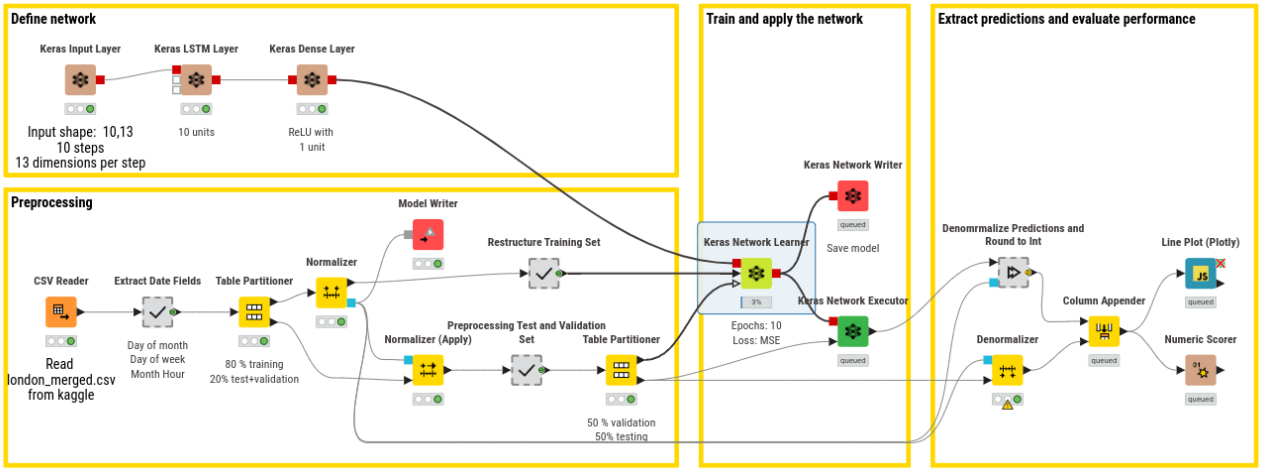
Flujo KNIME: LSTM Multivariada para Series de Tiempo

Visión General del Workflow

Este workflow implementa una **LSTM multivariada** para predicción de series de tiempo, dividido en **4 secciones principales**:

1. **Define network** (Definir la arquitectura)
2. **Preprocessing** (Preprocesamiento de datos)
3. **Train and apply the network** (Entrenar y aplicar el modelo)
4. **Extract predictions and evaluate performance** (Extraer predicciones y evaluar)



SECCIÓN 1: Define Network (Definir la Red)

Arquitectura de la LSTM:

Keras Input Layer → Keras LSTM Layer → Keras Dense Layer

1. Keras Input Layer

Parámetros configurados:

- **Input shape: 10,13**
  - **10 steps**: Ventana temporal de 10 pasos hacia atrás (lookback)
  - **13 dimensions per step**: 13 características/variables por cada paso temporal

Interpretación:

- La red recibe secuencias de 10 observaciones consecutivas
- Cada observación tiene 13 variables (features multivariadas)
- Shape completo: (batch_size, 10, 13)

2. Keras LSTM Layer

Parámetros configurados:

- **10 units**: 10 neuronas LSTM en la capa

Función:

- Procesa la secuencia temporal capturando dependencias a largo plazo
- Extrae patrones temporales de las 13 variables

3. Keras Dense Layer

Parámetros configurados:

- **ReLU with 1 unit**:
    - Función de activación: ReLU
    - 1 neurona de salida (predicción univariada)

Función:

- Capa de salida que produce la predicción final
- Aunque la entrada es multivariada, la salida es un solo valor

Flujo KNIME: LSTM Multivariada para Series de Tiempo

Visión General del Workflow

Este workflow implementa una **LSTM multivariada** para predicción de series de tiempo, dividido en **4 secciones principales**:

1. **Define network** (Definir la arquitectura)
2. **Preprocessing** (Preprocesamiento de datos)
3. **Train and apply the network** (Entrenar y aplicar el modelo)
4. **Extract predictions and evaluate performance** (Extraer predicciones y evaluar)

SECCIÓN 1: Define Network (Definir la Red)

Arquitectura de la LSTM:

`Keras Input Layer → Keras LSTM Layer → Keras Dense Layer`

1. Keras Input Layer

Parámetros configurados:

- **Input shape: 10,13**
    - **10 steps**: Ventana temporal de 10 pasos hacia atrás (lookback)
    - **13 dimensions per step**: 13 características/variables por cada paso temporal

Interpretación:

- La red recibe secuencias de 10 observaciones consecutivas

- Cada observación tiene 13 variables (features multivariadas)
- Shape completo: (batch_size, 10, 13)

2. Keras LSTM Layer

Parámetros configurados:

- **10 units**: 10 neuronas LSTM en la capa

Función:

- Procesa la secuencia temporal capturando dependencias a largo plazo
- Extrae patrones temporales de las 13 variables

3. Keras Dense Layer

Parámetros configurados:

- **ReLU with 1 unit**:
  - Función de activación: ReLU
  - 1 neurona de salida (predicción univariada)

Función:

- Capa de salida que produce la predicción final
- Aunque la entrada es multivariada, la salida es un solo valor

---

SECCIÓN 2: Preprocessing (Preprocesamiento)

Flujo de preprocesamiento:

CSV Reader → Extract Date Fields → Table Partitioner →
Normalizer → Normalizer (Apply) → Restructure Training/Test Sets

1. CSV Reader

Parámetros:

- **Read london_merged.csv from kaggle**

Función:

- Carga el dataset de series de tiempo (probablemente datos de clima/tráfico de Londres)

2. Extract Date Fields

Parámetros:

- **Day of month**
- **Day of week**
- **Month Hour**

Función:

- Extrae características temporales de la fecha
- Crea variables cíclicas para capturar estacionalidad

- Estas se suman a las 13 dimensiones de entrada

3. Table Partitioner

Parámetros:

- **80% training**
- **20% test+validation**

Función:

- División temporal respetando el orden cronológico
- 80% primeras observaciones → entrenamiento
- 20% últimas observaciones → test y validación

4. Normalizer

Función:

- Se ajusta (fit) con los datos de **training**
- Normaliza todas las variables (las 13 dimensiones)
- Guarda parámetros de normalización

5. Normalizer (Apply)

Función:

- Aplica la misma transformación al conjunto de **test+validation**
- Usa los parámetros del training set
- Evita data leakage

6. Restructure Training Set

Función:

- Convierte los datos tabulares en secuencias temporales
- Crea las ventanas de 10 pasos (lookback=10)
- Formato: cada fila → secuencia de 10 observaciones con 13 variables

7. Table Partitioner (segundo)

Parámetros:

- **50% validation**
- **50% testing**

Función:

- Del 20% reservado, divide en:
    - 10% para validación (durante entrenamiento)
    - 10% para testing final

SECCIÓN 3: Train and Apply the Network (Entrenar y Aplicar)

Flujo de entrenamiento:

Keras Network Writer → Keras Network Learner →
Keras Network Executor (con validación)

1. Model Writer → Keras Network Writer

Función:

- Conecta la arquitectura definida (Sección 1) con el proceso de entrenamiento
- Prepara el modelo para ser entrenado

2. Keras Network Learner

Parámetros:

- **Epochs: 10**
- **Loss: MSE**

Función:

- Entrena el modelo LSTM con los datos de training
- 10 épocas de entrenamiento
- Optimiza usando Mean Squared Error como función de pérdida
- Monitorea el progreso con el conjunto de validación

3. Keras Network Executor

Función:

- Aplica el modelo entrenado al conjunto de validación
- Genera predicciones para evaluar durante el entrenamiento
- Permite monitorear overfitting

SECCIÓN 4: Extract Predictions and Evaluate Performance

Flujo de evaluación:

Keras Network Executor → Denormalize Predictions and Round to Int →
Column Appender → Denormalizer → Numeric Scorer + Line Plot

1. Keras Network Executor (test)

Función:

- Aplica el modelo entrenado al conjunto de **test** (10% final)
- Genera predicciones en escala normalizada

2. Denormalize Predictions and Round to Int

Función:

- **Desnormaliza** las predicciones (escala normalizada → escala original)
- **Redondea** a enteros (si el target son conteos o valores discretos)
- Hace las predicciones interpretables

3. Column Appender

Función:

- Combina las predicciones desnormalizadas con los datos originales
- Prepara los datos para visualización y evaluación

4. Denormalizer

Función:

- Desnormaliza también los valores reales del target
- Garantiza que predicciones y valores reales estén en la misma escala

5. Numeric Scorer

Función:

- Calcula métricas de evaluación en escala original:
  - MSE (Mean Squared Error)
  - RMSE (Root Mean Squared Error)
  - MAE (Mean Absolute Error)
  - R^2

6. Line Plot (Plotly)

Función:

- Visualiza las predicciones vs valores reales en el tiempo
- Muestra el rendimiento del modelo gráficamente
- Permite identificar patrones de error

Características Clave de este Workflow

1. Entrada Multivariada, Salida Univariada:

- **Entrada**: 13 variables temporales (temperatura, humedad, tráfico, etc.)
- **Salida**: 1 variable objetivo (ej: demanda de bicicletas)

2. Lookback Window:

- Usa los **últimos 10 pasos temporales** para predecir el siguiente
- Captura dependencias de corto-medio plazo

3. División Temporal Correcta:

- 80% training (primeras observaciones)

- 10% validation (observaciones intermedias)
- 10% test (últimas observaciones)
- Respeta el orden cronológico

4. Normalización/Desnormalización:

- **Normaliza**: Todas las variables de entrada + target
- **Desnormaliza**: Predicciones y target para evaluación e interpretación
- Mantiene consistencia entre train/test

5. Features Temporales:

- Extrae **día del mes**, **día de la semana**, **hora del mes**
- Captura patrones cíclicos (diarios, semanales, mensuales)

Comparación: LSTM Univariada vs Multivariada

Aspecto	Univariada	Multivariada (este workflow)
Variables de entrada	1 (ej: solo temperatura)	13 (temperatura, humedad, hora, etc.)
Input shape	(10, 1)	(10, 13)
Complejidad	Menor	Mayor
Poder predictivo	Limitado a una variable	Aprovecha relaciones entre variables
Ejemplo	Predecir temperatura futura solo con temperaturas pasadas	Predecir demanda de bicicletas usando clima, hora, día, etc.

Conceptos Clave

¿Por qué 10 steps y 13 dimensions?

- **10 steps**: Ventana temporal de 10 observaciones (lookback)
  - Si los datos son horarios: últimas 10 horas
  - Si son diarios: últimos 10 días
- **13 dimensions**: Número de variables por observación
  - Variables originales del dataset
  - Variables derivadas (día, semana, mes, hora)

¿Por qué LSTM para series multivariadas?

- Las LSTM pueden capturar:
  - Dependencias temporales a largo plazo
  - Relaciones complejas entre múltiples variables
  - Patrones no lineales en el tiempo

Flujo de datos a través de la LSTM:

Input: [10 pasos × 13 variables] → LSTM(10 units) → Dense(1) → Output: [1 predicción]

Puntos Críticos del Workflow

1. **Restructure Sets**: Crucial para convertir datos tabulares en secuencias
2. **Doble normalización**: Training primero, luego Apply en test
3. **Doble desnormalización**: Predicciones Y valores reales
4. **División temporal**: NUNCA mezclar train/test en series de tiempo
5. **Round to Int**: Solo si el target son valores discretos

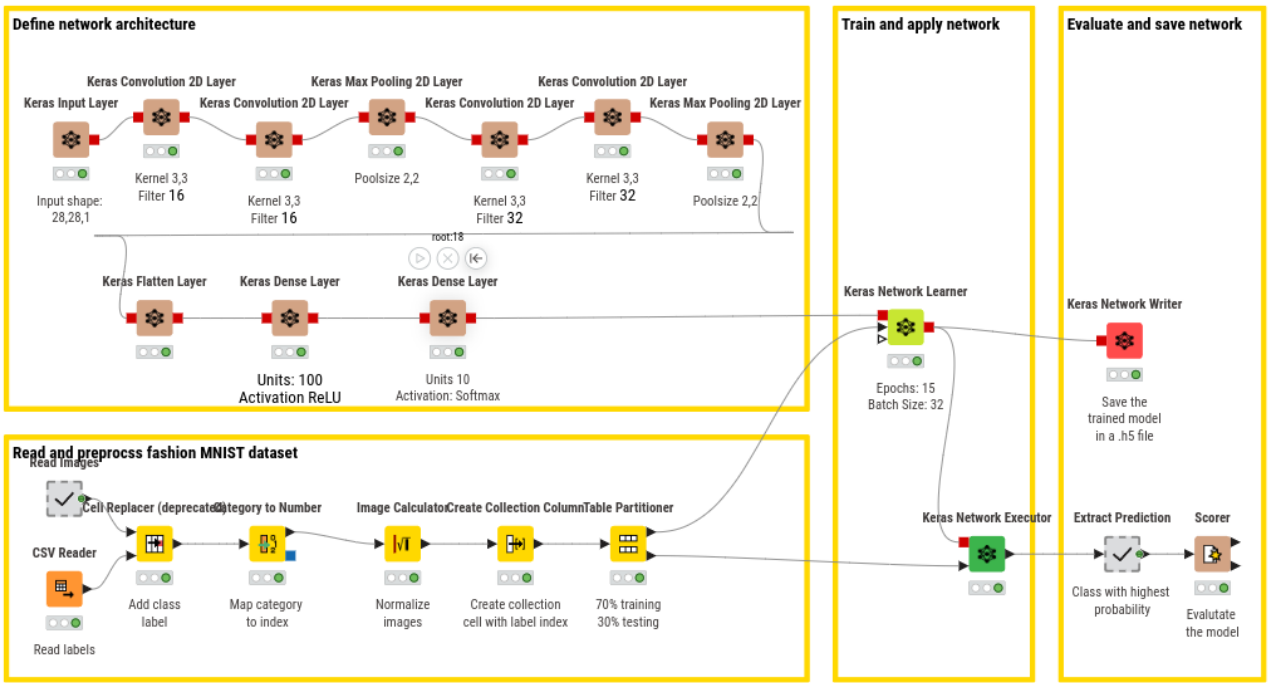
Este workflow es significativamente más complejo que un MLP de regresión estándar debido a la naturaleza temporal y multivariada de los datos.

Flujo KNIME: CNN para Clasificación de Fashion MNIST

Visión General del Workflow

Este workflow implementa una **Red Neuronal Convolucional (CNN)** para clasificar imágenes del dataset Fashion MNIST, dividido en **3 secciones principales**:

1. **Define network architecture** (Definir la arquitectura CNN)
2. **Read and preprocess Fashion MNIST dataset** (Leer y preprocesar datos)
3. **Train and apply network + Evaluate and save network** (Entrenar, aplicar y evaluar)



SECCIÓN 1: Define Network Architecture (Arquitectura CNN)

Arquitectura completa de la CNN:

Input Layer → Conv2D(16) → MaxPooling → Conv2D(32) → MaxPooling →
Flatten → Dense(100, ReLU) → Dense(10, Softmax)

1. Keras Input Layer

Parámetros:

- **Input shape: 28,28,1**
  - **28×28**: Dimensiones de la imagen (píxeles)
  - **1**: Canal único (escala de grises)

Interpretación:

- Fashion MNIST son imágenes en escala de grises de 28×28 píxeles
- Cada imagen es una prenda de ropa (camiseta, pantalón, zapato, etc.)

Bloque Convolucional 1:

2. Keras Convolution 2D Layer (primera)

Parámetros:

- **Kernel 3,3**
- **Filter 16**

Función:

- Aplica 16 filtros convolucionales de 3×3 píxeles
- Detecta características básicas (bordes, líneas, texturas)
- Output: (26, 26, 16) - se reduce por la convolución sin padding

3. Keras Max Pooling 2D Layer (primera)

Parámetros:

- **Poolsize 2,2**

Función:

- Reduce dimensiones espaciales a la mitad
- Toma el valor máximo de cada ventana de 2×2
- Output: (13, 13, 16)
- Reduce cómputo y proporciona invarianza espacial

Bloque Convolucional 2:

4. Keras Convolution 2D Layer (segunda)

Parámetros:

- **Kernel 3,3**
- **Filter 32**

Función:

- Aplica 32 filtros convolucionales de 3×3
- Detecta características más complejas (patrones, formas)
- Output: (11, 11, 32)

5. Keras Max Pooling 2D Layer (segunda)

Parámetros:

- **Poolsize 2,2**

Función:

- Segunda reducción dimensional
- Output: (5, 5, 32)
- Prepara para la transición a capas densas

Capas Totalmente Conectadas (Fully Connected):

6. Keras Flatten Layer

Función:

- Convierte la salida 3D (5, 5, 32) en un vector 1D
- Output: vector de 5 × 5 × 32 = **800 valores**
- Necesario para conectar con capas Dense

7. Keras Dense Layer (primera)

Parámetros:

- **Units: 100**
- **Activation: ReLU**

Función:

- Primera capa totalmente conectada
- 100 neuronas con activación ReLU
- Aprende combinaciones de alto nivel de las características extraídas
- ReLU introduce no linealidad

8. Keras Dense Layer (segunda - salida)

Parámetros:

- **Units: 10**
- **Activation: Softmax**

Función:

- Capa de salida con 10 neuronas (10 clases de Fashion MNIST)
- Softmax convierte las salidas en probabilidades que suman 1
- Cada neurona representa una clase:
    - 0: T-shirt/top
    - 1: Trouser
    - 2: Pullover
    - 3: Dress

- 4: Coat
- 5: Sandal
- 6: Shirt
- 7: Sneaker
- 8: Bag
- 9: Ankle boot

SECCIÓN 2: Read and Preprocess Fashion MNIST Dataset

Flujo de preprocesamiento:

CSV Reader → Cell Replacer → Category to Number →
Image Calculator → Create Collection Column → Table Partitioner

1. CSV Reader

Parámetros:

- **Read labels** (archivo de etiquetas)

Función:

- Lee las etiquetas del dataset Fashion MNIST
- Contiene la clase de cada imagen (0-9)

2. Cell Replacer (deprecated)

Parámetros:

- **Add class label**

Función:

- Reemplaza números de clase por etiquetas descriptivas
- Ejemplo: 0 → "T-shirt/top", 1 → "Trouser", etc.
- Hace los datos más interpretables

3. Category to Number → Map category to Index

Parámetros:

- **Map category to Index**

Función:

- Convierte las etiquetas categóricas de vuelta a números
- Formato requerido para el entrenamiento de la red
- Crea encoding numérico consistente

4. Image Calculator → Create Collection Column

Parámetros:

- **Normalize images**
- **Create collection cell with label index**

Función:

- **Normaliza** las imágenes (píxeles de 0-255 → 0-1)
- Crucial para convergencia de la CNN
- Agrupa imagen y etiqueta en una colección

5. Table Partitioner

Parámetros:

- **70% training**
- **30% testing**

Función:

- Divide el dataset en entrenamiento (70%) y prueba (30%)
- Fashion MNIST tiene 60,000 imágenes de entrenamiento

SECCIÓN 3: Train and Apply Network

Flujo de entrenamiento y evaluación:

Keras Network Learner → Keras Network Executor → Extract Prediction → Scorer

1. Keras Network Learner

Parámetros:

- **Epochs: 15**
- **Batch Size: 32**

Función:

- Entrena la CNN con los datos de entrenamiento
- 15 épocas de entrenamiento completo
- Batch size de 32: procesa 32 imágenes a la vez
- Optimiza usando backpropagation

2. Keras Network Executor

Función:

- Aplica el modelo entrenado al conjunto de prueba
- Genera predicciones para cada imagen de test
- Output: probabilidades para las 10 clases

3. Extract Prediction

Parámetros:

- **Class with highest probability**

Función:

- Extrae la clase con mayor probabilidad
- Convierte probabilidades en predicción final
- Ejemplo: [0.01, 0.02, 0.89, 0.05, ...] → Clase 2 (Pullover)

4. Scorer

Parámetros:

- **Evaluate the model**

Función:

- Calcula métricas de evaluación:
  - **Accuracy**: Porcentaje de predicciones correctas
  - **Precision, Recall, F1-Score**: Por cada clase
  - **Confusion Matrix**: Errores entre clases
- Determina el rendimiento del modelo

SECCIÓN 4: Evaluate and Save Network

1. Keras Network Writer

Parámetros:

- **Save the trained model in a .h5 file**

Función:

- Guarda el modelo entrenado en formato HDF5
- Permite reutilizar el modelo sin reentrenar
- Formato estándar de Keras/TensorFlow

Características Clave de este Workflow

1. Arquitectura CNN Clásica:

- **2 bloques convolucionales**: Extracción jerárquica de características
- **2 capas de pooling**: Reducción dimensional y robustez
- **2 capas densas**: Clasificación final

2. Progresión de Filtros:

- Primera conv: **16 filtros** (características simples)
- Segunda conv: **32 filtros** (características complejas)
- Patrón común: aumentar filtros en capas profundas

3. Reducción de Dimensionalidad:

Input: 28×28×1 → Conv1: 26×26×16 → Pool1: 13×13×16 →
Conv2: 11×11×32 → Pool2: 5×5×32 → Flatten: 800 →
Dense1: 100 → Dense2: 10

4. Fashion MNIST - 10 Clases:

El modelo clasifica prendas de ropa en 10 categorías distintas

Comparación: CNN vs MLP para Imágenes

Aspecto	MLP	CNN (este workflow)
Entrada	Vector plano (784 valores)	Imagen 2D (28×28×1)
Capas	Solo Dense	Conv2D + Pooling + Dense
Parámetros	~80,000+	~50,000 (más eficiente)
Invarianza espacial	No	Sí (pooling)
Detección de características	Manual	Automática (filtros)
Rendimiento en imágenes	~88% accuracy	~92-94% accuracy

Conceptos Clave

¿Por qué usar CNN para imágenes?

1. **Invarianza espacial**: Detecta patrones sin importar la posición
2. **Compartición de parámetros**: Los filtros se aplican a toda la imagen
3. **Jerarquía de características**:
    - Capas tempranas: bordes, líneas
    - Capas medias: texturas, patrones
    - Capas finales: formas completas (zapatos, camisas)

Flujo de información:

Imagen (28×28) → Características locales (bordes) →
Características medias (texturas) → Características globales (formas) →
Clasificación (10 clases)

Tamaño de kernels y filtros:

- **Kernel 3×3**: Tamaño estándar, eficiente y efectivo
- **16 → 32 filtros**: Aumentar capacidad en capas profundas
- **Poolsize 2×2**: Reduce dimensiones a la mitad manteniendo info importante

Proceso de Entrenamiento

Forward Pass (predicción):

1. Imagen entra → Convoluciones extraen características
2. Pooling reduce dimensiones
3. Flatten convierte a vector
4. Dense layers clasifican
5. Softmax genera probabilidades

Backward Pass (aprendizaje):

1. Calcular error entre predicción y etiqueta real
2. Propagar error hacia atrás (backpropagation)
3. Actualizar pesos de todos los filtros y capas
4. Repetir por 15 epochs con batches de 32 imágenes

Puntos Críticos del Workflow

1. **Normalización de imágenes**: Píxeles 0-255 → 0-1 (esencial para CNNs)
2. **Input shape correcto**: 28,28,1 (altura, ancho, canales)
3. **Flatten antes de Dense**: Las capas convolucionales producen salidas 3D
4. **Softmax en salida**: Para clasificación multiclase
5. **Batch size**: 32 es un buen balance entre velocidad y estabilidad

Fuentes y Recursos

Artículos Científicos:

Machine Learning para Marketing en KNIME Hub

- Villarroel Ordenes, F., & Silipo, R. (2021). Machine learning for marketing on the KNIME Hub: The development of a live repository for marketing applications. *Journal of Business Research*, 137(1), 393-410.
- DOI: 10.1016/j.jbusres.2021.08.036
- KNIME Hub Space: Machine Learning and Marketing

Libros y Guías:

Codeless Deep Learning with KNIME

- Autoras: Kathrin Melcher & Rosaria Silipo
- Editorial: Packt Publishing (Community Edition)
- Descripción: Guía completa para construir, entrenar y desplegar arquitecturas de redes neuronales profundas usando KNIME Analytics Platform
- KNIME Hub Space: Codeless Deep Learning with KNIME

Recursos Adicionales en KNIME Hub:

1. **Space de Machine Learning y Marketing**
    - Repositorio en vivo con aplicaciones prácticas de ML para marketing
    - Workflows y ejemplos reproducibles
    - URL: https://hub.knime.com/knime/spaces/Machine%20Learning%20and%20Marketing/~JyadcetnSt5U1vcw/
2. **Space de Deep Learning sin Código**
    - Ejemplos prácticos de CNNs, LSTMs, Autoencoders y más
    - Workflows listos para usar del libro "Codeless Deep Learning with KNIME"
    - URL: https://hub.knime.com/kathrin/spaces/Codeless%20Deep%20Learning%20with%20KNIME/~yMp8GBkT0Xwzx5X2/

Plataforma Principal:

KNIME Analytics Platform

- Plataforma open-source para ciencia de datos y machine learning
- Website oficial: www.knime.com
- Documentación: docs.knime.com