Análisis y predicción del rendimiento ofensivo de debutantes en las Grandes Ligas de Béisbol

Nombre: Armando Benavides Esteva

Titulación: Máster Universitario en Ciencia de Datos

Área: Minería de datos y machine learning

Nombre Consultor/a: Jerónimo Hernández Gónzalez

Nombre Profesor/a: Jordi Casas Roma

Fecha Entrega: 9/6/2019

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Análisis y predicción del rendimiento ofensivo de debutantes en las Grandes Ligas de Béisbol*
Nombre del autor:	*Armando Benavides Esteva*
Nombre del consultor/a:	Jerónimo Hernández Gónzalez
Nombre del PRA:	*Jordi Casas Roma*
Fecha de entrega (mm/aaaa):	06/2019
Titulación::	*Máster universitario en Ciencia de Datos*
Área del Trabajo Final:	*Minería de datos y machine learning*
Idioma del trabajo:	*Español*
Palabras clave	*Béisbol, novatos, analytics*

Resumen

En el presente trabajo se realiza un estudio para encontrar conocimiento que permita mejorar la toma de decisiones y, con mayor base, seleccionar los peloteros más preparados para jugar en las Ligas Mayores de Béisbol. Para esto, mediante una metodología de desarrollo ágil se crea una herramienta que predice el rendimiento ofensivo de los jugadores según sus estadísticas históricas en las Ligas Menores de Béisbol.

Para llegar al sistema final, siguiendo una metodología cuantitativa para la obtención de información, se consiguen los datos mediante web scrapping los cuales se analizan para comprenderlos en detalle. Un estudio de diferentes modelos de agrupamiento y clasificación, unido a pruebas y búsquedas de mejores parámetros ayudan a seleccionar los modelos que conformarán el núcleo del sistema predictivo. Los modelos permiten agrupar a los jugadores con características similares y clasificar las nuevas muestras. El conjunto de datos utilizado sobre los modelos es previamente procesado para facilitar el trabajo de los modelos y mejorar los resultados. La predicción de los nuevos jugadores se realiza basándose en el rendimiento histórico de los jugadores pertenecientes al mismo grupo, los cuales presentan características similares.

Los resultados obtenidos sitúan al presente sistema en posición ventajosa, logrando disminuir los errores de las predicciones considerablemente respecto a otros modelos del mismo ámbito. No obstante, es imprescindible considerar otros acercamientos sobre todo en los cálculos predictivos, así como mejorar la interpretación de los resultados para que las personas, independientemente de su conocimiento sobre la materia sean capaces de asimilarlos.

Abstract

In the present work a study is made to find knowledge that allows to improve the decision-making process and with more data to select the most prepared players to play in the MLB. For this purpose, using an agile development methodology, a tool that predicts the offensive performance of the players according to their historical statistics in minor leagues is created.

To reach the final system, following a quantitative methodology to obtain information, data is obtained through web scrapping, which is analyzed to understand better the info they offer. A study of different grouping and classification models, together with tests and searches of better parameters helps to select the models forming the core of the predictive system. The models allow to group players with similar characteristics and classify new samples. The dataset used on the models is previous processed to facilitate the work of the models and improve the results. The prediction of the new players is based on the historical performance of the players belonging to the same group, which have similar characteristics.

The results obtained place the present system in an advantageous position, managing to reduce the errors of the predictions considerably compared to other models in the same field. However, it is essential to consider other approaches especially in predictive calculations, as well as to improve the interpretation of results so that people, regardless of their knowledge of the subject, are able to assimilate them.

Índice

Lista de ilustraciones

Lista de tablas

Lista de gráficas

1. Introducción

1.1 Contexto y justificación del Trabajo

1.1.1 Contexto

La MLB[1], es uno de los escenarios más representativos del béisbol a nivel mundial, donde cientos de jugadores de diferentes partes del mundo se congregan para practicar este deporte. Cada organización perteneciente a la liga es responsable de adquirir los mejores jugadores para sus equipos y cuentan con los denominados sistemas de granjas, los cuales, fomentan y entrenan a los jugadores que pudieran beneficiar eventualmente al equipo perteneciente a la Major League Baseball. Los equipos que conforman las granjas forman parte de la MiLB[2], la cual consta de diferentes ligas asociadas a un nivel.

Durante las temporadas, muchos jugadores supuestamente preparados, son seleccionados de las granjas para jugar al más alto nivel. Aquellos que logran llegar a la MLB en algún momento de su carrera son considerados novatos[3]. Sin embargo, mientras unos logran permanecer durante un tiempo y con un desempeño determinado, otros, mayormente por presentar un desempeño pobre, son devueltos a las canteras en espera de otra oportunidad para aportar o rendir lo necesario para el equipo.

1.1.2 Justificación y motivación

Con la ofensiva como centro de análisis y teniendo en cuenta el desempeño histórico de los jugadores antes de hacer su debut en las ligas mayores, ¿Podría conocerse el rendimiento de estos en la liga de más alto nivel? ¿Se podrá saber con antelación que beisbolistas pueden desempeñarse de manera satisfactoria y/o ser tendentes a permanecer en la MLB? ¿De los jugadores que permanecen en la liga, quiénes podrían realizarlo de

[1] Las Ligas Mayores de Béisbol o Grandes Ligas de Béisbol, en inglés Major League Baseball (MLB) son ligas de béisbol profesional con equipos pertenecientes a Estados Unidos y Canadá. Está compuesta por la Liga Nacional y Liga Americana, con un total de 30 organizaciones. Cada una de ellas, presenta un equipo principal que participa en la MLB y contiene afiliados como parte del sistema de granjas.

[2] Las Ligas Menores de Béisbol, en inglés Minor League Baseball (MiLB) es una jerarquía de equipos de béisbol profesionales que están afiliados a Major League Baseball. Cada equipo de la MLB tiene su propia red de equipos de ligas menores (a veces llamados "equipos de granjas" o "ligas de granjas") que se utilizan para el desarrollo de jugadores.(Bernier, s. f.)

[3] Un jugador será considerado un novato a menos que, durante una temporada o temporadas anteriores, haya (a) excedido 130 turnos al bate o 50 entradas lanzadas en las Grandes Ligas; o (b) acumulado más de 45 días en la lista activa de un club o clubes de Grandes Ligas durante el período de límite de 25-jugadores (excluyendo el tiempo en el servicio militar y el tiempo en la lista de discapacitados).(MLB, s. f.)

manera sobresaliente? Desde el punto de vista del personal encargado de la toma de decisiones respecto a la promoción de jugadores, las respuestas a estas preguntas pudieran ser de mucha utilidad pues otorgarían una noción más acertada del tipo de desempeño que reflejarían cada uno de los peloteros. Podría encontrarse, además, el momento más cercano u óptimo según sus estadísticas, para el ascenso de ellos. Esto último trae como contraparte, la base y opción para decidir la necesidad de otorgarle más tiempo de entrenamiento en las ligas menores como parte de su preparación para la MLB.

Aunque en un primer momento, resolver estas interrogantes parecería ayudar solamente a las organizaciones de la liga, existe un sector que es mayoritario y en el cual, el autor de esta memoria se siente identificado. Además de amante de las estadísticas y el béisbol, le apasiona jugar las denominadas Ligas de fantasía[4], donde encuentra un buen entorno para poner en práctica todos los análisis e investigaciones realizados. Una de las modalidades que tienen este tipo de juegos virtuales es precisamente organizar y dirigir un equipo a través de los años y al igual que en la vida real, la selección de posibles novatos se hace presente con repercusión en la actual y posteriores temporadas virtuales. De este entorno surge la idea y motivación de desarrollo de este trabajo.

1.1.3 Resolución

Con el surgimiento y actual auge de tecnologías y herramientas para el tratamiento y análisis de datos, así como su demostrada utilidad para la obtención de conocimiento, el campo de la ciencia de datos se presta meritoriamente para resolver problemas de tópicos deportivos; específicamente del béisbol, donde pueden encontrarse datos de hace más de cien años con estadísticas que han definido y caracterizado a este deporte. Se proyecta utilizar tecnologías de este ámbito para obtener el conocimiento necesario que permita dar respuestas a las preguntas y solucione los objetivos planteados.

[4] Las ligas de fantasía en el béisbol comprenden cualquier número de juegos en los que el objetivo es acumular las estadísticas contabilizadas por los jugadores de béisbol en la vida real para el equipo virtual del usuario o "propietario". Para confeccionar los equipos se realizan, por lo general, drafts (o rondas de selecciones) entre los propietarios de equipos virtuales que han de competir ya sea en enfrentamientos cara-a-cara semanales o en categorías acumulativas donde se otorgan puntos de acuerdo a la posición relativa en esas categorías (también conocidas como Rotisserie). (Sabino, 2005)

1.2 Objetivos del Trabajo

El presente trabajo se enfoca en el análisis del desempeño ofensivo de los jugadores de ligas menores para predecir el mismo en las ligas mayores. Con este planteamiento como base, se expone como objetivo principal:

- Predecir el rendimiento ofensivo de los novatos en la MLB de acuerdo a su desempeño histórico en la MiLB.

De la misma manera se tienen como objetivos secundarios y como complemento importante de la investigación:

- Determinar la línea base de estadísticas ofensivas con las que suelen ascender a los jugadores.
- Definir grupos de novatos de acuerdo al rendimiento ofensivo en la MLB y determinar la línea base de estas estadísticas para predecir los jugadores que pudieran realizar su debut de forma satisfactoria.
- Realizar análisis para intentar identificar y predecir los novatos que por sus estadísticas tienden a permanecer en la liga.

1.3 Enfoque y método seguido

Los datos necesarios para realizar este tipo de investigación aparecen en su mayoría, de forma gratuita para su consumo en distintos sitios web. Se pueden encontrar con estructuras específicas de acuerdo al objetivo del sitio y con diferentes facilidades de acceso y obtención. Sin embargo, el presente trabajo requiere una estructura de datos peculiar para su correcta manipulación, así como un análisis específico; por lo que puede resultar complejo utilizar productos previamente existentes donde sea necesario realizar adaptaciones en vez de un producto nuevo. Aunque un estudio a profundidad del estado del arte ha de propiciar mas elementos para la decantación por una u otra estrategia, se decide optar por la creación de un producto nuevo.

La orientación del trabajo es principalmente de descubrimiento y aplicación para la práctica. No obstante, tendrá rasgos experimentales con comprobaciones y descripciones de hechos. La metodología utilizada será cuantitativa con un diseño empírico-analítico, sin embargo tendrá pequeñas pinceladas de investigación cualitativa debido al estudio en

el campo a efectuar. En el aspecto metodológico de desarrollo, se pretende utilizar Rapid Application Development[5].

1.4 Planificación del Trabajo

La complejidad y variedad de estructuras de datos requeridas hace necesaria la obtención de estos a través de diferentes sitios web utilizando técnicas de web scrapping. Se tiene la intención de utilizar *Python* como lenguaje de programación principal y la librería *beautifulsoup* para la obtención de datos. Mediante *pandas,* se pretenden organizar los datasets y exportarlos en un formato idóneo para su posterior análisis.

Teniendo la extracción y limpieza de los datos como primer paso, se procedería a efectuar diversos análisis estadísticos para comprender a detalle la información obtenida y pruebas de hipótesis para demostrar postulados. El uso de paquetes como *matplotlib* para los gráficos, *numpy* y *scipy* para las operaciones estarán presentes en esta sección de la investigación.

El siguiente acercamiento corresponde al proceso de creación de modelos predictivos, de clasificación y agrupamiento que, durante un proceso de experimentación se seleccionarían los más adecuados para solucionar los objetivos planteados. **Scikit-learn** en este aspecto, se presenta como la principal candidata para resolver estos problemas, aunque de forma general se estudiará la viabilidad del uso de alternativas en todos los momentos del desarrollo del trabajo.

Una vez completado el proceso de implementación se procedería a realizar el acabado del documento que ha sido actualizado en cada una de las fases o momentos de desarrollo. Retoques y modificaciones necesarias se efectuarán con el objetivo de que la información quede lo más concisa y clara posible.

Mediante el siguiente diagrama de Gantt se detallan las actividades y tiempos de cada una:

[5] Rapid Application Development (RAD), es un ciclo de vida de desarrollo diseñado para ofrecer un desarrollo mucho más rápido y resultados de mayor calidad que los que se obtienen con el ciclo de vida tradicional. Está diseñado para aprovechar al máximo el potente software de desarrollo que ha evolucionado recientemente. (Martin, 1991)

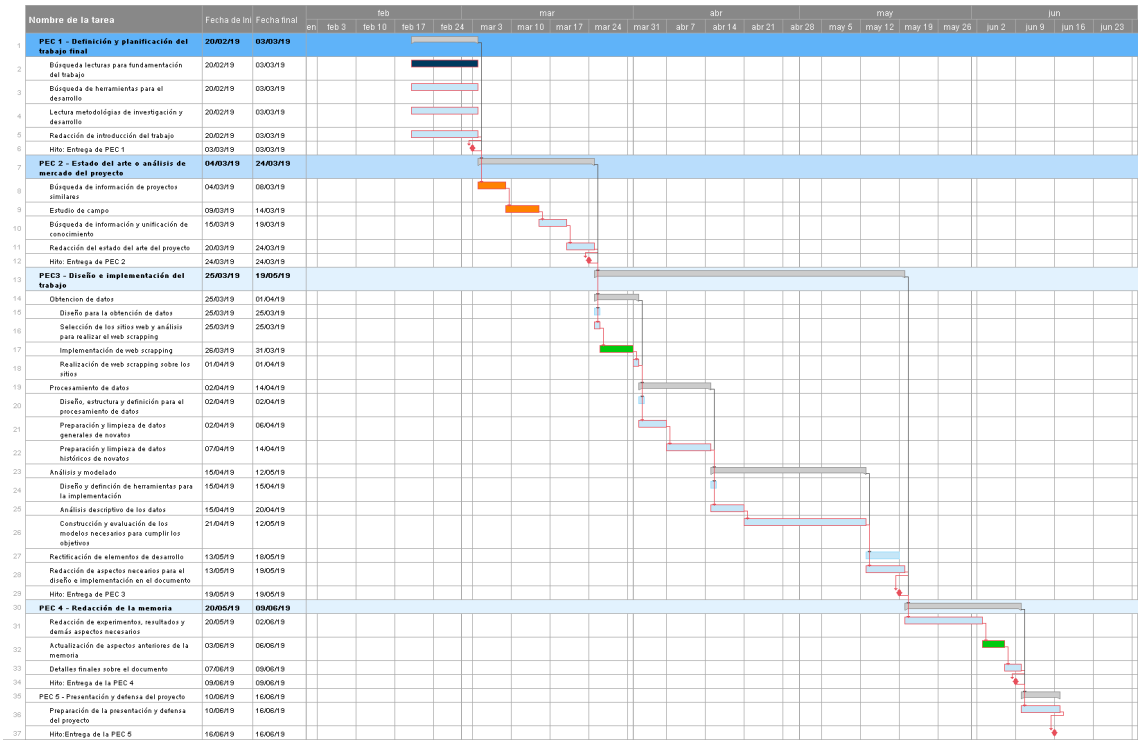


Ilustración 1Diagrama de Gantt de la planificación del proyecto

1.5 Breve sumario de productos obtenidos

El producto final puede considerarse como un sistema divido en diferentes módulos. El primer modulo estará encargado de la extracción de datos, el segundo lo compondrá un modelo que se encargará de agrupar a los jugadores con características similares, el tercero lo conformará un modelo clasificador de las nuevas muestras en los grupos creado y un cuarto módulo dedicado de realizar los cálculos para las predicciones. La unión de estos módulos propicia un conjunto de datos con las predicciones de varias estadísticas de cara al debut en las Ligas Mayores.

1.6 Breve descripción de los otros capítulos de la memoria

El presente trabajo está conformado por 10 capítulos. El primer capítulo e introductorio propicia el contexto del trabajo, así como los objetivos y la motivación para el desarrollo de este. Una planificación de desarrollo es expuesta además del enfoque y métodos seguidos. Se incluye también resumen de los productos obtenidos así como del contenido del documento.

El capítulo 2 corresponde al estado del arte. En este se documenta información acerca del estado actual de las investigaciones o productos relacionados con predicciones de rendimiento en el béisbol. El contenido sirve como base para el posterior desarrollo del sistema. El capítulo 3 incluye la forma en que se llegará a la solución del sistema, pasos a seguir y herramientas a utilizar.

En el capítulo 4 se detalla la estructura que van a tener los datos. Se aclara dónde y cómo se obtendrán, las pautas a seguir tanto para extracción de información como la confección de los conjuntos de datos, así como el proceso a llevar a cabo para completar las tareas. Se realizan también descripciones apoyados de gráficas sobre el contenido de los conjuntos de datos para una mejor comprensión. Incluye además, el proceso de selección de atributos del jugador, preprocesado y reducción de dimensionalidad. Se explica igualmente cómo queda conformado el conjunto de entrenamiento y prueba utilizado a lo largo del desarrollo del sistema.

El capítulo 5 aborda, en un primer espacio diferentes modelos de agrupamiento con los cuales se realizarán pruebas y búsquedas de parámetros para elegir el más adecuado a utilizar. Una vez concluidas, se efectúa una comparación entre los modelos hasta seleccionar uno. Intentando seleccionar un modelo de clasificación, en un segundo momento se realizan también diversas pruebas y búsquedas de mejores hiperparámetros para ayudar a seleccionar el modelo más adecuado para el sistema.

Una vez construido el núcleo del sistema es necesario realizar las predicciones. El capítulo 6 aborda como se llegan a las predicciones finales, mostrándose comparaciones respecto a otros sistemas que han intentado predecir el rendimiento de los jugadores. Explicado todo el proceso se presentan las conclusiones del trabajo, recomendaciones para mejorar el sistema, se presenta un glosario de ayuda y la bibliografía como últimos 4 capítulos de la memoria.

2. Estado del arte

Slowinski, en su artículo Projection Systems planteó: "*Una proyección, en general, es la mejor estimación del verdadero talento de un jugador durante un período de tiempo determinado."* (Slowinski, 2010). Predecir el rendimiento de los jugadores ha sido tema básico para los equipos y fanáticos del béisbol. A través del tiempo se han desarrollado investigaciones y creado sistemas para satisfacer diferentes o específicas necesidades como Marcel, Steamer, ZiPS, KATOH, PECOTA, Oliver y Chone. Un resumen de estos y otros modelos se presenta a continuación, los cuales servirán como base para el desarrollo de este trabajo.

Marcel

Creado por Tom Tango (Baseball-Reference, s. f.), es uno de los sistemas más conocidos y a la vez básicos que existen. Proyecta las estadísticas de los jugadores MLB teniendo en cuenta los valores de las mismas en las 3 temporadas anteriores(Tango, 2012). Tango pondera cada temporada, toma en cuenta los promedios de la liga para cada una, incluye el factor edad y ejecuta una regresión a la media. En el caso de los novatos sin experiencia en Grande Ligas, su proyección será el promedio de la liga.(Tango, 2004). La tabla de resultados contiene una columna de confianza que indica la fiabilidad del pronóstico; mientras más alto el valor, más certera la predicción(Tango, 2012).

PECOTA

Player Empirical Comparison and Optimization Test Algorithm (PECOTA, por sus siglas en inglés), es un Sistema desarrollado por Nate Silver en el año 2003 que utiliza comparaciones históricas de jugadores con trayectorias similares. Es considerado uno de los predictores más precisos y aunque se usa principalmente para predecir el desempeño de jugadores, puede aplicarse a equipos (MLB, s. f.). Como parte de sus características, PECOTA permite usar las estadísticas de ligas menores para proyectar el desempeño del jugador en las mayores. A su vez, utiliza promedios ponderados y regresión a la media para obtener un estimado del verdadero valor del talento del jugador; realizando también, un ajuste de su trayectoria basado en la información de los cambios en las estadísticas a través del tiempo de los jugadores comparables (Baseball Prospectus, s. f.).

Steamer

Steamer projections es un sistema de proyección de estadísticas de jugadores de béisbol creado por Jared Cross, Dash Davidson y Peter Rosenbloom(Cross, Davidson, & Rosenbloom, s. f.). Utiliza el rendimiento pasado y tendencias de envejecimiento para realizar la proyección (MLB, s. f.). Parecido a Marcel, utiliza un promedio ponderado de del rendimiento pasado e incluye regresión a la media de la liga, aunque la ponderación de cada año y el grado de regresión varía entre estadísticas (Druschel, 2016).

ZiPS

sZymborski Projection System (ZiPS), es un sistema de proyecciones de jugadores desarrollado por Dan Szymborski. Usa curvas de crecimiento y declive basados en el tipo de jugador para encontrar tendencias. Luego, factoriza esas tendencias en el desempeño pasado de los jugadores para llegar a las proyecciones. Utiliza, dependiendo de la edad de los jugadores, los últimos 3 o 4 años, pesando las temporadas más recientes. Parecido a otros sistemas, usa además las tendencias de envejecimiento para generar las proyecciones. Incluye también dentro de sus ecuaciones, factores de velocidad, datos de lesiones y referentes a las jugadas de los juegos (MLB, s. f.).

Los anteriores sistemas pueden considerarse como los más reconocidos y son utilizados por sitios de renombre como Fangraphs y Baseball Reference, los cuales proporcionan información sobre el béisbol y contienen una inmensa colección de datos de los jugadores y equipos de distintas ligas. No obstante, existen otros con filosofías similares o con puntos de vista diferentes que vale la pena mencionar:

- **Oliver**: Creado por Brian Cartwright, es un sistema de comparativa simple que utiliza promedios ponderados de las últimas 3 temporadas, así como ajustes de envejecimiento y regresión. La diferencia principal con otros sistemas es que la equivalencia con las Ligas Mayores la realiza tomando los números y ajustándolos al parque y la liga. Las proyecciones de este sistema son mejores cuando se muestra como se desempeñarán los jugadores jóvenes en la MLB (Slowinski, 2011).
- **Cairo**: Desarrollado en Revenge of te RLYW, imita en un principio a Marcel, incluyendo más adelante estadísticas de ligas menores, ajusta los efectos del parque, liga, así como la curva de envejecimiento según la estadística. Utiliza cuatro años de datos anteriores y toma en cuenta la edad y la posición en la regresión del jugador. Utilizan un

simulador denominado Diamond Mind para colocar estas proyecciones y estimar las proyecciones de los equipos después de 50 000 simulaciones (Slowinski, 2011).

- **Fans**: Ideada por FanGraphs después de finalizar la temporada del 2009. Consiste simplemente en que los fanáticos pronostiquen el rendimiento de cada jugador de acuerdo a su percepción. Las boletas que llenan los usuarios son compiladas y promediadas, el resultado obtenido será la proyección(Slowinski, 2011).
- **Chone**: Desarrollado por Sean Smith, utiliza 4 años de datos para los bateadores y 3 para los pitchers utilizando efectos del parque, liga y envejecimiento. Usa además datos de ligas menores y bolas bateadas. Fue considerado uno de los sistemas más precisos(Slowinski, 2011).

Otros sistemas como **KATOH** utilizan regresiones probabilísticas para pronosticar el bateo en Grandes Ligas con estadísticas de Ligas Menores (Mitchell, 2014). El sistema **Bill James**, tomando los 8 años anteriores y enfocándose en los 3 más cercanos tiene en cuenta la edad, el terreno de casa donde juegan, rendimiento pasado y el tiempo de juego esperado (Slowinski, 2011).

Desde otra perspectiva, utilizando técnicas de más reciente auge, se realizaron estudios y nuevos sistemas con objetivos similares. Uno de ellos fue la investigación elaborada por Gabriel Chandler y Guy Stevens quienes plantearon como problema proyectar el futuro éxito de los jugadores de béisbol de las Ligas Menores en cada nivel del sistema de granjas. Aplicando métodos basados en árboles, específicamente Random Forest, tuvieron en cuenta las estadísticas que más se correlacionaban con el éxito en MLB. Considerando además, las diferentes formas en que los equipos utilizan las estadísticas para manejar a los jugadores y como la posición del draft[6], se usa como medida de la habilidad del jugador durante su carrera en la MiLB (Chandler & Stevens, 2012). En el mismo documento, exponen algunos de los resultados:

- Los bateadores de las Ligas Menores no pueden ser juzgados con precisión por su desempeño al inicio de sus carreras.

[6] El draft de la MLB es un proceso de selección donde los equipos escogen a jugadores que cumplan cierto criterio de selección, generalmente amateurs para integrar sus filas. Durante 40 rondas los 30 equipos piden a los jugadores en el orden inverso a sus posiciones al cierre de la temporada anterior(MLB, s. f., s. f.)

- Aunque algunas conclusiones pueden ser posibles desde el inicio, el cuerpo de trabajo del jugador no será verdaderamente revelador del potencial MLB hasta que no haya progresado a niveles superiores.
- Los picks (selecciones) altos del draft suelen ser promovidos con un conjunto dado de estadísticas, lo que demuestra que el desempeño no es la única herramienta para predecir el valor de un prospecto.

Como más reciente sistema se encuentra **DeepBall**, desarrollado por Daniel Calzada en el año 2018; este utiliza una Red Neuronal Recurrente y ensembles para predecir el rendimiento del jugador. Se enfoca en proyectar los valores de 8 categorías utilizando la información previa a la temporada que se desea y aunque en un principio fue diseñado para predecir un solo año, puede realizarse para la cantidad de años que se deseen especificar, teniendo en cuenta que las estadísticas anteriores, serán las predichas por el modelo. El modelo tiene presente la influencia del parque donde se juega y se aplican regularizaciones y bagging para reducir la varianza. Finalmente DeepBall fue comparado con otros modelos como el mencionado Marcel, un Random Forest y una Red Neuronal Recurrente base cuyas pruebas sugerían que DeepBall realizaba mejores predicciones (Calzada, 2018).

3. Propuesta de solución

Acorde a las características del problema a resolver, el tiempo de desarrollo y los recursos humanos, resulta adecuado implementar un pequeño sistema para automatizar ciertos procesos. La metodología ágil R.A.D., se presta para estos propósitos ya que tiende a englobar la usabilidad, reusabilidad, utilidad y la rapidez de ejecución, utilizando un enfoque de construcción basado en componentes. El sistema final ha de ser capaz de extraer los datos, prepararlos, utilizarlos en los modelos y arrojar los resultados que serán analizados (Ilustración 2 Fases generales del desarrollo de sistema). Cada uno de estos aspectos pueden considerarse como las fases para el desarrollo del sistema y serán detallados en los siguientes capítulos

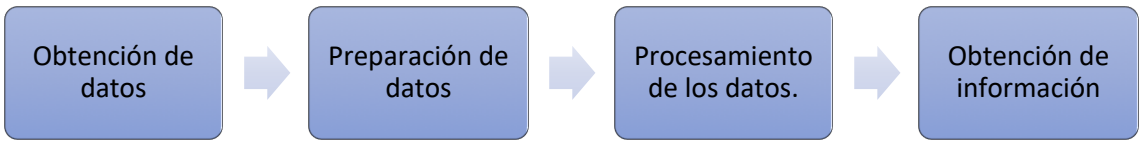


Ilustración 2 Fases generales del desarrollo de sistema

3.1 Propuesta del sistema

El estudio del estado del arte mostró diferentes acercamientos para predecir el rendimiento de jugadores. Las nuevas tecnologías existentes, sobre todo en el área de la Ciencia de Datos, ofrecen facilidades para recrearlos, mejorarlos o innovarlos. Haciendo uso de estas, se pretende crear un sistema formado principalmente por modelos de minerías de datos.

Obteniendo las estadísticas de los jugadores de las ligas menores un año antes de su debut en Grandes Ligas, se aplicará un algoritmo aglomerativo para encontrar y agrupar peloteros con características similares. Las estadísticas que el modelo ha de utilizar tienen que representar el nivel del jugador en el año correspondiente. Para lograr un aproximación, se calculará la diferencia de cada una de las métricas respecto a la media de la liga. Este tipo de acercamiento propicia una comparación más justa del rendimiento del jugador independientemente de la temporada[7].

[7] De manera hipotética se tiene que el jugador A conectó 35 hits(H) en el año 2016. Por otra parte el jugador B, en el año 2014 propició 30. Tomando simplemente estos valores, pudiéramos decir que el jugador A se desenvolvió mejor en esa estadística. Ahora, supongamos que la media de hits de la liga, en el año 2016 fue de 35 mientras que en año 2014, de 25. Analizando el caso con esta nueva información, el jugador A

Los grupos creados tendrán su reflejo estadístico en la temporada donde los jugadores realizaron su debut. De esta manera, mediante un algoritmo de clasificación, las nuevas muestras serán asignados a un grupo y se les proyectará el rendimiento esperado en su debut. La ilustración (Ilustración 3 Proceso hacia la proyección de jugadores) muestra de manera general el proceso descrito.

Ilustración 3 Proceso hacia la proyección de jugadores

Contextualizando el método de predicción: Supongamos que una nueva muestra cae en un determinado grupo. Dicho grupo, como promedio, va a exhibir una variación (representada porcentualmente) que consiste en la disminución o aumento del valor total de la métrica el año antes y después del debut. Si este nuevo jugador cuyo valor en la estadística S (Sencillos) antes de su debut es 45, es clasificado en un grupo donde la diferencia en el salto hacia la MLB para esa estadística de los jugadores similares es de -7%; el valor esperado para esa métrica será una disminución del 7%, aproximadamente 42.

3.2 Herramientas y métodos

Luego de un análisis de la complejidad de desarrollo se seleccionaron las herramientas y métodos a utilizar. Como lenguaje de programación se utiliza Python en su versión 3.7, compilado generalmente sobre Jupyter Notebook (aplicación para crear código en vivo) en la etapa de prueba y sobre el IDE PyCharm como parte de la integración del sistema. Ya con el lenguaje definido se definen las librerías disponibles para acometer las faenas necesarias.

Para la captura de los datos, se utilizó Beautiful Soup. Una de las limitantes presentes, era la ausencia de una base o conjunto de datos disponible acorde al futuro procesamiento. Por tal motivo se recurre al web scrapping, un método de obtención de datos muy conocido, cuya finalidad es la recopilación de información directamente de los sitios web.

se comportó de forma promedio, mientras que el jugador B, a pesar de haber conectado menos indiscutibles lo hizo por encima de la liga, lo que indica un mejor *performance* en esa métrica.

Beautiful Soup, de fácil manejo y gran prestigio, fue la librería que sirvió como herramienta para estas labores.

Scikit-learn es una reconocida herramienta de minería y análisis de datos. Sus disímiles implementaciones de algoritmos de clasificación, regresión, agrupamiento, preprocesamiento de datos, reducción de dimensionalidad, etc., lo colocan como la mejor candidata para llevar a cabo las funciones de preparación y minería de datos.

Otras librerías como *numpy* y *pandas* se utilizaron para la confección y organización de los conjuntos de datos así como para la ejecución de cálculos. Matplotlib y yellowbrick, ayudaron a graficar tanto modelos como gráficas necesarias para una mejor comprensión de la información.

Por otra parte y como complemento investigativo, se realizaron análisis generalmente descriptivos mediante el lenguaje de programación R y la plataforma R Studio. Entre las librerías destacadas se encuentran *ggplot* para graficar; *dplyr* para un mejor trabajo con data frames y *corrplot*, específicamente para la matriz de correlación.

4. Datos

4.1 Estructura de los datos

El desarrollo del trabajo se basa en el uso de estadísticas ofensivas. Generalmente, para seguir el rendimiento de un jugador se pueden llevar estadísticas que reflejen el total, el promedio o incluso algunas marcadas por fórmulas que van desde las más simples hasta las más enrevesadas[8]. Las estadísticas necesarias para el sistema (Tabla 1 Estadísticas base a utilizar) serán principalmente referentes a totales con los cuales podrán obtenerse estadísticas porcentuales.

Estadística	Tipo	Descripción
PA	Total	Comparecencias al bate
AB	Total	Veces al bate
S	Total	Sencillos
2B	Total	Dobles
3B	Total	Triples
HR	Total	Home Runs
H	Total	Hits (S+2B+3B+HR)
SH	Total	Hits de sacrificio
SF	Total	Elevados de sacrificio
BB	Total	Bases por bolas
SB	Total	Bases robadas
CS	Total	Capturado robando
HBP	Total	Golpeado por lanzador
BA (AVG)	Porcentual	Promedio de bateo (H/AB)

[8] Una lista completa con definiciones incluídas de las estadísticas utilizadas en la MLB puede encontrarse en su sitio web (MLB, s. f.).

OBP	Porcentual	Promedio de embasado {(H+BB+HBP)/(AB+BB+HBP+SF)}
SLG	Porcentual	Porcentaje de slugging {[(S) + (2*2B)+(3*3B)+(4*HR)]/AB}
OPS	Formula (Porcentual)	Embasado más Slugging (OBP+SLG)

Tabla 1 Estadísticas base a utilizar

Estos valores definen al jugador en diferentes aspectos o para varios eventos en un juego de béisbol. Sin embargo, como complemento pudieran necesitarse características de otro ámbito del jugador que aporten información alterna, sobre todo al momento de ofrecer los datos al modelo (Tabla 2 Datos complementarios a utilizar).

Dato	Descripción
Age	Edad del jugador
B	Lado de bateo (Derecho, izquierdo)
League_id	Id de combinaciones de las ligas en las que jugó el año antes del debut.

Tabla 2 Datos complementarios a utilizar

Se han de confeccionar tres datasets principales. El primero, con la información antes del debut, recibirá un tratamiento adecuado de tal manera que reflejen el desempeño del jugador por encima (o debajo) de la media de liga en cada métrica. El segundo, con la información el año del debut, mostrará los totales de cada estadística y junto con el conjunto de datos que almacene los totales antes del debut, como tercer dataset, se le podrán calcular las diferencias porcentuales a la nueva muestra. Otros conjuntos de datos, aunque no de mucho peso, ayudarán durante el proceso. Cada jugador conformará una fila con los valores respectivos para cada atributo. Aunque puede utilizarse más información para una mejor interpretación de los datos, la expuesta en las tablas anteriores es la fundamental para el desarrollo del sistema.

Sports Reference es un conjunto de sitios dedicado a proveer estadísticas y recursos para distintos deportes. Uno de los sitios o módulos pertenecientes es Baseball Reference; este está orientado al béisbol y contiene la información necesaria para el desarrollo del trabajo. La estructura final solicitada no se encuentra de la manera requerida en el sitio;

por tal motivo, es inevitable extraer la información de diferentes páginas para finalmente organizar los datasets. Teniendo en cuenta los requisitos de la estructura ha de obtenerse la siguiente información del sitio:

- Información básica del jugador
- Estadísticas antes del año de debut del jugador (en MiLB)
- Estadísticas del año del debut del jugador (en MLB)
- Estadísticas de cada liga en la que el jugador participó

Con el objetivo de obtener la información básica, se aprovechará una página del sitio que brinda para cada año registrado, información tanto personal como del debut. Ya con los jugadores que debutaron se buscarán sus estadísticas tanto en las ligas mayores como en las ligas menores. A su vez, se extraerán las estadísticas para cada liga en cada uno de los años. El proceso de web scrapping, luego de un análisis del sitio y requerimientos ha de seguir las siguientes pautas:

- Se extraerán las estadísticas del debut en MLB entre los años 2013 y 2017.
- Se extraerán las estadísticas del año antes del debut entre los años 2012 y 2016.
- No se tomarán en cuenta jugadores sin estadísticas el año anterior al debut.
- No se tomarán en cuenta jugadores con incorrecta referencia estadística según el año del debut. (Debuta un año y sus estadísticas para ese debut reflejan otro).
- Las dos ligas MLB (NL y AL) serán recopiladas de forma única.
- Las ligas menores tomadas en cuenta son representadas por los niveles siguientes:
  - AAA: Triple-A
  - AA: Doble-A
  - A+: A-Advanced
  - A: A
  - A-: Short-Season A
  - Rk: Rookie

La extracción fue realizada de manera ordenada teniendo en cuenta la necesidad de información tanto para las siguientes descargas como para los posteriores cálculos. Algunas de las reglas surgieron debido a errores tanto en la descarga como en la verificación de la integridad de los datos. Por lo general, los datos se encontraban bien organizados dentro del sitio y los algoritmos creados pudieron reutilizarse en su mayoría. La ilustración (Ilustración 4 Proceso de extracción y preparación de los datos) muestra el

proceso llevado a cabo para obtener finalmente los datasets deseados. Cada subproceso permite realizar un seguimiento de la operación que ejecuta, de esta manera evitamos perdida de datos, arrastre de errores y mejoramos el análisis de la información.

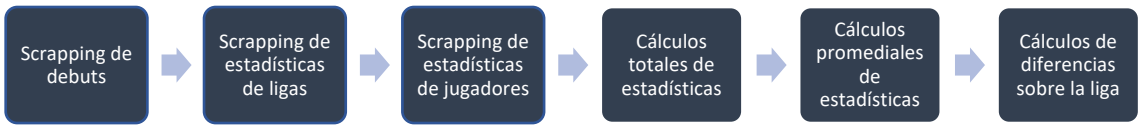


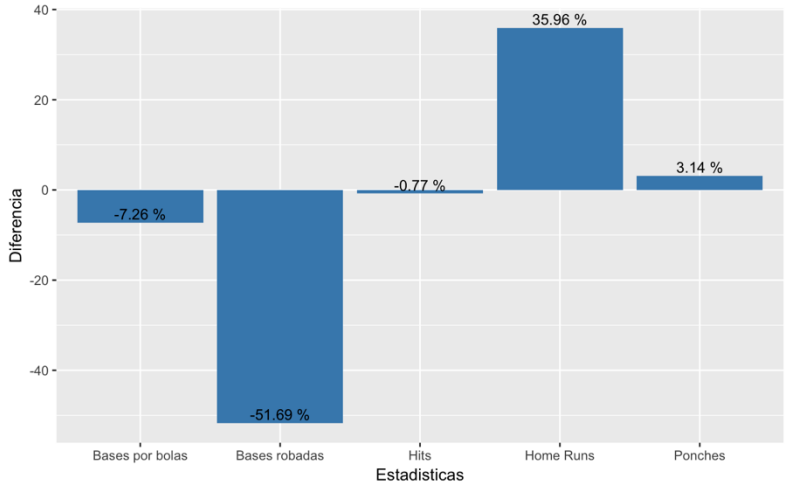
Ilustración 4 Proceso de extracción y preparación de los datos

4.2 Preparación y Análisis de los conjuntos datos

Una vez terminado el proceso de obtención de datos, quedan conformados los siguientes datasets:

debuts_info.csv: Es el primer dataset en crearse y refleja información básica de los debuts. Entre los años 2013 y 2017 se encontraron 515 jugadores que debutaron con una edad mínima de 20 años y máxima 33. El promedio de edad de los debutantes fue de 24.39 donde el 57.3% batean del lado derecho, un 29.7% del lado izquierdo y el 13% restante puede batear de ambos lados. Contiene datos complementarios para seguir el proceso de extracción como los *ids* de los jugadores y el *link* que los referencia a la página donde se encuentra en detalle la información de cada cual.

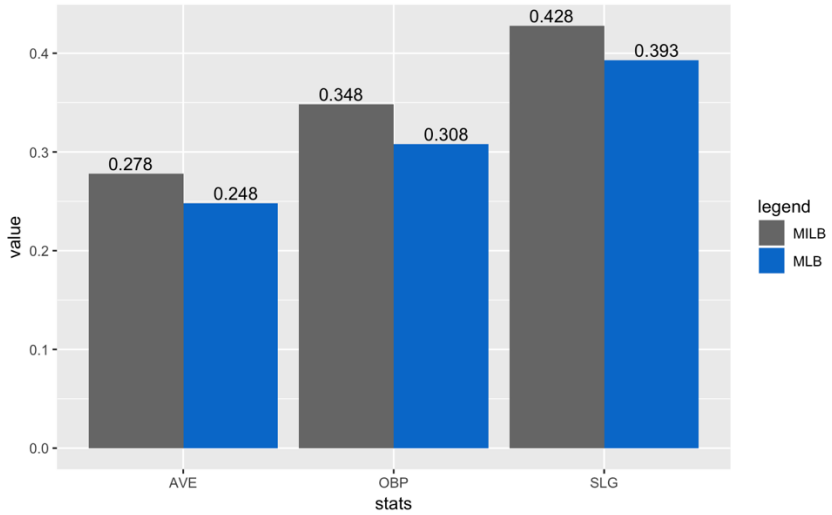
leagues_stats_data.csv: Almacena 18 variables en las que se incluyen las estadísticas que representan a cada uno de los niveles de las ligas. Los valores de las métricas contienen los totales registrados en el año. Agrupando todos los datos de los niveles de las ligas menores se confecciona la siguiente gráfica (Gráfica 1 Variación de ocurrencia de las estadísticas en la MLB respecto a la MiLB) que muestra, para algunas métricas, las diferencias existentes en la MLB respecto a la MiLB. Un resultado positivo indica el aumento de la frecuencia de ocurrencia de ese evento en la MLB (ocurrencia por cada comparecencia al bate (PA)). Lo más llamativo de la gráfica es la gran disminución de bases robadas en grandes ligas, aproximadamente un 51.69% menos que en las ligas menores.



Gráfica 1 Variación de ocurrencia de las estadísticas en la MLB respecto a la MiLB

milb_before_debut_stats.csv: Este conjunto de datos se conforma dada la lista de debuts y con una enorme influencia de algunas reglas de scrapping. Por tal motivo, de la cantidad inicial de muestras, quedaron reflejadas finalmente 493. Las estadísticas totales de los jugadores correspondientes al año anterior al debut son almacenadas en el dataset. Este archivo es de gran peso para realizar las predicciones.

mlb_debut_stats.csv: Su estructura es similar al anterior archivo**,** con la diferencia de almacenar las estadísticas de su primer año en MLB. Para el proceso se considera como uno de los datasets principales pues se utilizará para obtener el comportamiento esperado de las nuevas muestras. A continuación, se representa gráficamente la denominada *Slash Line* (AVG/OBP/SLG) o línea ofensiva de los jugadores. Mediante un diagrama de barras agrupado por estadísticas, se pueden comparar los valores de estas para los jugadores el año anterior al debut (en MiLB) y el año de su debut en MLB (Gráfica 2 Línea ofensiva de los jugadores antes (MiLB) y en el año de debut (MLB)). La tendencia, como se aprecia, es una disminución de la línea ofensiva.



Gráfica 2 Línea ofensiva de los jugadores antes (MiLB) y en el año de debut (MLB)

Los conjuntos de datos anteriores son la base para realizar los cálculos que dejaran preparado los datasets finales. Con esta información, nuevos archivos fueron creados.

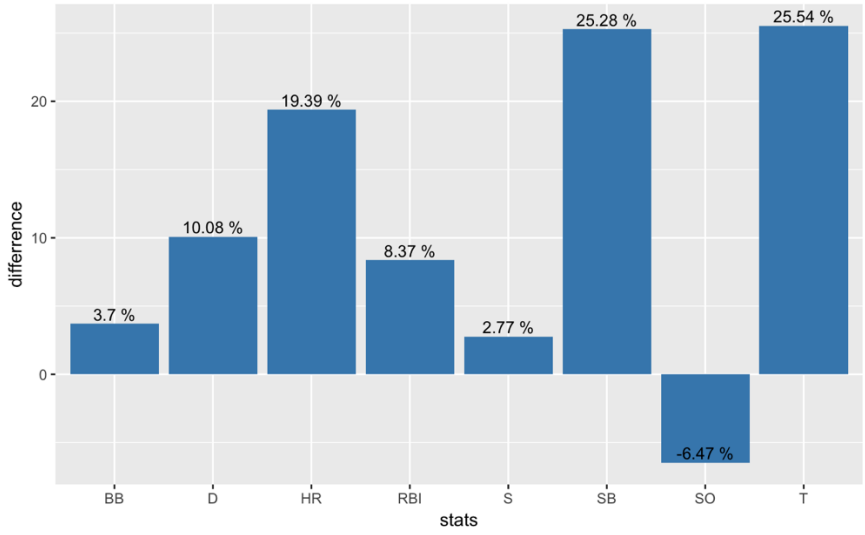
milb_league_stats_for_player.csv: El año antes de debutar, los jugadores pudieron pasar por diferentes niveles de las ligas menores. Por tal motivo y partiendo de la información guardada en **leagues_stats_data.csv**, se crea este conjunto de datos que almacena la suma de las estadísticas de los niveles correspondientes a cada jugador. Estos datos se usarán posteriormente para definir el comportamiento medio de la liga o las ligas en las que el jugador participó. Un nuevo atributo surge de esta suma y es **League_id** que identifica a cada combinación de niveles existente. De esta manera, peloteros con el mismo **League_id** pueden ser identificados con la misma trayectoria en cuanto a niveles. La única condición tomada en cuenta es que el orden en que se pasaron los niveles no influirá en el identificador, ejemplo: [AA,AAA] será identificada de la misma manera que [AAA,AA]. Surgen entonces 29 identificadores de los que se puede obtener que:

- El id más representativo es el correspondiente al nivel AA, lo que quiere decir que la mayor cantidad de jugadores de la muestra dan el salto a la MLB sólo desde las ligas correspondientes a ese nivel. Esto representa un 29%.
- Aproximadamente el 20% de los jugadores pasaron por los niveles AA y AAA, exclusivamente antes de debutar
- Un total de 79 jugadores (16%) realizaron el salto proviniendo solamente de las ligas AAA.

mlb_league_stats_for_player.csv: De manera más sencilla se confeccionó un conjunto de datos donde se almacenaban los totales de las estadísticas respectivos al año del debut de cada jugador. Los datos provinieron, principalmente de **leagues_stats_data.csv**.

Utilizando los datasets que contienen los totales tanto de los jugadores como de las ligas y sus combinaciones, se conformaron los conjuntos de datos con las frecuencias para cada estadística respecto a las comparecencias al bate (PA). Como resultado se obtienen 4 nuevos archivos **before_debut_players_ave.csv**, **debut_players_ave.csv**, **milb_league_ave.csv**, **mlb_league_ave.csv**, cuyos cálculos reflejaban las frecuencias de las estadísticas contenidas en **milb_before_debut_stats.csv**, **mlb_debut_stats.csv**, **milb_league_stats_for_player.csv** y **mlb_league_stats_for_player.csv** respectivamente.

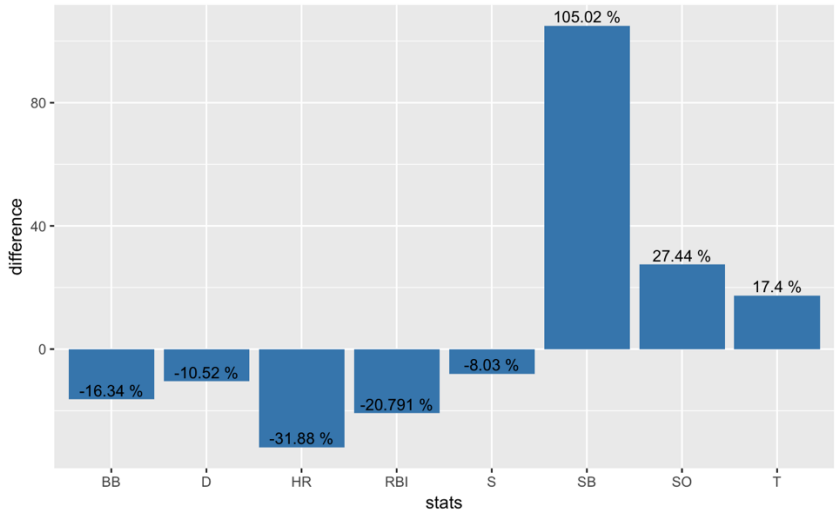
milb_players_values_over_league_average.csv: Finalmente se conforma el dataset deseado para utilizar en los modelos. Este guarda las diferencias de cada estadística del jugador respecto a la liga o combinaciones de liga el año antes de su debut. En la gráfica (Gráfica 3 Diferencia respecto a la liga de los jugadores antes del año del debut) se muestra la diferencia respecto a la liga en algunas estadísticas de los jugadores antes de realizar el debut. Evidentemente estos son seleccionados por tener de manera general un resultado superior al promedio de las ligas en las que participaron. Jugadores con grandes habilidades para conectar home runs, triples y robadores de bases, son buenos candidatos, al parecer, para dar el salto a las mayores.



Gráfica 3 Diferencia respecto a la liga de los jugadores antes del año del debut

mlb_players_values_over_league_average.csv: Como complemento de análisis se obtuvo este dataset el cual propicia las diferencias de los debutantes respecto a la MLB

de cada estadística. En la siguiente gráfica (Gráfica 4 Diferencia respecto a la liga de los jugadores el año del debut) se puede observar como los jugadores que generalmente lucían un resultado por encima en las ligas menores, exhiben en el año de su debut un rendimiento menor que la media de jugadores en la MLB. Como excepciones se encuentran los parámetros que indican velocidad, como bases robadas y triples, los cuales se mantienen incluso con mejores resultados.



Gráfica 4 Diferencia respecto a la liga de los jugadores el año del debut

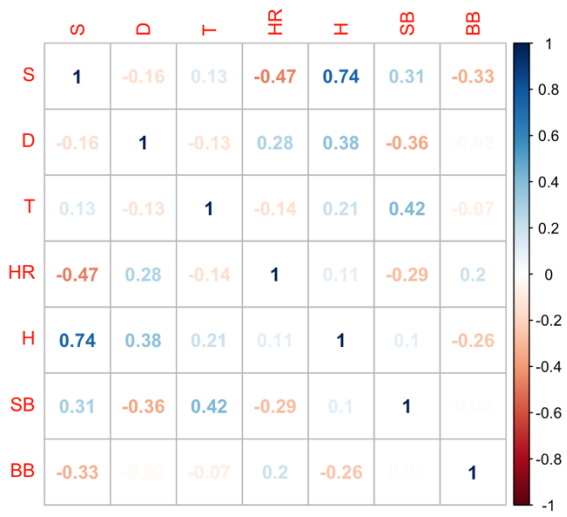
4.3 Preparación de los datos para el modelo

Una vez listo el dataset a utilizar en los modelos, se procede a prepararlo adecuadamente para su procesamiento. Uno de los aspectos importantes es el tratamiento de valores ausentes. Como parte de las reglas de scrapping, esta verificación se hace en cada momento de extracción así como de cálculos; no obstante, se vuelve a realizar una última verificación donde se eliminan todas las muestras que pudieran contener ausencias de datos. Después de este chequeo, el dataset no sufrió modificaciones y permaneció con 493 muestras.

El primer momento de la preparación de los datos será la selección de atributos de acuerdo a las habilidades de los jugadores. Cada estadística puede reflejar aspectos de ofensivos de un jugador como la velocidad, poder y contacto. Una vez seleccionados estos, se efectuará una prueba de correlación para evitar ofrecer información recurrente al modelo. Luego de unos ajustes a los tipos de datos, se finalizará con una reducción de dimensionalidad.

4.3.1 Selección de atributos representativos

Este proceso es un primer filtro que se realizará intentando que las estadísticas presentes representen distintos aspectos de la ofensiva de un jugador. El proceso específico de selección y extracción de atributos se efectuará más adelante mediante la reducción de dimensionalidad. Los datos serán divididos en dos tipos: estadísticos y complementarios. Los primeros representan el rendimiento por encima de la liga y los segundos aportan información complementaria que puede caracterizar al jugador, como la edad y lado de bateo. En el caso de las estadísticas, es posible encontrar algunas que ofrezcan información similar, esto no suele ser ni adecuado para los modelos ni para cumplir el objetivo de esta primera selección. Empíricamente, se pueden escoger las variables, sin embargo, una pequeña prueba de correlación puede ayudar para ver las relaciones entre estas. A su vez, un filtro preliminar será elegir estadísticas cuya dependencia de sucesos dentro del juego de béisbol sea menor[9]. En la matriz de correlación expuesta a continuación, se puede apreciar la fuerza que existe entre ellas.



Gráfica 5 Matriz de correlación entre estadísticas

La correlación más alta apreciada es la referente a **S** y **H** lo cuál tiene mucho sentido pues según la fórmula: **H= S+D+T+HR** y por lo general **S** es el tipo de evento o conexión más

[9] Por ejemplo, RBI (Runs Batted In), cuenta las carreras que el jugador impulsa, el problema es que para que esto ocurra, generalmente tienen que encontrarse otros compañeros de equipo en base y estos tener la habilidad de anotar la carrera dependiendo de la conexión. Suponiendo: Los jugadores A y B conectaron 8 hits cada uno en una semana. Sin embargo, el primero encontró corredores en base y pudo impulsar 3 carreras. El segundo solo pudo impulsar uno pues incluso cuando conectó home run, no hubo nadie en circulación en ninguno de los eventos. (Si un jugador conecta home run se impulsa a sí mismo). En otras palabras, son estadísticas que dependen tanto de la calidad del jugador como de la situación del juego y las habilidades del resto de jugadores.

frecuente en los jugadores. Incluso cuando **H** pudiera considerarse para permanecer entre las variables seleccionadas, el agrupar tantas estadísticas pudiera afectar el objetivo de definir al jugador en distintos aspectos. Para los otros atributos, la correlación no se consideró con la suficiente fuerza como para excluirlos.

Finalmente se definen los atributos que serán procesados:

- S (sencillos): Conexiones en las que se alcanza una base. Es una conexión común y puede representar a bateadores de corto alcance.
- D (dobles): Conexiones de dos bases. Puede exigir tanto poder como velocidad aunque no necesariamente en un alto grado.
- T (triples): Conexiones de tres bases. Jugadores con gran velocidad y con habilidad en el corrido de bases suelen conseguirlas con mayor facilidad.
- HR (home runs). Conexiones de 4 bases. La fuerza sale a relucir en esta métrica.
- SB: (bases robadas): La velocidad es primordial.
- BB (bases por bolas): Alcanzar una base sin tener que conectar. Puede reflejar la habilidad de seleccionar correctamente los lanzamientos para batear y de llegar a base sin necesidad de conectar hit.
- Age: Edad del jugador.
- B: Lado en el que se posiciona en el cajón de bateo.
- League_id: Ligas en las que participó.

4.3.2 Preprocesado

Los datos estadísticos son numéricos, unos con caracteres flotantes y otros enteros. Por otra parte, los atributos complementarios, excepto el que hace referencia a la edad, suelen ser categóricos. Una de las características de **Scikit-learn** es que, al menos los modelos a utilizar no suelen trabajar con valores categóricos. Bajo esta premisa se hace necesario en primera instancia convertir los datos a tipo numérico. La propia librería ofrece diferentes métodos para realizar esto dentro de su módulo *preprocessing*: **OneHotEncoder** y **LabelEncoder**.

OneHotEncoder, es una función que codifica los atributos categóricos creando una columna binaria para cada categoría y devolviendo una matriz dispersa o un arreglo. Por otra parte, **LabelEncoder**, codifica el valor categórico en clases entre 0 y n-1 clases. La gran diferencia entre estos algoritmos es que este último propicia un orden o peso a las clases. Como en este caso, una clase no tiene mayor importancia que otra, recurrir a este

método como solución final puede ser perjudicial al modelo. Conociendo esto, la decantación será codificar los valores categóricos hacia un arreglo numérico one-hot. Generalmente para obtener el arreglo correctamente, es necesario que cada categoría esté asociada a un valor numérico. Por tal motivo, se utilizó **LabelEncoder** para crear los valores numéricos correspondientes a cada clase y conformar finalmente un arreglo bidimensional con la clase y su valor, que se utilzaría en el método **OneHotEncoder.** Un ejemplo de la transformación de los identificadores de las combinaciones de liga (**League_id**), donde cada fila representa a la clase, se muestra a continuación (Tabla 3 Transformación de los identificadores de la liga en arreglos One-hot). Este proceso se realizó sobre este atributo y el referente al lado de bateo.

Leagues_id0	Leagues_id1	...	Leagues_id22	Leagues_id23
0	0	...	0	0
1	0	...	0	0
0	0	...	0	0
0	0	...	0	0

Tabla 3 Transformación de los identificadores de la liga en arreglos One-hot

La existencia de valores flotantes y enteros era también un problema presente. Contener números de diferentes escalas puede causar enormes inconvenientes para su interpretación por parte de los modelos. Una vía para escalar cada atributo a un rango es **MinMaxScaler**, función que preserva la forma original de la distribución y no reduce la importancia de los *outliers*. Con este método se acotarán todos los valores para cada atributo dentro del rango -1 a 1.

4.3.3 Reducción de dimensionalidad

En la lectura de un artículo de Chuan Xu (Xu, 2018) sobre el tema, se expone un fenómeno al que llaman la maldición de la dimensionalidad, y cito a modo de traducción: "*La maldición de la dimensionalidad ocurre porque la densidad de la muestra disminuye exponencialmente con el aumento de la dimensionalidad*". A raíz de la adición de los arreglos one-hot, el conjunto de datos aumentó a 37 dimensiones sobre un total de 493 muestras, lo cual puede ser considerada una proporción elevada. Esta situación puede provocar que el espacio de atributos se disperse y por tanto una disminución de la efectividad y rendimiento de los modelos, así como de sobreespecialización.

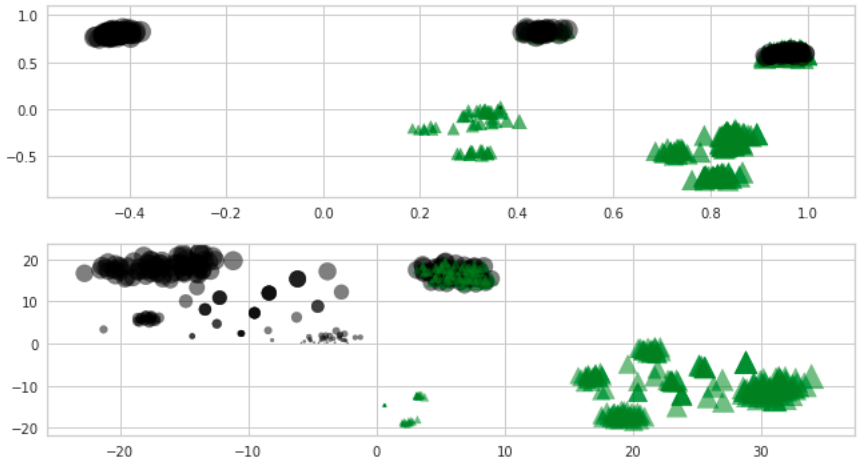
La reducción de la dimensionalidad es un proceso cuyo objetivo es reducir las dimensiones del conjunto de datos obteniendo las características principales. Puede ser

considerado como una combinación de un proceso de selección de características y extracción de características. El primero elimina las características redundantes o irrelevantes sin incurrir en la pérdida notable de información y el segundo se encarga de crear nuevas características en un espacio de menor dimensión.

Dos métodos son bien conocidos. El primero, Análisis de Componentes Principales (PCA, por sus siglas en inglés), descompone el conjunto de datos en componentes que expliquen una cantidad máxima de varianza, a mayor varianza mayor importancia. Mediante el cálculo de la matriz de covarianzas de los datos y el cálculo de los eigenvectors y sus eigenvalues[10], se seleccionan los n componentes. El segundo, t-distribuciones de Vecinos Estocásticos Incrustados (t-SNE, por sus siglas en inglés), convierte las similitudes entre los puntos de datos en probabilidades conjuntas, intentando minimizar la divergencia Kullback-Leibler entre las probabilidades de las bajas dimensiones incrustadas y los datos de alta dimensión (scikit-learn, s. f.). No es un modelo determinista por lo que se pueden obtener diferentes resultados.

La gráfica (Gráfica 6 Reducción a dos dimensiones mediante PCA (superior) y t-SNE (inferior)) muestra la reducción a dos componentes utilizando PCA y t-SNE. Los nuevos componentes mayormente parecen bien definidos, aunque algunas muestras se solapan. La diferencia visual principal es una dispersión de datos que genera t-SNE en una de las agrupaciones de una clase lo cual no ocurre con PCA, esto pudiera influenciar en el desenvolvimiento de los algoritmos de manera negativa. En conclusión, los componentes parecen más definidos después de la reducción con PCA, por este motivo se seguirá el proceso con los componentes que devuelve el método.

[10] Para una mejor comprensión de los conceptos de eigenvector y eigenvalues se puede recurrir a la explicación realizada por Victor Powell (Powell, 2009)



Gráfica 6 Reducción a dos dimensiones mediante PCA (superior) y t-SNE (inferior)

4.4 Conjunto de entrenamiento y prueba

Para trabajar adecuadamente sobre los modelos es necesario especificar un conjunto de entrenamiento y prueba. El conjunto de entrenamiento estaría dedicado a ayudar a la selección del mejor modelo. Realizando diferentes tipos de búsquedas de mejores parámetros y aplicando técnicas de validaciones cruzadas, estos datos nos mostrarán como se podría desenvolver cada modelo con el conjunto de datos. Por otra parte, el conjunto de prueba ayudará a corroborar estos resultados. Este conjunto servirá de medidor y calificador final de los modelos pues simula el comportamiento de estos con muestras no vistas.

Scikit-learn propicia la función **train_test_split** de **model_selection,** esta divide el conjunto de datos de manera aleatoria según el tamaño requerido. Estableciendo el parámetro **test_size** con valor 0.2, se define a utilizar un 20% de los datos totales como conjunto de prueba, por tanto, un 80% como conjunto de entrenamiento. El conjunto de entrenamiento y test lo conformarán 394 y 99 muestras aleatorias respectivamente.

5. Modelos

5.1 Agrupamiento

Para encontrar y agrupar a los jugadores similares es necesario buscar un modelo que identifique las características entre ellos y los agrupe. Se parte de la base que no se conocen de antemano los grupos a los que pertenece cada jugador por tanto la cantidad de clústeres a crear es desconocida. Esta situación referencia a algoritmos no supervisados y **Scikit-learn** provee varios con diferentes características.

Affinity propagation: encuentra agrupaciones sobre la base de maximizar la similitud total entre los puntos de datos y sus ejemplares. Los mensajes enviados entre puntos pueden corresponder tanto a la responsabilidad como a la disponibilidad. La responsabilidad es la evidencia acumulada de que una muestra *k* debe ser el ejemplar de otra *i* y la disponibilidad, es la evidencia acumulada de que la muestra *i* debe elegir la muestra *k* como su ejemplar. En conclusión, los ejemplares son elegidos por las muestras si son lo suficientemente similares a muchas muestras y si son elegidas por muchas muestras para ser representativas de sí mismas. La afinidad por defecto se basa en la *distancia euclidiana cuadrada negativa* entre puntos y es un algoritmo que puede ser útil cuando pueden existir muchos clústeres o el tamaño de estos es disparejo. Se puede encontrar información detallada sobre el algoritmo en la tesis **Affinity propagation: clustering data by passing messages** del Doctor en Filosofía Delbert Dueck (Dueck, 2009).

Mean-shift: asigna los puntos de datos a los grupos de manera iterativa moviéndose dentro de una densidad alta de puntos. Está basado en centroides y actualiza los candidatos para que el centroide sea la media de los puntos en determinada región. Una vez el cambio de centroide sea pequeño, el algoritmo se detendrá, lo que garantiza la convergencia. El algoritmo requiere múltiples búsquedas de vecinos más cercanos, lo que propicia una baja escalabilidad. Por otra parte, no requiere que se le especifique la cantidad de clústeres ya que es determinado por el algoritmo respecto a los datos. Suele ser utilizado en casos con muchos clústeres o que el tamaño de ellos no es parejo. Para un análisis a profundidad se puede recurrir al documento de Konstantinos G. Derpanis (Derpanis, 2005) Mean Shift Clustering.

DBSCAN: refiere al agrupamiento espacial basado en densidad de aplicaciones con ruido. Este algoritmo encuentra áreas de alta densidad que serán separados por áreas de poca densidad, entendiendo estas últimas como outliers. Utiliza dos parámetros principales denominados usualmente *eps y minPoints*. El primero puede entenderse como la densidad para formar los clústeres y especifica la cercanía que han de tener los puntos para formar el clúster. El segundo define la cantidad mínima de puntos para formar el clúster. Con una cantidad de puntos mayor o igual a *minPoints* y a una distancia *eps* de un punto arbitrario en la región se formará un clúster. El proceso continúa iterando sobre los puntos que no pertenecen a clústeres de manera aleatoria; si estos tienen menos puntos que *minPoints* a una distancia *eps*, entonces serán considerados como ruidos. A este algoritmo no es necesario que se le definan una cantidad de clústeres y estos pueden ser de diferentes tamaños. Para profundizar en el tema se puede recurrir al documento de Adriano Moreira, Maribel Y. Santos y Sofía Carneiro (Moreira, Santos, & Carneiro, 2005).

K-Means: Un algoritmo clásico cuyo objetivo es seleccionar los centroides que minimicen la distancia entre los puntos y el centroide. Esta distancia, muchas veces es medida como inercia o suma de cuadrados intra-clusters, responde a que tan coherente o cohesivo se encuentra el clúster. Requiere que los k-clústeres sean especificados y una vez hecho esto, el algoritmo establece los k centroides en el espacio de datos. Cada punto se va asignando al centroide más cercano e iterativamente se van optimizando y actualizando estos. El proceso termina ya sea cuando los centroides se estabilizaron o el número de iteraciones máximo se ha alcanzado. Es un algoritmo utilizado para propósitos generales y que garantiza la convergencia; sin embargo, se ha de tener en cuenta que es dependiente de la inicialización de los centroides. La sección de agrupamiento de la documentación de Scikit-learn (Scikit-learn, s. f.), provee cuantiosa información sobre el tema.

5.1.1 Affinity propagation

El método de **Scikit-learn**, **AffinityPropagation**, utiliza dos párametros principales para ayudar a definir los grupos, los cuales son *damping* y *preference.* El *damping* el cuál puede traducirse como factor de amortiguamiento, es la influencia que se tendrá sobre

los valores actuales respecto a los entrantes para evitar oscilaciones al momento de las actualizaciones. La preferencia, denotada por *preference,* influye en el número de clústeres a crear donde los puntos con mayores valores de preferencias tienen más probabilidades de ser seleccionados como ejemplares. En otras palabras, a mayor preferencia mayor cantidad de grupos a crear.

Para elegir el mejor modelo con este algoritmo, se realizó una búsqueda exhaustiva. Tomando como parámetros *damping* y *preference* se implementó un algoritmo que entrenara un modelo con cada combinación de parámetros. El rango de parámetros fue definido, en el caso del factor de amortiguamiento, por 8 valores aleatorios comprendidos entre -2 y -0.01. Para esto, se utilizó la función *randint* de numpy, la cual se encargará de devolver una cantidad especificada de valores comprendidos en el rango expuesto, intentando que estos sigan una distribución uniforme. Por otra parte, los valores de la preferencia se generaron mediante la función *uniform*, la cual propiciaría 5 números dentro del rango 0.5 y 0.99 que seguirían una distribución uniforme.

Una vez ejecutado el algoritmo, se muestran todas las combinaciones de hiperparámetros con sus respectivos coeficientes de silueta. La tabla (Tabla 4 Resultados de la búsqueda de mejores parámetros para Affinity Propagation) es una muestra de algunos de los resultados. El mejor coeficiente obtenido tiene un valor de aproximadamente 0.8056257034857032 y se obtiene combinando un damping de 0.8802440612938034 con una preferencia de -0.49538545513349996. Con estos hiperparámetros se efectúa la evaluación del modelo sobre el conjunto de prueba. El mejor modelo obtenido con Affinity Propagation, devuelve un coeficiente de silueta con datos no vistos igual a 0.7530711101521749. La gráfica (Gráfica 7 Grupos creados mediante Affinity propagation) muestra los grupos creados con los mejores hiperpar-ametros.

```
Clústers: 8, Preference: -1.421497851284349, Damping: 0.8802440612938034, Silueta: 0.7823153805290812
Clústers: 8, Preference: -1.421497851284349, Damping: 0.853031937154423, Silueta: 0.7823153805290812
Clústers: 48, Preference: -1.421497851284349, Damping: 0.6347084612729377, Silueta: 0.6783032547907649
Clústers: 8, Preference: -1.421497851284349, Damping: 0.9332242673278557, Silueta: 0.7823153805290812
Clústers: 8, Preference: -1.421497851284349, Damping: 0.7281198601834435, Silueta: 0.7823153805290812
Clústers: 11, Preference: -0.2027485654267407, Damping: 0.8802440612938034, Silueta: 0.7966508015441733
Clústers: 11, Preference: -0.2027485654267407, Damping: 0.853031937154423, Silueta: 0.7966508015441733
Clústers: 11, Preference: -0.2027485654267407, Damping: 0.6347084612729377, Silueta: 0.7966508015441733
Clústers: 11, Preference: -0.2027485654267407, Damping: 0.9332242673278557, Silueta: 0.7966508015441733
Clústers: 11, Preference: -0.2027485654267407, Damping: 0.7281198601834435, Silueta: 0.7966508015441733
Clústers: 10, Preference: -0.49538545513349996, Damping: 0.8802440612938034, Silueta: 0.8056257034857032
Clústers: 10, Preference: -0.49538545513349996, Damping: 0.853031937154423, Silueta: 0.8056257034857032
Clústers: 10, Preference: -0.49538545513349996, Damping: 0.6347084612729377, Silueta: 0.8056257034857032
Clústers: 10, Preference: -0.49538545513349996, Damping: 0.9332242673278557, Silueta: 0.8056257034857032
Clústers: 10, Preference: -0.49538545513349996, Damping: 0.7281198601834435, Silueta: 0.8056257034857032
Clústers: 15, Preference: -0.03798471499604572, Damping: 0.8802440612938034, Silueta: 0.7849152626863387
Clústers: 15, Preference: -0.03798471499604572, Damping: 0.853031937154423, Silueta: 0.7849152626863387
Clústers: 15, Preference: -0.03798471499604572, Damping: 0.6347084612729377, Silueta: 0.7849152626863387
Clústers: 15, Preference: -0.03798471499604572, Damping: 0.9332242673278557, Silueta: 0.7849152626863387
Clústers: 15, Preference: -0.03798471499604572, Damping: 0.7281198601834435, Silueta: 0.7849152626863387
```

Tabla 4 Resultados de la búsqueda de mejores parámetros para Affinity Propagation

Gráfica 7 Grupos creados mediante Affinity propagation

5.1.2 Mean-shift

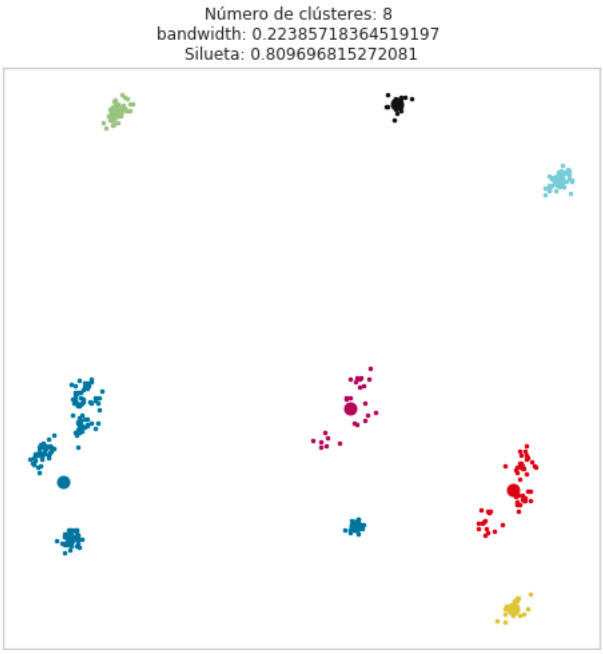
Este algoritmo utiliza el parámetro *bandwith* que establece el tamaño de la región donde buscar; a través de este, se pueden obtener más o menos clústeres. En la función MeanShift de Scikit-learn, aunque es posible especificarlo manualmente, se puede utilizar la función estimate_bandwidth, que es llamada en caso de no establecerse el *bandwith.* El estimador de la banda ancha, utiliza el cuantil (*quantile*) como parámetro principal.

Este parámetro puede especificarse de 0 a 1 y si se especifica 0.5 significa que se utilizará la mediana de todas las distancias.

Utilizando la función uniform se generan 20 valores aleatorios entre 0.01 y 0.4 como cuantiles para estimar el *bandwith*. Entrenando diferentes modelos según estas estimaciones se obtienen los coeficientes de la silueta (Tabla 5 Resultados de la búsqueda de mejores parámetros para Mean-shift). Cuantiles mayores a 0.4 propiciaban valores altos de *bandwith* y estos, generaban un solo clúster. No era viable calcular el coeficiente de la silueta en estos casos. Finalmente, se obtiene un 0.809696815272081 como mayor coeficiente mediante el modelo cuyo valor del bandwith es 0.22385718364519197. La gráfica (Gráfica 8 Agrupamientos mediante Mean-shift y los mejores hiperparámetros.) muestra los grupos creados. Se realiza entonces la evaluación del modelo sobre el conjunto de prueba alcanzando un valor de 0.7986126328099746.

```
Clústers: 16, quantile: 0.03805256810239698,Bandwidth: 0.04322593873424519, Silueta: 0.785285401790971
Clústers: 4, quantile: 0.21755581855679124,Bandwidth: 0.5177834944516603, Silueta: 0.7620660405160332
Clústers: 10, quantile: 0.08756904184844858,Bandwidth: 0.12912942532815777, Silueta: 0.7968546659129468
Clústers: 5, quantile: 0.21315222750438143,Bandwidth: 0.5141396945295674, Silueta: 0.7880987061304331
Clústers: 9, quantile: 0.10406420963918596,Bandwidth: 0.15507991378816974, Silueta: 0.7942120402810993
Clústers: 4, quantile: 0.2797986564889368,Bandwidth: 0.5995520766687503, Silueta: 0.7620660405160332
Clústers: 11, quantile: 0.07086483990150531,Bandwidth: 0.11506860809426551, Silueta: 0.7944105784685522
Clústers: 16, quantile: 0.041403125131626715,Bandwidth: 0.048756162499497775, Silueta: 0.785285401790971
Clústers: 51, quantile: 0.01818547560433278,Bandwidth: 0.02242585285667392, Silueta: 0.45692686873434485
Clústers: 8, quantile: 0.12312813059010708,Bandwidth: 0.22385718364519197, Silueta: 0.809696815272081
Clústers: 6, quantile: 0.1550445585465903,Bandwidth: 0.29669342515747293, Silueta: 0.788501641137111
Clústers: 5, quantile: 0.19905485013719001,Bandwidth: 0.5024891627796555, Silueta: 0.7880987061304331
Clústers: 4, quantile: 0.22449132073221728,Bandwidth: 0.5342900679441248, Silueta: 0.7620660405160332
Clústers: 4, quantile: 0.2585243424166812,Bandwidth: 0.5779048398141392, Silueta: 0.7620660405160332
Clústers: 4, quantile: 0.3140550160596858,Bandwidth: 0.7208691219968342, Silueta: 0.7620660405160332
Clústers: 4, quantile: 0.3205331496752705,Bandwidth: 0.7290907041948469, Silueta: 0.7620660405160332
Clústers: 4, quantile: 0.23953082311709548,Bandwidth: 0.5595014753915054, Silueta: 0.7620660405160332
Clústers: 28, quantile: 0.02989021674317042,Bandwidth: 0.03142172460670496, Silueta: 0.6993418185073847
Clústers: 5, quantile: 0.2093937725599022,Bandwidth: 0.5117987865840009, Silueta: 0.7880987061304331
Clústers: 9, quantile: 0.10048348540580673,Bandwidth: 0.14675659941537, Silueta: 0.7942120402810993
```

Tabla 5 Resultados de la búsqueda de mejores parámetros para Mean-shift



Gráfica 8 Agrupamientos mediante Mean-shift y los mejores hiperparámetros.

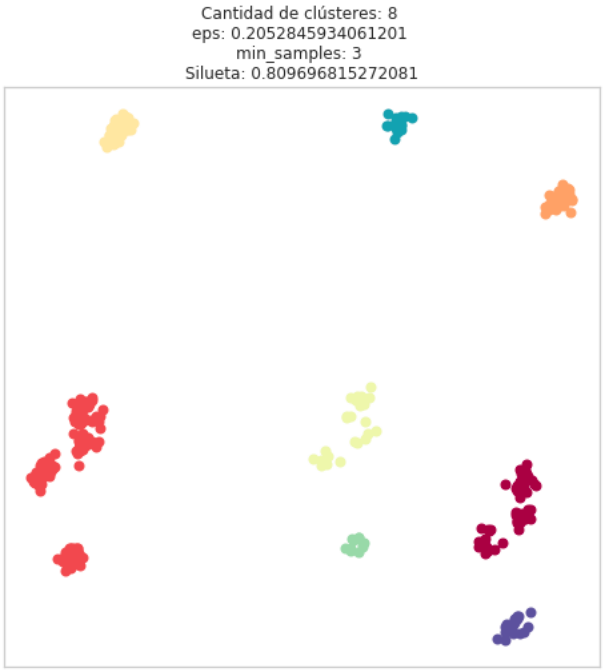
5.1.3 DBSCAN

La función DBSCAN() de Scikit-learn utiliza los parámetros *eps* como epsilon y *min_samples* para la mínima cantidad de puntos. Diez números aleatorios, siguiendo una distribución uniforme entre 0.03 y 0.6, serían los correspondientes a *eps*. Cinco valores enteros en un rango entre 3 y 40 serían seleccionados como *min_samples*. Realizando una búsqueda de rejilla con esos dos hiperparámetros y sus combinaciones, se obtienen los coeficientes de silueta para comprobar las definiciones de los clústeres. Con *eps* mayores a 0.6 se creaba un solo grupo por lo cual no era factible probar más allá. La tabla (Tabla 6 Resultados de la búsqueda de mejores parámetros para DBSCAN) muestra las diferentes combinaciones aplicadas a los modelos junto a los grupos creados y su coeficiente de silueta.

Con la mejor combinación, la cual presenta un coeficiente de 0.809696815272081, se utiliza el conjunto de prueba para ver el desempeño del modelo con ellos, obteniéndose 0.7986126328099746 como resultado del coeficiente. La gráfica (Gráfica 9 Agrupamientos con DBSCAN) presenta la creación de grupos con el conjunto de entrenamiento.

```
Clusters: 5, eps: 0.45896464302811835, min_samples:3 Silueta: 0.7880987061304331
Clusters: 4, eps: 0.45896464302811835, min_samples:35 Silueta: 0.7880987061304331
Clusters: 5, eps: 0.45896464302811835, min_samples:17 Silueta: 0.7880987061304331
Clusters: 5, eps: 0.45896464302811835, min_samples:5 Silueta: 0.7880987061304331
Clusters: 4, eps: 0.45896464302811835, min_samples:22 Silueta: 0.7880987061304331
Clusters: 4, eps: 0.5103439048800417, min_samples:3 Silueta: 0.7620660405160332
Clusters: 4, eps: 0.5103439048800417, min_samples:35 Silueta: 0.7622795763045073
Clusters: 4, eps: 0.5103439048800417, min_samples:17 Silueta: 0.7620660405160332
Clusters: 4, eps: 0.5103439048800417, min_samples:5 Silueta: 0.7620660405160332
Clusters: 4, eps: 0.5103439048800417, min_samples:22 Silueta: 0.7620660405160332
Clusters: 4, eps: 0.5403304594574654, min_samples:3 Silueta: 0.7620660405160332
Clusters: 4, eps: 0.5403304594574654, min_samples:35 Silueta: 0.7620660405160332
Clusters: 4, eps: 0.5403304594574654, min_samples:17 Silueta: 0.7620660405160332
Clusters: 4, eps: 0.5403304594574654, min_samples:5 Silueta: 0.7620660405160332
Clusters: 4, eps: 0.5403304594574654, min_samples:22 Silueta: 0.7620660405160332
Clusters: 9, eps: 0.19510315896375724, min_samples:3 Silueta: 0.7942120402810993
Clusters: 5, eps: 0.19510315896375724, min_samples:35 Silueta: 0.6115735788185115
Clusters: 8, eps: 0.19510315896375724, min_samples:17 Silueta: 0.7942120402810993
Clusters: 9, eps: 0.19510315896375724, min_samples:5 Silueta: 0.7942120402810993
Clusters: 6, eps: 0.19510315896375724, min_samples:22 Silueta: 0.6465275588158503
Clusters: 8, eps: 0.2052845934061201, min_samples:3 Silueta: 0.809696815272081
Clusters: 4, eps: 0.2052845934061201, min_samples:35 Silueta: 0.6423658969962835
Clusters: 7, eps: 0.2052845934061201, min_samples:17 Silueta: 0.809696815272081
Clusters: 8, eps: 0.2052845934061201, min_samples:5 Silueta: 0.809696815272081
Clusters: 5, eps: 0.2052845934061201, min_samples:22 Silueta: 0.6651133299800287
Clusters: 9, eps: 0.09953301092052357, min_samples:3 Silueta: 0.7942120402810993
Clusters: 5, eps: 0.09953301092052357, min_samples:35 Silueta: 0.5704328352452711
Clusters: 8, eps: 0.09953301092052357, min_samples:17 Silueta: 0.7731537155472057
Clusters: 9, eps: 0.09953301092052357, min_samples:5 Silueta: 0.7942120402810993
Clusters: 5, eps: 0.09953301092052357, min_samples:22 Silueta: 0.5841068297137068
```

Tabla 6 Resultados de la búsqueda de mejores parámetros para DBSCAN



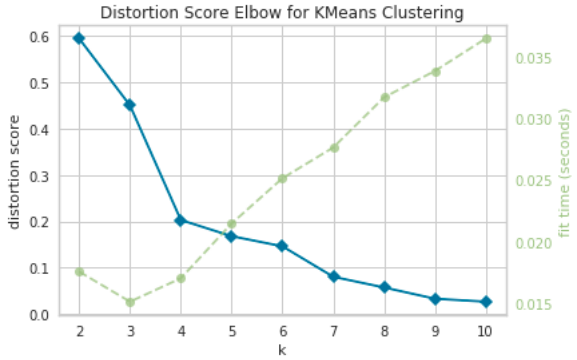
Gráfica 9 Agrupamientos con DBSCAN con los mejores hiperparámetros

5.1.4 K-Means

La tarea más común para la obtención de buenos resultados con este método es la selección de la cantidad *k* de clústeres. Uno de los métodos más conocidos es el denominado **Método del Codo**, donde después de una serie de ensayos iterativos sobre el modelo para distintos valores de *k,* se selecciona el punto donde el error deja de ser significante. Pudiera entenderse también como el punto donde la gráfica toma forma de

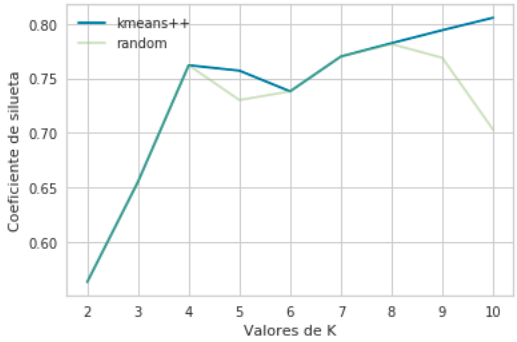
un codo. Este método utiliza generalmente como métrica la distorsión, que calcula la suma de distancias cuadradas de cada punto de su centro.

Utilizando **KElbowVisualizer** de la librería **yelowbirck**, función que implementa el método del codo, se realizan pruebas con valores para *k* desde 2 a 10 (Gráfica 10 Método del codo). La línea azul refleja la cantidad de clústeres respecto a la distorsión; pudiera considerarse 8 clústeres como valor óptimo. La línea verde manifiesta el tiempo de demora en el entrenamiento, a medida que aumenta la cantidad de clústeres lo suele hacer el tiempo.



Gráfica 10 Método del codo

A pesar de seleccionar un valor que pudiera ser el óptimo; mediante una búsqueda de rejilla se combinarán diferentes parámetros como la cantidad de clústeres y el tipo de inicialización. De esta manera se homogeneizará la forma de buscar y comparar los mejores resultados respecto a otros modelos. La función KMeans de Scikit-learn, provee dos formas de inicializar los clústeres: *kmeans++* y *random*. La primera inicializa los centroides de manera que se agilice la convergencia y la segunda lo hace aleatoriamente. Haciendo uso de *init*, parámetro encargado de estas inicializaciones y de la cantidad de clústeres k en un rango de 2 a 10, se encontrará la mejor combinación de estos y se le calculará el coeficiente de la silueta. A continuación, se muestra la gráfica (Gráfica 11 Coeficiente de la silueta para las combinaciones de hiperparámetros). La tabla (Tabla 7 Resultados de la búsqueda de mejores parámetros para Knn) muestra los mismos resultados de manera numérica.

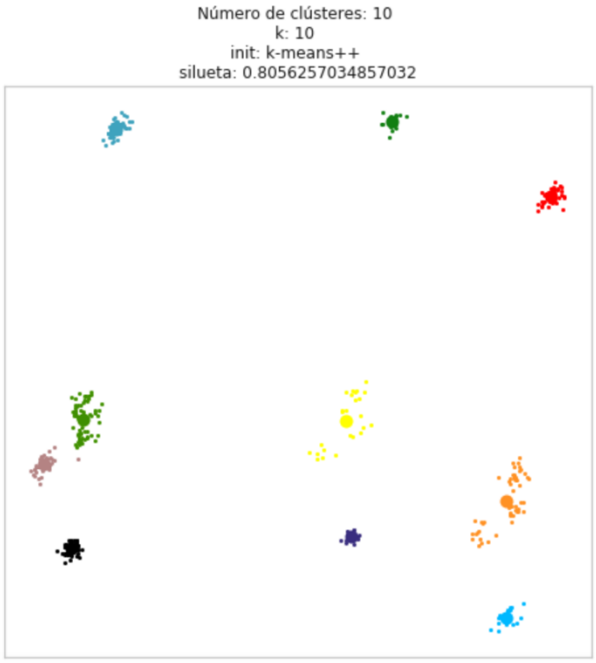


Gráfica 11 Coeficiente de la silueta para las combinaciones de hiperparámetros

```
Clusters: 2, Método: k-means++, Silueta: 0.563390023676846
Clusters: 2, Método: random, Silueta: 0.563390023676846
Clusters: 3, Método: k-means++, Silueta: 0.6551814922071754
Clusters: 3, Método: random, Silueta: 0.6551814922071754
Clusters: 4, Método: k-means++, Silueta: 0.7620660405160332
Clusters: 4, Método: random, Silueta: 0.7620660405160332
Clusters: 5, Método: k-means++, Silueta: 0.7571222667988744
Clusters: 5, Método: random, Silueta: 0.730161086387595
Clusters: 6, Método: k-means++, Silueta: 0.7381464940159943
Clusters: 6, Método: random, Silueta: 0.7381464940159943
Clusters: 7, Método: k-means++, Silueta: 0.7700724820382254
Clusters: 7, Método: random, Silueta: 0.7700724820382254
Clusters: 8, Método: k-means++, Silueta: 0.7823153805290812
Clusters: 8, Método: random, Silueta: 0.7814861452428293
Clusters: 9, Método: k-means++, Silueta: 0.7942120402810993
Clusters: 9, Método: random, Silueta: 0.7688893270533156
Clusters: 10, Método: k-means++, Silueta: 0.8056257034857032
Clusters: 10, Método: random, Silueta: 0.703087339343883
```

Tabla 7 Resultados de la búsqueda de mejores parámetros para Knn

Los mejores resultados se obtienen de tal forma que se generen 10 clústeres con el método de incialización k-means++ (Gráfica 12 Agrupamientos de K-means con los mejores hiperparámetros). Con este parámetros se procede a probar el modelo con datos no vistos. El coeficiente de la silueta con el conjunto de prueba arroja un 0.7530711101521749.



Gráfica 12 Agrupamientos de K-means con los mejores hiperparámetros

5.1.5 Selección del modelo de agrupamiento

Luego de haber realizado las pruebas con diferentes algoritmos obtuvimos un modelo representativo de cada uno, los cuales compararemos por el coeficiente de la silueta (Tabla 8 Comparación entre modelos de agrupamiento). Un coeficiente mayor indica mejor definición en los clústeres.

Algoritmo	Parámetros	Clústeres	Coeficiente de silueta sobre el conjunto de entrenamiento	Coeficiente de silueta sobre el conjunto de prueba
Affinity Propagation	damping: 0.8802440612938034 preferencia 0.49538545513349996	10	0.805625703	0.75307111
Mean Shift	quantile: 0.12312813059010708 bandwidth: 0.22385718364519197	8	0.809696815	0.798612633
DBSCAN	eps: 0.2052845934061201 min_samples: 3	8	0.809696815	0.798612633
K-means	k: 10 init: k-means++	10	0.805625703	0.75307111

Tabla 8 Comparación entre modelos de agrupamiento

Los modelos, logran definir de manera similar los clústeres con el conjunto de entrenamiento. Coeficientes sobre 0.8 aproximadamente, con 8 o 10 clústeres son resultados sobresalientes para todos. Por otra parte, con el conjunto de prueba, algunos lograron definir mejor los clústeres, son los casos de DBSCAN y Mean Shift. Ambos crean la misma cantidad de grupos y la misma definición tanto en conjunto de entrenamiento como de prueba con sus mejores hiperparámetros. Hasta este punto, cualquiera de los dos modelos puede ser elegido sin mayor problema; sin embargo, para aportar un elemento adicional para decantarse por uno de ellos fuera de la aleatoriedad u opinión, se realizará una prueba de rendimiento.

El procedimiento será sencillo, mediante el método mágico de **Python %%timeit**, se evaluará el tiempo de demora de cada modelo en formar los grupos. Este método mágico calcula el tiempo de demora tanto de una línea específica como de un conjunto de expresiones. Automáticamente **%%timeit** realizará iteraciones sobre el código y arrojará el tiempo promedio de demora en cada iteración. Aunque puede establecerse el número de corridas que se desean efectuar sobre cada línea o conjunto de código, el método mágico logra identificar la cantidad requerida por el código.

A través de 7 corridas de 100 repeticiones cada una se obtiene que **DBSCAN** demora de forma promedio 5.16 ms ± 280 µs y Mean-Shift 30.9 ms ± 1.48 ms, en el ordenador de prueba. El resultado no era sorprendente, en un estudio sobre el rendimiento de los modelos de agrupamiento realizado por Leland McInnes John Healy y Steve Astels (McInnes, Healy, & Astels, 2016) como parte de la documentación de **HDBSCAN**, **DBSCAN** aparece como los mejores en este aspecto. En ese estudio aparecen disímiles algoritmos incluyendo varios de los presentados en este documento. Teniendo en cuenta lo expuesto, se selecciona al modelo creado con DBSCAN para agrupar a los jugadores con características similares.

5.2 Clasificación

Una vez definidos los grupos se hace necesario encontrar un algoritmo que logre clasificar las nuevas muestras. Como en el caso de los modelos de agrupamiento, **Scikit-learn** provee múltiples algoritmos que pueden encargarse de hacer la clasificación. Un estudio sobre los que provee la librería nos permitió seleccionar dos como modelos principales:

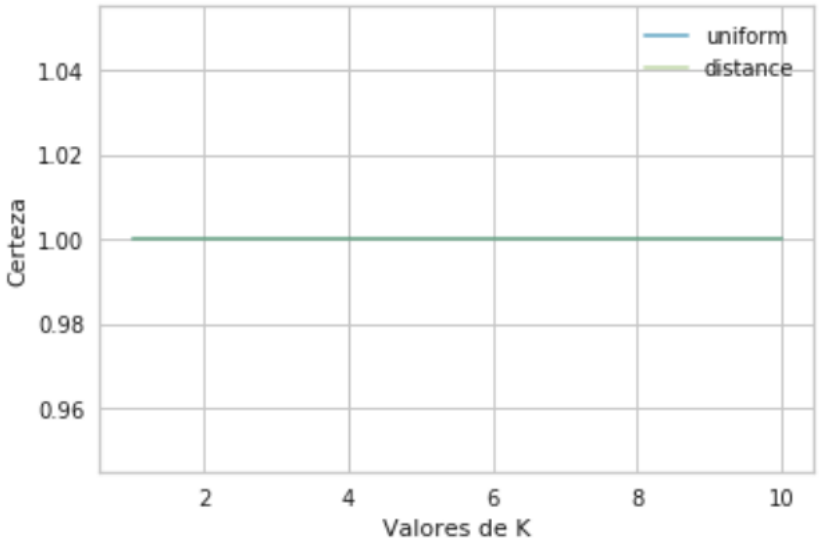
Nearest neighbors (K-nn): el funcionamiento de este modelo es bastante sencillo en su base pues se encarga de encontrar una cantidad definida de ejemplares o vecinos que se encuentren más cerca del nuevo punto. El número de vecinos puede ser definido manualmente o variar dependiendo de la densidad de puntos; en cualquier caso, la correcta elección de los k vecinos depende en gran medida del conjunto de datos. Aunque la distancia entre la nueva muestra y los ejemplares puede ser medida por distintas métricas, la más común es la distancia euclidiana. Este algoritmo no paramétrico, cae dentro de los considerados como de aprendizaje vago, es simple aunque sensible a outliers y puede utilizarse tanto en problemas de clasificación como de regresión.

Support Vector Machines (SVM): las Máquinas de soporte vectorial son algoritmos que construyen un hiperplano en un espacio de n dimensiones. El objetivo es encontrar un plano que maximice la distancia entre los puntos de las clases. Los algoritmos han de encontrar los puntos más cercanos a la línea que divide las clases; estos serán los vectores de soportes, e intentarán maximizar el margen entre la línea y el vector de soporte. Son considerados no paramétricos y no lineales; no obstante, con la utilización de las denominadas funciones kernel como la polinomial y la de base radial Gaussiana, se pueden solucionar problemas fuera de la linealidad. Los SVMs suelen ser efectivos en espacios de alta dimensión incluso cuando el número de dimensiones es mayor al número de muestras; sin embargo, realizar ajustes es primordial para evitar el sobre entrenamiento.

5.2.1 Nearest neighbors

Una de las claves para alcanzar los mejores resultados de este modelo es la selección de los *k* vecinos. El clasificador a utilizar es **KNeighborsClassifier** y además de los n*_neighbors,* pueden establecerse otros parámetros como *weights*(importancia que se le asigna a cada punto) y *algorithm* (algoritmo para calcular los vecinos más cercanos). Para optimizar el modelo se realizará la denominada prueba de rejilla intentando obtener los valores óptimos.

Utilizando el conjunto de entrenamiento y mediante **GridSearchCV** se entrenará un modelo **KNeighborsClassifier** para cada combinación, evaluándose mediante **cross validation** con un número de *splits* igual a 3. GridSearchCV es una función que realiza una búsqueda exhaustiva según parámetros especificados y un estimador. Los parámetros de la rejilla lo conformarán *weights* con los pesos *uniform* y *distance* y por otra parte los *n_neighbors*, con valores de 1 a 10. El hiperparámetro *algorithm* será seleccionado de manera automática por el modelo de acuerdo a los datos que se les propicie. Una vez terminada la búsqueda exhaustiva se obtiene la siguiente gráfica (Gráfica 13 Certeza del modelo según el valor de k y el peso otorgado a los vecinos) que muestra, debido a la gran definición y separación de los clústeres una certeza perfecta.



Gráfica 13 Certeza del modelo según el valor de k y el peso otorgado a los vecinos

La siguiente tabla muestra en más detalle los resultados de la búsqueda:

	mean_test_score	std_test_score	params
0	1.0	0.0	{'n_neighbors': 1, 'weights': 'uniform'}
1	1.0	0.0	{'n_neighbors': 1, 'weights': 'distance'}
2	1.0	0.0	{'n_neighbors': 2, 'weights': 'uniform'}
3	1.0	0.0	{'n_neighbors': 2, 'weights': 'distance'}
4	1.0	0.0	{'n_neighbors': 3, 'weights': 'uniform'}
5	1.0	0.0	{'n_neighbors': 3, 'weights': 'distance'}
6	1.0	0.0	{'n_neighbors': 4, 'weights': 'uniform'}
7	1.0	0.0	{'n_neighbors': 4, 'weights': 'distance'}
8	1.0	0.0	{'n_neighbors': 5, 'weights': 'uniform'}
9	1.0	0.0	{'n_neighbors': 5, 'weights': 'distance'}
10	1.0	0.0	{'n_neighbors': 6, 'weights': 'uniform'}
11	1.0	0.0	{'n_neighbors': 6, 'weights': 'distance'}
12	1.0	0.0	{'n_neighbors': 7, 'weights': 'uniform'}
13	1.0	0.0	{'n_neighbors': 7, 'weights': 'distance'}
14	1.0	0.0	{'n_neighbors': 8, 'weights': 'uniform'}
15	1.0	0.0	{'n_neighbors': 8, 'weights': 'distance'}
16	1.0	0.0	{'n_neighbors': 9, 'weights': 'uniform'}
17	1.0	0.0	{'n_neighbors': 9, 'weights': 'distance'}
18	1.0	0.0	{'n_neighbors': 10, 'weights': 'uniform'}
19	1.0	0.0	{'n_neighbors': 10, 'weights': 'distance'}

Tabla 9 Resultados de la búsqueda de rejilla

Según los mejores hiperparámetros se puede esperar una certeza de 1. Entrenando el modelo y aplicando estos resultados sobre el conjunto de prueba (X_test) se obtiene una certeza igualmente perfecta.

5.2.2 Support Vector Machine

Con este modelo se realizaron 3 pruebas diferentes enfocadas en los *kernels*. Todos los casos se efectuaron bajo el método de *cross validation* utilizando **RandomizerSearchCV**

y 10 combinaciones de parámetros elegidos al azar. **RandomizerSearchCV**, de manera aleatoria selecciona hiperparámetros para utilizarlos sobre un estimador. La diferencia entre la búsqueda aleatoria y la de rejilla es que esta última, prueba todas las posibles combinaciones entre los parámetros definidos y la primera prueba combinaciones aleatorias. No obstante, ambos métodos optimizan la búsqueda de mejores parámetros mediante validación cruzada.

La función aplicada para generar los modelos fue **SVC** y los valores aleatorios fueron generados mediante la función de **uniform** (conjunto de números que siguen una distribución uniforme). Se emplearon los parámetros, *C* para la penalidad del error y *gamma* como coeficiente que define la influencia de cada punto de entrenamiento; cuyos valores según el rango establecido, seguían una distribución uniforme. Para el kernel poly, se agregó el hiperparámetro *degree*, que indica el grado del polinomio y utilizaría números enteros aleatorios entre 2 y 5 creados con la función **random**. Al utilizar el conjunto de entrenamiento en las diferentes pruebas, se obtuvieron los resultados que se muestran a continuación, primero de acuerdo a la función *kernel rbf* (Radial Basis Function) (Tabla 10 Búsqueda aleatoria con SVM de kernel rbf), luego con *poly* (Polynomial) (Tabla 11 Búsqueda aleatoria con SVM de kernel poly) y finalmente con la función *linear* (Tabla 12 Búsqueda aleatoria con SVM de kernel linear).

	mean_test_score	std_test_score	params
0	1.0	0.0	{'C': 180.41524206175126, 'gamma': 0.08872005408131084}
1	1.0	0.0	{'C': 197.2225537113177, 'gamma': 0.08265994394715771}
2	1.0	0.0	{'C': 220.56745428510922, 'gamma': 0.03869444294249076}
3	1.0	0.0	{'C': 232.3398928348032, 'gamma': 0.03113778741641421}
4	1.0	0.0	{'C': 374.80469008812554, 'gamma': 0.05127203900924792}
5	1.0	0.0	{'C': 117.10634757340866, 'gamma': 0.09095745732745686}
6	1.0	0.0	{'C': 192.9456106866057, 'gamma': 0.055355286111398866}
7	1.0	0.0	{'C': 454.23605548227346, 'gamma': 0.06342379959139921}
8	1.0	0.0	{'C': 59.44902035418203, 'gamma': 0.09498321236134752}
9	1.0	0.0	{'C': 314.85402653570895, 'gamma': 0.034490561465708613}

Tabla 10 Búsqueda aleatoria con SVM de kernel rbf

	mean_test_score	std_test_score	params
0	0.406091	0.003386	{'C': 281.99067081318964, 'degree': 3, 'gamma': 0.035088301549235715}
1	0.406091	0.003386	{'C': 393.47780764325375, 'degree': 4, 'gamma': 0.052950185476498116}
2	0.908629	0.004799	{'C': 226.26826953353802, 'degree': 3, 'gamma': 0.09640939149023008}
3	0.931472	0.015344	{'C': 388.5804642644685, 'degree': 2, 'gamma': 0.0601914404419070 1}
4	0.512690	0.004945	{'C': 369.13994939171073, 'degree': 3, 'gamma': 0.043556336571835164}
5	0.908629	0.004799	{'C': 275.7447867559102, 'degree': 2, 'gamma': 0.06177091118319272}
6	0.873096	0.016061	{'C': 197.98891445862176, 'degree': 2, 'gamma': 0.04396027434485468}
7	0.807107	0.006473	{'C': 278.1612990418277, 'degree': 3, 'gamma': 0.06973724657578066}
8	0.913706	0.008509	{'C': 187.26372640139743, 'degree': 2, 'gamma': 0.07648725289085606}
9	0.807107	0.006473	{'C': 325.34752126651773, 'degree': 3, 'gamma': 0.056342770078706805}

Tabla 11 Búsqueda aleatoria con SVM de kernel poly

	mean_test_score	std_test_score	params
0	1.0	0.0	{'C': 493.4205994811673, 'gamma': 0.08488980864459342}
1	1.0	0.0	{'C': 63.33134060163342, 'gamma': 0.09308418826173724}
2	1.0	0.0	{'C': 435.94818103106417, 'gamma': 0.052883805712607215}
3	1.0	0.0	{'C': 296.63771787246463, 'gamma': 0.04090027038701303}
4	1.0	0.0	{'C': 28.380819411015665, 'gamma': 0.03451972416459009}
5	1.0	0.0	{'C': 402.42672429900574, 'gamma': 0.0014632023004602858}
6	1.0	0.0	{'C': 167.74958584557208, 'gamma': 0.040816869359094336}
7	1.0	0.0	{'C': 269.6978014689614, 'gamma': 0.09298556164127605}
8	1.0	0.0	{'C': 174.17299718298062, 'gamma': 0.03569532018962277}
9	1.0	0.0	{'C': 369.7506240548742, 'gamma': 0.04622179408898072}

Tabla 12 Búsqueda aleatoria con SVM de kernel linear

Excepto los modelos creados con la función kernel poly, las certezas de ellos con los mejores hiperparámetros fueron perfectas. El hiperparámetro *degree* afectó en gran medida la certeza de los modelos con función kernel poly. Tomando los modelos con mejores hiperparámetros para cada una de las funciones kernel, se utiliza el conjunto de prueba para clasificar las muestras no vistas. En todos los casos, la certeza fue perfecta excepto el modelo con kernel *poly* cuya precisión fue de 0.93.

5.2.3 Selección del modelo de clasificación

La medida a tomar en cuenta para la selección del modelo es la certeza. Los modelos Knn, kernel *linear* y *rbf* clasificaron de manera perfecta tanto el conjunto de entrenamiento como el de prueba. Otras métricas existen para medir el rendimiento de determinado algoritmo pero utilizan, al igual que la certeza, la tasa de verdaderos y falsos tanto positivos como negativos de alguna forma. Teniendo en cuenta esto, no serán de utilidad algunas de las otras métricas para decantarse por algún modelo.

Cualquiera de estos modelos puede ser seleccionado para clasificar los nuevos jugadores; la definición y separación de los grupos hace que el proceso de clasificación sea sencillo. Ante esta situación, seleccionar el modelo de menor complejidad, sobre todo en el aspecto

computacional, pudiera ser una estrategia válida. Al igual que con los modelos de agrupación mediremos el tiempo de ejecución de los algoritmos sobre los conjuntos de entrenamiento (Tabla 13 Comparación de tiempo de ejecución de los modelos). El método mágico **%%timeit** realizó 1000 iteraciones y 7 corridas sobre el código correspondiente a los modelos. El modelo Knn cuya certeza es perfecta sobre el conjunto de datos y presenta menor coste computacional será el encargado de clasificar a los jugadores.

Modelo	Tiempo promedio por iteración
Knn	374 µs ± 4.8 µs
SVM con kernel rbf	851 µs ± 15.1 µs
SVM con kernel poly	756 µs ± 24.1 µs
SVM con kernel linear	709 µs ± 21.5 µs

Tabla 13 Comparación de tiempo de ejecución de los modelos

6. Predicciones

Las predicciones se realizan sobre el conjunto de prueba, para realizar estas se necesitan dos conjuntos de datos. El primero tiene que contener los valores totales para cada jugador antes del debut y el segundo los valores totales el año del debut. Ambos han de tener tanto el grupo al que pertenecen como la referencia si corresponden al conjunto de entrenamiento o prueba. A los grupos del conjunto de entrenamiento se les calcula la variación que sufrieron el año del debut respecto al año anterior en cada estadística. Al final, cada métrica predicha sobre el conjunto de prueba será resultado de los valores del año antes del debut aumentado a la variación positiva o negativa que suele sufrir generalmente el grupo de los jugadores similares. La siguiente tabla (Tabla 14 Diferencia porcentual de los grupos el año antes y después del debut) contiene las variaciones porcentuales, generalmente negativas, de cada grupo (Group) en algunas estadísticas. Cada número representa el porciento que disminuirá (en estos casos) la estadística del jugador el año del debut.

Group	G	PA	AB	R	RBI	S	2B	3B	HR	SB	BB	SO
0	-61.911171	-66.729918	-66.766573	-67.535854	-67.343750	-69.539249	-66.000000	-78.461538	-49.367089	-79.797980	-65.166341	-54.994629
1	-73.443215	-78.088869	-77.634451	-80.656967	-81.029357	-80.305495	-81.243088	-83.360000	-75.462107	-89.811133	-81.829203	-70.906079
2	-81.184173	-85.521415	-85.503981	-88.509022	-86.288265	-87.264673	-87.625899	-92.592593	-77.966102	-97.233202	-85.607196	-79.686210
3	-71.280113	-78.234248	-77.631787	-80.884354	-80.974309	-80.177979	-78.669043	-76.699029	-75.267539	-90.566038	-82.455540	-72.730085
4	-56.132473	-66.735510	-65.928189	-67.948718	-68.560397	-70.656211	-74.114441	-63.636364	-55.102041	-73.880597	-72.500000	-58.396569
5	-84.256183	-88.575589	-88.115841	-89.801085	-91.343405	-87.592643	-89.003083	-92.537313	-91.379310	-92.656250	-91.559981	-86.659904
6	-71.607754	-76.705461	-75.890579	-79.074733	-79.980934	-78.947368	-76.744186	-81.553398	-75.449102	-82.009346	-82.200957	-66.803069
7	-64.313885	-71.501486	-70.807476	-74.673629	-75.725594	-74.290579	-78.851964	-77.313433	-66.573034	-85.356696	-75.909879	-63.556800

Tabla 14 Diferencia porcentual de los grupos el año antes y después del debut

Teniendo estos valores se efectúan las predicciones donde a cada estadística acumulativa del jugador antes del debut se le sumará el valor correspondiente a su grupo en esa métrica. Con los totales predichos se calcularán las estadísticas porcentuales. La tabla (Tabla 15 Predicción de estadísticas totales y porcentuales) muestra las predicciones finales de algunos jugadores.

Name	G	PA	AB	R	RBI	S	2B	3B	HR	SB	BB	SO	HBP	SF	SH	Group	H	AVE	OBP	SLG	OPS
Ozzie Albies	53	205	183	27	17	34	11	2	3	6	18	43	2	1	1	0	50	0.273	0.343	0.404	0.747
Brian Anderson	36	121	108	13	12	18	4	0	3	0	11	28	1	1	0	1	25	0.231	0.306	0.352	0.658
Christian Arroyo	34	113	106	11	9	18	8	0	1	0	5	20	0	1	1	3	27	0.255	0.286	0.358	0.644
Harrison Bader	35	114	104	14	11	16	4	1	5	1	7	38	2	1	0	1	26	0.250	0.307	0.452	0.759
Franklin Barreto	33	115	107	13	10	19	5	1	3	3	7	27	1	0	0	1	28	0.262	0.313	0.411	0.724
Cody Bellinger	51	158	140	21	22	20	4	0	12	2	16	39	1	1	0	4	36	0.257	0.335	0.543	0.878
Jorge Bonifacio	36	122	111	16	16	18	4	1	5	1	9	38	1	0	1	1	28	0.252	0.314	0.441	0.755
Lewis Brinson	28	96	91	12	12	13	5	1	4	2	4	25	0	1	0	1	23	0.253	0.281	0.462	0.743
Jaycob Brugman	58	203	185	25	27	30	9	3	5	2	14	50	1	3	0	4	47	0.254	0.305	0.416	0.721
Johan Camargo	36	115	108	10	9	17	6	1	1	0	4	27	0	0	3	6	25	0.231	0.259	0.333	0.592
Victor Caratini	33	111	99	12	9	18	6	0	1	0	10	27	1	1	0	6	25	0.253	0.324	0.343	0.667
Matt Chapman	36	129	115	18	18	11	5	1	9	1	12	50	1	1	0	1	26	0.226	0.302	0.522	0.824

Tabla 15 Predicción de estadísticas totales y porcentuales

Una vez se tienen los valores finales es necesario conocer que tan bien realiza las predicciones el sistema. Una de las métricas utilizadas es la denominada MSE. El Error Cuadrático Medio (MSE, por sus siglas en inglés), muestra la diferencia entre el valor predicho y el valor real, permitiendo verificar el comportamiento de los resultados. Sin embargo, los resultados tendrían más relevancia si se pudieran comparar con modelos ya existentes.

Durante la etapa investigativa del estado del arte, se encontraron comparaciones basadas en MSE de diferentes modelos donde se incluyen redes neuronales. El autor de estas pruebas utiliza como base una implementación de Marcel, el modelo más simple y compara los demás modelos mediante el MSE relativo a este. A modo de ejemplo, si Marcel presentaba un MSE de 0.0234 y la red neuronal 0.0123, en la tabla se iba a mostrar un 5.25% aclarando que la red neuronal mejoraba en ese porciento a Marcel. A la tabla (Tabla 16 Comparación relativa del coeficiente MSE respecto a Marcel) en cuestión se le añadirá el sistema creado, que combina un algoritmo de agrupamiento, clasificación y cálculos. Para identificarlo lo llamaremos Local-Predictor. Los nombres del resto de los modelos sólo hacen referencia a su núcleo de implementación.

Modelo/Sistema	AVE	SLG	OBP
Marcel (recreado)	0.03268	0.04738	0.02446
Random Forest	2.49%	1.97%	5.06%
Recurrent Neural Network	9.24%	8.78%	9.73%
Deep Neural Network	9.91%	8.78%	9.73%
Local-Predictor	91.9%	74.48%	89.88%

Tabla 16 Comparación relativa del coeficiente MSE respecto a Marcel

El sistema creado en este trabajo disminuye significativamente los errores en las predicciones de cualquiera de los otros presentados. Acotar que, a diferencia de **Local-Predictor**, los otros sistemas utilizaron datos no necesariamente de debutantes sino de todos los jugadores.

Como otra prueba comparativa sobre las mismas estadísticas, esta vez respecto al Error Medio Absoluto (MAE, por sus siglas en inglés), se presenta la siguiente tabla (Tabla 17 Comparación relativa del valor MAE de distintos modelos respecto a Marcel).

Modelo/Sistema	AVE	SLG	OBP
Marcel (recreado)	0.1207	0.1519	0.1045
Random Forest	-1.54%	-1.07%	-0.86%
Recurrent Neural Network	4.65%	4.75%	4.54%
Deep Neural Network	4.51%	4.69%	5.00%
Local-Predictor	66.52%	42.14%	61.84%

Tabla 17 Comparación relativa del valor MAE de distintos modelos respecto a Marcel

Como la vez anterior, **Local-Predictor** mejora el error relativo respecto a Marcel(recreado) y los demás modelos. Valores por encima del 40% en todos los casos representan las mejoras respecto a Marcel; a su vez, son mayores diferencias que los presentados por el resto de los sistemas en relación a los Errores Medios Absolutos que presenta el modelo recreado de Marcel.

7. Conclusiones

Mediante este trabajo se ha podido obtener un sistema capaz de predecir el rendimiento de los jugadores de las Ligas Menores de Béisbol el año de su debut en las Ligas Mayores de Béisbol. Gracias al análisis histórico estadístico de los jugadores antes y después del debut se ha podido reconocer la línea base estadística de los jugadores que son llamados a debutar en la MLB. Se han podido agrupar a los jugadores en grupos, de acuerdo a sus características. Teniendo en cuenta las predicciones de los juegos para un jugador, se puede predecir qué jugadores pudieran permanecer en la liga, al menos en su primer año.

Aunque se han obtenido resultados satisfactorios, predecir valores futuros con mayor certeza puede requerir el uso de otras técnicas de predicción como modelos de regresión. El análisis y comprensión de los datos en este tipo de trabajos ha de ser profundo para poder entender su naturaleza. El preprocesado de los datos ha sido clave en el éxito de los resultados. La planificación ha sido seguida adecuadamente, sin embargo, han tenido que introducirse cambios necesarios que han llevado a reestructuración de código, búsqueda y lectura de información adicional así como cambios de contexto. La flexibilidad de la metodología de desarrollo ha permitido realizar esto de manera transparente.

8. Recomendaciones

Los resultados arrojados muestran que este acercamiento va tomando el buen camino. Las comparaciones sitúan al nuevo sistema en una posición ventajosa disminuyendo los errores en las predicciones. A pesar de esto, existen consideraciones que pudieran mejorar los resultados:

- Durante todo el proceso de selección de algoritmos se utilizaron los datos de las diferencias de las estadísticas respecto a la liga, por tanto, intentar pronosticar esas diferencias en la MLB para cada debutante pudiera otorgar una diferente perspectiva y quizás una mejora en los resultados. Para esto, pudiera utilizarse un tercer modelo, tentativamente de regresión, para predecir los valores de la liga y sobre estos aplicar las diferencias.
- Incorporar información complementaria, como altura, peso, procedencia o cualquier característica que pudiera detallar a los jugadores.
- Incluso cuando los números pueden ser más específicos para comparar y analizar jugadores, se puede agregar una interpretación de estos realizada por el propio sistema utilizando lenguaje natural a modo de reporte. Por ejemplo: Jugador_A, se proyecta como un jugador sobresaliente, presenta gran contacto y poder, aunque carece de velocidad.

9. Glosario

D: Doble. Conexión donde se alcanzan dos bases

Draft: Proceso de selección de jugadores.

Embasarse: Alcanzar una base

HR: Home Run. Conexión donde se recorren todas las bases

MLB: Major League Baseball o Ligas Mayores de Béisbol

MiLB: Minor League Baseball o Ligas Menores de Béisbol

Pelotero: Beisbolista, jugador de béisbol.

Pitcher: Lanzador.

R: Carrera anotada.

RBI: Carrera impulsada

S: Sencillo. Conexión donde se alcanza una base

SB: Bases Robadas. Alcanzar una base a la que inicialmente por regla no tiene derecho.

SF: Fly de sacrificio.

SH: Golpe de sacrificio.

T: Triple. Conexión donde se alcanzan tres bases

10. Bibliografía

Baseball Prospectus. (s. f.). Baseball Prospectus | Glossary. *Baseball Prospectus*. Recuperado marzo 24, 2019, a partir de https://legacy.baseballprospectus.com/glossary/index.php?mode=viewstat&stat=476

Baseball-Reference. (s. f.). Marcel the Monkey Forecasting System. *Baseball-Reference.com*. Recuperado marzo 24, 2019, a partir de https://www.baseball-reference.com/about/marcels.shtml

Bernier, D. (s. f.). What is Minor League Baseball? *Pro Baseball Insider*. Recuperado marzo 24, 2019, a partir de http://probaseballinsider.com/what-is-minor-league-baseball/

Calzada, D. (2018). *DEEPBALL: MODELING EXPECTATION AND UNCERTAINTY IN BASEBALL WITH RECURRENT NEURAL NETWORKS*. University of Illinois at Urbana-Champaign, Urbana, Illinois.

Chandler, G., & Stevens, G. (2012). An Exploratory Study of Minor League Baseball Statistics.

Cross, J., Davidson, D., & Rosenbloom, P. (s. f.). Steamer page. STEAMER projections, . Recuperado marzo 24, 2019, a partir de http://steamerprojections.com/blog/about-2/

Derpanis, K. (2005, agosto 15). Mean Shift Clustering.

Druschel, H. (2016, febrero 22). druschel. *Beyond the Box Score*. Recuperado marzo 24, 2019, a partir de https://www.beyondtheboxscore.com/2016/2/22/11079186/projections-marcel-pecota-zips-steamer-explained-guide-math-is-fun

Dueck, D. (2009). AFFINITY PROPAGATION: CLUSTERING DATA BY PASSING MESSAGES, 154.

McInnes, L., Healy, J., & Astels, S. (2016). Benchmarking Performance and Scaling of Python Clustering Algorithms — hdbscan 0.8.1 documentation. Recuperado junio 8, 2019, a partir de https://hdbscan.readthedocs.io/en/latest/performance_and_scalability.html

Mitchell, C. (2014, diciembre 30). KATOH: Forecasting Major League Hitting with Minor League Stats. *The Hardball Times*. Recuperado marzo 24, 2019, a partir de https://tht.fangraphs.com/katoh-forecasting-a-hitters-major-league-performance-with-minor-league-stats/

MLB. (s. f.). Rules, Regulations and Statistics. *Major League Baseball*. Recuperado marzo 23, 2019a, a partir de http://mlb.mlb.com/mlb/official_info/about_mlb/rules_regulations.jsp

MLB. (s. f.). What is a Player Empirical Comparison and Optimization Test Algorithm (PECOTA)? | Glossary. *Major League Baseball*. Recuperado marzo 24, 2019b, a partir de http://m.mlb.com/glossary/projection-systems/player-empirical-comparison-and-optimization-test-algorithm

MLB. (s. f.). What is a Steamer? | Glossary. *Major League Baseball*. Recuperado marzo 24, 2019c, a partir de http://m.mlb.com/glossary/projection-systems/steamer

MLB. (s. f.). What is a sZymborski Projection System (ZiPS)? | Glossary. *Major League Baseball*. Recuperado marzo 23, 2019d, a partir de http://m.mlb.com/glossary/projection-systems/szymborski-projection-system

MLB. (s. f.). First-Year Player Draft Rules. *Major League Baseball*. Recuperado marzo 24, 2019e, a partir de http://mlb.mlb.com/mlb/draftday/rules.jsp

MLB. (s. f.). First-Year Player Draft FAQ. *Major League Baseball*. Recuperado marzo 24, 2019f, a partir de http://mlb.mlb.com/mlb/draftday/faq.jsp

MLB. (s. f.). Glossary. *Major League Baseball*. Recuperado junio 9, 2019g, a partir de http://m.mlb.com/glossary

Moreira, A., Santos, M., & Carneiro, S. (2005, julio 25). Density-based clustering algorithms – DBSCAN and SNN.

Powell, V. (2009). Eigenvectors and Eigenvalues explained visually. *Explained Visually*. Recuperado junio 2, 2019, a partir de http://setosa.io/ev/eigenvectors-and-eigenvalues/

Sabino, D. (2005). *Dominate Your Fantasy Baseball League*. Boston, MA.

scikit-learn. (s. f.). sklearn.manifold.TSNE — scikit-learn 0.21.2 documentation. Recuperado junio 3, 2019, a partir de https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html

Scikit-learn. (s. f.). Scikit-learn.kmeans. Recuperado junio 4, 2019, a partir de https://scikit-learn.org/stable/modules/clustering.html#k-means

Slowinski, S. (2010, febrero 25). Projection Systems | Sabermetrics Library. Recuperado marzo 23, 2019, a partir de https://library.fangraphs.com/principles/projections/

Slowinski, S. (2011, febrero 16). Resumen FG todas. *Sabermetrics Library*. Recuperado marzo 24, 2019, a partir de https://library.fangraphs.com/the-projection-rundown-the-basics-on-marcels-zips-cairo-oliver-and-the-rest/

Tango, T. (2004). Tango on Baseball. Recuperado marzo 23, 2019, a partir de http://www.tangotiger.net/archives/stud0346.shtml

Tango, T. (2012). Marcel 2012. Recuperado marzo 23, 2019, a partir de http://www.tangotiger.net/marcel/

Xu, C. (2018, junio 14). Why is Dimensionality Reduction so Important? *Chuan Xu*. Recuperado mayo 11, 2019, a partir de https://medium.com/@cxu24/why-dimensionality-reduction-is-important-dd60b5611543