NBA Crowdsourcing Comparisons

Akshay Sathiya, Pranav Khorana, Rahul Chawla, Sanket Manjesh
CS 6220 A - Big Data Systems and Analysis

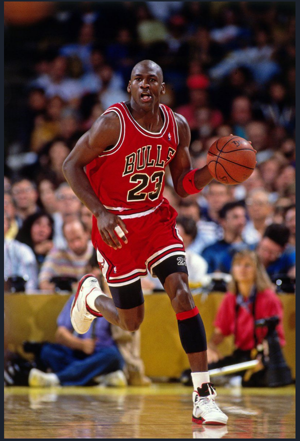
Objective

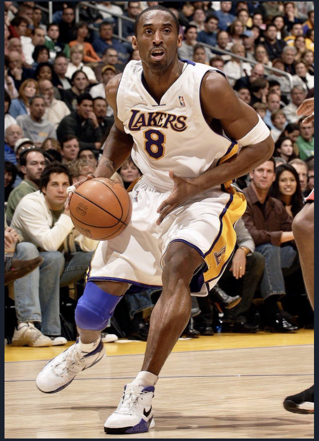
Use crowdsourcing to help solve a problem that cannot be solved by machine intelligence alone.

The Problem

How to determine which NBA player is “better”?

- Who was better, Michael Jordan in his prime (1990-1991) or Kobe Bryant in his prime (2005-2006)?





Statistics

Players can be compared by their statistics for a particular season, or across their entire careers

- Statistics
  - Points per game
  - Assists per game
    - Passes that directly resulted in a teammate scoring
  - Rebounds per game
    - Obtaining the ball after a missed shot
  - Blocks per game
  - Steals per game
  - Turnovers per game
  - Minutes played per game
  - Games played that season
  - Shooting percentage

Comparisons

Not a trivial task!

- Player stats and accolades are not necessarily comparable.
- Players have different positions, eras, teammates, and opponents.

Comparisons

Comparisons can be quite difficult.

- 2004-2005 MVP race
  - Steve Nash
    - 15.5 points per game
    - 11.5 assists per game
  - Shaquille O’Neal
    - 22.9 points per game
    - 10.4 rebounds per game
    - 2.3 blocks per game
  - Steve Nash wins MVP!
- Still debated to this day!

Crowdsourcing System

We believe that machine intelligence can solve this problem of determining which of two NBA players is better, with a little help from crowdsourcing.

Our crowdsourcing system will be a mobile app to collect user responses for player comparisons.

How It Works

Step 1: Get user vote

- User is presented with a random pair of NBA players and their stats (as listed earlier), then submits their vote for who they think is better.
- User can then see voting results and participate in a chat that shows the comments of other users who have cast their vote and weighed in on the debate. This will be the incentive.

How It Works

Step 2: Resolve user votes

- User votes for a particular pair of NBA players will be resolved by a majority vote to form labels.
  - 0 represents that the first player is better than the second, 1 represents that the second player is better than the first.

How It Works

Step 3: Create datasets

- A dataset can be assembled from the collected data, where the features are represented as vectors, in the format shown below.

[ppg1 apg1 rpg1 bpg1 spg1 tov1 gp1 mp1 sp1 ppg2 apg2 rpg2 bpg2 spg2 tov2 gp2 mp2 sp2]

- The label is 0 or 1, as described earlier.

How It Works

Step 4: Train/Validate ML Model(s)

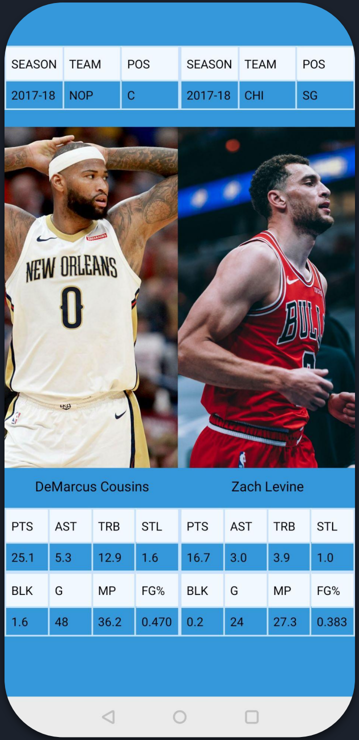
- A machine learning model(s) can be trained on this dataset to observe the extent to which machine intelligence can solve the original problem with support from crowdsourcing.
- Models
    - Logistic Regression Classifier
    - Random Forest Classifier

Impact

- Our product will determine rankings of popular NBA players, something that cannot be objectively determined with current datasets
- Our data is crowdsourced, thus resulting in up-to-date, relevant predictions that reflect current trends
- The final product will result in a centralized place for fans to contribute to popular basketball information and discussions
    - Such basketball platforms that take in user input to make wide scale predictions do not currently exist

User View

- Display 2 NBA players side-by-side with stats
    - Button to view player's achievements for that season
    - Add a view for forum/chat
- Taps on image to vote for a player
    - Vote sent to Firebase Realtime Database
- Generate random pair of players along with stats from Flask server
    - Realtime data from basketball-reference.com



SEASON	TEAM	POS	SEASON	TEAM	POS
2017-18	NOP	C	2017-18	CHI	SG

DeMarcus Cousins

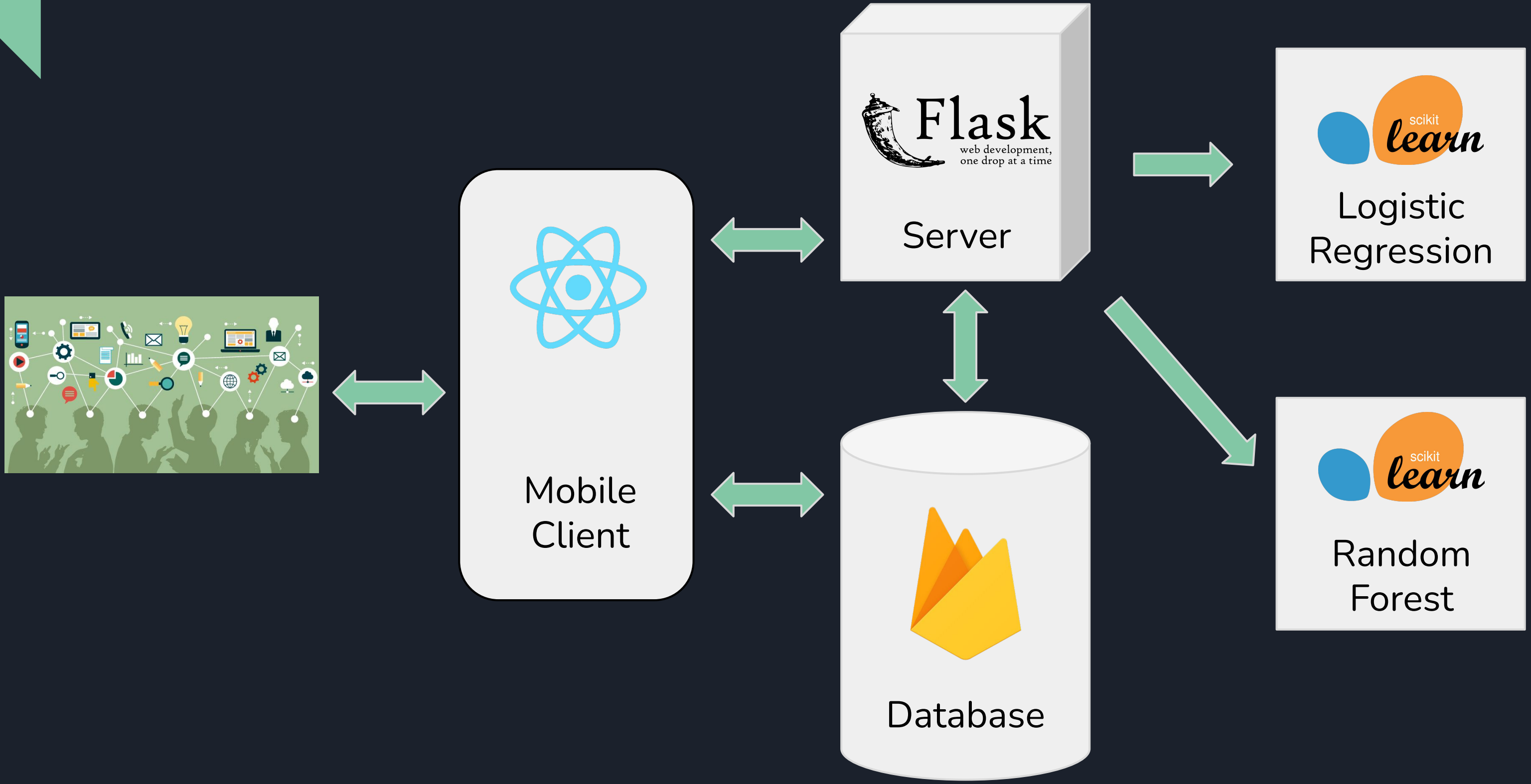
Zach Levine

PTS	AST	TRB	STL	PTS	AST	TRB	STL
25.1	5.3	12.9	1.6	16.7	3.0	3.9	1.0
BLK	G	MP	FG%	BLK	G	MP	FG%
1.6	48	36.2	0.470	0.2	24	27.3	0.383

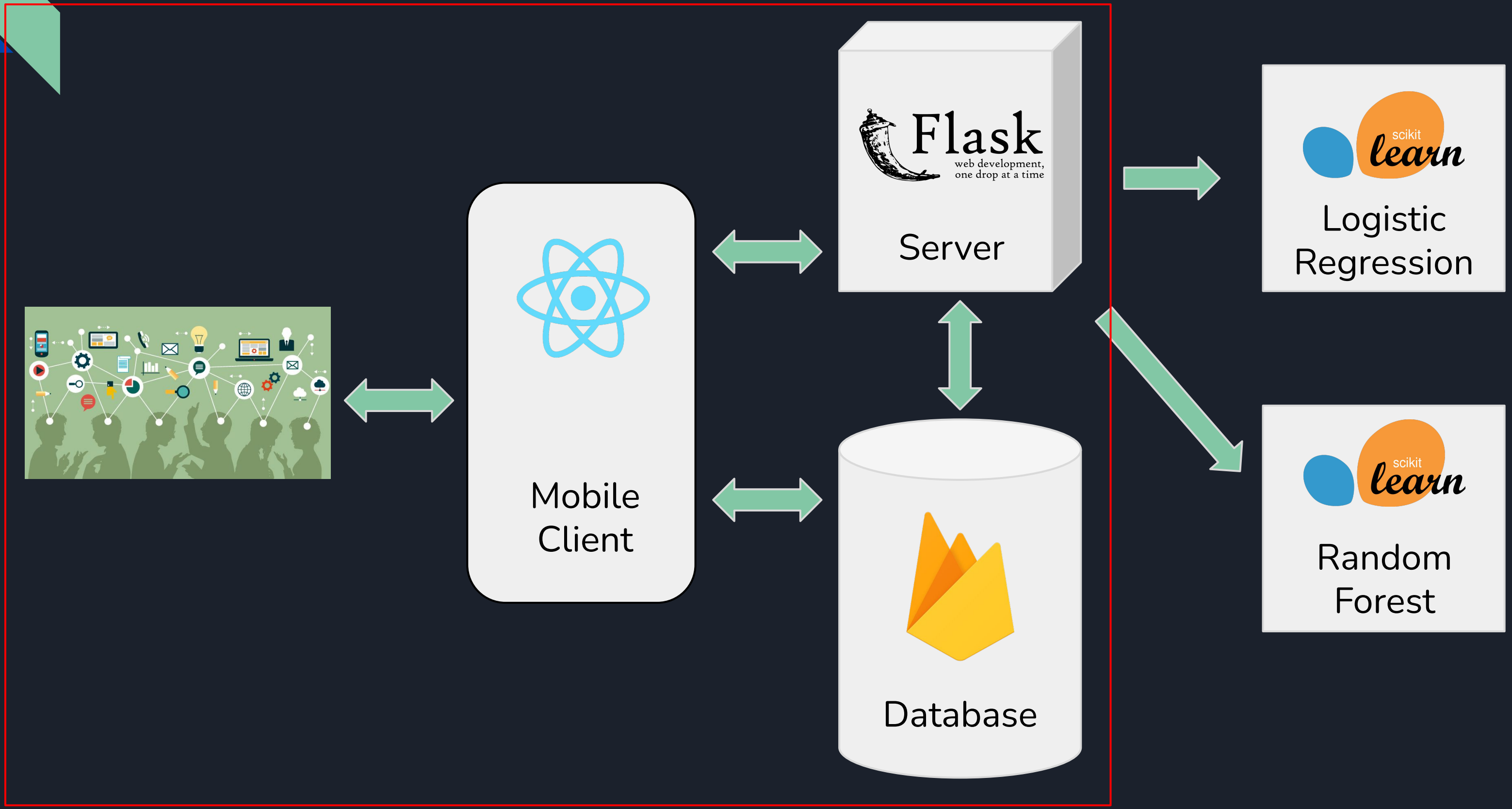
Tech Stack

- We are creating an application using React Native as the front-end technology for designing the components of the application
- Data for all players, users, and information related to voting for each user will be stored in a FireBase database
- We will use a Flask backend as a rest API to retrieve information from our FireBase database and feed to the front-end application after receiving requests from the front-end application using Axios.js
- ML models will be developed using Scikit-learn

System Architecture



Demo - Part 1

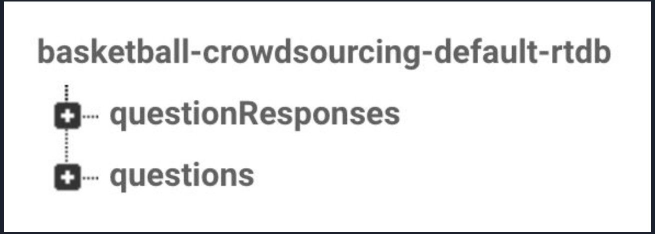


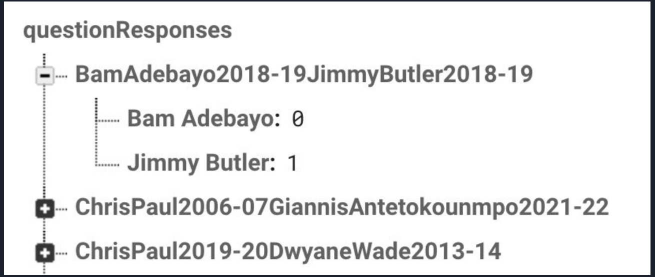
Flask
web development, one drop at a time
Server
Mobile Client
Database
scikit learn
Logistic Regression
scikit learn
Random Forest

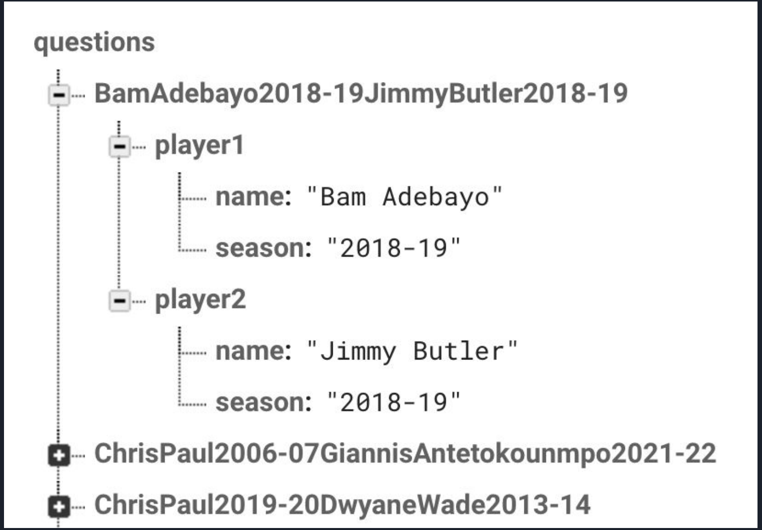
Open Source Packages (TODO)

- Firebase package to connect front-end application with the database and send/receive data for questions, players, votes
- FlatList package to render images side by side
- Table and Row components to display stats
- Gifted Chat package for clean chat UI
- Axios to make calls to Flask server from mobile app
- Used React navigation to switch between screens
- useFocusEffect to trigger events while switching

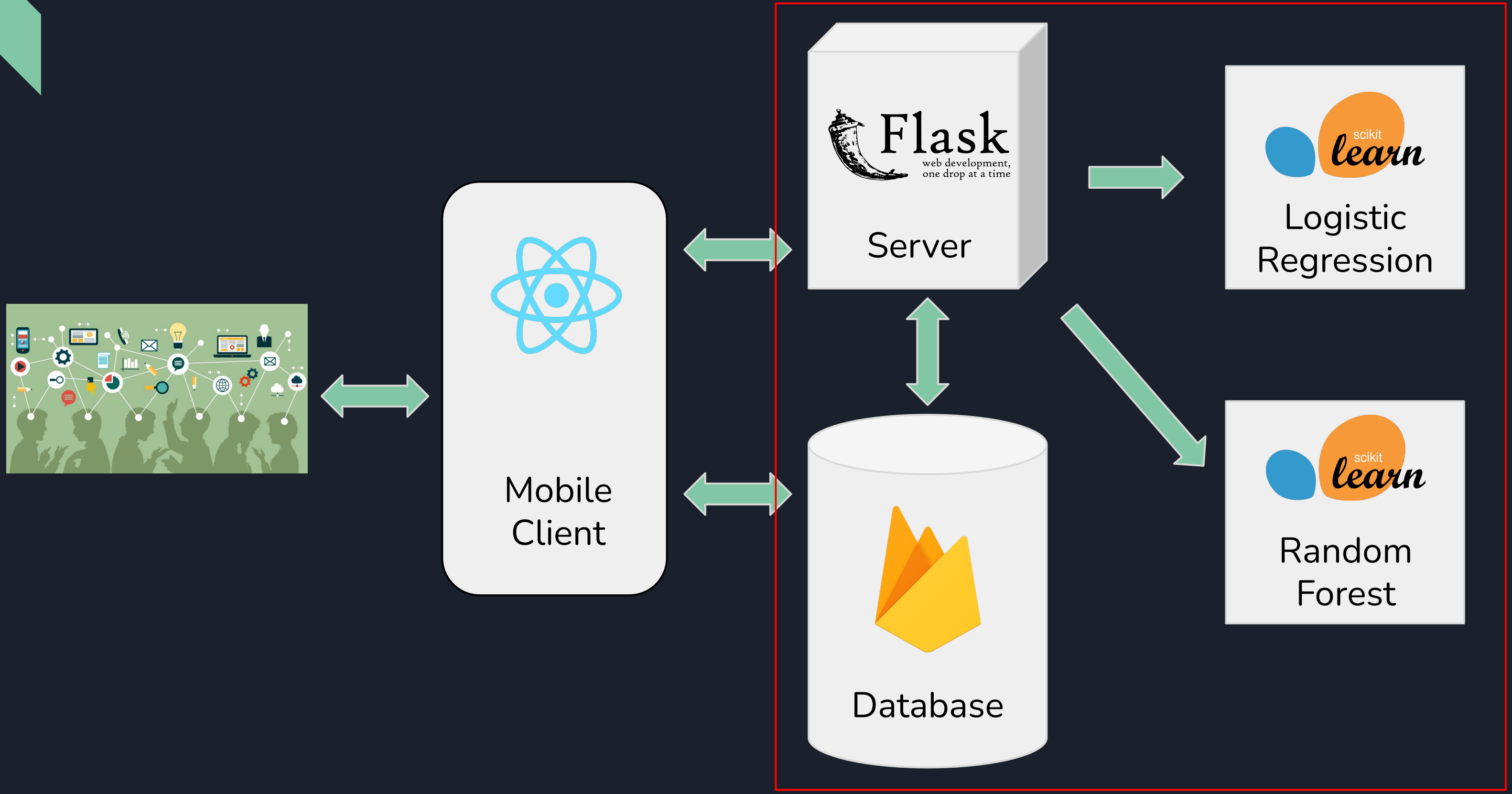
User Interface - Database Schema


basketball-crowdsourcing-default-rtdb
- questionResponses
- questions


questionResponses
- BamAdebayo2018-19JimmyButler2018-19
  - Bam Adebayo: 0
  - Jimmy Butler: 1
- ChrisPaul2006-07GiannisAntetokounmpo2021-22
- ChrisPaul2019-20DwyaneWade2013-14


questions
- BamAdebayo2018-19JimmyButler2018-19
  - player1
    - name: "Bam Adebayo"
    - season: "2018-19"
  - player2
    - name: "Jimmy Butler"
    - season: "2018-19"
- ChrisPaul2006-07GiannisAntetokounmpo2021-22
- ChrisPaul2019-20DwyaneWade2013-14

Demo - Part 2


Flask
web development, one drop at a time
Server
scikit learn
Logistic Regression
scikit learn
Random Forest
Mobile Client
Database

Open Source Packages

For this part of the project, we used the following open source packages

- basketball-reference-scraper
  - https://pypi.org/project/basketball-reference-scraper/
  - https://github.com/vishaalagartha/basketball_reference_scraper
- Flask
  - https://pypi.org/project/Flask/
  - https://flask.palletsprojects.com/en/2.0.x/
- scikit-learn
  - https://pypi.org/project/scikit-learn/
  - https://scikit-learn.org/stable/
- pandas
  - https://pypi.org/project/pandas/
  - https://pandas.pydata.org/

Flask API

- Created endpoints to retrieve statistics about the players
  - Team
  - Position
  - Points
  - AST, TRB, STL, BLK, Etc

- “/stats/player1/player2/season1/season2”
  - Retrieves all statistics, in JSON format for both players for the specified seasons
  - Via a python library, “basketball_reference_scraper”
    - Returns a dictionary of player stats, which are then concatenated for all players and seasons, and returned as a JSON

- Passes information to the the database, which then is utilized by the react-native front-end

Machine Learning Pipeline

- Three parts
  - Data collection
    - Pull data from the Firebase database
  - Data processing
    - Prepare dataset for training machine learning models
    - Save dataset
  - Model training
    - Train and test models
    - Save models

Data Collection

- Firebase database contains several crowdsources votes on who is better between many pairs of two players
- We developed a script to collect the data from the Firebase database and assemble it into a tabular format (pandas DataFrame)
- We can save this as a CSV file for future use

Data Processing

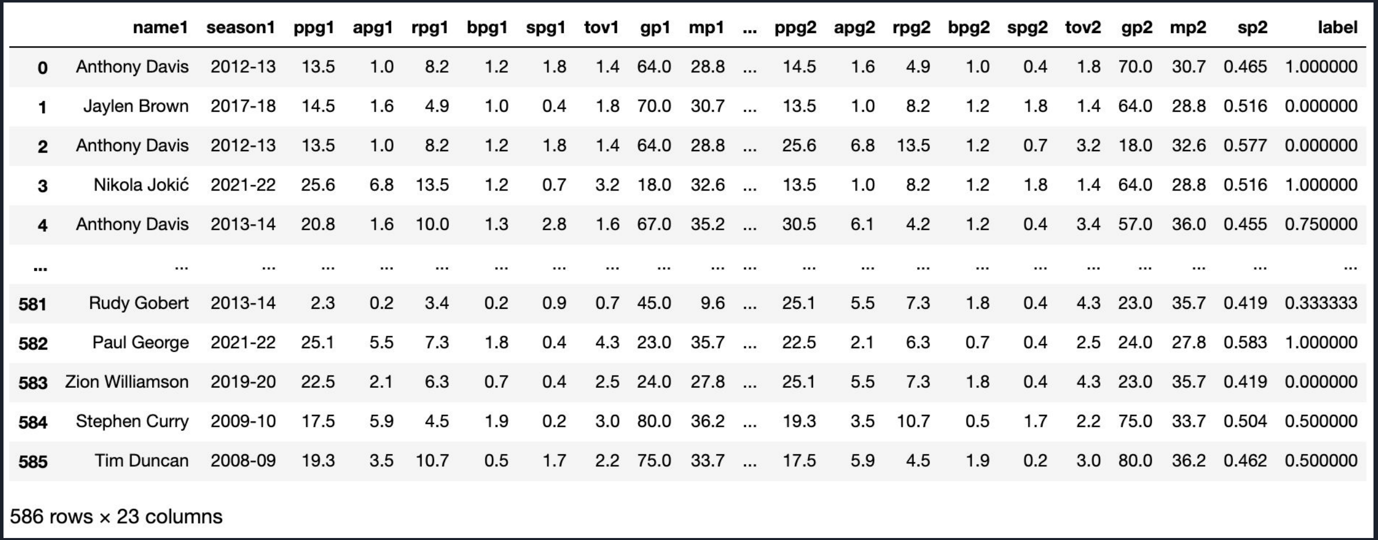
- After we collect the data, we extract feature vectors

[ppg1 apg1 rpg1 bpg1 spg1 tov1 gp1 mp1 sp1 ppg2 apg2 rpg2 bpg2 spg2 tov2 gp2 mp2 sp2]

- Then, we flip the feature vector so the model is not have sequential bias

[ppg2 apg2 rpg2 bpg2 spg2 tov2 gp2 mp2 sp2 ppg1 apg1 rpg1 bpg1 spg1 tov1 gp1 mp1 sp1]

Data



	name1	season1	ppg1	apg1	rpg1	bpg1	spg1	tov1	gp1	mp1	...	ppg2	apg2	rpg2	bpg2	spg2	tov2	gp2	mp2	sp2	label
0	Anthony Davis	2012-13	13.5	1.0	8.2	1.2	1.8	1.4	64.0	28.8	...	14.5	1.6	4.9	1.0	0.4	1.8	70.0	30.7	0.465	1.000000
1	Jaylen Brown	2017-18	14.5	1.6	4.9	1.0	0.4	1.8	70.0	30.7	...	13.5	1.0	8.2	1.2	1.8	1.4	64.0	28.8	0.516	0.000000
2	Anthony Davis	2012-13	13.5	1.0	8.2	1.2	1.8	1.4	64.0	28.8	...	25.6	6.8	13.5	1.2	0.7	3.2	18.0	32.6	0.577	0.000000
3	Nikola Jokić	2021-22	25.6	6.8	13.5	1.2	0.7	3.2	18.0	32.6	...	13.5	1.0	8.2	1.2	1.8	1.4	64.0	28.8	0.516	1.000000
4	Anthony Davis	2013-14	20.8	1.6	10.0	1.3	2.8	1.6	67.0	35.2	...	30.5	6.1	4.2	1.2	0.4	3.4	57.0	36.0	0.455	0.750000
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
581	Rudy Gobert	2013-14	2.3	0.2	3.4	0.2	0.9	0.7	45.0	9.6	...	25.1	5.5	7.3	1.8	0.4	4.3	23.0	35.7	0.419	0.333333
582	Paul George	2021-22	25.1	5.5	7.3	1.8	0.4	4.3	23.0	35.7	...	22.5	2.1	6.3	0.7	0.4	2.5	24.0	27.8	0.583	1.000000
583	Zion Williamson	2019-20	22.5	2.1	6.3	0.7	0.4	2.5	24.0	27.8	...	25.1	5.5	7.3	1.8	0.4	4.3	23.0	35.7	0.419	0.000000
584	Stephen Curry	2009-10	17.5	5.9	4.5	1.9	0.2	3.0	80.0	36.2	...	19.3	3.5	10.7	0.5	1.7	2.2	75.0	33.7	0.504	0.500000
585	Tim Duncan	2008-09	19.3	3.5	10.7	0.5	1.7	2.2	75.0	33.7	...	17.5	5.9	4.5	1.9	0.2	3.0	80.0	36.2	0.462	0.500000

586 rows × 23 columns

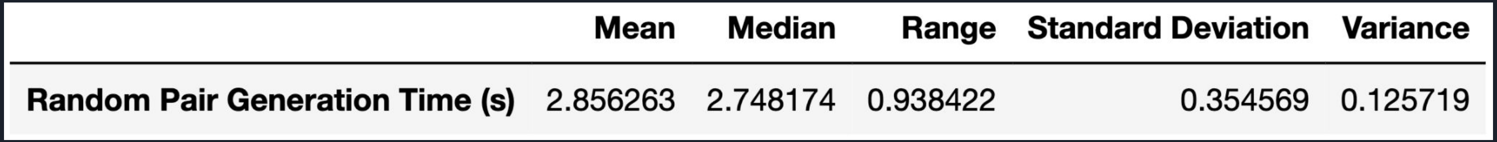
Machine Learning

- We train the following machine learning models (80-20 train/test split)
  - Logistic regression classifier
  - Random forest classifier
    - Considers all features for splits
    - 10 estimators, max depth 10
- Models are saved for future use

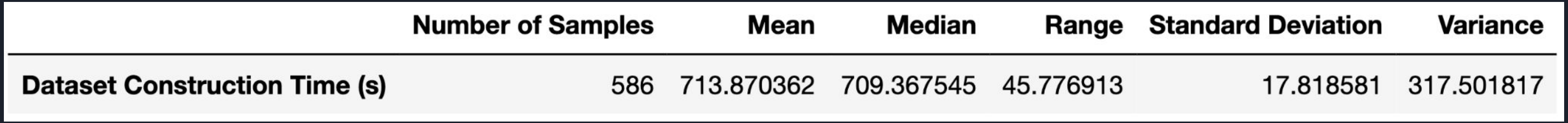
Experiments

- Random pair generation time
  - Measure mean, median, range, standard deviation, and variance of times it takes to get data for two random NBA players to compare.
- Construct dataset time
  - Measure mean, median, range, standard deviation, and variance of times for constructing the dataset.
- Model training
  - Train logistic regression and random forest classifiers (fit time and MSE, without scaling and with standard scaling).
- Model prediction time
  - Measure mean, median, range, standard deviation, and variance of prediction times for both logistic regression and random forest classifiers.

Experiment Results

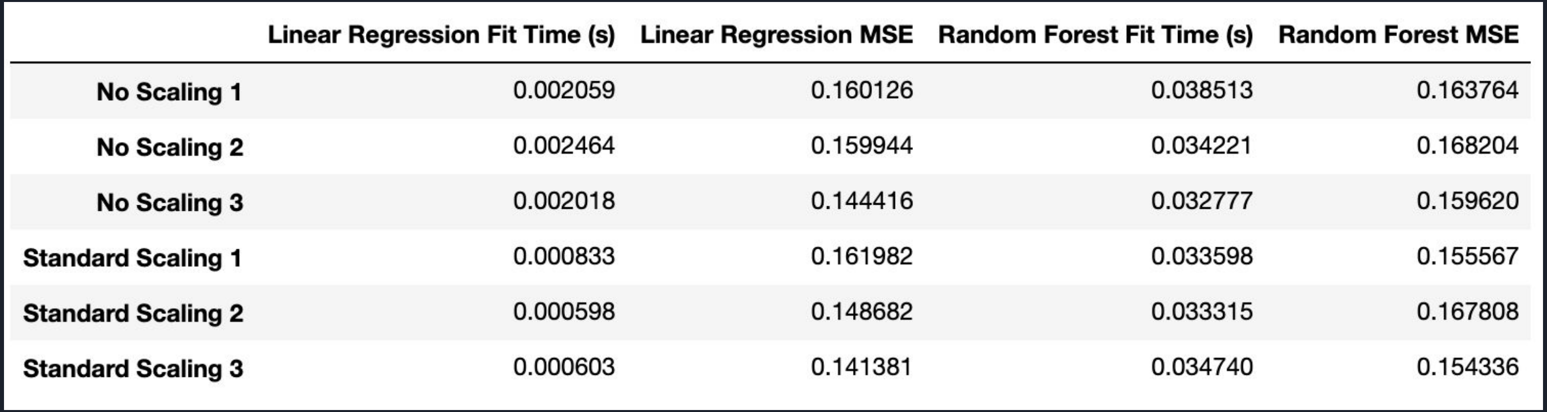


	Mean	Median	Range	Standard Deviation	Variance
Random Pair Generation Time (s)	2.856263	2.748174	0.938422	0.354569	0.125719



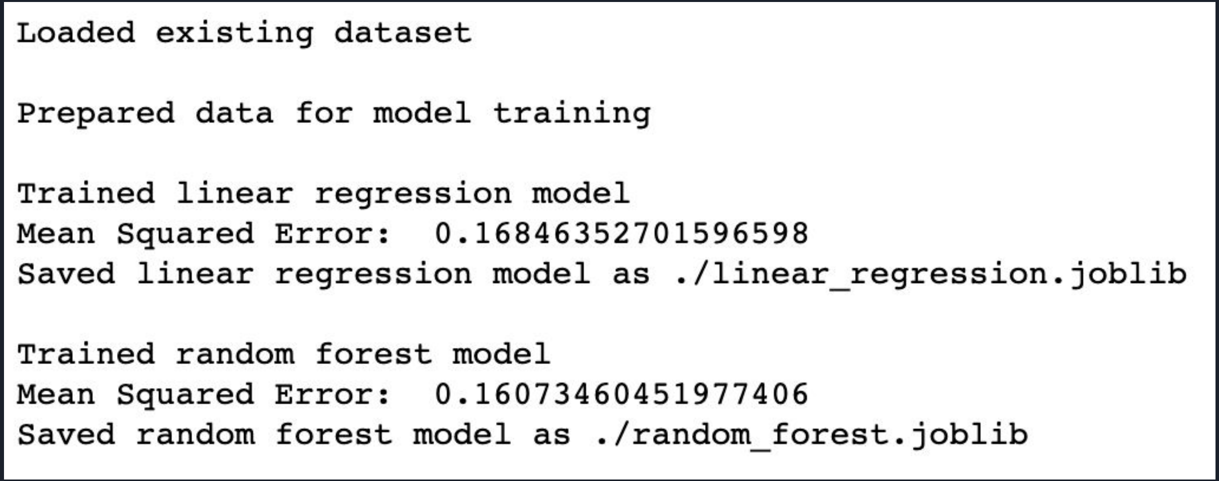
	Number of Samples	Mean	Median	Range	Standard Deviation	Variance
Dataset Construction Time (s)	586	713.870362	709.367545	45.776913	17.818581	317.501817

Experiment Results



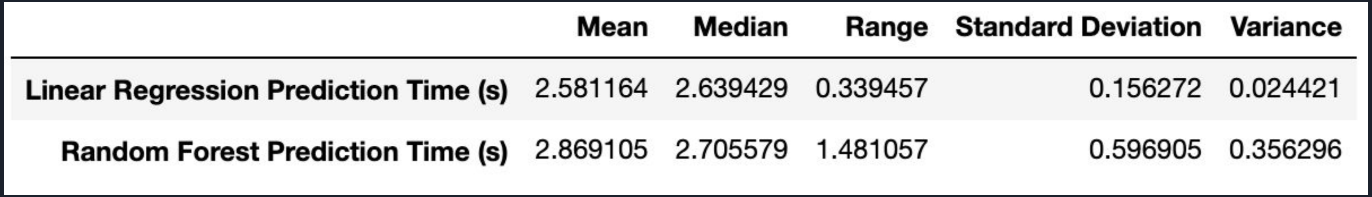
	Linear Regression Fit Time (s)	Linear Regression MSE	Random Forest Fit Time (s)	Random Forest MSE
No Scaling 1	0.002059	0.160126	0.038513	0.163764
No Scaling 2	0.002464	0.159944	0.034221	0.168204
No Scaling 3	0.002018	0.144416	0.032777	0.159620
Standard Scaling 1	0.000833	0.161982	0.033598	0.155567
Standard Scaling 2	0.000598	0.148682	0.033315	0.167808
Standard Scaling 3	0.000603	0.141381	0.034740	0.154336

Trained Models with Standard Scaling



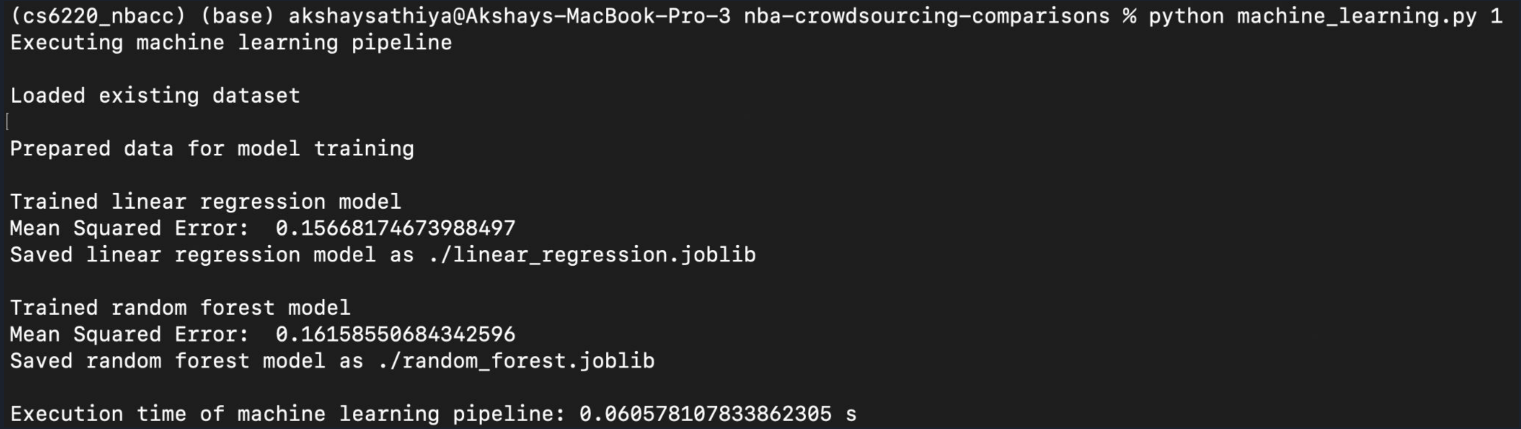
```
Loaded existing dataset

Prepared data for model training

Trained linear regression model
Mean Squared Error:  0.16846352701596598
Saved linear regression model as ./linear_regression.joblib

Trained random forest model
Mean Squared Error:  0.16073460451977406
Saved random forest model as ./random_forest.joblib
```

Experiment Results



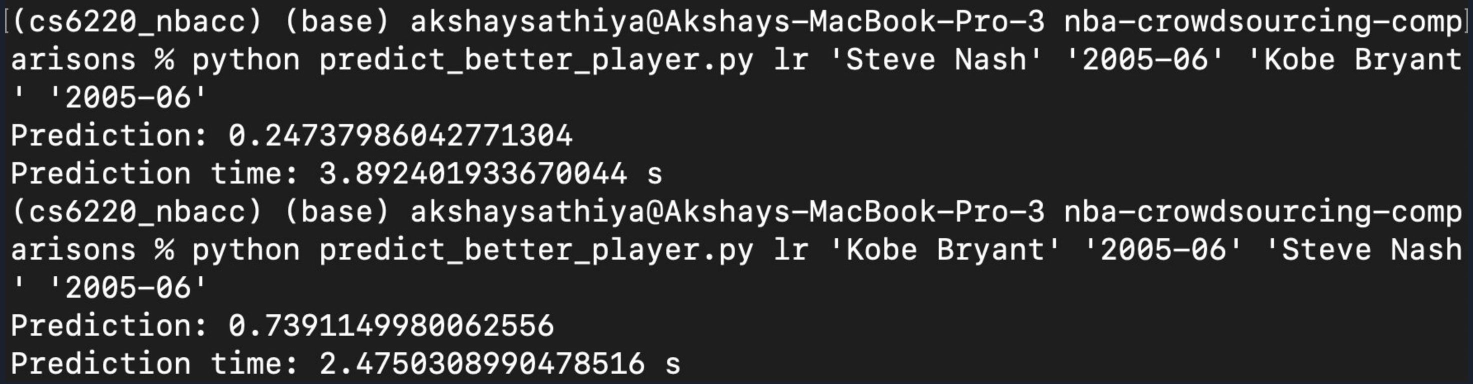
	Mean	Median	Range	Standard Deviation	Variance
Linear Regression Prediction Time (s)	2.581164	2.639429	0.339457	0.156272	0.024421
Random Forest Prediction Time (s)	2.869105	2.705579	1.481057	0.596905	0.356296

ML Pipeline and Predictions



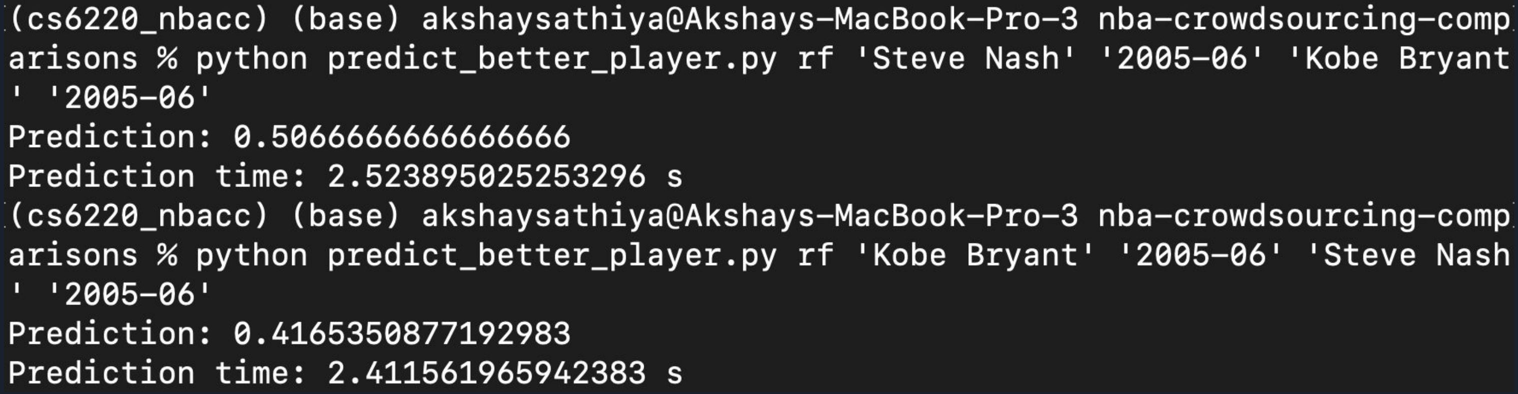
```
(cs6220_nbacc) (base) akshaysathiya@Akshays-MacBook-Pro-3 nba-crowdsourcing-comparisons % python machine_learning.py 1
Executing machine learning pipeline

Loaded existing dataset

Prepared data for model training

Trained linear regression model
Mean Squared Error:  0.15668174673988497
Saved linear regression model as ./linear_regression.joblib

Trained random forest model
Mean Squared Error:  0.16158550684342596
Saved random forest model as ./random_forest.joblib

Execution time of machine learning pipeline: 0.060578107833862305 s
```

ML Pipeline and Predictions



```
[(cs6220_nbacc) (base) akshaysathiya@Akshays-MacBook-Pro-3 nba-crowdsourcing-comp]
arisons % python predict_better_player.py lr 'Steve Nash' '2005-06' 'Kobe Bryant
' '2005-06'
Prediction: 0.24737986042771304
Prediction time: 3.892401933670044 s
(cs6220_nbacc) (base) akshaysathiya@Akshays-MacBook-Pro-3 nba-crowdsourcing-comp
arisons % python predict_better_player.py lr 'Kobe Bryant' '2005-06' 'Steve Nash
' '2005-06'
Prediction: 0.7391149980062556
Prediction time: 2.4750308990478516 s
```

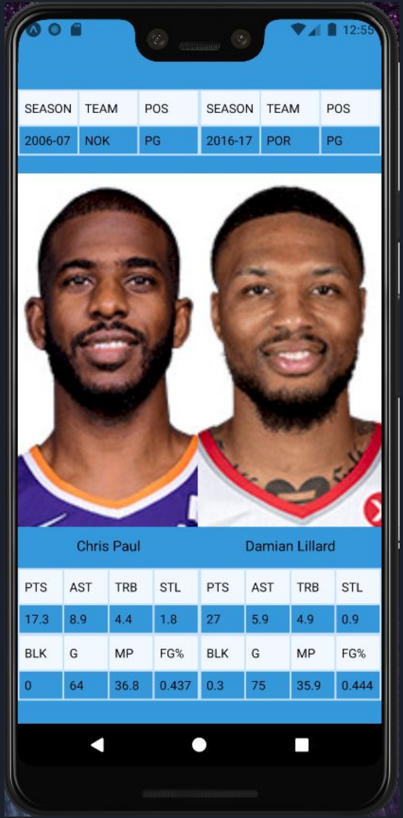
ML Pipeline and Predictions



```
(cs6220_nbacc) (base) akshaysathiya@Akshays-MacBook-Pro-3 nba-crowdsourcing-comp
arisons % python predict_better_player.py rf 'Steve Nash' '2005-06' 'Kobe Bryant
' '2005-06'
Prediction: 0.5066666666666666
Prediction time: 2.523895025253296 s
(cs6220_nbacc) (base) akshaysathiya@Akshays-MacBook-Pro-3 nba-crowdsourcing-comp
arisons % python predict_better_player.py rf 'Kobe Bryant' '2005-06' 'Steve Nash
' '2005-06'
Prediction: 0.4165350877192983
Prediction time: 2.411561965942383 s
```

Summary and Impact

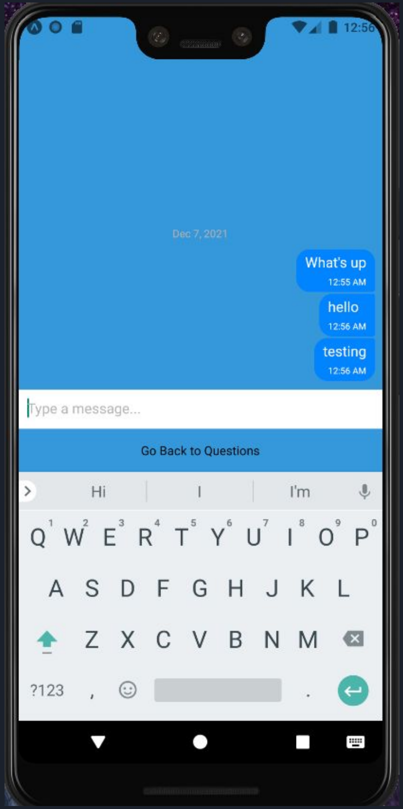
- We developed a crowdsourcing platform and machine learning models to determine who is better between two NBA players.
    - The crowdsourcing platform serves a sa centralized portal through which fans can answer some of the most trending and long-lasting debates as well as interact with other fans.
- Our crowdsourcing platform collected the results from users, which were used to train machine learning models.
    - These machine learning models can be used to determine the best NBA players.
        - Predictions for MVP, Rookie of the Year, Finals MVP
        - Predictions for certain player matchups and playoff series



SEASON	TEAM	POS	SEASON	TEAM	POS
2006-07	NOK	PG	2016-17	POR	PG

Chris Paul Damian Lillard

PTS	AST	TRB	STL	PTS	AST	TRB	STL
17.3	8.9	4.4	1.8	27	5.9	4.9	0.9
BLK	G	MP	FG%	BLK	G	MP	FG%
0	64	36.8	0.437	0.3	75	35.9	0.444

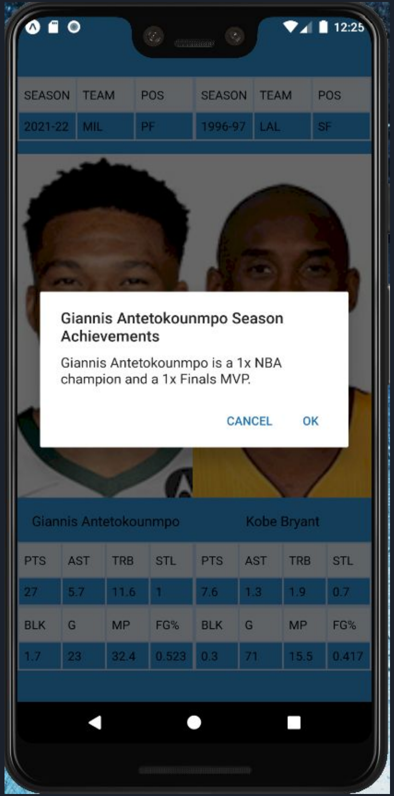


Dec 7, 2021

What's up
12:55 AM

hello
12:56 AM

testing
12:56 AM



SEASON	TEAM	POS	SEASON	TEAM	POS
2021-22	MIL	PF	1996-97	LAL	SF

Giannis Antetokounmpo Season Achievements

Giannis Antetokounmpo is a 1x NBA champion and a 1x Finals MVP.

Giannis Antetokounmpo Kobe Bryant

PTS	AST	TRB	STL	PTS	AST	TRB	STL
27	5.7	11.6	1	7.6	1.3	1.9	0.7
BLK	G	MP	FG%	BLK	G	MP	FG%
1.7	23	32.4	0.523	0.3	71	15.5	0.417

Machine Learning Model Output Explanation

- The two machine learning models used in this project are a linear regression and a random forest regressor.
- Each regressor outputs the percentage of votes player 1 is expected to get. This is a useful measure for determining if player 1 is better than player 2.
- If the prediction (regression output) is less than 0.5, then player 1 is worse than player 2.
- If the prediction (regression output) is equal to 0.5, then both players are equal.
- If the prediction (regression output) is greater than 0.5, then player 1 is better than player 2.

Further Experimentation

- Comparisons
  - In order to better understand the situations where our model works well and does not work as well, we will examine prediction performance with respect to these two factors.
    - Position
      - How does our model perform when comparing players of the same/different positions?
    - Decade
      - How does our model perform when comparing players that played in different decades?
    - Symmetry
      - If we switch the order of player 1 and player 2 in which they are passed to the models, does the prediction change accordingly?
    - Accuracy
      - Are the predictions by the models correct?
  - We will manually test three player-to-player comparisons for each case, and record the prediction and the time elapsed for the prediction.
    - Same position, same decade
    - Same position, different decade
    - Different position, same decade
    - Different position, different decade

Further Experimentation

- Same position, same decade (Random Forest Model)
- Kobe Bryant (SG) 2003-04, Tracy McGrady (SG) 2003-04
    - Comparison 1: 0.502 (3.991 s)
    - Comparison 2: 0.654 (2.080 s)
- Stephen Curry (PG) 2017-18, Rajon Rondo (PG) 2019-20
    - Comparison 1: 0.510 (2.891 s)
    - Comparison 2: 0.476 (2.048 s)
- Russell Westbrook (PG) 2016-17, James Harden(PG) 2019-20
    - Comparison 1: 0.381 (3.728 s)
    - Comparison 2: 0.365 (2.002 s)

Further Experimentation

- Same position, same decade (Linear Regression Model)
- Kobe Bryant (SG) 2003-04, Tracy McGrady (SG) 2003-04
  - Comparison 1: 0.629 (1.655 s)
  - Comparison 2: 0.525 (2.095 s)
- Stephen Curry (PG) 2017-18, Rajon Rondo (PG) 2019-20
  - Comparison 1: 0.667, (2.107 s)
  - Comparison 2: 0.435, (2.023 s)
- Russell Westbrook (PG) 2016-17, James Harden(PG) 2019-20
  - Comparison 1: 0.547 (2.744 s)
  - Comparison 2: 0.500 (2.750 s)

Further Experimentation

- Same position, different decade (Random Forest)
  - Stephen Curry (PG) 2015-16, Allen Iverson (PG) 2000-01
    - Comparison 1: 0.642, (2.046 s)
    - Comparison 2: 0.441, (2.248 s)
  - LeBron James (SF) 2009-10, Scottie Pippen (SF) 1994-95
    - Comparison 1: 0.462, (3.262 s)
    - Comparison 2: 0.465, (2.356 s)
  - Michael Jordan (1990-91), Kobe Bryant (SG) 2005-06
    - Comparison 1: 0.633, (3.159 s)
    - Comparison 2: 0.259, (2.226 s)
- Results are symmetrical when swapping player 1 and player 2 order. Predictions for the first and third comparisons are accurate, while the second is questionable. The first comparison can go either way.

Further Experimentation

- Same position, different decade (Linear Regression)
  - Stephen Curry (PG) 2015-16, Allen Iverson (PG) 2000-01
    - Comparison 1: 0.499, (2.265 s)
    - Comparison 2: 0.514, (2.501 s)
  - LeBron James (SF) 2009-10, Scottie Pippen (SF) 1994-95
    - Comparison 1: 0.329, (2.299 s)
    - Comparison 2: 0.643, (2.454 s)
  - Michael Jordan (1990-91), Kobe Bryant (SG) 2005-06
    - Comparison 1: 0.681, (2.190 s)
    - Comparison 2: 0.344, (2.127 s)
- Results are symmetrical when swapping player 1 and player 2 order. Predictions for the first and third comparisons are accurate, while the second is questionable. The first comparison can go either way.

Further Experimentation

- Different position, same decade (Random Forest)
  - LeBron James, 2017-2018 (F), Ben Simmons, 2018-2019 (PG)
    - Comparison 1: 0.719 (1.83s)
    - Comparison 2: 0.398 (1.42s)
  - James Harden, 2017-2018 (SG), Kawhi Leonard, 2018-2019 (PF)
    - Comparison 1: .540 (2.79s)
    - Comparison 2: .660 (1.46s)
  - Joel Embiid, 2017-2018 (C), Paul George, 2018-2019 (SF)
    - Comparison 1: .555 (2.65s)
    - Comparison 2: .317 (1.34s)
- Results appear to be symmetrical when swapping order of player inputs. The predictions are accurate for the first case, but not for the second and third cases.

Further Experimentation

- Different position, same decade (Linear Regression)
  - LeBron James, 2017-2018 (F), Ben Simmons, 2018-2019 (PG)
    - Comparison 1: 0.523 (1.41s)
    - Comparison 2: 0.367 (1.40s)
  - James Harden, 2017-2018 (SG), Kawhi Leonard, 2018-2019 (PF)
    - Comparison 1: .642 (1.49s)
    - Comparison 2: .436 (1.54s)
  - Joel Embiid, 2017-2018 (C), Paul George, 2018-2019 (SF)
    - Comparison 1: .436 (1.31s)
    - Comparison 2: .634 (1.34s)
- Results appear to be symmetrical when swapping order of player inputs. The predictions are accurate for all three pairs!

Further Experimentation

- Different position, different decade (Random Forest)
  - Kobe Bryant (SG) 2005-06, Scottie Pippen (SF) 1994-95
    - Comparison 1: 0.269, (2.584 s)
    - Comparison 2: 0.626, (2.271 s)
  - Allen Iverson (SG) 2000-01, Kevin Durant (PF) 2011-12
    - Comparison 1: 0.182, (2.265 s)
    - Comparison 2: 0.491, (2.212 s)
  - Michael Jordan (SG) 2002-03, LeBron James (SF) 2012-13
    - Comparison 1: 0.237, (2.268 s)
    - Comparison 2: 0.870, (2.313 s)
- Results are symmetric, when swapping player 1 and player 2 order, for the first and third cases, but not for the second. The predictions are accurate for the third comparison but not the first two cases.

Further Experimentation

- Different position, different decade (Linear Regression)
  - Kobe Bryant (SG) 2005-06, Scottie Pippen (SF) 1994-95
    - Comparison 1: .383 (1.56s)
    - Comparison 2: .601 (1.61s)
  - Allen Iverson (SG) 2000-01, Kevin Durant (PF) 2011-12
    - Comparison 1: .634 (1.55s)
    - Comparison 2: .347 (1.56s)
  - Michael Jordan (SG) 2002-03, LeBron James (SF) 2012-13
    - Comparison 1: 0.437 (1.63s)
    - Comparison 2: .562 (1.60s)
- Results appear to be symmetrical when swapping order of player inputs
- Pair one is inaccurate, although pair two and pair three are accurate

Analysis

- Linear Regression
  - The linear regression model maintained symmetry on most of its results. When the player input order was switched, the prediction/regression output changed accordingly, so the two complementing predictions would have a sum close to 1.
  - The linear regression model was also quite accurate. On most cases, it did predict the better player successfully, and in the cases where the comparison is quite close, the results conveyed that as well (both prediction outputs were close to 0.5).
  - Model performance on decade-position combinations
    - The linear regression model did well when comparing players across different decades, for both the same position and different position cases. It still did well, but slightly less well when comparing players across different decades, for the different position and the same decade case.
    - The linear regression model struggled when comparing players of the same position and the same decade.
    - This is likely because the differences in points, shooting percentage, and other stats are not as drastic between players in the same era as players in different eras. For instance, in today's NBA, the game revolves a lot more around the three point shot, which allows players to score more points, but at a lower percentage. Additionally, fouls/free throws are easier to get, which provides players with another way to score points. However, in the NBA over one/two decades ago, the game was a lot more physical, so players would score less points, but work closer to the basket and have a higher shooting percentage. Additionally, defensive stats were better during this era. Essentially, players with very similar stats are difficult to compare, and predictions quality is more likely to be low. Differences in player stats make them easier to compare and allow the model to make better predictions.

Analysis

- Random Forest
  - Random Forest Model would initially pick the correct response in terms of the better player, although the result would often predict the players to be a lot closer in terms of how much better one was than the other
  - When the order of the players was switched, the model did not remain symmetric and actually predicted that the worse player was better
  - This discrepancy in symmetry could be due to the actual structure of the random forest algorithm and its splitting algorithm, as the order that the players are passed in could influence how the stats themselves are split within the tree at each iteration and change the threshold values that determine predictions
  - The random forest seems to work much better for players compared from different decades and the same position and the different decades different positions as the players who were statistically better were chosen to be better by the model and the model remained accurate even when the player order was switched
  - The reason the model might perform for players from different decades is because there are large differences in stats across many categories since these players played in different eras with major differences in terms of competitions and rules, allowing the model to easily pick up on these discrepancies irregardless of the order the players were passed in

Conclusion

- Our crowdsourcing platform, combined with our two ML models allow us to predict the better NBA player from pairs
  - We provide an interface where fans can contribute information regarding popular and trending NBA players
- The platform allows us to collect results from users, which were used to train machine learning models.
  - These models are utilized to predict the best NBA players in various categories:
    - Different positions, different decades, same positions, same decades
- Both our Random Forest and Linear Regression models proved to accurately predict:
  - Symmetric property held true for majority of pairs
  - Most predictions in pair groups were accurate

Thanks for Listening!