
ANALÍTICA DE
DATOS
2020
1
6
7
Lorem ipsum dolor sit amet,
consectetuer adipiscing elit
10%

Una de las herramientas más populares y versátiles es Python, un lenguaje de programación ampliamente utilizado en el campo de la analítica de datos. Python cuenta con diversas librerías especializadas en el manejo de datos, como pandas y NumPy. Pandas ofrece funcionalidades avanzadas para la limpieza y manipulación de datos, permitiendo realizar tareas como filtrado, ordenamiento, agrupamiento y cálculos estadísticos de manera sencilla y eficiente. Por su parte, NumPy proporciona una infraestructura eficiente para trabajar con matrices y arreglos de datos, lo que resulta especialmente útil para el procesamiento de datos numéricos.

```
import pandas as pd
# Cargar el archivo de datos
data = pd.read_csv('ruta_del_archivo.csv')
# Visualizar los primeros registros del conjunto de datos
print("Antes de la limpieza:")
print(data.head())
# Eliminar registros duplicados
data = data.drop_duplicates()
# Eliminar valores faltantes
data = data.dropna()
# Realizar otras transformaciones y limpiezas según sea necesario
# ...
# Guardar el conjunto de datos limpio en un nuevo archivo
data.to_csv('ruta_del_archivo_limpo.csv', index=False)
# Visualizar los primeros registros del conjunto de datos limpio
print("Después de la limpieza:")
print(data.head())
```

En este ejemplo, el script carga un archivo CSV utilizando la función read_csv de pandas. Luego, se utilizan los métodos drop_duplicates y dropna para eliminar registros duplicados y valores faltantes, respectivamente. Puedes realizar otras transformaciones y limpiezas adicionales según tus necesidades, como cambiar formatos de fechas, corregir errores, etc. Finalmente, el conjunto de datos limpio se guarda en un nuevo archivo CSV utilizando el método to_csv.

Recuerda reemplazar 'ruta_del_archivo.csv' con la ubicación y nombre del archivo de datos que deseas limpiar. Asimismo, ajusta 'ruta_del_archivo_limpo.csv' al nombre y ubicación del archivo de salida limpio que deseas generar.

Este script es solo un ejemplo básico y puede ser adaptado y ampliado según los requisitos específicos de tu conjunto de datos.

```
# Cargar el paquete tidyverse
library(tidyverse)
# Leer el archivo CSV con los datos
datos <- read_csv("ruta_del_archivo.csv")
# Visualizar las primeras filas del conjunto de datos
head(datos)
# Verificar la estructura de los datos
```

```
str(datos)
# Limpiar los datos
# 1. Eliminar filas duplicadas
datos <- distinct(datos)
# 2. Tratar valores faltantes
# Opción 1: Eliminar filas con valores faltantes
datos <- drop_na(datos)
# Opción 2: Rellenar valores faltantes con la media
de la columna
datos <- datos %>%
         mutate(columna_con_valores_faltantes     =
ifelse(is.na(columna_con_valores_faltantes),

 mean(columna_con_valores_faltantes, na.rm = TRUE),

 columna_con_valores_faltantes))
# 3. Renombrar columnas (si es necesario)
#   datos   <-   rename(datos,   nueva_columna   =
antigua_columna)
# 4. Convertir tipos de datos (si es necesario)
#   datos   <-   mutate(datos,   nueva_columna   =
as.numeric(nueva_columna))
```

```
# 5. Filtrar registros según criterios específicos
# datos <- filter(datos, columna > valor)
# Guardar el conjunto de datos limpio en un nuevo archivo CSV
write_csv(datos, "ruta_del_archivo_limpio.csv")
# Resumen estadístico de los datos limpios
summary(datos)
```

Recuerda reemplazar "ruta_del_archivo.csv" y "ruta_del_archivo_limpio.csv" con las rutas adecuadas de tus archivos de datos de entrada y salida, respectivamente. Además, puedes ajustar las operaciones de limpieza según las necesidades específicas de tu conjunto de datos, como renombrar columnas, convertir tipos de datos o aplicar filtros adicionales.

Este script proporciona una base para la limpieza de datos en R, y puedes expandirlo o modificarlo según tus requisitos y preferencias.

## Recolección de datos:

Script en Python que realiza los procesos de normalización, estandarización y agregación de un conjunto de datos. Recuerda que este es solo un ejemplo básico y puedes personalizarlo según tus necesidades específicas.

```
import pandas as pd
from sklearn.preprocessing import MinMaxScaler, StandardScaler
# Cargar el conjunto de datos
data = pd.read_csv("ruta_del_archivo.csv") # Reemplazar "ruta_del_archivo.csv" por la ubicación de tu archivo de datos
# Normalización
scaler = MinMaxScaler() # Inicializar el objeto MinMaxScaler
data_normalized = scaler.fit_transform(data) # Aplicar la normalización a los datos
# Estandarización
scaler = StandardScaler() # Inicializar el objeto StandardScaler
data_standardized = scaler.fit_transform(data) # Aplicar la estandarización a los datos
# Agregación
aggregated_data = data.groupby("columna_agrupadora").agg({"columna1": "sum", "columna2": "mean", "columna3": "max"})
# Reemplazar "columna_agrupadora", "columna1", "columna2" y "columna3" por los nombres de las columnas relevantes en tu conjunto de datos
```

```
# Mostrar los resultados
print("Datos normalizados:")
print(data_normalized)
print("Datos estandarizados:")
print(data_standardized)
print("Datos agregados:")
print(aggregated_data)
```

Asegúrate de tener instaladas las bibliotecas necesarias antes de ejecutar el script. Puedes instalarlas utilizando el siguiente comando en tu entorno de Python:

```
pip install pandas scikit-learn
```

Recuerda reemplazar "ruta_del_archivo.csv" por la ruta real de tu archivo de datos, y ajustar las columnas utilizadas en la normalización, estandarización y agregación según tu conjunto de datos. Además, ten en cuenta que este es solo un ejemplo básico y es posible que necesites adaptarlo o ampliarlo según tus requerimientos específicos.

Script en R que muestra cómo realizar los procesos de normalización, estandarización y agregación en un conjunto de datos. Asegúrate de tener instalado R y las bibliotecas necesarias antes de ejecutar el script.

```
# Cargar el conjunto de datos
datos <- read.csv("ruta_del_archivo.csv")
# Proceso de normalización
datos$columna_normalizada <- scale(datos$columna)
# Proceso de estandarización
datos$columna_estandarizada <- (datos$columna -
mean(datos$columna)) / sd(datos$columna)
# Proceso de agregación
datos_agregados <- aggregate(datos$columna, by =
list(datos$grupo), FUN = sum)
# Guardar el conjunto de datos transformado
write.csv(datos,
"ruta_del_archivo_transformado.csv", row.names =
FALSE)
```

Asegúrate de reemplazar "ruta_del_archivo.csv" con la ubicación y el nombre de tu archivo de datos original. El resultado transformado se guardará en un nuevo archivo llamado "ruta_del_archivo_transformado.csv". También asegúrate de reemplazar "columna" con el nombre de la columna en la que deseas aplicar los procesos de normalización y estandarización. Además, ajusta "grupo" con el nombre de la columna que representa los grupos para el proceso de agregación.

En el script, la función scale se utiliza para realizar la normalización de la columna seleccionada. La normalización escala los valores de la columna para que tengan una media de cero y una desviación estándar de uno.

La estandarización se realiza restando la media de la columna y dividiendo por la desviación estándar. Esto garantiza que los valores resultantes tengan una media de cero y una desviación estándar de uno.

El proceso de agregación utiliza la función aggregate para sumar los valores de la columna seleccionada por cada grupo. En este caso, se asume que la columna "grupo" contiene los nombres de los grupos.

Recuerda personalizar el script según tus necesidades y las características específicas de tu conjunto de datos.