**I: Hồi quy Tuyến tính:**

**1. Giới thiệu mô hình**

Hồi quy tuyến tính hay **Linear Regessoin** là một thuật toán học máy có giám sát được xây dựng dùng để dự đoán một giá trị đầu ra liên tiếp (biến phụ thuộc) dựa trên các giá trị đầu vào (biến độc lập).

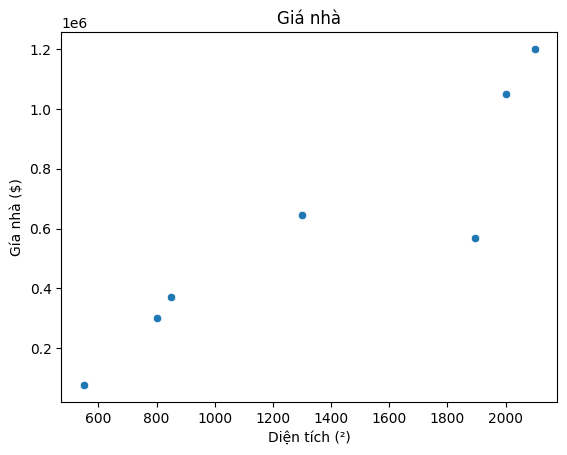
Ví dụ, ta có một mô hình hồi quy dự đoán giá nhà. Ta cung cấp biến độc lập ở đây là **diện tích** ngôi nhà, công việc của mô hình là cần dự đoán giá trị của biến phụ thuộc hay **giá nhà** dựa trên diện tích đã cho. Tất nhiên mô hình này được đào tạo bằng bộ dữ liệu tương đương là **diện tích** - **giá nhà** (trên thực tế, giá của ngôi nhà còn vào phụ thuộc nhiều yếu tố khác như: số phòng, vị trí địa lí, năm xây dựng, … bấy giờ ta sẽ xây dựng mô hình hồi quy đa tuyến tính **multiple linear regression**, có nghĩa biến độc lập lúc này bao gốm nhiều yếu tố)

Chú ý rằng, biến phụ thuộc phải là liên tục như câng nặng, tiền lương, giá cả, … chứ không thể phân lớp như giới tính (nam – nữ), tốt nghiệp (có – không). Riêng với các biến độc lập có thể liên tục hoặc phân lớp

Ví dụ: ta có dữ liệu về giá bán nhà như sau:

|  |  |  |
| --- | --- | --- |
|  | Diện tích(m²) | Giá($) |
| 0 | 2.000 | 1.050.000 |
| 1 | 800 | 300.000 |
| 2 | 850 | 370.000 |
| 3 | 550 | 78.000 |
| 4 | 2.100 | 1.200.000 |
| 5 | 1.300 | 645.000 |
| 6 | 1.893 | 567.000 |

Biểu diễn dữ liệu này bằng biểu đồ Scatter trong Python



Hình 1

Giả sử giờ ta có dữ liệu của một ngôi nhà mới với diện tích là 1540² thì ta cần dự đoán xem giá của ngôi nhà này là bao nhiêu dựa trên những dữ liệu đã cho.

Trước tiên, ta sẽ định nghĩa vài kí hiệu:

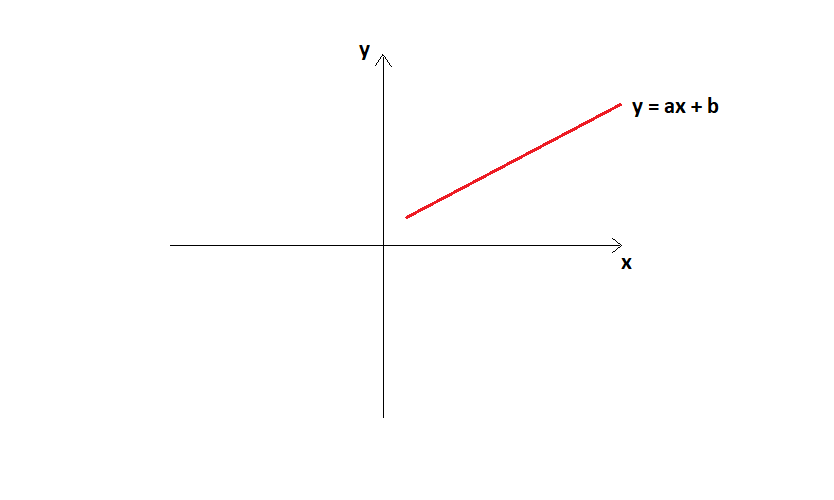
* xi : biến độc lập
* yi : biến phụ thuộc
* (xi,yi) : một cặp huấn luyện
* i = 1, 2, 3, …, n
* n là số cặp huấn luyện (x,y)

Với mô hình hồi quy tuyến tính, ta cố gắng xây dựng mối quan hệ giữa xi và yi, mối quan hệ này có thể được biểu diễn bằng một hàm giả thuyết

***yi = α*xi *+ β***

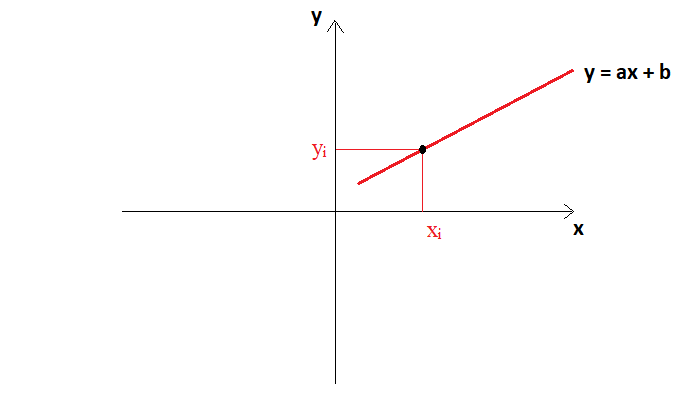
trong đó α và β là hai tham số của hàm mà ta chưa biết.

có thể thấy, hàm số phía trên là hàm bậc nhất mà nếu ta biểu diễn dưới không gian hai chiều thì sẽ được một đường thẳng với α và β được giả định cho trước



Hình 2

giả sử giờ có một xi bất kì, ta sẽ tính được yi tương ứng, cặp (xi,yi) được biểu diễn bằng một điểm trên đường thẳng y = αx + β

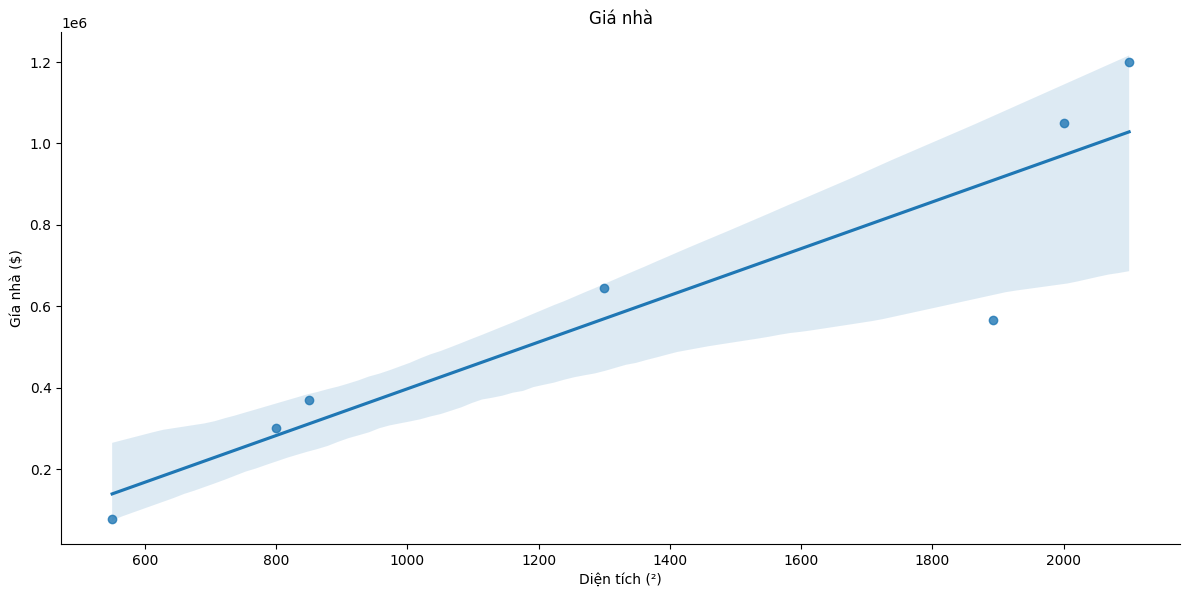


Hình 3

Quay lại với biểu đồ scatter giá nhà (hình 1) lúc nãy, nếu ta vẽ được một đường thẳng như vậy, ta dễ dàng dự đoán được giá của ngôi nhà mới dựa vào diện tích được cung cấp. Như ta thấy ở hình 3 cặp (xi,yi) nằm trên đường thẳng ***y = α*x *+ β*** hay nói cách khác đường thẳng này đi qua các cặp điểm (xi,yi). Với bộ dữ liệu có sẵn, các điểm (xi,yi) đã được cung cấp, làm thế nào ta có thể vẽ được đường thẳng

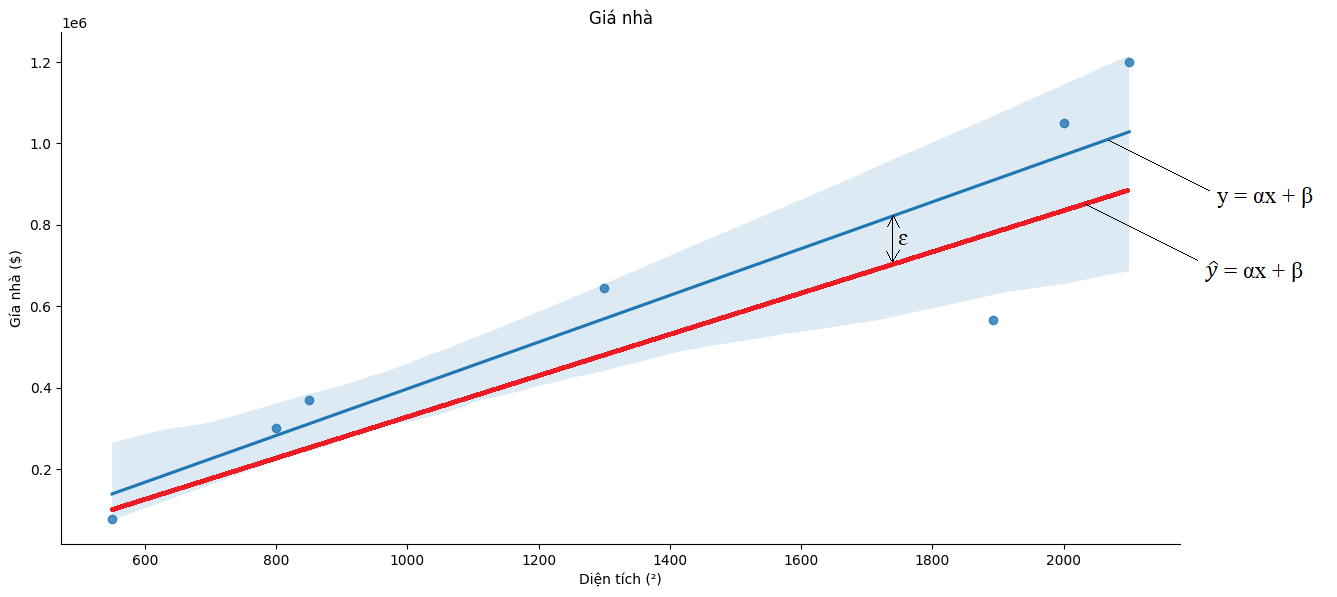
***y = α*x *+ β*** ?

để hình dùng, ta sẽ dùng thư viện seaborn của python để vẽ được đường thẳng như thế:



Hình 4

dễ dàng nhận thấy, đường thẳng này đi qua giữa các điểm (xi,yi). nếu không dùng thư viện thì sao, giả sử bằng mắt thường, ta thử vẽ một đường thẳng làm sao đi qua giữa các điểm trên biểu đồ



Hình 5

ta sẽ kí hiệu = αx + β là phương trình đường thẳng mà ta dự đoán, y = αx + β là phương trình đường thẳng thực tế. Có thể thấy rằng, mắt thường khó có thể kẻ được đường thẳng chính xác với thực tế, và giữa dự đoán với thực tế sẽ có một độ lệch ɛ= . Ta sẽ gọi độ lệch này là **lỗi**. Nếu ta giảm được ɛ về mức càng gần bằng 0 thì của ta càng chính xác. Vấn đề là làm sao để giảm giá ɛ này ?

rõ ràng là ta không thể thay đổ thông số x, y vì đây là dữ liệu. vì thế ta cần phải thay đổi hai hệ số α và β

**2. Hàm chi phí:**

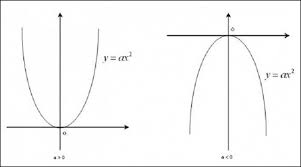
Với mỗi xi ta sẽ có giá trị dự đoán i  biểu diễn dưới hàm số i = αxi + β và giá trị y thực tế, lỗi của giá trị này là ɛi , tổng các lỗi được gọi là chi phí. Nếu ta giảm chi phí này xuống càng thấp thì giá trị dự đoán của ta càng chính xác. Trong mô hình hồi quy tuyến tính, ta sẽ huấn luyện mô hình với số lần nhất định, mỗi lần huấn luyện ta sẽ cập nhập lại hai tham số α và β và tính toán lại chi phí, đến khi chi phí giảm đển mức chấp nhận được, ta sẽ ngừng đào tạo và sử dụng α và β cuối cùng đề làm tham số cho phương trình y = αx + β, đến đây mô hình của ta được coi như hoàn tất.

Quay với lại hàm chi phí, ta sẽ dùng nguyên tắc **Trung bình Bình phương Lỗi – Mean Square Error (MSE)** để tính toán, mô tả bằng công thức như sau:

i – (αxi + β))2

**3. Cập nhập lai α và β:**

MSE là phương trình bậc 2, nếu ta biểu diễn trên không gian 2 chiều ta sẽ được một đường parabol có dạng



MSE đạt giá trị càng nhỏ khi đạo hàm của nó càng gần bằng 0, ta cần tính toán lại 2 tham số α và β (ban đầu ta sẽ tự cho khi xây dựng mô hình) dựa vào từng điểm trên đường cong parabol. Mỗi điểm này ta sẽ gọi là tốc độ học **learning rate**.

Nếu giải đạo hàm theo α và β ta sẽ được 2 phương trình:

Sau khi đã tính α’ và β’ mới, ta tiến hành cập nhập lại hai tham số này theo công thức với learning rate là giá trị của tốc độ học mà ta tự cho trước:

α = (α - α’) \* learning rate

β = (β – β’) \* laerning rate

Đến cuối cùng, ta sẽ có α và β tối ưu nhất, hàm số của ta cũng như thế mà hoàn thành.

**4. Giải thuật xây dựng mô hình:**

**Đầu vào:**

- x: tập biến độc lập

- y: tập biến phụ thuộc

- α, β

- learning\_rate: tốc độ học

- loop: số lần học

- x\_new: biến độc lập mới, dùng đề dự đoán biến phụ thuộc mới

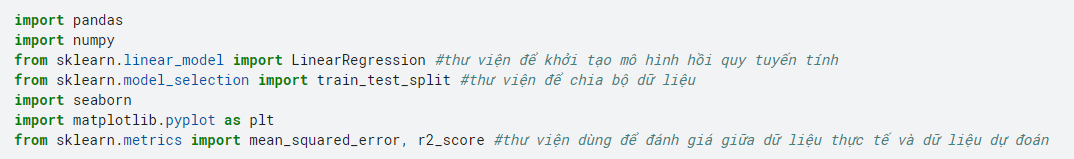
**Đầu ra:** giá trị phụ thuộc mới y\_new

|  |
| --- |
| **BEGIN:**  **for** mỗi giá trị k trong số lần học loop:  //tính toán hàm chi phi  **for** mỗi giá trị **i** trong độ dài len(x) của tập biến độc lập x:  tổng\_lỗi += (y[i] – (a \* x[i] + b))2  trung\_bình\_lỗi = tổng\_lỗi / len(x)  **end for**  //cập nhập lại α và β  **for** mỗi giá trị **i** trong độ dài len(x) của tập biến độc lập x:  tổng\_α += -2 \* x[i] \* (y[i] – (a \* x[i] + b))  tổng\_b += -2 \* (y[i] – (a \* x[i] + b))  **end for**  α = (tổng\_α / n) \* learning\_rate  β = (tổng\_β / n) \* learning\_rate  **end for**  //dự đoán giá trị phụ thuộc y\_new  **return** α \* x\_new + β  **END** |

**5. Xây dựng mô hình hồi quy tuyến tính với thư viện *sklearn.linear\_model***

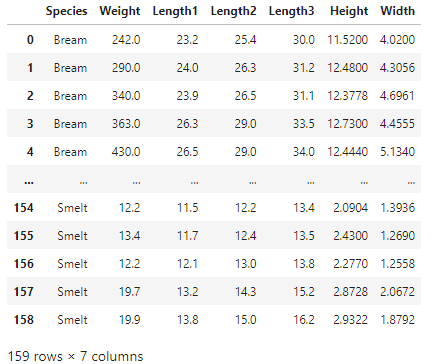
Chúng ta sẽ sử dụng bộ dữ liệu trong **thị thường cá** để mô tả việc xây dựng mô hình

* **Trước tiên ta sẽ import thư viện**



* Để ý thư viện ***sklearn.linear\_model* import *LinearRegression***: đây là thư viện chính để xây dựng mô hình hồi quy tuyến tính
* Thư viện ***sklearn.model\_selection* import *train\_test\_split***: dùng để chia một bộ dữ liệu thành 2 phần **train** và **test**
* Thư viện ***sklearn.metric* import *mean\_squared\_error, r2\_score***: dùng để đánh giá dữ liệu giữa dự đoán và thực tế
* **Import data:**





Giải thích dữ liệu: bộ dữ liệu này mô tả các kích thước và cân nặng của từng loài cá

+ Species: tên của loài cá

+ Weight: cân nặng của cá

+ Length1, Length2, Length3, Height, Width: các kích thước của cá

🡺 Với bộ dữ liệu này ta sẽ xây dựng mô hình hồi quy tuyến tính để dự đoán Weight (câng nặng) của cá dựa trên các thông số về kích thước.

* **Thống kê và tiền xử lý dữ liệu (đây là bước quan trọng trước khi xây dựng mô hình):**

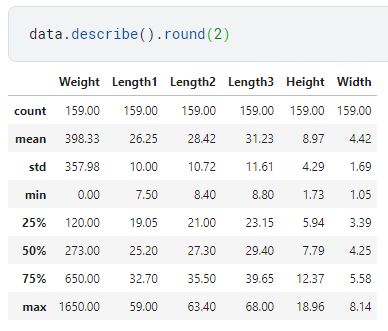
Cụ thể việc cần làm là:

+ Thay thế hoặc loại bỏ các giá trị rỗng

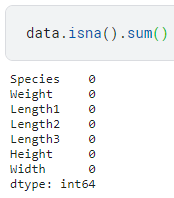
+ Mã hóa hoặc phân bin dữ liệu về dạng số (đối với dữ liệu phân lớp)

+ Xử lý outliers (ngoại lai)

* Thống kê dữ liệu:



* Kiểm tra dữ liệu rỗng:



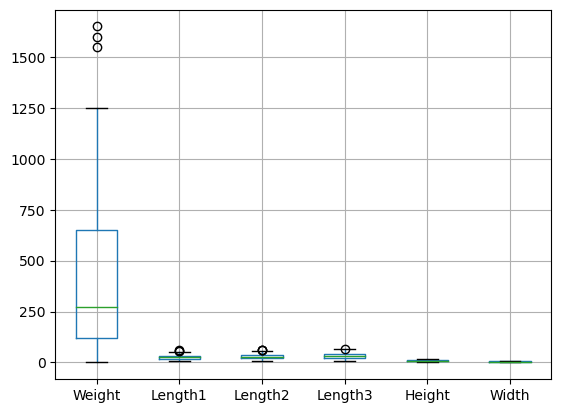
🡺 Không có giá trị rỗng trong các cột

* Kiểm tra outliers (ngoại lai):Sử dụng biểu đồ boxplot để quan sát

|  |  |
| --- | --- |
| Length1 | Lenght2 |
| Length3 | Weight |
| Width | Height |

Có thể dùng hàm boxplot của dataframe để xem ngoại lai của tất cả các cột (kiểu dữ liệu numeric và các thông số min max trong các cột không quá chênh lệch) trong dataframe:





🡺 Ta thấy các đặc trưng như: *Lenght1, Lenght2, Lenght3, Weight* chứa outliers*, Width* và *Height* thì không, ta tiến hành xử lý outliers

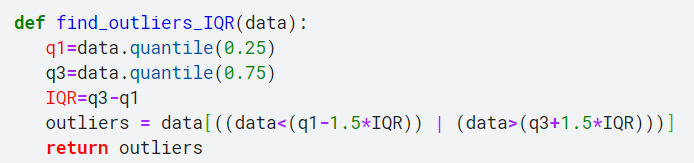
* **Loại bỏ outliers**:

Outliers ảnh hưởng rất nhiều đến tính đúng đắn của mô hình khi xây dựng. ta cần xử lý outliers bằng cách:

+ Tìm các vị trí chứa giá trị outliers

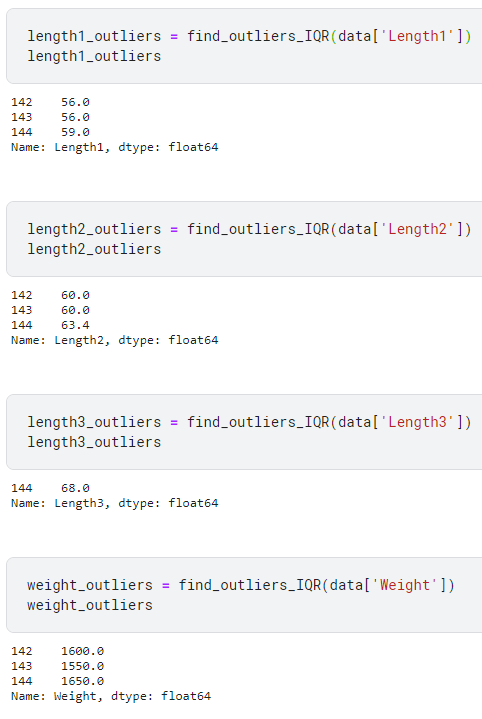
+ Kiểm tra, chỉnh sửa hoặc loại bỏ nếu cần thiết

* Dựng hàm tìm outliers dùng **interquartile range** **(IQR)**



Hàm nhận tham số đầu và là data cần tìm outliers, tính chỉ số IQR dựa và tứ vị phân và trả về outliers của data đó

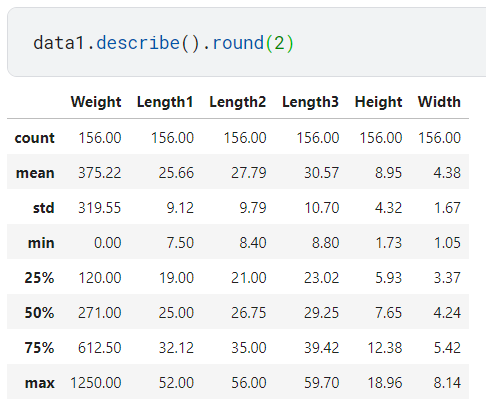
* Sử dụng hàm này và tìm outliers trên các đặc trưng



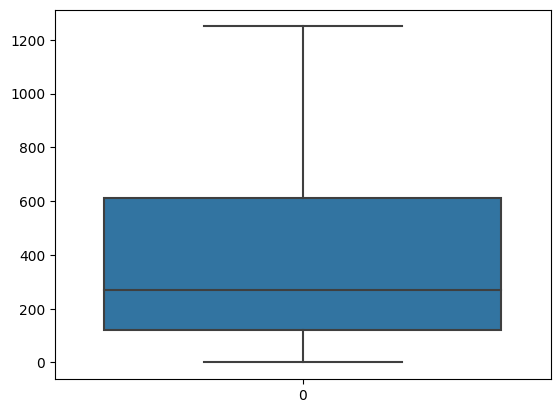
🡺 Nhận thấy rằng các hàng (records) tại index = [142,143,144] chứa giá trị ngoại lai trên tất cả các cột đang xét, vì vậy ta tiến hành loại bỏ những hàng này



* Thống kê lại dữ liệu mới

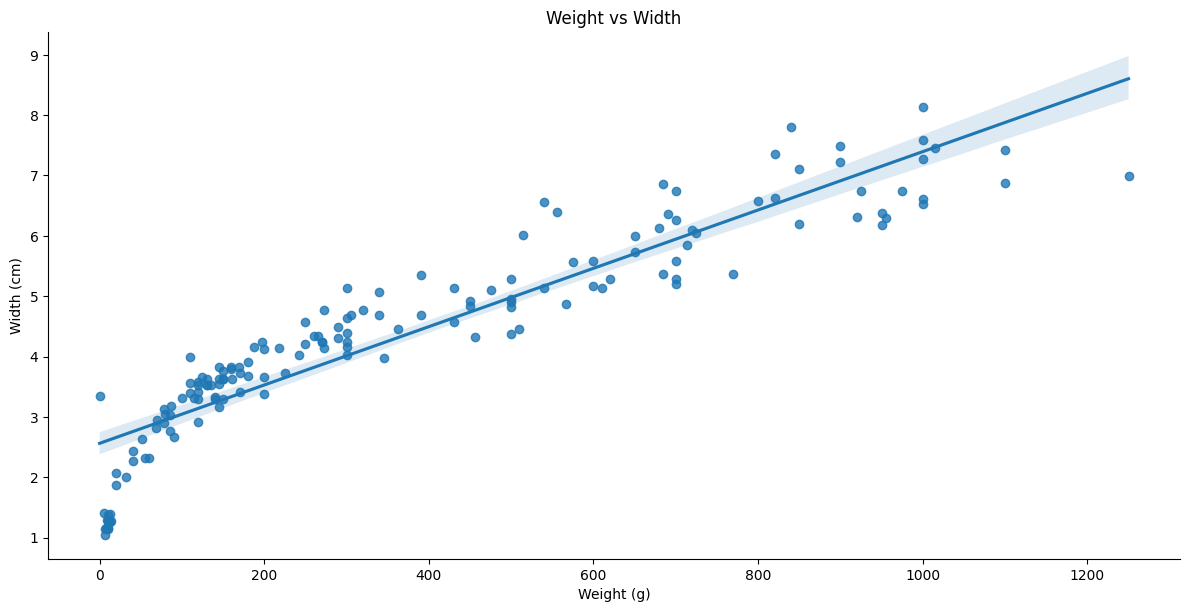


* Kiểm tra lại outliers trên từng đặc trưng bằng biểu đồ boxplot (đại diện bởi cột Weight)

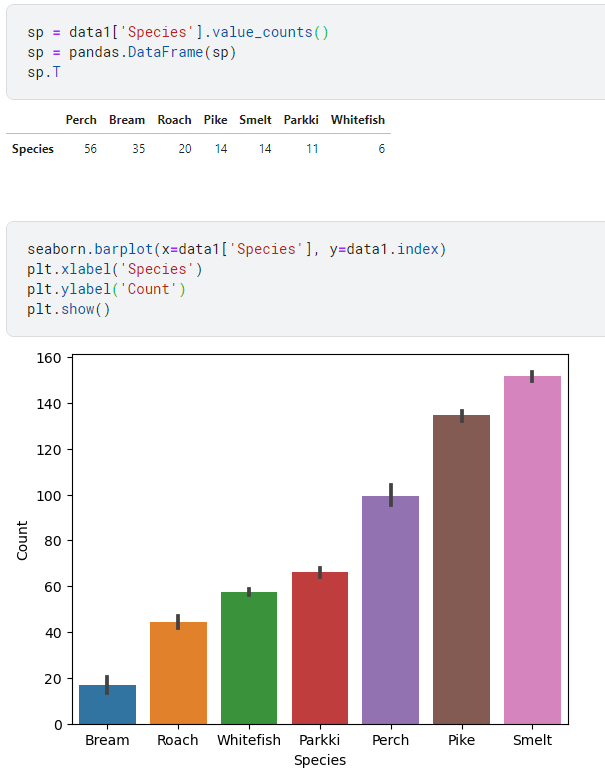


🡺 outliers đã được loại bỏ

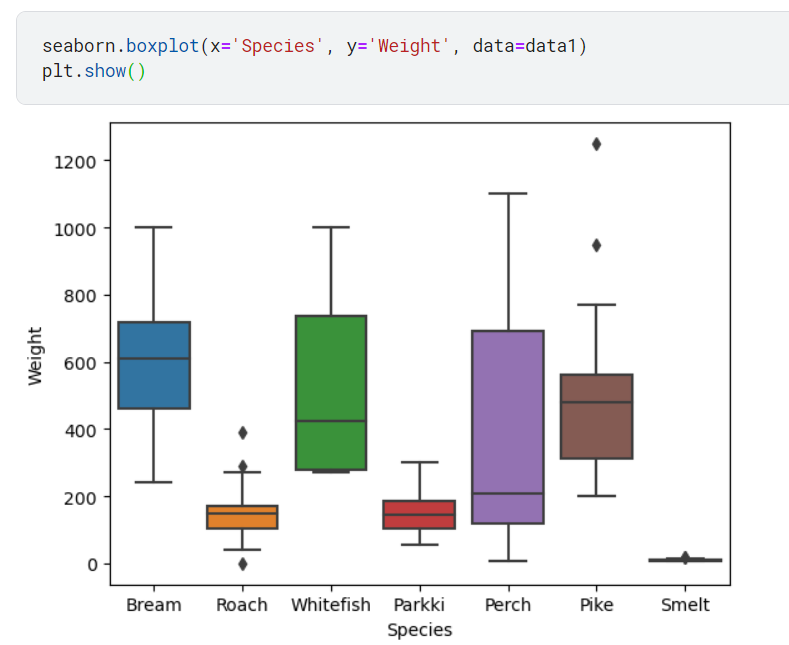
* Một vài biểu đồ để trực quan hóa dữa liệu
* biểu đồ lmplot giữa cột width (cm) và Weight (g), tương tự như giữa Weight với các cột khác, cả 2 có sự phụ thuộc theo tỉ lệ thuận



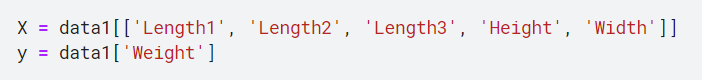
* Thống kê các loại cá và số lượng mỗi loài trong tập dữ liệu



* Biểu đồ boxplot giữa Species và Weight



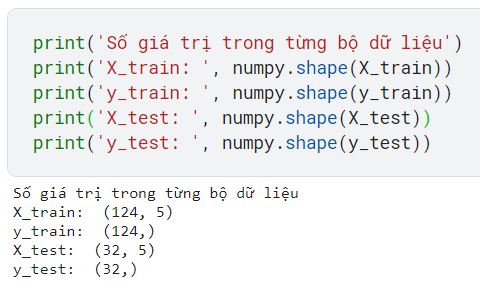
* **Xây dựng mô hình**
* Xác định biến độc lập và biến phụ thuộc, ở đây cân nặng của cá phụ thuộc vào kích thước nên biến phụ thuộc (y) và biến độc lập (X) sẽ là:



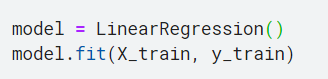
* Chia dữ liệu thành 2 phần (không giao nhau) **train** và **test** theo tỉ lệ 8:2



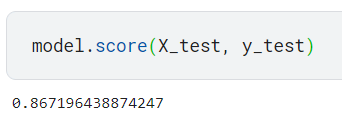
* Xem số giá trị trong từng bộ dữ liệu



* Xây dựng mô hình



* Dùng mô hình để đánh giá mức độ phụ thuộc của biến phụ thuộc với biến độc lập



🡺 0.86 tương đương với 86% giá trị của biến phụ thuộc phụ thuộc vào biến độc lập

* Hàm số giả thuyết của mô hình hồi quy đa tuyến tính (multiple linear regression) có dạng:

y = α1X1 + α2X2 + α3X3 + ... + αnXn + β

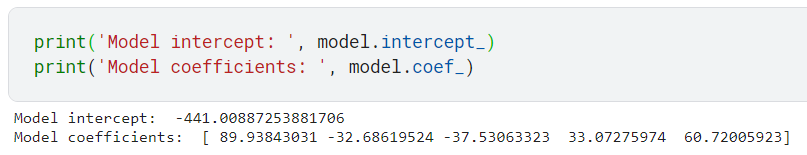
+ X1 – Xn: các biến độc lập

+ y: biến phụ thuộc

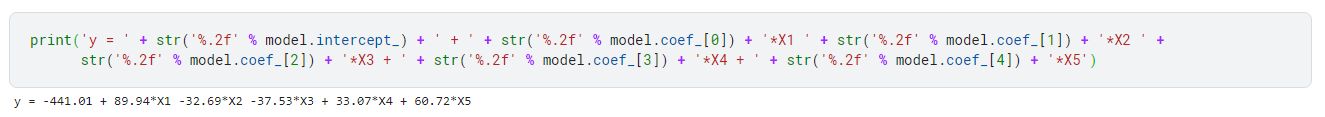
+ α và β: các tham số không thay đổi của hàm

+ n: số lượng biến độc lập

ta sẽ dùng hàm **intercept\_** và **coef\_** để tìm các giá trị **β** và **α**



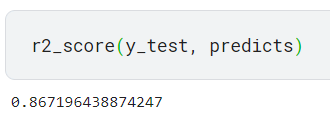
hiển thị hàm số giá thuyết



* **Dùng mô hình để dự đoán**

****

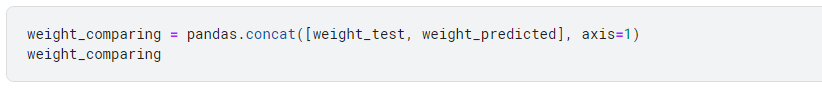
* **Đánh giá kết quả dự đoán**
* Đánh giá mức độ phụ thuộc của **predicts** và **y\_test**, ở đây là mức độ tương đương của các giá trị giữa 2 tập

****

**🡺** có 86% giá trị tương đương giữa 2 tập

* Tạo 1 **Dataframe** để so sánhgiữa 2 giá trị là **Weight** - thực tế và **Predicted Weight** - dự đoán







🡺 từ so sánh trên, ta có thể thấy các giá trị dự đoán có xu hướng sai khi Weight nhỏ

* Sử dụng biểu đồ boxplot để xem xét mức độ chênh lệnh giữa dự đoán và thực tế

|  |  |
| --- | --- |
|  |  |
|  |  |
|  | |

* Source code: [*https://www.kaggle.com/code/voduylong76/baocao*](https://www.kaggle.com/code/voduylong76/baocao)