

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

VISIÓN GENERAL PROYECTO MÉTRICAS STRYD	2
INTRODUCCIÓN.....	2
STRYD.....	2
OTRAS MÉTRICAS.....	4
FUENTE DE DATOS.....	6
PROCESOS DE ETL.....	8
<i>MAPPING: m_SEGMENTOS_TXT_TO_TABLE</i>	8
<i>EJECUCIÓN: m_SEGMENTOS_TXT_TO_TABLE</i>	11
<i>m_SEGMENTOS_TABLE_TO_TIPOOK.....</i>	11
<i>EJECUCIÓN: m_SEGMENTOS_TABLE_TO_TIPOOK.....</i>	14
FASE ANÁLISIS DE LOS DATOS	15
PREÁMBULO FASE DE ANÁLISIS DE DATOS.....	15
ESTADÍSTICA DESCRIPTIVA.....	15
<i>Librerías necesarias</i>	15
<i>Cargamos nuestro DataFrame ()</i>	16
<i>Revisar los datos: head().....</i>	16
<i>Dimensiones de los datos: shape</i>	16
<i>Tipo de datos: dtypes.....</i>	17
<i>Resumen: describe ().....</i>	17
<i>Correlaciones: corr()</i>	19
<i>Asimetría: skew()</i>	19
<i>Algunas métricas a tener en cuenta</i>	20
<i>Búsqueda de valores nulos o faltantes</i>	20
<i>Código completo.....</i>	21
VISUALIZACIÓN	22
<i>Preámbulo</i>	22
<i>Importar librerías necesarias</i>	22
<i>Cargar el conjunto de datos.....</i>	22
<i>Visualización Univariable.....</i>	23
<i>Histogramas</i>	23
<i>Densidad</i>	24
<i>Boxplots</i>	25
ESTUDIO VISUALIZACIÓN "POR_CP"	27
ESTUDIO VISUALIZACIÓN "DISTANCIA"	28
ESTUDIO VISUALIZACIÓN "RITMO"	29
ESTUDIO VISUALIZACIÓN "FREC_CARDIACA"	30
ESTUDIO VISUALIZACIÓN "CADENCIA"	31
ESTUDIO VISUALIZACIÓN "TSC"	32
ESTUDIO VISUALIZACIÓN "FP"	33
ESTUDIO VISUALIZACIÓN "LSS"	34
ESTUDIO VISUALIZACIÓN "OSC_VERTICAL"	35
JUPYTER NOTEBOOK.....	36
FUENTES:.....	36

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 watos (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 (CADERNCIA):** 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 Cardíaca (FREC_CARDIACA):** mide la frecuencia cardíaca 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.

Segmento	Tiempo en mov.	Distancia	Potencia	Ritmo	Cadencia	FC	Potencia vertical	RPV	TCS	RP	OV	Resistencia del aire
1	6:52	757 m	160 W	8:04 /km	158 spm	121 bpm	67 W	0.42	343 ms	12.4 kN/m	5.83 cm	0 %
2	6:44	749 m	163 W	8:59 /km	160 spm	132 bpm	68 W	0.42	342 ms	12.3 kN/m	5.73 cm	0 %
3	8:44	1.00 km	165 W	8:44 /km	160 spm	136 bpm	68 W	0.41	343 ms	11.9 kN/m	5.82 cm	0 %
4	8:44	1.00 km	167 W	8:43 /km	160 spm	144 bpm	68 W	0.41	342 ms	11.9 kN/m	5.74 cm	0 %
5	8:29	999 m	170 W	8:30 /km	160 spm	149 bpm	69 W	0.41	338 ms	11.9 kN/m	5.87 cm	0 %
6	8:33	750 m	165 W	8:44 /km	161 spm	150 bpm	68 W	0.41	341 ms	11.9 kN/m	5.69 cm	0 %
7	8:34	743 m	164 W	8:51 /km	161 spm	152 bpm	67 W	0.41	341 ms	12.1 kN/m	5.83 cm	0 %

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.

FP (FORM POWER)**RFP =** _____**POTENCIA (WATIOS)**

- **RE (Running Efectiveness):** (*lit. Eduard Barceló*) este 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.

Metros / Segundos**RE =** _____ = _____**Distancia / Duracion****Watios / kilogramos****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.

Distancia**L_ZANCADA =** _____**(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.

Oscilación Vertical**ROV** = _____**Longitud de Zancada**

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

LSS (Leg Spring Stiffness)**RLSS =** _____**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

Tabla de segmentos												
Segmento	Tiempo en mov.	Distancia	Potencia	Ritmo	Cadencia	FC	Potencia vertical	RPV	TCS	RP	OV	Resistencia del aire
1	8:52	757 m	180 W	8:04 /km	158 spm	121 bpm	87 W	0.42	343 ms	12.4 kN/m	5.63 cm	0 %
2	8:44	749 m	183 W	8:59 /km	160 spm	132 bpm	88 W	0.42	342 ms	12.3 kN/m	5.73 cm	0 %
3	8:44	1.00 km	185 W	8:44 /km	160 spm	136 bpm	88 W	0.41	343 ms	11.9 kN/m	5.82 cm	0 %
4	8:44	1.00 km	187 W	8:43 /km	160 spm	144 bpm	88 W	0.41	342 ms	11.9 kN/m	5.74 cm	0 %
5	8:29	999 m	170 W	8:30 /km	160 spm	149 bpm	69 W	0.41	338 ms	11.9 kN/m	5.87 cm	0 %
6	8:33	750 m	185 W	8:44 /km	161 spm	150 bpm	88 W	0.41	341 ms	11.9 kN/m	5.69 cm	0 %
7	8:34	743 m	184 W	8:51 /km	161 spm	152 bpm	87 W	0.41	341 ms	12.1 kN/m	5.63 cm	0 %

Predefinidos en aplicación web

Tabla de segmentos												
Segmento	Tiempo en mov.	Distancia	Potencia	Ritmo	Cadencia	FC	Potencia vertical	RPV	TCS	RP	OV	Resistencia del aire
1	8:59	1.00 km	161 W	8:59 /km	158 spm	123 bpm	87 W	0.42	342 ms	12.3 kN/m	5.89 cm	0 %
2	8:50	1.00 km	164 W	8:50 /km	160 spm	134 bpm	88 W	0.41	342 ms	12.2 kN/m	5.75 cm	0 %
3	8:43	1.00 km	166 W	8:43 /km	160 spm	140 bpm	88 W	0.41	343 ms	11.8 kN/m	5.80 cm	0 %
4	8:41	1.00 km	168 W	8:41 /km	160 spm	148 bpm	88 W	0.41	340 ms	12.0 kN/m	5.80 cm	0 %
5	8:34	1.00 km	168 W	8:34 /km	161 spm	150 bpm	88 W	0.41	339 ms	11.9 kN/m	5.75 cm	0 %
6	8:53	998 m	164 W	8:54 /km	161 spm	152 bpm	87 W	0.41	342 ms	12.1 kN/m	5.85 cm	0 %



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

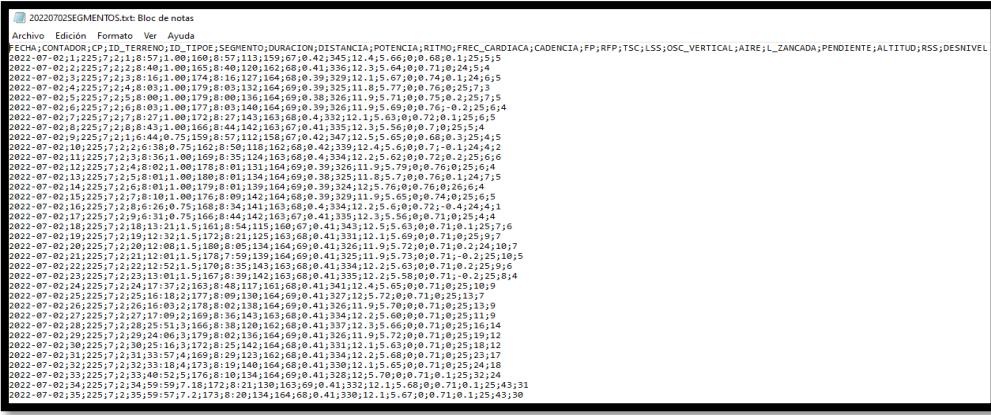
Segmento	Duración	Distancia	Potencia	Ritmo	Frecuencia cardíaca	Cadencia
1	6:52 min	0,76 km	160 W	9:04 /km	121 bpm	158 ppm
2	6:44 min	0,75 km	163 W	8:59 /km	132 bpm	160 ppm
3	8:44 min	1,00 km	165 W	8:44 /km	136 bpm	160 ppm
4	8:44 min	1,00 km	167 W	8:43 /km	144 bpm	160 ppm
5	8:29 min	1,00 km	170 W	8:29 /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	B	C	D	E	F	G	H	I	J	K	L
Segmento,"Duracióñ","Distancia","Potencia","Ritmo","Frecuencia cardíaca","Cadencia","Potencia vertical","Ratio de pasos","Pendiente","Altitud","Estráøs","Desnivel Acum."											
1	Segmento,"Duracióñ","Distancia","Potencia","Ritmo","Frecuencia cardíaca","Cadencia","Potencia vertical","Ratio de pasos","Pendiente","Altitud","Estráøs","Desnivel Acum."										
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	B	C	D	E	F	G	H	I	J	K	L
Segmento,"Duracióñ","Distancia","Potencia","Ritmo","Frecuencia cardíaca","Cadencia","Potencia vertical","Ratio de pasos","Pendiente","Altitud","Estráøs","Desnivel Acum."											
1	Segmento,"Duracióñ","Distancia","Potencia","Ritmo","Frecuencia cardíaca","Cadencia","Potencia vertical","Ratio de pasos","Pendiente","Altitud","Estráøs","Desnivel Acum."										
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

Source Analyzer

K.	Name	Datatype	Length/Pr...
1	FECHA	string	20
2	CONTADOR	string	20
3	CP	string	20
4	ID_TERRENO	string	20
5	ID_TIPOE	string	20
6	SEGMENTO	string	20
7	DURACION	string	20
8	DISTANCIA	string	20
9	POTENCIA	string	20
10	RITMO	string	20
11	FREC_CARDIACA	string	20
12	CADENCIA	string	20
13	FP	string	20
14	RFP	string	20
15	TSC	string	20
16	LSS	string	20
17	OSC_VERTICAL	string	20
18	AIRE	string	20
19	L_ZANCADA	string	20
20	PENDIENTE	string	20
21	ALTITUD	string	20
22	RSS	string	20
23	DESNIVEL	string	20

SOURCE

Table Columns Properties Metadata Extensions

Select table: FlatFile:SEGMENTOS

Column ...	Datatype	Prec	Scale	Not Null	Format	Key Type	Business Name
1 FECHA	string	20	0	□		NOT A KEY	
2 CONTADOR	string	20	0	□		NOT A KEY	
3 CP	string	20	0	□		NOT A KEY	
4 ID_TERRENO	string	20	0	□		NOT A KEY	
5 ID_TIPOE	string	20	0	□		NOT A KEY	
6 SEGMENTO	string	20	0	□		NOT A KEY	
7 DURACION	string	20	0	□		NOT A KEY	
8 DISTANCIA	string	20	0	□		NOT A KEY	
9 POTENCIA	string	20	0	□		NOT A KEY	
10 RITMO	string	20	0	□		NOT A KEY	
11 FREC_CARDIACA	string	20	0	□		NOT A KEY	
12 CADENCIA	string	20	0	□		NOT A KEY	
13 FP	string	20	0	□		NOT A KEY	
14 RFP	string	20	0	□		NOT A KEY	
15 TSC	string	20	0	□		NOT A KEY	
16 LSS	string	20	0	□		NOT A KEY	
17 OSC_VERTICAL	string	20	0	□		NOT A KEY	
18 AIRE	string	20	0	□		NOT A KEY	
19 L_ZANCADA	string	20	0	□		NOT A KEY	
20 PENDIENTE	string	20	0	□		NOT A KEY	
21 ALTITUD	string	20	0	□		NOT A KEY	
22 RSS	string	20	0	□		NOT A KEY	
23 DESNIVEL	string	20	0	□		NOT A KEY	

Target: STG_SEGMENTOS VARCHAR

Target Designer

K.	Name	Datatype	Length...
1	FECHA	varchar	20
2	CONTADOR	varchar	20
3	CP	varchar	20
4	ID_TERRENO	varchar	20
5	ID_TIPOE	varchar	20
6	SEGMENTO	varchar	20
7	DURACION	varchar	20
8	DISTANCIA	varchar	20
9	POTENCIA	varchar	20
10	RITMO	varchar	20
11	FREC_CARDIACA	varchar	20
12	CADENCIA	varchar	20
13	FP	varchar	20
14	RFP	varchar	20
15	TSC	varchar	20
16	LSS	varchar	20
17	OSC_VERTICAL	varchar	20
18	AIRE	varchar	20
19	L_ZANCADA	varchar	20
20	PENDIENTE	varchar	20
21	ALTITUD	varchar	20
22	RSS	varchar	20
23	DESNIVEL	varchar	20

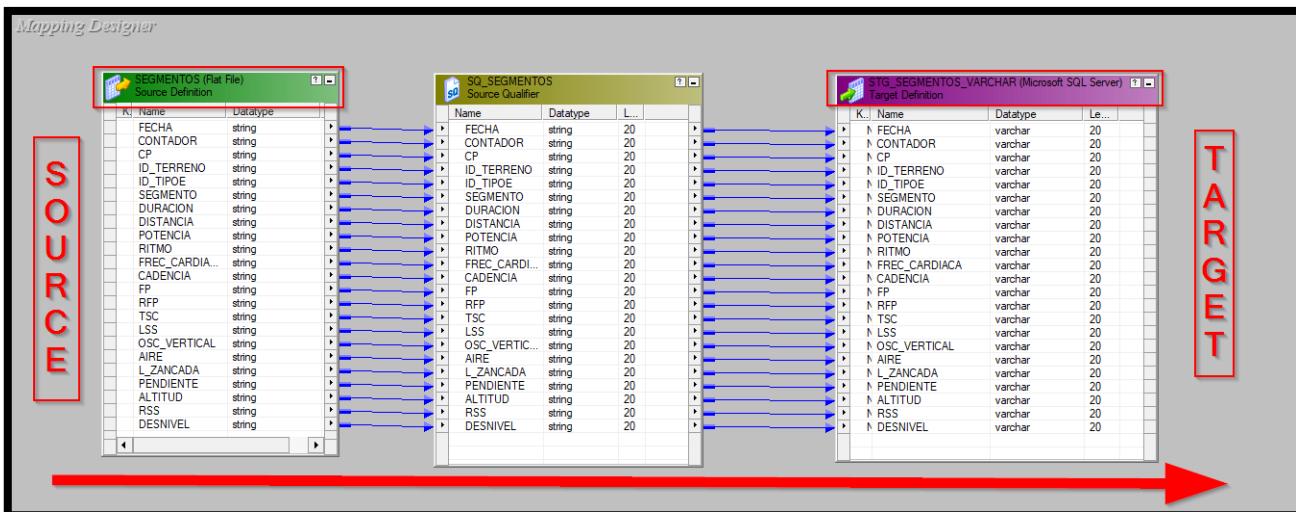
TARGET

Table Columns Indexes Metadata Extensions

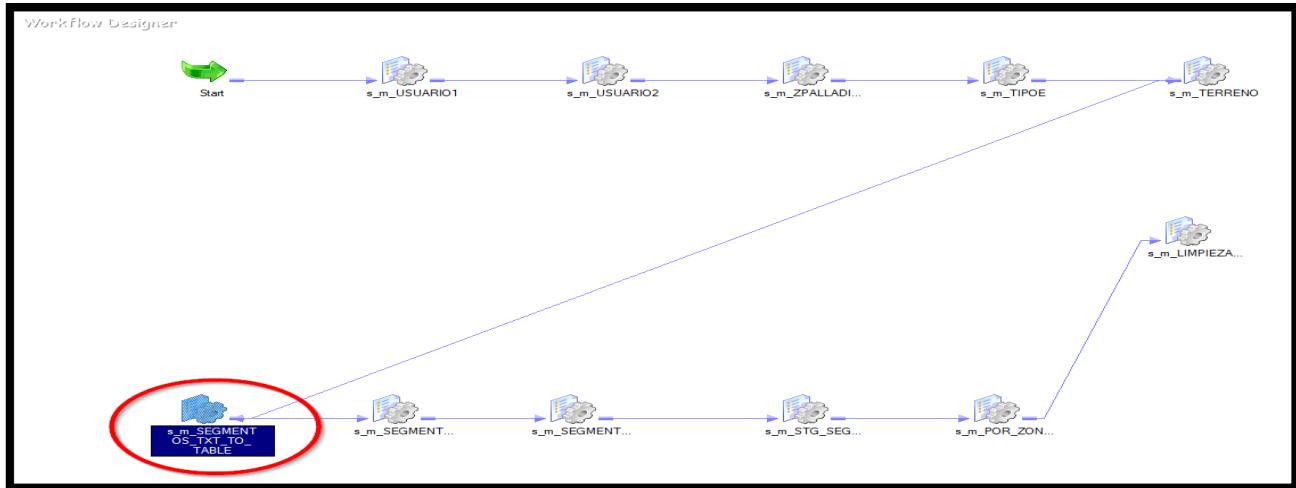
Select table: STG_SEGMENTOS VARCHAR

Column Name	Datatype	Prec	Scale	Not Null	Key Type	Business Name
1 FECHA	varchar	20	0	□	NOT A KEY	
2 CONTADOR	varchar	20	0	□	NOT A KEY	
3 CP	varchar	20	0	□	NOT A KEY	
4 ID_TERRENO	varchar	20	0	□	NOT A KEY	
5 ID_TIPOE	varchar	20	0	□	NOT A KEY	
6 SEGMENTO	varchar	20	0	□	NOT A KEY	
7 DURACION	varchar	20	0	□	NOT A KEY	
8 DISTANCIA	varchar	20	0	□	NOT A KEY	
9 POTENCIA	varchar	20	0	□	NOT A KEY	
10 RITMO	varchar	20	0	□	NOT A KEY	
11 FREC_CARDIACA	varchar	20	0	□	NOT A KEY	
12 CADENCIA	varchar	20	0	□	NOT A KEY	
13 FP	varchar	20	0	□	NOT A KEY	
14 RFP	varchar	20	0	□	NOT A KEY	
15 TSC	varchar	20	0	□	NOT A KEY	
16 LSS	varchar	20	0	□	NOT A KEY	
17 OSC_VERTICAL	varchar	20	0	□	NOT A KEY	
18 AIRE	varchar	20	0	□	NOT A KEY	
19 L_ZANCADA	varchar	20	0	□	NOT A KEY	
20 PENDIENTE	varchar	20	0	□	NOT A KEY	
21 ALTITUD	varchar	20	0	□	NOT A KEY	
22 RSS	varchar	20	0	□	NOT A KEY	
23 DESNIVEL	varchar	20	0	□	NOT A KEY	

Mapping: m_SEGMENTOS_TXT_TO_TABLE



Workflow: wf_ACTUALIZACIONES



Task: s_m_SEGMENTOS_TXT_TO_TABLE

Left Screenshot (Readers Tab):

- Connections:** None
- Properties:**
 - Attribute: SQ_SEGMENTOS - Source Qualifier, Value: Normal
 - Attribute: SEGMENTOS - File Reader, Value: File
 - Input Type: File
 - Concurrent read partitioning: Optimize throughput
 - Source file type: Direct
 - Source file directory: \${PMSSourceFileDir}DM_SEGMENTOS
 - Source filename: SEGMENTOS.txt
 - For Reader To Create Schema: No

Right Screenshot (Writers Tab):

- Connections:** PROYECTO
- Properties:**
 - Attribute: STG_SEGMENTOS_VARCHAR - Relational Writer, Value: Normal
 - Target load type: Insert (checked)
 - Update as Update (checked)
 - Update as Insert (unchecked)
 - Delete (checked)
 - Truncate target table option (checked)

EJECUCIÓN: m_SEGMENTOS_TXT_TO_TABLE

Informatica PowerCenter Workflow Monitor

Workflow Run

	Start Time	Completion Time	Status
wf_ACTUALIZACIONES	27/08/2022 11:15:18	In progress	Running
wf_ACTUALIZACIONES	27/08/2022 11:15:18	27/08/2022 11:15:38	Succeeded
s_m_USUARIO1	27/08/2022 11:15:41	27/08/2022 11:15:58	Succeeded
s_m_USUARIO2	27/08/2022 11:15:59	27/08/2022 11:16:18	Succeeded
s_m_ZPALLADINO	27/08/2022 11:16:18	27/08/2022 11:16:35	Succeeded
s_m_TIPOE	27/08/2022 11:16:36	27/08/2022 11:16:53	Succeeded
s_m_TERRENO	27/08/2022 11:16:54	27/08/2022 11:17:13	Succeeded
s_m_SEGMENTOS_TXT_TO_TABLE	27/08/2022 11:16:54	27/08/2022 11:17:13	Succeeded
s_m_SFREMFNTTDS_TARI_F_TO_TI	27/08/2022 11:17:14	27/08/2022 11:17:46	Completed

s_m_SEGMENTOS_TXT_TO_TABLE [27/08/2022 11:16:54]

Task Details

Attribute Name	Attribute Value
Instance Name	s_m_SEGMENTOS_TXT_TO_TABLE
Task Type	Session
Integration Service Name	DATAMART_SEGMENTOS_SERVICIO
Node(s)	node01_DESKTOP-8M9PCQ2
Start Time	27/08/2022 11:16:54
End Time	27/08/2022 11:17:13
Recovery Time(s)	
Status	Succeeded
Status Message	
Deleted	No
Version Number	1
Mapping Name	m_SEGMENTOS_TXT_TO_TABLE
Source Success Rows	1229
Source Failed Rows	0
Target Success Rows	1229
Target Failed Rows	0
Total Transformation Errors	0

Source/Target Statistics

Transformation Name	Node	Applied Rows	Affected Rows	Rejected Rows	Throughput (Rows/Sec)
s_m_SEGMENTOS	node01_DESKTOP-8M9PCQ2	1229	1229	0	1229
STG_SEGMENTOS...	node01_DESKTOP-8M9PCQ2	1229	1229	0	1229

Object Explorer

SQLQuery1.sql - D...ELEMENTOS (sa (80))

```
USE DATAMART_SEGMENTOS
GO
select * from STG_SEGMENTOS_VARCHAR
```

Results

FECHA	CONTADOR	CP	ID_TERRENO	ID_TIPOE	SEGMENTO	DURACION	DISTANCIA	POTENCIA	RITMO	FREC_CARDIACA	CADENCIA	FP	RPP	TS	
1224	2022-07-02	30	225	7	2	30	25.16	3	172	8.25	142	164	68	0.41	33
1225	2022-07-02	31	225	7	2	31	33.57	4	169	8.29	123	162	68	0.41	33
1226	2022-07-02	32	225	7	2	32	33.18	4	173	8.19	140	164	68	0.41	33
1227	2022-07-02	33	225	7	2	33	40.52	5	176	8.10	134	164	69	0.41	33
1228	2022-07-02	34	225	7	2	34	59.59	7.18	172	8.21	130	163	69	0.41	33
1229	2022-07-02	35	225	7	2	35	59.57	7.2	173	8.20	134	164	68	0.41	33

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 trabajar. Como podremos observar deberemos hacer algunas transformaciones mecánicas.

Source: STG_SEGMENTOS_VARCHAR

Source Analyzer

SOURCE

K.	Name	Datatype	Length...
1	FECHA	varchar	20
2	CONTADOR	varchar	20
3	CP	varchar	20
4	ID_TERRENO	varchar	20
5	ID_TIPOE	varchar	20
6	SEGMENTO	varchar	20
7	DURACION	varchar	20
8	DISTANCIA	varchar	20
9	POTENCIA	varchar	20
10	RITMO	varchar	20
11	FREC_CARDIACA	varchar	20
12	CADENCIA	varchar	20
13	FP	varchar	20
14	RFP	varchar	20
15	TSC	varchar	20
16	LSS	varchar	20
17	OSC_VERTICAL	varchar	20
18	AIRE	varchar	20
19	L_ZANCADA	varchar	20
20	PENDIENTE	varchar	20
21	ALTITUD	varchar	20
22	RSS	varchar	20
23	DESNIVEL	varchar	20

Table Columns Metadata Extensions

Select table: PROYECTO.STG_SEGMENTOS_VARCHAR

Column ...	Datatype	Prec	Scale	Not Null	Key Type	Business Name
1 FECHA	varchar	20	0	✓	NOT A KEY	
2 CONTADOR	varchar	20	0	✓	NOT A KEY	
3 CP	varchar	20	0	✓	NOT A KEY	
4 ID_TERRE...	varchar	20	0	✓	NOT A KEY	
5 ID_TIPOE	varchar	20	0	✓	NOT A KEY	
6 SEGMENTO	varchar	20	0	✓	NOT A KEY	
7 DURACION	varchar	20	0	✓	NOT A KEY	
8 DISTANCIA	varchar	20	0	✓	NOT A KEY	
9 POTENCIA	varchar	20	0	✓	NOT A KEY	
10 RITMO	varchar	20	0	✓	NOT A KEY	
11 FREC_CA...	varchar	20	0	✓	NOT A KEY	
12 CADENCIA	varchar	20	0	✓	NOT A KEY	
13 FP	varchar	20	0	✓	NOT A KEY	
14 RFP	varchar	20	0	✓	NOT A KEY	
15 TSC	varchar	20	0	✓	NOT A KEY	
16 LSS	varchar	20	0	✓	NOT A KEY	
17 OSC_VER...	varchar	20	0	✓	NOT A KEY	
18 AIRE	varchar	20	0	✓	NOT A KEY	
19 L_ZANCADA	varchar	20	0	✓	NOT A KEY	
20 PENDIENTE	varchar	20	0	✓	NOT A KEY	
21 ALTITUD	varchar	20	0	✓	NOT A KEY	
22 RSS	varchar	20	0	✓	NOT A KEY	
23 DESNIVEL	varchar	20	0	✓	NOT A KEY	

Target: STG_SEGMENTOS_TIPOOK

Target Designer

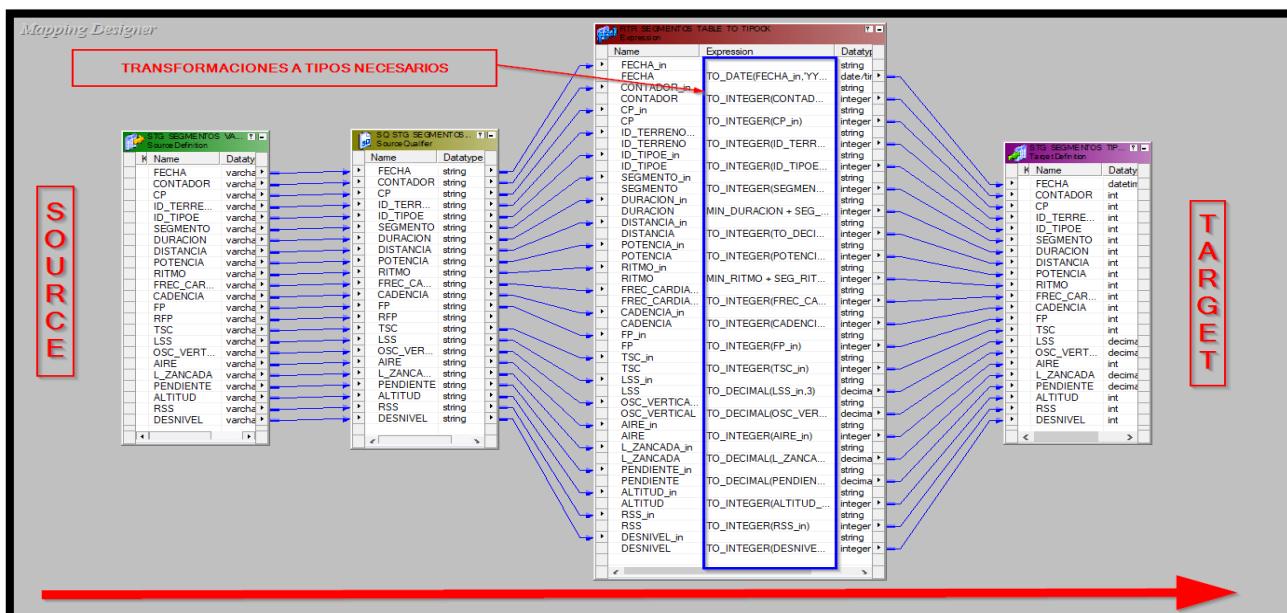
TARGET

K.	Name	Datatype	Length...
1	FECHA	datetime	23
2	CONTADOR	int	10
3	CP	int	10
4	ID_TERRENO	int	10
5	ID_TIPOE	int	10
6	SEGMENTO	int	10
7	DURACION	int	10
8	DISTANCIA	int	10
9	POTENCIA	int	10
10	RITMO	int	10
11	FREC_CARDIACA	decimal	15
12	CADENCIA	decimal	15
13	FP	decimal	15
14	TSC	decimal	15
15	LSS	decimal	15
16	OSC_VERTICAL	decimal	15
17	AIRE	decimal	15
18	L_ZANCADA	decimal	15
19	PENDIENTE	decimal	15
20	ALTITUD	int	10
21	RSS	int	10
22	DESNIVEL	int	10

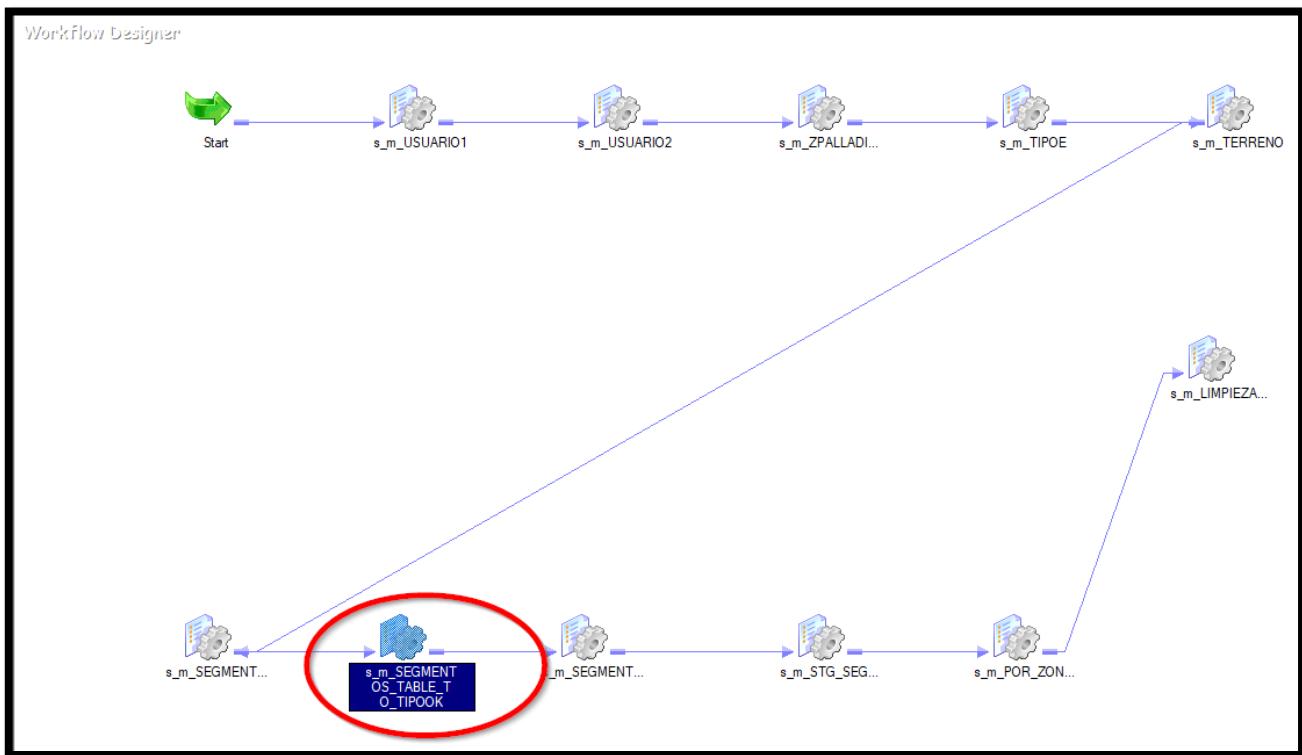
Table Columns Indexes Metadata Extensions

Select table: STG_SEGMENTOS_TIPOOK

Column Name	Datatype	Prec	Scale	Not ...	Key Type	Business Name
1 FECHA	datetime	23	3	✓	NOT A KEY	
2 CONTADOR	int	10	0	✓	NOT A KEY	
3 CP	int	10	0	✓	NOT A KEY	
4 ID_TERRE...	int	10	0	✓	NOT A KEY	
5 ID_TIPOE	int	10	0	✓	NOT A KEY	
6 SEGMENTO	int	10	0	✓	NOT A KEY	
7 DURACION	int	10	0	✓	NOT A KEY	
8 DISTANCIA	int	10	0	✓	NOT A KEY	
9 POTENCIA	int	10	0	✓	NOT A KEY	
10 RITMO	int	10	0	✓	NOT A KEY	
11 FREC_CARDIACA	int	10	0	✓	NOT A KEY	
12 CADENCIA	int	10	0	✓	NOT A KEY	
13 FP	int	10	0	✓	NOT A KEY	
14 TSC	int	10	0	✓	NOT A KEY	
15 LSS	decimal	15	3	✓	NOT A KEY	
16 OSC_VERTICAL	decimal	15	3	✓	NOT A KEY	
17 AIRE	int	10	0	✓	NOT A KEY	
18 L_ZANCADA	decimal	15	3	✓	NOT A KEY	
19 PENDIENTE	decimal	15	3	✓	NOT A KEY	
20 ALTITUD	int	10	0	✓	NOT A KEY	
21 RSS	int	10	0	✓	NOT A KEY	
22 DESNIVEL	int	10	0	✓	NOT A KEY	

Mapping: m_SEGMENTOS_TABLE_TO_TIPOOK

Workflow: wf_ACTUALIZACIONES



Task: s_m_SEGMENTOS_TABLE_TO_TIPOOK

Two side-by-side 'Edit Tasks' windows for the 's_m_SEGMENTOS_TABLE_TO_TIPOOK' task.

Left Window (Writers):

- General tab: Task type: Session, Select task: s_m_SEGMENTOS_TABLE_TO_TIPOOK.
- Properties tab: Shows a table with one row for 'STG_SEGMENTOS_TIPOOK' with 'Relational Writer' selected.
- Connections tab: Shows a connection named 'PROYECTO'.
- Properties tab (bottom): Shows settings for 'STG_SEGMENTOS_TIPOOK - Relational Writer' including 'Target load type: Normal', 'Insert' checked, 'Update as Update' checked, 'Update as Insert' unchecked, 'Update else Insert' unchecked, and 'Delete' checked.

Right Window (Readers):

- General tab: Task type: Session, Select task: s_m_SEGMENTOS_TABLE_TO_TIPOOK.
- Properties tab: Shows a table with one row for 'SQ_STG_SEGMENTOS VARCHAR' with 'Relational Reader' selected.
- Connections tab: Shows a connection named 'PROYECTO'.
- Properties tab (bottom): Shows settings for 'SQ_STG_SEGMENTOS VARCHAR - Source' including 'Sql Query' and 'Source Filter'.

Third 'Edit Tasks' window for the 's_m_SEGMENTOS_TABLE_TO_TIPOOK' task.

- General tab: Task type: Session, Select task: s_m_SEGMENTOS_TABLE_TO_TIPOOK.
- Properties tab: Shows a table with one row for 'RTR_SEGMENTOS_TABLE_TO_TIPOOK' with 'Normal' selected under 'Tracing Level'.

EJECUCIÓN: m_SEGMENTOS_TABLE_TO_TIPOOK

Object Explorer

SQLQuery1.sql - D...ELEMENTOS (sa (76))

```
use DATAMART_SEGMENTOS
GO
TRUNCATE TABLE STG_SEGMENTOS_TIPOOK
SELECT * FROM STG_SEGMENTOS_TIPOOK
```

Results Messages

FECHA	CONTADOR	CP	ID_TERRENO	ID_TIPOE	SEGMENTO	DURACION	DISTANCIA	POTENCIA	RITMO	FREC_CARDIACA	CADEN
-------	----------	----	------------	----------	----------	----------	-----------	----------	-------	---------------	-------

Informatica PowerCenter Workflow Monitor

Repository Edit View Tools Task Filters Help

Workflow Run: wf_ACTUALIZACIONES

Start Time	Completion Time	Status
30/08/2022 19:37:29	30/08/2022 19:41:12	Succeeded
30/08/2022 19:37:49	30/08/2022 19:37:49	Succeeded
30/08/2022 19:37:52	30/08/2022 19:38:09	Succeeded
30/08/2022 19:38:01	30/08/2022 19:38:30	Succeeded
30/08/2022 19:38:35	30/08/2022 19:38:52	Succeeded
30/08/2022 19:38:53	30/08/2022 19:39:11	Succeeded
30/08/2022 19:39:12	30/08/2022 19:39:29	Succeeded
30/08/2022 19:39:30	30/08/2022 19:39:47	Succeeded
30/08/2022 19:39:39	30/08/2022 19:39:47	Succeeded

s_m_SEGMENTOS_TABLE_TO_TIPOOK [30/08/2022 19:39:30]

Task Details

Attribute Value

Instance Name	Value
s_m_SEGMENTOS_TABLE_TO_TIPOOK	
Session	DATAMART_SEGMENTOS_SERVICIO
node01_DESKTOP-8M9PCQ2	
Start Time	30/08/2022 19:39:30
End Time	30/08/2022 19:39:47
Recovery Time(s)	
Status	Succeeded
Status Message	No
Deleted	1
Version Number	1229
Source Success Rows	1229
Source Failed Rows	0
Target Success Rows	1229
Target Failed Rows	0
Total Transformation Errors	0

Source/Target Statistics

Transformation Name	Node	Applied Rows	Affected Rows	Rejected Rows	Throughput (Rows/Sec)
s_m_SEGMENTOS_TABLE_TO_TIPOOK	node01_DESKTOP-8M9PCQ2	1229	1229	0	1229
STG_SEGMENTOS_TABLE_TO_TIPOOK	node01_DESKTOP-8M9PCQ2	1229	1229	0	1229

Object Explorer

SQLQuery1.sql - D...ELEMENTOS (sa (76))

```
SELECT * FROM STG_SEGMENTOS_TIPOOK
```

Results Messages

FECHA	CONTADOR	CP	ID_TERRENO	ID_TIPOE	SEGMENTO	DURACION	DISTANCIA	POTENCIA	RITMO	FREC_CARDIACA	CADEN	
1225	2022-07-02	31	225	7	2	31	2037	4000	169	509	123	162
1226	2022-07-02	32	225	7	2	32	1998	4000	173	499	140	164
1227	2022-07-02	33	225	7	2	33	2452	5000	176	490	134	164
1228	2022-07-02	34	225	7	2	34	3599	7180	172	501	130	163
1229	2022-07-02	35	225	7	2	35	3597	7200	173	500	134	164

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 se 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 dtypes
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 -2.072e-04	0.1996	-0.204
DISTANCIA	1.695e-01	1.000	0.125	...	0.884 9.190e-01	1.1996	
RITMO	-9.282e-01	0.125	1.000	...	0.078 -2.691e-02	0.173	
FREC_CARDIACA	4.198e-01	-0.069 -0.450	...	0.055	1.322e-01	-0.111	
CADENCIA	6.926e-01	-0.045 -0.705	...	-0.059 1.503e-02	-0.079		
TSC	-6.121e-01	0.166 0.642	...	0.070 1.583e-02	0.187		
FP	2.008e-01	0.066 -0.231	...	0.033 7.943e-02	0.040		
LSS	-3.089e-01	0.106 0.353	...	0.016 3.000e-02	0.050		
OSC_VERTICAL	1.054e-01	0.064 -0.100	...	-0.181 -1.315e-01	-0.218		
L_ZANCADA	8.347e-01	-0.204 -0.804	...	-0.075 4.288e-02	-0.035		
RFP	-9.021e-01	0.166 0.903	...	0.047 -1.051e-02	0.204		
RLSS	-3.089e-01	0.106 0.353	...	0.015 -2.183e-02	0.121		
ROV	-4.972e-01	0.110 0.559	...	0.020 -5.075e-02	0.138		
RE	1.397e-02	-0.036 0.033	...	-0.075 4.288e-02	-0.035		
AIRE	3.011e-01	-0.070 -0.223	...	0.013 5.406e-02	-0.034		
PENDIENTE	3.037e-01	-0.070 -0.152	...	0.094 -2.633e-02	-0.071		
ALTITUD	4.285e-02	-0.022 -0.039	...	0.033 -1.704e-02	-0.026		
DESNIVEL	-8.951e-02	0.084 0.078	...	1.000 8.727e-01	0.871		
RSS	-2.072e-04	0.919 -0.027	...	0.873 1.0000e+00	0.897		
DURACION	2.045e-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 -0.60891	0.20081 0.30891	0.10541 0.83468	-0.90210 -0.30891					
DISTANCIA	-0.16949	1.00000 0.12502	-0.06894 -0.04490	-0.04490 0.16574	0.06567 0.10647	0.06360 0.020385	-0.16606 0.16606	0.03600	-0.20385	0.16606	0.10647	
RITMO	-0.92822	0.12502 1.00000	-0.43621 -0.70546	-0.70546 0.64159	-0.23067 0.35270	-0.23067 -0.10577	0.35270 -0.04964	0.16248 0.52002	-0.46769 -0.47800	-0.46769 0.52002	-0.47800 0.29849	-0.25570
FREC_CARDIACA	0.41982	-0.06894 -0.43621	1.00000 0.32962	0.32962 0.00000	-0.03174 0.54750	-0.03174 0.29849	0.10208 0.43847	-0.25509 0.53328	-0.25509 0.53328	-0.25509 0.53328	-0.25509 0.53328	-0.25510
CADENCIA	0.69263	-0.04490 -0.70546	0.32962 1.00000	0.00000 0.46393	0.46393 0.47364	0.46393 0.53328	0.54750 0.90893	-0.46515 0.01688	-0.46515 0.01688	-0.46515 0.01688	-0.46515 0.01688	-0.46515
TSC	-0.61209	0.16574 0.64159	-0.29286 -0.03474	-0.03474 0.00000	0.00000 0.46393	0.00000 0.47364	0.29286 0.53328	-0.46515 0.12105	-0.46515 0.12105	-0.46515 0.12105	-0.46515 0.12105	-0.46515
FP	0.20081	0.06567 -0.23067	-0.23067 0.10208	-0.10208 0.54750	-0.46553 -1.00000	-0.46553 -0.68912	0.54750 0.90893	-0.46515 0.01688	-0.46515 0.01688	-0.46515 0.01688	-0.46515 0.01688	-0.46515
LSS	-0.30891	0.10647 0.36270	-0.25509 0.29849	-0.25509 0.29849	0.29849 0.87364	0.29849 0.68912	0.87364 0.71131	-0.68912 0.71131	-0.68912 0.71131	-0.68912 0.71131	-0.68912 0.71131	-0.68912
OSC_VERTICAL	0.10541	-0.06360 -0.10577	-0.04964 0.43847	-0.04964 0.43847	0.43847 0.53328	0.43847 0.90893	0.53328 0.71131	-0.10000 0.71131	-0.10000 0.71131	-0.10000 0.71131	-0.10000 0.71131	-0.10000
L_ZANCADA	0.83468	-0.20385 0.0417	0.16248 0.52002	0.16248 0.52002	-0.65115 0.01688	-0.65115 0.01688	-0.52002 -0.35077	-0.35077 0.03010	-0.35077 0.03010	-0.35077 0.03010	-0.35077 0.03010	-0.35076
RFP	-0.90210	0.16606 0.90259	-0.46769 -0.47800	-0.46769 -0.47800	0.84292 0.12105	0.84292 0.62413	0.47800 0.21993	-0.75139 0.00000	-0.75139 0.00000	-0.75139 0.00000	-0.75139 0.00000	-0.75139
RLSS	0.30891	0.10647 0.35270	-0.25510 0.29849	-0.25510 0.29849	0.87364 0.68912	0.87364 1.00000	0.29849 0.71130	-0.68912 0.71130	-0.68912 0.71130	-0.68912 0.71130	-0.68912 0.71130	-0.68912
ROV	-0.49721	0.11013 0.56881	-0.40562 0.00025	-0.40562 0.00025	0.90645 0.53407	0.90645 0.88715	0.00025 0.65515	-0.40562 0.65515	-0.40562 0.65515	-0.40562 0.65515	-0.40562 0.65515	-0.40562
RE	0.01397	-0.03601 -0.03251	-0.06244 0.00072	-0.06244 0.00072	-0.05433 0.07894	-0.05433 0.05909	0.00072 -0.05909	-0.05909 0.06971	-0.05909 0.06971	-0.05909 0.06971	-0.05909 0.06971	-0.05909
AIRE	0.30112	-0.03036 -0.22269	0.08725 0.15603	0.08725 0.15603	-0.14689 0.00630	-0.14689 0.00630	0.15603 -0.09770	-0.09770 0.00920	-0.09770 0.00920	-0.09770 0.00920	-0.09770 0.00920	-0.09770
PENDIENTE	0.30373	-0.06978 -0.13169	0.13701 0.15734	0.13701 0.15734	-0.07206 0.11614	-0.07206 0.11614	0.15734 0.06218	-0.07206 0.10624	-0.07206 0.10624	-0.07206 0.10624	-0.07206 0.10624	-0.07206
ALTITUD	0.04285	-0.02209 -0.03938	0.04217 -0.00386	0.04217 -0.00386	-0.04720 0.02972	-0.04720 0.02972	0.04217 -0.04256	-0.04256 0.02426	-0.04256 0.02426	-0.04256 0.02426	-0.04256 0.02426	-0.04256
DESNIVEL	-0.08951	0.88441 0.07767	0.05488 -0.05852	0.05488 -0.05852	0.07017 0.03321	0.07017 0.03321	-0.05852 0.01459	0.01459 0.01621	0.01459 0.01621	0.01459 0.01621	0.01459 0.01621	0.01459
RSS	-0.00027	0.91901 -0.02691	0.13224 0.01503	0.13224 0.01503	0.01883 0.07943	0.01883 0.07943	-0.02183 0.03600	0.03600 -0.13152	0.03600 -0.13152	0.03600 -0.13152	0.03600 -0.13152	-0.13152
DURACION	-0.20447	0.99644 0.17274	-0.11126 -0.07888	-0.11126 -0.07888	0.18733 0.03963	0.18733 0.03963	0.01126 0.12065	0.12065 0.05012	0.12065 0.05012	0.12065 0.05012	0.12065 0.05012	0.12065

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 perturbe “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)
print(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 NUESTRO 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 dtypes
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 NUESTRO 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())
```

	POR	CP	DISTANCIA	RITMO	FREC_CARDIACA	CADENCIA	TSC	FP	LSS	\
0	85.863	500	519	154	166	340	64	12.0		
1	84.816	1000	547	130	159	343	66	11.8		
2	84.293	1000	537	137	161	346	66	12.1		
3	84.816	1000	548	142	162	347	66	12.2		
4	84.816	1000	539	146	163	345	65	12.1		
5	84.816	1000	542	150	163	346	65	12.3		
6	84.816	1000	537	152	165	339	65	12.4		
7	85.863	600	524	155	165	341	64	12.1		
8	85.340	530	561	128	158	348	65	11.7		
9	84.816	520	528	132	167	345	66	12.2		
10	84.816	2000	545	134	160	345	66	12.1		
11	84.816	2000	540	144	162	346	66	12.2		
12	84.816	2000	539	151	164	342	65	12.3		
13	84.816	3000	541	136	161	345	66	12.0		
14	84.816	3000	539	149	164	343	65	12.3		

	OSC_VERTICAL	L_ZANCIADA	RFP	RLSS	ROV	RE	AIRE	PENDIENTE	\
0	5.06	0.697	39.024	146.341	7.228	0.965	1	-0.5	
1	5480.00	0.689	40.740	143.902	8.058	0.925	0	0.4	
2	5470.00	0.693	40.993	147.568	7.927	0.948	0	0.0	
3	5420.00	0.685	40.740	148.788	7.970	0.937	0	0.2	
4	5290.00	0.682	40.123	147.568	7.779	0.939	0	0.0	
5	5230.00	0.679	40.123	150.000	7.691	0.933	0	0.3	
6	5180.00	0.677	40.123	151.219	7.617	0.942	1	-0.2	
7	5070.00	0.688	39.024	147.568	7.347	0.946	1	-0.2	
8	5410.00	0.670	39.877	142.682	8.074	0.888	1	0.8	
9	5340.00	0.683	40.740	148.788	7.852	0.939	0	0.1	
10	5480.00	0.693	40.740	145.121	7.942	0.935	0	0.2	
11	5360.00	0.685	40.740	148.788	7.882	0.937	0	0.1	
12	5210.00	0.678	40.123	150.000	7.661	0.938	1	0.0	
13	5460.00	0.689	40.740	146.341	7.913	0.936	0	0.2	
14	5240.00	0.677	40.123	150.000	7.594	0.937	0	0.0	

	ALTITUD	DESNIVEL	RSS	DURACION
0	19	2.0	5	259
1	16	7.0	10	547
2	18	4.0	9	537
3	18	6.0	9	540
4	19	5.0	9	539
5	18	7.0	9	542
6	18	2.0	9	537
7	19	3.0	6	317
8	16	5.0	5	300
9	18	27.0	51	3000
10	17	11.0	18	1082
11	19	11.0	18	1080
12	18	9.0	18	1079
13	17	17.0	28	1622
14	18	13.0	28	1619

Visualización Univariable

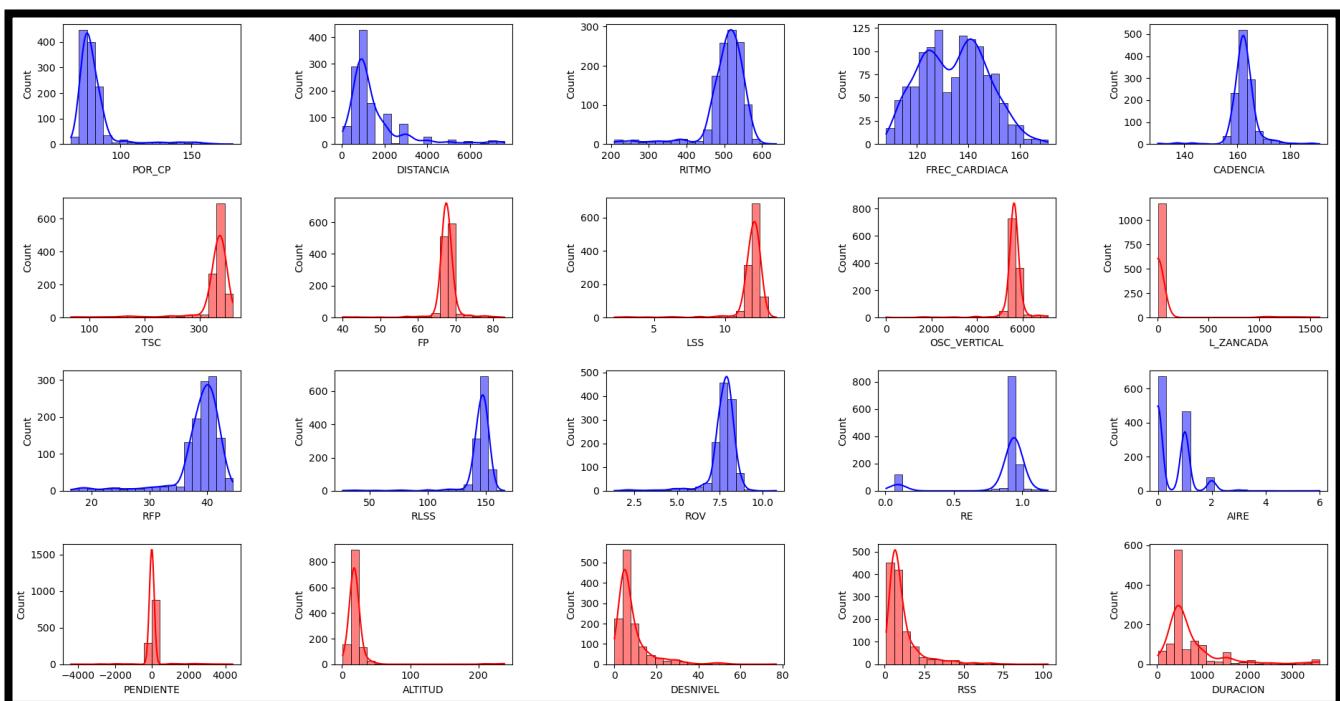
Como se ha comentado anteriormente, las gráficas univariable nos permiten visualizar los atributos individuales sin interacciones; las cuales, el objetivo principal de las mismas es aprender algo sobre la distribución, la tendencia y la propagación de cada atributo.

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.

```
# ****
# **** VIZUALIZACIONES ****
# ****
# 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["CAEDENCIA"], 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 distribuciones 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)

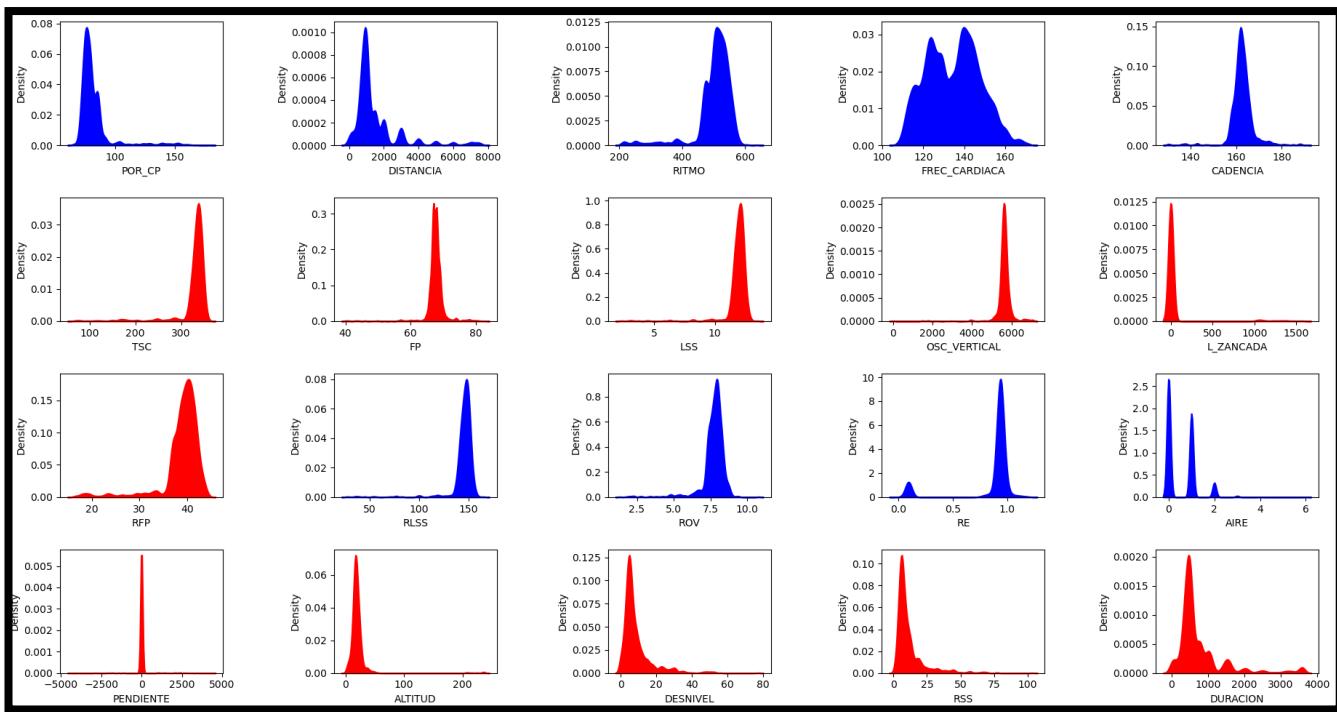
sns.kdeplot(data["ALTITUD"], ax = axes [3,1], shade = True, color = "Red", fill = True,
bw_adjust=.5, clip_on=False, alpha=1, linewidth=1.5)

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", saturation = 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)

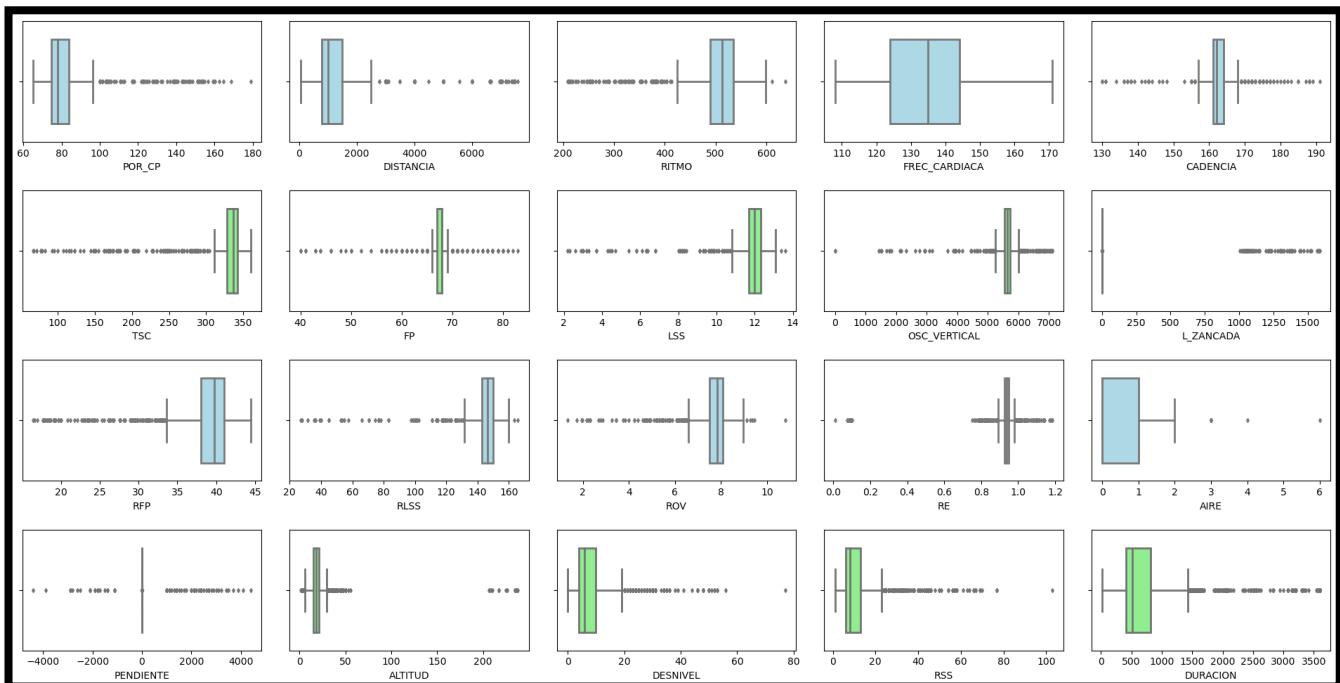
sns.boxplot(x = data["DESNIVEL"] , ax = axes [3,2], orient = "h", color = "lightgreen", saturation= 1, width = 0.7, dodge = True, fliersize = 3, linewidth = 2)

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("===== ")

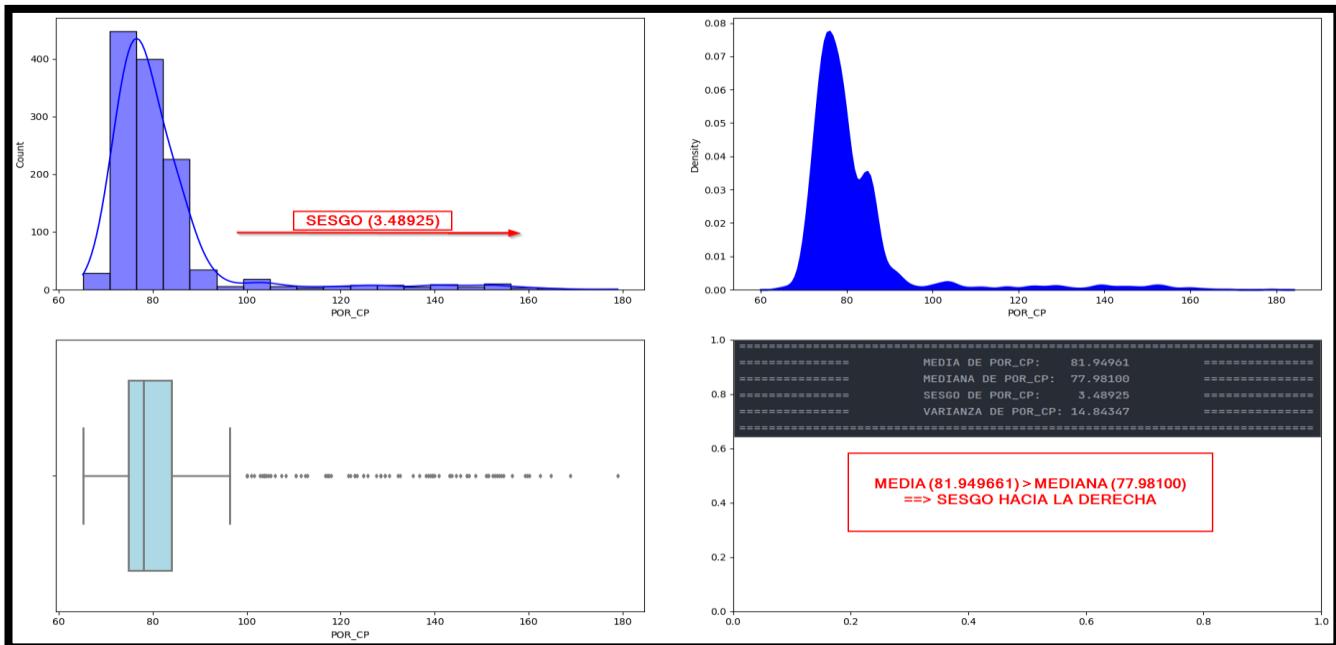
f, axes = plt.subplots(2,2, figsize =(11.7,8.27))

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"], saturation = 1, ax = axes [1,0],
width = 0.7, dodge = True, fliersize = 3, linewidth = 2)

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



Estudio Visualización "DISTANCIA"

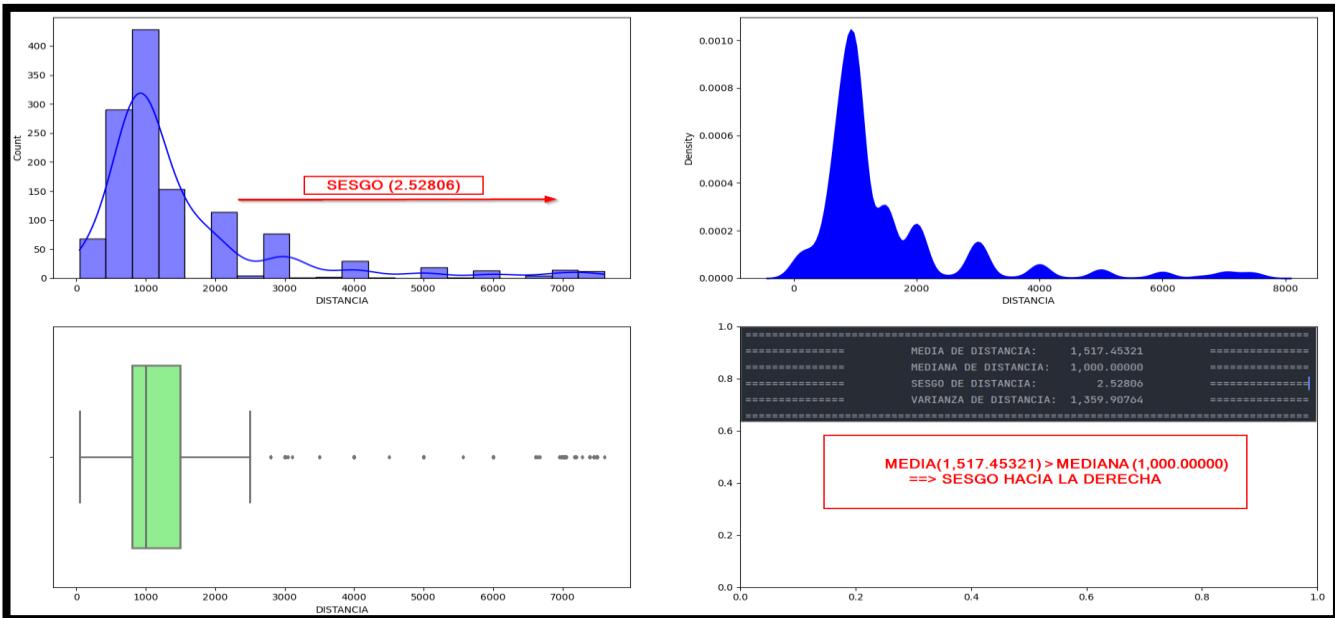
```
# Estudio Visualización "DISTANCIA"
# Métricas DISTANCIA
print("===== ")
print(f"          ===== MÉDIA DE DISTANCIA: {data['DISTANCIA'].mean():,.5f} ")
print(f"          ===== MEDIANA DE DISTANCIA: {data['DISTANCIA'].median():,.5f} ")
print(f"          ===== SESGO DE DISTANCIA: {data['DISTANCIA'].skew():,.5f}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

# 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"

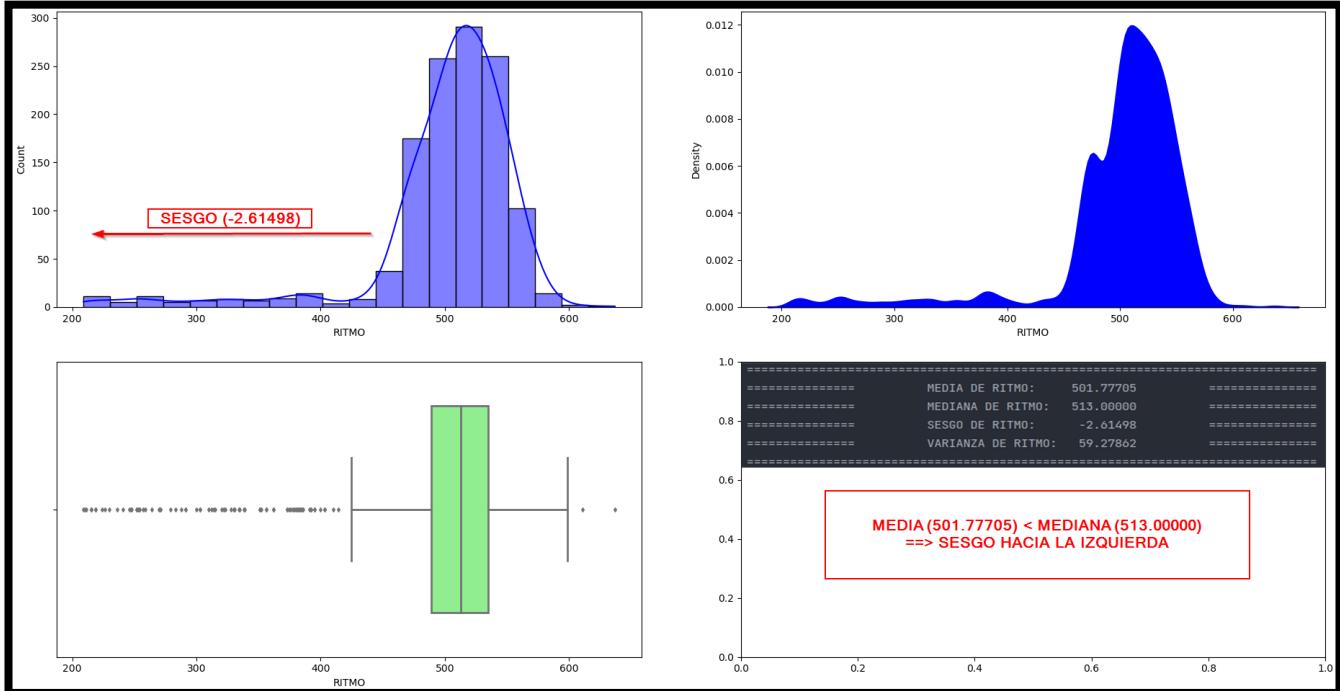
```
# Estudio Visualización "RITMO"
# Métricas RITMO
print("=====")
print(f"          ====== MEDIA DE RITMO: {data['RITMO'].mean():,.5f}")
print(f"          ====== MEDIANA DE RITMO: {data['RITMO'].median():,.5f}")
print(f"          ====== SESGO DE RITMO: {data['RITMO'].skew():,.5f}")
print(f"       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



Estudio Visualización "FREC_CARDIACA"

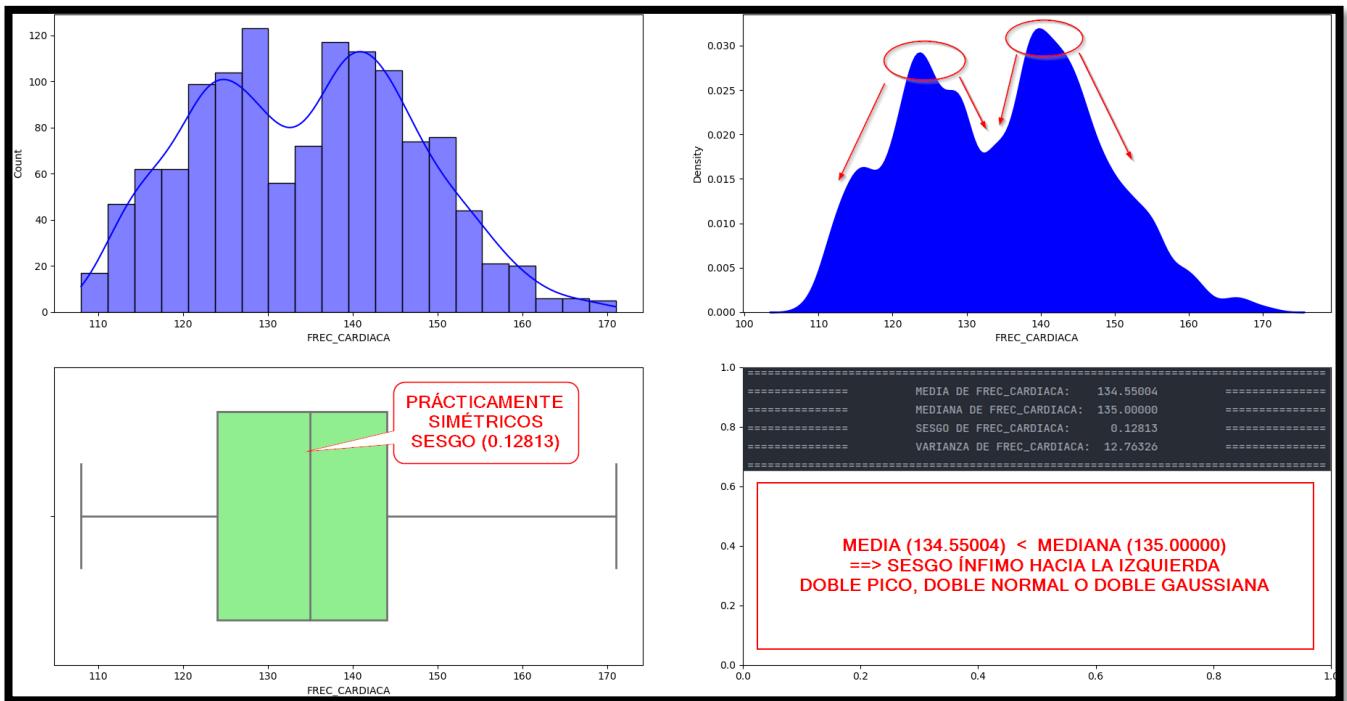
```
# Estudio Visualización "FREC_CARDIACA"
# Métricas FREC_CARDIACA
print("===== ")
print(f" ====== MEDIA DE FREC_CARDIACA: {data['FREC_CARDIACA'].mean():,.5f}")
print(f" ====== MEDIANA DE FREC_CARDIACA: {data['FREC_CARDIACA'].median():,.5f}")
print(f" ====== SESGO DE FREC_CARDIACA: {data['FREC_CARDIACA'].skew():,.5f}")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

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

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

sns.kdeplot(data["FREC_CARDIACA"], 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["FREC_CARDIACA"], 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 "CADENCIA"

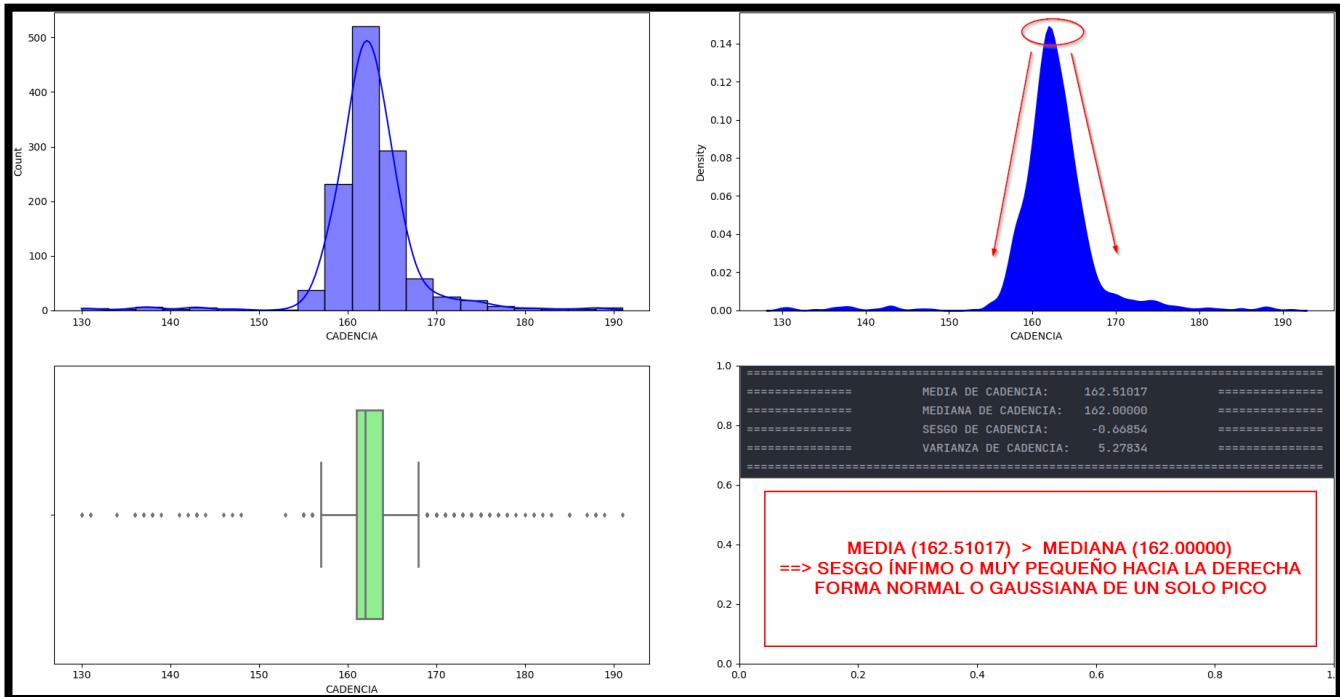
```
# Estudio Visualización "CADENCIA"
# Métricas CADENCIA
print("===== ")
print(f" {data['CADENCIA'].mean():,.5f} ===== MEDIA DE CADENCIA:")
print(f" {data['CADENCIA'].median():,.5f} ===== MEDIANA DE CADENCIA:")
print(f" {data['CADENCIA'].skew():,.5f} ===== SESGO DE CADENCIA:")
print(f" {data['CADENCIA'].std():,.5f}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

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

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

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

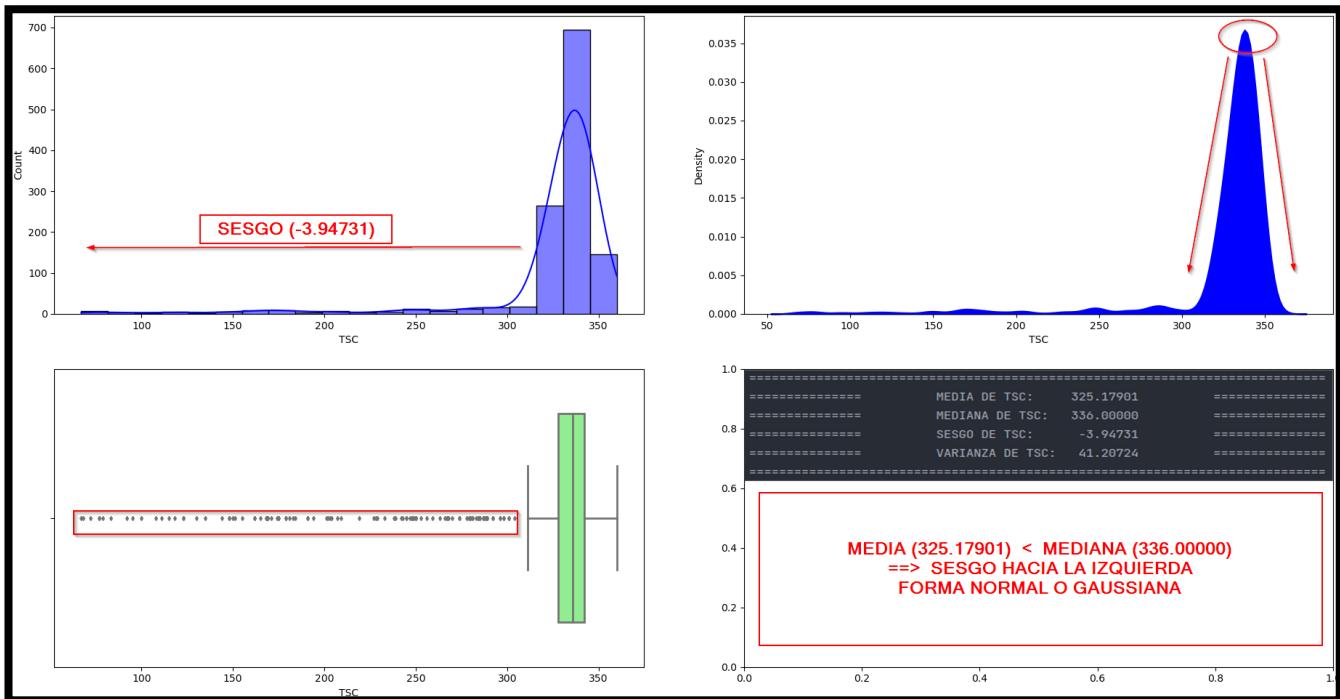


Estudio Visualización "TSC"

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

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

sns.histplot(data["TSC"], ax = axes [0,0], kde = True, bins = 20, color="Blue", fill=True)
sns.kdeplot(data["TSC"], 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["TSC"], ax = axes [1,0], orient = "h", color = "lightgreen", saturation
= 1, width = 0.7, dodge = True, fliersize = 3, linewidth = 2)
plt.tight_layout()
plt.show()
```

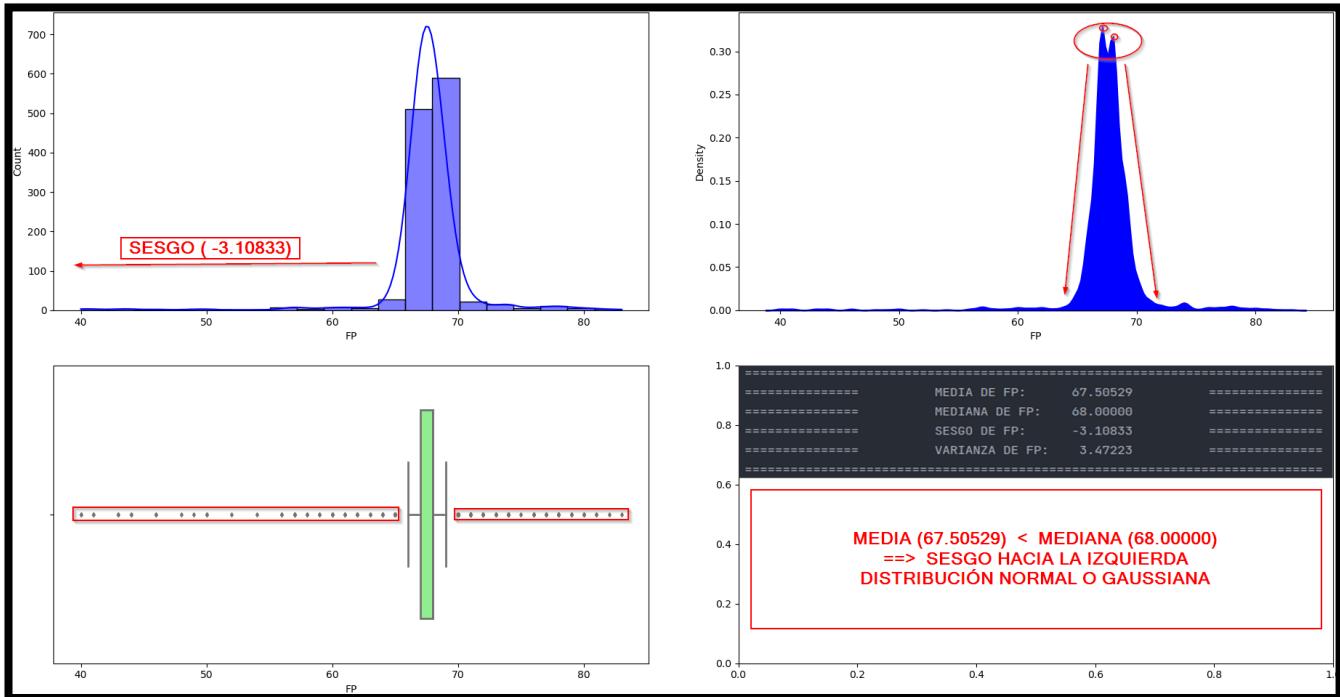


Estudio Visualización "FP"

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

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

sns.histplot(data["FP"], ax = axes [0,0], kde = True, bins = 20, color="Blue", fill=True)
sns.kdeplot(data["FP"], 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["FP"], ax = axes [1,0], orient = "h", color = "lightgreen", saturation
= 1, width = 0.7, dodge = True, fliersize = 3, linewidth =
2)
plt.tight_layout()
plt.show()
```

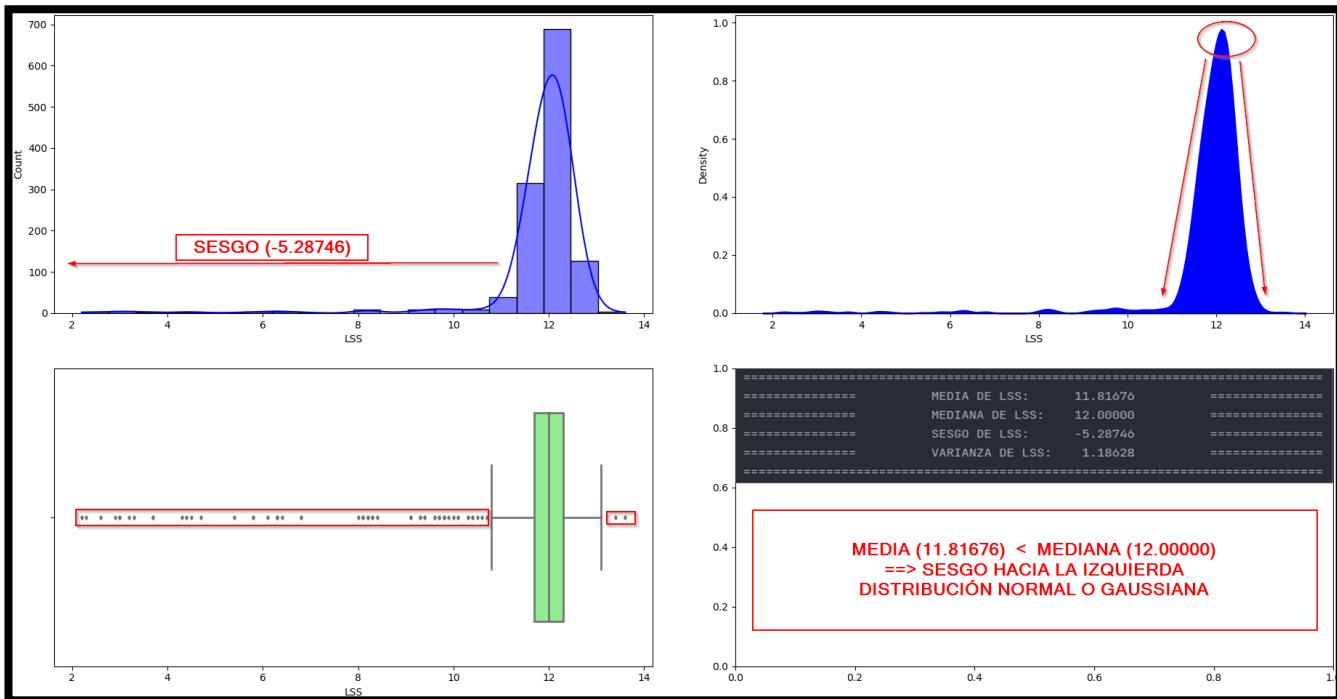


Estudio Visualización "LSS"

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

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

sns.histplot(data["LSS"], ax = axes [0,0], kde = True, bins = 20, color="Blue", fill=True)
sns.kdeplot(data["LSS"], 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["LSS"], ax = axes [1,0], orient = "h", color = "lightgreen", saturation
= 1, width = 0.7, dodge = True, fliersize = 3, linewidth = 2)
plt.tight_layout()
plt.show()
```



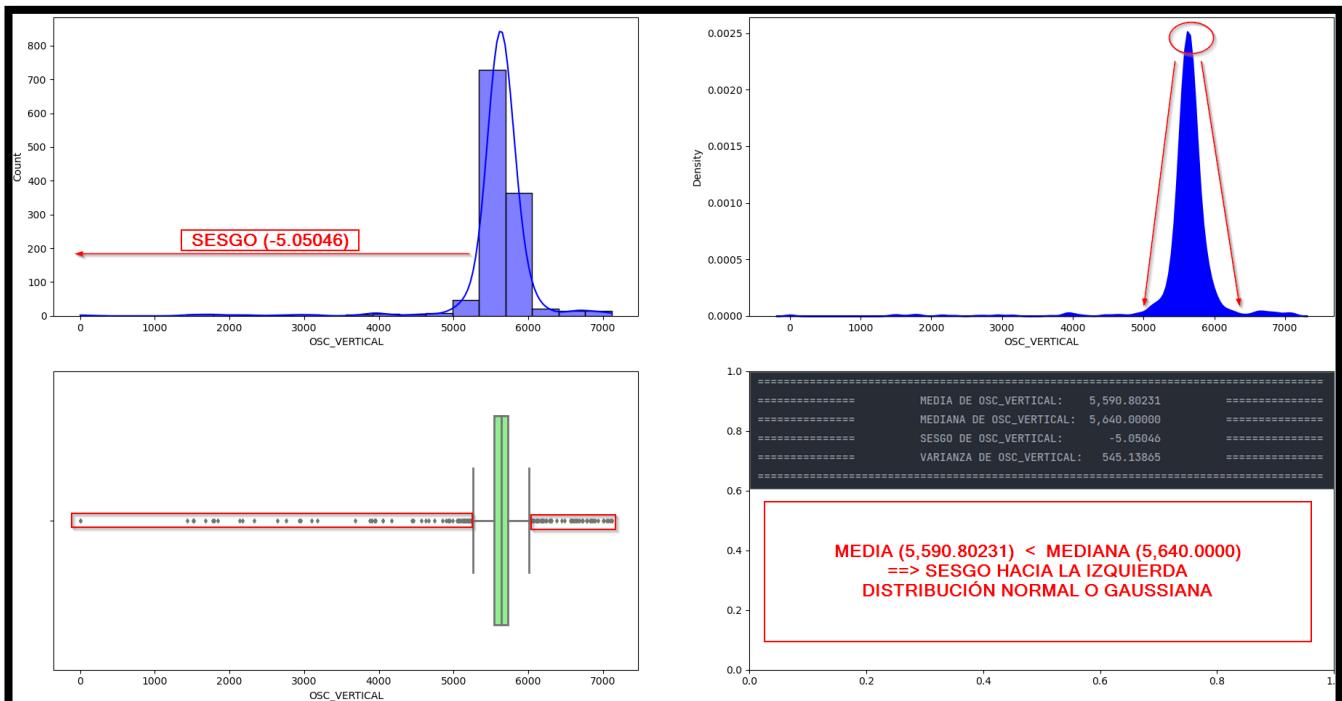
Estudio Visualización "OSC_VERTICAL"

```
# Estudio Visualización "OSC_VERTICAL"
# Métricas OSC_VERTICAL
print("=====-----")
print(f"          =====---")
{data['OSC_VERTICAL'].mean():,.5f}           MEDIA DE OSC_VERTICAL:
print(f"          =====---")
{data['OSC_VERTICAL'].median():,.5f}            MEDIANA DE OSC_VERTICAL:
print(f"          =====---")
{data['OSC_VERTICAL'].skew():,.5f}              SESGO DE OSC_VERTICAL:
print(f"          =====---")
{data['OSC_VERTICAL'].std():,.5f}                VARIANZA DE OSC_VERTICAL:
print("=====-----")
# Visualización Histograma, Densidad y Boxplot de OSC_VERTICAL
f, axes = plt.subplots(2,2, figsize =(11.7,8.27))

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

sns.kdeplot(data["OSC_VERTICAL"],      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["OSC_VERTICAL"],   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_Rj60S3wczDzgystpclwOS4kKI6R9/pb?fbclid=IwAR35xriNyEjStmE3fYp7BuESFklJyF-cwlivoOGpeDdtLjV39PKqChEFCvo
- Seaborn: <https://seaborn.pydata.org/index.html>