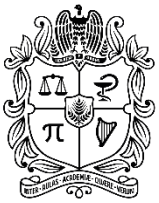

UNIVERSIDAD
NACIONAL
DE COLOMBIA

Aproximación metodológica para la integración de las metodologías DataOps y MLOps aplicadas al trading automático

Andrés David Tamayo Palomino

Universidad Nacional de Colombia
Facultad de Minas, Área Curricular de Sistemas e Informática
Medellín, Colombia
2021

Aproximación metodológica para la integración de las metodologías DataOps y MLOps aplicadas al trading automático

Andrés David Tamayo Palomino

Tesis de Maestría presentada como requisito parcial para optar al título de:

Maestría en Ingeniería-Analítica

Director:

Juan David Velásquez Henao, PhD

Línea de Investigación:

Analítica

Universidad Nacional de Colombia

Facultad de Minas, Área Curricular de Sistemas e Informática

Medellín, Colombia

2021

Dedicatoria

Este trabajo va dedicado a mis padres, familiares, amigos, y todas las personas que estuvieron apoyándome en todo el proceso de desarrollo de este trabajo, y que ha pesar de todas las adversidades, siempre estuvieron atentos y dispuestos a darme aliento en esta etapa de gran aprendizaje para mi vida.

Declaración de obra original

Yo declaro lo siguiente:

He leído el Acuerdo 035 de 2003 del Consejo Académico de la Universidad Nacional. «Reglamento sobre propiedad intelectual» y la Normatividad Nacional relacionada al respeto de los derechos de autor. Esta disertación representa mi trabajo original, excepto donde he reconocido las ideas, las palabras, o materiales de otros autores.

Cuando se han presentado ideas o palabras de otros autores en esta disertación, he realizado su respectivo reconocimiento aplicando correctamente los esquemas de citas y referencias bibliográficas en el estilo requerido.

He obtenido el permiso del autor o editor para incluir cualquier material con derechos de autor (por ejemplo, tablas, figuras, instrumentos de encuesta o grandes porciones de texto).

Por último, he sometido esta disertación a la herramienta de integridad académica, definida por la universidad.

Nombre: Andrés David Tamayo Palomino

Fecha 28/09/2021

Agradecimientos

Se agradece a la Universidad Nacional de Colombia, por brindar los espacios, herramientas y personal que ayudaron y facilitaron el desarrollo de este trabajo final.

Profunda gratitud y respeto al Doctor Juan David Velásquez Henao, principal colaborador durante todo el proceso. Gracias a su dirección, experiencia y profundo conocimiento, fue fundamental en la elaboración del trabajo final.

Resumen

Aproximación metodológica para la integración de las metodologías DataOps y MLOps aplicadas al trading automático

Descripción:

Las metodologías tradicionales usadas para el desarrollo de aplicaciones de ciencia de datos dan cabida a una gran deuda técnica en su proceso desarrollo que termina por ocasionar el fracaso de muchos proyectos. Para mitigar dicha deuda, se plantea una integración entre los enfoques de DataOps y MLOps, en donde se definen los beneficios, pasos y principios a seguir en la construcción de una aplicación enmarcada bajo esta integración propuesta. Posterior a ello se comparan dos aplicaciones de trading automático usando Deep learning, una de ella es desarrollada bajo las metodologías tradicionales y otra bajo la integración de los enfoques de MLOps y DataOps. Se concluye que la integración de ambos enfoques resulta altamente provechosa, permitiendo suplir las falencias de las metodologías tradicionales y las deficiencias de cada uno de los enfoques usados de forma independiente.

Palabras clave: MLOps, DataOps, Trading, Deep Learning

Abstract

Methodological approach for the integration of DataOps and MLOps methodologies applied to automatic trading.

Description:

The traditional methodologies used for the development of data science applications give place to a great technical debt in their development process that ends up causing the failure of many projects. To mitigate this debt, an integration between DataOps and MLOps approaches is proposed, where the benefits, steps and principles to be followed in the construction of an application framed under this proposed integration are defined. Subsequently, two automatic trading applications using Deep learning are compared, one of them is developed under traditional methodologies and the other one under the integration of MLOps and DataOps approaches. It is concluded that the integration of both approaches is highly useful, allowing to supply the shortcomings of the traditional methodologies and the deficiencies of each of the approaches used independently.

Keywords: MLOps, DataOps, Trading, Deep Learning

Contenido

Pág.

Lista de figuras

Pág.

Lista de tablas

Pág.

Lista de Símbolos y abreviaturas

Abreviaturas

Abreviatura	Término
ML	Machine learning
DevOps	Development Operations
MLOps	Machine learning operations
TF	Tensorflow
CRISP-DM	Cross Industry Standard Process for Data Mining
API	Application Programming Interface
KDD	Knowledge Discovery in Databases
SEMMA	Sample, Explore, Modify, Model, and Assess
SAS	Statistical Analysis System
ROI	Return on investment
CI	Continuous Integration
CD	Continuous Delivery
CT	Continuous training
CM	Continuous Monitoring
BI	Business Inteligence
KPI	Key Performance Indicator
TDSP	Team Data Science Process

1 Introducción

Con el aumento de la generación de datos a volúmenes nunca antes vistos, el incremento de aplicaciones que hacen uso de ellos ha crecido formidablemente en los últimos años. Esto se debe, entre otras razones, a la masificación de la inteligencia artificial en la cual se han dado grandes avances en las áreas de visión por computador, el entendimiento del lenguaje natural, los sistemas de recomendación e incluso la notable mejora de otros campos más tradicionales.

Desgraciadamente, un gran porcentaje de proyectos no se finalizan con éxito por múltiples factores que llevan a situaciones negativas dentro de la organización. Entre ellos se encuentra el uso de prácticas tradicionales de ingeniería de software que ignoran las particularidades del desarrollo de modelos en aprendizaje de máquinas; esto se debe al alto grado de experimentación requerido, el desafío que presenta conservar la calidad de un modelo a través del tiempo, la mala calidad de los datos usados y su naturaleza cambiante, entre otros. Estos fracasos conllevan a que las empresas tengan grandes deudas técnicas que van acompañadas de consecuencias económicas negativas, mala reputación, o en el mejor de los casos, a una gran pérdida de tiempo para el equipo encargado de la construcción de las aplicaciones, aumentando los costos de desarrollo.

En vista de que las prácticas tradicionales carecen de los elementos necesarios para hacer frente a los productos basados en datos, han surgido enfoques prácticos como DataOps y MLOps. Estos permiten que las iniciativas tengan una mayor probabilidad de éxito; además, proporcionan grandes ventajas a los equipos de desarrollo, al garantizar la calidad de cada uno de los procesos, la facilidad del mantenimiento de la aplicación a lo largo de su ciclo de vida, el despliegue automático y una robusta flexibilidad de todo el sistema; adicionalmente, permite una adaptación rápida en entornos altamente cambiantes. Asimismo, gracias al elevado grado de automatización incorporado en cada fase, se acelera el tiempo de entrega de estas soluciones, permitiendo participar en más iniciativas,

invertir más tiempo en actividades de innovación e investigación y, por supuesto, obtener retornos económicos con mayor velocidad.

Sin embargo, no existen propuestas metodológicas que se encarguen de vincular los enfoques de DataOps y MLOps, que, aunque se asemejan en un par de aspectos, tienen notables diferencias. Esto es debido a que DataOps es un enfoque que gira entorno al dato y que procura garantizar su calidad durante todo el ciclo de vida del producto, mientras que MLOps busca reducir las complejidades asociadas a la construcción y despliegue de modelos de machine learning. Esta diferencia de enfoques hace pensar que la integración de ambas metodologías podría brindar grandes ventajas y mejoras respecto a su uso individual, ya que se combinarían sus fortalezas; esto implica que la construcción de aplicaciones de machine learning se podría realizar con una mayor calidad y eficiencia en cualquier ámbito.

Es por esto, que en este capítulo se exploran las limitaciones de las aproximaciones tradicionales de desarrollo de modelos de aprendizaje de máquinas y los aportes de las metodologías de DataOps y MLOps, con el fin de determinar si es posible formular un proyecto de tesis de maestría a partir de la integración de ellas.

El resto de este capítulo está organizado así: En la Sección 1.1 se presentan los antecedentes de algunas de las metodologías tradicionales más utilizadas; luego, en la Sección 1.2, se plasman los problemas y dificultades presentes en los productos de datos en general y en el trading automático en particular. Posteriormente en la Sección 1.3 se plantea la hipótesis propuesta de este trabajo final; seguidamente en la Sección 1.4 se plantean los objetivos del presente trabajo. Y por último en la Sección 1.5, se presenta el mapa del documento que describe la estructura de este trabajo final.

1.1 Antecedentes

Diferentes estudios y encuestas [1] han mostrado que CRISP-DM es la metodología más utilizada en el desarrollo de proyectos de minería de datos y analítica con aproximadamente un 43%, seguida, por los procesos empíricos o no estructurados, y en tercer lugar está SEMMA, la cual se encuentra en declive, debido a que está estrechamente relacionada con el uso de productos de SAS [2].

La desventaja principal de estas metodologías radica en que se trata el desarrollo de productos de datos como un flujo secuencial y altamente lineal en donde se muestran los procesos de una forma muy simplificada y límites bien definidos en cada etapa [3] [4] esta visión señala que cada proceso se inicia una vez ha finalizado el anterior. Esto se debe a que este modelo de desarrollo es basado en las metodologías de desarrollo en cascada que se usaron durante los años 80's, las cuales tienen profundas deficiencias [5]. Estas características llevan a los equipos de desarrollo a destinar mucho tiempo de planificación y creación de estrategias, que tienen como fin una entrega única casi perfecta, la cual, en la práctica, no cumple con los requerimientos que el cliente desea en última instancia.

Uno de los errores más comunes de las metodologías más tradicionales, es que no dedican suficiente atención a un producto mínimo viable, que permita una rápida retroalimentación de los clientes para garantizar que los recursos y tiempo están siendo depositados en las construcciones de los requerimientos adecuados. Otro error, propiciado principalmente por el alto de grado de abstracción de dichas metodologías, es que se ignora una gran cantidad de complejidades presentes en diferentes etapas del desarrollo; esto se ve reflejado por ejemplo, en aquellos proyectos en los que se invierte demasiados recursos en optimizar la etapa de entrenamiento del modelo, y se ignora el esfuerzo que implicará la etapa de uso del modelo y su mantenimiento en un ambiente productivo [6].

En respuesta a ello, la industria ha propuesto una nueva metodología como la planteada por Microsoft llamada TDSP [7], en la que se mejoran significativamente algunos aspectos, ya que reconoce algunas de las características más importantes de los productos de datos, y gracias a ello, se enfatiza la necesidad de que los procesos sean ágiles, iterativos, estandarizados y colaborativos. Sin embargo, estas metodología aún no están teniendo una alta acogida, y presentan varias deficiencias en diferentes etapas que son solventadas por los enfoques de DataOps y MLOps.

1.2 Definición del problema real

1.2.1 Problemas asociados al despliegue de productos de analítica

La generación de grandes volúmenes de datos provenientes de innumerables fuentes y los desarrollos tecnológicos en hardware ligados a los grandes avances en investigación, son solo algunos de los elementos que han impulsado a que cada vez más compañías deseen incluir procesos de analítica e inteligencia artificial en sus negocios. Más aún, los recursos destinados a estos proyectos son cada vez mayores [8]. Sin embargo, las organizaciones se enfrentan al reto de llevar los trabajos académicos a un ambiente productivo, ya que muchos de los proyectos desarrollados en la industria discrepan fuertemente respecto a los trabajos académicos, ya que estos últimos suelen ser pruebas de concepto ejecutadas en ambientes locales que ignoran los requisitos de calidad, estabilidad y precisión que son exigidos a cualquier producto de datos en un ambiente productivo durante todo su ciclo de vida [4].

Una de las posibles consecuencias de la omisión de los requisitos de calidad, estabilidad y precisión, será que las iniciativas fallen rotundamente y que las empresas no logren sus objetivos de incorporar inteligencia artificial en sus negocios. Según [8] solo el 22% de las compañías que desarrollan proyectos de machine learning han desplegado un modelo en producción, pese a que el 50% del total de compañías encuestadas tienen como meta de mayor importancia desarrollar modelos para ser usados en un ambiente de producción; y el 40% del total tienen como prioridad desplegar los modelos en ambientes productivos [9].

El proceso de llevar a producción un modelo de analítica o machine learning es obligatorio, ya que solo un producto operando en esta etapa puede traer un valor real de negocio a la compañía; es decir, los proyectos tendrán un ROI de cero y un balance financiero negativo hasta que los modelos puedan ser convertidos en un productos de datos que sean usados en producción [10]. Por lo tanto, el tiempo necesario para llevar a cabo el despliegue de un producto de datos debería ser una de las métricas más importantes a optimizar en un proyecto de industria de analítica o machine learning.

Por otra parte, la automatización de procesos mecánicos y repetitivos en los flujos de trabajo de las organizaciones ha tenido un gran auge en los últimos años, debido a la ventaja competitiva que genera. El desarrollo de productos de analítica y machine learning no es ajeno a esta tendencia. Sin embargo, la decisión de optar por la automatización de los procesos de desarrollo de productos de datos conlleva a factores críticos y de alto riesgo en muchas organizaciones [3], por las dificultades asociadas al manejo de modelos y grandes volúmenes de datos, y al análisis del performance de los artefactos usados para la automatización. Esto se debe, entre otros factores a que la automatización de pruebas, despliegues y actividades de monitoreo del producto, requieren la adición y modificación de sus prácticas respecto al software tradicional debido a la complejidad inherente de los productos de datos. Estas dificultades, evidencian desde la práctica una gran brecha entre el desarrollo de un modelo y el despliegue del mismo, a pesar de los esfuerzos destinados a ello, resaltando lo desafiante de esta tarea y las carencias de los métodos tradicionales utilizados para el desarrollo y ejecución de los proyectos de analítica.

La razón principal de la brecha en el despliegue de las aplicaciones basadas en machine learning, es que estas distan en varios aspectos cruciales respecto al desarrollo y despliegue del software tradicional [10]. Asimismo, las metodologías tradicionales usadas habitualmente en el desarrollo de productos de datos no tienen un planteamiento que permita sortear exitosamente las dificultades de la etapa de despliegue; de esta forma, lo usual es encontrar que el desarrollo del modelo de machine learning no es el reto principal enfrentado por las organizaciones, sino, el construir el sistema de ML tal que pueda operarse de forma continua en producción.

El despliegue de aplicaciones basadas en machine learning también ha sufrido las mismas fallas y problemas propios del desarrollo de software, y por tanto, las mismas metodologías que han mejorado la producción de software, también pueden mejorar el desarrollo de productos de machine learning, tal como los métodos ágiles. Estos últimos permiten que las compañías reaccionen más rápidamente a los requerimientos del cliente y aceleran el tiempo de entrega del producto. Dichos métodos también mejoran sustancialmente el ROI, debido a cada característica que se desarrolla es entregada en cada iteración, lo que permite que sea inmediatamente monetizada en lugar de esperar a que el proyecto entero termine [4]. Para que esto sea exitoso, los equipos de trabajo encargados de las actividades operativas de infraestructura y mantenimiento, deben tener el agilísimo articulado en sus procesos [11], enfoque que es conocido como DevOps, tal que se realice

un trabajo en conjunto entre los equipos de desarrollo e infraestructura desde la etapa de diseño y desarrollo hasta el despliegue y soporte del producto de datos. Es bien sabido que DevOps minimiza los errores en cada entrega, reduce el tiempo requerido para solucionar problemas y facilita la incorporación de nuevas funcionalidades [12].

Lamentablemente, muchas de las falencias de las metodologías tradicionales para el desarrollo de productos de datos deben de ser solventadas por los científicos de datos, los cuales usualmente no tienen fuertes fundamentos en desarrollo de software [2] ni en enfoques como DevOps. Con esta clara necesidad, se hace preciso el desarrollo o modificación de las metodologías tradicionales, incorporando técnicas y herramientas de la ingeniería de software tradicional, las metodologías ágiles y DevOps. Así, con el propósito de suplir dichas necesidades y cumplir con los objetivos de los proyectos de analítica de forma eficiente han surgido propuestas como DataOps y MLOps. Es necesario recalcar que al igual como sucede con DevOps, dichas propuestas no son una metodología, lenguaje, plataforma o framework concreto, sino enfoques [13] que buscan agilizar y automatizar la gestión del ciclo de vida del dato y de los modelos analíticos correspondientemente. Dichos enfoques son independientes en su totalidad de la infraestructura o stack tecnológico utilizado en los proyectos [3].

DataOps es una combinación de herramientas y métodos que permiten el desarrollo de aplicaciones de analítica, garantizando una calidad impecable de los datos en todos sus procesos [11]. Por otra parte, MLOps hace referencia al uso de principios científicos, herramientas y técnicas de machine learning e ingeniería de software tradicional para diseñar y construir sistemas de inteligencia artificial, teniendo impacto directo en cada una de las etapas como la recolección de datos, la construcción del modelo y el despliegue en producción para el consumo final del producto de analítica [14]. El objetivo de MLOps es reducir el tiempo malgastado en operaciones rutinarias, permitiendo que los recursos expertos de la compañía puedan invertir más tiempo en la experimentación e implementación de nuevos desarrollos. MLOps al igual que DevOps, emerge del entendimiento de separar el desarrollo de los modelos de machine learning de su proceso de entrega.

Como consecuencia de lo anterior, aún existen muchas falencias inherentes a los procesos y metodologías de desarrollo de los productos de datos.

1.2.2 Problemas asociados al trading automático

Como se mencionó anteriormente, la generación y acumulación de datos ha presentado un crecimiento a una tasa desmesurada en los últimos años y el sector financiero no ha sido una excepción [15]. Tradicionalmente, los traders han utilizado el análisis técnico y el análisis fundamental para tomar decisiones de inversión. Como consecuencia del incremento exponencial de la información disponible, los traders ya no pueden realizar un análisis veloz y eficiente de la información disponible; por tanto, no es de extrañar el auge de los sistemas de trading automático, los cuales hacen uso de algoritmos para automatizar algunos o todos los elementos de una estrategia de inversión. Estos sistemas automáticos permiten tomar operaciones con mayor velocidad, optimizar la estrategia a lo largo del proceso de inversión, entrar y salir de operaciones, crear una gestión de riesgos, entre otros muchos beneficios [16].

Por otra parte, las series de precios de los activos transados en las bolsas de valores añaden factores que representan una mayor complejidad que otros tipos de datos, debido a que son altamente no lineales, no estacionarios y de naturaleza ruidosa [17]. Por ello, el uso de diversas técnicas de ML, que distan de las tradicionales en el sentido de que no hacen supuestos tan fuertes como la linealidad de los datos a procesar, han mostrado su capacidad de encontrar patrones ocultos en grandes cantidades de datos que permiten llevar a cabo predicciones con mayor precisión. Estos pronósticos son aprovechados por los sistemas automáticos de negociación para tomar decisiones comerciales.

Recientemente, se han considerado otras fuentes de datos complejas que pueden ser incorporadas en la toma de decisiones. Estas incluyen las emociones humanas, crisis económicas, intereses financieros y políticos, e incluso, variables tan volátiles e impredecibles como el clima, las cuales tienen impactos directos o indirectos sobre la economía [18]. Esto causa que la implementación de dichos modelos no sea una tarea trivial. Por consiguiente, para la construcción de un sistema de trading automático, efectivo y que presente mínimos errores en su ejecución, se deben incorporar elementos que permitan un alto nivel experimental de modelos, gran flexibilidad en su arquitectura y de rápida integración al sistema [15].

Aunque ya se cuenta con experiencias exitosas en desarrollo de sistemas de trading automático, continúan existiendo muchas limitantes en su mantenimiento debido a que dichos sistemas han sido desarrollados bajo enfoques tradicionales.

1.2.3 Definición del problema en analítica

Los progresos en hardware e investigación en analítica han crecido cuantiosamente en los últimos años, pero estos avances no representan un incremento equivalente en la productividad de los proyectos de analítica. Contradictoriamente, los informes [2] advierten que por el contrario, la tasa de éxito de los proyectos está disminuyendo, evidenciando que solo el 22% de las compañías tuvieron un crecimiento en sus ingresos y una rápida recuperación de sus inversiones en proyectos de ciencia de datos .

Uno de los factores que hacen más difícil la ejecución de un proyecto de analítica, es que los equipos están conformados por una variedad bastante amplia de roles, los cuales no son perfiles precisamente técnicos, como es el caso de los expertos en materia, analistas de negocio, economistas, gerentes, entre otros. Y, por otro lado, están los conocidos perfiles técnicos como desarrolladores, ingenieros de datos, científicos de datos, ingenieros de machine learning, personal de tecnologías de información (IT), etc. Cada uno de ellos maneja términos, herramientas y lenguajes diferentes, lo que aumenta el grado de dificultad de comunicación del equipo [3]. Por lo tanto, equipos conformados por perfiles totalmente diferentes, podrían estar atacando distintas etapas del proyecto, con diversas herramientas y siguiendo diferentes flujos de trabajo. Como una consecuencia de lo anterior, es común que cada célula de trabajo no tenga una idea clara o precisa de cuáles son las expectativas que tiene el equipo al que se va a entregar un proceso y esto puede implicar fricciones en el proceso de desarrollo del proyecto.

Por otro lado, existen nuevas fuentes de generación de datos, semiestructurados o no estructurados, tales como imágenes, texto, audio, entre otros. Estas nuevas fuentes de datos, expanden las oportunidades de negocio a los científicos de datos para crear aplicaciones y obtener provecho de estas; pese a esto, la mayoría de los científicos indican que los principales retos para la ejecución de un proyecto son: baja calidad de los datos [19], su alto volumen, el difícil acceso a las fuentes [20] y los cambios que sufren en el tiempo [21].

La dificultad en el acceso a las fuentes de datos se debe a que frecuentemente están almacenados en fuentes separadas e incomunicadas entre ellas, por lo cual se hacen necesarias numerosas plataformas y APIs para acceder a ellas. Integrar esta diversidad de fuentes, requiere de una amplia variedad de habilidades que son raramente reunidas en un solo individuo [2]; por ejemplo, los científicos de datos en la mayoría de ocasiones,

no son ingenieros de software y están más enfocados en el análisis de datos, la construcción y evaluación de experimentos para el desarrollo de modelos, en lugar de construir APIs que consuman diferentes fuentes de datos o el desarrollo de una aplicación de software end to end [22]. Así que los equipos de analítica tienen que solicitar accesos a todas estas fuentes, lo cual es un proceso bastante tedioso y tiende a convertirse en un gran cuello de botella en los proyectos. Además, como se ha mencionado, los datos de mala calidad eventualmente contienen errores de diferente índole, generando consecuencias negativas en todo el pipeline.

Cuando los datos erróneos pasan a través de todo el pipeline y son entregados a los usuarios o clientes mediante APIs, dashboards o reportes, pueden ser causantes de decisiones de negocio erradas, debido a que están basadas en datos de mala calidad. Estas decisiones tendrán impactos negativos como pérdida de oportunidades comerciales y financieras [23], e incluso, que los equipos de analítica junto con la aplicación pierdan credibilidad [4] mientras generan gran insatisfacción en el cliente. Con el fin de evitar estas consecuencias negativas, los pipelines de datos y modelos deben de ser monitoreados constantemente, para garantizar que estos se encuentran alineados con los datos que fueron entregados al pipeline de entrenamiento, bajo la premisa de que eran un reflejo fidedigno o una muestra lo suficientemente representativa de los datos que se iban a encontrar en producción [3]. Si algún factor de negocio cambia y el perfil de los datos muta respecto al perfil inicial, es posible que las salidas de la aplicación sean erróneas [22] y no se esté cumpliendo con el objetivo inicial para el que se construyó la aplicación.

Sumándose a las dificultades mencionadas, también está el hecho de que rara vez los clientes tienen bien definidos los requerimientos de negocio, ya que como no son habitualmente usuarios expertos en datos, aún no tienen claras las oportunidades de negocio hasta que se les enseñan [11], y en ocasiones, al proporcionar análisis a las partes interesadas del negocio, se generan más preguntas que respuestas, pues los conocimientos analíticos permiten a los usuarios comprender el negocio de nuevas formas, estimulando la creatividad, lo que conduce a nuevas solicitudes de mayor análisis y complejidad.

Estos nuevos conocimientos del negocio desembocan en nuevas ideas y cambios en los requerimientos del proyecto [11]; por ello, el equipo de analítica debe de estar en la capacidad de responder a estos eficazmente. Esto revela una de las grandes diferencias

que existe entre el desarrollo de software tradicional, el cual es más lineal, con reglas de negocio previamente establecidas y el desarrollo de proyectos de analítica, en donde, por el contrario, se suele iterar en diferentes etapas que suelen ser altamente experimentales [10]. Por estas razones, se hace necesario probar diversas variables, técnicas de entrenamiento, configuraciones de parámetros, entre otros [22]. Tal es el caso de la naturaleza cambiante de estos proyectos, que en la implementación del mismo, el equipo de analítica podría toparse con necesidades como la adquisición de más y diferentes datos, el uso de otras técnicas de modelamiento o incluso percatarse que las suposiciones que se hicieron al inicio del proyecto eran erróneas, llevando a múltiples reentrenamientos con datos de diferentes fuentes o transformaciones de los mismos, e incluso, discutir una reformulación del problema, métrica u objetivo de negocio [24].

Adicionalmente, los proyectos de machine learning poseen otra dificultad respecto al desarrollo de software tradicional, porque, en primer lugar, este tipo de aplicaciones no son deterministas [3], y en segundo lugar, involucran 3 niveles fundamentales: datos, modelo, y el código. Esto significa que en los sistemas basados en machine learning, el detonante para la generación de una nueva versión del proyecto podría ser la combinación de cambios en el código, en los datos o en los modelos; este problema es conocido como el principio de “Cambiar cualquier cosa cambia todo” [25], lo cual es una característica que complica aún más este tipo de desarrollos.

Una vez sorteadas las dificultades relacionadas con el desarrollo de un modelo o el pipeline de datos, la etapa final consiste en llevar el modelo a producción. Allí, los equipos se enfrentarán a retos que no se encuentran en los despliegues de aplicaciones de software convencionales, tales como que el perfil de los datos vaya evolucionando. Por ende, en el mejor caso, los modelos van a fallar en producción y en el peor, se producirán resultados basura de forma silenciosa [26].

Esto se debe a que una vez que el modelo es puesto en producción, se inicia un proceso de degradación [27] que bajará su rendimiento paulatinamente, dando como resultado salidas ligeramente erradas, mucho más difíciles de detectar en el sistema [28] en comparación a un error de software tradicional; esto puede generar consecuencias negativas que serán detectadas tardíamente.

Dicho esto, se puede evidenciar que la conservación de la calidad de los pronósticos a través del tiempo es otro gran obstáculo a vencer, lo cual se debe en parte a que las técnicas tradicionales de desarrollo de software suelen supervisar solo que la arquitectura y el software estén funcionando, y no están orientadas a detectar el mal funcionamiento de un modelo. Adicional a todo esto, algunas de las fuentes de otros problemas está relacionado con dependencias de datos, complejidad de los modelos, pruebas de monitoreo y los cambios del mundo real qué hay que plasmar [4].

Y como si las dificultades técnicas mencionadas anteriormente, no fueran suficientes, otro problema que afrontan las compañías es el déficit de personal de alta gerencia en proyecto de analítica. Esto se debe a que el punto de quiebre en la investigación masiva de analítica y el aumento exponencial en la generación de datos fue después del 2010. Gracias a ello, se tienen científicos con un alto nivel técnico pero una marcada deficiencia de gerentes experimentados en el manejo de la cultura del dato [2], sin liderazgo de analítica y que suelen cometer errores gigantescos al planificar los proyectos de esta índole. Eso se explica por la alta popularidad alcanzada por las metodologías tradicionales de minería de datos que simplifican en gran medida los pasos requeridos para materializar una necesidad o problema de negocio a un modelo de machine learning con todas sus implicaciones [3].

Recientemente han ganado terreno los enfoques de MLOps y DataOps, los cuales permiten hacer frente y solventar en gran medida algunas de las dificultades previamente mencionadas. Sin embargo, la aplicación de estos enfoques de forma aislada, no logra resolver completamente todos los obstáculos presentes en el desarrollo de un proyecto de machine learning y no existe una metodología que permita integrar ambos enfoques, y así, gozar de los beneficios que cada uno de ellos proporciona en su área de aplicación.

DataOps es una metodología que tiene como objetivo mejorar la calidad de los datos involucrando la automatización de procesos y las metodologías ágiles para alcanzar dicho objetivo. Por otro lado, MLOps tiene como prioridad agilizar la construcción y entrega a producción de los modelos de machine learning de la forma más eficiente posible, reduciendo la deuda técnica generada por el uso de las metodologías tradicionales en su proceso de desarrollo. Ambos enfoques serán tratados con mayor detalle en el Capítulo 2 de este trabajo final.

1.3 Hipótesis

La hipótesis en que se basa este trabajo final es la siguiente: es posible desarrollar una aproximación metodológica que permita integrar la bondades de las aproximaciones MLOps y DataOps, la cual permita superar las falencias individuales de cada una de estas metodologías y de las metodologías tradicionales usadas en los proyectos basados en datos.

1.4 Objetivos

1.4.1 Objetivo general:

Desarrollar una aproximación metodológica que permita integrar la bondades de las aproximaciones MLOps y DataOps, y ejemplificar su uso, en el caso de la negociación automática.

1.4.2 Objetivos específicos:

- Desarrollar una aproximación metodológica entre DataOps y MLOps
- Desarrollar un prototipo que ejemplifique el uso de la metodología en el contexto de la negociación automática

1.5 Mapa del documento

El resto del documento está organizado como sigue: En el Capítulo 2 se definen los enfoques de DataOps y MLOps, además se introducen similitudes y diferencias de ambos enfoques; luego, en el Capítulo 3 se desarrolla la propuesta de integración metodológica. Posteriormente, en el Capítulo 4 se desarrolla un comparativo que permite contrastar el uso de las metodologías tradicionales en producto de datos respecto a la metodología resultante de la integración, esto en el contexto del desarrollo de una aplicación de trading automático. Y por último en el Capítulo 5 se presentan las conclusiones del trabajo.

2 Similitudes y diferencias entre DataOps y MLOps

En este capítulo se presenta una discusión de las metodologías DataOps y MLOps, que tiene como fin realizar una contextualización de la metodología propuesta en el Capítulo 3.

2.1 Producto de datos

Un producto de datos es una pieza de software que consume y genera grandes volúmenes de datos que son consumidos por otros productos de datos. Este es el tipo de piezas de software que se construyen como resultado de los proyectos de inteligencia de negocios, analítica y machine learning. Se diferencian de las piezas de software tradicionales en que los datos son usados para la toma de decisiones. Ejemplo de este tipo de piezas son los tableros de control, los reportes automáticos para toma de decisiones, modelos de pronóstico y de soporte a las decisiones.

2.2 Ingeniería de Datos

Es el rol encargado de gestionar los datos en las organizaciones, y tiene como responsabilidades:

- La ingestión de datos desde diferentes fuentes.
- La optimización de las bases de datos para análisis.
- La remoción de los datos corruptos.
- El diseño, construcción, prueba y mantenimiento de arquitecturas de datos como sistemas de bases de datos y sistemas de procesamiento a gran escala.
- La puesta a disposición de los datos para que sean consumidos por los distintos grupos de trabajo los cuales incluyen a los equipos de ciencia de datos, machine learning, inteligencia de negocios, etc.
- Construcción de tuberías para procesamiento de grandes volúmenes de datos.

Este rol es altamente técnico y posee destrezas, conocimientos y habilidades en bases de datos relacionales y NoSQL, big data, computación en la nube, lagos de datos. Sus

capacidades técnicas le permiten gestionar datos estructurados, semiestructurados y no estructurados, tanto en ambientes locales como en arquitecturas en la nube.

2.3 DevOps

DevOps es un enfoque organizacional que destaca la comunicación y la colaboración multifuncional dentro y entre los equipos de trabajo, y que tiene como propósito cerrar la brecha entre los equipos de desarrollo de software (Dev) y Operaciones de IT (Ops).

Los equipos de desarrollo se encargan de implementar cada uno de los requisitos del sistema, de la construcción de pruebas y la adición de cada avance al sistema de control de versiones. Por su parte, los equipos de operaciones se encargan de administrar las plataformas tecnológicas sobre las que están soportadas las aplicaciones desarrolladas, reducción del costo de cada servicio, integración de cambios, generación de alertas, entregas funcionales de la aplicación, entre otros.

Aplicar el enfoque de DevOps en las organizaciones permite acelerar la entrega de cambios [29] gracias al uso de integración continua, pruebas de calidad, entregas iterativas e incrementales del producto con despliegues automáticos utilizando un conjunto de herramientas y prácticas de desarrollo en este proceso [30].

2.4 DataOps

El término DataOps, abrevia y fusiona las expresiones de datos y operaciones. Fue introducido por primera vez en 2015 por Lenny Liebmann [31], el cual lo definió como un conjunto de metodologías y prácticas técnicas, provenientes de las metodologías ágiles, ingeniería de software, DevOps, ingeniería de datos y la analítica de datos [32]. DataOps tiene el propósito de reducir la complejidad y el tiempo requerido para la ejecución de las operaciones en los productos de datos, mientras garantiza una alta calidad de las mismas [2].

Por otra parte, Ereth [33] indica que:

> *DataOps es el conjunto de prácticas, procesos y tecnologías que combinan una perspectiva integrada y orientada a procesos sobre los datos, con*

> *automatización y métodos ágiles utilizados en la ingeniería de software, para mejorar la calidad, la velocidad y la colaboración, con el fin de promover una cultura de mejora continua.*

Ambas definiciones tienen varios aspectos en común; el primero de ellos es que involucran a las metodologías ágiles en el desarrollo de estos productos, lo que facilita sustancialmente el trabajo conjunto entre los equipos de analítica y los usuarios, debido a que estas metodologías enfatizan la comunicación, colaboración e integración entre cada uno de los roles del equipo [11]. Una gran ventaja que proporciona integrar el agilísimo en el proceso de desarrollo de los productos de datos, es que se obtiene un equilibrio entre la innovación, la generación de ideas, los nuevos requerimientos y los controles de gestión de los pipelines de datos [34]. Así, los cambios en los requisitos por parte de los usuarios se tornan más simples, permitiendo que los equipos de analítica tengan un feedback más veloz y de esta manera se potencie la capacidad de mejorar e incorporar características adicionales en los productos de datos, para incrementar el impacto positivo sobre el negocio en cada una de las etapas del proyecto [2].

Adicionalmente, al trabajar el agilísimo acoplado a DevOps y algunas de las técnicas de ingeniería de software (como la automatización y ejecución de pruebas en cada componente) el equipo puede publicar nuevos pequeños incrementos de forma constante y segura, aprovechando las prácticas de CI/CD inherente de DevOps [35]. Esto permite responder rápidamente a los cambios de los requerimientos que proporcione el cliente debido a la flexibilidad y adaptabilidad que proporciona el uso de estas prácticas [2]. Es necesario recalcar, que en este proceso de entrega incremental de forma ágil no se descuidad la calidad de los datos en ningún momento, ya que gracias al uso de procesos de control estadístico estrictos, se garantiza una calidad de datos extremadamente alta y tasas de errores muy bajas en todo el proceso, resultado de la detección temprana de errores en datos [11].

DataOps también permite la integración de varios componentes a través de mecanismos de extensión, que permiten la lectura, y manipulación de diferentes tipos de bases de datos, bien sea internas o externas, que contengan datos estructurados y no estructurados. Además, permite especificar las dependencias existentes sobres los datos, ofreciendo una mejor experiencia a las personas involucradas en el ciclo de vida de los proyectos de analítica, gracias a que facilita múltiples etapas de todo el proceso de integración de datos

[36]. Gracias a esta integración, sumada a las altas velocidades de entregas incrementales, la monetización de la inversión en el desarrollo de productos de datos se inicia más pronto, ofreciendo confianza a los clientes y permitiendo que las personas involucradas en el desarrollo de estos productos puedan desarrollar más proyectos y a su vez mejorar las operaciones de analítica y su rendimiento [11].

Con esto se puede intuir que DataOps está presente en todo el ciclo de vida del dato [32],y es de gran utilidad para la optimización de la gestión de pipelines de datos, dashboards, consultas a base de datos, y cualquier otra aplicación que dependa de alguna manera de los datos, incluyendo las aplicaciones de aprendizaje de máquinas [37].

Los detalles de los principios y fases de este enfoque serán especificados en el Capítulo 3.

2.5 MLOps

MLOps se define como la cultura y práctica que hace uso de principios científicos, herramientas y técnicas de machine learning e ingeniería de software tradicional para diseñar y construir [14] cada una de las etapas que componen una aplicación de machine learning.

En cada una de estas etapas, MLOps busca estandarizar y simplificar la gestión de cada una de ellas [3]; esto se logra al descomponer sistemas complejos, en subcomponentes reutilizables, donde cada componente puede ser conectado en flujos de trabajos portable [38]. Además, MLOps busca maximizar la automatización de todos los pasos de la construcción del sistema de ML, incluyendo la integración de componentes, las pruebas, el despliegue, el versionamiento y el manejo de la infraestructura.

Cada una de las prácticas derivadas de MLOps, tiene como propósito la unificación del sistema de desarrollo de machine learning (equivalente al Dev de DevOps) y la operación del sistema de Machine Learning (equivalente al Ops de DevOps) [22];con esta unificación se podrá desarrollar y mantener productos de Machine Learning en producción de forma eficiente y fluida; esto reduce la fricción técnica de convertir una idea en un modelo llevado a productivo en el menor tiempo posible, minimizando los riesgos [10]. En consecuencia, esta práctica ayuda a las organizaciones y líderes de negocios a generar un valor a largo

plazo y reducir el riesgo asociado con las iniciativas de ciencia de datos, Machine Learning e inteligencia artificial [3].

MLOps, al igual que DataOps, acopla varios principios y técnicas de DevOps en sus procesos. Entre ellos están los pipelines de integración y despliegue continuo (CI/CD); estos han sido un enfoque importante para poder tener automatización, calidad y disciplina en la creación de procesos reproducibles y confiables para entregar nuevas versiones del producto de analítica a producción fácilmente y de forma masiva; por ejemplo en MLOps, los componentes de CI se extenderían a las pruebas y validación de esquemas, datos y modelos, mientras que los componentes de CD apoyan el despliegue de los pipelines de entrenamiento, como también el servicio de las predicciones finales o su aplicación. Además de otros componentes como pruebas continuas (CT) y monitoreo continuo (CM), en donde se auditan los procesos para confirmar que los modelos no solo están disponibles sino que también sean precisos, y si estos presentan errores, que sean re-entrenados rápidamente y actualizados de forma automática. Por ello, diferentes autores definen MLOps brevemente como la habilidad de aplicar los principios de DevOps a las aplicaciones de machine learning [39] .

Otro objetivo de MLOps es proveer una infraestructura de investigación común con diferentes niveles de abstracción que pueda cerrar la brecha entre los investigadores y practicantes y ayudar al equipo en la formación de un sistema complejo, codiseñado, que puede ser inmediatamente usado en producción. Esto permite que los científicos de datos puedan trabajar con un nivel de abstracción más alto del sistema, permitiendo a los ingenieros continuar mejorando el nivel más bajo de abstracción para evolucionar el software y el hardware sin romper o parar el sistema [38].

Los detalles de los principios y fases de este enfoque serán especificados en el Capítulo 3.

2.6 Similitudes y diferencias

2.6.1 Similitudes

La primera similitud notable entre MLOps y DataOps radica en que ambos enfoques son utilizados en el desarrollo de productos que giran en torno al dato. Por ello, las tareas que

hacen referencia a la creación, manipulación y mantenimiento de datasets son una de las actividades en común de ambos enfoques.

Como se mencionó anteriormente, las aplicaciones de datos presentan grandes retos y para hacer frente a estos, se hace necesaria una gran velocidad de ejecución de procesos que permita entregar los productos de datos con una alta frecuencia, eficiencia y que tengan la capacidad de escalar; es decir, que los productos puedan crecer de una forma que satisfagan el aumento de la demanda [2].

En el software tradicional algunos de estos problemas también están presentes, así que para sortear estos desafíos los equipos de desarrollo y de operaciones han desarrollado un conjunto de culturas, prácticas y tecnologías que permitan alcanzar y mejorar esa velocidad requerida para entregar valor en los tiempos definidos, a esto se le conoce como DevOps. Esta es la segunda similitud entre los enfoques, ya que tanto DataOps como MLOps podrían definirse de alguna manera como una extensión de la metodología de DevOps en aplicaciones de ciencias de datos [32] ambos enfoques incorporan altos niveles de colaboración, automatización, pruebas, y agilísimo en sus procesos; dichas prácticas provienen originalmente de la definición de DevOps.

Estas prácticas derivadas tendrán grandes impactos y beneficios, algunos de ellos van a permitir mejorar la colaboración entre los integrantes de diferentes niveles del proyecto [11]. La encuesta [40] indica despliegues de código 30 veces más frecuente y con un 50% menos de fallos; esto es explicado como consecuencia de los pipelines automatizados que permiten reducir el tiempo de despliegue y el tiempo necesario para solucionar problemas.

2.6.2 Diferencias

Como se discutió anteriormente, el objetivo de DataOps es agilizar la gestión de los procesos en los productos de datos, sin descuidar en ningún momento la calidad de los datos que fluyen a través de sus pipelines [41]. Esto resulta de gran utilidad en las aplicaciones de machine learning; por ello, MLOps se solapa en cierta medida con DataOps. No obstante, MLOps brinda más solidez a sus procesos mediante la adición de otras características inexistentes en DataOps [3], las cuales son de vital importancia para garantizar la calidad, gestión y monitoreo de un modelo de ML [42], eje principal del enfoque.

Parcialmente, y cuidando las diferencias implicadas, MLOps puede ser concebido en cierta medida como un subconjunto de DataOps, y este, a su vez ser un subconjunto de DevOps. Esto se puede evidenciar en las aplicaciones que hacen uso de DataOps pero que no hacen uso de MLOps, como las aplicaciones de BI, las cuales también tienen procesos de extracción y transformación de datos pero sin hacer uso de ningún modelo de machine learning en sus procesos. Por el contrario, en una aplicación de machine learning que sea desarrollada bajo el enfoque de MLOps, tradicionalmente suelen usarse varios de los principios de DataOps, ya que con estos se generan los datasets que serán suministrados a los algoritmos de aprendizaje para la generación de los modelos.

MLOps y DataOps no solo se diferencian en su fin, sino también en algunos componentes más específicos como la integración y el despliegue continuo. En integración continua, DataOps se centra en la ejecución de pruebas y validaciones de esquemas y datos, mientras que en MLOps el objetivo principal es ejecutar pruebas y validaciones sobre los modelos desarrollados [22]. Ahora en el proceso de despliegue continuo, DataOps despliega todo el pipeline de procesamiento de datos utilizado en el desarrollo, mientras que en MLOps el pipeline desarrollado para el entrenamiento del modelo no va a ir a producción, sólo el código de pre procesamiento de datos y los artefactos requeridos para la inferencia van a producción. Por último, MLOps adiciona un nuevo componente, el entrenamiento continuo o monitoreo continuo, en donde se ejecutan pruebas y monitoreo constante que permitan determinar la ejecución del reentrenamiento automático de modelos y que detecten algún posible drift en los datos [2].

2.6.3 Relación de Ingeniería de Datos y DataOps

La función principal de los ingenieros de datos en un equipo de ciencia de datos es garantizar que los datos sean gestionados de forma precisa, segura y eficiente en todos los pipelines de datos de las aplicaciones. El papel de DataOps en este proceso es facilitar y brindar las técnicas, herramientas y prácticas necesarias al ingeniero o científico de datos para garantizar una calidad elevada de los datos, orquestar procesos, aumentar la colaboración y comunicación en el equipo, proporcionar la agilidad necesaria para adaptarse rápidamente a cualquier cambio y brindar capacidad de escalabilidad necesaria para la gestión de grandes cantidades de datos.

2.7 Conclusiones

En este capítulo se discutieron las principales diferencias y similitudes entre los enfoques de DataOps y MLOps. Asimismo, algunas aclaraciones y definiciones necesarias que permiten tener una visión y contexto más claro del alcance y relación de estos enfoques con otros términos usados frecuentemente en los desarrollos de ciencias de datos.

3 Propuesta de integración de DataOps y MLOps

DataOps y MLOps están presentes desde las etapas iniciales de adquisición de datos hasta la entrega del producto, teniendo como objetivo alcanzar más conocimientos, y mejores análisis mientras se sigue siendo veloz, económico y de alta calidad. Así, que al hacer uso de ambos enfoques en un proyecto de analítica, el equipo se puede ver beneficiado de múltiples maneras; una de las más destacadas es el aumento de la velocidad de desarrollo en cada etapa. Cuando los procesos de analítica tienen mayor velocidad, sin dejar de lado y sin descuidar la calidad de los procesos, se puede desbloquear el potencial creativo de una organización, como también permite reducir el costo marginal para responder las preguntas de interés comercial con base a los datos [11].

Otro efecto positivo que tiene un gran beneficio en los equipos, es la estandarización de las configuraciones de cada proceso, y gracias a ello, los científicos de datos pueden incorporarse rápidamente o pasar de un equipo a otro y encontrar los mismos entornos de desarrollo. Esta estandarización mejora la eficiencia y reduce el tiempo dedicado a la configuración de un nuevo proyecto [43] ,y al tener un entorno ágil pero estandarizado, permite también adaptarse velozmente a los requerimientos cambiantes que puedan surgir debido a la generación de nuevos insights o debido a cambios en el negocio.

Asimismo la práctica de los enfoques puede beneficiar a las organizaciones al fomentar la colaboración entre equipos de científicos de datos, ingenieros de datos, analistas de datos, operaciones y propietarios de productos, permitiendo alinear los equipos de forma más adecuada y así ofrecer avances con mayor velocidad. Igualmente, se evitarán gran cantidad de imprevistos en los proyectos, gracias a que los procesos de control estadístico forman parte fundamental de las pruebas ejecutadas de forma automática en la mayoría de las etapas del proyecto, garantizando la calidad de los datos y por ende garantizan mayor fiabilidad en los análisis [11], dando como resultado una estimación más precisa de los tiempos requeridos para el desarrollo de cada fase y entregar valor de forma más predecible.

A continuación, se presenta la propuesta de integración de ambos enfoques, con el fin de aprovechar las bondades de cada uno de los enfoques.

3.1 Resumen

Ambos enfoques presentan similitudes y diferencias, las cuales son resumidas en la Tabla 1. En dicha tabla se presentan las distintas actividades relacionadas con el desarrollo y despliegue de un producto de datos; estas actividades fueron obtenidas al desagregar y comparar las fases propuestas en el desarrollo de aplicaciones usando el enfoque tradicional de machine learning y las propuestas de DataOps y MLOps. Como se puede observar, DataOps le da relevancia a las actividades relacionadas con los datos, las cuales incluyen su ingestión, limpieza y manipulación, versionamiento, pruebas, monitoreo, orquestación y gobierno. Mientras que MLOps se concentra en el desarrollo y puesta en operación de los modelos; en dicha tabla también se puede ver que el ML tradicional se enfoca en el desarrollo del modelo pero deja la tarea de su puesta en producción directamente al equipo de DevOps. Esta tabla comparativa evidencia claramente que existe una clara oportunidad de mejora si se desarrolla un enfoque que integre las propuestas de DataOps y MLOps. Dicha propuesta corresponde a la última columna de la Tabla 1.

Actividades desarrolladas en machine learning bajo los enfoques de DataOps y MLOps

Actividades	ML tradicional	DataOps	MLOps	Integración
Ingestión de datos	✓	✓	✓	✓
Análisis de datos	✓	✓	✓	✓
Preparación de datos	✓	✓	✓	✓
Entrenamiento del modelo	✓	×	✓	✓
Evaluación del modelo	✓	×	✓	✓
Validación del modelo	×	×	✓	✓
Despliegue del modelo	✓	×	✓	✓
Versionamiento de Datos	×	✓	×	✓
Versionamiento de Modelos	×	×	✓	✓
Versionamiento de código	✓	✓	✓	✓
Pruebas de datos	×	✓	×	✓
Pruebas de modelo	×	×	✓	✓
Pruebas de código	×	✓	✓	✓
Orquestación	×	✓	×	✓
Monitoreo de datos	×	✓	×	✓
Monitoreo de modelos	×	×	✓	✓

Monitoreo de infraestructura	×	✓	✓	✓
Integración Continua	×	✓	✓	✓
Despliegue Continuo	×	✓	✓	✓
Entrenamiento Continuo	×	×	✓	✓
Reproducibilidad y Trazabilidad de experimentos	×	×	✓	✓
Contenerizar y usar múltiples ambientes	×	✓	×	✓
Gobierno de Datos	×	✓	×	✓

Tabla 1. Diferencias y similitudes entre las metologías tradicionales, MLOps, DataOps

3.2 Consideraciones

El primer paso en cualquier proyecto de machine learning es llevar a cabo una profunda comprensión del negocio para luego definir el caso de uso, es decir el problema o la necesidad a solucionar en el proyecto, para posteriormente establecer uno o varios criterios de éxito [3], y una vez establecidas estas dos etapas en un proyecto, se inicia el proceso de desarrollo a un nivel más técnico. Este paso es obligatorio e independiente de la metodología, pero no se profundizará en este trabajo, ya que la propuesta integradora se enfoca en aspectos más técnicos.

Por otra parte, de DataOps se hereda el uso de las metodologías ágiles, las cuales no están estrictamente sujetas a algún marco ágil particular, ni acoge necesariamente las estrategias más comunes usadas en la ingeniería de software; esto es una consecuencia de la naturaleza experimental e iterativa de las operaciones que se han mencionado anteriormente; por lo tanto han de ser mucho menos predecibles los tiempos y actividades necesarios para su total desarrollo. Para superar estas dificultades, no es necesaria la invención de un nuevo marco ágil para ser usado en las aplicaciones de datos, sino que por el contrario basta solo con adaptar algunas de las prácticas de los marcos ya existentes y que han mostrado funcionar en la ingeniería de software. Entre los marcos ágiles más utilizados, Kanban sobresale respecto a las demás metodologías como Scrum y XP debido a su mayor capacidad de sobrellevar la incertidumbre inherente de este tipo de proyectos; esto se debe a su gran flexibilidad [2]. En otras metodologías como Scrum, la labor de estimar actividades de forma precisa es bastante ardua, lo que conlleva a malas estimaciones e incumplimiento de plazos [44].

También, es necesario mencionar que algunas de las actividades descritas en la integración de ambos enfoques, hace un poco más énfasis en los modelos de aprendizaje supervisado. Esto se ve reflejado en la adición de actividades como el etiquetado de datos, supervisión de ground truth, ingeniería de características, etc, las cuales son realizadas exclusivamente en este tipo de modelos. Sin embargo, el enfoque resultante de la integración también puede aplicarse en su mayoría en aplicaciones de analítica descriptiva y modelo de aprendizaje no supervisado, teniendo de igual manera un excelente impacto en este tipo de aplicaciones.

3.3 Marco general de la propuesta de integración

En cuanto al proceso de desarrollo técnico en los proyectos de machine learning, el objetivo principal es construir un modelo. Sin embargo, es bien conocido que los pipelines necesarios para el desarrollo de este, podrían ser bastante complejos y quizá tengan cientos de procedimientos [32]; pese a esto, es posible hacer una abstracción y simplificación de todo el proceso en 3 niveles o fases principales, como se puede ver en la Figura 1: datos, modelo y despliegue. En cada uno de estos niveles, se deben de aplicar varios principios y prácticas provenientes de DataOps y MLOps, con el objetivo de garantizar calidad, velocidad y flexibilidad de la metodología de desarrollo usada para la implementación de los proyectos de analítica. Esto principios y prácticas son: versionamiento, pruebas, automatización y reproducibilidad [24].

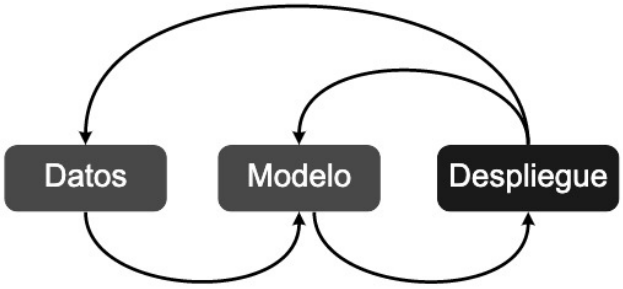

Figura 1 Niveles fundamentales

El nivel de datos tiene como objetivo la adquisición y preparación de los datos para la creación del dataset, lo cual se alcanza mediante la implementación de procesos ETL. Este nivel tiene una gran importancia debido a que la cantidad y la calidad de los datos, afectan el rendimiento de todo el sistema, desde la generación del modelo hasta su performance

en producción. El segundo nivel, corresponde al modelo de machine learning, y tiene como objetivo la construcción de los pipeline de ML, que son el core del proyecto; aquí se incluye el entrenamiento, evaluación, pruebas y empaquetamiento del modelo. Por último, se tiene el nivel de despliegue, en donde se llevan a cabo tareas como el paso a producción del modelo, monitoreo del comportamiento del modelo y el registro de cada solicitud de inferencia y predicción del modelo.

A continuación se va a abordar con mayor profundidad y detalle los elementos necesarios para la construcción de cada uno de los niveles que componen un proyecto de machine learning, ejecutado bajo el enfoque de MLOps integrado con las prácticas y principios presentes en DataOps.

3.4 Fases de la Metodología

3.4.1 Datos

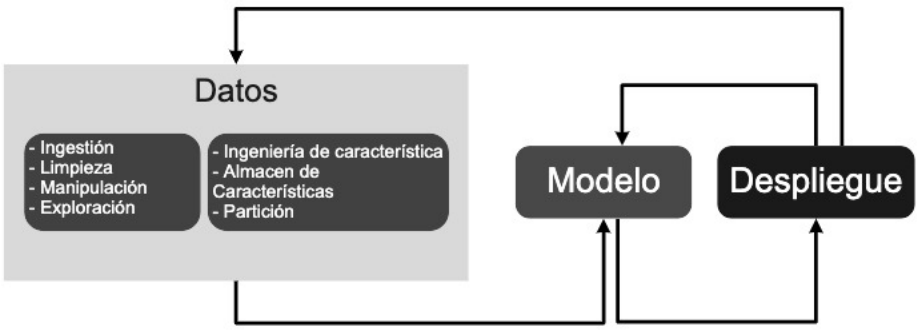

Figura 2. Nivel de Datos

En cualquier proyecto de analítica, el siguiente paso luego de una exhaustiva comprensión del negocio y del establecimiento de los criterios de éxito, es la adquisición y preparación de los datos. Esta etapa se caracteriza por requerir elevados recursos computacionales y tiempos de desarrollo; en esta fase, las pruebas de control estadístico juegan un papel fundamental, debido a que algún error en los datos en esta etapa se propaga por las etapas restantes. Por esto se puede afirmar, que el modelo, el cual es el artefacto principal en este tipo de proyectos va a terminar siendo tan bueno como sean sus datos y en última instancia, no será solo el modelo el que se vea afectado por la calidad de los datos, sino también, el desempeño general del sistema en producción.

3.4.1.1 Relación con la ingeniería de datos y la construcción de modelo en machine learning

El pipeline de datos anteriormente mencionado, se podría dividir en dos partes según su propósito. La primera parte hace referencia a las labores de ingeniería de datos, las cuales son la ingestión, limpieza, manipulación y exploración de los datos; estas actividades tienen como fin último producir datos de gran calidad, y serán especificadas desde la sección 3.4.1.2 hasta la 3.4.1.5. La otra parte, hace referencia a aquellas operaciones que aunque igualmente incluyen la manipulación y transformación de datos en sus procesos, se diferencian en que estas operaciones tienen como fin único el obtener un mejor ajuste del modelo a los datos; aquí se encuentran las fases de ingeniería de características, creación de repositorios de características y la división de los datos; estos procesos serán detallados desde la sección 3.4.1.6 hasta la 3.4.1.8. A continuación se listan las operaciones de todo el pipeline de datos que tendrá como objetivo general, la creación del dataset con el que se entrenará y evaluará el modelo.

3.4.1.2 Ingestión de datos

Durante esta fase se obtienen los datos en su formato original; en algunos casos podría ser necesaria la integración de diferentes fuentes, que a su vez podrían estar en diferentes formatos, con el fin de formar un solo conjunto de datos. Dependiendo también del volumen de datos, podría ser necesaria la creación o generación de datos sintéticos para enriquecer los datos ya obtenidos.

Ahora bien, es muy importante evitar la propagación de errores a través del pipeline de datos tan rápido como sea posible; para ello, se ejecutan algunas pruebas que permitan la detección temprana de posibles errores en los datos o en el proceso de ingesta [2]. Algunas de las verificaciones simples que se ejecutan son:

- Detección y eliminación de archivos duplicados.
- Comprobaciones del tamaño de los archivos, de tal forma que el tamaño de los archivos de origen coincida con los archivos recientemente adquiridos.
- Comparación del número de registros y archivos entre el origen y destino.

Existen otras pruebas más complejas que se deben de ejecutar sobre los datos, pero estas serán especificados en pasos posteriores a este, y en la sección dedicada al Principio de Pruebas con mayor detalle.

Otro aspecto de gran importancia e impacto en las organizaciones, es la práctica y uso de herramientas de gobierno de datos, ya que permiten que los datos puedan ser descubiertos y accedidos de forma fácil, confiable e intuitiva [45]. Los catálogos de datos son una de las herramientas de gobierno de datos de mayor relevancia, debido a que muchas organizaciones suelen tener una cantidad abrumadora de datos. Una de las posibles consecuencias del manejo de este gran volumen de información, es que una actividad tan habitual para los integrantes del equipo como lo es buscar y localizar los datos adecuados para el desarrollo de un proceso, se convierta en una tarea bastante demandante y engorrosa. Esto se debe a que el equipo deberá examinar miles de directorios, nombres y versiones de datos para dar con la información aparentemente indicada; esto genera problemas de confiabilidad sobre los datos seleccionados, ya que es posible que los nombres de los conjuntos de datos y de los campos, no guarden una relación con su contenido, haciendo mucho más difícil comprender que representan los datos inspeccionados [46].

Para hacer frente a dicha dificultad, se recurre al uso eficiente de los metadatos en los catálogos de datos; estos, permiten al equipo descubrir nuevas fuentes de información, para innovar y desarrollar nuevas iniciativas. Adicionalmente, permiten identificar rápidamente la procedencia de dichos datos, saber cómo se crearon, su información de uso, y con ayuda de etiquetas, identificar y comprender velozmente si dichos conjuntos de datos han de resultar útiles o no, incrementando la productividad de análisis y la velocidad en la que se obtiene la información adecuada [47].

Desgraciadamente, no basta simplemente con tener un fácil acceso y alta disponibilidad en los datos, ya que por ejemplo, en muchos sectores regulados como los financieros, médicos o políticos, la seguridad y la protección de los datos tienen una alta prioridad. Incluso, en algunas compañías no reguladas es necesario proteger los datos que son comercialmente sensibles o que permiten identificar claramente a los usuarios registrados en dichas base de datos [48]. Para ello, las compañías suelen construir datalakes y/o data warehouse que limitan y restringen el acceso a diversas fuentes dependiendo de varios

criterios; esto termina causando un cuello de botella para la organización, debido a que tradicionalmente el acceso a dichas fuentes se hace con la generación de tickets que tienden a ser un proceso altamente burocrático, debido a que deben de considerar muchos aspectos de seguridad, limitando a muchos usuarios de datos en su capacidad y velocidad de descubrimiento.

Para poder sortear estos problemas las organizaciones necesitan poner frenos en forma de reglas y restricciones para crear confianza [2]. La organización necesita confiar en las personas que tienen acceso a los datos y simplificar el proceso de acceso tanto como sea posible. Y por otra parte, los usuarios de los datos, necesitan confiar en que los datos que están utilizando son correctos; estos aspectos reflejan el gran papel que tienen los catálogos de datos para obtener fácilmente la trazabilidad y metadata de los datos.

3.4.1.3 Limpieza de datos

Como se ha mencionado en múltiples ocasiones, los datos adquiridos habitualmente contienen una gran cantidad de errores, como datos faltantes, duplicados, inválidos, etc; y para poder reparar estas imperfecciones se ha de requerir una cantidad enorme de esfuerzo por parte del equipo de datos [20]. Ahora bien, independientemente del tipo de error a ser solucionado, las actividades a realizar en esta etapa se pueden dividir en dos fases. La primera consiste en la identificación de errores y en caso de ser necesaria, la validación de esta detección con los expertos; y la segunda, es la corrección de estos errores identificados [49].

La fase de identificación de errores puede incluir técnicas de detección cuantitativa o cualitativa. La primera, involucra en su mayoría métodos estadísticos para identificar primordialmente anomalías en los datos [50] , mientras que las segundas, hacen uso del establecimiento de restricciones para encontrar aquellos datos que violan las reglas del negocio. Todas las técnicas utilizadas en esta fase, permiten cumplir con los modelo de calidad de datos establecidos en estándares como el ISO/IEC 25012 [51], el cual contiene aspectos como la completitud, la consistencia, la credibilidad, la precisión, entre otros, que tienen como fin último garantizar datos de una calidad elevada. A continuación se listarán algunas de las técnicas de detección y corrección de errores más comunes para esta fase de limpieza [48]:

- Detección y corrección de outliers mediante métodos basados en estadísticas, distancias e incluso en modelos; estos últimos son de gran utilidad en datasets que poseen una alta dimensionalidad.
- Definir y ejecutar la estrategia de imputación sobre los valores nulos presentes en el dataset, la cual dependerá de la naturaleza de la variable.
- Definir reglas sobre las variables dependiendo de su naturaleza; esto incluye restricciones de unicidad, de tipo, de unidades y rangos según el tipo.
- Limpieza de campos de texto, lo cual implica remover caracteres de puntuación, numéricos, espacios, etc. También se incluye técnicas como normalization, tokenization, lemmatization, stemming, entre otras.
- Corrección de formatos de fechas, teléfonos, monedas, correos y otras variables que tienen algún patrón regular intrínseco.
- Verificación de los campos de unión entre diferentes fuentes de datos.

Asimismo, es pertinente resaltar que en ocasiones es primordial poseer un dominio robusto del negocio, para así tomar buenas decisiones durante este entendimiento y limpieza de datos [3]. Esto se debe a que según el segmento de negocio, lo que podría considerarse como un dato erróneo o atípico en cierto ámbito, podría ser usual dentro de otro dominio; así que, sin este conocimiento específico, algunas de las suposiciones hechas en esta etapa podrían tener efectos perjudiciales sobre la aplicación, lo cual podría ser difícil de detectar más adelante sin una buena intuición del dominio del negocio.

3.4.1.4 Manipulación de datos

Luego de haber identificado y corregido los errores en la fase anterior, se procede a la ejecución de operaciones y transformaciones sobre los datos, los cuales serán el insumo principal en la etapa de análisis descriptivo y exploratorio. En esta fase se ejecutan operaciones como:

- Concatenación de fuentes.
- Agregaciones sobre las tablas de datos para el análisis de subpoblaciones.
- Muestreo.
- Filtrado.
- Redimensionamiento de tablas.

3.4.1.5 Exploración de datos

Una vez los datos han sido depurados, se procede a realizar un entendimiento detallado del contenido y estructura de estos datos. Este paso también es de gran importancia debido a que podría afectar decisiones futuras en la etapa de modelado, como que algoritmos conviene utilizar y que variables podrían tener un mayor impacto en el ajuste del modelo. Por otro lado, también puede ser utilizado para reconocer las propiedades y esquemas de estos datos, que servirán como base fundamental para plasmar las expectativas de la estructura y tipo de dato que habría de esperar el modelo en la fase de inferencia una vez este se encuentre en producción [3]. A continuación se especificará algunas de las labores necesarias en esta etapa:

- Documentación de todas y cada una de las suposiciones e inferencias hechas sobre los datos obtenidos, bien sean con o sin conocimiento del negocio.
- Desarrollar análisis exploratorios gráficos y cuantitativos, los cuales a su vez se pueden realizar de forma univariada o multivariada.
    - En el análisis gráfico se construyen:
        - Gráficos de relación entre las variables.
        - Gráficos de distribución y tendencia de los datos.
        - Gráficos de datos categóricos.
        - Visualización de modelos de regresión
    - En el análisis cuantitativo se construyen:
        - Medidas de dispersión.
        - Medidas de tendencia central.
        - Medidas de forma.
        - Medidas de correlaciones y covarianzas.
        - Dominio de las variables.

Estos análisis y estadísticas también deben de ser generados sobre algunas subpoblaciones de los datos, con el objetivo de descartar o reconocer posibles sesgos en los datos y evitar la propagación de esta problemática hacia etapas posteriores o también para entender el comportamiento del modelo en fases posteriores.

3.4.1.6 Ingeniería de características

La ingeniería de características hace uso del conocimiento estadístico, de ingeniería de datos, técnicas de reducción y del dominio de negocio para transformar, combinar, seleccionar o generar nuevas variables [52], bien sea añadiendo más información al dataset o utilizando las variables ya existentes. Algunas de las técnicas más utilizadas en esta fase son:

- Técnicas de reducción de dimensionalidad como PCA y t-SNE.
- Análisis de clústeres.
- Selección de características con métodos embebidos, recursivos o basados en correlaciones y filtros.
- Técnicas de escalamiento.
- Transformaciones no lineales como las logarítmicas o polinómicas.
- Bucketización.
- Codificación.
- Aplicación de operaciones aritméticas entre varias características.

Estos procedimientos tienen como objetivo facilitar el ajuste del modelo a los datos usados; esta fase busca hacer explicitas las características más ocultas de los datos, de tal forma que el modelo pueda capturar las relaciones complejas ocultas, y por lo tanto que no se concentre en relaciones superficiales fácilmente visibles. Para lograr esta explicites de las características de los datos, se recurre a la transformación de variables existentes, a la agregación de transformaciones de variables y a la agregación de nuevas variables. Esto conlleva a una mayor precisión del modelo en la solución del problema abordado y la posibilidad de solucionar problemas como el sesgo o el fairness presente en los datos gracias a las nuevas características.

Es necesario resaltar algunas consideraciones ha tener en cuenta en esta fase para reducir el costo computacional que podría darse en las fases de entrenamiento y operación en productivo del modelo. Esto se debe a que en algunos segmentos de negocio, estos tiempos podría impactar negativamente la experiencia del usuario o simplemente no cumplir con los requisitos del negocio. Un ejemplo de ello es perder una oportunidad de entrada al mercado financiero. Adicional a ello, un número mayor de variables y fuentes

diferentes podrían menguar la estabilidad y facilidad de mantenimiento del modelo en el tiempo, haciendo que el proyecto tenga una fragilidad y riesgo elevado.

Dicho esto, es de vital importancia que el equipo defina con anticipación el criterio para concluir el desarrollo de esta fase; ya que al buscar ilimitadamente más variables que estén involucradas en el negocio, se podría llegar al punto en el que hay incontables fenómenos o eventos que podrían afectar directa o indirectamente la variable objetivo. Así, esta etapa podría ser una tarea sin fin si no se establece un criterio para detenerla [3].

Es muy importante resaltar, que tanto la fase de limpieza de datos, como la fase de ingeniería de características, deben de ser ejecutadas exactamente igual en el proceso de entrenamiento e inferencia del modelo, esto con el fin de evitar discrepancias en el tratamiento de los datos de entrenamiento y de los datos que son recibidos en producción.

3.4.1.7 Almacén de características

El almacén de características hacen referencia a un repositorio que almacena de forma centralizada, las características creadas en la fase anterior; esto se hace con la intención de que diferentes equipos, de diferentes proyectos y de diferentes segmentos de negocio, puedan reusar características previamente construidas en otro momento [53]. Esto tiene un efecto positiva sobre la velocidad y resultado final del proyecto, ya que libera y ahorra gran cantidad de tiempo a los científicos o ingenieros que tendrían que construir dichas características desde cero, ya que usualmente es una tarea que consume gran cantidad de tiempo.

Un efecto directo del uso de almacenes de característica, es que lo encargados de construir las características podrán realizar otras tareas que aportan valor al negocio y evitar caer en el error de tener que replicar los costos computacionales y del tiempo de desarrollo que requirió el equipo que generó dichas características, bien sea en el mismo proyecto o en otros desarrollos. Hay que mencionar además, que no solo se ahorran recursos computacionales y de índole temporal, sino que también, se aprovecha de forma eficiente el conocimiento de cada frente de negocio para la generación de nuevas variables. Como se mencionó previamente, para crear nuevas características, es de gran utilidad tener una alta experticia en el segmento de negocio y gracias al uso de estos

almacenes, se podrá aprovechar este valioso conocimiento sin necesidad de invertir grandes esfuerzos en profundizar y conocer otro frente diferente de negocio.

Tradicionalmente, en los pipelines de datos, e incluso en gran cantidad de frameworks comerciales, el flujo habitual de los datos va desde la primera fase, donde se ingesta los datos sin procesar hasta el modelo en producción, es decir, un flujo end-to-end, en el cual las primeras etapas que corresponden al pipeline de datos son habitualmente unas de las más costosas computacionalmente; según Airbnb, la fase de ingestión de datos y las transformaciones necesarias para generar todas las características con las que se entrenará el modelo, podría significar un 60% del tiempo de desarrollo de un equipo para un proyecto [54].

Como resultado del uso de un almacén de características, la etapa de selección de características toma menos tiempo, gracias a la reutilización de los datos, y a que se evita el reprocesamiento de información. Por ello, el uso de un almacén de características permite descomponer el flujo del proyecto en dos. El primer flujo, “DataOps”, será el flujo responsable de la ingestión, validación, transformación, creación de características y su respectivo almacén. El segundo flujo, “MLOps”, será el encargado del entrenamiento, validación, despliegue y monitoreo del modelo, y por supuesto, también de la creación de la infraestructura que disponibiliza los servicios necesarios para llevar a cabo dichas operaciones [42]. De esta forma, el proceso para entrenar nuevos modelos y añadir nuevos datos serán independientes y modulares.

3.4.1.8 División de datos

Consiste en particionar los datasets generados en la fase anterior, en los conjuntos de entrenamiento, validación y prueba. Este proceso debe ser totalmente reproducible, incluso en aquellos casos en los que se hacen muestreos aleatorios. Dependiendo de la distribución de los datos obtenidos, los cuales fueron analizados en la fase de exploración, podría ser necesaria la aplicación de técnicas de sobremuestreo de clases o partición estratificada para obtener mejores resultados durante el entrenamiento del modelo.

3.4.2 Modelo

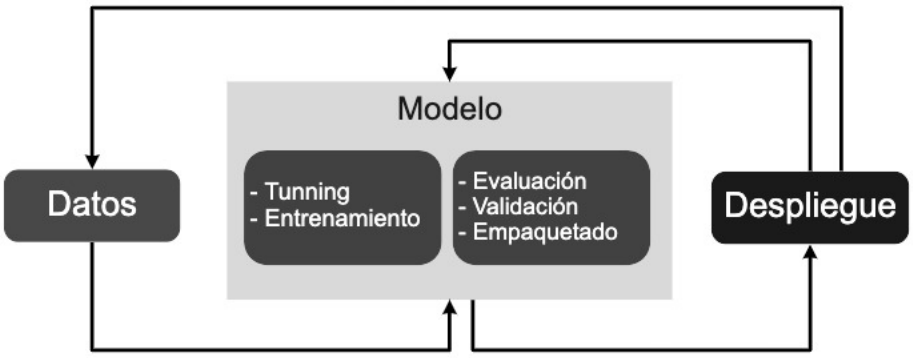

Figura 3. Nivel de Modelo

En esta etapa se genera el artefacto principal de los proyectos de analítica, el cual es el modelo. Esto se logra mediante la ejecución de algoritmos de machine learning y otros procedimientos que se van a detallar, pero no sin antes aclarar varias consideraciones sobre los modelos, ya que existe una brecha entre la definición teórica de un modelo de machine learning y lo que realmente significa e implica en la creación y gestión del modelo en la práctica. Al no ser consiente de esta diferencia, se puede caer en una deuda técnica muy grande y por ende, generar una posibilidad alta de fracaso, bien sea por efectos directos y visibles, o por efectos indirectos y silenciosos, los cuales son mucho más difíciles de identificar.

En teoría un modelo de machine learning busca proyectar un proceso real de la forma más precisa posible. Este modelo, luego de ser entrenado se reduce a una simple fórmula matemática [3], que retornará cierta salida dependiendo de las entradas. Estas entradas, idealmente han de ser lo más parecidas posibles a los datos de entrenamiento, debido a que los modelos tendrán un mejor performance, si los datos del pasado, es decir los datos de entrenamiento, son similares a los datos del futuro, es decir los datos presentados en producción.

Por otra parte, el modelo en la práctica agrupa más elementos que no se tienen muy presentes en la teoría, ya que para obtener un modelo, no basta simplemente con la obtención de la fórmula matemática que será posteriormente aplicada a los datos. Sino

que adicionalmente, se deben incluir todas las transformaciones y la selección de las variables necesarias para llevar los datos de entrada, hasta la estructura que necesita el modelo para ejecutar el entrenamiento; también, elementos como la métrica a optimizar es otro componente del modelo que ha de ser elegido cuidadosamente, ya que junto con los hiperparámetros seleccionados, han de ser un factor crítico en la obtención del modelo final. A continuación se detallarán las operaciones necesarias para la construcción del modelo de machine learning.

3.4.2.1 Entrenamiento y evaluación del modelo

El proceso de entrenamiento de un modelo consiste en proporcionar los datos proveniente del pipeline de ingeniería de datos a un algoritmo de aprendizaje automático [55]. Para ello es necesaria la previa especificación de la arquitectura del modelo, hiperparámetros, métricas y métodos de evaluación.

Posterior al entrenamiento, el siguiente proceso es la evaluación del modelo; para ello, se utiliza un dataset de pruebas representativo, que ha sido previamente construido en el pipeline de datos, para evaluar el desempeño del modelo en una situación similar a la operación en productivo. Las métricas obtenidas en esta fase, en cierta manera, van a garantizar la calidad del modelo entrenado y permitirán una comparabilidad entre los modelos recién entrenados y sus versiones previas. Es oportuno destacar la importancia que cobra una apropiada selección de métricas que se desean obtener en la evaluación del modelo. Ya que más allá de obtener la mejor métrica cruda de un modelo, esta, podría ser la responsable de la generación de modelos muy diferentes, y que podrían impactar el negocio de distintas formas [3].

Ahora bien, gracias a esta evaluación, el científico se puede percatar de varios problemas que pueden presentarse con el modelo, como una posible fuga de datos, un sobreajuste o un subajuste del modelo, entre otros. Por ello, es necesario un minucioso análisis del desempeño del modelo por parte del científico de datos, ya que podría identificar tempranamente alguna anomalía en el modelo, y ahorrar gran cantidad de trabajo adicional en el equipo.

No obstante, la identificación de dichos problemas podría ser una labor desafiante en algunos casos, y dependiendo del segmento de negocio u objetivo del proyecto, el impacto

de las inferencias hechas por el modelo podría ser mayor, por lo que la fase de evaluación podría exigir un mayor rigor. Por ello, en ocasiones se hace necesaria la ejecución de técnicas como el cross testing de diferentes métricas sobre el modelo, y la evaluación de diferentes subpoblaciones de los datos, también llamados slices [43]. El objetivo de esto es verificar si existe diferencia entre la evaluación con diferentes métricas aplicadas a diferentes fragmentos de datos, permitiendo incluso capturar de cierta manera la explicabilidad de las predicciones. Estas técnicas permitirán que los científicos de datos puedan comparar los modelos entrenados desde diferentes más criterios, y así, dar un juicio más preciso sobre un modelo. Sin embargo, en algunos casos, los datasets del mundo real podrían no ser suficientes para probar todos los posibles casos que pueden entrar a un modelo; para ello, es de gran utilidad crear datos sintéticos que puedan probar el modelo en estos casos especiales o limites, y así poder verificar su comportamiento frente a este tipo de datos y tomar acciones correctivas cuando sea necesario.

3.4.2.2 Validación funcional del modelo

Luego de obtener el conjunto de métricas mencionado, es necesario verificar que el último modelo obtenido tenga una mayor capacidad predictiva respecto al modelo previamente entrenado. A su vez, también es necesario verificar que el modelo, sin importar incluso que sea la primera versión entrenada, cumpla con los límites o thresholds previamente establecidos en la definición del proyecto [3] y sus criterios de éxito. Estos límites podrían estar definidos bien sea como métricas técnicas o de negocios, y una vez el modelo pasa estas validaciones, se procede a registrar o aprobar el modelo para que sea apto para despliegue [43]. Para poder llevar a cabo esta validación, es completamente indispensable que los modelos sean comparables entre si, ya que si no es posible comparar un modelo antiguo con una versión recientemente entrenada, se podría considerar incluso como tiempo perdido, ya que por definición, todos los entrenamientos de modelos que intentan resolver el mismo problema han de ser comparables cuantificablemente [26].

Bajo el enfoque de MLOps, es esencial que todas o la gran mayoría de operaciones hechas en esta etapa están altamente automatizadas. Sin embargo, resulta también conveniente que en este proceso exista un porcentaje de intervención humana para la toma de decisiones [45]. Esto se debe a que en ocasiones el científico de datos podría analizar el modelo de forma más profunda, y comunicar hallazgos al personal de negocios o al cliente,

para transmitir las implicaciones que pueda tener el performance actual del modelo y con base a ello tomar una decisión final.

Por último, es también necesario asegurarse de que todas las suposiciones previas que se hicieron, siguen teniendo vigencia, ya que el problema o reglas de negocio podrían haber cambiado fundamentalmente, y debido a esto, el modelo podría no ajustarse a los datos y cumplir con los nuevos criterios de éxito comercial.

3.4.2.3 Empaquetado del modelo

Posterior a la validación y aprobación del modelo para su despliegue, es necesario exportar el modelo con un formato específico que dependerá exclusivamente de donde se servirá. Este formato estará determinado por factores como la velocidad o latencia necesaria en la predicción, los recursos disponibles que tendrá el modelo en producción, e incluso, en el tipo de dispositivo o mecanismo de servicio que tendrá, como una aplicación web, dispositivo móvil, IoT, etc.

Esto implica que se debe considerar el hecho de que sea incluso necesaria una conversión del formato del modelo, o siendo un poco más riguroso, tener que utilizar técnicas de compresión. Esto es debido a que un factor agravante, diferente a los mencionados anteriormente, es que el costo computacional del uso del modelo en la fase productiva pueda ser cuantiosamente mayor al costo de entrenamiento. Esto es una consecuencia generada porque el modelo podría ser usado para predecir información millones de veces a gran velocidad, y entonces, al tener un modelo muy complejo, podría no solo tener un impacto económico sobre el negocio, sino también consecuencias energéticas y medioambientales [3]. Para mitigar este riesgo, el comprimir y simplificar el modelo, hará que la infraestructura necesaria para su operación sea menor, alivianado considerablemente dicho costo y sus otros efectos indirectos sobre el ambiente.

3.4.3 Despliegue

Dado que ya se tiene un modelo entrenado, evaluado, validado y empaquetado, el siguiente paso es integrarlo a la aplicación y desplegarlo en el ambiente objetivo, que es habitualmente el ambiente de producción. Este último suele ser bastante similar al ambiente de desarrollo, solo que lo riesgos comerciales ahora serán reales; allí, el modelo

recibirá nuevas entradas y ejecutará las predicciones. A continuación se detallan cuales son las operaciones necesarias para la implementación del último nivel del proyecto.

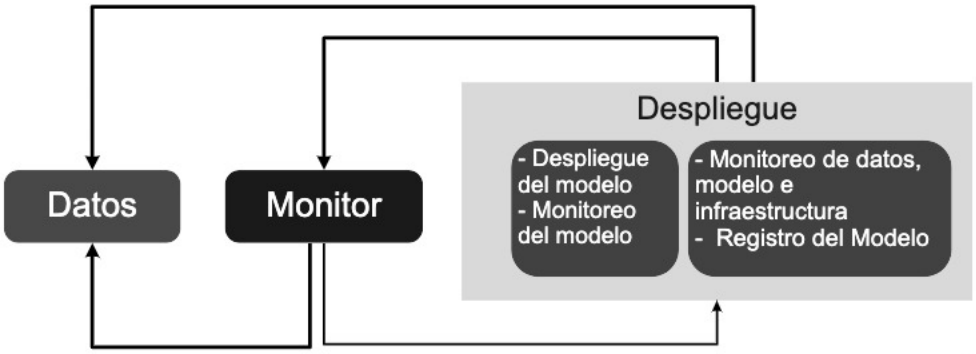

Figura 4. Nivel de Despliegue

3.4.3.1 Despliegue del modelo

En tanto el pipelines de ingeniería de datos y el entrenamiento de modelos se ejecuten adecuadamente, y el modelo tenga un performance offline esperado, este, deberá ser servido en el sistema; para ello, se pueden utilizar diferentes estrategias de despliegue, las cuales dependerán en cierta medida de la arquitectura e infraestructura necesaria para el uso del modelo en el negocio. En este proceso de despliegue, es importante que el entorno de desarrollo luzca tan semejante como sea posible al entorno productivo; esto implica evitar usar lenguajes, librerías, transformaciones o técnicas diferentes en la fase de desarrollo y la fase de producción [2].

Es necesario recalcar que el despliegue de una nueva versión de un modelo debe tratarse con sumo cuidado, ya que una decisión apresurada en un despliegue, podría significar inactividad del sistema, tiempos de inferencia elevados y errores altos en las predicciones, lo cual podría llegar a ser bastante perjudicial para el proyecto [3]. Para reducir este riesgo, se puede optar por un despliegue parcial de los modelos, lo cual permite contrastar métricas entre los usuarios que están expuestos a la versión anterior y los usuarios que usan el nuevo modelo; con base en estos resultados se podría tomar decisiones como: revertir los cambios, adquirir nuevos datos, generar nuevas características o probar nuevas arquitecturas para obtener un modelo totalmente diferente.

A continuación, se mostrarán algunas de las estrategias de despliegue parcial más utilizadas y las consideraciones a tener en cuenta para evitar errores o pérdidas a nivel económico o reputacional, debido a las consecuencias que pueda tener un comportamiento inesperado del modelo una vez este ha sido desplegado.

- **Modelo sombra** o **despliegue Blue/Green**: es un patrón que se utiliza con el fin de reemplazar un modelo por otro que ya se encuentra en producción, pero en lugar de apagar temporalmente el sistema, lo que se hace es enviar exactamente el mismo tráfico de producción al modelo antiguo y al nuevo. De esta manera, se podrá reunir evidencia sobre el comportamiento del nuevo modelo y decidir si puede ser publicado para reemplazar el antiguo, o en su defecto, rechazar esa versión [56] [57].

- **Despliegue Canary:** a diferencia de los modelos sombra, la carga enviada a los modelos que se van a comparar no es exactamente la misma. En este caso, solo un porcentaje bajo de la carga será dirigido al nuevo modelo, con el fin de minimizar la fracción de la población expuesta al nuevo modelo. Esto tiene el fin de minimizar el impacto de un modelo que no tenga el comportamiento esperado; posteriormente se analizará el comportamiento del modelo con esa fracción de los datos, para determinar si el modelo se publica o no [45].

3.4.3.2 Monitoreo del modelo

Si bien el modelo fue evaluado y validado antes de ser desplegado, el verdadero performance de este, solo podrá ser determinado de forma precisa una vez es evaluado en producción con datos que el modelo nunca ha visto antes; y es aquí en donde se presentan algunas de las mayores dificultades en los proyectos de machine learning. Esto se debe a que los cambios en reglas de negocio, datos, políticas y otros componentes importantes ocasionan una perdida de la capacidad de generalización del modelo y por ende, generan un mal comportamiento del sistema. Por lo tanto, lo que se busca en esta etapa, es minimizar el tiempo en que el modelo no responde a los cambios presentados. Esto no es una tarea trivial, ya que no es simplemente detectar el comportamiento errado del modelo, sino que también va a implicar el tiempo de solución del problema, bien sea con un reentrenamiento o incluso el desarrollo de un nuevo modelo.

Así que dependiendo del riesgo implicado en las decisiones tomadas con base en las salidas del modelo, la medición y la retroalimentación veloz del sistema cobran mayor relevancia para determinar si el sistema funciona tal y como se esperaba [2]. El componente de monitoreo debe de estar en la capacidad de alertar con prontitud al equipo de datos, ya que es posible que incluso antes de que el sistema pueda avisar los cambios sufridos en los datos, a nivel de negocio, algunas de las salidas del modelo ya han tenido algún tipo de repercusión negativa. Por ende, en algunos casos, podría ser incluso necesario que el sistema sea capaz de suspender o detener la operación de la aplicación de forma autónoma para evitar salidas erróneas que impliquen pérdidas económicas, malos diagnósticos, impactos negativos en la reputación de la organización, entre otras graves consecuencias.

Esta fase de monitoreo puede ser dividida en dos ejes básicos: el monitoreo de la infraestructura, y el monitoreo de los datos y el modelo; a continuación se describirán brevemente, las estrategias utilizadas para una construcción adecuada de este importante componente.

3.4.3.3 Monitoreo de datos, modelo e infraestructura

Gracias a la monitoreo de métricas clásicas de la infraestructura de una aplicación, como el uso de CPU, memoria, disco, red, disponibilidad del modelo y su latencia, entre otros, se puede determinar rápidamente el posible origen de algún problema, y tomar acciones para solucionarlo y evitarlo en un futuro. Este monitoreo puede ser llevado a otro nivel en sistemas más robustos en los cuales se puede monitorear frecuencias de despliegue, tiempo necesario para llevar una nueva característica a producción, porcentaje de fallos en los despliegues y disponibilidad de la aplicación [11].

Pero este monitoreo, no sólo se reduce a la recopilación de las métricas heredadas del desarrollo de software tradicional, sino que también, se puede implementar monitoreos para detectar cambios en la frecuencia y cantidad de tipos de datos que entran al modelo [58]. Dependiendo de la magnitud de dichos cambios se podría definir una actualización de la infraestructura debido al aumento o reducción en la demanda. Además, se puede visualizar cuellos de botella en los componentes del pipeline, los cuales podrían ser optimizados para mejorar los tiempos de ejecución de los procesos.

En contraste a las aplicaciones de software tradicional, en las que el proyecto termina una vez ha finalizado la implementación de este, en las aplicaciones de machine learning, una vez es terminada la implementación, viene uno de los retos más grandes del proyecto, y es poder conservar el modelo preciso en el tiempo [27]. Como se ha mencionado anteriormente, uno de los principales problemas que se enfrentan en este tipo de proyectos es la naturaleza cambiante que tienen los datos, esto puede ser debido a muchos factores, como por ejemplo políticas comerciales, cambios en precios, reglas de negocio, desastres naturales, conflictos internos, etc. A dicho fenómeno se le conoce como drift o desviación de los datos, y es el responsable de que el modelo se vuelva menos precioso con el tiempo [59]. Por ello, gran parte del esfuerzo que se realiza en la implementación de los componentes de monitoreo se enfocan en la detección temprana del drift.

Para poder detectar dichos cambios, es necesario tener criterios que permitan contrastar los datos usados previamente en el entrenamiento y prueba del modelo, y también, los datos en producción que han sido suministrados al modelo para sus labores de inferencia en el tiempo de operación de la aplicación. Por lo tanto, es de suma utilidad la recopilación, generación y seguimiento de estadísticas descriptivas de los datos, el esquema de los mismos, y las suposiciones bajo las que fueron construidos los datasets consumidos.

Una vez se tiene acceso al expediente de datos, se puede detectar si los datos de entrenamiento, sus distribuciones, esquemas o suposiciones han mutado, bien sea en magnitud, frecuencia o distribución respecto a los datos en producción [27]. Asimismo, se puede hacer uso de métodos de detección de anomalías que aprenden de los patrones temporales de los datos, facilitando el proceso de identificación de dichos cambios, para luego tomar acciones correctivas; por supuesto, validando los beneficios que se puedan obtener al tomar estas acciones y qué costo tendrá llevarlas a cabo [47].

No solo se debe monitorear los cambios en las propiedades de los datos, sino también el modelo debe ser analizado cautelosamente. Así que se debe estar en la capacidad de comparar las métricas actuales del modelo, una vez se obtiene el ground truth, es decir el valor real que el modelo debería haber predicho y compararla con la inferencia hecha por el modelo; de esta forma, se tendrán los criterios para contrastar el desempeño de diferentes versiones de modelos con los datos de producción. Esto tiene una debilidad muy

grande, debido a que en ocasiones no es fácil ni rápida la obtención del ground truth. Por ello, puede existir una brecha temporal considerable entre la predicción del modelo y la obtención del valor correcto, y en este periodo, el modelo podría estar teniendo comportamientos o salidas erróneas que no serían detectables rápidamente. Por ello, implementar un modelo en el que el drift es más rápido que el tiempo necesario para obtener el ground truth, es bastante arriesgado.

Para mitigar un poco dicho riesgo, es de gran ayuda los mecanismos de detección de cambios en los datos, y también, muy importante, poder validar el performance del modelo respecto a algún KPI de negocio, como es el crecimiento en ganancias, seguidores, visitas, etc. Esto para constatar de alguna manera que las salidas del modelo están teniendo un impacto positivo en el negocio, pero igualmente, tiene la dependencia de que en ocasiones solo se puede calcular dicho impacto tiempo después de la predicción, una vez es conocido el ground truth.

Inicialmente, es útil establecer alertas que serán emitidas cuando los datos no coinciden con las expectativas o estadísticas recopiladas en el historial de datos del que se habló previamente. Estas alertas pueden ser activadas cuando se superan rangos mínimos o máximos de alguna variable determinada del conjunto de características, o también, si los datos entrantes superan cierto umbral de nulos, infinitos o si la diferencia entre las estadísticas de entrenamiento y los datos en producción tienen un diferencial mayor al que se definió previamente. Con estas notificaciones se podría establecer acciones correctivas dependiendo del nivel de alerta lanzado una vez se ha identificado esta desviación. Estas acciones podrían verse materializadas en forma de un reentrenamiento o entrenamiento continuo [26], una vez se tenga una cantidad considerable de datos nuevos. También es posible una redefinición de los umbrales ya que es posible que no se adapten o funcionen bien para el nuevo tipo de datos y que lancen falsas alarmas. Adicionalmente se pueden implementar restauraciones automáticas de versiones previas de modelos, la redefinición de la frecuencia de entrenamiento de los modelos, o incluso la modificación de todo el modelo o del proceso de construcción de los datos de entrenamiento [58] [60].

3.4.3.4 Registro del modelo

Con el objetivo de tener una trazabilidad del comportamiento del modelo, bien sea para fines de auditoría, debug, u otros, es necesario tener un registro de eventos de todo el sistema, los cuales deberían de ser tan detallados como sea posible. Entre los tipos de eventos y datos a registrar se encuentran los siguientes:

- Metadatos del modelo, como la fecha de entrenamiento, métricas obtenidas en el entrenamiento y validación, datos y características con los que se entrenó y se evaluó, con el fin de poder utilizar el mismo conjunto de datos para la comparabilidad de los modelos, algoritmo e híper parámetros utilizados en el entrenamiento, versión del modelo, transformaciones hechas sobre los datos, entre otras [61].
- Registro y versionamiento de los valores de las nuevas observaciones que entran al modelo, para poder aplicar las técnicas de detección de drift mencionadas en el monitoreo de datos.
- Las salidas o inferencias hechas por el modelo, deben de ser registradas con el objetivo de que una vez se pueda tener acceso al ground truth se pueda verificar el performance que está presentando el modelo tan pronto como sea posible, y así emitir alertas de forma temprana.
- Explicabilidad del modelo, ya que en algunas industrias existe una alta regulación por parte de políticas comerciales, o gubernamentales, como es el sector bancario, salud, seguridad, entre otros. En esta industrias se suelen realizar procesos de auditoría cada cierto periodo de tiempo, en las que se debería estar en la capacidad de poder explicar el origen de una predicción y qué características tuvieron una mayor relevancia en esta decisión para así explicar de cierta manera su comportamiento. Esto se hace principalmente con el fin de evitar sesgos en los modelos, sobreajustes o fugas de información.

3.5 Principios de la Metodología

Anteriormente se han detallado los niveles que conforman un proyecto típico de Machine Learning, y a su vez, los procesos que se deben ejecutar dentro de cada uno de ellos. Estos procesos puede variar dependiendo de la estrategia de datos utilizada por la

empresa. Sin embargo, existe un conjunto de prácticas heredadas de DataOps y MLOps que son altamente recomendadas o incluso de carácter obligatorio, dependiendo del nivel de madurez del sistema que se desee construir. Estas prácticas son totalmente independientes del tipo de proyecto, y su aplicación ayudará a evitar gran parte de los problemas mencionados que terminan ocasionando la gran deuda técnica recurrente en los proyectos de ciencia de datos. Dichos principios son: versionamiento, pruebas, automatización y reproducibilidad. A continuación se detalla cada uno de los principios, y cómo se deben aplicar correctamente en cada uno de los niveles del proyecto.

3.5.1 Versionamiento

En el desarrollo de software tradicional, una de las técnicas más importantes en la construcción de un proyecto es disponer de un sistema de control de versiones para la gestión de cambios de código. El uso de un sistema de versionamiento de código tiene grandes ventajas, como la facilidad de solución de conflictos, trabajos coordinados, colaborativos, y asíncronos, que permiten una gestión más segura de cada una de las partes del código de un proyecto [26]. Pero como se ha mencionado anteriormente, las aplicaciones de machine learning distan en algunos aspectos fundamentales respecto al desarrollo de software tradicional, y el versionamiento es uno de los que más discrepa y que puede tener una mayor repercusión en el éxito a largo plazo de un proyecto de machine learning.

En el desarrollo de software tradicional, cuando se cambia el código, el cual es el artefacto principal, se cambia la versión del proyecto, y en caso de que sea necesario revertir algún cambio o funcionalidad, bastará con restaurar un commit, y desplegar o usar dicha versión restaurada. No obstante, en las aplicaciones de machine learning, este proceso de rollback de alguna versión no es tan elemental, debido a que el código no es el único ni el principal artefacto en este tipo de proyectos. En este caso, los artefactos principales suelen ser modelos compilados que tienen dependencias directas con el conjunto de datos con el que se entrenó, el esquema y distribución de dichos datos, y por último el código que se encarga de crear dicho modelo, desde el proceso de transformación de datos hasta el entrenamiento de este [42]. Debido a estas dependencias, se debe el principio de “cambiar cualquier cosa lo cambia todo”, ya que cualquier modificación de alguna de dichas dependencias, representará una versión completamente diferente del proyecto.

Dicho esto, se evidencia que existe una necesidad de versionar el modelo, los datos y el código, debido al alto nivel de experimentación requerido. En dicho proceso los científicos suelen modificar arquitecturas, hiperparámetros, características utilizadas, etc; y es probable que deban revertir algunos de los cambios realizados, ya que estos no siempre coinciden con una mejora del performance. Por otro lado, además de tener la trazabilidad de estos experimentos y poder recuperar aquellos que obtuvieron un mejor desempeño, el versionamiento también provee la capacidad de llevar a cabo auditorías en cualquier momento, para cualquier punto de desarrollo de la aplicación de forma exitosa [24]. A continuación se detalla la implementación del principio de versionamiento en los niveles de datos, modelo y despliegue.

3.5.1.1 Datos

Es bien sabido que los datos tienen una naturaleza intrínseca cambiante, por ello no hay ningún argumento para esperar que los datos simplemente no muten en el tiempo. Por eso, con el fin de poder reproducir los diferentes pipelines del proyecto en cualquier instante, es necesario tener una clara trazabilidad de sus cambios y además de la procedencia de los mismos [45], por ello, los datos deben de considerarse como uno de los 3 ejes de versionamiento principales en un proyecto.

Para llevar a cabo esta tarea de versionamiento de datos, no basta con simplemente versionar los datos usados en el entrenamiento, la validación y las pruebas, sino también el esquema de los datos. Este puede definirse como el artefacto que describe las propiedades y estructura de los datos [3]. Muchos artefactos y procesos como las arquitecturas, transformaciones de datos, pruebas, bases de datos, etc., derivan y tienen dependencias directas con el esquema. Debido a este estrecho acoplamiento con muchos componentes de todo el pipeline, es que el esquema debe ser uno de los artefactos a versionar más importantes en el proyecto; esto brinda la capacidad de garantizar la compatibilidad de nuevas versiones de datos, modelos y otros artefactos con mayor velocidad.

Por otro lado, los datasets usados suelen ser de gran tamaño, y utilizar herramientas de control de versiones de código tradicionales como git, para el versionamiento de los datos es una mala práctica. Esto se debe principalmente a que estas herramientas no fueron

diseñadas para el control de versiones de artefactos de dicha magnitud; para llevar a cabo esta tarea es necesaria la capacidad de escalamiento del sistema, control de la metadata, además de poder conocer que versión de los pipeline de pre procesamiento generaron los datos con los que se entrenó el modelo, entre otras actividades que van más allá del alcance de sistemas como git [21].

Para llevar a cabo esta labor, el uso de herramientas dedicadas al almacenamiento de artefactos de gran volumen cobra gran importancia, pero no basta solo esto, sino que también se debe estar en la capacidad de relacionar dichos artefactos con el sistema de control de código, de esta forma poder relacionar inequívocamente, qué artefactos y qué código son los responsables de cierto resultado.

3.5.1.2 Modelo

Como se mencionó anteriormente, el modelo es el artefacto principal de los proyectos de machine learning, y claramente debe considerarse como otro eje de versionamiento en el proyecto debido a que es posible tener múltiples versiones de un modelo, incluso si se está utilizando el mismo dataset para entrenarlos. Dicho esto, es evidente que no basta simplemente con tener versionado el conjunto de datos y su correspondiente esquema [3], sino que también, se debe versionar el modelo en el formato definido, los híper parámetros utilizados para su construcción y todo el registro de experimentos que se ejecutó para la construcción del modelo, de allí la importancia del uso de las herramientas que registren y almacenan la trazabilidad de cada ejecución de entrenamiento del modelo.

3.5.1.3 Código

El versionamiento del código es de vital importancia ya que funciona de forma transversal a todo el proyecto, unificando los principales componentes de todo el sistema. Conceptualmente, los pipelines son construidos con una gran variedad de herramientas que tienen diferentes propósitos, desde ingesta de datos y su transformación hasta visualización al usuario final, todos estos elementos son esencialmente código [2].

El código controla todo el proceso de análisis de datos de un extremo a otro, permitiendo vincular los datos que se ha versionado anteriormente, con el pipeline de datos. Posteriormente, se puede asociar el artefacto principal con el pipeline de entrenamiento de

los modelos, de esta forma no solo se tiene la trazabilidad de cuál fue el resultado de los experimentos realizados para la creación de un modelo, sino que también se tiene el pipeline necesario para la creación de este artefacto [26].

Por último y no menos importante se debe de versionar el pipeline de despliegue, aquí se tienen todas las configuraciones y definiciones de infraestructura necesarias para reproducir fácilmente el mismo ambiente de desarrollo en cualquier instancia donde ha de desplegarse el modelo.

3.5.2 Pruebas

Las pruebas tienen como objetivo el poder identificar con mayor facilidad y rapidez problemas en el proyecto [24]; de esta forma, se podrán tomar acciones correctivas de forma más precisa, en el menor tiempo posible, antes de que un error de cualquier índole se extienda por toda la secuencia de operaciones de la aplicación. Es esencial que dichas pruebas tengan un alto grado de automatización, pues de lo contrario, la aplicación de este principio habría de convertirse en una carga adicional de trabajo. Esto, por causa de que el equipo tendrá que realizar una ejecución manual de un gran número de pruebas para poder garantizar la ausencia de errores en los pipelines, lo cual implicaría un retraso en el avance del proyecto. Por ello, una de las estrategias más utilizadas es que por cada nueva funcionalidad añadida al pipeline, simultáneamente se añada una prueba para ella; así, mientras avanza la implementación del proyecto, la calidad del mismo va incrementando gradualmente, asegurando la integridad de los resultados obtenidos [2] .A pesar de que el alto grado de automatización en las pruebas tiene una gran prioridad en este principio, es importante conservar algunas pruebas manuales en la fase de validación del pipeline de entrenamiento, con el objetivo de comprobar posibles situaciones de sesgo o inequidad en el modelo, y que sea el humano quien tome la decisión si un modelo puede pasar a un ambiente de producción o no [45].

Asimismo, como se han expuesto diferencias en otros principios y procesos respecto al software tradicional, las pruebas que se deben desarrollar en un proyecto de machine learning no son la excepción. Estas, son más complicadas que las pruebas construidas en el desarrollo tradicional, pues no sólo son requeridas las clásicas pruebas unitarias y de integración, sino que también, se deben realizar pruebas de validación de los datos, del performance del modelo, entre otras que se van a mencionar a más adelante [22].

Siguiendo el mismo orden de ideas del principio de versionamiento, las pruebas también deben ser implementadas en los 3 niveles fundamentales del proyecto.

En el pipeline de datos, se prueban los datos, sus características y las transformaciones que se aplican sobre ellos. En el pipeline de entrenamiento del modelo, se ejecutan pruebas y validaciones del performance del modelo obtenido, y por último en el pipeline de despliegue se realizan pruebas de integración y validación de la infraestructura [45]. En cada componente deben existir varios tipos de pruebas: unitarias, de integración y end to end; en cuanto al porcentaje del volumen de pruebas de cada tipo en el proyecto, lo ideal es que se tenga una distribución de 70%, 20%, 10% correspondientemente [62].

3.5.2.1 Datos

Las pruebas ejecutadas sobre los datos que han de ser consumidos posteriormente en los diferentes pipelines del proyecto son de suma importancia debido a varias razones. Una de ellas es que una detección temprana de un error en los datos podría ahorrar mucho tiempo y costo computacional; además, cuando estos errores no son detectados de forma temprana, la identificación de la raíz del problema podría tomar mucho más tiempo a los científicos de datos debido a la cantidad y complejidad de operaciones que han de ser revisadas [63].

Los tipos de pruebas que se suelen construir en el nivel de datos se puede agrupar en 2 tipos. El primer tipo, son pruebas que validen reglas de negocio, para verificar que se cumplen con requisitos previamente establecidos en la definición del proyecto, y suelen ser reglas duras, por lo tanto, bastante precisas en la identificación de problemas. El segundo tipo es la detección de anomalías sobre los datos, esta puede hacerse bien sea por técnicas de detección de outliers o por comparación histórica de fuentes de datos antiguas que han sido validadas [47].

A continuación describen algunas de las pruebas más importantes que deben de aplicarse en el nivel de datos para evitar dichos problemas:

- Cómo primer filtro para garantizar la calidad de los datos, es de gran utilidad realizar pruebas que permitan validar los datos de entrada contra el esquema esperado, el cual ha sido generado previamente y construido a partir de cálculos estadísticos de los datos que han sido corroborados en iteraciones previas. Esto con el fin de asegurarse de que los valores de entrada están sobre el dominio o rango esperado. En caso de que estas pruebas fallen, el científico de datos sea advertido, y que sea él el que tome la decisión, si es necesario modificar el esquema porque los datos son correctos y validados a nivel de negocio o que por el contrario, rechace la versión de los datos recientemente adquiridos.

- El siguiente paso, luego de la validación de las expectativas correspondientes al esquema, es la ejecución de las pruebas sobre los procesos encargados de la transformación de datos. Las funciones o scripts encargadas de dichas transformaciones deberían de ser probados mediante el uso de pruebas unitarias que puedan verificar que estos procedimientos han sido ejecutados correctamente. Un ejemplo de ello es verificar que luego de que una característica numérica ha sido normalizada, debería de tener sus valores mínimos y máximos entre 0 y 1 respectivamente, del mismo modo con operaciones como imputaciones, codificaciones, estandarizaciones etc.
- Es de gran utilidad tener a disposición un conjunto de datos muy simple, el cual no es tan necesario que deba ser versionado ni actualizado constantemente y será utilizado como insumo para múltiples pruebas en los pipelines. El objetivo de utilizar este dataset en las pruebas no es el de evaluar estrictamente rendimiento del modelo u otros elementos, aunque en ocasiones es normal el uso de bajos umbrales para validar métricas técnicas [3]. En este caso, actúa como caso base que permitirá una ejecución de múltiples pruebas rápidas con un costo computacional bastante reducido, y gracias a ello se puede obtener diagnósticos veloces que permitirán identificar errores no tanto a nivel del performance final del modelo, sino problemas lógicos de base, de programación o incluso en cómo se están introduciendo los datos a la aplicación.

- A diferencia del conjunto de datos base mencionado anteriormente, el cual no representaba ninguna dificultad o reto para los pipelines, tiene un gran beneficio utilizar diferentes conjuntos de datos que, por el contrario, contengan situaciones

atípicas extremas. Esto puede darse en forma de valores extremos, variables de un tipo diferente al definido en el esquema de datos, o simplemente que tengan una gran cantidad de valores nulos. A pesar de que estas situaciones rara vez van a ser experimentadas por el modelo en producción; tienen el objetivo de protegerlo frente a estas situaciones y que la aplicación no vaya a fallar al enfrentar datos de esa naturaleza en producción.

- Pruebas de gran simplicidad podrían también tener un gran impacto, como por ejemplo un recuento de filas entre operaciones para garantizar una ejecución adecuada de productos cartesianos, join entre tablas, adición de características, etc [11].

Un punto muy importante a tener en cuenta es que si durante todo el proceso de desarrollo del modelo se ejecutan las pruebas y se calculan las métricas con exactamente el mismo conjunto de datos, el modelo se podría estar sobre ajustando con dicho conjunto. Por ello, es de vital importancia tener una correcta gestión de pruebas de datos, y procurar que la ejecución de las pruebas sean realizadas preferiblemente con datos adquiridos recientemente, es decir que sean un muestra representativa de los datos que se encuentran en producción [64]. Pero por otro lado, si constantemente se cambian los datos, será difícil determinar si los diferentes resultados obtenidos en algún pipeline, se deben a un problema a nivel de código o a nivel de datos [2]; por esto, una buena estrategia es cuidadosamente ir actualizando este dataset en un periodo de tiempo no muy lejano, pero tampoco con frecuencias diarias o menores.

3.5.2.2 Modelo

Dado que el modelo es un artefacto no determinista, las pruebas que se deben de hacer sobre ellos, también distan un poco de las pruebas realizadas en las aplicaciones de software tradicional. Por ello, el objetivo de esta fase es tratar de minimizar este factor no determinista sobre los modelos tanto como sea posible. A continuación se listan algunas de las pruebas más útiles en este nivel:

- Con el fin de evaluar un modelo se suele recopilar varios tipos de métricas; una vez obtenidas dichas métricas y con el fin de verificar la calidad del modelo antes de que este pueda ser promovido a un ambiente de producción, la prueba base que se suele construir es la definición de un threshold en la evaluación [65]. Este, tiene como objetivo garantizar que el nuevo modelo entrenado posea un performance superior al que se encuentra en producción y que cumple un criterio base de performance definido previamente. Un aspecto clave que debe de tratar de establecerse en esta fase, es tener la capacidad de relacionar las métricas técnicas de desempeño del modelo con las métricas de impacto real de negocio. El objetivo de esto, es poder establecer un criterio menos abstracto, que a su vez brinde la capacidad de dimensionar de alguna manera cuál será el impacto del modelo en el negocio, y tomar una decisión más acertada a nivel comercial [63].

- Es posible que al evaluar el modelo con las métricas anteriormente mencionadas, no se pueda tener un detalle más profundo del performance en diferentes subpoblaciones de los datos. Por ello, es altamente recomendado evaluar el modelo en slices; de esta forma se podría encontrar problemas de sesgo y equidad que son provocados debido a la falta de datos de dicha subpoblaciones o fallos en la recopilación de los datos [66].

- Uno de los problemas más frecuentes y difíciles de detectar, es que las operaciones necesarias para la transformación de las características que se transmiten al modelo puedan diferir en la etapa de entrenamiento y producción. Para probar que no se está cayendo en dicho error, se utiliza un conjunto de datos aislado y se envía junto con el modelo empaquetado, para evaluar el rendimiento que se obtiene al procesarlo en la fase de producción. Posteriormente se contrastará con el desempeño obtenido en la fase de entrenamiento y validación para determinar si esta disparidad se encuentra presente en el pipeline de datos. Por ello, una de las prácticas más recomendadas es asegurarse de utilizar el mismo pipeline de transformación de características en el entrenamiento y en la fase de producción.

- Es posible probar la arquitectura básica del modelo de forma bastante sencilla. Consiste en crear un conjunto de datos ficticio que cumplan con la estructura de datos proporcionada por el esquema de datos para luego simplemente hacer una

predicción sobre el modelo. De esta forma se puede verificar que el suministro de datos se dio de forma exitosa sobre la arquitectura y que la salida del modelo cumple con la forma esperada.

3.5.2.3 Código

Por último, está el componente de las pruebas de código, que tienen como objetivo principal garantizar el correcto funcionamiento de los diferentes pipelines e infraestructura utilizada en el proyecto y a su vez, la integración correcta de los diversos componentes del sistema. A continuación se listan algunas de las pruebas más comunes y útiles de este nivel.

- Para verificar que el algoritmo de optimización utilizado en el entrenamiento del modelo está funcionando, se generan datos aleatorios con el esquema esperado por el modelo, y se ejecutan un par de iteraciones de entrenamiento. El fin de este no es entrenar completamente el modelo y que converja de forma exitosa, sino, para asegurarse que está funcionando correctamente a nivel lógico el algoritmo. También sirve para verificar que al culminar el entrenamiento, se está almacenando de forma exitosa el artefacto correspondiente al modelo en el formato especificado.

- También se debe de crear pruebas de integración que se ejecuten regularmente sobre todos los pipeline. En estas pruebas se validan que tanto el pipeline de datos, entrenamiento y despliegue termina con éxito cada una de sus operaciones. Estas pruebas deben de estar automatizadas tanto como sea posible, y habitualmente serán ejecutadas antes de hacer merge de nuevas características a la rama principal y también antes de llevar el modelo a producción. Estas pruebas también se detallarán en profundidad en el principio de automatización, donde los componentes de integración continua y despliegue continuo juegan un papel esencial en la ejecución de estas [63].

3.5.3 Automatización

Uno de los objetivos del uso de MLOps en un proyecto, es automatizar procesos de desarrollo de un modelo de ML e integrarlos en la aplicación final como un componente o servicio de forma eficiente [24]. DataOps por su parte, también propone un nivel de automatización elevado en sus operaciones a través del pipeline de datos. De esta forma

se puede definir la madurez de una aplicación de machine learning que hace uso de estos dos enfoques según el nivel de automatización que proporciona. Esto se puede ver reflejado en la velocidad de entrenamiento de nuevos modelos, la adquisición e integración de nuevos datos, y el monitoreo y detección oportuna de errores en el sistema [22].

MLOps y DataOps incluyen algunas de las prácticas provenientes de DevOps, entre ellas están la integración continua (CI) y el despliegue continuo (CD). Ambas son una de las piedras angulares de DevOps, pero adicionalmente, MLOps suma el monitoreo continuo (CM) del que deriva el entrenamiento continuo (CT) [24]. Estas dos últimas se introdujeron previamente en el componente de monitoreo, por ello solo se van a desarrollar a detalle CI/CD.

CI/CD son un conjunto de prácticas provenientes del desarrollo ágil y de DevOps que permiten desplegar cambios y nuevas funciones dentro de la aplicación con mayor frecuencia y velocidad [47]. De esta forma se garantiza una alta calidad y estabilidad del sistema, reduciendo al máximo el riesgo proveniente de integrar nuevos desarrollo al mismo [3]; en este caso, aunque existen muchas similitudes, estas prácticas bajo los enfoques de MLOps y DataOps distan nuevamente de su uso respecto a las aplicaciones de software tradicional.

Algunas de las diferencias que tienen una mayor disparidad se deben a la necesidad de soportar una alta capacidad de escalamiento que deben de tener cada uno de los procesos de los pipelines. Por otro lado, las pruebas que son ejecutadas de forma automática sobre los datos cobran gran importancia, y a diferencia del software tradicional, no es práctico crear entornos de pruebas bajo demanda cada vez que se ejecuta el pipeline de CI/CD debido a que puede ser un proceso bastante dispendioso computacionalmente. A continuación se abordan las prácticas de CI/CD con un poco más de detalle.

3.5.3.1 Integración continua

A medida que se suman más procesos a la aplicación, y estos a su vez se entrelazan y generan dependencias entre ellos, se vuelve crucial que el proceso de integración se gestione y administre de forma sistemática y automatizada [67]. Este proceso debería de seguir operaciones estrictas, que permitan garantizar que los incrementos no van a introducir errores dentro del proyecto; para lograr este objetivo se utiliza la integración

continua. Dicho esto, se puede establecer que el objetivo principal de la integración continua es evitar el esfuerzo por parte del equipo para fusionar los avances de su trabajo sin introducir errores en este proceso [3].

El proceso de integración comienza en el momento en el que algún miembro del equipo hacen un commit de sus cambios en el repositorio central. Este commit es el que da ignición al pipeline de CI, el cual compila automáticamente la aplicación y ejecuta verificaciones de estándares de desarrollo, supervisión de cambios en los datos para generar alertas sobre ello, y por supuesto, la ejecución de las pruebas que se han establecido previamente. Una vez dichas pruebas han sido superadas, los cambios son aceptados en el repositorio o de lo contrario se notifica que las pruebas han fallado, y no se aceptan los cambios; seguidamente, el equipo debe comenzar un proceso de búsqueda del origen de los problemas para solventarlos tan pronto como sea posible. De esta forma se garantiza en cierta medida que no se ingresan errores al proyecto mediante cambios, avances o datos defectuosos.

Para que la integración continua funcione efectivamente, es necesario que el equipo adopte la práctica de enviar sus avances al repositorio con la frecuencia más alta posible, en el peor de los casos, una vez por día. De lo contrario, el proceso de administrar despliegues con una menor frecuencia, pero que contienen incrementos mucho más grandes y sofisticados, hace que sea más difícil de gestionar su proceso de integración con los avances del proyecto. La causa de esto, es que al tratar de integrar muchos cambios, el sistema podría comportarse de una forma inesperada y aislar el origen del problema podría ser más complejo y tomarle más tiempo al equipo [2].

3.5.3.2 Despliegue continuo

Una vez se ha ejecutado todo el pipeline de CI, todo el conjunto de pruebas ha sido superado y los nuevos avances han sido integrados al proyecto con éxito, y el despliegue continuo toma acción. CD brinda la capacidad a cada uno de los miembros del equipo de llevar cualquier tipo de cambios a producción; esto se hace de forma automatizada, rápida, eficiente y garantizando siempre una alta calidad del producto entregado en cualquier momento del tiempo [68].

Habitualmente, se trabaja conjuntamente con la práctica de CI, permitiendo que el equipo pueda desplegar y entregar al usuario final cada nuevo pequeño incremento que se hace en el desarrollo a un ambiente de producción. De esta forma, se puede obtener una versión totalmente funcional en cualquier instante del tiempo rápidamente, además de brindar la visibilidad y control de los niveles de variación de cada adición. Esto otorga al equipo la capacidad de tomar la decisión de publicar o no una versión de la aplicación, ya no por un criterio técnico, sino puramente comercial, de esta forma alcanzando esa anhelada capacidad de adaptación a los cambios con gran rapidez [45].

En síntesis, cada vez que se ejecuta el pipeline de CD, se crea un ambiente de prueba con características lo más cercanas posibles al ambiente de producción. Luego, se ejecutan las pruebas de integración y de aceptación, para verificar que la aplicación cumple con los requisitos de alto nivel del usuario; y una vez superadas dichas pruebas, se procede a generar una distribución o release de la aplicación para ser usada o entregada al usuario final.

3.5.3.3 Orquestación

Un componente adicional que hace parte del principio de automatización es la orquestación, está técnica es transversal a todo la aplicación y se hace necesaria para reducir la complejidad asociada a la ejecución de los múltiples pipelines que coexisten en los proyectos de machine learning, como es el pipeline de datos, entrenamiento y despliegue. Cada una de dichas tareas están compuestas por diferentes procesos, que a su vez hacen uso de múltiples herramientas, haciendo que la ejecución de cada proceso de forma manual sea una actividad bastante dispendiosa, y por supuesto, propensa a cometer errores [47]. Adicional a ello, muchas de las tareas que se ejecutan tienen una alta dependencia con otras tareas, y como si esto no fuera suficiente, algunas de ellas se ejecutan en temporalidades muy diferentes al resto del pipeline, por lo que deben de ser agendadas para una ejecución automática en su respectiva frecuencia [69].

Para dar solución a estas dificultades, muchos equipos optan por crear código personalizado que permita cerrar la brecha de comunicación y orquestación entre cada componente de los pipeline de machine learning. Pese a esto, Google concluye que esto suele ser otra de las razonas por las que los proyectos de machine learning fallan; esto se debe al hecho de que dichos desarrollos son frágiles, carecen de capacidad de

generalización, calendarización, escalamiento y manejo de errores [4]. En vista de esto, se hace evidente la necesidad de contar con herramientas sólidas que permitan ejecutar de forma confiable esta red de procesos altamente entrelazados en el momento adecuado.

Dicha red puede ser modelada mediante un grafo acíclico dirigido (DAG), donde los nodos representan el proceso a ejecutarse y las aristas denotan el flujo de información. Las conexiones presentes en el grafo son directas ya que los datos no pueden ir en ambos sentidos, es decir la salida de un proceso es la entrada de otro, pero no en sentido contrario [2]. Además, se debe plasmar el orden de ejecución de las tareas y sus correspondientes dependencias [69], para así evitar situaciones en donde alguna tarea se ejecute sin todas las dependencias completamente calculadas [43].

Adicional a estas características, se debería también estar en la capacidad de visualizar el progreso y flujo de datos en el grafo, activar el auto escalado para gestionar cientos de flujos simultáneamente [32], autenticar el acceso a la gestión del grafo, implementar políticas de reintentos flexibles, estar en capacidad de reutilizar componentes, utilizar triggers de diferentes tipos (como aquellos basados en tiempo, basados en dependencias de datos que detectan la creación o actualización de archivos, cambios en tablas de la base de datos, así como también la detección de la finalización o el fracaso de los ejecuciones previas) [47]. De esa forma se logra asegurar la reproducibilidad y auditabilidad de cada uno de los componentes de los pipelines [70], alivianado la carga del equipo y optimizando la ejecución de esta gran cantidad de procesos.

3.5.4 Reproducibilidad

Como se ha mencionado anteriormente, el desarrollo de una aplicación de machine learning requiere procesos altamente iterativos y en donde usualmente cada decisión importante está soportada por algún tipo de investigación o experimentación que permita justificar dicha decisión [24]. Algunas de estas investigaciones tienen el propósito de evaluar qué características y operaciones de procesamiento permiten modelar la realidad con mayor precisión; entre ellas, está la búsqueda de los mejores hiperparámetros, ajustar el equilibrio entre sesgo y varianza en el entrenamiento, cambios de variables, métricas, entre muchas otras operaciones [3].

Para que el equipo pueda llevar a cabo estas operaciones debe estar en la capacidad de ejecutar experimentos en paralelo, para así, iterar rápidamente por cada una de las posibilidades de cada etapa implicada en la construcción de la aplicación de machine learning [45]. La velocidad y agilidad necesarias para llevar a cabo esta actividad cobra gran importancia, ya que el entorno, los datos y otras variables pueden cambiar con facilidad en cualquier momento, haciendo incluso que en ocasiones sea necesario repetir todo o parte de este proceso cada vez que algo cambia. Pero como se ha descrito, la cantidad de variable susceptibles al cambio en cada etapa puede llegar a ser abrumadora haciendo de esta tarea una de las más difíciles de rastrear con precisión en todo el proceso desarrollo.

Para poder ejecutar con éxito las tareas de experimentación e investigación, es de gran utilidad que el equipo defina con anticipación un presupuesto computacional o de tiempo que se va a dedicar para experimentación, así como también el umbral o criterio de aceptación que cumpla con las expectativas técnicas o de negocio previamente establecidas. Una vez fijados dichos criterios, se debe encontrar la forma adecuada de realizar un seguimiento de cada uno de los experimentos que se van a ejecutar y sus correspondientes resultados. Debido a la complejidad antes mencionada, hacer un seguimiento de esta actividad de forma manual y descentralizada es bastante engorroso y producirá información que no es para nada confiable. El enfoque correcto para afrontar estas dificultades, es hacer uso de métodos automatizados para registrar y gestionar todos los experimentos de cada componente de forma centralizada [21], brindando transparencia entre los integrantes del equipo y permitiendo un análisis comparativo de cada resultado con suma facilidad al registrar las mejoras iterativas a lo largo del tiempo.

El fin último del uso de esta práctica, además de registrar cada una de estas acciones es poder reproducir y repetir con éxito algún experimento que ha alcanzado resultados prometedores. Reproducir un experimento implica poder tener acceso a cada configuración y parámetro que se usó para lograr dicho resultado, para poder reproducirlo exactamente igual; en caso de que eso no sea posible, el equipo y la aplicación perderán credibilidad [26], y los esfuerzos realizados previamente para alcanzar dichos resultados habrán sido en vano.

Es aquí donde se puede ver que el concepto de reproducibilidad en MLOps y DataOps dista un poco de la definición del ámbito académico. En este último, se define como que un experimento y sus hallazgos son descritos con tal detalle que otra persona o equipo son capaces de replicar este mismo experimento y obtener el mismo resultado o conclusiones. La definición de MLOps y DataOps coincide y se solapa en gran medida con esta anterior, en el hecho de que se requiere la capacidad de volver a ejecutar fácilmente el experimento exactamente de la misma forma. Sin embargo, también es necesario tener acceso y detalle de los supuestos, del control de la aleatoriedad de los métodos y algoritmos usados, a los datos con los que se entrenó y validó el modelo, el detalle de la implementación y documentación de cada pipeline. Complementario a lo anterior también se debe de tener un listado de los resultados obtenidos, métricas, gráficos, kpi's, entre otros aspectos que son fundamentales no solo para obtener el mismo resultado sino también para la comparabilidad y auditabilidad de la aplicación.

Por otra parte, los sistemas de control de versiones de datos y código facilitan considerablemente la labor de obtener el código y datos responsables de algún resultado, ayudando en gran manera a cumplir con el principio de reproducibilidad. Pese a esto, no basta con tener acceso a los mismos datos y código, sino que también se debe poder recuperar la especificación precisa del entorno en el que se ejecutó dicho experimento. Esto tiene una gran importancia debido a que cualquier variación mínima del sistema operativo, versión de los frameworks o librerías utilizadas, dependencias del sistema y otros componentes del entorno, podría alterar enormemente el resultado del experimento, incluso si se ha replicado exactamente los mismos pasos, configuraciones y datos antes usados.

Tradicionalmente el equipo de IT se ha encargado de preparar manualmente los entornos de desarrollo, pruebas y producción, pero esto puede desencadenar errores debido a posibles diferencias en los entornos que se acaban de mencionar. Pero, gracias a las prácticas utilizadas en DataOps y MLOps heredadas de DevOps, el equipo puede hacer uso de técnicas como la contenerización, que van a permitir empaquetar todas estas dependencias y configuraciones específicas del entorno de una forma muy efectiva y aislada. Por lo tanto, independientemente de la infraestructura donde se despliegue la aplicación, se podrá tener garantía total de poder ejecutar y obtener los mismo resultados obtenidos en los experimentos [3].

Estos contenedores son construidos a partir de scripts que contienen todas las dependencias necesarias del entorno; además, se debe recalcar que al tener la definición de la infraestructura y configuración como código, se puede combinar esta estrategia con el control de versiones y así garantizar que los entornos de producción y desarrollo se mantendrán sincronizados porque serán construidos siempre a partir del mismo código y al estar centralizados hacer actualizaciones en la infraestructura o ambiente será sumamente fácil. Otra gran ventaja que ofrece el uso de contenedores en la fase de desarrollo, es que el equipo estará en la capacidad de reutilizar componentes que otros miembros del equipo desarrollaron y con los que no están necesariamente familiarizados, sino que basta con conocer que recibe y retorna dicho componente y de esta forma se facilita la sinergia y trabajo en paralelo del equipo.

3.6 Conclusión

En este capítulo se detallaron cada una de las fases principales de la integración de ambos enfoques y los pasos y recomendaciones ha seguir en cada uno de ellos. También se especificaron los principios esenciales que permitirán mitigar en gran medida la deuda técnica descrita anteriormente en los proyectos de machine learning.

4 Comparación práctica entre metodologías

El objetivo es exponer de forma práctica una aproximación a la metodología necesaria para transformar una aplicación que ha sido desarrollada con base en una de las metodologías tradicionales más usadas como CRISP-DM. Esto se hará haciendo énfasis en las diferencias de las fases técnicas de ambas metodologías, ya que las labores de entendimiento de negocio, definición de KPI's y métricas de negocio se atribuyen en mayor medida a las actividades realizadas en las metodologías ágiles utilizadas por el equipo de trabajo lo cual no hace parte del alcance de este trabajo.

Se debe agregar, que en el desarrollo correspondiente a la metodología tradicional, se ignoró muchas de las buenas prácticas de ingeniería de software que deben de estar presentes en cualquier proyecto de machine learning. Esto se hace con el propósito de reflejar de forma más fidedigna, los desarrollos hechos en la industria por los científicos de datos, que como se ha mencionado previamente, no suelen ser ingenieros de software y desconocen muchas de dichas prácticas; por lo tanto, cometen errores que terminan en la adición de más deuda técnica al proyecto.

Dicho esto, se procederá a enmarcar el desarrollo de un sistema de trading bajo los enfoques de DataOps y MLOps y comparar los beneficios que traen dichos enfoques sobre un problema real como el pronóstico de series de tiempo de mercados financieros respecto a su implementación bajo una metodología de desarrollo tradicional. Acorde con el planteamiento de la integración de ambos paradigmas, se mostrará el desarrollo correspondiente a cada uno de los 3 niveles y los principios transversales que hacen parte de MLOps y DataOps, y simultáneamente, se contrastará de forma práctica con el desarrollo tradicional de esta aplicación. Al final del capítulo se encuentran los enlaces a los repositorios correspondientes a cada desarrollo.

4.1 Descripción conceptual del modelo de trading

El objetivo de esta sección es describir conceptualmente el modelo de trading desarrollado junto con la estrategia y técnicas usadas en el desarrollo de este.

En primer lugar se obtienen los datos de la fuente original, en este caso el proveedor de datos financieros es AlphaVentage, la cual es una API que proporciona datos globales de mercados financieros. En este caso se usaron datos diarios de los últimos 20 años de la compañía, acción o activo a pronosticar.

Luego se procede a crear las características generando múltiples indicadores financieros para cada día, en diferentes tamaños de ventana. Estos indicadores son señales o patrones producidos por las variables base como el precio de apertura, el volumen de transacciones, etc. Al proporcionar esta información al modelo, se podrían identificar y predecir futuros movimientos de precios de mercado y así tomar la decisión de crear una entrada o no hacer nada. Algunos de los indicadores más populares son el Índice de Fuerza Relativa (RSI), el Índice de Flujo de Dinero (MFI), la divergencia de convergencia de promedio móvil (MACD) y las Bandas de Bollinger [71].

Una vez se generan todas las características nuevas se procede a seleccionar las 225 características más relevantes que proporcionen una mayor explicabilidad al modelo, para ello se usó dos algoritmos f_classif y mutual_info_classif de la librería Scikit-learn. Es importante resaltar que podrían ser más o menos características, lo importante aquí es que el número de características al sacar una raíz cuadrada se obtenga un número entero como resultado. El propósito de esto es que posteriormente se formará una imagen con dimensiones NxN, donde N, es la raíz cuadrada del número de características seleccionadas.

Posterior a ello se realiza uno de los pasos más importantes en cualquier estrategia de trading automático y es la transformación de los datos crudos a datos que pueda consumir un modelo de aprendizaje supervisado, en otras palabras es la generación de la variable objetivo o etiquetado de datos.

Para ello se usó el algoritmo de etiquetado sugerido por Omer Berat Sezer y Murat Ozbayoglu [72]. Este, etiqueta cada registro en 3 diferentes tipos de operaciones, BUY, SELL y HOLD, los cuales se podrían describir como que el precio va a SUBIR, BAJAR o MANTENERSE respectivamente, transformando el tipo problema de machine learning a clasificación multi clase.

```
procedure LABELLING()
    windowSize = 11 days
    while(counterRow < numberOfDaysInFile)
        counterRow++
        If (counterRow > windowSize)
            windowBeginIndex = counterRow − windowSize
            windowEndIndex = windowBeginIndex + windowSize − 1
            windowMiddleIndex = (windowBeginIndex + windowEndIndex)/2
            for (i = windowBeginIndex; i <= windowEndIndex; i++)
                number = closePriceList.get(i)
                if(number < min)
                  min = number
                  minIndex = closePriceList.indexOf(min)
                if(number > max)
                  max = number
                  maxIndex = closePriceList.indexOf(max)
            if(maxIndex == windowMiddleIndex)
                result = "SELL"
            elif(minIndex == windowMiddleIndex)
                result = "BUY"
            else
                result = "HOLD"
```

Figura 5 Algoritmo de etiquetado

Los siguientes pasos son la partición de datos en entrenamiento, validación y pruebas, para luego aplicar las técnicas de escalamiento de datos, en este caso se aplicó MinMaxScaler.

Una vez los datos ya son aptos para ser consumidos por los algoritmos de entrenamiento de machine learning, se procede a convertir cada registro a imagen. El propósito de ello es poder entrenar una red neuronal convolucional (Ver Figura 7), la cual suele usarse para la clasificación e identificación de patrones en imágenes, en este caso se trata de una imagen de 15x15, que como se mencionó anteriormente dependerá exclusivamente del número de características e indicadores técnicos generados.

Figura 6 Ej. de características diarias financieras a Imagen

La finalidad de esta transformación es tratar de simular la estrategia en la que la mayoría de traders profesionales operan, la cual consiste en identificar patrones visuales de las series de tiempo de los activos, combinadas con algunos indicadores técnicos, de esta forma en lugar de estar haciendo cálculos financieros o realizando análisis fundamentales lento, de gran complejidad y alta subjetividad, se toman decisiones de operaciones de forma práctica y veloz.

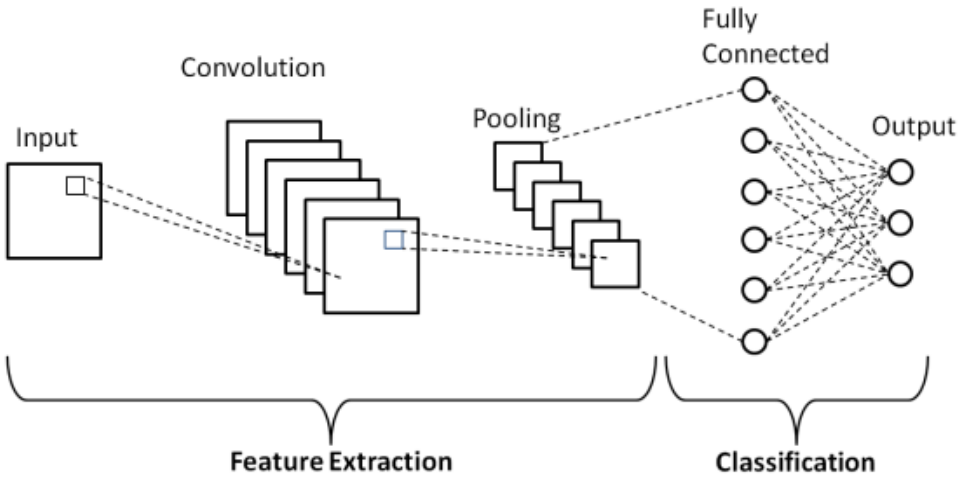

Figura 7. Tipo de arquitectura utilizado

Seguida a la transformación de los datos, se entrena el modelo sobre la red convolucional y se evalúa el modelo obtenido. Algunos de los resultados obtenidos en las pruebas realizadas son los siguientes:

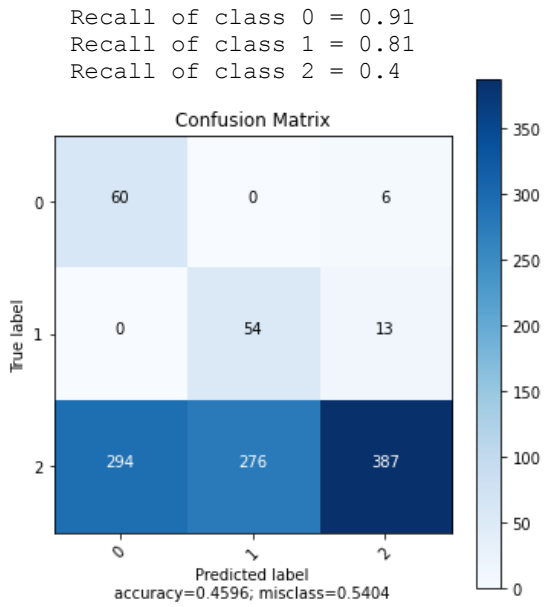

Figura 8. Matriz de confusión de modelo de clasificación

Al interpretar técnicamente este resultado, no se obtuvo un buen modelo en términos regulares para clasificar correctamente cada una de las clases, teniendo incluso una precisión bastante baja. Pero si se interpreta este resultado bajo un contexto financiero, se puede ver que el modelo es bastante cauteloso y certero en el momento de tomar operaciones que implican riesgo financiero.

Esto se puede notar al ver en como el modelo clasificó la notable mayoría de entradas como clase 2 (“Hold”), pese a que muchas de estas predicciones eran erradas. Por otra parte, la clase 0 y 1 (“Buy” y “Sell”) tuvieron muy pocas clasificaciones en la salida del modelo, pese a esto la métrica de recall de ambas fue bastante elevado. Esto se puede traducir en cuantas (%) de las entradas que recibía el modelo para estas clases se logró identificar de forma correcta. En síntesis, se podría decir que el modelo tiene una estrategia de operación bastante controlada o de bajo riesgo, al omitir entradas de forma precipitada en operaciones que no está seguro, por lo tanto ha de entrar muy pocas veces a alguna

operación de tipo compra o venta, pero aquellas en las que está seguro, lo hace de forma bastante certera.

Para cambiar esta estrategia a una quizá más arriesgada y que cuente con un número mayor de operaciones en un tiempo determinado, se deberá modificar la estrategia de etiquetado de los datos, ya que de ella depende en gran medida el éxito o fracaso del modelo y de la frecuencia de operación de este.

4.2 Comparación técnica del desarrollo de ambos enfoques

A continuación se describirá y comparará el desarrollo tradicional y el desarrollo realizado bajo el marco de la integración de los enfoques de MLOps y DataOps de la aplicación de trading automático bajo un enfoque más práctico y técnico. Algunos de los procesos que han de describirse en esta sección serán detallados de forma más especifica en la sección 4.3.

Iniciando el pipeline de datos se encuentra la adquisición de los datos, en la metodología tradicional se lleva a cabo ejecutando un notebook con el código para la conexión a la API, y seguidamente el almacenamiento en formato csv de los datos crudos recientemente adquiridos. Este proceso presenta una deuda técnica en el hecho de que no contiene ninguna verificación o prueba que permita detectar prematuramente si los datos recientemente adquiridos contienen errores. Así, que para corregir y disminuir dicha deuda, la ejecución de este se cambia a un archivo bash que ejecuta los scripts necesarios para la adquisición de dichos datos, el objetivo de esto es facilitar su ejecución y calendarización del proceso, y de forma automática poder ejecutar varias pruebas sobre los datos recientemente adquiridos, las cuales tienen el potencial de ahorrar una gran cantidad de tiempo al equipo en la búsqueda de errores o problemas provenientes de los datos. Dicho script contiene otros elementos que de igual manera marcan una diferencia importante, pero serán detalladas cuando se ejemplifique y detalle el principio de versionamiento.

Una vez adquiridos los datos, el próximo paso es la exploración y entendimiento de los datos, en ambos desarrollos (metodología tradicional e integración de MLOps y DataOps), se hace este análisis en jupyter notebooks, debido a la practicidad que proporciona el uso de esta herramienta ya que se puede relacionar código, gráficos y textos de forma

centralizada, lo cual resulta bastante conveniente al momento de hacer conclusiones, suposiciones y gráficos que permiten un rápido entendimiento e interpretación de los datos.

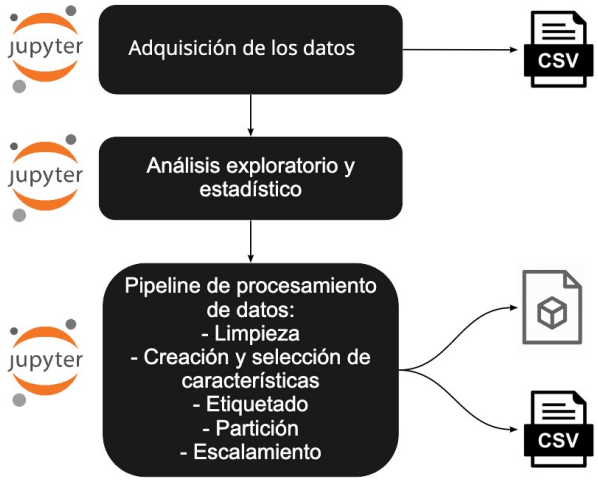

Figura 9. Pipeline de Datos Metodología tradicional

El próximo paso a seguir en la metodología tradicional es la limpieza de los datos, creación de características, partición y transformación de los datos, todas estas actividades son realizadas en un notebook el cual generará los datos listos para el entrenamiento del modelo y artefactos como el archivo asociado al escalamiento de los datos y las características seleccionadas.

En el desarrollo realizado bajo la integración de los enfoques, existen varios procesos y verificaciones adicionales antes de iniciar la etapa de entrenamiento del modelo. La primera diferencia está en la ejecución de pruebas de calidad sobre los datos y los métodos de versionamiento, los cuales serán detalladas en el principio de pruebas y versionamiento respectivamente. Posterior a esto, toma protagonismo una de las principales herramientas usadas en el proyecto, llamada TensorFlow Extended (TFX), la cual es una plataforma open source end-to-end para el despliegue a producción de pipelines de machine learning [73].

El primer componente del pipeline construido en TFX es el *ExampleGen* y es el encargado de consumir servicios o archivos externos para ingestarlos al pipeline los datos, en este caso los datos generados luego de la generación de características y particionamiento de

datos, estos datos son transformados a TFRecords los cuales permiten el entrenamiento de modelos con grandes volúmenes de datos que sobrepasan la capacidad de computo de la mayoría de equipos usados para el desarrollo de modelos.

Luego se ejecuta un componente llamado *StatisticsGen*, que se encarga de generar estadísticas básicas en cada partición declarada, dichas estadísticas serán el insumo de otros dos componentes, el primero será el *SchemaGen*, componente que cobra gran importancia ya que genera un esquema de los datos y que codifica las propiedades que se espera que satisfagan los futuros datos de entrada, y describe el tipo de las features, valores y dominio.

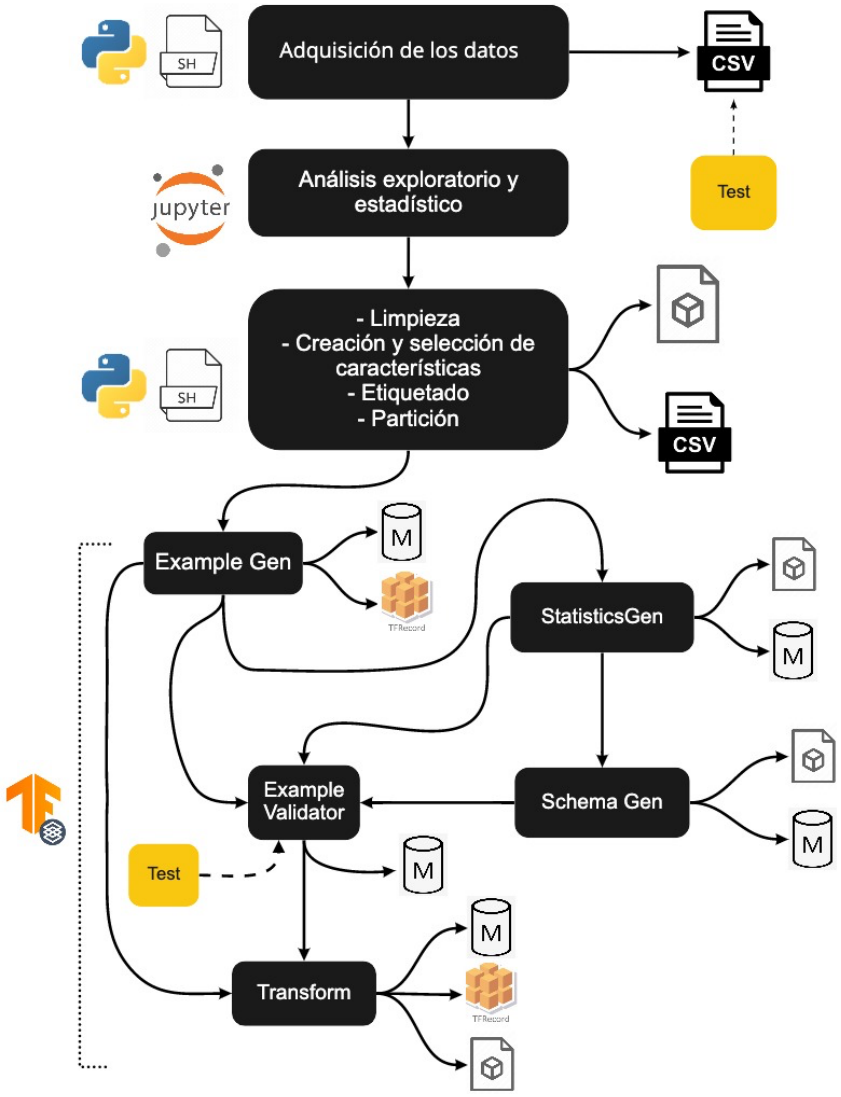

Figura 10. Pipeline de Datos en DataOps y MLOps

Estos dos componentes serán una entrada al componente *ExampleValidator,* en el cual se detectarán posibles anomalías, como contrastar los datos entrantes y el esquema previamente generado para probar que los datos nuevos encajan con los datos con los que se ha entrenado anteriormente algún modelo, es decir que cumple con las expectativas de los datos anteriores, detecta posibles cambios en las distribuciones entre las particiones generadas o respecto a datos anteriores de entrenamiento, así como también el drift en los datos al comparar cambios de las estadísticas generadas a través del tiempo.

El siguiente componente del pipeline a ejecutar es el *Transform* el cual se encarga de ejecutar la tareas de escalamiento, además guardan estadísticas sobre los datos antes y después de ser transformados, pero la verdadera ventaja que ofrece este componente es que garantiza que las transformaciones hechas sobre los datos de entrenamiento son almacenadas en el grafo del pipeline, por lo tanto serán las mismas que se harán sobre los datos de producción, eliminando otra posible fuente de sesgo, ya que a pesar de que en la metodología tradicional se almacenó el artefacto encargado del escalamiento, nada garantiza que se seleccione el artefacto adecuado, o actualizado en el momento de realizar las transformaciones en producción.

Es importante resaltar que en cada componente de TFX y en cada ejecución se registran metadatos asociados de cada componente, estos metadatos contienen información de su ejecución, los artefactos que se han generado, errores inesperados, entre otros, que van a permitir tener un registro confiable del proceso de construcción del pipeline y dado el momento poder depurar problemas con mayor facilidad. Estos metadatos registrados van a permitir interconectar los componentes de pipelines para poder responder preguntas asociadas como: ¿qué versión de Tensorflow creó el modelo X?, ¿Qué dataset fue el encargado de producir el modelo X con resultados Y?, ¿qué pipeline, qué entrenamiento y qué hyper parámetros fue el que generó el modelo X? entre otras preguntas que son de gran utilidad para auditorías, seguimiento de resultados, reproducibilidad, etc.

Una vez los datos han pasado a través de todo el pipeline de data datos, el paso a seguir es la construcción del modelo con los datos previamente preparados. En la metodología tradicional el entrenamiento se realiza en un notebook, en el cual se construye la arquitectura, se entrena el modelo y se evalúa, este proceso carece en su totalidad de

algún tipo de trazabilidad de los experimentos ejecutados en la búsqueda del mejor modelo posible, además la evaluación del modelo tampoco posee alguna validación automática del performance obtenido, por ende es científico de datos el que selecciona los criterios para determinar cuando un modelo es lo suficientemente bueno para detener el proceso de investigación y experimentación.

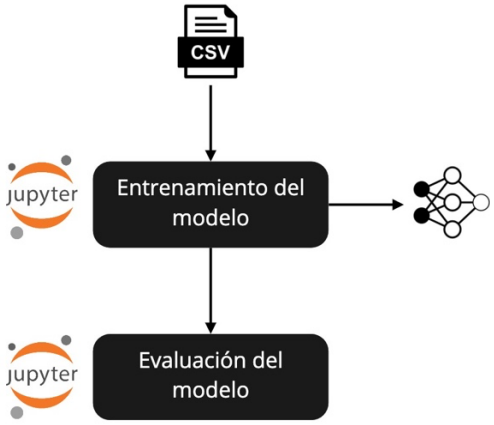

Figura 11. Pipeline de Modelo metodología tradicional

En el desarrollo realizado bajo la integración, se usa el componente *Tuner* para encontrar los mejores hiperparámetros los cuales serán proporcionados al componente *Trainer* posteriormente, ambos componentes consumen los datos del *ExampleGen*, el esquema generado en *SchemaGen*, el grafo de transformaciones construido en el componente *Transform* para entrenar el modelo. Una diferencia importante está en el hecho en el que aún sin evaluar el modelo entrenado, este es almacenado con propósitos de auditoría, además, se almacenan conjuntamente con el grafo de transformación, con el objetivo de evitar el antes mencionado sesgo de transformaciones en entrenamiento y producción. Cada uno de los experimentos o ejecuciones de los componentes *Trainer* y *Tuner* se registran con la herramienta MLFlow, de la cual se hablará más adelante en la sección 4.3.

Subsiguiente al entrenamiento, el próximo componente a ser ejecutado dentro del pipeline es el *Evaluator*, el cual se encarga de hacer un análisis profundo de los resultados de entrenamiento, permitiendo evaluar si el modelo es lo suficientemente bueno respecto al criterio establecido (métricas y thresholds), además también se compara el modelo recientemente entrenado con el último modelo entrenado y aprobado previamente para verificar que se obtuvo una mayor capacidad predictiva respecto a su predecesor. Una vez se han verificado estos dos aspectos, el modelo es marcado como apto, de esta forma el

siguiente componente *Pusher* podrá identificar y enviar aquellos modelos que son aptos para ser llevados a producción.

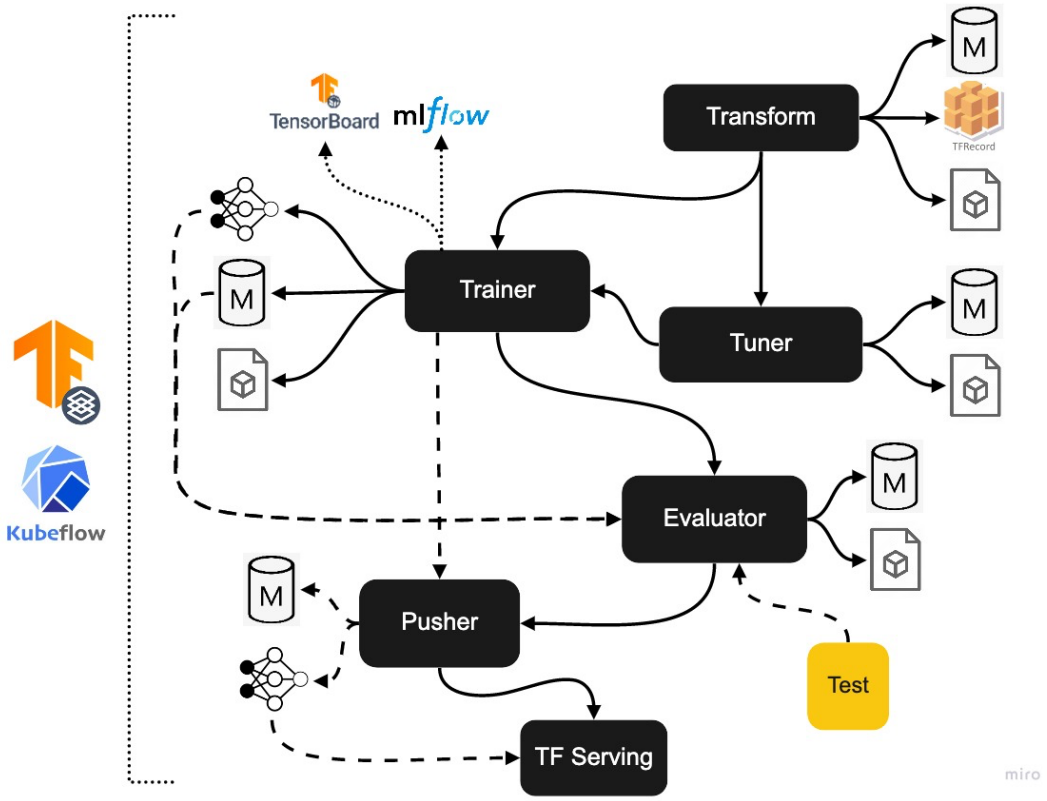

Figura 12. Pipeline de Modelo bajo MLOps y DataOps

4.3 Principios

4.3.1 Versionamiento

En la metodología tradicional, se suelen utilizar las herramientas de control de versiones de código para versionar los 3 niveles de fundamentales. Esto se hace comúnmente con herramientas como git y github, en las cuales se almacenan los códigos, y artefactos creados en la ejecución de los pipelines , como datos y modelos en un mismo repositorio.

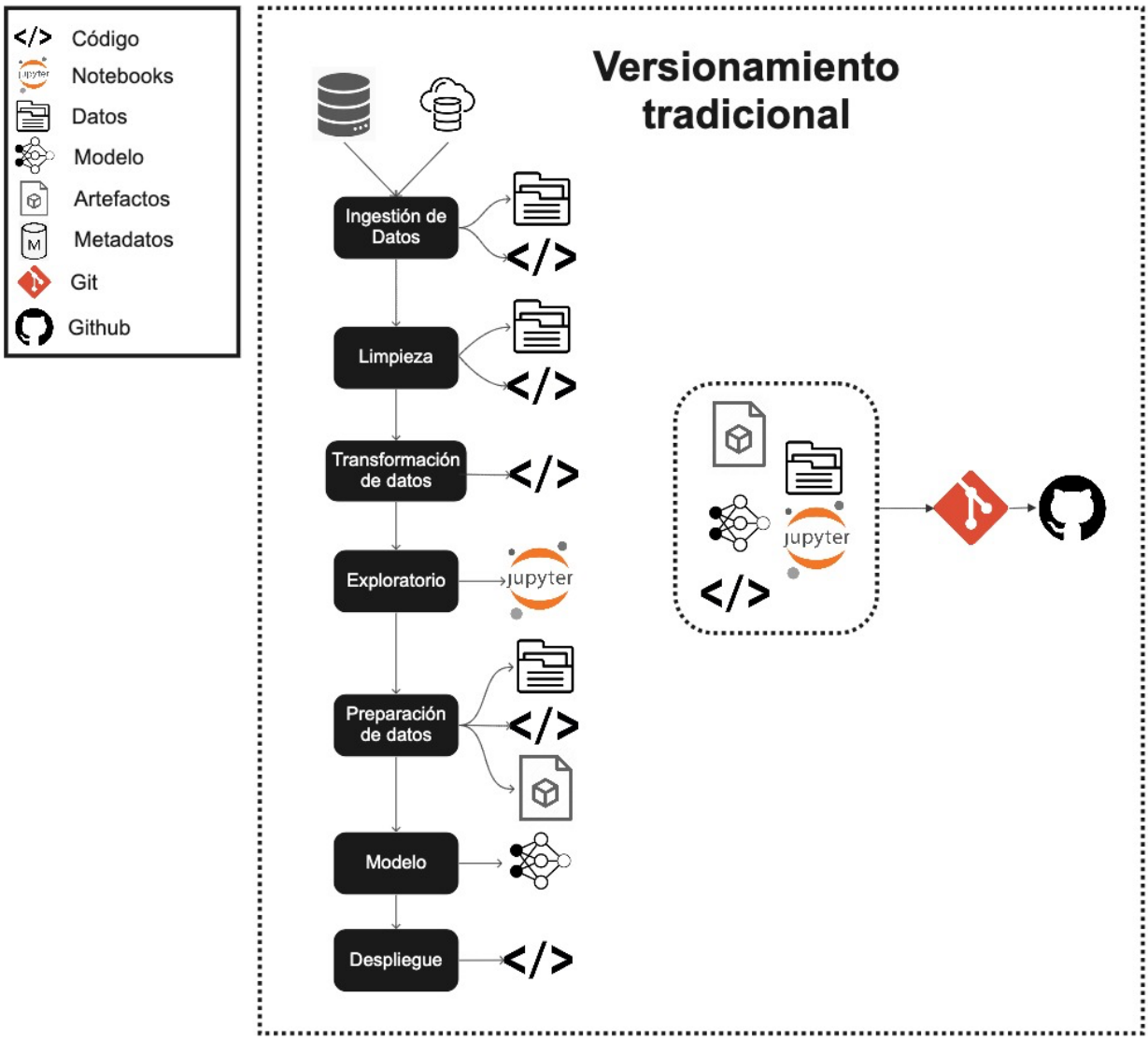

Figura 13. Versionamiento tradicional

En la integración de los enfoques, el versionamiento del eje de datos y modelo se realiza con una herramienta llamada Data Version Control (DVC), la cual permite enlazar el sistema de control de versiones de código con los artefactos generados en el eje de datos y modelo, además de poder manipular grandes archivos, datos, modelos de machine learning, etc. Esta herramienta es descrita en ocasiones como un git para datos, no solo por su propósito sino también porque se usa de forma semejante a git, por lo tanto la curva de aprendizaje del uso de esta herramienta es bastante suave. De esta forma al hacer uso conjunto de DVC con git y github, en el caso de tener que hacer un rollback de código, datos o modelos, se va a poder recrear exactamente los 3 ejes que produjeron los resultados que se tratan de restaurar.

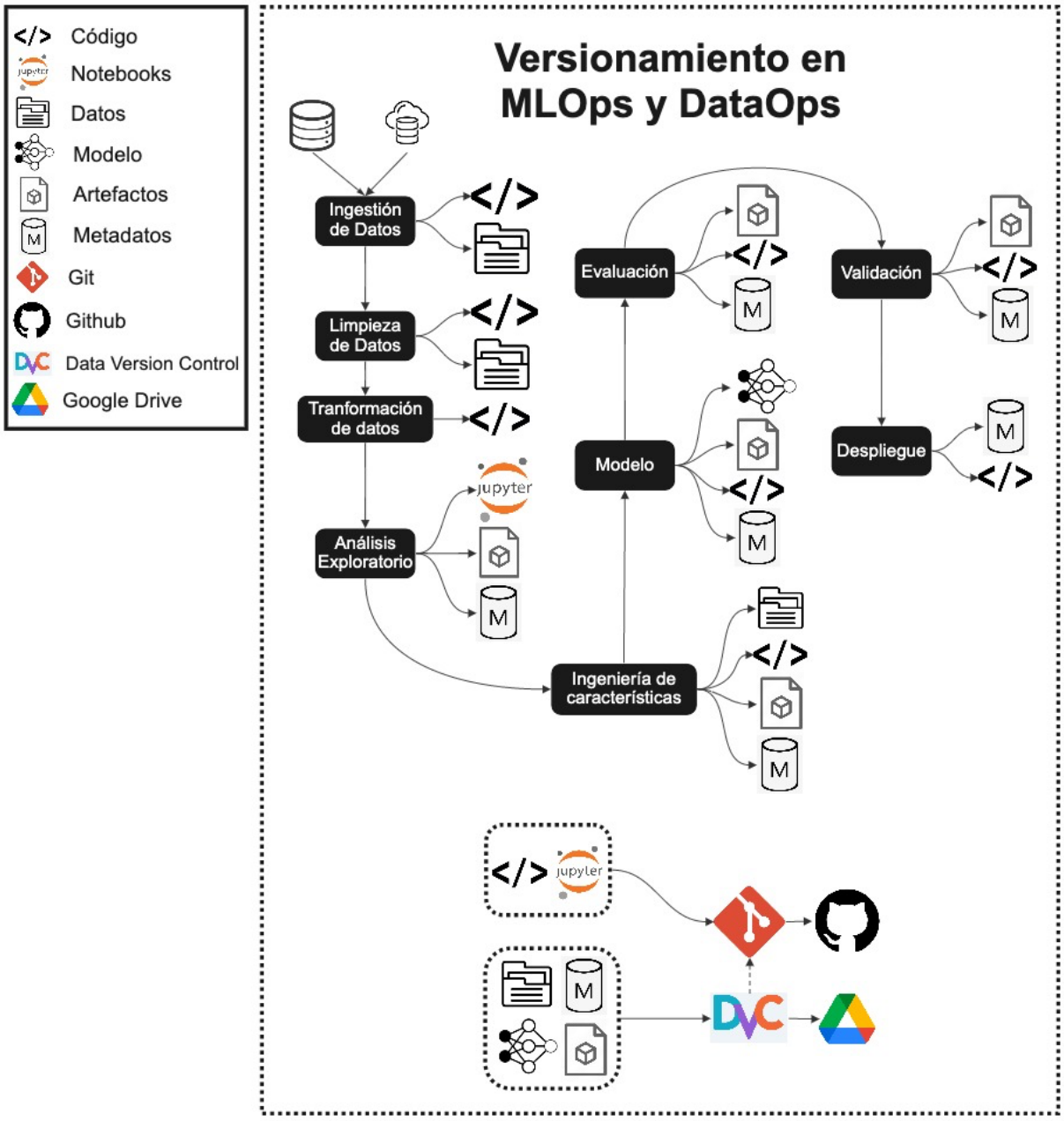

Figura 14. Versionamiento en MLOps y DataOps

En el desarrollo se versionan todos los archivos de datos que son generados para la construcción del dataset, todos los artefactos provenientes de la ejecución del pipeline, como el modelo, esquemas, estadísticas, etc, y por supuesto la base de datos de datos que contiene los metadatos del pipeline. El servicio de almacenamiento seleccionado fue

Google Drive, aunque no es la única opción de almacenamiento externo disponible y esto es una gran fortaleza de esta herramienta, ya que puede ser acoplada sobre muchas plataformas y frameworks existentes. De esta manera se corrige el frecuente error de almacenar los artefactos donde se almacena el código.

4.3.2 Pruebas

En la metodología tradicional rara vez se dedican esfuerzos en el desarrollo de algún tipo de prueba y es bastante común que las verificaciones técnicas y de negocio sean verificadas netamente bajo intervención humana y no de forma automática como debería de ser, por lo menos en gran porcentaje.

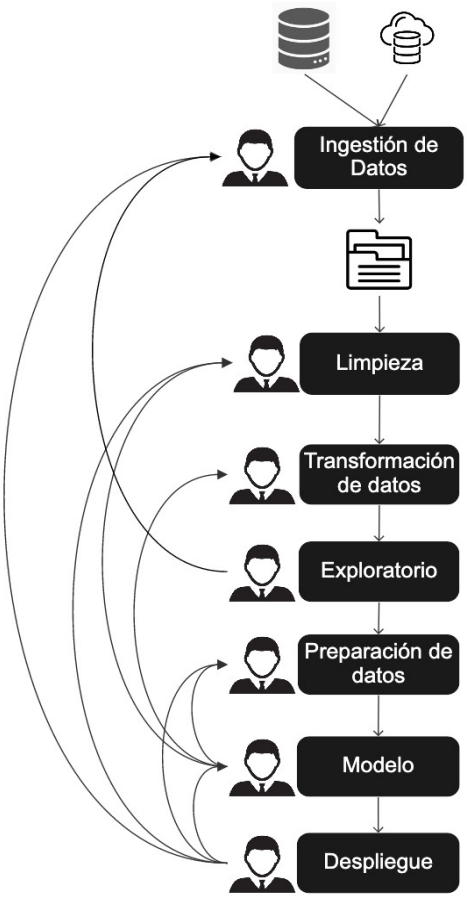

Figura 15. Pruebas en metodologías tradicionales

Así que el flujo común de las aplicaciones desarrolladas en las metodologías tradicionales consiste en que el encargado de cada actividad, desarrolla los procesos necesarios para la ejecución de cada fase y es el absoluto responsable de la identificación de alguna

anomalía, incongruencia, mala calidad de los datos o modelo. Esto, como se ha mencionado, es bastante propenso a errores por diversos factores. Adicional a ello, si se llegase a descubrir algún tipo de falla en el sistema, el proceso de localización de la fuente de dicho error podría ser bastante complicado, pues cómo se ve en la figura 15, un error puede venir de múltiples procesos.

En contraste, en el desarrollo realizado bajo el marco de la integración de ambos enfoques, se implementó un set amplio de pruebas que tratan de cubrir cada una de las fases de todo el pipeline. Las primeras pruebas son construidas sobre el eje de datos, donde las pruebas que se ejecutan en el proceso de adquisición de datos, son las primeras en ser ejecutadas, estas verifican la duplicidad de contenido, tamaño de los archivos y número de filas, pruebas simples pero que alertan sobre la presencia de errores ocurridos en esta fase.

El segundo conjunto de pruebas se encarga de validar que los datos adquiridos tengan las propiedades y características definidas en el esquema de datos; en caso de que la prueba falle, el pipeline fallara y el científico podrá tomar alguna decisión al respecto en si los datos efectivamente son erróneos o se debe modificar el esquema para que admita nueva información con la que antes no se contaba.

Posteriormente se busca cambios en las distribuciones de los datos, o algún tipo de anomalía o diferencias significativas respecto a datos anteriores, de esta forma alarmar al equipo de la presencia de algún tipo de drift sobre los datos. Por último se ejecutan pruebas de expectativas de datos, que son similares a las ejecutadas con el esquema pero con sutiles diferencias y comprobaciones de reglas duras como que cada archivo contenga un número superior de filas respecto a un límite definido para descartar la existencia de archivos truncados, carencia de nulos en el contenido o incluso pruebas que no tienen sentido lógico en el negocio, como que existan registros con exactamente la misma fecha.

En el nivel de modelo se ejecutan pruebas que validen que el performance del modelo es superior a un threshold previamente definido, seguido a ello se comprueba que este performance obtenido es superior al que tiene el modelo que está actualmente en producción, de esta forma tener criterios sólidos para definir si un modelo se lleva o no a producción.

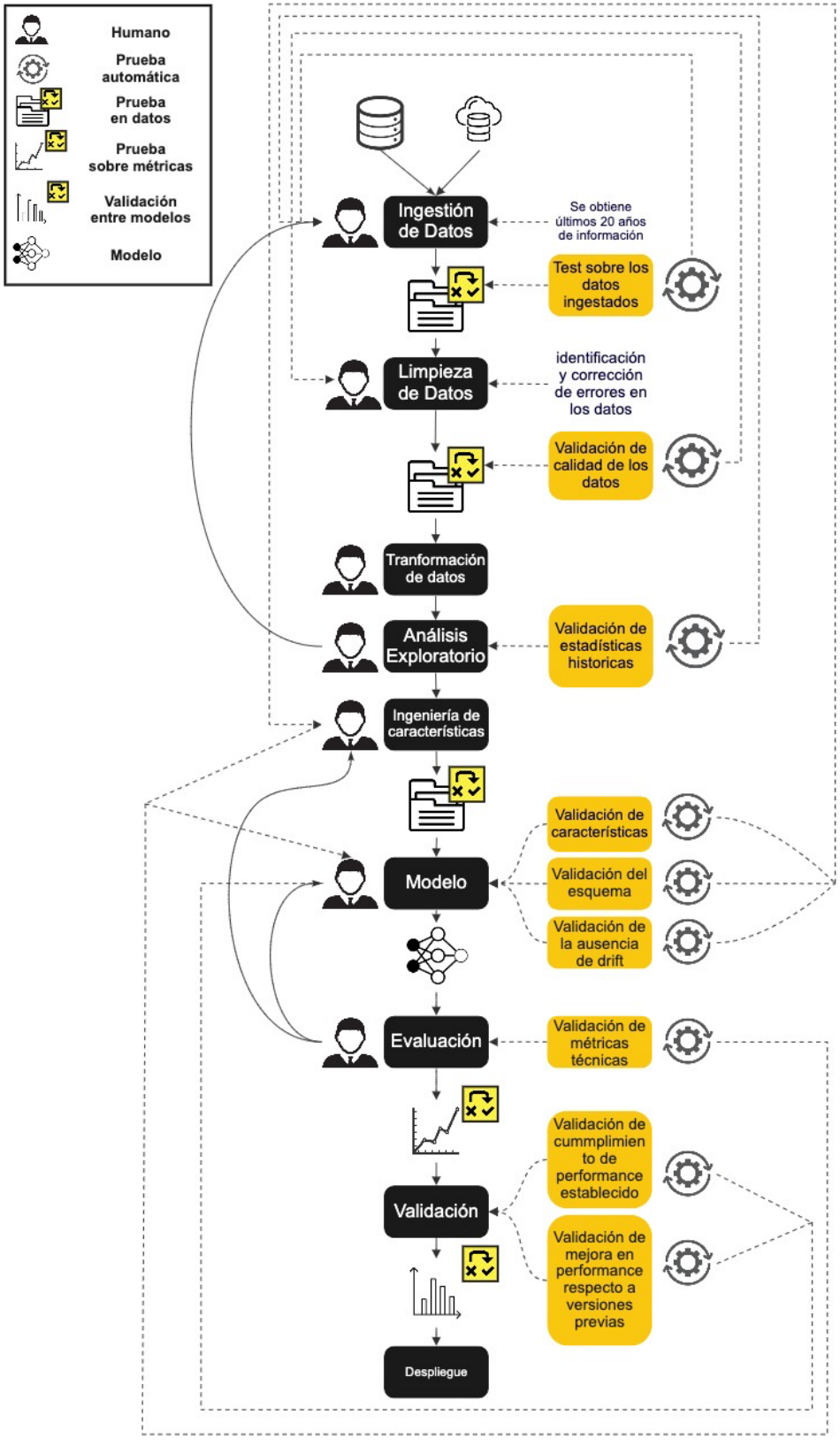

Figura 16. Flujo de Pruebas en el desarrollo y ejecución de la aplicación

Por último se tienen las pruebas del eje de código en las que se tienen pruebas unitarias que comprueban algunas reglas de negocio, como que los parámetros proporcionados a la API en el momento de obtener nuevo datos tengan sentido lógico, también que la variable objetivo no se encuentre entre las feature del modelo para evitar un data leakage, entre otros. Por último se tienen las pruebas de integración, en la que se ejecuta todo el pipeline con un pequeño dataset estático de pruebas, no con el objetivo de medir el performance del modelo, sino su funcionamiento lógico y de que se generen los artefactos pertinentes por cada ejecución del pipeline, por lo tanto al ejecutar la prueba, se verifica que se generan las carpetas que contienen los artefactos de cada componente del pipeline y se hace un conteo para asegurarse de que todos fueron ejecutados, independiente si su performance fue bueno o no. Algunas de las pruebas anteriormente mencionadas están inmersas en los pipeline de CI/CD que serán explicados posteriormente.

4.3.3 Automatización

4.3.3.1 Integración continua

En el desarrollo bajo la metodología tradicional, no existen ningún proceso de integración continua o que simplemente se ocupe de revisar que cada pequeño incremento que hace el equipo no altere o dañe los avances anteriores, esta responsabilidad se asigna en su totalidad al científico, analista, ingeniero o miembro del equipo que ha llevado estos cambios al repositorio, y aunque se trabajase con buenas prácticas de ingeniería de software como gitflow, de nuevo, nada garantiza de que los avances se acoplan de forma adecuada con el proyecto.

En el desarrollo bajo la integración, se construyó un proceso de integración continua con la ayuda de git hooks, estos permiten la ejecución automática de herramientas y scripts en el momento en el que se realiza un commit o un push, y si todo el flujo establecido es ejecutado de forma exitosa, el commit o el push al repositorio serán aceptados, de lo contrario el desarrollador deberá corregir los problemas detectados para intentar de nuevo agregar los cambios. Esto va a permitir poder garantizar en cierta medida que los avances del equipo no dañarían por lo menos el contenido previo a dichos avances. En este flujo de pruebas, se pueden agregar una gran variedad de verificaciones de diferentes propósitos, los implementados en el nuevo enfoque son:

- Ejecución automática de algunas de las pruebas unitarias descritas en el principio de pruebas.
- Verificación de cambios en los datos o artefactos para actualizar el versionamiento de ellos antes de actualizar el código.
- Verificación de que no se suban archivos pesados al repositorio.
- Verificación y corrección de styling, en donde se procesan automáticamente los diferentes tipos de archivos en un formato estándar, ordenar la importación de librerías e incluso borrar las salidas de las celdas de los notebooks para cumplir con buenas prácticas.
- Generación del reporte de cobertura de pruebas sobre cada componente o módulo de la aplicación.

Una vez cada una de dichas pruebas y validaciones han sido ejecutadas de forma exitosa el commit y/o push se pueden realizar de forma exitosa, y allí entra en ejecución el flujo de CD.

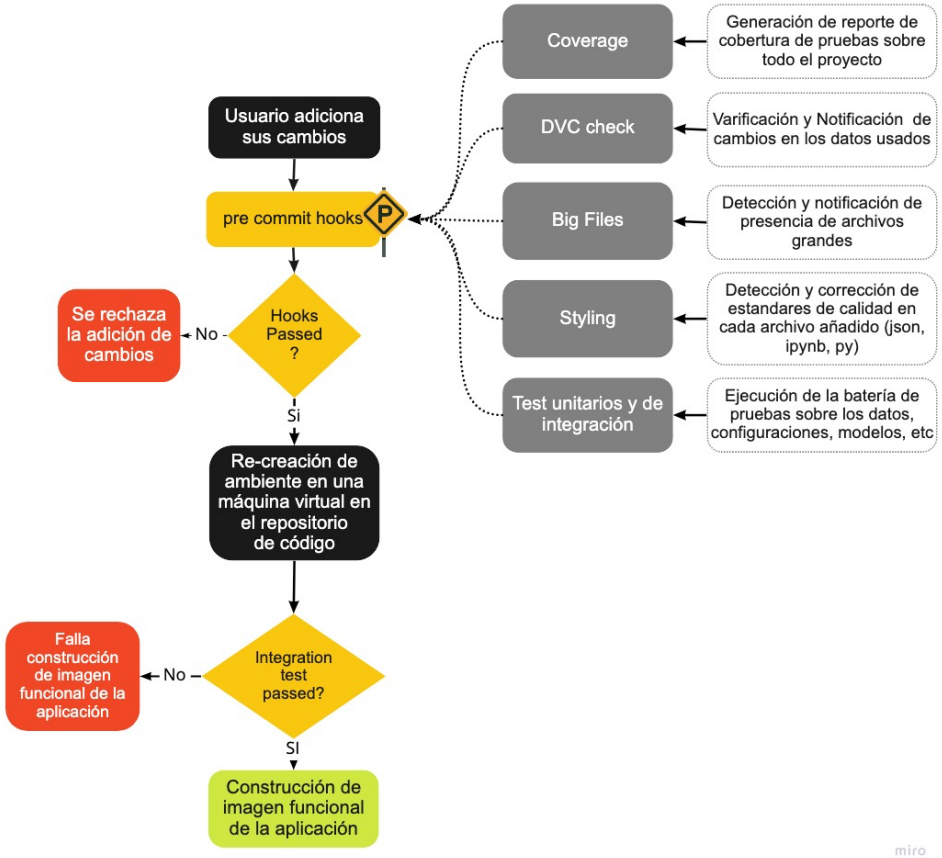

Figura 17. Flujo de pruebas en CI/CD

4.3.3.2 Despliegue Continuo

Asimismo, en la metodología tradicional no se desarrolló ninguna proceso o componente que permita cumplir con el objetivo de despliegue continuo, el cual es que con cada nuevo incremento integrado con éxito al proyecto se pueda generar una versión completamente funcional de este para ser usada en producción automáticamente. Para ello, se utilizó GitHub Actions, en donde se puede automatizar un workflow que pueden compilar, probar y desplegar código sobre el repositorio. Este flujo de trabajo es ejecutado de forma automática en cada push o pull request que se hace a GitHub.

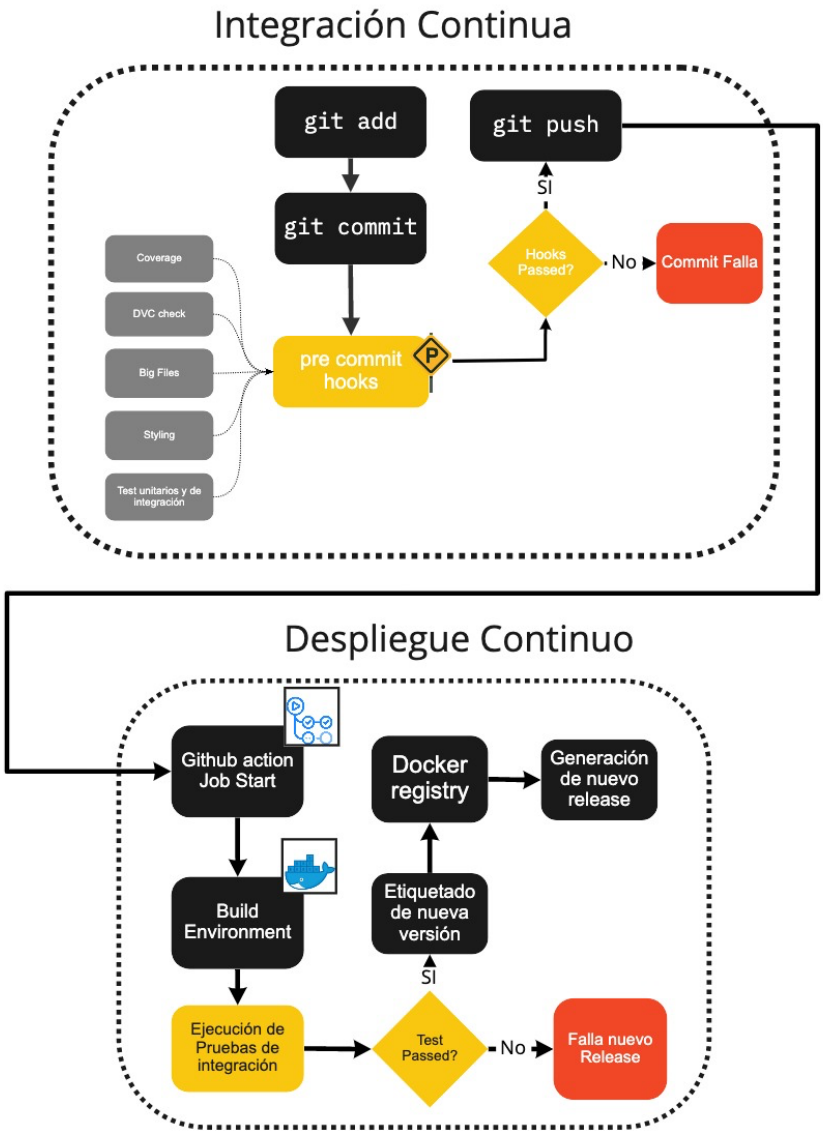

Figura 18. Proceso de CI y CD

El workflow construido consta de varias etapas, en las que inicialmente se construye un ambiente virtual con las mismas características con las que se desarrolló el proyecto, en este caso con una imagen de Ubuntu con la misma versión de python utilizada, y se instalan todas las dependencias del proyecto, luego de esto se ejecutan de nuevo las pruebas de integración de la aplicación y una vez estas pruebas han resultado exitosas, se etiqueta, se crea y se despliega una imagen al registro de imágenes de GitHub llamado GitHub packages, donde se publica una imagen que puede ser accedida y utilizada por cualquier usuario, para hacer uso de la aplicación en cualquier servicio o infraestructura.

4.3.3.3 Orquestación

En la metodología tradicional la ejecución del proyecto está dada de forma manual ejecutando el contenido de cada notebook, esto puede ser una tarea bastante engorrosa y proclive a una gran cantidad de errores, debido a la alta intervención humana.

En la nueva metodología se utilizó un orquestador enfocado a proyecto de machine learning llamado *kubeflow*, el cual permite gestionar pipelines de forma organizada, calendarizarlos, compararlos y examinar detalladamente la salida de cada componente dentro del pipeline, permitiendo auditarlos y depurarlos con suma facilidad en cada ejecución. Además de estas tareas, también se encarga de la ejecución paralela y distribuida de varios componentes que pueden tener una carga computacional elevada y al mismo tiempo auto escalar la infraestructura en caso de ser necesario.

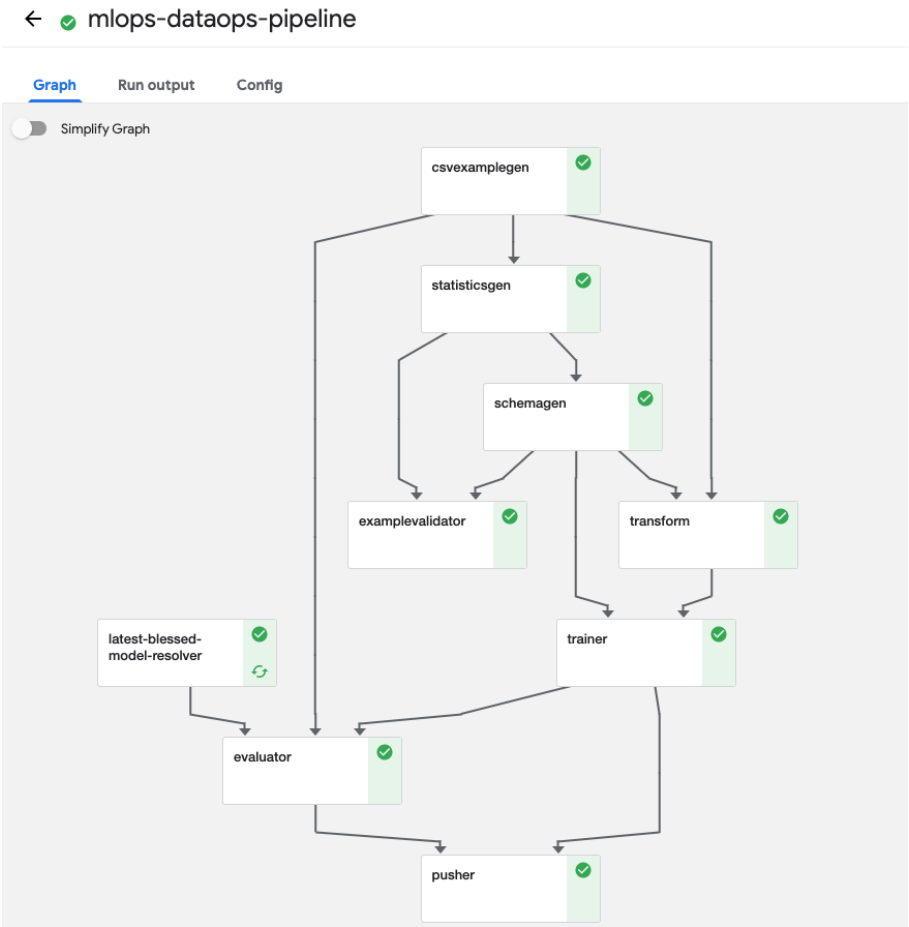

Figura 19. Ejecución del pipeline en Kubeflow

4.3.4 Reproducibilidad

En la metodología tradicional seguir el rastro de algún experimento o investigación realizada es prácticamente imposible, los resultados obtenidos previamente solo quedan almacenados en la memoria del científico encargado de esa experimentación, y si el equipo de desarrollo quisiera tener acceso a los resultados de dichos experimentos, resultaría imposible seguir el rastro de ellos y solo quedan los vagos recuerdos del científico. A esto se le suma la dificultad de ejecutar el proyecto en cualquier infraestructura debido a que no se tiene una estricta definición de las dependencias necesarias para la reproducción de la aplicación, por lo tanto lograr la reproducción de resultados fuera de la máquina utilizada para desarrollar el proyecto podría ser una tarea bastante complicada.

En el desarrollo bajo la integración, el proceso de rastrear los experimentos se realizó de forma automática y centralizada. Para ello se hizo uso de dos herramientas MLFLow y Tensorboard. El componente de MLFlow tracking es una API que proporciona una interfaz de usuario para loggear parámetros, código, versiones, métricas, archivos de salida cuando se ejecutan pipelines de machine learning y que permite posteriores visualizaciones del resultado de todos los experimentos realizados, estos resultados son almacenados en una carpeta en el repositorio, de esta forma, no solo el integrante del equipo encargado de realizar los experimentos tendrá acceso a estos, sino que también el resto de integrantes.

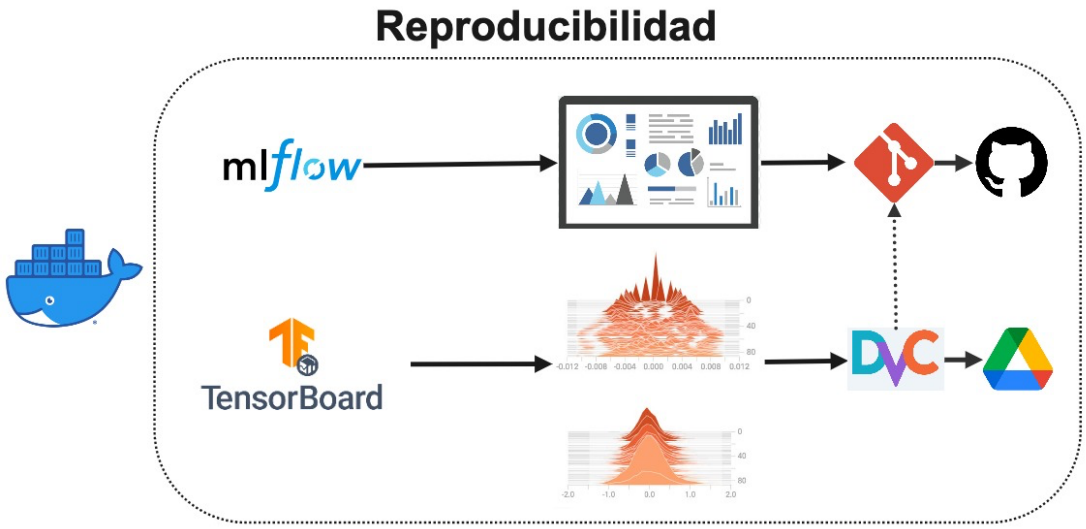

Figura 20. Herramientas y flujo usado para reproducibilidad de la aplicación

El componente de Tensorflow llamada Tensorboard, permite detallar gráficamente cómo ha sido el proceso del entrenamiento para determinar sobre y sub ajustes, anomalías en los entrenamientos y comparar también cada entrenamiento realizado.

Por último, para resolver los problemas de recreación del ambiente de desarrollo en cualquier infraestructura y la instalación de dichas dependencias, se ha utilizado contenedores de Docker para aprovechar la facilidad de empaquetamiento de todas las configuraciones de ambiente y que puedan ser reproducidas de forma fácil y consistente en cualquier entorno.

Repositorio de desarrollo bajo Metodología tradicional

Repositorio de desarrollo bajo Propuesta de integración de MLOps y DataOps

4.3.5 Conclusiones

En este capítulo se evidenció desde la práctica las deficiencias que presenta el desarrollo de aplicaciones bajo las metodologías tradicionales, en este caso una aplicación de trading automático, respecto a una aplicación desarrollada bajo la propuesta de integración de MLOps y DataOps. Además se detallaron algunas de las herramientas y técnicas que permiten ejecutar con mayor facilidad cada uno de los procesos y principios presentes en el nuevo enfoque.

5 Conclusiones

5.1 Respuesta a la pregunta de Investigación

En respuesta a la pregunta de investigación, es posible integrar los enfoques de DataOps y MLOps, los cuales se articulan y complementan de una forma bastante efectiva. De esta forma se puede solventar y reducir gran parte de la deuda técnica generada por el uso de las metodologías tradicionales, y por supuesto, superar las deficiencias que presentan cada uno de los enfoques de forma individual en algunas de las fases presentes en todo el ciclo de vida y construcción de una aplicación de machine learning.

5.2 Cumplimiento de objetivos

5.2.1 Objetivo general

Se elaboró una propuesta de integración metodológica, en la que se establecen inicialmente las diferencias y similitudes de cada enfoque, para luego fijar los pasos y requerimientos necesarios resultantes de la integración de los enfoques. Asimismo, se desarrolló una aplicación de trading automático, bajo el marco de la propuesta de integración de los enfoques de MLOps y DataOps y la metodología tradicional para así comparar las ventajas y desventajas de cada aplicación respectivamente.

5.2.2 Objetivo específicos

Se desarrolló una propuesta de aproximación metodológica entre DataOps y MLOps la cual consta de 3 niveles principales, Datos, Modelo y Despliegue, donde se establecen los pasos necesario para una correcta ejecución y construcción de un proyecto de machine learning. Además se plantearon 4 principios fundamentales los cuales son Versionamiento, Pruebas, automatización y reproducibilidad, las cuales son prácticas necesarias para la mitigación de muchos de los aspectos que generan la deuda técnica en este tipo de proyectos.

También se desarrollaron dos aplicaciones de trading automático, una bajo el marco de las metodologías tradicionales y otra bajo el marco de la integración propuesta. La aplicación toma los datos del activo de predecir, genera indicadores técnicos y transforma cada

registro a imagen para entrenar una red convolucional, la cual tendrá 3 tipos de salidas, “buy”,”sell”,”hold”, las cuales son el tipo de operación a ejecutar en el activo seleccionado. En este desarrollo se realizó una comparativa práctica de cada aplicación evidenciando las múltiples ventajas que ofrece la propuesta de integración sobre el desarrollo realizado bajo las metodologías tradicionales.

6 Bibliografía

[1] kdnuggets, «What main methodology are you using for your analytics, data mining, or data science projects?,» 2014. [En línea]. Available: https://www.kdnuggets.com/polls/2014/analytics-data-mining-data-science-methodology.html.

[2] H. Atwal, Practical DataOps: Delivering Agile Data Science at Scale, Apress, 2020.

[3] M.Treveil y D. Team, Introducing Mlops: How to Scale Machine Learning in the Enterprise, oreilly, 2020.

[4] G. H. D. G. E. D. T. P. D. Sculley, «Hidden Technical Debt in Machine Learning Systems,» *Advances in Neural Information Processing Systems,* 2015.

[5] G. S. d. Nascimento y A. A. D. Oliveira, «An Agile Knowledge Discovery in Databases Software Process,» *ICDKE ,* 2012.

[6] M. Taifi, Clean Machine Learning Code, leanpub, 2020.

[7] What is the Team Data Science Process?, 2020. [En línea]. Available: https://docs.microsoft.com/en-us/azure/architecture/data-science-process/overview.

[8] 2020 state of enterprise machine learning, «algorithmia,» 2020. [En línea]. Available: https://info.algorithmia.com/hubfs/2019/Whitepapers/The-State-of-Enterprise-ML-2020/Algorithmia_2020_State_of_Enterprise_ML.pdf?utm_campaign=The%20Batch&utm_source=hs_email&utm_medium=email&utm_content=80984419&_hsenc=p2ANqtz--sz-e2gfqUeDvVSmjsXfvwOnLHB2Z.

[9] valohai, «valohai.com,» [En línea]. Available: https://valohai.com/mlops/.

[10] Valohai, Practical MLOPS, 2020.

[11] C. Bergh, G. Benghiat y E. Strod, The DataOps Cookbook, vol. 2, 2019.

[12] C. Ebert, G. Gallardo, J. Hernantes y N. Serrano, «DevOps,» *IEEE,* 2016.

[13] «Qué es DataOps y MLOps?,» 2020. [En línea]. Available: https://anllogui.medium.com/qu%C3%A9-es-dataops-y-mlops-e07ef8281416.

[14] A.Burkov, Machine Learning Engineering, True Positive Inc, 2020.

[15] R. Dash y P. Dash, «A Hybrid Stock Trading Framework Integrating Technical Analysis with Machine Learning Techniques,» *ScienceDirect,* 2016.

[16] S. Jansen, Machine Learning for Algorithmic Trading, vol. 2, Packt Publishing, 2020.

[17] B. Huang, Y. Huan, L. Da, L. Zheng y Z. Zou, «Automated trading systems statistical and machine learning methods and hardware implementation: a survey,,» *ResearchGate,* 2018.

[18] M. Dempster y V.Leemans, «An automated FX trading system using adaptive reinforcement learning,,» *ScienceDirect,* 2006.

[19] Figure Eight, «Data Scientist Report 2018,» 2018.

[20] kaggle, «2017 Kaggle Machine Learning & Data Science Survey,» 2017.

[21] M. Loukides, «Machine learning requires a fundamentally different deployment approach,» oreilly, 2019. [En línea].

[22] Google, «MLOps: Continuous delivery and automation pipelines in machine learning,» 2020. [En línea]. Available: https://cloud.google.com/architecture/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning.

[23] P. Sugimura y F. Hartl, «Building a Reproducible Machine Learning Pipeline,» *arxiv,* 2018.

[24] D. L. Visengeriyeva, A. Kammer, I. Bär, A. Kniesz y M. Plöd, «Machine Learning Operations,» 2021. [En línea]. Available: https://ml-ops.org/.

[25] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary y M. Young, «Machine Learning:The High-Interest Credit Card of Technical Debt,» *NIPS,* 2014.

[26] B. Koller, «12 Factors of reproducible Machine Learning in production,» 2021. [En línea]. Available: https://towardsai.net/p/machine-learning/12-factors-of-reproducible-machine-learning-in-production.

[27] D. Talby, «Why Machine Learning Models Crash And Burn In Production,» forbes, 2019. [En línea]. Available: https://www.forbes.com/sites/forbestechcouncil/2019/04/03/why-machine-learning-models-crash-and-burn-in-production/?sh=8e69f632f437.

[28] L. Biewald, «Machine Learning Experiment Tracking,» towardsdatascience, 2020. [En línea]. Available: https://towardsdatascience.com/machine-learning-experiment-tracking-93b796e501b0.

[29] A. Dyck, R. Penners y H. Lichter, «Towards Definitions for Release Engineering and DevOps,» *ResearchGate,* 2015.

[30] R. Jabbari, N. B. Ali, K. Petersen y B. Tanveer, «What is DevOps?: A Systematic Mapping Study on Definitions and Practices,,» *ResearchGate,* 2016.

[31] L. Liebmann, «3 Reasons Why DataOps Is Essential for Big Data Success,» *InformationWeek,* 2017.

[32] altexsoft, «DataOps: Adjusting DevOps for Analytics Product Development,» 2021. [En línea]. Available: https://www.altexsoft.com/blog/dataops-essentials/.

[33] J. Ereth, «DataOps – Towards a Definition,» *ResearchGate,* 2018.

[34] C. Gaur, «Understanding Data Operation (DataOps) Tools and Best Practices,» 2018. [En línea]. Available: https://www.xenonstack.com/insights/data-operations.

[35] dataops.rocks, «What is DataOps?,» 2021. [En línea]. Available: https://www.dataops.rocks/.

[36] C. Pinkel, A. Schwarte, J. Trame y A. Nikolov, «DataOps: Seamless End-to-End Anything-to-RDF Data Integration,» *ResearchGate,* 2015.

[37] unraveldata, «Why dataops is critical for your business,» 2021. [En línea]. Available: https://www.unraveldata.com/resources/why-dataops-is-critical/.

[38] G. Fursin, «The Collective Knowledge project: making ML models more portable and reproducible with open APIs, reusable best practices and MLOps,» *arxiv,* 2020.

[39] D. Sweenor, S. Hillion, D. Rope, D. Kannabiran, T. Hill y M. O'Connell, ML Ops Operationalizing Data Science, O'Reilly, 2020.

[40] N. &. K. G. &. K. N. &. H. J. Forsgren, «2014 State of DevOps Report,» 2014.

[41] G. Maayan, «¿DevSecOps vs DataOps vs MLOps?,» 2020. [En línea]. Available: https://towardsdatascience.com/devsecops-vs-dataops-vs-mlops-93b49f0282b8.

[42] Fabio Buso; Jim Dowling, «MLOps with a Feature Store,» 2020. [En línea]. Available: https://www.logicalclocks.com/blog/mlops-with-a-feature-store.

[43] H. Hapke y C. Nelsono, Building Machine Learning Pipelines: Automating Model Life Cycles with TensorFlow, Oreilly, 2020.

[44] J. Saltz, I. Shamshurin y K. Crowston, «Comparing Data Science Project Management Methodologies via a Controlled Experiment,» de *Hawaii International Conference on System Sciences*, 2017.

[45] C. Windheuser, D. Sato y A. Wider, «Continuous Delivery for Machine Learning,» 2019. [En línea]. Available: https://martinfowler.com/articles/cd4ml.html.

[46] J. Riley, Understanding metadata, what is metadata, and what is it for?, niso, 2017.

[47] M. Martynov y D. Mezhensky, «5 Steps to Implementing a Successful DataOps Practice,» 2020. [En línea]. Available: https://blog.griddynamics.com/5-technology-enablers-for-dataops/.

[48] T. Nagle y T. C. Redman, «Only 3% of Companies' Data Meets Basic Quality Standards,» Harvard Business Review, 2017. [En línea]. Available: https://hbr.org/2017/09/only-3-of-companies-data-meets-basic-quality-standards.

[49] I. F. Ilyas y X. Chu, Data Cleaning, acm, 2019.

[50] Aggarwal y C. C., Outlier Analysis, springer, 2017.

[51] ISO 25000, «ISO/IEC 25012,» [En línea]. Available: https://iso25000.com/index.php/en/iso-25000-standards/iso-25012.

[52] M. Kuhn y K. Johnson, Feature Engineering and Selection: A Practical Approach for Predictive Models, CRC, 2019.

[53] Feast, «What is Feast?,» [En línea]. Available: https://docs.feast.dev/.

[54] Zipline, «Zipline—Airbnb's Declarative Feature Engineering Framework,» 2019. [En línea]. Available: https://databricks.com/session_eu19/zipline-airbnbs-declarative-feature-engineering-framework.

[55] aws, «Training ML Models,» [En línea]. Available: https://docs.aws.amazon.com/es_es/machine-learning/latest/dg/training-ml-models.html.

[56] Ram Vittal; Neelam Koshiya; Raghu Ramesh, «Deploy shadow ML models in Amazon SageMaker,» 2021. [En línea]. Available:

https://aws.amazon.com/es/blogs/machine-learning/deploy-shadow-ml-models-in-amazon-sagemaker/.

[57] christophergs, «Deploying Machine Learning Models in Shadow Mode,» 2020. [En línea]. Available: https://christophergs.com/machine%20learning/2019/03/30/deploying-machine-learning-applications-in-shadow-mode/.

[58] k. georgieva, «Monitoring ML pipelines,» 2020. [En línea]. Available: https://intothedepthsofdataengineering.wordpress.com/2020/02/13/monitoring-ml-pipelines/.

[59] A. Gonfalonieri, «Why Machine Learning Models Degrade In Production,» 2019. [En línea]. Available: https://towardsdatascience.com/why-machine-learning-models-degrade-in-production-d0f2108e9214.

[60] E. Breck, S. Cai, E. Nielsen, M. Salib y D. Sculley, «The ML Test Score: A Rubric for ML Production Readiness and Technical Debt Reduction,» *IEEE,* 2017.

[61] sentryone, «DataOps Simplify Managing Complex Data Environments,» [En línea]. Available: https://www.sentryone.com/dataops-overview#integrate.

[62] M. Wacker, «Just Say No to More End-to-End Tests,» 2015. [En línea]. Available: https://testing.googleblog.com/2015/04/just-say-no-to-more-end-to-end-tests.html.

[63] J. Le, «The 5 Components Towards Building Production-Ready Machine Learning Systems,» 2020. [En línea]. Available: https://data-notes.co/the-5-components-towards-building-production-ready-machine-learning-system-a4d5237ec04e.

[64] F. A. Hubis, W. Wu y C. Zhang, «Quantitative Overfitting Management for Human-in-the-loop ML Application Development with ease.ml/meter,» *arxiv,* 2020.

[65] M. Fowler, «ThresholdTest,» 2013. [En línea]. Available: https://martinfowler.com/bliki/ThresholdTest.html.

[66] S. Studer, T. B. Bui, C. Drescher, A. Hanuschkin, L. W. 2, S. Peters y K.-R. Müller, «Towards CRISP-ML(Q): A Machine Learning Process Model with Quality Assurance Methodology,» *arxiv,* 2021.

[67] T. Onyszko, «Is your data really helping your business?,» 2020. [En línea]. Available: https://www.predicagroup.com/blog/data-operations-guide/#stages.

[68] D. Farley y J. Humble, Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation, 2010.

[69] M. Schmitt, «Airflow vs. Luigi vs. Argo vs. MLFlow vs. KubeFlow,» 2020. [En línea]. Available: https://towardsdatascience.com/airflow-vs-luigi-vs-argo-vs-mlflow-vs-kubeflow-b3785dd1ed0c.

[70] H. K. Trevor Grant, B. Lublinsky, R. Liu y I. Filonenko, Kubeflow for Machine Learning, oreilly, 2020.

[71] J. Chen, «Technical Indicator,» 2021. [En línea]. Available: https://www.investopedia.com/terms/t/technicalindicator.asp.

[72] O. B. Sezer y M. Ozbayoglu, «Algorithmic Financial Trading with Deep Convolutional Neural Networks: Time Series to Image Conversion Approach,» *Applied Soft Computing,* 2018.

[73] Tensorflow, «Tensorflow extended,» 2021. [En línea]. Available: https://www.tensorflow.org/tfx.