storage. For the API server we will need something with a little more power so we are going with 4 vCPUs and 16 GiB of RAM along with the 30GB of SSD storage space. This will cost us $61.54 for the instance and $3.00 for the storage each month. This totals our EC2 cost to $98.27 each month.

For training models we will utilize google's colab pro. [10] This is $10 a month. It is unclear if the pricing is different for business use so we are going to assume it is the same for now per user. At a minimum we will have 3 employees so we are going to assume our cost of colab is $30 monthly. We will likely at some point need to invest into an on premise GPU enabled machine for training networks however colab will be enough for the first year. This works out to $360 yearly.

Our total cost of compute is around $6,743.04 the first year. It is safe to assume this cost will grow as more people use our product. However this cost of $6,743.04 is sufficient to get enough users where if we outgrow the compute we have it is really more of a good problem to have because it will mean a lot of people are using the service. The choice to focus primarily on cloud offerings allows us to be able to scale if the business starts increasing rapidly. These costs can be evaluated along with the current revenue at the point where we start out growing these initial hardware predictions.

We will also require a domain name which we can get something like analyticsBasketball.com for $8.88 a year. [8] Route 53 for DNS routing will cost a trivial amount. It is $0.50 a month for the first hosted zone and $0.40 per million queries up to a billion. It is unlikely we will get anything close to a substance cost on the DNS.

We are going to make the assumption this will be a "side hustle" for the team. That is we can't afford at this stage to pay salaries for development costs. Especially since quality developers (like ourselves) are expensive. There just isn't money at this stage to pay a competitive salary worth taking. It is less than ideal to have the development of this project work primarily in the free time of the team members but it is the only option unless major investment is made in the company.

The total costs yearly come out to around $6757.92. This is a rough estimate and this could change. However this is a good baseline and goal for revenue to make. This leads us to our business model. We have two main customer groups. Individuals and organizations interested in the NBA. Primarily individuals will be either sports betters, those in a fantasy league, or just superfans. Likely we can make a subscription model off these individuals for a reasonable price of $5 a month with a free trial for a month. Just looking at individuals we would need around 1,352 months paid. This would be around 123 people subscribing for a year (with the free month factored in).

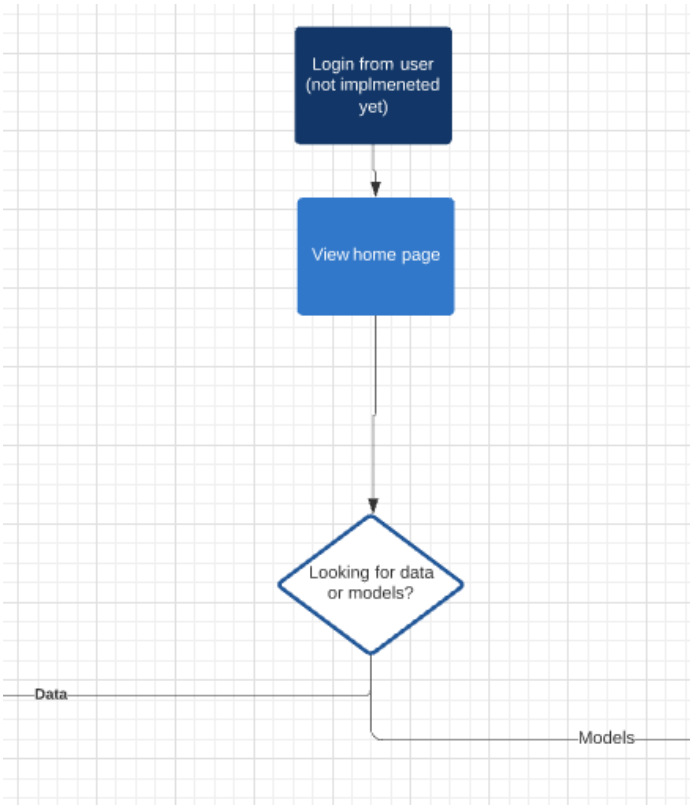

Fig. 1. Starting choice of the flow chart.

The other target for customers are organizations. These would be firms or startups likely trying to make money off sports betting. These customers will be willing to pay more but harder to get. We will also need a way of differentiation between our enterprise pricing model and our individual pricing model. We can do this by providing more up to date models for enterprise users and API access and even database query read access to enterprise users. The value to an enterprise customer could be very high so they will likely be able to pay much more. We will set the price to start $99 a month for enterprise customers. We would need only around 6 enterprise customers to cover the cost of running the platform.

Likely this cost strategy particular for enterprise customers will need to be updated as the product's value is more proven. With a mixture of enterprise customers and individuals we should be able to cover the monthly cost of the infrastructure. Since we have cloud hosting we will be able to scale up in the case where more customers are using the platform than expected or scale down if less customers are. This should allow a much more efficient business model that will allow us to adapt our costs to how many users we have.

IV. User Experience

From a high level the user will either want to access the data or models page. Our flowchart reflects this with a high level choice after the user has logged in (not implemented yet) and visited the home page.

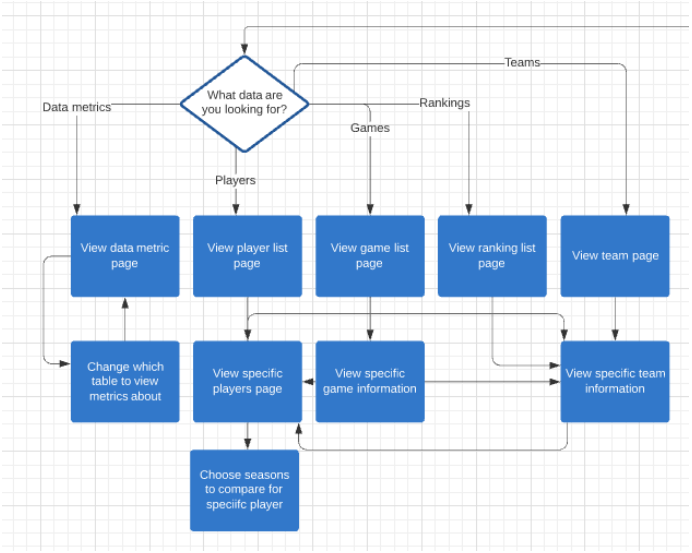

Fig. 2. Flow chart section for data access.

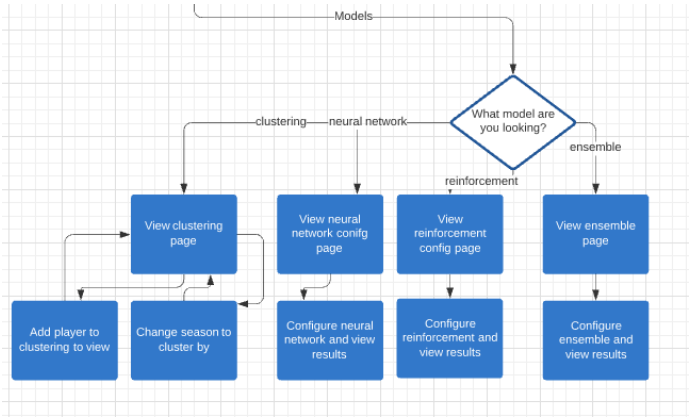

Fig. 3. Flow chart section for model access.

If the user entered the data choice they will have the ability to view the different aspects of the data we have in our database. They will have the ability to view general tables about the data in addition to specific information about players, games, and teams.

If the user entered the data choice they will have the ability to view the different aspects of the data we have in our database. They will have the ability to view general tables about the data in addition to specific information about players, games, and teams. If the user instead chooses models they will have the ability to interact with the different models in our system.

The way this would look like more concretely is they would be directed to the home page after signing in or registering (after this is implemented). They would be greeted by a list of features that the system provides along with a list of the team members. In addition there is a side navigation bar that is the primary way users will get around the system. In this side bar there is two sections, one containing all the data pages and another one containing all the models.

A. *Data user interface*

In the data section there are 5 pages, data quality and metrics, players, teams, and games. In the data quality metrics the users are greeted with a bar graph showing the amount of rows in the selected table, the amount of rows with any null columns, and the amount of rows with a null for each column. The user is also able to change the table option at the top of the page and the graph will automatically refresh with the table.

In the player option the user is presented with a table of information about all players. This would ideally show every single player in our database however due to performance limitations currently we only show the 2019 season players. This table is searchable and sortable. From here the user could search for a player they are interested and click on either their team id or their player id to get taken to the specific player or specific team page. The specific team page will be discussed after the team page is discussed.

In the specific player page there is a graph showing season summary statistics. There is also two select boxes with the seasons played the given player was in the NBA for. The user could change these seasons to compare how a player's performance changed over the years. They could see if a player is getting better or worse from previous seasons. In addition to this graph there is also a team history table showing what age they were, what season it was, and what team they were on.

In the team page a searchable and sortable table is presented with all the teams in the NBA. They can click on the team id to be taken to the specific team page. In the specific team page there is two bar charts shown. The first is a chart comparing the arena capacity. The second is a chart comparing how many players have been on the team team. In each bar chart the color is made distinct for which ever specific team the user is looking at to make it easy to compare the team of interest to all teams at whole. There is in addition, a table showing the players that have been on the team along with basic information about them and a link to the specific player page.

In the game page a searchable and sortable table is shown with all the games in the NBA. Due to performance limitations currently it only shows a small date range of games. The user can click on the game id of the game they are interested in to be taken to the specific game page. In the specific game page there is two tables. One representing the home team's players and their statistics for the game and another for the away team. The header of the table contains a link to each team along with a message indicating if they won or not. There is also a bar graph showing a comparison of all the players that scored a point and how many points each player scored. This bar chart colors the bar based upon the team the player is on.

The ranking page just presents a searchable and sortable table with a link to the relevant team.

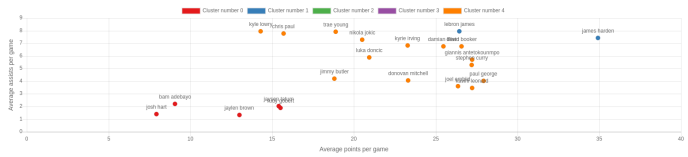

Fig. 4. Visualization of the cluster chart presented to the user.

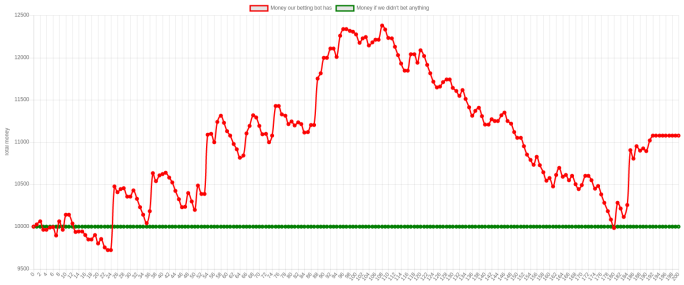

Fig. 5. Visualization of the betting bot's performance.

B. Model user interface

The model page has 4 pages, clustering, neural network, betting bot, and ensemble. In the clustering page the user is shown two charts. These charts are scatterplots of prefilled players of interest. The first chart shows average assists per game verses average points per game with the offensive cluster labels. The second chart shows average blocks per game verse average steals per game with the defensive cluster labels. The user then has the ability to change the season that these average statistics come from. The user can also choose to add a player by their player id or add a player by name. They will need to click submit for the season to change and the player they entered to be added. After submitted the player will be added to the plot, if there is no match for a player name a warning will appear informing the user of this.

The neural network page presents two form options. The first is the tweet that the neural network will use in the prediction of the result of the game. This tweet can either be made up or copy and pasted from a real player's Twitter account. There is also a starting date that utilizes a date picker component allowing a user to easily find the correct date to pick from. This starting date is used to find which game in the database to predict. There is a submit button that runs the model. There is also a clear report button that clears the models prediction for the user's convenience if they do not want to refresh the page and want the previous report to be cleared.

After clicking submit they will be shown the neural networks predicted probability for the home team winning and the away team winning. In addition they will be shown the scores in the first, second, and third quarters along with the ending score.

In the betting bot page they will be shown three form options. The first is the amount of money the betting bot starts

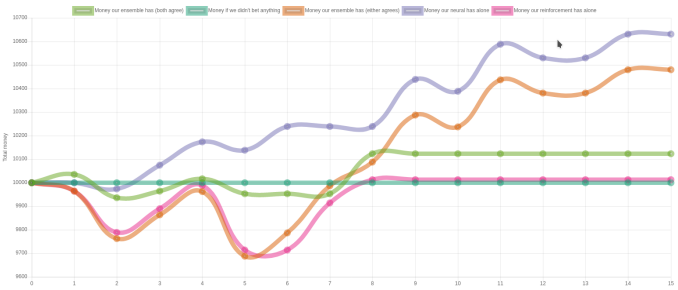

Fig. 6. Visualization of the ensemble's performance.

with. The second is the number of games to run the betting bot. The final option is the starting date of the betting bot. There is a submit button along with a clear report button for the same reason as the neural network page.

After clicking submit they will be greeted by three things. The first is a report showing how much money the betting bot ended up with. The report will also show how many games were bet on and profit. This is then broken down into favorite verse underdog bets and the respective profits of those bets. The next thing they will be shown is a graph of how much money the betting bot had verses the games the betting bet could bet on. It also provides a baseline line to allow easy comparison of it the betting bot preformed well. The final thing they will be shown is a match by match report showing information about the games the betting bot could have bet on along with an indication if the betting bot did bet on those games. There is also a profit and loss on shown each game along with a running profit and loss shown.

In the ensemble page the user is shown an form to configure the ensemble model. They can choose the ensemble's starting money. They can also choose the number of games to run the ensemble on. They can choose the starting date for how to run the ensemble model. They are also able to choose a threshold for the neural network. This is a probability value that the ensemble will treat as a level for which the neural network probability prediction needs to be at in order for the ensemble to consider the neural network to recommended betting on a game. There is also a space for two tweets for every number of game selected along with labeling the game index of the tweet. The user can enter in as many tweets as they want for the ensemble or choose to leave them blank. The user can click submit or clear a previous report if they want.

The ensemble result shows a summary for the strategy of both models agreeing. This shows the profit, money ended up with, and profit or loss. It also shows the profit or loss and number of games bet on the favorite or underdog teams. Then there is a chart showing the money if you didn't bet on anything, you bet only according to the neural network, only according to the reinforcement learning model, if you bet to the strategy of if only one of the models agree, or if you bet to the strategy of both needing to agree. There is also a table showing the match by match report. This gives information about each game along with probabilities that the

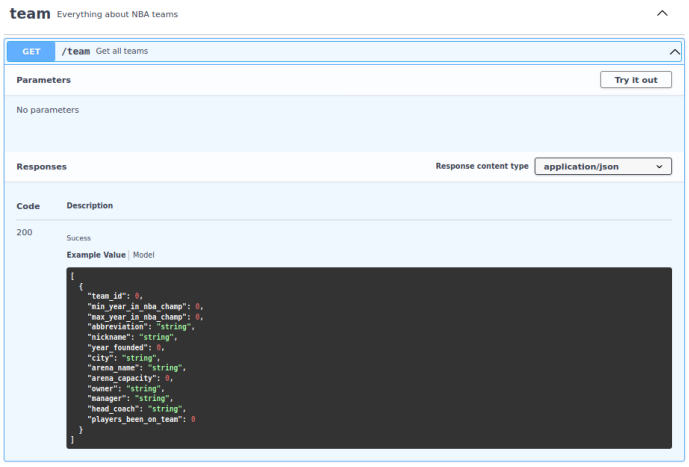

Fig. 7. Example of the API swagger documentation.

neural network predicted and the betting bots prediction. There is also a match by match profit or loss and a running profit or loss if you betted with the strategy of both models needing to agree.

C. API user interface

The user interface does not include solely the dashboard. Our users also include organizations that may have their own systems both for game prediction and NBA data. To provide for this we have our API which authorized users will be able to utilize. The API is documented to support this. We have utilized swagger both for creating the API but also for documented the API. The API is well documented so that we can provide it as another interface for our project. Some of our users will gain most of the benefit from our system from the API we provide rather than the dashboard.

V. Process

A. Data pipeline

The first stage of the data pipeline is ETL (extract, transform, and load). This step combines three data sources into one database.

The three data sources are a general NBA statistics dataset containing information about games, game details, players, teams, and rankings. The information about players is limited so another data source is used to augment this with things like heights and weights of players. Another dataset is also used to add in betting odds information to the database. These datasets were stored as csv files in the github repo except for the betting odds which was stored as an excel file.

The ranking, team, game detail data can be inserted into the database essentially as is with minimal work done. The player dataset from the NBA statistics however needs to be combined with the dataset containing more information about players. This is done by matching on name. Care was needed to be taken to consider NBA players with the same name. We had approximately 22 of these players in our dataset and dealt with them in cleansing. These were identified by finding players who had a name corresponding to a different player id. We did have a decent number of players we had in our NBA statistics dataset that we could not find in our players dataset. These were left for the cleansing dataset.

The game data also needed to be combined with the betting odds data. This was done by comparing the date of the game played along with the teams playing. We did have a number of games in the NBA statistics dataset that were not able to be found in the betting dataset and these were left for cleansing.

For the ETL step we utilized pandas for data processing, psycopg2 to interact with the database, and openpyxl to read the betting odds data since it was stored as an excel file.

The next step of the datapipe line was cleansing the data. In general, cleansing reads from the database then updates the database. We first created an analysis script that queried the database to find out exactly which tables had which columns as null. We found the player table had two types of nulls. One being players we were unable to find and another being players we were able to find but had missing information about them.

To cleanse the player table we called the official NBA website's unofficial API. It was somewhat clear they didn't want people doing what we're doing (mass calling the API and downloading data) so we ran into rate limits. [6] However we were still able to call the API with enough time in between calls. This helped us find players and players with the same name that were missing since we queried the API based on the player ids not their names. Steps needed to be taken to ensure the data from the website was in the same format as the data in our other datasets such as converting height and weight values. This helped eliminate most missing players however even the NBA website still had missing values. We did however get a much more cleansed player table after utilizing this API.

The team table was mostly complete except for some arena capacities. These were searched manually (which was possible since there were only 5 missing values) and entered in the script to completely make the team table null free.

The game details table was relatively complete except for a lot of missing starting positions and some games with all nulls. These rows with all nulls were assumed to have not played and were marked as such. The missing starting positions were filled in based on what starting position they were most likely to play.

The ranking table was entirely complete. The game table was entirely complete except for missing betting odds values in some cases. These betting odds were left as null and left to deal with specific models that would need to use them. This choice was made in the interest of time. Realistically our only option was to try and find another dataset which could end up having the same issue. We decided that this was an acceptable amount of null values and just something we would need to consider in future when using the data.

For cleansing we used psycopg2 for interaction with the database, and nba_api for interaction with the official NBA website. [2]

The next and final step of the pipeline was feature engineering. Feature engineering read from the database then updated

the database. The features we choose to create came from mostly domain specific knowledge. Thought was put into what features we thought were useful from both a data access side and analytics side.

We created a feature for if the home team in the game table won the previous game and the same for the away team. Feature engineering also created a season range feature for players giving insight on how long they have been in the NBA. It also created a ranking improved feature for the ranking table to tell if a team had improved since the previous ranking. The game details table had a top scorer feature added to mark the most impactful player by points scored. The team table had a number of players who have been on that team feature to give insight if a team is particularly aggressive in adding and dropping players from their roster. Feature engineering used psycopg2 for interaction with the database. These were the basic steps of the data pipeline.

These steps were combined with a simple bash script to combine these steps into one. The scripts also have decent logging to show the process and make sure the user of the script can tell something is going on since the scripts, especially cleansing can be time consuming accessing the official NBA website.

In addition, a data queue was set up using Apache Kafka to consume messages sent from producers. Docker and Docker-Compose were used to simplify the process of creating a Kafka instance. Docker and Docker-Compose allowed a lot of the hassle of installing Kafka to be avoided. Docker-Compose allowed an easy one command set up of the server. This queue had a consumer setup to insert messages from a certain message format into the database.

The queue expected messages of a specific format. It expected a JSON message with two attributes. The first attribute was the type. This could be player, team_ranking, game, or player_game_details. The second attribute was data. This expected a comma separated list of values corresponding to the different format type. These formats are all documented in the README of the data queue.

Then an example producer was made to test this process. In addition to Kafka, the Python library Kafka was used to connect to the Kafka instance. This could be useful in the future to set it up to automate the process of keeping the database up to date in real time as the NBA season plays out. We choose Kafka due to being inexperienced with Kafka. Previously one of the members had worked with RabbitMQ a decent amount, a competitor to Kafka. It appears though that Kafka has gained more popularity over RabbitMQ so experience is valuable in it.

B. Statistical Analysis

An important part of our project was the statistical analysis of the data. We used postgreSQL in R to look at certain parts of our database. We used our team table to make a dot plot that shows what year each NBA franchise started in order from oldest to youngest. The oldest teams included the New York Knicks, Golden State Warriors, and the Boston Celtics. The youngest NBA team is the New Orleans Pelicans.

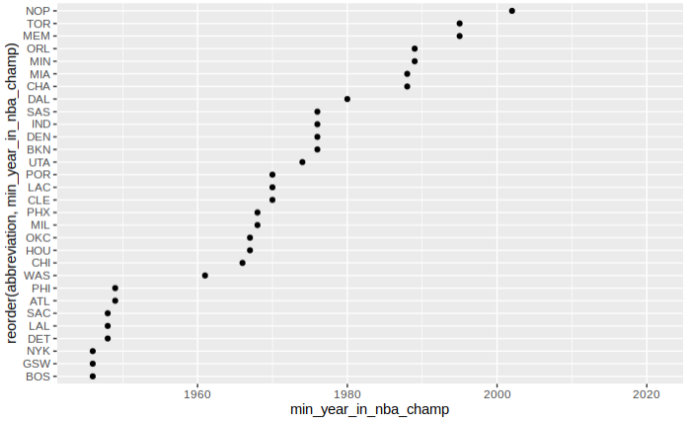

Fig. 8. Team age in the NBA.

```
[1] "Kobe Bryant Career Stats"
[1] "Career PPG"       "26.6214634146341"
[1] "Career Assists Per Game" "4.9180487804878"
[1] "Career Steals Per Game" "1.45463414634146"
[1] "Career Blocks Per Game" "0.382439024390244"
[1] "Career Turnovers Per Game" "3.11317073170732"
[1] "Career Rebounds Per Game" "5.28487804878049"
[1] "Career FG %"      "44.3705799151344"
[1] "Career Three Point %" "32.7527527527527"
[1] "Career FT %"      "83.9562330956479"
```

Fig. 9. Kobe Bryant Career Stats.

After this we used our player table to make a scatter plot that shows the height of an NBA player on the y-axis and the weights on the x-axis. As expected, when the height increased so did the weight of a given player. Another interesting thing we did with data was querying the data from the player table and the player game detail table to show certain NBA players career stats. In this particular case we found the career stats of Kobe Bryant, Lebron James, Kevin Durant, and Steph Curry. When we looked at this data, Lebron James had the most dominant stats in most of the categories compared to any of the other three players. You could do this with any player realistically if you have their player id, by simply plugging in the player id into the query.

We next decided to try to find the points per game of NBA teams in a given season. We were able to calculate the away points per game, home points per game, and the total points per game for the New York Knicks and The Milwaukee Bucks in the 2020 season. The Bucks had one of the highest scoring teams in 2020 with 118 points per game. The Knicks had about 104 points per game. The final component of our statistical analysis was to find a player's stats for a given season.

This was definitely the most difficult to compute because our database base doesn't have a table for per game stats, it

```
[1] "2020 Bucks Points Per Game"
[1] "Home PPG"          "118.578947368421"
[1] "Away PPG"          "119.058823529412"
[1] "Total PPG"         "118.805555555556"
```

Fig. 10. 2020 Bucks Points Per Game.

```
[1] "Russell Westbrook Stats 2016"
[1] "PPG"               "31.2444444444444"
[1] "Assists Per Game" "10.2666666666667"
[1] "Steals Per Game"  "1.63333333333333"
[1] "Blocks Per Game"   "0.366666666666667"
[1] "Turnovers Per Game" "5.27777777777778"
[1] "Rebounds Per Game" "10.3555555555556"
[1] "FG %"              "42.1860465116279"
[1] "Three Point %"     "33.9563862928349"
[1] "FT %"              "83.8709677419355"
```

Fig. 11. Russell Westbrook Stats 2016.

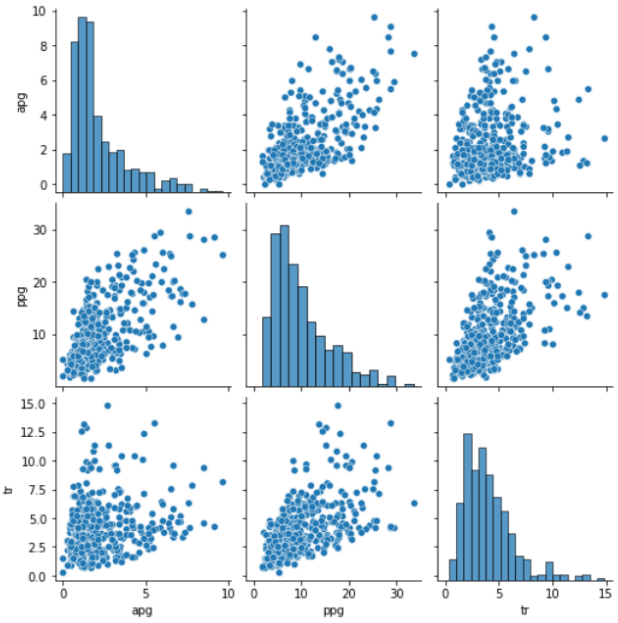

Fig. 12. Pairwise plot showing correlation between variables.

has a table for every player's stats in every game. For this we had to merge the player table with the player game detail table to get the per game stats for certain players. In this instance we calculated Steph Currys stats from 2016 as well as Russell Westbrook's Stats from 2016. Obviously Russell Westbrook had better stats than Steph Curry in this season because it was his MVP season. We decided to use this data to compare the best players and the worst players in the league.

We thought it would be a good idea to transfer this data into our own clustering model. We can use all players stats per game per season to see where players rank amongst each other in the NBA for a given season. We will use scatter plots with clusters to show where each player is.

C. Clustering

We chose clustering as one of our models for the project. To do this we made a colander notebook which uses Python. The first part of this was challenging because we wanted to get players points per game. Eventually we were able to get this by using a select statement to get a player's player id and player name where the season is 2019.

After this we used a for loop to select the average stats for players from the 2019 season. After getting this information we created a data frame of all these players that played more than 300 seconds in 2019, this was 437 players in total. We got the average of each column so we could see the trends. The stats were underwhelming because we took players that didn't play very much so the stats were on the low side. This is a good thing though because we want various types of players so we can clearly distinguish the best players from the worst players when we cluster them.

We next loaded a pairwise plot that included assists per game, points per game, and total rebounds per game to show the correlation between these variables. All of these variables correlated well with each other because most of the good players have dominant stats in all categories. We clustered this data next using Sklearn. For this we used assists per game and points per game as our variables. These are offensive stats so we were able to see who were the best offensive players in the league. We used 5 clusters and got the labels for the players. Next we added a cluster column to our data frame to easily see which players were in the same clusters. An interesting thing that we did was show how many players were in each cluster. Cluster 4 had the most players in it and cluster 3 had the least. The plot that we made showed each cluster in a different color and it looked like cluster 3 consisted of the best offensive players in the league. This makes sense because there aren't a whole lot of them which is why there's only 16 players. Overall the plot shows which players are most similar to each other in terms of points per game and assists per game.

To verify this we decided to take Luka Doncic and Giannis Antetokounmpo, who are two of the best offensive players in the league, and see if they were in the same cluster. To no surprise they were both in cluster 3 which consisted of the better offensive players. After doing the offensive players, we wanted To find the best defensive players in the league so we clustered players based on blocks and steals. This showed that cluster 2 had the least amount of players which indicates that these are the better defensive players. Our plot shows which players are most similar in terms of defense.

To verify this we took Og Anunoby and Draymond Green,

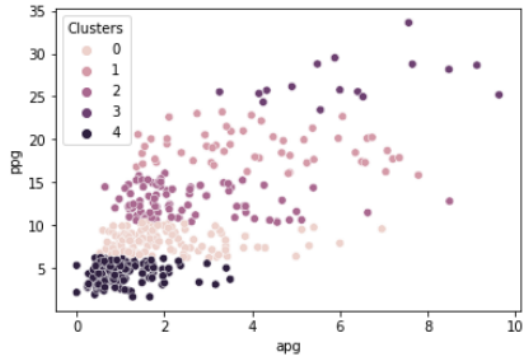

Fig. 13. Offensive stats showing the cluster labels.

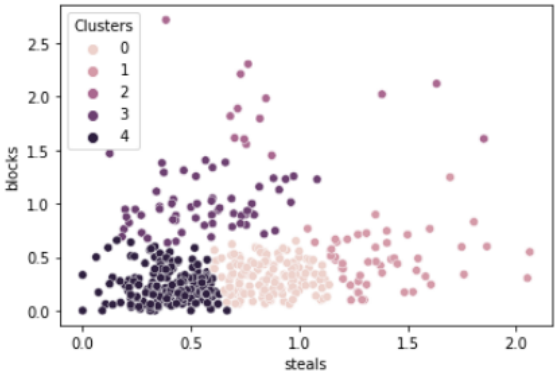

Fig. 14. Defensive stats showing the cluster labels.

who are two of the better defensive players in the league, and see if they were in the same cluster. They were both in the same cluster which is good. These models helped us learn a lot about the data.

We clustered NBA players based on offensive stats and defensive stats to demonstrate how all the players compare to each other. Overall we looked at many different parts of the data to get a good interpretation of all the data we would eventually be working with.

D. Sentiment Analysis

Sentiment analysis is a cornerstone of data science. It allows us to extract the connotations from a textual source such as social media posts, news articles and, reviews. This can be very useful in multiple different areas such as clustering of customers by their textual connotations, researching reviews on products or, indicating the mood of a person.

For our project we decided to extract NBA player's twitter posts in order to analyse whether the connotation of a player's post a day before a game affects their performance. Our hypothesis was that NBA players would post more negatively before losing as they will be less happy with their teams performances. We used sentiment analysis in our neural network to help the system predict the outcome of the games.

For this sentiment analysis we used the NLTK and Pandas Python libraries. Pandas is very useful for bringing in csv data into the Python platform. The NLTK library has many text analysis functions that allow us to analyze the sentiment of a tweet. In our code we first removed non alphabetical and stop words from the tweets. We then lemmatized the words so that they were all in similar form. We then took the positivity and negativity from each tweet and added them to a list.

We were able to get the twitter data from the Twitter API system. We made sure the players we were taking the tweets of were very skilled and had been in an all star team in the last 2 years. We used the data about which teams the player has played for in conjunction with the years they played with that team and were able to match the data with the games and player data allowing us to get information on the scores, outcome and tweets a team had before playing. We took this data to R to sort it by team data and name.

E. Neural Network

Neural Networks are based on the learning mechanism of the human brain. Neural Networks contain neurons that relate data to one another and find correlations between them. The neurons are also able to weigh the importance of the different data.

This method is excellent at finding correlation between data with a large number of rows and columns such as the data outputted from the R code. Before we attempted to put the data into a neural network we had to alter it so the method could be effective without data. Neural networks work best with data between 1 to 0 or 1 to -1 depending on how it is designed. Neural networks will have issues with columns containing categorical values.

For the data to be correctly processed we made sure all large integer values were between 1 to 0. We used the MinMaxScaler function to make the number in this format. For the team id we turned them into a boolean list with the OneHotEncoder as this column contained categorical variables. Each column of the list represents a team. When a team is being represented in the data the list marked them with a one if they were not they were represented with a 0.

We were able to split the data into a response and predictor sections. For the predictors section we included the month, year, team id, whether it is a home game, if there was previous tweet and the positivity of the tweet, The predictors also contains values from the previous game such as that games point, field goal percentage, field throw percentage, 3 point field goal percentage, assists, rebound and outcome. The response list contained only the outcome of the game for the given data point.

Using the train_test_split function from sklearn we were able to divide the list into a train and test split. We chose an 80% split for the train and test set. Through experimentation we found that three layers was the most optimal for neural networks performance. We based the number of neurons on our input layer. The first hidden layer has twice the neurons that our input layer had. Our second layer had the same number of neurons as our input. For our third layer we took the average of our input layer and output layer amounts of

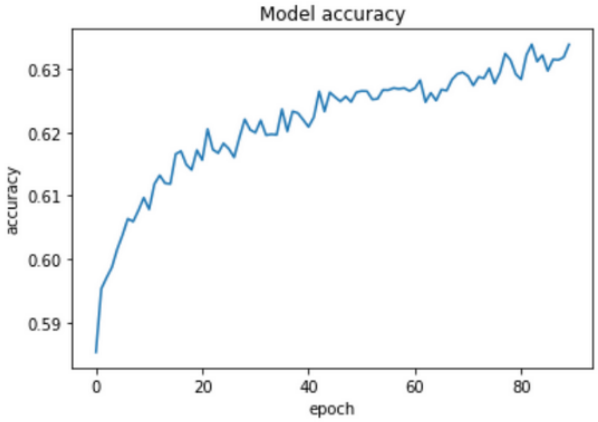

Fig. 15. Accuracy of the model while training.

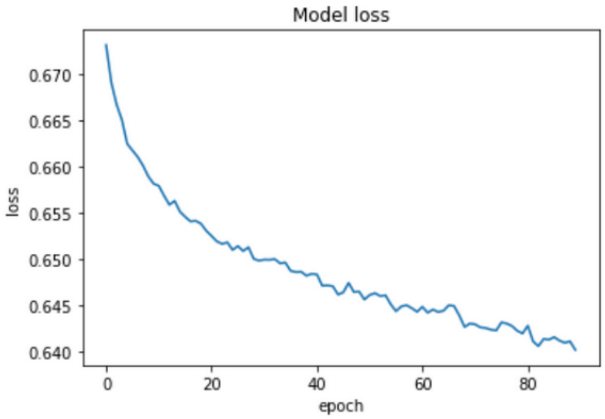

Fig. 16. Loss of the model while training.

neurons and got 23. We choose to use a tanh activation for the first four layers based on testing and its ability to handle negative numbers. For the output layer, we choose a sigmoid activation as it is great at handling output with a range of 1 to 0 like the outcome column. We choose to use epochs through testing using matplotlib to visualize how the epochs are affecting the model.

F. Reinforcement model

The reinforcement model started by getting all the game data and dropping all the rows with missing betting data. This was then split into a train and test set based on a date. The date allowed us to simulate a real world situation since we would be able to train our model on all the games that have occurred up to a point in which we would be on our own.

Then work was put into figuring out exactly what a money-line is and how to convert that to something more usable. Next betting strategies were testing to give a baseline of something to beat. Some betting strategies were always betting on home, always betting on away, always betting on the favorite, always betting on the underdog and some less sensible ones. Every betting strategy tried, lost money. This was expected since if

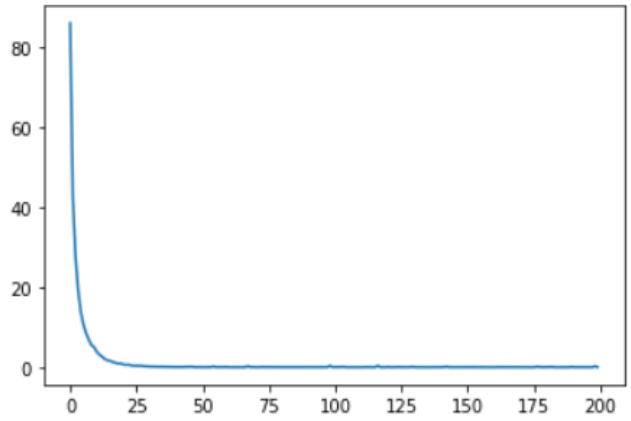

Fig. 17. Reinforcement model loss against training epochs.

all you had to do was bet on home then everyone would be rich off NBA betting.

This was important since now we had two things to try and beat. The first was a simple betting strategy. If our model wasn't as good as always betting on the favorite then it doesn't provide any value. The other baseline is of course if we are able to end up with more money than we started with.

After this a lot of research was spent into both what types of reinforcement learning models were out there and examples of them. We settled on Q learning since it is both one we have heard about and one that seemed like it was able to perform well in similar tasks. The task we found most similar to this one was a stock market example. Sports betting is close to the stock market because they both fundamentally involve using past information to try and predict an action to take in the present which may or may not pay off based on future information.

We found an example online of a stock market prediction of Q learning and spent a lot of time both running it and trying to understand each line. Q learning involves a lot of complex math and code so it took awhile to figure out how to apply this example to sports betting. After commenting on each line we started to get an idea of how to shift the example to our situation.

We mostly needed to change the action space to include only two actions instead of three. For each game we considered each team separately to decide if it was a good bet. In this case we can either bet on the team or not bet on the team. In the stock market example you can buy, sell, or hold.

We also needed to change how training occurred and the reward function. The reward of betting is different since after the game the payoff is immediate. You either win money or lose your money. In the stock market example you still just have stocks and can continue to hold the stocks after the price increases or decreases.

The model only considers the money lines of the teams. This is a simplification made mostly due to our inexperience with reinforcement learning and Q learning. This makes the model less about predicting NBA teams and more about gaming the

betting odds. Its main goal is to find patterns in money lines and outcomes and see if there are cases where for example a 1.5x bet has a 90% chance of working out. These patterns may exist and in theory could be identifiable by the reinforcement model.

After adapting the action space and the training and reward the math mostly remained the same for predicting new outcomes and updating the weights on the models based on the decisions it made in training. The model was then trained and evaluated on the training and test set.

Since the reinforcement model was not a classifier it we could not use the normal metric of accuracy to evaluate how well our model performed. However, it was clear our model should be evaluated on how well it is able to make money. We found the model to be hit or miss even on the train set. Some cases the model performed very well. Other cases it went broke. It does appear however that in the majority of cases the model ended up with more money than it started with, The figure shows how much money the model ended up after running for the entirety of the training data for each of the epochs the model was trained on.

There is not a direct upward trend so it appears our model did not greatly improve performance over the training run. This could tell us our model did not require that much training. In addition this could be indicative of our model not being able to capture the underlying trend of the betting data. It could be very likely this second one. This is the first time any of us have done reinforcement learning and the idea we could make something that would reliably figure out NBA betting is optimistic. If we could do this easily many people would be able to do this and make near infinite amounts of money off betting.

The model was then evaluated on the test set (data the model has not seen before). The graph shows the outcome of 100 runs of the model on the entire test set with the fully trained model.The model appeared to follow the same hit and miss pattern of the training set. It does however appear to have done about as well on the test set as it did on the train set. It still has the case where about half the time it ends up with more money than it started with. However it is certainly not reliable and would be very risky to try and utilize in practice.

After the evaluation of the model the model's weights were saved with Tensorflow to be loaded into the API. The technologies used were Tensorflow for the model, psycopg2 for getting the data from the database, and pandas for data processing.

G. Ensemble

The ensemble was implemented in the API and tested and evaluated in colab. The ensemble is very simple in that it just reports the probability of a team winning from the neural network's prediction along with the reinforcement's decision if it made a bet or not. This leaves a lot up to the user of the model to make a decision. The model doesn't provide a binary outcome or a probability. It is up to the user to make that outcome.

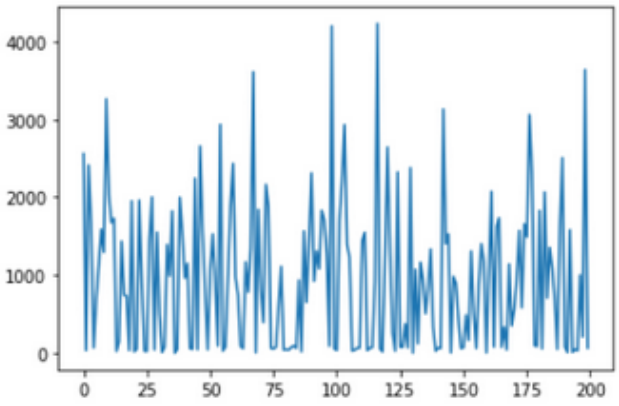

Fig. 18. Money reinforcement ended up after each epoch of training (started with 1000)

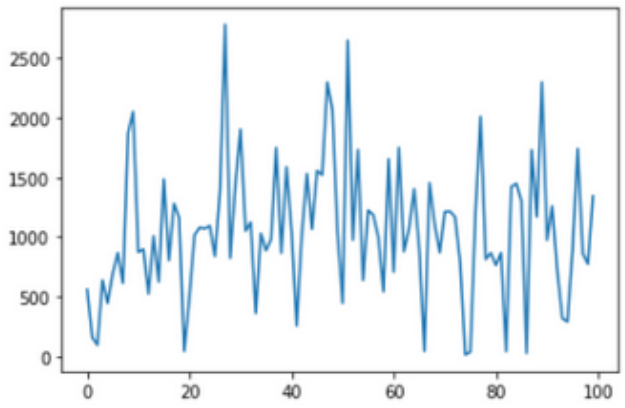

Fig. 19. Money ended up for 100 runs on the test set (started with 1000)

There are strategies that we tested and compared the ensemble to. The first is only making a break if both the neural network and ensemble agree on the outcome of a game. This would be a conservative approach that you would only bet if both models are sure. Another approach is if one or more of the models think a team is going to win then a bet would be made.

The ensemble along with different strategies were evaluated by collecting a dataset of values unseen to the reinforcement model. It would have been ideal if we had also been able to ensure that the values were unseen in the training of the neural network, but the neural network train and test set was split randomly. This is something that if more time was had and this model was going into production we would need to reexamine this.

This dataset of values was then processed into the format that the neural net expected. The neural network classified this dataset to give us what it thinks the outcome was. Then the reinforcement model was run 100 times on the dataset to give a sample of what the dataset expected. These outcomes of the neural network and reinforcement were compared among

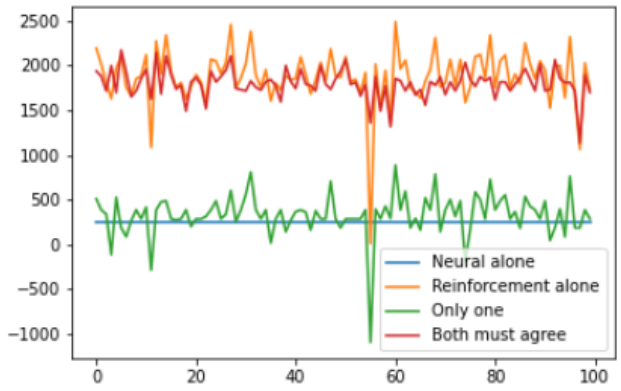

Fig. 20. Results of 100 runs of the ensemble on game data after 2017

four strategies. These strategies were only listen to the neural network, only listen to the reinforcement learning, bet if both say a team will win, bet if one of the network says it should. The results of this evaluation are listed in the table below assuming a starting money of $1000 and $100 bet values. Note the ensemble allowed for betting on margin so some values may be negative.

TABLE I
ENSEMBLE MODEL RESULTS

Evaluation Function	**Strategies**			
	Neural	***Reinforcement***	***Only one***	***Both***
median	253.90	1874.52	1796.23	332.15
mean	253.90	1884.44	1811.19	339.10
max	253.90	2487.04	2170.89	886.72
min	253.90	12.89	1132.56	-1091.43

The neural network underperformed and the reinforcement somewhat overperformed. The reinforcement alone generally was giving good returns. Both were similar to the reinforcement alone but a little less. Only one was similar to the neural network alone. It seems the neural network limited the performance of the ensemble. The neural network's performance made it so that the ensemble became weaker than just the reinforcement alone. This pattern is reflected in the chart. Both agreeing are dragged down to the reinforcement alone levels. Only one agreeing is dragged down to neural alone's level.

The ensemble model uses a combination of the technologies used for the reinforcement model and the neural network. It uses psycopg2 to get the data, sklearn to preprocess the data for the neural network, pandas for data processing, and tensorflow for the reinforcement model.

VI. Ethical/Privacy Implications

Ethical implications that come with all types of data. For example, in an ESPN article NBA teams are said to use increased technology to track players on and off the court. "Various teams have begun experimenting with sleep trackers, off-court movement monitors and fluid tests – including blood and sweat – in order to improve player health and performance" [1]. This movement happened so fast that the National Basketball Players Association had not been able to form an opinion on it. NBPA counsel Ron Klempner said. "Obviously, we'd have serious privacy and other fairness concerns on behalf of the players. We've barely left the starting line on these issues"(Torre & Haberstroh, 2014).

There is currently no rule that restricts teams from mining off-court data. There is a fear that is going around that leaking this data could potentially be held against them when it comes to contract talks. This is a huge problem that could arise in the NBA. This can be compared to our data that we are using about the players but on a much smaller scale. Obviously medical records are much more important to NBA players than what we are doing.

The privacy implications that tie to what we have done is using the players on court data for our project. These players did not consent to being used for our project. They didn't ask to be clustered, have their tweets in our data, or have been involved in statistical analysis.

It is easy to quickly dismiss this concern as something that comes with the territory of being a public figure. However this is something that we should consider because they are people before they are public figures. In addition this opens up the broader discussion of what control a person should have over their data. Not just NBA players data is being mined by people who they did consent to mine their data (or about as close as without consent by getting people to agree to terms and conditions they haven't ready). Society is only getting increasingly data driven. The conversation needs to be had for what rights people have to other peoples data including NBA players.

There is also an open question of how much data can be utilized to quantify decisions relating to sports or competition. It is essential for teams to invest in talent. There is of course a certain skill level required before being on a top NBA team. However if a team does not take chances on up and coming players than players who could be fantastic would never be given a chance. By only considering statistics and using platforms like ours to heavily influence decisions we could be denying players chances and teams chances of getting outstanding talent in a few years.

Going off this aspect, not everything a player does for a team can be quantified or at least easily quantified. A player who improves team's attitude could be immensely valuable. There is a risk in assuming that Basketball can be solved by looking at previous performances and crunching the numbers. By trying to quantify Basketball we could cause a scenario where we only value the things we can measure. Just because there isn't an easy way to measure something doesn't mean it isn't essential in a sport.

In addition, our model gives an outcome for a game we think would happen. Despite this being what our model thinks is going to happen there is no guarantee that this will be the outcome. People could lose all their money following the outputs of our models. This is especially worrying since we can't explain why the model didn't perform poorly. With

humans considering the outcome of a match they would have a reason for why they made the recommendation they did. Our reason is that our model's weights when combined with the input says this should happen. That explanation is awful (especially to someone who just lost money) but the best that can be given.

The neural network and reinforcement model specifically lacks explainability. Given a neural network it is difficult if not impossible to understand exactly why the model is making all the decisions it is making. The use of these models in real world scenarios brings up ethical questions. These models are often not perfect meaning they are going to make mistakes. When mistakes happen there is no way to understand exactly why the model made a mistake. These mistakes can be devastating to people. In the case of Basketball analytics a team could choose to not hire a player on the output of a model. The lack of explainability adds insult to injury to mistakes made by models when mistakes occur.

Another thing to consider is the ethics of sports betting. Gambling addiction is a major issue. By utilizing sports betting predictions we implicitly promote the use of sports betting. This is not something we wanted to do. We just wanted to predict and apply data science to interesting problems.

There are also many ethical ramifications of data analysis based on our social media accounts. Many companies social media platforms such as Twitter, Facebook and, Instagram take out private data and sell it to other business for profit. These social media platform take in little account how these business will use this data. An example of this is with Facebook providing their private data to Cambridge Analytica. Cambridge Analytica is a political consulting firm that had utilized private information taken from social media platform to build a system that could profile voter and target political adds to them. This firm was used during the 2016 presidential elections and the UK Brexit vote. This private data was taken from users without there consent or knowledge. [14]

In 2018, Mark Zuckerberg, the CEO of Facebook, went to capital hill to explain why his company had allowed it user's data to be taken by Cambridge Analytica to influence the 2016 elections. The judges berated Zuckerberg for his company's inability to provide safety. They said that Facebook not only takes information from their user but those close to them as well. The core of Facebook's business is the selling of users private data to other business. There complex AI systems are trained to analyse your interest and cluster you with like minded individuals. User are not the customer's they are the product [13]

This is all to highlight the how there could be an increasing tension in society about how data is used. By creating a platform utilizing data access and predictive analytics it is important for us to consider how this data will be used and who will use it.

VII. Future

We have accomplished everything in our wall of work. However, this is not to say that more time would not bring more improvement to the project. One thing that would be interesting to add is more historical data to the NBA data. The data instead could go back 50 years instead of around 15 years. This could potentially provide more access for models and users to analyze trends. In addition to historical data, current data could be added. We could hook up producers to our data queue to real time data to ensure our database is always up to date.

In addition, in the future more time could be allocated to cleansing the database further. We could utilize more data sources to work towards a complete dataset. The betting odds data is specifically one that we would like to cleanse in the future if possible. This would help enhance the value of our product having more complete data for both the models and the data access. We could also utilize more advanced feature engineering tactics like utilizing predictive models to find the most important features. Our text analysis could work with more advanced approaches too instead of checking the positivity or negativity of each word of a tweet in isolation.

Our models could be improved too. Clusterings could factor in a larger variety of players across more time periods. It could also take in more aspects of a player's game than it does now. The reinforcement model could also be reworked to consider more data than the betting data. The neural network could be tuned to try and score better on both seen and unseen data. The ensemble in addition could have more tuning done to try and find an optimal probability of the neural network to consider for the neural network to bet on. The ensemble could also factor in both the neural network's probability of a game winning against its moneyline to determine if the neural network should vote to bet.

More models in addition could be implemented. These new models could also benefit the ensemble model. Currently there is only two models in the ensemble. Adding more could make the ensemble preform with better accuracy and robustness. Some examples of models could have been Naive Bayes, logistic regression, random forests, and a LSTM.

The API in addition could use a cursor to provide access to some of the larger tables in the database. This would provide a way to get the entirety of the data reliability by breaking it up across smaller requests. The API would also likely require some sort of authentication system, rate limited, and logging system.

The API could also use more strict testing. The API test just calls the routes and shows the output for a couple cases. It could be helpful to make the API testing automatically verify if each response for each route and each case of the route is expected.

The user interface would need to implement along with the API a login and registration system for users. This would be needed so that people would actually need to pay to use the platform. The user interface on loading can have elements pop in and pop out quickly. The loading of a page could be much better designed to ensure a smoother feel for accessing the page. The user interface could also provide ways for users to predict the outcome of current upcoming games not in the

database. The user interface could also provide ways for users to enter in games and values that they enter from scratch. In addition the overall look of the user interface could be improved and modernized. More charts and figures could also be utilized to give a more insightful experience using the interface.

The search bar at the top of the user interface is not implemented because of time constraints but could provide an easy way if implemented to allow users to get to the data they want quickly. This was left in because of the value we think it could provide. A user could for example type in Lebron James and immediately be taken to that page.

Another thing that would help the project is creating a documentation server with mkdocs. While we do our best to document everything in the READMEs where necessary, we feel that a centralized location for documentation could be helpful. In addition we could specify coding style guidelines here. This could help us in the future write better and more consistent code. This could and would have been especially been helpful since not all of our team members have a Python background.

Another project wide specific thing that could be benefits is to add an CI/CD system. It could save time and ensure our data is perfect by making actions that run on each commit. These actions could rerun our data pipeline. Then after this occurs automated testing could happen to ensure that the pipeline ran successfully and nothing we did broke what we did previously.

VIII. Conclusion

The project spanned a wide range of tasks that put our data science skill set to the test. We first found and selected quality datasets to utilize. We then designed a database that allowed for easy analytics of the dataset. We then created a data pipeline that allowed for an automated way to perform the needed steps to be able to efficiently and effectively work with our data. These steps were inserting and combining all of our datasets into our database. We then cleansed our database utilizing the NBA API to accomplish this. We then performed feature engineering to create features we found useful in working with the dataset. We collected and downloaded tweets from NBA players using the twitter API.

We also then utilized NLTK to run text analysis on these tweets by NBA players. We created a neural network that utilized the text analysis along with other data from previous games to predict the winner of an NBA game. We then created models analyzing and working with the dataset. We created a clustering model to cluster players in the hope of gaining insight on types of players. We also created a model utilizing reinforcement learning to try to find and exploit patterns in NBA betting data. We then combined the reinforcement model with the neural network to try to create a new model better than the models individually.

We then designed a REST API using swagger. This REST API both exposed aspects of our database along with the models created. This REST API was then implemented with a Python Flask server. After the REST API was created, we designed and implemented a frontend for users of the data and models. We did this using bootstrap for general styling of the website along with jQuery for making the website dynamic and calling the API.

A lot of experience and lessons were learned along the way. We learned that it is challenging to combine datasets. It can get really messy fast dealing with things like different people with the same name. We learned in data cleansing that sometimes either due to time constraints or other constraints it isn't possible to always have a perfectly cleansed database.

As for the models we learned that prediction of NBA games is both incredibly complex and difficult. If someone was able to successfully predict NBA games, they would easily be a millionaire. We needed to accept that within this limited time we had for this project we would not be able to create state of the art models.

For the API we learned that designing APIs (and interfaces in general) is difficult. A lot of revision was needed from our initial swagger design due to scenarios where creating the frontend we thought of more interesting things to add. For example we thought season statistics would be very interesting to show for players but completely missed this in the design phase. Our process for the swagger API was inefficient and caused a lot of wasted time regenerating the server and combining the new routes and models.

We also found that deploying models can be very challenging. It was very important for us to save the models and load the models using the same library versions. Eventually we figured out how to ensure we were always using the most up to date versions of our model libraries. However even updating some packages to the same latest version can be tricky in Python. The biggest takeaway from this for us was to consider things like getting the same version of Tensorflow or Sklearn on the machine you are training the model and deploying the model before you create, train, and save your model.

We also found that designing a user interface and implementing a user interface is complex and time consuming. The amount of time we expected to spend creating the user interface was a fraction of the actual time needed. Estimation in software is always hard. This was especially true since only one of the members in our group had worked with jQuery or had experience with calling APIs.

Overall, we were able to get experience from the entire process of working with a real world dataset. We did every step from nitty gritty data extraction to designing charts in the user interface. We were also able to get more experience working in a professional environment working on a large project requiring communication skills.

References

[1] Torre, P. S., & Haberstroh, T. (2014, October 6). Players union looks at data protection. ESPN. https://www.espn.com/nba/story/_/id/11652885/nba-players-union-wants-ensure-privacy-data-collection.

[2] Swar. (n.d.). swar/nba_api. GitHub. https://github.com/swar/nba_api.

[3] America goes all in: The state of play in the US sports betting market. SportsPro. (n.d.). https://www.sportspromedia.com/from-the-magazine/us-sports-betting-market-gambling-revenue-states-2021-data.

[4] Yahoo! (n.d.). Simplebet CEO on the future of sports betting. Yahoo! Finance. https://finance.yahoo.com/video/simplebet-ceo-future-sports-betting-225123271.html.
[5] Bucquet, A., & Sarukkai, V. (n.d.). The Bank is Open: AI in Sports Gambling. stanford.edu. http://cs229.stanford.edu/proj2018/report/3.pdf
[6] Official NBA Stats. NBA Stats. (n.d.). https://www.nba.com/stats/.
[7] Sports Analytics Market Size, Trends: Industry Analysis Report 2021 to 2026 – Mordor Intelligence. Sports Analytics Market Size, Trends — Industry Analysis Report 2021 to 2026 – Mordor Intelligence. (n.d.). https://www.mordorintelligence.com/industry-reports/sports-analytics-market.
[8] Namecheap. analyticsbasketball. (n.d.). https://www.namecheap.com/domains/registration/results/?domain=analyticsbasketball.
[9] World's largest sports and humanitarian event builds legacy of inclusion with data-driven technology. SAS. (n.d.). https://www.sas.com/en_hk/customers/special-olympics-world-games-abu-dhabi.html.
[10] Google. (n.d.). Colab Pro. Google. https://colab.research.google.com/signup.
[11] Lewis, M. (2017, November 28). Tag: Predictive Sports Analytics. Sports Analytics Research from Mike Lewis. https://scholarblogs.emory.edu/esma/tag/predictive-sports-analytics/.
[12] AWS Pricing Calculator. (n.d.). https://calculator.aws/#/createCalculator/AuroraPostgreSQL.
[13] Singer, N. (2018, April 11). What You Don't Know About How Facebook Uses Your Data. The New York Times. https://www.nytimes.com/2018/04/11/technology/facebook-privacy-hearings.html.
[14] Guardian News and Media. (2018, March 17). Revealed: 50 million Facebook profiles harvested for Cambridge Analytica in major data breach. The Guardian. https://www.theguardian.com/news/2018/mar/17/cambridge-analytica-facebook-influence-us-election.
[15] G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.
[16] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.
[17] I. S. Jacobs and C. P. Bean, "Fine particles, thin films and exchange anisotropy," in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.
[18] K. Elissa, "Title of paper if known," unpublished.
[19] R. Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev., in press.
[20] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].
[21] M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.