Regresión lineal

Introducción

Desde diferentes campos como el científico, académico o empresarial, a menudo se requiere encontrar la relación entre una variable y otra(s) variable(s). Modelar esta relación entre variables mediante regresión lineal hace posible una interpretación de los datos confiable y con menos complejidad que otros modelos, logrando objetivos como encontrar información valiosa que permite tomar mejores decisiones o realizar nuevos descubrimientos.

De una manera formal, la regresión lineal podría definirse como una técnica o algoritmo que construye un modelo lineal, el cual representa una aproximación de la relación entre una variable dependiente o de respuesta con respecto a otras variables llamadas independientes o explicativas.

Hay 3 tipos de regresión lineal, la primera es la Regresión lineal simple, la segunda es Regresión lineal múltiple y la tercera es la Regresión no lineal.

La regresión lineal simple realiza un modelamiento de la relación lineal entre dos variables, por ejemplo la variable dependiente y y otra variable independendiente x. El modelo de regresión simple puede ser escrito de la siguiente manera:

$$y = \beta_0 + \beta_1 x + \epsilon$$

donde:

$y = variable\ dependiente$

$\beta_0 = valor\ de\ \ y\ \ donde\ \ x = 0\ (valor\ desconocido\ que\ se\ obtiene\ con\ el\ modelo)$

$\beta_1 = pendiente\ de\ la\ recta\ (valor\ desconocido\ que\ se\ obtiene\ con\ el\ modelo)$

$x = variable\ independiente$

$\epsilon = error$

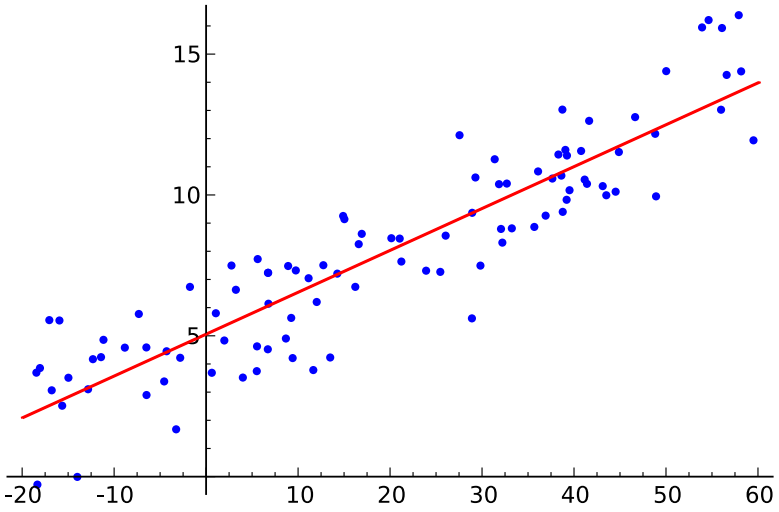


Fig.1. Ejemplo de regresión lineal simple [2]

La regresión lineal múltiple en un modelamiento de la relación lineal entre una variable dependiente y más de una independiente. La forma general de escribir la regresión lineal múltiple es así:

$$y = \beta_0 + \beta_1 x_1 + \ldots + \beta_n x_n + \epsilon$$

donde:

$y = variable\ dependiente$

$\beta_0, \beta_1, \beta_2, \ldots, \beta_n = coeficientes\ de\ regresión\ (valores\ desconocidos\ que\ se\ obtienen\ con$

$x_1, x_2, \ldots, x_n = variables\ independientes\ del\ modelo$

$\epsilon = error$

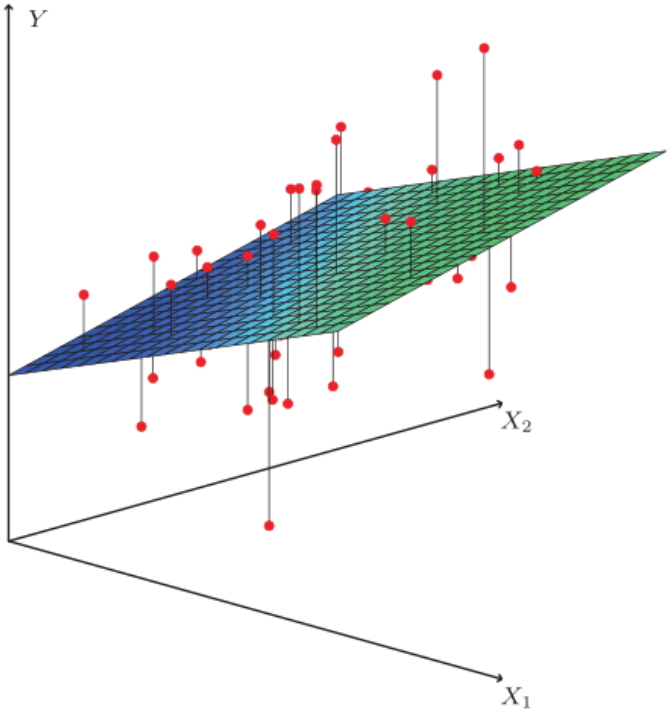


Fig. 2. Ejemplo de regresión lineal múltiple [3]

La regresión no lineal es un modelamiento donde se asume que la relación entre la variable dependiente y la(s) variable(s) independiente(s) es no lineal. Un ejemplo de regresión no lineal se puede escribir así :

$$y = \frac{\alpha}{1 + e^{\beta t}} + \epsilon$$

donde:

$y = variable\ dependiente$

$\alpha = parámetro\ del\ modelo$

$\beta = parámetro\ del\ modelo$

$t = variables\ independientes\ del\ modelo$

$\epsilon = error$

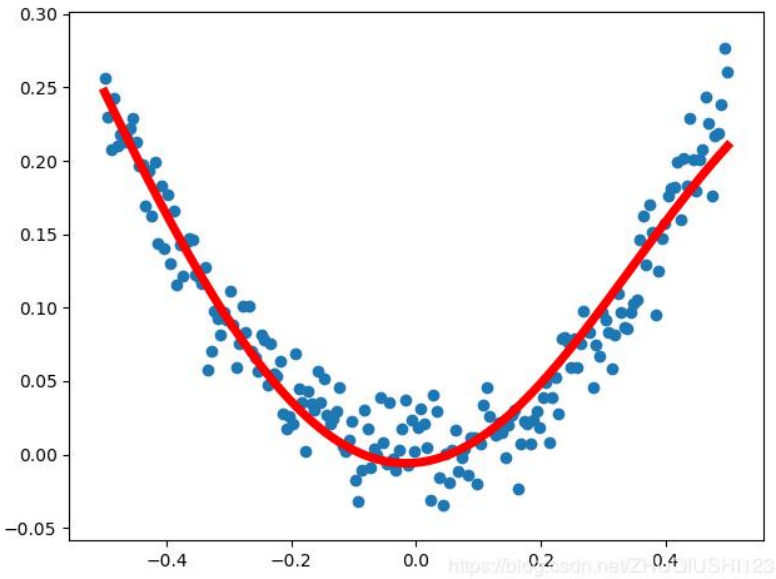


Fig. 3. Ejemplo de regresión no lineal [4]

Antes de intentar aplicar la regresión lineal, debe asegurarse de que los datos ajustarse a un modelo lineal, lo anterior se puede lograr si deben comprobarse estas hipótesis:

Las variables deben medirse a un nivel continuo. Algunas variables continuas son el tiempo, las ventas, el peso y las puntuaciones de las pruebas.
Utilice un diagrama de dispersión para determinar rápidamente si hay una relación lineal entre estas dos variables. Las observaciones deben ser independientes entre sí (es decir, no debe haber ninguna dependencia). Los datos no deberían tener valores atípicos significativos.
Comprobar la homocedasticidad, un concepto estadístico en el que las varianzas de la línea de

regresión lineal de mejor ajuste siguen siendo similares a lo largo de toda esa línea. Los residuales (errores) de la línea de regresión de mejor ajuste siguen el patrón de una distribución normal. [1]

Historia

La primera forma de regresión lineal documentada fue el método de los mínimos cuadrados que fue publicada por Legendre en 1805 [5]. Años después, en 1821 Gauss publicó en una parte de su trabajo, la demostración del teorema de Gauss-Márkov y Markov en 1900 redescubrió este teorema [6].

Fig.5. Carl Friedrich Gauss [7]

El término regresión fue introducido por Francis Galton en su libro "Natural inheritance" (1889) y fue confirmada por su amigo Karl Pearson. Su trabajo se centró en la descripción de los rasgos físicos de los descendientes (variable A) a partir de los de sus padres (variable B). Estudiando la altura de padres e hijos a partir de más de mil registros de grupos familiares, se llegó a la conclusión de que los padres muy altos tenían una tendencia a tener hijos que heredaban parte de esta altura, pero que revelaban también una tendencia a regresar a la media. Francis Galton generalizó esta tendencia bajo la "ley de la regresión universal": «Cada peculiaridad en un hombre es compartida por sus descendientes, pero en media, en un grado menor.»[8]

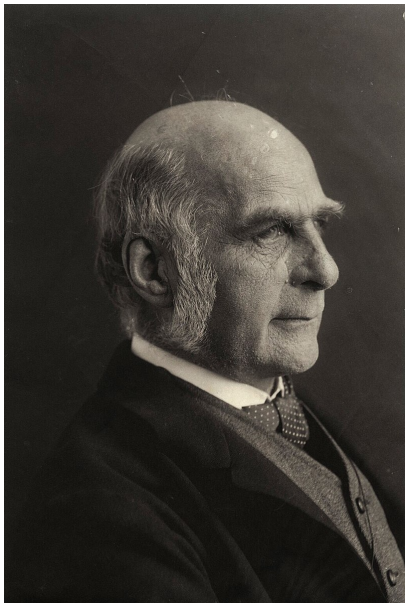
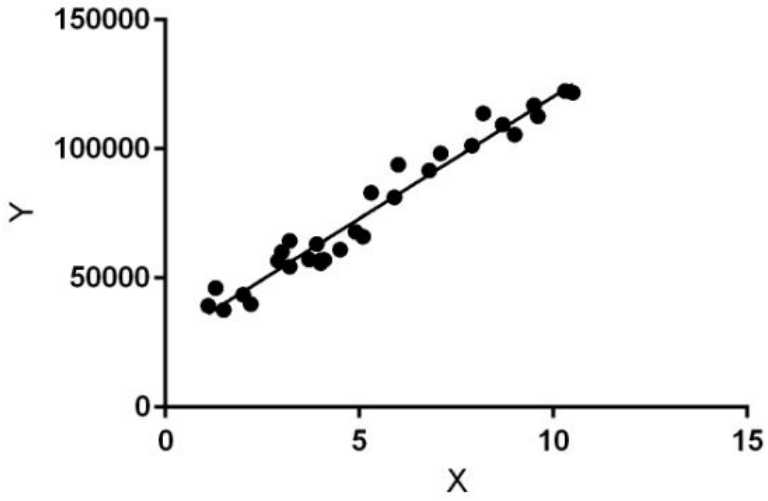


Fig.6. Francis Galton [9]

Definiciones para el algoritmo

La regresión lineal es una técnica usada en el campo del aprendizaje de maquina. Desde este campo se clasifica como un algoritmo de aprendizaje supervisado. Los algoritmos de aprendizaje supervisado requieren datos de entrenamiento y el modelo aprende por si mismo usando dichos datos.

El algoritmo de regresión lineal, específicamente regresión lineal simple, realiza la tarea de predecir el valor de la variable dependiente Y basado en la variable independiente X, por lo cual el algoritmo estima la relación lineal de la relación entre la X (la entrada) y Y (la salida). Un ejemplo aplicado de lo anterior es el siguiente:



Gráfica de regresión lineal. X representa el salario de un trabajador, Y representa los años de experiencia y la recta en los puntos representa el mejor ajuste obtenido con el algoritmo de regresión lineal

Matemáticamente esta relación lineal ente X y Y se describe como:

$$Y = \beta_0 + \beta_1 X + \epsilon$$

Donde β_0 y β_1 son los coeficientes que se obtienen con el algoritmo y ϵ representa el error cometido por el algoritmo.

De manera general el algoritmo parte de unos datos de entrenamiento conocidos como la tupla (X,y) que se pueden representar de la siguiente forma:

$$(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$$

Con estos datos de entrenamiento de tamaño n, el objetivo es estimar los coeficientes β_0 y β_1 que hagan el mejor ajuste posible al modelo lineal. Es decir que para todo x_i, y_i sea aproximadamente igual a $\beta_0 + \beta_1 x_i$.

El algoritmo recorre los datos de entrenamiento y en esas iteraciones, va actualizando y ajustando los coeficientes $\hat{\beta}_0$ y $\hat{\beta}_1$ y de está manera obtener la mejor estimación del modelo. Para lograr este objetivo, el modelo estima el valor y que minimice la diferencia entre este y del modelo y el y_i de los datos, esta diferencia se define como función de coste o error que dicho de otra manera, esta función mide que tan bien se va ajustando el modelo a los datos.

Para el caso de la regresión lineal, la función de coste más común es el **Error cuadrático medio (ECM)**, que mide el promedio de los errores al cuadrado. Formalmente se escribe como:

$$ECM(\beta_1, \beta_0) = \frac{1}{2N} \sum_{i=1}^{N} \left((y_i - (\beta_1 x_i + \beta_0) \right)^2$$

donde:
N = tamaño de los datos
y_i = valor y de los datos
$\beta_1 x_i + \beta_0$ = valor del modelo

Si $(\beta_1 x_i + \beta_0) = h_\beta(x_i)$, entonces:

$$ECM(\beta_1, \beta_0) = \frac{1}{2N} \sum_{i=1}^{N} \left((y_i - h_\beta(x_i) \right)^2$$

El valor $y_i - (\beta_1 x_i + \beta_0)$ es llamado residuo:

$$r_i = y_i - (\beta_1 x_i + \beta_0)$$

Como se mencionaba anteriormente el algoritmo requiere minimizar esta función de coste para lograr estimar el modelo. La minimización de la función de coste se realiza mediante el **Método del gradiente descendente**. De manera general el método del gradiente descendiente emplea el gradiente de la función ECM para elegir los parámetros β_0 y β_1 que desplazan el modelo a la

dirección opuesta al gradiente y de está manera reducir el EMC. El gradiente obtenido de la función ECM es:

$$\nabla f(\beta_1, \beta_0) = \begin{bmatrix} \frac{\partial f}{\partial \beta_1} \\ \frac{\partial f}{\partial \beta_0} \end{bmatrix} = \begin{bmatrix} \frac{1}{N} \sum -2x_i(y_i - (\beta_1 x_i + \beta_0) \\ \frac{1}{N} \sum -2(y_i - (\beta_1 x_i + \beta_0) \end{bmatrix}$$

El desplazamiento que se realiza se basa en la siguiente definición:

$$\beta_i^{siguiente} = \beta_i - \alpha \frac{\partial f}{\partial \beta_i}$$

α es un parámetro del modelo que se define como la tasa de aprendizaje, elegir un valor adecuado para este parámetro es crucial para una convergencia adecuada del algoritmo. Entre más pequeño sea α, más preciso será el resultado.

Algoritmo

Teniendo en cuenta las definiciones anteriores, los pasos del algoritmo son los siguientes:

1. Definir la tasa de aprendizaje con algún valor por ejemplo $\alpha = 0.5$ Establecer $\beta_0 = 0$ y $\beta_1 = 0$
2. Calcular la derivada parcial de de la función ECM con respecto a β_0 $\left(\frac{\partial f}{\partial \beta_0}\right)$ con los valores correspondientes de $x_i, y_i, \beta_0 y \beta_1$
3. Calcular la derivada parcial de de la función ECM con respecto a β_1 $\left(\frac{\partial f}{\partial \beta_1}\right)$ con los valores correspondientes de $x_i, y_i, \beta_0 y \beta_1$
4. Actualizar los valores de $\beta_0 y \beta_1$ mediante la fórmula:

$$\beta_i^{siguiente} = \beta_i - \alpha \frac{\partial f}{\partial \beta_i}$$

5. Repetir este proceso hasta que la función ECM sea muy pequeña con valores cercanos a cero, idealmente cero. Otro criterio de parada puede ser un número constante de iteraciones.

Ejemplo

Para este ejemplo se usara el siguiente Dataset Kaggle - Advertising [10] que corresponde al gasto de publicidad en televisión, en radio, en periódicos y las ventas generadas del producto. Para este conjunto de datos se miraran los datos de la columna TV que corresponden al gasto de publicidad en televisión para un producto, donde se va intentar predecir el número de ventas de un producto (columna Sales) mediante un algoritmo de regresión lineal usando gradiente descendiente.

Inicialmente obtenemos los datos que se encuentran en la ruta ./data/Advertising.csv, data set tomado de

In [8]:

```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt

advert = pd.read_csv('data/Advertising.csv')
advert.head()
```

Out[8]:

	Unnamed: 0	TV	Radio	Newspaper	Sales
0	1	230.1	37.8	69.2	22.1
1	2	44.5	39.3	45.1	10.4
2	3	17.2	45.9	69.3	9.3
3	4	151.5	41.3	58.5	18.5
4	5	180.8	10.8	58.4	12.9

Ahora extraemos los datos de las columnas correspondientes a las dos variables con las que vamos a trabajar:

$$X = Gastodepublicidadentelevisiónparaunproducto$$

$$y = Ventasdeunproducto$$

In [16]:

```
X = advert.to_numpy()[:,1]
Y = advert.to_numpy()[:,4]
x = X.reshape((-1, 1))
y = Y
```

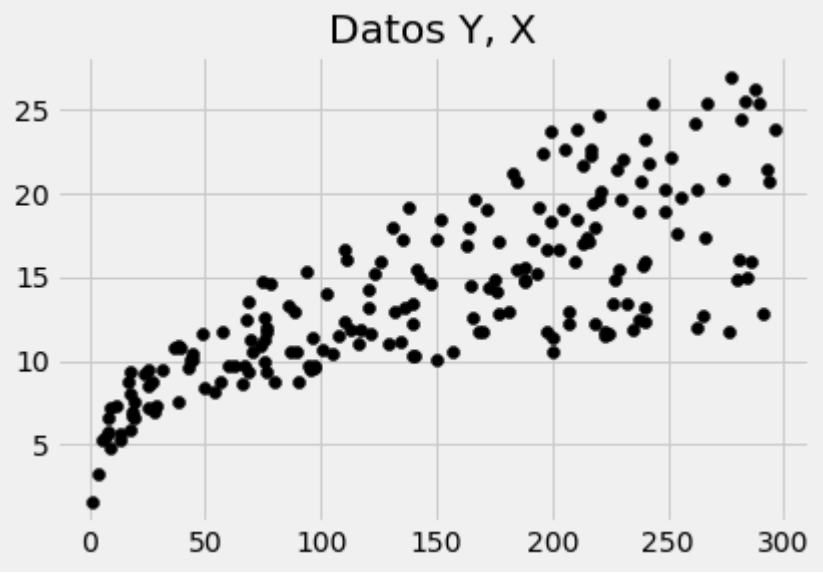
Ahora vamos a graficar los datos:

In [17]:

```
plt.style.use('fivethirtyeight')
plt.scatter(X, Y, color='black')
plt.gca().set_title("Datos Y, X")
```

Out[17]:

```
Text(0.5, 1.0, 'Datos Y, X')
```



A continuación se define un algoritmo de regresión lineal usando el método de gradiente descendiente en Python,.

In [18]:
```
class GradientDescentLinearRegression:
    h = lambda self, x, theta_0, theta_1: theta_0 + theta_1 * x
    def __init__(self, learning_rate=0.000000001, iterations=100000):
        self.learning_rate, self.iterations = learning_rate, iterations

    def fit(self, X, y):
        b_0 = 0
        b_1 = 0
        n = X.shape[0]
        for _ in range(self.iterations):
            f = np.subtract(y, self.h(X, b_0, b_1))
            b_0_gradient = - 1* f.sum() / n
            b_1_gradient = -1* X.dot(f.T).sum() / n
            b_0 = b_0 - (self.learning_rate * b_0_gradient)
            b_1 = b_1 - (self.learning_rate * b_1_gradient)
        self.b_1, self.b_0 = b_1, b_0

    def cost(self, X, y, b_0, b_1):
        n = X.shape[0]
        error = np.power(np.subtract(self.h(X, b_0, b_1), y), 2)
        return (1/(2*n)* error.sum())

    def predict(self, X):
        return self.b_1*X + self.b_0
```

Ejecución del algoritmo:

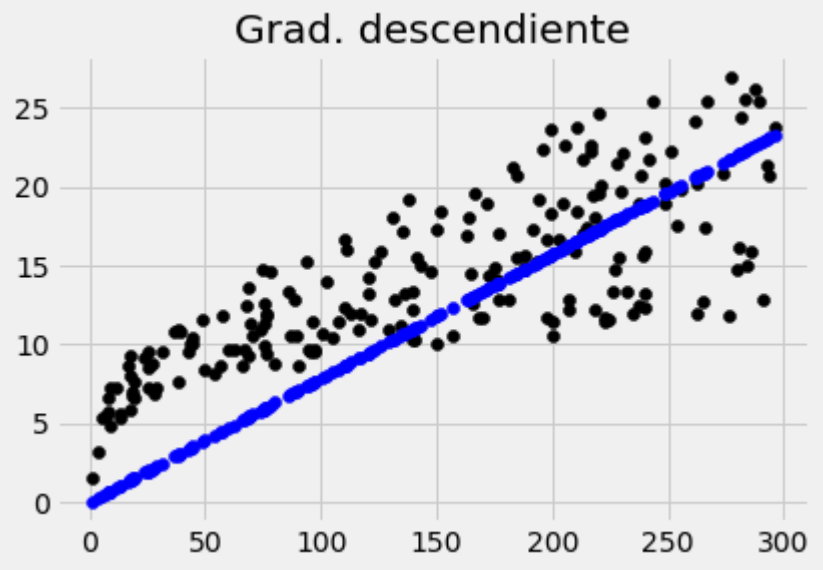
In [19]:
```
gdlr = GradientDescentLinearRegression()
gdlr.fit(X, y)
print('b_0:', gdlr.b_0)
print('b_1:', gdlr.b_1)
```

```
b_0: 0.0005775004975921139
b_1: 0.07864645898743544
```

Ahora vamos a graficar los datos de entrenamiento y la recta obtenida del modelo de regresión lineal

In [20]:
```
def plot(model, title):
    plt.style.use('fivethirtyeight')
    plt.scatter(X, y, color='black')
    plt.scatter(x, model.predict(x), color='blue')
    plt.gca().set_title(title)

plot(gdlr, "Grad. descendiente")
```



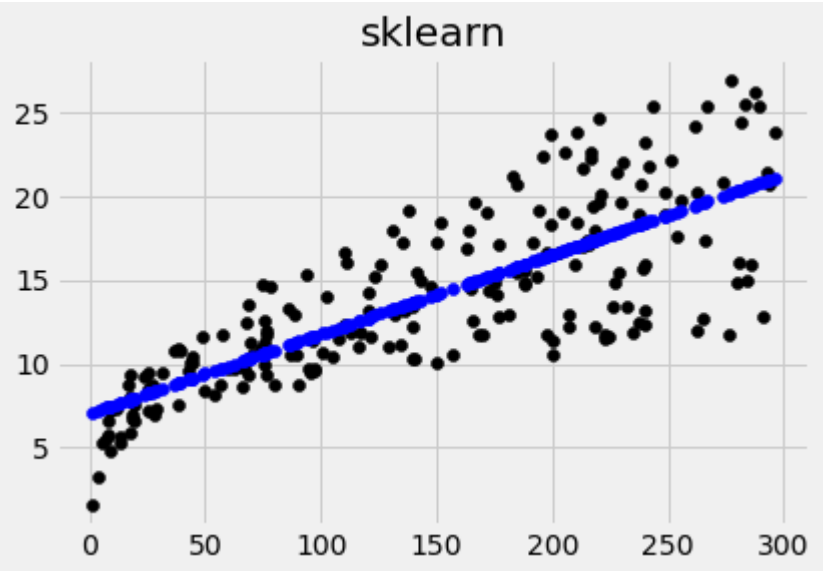
Para comparar el resultado anterior con otro modelo, se inicializa y ejecuta un algoritmo de regresión lineal de la librería sklearn.

In [21]:
```
from sklearn.linear_model import LinearRegression
sk_model = LinearRegression()
sk_model.fit(x, y)
print('b_0:', sk_model.intercept_)
print('b_1:', sk_model.coef_)
```

```
b_0: 7.032593549127693
b_1: [0.04753664]
```

Gráfica del modelo anterior:

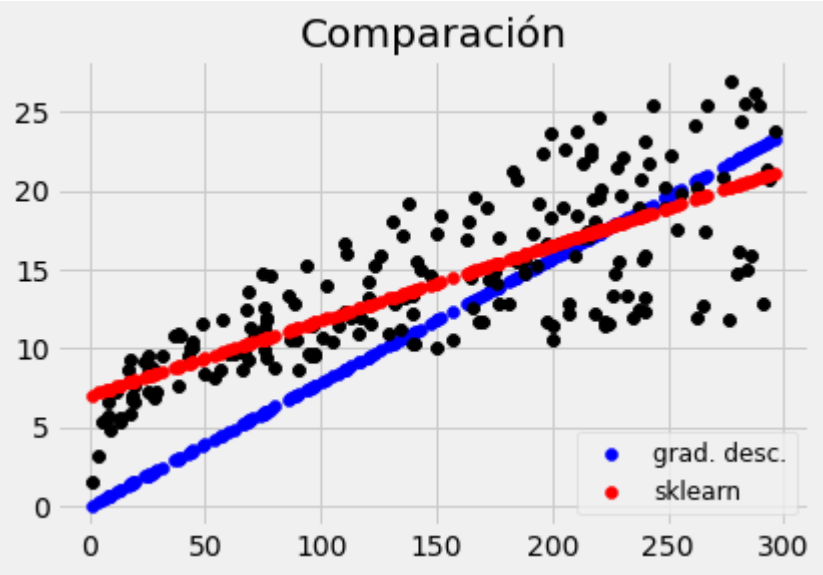
In [22]:
```
plot(sk_model, "sklearn")
```



Gráfica de comparación del los dos modelos.

In [29]:
```
plt.style.use('fivethirtyeight')
plt.scatter(X, y, color='black')
grad = plt.scatter(X, gdlr.predict(X), color='blue')
```

```
plt.style.use('fivethirtyeight')
plt.scatter(X, y, color='black')
sk = plt.scatter(x, sk_model.predict(x), color='red')
plt.gca().set_title("Comparación")
plt.legend((grad, sk),
           ('grad. desc.', 'sklearn'),
           scatterpoints=1,
           loc='lower right',
           ncol=1,
           fontsize=12)
plt.show()
```



De acuerdo a la gráfica anterior, donde la linea azul representa el modelo que usa gradiente descendiente y de rojo el modelo de sklearn, se observa que el modelo de sklearn es mucho más preciso.

Ahora vamos a realizar una predicción para un valor que no existe en los datos de entrenamiento, por ejemplo $x = 250$.

In [28]:
```
print("Modelo gradiente descendiente: " + str(gdlr.predict(np.array([[250]])))[0]
print("Modelo sklearn:                " + str(sk_model.predict(np.array([[250]]))
```

```
Modelo gradiente descendiente: 19.66219224735645
Modelo sklearn:                18.916753657382635
```

Lo anterior se interpreta que la hacer un gasto de $\$250$ en publicidad en televisión, se podrían obtener unas ventas de $\$18.91$ ó $\$19.66$

Conclusiones

La regresión lineal es un método considerablemente antiguo que en la actualidad se usa conjuntamente con las diferentes técnicas de análisis de datos modernas y se ve implementado en lenguajes de programación como Python

La vigencia de la regresión lineal en la actualidad se de principalmente por su menor complejidad con respecto a otros modelos o técnicas, resultados con una precisión considerable y su utilidad en

el análisis de datos.

El conocimiento requerido para el entendimiento de este algoritmo es principalmente en el área de matemáticas, puntualmente conceptos de calculo diferencial, funciones y álgebra lineal. Su implementación a pesar de poderse hacer con lápiz y papel, para grandes volúmenes de datos se requiere el uso de un poder de cálculo como el de un computador y para lo anterior se requiere el dominio de algún lenguaje de programación.

Bibliografía

[1] Regresión lineal (2021) [Internet], disponible en https://www.ibm.com/
[2] (Fig.2.) gráfico de regresión lineal simple (2020), [Internet], disponible en Wikipedia
[3] (Fig.3.) dlegorreta (2021), [Internet], disponible en dlegorreta - wordpress
[4] (Fig.4.) images3.programmersought.com (2021), [Internet], disponible en images3.programmersought.com
[5] Wikipedia (2021), [Internet], disponible en Wikipedia
[6] (2008) Gauss–Markov Theorem. In: The Concise Encyclopedia of Statistics. Springer, New York, NY. disponible en link.springer.com/
[7] (Fig.5.) wikimedia.org (2021), [Internet], disponible en wikimedia.org
[8] diccionario.sensagent.com (2021), [Internet], disponible en diccionario.sensagent.com
[9] (Fig.6.) wikimedia.org (2021), [Internet], disponible en wikimedia.org
[10] Kaggle - Advertising (2021), [Internet], disponible en kaggle.com