Regresión Lineal Simple: Mínimos Cuadrados

En esta sesión vamos a crear un modelo que nos permita predecir el grado de felicidad de las personas a partir de su nivel de ingresos. Decidimos que sea un modelo de regresión lineal y además como sólo tiene una feature (el nivel de ingresos) será un modelo de regresión simple.

El Problema

Se ha hecho una encuesta a 20 personas, preguntándoles por sus ingresos anuales en miles de euros y su nivel de felicidad en una escala del 0 al 10.
Los resultados obtenidos se guardan en las listas `x` (ingresos) e `y` (felicidad)

```
x = [25.2, 15.6, 26, 24, 39.2, 17.6, 3.6, 24, 10, 8.8, 35.2, 22.8,
31.6, 6, 11.2, 5.2, 22.4, 20.4, 31.2, 19.6]
y = [10, 5, 9, 8, 10, 8, 1, 9, 3, 3, 10, 5, 10, 2, 4, 4, 9, 6, 10, 7]
```

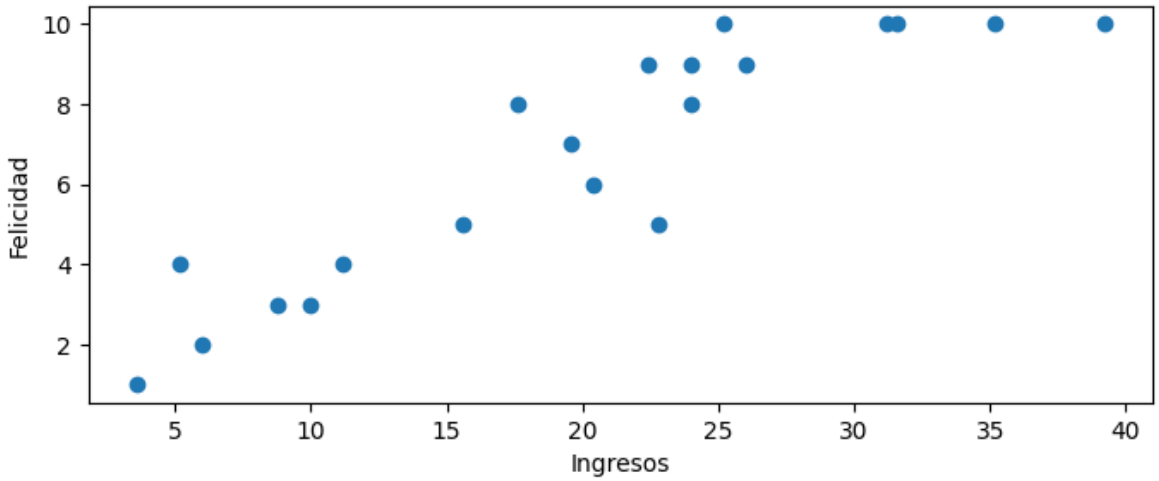
Obtener un modelo de regresión lineal, predecir la felicidad de alguien que gane 20000€

Modelado

In [1]:
```
import numpy as np
```

In [2]:
```
x = np.array([25.2, 15.6, 26, 24, 39.2, 17.6, 3.6, 24, 10, 8.8, 35.2, 22.8, 31.6
y = np.array([10, 5, 9, 8, 10, 8, 1, 9, 3, 3, 10, 5, 10, 2, 4, 4, 9, 6, 10, 7])
```

In [3]:
```
import matplotlib.pyplot as plt
plt.figure(figsize=(8,3))
ax = plt.axes()
ax.scatter(x,y)
ax.set_xlabel('Ingresos')
ax.set_ylabel('Felicidad');
```



Buscamos la recta $h(x)=w_0+w_1x$ que mejor ajuste los puntos, aplicando mínimos cuadrados. Para ello vamos paso a paso:

1. Creamos una matriz denominada con los puntos de nuestras features:

$$X=\begin{bmatrix} 1 & x_0 \\ 1 & x_1 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix}$$

In [4]:
```
X = np.array([np.ones(len(x)),
              np.array(x)]).transpose()

print(X.shape)
X
```

```
(20, 2)
```

Out[4]:
```
array([[ 1. , 25.2],
       [ 1. , 15.6],
       [ 1. , 26. ],
       [ 1. , 24. ],
       [ 1. , 39.2],
       [ 1. , 17.6],
       [ 1. ,  3.6],
       [ 1. , 24. ],
       [ 1. , 10. ],
       [ 1. ,  8.8],
       [ 1. , 35.2],
       [ 1. , 22.8],
       [ 1. , 31.6],
       [ 1. ,  6. ],
       [ 1. , 11.2],
       [ 1. ,  5.2],
       [ 1. , 22.4],
       [ 1. , 20.4],
       [ 1. , 31.2],
       [ 1. , 19.6]])
```

2. Creamos el vector de targets, con las medidas de felicidad:

$$y=\begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n\end{bmatrix}$$

In [5]:
```
y
```

```
Out[5]: array([10,  5,  9,  8, 10,  8,  1,  9,  3,  3, 10,  5, 10,  2,  4,  4,  9,
               6, 10,  7])
```

3. Resolvemos la ecuación matricial para obtener los coeficientes óptimos de la recta, los parámetros, que minimizan el MSE (recuerda que esa es nuestra función de pérdida para la regresión lineal):

$$w = (X^TX)^{-1}X^Ty$$

Usando nuestro querido numpy, ahora algo oxidado:

```
In [6]: w = np.linalg.inv(X.transpose().dot(X)).dot(X.transpose()).dot(y)
        w
```

```
Out[6]: array([1.20002504, 0.27277152])
```

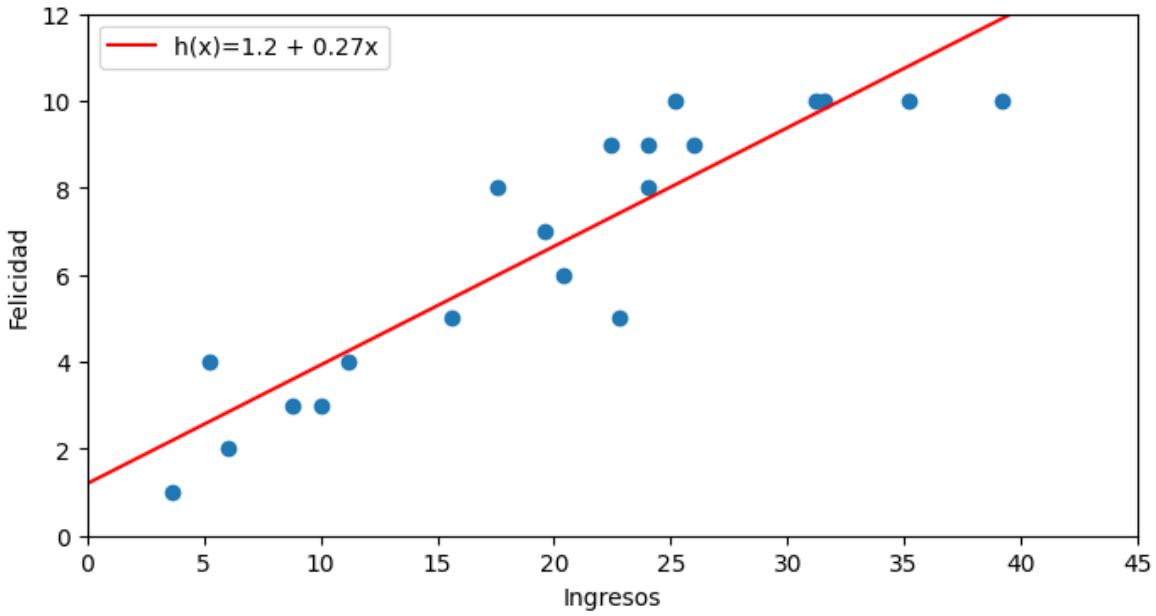
Comprobamos que la recta se ajusta a los puntos llamando a la siguiente función

```
In [7]: def plot_all(w0,w1):

            plt.figure(figsize=(8, 4))
            ax = plt.axes()
            ax.scatter(x, y)
            ax.set_ylim([0,12])
            ax.set_xlim([0,45])
            ax.set_xlabel('Ingresos')
            ax.set_ylabel('Felicidad')
            e = np.linspace(0,45,len(x))
            z = w0+w1*e

            plt.plot(e, z, '-r', label='h(x)='+str(round(w0,2))+' + '+ str(round(w1,2))+
            plt.legend(loc='upper left')
            plt.show()
```

```
In [8]: plot_all(w[0],w[1])
```



Finalmente, ¿cuál sería el nivel de felicidad de los que ganan 20.000€?

In [9]:
```
w[0] + w[1]*20
```

Out[9]:
```
6.65545543038788
```

Y la evaluación del modelo (evaluación de train, ya que ni tenemos set de validación ni de test):

In [10]:
```
from sklearn.metrics import mean_squared_error

predictions = [ (w[0] + w[1] * x_i) for x_i in x]
len(predictions), len(y)

mean_squared_error(y,predictions,squared= False)
```

Out[10]:
```
1.2079416155655123
```

Nos equivocamos en media en un 1.2 puntos. ¿Es eso bueno o malo? Es error de train y además no tenemos a quien preguntar... Aplicar entonces nuestro mejor criterio.