


DATA WRANGLING: MANIPULACIÓN, ENRIQUECIMIENTO Y TRANSFORMACIÓN EFICIENTE DE DATOS EN SERIES Y DATAFRAMES

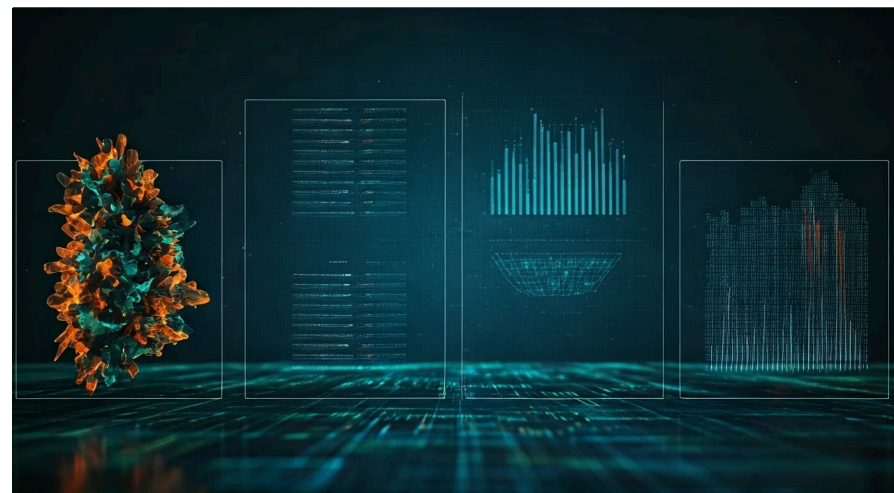
La preparación y transformación de datos es uno de los ejes fundamentales de la ingeniería y la ciencia de datos moderna. El término data wrangling ha surgido como un paraguas que abarca todas las operaciones destinadas a limpiar, estructurar, enriquecer y preparar datos crudos para su análisis, visualización, modelado o integración con otros sistemas. Si bien la obtención y almacenamiento de datos suele acaparar atención en etapas iniciales, en la práctica profesional la verdadera capacidad de extraer valor depende del dominio de las técnicas de data wrangling y su aplicación eficiente.

 **por Kibernet Capacitación S.A.**

DATA WRANGLING: QUÉ ES DATA WRANGLING

Data wrangling es el proceso de transformar datos en bruto—generalmente incompletos, desorganizados o inconsistentes—en conjuntos de datos limpios, estructurados y listos para el análisis. Este proceso no solo incluye la limpieza (remover valores nulos, corregir errores), sino también la transformación (ajustar formatos, normalizar escalas, convertir tipos de datos), el enriquecimiento (agregar información contextual o derivada), la reestructuración (pivotar, aplanar, dividir, combinar) y la reducción (seleccionar, filtrar, resumir).

En otras palabras, data wrangling es la "cocina" de los datos, donde los ingredientes crudos se preparan para ser consumidos por modelos de machine learning, algoritmos estadísticos o sistemas de visualización. Detrás de toda solución de inteligencia artificial robusta, existe un pipeline de wrangling bien diseñado.



EJEMPLO CONTEXTUAL



Múltiples fuentes de datos

Imagina un equipo de analistas en una aseguradora que recibe datos de siniestros de diversas fuentes: formularios web, registros telefónicos, correos electrónicos.



Formatos inconsistentes

Cada fuente usa formatos distintos, con errores tipográficos y convenciones locales.



Proceso de wrangling

El wrangling será necesario para consolidar, limpiar, traducir, enriquecer y preparar la data antes de estimar riesgos o detectar fraudes.

PRINCIPALES TAREAS DE DATA WRANGLING

El proceso de wrangling abarca un conjunto diverso de tareas, que suelen combinarse y repetirse iterativamente a lo largo del ciclo de vida del proyecto. Las más importantes son:



Limpieza

Identificación y corrección de valores nulos, inconsistencias, duplicados, errores de formato y registros anómalos.



Transformación

Modificación de la estructura o el contenido, como cambiar tipos de datos, normalizar, escalar, discretizar, o crear nuevas variables a partir de las existentes.



Enriquecimiento

Incorporación de datos externos, mapeos, categorizaciones o generación de features derivados.



Reestructuración

Cambios en la forma del dataset, como pivotar tablas, descomponer columnas, fusionar o dividir registros.



Filtrado y selección

Extracción de subconjuntos relevantes, aplicación de condiciones lógicas para depuración o análisis focalizado.

Estas tareas son relevantes tanto en estructuras tipo Serie (vectores unidimensionales, muy útiles para operaciones columnares o series temporales) como en DataFrames (tablas bidimensionales que generalizan la hoja de cálculo y la tabla SQL).

ORDENAMIENTO Y MANIPULACIÓN DE DATOS

Ordenar y manipular datos es uno de los primeros pasos tras la adquisición de un dataset. El orden puede tener impactos prácticos: influye en visualizaciones, facilita la detección de anomalías, prepara para procesos de resampling, y afecta algunos modelos secuenciales.

MUESTREOS ALEATORIOS

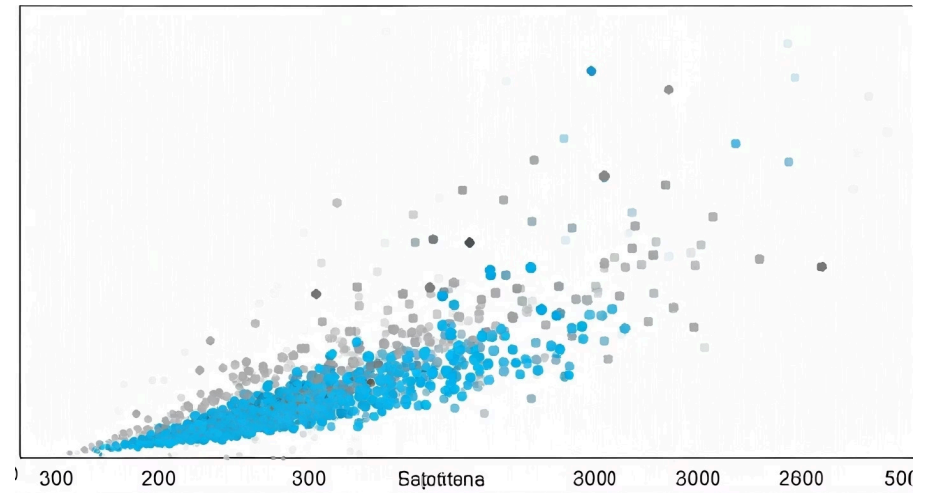
Muestrear aleatoriamente una porción del dataset es vital para análisis exploratorio, validación cruzada o pruebas de hipótesis sin procesar la totalidad de los datos. El muestreo ayuda a reducir la carga computacional y a obtener estimaciones representativas.

```
import pandas as pd

df = pd.read_csv('clientes.csv')

muestra = df.sample(frac=0.2, random_state=123) #
# Selecciona el 20% de las filas
```

Nota: El argumento `random_state` asegura reproducibilidad, clave en proyectos colaborativos.



PERMUTACIÓN DE LA DATA

Permutar la data significa reordenar todas las filas aleatoriamente, esencial antes de aplicar algoritmos sensibles al orden de los datos, como validaciones secuenciales o al dividir el dataset en entrenamiento y prueba.

```
df_permutado = df.sample(frac=1, random_state=1).reset_index(drop=True)
```

ORDENAMIENTO DE LA DATA

El ordenamiento por columnas clave ayuda a detectar registros extremos, errores de digitación o agrupar valores similares.

```
df_ordenado = df.sort_values(by=['fecha', 'monto'],
                              ascending=[True, False])
```

[illegible]

DETECCIÓN Y ELIMINACIÓN DE REGISTROS DUPLICADOS

Detectar duplicados es vital porque pueden surgir por errores de integración, fallas en APIs o procesos ETL defectuosos. Su presencia puede inflar métricas o distorsionar análisis.

```
duplicados = df.duplicated() # Serie booleana
df_limpio = df.drop_duplicates() # Elimina duplicados completos
```

Se puede especificar columnas relevantes y definir si se mantiene la primera o última ocurrencia.

[illegible][illegible]

REEMPLAZO DE VALORES

El reemplazo de valores permite estandarizar información categórica o corregir errores comunes de digitación. Es frecuente en integración de fuentes heterogéneas o al recibir datos de sistemas legados.

```
# Estandarizar género
df['genero'] = df['genero'].replace({'M': 'Masculino', 'F': 'Femenino'})

# Corregir errores de ciudad
df['ciudad'] = df['ciudad'].replace({'Sntiago': 'Santiago', 'Vina del Mr': 'Viña del Mar'})
```

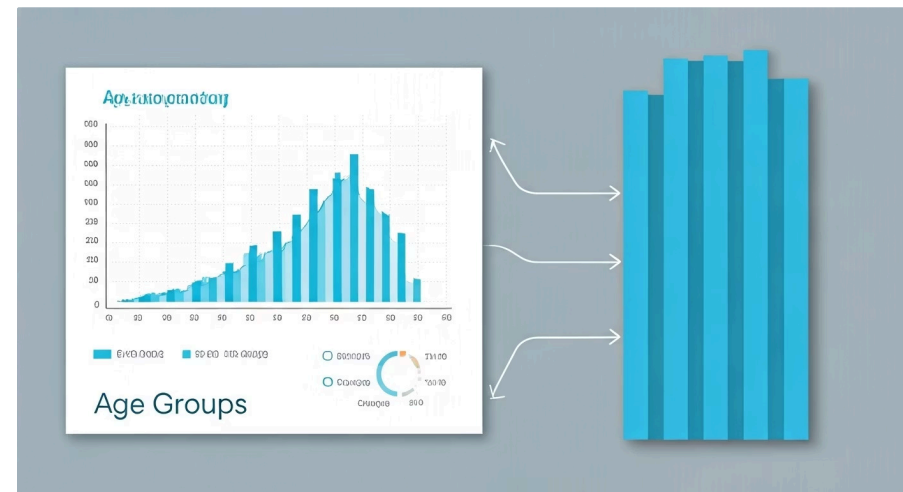
Además, el método `.replace()` admite listas, diccionarios y funciones, facilitando operaciones de mapeo y corrección en una sola línea.

DISCRETIZACIÓN Y BINNING (CONTENIDO OPCIONAL)

La discretización o binning consiste en convertir una variable continua en categorías agrupadas, útil para modelos que no aceptan continuos o para análisis de cohortes.

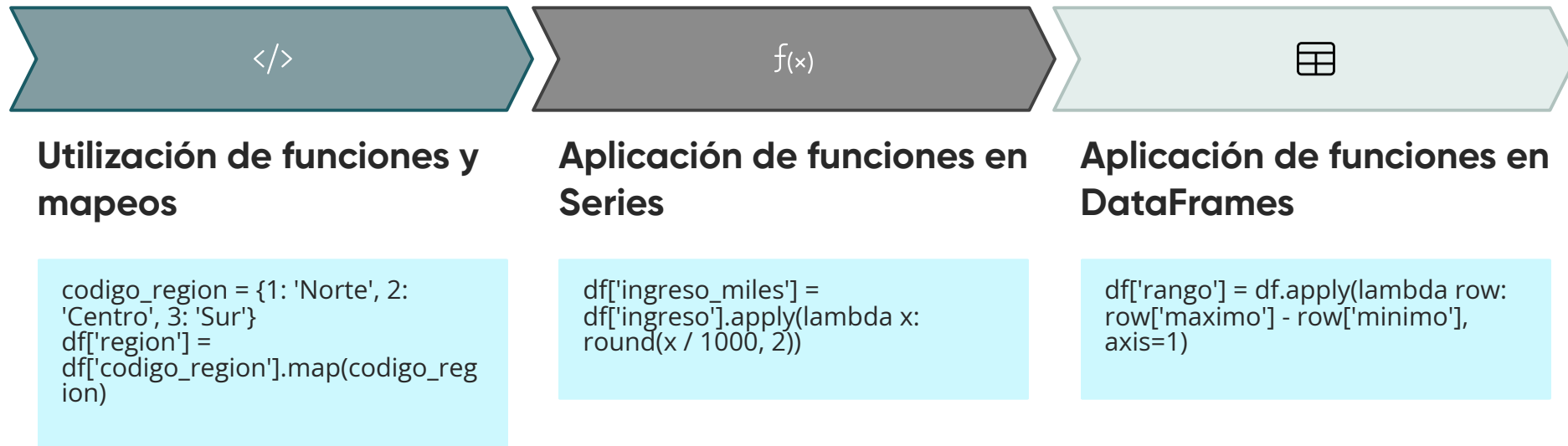
```
bins = [0, 18, 35, 60, 120]
labels = ['Menor', 'Joven Adulto', 'Adulto', 'Senior']
df['grupo_edad'] = pd.cut(df['edad'], bins=bins,
labels=labels)
```

Esto convierte la edad numérica en categorías de interés analítico.



ENRIQUECIMIENTO DE LA DATA: UTILIZACIÓN DE FUNCIONES Y MAPEOS

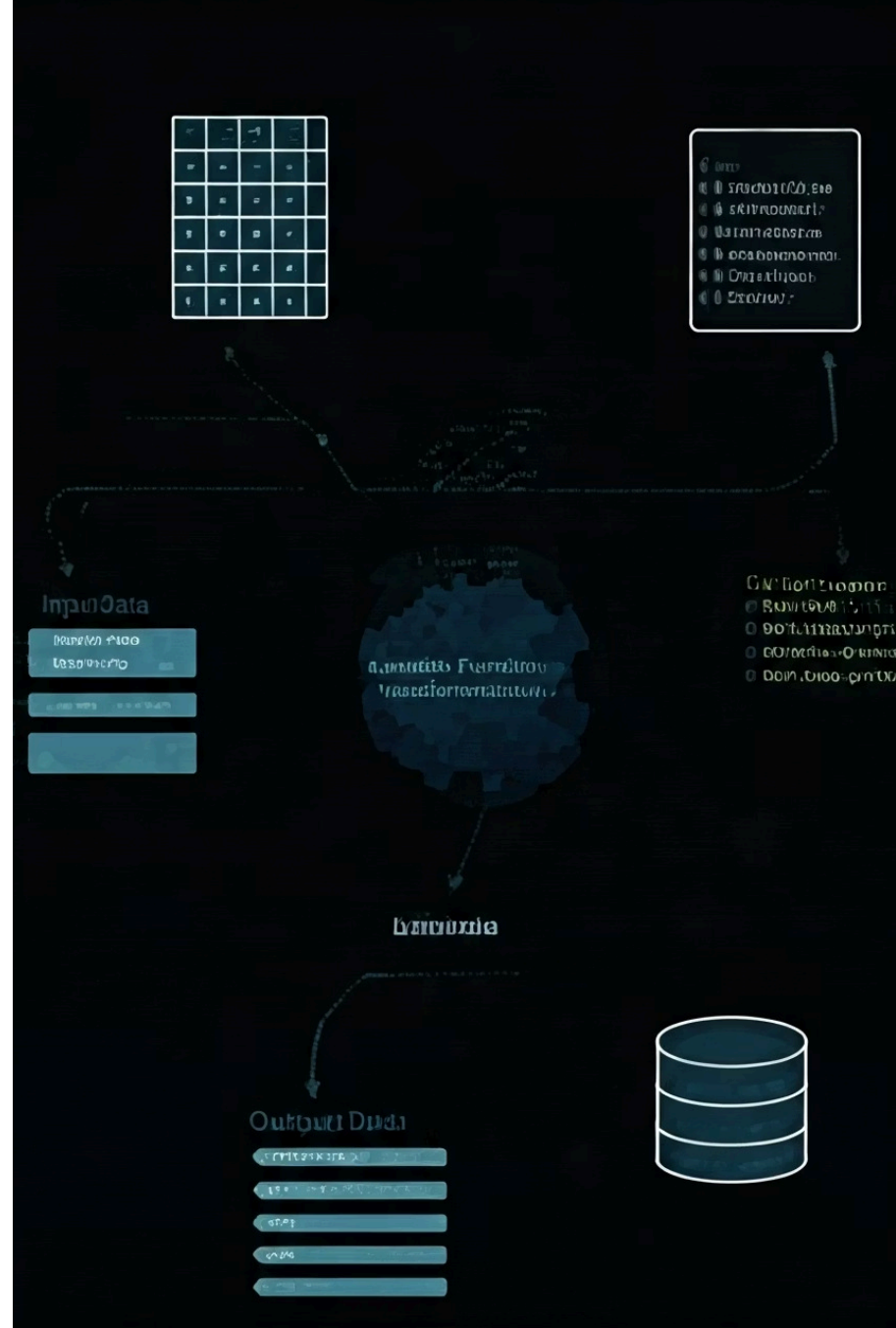
El enriquecimiento transforma datos básicos en información más valiosa, a través de la creación de variables, mapeos y cálculos personalizados.



UTILIZACIÓN DE EXPRESIONES LAMBDA EN LA APLICACIÓN DE FUNCIONES

Las lambdas son funciones anónimas, muy útiles para operaciones breves y cuando no se desea definir funciones externas.

```
df['score_normalizado'] = df['score'].apply(lambda x: (x - df['score'].mean()) / df['score'].std())
```



MANIPULACIÓN DE LA ESTRUCTURA DE UN DATAFRAME: AGREGAR O ELIMINAR COLUMNAS

Agregar columnas permite extender el dataset con nuevas features o resultados intermedios; eliminarlas simplifica la estructura, mejora la legibilidad y reduce el almacenamiento.

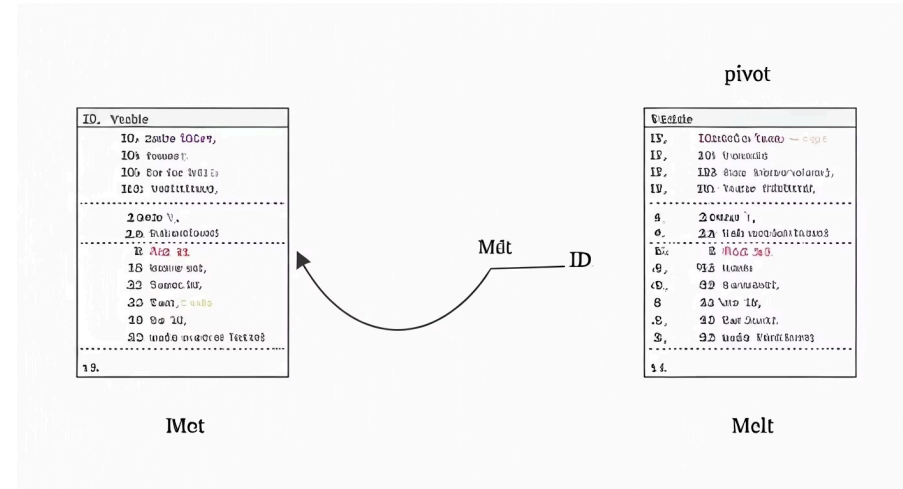
```
df['relacion'] = df['ventas'] / df['visitas']  
df = df.drop(['columna_antigua'], axis=1)
```

REDIMENSIÓN DE UN DATAFRAME

La redimensión incluye operaciones como pivotar, aplanar, transponer o "unpivotar" datos.

```
df_pivot = df.pivot(index='fecha', columns='producto',  
values='ventas')  
df_long = df.melt(id_vars=['fecha'], var_name='producto',  
value_name='ventas')
```

Estas operaciones son esenciales en análisis multivariado, reporting y dashboards.



OPERACIONES ADICIONALES EN DATAFRAMES



Convertir el tipo de dato de una columna

Asegurar los tipos correctos evita errores y mejora la eficiencia de los análisis.

```
df['codigo'] =  
df['codigo'].astype(str)  
df['fecha'] =  
pd.to_datetime(df['fecha'])
```



Definir y resetear índices

Un índice eficiente mejora la selección, el join y la agregación de datos.

```
df.set_index('id', inplace=True)  
df.reset_index(inplace=True)
```



Renombrar columnas e índices

La claridad en los nombres reduce errores y facilita la colaboración entre equipos.

```
df.rename(columns={'antigua':  
'nueva'}, inplace=True)
```


EFICIENCIA Y BUENAS PRÁCTICAS EN DATA WRANGLING EN PROYECTOS REALES

En el mundo real, donde los datasets pueden superar fácilmente el millón de registros y la colaboración entre equipos es la norma, la eficiencia y las buenas prácticas en wrangling son tan importantes como el conocimiento técnico de las funciones.

PROCESAMIENTO EFICIENTE Y ESCALABLE



Carga selectiva

Cuando trabajes con archivos grandes, carga solo las columnas y filas necesarias (usecols, nrows en read_csv).



Tipado adecuado

Define los tipos de datos al leer el archivo para ahorrar memoria (dtype en read_csv).



Operaciones vectorizadas

Prefiere operaciones de Pandas y NumPy sobre bucles explícitos en Python; estas aprovechan implementaciones en C subyacente.



Trabajo por lotes

Divide la carga de archivos grandes en fragmentos (chunksize) para evitar desbordamientos de memoria.



Persistencia eficiente

Guarda datasets intermedios en formatos compactos y rápidos como Parquet o HDF5.

BUENAS PRÁCTICAS ADICIONALES

Trazabilidad y reproducibilidad

- Scripts versionados: Automatiza pipelines con scripts versionados en git.
- Parámetros explícitos: Centraliza los parámetros (columnas clave, umbrales) en archivos de configuración.
- Seeds fijos: Define semillas para operaciones aleatorias, asegurando reproducibilidad en muestreos y splits.
- Logging: Registra los pasos, las transformaciones y los registros afectados; permite auditar y depurar.

Escalabilidad y colaboración

- Pipelines modulares: Divide los pipelines en etapas (limpieza, transformación, enriquecimiento, validación) para facilitar el testing y el reuso.
- Notebooks solo para prototipos: Usa Jupyter o similares para exploración, pero migra procesos robustos a scripts o pipelines reproducibles.
- Validación y testing: Implementa tests unitarios y validaciones automáticas (tipos, rangos, unicidad) antes de aplicar transformaciones masivas.

Documentación y seguridad

- Documenta cada transformación: Explica por qué se eliminó una columna o se discretizó una variable.
- Respaldo y versionado: Antes de eliminar o transformar datos críticos, haz copias de seguridad y usa versionado de datos (DVC, MLflow).
- Gestión de errores: Anticipa y captura excepciones, sobre todo en operaciones masivas o de integración de fuentes externas.

Automatización y monitoreo

- Automatiza tareas recurrentes: Usa frameworks como Airflow, Luigi o scripts agendados para tareas periódicas de wrangling.
- Monitorea la calidad de datos: Configura alertas si aparecen más valores nulos, duplicados o outliers de lo esperado; la calidad de los datos es dinámica y debe monitorearse.
- Auditoría y rollback: Siempre ten la capacidad de auditar y revertir cambios masivos.

CONCLUSIÓN

Data wrangling es mucho más que limpiar datos: es diseñar flujos robustos, eficientes y auditables que aseguren la calidad, la riqueza y la estructura de los datos antes de cualquier análisis o modelado. En la práctica profesional, dominar estas técnicas—y aplicarlas con eficiencia y buenas prácticas—es el diferenciador clave entre soluciones analíticas frágiles y sistemas de inteligencia robustos y sostenibles.

La eficiencia técnica, la reproducibilidad y la trazabilidad no solo optimizan los recursos, sino que minimizan errores, aceleran el ciclo de vida del dato y fortalecen la confianza en los resultados de negocio. Por eso, el data wrangling no es solo un prerequisite técnico, sino una disciplina estratégica dentro de la ingeniería y ciencia de datos contemporánea.

