Utilising Data Engineering Techniques to Obtain Basketball Data and Design Machine Learning Models

MSIN0166 - Individual Coursework

Data Engineering

GitHub Repository: https://github.com/sidnand24/data-eng-individual

UCL

April, 2022

Contents

1 Introduction

Data engineering is the practice of constructing infrastructure to facilitate the collection, storage and analysis of data at scale. Subsequently, the data can then be manipulated by end-users, such as data scientists, who may seek to perform further analysis. This project employs the ETL (Extract, Transform, Load) process to source data from numerous locations, ensure it is consistent and finally load it within a database. Following this, the aim shifts to using the data for a machine learning problem. The data selected for this project focuses on basketball with information regarding player statistics and general team details.

1.1 Project Mission

Firstly, the data is sourced from four separate websites, each contributing disparate information regarding the sport. Afterwards, the data is transformed and hosted within a constructed database to be used thereafter. Two extraction methods are employed, web scraping and application programming interfaces (APIs). Then, the data is finally used to help evaluate the efficacy of various machine learning models in their predictive capacity in determining the salary of professional basketball players. The project architecture is displayed in Figure 1. This project focuses on the 2021/2022 current season as The National Basketball Association (NBA) implements a salary cap that changes each year dependent on the previous season's revenue. The aim of such a policy is to ensure fairness, although it is a soft cap. Teams can sign players that exceed the limit by following a number of complicated exceptions (Augustin, 2009). Nevertheless, including previous year data may influence the performance of the models and is therefore omitted.

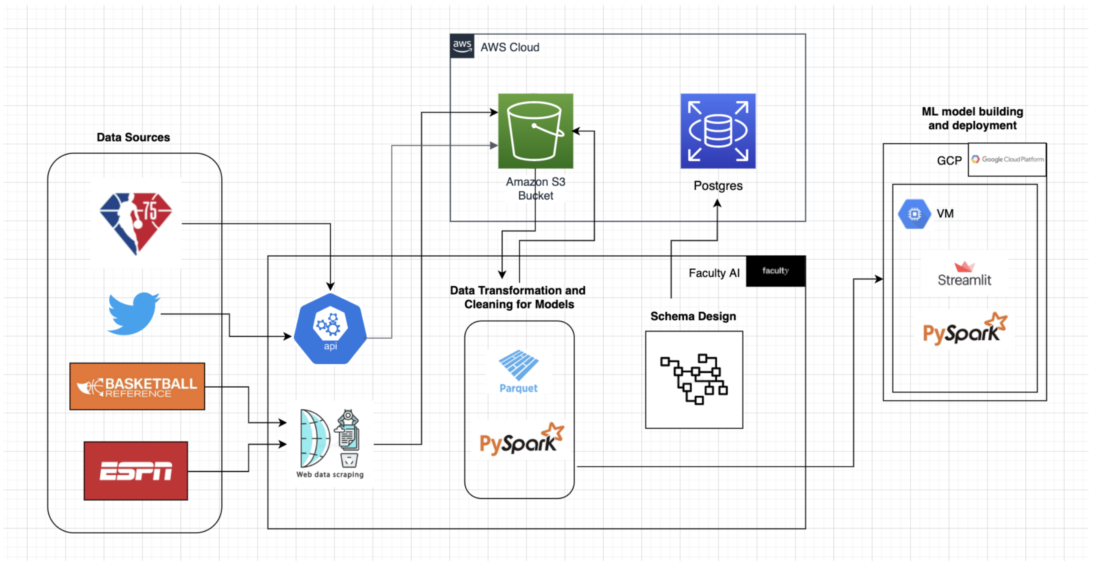
1.2 Business Value

The NBA is one of the most watched sports globally, attracting millions of viewers each game. Similar to other sports, a question that many ponder is how much the atheletes are paid. With various players earning tens of millions each year, the investment each team makes in their player selection is crucial. With a yearly limit to the total money each team can spend on their squad's salaries, determining which players to invest in is critical. With a good performing model, team managers can observe whether the money they are paying is merited. Furthermore, players can identify where to improve to maximise their earning potential and bargaining power. As a multitude of factors can influence the salary, such as previous injuries or contract length, the model is unlikely to predict the value with 100% accuracy. However it still provides a valuable estimate for players and managers alike.

2 Source Control

Version control was incorporated within the project through the use of Git. The Python scripts were then stored on GitHub, a Git repository hosting service. Maintaining versions of previous files ensured any changes were tracked and were easily observable. This allowed for better organisation when determining the current stage of the project. By utilising a Distributed Version Control System, future work by potential new collaborators can be more straightforward as individuals can have access to the full project history.

Figure 1. Project Architecture



3 Data Storage

Throughout the project, data storage was managed using Amazon Web Services (AWS) on account of the free tier offered. As the lead cloud computing platform, AWS is widely used within industry. Therefore, exposure to the service was desired. For security, AWS provides credentials in the form of an access key and secret access key for users to connect to the service. As best practice, these credentials were omitted from the code that was committed to GitHub for security reasons. A separate config file was created to store these sensitive values. After the collection and transformation of the data for input to the database, files were stored within an S3 bucket (Simple Storage Service) in Parquet format.

Apache Parquet format is a columnar storage format for Hadoop. Whilst using frameworks like Apache Spark, appropriate storage formats can be valuable to improve efficiency as the volume of data grows. For this project, Parquet was selected, rather than a contrasting row-oriented format due to the reduced file size and the benefit of Spark's capability of reading many Parquet files at once. Furthermore, with columnar based storage reading data can be easier if only certain columns are required for analysis. As Plase et al. (2017) argues Parquet files excel in data compression and queries from Parquet files are faster when compared with row-oriented formats such as Avro.

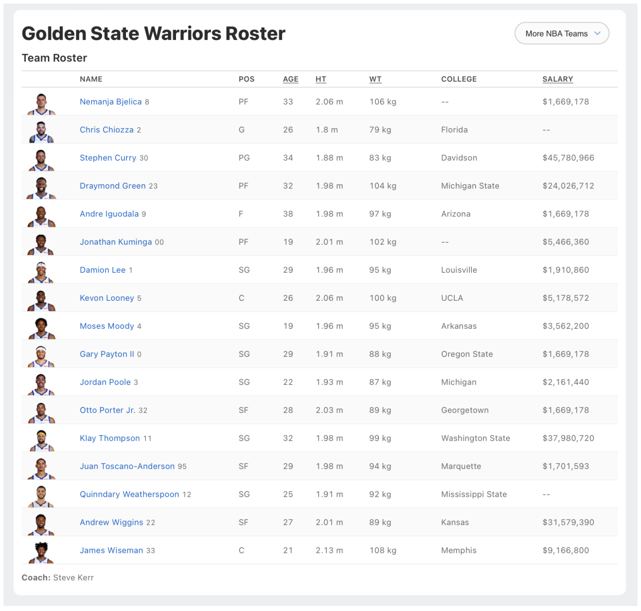
After further transformation following collection, in accordance with the schema designed, the data was then uploaded as a PostgreSQL database on RDS (Relational Database Service) within AWS. RDS was selected due to the reduced operational costs from hosting the database within AWS. The database was adjusted to enable access from any IP address. Although, this introduces security concerns, as the data involved is publicly available, the decision was justified for straightforward access.

4 Data Sourcing

4.1 ESPN

With the aim of predicting NBA player salary, information regarding the earnings for each player in the current season was requisite. To this end, the first source of data was ESPN, an American international cable sports channel. The website provides news on various sports, one of them being basketball. Initial web scraping, using the Beautiful Soup library, ensured the URL for each team's roster was collected. This offered the opportunity to get data for each player. Following this, multiple dataframes were created from the scraped table on each page as displayed in Figure 2. These dataframes were then converted into a PySpark DataFrame which was uploaded to S3 in Parquet format. Information surrounding the salary, as well as general information about the players, was obtained. For best practice, headers were inputted within the web scraping code to provide meta data about the request. Further, it helps identify oneself to the server. In addition, within the for-loop to extract the tables, each request was spaced out randomly to avoid overloading the server, and reduce the chance of being restricted from the website.

Figure 2. ESPN Roster Data



Golden State Warriors Roster

Team Roster

NAME	POS	AGE	HT	WT	COLLEGE	SALARY
Nemanja Bjelica 8	PF	33	2.06 m	106 kg	--	$1,669,178
Chris Chiozza 2	G	26	1.8 m	79 kg	Florida	--
Stephen Curry 30	PG	34	1.88 m	83 kg	Davidson	$45,780,966
Draymond Green 23	PF	32	1.98 m	104 kg	Michigan State	$24,026,712
Andre Iguodala 9	F	38	1.98 m	97 kg	Arizona	$1,669,178
Jonathan Kuminga 00	PF	19	2.01 m	102 kg	--	$5,466,360
Damion Lee 1	SG	29	1.96 m	95 kg	Louisville	$1,910,860
Kevon Looney 5	C	26	2.06 m	100 kg	UCLA	$5,178,572
Moses Moody 4	SG	19	1.96 m	95 kg	Arkansas	$3,562,200
Gary Payton II 0	SG	29	1.91 m	88 kg	Oregon State	$1,669,178
Jordan Poole 3	SG	22	1.93 m	87 kg	Michigan	$2,161,440
Otto Porter Jr. 32	SF	28	2.03 m	89 kg	Georgetown	$1,669,178
Klay Thompson 11	SG	32	1.98 m	99 kg	Washington State	$37,980,720
Juan Toscano-Anderson 95	SF	29	1.98 m	94 kg	Marquette	$1,701,593
Quinndary Weatherspoon 12	SG	25	1.91 m	92 kg	Mississippi State	--
Andrew Wiggins 22	SF	27	2.01 m	89 kg	Kansas	$31,579,390
James Wiseman 33	C	21	2.13 m	108 kg	Memphis	$9,166,800

Coach: Steve Kerr

4.2 Twitter

The social media, Twitter, generates a considerable volume of data daily, and it offers a platform for users to express raw public opinion that can be valuable for analysis. Such tweets were deemed beneficial for this study as they could provide a proxy popularity metric that could be used per player. The salary of each player is likely to be influenced by how commercial and marketable they can be. Therefore, it was assumed that a greater number of tweets in the last week would signify a more popular star. Nevertheless, this approach does suffer from some limitations. Tweet counts may spike due to bad publicity or bad performance. Future work may then seek to investigate collecting data from other social media networks such as Instagram (e.g.

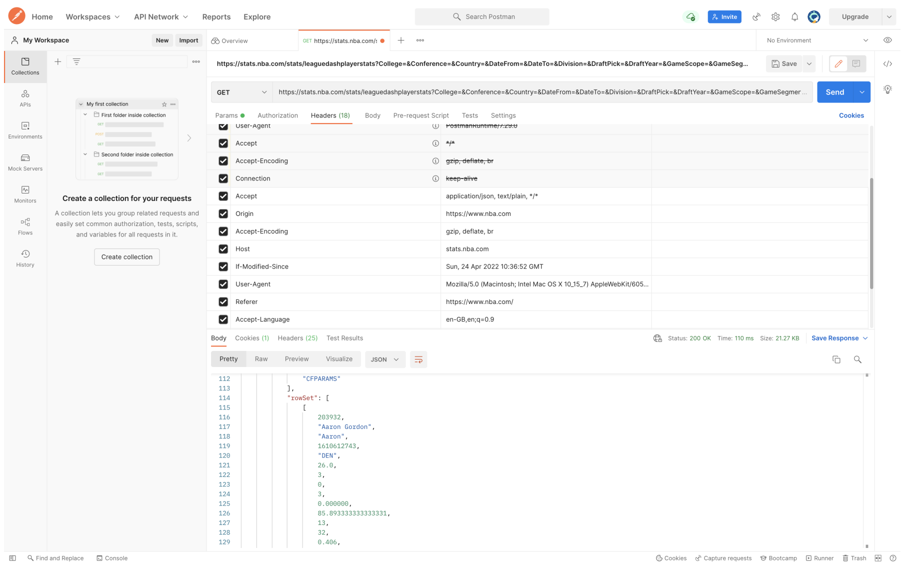
number of followers). Although, difficulties may still be observable in the form of obscure account handles unrepresentative of player names.

The Twitter API was utilised to extract such data. Firstly, a Twitter developer account was created. This is required to access the API as it provides consumer keys and authentication tokens. One in particular that is used is the bearer token. The Python module Tweepy grants access to the Twitter API with the bearer token, and by providing specific queries, data can be collected. Applying the data from ESPN, a random sample of 100 players was selected to get data for. Trying to obtain data for every player resulted in a 429 error as too many requests were given to the server. Therefore, data augmentation was implemented to fabricate values for the other players not selected in the queries. The query within Tweepy was also restricted to remove retweets and display the number of tweets per day over the last week. The previous week's tweets were desired as there is no limit to this functionality on the free tier of the developer account. Furthermore, tweet numbers can vary due to numerous factors so by focusing on each week separately, further research on trends of popularity across the season can also be facilitated with the data collected. The data was uploaded to S3 in a similar method to that collected from ESPN.

4.3 Basketball Reference

To obtain player statistics from the current season, Basketball Reference, a website that provides comprehensive information regarding basketball, was selected. The statistics for each player is hypothesised to be a crucial metric in determining salaries as they inadvertently highlight performance levels and quality. Total rather than per game stats were collected for the season as the per game statistics could be calculated post collection. At first, the aim was to obtain data from the official NBA website.

Figure 3. Postman API Request



GET https://stats.nba.com/stats/leaguedashplayerstats?College=&Conference=&Country=&DateFrom=&DateTo=&Division=&DraftPick=&DraftYear=&GameScope=&GameSegmer ...

Headers (18)

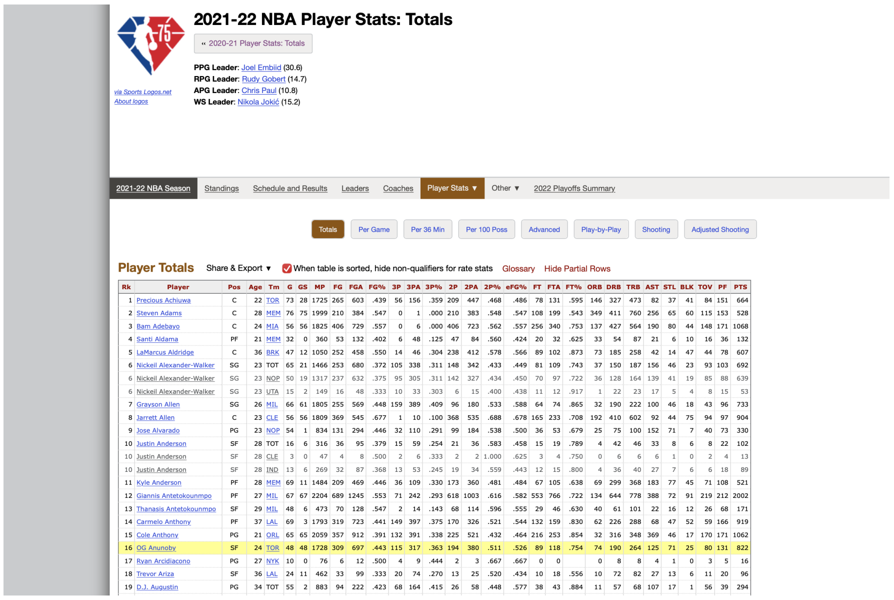
User-Agent	~~PostmanRuntime/7.29.0~~
Accept	~~*/*~~
Accept-Encoding	~~gzip, deflate, br~~
Connection	~~keep-alive~~
Accept	application/json, text/plain, */*
Origin	https://www.nba.com
Accept-Encoding	gzip, deflate, br
Host	stats.nba.com
If-Modified-Since	Sun, 24 Apr 2022 10:36:52 GMT
User-Agent	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/605...
Referer	https://www.nba.com/
Accept-Language	en-GB,en;q=0.9

Status: 200 OK Time: 110 ms Size: 21.27 KB

```
                "CFPARAMS"
            ],
            "rowSet": [
                [
                    203932,
                    "Aaron Gordon",
                    "Aaron",
                    1610612743,
                    "DEN",
                    26.0,
                    3,
                    0,
                    3,
                    0.000000,
                    85.893333333333331,
                    13,
                    32,
                    0.406,
```

It was determined that the stats data on the NBA website was not stored on the HTML source page, as the website was rendered client-side rather than server-side. Essentially, the website uses an API to fetch the data each time it is loaded to create the page client-side. Using the web browsers inspect tools, the API endpoint was found and using Postman, a platform to build and use APIs, the data was collected as shown in Figure 3. Nevertheless, when trying to incorporate the provided Python code from Postman, it did not run, perhaps due to inaccurate headers. Therefore Basketball Reference was chosen as a second choice, and the data was validated manually to be accurate.

Figure 4. Basketball Reference Data



2021-22 NBA Player Stats: Totals

PPG Leader: Joel Embiid (30.6)
RPG Leader: Rudy Gobert (14.7)
APG Leader: Chris Paul (10.8)
WS Leader: Nikola Jokić (15.2)

Player Totals

✓ When table is sorted, hide non-qualifiers for rate stats

Rk	Player	Pos	Age	Tm	G	GS	MP	FG	FGA	FG%	3P	3PA	3P%	2P	2PA	2P%	eFG%	FT	FTA	FT%	ORB	DRB	TRB	AST	STL	BLK	TOV	PF	PTS
1	Precious Achiuwa	C	22	TOR	73	28	1725	265	603	.439	56	156	.359	209	447	.468	.486	78	131	.595	146	327	473	82	37	41	84	151	664
2	Steven Adams	C	28	MEM	76	75	1999	210	384	.547	0	1	.000	210	383	.548	.547	108	199	.543	349	411	760	256	65	60	115	153	528
3	Bam Adebayo	C	24	MIA	56	56	1825	406	729	.557	0	6	.000	406	723	.562	.557	256	340	.753	137	427	564	190	80	44	148	171	1068
4	Santi Aldama	PF	21	MEM	32	0	360	53	132	.402	6	48	.125	47	84	.560	.424	20	32	.625	33	54	87	21	6	10	16	36	132
5	LaMarcus Aldridge	C	36	BRK	47	12	1050	252	458	.550	14	46	.304	238	412	.578	.566	89	102	.873	73	185	258	42	14	47	44	78	607
6	Nickeil Alexander-Walker	SG	23	TOT	65	21	1466	253	680	.372	105	338	.311	148	342	.433	.449	81	109	.743	37	150	187	156	46	23	93	103	692
6	Nickeil Alexander-Walker	SG	23	NOP	50	19	1317	237	632	.375	95	305	.311	142	327	.434	.450	70	97	.722	36	128	164	139	41	19	85	88	639
6	Nickeil Alexander-Walker	SG	23	UTA	15	2	149	16	48	.333	10	33	.303	6	15	.400	.438	11	12	.917	1	22	23	17	5	4	8	15	53
7	Grayson Allen	SG	26	MIL	66	61	1805	255	569	.448	159	389	.409	96	180	.533	.588	64	74	.865	32	190	222	100	46	18	43	96	733
8	Jarrett Allen	C	23	CLE	56	56	1809	369	545	.677	1	10	.100	368	535	.688	.678	165	233	.708	192	410	602	92	44	75	94	97	904
9	Jose Alvarado	PG	23	NOP	54	1	834	131	294	.446	32	110	.291	99	184	.538	.500	36	53	.679	25	75	100	152	71	7	40	73	330
10	Justin Anderson	SF	28	TOT	16	6	316	36	95	.379	15	59	.254	21	36	.583	.458	15	19	.789	4	42	46	33	8	6	8	22	102
10	Justin Anderson	SF	28	CLE	3	0	47	4	8	.500	2	6	.333	2	2	1.000	.625	3	4	.750	0	6	6	6	1	0	2	4	13
10	Justin Anderson	SF	28	IND	13	6	269	32	87	.368	13	53	.245	19	34	.559	.443	12	15	.800	4	36	40	27	7	6	6	18	89
11	Kyle Anderson	PF	28	MEM	69	11	1484	209	469	.446	36	109	.330	173	360	.481	.484	67	105	.638	69	299	368	183	77	45	71	108	521
12	Giannis Antetokounmpo	PF	27	MIL	67	67	2204	689	1245	.553	71	242	.293	618	1003	.616	.582	553	766	.722	134	644	778	388	72	91	219	212	2002
13	Thanasis Antetokounmpo	SF	29	MIL	48	6	473	70	128	.547	2	14	.143	68	114	.596	.555	29	46	.630	40	61	101	22	16	12	26	68	171
14	Carmelo Anthony	PF	37	LAL	69	3	1793	319	723	.441	149	397	.375	170	326	.521	.544	132	159	.830	62	226	288	68	47	52	59	166	919
15	Cole Anthony	PG	21	ORL	65	65	2059	357	912	.391	132	391	.338	225	521	.432	.464	216	253	.854	32	316	348	369	46	17	170	171	1062
16	OG Anunoby	SF	24	TOR	48	48	1728	309	697	.443	115	317	.363	194	380	.511	.526	89	118	.754	74	190	264	125	71	25	80	131	822
17	Ryan Arcidiacono	PG	27	NYK	10	0	76	6	12	.500	4	9	.444	2	3	.667	.667	0	0		0	8	8	4	1	0	3	5	16
18	Trevor Ariza	SF	36	LAL	24	11	462	33	99	.333	20	74	.270	13	25	.520	.434	10	18	.556	10	72	82	27	13	6	11	20	96
19	D.J. Augustin	PG	34	TOT	55	2	883	94	222	.423	68	164	.415	26	58	.448	.577	38	43	.884	11	57	68	107	17	1	56	39	294

Again using the Beautiful Soup library, the statistics for each player was obtained. This was relatively simple as the data resided in one large table as shown in Figure 4. Therefore, using the HTML tags and classes, the table was extracted and converted to a PySpark DataFrame to be uploaded as Parquet format to S3.

4.4 NBA.com

Finally, for additional data surrounding the teams each player is contracted to, an open source API client (nba_client) for the official NBA website was utilised. The APIs on the NBA website are predominantly undocumented and subject to frequent changes. Therefore, use of this API enables users to access the endpoints easier. With one preset query, information such as the nickname and location of each team, was obtained. Again this data was uploaded into S3 in Parquet format after being converted to a PySpark DataFrame.

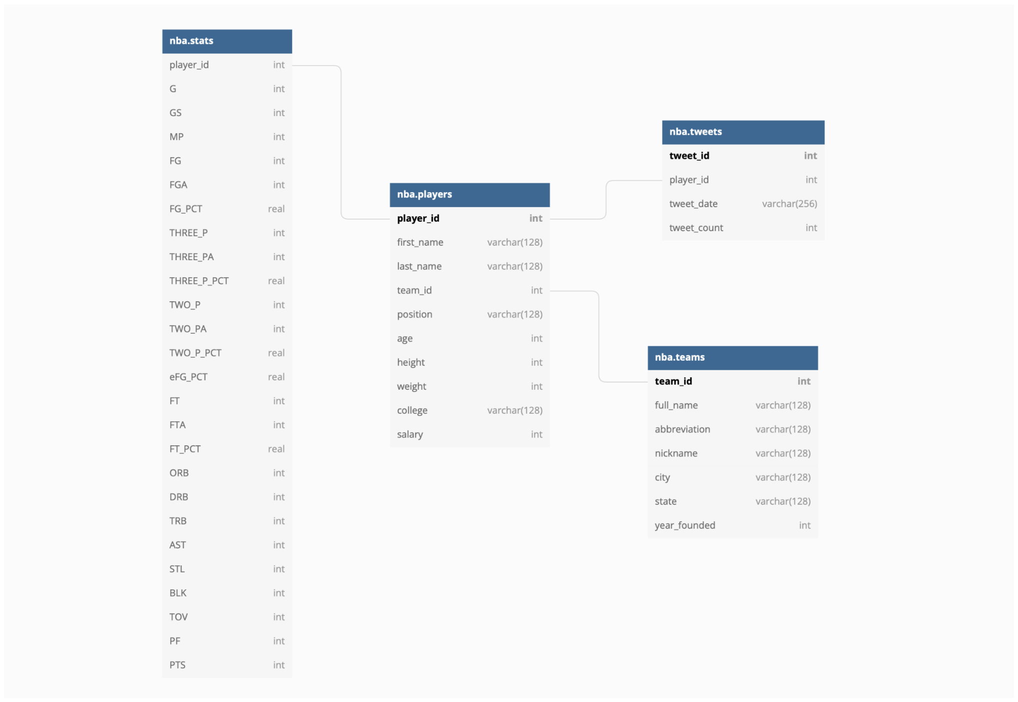
5 Database Design

When designing the database to host the collected data, the format was an essential decision. The approach opted for within this project was a relational database. Such databases organise the data into logically linked tables. Another type of database that was considered was a non-relational approach, sometimes referred to as NoSQL databases. Although, non-relational databases do not require a schema, a relational approach was selected due to the ease of forming queries on the data. As Leavitt (2010) proclaims, complex queries can be hard to formulate with NoSQL databases. Furthermore, the basketball data also comprised of easily discernible links, which facilitated the schema design.

5.1 Schema Creation

The database schema formulation was undertaken following the three normal forms of database normalisation. This helps to maintain data integrity, preventing the introduction of duplicate entries and redundant data. Introducing such data would lead to an inflated database, increasing storage costs and data inconsistencies which can be exasperated as data volume grows. The first normal form (1NF) is satisfied by confirming that the tables do not contain repeated values and that the data is atomic (can not be divided further). The second normal form (2NF) implies that the data follows the 1NF and does not have partial dependency if a composite primary key is present. Finally the third normal form is satisfied when the tables are in 2NF and there is no transitive partial dependency. Following all these, the schema designed is outlined in Figure 5.

Figure 5. Database Schema



The predominant table within the database is the 'players' table consisting of general information such as age, position and, importantly, the salary of each player. The name variable was split into first and last name to follow the 1NF. The primary key in this table is the unique player id. This key is referenced in the 'stats' table as a foreign key, where total stats across the season for each player are revealed. This is a one-to-one relationship. Also within the 'players' table is the team id for each player. This is a reference of the unique primary key in the teams table indicating general information for each team. Finally, the tweets table consists of a new primary key to ensure the primary key for each insert is unique. In addition, the player id is referenced as a foreign key to identify the star each tweet is attributed to.

6 Increasing Scalability

Throughout the project, Apache Spark, a distributed processing engine, is incorporated to increase the scalability of the work. As the volume of data increases, processing tasks in parallel can be very valuable. Furthermore, a benefit of using Spark is that processing is done in-memory, increasing speed. Also, through the use of Spark DataFrames, operations are only run when requested, which avoids bringing all the data into memory. To enable interaction with Python, PySpark is exploited, and it is also used for the model training and deployment when predicting NBA player salary.

7 Predicting NBA Player Salary

7.1 Machine Learning Preparation

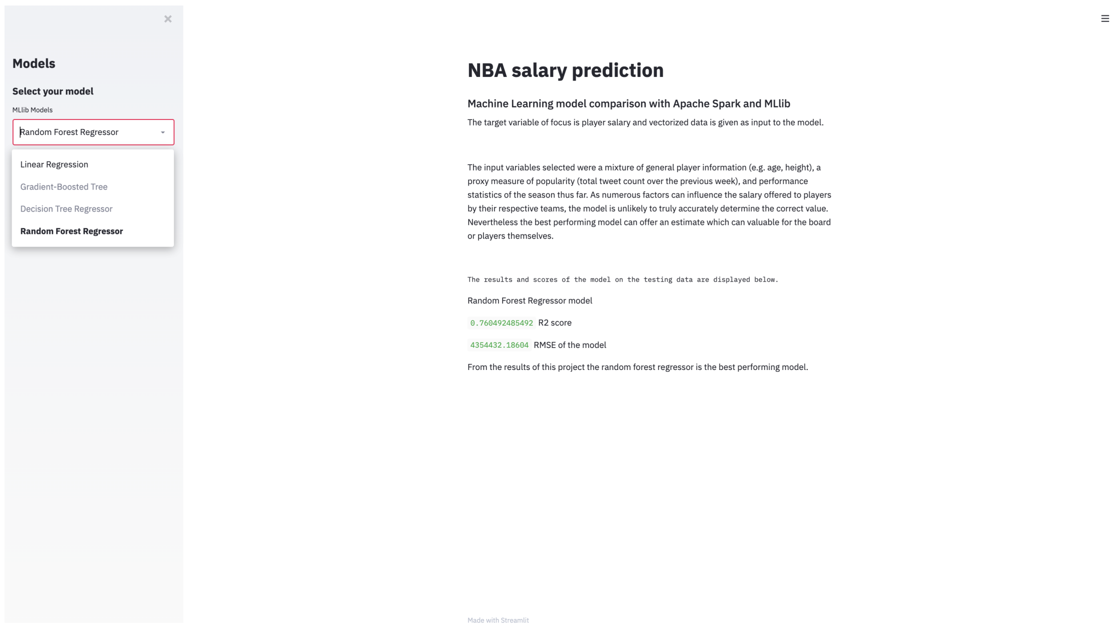
To undertake machine learning on the sourced data, the tables from the PostgreSQL database were imported and transformed using PySpark. The tables were joined to form a final dataset that could be used for the model training, keeping those variables that were desired, to help predict the salary. The full list of variables decided is listed in appendix A. Each of the stats was converted to per game metrics as it helps account for a varying number of games per player. Physical attributes of each player were also included as older players are likely to have fewer playing years remaining, so investment into them is hypothesised to be reduced. Furthermore, the selected statistics to be used for the models were based on the major metrics showcased by ESPN on their home NBA stats page. These were likely to be major characteristics in determining player performance and the salary that each player merits. Furthermore, the total tweet count over the previous week was calculated to supplement the variables with a proxy for popularity. As the position attribute was a string, it had to be converted to a numerical format to be used within the models. Using the *StringIndexer* and *OneHotEncoder* from PySpark's ml package the variable was converted. The *StringIndexer* creates a numerical representation for each position within the column. Then, the *OneHotEncoder* creates dummy variables for each category. Thereafter, the *VectorAssembler* was utilised to formulate a feature vector to be inputted into the models. The feature column and the label column were then saved in Parquet format with one partition and uploaded to GitHub.

7.2 Deployment

For this project, to run and deploy the models made with PySpark MLlib, Google Cloud Platform (GCP) was chosen using Streamlit. As Streamlit programs are run from the start, grid search to hyper-tune each model was avoided to reduce the time to render the entire page. Furthermore, this was the reason the Streamlit script included the pre-processed data to avoid computation cost. Four models from Spark's MLlib library were selected to evaluate; a linear regression, decision tree, gradient boosted tree and a random forest ensemble method. To compare the models, the R^2 and the root mean square error (RMSE) were preferred. Upon running each model, the random forest methodology seemed to offer the greatest accuracy.

The UI for the Streamlit program was created in a Python file consisting of a side bar where the user could select the desired model, and the scores displayed. This interface is highlighted in Figure 6. GCP was selected to run the Streamlit app due to their $300 of free credits offered on their trial. Furthermore, using a different service to AWS offered experience using another cloud computing service. A virtual machine with an Ubuntu operating system was created. Afterwards, the necessary packages to run the Streamlit app were downloaded onto the virtual machine. This included PySpark to run the machine learning models. The GitHub repository containing the pre-processed Parquet file was then cloned into the machine to run the Python script. To circumvent any potential costs, the Streamlit program was not left running. Nevertheless, future work may seek to achieve this for others to continue using the program and collaborators to improve upon the Streamlit app.

Figure 6. Streamlit Program



NBA salary prediction

Machine Learning model comparison with Apache Spark and MLlib

The target variable of focus is player salary and vectorized data is given as input to the model.

The input variables selected were a mixture of general player information (e.g. age, height), a proxy measure of popularity (total tweet count over the previous week), and performance statistics of the season thus far. As numerous factors can influence the salary offered to players by their respective teams, the model is unlikely to truly accurately determine the correct value. Nevertheless the best performing model can offer an estimate which can valuable for the board or players themselves.

```
The results and scores of the model on the testing data are displayed below.
```

Random Forest Regressor model

`0.760492485492` R2 score

`4354432.18604` RMSE of the model

From the results of this project the random forest regressor is the best performing model.

8 Conclusion

The database constructed for this project offers end-users a valuable resource to undertake further analysis regarding the NBA. Performance analysis across the season can be attempted which can also be focused on team-centric aspects rather than player metrics. In addition, the results of the machine learning model comparison can be utilised by teams to identify how much to invest in each player. General managers for each team can decide where to spend their designated salary cap each season.

8.1 Further Study

To expand on the current project, a wider variety of data can be collected. This may include statistics for each individual game or tweet text to potentially provide the opportunity for sentiment analysis. Moreover, using a workflow management tool such as Apache Airflow can help schedule and automate the extraction of data and the input into the PostgreSQL database. In particular the tweet counts and player statistics can be extracted each week to refresh and update the data. To improve reproducibility, Docker can be incorporated to containerise the application and code. This can help ensure the project runs on different environments.

References

Augustin, P. 2009. *Understanding the NBA Salary Cap: It Is Rocket Science*. URL https://bleacherreport.com/articles/224051-understanding-the-nba-salary-cap-it-is-rocket-science. 3

Leavitt, N. 2010. Will NoSQL databases live up to their promise? *Computer* 43:12–14. 7

Plase, D., L. Niedrite, and R. Taranovs. 2017. A comparison of HDFS compact data formats: Avro versus Parquet. *Mokslas–Lietuvos ateitis/Science–Future of Lithuania* 9:267–276. 4

A Independent Variables for the Models

Table 1. Variable Description

Variable	Description
age	Player age
height	Player height in inches
week_twets	The total number of tweets in previous week
MPG	Minutes per game
efg_pct	Effective field goal percentage (adjusts for different points per shot)
TRBPG	Total rebounds per game
APG	Assists per game
SPG	Steals per game
BPG	Blocks per game
TOPG	Turnovers per game
PPG	Points per game
PosVec	Encoded value for player position