UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE
INGENIERÍA DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

ANÁLISIS PREDICTIVO CON TÉCNICAS BIG DATA

GRADO EN INGENIERÍA
TELEMÁTICA

CRISTÓBAL NARANJO GARCÍA
MÁLAGA, 2016

E.T.S. DE INGENIERÍA DE TELECOMUNICACIÓN, UNIVERSIDAD DE MÁLAGA

Análisis predictivo con técnicas big data

Autor: Cristóbal Naranjo García

Tutor: Antonio J. Nebro y José Manuel García Nieto

Departamento: Lenguajes y Ciencias de la Computación

Titulación: Grado en Ingeniería Telemática

Palabras clave: Big data, análisis predictivo, machine learning, regresión lineal, Apache Spark, MongoDB

Resumen

El big data pretende analizar un enorme volumen de datos a la mayor velocidad posible teniendo en cuenta además que dichos datos pueden ser muy variados. Este documento recoge el trabajo realizado durante el desarrollo de una aplicación para llevar a cabo un análisis predictivo mediante técnicas big data mostrando los resultados a través de una aplicación Web. Se describen las herramientas que se han utilizado en el desarrollo de la aplicación, se detalla la implementación de la misma y finalmente se realiza un análisis predictivo mediante regresiones lineales. La finalidad de este trabajo es tomar un primer contacto con las técnicas y herramientas big data, las tecnologías Web de última generación y hacer uso de la técnicas de aprendizaje máquina para realizar las predicciones.

E.T.S. DE INGENIERÍA DE TELECOMUNICACIÓN, UNIVERSIDAD DE MÁLAGA

Predictive analysis with big data techniques

Author: Cristóbal Naranjo García

Supervisor: Antonio J. Nebro y José Manuel García Nieto

Department: Lenguajes y Ciencias de la Computación

Degree: Grado en Ingeniería Telemática

Keywords: Big data, predictive analysis, machine learning, linear regression, Apache Spark, MongoDB

Abstract

Big data aims at managing an enormous volume of data at the highest possible speed taking into account the data variety. This document gathers the work carried out during the development of a software application for predictive analysis using big data techniques, showing the results through a Web application. It also describes the tools used in the development phase, implementation details, and concludes with a predictive analysis based linear regressions. The purpose of this work is to take a first contact with big data techniques and tools, Web technologies and the use of machine learning algorithms to make predictions.

Agradecimientos

A mis padres, Cristóbal y Cristina por permitirme llegar hasta aquí.

A mi hermano, Alex, por sus sabios consejos.

A Alejandro, por saber motivarme y animarme hasta el último momento.

A Carlos, Mateo y Miguel, por aguantarme durante todos los años que ha durado esta etapa.

A mis amigos de San Pedro, lo conseguí.

A Jesús, por ser más que un profesor durante la carrera y ayudarme a encontrar a los tutores de mi proyecto.

A mis tutores, Antonio y José Manuel, por confiar en este *teleco* para realizar este proyecto.

Gracias a todos porque de una forma u otra me habéis permitido escribir estas líneas.

A mi familia y mis amigos,

por haberme apoyado siempre,

por creer en mí.

Contenido

Lista de Acrónimos

NBA	National Basketball Association
MEAN	MongoDB, Express, AngularJS y Node.js
API	Application Programming Interface
RDD	Resilient Distributed Datasets
SQL	Structured Query Language
JSON	JavaScript Object Notation
ECM	Error Cuadrático Medio
R^2	Coeficiente de determinación
IDE	Integrated Development Environment

Capítulo 1. Introducción

"Predecir es difícil, especialmente el futuro". (Niels Bohr).

A lo largo de esta memoria se va a desarrollar una aplicación para predecir resultados de partidos de la NBA.

El ser humano es impredecible, no hay un patrón que indique el comportamiento de todos ellos por igual. De hecho existe una ciencia que estudia su conducta, la psicología, pero aún no se ha descifrado todos los aspectos de la experiencia humana. Por eso para este proyecto se intentará encontrar una relación entre variables basándose únicamente en las matemáticas, dejando a un lado el comportamiento humano, e intentando encontrar una relación entre ellas.

El baloncesto tiene muchas variables que pueden influir en el resultado de un partido, asistencias, rebotes, faltas, etc. pero sin duda lo que marca el resultado final son los puntos. Los puntos serán la variable que se relacionará con los jugadores buscando una dependencia que permita obtener una predicción, un resultado.

Para obtener estos resultados, el análisis se enfocará en predecir la anotación de cada jugador independientemente del equipo en el que juegue o de los compañeros que salten a la pista el día del partido. Una vez obtenidos los resultados individuales se hará un computo y se procederá a mostrar el resultado final predicho, tanto del equipo como de los jugadores.

La fase de análisis se centrará en varios jugadores, no todos, para ajustar los modelos individualmente y así obtener una mejor predicción. Los modelos se ajustarán intentando obtener el menor error posible, pero como ya se ha indicado el ser humano es impredecible, y no siempre se podrá obtener un resultado aceptable.

El análisis se hará mediante regresiones lineales que permitirán obtener el resultado a partir de datos ya conocidos. Se ajustarán modelos y se utilizaran diferentes para al final obtener un resultado aceptable.

Este proyecto tiene fines educativos, no comercial, por lo que sólo se harán predicciones de partidos ya finalizados, en concreto se utilizará una base de datos de resultados de partidos de la NBA desde la temporada 1984-1985 hasta la temporada 2012-2013.

1.1. Motivación y objetivos

La NBA es la liga de baloncesto más importante del mundo y el análisis de datos (big data) una de las mayores tendencias en el mundo de la informática sobre todo para la transformación del mundo empresarial.

La estadística en los deportes es algo habitual, sobre todo para mostrar como está transcurriendo un partido. Pero utilizar la estadística para marcar la diferencia en un partido es algo que no en todos los deportes se hace. Sin duda una de las historias que motiva para realizar este trabajo es *"Moneyball: The Art of Winning an Unfair Game"* (2003, Michael Lewis) en la que el director general de un equipo de baseball intenta construir un equipo, basándose en la estadística, para ganar un campeonato dejando a un lado a los mejores jugadores y utilizando solo jugadores que estadísticamente le ayudarían a ganar el partido en el computo global, es decir, jugada a jugada.

El objetivo de este proyecto es utilizar la estadística para realizar un análisis de la anotación de los jugadores en un partido de baloncesto. Para realizar el análisis se utilizará una base de datos no estructurada (MongoDB) en la que se contiene información de todos los partidos de temporada regular desde la temporada 84-85 hasta la temporada 12-13. De esta información se extraerán los partidos para obtener el equipo que jugaba como local, y el visitante, los jugadores que estaban convocados para ese partido, y la anotación que realizaron.

Una vez obtenidos estos datos se modificarán y tratarán para obtener diferentes modelos con los cuales realizar las predicciones. Para ello se utilizará la herramienta Apache Spark que permite crear modelos de datos y manejar grandes cantidades de datos con rapidez.

Para predecir se utilizará un análisis de regresión lineal, que también nos lo proporciona la herramienta Apache Spark.

Una vez obtenida la predicción se mostrarán los resultados a través de una aplicación Web. Esta aplicación solo mostrará los resultados finales, mientras que durante la memoria se mostrarán otros resultados así como algunas figuras ilustrativas.

La finalidad de este proyecto es aprender a utilizar las herramientas *big data* que son necesarias para realizar el análisis, las herramientas para crear una aplicación Web, así como nuevos lenguajes de programación.

1.2. Organización de la memoria

La memoria se organiza de la siguiente forma:

El capítulo 2, herramientas y conceptos del desarrollo, pretende presentar las herramientas que se emplearán, así como dar a conocer las bases y el conocimiento que han permitido el desarrollo de la memoria.

El capítulo 3, diseño e implementación, muestra al lector el curso del diseño e implementación de la aplicación. Se aconseja al lector que que profundice en las referencias del capítulo 2 y en la bibliografía si no se tienen nociones básicas de las herramientas que se van a utilizar o del desarrollo de una aplicación Web ya que se obvian algunos conceptos.

El capítulo 4, análisis de resultados, es el capítulo donde se recogen los análisis realizados para los diferentes modelos utilizados. Permitirá obtener una visión general de los modelos y su efectividad.

El capítulo 5, conclusiones y trabajo futuro, contiene lo relativo a la discusión de mejoras o nuevas posibilidades que permitan mejorar la aplicación desarrollada desde la perspectiva del autor.

Capítulo 2. Herramientas y conceptos del desarrollo

2.1. Big data

Uno de los conceptos que ya ha aparecido en esta memoria y que volverá ha aparecer es el big data. El término big data hace referencia a una gran colección de datos y para tratar estos datos se han creado patrones de arquitectura específicos ya que con los patrones tradicionales se tardaría mucho tiempo en analizar los datos. En el caso de esta memoria se trataran más de 80.000 partidos con estadísticas de todos los jugadores que participaron, desde la anotación hasta las faltas personales que cometieron.

Uno de los primeros patrones que aparecieron para el procesamiento de grandes colecciones de datos fue el MapReduce, propiedad de Google. Este modelo se basa en dos funciones: Map y Reduce. La función Map permite filtrar y ordenar los datos mientras que la función Reduce permite realizar un resumen de datos que siguen el mismo patrón.

En este proyecto se utilizará una herramienta de código libre que permite realizar muchas más funciones e incorporar diferentes tecnologías. Se utilizará Apache Spark.

2.2. Apache Spark

Spark es una herramienta de código abierto para la computación distribuida a través de clústeres de ordenadores. Extiende del modelo MapReduce

soportando más tipos de computación, incluyendo consultas interactivas y procesamiento de flujos de datos en tiempo real. Aprovecha el procesamiento de datos en paralelo, manteniendo los datos en memoria y en el disco si es necesario, para ofrecer una mayor velocidad. Es una herramienta muy accesible ya que ofrece APIs para diferentes lenguajes, como Java o Scala, además de estar integrada con otras herramientas big data, como Hadoop, y diferentes tipos de bases de datos, como Cassandra o MongoDB. [1]

2.3. MongoDB

MongoDB es un sistema de base de datos no estructurado orientado a documentos, desarrollado bajo el concepto de código abierto. Los documentos tienen una estructura similar a JSON ya que es una representación binaria de este tipo, es decir, una colección de pares de la forma <nombre>:<valor> llamada BSON (Binary JSON). Una de las principales características de MongoDB es la capacidad de ejecutarse en múltiples servidores balanceando la carga y/o replicando los datos para mantener el sistema funcionando en caso de fallo, ya sea de hardware o por la saturación de un servidor. [2]

2.4. Aplicación Web MEAN

El acrónimo MEAN hace referencia a un conjunto de herramientas necesarias para crear una aplicación Web completa. La M hace referencia a la base de datos, MongoDB, de la cual ya se ha hablado anteriormente. La E es de Express [3], una infraestructura de aplicaciones Web Node.js mínima y flexible que proporciona un conjunto sólido de características para las aplicaciones Web y móviles. Con esta infraestructura se creará una interfaz (API) para intercambiar la información entre la base de datos, la aplicación Web y Spark. La A, AngularJS [4], es un framework de JavaScript, que junto a HTML y CSS permite crear una aplicación Web basada en el patrón Modelo Vista Controlador (MVC), es decir, es la herramienta que permite generar la interfaz de usuario de este proyecto. Por ultimo, N, Node.js[5] es un entorno en tiempo de ejecución para la capa de servidor. Será el motor de la aplicación Web que permitirá ejecutar el código JavaScript.

2.5. Análisis de regresión simple

El análisis de regresión es una técnica estadística para estudiar la relación entre variables. El análisis de regresión simple es el caso en el que solo existen dos variables. El análisis puede utilizarse para explorar y cuantificar la relación entre una variable dependiente, (Y) y una variable independiente (X), así como desarrollar una ecuación lineal con fines predictivos.

En problemas de regresión se dispone de una serie de datos de entrenamiento que representan las entradas y las salidas correspondientes de un sistema lineal. El objetivo de la regresión es descubrir la relación funcional entre la entrada y la salida de este sistema, para poder así predecir la salida del sistema cuando se le presenta de entrada un nuevo dato.

Si se representaran los datos de entrenamiento se obtendría una nube de puntos, con la regresión se pretende encontrar un modelo aproximado de la dependencia de las variables, es decir, encontrar la función que más se ajusta a esa nube de puntos, en este caso una línea, la línea de regresión, véase Figura 2.1 y Figura 2.2

El método más tradicional para realizar una regresión lineal es el ajuste por el método de mínimos cuadrados.

Siendo los datos $\{(x_i, y_i)\}$ para las variables X e Y y con el objetivo de determinar la función y = f(x), se trata de minimizar la función objetivo mínimo-cuadrática:

$$F = \sum_i (y_i - y_i^{est})^2 = \sum_i (y_i - f(x_i))^2 \quad (1)$$

Con $y_i^{est} = f(x_i)$, es decir, el valor de la función estimado por la regresión para cada x_i,y $e_i = y_i - y_i^{est}$ siendo el error cometido por el ajuste para el i-ésimo dato, minimizar la función objetivo significa minimizar el error cuadrático medio (ECM) y la media cuadrática de los errores (MC).

$$ECM = \frac{\sum_i e_i^2}{N} \quad (2)$$

$$MC = \sqrt{\frac{\sum_i e_i^2}{N}} \quad (3)$$

Al ser una regresión lineal se usará un ajuste lineal por lo que la función f(x) será del tipo Y = a + bX.

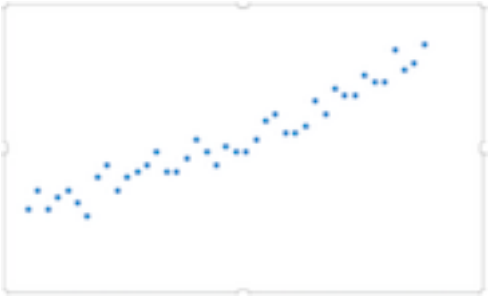

Figura 2.1 Diagrama de dispersión

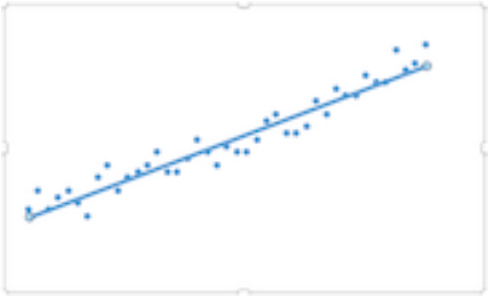

Figura 2.2 Diagrama de dispersión con ajuste lineal

Una de las variables que se usarán para comprobar el grado de ajuste de la regresión es el coeficiente de determinación (R^2). Este coeficiente determina la calidad del modelo para replicar los resultados, y la proporción de variación de los resultados que puede explicarse por el modelo. En el caso de la regresión lineal este coeficiente coincide con el cuadrado del coeficiente de correlación de Pearson. El coeficiente de determinación puede adquirir valores entre cero y uno, siendo uno un ajuste perfecto y cero un ajuste que no explica nada. [6]

La variable dependiente del análisis de este proyecto serán los puntos anotados en un partido, mientras que la variable independiente será el partido. De esta manera se calcularán como mínimo diez regresiones lineales para cada partido, una por jugador.

2.6. Lenguajes y entornos de programación

En la implementación de este proyecto se utilizarán dos lenguajes principalmente: Scala y JavaScript.

El lenguaje Scala es un lenguaje de programación multi-paradigma diseñado para expresar patrones comunes de programación en forma concisa, elegante y con tipos seguros. Será utilizado para desarrollar en Spark y se ha elegido para

iniciarse con los lenguajes de programación funcional, además de que Spark está escrito en su mayoría en Scala. Para el diseño del código en este lenguaje se ha utilizado el entorno de programación (IDE) IntelliJ IDEA debido a su facilidad para programar en Scala y a facilidad para utilizar Maven, la herramienta para la gestión y construcción de proyectos.

JavaScript es un lenguaje de programación interpretado, utilizado principalmente en la navegación Web. En JavaScript se desarrollará la aplicación Web completa. En esta parte no se utilizará un IDE, solo se utilizará un editor de textos, en este caso Atom.

2.7. Resumen de tecnologías usadas

Tabla 2.1 Resumen de tecnologías

Herramientas big data	
Apache Spark	Análisis de datos
Herramientas web	
AngularJS	Desarrollo de interfaz de usuario
Node.js	Desarrollo de servidor Web
Express	Desarrollo de la API
Base de datos	
MongoDB	Base de datos noSQL
Lenguajes de programación	
Scala	Análisis predictivo
JavaScript	API y Web
HTML	Interfaz de usuario Web

Capítulo 3. Diseño e implementación

3.1. Diseño de la aplicación

3.1.1. Casos de uso

En la Figura 3.1 se detallan los casos de uso de la aplicación.

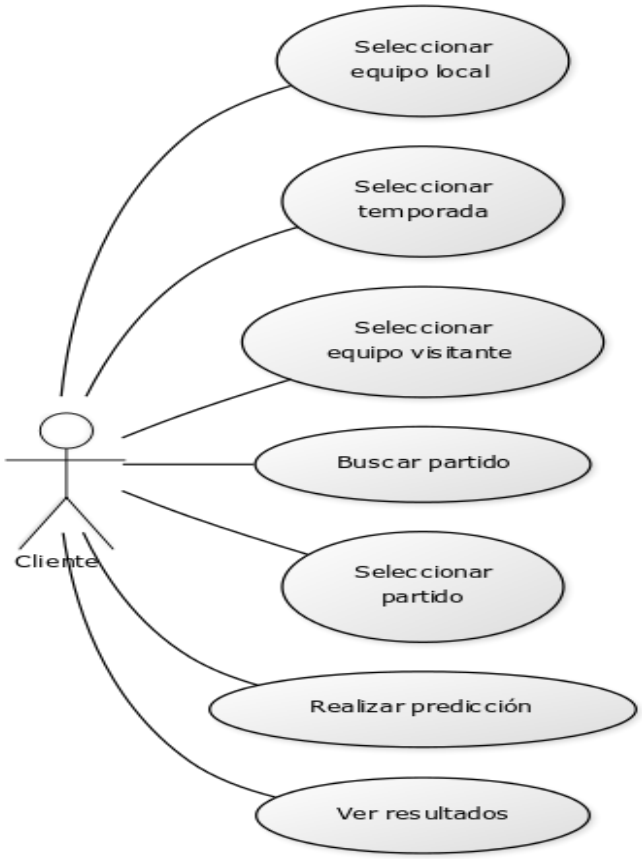

Figura 3.1 Casos de uso de la aplicación

3.1.2. Funcionamiento de la aplicación

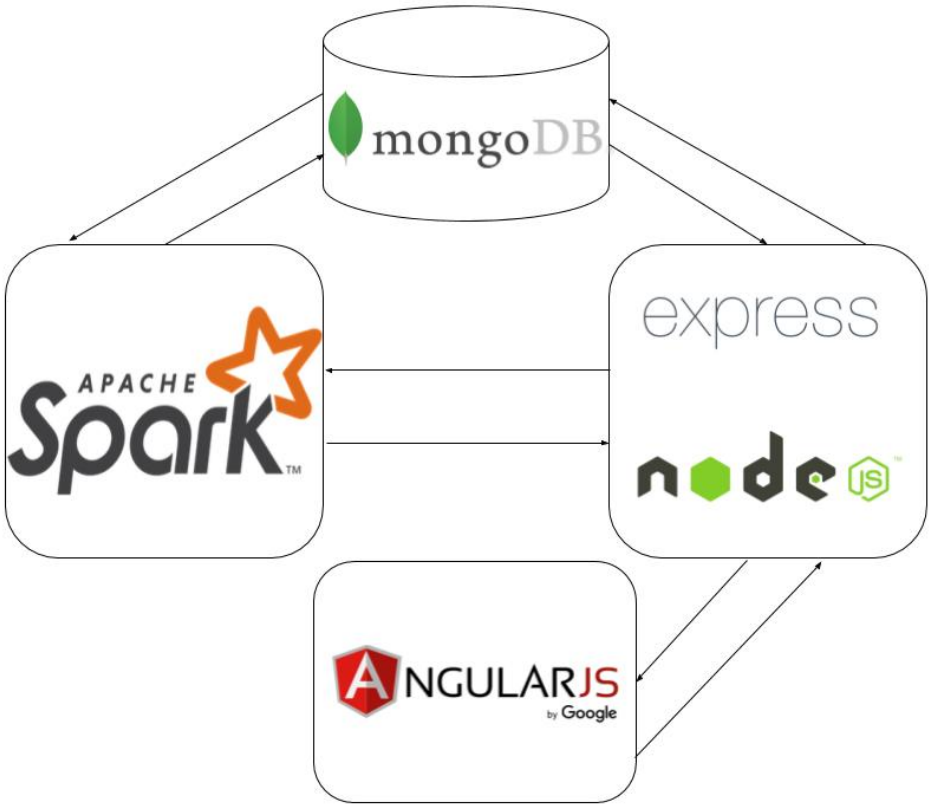

Figura 3.2 Diagrama de la aplicación

La aplicación se divide en cuatro módulos que formaran dos aplicaciones independientes y juntas formaran la aplicación completa. Los módulos son:

- Base de datos, MongoDB.
- Motor de análisis big data, Apache Spark.
- Servidor y API, Express y Node.js.
- Interfaz de usuario, AngularJS

Por un lado se realizará una aplicación de análisis predictivo formada por el motor de análisis big data y la base de datos. La otra aplicación será una aplicación Web formada por el servidor y la API, la base de datos y la interfaz de usuario. La aplicación al completo tiene un proceso de diez pasos que se definen a continuación:

1. Selección de datos en la interfaz de usuario, equipos local y visitante, temporada del partido, y petición al *backend* de búsqueda.

2. Consulta a la base de datos, MongoDB, mediante Express con los datos seleccionados en el paso 1.
3. Respuesta de la MongoDB a Express mediante un objeto con la información obtenida.
4. Respuesta de Express a la petición y carga de partidos en AngularJS.
5. Selección de datos completa en la interfaz de usuario, equipos, temporada y partido, y petición de predicción gestionada por la API con Express.
6. Creación de fichero con los datos y petición de análisis a Spark, se lanzará con Node.js y Express controlará la petición.
7. Consulta de Spark a MongoDB por los jugadores y equipos seleccionados.
8. Respuesta de MongoDB a Spark con la información.
9. Respuesta con el análisis predictivo de Spark a la petición de Express.
10. Respuesta a AngularJS con la predicción y carga de datos en la interfaz de usuario.

3.2. Desarrollo de la aplicación de análisis predictivo

3.2.1. Configuración del entorno de desarrollo

El primer paso para la implementación es crear un proyecto con el IDE utilizando una estructura Maven predefinida para Scala, con el fin de gestionar los paquetes necesarios. Gracias a Maven se evitará tener que descargar e instalar todas las dependencias de librerías del proyecto, y permitirá gestionar y construir el proyecto sencillamente.

Una vez creado el proyecto, se empezará añadiendo las principales dependencias al fichero Maven modelo (POM), la representación en lenguaje de etiquetas XML del proyecto, que son:

- El lenguaje Scala en su versión 2.10.6
- Las librerías de Apache Spark, tanto el núcleo, la librería SQL, como la librería de *Machine Learning* para la versión de Scala escogida anteriormente, en sus versiones 1.6.1.

- El conector para Spark y la base de datos MongoDB, versión 0.4

Las dependencias en código XML del fichero POM quedarían así:

```
<dependencies>
    <dependency>
        <groupId>org.scala-lang</groupId>
        <artifactId>scala-library</artifactId>
        <version>${scala.version}</version>
    </dependency>
    <dependency>
        <groupId>org.apache.spark</groupId>
        <artifactId>spark-core_2.10</artifactId>
        <version>1.6.1</version>
    </dependency>
    <dependency>
        <groupId>org.apache.spark</groupId>
        <artifactId>spark-mllib_2.10</artifactId>
        <version>1.6.1</version>
    </dependency>
    <dependency>
        <groupId>org.apache.spark</groupId>
        <artifactId>spark-sql_2.10</artifactId>
        <version>1.6.1</version>
    </dependency>
    <dependency>
        <groupId>org.apache.spark</groupId>
        <artifactId>spark-hive_2.10</artifactId>
        <version>1.6.1</version>
    </dependency>
    <dependency>
        <groupId>org.mongodb.spark</groupId>
        <artifactId>mongo-spark-
connector_2.10</artifactId>
        <version>0.4</version>
    </dependency>
</dependencies>
```

Una vez introducidas las dependencias ya se puede empezar a generar el bloque de código de la aplicación en Scala. En este bloque solo se hará uso de una clase, en la que se implementarán las diferentes funciones.

Para comenzar es necesario inicializar Spark pasándole los parámetros de configuración, este paso parece obvio pero es de vital importancia, ya que uno de los parámetros que se le pasará será la dirección donde se encuentra la base de datos, que gracias al conector “mongo-spark” es una tarea sencilla. Este

parámetro es de tipo texto y es una dirección que apunta a la base de datos, llamado "*spark.mongodb.input.uri*". Al ser un proyecto local siempre se usarán las direcciones "*localhost*". En la dirección de la base de datos también se incluye el nombre de la misma y la colección. Es necesario hacer un paréntesis y hablar sobre la base de datos.

3.2.2. Configuración de la base de datos

La base de datos se presenta en un fichero *dump.* Este fichero es una copia de seguridad de la base de datos realizada por Valeri Karpov durante una "Hackatom" de MongoDB con datos obtenidos de "basketball-reference.com". [7]

Es necesario restaurar esta copia en el entorno local para hacer uso de ella. Para esta tarea hay que descargar primero el servidor de MongoDB que permite utilizar estas bases de datos, instalarlo y posteriormente restaurar la copia con el comando *mongorestore.*

MongoDB provee una serie de utilidades para ser utilizadas a través de la terminal. En este proyecto se utilizará *mongorestore* para restaurar la copia de la base de datos, *mongod,* para lanzar el servidor de MongoDB y tener disponibles las bases de datos, y *mongo,* el cliente interactivo que provee la interfaz para realizar consultas a las bases de datos. Además del cliente *mongo,* se utilizará el cliente, también gratuito, MongoHub que nos permite visualizar las bases de datos de manera más simple.

Una vez restaurada la base de datos y lanzado el servidor, el cliente puede acceder a la base de datos llamada NBA, que se compone de la colección "games". Una colección son conjuntos de documentos. Si se realiza una búsqueda sin indicar ningún parámetro el cliente debe mostrar los 31.686 documentos que componen la colección, véase Figura 3.3. Cada documento hace referencia a un partido, y en él podemos encontrar el identificador único en la base de datos, la fecha del partido, los equipos y los jugadores que fueron convocados, y las estadísticas que realizaron. Dentro de las estadísticas tenemos más de 20 parámetros, asistencias, rebotes, robos, lanzamientos y porcentajes de acierto, minutos jugados, faltas y el más importante de todos, el que se utilizará en este proyecto, los puntos anotados.

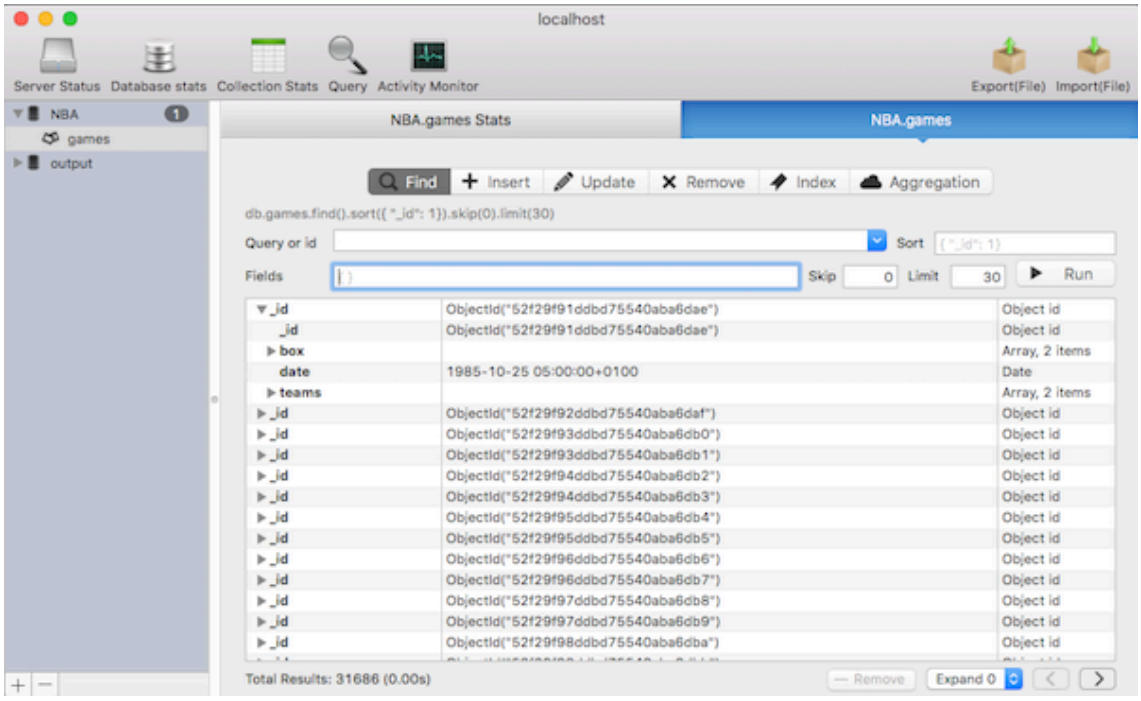

Figura 3.3 Resultado de una búsqueda sin parámetros en MongoHub

3.2.3. RDD, manipulación y filtrado de datos

Con la base de datos ya instalada y configurada, los parámetros de configuración en Spark quedarían:

```
val mongoConfig = new SparkConf()
mongoConfig.setAppName("MongoSpark")
mongoConfig.setMaster("local[4]")
mongoConfig.set("spark.mongodb.input.uri",
"mongodb://localhost:27017/NBA.games")
```

La aplicación necesita un nombre que se utiliza también como identificador del proceso en un clúster. En este proyecto solo se utilizará una máquina por lo que no hay problemas para encontrar el proceso, pero se recomienda utilizar este parámetro aunque no sea obligatorio. También se configura la máquina como local y el número de núcleos, nodos, que puede soportar. En este caso se configura con cuatro nodos, es decir, la máquina ejecutará cuatro hebras para realizar los cálculos en cuatro procesos paralelos.

Con Spark y MongoDB corriendo solo falta introducir la MongoDB en un tipo de dato que Spark pueda manejar, los RDDs. Un RDD es una colección distribuida de elementos, la abstracción que hace el núcleo de Spark para trabajar con datos. En Spark todo el trabajo es expresado creando nuevos RDDs,

transformando RDDs existentes o realizando operaciones en los RDDs para calcular un resultado. Cada RDD se divide en tantos nodos como tenga el clúster y se computa en paralelo.

Spark lee la MongoDB completa e introduce la información en un RDD que permite actuar sobre la información y transformarla. El RDD es un objeto inmutable y que será utilizado constantemente para buscar información y crear nuevos RDDs por lo que es recomendable guardar la información en memoria y que Spark pueda trabajar más rápido. Para guardar en memoria se utiliza la función *cache*.

Con la información en un objeto RDD de Spark ya solo queda empezar a filtrar y manipular los datos. Se utilizará un filtro dividido en dos bloques. En el primer bloque se filtrarán los datos mediante el proceso de agregación de MongoDB que consiste en manipular los datos para obtener una estructura más simple con la que Spark pueda tratar rápidamente.

La tubería de agregación, *aggregation pipeline*, en MongoDB consiste en un proceso de diferentes etapas. La función de agregación recibe un array de objetos en el que cada objeto es una operación que se realiza con los datos y el resultado de dicha operación es la entrada de la siguiente.

A continuación se muestra un ejemplo de agregación realizado mediante el cliente de MongoDB:

```
db.games.aggregate([
  {
    $match : {
       "box.players.player": "Kobe Bryant"
    }
  },{
    $unwind : '$box'
  },
   {
    $match : {
       "box.players.player": "Kobe Bryant"
    }
  },
  {
    $unwind : '$box.players'
  },
   {
    $match : {
       "box.players.player": "Kobe Bryant"
    }
  }
  ]);
```

Este ejemplo es una agregación de cinco etapas. En la primera etapa se buscan todos los partidos en los que Kobe Bryant ha participado. En la segunda etapa se deconstruye el array con los datos de los equipos y se obtienen dos elementos por partido, uno para el equipo local y otro para el equipo visitante. En la tercera se vuelve a buscar los partidos de Kobe Bryant, por lo que se eliminan los datos de los equipos contra los que ha jugado. En la cuarta se deconstruye el array para crear nuevos elementos para cada jugador. La ultima etapa filtra los datos y obtiene solo los de Kobe Bryant, todos los partidos que hay disponibles en la base de datos.

En el segundo bloque se utilizarán las herramientas que provee Spark para filtrar y obtener los datos necesarios para los diferentes modelos. El esquema original de la base de datos se puede observar en la Figura 3.4.

_id	ObjectId("52f2b467ddbd75540aba94c2")	Object id
▼ box		Array, 2 items
▼ 0		Object, 3 items
▶ players		Array, 12 items
▶ team		Object, 19 items
won	1	Integer
▼ 1		Object, 3 items
▶ players		Array, 10 items
▶ team		Object, 19 items
won	0	Integer
date	1995-02-08 06:00:00+0100	Date
▼ teams		Array, 2 items
▼ 0		Object, 5 items
name	Atlanta Hawks	String
abbreviation	ATL	String
score	111	Integer
home	true	Boolean
won	1	Integer
▼ 1		Object, 5 items
name	New Jersey Nets	String
abbreviation	NJN	String
score	88	Integer
home	false	Boolean

Figura 3.4 Esquema por defecto de la MongoDB

En este esquema se encuentran todos los parámetros disponibles. Para llegar a obtener solo los puntos de cada jugador, en el primer bloque se filtrara la MongoDB por etapas como en el ejemplo anterior. Se obtendrán todos los partidos de un jugador concreto sin importar el equipo u otro parámetro. Una vez obtenidos todos los partidos de un jugador comienza el filtrado del segundo bloque. En el primer paso del segundo bloque se crea una estructura nueva en la que se manipulan los datos para solo obtener los puntos anotados por el jugador. La nueva estructura se muestra a continuación.

```
root
 |-- box: struct (nullable = true)
 |    |-- players: struct (nullable = true)
 |    |    |-- pts: integer (nullable = false)
 |    |-- won: integer (nullable = false)
 |-- date: timestamp (nullable = true)
 |-- teams: array (nullable = true)
 |    |-- element: struct (containsNull = true)
 |    |    |-- name: string (nullable = true)
 |    |    |-- abbreviation: string (nullable = true)
 |    |    |-- score: integer (nullable = false)
 |    |    |-- home: boolean (nullable = false)
 |    |    |-- won: integer (nullable = false)
```

Una vez con esta estructura se utilizarán las herramientas de Spark para el filtrado. En el cambio de tipo de estructura el RDD se transforma en otro tipo de datos de Spark, un *Dataframe.* Este tipo de dato permite utilizar la nueva estructura como si fuera una tabla SQL, por lo que las funciones de filtrado serán las mismas que en una base de datos SQL. Spark permite hacer uso de la sintaxis de SQL para realizar el filtrado o utilizar funciones que proporciona la API del lenguaje. En este caso se optará por la segunda opción para familiarizarse con la API Scala de Spark.

En el apartado de análisis se utilizaran diferentes modelos que requerirán diferentes filtrados. Estos filtrados son muy parecidos entre sí por lo que en este apartado solo se explicará un ejemplo, filtrado para obtener resultados jugando el mismo mes del partido en años anteriores.

La fecha del partido que se quiere predecir es un parámetro conocido. Se crean tres filtros, para los días, el mes y los años. Los filtros son variables de tipo texto que se le pasarán mediante la función *filter* al *Dataframe* que contiene todos los partidos de un jugador con la estructura modificada del nivel uno de filtrado. Una vez pasado los filtros se seleccionan solo los puntos y se crea un *Dataframe* que contiene nombre del jugador y puntos en partidos anteriores. Con esto terminaría la fase de manipulación y filtrado de datos.

```
val dia = fecha(2)
val mes = fecha(1)
val año = fecha(0)
val filtroDia = "day(date)<="+dia
val filtroMes = "month(date)="+mes
val filtroAño = "year(date)<"+año
val resultHistorico =
dfJugador.filter(filtroAño).filter(filtroMes).filter(filtro
Dia)
  .select("box.players.pts")
  .map(puntos => (player,puntos.getInt(0)))
```

3.2.4. Regresión lineal. Modelo y predicción

Spark necesita un tipo de dato específico para realizar una regresión lineal, el *LabeledPoint.* Este tipo esta compuesto por un número y un vector. El número hace referencia a la variable independiente, mientras que el vector es la variable dependiente. Al ser un vector, Spark permite utilizar más de una variable pero en este caso solo haremos uso de una. En código un LabeledPoint se definiría:

```
val nuevoPunto =
LabeledPoint(Double(X), Vectors.dense(Double(Y)))
```

A los datos obtenidos en la fase de filtrado se les aplica una transformación para convertirlos a *LabeledPoint* y así poder crear un modelo a partir de la regresión lineal. Para crear este modelo se utilizará la función *fit* de las regresiones lineales de Spark y posteriormente se harán las predicciones utilizando la fórmula de la función Y = a + bX encontrada. Las variables a y b son diferentes para cada modelo. En Spark y en las regresiones lineales al término a se le llama "*intercept*" y al término b *"coefficient".*

Una vez obtenidos los parámetros a y b se ajusta la función para realizar la predicción y se obtiene el parámetro Y estimado. Con la predicción y el valor real ya se puede empezar a realizar el análisis calculando el error cuadrático medio.

```
val linReg = new
LinearRegression().setMaxIter(100).setFitIntercept(true)
val model = linReg.fit(parsedData)
val a = model.intercept
val b = model.coefficients(0)
```

En este punto se obtiene un modelo y una predicción. Al modelo definido hasta ahora se le conoce como modelo sobreentrenado. Para entender que es el sobreentrenamiento hay que explicar que es el Machine Learning.

El Machine Learning es el diseño y estudio de las herramientas que utilizan la experiencia pasada para tomar decisiones futuras. Tiene como objetivo generalizar. Aprender de los datos para encontrar patrones.

El modelo utilizado anteriormente se entrena con todos los datos, y realiza una predicción con todos los datos. Puede ser que utilizando todos los datos se encuentre un patrón que permita obtener un resultado aceptable en el análisis de regresión pero no se estaría generalizando, objetivo principal del Machine Learning, por eso se dice que se está sobreentrenando.

Para evitar el sobreentrenamiento en este proyecto se utilizará la retención de datos. La retención de datos consisten en fraccionar el conjunto de datos y utilizar una parte de ellos para el entrenamiento y otra para la evaluación. Con este método se busca analizar con más precisión como se comporta el modelo a medida que se va entrenando con diferentes bloques de datos obtenidos del mismo conjunto.

Se utilizará una proporción de 70-30 para los modelos de este proyecto, con lo que el ejemplo anterior quedaría de la siguiente forma:

```
val splits = data.randomSplit(0.70,0.30)
val trainingData = splits[0]
val testData = splits[1]
val linReg = new
LinearRegression().setMaxIter(100).setFitIntercept(true)
val model = linReg.fit(trainingData)
val a = model.intercept
val b = model.coefficients(0)
```

De esta manera se podrán evaluar mejor los diferentes modelos que se obtengan de la fase de filtrado y se evitará el sobreentrenamiento.

Los resultados finales se almacenan en un fichero de texto en el que se escribirán los equipos, los jugadores y la predicción.

3.3. Desarrollo de la aplicación Web

En el desarrollo de la aplicación Web distinguiremos dos partes, el *backend* y el *frontend*.

- El *backend* es la parte de la aplicación que actúa como servidor y que interacciona con la base de datos.
- El *frontend* es la parte que actúa como cliente y que proporciona la interfaz de usuario.

3.3.1. Backend

Al igual que en la aplicación de análisis predictivo se utilizó un gestor de dependencias (Maven), se utilizará para la aplicación Web Node.js como gestor. En un fichero llamado *package.json* se colocarán todas las dependencias necesarias para la aplicación y que serán descargadas una vez se ejecute Node.js en el terminal.

```
{
  "name"         : "TFG",
  "version"      : "0.0.1",
  "description"  : "App MEAN",
  "main"         : "app.js",
  "author"       : "Cristu Naranjo",
  "dependencies" : {
    "express"    : "4.14.0",
    "mongoose"   : "4.5.9",
    "serve-static": "1.11.1",
    "morgan"     : "1.7.0",
    "body-parser": "1.15.2",
    "method-override": "2.3.6",
    "angular"    : "1.5.8",
    "angular-aria": "1.5.8",
    "angular-material": "1.1.0",
    "angular-ui-router": "0.3.1"
  }
}
```

Express nos permitirá crear la interfaz para la comunicación de la aplicación Web. Mongoose es el conector que permite a Node.js utilizar MongoDB como base de datos. Serve-static es la dependencia para indicarle a Node.js dónde se encuentran los ficheros de la aplicación. Morgan permite obtener un log sobre los mensajes que muestre el servidor. Body-parser y method-override son dependencias para tratar con las peticiones HTTP. Y por ultimo Angular.js que será quién permita desarrollar el cliente, la interfaz de usuario.

Mongoose no es capaz de leer por sí solo la estructura de la MongoDB por lo que el primer paso es crear un modelo de la estructura. Mongoose proporciona una API para que esta tarea sea más sencilla.

Con el modelo creado en un fichero independiente, dos ficheros más serán los importantes en el apartado de *backend*. El fichero que lanzará el servidor, *app.js,* y el controlador que proporcionará la API de la aplicación Web, *controller.js.*

En el fichero *app.js* se configura Express indicándole dónde se encuentran los ficheros, el tipo de log que se desea, y como tratar las peticiones HTTP. Se configura la base de datos pasándole un parámetro con la dirección, del mismo modo que se hizo en la aplicación de análisis. También se le indica dónde se encuentra el fichero *controller.js* que manejará las peticiones. Para esta tarea es mejor utilizar un fichero extra, *routes.js,* dónde se indica el tipo de petición que va a aceptar y la dirección dónde la puede aceptar, y desde este fichero realizar una llamada al controlador que ejecutará la función pertinente.

Se van a utilizar dos direcciones para aceptar dos peticiones diferentes, búsqueda (/search) y predicción (/predict).

La petición de búsqueda se debe formar con cuatro parámetros, equipo local, equipo visitante, fecha de inicio de temporada, y fecha de final de temporada. Con estos cuatro parámetros se realiza con Mongoose una búsqueda en la base de datos que devuelve los partidos que cumplen las condiciones.

La petición de predicción se debe formar con cinco parámetros, equipo local, jugadores del equipo local, equipo visitante, jugadores del equipo visitante y fecha del partido. Estos parámetros se modifican y se guardan en un fichero de texto para que posteriormente Spark pueda leerlo. Una vez que Spark finalice

con la predicción generará otro fichero de texto dónde se encontraran las predicciones que serán devueltas a la aplicación en formato JSON.

Una vez realizada está configuración solo falta indicarle a Express dónde se encuentran los ficheros del *frontend,* se utilizará la carpeta *public*, y configurar el servidor para que escuche las peticiones en un puerto, se utilizará el 3001.

Con esto el *backend* queda configurado y a la espera de ser lanzado con Node.js. El comando para lanzar el servidor es:

```
node app.js
```

3.3.2. Frontend

El *frontend* será creado íntegramente con AngularJS.

Se va a crear un aplicación siguiendo el patrón MVC en el que se tendrán ficheros separados para cada parte. En esta aplicación no se usará un modelo para representar los datos, ya que se obtendrán directamente desde la base de datos o estarán de manera explicita en el código, "*hardcodeado"* en el controlador. De manera que la interfaz de usuario de la aplicación será definida en dos ficheros. El controlador, *main.controller.js,* y la vista, *main.view.js.*

La vista está dividida en dos bloques.

- En el primer bloque se eligen los equipos, local y visitante, y la temporada dónde se jugó el partido que se quiere predecir mediante unos menús desplegables. Una vez configurado estas tres variables será necesario realizar la petición de búsqueda al *backend,* para que nos proporcione la fecha o las fechas en las que se jugaron partidos de la selección, a través de un botón. El *backend* devuelve las fechas y tras la selección de una fecha se activa otro botón para realizar la predicción.
- El segundo bloque muestra los resultados reales del partido y la predicción realizada por la aplicación de análisis.

El controlador proporciona los datos de los equipos y las temporadas disponibles en la base de datos para los menús desplegables de selección de la vista. También contiene las dos funciones vinculadas a los botones de la vista, búsqueda y predicción.

- La función para la búsqueda solicita una petición al *backend* y obtiene los partidos completos que cumplen los parámetros de la búsqueda. Una vez obtenido los partidos se configura el menú desplegable para selección de fechas con las fechas de los partidos obtenidos.
- La función de predicción pasa los parámetros necesarios al *backend,* y espera una respuesta en formato JSON que contiene los jugadores y la puntuación predicha.

Capítulo 4. Análisis de resultados

En este capítulo se realizaran dos análisis. Cada análisis tendrá un modelo diferente y al final del capítulo se compararán los resultados de ambos.

Para realizar el análisis se ha optado por utilizar el mismo partido con el fin de encontrar el modelo que más se ajusta. El partido elegido es el Boston Celtics contra New York Knicks del día 26 de Marzo de 2013 (Temporada 12/13).

Los jugadores de los Celtics que formaron parte del partido fueron Jordan Crawford, Jeff Green, Paul Pierce, Avery Bradley, Brandon Bass, Terrence Williams, Jason Terry, Chris Wilcox, Shavlik Randolph y D.J. White. En los Knicks jugaron Carmelo Anthony, Raymond Felton, Pablo Prigioni, Iman Shumpert, Kenyon Martin, J.R. Smith, Jason Kidd, Steve Novak, Chris Copeland y James White.

4.1. Primer análisis (A1)

En el análisis A1 se ha optado por utilizar un modelo basado en toda la información disponible de los jugadores. Este modelo utilizará todos los partidos en los que el jugador haya participado antes sin tener en cuenta si estaba jugando en otro equipo o la fecha de los partidos. Durante este análisis se observaran los cambios en el modelo para cuatro jugadores, dos de los Celtics, Jordan Crawford y Jeff Green, y dos de los Knicks, Carmelo Anthony y Raymond Felton. Aunque el análisis solo se enfoque en estos jugadores, al final se mostraran los resultados de todos los jugadores.

Tal y como se proponía en el capitulo 2, en el apartado de regresión simple, se mostrarán los datos de entrenamiento como una nube de puntos para obtener un gráfico de dispersión. Estos gráficos se corresponden con los puntos anotados de los jugadores y el número de partidos. A través de estos datos lo que se intentará conseguir con la aplicación de análisis es la línea de regresión que más se ajuste a la nube de puntos.

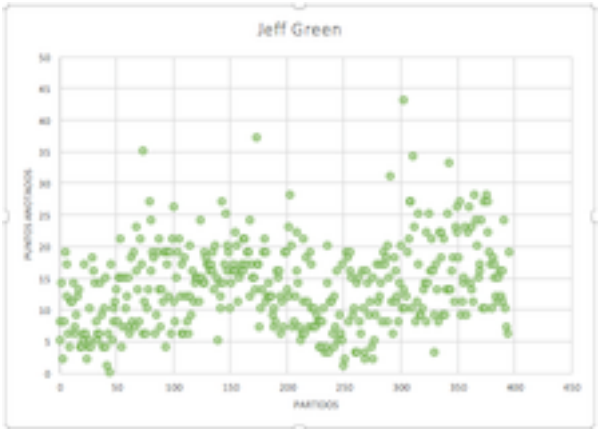

Figura 4.1 Gráfico de dispersión de Jeff Green

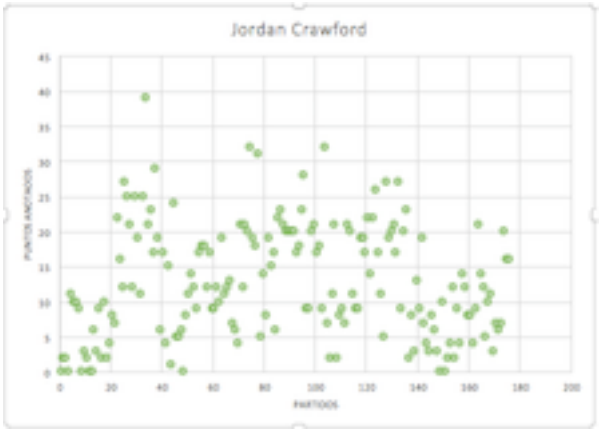

Figura 4.2 Gráfico de dispersión de Jordan Crawford

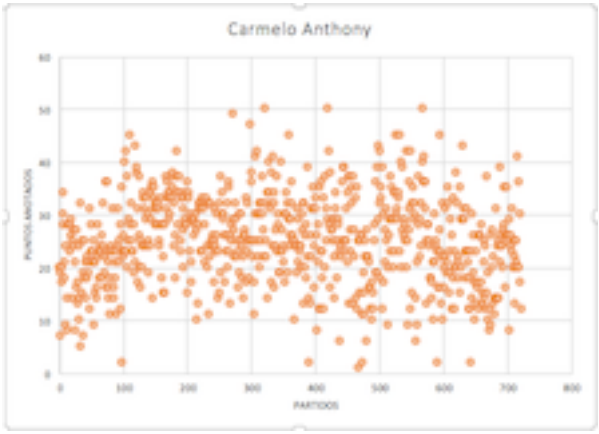

Figura 4.3 Gráfico de dispersión de Carmelo Anthony

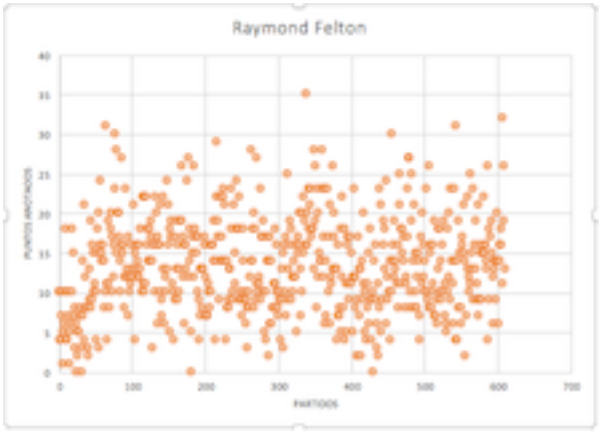

Figura 4.4 Gráfico de dispersión de Raymond Felton

Mediante una inspección visual se puede deducir que los jugadores más regulares obtendrán una mejor predicción ya que su nube de puntos es menos dispersa. En la Figura 4.2 y la Figura 4.4 se observa que no existe ninguna relación lineal. La Figura 4.1 parece que se ajustaría a una regresión

trigonométrica, es decir, tampoco parece tener relación lineal. En la Figura 4.3 se observa a un jugador más regular, el cual puede tener una regresión lineal que se ajuste a su nube de puntos.

Para poder ajustar mejor el modelo se mostrarán los resultados reales del partido y a partir de ahí se irán ajustando los modelos.

Una vez obtenida la anotación de los jugadores se comienza a entrenar el modelo. El primer resultado de este modelo se obtendrá realizando un sobreentrenamiento, es decir, utilizando todos los datos para entrenar el modelo.

Con este primer entrenamiento se obtienen los primeros resultados:

Tabla 4.1 Primeros resultados del modelo A1

Jugador	**Resultado real**	**Y = a + bX (predicción)**	**ECM**	**R^2**
Jeff Green	19	16.27	6.465	0.052
Jordan Crawford	14	12.80	8.023	1.7E-5
Carmelo Anthony	29	24.27	8.449	0.002
Raymond Felton	6	14.31	6.091	0.008

Como se ha deducido de la inspección visual estos jugadores con este modelo no tienen un buen ajuste. Sin embargo con estos datos se puede afirmar que el ajuste lineal es más efectivo en Raymond Felton que en Carmelo Anthony, algo que a simple viste no parecía deducirse. También se observa que el modelo que aporta más información es el de Jeff Green, tiene mayor coeficiente de determinación, y en la inspección visual era el único que no parecía tener ninguna relación lineal.

Para los siguientes resultados se utilizará un modelo con el factor 70% para el entrenamiento y 30% para la validación.

Tabla 4.2 Resultados del modelo A1 entrenado con factor 70-30

Jugador	**Resultado real**	**Y = a + bX (predicción)**	**ECM**	**R^2**
Jeff Green	19	17.13	6.416	0.083
Jordan Crawford	14	13.09	7.865	0.001
Carmelo Anthony	29	24.76	8.714	6.7E-4
Raymond Felton	6	14.34	6.265	0.004

Tras la simulación se obtienen unos resultados con los que se puede afirmar que las predicciones para ese partido han sido mejores. Se observa que el ECM ha descendido para todos los jugadores excepto Raymond Felton con lo cual el modelo con este factor es más efectivo que el sobreentrenado. El coeficiente de determinación también ha descendido para los jugadores de los Knicks, mientras que ha aumentado para los jugadores de los Celtics. Esto indica por el momento que los ajustes lineales tienen mas sentido en los jugadores de los Celtics a pesar de que en la inspección visual tenia más sentido para los jugadores de los Knicks.

Por último para conseguir el mejor ajuste se modificará la aplicación para que no se limite a seleccionar un solo conjunto de datos aleatoriamente para el entrenamiento del modelo, sino que se irán seleccionando conjuntos hasta obtener el que menor ECM proporcione. En concreto se escogen 100 diferentes conjuntos de datos para entrenar el modelo. Los resultados para el mejor modelo se representan en la Tabla 4.3.

Tabla 4.3 Resultados del modelo A1 entrenado con 100 conjuntos de datos diferentes

Jugador	**Resultado real**	**Y = a + bX (predicción)**	**ECM**	**R^2**
Jeff Green	19	15.52	5.929	0.035
Jordan Crawford	14	13.44	7.348	0.003
Carmelo Anthony	29	23.70	8.108	0.006
Raymond Felton	6	14.95	5.853	0.028

Atendiendo al coeficiente de determinación se observa que los modelos siguen sin explicar una relación lineal. A pesar de ello las predicciones se van aproximando cada vez más al resultado real y el ECM sigue descendiendo. Para sacar una conclusión de este modelo es mejor realizar el análisis A2 y comparar los resultados. Antes de pasar al análisis A2 se presentan los gráficos de dispersión con la línea de dispersión obtenida con los últimos resultados del análisis.

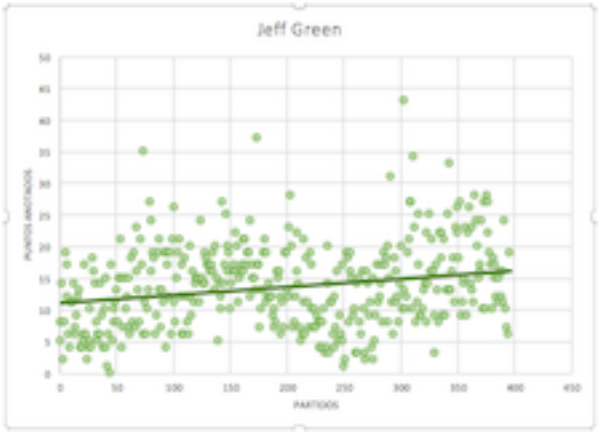

Figura 4.5 Dispersión y línea de regresión J.Green

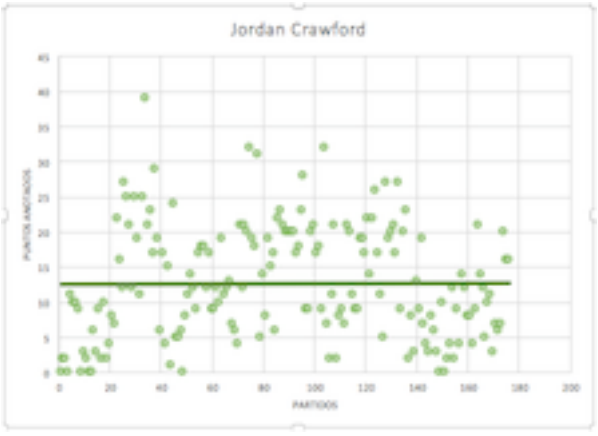

Figura 4.6 Dispersión y línea de regresión J.Crawford

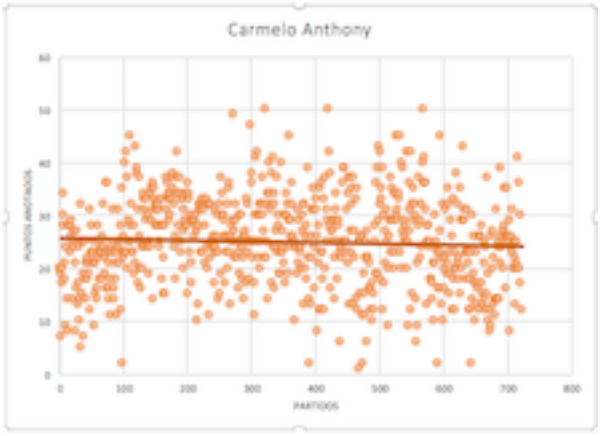

Figura 4.7 Dispersión y línea de regresión C.Anthony

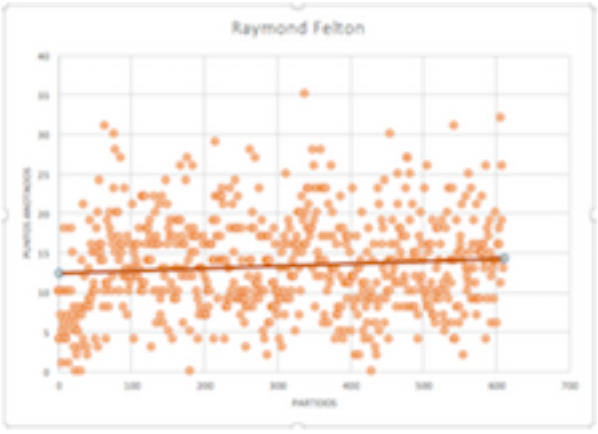

Figura 4.8 Dispersión y línea de regresión R.Felton

4.2. Segundo análisis (A2)

Para este análisis se usará un modelo compuesto de tres conjuntos de datos.

- Selección de partidos del jugador analizado enfrentándose al equipo contra el que disputa el partido teniendo en cuenta si jugaba como local o visitante y sin tener en cuenta el equipo al que pertenecía. Por ejemplo, en el caso de Carmelo Anthony se seleccionaran todos los partidos que ha disputado como visitante ante los Celtics, es decir, se incluirán resultados de cuando jugaba en los Denver Nuggets. (Histórico VS)
- Selección de partidos disputados en el histórico del mismo mes que se disputa el partido a predecir. Se tendrán en cuenta todos los partidos sin importar dónde jugaba (local o visitante) ni el equipo en el que jugaba. Por ejemplo, el partido analizado se disputó el día 26 de Marzo, por lo que se tendrán en cuenta todos los partidos disputados entre el 1 de Marzo y el 25 de Marzo de años anteriores. (Histórico Mes)
- Selección de los partidos disputados los últimos siete días de la temporada del partido a predecir. (Últimos 7 días)

El orden en el que se dispongan los datos influye en el modelo por lo que se ha decido unir los datos como se representan en la Tabla 4.4.

Tabla 4.4 Disposición de los datos modelo A2

Histórico Mes	Histórico VS	Últimos 7 días

Para guardar relación con el análisis A1 se utilizaran los mismos cuatro jugadores para ir observando la evolución del modelo.

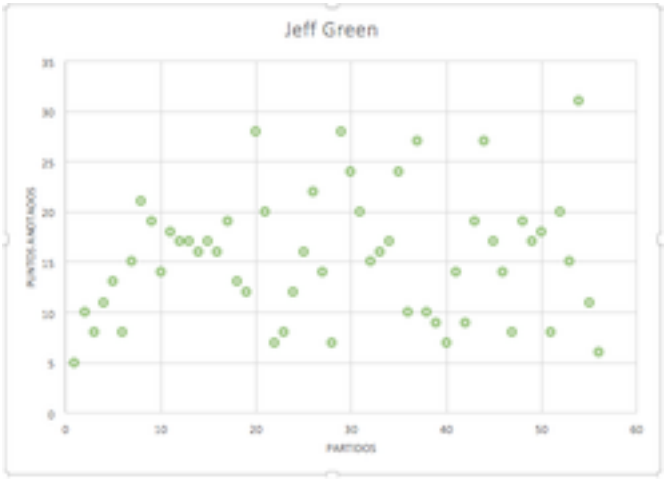

Figura 4.9 Dispersión modelo A2 J.Green

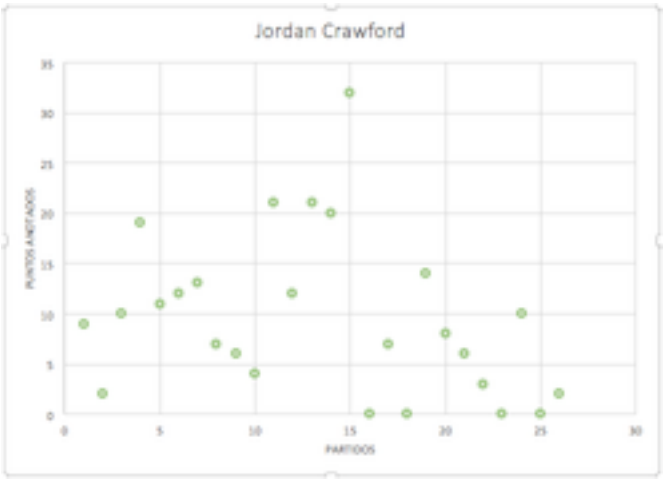

Figura 4.10 Dispersión modelo A2 J.Crawford

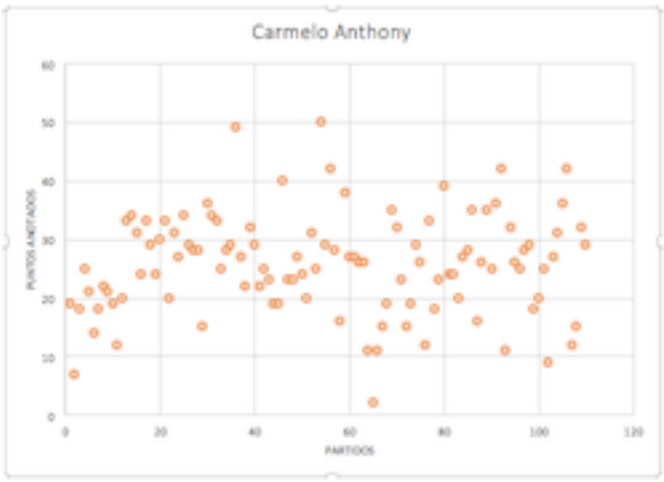

Figura 4.11 Dispersión modelo A2 C.Anthony

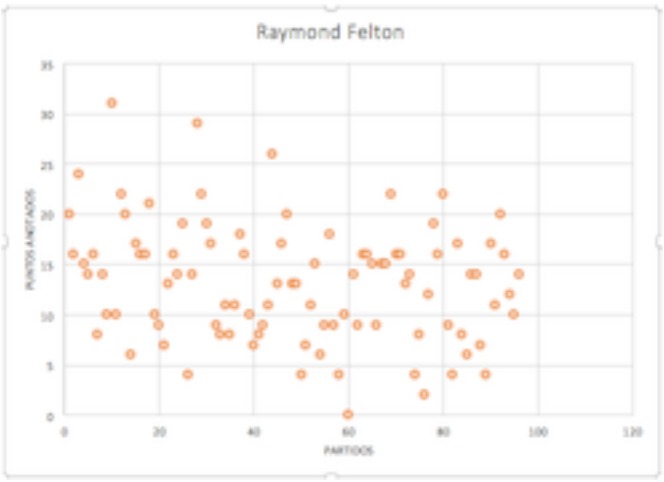

Figura 4.12 Dispersión modelo A2 R.Felton

Con este modelo a simple vista se observa que los gráficos de dispersión están más dispersos. Al haber menos datos se puede observar que en ninguna figura existe una relación lineal. En principio se puede decir que este modelo será peor que el utilizado en el análisis A1.

Los primeros resultados se mostraran con un modelo sobreentrenado en la Tabla 4.5

Tabla 4.5 Primeros resultados del modelo A2

Jugador	**Resultado real**	**Y = a + bX (predicción)**	**ECM**	**R^2**
Jeff Green	19	16.74	6.123	0.015
Jordan Crawford	14	5.54	7.528	0.015
Carmelo Anthony	29	26.32	8.480	0.002
Raymond Felton	6	10.89	5.665	0.051

Si se observan los valores del coeficiente de determinación se deduce que el análisis de regresión lineal sigue estando muy lejos de un ajuste perfecto. Los ECMs medios siguen siendo altos y las predicciones no son correctas.

En el análisis A1 se observó una mejora pasando a entrenar el modelo con el factor 70-30. Los resultados de este análisis se muestran en la Tabla 4.6.

Tabla 4.6 Resultados del modelo A2 entrenado con factor 70-30

Jugador	**Resultado real**	**Y = a + bX (predicción)**	**ECM**	**R^2**
Jeff Green	19	17.80	5.544	0.089
Jordan Crawford	14	5.48	7.431	0.078
Carmelo Anthony	29	26.25	8.505	0.004
Raymond Felton	6	11.00	5.701	0.051

Tras esta simulación se observa un aumento del coeficiente de determinación, que implica una pequeña mejora en el modelo. Sin embargo, el ECM de los

jugadores de los Knicks ha aumentado mientras que el de los Celtics ha disminuido. Y las predicciones siguen estando lejos del resultado real.

A continuación se prueba con la misma simulación que en el análisis A1, se dividen los datos de manera aleatoria y si entrena el modelo con cien conjuntos de datos diferentes. Los resultados se muestran en la Tabla 4.7.

Tabla 4.7 Resultados del modelo A2 entrenado con 100 conjuntos de datos diferentes

Jugador	**Resultado real**	**Y = a + bX (predicción)**	**ECM**	**R^2**
Jeff Green	19	15.04	5.310	0.027
Jordan Crawford	14	0.32	4.546	0.433
Carmelo Anthony	29	26.04	7.393	0.004
Raymond Felton	6	12.52	4.989	0.020

Con esta ultima simulación se observan cambios importantes. A pesar de que las predicciones son peores en general, el coeficiente de determinación de J. Crawford indica que con este modelo una regresión lineal puede obtener buenos resultados. Los ECMs se han reducido pero aun siguen siendo altos. A continuación se muestran los gráficos de dispersión con las líneas de regresión para este modelo.

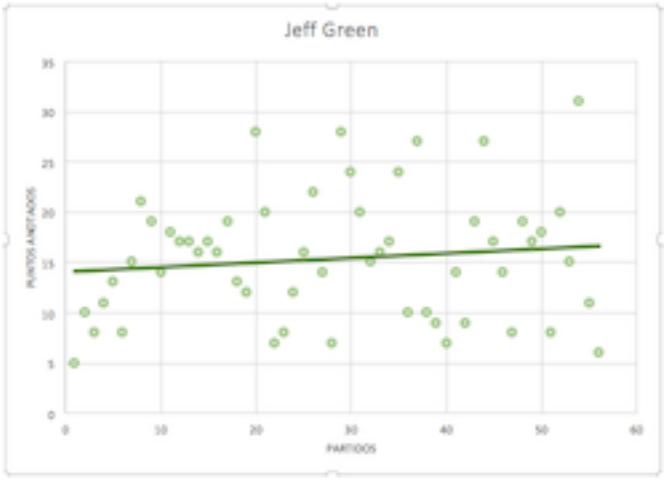

Figura 4.13 Dispersión y línea de regresión A2 J.Green

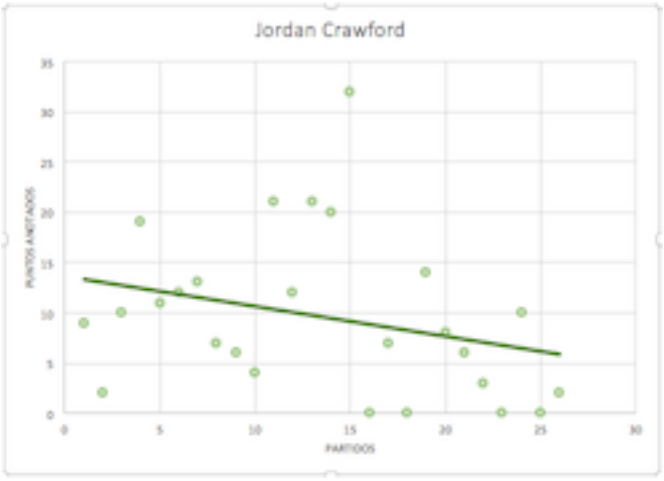

Figura 4.14 Dispersión y línea de regresión A2 J.Crawford

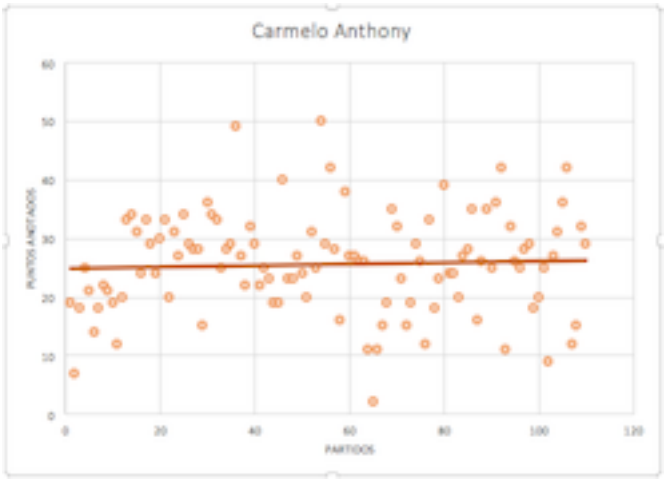

Figura 4.15 Dispersión y línea de regresión A2 C.Anthony

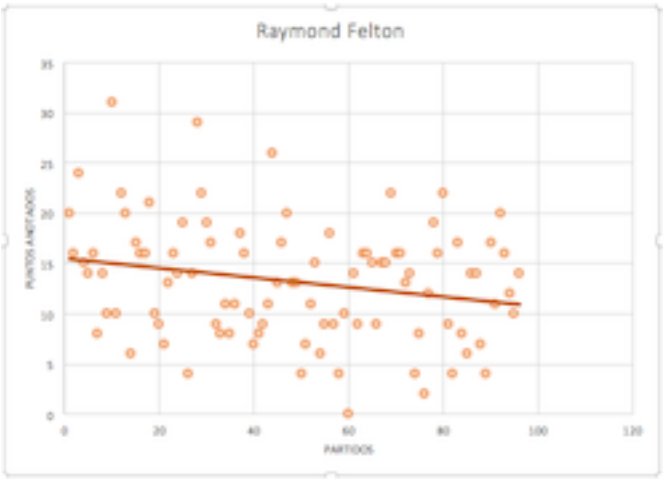

Figura 4.16 Dispersión y línea de regresión A2 R.Felton

4.3. Comparación de resultados

En los análisis se ha buscado reducir el ECM sin tener en cuenta las predicciones para el partido. Hay que tener en cuenta que las estadísticas mostradas en los análisis del ECM y del coeficiente de determinación se han obtenido en base a las predicciones realizadas para los conjuntos de datos comprendidos en el 30% del conjunto total, es decir, los modelos se han entrenado con el 70% y las predicciones se han hecho con los datos restantes. Los valores de las predicciones para el partido se estudian a continuación con los datos obtenidos en la ultima simulación de cada análisis para todos los jugadores del partido, Tabla 4.8.

Tabla 4.8 Resultados de los modelos A1 y A2 para todos los jugadores, y resultado real

Jugador	**R**	**A1**	**A2**	***Jugador***	**R**	**A1**	**A2**
Jordan Crawford	14	11	4,55	*Raymond Felton*	6	13,82	9,72
Avery Bradley	6	11,94	15,26	*Carmelo Anthony*	29	23,88	27,67
Terrence Williams	7	5,57	3,48	*Pablo Prigioni*	7	3,42	0
Paul Pierce	16	20,12	20,63	*James White*	0	0,26	0
Brandon Bass	11	11,32	10,97	*Chris Copeland*	1	13,26	11
Shavlik Randolph	2	2,88	1,11	*Steve Novak*	9	8,37	9,12
Jeff Green	19	15,62	18,66	*Iman Shumpert*	2	4,64	3,72
D.J. White	0	5,6	1,4	*J.R. Smith*	32	17	11,47
Jason Terry	10	16,05	14,99	*Kenyon Martin*	9	10,46	13,01
Chris Wilcox	0	7,77	5,36	*Jason Kidd*	5	9,6	11,41
TOTAL	85	107,87	96,41	TOTAL	100	104,71	97,12

La Tabla 4.8 muestra los resultados reales (R), los resultados del análisis A1 y los del análisis A2. A simple vista parece que el computo de los resultados predichos para el equipo de los Knicks es más fiable que el de los Celtics. Para los Knicks hay una diferencia de +4/-3 mientras que para los Celtics es de +23/+9. El ECM obtenido para estos resultados se muestra en la Tabla 4.9.

Tabla 4.9 ECM de los resultados de todos los jugadores

Equipo	**ECM – A1**	**ECM – A2**
Boston Celtics	20.39	26.54
New York Knicks	50,62	64,62

Se observa que el ECM es superior con los resultados de los Knicks que con los datos de los Celtics, sin embargo las predicciones parecían ser mejores para los Knicks. Se observa que analizando jugador a jugador, prediciendo el resultado y luego haciendo un computo global, los resultados no son buenos. La acumulación de errores en la predicción de cada jugador hace que la suma de los errores sea alta. Esto hace que las predicciones estadísticamente sea malas, a pesar de que en este caso parezcan que se han aproximado al resultado. Según estos parámetros que la predicción se haya acercado tanto en el resultado de los Knicks ha sido puro azar.

Se va a realizar un análisis directamente con los resultados anteriores de los equipos, es decir, se va a utilizar un modelo basado en el resultado final de los partidos, obteniendo todos los datos disponibles de la base de datos. El modelo será parecido al utilizado en el análisis A1, pero esta vez solo se tomarán los resultados finales, sin atender a los puntos anotados por jugador. Para ello se va a modificar la aplicación solo para realizar este análisis y mostrar unos resultados con los que comparar las predicciones obtenidas anteriormente y observar si un modelo global es mejor que un modelo individual.

4.4. Análisis de modelo global

Para obtener el modelo global se sigue usando un filtrado de dos bloques. En el primer bloque esta vez se usa un filtro de tres etapas para la agregación de MongoDB. En el segundo se utiliza un filtro que proporciona Spark para obtener solo los puntos anotados por los equipos. Al igual que en los análisis anteriores se utiliza un factor 70-30 para dividir los datos en dos conjuntos, uno de entrenamiento y otro de evaluación. Los resultados que se muestran en la Tabla 4.10 se han obtenido simulando con cien conjuntos de datos diferentes para obtener el menor ECM y la mejor predicción posible.

Tabla 4.10 Resultados del modelo global entrenado con 100 conjuntos de datos diferentes

Equipo	**Predicción**	**Res. Real**	**ECM**	**R^2**
Boston Celtics	87.49	85	12.69	0.025
New York Knicks	93.77	100	12.63	0.004

Las predicciones obtenidas son mejores que con los modelos obtenidos en los análisis A1 y A2. El ECM comparado con el del mejor modelo se reduce en un 50% aunque el coeficiente de determinación sigue indicando que un ajuste lineal no es el más adecuado. En la Figura 4.17 y la Figura 4.18 se muestran las líneas de regresión junto a la nube de puntos de ambos equipos. Se observa que la nube de puntos está menos dispersa lo que confirma los datos obtenidos en la Tabla 4.10. Comparando los gráficos de dispersión de todos los modelos se puede afirmar que de los utilizados en esta memoria, un modelo con resultados globales es más efectivo que un modelo basado en la predicción de cada jugador. Los datos obtenidos para cada modelo indican que los resultados son mejores para el modelo global pero aun siguen muy lejos de definirse perfectamente con un ajuste lineal.

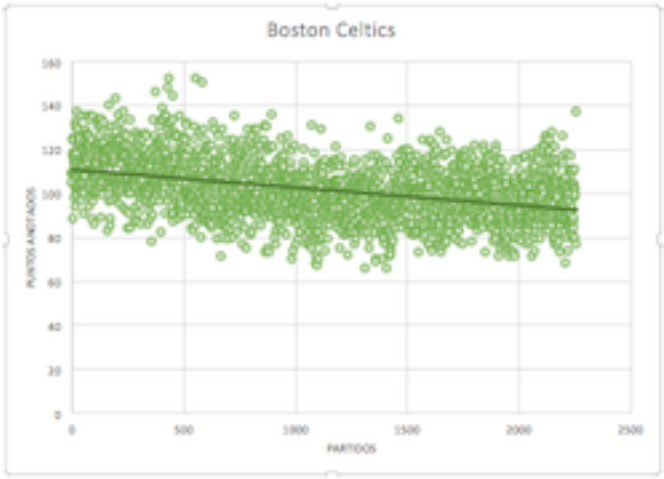

Figura 4.17 Dispersión y línea de regresión Boston Celtics

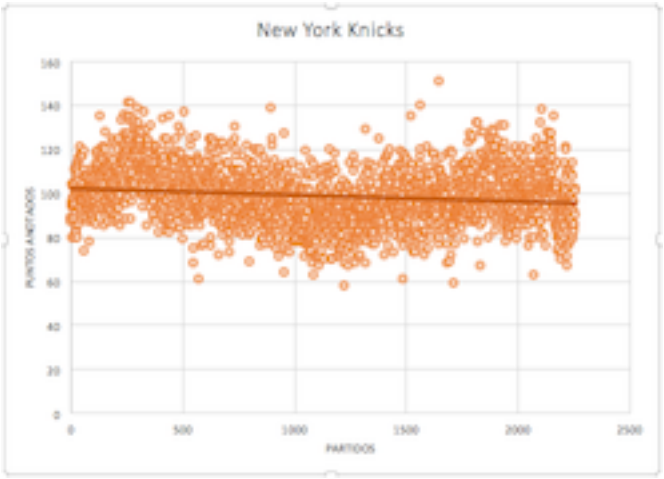

Figura 4.18 Dispersión y línea de regresión New York Knicks

Capítulo 5. Conclusiones y trabajo futuro

Una vez se han realizado los análisis para cada uno de los modelos expuestos en esta memoria y tras haber utilizado la aplicación para comprobar su funcionamiento se plantean a continuación las conclusiones a las que se ha llegado.

5.1. Aplicación de análisis predictivo

La aplicación desarrollada cumple los objetivos propuestos para el diseño inicial, dado que es capaz de predecir resultados y mostrarlos a través de la aplicación Web. Pese a ello cabe destacar algunos defectos relativos al diseño que se ha llevado a cabo.

A pesar de que se han utilizado diferentes modelos para realizar los análisis, la aplicación no permite seleccionar el modelo que se desea utilizar, ni guardar los modelos entrenados, ya que para cada partido se crea un nuevo modelo para cada jugador. El modelo utilizado para mostrar los resultados finales es el que mejor resultado ha obtenido, el modelo del análisis A1.

Las gráficas obtenidas en el análisis no se muestran en el resultado final de la aplicación. Tampoco los parámetros estudiados para interpretar los modelos, solo se muestran las predicciones de anotación, ya que estos parámetros sirven para comprender la eficacia de los modelos.

Estas desventajas no se encontraban como punto relevante u objetivo de la aplicación, la cual pretende en primera instancia asentar las bases para continuar un trabajo futuro sobre la misma. Se proponía utilizar diferentes modelos para realizar los análisis pero no para mostrar los resultados. Es por

ello, que no se ha profundizado más en la mejora de la aplicación dejando de esta manera abierta la posibilidad de ampliar dicha aplicación en la dirección que se estime más adecuada.

5.2. Líneas futuras

Como ya se ha indicado en el apartado anterior, la aplicación funciona de forma correcta, permitiendo realizar una predicción mediante técnicas big data. No obstante, es posible mejorar la aplicación en muchos aspectos, algunos de los cuales se detallan a continuación.

La selección del modelo que se quiere utilizar se presenta como una desventaja, el hecho de poder seleccionar un modelo diferente para realizar predicciones facilitaría el uso de la aplicación. Así mismo, guardar los modelos entrenados supondría un ahorro en tiempo de ejecución.

Al tener un componente de aplicación Web, la aplicación se puede mejorar incluyendo librerías para la elaboración de gráficas y poder realizar inspecciones visuales de los modelos utilizados, sobre todo con las gráficas de dispersión y líneas de regresión.

En este documento solo se ha realizado un análisis mediante regresiones lineales, observándose que en la mayoría de casos no eran efectivas. Una posible ampliación de la aplicación para que use diferentes ajustes, como hiperbólicos por ejemplo, proporcionaría diferentes predicciones con las que elaborar mejores análisis.

Esta memoria comienza con una cita que ahora se puede transformar y decir que predecir es sencillo, la dificultad está en acertar.

Referencias

[1] “Apache Spark.” http://spark.apache.org/

[2] “MongoDB.” https://www.mongodb.com/

[3] “Express.” http://expressjs.com/

[4] “AngularJS.” https://angularjs.org/

[5] “Node.js.” https://nodejs.org/en/

[6] Departamento Matemática Aplicada, UMA. Estadística Descriptiva, Curso 2015-2016.

[7] Valeri Karpov, “Crunching 30 Years of NBA Data with MongoDB Aggregation” in The Code Barbarian, Feb 2014. https://thecodebarbarian.wordpress.com/2014/02/14/crunching-30-years-of-nba-data-with-mongodb-aggregation/

Bibliografía

En este apartado se mostraran algunos libros que han servido como lectura previa para desarrollar esta memoria. Aunque no se hacen referencias explicitas a ellos se recomienda al lector su lectura previa si no se tienen nociones básicas de las herramientas que se utilizan para desarrollar la aplicación de este documento.

M. Zaharia, P. Wendell, A. Konwinski, H. Karau. Learning Spark. O'Reilly, 2015.

J. Wills, S. Owen, U. Laserson, S. Ryza. Advanced Analytics with Spark. O'Reilly, 2015.

M. Guller. Big Data Analytics with Spark. Apress, 2015.

M. Armbrust, R. S. Xin, C. Lian, Y. Huai, D. Liu, J. K. Bradley, X. Meng, T. Kaftan, M. J. Franklin, A. Ghodsi, M. Zaharia, "Spark SQL: Relational Data Processing in Spark", SIGMOD'15, Melbourne, Victoria, Australia, 2015.

K. Chodorow. MongoDB: The Definitive Guide. O'Reilly, 2013.

Apéndice A. CD-ROM

Con la intención de simplificar el acceso al código utilizado en el desarrollo de este trabajo y la redacción de la memoria correspondiente se provee un CD-ROM adjunto a la memoria.

El CD-ROM contiene:

- Una copia en formato digital del documento redactado, denominada “Análisis predictivo con técnicas big data.pdf”.
- Un directorio, “core”, que incluye los archivos de código utilizados para el desarrollo de la aplicación de análisis.
- Un directorio, “web”, que incluye los archivos de código utilizados para el desarrollo de la aplicación web, además de la aplicación de análisis empaquetada en un fichero JAR, y dos ficheros de texto que se utilizan en el funcionamiento de la aplicación.
- Un fichero llamado “games.bson” que contiene una copia de la base de datos (*dump* de MongoDB).