



Universidad de la Sabana - Facultad de Ingeniería

Maestría en Gerencia en Ingeniería

Herramientas de Big Data

Análisis de Bases de datos de NBA para simulación de estadísticas

By

Juan Camilo Martínez Correa

Chía, Colombia junio 2024

Tabla de contenido

Resumen ejecutivo	3
Marco teórico	3
Librerías y componentes instalados en el ambiente de desarrollo	3
Conceptos	1
Herramientas	1
Base de Datos	2
Planteamiento del Problema	2
Pregunta Problema	2
Objetivos	2
Análisis exploratorio	2
Creación del contenedor	2
Revisión de la estructura:	3
Limpieza de Datos	7
Verificación de la integridad de los datos	8
Clustering	8
Data base en AWS	12
Creación de la base de datos	12
Parámetros	12
Visualización de datos	13
Visualización con Power BI	13
Visualización con Python	16
Oportunidades de mejora	17
Modelos de Machine Learning o Redes Neuronales	17
Webscraping	17
Automatización o Prefect	18
Conclusiones	18
Referencias bibliográficas	19
Anexos	22
Anexo 1	22

Resumen ejecutivo

En el proyecto de análisis de datos en la NBA, se utilizan herramientas como SQLAlchemy, Pandas, NumPy, Plotly, Matplotlib, Psycopg2, entre otras, para explorar cómo las características físicas de los jugadores impactan el rendimiento de los equipos. A través de técnicas como el análisis exploratorio, clustering y machine learning, se identifican tendencias y relaciones entre las estadísticas de los jugadores y el éxito de los equipos. Los resultados apuntan a la importancia de considerar aspectos ofensivos y defensivos en las estrategias de juego, demostrando el potencial del análisis de Big Data en el ámbito deportivo y ofreciendo una base sólida para la toma de decisiones basadas en datos en la NBA y más allá.

Marco teórico

El presente marco teórico describe las herramientas, librerías y componentes instalados en el ambiente de desarrollo utilizado para proyectos de análisis de datos, aprendizaje automático y desarrollo de aplicaciones. Este entorno incluye una combinación de herramientas para manejo de bases de datos, visualización de datos, procesamiento de datos y desarrollo de software.

Librerías y componentes instalados en el ambiente de desarrollo

- SQLAlchemy: ORM (Object-Relational Mapping) para Python que facilita la interacción con bases de datos relacionales. Esta librería es independiente del motor de base de datos que se utiliza siempre y cuando implementen el estándar Python DBAPI.
- Pandas: Librería de Python para análisis de datos que proporciona estructuras de datos y herramientas de manipulación, es utilizada para la manipulación y análisis de datos, construida bajo otras dos librerías como lo es Numpy y Matplotlib.
- NumPy: Librería de Python para computación numérica que proporciona matrices multidimensionales y funciones matemáticas, incorporando arrays que permiten representar colecciones de datos en varias dimensiones.
- Ipykernel: Kernel de Jupyter Notebook para soportar Python y otros lenguajes en el entorno de trabajo, permitiendo utilizar diferentes versiones de Python en un entorno virtual o conda, por medio de una buena integración con los notebooks de Jupyter.
- Plotly: Librería de Python para visualización de datos estática y gráficos bien sea de dispersión, líneas, barras, entre otros; brindando una buena personalización según las necesidades que tenga el usuario.
- Matplotlib: Librería de Python para visualización de datos estática y gráficos a partir de datos contenidos en vectores, listas, entre otros; los tipos de gráficos son diagramas de barras, sectores, cajas y bigotes, líneas, áreas, mapas de color, y demás.
- Psycopg2: Adaptador de Python para la base de datos PostgreSQL que permite la conexión y ejecución de consultas, permitiendo establecer conexiones a base de datos PostgreSQL, manipular la base de datos, e incluso ejecutar consultas SQL en la base de datos.
- StringIO: Módulo de Python que permite la manipulación de texto al estilo archivos en memoria; permitiendo ser utilizado como entrada o salida de las distintas funciones de un objeto en un archivo estándar.

Conceptos

- Kmeans: Algoritmo de agrupamiento utilizado en machine learning, permitiendo dividir el conjunto de datos en diferentes grupos denominados K groups o clusters, buscando que sean los más similares entre sí.
- Join: Operación en bases de datos que combina registros de dos o más tablas, permitiendo unir elementos de una secuencia en una cadena a partir de datos encontrados en diferentes tablas o listas con un delimitador específico.
- Machine Learning: Proceso de aprendizaje automático para identificar patrones que relacionan elementos, por medio del entrenamiento de una computadora para que aprenda a encontrar patrones, realizar predicciones sin necesariamente tener una programación definida.
- PostgreSQL: Motor de base de datos relacional de código abierto, que permite tener datos más compatibles, estables, entre otros, soportando datos para no tener restricción de licencia o riesgo de implementación excesiva.
- SQLite: Motor de base de datos ligero y autónomo compatible con SQL, se integra al programa con el que se establece conexión, posibilitando la relación entre tablas, trabajar con base de datos virtuales, entre otras funciones.
- SQL_query: Consulta estructurada que se utiliza para recuperar datos de una base de datos, utilizando una estructura similar al SQL para manipular y filtrar los datos, útil para grandes conjuntos de datos y análisis complejos de estos.
- Cluster: Agrupación de elementos similares en un conjunto de datos, permitiendo almacenar y procesar grandes cantidades de información por medio del conjunto de servidores interconectados que los componen para manipular, almacenar y demás los datos.
- NLP: Procesamiento del lenguaje natural, campo de la inteligencia artificial que se ocupa del entendimiento del lenguaje humano; este se compone de distintos factores como aprendizaje automático, análisis de texto, y lingüística computacional.
- Json: Formato de intercambio de datos ligero y fácil de leer para uso en APIs y comunicaciones, siendo este un lenguaje sencillo y legible, para representar datos estructurados de forma organizada y concisa.
- Tokenizer: Herramienta para dividir texto en unidades más pequeñas, como palabras o frases.

Herramientas

- Docker: Plataforma de contenedores que facilita el despliegue de aplicaciones, utilizando el kernel de Linux para dividir procesos y ejecutarlos de forma independiente permitiendo una implementación rápida de los contenedores creados.
- DBeaver: Herramienta de administración y desarrollo de bases de datos, esta herramienta permite exportar base de datos de forma sencilla y rápida, teniendo una gran compatibilidad con diferentes tipos de datos y una configuración personalizada de estas.
- Miniconda: Distribución mínima de Anaconda, paquete de Python para ciencia de datos, siendo esta una versión más pequeña de Anaconda donde se pueden conectar y descargar distintas librerías, funciones y paquetes.
- Visual Studio: Entorno de desarrollo integrado (IDE) de Microsoft, siendo una plataforma que permite editar, depurar y complicar código, incluyendo distintos compiladores, herramientas de código entre otras funciones para desarrolladores.

- Python: Lenguaje de programación interpretado, versátil y fácil de leer la cual permite automatizar tareas, realizar análisis de datos, entre otros; pudiéndose utilizar en gran variedad de programas.
- Jupyter: Plataforma interactiva para la creación y compartir de documentos con código y visualizaciones; teniendo como principales componentes núcleos y dashboard, para ejecutar un lenguaje y devolver respuestas apropiadas.
- PowerBI: Herramienta de visualización de datos de Microsoft; permitiendo gracias a su colección de servicio, conectores y aplicaciones, convertir orígenes de datos sin relación en información interactiva, coherente y atractiva visualmente.

Base de Datos

Para empezar a entender la información encontrada en la base de datos es importante conocer un poco más sobre la temática que trata, por eso identificar qué es y cómo funciona la NBA, ya que se contara con alrededor de 2.5 GB de información correspondiente al historial de algunas temporadas de la NBA, pero ¿esto qué significa?

La National Basketball League o más conocida como NBA es categorizada como la mejor liga de baloncesto en el mundo, esta tiene origen desde 1946, y consta de 30 franquicias con presencia en Estados Unidos y Canadá.

En nuestra base de datos encontraremos diferentes tablas entre las cuales enlistamos a continuación:

- | | |
|------------------------|----------------------|
| 1. common_player_info | 9. officials |
| 2. draft_combine_stats | 10. other_Stats |
| 3. draft_history | 11. Play_by_play |
| 4. game | 12. Player |
| 5. game_info | 13. team |
| 6. game_summary | 14. team_details |
| 7. inactive_players | 15. team_history |
| 8. line_score | 16. team_info_common |

A lo largo y ancho de las diferentes tablas encontraremos la información correspondiente al histórico de 30 equipos, + 4800 jugadores, 65000 juegos, los resultados de los juegos y una data de juego de más de 13 M de filas.

***Nota:** La Base de datos es tomada de Wyatt walsh NBA data base, en kaggle bajo las licencias CC BY-SA 4.0 (seguir en referencias)

Planteamiento del Problema

En una liga de la elite del baloncesto como la NBA, identificar los factores contribuyentes al éxito de los equipos, para ello se empieza a generar la hipótesis de si las características físicas de los Jugadores pueden poner en ventaja o desventaja a su equipo y, con siglo, identificar esas características a nivel histórico y generar un análisis histórico de la correlación de estos factores

y las victorias del equipo, no solo permite analizar la toma de decisiones de los dueños y puede beneficiar a las personas del entorno de la NBA.

Pregunta Problema

¿Existe una correlación entre las variables físicas de los jugadores cómo son su peso, estatura y salto en el resultado de los juegos entre los diferentes equipos?

Objetivos

- General
 - Identificar si las variables físicas de los jugadores de diferentes equipos que conforman la NBA como lo es estatura, peso, salto y envergadura influyen en los resultados obtenidos por estos.
- Específicos
 - Correlacionar de forma correcta la información para poder obtener la mayor cantidad de datos en temporada por equipo
 - Identificar el comportamiento de los equipos a medida que tienen un nivel de características físicas de sus jugadores por encima del promedio
 - Identificar y fraccionar las victorias de cada equipo por temporada, clasificando datos como puntos a favor y contra.
 - Clasificación de la defensa por equipos mediante el conteo de los bloqueos realizados en el histórico de cada juego

Análisis exploratorio

Teniendo en cuenta los objetivos establecidos es importante empezar a identificar la organización de la información obtenida, con el fin de poder ver cómo podemos trabajar la información y empezar a correlacionar las variables mediante llaves primarias o foráneas

Creación del contenedor

Se utiliza Docker como aplicación para la creación del contenedor donde guardaremos nuestra información de forma local, por lo que se realiza la creación de este bajo un código. yml

```

docker-compose.yml
1  version: '4.29.0'
2  services:
3    db:
4      image: postgres:latest
5      environment:
6        - POSTGRES_DB=NBA
7        - POSTGRES_PASSWORD=password
8        - POSTGRES_USER=user
9      ports:
10       - "5432:5432"
11

```

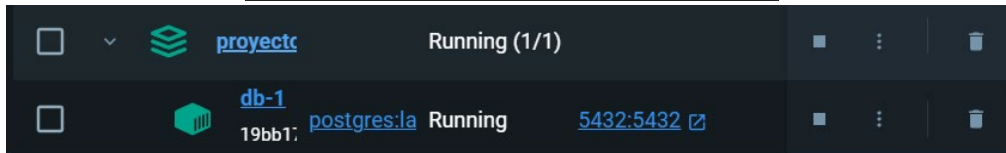


Figura 1: Comprobación de la creación y ejecución del contenedor

Revisión de la estructura:

Inicialmente se identifica que la base de datos se presenta como un SQLite, con el fin de adaptar la información obtenida a una base de datos donde se puede trabajar con consultas más estructuradas, realizando mediante un código en Python la carga de esta información en SQL-postgres a la base de datos “NBA” creada anteriormente.

```

import os
import psycopg2
import csv

# Ruta a la carpeta donde se encuentran los archivos CSV
carpeta_csv = 'C:/Users/JUAN CAMILO/OneDrive - Universidad de La Sabana/MAESTRIA EN GERENCIA EN INGENIERIA/PRIMER SEMESTRE/Herramientas de Big Data/PROYECTO FINAL/NBA/NBA'

# Establecer la conexión a la base de datos PostgreSQL
conexion = psycopg2.connect(
    dbname="NBA",
    user="user",
    password="password",
    host="localhost",
    port="5432"
)

# Crear un cursor para ejecutar comandos SQL
cursor = conexion.cursor()

# Obtener la lista de archivos CSV en la carpeta
archivos_csv = [f for f in os.listdir(carpeta_csv) if f.endswith('.csv')]

# Iterar sobre cada archivo CSV
for archivo_csv in archivos_csv:
    tabla = archivo_csv.split('.')[0] # Nombre de la tabla (sin extensión)
    ruta_completa = os.path.join(carpeta_csv, archivo_csv)
    with open(ruta_completa, 'r', newline='') as f:
        lector_csv = csv.reader(f)
        encabezado = next(lector_csv) # Leer la primera fila como encabezado
        # Verificar si la tabla existe
        cursor.execute(f"SELECT EXISTS (SELECT 1 FROM information_schema.tables WHERE table_name = '{tabla}')")
        tabla_existe = cursor.fetchone()[0]
        # Si la tabla no existe, crearla
        if not tabla_existe:
            cursor.execute(f"CREATE TABLE {tabla} ({', '.join([f'({columna}) VARCHAR(255)' for columna in encabezado])})")
            conexion.commit()
        # Insertar los datos en la tabla
        for fila in lector_csv:
            placeholders = ', '.join(['%' for _ in range(len(encabezado))]) # Crear los marcadores de posición (%)
            consulta = f"INSERT INTO {tabla} ({', '.join(encabezado)}) VALUES ({placeholders})"
            cursor.execute(consulta, fila) # Utilizar los valores de la fila como argumento para la consulta
        cursor.execute(consulta, fila) # Utilizar los valores de la fila como argumento para la consulta
        conexion.commit()

# Cerrar el cursor y la conexión
cursor.close()
conexion.close()

```

Figura 2: Código para cargar las tablas a la base de datos postgres a partir de los CSV

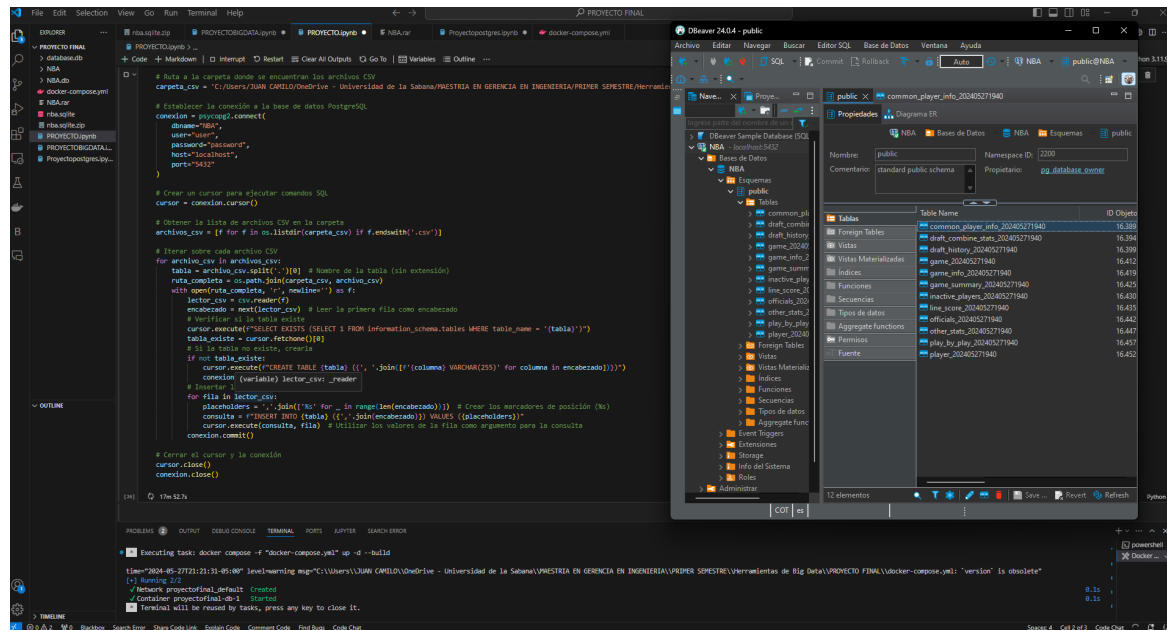


Figura 2.1: Código para cargar las tablas a la base de datos postgres a partir de los CSV

Una vez cargadas las tablas, se procede a realizar una consulta para identificar la carga a nuestra base de datos y obtener el título para llamarlas de forma adecuada.

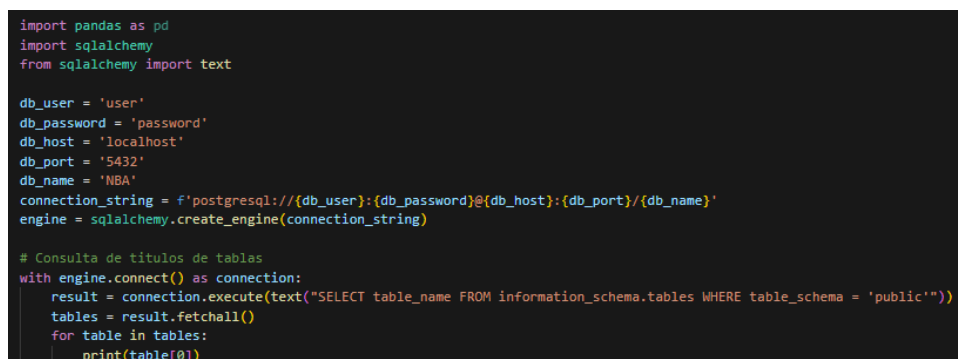


Figura 3: Código para revisión de las tablas cargadas a la base de datos

Para continuar con la revisión de la estructuración de nuestras tablas, se utiliza inicialmente la aplicación de DBeaver, para empezar a identificar las correlaciones que podemos hacer entre tablas, del mismo modo utilizaremos consultas de SQL, mediante Python y mediante DBeaver directamente.

Una vez realizada la conexión de DBeaver con la base de datos, se procede a utilizar la herramienta de ER para crear las primeras correlaciones entre las tablas a utilizar. (Ver completo en anexo 1)

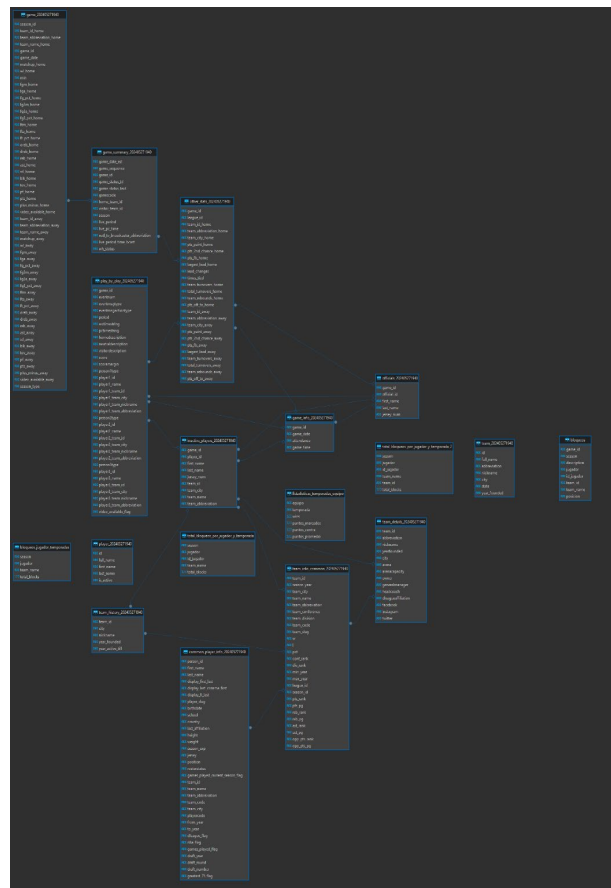


Figura 4: Diagrama ER de la base de datos

La primera consulta se realiza para identificar que se tenga toda la información correspondiente de a qué equipo pertenece cada jugador.

```
engine = sqlalchemy.create_engine('postgresql://user:password@localhost:5432/NBA')
sql_query = '''
SELECT p.full_name, ip.team_name
FROM player_202405271940 p
JOIN inactive_players_202405271940 ip ON p.id = ip.player_id
WHERE p.is_active = '1' AND ip.team_name = 'Cavaliers'
GROUP BY p.full_name, ip.team_name
'''
T1 = pd.read_sql(sql_query, engine)
T1.head()
```

	full_name	team_name
0	Andre Drummond	Cavaliers
1	Caris LeVert	Cavaliers
2	Cedi Osman	Cavaliers
3	Collin Sexton	Cavaliers
4	Danny Green	Cavaliers

Figura 5: Funcionamiento de las entidades y relación

Se procede a realizar varias consultas para indagar un poco más de los datos.

```
## LISTADO DE EQUIPOS
import pandas as pd
from sqlalchemy import create_engine

# Conexión a la base de datos PostgreSQL
engine = create_engine('postgresql://user:password@localhost:5432/NBA')

# Ejecutar la consulta para obtener la lista de equipos
query = """
SELECT team_name, COUNT(person_id) AS player_count
FROM common_player_info_202405271940
GROUP BY team_name;
"""

teams_df = pd.read_sql(query, engine)
teams = teams_df['team_name'].tolist()
print(teams)
```

```
# Consulta SQL
consulta = '''
WITH juego AS (
    SELECT
        g.game_id AS Numero_de_juego,
        gs.season AS Temporada,
        g.team_name_home AS Nombre_local,
        g.team_name_away AS Nombre_visitante,
        g.wl_home as Ganados_local,
        g.wl_away as Ganados_visitante,
        g.pts_home as Puntos_local,
        g.pts_away as Puntos_visitantes
    FROM
        game_202405271940 g
    JOIN
        game_summary_202405271940 gs ON g.game_id = gs.game_id
    WHERE
        gs.season::INTEGER >= 2020
)
SELECT *
FROM juego
ORDER BY Temporada DESC;
'''

# Leer datos usando pandas
juegos_temporada = pd.read_sql(consulta, engine)

# Subir la tabla a la base de datos
juegos_temporada.to_sql('juegos_temporada', con=engine, if_exists='replace', index=False)
```

```
# Consulta SQL
consulta2 = '''SELECT jt.nombre_local AS Equipo, temporada,
    SUM(CASE WHEN ganados_local = 'W' THEN 1 ELSE 0 END) AS wins,
    SUM(CAST(puntos_local AS float)) as Puntos_Marcados,
    SUM(CAST(puntos_visitantes AS float)) as Puntos_contra,
    (SUM(CAST(puntos_local AS float))/41) as Puntos_Promedio
FROM juegos_temporada jt
GROUP BY nombre_local, temporada

UNION ALL

-- Select wins and points for each team as visiting team for all seasons
SELECT jt.nombre_visitante AS Equipo, temporada,
    SUM(CASE WHEN ganados_visitante = 'W' THEN 1 ELSE 0 END) AS wins,
    SUM(CAST(puntos_visitantes AS float)) as Puntos_Marcados,
    SUM(CAST(puntos_local AS float)) as Puntos_contra,
    (SUM(CAST(puntos_visitantes AS float))/41) as Puntos_Promedio
FROM juegos_temporada jt
GROUP BY nombre_visitante, temporada

ORDER BY temporada, wins DESC;
'''

# Leer datos usando pandas
Estadisticas_temporadas_equipo = pd.read_sql(consulta2, engine)

# Subir la tabla a la base de datos
Estadisticas_temporadas_equipo.to_sql('Estadisticas_temporadas_equipo', con=engine, if_exists='replace', index=False)
```

Figura 6: Creación de tablas de interés

Limpieza de Datos

Una vez realizadas las consultas iniciales, se logra identificar que hay que realizar varias “limpiezas” de datos, ya que al ser datos históricos se encontraron registros donde cierta información es nula, lo que puede generar algunos problemas en los análisis realizados, es por este motivo que hay que realizar unas futuras consultas con indicaciones de que no genere datos nulos.

season	jugador	team_id	id_jugador	total_blocks	height_wo_shoes	weight	wingspan	standing_reach	max_vertical_leap
2020	Cam Reddish	1610612737	1629629	4	78.5	207.8	84.5	105.5	
2020	De'Andre Hunter	1610612737	1629631	4					
2020	Tony Snell	1610612737	203503	4	78.0	198.2	83.5	105.5	36.5
2020	Bruno Fernando	1610612737	1628981	4	80.75	233.2	88.25	109.0	35.0
2020	Kris Dunn	1610612737	1627739	4	75.0		81.5	100.0	
2020	John Collins	1610612737	1628381	4	80.25	225.2	83.25	106.5	37.5
2020	Bruno Fernando	1610612737	1628981	4	80.75	237	87.25	110.0	33.5
2020	Skylar Mays	1610612737	1630219	4	75.0	204	78.0	99.0	37.5
2020	Nathan Knight	1610612737	1630233	4	80.0	235	86.0	109.5	31.5
2020	Solomon Hill	1610612737	203524	4	77.5	226	81.0	103.0	37.5
2020	Trae Young	1610612737	1629027	4	72.5	177.8	75.0	95.5	
2020	Grant Williams	1610612738	1629684	4	77.75	240.2	81.75	104.5	31.5
2020	Tristan Thompson	1610612738	202684	4	79.5	227.4	85.25	108.5	35.0
2020	Moritz Wagner	1610612738	1629021	4	82.5	241.4	84.0	108.0	34.0
2020	Kemba Walker	1610612738	202689	4	71.5	184	75.5	91.5	39.5
2020	Carson Edwards	1610612738	1629035	4	70.75	195.6	78.25	94.5	34.5
2020	Semi Ojeleye	1610612738	1628400	4	77.25	241.4	81.75	102.0	40.5
2020	Romeo Langford	1610612738	1629641	4	76.5		83.0	103.0	
2020	Jeff Teague	1610612738	201952	4	72.25	175.2	79.5	98.5	36.5
2020	Tremont Waters	1610612738	1629682	4	69.5	172.4	74.25	93.5	40.5
2020	Moritz Wagner	1610612738	1629021	4	82.0	231.2	84.0	108.0	32.5
2020	Marcus Smart	1610612738	203935	4	74.0	227.2	81.25	99.0	36.0
2020	Thon Maker	1610612739	1627748	4	83.75	216	87.0	110.5	36.5
2020	Dylan Windler	1610612739	1629685	4	78.25	195.8	82.0	104.5	37.5
2020	Jarrett Allen	1610612739	1628386	4	81.0	233.6	89.25	109.5	35.5
2020	Lamar Stevens	1610612739	1630205	4	77.75	228	81.0	101.0	41.0
2020	Taurean Prince	1610612739	1627752	4	78.5	220.2	83.5	102.0	36.0
2020	JaVale McGee	1610612739	201580	4	83.0	241	90.0	114.5	32.5

Figura 7: Identificación de datos con registros

Mediante esta limpieza también se identifica que, para poder utilizar la información de manera óptima, se puede utilizar las últimas temporadas como fuente de datos, ya que estas cuentan con la información más completa, lo cual lleva a utilizar consultas con “filtros” de fecha bajo la columna “season”, limpiando así los datos.

Una vez identificadas estas falencias en el data set, se procede a generar nuevas tablas, donde estas ya no tendrán datos innecesarios.

```
create table Caracteristicas_fisicas_bloques2 as
SELECT
    tbpjyt2.season,
    tbpjyt2.jugador,

    tbpjyt2.team_id,
    tbpjyt2.id_jugador,
    COUNT(*) AS total_blocks,
    dcs.height_wo_shoes,
    dcs.weight,
    dcs.wingspan,
    dcs.standing_reach,
    dcs.max_vertical_leap
FROM
    total_bloques_por_jugador_y_temporada_2 tbpjyt2
JOIN
    common_player_info_202405271940 cpi ON tbpjyt2.id_jugador = cpi.person_id
JOIN
    draft_combine_stats_202405271940 dcs ON tbpjyt2.id_jugador = dcs.player_id
GROUP BY
    tbpjyt2.season, tbpjyt2.jugador, tbpjyt2.team_id, tbpjyt2.id_jugador, dcs.height_wo_shoes, dcs.weight, dcs.wingspan, dcs.standing_reach, dcs.max_vertical_leap
ORDER BY
    tbpjyt2.season, tbpjyt2.team_id, total_blocks DESC;
```

Figura 8: Ejemplo consultas limpieza de datos

Verificación de la integridad de los datos

Para la verificación de la limpieza de los datos, se realiza la búsqueda de registros nulos, buscando identificar nuevas falencias que puedan afectar el análisis.

nbc_game_date_est	nbc_game_id	nbc_season	nbc_description	nbc_jugador	nbc_id_jugador	nbc_team_id	nbc_team_name	nbc_position
2020-12-23 00:00:00	0022000020	2020	Lillard BLOCK (1 BLK)	Mike Conley	201144	1610612762	Jazz	Guard
2020-12-23 00:00:00	0022000020	2020	Lillard BLOCK (1 BLK)	Mike Conley	201144	1610612762	Jazz	Guard
2020-12-23 00:00:00	0022000020	2020	Lillard BLOCK (1 BLK)	Mike Conley	201144	1610612762	Jazz	Guard
2020-12-23 00:00:00	0022000020	2020	Lillard BLOCK (1 BLK)	Mike Conley	201144	1610612762	Jazz	Guard
2020-12-23 00:00:00	0022000020	2020	Lillard BLOCK (1 BLK)	Mike Conley	201144	1610612762	Jazz	Guard
2020-12-23 00:00:00	0022000020	2020	Lillard BLOCK (1 BLK)	Mike Conley	201144	1610612762	Jazz	Guard
2020-12-23 00:00:00	0022000020	2020	Lillard BLOCK (1 BLK)	Mike Conley	201144	1610612762	Jazz	Guard
2020-12-23 00:00:00	0022000020	2020	Lillard BLOCK (1 BLK)	Mike Conley	201144	1610612762	Jazz	Guard
2020-12-23 00:00:00	0022000020	2020	Lillard BLOCK (1 BLK)	Mike Conley	201144	1610612762	Jazz	Guard
2020-12-23 00:00:00	0022000020	2020	Lillard BLOCK (1 BLK)	Mike Conley	201144	1610612762	Jazz	Guard
2020-12-23 00:00:00	0022000020	2020	Lillard BLOCK (1 BLK)	Mike Conley	201144	1610612762	Jazz	Guard
2020-12-23 00:00:00	0022000020	2020	Lillard BLOCK (1 BLK)	Mike Conley	201144	1610612762	Jazz	Guard
2020-12-23 00:00:00	0022000020	2020	Lillard BLOCK (1 BLK)	Mike Conley	201144	1610612762	Jazz	Guard
2020-12-23 00:00:00	0022000020	2020	Lillard BLOCK (1 BLK)	Mike Conley	201144	1610612762	Jazz	Guard
2020-12-23 00:00:00	0022000020	2020	Giles III BLOCK (1 BLK)	Miyi Oni	1629671	1610612762	Jazz	Guard-Forward
2020-12-23 00:00:00	0022000020	2020	Giles III BLOCK (1 BLK)	Miyi Oni	1629671	1610612762	Jazz	Guard-Forward
2020-12-23 00:00:00	0022000020	2020	Giles III BLOCK (1 BLK)	Miyi Oni	1629671	1610612762	Jazz	Guard-Forward
2020-12-23 00:00:00	0022000020	2020	Giles III BLOCK (1 BLK)	Miyi Oni	1629671	1610612762	Jazz	Guard-Forward
2020-12-23 00:00:00	0022000018	2020	Towns BLOCK (1 BLK)	Mason Plumlee	203486	1610612765	Pistons	Forward-Center
2020-12-23 00:00:00	0022000018	2020	Towns BLOCK (1 BLK)	Mason Plumlee	203486	1610612765	Pistons	Forward-Center
2020-12-23 00:00:00	0022000018	2020	Towns BLOCK (1 BLK)	Mason Plumlee	203486	1610612765	Pistons	Forward-Center
2020-12-23 00:00:00	0022000018	2020	Towns BLOCK (1 BLK)	Mason Plumlee	203486	1610612765	Pistons	Forward-Center
2020-12-23 00:00:00	0022000018	2020	Towns BLOCK (1 BLK)	Mason Plumlee	203486	1610612765	Pistons	Forward-Center
2020-12-23 00:00:00	0022000018	2020	Towns BLOCK (1 BLK)	Mason Plumlee	203486	1610612765	Pistons	Forward-Center
2020-12-23 00:00:00	0022000018	2020	Towns BLOCK (1 BLK)	Mason Plumlee	203486	1610612765	Pistons	Forward-Center
2020-12-23 00:00:00	0022000018	2020	Towns BLOCK (1 BLK)	Mason Plumlee	203486	1610612765	Pistons	Forward-Center
2020-12-23 00:00:00	0022000004	2020	Jones BLOCK (1 BLK)	Dwight Powell	203939	1610612742	Mavericks	Forward-Center

Figura 9: Comprobación efectividad de limpieza de datos

Clustering

Una vez se obtiene la tabla con la información limpia, es importante empezar a observar la correlación entre las variables, con el fin de llegar a la agrupación de equipos con características similares, en este caso se utilizarían las columnas de variables numéricas para poder realizar las agrupaciones en los clusters, del mismo modo se utilizará la técnica del codo para identificar el número de clúster a realizar.

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import seaborn as sns

# Cargar el dataset
df = pd.read_csv('C:/Users/JUAN CARLO/OneDrive - Universidad de la Sabana/MAESTRIA EN GERENCIA EN INGENIERIA/PRIMER SEMESTRE/Herramientas de Big Data/PROYECTO FINAL/analitica_limpia_202405300005.csv')

# Seleccionar las columnas relevantes
features = [
    'wins',
    'avg_height_wo_shoes',
    'avg_weight',
    'avg_wingspan',
    'avg_standing_reach',
    'avg_max_vertical_leap',
    'total_blocks',
    'puntos_promedio',
    'puntos_atracados',
    'puntos_contra'
]

X = df[features]

# Normalizar los datos
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Determinar el número óptimo de clusters usando el método del codo
inertia = []
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k, random_state=0)
    kmeans.fit(X_scaled)
    inertia.append(kmeans.inertia_)

plt.figure(figsize=(9, 5))
plt.plot(range(1, 11), inertia, marker='o')
plt.title('Método del codo')
plt.xlabel('Número de clusters')
plt.ylabel('Inercia')
plt.show()

# Elegir el número óptimo de clusters (e.g., 3)
n_clusters = 3
kmeans = KMeans(n_clusters=n_clusters, random_state=0)
df['cluster'] = kmeans.fit_predict(X_scaled)

# Visualizar los clusters
sns.pairplot(df, vars=features, hue='cluster', palette='viridis')
plt.show()

# Analisis de los resultados
for cluster in range(0, n_clusters):
    print(f"Cluster {cluster}")
    print(df[df['cluster'] == cluster][features].describe())
    print()
```

Figura 10: Código para la clusterización

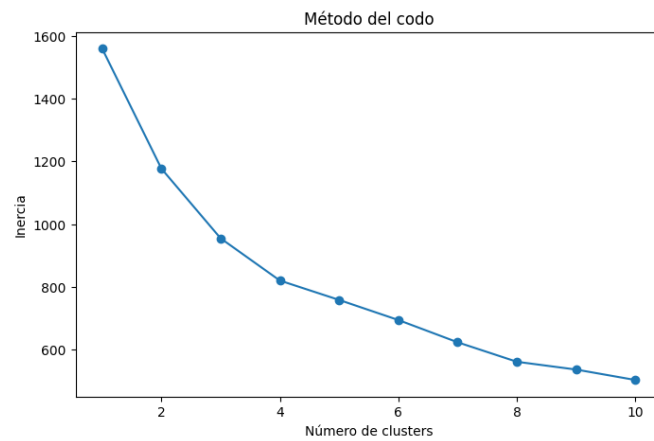


Figura 11: Metodología del Codo

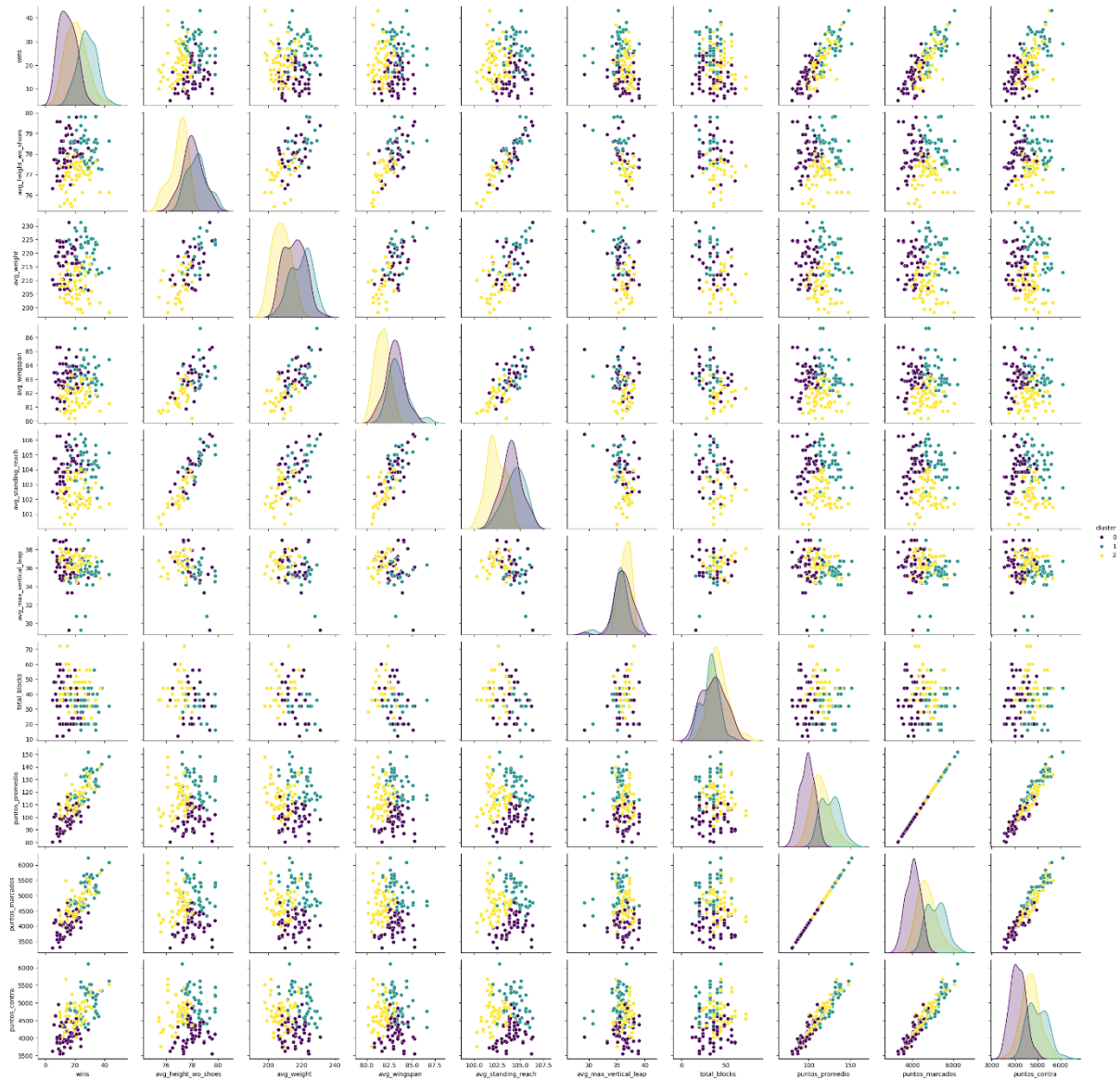


Figura 12: Gráfica de la clusterizacion 2D

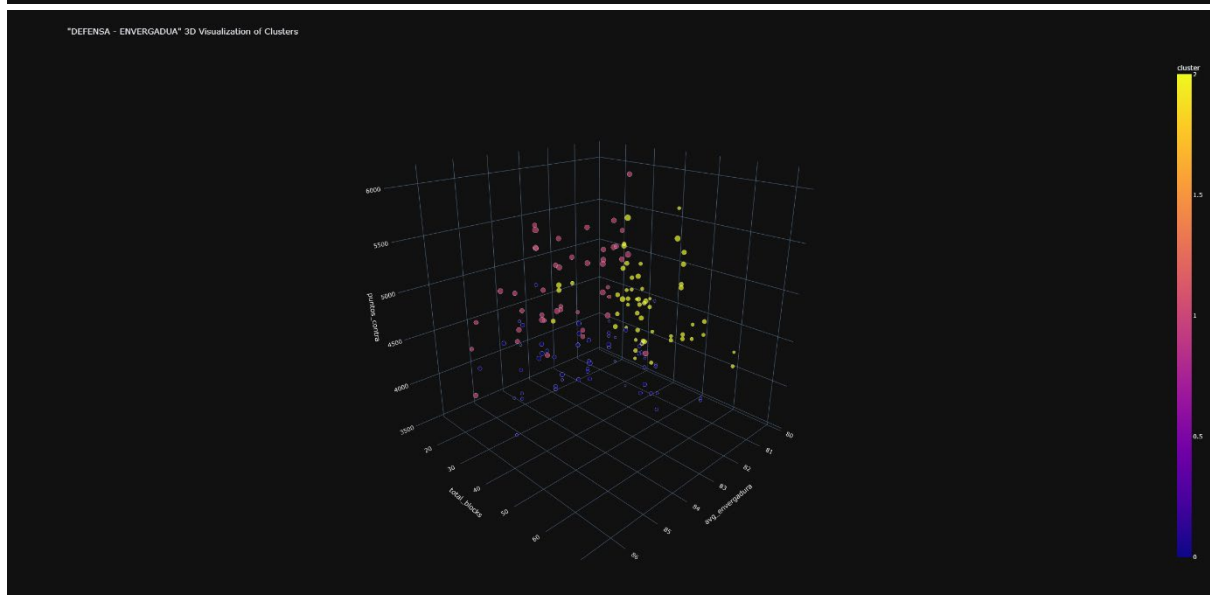
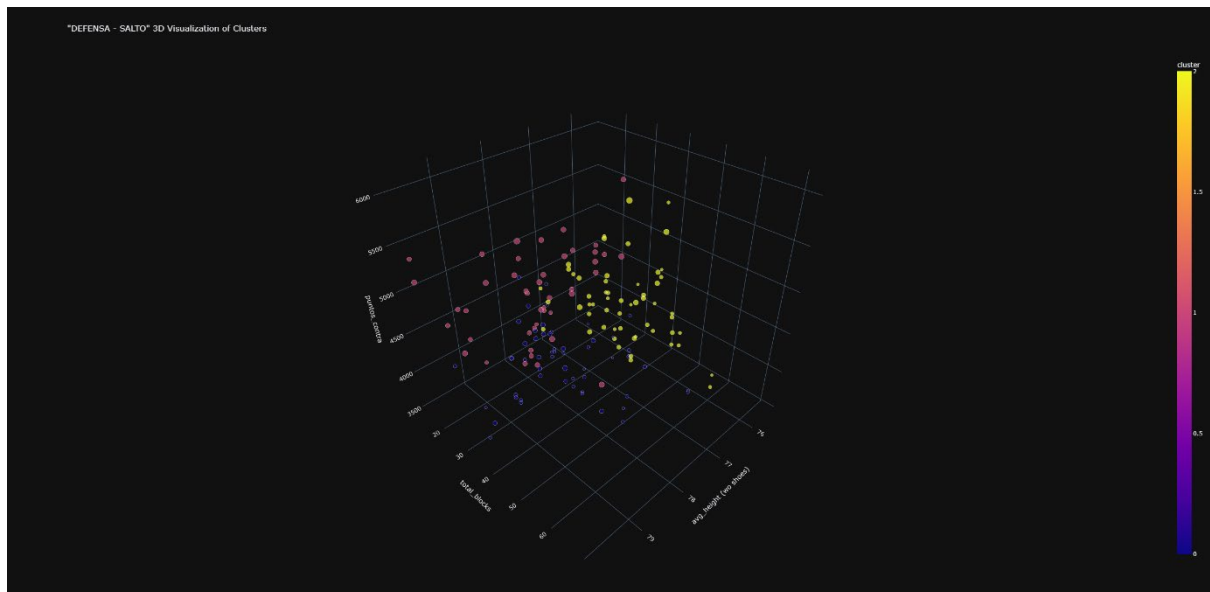
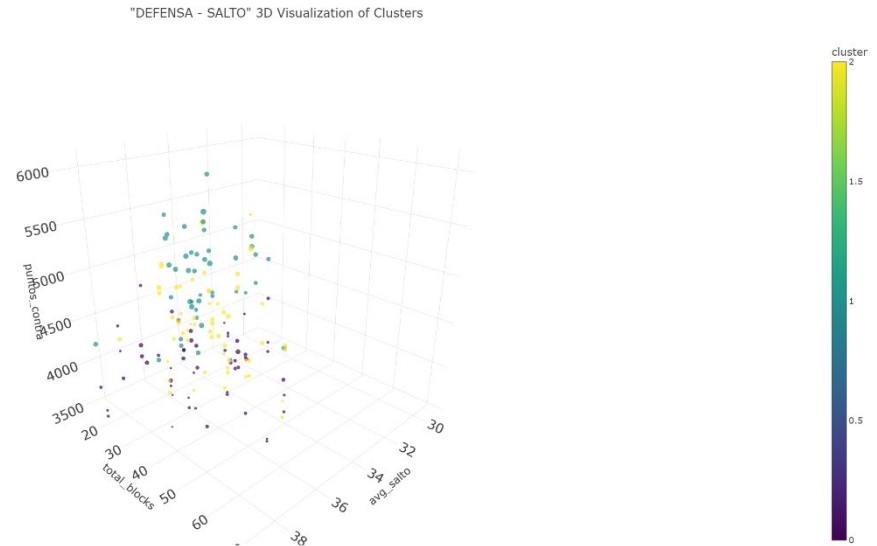


Figura 13: Gráfica de la clusterizacion 3D

Análisis Clusters

En las gráficas de tres dimensiones se evalúan las variables defensivas de bloqueos, puntos en contra y en cada grafica un aspecto físico, relacionados con Salto Máximo, Altura y Envergadura. La variable de partidos ganados se identifica con el tamaño de cada punto. Esto nos permite evaluar el comportamiento de los equipos independientemente para cada temporada, y observar la relación entre equipos con características similares en cuanto a defensa se refiere.

Si bien la técnica del codo nos permite identificar 3 clusters viables para el análisis, en las gráficas se observan cierta agrupación de los equipos según las variables, pero en ningún caso se encuentran clusters claramente identificables, por lo cual se puede apuntar en primera instancia que las variables físicas y desempeño defensivo no están directamente relacionados con el desempeño de los equipos durante cada temporada.

Estadísticas descriptivas

Una vez identificadas las llaves de las tablas creadas se procede a realizar varias consultas SQL, uniendo diferentes datos de tablas mediante sentencias “JOIN” y “WHERE” con el fin de por iniciar a realizar estadística con data relevante y útil.

Data base en AWS

Creación de la base de datos

- Se crea servidor para alojamiento de base de datos de los resultados obtenidos luego del análisis anterior, para lo cual se asignan los siguientes parámetros de creación:
- Creación de base de datos: Creación Estándar
- Opciones del Motor: PostgreSQL
- Versión del motor: PostgreSQL 16.1-R2
- Identificador de instancias de bases de datos: nba
- Nombre de usuario maestro: Postgres
- Administración de Credenciales: Autoadministrado
- Contraseña maestra: 12345678
- Tipo de almacenamiento: SSD de uso general (gp2)
- Almacenamiento asignado: 20
- Acceso público: Sí
- Puerto de la base de datos: 5434

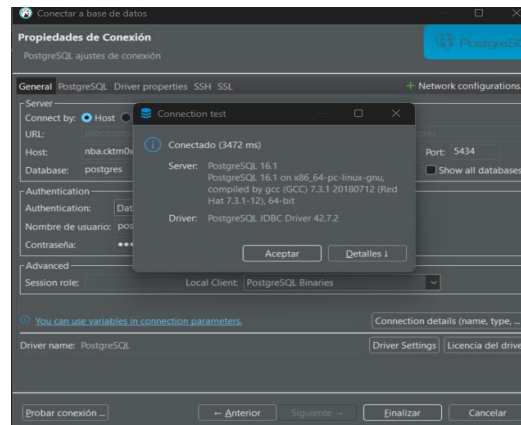
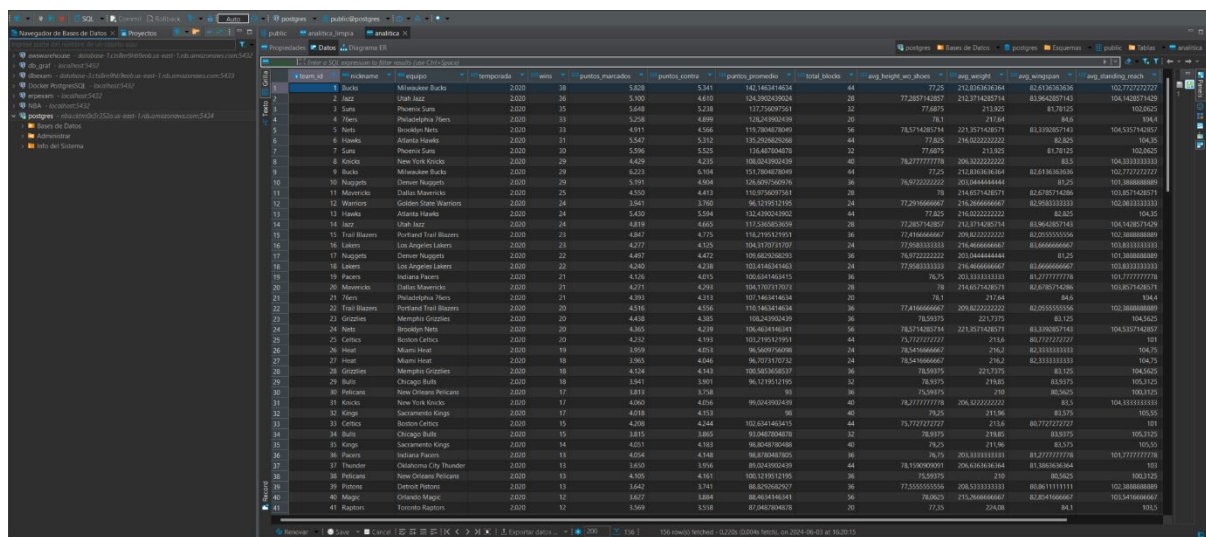
Parámetros

Luego de creado el acceso AWS, es necesario dirigirse a la configuración de grupos de seguridad de la VPC, procedemos a editar las reglas de entrada y agregar una nueva regla:

- Tipo: TCP personalizado (Esto se hace para habilitar el uso del puerto 5434)

- Intervalos de Puerto: 5434
- Origen: Anywhere-IPv4
- Finalmente se genera la database en AWS con:
- Endpoint: nba.cktm0x5r352o.us-east-1.rds.amazonaws.com
- Puerto:5434
- database name: postgres

Prueba de conexión a la base de datos usando **DBeaver**

team_id	nombre	equipo	temporada	wins	puntos	puntos_por_jugador	puntos_porcentaje	puntos_porcentaje	total_blocks	avg_height_wings	avg_weight	avg_wingspan	avg_standing_reach
1	Bucks	Milwaukee Bucks	2020	28	5,623	5,341	142,140,814	77,25	212,317,142,8714	82,619,368,324	102,772,727,272	102,772,727,272	102,772,727,272
2	Jazz	Utah Jazz	2020	36	5,500	4,410	124,390,430,824	32	77,280,742,857	212,317,142,8714	81,964,807,743	104,142,871,429	104,142,871,429
3	Suns	Phoenix Suns	2020	35	5,648	5,238	137,740,977,581	28	77,6875	211,921	81,781,25	104,142,871,429	104,142,871,429
4	Pacers	Indiana Pacers	2020	31	5,228	4,499	132,621,990,439	20	76,1	211,64	84,6	104,142,871,429	104,142,871,429
5	Nets	Brooklyn Nets	2020	33	4,911	4,566	116,786,467,849	20	76,157,142,8714	221,317,142,8714	81,190,807,743	104,142,871,429	104,142,871,429
6	Hawks	Atlanta Hawks	2020	31	5,567	5,312	135,292,602,586	44	77,825	216,022,222,222	82,825	104,142,871,429	104,142,871,429
7	Suns	Phoenix Suns	2020	20	5,586	5,325	134,407,704,879	32	77,6875	211,921	81,781,25	104,142,871,429	104,142,871,429
8	Knicks	New York Knicks	2020	29	4,429	4,235	106,024,990,439	40	76,177,777,778	206,322,222,222	83,5	104,142,871,429	104,142,871,429
9	Raptors	Mississippi Bulls	2020	29	6,223	6,104	151,786,467,849	44	77,25	212,317,142,8714	82,619,368,324	102,772,727,272	102,772,727,272
10	Nuggets	Denver Nuggets	2020	29	5,391	4,964	124,390,430,824	36	76,122,222,222	201,944,444,444	81,25	104,142,871,429	104,142,871,429
11	Mavericks	Dallas Mavericks	2020	25	4,550	4,413	116,973,607,751	24	78	214,657,142,8714	82,678,742,867	104,142,871,429	104,142,871,429
12	Warriors	Golden State Warriors	2020	24	5,041	5,760	94,129,121,995	28	77,299,666,667	216,266,666,667	82,581,131,131	102,772,727,272	102,772,727,272
13	Hornets	Atlanta Hawks	2020	24	5,450	5,294	132,621,990,439	44	77,825	216,022,222,222	82,825	104,142,871,429	104,142,871,429
14	Jazz	Utah Jazz	2020	24	4,819	4,665	117,586,467,849	36	77,280,742,857	212,317,142,8714	81,964,807,743	104,142,871,429	104,142,871,429
15	Trail Blazers	Portland Trail Blazers	2020	23	4,867	4,775	114,719,517,951	36	77,666,667	206,322,222,222	82,655,555,556	104,142,871,429	104,142,871,429
16	Lakers	Los Angeles Lakers	2020	23	4,277	4,125	104,129,121,995	34	77,593,833,333	216,466,666,667	81,666,666,667	104,142,871,429	104,142,871,429
17	Nuggets	Denver Nuggets	2020	22	4,497	4,472	108,623,502,586	34	76,972,222,222	201,944,444,444	81,25	104,142,871,429	104,142,871,429
18	Lakers	Los Angeles Lakers	2020	22	4,240	4,236	104,142,871,429	34	77,593,833,333	216,466,666,667	81,666,666,667	104,142,871,429	104,142,871,429
19	Pacers	Indiana Pacers	2020	21	4,126	4,015	104,142,871,429	36	76,25	201,944,444,444	81,25	104,142,871,429	104,142,871,429
20	Mavericks	Dallas Mavericks	2020	21	4,271	4,269	104,170,717,951	28	78	214,657,142,8714	82,678,742,867	104,142,871,429	104,142,871,429
21	Nets	Brooklyn Nets	2020	21	4,293	4,213	107,142,871,429	40	76,1	211,64	84,6	104,142,871,429	104,142,871,429
22	Trail Blazers	Portland Trail Blazers	2020	20	4,516	4,556	116,146,341,464	36	77,666,667	206,322,222,222	82,655,555,556	104,142,871,429	104,142,871,429
23	Grizzlies	Memphis Grizzlies	2020	20	4,438	4,385	104,249,902,439	36	76,59375	221,7375	81,125	104,142,871,429	104,142,871,429
24	Nets	Brooklyn Nets	2020	20	4,365	4,239	106,462,614,641	36	76,257,142,8714	221,317,142,8714	81,190,807,743	104,142,871,429	104,142,871,429
25	Celtics	Boston Celtics	2020	20	4,259	4,193	103,795,517,951	44	75,772,727,272	215,8	80,772,727,272	104,142,871,429	104,142,871,429
26	Hornets	Atlanta Hawks	2020	19	4,599	4,653	94,569,750,986	24	76,549,666,667	216,2	82,333,333,333	104,142,871,429	104,142,871,429
27	Hornets	Atlanta Hawks	2020	18	3,565	4,046	94,702,317,951	24	76,549,666,667	216,2	82,333,333,333	104,142,871,429	104,142,871,429
28	Grizzlies	Memphis Grizzlies	2020	18	4,124	4,143	106,165,360,837	36	76,59375	221,7375	81,125	104,142,871,429	104,142,871,429
29	Bulls	Chicago Bulls	2020	18	3,641	3,591	94,129,121,995	32	76,59375	219,85	81,59375	104,142,871,429	104,142,871,429
30	Pelicans	New Orleans Pelicans	2020	17	3,818	3,736	91,81	36	76,59375	219,85	81,59375	104,142,871,429	104,142,871,429
31	Knicks	New York Knicks	2020	17	4,060	4,096	96,024,990,439	40	76,177,777,778	206,322,222,222	83,5	104,142,871,429	104,142,871,429
32	Kings	Sacramento Kings	2020	17	4,018	4,553	96	40	76,25	211,96	81,59375	104,142,871,429	104,142,871,429
33	Celtics	Boston Celtics	2020	15	4,289	4,244	101,624,146,341	44	73,772,727,272	215,8	80,772,727,272	104,142,871,429	104,142,871,429
34	Bulls	Chicago Bulls	2020	15	3,815	3,805	91,047,704,879	32	76,59375	219,85	81,59375	104,142,871,429	104,142,871,429
35	Kings	Sacramento Kings	2020	14	4,051	4,168	98,048,786,467	40	76,25	211,96	81,59375	104,142,871,429	104,142,871,429
36	Pacers	Indiana Pacers	2020	13	4,064	4,146	94,702,317,951	36	76,76	201,944,444,444	81,25	104,142,871,429	104,142,871,429
37	Thunder	Oklahoma City Thunder	2020	13	3,650	3,954	89,024,990,439	44	76,150,900,901	206,316,361,64	81,59375	104,142,871,429	104,142,871,429
38	Pelicans	New Orleans Pelicans	2020	13	4,105	4,161	94,129,121,995	36	76,59375	219,85	81,59375	104,142,871,429	104,142,871,429
39	Pelicans	New Orleans Pelicans	2020	13	3,640	3,741	88,206,262,587	36	77,055,555,556	206,313,131,131	80,811,111,111	104,142,871,429	104,142,871,429
40	Magi	Cleveland Magi	2020	12	3,627	3,894	88,461,414,141	36	76,0625	215,266,666,667	82,851,566,667	104,142,871,429	104,142,871,429
41	Raptors	Toronto Raptors	2020	12	3,569	3,518	87,048,786,467	32	77,25	214,6	84,1	104,142,871,429	104,142,871,429

Figura 14: Prueba de conexión de la base de datos con DBeaver

Visualización de datos

Visualización con Power BI

Power BI, tiene diferentes tipos de conexiones para obtener los datos, se toma en cuenta esta versatilidad para utilizar una conexión al local host donde se obtendrá la información

Base de datos PostgreSQL

Servidor

Base de datos

Modo Conectividad de datos

☒ Importar
 ☐ DirectQuery

Opciones avanzadas

Aceptar

Cancelar

Figura 15: Método de conexión Power BI - Sqlpostgres

Luego de realizar exitosamente la conexión, se procede a importar las tablas de la data ya limpia, para empezar, volver a realizar las conexiones entre llaves y empezar a trabajar los datos.

Tablero 1

Hoja 1

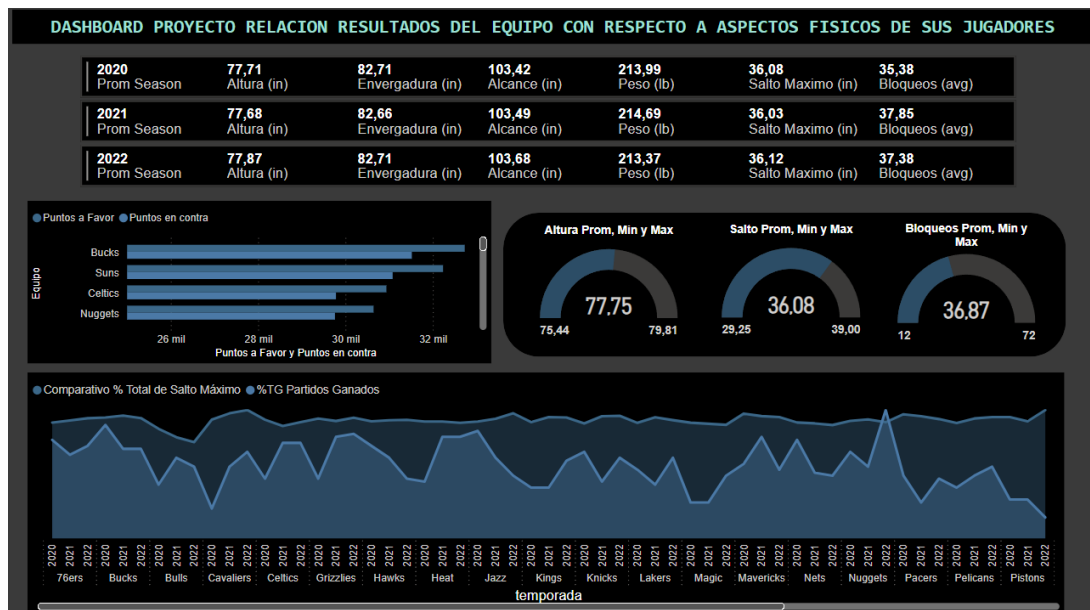


Figura 16: Analítica características físicas y desempeño de equipos

El informe dinámico permite observar el promedio de cada aspecto físico por temporada en la parte superior, además de que permite tener una consulta rápida de los informes por temporada. En este caso durante las tres temporadas se puede observar que los factores físicos promedios no tienen un cambio importante durante las tres temporadas, aunque si se logra identificar una mejora significativa en el alcance. Por parte de los equipos con mayor cantidad de puntos a favor y en contra resaltan los Bucks, Suns, Celtics y Nuggets, y se puede observar que los equipos con mayor cantidad de puntos a favor, también cuenta con una gran cantidad de puntos en contra, lo cual permite identificar una marcada tendencia a la ofensiva, clave en el resultado de los equipos.

Los análisis KPI's permiten identificar la Altura, Salto y bloqueos y analizarlos contra el mínimo y máximo de cada variable. Si bien el promedio de altura y salto están por encima del

punto central, se puede observar que los bloqueos no son comunes, salvo algunos casos excepcionales.

Para el análisis final se encuentra la gráfica que permite compara la relación del salto máximo con los partidos ganados de cada equipo por temporada, esto como valor porcentual del máximo en cada caso, para hacerlos visualmente comparables. Con esta grafica se puede identificar claramente que no hay una variación considerable en cuanto al salto, y salvo los bulls en 2022 con un valor mínimo, su desempeño fue un desempeño promedio.

En cuanto a la variación de victorias vemos equipos con las mismas características, pero con resultados completamente diferentes, lo que permite concluir que las variables físicas como tal no presentan mayor variación entre los equipos de la NBA, y tampoco son determinantes en el desempeño de los equipos, o por lo menos no son el único aspecto que influye, a eso se debería sumar variables como táctica, técnica, talento y estrategia, claves para la consecución de objetivos.

Hoja 2

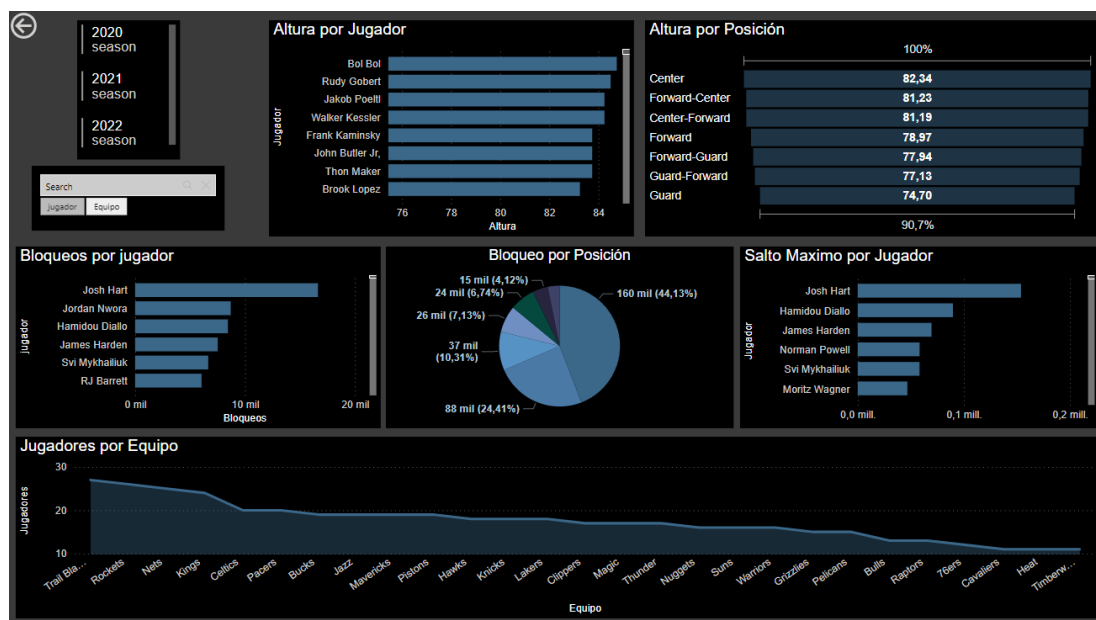


Figura 17: Análisis consolidado del desempeño de jugadores según características físicas

En este informe nos enfocamos en el análisis por jugador, y evaluamos las características físicas según la posición y el desempeño de este. Los jugadores más altos del análisis son Bol Bol y Rudy Gobert, con una altura de 84,75 y 84,5 pulgadas respectivamente (aproximadamente 215 cm) siendo la altura más baja 70 pulgadas (178 cm), una diferencia de 37 cm entre los jugadores más altos y los más bajos.

Otro aspecto analizado es la altura por posición, siendo la posición con el mayor promedio de altura el Center, y el promedio más bajo el Guard, con el Forward y sus variaciones en el punto promedio de la altura.

Ahora analizando los bloqueos por jugador, vemos que Josh Hart es el jugador con más bloqueos, de 75,75 pulgadas (192 cm), lo que permite observar que no necesariamente el de mayor altura es el que tiene más bloqueos efectivos.

Por otro lado, otro de los aspectos a considerar en cuanto a bloqueos es la posición, ya que según el lugar donde se desempeñen, pueden tener mayor oportunidad de bloquear un lanzamiento o no. Podemos observar que la mayor cantidad de bloqueos (44,13%) son desempeñados por jugadores en la posición “Guard”, y si recordamos análisis anteriores, esta posición cuenta con el promedio de altura más bajo, siendo concluyente la afirmación de que la altura no influye directamente en la cantidad de bloqueos.

Ahora bien, si analizamos la característica física “Salto máximo por jugador”, la historia es diferente, podemos ver que Josh Hart, el jugador con mayor cantidad de bloqueos, también es el jugador con mayor Salto de la NBA. No solo eso, también se puede observar que de los seis (6) jugadores con mayor cantidad de bloqueos, cuatro (4), están entre los 6 jugadores con mayor salto máximo, definiendo así que, si bien la altura no es una variable determinante en la cantidad de bloqueos, el salto máximo si lo es.

Por último, se analiza la cantidad de jugadores por equipo, lo cual permite observar los equipos con mayor cantidad de jugadores en el análisis, observando así los equipos con mayor rotación de jugadores.

Visualización con Python

Tablero 2

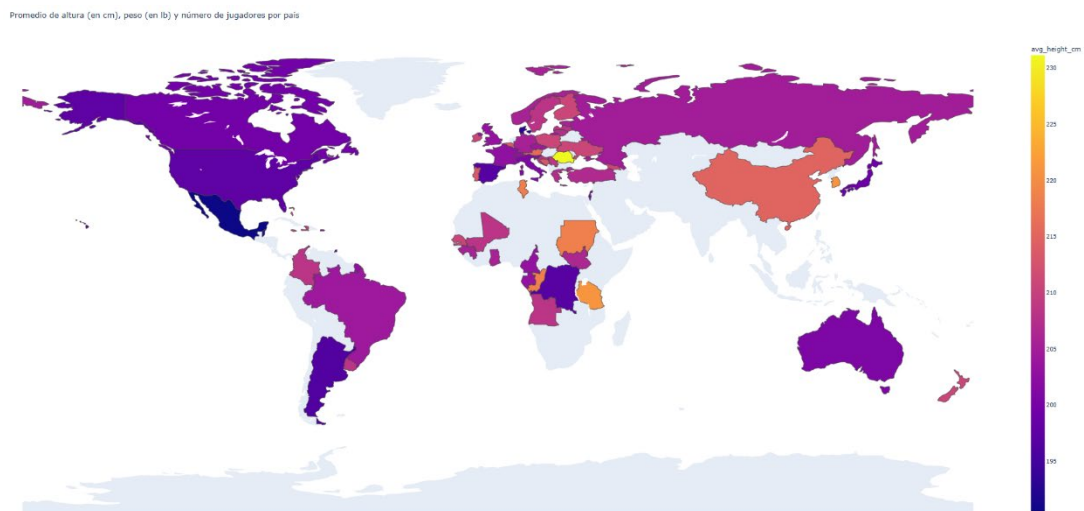


Figura 18: Gráfico de distribución de altura promedio por país

Número de jugadores por país

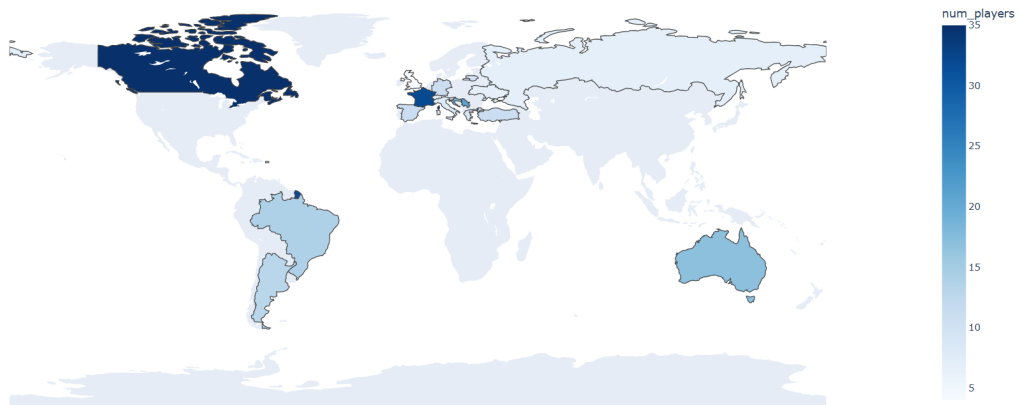


Figura 19: Mapa unificado (número de jugadores, altura promedio, peso promedio)

Oportunidades de mejora

Modelos de Machine Learning o Redes Neuronales

Modelo ARIMA – SARIMAX

Al tomar en cuenta la oportunidad de trabajar con los datos obtenidos se considera como oportunidad de mejora la posibilidad de generar un modelo tipo SARIMAX o ARIMA que, con el uso de datos históricos y el entrenamiento de un modelo autorregresivo, permita concluir cual es el posible ganador o resultado de un partido.

Datos que se podrían utilizar:

- Resultados de partidos anteriores
- Estadísticas de los equipos
- Estadísticas de los jugadores
- Información de calendario
- Datos de apuestas Históricas

Luego de la identificación de parámetros de modelo se deberá establecer un porcentaje de datos de entrenamiento y una de prueba para empezar a calcular la efectividad del modelo entrenado.

Webscraping

Mantener la información actualizada es fundamental para el uso de los tableros y los modelos asociados, es por eso por lo que realizar modelos de obtención de datos puede ser importante, una vez identificado el “proveedor” de la información se puede empezar a gestionar metodologías que permitan crear un ETL enfocado a la obtención de información.

Para esta clase de información se podrá generar procesos de webscraping, que permita extraer información de los equipos y partidos a través de páginas como ESPN o la misma NBA.

Una vez se obtenga la información, se organizará mediante librerías como pandas para generar un dataframe utilizable, una vez generado esto podremos realizar una limpieza de datos para identificar caracteres que generen problemas de compatibilidad con los datos ya existentes en el modelo.

Una vez se realiza la limpieza de datos, se debe realizar la carga de la nueva información a la base de datos ya existente, para mantener la información actualizada.

Automatización o Prefect

Utilizar procesos de automatización o flujos, permitirá ejecutar el webscraping y el modelo estadístico, trabajen de forma repetitiva y autónoma, con esto se podría mantener actualizado nuestros tableros para toma de decisiones.

Conclusiones

Las variables físicas como altura, envergadura, salto máximo entre otras no influyen directamente en los resultados del equipo, esto se debe a que directamente la diferencia física entre los jugadores de la NBA no es tan representativa como para marcar una diferencia clave.

Las variables físicas no son los únicos aspectos por considerar, se deberían analizar variables como técnica, táctica, estrategia y talento que en conjunto influyen directamente en el desempeño y resultados.

Relacionando la variable física de altura con el número de bloqueos, se puede concluir que no están directamente relacionadas. Se analizó la cantidad de bloqueos con respecto a la altura, observando que no fue concluyente, el jugador Josh Hart es el jugador con más bloqueos con diferencia y está a 23 cm del jugador más alto. Por otro lado, el 44,13% de los bloqueos de la NBA, son realizados por jugadores de la posición “Guard”, siendo los jugadores de esta posición los más bajos en promedio con una altura de 74,7 pulgadas (190 cm).

Ahora bien, las variables físicas si pueden influir directamente en el desempeño de los jugadores en algunos aspectos del juego, por ejemplo, el salto máximo es clave en el número de bloqueos de un jugador, en el análisis se puede observar que, de los 6 jugadores con mayor cantidad de bloqueos, 4 están entre los jugadores con mayor salto de la NBA.

Referencias bibliográficas

Academia Android et al. (2014) Sqlite: Introducción y herramientas de administración, Academia Android. Available at: <https://academiaandroid.com/sqlite-introduccion-herramientas-administracion/> (Accessed: 03 June 2024).

Alberca, A.S. (2020) La Librería matplotlib, Aprende con Alf. Available at: <https://aprendeconalf.es/docencia/python/manual/matplotlib/> (Accessed: 03 June 2024).

Alberca, A.S. (2022) La librería numpy, Aprende con Alf. Available at: <https://aprendeconalf.es/docencia/python/manual/numpy/> (Accessed: 03 June 2024).

Alina.caravaca@axarnet.es (2023) Qué es un archivo .json y para qué sirve **【guía completa】** , Hostinet. Available at: <https://www.hostinet.com/formacion/disenio-web/json/> (Accessed: 03 June 2024).

Chandra, N. (no date) Ipykernel: Explicación del kernel De python para jupyter notebooks, EcoAGI. Available at: <https://ecoagi.ai/es/topics/Python/ipykernel> (Accessed: 03 June 2024).

Davidiseminger (no date) ¿Qué es power bi? - power bi, Power BI | Microsoft Learn. Available at: <https://learn.microsoft.com/es-es/power-bi/fundamentals/power-bi-overview> (Accessed: 03 June 2024).

digital, N. marketing (2024) Guía completa sobre el funcionamiento de una transacción en mysql, Nativos Digitales. Available at: <https://ndmarketingdigital.com/como-funciona-una-transaccion-en-mysql/> (Accessed: 03 June 2024).

Gráficos interactivos con r: Plotly. (No date) Available at: https://rstudio-pubs-static.s3.amazonaws.com/914075_c5ee53c38d93421895868f435dfde789.html (Accessed: 03 June 2024).

Greyrat, R. (no date) Módulo Stringio en python, Barcelona Geeks. Available at: <https://barcelonageeks.com/modulo-stringio-en-python/> (Accessed: 03 June 2024).

IDE and code editor for software developers and teams (2024) Visual Studio. Available at: <https://visualstudio.microsoft.com/> (Accessed: 03 June 2024).

Id, N. (2022) Descubre Todo El funcionamiento de la NBA, La Mejor Liga de Baloncesto del Mundo. ¿Cuántos partidos juegan? ¿Cuántas conferencias hay? ¿Cómo funcionan los playoffs?, NBA ID - Noticias: Qué es la NBA y cómo funciona. Available at: <https://spain.id.nba.com/noticias/nba-que-es-como-funciona> (Accessed: 24 May 2024).

j2logo (2022) SQLAlchemy. tutorial de python sqlalchemy. Guía de inicio, J2LOGO. Available at: <https://j2logo.com/python/sqlalchemy-tutorial-de-python-sqlalchemy-guia-de-inicio/> (Accessed: 03 June 2024).

Juanweb (2023) Conectando python a bases de Datos con psycopg2, CodigosPython. Available at: <https://codigospython.com/conectando-python-a-bases-de-datos-con-psycopg2/> (Accessed: 03 June 2024).

Machine learning: ¿Qué es el aprendizaje automático y cómo funciona? (no date) Algotive. Available at: <https://www.algotive.ai/es-mx/blog/machine-learning-que-es-el-aprendizaje-autom%C3%A1tico-y-c%C3%B3mo-funciona> (Accessed: 03 June 2024).

Método join() en python (2023a) The Data Schools. Available at: <https://thdataschools.com/python/strings/join-metodo-string/> (Accessed: 03 June 2024).

Miniconda# (no date) Miniconda - Anaconda documentation. Available at: <https://docs.anaconda.com/free/miniconda/index.html> (Accessed: 03 June 2024).

Moraguez, por E.R. (2023) ¿Qué es pandas (librería de Python): Cómo Funciona y para qué sirve?, LovTechnology. Available at: <https://lovtechnology.com/que-es-pandas-libreria-de-python-como-funciona-y-para-que-sirve/> (Accessed: 03 June 2024).

¿Qué es el procesamiento de lenguaje natural? (No date) Available at: <https://aws.amazon.com/what-is/nlp/> (Accessed: 03 June 2024).

¿Qué es python y para qué se USA? Guía para Principiantes (no date) Coursera. Available at: <https://www.coursera.org/mx/articles/what-is-python-used-for-a-beginners-guide-to-using-python> (Accessed: 03 June 2024).

Ramírez, L. (2023) Algoritmo K-means: ¿Qué es y cómo funciona?, Thinking for Innovation. Available at: <https://www.iebschool.com/blog/algoritmo-k-means-que-es-y-como-funciona-big-data/> (Accessed: 03 June 2024).

team, I. editorial (2023) Jupyter Notebook, IONOS Digital Guide. Available at: <https://www.ionos.com/digitalguide/websites/web-development/jupyter-notebook/> (Accessed: 03 June 2024).

Walsh, W. (2023) NBA database, Kaggle. Available at: <https://www.kaggle.com/datasets/wyattowalsh/basketball/data> (Accessed: 25 May 2024).

What is Docker? (no date) Red Hat - We make open source technologies for the enterprise. Available at: <https://www.redhat.com/en/topics/containers/what-is-docker> (Accessed: 03 June 2024).

What is postgresql? (2021) IBM. Available at: <https://www.ibm.com/topics/postgresql> (Accessed: 03 June 2024).

What is a query - definition, meaning and examples (2023) Arimetrics. Available at: <https://www.arimetrics.com/en/digital-glossary/query> (Accessed: 03 June 2024).

Rodolfo Meza. (2022). Guia de Clase Herramientas Big Data. conceptos generales. https://d6jdw2l918v1s.cloudfront.net/modulo000/000.0_estructura_del_curso.html (accessed: 03 June 2024)

***Nota:** Cabe resaltar que se utiliza chat GPT-4.0 para consultas rápidas de uso y corrección de librerías en Python, la referenciación con link de información se limita por las referencias de chat GPT-4.0 corresponde a repositorios inexistentes y no se cita a sí mismo.

Anexo 1

