MANEJO DE VALORES PERDIDOS Y OUTLIERS

M3: OBTENCIÓN Y PREPARACIÓN DE DATOS

AE4: APLICAR TÉCNICAS PARA LA IDENTIFICACIÓN Y TRATAMIENTO DE VALORES PERDIDOS EN UN SET DE DATOS UTILIZANDO LIBRERÍAS DE PYTHON PARA LA RESOLUCIÓN DE UN PROBLEMA.

alkemy

Introducción👩‍💻

El análisis de datos es un proceso crítico en la toma de decisiones en múltiples industrias, desde el sector financiero hasta la investigación médica y la inteligencia artificial. Uno de los principales desafíos en este proceso es la **calidad de los datos**, ya que un conjunto de datos incompleto o contaminado con valores atípicos puede llevar a conclusiones erróneas y modelos predictivos poco precisos.

Dos de los problemas más comunes que afectan la calidad de los datos son los **valores perdidos** y los **outliers**. Los valores perdidos pueden surgir por diversas razones, como fallos en la recopilación de datos, errores de ingreso o incluso restricciones en la captura de información. Ignorar estos valores puede generar análisis sesgados o incluso hacer que algunos modelos de machine learning fallen por completo.

Por otro lado, los **outliers** o valores atípicos son datos que se desvían significativamente de la tendencia general del conjunto de datos. Estos valores pueden ser resultado de errores en la medición o representar información valiosa sobre eventos extremos o anomalías en el fenómeno estudiado. Identificar y manejar adecuadamente los outliers es fundamental para mejorar la precisión de los modelos analíticos y garantizar interpretaciones confiables.

Aprendizaje esperado

Cuando finalices la lección serás capaz de:

1. Comprender qué son los valores perdidos y por qué ocurren en los conjuntos de datos.
2. Identificar y visualizar valores perdidos en un **DataFrame** de Pandas.
3. Aplicar estrategias de filtrado y eliminación de valores perdidos según el contexto del análisis.
4. Utilizar diversas técnicas de imputación para reemplazar valores faltantes en datos cuantitativos y cualitativos.
5. Comprender qué son los **outliers**, cómo afectan el análisis de datos y cuándo deben eliminarse o conservarse.
6. Implementar métodos estadísticos y gráficos para detectar outliers en un conjunto de datos.
7. Aplicar filtros para eliminar valores atípicos y mejorar la calidad de los datos.

MANEJO DE VALORES PERDIDOS Y OUTLIERS

1. QUÉ ES UN VALOR PERDIDO

Un **valor perdido** se refiere a la ausencia de información en un conjunto de datos. En términos computacionales, Pandas representa los valores perdidos como NaN (*Not a Number*) o como valores nulos (None). Estos valores pueden aparecer por múltiples razones:

- **Errores en la captura de datos**: Sensores defectuosos, problemas en formularios digitales o pérdida de paquetes de información.
- **Falta de respuesta**: Encuestas incompletas, datos que no fueron proporcionados por los participantes.
- **Integración de bases de datos**: Fusión de varias fuentes donde algunas contienen información parcial.
- **Reglas de negocio**: Campos opcionales que no siempre se completan en sistemas empresariales.
- **Problemas de formato**: Datos mal formateados que no son reconocidos por el sistema.
- **Errores humanos**: Ocurren cuando los datos se ingresan manualmente y algunos valores se omiten por accidente.
- **Fallas tecnológicas**: En sistemas de recopilación de datos automatizados, pueden producirse errores que impidan la captura completa de información.

La presencia de valores perdidos afecta la interpretación de los datos, las visualizaciones y los modelos predictivos. Dependiendo de la cantidad de datos faltantes y la importancia de la variable afectada, es crucial tomar decisiones sobre cómo tratarlos.

Ejemplo de creación de un DataFrame con valores perdidos:

```
import pandas as pd
import numpy as np

# Creación de un DataFrame con valores nulos
datos = {'Nombre': ['Ana', 'Luis', 'Carlos', 'Sofía', np.nan],
         'Edad': [25, None, 30, 28, 22],
         'Ciudad': ['Madrid', 'Barcelona', np.nan, 'Sevilla', 'Bilbao']}

df = pd.DataFrame(datos)
print(df)
```

2. IDENTIFICACIÓN DE VALORES PERDIDOS

Para detectar valores perdidos en Pandas, se pueden utilizar diversas funciones que permiten cuantificar y visualizar la cantidad de datos ausentes en un DataFrame.

```
# Contar la cantidad de valores nulos en cada columna
print(df.isnull().sum())
```

Para visualizar la distribución de valores perdidos en un dataset grande, se pueden usar herramientas gráficas como seaborn o missingno:

```
import missingno as msno
import matplotlib.pyplot as plt

# Visualizar los valores perdidos en un gráfico de matriz
msno.matrix(df)
plt.show()
```

En este gráfico, los valores ausentes aparecen en color diferente, lo que permite identificar patrones en la distribución de los datos faltantes.

Para detectar patrones en la presencia de valores nulos en diferentes columnas:

```
msno.heatmap(df)
plt.show()
```

Este gráfico de calor muestra correlaciones entre columnas con valores faltantes, lo que puede indicar relaciones en la estructura de los datos.

3. FILTRADO DE LA DATA PERDIDA

Eliminación de Filas o Columnas con Valores Perdidos

Si los valores faltantes representan una fracción muy pequeña del conjunto de datos, se pueden eliminar para evitar sesgos.

```
# Eliminar filas con valores perdidos
df_sin_nulos = df.dropna()
```

Si una columna entera tiene un alto porcentaje de valores nulos, puede ser eliminada:

```
df_sin_columnas_nulas = df.dropna(axis=1)
```

Este método es útil cuando la eliminación no afecta significativamente la información general.

4. IMPUTACIÓN DE DATOS

En muchos casos, eliminar datos no es la mejor opción, por lo que se recurre a la imputación, es decir, completar los valores faltantes con estimaciones basadas en otros datos. La imputación de datos permite conservar la mayor cantidad posible de información y evitar la reducción del tamaño del conjunto de datos, lo que es especialmente útil en situaciones donde la eliminación de valores perdidos podría causar sesgos o afectar negativamente los análisis.

Existen diversos métodos para la imputación de datos, cada uno adecuado para distintos tipos de conjuntos de datos y contextos analíticos. La selección del método correcto depende de la naturaleza de la variable afectada, la cantidad de valores faltantes y el impacto que estos pueden tener en el análisis posterior.

Imputación con valores estadísticos

Una de las formas más comunes de imputación es el uso de valores estadísticos, donde se reemplazan los valores perdidos con medidas de tendencia central, como la media, la mediana o la moda. Esta técnica es ampliamente utilizada debido a su facilidad de implementación y efectividad en muchos casos.

- Imputación con la media: Se utiliza cuando los datos siguen una distribución normal. La media es útil para mantener la coherencia en distribuciones simétricas, aunque puede verse afectada por la presencia de outliers.
- Imputación con la mediana: Se recomienda cuando los datos contienen outliers, ya que la mediana es menos sensible a valores extremos.
- Imputación con la moda: Se emplea principalmente en datos categóricos, donde los valores faltantes se reemplazan con la categoría más frecuente.

Ejemplo de imputación con la media y la mediana:

```
import pandas as pd
import numpy as np

# Crear un DataFrame con valores faltantes
datos = {'Edad': [23, 25, np.nan, 30, 28, 22, np.nan, 29],
         'Salario': [50000, 54000, 58000, np.nan, 62000, np.nan, 66000, 70000]}

df = pd.DataFrame(datos)

# Imputar valores faltantes con la media
df['Edad'].fillna(df['Edad'].mean(), inplace=True)
df['Salario'].fillna(df['Salario'].median(), inplace=True)

print(df)
```

En este ejemplo, los valores faltantes en la columna **Edad** se reemplazan con la media de la columna, mientras que los valores de **Salario** se imputan con la mediana.

Imputación de valores categóricos con la moda

Cuando se trabaja con datos categóricos, la imputación con la moda es una estrategia común. Dado que las categorías no tienen valores numéricos, no es posible calcular una media o mediana, por lo que el método más sencillo es asignar la categoría más frecuente a los valores perdidos.

Ejemplo de imputación con la moda:

```
# Crear un DataFrame con valores categóricos faltantes
datos = {'Producto': ['Laptop', 'Tablet', None, 'Smartphone', 'Tablet'],
         'Marca': ['Dell', 'Apple', 'Samsung', None, 'Apple']}

df = pd.DataFrame(datos)

# Imputar valores faltantes con la moda
df['Producto'].fillna(df['Producto'].mode()[0], inplace=True)
df['Marca'].fillna(df['Marca'].mode()[0], inplace=True)

print(df)
```

En este caso, los valores ausentes en la columna **Producto** y **Marca** se reemplazan con la categoría más frecuente de cada columna.

Imputación con una nueva categoría

Otra estrategia para manejar valores perdidos en datos categóricos es la creación de una **nueva categoría** que represente los datos ausentes. Esto es útil en encuestas o bases de datos donde la ausencia de un valor tiene un significado particular.

```
df['Producto'].fillna('Desconocido', inplace=True)
df['Marca'].fillna('Sin Marca', inplace=True)
```

Este método permite diferenciar los valores imputados de los datos originales y es útil en modelos donde la ausencia de un dato puede ser un factor relevante.

Imputación basada en relaciones con otras variables

Cuando los valores faltantes tienen una relación con otras variables en el conjunto de datos, es posible imputarlos utilizando esas relaciones. Por ejemplo, en un dataset de salarios, el salario de una persona puede estar correlacionado con su nivel educativo o experiencia laboral.

Ejemplo de imputación basada en relaciones:

```
# Supongamos que los salarios dependen de la edad
df['Salario'].fillna(df.groupby('Edad')['Salario'].transform('mean'), inplace=True)
```

En este caso, los valores faltantes en **Salario** se imputan con el salario promedio de personas con la misma edad.

Imputación mediante modelos predictivos

Los modelos de machine learning también pueden usarse para predecir los valores faltantes. Algoritmos como regresión lineal, árboles de decisión o k-vecinos más cercanos (**KNN**) pueden estimar valores ausentes con base en otras características del conjunto de datos.

Ejemplo de imputación con KNN:

```
from sklearn.impute import KNNImputer

# Crear un imputador KNN
imputer = KNNImputer(n_neighbors=2)
df_imputado = pd.DataFrame(imputer.fit_transform(df), columns=df.columns)
```

Este método es más avanzado y requiere entrenamiento previo, pero puede ser útil en datasets complejos con múltiples relaciones entre variables.

5. IMPUTACIÓN DE VALORES CUALITATIVOS

Los valores cualitativos o categóricos en un conjunto de datos pueden contener valores faltantes, lo que puede afectar análisis exploratorios y modelos predictivos. A diferencia de los valores numéricos, que pueden completarse con estadísticas como la media o la mediana, los valores cualitativos requieren enfoques diferentes para su imputación.

Los datos cualitativos pueden representar categorías como nombres de ciudades, tipos de productos, clasificaciones o respuestas de encuestas. La ausencia de estos valores puede ser problemática, ya que afecta análisis de segmentación, modelos de machine learning y generación de reportes.

Métodos de imputación para valores cualitativos

Existen varias estrategias para manejar valores faltantes en datos cualitativos, entre ellas:

1. Imputación con la moda: Se reemplazan los valores faltantes con la categoría más frecuente en la columna.
2. Imputación con una nueva categoría: Se asigna un valor como "Desconocido" o "Otro" para representar los valores faltantes.

3. Imputación basada en agrupaciones: Se rellenan los valores en función de otra variable relacionada.
4. Imputación con modelos predictivos: Se usan modelos de machine learning para estimar los valores faltantes.

Ejemplo práctico de imputación con la moda:

```
import pandas as pd

# Crear un DataFrame con valores categóricos faltantes
datos = {'Producto': ['Laptop', 'Tablet', None, 'Smartphone', 'Tablet'],
         'Marca': ['Dell', 'Apple', 'Samsung', None, 'Apple']}

df = pd.DataFrame(datos)

# Imputar valores faltantes con la moda
df['Producto'].fillna(df['Producto'].mode()[0], inplace=True)
df['Marca'].fillna(df['Marca'].mode()[0], inplace=True)

print(df)
```

La imputación con la moda es una solución rápida y efectiva cuando los datos tienen una categoría dominante. Sin embargo, en casos donde las categorías están distribuidas equitativamente, se recomienda evaluar otros métodos.

Imputación con una nueva categoría

A veces, es útil crear una categoría específica para los valores faltantes, lo que permite diferenciarlos de los datos existentes sin introducir sesgos.

```
df['Producto'].fillna('Desconocido', inplace=True)
df['Marca'].fillna('Sin Marca', inplace=True)
```

Este método es útil en análisis donde la presencia de valores nulos puede tener un significado particular, como en encuestas o registros administrativos.

6. QUÉ ES UN OUTLIER

Un outlier o valor atípico es un dato que se aleja significativamente del resto de los valores. Puede deberse a errores en la medición o representar eventos inusuales.

Los outliers pueden detectarse utilizando métodos estadísticos:

- Regla de 1.5 veces el IQR
- Desviación estándar
- Diagramas de caja (Boxplots)

```
import numpy as np

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

limite_inferior = Q1 - 1.5 * IQR
limite_superior = Q3 + 1.5 * IQR

outliers = df[(df['Edad'] < limite_inferior) | (df['Edad'] > limite_superior)]
print(outliers)
```

7. DETECCIÓN Y FILTRADO DE OUTLIERS

¿Qué es un outlier?

Un outlier o valor atípico es una observación que se encuentra significativamente separada del resto de los datos. Estos valores pueden surgir debido a errores en la medición, fallas en los sensores, entrada incorrecta de datos o, en algunos casos, representar eventos genuinos pero extremos.

Los outliers pueden sesgar análisis estadísticos, afectar la precisión de modelos de machine learning y generar visualizaciones distorsionadas. Por ello, es fundamental identificarlos y decidir si deben eliminarse, ajustarse o conservarse.

Métodos de detección de outliers

Existen varias técnicas para detectar outliers en un conjunto de datos:

1. Método del rango intercuartil (IQR): Se basa en los percentiles 25 y 75 para identificar valores extremos.

$$[Q1 - 1.5 \times IQR, Q3 + 1.5 \times IQR]$$

Código:

```python
import pandas as pd

# Simulación de datos
data = pd.DataFrame({'valor': [10, 12, 13, 14, 15, 16, 17, 80]})

# Calcular IQR
Q1 = data['valor'].quantile(0.25)
Q3 = data['valor'].quantile(0.75)
IQR = Q3 - Q1

# Filtrar outliers
outliers = data[(data['valor'] < Q1 - 1.5 * IQR) | (data['valor'] > Q3 + 1.5 * IQR)]
print("Outliers por IQR:\n", outliers)
```

2. Z-score: Detecta valores que se alejan más de un cierto número de desviaciones estándar de la media.

Código:

Usa la media y desviación estándar. Se considera outlier un dato con un Z-score > 3 o < -3.

```python
from scipy.stats import zscore

data['z_score'] = zscore(data['valor'])
outliers_z = data[(data['z_score'] > 3) | (data['z_score'] < -3)]
print("Outliers por Z-score:\n", outliers_z)
```

3. Diagramas de caja (Boxplots): Visualizan la distribución de los datos y resaltan valores extremos.

```python
import seaborn as sns
import matplotlib.pyplot as plt

sns.boxplot(x=data['valor'])
plt.title("Boxplot de valores")
plt.show()
```

Explicación: El boxplot muestra la mediana, cuartiles y outliers como puntos fuera de los "bigotes".

4. Histogramas y gráficos de dispersión: Permiten identificar visualmente la presencia de outliers.

Histograma:

```
plt.hist(data['valor'], bins=10, color='skyblue', edgecolor='black')
plt.title("Histograma de valores")
plt.xlabel("Valor")
plt.ylabel("Frecuencia")
plt.show()
```

Explicación: Permite observar si existen valores extremos con baja frecuencia.

Gráfico de Dispersión:

```
# Simulamos un eje x
data['indice'] = range(len(data))
plt.scatter(data['indice'], data['valor'], color='orange')
plt.title("Gráfico de dispersión")
plt.xlabel("Índice")
plt.ylabel("Valor")
plt.show()
```

Explicación: Útil para detectar outliers en relación con la posición o el tiempo.

5. Modelos basados en aprendizaje automático: Algoritmos como Isolation Forest pueden detectar anomalías en conjuntos de datos grandes.

Ejemplo práctico de detección de outliers con IQR:

```
import numpy as np

# Crear un DataFrame con valores numéricos
datos = {'Edad': [23, 25, 26, 30, 120, 22, 27, 29, 24, 300]}  # 120 y 300 son outliers

df = pd.DataFrame(datos)

# Calcular los cuartiles y el rango intercuartil
Q1 = df['Edad'].quantile(0.25)
Q3 = df['Edad'].quantile(0.75)
IQR = Q3 - Q1

# Definir límites para detectar outliers
limite_inferior = Q1 - 1.5 * IQR
limite_superior = Q3 + 1.5 * IQR

# Filtrar outliers
df_sin_outliers = df[(df['Edad'] >= limite_inferior) & (df['Edad'] <= limite_superior)]
print(df_sin_outliers)
```

En este ejemplo, se eliminan los valores 120 y 300, que son significativamente diferentes del resto de los datos.

Detección de outliers con Z-score

Otra forma de detectar valores atípicos es utilizando la puntuación Z-score, que mide cuántas desviaciones estándar está un valor con respecto a la media.

```
from scipy import stats

# Calcular el Z-score de cada dato
df['Z-score'] = np.abs(stats.zscore(df['Edad']))

# Filtrar outliers basados en un umbral (por ejemplo, Z-score > 3)
df_sin_outliers = df[df['Z-score'] < 3]
print(df_sin_outliers)
```

El Z-score es útil cuando los datos siguen una distribución normal y permite ajustar el umbral de detección según la aplicación.

Eliminación de outliers

Después de identificar los outliers, se debe decidir si eliminarlos, ajustarlos o transformarlos. En algunos casos, los outliers contienen información valiosa y no deben eliminarse arbitrariamente.

```
df_limpio = df[(df['Edad'] > limite_inferior) & (df['Edad'] < limite_superior)]
```

Sin embargo, en escenarios como predicción de fraudes o detección de anomalías en redes, los outliers pueden representar casos de interés en lugar de errores.

Cierre ✍️

El manejo de valores perdidos y la imputación de datos son procesos fundamentales en el análisis y preparación de datos. La presencia de valores faltantes puede afectar significativamente la calidad de los modelos predictivos y la interpretación de los resultados. En este manual, hemos explorado diversas técnicas para identificar, eliminar e imputar valores nulos en conjuntos de datos estructurados.

La selección del método de imputación adecuado depende del tipo de dato afectado y del contexto del análisis. En el caso de datos numéricos, la imputación mediante la media, la mediana o modelos predictivos como KNN puede ser efectiva. Para datos categóricos, la moda o la asignación de una nueva categoría permiten preservar la integridad del conjunto de datos sin introducir sesgos significativos.

Es crucial que los analistas de datos evalúen el impacto de la imputación en sus modelos y realicen pruebas para determinar la estrategia más apropiada. La decisión de eliminar valores faltantes o imputarlos debe tomarse en función del porcentaje de datos ausentes y la relevancia de la variable en el estudio.

Además, la detección y el tratamiento de valores atípicos juegan un papel importante en la calidad del análisis. Los outliers pueden representar errores en los datos o, en algunos casos, eventos de interés. Utilizar herramientas como el rango intercuartílico (IQR) y el Z-score permite identificar y manejar adecuadamente estos valores atípicos para evitar sesgos en los resultados.

Referencias📚

1. Pandas Development Team. (n.d.). *Pandas documentation*. https://pandas.pydata.org/docs/
2. VanderPlas, J. (2016). *Python data science handbook: Essential tools for working with data*. O'Reilly Media.
3. McKinney, W. (2017). *Python for data analysis: Data wrangling with pandas, NumPy, and IPython* (2nd ed.). O'Reilly Media.
4. Müller, A. C., & Guido, S. (2017). *Introduction to machine learning with Python: A guide for data scientists*. O'Reilly Media.
5. Molin, S. (2019). *Hands-on data analysis with pandas: Efficiently perform data collection, wrangling, analysis, and visualization using Python*. Packt Publishing.
6. Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning: Data mining, inference, and prediction* (2nd ed.). Springer.
7. James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning: With applications in R*. Springer.

¡Muchas gracias!

Nos vemos en la próxima lección