Tratamiento de Valores Perdidos y Outliers en Análisis de Análisis de Datos

La calidad de los datos es la piedra angular en cualquier proyecto de análisis, modelado o toma de decisiones automatizadas. En contextos reales, los datasets rara vez vienen completos y limpios: registros faltantes, inconsistencias y valores atípicos aparecen constantemente en fuentes como sensores, transacciones financieras o plataformas digitales. La correcta identificación y tratamiento de valores perdidos (missing values) y outliers es una etapa crítica para evitar sesgos, distorsiones y pérdidas de valor en el pipeline de Data Science, Machine Learning o ingeniería de datos.

En la industria, ignorar este proceso puede generar desde interpretaciones erróneas hasta riesgos financieros o reputacionales. Además, el manejo de estos problemas es transversal: afecta el entrenamiento de modelos, la visualización de datos y la confiabilidad de los reportes. Abordaremos tanto técnicas clásicas como modernas, desde la eliminación simple hasta algoritmos de imputación y detección avanzados, reforzando la conexión con áreas como estadística aplicada, aprendizaje automático y producción de pipelines robustos.

R **por Kibernum Capacitación S.A.**

Preguntas de Activación

Riesgos de valores perdidos y outliers

¿Qué riesgos puede generar la presencia de valores perdidos y outliers en los resultados de un análisis predictivo en tu área profesional?

Estrategias para valores faltantes

¿Es preferible eliminar, imputar o simplemente ignorar los valores faltantes? ¿Por qué?

Experiencia con outliers

¿Puedes recordar un caso real (propio o conocido) donde un dato "extraño" (outlier) haya cambiado el sentido de una decisión?

¿Qué es un Valor Perdido?

Definición

Un valor perdido es cualquier dato que no está presente en un registro, ya sea por omisión, error, política o falta de aplicabilidad.

Causas comunes

- Errores de captura
- Deterioro de datos
- Políticas empresariales
- Campos no aplicables

Analogía

Imagina llenar una encuesta: si olvidas tu edad o dejas en blanco la ciudad, ese "hueco" representa un valor perdido.

Ejemplo práctico

En un dataset bancario podemos encontrar distintos tipos de valores perdidos representados como:

- None
- NaN
- Cadena vacía ("")
- Valores especiales (-999)

Identificación de Valores Perdidos en Python

Para identificar valores perdidos en Python, utilizamos principalmente la biblioteca Pandas con sus funciones especializadas:

Funciones principales

- .isnull() - Identifica valores nulos
- .isna() - Alternativa para identificar valores nulos
- .sum() - Suma los valores nulos para contarlos

Reconocimiento de marcadores

Es importante identificar marcadores especiales que representan valores perdidos:

- NA, NULL
- Valores como -999
- Símbolos como "?"

Parámetro na_values

Al cargar datos, es fundamental definir el parámetro na_values para reconocer correctamente los valores perdidos desde el inicio.

```python
import pandas as pd

data = {
    'Nombre': ['Ana', 'Luis', 'Marta', 'Pedro'],
    'Edad': [32, None, 28, 44],
    'Ingreso': [2500, 3200, None, 4100],
    'Ciudad': ['Madrid', None, 'Sevilla', 'Málaga'],
    'Estado Civil': ['Soltera', 'Casado', None, 'Casado']
}

df = pd.DataFrame(data)

print(df.isnull())
print(df.isnull().sum())
```

El conteo de valores perdidos por columna permite diagnosticar el problema.

Estrategias de Tratamiento: Eliminación vs Imputación

Eliminación de datos

- Drop de filas completas (dropna())
- Columnas con umbrales (dropna(axis=1, thresh=...))
- Eliminación condicional (how='all', how='any')

Ventajas

- Simplicidad de implementación
- Evita introducir supuestos

Desventajas

- Riesgo de pérdida de información
- Posible introducción de sesgo

Imputación de datos

Reemplazo de valores perdidos por estimaciones:

- Media, mediana, moda (simple)
- Imputación condicional (por subgrupos)
- Imputación por modelos (regresión, KNN, MICE)

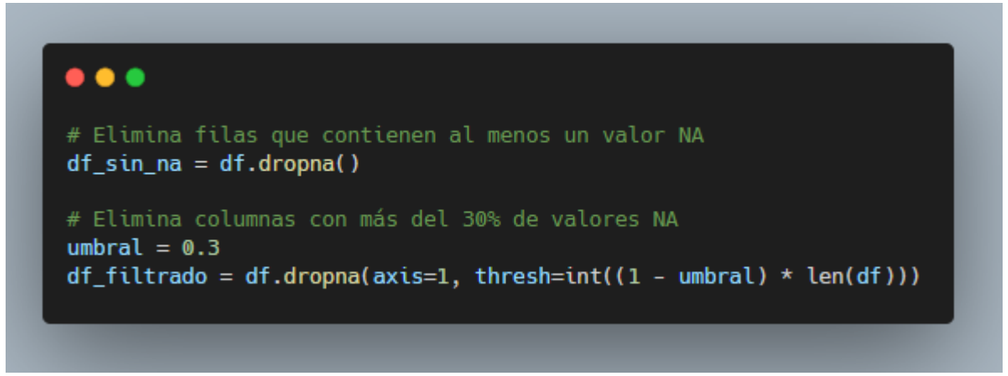
Ventajas

- Conserva el tamaño del dataset
- Puede capturar patrones subyacentes

Desventajas

- Potencial introducción de sesgo
- Subestimación de la varianza

Código de Eliminación de Datos



```python
# Elimina filas que contienen al menos un valor NA
df_sin_na = df.dropna()

# Elimina columnas con más del 30% de valores NA
umbral = 0.3
df_filtrado = df.dropna(axis=1, thresh=int((1 - umbral) * len(df)))
```

La eliminación de datos es una estrategia simple pero efectiva en ciertos casos. Podemos eliminar filas completas que contengan al menos un valor NA utilizando el método dropna() sin parámetros. También podemos ser más selectivos y eliminar columnas que tengan más de cierto porcentaje de valores perdidos, como se muestra en el ejemplo con el umbral del 30%.

Esta técnica es especialmente útil cuando:

- La cantidad de valores perdidos es pequeña respecto al total
- Los datos faltantes están distribuidos aleatoriamente (MCAR)
- No queremos introducir sesgos con imputaciones

Sin embargo, debemos ser cautelosos, ya que podríamos perder información valiosa si eliminamos demasiados registros.

Código de Imputación Básica

```python
# Reemplaza valores faltantes en 'Edad' con la media
df['Edad'] = df['Edad'].fillna(df['Edad'].mean())

# Reemplaza valores faltantes en 'Ingreso' con la mediana
df['Ingreso'] = df['Ingreso'].fillna(df['Ingreso'].median())

# Reemplaza valores faltantes en 'Ciudad' con la moda
df['Ciudad'] = df['Ciudad'].fillna(df['Ciudad'].mode()[0])
```

Imputación por media

Utilizamos la media para la variable 'Edad', lo cual es adecuado cuando la distribución es aproximadamente normal y no hay outliers significativos.

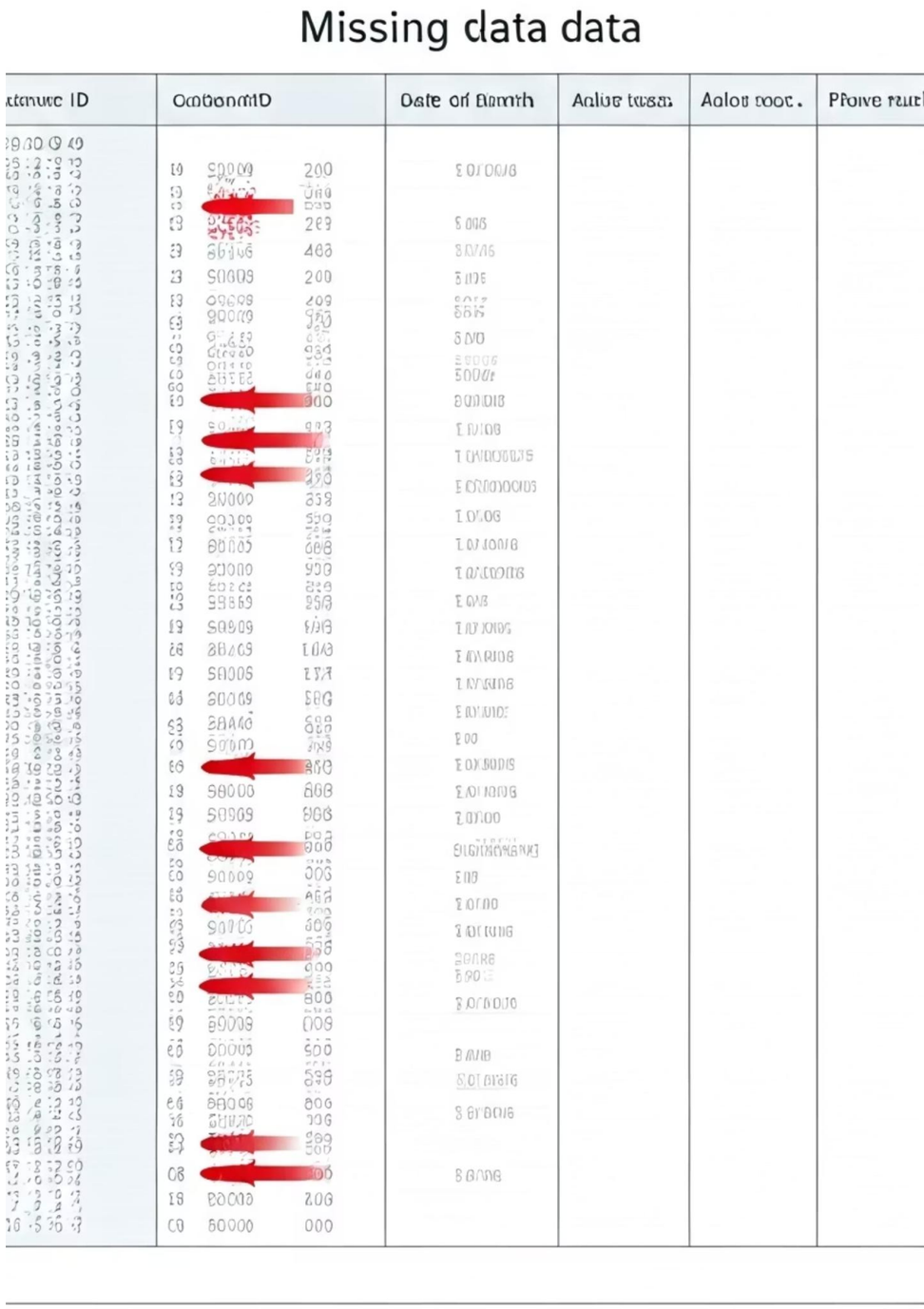
Imputación por mediana

Para 'Ingreso' usamos la mediana, que es más robusta frente a outliers y distribuciones sesgadas, comunes en variables financieras.

Imputación por moda

En variables categóricas como 'Ciudad', la moda (valor más frecuente) es la estrategia más común de imputación.

Estas técnicas son rápidas y fáciles de implementar, pero pueden introducir sesgos o subestimar la varianza real de los datos.



Imputación Avanzada

Imputación Multivariante: MICE

Multiple Imputation by Chained Equations:

- Considera correlaciones entre múltiples variables
- Utiliza IterativeImputer de scikit-learn
- Genera múltiples imputaciones para capturar incertidumbre

```
from sklearn.impute import KNNImputer

imputer = KNNImputer(n_neighbors=3)

df[['Edad', 'Ingreso']] = imputer.fit_transform(df[['Edad', 'Ingreso']])
```

Imputación con Machine Learning

- KNNImputer: utiliza vecinos cercanos
- Regresión: predice valores faltantes
- Árboles de decisión: captura relaciones no lineales

Ventajas

- Captura patrones complejos
- Útil en datos no MCAR (Missing Completely At Random)

Desventajas

- Mayor costo computacional
- Requiere preprocesamiento

Imputación de Variables Cualitativas

Estrategias principales

- Imputación por moda (valor más frecuente)
- Creación de nueva categoría ("Desconocido")
- Imputación por similitud (basada en otras variables)

Consideraciones

- Preservar la distribución original
- Evaluar el impacto en análisis posteriores
- Documentar la estrategia utilizada

```
df['Ciudad'] = df['Ciudad'].fillna('Desconocido')
df['Estado Civil'] = df['Estado Civil'].fillna(df['Estado Civil'].mode()[0])
```

En el primer caso, creamos una nueva categoría "Desconocido" para los valores faltantes en 'Ciudad', lo que permite identificar claramente estos registros en análisis posteriores. En el segundo caso, utilizamos la moda para imputar los valores faltantes en 'Estado Civil', asumiendo que el valor más común es una buena aproximación.

La elección entre estas estrategias depende del contexto del problema, la cantidad de valores faltantes y el impacto potencial en el análisis.

¿Qué es un Outlier?

Definición

Un outlier es una observación que se aleja significativamente del patrón general de los datos.

Causas

- Errores de medición o registro
- Fraudes o comportamientos anómalos
- Eventos extraordinarios
- Procesos reales pero infrecuentes

Tipos de Outliers

Univariantes

Valores extremos en una sola variable

Multivariantes

Combinaciones anómalas de variables que individualmente parecen normales

Contextuales

Valores anómalos solo en ciertos contextos (ej. temporales, geográficos)

Detección de Outliers: Métodos Estadísticos

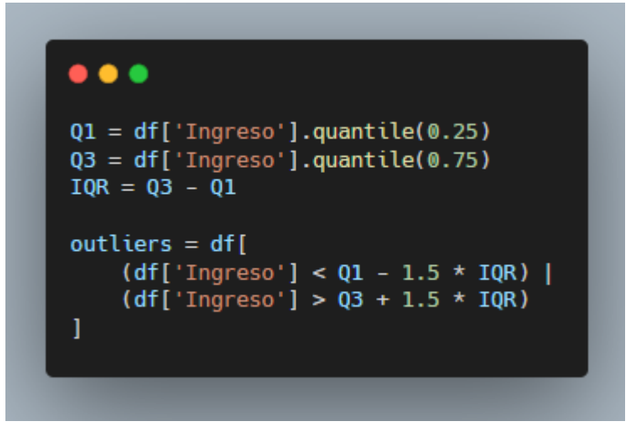
Regla de 1.5 x IQR

Identifica como outliers los valores que están:

- Por debajo de Q1 - 1.5 * IQR
- Por encima de Q3 + 1.5 * IQR

Donde:

- Q1: Primer cuartil (percentil 25)
- Q3: Tercer cuartil (percentil 75)
- IQR: Rango intercuartílico (Q3 - Q1)



```
Q1 = df['Ingreso'].quantile(0.25)
Q3 = df['Ingreso'].quantile(0.75)
IQR = Q3 - Q1

outliers = df[
    (df['Ingreso'] < Q1 - 1.5 * IQR) |
    (df['Ingreso'] > Q3 + 1.5 * IQR)
]
```

Z-score

Identifica como outliers los valores que están a más de cierto número de desviaciones estándar de la media (típicamente 3).

```
from scipy import stats
import numpy as np

z_scores = np.abs(stats.zscore(df['Ingreso'].dropna()))
outliers = df[z_scores > 3]
```

Esta técnica asume que los datos siguen una distribución aproximadamente normal, lo cual debe verificarse previamente.

Visualización de Outliers

Boxplot

Muestra la distribución de los datos y marca los outliers como puntos individuales fuera de los "bigotes".

Scatter Plot

Útil para detectar outliers multivariantes, mostrando relaciones entre dos variables.

Histogramas

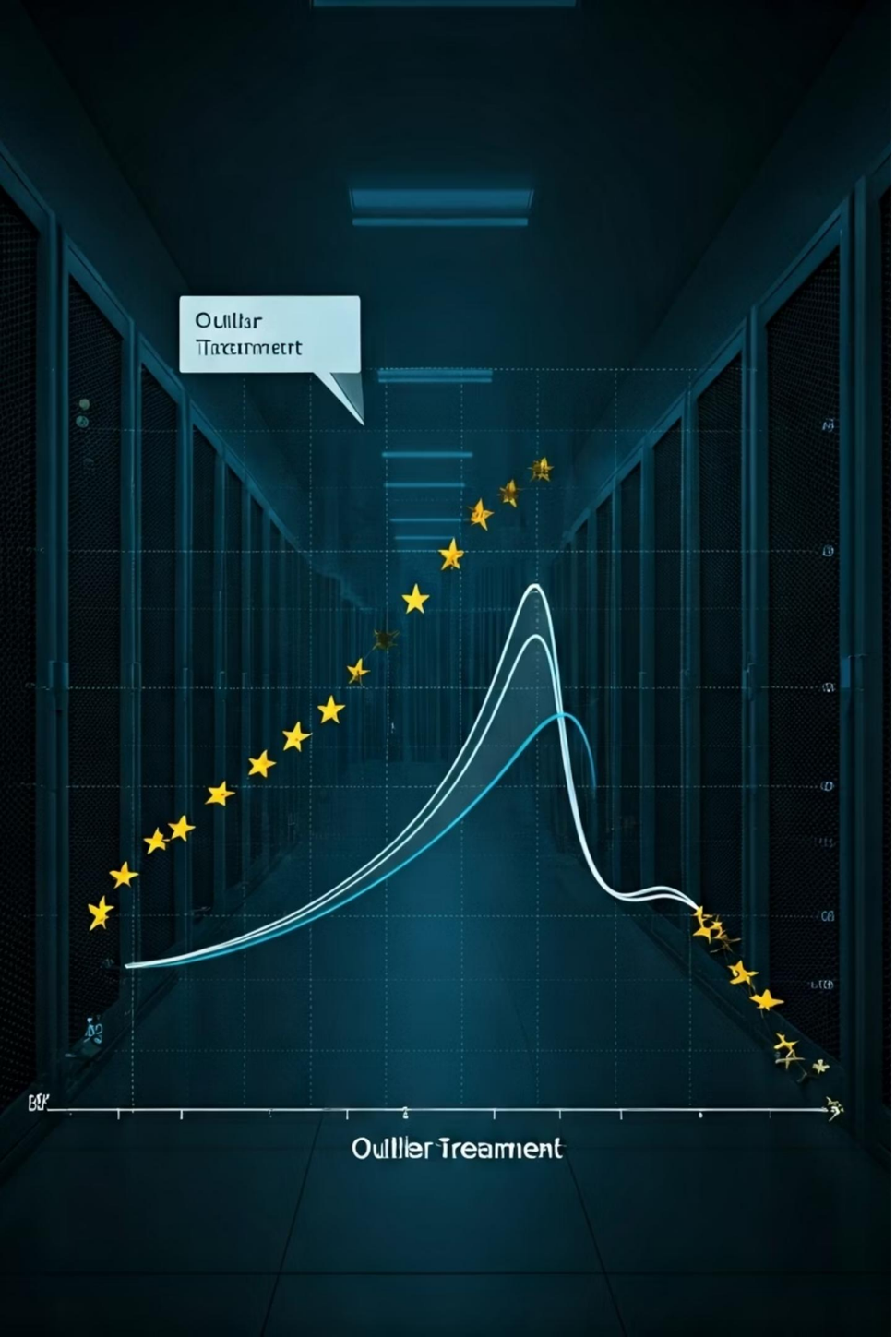
Permiten visualizar la distribución completa y detectar valores que se alejan del patrón general.

```
import matplotlib.pyplot as plt

plt.boxplot(df['Ingreso'].dropna())
plt.title('Boxplot de Ingresos')
plt.show()
```

Técnicas Avanzadas

- Isolation Forest: algoritmo basado en árboles que aísla observaciones anómalas
- DBSCAN: agrupamiento basado en densidad que identifica puntos en regiones de baja densidad
- LOF (Local Outlier Factor): mide la desviación local de densidad de un punto respecto a sus vecinos



Tratamiento de Outliers

Eliminación

Remover completamente los outliers del dataset.

- Ventaja: simple y efectivo
- Desventaja: pérdida de información potencialmente valiosa

Transformación

Aplicar funciones matemáticas para reducir el impacto de los outliers.

- Transformación logarítmica
- Winsorización (recortar valores extremos)
- Transformación Box-Cox

Imputación

Reemplazar outliers por valores más representativos.

- Percentiles (ej. P95 para valores extremos superiores)
- Métodos estadísticos robustos

Es fundamental evaluar si el outlier es un error o representa un fenómeno relevante antes de decidir su tratamiento. En algunos casos, los outliers contienen información valiosa sobre eventos raros pero importantes.

Pipeline Integrado y Producción

Diseño de pipelines reproducibles

Crear flujos de trabajo automatizados con scikit-learn y pandas que puedan ejecutarse consistentemente.

Carga y validación inicial

Incorporar verificaciones de calidad desde el momento de la carga de datos.

Detección y limpieza

Implementar métodos robustos para identificar y tratar valores perdidos y outliers.

Registro y versionado

Documentar cada transformación y mantener versiones de los datos en cada etapa.

Buenas Prácticas

- Testing: verificar que las transformaciones funcionan como se espera
- Configuración externa: parametrizar umbrales y estrategias
- Alertas: notificar cuando se detectan problemas graves
- Logs: registrar todas las operaciones para auditoría

Código de Pipeline Simplificado

```python
from sklearn.impute import KNNImputer, SimpleImputer
import pandas as pd

def preprocesar_sensores(filepath):
    df = pd.read_csv(filepath)

    # Eliminar filas con más del 50% de valores nulos
    df = df[df.isnull().mean(axis=1) < 0.5]

    # Eliminar outliers en columnas numéricas
    for col in ['temperatura', 'presion']:
        Q1 = df[col].quantile(0.25)
        Q3 = df[col].quantile(0.75)
        IQR = Q3 - Q1
        df = df[(df[col] >= Q1 - 1.5 * IQR) & (df[col] <= Q3 + 1.5 * IQR)]

    # Imputación
    num_cols = ['temperatura', 'presion']
    cat_cols = ['estado']

    df[num_cols] = KNNImputer(n_neighbors=3).fit_transform(df[num_cols])
    df[cat_cols] = SimpleImputer(strategy='most_frequent').fit_transform(df[cat_cols])

    return df
```

Este pipeline simplificado realiza las siguientes operaciones:

1. Carga los datos desde un archivo CSV
2. Elimina filas con más del 50% de valores nulos
3. Detecta y elimina outliers en las variables numéricas usando la regla del IQR
4. Imputa valores faltantes en variables numéricas usando KNN
5. Imputa valores faltantes en variables categóricas usando la moda
6. Retorna el dataframe limpio y procesado

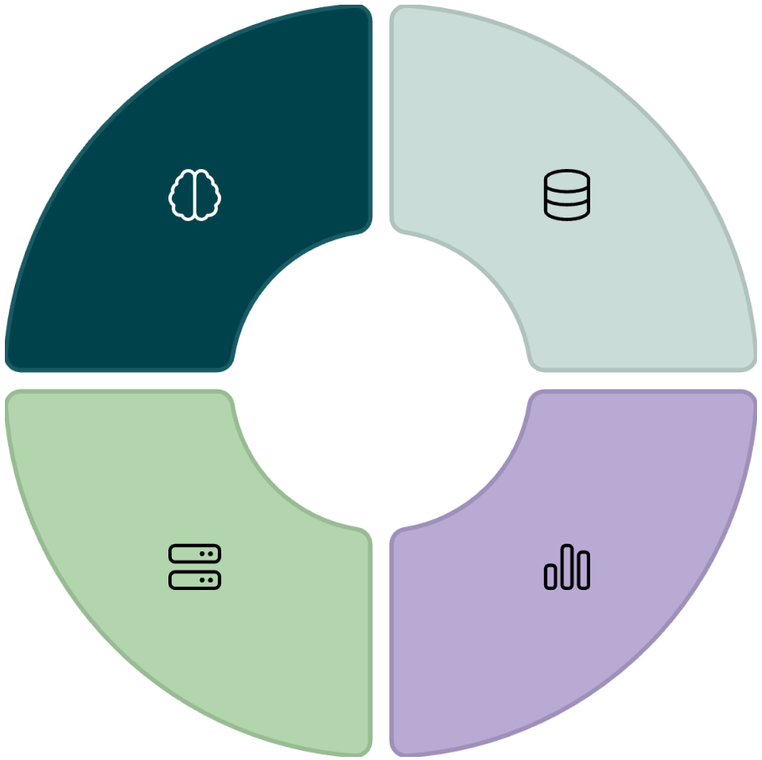
Conexiones Interdisciplinarias

Machine Learning

La calidad de los datos impacta directamente en el rendimiento de los modelos. Algunos algoritmos son más sensibles a outliers (ej. regresión lineal) mientras que otros son más robustos (ej. árboles de decisión).

Operaciones y Despliegue

Los pipelines de limpieza deben integrarse en los flujos de trabajo de producción, con monitoreo continuo y alertas automatizadas.



Big Data

En entornos de datos masivos, la detección y tratamiento de valores anómalos requiere técnicas escalables y distribuidas.

Visualización

Las técnicas de visualización son fundamentales tanto para detectar problemas como para comunicar resultados de forma efectiva.

Ejercicio Propuesto

Limpieza y preprocesamiento de un dataset real de sensores industriales

Objetivo

Desarrollar un pipeline en Python que detecte, trate y documente valores perdidos y outliers en un archivo de sensores, generando un reporte reproducible.

Contexto

Recibes un archivo CSV ("sensores.csv") con variables numéricas ("temperatura", "presion") y categóricas ("estado"). El archivo contiene valores perdidos, marcadores especiales (NA, -999) y outliers.

Requerimientos técnicos

- Python 3.x, pandas, scikit-learn
- Colab/Jupyter Notebook

Modalidad

Individual

Tiempo estimado

90 minutos

Entregable

- Notebook ejecutable con código, gráficos, comentarios y archivo final
- Informe breve (máx. 1 pág) justificando cada decisión de limpieza

Instrucciones del Ejercicio

1

Carga y visualización

Cargar el archivo y visualizar resumen de valores perdidos y outliers por variable.

2

Reemplazo de marcadores

Reemplazar marcadores especiales por NaN al cargar el archivo.

3

Eliminación condicional

Descartar filas con más del 50% de datos nulos.

4

Detección de outliers

Detectar outliers en variables numéricas usando la regla de 1.5 x IQR.

5

Eliminación de outliers

Eliminar outliers extremos.

6

Imputación numérica

Imputar valores numéricos con KNN (n_neighbors=3).

7

Imputación categórica

Imputar variables categóricas con la moda.

8

Visualización comparativa

Visualizar resultado antes/después (boxplot, tabla resumen).

9

Documentación

Documentar el pipeline y justificar decisiones.

10

Exportación

Guardar el resultado limpio a un nuevo CSV.

Solución Esperada (Resumen)

Carga de datos con na_values

Utilizar el parámetro na_values para identificar correctamente todos los tipos de valores perdidos desde el inicio.

Eliminación de filas

Descartar registros con más del 50% de valores nulos para mantener la integridad del dataset.

Detección y eliminación de outliers

Aplicar la regla del IQR para identificar y eliminar valores extremos que podrían distorsionar el análisis.

Imputación KNN y moda

Utilizar técnicas avanzadas para variables numéricas y la moda para categóricas.

Visualización comparativa

Mostrar el impacto de la limpieza mediante gráficos antes/después.

Código Sugerido para el Ejercicio

Este código implementa los pasos principals del ejercicio propuesto:

1. Carga el dataset con reconocimiento de valores NA y -999 como nulos
2. Filtra filas con más del 50% de valores perdidos
3. Detecta y elimina outliers en las variables numéricas
4. Imputa valores faltantes usando KNN para variables numéricas
5. Imputa valores faltantes usando la moda para variables categóricas
6. Visualiza los resultados con un boxplot

```python
import pandas as pd
from sklearn.impute import KNNImputer, SimpleImputer
import matplotlib.pyplot as plt

# Cargar datos y considerar valores faltantes
df = pd.read_csv('sensores.csv', na_values=['NA', '-999'])

# Filtrar filas con menos del 50% de datos faltantes
df = df[df.isnull().mean(axis=1) < 0.5]

# Eliminación de outliers usando IQR para columnas temperatura y presion
for col in ['temperatura', 'presion']:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    df = df[(df[col] >= Q1 - 1.5 * IQR) & (df[col] <= Q3 + 1.5 * IQR)]

# Imputar valores faltantes con KNN para temperatura y presion
df[['temperatura', 'presion']] = KNNImputer(n_neighbors=3).fit_transform(df[['temperatura',
'presion']])

# Imputar valores faltantes para la columna estado usando la moda
df['estado'] = SimpleImputer(strategy='most_frequent').fit_transform(df[['estado']])

# Visualizar boxplot de temperatura
plt.boxplot(df['temperatura'])
plt.show()
```

Anexos Técnicos y Referencias

Documentación Oficial

- Documentación pandas
- Documentación scikit-learn imputers

Libros Recomendados

- Python for Data Analysis (Wes McKinney)
- Hands-On Machine Learning with Scikit-Learn & TensorFlow
- Feature Engineering for Machine Learning

Recursos Online

- Kaggle: tutoriales sobre limpieza de datos
- Towards Data Science: artículos sobre tratamiento de outliers
- Stack Overflow: soluciones a problemas comunes

Herramientas Complementarias

- Great Expectations: validación de datos
- Pandas Profiling: análisis exploratorio automatizado
- MLflow: seguimiento de experimentos

Preguntas Abiertas de Reflexión

1 Limitaciones de los métodos

¿Qué limitaciones identificaste en los métodos de imputación y detección de outliers? ¿Cuándo podría ser riesgoso aplicar cada técnica?

2 Automatización y auditoría

¿Cómo puedes automatizar y auditar los procesos de limpieza de datos en un pipeline real?

3 Impacto en producción

¿De qué manera podría impactar una mala gestión de valores perdidos/outliers en un modelo de Machine Learning desplegado en producción?

Estas preguntas buscan promover una reflexión crítica sobre las técnicas aprendidas y su aplicación en contextos reales, considerando las implicaciones prácticas y éticas de las decisiones tomadas durante el proceso de limpieza de datos.

Resumen: 5 Puntos Clave

1

Calidad de datos

La calidad de los datos es esencial para análisis y modelos robustos.

Identificación

Los valores perdidos pueden ser explícitos o implícitos; su identificación es crítica.

Tratamiento

El tratamiento incluye eliminación e imputación, desde estrategias simples hasta avanzadas (KNN, MICE).

Outliers

Los outliers distorsionan estadísticas; su detección requiere métodos estadísticos y visualización.

Pipeline

El pipeline de limpieza debe ser reproducible, documentado y auditable para la producción.

Conclusiones y Próximos Pasos

Lo que hemos aprendido

- Identificación y tratamiento de valores perdidos
- Técnicas de imputación básicas y avanzadas
- Detección y manejo de outliers
- Diseño de pipelines robustos
- Conexiones con otras áreas de la ciencia de datos

Próximos pasos

- Profundizar en técnicas específicas para tipos de datos complejos
- Explorar métodos de validación de la calidad de datos
- Implementar pipelines automatizados en entornos de producción
- Evaluar el impacto de las estrategias de limpieza en modelos predictivos

La limpieza y preparación de datos es un proceso continuo que requiere tanto conocimientos técnicos como criterio profesional. Las decisiones tomadas en esta etapa tienen un impacto directo en la calidad y fiabilidad de los resultados analíticos y predictivos.