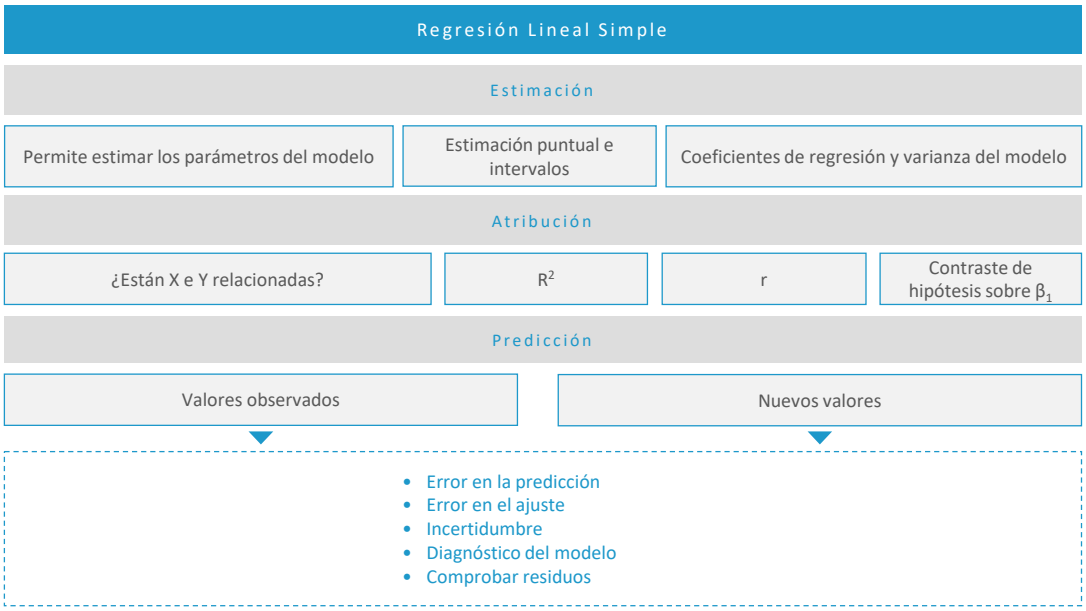
Técnicas Multivariantes

Técnicas de regresión

Índice

Esquema



Ideas clave

4.1 Introducción y objetivos

En este tema aprenderás las bases sobre las que se asientan las diferentes técnicas de regresión. Se presentará el modelo de regresión simple, se tratará la estimación de los parámetros del modelo, la atribución de la variable predictora y el empleo de las técnicas de regresión en la predicción. Por último, se pondrán en práctica los conceptos aprendidos empleando el paquete **Scikit-Learn (sklearn)** en Python.

Las técnicas de regresión es uno de los principales pilares en los que se asienta el ajuste, la estimación y la predicción estadística. Los orígenes de las técnicas de regresión datan de principios del siglo XIX, con métodos propuestos por Gauss, Lagrange y, epecialmente, por Galton en 1877. Galton estudió la dependencia de la estatura de los hijos (Y) respecto a la de sus padres (X) y descubrió que los padres altos tienen, en general, hijos altos, pero, en promedio, no tan altos como sus padres; los padres bajos tienen hijos bajos, pero, en promedio, más altos que sus padres. Este hallazgo lo denominó *regresión* a la media. A partir de ese momento, los modelos estadísticos que explican la dependencia de una variable Y respecto de una o más varias variables X se denominan modelos de regresión (Peña 1987).

Durante el siglo XX, las técnicas de regresión se fueron adaptando a las diferentes necesidades estadísticas: predicción de nuevos datos, creación de modelos de regresión y evaluación de la significatividad de las variables de predicción en dichos modelos. Además, la regresión ha servido como base de alguna de las técnicas más empleadas en la estadística clásica como son: la ANOVA, la regresión logística, la ANCOVA, los modelos lineales generalizados, etc.

Así pues, en este tema se explica cómo hacer una regresión lineal simple utilizando un ajuste mínimos cuadrados, que es una técnica de estadística clásica, que está orientada a obtener la expresión del modelo que minimice cierta función de los errores, esto es, una recta que se ajuste bien a los datos de la muestra y no tanto a extrapolar (aunque sin duda también se puede utilizar para predecir).

4.2 Regresión lineal simple

"Todos los modelos están equivocados, pero algunos son útiles".

Esta cita del estadístico británico George Box puede servir para hacernos reflexionar sobre qué son los modelos y por qué son útiles. Un modelo es una representación matemática que pretende aproximarse a la realidad (generada por los datos) y, a menudo, hacer predicciones a partir de dicha aproximación.

Uno de los modelos más sencillos que podemos formular es el modelo de regresión simple, donde una variable Y (llamada variable de respuesta, de salida, o variable dependiente) se formula a partir de una variable X (llamada variable predictora, de entrada o variable independiente), más un determinado error ε, que se distribuye de manera aleatoria. Este modelo se representa en la Ecuación 1:

$$Y = \beta_0 + \beta_1 \cdot X, \tag{1}$$

donde Y e X se corresponden con las variables y β_0 y β_1 son los parámetros de la regresión y se deben estimar a partir de los datos.

Si queremos particularizar para las diferentes observaciones de las poblaciones de X e

Y, podemos redefinir la Ecuación 1 y formularla tal y como se detalla en la Ecuación 2.

$$Y_i = \beta_0 + \beta_1 \cdot X_i + \varepsilon_i, \tag{2}$$

donde se añade la contribución del error a la fórmula. Por último, en el caso de particularizar para las distintas observaciones de una muestra de las poblaciones X e Y, representaremos el modelo de regresión muestral como se muestra en la Ecuación 3.

$$y_i = \hat{b_0} + \hat{b_1} \cdot x_i + e_i, \tag{3}$$

donde y_i es el valor de la muestra (en todo momento, y aunque no aparezca en la anotación explícitamente se sobreentiende que y_i es realmente $y_i|x_i$).

Además, se puede definir a $\hat{y}_i$ como el valor del ajuste. Es decir:

$$\hat{y}_i = \hat{b_0} + \hat{b_1} \cdot x_i. \tag{4}$$

De las Ecuaciones 3 y 4, es inmediato obtener que el error e_i es la diferencia entre el valor real y el ajustado de la observación, tal y como se muestra en la Ecuación 5.

$$e_i = \hat{y}_i - y_i. \tag{5}$$

En la Figura 1 se muestran los distintos elementos relacionados con un modelo de regresión simple.

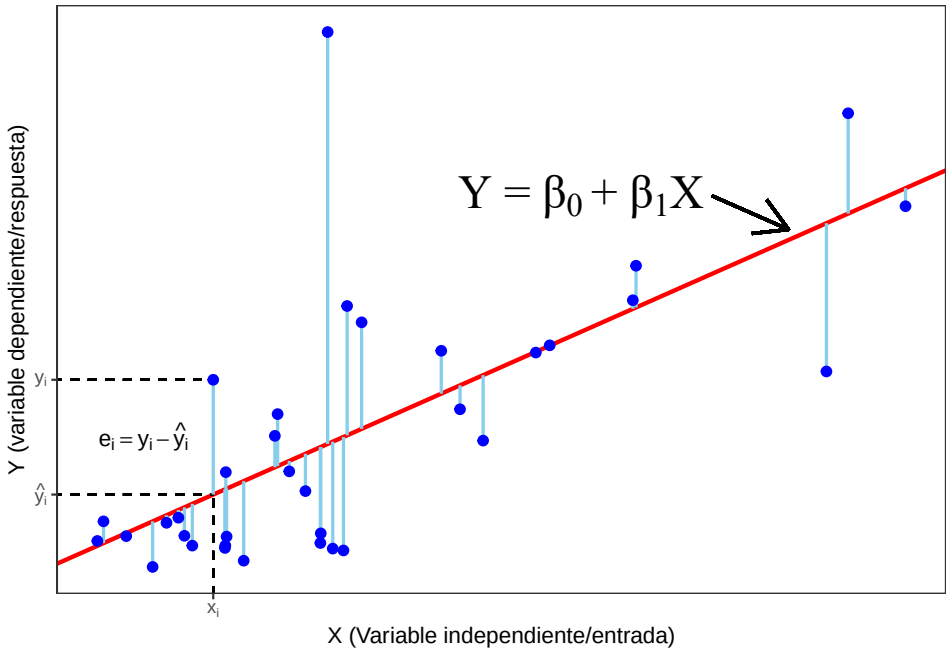


Figura 1: Elementos de un modelo de regresión simple

Una vez que se ha profundizado en las distintas anotaciones empleadas a la hora de referirnos a las ecuaciones de un modelo (en este caso de regresión lineal simple, pero extrapolable a otro tipo de modelos) es interesante hacer notar que en el contexto del aprendizaje automático nos referiremos, en la mayoría de las ocasiones, al modelo que relaciona las variables (Ecuación 1).

En los modelos de regresión lineal simple se deben comprobar los siguientes supuestos:

- El modelo debe ser lineal, por lo que no en todos los conjuntos de datos es apropiado aplicar la regresión lineal.

- Los errores o residuos se distribuyen como una variable normal de media $\mu = 0$ y desviación típica σ constante (homocedasticidad).

- Dichos errores son variables aleatorias independientes, es decir, que: $Cov(e_i, e_j) = 0,\ \forall i \neq j$.

Si alguno de estos supuestos no se cumplen, puede ser un indicador de que falta alguna

variable por incluir en el modelo, o que es necesario acudir a otro tipo de regresión más adecuada para resolver el problema planteado.

4.3 Parámetros del modelo de regresión lineal simple

El modelo de regresión tiene tres parámetros:

- Los coeficientes de regresión, β_0 y β_1, que se han introducido en la Ecuación 1. β_0 es el intercepto o interceptación de la recta de regresión y se puede expresar como la esperanza (o valor esperado) de y cuando x es igual a 0 ($E(y|x=0)$). β_1 es la pendiente de la recta de regresión e indica el cambio en la esperanza de y cuando se produce un incremento unitario en la variable x. Por lo tanto, se puede definir β_1 como:

$$\beta_1 = E(y|x+1) - E(y|x) \tag{6}$$

- El tercer parámetro del modelo de regresión es la varianza del modelo σ^2 y se determina a través de los residuos del modelo.

Causa-efecto o asociación

En muchas ocasiones, se quiere demostrar que existe causa-efecto entre una variable *de causa* o predictora X y una variable *de respuesta* Y. Sin embargo, en la mayoría de las ocasiones, cuando los datos son observacionales, tan solo se podrá demostrar que existe una asociación entre ambas variables. Para poder demostrar causalidad, es necesario emplear datos experimentales, que se obtienen a partir de estudios donde

se controlan los valores de la variable predictora (por ejemplo en ensayos clínicos). En última instancia la causalidad no termina de probarse hasta que no se halla la relación entre las variables fuera de la estadística. Por ejemplo, la longitud del radio correla positivamente con la del cúbito y el argumento es que tienen una función fisiológica común.

Mínimos cuadrados ordinarios

Uno de los procesos más habituales para estimar los parámetros β_0 y β_1 de una regresión lineal simple mediante los valores de las observaciones de una muestra es el de mínimos cuadrados ordinarios (OLS, por sus siglas en inglés), que consiste en considerar a la diferencia cuadrática entre cada observación y_i de la variable respuesta y su correspondiente valor ajustado $\hat{y}_i = \hat{b_0} + \hat{b_1} \cdot x_i$ (denotada como el estadístico q) y calcular aquellos valores de $\hat{b_0}$ y $\hat{b_1}$ que minimicen q.

$$q(\hat{b_0}, \hat{b_1}) = \sum_{i=1}^{n} (y_i - (\hat{b_0} + \hat{b_1} \cdot x_i))^2 \tag{7}$$

Resolviendo el proceso de minimización, se obtienen las fórmulas de los estimadores $\hat{b_1}$ y $\hat{b_0}$ de los parámetros β_1 y β_0 tal y como se muestran en las Ecuaciones 8 y 9 respectivamente.

$$\hat{b_1} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}, \tag{8}$$

$$\hat{b_0} = \bar{y} - \hat{b_1} \cdot \bar{x}, \tag{9}$$

siendo $\bar{x}$ e $\bar{y}$ las medias muestrales de las variables X e Y.

Con los estimadores de la interceptación y de la pendiente, queda definida la recta de regresión del ajuste. Algunas de las propiedades que presenta dicha recta de regresión son:

- La suma de todos los residuos es cero, $\sum_{i=1}^{n} e_1 = 0$.
- La suma de los valores reales de la variable dependiente Y es igual a la suma de los valores ajustados, $\sum_{i=1}^{n} y_i = \sum_{i=1}^{n} \hat{y}_i$ (y por tanto las medias también son iguales $\bar{y} = \bar{\hat{y}}$.
- La recta de regresión del ajuste del modelo siempre pasa por el punto $(\bar{x},\ \bar{y})$.

Estimación de la varianza del modelo

Se ha visto que en el modelo de regresión lineal simple existen tres parámetros que es necesario estimar. Los parámetros de regresión, que ya se han abordado, y el parámetro correspondiente con la varianza del modelo σ^2. Un estimador insesgado de la varianza del modelo σ^2 es s^2, que se define con la fórmula que se muestra en la Ecuación 10:

$$s^2 = \frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n-2} = \frac{\sum_{i=1}^{n} e_i^2}{n-2}, \tag{10}$$

donde al término del numerador de s^2 se le denomina como suma de cuadrados residual o suma de cuadrados del error y se le suele denotar como SSE.

Como nota, es interesante comentar que existen otros métodos, aparte del método de los mínimos cuadrados ordinarios, para la estimación de los parámetros de los modelos de regresión. De entre ellos destaca el método de máxima verosimilitud (MLE, por sus siglas en inglés). En este método se seleccionan los estimadores que son más compatibles con los datos. Con este método se obtienen los mismos estimadores de β_0

y $beta_1$ ($\hat{b_0}$ y $\hat{b_1}$), pero el estimador de σ^2, en este caso, se obtiene tal y como se indica en la Ecuación 11.

$$\hat{\sigma}^2 = \frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n} = \frac{\sum_{i=1}^{n} e_i^2}{n}. \tag{11}$$

4.4 Atribución

El modelo de regresión lineal permite, aparte de estimar los parámetros del modelo, evaluar si la variable predictora es significativa (con un nivel de confianza 1-α) en la predicción de la variable respuesta. Para ello, se puede calcular el intervalo de confianza (con un nivel de 1-α) de la estimación del parámetro β_1 y comprobar si dicho intervalo contiene el 0: si no lo contiene, la variable predictora X es significativa para la predicción de la variable respuesta Y; si el intervalo de confianza contiene el 0, no lo es.

La fórmula que permite calcular el intervalo de confianza (con un nivel de 1-α) se muestra en la Ecuación 12

$$b1 \pm t_{1-(\alpha/2)}(n-2) \cdot s_{b1} \quad \text{siendo } s_{b1}^2 = \frac{s^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2}, \tag{12}$$

donde $t_{1-(\alpha/2)}(n-2)$ se corresponde con la densidad de la distribución de la t de student con n-2 grados de libertad. Cuando n es grande, se puede aproximar la distribución t por la distribución normal estandarizada, y para el valor habitual de $\alpha = 0.05$, este término se puede aproximar por el valor de 1.96.

Hay que tener en cuenta, que en la mayoría de los programas estadísticos, cuando se estiman los coeficientes de regresión, se muestran los valores de la estimación puntual del error estándar del estimador (s_{b1}) y el p. valor correspondiente a la significatividad del parámetro de regresión. En estos casos, un p. valor menor de α, es equivalente a

indicar que el intervalo de confianza del estimador del parámetro al nivel de $1-\alpha$ no contiene el 0.

4.5 Predicción

La mayoría de los modelos, además de para ajustar los valores de una muestra a un modelo estadístico, sirven para poder realizar predicciones de la variable de salida o de respuesta y, a partir de nuevos valores de X. En estos casos, se puede de manera similar a la estimación de los parámetros, obtener estimaciones puntuales o intervalos de confianza de las predicciones. La predicción puntual y el intervalo de confianza de una predicción de $Y|X$ se presentan en las Ecuaciones 13 y 14 respectivamente.

$$\hat{y}_{nuevo}|x_{nuevo} = \hat{b_0} + \hat{b_1} \cdot x_{nuevo}, \tag{13}$$

$$\hat{y}_{nuevo}|x_{nuevo} \pm t_{1-(\alpha/2)}(n-2) \cdot s_{pred}, \quad \text{siendo } s^2_{pred} = s^2 \cdot (1 + \frac{1}{n} + \frac{(x_{nueva} - \bar{x})^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2}) \tag{14}$$

4.6 Descomposición de la suma de cuadrados

En cuanto a la variable dependiente, Y, la varianza de ésta se puede descomponer en un término correspondiente a la variable independiente X y un término debido al error. A esta descomposición se le conoce como suma de cuadrados y se representa

mediante la Ecuación 15.

$$\sum_{n=i}^{n}(y_i - \bar{y})^2 = \sum_{n=i}^{n}(y_i - \hat{y}_i)^2 + \sum_{n=i}^{n}(\hat{y}_i - \bar{y})^2, \tag{15}$$

que es equivalente a escribir:

$$SST = SSE + SSR, \tag{16}$$

donde:

- **SST** es la suma de cuadrados total y es una medida de la variabilidad de la muestra de Y respecto a la media muestral.
- **SSE** es la suma de cuadrados residual, o suma de cuadrados del error, y es una medida de la variabilidad de los datos de Y respecto a los valores ajustados.
- **SSR** es la suma de cuadrados de la regresión y es la parte de la varianza que explica el modelo. Es una medidad de la variabilidad de los valores ajustados respecto a la media muestral.

Para desarrollar la idea de la descomposición en suma de cuadrados, se pueden considerar los casos siguientes:

- Si y_i es igual a $\hat{y}_i$, entonces los residuos son todos cero y, por tanto, SSE = 0. Esta sería la situación ideal donde todos los valores observados de la muestra de Y se encontrarían sobre la línea de regresión y SST = SSR.
- En el otro extremo, si $\hat{y}_i$ = $\bar{y}$, el modelo ajustado no explicaría nada de la varianza de la variable dependiente y, por tanto, SST = SSE. En este caso nos encontraríamos

con el peor escenario de un modelo de regresión, donde la pendiente de la recta de regresión (β_1) sería 0 y la interceptación sería la media muestral de la variable dependiente $\bar{y}$.

Normalmente, nos encontraremos en una situación intermedia entre los dos supuestos anteriores. Además, esta descomposición en suma de cuadrados es útil para poder realizar la tabla del análisis de la varianza (tabla ANOVA), con la cual se pueden realizar diversos contrastes de hipótesis útiles en los modelos de regresión (β_1 es distinto de 0, el modelo de regresión es significativo, etc.). Por último, la descomposición de la suma de cuadrados, sirve para obtener medidas de la bondad del ajuste del modelo.

El coeficiente de determinación

En los modelos de regresión, una de las medidas de la bondad del ajuste más empleada es el coeficiente de determinación o R^2, que representa la proporción de la varianza de Y explicada por el modelo y que se puede representar como el cociente entre la suma de cuadrados de la regresión respecto a la suma de cuadrados total, o de manera equivalente, como el complementario del cociente de la suma de cuadrados del error respecto a la suma de cuadradros total, tal y como se presenta en la Ecuación 17.

$$R^2 = \frac{SSR}{SST} = \frac{SST - SSE}{SST} = 1 - \frac{SSE}{SST}. \tag{17}$$

El coeficiente de determinación es adimensional, toma valores entre 0 y 1, ya que $0 \leq SSR \leq SST$ y está relacionado con la capacidad predictiva de la variable X sobre Y. De hecho, para el caso extremo donde el ajuste $\hat{y}_i$ coincide con y_i, implicaría que $R^2 = 1$. En el otro extremo, visto en el apartado de la descomposición en suma de cuadrados, donde β_1 era igual a 0, implicaría que $R^2 = 0$. En la Figura 2 se muestran dos conjuntos de datos donde en el panel de la izquierda el coeficiente de determinación es $R \approx 0$ y en el de la derecha es $R = 1$.

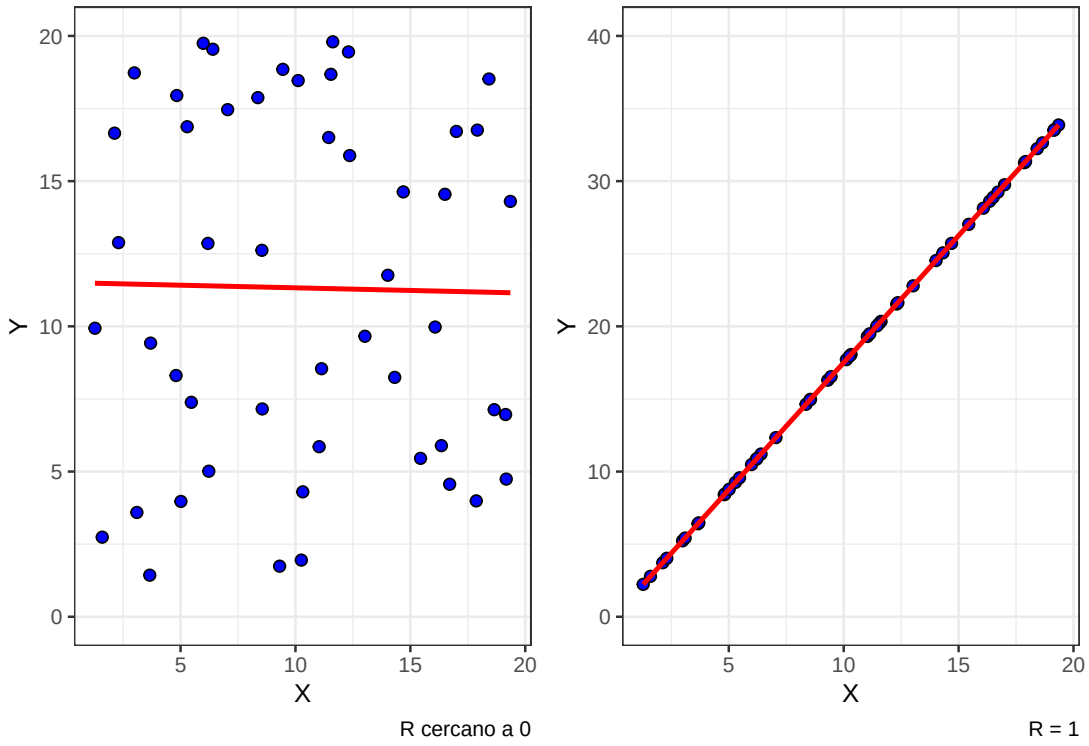


Figura 2: El coeficiente de determinación

Aunque valores altos del coefieciente de determinación pueden indicar que el modelo de regresión lineal realiza un buen ajuste de los datos, hay ocasiones en que puede no ser del todo cierto. Por ejemplo, cuando existe una relación no lineal que capture mejor la relación entre X e Y. Por otro lado, en los casos en los que este valor sea pequeño, no implica que no exista una relación entre las variables X e Y, únicamente implicaría que la relación lineal es débil.

Además, el coeficiente de determinación en un modelo de regresión lineal simple, coincide con el cuadrado del coeficiente de correlación muestral r, que se puede calcular según la Ecuación 18.

$$r = \frac{s_{xy}}{s_x \cdot s_y} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \hat{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}. \quad (18)$$

El coeficiente de correlación muestral también es adimensional, y puede tomar valores entre -1 y 1. Cuando $r = 1$, existirá una correlación muy fuerte y positiva entre ambas

variables; en cambio, si $r = -1$, esta correlación será negativa; y si $r = 0$, no existirá relación **lineal** entre las observaciones de las variables X e Y.

Es importante hacer notar que aunque los coeficientes de correlación (y los de determinación) entre X e Y y entre Y e X son iguales, los parámetros de regresión, β_0 y β_1 (y sus estimadores $\hat{b_0}$ y $\hat{b_1}$), no coincidirán si permutamos la variable dependiente por la independiente en el modelo de regresión. Se puede comprobar, que la relación entre la estimación de la pendiente de la recta de regresión $\hat{b_1}$ y el coeficiente de correlación entre X e Y está determinado según la Ecuación 19.

$$r = \hat{b_1} \cdot \frac{s_x}{s_y}. \tag{19}$$

Comprobar el modelo

Una vez que se ha realizado un modelo de regresión lineal, es necesario comprobar que el modelo es adecuado para los datos del problema. Para ello, se debe realizar un análisis de los residuos, en el que se pueden emplear algunas de las técnicas mostradas en el Tema 2 para comprobar:

- La aleatoriedad de los residuos (visualmente, con un test de rachas, etc.).
- La normalidad de los residuos (test de *Kolmogorov-Smirnov*, test de *Shapiro-Wilk*, gráfico q-q).
- La homogeneidad de la varianza.
- Detectar si algún residuo tiene un comportamiento anómalo.

Si alguno de los supuestos no se cumplen o si el coeficiente de determinación es bajo, puede ser un indicador de que el modelo regresión lineal simple no es el adecuado

para el tipo de problema abordado y es necesario buscar alternativas, como pueden ser:

- Realizar una transformación de las variables (logaritmo, potencias, transformación de Box-Cox, etc.). En este sentido, hay que distinguir en que variable se va a realizar la transformación.

  - Realizar una transformación sobre la variable predictora X en los casos en los que se cumplen los supuestos sobre los residuos, pero el ajuste lineal no sea bueno (la variable Y no se modifica, ya que podría alterar la distribución de los residuos).

  - Realizar una transformación sobre la variable respuesta Y en los casos en los que se incumple alguno de los supuestos sobre los residuos.

- Utilizar otro modelo que parezca más apropiado, aunque puede añadir complejidad al problema.

4.7 La regresión lineal simple con Python

Creación del modelo

Una vez que hemos aprendido los pormenores de la regresión lineal simple es el momento de poner los conocimientos en práctica. En Python, existen diversas librerías para realizar modelos de regresión. Nosotros, vamos a emplear la librería **Scikit-Learn (sklearn)** y más concretamente la función *linear_model.LinearRegression()*. Esta función permite definir un objeto *modelo de regresión lineal* de forma genérica. En este objeto viene definido el método *.fit()*, donde se puede indicar, la variable predictora y la variable respuesta como argumentos.

Vamos a realizar un modelo de regresión lineal sobre el conjunto de datos que se encuentra en *rls_ex1.csv*. *rls_ex1* contiene la productividad de distintos procesos industriales de diferentes empresas, donde la variable X contiene las horas de **trabajo** realizadas y la variable Y contiene el retorno económico (las **ganancias**) de dicho proceso. El primer paso a realizar cuando trabajamos en Python es cargar los paquetes y funciones que se van a utilizar (recordar que deben haberse instalado previamente).

```
# cargar librerias-----------------------------------------------
import pandas as pd
import numpy as np
from pandas.core.common import flatten
from plotnine import *
from array import *
import scipy.stats as stats
import math
import matplotlib as mpl
import matplotlib.pyplot as plt
import statsmodels.api as sm
from sklearn import linear_model
from sklearn.metrics import mean_squared_error, r2_score
import statsmodels.formula.api as smf
import statsmodels.stats.api as sms
```

Una vez cargadas las librerías necesarias, el primer paso consiste en cargar el archivo en un data.frame mediante la función *pd.read_csv()*. Cuando se explora un conjunto de datos por primera vez es conveniente realizar el método *.describe()* sobre el mismo.

```
# cargar base----------------------------------------------------
rls_ex1 = pd.read_csv("datos/procesados/rls_ex1.csv")
# describir base-------------------------------------------------
rls_ex1.describe()
```

```
##                x           y
```

```
## count    34.000000    34.000000
## mean     86.478831   523.345975
## std      60.683849   443.033484
## min      11.761794    89.704189
## 25%      48.386186   177.947704
## 50%      73.586046   367.889547
## 75%     114.434185   793.047566
## max     243.544835  1833.782283
```

Se observa que se tienen 34 observaciones y que no existen datos faltantes. Ahora, se comprueba visualmente en un gráfico de dispersión, en la Figura 3, si existe alguna relación entre X e Y.

```
# pintar x e y-------------------------------------------------
fig = plt.figure(figsize=(5,5))
plt.scatter(rls_ex1.x, rls_ex1.y)
# definir ejes-------------------------------------------------
plt.xlabel("trabajo (horas)")
plt.ylabel("ganancias (euros)")
# mostrar grafico----------------------------------------------
plt.show()
```

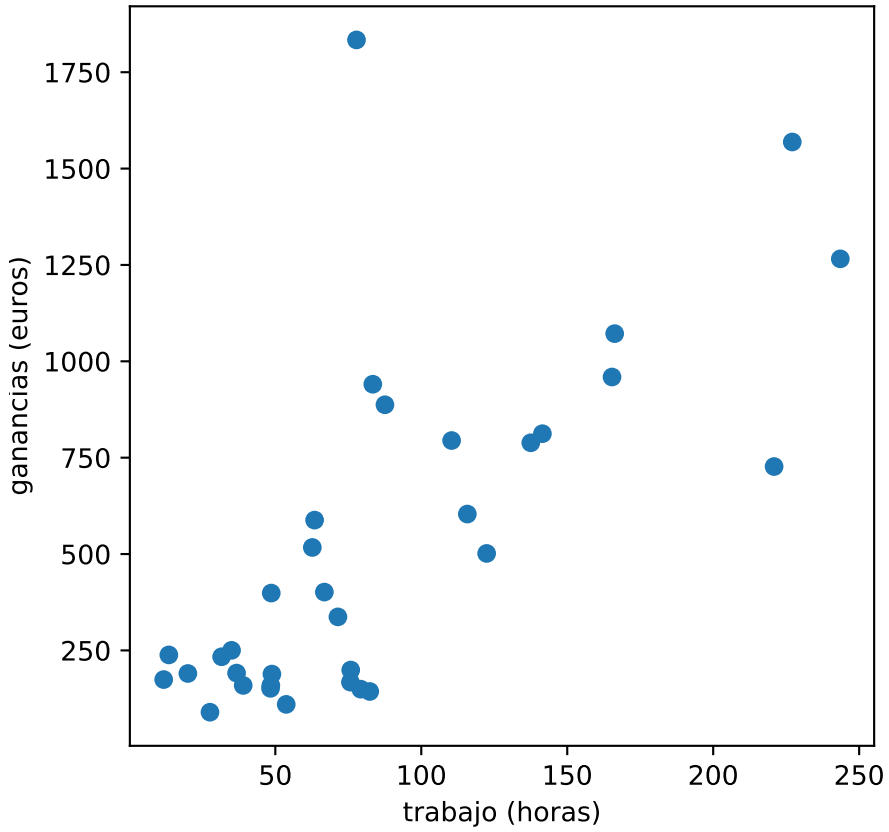


Figura 3: Relación trabajo-ganancias

Se aprecia que podría existir una correlación lineal entre ambas variables, a pesar de que hay un dato que podría ser un outlier. Se procede a ajustar un modelo de regresión lineal con el paquete *Scikit-Learn*. Para poder introducir las variables en el modelo de regresión es necesario definirlas como un array de 2 dimensiones. Esto se logra con el método *.values.reshape(-1, 1)*.

```
# extraer variables y convertir en np.array----------------------
x = rls_ex1["x"].values.reshape(-1,1)
# x.shape
y = rls_ex1["y"].values.reshape(-1,1)
# y.shape
# crear el modelo------------------------------------------------
lm1 = linear_model.LinearRegression()
```

```
lm1.fit(x, y)
# se pueden obtener los coeficientes-----------------------------
```

```
## LinearRegression()
```

```
print("Beta1: ", lm1.coef_)
# Este es el valor donde corta el eje Y (en X=0)
```

```
## Beta1:  [[5.1591244]]
```

```
print("Beta0: ", lm1.intercept_)
```

```
## Beta0:  [77.19092689]
```

Por lo tanto, el modelo de regresión lineal obtenido es:

$$Y = 77.19 + 5.16 \cdot X$$

Representación gráfica

La recta de regresión obtenida se puede introducir en la gráfica anterior, obteniéndose el resultado que se observa en la Figura 4. Hay que tener en cuenta que para poder dibujar la recta de regresión es necesario antes definir una variable que contenga los valores ajustados de y, es decir, los $\hat{y}$ del modelo. En el código se ha definido a este *array* como y_{pred}.

```
# obtener valores modelo-----------------------------------------
# pintar x e y---------------------------------------------------
fig = plt.figure(figsize=(5,5))
plt.scatter(rls_ex1.x, rls_ex1.y)
# pintar recta de regresion--------------------------------------
```

```
y_pred = lm1.predict(x)


plt.plot(rls_ex1.x, y_pred, color='red', linewidth=3)


# definir ejes----------------------------------------------------
plt.xlabel("trabajo (horas)")
plt.ylabel("ganancias (euros)")
# mostrar grafico-------------------------------------------------
plt.show()
```

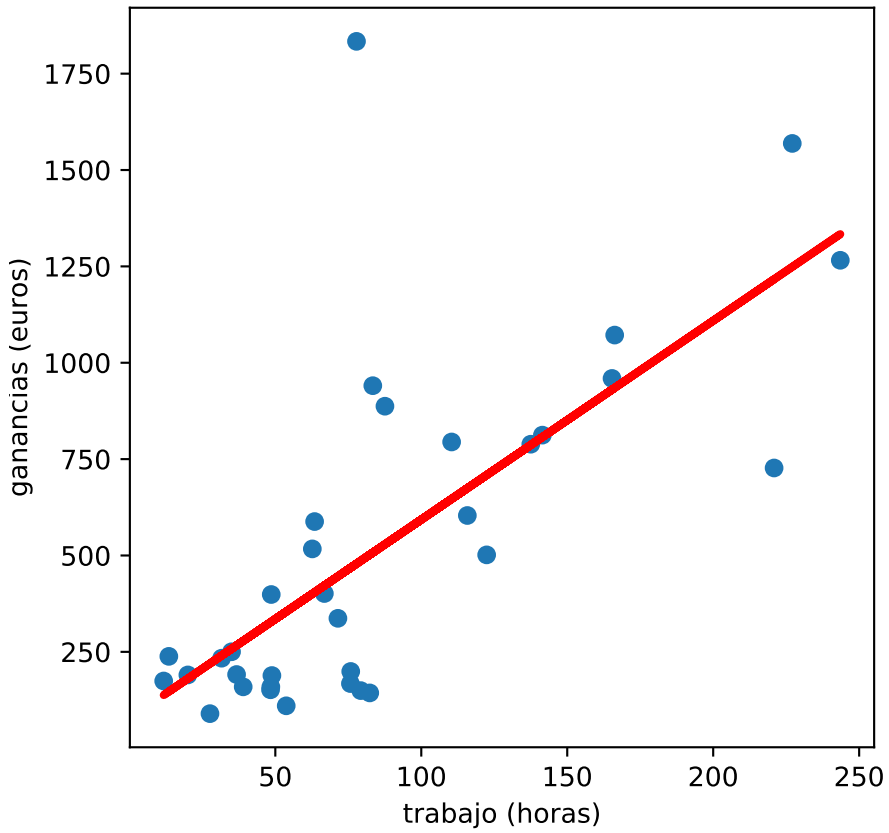


Figura 4: Modelo de regresión lineal simple trabajo-ganancias

Coeficiente de determinación

También se puede obtener el coeficiente de determinación, R^2, del modelo. El coeficiente de determinación se puede calcular mediante la función de *Scikit-Learn r2_score()*, en la cual debemos introducir como argumentos los valores reales de la muestra observada de la Variable Y, es decir, la variable (el *array*) y en Python, y la variable y_{pred}, que se ha empleado para dibujar la recta de regresión.

```
# extraer R2-----------------------------------------------------
print("Coeficiente de det. R2: %.2f"
      % r2_score(y, y_pred))
```

```
## Coeficiente de det. R2: 0.50
```

Este resultado indica que existe asociación entre la variable predictora y la variable respuesta, pero el modelo no consigue explicar, aproximadamente, la mitad de la varianza total, en términos de suma de cuadrados de la respuesta.

Varianza del modelo

Se puede obtener una estimación de la varianza del modelo, σ^2, aplicando la función mean_squared_error(), en la cual debemos introducir como argumentos, de nuevo, los valores de y e $\hat{y}$. Se puede comprobar que el estimador de la varianza del modelo obtenido es el correspondiente al método de máxima verosimilitud. Si queremos calcular un estimador insesgado de la varianza del modelo, podemos introducir la fórmula de la Ecuación 10 y resolverlo en Python. En este caso, se observa que el valor obtenido es algo mayor.

```
# calcular sigma2------------------------------------------------
print("Varianza del modelo, s2: %.2f"
      % mean_squared_error(y, y_pred))
```

```
# estimador de la varianza por MLE
```

```
## Varianza del modelo, s2: 95372.31
```

```
y_resta = (y - y_pred) ** 2
sum(y_resta)/34


# estimador de la varianza por OLS
```

```
## array([95372.31261032])
```

```
y_resta = (y - y_pred) ** 2
sum(y_resta)/32
```

```
## array([101333.08214846])
```

Valores reales versus valores ajustados

Para comprobar la bondad del ajuste del modelo, aparte del coeficiente de determinación y de la varianza, se puede realizar un gráfico (ver Figura 5) donde se comparan los valores reales de la muestra frente a los valores del ajuste. Cuanto más se aproximen los puntos del gráfico de dispersión a la bisectriz del primer cuadrante, mejor será el ajuste del modelo.

```
# pintar reales vs predichos
fig = plt.figure(figsize=(5,5))
plt.scatter(y, y_pred)
plt.xlabel("puntos reales")
plt.ylabel("puntos ajuste")
plt.plot([0, max(y)], [0, max(y)], color = 'red', linewidth = 3)
plt.show()
```

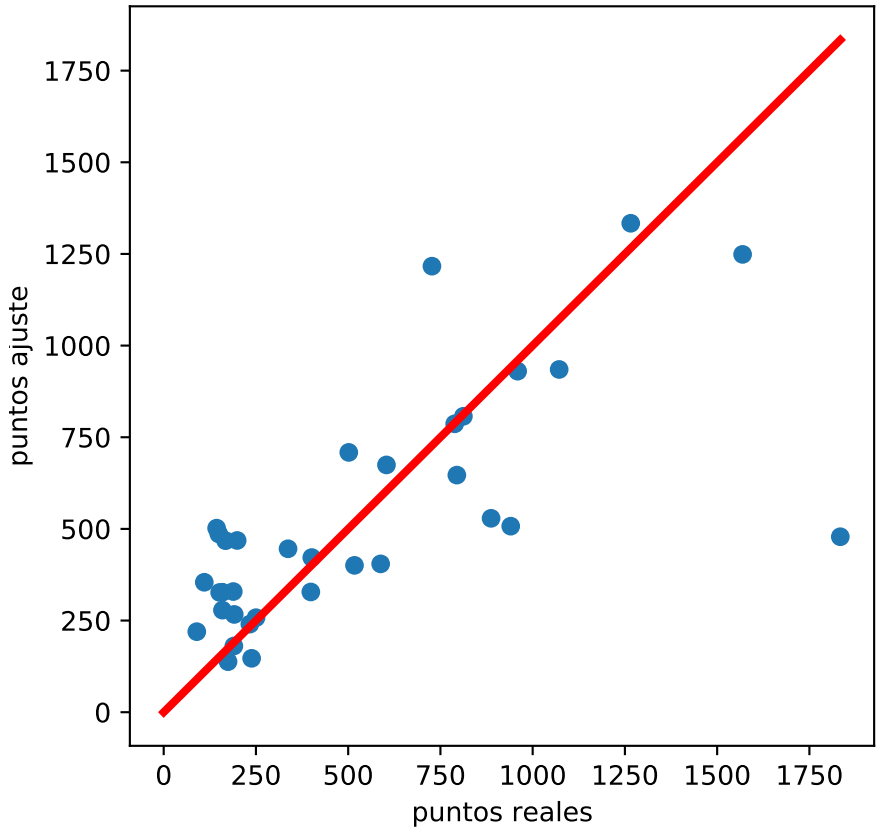


Figura 5: Valor real vs Valor ajustado

4.8 Comprobación del modelo

Por último, se realiza la comprobación del modelo. Para ello, se procede al análisis de sus residuos, representados en la Figura 6.

```
# obtener residuos----------------------------------------------
residuos1 = y - y_pred
plt.scatter(x, residuos1)
plt.xlabel("trabajo (horas)")
plt.ylabel("residuos")
plt.plot([0, max(x)], [0, 0], color = 'red', linewidth = 3)
plt.show()
```

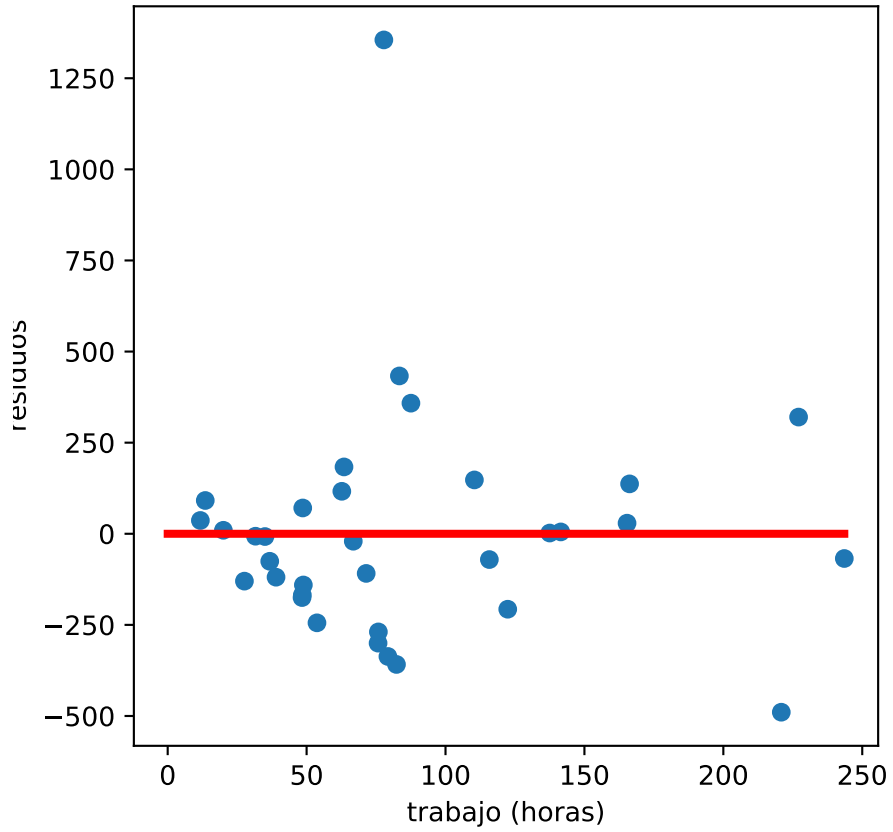


Figura 6: distribución de los residuos del modelo de regresión

Gráficamente, los residuos se distribuyen sin tendencia apreciable, pero con un *outlier* entre las 50 y las 100 horas, por lo que podemos sospechar de que se cumpla la normalidad. A continuación, se aplican los test vistos en el tema 2, la prueba de *Shapiro-Wilk*, para comprobar la normalidad; y la prueba de *Breusch-Pagan*, para comprobar la homogeneidad de la varianza.

```
# se realiza la prueba s-w-------------------------------------
sh_result = stats.shapiro(residuos1)
# dar formato a la salida--------------------------------------
print("Test Shapiro-Wilk, p.valor: %5.5f" %(
sh_result.pvalue))

# y1-----------------------------------------------------------
```

```
## Test Shapiro-Wilk, p.valor: 0.00002
```

```
m1 = sm.OLS(y, sm.add_constant(x)).fit()
bp1 = sms.het_breuschpagan(resid = m1.resid,
exog_het = m1.model.exog)[1]
print("El resultado del test Breusch-Pagan es: p.valor = %5.3f"
%(bp1))
```

```
## El resultado del test Breusch-Pagan es: p.valor = 0.834
```

Con los resultados de los test, se comprueba la homegeneidad de la varianza de los residuos, pero no su normalidad. Por lo tanto, sería necesario estudiar la procedencia del outlier para ver si se trata de un dato bien tomado o de un error grosero. Solo en este segundo caso debería eliminarse. Si no es posible eliminar el punto anómalo, cabría plantearse cambiar de modelo de regresión, realizar alguna transformación sobre la variable Y, buscar alguna otra variable predictora que fuera relevante y añadirla al modelo o aplicar algún método de regresión avanzada como los que veremos en el Tema 6.

Material audiovisual

Accede al vídeo: Regresión

4.9 Referencias bibliográficas

Geron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems.* 2 edition.

Pat Fernández, L. A. (2013). *Introducción a los modelos de regresión*. Plaza y Valdés, México, D.F.

Peña, D. (1987). *Regresión y diseño de experimentos*.

4.10 Ejercicios resueltos

Ejercicio 1.

Con este ejercicio perfeccionarás el manejo de los modelos de regresión lineal simple y profundizarás en el dataset **California housing**.

Carga el dataset **California housing**, que empezaste a analizar en el tema anterior, y realiza un modelo de regresión lineal simple de la mediana de los valores de las casas (variable respuesta Y) con cada variable numérica del dataset. Si únicamente pudieses escoger una variable para predecir el valor mediano de las casas del barrio (*median_house_value*), ¿qué variable seleccionarías?

Solución

El primer paso, como viene siendo habitual, es importar las librerías que se van a emplear en la resolución de los ejercicios:

```
# cargar librerias-----------------------------------------------
import pandas as pd
import numpy as np
from pandas.core.common import flatten
from plotnine import *
from array import *
import scipy.stats as stats
import math
import matplotlib as mpl
import matplotlib.pyplot as plt
import statsmodels.api as sm
from sklearn import linear_model
from sklearn.metrics import mean_squared_error, r2_score
import statsmodels.formula.api as smf
import statsmodels.stats.api as sms
```

Una vez que se han importado las librerías, se carga el dataset **California housing** (en el Tema 3 viene una explicación de cómo descargarlo de la web desde Python) en un dataframe.

```
# path que se va a crear en nuestro sistema----------------------
HOUSING_PATH = os.path.join("datasets", "housing")

# definir una funcion que cargue el csv en un dataframe----------
def load_housing_data(housing_path=HOUSING_PATH):
    csv_path = os.path.join(housing_path, "housing.csv")
    return pd.read_csv(csv_path)

housing = load_housing_data()
```

A continuación se realiza el ajuste de un modelo de regresión para cada una de las variables numéricas del dataset con la variable respuesta *median_house_value*. Si

recordamos del tema 3, o si volvemos a ejecutar el método *.info()* se observa que existen 8 variables numéricas (aparte de *median_house_value*).

```
# recordatorio variables numericas------------------------------
housing.info()
```

```
## <class 'pandas.core.frame.DataFrame'>
## RangeIndex: 20640 entries, 0 to 20639
## Data columns (total 10 columns):
##  #   Column              Non-Null Count  Dtype
## ---  ------              --------------  -----
##  0   longitude           20640 non-null  float64
##  1   latitude            20640 non-null  float64
##  2   housing_median_age  20640 non-null  float64
##  3   total_rooms         20640 non-null  float64
##  4   total_bedrooms      20433 non-null  float64
##  5   population          20640 non-null  float64
##  6   households          20640 non-null  float64
##  7   median_income       20640 non-null  float64
##  8   median_house_value  20640 non-null  float64
##  9   ocean_proximity     20640 non-null  object
## dtypes: float64(9), object(1)
## memory usage: 1.6+ MB
```

Longitude

Se ajusta el modelo de regresión lineal simple y se calcula el coeficiente de determinación.

```
#definir variable respuesta
y = housing["median_house_value"].values.reshape(-1,1)
# Ajuste con longitude
```

```
# extraer variable y convertir en np.array----------------------
x = housing["longitude"].values.reshape(-1,1)
# x.shape
# crear el modelo-----------------------------------------------
lm1 = linear_model.LinearRegression()
lm1.fit(x, y)
```

```
## LinearRegression()
```

```
y_pred = lm1.predict(x)

print("Coeficiente de det. R2: %.4f"
      % r2_score(y, y_pred))
```

```
## Coeficiente de det. R2: 0.0021
```

El coeficiente de determinación es muy bajo, por lo que esta variable no sería una buena predictora en un modelo de regresión lineal simple.

Latitude

Se ajusta el modelo de regresión lineal simple y se calcula el coeficiente de determinación.

```
#definir variable respuesta
y = housing["median_house_value"].values.reshape(-1,1)
# Ajuste con latitude
# extraer variable y convertir en np.array----------------------
x = housing["latitude"].values.reshape(-1,1)
# x.shape
# crear el modelo-----------------------------------------------
```

```
lm1 = linear_model.LinearRegression()
lm1.fit(x, y)
```

```
## LinearRegression()
```

```
y_pred = lm1.predict(x)

print("Coeficiente de det. R2: %.4f"
      % r2_score(y, y_pred))
```

```
## Coeficiente de det. R2: 0.0208
```

El coeficiente de determinación es algo mayor que el de longitude, pero sigue siendo muy bajo, por lo que esta variable no sería una buena predictora en un modelo de regresión lineal simple.

Housing_median_age

Se ajusta el modelo de regresión lineal simple y se calcula el coeficiente de determinación.

```
#definir variable respuesta
y = housing["median_house_value"].values.reshape(-1,1)
# Ajuste con housing_median_age
# extraer variable y convertir en np.array----------------------
x = housing["housing_median_age"].values.reshape(-1,1)
# x.shape
# crear el modelo-----------------------------------------------
lm1 = linear_model.LinearRegression()
lm1.fit(x, y)
```

```
## LinearRegression()
```

```
y_pred = lm1.predict(x)

print("Coeficiente de det. R2: %.4f"
      % r2_score(y, y_pred))
```

```
## Coeficiente de det. R2: 0.0112
```

El coeficiente de determinación es muy bajo, por lo que esta variable no sería una buena predictora en un modelo de regresión lineal simple.

Total_rooms

Se ajusta el modelo de regresión lineal simple y se calcula el coeficiente de determinación.

```
#definir variable respuesta
y = housing["median_house_value"].values.reshape(-1,1)
# Ajuste con total_rooms
# extraer variable y convertir en np.array----------------------
x = housing["total_rooms"].values.reshape(-1,1)
# x.shape
# crear el modelo-----------------------------------------------
lm1 = linear_model.LinearRegression()
lm1.fit(x, y)
```

```
## LinearRegression()
```

```
y_pred = lm1.predict(x)

print("Coeficiente de det. R2: %.4f"
```

```
      % r2_score(y, y_pred))
```

```
## Coeficiente de det. R2: 0.0180
```

El coeficiente de determinación es muy bajo, por lo que esta variable no sería una buena predictora en un modelo de regresión lineal simple.

Total_bedrooms

Se ajusta el modelo de regresión lineal simple y se calcula el coeficiente de determinación. En esta variable existen datos faltantes o NA, algo muy habitual cuando se manejan datos en el mundo real. Cuando nos encontramos ante este tipo de datos es necesario analizar cuál es la causa de que existan y, dependiendo de su posible origen, se pueden tratar de imputar o se pueden eliminar las observaciones. De momento, y para este ejercicio, eliminaremos las parejas de observaciones que tienen NA. Para ello, se utiliza el método *.dropna()* sobre el *data.frame*, donde le vamos a indicar en el argumento *axis* que se va a eliminar por filas *(axis = 0)* y que eliminaremos las observaciones que tengán algún *NA*.

```
#definir variable respuesta
housing_2 = housing.dropna(axis=0, how='any')
y = housing_2["median_house_value"].values.reshape(-1,1)
# Ajuste con total_bedrooms
# extraer variable y convertir en np.array----------------------
x = housing_2["total_bedrooms"].values.reshape(-1,1)
# x.shape
# crear el modelo-----------------------------------------------
lm1 = linear_model.LinearRegression()
lm1.fit(x, y)
```

```
## LinearRegression()
```

```
y_pred = lm1.predict(x)

print("Coeficiente de det. R2: %.4f"
      % r2_score(y, y_pred))
```

```
## Coeficiente de det. R2: 0.0025
```

En este caso, el coeficiente de determinación tampoco es bueno.

Population

Se ajusta el modelo de regresión lineal simple y se calcula el coeficiente de determinación.

```
#definir variable respuesta
y = housing["median_house_value"].values.reshape(-1,1)
# Ajuste con population
# extraer variable y convertir en np.array----------------------
x = housing["population"].values.reshape(-1,1)
# x.shape
# crear el modelo-----------------------------------------------
lm1 = linear_model.LinearRegression()
lm1.fit(x, y)
```

```
## LinearRegression()
```

```
y_pred = lm1.predict(x)

print("Coeficiente de det. R2: %.4f"
      % r2_score(y, y_pred))
```

```
## Coeficiente de det. R2: 0.0006
```

El coeficiente de determinación es muy bajo, por lo que esta variable no sería una buena predictora en un modelo de regresión lineal simple.

Households

Se ajusta el modelo de regresión lineal simple y se calcula el coeficiente de determinación.

```
#definir variable respuesta
y = housing["median_house_value"].values.reshape(-1,1)
# Ajuste con households
# extraer variable y convertir en np.array----------------------
x = housing["households"].values.reshape(-1,1)
# x.shape
# crear el modelo-----------------------------------------------
lm1 = linear_model.LinearRegression()
lm1.fit(x, y)
```

```
## LinearRegression()
```

```
y_pred = lm1.predict(x)

print("Coeficiente de det. R2: %.4f"
      % r2_score(y, y_pred))
```

```
## Coeficiente de det. R2: 0.0043
```

El coeficiente de determinación es muy bajo, por lo que esta variable no sería una buena predictora en un modelo de regresión lineal simple.

Income

Se ajusta el modelo de regresión lineal simple y se calcula el coeficiente de determinación.

```
#definir variable respuesta
y = housing["median_house_value"].values.reshape(-1,1)
# Ajuste con median_income
# extraer variable y convertir en np.array----------------------
x = housing["median_income"].values.reshape(-1,1)
# x.shape
# crear el modelo-----------------------------------------------
lm1 = linear_model.LinearRegression()
lm1.fit(x, y)
```

```
## LinearRegression()
```

```
y_pred = lm1.predict(x)

print("Coeficiente de det. R2: %.4f"
      % r2_score(y, y_pred))
```

```
## Coeficiente de det. R2: 0.4734
```

Se observa que el coeficiente de determinación es mucho mejor que todos los anteriores calculados. Así pues, si tuviesemos que escoger una única variable para realizar un modelo de regresión lineal simple, escogeríamos la variable *median_income*. No obstante, aunque sea el mejor de todos, sigue mostrando una correlación lineal bastante

débil. Una de las razones posibles es que haya que tener en cuenta varias variables en el modelo, y no una sola. Esto se llama regresión lineal múltiple y se verá en el próximo tema.

Ejercicio 2

Profundiza en la relación existente entre las variables *median_house_value* y *median_income*, obtén:

a) La ecuación del modelo de regresión lineal simple con la estimación de los coeficientes.

b) La estimación por MLE y por OLS de la varianza del modelo.

Solución

a) Se estiman los parámetros de regresión o coeficientes del modelo:

```
# se pueden obtener los coeficientes----------------------------
print("Beta1: ", lm1.coef_)
# Este es el valor donde corta el eje Y (en X=0)
```

```
## Beta1:  [[41793.8492019]]
```

```
print("Beta0: ", lm1.intercept_)
```

```
## Beta0:  [45085.57670327]
```

Por lo tanto, el modelo de regresión lineal obtenido es:

$$Y = 45085.58 + 41793.85 \cdot X,$$

representado gráficamente en la Figura 7.

```
# obtener valores modelo----------------------------------------
# pintar x e y--------------------------------------------------
fig = plt.figure(figsize=(5,5))
plt.scatter(x, y)
# pintar recta de regresion-------------------------------------
y_pred = lm1.predict(x)

plt.plot(x, y_pred, color='red', linewidth=3)

# definir ejes--------------------------------------------------
plt.xlabel("ingresos (decenas de miles de dolares)")
plt.ylabel("precio casas (dolares)")
# mostrar grafico-----------------------------------------------
plt.show()
```

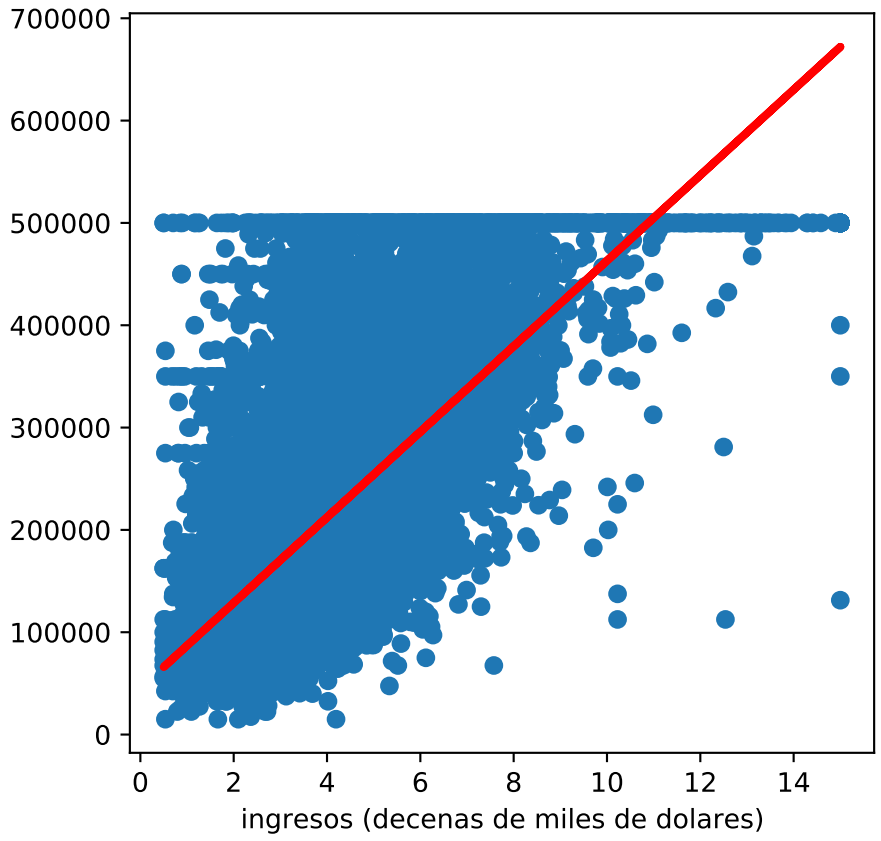


Figura 7: Modelo de regresión lineal simple

Se observa que existe una correlación positiva entre ambas variables. No obstante, el modelo de regresión lineal no parece ajustarse bien a la nube de puntos. En este ejercicio se muestra la correspondiente recta de regresión para ilustrar esto (es especialmente útil fijarse en los valores correspondientes a $X > 10$)

b) Se obtienen los valores de los estimadores de la varianza del modelo por MLE y el obtenido por OLS.

```
# calcular sigma2-------------------------------------------------
y_resta = (y - y_pred) ** 2
print(sum(y_resta)/len(y))
# estimador de la varianza por OLS-------------------------------
```

```
## [7.0113115e+09]
```

```
print(sum(y_resta)/(len(y)-2))
```

```
## [7.01199096e+09]
```

En este caso, al ser n muy grande, no existe apenas diferencia entre ambos estimadores.

Ejercicio 3

a) Parece que sí que hay relación entre *median_house_value* y *median_income*, pero ¿es estadísticamente significativa con un $\alpha = 0.05$?

b) Sí en un nuevo barrio, sabemos que los ingresos medianos son $33700\$$, ¿Qué valor cabría esperar de la mediana del precio de las casas?

Solución

a) Para comprobar la atribución de la variable en el módelo se puede realizar un contraste de hipótesis con $H_0: \quad \beta_1 = 0$, o, equivalentemente, obtener el intervalo de confianza al nivel $1 - \alpha$. Para obtener este intervalo, se hace uso de la Ecuación 12, donde se va a aproximar el valor del 95 % de la distribución t de student por 1.96 (valor de la normal estándar).

$$IC_{0.95}\ b1 = b1 \pm 1.96 \cdot s_{b1}, \text{ con } s^2_{b1} = \frac{s^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2},$$

```
# calcular ICbeta1-----------------------------------------------
# calcular numerador sb1^2
s2 = sum(y_resta)/(len(y)-2)
```

```
# calcular denominador sb1^2
den = np.var(x) * len(x)
# calcular sb1
sb1 = (s2/den) ** 0.5
amplitud = 1.96 * sb1
print("El IC al 0.95 de b1 es:", lm1.coef_, "+/-", amplitud)
```

```
## El IC al 0.95 de b1 es: [[41793.8492019]] +/- [601.33927762]
```

El intervalo de confianza al 0.95 de la estimación de β_1 no contiene al 0 y por lo tanto se puede decir que la asociación entre *median_house_value* y *median_income* es estadísticamente significativa para un $\alpha = 0.05$.

b) Para obtener el valor predictivo, debemos sustituir en la ecuación del modelo de regresión lineal el valor de X por el valor del enunciado 33.7 decenas de miles $\$$. También se puede hacer mediante el método *.predict()* del modelo de regresión. Cabe destacar que estamos utilizando la recta de regresión para hacer predicciones únicamente a modo de ejemplo, ya que en este caso no sería un modelo adecuado al tipo de datos que tenemos.

```
x_nueva = 33.7
y_nueva = lm1.intercept_ + lm1.coef_ * x_nueva
print("y nueva = %.2f" % y_nueva)

# comprobacion
```

```
## y nueva = 1453538.29
```

```
x_array = np.array(x_nueva)
y_predicha = lm1.predict(x_array.reshape(1, -1))
print("y predicha = %.2f" % y_predicha)
```

```
## y predicha = 1453538.29
```

Como era de esperar, se obtiene el mismo resultado empleando ambas estrategias: el valor mediano de las casas del barrio es de $1453538.29\ \$$.