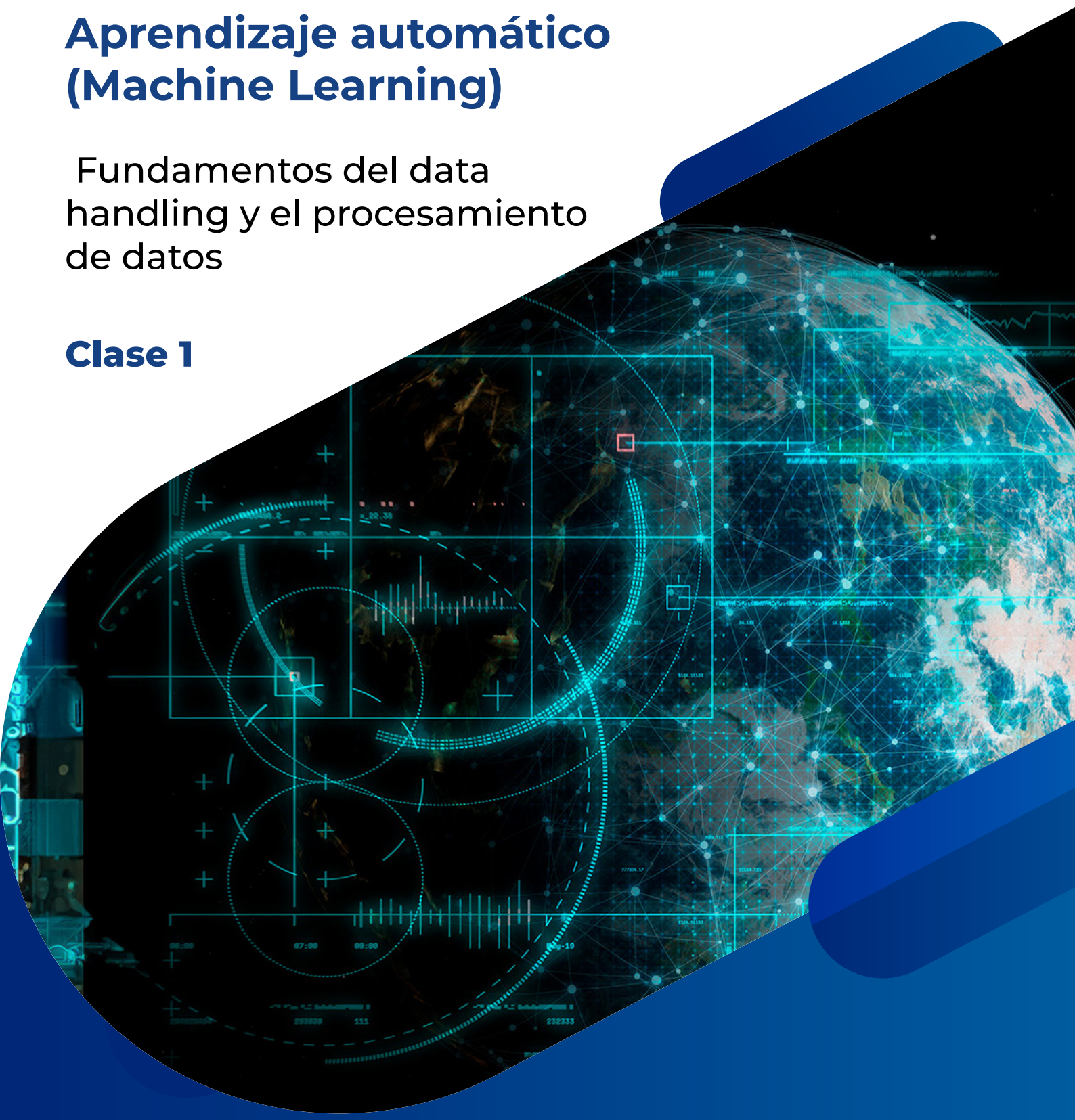
PUCE Virtual
EDUCACIÓN FLEXIBLE

Aprendizaje automático (Machine Learning)

Fundamentos del data handling y el procesamiento de datos

Clase 1



MAESTRÍA EN
SISTEMAS DE INFORMACIÓN
Mención Data Science

La excelencia no se improvisa

1. INTRODUCCIÓN DE LA CLASE

El manejo de datos, conocido como *data handling*, es una parte fundamental del flujo de trabajo en ciencia de datos y aprendizaje automático, ya que asegura que los datos sean de calidad, estén organizados y sean accesibles para su análisis posterior. Este proceso incluye tareas como la limpieza de datos, la detección de valores atípicos, la normalización y la transformación a formatos adecuados para su uso en modelos predictivos. Además, implica la comprensión de las fuentes de datos y el uso de herramientas como Python, SQL o R, para interactuar con ellos de manera eficiente (Han et al., 2022). **La calidad del manejo de datos influye directamente en la precisión y reproducibilidad de los resultados**, lo que lo convierte en un paso crítico en cualquier proyecto basado en datos (Aggarwal, 2015).

Por otro lado, el procesamiento básico de datos abarca las primeras etapas de preparación de los datos, donde se realizan operaciones esenciales como la imputación de valores faltantes, la detección y eliminación de valores atípicos y la codificación de variables categóricas. Estas tareas son necesarias para garantizar que los datos estén en un formato adecuado para el análisis o modelado posterior (James et al., 2017). Además, en esta etapa se lleva a cabo el análisis exploratorio de datos (EDA), que utiliza estadísticas descriptivas y visualizaciones para identificar patrones y relaciones claves en el conjunto de datos (Shmueli et al., 2021). **Un procesamiento adecuado no solo mejora la calidad del *dataset*, sino que también establece las bases para un análisis más profundo y confiable**.

Clase 1. Fundamentos del *data handling* y el procesamiento de datos

1. Preprocesamiento de datos

El preprocesamiento de datos es una etapa crucial en el análisis de datos y la minería de datos, ya que preparan los datos brutos para su posterior análisis. Esta fase implica una serie de técnicas y transformaciones que aseguran que los datos sean precisos, consistentes y adecuados para los modelos analíticos que se aplicarán. Sin un adecuado preprocesamiento, los resultados de los análisis pueden ser engañosos o inexactos. A continuación, se presentan algunos ejemplos de las técnicas comunes de preprocesamiento y su relevancia en diferentes contextos.

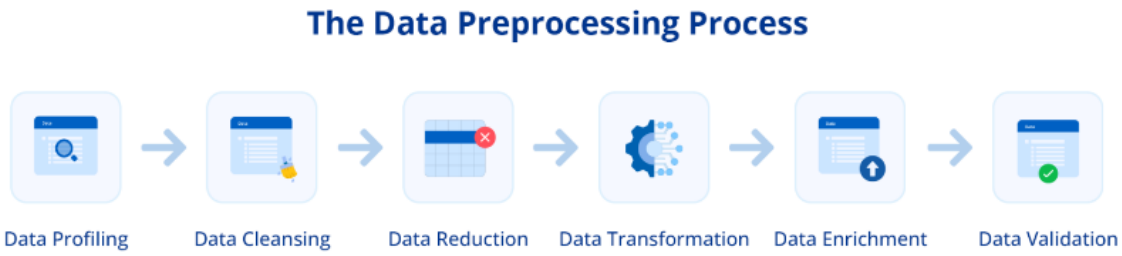


Figura N. º 1. *Data Preparation Process*.
Fuente: https://www.astera.com/es/type/blog/data-preprocessing/

Limpieza de datos

Uno de los primeros pasos en el preprocesamiento es la limpieza de datos, que se refiere a la **identificación y corrección de errores o inconsistencias en los datos**. Esto puede incluir la eliminación de duplicados, el manejo de valores perdidos y la corrección de errores tipográficos.

Ejemplo: En un conjunto de datos de ventas, puede haber registros duplicados para una misma transacción. La limpieza de datos implica eliminar estos duplicados para asegurar que cada venta se cuente solo una vez. Además, si un registro tiene un valor perdido en el campo de la cantidad vendida, se puede optar por imputar un valor promedio o eliminar el registro, dependiendo de la cantidad de datos faltantes (Rahm & Do, 2000).

Figura N.º 2. Data Cleaning Process.
Fuente: https://airbyte.com/data-engineering-resources/data-cleansing

Reducción de datos

La reducción de datos **busca simplificar el conjunto de datos original mediante la eliminación de variables irrelevantes o redundantes**, lo que puede mejorar la eficiencia del análisis. Esta etapa es crucial cuando se trabaja con conjuntos de datos grandes que contienen muchas variables.

Ejemplo: En un análisis de satisfacción del cliente, se puede contar con cientos de preguntas en una encuesta. Algunas de estas preguntas pueden ser redundantes o tener correlaciones altas. Aplicando técnicas de reducción de dimensionalidad, como el Análisis de Componentes Principales (PCA), se pueden combinar varias variables relacionadas en un número menor de componentes que retengan la variabilidad más significativa del conjunto de datos original (Jolliffe & Cadima, 2016).

La siguiente figura muestra un esquema del proceso de reducción de datos:

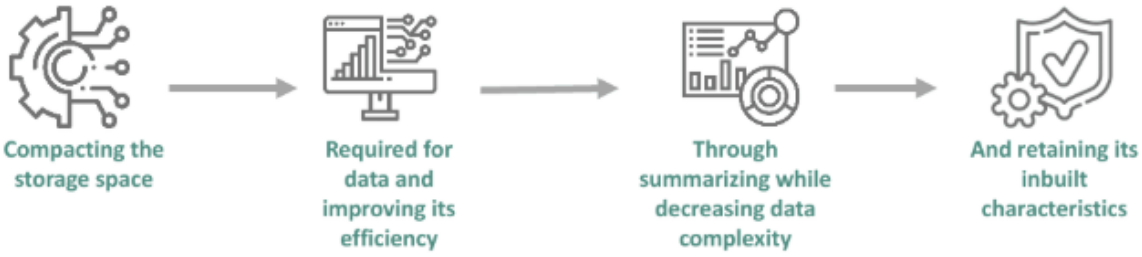


Figura N.º 3. *Data Reduction Process*.
Fuente: https://www.wallstreetmojo.com/data-reduction/

Transformación de datos

La transformación de datos es la etapa donde **los datos son convertidos a un formato adecuado para el análisis**. Esto puede incluir normalización, estandarización o codificación de variables categóricas.

Ejemplo: En un conjunto de datos que contiene información de clientes, donde se incluyen ingresos en diferentes escalas (algunos en miles y otros en millones), la normalización es esencial. Se pueden transformar los ingresos a una escala común; por ejemplo, utilizando una escala de 0 a 1. Esto ayuda a asegurar que todas las variables se consideren de manera equitativa en el análisis (Khan et al., 2019).

El siguiente diagrama muestra cómo se puede llevar a cabo un proceso de extracción, carga y transformación de datos con varios *inputs*.

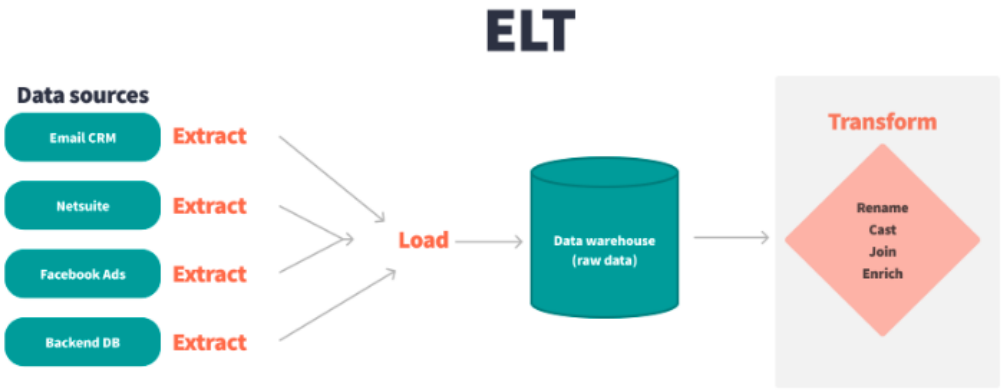


Figura N.º 4. *ELT Process*.
Fuente: https://www.getdbt.com/analytics-engineering/transformation

Enriquecimiento de datos

El enriquecimiento de datos implica **agregar información adicional que puede mejorar la calidad y el valor de los datos existentes**. Esta etapa es particularmente útil para crear un contexto más completo, que permita un análisis más profundo.

Ejemplo: Una empresa que recopila datos de ventas puede enriquecer sus datos de clientes incorporando información demográfica, como edad, nivel educativo y ubicación geográfica. Al agregar estos atributos, la empresa puede segmentar mejor a sus clientes y diseñar estrategias de marketing más efectivas, basadas en el comportamiento de compra (Bizer et al., 2009).

En el siguiente diagrama se aprecia un ejemplo de enriquecimiento de datos. Recopilando diferentes aspectos del cliente en relación con sus ingresos, egresos, estrategias de marketing, etc.

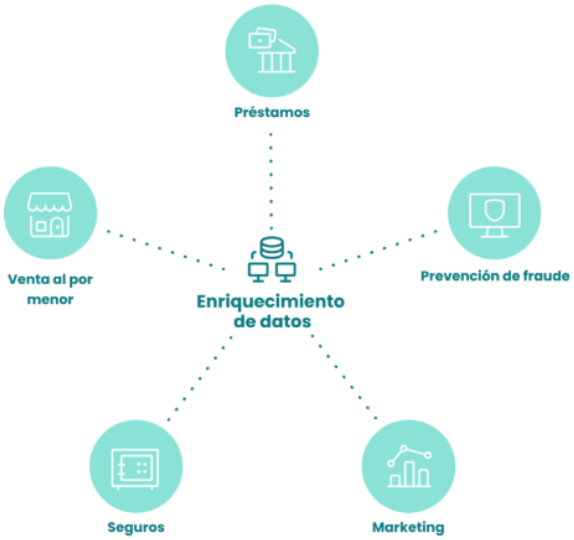


Figura N.º 5. Enriquecimiento de datos.
Fuente: https://seon.io/es/recursos/enriquecimiento-de-datos/

Validación de datos

La validación de datos es la etapa final del preprocesamiento y se refiere a la verificación de que los datos cumplen con criterios específicos de calidad y precisión. Esta etapa es esencial para garantizar que los datos sean confiables antes de ser utilizados en análisis posteriores.

Ejemplo: Después de haber limpiado, reducido, transformado y enriquecido un conjunto de datos, es crucial realizar una validación. Esto puede incluir la verificación de que todos los valores están dentro de rangos aceptables, que no existen duplicados y que los formatos de datos son consistentes (Pipino, Lee, & Wang, 2002). Por ejemplo, si un campo de fecha debe seguir un formato específico (como DD/MM/YYYY), se debe comprobar que todos los registros cumplan con este formato.

Importancia del preprocesamiento de datos

El preprocesamiento de datos es un paso crítico en el análisis de datos y la minería de datos, ya que se encarga de transformar los datos brutos en un formato adecuado para el análisis posterior. La calidad de los datos es fundamental, ya que datos erróneos o mal estructurados pueden llevar a conclusiones inexactas y decisiones equivocadas. A continuación, se presentan ejemplos que destacan la importancia del preprocesamiento de datos en diversas aplicaciones y contextos.

Mejora de la calidad de los datos

Otro de los beneficios del preprocesamiento es la mejora de la calidad de los datos. Esto se logra mediante la depuración o limpieza de datos, que implica identificar y corregir errores, como duplicaciones o valores perdidos.

Ejemplo: En un estudio de mercado, una empresa puede recopilar datos de encuestas para entender las preferencias de los consumidores. Si el conjunto de datos incluye múltiples registros para la

misma persona, debido a errores de entrada, esto puede llevar a una interpretación errónea de las preferencias. Al limpiar estos datos, la empresa puede obtener una visión más precisa y representativa del comportamiento del consumidor (Rahm & Do, 2000).

Aumento de la eficiencia analítica

El preprocesamiento también contribuye a la eficiencia analítica. Al reducir el volumen de datos a través de la eliminación de atributos irrelevantes o redundantes, se facilita el análisis.

Ejemplo: En un análisis de rendimiento académico en instituciones educativas, puede existir un conjunto de datos con cientos de variables sobre estudiantes, cursos y calificaciones. Si se eliminan las variables que no aportan información significativa, como comentarios de texto no estructurados, los analistas pueden concentrarse en las variables que realmente afectan al rendimiento. Esto no solo acelera el proceso de análisis, sino que también mejora la capacidad de interpretación de los resultados (Jolliffe & Cadima, 2016).

Facilita el aprendizaje automático

El preprocesamiento de datos es esencial en el ámbito del aprendizaje automático. Muchos algoritmos requieren que los datos estén en un formato específico y que estén normalizados o estandarizados.

Ejemplo: En un modelo predictivo para la detección de fraudes en transacciones financieras, es crucial que los datos de las transacciones sean normalizados. Si los datos de las transacciones incluyen montos que varían en rangos muy diferentes, esto puede afectar al rendimiento del modelo. La normalización asegura que todos los atributos tengan un impacto equitativo en el resultado final del modelo (Khan et al., 2019).

Mejora de la interpretabilidad

El preprocesamiento también mejora la interpretabilidad de los resultados. Al transformar los datos en un formato más comprensible, los analistas pueden comunicar sus hallazgos de manera más efectiva.

Ejemplo: En un análisis de satisfacción del cliente, convertir respuestas de encuestas en datos categóricos (como ‘satisfecho’, ‘neutral’ o ‘insatisfecho’) permite que los resultados sean más fáciles de entender para los tomadores de decisiones. Esto ayuda a las empresas a identificar áreas de mejora de manera clara y concisa (Feldman & Sanger, 2007).

Validación y confiabilidad

El preprocesamiento de datos asegura que los datos sean válidos y confiables, lo que es esencial para la toma de decisiones informadas.

Ejemplo: En estudios científicos, los investigadores deben asegurarse de que los datos recopilados sean precisos y válidos antes de realizar cualquier análisis. El preprocesamiento, que incluye la validación de datos, permite a los investigadores verificar que los datos cumplan con criterios

específicos de calidad. Esto aumenta la confianza en los resultados y las conclusiones del estudio (Pipino, Lee, & Wang, 2002).

Detección de *outliers*

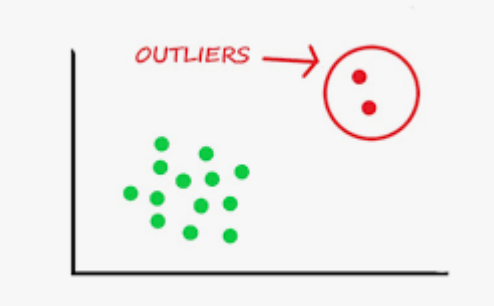


Figura N.º 6. Detección de *outliers*.
Fuente: https://medium.com/@yennhi95zz/the-importance-of-outlier-detection-in-machine-learning-methods-and-implementation-in-Python-125e3d5ada7d

La detección de *outliers*, o valores atípicos, es un componente crítico en el análisis de datos, ya que estos puntos pueden influir de manera significativa en la validez y la precisión de los resultados obtenidos a partir de un conjunto de datos. Un *outlier* se define como un dato que se encuentra considerablemente alejado de otros valores en un conjunto. Esta anomalía puede deberse a variaciones naturales en el fenómeno que se está estudiando, errores en la recolección de datos o condiciones extraordinarias. Identificar y tratar adecuadamente los *outliers* es fundamental para garantizar la integridad de los análisis posteriores.

Importancia de la detección de *outliers*

Los *outliers* pueden distorsionar las medidas estadísticas, como la media y la desviación estándar; esto puede llevar a conclusiones erróneas. Por ejemplo, en un análisis de ingresos, un único ingreso extremadamente alto puede elevar la media, dando la impresión de que el nivel de ingreso de la población es más alto de lo que realmente es. Por lo tanto, la detección y el manejo adecuado de los *outliers* son esenciales para proporcionar un análisis preciso y representativo (Iglewicz & Hoaglin, 1993).

1.1. Funciones en Python:

Python es uno de los lenguajes más utilizados en ciencia de datos debido a su versatilidad y a la gran cantidad de bibliotecas disponibles para el manejo y análisis de datos. Las funciones en Python son bloques **reutilizables** de código, que permiten realizar tareas específicas, facilitando la organización y modularidad del trabajo. En el contexto del manejo de datos, estas funciones se emplean para **limpiar, transformar y analizar datos**. Por ejemplo, funciones nativas como *len()*, *sum()* y *type()* proporcionan información básica sobre estructuras de datos, mientras que funciones

personalizadas permiten automatizar procesos complejos (Van Rossum & Drake, 2009). Adicionalmente, bibliotecas como **NumPy** y **Pandas** ofrecen funciones específicas para operaciones matemáticas y de manipulación de datos, lo que hace que Python sea indispensable para proyectos relacionados con datos (McKinney, 2017).

Python como lenguaje de programación
[https://www.redalyc.org/articulo.oa?id=181531232001]

1.2. Tipos de datos en Python

El manejo de datos en Python requiere una comprensión sólida de los tipos de datos que este lenguaje soporta, como **enteros, flotantes, cadenas y booleanos, entre otros**. Además, Python incluye estructuras de datos más complejas como **listas, tuplas, diccionarios y conjuntos,** que son fundamentales para organizar y procesar información. Cada tipo de dato tiene un propósito específico y capacidades únicas; por ejemplo, las listas son útiles para colecciones ordenadas de elementos, mientras que los diccionarios facilitan la asociación de claves y valores (Van Rossum & Drake, 2009). Comprender y elegir el tipo de dato adecuado para cada tarea es crucial para optimizar el manejo de datos y garantizar la eficiencia en el procesamiento. Este conocimiento se amplía al trabajar con bibliotecas de manejo de datos que introducen estructuras especializadas, como *arrays* en NumPy y *dataframes* en Pandas (McKinney, 2017).

1.3. Series y *dataframes*

Las series y *dataframes* son estructuras fundamentales de la biblioteca Pandas, diseñada para el análisis de datos en Python. Una serie es una estructura unidimensional similar a una columna en una hoja de cálculo o un *array*, que permite almacenar datos junto con etiquetas de índice. Por otro lado, un *dataframe* es una estructura bidimensional que organiza datos en filas y columnas, facilitando la manipulación, limpieza y análisis de grandes conjuntos de datos (McKinney, 2017). Estas estructuras proporcionan métodos eficientes para realizar operaciones como selección, filtrado, agrupación y agregación de datos, haciéndolas ideales para proyectos de ciencia de datos y aprendizaje automático. Su facilidad de integración con otras bibliotecas de Python las convierte en herramientas indispensables para el manejo de datos en cualquier etapa del análisis (Wes McKinney, 2017).

1.4. Missing Values

En ciencia de datos, los ***missing values*** o valores faltantes se refieren a aquellos datos que están ausentes o no disponibles en un conjunto de datos. Este es un problema común en las bases de datos reales, ya que los datos pueden faltar por diversas razones, como errores en la recolección de datos, problemas en la transmisión de datos o simplemente porque no se ha registrado información para ciertas variables o instancias.

El manejo adecuado de los *missing values* es crucial para el análisis y la modelización, ya que muchos algoritmos no pueden procesar datos incompletos o pueden dar resultados erróneos si no se tratan adecuadamente. Existen varias formas de manejar los *missing values*, como:

- **Eliminar los registros con valores faltantes** (aunque esto puede reducir el tamaño del conjunto de datos).
- **Imputar los valores faltantes** (es decir, reemplazar los valores faltantes con algún valor estimado, como la media, mediana o moda).
- **Usar algoritmos que puedan manejar los *missing values* de forma interna**, como algunos modelos de árboles de decisión.

Ejemplo en Python

Supongamos que tenemos un conjunto de datos de ejemplo en formato de un *dataframe* de Pandas que contiene algunos valores faltantes. A continuación, te muestro cómo manejar estos valores faltantes utilizando el enfoque de imputación:

```python
import pandas as pd
import numpy as np

# Crear un DataFrame de ejemplo con valores faltantes
data = {
    'Edad': [25, 30, np.nan, 35, 40],
    'Salario': [50000, np.nan, 60000, 55000, 70000],
    'Ciudad': ['Madrid', 'Barcelona', 'Madrid', np.nan, 'Sevilla']
}

df = pd.DataFrame(data)

# Mostrar el DataFrame original
print("DataFrame original:")
print(df)

# Imputación de valores faltantes:
# 1. Imputar la edad con la media de la columna
df['Edad'] = df['Edad'].fillna(df['Edad'].mean())

# 2. Imputar el salario con la mediana de la columna
df['Salario'] = df['Salario'].fillna(df['Salario'].median())

# 3. Imputar la ciudad con la moda de la columna (valor más frecuente)
df['Ciudad'] = df['Ciudad'].fillna(df['Ciudad'].mode()[0])

# Mostrar el DataFrame después de imputar los valores faltantes
print("\nDataFrame después de imputar los valores faltantes:")
print(df)
```

```
DataFrame original:
   Edad  Salario     Ciudad
0  25.0  50000.0     Madrid
1  30.0      NaN  Barcelona
2   NaN  60000.0     Madrid
3  35.0  55000.0        NaN
4  40.0  70000.0    Sevilla

DataFrame después de imputar los valores faltantes:
   Edad  Salario     Ciudad
0  25.0  50000.0     Madrid
1  30.0  57500.0  Barcelona
2  32.5  60000.0     Madrid
3  35.0  55000.0     Madrid
4  40.0  70000.0    Sevilla
```

Figura N.º 7. *Missing Values*

1.5. *Merging dataframes*

En el manejo de datos, la **combinación de *dataframes* es una operación esencial que permite unir diferentes conjuntos de datos para analizarlos en conjunto**. En Python, la biblioteca Pandas proporciona métodos como **merge()** y **concat()** para realizar estas operaciones. La función **merge()** permite combinar *dataframes* en función de una o más claves comunes, similar a las uniones en SQL, mientras que **concat()** se utiliza para concatenar *dataframes* a lo largo de un eje (filas o columnas). Estas herramientas son fundamentales para consolidar datos provenientes de diversas fuentes en un solo conjunto, asegurando así una integración fluida y coherente (McKinney, 2017). El uso correcto de estas técnicas evita redundancias y facilita un análisis más profundo de los datos combinados (Han et al., 2022).

Ejemplo: La combinación de *dataframes* es una técnica utilizada para unir diferentes conjuntos de datos basándose en claves comunes. En Pandas, el método merge() permite realizar combinaciones similares a las uniones en SQL, como *inner join*, *left join*, entre otras. Por ejemplo, si se tienen dos conjuntos de datos: uno con información de clientes (clientes) y otro con sus compras (compras), pueden combinarse usando la clave común id_cliente:

```python
# Importamos la pandas
import pandas as pd

# Definimos dos dataframes
clientes = pd.DataFrame({'id_cliente': [1, 2], 'nombre': ['Juan', 'Ana']})
compras = pd.DataFrame({'id_cliente': [1, 2, 1], 'compra': ['Libro', 'Teléfono', 'Lápiz']})

# Combinamos ambos dataframes
df_combinado = pd.merge(clientes, compras, on='id_cliente', how='inner')
print(df_combinado)
```

Figura N.º 8. *Merging dataframes*

1.6. Agrupación de datos (*GroupBy)*

La operación de agrupación de datos, implementada mediante el método **groupby()** en Pandas, es una técnica crucial para realizar análisis agregados. Esta funcionalidad permite dividir un conjunto de datos en grupos basados en una o más variables, aplicar funciones de agregación como suma, promedio o conteo, y combinar los resultados en una nueva estructura. Por ejemplo, agrupar datos por categorías como regiones o períodos de tiempo facilita el análisis comparativo y la identificación de tendencias (McKinney, 2017). La agrupación no solo optimiza el procesamiento, sino que también proporciona una base sólida para el análisis estadístico y predictivo en proyectos de aprendizaje automático (Shmueli et al., 2021).

Ejemplo: La agrupación de datos permite analizar subconjuntos basados en una o más características. En Pandas, groupby() divide un *dataframe* en grupos y aplica funciones de agregación como mean() o sum(). Por ejemplo, para calcular el total de compras por cliente:

```
df_combinado.groupby('nombre')['compra'].count()
```

Figura N.º 9. *Group By*

Este enfoque es ideal para identificar patrones o tendencias dentro de subgrupos específicos de datos, permitiendo un análisis detallado y enfocado (Shmueli et al., 2021).

Análisis del uso de Python como lenguaje de programación para cálculos estadísticos. [https://www.redalyc.org/journal/5732/573270857001/573270857001.pdf]

1.7. Escalado de datos (*scaling*)

El escalado de datos es una etapa clave en el preprocesamiento, especialmente en algoritmos de aprendizaje automático sensibles a la magnitud de las variables, como las máquinas de soporte vectorial (SVM) o el análisis de componentes principales (PCA). Este proceso ajusta las características de un conjunto de datos a una escala común, utilizando técnicas como la normalización (ajustar los valores entre 0 y 1) o la estandarización (centrar los datos en la media y escalar por la desviación estándar). Python proporciona herramientas como *StandardScaler* y *MinMaxScaler* de la biblioteca *Scikit-learn* para realizar estas transformaciones de manera eficiente (Pedregosa et al., 2011). El escalado adecuado garantiza un mejor rendimiento y estabilidad de los modelos predictivos (Han et al., 2022).

Ejemplo: El escalado ajusta las características a una escala uniforme, mejorando el rendimiento de los modelos de aprendizaje automático. Con *Scikit-learn* se puede usar *StandardScaler* para centrar los datos en la media y escalar por la desviación estándar:

```python
# Importamos librerías
from sklearn.preprocessing import StandardScaler
import numpy as np

# Generamos datos de prueba
datos = np.array([[1, 2], [3, 4], [5, 6]])

# Instanciamos la clase StandardScaler
scaler = StandardScaler()

# Aplicamos el método fit_transform para scalar los datos
datos_escalados = scaler.fit_transform(datos)

#Imprimimos los datos por pantalla
print(datos_escalados)
```

Figura N.º 10. *Scaling*

Esto es especialmente útil para algoritmos como SVM o PCA, donde la magnitud de las variables puede afectar significativamente al resultado (Pedregosa et al., 2011).

1.8. *Pivot Table*

En ciencia de datos, una ***pivot table*** (o tabla dinámica) es una herramienta que permite **reorganizar** y **resumir** un conjunto de datos, para hacer análisis más eficientes y obtener *insights*. Se utiliza comúnmente para agrupar y agregar datos de manera que sea más fácil interpretar patrones y relaciones en grandes volúmenes de información.

En términos sencillos, las ***pivot tables*** permiten:

- **Agrupar datos** en categorías.
- **Aplicar funciones de agregación** como sumas, promedios, recuentos, etc., a esos grupos.
- **Reorganizar la forma del *dataframe*** para analizar los datos desde diferentes perspectivas.

La tabla dinámica tiene típicamente dos ejes:

1. **Filas**: donde se agrupan los datos.
2. **Columnas**: donde se agrupan las categorías o variables.
3. **Valores**: que contienen los datos que se quieren resumir (por ejemplo, sumas, promedios, conteos).

Este tipo de herramienta es muy útil para hacer análisis de datos exploratorios (EDA), donde se busca encontrar patrones y relaciones entre diferentes variables de manera rápida.

Ejemplo en Python

```python
import pandas as pd

# Crear un DataFrame de ejemplo con datos de ventas
data = {
    'Fecha': ['2025-01-01', '2025-01-01', '2025-01-02', '2025-01-02', '2025-01-03'],
    'Tienda': ['Tienda A', 'Tienda B', 'Tienda A', 'Tienda B', 'Tienda A'],
    'Producto': ['Producto 1', 'Producto 1', 'Producto 2', 'Producto 2', 'Producto 1'],
    'Ventas': [100, 150, 200, 250, 120],
    'Precio': [10, 10, 15, 15, 10]
}

df = pd.DataFrame(data)

# Mostrar el DataFrame original
print("DataFrame original:")
print(df)

# Crear una tabla dinámica para sumar las ventas por tienda y producto
pivot = pd.pivot_table(df, values='Ventas', index='Tienda', columns='Producto', aggfunc='sum', fill_value=0)

# Mostrar la tabla dinámica
print("\nTabla dinámica de ventas por tienda y producto:")
print(pivot)
```

Figura N.º 11. *Pivot table.*

Pivot tables [https://research.moodle.org/56/1/15%20-%20Dierenfeld%20-%20Learning%20Analytics%20with%20Excel%20Pivot%20Tables.pdf]

1.9. Funcionalidad *date/time*

El manejo de datos temporales es esencial en muchos proyectos de análisis y aprendizaje automático, como series temporales o modelado predictivo. Python, a través de Pandas, ofrece potentes herramientas para trabajar con fechas y tiempos, utilizando objetos como **datetime** y **Timedelta**. Funciones como **to_datetime()** permiten convertir cadenas de texto a formatos de fecha, mientras que métodos como **resample()** y **dt** facilitan la agrupación, agregación y transformación de datos en función de intervalos de tiempo específicos (McKinney, 2017). Estas capacidades son fundamentales para identificar patrones temporales, realizar análisis históricos y construir modelos predictivos basados en tendencias temporales (Shmueli et al., 2021).

Ejemplo: El manejo de datos temporales es crucial en proyectos como series de tiempo. En Pandas se puede usar **to_datetime()** para convertir cadenas a fechas, y realizar operaciones como reprogramación con **resample()**:

```python
import pandas as pd

fechas = pd.date_range(start='2024-01-01', periods=5, freq='D')
datos = pd.DataFrame({'fecha': fechas, 'valor': [1, 2, 3, 4, 5]})

datos['fecha'] = pd.to_datetime(datos['fecha'])
datos_resampleado = datos.resample('2D', on='fecha').sum()
print(datos_resampleado)
```

Figura N.º 12. Funcionalidad Date/Time

Estas herramientas son esenciales para identificar patrones temporales o realizar análisis basados en intervalos específicos (McKinney, 2017).

Referencias citadas en la Clase 1

- Aggarwal, C. C. (2015). *Data mining: The textbook*. Springer.
- Han, J., Kamber, M., & Pei, J. (2022). *Data mining: Concepts and techniques* (4th ed.). Morgan Kaufmann.
- James, G., Witten, D., Hastie, T., & Tibshirani, R. (2017). *An introduction to statistical learning with applications in R* (2nd ed.). Springer.
- Shmueli, G., Bruce, P. C., & Patel, N. R. (2021). *Data mining for business analytics: Concepts, techniques, and applications in R*. Wiley.
- McKinney, W. (2017). *Python for data analysis: Data wrangling with Pandas, NumPy, and Jupyter* (2nd ed.). O'Reilly Media.
- Van Rossum, G., & Drake, F. L. (2009). *Python 3 Reference Manual*. CreateSpace.
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research, 12*, 2825–2830.

Definición de los términos citados en la Clase 1

Serie.
Una s**erie** en Python, específicamente en la biblioteca **Pandas**, es una estructura de datos unidimensional similar a un *array* o lista, pero con la ventaja de que cada elemento tiene asociado un índice (que puede ser numérico o etiquetado).

Dataframe.
Un ***dataframe*** en Python, proporcionado por la biblioteca **Pandas**, es una estructura de datos bidimensional similar a una tabla (como una hoja de cálculo o una tabla SQL). Contiene filas y columnas, donde cada columna puede tener un tipo de dato diferente.

PUCE Virtual
EDUCACIÓN FLEXIBLE

La excelencia no se improvisa

síguenos

MDASC - 12 - 241