Modelos de Regresión Lineal simple con Python

Micucci, Fernanda Daniela
HCMA
2021

Objetivos generales

- *Considerar conjuntos de datos sobre los que pueda analizarse si es posible aplicar el método de Regresión Lineal Simple.*
- *Utilizar diferentes herramientas de Python (paquetes y métodos) para estudiar si se pueden encontrar modelos lineales adecuados para describir la relación entre las variables dadas.*
- *Hacer un análisis sobre los resultados obtenidos con cada método y la comparación con el método de regresión que utilizan programas gráficos (en este caso xmgrace)*

Regresión Lineal

Cuando hablamos de Regresión Lineal, nos referimos al método estadístico que busca dar un modelo (mediante el ajuste de una ecuación lineal) entre una (Regresión Simple) o varias (Regresión Múltiple) variables independientes, y una variable dependiente

A continuación se dará un contexto teórico del Modelo de Regresión Lineal Simple...

Regresión Lineal Simple

Consideremos variables aleatorias X *e* Y *y observaciones*

$$(x_i, y_i) i = 1, \ldots, n$$

El Modelo de Regresión Lineal Simple se define como:

$$Y = \beta_0 + \beta_1 x + \in$$

con $\in \sim N(0, \sigma^2)$

Regresión Lineal Simple

Al observar el conjunto de datos, se pueden estimar (de manera analítica) los parámetros β_0 *y* β_1 *utilizando el* **Método de cuadrados mínimos:**

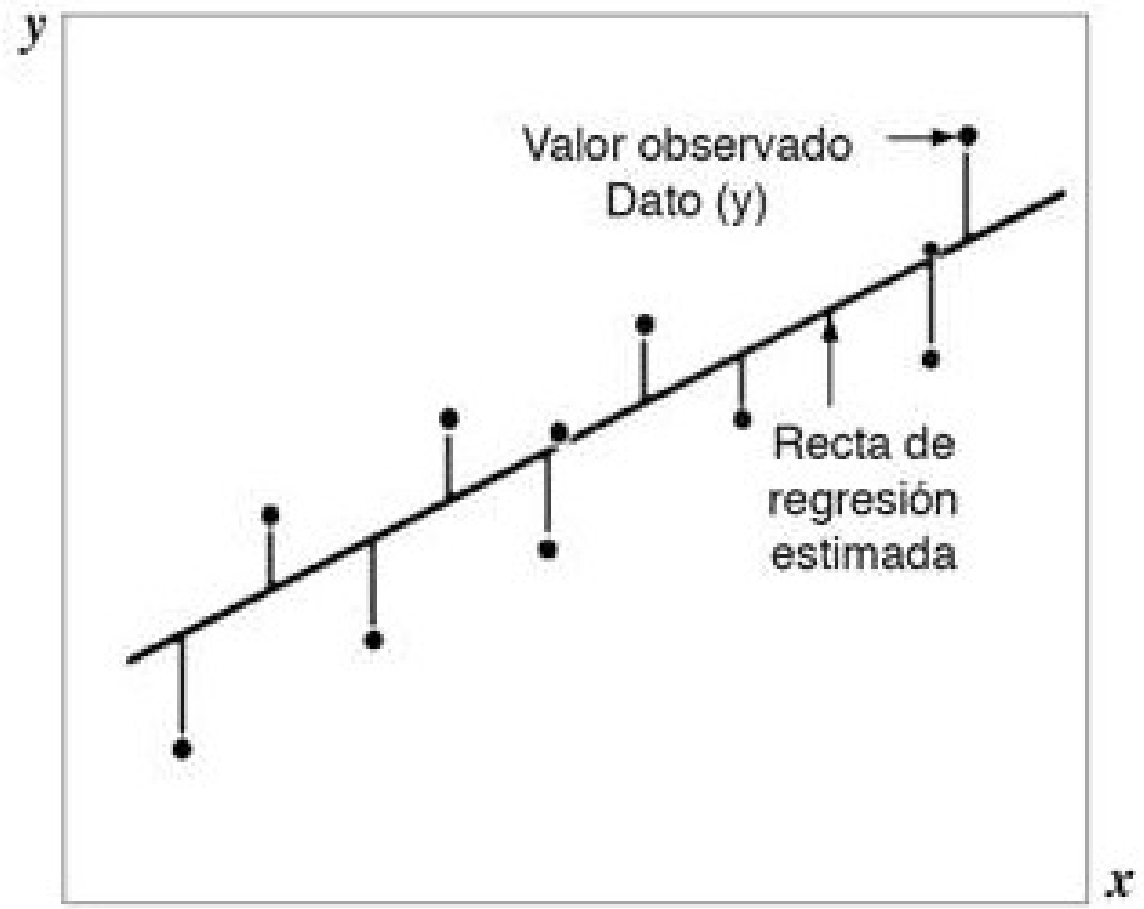
este método se utiliza para calcular la recta de regresión lineal que minimiza los residuos, esto es, las diferencias entre los valores reales y los estimados por la recta.

$$\hat{Y} = \hat{\beta}_0 + \hat{\beta}_1 x$$

$$\hat{\beta}_1 = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2}$$

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

Regresión Lineal Simple



Regresión Lineal Simple

Un vez que se encuentran las estimaciones de los parámetros y , por lo tanto, hallamos la ecuación de la recta , queremos ver qué tan bueno es el ajuste realizado; para esto se analiza el ***Coeficiente de determinación:***

$$R^2 = 1 - \frac{\sum (y_i - \hat{y}_i)^2}{\sum (y_i - \bar{y}_i)^2}$$

El coeficiente R^2 es un valor acotado entre 0 y 1, que explica la proporción de la variación total de las y_i explicada por las $\hat{y}_i$, es decir, por el modelo de regresión dado

Python y la Regresión Lineal

Al trabajar con Python tenemos distintas herramientas con las cuales intentar llegar a un modelo adecuado para explicar la relación entre las variables.
Según el tipo de método que queremos implementar, contamos con paquetes que nos permiten trabajar los datos de distintas maneras.

¡A continuación se darán ejemplos de análisis de conjuntos de datos usando distintos métodos!

Primer caso

Para este caso se utilizaron las librerías:

- ***Numpy:*** *especializada en el cálculo numérico y el análisis de grandes volúmenes de datos*
- ***stats (de scipy):*** *este subpaquete incluye funciones estadísticas y también una lista de variables aleatorias*
- ***matplotlib.pyplot:*** *es una interfaz destinada principalmente a ploteos interactivos y casos simples de generación de ploteos*

Primer caso

*En este primer análisis se estudia el conjunto de datos **“cars”** dado por el paquete Datasets de R (lenguaje con un enfoque al análisis estadístico).*
Es un conjunto bivariado con las siguientes características:

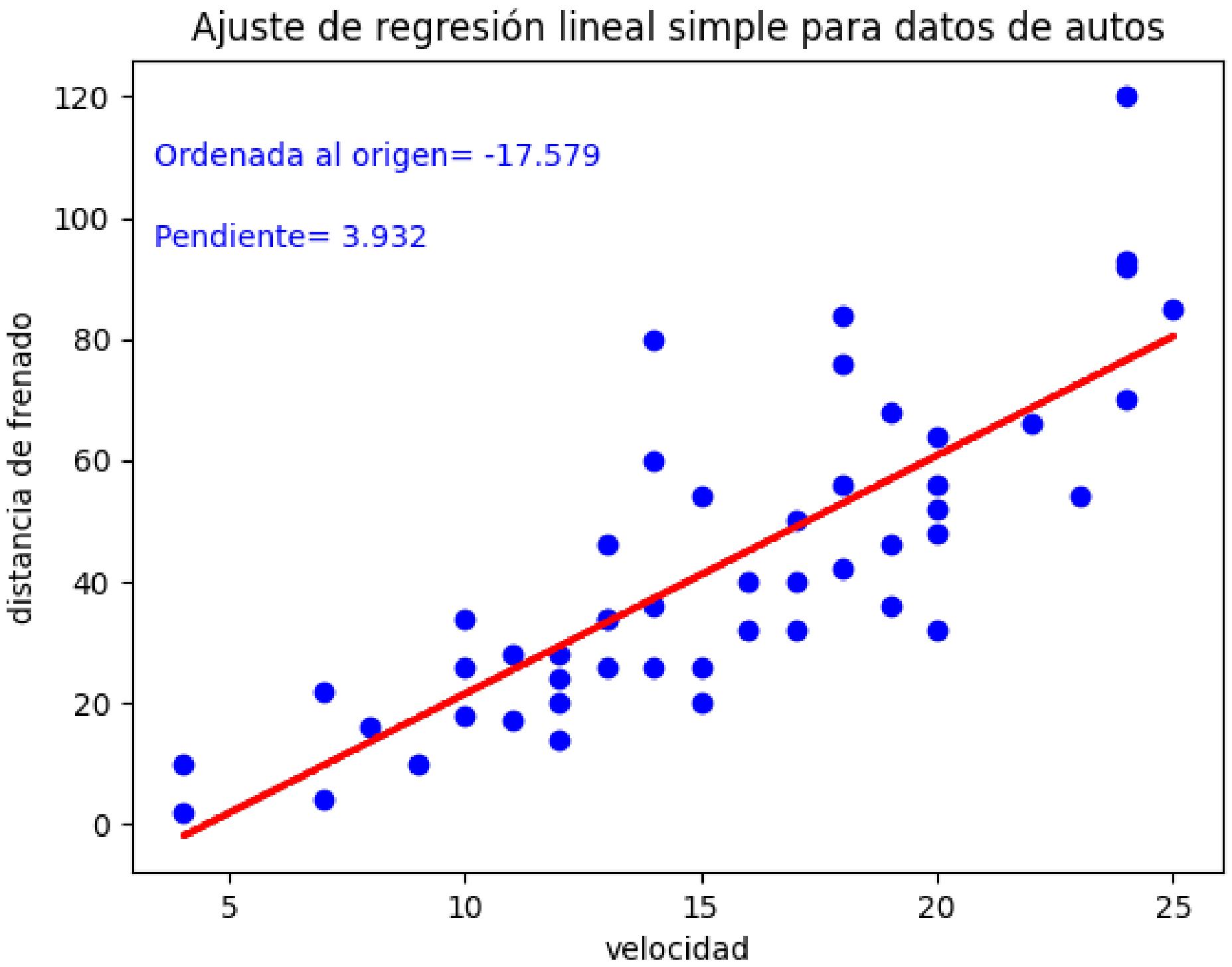
50 observaciones de 2 variables medidas en autos en venta en la década de 1920:

X: “velocidad de un auto (mph)”

Y: “distancia de frenado de un auto (metros)”

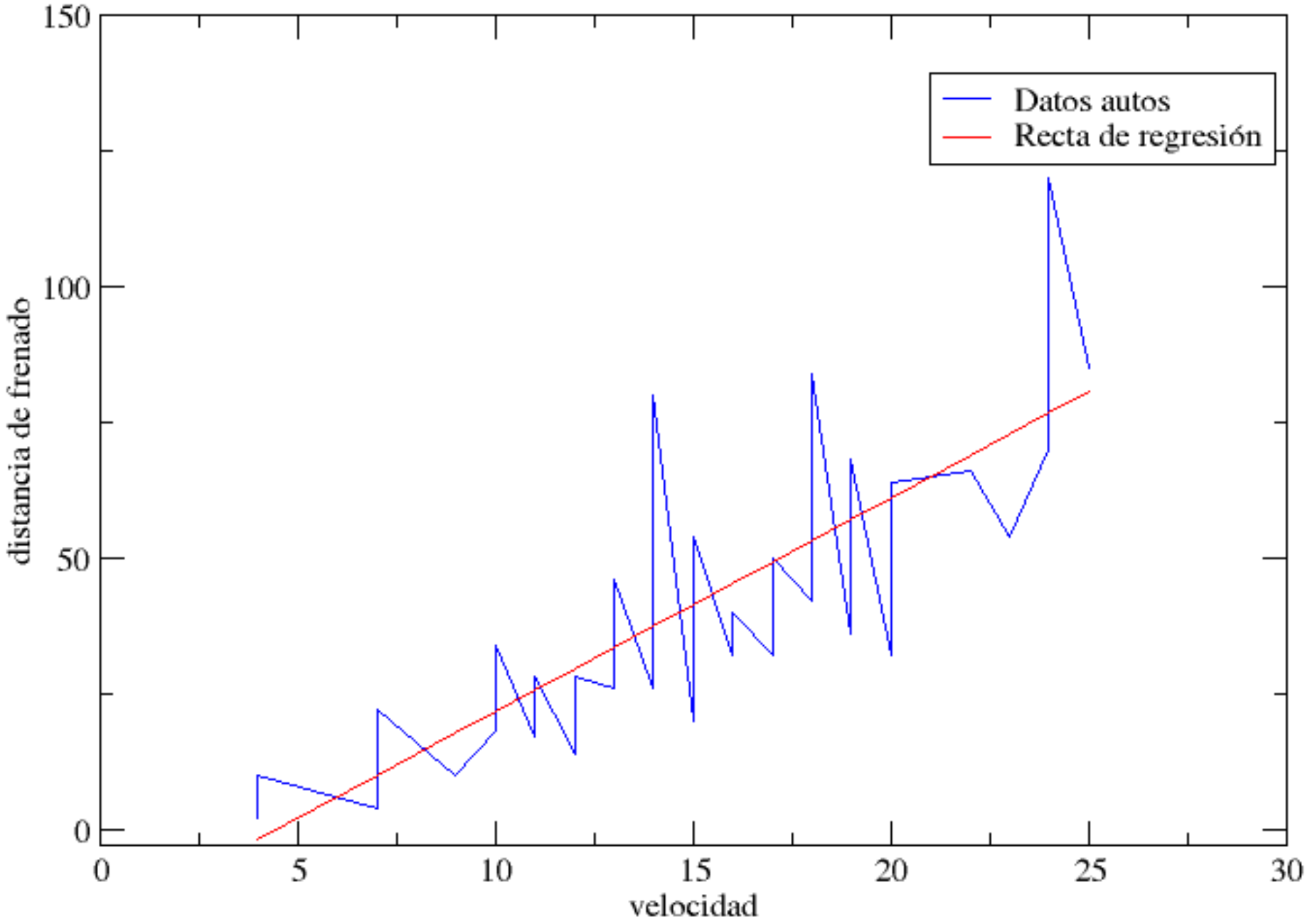
```
gradient, intercept, r_value, p_value, std_err=stats.linregress(x,y)
min_x=np.min(x)
max_x=np.max(x)

# Ahora utilizamos los datos anteriores y armamos la ecuación que estima la recta del modelo lineal:
x_fit=np.linspace(min_x,max_x,1000)
y_fit=gradient*x_fit+intercept
```

*Donde **gradient** estima la pendiente, **intercept** estima la ordenada al origen, y con **(r_value)²** obtenemos la precisión del modelo.*


Ajuste de regresión lineal simple para datos de autos
Ordenada al origen= -17.579
Pendiente= 3.932
distancia de frenado
velocidad
0
20
40
60
80
100
120
5
10
15
20
25

Primer caso

Se realizó un ajuste lineal utilizando el programa gráfico xmgrace para comparar los resultados obtenidos.


Datos autos
Recta de regresión
distancia de frenado
velocidad
0
50
100
150
0
5
10
15
20
25
30

Primer caso

Se obtuvieron los siguientes datos:

Con Python	Con xmgrace
$\hat{\beta}_0 = -17.57909$	$\hat{\beta}_0 = -17.57909$
$\hat{\beta}_1 = 3.932408$	$\hat{\beta}_1 = 3.932409$
$\hat{Y} = -17.57909 + 3.932408\,x$	$\hat{Y} = -17.57909 + 3.932409\,x$
$R = 0.80689$	$R = 0.8068$
$R^2 = 0.65107$	$R^2 = 0.6509$

Librería Scikit Learn

*La librería **Scikit Learn** (sklearn) es una de las más usadas en Python a la hora de hacer regresión lineal.*

Proporciona las siguientes clases:

** **train_test_split** se usa para estimar el rendimiento de los algoritmos de aprendizaje automático, utilizados para hacer predicciones sobre datos que no se usan para entrenar el modelo (separación datos de entrenamiento y prueba).*

** **LinearRegression** ajusta un modelo lineal para minimizar la suma residual de cuadrados entre los objetos observados en el conjunto de datos y los objetos predichos por la aproximación lineal.*

Segundo caso (sklearn)

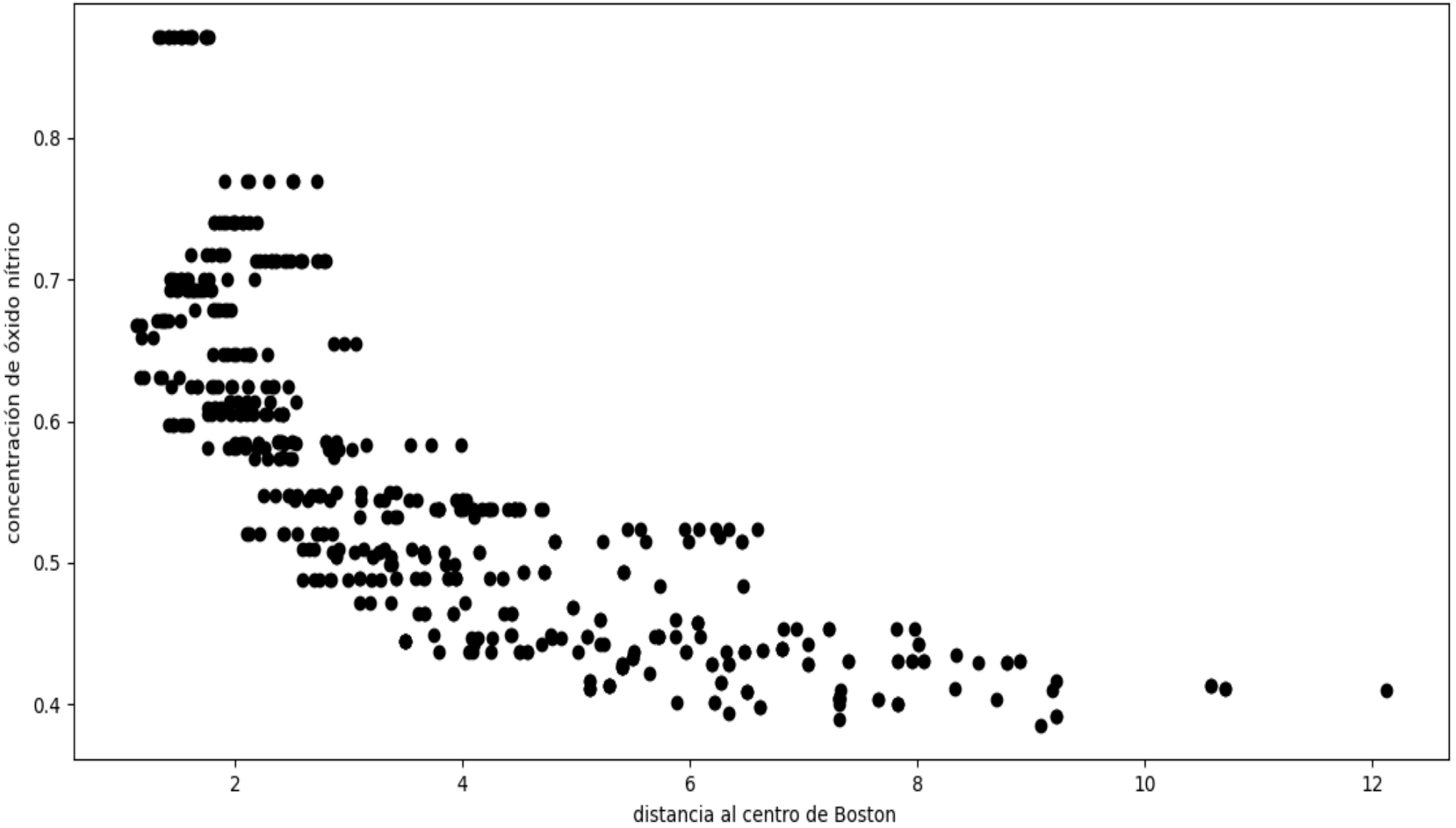
Se analizan las siguientes dos variables de las catorce medidas sobre casas de ***Boston*** *(Massachusetts)*

NOX: *"Concentración de óxidos nítricos en agua (partes por 10 millones)"*

DIS: *"Distancia a centro de empleo de Boston (Km)"*

Segundo caso (ejemplo)

```
#Hacemos la regresión:
from sklearn.model_selection import train_test_split
#Separo los datos de "train" en entrenamiento y prueba para probar los algoritmos
#(esta función divide los datos en prueba y entrenamiento para realizar la predicción)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
#utilizamos la siguiente función para la regresión
lr = linear_model.LinearRegression()
#Entreno el modelo
lr.fit(X_train, y_train)
#Realizo la predicción
Y_pred = lr.predict(X_test)
```


concentración de óxido nítrico
0.8
0.7
0.6
0.5
0.4
2
4
6
8
10
12
distancia al centro de Boston

Segundo caso

Tamaño del test	Recta ajustada	Precisión del modelo
0.05	$\hat{Y} = -0.04187\,x + 0.712045$	*0.59497*
0.1	$\hat{Y} = -0.042135\,x + 0.714556$	*0.58716*
0.2	$\hat{Y} = -0.04188\,x + 0.71476$	*0.59341*
0.5	$\hat{Y} = -0.04365\,x + 0.7195$	*0.5746*
0.9	$\hat{Y} = -0.0483\,x + 0.7564$	*0.6404*

Segundo caso

Al hacer estas predicciones, observamos que para varias ejecuciones con el mismo tamaño del test no obtuvimos exactamente la misma recta ni la misma precisión (aunque las diferencias son mínimas).

Lo vemos para el caso del tamaño del test 0.9 con 5 ejecuciones:

Segundo caso

$$\hat{Y} = -0.02987\,x + 0.6644 \qquad R^2 = 0.541968$$

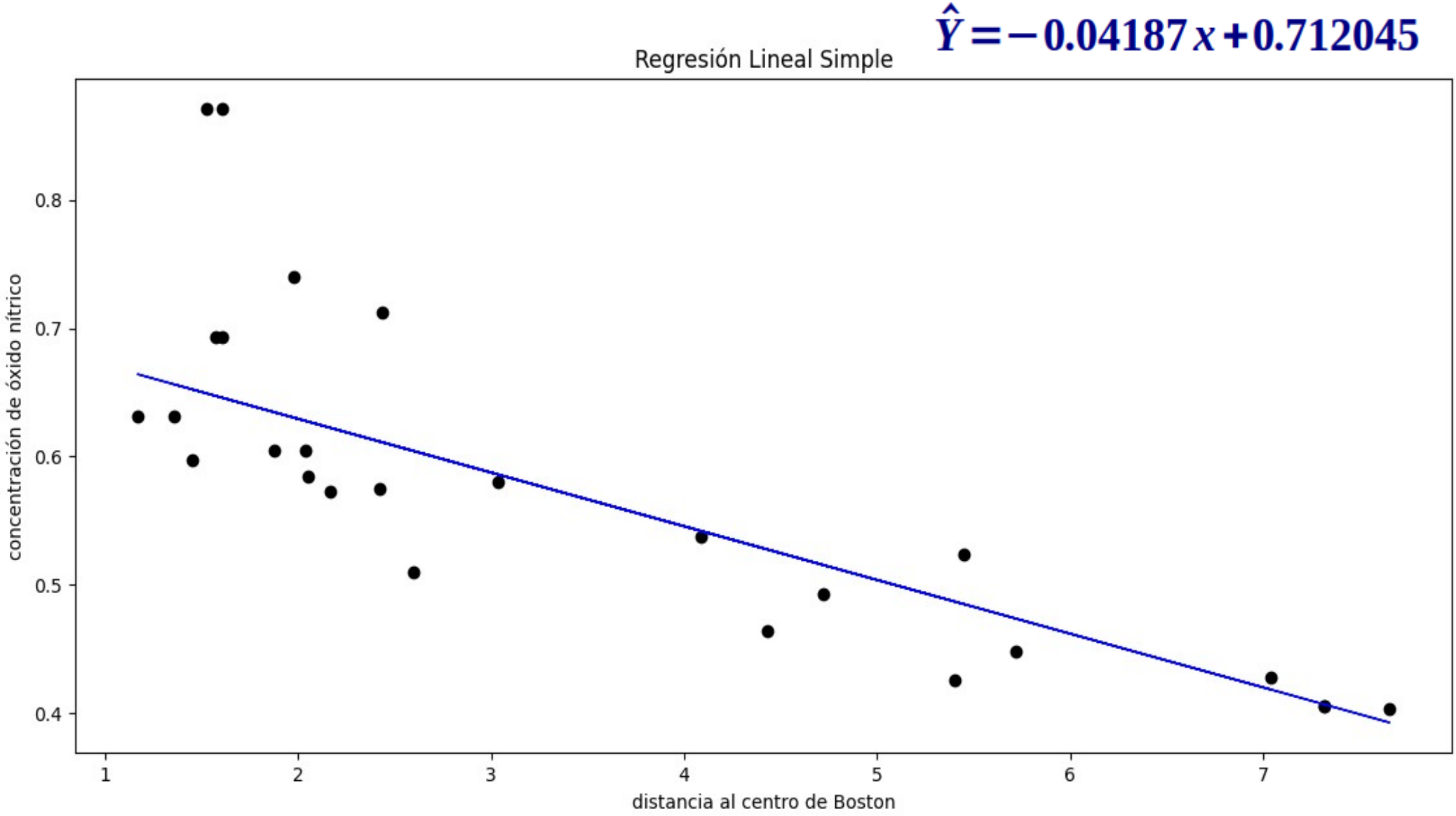
$$\hat{Y} = -0.04245\,x + 0.7080 \qquad R^2 = 0.65287$$

$$\hat{Y} = -0.04274\,x + 0.7254 \qquad R^2 = 0.69759$$

$$\hat{Y} = -0.04756\,x + 0.7475 \qquad R^2 = 0.6195$$

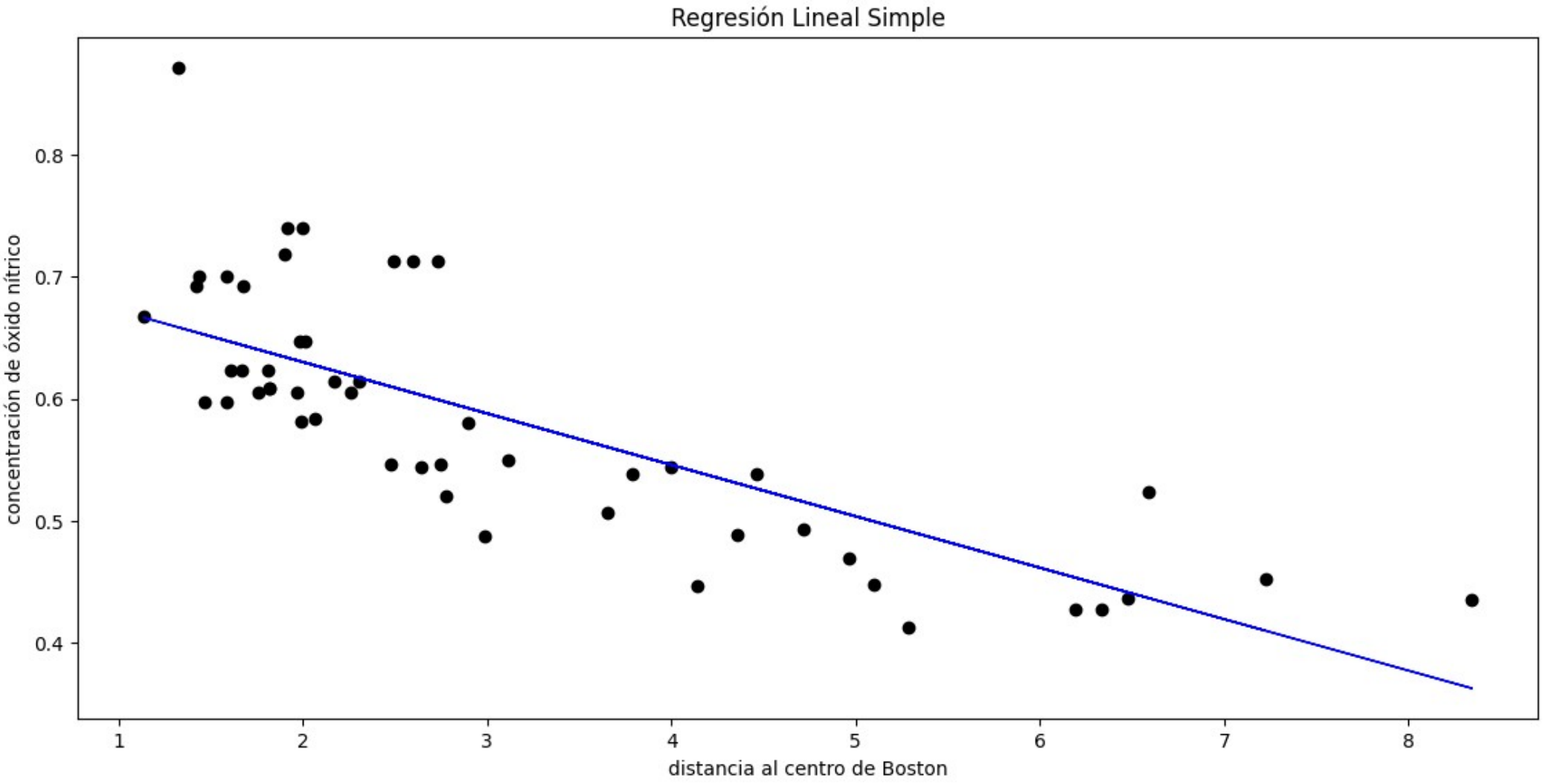
$$\hat{Y} = -0.03809\,x + 0.69396 \qquad R^2 = 0.6596$$

tamaño 0.05


$\hat{Y} = -0.04187\,x + 0.712045$
Regresión Lineal Simple
concentración de óxido nítrico
distancia al centro de Boston
0.8
0.7
0.6
0.5
0.4
1
2
3
4
5
6
7

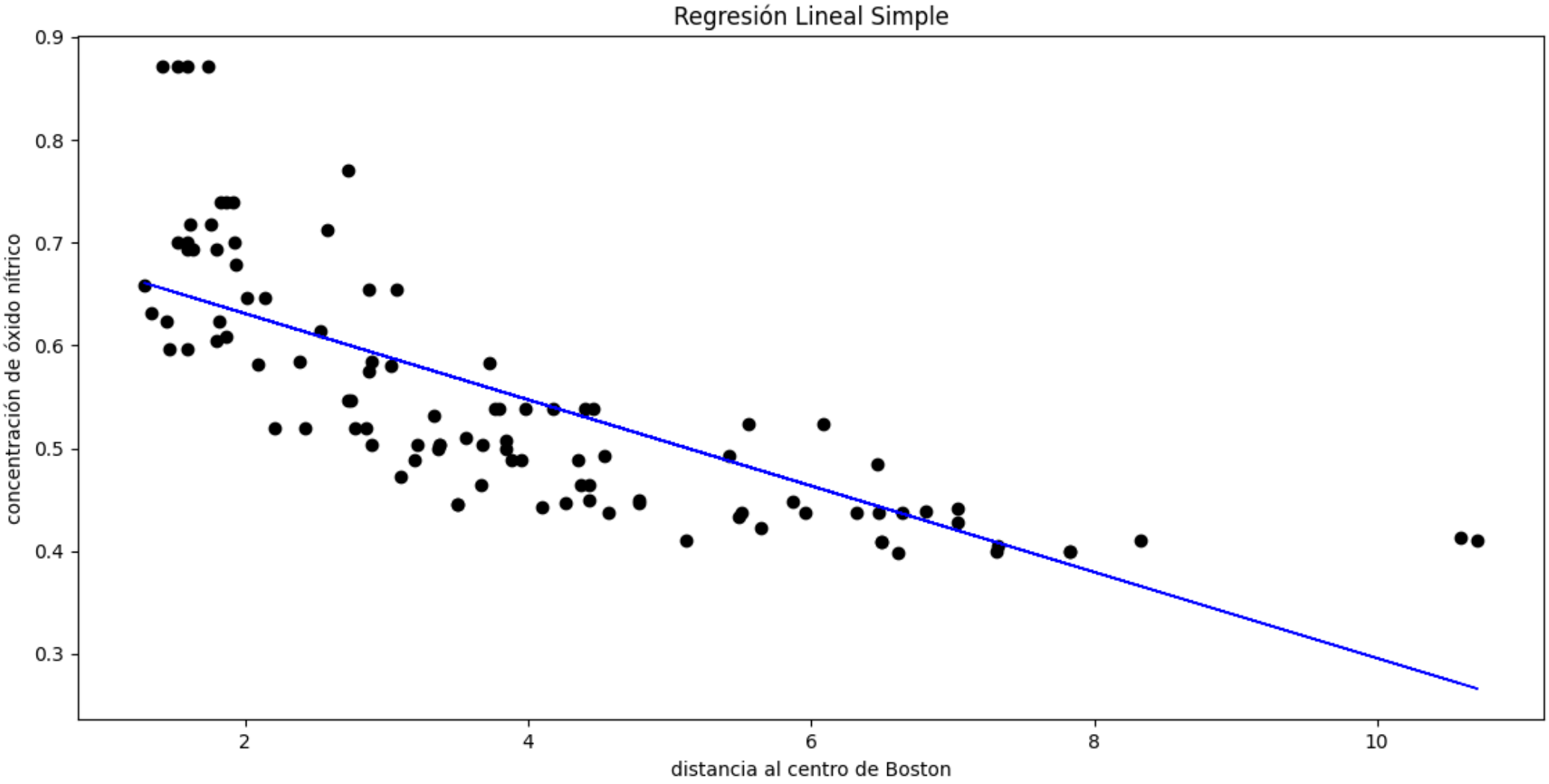
tamaño 0.1

$$\hat{Y} = -0.042135\,x + 0.714556$$



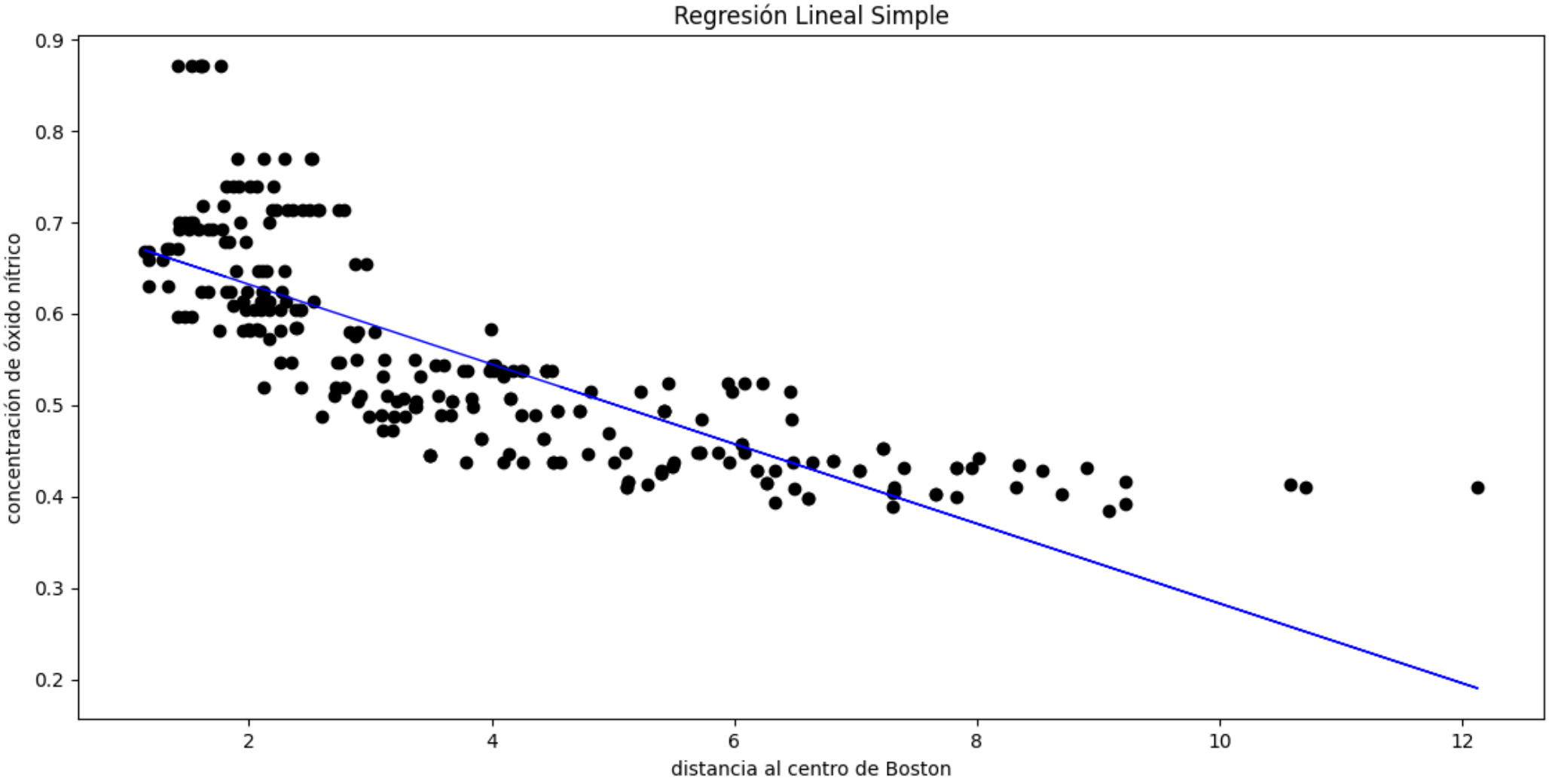
tamaño 0.2

$$\hat{Y} = -0.04188\,x + 0.71476$$



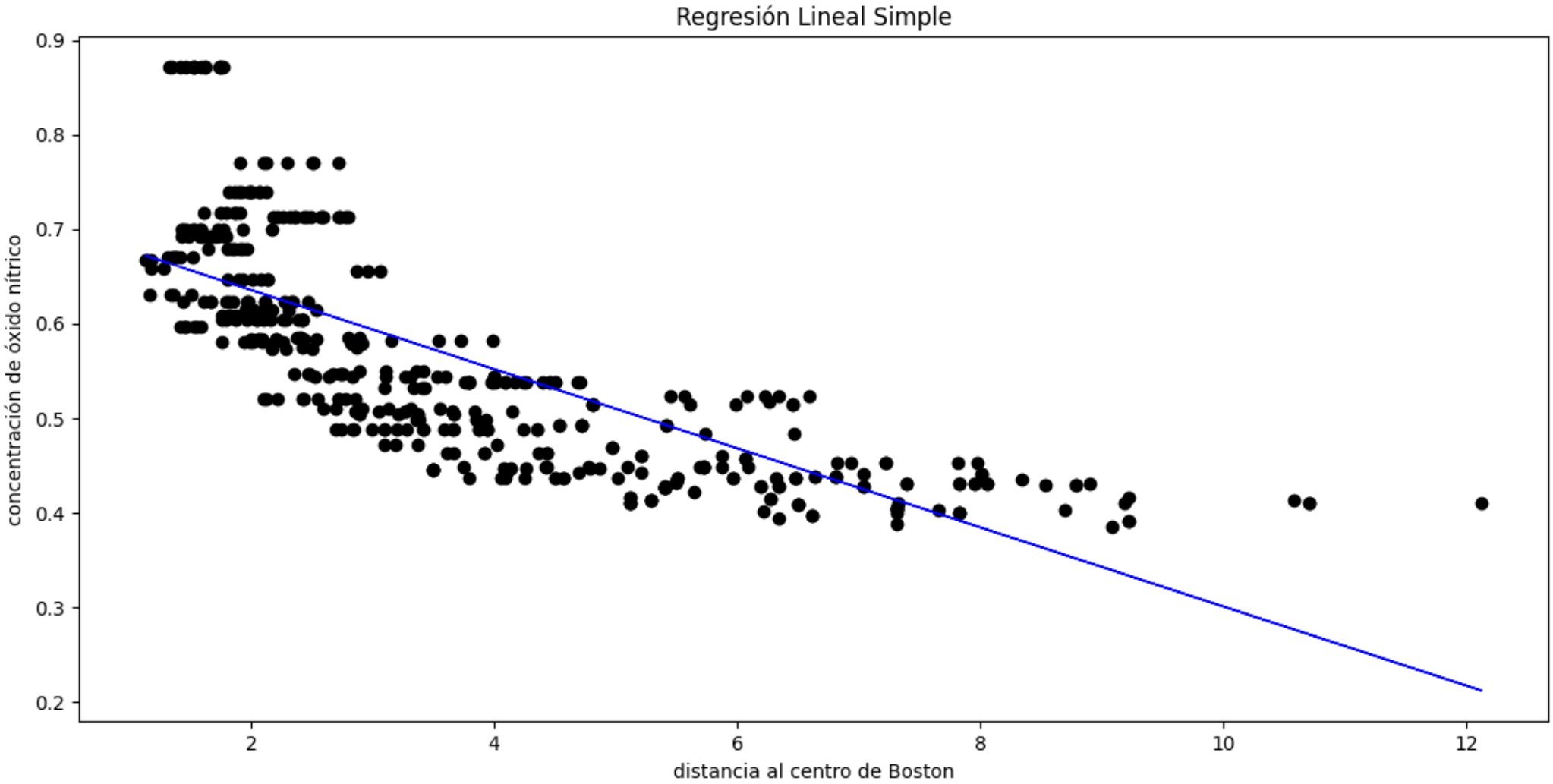
tamaño 0.5

$$\hat{Y} = -0.04365\,x + 0.7195$$



tamaño 0.9

$$\hat{Y} = -0.0483\,x + 0.7564$$



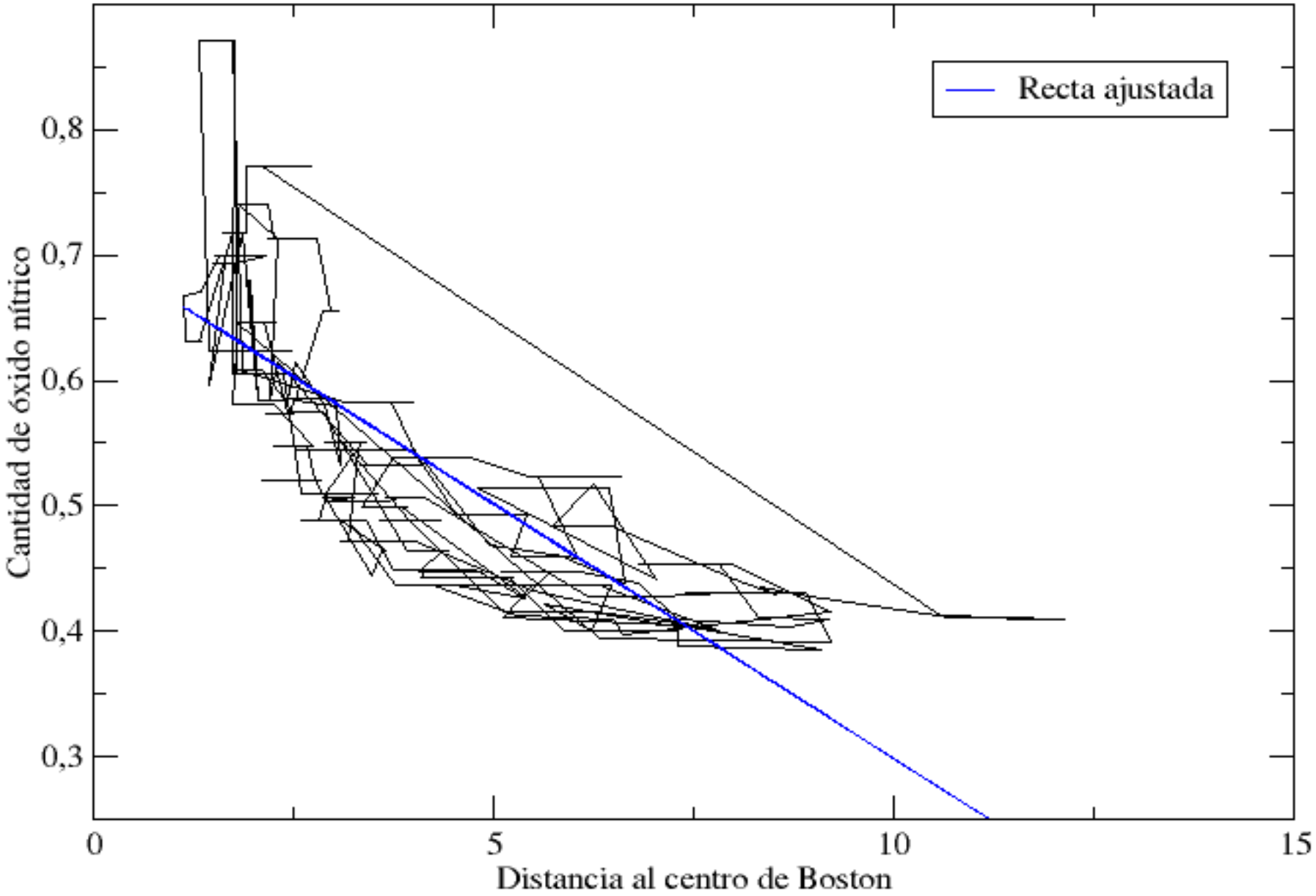
Segundo caso

Realizando el ajuste en xmgrace, se obtuvieron los datos:

$$\hat{Y} = -0.040605\,x + 0.70501$$

R=-0.7458124

R^2=0.5562 (precisión del modelo)


Recta ajustada
Cantidad de óxido nítrico
0,8
0,7
0,6
0,5
0,4
0,3
0
5
10
15
Distancia al centro de Boston

Tercer ejemplo (statsmodel)

Statsmodel *es un módulo de Python que proporciona clases y funciones para realizar pruebas y exploración de datos estadísticos.*

- ***¿Qué diferencias hay entre sklearn y statsmodel?:***

  Statsmodel sigue en gran medida el modelo tradicional en el que queremos saber qué tan bien un modelo determinado se ajusta a los datos y qué variables "explican" el resultado, o cuál es el tamaño del efecto.

  Scikit-learn sigue la tradición del aprendizaje automático, donde la tarea principal admitida es elegir el "mejor" modelo para la predicción.

Tercer ejemplo (statsmodel)

Veremos un ejemplo sencillo en el que se implementa este paquete y de qué manera realiza la Regresión:

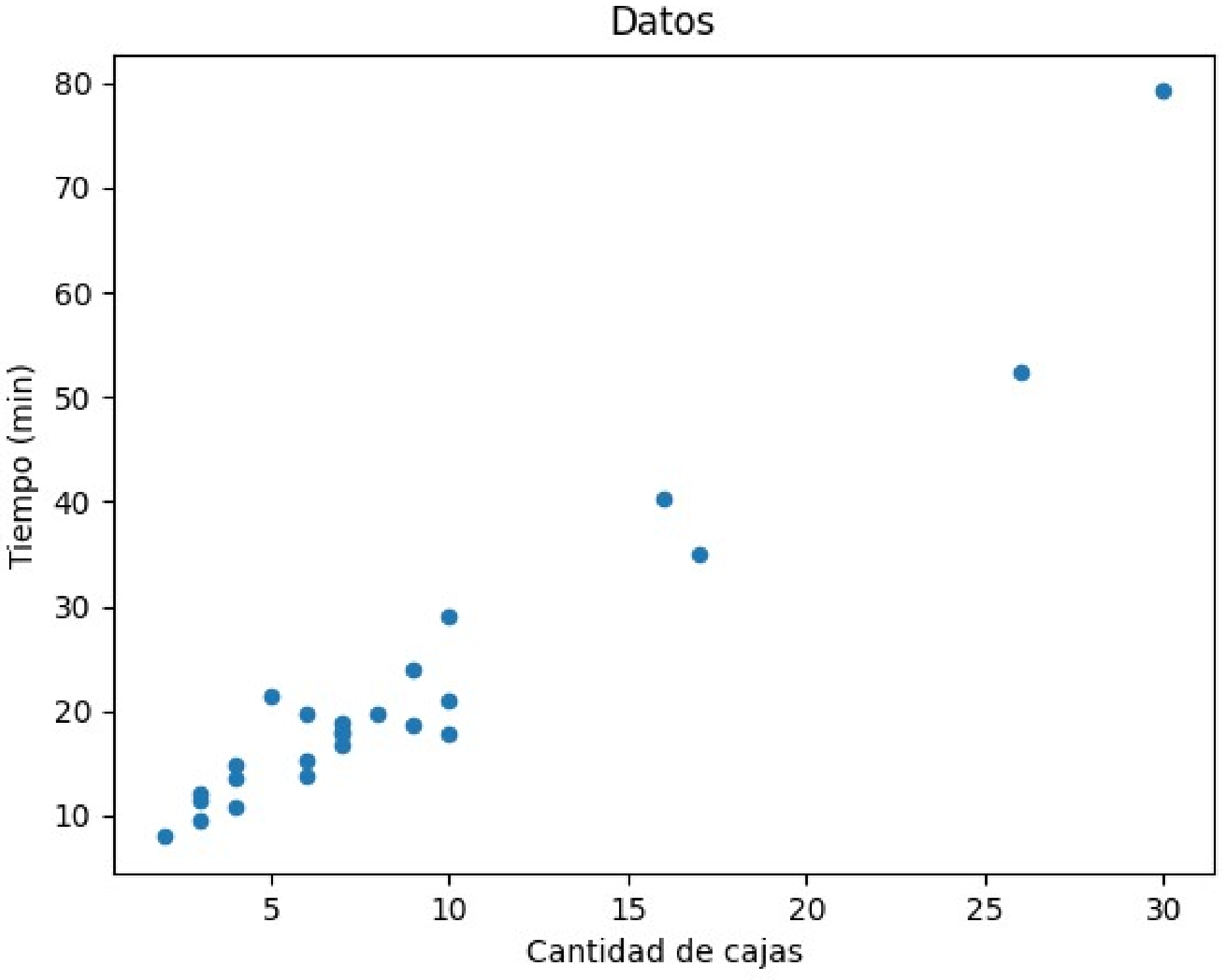
Tercer ejemplo

- *El dataset cuenta con 3 variables y 25 datos sobre la fabricación de cajas. Las variables usadas son:*

X_1: " cantidad de cajas fabricadas"

Y: " tiempo de fabricación (mín)"

A continuación vemos un gráfico de la distribución de los datos...


Datos
Tiempo (min)
Cantidad de cajas
80
70
60
50
40
30
20
10
5
10
15
20
25
30

Tercer ejemplo (Statsmodel)

```
# creamos el modelo de regresión:
#declaramos los valores de X e y
X = dt["x1"]
X = sm.add_constant(X)
y = dt["y"]
mod = smf.ols('y ~ x1', data=dt).fit()
print(mod.summary()) # imprimimos las estadísticas del modelo:
```

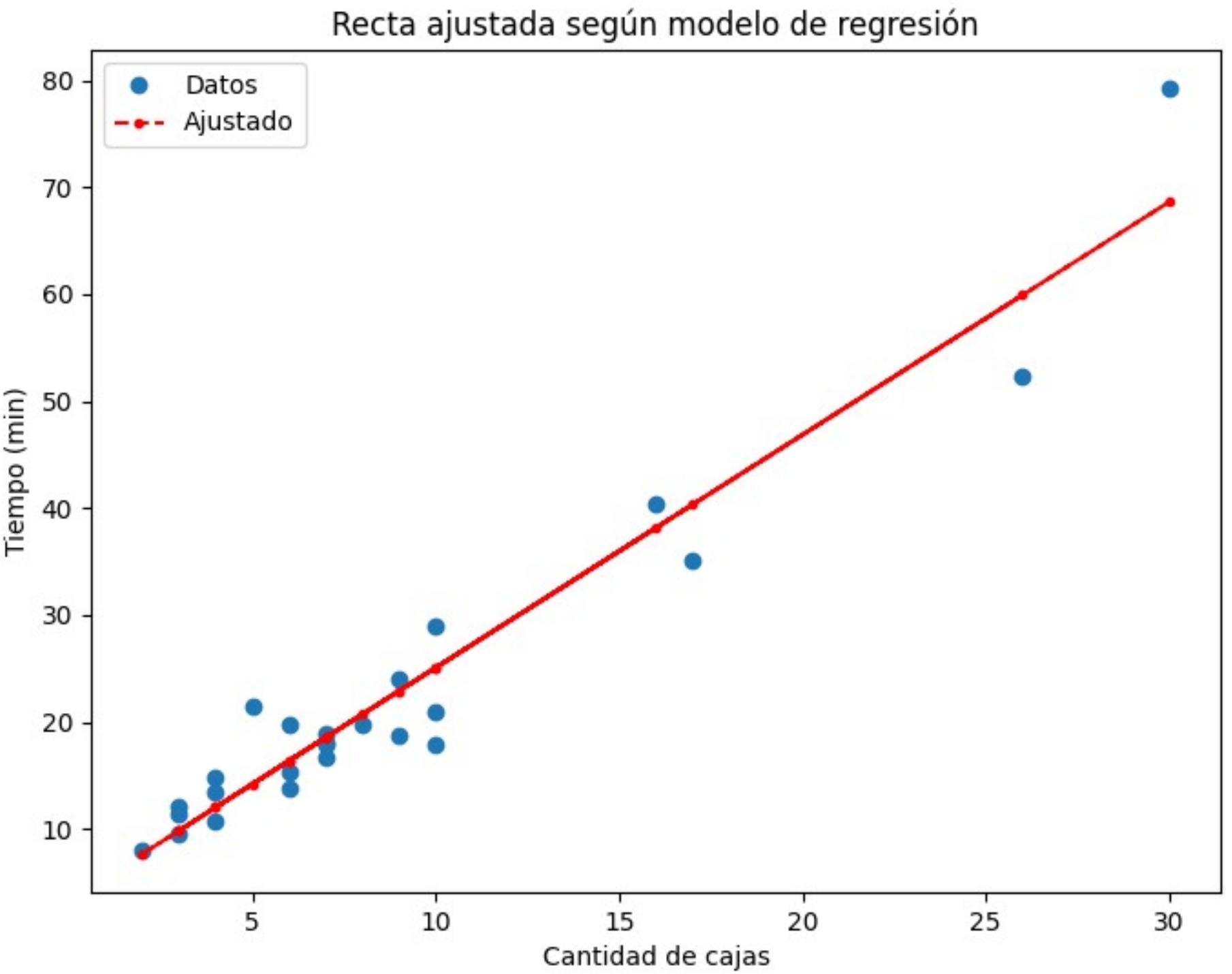
Tercer ejemplo (Statmodel)

Los datos del ajuste son:

$$\hat{Y} = 2.1762\,x + 3.3208$$

R=0.9643

R^2=0.930


Recta ajustada según modelo de regresión
Datos
Ajustado
Tiempo (min)
Cantidad de cajas
10
20
30
40
50
60
70
80
5
10
15
20
25
30

Tercer ejemplo (Statmodel)

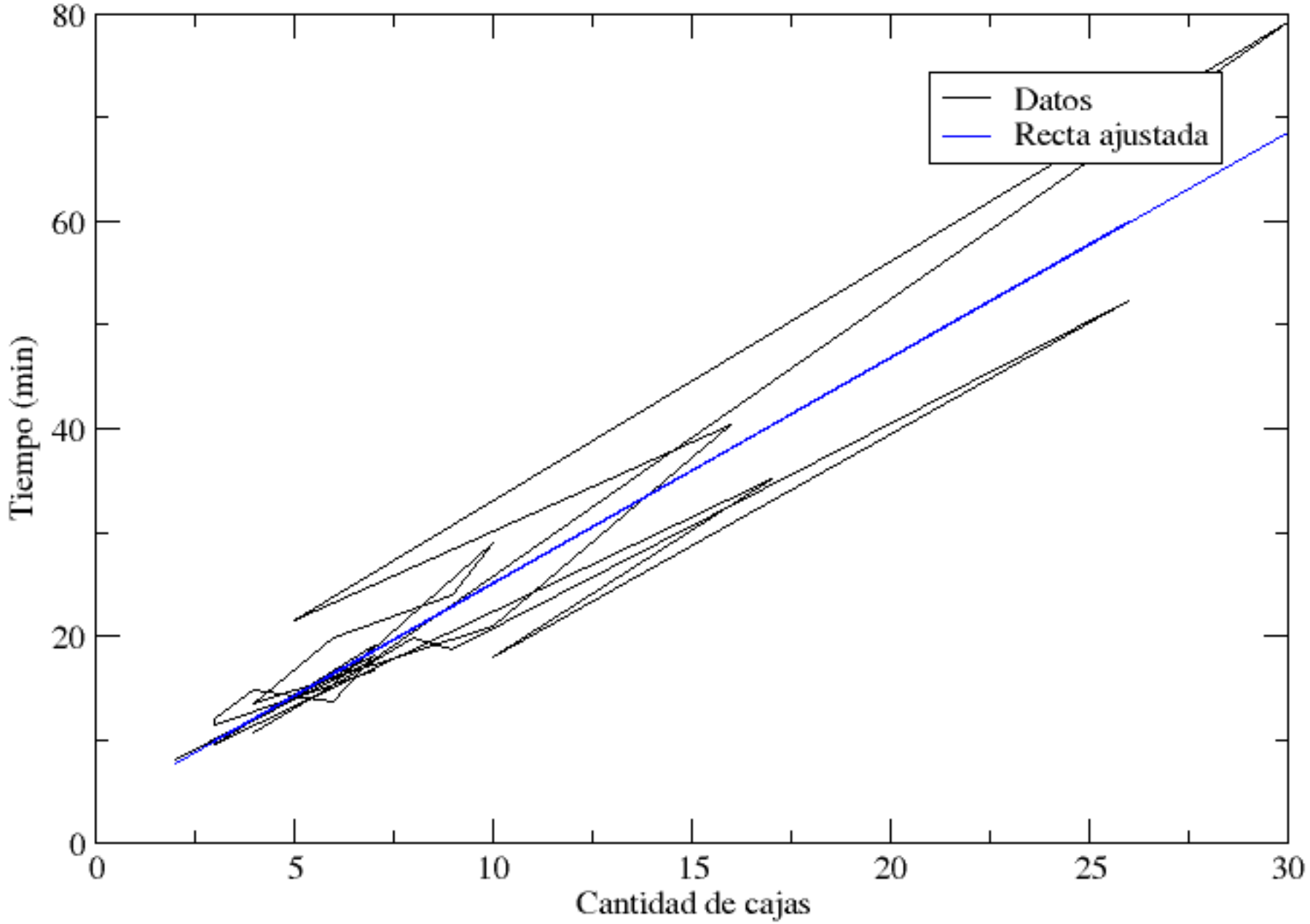
Se hizo un ajuste comparativo para estos datos en xmgrace, cuyos resultados son:

$$\hat{Y} = 2.1762\,x + 3.3208$$

$$R = 0.96461146$$

$$R^2 = 0.93048$$

El gráfico se muestra a continuación...


Datos
Recta ajustada
Tiempo (min)
Cantidad de cajas
0
20
40
60
80
0
5
10
15
20
25
30

Conclusión

Si bien los conjuntos de datos no obtuvieron como resultado modelos tan adecuados para explicar la relación entre las variables (lo cual no era el eje del trabajo tampoco), se destaca que al comparar los valores de regresión entre cada método y el utilizado en xmgrace, las similitudes fueron grandes, aunque el que más difiere es el método de predicción visto en segundo lugar.

Sin embargo, la diferencia entre ellos permite trabajar los conjuntos según lo esperado en nuestro trabajo, y en casos más específicos, aunque sea el más "inestable", el método proporcionado por sklearn nos da una idea de que el entrenamiento y predicción tiene variabilidad que a veces puede ser desfavorable (por ejemplo si se necesitara ser lo más próximos a la exactitud como sea posible).

Bibliografía

https://yuasaavedraco.github.io/Docs/Regresi%C3%B3n_Lineal_Simple_con_Python.html

https://stat.ethz.ch/R-manual/R-devel/library/datasets/html/00Index.html

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

https://www.statsmodels.org/stable/index.html

http://mate.dm.uba.ar/~meszre/apunte_regresion_lineal_szretter.pdf

Wackerly,Mendenhall,Scheaffer, Estadística Matemática con Aplicaciones

Bibliografía

Enlaces de datasets:

- *Dataset **cars**:*
  https://stat.ethz.ch/R-manual/R-devel/library/datasets/html/cars.html

- *Dataset **boston**:*

  https://www.cs.toronto.edu/~delve/data/boston/bostonDetail.html

- *Dataset **cajas**:*

  https://raw.githubusercontent.com/fhernanb/Python-para-estadistica/master/03%20Regression/Regresi%C3%B3n%20lineal%20simple/softdrink.csv