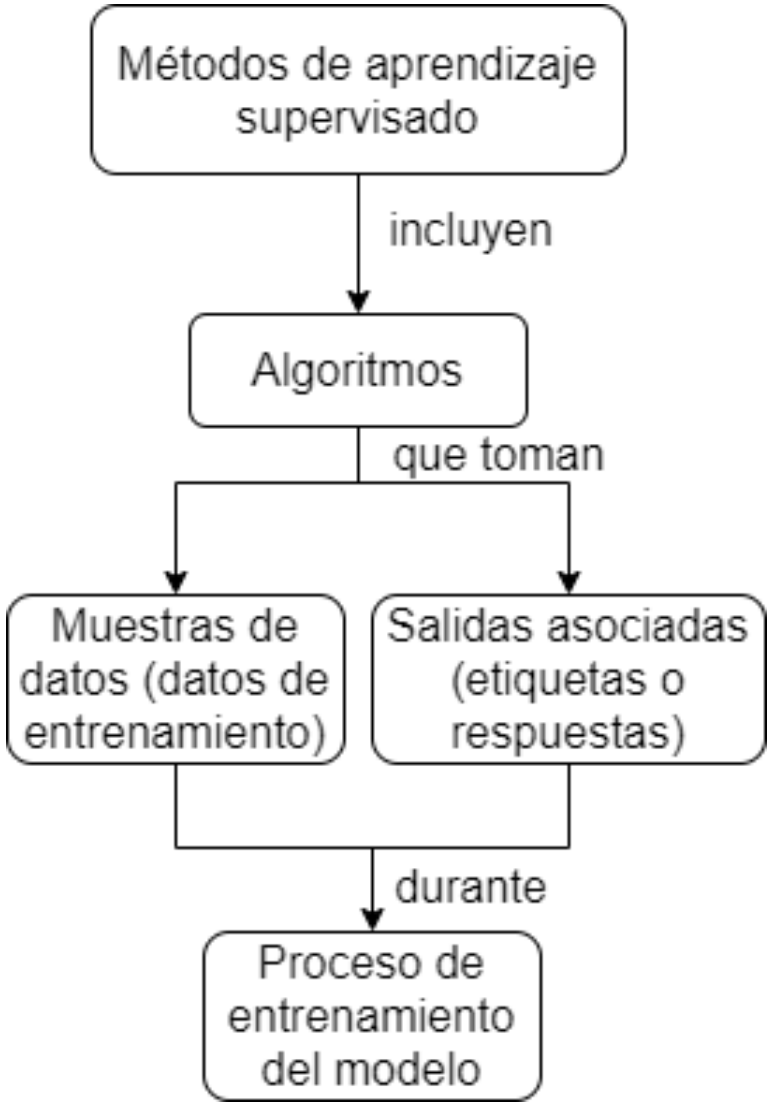
UNIVERSIDAD SANTO TOMÁS
PRIMER CLAUSTRO UNIVERSITARIO DE COLOMBIA
SECCIONAL TUNJA

VIGILADA MINEDUCACIÓN - SNIES 1732

MACHINE LEARNING CON PYTHON

INTRODUCCIÓN

Aprendizaje supervisado



Aprendizaje supervisado

El objetivo principal es definir un mapeo o asociación entre las muestras de datos de entrada ***x*** y sus correspondientes salidas ***y***, basándose en múltiples instancias de datos de entrenamiento. Este conocimiento aprendido se puede utilizar en el futuro para predecir una salida ***y'*** para cualquier nueva muestra de datos de entrada ***x'*** que antes se desconocía o no se veía durante el proceso de entrenamiento del modelo. Estos métodos se denominan supervisados porque el modelo aprende sobre muestras de datos donde las respuestas o etiquetas de salida deseadas ya se conocen de antemano en la fase de entrenamiento.

Aprendizaje supervisado

El aprendizaje supervisado a menudo requiere un esfuerzo humano para construir el conjunto de capacitación, pero luego lo automatiza y, a menudo, acelera una tarea que de otro modo sería laboriosa o inviable.

Aprendizaje supervisado

Los métodos de aprendizaje supervisado son de dos clases, según el tipo de tareas de aprendizaje automático que pretenden resolver:

- **Clasificación**: el objetivo es predecir una etiqueta de clase, que es una elección de una lista predefinida de posibilidades. Puede ser binaria (distinguir entre dos clases) o multiclase (clasificación entre más de dos clases). Ejemplo de binaria: responder pregunta sí o no – ¿Este es un correo spam?. Ejemplo de multiclase: la selección de especie de las flores de iris.
- **Regresión**: el objetivo es predecir un número real continuo. Ejemplo: predecir los ingresos anuales de una persona, conociendo su nivel de educación, su edad y el lugar donde vive.

Aprendizaje supervisado

Si se requiere hallar un valor continuo, entonces el problema es de regresión, de lo contrario se habla de clasificación.

Ejemplo: los ingresos son valores continuos, mientras seleccionar una especie de flor es un valor discreto, dentro de un número limitado de posiblidades.

Generalización: ocurre cuando el modelo es capaz de hacer predicciones precisas sobre datos invisibles.

REGRESIÓN LINEAL

Es una técnica que se utiliza para poder predecir variables continuas dependientes, a partir de un conjunto de variables independientes.

Es de carácter paramétrico, debido a que las suposiciones se realizan a partir de un conjunto de datos previo conocido.

REGRESIÓN LINEAL

EN TÉRMINOS MATEMÁTICOS: se utiliza la ecuación de la recta para aproximar los datos.

$$y = ax + b$$

y: variable dependiente o variable a predecir.
x: variable independiente o variable predictora
a: pendiente
b: intersección con el eje y

REGRESIÓN LINEAL

Con la regresión lineal se busca minimizar la distancia vertical entre los datos y la línea.

Uno de los métodos más usados es el conocido como Mínimos Cuadrados, donde se busca reducir el error entre la línea resultante y los puntos de los datos.

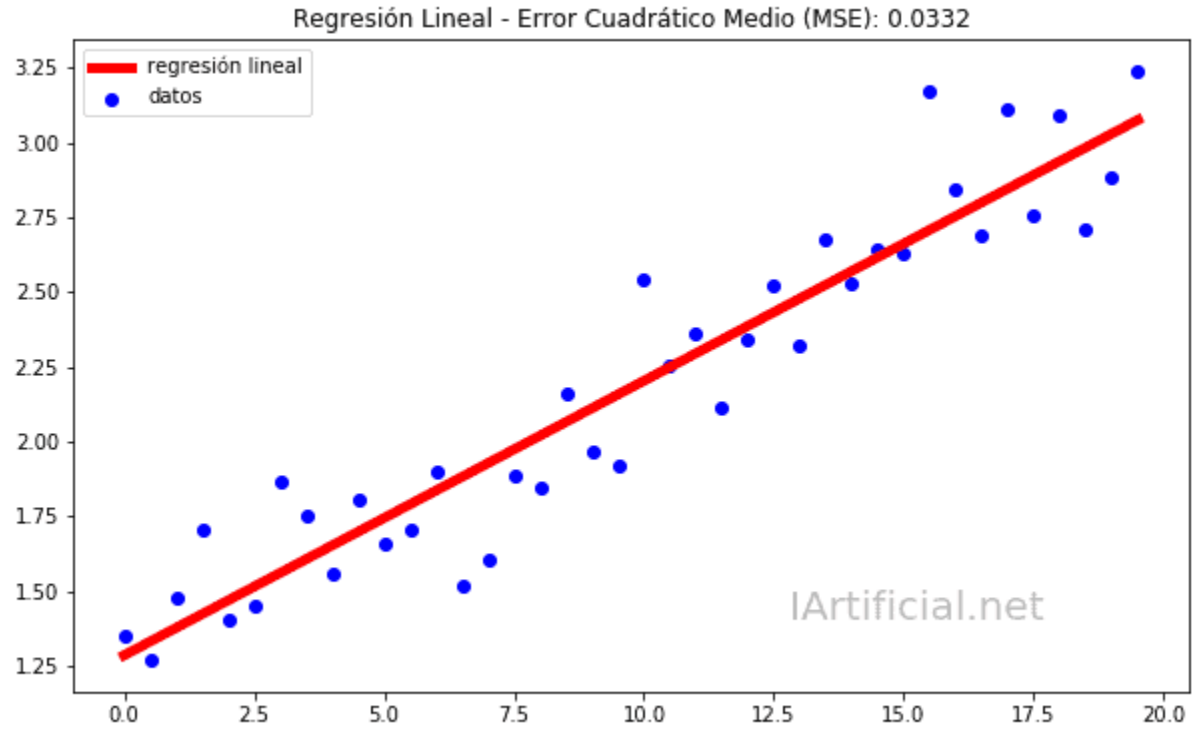


Imagen tomada de: https://www.iartificial.net/regresion-lineal-con-ejemplos-en-python/

REGRESIÓN LINEAL: SCIKIT LEARN

Es una librería de Python para implementar el algoritmo de regresión lineal.

En la documentación, vienen datasets útiles para practicar: https://scikit-learn.org/stable/datasets/toy_dataset.html

Ejemplo 1: Regresión lineal

1. Se debe establecer un problema que:
   - Se conozca el contexto.
   - Se busque predecir algún dato o evento.

Ejemplo 1: Regresión lineal

1. Se debe establecer un problema que:
   - Se conozca el contexto.
   - Se busque predecir algún dato o evento.

2. Se requieren datos sobre el problema que
   - Den resolución anterior al problema que se está resolviendo.
   - Contenga una buena cantidad de datos.
   - Sean fácil de conseguir o que ya se tengan.
   - Estén en formato de tabla (Excel o csv)

Ejemplo 1: Regresión lineal

1. Se debe establecer un problema que:
   - Se conozca el contexto.
   - Se busque predecir algún dato o evento.

2. Se requieren datos sobre el problema que
   - Den resolución anterior al problema que se está resolviendo.
   - Contenga una buena cantidad de datos.
   - Sean fácil de conseguir o que ya se tengan.
   - Estén en formato de tabla (Excel o csv)

3. Se debe contar con un software para programar el modelo.

Ejemplo 1: Regresión lineal

Ejemplos de problemas:

- ¿Cuánto tiempo empleará un corredor en la próxima maratón?
- ¿Qué menú ofrecer a un cliente nuevo?, según su edad, género y estilo.
- ¿Cuánto me gastaré en el próximo mes?

EJEMPLO PRÁCTICO

Ejemplo 1: Regresión lineal

Se deben seguir tres pasos básicos:

1. Preparar los datos
2. Entrenar el modelo
3. Realizar las predicciones.

Ejemplo 1: Regresión lineal

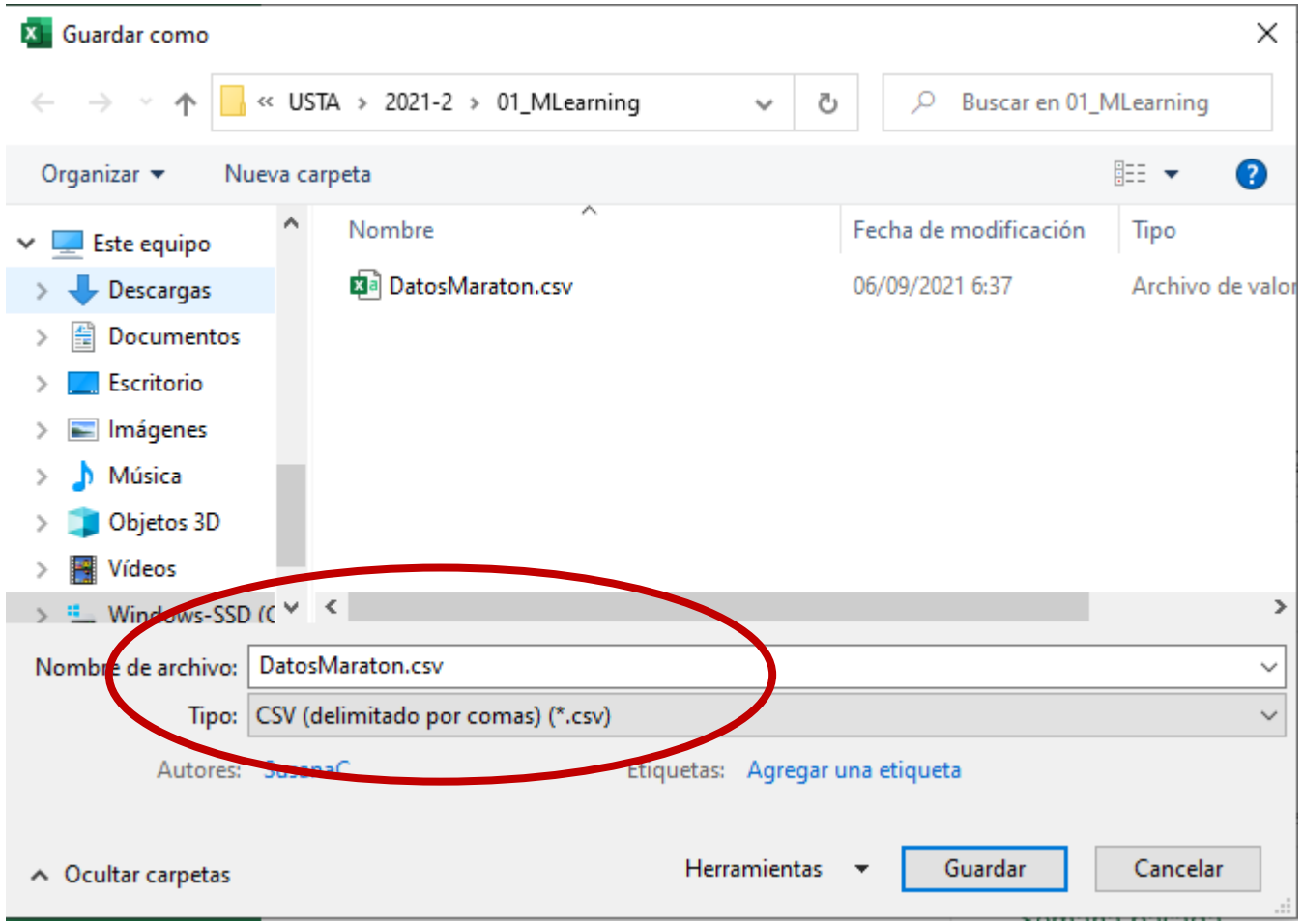
Ejemplo: ¿Cuánto tiempo gastará un corredor en correr una maratón?

1. Se busca o extrae un conjunto de datos acerca del problema. Para este ejemplo se descargaron los datos de https://www.kaggle.com/girardi69/marathon-time-predictions/version/2

Ejemplo 1: Regresión lineal

	id	Marathon	Name	Category	km4week	sp4week	CrossTraining	Wall21	MarathonTime	CATEGORY
2	1	Prague17	Blair MORGAN	MAM	132,8	1.443.478.261		1,16	2,37	A
3	2	Prague17	Robert Heczko	MAM	68,6	136.744.186		1,23	2,59	A
4	3	Prague17	Michon Jerome	MAM	82,7	1.352.043.597		1,3	2,66	A
5	4	Prague17	Daniel OrÂ lek	M45	137,5	1.225.854.383		1,32	2,68	A
6	5	Prague17	LukÂ ? MrÂ zek	MAM	84,6	1.394.505.495		1,36	2,74	A
7	6	Prague17	David Pecina	M40	42,2	1.361.290.323		1,32	2,78	A
8	7	Prague17	Tomas Drabek	M40	89	1.259.433.962		1,38	2,81	A
9	8	Prague17	Jan Rada	M45	106	1.269.461.078		1,41	2,84	A
10	9	Prague17	Tomas Drabek	MAM	70	137.704.918	ciclista 1h	1,38	2,83	A
11	10	Prague17	martin ?indelÂ ?	M45	84,2	1.336.507.937		1,35	2,86	A
12	11	Prague17	Maksim Remezau	MAM	93,5	13,2		1,42	2,87	A
13	12	Prague17	Jaroslaw Marchewka	M50	65,7	1.336.271.186		1,4	2,87	A
14	13	Prague17	TomÂ ? K?e?ek	M45	53,5	1.407.894.737	ciclista 4h	1,37	2,88	A
15	14	Prague17	Ji?ØŒ Polcar	M40	84,4	1.383.606.557		1,41	2,88	A
16	15	Prague17	Denis Wachtl	MAM	76,8	1.294.382.022		1,44	2,89	A
17	16	Prague17	David Lehnen	MAM	76,1	149.704.918		1,45	2,9	A
18	17	Prague17	JÙ¾rgen Steiner	M55	112,3	1.259.439.252	ciclista 12h	1,44	2,91	A

Ejemplo 1: Regresión lineal



Si se tiene en formato de tabla, convertir a csv

¡Siempre hacia lo alto!

1. PREPARAR LOS DATOS:

a) Cargar los datos

Ejemplo 1: Regresión lineal

PODEMOS ENCONTRAR DATASETS (CONJUNTO DE DATOS) GRATUITOS EN EL SIGUIENTE LINK:

https://www.kaggle.com/datasets

Ejemplo 1: Regresión lineal

En Jupyter:

In [5]:
```
import pandas as pd
import numpy as np
datos_maraton =pd.read_csv("MarathonData.csv")
datos_maraton
```

Out[5]:

	id	Marathon	Name	Category	km4week	sp4week	CrossTraining	Wall21	MarathonTime	CATEGORY
0	1	Prague17	Blair MORGAN	MAM	132.8	14.434783	NaN	1.16	2.37	A
1	2	Prague17	Robert Heczko	MAM	68.6	13.674419	NaN	1.23	2.59	A
2	3	Prague17	Michon Jerome	MAM	82.7	13.520436	NaN	1.30	2.66	A
3	4	Prague17	Daniel Or lek	M45	137.5	12.258544	NaN	1.32	2.68	A
4	5	Prague17	Luk ? Mr zek	MAM	84.6	13.945055	NaN	1.36	2.74	A
...	...	...	...	...	...	...	...	...	...	...
82	83	Prague17	Stefano Vegliani	M55	50.0	10.830325	NaN	2.02	3.93	D
83	84	Prague17	Andrej Madliak	M40	33.6	10.130653	ciclista 3h	1.94	3.93	D
84	85	Prague17	Yoi Ohsako	M40	55.4	11.043189	NaN	1.94	3.94	D
85	86	Prague17	Simon Dunn	M45	33.2	11.066667	NaN	2.05	3.95	D
86	87	Prague17	Pavel ?imek	M40	17.9	10.848485	ciclista 5h	2.05	3.98	D

87 rows × 10 columns

El archivo debe estar en la carpeta raíz

Ejemplo 1: Regresión lineal

En Colab:

```
from google.colab import files
datos = files.upload()
```

Elegir archivos MarathonData.csv

- **MarathonData.csv**(application/vnd.ms-excel) - 5664 bytes, last modified: 20/9/2019 - 100% done

```
Saving MarathonData.csv to MarathonData.csv
```

```
import pandas as pd
import io
datos_maraton = pd.read_csv(io.BytesIO(datos['MarathonData.csv']))
datos_maraton
```

	id	Marathon	Name	Category	km4week	sp4week	CrossTraining	Wall21	MarathonTime	CATEGORY
0	1	Prague17	Blair MORGAN	MAM	132.8	14.434783	NaN	1.16	2.37	A
1	2	Prague17	Robert Heczko	MAM	68.6	13.674419	NaN	1.23	2.59	A
2	3	Prague17	Michon Jerome	MAM	82.7	13.520436	NaN	1.30	2.66	A
3	4	Prague17	Daniel Or lek	M45	137.5	12.258544	NaN	1.32	2.68	A
4	5	Prague17	Luk ? Mr zek	MAM	84.6	13.945055	NaN	1.36	2.74	A

Ejemplo 1: Regresión lineal

Observar un campo en especial del total de los datos

```
datos_maraton["MarathonTime"]
```

```
0     2.37
1     2.59
2     2.66
3     2.68
4     2.74
      ...
82    3.93
83    3.93
84    3.94
85    3.95
86    3.98
Name: MarathonTime, Length: 87, dtype: float64
```

Ejemplo 1: Regresión lineal

Observar información general sobre el conjunto de datos

```
datos_maraton.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 87 entries, 0 to 86
Data columns (total 10 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   id             87 non-null     int64  
 1   Marathon       87 non-null     object 
 2   Name           87 non-null     object 
 3   Category       81 non-null     object 
 4   km4week        87 non-null     float64
 5   sp4week        87 non-null     float64
 6   CrossTraining  13 non-null     object 
 7   Wall21         87 non-null     object 
 8   MarathonTime   87 non-null     float64
 9   CATEGORY       87 non-null     object 
dtypes: float64(3), int64(1), object(6)
memory usage: 6.9+ KB
```

Ejemplo 1: Regresión lineal

Convertir algún campo a numérico

```
datos_maraton['Wall21'] = pd.to_numeric(datos_maraton['Wall21'],errors='coerce')
```

https://pandas.pydata.org/docs/reference/api/pandas.to_numeric.html

Ejemplo 1: Regresión lineal

Ver información estadística de las columnas numéricas del conjunto de datos

```
datos_maraton.describe()
```

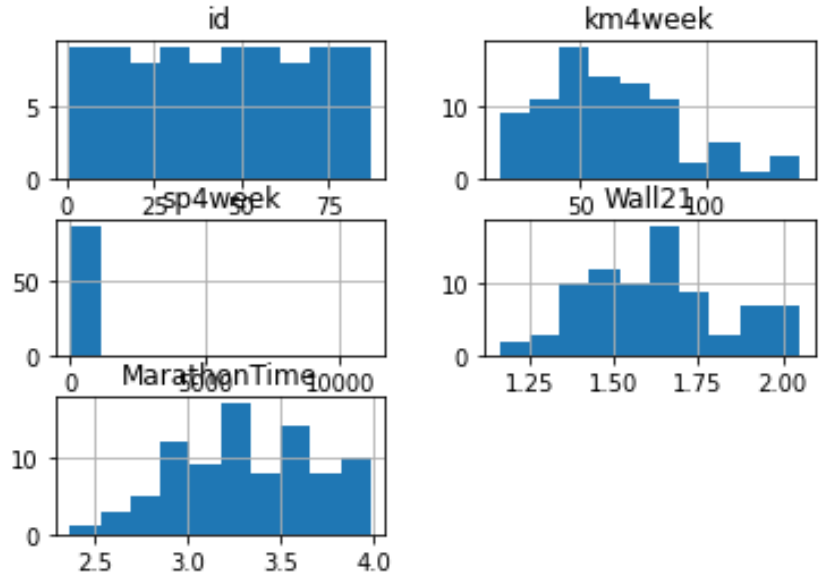
	id	km4week	sp4week	Wall21	MarathonTime
count	87.000000	87.000000	87.000000	81.000000	87.000000
mean	44.000000	62.347126	139.840706	1.630617	3.319080
std	25.258662	26.956019	1191.427864	0.210490	0.376923
min	1.000000	17.900000	8.031414	1.160000	2.370000
25%	22.500000	44.200000	11.498168	1.450000	3.045000
50%	44.000000	58.800000	12.163424	1.620000	3.320000
75%	65.500000	77.500000	12.854036	1.760000	3.605000
max	87.000000	137.500000	11125.000000	2.050000	3.980000

Ejemplo 1: Regresión lineal

Ver histograma de datos numéricos

```
datos_maraton.hist()
```

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f88ef867510>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f88ef846b50>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f88ef80b210>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f88ef7c0890>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f88ef776f10>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f88ef7395d0>]],
      dtype=object)
```



¡Siempre hacia lo alto!

1. PREPARAR LOS DATOS:

b) Escoger los campos más relevantes del conjunto de datos

Ejemplo 1: Regresión lineal

Se seleccionan los datos o columnas más relevantes para el objetivo de la predicción, en este caso serían:

	id	Marathon	Name	Category	km4week	sp4week	CrossTraining	Wall21	MarathonTime	CATEGORY
0	1	Prague17	Blair MORGAN	MAM	132.8	14.434783	NaN	1.16	2.37	A
1	2	Prague17	Robert Heczko	MAM	68.6	13.674419	NaN	1.23	2.59	A
2	3	Prague17	Michon Jerome	MAM	82.7	13.520436	NaN	1.30	2.66	A
3	4	Prague17	Daniel Or lek	M45	137.5	12.258544	NaN	1.32	2.68	A
4	5	Prague17	Luk ? Mr zek	MAM	84.6	13.945055	NaN	1.36	2.74	A

Ejemplo 1: Regresión lineal

Se seleccionan los datos o columnas más relevantes para el objetivo de la predicción, en este caso serían:

	id	Marathon	Name	Category	km4week	sp4week	CrossTraining	Wall21	MarathonTime	CATEGORY
0	1	Prague17	Blair MORGAN	MAM	132.8	14.434783	NaN	1.16	2.37	A
1	2	Prague17	Robert Heczko	MAM	68.6	13.674419	NaN	1.23	2.59	A
2	3	Prague17	Michon Jerome	MAM	82.7	13.520436	NaN	1.30	2.66	A
3	4	Prague17	Daniel Or lek	M45	137.5	12.258544	NaN	1.32	2.68	A
4	5	Prague17	Luk ? Mr zek	MAM	84.6	13.945055	NaN	1.36	2.74	A

Ejemplo 1: Regresión lineal

Se borran los que no tienen relación con la variable a analizar

```
datos_maraton = datos_maraton.drop(columns=['Name'])
datos_maraton = datos_maraton.drop(columns=['id'])
datos_maraton = datos_maraton.drop(columns=['Marathon'])
datos_maraton = datos_maraton.drop(columns=['CATEGORY'])
datos_maraton
```

	Category	km4week	sp4week	CrossTraining	Wall21	MarathonTime
0	MAM	132.8	14.434783	NaN	1.16	2.37
1	MAM	68.6	13.674419	NaN	1.23	2.59
2	MAM	82.7	13.520436	NaN	1.30	2.66
3	M45	137.5	12.258544	NaN	1.32	2.68
4	MAM	84.6	13.945055	NaN	1.36	2.74
...	...	...	...	...	...	...
82	M55	50.0	10.830325	NaN	2.02	3.93
83	M40	33.6	10.130653	ciclista 3h	1.94	3.93
84	M40	55.4	11.043189	NaN	1.94	3.94
85	M45	33.2	11.066667	NaN	2.05	3.95
86	M40	17.9	10.848485	ciclista 5h	2.05	3.98

87 rows × 6 columns

1. PREPARAR LOS DATOS:

c) Verificar si hay datos nulos y tomar acción

Ejemplo 1: Regresión lineal

Debemos verificar si hay datos nulos en el conjunto de datos y tomar la decisión de si se rellenan los faltantes o se elimina todo el registro

```
datos_maraton.isna().sum()
```

```
Category          6
km4week           0
sp4week           0
CrossTraining    74
Wall21            6
MarathonTime      0
dtype: int64
```

Ejemplo 1: Regresión lineal

Debemos verificar si hay datos nulos en el conjunto de datos y tomar la decisión de si se rellenan los faltantes o se elimina todo el registro

```
datos_maraton.isna().sum()
```

```
Category          6
km4week           0
sp4week           0
CrossTraining    74
Wall21            6
MarathonTime      0
dtype: int64
```

Rellenar con ceros

```
datos_maraton["CrossTraining"] = datos_maraton["CrossTraining"].fillna(0)
datos_maraton
```

	Category	km4week	sp4week	CrossTraining	Wall21	MarathonTime
0	MAM	132.8	14.434783	0	1.16	2.37
1	MAM	68.6	13.674419	0	1.23	2.59
2	MAM	82.7	13.520436	0	1.30	2.66
3	M45	137.5	12.258544	0	1.32	2.68
4	MAM	84.6	13.945055	0	1.36	2.74
...	...	...	...	...	...	...

Ejemplo 1: Regresión lineal

Debemos verificar si hay datos nulos en el conjunto de datos y tomar la decisión de si se rellenan los faltantes o se elimina todo el registro

```
datos_maraton.isna().sum()
```

```
Category          6
km4week           0
sp4week           0
CrossTraining    74
Wall21            6
MarathonTime      0
dtype: int64
```

Eliminar registros

```
datos_maraton = datos_maraton.dropna(how='any')
datos_maraton
```

	Category	km4week	sp4week	CrossTraining	Wall21	MarathonTime
0	MAM	132.8	14.434783	0	1.16	2.37
1	MAM	68.6	13.674419	0	1.23	2.59
2	MAM	82.7	13.520436	0	1.30	2.66
3	M45	137.5	12.258544	0	1.32	2.68
4	MAM	84.6	13.945055	0	1.36	2.74
...	...	...	...	...	...	...
82	M55	50.0	10.830325	0	2.02	3.93
83	M40	33.6	10.130653	ciclista 3h	1.94	3.93
84	M40	55.4	11.043189	0	1.94	3.94
85	M45	33.2	11.066667	0	2.05	3.95
86	M40	17.9	10.848485	ciclista 5h	2.05	3.98

81 rows × 6 columns

1. PREPARAR LOS DATOS:

d) Aplicar tratamiento a valores NO numéricos

Ejemplo 1: Regresión lineal

El modelo requiere que los datos incluidos sean de tipo numérico, por tanto, aquellos que no lo sean, deben tener una transformación. En nuestro caso hay dos campos NO numéricos:

```
datos_maraton = datos_maraton.dropna(how='any')
datos_maraton
```

	Category	km4week	sp4week	CrossTraining	Wall21	MarathonTime
0	MAM	132.8	14.434783	0	1.16	2.37
1	MAM	68.6	13.674419	0	1.23	2.59
2	MAM	82.7	13.520436	0	1.30	2.66
3	M45	137.5	12.258544	0	1.32	2.68
4	MAM	84.6	13.945055	0	1.36	2.74
...	...	...	...	...	...	...
82	M55	50.0	10.830325	0	2.02	3.93
83	M40	33.6	10.130653	ciclista 3h	1.94	3.93
84	M40	55.4	11.043189	0	1.94	3.94
85	M45	33.2	11.066667	0	2.05	3.95
86	M40	17.9	10.848485	ciclista 5h	2.05	3.98

81 rows × 6 columns

Ejemplo 1: Regresión lineal

El modelo requiere que los datos incluidos sean de tipo numérico, por tanto, aquellos que no lo sean, deben tener una transformación. En nuestro caso hay dos campos NO numéricos:

1. Verificar cuántos valores diferentes tenemos.
2. Asignar un valor numérico a cada diferente categoría o valor

```
datos_maraton = datos_maraton.dropna(how='any')
datos_maraton
```

	Category	km4week	sp4week	CrossTraining	Wall21	MarathonTime
0	MAM	132.8	14.434783	0	1.16	2.37
1	MAM	68.6	13.674419	0	1.23	2.59
2	MAM	82.7	13.520436	0	1.30	2.66
3	M45	137.5	12.258544	0	1.32	2.68
4	MAM	84.6	13.945055	0	1.36	2.74
...	...	...	...	...	...	...
82	M55	50.0	10.830325	0	2.02	3.93
83	M40	33.6	10.130653	ciclista 3h	1.94	3.93
84	M40	55.4	11.043189	0	1.94	3.94
85	M45	33.2	11.066667	0	2.05	3.95
86	M40	17.9	10.848485	ciclista 5h	2.05	3.98

81 rows × 6 columns

Ejemplo 1: Regresión lineal

El modelo requiere que los datos incluidos sean de tipo numérico, por tanto, aquellos que no lo sean, deben tener una transformación. En nuestro caso hay dos campos NO numéricos:

1. Verificar cuántos valores diferentes tenemos.
2. Asignar un valor numérico a cada diferente categoría o valor

```
datos_maraton['CrossTraining'].unique()
```

```
array([0, 'ciclista 1h', 'ciclista 4h', 'ciclista 13h', 'ciclista 3h',
       'ciclista 5h'], dtype=object)
```

Ejemplo 1: Regresión lineal

El modelo requiere que los datos incluidos sean de tipo numérico, por tanto, aquellos que no lo sean, deben tener una transformación. En nuestro caso hay dos campos NO numéricos:

1. Verificar cuántos valores diferentes tenemos.
2. Asignar un valor numérico a cada diferente categoría o valor

```
datos_maraton['CrossTraining'].unique()
```

```
array([0, 'ciclista 1h', 'ciclista 4h', 'ciclista 13h', 'ciclista 3h',
       'ciclista 5h'], dtype=object)
```

```
valores_cross = {"CrossTraining":  {'ciclista 1h':1, 'ciclista 3h':2, 'ciclista 4h':3, 'ciclista 5h':4, 'ciclista 13h':5}}
datos_maraton.replace(valores_cross, inplace=True)
datos_maraton
```

Ejemplo 1: Regresión lineal

El modelo requiere que los datos incluidos sean de tipo numérico, por tanto, aquellos que no lo sean, deben tener una transformación. En nuestro caso hay dos campos NO numéricos:

1. Verificar cuántos valores diferentes tenemos.
2. Asignar un valor numérico a cada diferente categoría o valor

```
datos_maraton['CrossTraining'].unique()
```

```
array([0, 'ciclista 1h', 'ciclista 4h', 'ciclista 13h', 'ciclista 3h',
       'ciclista 5h'], dtype=object)
```

```
valores_cross = {"CrossTraining":  {'ciclista 1h':1, 'ciclista 3h':2, 'ciclista 4h':3, 'ciclista 5h':4, 'ciclista 13h':5}}
datos_maraton.replace(valores_cross, inplace=True)
datos_maraton
```

Hacer lo mismo para el campo Category

2. ENTRENAMIENTO:

a) Revisar la correlación entre la variable principal y las demás

Ejemplo 1: Regresión lineal

La variable a predecir es:

	Category	km4week	sp4week	CrossTraining	Wall21	MarathonTime
0	MAM	132.8	14.434783	0	1.16	2.37
1	MAM	68.6	13.674419	0	1.23	2.59
2	MAM	82.7	13.520436	0	1.30	2.66
3	M45	137.5	12.258544	0	1.32	2.68
4	MAM	84.6	13.945055	0	1.36	2.74
...	...	...	...	...	...	...
82	M55	50.0	10.830325	0	2.02	3.93
83	M40	33.6	10.130653	ciclista 3h	1.94	3.93
84	M40	55.4	11.043189	0	1.94	3.94
85	M45	33.2	11.066667	0	2.05	3.95
86	M40	17.9	10.848485	ciclista 5h	2.05	3.98

81 rows × 6 columns

Ejemplo 1: Regresión lineal

La variable a predecir es:

	Category	km4week	sp4week	CrossTraining	Wall21	MarathonTime
0	MAM	132.8	14.434783	0	1.16	2.37
1	MAM	68.6	13.674419	0	1.23	2.59
2	MAM	82.7	13.520436	0	1.30	2.66
3	M45	137.5	12.258544	0	1.32	2.68
4	MAM	84.6	13.945055	0	1.36	2.74
...	...	...	...	...	...	...
82	M55	50.0	10.830325	0	2.02	3.93
83	M40	33.6	10.130653	ciclista 3h	1.94	3.93
84	M40	55.4	11.043189	0	1.94	3.94
85	M45	33.2	11.066667	0	2.05	3.95
86	M40	17.9	10.848485	ciclista 5h	2.05	3.98

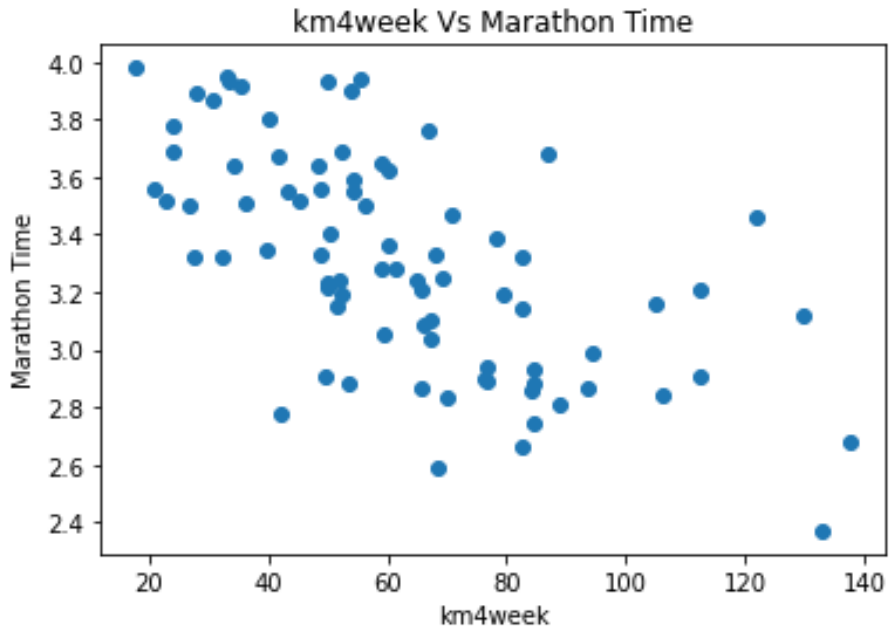
81 rows × 6 columns

Las demás son variables predictoras

Ejemplo 1: Regresión lineal

MaratonTime vs km4week

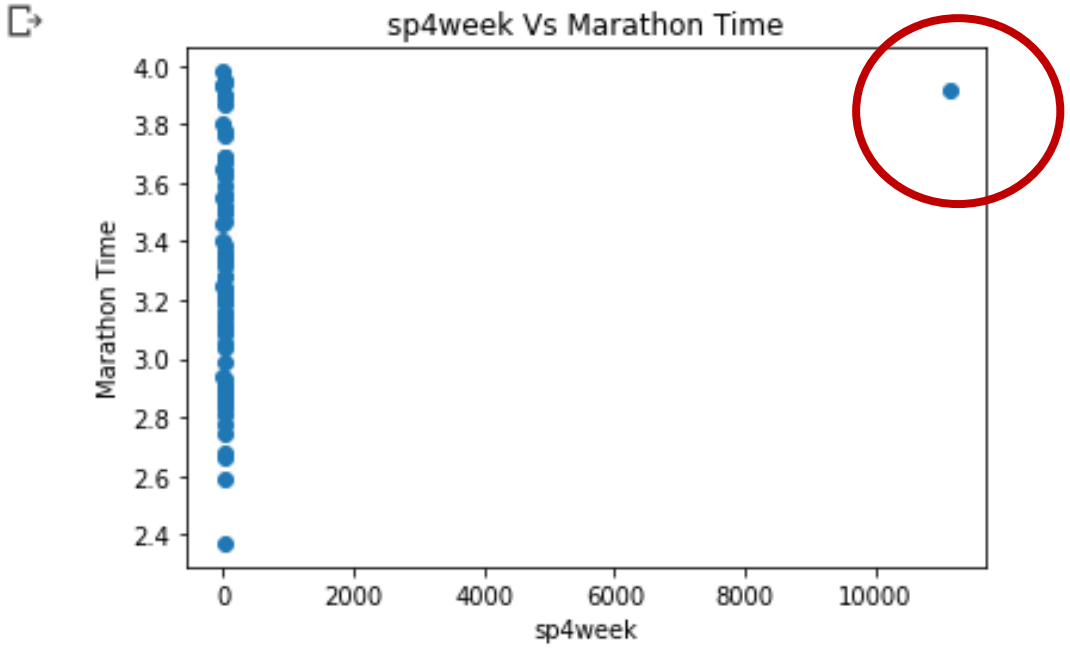
```
import matplotlib.pyplot as plt
plt.scatter(x = datos_maraton['km4week'], y=datos_maraton['MarathonTime'])
plt.title('km4week Vs Marathon Time')
plt.xlabel('km4week')
plt.ylabel('Marathon Time')
plt.show()
```



Ejemplo 1: Regresión lineal

MaratonTime vs sp4week

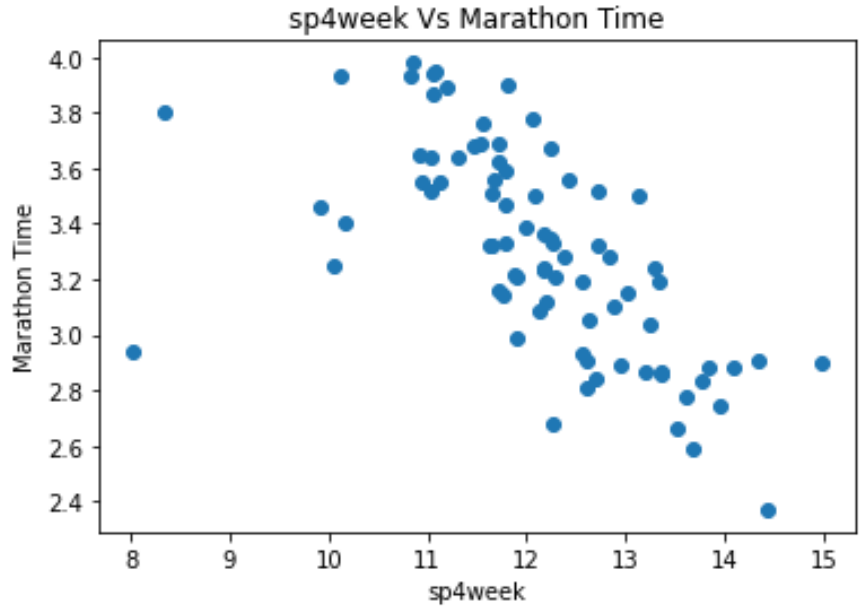
```
plt.scatter(x = datos_maraton['sp4week'], y=datos_maraton['MarathonTime'])
plt.title('sp4week Vs Marathon Time')
plt.xlabel('sp4week')
plt.ylabel('Marathon Time')
plt.show()
```



Ejemplo 1: Regresión lineal

MaratonTime vs sp4week

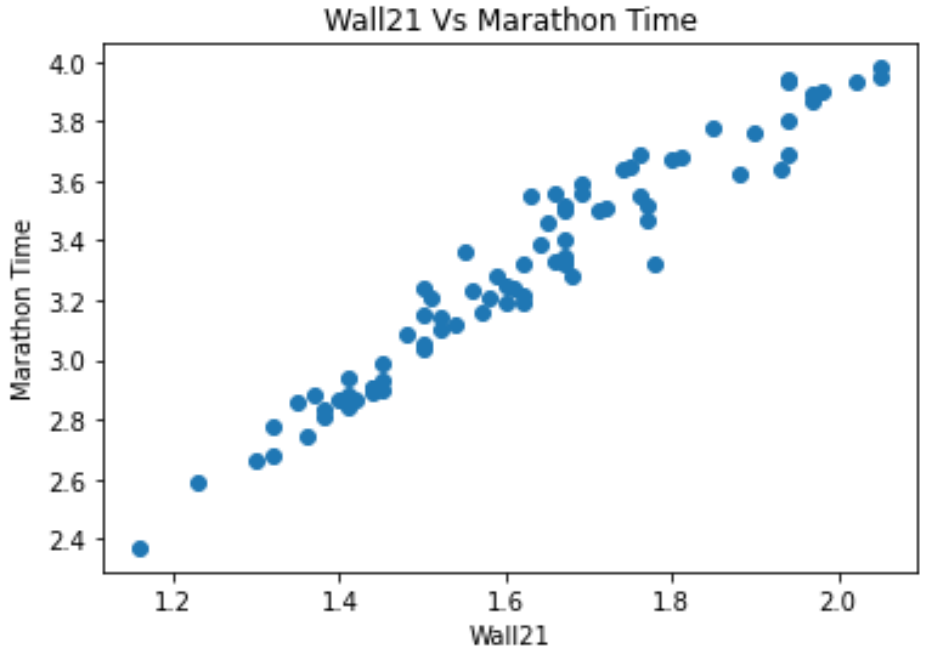
```
datos_maraton = datos_maraton.query('sp4week<2000')
plt.scatter(x = datos_maraton['sp4week'], y=datos_maraton['MarathonTime'])
plt.title('sp4week Vs Marathon Time')
plt.xlabel('sp4week')
plt.ylabel('Marathon Time')
plt.show()
```



Ejemplo 1: Regresión lineal

MaratonTime vs Wall21

```
plt.scatter(x = datos_maraton['Wall21'], y=datos_maraton['MarathonTime'])
plt.title('Wall21 Vs Marathon Time')
plt.xlabel('Wall21')
plt.ylabel('Marathon Time')
plt.show()
```



2. ENTRENAMIENTO:

b) Dividir el conjunto de datos en 2: de entrenamiento y de validación

Ejemplo 1: Regresión lineal

Se escoge un porcentaje importante para formar el conjunto de datos de entrenamiento (puede ser el 70% o el 80%) y el resto harán parte del conjunto de validación.

```
datos_entrenamiento = datos_maraton.sample(frac=0.8,random_state=0)
datos_test = datos_maraton.drop(datos_entrenamiento.index)
datos_entrenamiento
```

	Category	km4week	sp4week	CrossTraining	Wall21	MarathonTime
54	3	70.7	11.783333	0	1.77	3.47
28	2	51.6	13.008403	0	1.50	3.15
31	1	79.4	13.344538	0	1.60	3.19
84	3	55.4	11.043189	0	1.94	3.94
47	2	39.6	12.247423	0	1.67	3.35
...	...	...	...	...	...	...
55	1	26.9	13.121951	0	1.67	3.50
20	1	94.5	11.886792	0	1.45	2.99
79	1	53.9	11.802920	0	1.98	3.90
8	1	70.0	13.770492	1	1.38	2.83
13	3	84.4	13.836066	0	1.41	2.88

64 rows × 6 columns

2. ENTRENAMIENTO:

c) Separar la variable de predicción de las predictoras en cada subconjunto de datos

Ejemplo 1: Regresión lineal

Separar la variable y de las demás que conforman los subconjuntos de datos

```
[31] etiquetas_entrenamiento = datos_entrenamiento.pop('MarathonTime')
     etiquetas_test = datos_test.pop('MarathonTime')
```

```
etiquetas_entrenamiento
```

```
54    3.47
28    3.15
31    3.19
84    3.94
47    3.35
      ...
55    3.50
20    2.99
79    3.90
8     2.83
13    2.88
Name: MarathonTime, Length: 64, dtype: float64
```

Ejemplo 1: Regresión lineal

En el subconjunto de datos_entrenamiento ya no está la variable de predicción

```
datos_entrenamiento
```

	Category	km4week	sp4week	CrossTraining	Wall21
54	3	70.7	11.783333	0	1.77
28	2	51.6	13.008403	0	1.50
31	1	79.4	13.344538	0	1.60
84	3	55.4	11.043189	0	1.94
47	2	39.6	12.247423	0	1.67
...	...	...	...	...	...
55	1	26.9	13.121951	0	1.67
20	1	94.5	11.886792	0	1.45
79	1	53.9	11.802920	0	1.98
8	1	70.0	13.770492	1	1.38
13	3	84.4	13.836066	0	1.41

64 rows × 5 columns

2. ENTRENAMIENTO:

d) Ejecutar el entrenamiento

Ejemplo 1: Regresión lineal

Se utiliza la librería sklearn y se indica qué tipo de modelo es: en este caso es de Regresión lineal

```
from sklearn.linear_model import LinearRegression
modelo = LinearRegression()
modelo.fit(datos_entrenamiento,etiquetas_entrenamiento)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

3. VALIDACIÓN:

a) Hacer algunas predicciones

Ejemplo 1: Regresión lineal

Se utiliza el subconjunto de datos de prueba

```
predicciones = modelo.predict(datos_test)
predicciones
```

```
array([2.79390706, 2.81599412, 3.05747527, 3.0497715 , 3.0601308 ,
       3.29473308, 3.36211907, 3.36226188, 3.17473152, 3.29138286,
       3.37757783, 3.5216523 , 3.5919168 , 3.55016407, 3.66416024,
       3.85281097])
```

3. VALIDACIÓN:

b) Se comparan los datos de las predicciones con la etiqueta de la variable de predicción del subconjunto de validación

Ejemplo 1: Regresión lineal

Se comparan las etiquetas del test contra las predicciones

```
import numpy as np
from sklearn.metrics import mean_squared_error
error = np.sqrt(mean_squared_error(etiquetas_test, predicciones))
print("Error porcentual : %f" % (error*100))
```

```
Error porcentual : 11.030345
```

3. VALIDACIÓN:

c) Se crea un nuevo registro y se analiza cómo es el resultado

Ejemplo 1: Regresión lineal

Se agrega un nuevo corredor y se predice el resultado respecto al registro nuevo

```
nuevo_corredor = pd.DataFrame(np.array([[1,400,20,0,1.4]]),columns=['Category', 'km4week','sp4week', 'CrossTraining','Wall21'])
nuevo_corredor
```

	Category	km4week	sp4week	CrossTraining	Wall21
0	1.0	400.0	20.0	0.0	1.4

```
modelo.predict(nuevo_corredor)
```

```
array([2.199872])
```

ACTIVIDAD (TAREA)

A partir del dataset dado, realizar el mismo proceso seguido para preparar los datos. Entregar el archivo con extensión ipynb donde se evidencien las ejecuciones.

Información:

1) ID number
2) Diagnosis (M = malignant, B = benign)
    a) radius (mean of distances from center to points on the perimeter)
    b) texture (standard deviation of gray-scale values)
    c) Perimeter
    d) area

ACTIVIDAD (TAREA)

e) smoothness (local variation in radius lengths)
f) compactness (perimeter^2 / area - 1.0)
g) concavity (severity of concave portions of the contour)
h) concave points (number of concave portions of the contour)
i) Symmetry
j) fractal dimension ("coastline approximation" - 1)

REFERENCIAS BIBLIOGRÁFICAS

Sarkar, D., Bali, R. and Sharma T. (2018). Practical Machine Learning with Python. A Problem-Solver's Guide to Building Real-World Intelligent Systems

Muller, A. and Guido, S. (2017). Introduction to Machine Learning with Python