SHOW CODE

Trevor Ward
tward21@jcu.edu
https://github.com/tward21/NBA_HOF_Predictor
https://www.linkedin.com/in/trevor-ward-35148a17b/
October 26, 2020

Predicting NBA Hall of Famers Using Machine Learning in Python

With the 2020 NBA season just finishing up, and with LeBron James having secured his fourth NBA finals with his third team, the Los Angeles Lakers, the topic of who is the greatest basketball of all time is often being debated. Being from Cleveland and a huge LeBron bandwagoner, I thought it would be fun to create an assortment of machine learning models in an attempt to predict which players, amongst all who have ever played in NBA history, should definitively make the NBA Hall of Fame, and perhaps at the same time prove which player is truly the Greatest of All Time.

About the Data

This dataset was scraped from basketball-reference.com using a web scraper, done both using the Python tool BeautifulSoup, as well as by utilizing the Python library Pandas. Player stats and accolades, including things like total points, rebounds, and assists, as well as accolades like number of championships won, total MVP awards, and All NBA Team selections, were found and scraped into pandas DataFrames, allowing for machine learning models to be trained and deployed. Information was scraped for every season from 1950 (the first official season of the NBA) until 2020.

SHOW CODE

SHOW CODE

SHOW CODE

SHOW CODE

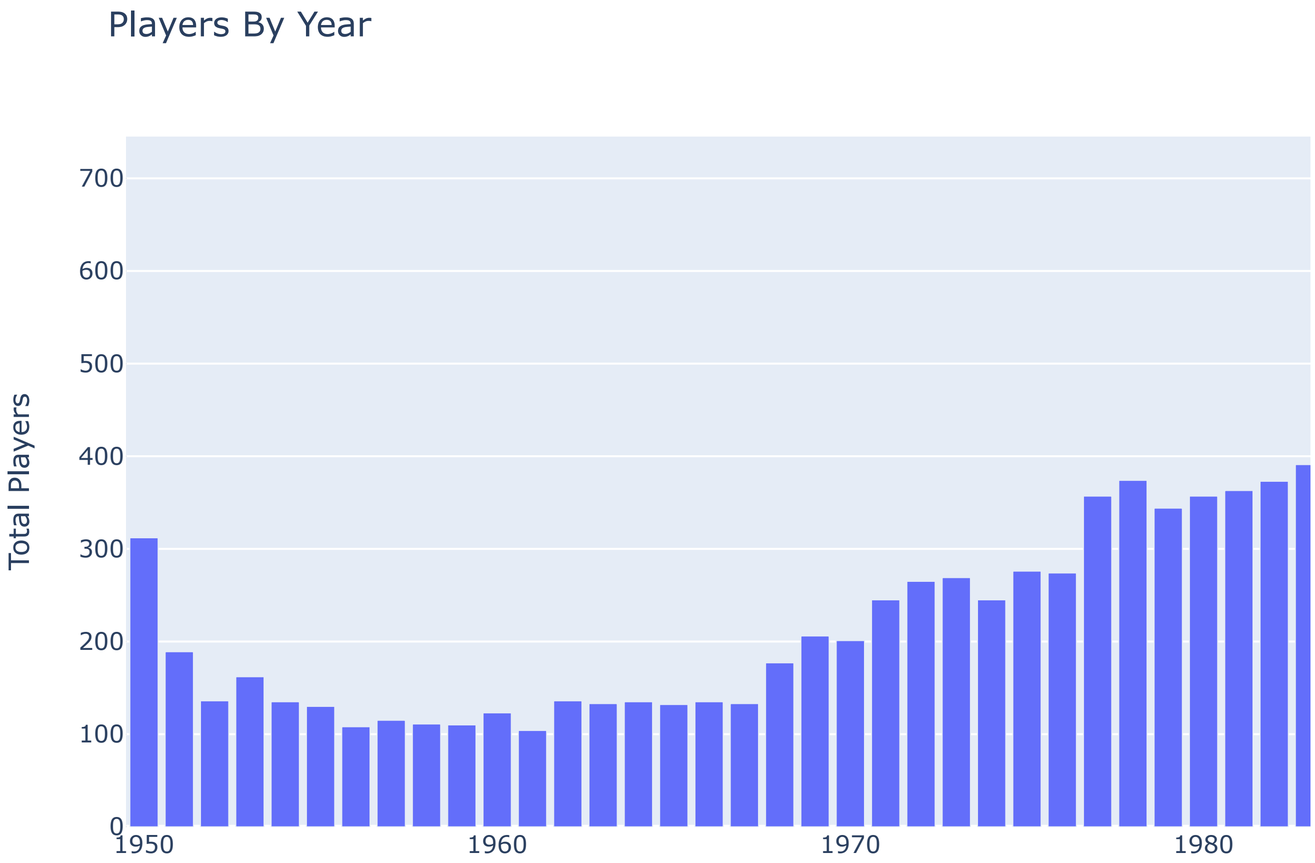
SHOW CODE

SHOW CODE

SHOW CODE

Basic Visualization

SHOW CODE

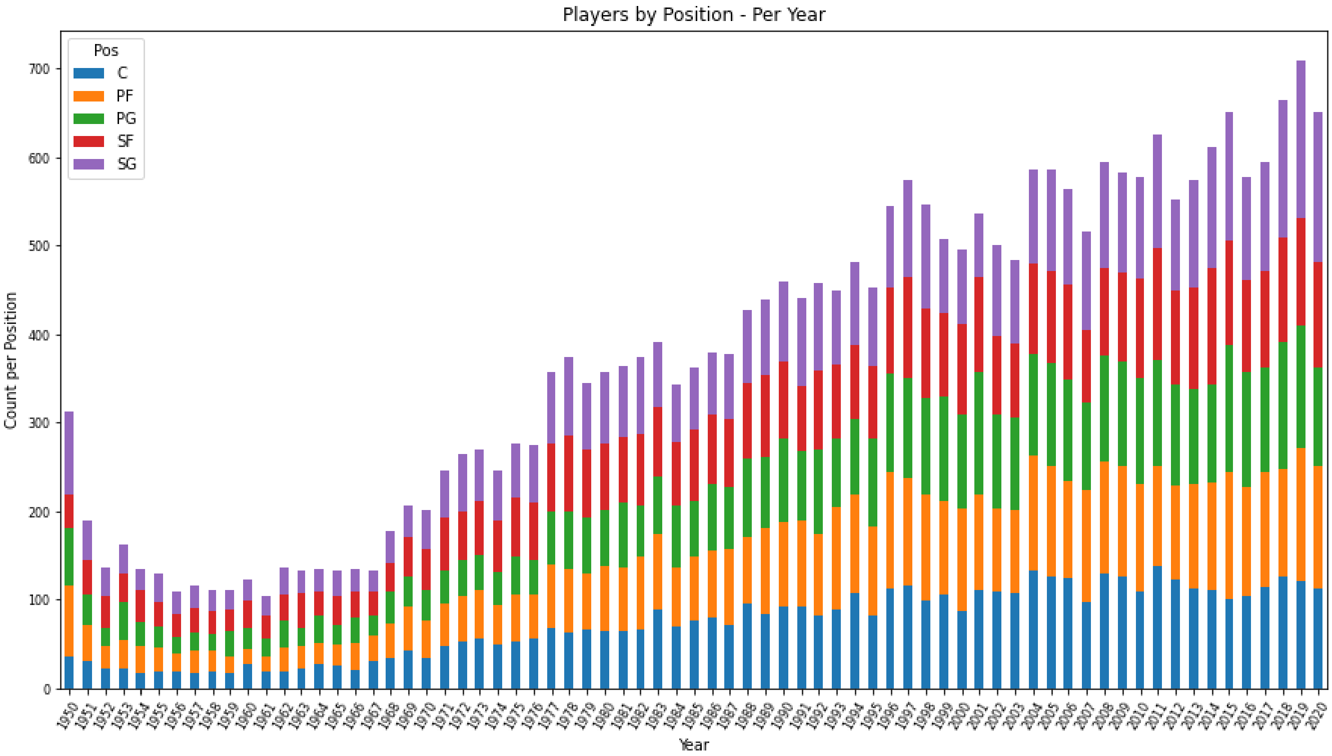


SHOW CODE

SHOW CODE

SHOW CODE

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5628117160>
```

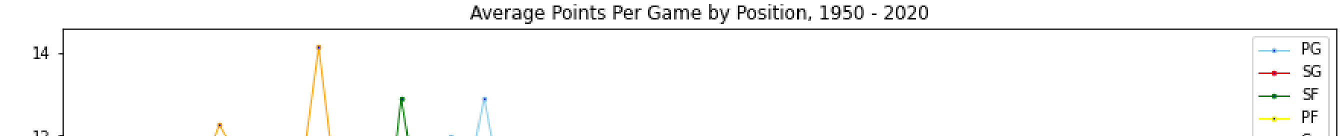


SHOW CODE

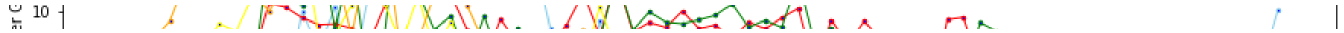
SHOW CODE

SHOW CODE

```
Text(0, 0.5, 'Points Per Game')
```

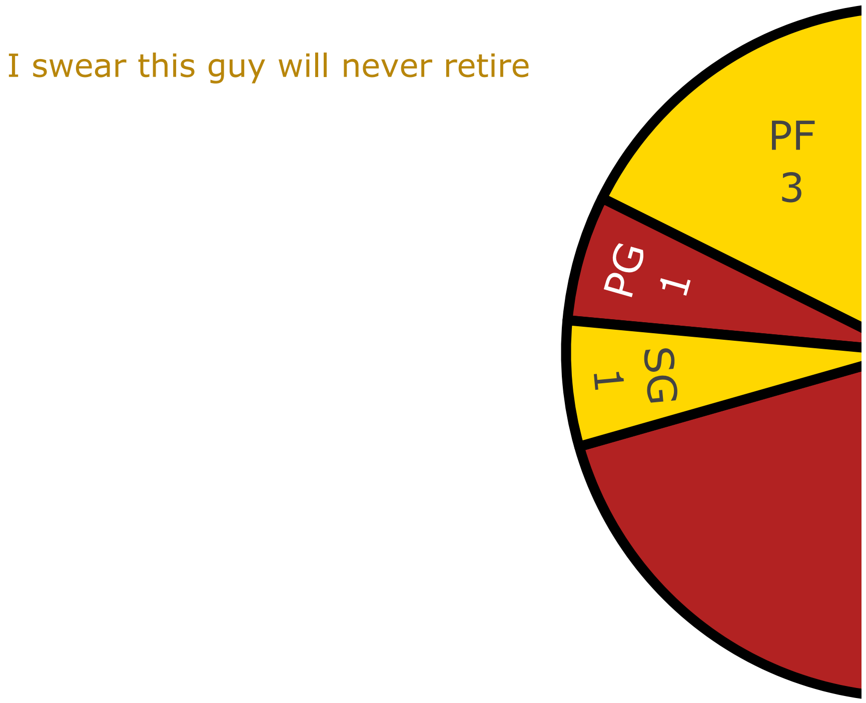


SHOW CODE



SHOW CODE

LeBron James's Total Years Played by Listed Position



Scraping, Cleaning Data

SHOW CODE

SHOW CODE

SHOW CODE

SHOW CODE

SHOW CODE

SHOW CODE

SHOW CODE

SHOW CODE

SHOW CODE

SHOW CODE

SHOW CODE

SHOW CODE

SHOW CODE

SHOW CODE

SHOW CODE

SHOW CODE

SHOW CODE

SHOW CODE

Modeling

Very low percentage of players are in the hall of fame right now (about 3 percent of the ~4000 total players that have played all time). The goal is to produce a model that predicts near perfect results, but not 100% accurate, as there are players in the NBA playing today or that recently retired that will most likely make the hall of fame. Creating a model that does not overfit on the data, but does have an accuracy greater than 96% (the accuracy that could be achieved simply by labeling every player as not in the Hall of Fame) requires model tuning by trial and error.

SHOW CODE

```
0    0.967482
1    0.032518
Name: hallOfFame, dtype: float64
```

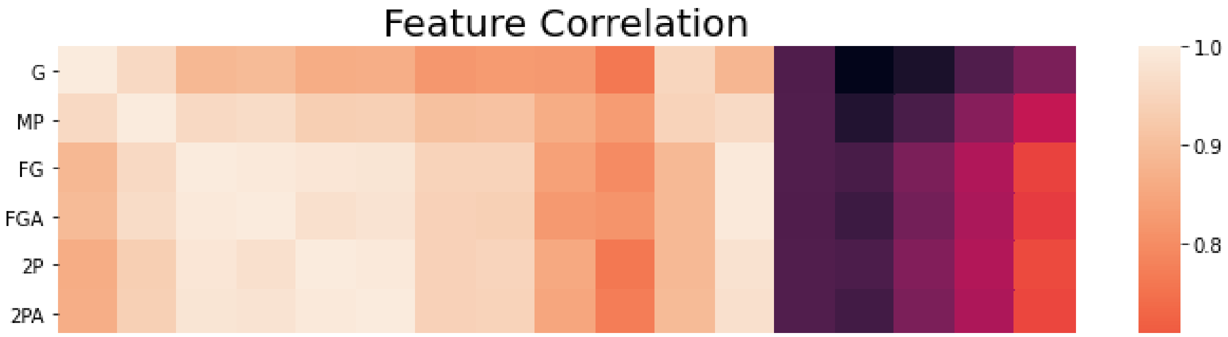
SHOW CODE

Feature Correlation

Most of players' in game stats are highly correlated, most so with stats like points and 2pt field goals attempted, and least so with stats like assists and rebounds (most players are likely not to have high numbers of both). Awards info is less correlated, except for all star appearences, which makes sense as it is the award most influenced by having good in game stats, that the most players are allowed to have in a year (only 1 MVP per year, only 5 all NBA Team selections per team, etc).

SHOW CODE

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5627520a20>
```



Logistic Regression

SHOW CODE

```
Train accuracy score: 0.9868613138686131
Test accuracy score: 0.9785185185185186
[[2632   13]
 [  23   72]]
```

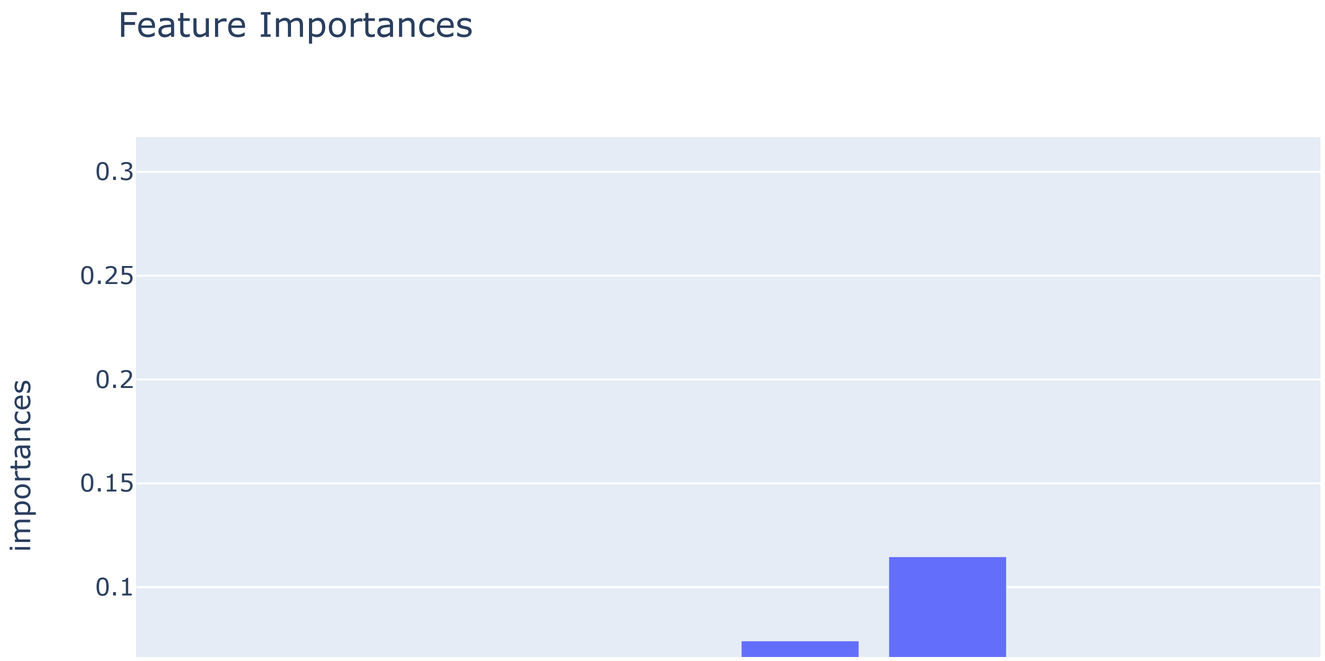
Random Forest

SHOW CODE

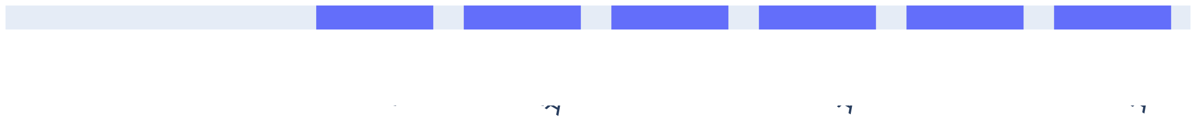
```
Train accuracy score: 0.9897810218978103
Test accuracy score: 0.9807407407407407
[[1303    9]
 [  17   21]]
```

All star game appearences appears to be the stat most indicative of whether a player will make the Hall of Fame or not. This makes sense, as All Star Game appearences are a good measure of for how many years a player has stood apart as a 'star' player. More than 20 players total can make the all star game each year, so it is a good way to keep track of and predict the top players in the game over time.

SHOW CODE



SHOW CODE



SHOW CODE

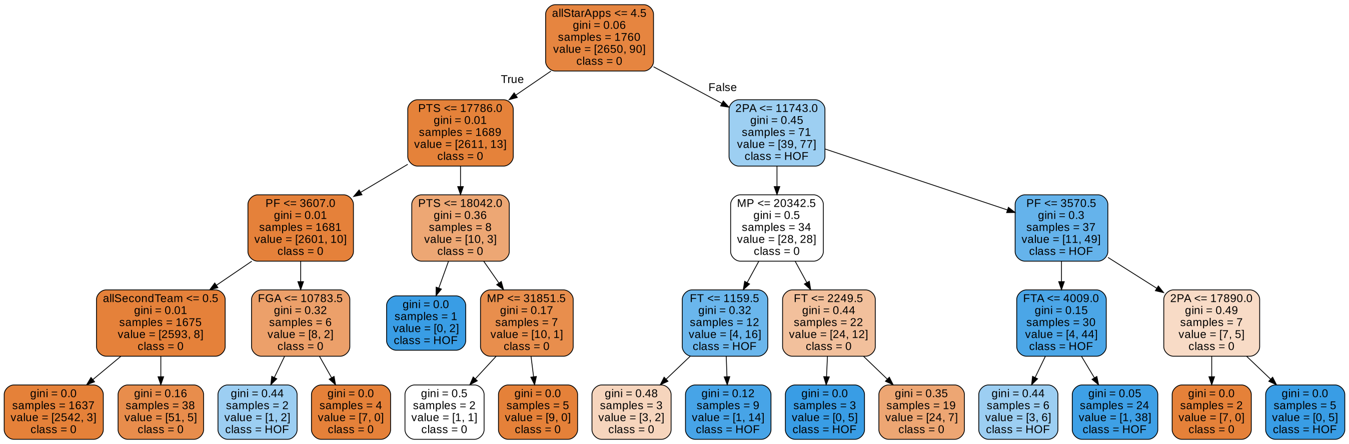
Notable False Positives classified by the Random Forest - the most accurate of my measured models. These players represent those who are not actually in the Hall of Fame (yet?), but that the model classifies as being in the Hall of Fame.

SHOW CODE

```
691          Chris Webber
1110        Dirk Nowitzki
1210          Dwyane Wade
1775         James Harden
2347         Kevin Durant
2431    LaMarcus Aldridge
2459          Larry Foust
2494         LeBron James
3061            Pau Gasol
Name: Player, dtype: object
171         Anthony Davis
580       Carmelo Anthony
681            Chris Paul
1206        Dwight Howard
3079          Paul Pierce
3426    Russell Westbrook
Name: Player, dtype: object
```

Sample of one random tree from the random forest

SHOW CODE



Nearest Neighbors Clasifier

SHOW CODE

```
Train accuracy score: 0.9923357664233576
Test accuracy score: 0.9785185185185186
[[2636    9]
 [  12   83]]
```

XGBoost Classifier

SHOW CODE

```
Train accuracy score: 0.9923357664233576
```

Support Vector Machine

```
 [  18   20]]
```

SHOW CODE

```
Train accuracy score: 0.9868613138686131
Test accuracy score: 0.9740740740740741
[[1299   13]
 [  22   16]]
```

Keras Neural Network

SHOW CODE

```
Epoch 1/60
305/305 [==============================] - 0s 831us/step - loss: 0.0739 - accuracy: 0.96
Epoch 2/60
305/305 [==============================] - 0s 857us/step - loss: 0.0346 - accuracy: 0.96
Epoch 3/60
305/305 [==============================] - 0s 857us/step - loss: 0.0243 - accuracy: 0.97
Epoch 4/60
305/305 [==============================] - 0s 816us/step - loss: 0.0198 - accuracy: 0.97
Epoch 5/60
305/305 [==============================] - 0s 800us/step - loss: 0.0178 - accuracy: 0.98
Epoch 6/60
305/305 [==============================] - 0s 878us/step - loss: 0.0165 - accuracy: 0.98
Epoch 7/60
305/305 [==============================] - 0s 821us/step - loss: 0.0156 - accuracy: 0.98
Epoch 8/60
305/305 [==============================] - 0s 818us/step - loss: 0.0149 - accuracy: 0.98
Epoch 9/60
305/305 [==============================] - 0s 813us/step - loss: 0.0144 - accuracy: 0.98
Epoch 10/60
305/305 [==============================] - 0s 798us/step - loss: 0.0140 - accuracy: 0.98
Epoch 11/60
305/305 [==============================] - 0s 879us/step - loss: 0.0133 - accuracy: 0.98
Epoch 12/60
305/305 [==============================] - 0s 814us/step - loss: 0.0131 - accuracy: 0.98
Epoch 13/60
305/305 [==============================] - 0s 814us/step - loss: 0.0130 - accuracy: 0.98
Epoch 14/60
305/305 [==============================] - 0s 860us/step - loss: 0.0129 - accuracy: 0.98
Epoch 15/60
305/305 [==============================] - 0s 870us/step - loss: 0.0126 - accuracy: 0.98
Epoch 16/60
305/305 [==============================] - 0s 809us/step - loss: 0.0123 - accuracy: 0.98
Epoch 17/60
305/305 [==============================] - 0s 806us/step - loss: 0.0122 - accuracy: 0.98
Epoch 18/60
305/305 [==============================] - 0s 843us/step - loss: 0.0119 - accuracy: 0.98
Epoch 19/60
305/305 [==============================] - 0s 861us/step - loss: 0.0119 - accuracy: 0.98
Epoch 20/60
305/305 [==============================] - 0s 867us/step - loss: 0.0118 - accuracy: 0.98
Epoch 21/60
305/305 [==============================] - 0s 945us/step - loss: 0.0116 - accuracy: 0.98
Epoch 22/60
305/305 [==============================] - 0s 950us/step - loss: 0.0116 - accuracy: 0.98
Epoch 23/60
305/305 [==============================] - 0s 916us/step - loss: 0.0113 - accuracy: 0.98
Epoch 24/60
305/305 [==============================] - 0s 823us/step - loss: 0.0107 - accuracy: 0.98
Epoch 25/60
305/305 [==============================] - 0s 809us/step - loss: 0.0113 - accuracy: 0.98
Epoch 26/60
305/305 [==============================] - 0s 824us/step - loss: 0.0110 - accuracy: 0.98
Epoch 27/60
305/305 [==============================] - 0s 861us/step - loss: 0.0111 - accuracy: 0.98
Epoch 28/60
305/305 [==============================] - 0s 873us/step - loss: 0.0110 - accuracy: 0.98
Epoch 29/60
```

```
305/305 [==============================] - 0s 820us/step - loss: 0.0107 - accuracy: 0.98
Epoch 30/60
305/305 [==============================] - 0s 879us/step - loss: 0.0109 - accuracy: 0.98
Epoch 31/60
305/305 [==============================] - 0s 861us/step - loss: 0.0109 - accuracy: 0.98
Epoch 32/60
305/305 [==============================] - 0s 859us/step - loss: 0.0105 - accuracy: 0.98
Epoch 33/60
305/305 [==============================] - 0s 919us/step - loss: 0.0107 - accuracy: 0.98
Epoch 34/60
305/305 [==============================] - 0s 860us/step - loss: 0.0106 - accuracy: 0.98
Epoch 35/60
305/305 [==============================] - 0s 806us/step - loss: 0.0104 - accuracy: 0.98
Epoch 36/60
305/305 [==============================] - 0s 806us/step - loss: 0.0104 - accuracy: 0.98
Epoch 37/60
305/305 [==============================] - 0s 829us/step - loss: 0.0105 - accuracy: 0.98
Epoch 38/60
305/305 [==============================] - 0s 875us/step - loss: 0.0103 - accuracy: 0.98
Epoch 39/60
305/305 [==============================] - 0s 865us/step - loss: 0.0101 - accuracy: 0.98
Epoch 40/60
305/305 [==============================] - 0s 850us/step - loss: 0.0104 - accuracy: 0.98
Epoch 41/60
305/305 [==============================] - 0s 819us/step - loss: 0.0103 - accuracy: 0.98
Epoch 42/60
```

SHOW CODE

```
[[1300   12]
 [  15   23]]
```

```
Epoch 45/60
```

Model Test Scores

```
Epoch 47/60
```

Since it is difficult to tell what models will perform best on a given dataset, an assortment of models were tested to see which format returned the best results. Models were trained using trained using Sklearn's library train-test split to break up the stats dataframe into a train and validation dataset (split into thirds, 67% and 33% respectively). The most accurate of these models was the normal Random Forest model, followed closely by the Keras neural network. Feature importances were produced by the Logistic Regression model. All models were trained on the X_training dataset, with validation data containing the actual values of who is in the Hall of Fame coming from the targetDf. Models were then all tested on the 33% validation data, all of which the model had not yet seen. Accuracy scores in the graph below represent accuracies of each model on the validation data.

```
Epoch 55/60
```

SHOW CODE

	ModelNames	ModelPercentAccurate
1	RandomForest	0.9807
5	NeuralNetwork	0.9800
0	LogReg	0.9785
3	XGBoost	0.9778
4	SVM	0.9741

SHOW CODE

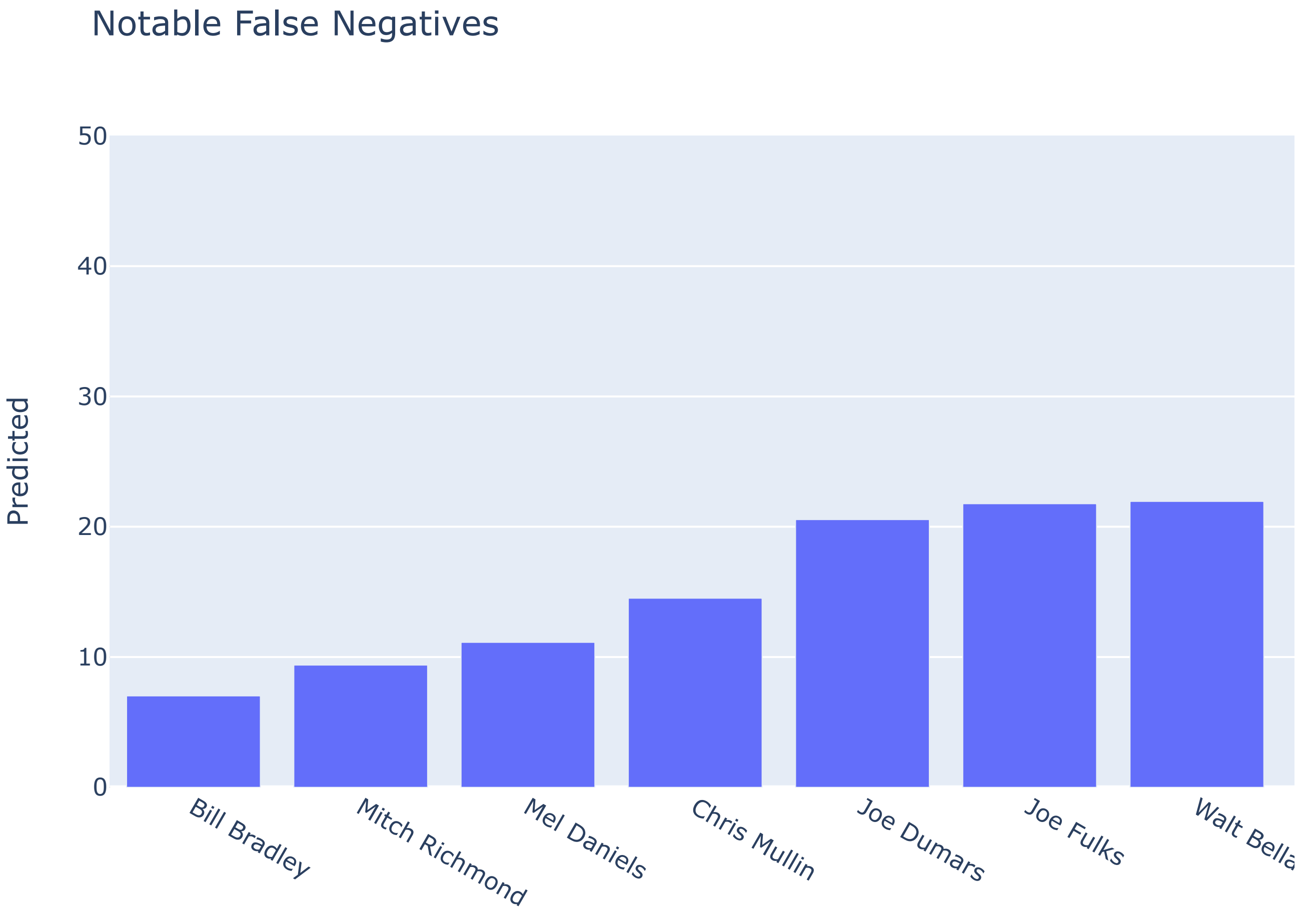
SHOW CODE

SHOW CODE

Predictions for current players can also be leveraged by using the Keras Neural Network. Since the network produces 2 percentages, how confident it is a certain player is in the Hall of Fame and how confident they are not, predictions on how likely each player is to make the Hall of Fame can be explored. Below are graphs depicting the players with the highest probability of making the Hall of Fame, as well as some notable False Positives, players the model predicted should be in the Hall of Fame that are not, and False Negatives, players that the model believes should not be in the Hall of Fame that are.

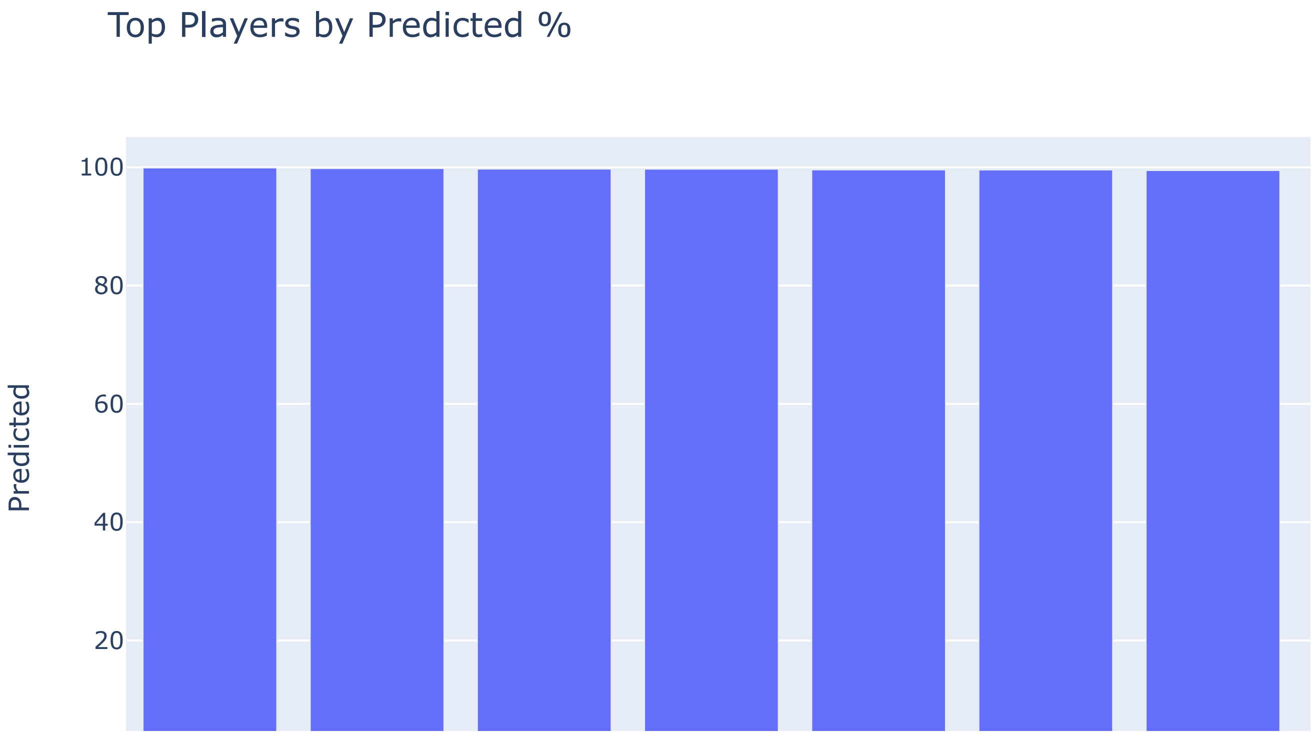
SHOW CODE

SHOW CODE



SHOW CODE

SHOW CODE



Conclusion

It seems machine learning can in fact be successfully applied to NBA data to give an idea of what players should and should not be in the NBA's Hall of Fame. This has been my favorite machine learning project to work on to date, as I'm a huge NBA fan and I found it to be very insightful exploring what type of accolades, accomplishments, and overall careers often lead to being immortalized in the Naismith Hall of Fame. This project has lead me to ask the question, should machine learning algorithms be applied to NBA career data to determine if someone should or should not be enshrined? Is it a good thing that the honor is voted on by humans instead?

Thanks for reading!

Some fun predictions for my fellow Cleveland fans below

```
find_HOF('LeBron James')
```

	Player	PredConfidence
2494	LeBron James	99.778

```
find_HOF('Kevin Love')
```

	Player	PredConfidence
2361	Kevin Love	31.175

```
find_HOF('Stephen Curry')
```

	Player	PredConfidence
3598	Stephen Curry	87.645

```
find_HOF('Draymond Green')
```

	Player	PredConfidence
1181	Draymond Green	34.607