

De la física al aprendizaje automático: Aplicaciones en la predicción y optimización de procesos industriales

Mitchell Mirano

Facultad de Ciencias Fisicas – UNMSM

21 de noviembre de 2025

- 1 Fundamentos de Inteligencia Artificial
- 2 Casos de Optimización de Procesos Industriales
- 3 Física y Machine Learning

1. Inteligencia Artificial (IA)

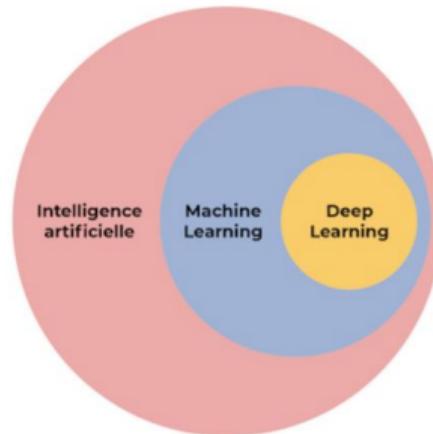
- Disciplina general que busca desarrollar sistemas capaces de razonar, planificar o tomar decisiones.
- Ejemplos: planificación automática, agentes inteligentes, lógica de primer orden.

2. Aprendizaje Automático (Machine Learning)

- Subcampo de la IA basado en modelos matemáticos que **aprenden patrones a partir de datos**.
- Requiere un conjunto de entrenamiento y un mecanismo de optimización.

3. Deep Learning / Redes Neuronales

- Variante del aprendizaje automático basada en arquitecturas de múltiples capas.
- Capaz de modelar **relaciones no lineales complejas** en espacios de alta dimensión.



Le Deep Learning est une sous-discipline du Machine Learning

Figure: IA ⊂ ML ⊂ Deep Learning

Paradigmas del Aprendizaje Automático

¿Cómo aprenden las máquinas a partir de los datos?

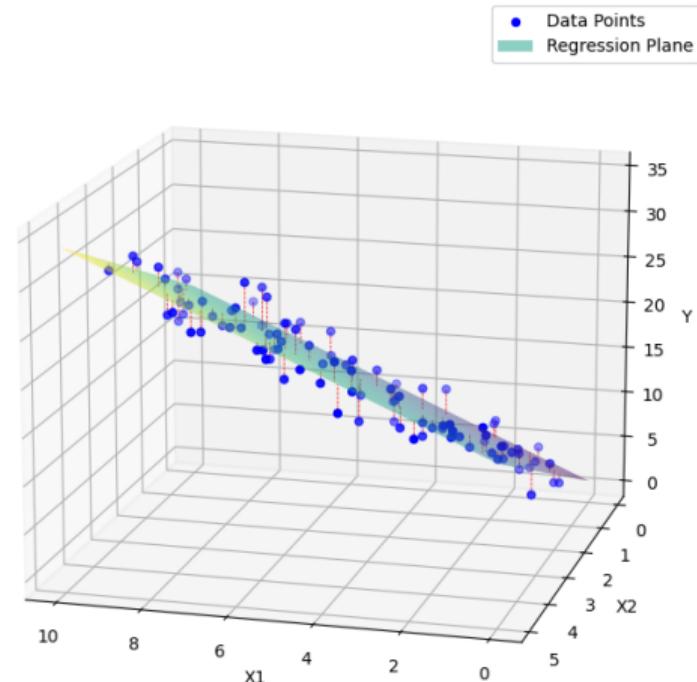
Aprendizaje Supervisado: Regresión

Definición

Objetivo: Predecir una **variable continua**, es decir, un número real dentro de un rango.

Aplicaciones Comunes

- **Finanzas:** Estimación de precios de vivienda o acciones.
- **Medio Ambiente:** Predicción de la temperatura o niveles de polución.
- **Comercio:** Proyecciones detalladas de ventas futuras.



Aprendizaje Supervisado: Clasificación

Definición

Objetivo: Asignar una instancia a una **categoría o clase discreta** predefinida.

Aplicaciones Comunes

- **Seguridad:** Detección de *spam* (Clase A) vs. *No-spam* (Clase B).
- **Medicina:** Diagnóstico binario (enfermo/sano).
- **Visión:** Reconocimiento de dígitos o de objetos en imágenes.

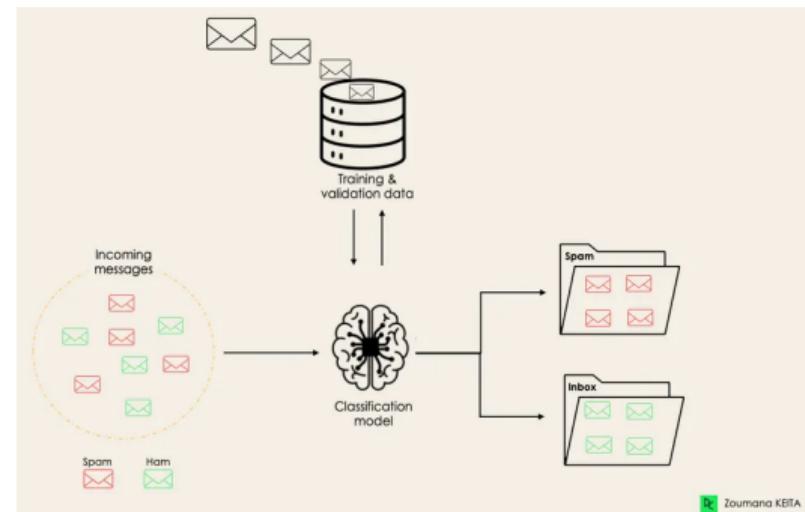


Figure: Separación de datos en clases discretas.

Aprendizaje No Supervisado: Clustering (Agrupamiento)

Definición

Objetivo: Identificar **grupos o estructuras naturales** en los datos. Es un problema de Aprendizaje **No Supervisado**.

Aplicaciones Comunes

- **Marketing:** Segmentación de clientes basada en comportamiento de compra.
- **Investigación:** Agrupamiento automático de documentos o artículos científicos.
- **Análisis de Datos:** Detección de anomalías o *outliers*.

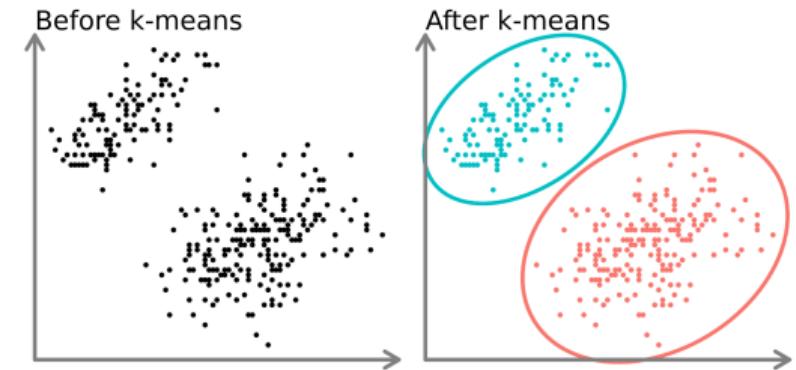


Figure: Identificación de estructuras sin etiquetas previas.

Aprendizaje por Refuerzo (RL)

Definición

Objetivo: Entrenar un **agente** para tomar decisiones secuenciales en un entorno, maximizando una **recompensa** acumulada.

Aplicaciones Comunes

- **Robótica:** Control y navegación autónoma.
- **Finanzas:** Optimización de carteras de inversión.
- **Juegos:** Desarrollo de IA que supera a humanos (e.g., AlphaGo).

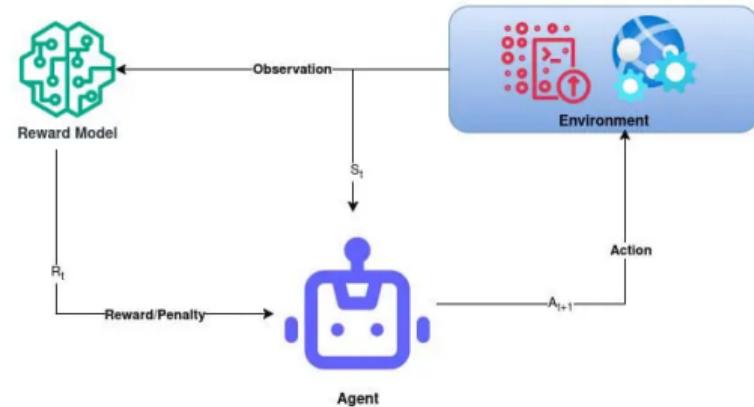


Figure: Diagrama del ciclo de interacción en RL.

Algoritmos Comunes por Paradigma de Aprendizaje

Aprendizaje Supervisado

- Regresión

- Regresión Lineal (OLS)
- Árboles de Decisión (CART)
- LightGBM / XGBoost
- Redes Neuronales (MLP)
- Support Vector Regression (SVR)

- Clasificación

- Regresión Logística
- SVM (Support Vector Machine)
- Random Forest / Gradient Boosting
- LightGBM / XGBoost
- Redes Neuronales (Softmax)
- Transformers (para clasificación avanzada)

Aprendizaje No Supervisado

- Clustering

- K-means
- DBSCAN
- Gaussian Mixture Models (GMM)

- Reducción de Dimensión

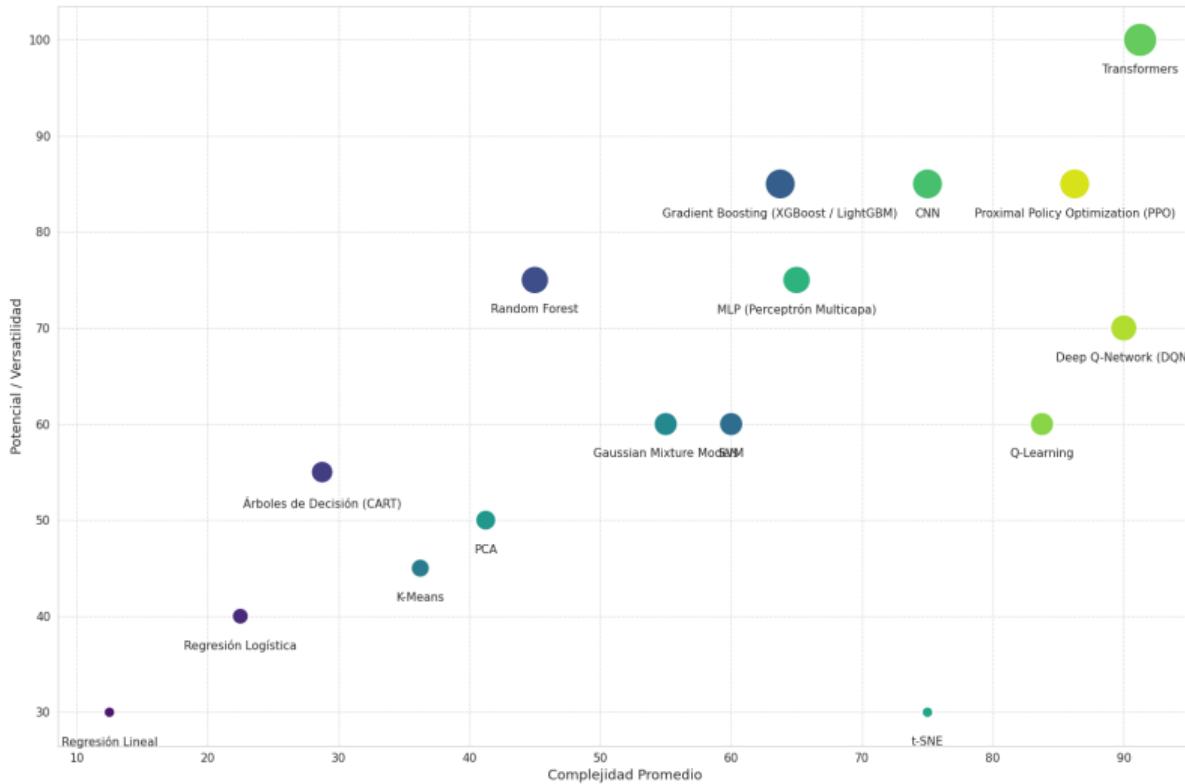
- PCA
- t-SNE
- Autoencoders

Aprendizaje por Refuerzo

- Q-Learning
- Deep Q-Networks (DQN)
- Proximal Policy Optimization (PPO)

Comparativa de Complejidad vs Potencial de los Algoritmos

Algoritmos de Machine Learning: Complejidad vs. Potencial/Versatilidad (Etiquetas Mejoradas)



ML como un Problema de Optimización

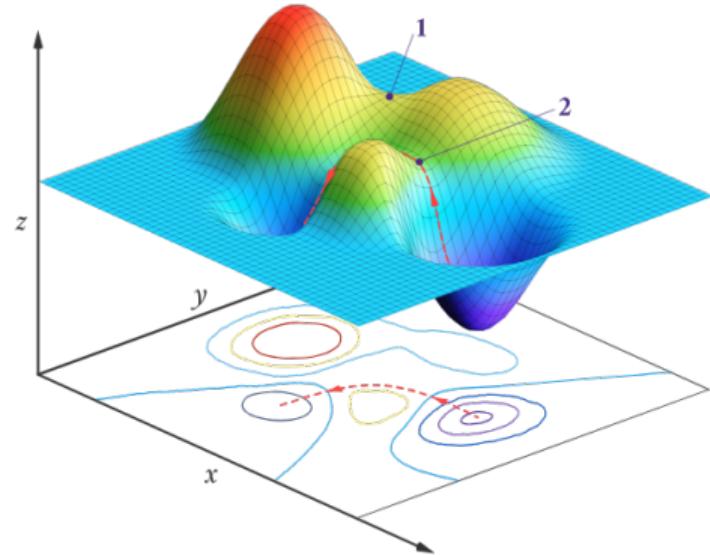
El aprendizaje automático se formula como la minimización de una **función de pérdida** $\mathcal{L}(y, \hat{y})$, la cual cuantifica la discrepancia entre la predicción del modelo $\hat{y} = f(x; \theta)$ y el valor observado y :

$$\theta^* = \arg \min_{\theta} \mathcal{L}(y, f(x; \theta))$$

donde θ representa los parámetros del modelo y $f(x; \theta)$ la función predictiva.

Analogía física:

- La función de pérdida actúa como el *potencial del sistema*.
- El aprendizaje equivale a una búsqueda hacia el **estado de mínima energía**, análogo al equilibrio termodinámico.



Casos de optimización de procesos industriales

¿Que problemas pueden ser resueltos con ML?

Caso 1:ARPL - Auto Service Desk

Los colaboradores de las empresas del Grupo UNACEM solicitan soporte técnico y soluciones informáticas mediante correos enviados a la mesa de servicio. Estos mensajes son registrados en la plataforma KACE y posteriormente asignados por un analista a un especialista de ARPL, según la naturaleza del incidente.

Limitaciones del proceso actual:

- La asignación se realiza de forma manual, generando tiempos de espera promedio de **30 minutos por caso**.
- El analista frecuentemente se encuentra atendiendo otras actividades, lo que incrementa los tiempos de respuesta.
- El alto volumen de correos diarios retrasa el análisis y clasificación.
- Según la criticidad del incidente, la atención debe ejecutarse en un rango de **4 a 8 horas hábiles** para no afectar la continuidad operativa.

Este contexto evidencia la necesidad de automatizar la etapa de análisis y asignación de solicitudes para mejorar la eficiencia y reducir tiempos de atención.

Diagrama del Problema



Solución con ML(Clasificación de texto)

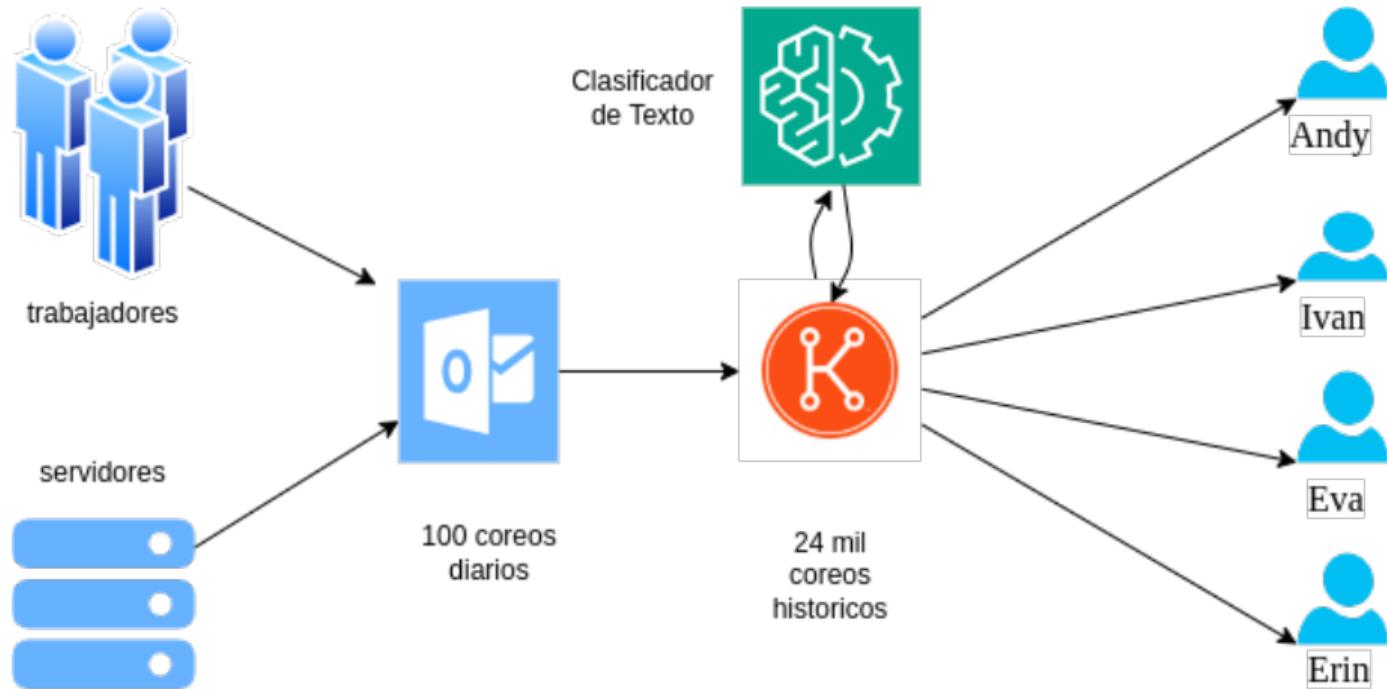


Figure: Cada 5 minutos se buscan correos en la plataforma KACE y se clasifican.

Clasificador de Texto(Vectorización)

Utilizamos el modelo pre-entrenado de embeddings de Hugging Face

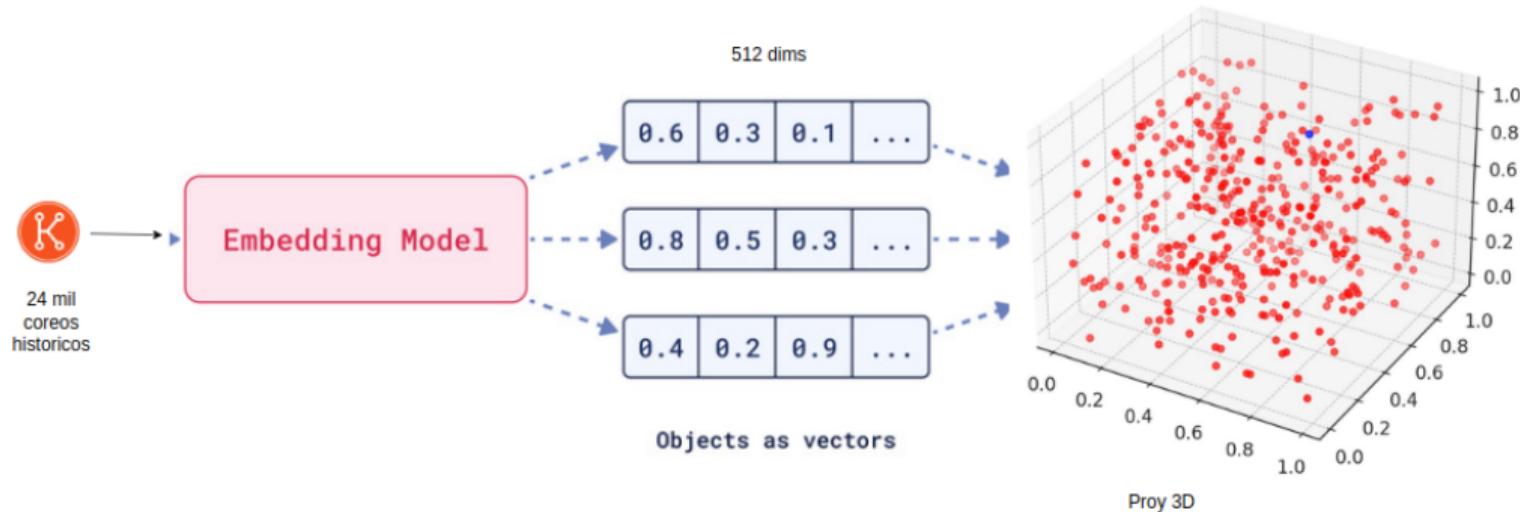


Figure: Convertimos los 24 mil correos historicos en vectores de 512 dimensiones.

Clasificador de Texto(Clasificación de vectores)

Entrenamos una red neuronal que identifique quien va a atender el correo en base a los vectores de 512 dimensiones.

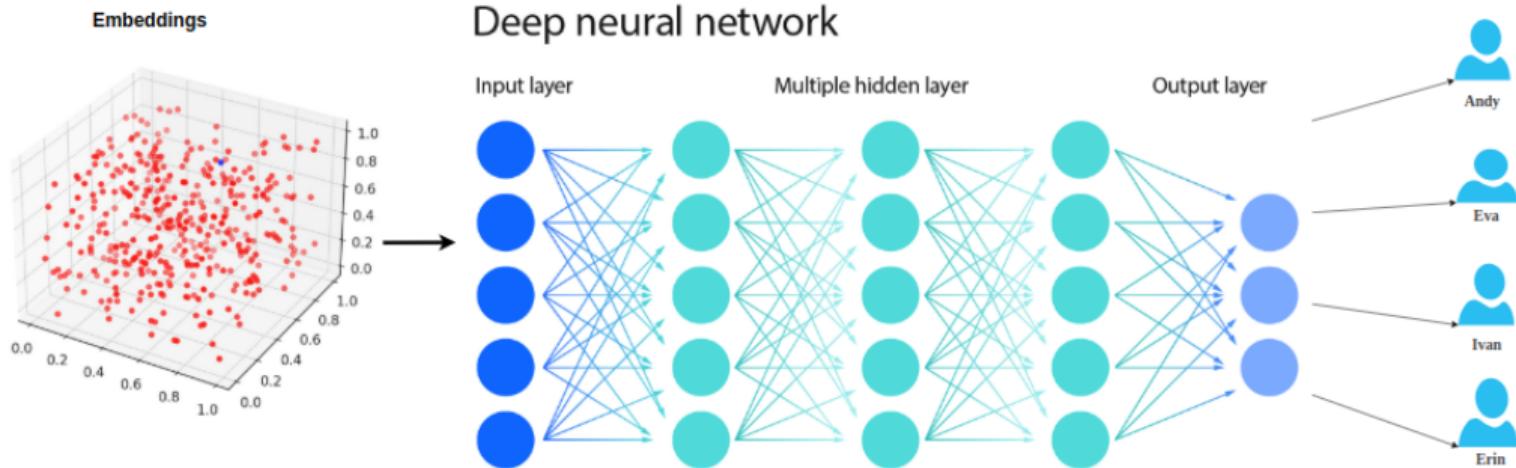


Figure: La red neuronal identifica quien va a atender el correo.

Neurona Artificial: Definición e Inspiración

Definición

Es la unidad fundamental de las Redes Neuronales, diseñada para simular el proceso de **activación e inhibición** de las neuronas biológicas. Su función principal es recibir señales y producir una única salida.

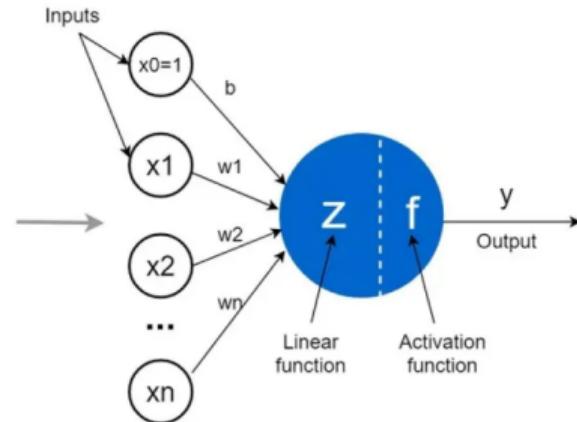


Figure: Paralelo entre neurona biológica y unidad artificial.

Suma Ponderada (z): Fundamento de la Neurona

Definición Vectorial

La activación interna o señal integrada (z) se calcula como el **producto escalar** entre el vector de pesos $\mathbf{w} \in \mathbb{R}^n$ y el vector de entrada $\mathbf{x} \in \mathbb{R}^n$, más un término escalar de sesgo ($b \in \mathbb{R}$):

$$\hat{y} = z = \mathbf{w}^\top \mathbf{x} + b$$

Interpretación Matemática:

- z representa una **combinación lineal** de las entradas ponderadas por sus pesos.
- El sesgo b introduce un desplazamiento adicional que evita la restricción de pasar por el origen.
- Este modelo constituye la base de toda red neuronal feed-forward.

Neurona Artificial: Formulación Matricial (I)

operación matricial den dentro de una para $\mathbf{X}(\text{datos}) \in \mathbb{R}^{n \times m}$

$$\hat{\mathbf{y}} = \mathbf{z} = \mathbf{X}\mathbf{w} + \mathbf{b}$$

$$\begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_n \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1m} \\ x_{21} & x_{22} & \dots & x_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{nm} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_m \end{bmatrix} + \begin{bmatrix} b \\ b \\ \vdots \\ b \end{bmatrix}$$

En nuestro caso $n=24\ 000$ correos y $m=512$ dimensiones de los embeddings.

Capa de una Red Neuronal: Formulación Matricial (II)

Calculos en una capa de una red neuronal para $\mathbf{X}(\text{datos}) \in \mathbb{R}^{n \times m}$ con k neuronas por tanto $\mathbf{W}(\text{pesos}) \in \mathbb{R}^{m \times k}$ y el vector de sesgos $\mathbf{B}(\text{sesgos}) \in \mathbb{R}^{n \times k}$

$$\begin{bmatrix} z_{11} & z_{12} & \dots & z_{1k} \\ z_{21} & z_{22} & \dots & z_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ z_{n1} & z_{n2} & \dots & z_{nk} \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1m} \\ x_{21} & x_{22} & \dots & x_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{nm} \end{bmatrix} \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1k} \\ w_{21} & w_{22} & \dots & w_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m1} & w_{m2} & \dots & w_{mk} \end{bmatrix} + \begin{bmatrix} b_1 & b_2 & \dots & b_k \\ b_1 & b_2 & \dots & b_k \\ \vdots & \vdots & \ddots & \vdots \\ b_1 & b_2 & \dots & b_k \end{bmatrix}$$

Ahora pasamos \mathbf{Z} por una función de activación $a(\mathbf{Z})$ para obtener la salida final de la capa.

$$\hat{\mathbf{Y}} = a(\mathbf{Z}) = a(\mathbf{XW} + \mathbf{B}) \in \mathbb{R}^{n \times k}$$

Que se convertirá en la entrada de la capa siguiente.

Función de Activación: Sigmoide

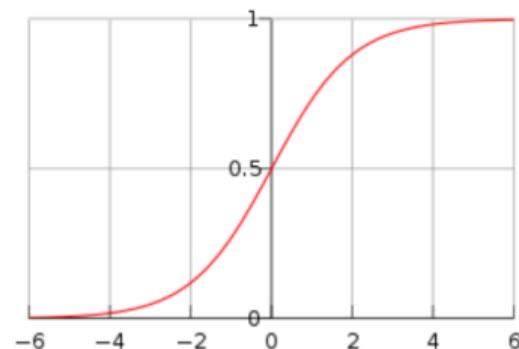
Definición: Función no lineal que comprime la entrada en el rango $(0, 1)$, comúnmente usada en modelos probabilísticos y neuronas binarias.

$$f(x) = \frac{1}{1 + e^{-x}}$$

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Aplicaciones:

- Clasificación binaria $p(y = 1|x)$.
- Capa de salida en regresión logística.
- Modelos donde se requiere una interpretación probabilística.



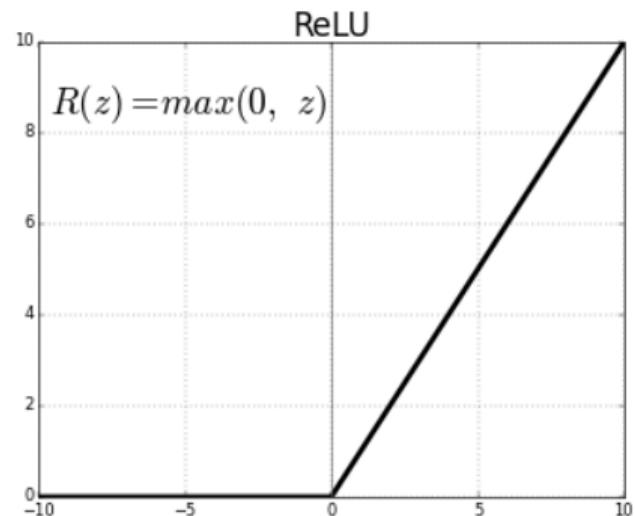
Función de Activación: ReLU

Definición: Activa solo valores positivos de la entrada, anulando los negativos. Introduce no linealidad con bajo costo computacional.

$$\text{ReLU}(x) = \max(0, x)$$

Aplicaciones:

- Capas ocultas en redes neuronales profundas (DNN, CNN).
- Mejora la convergencia del entrenamiento.
- Evita el problema de saturación del gradiente.



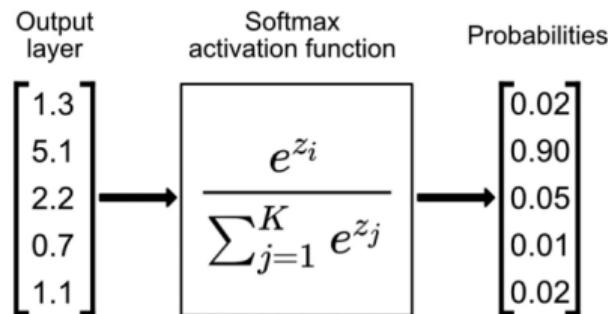
Función de Activación: Softmax

Definición: Convierte un vector de valores reales en una distribución de probabilidad sobre K clases.

$$\text{softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=1}^K e^{x_j}}$$

Aplicaciones:

- Capa de salida en clasificación multiclas.
- Modelos probabilísticos como redes neuronales bayesianas.
- Permite interpretar la salida como $p(y = k|x)$.



Red Neuronal como Composición de Funciones (II)

En forma más explícita, la salida de la red neuronal para una entrada $\mathbf{X} \in \mathbb{R}^{n \times m}$ y c categorías se obtiene aplicando sucesivamente las transformaciones de cada capa:

$$\begin{aligned}\mathbf{X}^{(1)} &= a^{(1)} (\mathbf{Z}^{(1)}) = a^{(1)} (\mathbf{X} \mathbf{W}^{(1)} + \mathbf{b}^{(1)}), \\ &\vdots \\ \mathbf{X}^{(L-1)} &= a^{(L-1)} (\mathbf{Z}^{(L-2)}) = a^{(L-1)} (\mathbf{X}^{(L-2)} \mathbf{W}^{(L-2)} + \mathbf{b}^{(L-2)}), \\ \hat{\mathbf{Y}} &= \text{Softmax} (\mathbf{Z}^{(L)}) = a^{(L)} (\mathbf{X}^{(L-1)} \mathbf{W}^{(L)} + \mathbf{b}^{(L)}) \in \mathbb{R}^{n \times c}.\end{aligned}$$

Por tanto debemos asegurarnos que los pesos de la capa de salida $\mathbf{W}^{(L)} \in \mathbb{R}^{L_x \times c}$
Donde:

- L_x es el numero de neuronas de la cap anterior.
- c es el número de categorías o clases.

Prediciendo quien va a atender el correo

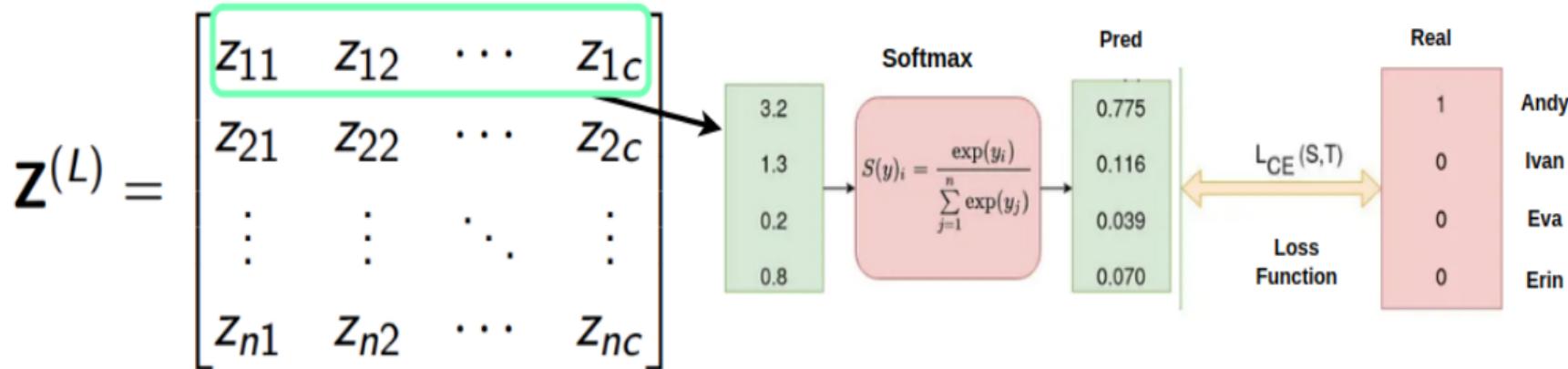


Figure: Quien va a atender el correo es el que tiene mayor probabilidad.

Función de pérdida: Entropía cruzada categórica

Definición (una muestra): dada una distribución objetivo $y = (y_1, \dots, y_K)$ (p.ej. one-hot) y la predicción de la red $p = (p_1, \dots, p_K)$ (resultado de softmax), la *entropía cruzada* se define como

$$\mathcal{L}_{CE}(y, p) = - \sum_{k=1}^K y_k \log p_k.$$

Para un conjunto de n muestras usamos el promedio:

$$\mathcal{L}_{CE} = -\frac{1}{n} \sum_{i=1}^n \sum_{k=1}^K y_{i,k} \log p_{i,k}.$$

Ahora solo queda optimizar \mathcal{L} para encontrar los mejores W (pesos) y B (sesgos).

Retos Encontrados en el Proceso

- Rotación temporal del personal especializado (vacaciones o ausencias).
- Distribución desequilibrada de correos entre especialistas.
- Restricciones en la asignación por empresa (no todos pueden atender todos los casos).
- Cambios de puesto o reasignación de especialistas a nuevas funciones.

Impacto

Estos factores generan sesgos en el modelo de clasificación, pérdida de precisión y riesgo de asignaciones incorrectas o inefficientes.

Soluciones vía ML y datos

- Agrupación de especialistas por perfil funcional (DBA, soporte IT, aplicaciones, ciberseguridad).
- Reentrenamiento del modelo con balanceo de clases para corregir sesgos de asignación.

No todos los requerimientos son satisfechos con soluciones vía ML y datos.

Soluciones vía lógica o software adicional

- Validación de disponibilidad según vacaciones o ausencia.
- Restricción dinámica de especialistas por empresa.

Resultados del Modelo de Clasificación

Eficiencia Operativa

- **Reducción del tiempo de asignación:** de 30 minutos (manual) a **5 minutos** (automatizado). → **83% de mejora en velocidad de respuesta.**

Efectividad del Modelo

- **Precisión de clasificación:** 93% en la asignación del correo al especialista adecuado.
- **Generalización robusta** frente a variaciones lingüísticas en los correos (gracias al uso de embeddings).

Caso 2: UNICON - Optimización de la Programación

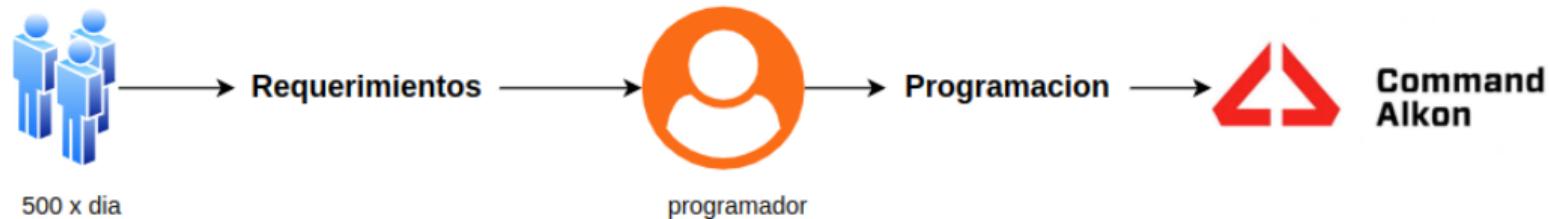
El sistema actual de programación (*Command Series*), encargado de gestionar los pedidos de concreto, no incorpora variables estadísticas del comportamiento del cliente. Actualmente, la programación se basa en tiempos estimados a partir del historial de despachos(promedios) y del volumen solicitado, sin considerar dinámicas operativas reales.

Limitaciones identificadas:

- Los pedidos pueden ser cancelados o modificados (volumen ajustado) sin capacidad predictiva.
- La estimación temporal carece de precisión, generando retrasos en la llegada de los mixers a obra y planta.
- Esto ocasiona una **no atención del 15% del volumen programado**, impactando negativamente en la productividad y la puntualidad con los clientes.

Diagrama del proceso antes del despacho

Cada dia unicon recibe aproximadamente 500 pedidos de concreto. El personal del area de programación recibe los requerimientos de los clientes y los transforma en una programación en el software Command Series.



Los requerimientos del pedido del cliente

- **Volumen** de concreto: 50 m^3 .
- **Fecha** de entrega: 22 de noviembre del 2025.
- **Hora** de inicio del despacho: 9:00 AM.
- **Producto** concreto 230 kg/m^3 (resistencia).
- **Lugar** de entrega: UNMSM - FCF.
- **Espaciamiento** entre mixers: 30 minutos.

La programación del pedido

Para programar el requerimiento primero se debe decidir desde que planta se realizara el despacho: Planta Materiales y cuanto tiempo se tarda en llegar desde la planta a la obra segun google maps.

m^3	Ini carga	Fin carga	A obra	Lleg. obr	Ini desc	Fin desc	A pta	Lleg pta
8	08:15	08:23	08:30	09:00	09:15	09:45	09:55	10:25
8	08:45	08:53	09:00	09:30	09:45	10:15	10:25	10:55
8	09:15	09:23	09:30	10:00	10:15	10:45	10:55	11:25
8	09:45	09:53	10:00	10:30	10:45	11:15	11:25	11:55
8	10:15	10:23	10:30	11:00	11:15	11:45	11:55	12:25
6	10:45	10:53	11:00	11:30	11:45	12:15	12:25	12:55
4	11:15	11:23	11:30	12:00	12:15	12:45	12:55	13:25

Programación de muchos pedidos

Unicon atiende al dia mas de 500 pedidos de concreto. Distribuidos en sus mas de 30 plantas de despacho. El personal de programación debe programar todos los requerimientos y la final se obtiene el siguiente grafico.

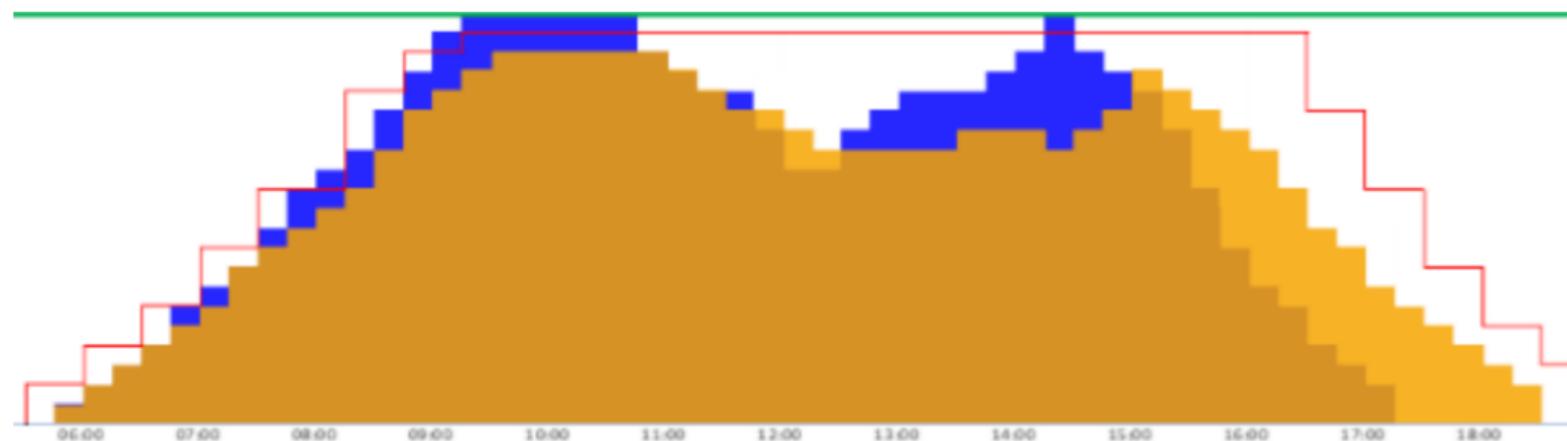


Figure: El grafico muestra cuantos camiones se necesitan cada 30 mins listos para despachar.

Solución con ML (Regresión para predecir cada intervalo de tiempo)

Lo que se buscaba era una solución que permitiera predecir el tiempo de llegada a planta. Para ello entrenamos un modelo especializado en predecir la duración de cada intervalo de tiempo.

Intervalo	Modelo	Algoritmo
Lleg. obr - A obra	Modelo de: Viaje a obra	RandomForestRegressor
Ini desc - Lleg. obr	Modelo de: TEO	RandomForestRegressor
Fin desc - Ini desc	Modelo de: Descarga	RandomForestRegressor
A pta - Fin desc	Modelo de: Salida Obra	RandomForestRegressor
Lleg pta - A pta	Modelo de: Regreso a planta	RandomForestRegressor

Table: Intervalos operativos, modelo asociado y algoritmo recomendado.

Función de pérdida: Error cuadrático medio (MSE)

El **Error Cuadrático Medio (MSE)** es una función de pérdida utilizada en problemas de regresión para cuantificar la discrepancia entre los valores reales y las predicciones del modelo. Se define como:

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

donde:

- y_i es el valor observado,
- \hat{y}_i es el valor predicho por el modelo,
- n es el número total de observaciones.

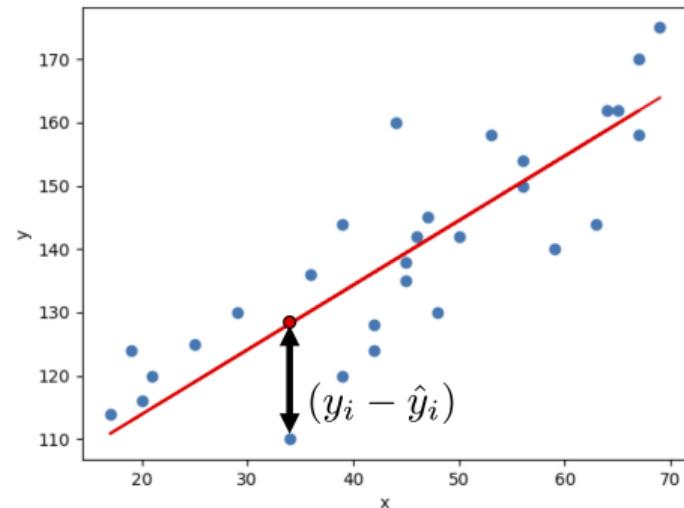


Figure: Representación gráfica del error cuadrático

El Árbol de Regresión

Un **árbol de regresión** divide recursivamente el espacio de variables predictoras con el objetivo de minimizar el error cuadrático medio (MSE). En cada iteración, se evalúa un punto de partición s y se selecciona aquel que produce la menor suma de errores dentro de cada región.

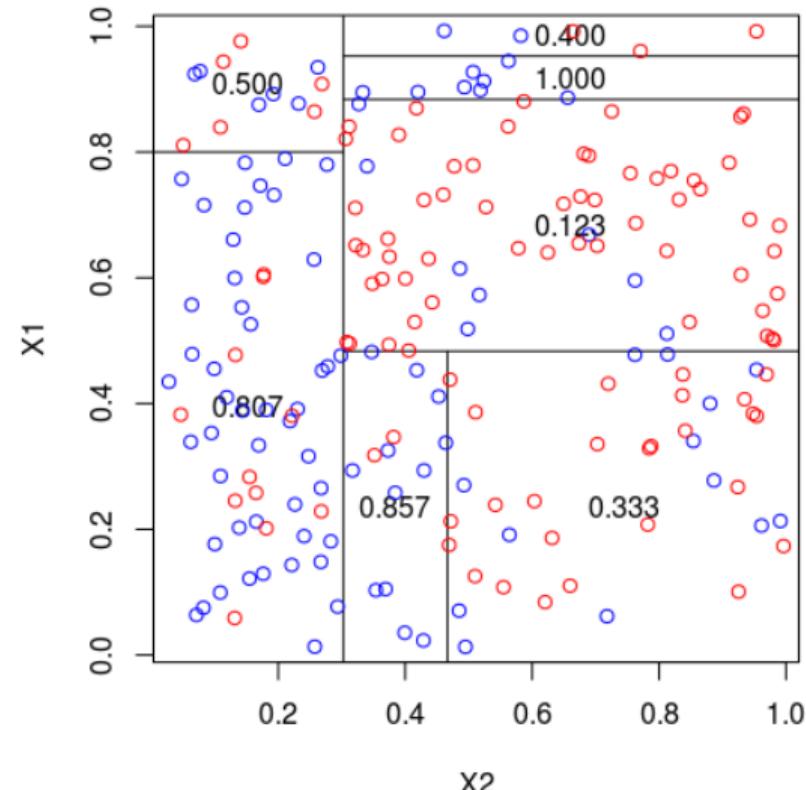
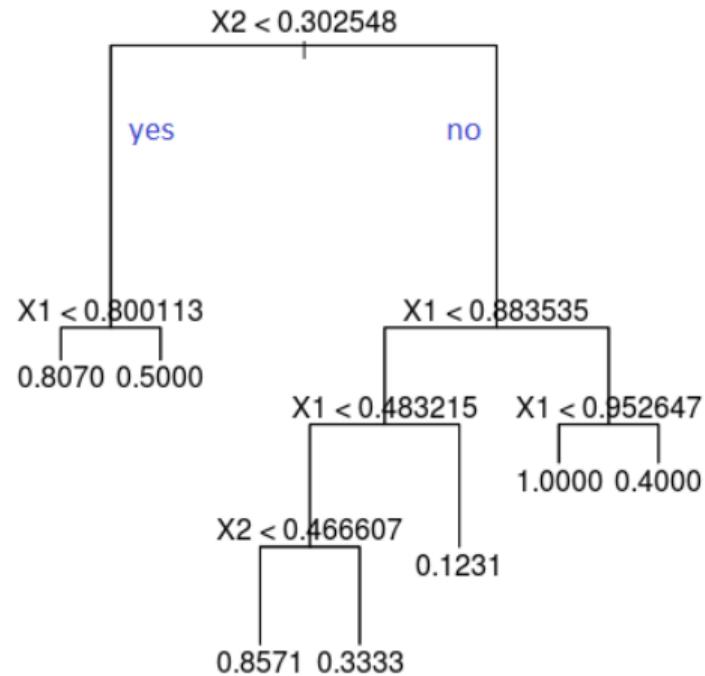
$$\text{División óptima} = \arg \min_s \left(\sum_{y_i \in R_1(s)} (y_i - \bar{y}_{R_1})^2 + \sum_{y_i \in R_2(s)} (y_i - \bar{y}_{R_2})^2 \right)$$

donde:

- $R_1(s)$ y $R_2(s)$ son los subconjuntos generados por la división,
- \bar{y}_{R_j} es la media de los valores objetivo en la región R_j .

El proceso continúa de forma jerárquica hasta que se cumple un criterio de parada (profundidad máxima, número mínimo de observaciones o mejora marginal insuficiente).

Árbol de Regresión — Representación Visual



Random Forest Regressor

El **Random Forest Regressor** es un modelo de ensamble formado por múltiples árboles de regresión, entrenados sobre subconjuntos aleatorios tanto de los datos como de las variables en cada partición.

La predicción final se obtiene como el promedio de las predicciones individuales de los árboles:

$$\hat{y} = \frac{1}{T} \sum_{t=1}^T \hat{y}^{(t)}$$

donde T es el número total de árboles.

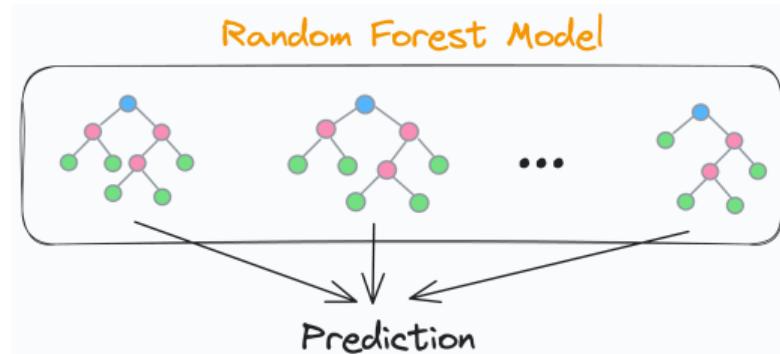


Figure: Esquema de Random Forest Regressor.

Variables usadas durante el entrenamiento e importancia de las mismas

Variable	Importancia	Variable	Importancia
load_num	13.87%	lag_1	8.73%
load_size	7.45%	lag_3	7.44%
lag_2	7.37%	last_day_avg	5.12%
plant_code	4.99%	last_week_day_avg	4.99%
hour	4.92%	day_of_month	3.97%
last_week_avg	3.89%	truck_spacing_mins	3.89%
minute	3.41%	week	2.97%
last_month_avg	2.82%	structure	2.28%
day_of_week	2.11%	seg obr	1.45%
is_first_dispatch	1.36%	prod_cat	1.23%
month	1.19%	building	1.16%
schd_truck_type	1.15%	bomb_type	1.02%
turn	0.71%	is_with_bomb	0.49%

Retos encontrados en el Proceso

- Datos nulos y faltantes.
- Datos inconsistentes (tiempos negativos).
- En el momento del despacho el mixer programado puede cambiar.
- El cliente puede cancelar el pedido.
- El cliente puede ajustar el volumen solicitado.
- La obra no empieza a la hora indicada.
- El cliente secuestra los mixers.

Resultados obtenidos: Error programación manual(CMS) vs ML

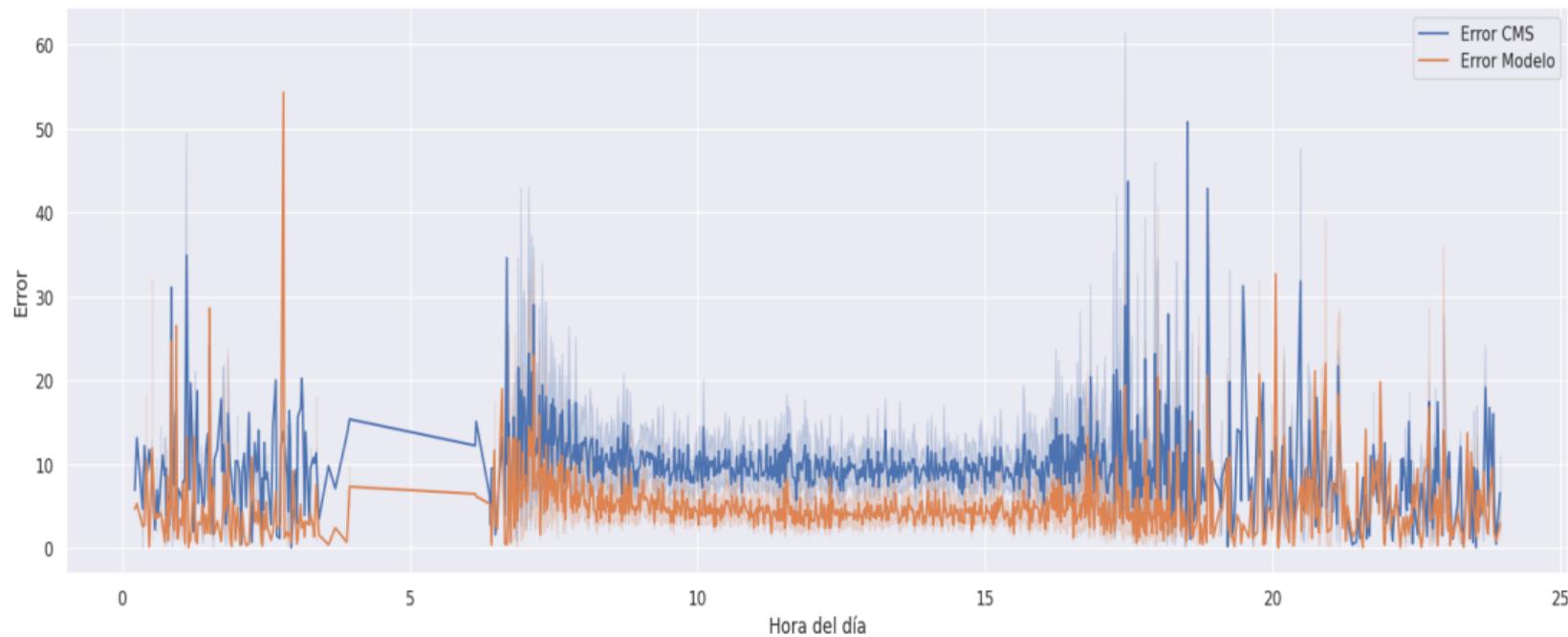


Figure: Error manual(Azul) vs Error ML(Amarillo).

Resultados obtenidos: Comportamiento Programado vs Predicho vs Real

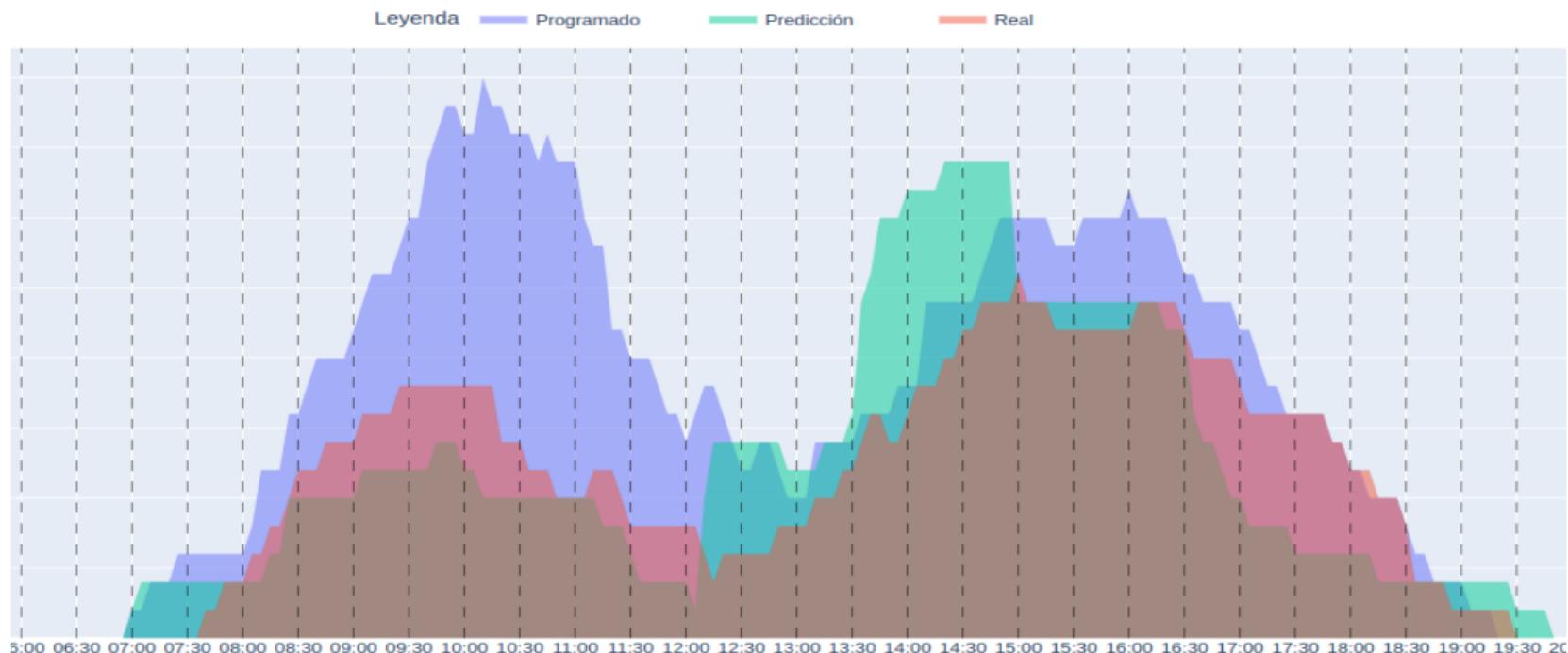


Figure: La programación con ML se aproxima de mejor manera al comportamiento real.

¿Cuál es el legado de la física al ML?

De los principios fundamentales de la naturaleza
a la inteligencia artificial

1. El Legado: Mecánica Estadística y Energía

De la distribución de Boltzmann a las Redes Neuronales

La física estadística describe cómo sistemas complejos tienden a estados de **mínima energía** (equilibrio térmico).

Aplicación en ML:

- **Modelos Basados en Energía:** Arquitecturas como las *Máquinas de Boltzmann* o redes de Hopfield definen una "función de energía" para el estado de sus neuronas.
- El proceso de aprendizaje consiste en encontrar la configuración de pesos que minimiza esta energía del sistema.

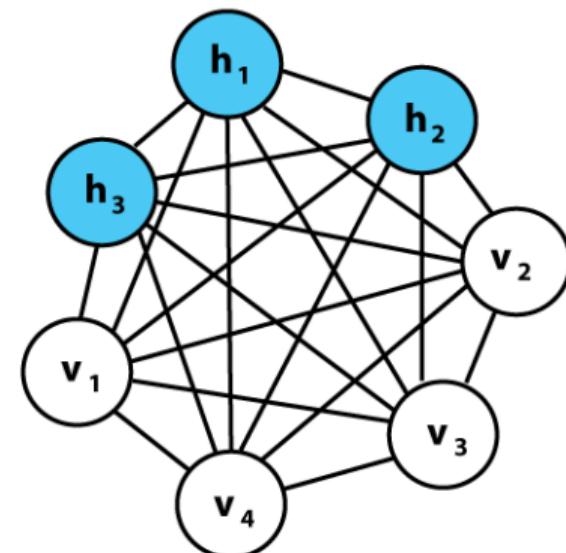


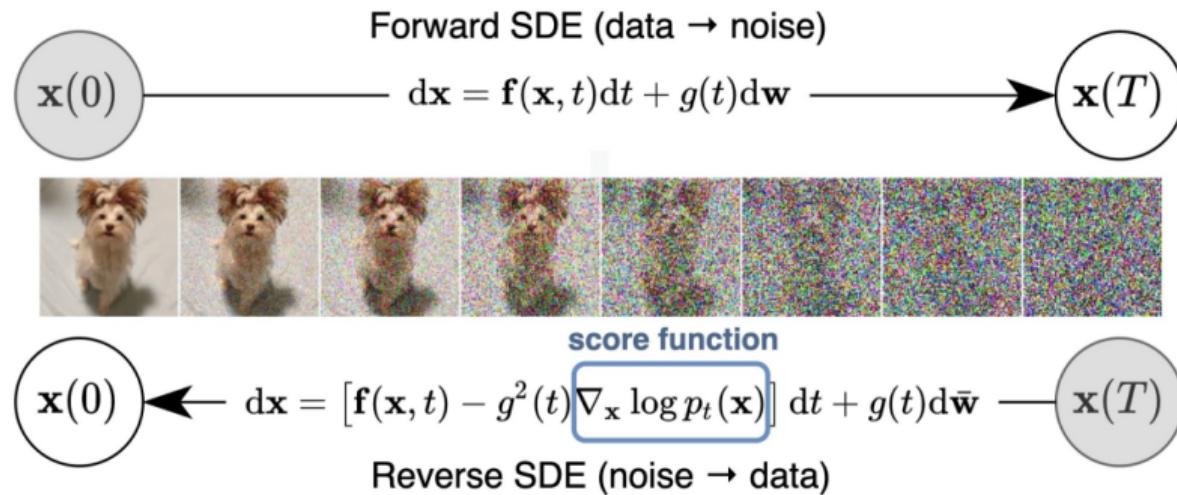
Figure: Redes que buscan estados de "baja energía".

2. El Legado: Difusión y Termodinámica

De la dispersión al arte generativo

La física describe cómo una estructura (como una imagen ' $x(0)$ ' de un perro) se degrada gradualmente en ruido puro ' $x(T)$ ' a lo largo del tiempo. Es el proceso natural de **difusión** y aumento de la entropía.

Partiendo del ruido puro (' $x(T)$ '), el modelo utiliza una "función de score" para guiar la reconstrucción, eliminando el ruido paso a paso hasta generar una imagen coherente (' $x(0)$ ').



3. El Legado: Teoría de Grupos y Simetrías

Reconociendo patrones sin importar la posición

En física, las leyes fundamentales no cambian si te mueves en el espacio (**invarianza translacional**) o rotas el sistema.

Aplicación en ML (Visión por Computador):

- Las **Redes Convolucionales (CNNs)** incorporan esta invarianza física en su diseño.
- Usan el mismo filtro (kernel) deslizándose por toda la imagen, lo que les permite reconocer un objeto (ej. un gato) sin importar en qué parte de la foto se encuentre.

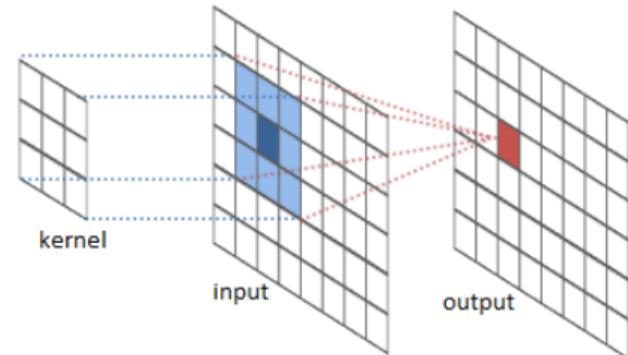


Figure: Convolución: aplicando la misma "regla" en todo el espacio.

4. El Legado: El Principio de Mínima Acción

La base fundamental del entrenamiento

Un principio universal de la física clásica: la naturaleza siempre sigue el camino que minimiza la "acción" (el costo energético de la trayectoria).

Aplicación en ML (Optimización):

- Es el fundamento filosófico del entrenamiento de redes neuronales.
- El algoritmo de **Backpropagation** y el **Descenso de Gradiente** buscan la trayectoria óptima en el espacio de parámetros (pesos) que minimice la función de costo (error).

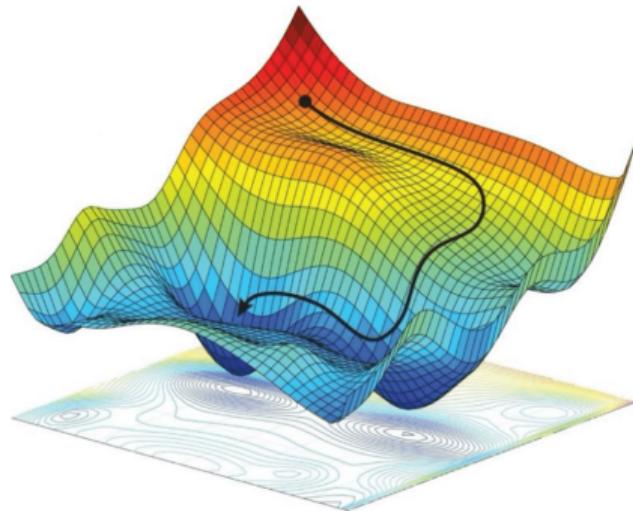


Figure: Buscando la trayectoria de mínimo "costo".

3. La Revolución Inversa: ML potenciando la Física

Hoy, la inteligencia artificial devuelve el favor resolviendo problemas físicos intratables:

- **Aceleración de Simulaciones:** Redes neuronales que actúan como *surrogates* para emular simulaciones de fluidos (CFD) o clima miles de veces más rápido.
- **Física de Partículas (CERN):** Clasificación de eventos en colisionadores y detección de anomalías en la búsqueda de nueva física.
- **PINNs (Physics-Informed Neural Networks):** Redes que integran ecuaciones diferenciales (PDEs) en su función de pérdida, respetando las leyes físicas (conservación de masa/energía) durante el aprendizaje.
- **Astronomía:** Clasificación automática de galaxias y detección de exoplanetas en datos ruidosos.

Conclusión

*“Ambos campos buscan estados de equilibrio:
la física mediante la minimización de la **energía**,
el aprendizaje automático mediante la minimización del **error**.”*