Universidad de Valladolid

Trabajo Fin de Grado

FACULTAD DE CIENCIAS

Grado en Estadística

Evaluación de la eficiencia de los jugadores de la NBA utilizando Análisis Envolvente de Datos

Autor:

D. Pablo Martín Villares

Tutores:

D. Jesús A. Tapia García

D. Bonifacio Salvador González

Agradecimientos

Transmitir mi más sincero agradecimiento a todos aquellos que me han ayudado a lo largo de esta etapa y han colaborado en esta investigación.

En primer lugar, a mis tutores, Jesús y Bonifacio, por su ayuda en la planificación, organización y resolución de este Trabajo de Fin de Grado.

En segundo lugar, a mi familia, mi madre Isabel, mi padre Alberto, mi hermana María y mi sobrino Lucas. A mis amigos y a mi pareja Raquel que han estado a lo largo de toda mi carrera apoyándome en todo momento y animándome a seguir adelante.

También, expresar mi más sentido agradecimiento a la Universidad de Valladolid por acogerme dentro de sus aulas y hacerme sentir como en casa. Después de este período de investigación escribo este apartado de agradecimientos para finalizar mi TFG. Sin duda, ha sido un período de aprendizaje científico y personal.

Desarrollar este estudio ha tenido un gran impacto en mi persona y es por eso que me gustaría agradecer a todas aquellas personas que me han apoyado durante este proceso.

A todos ellos, mil gracias

Índice general

Página

Índice de figuras

Página

Índice de tablas

Página

CAPÍTULO 1
Introducción

La National Basketball Association, más conocida por sus siglas NBA, es una liga de baloncesto profesional que se disputa en Estados Unidos. Es una entidad privada, por lo que no depende oficialmente de la Federación Estadounidense de Baloncesto.

El 6 de junio de 1946 se creó bajo el nombre de Basketball Association of America (BAA), más tarde renombrada como NBA, cuando se fusionan las ligas profesionales ya existentes, National Basketball League (NBL), fundada en 1937 y la Basketball Association of America (BAA), creada en 1946.

En su primera temporada, la 1946-47, se disputó con 11 franquicias: Los Boston Celtics, Philadelphia Warriors, New York Knicks, Washington Capitols, Providence Steamrollers, Toronto Huskies, Chicago Stags, St. Louis Bombers, Cleveland Rebels, Detroit Falcons y Pittsburgh Ironmen. Destacando a los Boston Celtics y New York Knicks como únicas franquicias que se mantienen desde aquella temporada hasta nuestra actualidad sin haber cambiado sus colores, nombre, ni ciudad. Desde la temporada 2004-05 la NBA cuenta con 30 franquicias, las cuales están divididos en dos conferencias: la este y la oeste, cada una compuesta por 15 franquicias, a su vez. Y estas conferencias se dividen en 3 divisiones de 5 franquicias cada una.

CONFERENCIA ESTE:

- División atlántica: New York Knicks, Boston Celtics, New Jersey Nets, Philadelphia 76ers y Toronto Raptors
- División central: Chicago Bulls, Cleveland Cavaliers, Detroit Pistons, Indiana Pacers y Milwaukee Bucks
- División sudeste: Atlanta Hawks, Charlotte Bobcats, Miami Heat, Orlando Magic y Washington Wizards

CONFERENCIA OESTE:

- División sudoeste: Dallas Mavericks, Houston Rockets, Memphis Grizzlies, New Orleans Hornets y San Antonio Spurs.
- División noroeste: Denver Nuggets, Minnesota Timberwolves, Portland Trail Blazers, Seattle Supersonics y Utah Jazz.
- División pacífica: Golden State Warriors, Phoenix Suns, Sacramento Kings, Los Angeles Clippers y Los Angeles Lakers.

Figura 1.1: Mapa de Estados Unidos dividido por conferencias

Una temporada normal de la NBA, se divide en la temporada regular y los playoffs. La primera parte es en la que los 30 equipos intentan llegar al mejor balance posible en el ratio victorias-derrotas. Cada equipo disputará 82 partidos (esta temporada 2020-2021 solo se disputarás 72 partidos por la situación del COVID-19).

De los 15 equipos de cada conferencia, los 8 mejores con mejor balance son quienes jugaran la post-temporada, por lo tanto, son 16 equipos los que juegan playoffs y 14 los que ya se ponen a preparar en la siguiente temporada.

La temporada regular suele comenzar en la segunda quincena Octubre y finaliza sobre los primeros días de abril, es decir, en 5 meses y medio juegan 82 partidos, una cosa que sobre todo en este último tiempo se cuestionó mucho por la carga física que esto requiere, pues los equipos están jugando cada día de por medio, o a veces dos partidos en dos días.

La duración de un partido de baloncesto en la NBA está establecido en 4 parciales o cuartos de 12 minutos cada uno, con un descanso tras los dos primeros. Mientras el balón no está en juego el tiempo se para, por lo que la duración real de un partido de baloncesto es indefinido. A eso hay que añadir que, si al terminar el encuentro hubiera un empate, se disputa un tiempo añadido de 5 minutos. Las reglas del baloncesto establecen que se jugarán tantas prórrogas como sean necesarias hasta que no haya empate al finalizar el tiempo de juego.

Los principales objetivos de este TFG son lograr obtener una base de datos con la que sea posible medir la eficiencia de los jugadores de la NBA y lograr establecer un ranking entre los jugadores para una o más temporadas y uno o más equipos. La principal metodología utilizada para lograr estos objetivos es el Análisis Envolvente de Datos (DEA).

CAPÍTULO 2

Análisis envolvente de datos

2.1. Introducción

El Análisis Envolvente de Datos (DEA: Data Envelopment Analysis) es una metodología basada en modelos de programación lineal, propuesta por primera vez en 1978 por Charnes, Cooper y Rhodes, para estudiar la eficiencia relativa de una serie de cada una de las unidades de decisión.

DEA nace como una técnica para evaluar la eficiencia de una serie de elementos, denominados normalmente Unidades de toma de Decisión, en adelante DMU (Decision Making Unit), utilizándose para dicha evaluación múltiples entradas y salidas para cada una de las DMUs consideradas. Dichas DMUs deben ser comparables, es decir, tanto sus entradas como sus salidas deben ser medibles en unidades homogéneas para todas ellas.

DEA es una técnica de programación matemática que permite la construcción de una superficie envolvente o frontera eficiente, a partir de los datos disponibles del conjunto de DMUs, de forma que las unidades que determinan la frontera serán las unidades eficientes y aquellas que no pertenecen a la misma serán unidades ineficientes.

Farrell supuso que la frontera de producción era conocida. Sin embargo, en la práctica esto no es así y, por tanto, es necesario estimarla.

Los métodos de estimación para construir la frontera de producción pueden clasificarse en métodos paramétricos o no-paramétricos, en función de que se requiera o no especificar una forma funcional que relacione los inputs con los outputs. A su vez, pueden emplearse métodos estadísticos o no para estimar la frontera que, en última instancia, puede ser especificada como estocástica (aleatoria) o determinista.

El Análisis Envolvente de Datos es una técnica no-paramétrica, determinista, que recurre a la programación matemática.

La metodología DEA requiere lo primero de la identificación del conjunto de posibilidades de producción del problema, esto es, definir los posibles puntos de operación. Las dos alternativas más frecuentes son las tecnologías denominadas retornos de escala constante y retornos de escala variable. La tecnología constante considera como unidad admisible dentro del problema cualquier combinación lineal de las DMUs observadas, mientras que en la tecnología variables sólo se consideran admisibles las combinaciones lineales convexas.

Un segundo paso consiste en la selección del modelo DEA adecuado al problema a resolver. Existen multitud de modelos DEA, todos ellos con el mismo objetivo: encontrar un punto admisible de mayor productividad con el que puedan compararse las diferentes DMUs del problema. De esta forma, dada una cierta DMU_0, se formula un modelo de programación lineal que busca una combinación lineal de las DMUs existentes, definiendo de esta forma un conjunto de puntos tecnológicamente admisibles que usan menos salidas que DMU_0 y/o produce más salidas que la DMU_0.

Si ningún punto es mejor que DMU_0, entonces se le denomina unidad eficiente. Cuando una unidad es mejor que otra significa que tiene menos entradas y/o más salidas que la unidad considerada. Si por el contrario la DMU_0 no es eficiente, el modelo la proyecta sobre la frontera eficiente y mide la eficiencia de la DMU_0 en términos de la disminución del consumo de las salidas totales e incremento en la producción de salidas que se tendría que realizar para que fuese eficiente. Hay diferentes maneras de realizar la proyección y medición de la distancia entre la DMU_0 y el punto sobre el que se proyecta. Así, la orientación de entrada consiste en la reducción tanto como sea posible de todos los recursos de forma equi-proporcional sin reducir las salidas. Por otra parte, la orientación de salida consiste en incrementar tanto como sea posible los productos de forma equi-proporcional sin un incremento de los salidas. Existen modelos no radiales con orientación de entrada o de salida así como modelos con orientación de entrada-salida que intentan conseguir tanto reducción de recursos como incrementos de productos.

2.2. Conceptos básicos

Vamos ahora a definir los conceptos básicos en los que se basan los modelos DEA.

Unidad productiva

Cualquier organización que genere productos o entradas consumiendo ciertos recursos o salidas, y teniendo la capacidad de poder modificar tanto el nivel de los recursos consumidos como el de la producción creada. Por tener esta capacidad de decisión, también se usa para este concepto, Unidad de Toma de Decisiones, DMU (Decision Making Unit).

Productividad

La productividad de una determinada unidad se define como la relación existente entre los resultados que obtiene y los recursos empleados para su producción. Es una forma de medir cómo de bien se están aprovechando dichos recursos. Para el caso de una sola salida y una sola entrada:

$$Productividad = \frac{\text{Producción creada}}{\text{Recurso consumido}} = \frac{Salida}{Entrada} \tag{2.1}$$

Para el caso en el que hay m entradas y s salidas, definiendo x_{ij} como la cantidad de entrada o recurso 'i' utilizado por la unidad 'j', y y_{kj} como a la cantidad de salida o producto 'k' que produce la misma unidad, y u_{ij} y v_{kj} respectivamente como los pesos que eliminan la dimensión de las entradas y salidas, se obtiene:

$$Entrada_j = \sum_{i=1}^{m} u_{ij} x_{ij} \qquad\qquad Salida_j = \sum_{k=1}^{s} v_{kj} y_{kj} \tag{2.2}$$

Redefiniendo la productividad como:

$$Productividad_j = \frac{\sum_{k=1}^{s} v_{kj} y_{kj}}{\sum_{i=1}^{m} u_{ij} x_{ij}} \tag{2.3}$$

Eficiencia relativa

La eficiencia vendría a ser la capacidad que tiene una unidad para obtener la salida máxima a partir de un conjunto de entradas, se obtiene al comparar el valor observado de cada unidad con el valor de la unidad de máxima productividad. Siendo la formula de la eficiencia:

$$Eficiencia_j = \frac{Productividad_j}{Productividad_{max}} = \frac{Salida_j / Entrada_j}{Salida_{max} / Entrada_{max}} \tag{2.4}$$

Siendo 'j' el subíndice de la unidad a estudiar y 'max' el subíndice de la unidad de máxima productividad.

Se hablará de eficiencia global cuando la unidad que se escoge como referencia con mayor productividad esta entre todas las que se disponen para el estudio. Mientras que la eficiencia técnica será cuando para dicha unidad solo se tienen en cuenta las unidades del mismo tamaño que la de la unidad que se quiere calcular.

2.3. Concepto intuitivo de envolvente

Para establecer el concepto de envolvente de una manera intuitiva se plantean tres casos con sus correspondientes ejemplos.

- Caso 1. Un Input y un Output.

Partiendo de un conjunto de n unidades, cada una de las cuales produce un único output (y) usando un único input (x), puede obtenerse un indicador de eficiencia para cada una de las n unidades consideradas (cociente entre el output y el input) y realizar, a partir de las puntuaciones obtenidas, una clasificación de eficiencia. Siendo la unidad más eficiente aquella cuyo cociente sea mayor.

DMU	**Input (x)**	**Output(y)**
A	1	2
B	3	3
C	2	2
D	2	1

Tabla 2.1: Datos caso 1 Input y 1 Output.

Una manera habitual de medir la eficiencia es a través de la productividad, es decir, rentabilidad de la inversión. Así, la DMU A sería la más eficiente, puesto que la razón salidas por entradas es la más alta.

La eficiencia de cada una de las DMUs puede ser evaluada en relación con la DMU A, de tal forma que esta medida de eficiencia relativa tomaría valores entre 0 y 1, tal y como muestra la siguiente tabla:

DMU	**Output/Input**	**Evaluación respecto DMU A**	**Eficiencia relativa**
A	2	2/2	1
B	1	1/2	0,5
C	1	1/2	0,5
D	0,5	0,5/2	0,25

Tabla 2.2: Eficiencia relativa

Ejemplo 2. *En la Figura 2.1 se han representado los datos de la tabla 2.1 relativos a las cuatro filiales de un grupo empresarial.*

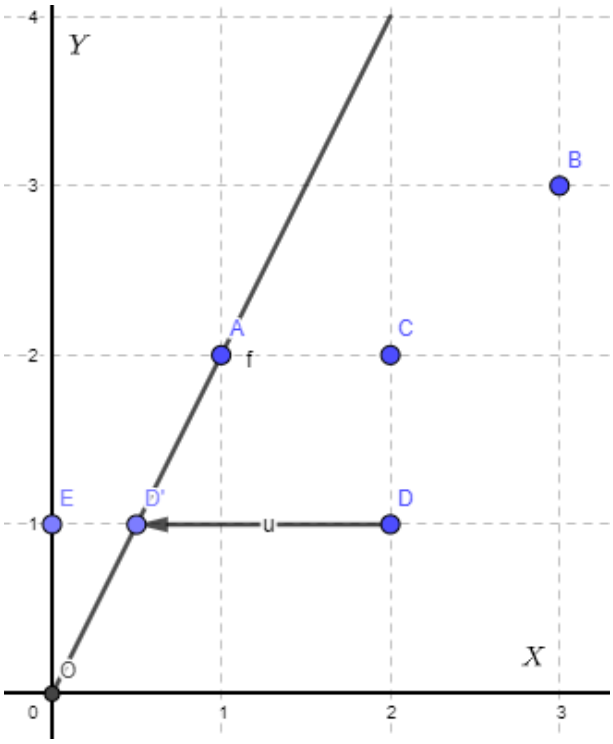

Figura 2.1: Frontera eficiente para 1 Input y 1 Output

Suponiendo rendimientos constantes a escala, la recta que parte del origen O y pasa por el punto que representa a la DMU D constituye el mayor rendimiento que puede ser alcanzado. Esta recta determina la frontera eficiente y, como se comentó anteriormente, la eficiencia (relativa) del resto de DMUs vendrá dada por la desviación respecto a la misma.

Siguiendo con el ejemplo, y como puede verse en la Figura 2.1, la eficiencia técnica de la DMU D vendrá dada por:

$$\text{Eficiencia Técnica de D} = ET_D = \frac{ED'}{ED} \tag{2.5}$$

Es decir, la eficiencia (relativa) técnica de la DMU D se obtiene como el cociente entre la distancia euclídea entre ED' y la distancia ED. Así, la eficiencia técnica de la unidad evaluada será:

$$ET_D = \frac{ED'}{ED} = \frac{\sqrt{(0,5-0)^2+(1-1)^2}}{\sqrt{(2-0)^2+(1-1)^2}} = \frac{0,5}{2} = 0,25 \text{ (o el } 25\,\%) \tag{2.6}$$

Que como se puede comprobar es la misma que la reflejada en la tabla 2.2. También podría decirse que la ineficiencia técnica de la DMU D es del 75 %.

- Caso 2. Un Input y dos Output.

Supóngase que el conjunto de 5 unidades producen dos outputs (y_1, y_2) empleando un único input (x). En este caso podría considerarse, para cada unidad, el output producido por unidad input, es decir, los cocientes: $y_1/x, y_2/x$.

Para este caso la evaluación de la eficiencia resulta más compleja, dado que una unidad puede tener el mayor valor en la relación y_1/x y no suceder en la relación y_2/x, en la que el mayor valor lo presenta una unidad distinta. En el siguiente ejemplo se planteara como obtener la puntuación de eficiencia técnica de cada una de las n unidades consideradas

Los datos de los que se disponen son los que se muestran en la siguiente tabla:

DMU (unidad)	**Input (x)**	**Output1 (y_1)**	**Output2 (y_2)**
A	15	30	45
B	13	26	52
C	12	12	36
D	6	18	12
E	8	32	8
F	11	22	22

Tabla 2.3: Datos para el caso de 1 Input y 2 Outputs

Tal y como se ha comentado, pueden considerarse dos índices: (y_1/x) y (y_2/x). Los resultados obtenidos al generar estos índices son los siguientes:

DMU (unidad)	**Índice 1** (y_1/x)	**(Índice 2** $y_2/x)$
A	2	3
B	2	4
C	1	3
D	3	2
E	4	1
F	2	2

Tabla 2.4: Output obtenido por unidad de Input consumida

Como puede comprobarse en la tabla 2.4, la DMU E es el que obtiene el mayor rendimiento en lo referente al índice 1, pero en lo relativo al índice 2 por ejemplo el mejor desempeño corresponde a la DMU B, por lo que la frontera eficiente será la que una estos dos puntos.Ninguna DMU situada sobre la frontera eficiente puede, dado el nivel de inputs, mejorar uno de sus outputs sin empeorar el otro.

En la Figura 2.2 se han representado los datos contenidos en la tabla anterior. La DMU B es el más eficiente en la obtención del output y_2, en tanto que la DMU E lo es en el output y_1.

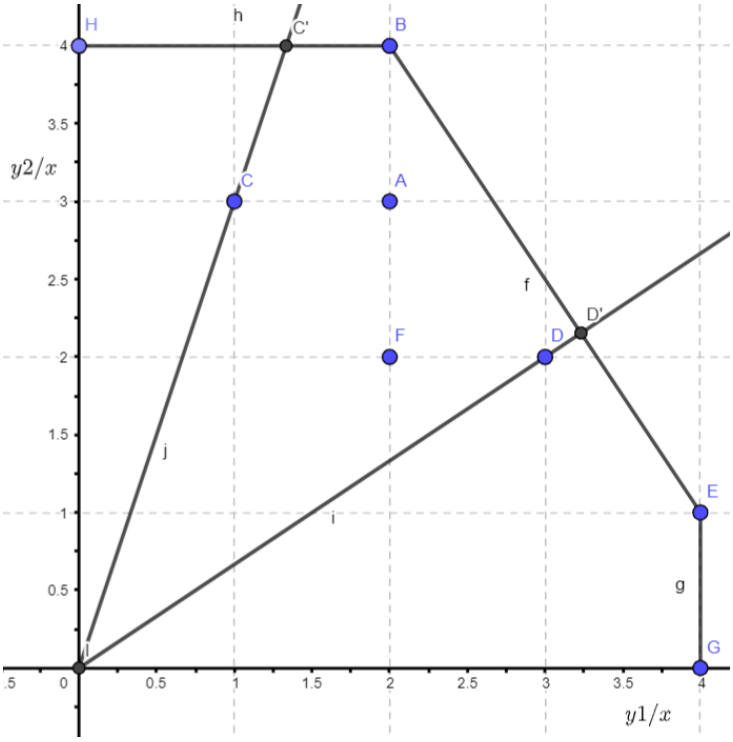

Figura 2.2: Frontera eficiente para 1 Input y 2 Outputs

Las DMUs B y E son eficientes técnicamente, es decir: $ET_B = 1$ y $ET_E = 1$.

Dibujando una línea que una todas las unidades eficientes y prolongándola de forma paralela a los ejes se obtiene la frontera de posibilidades de producción (FPP), es decir, el límite entre los niveles de producción alcanzables e inalcanzables.

Las DMUs (unidades) que permanecen por debajo de la frontera eficiente, que envuelve a aquellos, son calificados como DMUs ineficientes técnicamente. La eficiencia relativa de estas unidades ineficientes puede obtenerse como la relación entre la longitud de la línea desde el origen hasta la unidad considerada y la longitud de la línea que une el origen con el punto proyectado sobre la frontera eficiente. Así, por ejemplo, en el caso de la DMU D se tendría:

$$\text{Eficiencia Técnica de DMU D} = ET_D = \frac{OD}{OD'} \tag{2.7}$$

Es decir, la eficiencia técnica de la DMU D es el cociente entre la distancia del punto O al punto D y la distancia del punto O al punto D' . Así, para calcular la eficiencia de D es necesario conocer las coordenadas del punto D', que se corresponderá con la intersección entre la recta que pasa por los puntos A y E y la recta que pasa por los puntos O y D.

$$ET_D = \frac{OD}{OD'} = \frac{\sqrt{(3-0)^2 + (2-0)^2}}{\sqrt{(3,23-0)^2 + (2,15-0)^2}} = 0,802373507 \tag{2.8}$$

La eficiencia de la DMU D es del 80,2 % o, de otra forma, su ineficiencia es del 19,8 %; lo que equivale a decir que la DMU D, para ser eficiente debería incrementar un 19,8 % las salidas. Operando de forma análoga a como se ha hecho con la unidad D, se obtendrían las puntuaciones de eficiencia técnica (relativa) del resto de DMUs.

DMU (unidad)	**Eficiencia técnica (relativa) en %**
A	61,22
B	100
C	50,38
D	80,23
E	100
F	36,44

Tabla 2.5: Eficiencia relativa para el caso de 1 Input y 2 Output

2.4. Definición del conjunto de posibilidades de producción

La medida de la eficiencia de una unidad mediante la técnica DEA implica dos pasos básicos:

1. La construcción del conjunto de posibilidades de producción.

2. La estimación del máximo aumento posible del output o de la máxima disminución posible de los inputs de la unidad dentro del conjuntos de posibilidades de producción.

El interés ahora se centra en determinar el conjunto de procesos productivos que se consideran factibles, a partir de los datos observados. Para ello la forma más fácil de describir los planes de producción posibles es enumerar todas las combinaciones de factores y de productos tecnológicamente factibles. El conjunto de estas combinaciones de denominará conjunto de producción.

Así, el conjunto de posibilidades de producción (CPP) puede definirse como el conjunto de procesos productivos tecnológicamente factibles. Puesto que la tecnología no es conocida, la construcción del conjunto de posibilidades de producción se realizará a partir de las combinaciones input-output observadas.

Suponiendo un proceso productivo que emplea un conjunto de inputs x para producir un conjunto de outputs y.

Las características del conjunto de procesos productivos que definen el CPP son:

1. Es tecnológicamente posible no producir ni inputs ni outputs.

2. Si dos procesos productivos pertenecen al CPP, todas sus combinaciones lineales convexas también pertenecen al CPP.

3. Una unidad productiva es capaz de producir la misma cantidad de output utilizando una cantidad mayor de cualquier input. Es decir, es posible desechar el exceso de inputs sin coste.

4. Una unidad productiva es capaz de producir una cantidad menor de cualquier output utilizando las mismas cantidades de input.

5. Es posible reescalar la actividad de cualquier proceso productivo perteneciente a P.

2.5. Caracterización de los modelos DEA

Los modelos DEA pueden ser clasificados en función de:

1. El tipo de medida de eficiencia que proporcionan: modelos radiales y no radiales.

2. La orientación del modelo: input orientado, output orientado o input-output orientado.

3. La tipología de los rendimientos a escala que tiene la tecnología de producción, entendida ésta como la forma en que los inputs son combinados para obtener un conjunto de outputs, de tal forma que esa combinación de factores puede tener rendimientos a escala: constantes o variables.

Respecto al primer punto, los modelos DEA que se van a tratar, proporcionan medidas de eficiencia de tipo radial (proporcional). Los otros dos puntos son explicados a continuación.

2.6. Orientación del modelo

La eficiencia puede ser caracterizada con relación a dos orientaciones básicas, pudiendo hacer referencia a modelos:

1. Input orientados: buscan, dado el nivel de outputs, la máxima reducción proporcional en el vector de inputs mientras permanece en la frontera de posibilidades de producción. Una unidad no es eficiente si es posible disminuir cualquier input sin alterar sus outputs.

2. Output orientados: buscan, dado el nivel de inputs, el máximo incremento proporcional de los outputs permaneciendo dentro de la frontera de posibilidades de producción. En este sentido una unidad no puede ser caracterizada como eficiente si es posible incrementar cualquier output sin incrementar ningún input y sin disminuir ningún otro output.

Teniendo en cuenta las orientaciones definidas, una unidad será considerada eficiente si, y solo si, no es posible incrementar las cantidades de output manteniendo fijas las cantidades de inputs utilizadas ni es posible disminuir las cantidades de inputs empleadas sin alterar las cantidades de outputs obtenidas.

En la Figura 2.3 se ha representado, el caso de un único input y un único output, y en ella puede verse como la unidad A es ineficiente, se sitúa por debajo de la frontera.

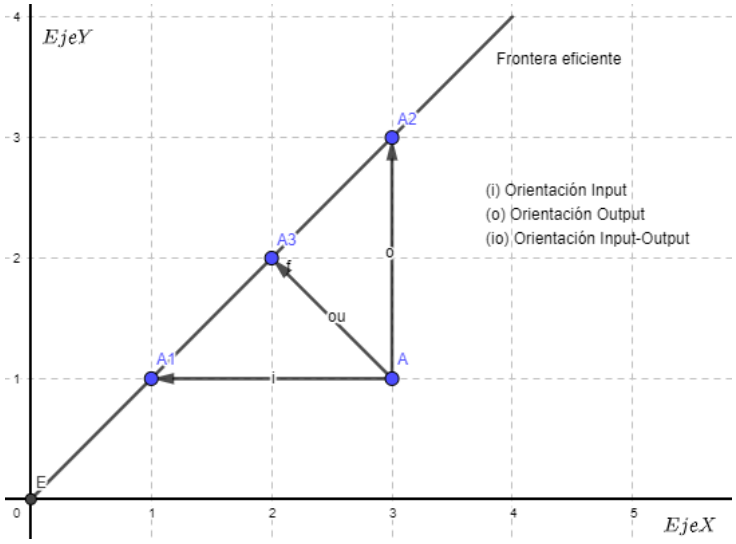

Figura 2.3: Orientaciones en modelos DEA

Desde el punto de vista de un modelo input orientado, la unidad A podría reducir la cantidad de input x y seguir produciendo la misma cantidad de output y, es decir, la unidad A debería tomar como referencia la mejor práctica de la unidad A_1. De igual forma, al considerar la evaluación de la eficiencia a través de modelos outputs orientados, la unidad A sería calificada como ineficiente. Esta unidad podría, consumiendo la misma cantidad de input, producir una mayor cantidad de output.

Como puede observarse en la Figura 2.3 cabe la posibilidad de considerar una tercera opción, correspondiente a los denominados modelos input-output orientados (también llamados no orien-

tados), que buscan simultáneamente la reducción input y el aumento output equiproporcional. En este caso se mide la "distancia hacia la frontera" que representa a la unidad evaluada.

2.7. Tipología de los rendimientos a escala

Los rendimientos a escala, que indican los aumentos de la producción que son el resultado del incremento de los factores de producción en el mismo porcentaje, pueden ser constantes, crecientes o decrecientes:

1. Rendimientos constantes a escala: cuando el incremento porcentual del Output es igual al incremento porcentual de los recursos productivos.
2. Rendimientos crecientes a escala: se dice que la tecnología exhibe este tipo de rendimientos cuando el incremento porcentual del output es mayor que el incremento porcentual de los factores.
3. Rendimientos decrecientes a escala: cuando el incremento porcentual del output es menor que el incremento porcentual de los inputs.

2.8. Modelos DEA

Los modelos DEA se distinguen por la medida de la eficiencia, la orientación del modelo para calcularla y la tipología de los rendimientos a escala que caracterizan la frontera eficiente.

2.8.1. Modelo CCR

El modelo DEA-CCR, denominado así por haber sido desarrollado por Charnes, Cooper y Rhodes. El modelo proporciona medidas de eficiencia radiales, pudiendo ser input u output orientadas y supone convexidad, fuerte eliminación gratuita de inputs y outputs y rendimientos constantes a escala.

El modelo DEA-CCR puede escribirse, en términos generales, de tres formas distintas: fraccional, multiplicativa y envolvente. Aunque solo se verá el modelo multiplicativo.

El modelo DEA-CCR Input orientado en forma de cociente puede ser linealizado siguiendo la transformación lineal de Charnes y Cooper, que selecciona la solución (μ, δ) para que $\sum_{i=1}^{m} \delta x_{io}$. Realizando dicho cambio de variable se obtiene el modelo en forma multiplicativa:

$$\begin{aligned} &\underset{\mu,\delta}{\text{Max}} && W_0 = \mu^T y_0 \\ &\text{Sujeto a:} \\ &\delta^T x_0 = 1 \\ &\mu^T Y + -\delta^T X \leq 0 \\ &\mu^T, \delta^T \geq I\varepsilon \end{aligned} \tag{2.9}$$

donde:

1. Y es una matriz de outputs de orden (sxn)

$$\begin{pmatrix} y_{11} & y_{12} & \cdots & y_{1n} \\ y_{21} & y_{22} & \cdots & y_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ y_{s1} & y_{s2} & \cdots & y_{sn} \end{pmatrix}$$

2. y_0 representa el vector Output de la Unidad que está siendo evaluada

3. X es una matriz de inputs de orden (mxn)

$$\begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{pmatrix}$$

4. x_0 representa el vector inputs de la Unidad que está siendo evaluada

5. μ es el vector (sx1) de pesos outputs y δ es el vector (mx1) de pesos inputs

La unidad será calificada de eficiente si $w_0^* = 1$ y existe al menos un óptimo (μ^*, δ^*) con $\mu^* > 0$ y $\delta^* > 0$

2.8.2. Modelo BCC

El modelo DEA-BCC, así denominado por haber sido desarrollado por Banker, Charnes y Cooper. Si el modelo DEA-CCR consideraba rendimientos constantes a escala, el modelo DEA-BCC relaja este supuesto, que en gran parte de las ocasiones resulta excesivamente restrictivo y por tanto irreal, permitiendo que la tipología de rendimiento a escala que en un momento determinado caracterice la tecnología sea variable, esto es: constante, creciente o decreciente.

De lo dicho puede desprenderse que los fundamentos del modelo DEA-BCC se encuentran en el modelo DEA-CCR, puesto que el primero es una extensión del segundo.

Modelo BCC Input orientado

El modelo DEA-BCC es realmente una extensión del modelo DEA-CCR. Por tanto, su formulación es similar. La diferencia fundamental entre el modelo DEA-CCR y DEA-BCC es que éste introduce el supuesto de rendimientos variables a escala y el anterior considera rendimientos constantes a escala.

El objetivo del modelo DEA-BCC Input orientado es hallar un hiperplano que permaneciendo por encima de todas las unidades, minimice la distancia horizontal desde dicho hiperplano a la unidad.

La forma multiplicativa del modelo, obtenida de nuevo al aplicar la transformación realizada por Charnes y Cooper sobre el modelo fraccional, puede expresarse matricialmente de la siguiente forma:

$$\begin{aligned} &\underset{\mu,\delta,k}{\text{Max}} && W_0 = \mu^T y_0 + k_0 \\ &\text{Sujeto a:} \\ &\delta^T x_0 = 1 \\ &\mu^T Y + k_0 \leq \delta^T X \\ &\mu^T, \delta^T \geq I\varepsilon \\ &k_0 \text{ no restringida} \end{aligned} \tag{2.10}$$

La Unidad evaluada será calificada de eficiente si $w_0^* = 1$ y existe al menos un óptimo (μ^*, δ^*) con $\mu^* > 0 y \delta^* > 0$. El valor k_A^* es usado para identificar el tipo de rendimiento a escala en el que opera localmente la unidad evaluada.

También es posible, a partir de los valores óptimos de los pesos o multiplicadores de inputs y outputs, determinar los porcentajes de contribución input/output.

Si se compara el problema dado en el modelo 2.10 con el modelo 2.9 se puede observar como la definición de la medida de eficiencia bajo el supuesto de rendimiento variables a escala, $W_0 = \mu^T y_0 + k_0$, es similar a aquella que supone rendimientos constantes a escala $W_0 = \mu^T y_0$. La única diferencia entre ambas es que para el segundo caso se le suma un término constante k_0 (que en el supuesto de rendimientos contantes toma el valor cero).

De tal manera que si la solución óptima del modelo 2.10 para la DMU, que se supone eficiente:

- $k_0^* > 0$ para todas las soluciones óptimas, prevalecen rendimientos crecientes a escala.
- $k_0^* = 0$ para cualquier solución óptima, prevalecen rendimientos constantes a escala.
- $k_0^* < 0$ para todas las soluciones óptimas, prevalecen rendimientos decrecientes a escala.

Modelo BCC Output orientado

Si,por el contrario, se quiere evaluar la eficiencia de una unidad desde el punto de vista de la maximización de los outputs, dado el nivel de inputs, debe recurrirse al modelo DEA-BCC output orientado.

Un cambio en la orientación del modelo prácticamente equivale a invertir el cociente entre el output virtual (output total) y el input virtual (input total). Linealizando el modelo DEA-BCC output orientado con forma de cociente se obtiene el modelo en forma multiplicativa:

$$\begin{aligned} &\underset{\mu,\delta,k}{\text{Max}} && W_0 = \delta^T x_0 + k_0 \\ &\text{Sujeto a:} \\ &\mu^T y_0 = 1 \\ &\delta^T X + k_0 \geq \mu^T Y \\ &\mu^T, \delta^T \geq I\varepsilon \\ &k_0 \text{ no restringida} \end{aligned} \tag{2.11}$$

La eficiencia técnica Output pura de la Unidad0 vendrá dada por $\frac{1}{w_0^*}$, de tal forma que ésta será eficiente si $w_0^* = 1$.

Como sucedía con el modelo DEA-BCC Input orientado, el signo que tome k (positivo, negativo o nulo) en la solución óptima indica el tipo de rendimiento a escala para una unidad. Sin embargo, a diferencia del modelo 2.10, en el modelo 2.11 el término constante se encuentra asociado con el valor input y su signo está invertido, de tal forma que:

- $k_0^* > 0$ para todas las soluciones óptimas, prevalecen rendimientos decrecientes a escala.
- $k_0^* = 0$ para cualquier solución óptima, prevalecen rendimientos constantes a escala.
- $k_0^* < 0$ para todas las soluciones óptimas, prevalecen rendimientos crecientes a escala.

2.8.3. Modelo Aditivo

Es un modelo que considera problemas que operan con retornos de escala variables. Fue introducido por primera vez por Charnes y más tarde fue elaborado por Banker.

La diferencia principal del modelo aditivo con los modelos BCC y CCR recae en que se utiliza una métrica rectangular en lugar de la radial utilizada por los anteriores. Ya sea orientación de entrada o salida siempre se maximizan las holguras, y por tanto el modelo aditivo no distingue dichos tipos de orientaciones.

Las unidades eficientes en el modelo BCC resultan eficientes si se utiliza el modelo aditivo y viceversa. Sin embargo, cuando una unidad resulta ineficiente, las fuentes de ineficiencia y su importancia detectadas por el modelo BCC, se diferencian de las encontradas por el modelo aditivo. La razón de ello se debe a la diferente métrica utilizada por ambos métodos para la evaluación de la eficiencia.

Una ventaja de este modelo es la de ser invariante frente a traslaciones. Las traslaciones son necesarias cuando, por ejemplo, alguna de las variables (entradas o salidas) puede tomar valores negativos. En estos casos, resulta conveniente el empleo del modelo aditivo.

El dual de este modelo es el siguiente:

$$\text{Max} \sum_{k=1}^{s} v_{kj} y_{kj} - \sum_{i=1}^{m} u_{ij} x_{ij} + \zeta$$

Sujeto a:

$$\begin{aligned} &\sum_{k=1}^{s} v_{kj} y_{kj} - \sum_{i=1}^{m} u_{ij} x_{ij} + \zeta \leq 0 && j = 1, 2, ..., n \\ &u_{kj} \geq 1 && j = 1, 2, ..., s \\ &v_{kj} \geq 1 && i = 1, 2, ..., m \end{aligned} \tag{2.12}$$

2.9. Análisis en Componentes Principales

El Análisis en componentes Principales(PCA) es una técnica de análisis de datos multivariantes cuyo problema central es la reducción de la dimensionalidad: si es posible describir con precisión los valores de p variables por un pequeño subconjunto $k < p$ de ellas, se habrá reducido la dimensión del problema a costa de una pequeña perdida de información.

El PCA pertenece a la familia de técnicas conocida como *Aprendizaje no supervisado*, en los que el objetivo no es predecir el valor de la respuesta Y no se tiene en cuenta ya que el objetivo no es predecir Y, si no extraer información empleando los predictores, por ejemplo, para identificar subgrupos.

El análisis de componentes principales tiene el siguiente objetivo: dadas n observaciones de p variables, se analiza si es posible representar adecuadamente esta información con un número menor de variables construidas como combinaciones lineales de las originales. Tiene dos utilidades:

- Permite representar óptimamente en un espacio de dimensión pequeña observaciones de un espacio general p-dimensional. En este sentido, componentes principales es el primer paso para identificar las posibles variables latentes, o no observadas que generan los datos.
- Permite transformar las variables originales, en general correladas, en nuevas variables incorreladas, facilitando la interpretación de los datos

Supóngase que existe una muestra con n individuos cada uno con p variables $(X_1, X_2, \ldots, X_p)$, es decir, con p dimensiones. El PCA permite encontrar un número de factores k, con $(k < p)$, que explican aproximadamente lo mismo que las p variables originales. Donde antes se necesitaban p valores para caracterizar a cada individuo, ahora bastan k valores. Cada una de estas k nuevas variables recibe el nombre de componente principal.

2.9.1. Cálculo de las componentes principales

En el análisis de componentes principales se dispone de una muestra de tamaño n acerca de p variables $X_1, X_2, ..., X_p$ inicialmente correladas, para después obtener un número k de variables incorreladas, con $k <= p$, $Z_1, Z_2, ..., Z_k$, las cuales son combinación lineal de las variables iniciales y que expliquen la mayor parte o la totalidad de su variabilidad.

La primera componente principal, al igual que todas las restantes, es la combinación lineal normalizada de dichas variables. Esta se diferencia de las demás por ser la que tiene mayor varianza:

$$Z_{1i} = \phi_{11}X_{1i} + \phi_{21}X_{2i} + \cdots + \phi_{p1}X_{pi} \tag{2.13}$$

La ecuación puede expresarse de forma abreviada como $Z_1 = X\phi_1$ y matricialmente como:

$$\begin{pmatrix} Z_{11} \\ Z_{12} \\ \vdots \\ Z_{1n} \end{pmatrix} = \begin{pmatrix} X_{11} & X_{21} & \cdots & X_{p1} \\ X_{12} & X_{22} & \cdots & X_{p2} \\ \vdots & \vdots & \ddots & \vdots \\ X_{1n} & X_{2n} & \cdots & X_{pn} \end{pmatrix} \begin{pmatrix} \phi_{11} \\ \phi_{12} \\ \vdots \\ \phi_{1p} \end{pmatrix} \tag{2.14}$$

Tanto si las X están tipificadas, como si están expresadas como desviaciones respecto de su media muestral, la media de Z_1 tiene que ser cero, es decir:

$$\sum_{j=1}^{n} Z_{1j} = 0 \tag{2.15}$$

La varianza de Z_1, que se quiere maximizar, será:

$$Var(Z_1) = \frac{1}{n} \sum_{j=1}^{n} Z_{1j}^2 \tag{2.16}$$

La primera componente Z_1 se obtiene de forma que su varianza sea máxima, sujeta a la restricción de que la suma de los pesos ϕ al cuadrado sea igual a la unidad, es decir, la variable de los pesos ϕ se encuentra normalizada:

$$\sum_{j=1}^{p} \phi_{j1}^2 = 1 \tag{2.17}$$

Los términos $\phi_{11}, \ldots, \phi_{1p}$ se denominan loadings y son los que definen cada componente. ϕ_{11} es el loading de la primera componente principal de la variable X1. Los loadings pueden interpretarse como el peso que aporta cada variable en cada componente y esto nos permite conocer que información recoge cada una de las componentes.

Una vez calculada la primera componente (Z_1) se calcula la segunda componente (Z_2) repitiendo los mismos pasos, y añadiendo la condición de que la combinación lineal de la segunda no pude estar correlacionada con la primera componente. El proceso se repetirá hasta calcular todas las posibles componentes o hasta que se decida detener el proceso. El orden de importancia de las componentes viene dado por la varianza que recoge cada una de ellas.

CAPÍTULO 3

Base de datos de jugadores de la NBA

Para realizar el análisis de eficiencia de los diferentes jugadores de la NBA en cada equipo se precisa disponer de datos input y output de cada uno de los jugadores.

Para ello se cuenta con el fichero de datos 'boxscores_basic.csv' generado por Hector Sanz Niño en su TFG Desarrollo de un Data Lake para la gestión de datos de estadísticos de la competición NBA (National Basketball Association)[7], que contiene las estadísticas de cada partido desde 1990 hasta 2020 para todos los jugadores. Además, se necesita obtener los salarios de cada jugador para cada temporada de dichas temporadas.

3.1. Fichero de datos BoxScores

El fichero 'boxscores_basic.csv' contiene un total de 5.754.369 filas con la información correspondiente a las estadísticas de cada jugador para todos los partidos desde 1990 hasta 2020. La información viene condensada en las siguientes columnas:

- game_id: Es el identificador del partido, esta compuesto por la fecha en formato YYYYMMDD seguido del identificador del equipo local y del identificador del equipo visitante
- team_id: Es el identificador del equipo al que pertenece el jugador
- box_type: Indica si las estadísticos de la fila hacen referencia al total del partido, al cuarto correspondiente, al primer o segundo tiempo o a alguna de las prorrogas
- date: Es la fecha del partido en formato YYYY-MM-DD
- ishome: Valor booleano que indica True si es el equipo local
- pnum: Es el número del jugador
- player: Nombre y apellido del jugador
- player_href: Enlace HTML que lleva a información específica del jugador
- player_csk: Apellido y nombre separado por una coma
- player_id: Identificador único que hace referencia al jugador
- mp: Minutos jugados
- sp: Segundos jugados
- fg: Total de canastas encestadas

- fga: Total de lanzamientos realizados
- fg_pct: Porcentaje de acierto en tiros de 3 puntos
- fg3: Total de canastas de 3 puntos encestadas
- fg3a: Total de lanzamientos de 3 puntos realizados
- fg3_pct: Porcentaje de acierto en tiros de 3 puntos
- ft: Total de tiros libres encestados
- fta: Total de tiros libres realizados
- ft_pct: Porcentaje de acierto en tiros libres
- orb: Rebotes ofensivos
- drb: Rebotes defensivos
- trb: Total de rebotes
- ast: Asistencias realizadas
- stl: Robos realizados
- blk: Bloqueos realizados
- tov: Perdidas de balones totales
- pf: Faltas personales
- pts: Puntos totales anotados
- plus_minus: Refleja el balance de puntos que consiguió el equipo cuando el jugador estaba en la cancha
- reason: Motivo por el que no pudo jugar el jugador el partido

Dado que el fichero tiene información que para el desarrollo del proyecto no va a ser necesario, lo primero que se hace es realizar una selección de los datos para quedarnos solo con las filas y columnas que nos interesan.

Dado que la variable plus_minus se ha decidido introducir entre los outputs a analizar y que esta variable solo tiene datos a partir de la temporada 1996 el primer filtrado a realizar es quedarnos con las filas mayores a la fecha 24/08/1996.

En el fichero vienen filas con información de las estadísticas que ha tenido cada jugador a lo largo del partido y tambien desglosado en cada cuarto, prorroga o medio tiempo del partido. Para nuestros análisis solo nos interesa la información de las estadísticas obtenidas por cada jugador a lo largo del partido completo, por lo tanto se filtran los datos que contengan el valor box_type = "game-basic".

Por último, solo se necesita la información de los jugadores que hayan disputado el partido, es decir, el que haya jugado al menos un segundo. Por lo tanto, se filtran por $sp > 0$.

Finalmente se eliminan las columnas que no aportan ningún tipo de información, como son *game_id,box_type, ishome, pnum, player_href, player_csk, player_id, reason.* Y se eliminan también las columnas cuya información ya viene reflejada en otras columnas *mp, fg_pct, fg3_pct, ft_pct.*

Al realizar esta acción se trabajara de una forma mas limpia y se conseguirá recudir el número de filas a un total de 617.837 y disminuir el peso del fichero de prácticamente 1 Gb a tan solo 62 MB.

3.2. Fechas de temporadas

En los datos del fichero 'boxscores_basic.csv' se dispone de la información sobre qué fecha se disputó cada partido, pero no se dispone a que temporada pertenece cada partido. Por lo que para ello se decidió crear la función Python *read_calendar()* que se encargase de ello, explicada en la sección read_calendar

```
season,start_season,end_season
1996,31/10/1996,22/08/1997
1997,22/08/1997,10/10/1998
1998,10/10/1998,29/08/1999
1999,29/08/1999,25/08/2000
2000,25/08/2000,22/08/2001
2001,22/08/2001,20/08/2002
2002,20/08/2002,21/08/2003
2003,21/08/2003,24/08/2004
2004,24/08/2004,27/08/2005
2005,27/08/2005,25/08/2006
2006,25/08/2006,22/08/2007
2007,22/08/2007,22/08/2008
2008,22/08/2008,20/08/2009
2009,20/08/2009,21/08/2010
2010,21/08/2010,18/09/2011
2011,18/09/2011,25/08/2012
2012,25/08/2012,24/08/2013
2013,24/08/2013,21/08/2014
2014,21/08/2014,21/08/2015
2015,21/08/2015,22/08/2016
2016,22/08/2016,14/08/2017
2017,14/08/2017,12/08/2018
2018,12/08/2018,17/08/2019
2019,17/08/2019,12/03/2020
```

Figura 3.1: Archivo CSV con las fechas de inicio y fin de las temporadas

3.3. Salario de los jugadores

Para la obtención de los salarios de cada jugador se obtuvo de dos sitios diferentes ya que no se consiguió encontrar un único repositorio que abarcase la información de los salarios desde 1990 hasta 2020.

Para los años correspondientes desde la temporada 1990 hasta la temporada 2017 se obtuvo de la página https://www.kaggle.com/whitefero/nba-player-salary-19902017 un archivo Excel, que posteriormente se transformó a formato CSV en el fichero salaries_1990.csv para poder trabajar con el. El fichero contenía las siguientes columnas:

- Player Name: Nombre y apellidos del jugador.
- Salary: Salario anual del jugador.
- Season: Año de inicio y año de fin de la temporada correspondiente.
- Season Start: Año de inicio de la temporada correspondiente.
- Season End: Año de fin de la temporada correspondiente.
- Team: Siglas del equipo en que competía ese año el jugador
- Full Team Name: Nombre completo del equipo en que competía ese año el jugador

En la siguiente Figura se puede observar un ejemplo de la estructura del Excel Correspondiente.

	A	B	C	D	E	F	G
1	Player Name	Salary	Season	Season Start	Season End	Team	Full Team Name
2	A.C. Green	1.750.000 €	1990-1991	1990	1991	LAL	Los Angeles Lakers
3	A.J. English	275.000 €	1990-1991	1990	1991	WAS	Washington Wizards
4	A.J. Wynder	30.000 €	1990-1991	1990	1991	BOS	Boston Celtics
5	Adrian Caldwell	275.000 €	1990-1991	1990	1991	HOU	Houston Rockets
6	Adrian Dantley	400.000 €	1990-1991	1990	1991	MIL	Milwaukee Bucks
7	Alaa Abdelnaby	395.000 €	1990-1991	1990	1991	POR	Portland Trail Blazers
8	Alan Ogg	130.000 €	1990-1991	1990	1991	MIA	Miami Heat
9	Alec Kessler	1.600.000 €	1990-1991	1990	1991	MIA	Miami Heat
10	Alex English	1.500.000 €	1990-1991	1990	1991	DAL	Dallas Mavericks
11	Alexander Volkov	650.000 €	1990-1991	1990	1991	ATL	Atlanta Hawks
12	Alton Lister	1.700.000 €	1990-1991	1990	1991	GSW	Golden State Warriors
13	Alvin Robertson	870.000 €	1990-1991	1990	1991	MIL	Milwaukee Bucks
14	Andre Turner	110.000 €	1990-1991	1990	1991	PHI	Philadelphia 76ers
15	Andrew Lang	600.000 €	1990-1991	1990	1991	PHO	Phoenix Suns
16	Andy Toolson	120.000 €	1990-1991	1990	1991	UTA	Utah Jazz
17	Anthony Cook	370.000 €	1990-1991	1990	1991	DEN	Denver Nuggets

Figura 3.2: Ejemplo de la estructura del fichero Salaries_1990_2020.csv

Para los últimos 3 años de datos se sacaron los datos de la página web de ESPN, `http://www.espn.com/nba/salaries/_/year/2020/page/1`. En la siguiente figura se puede ver un ejemplo de la pagina que muestra una parte de los datos de 2020.

RK	NAME	TEAM	SALARY
31	D'Angelo Russell, PG	Minnesota Timberwolves	$27,285,000
32	Karl-Anthony Towns, C	Minnesota Timberwolves	$27,285,000
33	Kristaps Porzingis, C	Dallas Mavericks	$27,285,000
34	Otto Porter Jr., SF	Chicago Bulls	$27,250,575
35	Bradley Beal, SG	Washington Wizards	$27,093,019
36	Andre Drummond, C	Cleveland Cavaliers	$27,093,019
37	Anthony Davis, PF	Los Angeles Lakers	$27,093,018
38	Hassan Whiteside, C	Portland Trail Blazers	$27,093,017
39	Jrue Holiday, PG	New Orleans Pelicans	$26,231,111
40	LaMarcus Aldridge, C	San Antonio Spurs	$26,000,000

528 Results 1 of 14

Figura 3.3: Ejemplo de las tablas de la página ESPN

Como para cada año los datos se encuentran en una pagina web diferente, teniendo que reemplazar la parte '/year/2020' de la URL por los diferentes años. Y dentro de cada año existen diferentes paginas, teniendo que cambiar la parte '/page/1' de la URL la parte por las diferentes páginas. Realizar este trabajo a mano se podía hacer eterno y muy costoso, por lo que se desarrolló un proceso Python que va accediendo año a año y página a página hasta completar todas las páginas, en función de las que contenga cada año, y extrae los datos de los salarios a un CSV denonimando salaries_web.csv con el total de los salarios.

Una vez obtenidos ambos ficheros había que proceder a unirlos para poder trabajar de forma más cómoda, para ello se creó la función Python *merge_calendars()*.

La función consiste en la lectura de ambos ficheros, seleccionando solo las columnas del nombre del jugador, el salario, el año de inicio de la temporada y el equipo en el que jugaba

3.4. Selección de Inputs

Para la elección de los Inputs se debe elegir los recursos que necesita o se invierten en un jugador de baloncesto a lo largo de una temporada regular para producir el mayor numero posible de outputs. Los recursos que más valor aportan son el tiempo y el dinero, por lo que finalmente se decidieron escoger los siguientes inputs:

- El tiempo, en segundos, jugado por cada jugador a lo largo de toda la temporada.
- El salario neto, en dolares, que cobró cada jugador en dicha temporada.

3.5. Selección de outputs

Para la selección de los outputs hay que seleccionar las características, productos o servicios que es capaz de producir un jugador a lo largo de una temporada y que mejor representen a dicho jugador.

En la base de datos boxsxores_basic.cav se dispone de los siguientes atributos, *sp, fg, fga, fg3, fg3a, ft, fta, orb, drb, trb, ast, stl, blk, tov, pf, pts, plus_minus*, descriptos en la sección 3.1.

Debido a que se dispone de demasiadas variables, es necesario realizar una reducción de la dimensionalidad para así poder eliminar variables irrelevantes que no aportan información, poder reducir la complejidad del modelo y disminuir el coste computacional.

Para ello se decidió emplear la técnica aprendida en la asignatura de Análisis de Datos, el Análisis en Componentes Principales, esta técnica nos permite disminuir el número de variables disponibles sacrificando para ello la perdida de un porcentaje de la información explicada por las variables.

Para ello se decide quedarse solo con un 90 % de la variabilidad explicada con las 17 variables disponibles. Permitiéndonos reducir los outputs de 17 atributos a 4 dimensiones o outputs, aunque según el caso a analizar podrían ser 3 o 5 dimensiones también.

Por lo tanto ya se dispone de nuestros 2 inputs (Salario y segundos jugados) y nuestros outputs, correspondientes a las dimensiones extraídas mediante el PCA.

CAPÍTULO 4

Programa

4.1. Tecnología Python

En términos técnicos, Python es un lenguaje de programación de alto nivel, orientado a objetos, con una semántica dinámica integrada, principalmente para el desarrollo web y de aplicaciones informáticas.

Python es relativamente simple, por su facilidad de aprendizaje, ya que requiere una sintaxis única que se centra en la legibilidad. Los desarrolladores pueden leer y traducir el código Python mucho más fácilmente que otros lenguajes.

Además, soporta el uso de módulos y paquetes, lo que significa que los programas pueden ser diseñados en un estilo modular y el código puede ser reutilizado en varios proyectos. Una vez se ha desarrollado un módulo o paquete, se puede escalar para su uso en otros proyectos, y es fácil de importar o exportar.

Por otro lado, uno de los beneficios más importantes de Python es que tanto la librería estándar como el intérprete están disponibles gratuitamente, tanto en forma binaria como en forma de fuente. Tampoco hay exclusividad, ya que Python y todas las herramientas necesarias están disponibles en todas las plataformas principales,por lo tanto, es una opción multiplataforma.

4.2. Descripción de las librerias utilizadas

4.2.1. Numpy

NumPy es un paquete de Python que significa “Numerical Python”, es la librería principal para la informática científica, proporciona potentes estructuras de datos, implementando matrices y matrices multidimensionales.

Incorpora una nueva clase de objetos llamados arrays que permite representar colecciones de datos de un mismo tipo en varias dimensiones, y funciones muy eficientes para su manipulación.

4.2.2. Pandas

Pandas es una librería de código abierto de Python que proporciona herramientas de análisis y manipulación de datos de alto rendimiento utilizando sus potentes estructuras de datos. El nombre de Pandas se deriva del término “Panel Data” y es la librería de análisis de datos de Python.

Las principales características de esta librería son:

- Define nuevas estructuras de datos basadas en los arrays de la librería NumPy pero con nuevas funcionalidades.
- Permite leer y escribir fácilmente ficheros en formato CSV, Excel y bases de datos SQL.
- Permite acceder a los datos mediante índices o nombres para filas y columnas.
- Ofrece métodos para reordenar, dividir y combinar conjuntos de datos.
- Realiza todas estas operaciones de manera muy eficiente.

4.2.3. Scipy

Scipy es una biblioteca de código abierto de herramientas y algoritmos matemáticos. Contiene módulos para optimización, álgebra lineal, integración, interpolación, funciones especiales, FFT, procesamiento de señales e imagen, resolución de EDOs y otras tareas relacionadas con la ciencia e ingeniería. Está dirigida al mismo tipo de usuarios que los de aplicaciones como MATLAB, GNU Octave, y Scilab.

En este proyecto se usará esta librería para resolver los problemas de programación lineal que se plantean a la hora de realizar los algoritmos DEA.

4.2.4. Sklearn

Sklearn es una librería de python para Machine Learning y Análisis de Datos. Está basada en NumPy, SciPy y Matplotlib. La ventajas principales de Sklearn son su facilidad de uso y la gran cantidad de técnicas de aprendizaje automático que implementa.

Se usará esta librería para calcular el modelo PCA y extraer las componentes principales necesarias correspondientes a los Outputs del análisis DEA.

4.3. Descripción de los procesos

4.3.1. read_calendar

Dicha función consiste en la lectura de la columna date para todas las filas del archivo 'boxscores_basic.csv'. Tras ordenar las fechas cronológicamente, comprueba para cada partido/fila si la diferencia en días con el siguiente partido/fila es mayor de 90 días. Si es así, se ha encontrado el último partido de una temporada y el primer partido de la siguiente, por lo que se almacena la fecha intermedia de ambas, indicando que todos los partidos inferiores a esa fecha corresponden a una temporada, y todos los partidos superiores a esa fecha corresponden a la siguiente.

4.3.2. princomp

La función recibe un array con n columnas y un porcentaje p con valores entre 0 y 1. Y devuelve un array con las m componentes principales correspondientes al PCA de los datos con al menos un p % de la variabilidad explicada.

4.3.3. read_salary

La función accede a las páginas web de ESPN [4] correspondientes a los años 2019 y 2020, descarga las tablas con los salarios de los jugadores para esos y años y los guarda en un archivo CSV.

4.3.4. reduce_data

Esta función filtra la base de datos boxscore_basic.csv, quedándose solo con los partidos superiores a la fecha 24/08/1996, las filas correspondientes a los partidos completos y las filas en las que los jugadores al menos habían disputado un segundo del partido. Guardando el resultado en un nuevo fichero llamado bs_basic.csv.

4.3.5. filter_data

Esta función realiza un primer filtrado de las estadísticas y salarios de los jugadores en función del equipo temporada o jugador pasados como parámetros, quedándose solo con las filas que contienen dichos valores.

Una vez filtrado, agrupa las estadísticas por jugador y extrae las columnas correspondientes a los segundos jugados y al salario de cada jugador y lo guarda en un array.

Con las columnas correspondientes a las estadísticas restantes realiza un PCA y guarda las componentes principales en otro array.

Finalmente, la función devuelve el primer array con los inputs, el segundo array con los outputs y un tercer array con los nombres de los jugadores a analizar.

4.3.6. BCC_IO

Esta función recibe dos arrays, uno para los inputs y otro para los ouputs. Transforma los arrays en los problemas de programación lineal basándose en el modelo BCC con orientación input 2.10 realizando una iteración para cada DMU. Y finalmente, devuelve un array con la eficiencia de cada DMU.

4.3.7. BCC_OO

Esta función recibe dos arrays, uno para los inputs y otro para los ouputs. Transforma los arrays en los problemas de programación lineal basándose en el modelo BCC con orientación output 2.11 realizando una iteración para cada DMU. Y finalmente, devuelve un array con la eficiencia de cada DMU.

4.3.8. CCR

Esta función recibe dos arrays, uno para los inputs y otro para los ouputs. Transforma los arrays en los problemas de programación lineal basándose en el modelo CCR 2.9 realizando una iteración para cada DMU. Y finalmente, devuelve un array con la eficiencia de cada DMU.

4.3.9. Additive

Esta función recibe dos arrays, uno para los inputs y otro para los ouputs. Transforma los arrays en los problemas de programación lineal basándose en el modelo Additive 2.9 realizando una iteración para cada DMU. Y finalmente, devuelve un array con la eficiencia de cada DMU.

4.3.10. DEA_models

Esta función recibe 3 arrays, uno para los inputs, otro para los outputs y otro con los nombres de las DMU. Llama a las funciones BCC_IO,BCC_OO, CCR y Additive y muestra por pantalla un informe con los resultados de las eficiencias de dichos métodos.

CAPÍTULO 5

Ejemplo de los resultados para un caso concreto

En este capítulo se va a tratar de observar como funciona por dentro los procesos Python para un caso concreto, para ello se ha elegido el caso en el que se analizan los jugadores del equipo LAL (Los Angeles Lakers) para la temporada 2019.

Lo primero que se hace es llamar a la función filter_data, pasándole los parámetros season=2019 y team="LAL". Esta función leerá los ficheros bs_basic.csv, que contiene las estadísticas de los jugadores y all_salaries.csv, que contiene los salarios.

Realiza un filtrado de los datos quedándose solo con las filas correspondientes a dicha temporada y equipo, pasando de tener 617.837 filas a tan solo 705 filas en el fichero bs_basic y de 12.866 filas a 17 filas en el de all_salaries, como se puede observar en las Figuras 5.1 y 5.2.

	player	salary	season	team_id
11957	LeBron James	37436858	2019	LAL
11958	Anthony Davis	27093018	2019	LAL
11959	Danny Green	14634146	2019	LAL
11960	Kentavious Caldwell-Pope	8089282	2019	LAL
11961	Avery Bradley	4767000	2019	LAL
11962	JaVale McGee	4000000	2019	LAL
11963	Quinn Cook	3000000	2019	LAL
11964	Alex Caruso	2750000	2019	LAL
11965	Rajon Rondo	2564753	2019	LAL
11966	Kyle Kuzma	1974600	2019	LAL
11967	Markieff Morris	1750000	2019	LAL
11968	Dwight Howard	1620564	2019	LAL
11969	Jared Dudley	1620564	2019	LAL
11970	David Stockton	1378242	2019	LAL
11971	Talen Horton-Tucker	898310	2019	LAL
11972	Dion Waiters	375385	2019	LAL
11973	JR Smith	289803	2019	LAL

Figura 5.1: Salarios filtrados por equipo y temporada

	team_id	date	player	sp	fg	fga	fg3	fg3a	ft	fta
694	LAL	2020-03-08	Alex Caruso	392	0	1	0	0	0	0
695	LAL	2020-03-10	Anthony Davis	2193	9	19	4	8	4	5
696	LAL	2020-03-10	LeBron James	2092	12	22	4	9	1	5
697	LAL	2020-03-10	Avery Bradley	1868	4	11	2	8	0	0
698	LAL	2020-03-10	Danny Green	1739	2	5	2	4	0	0
699	LAL	2020-03-10	JaVale McGee	1006	2	5	0	0	0	0
700	LAL	2020-03-10	Kentavious Caldwell-Pope	1492	3	7	1	4	0	0
701	LAL	2020-03-10	Kyle Kuzma	1346	6	9	0	2	2	2
702	LAL	2020-03-10	Rajon Rondo	938	1	3	1	2	0	0
703	LAL	2020-03-10	Markieff Morris	881	0	2	0	1	0	0
704	LAL	2020-03-10	Alex Caruso	845	1	4	1	2	0	0

Figura 5.2: Estadísticas de los jugadores filtradas por partido, equipo y temporada

Como se necesitan las estadísticas a nivel de temporada y no a nivel de partido, se agrupan las filas por temporada, año y jugador y se suman todos los atributos quedándonos con solo 17 filas como se puede ver en la Figura 5.3.

	team_id	season	player	sp	fg	fga	fg3	fg3a	ft
0	LAL	2019	Alex Caruso	61916	110	260	38	107	58
1	LAL	2019	Anthony Davis	113364	508	994	65	194	386
2	LAL	2019	Avery Bradley	71188	170	383	63	173	20
3	LAL	2019	Danny Green	91751	178	425	113	299	32
4	LAL	2019	Dwight Howard	71575	188	257	3	5	89
5	LAL	2019	JaVale McGee	61375	181	283	3	6	49
6	LAL	2019	Jared Dudley	18646	20	50	16	34	2
7	LAL	2019	Kentavious Caldwell-Pope	96206	220	466	87	221	74
8	LAL	2019	Kostas Antetokounmpo	306	0	0	0	0	0
9	LAL	2019	Kyle Kuzma	79633	254	588	70	236	96
10	LAL	2019	LeBron James	125662	586	1176	133	381	239
11	LAL	2019	Markieff Morris	7053	14	36	6	21	4
12	LAL	2019	Quinn Cook	24510	76	173	25	66	6
13	LAL	2019	Rajon Rondo	59050	137	328	41	125	27
14	LAL	2019	Talen Horton-Tucker	319	0	1	0	1	0
15	LAL	2019	Troy Daniels	27346	62	158	40	112	10
16	LAL	2019	Zach Norvell	291	0	1	0	0	0

Figura 5.3: Estadísticas totales por temporada para cada jugador

Una vez filtrados ambos ficheros se realiza una unión de ambos, quedándonos con los jugadores que tienen tanto salario como estadísticas, ya que se puede dar el caso como con los jugadores David Stockton, Dion Waiters o JR Smith que se ha obtenido su salario pero no se tiene estadísticas de esa temporada, por no haber jugado o porque no venían en los datos que se nos han proporcionado. Y lo mismo pero al revés, los jugadores Kostas Antetokounmpo, Zach Norvell y Troy Daniels tienen estadísticas pero no se tiene su salario anual. Una vez descartados dichos jugadores se tiene un total de 14 jugadores o DMU's como se puede ver a continuación:

	player	salary	season	team_id	sp	fg	fga
0	LeBron James	37436858	2019	LAL	125662	586	1176
1	Anthony Davis	27093018	2019	LAL	113364	508	994
2	Danny Green	14634146	2019	LAL	91751	178	425
3	Kentavious Caldwell-Pope	8089282	2019	LAL	96206	220	466
4	Avery Bradley	4767000	2019	LAL	71188	170	383
5	JaVale McGee	4000000	2019	LAL	61375	181	283
6	Quinn Cook	3000000	2019	LAL	24510	76	173
7	Alex Caruso	2750000	2019	LAL	61916	110	260
8	Rajon Rondo	2564753	2019	LAL	59050	137	328
9	Kyle Kuzma	1974600	2019	LAL	79633	254	588
10	Markieff Morris	1750000	2019	LAL	7053	14	36
11	Dwight Howard	1620564	2019	LAL	71575	188	257
12	Jared Dudley	1620564	2019	LAL	18646	20	50
13	Talen Horton-Tucker	898310	2019	LAL	319	0	1

Figura 5.4: Jugadores finales sobre los que se realiza el estudio

Lo primero que se realizará es obtener los inputs para cada jugador, seleccionando las variables *sp y salary* y realizando una normalización de ambas variables dividiendo ambas por su media, tomando así la media total de ambas valor 1.

	sp	salary
0	61916.00000	2750000
1	113364.00000	27093018
2	71188.00000	4767000
3	91751.00000	14634146
4	71575.00000	1620564
5	61375.00000	4000000
6	18646.00000	1620564
7	96206.00000	8089282
8	79633.00000	1974600
9	125662.00000	37436858
10	7053.00000	1750000
11	24510.00000	3000000
12	59050.00000	2564753
13	319.00000	898310

Figura 5.5: Inputs, sp y salary

	sp	salary
0	0.98252	0.34314
1	1.79892	3.38062
2	1.12965	0.59482
3	1.45596	1.82602
4	1.13579	0.20221
5	0.97393	0.49911
6	0.29589	0.20221
7	1.52665	1.00937
8	1.26366	0.24639
9	1.99407	4.67130
10	0.11192	0.21836
11	0.38894	0.37433
12	0.93704	0.32003
13	0.00506	0.11209

Figura 5.6: Inputs, sp y salary normalizados

Posteriormente se obtienen los outputs, seleccionando el resto de variables disponibles, *fg, fga, fg3, fg3a, ft, fta, orb, drb, trb, ast, stl, blk, tov y plus_minus* y realizando sobre ellas un análisis en componentes principales para reducir su dimensionalidad.

Se obtiene una reducción de la dimensionalidad pasando de 14 variables a tan solo 3 componentes, de las cuales la primera tiene un 70,63 % de la variabilidad explicada, la segunda un 8,5 con un 94 % y un 4,87 % la última. Dejándonos un total el 94,07 % de la variabilidad explicada.

	PC1	PC2	PC3
0	-0.75473	-0.63681	0.57995
1	5.99880	2.36123	0.25735
2	-0.83107	-1.16343	0.55095
3	1.00428	-1.64812	2.02912
4	0.81875	3.32359	-0.08538
5	0.02351	2.67693	0.37781
6	-3.31461	-0.01353	-0.36413
7	0.28447	-1.31626	0.78406
8	0.70844	-0.44890	-0.18211
9	7.16161	-2.28292	-1.45111
10	-3.62190	0.23413	-0.58964
11	-2.69493	-0.39912	-0.37193
12	-0.83716	-0.86264	-0.87756
13	-3.94547	0.17584	-0.65739

Figura 5.7: Componentes principales extraídas en el ejemplo

El análisis en componentes principales devuelve los valores centrados en el origen, por lo que hay valores positivos y negativos en todas las componentes. Por lo que se tiene el problema de

que el análisis DEA no acepta valores negativos o nulos.

Teniendo en cuenta la invarianza del análisis DEA respecto a cambios de localización no hay ninguna pérdida de información al realizar una traslación en cada uno de los ejes o componentes del ACP, sumando a todas las proyecciones de los individuos una cantidad constante, de modo que los valores finales sean todos positivos. Esta cantidad será el valor mínimo de cada componente + 1, para que así los valores sean estrictamente mayores que cero.

	PC1	PC2	PC3
0	4.19074	2.64611	3.03106
1	10.94427	5.64415	2.70846
2	4.11440	2.11949	3.00206
3	5.94975	1.63481	4.48023
4	5.76422	6.60652	2.36573
5	4.96898	5.95986	2.82892
6	1.63086	3.26940	2.08698
7	5.22994	1.96666	3.23517
8	5.65391	2.83403	2.26900
9	12.10708	1.00000	1.00000
10	1.32357	3.51705	1.86147
11	2.25054	2.88381	2.07918
12	4.10831	2.42028	1.57355
13	1.00000	3.45876	1.79372

Figura 5.8: Componentes principales tras realizar la traslación

Finalmente se llama a la función DEA_models(), que a su vez llama a las funciones que calculan los modelos CCR, BCC_IO y BCC_OO y termina mostrándonos los resultados por pantalla junto a la media de estos resultados para cada DMU.

Se muestra un ejemplo del funcionamiento de la función que analiza el modelo BCC_IO para los datos de este ejemplo.

Teniendo todos los inputs (X) y outputs (Y), y los inputs y outputs correspondientes a la DMU Alex_Caruso_LAL_2019, respectivamente, X_A e Y_a :

$$X = \begin{pmatrix} 0{,}983 & 1{,}799 & 1{,}130 & 1{,}456 & 1{,}136 & 0{,}974 & 0{,}296 & 1{,}527 & \cdots & 0{,}005 \\ 0{,}343 & 3{,}381 & 0{,}595 & 1{,}826 & 0{,}202 & 0{,}499 & 0{,}202 & 1{,}009 & \cdots & 0{,}112 \end{pmatrix}$$

$$Y = \begin{pmatrix} 4{,}191 & 10{,}944 & 4{,}114 & 5{,}950 & 5{,}764 & 4{,}969 & 1{,}631 & 5{,}230 & \cdots & 1{,}000 \\ 2{,}646 & 5{,}644 & 2{,}119 & 1{,}635 & 6{,}607 & 5{,}960 & 3{,}269 & 1{,}967 & \cdots & 3{,}459 \\ 3{,}031 & 2{,}708 & 3{,}002 & 4{,}480 & 2{,}366 & 2{,}829 & 2{,}087 & 3{,}235 & \cdots & 1{,}794 \end{pmatrix}$$

$$X_A = \begin{pmatrix} 0{,}983 \\ 0{,}343 \end{pmatrix}, Y_A = \begin{pmatrix} 4{,}191 \\ 2{,}646 \\ 3{,}031 \end{pmatrix}$$

Se implementa el modelo 2.10 con los datos correspondientes a la DMU Alex_Caruso_LAL_2019. El modelo obtenido es el siguiente:

$$\underset{\mu,\delta,k}{\text{Max}} \quad W_A = \begin{pmatrix}\mu_1\mu_2\mu_3\end{pmatrix} \begin{pmatrix}4{,}191\\2{,}646\\3{,}031\end{pmatrix} + k_A$$

Sujeto a:

$$\begin{pmatrix}\delta_1\delta_2\end{pmatrix} \begin{pmatrix}0{,}983\\0{,}343\end{pmatrix} = 1$$
$$\begin{pmatrix}\mu_1\mu_2\mu_3\end{pmatrix} \begin{pmatrix}4{,}191 & 10{,}944 & \cdots & 1{,}000\\2{,}646 & 5{,}644 & \cdots & 3{,}459\\3{,}031 & 2{,}708 & \cdots & 1{,}794\end{pmatrix} + k_A \leq \begin{pmatrix}\delta_1\delta_2\end{pmatrix} \begin{pmatrix}0{,}983 & 1{,}799 & \cdots & 0{,}005\\0{,}343 & 3{,}381 & \cdots & 0{,}112\end{pmatrix} \tag{5.1}$$

$\mu^T, \delta^T \geq I\varepsilon$

k_A no restringida

$$\underset{\mu,\delta,k}{\text{Max}} \quad W_A = 4{,}191\mu_1 + 2{,}646\mu_2 + 3{,}031\mu_3 + k_A$$

Sujeto a:

$$\begin{aligned}
&0{,}983\delta_1 + 0{,}343\delta_2 = 1\\
&4{,}191\mu_1 + 2{,}646\mu_2 + 3{,}031\mu_3 + k_A \leq 0{,}983\delta_1 + 0{,}343\delta_2\\
&10{,}944\mu_1 + 5{,}644\mu_2 + 2{,}708\mu_3 + k_A \leq 1{,}799\delta_1 + 3{,}381\delta_2\\
&4{,}114\mu_1 + 2{,}119\mu_2 + 3{,}002\mu_3 + k_A \leq 1{,}130\delta_1 + 0{,}595\delta_2\\
&5{,}950\mu_1 + 1{,}635\mu_2 + 4{,}480\mu_3 + k_A \leq 1{,}456\delta_1 + 1{,}826\delta_2\\
&5{,}764\mu_1 + 6{,}607\mu_2 + 2{,}366\mu_3 + k_A \leq 1{,}136\delta_1 + 0{,}202\delta_2\\
&4{,}969\mu_1 + 5{,}960\mu_2 + 2{,}829\mu_3 + k_A \leq 0{,}974\delta_1 + 0{,}499\delta_2\\
&1{,}631\mu_1 + 3{,}269\mu_2 + 2{,}087\mu_3 + k_A \leq 0{,}296\delta_1 + 0{,}202\delta_2\\
&5{,}230\mu_1 + 1{,}967\mu_2 + 3{,}235\mu_3 + k_A \leq 1{,}527\delta_1 + 1{,}009\delta_2\\
&5{,}654\mu_1 + 2{,}834\mu_2 + 2{,}269\mu_3 + k_A \leq 1{,}264\delta_1 + 0{,}246\delta_2\\
&12{,}107\mu_1 + 1{,}000\mu_2 + 1{,}000\mu_3 + k_A \leq 1{,}994\delta_1 + 4{,}671\delta_2\\
&1{,}324\mu_1 + 3{,}517\mu_2 + 1{,}861\mu_3 + k_A \leq 0{,}112\delta_1 + 0{,}218\delta_2\\
&2{,}251\mu_1 + 2{,}884\mu_2 + 2{,}079\mu_3 + k_A \leq 0{,}389\delta_1 + 0{,}374\delta_2\\
&4{,}108\mu_1 + 2{,}420\mu_2 + 1{,}574\mu_3 + k_A \leq 0{,}937\delta_1 + 0{,}320\delta_2\\
&1{,}000\mu_1 + 3{,}459\mu_2 + 1{,}794\mu_3 + k_A \leq 0{,}005\delta_1 + 0{,}112\delta_2\\
&\mu^T, \delta^T \geq I\varepsilon\\
&k_A \text{ no restringida}
\end{aligned} \tag{5.2}$$

Resolviendo la ecuación mediante programación lineal nos devuelve el resultado 1.00 indicándonos que la Unidad Alex_Caruso_LAL_2019 es eficiente. Se realiza este proceso para todos los jugadores y para los modelos BCC input orientado, BCC output orientado, CCR Y Additive.

Modelo	Alex Caruso	Anthony Davis	Avery Bradley	Danny Green
BCC_IO	1,000	1,000	0,812	1,000
BCC_OO	1,000	1,000	0,916	1,000
CCR	0,649	0,304	0,448	0,282
Additive	1,000	1,000	0,911	1,000

Modelo	Dwight Howard	JaVale McGee	Jared Dudley	Kentavious Caldwell-Pope
BCC_IO	1,000	1,000	0,824	0,753
BCC_OO	1,000	1,000	0,966	0,928
CCR	1,000	0,637	0,673	0,368
Additive	1,000	1,000	0,950	0,836

Modelo	Kyle Kuzma	LeBron James	Markieff Morris	Quinn Cook
BCC_IO	0,876	1,000	0,712	0,761
BCC_OO	0,969	1,000	0,966	0,903
CCR	0,857	0,253	0,574	0,484
Additive	0,721	1,000	0,939	0,778

Modelo	Rajon Rondo	Talen Horton-Tucker
BCC_IO	0,777	1,000
BCC_OO	0,817	1,000
CCR	0,674	1,000
Additive	0,606	1,000

Si se tiene en cuenta los modelos BCC, tanto con orientación input como output, se puede concluir que los mejores jugadores del equipo Los Angeles Lakers para la temporada 2019 serían Alex Caruso, Anthony Davis, Danny Green, Dwight Howard, JaVale McGee, LeBron James y Talen Horton-Tucker puesto que tienen un 1 de eficiencia. Mientras que de los jugadores ineficientes, Avery Bradley, Jared Dudley, Kentavious Caldwell-Pope, Kyle Kuzma, Markieff Morris, Quinn Cook y Rajon Rondo, el más ineficiente sería Markieff Morris con un 71,2 % de eficiencia para el caso input y Rajan Rondo para el caso output.

Si por el contrario se tiene en cuenta en el modelo CCR solo las DMU Dwight Howard y Talen Horton-Tucker serían las unidades óptimas. Mientras que el resto de DMU son ineficientes.

Y, finalmente fijandonos en el modelo aditivo, las DMUs eficientes son Alex Caruso Anthony Davis, Danny Green, Dwight Howard, JaVale McGee, LeBron James y Talen Horton-Tucker, siendo Rajan Rondo el jugador menos eficiente con un 60,6 %

De todo esto se podría llegar a la conclusión de que los mejores jugadores del equipo son tanto Dwight Howard como Talen Horton-Tucker, puesto que son los únicos eficientes en todos los modelos. Mientras que por el otro lado, el peor jugador sería Rajon Rondo ya que es el jugador menos eficiente en el mayor de los casos. Lo que nos haría pensar que es un jugador al que el equipo debería de reducir el salario o incluso vender para poder reemplazarle por otro jugador.

CAPÍTULO 6
Conclusiones

El resultado de este Trabajo de Fin de Grado ha sido gratamente reconfortable, después de muchas horas revisando y estudiando libros, documentos y revistas especializadas, he conseguido elaborar este trabajo que profundiza en el análisis envolvente de datos (DEA) como herramienta para medir la eficiencia en los jugadores de la NBA. Lo que me ha permitido adquirir desde cero todos los conocimientos necesarios sobre el análisis DEA.

Este proyecto empezó con el objetivo de realizar un análisis de la eficiencia para el caso particular de los jugadores de un equipo y una temporada de la NBA, pero ha sido llevado más allá, permitiéndonos realizar el análisis que se desee para cualquier equipo o temporada.

Este proyecto también me ha permitido conocer más a fondo cómo se realiza un proyecto de investigación, los pasos que se llevan a cabo y la forma correcta de realizarlo.

Este documento escrito ha sido realizado con la ayuda de LateX y Overleaf que me ha dado la facilidad para incluir todo el código implementado y las formulas teóricas empleadas. Para ello también ha sido necesario aprender y manejar su uso, ya que los resultados que aporta presenten una alta calidad tipográfica pero requiere de un profundo aprendizaje para su manejo.

Ya que este TFG permite conocer qué jugadores son eficientes y cuáles no lo son, podría permitir a los equipos de la NBA evaluar a dichos jugadores al final de cada temporada para así renovar a los que sí que sean eficientes y no renovar o reducir el salario a los no lo son, pudiendo así mejorar la calidad de su equipo.

Este proyecto es extrapolable a cualquier equipo de otros deportes, para los cuales habría que reelegir cuáles serían las entradas y salidas para cada jugador y recopilar los datos correspondientes.

CAPÍTULO 7
Bibliografía

[1] Cienciadedatos. Análisis de Componentes Principales (Principal Component Analysis, PCA) y t-SNE
https://www.cienciadedatos.net/documentos/35_principal_component_analysis

[2] Kaggle. NBA - Player Salary (1990-2017)
https://www.kaggle.com/whitefero/nba-player-salary-19902017

[3] Basketball Reference. Glossary
https://www.basketball-reference.com/about/glossary.html

[4] ESPN. NBA Player Salaries
https://www.espn.com/nba/salaries/_/

[5] Vicente C. y Olga B. (2000). Evaluación de la Eficiencia mediante el análisis envolvente de datos. Universidad de Valencia, España.
https://www.uv.es/vcoll/libros/2006_evaluacion_eficiencia_DEA.pdf

[6] Héctor S. (2020). Desarrollo de un Data Lake para la gestión de datos de estadísticos de la competición NBA (National Basketball Association). Universidad de Valladolid, Valladolid, España.
https://uvadoc.uva.es/handle/10324/44398

[7] Álvaro B. (2020). Aprendizaje automático de modelos para la predicción de resultados de partidos de la NBA. Universidad de Valladolid, Valladolid, España.
https://uvadoc.uva.es/handle/10324/44144

[8] Javier M. (2020). Métodos estadísticos en competiciones deportivas de baloncesto: la NBA. Universidad de Valladolid, Valladolid, España.
https://uvadoc.uva.es/handle/10324/43847

[9] Joseph S. (2002). Preparing Your Data for DEA.
https://www.researchgate.net/publication/226132954_Preparing_your_data_for_DEA

[10] Data Envelopment Analysis (DEA)
https://es.mathworks.com/matlabcentral/fileexchange/53145-data-envelopment-analysis-dea

[11] ElPerimetro. ¿Qué es la NBA?
https://elperimetro.es/que-es-la-nba/

[12] Fandom. ¿Qué es la NBA?
https://nba.fandom.com/es/wiki/Que_es_la_NBA%3F

[13] Docs Spict. Linprog Function
https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.linprog.html

[14] EL Análisis Envolvente de Datos (DEA). Universidad de Alicante
https://rua.ua.es/dspace/bitstream/10045/19658/6/Materiales.Teoria.Bloque.III.pdf

[15] The LateX Project
https://www.latex-project.org/

[16] Overleaf
https://es.overleaf.com/

Todos los enlaces han sido verificados por última vez el día 6 de Julio de 2021.

CAPÍTULO 8

Anexo: Código

8.1. merge_calendars()

```
def merge_calendars():
    excel_salaries = pd.read_csv('player_Salaries.csv')
    excel_salaries = excel_salaries[
        ["Player Name", "Salary", "Season Start", "Team"]]
    excel_salaries.columns = ["player", "salary", "season", "team_id"]

    web_salaries = pd.read_csv('salaries_web.csv')
    web_salaries = web_salaries[["Player", "Salary", "Season", "team"]]
    web_salaries.columns = ["player", "salary", "season", "team_id"]

    all_salaries = excel_salaries.merge(web_salaries, how='outer')

    all_salaries.to_csv('all_salaries.csv', index=False)
```

8.2. read_calendar()

```
def read_calendar():
    all_data = pd.read_csv("boxscores_1969_2020/bs_basic.csv",
                           usecols=["date"])
    all_data = all_data.sort_values("date").drop_duplicates().reset_index()
    all_data.date = pd.to_datetime(all_data.date)
    calendar_season = pd.DataFrame(columns=["start_season", "end_season"])
    calend_start = [all_data.iloc[0].date - timedelta(days=1)]

    for i in all_data.index - 1:
        end_season = all_data.iloc[i].date
        start_season = all_data.iloc[i + 1].date
        diff = (start_season - end_season).days
        if diff > 90:
            print(diff)
            print(end_season, start_season)
            calend_start.append(end_season + (start_season - end_season) / 2)

    calend_start.append(all_data.iloc[-1].date + timedelta(days=1))
```

```
    for idx in range(len(calend_start) - 1):
        season_name = calend_start[idx].strftime(
            "%Y")  # + calend_start[idx + 1].strftime("%Y")
        calendar_season.loc[season_name] = [
            calend_start[idx].strftime("%d/%m/%Y"),
            calend_start[idx + 1].strftime("%d/%m/%Y")]

    calendar_season.reset_index(inplace=True)
    calendar_season = calendar_season.rename(columns={'index': 'season'})
```

8.3. read_salary()

```
def read_salary():
    all_salaries = pd.DataFrame()
    for year in range(2019, 2021):
        end = False
        page = 1
        while not end:
            web = f'http://www.espn.com/nba/salaries/_/year/' \
                  f'{year}/page/{page}/seasontype/3/'
            table = pd.read_html(web)  # Lee las tablas de la pagina web
            table = table[0]  # Se queda con la primera tabla
            table.columns = table.iloc[
                0]  # Asigna la primera fila como nombre de las columnas
            table = table.iloc[1:, 1:]  # Elimina la primera fila
            # Transforma las columnas des esta tabla y las a ade al total
            if not table.empty:
                table["Season"] = str(year - 1)
                table.columns = ["Player", "Team", "Salary", "Season"]
                all_salaries = all_salaries.append(table)
                page += 1
            else:
                end = True
    all_salaries = all_salaries[all_salaries.Player != "NAME"]
    all_salaries.Salary = all_salaries.Salary.replace({"\\$": "", ",": ""},
                                                      regex=True)
    all_salaries.Player = all_salaries.Player.replace(
        {", PG": "", ", PF": "", ", C": "", ", SF": "", ", SG": "", ", G": "",
         ", F": ""}, regex=True)
    all_salaries.reset_index(inplace=True, drop=True)

    # Lee el fichero teams que contiene los pares nombre equipo - siglas equipo
    teams = pd.read_csv("teams.csv")

    final_salaries = pd.merge(all_salaries, teams, left_on="Team",
                              right_on="full_team")

    final_salaries.to_csv('salaries_web.csv', index=False)
```

8.4. reduce_data()

```
def reduce_data():
    calendar = pd.read_csv("calendar_season.csv")
    calendar.start_season = pd.to_datetime(calendar.start_season)
    calendar.end_season = pd.to_datetime(calendar.end_season)

    data = pd.read_csv(
        'boxscores_1969_2020/boxscores_basic.csv')  # , nrows=100)
    data.date = pd.to_datetime(data.date)
    data = data[data.date > "24/08/1996"]
    data = data[data.box_type == "game-basic"]
    data = data[data.sp > 0]
    data = data[
        ["team_id", "date", "player", "sp", "fg", "fga", "fg3", "fg3a", "ft",
         "fta", "orb", "drb", "trb", "ast", "stl", "blk", "tov", "pf", "pts",
         "plus_minus"]]
    data.plus_minus = data.plus_minus.fillna(0)
    data["season"] = 0
    for _, row in calendar.iterrows():
        data.loc[(data.date < row.end_season) & (
                data.date > row.start_season), "season"] = row.season
    data.to_csv("boxscores_1969_2020/bs_basic.csv", index=False)
```

8.5. filter_data()

```
def filter_data(team=None, season=None, player=None):
    data = pd.read_csv("boxscores_1969_2020/bs_basic.csv")
    salaries = pd.read_csv('all_salaries.csv')
    group = ["player"]
    if team and season and player:
        print("error")
        exit(-1)
    if team:
        data = data[data.team_id == team]
        salaries = salaries[salaries.team_id == team]
        if data.empty:
            print("error team")
            exit(-1)
    else:
        group.append("team_id")

    if season:
        data = data[data.season == season]
        salaries = salaries[salaries.season == season]
        if data.empty:
            print("error season")
            exit(-1)
```

```
    else:
        group.append("season")

    if player:
        data = data[data.player == player]
        salaries = salaries[salaries.player == player]
        if data.empty:
            print("error_player")
            exit(-1)

    data_end = data.groupby(["team_id", "season", "player"])[
        "sp", "fg", "fga", "fg3", "fg3a", "ft", "fta", "orb", "drb",
        "trb", "ast", "stl", "blk", "tov", "pf", "pts", "plus_minus"].apply(
        lambda x: x.astype(float).sum())
    data_end.reset_index(inplace=True)
    all_data = data_end.merge(salaries, on=['player', 'season', 'team_id'],
                              how='inner').sort_values("season")
    names_DMU = (all_data.player + "_" + all_data.team_id + "_" +
                 all_data.season.astype(str)).to_list()
    x = all_data[['sp', 'salary']]
    x = x / x.mean()
    x = np.array(x).T

    y = all_data[["fg", "fga", "fg3", "fg3a", "ft", "fta", "orb", "drb",
                  "trb", "ast", "stl", "blk", "tov", "plus_minus"]]
    y = princomp.princomp(y, 0.9)
    y = np.array(y).T
    return x, y, names_DMU
```

8.6. princomp()

```
def princomp(data, percentage):
    # Normalizamos los datos
    scaler = StandardScaler()
    scaler.fit(data)
    x_scaled = scaler.transform(data)

    # Instanciamos objeto PCA y aplicamos
    pca = PCA(n_components=percentage)
    # Obtenemos los componentes principales
    pca.fit(x_scaled)
    # Convertimos nuestros datos con las nuevas dimensiones de PCA
    x_pca = pca.transform(x_scaled)
    expl = pca.explained_variance_ratio_
    print(expl)
    print('suma:', sum(expl))

    # Cambiamos el nombre de las columnas
    columns = ['PC' + str(s) for s in list(range(1, len(expl) + 1))]
```

```
    x_pca2 = pd.DataFrame(x_pca , index=data.index, columns=columns)
    x_pca_end = x_pca2 - x_pca2.min() + 1
    return x_pca_end
```

8.7. BCC_IO()

```
def BCC_IO(x, y):
    n_input, total_dmu = x.shape
    m_output, _ = y.shape
    Z = np.zeros(total_dmu)

    A_ub = np.concatenate(
        (np.matrix(y.T), np.matrix(-x.T), np.ones((total_dmu, 1))), axis=1)

    b_ub = np.zeros((total_dmu, 1))

    b_eq = 1

    lb = np.concatenate((np.zeros((n_input + m_output, 1)), [[-np.inf]]))
    ub = np.array([[None]] * (n_input + m_output + 1))
    bounds = np.concatenate((lb, ub), axis=1)
    for dmu in range(total_dmu):
        print(dmu, total_dmu)
        f = np.concatenate(
            (np.matrix(-y[:, dmu]), np.zeros((1, n_input)), [[-1]]), axis=1)
        A_eq = np.concatenate(
            (np.zeros((1, m_output)), np.matrix(x[:, dmu]),
             np.zeros((1, 1))), axis=1)
        z = linprog(f, A_ub=A_ub, b_ub=b_ub, A_eq=A_eq, b_eq=b_eq,
                    bounds=bounds, method='simplex')
        Z[dmu] = -z.fun

    return Z
```

8.8. BCC_OO()

```
def BCC_OO(x, y):
    n_input, total_dmu = x.shape
    m_output, _ = y.shape
    Z = np.zeros(total_dmu)

    A = np.concatenate(
        (np.matrix(y.T), np.matrix(-x.T), np.ones((total_dmu, 1))), axis=1)

    b = np.zeros((total_dmu, 1))

    beq = 1
```

```
    lb = np.concatenate((np.zeros((n_input + m_output, 1)), [[-np.inf]]))
    ub = np.array([[None]] * (n_input + m_output + 1))
    bounds = np.concatenate((lb, ub), axis=1)
    for dmu in range(total_dmu):
        f = np.concatenate(
            (np.zeros((1, m_output)), np.matrix(x[:, dmu]), [[-1]]), axis=1)
        Aeq = np.concatenate(
            (np.matrix(y[:, dmu]), np.zeros((1, n_input)), np.zeros((1, 1))),
            axis=1)
        z = linprog(f, A_ub=A, b_ub=b, A_eq=Aeq, b_eq=beq, bounds=bounds,
                    method='simplex')
        Z[dmu] = 1 / z.fun

    return Z
```

8.9. CCR()

```
def CCR(x, y):
    n_input, total_dmu = x.shape
    m_output, _ = y.shape
    Z = np.zeros(total_dmu)

    A_ub = np.concatenate((np.matrix(y.T), np.matrix(-x.T)), axis=1)

    b_ub = np.zeros((total_dmu, 1))

    b_eq = 1

    lb = np.array((np.zeros((n_input + m_output, 1))))
    ub = np.array([[None]] * (n_input + m_output))
    bounds = np.concatenate((lb, ub), axis=1)
    for dmu in range(total_dmu):
        f = np.concatenate((np.matrix(-y[:, dmu]), np.zeros((1, n_input))),
                           axis=1)
        A_eq = np.concatenate((np.zeros((1, m_output)), np.matrix(x[:, dmu])),
                              axis=1)
        z = linprog(f, A_ub=A_ub, b_ub=b_ub, A_eq=A_eq, b_eq=b_eq,
                    bounds=bounds, method='simplex')
        Z[dmu] = -z.fun

    return Z
```

8.10. Additive()

```
def Additive(x, y):
    n_input, total_dmu = x.shape
    m_output, _ = y.shape
    Z = np.zeros(total_dmu)
```

```
    A_ub = np.concatenate(
        (np.matrix(y.T), np.matrix(-x.T), -np.ones((total_dmu, 1))), axis=1)

    b_ub = np.zeros((total_dmu, 1))

    lb = np.concatenate((np.ones((n_input + m_output, 1)), [[-np.inf]]))
    ub = np.array([[None]] * (n_input + m_output + 1))
    bounds = np.concatenate((lb, ub), axis=1)
    for dmu in range(total_dmu):
        f = np.concatenate((np.matrix(-y[:, dmu]), np.matrix(x[:, dmu]), [[1]]
                            axis=1)

        z = linprog(f, A_ub=A_ub, b_ub=b_ub, bounds=bounds)
        sol = z.x
        u = sol[0:m_output]
        v = sol[m_output:m_output + n_input]
        v0 = sol[n_input + m_output]
        Y = u.T.dot(y[:, dmu])
        X = v.T.dot(x[:, dmu])
        eff = Y / (X + v0)
        eff = round(eff * 100000) / 100000
        Z[dmu] = eff

    return Z


def CCR(x, y):
    n_input, total_dmu = x.shape
    m_output, _ = y.shape
    Z = np.zeros(total_dmu)

    A_ub = np.concatenate((np.matrix(y.T), np.matrix(-x.T)), axis=1)

    b_ub = np.zeros((total_dmu, 1))
```

8.11. DEA_models()

```
def DEA_models(x, y, DMU_names):
    result1 = BCC_IO(x, y)
    result2 = BCC_OO(x, y)
    result3 = CCR(x, y)
    result4 = Additive(x, y)

    final_result = pd.DataFrame([result1, result2, result3, result4],
                                columns=DMU_names,
                                index=["BCC_IO", "BCC_OO", "CCR", "Additive"])
    print(final_result)
    final_result.to_csv("final_result.csv",decimal=",",float_format='%.3f')
```