Modelo predictivo de métricas Stryd mediante Machine Learning.
"En este documento desarrollaremos la fase de análisis de los datos con los que trabajamos para desarrollar nuestro modelo predictivo"

FASE: Análisis de Datos - Estadística Descriptiva

David Víctor Gómez Ramírez
Técnico Superior en Desarrollo de Aplicaciones Multiplataforma especialidad en Big Data

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.

Deberemos entonces conocer:

- La forma.
- El tamaño.
- El tipo.
- El diseño general que tienen los datos.

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

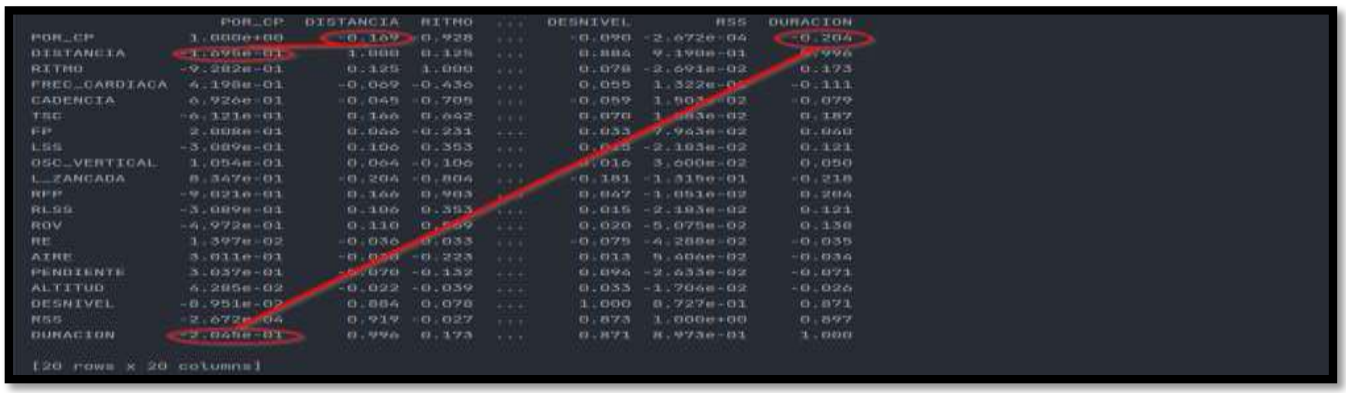
RFP	RLSS	ROV	RE	AIRE	PENDIENTE	ALTITUD	DESNIVEL	RSS	DURACION
1229.000000	1229.000000	1229.000000	1229.000000	1229.000000	1229.000000	1229.000000	1228.000000	1229.000000	1229.000000
38.749317	144.106313	7.676790	0.856816	0.537836	38.611798	20.847844	8.609935	11.626526	771.563059
4.314841	14.466858	0.860692	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	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	0.83468	-0.90210	-0.30891
DISTANCIA	-0.16949	1.00000	0.12504	-0.06894	-0.04490	0.16574	0.06567	0.10647	0.06360	-0.20385	0.16606	0.10647
RITMO	-0.92822	0.12504	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	-0.40503	0.87364	0.53328	-0.65115	0.84292	0.87364
FP	0.20081	0.06567	-0.23067	0.10206	0.54750	-0.40503	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	0.83468	-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	-0.48562	0.00025	0.90645	0.53407	0.88715	0.65515	-0.47255	0.77841	0.88716
RE	0.01397	-0.03601	-0.03251	-0.06244	0.00072	-0.05433	-0.07894	-0.05909	-0.06971	0.06387	-0.01355	-0.05909
AIRE	0.30112	-0.03036	-0.22269	0.08725	0.15603	-0.14689	-0.00630	-0.09770	-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	0.05486	-0.05862	0.07017	0.03321	0.01469	0.01621	-0.18089	0.04600	0.01469
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	0.12065

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 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())
```

Jupyter Notebook

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