Ingeniería de Datos para ML

Unidad 3: Ingeniería de Datos para ML

Dr. Humberto Farias Aroca
Universidad de La Serena

Objetivo General:

Profundizar en los aspectos técnicos de la ingeniería de datos necesarios para el machine learning.

Contenidos:

1. ¿Qué son los Pipelines de Datos?
2. Componentes de un Pipeline de Datos.
3. Fuentes de Datos: Extracción de datos.
4. Limpieza de Datos: Técnicas y herramientas.
5. Transformación de Datos: Métodos clave.
6. Carga de Datos: Procesos de almacenamiento.
7. Automatización de Pipelines: Herramientas y plataformas.
8. Monitorización del Pipeline: Estrategias esenciales.
9. Gestión de la Calidad de Datos.
10. Escalabilidad de Pipelines.
11. Seguridad en Pipelines de Datos.

Introducción

Importancia de la ingeniería de datos en ML

- **Base fundamental**: La ingeniería de datos es el pilar que sostiene todos los proyectos de ML, asegurando que los datos estén listos para su análisis y modelado.
- **Mejora la precisión**: Los datos adecuadamente preparados incrementan significativamente la precisión y efectividad de los modelos de ML.
- **Optimización de recursos**: Facilita el uso eficiente de los recursos computacionales, reduciendo los costos y tiempos de entrenamiento.
- **Calidad de datos**: Asegura la coherencia, accesibilidad y calidad de los datos, elementos esenciales para la robustez de las soluciones predictivas.
- **Impacto en decisiones**: Los datos bien gestionados permiten tomar decisiones basadas en información precisa y confiable.

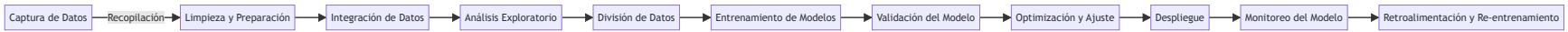
¿Qué son los Pipelines de Datos?

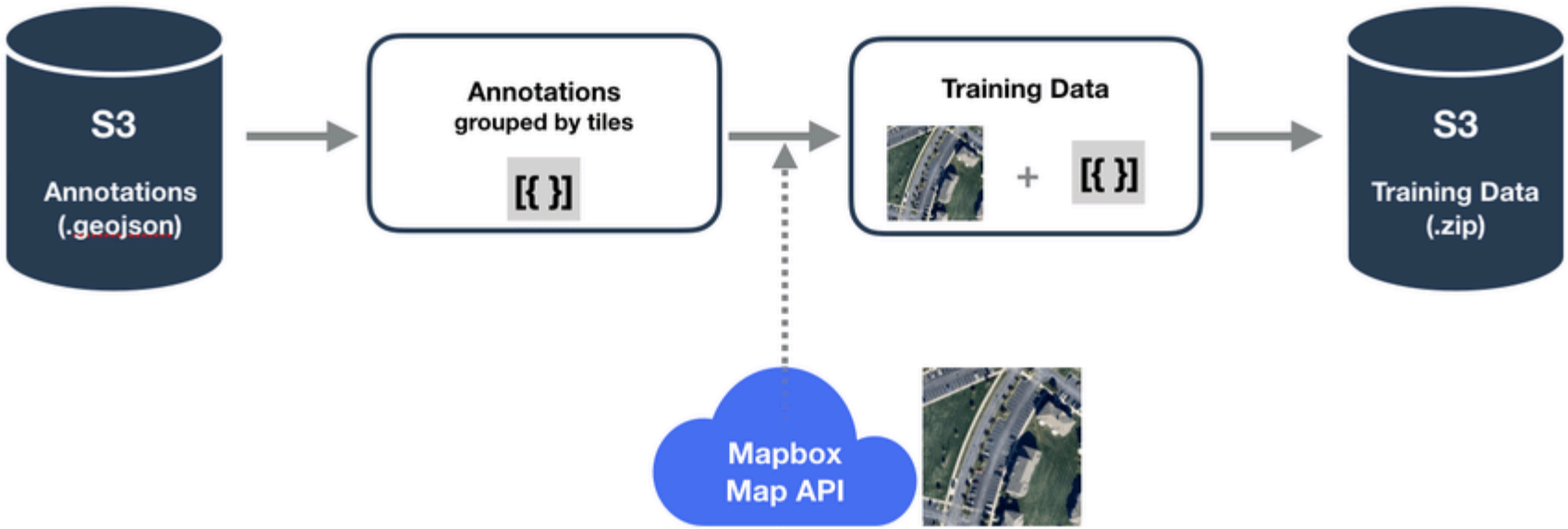
Definición y propósito

- **Definición**: Un pipeline de datos es una serie de pasos procesales configurados para la transferencia y transformación automatizada de datos desde una fuente hasta un destino.
- **Flujo sistemático**: Organiza el flujo de datos de manera eficiente y predecible, asegurando que los datos se manejen consistentemente.
- **Automatización**: Los pipelines automatizan el proceso de limpieza, validación, y transformación de datos, reduciendo errores humanos y aumentando la eficiencia.
- **Objetivo**: Facilitar el análisis de datos y el desarrollo de modelos de ML al proporcionar datos limpios y bien estructurados.
- **Beneficio clave**: Mejora significativamente la accesibilidad y calidad de los datos para los analistas y científicos de datos, permitiendo un análisis más rápido y decisiones basadas en datos más precisas.

Ejemplo de Pipeline de Datos y Machine Learning

Diagrama en Mermaid





Componentes de un Pipeline de Datos

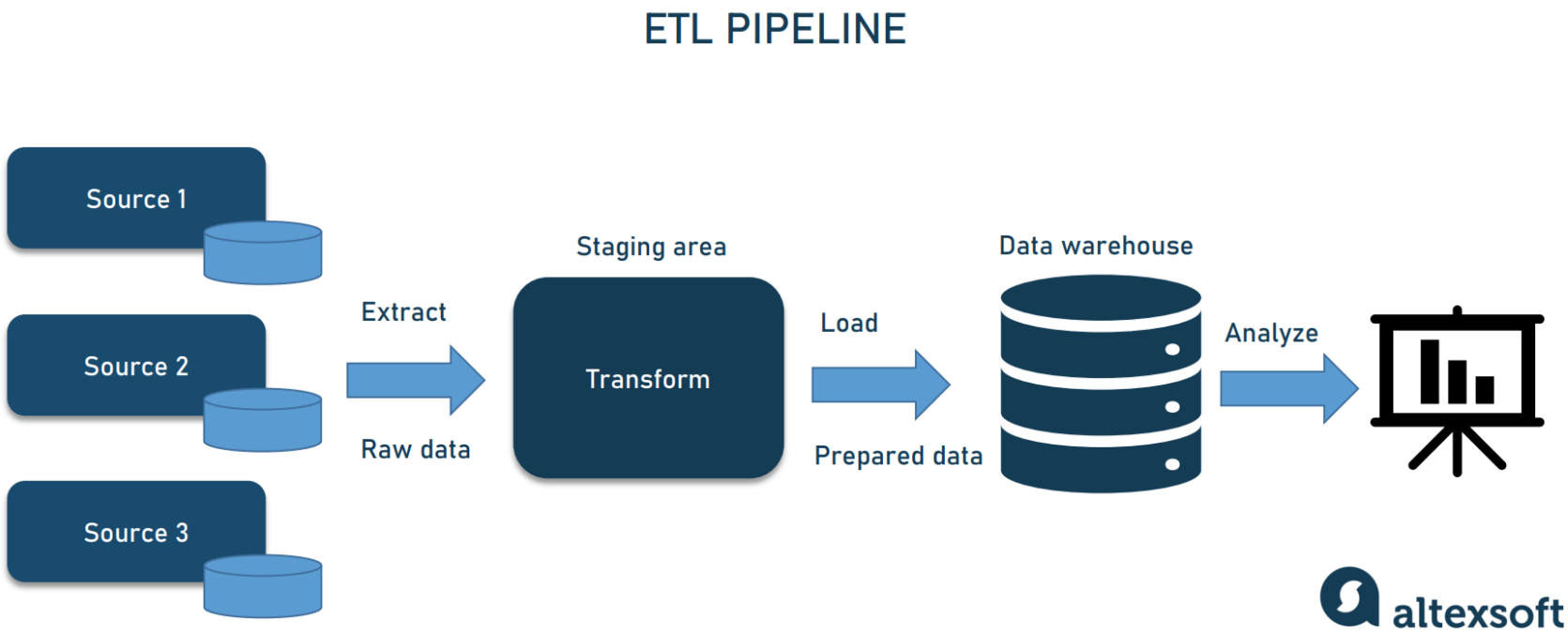
Elementos clave

- **Extracción de Datos**: Primera etapa donde los datos son recolectados de diversas fuentes, como bases de datos, archivos, o APIs.
- **Transformación de Datos**: Los datos recolectados son transformados para asegurar su calidad y adecuación para el análisis; esto incluye limpieza, normalización, y enriquecimiento.
- **Carga de Datos (Load)**: Los datos transformados son cargados en un sistema de destino, que puede ser una base de datos, un lago de datos o un almacén de datos para su almacenamiento y consulta.
- **Automatización**: Los procesos de ETL son típicamente automatizados para garantizar eficiencia y consistencia a través de herramientas especializadas.
- **Monitoreo y Mantenimiento**: Supervisión continua del pipeline para detectar y corregir errores, y ajustar procesos conforme cambian las necesidades de datos.

Este flujo ETL es crucial para el manejo efectivo de los datos en cualquier entorno de ML, asegurando que los datos estén listos y disponibles para análisis y modelado.

Componentes de un Pipeline de Datos

Elementos clave



Fuentes de Datos: Extracción de datos

Métodos y desafíos

- **Diversidad de Fuentes**: Datos pueden ser extraídos de bases de datos, redes sociales, dispositivos IoT, y más. La diversidad presenta retos en la homogeneización.
- **Técnicas de Extracción**: Incluyen scraping web, APIs, conexiones directas a bases de datos, y flujos de datos en tiempo real.
- **Desafíos de Calidad**: La calidad de datos varía, lo que requiere métodos robustos de validación y limpieza para asegurar la integridad.
- **Desafíos de Escala**: Manejar grandes volúmenes de datos y su almacenamiento presenta desafíos significativos en términos de rendimiento y costos.
- **Automatización y Monitoreo**: Esencial para gestionar la extracción de datos eficientemente y responder a problemas en tiempo real.

Tecnologías y Lenguajes para la Extracción de Datos

Ejemplos comunes

- **Lenguajes de Programación**: Python y R son ampliamente usados debido a sus librerías de manipulación de datos como Pandas y dplyr.
- **Herramientas de ETL**: Talend, Informatica, y Apache NiFi ofrecen soluciones robustas para automatizar el proceso de extracción, transformación y carga.
- **Bases de Datos**: SQL para bases de datos relacionales y NoSQL para datos no estructurados, facilitando la extracción específica y eficiente.
- **Plataformas de Big Data**: Hadoop y Spark son esenciales para manejar grandes volúmenes de datos distribuidos, con capacidades avanzadas de procesamiento y almacenamiento.
- **APIs y Web Scraping**: Utilizar APIs para acceder a datos de servicios web o scraping para extraer datos de sitios web con herramientas como Beautiful Soup o Scrapy.

Estos ejemplos representan tecnologías clave que permiten la extracción efectiva y eficiente de datos de múltiples fuentes.

Contexto de Cloud Computing en la Extracción de Datos

Ventajas y herramientas comunes

- **Flexibilidad y Escalabilidad**: La nube ofrece recursos escalables que se adaptan a las necesidades de extracción de datos, permitiendo manejar picos de demanda de forma eficiente.
- **Herramientas Especializadas**: Servicios como AWS Glue, Google Cloud Dataflow y Azure Data Factory facilitan procesos de ETL completamente gestionados y escalables.
- **Costo-Eficiencia**: Los servicios en la nube reducen la necesidad de infraestructura física y el mantenimiento asociado, optimizando los costos.
- **Acceso y Colaboración**: Facilita el acceso remoto a los datos y la colaboración entre equipos distribuidos.
- **Seguridad Mejorada**: Proveedores de nube ofrecen robustas medidas de seguridad, ayudando a proteger los datos durante la extracción y el almacenamiento.

Estas características hacen que el cloud computing sea una opción preferida para modernizar y potenciar los pipelines de extracción de datos.

Limpieza de Datos: Técnicas y herramientas

Mejorando la calidad de los datos

- **Identificación de Errores y Valores Atípicos**: Uso de técnicas estadísticas para detectar y manejar valores inusuales o incorrectos.
- **Manejo de Valores Faltantes**: Aplicación de métodos como la imputación, eliminación o interpolación para tratar los datos incompletos.
- **Estandarización y Normalización**: Conversión de formatos y escalas a estándares comunes para facilitar su análisis y procesamiento.
- **Deduplicación**: Eliminación de registros repetidos o redundantes para asegurar la unicidad en los datasets.
- **Validación de Datos**: Implementación de reglas de validación para garantizar la precisidad y coherencia de los datos.

Estas estrategias son cruciales para garantizar la fiabilidad y utilidad de los datos en cualquier proyecto de análisis o machine learning.

Ejemplos de Tecnologías y Lenguajes para la Limpieza de Datos

Herramientas Especializadas

- **Python y R**: Dominan en el manejo de datos con librerías como pandas y tidyverse respectivamente, que facilitan la limpieza y manipulación de datos.
- **OpenRefine**: Herramienta poderosa para la limpieza y transformación de datos grandes, permitiendo manejar incoherencias de forma eficiente.
- **SQL**: Utilizado para consultas complejas que limpian y transforman datos directamente en bases de datos.

Estas tecnologías representan soluciones clave para asegurar la calidad de los datos en los proyectos de análisis y ML.

Contexto de Cloud Computing en la Limpieza de Datos

Plataformas y Servicios

- **AWS Glue**: Servicio de ETL en la nube que prepara y carga datos de manera segura.
- **Google Cloud DataPrep**: Herramienta de Google Cloud para explorar, limpiar y preparar datos estructurados y no estructurados.
- **Azure Data Factory**: Integra transformación de datos y capacidades de limpieza para preparar datos para el análisis.

El uso de la nube para la limpieza de datos ofrece escalabilidad, eficiencia y mejor colaboración entre equipos dispersos geográficamente.

Transformación de Datos: Métodos clave

Preparación para el análisis

- **Normalización**: Ajusta las escalas de los datos para que las características tengan un impacto equitativo en el análisis.
- **Escalado**: Transforma datos a un rango específico, como 0-1, mejorando la convergencia en algoritmos de aprendizaje.
- **Agrupación**: Técnicas como la codificación de variables categóricas para convertirlas en formatos numéricos, facilitando su procesamiento por modelos de ML.
- **Reducción de Dimensionalidad**: Aplica técnicas como PCA para reducir el número de variables, enfocando en las más informativas.

Estos métodos son cruciales para preparar los datos correctamente, garantizando análisis y modelados efectivos.

Ejemplos de Tecnologías y Lenguajes para la Transformación de Datos

Herramientas Especializadas

- **Scikit-learn**: Ofrece herramientas para escalado, normalización y reducción de dimensionalidad.
- **Pandas**: Facilita la manipulación y transformación de datos estructurados.
- **R (dplyr, tidyr)**: Potentes paquetes para transformar y preparar datos para análisis.

Estas herramientas permiten a los analistas y científicos de datos preparar eficientemente los datos para el modelado y análisis avanzado.

Contexto de Cloud Computing en la Transformación de Datos

Plataformas y Servicios

- **Amazon Redshift**: Servicio de almacenamiento que incluye funcionalidades SQL para transformaciones complejas de datos.
- **Google BigQuery**: Ofrece capacidades de manipulación de datos a gran escala mediante SQL en la nube.
- **Azure Stream Analytics**: Permite procesar y transformar datos en tiempo real en la nube.

Estas plataformas proporcionan la capacidad de manejar y transformar grandes volúmenes de datos en un entorno escalable y accesible.

Carga de Datos: Procesos de almacenamiento

Almacenamiento óptimo para acceso

- **Estrategias de Almacenamiento**: Decidir entre bases de datos SQL vs. NoSQL basado en la estructura de los datos y los requisitos de consulta.
- **Sistemas de Almacenamiento**: Utilizar almacenes de datos que soporten operaciones de lectura y escritura rápidas y eficientes, como sistemas de almacenamiento en columnas para análisis de grandes volúmenes de datos.
- **Accesibilidad**: Asegurar que los datos estén almacenados en formatos y sistemas que permitan fácil acceso y manipulación por las herramientas de análisis.
- **Optimización del Almacenamiento**: Implementar técnicas como la indexación y la partición para mejorar el rendimiento de las consultas.

Ejemplos de Tecnologías y Lenguajes para esta Etapa

Herramientas Especializadas

- **Bases de Datos Relacionales**: PostgreSQL, MySQL para transacciones y operaciones que requieren integridad de datos y complejas relaciones.
- **Bases de Datos NoSQL**: MongoDB, Cassandra para manejar grandes volúmenes de datos no estructurados con esquemas flexibles.
- **Almacenes de Datos**: Amazon Redshift, Google BigQuery, que ofrecen capacidades de procesamiento de consultas masivas optimizadas para análisis.
- **Herramientas de Gestión de Datos**: Apache Hadoop para procesamiento distribuido y Apache Spark para procesamiento en memoria rápida.

Contexto de Cloud Computing en la Carga de Datos

Plataformas y Servicios

- **Servicios de Almacenamiento en la Nube**: Amazon S3, Google Cloud Storage ofrecen soluciones escalables y seguras para almacenamiento de datos.
- **Plataformas de Datos como Servicio (DaaS)**: Plataformas como Snowflake y Azure SQL Database que proporcionan servicios de bases de datos gestionados que escalan automáticamente según la demanda.
- **Integración y Orquestación**: Herramientas como AWS Glue y Azure Data Factory para orquestar y automatizar flujos de trabajo de datos en la nube.
- **Ventajas de Cloud**: Flexibilidad, escalabilidad y reducción de costos operativos al eliminar la necesidad de gestionar infraestructura física.

Automatización de Pipelines: Herramientas y plataformas

Herramientas para la eficiencia

- **Apache Airflow**: Herramienta de código abierto que permite diseñar, programar y monitorizar flujos de trabajo complejos.
- **Apache NiFi**: Plataforma integrada para automatizar el movimiento y la gestión de datos entre sistemas.
- **Luigi**: Desarrollado por Spotify, permite la gestión de pipelines de tareas complejas, enfocado en la estabilidad y la reusabilidad.
- **Azure Data Factory**: Servicio de integración de datos que facilita la creación, programación y orquestación de flujos de datos.
- **AWS Step Functions**: Servicio para coordinar múltiples servicios de AWS en flujos de trabajo automatizados.

Estas herramientas y plataformas permiten automatizar tareas repetitivas, asegurar la coherencia en la ejecución y mejorar la eficiencia de los pipelines de datos.

Ejemplos de Tecnologías y Lenguajes para esta Etapa

Herramientas Especializadas

- **Python y R**: Lenguajes de programación frecuentemente utilizados para scripts de automatización en data science.
- **Jenkins y CircleCI**: Herramientas de integración continua que pueden automatizar la ejecución de pipelines de prueba y despliegue.
- **Terraform**: Utilizado para la gestión de infraestructura como código, puede automatizar la configuración de entornos de datos.
- **Kubernetes**: Ideal para orquestar contenedores de aplicaciones, especialmente útil en entornos de microservicios que manejan datos.

Estas tecnologías facilitan la automatización y gestión eficiente de los pipelines de datos, apoyando tanto el desarrollo como la operación.

Contexto de Cloud Computing en la Automatización de Pipelines

Plataformas y Servicios

- **Google Cloud Composer**: Servicio de gestión de flujos de trabajo basado en Apache Airflow, optimizado para la nube.
- **Amazon Glue**: Servicio ETL administrado que prepara y combina datos para el análisis, integrando datos almacenados en varias fuentes de AWS.
- **Azure Logic Apps**: Permite la creación de flujos de trabajo automatizados y aplicaciones integradas sin necesidad de escribir código.
- **Beneficios del Cloud**: Acceso a escalabilidad instantánea, alta disponibilidad, y reducción de costes operativos al usar servicios en la nube para la automatización de pipelines.

Estas plataformas de cloud computing ofrecen soluciones robustas que permiten a las empresas automatizar y escalar la gestión de sus pipelines de datos sin grandes inversiones en hardware o mantenimiento.

Monitorización del Pipeline: Estrategias esenciales

Manteniendo la integridad del pipeline

- **Definición de Métricas de Salud**: Establecer métricas clave para evaluar la salud del pipeline, como tiempos de ejecución, tasas de error, y calidad de los datos.
- **Alertas y Notificaciones**: Implementar sistemas de alerta que notifiquen al equipo de operaciones o a los desarrolladores cuando se detecten anomalías.
- **Logging Detallado**: Mantener registros detallados de todas las operaciones y transformaciones dentro del pipeline para facilitar la depuración y el análisis post-mortem.
- **Revisión Continua de Rendimiento**: Programar revisiones regulares del rendimiento del pipeline para asegurar que continúa cumpliendo con los requisitos operativos y de negocio.
- **Automatización de la Recuperación**: Desarrollar mecanismos de recuperación automática para minimizar el tiempo de inactividad y garantizar la continuidad del servicio.

Ejemplos de Tecnologías y Lenguajes para esta Etapa

Herramientas Especializadas

- **Prometheus y Grafana**: Utilizados para la monitorización del rendimiento y visualización de métricas en tiempo real.
- **Elasticsearch, Logstash y Kibana (ELK Stack)**: Poderosa combinación para manejar, buscar y visualizar grandes volúmenes de datos de log.
- **Splunk**: Herramienta que permite buscar, monitorear y analizar datos de máquina generados por aplicaciones, sistemas e infraestructura.
- **Python y Bash**: Lenguajes de scripting comunes para escribir scripts de monitoreo y alerta personalizados.

Estas tecnologías proporcionan las capacidades necesarias para implementar una monitorización efectiva y responder rápidamente a los problemas que puedan surgir.

Contexto de Cloud Computing en la Monitorización de Pipelines

Plataformas y Servicios

- **AWS CloudWatch**: Servicio de monitorización para recursos y aplicaciones de AWS que recoge datos de log, métricas y eventos.
- **Google Cloud Monitoring**: Ofrece visibilidad completa sobre el rendimiento, la capacidad y la salud de las aplicaciones y servicios en Google Cloud.
- **Microsoft Azure Monitor**: Recopila, analiza y actúa sobre los datos telemétricos de la nube y los entornos locales para maximizar la disponibilidad y el rendimiento.
- **Ventajas del Cloud**: Los servicios en la nube ofrecen escalabilidad, flexibilidad y robustez necesarias para soportar soluciones de monitorización avanzadas a gran escala.

Estas plataformas en la nube están especialmente diseñadas para integrarse sin problemas con sistemas y aplicaciones modernas, proporcionando herramientas avanzadas para la monitorización y gestión de la salud del pipeline de datos.

Gestión de la Calidad de Datos

Asegurando la excelencia

- **Establecimiento de Normas de Calidad**: Definir claramente las métricas de calidad de datos que incluyen exactitud, completitud, consistencia, y relevancia para asegurar que los datos cumplan con los estándares necesarios para el análisis.
- **Validación de Datos en Entrada**: Implementar controles de validación en los puntos de entrada de datos para detectar y corregir errores inmediatamente. Esto puede incluir verificaciones de tipo de datos, rangos aceptables y la integridad referencial.
- **Limpieza de Datos**: Aplicar técnicas de limpieza regularmente para eliminar duplicados, corregir errores y tratar valores faltantes, utilizando herramientas automatizadas y scripts personalizados.

Gestión de la Calidad de Datos

Asegurando la excelencia

- **Auditoría y Rastreo de Datos**: Mantener registros de auditoría para el seguimiento de la procedencia y los cambios en los datos a lo largo del tiempo, lo que facilita la identificación de la fuente de problemas de calidad y la verificación de la transformación de datos.
- **Pruebas Continuas**: Establecer un proceso de pruebas continuas que incluya pruebas automáticas para validar la calidad de los datos después de cada actualización o modificación en el sistema de procesamiento de datos.
- **Retroalimentación del Usuario Final**: Incorporar la retroalimentación de los usuarios finales sobre la calidad de los datos en el proceso de mejora continua, asegurando que los datos cumplan con las expectativas y necesidades de los usuarios.

Estos métodos constituyen un marco integral para la gestión de la calidad de los datos, asegurando que la información manejada a lo largo de los pipelines sea de alta calidad y adecuada para la toma de decisiones y análisis avanzados.

Escalabilidad de Pipelines en MLOps

Preparándose para el crecimiento

- **Diseño Modular y Desacoplado**: Construir pipelines con componentes que se puedan escalar o modificar de manera independiente. Esto permite actualizar algoritmos de ML, gestionar grandes volúmenes de datos o introducir nuevas tecnologías sin interrumpir el proceso operativo.
- **Automatización y Orquestación Robusta**: Utilizar plataformas de orquestación como Kubeflow, que se integran con Kubernetes para gestionar y escalar automáticamente los pipelines de ML de acuerdo con la demanda.
- **Infraestructura de Cloud Elástica**: Emplear servicios de cloud computing que ofrecen elasticidad y escalabilidad automática, como AWS, Google Cloud y Azure. Estos servicios pueden proporcionar recursos computacionales adicionales en minutos para manejar picos de carga. rgo plazo.

Escalabilidad de Pipelines en MLOps

Preparándose para el crecimiento

- **Optimización de Recursos mediante Tecnologías Específicas de ML**: Implementar técnicas específicas de ML como entrenamiento distribuido y batch processing para mejorar la eficiencia operativa y la velocidad de procesamiento de datos.
- **Arquitecturas Basadas en Microservicios y Eventos**: Adoptar microservicios para descomponer funciones complejas en servicios más pequeños y manejables que operan sobre una arquitectura orientada a eventos, facilitando así la escalabilidad y la mantenibilidad.
- **Monitoreo Proactivo y Logs Detallados**: Integrar soluciones avanzadas de monitoreo y logging que proporcionen insights en tiempo real sobre el rendimiento del pipeline y detecten problemas antes de que afecten las operaciones.

Estas estrategias proporcionan una base sólida para que los pipelines de MLOps no sólo manejen las cargas actuales, sino que también estén preparados para adaptarse a las necesidades futuras, asegurando así que el proceso de machine learning sea eficiente y escalable a la

Seguridad en Pipelines de Datos

Protegiendo la información

En el contexto de MLOps, asegurar la integridad y confidencialidad de los datos a lo largo de todo el pipeline es crucial. Aquí exploramos algunas consideraciones y prácticas clave para proteger los datos en cada etapa del pipeline de datos de machine learning.

- **Autenticación y Autorización**: Implementar controles de acceso robustos en cada punto del pipeline. Utilizar autenticación multifactor y gestión de identidades para controlar el acceso a los datos y recursos del pipeline.
- **Encriptación**: Aplicar encriptación de datos en reposo y en tránsito. Utilizar protocolos seguros como TLS para la transmisión de datos y algoritmos fuertes de encriptación para los datos almacenados.
- **Auditoría y Monitoreo de Acceso**: Llevar un registro detallado de quién accede a los datos y cuándo. Utilizar herramientas de monitoreo para detectar y alertar sobre actividades sospechosas o no autorizadas.

Seguridad en Pipelines de Datos

Protegiendo la información

- **Gestión de Configuraciones y Vulnerabilidades**: Mantener las plataformas, las herramientas y las dependencias del pipeline actualizadas y configuradas de manera segura para protegerse contra vulnerabilidades conocidas.
- **Seguridad de Datos en Modelos de ML**: Asegurar que los datos utilizados en el entrenamiento de modelos no expongan información sensible o permitan inferencias que puedan comprometer la privacidad del usuario.
- **Pruebas de Seguridad Regularmente**: Realizar pruebas de penetración y auditorías de seguridad regularmente para identificar y mitigar posibles brechas de seguridad.

Estas prácticas ayudan a crear un entorno de MLOps seguro, donde la protección de los datos se mantiene en todas las fases del desarrollo y operación de modelos de machine learning, desde la recopilación de datos hasta la implementación y monitorización de modelos en producción.

Desafíos Comunes y Soluciones

Superando obstáculos en la Ingeniería de Datos para MLOps

- **Integración de Diversas Fuentes de Datos**:
  - **Desafío**: Integrar y homogeneizar datos de múltiples fuentes, que a menudo tienen formatos inconsistentes o están incompletos.
  - **Solución**: Implementar herramientas de integración de datos como Apache NiFi o Talend que automatizan la transformación y normalización de datos. Utilizar técnicas de ETL robustas para asegurar la consistencia.
- **Gestión de Grandes Volumen de Datos**:
  - **Desafío**: Manejar eficientemente grandes volúmenes de datos, lo que puede llevar a problemas de rendimiento y escalabilidad.
  - **Solución**: Adoptar tecnologías de procesamiento distribuido como Apache Spark o Hadoop. Utilizar soluciones de almacenamiento en la nube que ofrecen escalabilidad y rendimiento, como Amazon S3 o Google Cloud Storage.

Desafíos Comunes y Soluciones

Superando obstáculos en la Ingeniería de Datos para MLOps

- **Calidad y Limpieza de Datos**:
  - **Desafío**: Asegurar la calidad y la limpieza de los datos, que son críticos para el rendimiento de los modelos de ML.
  - **Solución**: Utilizar herramientas de calidad de datos como Deequ o Great Expectations para definir y verificar expectativas de calidad de datos. Automatizar la limpieza de datos mediante scripts y pipelines dedicados.
- **Latencia en Tiempo Real y Procesamiento**:
  - **Desafío**: Reducir la latencia en aplicaciones que requieren respuestas en tiempo real.
  - **Solución**: Implementar arquitecturas orientadas a eventos con Apache Kafka para el procesamiento en tiempo real. Optimizar los pipelines con procesamiento en memoria.

Desafíos Comunes y Soluciones

Superando obstáculos en la Ingeniería de Datos para MLOps

- **Seguridad y Privacidad de los Datos**:
  - **Desafío**: Proteger los datos y garantizar la privacidad, especialmente con regulaciones como GDPR y HIPAA.
  - **Solución**: Encriptar datos en reposo y en tránsito, implementar políticas de seguridad estrictas y emplear técnicas de anonimización de datos cuando sea necesario.
- **Colaboración y Versionado**:
  - **Desafío**: Coordinar el trabajo entre equipos de datos, desarrollo y operaciones que a menudo utilizan herramientas y procesos diferentes.
  - **Solución**: Utilizar plataformas de MLOps como MLflow o DVC (Data Version Control) para el versionado de datos y modelos, y promover prácticas de CI/CD para la colaboración efectiva.

Mejores Prácticas en Ingeniería de Datos

Consejos para el éxito en MLOps

- **Automatización de Procesos**: Automatiza tanto como sea posible para minimizar el error humano y aumentar la reproducibilidad. Utiliza herramientas de orquestación de workflows como Apache Airflow para automatizar los pipelines de ETL y MLOps.

- **Pruebas Rigurosas**: Implementa pruebas continuas a lo largo del pipeline, incluyendo pruebas de datos, pruebas de código y pruebas de rendimiento del modelo. Esto asegura que cualquier problema se detecte y resuelva rápidamente.

- **Monitorización y Logging**: Establece una monitorización exhaustiva del pipeline y sistemas de logging para mantener un control sobre el rendimiento y detectar problemas de manera proactiva. Utiliza herramientas como Prometheus y Grafana para visualizar y monitorizar las métricas del sistema.

- **Gestión de la Calidad de los Datos**: Define y aplica estrictos controles de calidad de datos en cada etapa del pipeline. Utiliza herramientas como Apache Griffin o Deequ para asegurar que los datos sean precisos, consistentes y completos.

Mejores Prácticas en Ingeniería de Datos

Consejos para el éxito en MLOps

- **Documentación Completa**: Mantén una documentación detallada de todos los aspectos del pipeline, incluyendo la arquitectura del sistema, configuraciones, dependencias y cambios realizados. Esto es crucial para el mantenimiento y la escalabilidad a largo plazo.
- **Escalabilidad y Flexibilidad**: Diseña los pipelines para que sean escalables y flexibles. Asegúrate de que pueden manejar aumentos en el volumen de datos y adaptables a nuevas tecnologías y algoritmos.
- **Seguridad de Datos**: Aplica medidas de seguridad robustas, incluyendo encriptación de datos, gestión segura de claves y políticas de acceso estrictas para proteger los datos y cumplir con las regulaciones de privacidad.
- **Colaboración Interdisciplinaria**: Fomenta una cultura de colaboración entre los equipos de datos, desarrollo y operaciones. Utiliza plataformas de MLOps que soporten la colaboración y la integración continua para un desarrollo ágil.
- **Uso de Contenedores y Microservicios**: Emplea contenedores y microservicios para encapsular diferentes partes del pipeline, lo que facilita la gestión, la actualización y el despliegue independiente de componentes del sistema.

Ingeniería de Datos para ML

Unidad 4

Desarrollo de Modelos de ML: Principios y Prácticas.

Dr. Humberto Farias Aroca
Universidad de La Serena

Objetivos de la Unidad

- Comprender **principios fundamentales** del desarrollo de modelos de ML.
- Aplicar **prácticas recomendadas** alineadas con objetivos empresariales.
- Mejorar la **colaboración** entre equipos de datos y operaciones.

Fundamentos del Desarrollo de Modelos

Fundamentos del Desarrollo de Modelos

Tipos de Modelos y Sus Aplicaciones

- **Modelos Supervisados**: Utilizados para tareas de predicción como clasificación y regresión. Ejemplos incluyen la predicción de fraudes bancarios o la estimación de precios de viviendas.
- **Modelos No Supervisados**: Aplicados en clustering y detección de anomalías, como segmentación de clientes en marketing o detección de actividades sospechosas en transacciones.
- **Aprendizaje por Refuerzo**: Desarrollado para decisiones secuenciales y optimización, como en la navegación de robots o en juegos de estrategia.

Fundamentos del Desarrollo de Modelos

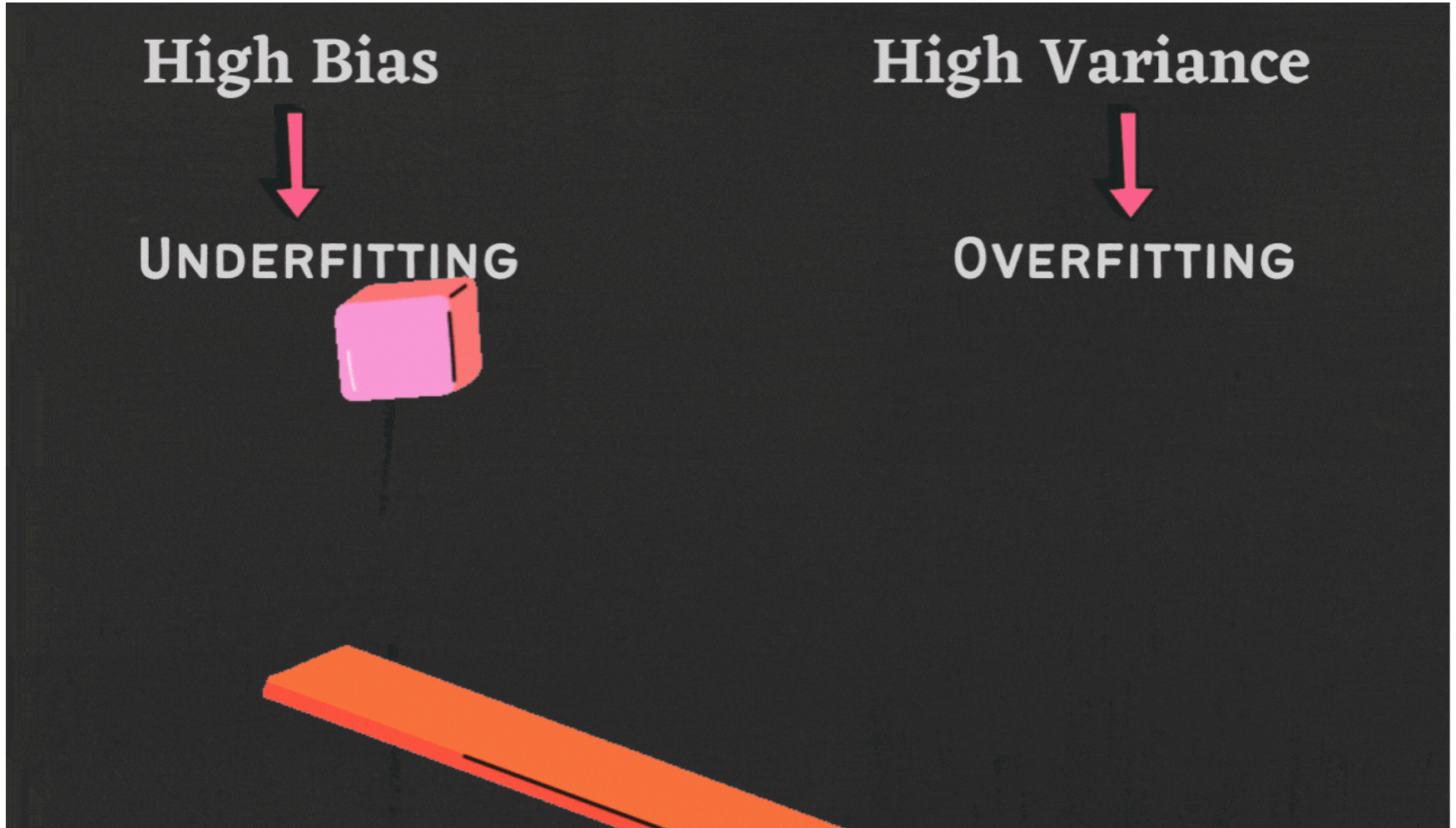
Selección de Algoritmos:

- **Basado en la Naturaleza de los Datos**: La elección del algoritmo a menudo depende de la estructura y tipo de datos disponibles, así como del objetivo específico del modelo.
- **Compromiso entre Precisión y Complejidad**: Consideraciones sobre el tiempo de entrenamiento y los recursos computacionales disponibles, especialmente relevante en entornos de producción.
- **Prejuicios e Imparcialidad**: Seleccionar algoritmos que minimicen el riesgo de sesgo y promuevan resultados justos y éticos.

Fundamentos del Desarrollo de Modelos

Importancia del Diseño de Datos:

- **Calidad de Datos**: Asegurar la calidad de los datos es primordial, ya que los modelos de ML dependen directamente de la precisión y relevancia de los datos alimentados.
- **Preparación de Datos**: Incluye limpieza de datos, manejo de datos faltantes, y transformaciones necesarias para adecuar los datos a las necesidades del algoritmo seleccionado.
- **Ingeniería de Características**: El proceso de usar el conocimiento del dominio para seleccionar, modificar o crear nuevas características que aumenten la capacidad predictiva del modelo.


High Bias
UNDERFITTING
High Variance
OVERFITTING

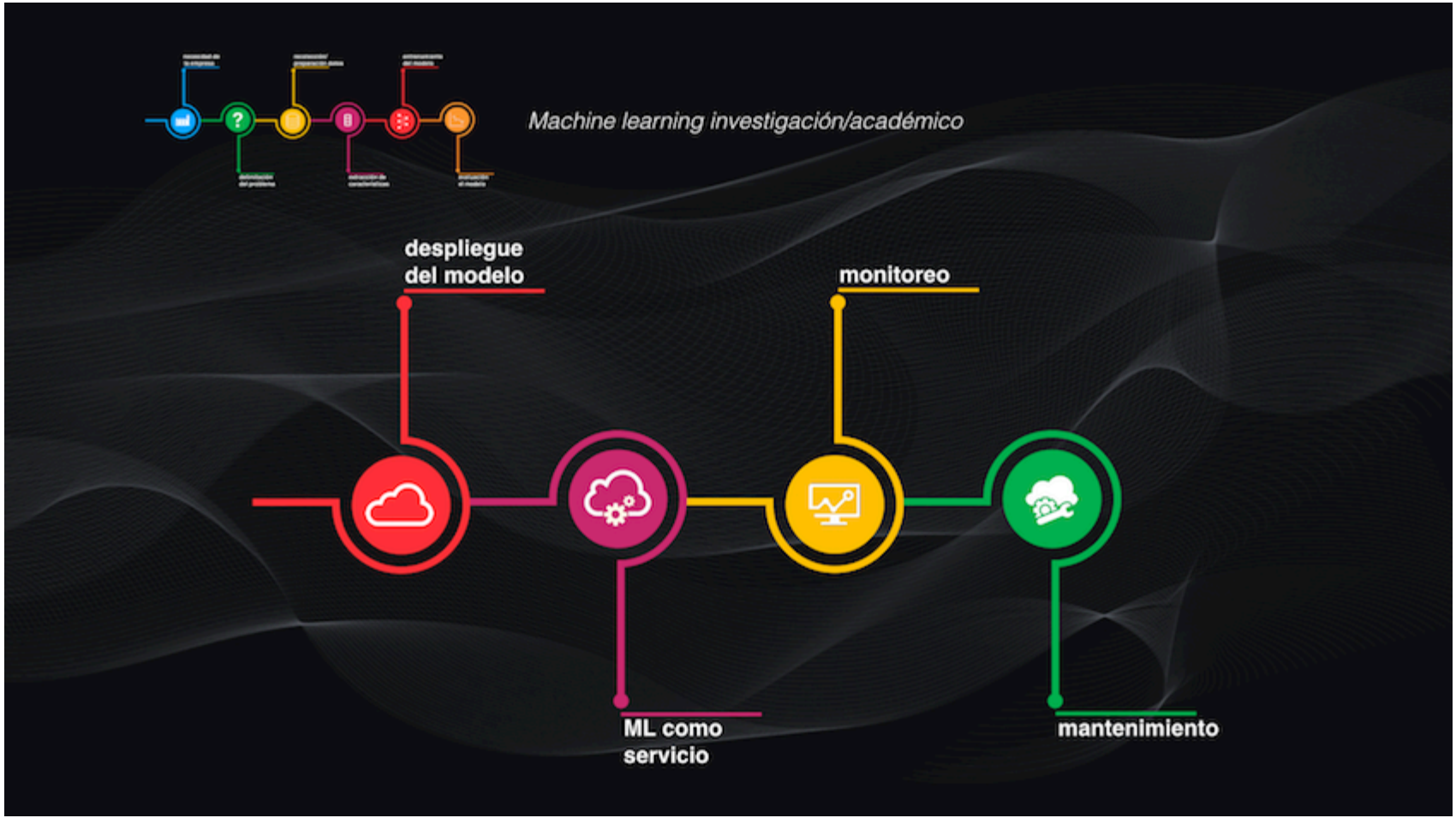
Fundamentos del Desarrollo de Modelos

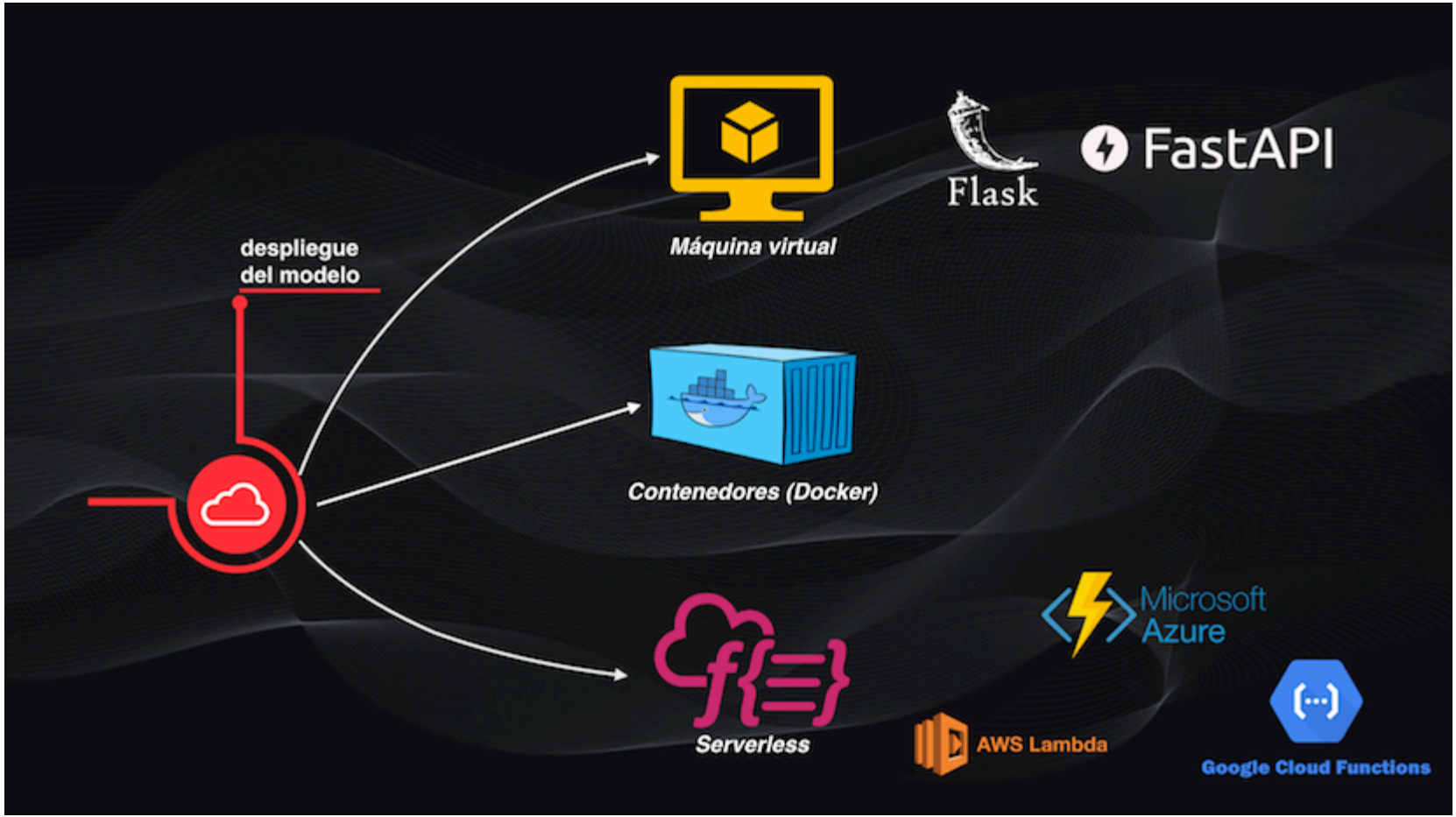
Desafíos Comunes:

- **Sobreajuste y Subajuste**: Balancear la complejidad del modelo para asegurar una buena generalización en datos no vistos.
- **Validación y Pruebas**: Implementación de estrategias robustas para evaluar la efectividad del modelo, como la validación cruzada y la evaluación basada en nuevos conjuntos de datos.

Diseño de Modelos para Robustez y Escalabilidad

- **Principios de Diseño Robusto**:
  - Diseñar con la expectativa de cambios en los patrones de datos y volúmenes.
  - Incorporar mecanismos para la actualización fácil y rápida del modelo sin degradar el rendimiento.
- **Escalabilidad en MLOps**:
  - Utilizar arquitecturas de microservicios o contenedores para facilitar la escalabilidad horizontal de los modelos.
  - Emplear técnicas de procesamiento en paralelo y distribuido para manejar grandes volúmenes de datos.
- **Consideraciones Prácticas**:
  - Validar el modelo con diferentes configuraciones de infraestructura.
  - Optimizar el código del modelo para aprovechar hardware específico (como GPUs).


Machine learning investigación/académico
despliegue del modelo
monitoreo
ML como servicio
mantenimiento

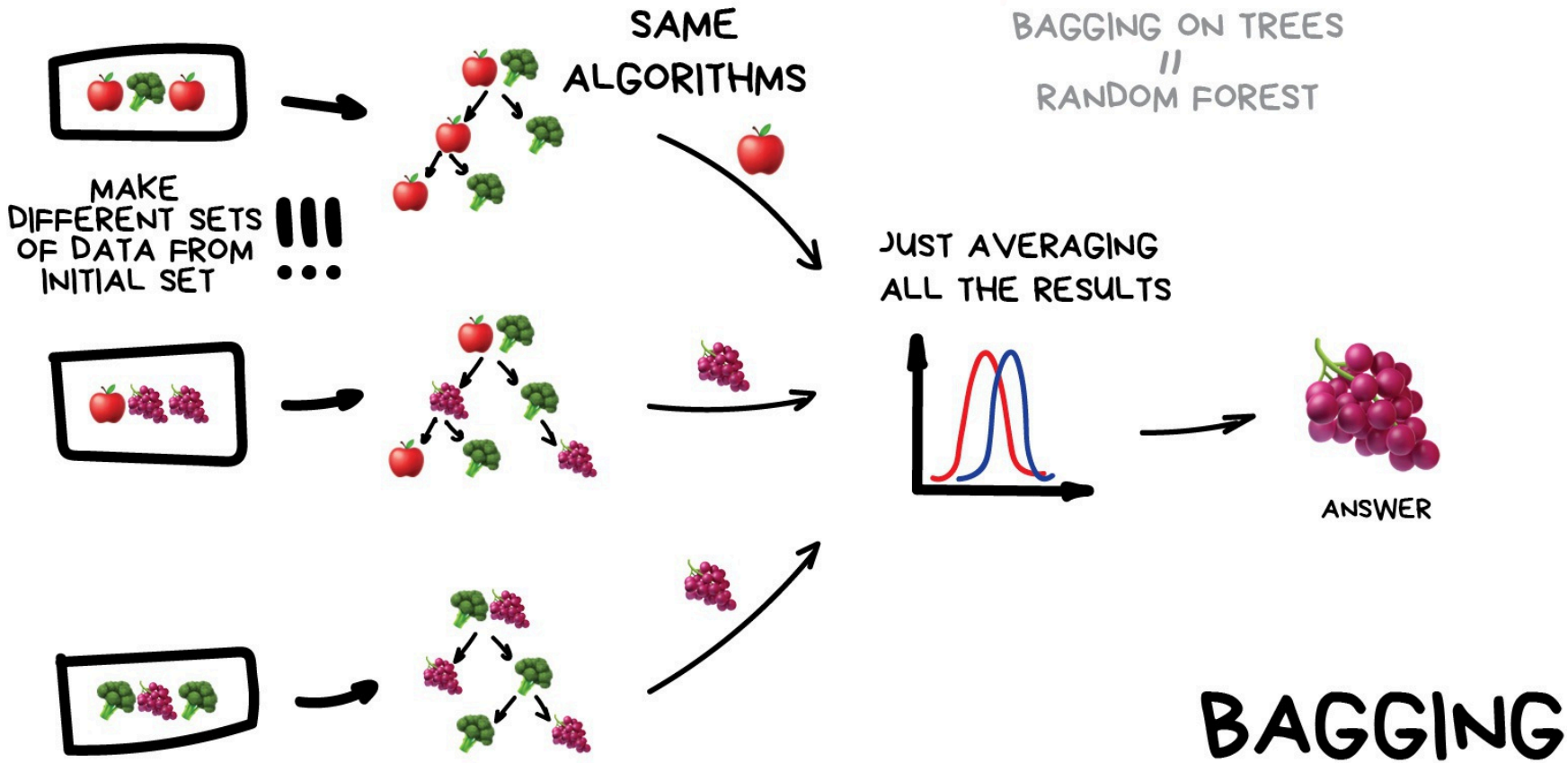

despliegue
del modelo
Máquina virtual
Flask
FastAPI
Contenedores (Docker)
Serverless
Microsoft
Azure
AWS Lambda
Google Cloud Functions

Técnicas de Ensamble y Regularización

- **Técnicas de Ensamble**:
  - Explicar el uso de métodos como Bagging, Boosting y Stacking para mejorar la precisión y estabilidad de los modelos.
  - Ejemplos incluyen Random Forests para bagging y AdaBoost para boosting.
- **Regularización**:
  - Introducir conceptos de regularización como L1 (Lasso) y L2 (Ridge) para prevenir el sobreajuste.
  - Discutir cómo la regularización modifica la función de pérdida para penalizar modelos demasiado complejos.
- **Beneficios de Estas Técnicas**:
  - Reducir la varianza sin aumentar considerablemente el sesgo.
  - Mejorar la generalización del modelo a nuevos datos.

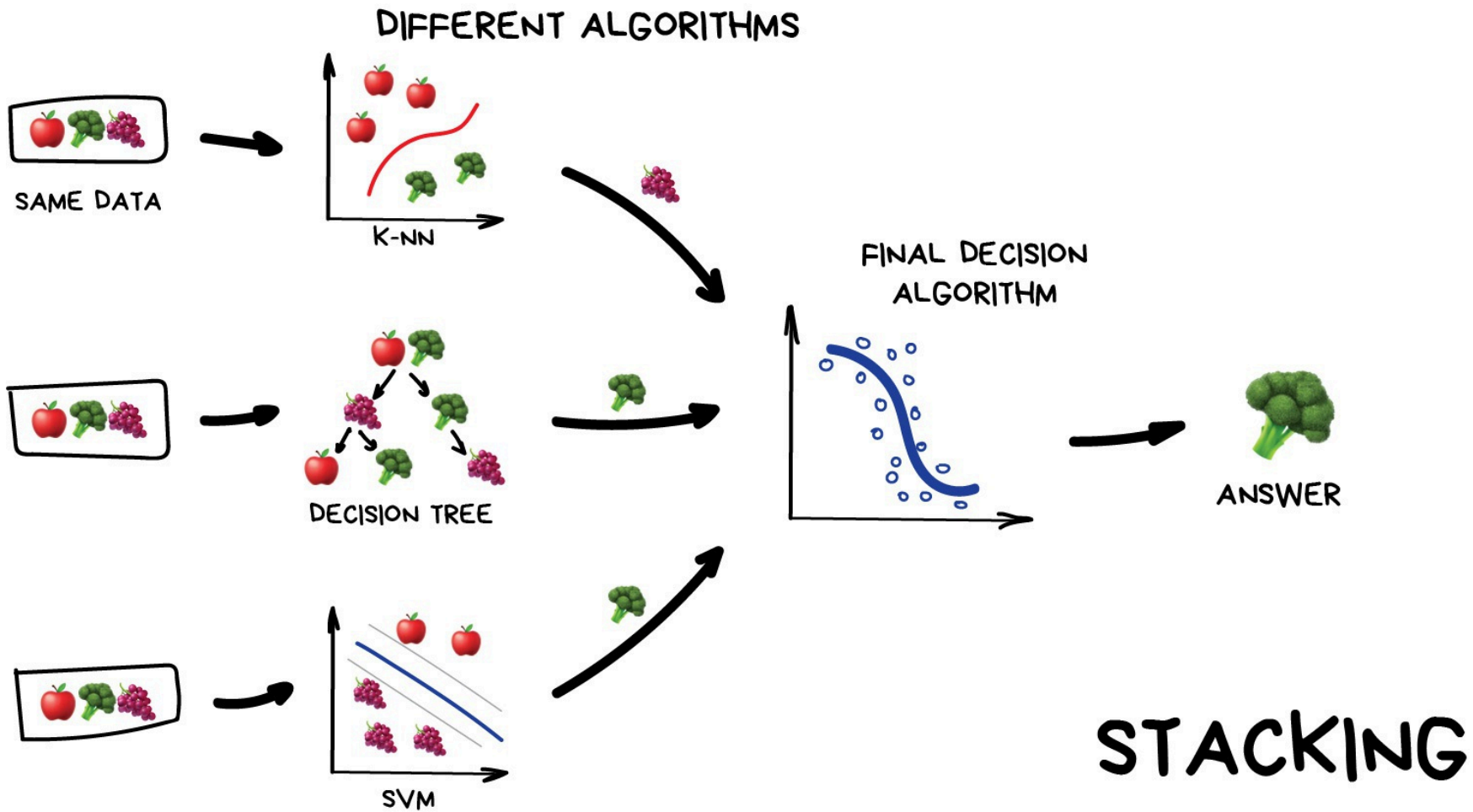
Técnicas de Ensamble - Bagging

- **¿Qué es Bagging?**:
  - Bagging, o Bootstrap Aggregating, es una técnica de ensamble que mejora la estabilidad y precisión de los modelos de machine learning. Consiste en crear múltiples versiones de un predictor (como un árbol de decisión) y utilizar sus predicciones agregadas para formar la predicción final.
- **Cómo Funciona**:
  - Se generan múltiples conjuntos de datos de entrenamiento al azar mediante muestreo con reemplazo (bootstrap).
  - Un modelo separado se entrena con cada uno de estos conjuntos.
  - La predicción final se realiza por mayoría de votos (para clasificación) o promedio (para regresión).
- **Ejemplo Práctico - Random Forest**:
  - Random Forest es una implementación popular de bagging donde se utilizan múltiples árboles de decisión para obtener una predicción robusta y menos propensa al sobreajuste.


SAME ALGORITHMS
BAGGING ON TREES
=
RANDOM FOREST
MAKE DIFFERENT SETS OF DATA FROM INITIAL SET
!!!
JUST AVERAGING ALL THE RESULTS
ANSWER
BAGGING

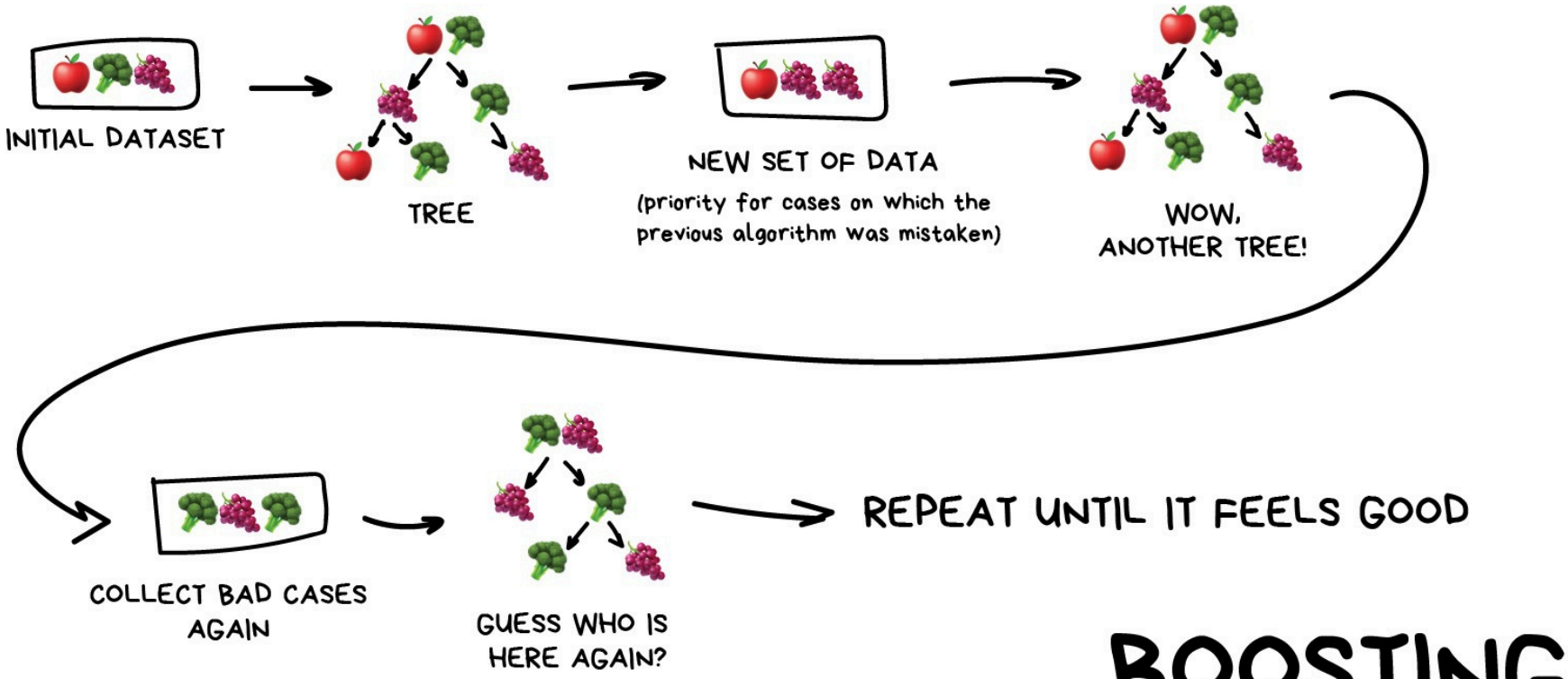
Técnicas de Ensamble - Stacking

- **¿Qué es Stacking?**:
  - Stacking (apilamiento) es una técnica de ensamble que combina las predicciones de múltiples modelos de machine learning para producir una predicción final. Stacking utiliza un modelo de aprendizaje (meta-modelo) para realizar la predicción final basándose en las predicciones de varios modelos base.
- **Cómo Funciona**:
  - Varios modelos base, que pueden ser de diferentes tipos (por ejemplo, árboles de decisión, regresiones, máquinas de soporte vectorial), son entrenados con el conjunto de datos completo.
  - Las predicciones de estos modelos base se utilizan como entradas (junto con los datos originales, en algunos casos) para entrenar un meta-modelo que hace la predicción final.
- **Beneficios del Stacking**:
  - Combina las fortalezas y reduce las debilidades de los modelos individuales.


DIFFERENT ALGORITHMS
SAME DATA
K-NN
DECISION TREE
SVM
FINAL DECISION ALGORITHM
ANSWER
STACKING

Técnicas de Ensamble - Boosting

- **¿Qué es Boosting?**:
  - Boosting es un método de ensamble que combina múltiples modelos débiles secuenciales para formar un modelo fuerte. A diferencia de bagging, boosting busca mejorar el error de los modelos anteriores en la secuencia.
- **Cómo Funciona**:
  - Se entrena el primer modelo en el conjunto de datos y se identifican los errores.
  - Los modelos subsiguientes se entrenan poniendo más énfasis en los casos difíciles o errores cometidos por los modelos anteriores.
  - La predicción final se hace basada en una suma ponderada de las predicciones de todos los modelos.
- **Ejemplo Práctico - AdaBoost**:
  - AdaBoost (Adaptive Boosting) es uno de los algoritmos de boosting más utilizados. Comienza asignando pesos iguales a todas las instancias de entrenamiento y aumenta los pesos de las instancias


INITIAL DATASET
TREE
NEW SET OF DATA
(priority for cases on which the previous algorithm was mistaken)
WOW, ANOTHER TREE!
COLLECT BAD CASES AGAIN
GUESS WHO IS HERE AGAIN?
REPEAT UNTIL IT FEELS GOOD
BOOSTING

Evitando el Sobreajuste

- **Identificación del Sobreajuste**:
  - Señales de que un modelo está sobreajustado incluyen un excelente rendimiento en el conjunto de entrenamiento pero un rendimiento pobre en el conjunto de validación o prueba.
- **Estrategias para Prevenir el Sobreajuste**:
  - Utilización de técnicas de validación como la validación cruzada para evaluar la generalización del modelo.
  - Aplicación de técnicas de regularización y reducción de la complejidad del modelo.
  - Incremento de los datos de entrenamiento o uso de técnicas de aumento de datos.
- **Monitoreo Continuo**:
  - Establecer sistemas de monitoreo para evaluar el rendimiento del modelo de forma continua y detectar signos tempranos de sobreajuste.

Evaluación de Modelos

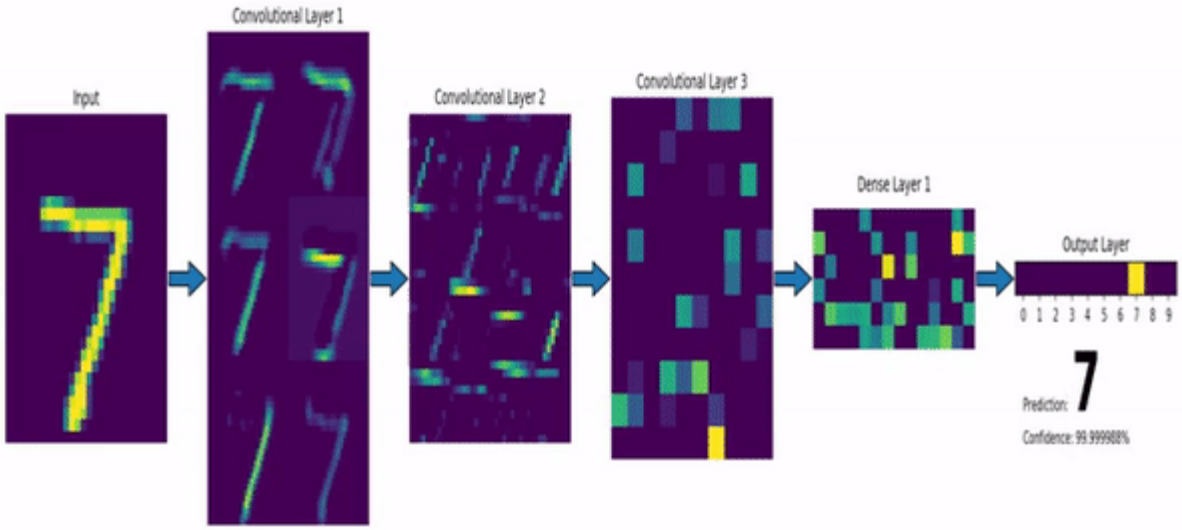
Validación Cruzada y Métricas de Rendimiento

- **Validación Cruzada**:
  - Es una técnica para evaluar la capacidad de generalización de un modelo, útil especialmente en situaciones donde los datos son limitados.
  - Implica dividir el conjunto de datos en múltiples subconjuntos (folds) y realizar entrenamientos iterativos, donde cada fold se usa una vez como conjunto de prueba mientras los restantes constituyen el conjunto de entrenamiento.
  - Las versiones comunes incluyen la validación cruzada k-fold y la validación cruzada de Leave-One-Out (LOO).

- **Métricas de Rendimiento**:
  - **Clasificación**: Precisión, Recall, F1-Score, AUC-ROC.
  - **Regresión**: Error Cuadrático Medio (MSE), Error Absoluto Medio (MAE), Coeficiente de Determinación (R^2).

Importancia de la Interpretación y Explicabilidad

- **Interpretación de Modelos**:
  - Refiere a la capacidad de describir los mecanismos internos y las decisiones de un modelo en términos comprensibles para los humanos.
  - Es crucial para validar la corrección de los modelos, especialmente en áreas sensibles como el crédito, la salud y la justicia.
- **Explicabilidad**:
  - Abarca las técnicas y métodos utilizados para hacer transparentes las decisiones de los modelos, como LIME, SHAP, y diagramas de árbol de decisión.
  - La explicabilidad no solo ayuda a ganar confianza entre los usuarios sino que también facilita el cumplimiento de normativas como GDPR, que exige una "explicación" de las decisiones automatizadas.


Input
Convolutional Layer 1
Convolutional Layer 2
Convolutional Layer 3
Dense Layer 1
Output Layer
0 1 2 3 4 5 6 7 8 9
Prediction:
7
Confidence: 99.999988%

- **Desafíos y Soluciones**:
  - Muchos modelos avanzados, especialmente los basados en aprendizaje profundo, son inherentemente opacos (cajas negras).
  - Trabajar hacia modelos más interpretables y explicables sin sacrificar el rendimiento es uno de los desafíos clave en la ciencia de datos moderna.

Unidad 5: Pruebas y Validación de Modelos de ML

Profundizaremos en cómo validar y probar modelos de machine learning para asegurar su eficacia y fiabilidad.

Objetivos de la Unidad

- Entender y aplicar diversas técnicas de **validación** para modelos de ML.
- Aprender a implementar y evaluar **pruebas robustas** de modelos.
- Utilizar **métricas de rendimiento** para evaluar la eficacia de los modelos.
- Explorar la importancia de la **interpretación y explicabilidad** en modelos de ML.

Introducción a las Pruebas y Validación

La validación de modelos es crucial para:

- Evaluar la **capacidad de generalización** de los modelos.
- Asegurar que el rendimiento del modelo es **consistente y fiable**.
- Identificar y mitigar problemas antes de la implementación en producción.

Técnicas de Validación Cruzada

- **K-fold Cross Validation**
- **Leave-One-Out (LOO)**
- **Time Series Cross Validation**

Estas técnicas nos ayudan a entender cómo el modelo se comportará con **datos no vistos**.

K-fold Cross Validation

- Divide el conjunto de datos en **K subconjuntos** (o folds).
- Cada fold se utiliza una vez como conjunto de prueba mientras los otros K-1 actúan como conjunto de entrenamiento.
- Este proceso se repite K veces, con cada fold usado exactamente una vez como conjunto de prueba.

Ventajas:

- Todas las observaciones se utilizan para entrenamiento y prueba.
- Reduce la variabilidad de un solo ensayo de la división train-test.

Uso típico:

- Ideal para conjuntos de datos de tamaño moderado y cuando se busca un balance entre

Leave-One-Out Cross Validation (LOO)

- Una forma especial de cross validation donde **K (número de folds) es igual al número de datos** en el conjunto.
- En cada iteración, se deja un solo dato como el conjunto de prueba y el resto como el conjunto de entrenamiento.
- Esto se repite de manera que cada dato se usa exactamente una vez como prueba.

Ventajas:

- Maximiza el tamaño del conjunto de entrenamiento.
- Útil para obtener estimaciones detalladas y menos sesgadas del rendimiento del modelo.

Uso típico:

- Adecuado para conjuntos de datos pequeños donde la exclusión de cualquier dato podría llevar a sesgos significativos en el entrenamiento.

Time Series Cross Validation

- Variante específica diseñada para datos secuenciales (temporales).
- No aleatoriza los datos, preservando el orden cronológico.
- Divide los datos en folds que respetan la secuencia temporal, generalmente expandiendo la ventana de entrenamiento en cada iteración.

Ventajas:

- Mantiene la estructura temporal, crucial para la validez del modelo en series temporales.
- Permite evaluar cómo se desempeñará el modelo a medida que se dispone de más datos a lo largo del tiempo.

Uso típico:

- Ideal para proyecciones financieras, análisis económico, pronósticos meteorológicos y cualquier situación donde la secuencia temporal es crítica.

Métricas de Evaluación de Modelos

- **Clasificación**: Precisión, Recall, F1-Score, AUC-ROC.
- **Regresión**: MSE, RMSE, MAE, R^2.

Elegir la métrica correcta es esencial para alinear el rendimiento del modelo con los **objetivos del negocio**.

Métricas de Evaluación para Modelos de Clasificación

- **Precisión (Accuracy)**: Proporción de predicciones correctas respecto al total de casos. Útil cuando las clases están equilibradas.
- **Recall (Sensibilidad)**: Capacidad del modelo de identificar todas las instancias relevantes. Esencial en casos como la detección de enfermedades.
- **F1-Score**: Media armónica de la precisión y el recall. Importante cuando se busca un balance entre precisión y recall.
- **AUC-ROC**: Área bajo la curva del Receiver Operating Characteristic. Mide la capacidad del modelo para distinguir entre clases.

Estas métricas ayudan a evaluar la efectividad de los modelos de clasificación, cada una desde un enfoque diferente, permitiendo ajustar el modelo según lo que sea más crítico para el problema en cuestión.

Métricas de Evaluación para Modelos de Regresión

- **MSE (Mean Squared Error)**: Promedio de los cuadrados de los errores. Penaliza fuertemente los errores grandes.
- **RMSE (Root Mean Squared Error)**: Raíz cuadrada del MSE. Proporciona una medida del error en las mismas unidades de la variable de respuesta.
- **MAE (Mean Absolute Error)**: Promedio de los valores absolutos de los errores. Menos sensible a los valores atípicos en comparación con el MSE.
- **R^2 (Coeficiente de Determinación)**: Proporción de la variabilidad total de la respuesta que es explicada por el modelo.

Estas métricas son cruciales para entender cómo de bien un modelo de regresión se ajusta a los datos observados, y cada una ofrece una perspectiva diferente sobre los errores del modelo.

Pruebas de Robustez y Simulación

Implementamos pruebas que simulan:

- Condiciones extremas y raras.
- Entradas inesperadas o ruidosas.

Estas pruebas aseguran que el modelo es **robusto y fiable**.

Interpretación y Explicabilidad del Modelo

Herramientas como **LIME** y **SHAP** ayudan a:

- Mejorar la transparencia de los modelos.
- Facilitar el cumplimiento de normativas como GDPR.

La explicabilidad es vital para la **aceptación y confianza** en los modelos de ML.