Trabajo Fin de Máster

Máster Data Science

Análisis Predictivo del Porcentaje de Triples de Jugadores Pivot en la NBA.

Autor:
Marc Tarín Vilar

Tutor:
Antonio Pita Lozano

ÍNDICE

1. Introducción

El trabajo de investigación que presentamos a continuación se centra en el análisis predictivo del porcentaje de triples de jugadores pivot en la NBA (National Basketball Association) mediante los datos históricos de esta liga.

El objetivo es intentar predecir, utilizando los datos que se tienen de las diferentes épocas ya transcurridas en la NBA, qué mejoras en el porcentaje de triples deberán conseguir los jugadores que desempeñen la posición de pívot para conseguir llegar a jugar en esta liga.

Para llevar a cabo este análisis, se ha utilizado un conjunto de datos que combina información de diversas temporadas de la NBA, incluyendo las estadísticas de jugadores y los datos específicos sobre sus habilidades en porcentajes de tiros de tres puntos.

Viendo la evolución en los porcentajes de tiros de tres puntos y cómo estas posiciones, que en décadas anteriores no conseguían unos buenos porcentajes, cada vez están mejorandolos y nuestra misión es conseguir una aproximación de donde deberá estar un jugador de estás características para que cuenten con él en esta liga.

La importancia de anticiparse qué necesidades debe cubrir un equipo de baloncesto en ciertas posiciones es fundamental viendo la evolución que el mundo del deporte está sufriendo en los últimos años y conseguir guiar tanto a los jugadores como a los entrenadores hacia esa evolución es lo que da tanto valor a este proyecto.

2. Metodología

A continuación explicamos el tipo de metodología que hemos utilizado para desarrollar este proyecto y el tipo de modelado predictivo que hemos ido implementando para encontrar los resultados obtenidos.

2.1 Recolección y Preprocesamiento de Datos

Inicialmente, se estudiaron tres conjuntos de datos en formato CSV que contenían información relevante para nuestro estudio:

- Datos históricos de la NBA,
- Estadísticas individuales de los jugadores.
- Datos individuales de los jugadores.

Estos conjuntos de datos fueron compaginados utilizando la biblioteca de Python, Pandas, y manipulados con NumPy para la concatenación y limpieza de datos necesaria.

Mediante la biblioteca de Matplotlib se diseñaron los gráficos necesarios para la visualización de los datos extraídos.

2.2 Análisis Exploratorio

Tras la consolidación de los datos, se llevó a cabo un análisis exploratorio para comprender mejor la distribución de los datos y detectar posibles tendencias o patrones en la evolución del porcentaje de triples de los jugadores pivot a lo largo de las temporadas.

2.3 Modelado Predictivo

Para predecir el porcentaje de triples de los jugadores pivots en los próximos cinco años a partir de la última temporada registrada (2017), se implementaron varios modelos de entrenamiento de predicciones para llegar al más acertado.

Modelo de Holt: Utilizado para modelar series de tiempo y predecir tendencias futuras. Se aplicó para capturar posibles patrones estacionales o tendencias de crecimiento en el porcentaje de triples de los jugadores pivots.

Modelo Random Forest: Un algoritmo de aprendizaje automático que utiliza múltiples árboles de decisión para hacer predicciones. Se utilizó para capturar relaciones no lineales entre las variables predictoras y el porcentaje de triples.

Modelo de Redes Neuronales: Una técnica de aprendizaje profundo que puede capturar relaciones complejas entre las variables de entrada y salida. Se implementó para explorar patrones más intrincados en los datos.

Modelo de Regresión Lineal y Polinómica: Utilizados como modelos base para comparar con enfoques más complejos. Se aplicaron para evaluar la eficacia de modelos más simples en la predicción del porcentaje de triples.

Modelo de Arima: también se utilizó el modelo ARIMA (Autoregressive Integrated Moving Average) para explorar aún más la evolución del porcentaje de triples de los jugadores pivot en la NBA. Este modelo se aplicó como una herramienta adicional para capturar posibles patrones temporales y comportamientos de tendencia en los datos.

3. EXPLICACIÓN DE MANIPULACIÓN DE DATOS Y LIMPIEZA Y PREPARACIÓN DEL DATO

El estudio que presentamos a continuación se centra en el análisis de datos relacionados con la NBA, con el objetivo de comprender la evolución de las estadísticas de los jugadores y las tendencias de la liga a lo largo del tiempo, específicamente en el porcentaje de triples de los pivots de esta liga.

Para ello, se han utilizado tres conjuntos de datos en formato CSV que abarcan diferentes aspectos de la NBA: datos individuales de los jugadores, estadísticas individuales de los mismos y los datos históricos de estadísticas de la NBA durante varios años.

3.1 Tratamiento de los datos con Pandas y Numpy

Para realizar un análisis exhaustivo de los datos, se emplearon las bibliotecas pandas y numpy de Python. Estas herramientas proporcionaron una amplia gama de funcionalidades para la manipulación y limpieza de datos, así como para la realización de operaciones numéricas y estadísticas necesarias para entenderlos.

- Pandas: Se utilizó para cargar los datos de los tres conjuntos CSV en estructuras de datos llamadas DataFrames. Gracias a ello se consiguió una fácil visualización y manipulación de los datos, incluyendo la selección de columnas relevantes, la eliminación de valores nulos y la conversión de tipos de datos.

```
# Cargar archivos

df_player_data = pd.read_csv('res/player_data.csv')
df_players = pd.read_csv('res/players.csv')
df_season_stats = pd.read_csv('res/Seasons_Stats.csv')
```

```
df_players.head()
```

	Unnamed: 0	Player	height	weight	collage	born	birth_city	birth_state
0	0	Curly Armstrong	180.0	77.0	Indiana University	1918.0	NaN	NaN
1	1	Cliff Barker	188.0	83.0	University of Kentucky	1921.0	Yorktown	Indiana
2	2	Leo Barnhorst	193.0	86.0	University of Notre Dame	1924.0	NaN	NaN
3	3	Ed Bartels	196.0	88.0	North Carolina State University	1925.0	NaN	NaN
4	4	Ralph Beard	178.0	79.0	University of Kentucky	1927.0	Hardinsburg	Kentucky

```
df_season_stats.head()
```

	Unnamed: 0	Year	Player	Pos	Age	Tm	G	GS	MP	PER	...	FT%	ORB	DRB	TRB	AST	STL	BLK	TOV	PF	PTS
0	0	1950.0	Curly Armstrong	G-F	31.0	FTW	63.0	NaN	NaN	NaN	...	0.705	NaN	NaN	NaN	176.0	NaN	NaN	NaN	217.0	458.0
1	1	1950.0	Cliff Barker	SG	29.0	INO	49.0	NaN	NaN	NaN	...	0.708	NaN	NaN	NaN	109.0	NaN	NaN	NaN	99.0	279.0
2	2	1950.0	Leo Barnhorst	SF	25.0	CHS	67.0	NaN	NaN	NaN	...	0.698	NaN	NaN	NaN	140.0	NaN	NaN	NaN	192.0	438.0
3	3	1950.0	Ed Bartels	F	24.0	TOT	15.0	NaN	NaN	NaN	...	0.559	NaN	NaN	NaN	20.0	NaN	NaN	NaN	29.0	63.0
4	4	1950.0	Ed Bartels	F	24.0	DNN	13.0	NaN	NaN	NaN	...	0.548	NaN	NaN	NaN	20.0	NaN	NaN	NaN	27.0	59.0

5 rows × 53 columns


```
df_season_stats.head()
```

	Unnamed: 0	Year	Player	Pos	Age	Tm	G	GS	MP	PER	...	FT%	ORB	DRB	TRB	AST	STL	BLK	TOV	PF	PTS
0	0	1950.0	Curly Armstrong	G-F	31.0	FTW	63.0	NaN	NaN	NaN	...	0.705	NaN	NaN	NaN	176.0	NaN	NaN	NaN	217.0	458.0
1	1	1950.0	Cliff Barker	SG	29.0	INO	49.0	NaN	NaN	NaN	...	0.708	NaN	NaN	NaN	109.0	NaN	NaN	NaN	99.0	279.0
2	2	1950.0	Leo Barnhorst	SF	25.0	CHS	67.0	NaN	NaN	NaN	...	0.698	NaN	NaN	NaN	140.0	NaN	NaN	NaN	192.0	438.0
3	3	1950.0	Ed Bartels	F	24.0	TOT	15.0	NaN	NaN	NaN	...	0.559	NaN	NaN	NaN	20.0	NaN	NaN	NaN	29.0	63.0
4	4	1950.0	Ed Bartels	F	24.0	DNN	13.0	NaN	NaN	NaN	...	0.548	NaN	NaN	NaN	20.0	NaN	NaN	NaN	27.0	59.0

- Numpy: Esta biblioteca fue fundamental para realizar operaciones numéricas en los datos, como el cálculo de promedios, desviaciones estándar y otras estadísticas importantes para el análisis.

```
media_3P_año = df_clean.groupby('Year')['3P%'].mean()
media_3P_año
```

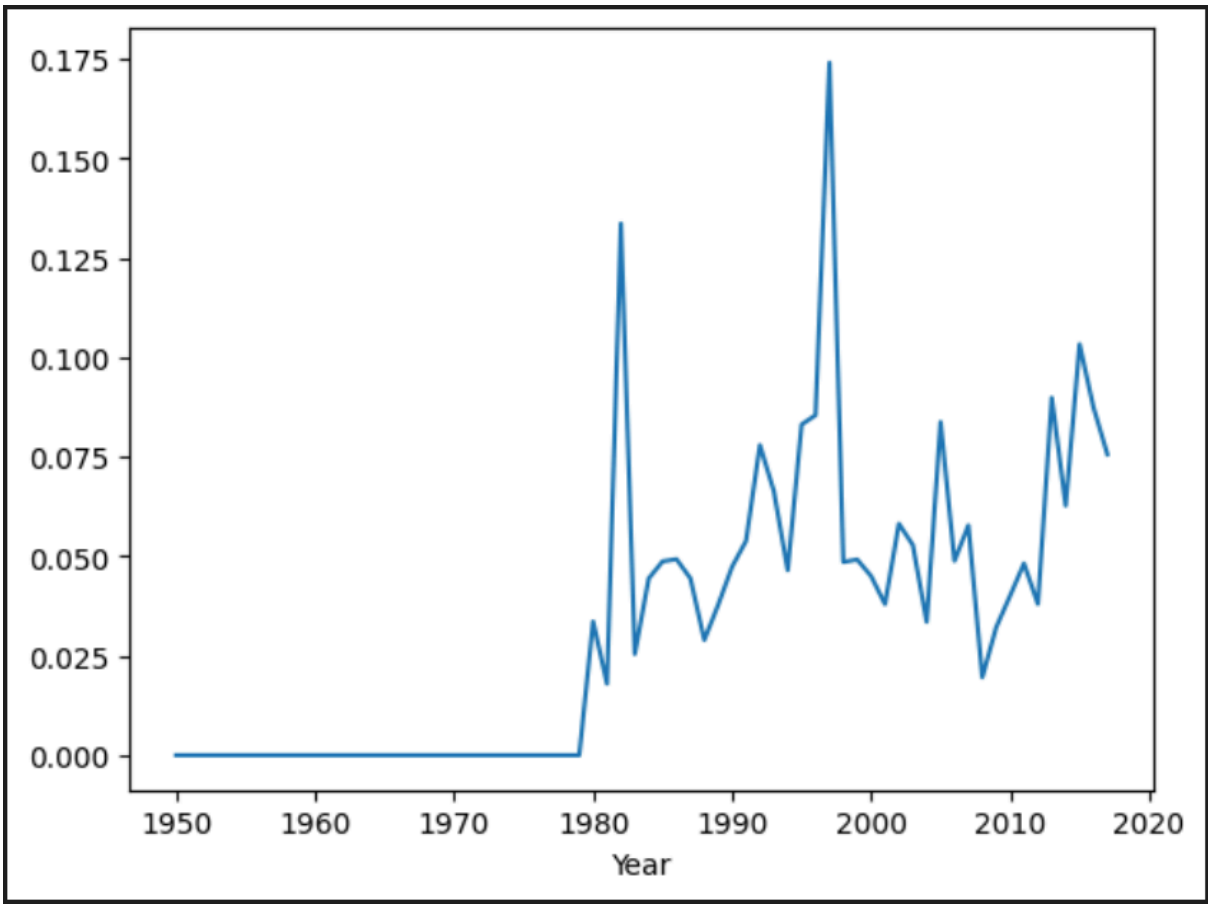
```
Year
1950.0    0.000000
1951.0    0.000000
1952.0    0.000000
1953.0    0.000000
1954.0    0.000000
            ...
2013.0    0.234282
2014.0    0.243674
2015.0    0.250627
2016.0    0.267248
2017.0    0.278438
```

3.2 Visualización de datos con Matplotlib

Una vez que los datos se procesaron y se prepararon, se utilizó la biblioteca Matplotlib para visualizar los resultados del análisis.

Matplotlib ofrece una variedad de herramientas para crear gráficos informativos y visualmente atractivos, lo que facilita la comprensión de los patrones y tendencias presentes en los datos.

- Gráficos de líneas y dispersión: Se utilizaron para representar la evolución de las estadísticas de los jugadores a lo largo de los años, así como para comparar diferentes métricas entre sí.
- Histogramas y diagramas de caja: Estos gráficos fueron útiles para explorar la distribución de los datos y detectar posibles valores atípicos o tendencias en las estadísticas de la liga.



Por lo que podemos concluir que el uso de pandas, numpy y matplotlib nos permitió una exploración completa y visualmente atractiva de los datos de la NBA, lo que proporcionó insights valiosos sobre la evolución de las estadísticas de los jugadores y las dinámicas de la NBA a lo largo de sus temporadas.

3.3 Proceso de manipulación y limpieza de datos.

Al cargar los datos nos dimos cuenta que no todas las columnas que incluían los mismos datos tenían el mismo nombre de columna por lo que para poder unificar todos los datos en un mismo csv tuvimos que modificarlas todas con un mismo identificador.

```
# Renombrar la columna name por Player.

df_data = df_player_data.rename(columns = {'name':'Player'})
df_data
```

	Player	year_start	year_end	position	height	weight	birth_date	college
0	Alaa Abdelnaby	1991	1995	F-C	6-10	240.0	June 24, 1968	Duke University
1	Zaid Abdul-Aziz	1969	1978	C-F	6-9	235.0	April 7, 1946	Iowa State University
2	Kareem Abdul-Jabbar	1970	1989	C	7-2	225.0	April 16, 1947	University of California, Los Angeles
3	Mahmoud Abdul-Rauf	1991	2001	G	6-1	162.0	March 9, 1969	Louisiana State University
4	Tariq Abdul-Wahad	1998	2003	F	6-6	223.0	November 3, 1974	San Jose State University
...	...	...	...	...	...	...	...	...

Al igual que los datos que se encontraban en las columnas también estaban en diferentes connotaciones, por ejemplo el peso o la altura, así como la posición que ocupaba cada jugador.

Por lo que se tuvo que eliminar las columnas repetidas para focalizarnos sólo en los datos que nos dieran valor y fueran relevantes para el estudio.

```
# Eliminando las columnas height_y(pies) y weight_y(libras)

df_clean = df_clean.drop(['height_y','weight_y'], axis=1)
```

Es importante eliminar los NaN que aparecen en algunas columnas puesto que identificamos que todos ellos estaban en las épocas iniciales donde no existía la línea de tres puntos y por lo que se decidió eliminarlos para no influir en el estudio.

```
# Eliminar los NaN del dataframe

df_clean = df_info_all.fillna(0)
```

Una vez tuvimos los tres csv con toda la información unificada con la misma nomenclatura pasamos a realizar la unión de los 3 csv en uno para su preparación y siguiente estudio.

```
# Unión de los tres Dataframe por Player.

df_ply_data = pd.merge(df_players, df_data, on= 'Player')
df_info_all = pd.merge(df_ply_data, df_season_stats, on='Player')
df_info_all.head()
```

La siguiente tarea que llevamos a cabo fue eliminar todas las columnas que no nos aportaban datos importantes para el estudio que estamos realizando como el año de universidad, cumpleaños, lugar de nacimiento…

```
# Eliminando columnas no necesarias para la investigación

df_clean = df_clean.drop(['birth_city','birth_state','collage','born','Unnamed: 0_y','Unnamed: 0_x'], axis=1)
```

De esta manera conseguimos un único csv con toda la información necesaria para realizar los estudios siguientes, tener los datos listo y adecuados para crear los modelos de aprendizaje que se utilizarán a continuación.

```
# DATAFRAME LIMPIO Y LISTO PARA TRABAJAR

df_clean
```

	Player	height_x	weight_x	year_start	year_end	position	birth_date	Year	Pos	Age	...	FT%	ORB	DRB	TRB	AST	STL	BLK	TOV	PF	PTS
0	Curly Armstrong	180.0	77.0	1949	1951	G-F	November 1, 1918	1950.0	G-F	31.0	...	0.705	0.0	0.0	0.0	176.0	0.0	0.0	0.0	217.0	458.0
1	Curly Armstrong	180.0	77.0	1949	1951	G-F	November 1, 1918	1951.0	G-F	32.0	...	0.644	0.0	0.0	89.0	77.0	0.0	0.0	0.0	97.0	202.0
2	Cliff Barker	188.0	83.0	1950	1952	G	January 15, 1921	1950.0	SG	29.0	...	0.708	0.0	0.0	0.0	109.0	0.0	0.0	0.0	99.0	279.0
3	Cliff Barker	188.0	83.0	1950	1952	G	January 15, 1921	1951.0	SG	30.0	...	0.649	0.0	0.0	100.0	115.0	0.0	0.0	0.0	98.0	152.0
4	Cliff Barker	188.0	83.0	1950	1952	G	January 15, 1921	1952.0	SG	31.0	...	0.588	0.0	0.0	81.0	70.0	0.0	0.0	0.0	56.0	126.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
23463	Troy Williams	198.0	97.0	2017	2018	F	December 30, 1994	2017.0	SF	22.0	...	0.857	9.0	15.0	24.0	6.0	3.0	1.0	6.0	18.0	58.0
23464	Kyle Wiltjer	208.0	108.0	2017	2017	F	October 20, 1992	2017.0	PF	24.0	...	0.500	4.0	6.0	10.0	2.0	3.0	1.0	5.0	4.0	13.0
23465	Stephen Zimmerman	213.0	108.0	2017	2017	C	September 9, 1996	2017.0	C	20.0	...	0.600	11.0	24.0	35.0	4.0	2.0	5.0	3.0	17.0	23.0
23466	Paul Zipser	203.0	97.0	2017	2018	G-F	February 18, 1994	2017.0	SF	22.0	...	0.775	15.0	110.0	125.0	36.0	15.0	16.0	40.0	78.0	240.0
23467	Ivica Zubac	216.0	120.0	2017	2018	C	March 18, 1997	2017.0	C	19.0	...	0.653	41.0	118.0	159.0	30.0	14.0	33.0	30.0	66.0	284.0

23468 rows × 58 columns

4. EXPLICACIÓN DE LOS MODELOS UTILIZADOS PARA LAS PREDICCIONES

4.1 MODELO DE HOLT

Importación de bibliotecas:

Se importan las bibliotecas necesarias, incluyendo pandas para la manipulación de datos, numpy para operaciones numéricas, matplotlib para visualización de datos y ExponentialSmoothing de statsmodels.tsa.holtwinters para implementar el modelo de Holt.

Preprocesamiento de datos:

- pd.to_datetime(): Convierte la columna 'Year' en formato de fecha.
- set_index(): Establece el índice del DataFrame como la columna 'Year'.

División de datos:

- train_data: Selecciona todos los datos de porcentaje de triples ('3P%') excepto los últimos 5 años para el conjunto de entrenamiento.
- test_data: Selecciona los últimos 5 años de datos de porcentaje de triples para el conjunto de prueba.

Entrenamiento del modelo Holt:

- ExponentialSmoothing(): Crea una instancia del modelo de suavizado exponencial (Holt).
- fit(): Ajusta el modelo a los datos de entrenamiento.

Predicciones:

- predict(): Genera predicciones para el conjunto de prueba utilizando el modelo entrenado. Se especifica el inicio y fin de las fechas de predicción.

Visualización de resultados:

- Se grafican los datos de entrenamiento, los datos de prueba y las predicciones.
- plt.xlabel() y plt.ylabel(): Etiqueta los ejes x e y del gráfico, respectivamente.
- plt.legend(): Muestra la leyenda del gráfico con etiquetas para cada serie de datos.
- plt.show(): Muestra el gráfico.

En resumen, el código realiza la preparación de los datos, entrena un modelo de Holt utilizando el conjunto de entrenamiento, genera predicciones para el conjunto de prueba y finalmente visualiza los datos de entrenamiento, los datos de prueba y las predicciones realizadas por el modelo.

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.holtwinters import ExponentialSmoothing


tabla_resultado_P['Year'] = pd.to_datetime(tabla_resultado_P['Year'], format='%Y')
tabla_resultado_P.set_index('Year', inplace=True)

# Dividir el conjunto de datos en entrenamiento y prueba
train_data = tabla_resultado_P['3P%'].iloc[:-5]  # Tomar todos menos los últimos 5 años
test_data = tabla_resultado_P['3P%'].iloc[-5:]   # Tomar los últimos 5 años para la evaluación

# Crear y entrenar el modelo Holt
model = ExponentialSmoothing(train_data, trend='add', seasonal='add', seasonal_periods=12)
result = model.fit()

# Realizar predicciones
predictions = result.predict(start=test_data.index[0], end=test_data.index[-1])

# Graficar resultados
plt.plot(train_data.index, train_data, label='Datos de entrenamiento')
plt.plot(test_data.index, test_data, label='Datos de prueba', color='orange')
plt.plot(predictions.index, predictions, label='Predicciones', color='green')
plt.xlabel('Year')
plt.ylabel('3P%')
plt.legend()
plt.show()
```

4.2 MODELO DE RANDOM FOREST

Importación de bibliotecas:

Se importan las bibliotecas necesarias, incluyendo RandomForestRegressor de sklearn.ensemble para implementar el modelo de Random Forest, train_test_split de sklearn.model_selection para dividir los datos en conjuntos de entrenamiento y prueba, mean_squared_error de sklearn.metrics para evaluar el modelo, numpy para operaciones numéricas y matplotlib.pyplot para visualización de datos.

División de datos:

- train_test_split(): Divide los datos en conjuntos de entrenamiento y prueba. Se selecciona la columna 'Year' como variable independiente (X) y la columna '3P%' como variable dependiente (y). Se utiliza el 80% de los datos para entrenamiento y el 20% para la prueba.

Creación y entrenamiento del modelo Random Forest:

- RandomForestRegressor(): Crea una instancia del modelo de regresión de Random Forest con 100 árboles.
- fit(): Entrena el modelo utilizando los datos de entrenamiento.

Predicciones y evaluación del modelo:

- predict(): Realiza predicciones sobre los datos de prueba.
- mean_squared_error(): Calcula el error cuadrático medio entre las predicciones y los valores reales de los datos de prueba.

Predicciones futuras:

- Se generan predicciones para los próximos 5 años (2018-2022) utilizando el modelo entrenado.

Visualización de resultados:

- Se grafican los datos históricos y las predicciones futuras.
- plt.scatter(): Se utilizan gráficos de dispersión para mostrar los datos históricos y las predicciones futuras.

- plt.xlabel() y plt.ylabel(): Etiqueta los ejes x e y del gráfico, respectivamente.
- plt.legend(): Muestra la leyenda del gráfico con etiquetas para cada serie de datos.
- plt.show(): Muestra el gráfico.

En resumen, el código divide los datos en conjuntos de entrenamiento y prueba, entrena un modelo de Random Forest utilizando los datos de entrenamiento, realiza predicciones sobre los datos de prueba, evalúa el rendimiento del modelo y finalmente genera predicciones futuras y las visualiza junto con los datos históricos.

```python
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
import numpy as np
import matplotlib.pyplot as plt



# Dividir el conjunto de datos en entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(tabla_resultado_P['Year'].values.reshape(-1, 1), tabla_resultado_P['3P%'].values, test_size=0.2, random_state=42)

# Crear un modelo de Random Forest
model = RandomForestRegressor(n_estimators=100, random_state=42)

# Entrenar el modelo
model.fit(X_train, y_train)

# Realizar predicciones
y_pred = model.predict(X_test)

# Evaluar el modelo
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse}')

# Predecir para los próximos 5 años
future_years = np.arange(2018, 2023).reshape(-1, 1)
future_predictions = model.predict(future_years)

# Graficar resultados
plt.scatter(tabla_resultado_P['Year'], tabla_resultado_P['3P%'], label='Datos históricos')
plt.scatter(future_years, future_predictions, color='green', label='Predicciones futuras (Random Forest)')
plt.xlabel('Year')
plt.ylabel('3P%')
plt.legend()
plt.show()
```

4.3 ARIMA

Creación del modelo ARIMA:

- ARIMA(): Se realiza un modelo ARIMA utilizando la función ARIMA de la biblioteca statsmodels.
- Se especifica endog, la variable endógena, es decir, la variable de interés que se está modelando o pronosticando, como la serie temporal de los datos de entrenamiento tabla_media_triples_P_2000, y se establece el orden del modelo ARIMA como (5,1,1), indicando un componente autoregresivo de orden 5, un componente de diferenciación de orden 1 y un componente de media móvil de orden 1.

Entrenamiento del modelo:

- fit(): Se ajusta el modelo ARIMA a los datos de entrenamiento mediante el método fit().

Resumen del modelo:

- summary(): Se imprime un resumen del ajuste del modelo ARIMA, que incluye estadísticas como coeficientes, errores estándar, valores p y otros detalles relevantes del modelo.

Predicciones:

- forecast(): Se utilizan las predicciones del modelo ARIMA para predecir los próximos 5 años (2010-2014) después de los datos de entrenamiento.
- predicciones.index: Se establece el índice de las predicciones con años correspondientes a los próximos 5 años.

Visualización de resultados:

- Se crea una figura y un eje para el gráfico.
- plot(): Se grafican los datos de entrenamiento y las predicciones en el mismo gráfico.
- plt.show(): Se muestra el gráfico.

En resumen, el código crea un modelo ARIMA, lo ajusta a los datos de entrenamiento, realiza predicciones para los próximos 5 años y luego visualiza tanto los datos de entrenamiento como las predicciones en un mismo gráfico.

```
arima = ARIMA(endog = tabla_media_triples_P_2000, order = (5,1,1))
arima_fit = arima.fit()
print(arima_fit.summary())
```

```
predicciones = arima_fit.forecast(steps= 5)
predicciones.index = [2010.0,2011.0,2012.0, 2013.0,2014.0]
```

```
fig, ax = plt.subplots(1,1)
tabla_media_triples_P.plot(ax = ax, label = 'Train')
predicciones.plot(ax = ax, label = 'Predicciones')
plt.show()
```

4.4 REGRESIÓN LINEAL

División de datos:

- train_test_split(): Los datos se dividen en conjuntos de entrenamiento y prueba. Las variables independientes (X) corresponden a los años ('Year') y las variables dependientes (y) corresponden al porcentaje de triples ('3P%'). Se utiliza el 80% de los datos para entrenamiento y el 20% para la prueba.

Creación del modelo de regresión lineal:

- LinearRegression(): Se realiza un modelo de regresión lineal utilizando la función LinearRegression de la biblioteca scikit-learn.

Entrenamiento del modelo:

- fit(): Se ajusta el modelo de regresión lineal a los datos de entrenamiento. El modelo aprenderá la relación entre los años y el porcentaje de triples a partir de los datos de entrenamiento.

Predicciones y evaluación del modelo:

- predict(): Se utilizan las variables independientes del conjunto de prueba para predecir el porcentaje de triples utilizando el modelo entrenado.
- mean_squared_error(): Se calcula el error cuadrático medio entre las predicciones y los valores reales del conjunto de prueba. Este valor proporciona una medida de la calidad de las predicciones del modelo.

Predicciones futuras:

- Se generan predicciones para los próximos 5 años (2018-2022) utilizando el modelo entrenado. Se crea un array future_years con los años correspondientes a estos próximos años y se utiliza el método predict() del modelo de regresión lineal para obtener las predicciones.

Visualización de resultados:

- Se grafican los datos históricos y las predicciones del modelo de regresión lineal en el mismo gráfico.

- Se utiliza plt.scatter() para mostrar los datos históricos y las predicciones futuras.
- Se utiliza plt.plot() para trazar la línea de regresión lineal que representa la relación entre los años y el porcentaje de triples.
- Se establecen etiquetas para los ejes x e y del gráfico y se muestra una leyenda para diferenciar entre los datos históricos, la regresión lineal y las predicciones futuras.

En resumen, el modelo de regresión lineal se utiliza para modelar la relación entre los años y el porcentaje de triples de los jugadores pivot en la NBA. El modelo se entrena con datos históricos y se utiliza para hacer predicciones sobre el porcentaje de triples en los próximos años. La visualización proporciona una representación clara de cómo el porcentaje de triples ha evolucionado y se espera que evolucione en el futuro.

```python
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import numpy as np
import matplotlib.pyplot as plt

# Supongamos que df es tu DataFrame con las columnas 'Años' y 'PorcentajeTriples'

# Dividir el conjunto de datos en entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(tabla_resultado_P['Year'].values.reshape(-1, 1), tabla_resultado_P['3P%'].values, test_size=0.2, random_state=42)

# Crear un modelo de regresión lineal
model = LinearRegression()

# Entrenar el modelo
model.fit(X_train, y_train)

# Realizar predicciones
y_pred = model.predict(X_test)

# Evaluar el modelo
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse}')

# Predecir para los próximos 5 años
future_years = np.arange(2018, 2023).reshape(-1, 1)
future_predictions = model.predict(future_years)

# Graficar resultados
plt.scatter(tabla_resultado_P['Year'], tabla_resultado_P['3P%'], label='Datos históricos')
plt.plot(X_test, y_pred, color='red', label='Regresión lineal')
plt.scatter(future_years, future_predictions, color='green', label='Predicciones futuras')
plt.xlabel('Year')
plt.ylabel('3P%')
plt.legend()
plt.show()
```

4.5 REGRESIÓN POLINÓMICA

Creación del modelo de regresión polinómica:

- Se crea un modelo de regresión polinómica de grado 2 utilizando la función make_pipeline de scikit-learn. Este modelo combina dos pasos:
  - PolynomialFeatures(degree=2): Transforma las características existentes en características polinómicas de segundo grado.
  - LinearRegression(): Utiliza una regresión lineal para ajustar los datos transformados.

Entrenamiento del modelo:

- Se ajusta el modelo de regresión polinómica a los datos de entrenamiento utilizando el método fit().

Predicciones y evaluación del modelo:

- Se utilizan las variables independientes del conjunto de prueba para hacer predicciones sobre el porcentaje de triples utilizando el modelo entrenado.
- Se calcula el error cuadrático medio entre las predicciones y los valores reales del conjunto de prueba utilizando mean_squared_error().

Predicciones futuras:

- Se generan predicciones para los próximos 5 años (2018-2022) utilizando el modelo entrenado. Esto se hace con el método predict().

Visualización de resultados:

- Se grafican los datos históricos y las predicciones del modelo de regresión polinómica en el mismo gráfico.
- Se utiliza plt.scatter() para mostrar los datos históricos y las predicciones futuras.
- Se establecen etiquetas para los ejes x e y del gráfico y se muestra una leyenda para diferenciar entre los datos históricos y las predicciones futuras del modelo polinómico.

Este enfoque polinómico permite capturar relaciones no lineales entre los años y el porcentaje de triples, lo que podría ser útil para modelar patrones más complejos en los datos históricos y hacer predicciones más precisas para el futuro.

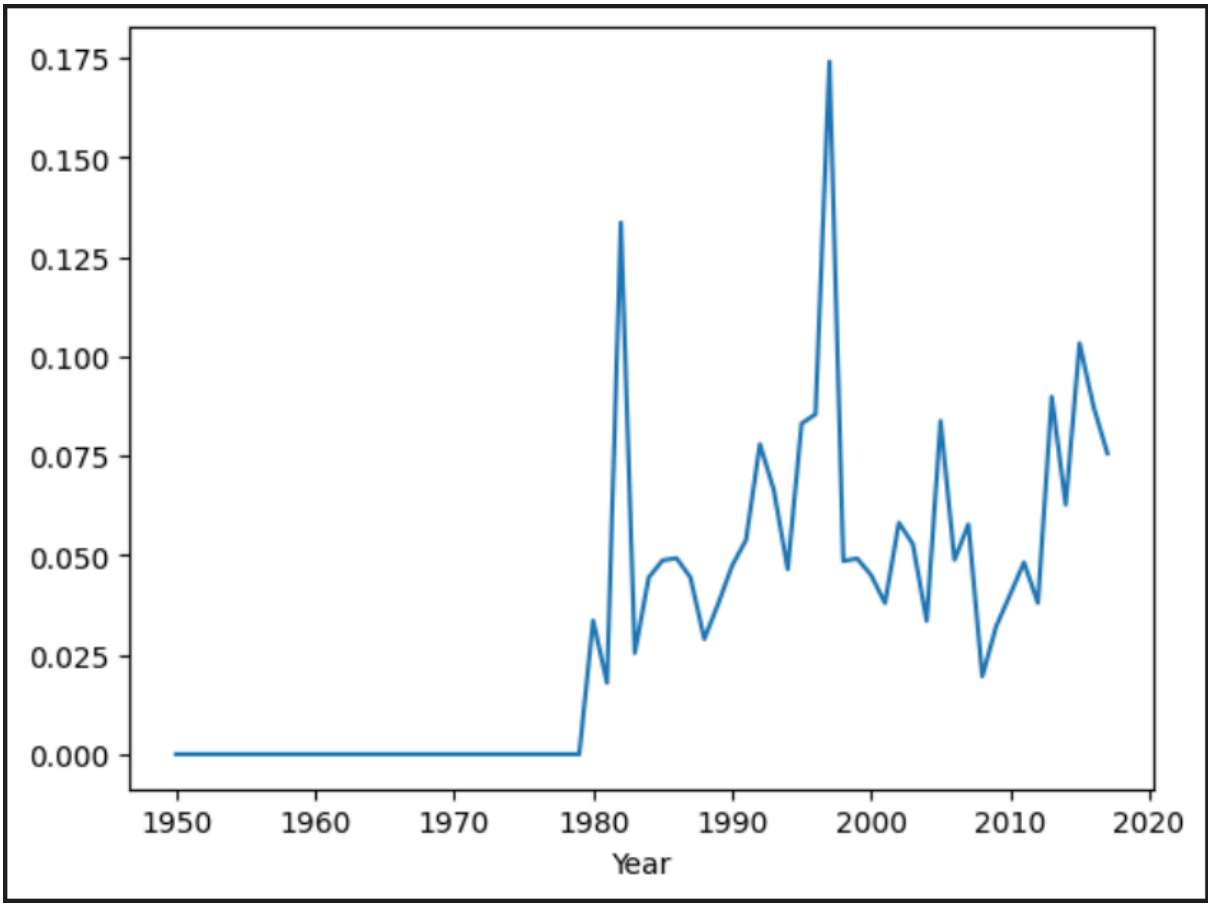
```python
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import make_pipeline

# Crear un modelo de regresión polinómica de grado 2
model = make_pipeline(PolynomialFeatures(degree=2), LinearRegression())

# Entrenar el modelo
model.fit(X_train, y_train)

# Realizar predicciones
y_pred_poly = model.predict(X_test)

# Evaluar el modelo
mse_poly = mean_squared_error(y_test, y_pred_poly)
print(f'Mean Squared Error (Polinómico): {mse_poly}')

# Predecir para los próximos 5 años
future_predictions_poly = model.predict(future_years)

# Graficar resultados
plt.scatter(tabla_resultado_P['Year'], tabla_resultado_P['3P%'], label='Datos históricos')
plt.scatter(future_years, future_predictions_poly, color='purple', label='Predicciones futuras (Polinómico)')
plt.xlabel('Year')
plt.ylabel('3P%')
plt.legend()
plt.show()
```

5. Resultados

Para comprobar si los modelos explicados anteriormente nos daban unos resultados fiables para poder tomar una decisión óptima de cómo un jugador, en este caso en la posición de pívot, debería alcanzar cierto porcentaje de triple para poder llegar a la NBA.

Se trabajó con datos ya existentes y se predijeron para comprobar que tan fiable era esa predicción.

Los datos iniciales eran estos:



Se estudió los últimos cinco años para comprobar si con los datos que ya se tenían se podría aproximar a los datos ya existentes.

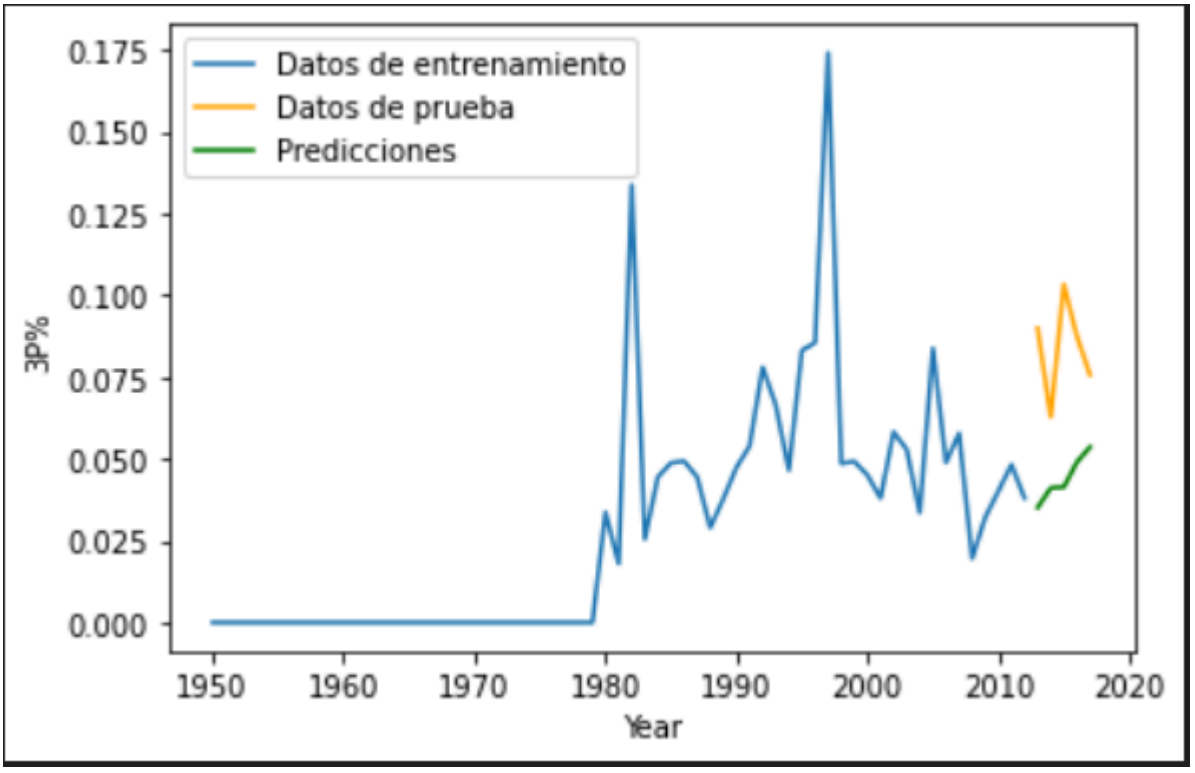
A continuación mostraremos los datos obtenidos con cada método y el porqué de su poca fiabilidad para predecir buenos datos en un futuro.

5.1 Método de Holt

El método de Holt es un modelo de suavizado exponencial que puede no ser completamente confiable para pronosticar el porcentaje de triples de los jugadores pivot en la NBA debido a su suposición de que los datos siguen una tendencia y una estacionalidad constantes.

Sin embargo, en la NBA, las dinámicas de los equipos, los jugadores y las estrategias pueden cambiar drásticamente de una temporada a otra, lo que podría llevar a fluctuaciones impredecibles en el porcentaje de triples.

Por lo tanto, el modelo de Holt podría no ser capaz de capturar adecuadamente tales cambios y, como resultado, las predicciones podrían ser inexactas.



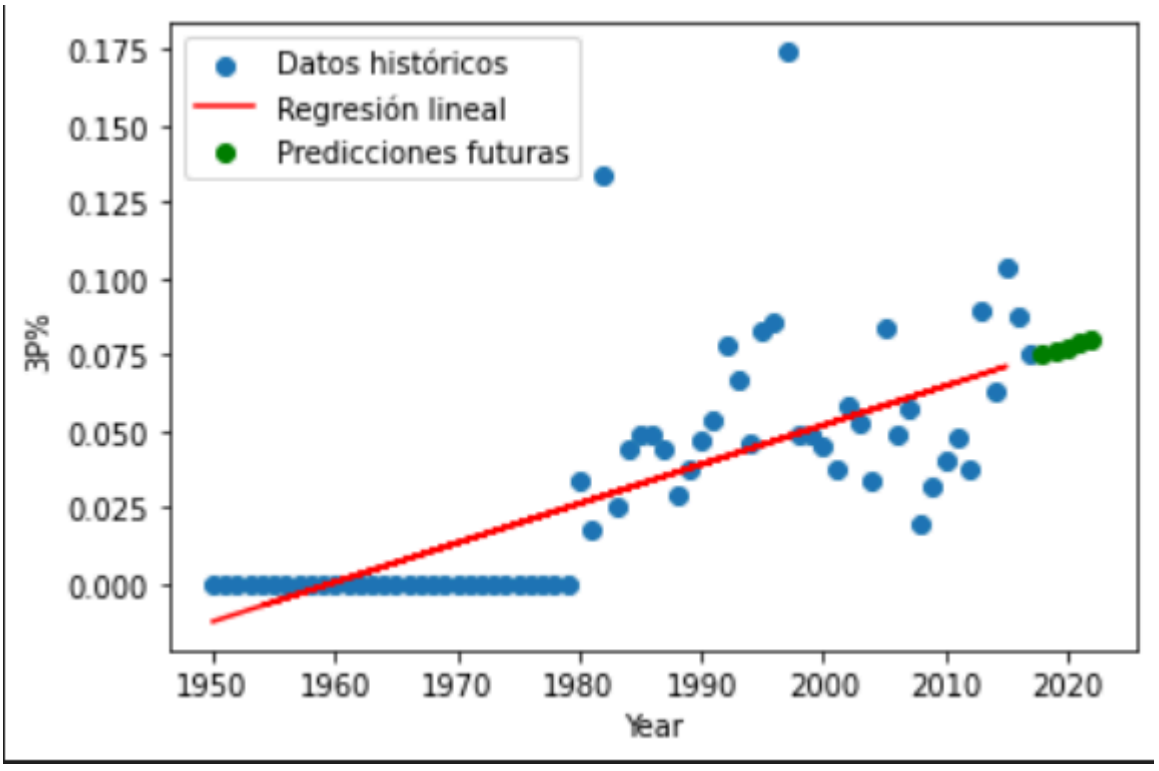
5.2 Regresión Lineal y Regresión Polinómica

La regresión lineal y la regresión polinómica son modelos que suponen una relación lineal o polinómica entre las variables independientes (años) y la variable dependiente (porcentaje de triples de los jugadores pivot).

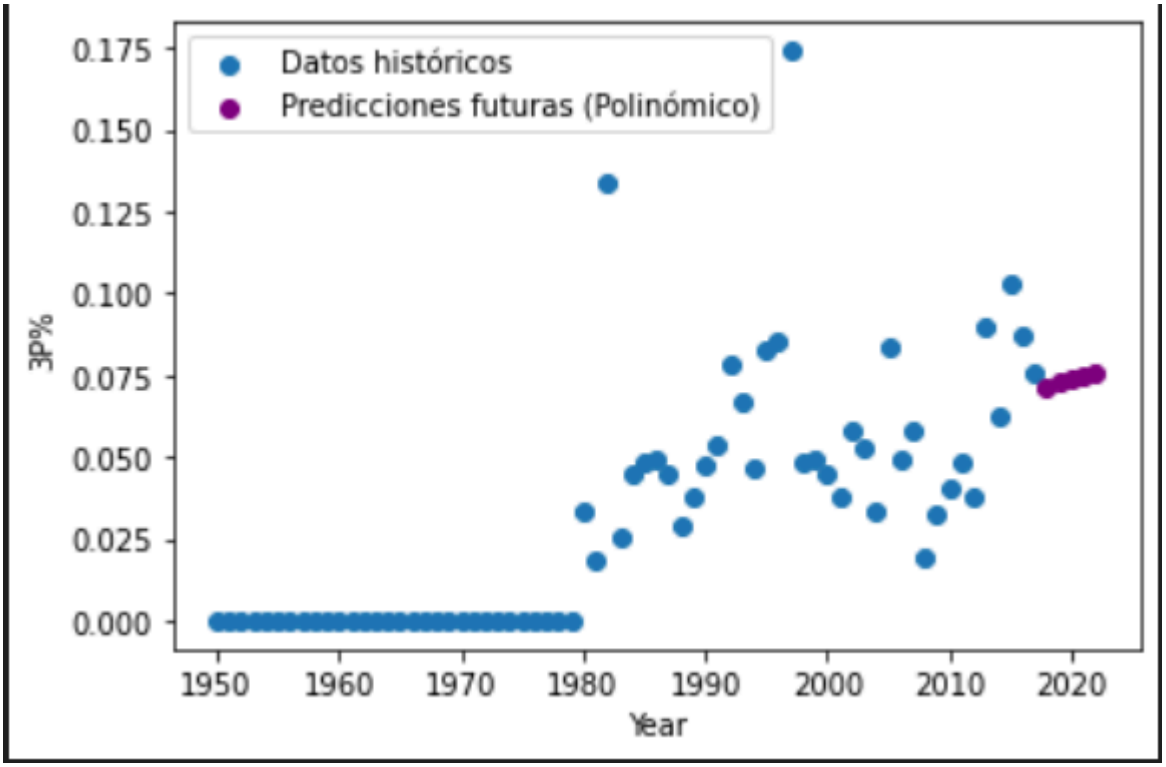
Sin embargo, en la NBA, las tendencias y los patrones pueden ser no lineales y estar influenciados por una variedad de factores complejos, como cambios en las reglas del juego, lesiones de jugadores clave, y la evolución de las estrategias de equipo.

Estos modelos pueden no ser capaces de capturar adecuadamente tales relaciones complejas, lo que podría conducir a predicciones poco fiables, especialmente cuando se enfrentan a datos con comportamientos no lineales.

Regresión Lineal:



Regresión Polinómica

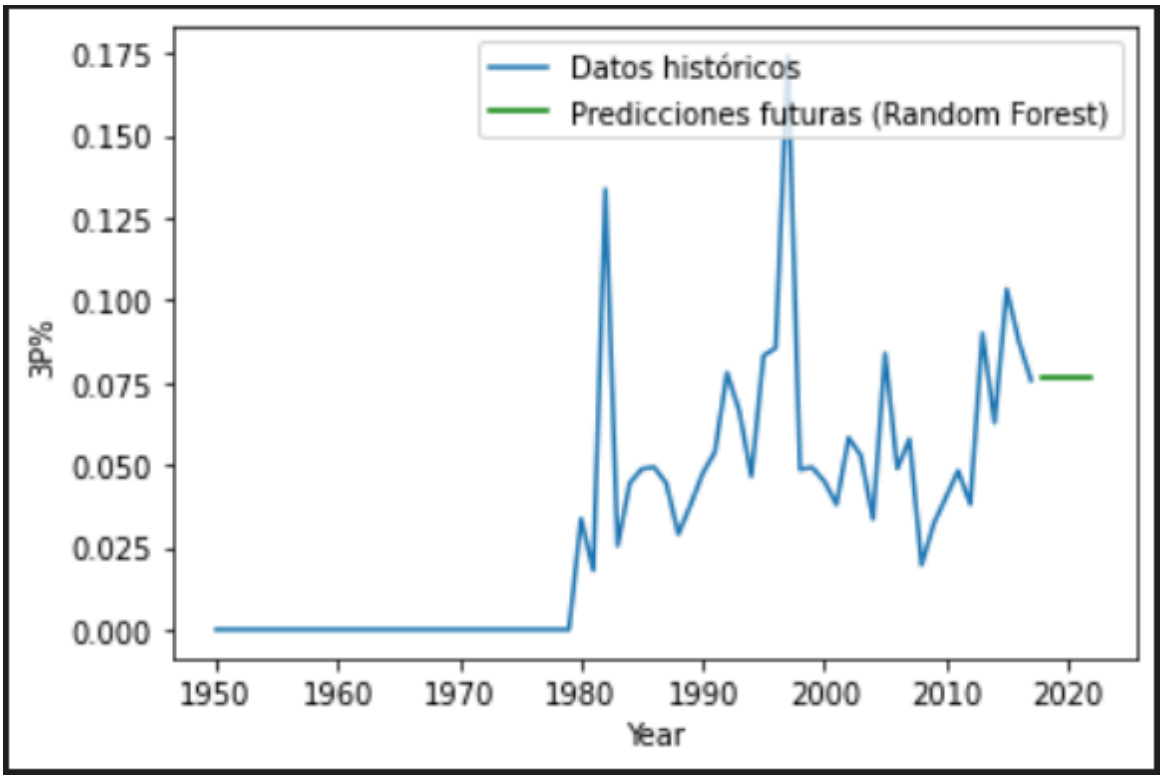


5.3 Modelo de Random Forest

Aunque el modelo de Random Forest es más flexible que la regresión lineal y puede capturar relaciones no lineales entre las variables, puede ser propenso a sobre ajustarse a los datos de entrenamiento si se le permite demasiada flexibilidad.

Esto significa que el modelo puede aprender las fluctuaciones y los patrones específicos de los datos de entrenamiento en lugar de generalizar correctamente para hacer predicciones precisas sobre datos nuevos o futuros.

Por lo tanto, las predicciones realizadas por un modelo de Random Forest pueden no ser fiables si el modelo está sobre ajustado a los datos de entrenamiento.



5.4 Modelo de Redes Neuronales

Las redes neuronales son capaces de modelar relaciones complejas en los datos, pero pueden requerir grandes cantidades de datos y un ajuste cuidadoso de los hiper parámetros para obtener un rendimiento óptimo.

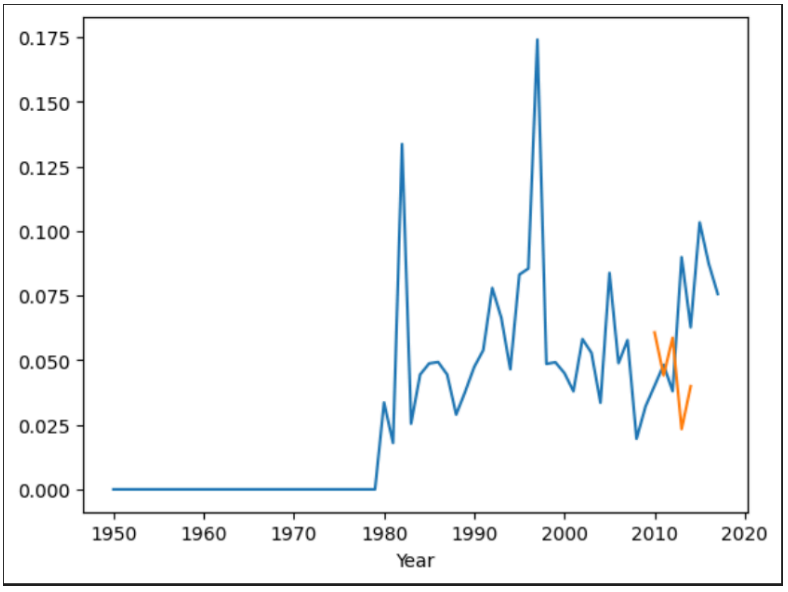
Además, la interpretación de las predicciones de una red neuronal puede ser difícil, lo que puede limitar su utilidad en términos de explicar los factores que afectan el porcentaje de triples de los jugadores pivot en la NBA. Además, las redes neuronales pueden ser propensas al sobreajuste si no se ajustan adecuadamente, lo que podría llevar a predicciones poco fiables en datos nuevos o futuros.

5.5 Modelo ARIMA

El modelo ARIMA es adecuado para datos de series temporales estacionarios, pero puede no ser suficiente para capturar la variabilidad y las tendencias no lineales presentes en los datos de la NBA.

Además, este modelo asume que los datos tienen una estructura temporal clara y predecible, lo cual puede no ser el caso en un entorno tan dinámico como el baloncesto profesional.

Por lo tanto, el modelo ARIMA puede no ser completamente confiable para hacer predicciones precisas sobre el porcentaje de triples de los jugadores pivot en la NBA.



Los resultados de los modelos predictivos revelaron diferentes perspectivas sobre la evolución futura del porcentaje de triples de los jugadores pivot en la NBA.

Tras el estudio, cada modelo ofreció predicciones con poco grado de precisión, se observaron diferencias significativas en sus enfoques y resultados.

Este estudio nos proporciona una visión integral del análisis predictivo del porcentaje de triples de los jugadores pivot en la NBA, utilizando una variedad de técnicas y modelos. No se han podido obtener resultados prometedores, pero es importante tener en cuenta que cualquier predicción futura está sujeta a la incertidumbre inherente en los datos y a las limitaciones de los modelos utilizados.

6. Futuras Direcciones

Para futuras investigaciones, se sugiere explorar en mayor profundidad otros factores que puedan influir en el porcentaje de triples de los jugadores pivot mediante información más precisa que se pueda obtener durante los próximos años, como cambios en las reglas del juego, avances en la tecnología de entrenamiento o la evolución del estilo de juego en la NBA.

Además, sería beneficioso mejorar la precisión de las predicciones mediante la incorporación de más datos y la optimización de los modelos predictivos durante los años venideros gracias a las mejoras de obtención de datos que se generen.

7. Conclusión

Tras una exhaustiva evaluación visual de los resultados obtenidos de cada uno de los modelos de entrenamiento utilizados en este proyecto, se puede observar que ninguno de ellos se ajusta adecuadamente a la tendencia de la serie temporal de datos de porcentaje de triples de los jugadores pivot en la NBA.

A pesar de los diferentes enfoques y niveles de complejidad de los modelos empleados, ninguno logra capturar de manera satisfactoria las fluctuaciones y patrones presentes en los datos.

El análisis detallado revela que las predicciones generadas por el método de Holt, la regresión lineal, la regresión polinómica, el modelo de Random Forest, las redes neuronales y el modelo ARIMA exhiben desviaciones significativas con respecto a la tendencia general de la serie temporal.

Estas desviaciones sugieren que los modelos no son capaces de capturar la complejidad inherente de los datos de la NBA, que están influenciados por una multitud de factores dinámicos, como cambios en las reglas del juego, lesiones de jugadores clave, la evolución de las estrategias de equipo… que entre otros factores modifican de manera significativa los datos que se han utilizado.

También se valoró crear un 'Tableau' con toda la información obtenida durante el estudio pero al crearlo no se obtuvo ningún valor añadido del que ya habíamos obtenido mediante los procesos realizados anteriormente y se decidió no añadirlo al proyecto.

En consecuencia, basándonos en los datos obtenidos para la realización del estudio, es evidente que ninguno de los modelos proporciona predicciones lo suficientemente precisas como para ser consideradas útiles para una predicción acertada del porcentaje de triples de los jugadores pivot en la NBA.

Estos resultados hallados en el estudio destacan la complejidad de modelar y predecir el comportamiento en un entorno tan dinámico como el baloncesto profesional en la NBA, lo que subraya la importancia de considerar cuidadosamente el enfoque de modelado y la selección de datos al abordar problemas de predicción en este campo para futuros estudios sobre este tipo de materia.

8. Puesta en valor del proyecto

8.1 Toma de Decisiones

El análisis de los datos de la NBA, especialmente relacionados con el porcentaje de triples de los jugadores pivot a lo largo del tiempo, puede proporcionar información valiosa para entrenadores, directores técnicos y analistas de equipos.

Estos datos pueden ayudar en la toma de decisiones informadas sobre estrategias de juego, selección de jugadores y desarrollo de habilidades.

8.2 Optimización de Rendimiento

Comprender las tendencias y patrones en el porcentaje de triples de los jugadores pivot puede ayudar a los equipos a identificar áreas de mejora en su juego. Los análisis detallados podrían revelar patrones de rendimiento en ciertos escenarios de juego, lo que permitiría a los equipos enfocar sus entrenamientos en áreas específicas para mejorar su eficacia en la cancha.

8.3 Análisis de Tendencias y Evolución

El proyecto puede ayudar a los seguidores de la NBA y los analistas deportivos a comprender la evolución del juego a lo largo del tiempo. Identificar cómo ha cambiado el porcentaje de triples de los jugadores pivot a lo largo de las temporadas puede proporcionar información sobre las tendencias en las estrategias de juego, el desarrollo de habilidades de los jugadores y los cambios en las reglas del juego.

8.4 Aplicaciones en Entrenamiento y Desarrollo

Los datos y análisis obtenidos pueden ser útiles en el desarrollo de programas de entrenamiento y desarrollo de jugadores. Identificar las habilidades específicas que contribuyen al éxito en el porcentaje de triples puede informar la creación de programas de entrenamiento personalizados para mejorar el rendimiento de los jugadores en esta área.

8.5 Generación de Contenido y Engagement

La información obtenida de este proyecto puede ser utilizada para generar contenido relevante y atractivo para los aficionados al baloncesto y seguidores de la NBA. Los análisis y visualizaciones pueden alimentar debates interesantes sobre estrategias de juego, desempeño de jugadores y evolución del deporte, lo que puede aumentar el engagement de la audiencia en plataformas digitales y redes sociales, así como el engagement dentro de esta liga.

Este tipo de estudio no solo podría proporcionar información valiosa sobre el porcentaje de triples de los jugadores pivot en la NBA, sino que también tiene el potencial de influir en la toma de decisiones en el ámbito deportivo, mejorar el rendimiento de los equipos y jugadores, y generar interés y compromiso entre los seguidores del baloncesto.

9. Contribuciones

El trabajo se realizó de manera individual por lo tanto todos los apartados del estudio los realizó el autor del proyecto, Marc Tarín Vilar.

9.1 Conceptualización del Caso de Uso

- Conceptualizar el caso de uso y definió los objetivos del proyecto.
- Proporcionar orientación y asesoramiento en la conceptualización del caso de uso.

9.2 Extracción de Datos

- Búsqueda y extracción de datos del repositorio de Kaggle.
- Proporcionar scripts y herramientas para la extracción eficiente de datos.

9.3 Plataforma Tecnológica

- Desarrollar la plataforma tecnológica utilizada en el proyecto.
- Configurar y mantener la infraestructura de la plataforma tecnológica.

9.4 Tratamiento y Limpieza de Datos

- Realizar la limpieza inicial de los datos y el preprocesamiento inicial.
- Implementar técnicas avanzadas de limpieza de datos para abordar problemas específicos en los conjuntos de datos.

10. Bibliografía

Pandas Development Team. (2023). Pandas user guide (Version 1.3.3) [Documentación oficial]. Recuperado de https://pandas.pydata.org/docs/user_guide/index.html

NumPy Contributors. (2023). NumPy User Guide: Basics of NumPy indexing and slicing (Version estable). Recuperado de https://numpy.org/doc/stable/user/basics.indexing.html

Matplotlib Development Team. (2023). Matplotlib User Guide: Quick Start (Versión estable). Recuperado de https://matplotlib.org/stable/users/explain/quick_start.html

Kaggle. (s.f.). Datos para el estudio. Recuperado de https://www.kaggle.com/

KSchool. (s.f.). Información extraída de las clases impartidas. Recuperado de https://kschool.com/

Amat Rodrigo, J. (2020). Random Forest con Python. Recuperado de https://cienciadedatos.net/documentos/random-forest-con-python

Naren8520. (s.f.). Pronóstico Exponenciales: Método Holt-Winters y Modelo de Tendencia Amortiguada [Repositorio de GitHub]. Recuperado de https://github.com/Naren8520/Pronostico-Exponenciales-Metodo-Holt-Winters-y-Modelo-de-Tendencia-Amortiguada-con-RStudio

Amat Rodrigo, J. (2017). Métodos de regresión no lineal: regresión polinómica, regression splines, smooth splines y GAMs. Recuperado de https://cienciadedatos.net/documentos/32_metodos_de_regresion_no_lineal_polinomica_splines_gams

Amat Rodrigo, J., & Escobar Ortiz, J. (2023). Modelos ARIMA y SARIMAX con Python. Recuperado de https://cienciadedatos.net/documentos/py51-modelos-arima-sarimax-python