UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Predicción y análisis de interacciones de usuarios en plataformas de enseñanza online

Miguel Ángel González-Gallego Sosa
Tutor: Estrella Pulido Cañabate

Junio 2016

Predicción y análisis de interacciones de usuarios en plataformas de enseñanza online

AUTOR: Miguel Ángel González-Gallego Sosa
TUTOR: Estrella Pulido Cañabate

Dpto. Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Junio de 2016

Resumen

Las plataformas de enseñanza online generan gran cantidad de metadatos sobre las interacciones entre los estudiantes y con la plataforma. Esta información puede ser aprovechada por los profesores de los cursos para mejorar el curso y la experiencia docente de los estudiantes. En este contexto el objetivo de este TFG es el análisis de las interacciones realizadas por los estudiantes en cursos online y la predicción del comportamiento del estudiante utilizando su patrón de acceso a la plataforma.

Debido al volumen de datos que se maneja se hará uso herramientas de computación en paralelo como Apache Spark para preprocesar los datos generados por la plataforma. Mediante Apache Spark se creará una aplicación que extraiga el patrón de acceso de los estudiantes a la plataforma y disminuya la gran cantidad de metadatos generada en un curso online.

Por último, se aplicarán algoritmos de aprendizaje automático para predecir variables de interés sobre la interacción de los estudiantes con el curso como la probabilidad de abandono o el rendimiento académico. Esto también se realizará con la herramienta Apache Spark. En concreto, se utilizará el algoritmo *Random Forest* de la librería MLlib de Spark con la finalidad de obtener el mejor resultado a la hora de predecir las variables de interés del curso.

Palabras clave

Big Data, Apache Spark, edX, Mooc, RDD, Random Forest.

Abstract

Online education platforms generate a lot of metadata about interactions among students and with the platform. This information can be harnessed by teachers to improve the course and student's teaching experience. In this context the aim of this study is the analysis of interactions performed by students and the prediction of student's behavior using his access patterns to platform.

Due to the volume of data handled, we use a tool for parallel computing such as Apache Spark for preprocessing the data generated by the platform. We create an application that extracts the access patterns to platform and decreases the volume of the metadata generated in this online course.

Finally, we apply machine learning algorithms to predict target variables related to the interactions of students enrolled in the course, for example the dropout rate or the academic performance. We also use the tool Apache Spark for this task. Specifically, we apply the algorithm *Random Forest* from the library MLlib in order to get the best result in predicting the course's target variables.

Keywords

Big Data, Apache Spark, edX, Mooc, RDD, Random Forest

Agradecimientos

En primer lugar agradecer a mi familia por todo el apoyo recibido a lo largo de la carrera haciéndome más llevadero el camino. Por otro lado agradecer a Estrella y Gonzalo el seguimiento durante el proyecto y la ayuda en él, además de la oportunidad que me han brindado al poder realizar este trabajo. Agradezco también a mis compañeros de carrera y amigos, todo el apoyo recibido a lo largo del trayecto y sobre todo a la Cátedra UAM/IBM, que me ha proporcionado una beca, un entorno de colaboración con otros estudiantes y la oportunidad de participar en actividades formativas.

INDICE DE CONTENIDO

INDICE DE FIGURAS

INDICE DE TABLAS

1 Introducción

La idea de este proyecto surge de la motivación por mejorar el rendimiento de los estudiantes y en general, la educación a través internet. La motivación personal ha sido la de poder ampliar mis conocimientos en las áreas del Big Data y del aprendizaje automático. Además de poder aplicar los conocimientos adquiridos en el grado a un problema real, en el ámbito del aprendizaje automático y de la programación en general, en el marco de la cátedra UAM/IBM que me proporciona experiencia laboral de cara a mi futuro profesional.

1.1 Motivación

Los Moocs son cursos en línea masivos que están cobrando una gran relevancia en el ámbito de la educación a través de internet. La educación online es de gran utilidad, debido a que no es necesaria la presencia del alumno para adquirir los conocimientos impartidos en un curso, por lo tanto los Moocs crean una conexión a distancia entre profesores y estudiantes de forma gratuita. Existen distintas plataformas como que ofrecen este tipo cursos online, como son edX [1], Coursera [2], Iversity [3], etc. Cabe resaltar que la Universidad Autónoma de Madrid (UAM) está involucrada en la educación a través de internet, con iniciativas como por ejemplo el curso: "*Aprende a programar tu primera App*" a través de la plataforma edX.

Los Moocs están formados por equipos de profesores universitarios u otras instituciones docentes. Estos profesores son los encargados de compartir el material con el cual se impartirá el curso, videos, documentos, problemas etc. Todo esto es utilizado por los estudiantes de manera online, y sus interacciones quedan registradas en los logs de la plataforma en la que se imparte el curso. Dado el volumen de estudiantes matriculados en estos cursos y el gran número de acciones que realizan, se genera un gran volumen de datos de esas interacciones con información que puede ser relevante o irrelevante dependiendo del estudio que se quiera realizar.

Por lo tanto utilizando herramientas del Big Data como Apache Spark, para el procesamiento de grandes volúmenes de datos, se pretende llevar a cabo un preprocesamiento de la información registrada en los logs con la finalidad de obtener únicamente información relevante para nuestros casos de estudio acerca de los estudiantes, eliminando interacciones innecesarias, interacciones repetidas e información superflua de las propias interacciones relevantes.

En definitiva el objetivo de este proyecto ha sido la predicción de las interacciones de los estudiantes con la plataforma a partir de sus interacciones previas utilizando herramientas del Big Data como Apache Spark y MLlib para el procesamiento y predicción de grandes volúmenes de datos.

1.2 Objetivos

Los objetivos de este trabajo han sido:

- Limpiar y extraer la información relevante del log de un curso con el objetivo de obtener información útil sobre él. Esta información a procesar es de gran volumen y posee inconsistencias lo que hace más compleja la realización de la tarea. Para

solventar este grado de dificultad, emplearemos técnicas de Big Data, como el modelo de programación MapReduce que es una tecnología proporcionada por Apache Spark. Este objetivo requiere de la investigación de los eventos más relevantes del log y ha sido de elevada complejidad debido a la escasa documentación.

- El objetivo principal de este trabajo es la predicción del rendimiento académico de los estudiantes a partir de la información proporcionada por sus interacciones con el curso. Esta predicción se llevará a cabo mediante herramientas de aprendizaje automático. Con ellas seremos capaces de predecir la nota que van a sacar los estudiantes, y el resultado de los ejercicios, a través de la información referente a los problemas que han realizado. También se comparará el error resultante de la predicción de la nota en las diferentes semanas y se analizarán los errores obtenidos. Por otro lado se estudiarán los errores procedentes de la predicción del resultado de los problemas a medida que vamos avanzando en el curso.

1.3 Competencias adquiridas.

La complejidad del trabajo ha ocasionado la necesidad de ampliar los conocimientos y de adquirir nuevos, como por ejemplo las siguientes herramientas, técnicas y lenguajes:

- **Apache Spark**: herramienta para el manejo de grandes volúmenes de datos.[4]
- **Modelo de programación MapReduce**: es un modelo de programación diseñado para el procesamiento en paralelo de grandes volúmenes de datos dividiendo el trabajo en un grupo de tareas independientes [5].
- **RDDs**: son colecciones de datos distribuidas sobre las que trabaja Spark [6].
- **Python**: lenguaje de programación interpretado que soporta la creación de clases programación imperativa y funcional.
- **Random Forest:** algoritmo de predicción de datos basado en árboles de decisión [7].
- **MLlib:** librería de aprendizaje automático proporcionada por Spark [8].
- **Sklearn:** librería de aprendizaje automático proporcionada por Python [9].

1.4 Organización de la memoria

La memoria consta de los siguientes capítulos:

- **Estudio del estado del arte:** en este capítulo se expondrá el contexto en que se sitúan tanto los métodos que se van a emplear como las herramientas que se manejan para ello.
- **Desarrollo:** en este capítulo se explicarán los formatos de los eventos más relevantes del log además de la metodología seguida para crear el fichero que contiene las interacciones importantes de cada estudiante ordenadas cronológicamente. Posteriormente se expondrá el proceso para la creación de la

aplicación de clasificación del rendimiento de los estudiantes en función de los problemas que han realizado.

- **Pruebas y resultados:** en este capítulo se explicarán las pruebas realizadas, aplicando la metodología explicada en el apartado de desarrollo. Además se explicarán los resultados obtenidos en cada etapa.

- **Conclusiones y trabajo futuro:** en este capítulo se resumirán las ideas obtenidas tras la conclusión del trabajo y como han sido resueltos los objetivos establecidos al comienzo del mismo. Además se indicarán las posibles líneas de trabajo futuro.

2 Estado del arte

En este capítulo se describen los aspectos más relevantes acerca de los temas sobre los que trata el trabajo.

2.1 Mooc

Los MOOC (acrónimo en inglés de Massive Open Online Course) o COMA en español (Curso Online Masivo Abierto) son cursos en línea dirigidos a un amplio número de participantes a través de Internet según el principio de educación abierta y masiva [10].

2.1.1 Características de los Moocs

Algunas características de este tipo de cursos son las siguientes:

- Tienen carácter masivo es decir se permite el acceso de numerosos estudiantes.
- El acceso es libre y abierto. No son necesarios conocimientos previos, ni pertenecer a la institución que imparte el curso.
- El acceso es gratuito, no es necesario el pago por los contenidos que oferta el curso.
- Los cursos son accesibles en línea de manera que se utilizan distintos tipos de contenido como audio, texto, video y animación que permiten explotar la potencialidad de internet.
- Se trata de cursos orientados al aprendizaje por lo que se realizan pruebas que garantizan haber adquirido los conocimientos que oferta el curso.

2.1.2 edX

edX es una plataforma de cursos en línea masivos y abiertos (MOOC) fundada en mayo de 2012 para ofertar cursos online de nivel universitario sobre un amplio rango de disciplinas accesibles para todo el mundo y gratuitos con el objetivo de fomentar la investigación y el aprendizaje [11].

Este trabajo se basa en datos registrados en un log de la plataforma edX. En concreto los datos pertenecen al curso de programación en Android impartido por profesores de la Escuela Politécnica Superior de la Universidad Autónoma de Madrid.

2.2 Big Data

Big Data hace referencia a cantidades masivas de datos que han sido acumuladas con el tiempo y requieren un alto coste de análisis y manejo, mediante las herramientas informáticas tradicionales.

Dicho concepto engloba infraestructuras, tecnologías y servicios que han sido creados para dar solución al procesamiento de enormes conjuntos de datos estructurados, no

estructurados o semi-estructurados (mensajes en redes sociales, señales de móvil, archivos de audio, sensores, imágenes digitales, datos de formularios, emails, datos de encuestas, logs, etc.) que pueden provenir de sensores, micrófonos, cámaras, escáneres médicos, o imágenes.

2.2.1 Dimensiones Del Big Data

El Big Data no solo hace referencia a cantidades masivas de datos sino que se analiza desde cinco dimensiones diferentes que se detallan a continuación [12].

Volumen: el volumen de los datos almacenados en las empresas ha pasado de ocupar megabytes y gigabytes a "petabytes". En el año 2000 se generaron 800.000 petabytes (PB) de información y se espera que en 2020 esta cifra alcance los 35 zetabytes (ZB).

Se estima que este año habrá 18.9 billones de dispositivos conectados. Esto conllevaría que el tráfico de datos móviles alcance 10.8 Exabytes mensuales o 130 Exabytes anuales. Este volumen de tráfico equivale a 33 billones de DVDs anuales.

Valor: esta dimensión consiste en ser capaces de tener el criterio adecuado para analizar solo la información útil en tiempos mínimos y a un coste aceptable.

Variedad: la variedad de datos ha explotado, pasando de ser datos almacenados y estructurados, guardados en bases de datos empresariales, a ser desestructurados y semiestructurados, como por ejemplo, audio, video, XML, etc.

Veracidad: hace referencia a la validez o confianza de los datos que se van a utilizar. Esta característica es de gran relevancia ya que determinará la calidad de los resultados y la fiabilidad de los mismos.

Velocidad: la velocidad refleja la frecuencia con la que los datos se generan, almacenan y comparten.

La velocidad del proceso y captura de datos ha aumentado significativamente. Los modelos basados en inteligencia de negocios generalmente suelen tardar días en procesarse, frente a las necesidades analíticas "casi" en tiempo real de las soluciones financieras.

2.2.2 El Big Data y los Moocs en España

Kenneth Cukier es un periodista estadounidense, gurú en Big Data que ha publicado múltiples artículos y conferencias. Este autor describe en "*La revolución de los datos masivos*" [13] cómo el universo educativo puede beneficiarse de esos datos.

"Lo que ha sucedido en Estados Unidos, con los cursos en línea, llamados MOOCs, cursos masivos abiertos en línea, es que los profesores pueden ver cuando los estudiantes están viendo sus cursos, y cuando se detienen, cuando releen una lección. Y un profesor de Standford se dio cuenta de que hacia la lección siete u ocho todos los estudiantes regresaron a la lección número tres. Esa lección era una clase de repaso de matemáticas y mostraba que a medida que los estudiantes avanzaban más en el curso, estaban menos seguros de sus bases en matemáticas. Normalmente, un profesor no tiene porqué saber que la clase se está quedando atrás, pero de repente, el profesor podía ver esto en los datos y podía aprender dos cosas. En primer lugar, que debía preparar mejor a sus alumnos. Y en

segundo, que debía insistir más en esa dificultad en particular ya que los estudiantes se detenían y regresaban a esa lección".

Kenneth Cukier

Este mismo autor también anuncia una educación que se adapta a las necesidades de cada alumno.

"Tenemos que proponer a nuestros hijos otro sistema educativo ya que éste fue concebido en una época diferente, en la era industrial, mecanicista, basada en una línea de montaje. Ahora se puede adaptar a las recomendaciones de Amazon y Google que se ajustan exactamente a nuestros intereses. Necesitamos una educación que se adapte a nuestras necesidades y esa es la mejor manera de aprender."

El informe de la Comisión Europea "*European MOOC Scoreboard*" sitúa a España como líder en el ranking de producción de MOOCs en estos 3 últimos años con un total de 423 Moocs [14].

2.3 Spark

Spark es un sistema de computación en clúster, rápido y de propósito general para procesamiento de datos a gran escala [4]. Es la evolución de Hadoop y realiza las mismas tareas de procesamiento y análisis con mejores tiempos de respuesta y con funcionalidades adicionales que mejoran su rendimiento. En la Figura 1 se puede observar una comparativa de su rendimiento.

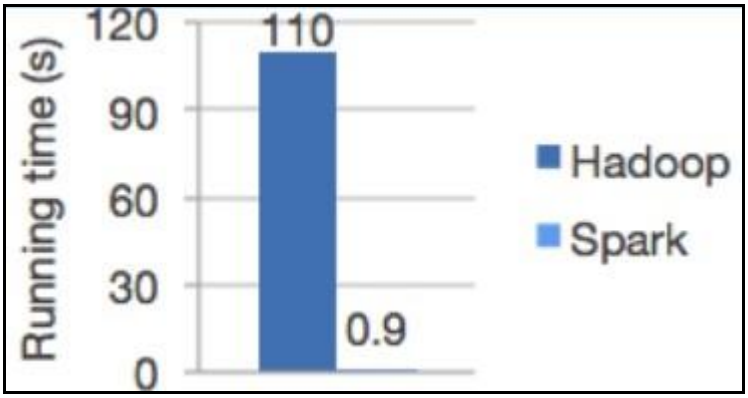


Figura 1. Spark vs Hadoop

2.3.1 Arquitectura de Spark

Spark proporciona API's de alto nivel para Java, Python y Scala. Además de varias herramientas de alto nivel incluyendo: Spark SQL para SQL y procesamiento de datos estructurados, MLlib para machine learning, GraphX para procesamiento de grafos y Spark Streaming.

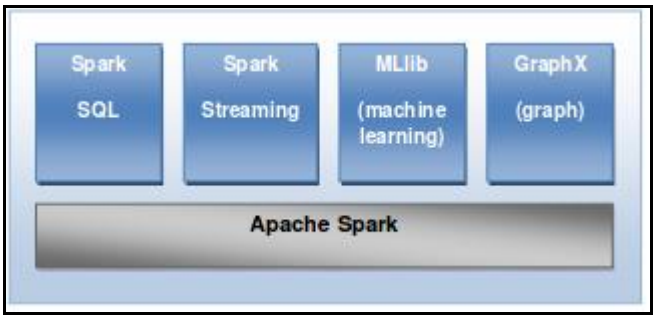


Figura 2. Arquitectura Spark

Spark se puede ejecutar localmente. Sólo es necesario tener instalado Java, haciendo que la variable JAVA_HOME apunte a dicha instalación de java.

Cada aplicación de Spark consiste en un programa *driver*. Este programa es el proceso que ejecuta la función *main* de la aplicación e invoca varias operaciones que son realizadas por los nodos *workers*, que son los núcleos del ordenador que ejecutan código Spark, de manera paralela en el clúster. [15]

2.3.2 RDD (Resilient Distributed Dataset)

En Spark se trabaja sobre colecciones de datos denominados RDDs. Estas colecciones se caracterizan porque las operaciones realizadas sobre ellas se ejecutan en paralelo. Además poseen las siguientes características:

- Son inmutables. Es decir una vez creados no se pueden modificar.
- Se pueden transformar para crear nuevos RDDs o realizar acciones sobre ellos pero no modificar.
- Se guarda la secuencia de transformaciones para poder recuperar RDDs de forma eficiente si alguna máquina se cae.
- Están distribuidos en el clúster en los nodos workers.

2.3.3 Operaciones con RDD

Las operaciones que se realizan con RDDs son dos: acciones y transformaciones. Las transformaciones son aquellas que devuelven un nuevo RDD, mientras que las acciones devuelven un resultado.

Las transformaciones se realizan en modo “lazy evaluation” o “evaluación perezosa”, es decir, Spark no realiza ninguna operación hasta que no encuentra una acción. De esta manera se reduce el número de veces que se recuperan o recorren los datos agrupándolas todas juntas y reduciendo el tiempo de computación. La Figura 3 muestra algunas de las transformaciones que podemos realizar sobre RDDs.

Las acciones son evaluaciones que fuerzan la ejecución de las transformaciones. La Figura 4 muestra algunas de las acciones que se pueden realizar.

2.3.4 MLlib

La librería de aprendizaje automático en Spark es MLlib. Esta librería provee a los programas de algoritmos y utilidades como clasificación, regresión, clustering...

Respecto a la clasificación y regresión, la Figura 5 muestra los métodos disponibles en MLlib.

Función	Descripción
map	Aplica una función a cada elemento del RDD y retorna un nuevo RDD con el resultado.
flatmap	Aplica una función a cada elemento del RDD y retorna un RDD de los contenidos de los iteradores retornados. Usados para extraer palabras.
filter	Retorna un RDD que contiene los elementos que pasaron la condición enviada para filtrar.
distinct	Remueve duplicados.
sample(withReplacement, fraction, [seed])	Toma una muestra de un RDD.
union	Produce un RDD que contiene los elementos de dos RDD.
intersection	Retorna un RDD que contiene solo los elementos que coinciden en dos RDD.
subtract	Remueve los elementos contenidos en un RDD. Por ejemplo: train data.
cartesian	Producto cartesiano con otro RDD.

Figura 3. Transformaciones RDDs

Función	Descripción
collect()	Retorna todos los elementos de un RDD.
count()	Retorna el número de elementos en un RDD.
take(num)	Retorna el num de elementos del RDD.
top(num)	Retorna los top num elementos del RDD.
takeOrdered(num)(ordering)	Retorna num elementos ordenados según ordering.
takeSample(withReplacement, num, [seed])	Retorna num elementos aleatorios.
reduce(func)	Combina los elementos del RDD juntos en paralelo.
fold(zero)(func)	Igual que reduce pero con el zero value indicado.
aggregate(zeroValue)(seqOp, combOp)	Similar a reduce pero usado para retornar un tipo diferente.
foreach(func)	Aplica func a cada elemento del RDD.

Figura 4. Acciones RDDs

Problema	Métodos para resolver con MLlib
Clasificación binaria	Linear SVMs, logistic regression, decision trees, naive Bayes.
Clasificación multiclase	decision trees, naive Bayes.
Regresión	Linear least squares, Lasso, ridge regression, decision trees.

Figura 5. Métodos de clasificación en MLlib

2.4 Random Forest

Random forest es un algoritmo de predicción que combina árboles de decisión mediante la técnica *Bagging*, donde cada árbol de decisión es construido a partir de atributos aleatorios y muestras aleatorias.

La técnica *Bagging* consiste en la creación de diferentes modelos de predicción usando muestras aleatorias y combinando los resultados de estos modelos.

En *Random forest* se crean varios árboles de decisión con atributos aleatorios y muestras aleatorias sin repetirse estas últimas muestras entre los árboles creados. Seguidamente se realiza la predicción de la clase en cada árbol de decisión. Una vez se tienen todas las predicciones se selecciona como resultado la clase mayoritaria entre todas las clases predichas en los árboles.

2.5 Métricas

Las métricas utilizadas en la clasificación son las siguientes:

- *Matriz de confusión*: es una forma de visualización de los resultados de predicción en la cual las filas pueden representar los valores reales y las columnas los predichos o viceversa. Compara los valores reales con los de predicción por lo que es una herramienta útil para entender fácilmente los resultados obtenidos.

- *Precisión*: muestra de todos los valores predichos cuantos hemos acertado de manera que conforme más nos acerquemos al valor *1.0* mejor predecirá nuestro algoritmo.

- *Recall:* muestra de todos los valores reales cuantos somos capaces de acertar. Cuanto más se aproxime a *1.0* este valor tendremos una mayor fiabilidad de la predicción de la etiqueta en concreto de la que estamos calculando el *recall*.

2.6 Predicciones en Moocs

Hoy en día se han realizado algunos estudios acerca de los Moocs teniendo en cuenta variables como el tiempo que ha estado un estudiante viendo un vídeo, el número de eventos realizados por un estudiante en un curso, etc.

Con estos estudios se ha intentado predecir el abandono en este tipo de cursos y la posibilidad de obtener un certificado en función de la nota final del estudiante.

2.6.1 Predicción de abandono en Moocs

Tres científicos del MIT -Sebastian Boyer, Kalyan Veeramachanen y Una-May O'Reilly- defienden que una de las claves está en saber, antes de que acabe el curso, quién lo ha abandonado para tratar de "repescarle". En su artículo *Likely to stop? Predicting Stopout in Massive Open Online Course* [16] analizan cómo lograrlo mediante una técnica que denominan *transfer learning*.

Se trata de construir modelos iniciales basados en conjuntos de datos previos, para luego aplicar dichos modelos a conjuntos de datos en tiempo real. Esto es lo que ellos llaman *transfer learning*, que consiste en hacer corresponder los datos de un estudiante de los modelos previos con los datos recogidos en tiempo real para predecir su comportamiento. Su modelo usa cerca de 20 variables para monitorizar el comportamiento del estudiante en la plataforma. Extraídas de manera semanal, estas variables incluyen desde datos simples como la suma de tiempo que un estudiante pasa en la plataforma a la semana o el número de problemas resueltos cada siete días, hasta datos más complejos como la anticipación a la fecha de entrega.

Su sistema les ha permitido demostrar que el abandono de los cursos es un problema que se puede medir con una semana de antelación y un 70% de acierto.

Su hipótesis es que este alto abandono se debe a dos factores. Uno es que no todo el mundo que se registra en un curso tiene intención de acabarlo. Otros pueden desear terminarlo pero se encuentran con problemas en la vida real que lo impiden.

2.6.2 Predicción de obtención de certificado en Moocs

Artículos como *Motivation Classification and Grade Prediction for MOOCs Learners* [17] demuestran que se han realizado estudios con la intención de predecir si un estudiante va a obtener el certificado del curso. En este estudio se tienen en cuenta la media de los eventos totales del estudiante, la media de las visualizaciones de videos, la media de post's de los estudiantes y el número de días activos. Estos datos se comparan con la calificación final obtenida y se clasifican mediante máquinas de soporte vectorial (SVM) obteniendo un porcentaje de acierto del 90-95%.

En los siguientes capítulos se describen los análisis realizados en este trabajo con el objetivo de predecir la nota final de un estudiante a partir de su rendimiento semanal. Este análisis será de gran utilidad ya que permitirá al profesor intervenir lo antes posible y proporcionar una ayuda extra al estudiante que lo necesite.

3 Desarrollo

En este apartado se describe cómo se han aplicado, al análisis de logs de MOOCs, las diversas técnicas anteriormente explicadas. El trabajo consiste en dos partes diferenciadas: la transformación de los logs en un formato más fácil de procesar por posibles aplicaciones que vayan a trabajar sobre esta información y un ejemplo de aplicación de clasificación automática que parte de esa información simplificada. El objetivo de la primera parte es eliminar información redundante, reducir su volumen, y facilitar el procesado de los logs para futuras aplicaciones que tengan que trabajar sobre los logs. La segunda parte implementa una aplicación de clasificación sobre los datos procesados. Para ello previamente se han analizado Pyspark y el formato de logs en edX.

3.1 Pyspark

Pyspark es un intérprete interactivo de Python con Apache Spark. Para poder utilizarlo en este proyecto se ha instalado la distribución Anaconda de Python que integra un gran número de paquetes de código abierto de Python. Entre estos paquetes se encuentra el intérprete interactivo Ipython que ejecutaremos mediante Jupyter Notebook, un entorno interactivo web de ejecución de código.

3.2 Análisis de los logs de edX

La plataforma edX genera ficheros de log muy completos con la información de las interacciones de los estudiantes con la plataforma. Todos los eventos que desencadena cada estudiante se almacenan en estos logs en formato JSON. Cada uno de estos eventos se guarda en una línea de los logs. Un ejemplo de un evento de tipo ver vídeo (play_video) del log se muestra en la Figura 6.


```
{"username": "", "event_type": "play_video", "ip": "81.47.192.242", "agent": "Mozilla/5.0 (Windows NT 6.1;
WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/40.0.2214.111 Safari/537.36", "host": "courses.edx.
org", "session": "0dc7d4723ae1d87341c342764c0ddeb2", "referer": "https://courses.edx.org/courses/UAMx/Andro
id301x/1T2015/courseware/a1d3d5d385ba416e91e1a5c7e3692495/c7289b3a2c6542b4bd6564e14dfbfd6b/",
"accept_language": "", "event": "{\"id\":\"i4x-UAMx-Android301x-video-
cd5a68e3063e41f1b48fa37129238b10\",\"currentTime\":160,\"code\":\"6gUDHRyOAHY\"}", "event_source":
"browser", "context": {"user_id": 6508988, "org_id": "UAMx", "course_id": "UAMx/Android301x/1T2015",
"path": "/event"}, "time": "2015-03-11T09:21:31.954945+00:00", "page": "https://courses.edx.org/courses/UAM
x/Android301x/1T2015/courseware/a1d3d5d385ba416e91e1a5c7e3692495/c7289b3a2c6542b4bd6564e14dfbfd6b/"}
```


Figura 6. Ejemplo de evento de video en log.

Este evento posee diferentes etiquetas [10]. Algunas de las más relevantes son las siguientes:

- User_id: contiene el id del estudiante y se encuentra dentro del JSON definido tras la etiqueta “context” del JSON principal.
- Event_type: se trata de la etiqueta que identifica el tipo de evento, en este caso play_video.
- Time: esta etiqueta contiene el timestamp del evento.
- CurrentTime: momento del video en el que se produce el evento de play.

Podemos comprobar que no toda la información de este evento es necesaria, por ejemplo la etiqueta *agent* que nos describe el navegador utilizado por el estudiante, o la dirección ip desde la que accede no nos proporcionan información útil. Por lo tanto se debe hacer un procesado previo que nos permita seleccionar la información relevante de este tipo de evento.

Además, los eventos no se encuentran ordenados por el *timestamp* del evento sino por el orden en que se almacenan en el fichero por lo que será necesario ordenarlos cuando se extraiga la información de los estudiantes.

También dentro del log hay eventos como el de la Figura 7 que no nos proporcionan información útil ya que este tipo de evento no se encuentra descrito en la documentación proporcionada por edX y no podemos saber qué significa. Por lo tanto debemos ignorar este tipo de eventos.

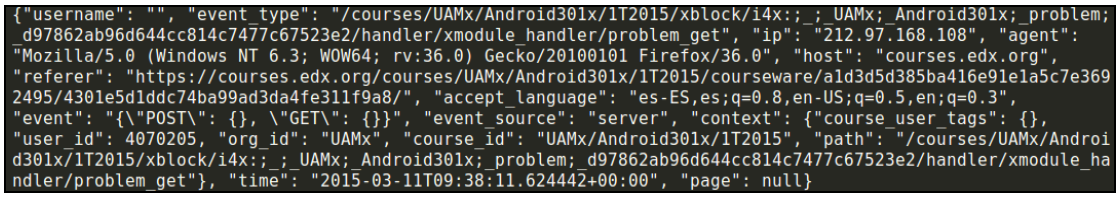


Figura 7. Ejemplo evento no útil.

Primeramente se realizó un estudio de los eventos para identificar los que podrían proporcionarnos información útil sobre las interacciones de los estudiantes con la plataforma. A continuación se describen los eventos que hemos considerado de mayor interés:

- **Vídeos:** estos son los eventos que se desencadenan cuando el estudiante interacciona con los vídeos del curso. Dentro de estos eventos hemos considerado útiles los de pausar un vídeo, darle a play, cargar un vídeo, desplazarse en la línea de tiempo de un vídeo y aumentar o disminuir su velocidad. En todos los casos queda registrado el id del vídeo sobre el que el estudiante realiza el evento, al igual que el tiempo del vídeo en el que lo hace. En el caso de tratarse de un desplazamiento en el vídeo se registra la posición del vídeo inicial y final en segundos. Si se trata de un evento de modificación de la velocidad de vídeo, guardamos también la velocidad nueva y antigua.

- **Foro:** estos son los eventos que se desencadenan cuando el estudiante interacciona con el foro del curso. Dentro de estos eventos hemos considerado útiles los siguientes tipos: creación de hilos, respuestas en hilos, comentarios a respuestas y búsquedas en el foro. Para los tres primeros guardamos la siguiente información: el texto del mensaje que ha escrito el estudiante, si el estudiante sigue el mensaje, el id del hilo, el id de la respuesta en caso de que sea una respuesta y el id del foro del curso. En el caso de búsquedas en el foro nos quedaremos con el texto de la consulta.

- **Documentos:** estos son los eventos que se desencadenan cuando el estudiante abre un documento en pdf del curso. De este evento únicamente recogemos el id del capítulo.

- **Problemas:** estos son los eventos que se desencadenan cuando el estudiante realiza un problema del curso. Dentro de estos eventos nos quedamos con el id del problema y el número de intentos que el estudiante hace para resolver el problema. Además, dado que cada problema puede estar formado por subproblemas que llamaremos ejercicios, también guardaremos el número de ejercicios que posee cada problema y, para cada ejercicio, guardamos las respuestas y si han sido correctas o incorrectas.

- **Autoevaluación:** estos son los eventos que se desencadenan cuando el estudiante realiza una autoevaluación en la que se evalúa él mismo respecto a una actividad o proyecto que ha realizado. En este tipo de eventos quedan registrados el id de la autoevaluación y las partes que posee la autoevaluación. Para cada parte, se registra también la calificación máxima posible en cada una de las mismas y la nota que se pone el estudiante junto con un feedback.

Adicionalmente para todos los eventos guardamos el id de estudiante que lo desencadena y el *timestamp* en el cual se produjo el evento.

3.3 Procesado de logs

Una vez identificados los eventos de interés en los logs, se procesan los ficheros de logs para extraerlos. Este paso se realiza mediante un programa en Pyspark. El programa se encarga de reducir el volumen de los logs, eliminando eventos que no hemos considerado relevantes en este estudio, y reduciendo la información sobre los que sí interesan tal como se ha descrito en la sección anterior. El objetivo es extraer la información que hemos considerado relevante sobre cada estudiante para ponerla en un formato más fácil de procesar. Cada estudiante quedará representado básicamente mediante su id de estudiante y una lista de eventos realizados por ese estudiante y ordenados cronológicamente. Al final los JSON de cada estudiante serán guardados en un fichero, en el que cada línea del fichero tendrá el JSON de un estudiante en el formato que hemos indicado anteriormente.

Para el procesado de datos se han seguido los siguientes pasos: en primer lugar se han cargado todos los datos de los logs. Tras ello se han filtrado los datos para separar los eventos de cada tipo: vídeos, documentos, problemas, foro y autoevaluaciones. De cada evento se ha sacado la información útil para cada estudiante en formato JSON. Una vez hecho esto se han unido todos los eventos de cada estudiante de manera que queden ordenados cronológicamente y, finalmente, se ha volcado toda la información extraída a un fichero.

3.3.1 Procesado de eventos de vídeo

La información a extraer de los eventos de vídeo incluye todo lo relacionado con los tiempos de los eventos así como el identificador del vídeo. En el caso de los vídeos primero vamos a reducir el id del vídeo a una información más interpretable. Sustituiremos el id original por un id que estará relacionado con el orden cronológico del video. Esto lo haremos mediante la creación de un diccionario que contenga la relación entre el id del vídeo real y un id proporcionado por nosotros.

Para crear este diccionario se han recorrido todos los datos para extraer todos los identificadores originales y asignarles uno nuevo. Por ejemplo, hemos asignado el id de vídeo "*31*" al vídeo con id real:

"i4x-UAMx-Android301x-video-500e185ce4884abfba40c95c2b9c615a"

Una vez creado este diccionario, realizamos el filtro para seleccionar los siguientes eventos de vídeo: "*load_video", "play_video", "seek_video", "stop_video", "pause_video", "speed_change_video".* Para cada uno de estos eventos se extrae la información relevante y se sustituye el id del video por el asignado en el diccionario

Una vez tenemos esta información sobre los eventos de vídeo de cada estudiante en un RDD los agrupamos por la clave (id de estudiante) mediante la operación de Spark **`reduceByKey`**. Tras ello se obtiene una colección que contiene, para cada estudiante, una lista con las acciones que ha realizado sobre los vídeos.

Finalmente, se ordenan temporalmente los eventos de vídeo de cada estudiante y seguidamente eliminamos los eventos de vídeos incoherentes o duplicados en un periodo de tiempo cercano. Esto es debido a que el servidor de edX no registra correctamente algunos eventos en el log. Por ejemplo, se puede observar en el log que hay eventos de *play_video* repetidos en un periodo de tiempo muy cercano y esto no es posible. También se observan algunos eventos de desplazamiento de video incoherentes. Una vez realizadas estas operaciones tendremos un RDD que contiene, para cada estudiante, un id de estudiante y una lista de eventos de vídeo ordenados cronológicamente y con coherencia entre sus interacciones.

3.3.2 Foro

Para los eventos del foro buscaremos aquellos registros del log cuyo tipo de evento contenga las siguientes cadenas: *"edx.forum.thread.created"*, "*edx.forum.response.created", "edx.forum.comment.created", "edx.forum.searched"* que corresponden con eventos de creación de hilos, respuesta a hilos, comentarios a respuestas y búsquedas en foro, respectivamente.

Nos quedaremos con la información mencionada en el apartado 3.1. Se debe guardar el cuerpo del mensaje del foro o, en caso de tratarse de una búsqueda en el foro, el texto de la consulta. Esto puede ocasionar errores a la hora de que se procese el JSON debido a que en el contenido se encuentren caracteres extraños. Por lo tanto, habrá que identificar y eliminar dichos caracteres que podrían no permitirnos generar el nuevo evento del estudiante en formato JSON.

Finalmente guardamos esta información nuevamente en otro RDD con los ids de estudiante y sus respectivas acciones en el foro.

3.3.3 Documentos

Estos eventos vienen determinados en el log por el tipo de evento "*textbook.pdf.chapter.navigated*" que indican que el estudiante ha estado navegando por un documento identificado mediante un id en el JSON del evento.

Filtramos los eventos y nos quedamos solo con este tipo de evento que nos indica el capítulo que ha abierto el estudiante Finalmente, realizando unas transformaciones similares a las anteriores, obtendremos un RDD de tipo clave-valor formado por los ids de estudiante y los capítulos que ha abierto cada uno de ellos.

3.3.4 Problemas

En los problemas, al igual que en los vídeos, necesitamos reducir su id a un id propio que nos permita identificar con mayor facilidad el problema. Pero en este caso no poseemos un JSON que nos relacione el id de problema real con el id propio y que esté ordenado cronológicamente. Es decir, que el id de problema 1 corresponda al primer problema de la primera semana, el 2 al segundo, etc. Por lo tanto, se ha tenido que crear un pequeño programa que recorriese todos los logs del curso buscando los ids de problemas y añadiéndolos a un RDDs formado por el id de problema y la primera fecha en la que algún estudiante interacciona con el problema. Una vez hecho esto tenemos que ordenar por tiempo la aparición de cada id de problema. Esto lo hacemos mediante la función de Spark **`sortBy`**. Después recogemos los resultados y guardamos cada id de problema por orden, el primer id de problema tendrá el id propio 1, el segundo el 2, y así sucesivamente. De esta manera un id de problemas como por ejemplo:

"*i4x://UAMx/Android301x/problem/768d466a8b6b4bcaaffc12e99bfd68fb*"

pasa a tener el valor 1 en nuestro nuevo diccionario, que exportaremos a formato JSON.

Una vez hecho esto extraemos del log los eventos que tienen el tipo de evento "problem_check" y como evento fuente "server". A partir de ahí continuamos con la recolección de la información útil definida en el apartado 3.1.

En el caso de que haya caracteres extraños en las respuestas de los estudiantes se intenta solventar eliminando los caracteres que causan dichos errores durante el proceso de conversión a formato JSON. Si no es posible ese evento se guardará como un evento de error que será ignorado en las demás etapas. Una vez recogida la información tenemos de nuevo un RDD con los ids de estudiantes y su información referente a los problemas realizados.

3.3.5 Procesado de eventos de autoevaluación

En los eventos de autoevaluación realizamos el filtro de eventos que contienen la cadena "*openassessmentblock.self_assess"* como tipo de evento ya que estos JSON son los que contienen la información que hemos mencionado en el apartado 3.1.

Una vez recogida la información de los eventos de autoevaluación, la almacenamos en un RDD formado por los ids de estudiante y sus acciones en formato JSON. El formato utilizado es equivalente al de salida de los videos, esto es, id de estudiante y lista de eventos ordenados cronológicamente.

3.3.6 Unión de RDDs

Ahora tenemos todos los eventos separados en diferentes colecciones (RDDs) de tipo clave-valor donde la clave es el id de estudiante. A continuación se combinan todos los RDD en uno solo con la operación **`unión`** y se reducen mediante la operación **`reduceBykey`** de Spark. Esta operación los agrupa por la clave, en nuestro caso por el id de estudiante, de manera que tendremos un único RDD de tipo clave-valor formado por ids de estudiantes y todos los eventos que ha realizado cada estudiante a lo largo del curso.

Ahora bien, estos eventos no están ordenados cronológicamente sino por tipo de evento. Por ello es necesario ordenarlos cronológicamente antes de volcarlos a un fichero para su posterior utilización.

La información de salida del programa en formato JSON para la tupla de un estudiante es la que se muestra en la Figura 8. Como se puede apreciar en la figura, en este caso lo primero que hizo el estudiante fue cargar un vídeo, luego pulsar play, etc. Si siguiésemos avanzando en la lista de eventos veríamos más eventos del mismo estudiante: problemas, lecturas de documentos, etc.

3.4 Aplicación de clasificación automática de problemas

Para poder analizar el rendimiento de los estudiantes, se utiliza la información almacenada sobre los problemas, ya que se considera que un estudiante que realiza los problemas del curso tiene una mayor probabilidad de no abandonar la asignatura y de obtener mejor nota que otro estudiante que no los haga. Por lo tanto, los atributos referentes a los problemas realizados por los estudiantes pueden proporcionar información para una correcta predicción de variables de interés, como la nota del examen final o el resultado de un problema a partir de los anteriores.

El problema concreto que se afronta es la predicción de las respuestas de los estudiantes a un ejercicio a partir de las respuestas a los ejercicios anteriores realizados por él. Este análisis se va a estructurar en una secuencia de problemas de clasificación. Primero, se predice la respuesta que va a dar el estudiante al segundo ejercicio usando solo el primer ejercicio como atributo de clasificación y se calcula el error de los estudiantes en el conjunto de datos de test. Después, se calcula el error del tercer ejercicio con el primer y segundo ejercicios como atributos de clasificación, y así sucesivamente hasta completar el cálculo del error de todos los ejercicios.

El resto de eventos o atributos no serán utilizados con fines de predicción en este trabajo, pero se guardan en el fichero mencionado anteriormente con un formato fácilmente utilizable, para poder realizar predicciones de variables de interés en futuros trabajos o generar estadísticas, como por ejemplo la hora más frecuente para realizar problemas, en que época del curso se ven más vídeos, cuales son los vídeos que más ven los estudiantes, análisis de texto en los comentarios del foro, etc.

Figura 8. Eventos de video procesados

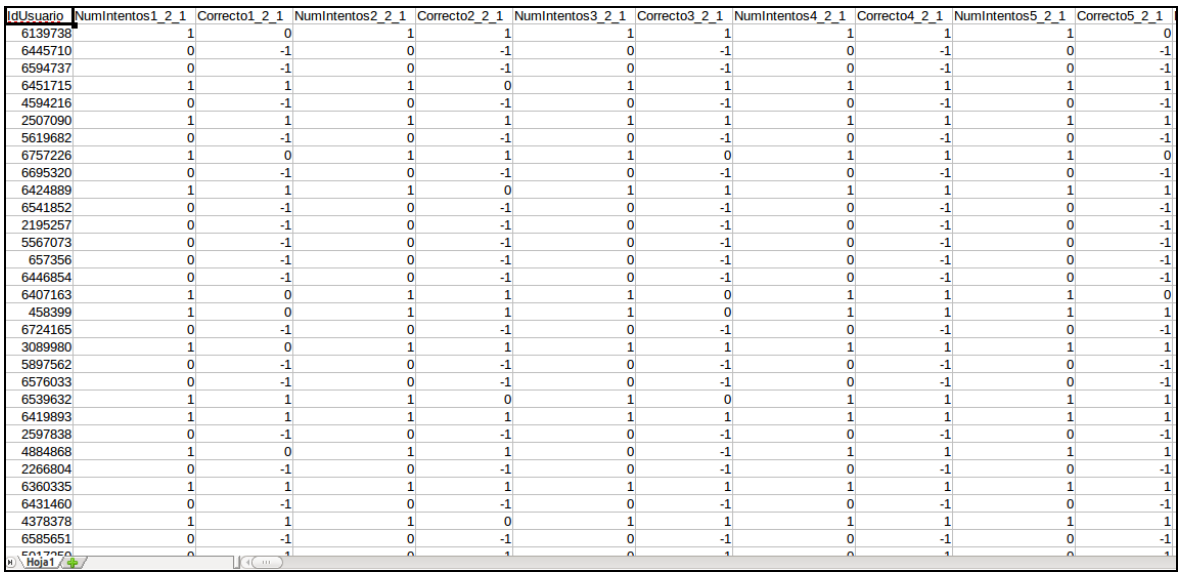
3.4.1 Formato de clasificación

Para extraer la información relevante para conseguir nuestro objetivo, se han seguido los siguientes pasos. En primer lugar, se ha convertido la información disponible sobre los problemas en una información utilizable por los clasificadores. Para ello recordemos que los problemas están compuestos de subejercicios por lo que se debe realizar un mapa que asigne a cada id de problema el número de subejercicios que posee. Una vez hecho esto recorremos la lista de eventos de cada estudiante y nos quedamos solo con aquellos eventos relacionados con problemas.

Con estos eventos, para cada problema guardamos en un RDD el número de intentos de dicho problema y, para cada ejercicio, si es correcto o no. De tal manera que asignamos *"0"* en caso de fallo o *"1"* en caso de acierto. Si el estudiante no ha realizado el ejercicio, el número de intentos del mismo es *"0"* y su calificación *"-1"*. Una vez hecho esto tenemos un RDD que contiene, para cada estudiante, una lista de los problemas que ha realizado con su respectiva valoración.

Después recogemos los datos del RDD con la acción **`collect`** y lo imprimimos en un fichero de manera que, como primer atributo, tengamos el id del estudiante y seguidamente

los números de intentos de los problemas junto con sus correspondientes aciertos o fallos. Todo ello ordenado por el id de problema que hemos generado. En la Figura 9 se puede ver un extracto de la tabla de salida obtenida.



IdUsuario	NumIntentos1_2_1	Correcto1_2_1	NumIntentos2_2_1	Correcto2_2_1	NumIntentos3_2_1	Correcto3_2_1	NumIntentos4_2_1	Correcto4_2_1	NumIntentos5_2_1	Correcto5_2_1
6139738	1	0	1	1	1	1	1	1	1	0
6445710	0	-1	0	-1	0	-1	0	-1	0	-1
6594737	0	-1	0	-1	0	-1	0	-1	0	-1
6451715	1	1	1	0	1	1	1	1	1	1
4594216	0	-1	0	-1	0	-1	0	-1	0	-1
2507090	1	1	1	1	1	1	1	1	1	1
5619682	0	-1	0	-1	0	-1	0	-1	0	-1
6757226	1	0	1	1	1	0	1	1	1	0
6695320	0	-1	0	-1	0	-1	0	-1	0	-1
6424889	1	1	1	0	1	1	1	1	1	1
6541852	0	-1	0	-1	0	-1	0	-1	0	-1
2195257	0	-1	0	-1	0	-1	0	-1	0	-1
5567073	0	-1	0	-1	0	-1	0	-1	0	-1
657356	0	-1	0	-1	0	-1	0	-1	0	-1
6446854	0	-1	0	-1	0	-1	0	-1	0	-1
6407163	1	0	1	1	1	0	1	1	1	0
458399	1	0	1	1	1	0	1	1	1	1
6724165	0	-1	0	-1	0	-1	0	-1	0	-1
3089980	1	0	1	1	1	1	1	1	1	1
5897562	0	-1	0	-1	0	-1	0	-1	0	-1
6576033	0	-1	0	-1	0	-1	0	-1	0	-1
6539632	1	1	1	0	1	0	1	1	1	1
6419893	1	1	1	1	1	1	1	1	1	1
2597838	0	-1	0	-1	0	-1	0	-1	0	-1
4884868	1	0	1	1	0	-1	1	1	1	1
2266804	0	-1	0	-1	0	-1	0	-1	0	-1
6360335	1	1	1	1	1	1	1	1	1	1
6431460	0	-1	0	-1	0	-1	0	-1	0	-1
4378378	1	1	1	0	1	1	1	1	1	1
6585651	0	-1	0	-1	0	-1	0	-1	0	-1

Figura 9. Ejemplo formato clasificatorio

3.4.2 Elección de librería de aprendizaje automático

Una vez tenemos nuestro fichero en formato de clasificación debemos elegir la librería que usaremos para llevar a cabo las predicciones mediante *Random forest.* Hemos contemplado dos posibilidades respecto a la librerías a utilizar para la predicción de datos: MLlib y Sklearn. Al final la librería por la que nos hemos decantado es MLlib debido a las siguientes razones:

- **Rapidez:** la librería MLlib viene implementada sobre Spark por lo que la ejecución se distribuye entre los núcleos del ordenador mejorando así la velocidad de ejecución a la hora de entrenar el modelo y de realizar las clasificaciones.

- **Atributos categóricos:** como hemos mencionado anteriormente nuestra intención es utilizar *Random Forest* como algoritmo de clasificación debido a su precisión. Sin embargo, la implementación de Random forest de la librería sklearn no posee la capacidad de añadir atributos categóricos a dicho algoritmo, mientras que MLlib sí que dispone de esa funcionalidad.

3.4.3 Fichero con formato MLlib

El primer paso para poder utilizar la librería MLlib es transformar nuestro fichero en un formato que sea capaz leer dicha librería mediante la función loadLibSVMFile de la clase MLUtils. Para ello utilizaremos la librería *pandas* de Python, una librería destinada al análisis de datos que proporciona unas estructuras de datos flexibles y que permite trabajar con ellas de forma muy eficiente. En concreto emplearemos dos de las estructuras que nos proporciona pandas:

- **Series:** son arrays unidimensionales con indexación (arrays con índice o etiquetados), similares a los diccionarios. Pueden generarse a partir de diccionarios o de listas.

- **DataFrame:** Son estructuras de datos similares a las tablas de bases de datos relacionales como SQL.

Mediante estas herramientas creamos un fichero con el formato necesario para cargarlo en MLlib sustituyendo los *"-1"* que indican que el ejercicio no se ha realizado, por un *"2"* que indicará lo mismo, ya que *Random Forest* utiliza enteros positivos como formato para codificar atributos categóricos.

Una vez realizadas estas transformaciones los datos quedan como se muestra en la Figura 10. Para cada fila, el primer valor nos indica la clase, es decir si el problema se ha realizado correctamente, incorrectamente o si no se ha realizado y los demás atributos nos indican los resultados en los problemas previos al actual.

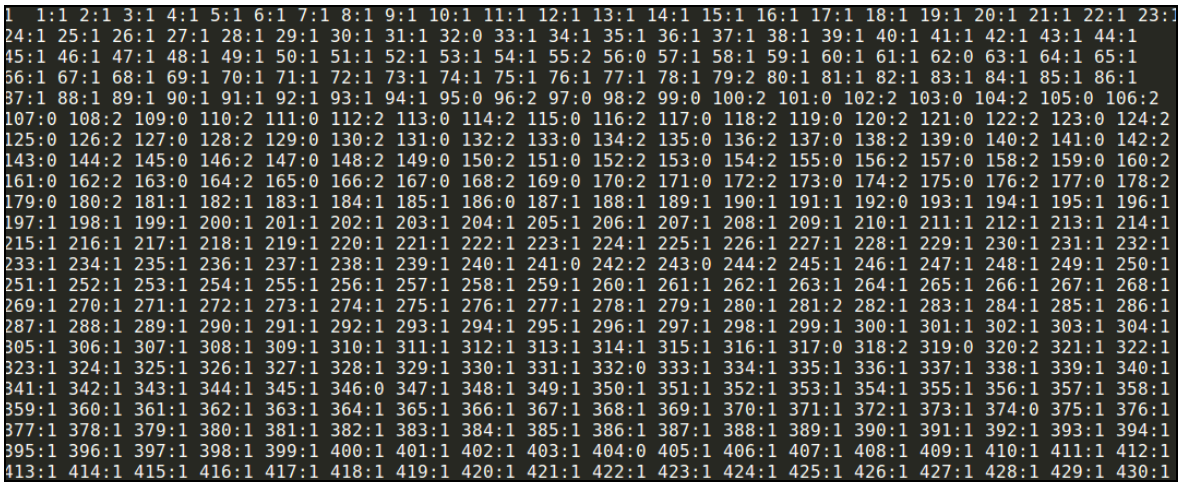


Figura 10 Ejemplo formato para procesar por MLlib

3.4.4 Predicción de ejercicios

Se deben predecir todos los ejercicios de los problemas a partir de ficheros con el mismo formato que el anterior. El proceso consiste en la creación de numerosos ficheros en los que se va variando la variable a predecir y la aplicación a cada uno de ellos del algoritmo de clasificación *Random Forest* obteniendo así los diferentes porcentajes de error para la predicción de cada ejercicio. Hemos utilizado *Random Forest* con 300 árboles, 4 como grado de profundidad, impureza "*Gini*" y maxBins de "32" que hace referencia al número máximo de ramas que se puede crear por nodo del árbol.

Además, gracias a la clase MulticlassMetrics de MLlib podemos obtener métricas como recall, precisión, matriz de confusión, falsos positivos, etc. Para este problema, sin embargo, hemos utilizado la métrica de *error* que cuenta el número de veces que son distintas la etiqueta del valor real y el predicho y lo divide entre el total de datos de test.

$$\frac{\sum_{test}[real != predicha]}{numero\ datos\ test}$$

Esta fórmula la hemos implementado con la ayuda de la tecnología **`map`** de Spark aplicada a un RDD que contiene la etiqueta de la clase real y la clase predicha por el clasificador.

3.5 Predicción de la nota del examen

El objetivo de este proceso consiste en intentar predecir la nota del examen final a partir de los resultados obtenidos por cada estudiante en los ejercicios. Para estimar la nota del examen final de un estudiante se tiene que crear nuevo fichero de datos que contenga la nota del examen final y todos los ejercicios realizados por ese estudiante antes del examen final, es decir, los primeros 138 problemas.

Una vez hemos creado dicho fichero utilizaremos *Random Forest* con regresión para predecir la nota. Se hacen una serie de transformaciones con la finalidad de poder predecir la nota de los estudiantes por semana además de un análisis de los estudiantes predichos como suspensos con la intención de ayudarles lo antes posible.

3.5.1 Creación del fichero para la predicción de la nota del examen

La primera tarea es eliminar del fichero CSV que contiene todas las notas de todo de los estudiantes (actividades, proyectos...) aquellos estudiantes cuya nota del examen es *"0"* ya que estos estudiantes han abandonado el curso y a nosotros nos interesan aquellos estudiantes que han completado el curso entero.

La segunda tarea consiste en transformar el fichero resultante en otro que relacione la información de los problemas de ese estudiante con la nota obtenida en el examen, eliminando los problemas correspondientes a la parte del examen final. Para ello utilizamos la herramienta *pandas*, ya mencionada anteriormente y que nos sirve para el manejo de datos estructurados.

Correcto135_2_1	NumIntentos136_2_1	Correcto136_2_1	NumIntentos137_2_1	Correcto137_2_1	NumIntentos138_2_1	Correcto138_2_1	Final
-1.0	2.0	0.0	1.0	1.0	1.0	0.0	0.6101694915
1.0	0.0	-1.0	1.0	1.0	1.0	1.0	0.7796610169
1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.8644067797
1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.7966101695
1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.9491525424
1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.9322033898
1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.7796610169
1.0	2.0	0.0	1.0	1.0	1.0	1.0	0.813559322
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.813559322
1.0	2.0	1.0	1.0	1.0	1.0	1.0	0.813559322
1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.9661016949
1.0	2.0	0.0	1.0	1.0	1.0	1.0	0.7118644068
1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.813559322
1.0	2.0	0.0	1.0	1.0	1.0	1.0	0.7627118644
1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.8813559322
1.0	1.0	1.0	1.0	0.0	1.0	1.0	0.7457627119
0.0	0.0	-1.0	1.0	1.0	1.0	0.0	0.593220339
1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.7966101695
1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.9322033898
1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.9491525424
1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.8813559322
0.0	2.0	0.0	1.0	0.0	1.0	0.0	0.3559322034
1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.8644067797
1.0	2.0	0.0	1.0	1.0	1.0	0.0	0.5423728814
0.0	1.0	1.0	1.0	1.0	1.0	1.0	0.8983050847

Figura 11. Fichero con problemas de estudiante y nota de examen

De esta manera obtenemos un fichero CSV como el de la Figura 11. En la primera columna tenemos el id de estudiante y, a continuación, los resultados de todos los ejercicios que ha realizado durante el curso junto con el número de intentos para cada uno. En último lugar se encuentra la nota del examen final para cada estudiante.

3.5.1 Predicción de la nota del estudiante

Una vez disponemos de ese fichero utilizamos de nuevo *Random Forest*, para predecir la nota y, en este caso, necesitamos utilizar regresión para obtener la diferencia entre la nota real y la predicha.

En este caso para el cálculo de la precisión de la predicción hemos utilizado la medida de *error cuadrático medio.* El *error cuadrático medio* (ECM) de un estimador mide el promedio de los errores al cuadrado, esto es

$$\frac{\sum_{test}(real - predicha)^2}{Numero\ datos\ test}$$

También se ha utilizado la métrica del *error absoluto* que mide la diferencia en valor absoluto entre la predicción y el valor real.

$$\frac{\sum_{test}|real - predicha|}{Numero\ datos\ test}$$

Hemos guardado en un fichero la nota real junto con su predicción con la finalidad de poder realizar gráficas comparando ambos valores.

3.5.2 Predicción incremental de la nota del examen

Se pretende ver la evolución del error de predicción a lo largo de las semanas. Para ello hay que calcular el error absoluto cada semana, comparar los errores absolutos en cada una de ellas e intentar que el error conforme se vaya acercando el examen final disminuya ya que disponemos de más ejercicios que podemos utilizar como atributos en la clasificación.

Se ha realizado un programa que divide nuestros datos de problemas en función de la semana a la que pertenecen y se generan así ficheros para las semanas que dura el curso. Para esto haremos uso del error absoluto ya que nos dará la diferencia media en puntos entre las notas reales y las predichas.

3.5.3 Ayuda a estudiantes suspensos

Otro de nuestros objetivos es predecir lo antes posible si un estudiante va a suspender el examen final. De esta manera será posible avisarle y ofrecerle material de ayuda con la pretensión de que ese estudiante consiga aprobar dicho examen.

Para ello se debe predecir la nota siguiendo el procedimiento descrito en el apartado anterior. Una vez obtenidas las etiquetas con la nota real y la nota predicha, filtramos para quedarnos con aquellos estudiantes suspensos y vemos cuál es su nota predicha.

Observando la nota predicha se puede establecer un umbral a partir del cual avisar a los estudiantes que van a suspender el examen final. Por ejemplo, si nuestro clasificador predice que la nota del examen final es un “4” se podría avisar a ese estudiante de que tiene una alta probabilidad de suspender el examen final.

Además, como se verá en el capítulo siguiente, se ha desarrollado un programa que calcula la matriz de confusión para la predicción de estudiantes suspensos y aprobados en función de la semana especificada.

4 Integración, pruebas y resultados

En este apartado se explicarán las pruebas realizadas y sus resultados obtenidos. Comenzaremos describiendo el preprocesamiento de los logs que contiene los eventos asociados a cada estudiante. A continuación presentaremos las pruebas y resultados obtenidos de predicción del resultado de los problemas y de la calificación del examen final de los estudiantes.

4.1 Creación del fichero de eventos

Para la realización del procesado y limpieza de datos se ha ido imprimiendo por pantalla el contenido de los RDDs en cada una de las diferentes fases del programa: recolección de eventos de vídeo, de foro, de documentos, de problemas y de autoevaluaciones.

Con la finalidad de hacer pruebas y comprobar el funcionamiento correcto del programa las pruebas se realizaron sobre el fichero de log correspondiente a un día. Una vez comprobado que los análisis funcionan correctamente con los eventos de un día podremos utilizar todos los logs y obtener definitivamente la información relevante.

Para comprobar, por ejemplo, que los eventos de video estaban en un formato correcto, se ha utilizado la función ***take*** de Spark obteniendo resultados como los de la Figura 12.

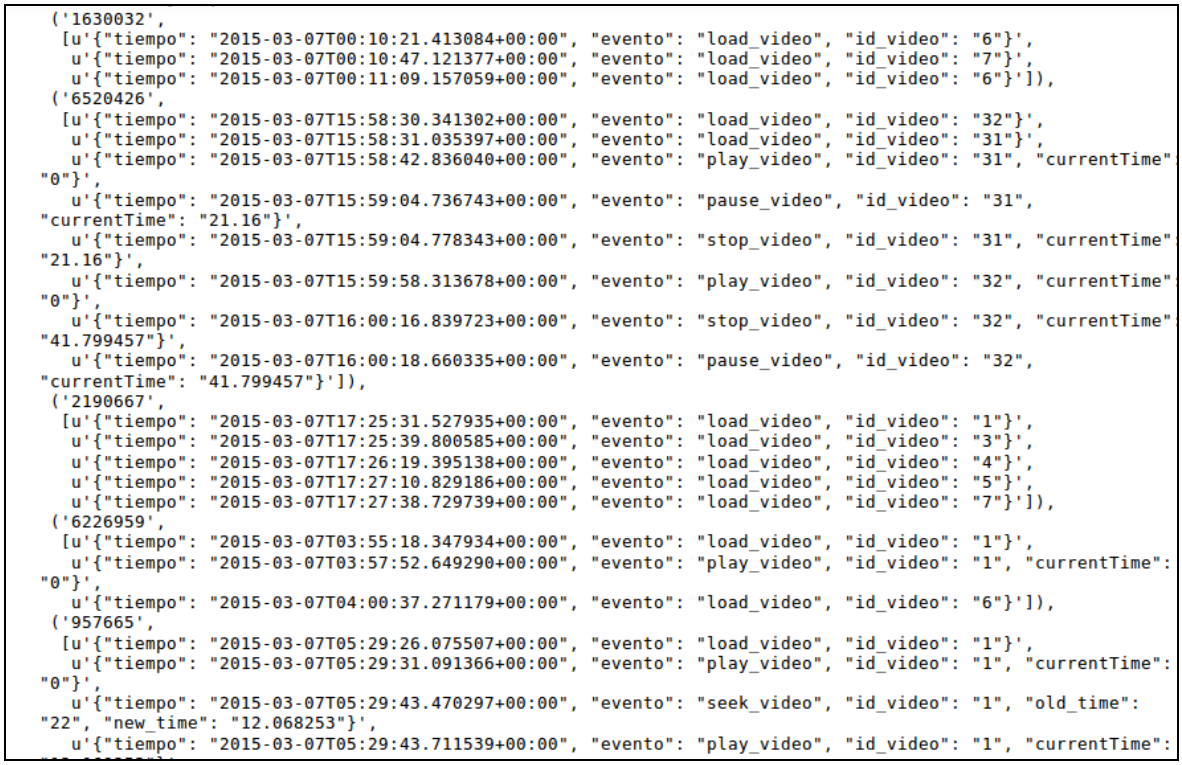


Figura 12. Ejemplo de formato correcto de videos

Se puede observar en la figura que en este RDD tenemos dos partes: el id de estudiante y su lista de eventos de vídeo asociadas. Este RDD es similar para los eventos de documentos, foro, problemas y autoevaluación.

Tras su correspondiente ordenación imprimimos los datos de los estudiantes para ese día del log con la finalidad de comprobar el correcto funcionamiento del programa. El formato de impresión consiste en un id de estudiante seguido de su lista de eventos realizados durante ese día en el log y ordenado cronológicamente. Esta información se vuelca en un fichero en formato JSON para su posterior utilización.

La evolución de los RDDs a lo largo del programa, es decir, las transformaciones que sufren se puede observar en la Figura 13.

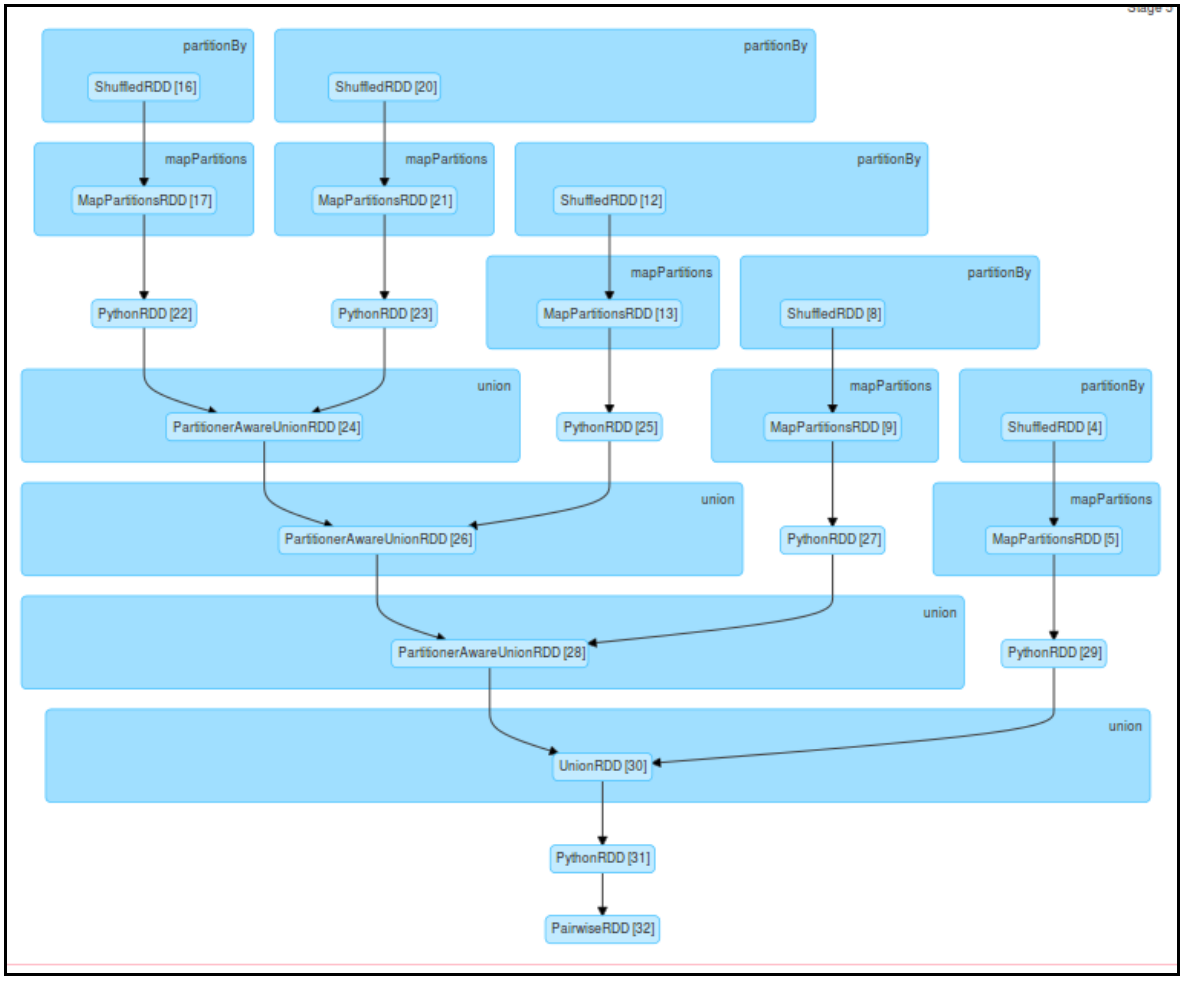


Figura 13. Evolución de los RDDs

En la Figura 13 se puede apreciar cómo cada RDD sufre sus propias transformaciones hasta finalmente unirse en un único RDD que contiene el identificador de estudiante y su respectiva lista de eventos.

4.2 Creación de los ficheros con la información de los problemas

La siguiente tarea ha consistido en extraer del fichero de log la información referente a los problemas. Para ello, el primer paso es crear un diccionario que relacione el identificador del problema con su número de ejercicios (ya que cada problema estaba formado por un

número de ejercicios). La prueba realizada para ver que esto funcionaba correctamente fue la impresión por pantalla del mapa.

Una vez hecho esto procedemos a la extracción de los eventos de los problemas realizados por los estudiantes. Para ello hicimos pruebas comprobando el formato del RDD mediante la función **`take`** que nos proporciona Spark y podíamos así comprobar si estábamos obteniendo correctamente la información de los problemas. En la Figura 14 se muestra una parte del RDD.

```
(u'6139738',
 {1: [{u'1_2_1': [1, 0]}],
  2: [{u'2_2_1': [1, 1]}],
  3: [{u'3_2_1': [1, 1]}],
  4: [{u'4_2_1': [1, 1]}],
  5: [{u'5_2_1': [1, 0]}],
  6: [{u'6_2_1': [1, 0]}],
  7: [{u'7_2_1': [1, 0]}],
  8: [{u'8_2_1': [1, 0]}],
  9: [{u'9_2_1': [1, 0]}],
  10: [{u'10_2_1': [1, 0]}],
  11: [{u'11_2_1': [1, 0]}],
  12: [{u'12_2_1': [1, 0]}],
  13: [{u'13_2_1': [1, 0]}],
  14: [{u'14_2_1': [1, 0]}],
  15: [{u'15_2_1': [1, 1]}],
  16: [{u'16_2_1': [1, 1]}],
  17: [{u'17_2_1': [1, 1]}],
  18: [{u'18_2_1': [1, 1]}],
  19: [{u'19_2_1': [1, 1]}],
  20: [{u'20_2_1': [1, 1]}],
  21: [{u'21_2_1': [1, 0]}],
  22: [{u'22_2_1': [1, 1]}],
  23: [{u'23_2_1': [1, 0]}],
  24: [{u'24_2_1': [1, 1]}],
  25: [{u'25_2_1': [1, 1]}],
  26: [{u'26_2_1': [1, 0]}],
  27: [{'27_2_1': [0, -1]}],
  28: [{'28_2_1': [0, -1]}],
  29: [{u'29_2_1': [1, 0]}],
  30: [{u'30_2_1': [1, 0]}],
  31: [{'31_2_1': [0, -1]}],
  32: [{'32_2_1': [0, -1], '32_3_1': [0, -1], '32_4_1': [0, -1]}],
  33: [{'33_2_1': [0, -1], '33_3_1': [0, -1]}],
  34: [{'34_2_1': [0, -1], '34_3_1': [0, -1], '34_4_1': [0, -1]}],
  35: [{'35_2_1': [0, -1], '35_3_1': [0, -1]}],
  36: [{'36_2_1': [0, -1]}],
  37: [{'37_2_1': [0, -1]}],
```

Figura 14. RDD de un estudiante con el id de problema, id de ejercicio y sus resultados

Se puede apreciar que el RDD posee el id de estudiante junto con una lista de los ids de ejercicios y los resultado de los mismos. Una vez hecho todo esto se vuelca a un fichero en el mismo formato que se ha indicado en el punto 3.3.1 del desarrollo.

4.3 Predicción del resultado obtenido en los ejercicios

En este apartado describiremos los resultados de clasificación que hemos obtenido a partir de la información de los ejercicios y las pruebas realizadas.

La primera clasificación se realiza sobre un fichero que contiene todos los estudiantes matriculados en el curso, incluidos aquellos estudiantes que no han realizado ningún problema en el curso (7000 estudiantes). Esta clasificación se realiza midiendo el error de

predicción de cada uno de los problemas comenzando por el 2 hasta el 196. La validación se ha realizado particionado el conjunto de datos en un conjunto de entrenamiento y otro de test, con un tamaño del 70% y 30%, respectivamente.

Los resultados de la predicción del último ejercicio se muestran en la Tabla 1. Si observamos el error que se muestra en a la hora de la predicción del último ejercicio observaremos que se trata de un error ínfimo siendo este de un 0.5%.

Predicha / ***Real***	***Fallo***	***Acierto***	***No realizado***
Fallo	3	0	0
Acierto	9	131	3
No realizado	0	0	2043
Precisión	0.25	1.0	0.99
Recall	1.0	0.92	1.0
Error problema 196 ejercicio 1 *= 0.05*			

Tabla 1. Información sobre el problema 196 ejercicio 1

Sin embargo, analizando este resultado en detalle vemos que es debido en parte a que 4000 estudiantes no realizan ningún ejercicio en el curso y 6500 no han respondido al último ejercicio. Esto hace que sean fáciles de identificar para el algoritmo de aprendizaje y que para estos estudiantes siempre se prediga que el ejercicio no se ha realizado, lo que ocasiona unos resultados optimistas. Por ello, para obtener un valor que se asemeje más a la realidad debemos eliminar esos estudiantes.

El error obtenido tras eliminar los usuarios mencionados anteriormente es variable en función del ejercicio que queremos predecir. Por ejemplo, para el problema 11 obtenemos la información de la Tabla 2.

Predicha / *Real*	***Fallo***	***Acierto***	***No realizado***
Fallo	280	125	18
Acierto	132	199	6
No realizado	6	2	163
Precisión	0.67	0.61	0.87
Recall	0.66	0.59	0.95
Error problema 11 ejercicio 1 *= 0.34*			

Tabla 2. Información sobre el problema 11 ejercicio 1

Observando la Tabla 2, podemos comprobar que el error es de aproximadamente *0.337* (*33.7%*) es decir fallamos la predicción del ejercicio 11 en un *33.7%* de los casos. En la matriz de confusión podemos ver que el acierto viene dado por la correcta predicción de que el estudiante no va a realizar el ejercicio ya que tenemos una precisión de *0.871*. La subida del error se produce debido a que aún tenemos pocos ejercicios para predecir si va a hacer correctamente el ejercicio el estudiante y la precisión de si lo hace correcto o falla es de *0.61* y *0.66* respectivamente.

Ahora bien si buscamos la predicción de un ejercicio en semanas más avanzadas se obtiene la información mostrada en la Tabla 3.

Predicha / *Real*	***Fallo***	***Acierto***	***No realizado***
Fallo	6	0	5
Acierto	3	104	4
No realizado	1	0	808
Precisión	0.6	1.0	0.99
Recall	0.55	0.94	0.99
Error problema 129 ejercicio 1 *= 0.02*			

Tabla 3. Información sobre el problema 129 ejercicio 1

Podemos comprobar aquí que en la mayoría de los casos, la predicción corresponde a la no realización del ejercicio. Esto quiere decir que han abandonado numerosos estudiantes y que predecimos con aproximadamente un *99%* de probabilidad si el estudiante realizará el ejercicio o no. Esto podría ser útil para determinar qué estudiantes tienen una mayor probabilidad de abandonar.

Predecimos además con completa exactitud el acierto por parte del estudiante en el ejercicio *"129"* con una precisión de *1.0.*

Podemos observar también que muy pocos de los estudiantes que realizan los ejercicios fallan. Por ello se predicen muchos más aciertos que fallos. Aun así, cuando predecimos fallo en el ejercicio, acertamos la mayoría de las veces con una precisión de *0.6.*

Finalmente, el porcentaje de error que tenemos en este ejercicio es del *2%* aproximadamente. Esto es debido a que a esas alturas del curso, de los aproximadamente 3000 estudiantes que empezaron sólo un *20%* continúa. Por eso la mayoría de las predicciones aciertan que los estudiantes no harán el siguiente problema.

Todo lo expuesto anteriormente queda reflejado en la gráfica de la Figura 15.

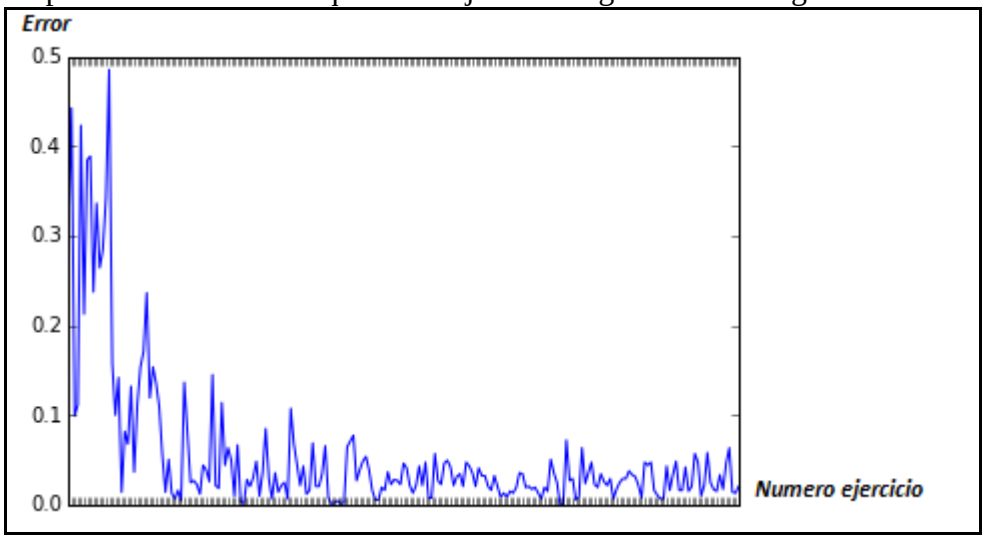


Figura 15. Grafica incluyendo los estudiantes que abandonan el curso

En el eje Y se encuentra representado el error en la predicción y en el eje X el ejercicio al que corresponde el error.

Como podemos observar, a medida que vamos avanzando en el curso, es decir, haciendo ejercicios de semanas más avanzadas, el error disminuye debido a que predecimos que la mayoría de los estudiantes no realizan los ejercicios.

Estos resultados nos pueden ser útiles para ver las razones por las cuales los estudiantes dejan de hacer ejercicios, a partir de qué ejercicio la mayoría abandonan el curso etc. Pero si lo que queremos es centrarnos solo en los estudiantes que acaban el curso y predecir si van a realizar correctamente o no el próximo ejercicio debemos quedarnos con los aproximadamente 500 estudiantes que terminan el curso y analizar los resultados que obtienen en los problemas.

Tras la ejecución del mismo programa pero solo para los estudiantes mencionados obtenemos la gráfica representada en la Figura 16.

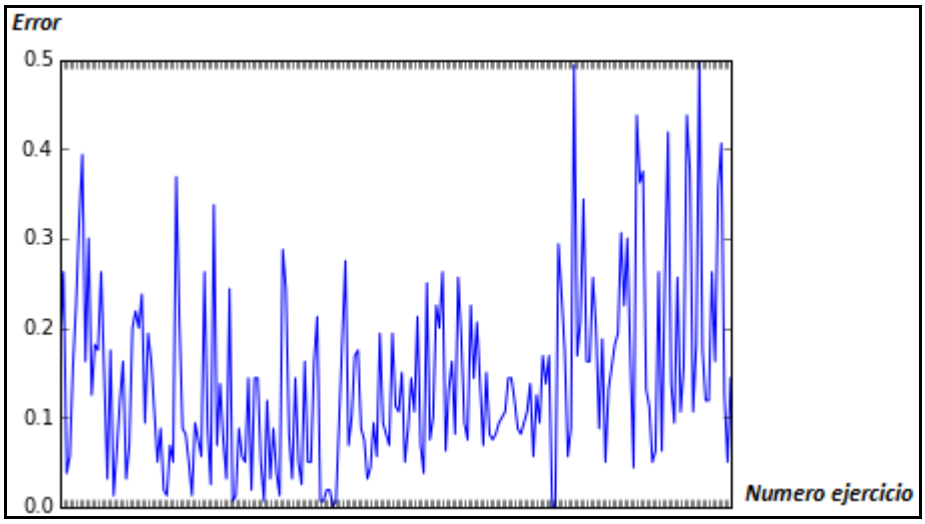


Figura 16. Gráfica sin incluir a los estudiantes que abandonan el curso

Aquí, observamos un incremento del error en la predicción de los ejercicios. Aun así el error, en ningún caso, supera el *0.5* lo que quiere decir que acertamos la mayoría de las veces. Esto es debido a que ya no predecimos con mucha frecuencia que el estudiante no va a realizar el ejercicio porque todos los estudiantes que tenemos terminan el curso y nos cuesta más clasificar si el estudiante va hacer bien o mal el ejercicio.

Si examinamos la matriz de confusión y las métricas del ejercicio 2 del problema 87 (Tabla 4), que se trata de un problema que se encuentra más o menos a la mitad del curso y por lo tanto tenemos suficientes problemas como información para predecir, además de ser uno de los que menor error de predicción poseen. Podemos comprobar que realmente los estudiantes que finalizan el curso son estudiantes que, por lo general, hacen bien los ejercicios. Además, gracias al *recall*, observamos que siempre que predecimos que el estudiante va a fallar el ejercicio, acertamos. Predecimos que 142 estudiantes resuelven bien el ejercicio y 135 de ellos lo hacen. Esto es un buen resultado. La precisión también es perfecta ya que siempre que predecimos acierto acertamos.

Predicha / ***Real***	***Fallo***	***Acierto***	***No realizado***
Fallo	4	0	0
Acierto	7	135	0
No realizado	0	0	14
Precisión	0.36	1.0	1.0
Recall	1.0	0.95	1.0
Error problema 87 ejercicio 2 *= 0.04*			

Tabla 4. Información sobre el problema 87 ejercicio 2

Tras analizar todo esto podemos ver que somos capaces de saber qué estudiante va a fallar en el ejercicio y, por lo tanto, se le podría avisar o proporcionar material de refuerzo para evitar que falle.

4.4 Predicción de la calificación en el examen final

En este apartado describiremos los resultados obtenidos para la predicción de la calificación obtenida por los estudiantes en el examen final de la asignatura, utilizando como atributos la información referente a todos los problemas realizados por los estudiantes a lo largo del curso.

En primer lugar se hace la división de datos tal y como se explica en el apartado 3.5 y se entrena mediantes *Random Forest* con 300 árboles. En la Figura 17 se indica la nota predicha y la real respectivamente, en la que se tiene en cuenta todos los problemas realizados por los estudiantes antes del examen final, para validación cruzada en 3 particiones.

```
Real              Predicha
(0.6101694915, 0.6637491751114308)
(0.8644067797, 0.8671451206870153)
(0.9322033898, 0.8590013129509693)
(0.813559322, 0.7064537977187677)
(1.0, 0.861198499534633)
(0.7118644068, 0.7560030205254581)
(0.7796610169, 0.8535359020235622)
(0.8813559322, 0.8624287210052631)
(0.9491525424, 0.8553167504264786)
(0.7627118644, 0.8363851049321308)
(0.7796610169, 0.7631215193334653)
(0.9152542373, 0.8723709537807387)
(0.8305084746, 0.8585569737382163)
(0.8644067797, 0.8402637238924355)
(0.8813559322, 0.8564379513058799)
(0.8983050847, 0.856415241064242)
(0.6101694915, 0.6666160135578781)
(0.9152542373, 0.8605841634525306)
(0.7457627119, 0.861804409327255)
(1.0, 0.8585556384248391)
(0.8813559322, 0.8492140587284904)
(0.6610169492, 0.7545523389729452)
(0.1016949153, 0.4714324915514757)
(0.3050847458, 0.5563164506920919)
(0.7627118644, 0.8438020076436896)
(0.7966101695, 0.6668860593254813)
(0.8644067797, 0.8375127762959502)
(0.7457627119, 0.7102933216529514)
(0.813559322, 0.8446898666888849)
(0.6949152542, 0.5994464357758766)
(0.8644067797, 0.8492705777084362)
(0.7457627119, 0.8358282031833314)
(0.3728813559, 0.8324145533121322)
(0.5254237288, 0.6638864569222278)
(0.7627118644, 0.8499001346431588)
(0.8305084746, 0.8269787475549941)
(0.8305084746, 0.8328896425935189)
(0.9491525424, 0.8633288976622752)
(0.8474576271, 0.8255091262358875)
(0.8813559322, 0.8471414885798415)
(0.7627118644, 0.8321698469058919)
```

Figura 17. Predicciones de la nota del examen final de los estudiantes.

Los errores obtenidos para la última semana son los siguientes:

Error cuadrático medio *= 0.030*

Error absoluto *= 0.075*

Con estos resultados si predijésemos la nota sobre *"10",* tendríamos un error de *"0.75"* décimas respecto a la original.

Esta información puede hacernos pensar si el examen final es realmente necesario ya que parece posible predecir con bastante exactitud la calificación obtenida por un estudiante a partir de los ejercicios realizados.

Por otra parte, nos interesa ahora ver a partir de qué semana del curso podríamos predecir la calificación en el examen final con un error aceptable.

Tal y como se explica en el apartado 3.5 la siguiente tarea consiste en la predicción de la nota en función de la semana que nos encontramos del curso mediante los problemas que han realizado los estudiantes hasta entonces. Por ejemplo, para la primera semana (que incluye el test inicial de java del curso) los errores obtenidos son los siguientes:

Error cuadrático medio *= 0.065*

Error absoluto *= 0.104*

Que es una predicción razonablemente buena de *"1.04"* puntos de diferencia en media. Para entender mejor este resultado veamos la predicción de forma gráfica. .

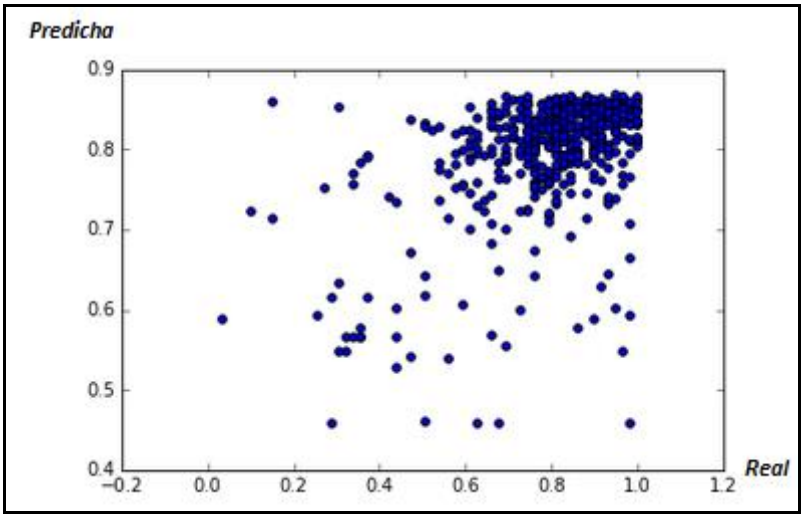


Figura 18. Grafica nube de puntos de nota predicha y nota real

En la Figura 18, se representa en el eje Y la clase predicha por el regresor y en el eje X la clase real. Podemos observar que se acumulan los puntos en una zona que es la más cercana a la media de los datos de entrenamiento, esta media es "*0.803",* por lo que este error tan bajo es debido a que las notas finales de los estudiantes están en gran parte próximas a la media.

En la Figura 19 se puede apreciar cómo van avanzando las predicciones en función de los atributos usados para el entrenamiento. Cada gráfica incluye todos los ejercicios de la semana y las anteriores. Se muestran los datos desde la semana uno (que coincide con la gráfica de la Figura 17) hasta la semana seis. Las figuras van en orden de izquierda a

derecha y de abajo a arriba desde la semana 1 hasta la semana 6. El análisis se ha realizado mediante validación cruzada de tres particiones.

En estas gráficas podemos observar claramente cómo las predicciones van mejorando ya que si trazásemos la función *f(x)=x* veríamos cómo los puntos se van acercando a esa línea por arriba y por abajo, lo que significa un menor error de predicción.

Una vez hecho esto nos interesa también evaluar cómo mejora el error respecto a si predijésemos para todos los estudiantes siempre la nota media. En la gráfica de la Figura 20 se muestra para cada semana la diferencia de error en función de si predecimos la media siempre. Por ejemplo si para la semana 1 con *Random Forest* predecimos con un error absoluto de *"0.11"*, esto es "*1.1"* puntos. Si predecimos dando el valor de la media para todos los estudiantes, tenemos un error absoluto de "*0.132*" que son "*1.32"* puntos. La mejora consistiría en la diferencia entre "*1.32"* y "*1.1"* esto es *"0.22"* decimas de mejora y es lo que queda representado en la gráfica.

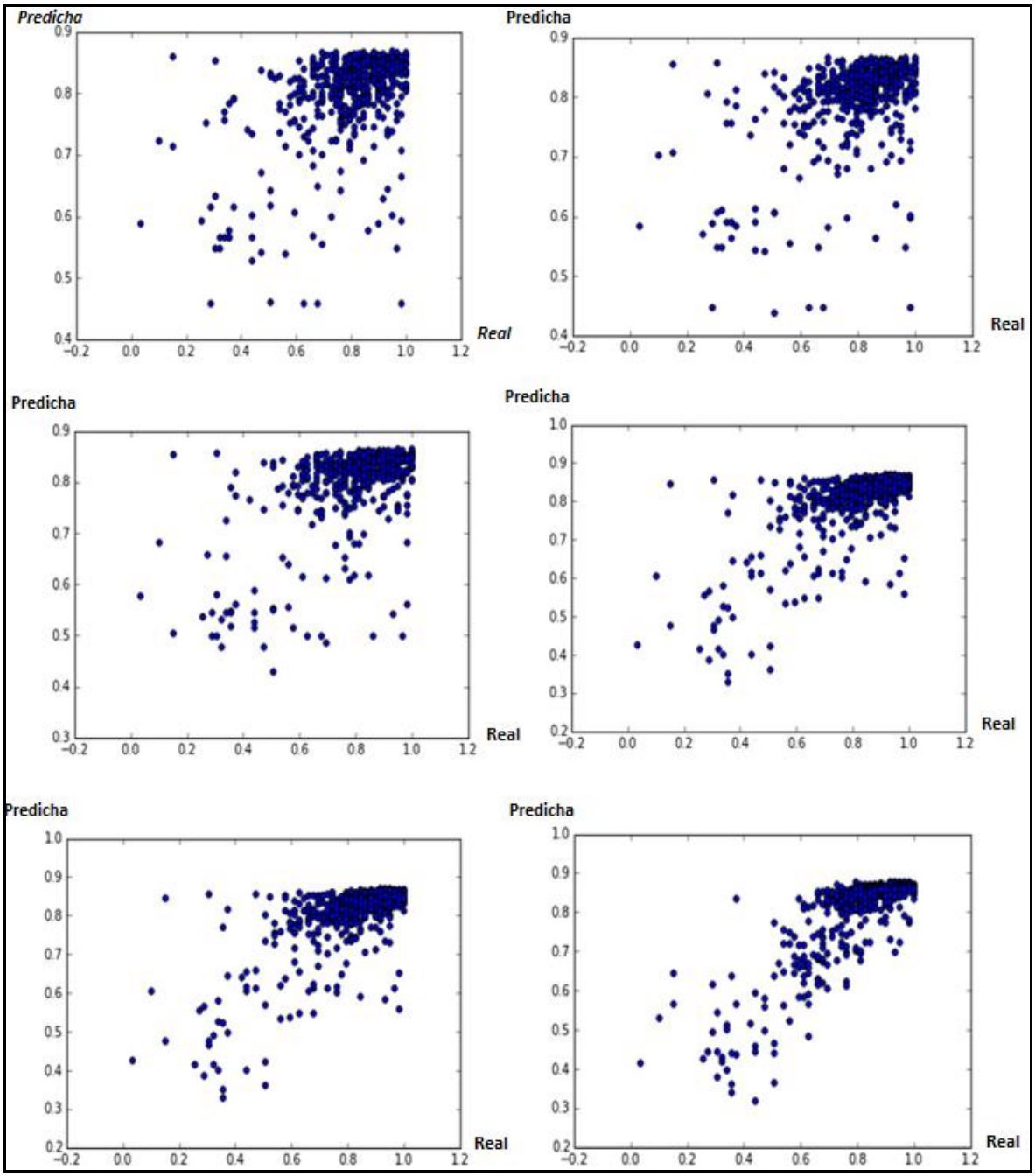


Figura 19. Evolución de la predicción de la calificación final en función de la semana

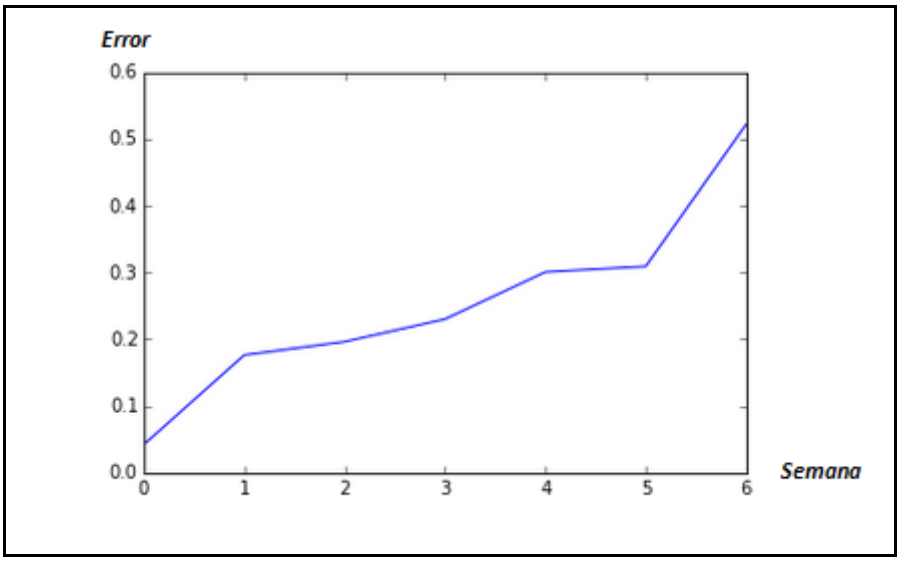


Figura 20. Mejora del error respecto a la predicción de la media

Podemos ver en la Figura 20 cómo la mejora respecto a la predicción de la media mejora según avanzamos en el curso y se dispone de más información sobre los estudiantes. Por lo tanto se predice muy bien la calificación cuanto más cerca nos encontremos de las últimas semanas.

El objetivo es avisar lo antes posible a los estudiantes que tienen más probabilidad de fracasar en la evaluación final para que puedan poner los medios oportunos para evitarlo.

En primer lugar se ha realizado la regresión tal y como hemos explicado al comienzo de este apartado y con la tecnología `filter` y `count` proporcionada por Spark se ha realizado un filtro y se ha contado el número de estudiantes cuya nota predicha y real sean respectivamente: suspensa-suspensa, aprobada-suspensa, aprobada-aprobada, suspensa-aprobada. Una vez hecho esto imprimimos una matriz de confusión, representada en la Tabla 5, en la que pueda verse el número de estudiantes que predecimos que fracasarán en la última semana, es decir con los datos de los resultados de los ejercicios hasta la semana 6.

Predicha / ***Real***	***Suspenso***	***Aprobado***
Suspenso	16	15
Aprobado	4	461

Tabla 5. Matriz de confusión de suspensos y aprobados

Podemos observar que hay 31 suspensos reales de los cuales predecimos correctamente 16. Es decir podríamos avisar a 16 estudiantes de esos 31 de que van a suspender el examen si no refuerzan sus conocimientos. También vemos que acertamos la mayoría de los aprobados. Por eso el error obtenido es tan bajo.

La Figura 21 muestra la nota predicha para todos los estudiantes que fracasan en el examen final. El formato es el mismo que en figuras anteriores: el primer valor es la nota real y el segundo la nota predicha.

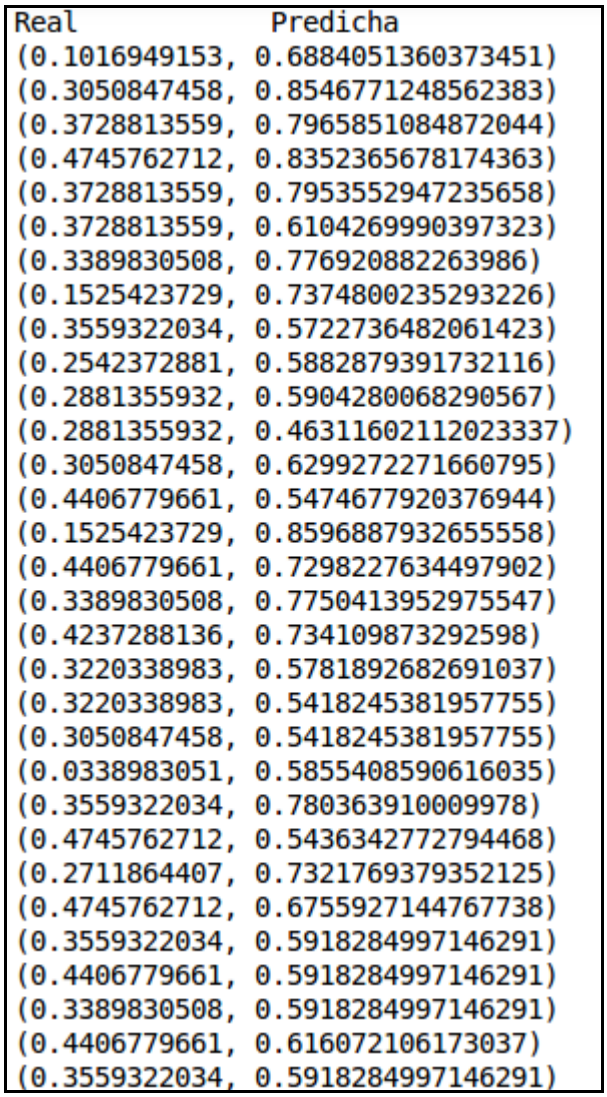


Figura 21. Predicciones de estudiantes suspensos

Gracias a la información que aparece en dicha figura, observamos que podríamos avisar a aquellos estudiantes que tienen una calificación predicha por debajo de *0.6*. De esta manera ayudaríamos a 27 de los 31 estudiantes suspensos en la última semana.

Ahora se pretende ir comprobando por semana a cuantos estudiantes podríamos avisar del suspenso de los 31 suspensos reales. Para ello de nuevo se realiza regresión mediante *Random Forest* con 300 árboles y validación cruzada de 3 particiones, filtrando y contamos aquellos estudiantes cuya nota predicha sea inferior a *0.6*. Los resultados se observan en la Tabla 6.

Semana	*Estudiantes Ayudados*
1	13
2	15
3	19
4	16
5	21
6	27

Tabla 6. Predicción de estudiantes que fracasarán en el examen final

Para cada semana comprobamos que podemos ir avisando a varios estudiantes de los 31 suspensos con la finalidad de que refuercen sus conocimientos y así sean capaces de aprobar el examen final.

5 Conclusiones y trabajo futuro

5.1 Conclusiones

El objetivo de este TFG es el análisis de las interacciones realizadas por los estudiantes en cursos online y la predicción del comportamiento del estudiante utilizando su patrón de acceso a la plataforma. Para llevar a cabo este objetivo se requieren 2 fases, procesado y limpieza de datos, y clasificación.

La primera fase se ha realizado satisfactoriamente gracias a la tecnología de Pyspark, intérprete interactivo de Python con Apache Spark. El trabajo con colecciones de datos distribuidas en este ámbito ha sido complejo, ya que no estamos habituados a trabajar con esta tecnología en la que se requiere la programación funcional para el tratamiento de los datos almacenados en los mismos. Aun así las ventajas de utilizarlos son amplias ya que reducen significativamente la cantidad de código, el tiempo de ejecución de nuestro programa y nos permiten trabajar con grandes volúmenes de datos.

El procesado y limpieza de datos ha sido un trabajo costoso debido a la necesidad de un análisis exhaustivo de los tipos de eventos procedentes del edX y de las tecnologías mencionadas anteriormente. La información de los eventos no viene completamente detallada en la plataforma edX por lo que ha sido imprescindible comparar distintos tipos de eventos para discernir entre la información relevante e irrelevante. Al final obtenemos un preprocesamiento de fichero en el cual quedará reflejado el id de cada estudiante junto con sus eventos relevantes ordenados cronológicamente.

En la segunda fase, se ha realizado predicción del resultado de los ejercicios en función de los realizados previamente por el estudiante. Hemos hecho este análisis de dos maneras distintas según los estudiantes que hemos tenido en cuenta. Un primer análisis en el cual hemos usado todos los estudiantes que habían comenzado el curso y otro solo teniendo en cuenta aquellos estudiantes que terminan el curso. Para todos los estudiantes que empiezan el curso se ha calculado el error en cada uno de los ejercicios realizado y se ha comprobado que iba disminuyendo el error a medida que avanzaban las semanas. Esto se deba a que nuestro clasificador empieza a predecir que la mayoría de los estudiantes no realizan los ejercicios ya que gran parte de esos estudiantes no acababan el curso. Esta información podría ser utilizada para predecir el abandono del curso de un estudiante. Si quitamos estos estudiantes que abandonan el curso de nuestros datos, el problema de clasificación se vuelve más complejo ya que los datos no estás desequilibrados hacía la clase “no realiza ejercicio”. Aun así nuestro error de clasificación solo supera el *45%* en dos ejercicios lo que significa que en la mayor parte los casos vamos a predecir correctamente el resultado de los ejercicios en función del estudiante que lo realiza. Por lo tanto los resultados de predicción de ejercicios han resultados satisfactorios.

Con la finalidad de ayudar a aquellos estudiantes que van suspender, se ha predicho la nota del examen final en función de los problemas que han realizado los estudiantes por cada semana, es decir, si se predice la nota del examen en la semana 3 utilizaremos los problemas del estudiante realizados hasta esa semana como información para clasificar. Para realizar estas predicciones hemos utilizado como atributos del clasificador la información del resultado de los problemas hechos por los estudiantes de forma progresiva durante el curso. Si hablamos de la predicción de la nota por semana utilizando como

atributos los ejercicios realizados por los estudiantes, comprobamos que para la primera semana predecimos la nota con "*1.04*"(sobre 10) puntos de diferencia respecto a la nota original, sin embargo si predecimos la nota para la última semana la predecimos con "*0.75*" puntos de diferencia, por lo tanto al poseer un error tan bajo de predicción podríamos poner las notas sin necesidad de realizar un examen final o dando la posibilidad de presentarse a aquellos estudiantes que no estén conformes con su nota predicha.

A la hora de intentar ayudar a aquellos estudiantes que suspenden el curso, hemos comprobado que conforme avanzan las semanas podemos avisar a 27 estudiantes de los 31 reales que suspenden el curso. A partir de este resultado podríamos dar material extra a estos estudiantes para intentar rescatarles y que aprueben la asignatura o incluso concertar tutorías online que les permitan reforzar aquellos conocimientos que tengan peor asimilados.

En definitiva, con todas las técnicas expuestas anteriormente hemos observado que podemos predecir el comportamiento de estudiantes con cierta precisión. Esto puede resultar muy útil para avisar a aquellos que van a suspender el curso o para intentar motivar y recuperar a aquellos estudiantes que predecimos que no van a realizar los ejercicios porque quieren abandonar el curso.

5.2 Trabajo futuro

El trabajo futuro en este ámbito es muy amplio. Los resultados obtenidos se han llevado a cabo teniendo en cuenta un número reducido de atributos, en concreto solo teniendo en cuenta los resultados de los ejercicios previos. Una línea que queremos seguir es la de tener en cuenta otros tipos de interacciones de los estudiantes con el curso como por ejemplo, los eventos de video combinados con los de problemas y los eventos de documentos. Adicionalmente estos atributos también podrían ser de utilidad para predecir otras variables, como la probabilidad de abandono.

Adicionalmente se podrían comparar rendimientos de computación de diferentes clúster a la hora de realizar el procesamiento de los logs con la información útil de los estudiantes ya que es una tarea que requiere de un tiempo considerable y consume bastantes recursos. También se podrían comparar rendimientos utilizando distintas herramientas de Big Data o incluso sin usarlas, por ejemplo comparando el tiempo utilizado por Apache Spark con el tiempo que se tardaría con un enfoque secuencial.

Otro trabajo futuro sería el de probar los resultados obtenidos en este curso en futuras ediciones para ver si los resultados generalizan bien.

Referencias

[1] «edX » https://www.edx.org/ [Accedido: 27-mayo-2016]

[2] «Coursera » https://es.coursera.org/ [Accedido: 27-mayo-2016]

[3] «Iversity » https://iversity.org/ [Accedido: 27-mayo-2016]

[4] «Spark» http://spark.apache.org/ [Accedido: 17-mayo-2016]

[5] «MapReduce » https://www.ibm.com/developerworks/ssa/cloud/library/cl-mapreduce/ [Accedido: 27-mayo-2016]

[6]«RDD » https://spark.apache.org/docs/0.8.1/api/core/org/apache/spark/rdd/RDD.html [Accedido: 27-mayo-2016]

[7] «Random Forest» http://scikitlearn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html [Accedido: 27-mayo-2016]

[8] «MLlib » http://spark.apache.org/mllib/ [Accedido: 27-mayo-2016]

[9] « Scikit-Learn » http://scikit-learn.org/stable/ [Accedido: 27-mayo-2016]

[10] «MOOC» https://es.wikipedia.org/wiki/Mooc [Accedido: 17-mayo-2016]

[11] «EdX» https://es.wikipedia.org/wiki/EdX [Accedido: 17-mayo-2016]

[12]«5 Vs del Big Data» https://bigdata400.wordpress.com/2014/11/11/las-5-vs-que-caracterizan-el-concepto-de-big-data/ [Accedido: 26-mayo-2016]

[13]«Big Data al servicio de la educacion» http://www.newsjs.com/url.php?p=http://es.euronews.com/2015/05/22/big-data-al-servicio-de-la-educacion/ [Accedido: 25-mayo-2016]

[14]«OpenEducationEuropa» http://openeducationeuropa.eu/en/open_education_scoreboard [Accedido: 17-mayo-2016]

[15] «Cluser mode view» http://spark.apache.org/docs/latest/cluster-overview.html [Accedido 23-mayo-2016]

[16]«*Likely to stop? predicting stopout in massive open online*» http://arxiv.org/pdf/1408.3382v1.pdf [Accedido 27-mayo-2016]

[17] «Motivation Classification and Grade Prediction for MOOCs Learners» http://dx.doi.org/10.1155/2016/2174613 [Accedido: 17-mayo-2016]

[18]«Events in the tracking logs »
http://edx.readthedocs.io/projects/devdata/en/latest/internal_data_formats/tracking_logs.html#id86 [Accedido 23-mayo]