MODELO PREDICCIÓN STRYD MEDIANTE MACHINE LEARNING

Fases de un proyecto de Modelado

El objeto de este proyecto es la realización de un modelo de Machine Learning para la predicción de una/s métrica/s proporcionadas por un dispositivo que mide la potencia que generamos cuando realizamos la actividad física de correr. Como podemos observar esta contextualización es bastante vaga e indefinida, pero como todo proyecto debemos empezar por algún punto.

David Víctor Gómez Ramírez

Técnico Superior en Desarrollo de Aplicaciones Multiplataforma especialidad en BIGDATA

Marzo 2022

Visión General Proyecto Métricas Stryd

INTRODUCCIÓN

El objeto de este proyecto es la realización de un modelo de Machine Learning para la predicción de una/s métrica/s proporcionadas por un dispositivo que mide la potencia que generamos cuando realizamos la actividad física de correr. Como podemos observar esta contextualización es bastante vaga e indefinida, pero como todo proyecto debemos empezar por algún punto.

STRYD

¿Qué es STRYD?

Stryd es un medidor de la potencia que generamos cuando practicamos “running” y, aunque en concepto es muy similar a los medidores de potencia del “ciclismo”, tienen diferencias sustanciales ya que Stryd no mide la potencia de forma directa sino que utiliza un algoritmo para su cálculo.

Para este proyecto personal, Stryd no es más que una herramienta más, por lo que no ahondaremos en su funcionamiento pero sí en las métricas y datos que nos aporta y que serán la base de nuestro modelo predictivo, es decir, será la fuente principal de la que obtendremos los datos de nuestro modelo.

Por simplificar y ubicarnos mejor en el contexto diremos que Stryd es un dispositivo que mide diferentes valores cuando realizamos entrenamientos de “running” y cuyo eje central es la potencia medida en watios (W), pero tenemos muchas más y que serán muy importantes (o no) para nuestro modelo:

- ***Tiempo (DURACION):*** variable fundamental de cualquier entrenamiento y que mide la duración del mismo.
- ***Distancia (DISTANCIA):*** mide la longitud de los entrenamientos.
- ***Ritmo (RITMO):*** es la métrica por excelencia que relaciona “Tiempo” y “Distancia”.

- ***Altimetría y Desnivel (ALTITUD – DESNIVEL):*** mediante mide la altitud y sus variaciones.

- ***Cadencia (CADENCIA):*** son las zancadas o pasos que damos mientras corremos y se mide en zancadas o pasos por minuto.
- ***TSC (Tiempo de contacto con el suelo):*** podríamos definirlo como la duración medida en ms en el que nuestro pie está en contacto con el suelo.

- ***Oscilación vertical (OSC_VERTICAL):*** es la distancia vertical que recorre el cuerpo desde el punto en que el centro de gravedad está más bajo hasta el punto más alto.
- ***Leg Spring Stiffness (LSS):*** se trataría del acrónimo inglés LSS (Leg Spring Stifness) que adaptado al castellano vendría a ser cómo la rigidez del resorte.

- ***Pendiente (PENDIENTE):*** mide el porcentaje de pendiente media durante la distancia recorrida.
- ***RSS (RSS):*** mide el estrés que ha generado el entrenamiento o segmento.

- ***Potencia (POTENCIA):*** mide la potencia media generada durante un entrenamiento o segmento.
- ***Frecuencia Cardiaca (FREC_CARDIACA):*** mide la frecuencia cardiaca por entrenamiento o segmento medidas en pulsaciones por minuto (ppm)
- ***Aire (AIRE):*** mide la resistencia de aire que encontramos en un entrenamiento o segmento.
- ***Form Power (FP):*** mide la potencia vertical y lateral que generamos cuando nos desplazamos en un entrenamiento o segmento.

- ***Segmento:*** es un concepto que deberemos tener muy presente y que no es más que pequeñas partes del entrenamiento de los que podemos extraer la misma información.

OTRAS MÉTRICAS

- ***RFP (Ratio Form Power):*** (lit. Eduard Barceló) el ratio del Form Power nos va a decir qué porcentaje de la potencia que estamos aplicando se va hacia arriba y no hacia adelante. Para utilizar este valor correctamente, tenemos que saber que:

  - ❖ A medida que la potencia bruta se incrementa, el ratio empeora.
  - ❖ A medida que se acumula la fatiga, el ratio empeora igualmente.
  - ❖ Si queremos compararnos entre corredores, hay que hacerlo a % del FTP iguales.

Finalmente como referencia para conocer el orden de magnitud del FPR hay que saber que al FTP y en condiciones de carrera llanas:

- ❖ Un valor superior al 25 % es para un corredor con mala técnica.
- ❖ Un valor entre el 23-25 % es la media para la mayoría de corredores populares.
- ❖ Un valor entre el 20-23 % es un valor muy bueno.
- ❖ Valores por debajo del 20 % los obtienen sólo corredores de élite de clase mundial.

$$RFP = \frac{FP\ (FORM\ POWER)}{POTENCIA\ (WATIOS)}$$

- ***RE (Running Efectiveness):*** *(lit. Eduard Barceló) e*ste parámetro es uno de los más interesantes que el dispositivo Stryd ofrece en relación a la técnica de carrera y cómo mejoramos nuestra efectividad.

  La efectividad de carrera es la ratio entre la velocidad y la potencia. Se calcula mediante el cociente de la velocidad en metros/segundo por la potencia en vatios/kilogramo.

  Dicho así suena complicado, muy complicado, pero el concepto es muy sencillo. Simplemente te indica a qué velocidad te permite correr 1 vatio/kg. Cuanta más velocidad puedas imprimir con ese vatio, mayor efectividad.

  Por tanto, cuanto más grande sea el número, mejor técnica tendrá el atleta y esto le permitirá, a igualdad de umbral de potencia funcional y potencia relativa que otro atleta, un rendimiento superior a su homólogo pero con un índice inferior de efectividad de carrera.

$$RE = \frac{Metros\ /\ Segundos}{Watios\ /\ kilogramos} = \frac{Distancia\ /\ Duracion}{Potencia/\ Peso\ Stryd}$$

- ***L_ZANCADA (Longitud de zancada):*** la L_ZANCADA mide la distancia medida en metros de cada zancada en segmentos específicos y para su obtención son necesarias diferentes métricas anteriormente vistas.

$$L\_ZANCADA = \frac{Distancia}{(Duracion\ /\ 60)\ x\ Cadencia}$$

- ***ROV (Ratio Oscilación Vertical):*** ratio entre la oscilación vertical y la longitud de zancada, a pesar de ser una métrica interesante para este modelo su aportación será más bien nula.

$$ROV = \frac{Oscilación\ Vertical}{Longitud\ de\ Zancada}$$

- ***RLSS (Ratio Leg Spring Stiffness):*** ratio entre LSS (muelle o resorte al correr) y Peso Stryd.

$$RLSS = \frac{LSS\ (Leg\ Spring\ Stiffness)}{Peso\ Stryd}$$

Notas Importante:

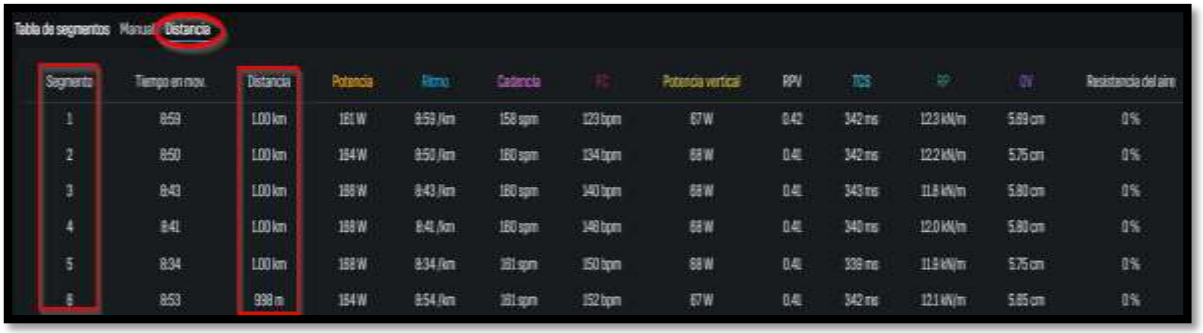
- **Excepto la métrica RFP, que es aportada por la aplicación Stryd, el resto de métricas para su estudio deberán ser calculadas.**
- **Cuando hablamos de Peso Stryd, nos estamos refiriendo al peso que se introdujo por primera vez en el dispositivo y que no se deberá cambiar para poder comparar las métricas de forma correcta.**

FUENTE DE DATOS

La herramienta Stryd dispone de su propia aplicación móvil y de su equivalente de escritorio en formato web, para poder extraer los datos de los entrenamientos deberemos descargarnos los archivos desde la aplicación móvil en formato csv. Por cada entrenamiento deberemos descargarnos 3 archivos aunque en principio sólo utilizaremos 2. Un archivo en los que las distancias de los segmentos se han definido manualmente y un segundo donde las distancia de los segmentos está predefinida a 1000m.

Manualmente aplicación web

Predefinidos en aplicación web



18:14

Segmentos

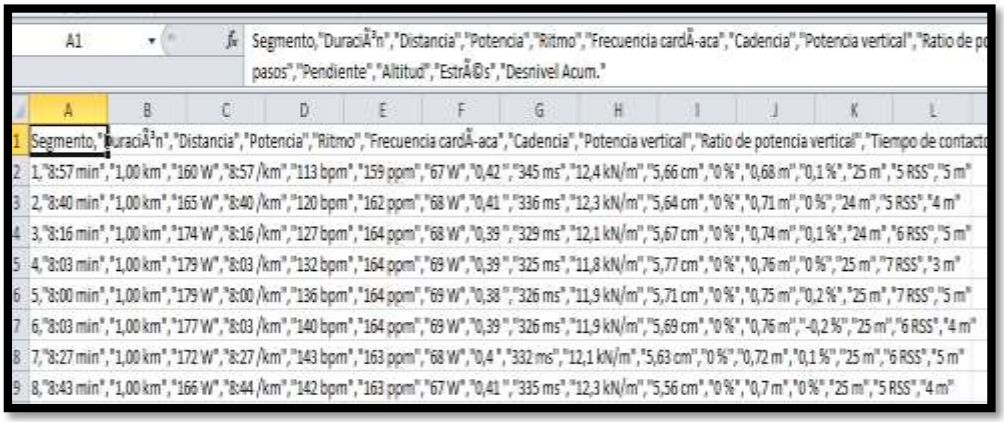
Segmento	Duración	Distancia	Potencia	Ritmo	Frecuencia cardíaca	Cadencia
1	8:59 min	1,00 km	161 W	8:59 /km	123 bpm	158 ppm
2	8:51 min	1,00 km	164 W	8:51 /km	134 bpm	160 ppm
3	8:43 min	1,00 km	166 W	8:43 /km	140 bpm	160 ppm
4	8:40 min	1,00 km	168 W	8:40 /km	146 bpm	160 ppm
5	8:34 min	1,00 km	168 W	8:34 /km	150 bpm	161 ppm
6	8:53 min	1,00 km	164 W	8:54 /km	152 bpm	161 ppm

18:14

Segmentos

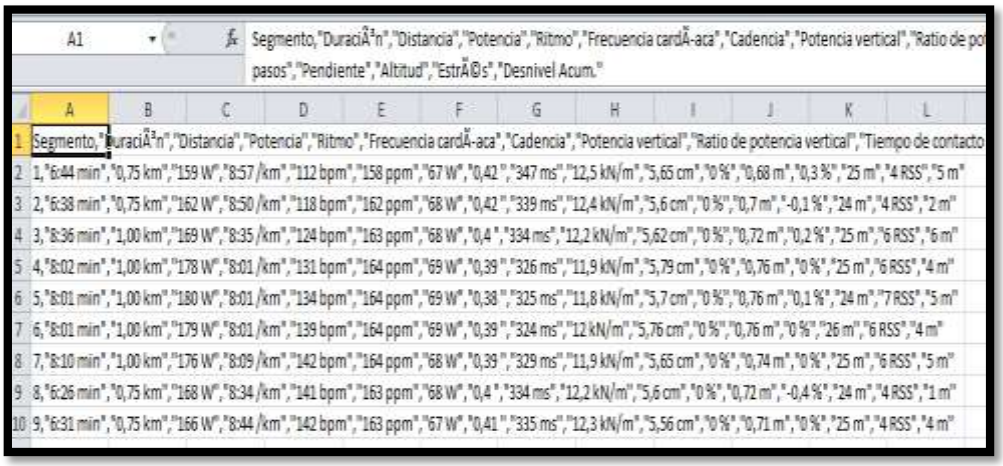
Segmento	Duración	Distancia	Potencia	Ritmo	Frecuencia cardíaca	Cadencia
1	6:52 min	0,76 km	160 W	9:04 /km	121 bpm	158 ppm
2	6:44 min	0,75 km	163 W	8:59 /km	132 bpm	160 ppm
3	8:44 min	1,00 km	165 W	8:44 /km	136 bpm	160 ppm
4	8:44 min	1,00 km	167 W	8:43 /km	144 bpm	160 ppm
5	8:29 min	1,00 km	170 W	8:29 /km	149 bpm	160 ppm
6	6:33 min	0,75 km	165 W	8:44 /km	150 bpm	161 ppm
7	6:34 min	0,74 km	164 W	8:50 /km	152 bpm	161 ppm

Mediante la herramienta Powercenter (no confundir con PowerCenter) de Stryd podremos obtener los datos de los segmentos que queramos y así poder obtener más información.



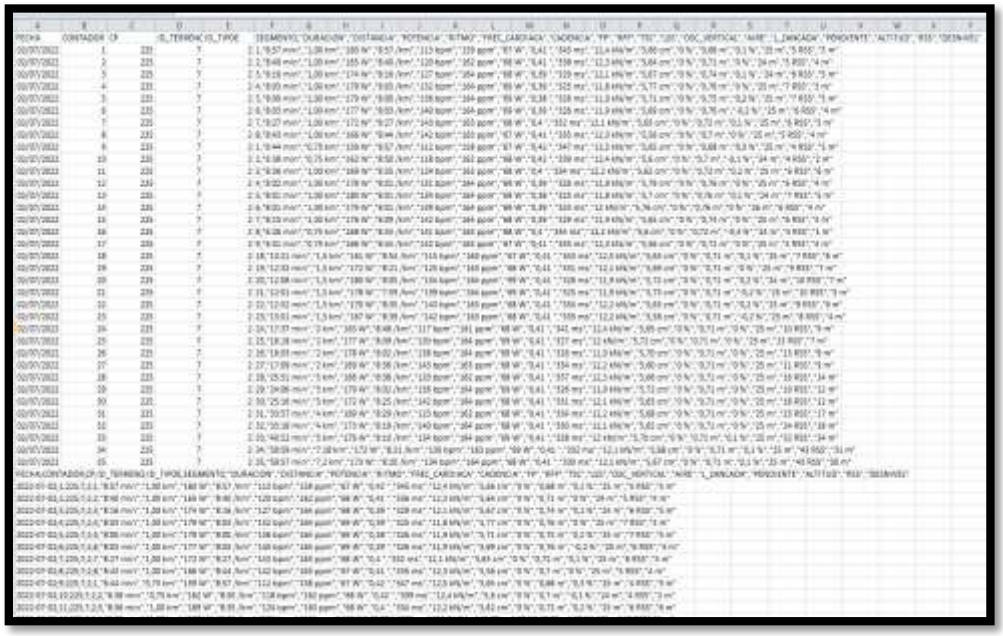
A1 Segmento,"DuraciÃ³n","Distancia","Potencia","Ritmo","Frecuencia cardÃ-aca","Cadencia","Potencia vertical","Ratio de pasos","Pendiente","Altitud","EstrÃ®s","Desnivel Acum."

	A	B	C	D	E	F	G	H	I	J	K	L
1	Segmento,"DuraciÃ³n","Distancia","Potencia","Ritmo","Frecuencia cardÃ-aca","Cadencia","Potencia vertical","Ratio de potencia vertical","Tiempo de contacto											
2	1,"8:57 min","1,00 km","160 W","8:57 /km","113 bpm","159 ppm","67 W","0,42 ","345 ms","12,4 kN/m","5,66 cm","0 %","0,68 m","0,1 %","25 m","5 RSS","5 m"											
3	2,"8:40 min","1,00 km","165 W","8:40 /km","120 bpm","162 ppm","68 W","0,41 ","336 ms","12,3 kN/m","5,64 cm","0 %","0,71 m","0 %","24 m","5 RSS","4 m"											
4	3,"8:16 min","1,00 km","174 W","8:16 /km","127 bpm","164 ppm","68 W","0,39 ","329 ms","12,1 kN/m","5,67 cm","0 %","0,74 m","0,1 %","24 m","6 RSS","5 m"											
5	4,"8:03 min","1,00 km","179 W","8:03 /km","132 bpm","164 ppm","69 W","0,39 ","325 ms","11,8 kN/m","5,77 cm","0 %","0,76 m","0 %","25 m","7 RSS","3 m"											
6	5,"8:00 min","1,00 km","179 W","8:00 /km","136 bpm","164 ppm","69 W","0,38 ","326 ms","11,9 kN/m","5,71 cm","0 %","0,75 m","0,2 %","25 m","7 RSS","5 m"											
7	6,"8:03 min","1,00 km","177 W","8:03 /km","140 bpm","164 ppm","69 W","0,39 ","326 ms","11,9 kN/m","5,69 cm","0 %","0,76 m","-0,2 %","25 m","6 RSS","4 m"											
8	7,"8:27 min","1,00 km","172 W","8:27 /km","143 bpm","163 ppm","68 W","0,4 ","332 ms","12,1 kN/m","5,63 cm","0 %","0,72 m","0,1 %","25 m","6 RSS","5 m"											
9	8,"8:43 min","1,00 km","166 W","8:44 /km","142 bpm","163 ppm","67 W","0,41 ","335 ms","12,3 kN/m","5,56 cm","0 %","0,7 m","0 %","25 m","5 RSS","4 m"											



A1 Segmento,"DuraciÃ³n","Distancia","Potencia","Ritmo","Frecuencia cardÃ-aca","Cadencia","Potencia vertical","Ratio de pasos","Pendiente","Altitud","EstrÃ®s","Desnivel Acum."

	A	B	C	D	E	F	G	H	I	J	K	L
1	Segmento,"DuraciÃ³n","Distancia","Potencia","Ritmo","Frecuencia cardÃ-aca","Cadencia","Potencia vertical","Ratio de potencia vertical","Tiempo de contacto											
2	1,"6:44 min","0,75 km","159 W","8:57 /km","112 bpm","158 ppm","67 W","0,42 ","347 ms","12,5 kN/m","5,65 cm","0 %","0,68 m","0,3 %","25 m","4 RSS","5 m"											
3	2,"6:38 min","0,75 km","162 W","8:50 /km","118 bpm","162 ppm","68 W","0,42 ","339 ms","12,4 kN/m","5,6 cm","0 %","0,7 m","-0,1 %","24 m","4 RSS","2 m"											
4	3,"8:36 min","1,00 km","169 W","8:35 /km","124 bpm","163 ppm","68 W","0,4 ","334 ms","12,2 kN/m","5,62 cm","0 %","0,72 m","0,2 %","25 m","6 RSS","6 m"											
5	4,"8:02 min","1,00 km","178 W","8:01 /km","131 bpm","164 ppm","69 W","0,39 ","326 ms","11,9 kN/m","5,79 cm","0 %","0,76 m","0 %","25 m","6 RSS","4 m"											
6	5,"8:01 min","1,00 km","180 W","8:01 /km","134 bpm","164 ppm","69 W","0,38 ","325 ms","11,8 kN/m","5,7 cm","0 %","0,76 m","0,1 %","24 m","7 RSS","5 m"											
7	6,"8:01 min","1,00 km","179 W","8:01 /km","139 bpm","164 ppm","69 W","0,39 ","324 ms","12 kN/m","5,76 cm","0 %","0,76 m","0 %","26 m","6 RSS","4 m"											
8	7,"8:10 min","1,00 km","176 W","8:09 /km","142 bpm","164 ppm","68 W","0,39 ","329 ms","11,9 kN/m","5,65 cm","0 %","0,74 m","0 %","25 m","6 RSS","5 m"											
9	8,"6:26 min","0,75 km","168 W","8:34 /km","141 bpm","163 ppm","68 W","0,4 ","334 ms","12,2 kN/m","5,6 cm","0 %","0,72 m","-0,4 %","24 m","4 RSS","1 m"											
10	9,"6:31 min","0,75 km","166 W","8:44 /km","142 bpm","163 ppm","67 W","0,41 ","335 ms","12,3 kN/m","5,56 cm","0 %","0,71 m","0 %","25 m","4 RSS","4 m"											



Tras unificar ***"SEGMENTOS1K"*** y ***"SEGMENTOS_MANUAL"*** realizaremos algunas transformaciones para adaptar los datos a nuestras necesidades en un archivo de texto plano ***"SEGMENTOS.txt"***

PROCESOS DE ETL

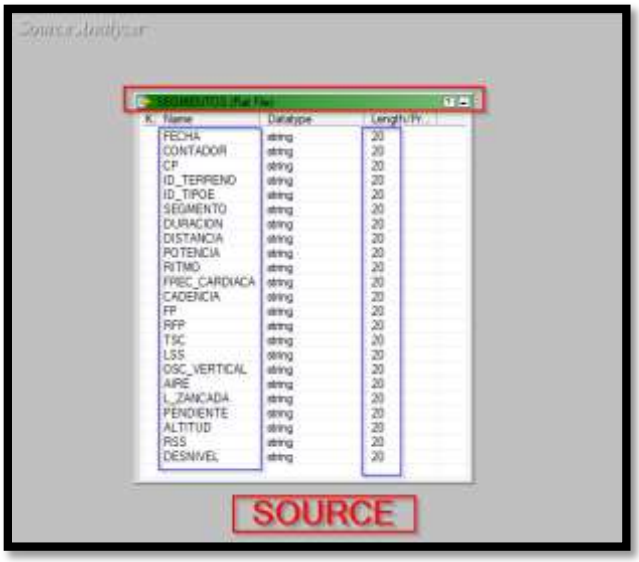
Nos podríamos extender y podríamos debatir sobre qué herramienta es mejor o peor, cual nos gusta más o menos o es más apropiada o menos apropiada para las tareas de ETL. Particularmente y por su sencillez utilizaremos ***Informatica PowerCenter*** y como base de datos SQL SERVER sin despreciar otras posibles herramientas y soluciones.

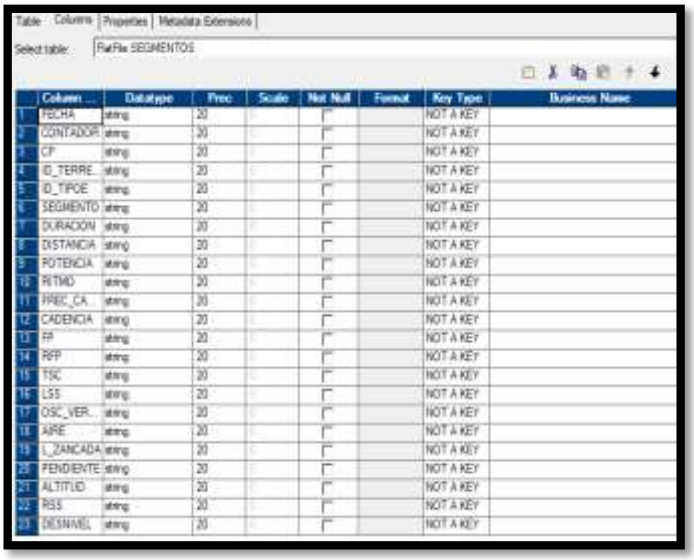
Deberemos crear diferentes "mappings" y otros tantos "workflows" con el objetivo de llevar nuestro archivo de texto plano "SEGMENTOS.txt" hasta la tabla "STG_SEGMENTOS", realizando conversiones de tipos, transformaciones, operaciones, filtros, etc…

MAPPING: m_SEGMENTOS_TXT_TO_TABLE

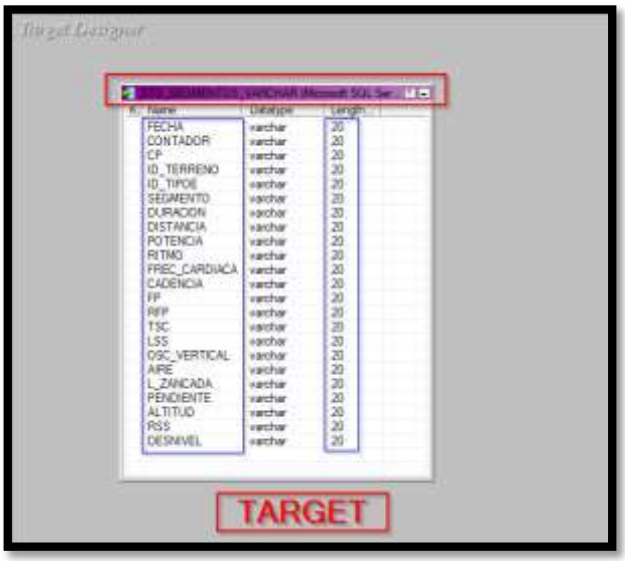
Mediante esta transformación, transportaremos nuestros almacenados en "SEGMENTOS.txt" a un formato más estructurado, es decir, formato tabla. Básicamente se trata de un mapeado directo desde la fuente (source) al destino (target), todos los campos se almacenarán en un primer momento en tipo varchar(20).

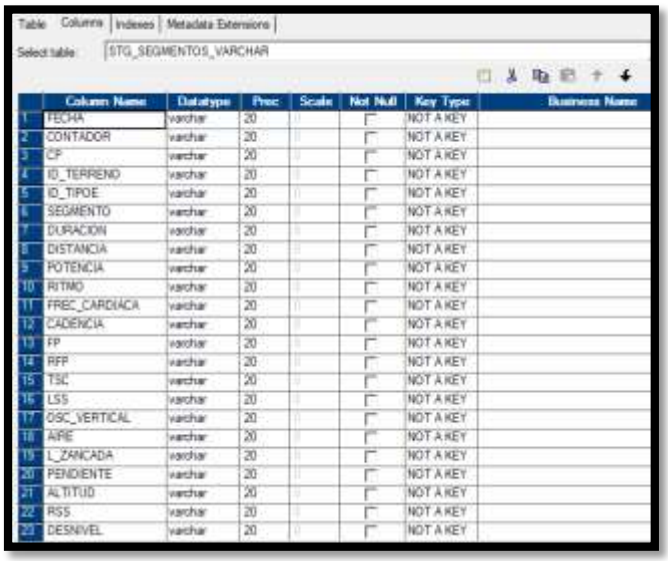
Source: SEGMENTOS.txt



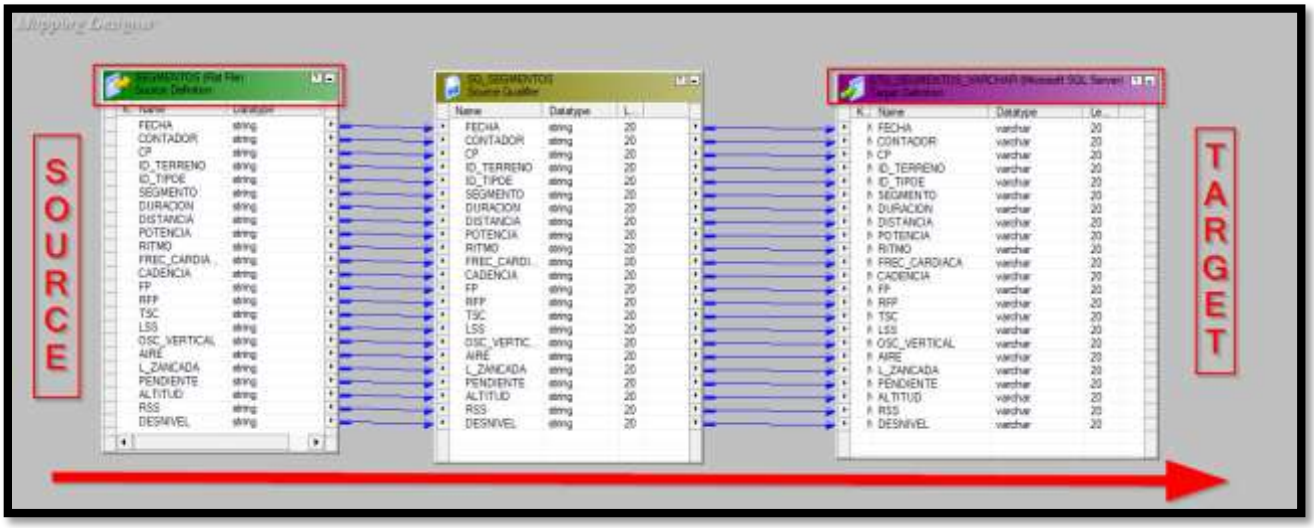


Target: STG_SEGMENTOS_VARCHAR

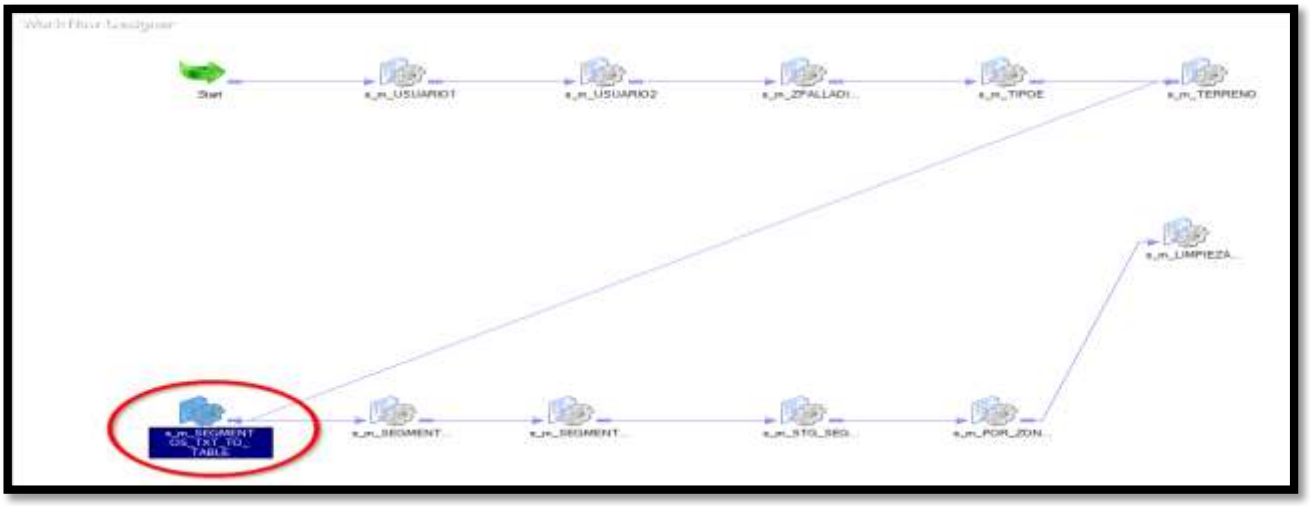




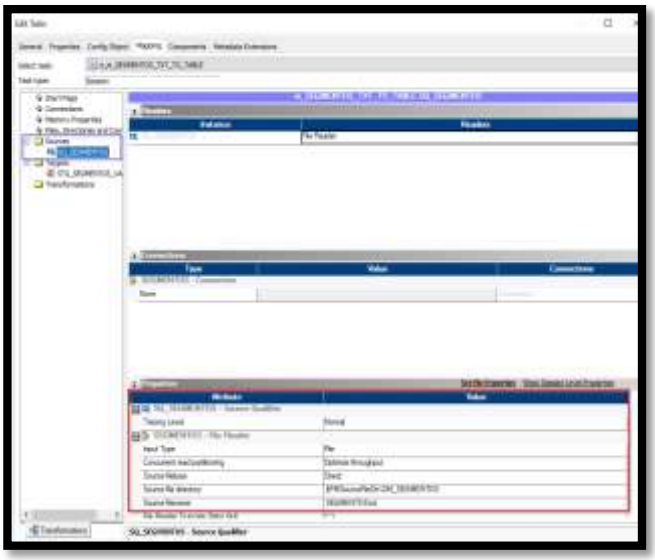
Mapping: m_SEGMENTOS_TXT_TO_TABLE

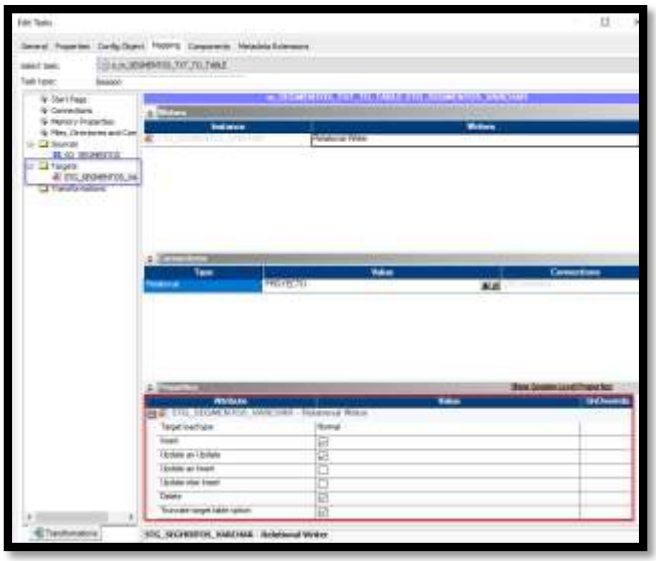


Workflow: wf_ACTUALIZACIONES

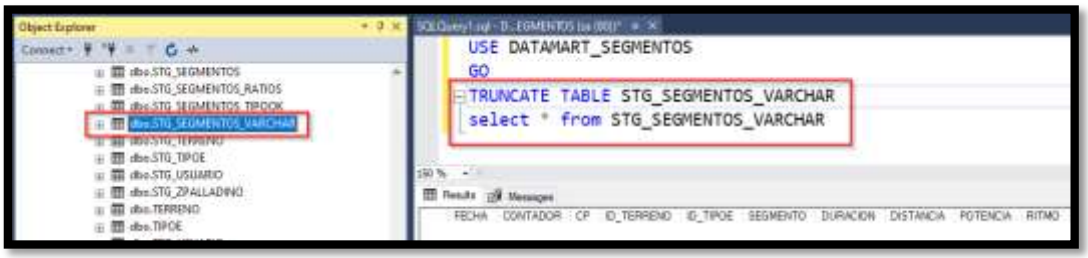


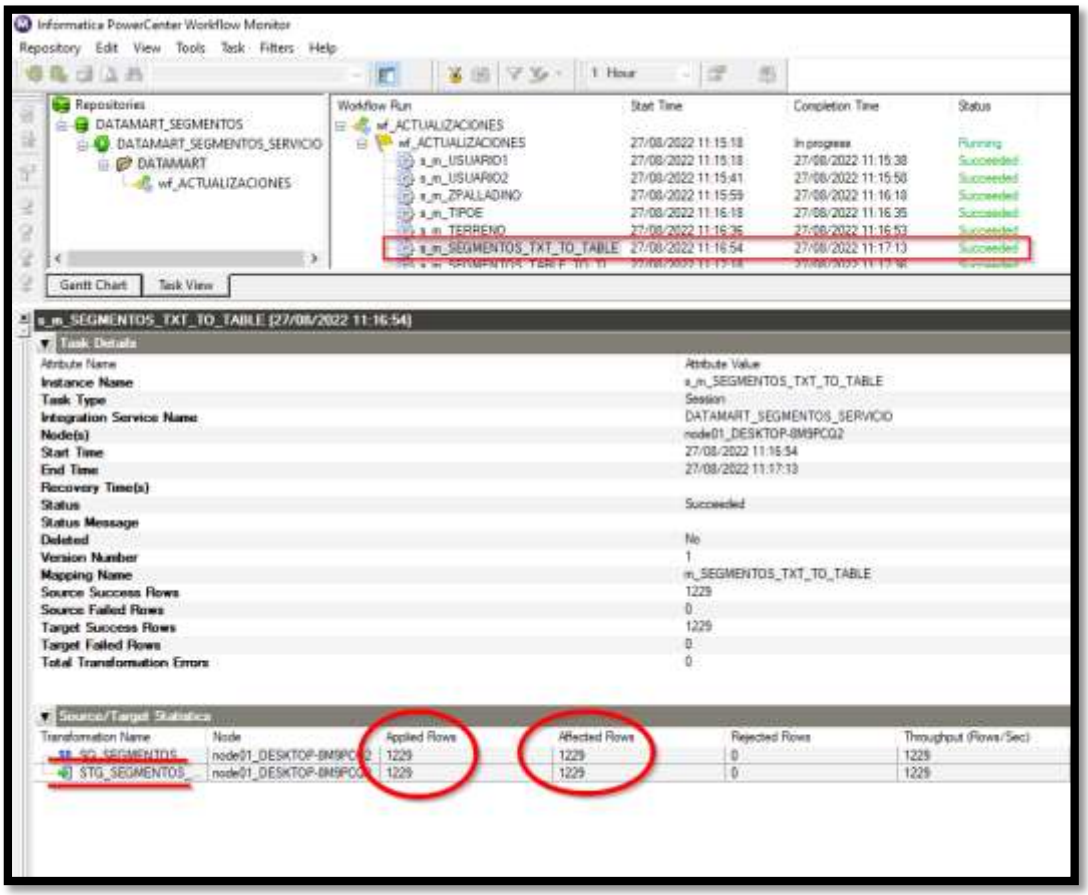
Task: s_m_SEGMENTOS_TXT_TO_TABLE

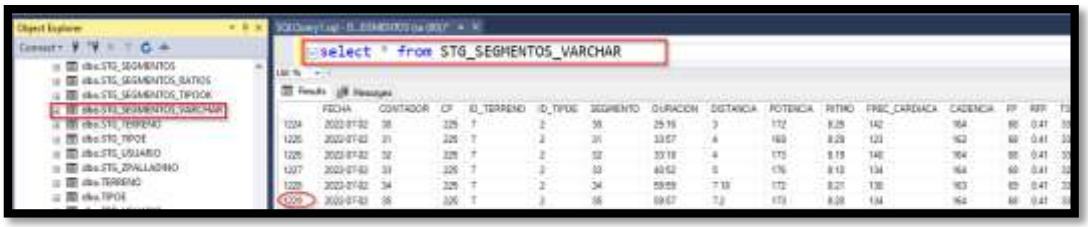




EJECUCIÓN: m_SEGMENTOS_TXT_TO_TABLE



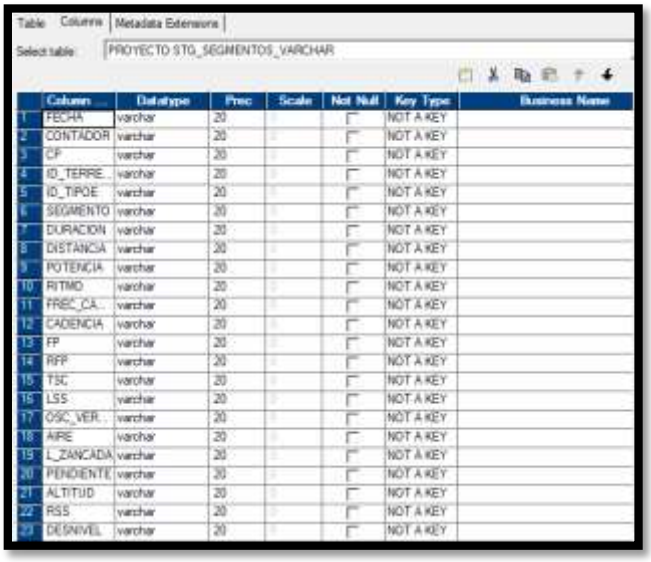




m_SEGMENTOS_TABLE_TO_TIPOOK

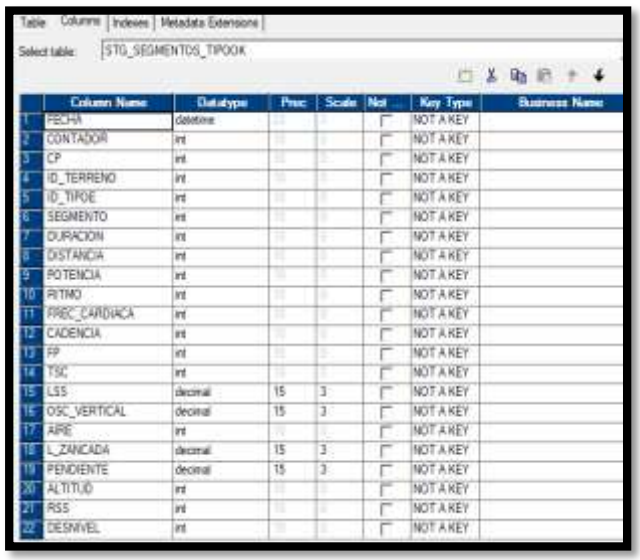
Mediante esta transformación, asignaremos a cada atributo su tipo correspondiente, es decir, en un primer momento los extraeremos de la tabla "SEGMENTOS_TXT_TO_TABLE" donde se encuentran todos con el tipo varchar (20) y le asignaremos el tipo con el que podamos trabajat. Como podremos observar deberemos hacer algunas transformaciones mecánicas.

Source: STG_SEGMENTOS_VARCHAR



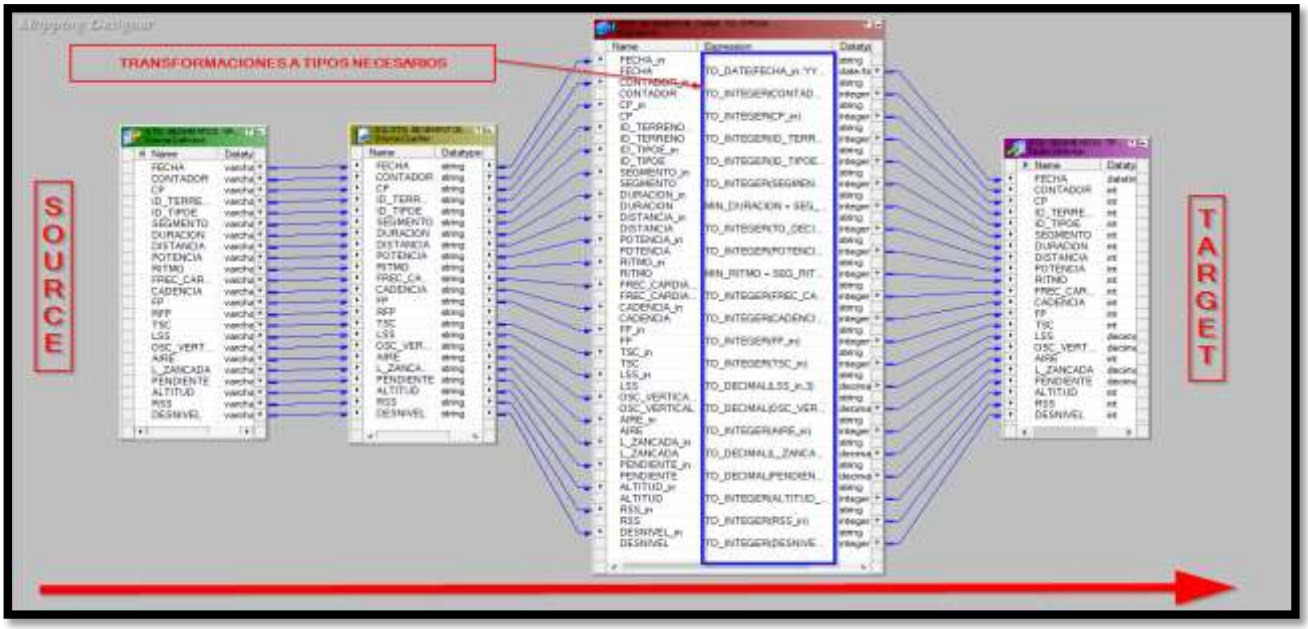
	Column	Datatype	Prec	Scale	Not Null	Key Type	Business Name
1	FECHA	varchar	20			NOT A KEY	
2	CONTADOR	varchar	20			NOT A KEY	
3	CP	varchar	20			NOT A KEY	
4	ID_TERRE...	varchar	20			NOT A KEY	
5	ID_TIPOE	varchar	20			NOT A KEY	
6	SEGMENTO	varchar	20			NOT A KEY	
7	DURACION	varchar	20			NOT A KEY	
8	DISTANCIA	varchar	20			NOT A KEY	
9	POTENCIA	varchar	20			NOT A KEY	
10	RITMO	varchar	20			NOT A KEY	
11	FREC_CA...	varchar	20			NOT A KEY	
12	CADENCIA	varchar	20			NOT A KEY	
13	FP	varchar	20			NOT A KEY	
14	RFP	varchar	20			NOT A KEY	
15	TSC	varchar	20			NOT A KEY	
16	LSS	varchar	20			NOT A KEY	
17	OSC_VER...	varchar	20			NOT A KEY	
18	AIRE	varchar	20			NOT A KEY	
19	L_ZANCADA	varchar	20			NOT A KEY	
20	PENDIENTE	varchar	20			NOT A KEY	
21	ALTITUD	varchar	20			NOT A KEY	
22	RSS	varchar	20			NOT A KEY	
23	DESNIVEL	varchar	20			NOT A KEY	

Target: STG_SEGMENTOS_TIPOOK

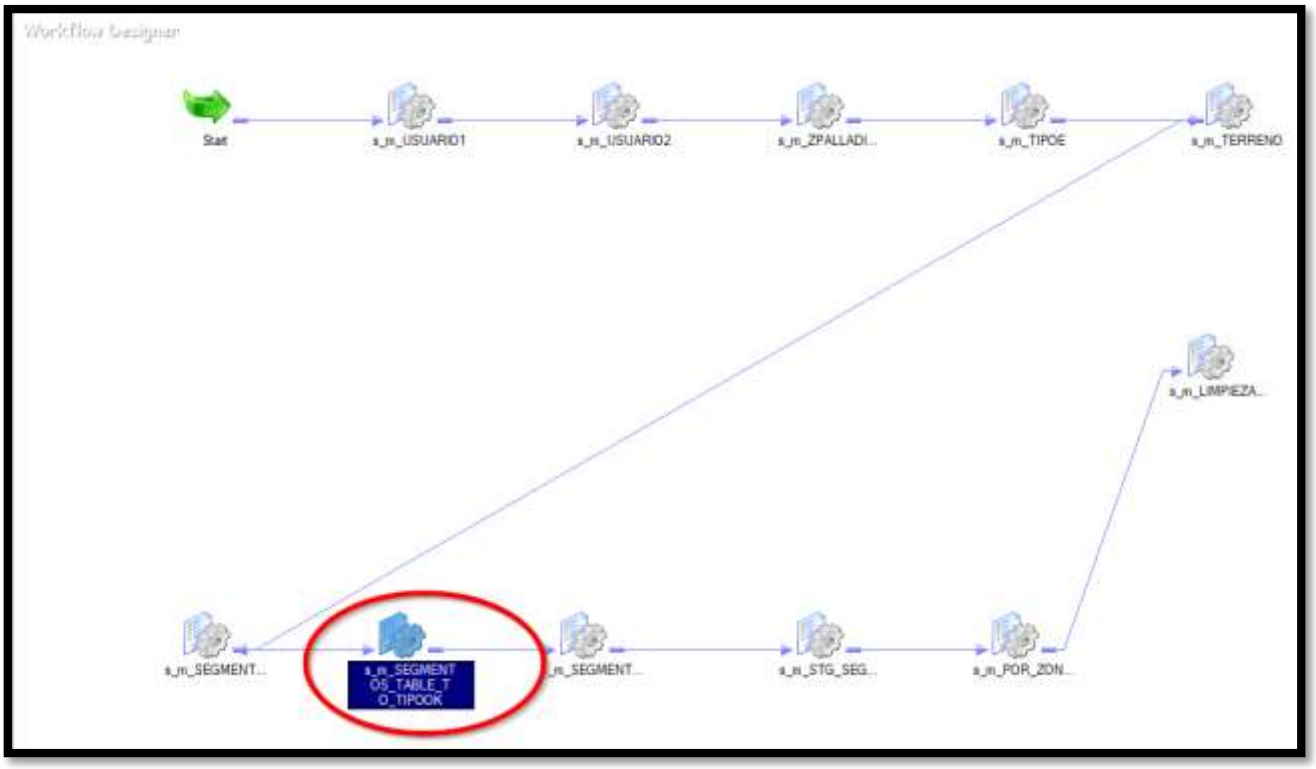


	Column Name	Datatype	Prec	Scale	Not ...	Key Type	Business Name
1	FECHA	datetime				NOT A KEY	
2	CONTADOR	int				NOT A KEY	
3	CP	int				NOT A KEY	
4	ID_TERRENO	int				NOT A KEY	
5	ID_TIPOE	int				NOT A KEY	
6	SEGMENTO	int				NOT A KEY	
7	DURACION	int				NOT A KEY	
8	DISTANCIA	int				NOT A KEY	
9	POTENCIA	int				NOT A KEY	
10	RITMO	int				NOT A KEY	
11	FREC_CARDIACA	int				NOT A KEY	
12	CADENCIA	int				NOT A KEY	
13	FP	int				NOT A KEY	
14	TSC	int				NOT A KEY	
15	LSS	decimal	15	3		NOT A KEY	
16	OSC_VERTICAL	decimal	15	3		NOT A KEY	
17	AIRE	int				NOT A KEY	
18	L_ZANCADA	decimal	15	3		NOT A KEY	
19	PENDIENTE	decimal	15	3		NOT A KEY	
20	ALTITUD	int				NOT A KEY	
21	RSS	int				NOT A KEY	
22	DESNIVEL	int				NOT A KEY	

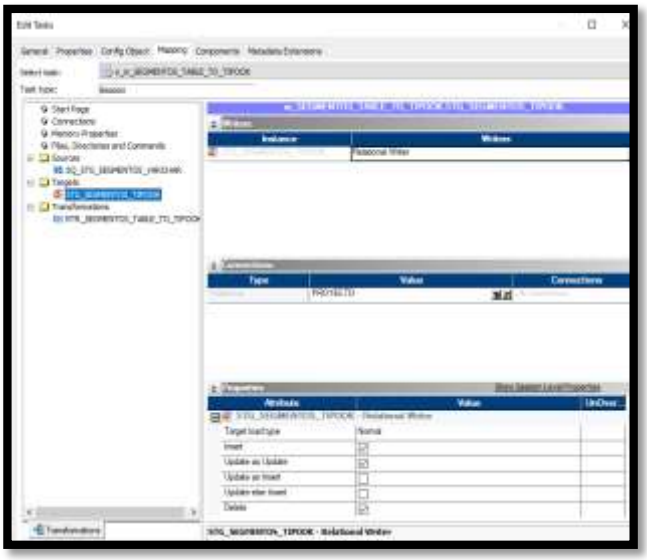
Mapping: m_SEGMENTOS_TABLE_TO_TIPOOK

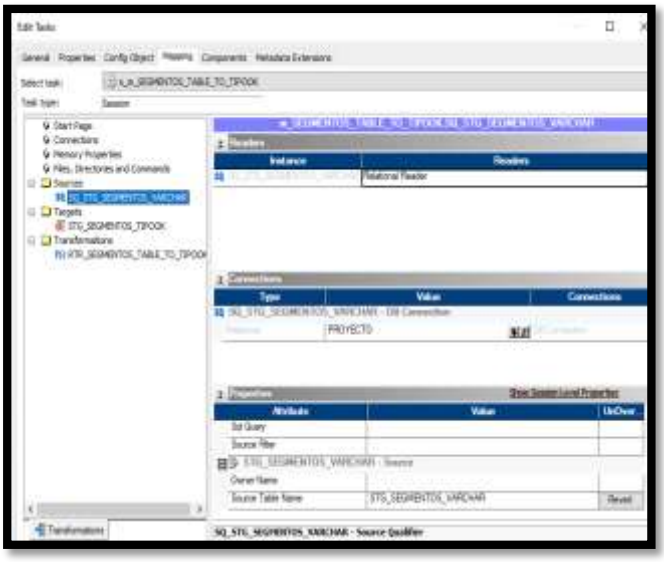


Workflow: wf_ACTUALIZACIONES

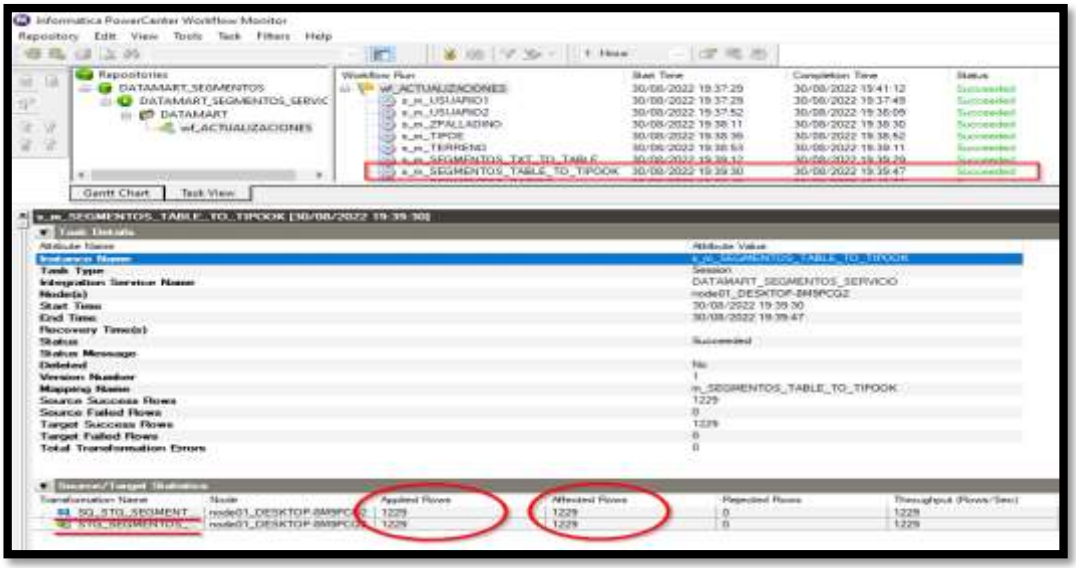


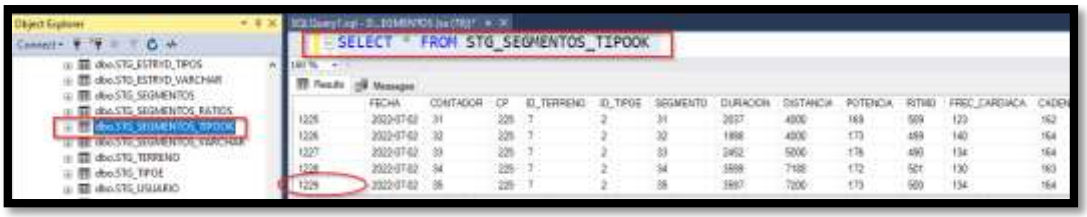
Task: s_m_SEGMENTOS_TABLE_TO_TIPOOK





EJECUCIÓN: m_SEGMENTOS_TABLE_TO_TIPOOK





FASE ANÁLISIS DE LOS DATOS

PREÁMBULO FASE DE ANÁLISIS DE DATOS

Unos de los errores principales que se cometen cuando se comienza a trabajar con machine learning es tomar decisiones directamente a través de los algoritmos sin un análisis previo del conjunto de datos a trabajar. Es importante que entendamos que, un análisis de datos y un buen preprocesamiento de los mismos antes de realizar un tratamiento de modelado, nos llevará no solamente a obtener mejores y más confiables resultados, sino que entenderemos a la perfección nuestro conjunto de datos dando una gran experiencia en el tiempo en la fase de análisis y tratamiento de datos.

La inspección de nuestro conjunto de datos es fundamental para poder entender mejor que técnica utilizar; además, nos ayudará a desarrollar nuestra intuición y hacernos preguntas sobre ellos. Las múltiples perspectivas de sus datos lo desafiarán a pensar en los datos de manera diferente.

En esta fase intentaremos extraer información de nuestros datos para poder entender mejor la distribución de los mismos. Para ello deberemos utilizar algunas herramientas o instrucciones que faciliten esta labor.

Necesidad de comprender los datos a la perfección.

- Finalidad: Tener un modelo robusto y fiable
- Entender nuestros datos a través de la observación estadística como: las dimensiones, tipo de datos, distribución de clases, entre otros.
- Trabajar estadísticas avanzadas en el análisis de un conjunto de datos como desviaciones estándar y sesgo de los datos.
- Entender las relaciones entre los atributos mediante el cálculo de correlaciones.

Python (y sus librerías) nos da una gran variedad de funciones para conocer en profundidad nuestros datos.

- A través de ellas vamos a conocer muchos detalles del conjunto de datos.
- Nos aproximamos a qué técnica dará mejor resultado.

ESTADÍSTICA DESCRIPTIVA

Librerías necesarias

Importaremos las librerías necesarias para nuestro proyecto, se podrán ir añadiendo según necesidades.

```
# ********************************************************************************
# ***************              LIBRERÍAS A IMPORTAR              ***************
# ********************************************************************************
import pandas                       as pd
import matplotlib.pyplot            as plt
import pandas                       as pd
import numpy                        as np
import seaborn                      as sns
```

Cargamos nuestro DataFrame ()

```
# Cargamos los datos contenidos en "SEGMENTOS_csv.csv"
data = pd.read_csv('SEGMENTOS_csv.csv')
print(data)
```

```
C:\Users\Usuario\anaconda3\envs\david\python.exe C:/Users/Usuario/PycharmProjects/ana1/PROYECTO_STRYD/Fase_Analisis_Datos.py
      POR_CP  DISTANCIA  RITMO  FREC_CARDIACA  ...  ALTITUD  DESNIVEL  RSS  DURACION
0     85.863        500    519            154  ...       19       2.0    5       259
1     84.816       1000    547            130  ...       16       7.0   10       547
2     84.293       1000    537            137  ...       18       4.0    9       537
3     84.816       1000    540            142  ...       18       6.0    9       540
4     84.816       1000    539            146  ...       19       5.0    9       539
...      ...        ...    ...            ...  ...      ...       ...  ...       ...
1224  79.555       1000    480            136  ...       25       5.0    7       480
1225  78.666       1000    483            140  ...       25       4.0    6       483
1226  76.444       1000    507            143  ...       25       5.0    6       507
1227  73.777       1000    524            142  ...       25       4.0    5       523
1228  70.666        750    537            112  ...       25       5.0    4       404

[1229 rows x 20 columns]
```

Revisar los datos: head()

Revisaremos las primeras 15 filas de los datos utilizando la función ***head ()*** en el DataFrame de Pandas. Puede ver que la primera columna enumera el número de fila, lo cual es útil para hacer referencia a una observación específica.

```
# ********************************************************************************
# ***************              ANÁLISIS DE LOS DATOS              ***************
# ********************************************************************************
# Función head()
print(data.head(15))
```

```
    POR_CP  DISTANCIA  RITMO  FREC_CARDIACA  ...  ALTITUD  DESNIVEL  RSS  DURACION
0   85.863        500    519            154  ...       19       2.0    5       259
1   84.816       1000    547            130  ...       16       7.0   10       547
2   84.293       1000    537            137  ...       18       4.0    9       537
3   84.816       1000    540            142  ...       18       6.0    9       540
4   84.816       1000    539            146  ...       19       5.0    9       539
5   84.816       1000    542            150  ...       18       7.0    9       542
6   84.816       1000    537            152  ...       18       2.0    9       537
7   85.863        600    524            155  ...       19       3.0    6       317
8   85.340        530    561            128  ...       16       5.0    5       300
9   84.816       5570    538            145  ...       18      27.0   51      3000
10  84.816       2000    545            134  ...       17      11.0   18      1082
11  84.816       2000    540            144  ...       19      11.0   18      1080
12  84.816       2000    539            151  ...       18       9.0   18      1079
13  84.816       3000    541            136  ...       17      17.0   28      1622
14  84.816       3000    539            149  ...       18      13.0   28      1619

[15 rows x 20 columns]
```

Dimensiones de los datos: shape

Podemos revisar la forma y el tamaño del conjunto de datos imprimiendo la propiedad ***shape*** en el DataFrame de Pandas (data). Los resultados se enumeran en filas y luego en columnas. Vemos que el conjunto de datos tiene 1229 filas y 20 columnas.

```
# ********************************************************************************
# ***************              ANÁLISIS DE LOS DATOS              ***************
# ********************************************************************************
# Función shape
print(data.shape)
```

```
(1229, 20)
```

Tipo de datos: dtypes

Podemos enumerar los tipos de datos utilizados por el DataFrame (data) para caracterizar cada atributo utilizando la propiedad ***dtypes.*** Como podemos observar los atributos o características de dividen en dos tipos diferentes:

- ***float64:*** POR_CP, LSS, OSC_VERTICAL, L_ZANCADA, RFP, RLSS, ROV, RE, PENDIENTE y DESNIVEL.
- ***int64:*** DISTANCIA, RITMO, FREC_CARDIACA, CADENCIA, TSC, FP, AIRE, ALTITUD, RSS y DURACION.

```
# ********************************************************************************
# ***************              ANÁLISIS DE LOS DATOS              ***************
# ********************************************************************************
# Función dtaypes
print(data.dtypes)
```

```
POR_CP           float64
DISTANCIA          int64
RITMO              int64
FREC_CARDIACA      int64
CADENCIA           int64
TSC                int64
FP                 int64
LSS              float64
OSC_VERTICAL     float64
L_ZANCADA        float64
```

```
RFP              float64
RLSS             float64
ROV              float64
RE               float64
AIRE               int64
PENDIENTE        float64
ALTITUD            int64
DESNIVEL         float64
RSS                int64
DURACION           int64
dtype: object
```

Resumen: describe ()

Vemos que obtenemos muchos datos. Al describir los datos de esta manera, vale la pena tomarse un tiempo y revisar las observaciones de los resultados. Observando los datos podremos intuir situaciones anómalas (a priori) como por ejemplo la diferencia que hay entre min y max en DISTANCIA y una varianza (std) tan elevada. Es tan sólo una pequeña muestra de las acciones que debemos llevar a cabo para comprender mejor el comportamiento de nuestros datos.

```
# ********************************************************************************
# ***************              ANÁLISIS DE LOS DATOS              ***************
# ********************************************************************************
# Función describe()
print(data.describe())
```

```
          POR_CP    DISTANCIA  ...         RSS     DURACION
count  1229.000000  1229.000000  ...  1229.000000  1229.000000
mean     81.949606  1517.453214  ...    11.626526   771.563059
std      14.843470  1359.907645  ...    11.190080   686.143498
min      65.137000    50.000000  ...     1.000000    19.000000
25%      74.770000   800.000000  ...     6.000000   411.000000
50%      77.981000  1000.000000  ...     8.000000   520.000000
75%      83.944000  1500.000000  ...    13.000000   817.000000
max     178.899000  7600.000000  ...   103.000000  3599.000000

[8 rows x 20 columns]
```

	POR_CP	DISTANCIA	RITMO	FREC_CARDIACA	CADENCIA	TSC	FP	LSS	OSC_VERTICAL	L_ZANCADA
count	1229.000000	1229.000000	1229.000000	1229.000000	1229.000000	1229.000000	1229.000000	1229.000000	1229.000000	1229.000000
mean	81.949606	1517.453214	501.777055	134.550041	162.510171	325.179007	67.505289	11.816762	5590.802311	54.375820
std	14.843470	1359.907645	59.278623	12.763259	5.278338	41.207236	3.472228	1.186280	545.138649	255.487712
min	65.137000	50.000000	209.000000	108.000000	130.000000	67.000000	40.000000	2.200000	0.980000	0.622000
25%	74.770000	800.000000	489.000000	124.000000	161.000000	328.000000	67.000000	11.700000	5540.000000	0.697000
50%	77.981000	1000.000000	513.000000	135.000000	162.000000	336.000000	68.000000	12.000000	5640.000000	0.721000
75%	83.944000	1500.000000	535.000000	144.000000	164.000000	342.000000	68.000000	12.300000	5730.000000	0.753000
max	178.899000	7600.000000	637.000000	171.000000	191.000000	360.000000	83.000000	13.600000	7120.000000	1589.000000

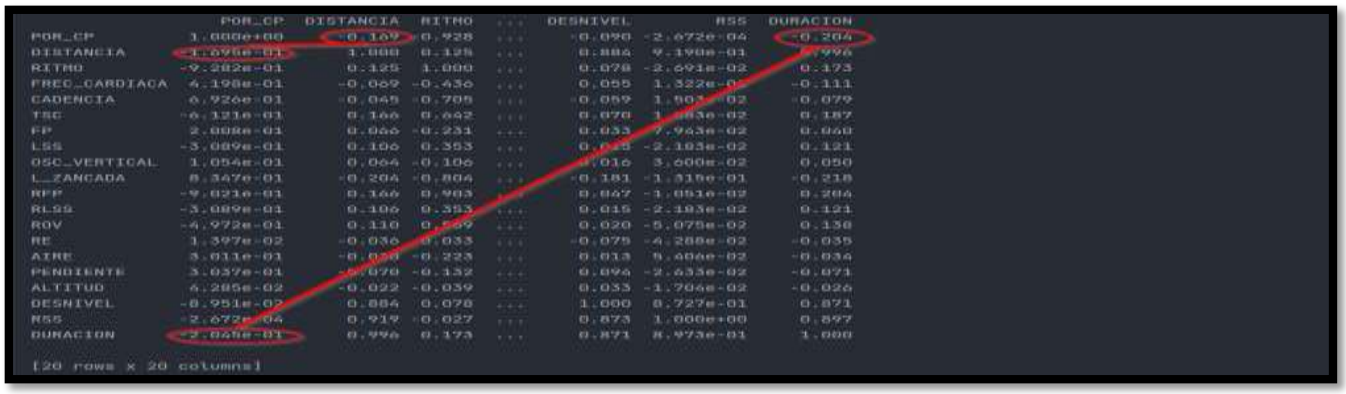
RFP	RLSS	ROV	RE	AIRE	PENDIENTE	ALTITUD	DESNIVEL	RSS	DURACION
1229.000000	1229.000000	1229.000000	1229.000000	1229.000000	1229.000000	1229.000000	1228.000000	1229.000000	1229.000000
38.749317	144.106313	7.676790	0.856816	0.537836	38.611798	20.847844	8.609935	11.626526	771.563059
4.314841	14.466858	0.860692	0.256163	0.670147	535.671963	22.845511	8.532032	11.190080	686.143498
16.422000	26.829000	1.342000	0.008000	0.000000	-4400.000000	1.000000	0.000000	1.000000	19.000000
38.068000	142.682000	7.486000	0.925000	0.000000	-0.100000	15.000000	4.000000	6.000000	411.000000
39.759000	146.341000	7.830000	0.938000	0.000000	0.000000	18.000000	6.000000	8.000000	520.000000
41.040000	150.000000	8.085000	0.947000	1.000000	0.200000	21.000000	10.000000	13.000000	817.000000
44.444000	165.853000	10.787000	1.186000	6.000000	4400.000000	238.000000	77.000000	103.000000	3599.000000

```
          POR_CP  DISTANCIA     RITMO  ...  DESNIVEL      RSS  DURACION
count  1229.000    1229.000  1229.000  ...  1228.000  1229.000  1229.000
mean     81.950    1517.453   501.777  ...     8.610    11.627   771.563
std      14.843    1359.908    59.279  ...     8.532    11.190   686.143
min      65.137      50.000   209.000  ...     0.000     1.000    19.000
25%      74.770     800.000   489.000  ...     4.000     6.000   411.000
50%      77.981    1000.000   513.000  ...     6.000     8.000   520.000
75%      83.944    1500.000   535.000  ...    10.000    13.000   817.000
max     178.899    7600.000   637.000  ...    77.000   103.000  3599.000
```

Correlaciones: corr()

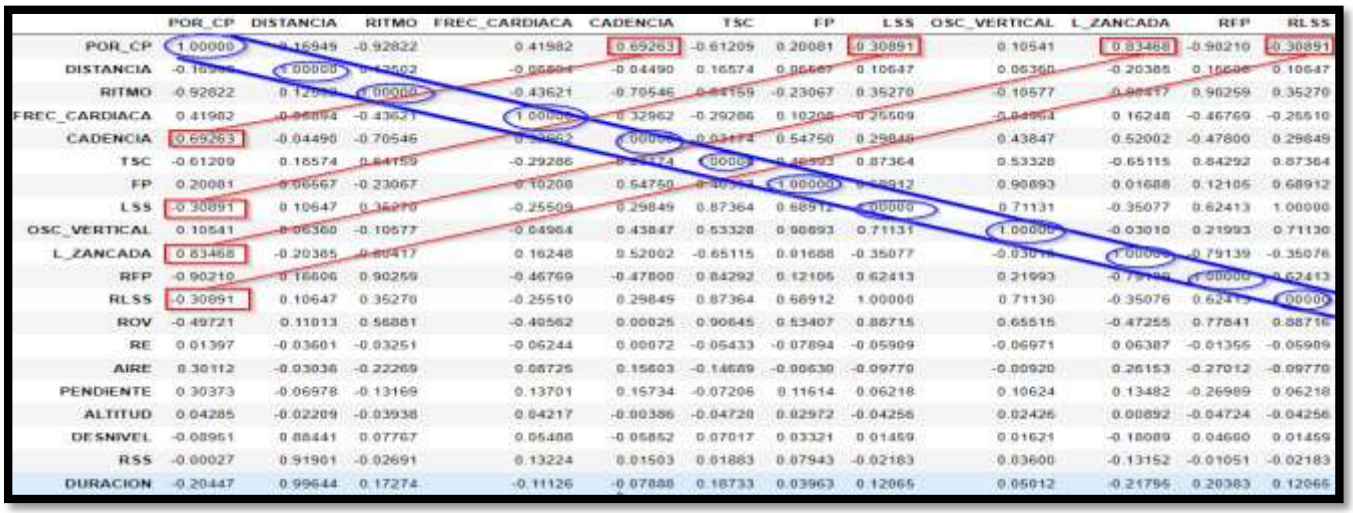
Podemos usar la función corr() para calcular una matriz de correlación. La matriz enumera todos los atributos en la parte superior y lateral, para dar correlación entre todos los pares de atributos (dos veces, porque la matriz es simétrica). Puede ver que la línea diagonal a través de la matriz desde las esquinas superior izquierda a inferior derecha de la matriz muestra una correlación perfecta de cada atributo consigo mismo. En un análisis gráfico podremos observar mejor esta correlación de índices.

```
# ************************************************************************
# ***************              ANÁLISIS DE LOS DATOS              ***************
# ************************************************************************
# Función describe()
pd.set_option('display.width',      100)
pd.set_option('display.precision',  5)
print(data.corr())
```



	POR_CP	DISTANCIA	RITMO	...	DESNIVEL	RSS	DURACION
POR_CP	1.000e+00	-0.169	0.928	...	-0.090	-2.672e-04	-0.204
DISTANCIA	-1.690e-01	1.000	0.125	...	0.884	9.190e-01	0.996
RITMO	-9.282e-01	0.125	1.000	...	0.078	-2.691e-02	0.173
FREC_CARDIACA	4.198e-01	-0.069	-0.436	...	0.055	1.322e-02	-0.111
CADENCIA	6.926e-01	-0.045	-0.705	...	-0.059	1.801e-02	-0.079
TSC	-6.121e-01	0.166	0.642	...	0.070	1.883e-02	0.187
FP	2.008e-01	0.066	-0.231	...	0.033	7.963e-02	0.060
LSS	-3.089e-01	0.106	0.353	...	0.015	-2.193e-02	0.121
OSC_VERTICAL	1.054e-01	0.064	-0.106	...	1.016	3.600e-02	0.050
L_ZANCADA	8.347e-01	-0.204	-0.804	...	-0.181	-1.319e-01	-0.218
RFP	-9.021e-01	0.166	0.903	...	0.047	-1.091e-02	0.204
RLSS	-3.089e-01	0.106	0.353	...	0.015	-2.193e-02	0.121
ROV	-4.972e-01	0.110	0.599	...	0.020	-5.075e-02	0.130
RE	1.397e-02	-0.036	-0.033	...	-0.075	-4.288e-02	-0.035
AIRE	3.011e-01	-0.030	-0.223	...	0.013	9.406e-02	-0.034
PENDIENTE	3.037e-01	-0.070	-0.132	...	0.096	-2.433e-02	-0.071
ALTITUD	6.285e-02	-0.022	-0.039	...	0.033	-1.706e-02	-0.026
DESNIVEL	-8.951e-02	0.884	0.078	...	1.000	8.727e-01	0.871
RSS	-2.672e-04	0.919	-0.027	...	0.873	1.000e+00	0.897
DURACION	-2.048e-01	0.996	0.173	...	0.871	8.973e-01	1.000

[20 rows x 20 columns]



	POR_CP	DISTANCIA	RITMO	FREC_CARDIACA	CADENCIA	TSC	FP	LSS	OSC_VERTICAL	L_ZANCADA	RFP	RLSS
POR_CP	1.00000	-0.16949	-0.92822	0.41982	0.69263	-0.61209	0.20081	-0.30891	0.10541	0.83488	-0.90210	-0.30891
DISTANCIA	-0.16949	1.00000	0.12502	-0.06894	-0.04490	0.16574	0.06567	0.10647	0.06360	-0.20385	0.16606	0.10647
RITMO	-0.92822	0.12502	1.00000	-0.43621	-0.70546	0.64159	-0.23067	0.35270	-0.10577	0.80417	0.90259	0.35270
FREC_CARDIACA	0.41982	-0.06894	-0.43621	1.00000	0.32962	-0.29286	0.10206	-0.25509	0.04964	0.16248	-0.46769	-0.25510
CADENCIA	0.69263	-0.04490	-0.70546	0.32962	1.00000	-0.53174	0.54750	0.29849	0.43847	0.52002	-0.47800	0.29849
TSC	-0.61209	0.16574	0.64159	-0.29286	-0.53174	1.00000	0.48523	0.87364	0.53328	-0.65115	0.84292	0.87364
FP	0.20081	0.06567	-0.23067	0.10206	0.54750	0.48523	1.00000	0.68912	0.90893	0.01688	0.12105	0.68912
LSS	-0.30891	0.10647	0.35270	-0.25509	0.29849	0.87364	0.68912	1.00000	0.71131	-0.35077	0.62413	1.00000
OSC_VERTICAL	0.10541	0.06360	-0.10577	0.04964	0.43847	0.53328	0.90893	0.71131	1.00000	-0.03010	0.21993	0.71130
L_ZANCADA	0.83488	-0.20385	-0.80417	0.16248	0.52002	-0.65115	0.01688	-0.35077	-0.03010	1.00000	-0.79139	-0.35076
RFP	-0.90210	0.16606	0.90259	-0.46769	-0.47800	0.84292	0.12105	0.62413	0.21993	-0.79139	1.00000	0.62413
RLSS	-0.30891	0.10647	0.35270	-0.25510	0.29849	0.87364	0.68912	1.00000	0.71130	-0.35076	0.62413	1.00000
ROV	-0.49721	0.11013	0.58881	-0.40562	0.00025	0.90645	0.53407	0.86715	0.65515	-0.47255	0.77841	0.86716
RE	0.01397	-0.03601	-0.03251	-0.06244	0.00072	-0.05433	-0.07894	-0.05909	-0.06971	0.06387	-0.01356	-0.05909
AIRE	0.30112	-0.03036	-0.22269	0.08725	0.15603	-0.14689	-0.00630	-0.09779	-0.00920	0.26153	-0.27012	-0.09770
PENDIENTE	0.30373	-0.06978	-0.13169	0.13701	0.15734	-0.07206	0.11614	0.06218	0.10624	0.13482	-0.26989	0.06218
ALTITUD	0.04285	-0.02209	-0.03938	0.04217	-0.00386	-0.04720	0.02972	-0.04256	0.02426	0.00892	-0.04724	-0.04256
DESNIVEL	-0.08951	0.88441	0.07767	0.05488	-0.05852	0.07017	0.03321	0.01459	0.01621	-0.18089	0.04600	0.01459
RSS	-0.00027	0.91901	-0.02691	0.13224	0.01503	0.01883	0.07943	-0.02183	0.03600	-0.13152	-0.01051	-0.02183
DURACION	-0.20447	0.99644	0.17274	-0.11126	-0.07888	0.18733	0.03963	0.12065	0.05012	-0.21795	0.20383	0.12066

Asimetría: skew()

Podemos calcular el sesgo de cada atributo utilizando la función skew(). El resultado de inclinación muestra una inclinación positiva (derecha) o negativa (izquierda). Los valores más cercanos a cero muestran menos sesgo. Mediante la visualización de la distribución de los datos podremos confirmar la existencia de sesgo.

```
# ************************************************************************
# ***************              ANÁLISIS DE LOS DATOS              ***************
# ************************************************************************
# Función skew()
print(data.skew())
```

```
[20 rows x 20 columns]
POR_CP          3.48925
DISTANCIA       2.52806
RITMO          -2.61498
FREC_CARDIACA   0.12813
CADENCIA       -0.66854
TSC            -3.94731
FP             -3.10833
LSS            -5.28746
OSC_VERTICAL   -5.05046
L_ZANCADA       4.66118
RFP            -2.86990
RLSS           -5.28741
ROV            -3.45735
RE             -2.58906
AIRE            1.36583
PENDIENTE       1.87532
ALTITUD         8.09177
DESNIVEL        2.81411
RSS             3.01400
DURACION        2.45963
dtype: float64
```

Algunas métricas a tener en cuenta

```
# Descripción de los datos y de diferentes métricas
# como media, varianza, percentiles,
# mínimos, máximos, etc.....
pd.set_option('display.width',      100)
pd.set_option('display.precision',  3)
print(data.describe())
```

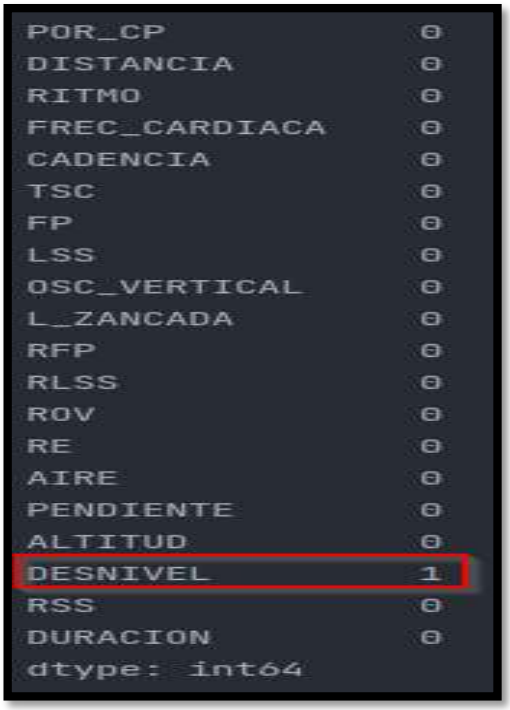
	POR_CP	DISTANCIA	RITMO	...	DESNIVEL	RSS	DURACION
count	1229.000	1229.000	1229.000	...	1228.000	1229.000	1229.000
mean	81.950	1517.453	501.777	...	8.610	11.627	771.563
std	14.843	1359.908	59.279	...	8.532	11.190	686.143
min	65.137	50.000	209.000	...	0.000	1.000	19.000
25%	74.770	800.000	489.000	...	4.000	6.000	411.000
50%	77.981	1000.000	513.000	...	6.000	8.000	520.000
75%	83.944	1500.000	535.000	...	10.000	13.000	817.000
max	178.899	7600.000	637.000	...	77.000	103.000	3599.000

Conclusiones: a priori y, como ejemplo, podemos observar una media (mean) de 1517.453 en la característica “DISTANCIA” con un valor mínimo (min) de 50 y máximo (max) de 7600 diferencias muy grandes en valores absolutos por lo que deberemos tener en cuenta que no se haya ”colado” un valor extremo que puerta “meter” ruido en nuestro modelo predictivo (fig. 3.1).

Búsqueda de valores nulos o faltantes

Mediante data.isnull().sum() haremos un conteo de aquellos registros que vengas informados a ***null*** o ***vacíos***, con el fin de poder solucionarlo antes de las visualización de los datos. Y como podemos observar se nos ha “colado” en la característica o atributo “DESNIVEL “, un registro, por lo que lo rellenaremos con la media de los mismos.

```
# Buscamos datos nulos o faltantes
print(data.isnull().sum())
# Completamos los datos nulos con la media de cada uno
data['DESNIVEL']        = data['DESNIVEL'].fillna(data['DESNIVEL'].median())
# Buscaremos si hay algún valor nulo o vacío en nuestro DataFrame (data)
data.isnull().sum()
```



```
POR_CP           0
DISTANCIA        0
RITMO            0
FREC_CARDIACA    0
CADENCIA         0
TSC              0
FP               0
LSS              0
OSC_VERTICAL     0
L_ZANCADA        0
RFP              0
RLSS             0
ROV              0
RE               0
AIRE             0
PENDIENTE        0
ALTITUD          0
DESNIVEL         1
RSS              0
DURACION         0
dtype: int64
```

```
POR_CP           0
DISTANCIA        0
RITMO            0
FREC_CARDIACA    0
CADENCIA         0
TSC              0
FP               0
LSS              0
OSC_VERTICAL     0
L_ZANCADA        0
RFP              0
RLSS             0
ROV              0
RE               0
AIRE             0
PENDIENTE        0
ALTITUD          0
DESNIVEL         0
RSS              0
DURACION         0
dtype: int64
```

Código completo

```
# ********************************************************************************
# ***************              LIBRERÍAS A IMPORTAR              ***************
# ********************************************************************************
import pandas                       as pd
import matplotlib.pyplot            as plt
import pandas                       as pd
import numpy                        as np
import seaborn                      as sns

# ********************************************************************************
# ***************           CARGAMOS NUESTO DATAFRAME            ***************
# ********************************************************************************
# Cargamos los datos contenidos en "SEGMENTOS_csv.csv"
data = pd.read_csv('SEGMENTOS_csv.csv')
print(data)

# ********************************************************************************
# ***************             ANÁLISIS DE LOS DATOS              ***************
# ********************************************************************************
# Función head()
print(data.head(15))
# Función shape
print(data.shape)
# Función dtaypes
print(data.dtypes)
# Función describe()
print(data.describe())
# Función describe()
pd.set_option('display.width',      100)
pd.set_option('display.precision',  5)
print(data.corr())
# Función skew()
print(data.skew())
# Buscamos datos nulos o faltantes
print(data.isnull().sum())
# Completamos los datos nulos con la media de cada uno
data['DESNIVEL']        = data['DESNIVEL'].fillna(data['DESNIVEL'].median())
# Buscaremos si hay algún valor nulo o vacío en nuestro DataFrame (data)
print(data.isnull().sum())
```

VISUALIZACIÓN

Preámbulo

Lo primero que debemos realizar a la hora de trabajar con machine learning es visualizar nuestros datos para conocer su comportamiento y distribución. Esta primera observación de datos posibilita aprender más sobre ellos siendo la forma más rápida y útil de conocer qué técnicas son las más adecuadas en pre y pos procesamiento. En este sentido en esta tercera sección trabajaremos:

- Cómo crear gráficos para entender cada atributo de manera independiente.
- Cómo crear gráficos para entender las relaciones entre los diferentes atributos.

Los gráficos de las relaciones entre los atributos pueden darnos una idea de los atributos que pueden ser redundantes, los métodos de remuestreo que pueden ser necesarios y, en última instancia, la dificultad de un problema de predicción. Para ello, la fase de visualización puede dividirse en las siguientes partes:

- ***Visualización univariable:*** cuando queremos visualizar un atributo de manera independiente a los demás.
- ***Visualización multivariable***: cuando queremos visualizar la interacción entre los diferentes atributos de nuestro conjunto de datos.

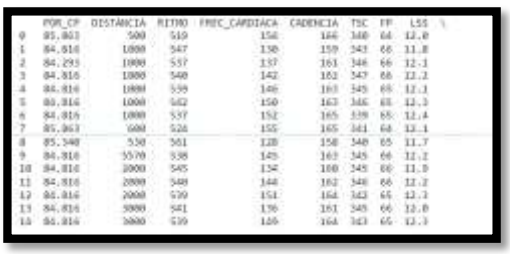
Importar librerías necesarias

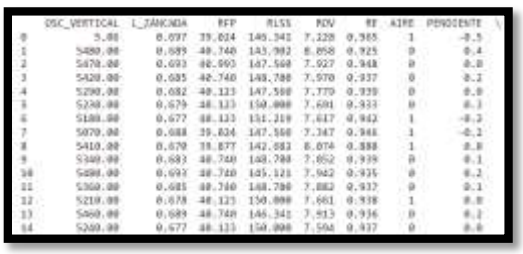
```
# ********************************************************************************
# ***************                LIBRERÍAS A IMPORTAR                ***************
# ********************************************************************************
import pandas                           as pd
import matplotlib.pyplot                as plt
import pandas                           as pd
import numpy                            as np
import seaborn                          as sns
```

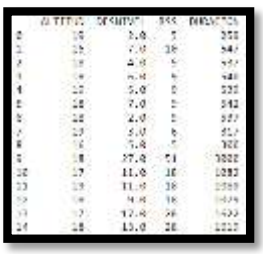
Cargar el conjunto de datos

Para esta práctica vamos a cargar el conjunto de datos de nuestro proyecto "SEGMENTOS_csv.csv" para hacer observaciones con las funciones que nos permitan hacer diferentes tipos de visualizaciones. Además y, conocedores que en "DESNIVEL" hay un registro vacío, procederemos a su resolución para un mejor tratamiento.

```
# ********************************************************************************
# ***************            CARGAMOS NUESTO DATAFRAME           ***************
# ********************************************************************************
# Cargamos los datos contenidos en "SEGMENTOS_csv.csv"
data = pd.read_csv('SEGMENTOS_csv.csv')
# Completamos los datos nulos con la media de cada uno
data['DESNIVEL']        = data['DESNIVEL'].fillna(data['DESNIVEL'].median())
```







Visualización Univariable

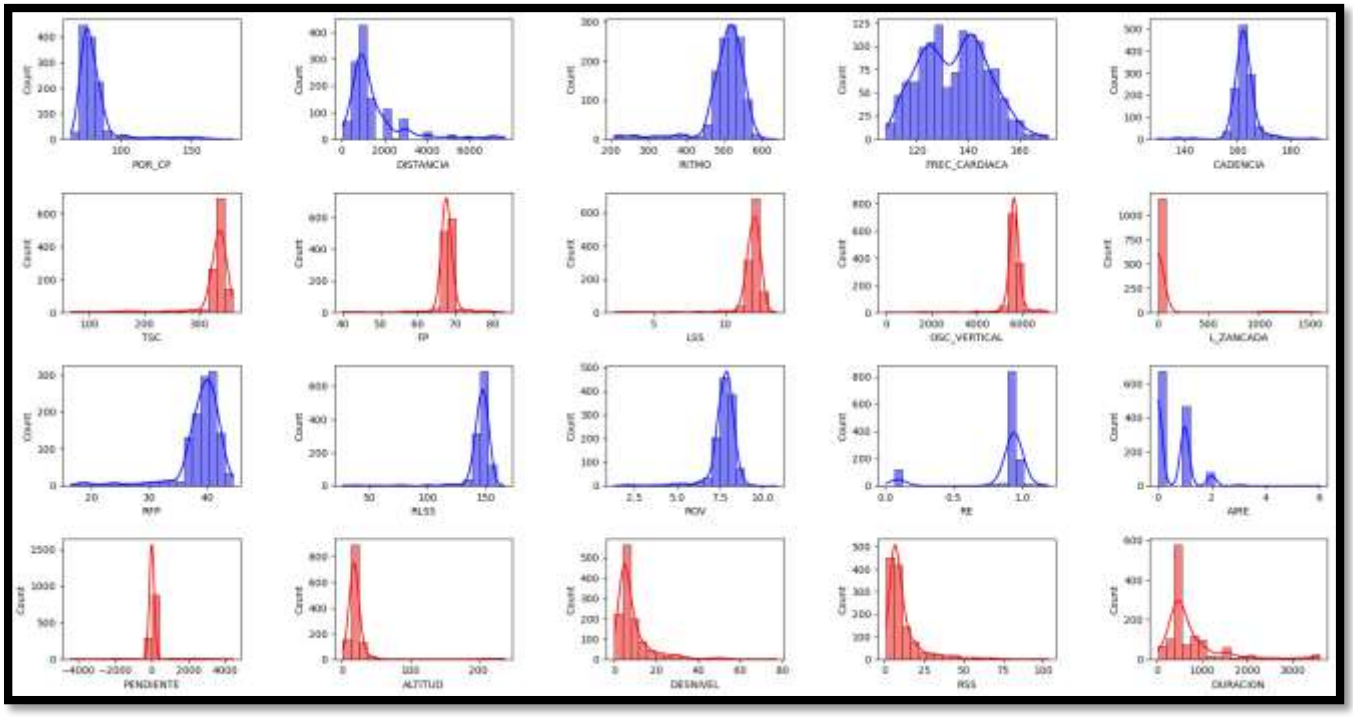
Como se ha comentado anteriormente, las gráficas univariable nos permiten visualizar los atributos individuales sin interacciones; las cuales, el objetivo principal de las mismas es aprender algo sobre la distribución, la tendencia y la propagación de cada atributo.

A continuación se describen las más relevantes.

Histogramas

A partir de la forma de los contenedores, puede tener una idea rápida de si un atributo es gaussiano, sesgado o incluso tiene una distribución exponencial. También puede ayudarlo a ver posibles valores atípicos, por lo que tanto Matplotlib como Seabornpueden ser potentes librerías de visualización de datos.

```
# ********************************************************************************
# ******************              VISUALIZACIONES              ******************
# ********************************************************************************
#   HISTOGRAMAS O DISTRIBUCIÓN CON DENSIDAD
f, axes = plt.subplots(4,5, figsize =(11.7,8.27))
sns.histplot(data["POR_CP"],                ax= axes [0,0],kde = True, bins = 20, color="Blue", fill=True)
sns.histplot(data["DISTANCIA"],             ax= axes [0,1],kde = True, bins = 20, color="Blue", fill=True)
sns.histplot(data["RITMO"],                 ax= axes [0,2],kde = True, bins = 20, color="Blue", fill=True)
sns.histplot(data["FREC_CARDIACA"],         ax= axes [0,3],kde = True, bins = 20, color="Blue", fill=True)
sns.histplot(data["CADENCIA"],              ax= axes [0,4],kde = True, bins = 20, color="Blue", fill=True)
sns.histplot(data["TSC"],                   ax= axes [1,0],kde = True, bins = 20, color="Red", fill=True)
sns.histplot(data["FP"],                    ax= axes [1,1],kde = True, bins = 20, color="Red", fill=True)
sns.histplot(data["LSS"],                   ax= axes [1,2],kde = True, bins = 20, color="Red", fill=True)
sns.histplot(data["OSC_VERTICAL"],          ax= axes [1,3],kde = True, bins = 20, color="Red", fill=True)
sns.histplot(data["L_ZANCADA"],             ax= axes [1,4],kde = True, bins = 20, color="Red", fill=True)
sns.histplot(data["RFP"],                   ax= axes [2,0],kde = True, bins = 20, color="Blue", fill=True)
sns.histplot(data["RLSS"],                  ax= axes [2,1],kde = True, bins = 20, color="Blue", fill=True)
sns.histplot(data["ROV"],                   ax= axes [2,2],kde = True, bins = 20, color="Blue", fill=True)
sns.histplot(data["RE"],                    ax= axes [2,3],kde = True, bins = 20, color="Blue", fill=True)
sns.histplot(data["AIRE"],                  ax= axes [2,4],kde = True, bins = 20, color="Blue", fill=True)
sns.histplot(data["PENDIENTE"],             ax= axes [3,0],kde = True, bins = 20, color="Red", fill=True)
sns.histplot(data["ALTITUD"],               ax= axes [3,1],kde = True, bins = 20, color="Red", fill=True)
sns.histplot(data["DESNIVEL"],              ax= axes [3,2],kde = True, bins = 20, color="Red", fill=True)
sns.histplot(data["RSS"],                   ax= axes [3,3],kde = True, bins = 20, color="Red", fill=True)
sns.histplot(data["DURACION"],              ax= axes [3,4],kde = True, bins = 20, color="Red", fill=True)
plt.tight_layout()
plt.show()
```



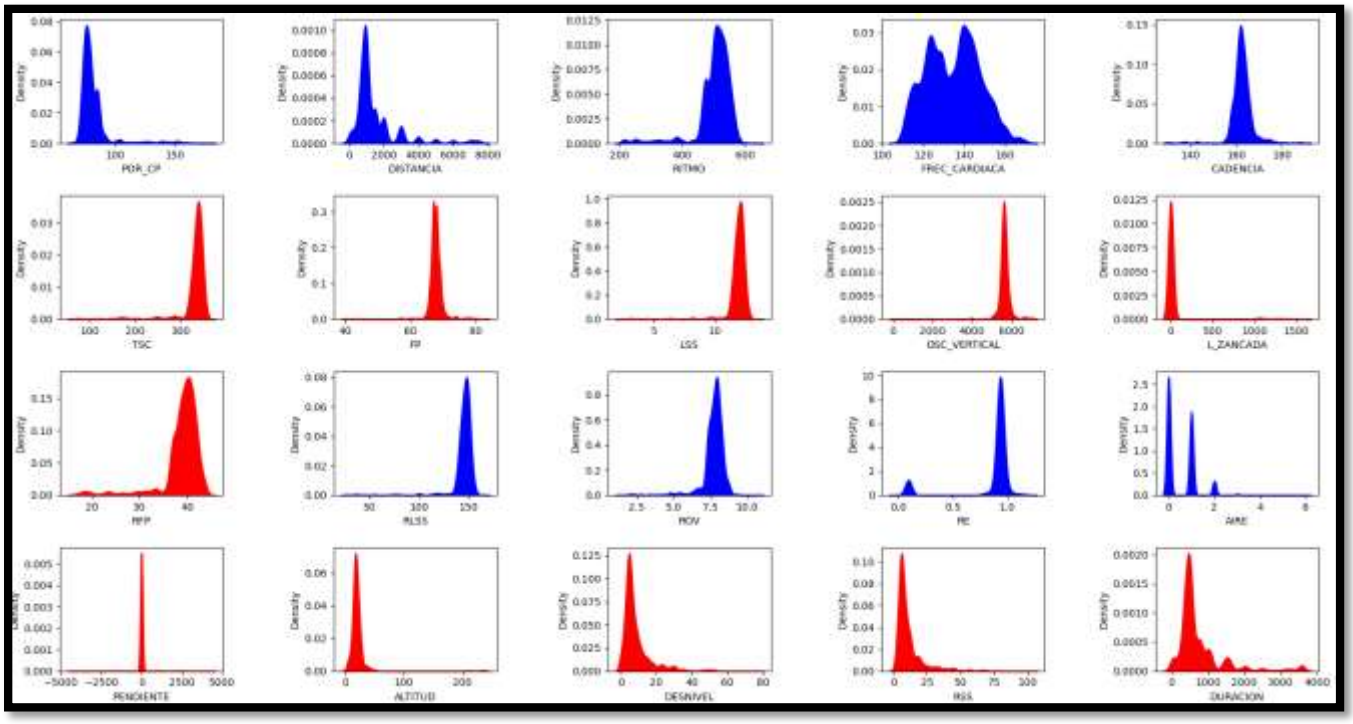
Siempre, a priori, y visualmente hablando, podemos observar que la mayoría de los atributos suelen tener una distribución casi gaussiana o normal (algunas simples, otras con doble campana FREC_CARDIACA y REC y otra triple campana AIRE)y aparentemente alguna exponencial L_ZANCADA. También podemos observar la existencia de sesgo en casi todas las distribucione siendo este menor o casi inexistente en FREC_CARDIACA y CADENCIA. Esto es interesante porque muchas técnicas de aprendizaje automático suponen una distribución univariada gaussiana en las variables de entrada.

Densidad

Las gráficas se ven como un histograma abstracto con una curva suave dibujada a través de la parte superior de cada contenedor, al igual que su ojo intentó hacer con los histogramas. Podemos ver que la distribución de cada atributo es más clara que los histogramas

```
# GRAFICOS DE DENSIDAD
f, axes = plt.subplots(4,5, figsize =(11.7,8.27))
sns.kdeplot(data["POR_CP"],          ax = axes [0,0], shade = True, color = "Blue", fill = True,
                                     bw_adjust=.5, clip_on=False, alpha=1, linewidth=1.5)

sns.kdeplot(data["DISTANCIA"],       ax = axes [0,1], shade = True, color = "Blue", fill = True,
                                     bw_adjust=.5, clip_on=False, alpha=1, linewidth=1.5)

sns.kdeplot(data["RITMO"],           ax = axes [0,2], shade = True, color = "Blue", fill = True,
                                     bw_adjust=.5, clip_on=False, alpha=1, linewidth=1.5)

sns.kdeplot(data["FREC_CARDIACA"],   ax = axes [0,3], shade = True, color = "Blue", fill = True,
                                     bw_adjust=.5, clip_on=False, alpha=1, linewidth=1.5)

sns.kdeplot(data["CADENCIA"],        ax = axes [0,4], shade = True, color = "Blue", fill = True,
                                     bw_adjust=.5, clip_on=False, alpha=1, linewidth=1.5)

sns.kdeplot(data["TSC"],             ax = axes [1,0], shade = True, color = "Red", fill = True,
                                     bw_adjust=.5, clip_on=False, alpha=1, linewidth=1.5)

sns.kdeplot(data["FP"],              ax = axes [1,1], shade = True, color = "Red", fill = True,
                                     bw_adjust=.5, clip_on=False, alpha=1, linewidth=1.5)

sns.kdeplot(data["LSS"],             ax = axes [1,2], shade = True, color = "Red", fill = True,
                                     bw_adjust=.5, clip_on=False, alpha=1, linewidth=1.5)

sns.kdeplot(data["OSC_VERTICAL"],    ax = axes [1,3], shade = True, color = "Red", fill = True,
                                     bw_adjust=.5, clip_on=False, alpha=1, linewidth=1.5)

sns.kdeplot(data["L_ZANCADA"],       ax = axes [1,4], shade = True, color = "Red", fill = True,
                                     bw_adjust=.5, clip_on=False, alpha=1, linewidth=1.5)

sns.kdeplot(data["RFP"],             ax = axes [2,0], shade = True, color = "Red", fill = True,
                                     bw_adjust=.5, clip_on=False, alpha=1, linewidth=1.5)

sns.kdeplot(data["RLSS"],            ax = axes [2,1], shade = True, color = "Blue", fill = True,
                                     bw_adjust=.5, clip_on=False, alpha=1, linewidth=1.5)

sns.kdeplot(data["ROV"],             ax = axes [2,2], shade = True, color = "Blue", fill = True,
                                     bw_adjust=.5, clip_on=False, alpha=1, linewidth=1.5)

sns.kdeplot(data["RE"],              ax = axes [2,3], shade = True, color = "Blue", fill = True,
                                     bw_adjust=.5, clip_on=False, alpha=1, linewidth=1.5)

sns.kdeplot(data["AIRE"],            ax = axes [2,4], shade = True, color = "Blue", fill = True,
                                     bw_adjust=.5, clip_on=False, alpha=1, linewidth=1.5)

sns.kdeplot(data["PENDIENTE"],       ax = axes [3,0], shade = True, color = "Red", fill = True,
                                     bw_adjust=.5, clip_on=False, alpha=1, linewidth=1.5)

sns.kdeplot(data["ALTITUD"],         ax = axes [3,1], shade = True, color = "Red", fill = True,
                                     bw_adjust=.5, clip_on=False, alpha=1, linewidth=1.5)

sns.kdeplot(data["DESNIVEL"],        ax = axes [3,2], shade = True, color = "Red", fill = True,
                                     bw_adjust=.5, clip_on=False, alpha=1, linewidth=1.5)

sns.kdeplot(data["RSS"],             ax = axes [3,3], shade = True, color = "Red", fill = True,
                                     bw_adjust=.5, clip_on=False, alpha=1, linewidth=1.5)

sns.kdeplot(data["DURACION"],        ax = axes [3,4], shade = True, color = "Red", fill = True,
                                     bw_adjust=.5, clip_on=False, alpha=1, linewidth=1.5)
```

```
plt.tight_layout()
plt.show()
```

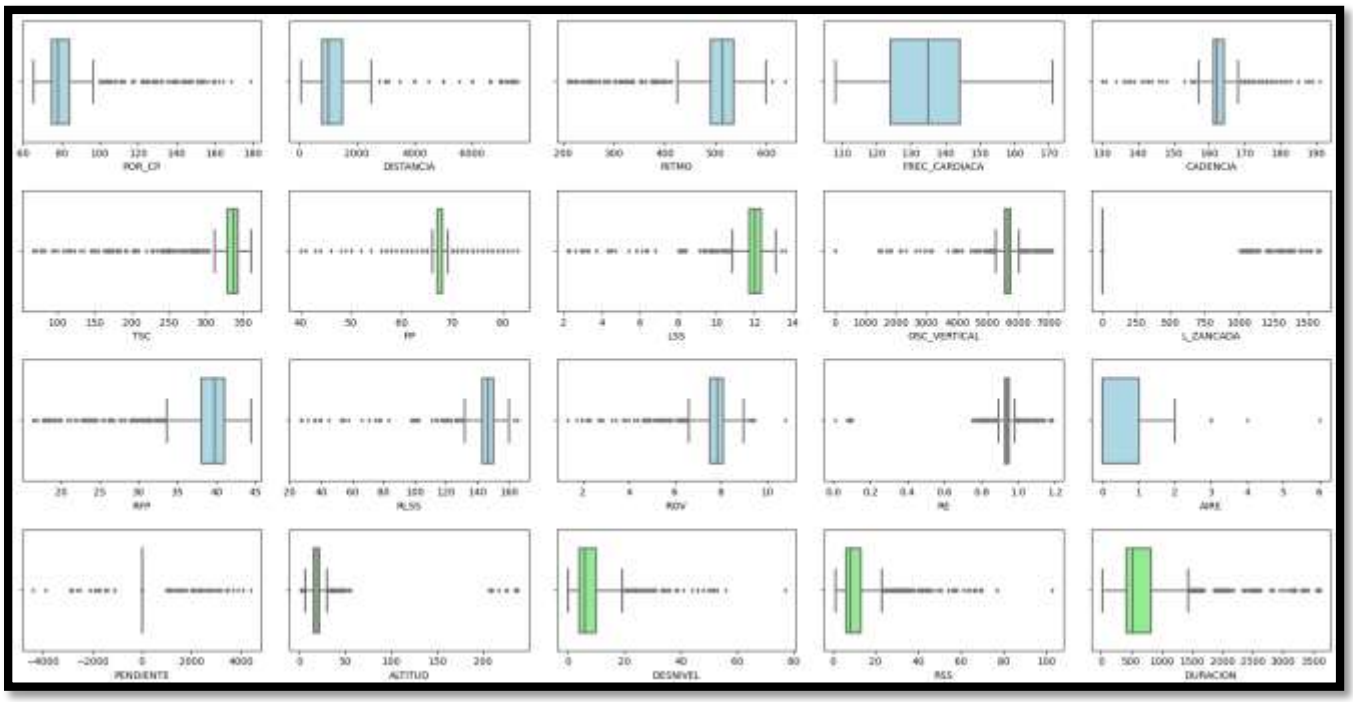


Boxplots

Podemos ver que la extensión de los atributos es bastante diferente. Algunos como la FREC_CARDIACA, L_ZANCADA y RFP parecen bastante sesgados hacia valores más pequeños.

```
# GRAFICOS BOXPLOTS
f, axes = plt.subplots(4,5, figsize =(11.7,8.27))
sns.boxplot(x = data["POR_CP"],          ax = axes [0,0], orient = "h", color = "lightblue", saturation= 1,
                                         width = 0.7, dodge = True, fliersize = 3, linewidth = 2)

sns.boxplot(x = data["DISTANCIA"],       ax = axes [0,1], orient = "h", color = "lightblue", saturation= 1,
                                         width = 0.7, dodge = True, fliersize = 3, linewidth = 2)

sns.boxplot(x = data["RITMO"],           ax = axes [0,2], orient = "h", color = "lightblue", saturation= 1,
                                         width = 0.7, dodge = True, fliersize = 3, linewidth = 2)

sns.boxplot(x = data["FREC_CARDIACA"],   ax = axes [0,3], orient = "h", color = "lightblue", saturation= 1,
                                         width = 0.7, dodge = True, fliersize = 3, linewidth = 2)

sns.boxplot(x = data["CADENCIA"],        ax = axes [0,4], orient = "h", color = "lightblue", saturation= 1,
                                         width = 0.7, dodge = True, fliersize = 3, linewidth = 2)

sns.boxplot(x = data["TSC"],             ax = axes [1,0], orient = "h", color = "lightgreen",saturation= 1,
                                         width = 0.7, dodge = True, fliersize = 3, linewidth = 2)

sns.boxplot(x = data["FP"],              ax = axes [1,1], orient = "h", color = "lightgreen",saturation= 1,
                                         width = 0.7, dodge = True, fliersize = 3, linewidth = 2)

sns.boxplot(x = data["LSS"],             ax = axes [1,2], orient = "h", color = "lightgreen",saturation= 1,
                                         width = 0.7, dodge = True, fliersize = 3, linewidth = 2)

sns.boxplot(x = data["OSC_VERTICAL"],    ax = axes [1,3], orient = "h", color = "lightgreen",saturation= 1,
                                         width = 0.7, dodge = True, fliersize = 3, linewidth = 2)

sns.boxplot(x = data["L_ZANCADA"],       ax = axes [1,4], orient = "h", color = "lightgreen",saturation= 1,
                                         width = 0.7, dodge = True, fliersize = 3, linewidth = 2)

sns.boxplot(x = data["RFP"],             ax = axes [2,0], orient = "h", color = "lightblue",saturation= 1,
                                         width = 0.7, dodge = True, fliersize = 3, linewidth = 2)

sns.boxplot(x = data["RLSS"],            ax = axes [2,1], orient = "h", color = "lightblue",saturation = 1,
                                         width = 0.7, dodge = True, fliersize = 3, linewidth = 2)
```

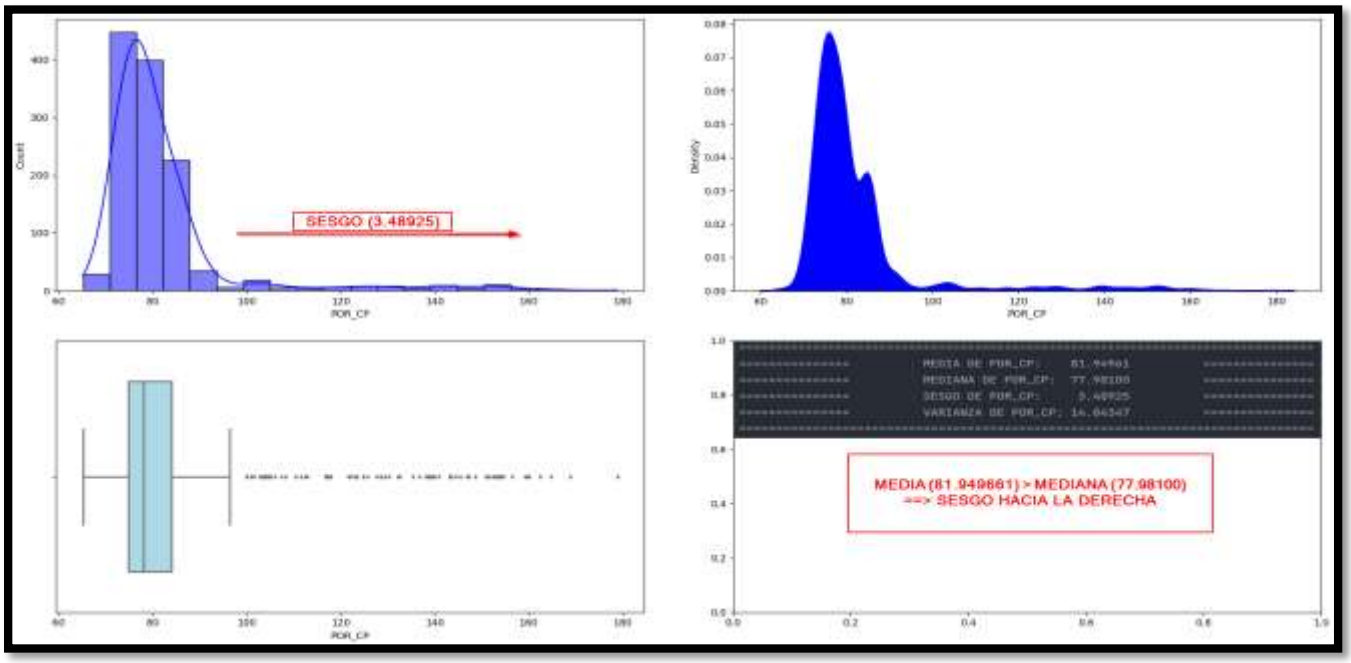
```
sns.boxplot(x = data["ROV"],            ax = axes [2,2], orient = "h", color = "lightblue",saturation = 1,
                                        width = 0.7, dodge = True, fliersize = 3, linewidth = 2)

sns.boxplot(x = data["RE"],             ax = axes [2,3], orient = "h", color = "lightblue",saturation = 1,
                                        width = 0.7, dodge = True, fliersize = 3, linewidth = 2)

sns.boxplot(x = data["AIRE"],           ax = axes [2,4], orient = "h", color = "lightblue", aturation = 1,
                                        width = 0.7, dodge = True, fliersize = 3, linewidth = 2)

sns.boxplot(x = data["PENDIENTE"],      ax = axes [3,0], orient = "h", color = "lightgreen",saturation= 1,
                                        width = 0.7, dodge = True, fliersize = 3, linewidth = 2)

sns.boxplot(x = data["ALTITUD"],        ax = axes [3,1], orient = "h", color = "lightgreen",saturation= 1,
                                        width = 0.7, dodge = True, fliersize = 3, linewidth = 2)

sns.boxplot(x = data["DESNIVEL"],       ax = axes [3,2], orient = "h", color = "lightgreen",saturation= 1,
                                        width = 0.7, dodge = True, fliersize = 3, linewidth = 2)

sns.boxplot(x = data["RSS"],            ax = axes [3,3], orient = "h", color = "lightgreen",saturation= 1,
                                        width = 0.7, dodge = True, fliersize = 3, linewidth = 2)

sns.boxplot(x = data["DURACION"],       ax = axes [3,4], orient = "h", color = "lightgreen",saturation= 1,
                                        width = 0.7, dodge = True, fliersize = 3, linewidth = 2)

plt.tight_layout()
plt.show()
```



Estudio Visualización "POR_CP"

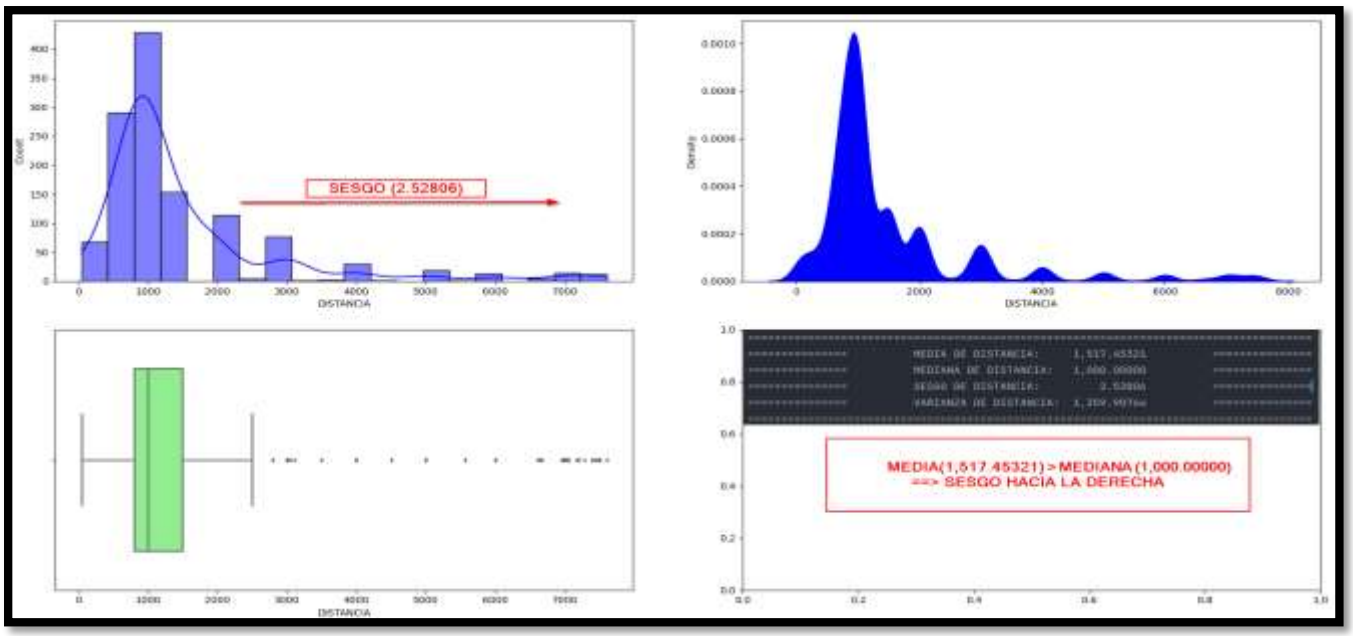
```
# Estudio Visualización "POR_CP"
# Métricas POR_CP
print("                ================================================================================")
print(f"                ===============          MEDIANA DE POR_CP:  {data['POR_CP'].median():,.5f}
===============")
print(f"                ===============          MEDIA DE POR_CP:    {data['POR_CP'].mean():,.5f}
===============")
print(f"                ===============          SESGO DE POR_CP:     {data['POR_CP'].skew():,.5f}
===============")
print("                ================================================================================")
f, axes = plt.subplots(2,2, figsize =(11.7,8.27))

sns.histplot(data["POR_CP"],        ax = axes [0,0], kde = True, bins = 20, color="Blue", fill=True)

sns.kdeplot(data["POR_CP"],         ax = axes [0,1], shade = True, color = "Blue", fill = True,
                                    bw_adjust = .5, clip_on = False, alpha=1, linewidth=1.5)

sns.boxplot(x = data["POR_CP"],     ax = axes [1,0], orient = "h", color = "lightblue", saturation = 1,
                                    width = 0.7, dodge = True, fliersize = 3, linewidth = 2)
```

```
plt.tight_layout()
plt.show()
```



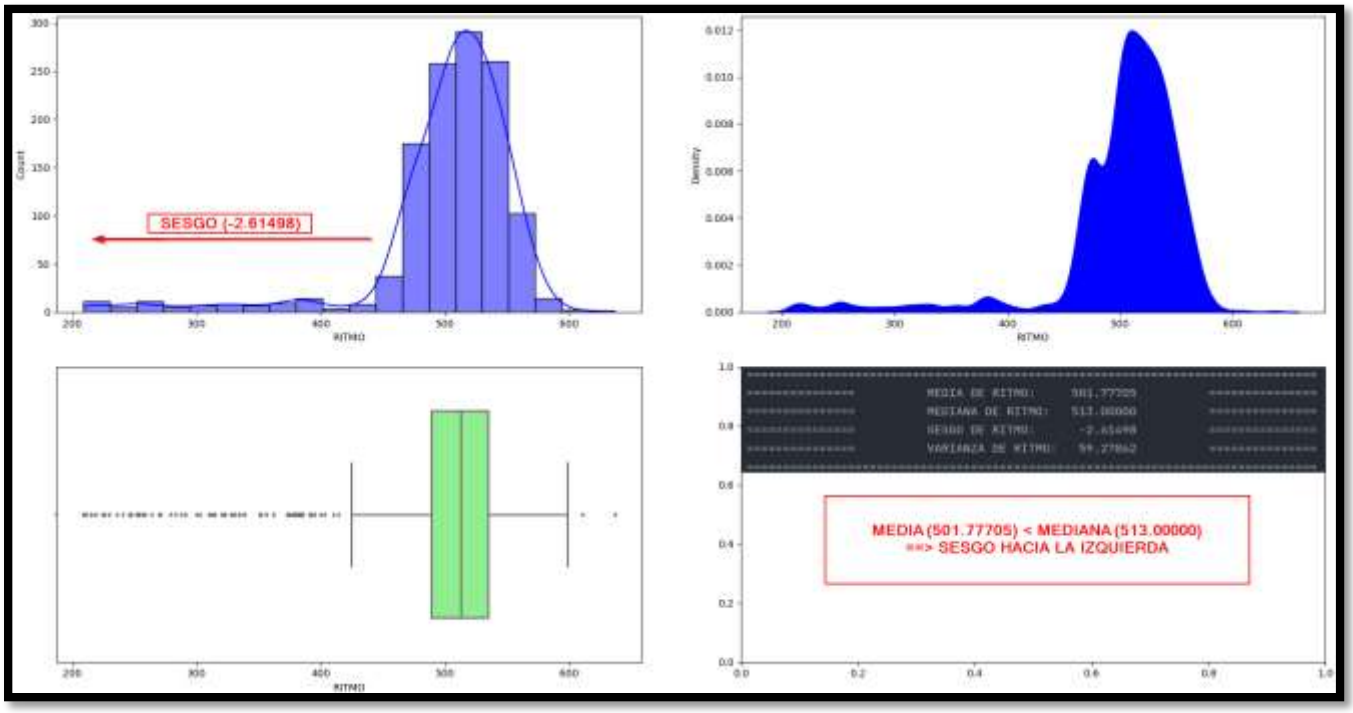
Estudio Visualización "DISTANCIA"

```
# Visualización Histograma, Densidad y Boxplot de DISTANCIA
f, axes = plt.subplots(2,2, figsize =(11.7,8.27))

sns.histplot(data["DISTANCIA"],      ax = axes [0,0], kde = True, bins = 20, color="Blue", fill=True)

sns.kdeplot(data["DISTANCIA"],       ax = axes [0,1], shade = True, color = "Blue", fill = True,
                                        bw_adjust=.5, clip_on=False, alpha=1, linewidth=1.5)

sns.boxplot(x = data["DISTANCIA"],    ax = axes [1,0], orient = "h", color = "lightgreen", saturation = 1,
                                        width = 0.7, dodge = True, fliersize = 3, linewidth = 2)
plt.tight_layout()
plt.show()
```



Estudio Visualización "RITMO"

```
# Estudio Visualización "RITMO"
# Métricas RITMO
print("               ================================================================================")
print(f"               ================          MEDIA DE RITMO:     {data['RITMO'].mean():,.5f}
================")
print(f"               ================          MEDIANA DE RITMO:   {data['RITMO'].median():,.5f}
================")
print(f"               ================          SESGO DE RITMO:      {data['RITMO'].skew():,.5f}
================")
print(f"               ================          VARIANZA DE RITMO:   {data['RITMO'].std():,.5f}
================")
print("               ================================================================================")

# Visualización Histograma, Densidad y Boxplot de RITMO
f, axes = plt.subplots(2,2, figsize =(11.7,8.27))

sns.histplot(data["RITMO"],     ax = axes [0,0], kde = True, bins = 20, color="Blue", fill=True)

sns.kdeplot(data["RITMO"],      ax = axes [0,1], shade = True, color = "Blue", fill = True,
                                bw_adjust=.5, clip_on=False, alpha=1, linewidth=1.5)

sns.boxplot(x = data["RITMO"],   ax = axes [1,0], orient = "h", color = "lightgreen", saturation = 1,
                                width = 0.7, dodge = True, fliersize = 3, linewidth = 2)
plt.tight_layout()
plt.show()
```



Jupyter Notebook

https://modelo-metrica-stryd-machine-learning.s3.eu-west-1.amazonaws.com/python/FASE_Analisis_Datos_Estadistica_Descriptiva.ipynb

FUENTES:

- **Eduard Barceló:** https://www.eduardbarcelo.com/metricas-de-stryd/
- **Correr una Maratón:** https://www.correrunamaraton.com/stryd-medidor-potencia/
- **Palladino Power Project:** https://docs.google.com/document/d/e/2PACX-1vTXIwQE99RiAfVJOdjUIlhZj7_D0k0LHE0U9gDatL1p4TXEVZZ_Rj60S3wczDzgystpcIwOS4kKI6R9/pub?fbclid=IwAR35xriNyEjStmE3fYp7BuESFkIJyF-cwIIvoOGpeDdtLjV39PKqChEFCvo