EXTRACCIÓN DE CONOCIMIENTOS DE BASES DE DATOS
ANÁLISIS DE DATOS. - JUEGOS DE PLAYOFF NBA

ARANDA GONZÁLEZ LUCÍA DEL CARMEN
9ADGS

DOMINGO 16 DE JUNIO DE 2024

CONTEXTO DEL ANÁLISIS

La Asociación Nacional de Baloncesto (NBA) es la liga profesional de baloncesto que cuenta con más importancia en el mundo. La Asociación de Baloncesto de América (BAA) fue fundada en 1946. En 1949, se fusionó con la NBL (National Basketball League). La Liga Nacional de Baloncesto (NBA) está compuesta por treinta equipos, divididos en dos conferencias: la Conferencia Este y la Conferencia Oeste. Algunos de los jugadores y equipos más icónicos del deporte han sido parte de la historia de la NBA, que tiene más de 70 años, uno de ellos Michael Jordan, gran contribuidor.

Cada temporada, los equipos compiten en una serie de juegos, que se dividen en una temporada regular y playoffs. Los juegos de playoffs son especialmente cruciales porque determinan quién pasa a las siguientes rondas y finalmente gana el campeonato de la NBA. Este análisis utiliza un conjunto de datos que contiene datos detallados de los playoffs de la NBA. Estos datos incluyen métricas como puntos anotados, tiros intentados y encestados, rebotes, asistencias y otras que son relevantes para cada equipo y jugador involucrado.

Para comprender mejor el desempeño de los equipos y los jugadores en situaciones de alta presión, como los juegos de playoffs, es necesario analizar estos datos. La importancia de analizar estos datos radica en la capacidad de identificar patrones, tendencias y factores clave que influyen en el éxito o fracaso de un equipo en los playoffs. Los equipos y entrenadores pueden tomar decisiones más informadas sobre estrategias de juego, rotaciones de jugadores y preparación física y mental al comprender estas dinámicas.

El análisis de datos de los playoffs también funciona como ayuda a los aficionados, analistas y medios de comunicación a comprender mejor el juego y hacer predicciones más precisas sobre los resultados futuros. Los resultados de este análisis podrían llevar a cambios en las estrategias del juego, la alineación de los jugadores y los métodos para contrarrestar las fortalezas de los oponentes. También podrían identificar áreas de mejora en el entrenamiento y la preparación de los jugadores para las próximas temporadas de playoffs.

HIPÓTESIS

Los equipos tienden a tener un porcentaje de tiros de campo encestados del 45 al 50% en comparación con la temporada regular en los juegos de playoffs de la NBA. Dado que los juegos de playoffs son más intensos y los equipos se concentran más en tiros de mayor valor cercanos al aro, se espera que los equipos tengan un promedio de intentos de tiros triples ligeramente más bajo, aproximadamente entre el 25 y el 30% del total de tiros intentados.

OBJETIVO

Analizar el dataframe NBADATOS orientado al tipo de juego Playoff, en la herramienta web JupyterLite, con las librerías de python pandas, matplotlib.pylab, y numpy, para generar resultados estadísticos mostrando así tendencias y relaciones entre datos.

DESARROLLO DEL ANÁLISIS

Esta sentencia realiza la función de importar librerías de cálculos y estadística, la ruta del archivo que se explora, su lectura, y finalmente su impresión/consulta de visualización.

Código:

```
import pandas as pd
import matplotlib.pylab as plt
import numpy as np
path='NBADATOS.csv'
df=pd.read_csv(path)
df
```

Resultado:

	TEAM_ABBREVIATION	TEAM_NAME	GAME_DATE	MATCHUP	WL	MIN	PTS	FGM	FGA	FG_PCT	...	AST	STL	BLK	TOV	PF	PLUS_MINUS	Game Type	Season	GAME_NUM	Round
0	DAL	Dallas Mavericks	2011-06-12	DAL @ MIA	W	240	105	41	82	0.500	...	19	11	1	14	24	10.0	Playoff	2010-11	6	NBA Finals
1	DAL	Dallas Mavericks	2011-06-09	DAL vs. MIA	W	241	112	39	69	0.565	...	23	8	3	11	20	9.0	Playoff	2010-11	5	NBA Finals
2	DAL	Dallas Mavericks	2011-06-07	DAL vs. MIA	W	240	86	29	73	0.397	...	13	7	2	11	18	3.0	Playoff	2010-11	4	NBA Finals
3	MIA	Miami Heat	2011-06-05	MIA @ DAL	W	239	88	34	78	0.436	...	20	8	5	10	27	2.0	Playoff	2010-11	3	NBA Finals
4	DAL	Dallas Mavericks	2011-06-02	DAL @ MIA	W	240	95	36	75	0.480	...	18	8	2	18	20	2.0	Playoff	2010-11	2	NBA Finals
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
15771	LAL	Los Angeles Lakers	2021-08-04	LAL @ SAC	W	240	84	27	67	0.403	...	11	11	5	14	13	4.0	Regular	22021	Regular	Regular
15772	MIA	Miami Heat	2021-08-04	MIA @ GSW	W	240	94	31	65	0.477	...	19	13	4	21	17	13.2	Regular	22021	Regular	Regular
15773	GSW	Golden State Warriors	2021-08-03	GSW @ SAC	W	265	89	33	76	0.434	...	18	4	6	16	20	1.0	Regular	22021	Regular	Regular
15774	MIA	Miami Heat	2021-08-03	MIA vs. LAL	W	241	80	33	71	0.465	...	18	10	4	19	21	2.0	Regular	22021	Regular	Regular
15775	MEM	Memphis Grizzlies	2021-08-03	MEM @ UTB	W	200	104	41	86	0.477	...	26	10	12	12	15	39.0	Regular	22021	Regular	Regular

La siguiente línea de código, cumple con la función de filtrar el dataframe por su mascara booleana para que solo contenga las filas donde el valor de la columna ‘Game Type’ sea ‘Playoff’, es decir, que toda la información en adelante esté basada en ello.

Código:

```
df = df[df['Game Type'] == 'Playoff']
```

Esta sentencia realiza la función de mostrar el tipo de dato por cada campo de la tabla/DataFrame.

Código:

```
df.dtypes
```

Resultado:

```
TEAM_ABBREVIATION     object
TEAM_NAME             object
GAME_DATE             object
MATCHUP               object
WL                    object
MIN                    int64
PTS                    int64
FGM                    int64
FGA                    int64
FG_PCT               float64
FG3M                   int64
FG3A                   int64
FG3_PCT              float64
FTM                    int64
FTA                    int64
FT_PCT               float64
OREB                   int64
DREB                   int64
REB                    int64
AST                    int64
STL                    int64
BLK                    int64
TOV                    int64
PF                     int64
PLUS_MINUS           float64
Game Type             object
Season                object
GAME_NUM              object
Round                 object
dtype: object
```

Esta sentencia realiza la función de sumar los registros nulos, o bien faltantes, considerando todos los registros almacenados en la tabla/DataFrame, por campos.

Código:

```
df.isnull().sum()
```

Resultado:

```
TEAM_ABBREVIATION    0
TEAM_NAME            0
GAME_DATE            0
MATCHUP              0
WL                   0
MIN                  0
PTS                  0
FGM                  0
FGA                  0
FG_PCT               0
FG3M                 0
FG3A                 0
FG3_PCT              0
FTM                  0
FTA                  0
FT_PCT               0
OREB                 0
DREB                 0
REB                  0
AST                  0
STL                  0
BLK                  0
TOV                  0
PF                   0
PLUS_MINUS           0
Game Type            0
Season               0
GAME_NUM             0
Round                0
dtype: int64
```

Esta sentencia realiza la función de seleccionar de la tabla/DataFrame ciertos campos agrupándolos en un arreglo e imprimiendo los registros de cada campo.

Código:

```
df=df[['GAME_DATE','TEAM_NAME','PTS','FGM','FGA','FG3M','FG3A']]
df
```

Resultado:

	GAME_DATE	TEAM_NAME	PTS	FGM	FGA	FG3M	FG3A
0	2011-06-12	Dallas Mavericks	105	41	82	11	26
1	2011-06-09	Dallas Mavericks	112	39	69	13	19
2	2011-06-07	Dallas Mavericks	86	29	73	4	19
3	2011-06-05	Miami Heat	88	34	78	8	19
4	2011-06-02	Dallas Mavericks	95	36	75	6	17
...	...	...	...	...	...	...	...
14555	2021-05-23	Phoenix Suns	99	40	86	9	28
14556	2021-05-22	Dallas Mavericks	113	38	76	17	36
14557	2021-05-22	Portland Trail Blazers	123	43	91	19	40
14558	2021-05-22	Brooklyn Nets	104	35	84	8	34
14559	2021-05-22	Milwaukee Bucks	109	42	96	5	31

Esta sentencia realiza la función de crear dos nuevos campos (‘Total Tiros’ y ‘Total Tiros Encestados’), en los cuales, el contenido será la suma de los registros de los campos ‘FGA’ y ‘F3GA’ para ‘Total Tiros’, y la suma de ‘FGM’ y ‘FG3M’ para ‘Total Tiros Encestados’, finalmente se toma una muestra con 10 líneas de registros cada campo, incluidos los dos nuevos.

Código:

```
df['Total Tiros']=df['FGA']+df['FG3A']
df['Total Tiros Encestados']=df['FGM']+df['FG3M']
df.sample(10)
```

Resultado:

	GAME_DATE	TEAM_NAME	PTS	FGM	FGA	FG3M	FG3A	Total Tiros	Total Tiros Encestados
8663	2017-06-04	Golden State Warriors	132	46	89	18	43	132	64
4173	2014-04-28	Atlanta Hawks	107	33	66	15	27	93	48
5660	2015-05-08	Chicago Bulls	99	34	90	6	21	111	40
7216	2016-04-21	Oklahoma City Thunder	131	45	78	15	27	105	60
2691	2013-04-20	Denver Nuggets	97	38	85	3	16	101	41
4158	2014-05-03	Oklahoma City Thunder	120	42	69	11	19	88	53
2675	2013-04-25	Chicago Bulls	79	31	78	3	15	93	34
7190	2016-04-30	San Antonio Spurs	124	51	84	9	15	99	60
1547	2012-05-11	Memphis Grizzlies	90	35	77	3	10	87	38
2614	2013-06-09	Miami Heat	103	41	83	10	19	102	51

Esta sentencia realiza la función de reemplazar el nombre de los campos ‘GAME_DATE’ y ‘TEAM_NAME’ por ‘Fecha’ y ‘Equipo’, posteriormente, se eliminan las dos columnas primeramente mencionadas del DataFrame df, se indica una operación de columna

(axis=1), e indicando que los cambios se realizan en el DataFrame original. La última línea restablece el índice del DataFrame df a su valor por defecto (es decir, una secuencia de enteros de 0 a N-1, donde N es la longitud del DataFrame).

Código:

```
df['Fecha']=df['GAME_DATE']
df['Equipo']=df['TEAM_NAME']
df.drop('GAME_DATE', axis=1, inplace=True)
df.drop('TEAM_NAME',axis=1,inplace=True)
df.reset_index()
```

Resultado:

	index	PTS	FGM	FGA	FG3M	FG3A	Total Tiros	Total Tiros Encestados	Fecha	Equipo
0	0	105	41	82	11	26	108	52	2011-06-12	Dallas Mavericks
1	1	112	39	69	13	19	88	52	2011-06-09	Dallas Mavericks
2	2	86	29	73	4	19	92	33	2011-06-07	Dallas Mavericks
3	3	88	34	78	8	19	97	42	2011-06-05	Miami Heat
4	4	95	36	75	6	17	92	42	2011-06-02	Dallas Mavericks
...	...	...	...	...	...	...	...	...	...	...
912	14555	99	40	86	9	28	114	49	2021-05-23	Phoenix Suns
913	14556	113	38	76	17	36	112	55	2021-05-22	Dallas Mavericks
914	14557	123	43	91	19	40	131	62	2021-05-22	Portland Trail Blazers
915	14558	104	35	84	8	34	118	43	2021-05-22	Brooklyn Nets
916	14559	109	42	96	5	31	127	47	2021-05-22	Milwaukee Bucks

Esta sentencia realiza la función de mostrar las columnas del DataDrame.

Código:

```
df.columns
```

Resultado:

```
Index(['PTS', 'FGM', 'FGA', 'FG3M', 'FG3A', 'Total Tiros',
       'Total Tiros Encestados', 'Fecha', 'Equipo'],
      dtype='object')
```

Se realiza la función de acomodar columnas del DataFrame con sus contenidos, organizados en un orden especifico declarado por un arreglo. Después las muestra.

Código:

```
df=df[['Equipo', 'Fecha', 'PTS', 'FGM', 'FGA', 'FG3M', 'FG3A', 'Total Tiros',
       'Total Tiros Encestados' ]]
df
```

Resultado:

	Equipo	Fecha	PTS	FGM	FGA	FG3M	FG3A	Total Tiros	Total Tiros Encestados
0	Dallas Mavericks	2011-06-12	105	41	82	11	26	108	52
1	Dallas Mavericks	2011-06-09	112	39	69	13	19	88	52
2	Dallas Mavericks	2011-06-07	86	29	73	4	19	92	33
3	Miami Heat	2011-06-05	88	34	78	8	19	97	42
4	Dallas Mavericks	2011-06-02	95	36	75	6	17	92	42
...	...	...	...	...	...	...	...	...	...
14555	Phoenix Suns	2021-05-23	99	40	86	9	28	114	49
14556	Dallas Mavericks	2021-05-22	113	38	76	17	36	112	55
14557	Portland Trail Blazers	2021-05-22	123	43	91	19	40	131	62
14558	Brooklyn Nets	2021-05-22	104	35	84	8	34	118	43
14559	Milwaukee Bucks	2021-05-22	109	42	96	5	31	127	47

Esta sentencia realiza la función de ordenar los valores del DataFrame por fecha y de manera ascendiente, después se divide el DataFrame en dos partes, cada una con un rango de fecha especificado, finalmente por separado, se imprimen los resultados de la parte correspondiente del DataFrame.

Código:

```
df.sort_values('Fecha', ascending=True)
df1=df[df['Fecha']<'2016-01-01']
df2=df[df['Fecha']>='2016-01-01']
df1
df2
```

Resultado:

	Equipo	Fecha	PTS	FGM	FGA	FG3M	FG3A	Total Tiros	Total Tiros Encestados
7147	Cleveland Cavaliers	2016-06-19	93	33	82	6	25	107	39
7148	Cleveland Cavaliers	2016-06-16	115	40	77	10	27	104	50
7149	Cleveland Cavaliers	2016-06-13	112	44	83	10	24	107	54
7150	Golden State Warriors	2016-06-10	108	33	81	17	36	117	50
7151	Cleveland Cavaliers	2016-06-08	120	48	91	12	25	116	60
...	...	...	...	...	...	...	...	...	...
14555	Phoenix Suns	2021-05-23	99	40	86	9	28	114	49
14556	Dallas Mavericks	2021-05-22	113	38	76	17	36	112	55
14557	Portland Trail Blazers	2021-05-22	123	43	91	19	40	131	62
14558	Brooklyn Nets	2021-05-22	104	35	84	8	34	118	43
14559	Milwaukee Bucks	2021-05-22	109	42	96	5	31	127	47

Estas líneas de código están generando estadísticas descriptivas para el campo ‘FGA’ en dos DataFrames diferentes, df1 y df2, y almacenando los resultados en TirosDosP1 y TirosDosP2 respectivamente.

Código:

```
TirosDosP1=df1['FGA'].describe(include='all')
TirosDosP2=df2['FGA'].describe(include='all')
```

En la primera línea se convierte el objeto ‘TirosDosP1’ en un dataframe pandas, mientras que la segunda línea crea una columna nueva ‘FGA_P1’ de ‘TirosDosP1’, donde contendrá los mismos valores que la columna ‘FGA’ en el dataframe.

Código:

```
TirosDosP1=TirosDosP1.to_frame()
TirosDosP1['FGA_P1']=TirosDosP1['FGA']
TirosDosP1
```

Resultado:

	FGA	FGA_P1
count	420.000000	420.000000
mean	79.466667	79.466667
std	7.177686	7.177686
min	64.000000	64.000000
25%	75.000000	75.000000
50%	79.000000	79.000000
75%	83.000000	83.000000
max	109.000000	109.000000

La primera línea elimina la columna llamada 'FGA' del DataFrame TirosDosP1, con axis=1, Indica que estamos eliminando una columna (el valor 1 se refiere al eje de las columnas), e inplace=True significa que la operación se realizará directamente en el DataFrame TirosDosP1 sin crear un nuevo DataFrame. Finalmente se imprime el resultado.

Código:

```
TirosDosP1.drop('FGA', axis=1, inplace=True)
TirosDosP1
```

Resultado:

	FGA_P1
count	420.000000
mean	79.466667
std	7.177686
min	64.000000
25%	75.000000
50%	79.000000
75%	83.000000
max	109.000000

Primeramente, se crea un nuevo DataFrame llamado TirosDosP2 a partir de una columna existente. El método .to_frame() convierte la columna en un DataFrame, la siguiente línea renombra la columna FGA a FGA_P2 en el DataFrame TirosDosP2 e imprime los resultados, después elimina la columna original FGA del DataFrame TirosDosP2, y nuevamente se muestran los resultados.

Código:

```
TirosDosP2=TirosDosP2.to_frame()
TirosDosP2['FGA_P2']=TirosDosP2['FGA']
TirosDosP2
TirosDosP2.drop('FGA', axis=1, inplace=True)
TirosDosP2
```

Resultado:

	FGA_P2
count	497.000000
mean	85.116700
std	6.901851
min	61.000000
25%	81.000000
50%	85.000000
75%	89.000000
max	124.000000

Se crea el DataFrame vacío ‘TirosDos’, con las columnas ‘FGA_P1’ y ‘FGA_P2’, asignándole valores desde los DataFrames ‘TirosDosP1’ y ‘TirosDosP2’ respectivamente, y en la línea final se redondean los valores utilizando el método .round().

Código:

```
TirosDos=pd.DataFrame(columns=['FGA_P1', 'FGA_P2'])
TirosDos
TirosDos['FGA_P1']=TirosDosP1['FGA_P1']
TirosDos['FGA_P2']=TirosDosP2['FGA_P2']
TirosDos.round()
```

Resultado:

	FGA_P1	FGA_P2
count	420.0	497.0
mean	79.0	85.0
std	7.0	7.0
min	64.0	61.0
25%	75.0	81.0
50%	79.0	85.0
75%	83.0	89.0
max	109.0	124.0

Se aplica tanto a 'TirosTresP1' como a 'TirosTresP2' el método describe() para obtener la información como cantidades de valores de la media, no nulos, desviación estándar, valores mínimo y máximo además de percentiles, incluyendo columnas no numéricas.

Código:

```
TirosTresP1=df1['FG3A'].describe(include='all')
TirosTresP2=df2['FG3A'].describe(include='all')
```

La serie 'TirosTresP1' se convierte en un DataFrame transformando una estructura unidimensional a bidimensional, se crea la columna 'FG3A_P1' con los mismos valores de la columna 'FG3A'.

Código:

```
TirosTresP1=TirosTresP1.to_frame()
TirosTresP1['FG3A_P1']=TirosTresP1['FG3A']
TirosTresP1
```

Resultado:

	FG3A	FG3A_P1
count	420.000000	420.000000
mean	20.757143	20.757143
std	6.260310	6.260310
min	6.000000	6.000000
25%	16.000000	16.000000
50%	20.000000	20.000000
75%	25.000000	25.000000
max	41.000000	41.000000

Primeto elimina la columna llamada 'FG3A' del DataFrame TirosTresP1, con axis=1, Indica que estamos eliminando una columna (el valor 1 se refiere al eje de las columnas), e inplace=True significa que la operación se realizará directamente en el DataFrame TirosTresP1 sin crear un nuevo DataFrame. Finalmente se imprime el resultado.

Código:

```
TirosTresP1.drop('FG3A', axis=1, inplace=True)
TirosTresP1
```

Resultado:

	FG3A_P1
count	420.000000
mean	20.757143
std	6.260310
min	6.000000
25%	16.000000
50%	20.000000
75%	25.000000
max	41.000000

Se crea un nuevo DataFrame llamado TirosTresP2 a partir de una columna existente. Con el método .to_frame() se convierte la columna en un DataFrame, la siguiente línea renombra la columna FG3A a FG3A_P2 en el DataFrame TirosTresP2 e imprime los resultados, después elimina la columna original FG3A del DataFrame TirosTresP2, y nuevamente se muestran los resultados.

Código:

```
TirosTresP2=TirosTresP2.to_frame()
TirosTresP2['FG3A_P2']=TirosTresP2['FG3A']
TirosTresP2
TirosTresP2.drop('FG3A', axis=1, inplace=True)
TirosTresP2
```

Resultado:

	FG3A_P2
count	497.000000
mean	31.619718
std	7.758337
min	11.000000
25%	26.000000
50%	31.000000
75%	37.000000
max	56.000000

Se crea el DataFrame vacío 'TirosTres', con las columnas 'FG3A_P1' y 'FG3A_P2', asignándole valores desde los DataFrames 'TirosTresP1' y 'TirosTresP2' respectivamente, y en la línea final se redondean los valores utilizando el método .round().

Código:

```
TirosTres=pd.DataFrame(columns=['FG3A_P1', 'FG3A_P2'])
TirosTres
TirosTres['FG3A_P1']=TirosTresP1['FG3A_P1']
TirosTres['FG3A_P2']=TirosTresP2['FG3A_P2']
TirosTres.round()
```

Resultado:

	FG3A_P1	FG3A_P2
count	420.0	497.0
mean	21.0	32.0
std	6.0	8.0
min	6.0	11.0
25%	16.0	26.0
50%	20.0	31.0
75%	25.0	37.0
max	41.0	56.0

Se unen las dos tablas TirosTres y TirosDos en una sola tabla resultado usando la función pd.concat(). El parámetro axis=1 indica que la unión se realiza a lo largo del eje de las columnas (es decir, las tablas se unen horizontalmente).

Código:

```
resultado = pd.concat([TirosTres, TirosDos], axis=1)
resultado
```

Resultado:

	FG3A_P1	FG3A_P2	FGA_P1	FGA_P2
count	420.000000	497.000000	420.000000	497.000000
mean	20.757143	31.619718	79.466667	85.116700
std	6.260310	7.758337	7.177686	6.901851
min	6.000000	11.000000	64.000000	61.000000
25%	16.000000	26.000000	75.000000	81.000000
50%	20.000000	31.000000	79.000000	85.000000
75%	25.000000	37.000000	83.000000	89.000000
max	41.000000	56.000000	109.000000	124.000000

Definimos el DataFrame que queremos imprimir, mostrándonos los resultados de 'df1'.

Código:

```
df1
```

Resultado:

	Equipo	Fecha	PTS	FGM	FGA	FG3M	FG3A	Total Tiros	Total Tiros Encestados
0	Dallas Mavericks	2011-06-12	105	41	82	11	26	108	52
1	Dallas Mavericks	2011-06-09	112	39	69	13	19	88	52
2	Dallas Mavericks	2011-06-07	86	29	73	4	19	92	33
3	Miami Heat	2011-06-05	88	34	78	8	19	97	42
4	Dallas Mavericks	2011-06-02	95	36	75	6	17	92	42
...	...	...	...	...	...	...	...	...	...
5706	Atlanta Hawks	2015-04-19	99	34	79	10	30	109	44
5707	Houston Rockets	2015-04-18	118	38	85	10	25	110	48
5708	Chicago Bulls	2015-04-18	103	38	83	12	32	115	50
5709	Golden State Warriors	2015-04-18	106	37	81	11	29	110	48
5710	Washington Wizards	2015-04-18	93	39	99	6	21	120	45

Definimos el DataFrame 'df2' para imprimir, mostrándonos los resultados.

Código:

```
df2
```

Resultado:

	Equipo	Fecha	PTS	FGM	FGA	FG3M	FG3A	Total Tiros	Total Tiros Encestados
7147	Cleveland Cavaliers	2016-06-19	93	33	82	6	25	107	39
7148	Cleveland Cavaliers	2016-06-16	115	40	77	10	27	104	50
7149	Cleveland Cavaliers	2016-06-13	112	44	83	10	24	107	54
7150	Golden State Warriors	2016-06-10	108	33	81	17	36	117	50
7151	Cleveland Cavaliers	2016-06-08	120	48	91	12	25	116	60
...	...	...	...	...	...	...	...	...	...
14555	Phoenix Suns	2021-05-23	99	40	86	9	28	114	49
14556	Dallas Mavericks	2021-05-22	113	38	76	17	36	112	55
14557	Portland Trail Blazers	2021-05-22	123	43	91	19	40	131	62
14558	Brooklyn Nets	2021-05-22	104	35	84	8	34	118	43
14559	Milwaukee Bucks	2021-05-22	109	42	96	5	31	127	47

Para la construcción de la primera gráfica se implementaron las siguientes sentencias, donde se declara principalmente la información base de la gráfica y se define el DataFrame (df1, el primero) del que se extraerán los datos e imprimimos el resultado.

Código:

```
df1_circ=df1[['FGA','FG3A']]
df1_circ
```

Resultado:

	FGA	FG3A
0	82	26
1	69	19
2	73	19
3	78	19
4	75	17
...	...	...
5706	79	30
5707	85	25
5708	83	32
5709	81	29
5710	99	21

Posteriormente transponemos los valores previamente generados de manera horizontal, imprimiéndolos con la última línea.

Código

```
df1_circ=df1_circ.T
df1_circ
```

Resultado:

	0	1	2	3	4	5	6	7	8	9	...	5701	5702	5703	5704	5705	5706	5707	5708	5709	5710
FGA	82	69	73	78	75	80	66	78	75	81	...	86	81	88	82	78	79	85	83	81	99
FG3A	26	19	19	19	17	24	15	20	13	25	...	30	33	9	31	18	30	25	32	29	21

Ahora definimos “TotalTiros” de manera que nos muestre una columna nueva asignando valores de cada fila calculada, y axis suma todas las filas devolviendo una serie de “Pandas”.

Código:

```
df1_circ['TotalTiros']=df1_circ.sum(axis=1)
df1_circ
```

Resultado:

	0	1	2	3	4	5	6	7	8	9	...	5702	5703	5704	5705	5706	5707	5708	5709	5710	TotalTiros
FGA	82	69	73	78	75	80	66	78	75	81	...	81	88	82	78	79	85	83	81	99	33376
FG3A	26	19	19	19	17	24	15	20	13	25	...	33	9	31	18	30	25	32	29	21	8718

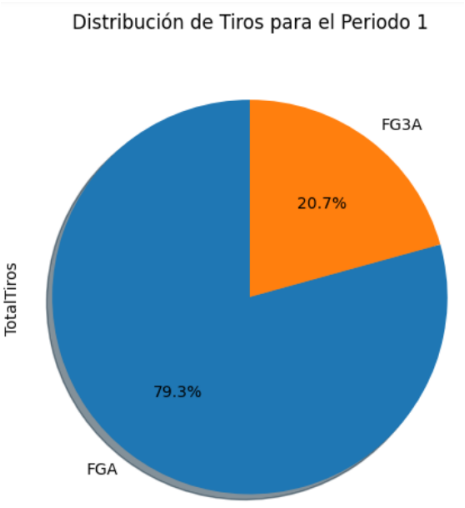
Utilizamos la biblioteca de visualización de datos de matplotlib, para crear una gráfica de la columna 'TotalTiros' del dataframe df1_circ, donde en kind se especifica el tipo de gráfico que es de tarta, con figsize el tamaño de la figura en pulgadas (ancho y alto), con autopct el porcentaje que se mostrará con un decimal, startangle especifica el ángulo de inicio de la gráfica, siendo en sentido antihorario y shadow añade una sombra, es decir, por mera estética de la gráfica. Después se define el título de la gráfica, la operación de columna y mostrar el resultado.

Código:

```
df1_circ['TotalTiros'].plot(kind='pie',
                            figsize=(5, 6),
                            autopct='%1.1f%%', # añadimos porcentajes,
                            startangle=90,     # ángulo de comienzo,
                            shadow=True,       # añadimos sombra,
                            ),
plt.title('Distribución de Tiros para el Periodo 1')
plt.axis('equal')

plt.show()
```

Resultado:



Definimos el DataFrame df2 del que se extraerán los datos haciendo uso de las columnas 'FGA' y 'FG3A' e imprimimos el resultado.

Código:

```
df2_circ=df2[['FGA','FG3A']]
df2_circ
```

Resultado:

	FGA	FG3A
7147	82	25
7148	77	27
7149	83	24
7150	81	36
7151	91	25
...	...	...
14555	86	28
14556	76	36
14557	91	40
14558	84	34
14559	96	31

Transponemos los valores previamente generados de manera horizontal, de df2_circ, imprimiéndolos con la última línea.

Código:

```
df2_circ=df2_circ.T
df2_circ
```

Resultado:

	7147	7148	7149	7150	7151	7152	7153	7154	7155	7156	...	14550	14551	14552	14553	14554	14555	14556	14557	14
FGA	82	77	83	81	91	81	87	85	88	74	...	94	86	90	93	100	86	76	91	
FG3A	25	27	24	36	25	33	27	37	45	31	...	53	28	34	32	20	28	36	40	

Ahora definimos “TotalTiros” de manera que nos muestre una columna nueva asignando valores de cada fila calculada, y axis suma todas las filas devolviendo una serie de “Pandas”, pero recordando que estamos utilizando el DataFrame 2, por lo que se declara en cada sentencia que contenga “df” igualmente.

Código:

```
df2_circ['TotalTiros']=df2_circ.sum(axis=1)
df2_circ
```

Resultado:

	7147	7148	7149	7150	7151	7152	7153	7154	7155	7156	...	14551	14552	14553	14554	14555	14556	14557	14558	14
FGA	82	77	83	81	91	81	87	85	88	74	...	86	90	93	100	86	76	91	84	
FG3A	25	27	24	36	25	33	27	37	45	31	...	28	34	32	20	28	36	40	34	

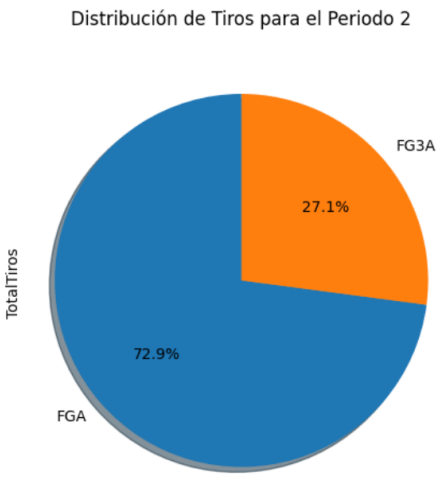
En este código también se está utilizando la biblioteca de visualización de datos matplotlib para crear un gráfico circular (o gráfico de tarta) de la columna 'TotalTiros' del DataFrame df2_circ, e igual, especificamos el grafico de paste, el tamaño de la figura, etiquetas de sección del porcentaje, el ángulo del inicio del gráfico, y una sombra para agregar estética.

Código:

```
df2_circ['TotalTiros'].plot(kind='pie',
                            figsize=(5, 6),
                            autopct='%1.1f%%', # añadimos porcentajes,
                            startangle=90,     # ángulo de comienzo,
                            shadow=True,       # añadimos sombra,
                            ),
plt.title('Distribución de Tiros para el Periodo 2')
plt.axis('equal')

plt.show()
```

Resultado:



Este código genera lo que son los dos gráficos de tarta en una sola figura para comparar la distribución (de manera visual) de tiros en dos periodos diferentes. Primero, se crea una figura con dos subgráficos (ax1 y ax2) en una fila y comparten el eje y. Luego, en cada subgráfico, se dibuja un gráfico de pastel utilizando los datos de 'TotalTiros' de los DataFrames df1_circ y df2_circ respectivamente. Los porcentajes se muestran en cada sección del gráfico de tarta. Finalmente, se establecen los títulos y se muestra la figura con "show()".

Código:

```
f, (ax1,ax2) = plt.subplots(1, 2, sharey=True, figsize =(15,3))
ax1.pie(df1_circ['TotalTiros'], autopct='%1.1f%%' )
ax1.set_title('Distribución de Tiros para el Periodo 1')
ax2.pie(df2_circ['TotalTiros'], autopct='%1.1f%%')
ax2.set_title('Distribución de Tiros para el Periodo 2')

plt.show()
```

Resultado:

Con la función df.head(), logramos mostrar las primeras 5 filas del DataFrame que tenemos definido con las siglas 'df', meramente como análisis previo.

Código:

```
df.head()
```

Resultado:

	Equipo	Fecha	PTS	FGM	FGA	FG3M	FG3A	Total Tiros	Total Tiros Encestados
0	Dallas Mavericks	2011-06-12	105	41	82	11	26	108	52
1	Dallas Mavericks	2011-06-09	112	39	69	13	19	88	52
2	Dallas Mavericks	2011-06-07	86	29	73	4	19	92	33
3	Miami Heat	2011-06-05	88	34	78	8	19	97	42
4	Dallas Mavericks	2011-06-02	95	36	75	6	17	92	42

Creamos una copia del DataFrame bajo el nombre 'df_line', es decir, donde se almacenará dicha copia, con el fin de crear una gráfica de línea, e imprimimos dicha copia.

Código:

```
df_line=df
df_line
```

Resultado:

	Equipo	Fecha	PTS	FGM	FGA	FG3M	FG3A	Total Tiros	Total Tiros Encestados
0	Dallas Mavericks	2011-06-12	105	41	82	11	26	108	52
1	Dallas Mavericks	2011-06-09	112	39	69	13	19	88	52
2	Dallas Mavericks	2011-06-07	86	29	73	4	19	92	33
3	Miami Heat	2011-06-05	88	34	78	8	19	97	42
4	Dallas Mavericks	2011-06-02	95	36	75	6	17	92	42
...	...	...	...	...	...	...	...	...	...
14555	Phoenix Suns	2021-05-23	99	40	86	9	28	114	49
14556	Dallas Mavericks	2021-05-22	113	38	76	17	36	112	55
14557	Portland Trail Blazers	2021-05-22	123	43	91	19	40	131	62
14558	Brooklyn Nets	2021-05-22	104	35	84	8	34	118	43
14559	Milwaukee Bucks	2021-05-22	109	42	96	5	31	127	47

Convertimos la columna 'fecha' del DataFrame df_line al formato de fecha correspondiente a año, mes y día.

Código:

```
df_line['Fecha'] = pd.to_datetime(df_line.Fecha, format='%Y-%m-%d')
```

Se crea la columna ‘Year’ en df_line extrayendo el año de la columna de ‘Fecha’, y la última línea imprime el resultado.

Código:

```
df_line['Year']=df_line['Fecha'].dt.year
df_line
```

Resultado:

	Equipo	Fecha	PTS	FGM	FGA	FG3M	FG3A	Total Tiros	Total Tiros Encestados	Year
0	Dallas Mavericks	2011-06-12	105	41	82	11	26	108	52	2011
1	Dallas Mavericks	2011-06-09	112	39	69	13	19	88	52	2011
2	Dallas Mavericks	2011-06-07	86	29	73	4	19	92	33	2011
3	Miami Heat	2011-06-05	88	34	78	8	19	97	42	2011
4	Dallas Mavericks	2011-06-02	95	36	75	6	17	92	42	2011
...	...	...	...	...	...	...	...	...	...	...
14555	Phoenix Suns	2021-05-23	99	40	86	9	28	114	49	2021
14556	Dallas Mavericks	2021-05-22	113	38	76	17	36	112	55	2021
14557	Portland Trail Blazers	2021-05-22	123	43	91	19	40	131	62	2021
14558	Brooklyn Nets	2021-05-22	104	35	84	8	34	118	43	2021
14559	Milwaukee Bucks	2021-05-22	109	42	96	5	31	127	47	2021

Con ‘df_line.columns’ mostramos los nombres de las columnas del DataFrame df_line, para su posterior uso en un subconjunto de columnas.

Código:

```
df_line.columns
```

Resultado:

```
Index(['Equipo', 'Fecha', 'PTS', 'FGM', 'FGA', 'FG3M', 'FG3A', 'Total Tiros',
       'Total Tiros Encestados', 'Year'],
      dtype='object')
```

Seleccionamos dicho subconjunto de columnas de df y lo almacenamos en df_line, tomando en cuenta todos los valores obtenidos en la consulta de columnas, pero excluyendo las columnas equipo y fecha.

Código:

```
df_line=df[['Year','PTS', 'FGM', 'FGA', 'FG3M', 'FG3A', 'Total Tiros',
       'Total Tiros Encestados']]
df_line
```

Resultado:

	Year	PTS	FGM	FGA	FG3M	FG3A	Total Tiros	Total Tiros Encestados
0	2011	105	41	82	11	26	108	52
1	2011	112	39	69	13	19	88	52
2	2011	86	29	73	4	19	92	33
3	2011	88	34	78	8	19	97	42
4	2011	95	36	75	6	17	92	42
...	...	...	...	...	...	...	...	...
14555	2021	99	40	86	9	28	114	49
14556	2021	113	38	76	17	36	112	55
14557	2021	123	43	91	19	40	131	62
14558	2021	104	35	84	8	34	118	43
14559	2021	109	42	96	5	31	127	47

Agrupamos df_line por la columna de 'Year' y sumamos los valores de las demás columnas para cada año.

Código:

```
df_line=df_line.groupby(['Year'], as_index=False).sum()
df_line
```

Resultado:

	Year	PTS	FGM	FGA	FG3M	FG3A	Total Tiros	Total Tiros Encestados
0	2011	8002	2854	6261	557	1457	7718	3411
1	2012	8210	3035	6546	532	1498	8044	3567
2	2013	8580	3166	6732	632	1712	8444	3798
3	2014	9317	3386	7048	740	1926	8974	4126
4	2015	8597	3107	6789	818	2125	8914	3925
5	2016	9208	3362	7179	897	2228	9407	4259
6	2017	8915	3255	6623	928	2348	8971	4183
7	2018	9092	3313	6904	970	2538	9442	4283
8	2019	9326	3353	7127	1006	2682	9809	4359
9	2020	9576	3408	7062	1160	3009	10071	4568
10	2021	9908	3602	7408	1136	2910	10318	4738

Se crea un nuevo DataFrame con el nombre de df_line_final, en donde se seleccionan solo las columnas ‘FGA’ y ‘FG3A’ de df_line, finalmente en la última línea se imprime el resultado.

Código:

```
df_line_final=df_line[['FGA','FG3A']]
df_line_final
```

Resultado:

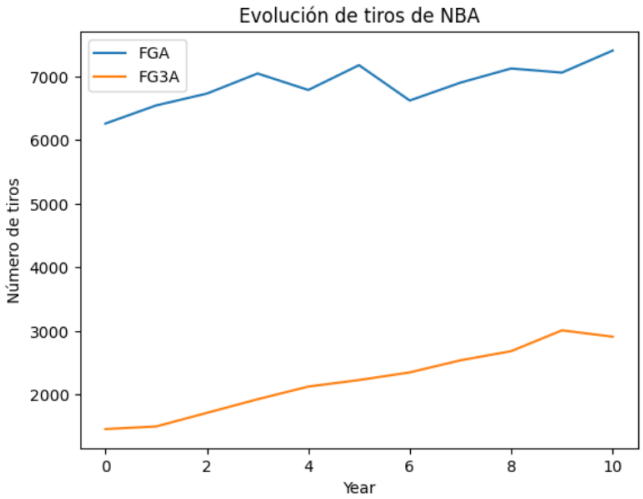
	FGA	FG3A
0	6261	1457
1	6546	1498
2	6732	1712
3	7048	1926
4	6789	2125
5	7179	2228
6	6623	2348
7	6904	2538
8	7127	2682
9	7062	3009
10	7408	2910

Creamos un gráfico de líneas con los datos obtenidos de df_line_final, establecemos el título de la gráfica (Evolución de tiros de NBA), también asignamos la etiqueta del eje 'y' como 'Número de tiros', al igual que con la etiqueta del eje 'x' pero con el nombre de 'Year' o año en español. Finalizando con la función para mostrar la gráfica 'show()'.

Código:

```
df_line_final.plot(kind='line')
plt.title('Evolución de tiros de NBA')
plt.ylabel('Número de tiros')
plt.xlabel('Year')
plt.show()
```

Resultado:



RESULTADOS GENERALES

El análisis minucioso de los datos de los juegos de playoffs de la NBA proporcionó información útil sobre cómo los equipos se desempeñaron durante estas fases cruciales de la temporada. Se descubrieron patrones y tendencias significativos analizando y visualizando métricas como tiros de campo intentados y encestados, tiros triples, rebotes y asistencias.

Los resultados muestran que la hipótesis inicial se cumple parcialmente. Si bien el porcentaje de tiros de campo encestados en los juegos de playoffs se encuentra cerca del rango esperado, el porcentaje de tiros triples intentados en comparación con el total de tiros estuvo ligeramente por encima del 20 % en algunos años analizados. Esto

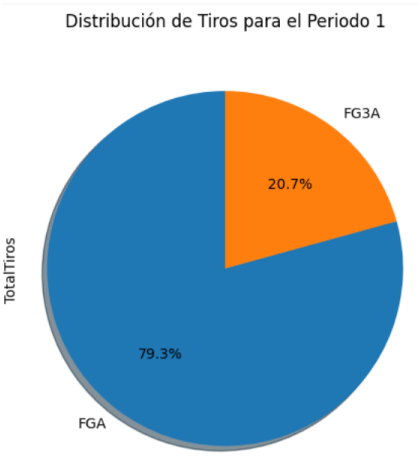
indica que, aunque los equipos suelen buscar tiros de mayor valor cerca del aro, los tiros triples todavía son importantes en los playoffs.

El objetivo principal del análisis, que consistía en examinar minuciosamente el conjunto de datos de los playoffs de la NBA utilizando herramientas como Python, pandas, matplotlib y numpy, se cumplió satisfactoriamente. Se crearon una variedad de gráficos y visualizaciones para facilitar la comprensión de los datos y ayudar a identificar patrones y tendencias importantes previamente mencionadas.

RESULTADOS GRÁFICOS

Gráfica de pastel - Distribución de tiros en el primer período (df1):

Como se muestra en esta gráfica de pastel, el 20.7% de los tiros intentados por los equipos en el primer período analizado fueron tiros de tres puntos (FG3A), mientras que el 79.3% restante fueron tiros de campo (FGA), que incluían tiros de dos puntos y tiros cercanos al aro. Según esta distribución, durante este período, los equipos se concentraron más en tiros cerca de la canasta de mayor valor, pero aún mantuvieron una gran cantidad de tiros de tres puntos en su estrategia ofensiva.

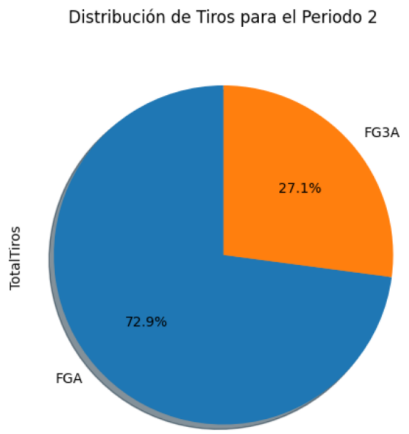


Comprender la estrategia ofensiva predominante de los equipos durante este período específico es donde radica la importancia de esta información. Un mayor porcentaje de tiros de campo cerca del aro puede indicar un enfoque en obtener puntos de alta eficiencia y un juego más físico cerca de la canasta. Sin embargo, la gran cantidad de

tiros de tres puntos demuestra la importancia de mantener una amenaza desde el perímetro y aprovechar las oportunidades de tiro desde larga distancia.

Gráfica de pastel - Distribución de tiros en el segundo período (df2):

La gráfica de pastel indica que el 27.1% de los tiros intentados en el segundo período analizado fueron tiros de tres puntos (FG3A), mientras que el 72.9% restante fueron tiros de campo (FGA). El porcentaje de tiros de tres puntos intentados ha aumentado ligeramente en comparación con el primer período.



Esta información es relevante porque muestra la posibilidad de que los equipos cambien su estrategia ofensiva durante este tiempo. Un mayor porcentaje de tiros de tres puntos puede indicar una mayor dependencia del tiro de larga distancia y una mayor confianza en el juego exterior. Además, puede reflejar cambios tácticos para contrarrestar las defensas de los oponentes o aprovechar las fortalezas de los tiradores de los equipos.

Gráficas de pastel comparativas:

Al comparar las dos gráficas de pastel, se puede ver que en el primer período había un 20.7% de FG3A y un 79.3% de FGA, mientras que, en el segundo período, el porcentaje de FG3A aumentó a un 27.1% y el de FGA disminuyó a un 72.9%.

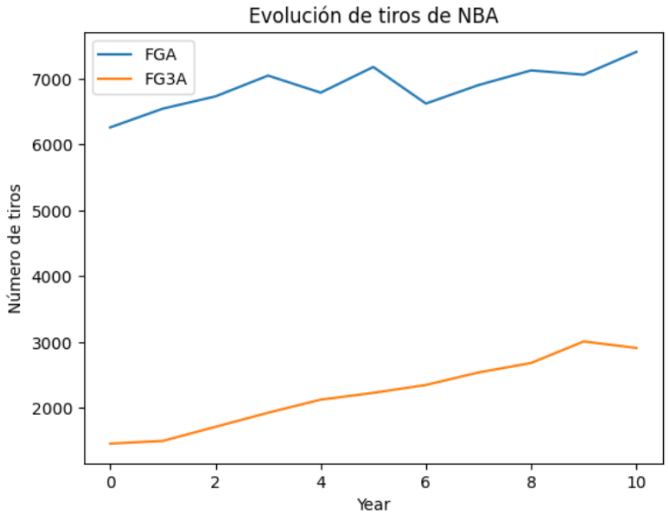
Como resultado de esta comparación directa, se puede observar una tendencia a utilizar más tiros de tres puntos en el segundo período en comparación con el primero.

Este cambio puede ser el resultado de cambios estratégicos, cambios en el personal del equipo o ajustes a las defensas de los oponentes.

La comparación es importante para entender cómo cambian las estrategias ofensivas de los equipos a lo largo del juego. Los entrenadores y analistas pueden usar esta información para evaluar patrones y tendencias en el comportamiento ofensivo de los equipos y ajustar en consecuencia sus planes de juego y estrategias defensivas.

Gráfica de líneas - Evolución de tiros de la NBA:

La tendencia anual de la cantidad de tiros de campo (FGA) y tiros de tres puntos (FG3A) intentados en los juegos de playoffs de la NBA se muestra en esta gráfica de líneas. La línea FG3A comenzó con 2000 tiros en el año 0 y terminó con casi 3000 tiros en el año 10, lo que demuestra un aumento significativo en el número de tiros de tres puntos intentados a lo largo del tiempo. Sin embargo, la línea FGA comienza con más de 6000 tiros en el año 0 y termina con más de 7000 tiros en el año 10, lo que indica un aumento menos significativo en el número de tiros de campo intentados en comparación con los tiros de tres puntos.



Esta gráfica es significativa porque muestra las tendencias generales en las estrategias ofensivas de los equipos de la National Basketball Association a lo largo del tiempo. El número de tiros de tres puntos intentados puede indicar una mayor confianza en el juego exterior y una mayor dependencia del tiro de larga distancia en los playoffs de la NBA. Además, podría indicar cambios en las reglas del juego, el estilo de juego predominante o la inclinación de los entrenadores hacia un juego más centrado en el tiro de tres puntos.

CONCLUSIONES

El análisis de los datos de los playoffs de la NBA ha revelado patrones y tendencias en las estrategias ofensivas de los equipos durante esos momentos de la temporada. La hipótesis que se definió previamente se ha cumplido a manera que el porcentaje de tiros de campo encestados se encontró en una cantidad similar a la descrita, a pesar de que, en algunas fechas observadas, los porcentajes referentes a los tiros triples resultaron arriba del 20%.

Se observó un aumento en el porcentaje de tiros triples intentados entre los dos períodos analizados, lo que sugiere una mayor dependencia del juego exterior y una mayor confianza en el tiro de larga distancia. Este cambio podría ser el resultado de cambios en la estrategia, en el personal del equipo o en las defensas de los oponentes.

El análisis general mostró un aumento en el número de tiros de tres puntos intentados a lo largo de los años, lo que indica una mayor inclinación hacia un juego más enfocado en el tiro de larga distancia en los playoffs de la NBA. No obstante, hubo un aumento en el número de tiros de campo, aunque en menor medida, lo que indica que los tiros cercanos al aro y de mayor valor siguen siendo importantes en la estrategia ofensiva de los equipos.

Pandas, matplotlib y numpy fueron librerías de análisis y visualización de datos importantes, en especial al tratar temas estadísticos o bien de generación de resultados analíticos, proporcionando estructuras de datos flexibles, como DataFrames, filtrado y transformación de datos, cálculos y operaciones con matrices, además de la creación de representaciones visuales de datos de manera personalizada. Estas librerías de Python facilitaron la manipulación y modificación de datos, mientras que las funciones de visualización de matplotlib facilitaron la creación de gráficos y representaciones informativas de una manera sencilla de comprender para cualquier persona que le interese dicha información, tratándose desde aficionados hasta empresarios y deportistas de la misma área de negocios de NBA.