Universidades de Burgos, León y Valladolid

Máster Universitario

Inteligencia de Negocio y Big Data en Entornos Seguros

TFM del Máster Inteligencia de Negocio y Big Data en Entornos Seguros

Metodologías de Control de Calidad y Pruebas para Soluciones Tecnológicas basadas en AI/ML y Big Data

Presentado por Francisco Díaz Gil
en la Universidad de Valladolid
15 de Julio de 2022

Tutores: Carlos Enrique Vivaracho Pascual
José Javier Yebes Torres

Resumen

Los procesos de pruebas y control de calidad son parte principal de cualquier proyecto de desarrollo de software. En general, se trata de asegurar el éxito del producto final mediante un ciclo que incluya diseño, implementación, prueba y despliegue/integración, en ese orden. Los ingenieros de control de calidad trabajan durante todo el ciclo de vida del desarrollo de software utilizando metodologías ágiles y probando todo el progreso en incrementos pequeños e iterativos.

Las soluciones tecnológicas basadas en Inteligencia Artificial y Aprendizaje Automático soportan de forma natural los procesos iterativos de aprendizaje continuo. Requieren asegurar la calidad del dato de entrada, evaluar la salida esperada de los modelos entrenados usando unas métricas definidas y explorar las configuraciones paramétricas de dichos modelos. En muchas ocasiones, la solución final requiere combinar distintos modelos, algoritmos, heurísticas y etapas de pre-/post-procesado que elevan la complejidad de los procesos de pruebas y el control de calidad. Estos procesos deben ser incorporados no solo en la etapa previa a la puesta en producción sino también como parte del mantenimiento continuo cada vez que se producen cambios en alguna parte del sistema.

El presente Trabajo Final de Máster aborda la revisión de los procesos y herramientas de Aseguramiento de la Calidad típicos del desarrollo software y su posible aplicación al área de Aprendizaje Automático. También se realiza un análisis de las nuevas metodologías *Data Centric* y *MLOps* para el control de calidad de soluciones tecnológicas complejas basadas en Inteligencia Artificial. Por último, se plantea la aplicación de dichas estrategias y conceptos de Aseguramiento de la Calidad a un caso de uso práctico en el área del procesamiento automático de documentos.

Descriptores

Inteligencia artificial, aprendizaje automático, pruebas, aseguramiento de la calidad, automatización, centrado en datos, operaciones de aprendizaje automático, procesamiento inteligente de documentos.

Abstract

Testing and quality control processes are a main part of any software development project. In general, it is about ensuring the success of the final product through a cycle that includes design, implementation, testing, and deployment/integration, in that order. QA engineers work throughout the software development lifecycle using agile methodologies and testing all progress in small, iterative increments.

Technological solutions based on Artificial Intelligence and Machine Learning naturally support iterative continuous learning processes. They require ensuring the quality of the input data, evaluating the expected output of the trained models using defined metrics, and exploring the parametric configurations of those models. On many occasions, the final solution requires combining different models, algorithms, heuristics and pre-/post-processing stages that increase the complexity of the testing and quality control processes. These processes must be incorporated not only in the stage prior to putting into production but also as part of ongoing maintenance whenever changes occur in any part of the system.

This Master's Final Project addresses the review of the processes and tools of Quality Assurance typical of software development and its possible application to the area of Machine Learning. An analysis of the new *Data Centric* and *MLOps* methodologies for the quality control of complex technological solutions based on Artificial Intelligence is also carried out. Finally, the application of these strategies and concepts of Quality Assurance to a practical use case in the area of automatic document processing is proposed.

Keywords

Artificial intelligence (AI), machine learning (ML), testing, quality assurance (QA), automation, *data centric*, machine learning operations (*MLOps*), intelligent document processing (IDP).

Índice General

Índice de Figuras

Índice de Tablas

Parte I

Introducción

Capítulo 1

Introducción

Las *pruebas de software* son una parte esencial de cualquier software o proyecto de desarrollo y son cada vez más críticas para la validación y la confianza en el mercado actual. Las pruebas de software siempre han existido, aunque no tal y como se conocen ahora, puesto que han sufrido una evolución en el tiempo. A continuación se indican brevemente las diferentes etapas [1] de las pruebas de software a lo largo de la historia:

- **Depuración**. Esta fase sucedió a principios de la década de 1950, cuando no había distinción entre pruebas y depuración y la atención se centró en la corrección de errores[1]. No existía el concepto de pruebas o probadores.
- **Demostración**. Esta fase transcurrió desde 1957 a 1978 y ya se hizo la distinción entre depuración y prueba. Además, la prueba se llevaba a cabo como una actividad separada. El principal objetivo de las pruebas de software era asegurar que se cumplieran los requisitos del software.
- **Destrucción**. Esta fase tuvo lugar entre 1979 y 1982 y la atención se centró en descifrar el código y encontrar los errores en él. El enfoque orientado a la destrucción también fracasó porque el software nunca se lanzaría dado que al corregir un error también se podría generar otro error. No hubo un enfoque de prevención de defectos[2] durante esta fase.
- **Evaluación**. Esta fase transcurrió desde 1983 a 1987 y la atención se centró en evaluar y medir la calidad del software. Las pruebas mejoraron el índice de confianza sobre el funcionamiento del software. Esto era principalmente aplicable a software de gran tamaño.

[1] Acción humana que produce un resultado incorrecto.
[2] Presencia de un error en el software.

- **Prevención**. Esta fase tuvo lugar entre 1988 y 2000 y supuso un nuevo punto de vista. Las pruebas estaban enfocadas en demostrar que el software cumplía con su especificación, detectando fallos[3] y previniendo defectos. En esta etapa también surgieron las pruebas exploratorias, cuyo objetivo era probar y explorar el software en un intento de encontrar más errores.

A partir de ese momento todo evolucionó de una forma mucho más rápida. Desde el *modelo de desarrollo en cascada*, en el cual las pruebas comenzaban después del desarrollo y prácticamente sin espacio para realizar cambios en los requisitos, hasta los modelos *agile*, en los que las pruebas son parte del ciclo de vida del desarrollo desde el inicio para garantizar que la calidad se incorpore desde ese instante.

Posteriormente llegaron los entornos *DevOps*[4], que consisten en un desarrollo cíclico de comunicación constante y mejoras iterativas. En este caso, la automatización es fundamental para ejecutar casos de prueba al crear e implementar código nuevo. Y también la canalización *CI/CD*[5] que permite entregar cambios de código con mayor frecuencia y confiabilidad y que requiere la aplicación de un nuevo elemento, las pruebas continuas.

También surgen nuevos conceptos relacionados con las pruebas como el desarrollo basado en el comportamiento (*Behavior Driven Development*, BDD, en inglés) y el desarrollo basado en pruebas (*Test-Driven Development*, TDD, en inglés). Las herramientas de pruebas de automatización o las pruebas de *API*[6] marcaron otro punto de inflexión en la historia de las pruebas. Finalmente, la era actual se está moviendo hacia las pruebas con herramientas de Inteligencia Artificial.

¿En qué punto se encuentran ahora las pruebas de software? El Informe de Calidad Mundial 2021-22 [2] nos indica las claves para entender la situación actual:

1. **Centrarse en lo que importa**. El foco debe ser la experiencia del cliente y satisfacer sus necesidades con eficacia y rapidez.

[3] Manifestación física o funcional de un defecto.
[4] Development Operations, https://www.redhat.com/es/topics/devops
[5] Continuous Integration and Continuous Delivery/Deployment, https://www.redhat.com/es/topics/devops/what-is-ci-cd
[6] Application Programming Interface.

2. **Estandarizar el uso de la automatización de pruebas y utilizarla de principio a fin**. La prioridad debe ser la automatización para aumentar la calidad del software y utilizarla en todas las fases del proceso.

3. **AI[7]/ML[8], de la teoría a la práctica**. La Inteligencia Artificial y el Aprendizaje Automático deben transformar la ingeniería de calidad.

4. **Insistir en la disponibilidad de entornos y datos de prueba**. Es necesario invertir en soluciones de disponibilidad de entornos y datos en tiempo real como parte de la estrategia organizativa.

5. **Conseguir que la dirección apoye las iniciativas de la industria inteligente**. Es fundamental contar con el apoyo del equipo directivo demostrando los cambios con resultados. Este apoyo, sin duda, facilitará la evolución.

1.1. Motivación

El *aseguramiento de la calidad* de software (ver Sección 2.3), que inicialmente se trataba de una actividad independiente, ha evolucionado hasta llegar a convertirse en una función totalmente integrada en las organizaciones. De hecho, ha pasado a ser una parte integral de la transformación digital de las empresas, dado que contribuye al crecimiento empresarial y a los resultados de negocio.

Con el desarrollo del Big Data y los sistemas de AI/ML surge la necesidad de aplicar el control de calidad a los datos y al ciclo de vida y configuración de los modelos y soluciones AI/ML, como ya se hacía en el software de los sistemas de desarrollo tradicional. Por lo tanto, es necesario revisar cómo se pueden trasladar los conceptos básicos de calidad en la ingeniería del software a las soluciones AI/ML. Además, se desarrollan nuevas tendencias al amparo de dichos sistemas, como *Data Centric* y *MLOps*, que sin duda ayudarán a fortalecer el compromiso de las soluciones AI/ML con la calidad a todos los niveles.

La calidad es un concepto que está presente en todos los ámbitos pero el desarrollo de las áreas de Inteligencia Artificial y Aprendizaje Automático pueden ayudar a la consolidación definitiva de esta tendencia.

[7] Artificial Intelligence.
[8] Machine Learning.

1.2. Objetivos

En la actualidad, los modelos de desarrollo de software se basan en metodologías ágiles. Esto implica que el aseguramiento de la calidad se lleve a cabo durante todo el ciclo de vida del desarrollo dando como resultado pequeños incrementos iterativos. A su vez, las soluciones basadas en sistemas AI/ML soportan de forma natural estos procesos iterativos. Partiendo de esta base, este Trabajo Final de Máster pretende aplicar las nociones básicas de los procesos de aseguramiento de la calidad al área de los sistemas AI/ML para después analizar algunas metodologías concretas que sean útiles para aplicar el control de calidad a las soluciones AI/ML. Por último se analizará la aplicación de estas metodologías en un caso de uso concreto, el *Procesamiento Inteligente de Documentos*. Partiendo de esta línea de trabajo se identifican los siguientes objetivos:

- OBJ-1: Definición de los conceptos básicos del aseguramiento de la calidad del desarrollo software tradicional.
  - OBJ-1.1: Establecer las bases de los elementos principales del aseguramiento de la calidad.
  - OBJ-1.2: Analizar las tendencias y el futuro del aseguramiento de la calidad.
- OBJ-2: Aplicar los conceptos del aseguramiento de la calidad a los sistemas AI/ML.
  - OBJ-2.1: Explicar los conceptos AI/ML y sus diferencias.
  - OBJ-2.2: Analizar las diferencias de estos sistemas respecto a los sistemas de software tradicional.
  - OBJ-2.3: Plantear los desafíos y aspectos críticos que implican estos sistemas.
  - OBJ-2.4: Estudiar los modelos de pruebas de estos sistemas así como sus herramientas.
- OBJ-3: Explorar nuevas tendencias de aseguramiento de la calidad en sistemas AI/ML.
  - OBJ-3.1: Analizar el concepto *Data Centric*.
  - OBJ-3.2: Analizar el concepto *MLOps*.

•OBJ-4: Aplicar las nuevas tendencias y conceptos de QA al *Procesamiento Inteligente de Documentos* (IDP).

- OBJ-4.1: Presentar un sistema típico IDP.
- OBJ-4.2: Establecer la relación entre las etapas de un proceso IDP y las nuevas tendencias QA en sistemas AI/ML

Con la consecución de los objetivos descritos anteriormente, se pretende aplicar a un caso de uso concreto, el *Procesamiento Inteligente de Documentos*, las nuevas tendencias y conceptos de QA en sistemas AI/ML (OBJ-4). Para ello se describen inicialmente los conceptos básicos del Aseguramiento de la Calidad (OBJ-1). Posteriormente se explica la forma de aplicar esos conceptos en los sistemas AI/ML (OBJ-2). Y por último, se analizan en detalle dos nuevas tendencias del Aseguramiento de la Calidad en sistemas AI/ML (OBJ-3), *Data Centric* y *MLOps*, cuya aplicación se explicará en las etapas del *Procesamiento Inteligente de Documentos*.

1.3. Organización de la Memoria

El presente Trabajo Final de Máster se ha dividido y estructurado en diferentes capítulos en los que se abordan distintos aspectos de interés del proyecto. Los capítulos que forman el documento, así como la temática de estos son los siguientes:

Capítulo 1. Introducción: En este capítulo se llevará a cabo una introducción del contexto en el que se engloba este proyecto, así como la motivación de este y los objetivos que se pretenden lograr tras su finalización. También incluye la presente sección relativa a la organización de la memoria.

Capítulo 2. Procesos y Herramientas Quality Assurance (QA): En este capítulo se revisan conceptos básicos relacionados con las pruebas y el aseguramiento de la calidad en el desarrollo de los proyectos de software tradicional.

Capítulo 3. Aplicación de Procesos QA al área AI/ML: En este capítulo se explican los términos Inteligencia Artificial y Aprendizaje Automático así como la aplicación de la metodología de aseguramiento de la calidad de los proyectos de software tradicional en este nuevo área. Se analizarán los desafíos y los aspectos más críticos de esta implementación.

Capítulo 4. Nuevas Tendencias de Procesos QA en AI/ML: En este capítulo se analizan las nuevas tendencias de aseguramiento de la calidad en los sistemas de Inteligencia Artificial y Aprendizaje Automático. Fundamentalmente se explicará la evolución desde *Model Centric* hasta las dos tendencias actuales, *Data Centric* y *MLOps*.

Capítulo 5. Aplicación de Estrategias y Procesos QA en IDP: En este capítulo se establece la relación entre las nuevas tendencias de aseguramiento de la calidad en los sistemas de Inteligencia Artificial y Aprendizaje Automático, *Data Centric* y *MLOps*, y las etapas de un caso de uso práctico, el *Procesamiento Inteligente de Documentos*.

Capítulo 6. Conclusiones Finales: En este capítulo se presentarán las conclusiones obtenidas a lo largo del desarrollo del proyecto.

1.4. Planificación

Dado que este trabajo se realiza bajo la asignatura de Trabajo Final de Máster, su alcance y duración deben ajustarse al número de horas fijado para dicha asignatura (9 créditos ECTS), lo que se traduce en una carga de 225 horas.

El proyecto se ha dividido en cuatro bloques, comenzando el trabajo el día 1 de febrero de 2022, y concluyendo el último bloque el día 30 de junio del mismo año. Para facilitar la estructuración y organización de las tareas se establecieron puntos de control cada dos semanas que sirvieron para comprobar si se estaban cumpliendo los objetivos fijados en la planificación.

Los bloques que se han definido para satisfacer los objetivos del proyecto son los siguientes:

- **B-01: Definición de los conceptos básicos del aseguramiento de la calidad del desarrollo software tradicional (*OBJ-1*).**

  Este bloque tiene como objetivo establecer los conceptos básicos relacionados con el aseguramiento de la calidad así como el estudio de las posibles tendencias de futuro.

- **B-02: Aplicar los conceptos del aseguramiento de la calidad a los sistemas AI/ML (*OBJ-2*).**

  Este bloque tiene como objetivo entender los sistemas AI/ML para plantear de forma efectiva la aplicación de los conceptos relacionados con el aseguramiento de la calidad vistos en los sistemas de software tradicional.

- **B-03: Explorar nuevas tendencias de aseguramiento de la calidad en sistemas AI/ML (*OBJ-3*).**

  Este bloque tiene como objetivo analizar las nuevas tendencias de aseguramiento de la calidad en sistemas AI/ML, *Data Centric* y *MLOps*.

- **B-04: Aplicar las nuevas tendencias y conceptos de QA al Procesamiento Inteligente de Documentos (IDP) (*OBJ-4*).**

  Este bloque tiene como objetivo entender el mecanismo de un proceso IDP e intentar aplicar las tendencias *Data Centric* y *MLOps* en las etapas de dicho sistema.

A continuación se presenta el desglose de las tareas que conforman cada bloque del proyecto (ver Tablas 1.1 - 1.4):

Tarea	Descripción
T1.1	Documentar la introducción del proyecto
T1.2	Establecer las bases de los elementos principales del aseguramiento de la calidad
T1.3	Analizar las tendencias y el futuro del aseguramiento de la calidad

Tabla 1.1: Desglose de las tareas asociadas al Bloque 1

Tarea	Descripción
T2.1	Explicar los conceptos AI/ML y sus diferencias
T2.2	Analizar las diferencias de estos sistemas respecto a los sistemas de software tradicional
T2.3	Plantear los desafíos y aspectos críticos que implican estos sistemas
T2.4	Estudiar los modelos de pruebas de estos sistemas así como sus herramientas

Tabla 1.2: Desglose de las tareas asociadas al Bloque 2

Tarea	Descripción
T3.1	Analizar el concepto *Data Centric*
T3.2	Analizar el concepto *MLOps*

Tabla 1.3: Desglose de las tareas asociadas al Bloque 3

Tarea	Descripción
T4.1	Presentar un sistema típico IDP
T4.2	Establecer la relación entre las etapas de un proceso IDP y las nuevas tendencias QA en sistemas AI/ML
T4.3	Documentar las conclusiones del proyecto

Tabla 1.4: Desglose de las tareas asociadas al Bloque 4

En la tabla 1.5 se muestra la distribución de las tareas entre los distintos bloques así como su planificación y su carga efectiva en horas de trabajo. Mientras que en la Figura 1.1 se presenta el diagrama de Gantt correspondiente a la planificación establecida para el proyecto.

Bloque	Inicio	Fin	Tareas	Horas
B-01	01/02/2022	01/03/2022	T1.1; T1.2; T1.3	35 horas
B-02	21/02/2022	31/03/2022	T2.1; T2.2; T2.3; T2.4	55 horas
B-03	01/04/2022	01/06/2022	T3.1; T3.2;	80 horas
B-04	23/05/2022	30/06/2022	T4.1; T4.2; T4.3	55 horas

Tabla 1.5: Distribución de tareas por bloques

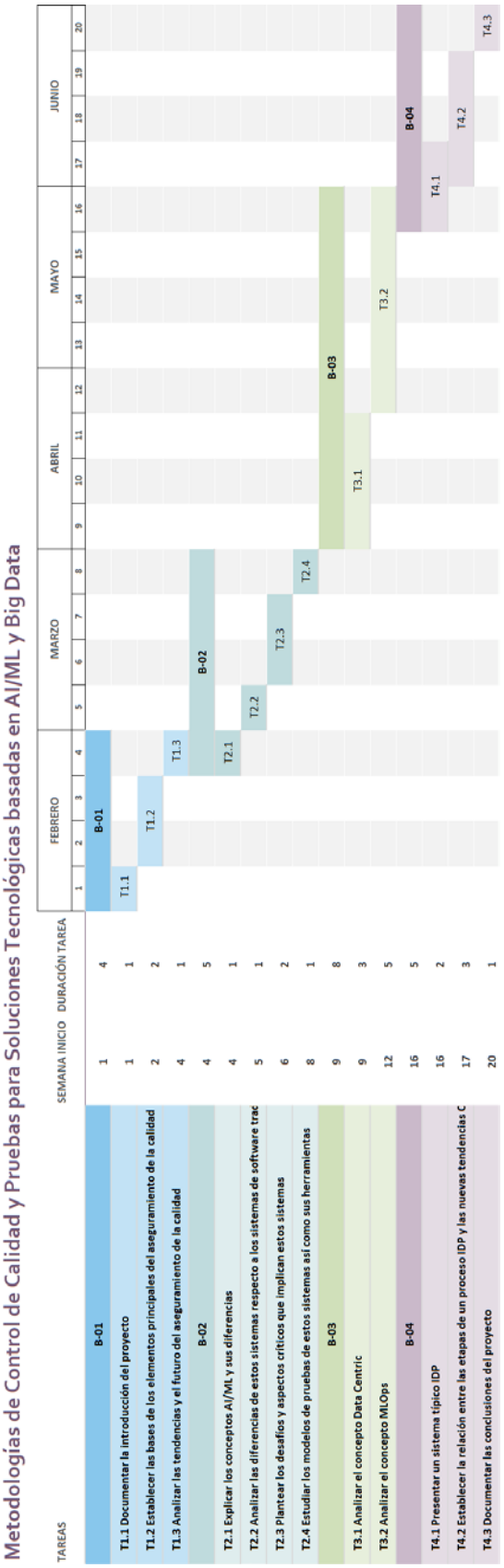

Figura 1.1: Diagrama de Gantt del proyecto

Parte II

Memoria

Capítulo 2

Procesos y Herramientas Quality Assurance (QA)

2.1. Definición de prueba y su importancia

La *prueba de software* es el proceso mediante el cual se encuentran errores en el producto desarrollado [3, págs. 13–16]. También verifica si los resultados reales coinciden con los resultados esperados y ayuda en la identificación de defectos, falta de requisitos o brechas. La fase de pruebas es uno de los últimos pasos antes del lanzamiento del producto al mercado. Incluye examen, análisis, observación y evaluación de diferentes aspectos de un producto. Después de realizar las pruebas, los probadores informan los resultados al equipo de desarrollo. El objetivo final es entregar un *producto de calidad* al cliente. Este es el motivo por el cual las pruebas de software son tan importantes.

Es común que muchas empresas se salten la fase de pruebas. Algunas empresas esgrimen argumentos económicos y los responsables piensan que no tendrá mayores consecuencias. Pero para dar una primera impresión positiva, es imprescindible probar el producto en busca de errores. En el lado contrario, también existen empresas que necesitan mantener su cartera de clientes y su nivel de calidad. Por lo tanto, deben garantizar la entrega de productos libres de errores al usuario final. A continuación se detallan algunos motivos que justifican por qué las pruebas son fundamentales para un buen desarrollo de software:

- **Mejoran la calidad del producto**. Una empresa puede aportar valor a sus clientes solo cuando el producto entregado responde a los requerimientos, es robusto y tolerante a fallos. Y para lograrlo, las organizaciones deben asegurarse que los usuarios no encuentren ningún problema al usar su producto. La forma infalible de conseguirlo es lograr que su producto esté libre de errores. Los equipos deben concentrarse en probar las aplicaciones y corregir los errores que se detectan durante las pruebas antes de lanzar el producto. La calidad de la entrega aumenta cuando el equipo resuelve los problemas antes de que el producto llegue al cliente.

- **Mejoran la ciberseguridad**. En ocasiones, cuando los clientes usan el producto, están obligados a revelar algún tipo de información personal. Para evitar que las diferentes amenazas en términos de ciberseguridad, como por ejemplo, el ransomware, phishing o troyanos se apoderen de estos datos o pongan en peligro la integridad del sistema, las pruebas de seguridad son imprescindibles antes de lanzar el software. Cuando una organización sigue un proceso de prueba adecuado, garantiza un producto seguro que, a su vez, hace que los clientes se sientan seguros mientras usan el producto. Por ejemplo, las aplicaciones bancarias o las tiendas de comercio electrónico necesitan los datos de pago.

- **Detectan compatibilidad con diferentes dispositivos y plataformas**. Actualmente, la sociedad está inmersa en la era de los dispositivos móviles y por ello es imprescindible probar la compatibilidad de un dispositivo con un producto. Si una empresa desarrolló un sitio web, el probador debe verificar si el sitio web se ejecuta en diferentes resoluciones de dispositivos. Además, también debería ejecutarse en diferentes navegadores para comprobar la compatibilidad de la aplicación en todos ellos.

2.2. Características de las pruebas

Existe una definición con nivel de norma, **ISO/IEC 25010** [4], que detalla las ocho características de calidad del modelo. A continuación se enumeran dichas características:

- **Adecuación Funcional**. Se refiere a la capacidad que tiene el producto que se desarrolla para proporcionar una serie de funciones que satisfagan las necesidades planteadas al inicio del proceso.

- **Eficiencia de Desempeño**. Se refiere a los recursos que se utilizan en unas condiciones concretas.

- **Compatibilidad**. Se refiere a la capacidad que tienen los sistemas o componentes para intercambiar información y/o realizar determinadas funciones.
- **Usabilidad**. Se refiere a la capacidad de un producto para ser usado por parte de un usuario final en determinadas condiciones.
- **Fiabilidad**. Se refiere a la capacidad que tienen los sistemas o componentes para realizar las funciones especificadas.
- **Seguridad**. Se refiere a la capacidad de evitar que personas o sistemas no autorizados puedan acceder a la información y los datos.
- **Mantenibilidad**. Se refiere a la capacidad de un producto para que sea modificado de forma eficaz ante ciertas necesidades correctivas o evolutivas.
- **Portabilidad**. Se refiere a la capacidad de un producto para que sea transferido de un entorno a otro de forma efectiva.

En la Figura 2.1 se visualizan dichas características:

Figura 2.1: Características de Calidad del Modelo. Fuente: [4]

2.3. Pruebas frente a Aseguramiento de la Calidad

Habitualmente se habla de pruebas y aseguramiento de la calidad refiriéndose al mismo concepto. Es como si se dijera que probador y asegurador de la calidad se refieren al mismo perfil profesional. En realidad no es así.

Las **pruebas de software** permiten explorar un sistema para comprobar cómo funciona y encontrar los posibles defectos. Se utilizan varios métodos para probar el producto, localizar errores y comprobar si se han solucionado. Las pruebas proporcionan a los clientes la posibilidad de ver si el producto desarrollado cumple con sus expectativas en cuanto a su diseño, compatibilidad o funcionamiento.

El **aseguramiento de la calidad** (*Quality Assurance*, QA, en inglés) es un conjunto de métodos y actividades diseñadas para garantizar que el software desarrollado se corresponda con todas las especificaciones a nivel de requisitos, diseño e implementación. Es una estrategia planificada de la evaluación del proceso de prueba dirigida al rendimiento de un producto de calidad. El aseguramiento de la calidad encuentra formas de prevenir posibles errores en el proceso de desarrollo de software. Y también se ocupa de otras cuestiones relacionadas con la gestión, como por ejemplo, los métodos de desarrollo o el análisis de proyectos.

En la Tabla 2.1 se observan las principales diferencias entre ambos conceptos:

	Aseguramiento de la Calidad	**Pruebas**
Definición	Actividad diseñada para asegurar que el software se corresponde con las especificaciones	El proceso de exploración de un sistema con el fin de encontrar defectos
Enfoque	Control de calidad y cumplimiento de los requisitos	Inspección del sistema y localización de errores
Orientación	Orientado a proceso	Orientado a producto
Actividad	Preventiva	Correctiva
Objetivo	Asegurar la calidad	Controlar la calidad

Tabla 2.1: Aseguramiento de la Calidad vs. Pruebas

2.4. Clasificación de las pruebas

Existen muchos tipos de pruebas para verificar que los cambios realizados en el código funcionan tal y como se esperan. Pero no todas son iguales puesto que suelen orientarse a comprobar determinados aspectos de un sistema de software. A continuación se cita una posible clasificación de pruebas [3, págs. 39–41] que se basa en las directrices del **ISTQB**[1].

[1] International Software Testing Qualifications Board, https://www.istqb.org/

2.4.1. Pruebas Funcionales

Las pruebas funcionales verifican cada función de una aplicación o software. El probador verifica la funcionalidad con un conjunto específico de requisitos. Probar el comportamiento del software o *lo que el sistema hace* es la principal preocupación. También se pueden considerar **pruebas de caja negra** porque lo que se verifica es el comportamiento externo del sistema sin importar lo que haya dentro.

En la Figura 2.2 se muestra un esquema sencillo sobre las pruebas de caja negra:

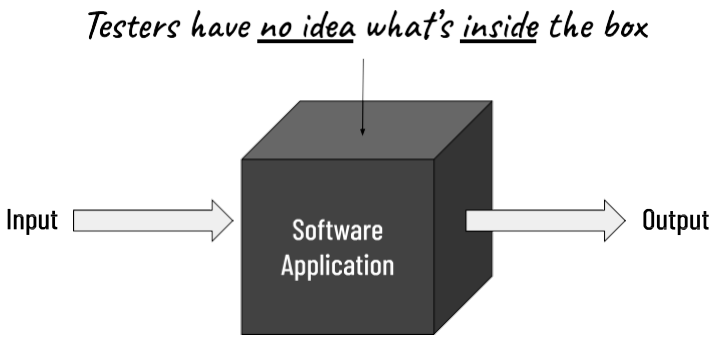

Figura 2.2: Pruebas de Caja Negra. Fuente: [5]

Las pruebas funcionales se pueden ejecutar en los siguientes niveles de pruebas:

- **Pruebas Unitarias**. Estas pruebas son relativamente pequeñas pero muy numerosas y se lanzan teniendo acceso al código que se encuentra en construcción y apoyado en los entornos de desarrollo. Esta situación provoca que se detecten defectos que son corregidos rápidamente por el desarrollador.

- **Pruebas de Componentes**. Estas pruebas solo aplican a cada componente de forma aislada sin integrar unos componentes con otros.

- **Pruebas de Integración**. Estas pruebas verifican que los diferentes componentes del software interactúan correctamente, no solo entre sí sino también con otras partes del sistema.

- **Pruebas de Sistemas**. Estas pruebas verifican el cumplimiento de las especificaciones del software cuando ha sido totalmente integrado.

2.4.2. Pruebas No Funcionales

Las pruebas no funcionales consideran parámetros como la confiabilidad, la usabilidad y el rendimiento. Por lo tanto, se centran en *cómo trabaja el sistema*. Estas pruebas también pueden ejecutarse en los niveles mencionados en el punto anterior (ver Sección 2.4.1) y en la mayoría de los casos se utilizan técnicas de **pruebas de caja negra**. Atienden a la siguiente clasificación:

- **Pruebas de Carga**. Estas pruebas simulan demanda sobre una aplicación de software y miden el resultado. Estas pruebas se realizan bajo demanda esperada pero también en condiciones de sobrecarga.
- **Pruebas de Estrés**. Estas pruebas se ejecutan con demandas mayores a la capacidad operativa, en ocasiones hasta llegar al punto de ruptura.
- **Pruebas de Rendimiento**. Estas pruebas verifican el rendimiento o la velocidad de la aplicación cuando se somete a la carga de trabajo requerida.
- **Pruebas de Usabilidad**. Estas pruebas exploran la facilidad de uso del usuario final en términos de aprendizaje y operación.
- **Pruebas de Mantenibilidad**. Estas pruebas evalúan la facilidad o dificultad para realizar el mantenimiento de un sistema.
- **Pruebas de Fiabilidad**. Estas pruebas validan los requerimientos asociados a las definiciones de disponibilidad y tolerancia a fallos de un sistema.
- **Pruebas de Portabilidad**. Estas pruebas determinan la facilidad o dificultad para mover un software de un entorno a otro.

2.4.3. Pruebas Estructurales

Las pruebas estructurales intentan verificar todos los posibles flujos de ejecución del software en base al código de la aplicación, por lo tanto, el usuario que diseña los casos de prueba debe tener acceso al código fuente. En este caso se aplican técnicas de **pruebas de caja blanca** dado que es necesario conocer cómo se ha desarrollado el sistema.

En la Figura 2.3 se muestra un esquema sencillo sobre las pruebas de caja blanca:

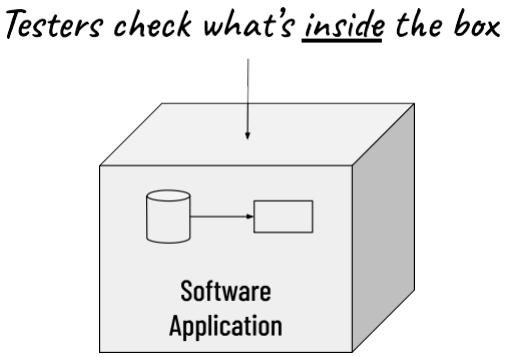

Figura 2.3: Pruebas de Caja Blanca. Fuente: [6]

Habitualmente se utilizan herramientas para calcular la cobertura del código. Estas herramientas, a través de un análisis del código de la aplicación y diferentes técnicas, permiten identificar los caminos que no han sido recorridos o las partes del código que no han sido probadas.

2.4.4. Pruebas Relacionadas con Cambios

Este tipo de pruebas se conocen de forma más habitual como **pruebas de regresión**. Cuando se soluciona un error que previamente fue detectado en una prueba, seguramente se añade o se elimina parte del código. Estas modificaciones podrían generar nuevos errores, no solo en el componente afectado, sino también en otros. Para evitar esta posible cadena de errores se ejecutan las pruebas de regresión. La regresión implica volver a ejecutar una serie de pruebas sobre un componente o módulo del sistema que ha sufrido una modificación para detectar posibles nuevos errores.

2.5. Etapas de las pruebas

En ocasiones se piensa en la fase de pruebas como una etapa única en la que se realizan las pruebas de software de un producto y que a su vez pertenece al ciclo de vida del desarrollo software, **SDLC**[2]. Y no solo eso, sino que además, esta etapa se realiza al final del proceso de desarrollo.

[2]Software Development Life Cycle.

Sin embargo, esta fase, que se podría denominar ciclo de vida de las pruebas software, **STLC**[3], también tiene sus propias etapas. Y cada vez de forma más habitual, dicha fase comienza más temprano en el ciclo de vida del desarrollo para garantizar realmente los mejores resultados. En la Figura 2.4 se pueden observar las etapas [7] de ambos ciclos de vida:

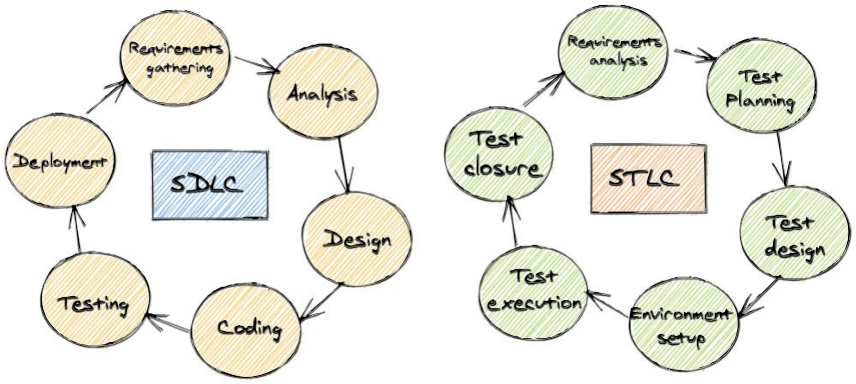

Figura 2.4: Ciclo SDLC vs. Ciclo STLC. Fuente: [8]

A continuación se presentan las etapas del ciclo STLC:

1. **Análisis de requisitos**. Se evalúan los requisitos del proyecto para ver si son claros, concisos, completos y comprobables. Esto no solo ayuda en la prevención de errores, sino que también facilita la realización de otras tareas de control de calidad posteriores.

2. **Planificación de pruebas**. Se define la estrategia que seguirán las pruebas y, al mismo tiempo, se determina el alcance, la planificación, los tipos de prueba, los procesos de seguimiento de errores y las técnicas de informes.

3. **Diseño y desarrollo de casos de pruebas**. Se parte de un plan de pruebas a partir del cual los probadores diseñan y desarrollan los casos de prueba. Estos casos deberían cubrir todas las situaciones posibles.

4. **Configuración del entorno de pruebas**. La configuración del entorno de pruebas es uno de los aspectos críticos del proceso de prueba. El equipo de pruebas deberá realizar una prueba de preparación llamada prueba de humo (*smoke test*, en inglés)[4] del entorno de pruebas para determinar si es satisfactoria.

[3]Software Testing Life Cycle.

[4]Revisión rápida de un producto de software para comprobar que funciona y no tiene defectos evidentes.

5. **Ejecución de pruebas**. Se ejecutan todas las pruebas diseñadas, incluyendo pruebas unitarias, pruebas de API, pruebas de UI[5], pruebas manuales y pruebas automatizadas, entre otras. También se generan informes detallados con toda la información recopilada de esas pruebas.

6. **Cierre de ciclo de pruebas**. El equipo de pruebas debe verificar y analizar los resultados de las pruebas. No solo se deben analizar los resultados sino que también se deben considerar otros factores como la calidad del producto final, la cobertura de las pruebas ejecutadas y el coste del proyecto.

2.6. Pruebas manuales frente a pruebas automáticas

Hoy en día existe un clara tendencia para categorizar las pruebas en dos tipos principales: pruebas manuales y pruebas automáticas [3, págs. 79–80].

La **prueba manual** es una prueba del software donde un probador ejecuta las pruebas manualmente y verifica todas las características esenciales de la aplicación o software dado. En este proceso, los probadores de software ejecutan los casos de prueba y generan los informes de prueba sin la ayuda de ninguna herramienta de prueba de software de automatización. Es un método clásico de todos los tipos de pruebas y ayuda a encontrar errores en los sistemas de software.

La **prueba automática** es una prueba del software donde el probador escribe código/guiones de prueba para automatizar la ejecución de la prueba. Los probadores usan herramientas de automatización apropiadas para desarrollar los scripts de prueba y validar el software. El objetivo es completar la ejecución de la prueba en menos tiempo. Las pruebas automatizadas le permiten ejecutar tareas repetitivas y pruebas de regresión sin intervención manual. Aunque todos los procesos se realizan automáticamente, la automatización requiere un esfuerzo manual para crear los scripts de prueba iniciales.

[5] User Interface.

En la Tabla 2.2 se observan las principales diferencias entre ambas opciones:

Manual	**Automática**
Hecho manualmente por evaluadores de control de calidad	Hecho automáticamente usando herramientas y scripts de automatización
Consume mucho tiempo y es menos eficiente	Más pruebas en menos tiempo y mayor eficiencia
Tareas totalmente manuales	La mayoría de las tareas se pueden automatizar
Difícil garantizar una cobertura de prueba suficiente	Fácil garantizar una mayor cobertura de prueba

Tabla 2.2: Pruebas Manuales vs. Pruebas Automáticas

La automatización de las pruebas, aunque inicialmente pueda suponer un desafío por el esfuerzo que supone crear los scripts de prueba, aporta importantes beneficios:

- **Ahorra tiempo**. La implementación de pruebas automatizadas puede ayudar a mejorar la calidad del código y la velocidad de desarrollo. Cuando los cambios de código no generen errores, los desarrolladores tendrán más tiempo para concentrarse en cada objetivo de entrega.
- **Mayor cobertura de prueba**. Las pruebas largas generalmente requieren mucho tiempo y son laboriosas de realizar manualmente. Esto se puede ejecutar con pruebas automatizadas desatendidas en diferentes máquinas y con diferentes configuraciones, generando una mayor cobertura. Las pruebas automatizadas también brindan a los probadores más tiempo y esfuerzo para concentrarse en otras tareas, lo que lleva a una mayor calidad de la aplicación.
- **Mayor precisión**. Los errores humanos son inevitables durante la ejecución de las pruebas manuales sobre todo cuando son repetitivas y monótonas. Si se ejecutan pruebas automatizadas, se pueden evitar los riesgos de errores humanos, aumentar la precisión y ahorrar tiempo.

Pero la apuesta por la **automatización** debe evaluarse detenidamente. Es necesaria una implicación por parte de toda la estructura de la organización puesto que no es una tarea exclusiva del equipo que ejecuta las pruebas. Y para que tenga éxito, puede que haya que cambiar los procesos, detallar más los requisitos o introducir nuevas prácticas de codificación.

También hay que definir la estrategia para encontrar la mejor herramienta de automatización puesto que seguramente no será única. Y sobre todo, una idea muy clara: no se puede automatizar todo.

Se debe definir un criterio claro respecto a esta tarea e identificar qué pruebas se deben automatizar y por qué vale la pena esfuerzo. Será una tarea de análisis continuo hasta alcanzar el nivel de madurez deseado.

2.7. Tendencias y futuro de las pruebas

La sociedad está inmersa en un escenario de cambio continuo que se ha visto superado debido a la pandemia [9]. El ritmo del proceso de **aceleración digital** ha sido alterado en unas pocas semanas. Las empresas han puesto el foco en la **digitalización** y eso también impacta en los modelos actuales de trabajo. Y por supuesto, si impacta en los modelos, también lo hará en las pruebas, puesto que forman parte de dicho modelo. Un objetivo importante que deben lograr los probadores es entregar productos más rápido y, lo que es más importante, mantener la calidad. A continuación se enumeran algunos de los posibles aspectos que marcarán la tendencia y el futuro de las pruebas:

1. **Pruebas Exploratorias**. Son un tipo de prueba en el que se explora la aplicación desde diferentes perspectivas para buscar riesgos y defectos. No es necesario redactar formalmente casos de prueba para realizar pruebas exploratorias; en su lugar, se sigue un enfoque sin guión que permite explorar una aplicación de forma intuitiva utilizando el conocimiento del probador. Por supuesto, para realizar pruebas exploratorias es necesario tener un profundo conocimiento del sistema desde el punto de vista del usuario final.

2. **TCoE**[6] **- Centro de Pruebas de Excelencia**. Dado que la calidad es cada día un aspecto más importante del software, este tipo de centros permiten potenciar y centralizar la calidad dentro de la organización. Se trata de crear un grupo que defina, implemente y defienda estos estándares de calidad para que todos los departamentos de la empresa sean partícipes de esta política. Los equipos de TCoE son muy útiles cuando una organización dispone de equipos de prueba dispersos en los diferentes proyectos. De tal forma que los probadores integrados en un TCoE tendrán más facilidades para establecer los estándares de calidad de una organización.

[6] Testing Center of Excellence.

3. **Automatización de Pruebas en el Sprint**. Las metodologías ágiles en los proyectos de desarrollo software persiguen ofrecer en poco tiempo pequeñas funcionalidades de software en funcionamiento obteniendo de esta forma la satisfacción del cliente. En los proyectos ágiles es necesario adaptar la automatización *in-sprint*. En este enfoque, la automatización de casos de prueba o pruebas de aceptación se programa y ejecuta dentro del mismo sprint. En la actualidad, muchos equipos realizan la automatización de pruebas en el sprint posterior al desarrollo o en un momento posterior antes de cualquier lanzamiento. Con el tiempo, esto solo provocará retrasos en la automatización. Al introducir la automatización en el sprint, se puede asegurar que cada compilación con las historias desarrolladas en el mismo sprint funcione como se espera y ayude a identificar los defectos en una etapa mucho más temprana.

4. **QAOps**. El término *DevOps* está de moda y cada vez tiene más impacto en la industria del software. Algunas empresas están comenzando a ver cómo la figura del asegurador de la calidad, podría desempeñar un papel similar con *QAOps*. Esta práctica se refiere a mantener la calidad del software al abordarlo con una mentalidad de *DevOps*. En términos simples, *QAOps* tiene como objetivo mejorar el proceso de entrega de software, haciéndolo más rápido y estable sin comprometer la calidad de su producto. *QAOps* toma las ideas centrales de las pruebas continuas en *DevOps*, como *CI/CD*, y reúne a los equipos para trabajar en el *pipeline*[7]. En la Figura 2.5 se pueden observar las actividades que integran *QAOps*:

Figura 2.5: QAOps. Fuente: [10]

[7]Canalización o tubería que consiste en una cadena de procesos conectados de forma tal que la salida de cada elemento de la cadena es la entrada del próximo.

5. **AI/ML**. Muchas empresas están adoptando herramientas de automatización de pruebas basadas en AI/ML como *Applitools*, *Functionize*, *ReportPortal*, (ver Sección 3.7) para fortalecer sus capacidades de prueba. Si las empresas comienzan a utilizar herramientas de pruebas basadas en AI/ML, sus procesos de prueba serán más robustos y rápidos. También los equipos de desarrollo comenzarán a recibir comentarios sobre los desarrollos mucho más rápido y, por supuesto, al evitar errores humanos también aumentará la confianza en las pruebas.

Capítulo 3

Aplicación de Procesos QA al área AI/ML

3.1. Introducción: AI/ML y sus diferencias

El uso de los términos **Inteligencia Artificial** (*Artificial Intelligence*, AI, en inglés), **Aprendizaje Automático** (*Machine Learning*, ML, en inglés) y **Aprendizaje Profundo** (*Deep Learning*, DL, en inglés) ha crecido mucho durante los últimos años [11], aunque son términos que existen hace tiempo. Incluso, en ocasiones, estos conceptos se entremezclan y aplican de forma errónea. Con lo cual, el primer paso debe ser aclarar estos conceptos.

La Inteligencia Artificial surge a finales de la primera mitad del siglo XX. Se trata de una rama de la Ciencia, y en concreto de la Informática, que se define como la capacidad de una máquina para resolver un problema tal y como lo haría una persona. En la década de 1950 fue **Alan Turing** quien sugirió que las máquinas también podrían usar la razón y la información disponible para tomar decisiones, llegando a escribir un artículo al respecto. Pero en aquella época, tanto el coste de las máquinas como su capacidad operativa fueron factores que limitaron el desarrollo de la AI. Era necesaria una prueba de concepto (*Proof of Concept*, PoC en inglés)[1] para lanzar el desarrollo de la AI. En 1956 hubo un intento de lanzar esta prueba por parte de un grupo de investigadores pero finalmente no fue posible. Entre 1957 y 1974 hubo una cierta expansión de la AI, gracias a la mejora de los ordenadores. Eran más económicos y tenían una capacidad de almacenamiento mayor.

[1] Es una implementación resumida de una idea, realizada con el propósito de verificar que el concepto o teoría en cuestión es susceptible de ser explotada de una manera útil.

Otro punto de inflexión fue la década de 1980 debido a que se disponía de más fondos económicos y a la expansión de las herramientas algorítmicas. Es en este momento cuando surge por primera vez el término *aprendizaje profundo*. A finales de esa década disminuyeron los fondos y la AI perdió el foco. Pero fue justamente en ese momento y durante las décadas de 1990 y 2000, cuando se produjeron los mayores avances en este campo. Y así hasta llegar a la actualidad, momento en el cual se puede decir que la AI está en todas partes.

Pero si además de resolver un problema como lo haría una persona, también se requiere que la máquina aprenda por sí sola, entonces se hablaría de **Machine Learning (ML)**, o lo que es lo mismo, Aprendizaje Automático.

Por último, y aunque no forma parte del alcance de este trabajo, se define el concepto **Deep Learning (DL)** o Aprendizaje Profundo [12] [13], que es una parte concreta del Machine Learning. En algunos artículos se considera que el Aprendizaje Profundo es lo más parecido a una simplificación de los procesos biológicos que suceden en nuestras neuronas. Sin embargo, la idea más extendida es que se trata de un proceso capaz de tomar decisiones adecuadas y concretas solo cuando la cantidad de datos a tratar es suficientemente grande, pero sin llegar a alcanzar los procesos de nuestro cerebro. El Aprendizaje Profundo ha permitido grandes avances en diferentes campos como el reconocimiento de voz, el reconocimiento de objetos visuales, la detección de objetos e incluso la genética. Y en ello ha influido notablemente el desarrollo de un método de aprendizaje, la *retropropagación*. Este algoritmo indica cómo una máquina debe cambiar los parámetros internos que se utilizan para calcular la representación en cada capa a partir de la representación en la capa anterior.

Por lo tanto, la Inteligencia Artificial, el Machine Learning y el Deep Learning son términos relacionados entre sí, de tal forma que cada uno engloba al siguiente. Esta relación es la que se muestra en la Figura 3.1:

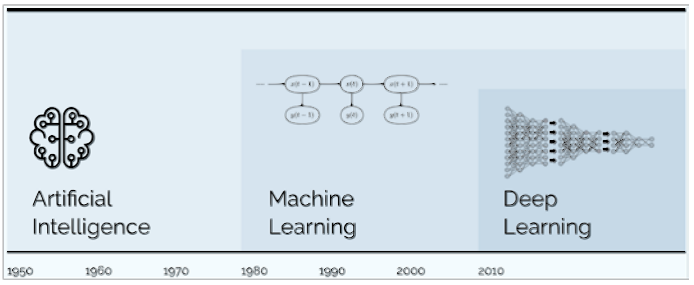

Figura 3.1: Relación AI, ML, DL. Fuente: [14]

Con lo cual, AI resuelve tareas que requieren inteligencia humana, mientras que ML es un subconjunto de AI que resuelve tareas concretas aprendiendo a partir de los datos y realizando predicciones. Esto quiere decir, como se aprecia en la Figura 3.1, que todo el ML es AI, pero no toda la AI es ML. La Tabla 3.1 muestra estas diferencias clave:

Inteligencia Artificial	**Machine Learning**
Aplica conocimientos o habilidades	Adquiere conocimientos o habilidades con el tiempo
Prioriza el éxito sobre la precisión	Prioriza la precisión sobre el éxito
Simula la inteligencia natural	Aprende continuamente de un conjunto de datos
Simula las respuestas humanas a los problemas	Crea algoritmos de aprendizaje continuo
Busca la solución óptima	Busca cualquier solución, sea óptima o no

Tabla 3.1: Inteligencia Artificial vs. Machine Learning

3.2. Sistemas de software tradicional vs. Sistemas AI/ML

Una vez explicados los términos AI/ML se debe tener en cuenta que el planteamiento de las pruebas de dichos sistemas no puede ser el mismo que para los sistemas de software tradicionales.

En estos últimos, las personas escriben la lógica que interactúa con los datos de entrada para generar el comportamiento deseado. Las pruebas de software ayudan a garantizar que esta **lógica escrita** se alinee con el comportamiento real esperado.

Sin embargo, en los sistemas de AI/ML, las personas aportan el comportamiento deseado como ejemplos de los datos de entrada y la salida esperada durante el entrenamiento. El proceso de optimización del modelo genera la lógica del sistema. En este caso el término es conocido como **lógica aprendida**.

La Figura 3.2 refleja las diferencias entre el ciclo de desarrollo de un sistema de software tradicional y el de un sistema AI/ML:

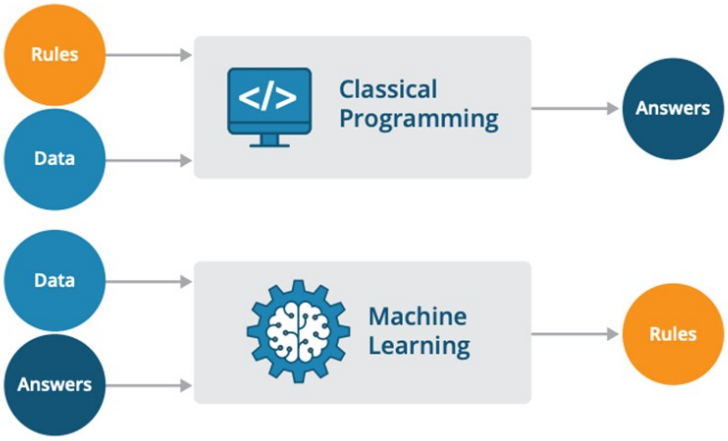

Figura 3.2: Sistema Tradicional vs. Sistema AI/ML. Fuente: [15]

El flujo de trabajo habitual de un sistema AI/ML incluye tres etapas:

1. **Gestión de Datos**. En esta etapa se adquieren y preparan los datos. En los sistemas AI/ML, los datos de entrenamiento son fundamentales en el rendimiento de un modelo. Se deben tener en cuenta los siguientes aspectos:
   - *Grandes conjuntos de datos*.
   - *Calidad de los datos*.
   - *Seguridad de los datos*.
2. **Experimentación**. En esta etapa se realiza el desarrollo del modelo, su entrenamiento y la evaluación posterior para verificar si alcanza los valores de precisión necesarios. Se deben tener en cuenta los siguientes aspectos:
   - *Calidad del código*.
   - *Evaluación de la precisión*.
   - *Reentrenamiento*.
3. **Implementación en Producción**. En esta etapa se dispone de un modelo entrenado y con cierta precisión que se despliega en producción para realizar predicciones con datos en vivo y/o no observados previamente durante la etapa de entrenamiento.

3.3. Desafíos de las pruebas en sistemas AI/ML

La AI está presente en todas partes. Inicialmente se desarrolló para dar servicio en las grandes empresas tecnológicas, pero hoy en día su uso se ha generalizado. Y por ello, porque una implementación de AI/ML sigue siendo en esencia un sistema de software, es necesario definir una estrategia de prueba adecuada, porque si no es así, no tendrá un buen resultado a nivel comercial. Se trata de un reto, puesto que no es lo mismo probar un software tradicional que probar los sistemas basados en AI/ML, que son mucho más complejos. No solo por el software, sino porque en este caso, los datos también provocan que el sistema aprenda y no dependa únicamente de las reglas que defina el software. En los sistemas basados en AI/ML el dato puede ajustar valores internamente que provoquen que la salida del sistema sea diferente.

A continuación se explican los **desafíos** [16] de las pruebas en los sistemas AI/ML:

- **No determinista**. Los sistemas AI/ML no son deterministas. Esto significa que tienden a mostrar diferentes comportamientos para la misma entrada.
- **Gestión de grandes volúmenes de datos**. Los sistemas AI/ML recopilan grandes volúmenes de datos. Esto crea conjuntos de datos inmanejables que presentan problemas de almacenamiento y análisis.
- **Datos de entrenamiento adecuados y precisos**. Los sistemas AI/ML dependen de los datos etiquetados. El 80 % del tiempo de un científico de datos se dedica a preparar el conjunto de datos de entrenamiento.
- **Propenso al sesgo humano**. Los sistemas AI/ML son entrenados y probados por personas y, por lo tanto, son vulnerables al sesgo[2] humano. Las pruebas de los sistemas AI/ML deben probar el sistema en busca de sesgo humano y eliminarlo.
- **Ampliación de defectos**. Los sistemas AI/ML pueden intensificar, en gran medida, un solo defecto, lo que dificulta la identificación del problema específico.
- **Desafíos de entrenamiento**. Después de probar y validar los sistemas tradicionales, no es necesario volver a probarlos hasta que se modifique el software. Sin embargo, los sistemas AI/ML aprenden, entrenan y se ajustan constantemente con nuevos datos y entradas. El desafío surge cuando hay eventos inesperados. En tal escenario, se vuelve difícil cotejar los datos y entrenar el sistema.

[2] Error sistemático en el que se puede incurrir cuando, al hacer muestreos o ensayos se seleccionan o favorecen unas respuestas frente a otras.

3.4. Aspectos críticos de las pruebas en sistemas AI/ML

Al igual que ocurre en las pruebas de los sistemas realizados bajo los protocolos del desarrollo de software tradicional, en las pruebas de los sistemas AI/ML también existen ciertos **aspectos críticos** [17] que se deben tener en cuenta al realizar las pruebas de dichos sistemas. Estas soluciones están basadas en procesos iterativos que necesitan asegurar la calidad del dato de entrada, evaluar la salida esperada en base a unas métricas y explorar la configuración de los hiperparámetros del modelo. Y todo ello eleva la complejidad de las pruebas que se deben realizar. A continuación se detallan los aspectos más críticos de las pruebas de los sistemas AI/ML:

- **Validación de datos**. La efectividad de un sistema AI/ML se basa en la calidad de los datos de entrenamiento, incluidos aspectos como el sesgo y la variedad, que se mencionaron en la sección anterior (ver Sección 3.3). Para ello es necesario etiquetar vídeos, imágenes y datos de texto, así como auditar y verificar los datos de prueba utilizados por el sistema AI/ML. Se realizan pruebas de precisión para validar si los datos representan de forma fehaciente el mundo real y se valida el conjunto de datos para cualquier sesgo implícito. La validación de los datos es fundamental para que los sistemas AI/ML dispongan de la entrada correcta.

- **Pruebas de algoritmos**. Los algoritmos, que procesan datos y brindan información, son el motor de los sistemas AI/ML. Las pruebas de los algoritmos implican validar el modelo, analizar su capacidad de aprendizaje y monitorizar su eficiencia. Algunos ejemplos de algoritmos son:

  - *Clustering*.
  - *Clasificadores*.
  - *Redes Neuronales Convolucionales*.
  - *Traductores de texto*.

- **Pruebas no funcionales**. Se deben realizar pruebas de escalabilidad, rendimiento y seguridad para los sistemas AI/ML. Se accede a muchos sistemas AI/ML a través de capas API mediante sistemas de usuario interactivos. Es necesario asegurar que la privacidad y la seguridad no se vean comprometidas a través de estos sistemas externos. Esto también incluye aspectos como el cumplimiento normativo. Estas pruebas no funcionales también producirán métricas operativas como velocidad de inferencia, precisión de clasificación y error absoluto medio.

- **Pruebas de integración de sistemas**. Los sistemas AI/ML están diseñados para operar en el contexto más amplio de otros sistemas y para resolver problemas específicos. Por lo tanto, las pruebas de integración son de suma importancia cuando se implementan juntos varios sistemas AI/ML con objetivos en conflicto. Cada vez son más y más sistemas que absorben características de AI/ML, con lo cual es importante que se prueben a fondo.
- **Pruebas de interacción inteligente**. En este apartado se podrían incluir las pruebas de todos aquellos dispositivos con los que se interacciona en mayor o menor medida casi de forma diaria y que incluso en algunos casos tienen la capacidad para aprender y modificar su comportamiento con el tiempo:
  - *Dispositivos (Siri, Alexa, etc.)*.
  - *Drones*.
  - *Coches autónomos*.
- **Monitorización y pruebas continuas en producción**. Una vez que se despliega el sistema en producción es necesario realizar tareas continuas de monitorización y pruebas. El modelo en producción recibe constantemente nuevos datos para hacer predicciones sobre él. Sin embargo, estos datos pueden tener una distribución de probabilidad diferente a la que ha entrenado el modelo. El uso del modelo original con la nueva distribución de datos provocará una caída en el rendimiento del modelo. Para evitar la degradación del rendimiento, se deben supervisar estos cambios. En términos generales, existen dos elementos que provocan estos cambios:
  - *Deriva del Concepto*. También denominado *Concept Drift* en inglés, se refiere a que las propiedades estadísticas de la variable de destino, que el modelo intenta predecir, cambian con el tiempo de forma inesperada.
  - *Deriva del Dato*. También denominado *Data Drift* en inglés, se refiere a una variación de los datos que se usaron para probar y validar el modelo antes de implementarlo en producción.

3.5. Tipos de Aprendizaje ML

Como ya se ha comentado en apartados anteriores, el aprendizaje automático (ML) representa una clase de software que aprende de un conjunto de datos dado y luego hace predicciones sobre el nuevo conjunto de datos en función de su aprendizaje. En otras palabras, los modelos de ML se entrenan con un conjunto de datos existente para hacer la predicción en un nuevo conjunto de datos. Las diferentes clases de algoritmos [18, págs. 26–27] de aprendizaje automático son:

- **Aprendizaje supervisado**. Estos modelos se entrenan con un conjunto de ejemplos en los que los resultados de salida son conocidos. De forma general, se puede describir como *orientado a tareas*, dado que su objetivo es proporcionar entradas de ejemplo al algoritmo hasta que pueda realizar la tarea con precisión. El aprendizaje supervisado se utiliza para resolver problemas que se pueden agrupar en las siguientes categorías:
  - *Regresión*. Los modelos de regresión se utilizan para hacer predicciones numéricas.
  - *Clasificación*. Los modelos de clasificación se utilizan para predecir la clase de un dato dado.

  A continuación se indican algunos de los ejemplos más habituales de aplicación para los algoritmos de aprendizaje supervisado:
  - *Popularidad de los anuncios*.
  - *Clasificación de spam*.
  - *Reconocimiento facial*.
- **Aprendizaje no supervisado**. Estos modelos tratan con datos sin etiquetar cuya estructura es desconocida. De forma general, se puede describir como *basado en datos*, dado que los resultados de estas tareas están controlados por los datos y sus propiedades. El aprendizaje no supervisado se utiliza para resolver problemas que se pueden agrupar en las siguientes categorías:
  - *Agrupamiento*. El agrupamiento, también conocido como clustering, es una técnica exploratoria de análisis de datos que se usa para organizar información sin tener un conocimiento previo de su estructura.
  - *Asociación*. Las reglas de asociación se utilizan de forma que permitan ayudar a la toma de decisiones mediante la identificación de relaciones existentes entre datos, denominados ítems.

A continuación se indican algunos de los ejemplos más habituales de aplicación para los algoritmos de aprendizaje no supervisado:

- *Sistemas de recomendación*.
- *Hábitos de compra*.
- *Agrupación de registros de usuarios*.

- **Aprendizaje reforzado**[3]. Se basa en aplicar los principios de psicología conductista a las inteligencias artificiales, con el fin de que puedan aprender por sí mismas en base a la experiencia adquirida. Dicho de otra forma, el aprendizaje por refuerzo se basa en *aprender de los errores*.

  A continuación se indican algunos de los ejemplos más habituales de aplicación para los algoritmos de aprendizaje reforzado:

  - *Videojuegos*.
  - *Simulación industrial*.
  - *Gestión de recursos*.

Esta clasificación es la que se muestra en la Figura 3.3:

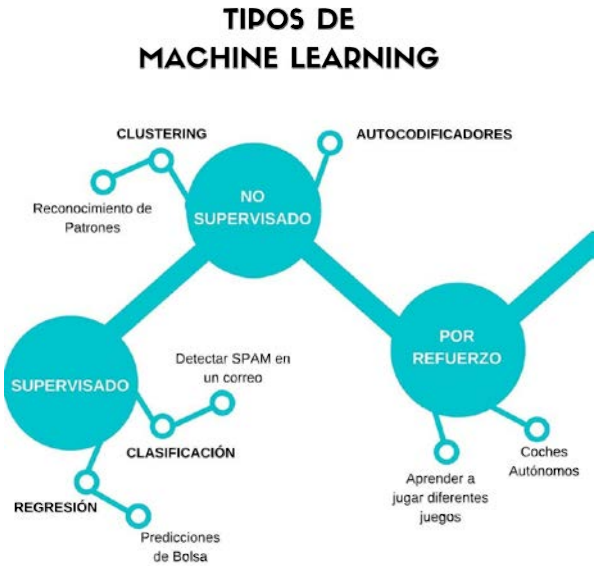

Figura 3.3: Tipos de Aprendizaje Automático. Fuente: [19]

[3]El aprendizaje por refuerzo es un área del aprendizaje automático cuya ocupación es determinar qué acciones debe escoger un agente de software en un entorno dado con el fin de maximizar alguna noción de recompensa.

3.6. Modelos de pruebas ML

En este apartado se van a describir una serie de modelos de pruebas que aplican a los sistemas ML, algunos de los cuales también se utilizan en las pruebas de los sistemas de software tradicional:

3.6.1. Pruebas de Caja Negra

Al igual que ocurre en las pruebas de los sistemas de software tradicionales, las pruebas de caja negra y caja blanca también se utilizan para los modelos ML [18, págs. 67–72]. En este caso, la dificultad radica en obtener conjuntos de datos de entrenamiento que sean lo suficientemente grandes y completos para adaptarse a los objetivos de las pruebas de los modelos ML. Durante la fase de desarrollo del modelo, los científicos de datos prueban el rendimiento del modelo comparando los resultados del modelo (valores previstos) con los valores reales. Algunas de las técnicas utilizadas para realizar pruebas de caja negra en modelos ML son:

3.6.1.1. Prueba de rendimiento del modelo

Esta técnica implica probar con datos de prueba o nuevos conjuntos de datos y comparar el rendimiento del modelo en términos de parámetros tales como precisión, exhaustividad y valor-F, con los valores predeterminados del modelo en producción.

Cualquier modelo realiza una clasificación sobre un conjunto de datos según se indica en la Tabla 3.2:

	Clasificado Positivo	**Clasificado Negativo**
Ejemplo Positivo	Verdadero Positivo (VP)	Falso Negativo (FN)
Ejemplo Negativo	Falso Positivo (FP)	Verdadero Negativo (VN)

Tabla 3.2: Matriz de Confusión

- *Verdadero Positivo (True Positive)*. El valor real es positivo y la prueba también predijo que era positivo.
- *Falso Positivo (False Positive)*. El valor real es negativo y la prueba predijo que el resultado es positivo.
- *Falso Negativo (False Negative)*. El valor real es positivo y la prueba predijo que el resultado es negativo.
- *Verdadero Negativo (True Negative)*. El valor real es negativo y la prueba también predijo que el resultado era negativo.

A continuación se explican los términos utilizados para cuantificar el rendimiento del modelo:

- **Precisión (Precision)**. La métrica de precisión permite medir la calidad del modelo ML en tareas de clasificación.

$$precision = \frac{VP}{VP + FP} = \frac{clasificados\ correctamente\ como\ positivos}{todos\ los\ clasificados\ como\ positivos}$$

  Donde lo ideal es que $precision = 1$

- **Exhaustividad (Recall)**. La métrica de exhaustividad permite medir la cantidad de información que el modelo ML es capaz de identificar con éxito.

$$recall = \frac{VP}{VP + FN} = \frac{clasificados\ correctamente\ como\ positivos}{todos\ los\ positivos}$$

  Donde lo ideal es que $recall = 1$

- **Valor-F (F1 Score)**. La métrica de valor-F se utiliza para combinar las medidas de precisión y exhaustividad en un solo valor. Esta métrica permite comparar varias soluciones en términos de rendimiento al combinar la precisión y la exhaustividad.

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

  Donde lo ideal es que $F1 = 1$

3.6.1.2. Prueba metamórfica

Esta técnica intenta reducir el problema del **oráculo**[4]. Para el caso más simple, un oráculo podría ser una comparación directa de la salida del programa con la respuesta correcta. Los oráculos más complicados pueden implicar ejecutar otro programa para determinar si la salida del programa de destino es correcta. Para los sistemas AI/ML, el problema es que a menudo no hay oráculo sin intervención humana. Se han propuesto varios métodos para mitigar el problema del oráculo, incluido el uso de pseudo-oráculo y pruebas metamórficas.

[4]Test oracle. En las pruebas de software, un oráculo se refiere a un mecanismo que puede decir si un programa funciona correctamente.

La prueba metamórfica [20] es una metodología de prueba de software para mitigar el problema del oráculo. La idea general es describir la funcionalidad del sistema en términos de relaciones genéricas entre entradas y las transformaciones genéricas de esas entradas y sus salidas, en lugar de mapeos de entradas específicas a salidas específicas. Esta idea se resume en el concepto de **relación metamórfica**. La transformación definida de la entrada debe tener un efecto conocido o medible en la salida y comprobar que esta relación se mantiene después de la transformación.

De forma más general, las pruebas metamórficas se refieren a definir tales transformaciones y observar su impacto en el resultado. La eficacia y la aplicabilidad dependen entonces de cómo y en qué extensión se puedan definir. Algunos de los campos de aplicación más habituales son los coches autónomos, la traducción de idiomas, las imágenes médicas o los drones autónomos.

Los conceptos que se deben tener en cuenta para realizar pruebas metamórficas en sistemas basados en ML son los siguientes:

- **Transformaciones metamórficas**. Estas transformaciones no tienen por qué ser muy complejas. Se debe considerar cómo la misma entrada podría cambiar en su entorno de uso previsto y cómo dicho cambio se podría implementar con un esfuerzo mínimo (o razonable) como una transformación.
- **Relaciones metamórficas**. Estas relaciones se construyen preguntando cómo se puede cambiar la entrada de ML y qué efecto debería tener en la salida. A veces, esto requiere un gran conocimiento del dominio para identificar los cambios más relevantes.
- **Oráculos**. Los oráculos verifican que la transformación realizada devuelve como resultado una salida aceptable.
- **Datos de prueba**. En este tipo de pruebas, al igual que en otras, la selección de los datos de prueba es muy importante. Si bien la automatización de las pruebas metamórficas puede ser bastante sencilla una vez que se determinan las relaciones y se crean las transformaciones, el espacio de datos de entrada, la cantidad de transformaciones y todas sus combinaciones pueden crecer rápidamente.

Dos décadas después de su introducción, las pruebas metamórficas se han convertido en una técnica madura y ampliamente aceptada, con diferentes aplicaciones. Algunas de las más destacadas son las siguientes:

- Motores de búsqueda, como por ejemplo, Google.
- APIs Web, como por ejemplo, Spotify.

- Sistemas de conducción autónoma, como por ejemplo, Apollo.
- Traductores, como por ejemplo, Google Translate.
- Sistemas de telemetría, como por ejemplo, NASA.

3.6.1.3. Codificación dual

La técnica propuesta se basa en los principios de la **Teoría de la codificación dual** planteada como hipótesis por Allan Paivio, de la Universidad de Western Ontario, en 1971. De acuerdo con esta teoría, nuestro cerebro utiliza dos sistemas diferentes, el verbal y el no verbal o visual, para recopilar, procesar, almacenar y recuperar la información relacionada con un tema en particular. Uno de los supuestos clave de la teoría de la **codificación dual** son las conexiones (también denominadas conexiones referenciales) que vinculan las representaciones verbales y no verbales en una red asociativa compleja. La Figura 3.4 muestra una representación de la teoría de codificación dual:

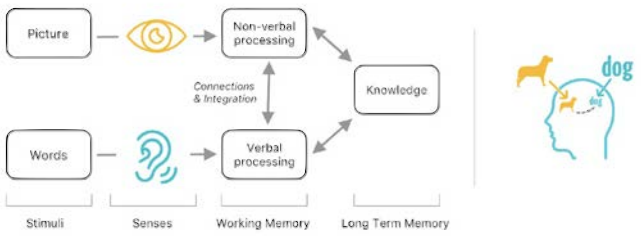

Figura 3.4: Principios de la Codificación Dual. Fuente: [21]

Se propone que esta misma idea se utilice para probar la predicción de los modelos ML y así determinar su corrección. En base a esta teoría, los modelos se construyen usando diferentes algoritmos y haciendo uso del mismo conjunto de características o similar. Más tarde, en el momento de probar las predicciones, se alimentan dos o más modelos con los mismos datos de entrada y se compara la corrección de sus predicciones. La idea es verificar la calidad del programa probando dos implementaciones diferentes del mismo programa (siendo una el programa principal) para un conjunto dado de entradas y comparando sus salidas para ver si son correctas. En caso de que haya predicciones del modelo principal que difieran de las realizadas a partir de modelos alternativos, se podrían plantear los defectos en el sistema de seguimiento de errores para que los científicos de datos los analicen más a fondo.

Los modelos ML se han denominado *no comprobables* debido a la ausencia de un oráculo. La ausencia de oráculo significaría que no se encontraron a los probadores para verificar la corrección o no se pudieron escribir los scripts de prueba o que es muy difícil probar el programa dada la complejidad asociada con la prueba. En el caso de los modelos ML, el valor esperado no se conoce de antemano, ya que se utilizan para realizar las predicciones. Por lo tanto, sería muy difícil escribir el script de prueba o encontrar probadores que pudieran verificar la exactitud de la salida del modelo con el valor esperado.

Dado que la prueba de codificación dual trata de probar la salida de un modelo comparando las salidas con las salidas de otros modelos creados usando diferentes algoritmos, esto podría automatizarse de la siguiente manera:

1. Los modelos creados con diferentes algoritmos se implementan en entornos de prueba.
2. Los modelos se exponen como servicios REST.
3. Un script de prueba invoca a los modelos utilizando el protocolo REST.
4. Se comparan las predicciones de cada modelo y si fuera necesario se plantea el defecto en la herramienta de seguimiento de errores.

3.6.1.4. Cobertura guiada

Fuzzing es una técnica conocida y ampliamente utilizada en los sistemas de software tradicionales. **Fuzz testing** es una técnica de prueba de software automatizada que implica proporcionar datos no válidos, inesperados o aleatorios como entradas a un programa de software.

Los errores de software a menudo surgen cuando se presentan entradas problemáticas al sistema. Si la lógica del programa no tuvo en cuenta estas entradas problemáticas, el componente de software puede fallar o comportarse de manera inadecuada. Las pruebas de fuzz buscan entradas problemáticas siguiendo una estrategia de generación automática de entradas. Con lo cual, si las entradas problemáticas se detectan de forma temprana, el sistema en general se vuelve más fiable y seguro.

Por lo tanto, es necesario encontrar las entradas que provocan que el sistema no se comporte correctamente. La primera idea podría ser utilizar una búsqueda totalmente aleatoria. Se podrían generar entradas modificando valores aleatoriamente hasta que algo se rompa. Sin embargo, esto tiene varias deficiencias. Fundamentalmente, es muy ineficiente, porque encontrar casos de error relevantes puede ser difícil y computacionalmente costoso. Además, los sistemas ML a menudo fallan en silencio. Esto quiere decir que los errores relevantes son sutiles y difíciles de encontrar y, por lo general, no provocan que el programa se *bloquee*.

Para aplicar con éxito las pruebas de fuzz, es necesario ser más eficiente que una búsqueda totalmente aleatoria. La mayoría de los enfoques se basan en la idea de mutar una entrada inicial en función de un conjunto específico de reglas y operaciones, como por ejemplo **DLFuzz**[5] [22] y **DeepHunter**[6] [23].

Las pruebas de fuzz se convierten en un componente primordial para probar los sistemas ML en la práctica. Permiten hacer una prueba de estrés del sistema y tener una idea más clara de cómo funcionará el sistema en la práctica al aprovechar un conjunto de datos sintéticos más grande. Esto es clave para comprender si el sistema funciona bien cuando debería y si el sistema falla correctamente cuando se le presentan entradas desafiantes. Es una forma de encontrar los puntos ciegos del sistema durante el desarrollo y evitar que sucedan durante la puesta en producción.

Algunas de las organizaciones más importantes del mundo, tales como Google, Microsoft o el Departamento de Defensa de los EE.UU., están implementando fuzzing como parte de sus operaciones de control de calidad y ciberseguridad.

3.6.2. Pruebas Retrospectivas

Estas pruebas también se conocen como **Backtesting**. Es un modelo predictivo basado en datos históricos. Esta técnica es popular para estimar el rendimiento de modelos anteriores. Principalmente se utiliza en los sectores financieros, especialmente para el comercio, la inversión, la detección de fraudes o la evaluación del riesgo crediticio pero es poco común su aplicación en los sistemas ML.

La idea para implementarlo es que en cada momento del conjunto de datos se entrena el modelo con datos conocidos/pasados en ese momento y se prueba con datos desconocidos/futuros en ese momento. Algunas de sus características más importantes son las siguientes:

- Dada su secuencialidad y que las condiciones son lo más cercanas a la situación real, estas pruebas ofrecen la mejor garantía posible sobre el rendimiento del modelo en el futuro.
- No se utilizan conjuntos de datos de validación dado que cada punto de los datos es parte de los conjuntos de prueba y entrenamiento.
- No es necesaria ni tampoco se recomienda la validación cruzada porque la selección de los hiperparámetros del modelo se realiza de forma secuencial.

[5] Differential Fuzzing Testing of Deep Learning Systems.
[6] Coverage-Guided Fuzz Testing Framework for Deep Neural Networks.

- Se puede ajustar el tamaño y comportamiento de los conjuntos de entrenamiento y prueba en función de las características del modelo siempre que se mantenga la separación entre datos pasados y datos futuros.

Uber ejecuta una de las infraestructuras de aprendizaje automático más grandes del mundo. Uber ejecuta miles de modelos de pronóstico en diversas áreas, como la planificación de viajes o la gestión de presupuestos. Y además debe intentar garantizar la precisión de esos modelos de pronóstico. La cantidad de modelos y la escala de computación hacen que el entorno de Uber sea relativamente poco práctico para la mayoría de los marcos de backtesting. Por ese motivo, Uber presentó un nuevo servicio completamente creado desde cero para realizar pruebas retrospectivas de los modelos de aprendizaje automático a escala.

No todos los backtests son iguales. En el caso de Uber, el gigante del transporte necesitaba considerar elementos como la cantidad de ciudades o la ventana de prueba para respaldar los modelos de prueba de manera eficiente. Los modelos que funcionan bien para una ciudad no necesariamente funcionan bien para otra. De manera similar, algunos modelos necesitaban ser probados en tiempo real, mientras que otros pueden permitirse ventanas más grandes. Uber identificó cuatro vectores clave que eran relevantes para realizar una prueba retrospectiva de los modelos de pronóstico:

- *Número de ventanas de backtesting*.
- *Número de ciudades*.
- *Número de parámetros del modelo*.
- *Número de modelos de pronóstico*.

3.6.3. Pruebas Requisitos No Funcionales

La característica fundamental de los modelos ML es permitir que los algoritmos aprendan. Pero recientemente se presta también mucha atención a otras cualidades de estos modelos como por ejemplo, la privacidad, la seguridad, la capacidad de prueba, la transparencia, las cuestiones éticas, los diferentes tipos de sesgos e incluso la eficiencia energética y el impacto medioambiental. Todas estas cualidades se denominan **Requisitos No Funcionales** (*Non Functional Requirements*, NFR, en inglés). Aunque el significado y uso de algunos de estos requisitos ya hace tiempo que se establecieron en los sistemas de software tradicional, en los modelos ML no se puede aplicar la misma lógica.

La seguridad de un sistema, y últimamente con mayor motivo, se ha convertido en una característica fundamental. La ejecución de pruebas de seguridad ya es un elemento básico en el desarrollo de un sistema para su puesta en producción. Y a pesar de ello y de todos los recursos destinados, seguirán existiendo brechas de seguridad. A continuación se exponen algunos ejemplos conocidos:

- *Canva*. En mayo de 2019, un pirata informático rompió la seguridad de este sitio web de diseño gráfico y robó los datos de más de 139 millones de usuarios.
- *Google Plus*. En diciembre de 2018, Google informó que, debido a un error en la actualización de la API de Google+, los detalles de más de 52,5 millones de usuarios se vieron afectados. El error permitía el acceso a desarrolladores externos.
- *Yahoo*. Los datos de más de 500 millones de usuarios se vieron afectados en 2014 debido a un ciberataque.

3.7. Herramientas de pruebas en sistemas AI/ML

En este apartado se van a presentar una serie de herramientas que se basan en sistemas AI/ML y que se utilizan para realizar pruebas. Se trata de automatizar los procesos de prueba con herramientas basadas en AI/ML. Dichas herramientas se pueden clasificar en cinco categorías [24] diferentes:

3.7.1. Diferenciales

Las herramientas diferenciales identifican los problemas de calidad del código o las vulnerabilidades de seguridad y también se utilizan para generar regresiones. Se basan en el escaneo del código y a partir de ahí, realizan automatizaciones de pruebas unitarias. De esta forma, se generarán versiones de forma más rápida, la calidad mejorará y sobre todo, la productividad del equipo aumentará dado que no tendrá que abordar la realización de estas tareas. Algunos ejemplos de este tipo de herramientas son las siguientes:

- **Launchable**[7]. Launchable es una plataforma de inteligencia para todas las pruebas de software que acorta los tiempos de espera para las pruebas, acelera y optimiza la eficiencia del proceso de integración continua, lo que lleva a enviar el software de mayor calidad más rápido.

[7] https://www.launchableinc.com/

- **DiffBlue**[8]. Diffblue Cover utiliza la AI para escribir automáticamente pruebas unitarias en código Java. También escribe pruebas de regresión que encuentran errores en los cambios de código que se realizan.

3.7.2. Visuales

A diferencia de las herramientas diferenciales, las pruebas visuales se centran en la capa de la experiencia del usuario y la apariencia de una interfaz de usuario en todas las plataformas digitales posibles (móviles y web principalmente). Estas herramientas deben abordar los cambios constantes realizados en la capa de la interfaz de usuario junto con un número cada vez mayor de plataformas, tamaños de pantalla y configuraciones que hacen que la cobertura de prueba crezca de forma exponencial y que dificulte la labor del equipo que realiza y diseña los casos de prueba. Algunos ejemplos de este tipo serían:

- **Applitools**[9]. Applitools es un programa de monitorización y prueba de interfaz de usuario visual impulsado por AI. Es una plataforma de prueba de software de extremo a extremo que pueden utilizar los ingenieros y los probadores manuales, así como los equipos de automatización de pruebas, *DevOps* y transformación digital.
- **Percy.io**[10]. Percy prueba automáticamente la interfaz de usuario en navegadores y pantallas, ahorrando tiempo y recursos que se gastan en pruebas manuales.

3.7.3. Declarativas

Las herramientas declarativas tienen como objetivo mejorar la productividad y la estabilidad de la automatización de pruebas. Sus tareas principales son eliminar acciones tediosas, propensas a errores y repetitivas, a través de la automatización inteligente. Ejemplos de este tipo de herramientas podrían ser:

- **Functionize**[11]. Functionize es una ventanilla única para la creación, ejecución, mantenimiento y análisis de pruebas, con capacidades para pruebas funcionales, visuales, de rendimiento y de carga.
- **Tricentis**[12]. Tricentis ofrece una forma fundamentalmente diferente de abordar las pruebas de software, acelerando drásticamente la transformación digital, la entrega de aplicaciones y la migración a la nube. El enfoque de pruebas continuas está totalmente automatizado, sin código e impulsado por AI.

[8] https://www.diffblue.com/
[9] https://applitools.com/
[10] https://percy.io/
[11] https://www.functionize.com/
[12] https://www.tricentis.com/

3.7.4. Autorreparativas

Algunos de los motivos principales por los que los sistemas AI/ML se introducen en el espacio de la automatización de pruebas serían la inconsistencia, la confiabilidad y el mantenimiento de la automatización de las pruebas. Cuando un conjunto de pruebas se automatiza, casi siempre requiere de ajustes por cambios en la plataforma o en el entorno. Por ello surgen nuevas herramientas que se basan en mecanismos de grabación y reproducción y donde el motor de ML reside en la autorreparación de esos scripts grabados. Como ejemplos se podrían nombrar las siguientes herramientas:

- **Testim**[13]. Es una plataforma de pruebas funcionales automatizadas basada en AI/ML que acelera la creación, ejecución y gestión de pruebas automatizadas. Chrome, Firefox, Edge, IE, Safari y Android se encuentran entre los navegadores y sistemas operativos que puede usar la herramienta.
- **Mabl**[14]. Mabl permite integrar pruebas automatizadas de extremo a extremo en todo el ciclo de vida del desarrollo de software.

3.7.5. Reporte y análisis

Independientemente de la herramienta que finalmente se utilice para gestionar las pruebas de la aplicación, si se escalan las versiones de software que se generen, también se están escalando los datos de prueba y los informes que genera. Los equipos de trabajo deben dar sentido a todas las fuentes de datos que recibe el sistema y tomar las decisiones más acertadas basadas en dichos datos. Y para ello es necesario clasificar los datos, dividirlos e incluir la causa de los errores. Para todo ello se puede utilizar la siguiente herramienta:

- **ReportPortal**[15]. Se trata de un tablero de control de automatización de pruebas impulsado por AI para adquirir, agregar y analizar informes de prueba para determinar el estado de la versión.

Una vez vista esta clasificación relativa a las herramientas automáticas, también se podrían considerar las herramientas manuales. Las herramientas que se pueden usar para desarrollar un modelo de aprendizaje automático también se pueden usar para probar dicho modelo.

[13] https://www.testim.io/
[14] https://www.mabl.com/
[15] https://reportportal.io/

A continuación se enumeran algunas de las herramientas más utilizadas:

- **WEKA**. Weka es una colección de algoritmos ML para tareas de minería de datos. Contiene herramientas para preparación de datos, clasificación, regresión, agrupamiento, asociación reglas de minería y visualización. Weka es un software de código abierto.
- **PyCharm**. PyCharm es un *entorno de desarrollo integrado* (*Integrated Development Environment*, IDE, en inglés) utilizado en la programación informática, específicamente para el lenguaje de programación Python. Este IDE permite la ejecución de diferentes estructuras o marcos de trabajo (*frameworks*, en inglés) de pruebas [25]. Algunos de ellos son los siguientes: Python unittests, Pytest, Python nosetests, tox, TwistedTrial ó Python doctests. Y algunas de sus características, no presentes en todos ellos, son las siguientes: ejecutar o depurar la configuración, creación de pruebas, ejecución de pruebas e inspecciones de código.
- **Spyder**. Spyder es un entorno de desarrollo integrado y multiplataforma de código abierto para programación científica en el lenguaje Python. Al igual que PyCharm, dispone en este caso de Spyder Unittest, un marco de trabajo de pruebas unitarias que permite ejecutar conjuntos de prueba. Además, también soporta los módulos Python unittests, Pytest y Python nosetests que vimos en el IDE PyCharm.

Capítulo 4

Nuevas Tendencias de Procesos QA en AI/ML

4.1. Introducción

Un sistema AI, básicamente, es la suma de dos elementos: **Código** + **Datos**. El código se refiere al *modelo* o *algoritmo* que se implementa. Pero también se refiere al pre- y post-procesado, la fase de integración y las interfaces. Si además se incluye el modelo aprendido, tendremos una gran cantidad de parámetros en forma de valores numéricos (desde cientos a billones, dependiendo del tamaño de la solución ML y/o red neuronal) que son los que se calculan durante la fase de entrenamiento. La combinación de estos parámetros junto con los hiperparámetros, que se podrían definir como los elementos de configuración del modelo que se pueden ajustar durante el entrenamiento, es fundamental para alcanzar los resultados esperados. Por otro lado, los datos se refieren al *conjunto de datos* o *dataset* que se utilizará para entrenar y validar el modelo mencionado anteriormente.

En esta sección se explicará cuál ha sido el enfoque durante mucho tiempo y por qué y hacia dónde confluyen las nuevas tendencias en el desarrollo de soluciones de sistemas AI/ML. Dentro de esas nuevas tendencias se mencionarán, principalmente, **Data Centric** y **MLOps**[1].

4.1.1. Model Centric

Model Centric se refiere al concepto *centrado en el modelo*. Hasta no hace mucho tiempo, la comunidad científica ha dedicado todos sus esfuerzos a la construcción y mejora de nuevos modelos.

[1] Machine Learning Operations.

Si el enfoque se centra en el modelo hay que preguntarse cómo se puede cambiar el modelo para mejorar el rendimiento. Esta tendencia centrada en el modelo se justifica con varios argumentos:

1. Los profesionales de este área pueden aplicar todos sus conocimientos para mejorar los algoritmos.
2. Una gran parte de la investigación académica en la rama de AI también está centrada en la mejora de los modelos y en cierta medida, son las grandes empresas tecnológicas las que financian dichas investigaciones.
3. La digitalización, los avances relativos a la capacidad de computación y almacenamiento, la expansión de las soluciones *cloud* y la ubicuidad de los datos han permitido el acceso a enormes cantidades de datos y al hardware especializado en procesarlo. De esta forma, se pasó de los algoritmos clásicos de ML, a las redes neuronales y sus versiones cada vez más profundas. Por lo tanto, las mejoras de los sistemas AI/ML se buscan aumentando la capacidad de dichos modelos y sus diferentes configuraciones e hiperparámetros.
4. Además, debido al éxito del Aprendizaje Profundo, se exploran cada vez más aplicaciones dando lugar a múltiples modelos adaptados a cada aplicación concreta o bien, grandes modelos entrenados con un conjunto de datos pero aplicables a otros conjuntos de datos distintos, como, por ejemplo, las redes neuronales profesor-alumno y otras estrategias de transferencia del aprendizaje (*transfer learning*, en inglés).
5. Igualmente importante, crece la necesidad de explicar qué pasa dentro de esas redes neuronales y cómo toman decisiones u optimizan la fase de entrenamiento sin perder precisión en los resultados. Esto conlleva el estudio detallado de los modelos AI/ML y sus razones.

Bajo este enfoque, los sistemas ML se desarrollan mediante un proceso *iterativo* [26] de realización de pruebas con el objetivo de mejorar el rendimiento del modelo. En este caso, se recopilan todos los datos posibles y el modelo se desarrolla en base a dichos datos. Dicho de otra forma, los datos se mantienen fijos y se mejora el modelo hasta que se alcanzan los resultados esperados. Este enfoque requiere más tiempo para mejorar el rendimiento del modelo. También se necesita mucha experiencia y conocimiento tanto de los modelos ML como del problema planteado.

4.1.2. Evolución a Data Centric/MLOps

De lo comentado en la sección anterior (ver Sección 4.1.1) se puede deducir que existen argumentos de peso para pensar que la tendencia centrada en el modelo, sin ser perfecta, cumple su papel y satisface a los profesionales que se encargan de realizar dicha tarea. Entonces, ¿por qué es necesario un cambio de tendencia? ¿Qué justifica dicho cambio? Los argumentos podrían ser los siguientes:

1. La actividad del área AI se centró durante tanto tiempo en el modelo que hoy en día existen herramientas en el mercado que permiten aplicar una gran parte de los algoritmos sin necesidad de realizar ningún tipo de codificación. Dicho de otra forma, la arquitectura del modelo ya es lo suficientemente buena y mantener un enfoque centrado en el modelo no mejorará demasiado el rendimiento de los algoritmos.

2. Los datos nos están invadiendo. Hasta ahora se partía de grandes conjuntos de datos que servían como referencia para entrenar los modelos y mejorar su rendimiento. Pero no todo es volumen, sino que se necesitan unos datos de calidad. Es necesario trabajar con datos que sean mejores porque la mejora en la calidad de los datos garantizará una mejora en el rendimiento del modelo.

3. A nivel práctico, hay pequeñas y grandes empresas que no son capaces de generar o adquirir el volumen de datos necesarios para aplicar AI/ML. Puede ser por disponibilidad del dato o por coste y esfuerzo de obtenerlo y mantenerlo. Por ello, menos datos pero de más calidad pueden hacer viable muchos casos de uso nuevos para las tecnologías de AI/ML.

En la Tabla 4.1 se observan las principales diferencias entre las tendencias *Model Centric* y *Data Centric*:

Model Centric	**Data Centric**
Trabaja el código	Trabaja sobre los datos
Optimiza el modelo	Herramientas de calidad de datos
Datos inconsistentes	Datos consistentes
Corrigen datos tras preprocesamiento	Códigos/algoritmos fijos
Mejora iterando el modelo	Itera la calidad de los datos

Tabla 4.1: Centrado en el Modelo vs. Centrado en el Dato

Pero no basta con cambiar la tendencia y *centrarse en los datos*, porque, como ya se ha comentado, dado el volumen que se maneja, es necesario acotar todo ello en un marco o metodología. Dicho marco se denomina **MLOps** u Operaciones de Aprendizaje Automático. Se trata de una colección de herramientas que respalda las etapas del ciclo de vida de la AI centrada en los datos como, entrenar el modelo, realizar análisis de errores, obtener más datos o dar coherencia a sus etiquetas y que la convierten en un proceso eficiente y sistemático. *MLOps* nace con una filosofía similar a **DevOps**[2] pero son bastante diferentes en su ejecución.

Uno de los principales promotores de este cambio de tendencia o evolución es **Andrew Ng** [27], profesor asociado en el departamento de Ciencias de la Computación y del departamento de Ingeniería Electrónica de la Universidad de Stanford. También trabaja como director del laboratorio de Inteligencia Artificial en Stanford y es el cofundador de Coursera, la plataforma de educación en línea. Además, es el fundador de DeepLearning AI, empresa creada para satisfacer la necesidad de una educación en AI a nivel mundial. Esta persona está lanzando la campaña que persigue el cambio de tendencia y de enfoque por parte de los profesionales de la AI. Diferentes estudios demuestran que el 80 % del tiempo se dedica a la preparación de los datos. Además, a medida que los grandes conjuntos de datos se vuelven más habituales, se utilizan estos conjuntos para alimentar los modelos y superar los errores sin tener en cuenta que estos datos pueden ser de baja calidad o ruidosos. Es cierto que trabajar con los datos, sobre todo cuando el volumen es muy grande, puede ser una tarea muy tediosa, pero si se consigue alcanzar un baremo de calidad, el rendimiento del modelo mejorará. Esta idea es la que defiende Andrew Ng, la **calidad de los datos**.

La Figura 4.1 muestra de forma muy gráfica la diferencia esencial entre ambas tendencias:

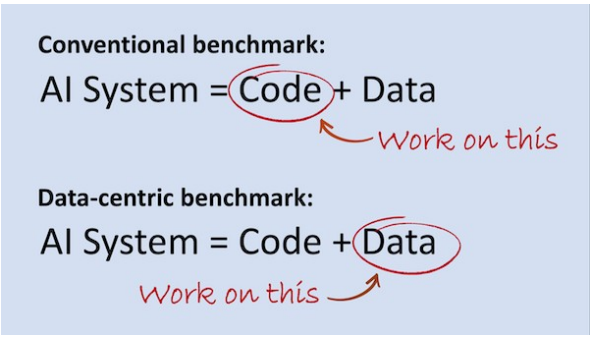

Figura 4.1: Centrado en el Modelo vs. Centrado en el Dato. Fuente: [28]

[2]Development Operations.

4.2. Data Centric

La tendencia que se *centra en el dato* está liderando la **transformación digital** de las empresas. Hasta ahora se trataban los datos, pero no de forma inteligente. Esta situación genera muchas ineficiencias porque los datos no están organizados y por lo tanto no se pueden explotar correctamente. Ahora las empresas necesitan dar el protagonismo al dato. Y una vez que el dato sea el protagonista, se le podrá aplicar una capa de inteligencia a través de los sistemas AI/ML. De esta forma, el negocio podrá disfrutar de este dato transformado en tiempo real.

Es cierto que los científicos de datos piensan que las tareas relacionadas con el preprocesamiento de los datos, la limpieza y el etiquetado no son propias de su área. Pero es necesario cambiar este enfoque dentro de la comunidad. Muchos estudios han demostrado que una vez que la iteración del modelo ha llegado a su techo, solo se mejora el rendimiento aumentando la calidad de los datos.

4.2.1. Definición y principios

Data Centric se refiere al concepto *centrado en el dato*. Esto implica que ahora la dedicación se centra en etiquetar, administrar, aumentar, dividir y reparar los datos de una forma eficiente, manteniendo el modelo relativamente fijo. Pero como se mencionó en el apartado anterior, no se trata de cambiar únicamente el enfoque técnico sino que también es necesario un **cambio en la cultura** y en la comunidad científica [29]. Es tanto un cambio tecnológico como metodológico. Por supuesto, esta tendencia no trata de desterrar el enfoque centrado en el modelo. Para que un sistema AI/ML tenga éxito será necesario combinar ambas tendencias de forma coherente y lógica.

De la misma forma que existen modelos para desarrollar los algoritmos de los sistemas AI/ML, también debería existir una metodología o un procedimiento para depurar los datos. Se pueden enumerar algunos consejos útiles:

1. Las etiquetas de los datos deben ser consistentes y se deben asignar de una forma determinista, no aleatoria.
2. Es necesario utilizar varios etiquetadores para detectar las inconsistencias de los datos.
3. Se debe definir la forma de etiquetar y debe estar documentado en unas instrucciones.

4. Una vez detectados los datos ruidosos, hay que descartarlos. El volumen no siempre aporta calidad.

5. Con el análisis de errores es posible centrarse en los subconjuntos de datos con mayor área de mejora.

Todos estos pasos forman parte de un ciclo iterativo, no se ejecutan una sola vez. Igual que se iteran los modelos ajustando los hiperparámetros, también se iteran los datos para mejorarlos en cada ciclo.

4.2.2. Data driven vs. Data centric

La fiebre del dato invade la sociedad y las organizaciones. El término **Big Data**[3] provoca que todas las empresas quieran acumular datos. Un alto porcentaje de las grandes empresas están ejecutando proyectos o invirtiendo en Big Data. Pero tener más y más datos no significa que la empresa esté centrada en el dato o tenga esa filosofía de desarrollo o negocio. De hecho, si las empresas se dedican a acumular conjuntos de datos, cada uno con su propio modelo y simplemente los almacena en un *data lake*[4] en realidad se estará alejando respecto a la tendencia a la que pretendía acercarse.

Por lo tanto, es importante conocer la relación de la empresa con el dato para saber en qué estadio de la relación se encuentra. Según el grado de relación [30] se pueden definir tres conceptos:

- **Data Informed**. El concepto se puede traducir como *informado por los datos*. Este nivel sería el más bajo en relación a la integración de los datos en la empresa. Los datos se recopilan y se organizan. El personal de la empresa sabe cómo debe acceder y utilizar dichos datos, pero no existe un concepto de trabajo y uso relacionado con la ciencia de datos o con los sistemas ML.

- **Data Driven**. El concepto se puede traducir como *basado en los datos*. Este nivel de integración utiliza los datos de forma activa para tomar decisiones a nivel empresarial. Estos datos deben ser presentados como una información que permita tomar decisiones. Los equipos de trabajo conocen el potencial de estos datos pero no se llega a explotar en su totalidad.

[3] Conjuntos de datos tan grandes y complejos que precisan de aplicaciones informáticas no tradicionales de procesamiento de datos para tratarlos adecuadamente.

[4] Datos almacenados en su formato natural, sin procesar.

- **Data Centric**. El concepto se puede traducir como *centrado en los datos*. Una definición de este concepto podría ser la siguiente: *'Una empresa centrada en datos es aquella en la que toda la funcionalidad de la aplicación se basa en un modelo de datos único, simple y extensible'* [31]. Sería el nivel más alto de integración de datos que se puede lograr en la infraestructura de una empresa. Este marco permite crear un modelo central de datos fiable y reutilizable, con lo cual, se evita la formación de silos de información para cada proyecto o iniciativa. En este caso, el núcleo principal serán los datos. El resto, las aplicaciones y los *pipelines* serán efímeros para cada proyecto. Los datos son los que perdurarán en el tiempo.

En ocasiones tiende a confundirse la acumulación de datos con el concepto de estar centrado en los datos. Muchas empresas tienen esa capacidad para recopilar y almacenar grandes cantidades de datos. Pero si los equipos no tienen una metodología para aplicar a esos datos, el resultado serán únicamente una gran cantidad de datos. Y de esta forma las aplicaciones seguirán utilizando sus datos en un silo separado que no utilizarán el resto de aplicaciones. Con lo cual, en lugar de estar centrada en los datos, la empresa estará centrada en las aplicaciones o en el modelo. La Figura 4.2 muestra la filosofía de estas dos tendencias relacionadas con el dato:

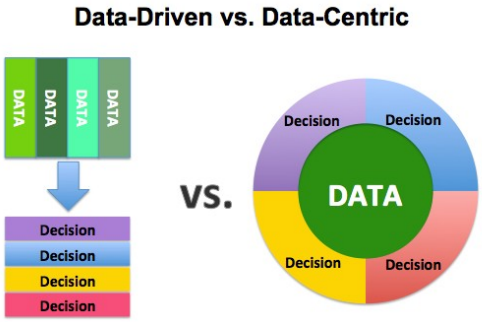

Figura 4.2: Basado en Datos vs. Centrado en Datos. Fuente: [32]

Si la organización quiere centrarse en los datos necesita un modelo, herramientas y las habilidades necesarias para llevarlo a cabo. Ese modelo debe especificar de forma clara cómo usará los datos y después, esos datos deben adaptarse al modelo. Pero quizá lo más importante sea implantar una cultura, **la cultura de los datos**. El dato debe ser el activo permanente y continuo de la organización. El resto de elementos, como las aplicaciones de software, podrán cambiar a lo largo del tiempo. Esta podría ser la filosofía a seguir para llegar a ser una empresa centrada en el dato.

4.2.3. Data quantity vs. Data quality

Estamos en la era del Big Data. Durante los últimos años y de forma diaria, se recopila una cantidad inmensa de datos con múltiples orígenes. Toda esta información es tratada por las herramientas y sistemas AI/ML que deben extraer el valor de dichos datos. Sin embargo, estos mecanismos no son capaces de distinguir los datos válidos de los no válidos. Estos datos no válidos podrían llevar a resultados erróneos y lo que es peor, esos resultados se podrían utilizar para tomar decisiones, con lo cual, el resultado final podría ser catastrófico.

En general, si se dispone de más datos, estos conducirán a unos resultados mejores. Pero esta idea puede llevar a las empresas a decisiones erróneas. Actualmente, mientras que el coste de almacenamiento de los datos se ha reducido, la potencia de procesamiento ha aumentado, con lo cual, las empresas podrían optar por almacenar más y más datos para alimentar los sistemas AI/ML. Pero estos sistemas llegan a un punto en el cual se pueden saturar ante la entrada masiva de datos. Cuando esto ocurra, es posible que haya llegado el momento de replantearse la situación. Quizá sea recomendable reducir la **cantidad** de datos y optar por aumentar y mejorar la **calidad** de los datos de entrada al sistema. Una pequeña mejora en la calidad de los datos, puede suponer un rendimiento más óptimo de los sistemas AI/ML.

En resumen, se puede afirmar que tener una gran cantidad de datos es una ventaja, no una necesidad. Pero se puede lograr más con menos datos siempre que la calidad sea mayor. Existen varios aspectos [33] que determinan la calidad de los datos:

- **Integridad**. Indica la falta de cambios en la información entre dos actualizaciones. Sería el término opuesto a corrupción de datos.
- **Singularidad**. Indica si los datos están duplicados dentro del conjunto de datos.
- **Coherencia**. Indica si existen similitudes entre datos que representan los mismos elementos en base a unos requisitos.
- **Precisión**. Indica si se describe correctamente el objeto y sus atributos o características.
- **Validez**. Indica si los datos se encuentran en el formato adecuado. Los datos no válidos, como los datos inexactos, pueden conducir a resultados defectuosos.
- **Puntualidad**. Indica si los datos están actualizados y se reciben en el momento esperado. De esta forma, la información se utilizará de forma eficiente.

4.2.4. La importancia del dataset

La palabra **Dataset** significa, literalmente, *conjunto de datos*. En cualquier proyecto de AI, el primer paso será conseguir un buen conjunto de datos. No cualquier conjunto de datos, sino uno cuyos datos tengan calidad. Esto es muy importante porque los algoritmos de los sistemas AI/ML dependen en gran medida de la calidad de los datos. Por lo tanto, si los datos son erróneos o contienen información errónea, las conclusiones y, por supuesto, las decisiones que se tomen, también serán erróneas.

Antes de abordar las posibles fuentes de estos conjuntos de datos, sería conveniente conocer los *tipos de datos* [34] que se pueden encontrar en un dataset. Se pueden clasificar en base a dos aspectos diferentes:

1. **Tipos de datos según su Origen**. Existen cinco posibles fuentes de datos:
   - *Big Transaction Data*. Son datos que se generan en las grandes transacciones como, por ejemplo, facturación o comunicación (llamadas, mensajería, pagos con tarjeta).
   - *Web y RRSS*[5]. Son datos que generan los usuarios en sus interacciones con la web o sus redes sociales (búsquedas en navegadores, Meta, Twiter).
   - *Datos Biométricos*. Son datos que aportan las personas a partir de sus características fisiológicas (reconocimiento facial, huella dactilar).
   - *Datos M2M*[6]. Son datos que se obtienen a través de las tecnologías que facilitan la comunicación entre dispositivos (señales GPS u otros sensores).
   - *Datos generados por seres humanos*. Son datos que se generan por la interacción de las personas (emails, grabaciones CAT[7]).
2. **Tipos de datos según su Estructura**. En este caso se refiere a la estructura del dato:
   - *Datos Estructurados*. Este tipo de dato está organizado, ordenado, tiene estructura. Y por ello, es más sencillo de procesar, como los datos tabulares o las bases de datos.
   - *Datos No Estructurados*. Este tipo de dato no tiene una estructura interna, con lo cual, para procesarlo será necesario identificarlo, ordenarlo y almacenarlo, como las imágenes, el texto o el sonido.

[5] Redes Sociales.
[6] Machine To Machine.
[7] Centro de Atención al Cliente.

- *Datos Semiestructurados*. Este tipo de dato es una mezcla de los anteriores porque no tiene una estructura clara pero sí tiene una organización más o menos definida, como las páginas web o los formatos XML[8] y JSON[9].

Una vez aclarados los tipos de datos que se pueden encontrar en un dataset y su clasificación, es necesario encontrar fuentes de conjuntos de datos fiables, puesto que es uno de los aspectos más importantes de cualquier proyecto. Existen conjuntos de datos públicos asociados a benchmarks de investigación en aplicaciones concretas, conjuntos de datos generados en el ámbito académico-científico y conjuntos de datos privados generados con tecnología y/o medios propios de la empresas. Todas estas fuentes tienen puntos fuertes, puntos débiles y algunas están especializadas en tipologías concretas de datos. En definitiva, existirán muchas fuentes de datos posibles, pero es recomendable centrarse en aquellas que tengan un cierto grado de calidad en los datos que incluyen. A continuación se enumeran algunas fuentes concretas muy conocidas:

1. **Google Dataset Search**. Dispone del conjunto de datos más completo y también el más amplio. Al tratarse de un producto de Google, tiene un motor de búsqueda muy potente pero también puede realizar un filtrado muy específico.

2. **Kaggle**. Dispone de conjuntos de datos que se pueden usar para aprender AI/ML a través de las competiciones que organiza. También publica los conjuntos de datos junto a sus características. Es un sitio muy importante para la comunidad de ciencia de datos.

3. **Data.gov**. Dispone de conjuntos de datos públicos clasificados por diferentes temas. Está promovido por el Gobierno de los EE.UU. para potenciar la investigación y el desarrollo en las comunidades de ciencia de datos.

4. **KITTI**[10] **vision benchmark**. Dispone del conjunto de datos más popular para su uso en robótica móvil y conducción autónoma. Se trata de horas de grabaciones de escenarios de tráfico grabados con una variedad de modalidades de sensores, que incluyen RGB de alta resolución, cámaras estéreo en escala de grises y un escáner láser 3D.

[8] eXtensible Markup Language
[9] JavaScript Object Notation
[10] Karlsruhe Institute of Technology and Toyota Technological Institute.

Una vez encontrada dicha fuente, es necesario realizar determinados procesos de limpieza y transformación para aumentar la calidad de los datos. Estas tareas ya no forman parte del conjunto de datos como tal, sino que son transformaciones que se llevan a cabo para depurar algunos de los defectos más comunes que se pueden encontrar en el conjunto de datos y así mejorar la calidad del dato. Algunos de estos defectos son los siguientes:

- **Ausencia de valores**. En el conjunto de datos se pueden encontrar valores vacíos que dificultarán el entrenamiento del sistema ML.
- **Inconsistencia de datos**. En el conjunto de datos se pueden encontrar errores de formato o de tipología y será necesario aplicar técnicas de validación.
- **Valores duplicados**. En el conjunto de datos se pueden encontrar datos duplicados que es necesario detectar y eliminar.
- **Outliers**[11]. En el conjunto de datos se pueden encontrar datos anómalos que podrían distorsionar la distribución de los datos.

4.2.5. Ecosistema de herramientas

Hasta ahora se han mencionado una serie de conceptos clave en torno a los sistemas AI/ML. Algunos existen desde los inicios pero otros se han ido modelando con el avance y el desarrollo de diferentes ideas:

- Los datos son el nuevo petróleo.
- Centrado en datos vs. Centrado en el modelo.
- Impacto del dato en el rendimiento del modelo.
- Calidad del dato vs. Cantidad de datos.

Pero en realidad, se trata de un solo concepto. Lo más importante es **el dato y su calidad**, y por ello es necesario prestar más atención a este aspecto, mucha más de la que ha recibido hasta ahora. Sin embargo, los datos, una vez recopilados, no se utilizan en bruto. Desde que se recogen los datos hasta que se aplican a un modelo ML pasan por una serie de procesos y transformaciones que permiten depurarlos y que sean útiles. Estos procesos, los relacionados con la recopilación y el etiquetado de los datos, no siguen un estándar y eso afecta a los sistemas ML en general y a su rendimiento en particular.

[11] Valores atípicos.

Para conseguir ese impulso que relance de forma definitiva las iniciativas centradas en el dato es necesario disponer de un ecosistema de herramientas de sistemas ML centradas en datos. Se pueden agrupar estas herramientas en base a la siguiente clasificación [35]:

1. **Etiquetado de datos internos**. El etiquetado de datos ocupa una gran parte del tiempo del proceso de tratamiento de los datos. Habitualmente se cuenta con un equipo de personas que se encargan del etiquetado de los datos, fundamentalmente los no estructurados, como imágenes o texto. Los etiquetadores deben anotar los datos y garantizar que están listos para ser utilizados por el modelo. Coordinando este equipo es necesaria la figura de un ingeniero de datos que tenga una visión más amplia del uso de estos datos garantizando la validez de las etiquetas asignadas y asegurando que se utiliza un criterio en el etiquetado. Con lo cual, para realizar una tarea tan ardua, parece viable el uso de herramientas que faciliten dicho trabajo:
   - *Watchful*[12]. Esta herramienta automatiza el proceso de etiquetado de datos en el flujo de trabajo del sistema ML. Aporta diferentes soluciones por rol (científicos de datos, managers) o por industria (salud, comercio electrónico, finanzas, telecomunicaciones).
   - *Heartex*[13]. Esta empresa ofrece la herramienta LabelStudio en tres versiones diferentes: Community, Team y Enterprise. LabelStudio Community es una herramienta de código abierto para el etiquetado de datos, mientras que Team y Enterprise agregan mayores capacidades y facilidades para el etiquetado.
   - *Dataloop*[14]. Esta empresa agiliza el proceso de preparación de datos visuales para sistemas ML. La gestión no se centra en el etiquetado sino que ofrece un solución extremo a extremo. También ofrece soluciones para diferentes industrias.
2. **Tratamiento de datos externos**. No solo es necesario etiquetar los datos internos. Puede que para enriquecer el conjunto de datos de una empresa sea necesario utilizar fuentes de datos públicas. La dificultad radica en encontrar un conjunto de datos que sea relevante, que aporte valor y que no genere riesgos.

[12] https://watchful.ai/
[13] https://heartex.com/
[14] https://dataloop.ai/

Aquí también surgen diferentes herramientas que permiten descubrir esos datos relevantes y aumentar el abanico de datos de la empresa:

- *Explorium*[15]. Esta empresa proporciona herramientas que permiten eliminar las barreras para adquirir grandes conjuntos de datos de las fuentes de datos externas correctas e integrarlos en sus análisis y modelos predictivos para mejorar la precisión. Explorium facilita la conexión a una amplia gama de fuentes de datos externas, públicas y de alta calidad a través de una plataforma de administración de datos. Ofrece diferentes soluciones por roles, casos de uso e industrias.
- *Fidap*[16]. Esta empresa ofrece una app orientada al sector financiero que permite consultar datos de acciones que facilitarán al usuario la toma de decisiones en el mercado de inversión.

3. **Datos sintéticos**. Existe una tendencia, la creación y el uso de datos sintéticos, que ya se ha usado bastante pero a pequeña escala y actualmente está tomando más fuerza. Los datos sintéticos se generan a partir de simulaciones, a través de algoritmos, por ejemplo, y se trata de una alternativa al uso de datos reales. Son datos artificiales, pero al ser creados a través de simulaciones, son tan válidos como los datos reales para entrenar modelos de sistemas AI/ML. Un caso de uso podría ser el vehículo autónomo, para simular múltiples situaciones que son inviables o arriesgadas en el mundo real. Se trata de una alternativa más económica al uso de datos reales y también muy útil si, por ejemplo, se trata de situaciones en las que sea necesario preservar la privacidad. A pesar de tratarse de una iniciativa novedosa, ya existen algunas opciones para generar este tipo de datos:

   - *Datomize*[17]. Esta empresa proporciona una herramienta con un enfoque centrado en los datos para los sistemas ML generando datos de entrenamiento óptimos que aumentan la cantidad, reducen la cantidad de errores y equilibran los datos. De esta forma, esta herramienta genera unos datos sintéticos óptimos con un sesgo muy bajo. Esta solución permite sintetizar, aumentar, etiquetar de forma automática y realizar una simulación completa para cada modelo ML.

[15] https://www.explorium.ai/
[16] https://www.fidap.com/
[17] https://www.datomize.com/

- *Synthesis AI*[18]. Esta empresa ofrece una herramienta que permite el acceso a una plataforma de datos sintéticos a través de *APIs*. Estas APIs devuelven, bajo demanda, imágenes diversas y perfectamente etiquetadas a una gran velocidad puesto que se basa en una plataforma de generación en la nube y altamente escalable. Su oferta abarca los servicios de teleconferencia, la verificación de identidad y la monitorización de la conducción. Esta empresa fue destacada en la lista de empresas más innovadoras del mundo en 2022.

4. **Observabilidad de datos**. Este concepto se refiere a la capacidad de una empresa para analizar y evaluar la calidad de los datos que manejan sus modelos ML y los posibles problemas asociados. Si esta idea se aplica en la fase inicial del ciclo de vida de los datos, durante la recopilación, será más fácil garantizar datos útiles y de calidad, lo cual redundará en mayor productividad y clientes más satisfechos. Asociados a este concepto también existen diferentes alternativas en el mercado actual:

   - *Monte Carlo*[19]. Esta empresa ofrece una plataforma de observación de los datos que permite que aumente la confianza de los equipos en sus datos dado que elimina el tiempo de inactividad de dichos datos. Monte Carlo emplea sistemas ML para aprender cómo se ven los datos, identifica su tiempo de inactividad y evalúa el impacto. Además, permite esta observación sin extraer los datos de su entorno. Es la única solución que obtiene una certificación *SOC*[20].
   - *Anomalo*[21]. Esta solución detecta automáticamente los problemas de los datos y analiza sus causas raíz. A través de sistemas ML aprende la estructura y los patrones de los datos, detecta los problemas e intenta encontrar la causa raíz ofreciendo además una implementación *In-VPC* [22] que garantiza la máxima privacidad y seguridad de los datos.
   - *Soda*[23]. Esta empresa proporciona una plataforma de monitorización de datos que permite definir a los equipos cómo son los datos óptimos y resolver problemas antes de soportar un impacto posterior. Permite mejorar los flujos de trabajo de extremo a extremo, agilizar las resolución de incidencias, disminuir el tiempo de inactividad de los datos e integrar otras herramientas que esté utilizando el equipo de datos. Existe una herramienta de código abierto para desarrolladores, para después escalar a la opción Team o Enterprise.

[18] https://synthesis.ai/
[19] https://www.montecarlodata.com/
[20] Service Organization Control. Es un estándar de aseguramiento global para reportar sobre los controles de una organización que presta servicios a terceros.
[21] https://www.anomalo.com/
[22] Virtual Private Cloud.
[23] https://www.soda.io/

4.3. MLOps

4.3.1. Definición de MLOps

MLOps es una disciplina muy reciente cuyo objetivo es intentar que el desarrollo y la implementación de sistemas ML sean más sistemáticos y confiables. Una de las tareas más importantes de *MLOps* es asegurar que los datos de muy alta calidad estén disponibles en todas las fases del ciclo de vida de un proyecto ML. Con lo cual, *MLOps* será un aliado para la tendencia *Data Centric*, tal y como se explicó en el apartado anterior (ver Sección 4.2). También se define *MLOps* como una extensión de la metodología *DevOps* al incluir los diferentes procesos de ML y ciencia de datos en el ciclo del desarrollo y operaciones para conseguir un desarrollo del sistema ML más productivo.

El desarrollo de este área será fundamental para fomentar el uso de modelos ML centrados en el dato, además de ser un campo para el desarrollo de nuevos trabajos de los diferentes roles inmersos en los sistemas AI/ML. Se espera que el desarrollo de *MLOps* genere líneas de especialización en áreas específicas como el transporte, la robótica o la medicina. Las soluciones que se especializan en una rama siempre ofrecen mejor rendimiento que las soluciones generalistas.

La Figura 4.3 muestra de forma muy clara las áreas de conocimiento que engloba *MLOps*:

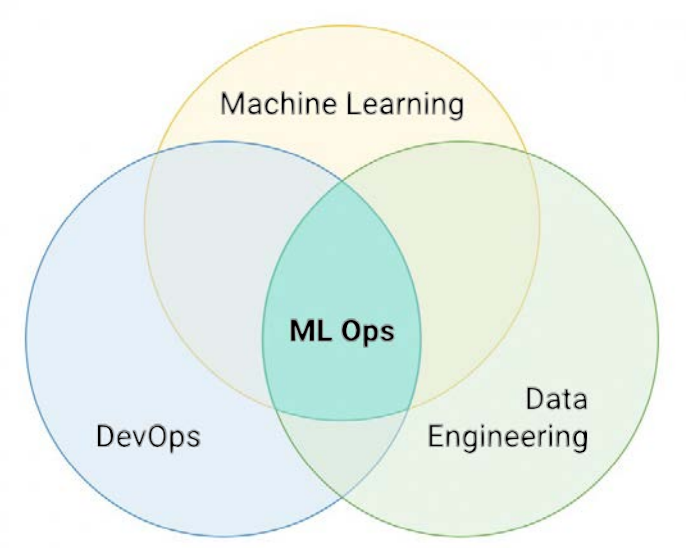

Figura 4.3: MLOps. Fuente: [36]

4.3.2. Principios de MLOps

Existen una serie de principios o prácticas recomendadas que, si son seguidas por las empresas, pueden permitir alcanzar el nivel de desarrollo de *MLOps* necesario para obtener los beneficios derivados de su aplicación. De forma resumida, *MLOps* debe ser colaborativo, continuo, reproducible, automatizado y validado. A continuación se analizan cada uno de estos principios [37]:

- **Colaborativo**. Para alcanzar el éxito en una implementación de *MLOps*, es necesario que personas con diferentes perfiles trabajen y *colaboren* en el mismo equipo. Se trata de crear equipos híbridos y colaborativos que pueden estar formados por científicos de datos, ingenieros ML y profesionales de operaciones de TI. Todos estos perfiles deben trabajar de forma transparente durante todo el proceso de creación del modelo ML, desde la etapa inicial de los datos hasta la monitorización del modelo desplegado en producción, para garantizar que cada uno de los miembros del equipo conoce los detalles de cada una de las etapas del proceso.
- **Continuo**. En la implementación de modelos ML se desarrolla el concepto de *pipeline*, una secuencia de acciones que se aplican a los datos entre el destino y el origen. Gracias a estos procesos *continuos*, se obtienen muchos beneficios en la gestión de operaciones, la reutilización de código o la escalabilidad. Las prácticas más habituales, algunas de las cuales tienen su origen en la metodología *DevOps*, son las siguientes:
  - *Integración Continua*[24]. Esta práctica se refiere, no solo a probar el código y los componentes software, sino también los datos, los esquemas de datos y los modelos.
  - *Entrega Continua*[25]. Esta práctica se refiere, no solo a la entrega de un servicio, sino a un sistema que debe implementar otro servicio de forma automática.
  - *Entrenamiento Continuo*[26]. Esta práctica es exclusiva de los modelos ML, dado que se vuelve a entrenar de forma automática el modelo para construirlo de nuevo.
  - *Monitorización Continua*[27]. Esta práctica se refiere a la monitorización de los datos de producción y los indicadores de rendimiento del modelo.

[24] CI, Continuous Integration.
[25] CD, Continuous Delivery.
[26] CT, Continuous Training.
[27] CM, Continuous Monitoring.

- **Reproducible**. Para que se pueda *reproducir* un modelo ML es fundamental el control de versiones. Pero no solo el control de versiones relativo al código, como ocurre en los modelos de desarrollo de software tradicional, sino también el control de versiones relativo al modelo, a los datos de entrenamiento o a los hiperparámetros del modelo. De tal forma que, dada la misma entrada de datos, todas y cada una de las fases del desarrollo de un modelo ML, procesamiento de datos, entrenamiento del modelo e implementación del modelo, producen resultados similares.

- **Automatizado**. Esta práctica es el objetivo final de cualquier equipo que implemente un modelo ML, *automatizar* todos los pasos de un flujo de trabajo ML para conseguir que no haya intervención manual. La automatización es fundamental para reducir, no solo tiempo, sino también costes. Y además, las pruebas automatizadas también ayudarían a encontrar problemas en las etapas iniciales de desarrollo del modelo.

- **Validado**. Esta práctica debe asegurarse a través de las *pruebas* y la *monitorización* del sistema. Las pruebas se podrían realizar en los tres niveles del sistema ML: pruebas de datos, pruebas del modelo ML y pruebas de la infraestructura ML. Por otro lado, es necesario supervisar el modelo ML para garantizar que funciona tal y como se espera. Habrá que monitorizar el rendimiento de las predicciones del modelo pero también otras métricas estándar. En cualquier caso, dado que podrían existir problemas que afectaran a partes concretas del modelo, sería necesario monitorizar el rendimiento, no solo de forma global, sino también de forma particular en estas partes concretas del modelo.

4.3.3. Etapas MLOps

El desarrollo e implementación de un sistema *MLOps* tiene diferentes etapas. En la Figura 4.4 se muestran las etapas de un proyecto *MLOps*:

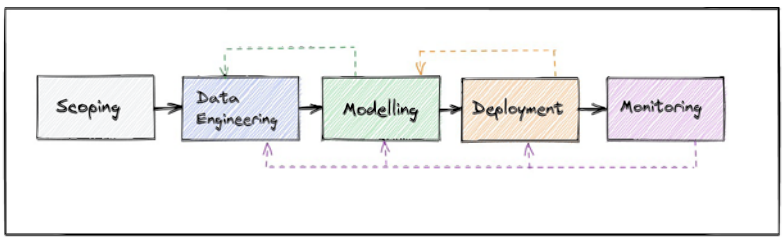

Figura 4.4: Etapas MLOps. Fuente: [38]

A continuación se explican las etapas de dicho proceso:

1. **Alcance**. En esta etapa se define el proyecto. Se intentan encontrar las soluciones a un determinado problema, se analizan los requisitos y la disponibilidad de los datos. También es necesario verificar si para resolver el problema planteado se requiere de un sistema ML. Una vez planteadas las posibles soluciones también hay que evaluar la viabilidad de cada una de las soluciones. En resumen, esta fase ayuda a los equipos de trabajo a tener una idea más clara sobre el problema en cuestión.

2. **Ingeniería de Datos**. En esta etapa se realiza todo el trabajo sobre los datos. Existen diferentes subetapas a tener en cuenta como la recopilación, la limpieza, el formato, el etiquetado y la organización de los datos. Es una etapa fundamental, puesto que, como se ha comentado, el dato es fundamental dentro del modelo ML.

3. **Modelado**. En esta etapa se realiza el trabajo sobre el modelo ML, que podría considerarse más importante que la fase de Ingeniería de Datos, pero en realidad no es así. Es una fase importante, pero el enfoque debe estar en el dato. Para establecer el modelo adecuado se pueden seguir estos criterios:

   - *Línea Base*. Se puede comenzar realizando un estudio sobre modelos similares ya existentes y a partir de este estudio, ya se puede tener una idea inicial sobre cómo proceder para llegar a la solución.
   - *Centrado en Datos*. Es la idea desarrollada en la sección anterior (ver Sección 4.2). Si se hace un esfuerzo para obtener datos de calidad, mejorarán los resultados del modelo que se implemente.
   - *Control de Versiones*. El control de versiones permite volver a una versión de trabajo anterior ante ciertos errores. En este caso, el versionado afecta no solo al código sino también a los hiperparámetros del modelo o a los conjuntos de datos.

4. **Despliegue**. En esta etapa también existen diferentes alternativas, es decir, no se despliega en el entorno productivo y directamente se pone en funcionamiento. Según se trate o no de un producto nuevo o de alguna mejora sobre un producto existente, se pueden plantear algunas opciones de despliegue:

   - *Shadow*. También denominada implementación en la sombra. En esta opción se despliega el modelo, pero al final, la decisión, independientemente de lo que indique la predicción del modelo, la toma una persona. Es una forma de medir cómo está funcionando el modelo.

- *Canary*. También denominada implementación canaria. En esta opción se despliega el modelo, pero solo está expuesto a una pequeña parte de los datos. De tal forma que, según los valores de rendimiento del modelo, se puede aumentar de forma gradual el tráfico de datos o se puede retirar el modelo para corregir los errores detectados y ajustarlo.
- *Blue Green*. También denominada implementación azul-verde. En esta opción se trabaja con dos versiones: la azul sería la versión antigua del modelo y la verde sería la versión nueva del modelo. El tráfico de datos se va transfiriendo de la versión azul a la verde cuando ambas se encuentran en producción. Una vez hecha la transferencia, se puede conservar la versión azul, por si fuera necesario restaurar la versión.

5. **Monitorización**. En esta etapa se monitoriza el rendimiento del modelo ML. El proceso de implementación de un sistema ML, como ya se ha comentado, es iterativo. Y la iteración se basa en la supervisión del modelo una vez implementado. Cualquier modelo o sistema tiene un margen de mejora pero solo mediante un proceso adecuado de supervisión será posible alcanzar dicha mejora. Para ello será necesario tener en cuenta los siguientes aspectos:
   - *Supervisión del pipeline ML*.
   - *Posibles cambios en los datos de entrada*.
   - *Identificar las métricas adecuadas*.

4.3.4. MLOps vs. DevOps

Antes de establecer las diferencias entre *MLOps* y *DevOps*, puede que sea conveniente aclarar el concepto *DevOps*. El término *DevOps* aglutina los conceptos Desarrollo (*Development*, Dev, en inglés) y Operaciones (*Operations*, Ops, en inglés). Esta metodología pretende agilizar el desarrollo de software, desplegarlo y monitorizarlo, todo ello de manera iterativa. Permite un trabajo colaborativo entre los equipos de desarrollo y operaciones con el fin de lanzar funcionalidades más rápido. Esto, sin duda, aumenta, no solo la satisfacción del cliente, sino también de los propios equipos de desarrollo.

Una vez aclarado el concepto *DevOps*, será más sencillo entender el concepto *MLOps* como una extensión del anterior, tal y como se mencionó en el inicio de esta sección (ver Sección 4.3.1). El término *MLOps* aglutina los conceptos Aprendizaje Automático (*Machine Learning*, ML, en inglés) y Operaciones (*Operations*, Ops, en inglés).

De forma análoga, pretende agilizar el desarrollo de los modelos ML, validarlos y monitorizarlos, todo ello de manera iterativa, permitiendo, en este caso, un trabajo colaborativo entre los equipos de ciencia de datos y operaciones. El objetivo es crear soluciones de ML cuyo rendimiento alcance un determinado umbral. En la Figura 4.5 se muestra, de forma sencilla, un ciclo *MLOps*:

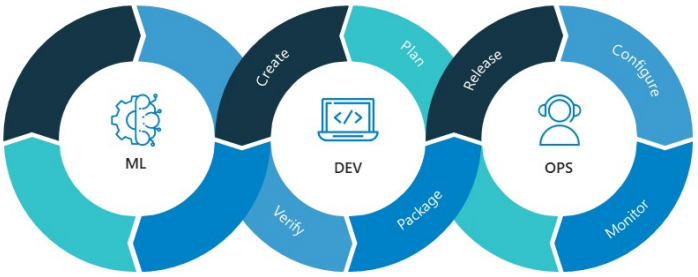

Figura 4.5: Ciclo MLOps. Fuente: [39]

Aunque, de algún modo, una metodología deriva de la otra, y por lo tanto, tienen ciertas similitudes, también existen diferencias [40]. Se enumeran a continuación:

1. **Equipo**. En *DevOps*, los equipos están formados por desarrolladores de software, que son los encargados de crear la aplicación. En *MLOps*, los equipos están formados por científicos de datos que se encargan de escribir el código de los modelos y por ingenieros ML, encargados de desplegar e integrar el modelo como parte de un sistema mayor.

2. **Desarrollo**. En *DevOps*, el código será la base de una aplicación que se implementará y validará con unas pruebas. En *MLOps*, el objetivo final es construir y entrenar un modelo que posteriormente recibirá unos datos de entrada, por lo tanto tiene un carácter más experimental. Otro aspecto a tener en cuenta en el desarrollo es la degradación. Las aplicaciones desarrolladas con metodología *DevOps*, en general, no se degradan y es relativamente sencillo agregar nueva funcionalidad. Mientras que las aplicaciones desarrolladas con metodología *MLOps* sí están expuestas a la degradación porque con el paso del tiempo pueden cambiar los datos de entrada al modelo y eso puede provocar un funcionamiento erróneo.

3. **Control de Versiones**. En *DevOps*, el control de versiones implica únicamente el control de cambios en el código. En *MLOps*, dado que se trata de un ciclo iterativo, es necesario el control en cada ciclo, pero no solo del código del modelo ML, sino también de los hiperparámetros, las métricas que controlan el rendimiento del modelo y los datos utilizados en el entrenamiento.

4. **Pruebas**. En *DevOps*, las pruebas que se ejecutan son las unitarias y las de integración. En *MLOps*, además de las pruebas del modelo, son necesarias pruebas para validar el modelo, los datos y el modelo entrenado.

5. **Pipelines**. En *DevOps*, solo se requiere que el *pipeline* incluya integración continua (CI) y entrega continua (CD). En *MLOps*, aunque también se incluyen estos *pipelines*, se ejecutan de forma diferente. La integración continua no solo valida código y componentes, como ocurre en *DevOps*, sino que también prueba y valida datos y modelos. Por otro lado, la entrega continua no se refiere solo a un paquete de software sino a un sistema o modelo ML. Pero además, *MLOps* incluye otro elemento en el *pipeline*, entrenamiento continuo (CT), que se encarga del reentrenamiento del modelo ML.

En la Tabla 4.2 se resumen las principales diferencias entre *DevOps* y *MLOps*:

	DevOps	**MLOps**
Roles	Ingenieros de Software, Ingenieros *DevOps*	Científicos de Datos, Ingenieros ML
Desarrollo	Desarrollo lineal con objetivo conocido	Desarrollo experimental por naturaleza
Pruebas	Unitarias e Integración	Validación del modelo, del modelo entrenado y de los datos
Despliegue	Build inicial y releases *CI/CD*	Entrenamiento continuo del modelo
Producción	Monitorizar en busca de errores	Monitorizar perfiles de datos, generar alertas

Tabla 4.2: DevOps vs. MLOps

4.3.5. Implementación MLOps

Como se comentó en un apartado anterior (ver Sección 4.3.2), uno de los principios de *MLOps* es la **automatización**. Es el objetivo final de cualquier equipo que implemente un modelo ML. La automatización de todos los pasos de un flujo de trabajo reducirá la intervención manual, el tiempo de ejecución y también los costes.

Sin embargo, no resulta sencillo implementar la automatización de todos los pasos de un flujo de trabajo o *pipeline*. Justamente, ese nivel de automatización es lo que define el nivel de madurez del modelo [41]. Por ejemplo, si se consigue automatizar una etapa del proceso, el modelo será maduro en un nivel y así sucesivamente.

Habitualmente se considera que el nivel inicial o nivel cero no implica automatización. Según se vayan alcanzando los hitos de automatización de las diferentes etapas se adquirirá un nuevo nivel en la escala de madurez del modelo. Según el modelo a seguir, existirán *n* niveles de madurez y cada nivel tendrá una definición de la automatización de dicho nivel.

De forma genérica se definen tres niveles y cada uno de ellos se clasificará en base a los siguientes parámetros:

- **Equipos involucrados**. Este parámetro se refiere, no solo a los perfiles de las personas que componen los equipos que trabajan en la automatización del modelo ML, sino también a cómo se relacionan entre ellos y su grado de interacción.
- **Estrategia de Recopilación de datos**. Este parámetro se refiere a la forma en la que se recogen los datos pero también a cómo se almacenan y las diferentes fases por las que transcurren hasta que están listos para ser usados.
- **Implementación del Modelo**. Este parámetro se refiere tanto a lo relativo al desarrollo del modelo como a la liberación de las versiones de dicho modelo.
- **CI/CD**. Este parámetro se refiere a la integración y el despliegue continuo. Estos conceptos se explicaron en una sección anterior (ver Sección 4.3.2) y su implementación no se alcanza de forma plena hasta llegar al mayor nivel de madurez del modelo.
- **Integración y Pruebas de Aplicaciones**. Este parámetro se refiere a cómo se integran las diferentes partes del modelo y también a las pruebas que se realizan. En los niveles más bajos de madurez depende de la experiencia del equipo de trabajo.

- **Monitorización**. Este parámetro se refiere a la supervisión del sistema para evitar la degradación del rendimiento del modelo u otras variaciones en su comportamiento.

Como ya se comentó en el párrafo anterior, se definirán tres niveles y cada uno de ellos se categorizará en función de los parámetros que se comentaron previamente.

4.3.5.1. MLOps Level 0

El nivel 0 de *MLOps* es el nivel más básico o inicial de madurez. En realidad, en este punto se considera que todas las actividades se realizan de forma manual, con lo cual, también se conoce como *sin MLOps*. En resumen, en este nivel, las tareas podrían ser similares a las que se realizan en un ciclo de desarrollo de modelo ML tradicional.

Según los parámetros ya mencionados, sus características son las siguientes:

- *Equipos involucrados*. Los equipos formados por científicos de datos e ingenieros ML trabajan de forma aislada.
- *Estrategia de Recopilación de datos*. No existe el concepto de la importancia del dato. Los datos existen pero están dispersos, no se explotan de forma adecuada.
- *Implementación del Modelo*. Estas tareas se realizan de forma manual y están basadas en scripts.
- *CI/CD*. En este nivel es inexistente.
- *Integración y Pruebas de Aplicaciones*. Este punto queda supeditado a la experiencia de las personas que forman los equipos de trabajo.
- *Monitorización*. No se realiza un seguimiento del rendimiento del modelo en producción.

4.3.5.2. MLOps Level 1

El nivel 1 de *MLOps* es el nivel intermedio de madurez. Su objetivo es conseguir un entrenamiento continuo de los modelos ML y para ello los *pipeline* deben estar automatizados. Estos *pipelines* incluyen la automatización del proceso de capacitación con nuevos datos, el reentrenamiento del modelo y la automatización de la validación tanto de los datos como del modelo.

Según los parámetros ya mencionados, sus características son las siguientes:

- *Equipos involucrados*. El nivel de comunicación entre los equipos de trabajo involucrados en el desarrollo del modelo ML aumenta. Ya no trabajan de forma aislada.
- *Estrategia de Recopilación de datos*. Los datos se almacenan y gestionan de forma organizada e incluso se automatizan algunas tareas relativas a la recopilación o el análisis de los datos.
- *Implementación del Modelo*. La fase de desarrollo se automatiza y también se comienza a automatizar el proceso de entrega.
- *CI/CD*. En este nivel se logra la entrega continua de modelos pero no la integración continua.
- *Integración y Pruebas de Aplicaciones*. Se ejecutan pruebas con cada lanzamiento y en este nivel no existe tanta dependencia de la experiencia de los equipos de trabajo.
- *Monitorización*. Se realizan tareas de supervisión e incluso se implementan alertas que podría provocar el reentrenamiento del modelo.

4.3.5.3. MLOps Level 2

El nivel 2 de *MLOps*, partiendo de la automatización de los *pipelines* del nivel anterior, trata de proporcionar actualizaciones rápidas y fiables para dichos *pipelines* mediante la automatización de la integración y la entrega continua, *CI/CD*. Se trata del nivel más alto de automatización y, por lo tanto, el objetivo final de cualquier modelo ML.

Según los parámetros ya mencionados, sus características son las siguientes:

- *Equipos involucrados*. En este nivel, solo una confirmación en el código puede desencadenar un conjunto de actividades automatizadas debido al alto grado de interacción de los equipos formados por científicos de datos e ingenieros ML.
- *Estrategia de Recopilación de datos*. Existen data lakes en los que los datos se almacenan correctamente y existen *pipelines* automatizados para todas las tareas relativas al dato. Ahora sí existe un control estricto y completo del dato.

- *Implementación del Modelo*. En este nivel se buscan otro tipo de soluciones alejadas de la creación y mantenimiento de colecciones de scripts.
- *CI/CD*. En este nivel se introduce de forma completa la integración y la entrega continua.
- *Integración y Pruebas de Aplicaciones*. Se ejecutan pruebas automatizadas para cada versión del modelo.
- *Monitorización*. Existe una supervisión continua que recopila las estadísticas del rendimiento del modelo. Esto permite evitar desvíos o deterioros del modelo aplicando medidas correctoras de forma inmediata.

4.3.5.4. Otros modelos de madurez MLOps

Aunque el modelo de niveles de madurez explicado anteriormente sería el estándar, no existe un modelo universal. Tanto Google como Microsoft han definido sus propios modelos de niveles de madurez:

1. **Google Cloud**. Su definición del modelo de madurez *MLOps* se basa en tres niveles:
   - *Proceso Manual*.
   - *Automatización de la canalización de ML*.
   - *Automatización de la canalización de CI/CD*.
2. **Microsoft Azure**. Su definición del modelo de madurez *MLOps* se basa en cinco niveles:
   - *Sin MLOps*.
   - *DevOps pero sin MLOps*.
   - *Entrenamiento Automatizado*.
   - *Despliegue de Modelo Automatizado*.
   - *Operaciones Automatizadas completas de MLOps*.

En la Figura 4.6 se visualizan los modelos de los niveles de madurez de estas dos empresas:

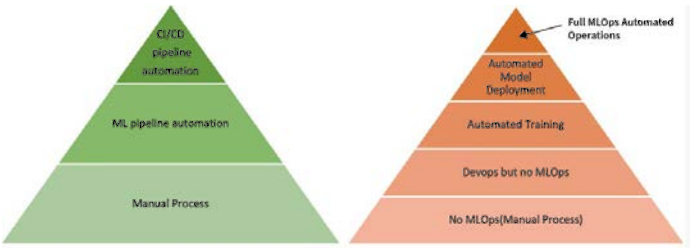

Figura 4.6: Modelo de Madurez de Google vs. Modelo de Madurez de Microsoft. Fuente: [42]

4.3.6. Herramientas MLOps

Existen herramientas *MLOps* que pueden realizar las diferentes tareas que debe ejecutar el equipo que desarrolla un modelo ML, pero en general, se pueden dividir en dos grandes grupos: las **soluciones Custom** y las **soluciones E2E**. A partir de esta gran división también se podría hacer referencia al origen o proveedor de cada una de estas soluciones *MLOps*. Principalmente existen herramientas de *código abierto* o *herramientas comerciales* gestionadas por las grandes empresas tecnológicas.

4.3.6.1. Soluciones Custom

Las herramientas *MLOps* que se encuadran dentro de las soluciones Personalizadas (*Custom*, en inglés) son, fundamentalmente, de código abierto [43]. Son aquellas que se concentran en una tarea específica dentro del flujo de tareas de *MLOps*. Prácticamente la mitad de las empresas TI utilizan este tipo de herramientas frente a las soluciones comerciales. Estas herramientas, según la tarea del flujo *MLOps* que realizan, se pueden clasificar en base al siguiente esquema:

- **Gestión de Datos**. Estas herramientas se encargan de la gestión de los datos. Dentro de este bloque existen herramientas para el etiquetado y el control de versiones:
  - *Etiquetado*. Son herramientas que se encargan de etiquetar los datos que posteriormente se utilizarán para entrenar los modelos ML, como *Snorkel*[28] o *Doccano*[29].

[28] https://snorkel.ai/
[29] https://github.com/doccano/doccano

- *Control de Versiones*. Son herramientas que permiten administrar diferentes versiones de conjuntos de datos y almacenarlos puesto que de esta forma su identificación será más sencilla, como *Comet*[30] o *LakeFS*[31].

- **Modelado**. Estas herramientas se encargan de diferentes tareas relacionadas con el desarrollo y mejora del modelo ML. Dentro de este bloque existen herramientas para la ingeniería de funciones, el seguimiento de experimentos y la optimización de los hiperparámetros:

  - *Ingeniería de funciones*. Son herramientas que transforman datos no procesados de tal forma que se presenten como una entrada viable para los algoritmos ML, como *AutoFeat*[32] o *Feast*[33].
  - *Seguimiento de Experimentos*. Son herramientas que permiten realizar un seguimiento de todos y cada uno de los pasos que se dan en un experimento de un modelo ML, desde que se preparan los datos hasta la implementación final del modelo ML, como *ModelDB*[34] o *TensorBoard*[35].
  - *Optimización de Hiperparámetros*. Son herramientas que permiten encontrar la mejor combinación de hiperparámetros para un algoritmo ML. La forma más habitual sería realizar varios experimentos con diferentes combinaciones de valores de los hiperparámetros y analizar su rendimiento con alguna métrica, como *Hiperopt*[36] o *Scikit-Optimize*[37].

- **Operacionalización**. Estas herramientas se encargan de tareas relacionadas con la parte operativa del modelo, la relacionada con la puesta en producción o despliegue. Dentro de este bloque existen herramientas para la implementación o puesta en servicios de los modelos y la supervisión de los modelos:

  - *Implementación/Servicio*. Son herramientas que se encargan de empaquetar un modelo ML y sus dependencias para que se pueda ejecutar en producción, como *Kubeflow*[38] o *Torch Serve*[39].

[30] https://www.comet.ml/site/
[31] https://lakefs.io/
[32] https://github.com/cod3licious/autofeat
[33] https://feast.dev/
[34] https://github.com/VertaAI/modeldb
[35] https://www.tensorflow.org/tensorboard?hl=es-419
[36] http://hyperopt.github.io/hyperopt/
[37] https://scikit-optimize.github.io/stable/
[38] https://www.kubeflow.org/
[39] https://pytorch.org/serve/

- *Supervisión de Modelos*. Son herramientas que realizan el seguimiento del rendimiento de un modelo en producción en base a diferentes métricas e identificando posibles problemas con el establecimiento de alertas, como *Evidently AI*[40] o *Arize*[41].

4.3.6.2. Soluciones E2E

Las herramientas *MLOps* que se encuadran dentro de las soluciones Extremo a Extremo (*End To End*, E2E, en inglés) son soluciones comerciales [44]. Son aquellas que ofrecen la posibilidad de crear, entrenar y desplegar los modelos ML. Por lo tanto, son herramientas que abarcan todo el ciclo de trabajo de un modelo ML. Estas herramientas tienen su origen en las grandes empresas tecnológicas. A continuación se enumeran algunas de las más importantes:

- **Amazon SageMaker**[42]. Esta herramienta permite optimizar todas las etapas del ciclo de vida de un modelo ML, automatizando las mejores prácticas de *MLOps*.

- **Databricks**[43]. Esta herramienta, *Machine Learning de Databricks*, se basa en una arquitectura de lago abierto con *Delta Lake*, lo cual implica acceder y explorar cualquier tipo de datos.

- **Cloudera**[44]. Esta herramienta ofrece una característica diferencial, la solución *SDX*. Esta característica proporciona a los usuarios una mayor visibilidad y una gestión guiada para la seguridad de los datos, sobre todo cuando se trabaja con datos confidenciales.

- **Iguazio**[45]. Esta herramienta se distingue por su *gestión de las características*, que son propiedades que se utilizan como entradas para un modelo ML. Generar una característica nueva, llamada ingeniería de características, es un proceso muy costoso. Iguazio dispone de una tienda de características integradas que resuelve estos problemas.

- **MLFlow**[46]. Esta herramienta es una plataforma de código abierto que permite más personalizaciones que muchos de los productos de código cerrado, a través de varios componentes: *seguimiento de flujo ML*, *proyectos de MLFlow*, *modelos de MLFlow* y *registro de modelos*.

[40] https://evidentlyai.com/
[41] https://arize.com/
[42] https://aws.amazon.com/es/sagemaker/mlops/
[43] https://databricks.com/product/machine-learning
[44] https://www.cloudera.com/products/cloudera-data-platform.html
[45] https://www.iguazio.com/
[46] https://mlflow.org/

Estas son algunas de las herramientas más importantes. Como se comentó anteriormente, también las grandes empresas tecnológicas, como Google o Microsoft, tienen sus herramientas:

- **Microsoft Azure**. Dispone de su propia suite compuesta por las siguientes herramientas: *Azure Machine Learning*, *Azure Pipelines*, *Azure Monitor* y *Azure Kubernetes Services*.
- **Google Cloud**. También Google tiene su conjunto de aplicaciones *MLOps* compuesto por las siguientes herramientas: *Dataflow*, *AI Platform Notebook*, *TFX*, *Kubeflow Pipelines* y *Google Kubernetes Engine*.

Capítulo 5

Aplicación de Estrategias y Procesos QA en IDP

5.1. Introducción

Algunos estudios indican que, llegado el año 2025, los datos en todo el mundo superarán los 175 zettabytes. Y este es el mayor desafío al que se enfrentan las empresas hoy en día, puesto que, a pesar de tener una gran cantidad de datos, deben utilizarlos de una manera inteligente para alcanzar el éxito empresarial.

La mayor parte de la información, en torno al 80 % de los datos, está incorporada en formatos digitales no estructurados como documentos, correos electrónicos e imágenes. Más concretamente, ejemplos de datos digitales no estructurados pueden ser los siguientes: tablas, figuras y texto libre renderizados en documentos de cualquier tipo; formularios, facturas y recibos; fotos de dichos documentos en papel. Esto representa un verdadero obstáculo para la transformación digital y la automatización. Tanto los datos no estructurados como los semiestructurados deben convertirse en información estructurada utilizable.

El **Procesamiento Inteligente de Documentos** (*Intelligent Document Processing*, IDP, en inglés) tuvo sus inicios en el desarrollo de las primeras soluciones de **Reconocimiento Óptico de Caracteres** (*Optical Character Recognition*, OCR, en inglés), cuyo alcance fue convertir imágenes de caracteres en texto codificado. Pero la tecnología evolucionó para incorporar otro tipo de capacidades, como el **Procesamiento del Lenguaje Natural** (*Natural Language Processing*, NLP, en inglés). Esto permitió que el procesamiento de documentos avanzara más allá del simple reconocimiento de caracteres y permitió cierto nivel de interpretación y comprensión del texto.

De hecho, las soluciones mencionadas anteriormente se encuadran dentro de sistemas de AI y tienen mucha relación con las áreas **Visión por Computador** (*Computer Vision*, CV, en inglés) y NLP, ambos pilares tecnológicos fundamentales que impulsan IDP. Incluso combinan perfectamente con la **Automatización de Procesos Robóticos** (*Robotic Process Automation*, RPA, en inglés). Con lo cual, conviene diferenciar claramente estos términos para no confundirlos:

- *IDP no es solo OCR*. Si bien IDP incorpora OCR y tecnología de captura de datos, IDP es la tecnología de extracción de datos que supera las limitaciones del OCR tradicional en la extracción de datos de documentos más complejos. Es capaz de interpretar y extraer información semántica y relevante para diversos casos de uso.
- *IDP no es RPA*. Para IDP es esencial comprender el contenido en contexto antes de que se puedan extraer datos para varios procesos posteriores. El objetivo de la RPA es automatizar procesos ya existentes y, por lo tanto, requiere el apoyo de IDP para tomar decisiones sobre el contenido.

IDP automatiza la extracción de datos a partir de cualquier tipo de documento, ya sea escrito a mano, impreso o digital. Aplica la AI y los modelos ML y DL para clasificar, categorizar y extraer información relevante y validar los datos extraídos. Independientemente del tipo de documento que se procese, el objetivo de IDP es extraer información relevante en diversos casos de negocio.

5.2. Etapas

Las tecnologías de captura de documentos agilizan y automatizan el proceso de ingesta, reconocimiento y clasificación de documentos para que se pueda extraer información importante de manera rápida y precisa. Estas tecnologías tienen la capacidad de completar estas tareas con un alto grado de automatización. En realidad, IDP es un enfoque para la transformación digital a través de la automatización.

La Figura 5.1 muestra de forma gráfica las etapas IDP:

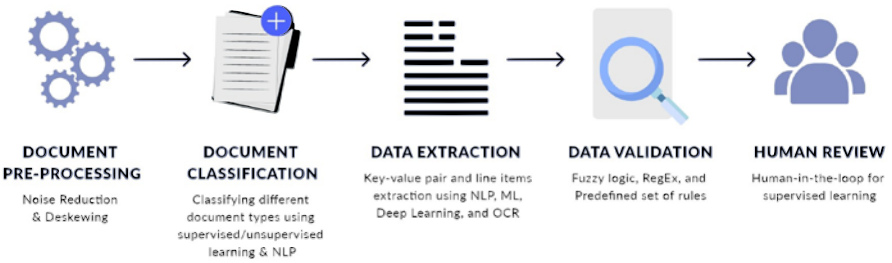

Figura 5.1: Etapas IDP. Fuente: [45]

A continuación se detallan cada una de las etapas [46] que forman el flujo de tareas de IDP.

5.2.1. Recopilación de Documentos

IDP siempre comienza con la ingesta de datos de varias fuentes, tanto digitales como en papel. Para recibir los datos ya digitalizados, la mayoría de las soluciones IDP integran o permiten desarrollar interfaces personalizadas. Si además es necesario recopilar documentos escritos a mano o en papel, la tecnología IDP se integra con hardware, como escáneres, cámaras de fotos o Apps del móvil, que aceleran el proceso de escaneo y la digitalización de los documentos.

Esta etapa cubre la automatización de todo lo relacionado con la adquisición y el almacenamiento de datos no estructurados sin procesar, como escaneos o fotos. Incluye automatización de software y hardware.

5.2.2. Preprocesamiento de Documentos

El objetivo de la siguiente etapa es mejorar la calidad y la precisión de los documentos escaneados o capturados con cámara. Para que esa distinción sea clara y pueda llevarse a cabo, en esta etapa se emplean algunas de las siguientes técnicas básicas:

- *Binarización*. La binarización es la técnica para convertir una imagen en color en píxeles en blanco y negro.

- *Eliminación de Inclinación*. Cuando se escanea un documento la imagen puede no estar completamente alineada y esta situación no es la mejor para la técnica de OCR. Para corregirlo, se emplean técnicas como el *método de perfil de proyección*, que se utiliza principalmente para la segmentación de objetos de texto presentes dentro de documentos de texto o el *método de la transformada de Hough*, que es una herramienta que permite detectar curvas en una imagen.
- *Reducción del Ruido*. El objetivo de esta técnica es deshacerse de los pequeños puntos o parches no deseados para que OCR no confunda estos puntos con caracteres.

El análisis de esta etapa, el preprocesamiento de documentos, es clave para conocer el alcance de la solución IDP respecto al dato. En esta etapa, los datos se convierten a un formato estructurado al que pueden acceder otras aplicaciones. Una vez que los datos se convierten, están disponibles para su uso en toda la empresa. El personal no tiene que buscar información porque está disponible en un depósito centralizado de datos estructurados. Por lo tanto, estaríamos hablando de una solución *Data Centric*.

Partiendo de este enfoque *Data Centric*, se podría plantear el uso de algunas de las herramientas comentadas (ver Sección 4.2.5). En concreto, se podría utilizar alguna herramienta para el etiquetado de los datos, dado que es una tarea que ocupa una gran parte del tiempo del proceso de tratamiento de los datos. También se podría utilizar alguna herramienta relacionada con la observabilidad de los datos, dado que podría ser interesante analizar y evaluar la calidad de los datos que van a manejar los modelos ML que se utilizarán en las etapas posteriores de la solución IDP, para garantizar unos datos útiles y de calidad. E incluso, aunque se disponga de una gran cantidad de datos, también sería interesante gestionar datos sintéticos con alguna aplicación puesto que podría ser una forma efectiva de modelar y generar datos de los que no dispone la empresa.

IDP se puede aplicar a una amplia gama de aplicaciones comerciales en todas las industrias donde los trabajadores continúan procesando documentos manualmente. Los ejemplos incluyen casos de uso horizontales, como el procesamiento de órdenes de compra y facturas, así como casos de uso específicos de la industria, como solicitudes de seguros, información de clientes y solicitudes de préstamos, por nombrar algunos. En este sentido, las industrias centradas en datos son los usuarios finales clave de las soluciones IDP.

5.2.3. Clasificación de Documentos

El éxito de un sistema IDP dependerá en gran medida de cómo se haya realizado la clasificación de los documentos de entrada. Para clasificar esas entradas, se recurre, por ejemplo, a los algoritmos de NLP, a partir del contenido textual, que dividirían los documentos según el tipo de contenido destacado o también a algoritmos de CV, cuando se trata de clasificar imágenes de documentos o escaneos. En cualquier caso, la clasificación se realizará a través de tres pasos:

1. *Identificación del Formato*. El primer paso sería averiguar si el archivo es un documento PDF, JPF, PNG o cualquier otro formato.
2. *Identificación de la Estructura*. En este caso son de gran ayuda las soluciones OCR, puesto que tratarían de diferenciar entre documentos estructurados, semiestructurados y no estructurados. Los documentos estructurados tienen una plantilla y, por lo tanto, serían los primeros en ser identificados. Mientras que los documentos semiestructurados tienen algún tipo de estructura, lo cual implica que podrían contener la misma información en diferentes partes del documento, como por ejemplo, una factura. Los documentos no estructurados, sin embargo, apenas tienen estructura, pero las empresas también necesitan extraer datos de ellos, como ocurre en los contratos.
3. *Identificación del Tipo de Documento*. Si el paso anterior se ha realizado con éxito, será más sencillo identificar el tipo de documento. Es fundamental identificar correctamente un tipo de documento para situarlo en la cola de extracción de datos adecuada.

A continuación se analizan las técnicas que se pueden utilizar para llevar a cabo la clasificación de documentos:

- **Aprendizaje Supervisado**. En esta técnica, el sistema aprende de ejemplos que tienen tanto entradas como sus correspondientes clases o salidas. El algoritmo se entrena en un conjunto de documentos etiquetados manualmente. Una vez que se completa el entrenamiento, el clasificador puede predecir categorías incluso para tipos de datos o documentos nunca antes vistos.
- **Aprendizaje No Supervisado**. En esta técnica los documentos similares se clasifican en diferentes grupos. Esta clasificación se puede realizar en función de la plantilla, las palabras o etiquetas de la fuente. Estos algoritmos pueden lograr una mayor precisión si se definen y ajustan reglas específicas.

- **Basado en Reglas**. Esta técnica es uno de los métodos tradicionales para la clasificación de documentos que aprovecha la capacidad de comprensión del lenguaje natural de un sistema y escribe reglas gramaticales que ayudarían al sistema para que actúe como una persona en la tarea de clasificación de un documento. Este método puede tener una mayor precisión pero su construcción es complicada porque requiere mucho tiempo y es difícil de escalar.

Una vez analizados los pasos que se siguen para realizar la clasificación de documentos así como las técnicas con las que se puede llevar a cabo, se puede concluir, según se expuso en el apartado relativo a la implementación de *MLOps* (ver Sección 4.3.5), que el objetivo de esta etapa sería alcanzar un nivel 2 de *MLOps*. Para ello, los equipos de trabajo deben estar involucrados, debe existir una estrategia adecuada de recopilación de datos, tanto la fase de desarrollo como la de entrega estarán automatizadas, se realizará una entrega continua de modelos, se ejecutarán pruebas con cada lanzamiento, se realizarán tareas de supervisión para el reentrenamiento del modelo y existirá integración continua.

Respecto al despliegue, en esta etapa podría ser apropiado el modelo Canary, que consiste en cambiar de forma gradual el tráfico de la versión A a la versión B de tal forma que el modelo sólo está expuesto a una pequeña porción de los datos. Es un despliegue interesante para analizar la tasa de error y supervisar el rendimiento del modelo porque además implica un retroceso rápido si fuera necesario.

También se podría plantear el uso de algunas de las herramientas *MLOps* comentadas (ver Sección 4.3.6). En esta etapa podría ser interesante utilizar alguna de las herramientas de modelado, en concreto las utilizadas para el seguimiento de experimentos o la optimización de hiperparámetros y también alguna herramienta de operacionalización, sobre todo las relativas a la supervisión de los modelos.

5.2.4. Extracción de Información

La clave del ciclo de IDP es la etapa de extracción de datos, la cual implica extraer información de los documentos. Los modelos ML se encargan de obtener información específica, como fechas, nombres o cifras de la documentación preprocesada y clasificada en las etapas anteriores. Habitualmente, las soluciones de IDP tienen una biblioteca de varios modelos ML, cada uno de ellos diseñado para funcionar en uno o varios campos en concreto.

Dentro de esta etapa existen, fundamentalmente, tres modelos de extracción:

- **Pares Clave - Valor**. Se trata de extraer los valores asignados a identificadores clave únicos, por ejemplo, los formularios.
- **Tablas**. Se trata de extraer los elementos de línea dispuestos en forma de tabla.
- **Entidades**. Se trata de asignar una etiqueta con significado a cada palabra o segmento de texto del documento, por ejemplo, fecha y hora, direcciones y valores numéricos.

Y esta extracción se puede realizar basada en varias estrategias:

1. **OCR**. El uso de diferentes motores OCR es una de las características que definen los modelos de IDP. Las soluciones más sofisticadas aplican un método de varias capas que combina varias salidas hasta obtener una precisión casi perfecta. Aún así, durante este proceso podrían producirse algunos errores:
   - *Errores en la Detección de Palabras*. La mala calidad de la imagen puede provocar fallos al detectar bloques de texto.
   - *Errores en la Segmentación de Palabras*. La detección incorrecta de los espacios entre palabras o alineaciones pueden provocar la incorrecta interpretación de una palabra.
   - *Errores en la Segmentación de Caracteres*. La imposibilidad de detectar caracteres individuales en una palabra segmentada.
   - *Errores en el Reconocimiento de Caracteres*. Una imagen de un carácter delimitada puede impedir la identificación del carácter correcto.
2. **Basada en Reglas**. Los modelos basados en reglas se pueden aplicar en documentos estructurados y semiestructurados. Estos modelos pueden identificar pares clave-valor y elementos de línea, tomando una referencia de posición en un documento.
3. **Basada en el Aprendizaje**. Los modelos DL y las técnicas de extracción de datos híbridas OCR basadas en ML necesitan aprendizaje supervisado/no supervisado para entrenar sus modelos. La eficiencia de estos modelos está condicionada por la tasa de precisión.

Una vez analizados los modelos y las estrategias que se siguen para realizar la extracción de información, se puede concluir, según se expuso en el apartado relativo a la implementación de *MLOps* (ver Sección 4.3.5), que el objetivo de esta etapa sería alcanzar un nivel 2 de *MLOps*. Por lo tanto, sería fundamental alcanzar la integración continua, es decir, que se diseñen, prueben y combinen los nuevos cambios en el código de una aplicación con regularidad en un repositorio compartido. La extracción de información es una etapa en la que puede ser preferible un mayor nivel de precisión frente a la automatización.

Respecto al despliegue, en esta etapa también podría ser apropiado el modelo Canary. Su única desventaja es que su despliegue es lento, pero es preferible frente a la cantidad de recursos que necesitarían los modelos Blue Green o Shadow.

Igualmente se podría plantear el uso de algunas de las herramientas *MLOps* comentadas (ver Sección 4.3.6). En esta etapa podría ser interesante utilizar alguna de las herramientas de gestión de datos, tanto las relativas al etiquetado como al control de versiones.

5.2.5. Validación de Información

Durante esta etapa se detectan las imprecisiones de los datos extraídos. Se aplican reglas de validación de datos para que se pueda detectar y marcar cualquier inexactitud para corregirla posteriormente. Para ello se puede aplicar el uso de **RegEx**, o expresiones regulares. Se trata de una herramienta para la coincidencia de patrones. Además, los modelos ML perfeccionan los datos extraídos, por ejemplo, mediante la corrección de errores ortográficos o el ajuste de datos a formatos estándar. Los datos extraídos pasan por una serie de comprobaciones de validación automatizadas o manuales para garantizar la precisión de los resultados del procesamiento.

El análisis de esta etapa, la validación de información, también es importante para conocer el alcance de la solución IDP respecto al dato. En esta etapa los datos se validan con diferentes criterios o técnicas con el objetivo final de mejorar la precisión del sistema IDP. De esta forma, cualquier corrección actúa como una entrada al sistema para que la precisión mejore en extracciones futuras. Este ciclo de retroalimentación aumentará la precisión del sistema. Con lo cual, esta etapa se analiza desde el punto de vista de las soluciones *Data Centric*.

Partiendo de este enfoque, también se podría plantear el uso de algunas de las herramientas comentadas (ver Sección 4.2.5). En concreto, se podría utilizar alguna herramienta relacionada con la observabilidad de los datos. Es cierto que quizá sea más interesante su uso en las fases iniciales del ciclo de vida de los datos, pero también podría aplicarse en esta etapa del ciclo IDP.

5.2.6. Validación con Intervención Humana

Dado que ningún modelo de extracción de datos es preciso al 100 %, será necesario incluir una etapa de intervención de un equipo de personas en el ciclo de IDP. Esta etapa aprovecha un marco de aprendizaje automático denominado **Human In The Loop (HITL)**, mediante el cual los datos problemáticos se envían a dichos equipos para revisarlos y corregirlos. Este enfoque permite que el modelo de validación aprenda continuamente y mejore su precisión con el tiempo, puesto que cuantos más documentos se procesen y revisen, mejor será la precisión del modelo de extracción de datos.

Esta etapa, la validación con intervención humana, también está directamente relacionada con el dato. Se trata de un segundo filtro que permite seguir aumentando la precisión del sistema. Al igual que en la etapa anterior, cualquier corrección actúa como una entrada al sistema para que la precisión mejore en extracciones futuras.

Las soluciones IDP deben ofrecer no sólo unas excelentes características de extracción y clasificación, sino también grandes capacidades de validación de datos y bucle de retroalimentación para gestionar de forma eficiente las posibles variaciones e imprecisiones de los datos.

5.2.7. Integración

Una vez que los datos se extraen, el software puede enviarlos a la base de datos o exportarlos en múltiples formatos. Los flujos de trabajo de IDP permiten a los usuarios convertir documentos en diferentes formatos, como JSON, XML o PDF. El archivo se pasa a un proceso comercial o un depósito de datos a través de API. Con lo cual, esta información estará disponible para su consumo inmediato y podrá ser utilizada por la empresa para tomar acciones rápidas y ofrecer un servicio eficiente a sus clientes.

5.3. Benefi

Esta tecnología se focaliza en la eliminación de tareas que sean repetitivas y manuales y en la conversión de datos no estructurados en formatos que puedan ser interpretables en aplicaciones y sistemas. Los datos son fundamentales para el flujo de trabajo de las empresas puesto que facilitan el progreso y la mejora de los procesos que se llevan a cabo en ella.

Con lo cual, se puede decir que son varios los beneficios que IDP ofrece a los usuarios:

1. **Eficacia**. Como se comentaba anteriormente, la automatización de los procesos conlleva la eliminación de las tareas manuales. Trasladando este aspecto a un flujo de trabajo que esté centrado en documentos, se traduce en una mayor eficiencia y eficacia en el flujo de trabajo diario.

2. **Ahorro**. Adoptar la tecnología IDP implica un cambio significativo en el tiempo de procesamiento y una reducción de los costes de mano de obra muy importante. Los procesos automáticos permiten completar el trabajo en un período de tiempo más corto, lo cual también implica una reducción de los costes operativos frente a las tareas manuales.

3. **Velocidad**. Sin duda se trata de la ventaja más obvia en la tecnología IDP, la velocidad con la que se procesan grandes volúmenes de datos si se compara con el tiempo que costaría hacerlo de forma manual.

4. **Calidad**. Uno de los objetivos de cualquier empresa es proporcionar un servicio de calidad a sus clientes. IDP implica la ausencia de cualquier posibilidad de error humano durante el procesamiento de documentos. Pero la calidad no es solo la ausencia de errores. También es organizar la información de forma correcta, en una ubicación segura y de fácil acceso.

5. **Confidencialidad**. En muchas ocasiones, las organizaciones manejan datos confidenciales que es necesario proteger. IDP es una tecnología que mantiene la información protegida almacenándola en un lugar seguro.

6. **Fiabilidad**. Parece obvio que cuando se trata de realizar tareas repetitivas, frente a las máquinas, las personas tienen la tendencia a equivocarse y cometer errores. IDP demuestra ser fiable aunque trate con grandes volúmenes de datos y garantiza el éxito del proceso automatizado de un flujo de trabajo basado en clasificar y extraer información.

7. **Rendimiento**. Un sistema IDP mejora significativamente el rendimiento automatizando las tareas de trabajo manual y minimizando el tiempo necesario para procesar los documentos.

8. **Escalabilidad**. IDP se puede aplicar a múltiples aplicaciones en múltiples áreas. No requiere ninguna instalación y no es específico de ningún proceso en concreto.

9. **Compatibilidad**. IDP está diseñado de forma que su integración en aplicaciones comerciales y en los sistemas de las organizaciones sea viable sin incurrir en costes de integración.

5.4. Aplicación de IDP en diferentes sectores

Son varias las industrias que actualmente se están beneficiando de la aplicación de tecnologías IDP. A continuación se enumeran algunos de los sectores [47] en los que se está aplicando:

- **Banca y Finanzas**. El sector bancario es una área en el que se manejan muchos documentos, con muchas páginas y que habitualmente requieren mucho esfuerzo a nivel humano para procesarlos, lo cual podría implicar un porcentaje de error bastante alto. La tecnología IDP podría procesar de forma automática extractos bancarios, balances, estados de flujo de efectivo, declaraciones de impuestos y otros documentos, a gran velocidad y con un mínimo riesgo de errores, por lo que el esfuerzo de los equipos de personas se centraría en la supervisión de los algoritmos.

- **Salud**. En el área de la salud, el mantenimiento de los registros de pacientes es esencial. Es necesario acceder de forma sencilla y bajo demanda a la información que se pueda necesitar de cualquier paciente en un momento determinado. También se debe tener en cuenta la diversidad de gestiones que se pueden incluir al respecto: admisión, identificación, gestión de resultados de pruebas o recetas. Por lo tanto, la digitalización a través de IDP permitiría que todos los registros y diagnósticos médicos se pudieran almacenar en un solo lugar y solo fueran accesibles cuando fuera estrictamente necesario.

- **RRHH**. El área de Recursos Humanos de las empresas también maneja una gran cantidad de datos: datos de empleados, datos de reclutamiento, estadísticas de progresión profesional, registros financieros, revisiones de progreso personal y datos de capacitación. Toda esa información se recopila prácticamente a diario, con lo cual, el esfuerzo para realizar esta tarea de forma manual puede llegar a ser inasumible. Las soluciones automatizadas que utilizan IDP mitigan el riesgo de perder información, tiempo y empleados valiosos. Esto también ayuda a convertir los datos recopilados en información para que el equipo de Recursos Humanos tenga un acceso rápido y sencillo a los datos relevantes.

- **Jurídico**. Los proveedores de servicios legales archivan y auditan, de forma diaria, documentos relacionados con fusiones y adquisiciones, registros de propiedad, detección de fraude y administración de contratos, entre otros. Estos documentos se deben archivar de la forma más organizada posible para facilitar la revisión de los casos de los clientes. Con lo cual, emplear un sistema automatizado que utiliza IDP para administrar datos y documentación ayudaría a mantener la seguridad y mejorar la calidad del trabajo.

- **Seguros**. Los procesos que se llevan a cabo en el área de seguros implican la recopilación de un gran volumen de datos de clientes para realizar el análisis de perfil de cada uno de ellos. Una entrada manual provocaría una reducción de la velocidad de procesamiento así como un aumento de los costes. Esto convierte a la tecnología IDP en la solución ideal, no solo para automatizar el procesamiento de documentos, sino para ofrecer al cliente el mayor nivel de satisfacción posible en un entorno altamente competitivo.

Parte III

Conclusiones Finales

Capítulo 6

Conclusiones y Trabajo Futuro

6.1. Conclusiones

Este Trabajo Final de Máster ha presentado las recientes tendencias en metodologías de control de calidad y pruebas para soluciones tecnológicas basadas en AI/ML y Big Data. En particular, durante el desarrollo del trabajo, el alumno ha estudiado los conceptos *Data Centric* y *MLOps* para el Aseguramiento de la Calidad en sistemas AI/ML y ha analizado un caso práctico, el *Procesamiento Inteligente de Documentos*, en donde la aplicación de estas metodologías es de especial relevancia. Los siguientes párrafos resumen la consecución de los objetivos planteados al inicio del proyecto.

En primer lugar era necesario tener el contexto adecuado. Se aportaron conceptos básicos relacionados con las **pruebas** como su definición, la clasificación de las pruebas, las etapas propias del ciclo de pruebas, las diferencias entre pruebas manuales y automáticas o las diferencias entre pruebas y **Aseguramiento de la Calidad**.

La **Inteligencia Artificial** (AI), el **Aprendizaje Automático** (ML) y el Big Data son los elementos que están impulsando el cambio. Los sistemas AI/ML son, al fin y al cabo, sistemas de software, por lo tanto, se ha estudiado la aplicación de los procesos y conceptos de aseguramiento de la calidad al ciclo de vida y la configuración de los modelos AI/ML y también a los datos. Pero para aplicar estos conceptos básicos de la calidad en la ingeniería del software a los sistemas AI/ML se han tenido en cuenta tanto los desafíos que supone ejecutar las pruebas de estos sistemas frente a los sistemas de software tradicional, como ciertos aspectos críticos que dificultan su aplicación.

También se ha realizado un análisis de las nuevas metodologías *Data Centric* y *MLOps* para el control de calidad de soluciones tecnológicas complejas basadas en Inteligencia Artificial. Estas dos tendencias revelan dos conceptos clave: la importancia del **dato** y la **automatización** de los procesos.

Una vez planteados los conceptos básicos y las tendencias más actuales, se ha realizado una aproximación a un caso de uso práctico, el **Procesamiento Inteligente de Documentos**. En los sistemas IDP se usan diferentes tecnologías de AI para clasificar, categorizar y extraer información relevante y validar los datos extraídos. Además de explicar los beneficios de los sistemas IDP y su aplicación en los diferentes sectores y ramas, se ha analizado cómo las tendencias actuales, *Data Centric* y *MLOps*, influyen en cada una de estas etapas del proceso IDP. Para ello se han analizado diferentes aspectos, como por ejemplo, el nivel de relevancia del dato en la solución IDP, las herramientas *Data Centric* / *MLOps* que se podrían aplicar, el nivel de implementación de *MLOps* o el tipo de despliegue que se podría recomendar en cada una de las etapas.

6.2. Trabajo Futuro

En este Trabajo Final de Máster se han analizado nuevas tendencias de aseguramiento de la calidad en sistemas AI/ML como *Data Centric* y *MLOps* y su posible aplicación a un caso de uso concreto, los sistemas IDP. Sin embargo, todavía existen otras líneas de trabajo futuro que no se han desarrollado en este proyecto por diferentes motivos, como el nivel de detalle y alcance. Estas líneas de trabajo futuro podrían ser las siguientes:

1. **Refinamiento, mejora y adaptación de los procesos Data Centric y MLOps a casos de uso concretos.** En este trabajo se ha abordado la aplicación de estas tendencias al *Procesamiento Inteligente de Documentos*, pero todavía se debe recorrer mucho camino para estandarizar y escalar la aplicación de estos conceptos y herramientas a otros casos de uso. De cualquier forma, a pesar de la estandarización, siempre será necesario un cierto nivel de customización para cada aplicación o caso de uso concreto.

2. **Desarrollo de herramientas y automatizaciones basadas en los conceptos de Data Centric y MLOps.** Están surgiendo diferentes opciones, tanto a nivel de herramientas como de procesos de automatización basados en *Data Centric* y *MLOps*, pero todavía queda mucho camino por recorrer y explorar.

3. **Evaluación de posibles costes derivados de la implementación.** En este Trabajo Final de Máster no se han estudiado los costes de recursos humanos, los costes del software y hardware que supondría el desarrollo de las soluciones y los costes de operación del día a día o mantenimiento, una vez que la solución ya esté implementada.

Parte IV

Bibliografía

Bibliografía

[1] QALovers. Las fases en la historia del testing . https://www.qalovers.com/2021/02/historia-testing.html. (Visitado 19-febrero-2022).

[2] Capgemini. World Quality Report 2021-22. https://www.sogeti.com/explore/reports/world-quality-report-2021-22/. (Visitado 19-febrero-2022).

[3] ISTQB. *Certified Tester - Foundation Level (CTFL) Syllabus - Version 2018 v3.1.1*, July 2021.

[4] ISO. ISO/IEC 25010 - System and software quality models. https://iso25000.com/index.php/en/iso-25000-standards/iso-25010. (Visitado 21-febrero-2022).

[5] Tuskr. Black-Box Testing. https://tuskr.app/learn/black-box-testing/. (Visitado 22-febrero-2022).

[6] Tuskr. White-Box Testing. https://tuskr.app/learn/white-box-testing/. (Visitado 21-febrero-2022).

[7] Ava Franklin. Software Testing Life Cycle – A Complete Guide For 2022. https://www.goodcore.co.uk/blog/software-testing-life-cycle/. (Visitado 22-febrero-2022).

[8] Laveena Ramchandani. Software Testing Life Cycle (STLC) – Benefits & Phases. https://blog.testproject.io/2020/11/03/software-testing-life-cycle-stlc/. (Visitado 04-marzo-2022).

[9] Chris Taylor. Top 6 Software QA Trends for 2021. https://www.bairesdev.com/blog/top-software-qa-trends-2021/. (Visitado 07-marzo-2022).

[10] Emma Dallas. Trends 2019: QAOps in Softwa- re Testing. https://blog.qatestlab.com/2019/04/16/ trends-2019-qaops-in-software-testing/. (Visitado 27-febrero-2022).

[11] Rockwell Anyoha. The History of Artificial Intelligence. https://sitn. hms.harvard.edu/flash/2017/history-artificial-intelligence/. (Visitado 12-marzo-2022).

[12] D. Rumelhart, G. Hinton, and R. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[13] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521:436–444, 2015.

[14] Peltarion. AI Concepts. https://peltarion.com/knowledge-center/ documentation/get-started-with-the-platform/ai-concepts/. (Visitado 15-marzo-2022).

[15] Sravya Reddysetty. Traditional Programming vs Machi- ne Learning. https://sravya-tech-usage.medium.com/ traditional-programming-vs-machine-learning-e9bbed5e491c/. (Visitado 18-marzo-2022).

[16] Song Huang, Er-Hu Liu, Zhan-Wei Hui, Shi-Qi Tang, and Suo-Juan Zhang. Challenges of Testing Machine Learning Applications. *International Journal of Performability Engineering*, 14(6):1275, 2018.

[17] David J. Hand and Shakeel Khan. Validating and Verifying AI Systems. *Patterns*, 1(3):100037, 2020.

[18] ISTQB. *Certified Tester - AI Testing (CT-AI) Syllabus - Version 1.0*, October 2021.

[19] Omar Sanseviero. AI en 3 minutos: Tipos de Ma- chine Learning. https://medium.com/ai-learners/ ai-en-3-minutos-tipos-de-machine-learning-945b708ac78/. (Visitado 26-marzo-2022).

[20] Hong Zhu, Dongmei Liu, Ian Bayley, Rachel Harrison, and Fabio Cuzzolin. Datamorphic Testing: A Methodology for Testing AI Applications. *CoRR*, abs/1912.04900, 2019.

[21] Kellie Carlson. Dual-coding theory in UX Design. https://k-carlson180.medium.com/ dual-coding-theory-in-ux-design-96db2a264456/. (Visitado 28-marzo-2022).

[22] Jianmin Guo, Yu Jiang, Yue Zhao, Quan Chen, and Jiaguang Sun. DLFuzz: differential fuzzing testing of deep learning systems. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ACM, oct 2018.

[23] Xiaofei Xie, Lei Ma, Felix Juefei-Xu, Hongxu Chen, Minhui Xue, Bo Li, Yang Liu, Jianjun Zhao, Jianxiong Yin, and Simon See. DeepHunter: Hunting Deep Neural Network Defects via Coverage-Guided Fuzzing, 2018.

[24] Eran Kinsbruner. Classification of Advanced AI and ML Testing Tools. https://www.perfecto.io/webinars/ classification-advanced-ai-and-ml-testing-tools. (Visitado 29-marzo-2022).

[25] PyCharm. Testing frameworks. https://www.jetbrains.com/help/ pycharm/testing-frameworks.html. (Visitado 20-abril-2022).

[26] Reinhard Pröll. *Towards a Model-Centric Software Testing Life Cycle for Early and Consistent Testing Activities*. doctoralthesis, Universität Augsburg, 2021.

[27] Towards data-centric machine learning: a short review. *ljvmiranda921.github.io*, 2021.

[28] Andrew Ng. The Batch. https://read.deeplearning.ai/the-batch/ issue-93/. (Visitado 02-abril-2022).

[29] The Data-Centric Manifesto. Principles. http://www. datacentricmanifesto.org/principles/. (Visitado 08-abril-2022).

[30] Dave McComb. The Data-Centric Revolution: Data-Centric vs. Data-Driven. https://tdan.com/ the-data-centric-revolution-data-centric-vs-data-driven/ 20288. (Visitado 08-abril-2022).

[31] Dave McComb. *The Data-Centric Revolution: Restoring Sanity to Enterprise Information Systems*. Technics Publications, 2019.

[32] Carol Dunn. The Difference Between Data-centric and Data-driven. https://www.asti.com/ the-difference-between-data-centric-and-data-driven/. (Visitado10-abril-2022).

[33] Jonathan Johnson and Muhammad Raza. Data Integrity vs Data Quality: An Introduction. https://www.bmc.com/blogs/ data-integrity-vs-data-quality/. (Visitado 11-abril-2022).

[34] Korporate. Los cinco tipos de fuentes de datos. https://grupokorporate.com/los-cinco-tipos-de-fuentes-de-datos/. (Visitado 12-abril-2022).

[35] Kaitlyn Henry and Maor Fridman. The Data-Centric AI Movement and Opportunities for the MLOps Ecosystem. https://openviewpartners.com/blog/data-ml-ops/. (Visitado 10-abril-2022).

[36] Cristiano Breuel. The Road to MLOps: Machine Learning as an Engineering Discipline. https://builtin.com/machine-learning/mlops. (Visitado 21-abril-2022).

[37] Dr. Larysa Visengeriyeva, Anja Kammer, Isabel Bär, Alexander Kniesz, and Michael Plöd. MLOps Principles. https://ml-ops.org/content/mlops-principles. (Visitado 30-abril-2022).

[38] Yashaswi Nayak. A Gentle Introduction to MLOps. https://towardsdatascience.com/ a-gentle-introduction-to-mlops-7d64a3e890ff. (Visitado 24- abril-2022).

[39] Edwin Webster. What is MLOps? – Benefits, how it works, and DevOps vs. MLOps. https://nealanalytics.com/blog/what-is-mlops/. (Visitado 24-abril-2022).

[40] Gaurav Sharma. MLOps vs DevOps: Let's Understand the Differences? https://www.analyticsvidhya.com/blog/2022/01/mlops-vs-devops-lets-understand-the-differences/. (Visitado 26-abril-2022).

[41] Satvik Garg, Pradyumn Pundir, Geetanjali Rathee, P. K. Gupta, Somya Garg, and Saransh Ahlawat. On Continuous Integration / Continuous Delivery for Automated Deployment of Machine Learning Models using MLOps. *CoRR*, abs/2202.03541, 2022.

[42] Sibanjan Das. MLOps for Enterprise AI. https://dzone.com/articles/mlops-for-enterprise-ai/. (Visitado 30-abril-2022).

[43] Philipp Ruf, Manav Madan, Christoph Reich, and Djaffar Ould-Abdeslam. Demystifying MLOps and Presenting a Recipe for the Selection of Open-Source Tools. *Applied Sciences*, 11(19), 2021.

[44] Nipuni Hewage and Dulani Meedeniya. Machine Learning Operations: A Survey on MLOps Tool Support. 2022.

[45] Docsumo. Overview of Intelligent Document Processing (IDP) and its Benefits. https://docsumo.com/blog/intelligent-document-processing-idp. (Visitado 06-mayo-2022).

[46] Width AI. How Intelligent Document Processing Uses Machine Learning To Remove Manual Processes From Your Business. https://www.width.ai/post/intelligent-document-processing. (Visitado 03-mayo-2022).

[47] Becoming Human. Intelligent document processing: a complete guide. https://becominghuman.ai/intelligent-document-processing-a-complete-guide-e23f054aefd7. (Visitado 04-mayo-2022).