Punto 6: Limpieza de datos.

¿En qué consiste la limpieza de datos en data Science y por qué es tan crucial esta etapa?

La limpieza de datos es un proceso fundamental en *Data Science* que implica preparar y transformar los datos brutos en un formato adecuado para el análisis y modelado. Es esencial para aplicaciones como *Machine Learning* y *Power BI*, ya que los datos sin procesar pueden contener numerosos errores.

La limpieza de datos es esencial porque garantiza que los análisis y modelos construidos a partir de los datos sean fiables y precisos. Datos sucios o mal preparados pueden llevar a:

Afectar la precisión de los Modelos y dar lugar a predicciones incorrectas, llevando a decisiones equivocadas y a un impacto empresarial negativo.

La presencia de datos inconsistentes o duplicados puede hacer que los algoritmos de aprendizaje automático sean menos eficientes y más lentos.

Datos mal preparados pueden ser difíciles de interpretar y comunicar, dificultando la toma de decisiones basada en datos.

Los datos en el mundo real rara vez son homogéneos y directamente intuitivos. Asimismo, realizar este proceso puede reducir posibles sesgos en el análisis. He aquí varias de las razones más comunes:

Eliminación de Errores y Valores Atípicos:

Los datos crudos suelen contener errores, inconsistencias y valores atípicos (*outliers*) que pueden distorsionar el análisis y llevar a conclusiones incorrectas. La limpieza ayuda a identificar y corregir estos problemas para asegurar la integridad de los datos. Ejemplo de esto podemos citar, encontrar la variable edad de una persona en 580. Inmediatamente nos indica que es un error de entrada de datos, conocido en inglés como *“data entry error”*, probablemente la edad sea 58.

Trabajar datos Nulos:

No se ha introducido información alguna en alguno de los elementos de la variable. Estos pueden deberse a falta de información o bien a no haber podido estimar la variable, un error de entrada, o bien hubo un fenómeno externo que impidió la recolección de datos. La limpieza de datos implica decidir cómo tratar estos valores, ya sea mediante su eliminación, imputación o sustitución.

Normalización y Estandarización:

Los datos provenientes de diversas fuentes pueden tener formatos diferentes y escalas variadas. La limpieza de datos incluye la normalización y estandarización de estos datos para garantizar que sean comparables y coherentes.

Eliminación de Duplicados:

Los datos duplicados pueden inflar artificialmente ciertos resultados y análisis. La limpieza de datos implica identificar y eliminar duplicados para mantener la precisión del conjunto de datos.

Aseguramiento de la Calidad de los Datos:

La calidad de los datos es esencial para construir modelos precisos y confiables. La limpieza de datos garantiza que los datos sean precisos, completos, consistentes y relevantes para el análisis. por ejemplo, tener la variable *precio de venta* en **números negativos**.

Existen diversas técnicas para realizar esta etapa, una de las mas populares y empleada en este proceso es el uso de la librería pandas de Python, de las cuales existen algunas sentencias que son claves, por ejemplo:

1. **df.dropna(inplace=True)**

Propósito: Eliminar filas (o columnas) que contienen valores nulos **(NaN)** en el **DataFrame df**

Funcionamiento:

dropna(): Por defecto, elimina cualquier fila que tenga al menos un valor nulo.

Inplace = True: Esta opción modifica el **DataFrame** original '**df**' directamente en lugar de crear y retornar una copia con las filas eliminadas.

Ejemplo:

```
import pandas as pd

data = {
  'A': [1, 2, None],
  'B': [4, None, 6]
}
df = pd.DataFrame(data)
df.dropna(inplace=True)

print(df)
```

Resultado:

```
   A  B
0  1  4
```

La fila 1 y 2 se eliminan porque contienen valores 'None' (nulos).

2. **df.drop_duplicates(inplace=True)**

Propósito: Eliminar filas duplicadas en el **DataFrame df**.

Funcionamiento:

drop_duplicates(): Identifica y elimina filas duplicadas en el **DataFrame**.

inplace=True: Modifica el **DataFrame** original directamente.

Ejemplo:

```
import pandas as pd

data = {
   'A': [1, 2, 2],
   'B': [4, 5, 5]
}
df = pd.DataFrame(data)
df.drop_duplicates(inplace=True)

print(df)
```

Resultado:

```
   A  B
0  1  4
1  2  5
```

La segunda aparición de la fila con valores (2, 5) se elimina.

3. **df["variable_numerica"] = df["variable_numerica"].astype("float")**

Propósito: Convertir la columna 'variable_numerica' del **DataFrame df** a tipo de dato 'float'.

Funcionamiento:

astype("float"): Cambia el tipo de los datos en la columna especificada a 'float'.

df["variable_numerica"]: Selecciona la columna específica que se desea convertir.

Ejemplo:

```
import pandas as pd
data = {
   'variable_numerica': ['1.1', '2.2', '3.3']
}
df = pd.DataFrame(data)
df["variable_numerica"] = df["variable_numerica"].astype("float")
print(df.dtypes)
print(df)
```

Resultado:

```
variable_numerica    float64
dtype: object
  variable_numerica
0                 1.1
1                 2.2
2                 3.3
```

La columna 'variable_numerica' se convierte de 'object' (cadenas de texto) a 'float64'.

En resumen, estas instrucciones se utilizan para limpiar y preparar los datos, eliminando filas con valores nulos, eliminando filas duplicadas, y asegurando que las columnas tienen el tipo de dato correcto para análisis posteriores.

Podemos concluir que la limpieza de datos es una etapa crítica en cualquier proyecto de *Data Science*, ya que asegura que los datos utilizados son adecuados y de alta calidad, lo que a su vez mejora la precisión y eficacia del análisis y modelado subsiguiente

Punto 7: Análisis exploratorio de datos.

¿Cuál es la importancia de realizar un buen análisis exploratorio de datos en Data Science?

Los científicos de datos utilizan el análisis de datos exploratorios para analizar e investigar conjuntos de datos y resumir sus características principales, a menudo empleando métodos de visualización de datos.

Esta etapa es de muy importante en *Data Science* por varias razones:

1. Puede ayudar a localizar errores obvios.
2. Comprender mejor los patrones dentro de los datos, detectar valores atípicos o eventos anómalos y encontrar relaciones interesantes entre las variables.
3. Ayuda a los científicos de datos a formular hipótesis preliminares sobre los datos. Estas hipótesis pueden luego ser probadas mediante técnicas estadísticas o modelos de *machine learning*. Sin una exploración adecuada, es difícil plantear preguntas de investigación relevantes.
4. Permite la identificación de las variables más relevantes y la comprensión de su influencia en el resultado de interés. Esta selección de características es vital para construir modelos predictivos eficientes y efectivos.
5. Utiliza visualizaciones como histogramas, diagramas de dispersión, diagramas de caja y mapas de calor para ilustrar las distribuciones de las variables y las relaciones entre ellas. Estas visualizaciones no solo facilitan la comprensión de los datos, sino que también ayudan a comunicar hallazgos a otros *stakeholders*.
6. Para ciertos modelos estadísticos y de machine learning, es importante que se cumplan ciertos supuestos (e.g., normalidad, homocedasticidad). EDA permite validar estos supuestos antes de proceder con el modelado.
7. EDA puede revelar si hay demasiadas variables irrelevantes o redundantes que pueden ser eliminadas o combinadas. Esto simplifica los modelos y mejora su interpretabilidad y rendimiento.
8. Un análisis exploratorio exhaustivo asegura que los datos están en la mejor forma posible para el modelado. Esto incluye transformar variables, manejar datos faltantes, y normalizar o estandarizar datos según sea necesario.

Para el análisis exploratorio de datos se emplean diversas técnicas descriptivas, sobre todo, visuales como:

Mapas de calor: visualización de datos que utiliza colores para comparar y contrastar números en un conjunto de datos; también se conoce como matrices de sombreado.

Histogramas: Un histograma es un gráfico de barras que agrupa números en una serie de intervalos, especialmente cuando hay una variable infinita, como los pesos y las medidas.

Gráfica de líneas: Uno de los tipos más básicos de gráficos que traza puntos de datos en un gráfico; tiene una gran cantidad de usos en casi todos los campos de estudio.

Pictogramas: sustituyen los números por imágenes para explicar visualmente los datos. Son habituales en el diseño de infografías, así como en los elementos visuales que los científicos de datos pueden utilizar para explicar hallazgos complejos a los profesionales que no son científicos de datos y al público.

Diagramas de dispersión o *scatterplots*: Suelen utilizarse para mostrar dos variables en un conjunto de datos y luego buscar correlaciones entre ellos.

Las siguientes sentencias se utilizan para obtener una comprensión inicial de los datos mediante la visualización y el resumen estadísticas clave:

En el análisis exploratorio de datos (EDA), las siguientes sentencias se utilizan para obtener una comprensión inicial de los datos mediante la visualización y el resumen de estadísticas clave.

1. **print(df.describe())**

Descripción:

La función **df.describe()** proporciona un resumen estadístico de las columnas numéricas del *DataFrame* **df**.

Uso:

Esta función se utiliza para obtener una visión rápida de los estadísticos básicos de las variables numéricas tales como:

count: Número de valores no nulos.

mean: Media aritmética.

std: Desviación estándar.

min: Valor mínimo.

25%, 50% (mediana) y **75%**: Cuartiles.

max: Valor máximo.

Ejemplo:

```
import pandas as pd
# Suponiendo que ya tienes un DataFrame llamado df
print(df.describe())
```

Resultado:

Este comando imprimirá un resumen estadístico de todas las columnas numéricas del *DataFrame*, facilitando la identificación de distribuciones y posibles valores atípicos.

2. df.hist()

Descripción:

La función **df.hist**() genera histogramas para todas las columnas numéricas del *DataFrame* **df**.

Uso:

Los histogramas son útiles para visualizar la distribución de las variables numéricas. Ayudan a identificar la forma de la distribución (e.g., normal, sesgada, bimodal) y detectar valores atípicos.

Ejemplo:

```
import matplotlib.pyplot as plt
# Genera histogramas para todas las columnas numéricas
df.hist()

# Muestra las gráficas generadas
```

plt.show()

Resultado:

Este comando creará una serie de subgráficas, cada una mostrando un histograma de una columna numérica del *DataFrame*. La función **plt.show()** es necesaria para mostrar las gráficas cuando se trabaja en algunos entornos como scripts o *Jupyter Notebooks*.

3. df.corr()

Descripción:

La función **df.corr()** calcula el coeficiente de correlación entre las columnas numéricas del *DataFrame* **df**.

Uso:

El coeficiente de correlación mide la relación lineal entre variables. Un valor cercano a 1 indica una fuerte correlación positiva, -1 indica una fuerte correlación negativa, y 0 indica que no hay correlación lineal.

Se utiliza para identificar relaciones entre variables que pueden ser importantes para el análisis y modelado predictivo.

Ejemplo:

```
# Calcula la matriz de correlación
correlation_matrix = df.corr()
print(correlation_matrix)
```

Resultado:

Este comando imprimirá una matriz de correlación que muestra los coeficientes de correlación entre cada par de columnas numéricas en el *DataFrame*.

Resumiendo, estas sentencias son herramientas fundamentales en el EDA:

df.describe() proporciona un resumen estadístico básico que ayuda a entender la distribución de los datos numéricos.

df.hist() permite visualizar las distribuciones de las variables y detectar patrones y anomalías.

df.corr() ayuda a identificar relaciones lineales entre variables, informando sobre la posible colinealidad o independencia de los datos.

Estas funciones proporcionan una base sólida para realizar análisis más profundos y prepararse para el modelado predictivo en proyectos de *Data Science*.