

Limpieza y Transformación de Datos - Día 3

Change Status

pending

 40 min

 Learning Objectives

- 1 Identificar y resolver problemas comunes de calidad de datos en datasets reales
- 2 Aplicar técnicas de limpieza sistemática: duplicados, formatos, normalización
- 3 Implementar transformaciones básicas que preparan datos para análisis

Theory

Practice

Quiz

Evidence

 Actividades y Aprendizajes

Aprende todo sobre funciones y módulos en Python con ejemplos prácticos.

Task 1: Diagnóstico de Calidad de Datos (10 minutos)

La limpieza de datos no es un paso opcional, sino el **foundation mismo del análisis de datos**. Datos sucios llevan inevitablemente a conclusiones erróneas, y la capacidad de detectar y corregir problemas de calidad es una de las habilidades más valiosas en análisis de datos.

Los Cinco Pecados Capitaes de los Datos

Duplicados: Registros idénticos o esencialmente iguales que contaminan análisis estadísticos y distorsionan métricas.

Datos faltantes: Valores ausentes que pueden causar errores en cálculos o sesgar resultados si no se manejan apropiadamente.

Formatos inconsistentes: Fechas en formatos diferentes, números con separadores variados, texto con capitalización irregular.



Valores inválidos: Datos que violan reglas de negocio o restricciones lógicas (edades negativas, fechas futuras inválidas).

Outliers extremos: Valores que, aunque técnicamente válidos, distorsionan análisis por ser estadísticamente improbables.

Estrategia Sistemática de Diagnóstico

Inspección visual inicial: `df.head()`, `df.tail()`, `df.sample()` - primeras impresiones de la estructura y contenido.

Análisis de tipos: `df.dtypes`, `df.info()` - verificar que los tipos de datos correspondan al contenido real.

Estadísticas descriptivas: `df.describe()` - detectar valores extremos o distribuciones inesperadas.

Conteo de valores únicos: `df.nunique()`, `df['columna'].value_counts()` - identificar categorías inesperadas.

Búsqueda de patrones: Expresiones regulares para detectar formatos inconsistentes o valores inválidos.

Task 2: Operaciones de Limpieza Básicas (12 minutos)

La limpieza de datos sigue un **workflow sistemático** que transforma datos crudos en información confiable y consistente.

Eliminación de Duplicados

Detección: `df.duplicated()` identifica filas duplicadas completas.

Análisis: `df[df.duplicated()]` permite inspeccionar qué se considera duplicado.

Eliminación: `df.drop_duplicates()` mantiene la primera ocurrencia, elimina las demás.

Duplicados parciales: `df.drop_duplicates(subset=['columna1', 'columna2'])` para duplicados basados en columnas específicas.

Corrección de Tipos de Datos

Conversión automática: `pd.to_numeric()`, `pd.to_datetime()` convierten strings a tipos apropiados.

Manejo de errores: Parámetro `errors='coerce'` convierte valores inválidos a NaN.

Tipos categóricos: `df['columna'].astype('category')` para variables categóricas con dominio limitado.



Normalización de Texto

Capitalización consistente: `df['columna'].str.lower()`, `df['columna'].str.title()`.

Eliminación de espacios: `df['columna'].str.strip()` remueve espacios al inicio y fin.

Reemplazo de patrones: `df['columna'].str.replace()` para correcciones sistemáticas.

Task 3: Transformaciones Básicas de Datos (8 minutos)

Las transformaciones convierten datos limpios en información más útil y analizable.

Creación de Columnas Calculadas

Operaciones aritméticas: `df['total'] = df['precio'] * df['cantidad']` - cálculos directos.

Funciones de strings: `df['dominio'] = df['email'].str.split('@').str[1]` - extracción de componentes.

Funciones condicionales: `np.where()`, `.map()` para lógica if-then-else.

Codificación y Categorización

Bining numérico: `pd.cut()` divide valores continuos en categorías discretas.

Label encoding: `df['categoria_codificada'] = df['categoria'].astype('category').cat.codes`.

One-hot encoding: `pd.get_dummies()` para variables categóricas en análisis estadístico.

Reformateo de Datos

Pivot y melt: `df.pivot()`, `df.melt()` para cambiar entre formatos wide y long.

Stack y unstack: Reorganizar índices jerárquicos.

Transpose: `df.T` para rotar la tabla completa.