+ Logistic Regression là một mô hình học máy được sử dụng cho bài toán classification, tức dự đoán một đối tượng thuộc vào một trong hai nhóm.

+ Logistic Regression làm việc dựa trên nguyên tắc của hàm sigmoid - một hàm phi tuyến tự chuyển input thành xác suất thuộc về một trong hai lớp nhị phân ("có" hoặc "không", hay là "0" hoặc "1").

-> Hàm Sigmoid được biểu diễn như sau:

S(z) = 1 / (1 + e ^ (-z))

\* Chú thích: Hàm Sigmoid nhận input là z bất kỳ và trả về output là một giá trị xác suất nằm trong khoảng [0, 1]. Thì input z của hàm sigmoid trong Logistic Regression được tính bằng công thức là: z = Xw với X: là một matrix và w là weight (trọng số) của ma trận đó.

+ Việc huấn luyện mô hình Logistic Regression là tìm ra trọng số w sao cho predicted output của hàm Sigmoid gần với output thực tế nhất.

* Logistic Regression có dạng tổng hợp sau:

+ Phân loại class (lớp): Dựa vào giá trị xác suất được dự đoán từ mô hình, ta có thể phân loại input X vào lớp 1 (hoặc lớp “có”) nếu P(Y=1∣X)>0.5 và vào lớp 0 (hoặc lớp “không”) nếu ngược lại.

+ Để đánh giá performance của model Logistic Regression thì ta sử dụng hàm Cross-Entropy (hay còn gọi là Log Loss). Hàm Cross-Entropy là một hàm số để đo mức độ lỗi của mô hình tạo ra trong quá trình dự đoán output từ input đã cho.

+ Hàm mất mát Cross-Entropy được định nghĩa như sau:

*L*(*w*)=−*(1/n) x (n)*​∑(*i*=1)​[*yi*​log(*pi)*​+(1−*yi*​)log(1−*pi*​)]

*Trong đó:*

n: số lượng mẫu dữ liệu trong tập huấn luyện.

y: giá trị thực tế của output thứ i.

p: xác suất dự đoán thuộc class 1 (hay có thể là “đúng”) của model có input thứ i

+ Trong quá trình train model (huấn luyện mô hình), chúng ta sẽ luôn luôn tìm cách cập nhật bộ trọng số *w* sao cho giá trị của Cross-Entropy nhỏ nhất có thể. -> Điều này giúp cho model dự đoán output tốt nhất. Thì để tìm giá trị tối ưu cho *w,* chúng ta sử dụng thuật toán Gradient Descent.

* Công thức Gradient của Cross-Entropy là:

∇*L*(*w*)=*(1/n)*​*X^T*(*S*(*Xw*)−*Y*)

+ `liblinear` là một thư viện được sử dụng để giải quyết các bài toán tối ưu hóa cho các mô hình học máy tuyến tính, bao gồm Logistic Regression.

+ Thuật toán `liblinear` sử dụng một phương pháp tối ưu hóa dựa trên Gradient Descent, nhưng được tối ưu hóa cho các mô hình tuyến tính và có thể xử lý được các bài toán với số lượng đặc trưng lớn.

+ Trong thư viện `scikit-learn` của Python, ta có thể chọn sử dụng `liblinear` cho Logistic Regression bằng cách thiết lập tham số `solver='liblinear'` khi khởi tạo mô hình.

Tuy nhiên, cần lưu ý rằng `liblinear` chỉ hoạt động tốt với các bài toán nhỏ đến trung bình. Đối với các bài toán có kích thước lớn, bạn có thể cần sử dụng các thuật toán khác như `sag` hoặc `saga`.