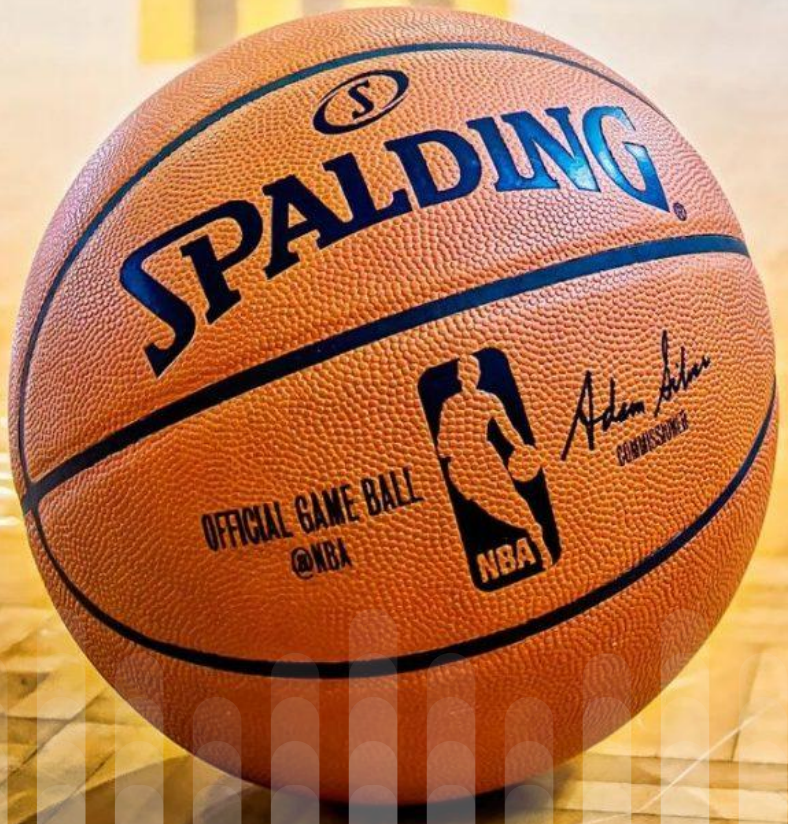
NBA Analysis



- Danny Abouchakra
- Danielle DiGioacchino
- Keshav Gupta
- Shantanu Vaidya

Project Topic and Reasons for Selection?

Using NBA data sets, our team will run supervised machine learning to answer whether or not the outcome of a game can be predicted with high accuracy.

Reasons:

- Large dataset available
- We share interest in the sport.
- It is a famous sport across North America.
- Lastly, If we predict game outcomes with high accuracy we could make some profits from gambling.

What Questions Do We Hope To Answer?

1. Will we be able to predict the outcome of a game based on a team's roster and player statistics?
2. Will certain player statistics be less or more important towards the accuracy of our prediction?
3. Does including more years worth of player statistics help to better predict our outcome, versus just looking at the last season?

Null Hypothesis

There is no correlation between NBA player historical averages, and the final score of a basketball game

Data Source? https://www.basketball-reference.com

Technologies, Tools, Algorithms, Languages Used

- Python 3.9.6
- Conda 4.10.3
- Jupyter-Notebook 6.3.0
- Pandas 1.2.4
- Scikit-learn 1.0.1
- PostgreSQL 14.0
- pgAdmin 4 5.7
- Multiple Linear Regression Model
- Javascript 1.7
- CSS/Bootstrap 4.0.0
- HTML5
- D3.js v5
- Flask 1.1.2

Data Exploration Phase of the Project

- Web Scraping

```
def createPlayerDF(stat, year):
    # Set url for given year
    url = f'https://www.basketball-reference.com/leagues/NBA_{year}_{stat}.html'
    page = requests.get(url)

    # Convert the page html to a soup object
    soup = BeautifulSoup(page.content, 'html.parser')

    # Find the sought after table of data
    table = soup.find_all(class_="full_table")

    # Store the headers/column names
    head = soup.find(class_="thead")
    column_names_raw = [head.text for item in head][0]

    # Clean the column_names_raw list
    column_names = column_names_raw.replace("\n",",").split(",")[2:-1]

    # Create the dataframe
    players = []

    for i in range(len(table)):
        player_ = []

        for td in table[i].find_all("td"):
            player_.append(td.text)

        players.append(player_)

    df = pd.DataFrame(players, columns=column_names).set_index("Player")

    # Cleaning the player's name from occasional special characters
    df.index = df.index.str.replace('*', "", regex=True)

    return df
```

Data Exploration Phase of the Project

- Create Dataframes

```
# Get nba players data into dataframes from the year 2016 - present
currentYear = datetime.now().year
startYear = 2016
year_totals = {}

for year in range(startYear, currentYear+1):
    year_totals[year] = createPlayerDF('totals', str(year))
```

```
# Get nba team rosters into dataframes
nba_teams = teams.get_teams()
nba_team_abr = [team['abbreviation'] for team in nba_teams]
team_rosters = {}

# Convert abreviation for Brooklyn, Pheonix, & Charlotte for basketball-reference.com
nba_team_abr[14] = "BRK"
nba_team_abr[19] = "PHO"
nba_team_abr[29] = "CHO"

for team in nba_team_abr:
    team_rosters[team] = createRosters(team)
```

Data Exploration Phase of the Project

- Data Cleaning

```
games_df
```

ON_ID	TEAM_ID	TEAM_ABBREVIATION	TEAM_NAME	GAME_ID	GAME_DATE	MATCHUP	WL	MIN	PTS	FGM	FGA	FG_PCT	FG3M	FG3A	FG3_PCT	FTM	FTA	FT_
22021	1610612737	ATL	Atlanta Hawks	0022100670	2022-01-19	ATL vs. MIN	None	120	61	20	39	0.513	7	18	0.389	14	16	0
22021	1610612737	ATL	Atlanta Hawks	0022100660	2022-01-17	ATL vs. MIL	W	242	121	38	86	0.442	15	36	0.417	30	32	0
22021	1610612737	ATL	Atlanta Hawks	0022100643	2022-01-15	ATL vs. NYK	L	240	108	40	80	0.500	16	44	0.364	12	17	0
22021	1610612737	ATL	Atlanta Hawks	0022100636	2022-01-14	ATL @ MIA	L	240	118	38	79	0.481	16	38	0.421	26	30	0
22021	1610612737	ATL	Atlanta Hawks	0022100621	2022-01-12	ATL vs. MIA	L	239	91	31	82	0.378	13	45	0.289	16	23	0
22021	1610612737	ATL	Atlanta Hawks	0022100597	2022-01-09	ATL @ LAC	L	238	93	35	78	0.449	7	26	0.269	16	19	0
22021	1610612737	ATL	Atlanta Hawks	0022100589	2022-01-07	ATL @ LAL	L	240	118	43	96	0.448	13	36	0.361	19	27	0

```
In [10]: .drop(columns=["SEASON_ID", "TEAM_ID", "TEAM_ABBREVIATION", "TEAM_NAME", "GAME_ID", "GAME_DATE", "MATCHUP", "WL", "MIN", "FG_P
```

Our Machine Learning Model

In [6]:
```
# Split our preprocessed data into our features and target arrays
X = game_features.drop("PTS", 1)
y = game_features["PTS"]

# Split the preprocessed data into a training and testing dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)
```

In [7]:
```
# Set & fit model for multiple linear regression
model = LinearRegression()
model.fit(X_train, y_train)
```

Our Machine Learning Model

Prediction Accuracy

In [9]:

```
# Evaluate model
print(f'Training Score: {model.score(X_train, y_train)}')
print(f'Testing Score: {model.score(X_test, y_test)}')
```

```
Training Score: 0.6472230888555569
Testing Score: 0.6563165255743577
```

Database Development Process

- SQL

In [14]:
```
# Make the connection string
db_string = f"postgresql://postgres:{db_password}@127.0.0.1:5432/NBA_data"
```

In [15]:
```
# Create database engine
engine = create_engine(db_string)
```

In [16]:
```
player_total_df.to_sql(name='player_totals', con=engine)
```

In [10]:
```
team_roster_df.to_sql(name='team_rosters', con=engine)
```

In [11]:
```
year_total_df.to_sql(name='year_totals', con=engine)
```

Database Development Process

	index bigint	PTS bigint	FGA bigint	FG3A bigint	FTA bigint	OREB bigint	DREB bigint	AST bigint	STL double precision	BLK bigint	TOV bigint	PF bigint
1	0	61	39	18	16	3	9	16	3	2	5	8
2	1	121	86	36	32	10	36	25	7	8	13	18
3	2	108	80	44	17	4	27	23	5	5	9	20
4	3	118	79	38	30	9	25	26	6	6	15	27
5	4	91	82	45	23	6	31	20	8	3	14	15
6	5	93	78	26	19	6	33	19	6	2	13	14
7	6	118	96	36	27	13	41	29	3	0	15	18
8	7	108	86	30	30	10	38	20	8	5	10	14
9	8	131	88	41	24	9	30	31	6	1	12	24
10	9	121	97	38	21	14	30	23	8	4	2	23
11	10	117	92	26	26	10	27	26	6	3	11	17
12	11	118	89	34	26	11	34	22	3	5	10	14
13	12	87	87	38	14	8	38	23	8	5	12	12
14	13	98	82	24	19	11	35	20	5	4	10	19
15	14	98	80	31	29	13	36	21	4	8	14	19
16	15	115	88	24	34	10	27	22	6	3	15	17
17	16	111	85	31	16	6	34	25	11	5	13	16
18	17	126	85	39	31	8	32	20	9	6	12	26
19	18	105	94	34	24	16	33	21	10	3	14	19
20	19	121	90	49	19	11	38	31	5	8	10	20
21	20	127	93	37	18	10	35	29	4	3	10	19

Database Development Process

- SQL