UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

GRADO EN INGENIERÍA INFORMÁTICA

MLOPS PARA EL DESARROLLO Y PUESTA EN PRODUCCIÓN DE MODELOS DE MACHINE LEARNING

MLOPS FOR THE DEVELOPMENT AND PRODUCTION DEPLOYMENT OF MACHINE LEARNING MODELS

Realizado por
PABLO VALDERRAMA SANTIAGO

Tutorizado por
LLANOS MORA LÓPEZ

Departamento
LENGUAJES Y CIENCIAS DE LA COMPUTACIÓN
UNIVERSIDAD DE MÁLAGA

MÁLAGA, septiembre 2021

Resumen

Machine Learning Model Operationalization Management (MLOps) constituye una metodología de trabajo orientada al desarrollo de modelos de predicción basados en algoritmos de Machine Learning. Esta metodología está conformada por un conjunto exhaustivo de principios, recomendaciones, directrices y buenas prácticas enfocadas en el abordaje metodológico del desarrollo de modelos de Machine Learning desde su experimentación inicial hasta su puesta en producción. Para alcanzar este objetivo, esta metodología propone una división del desarrollo de estos proyectos en 4 fases consecutivas. Estas fases comprenden las tareas de desarrollo de modelos, preparación de los modelos para el despliegue en producción, el despliegue en producción y la monitorización de los modelos desplegados. Este Trabajo de Fin de Grado explora de forma teórica este conjunto de principios metodológicos estudiando cada una de las fases de desarrollo propuestas. De forma paralela, se aborda el desarrollo de un modelo de predicción de Machine Learning para la predicción de consumos energéticos horarios individuales y su puesta en producción. En este desarrollo se adoptará MLOps para llevar a cabo el entrenamiento de modelos basados en RandomForest y K-Means, el diseño de un código para dar ejecución a estos modelos, el diseño de imágenes para su contenerización y su despliegue en un clúster de Kubernetes.

Palabras clave: MLOps, modelo, predicción, Machine Learning, entrenamiento, despliegue, Kubernetes, producción, pipeline

Abstract

Machine Learning Model Operationalization Management (MLOps) is a working methodology oriented to the development of prediction models based on Machine Learning algorithms. This methodology consists of an exhaustive set of principles, recommendations, guidelines and best practices focused on the methodological approach to the development of Machine Learning models from their initial experimentation to their implementation in production. To achieve this objective, this methodology proposes a division of the development of these projects in 4 consecutive phases. These phases comprise the tasks of model development, model preparation for production deployment, production deployment and monitoring of the deployed models. This Final Degree Project explores theoretically this set of methodological principles by studying each of the proposed development phases. In parallel, the development of a Machine Learning prediction model for the prediction of individual hourly energy consumption and its deployment in production is addressed. In this development, MLOps will be adopted to carry out the training of models based on RandomForest and K-Means, the design of a code to execute these models, the design of images for their containerization and their deployment in a Kubernetes cluster.

Keywords: MLOps, model, prediction, Machine Learning, training, deployment, Kubernetes, production, pipeline

Índice

1

Introducción

1.1 Motivación

En la actualidad, las técnicas y herramientas de Machine Learning (conocido como Aprendizaje Automático en español) están siendo adoptadas en la práctica totalidad de industrias y disciplinas; sin embargo, más de la mitad de los análisis estadísticos y modelos de Machine Learning creados por las organizaciones nunca llegan a desplegarse en producción.

La puesta en producción de los modelos de predicción experimentales desarrollados por investigadores o científicos de datos constituye un desafío técnico en el que están involucrados diversos campos y disciplinas de las ciencias de la computación. Un desafío que, frecuentemente, no se aborda con éxito.

Actualmente existen avanzadas y potentes herramientas de Machine Learning que permiten la construcción de sistemas complejos con gran rapidez. Sin embargo, estos desarrollos acelerados de modelos de Machine Learning suelen acarrear una gran deuda técnica (Sculley et al., 2014) que las organizaciones tendrán que asumir a la hora de implementar estos modelos como sistemas preparados para dar servicio a gran escala de forma fiable y automatizada.

A raíz de la necesidad de solventar estas dificultades han surgido, durante los últimos años, diversas disciplinas y metodologías enfocadas en la disminución de la deuda técnica y la estandarización del ciclo de vida del desarrollo de proyectos basados en Machine

Learning.

Estos principios se engloban en el denominado Machine Learning Model Operationalization Management (MLOps), un concepto de muy reciente aparición que está rápidamente empezando a constituir un componente crítico en el desarrollo y despliegue exitoso de los modelos de Machine Learning (Visengeriyeva et al., s.f.).

El nombre de MLOps y su definición están basados ampliamente en el concepto de DevOps (Atlassian, s.f.), una disciplina muy extendida y generalizada que tiene como objetivo la estandarización del proceso de desarrollo de *software* y su integración, actualización y despliegue. Aunque similares en objeto, DevOps no puede ser directamente aplicado a los proyectos de Machine Learning. Este impedimento tiene su origen en la naturaleza dinámica y mutable de los datos, que cambian junto al fenómeno del mundo físico que se desea modelar. Existe, por tanto, la necesidad de adaptar continuamente los modelos desarrollados para reflejar estos cambios en los datos disponibles. DevOps aborda el desarrollo de proyectos con un código que permanece estático una vez desplegado y, por tanto, no abarca el desarrollo de proyectos que, además de código, también están basados en datos. MLOps surge para suplir estas carencias.

MLOps es una disciplina relativamente nueva cuyo surgimiento se puede ubicar entre los años 2018 y 2019. Sin embargo, con anterioridad a esta, podemos encontrar también otras metodologías similares relacionadas con el ámbito de la ciencia de datos y el Machine Learning que pueden ser confundidas con MLOps, pero que difieren sustancialmente en objetivo y cuyos principios no están enfocados específicamente en la estandarización del ciclo de vida de proyectos basados en Machine Learning. Entre ellas, podemos encontrar:

- AIOps, una disciplina orientada a la aplicación de inteligencia artificial a los procesos de DevOps. Tiene como objetivo automatizar y mejorar los procesos habituales que forman parte del desarrollo y despliegue del *software* general mediante agregación de datos y análisis avanzados. (IBM Cloud Education, 2020)
- DataOps, un término introducido por IBM en 2014 que hace

referencia a una serie de buenas prácticas y principios enfocados en la calidad de los datos, la gestión de metadatos y el posterior análisis de estos (Quoma, 2019).

DataOps, como disciplina, interseca con MLOps en algunos niveles; sin embargo, MLOps está más evolucionado y proporciona mayor robustez cuando es aplicado a este tipo de proyectos. Esto es debido a que MLOps cuenta con una serie de características concretas que están enfocadas específicamente en el abordaje completo de estos desarrollos, desde su proyección hasta su puesta en servicio para los usuarios finales.

MLOps establece una metodología aplicada a los proyectos de Machine Learning abordando íntegramente sus fases de desarrollo:

- El desarrollo y entrenamiento de modelos.
- La preparación del modelo para su puesta en producción.
- La puesta en producción del modelo.
- La monitorización y reentrenamiento de modelos.

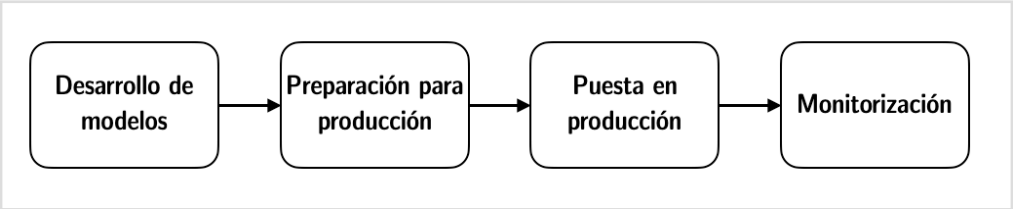

Figura 1.1: Esquema del ciclo de vida MLOps

Gracias a su conjunto exhaustivo de principios, recomendaciones, directrices y buenas prácticas, adoptar esta disciplina constituirá un elemento esencial en el desarrollo exitoso de cualquier proyecto de Machine Learning y, por tanto, resulta de gran interés su estudio, análisis y posterior aplicación en el desarrollo del caso práctico propuesto como parte de este trabajo.

Hay otros campos y disciplinas de la informática que no están abordados explícitamente en MLOps pero que están directamente involucrados en el desarrollo de proyectos de este tipo. Así, forman parte de estos proyectos especialidades como pueden ser: la seguridad de la información, el uso y administración de bases de datos, la administración de sistemas operativos, la ingeniería del *software* y las redes y sistemas distribuidos, entre otros.

1.2 Objetivos

Este proyecto tiene como objetivo la exploración del ciclo de vida de los proyectos de Machine Learning en todas sus fases de desarrollo desde el punto de vista de MLOps como metodología de trabajo. Así, serán estudiados cada uno de los elementos y etapas que forman parte del ciclo de vida de estos proyectos y se abordarán las directrices, métodos de trabajo, orientaciones y buenas prácticas que corresponden a cada una de las fases. Para ello, será necesario el estudio y análisis de MLOps como disciplina y la exploración de su conjunto de principios.

Además, para poder ejemplificar y poner en práctica cada uno de los elementos metodológicos estudiados, se abordará, de forma simultánea, el desarrollo de un proyecto real de Machine Learning en todas sus fases, desde la experimentación inicial hasta su puesta en producción.

Por tanto, como caso práctico, se propone el desarrollo de un modelo de Machine Learning para la predicción de consumos energéticos horarios individuales y su puesta en producción en una infraestructura *hardware* de forma que pueda ser de utilidad en aquellos puntos de la red de suministro donde exista la necesidad de conocer predicciones de la demanda eléctrica.

Para ello, será necesario el entrenamiento y desarrollo del modelo de predicción y su correspondiente contenerización y despliegue. Para el almacenamiento de todos los datos necesarios para el funcionamiento del sistema, será necesario, asimismo, desplegar una base de datos que se ajuste a las necesidades del proyecto. Para el despliegue de procesos se llevará a cabo el aprovisionamiento de un clúster de orquestación de contenedores. Se requerirá, además, la implementación de los *pipelines* necesarios para la automatización de las tareas de contenerización y despliegue. Por último, se pondrán en funcionamiento medios para la monitorización del sistema.

Todo este desarrollo se llevará a cabo adoptando el conjunto de principios aportado por MLOps, siguiendo las prácticas de seguridad informática necesarias para la protección de datos personales y, además, respetando el conjunto de buenas prácticas del desarrollo de *software* general.

1.3 Estructura de la memoria

Este Trabajo de Fin dedicará un capítulo por cada fase de desarrollo de los proyectos basados en Machine Learning, además de los capítulos introductorio y de conclusión. Por tanto, este trabajo consta de 6 capítulos, cuyos títulos son: Introducción, Desarrollo de modelos, Preparación para producción, Despliegue en producción, Monitorización de modelos y Conclusiones.

Excepto el introductorio y el de conclusiones, cada uno de estos capítulos abordará los contenidos teóricos correspondientes a cada una de las fases de desarrollo de los proyectos de Machine Learning. En cada uno de estos capítulos, el último apartado abordará el desarrollo del caso práctico en su fase correspondiente.

2

Desarrollo de modelos

Este capítulo explorará la primera de las cuatro principales fases de desarrollo en las que se dividirá el ciclo de vida de los proyectos basados en Machine Learning.

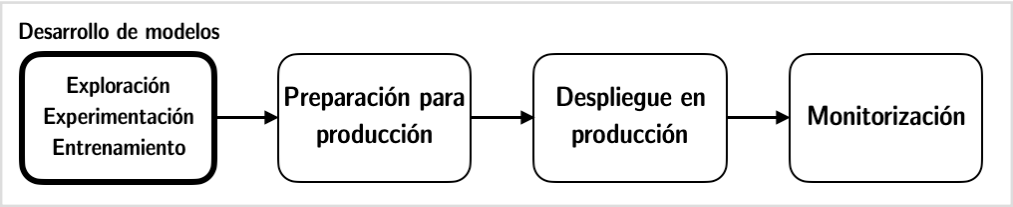

Figura 2.1: Esquema del ciclo de vida MLOps con desarrollo de modelos ampliado

Se introducirán los conceptos de modelo predictivo y modelo de Machine Learning y se abordarán los primeros aspectos a tener en cuenta durante el desarrollo desde la óptica metodológica del MLOps. Finalmente, se introducirá y dará comienzo al desarrollo del caso práctico en su primera fase.

2.1 Introducción

Un modelo es una representación teórica y matemática de algún

aspecto de la realidad. Estos modelos tratan de abstraer y generalizar aquellos fenómenos del mundo físico que sea necesario modelar y, haciendo uso de estos, es posible realizar predicciones o estimaciones de su comportamiento. En concreto, los modelos de Machine Learning utilizan técnicas algorítmicas que les permiten inducir estas reglas y representaciones matemáticas de forma automática a partir de un conjunto de datos de entrenamiento.

Un modelo de Machine Learning, una vez entrenado, se puede entender como una fórmula matemática que produce resultados a partir de unos datos de entrada concretos. Este resultado puede ser utilizado como estimación del comportamiento del fenómeno que se trata de modelar.

Estos modelos están basados en teorías estadísticas y sus algoritmos están diseñados para construir esas fórmulas matemáticas de forma automática a partir de los datos que le sean proporcionados durante su entrenamiento. Estos datos representan el mundo tal y como este podía ser encontrado en el estado y momento en el que fueron recolectados.

De esta forma, los modelos generados representarán el mundo tal y como este era en el pasado; sin embargo, gracias a su capacidad de generalización, estos pueden ser usados para realizar predicciones sobre el futuro. Aún así, el mundo y los fenómenos físicos estudiados son dinámicos y evolucionan, dejando de esta manera obsoletos los modelos que fueron entrenados demasiado atrás en el tiempo. Debido a esta circunstancia, deberán ponerse en funcionamiento medios y mecanismos para la monitorización y reentrenamiento de los modelos en el momento en que se detecte que el rendimiento se ha degradado más allá de los valores establecidos como aceptables.

Existen dos clasificaciones principales en las que se pueden dividir los algoritmos de Machine Learning:

- Algoritmos supervisados: son aquellos que necesitan ser entrenados introduciendo, junto a sus datos de entrenamiento, aquel valor correcto que esperamos que el algoritmo sea capaz de producir si se ejecutase con un dato de entrada concreto. Así, por ejemplo, si tratamos de entrenar un algoritmo supervisado para realizar clasificación, será necesario

proporcionar al algoritmo, junto al conjunto de datos de entrenamiento, la clase o categoría correcta en la que debe clasificarse cada dato.

- Algoritmos no supervisados: son aquellos que no requieren conocer el valor correcto o esperado durante su entrenamiento. De esta forma, siguiendo con el mismo ejemplo, un algoritmo de clasificación no supervisado no requerirá, durante su entrenamiento, la clase o categoría correcta asociada a cada dato. El algoritmo generará de forma autónoma las distintas clases o categorías y realizará la clasificación en base a estas.

En general, la construcción de modelos de Machine Learning requiere de una una serie de componentes básicos que serán necesarios durante el proceso de desarrollo. A su vez, cada uno de los componentes requerirá tener en cuenta una serie de consideraciones concretas. Estos elementos son:

- Datos de entrenamiento: se debe contar con un conjunto de datos de calidad que represente con fidelidad el aspecto del mundo que se desee modelar. Los datos de entrenamiento pueden sufrir distintos tipos de sesgos que, de no ser corregidos, darán lugar a un entrenamiento incorrecto o sesgado del modelo. La cuidadosa recolección de datos fiables de entrenamiento constituye un elemento fundamental del proceso de entrenamiento de modelos. Algunos algoritmos requieren una enorme cantidad de datos de entrenamiento por lo que, en estos casos, también será necesario el correcto almacenamiento y gestión de estas grandes cantidades de datos.

- Métricas de rendimiento: durante el proceso de desarrollo de modelos se tratará de optimizar una serie de valores que representen la calidad predictiva del modelo y su adecuación y fiabilidad. Estos valores, por lo general, serán el resultado de calcular, mediante distintas técnicas, el error entre los datos reales y los datos predichos por el modelo. Además, existen aplicaciones que requieren que el modelo sea capaz de ejecutarse con rapidez por lo que, en estos casos, también serán necesarias métricas sobre la eficiencia o rendimiento computacional del modelo en ejecución.

- Algoritmo de Machine Learning: existen diversos algoritmos de Machine Learning que pueden ser usados con diversos propósitos y en distintos ámbitos. En general, cada algoritmo presentará una serie de ventajas y desventajas que deberán tenerse en cuenta a la hora de elegir el algoritmo más apropiado para cada aplicación. Más adelante se expondrá un esquema general de algunos de los tipos de algoritmo disponibles y sus consideraciones correspondientes.

- Hiperparámetros: cada algoritmo requiere ser configurado mediante una serie de valores que alteren o corrijan el algoritmo durante su ejecución, estos valores son los hiperparámetros. En el caso de, por ejemplo, un algoritmo basado en árboles de decisión, uno de sus hiperparámetros será la profundidad del árbol.

- Conjunto de datos de evaluación: para poder evaluar el modelo en base a las métricas definidas previamente, será necesario contar con un segundo conjunto de datos distinto e independiente sobre el que ejecutar el modelo. De esta forma, el análisis de error será más fiable al representar de forma más certera el comportamiento que mostrará el modelo con nuevos datos no pertenecientes a su conjunto de entrenamiento.

Como ha sido expuesto anteriormente, la elección de un algoritmo concreto dependerá de la aplicación y del conjunto de ventajas y desventajas asociados a cada uno de ellos. En general, podemos realizar una breve exposición de algunos de los principales algoritmos y sus características, mostrando así una perspectiva concreta del estado de la técnica actual en el apartado de algoritmos de Machine Learning.

Así, podemos hacer una breve clasificación de algoritmos en tres tipos y presentar sus principales consideraciones. Estos son:

- Algoritmos de tipo lineal: podemos encontrar dos principales algoritmos dentro de esta categoría: la regresión lineal ("Linear regression", 2021) y la regresión logística ("Logistic regression", 2021). Estos algoritmos están ampliamente extendidos y generalizados ya que, a pesar de su sencillez, siguen resultando útiles en numerosas aplicaciones en la actualidad. Estos

algoritmos sufren una marcada tendencia al sobreajuste.

- Algoritmos basados en árboles: entre los que podemos encontrar Árboles de Decisión, Random Forest ("Random forest", 2021) o Gradient Boosting ("Gradient boosting", 2021). Estos algoritmos pueden ser inestables ya que pequeñas variaciones en los datos de entrada pueden provocar diferencias sustanciales en los modelos generados. Además, pueden tener un bajo rendimiento computacional, lo que puede ser perjudicial en algunas aplicaciones.

- Algoritmos basados en redes neuronales: en esta categoría el principal y más importante algoritmo es el Deep Learning ("Deep learning", 2021). Este tipo de algoritmos, a pesar de sus excelentes capacidades para abstraer conceptos, son muy lentos de entrenar y requieren grandes cantidades de energía y datos de entrenamiento. Además, una vez entrenados, son prácticamente imposibles de comprender, lo cual puede resultar perjudicial en aquellas aplicaciones en las que se requiera dar una justificación del dato generado por el modelo.

La elección del tipo de algoritmo requerirá tener en cuenta una serie de consideraciones desde el punto de vista del MLOps que van más allá de su adecuación como modelo predictivo.

2.2 Exploración de datos y elección de variables explicativas

Como ya ha sido discutido anteriormente, la disponibilidad de un conjunto de datos de entrenamiento de calidad es un requerimiento esencial para el correcto entrenamiento de un modelo de Machine Learning. Por este motivo, uno de las primeras tareas a abordar por parte de los científicos de datos consiste en la exploración y análisis preliminar de los datos disponibles. Estos datos pueden presentar una serie de problemas que pueden impedir su uso. Algunos problemas de este tipo pueden ser: inconsistencia, falta de compleción o falta de precisión, entre otros. Por tanto, estos procesos de exploración inicial deberán llevar a cabo una serie de tareas enfocadas en detectar estos problemas y tratar de darles solución.

Para poder llevar a cabo estas tareas, resulta necesario tener a

disposición conocimiento experto sobre el ámbito o disciplina para el que va a desarrollarse un modelo concreto. Este conocimiento podría estar relacionado con, por ejemplo, ámbitos como: la economía, las energías renovables, la medicina, las ciencias sociales o las ciencias naturales, por nombrar algunos. Este conocimiento puede ser aportado por el propio científico de datos o, más comúnmente, por otro miembro del equipo experto en el ámbito de aplicación, quien aportará al resto del equipo toda la información que sea necesaria en cada fase del desarrollo del proyecto.

Gracias a este acceso a conocimiento experto, los científicos de datos pueden llevar a cabo las tareas necesarias para abordar esta exploración de datos inicial y tratar de solventar los errores detectados. Entre este conjunto de tareas podemos encontrar, entre otros:

- Manipulación, limpieza y formateado de datos: los datos pueden ser recibidos en formatos no estandarizados y requerir de una manipulación previa para darles el formato y la forma que sea compatible con las herramientas y procesos que se aplicarán sobre los datos más adelante. Una técnica útil en el caso del procesamiento de variables cualitativas es el *one-hot encoding* (DeepAi, s.f.) el cual permite representarlas mediante valores numéricos que optimizan su almacenamiento y permiten que se puedan aplicar sobre ellos algoritmos y análisis que no son, a priori, compatibles con variables cualitativas.

- Análisis estadísticos básicos: resulta útil situar los datos en un marco descriptivo estadístico haciendo uso de distintas métricas como podrían ser medias, medianas o varianzas, así como tratar de detectar las posibles distribuciones de probabilidad que se ajusten a la distribución encontrada en los datos.

- Detección de *outliers*: haciendo uso de los datos estadísticos obtenidos anteriormente se puede llevar a cabo la detección de datos anómalos (también conocidos como *outliers*). Estos datos anómalos pueden dar lugar a errores de análisis o entrenamiento, por lo que puede considerarse su eliminación.

- Detección de correlaciones: puede resultar de utilidad ejecutar análisis estadísticos que permitan detectar las posibles

correlaciones lineales existentes entre las distintas variables disponibles en los datos. Según el algoritmo o análisis a realizar, puede considerarse la eliminación de variables con alta correlación.

Durante la realización de estas tareas iniciales resultará útil el conocimiento experto mencionado anteriormente, gracias al cual pueden detectarse problemas que son propios del ámbito de aplicación y que pueden ser difíciles de detectar mediante los análisis estadísticos habituales.

Una vez realizada esta limpieza, formateo y análisis preliminar, el equipo debe plantearse qué variables deberán o no ser utilizadas en las siguientes fases de entrenamientos de modelos, esta vez desde un punto de vista MLOps de rendimiento computacional. Realizar este cribado es necesario ya que un número alto de variables explicativas, a pesar de poder resultar en un mejor y más preciso modelo, puede acarrear que el modelo resultante sea demasiado costoso de ejecutar en términos de recursos computacionales. Además, un alto número de variables implicará un mayor esfuerzo en mantener los flujos de datos de entrada y asegurar su adecuación.

A diferencia de los desarrollos tradicionales, plantear el desarrollo de modelos desde un enfoque MLOps implica tener en consideración, con bastante antelación en fases tempranas, cuestiones que serán relevantes en fases más avanzadas del proyecto. En este caso, por ejemplo, hemos estudiado cuestiones referentes al rendimiento computacional que no serán relevantes hasta fases más avanzadas del desarrollo, favoreciendo así la reducción de la deuda técnica del proyecto.

2.3 Experimentación, entrenamiento y evaluación de modelos

Una vez completo todo el conjunto de tareas de procesado de datos inicial se puede dar comienzo a la experimentación. Esta experimentación constituye una de las responsabilidades más importantes del científico de datos y será una de las fases en las que más tiempo y recursos se dedique por parte de estos profesionales.

En esta parte del desarrollo se explorarán cada uno de los elementos

detallados en el apartado 2.2, estos son, entre otros: datos de entrenamiento, métricas de rendimiento, algoritmos de Machine Learning e hiperparámetros.

Las posibles combinaciones de cada uno de estos elementos puede resultar en un conjunto de opciones de enorme tamaño, por lo que puede resultar necesario imponer restricciones sobre el tiempo o recursos computacionales empleados en esta fase experimental. Además, será útil establecer los umbrales de aceptación para las distintas métricas usadas para evaluación y poder así rechazar con rapidez aquellos resultados insatisfactorios.

Cada resultado obtenido durante la experimentación debe ser documentado de tal forma que cada experimento pueda ser reproducido. Todos los experimentos que den lugar a conclusiones definitivas tendrán que ser reproducidos y reexaminados para verificar y justificar todas las decisiones tomadas.

Para poder cumplir con estas necesidades, existen herramientas y plataformas de Machine Learning que permiten la exploración de las distintas combinaciones de elementos (hiperparámetros, algoritmos, variables explicativas, etc.) y su versionado automático.

Entre las técnicas disponibles en esta fase de experimentación y entrenamiento, merece la pena hacer especial mención a una de las más importantes y generalizadas en este tipo de procesos: la validación cruzada. Este método es de utilidad para evaluar la calidad predictiva del modelo mediante una serie de estrategias orientadas a la obtención de subconjuntos de datos a partir del conjunto original. Con la validación cruzada *k-fold* ("Cross-validation (statistics)", 2021), por ejemplo, se rotan múltiples veces los subconjuntos que se usarán para entrenamiento y validación. De esta manera se puede entrenar y evaluar el modelo alcanzando resultados que pueden optimizar la capacidad de generalización del modelo, evitando sobreajustes o falta de entrenamiento.

Dependiendo del ámbito de aplicación, es posible que sea necesario tener en cuenta consideraciones que estén relacionadas con requisitos de justicia o igualdad social. En este sentido, por ejemplo, puede ser necesario tomar precauciones para evitar que los modelos contengan sesgos que beneficien o perjudiquen ciertos sectores, identidades o

minorías de la sociedad.

2.4 Versionado y reproducibilidad del modelo ejecutable

En la fase anterior, tras la exploración y experimentación realizada por parte de los científicos de datos, se habrá obtenido un modelo definitivo que será usado en el resto de fases del proyecto. Este modelo estará contenido en un artefacto fruto de la ejecución del algoritmo de Machine Learning utilizado. Este artefacto, por sí solo, no es ejecutable y no se puede emplear tal cual como implementación en el despliegue definitivo que dará servicio a los usuarios finales. En cambio, debe diseñarse un código ejecutable que sea capaz de procesar los datos de entrada necesarios, darles el formato que el artefacto requiere para su ejecución, ejecutarlo y, finalmente, interpretar y procesar los resultados obtenidos para su almacenamiento o envío.

Este *script*, a su vez, contendrá los hiperparámetros escogidos correspondientes al artefacto y utilizará una serie de dependencias concretas para su ejecución.

Al igual que las necesidades de versionado que pudimos encontrar en el apartado anterior, esta implementación ejecutable del modelo también debe ser sometida a un control de versiones que permita explorar y revertir todos los cambios que ha sufrido durante su desarrollo. Para ello, ya existen herramientas de versionado de código muy extendidas y generalizadas que suplen esta necesidad, como puede ser Git (s.f.). Además, para el versionado de artefactos también existen soluciones disponibles que solventan estas necesidades y que se pueden encontrar ofrecidas en distintas plataformas de soluciones *cloud* para el desarrollo de *software* y Machine Learning.

Una vez completo el desarrollo del *script* en esta fase de desarrollo, este debe ser entregado a los miembros del equipo que, en sucesivas fases del proyecto, estarán encargados de desplegar este código en una infraestructura de producción. Sin embargo, este código es desarrollado en un entorno concreto y haciendo uso de unas dependencias específicas que deben ser reproducidos con exactitud allá donde se decida ejecutar el código.

Esta necesidad surge de la estrecha relación que existe entre los resultados del modelo y las dependencias usadas para el desarrollo del código ejecutable. Estas dependencias, que tomarán la forma de librerías o paquetes, podrán ser encontradas en distintas versiones, cada una de las cuales contará con una serie de características y errores concretos.

Debido a estas diferencias entre versiones, el mismo *script* del modelo ejecutado en dos entornos diferentes puede arrojar resultados totalmente distintos e inesperados con respecto a aquellos obtenidos durante la experimentación. Por tanto, además de todas las consideraciones tomadas con respecto al versionado de código y artefactos, se deberán realizar esfuerzos adicionales para el registro de las dependencias utilizadas y sus respectivas versiones. Esta información será utilizada en fases avanzadas de despliegue para garantizar un entorno de ejecución idéntico al usado durante la experimentación y desarrollo del modelo ejecutable.

Para cubrir esta necesidad existen herramientas de control de entornos que permiten el registro automatizado de toda la información requerida para la reproducibilidad del código.

2.5 Caso práctico

Como fue expuesto en el capítulo introductorio, este Trabajo de Fin de Grado propone el desarrollo de un caso práctico que, de forma paralela a la exploración teórica del MLOps, sirva de ejemplificación y puesta en práctica de los conceptos estudiados.

MLOps constituye una amplia disciplina que abarca numerosos principios enfocados en múltiples ámbitos de aplicación. Para cada uno de estos ámbitos, solamente un subconjunto de los principios aportados por MLOps serán relevantes.

En este caso, se abordará el desarrollo de un modelo diseñado para la predicción de consumos energéticos. En este desarrollo, por tanto, se podrá dar aplicación solo a aquel subconjunto de principios que sea de relevancia para este caso práctico concreto. Así, por ejemplo, consideraciones tales como las relacionadas con la igualdad o justicia social no tendrán cabida en este proyecto. En cambio, habrá que

hacer especial hincapié en, por ejemplo, la reproducibilidad del código y el versionado de dependencias.

Por otra parte, cada proyecto distinto se desarrollará en base a una consecución de fases única y específica, adaptándose en cada caso a la naturaleza y necesidades de cada proyecto.

Por tanto, en esta etapa inicial del proyecto, se dará comienzo al desarrollo del modelo tal y como ha sido expuesto teóricamente en este capítulo.

2.5.1 Herramientas utilizadas

Para el diseño de todo el código necesario en este apartado del proyecto, se emplearán los lenguajes de programación R (The R Foundation, s.f.) y Python (Python Software Foundation, 2021), así como diversas librerías habituales para estos lenguajes y estas actividades. Para el registro y reproducibilidad de dependencias se utilizará Renv (Ushey, 2021), sobre el cual se darán más detalles a continuación. Se utilizará RStudio (RStudio, 2021) como IDE (*Integrated Development Environment*) principal de trabajo.

2.5.2 Datos de entrenamiento

Los datos de medida eléctrica se proporcionan mediante un conjunto ficheros que cuentan con un formato estandarizado. La especificación de este formato es proporcionada por Red Eléctrica de España (Red Eléctrica de España, 2021), organismo encargado de la estandarización de este tipo de ficheros de intercambio de datos de medida eléctrica.

En concreto, el fichero que nos resultará de utilidad para este desarrollo se trata del F5D (Red Eléctrica de España, 2020), el cual detalla la información correspondiente a los consumos y vertidos eléctricos horarios por usuario.

Estos datos se recibirán adecuadamente anonimizados, cumpliendo así con los requerimientos de protección de datos. Estos datos fueron cedidos por una comercializadora eléctrica colaboradora.

2.5.3 Procesado de datos y análisis preliminar

En este caso, gracias a que los ficheros son recibidos con un formato estandarizado, las tareas de limpieza y procesado de datos inicial se simplifican considerablemente. Se diseñará una función que, dado un directorio de ficheros F5D, los lea, procese y cargue en memoria. Como parte del formateado de datos inicial, será necesaria la corrección del dato horario que identifica cada medida, transformando las cadenas de texto en objetos capaces de ser procesados por las librerías y herramientas utilizadas a continuación.

Una vez importados los datos con el formato correcto, se puede llevar a cabo un análisis básico preliminar que permita situar los datos en un marco estadístico descriptivo. En este caso, por ejemplo, se calculan las medias de los consumos medios anuales por usuario. En este análisis, además, se añade la mediana para poder representar los datos mediante un estadístico robusto. Este tipo de estadísticos proporcionan resultados que son menos sensibles a datos anómalos y, por tanto, más representativos.

year <dbl>	media_periodo_1 <dbl>	media_periodo_2 <dbl>	mediana_periodo_1 <dbl>	mediana_periodo_2 <dbl>
2019	1808475	1436267	1484128	1111388
2020	1906654	1495313	1551391	1157649

2 rows

Figura 2.2: Resultado del análisis de medias y medianas por periodos

2.5.4 Entrenamiento del modelo

La técnica de predicción utilizada en este proyecto hace uso de dos algoritmos de Machine Learning distintos, cuyo resultado se combinará para computar la predicción final. Estos algoritmos, basados en Random Forest y K-Means ("*k*-means clustering", 2021), producirán varios artefactos distintos tras el correspondiente entrenamiento de cada uno. Estos artefactos serán almacenados como archivos *.rds*, formato de archivo utilizado para almacenar objetos del entorno de trabajo de la sesión de R. Estos ficheros, tras su correspondiente evaluación y validación, constituirán el resultado final del trabajo llevado a cabo en esta fase del desarrollo y serán entregados a los miembros del equipo encargados de abordar las siguientes fases del proyecto.

Para este entrenamiento se dividió el conjunto de entrenamiento en dos subconjuntos siguiendo una estrategia de validación cruzada. El conjunto de datos se obtuvo haciendo uso de la función de lectura y procesado preliminar de datos definida anteriormente.

Figura 2.3: Artefactos procedentes del entrenamiento de modelos

Además de los datos de entrenamiento procedentes de los registros de medida eléctrica, se utilizaron datos meteorológicos como variables explicativas adicionales. El método de obtención de estos datos será detallado más adelante.

El proceso de entrenamiento se llevó a cabo siguiendo el método de entrenamiento estándar correspondiente a cada uno de estos algoritmos.

2.5.5 Evaluación del modelo

Una vez obtenidos los artefactos resultantes del proceso de entrenamiento, estos serán sometidos a evaluación. En este proceso de evaluación se emplearán métricas de rendimiento y sus correspondientes umbrales de aceptación. En este caso, la métrica de error empleada será el MAPE, métrica de error correspondiente a la media de los errores porcentuales.

$$MAPE = \frac{1}{n} \sum \frac{|error|}{consumo}$$

Figura 2.4: Fórmula del MAPE

En este tipo de evaluaciones, el científico de datos puede, habitualmente, elaborar un informe que sirva para comunicar los

datos al resto de miembros del equipo.

Este proceso de entrenamiento y evaluación, que forma parte de las tareas experimentales, puede sufrir tantas iteraciones como sean necesarias mientras los resultados no sean satisfactorios.

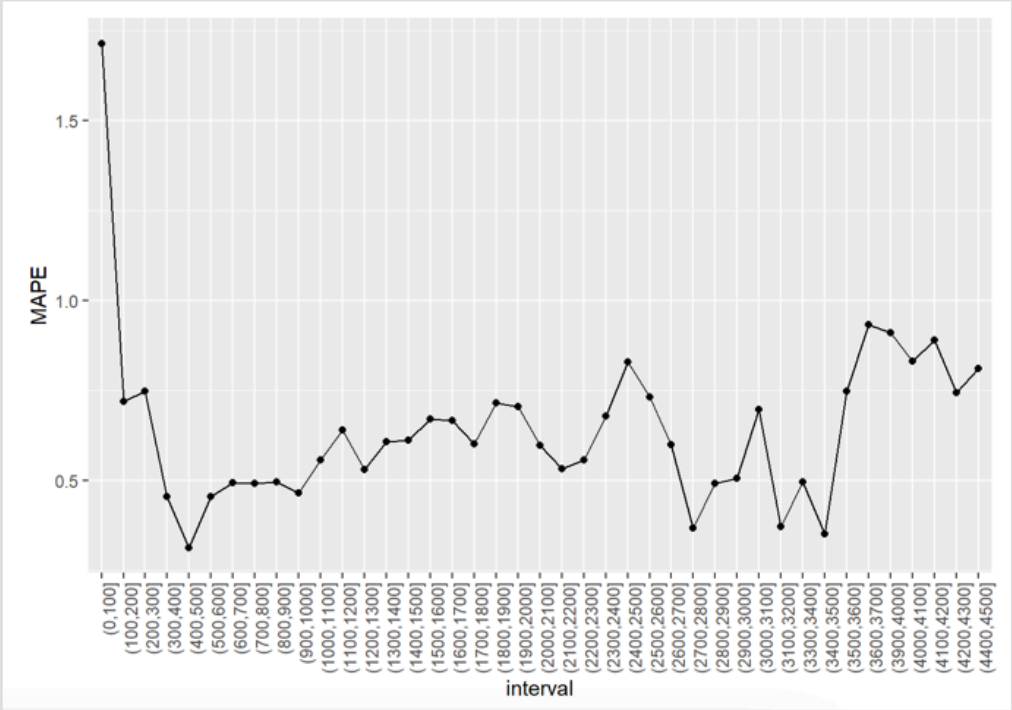

Figura 2.5: Resultado del análisis de errores según intervalos de consumo

2.5.6 Implementación ejecutable del modelo

Una vez obtenidos los artefactos definitivos resultantes del entrenamiento y evaluación, estos se usarán para diseñar el código ejecutable que, haciendo uso de estos artefactos, podrá computar las predicciones requeridas. Este código ejecutable será desplegado en la infraestructura de producción para dar finalmente servicio.

Este *script* orquestará todos los procesos encargados de la recolección de datos de entrada de medida eléctrica y meteorológicos y, haciendo uso de estos, ejecutará el modelo. Además, tras la ejecución del modelo, almacenará los resultados correctamente formateados en la base de datos del sistema. Para ello, se diseñarán 4 módulos distintos constituidos por el *script* principal de orquestación, el sistema de

descarga de datos meteorológicos, el sistema de consulta y almacenado en base datos y el modelo ejecutable:

- Código principal: este fichero será el punto de entrada al sistema y el encargado de orquestar todos los módulos y procesos tal y como ha sido expuesto anteriormente. En este caso, se requerirá la ejecución diaria del modelo para obtener las predicciones del día posterior.

- Módulo de descarga de datos meteorológicos: este componente del sistema será de utilidad para llevar a cabo la descarga de todos los datos meteorológicos necesarios a través del proveedor de datos meteorológicos seleccionado. En este caso, el proveedor usado para la implementación es OpenWeather (2021), el cual proporciona los servicios que este proyecto requiere.

- Módulo de acceso a base de datos: los datos de históricos de consumo, predicciones de consumo y predicciones meteorológicas serán almacenados en una base de datos desplegada específicamente para este proyecto. El acceso a esta base de datos se realizará mediante las funciones definidas en este módulo, las cuales actuarán de interfaz para la manipulación y consulta. Los detalles sobre el despliegue, aprovisionamiento y estructura de esta base de datos será especificado en el capítulo 4.

- Módulo de ejecución del modelo: en este fichero se encontrará la función que computará las predicciones finales a partir de todos los datos de entrada y los artefactos entrenados. Los detalles de esta implementación no son relevantes para el alcance y objeto de este Trabajo de Fin de Grado.

2.5.7 Registro y versionado de dependencias

Como hemos tenido ocasión de estudiar en el apartado teórico de este capítulo, resulta de gran importancia realizar esfuerzos por replicar con exactitud los entornos de ejecución allá donde el código vaya a ser utilizado. Para poder suplir esta necesidad utilizaremos Renv, una herramienta que, de forma automatizada, realizará el registro de las dependencias utilizadas y sus respectivas versiones. Renv almacenará este registro en una serie de ficheros que contendrán, de forma estructurada, las dependencias

correspondientes a cada lenguaje y herramienta utilizados.

2.5.8 Posibles desarrollos adicionales

Los artefactos obtenidos como resultado del entrenamiento de los modelos de Machine Learning pueden ser almacenados en una plataforma de control de versiones de artefactos de este tipo. Esto será de utilidad para el registro de todos los artefactos desarrollados y será de utilidad para permitir revertir artefactos a versiones anteriores.

Además, gracias al almacenamiento de estos artefactos en una plataforma específica, será posible la carga y descarga de estos ficheros sin necesidad de gestionar el almacenamiento de estos artefactos desde el entorno de desarrollo.

3

Preparación para producción

Una vez completa la primera fase de entrenamiento y obtenido un modelo final, este debe ser sujeto a una serie de consideraciones y procesos adicionales antes de poder ser desplegado en producción. Este capítulo, por tanto, abordará esta segunda fase del desarrollo del proyecto.

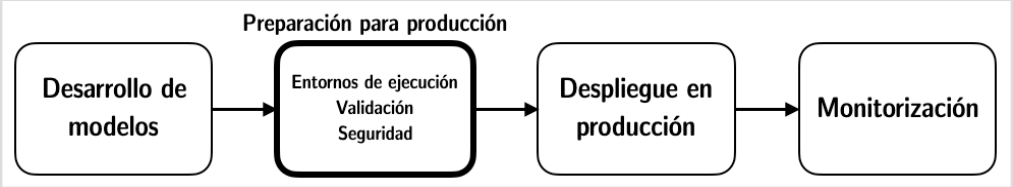

Figura 3.1: Esquema del ciclo de vida MLOps con preparación para producción ampliado

Se introducirán las consideraciones referentes a los entornos de ejecución, la validación de modelos y la seguridad de modelos. Finalmente, se proseguirá con el desarrollo del caso práctico.

3.1 Entornos de ejecución

Como ya hemos podido discutir en el anterior capítulo, asegurar unos entornos de ejecución idénticos al entorno de desarrollo es un

requisito fundamental para el correcto funcionamiento de los modelos desarrollados.

Abordar este problema, dada su complejidad e importancia, requiere realizar esfuerzos en distintas fases del desarrollo del proyecto. En el capítulo anterior comenzamos a tomar en consideración la reproducibilidad de entornos mediante el registro de las dependencias utilizadas por el modelo y su código ejecutable; sin embargo, esta información por sí sola no es suficiente a la hora de reproducir un entorno de ejecución.

Los entornos de producción pueden tomar muchas formas distintas: sistemas empotrados, plataformas *cloud* de ciencia de datos, clústeres de computación distribuida, servidores propios de la organización, etc. Cada uno de ellos requerirá una configuración e instalación concreta que será específica para cada entorno. Además, cabe la posibilidad de que existan incompatibilidades entre las dependencias requeridas y la plataforma o sistema operativo utilizado en producción. En casos extremos, puede que incluso sea necesaria la refactorización completa del código en un lenguaje distinto para adaptarse al entorno de producción usado por la organización.

El despliegue de modelos puede volverse aún más complejo cuando, en una misma plataforma, debe darse ejecución a más de un modelo simultáneamente. En estos casos, los diferentes requerimientos de dependencias en sus distintas versiones pueden crear conflictos e incompatibilidades al no ser posible aislar distintas versiones de una misma dependencia del sistema.

En el pasado, esta compleja tarea ha consumido gran parte del tiempo y esfuerzo de los profesionales encargados de estos desarrollos; sin embargo, durante los últimos años, estamos asistiendo al surgimiento de tecnologías de contenerización ("OS-level virtualization", 2021) que pueden ser utilizadas para dar solución a este problema de forma fácil y efectiva.

Estas tecnologías, haciendo uso de los llamados "contenedores", permiten la encapsulación del código junto a todas sus dependencias en un artefacto que puede ser compartido y desplegado fácilmente y de forma aislada en prácticamente cualquier plataforma compatible con esta tecnología. Este artefacto puede contener los modelos de

Machine Learning y sus correspondientes artefactos, los ficheros de configuración, las dependencias de código y las dependencias del sistema que el proyecto requiera, listo para ser ejecutado sin necesidad de ningún paso extra de instalación o configuración. Además, al no tratarse de sistemas de virtualización, no existen penalizaciones de rendimiento computacional con su uso.

Entre las soluciones de contenerización existentes, Docker (2021a) es la tecnología de contenerización más extendida y generalizada en la actualidad. Docker permite la utilización de estos contenedores mediante la definición de imágenes Docker a partir de los ficheros Dockerfile (Docker, 2021b), los cuales permiten detallar, de forma declarativa y estructurada, toda la información necesaria para recrear un entorno de ejecución y todo su contenido.

3.2 Validación de modelos

Antes de ser desplegados en producción, estos modelos ejecutables deben ser sometidos a un proceso de validación previo que asegure su adecuación, corrección y buen funcionamiento.

Los modelos finales, a pesar de la evaluación de precisión y fiabilidad ejecutada en fases anteriores, pueden presentar errores de implementación y tener, en determinadas circunstancias, comportamientos no deseados. Dependiendo del ámbito de aplicación, estos errores pueden tener un gran impacto y suponer riesgos y daños de gran alcance. Estos riesgos pueden ser, por ejemplo, de tipo financiero, legales, de seguridad o de reputación, entre otros.

En estos casos de alto riesgo es necesario que el equipo completo tenga en consideración los riesgos a los que está expuesto el proyecto y se diseñe el proceso de validación acorde a la magnitud de estos. Además, a pesar de existir una fase dedicada exclusivamente a esta validación, los riesgos deben tenerse en consideración desde la fase inicial del desarrollo del proyecto, pudiendo así mitigar el riesgo en el mayor grado posible.

En el desarrollo general de *software* ya existen herramientas y metodologías para *quality assurance* (abreviado como QA y, menos frecuentemente, conocido como Aseguramiento de la Calidad en

español) ("Quality assurance", 2021); sin embargo, no existen herramientas de QA lo suficientemente maduradas para proyectos de Machine Learning. Por este motivo, este proceso puede resultar un poco más complejo cuando se trata de proyectos de este tipo.

MLOps propone una serie de técnicas para tratar de afrontar esta validación de modelos. Entre ellas, una de las más importantes se trata de la validación con conjuntos de datos sintéticos prediseñados. Siguiendo esta estrategia, se diseñarán diferentes conjuntos de datos preparados para realizar la evaluación de distintos aspectos concretos de la calidad del modelo. Por ejemplo, pueden diseñarse conjuntos de datos de validación con datos extremos o faltantes, de forma que pueda comprobarse que el código del modelo ejecutable es capaz de gestionar satisfactoriamente este tipo de escenarios excepcionales. Otro tipo de conjunto de datos de validación puede consistir en datos que simulen escenarios más realistas. Haciendo uso de estos puede ejecutarse el modelo y validar el resultado con distintas métricas de precisión o rendimiento computacional.

Este proceso puede realizarse de forma automatizada cada vez que se desarrolle una nueva versión del modelo. Esta validación automatizada formará parte del *pipeline* automatizado encargado de la contenerización del modelo y su despliegue. Estos *pipelines* de contenerización y despliegue automatizado se estudiarán con más detalle en el próximo capítulo.

3.3 Seguridad de modelos

Los modelos pueden ser víctimas de ataques maliciosos que tengan como objetivo obtener algún tipo de beneficio, esquivar medidas de seguridad o simplemente ocasionar daños de distinto tipo a la organización. Como cualquier otro tipo de desarrollo *software*, la implementación del modelo puede tener *bugs* o sufrir filtraciones de datos. Sin embargo, los modelos de Machine Learning se enfrentan a otra serie de ataques adicionales que hacen uso de las características concretas de los algoritmos de Machine Learning empleados.

Uno de estos tipos posibles de ataques son los ataques adversarios (Goodfellow et al., 2017). Este tipo de ataques, orientados principalmente a los modelos basados en Deep Learning, tratan de encontrar pequeñas variaciones en los datos de entrada de tal forma

que provoquen resultados erróneos e inesperados en los resultados del modelo.

Este tipo de ataques pueden tener lugar una vez que el modelo ya ha sido desplegado, pero también pueden ocurrir en el momento del entrenamiento. En este caso, un atacante toma control del sistema e introduce datos sintéticos con el objeto de dar lugar a un entrenamiento que presente un comportamiento deseado por el atacante. Los modelos de regresión logística, por ejemplo, son invulnerables a este tipo de ataques.

En el proceso de desarrollo de modelos basados en Machine Learning será necesario, por tanto, realizar esfuerzos por adoptar las buenas prácticas de seguridad informática generales y además, en función del tipo de modelo usado, tener en consideración los posibles riesgos de seguridad que puede acarrear su uso.

3.4 Caso práctico

Tras la finalización de la primera fase del proyecto en el capítulo anterior, obtuvimos unos artefactos fruto de la ejecución de los algoritmos de Machine Learning y un código ejecutable preparado para orquestar todos los procesos de recolección de datos, ejecución de modelos y almacenado y envío de resultados. Además, se utilizaron herramientas para el registro de las dependencias y versiones utilizadas.

Antes de poder ser desplegado en producción, este conjunto de código y artefactos Machine Learning debe ser encapsulado en una imagen o artefacto final que replique con exactitud la configuración y entorno de ejecución empleados durante el desarrollo. Este proceso se llevará a cabo siguiendo las consideraciones teóricas expuestas en este capítulo.

Finalmente, el artefacto final será sometido a las consideraciones de validación y seguridad que sean relevantes para este proyecto concreto.

3.4.1 Herramientas utilizadas

Para esta fase del proyecto, la herramienta de trabajo principal será

Docker. Esta herramienta está compuesta de utilidades de línea de comandos, interfaces gráficas de gestión de contenedores e imágenes y un servicio de sistema. En este caso, se diseñará una serie de imágenes haciendo uso de sendos Dockerfiles, estos se ejecutarán y verificarán haciendo uso de estas herramientas.

3.4.2 Diseño de imagen base

Los equipos de ciencia de datos encargados del desarrollo de estos proyectos emplearán un sistema operativo y una serie de dependencias que serán en mayor o menor medida comunes en la gran mayoría de los proyectos desarrollados por la organización. Por este motivo, una parte considerable de las dependencias empleadas en cada proyecto serán comunes en todos los entornos de desarrollo y podrán ser instaladas en una imagen base común. Los distintos proyectos utilizarán esta imagen base y añadirán sobre ella solamente las dependencias específicas de cada proyecto concreto.

De esta manera, el trabajo correspondiente al diseño y mantenimiento de las imágenes Docker se divide y puede realizarse un esfuerzo común por el mantenimiento y diseño de esta imagen base. Además, al disponer de una imagen prediseñada y con dependencias preinstaladas, el proceso de creación de las imágenes específicas de cada proyecto consumirá mucho menos tiempo y recursos computacionales. Este ahorro de tiempo y recursos será importante cuando estas imágenes se construyan como parte de los *pipelines* automatizados que se diseñarán en la siguiente fase del proyecto.

Como primer paso, por tanto, se llevará a cabo el diseño de una imagen base que contendrá las dependencias de sistema, lenguajes de programación y dependencias comunes usadas generalmente en este tipo de proyectos. Esta imagen contendrá las instalaciones correspondientes a los lenguajes R y Python y estará basada en el sistema operativo Ubuntu (Canonical, 2021), además se incluirán algunas dependencias básicas.

En este desarrollo se tuvieron en consideración las buenas prácticas referentes al diseño de imágenes Docker. Por ejemplo, se han realizado esfuerzos por reducir al mínimo el número de capas y por ordenar las capas de forma que se optimice el máximo

aprovechamiento de la *cache* de capas de Docker.

3.4.3 Diseño de la imagen del modelo ejecutable

Una vez definida una imagen base que contenga el sistema operativo, los lenguajes de programación y las dependencias básicas, este podrá ser usado para la definición de la imagen que contenga el modelo ejecutable:

```
FROM eu.gcr.io/project-code/r-python-base
WORKDIR /MCE
COPY . .
RUN R -e 'renv::restore()'
RUN pip install -r requirements.txt
CMD ["Rscript","main.R"]
```

Figura 3.2: Dockerfile correspondiente a la imagen del modelo ejecutable

Esta imagen será llamada MCE como siglas de Modelo de Consumos Energéticos.

Gracias al esfuerzo llevado a cabo en el diseño de la imagen base, la especificación de la imagen del modelo ejecutable puede simplificarse considerablemente. En este Dockerfile podemos observar que como imagen base está siendo usada la imagen base diseñada en el apartado anterior. A continuación, se realiza la copia del directorio completo del proyecto respetando toda su estructura, se ejecuta la instalación de las dependencias concretas del proyecto haciendo uso de los ficheros de registros de dependencias creados en el capítulo anterior y, finalmente, se establece como punto de entrada la ejecución del fichero principal de orquestación.

3.4.4 Seguridad del modelo

Como hemos podido estudiar en el apartado teórico de este capítulo, los modelos expuestos al público pueden ser objeto de una serie de ataques que aprovechen las características concretas del modelo de Machine Learning utilizado. Sin embargo, este proceso de predicción se ejecutará diariamente sin ningún tipo de conexión ni interacción directa con los usuarios finales. Los datos de entrada serán obtenidos de forma automatizada sin intervención de los usuarios, por lo que no podrán tener lugar ataques con datos de entrada sintéticos. Este

contenedor, además, estará protegido por las medidas de seguridad propias de la plataforma en la que será ejecutado. Estas medidas de seguridad permitirán, además, bloquear todas las conexiones de red entrantes al contenedor.

Por tanto, no existen medidas de seguridad adicionales que deban tomarse en consideración en esta fase del proyecto.

3.4.5 Posibles desarrollos adicionales

Como parte de las tareas de validación recomendadas en esta fase del proyecto, podría llevarse a cabo el diseño de una serie de conjuntos de datos sintéticos de validación que sirva para evaluar distintos aspectos de la calidad del modelo ejecutable. Estos datos se proveerían al modelo ejecutable como datos de entrada y se verificarían sus resultados.

Este proceso podría acometerse mediante la implementación de dos pasos consecutivos adicionales:

- El primer paso consistiría en la ejecución de la imagen del modelo ejecutable con una configuración de conexión correspondiente a la de una base de datos secundaria de validación. En esta base de datos de validación se consultarían los conjuntos de datos de validación y se almacenarían los resultados correspondientes a la ejecución del modelo.

- El segundo paso consistiría en la ejecución de un *script* que, conectado a la misma base de datos de validación, verificase los resultados del modelo haciendo uso de distintas métricas y umbrales de aceptación.

Mediante este procedimiento solo sería necesario especificar, mediante las variables de entorno correspondientes, nuevos datos de conexión y, por tanto, no habría necesidad de hacer modificaciones al código del modelo ejecutable.

El diseño de estos conjuntos de datos sintéticos supone un esfuerzo que excede el objeto y alcance de este Trabajo de Fin de Grado.

4

Despliegue en producción

Durante todas las fases de desarrollo expuestas a lo largo de los capítulos anteriores, se han estudiado las consideraciones que los equipos de ciencia de datos a cargo de los proyectos de Machine Learning deben tomar para garantizar un despliegue exitoso del modelo en producción. En este capítulo se explorará esta fase de puesta en producción según la metodología MLOps, en la que finalmente se emplearán los resultados de todos los esfuerzos llevados a cabo en fases anteriores.

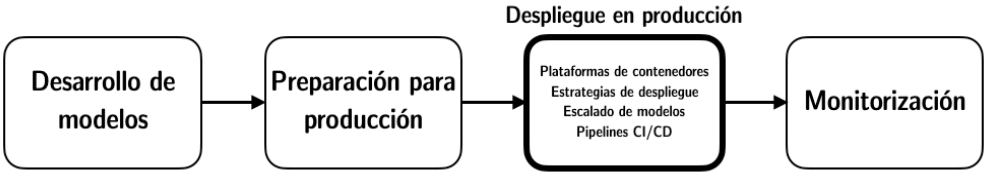

Figura 4.1: Esquema del ciclo de vida MLOps con despliegue en producción ampliado

Finalmente, este capítulo abordará la tercera fase de desarrollo del caso práctico, en la que se aplicarán los conceptos relevantes estudiados en este capítulo.

4.1 Plataformas de ejecución de contenedores

Como hemos tenido ocasión de estudiar en el anterior capítulo, a pesar de la existencia de algunas soluciones y plataformas para el despliegue de modelos de Machine Learning, la tecnología de contenedores se presenta como una solución fácil y efectiva capaz de solventar de forma íntegra los problemas relacionados con la reproducibilidad de entornos y el despliegue e instalación de *software*.

En los proyectos de Machine Learning que hagan uso de esta tecnología, se llevará a cabo un esfuerzo para el diseño de una serie de imágenes que, finalmente, serán ejecutadas mediante contenedores en una plataforma de producción.

Esta plataforma de producción debe ser capaz de orquestar y ejecutar estos contenedores. Además, será necesario garantizar que los procesos desplegados estén siempre disponibles y suficientemente escalados.

Kubernetes (2021a) es una plataforma de orquestación de contenedores *open-source* que se ha establecido como la solución por defecto elegida para llevar a cabo el despliegue de aplicaciones contenerizadas. Esta plataforma simplifica enormemente este proceso y aporta muchas otras ventajas adicionales.

Entre estas ventajas, Kubernetes es capaz de monitorizar el estado de todos los contenedores desplegados y reiniciar aquellos que hayan alcanzado algún estado excepcional. Además, permite escalar contenedores de forma automática en los momentos en el que estos experimenten un aumento de la demanda de recursos computacionales.

Gracias a este conjunto de características y funcionalidades, Kubernetes se presenta como una solución idónea para suplir todas las necesidades de despliegue, escalado y monitorización de modelos de Machine Learning contenerizados.

4.2 Estrategias de despliegue

Cuando se desarrollan nuevas versiones de los modelos de predicción,

será necesario desplegarlos en producción y sustituirlos por los modelos actualmente en servicio. Realizar este proceso requiere de una planificación y estrategia concretos.

Los modelos de predicción pueden dar servicio de dos formas distintas:

- En tiempo real: los modelos que den servicio en tiempo real responderán a peticiones recibidas por parte de usuarios finales o de otros procesos automatizados, los cuales solicitarán predicciones sobre datos de entrada concretos. El modelo, por tanto, deberá ejecutarse y dar una respuesta en el menor tiempo posible.

- Por lotes: estos modelos realizarán la predicción de un gran conjunto de datos con una frecuencia preestablecida. Estos modelos no responderán en tiempo real a ninguna petición y solo estarán en ejecución durante el intervalo de tiempo que dure la predicción del lote de datos de entrada.

Durante el proceso de actualización de versiones será imprescindible evitar al máximo la pérdida de servicio. Esto será de especial importancia cuando se trate de modelos en tiempo real. Además, será necesario llevar a cabo un último proceso de validación estudiando el comportamiento del modelo cuando es ejecutado con los datos reales del entorno de producción.

Entre las estrategias de despliegue disponibles podemos destacar dos de las más importantes y utilizadas:

- *Canary deployments*: en este tipo de despliegues, la nueva versión del modelo se despliega junto a la versión antigua en el entorno de producción. A continuación, un pequeño porcentaje de la carga de trabajo es redirigida al nuevo modelo (Fernandez, 2020). De esta manera, es posible validar la estabilidad y corrección de la nueva versión con datos reales de producción. Una vez validado, toda la carga de trabajo se apunta al nuevo modelo y el antiguo es eliminado.

- *Blue-green deployments*: en este caso, al igual que el anterior, el nuevo modelo se despliega junto al antiguo de forma simultánea

en el entorno de producción. Esta vez, la carga de trabajo no se divide, sino que es duplicada y procesada por ambos modelos a la vez. El modelo nuevo almacenará sus resultados en una colección de predicciones de validación donde podrán comprobarse sus resultados. Una vez validado el buen funcionamiento del nuevo modelo, sus resultados se almacenarán en la colección de predicciones normal de producción y el antiguo será eliminado (Red Hat, 2021).

Estas estrategias de despliegue permiten evitar pérdidas de servicio y, a la vez, verificar el buen funcionamiento de las nuevas versiones de los modelos.

Kubernetes es capaz de ejecutar estas estrategias de despliegue de forma automatizada sin ningún tipo de intervención manual.

4.3 Escalado de modelos

Conforme el número de peticiones o carga de trabajo de los modelos de predicción aumente, será necesario garantizar que estos cuenten con suficientes recursos computacionales para poder ejecutar las predicciones en el tiempo requerido.

En el caso de modelos en tiempo real, este será un requisito imprescindible para poder dar una respuesta rápida a cada petición. En estos casos, el contenedor del modelo puede replicarse tantas veces sea necesario para que la carga de trabajo pueda estar suficientemente repartida. En otros casos, también pueden aumentarse los recursos computacionales asignados a un contenedor concreto sin necesidad de replicarlo.

Kubernetes también puede realizar este tipo de escalados de forma totalmente automatizada.

4.4 *Pipelines* CI/CD

Una pieza clave de la metodología MLOps consiste en la automatización, en la mayor medida de lo posible, de todos los procesos que puedan ser automatizados. Este esfuerzo de automatización tiene como objetivo liberar a los equipos de ciencia de datos de tareas que no estén relacionadas con el trabajo

estadístico y de desarrollo de modelos. Así, se realizarán esfuerzos por automatizar todas las tareas de contenerización, validación y despliegue.

CI/CD ("CI/CD", 2021) es un acrónimo generalizado que hace referencia a *Continuous Integration* y *Continuous Delivery* (o, en ocasiones, *Continuous Deployment*). Estos son conceptos relacionados con los procesos de integración, validación y despliegue automatizados del *software* general. Estos conceptos también pueden ser aplicados a los proyectos de Machine Learning.

Existen múltiples soluciones para el desarrollo de *pipelines* CI/CD ofrecidas en las distintas plataformas de servicios *cloud*, así como distintas herramientas de software libre. Todas ellas pueden ser utilizadas con el propósito perseguido en los proyectos de Machine Learning.

Si, una vez llegados a esta fase del proyecto, se han seguido las directrices metodológicas de MLOps, la construcción de los *pipelines* finales resultará una tarea sencilla en la que se pondrán en común todos los esfuerzos realizados a lo largo de todas las fases del proyecto.

4.5 Caso práctico

Como fruto de todos los desarrollos llevados a cabo hasta ahora, hemos obtenido una imagen Docker que contiene el código del modelo ejecutable, los artefactos correspondientes al entrenamiento de los modelos, las dependencias y toda la estructura de ficheros necesaria para su ejecución. En esta fase del proyecto, esta imagen se desplegará en un entorno de producción para dar finalmente servicio.

Esta tarea requerirá el aprovisionamiento de la base de datos del sistema y el clúster de Kubernetes necesario para la ejecución del modelo. Finalmente, se diseñarán *pipelines* automatizados que realicen las tareas de contenerización y despliegue.

4.5.1 Herramientas utilizadas

La base de datos utilizada en este proyecto es MongoDB (2021a), la cual será desplegada mediante la plataforma *cloud* MongoDB Atlas

(MongoDB, 2021b). Esta base de datos es de tipo NoSQL y, por tanto, estará organizada mediante colecciones de documentos. Este tipo de base de datos ofrece un buen rendimiento, gran flexibilidad y facilidad de uso. Por estas razones, esta base de datos constituye una buena solución para suplir las necesidades de este proyecto.

La plataforma de orquestación de contenedores utilizada en este proyecto es Kubernetes. El clúster será desplegado mediante la plataforma *cloud* Google Kubernetes Engine (Google, 2021a).

La implementación de los pipelines automatizados se realizará mediante la plataforma *cloud* Google Cloud Build (Google, 2021b).

El almacenamiento de todas las imágenes Docker generadas se almacenará en un repositorio de imágenes del servicio Google Container Registry (Google, 2021c).

4.5.2 Aprovisionamiento de la infraestructura

En primer lugar se llevará a cabo el aprovisionamiento de la base de datos. Este se realizará mediante el procedimiento estándar indicado en la documentación de la plataforma.

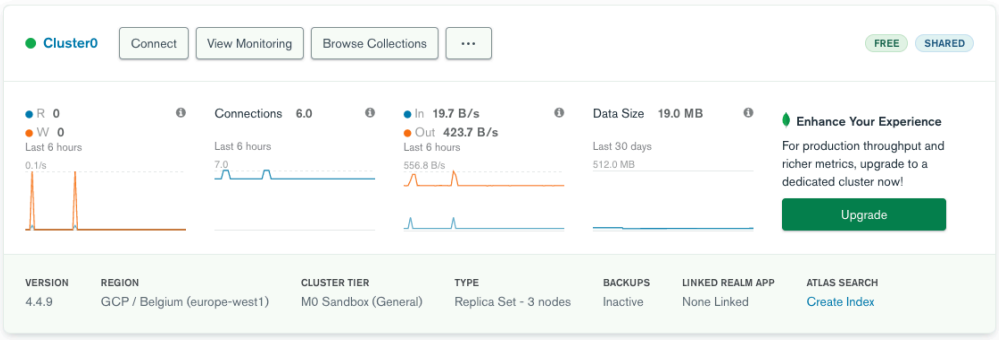

Figura 4.2: Despliegue de la base de datos mostrado desde la consola de MongoDB Atlas

A continuación se llevará a cabo el aprovisionamiento del clúster de Kubernetes. En este caso, el tipo de clúster escogido es "Autopilot" (Google, 2021d) el cual realiza la gestión automática de la infraestructura hardware subyacente. Este se realizará mediante el procedimiento estándar indicado en la documentación de la plataforma.

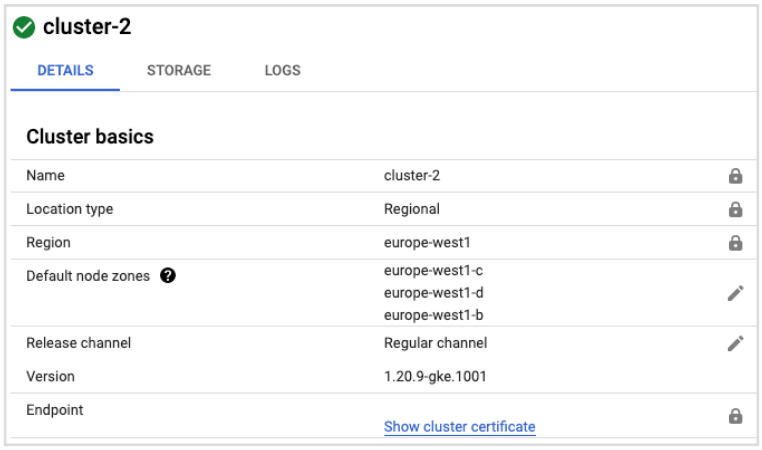

Figura 4.3: Despliegue del clúster de Kubernetes mostrado desde la consola de Google Kubernetes Engine

4.5.3 Despliegue blue-green

La estrategia de despliegue escogida para este proyecto es el despliegue *blue-green*. Este modelo de predicción se ejecutará por lotes de forma preprogramada. Esta estrategia de despliegue, por tanto, será adecuada para este proyecto concreto, tal y como ha sido estudiado en el apartado teórico de este capítulo.

Para ello, se diseñarán dos despliegues distintos mediante sendos objetos CronJob (Kubernetes, 2021b). Estos objetos Kubernetes permiten definir ejecuciones programadas de una imagen concreta. En este caso, se programará el modelo para ejecutarse diariamente a las 11:00 horas.

El primero de los objetos CronJob corresponderá al despliegue del modelo en producción (mostrado en la figura 4.4). Este modelo consultará los históricos de consumo de los usuarios desde la base de datos de producción y almacenará los resultados de la ejecución del modelo en una colección de predicciones de producción.

El segundo objeto CronJob será el correspondiente al modelo de preproducción, este será el modelo que se encontrará en fase de validación (mostrado en la figura 4.5). Este modelo, de igual forma, consultará los históricos de consumo de los usuarios desde la base de datos de producción, sin embargo, almacenará los resultados en una

```yaml
apiVersion: batch/v1beta1
kind: CronJob
metadata:
  name: mce-prod
spec:
  schedule: "0 11 * * *"
  jobTemplate:
    spec:
      backoffLimit: 0
      template:
        spec:
          containers:
          - name: mce
            image: eu.gcr.io/project-code/mce
            env:
            - name: MONGO_USER
              value: "user"
            - name: MONGO_PASS
              value: "pass"
            - name: MONGO_HOST
              value: "host"
            - name: MONGO_DB
              value: "PROD"
            - name: MONGO_PRED_COL
              value: "predictions_prod"
            - name: MONGO_METEO_COL
              value: "meteo_forecasts_prod"
            imagePullPolicy: Always
            resources:
              requests:
                memory: 1G
                ephemeral-storage: 100k
                cpu: 10m
          restartPolicy: Never
```

Figura 4.4: Fichero "mce-prod.yaml" correspondiente al objeto CronJob del modelo de producción

colección de predicciones de preproducción. De esta forma, los resultados de la ejecución de este modelo no serán utilizados, pero podrán ser evaluados mediante las métricas oportunas.

En ambos ficheros *yaml* (YAML, s.f.) de definición de los objetos CronJob, se han especificado los datos de conexión a la base datos mediante las variables de entorno correspondientes. Además, se han especificado los recursos computacionales que deben ser asignados a cada contenedor.

```yaml
apiVersion: batch/v1beta1
kind: CronJob
metadata:
  name: mce-pre
spec:
  schedule: "0 11 * * *"
  jobTemplate:
    spec:
      backoffLimit: 0
      template:
        spec:
          containers:
          - name: mce-pre
            image: eu.gcr.io/project-code/mce-pre
            env:
            - name: MONGO_USER
              value: "user"
            - name: MONGO_PASS
              value: "pass"
            - name: MONGO_HOST
              value: "host"
            - name: MONGO_DB
              value: "PROD"
            - name: MONGO_PRED_COL
              value: "predictions_pre"
            - name: MONGO_METEO_COL
              value: "meteo_forecasts_pre"
            imagePullPolicy: Always
            resources:
              requests:
                memory: 1G
                ephemeral-storage: 100k
                cpu: 10m
          restartPolicy: Never
```

Figura 4.5: Fichero "mce-pre.yaml" correspondiente al objeto CronJob del modelo de preproducción

Una vez el modelo de preproducción sea validado, podrá ser desplegado como modelo de producción usando la definición del objeto CronJob correspondiente.

4.5.4 Diseño de pipelines automatizados

Como paso final del desarrollo se implementarán los *pipelines* que automatizarán el proceso de contenerización y despliegue.

Los *pipelines* de Google Cloud Build se definen mediante ficheros *yaml* donde se especifican los distintos pasos que deben ser ejecutados.

En este caso se diseñarán tres *pipelines* distintos:

- *Pipeline* del modelo de producción: este pipeline estará encargado de la creación de la imagen del modelo a partir del Dockerfile definido en el apartado 3.4.3. Consta de dos pasos: un primer paso de construcción y almacenamiento de la imagen y un segundo paso de despliegue en el clúster de Kubernetes. Este segundo paso de despliegue usará el fichero *yaml* del objeto CronJob definido en el apartado anterior.

```yaml
steps:
- name: 'gcr.io/kaniko-project/executor:latest'
  args:
  - --destination=eu.gcr.io/$PROJECT_ID/mce
  - --cache=true
- name: "gcr.io/cloud-builders/gke-deploy"
  args:
  - run
  - --filename=mce-prod.yaml
  - --image=eu.gcr.io/$PROJECT_ID/mce
  - --location=europe-west1
  - --cluster=cluster-2
```

Figura 4.6: fichero "cloudbuild-prod.yaml" correspondiente al *pipeline* de contenerización y despliegue del modelo de producción

- *Pipeline* del modelo de preproducción: este pipeline será idéntico al anterior pero se especifica un nombre distinto de imagen y se especifica el fichero del objeto CronJob correspondiente al despliegue de preproducción (Figura 4.7).

- *Pipeline* de la imagen base: se proporciona un pipeline para la contenerización y almacenamiento automáticos de nuevas versiones de la imagen base (Figura 4.8).

Todas estas imágenes serán almacenadas automáticamente en un

```
steps:
- name: 'gcr.io/kaniko-project/executor:latest'
  args:
  - --destination=eu.gcr.io/$PROJECT_ID/mce-pre
  - --cache=true
- name: "gcr.io/cloud-builders/gke-deploy"
  args:
  - run
  - --filename=mce-pre.yaml
  - --image=eu.gcr.io/$PROJECT_ID/mce-pre
  - --location=europe-west1
  - --cluster=cluster-2
```

Figura 4.7: fichero "cloudbuild-pre.yaml" correspondiente al *pipeline* de contenerización y despliegue del modelo de preproducción

```
steps:
- name: 'gcr.io/kaniko-project/executor:latest'
  args:
  - --destination=eu.gcr.io/$PROJECT_ID/r-python-base
  - --cache=true
```

Figura 4.8: fichero "cloudbuild.yaml" correspondiente al *pipeline* de contenerización y almacenamiento de la imagen base

repositorio de Google Container Registry.

Como se puede apreciar, la definición de estos pipelines resulta muy simple y concisa. Esta simplicidad ha sido lograda gracias a todos los esfuerzos de contenerización realizados a lo largo de todas las fases del proyecto.

Estos pipelines podrán ser ejecutados fácilmente mediante la utilidad de linea de comandos (Google, 2021e) proporcionada por Google Cloud Build.

4.5.5 Posibles desarrollos adicionales

- Estos *pipelines,* tal y como han sido diseñados en este proyecto, se ejecutan de forma manual desde el entorno de desarrollo. Sin embargo, estos *pipelines* pueden ser configurados para ser

ejecutados de forma automática cada vez que se detecte un nuevo *push* de código en el repositorio Git utilizado por el proyecto. Para poder configurar esta ejecución automática del *pipeline* es necesario haber efectuado previamente el almacenamiento de los artefactos Machine Learning en un sistema de versionado de artefactos, tal y como fue indicado en el apartado 2.5.8, ya que estos artefactos no pueden ser almacenados en un repositorio de código estándar.

- Las variables de entorno han sido proporcionadas a los contenedores mediante su especificación explícita en los ficheros *yaml* de cada objeto Kubernetes. Sin embargo, estas variables de entorno pueden ser especificadas de forma indirecta mediante un servicio de gestión de credenciales. Esto es conveniente para que estas credenciales no sean cargadas en los repositorios de código junto a los ficheros *yaml.*

- Como se indicó en el apartado 4.3, es importante asegurar un escalado adecuado de todos los procesos desplegados. Este escalado podría realizarse mediante las funcionalidades de escalado automático propias de Kubernetes.

- El proceso de validación expuesto en el apartado 3.4.5 podría ser incluido como un par de pasos adicionales en el *pipeline* de contenerización y despliegue diseñado anteriormente.

5

Monitorización de modelos

Una vez los modelos de Machine Learning han sido desplegados en producción, los proyectos alcanzan la cuarta y última fase del ciclo de vida de los proyectos basados en Machine Learning.

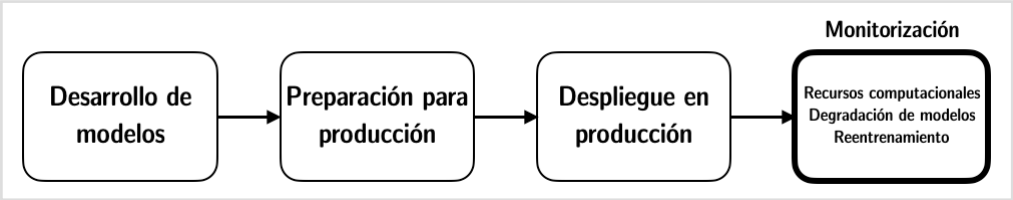

Figura 5.1: Esquema del ciclo de vida MLOps con monitorización ampliado

En esta fase del proyecto se realizará una monitorización de los modelos desplegados en producción para verificar, de forma continua, que el rendimiento y la calidad del modelo permanecen dentro de los umbrales de aceptación. MLOps proporciona una serie de recomendaciones y directrices enfocadas en el abordaje de estas tareas.

5.1 Monitorización de recursos

Como cualquier otro proceso software desplegado en producción, los modelos de Machine Learning deben ser sujetos a una monitorización

de su correcta ejecución y de su empleo de recursos computacionales.

En este sentido, será necesario monitorizar el uso de CPU, RAM, red y espacio de disco. Además se deberá comprobar que el proceso sigue en ejecución sin haber alcanzado ningún estado excepcional. Adicionalmente, según el tipo de modelo, será necesario verificar que este puede dar respuesta en el tiempo requerido.

Si los modelos han sido desplegados haciendo uso de Kubernetes, este tipo de monitorización puede llevarse a cabo mediante las funcionalidades propias de esta plataforma. Kubernetes ofrece facilidades para monitorizar los recursos computacionales de cada contenedor desplegado y actuar en el caso de que este alcance estados excepcionales o de error.

Además, será necesario monitorizar los *logs* producidos por el código en ejecución. Estos *logs* informan sobre el estado y las operaciones efectuadas por el modelo y son de utilidad para auditar su comportamiento. Kubernetes dispone de funcionalidades para la agregación y monitorización de los *logs* producidos por los contenedores.

5.2 Degradación de modelos

Además de la monitorización básica de recursos expuesta en el apartado anterior, los modelos de Machine Learning requieren un tipo de monitorización especial para la verificación de su rendimiento y calidad como modelo predictivo.

Esta monitorización es necesaria ya que, como fue estudiado en el apartado 2.1, los modelos tratan de representar fenómenos del mundo físico que son dinámicos y variables en el tiempo. Esta circunstancia hace que los modelos pierdan capacidad predictiva conforme los datos de entrada se van diferenciando progresivamente con respecto a los datos usados para el entrenamiento.

Existen, a grandes rasgos, dos tipos de ámbitos de aplicación en los que se empleen modelos de Machine Learning:

- En el primero de ellos, es fácil y rápido conocer los resultados correctos que el modelo debería haber predicho. Por ejemplo,

en el caso de un modelo encargado de predecir los consumos eléctricos del día de mañana, los valores correctos se conocerán al día siguiente, justo cuando se efectúen las medidas eléctricas correspondientes al consumo real de los usuarios.

- En el segundo ámbito ocurre justo lo contrario: resulta muy difícil o lento obtener los resultados correctos. Un ejemplo de este ámbito de aplicación puede ser la predicción del fraude en las transacciones efectuadas en algún tipo de sistema de intercambio de divisas. En este caso, la comprobación de que las transacciones sospechosas sean fraude o no consiste en un largo y lento proceso manual.

Para la evaluación del rendimiento predictivo del modelo, el primer caso resulta ser el más favorable. En estos casos pueden computarse los errores cometidos mediante las métricas más adecuadas y, a partir de estos resultados, alcanzar conclusiones sobre su adecuación.

En el segundo caso, esta tarea se vuelve más compleja. Para realizar este tipo de evaluación pueden utilizarse una serie técnicas estadísticas que sirvan para detectar la variación que los datos de entrada pueden haber sufrido con respecto a los de entrenamiento. Entre ellas, las más básicas consistirán en el estudio descriptivo de los datos mediante el cómputo de una serie de estadísticos y el estudio de su distribución. Entre las más complejas podemos encontrar técnicas como el test de Kolmogorov-Smirnov ("Kolmogorov-Smirnov test", 2021) o el test Chi-cuadrado ("Chi-squared test", 2021).

Estos tests podrán realizarse de forma automatizada con la frecuencia que el proyecto requiera. En estos tests automatizados, además, podrán configurarse alertas que notifiquen a los científicos de datos cuando alguna de las métricas no se encuentre dentro de los umbrales de aceptación.

5.3 Reentrenamiento de modelos

En el momento en el que, como parte de esta monitorización, se detecte la degradación del modelo, el equipo encargado de su desarrollo puede plantearse volver a entrenar el modelo con datos más recientes.

Esta decisión dependerá del caso concreto y del ámbito de aplicación. Por ejemplo, áreas como la ciberseguridad requieren un reentrenamiento de modelos muy frecuente. En otras aplicaciones, como en el caso de modelos físicos, estos son más estables y requieren menos reentrenamiento.

Además, los equipos deberán valorar si el coste asociado al esfuerzo del reentrenamiento merecerá la pena con respecto a la posible mejora obtenida en el modelo.

En algunos casos, la decisión de reentrenar los modelos por parte del equipo estará limitada por la cantidad de datos de entrenamiento disponibles. Por tanto, este reentrenamiento no será posible por la falta de datos de entrenamiento recolectados hasta ese momento.

Existen aplicaciones para las cuales puede resultar conveniente el diseño de sistemas de reentrenamiento automático. Estos sistemas harán usos de algoritmos de Machine Learning que podrán reentrenarse a sí mismos periódicamente. Estos algoritmos, sin embargo, pueden ser muy costosos de configurar y mantener. En estos casos, las alertas emitidas por los sistemas de evaluación automáticos expuestos en el apartado anterior podrán ser utilizados para activar el proceso de reentrenamiento automáticamente.

Las nuevas versiones de los modelos obtenidas como resultado del reentrenamiento se desplegarán en producción siguiendo alguna de las estrategias de despliegue expuestas en el apartado 4.2 del anterior capítulo.

5.4 Caso práctico

En este punto del desarrollo, el proyecto ha alcanzado su cuarta y última fase. Contamos con un modelo de predicción fruto de los desarrollos correspondientes a todas las fases del proyecto abordadas hasta este punto y hemos llevado a cabo su despliegue exitoso en un entorno de producción. Ahora, este modelo de predicción puede ser sometido a los procesos de monitorización estudiados en este capítulo.

5.4.1 Monitorización de recursos

Gracias a las funcionalidades de monitorización de recursos disponibles en Kubernetes, estas tareas de monitorización se simplifican considerablemente. En este caso, la consola de Google Kubernetes Engine nos proporciona información sobre el uso de CPU, RAM y espacio de disco efectuado por el contenedor del modelo. Esta información es presentada mediante una serie de gráficas interactivas.

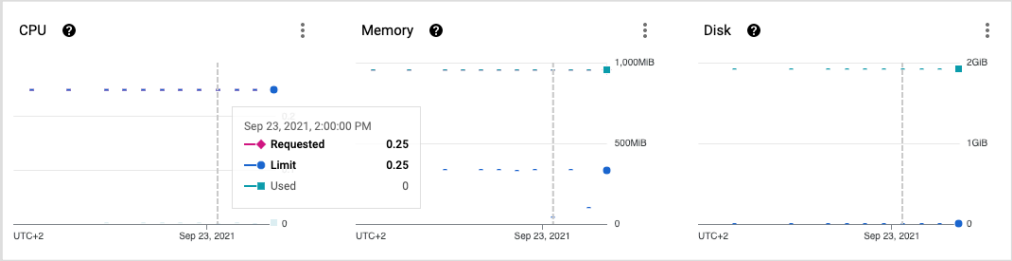

Figura 5.2: gráficas de monitorización de recursos computacionales mostradas desde la consola de Google Kubernetes Engine.

Además, desde esta consola también podemos acceder a los *logs* producidos por el contenedor durante su ejecución.

Figura 5.3: *logs* del contenedor del modelo mostrados desde la consola de Google Kubernetes Engine

Gracias a estas herramientas de monitorización podrá verificarse el correcto funcionamiento del modelo y su uso de recursos. Además, gracias a la agregación de *logs* podrá llevarse a cabo la auditoría de las operaciones llevadas a cabo por el modelo.

5.4.2 Posibles desarrollos adicionales

En este proyecto concreto es posible conocer los valores correctos de forma diaria y solo con un día de retraso, por tanto, sería posible abordar una monitorización de la calidad predictiva del modelo mediante métricas de error tal y como ha sido expuesto en el apartado 5.2.

Para llevar a cabo esta monitorización podría diseñarse un *script* que, con una frecuencia determinada, consulte desde la base de datos los datos de predicción y de históricos de consumo y realice el cómputo de distintas métricas de error. Podrían configurarse, además, alertas que notifiquen en el caso de que se obtengan valores inesperados. Este *script* de monitorización podrían ser desplegado mediante un objeto CronJob en el clúster de Kubernetes.

6

Conclusiones

En este capítulo final se tratará de sintetizar todos los conceptos abordados a lo largo de los capítulos elaborados a lo largo de este Trabajo de Fin de Grado. Finalmente se intentarán concluir los resultados obtenidos como parte del desarrollo del caso práctico abordado en este proyecto.

6.1 Conclusiones

Esta exploración teórica del MLOps como metodología de trabajo en los proyectos basados en Machine Learning ha servido para sintetizar y agregar todos los principios, directrices y buenas prácticas que son de aplicación en cada fase del desarrollo de este tipo de proyectos. Así, se ha establecido una división en fases concreta y una consecución temporal para su desarrollo. Este conjunto de organización, temporización y directrices ha permitido exponer un método para el abordaje de este tipo de proyectos de tal forma que se faciliten los despliegues, se minimicen los riesgos, se reduzca la deuda técnica, se garantice la seguridad informática, se reduzcan las pérdidas de servicio y se agilice la coordinación y colaboración entre los miembros del equipo encargados de estos desarrollos.

A su vez, de forma paralela, el desarrollo del caso práctico ha permitido poner en práctica y servir de ejemplificación para aquellos conceptos y directrices metodológicas relevantes para el proyecto propuesto.

6.2 Líneas futuras

Como parte de la exposición del caso práctico correspondiente a cada fase del proyecto, se han incluido secciones en las que se han sugerido posibles desarrollos adicionales que podrían llevarse a cabo para dar compleción y mejorar los trabajos llevados a cabo.

Por tanto, a lo largo de la exposición del desarrollo del caso práctico llevada a cabo como parte de cada capítulo, se han incluido propuestas de líneas futuras y posibles desarrollos que podrían llevarse a cabo.

Estos desarrollos adicionales terminarían de abordar las recomendaciones y directrices dictadas por MLOps para el desarrollo de este caso práctico concreto.

Referencias

Atlassian (s.f.). *DevOps: Breaking the development-operations barrier.* https://www.atlassian.com/devops

Canonical (2021). *The story of Ubuntu.* https://ubuntu.com/about

Chi-squared test (2021, 24 de agosto). En *Wikipedia.* https://en.wikipedia.org/wiki/Chi-squared_test

CI/CD (2021, 22 de septiembre). En *Wikipedia.* https://en.wikipedia.org/wiki/CI/CD

Cross-validation (statistics) (2021, 16 de julio). En *Wikipedia.* https://en.wikipedia.org/wiki/Cross-validation_(statistics)

Deep learning (2021, 20 de septiembre). En *Wikipedia.* https://en.wikipedia.org/wiki/Deep_learning

DeepAi (s.f.). *One Hot Encoding.* https://deepai.org/machine-learning-glossary-and-terms/one-hot-encoding

Docker (2021a). *Docker overview.* https://docs.docker.com/get-started/overview/

Docker (2021b). *Dockerfile reference.* https://docs.docker.com/engine/reference/builder/

Fernandez, T. (2020, 1 de septiembre). *What is Canary Deployment?* Semaphore. https://semaphoreci.com/blog/what-is-canary-deployment

Git (s.f.). *About Git.* https://git-scm.com/about

Goodfellow, I., Papernot, N., Huang, S., Duan, R., Abbeel, P., & Clark, J. (2017, 24 de febrero). *Attacking Machine Learning with Adversarial Examples.* OpenAI. https://openai.com/blog/adversarial-example-research/

Google (2021a). *Google Kubernetes Engine*. https://cloud.google.com/kubernetes-engine

Google (2021b). *Concepts (Cloud Build Documentation)*. https://cloud.google.com/build/docs/concepts

Google (2021c). *Container Registry documentation*. https://cloud.google.com/container-registry/docs

Google (2021d). *Autopilot overview*. https://cloud.google.com/kubernetes-engine/docs/concepts/autopilot-overview

Google (2021e). *Starting a build from the command line and API*. https://cloud.google.com/build/docs/running-builds/start-build-command-line-api

Gradient boosting (2021, 26 de septiembre). En *Wikipedia*. https://en.wikipedia.org/wiki/Gradient_boosting

IBM Cloud Education (2020, 15 de abril). *What is AIOps?* IBM. https://www.ibm.com/cloud/learn/aiops

k-means clustering (2021, 20 de septiembre). En *Wikipedia*. https://en.wikipedia.org/wiki/K-means_clustering

Kolmogorov-Smirnov test (2021, 9 de septiembre). En *Wikipedia*. https://en.wikipedia.org/wiki/Kolmogorov%E2%80%93Smirnov_test

Kubernetes (2021a, 23 de julio). *What is Kubernetes?* https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/

Kubernetes (2021b, 20 de agosto). *CronJob*. https://kubernetes.io/docs/concepts/workloads/controllers/cron-jobs/

Linear regression (2021, 18 de septiembre). En *Wikipedia*. https://en.wikipedia.org/wiki/Linear_regression

Logistic regression (2021, 21 de septiembre). En *Wikipedia*. https://en.wikipedia.org/wiki/Logistic_regression

MongoDB (2021a). *Getting Started*. https://docs.mongodb.com/manual/tutorial/getting-started/

MongoDB (2021b). *MongoDB Atlas.* https://www.mongodb.com/es/cloud/atlas

OpenWeather (2021). *One Call Api.* https://openweathermap.org/api/one-call-api

OS-level virtualization (2021, 12 de septiembre). En *Wikipedia.* https://en.wikipedia.org/wiki/OS-level_virtualization

Python Software Foundation (2021). *About Python.* https://www.python.org/about/

Quality assurance (2021, 13 de septiembre). En *Wikipedia.* https://en.wikipedia.org/wiki/Quality_assurance

Quoma, S. (2019, 18 de diciembre). *What is DataOps?* IBM. https://www.ibm.com/blogs/journey-to-ai/2019/12/what-is-dataops/

Random Forest (2021, 13 de septiembre). En *Wikipedia.* https://en.wikipedia.org/wiki/Random_forest

Red Eléctrica de España (2020). *Sistema de Información de Medidas Eléctricas. Ficheros para el intercambio de información de medida (Versión 34, febrero de 2020).* https://www.ree.es/sites/default/files/01_ACTIVIDADES/Documentos/Documentacion-Simel/ficheros_para_el_intercambio_de_informacion_de_medida_v34_febrero2020.pdf

Red Eléctrica de España (2021). *ESIOS. Sistema de Información del Operador Del Sistema.* https://www.esios.ree.es/es

Red Hat (2021). *What is blue green deployment?* https://www.redhat.com/en/topics/devops/what-is-blue-green-deployment

RStudio (2021). *RStudio. Take control of your R code.* https://www.rstudio.com/products/rstudio/

Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., Chaudhary, V., & Young, M. (2014). *Machine Learning:*

The High Interest Credit Card of Technical Debt. Google. https://research.google/pubs/pub43146/

The R Foundation (s.f.). *What is R?* https://www.r-project.org/about.html

Ushey, K. (2021, 21 de julio). *Introduction to renv.* RStudio GitHub. https://rstudio.github.io/renv/articles/renv.html

Visengeriyeva, L., Kammer, A., Bär, I., Kniesz, A., & Plöd, M. (s.f.). *MLOps Principles.* MLOps. https://ml-ops.org/content/mlops-principles

YAML (s.f.). *The Official YAML Web Site.* https://yaml.org

Bibliografía

Arundel, J., & Domingus, J. (2019). *Cloud Native DevOps with Kubernetes: Building, Deploying, and Scaling Modern Applications in the Cloud*. O'Reilly Media.

Burns, B. Beda, J., & Hightower, K. (2019). *Kubernetes: Up and Running: Dive Into the Future of Infrastructure* (2ª edición). O'Reilly Media.

Dunning, T., & Friedman, E. (2017). *Machine Learning Logistics: Model Management in the Real World.* O'Reilly Media.

Gift, N., & Deza, A. (2021). *Practical MLOps: Operationalizing Machine Learning Models.* O'Reilly Media.

Kane, S. P., & Matthias, K. (2018). *Docker: Up & Running: Shipping Reliable Containers in Production* (2ª edición). O'Reilly Media.

Lakshmanan, V., Robinson, S., & Munn, M. (2020). *Machine Learning Design Patterns: Solutions to Common Challenges in Data Preparation, Model Building, and MLOps.* O'Reilly Media.

Lantz, B. (2015). *Machine Learning with R* (2ª edición). Packt Publishing.

Loukides, M. (2020, 9 de junio). *Machine Learning and the Production Gap.* O'Reilly. https://www.oreilly.com/radar/machine-learning-and-the-production-gap/

Luksa, M. (2018). *Kubernetes in Action.* Manning Publications.

Raj, E. (2021). *Engineering MLOps: Rapidly build, test, and manage production-ready machine learning life cycles at scale.* Packt Publishing.

Shah, A. (2020, 21 de junio). *Challenges Deploying Machine Learning Models to Production. MLOps: DevOps for Machine Learning.* Towards data science. https://towardsdatascience.com/challenges-deploying-machine-learning-models-to-production-ded3f9009cb3

Singh, P. (2020). *Deploy Machine Learning Models to Production: With Flask, Streamlit, Docker, and Kubernetes on Google Cloud Platform.* Apress.

Talby, D. (2018, 5 de junio). *Lessons learned turning machine learning models into real products and services. Why model development does not equal software development.* O'Reilly. https://www.oreilly.com/radar/lessons-learned-turning-machine-learning-models-into-real-products-and-services/

Treveil, M., Omont, N., Stenac, C., Lefevre, K., Phan, D., Zentici, J., Lavoillotte, A., Miyazaki, M., & Heidmann, L. (2020). *Introducing MLOps: How to Scale Machine Learning in the Enterprise.* O'Reilly Media.

UNIVERSIDAD DE MÁLAGA | uma.es

E.T.S de Ingeniería Informática
Bulevar Louis Pasteur, 35
Campus de Teatinos
29071 Málaga

E.T.S. DE INGENIERÍA INFORMÁTICA