Python para el análisis de datos- Lectura 10

Ing. Pedro Rotta

Universidad de Piura - Vida Universitaria

Enero-2022

Regresión lineal

La regresión lineal es el método principal de machine learning. En este algoritmo modelamos un cojunto de datos consiguiendo la siguiente ecuación.

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + ...\theta_n x_n$$

El objetivo del algoritmo es obtener los coeficientes θ que conforman la ecuación en donde los valores x son datos de entrada y $\hat{y}$ es el valor predicho de la salida.

La función de costo es :

$$C(\theta) = \frac{1}{2} \sum (\hat{y} - y)^2$$

Regresión lineal

El algoritmo para desarrollar la regresión es el que continúa:

1) Extracción de características en una matriz X y del objetivo en un vector y

2) División del conjunto de datos disponible en 2 con el comando_test_split.

3)Creación del objeto que será el modelo de regresión lineal

4)Entrenamiento del modelo

5)Comprobación de resultados.

Para computarizar la regresión lineal usamos el método de los `mínimos cuadrados`. Para mirar un ejemplo bidimensional, miremos el problema 1

Regresión lineal múltiple y adaptada a la no linearidad

Con este mismo procedimiento podemos realizar una regresión lineal con varias entradas, como se ve en el problema 2.

También, es posible desarrollar un análisis de regresión no lineal,es decir,una `regresión polinomial`.

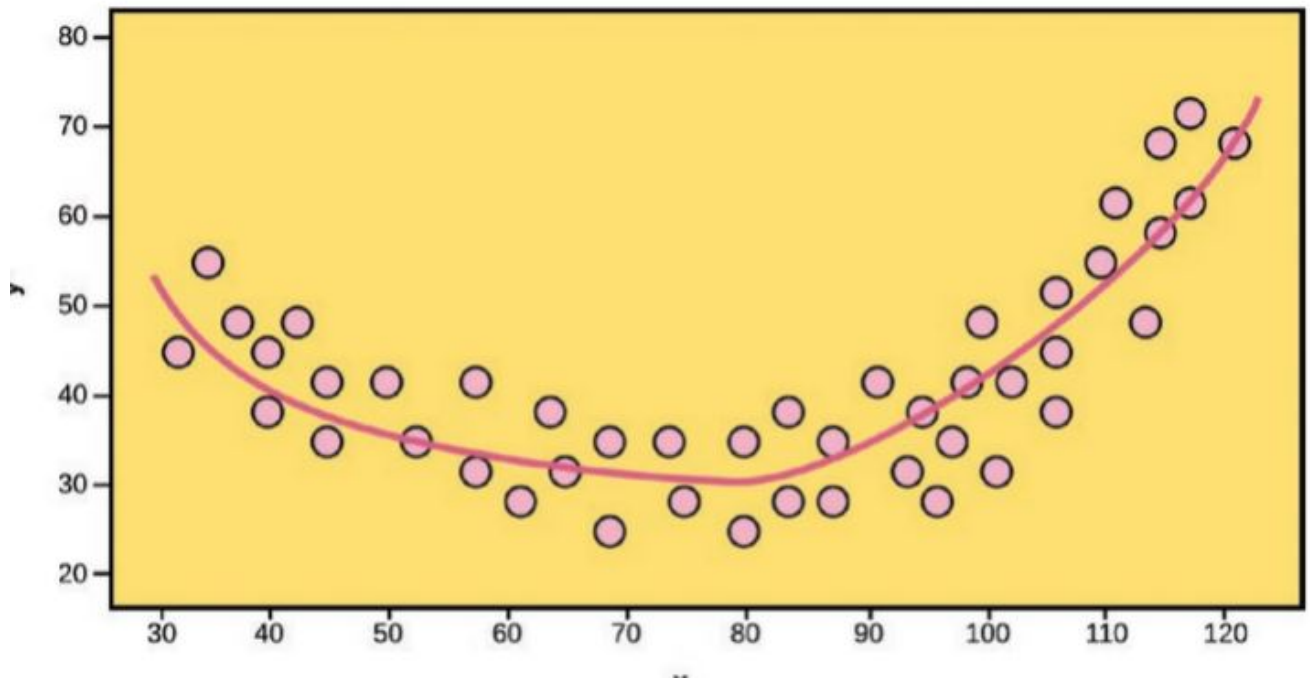
Una regresión polinomial, ajusta a la data a partir de una regresión no lineal añadiendo términos polinomiales de orden superior en el conjunto de datos.

La formalización quedaría de la forma:

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_2 + \theta_4 x_2^2 ... \theta_n x_n + \theta_n x_n^2$$

Regresión lineal adaptada a la no linealidad

Como será de suponer, lo que se tendrá será una regresión como se muestra en esta figura.



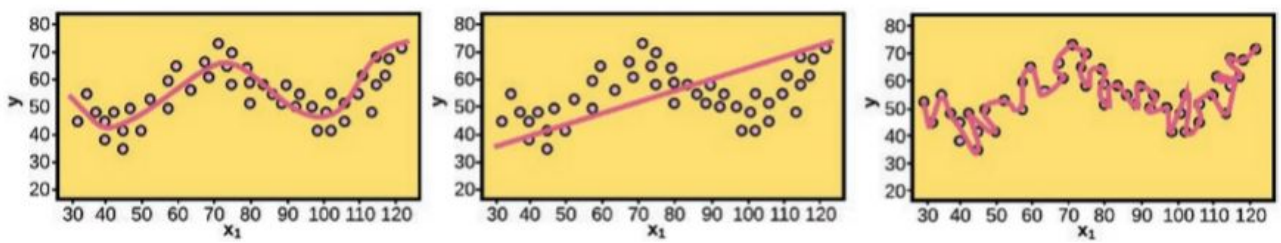
Cabe resaltar,que en realidad sobre la base de esta aproximación, se sigue teniendo una interacción lineal. Este tipo de análisis puede generar sobreajuste en los datos.

Sobreajuste y Subajuste.

Cabe resaltar los dos problemas que pueden existir al momento de encontrarnos diseñando un algoritmo de machine learning.

El **Subajuste** o modelo con alto **Bias** es cuando el modelo es muy sencillo, y por ende, no predice correctamente el comportamiento de nuestro datos de entrenamiento, y en menor cantidad los datos de prueba.

El **sobreajuste** o modelo con alta **varianza** es cuando el modelo ajusta perfectamente los datos de entrenamiento, pero fallará al momento de que se le muestre nueva data.



Computarización de un modelo lineal adaptado

La computarización de un modelo adaptado se da gracias a la ayuda del módulo preprocessing de Sklearn. La sintaxis del método a llamar en python es:

```
from sklearn.preprocessing import PolynomialFeatures
```

```
pol_feat = PolynomialFeatures(n)
```

```
Xho =polynomial_features.fit_transform(X)
```

Ver problema 3.

Datasets de prueba en Sklearn

Normalmente es díficil encontrar tipos de datos para pruebas o práctica. Es por esa razón que Sklearn, ofrece una colección de conjuntos de datos. Para usarlo su sintaxis correspondiente es:

```
from sklearn import datasets

datos = datasets.load_dataset()
```

Donde dataset representa los tipos de conjuntos de datos que puede tener.

Cada conjunto devuelve una tupla de datos separada en features y target. La sintaxis para obtenerla, si usamos estos datos es:

```
X = datos.data

y = datos.target
```

Los datos de práctica están en `https://scikit-learn.org/stable/datasets/toy_dataset.html`