07-011

METHODOLOGY FOR INTEGRATING DATA-BASED MODELS IN PRODUCTION ENVIRONMENTS

Bustamante Arias, Luis [(1)]; Rodríguez Montequín, Vicente [(2)]; Villanueva Balsera, Joaquín Manuel [(2)]; García González, Javier [(2)]; Mesa Fernández, José Manuel [(2)]

[(1)] ArcelorMittal, [(2)] Universidad de Oviedo

At present, there are numerous methodologies that describe what to do in the different stages of data modelling projects, but these do not describe how to do it or what architectures or technologies to use for. In practice, a multitude of heterogeneous technologies and tools coexist and are used ad hoc according to the needs of each model. In most cases, the processes of training, validation, deployment, monitoring and re-training of the models are carried out without a common technology framework that serves as a reference for all projects belonging to the same aim, which hinders the management of the models and their life cycle. The present work presents a proposal of methodology and architecture based on virtualization technologies, that serves as a guide for the correct execution of data modelling projects and that allows to manage the phases of data analysis, model development, production implementation, monitoring and testing of new configurations.

Keywords: *Project management; Machine Learning; System architecture*

METODOLOGÍA DE INTEGRACIÓN DE MODELOS BASADOS EN DATOS EN ENTORNOS DE PRODUCCIÓN

En la actualidad, existen numerosas metodologías que describen qué hacer en las diferentes etapas de proyectos de modelado de datos, pero estas no describen cómo hacerlo ni que arquitecturas o tecnologías emplear para el correcto desempeño de las mismas. En la práctica, conviven multitud de tecnologías y herramientas heterogéneas que se utilizan ad hoc según las necesidades de cada modelo. En la mayoría de los casos, los procesos de entrenamiento, validación, despliegue, monitorización y reentrenamiento de los modelos se llevan a cabo sin un marco de tecnologías común que sirva de referencia para todos los proyectos pertenecientes al mismo ámbito, lo cual dificulta muchísimo la gestión de los modelos y su ciclo de vida. El presente trabajo presenta una propuesta de metodología y arquitectura basada en tecnologías de virtualización, que sirva como guía para la correcta ejecución de proyectos de modelado de datos y que permita gestionar las fases de análisis de datos, desarrollo de modelos, implantación en producción, testing de nuevas configuraciones y monitorización de estos.

Palabras clave: *Gestión de proyectos; Aprendizaje Automático; Arquitectura de sistemas*

Correspondencia: Vicente Rodríguez Montequín

montequi@api.uniovi.es

1. Introducción

En la actualidad los sistemas inteligentes permiten dar soluciones a infinidad de problemas que hasta hace unos años las personas eran incapaces de resolver. Para el correcto desarrollo y explotación de dichos sistemas se necesitan conocimientos de distintas áreas tales como la inteligencia artificial, la cual se encarga del estudio y diseño de algoritmos capaces de tomar decisiones inteligentes, la ciencia de datos cuyo objetivo es el tratamiento, análisis y modelado de problemas, y la ingeniería de datos, capaz de diseñar arquitecturas e implementar sistemas capaces de ejecutar dichos modelos eficientemente y satisfacer la demanda de los usuarios. Con el objetivo de orientar y ayudar a los expertos de las diferentes áreas de conocimiento a cooperar y resolver los problemas que se les plantean a la hora del desarrollo de proyectos de modelado de datos, surge la necesidad de elaborar y aplicar metodologías que definan qué tareas se deben realizar en cada etapa del ciclo de vida del proyecto para su correcta ejecución.

En la actualidad, existen numerosas metodologías que describen qué hacer en las diferentes etapas de proyectos de modelado de datos, pero estas no describen cómo hacerlo ni que arquitecturas o tecnologías emplear para el correcto desempeño de las mismas. Por otro lado, no existe un marco de tecnologías común que sirva de referencia para todos los proyectos de este ámbito, sino que existen multitud de herramientas y sistemas heterogéneos.

La cada vez mayor importancia que están adquiriendo los modelos basados en datos está transformando drásticamente la forma de trabajo. Hasta hace poco, lo más frecuente era proceder al entrenamiento de los modelos de modo off-line (sin tratar datos en vivo de los usuarios ni garantizar que el servicio que proporcionan continúe disponible durante dicho entrenamieno), para posteriormente poner el modelo en producción. El número de modelos en producción era pequeño, y el seguimiento del ajuste de cada modelo se podía hacer de forma individualizada, podríamos decir incluso artesanal. Sin embargo, el número de modelos cada vez es mayor y con requisitos de monitorización más exigentes, por lo que se necesitan metodologías y tecnologías que permitan:

- Minimizar el tiempo de puesta en producción para agilizar el futuro despliegue de un mayor número de modelos.
- Definir métricas de monitorización para los modelos de tipo aprendizaje supervisado. Se entiende por modelos supervisados aquellos en los que en la fase de entrenamiento se proporciona un conjunto de observaciones definidas mediante variables o características, además de la o las salidas esperadas para cada una de estas instancias del conjunto de datos de entrada.
- Poder visualizar la calidad de las predicciones que los modelos están enviando al usuario y si estas comienzan a degradarse, reentrenar los modelos con nuevos datos.
- Evitar problemas de compatibilidad de librerías y dependencias entre unos modelos y otros. Cada uno debería de tener su propio entorno de entrenamiento independiente.

Para lograr esta transformación, conocidas empresas internacionales están desarrollando sus propias plataformas. Uno de los casos más conocidos es la plataforma de Uber, denominada Michelangelo (Hermann & Del Balso, 2017), aunque también son destacados los modelos de empresas como Amazon, Netflix, Twitter, Paypal o Mango. Los intentos de racionalizar el proceso no se centran solo en la parte tecnológica, sino también en establecer un proceso de trabajo definido o *workflow*. Así por ejemplo, el trabajo de Zaharia et al. (2018) describe el flujo de trabajo y el ciclo de vida seguido para los proyectos de

Machine Learning en una organización, denominado MLFlow. De forma similar, Hummer et al (s. f.) describen un ciclo de vida basado en DevOps, denominad en este caso ModelOps.

El presente trabajo recoge una propuesta de metodología y arquitectura que sirve como guía para la correcta ejecución de proyectos de modelado de datos, concretamente acerca de modelos denominados *Machine Learning*; subcampo de la inteligencia artificial encargada de la elaboración de modelos capaces de generalizar comportamientos a partir de una información de entrada. Quedan fuera del ámbito de aplicación aquellos sistemas más orientados a técnicas denominadas Deep Learning, por correr la mayor parte de estos sobre entornos muy específicos (Tensor Flow, Keras, etc). Se plantea una propuesta de arquitectura que permite gestionar, de acuerdo a una metodología, las fases de análisis de datos, desarrollo de modelos de *Machine Learning*, implantación en producción, pruebas de nuevas configuraciones y monitorización de estos. La arquitectura está especialmente pensada para empresas donde debido a requisitos de confidencialidad de los datos que manejan no son aceptables soluciones basadas en la nube o *"cloud computing"*, sino que se exigen soluciones *"on premise"* (solución software cuya instalación se lleva a cabo dentro de los servidores y la infraestructura TIC de la propia empresa). Además, los lenguajes utilizados para el modelado son preferentemente R y Python.

2. Análisis de plataformas existentes para *Machine Learning*

Como punto de partida se plantea la posibilidad de adoptar una de las múltiples plataformas existentes. En este sentido es de gran ayuda el informe de Gartner que anualmente analiza las plataformas punteras en el mercado (Idoine et al., 2018). Una de las conclusiones alcanzadas en el último informe es que las tres empresas más grandes con plataformas desarrolladas para *Data Science* y *Machine Learning* no han sido capaces de entrar o seguir en el segmento de líderes, lo cuál da muestra de la gran evolución que existe en este segmento. A continuación, se muestra el análisis propio realizado respecto a las plataformas existentes comercialmente que a priori podrían cubrir las necesidades planteadas.

- Machine Learning Server (HeidiSteen, s. f.-b): Plataforma de Microsoft que permite el despliegue local con distintas configuraciones para construir una plataforma de aprendizaje automático, además de soportar los lenguajes de programación R y Python. Sin embargo, al no ser una tecnología Open-source necesita ser licenciada, concretamente esta debe licenciarse junto con una versión de SQL Server lo que limita la libertad de elección de la base de datos a utilizar, por ejemplo, si se requiriera una base de datos NoSQL esta plataforma no sería factible. Por último, la plataforma no implementa ningún control del rendimiento los modelos en producción, ni forma de reentrenarlos y actualizarlos cuando la precisión de estos comienza a degradarse.
- H2O («H2O», s. f.): La plataforma facilita el proceso de puesta en producción de modelos de *Machine Learning* que ellos mismos proporcionan. Sin embargo, la plataforma está orientada al tratamiento de Big Data, puesto que las bases de datos que esta soporta están orientadas al procesamiento de grandes cantidades de datos; las necesidades de este caso de aplicación concreto no requieren de tales plataformas. Por otro lado, al serializar los modelos en objetos Java pueden surgir dificultades a la hora de importar estos en las aplicaciones .NET actuales de los usuarios.
- Azure Databricks («Azure Databricks | Microsoft Azure», s. f.): Proporciona un entorno de trabajo para la colaboración entre los diferentes roles involucrados en un proyecto de modelado de datos, además del procesamiento en la nube y acceso a múltiples herramientas de Apache, como por ejemplo Spark. El problema de esta tecnología es que no permite el despliegue local, solo en el Cloud de Azure además

de ser una herramienta totalmente de pago a pesar de utilizar numerosas herramientas Open-source de Spark.

- Azure Machine Learning («Machine Learning Studio | Microsoft Azure», s. f.): Plataforma en la nube que facilita el esfuerzo de desarrollar modelos de datos y realizar análisis mediante una interfaz que actúa a modo de lienzo interactivo. Está más enfocada a usuarios que desconocen los lenguajes de programación que a usuarios avanzados de R y Python. Por el contrario, presenta los inconvenientes de que no permite despliegue local, ni ofrece métricas para monitorizar los modelos en producción.
- DeployR (HeidiSteen, s. f.-a): Solo permite soporte para trabajar con modelos desarrollados en R. Es una tecnología que ha tenido su auge en el pasado cuando era Open-source, actualmente debido a que es una herramienta de pago, debe licenciarse con Machine Learning Server y no aporta novedades respecto otras tecnologías Open-Source que ofrecen características similares; ha entrado en desuso.
- OpenCPU («OpenCPU - Producing and Reproducing Results», s. f.): Permite la puesta en producción local de modelos desarrollados en R. Facilita la creación de API´s y el procesamiento de datos distribuido, liberando carga de computación del equipo del usuario. Pero no soporta Python ni el ofrece métricas de rendimiento ni opciones de actualización de los modelos.
- SAS Decision Manager («SAS Decision Manager Customer Product Page», s. f.): Es una herramienta que facilita la construcción de workflow a lo largo del ciclo de vida de un proyecto de modelado de datos. Además de ser la única herramienta que proporciona mecanismos de métricas de rendimiento y actualización de modelos para los sistemas en producción. Sin, embargo no ofrece compatibilidad alguna con los lenguajes Python y R, sino que la gestión de cada etapa se debe personalizar a través del lenguaje de programación de SAS. Esto conlleva a una formación extra de todos los miembros del equipo que vayan a hacer uso de SAS y el abandono de los lenguajes con los que ellos trabajaban hasta el momento. Lo que hace de SAS una herramienta muy costosa en cuanto a formación, además de adquisición.
- Google Cloud Machine Learning («ML Engine | Cloud Machine Learning Engine (Cloud ML Engine)», s. f.): Servicio Cloud que se encarga de cubrir las fases de entrenamiento y puesta en producción de determinados modelos de *Machine Learning* ofrecidos por la librería scikit-learn o el algoritmo XGBoost, además de proporcionar soporte a todos los modelos *Deep Learning* desarrollados con la librearía de TensorFlow. No permite despliegue local, ni soporte para modelos personalizados que el usuario desarrolle en lenguajes como R y Python. Además, para cubrir determinadas fases del ciclo de vida de desarrollo es necesario recurrir a otros servicios de Cloud de Google. Tampoco ofrece métricas de rendimiento y mecanismos de actualización de los modelos. En definitiva, Google Cloud Machine Learning es un servicio orientado más a dar capacidad de cómputo para modelos *Deep Learning* que a servir como plataforma de soporte para todo tipo de algoritmos de *Machine Learning*.
- Amazon SageMaker («Modelos y algoritmos de Machine Learning | Amazon SageMaker en AWS», s. f.): esta plataforma tiene como objetivo facilitar el workflow de acceso a datos, entrenamiento y puesta en producción de modelo de datos. Ofrece, además, algoritmos predefinidos de cara simplificar el modelado por parte de usuarios sin conocimientos de programación. A pesar de ello, esta plataforma no permite el despliegue local.

Del análisis se concluye que no existe ninguna plataforma comercial que satisfaga plenamente las necesidades marcadas para el caso de aplicación, por lo que se plantea una arquitectura propia que permita la definición de un flujo de trabajo único para los proyectos de modelado de datos, así como las tecnologías necesarias para que pueda realizarse cada uno de los pasos implícitos dentro del ciclo de vida del proyecto.

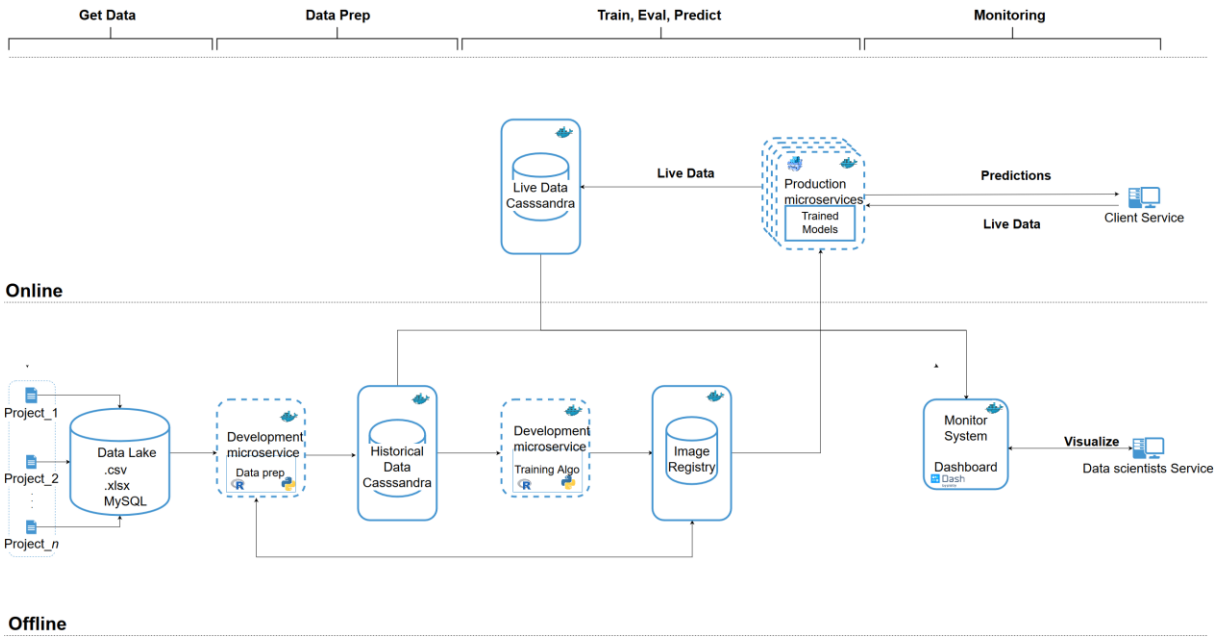
3. Descripción de la arquitectura propuesta

La arquitectura pretende dar soporte a los siguientes pasos del flujo de trabajo: obtención de datos, preparación de datos, entrenamiento y monitorización.

La arquitectura finalmente propuesta está orientada a microservicios. Este diseño permite descomponer las diferentes funcionalidades que conlleva la ejecución de un proyecto de modelado de datos en pequeños servicios cuya comunicación se realiza a través de mecanismos ligeros. Estos servicios serán gestionados y encapsulados a través de contenedores.

La Figura 1 contiene cada uno de los componentes que forma parte del sistema. Como se puede apreciar, las tecnologías se agrupan de acuerdo a los cuatro pasos del flujo de trabajo. Además, se delimita cuando se realiza el trabajo *offline* y *online*. La línea de *online* marca la separación entre lo que podemos denominar los microservicios de desarrollo y los microservicios de producción (siendo estos últimos los situados en la parte superior de la línea).

Figura 1 Arquitectura de despliegue de modelos en producción



Se comienza con una base de datos en la que se dispone de todos los datos necesarios para comenzar con el proyecto, denominada *Datalake*. En este caso de aplicación concreto, se trata de un repositorio de datos gestionado con MySQL, aunque la fuente de datos en un entorno general podría ser muy diversa.

Uno de los servicios que incorpora la arquitectura es permitir a los desarrolladores de modelos la ejecución remota del proceso de análisis y entrenamiento de los modelos. La fase de entrenamiento puede requerir mucho tiempo y una gran capacidad de cómputo qué

los desarrolladores no poseen en sus equipos locales. Por este motivo, se proponen dos tecnologías que permiten la ejecución remota de código de los lenguajes R y Python en servidores dedicados. Existen dos variantes para la preparación de datos, una para R y otra para Phyton. Las Figuras 2 y 3 describen cada uno de los procesos. En el caso de R, el procesamiento de los datos se realiza con la herramienta RStudio Server («RStudio», 2014), mientras que en el caso de Phyton se emplea el IDE Pycharm («PyCharm», s. f.).

- RStudio Server es la plataforma Open-source que provee una interfaz web a una versión de R ejecutándose en un servidor remoto Linux. El IDE es similar a RStudio lo que facilitaría la adaptación del usuario al entorno de trabajo ya que actualmente es el IDE que el equipo de desarrollo está utilizando; incorpora las herramientas de visualización de gráficos, historial, debugging y gestión del workspace. El despliegue de RStudio Server se realizará a partir de una imagen base predefinida que contenga las librerías y dependencias específicas que el equipo considere oportunas, de tal manera que cada desarrollo de los distintos modelos R se basen en las mismas librerías con el mismo número de versión. Una vez se ejecute la imagen el contenedor se creé, los ingenieros de datos podrán tener acceso remoto a su directorio de trabajo y mayor capacidad de cómputo.
- Pycharm es un IDE de desarrollo para Python. A diferencia de RStudio Server, que proporcionaba un servidor remoto donde ejecutar, almacenar y gestionar los ficheros de R, Pycharm únicamente proporciona la ejecución remota de código, a través de conexión *ssh* con el servidor o haciendo uso de contenedores. El usuario debe de gestionar localmente sus scripts. A la hora de ejecutar código remoto Pycharm proporciona la opción de desplegar un contenedor, ejecutar el código remotamente y retornar el resultado al usuario cuando el contenedor finalice su ejecución. Por cada ejecución se crea un nuevo contenedor el cual monta el directorio del proyecto de Pycharm al contenedor para su ejecución; posteriormente el contenedor es eliminado. Todas las dependencias y librerías deben estar instaladas en el contenedor que se va a utilizar como intérprete remoto.

La salida en ambos casos es un conjunto de datos histórico gestionado mediante Cassandra («Apache Cassandra», s. f.). Cassandra es la plataforma utilizada en esta arquitectura para almacenar los datos históricos, nuevos datos que se obtengan de las aplicaciones de usuario y predicciones realizadas por los modelos. Debido a la diversidad de datos que se puedan recibir en el futuro, se ha optado por una base de datos NoSQL para almacenar datos no estructurados. Cassandra es el sistema de gestión líder de bases de datos distribuidas NoSQL y orientado por columnas, diseñada para gestionar grandes cantidades de datos. Las estructuras de datos usadas en Cassandra son más específicas que las utilizadas en bases de datos relacionales, permitiendo mayor velocidad de lectura y escritura; un uso típico de Cassandra es en aplicaciones web que trabajan con peticiones en tiempo real.

Figura 2 Preparación de datos en R

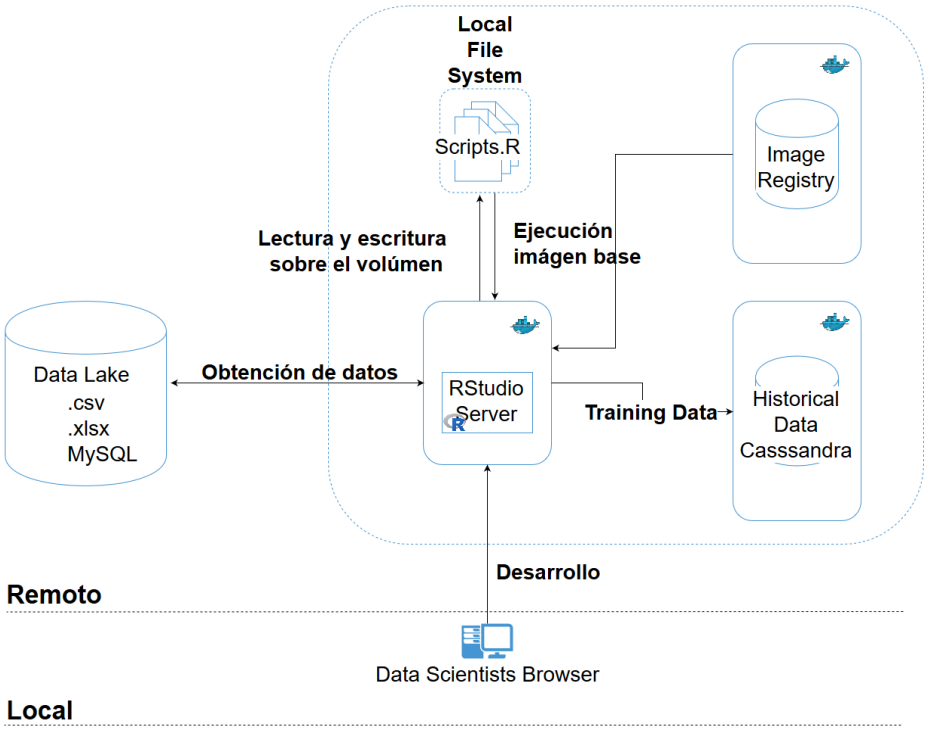
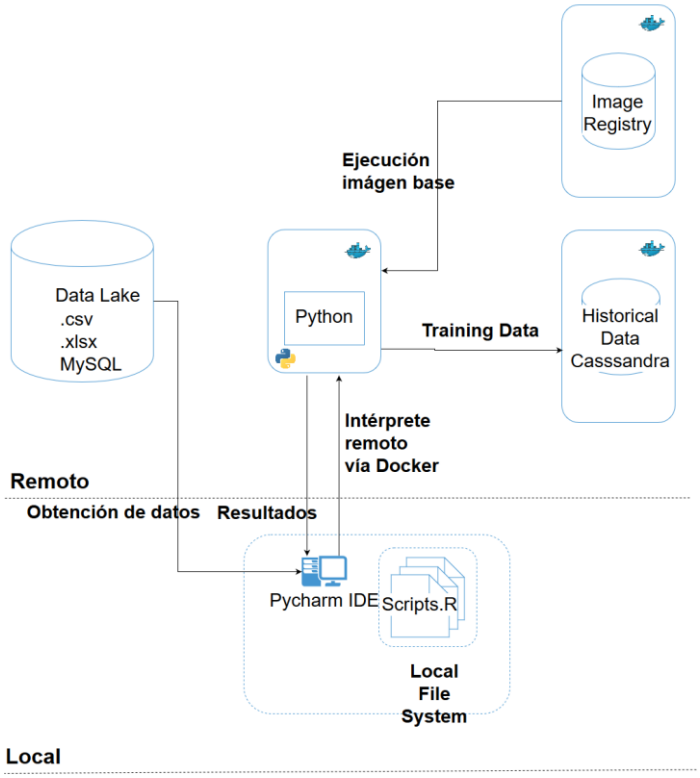


Figura 3 Preparación de datos en Phyton



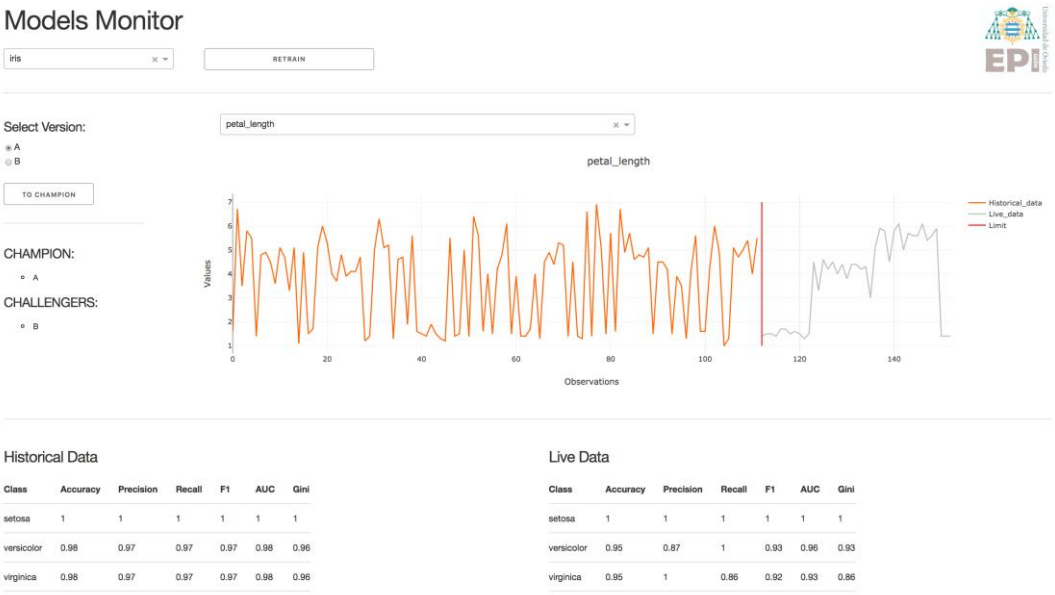
A continuación, se tiene el proceso de entrenamiento, el cual tiene lugar también dentro de los microservicios de desarrollo. Finalizada la elaboración de cada modelo junto con cada una de las versiones elaboradas para realizar la validación se elaboran los correspondientes servicios de producción. Estos servicios de producción se pueden desplegar como instancias únicas o utilizando la tecnología *Docker Swarm* (tecnología para desplegar contenedores en múltiples nodos) en el caso de ser necesaria alta disponibilidad. Una vez en producción, estos devolverían las predicciones a las aplicaciones de los usuarios y almacenarían el *"Live Data"* o datos en vivo en la instancia de la base de datos dedicada a albergar dicha información. Toda esta información se hace accesible al usuario a través de un cuadro de mando de monitorización.

El diseño del cuadro de mando de monitorización propuesto se muetra en la Figura 4. El contenido de cada uno de los campos de la aplicación se genera de forma dinámica en base a los cambios que se produzcan en Cassandra y las acciones del usuario en la aplicación; este tipo de componentes se denominan componentes reactivos. Para el desarrollo del cuadro de mando se emplea el framework Dash («Dash by Plotly», s. f.) desarrollado en Phyton, que permite la creación de componentes reactivos mediante el uso de *callbacks* en las funciones que forman parte del controlador de la aplicación. Los componentes reactivos de la aplicación son los siguientes:

- Selector de modelo: Permite seleccionar el modelo en producción que se quiere visualizar en la aplicación web.
- Reentrenamiento: Reentrena el modelo seleccionado con todos los datos disponibles en Cassandra, es decir, se agregan los datos históricos junto con los datos en vivo. Puesto que para cada modelo pueden existir varias versiones, se tendrá en cuenta la versión seleccionada por el usuario.
- Selección de versión: Permite visualizar la evolución gráfica y las métricas numéricas de las diferentes versiones entrenadas de un mismo modelo.
- To Champion: Botón que permite convertir una versión *challenger* o candidata a *champion*, lo que se traduce en que las respuestas a las peticiones de los usuarios serán realizadas por la versión del nuevo modelo seleccionado.
- Lista actual de modelos *champion* y *challenger*.
- Selector de variable: Seleccionado un modelo y su versión, permite elegir entre las diferentes variables del modelo y mostrar su evolución. Existen dos tipos de variables a mostrar:
    - Inputs: Variables de entrada del modelo. Se muestra una evolución de los valores que ha tomado la variable tanto en el conjunto histórico como el conjunto en vivo.
    - Output: Variable que se quiere predecir. Para este tipo de variables se muestra la comparativa entre los valores reales y predichos en los conjuntos histórico y en vivo.
- Gráfico de monitorización: Seleccionado un modelo, versión y variable que se quiere visualizar se muestra el gráfico de la evaluación o comparación de valores predichos y reales; dependiendo del tipo de la variable seleccionada.
- Métricas de monitorización para el conjunto histórico y los datos en vivo: Se muestran dos tablas con los valores numéricos de las métricas de monitorización, que dependerán de si el modelo seleccionado es de tipo clasificación o regresión.

El cuadro de mando de monitorización se actualiza 2 veces por minuto para realizar un seguimiento de los cambios en la base de datos Cassandra.

Figura 4 Esquema del cuadro de mando



Para probar la plataforma finalmente adoptada se han realizado pruebas modelando problemas de *Machine Learning* académicos, obteniendo resultados satisfactorios.

4. Conclusiones

El resultado de este trabajo ha sido la implantación de una plataforma que permita la gestión completa del ciclo de vida asociado a los proyectos de *Machine Learning* atendiendo a las necesidades específicas del entorno de trabajo donde se ha aplicado. Al no existir ninguna tecnología que cubra las necesidades planteadas se optó por el diseño de un sistema que permite el desarrollo remoto y puesta en producción de los modelos basados en datos en base a microservicios que los miembros del equipo pueden crear y utilizar según sus necesidades.

Debido a la diversa pluralidad de modelos de aprendizaje existentes hoy en día. La solución planteada en este proyecto no pretende ser un prototipo cerrado, sino que, pretende plantear una propuesta de solución que sirva de base para que pueda ser extendido a cualquier proyecto y que cubra las necesidades del equipo encargado del desarrollo de modelos de *Machine Learning*.

Referencias

Apache Cassandra. (s. f.). Recuperado 12 de abril de 2019, de http://cassandra.apache.org/

Azure Databricks | Microsoft Azure. (s. f.). Recuperado 12 de abril de 2019, de

https://azure.microsoft.com/es-es/services/databricks/

Dash by Plotly. (s. f.). Recuperado 12 de abril de 2019, de Plotly website: https://plot.ly/products/dash/

H2O. (s. f.). Recuperado 12 de abril de 2019, de Open Source Leader in AI and ML website: https://www.h2o.ai/products/h2o/

HeidiSteen. (s. f.-a). About DeployR - DeployR 8.x. Recuperado 12 de abril de 2019, de https://docs.microsoft.com/en-us/machine-learning-server/deployr/deployr-about

HeidiSteen. (s. f.-b). Welcome to Machine Learning Server. Recuperado 12 de abril de 2019, de https://docs.microsoft.com/en-us/machine-learning-server/what-is-machine-learning-server

Hermann, J., & Del Balso, M. (2017). Meet Michelangelo: Uber's machine learning platform. *URL https://eng. uber. com/michelangelo*.

Hummer, W., Muthusamy, V., Rausch, T., Dube, P., & El Maghraoui, K. (s. f.). *ModelOps: Cloud-based Lifecycle Management for Reliable and Trusted AI*.

Idoine, C., Krensky, P., Brethenoux, E., Hare, J., Sicular, S., & Vashisth, S. (2018). Magic Quadrant for Data Science and Machine-Learning Platforms. *Gartner, Inc*.

Machine Learning Studio | Microsoft Azure. (s. f.). Recuperado 12 de abril de 2019, de https://azure.microsoft.com/es-es/services/machine-learning-studio/

ML Engine | Cloud Machine Learning Engine (Cloud ML Engine). (s. f.). Recuperado 12 de abril de 2019, de Google Cloud website: https://cloud.google.com/ml-engine/

Modelos y algoritmos de Machine Learning | Amazon SageMaker en AWS. (s. f.). Recuperado 12 de abril de 2019, de Amazon Web Services, Inc. website: https://aws.amazon.com/es/sagemaker/

OpenCPU - Producing and Reproducing Results. (s. f.). Recuperado 12 de abril de 2019, de https://www.opencpu.org/

PyCharm: the Python IDE for Professional Developers by JetBrains. (s. f.). Recuperado 12 de abril de 2019, de JetBrains website: https://www.jetbrains.com/pycharm/

RStudio. (2014, abril 4). Recuperado 12 de abril de 2019, de RStudio website: https://www.rstudio.com/products/rstudio/

SAS Decision Manager Customer Product Page. (s. f.). Recuperado 12 de abril de 2019, de http://support.sas.com/software/products/decision-manager/index.html

Zaharia, M., Chen, A., Davidson, A., Ghodsi, A., Hong, S. A., Konwinski, A., … Parkhe, M. (2018). Accelerating the Machine Learning Lifecycle with MLflow. *Data Engineering*, 39.