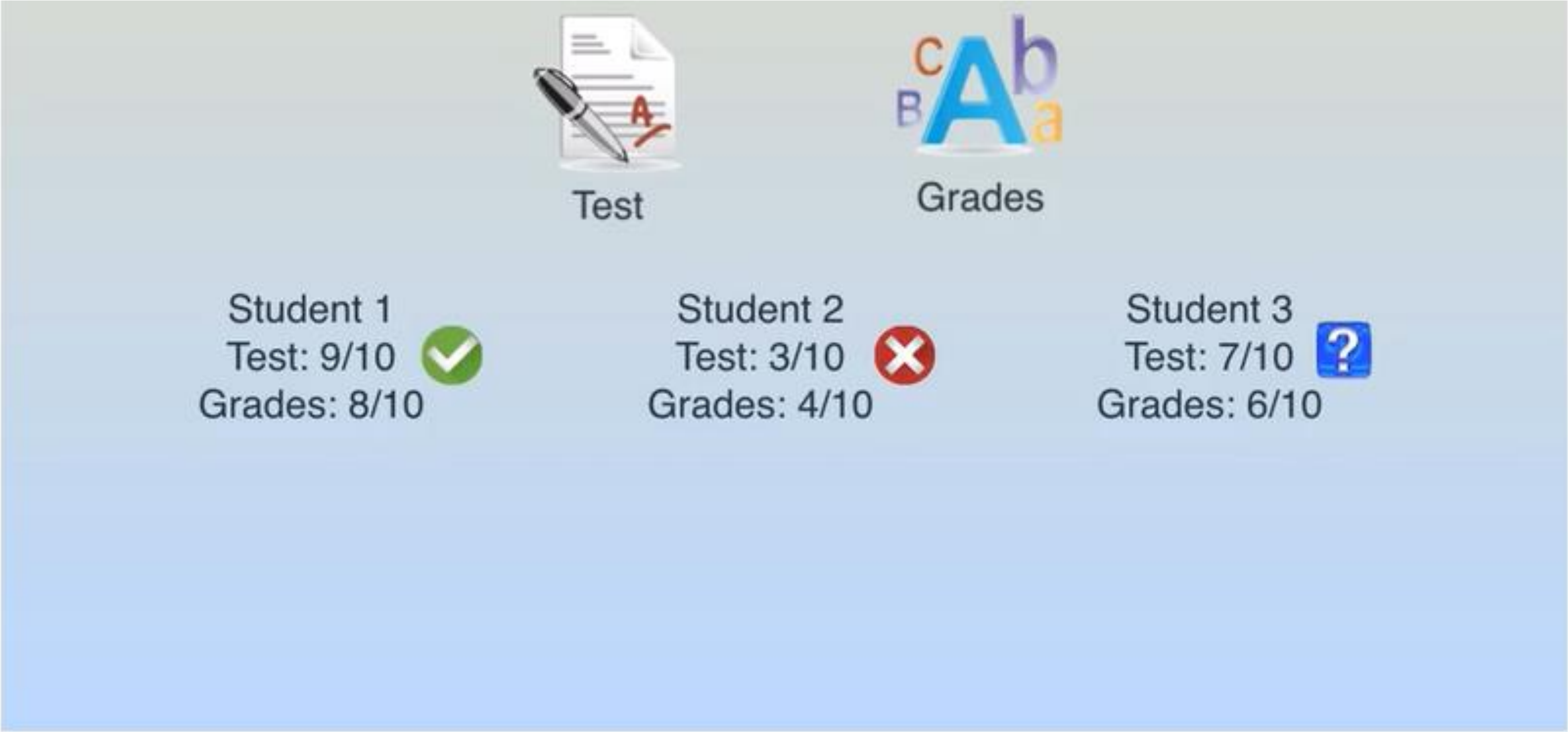
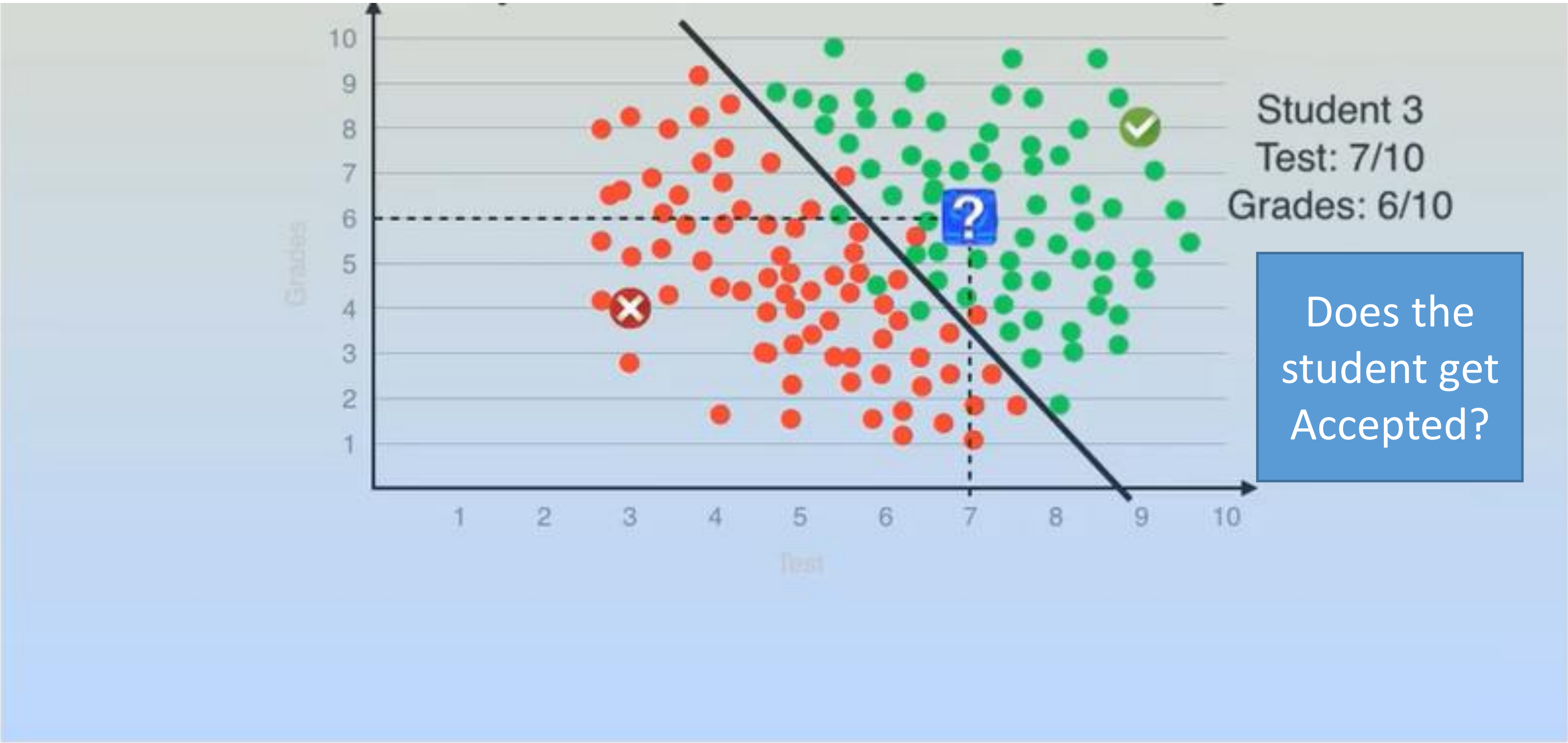
Logistic Regression

Students' Acceptance at a University



Students’ Acceptance at a University

Logistic Regression



Does the student get Accepted?

Logistic Regression

Machine Learning Approach

Logistic regression

- Logistic regression is one of the most popular machine learning algorithms for binary classification.

- It is a simple algorithm that performs very well on a wide range of problems.

Linear Regression:

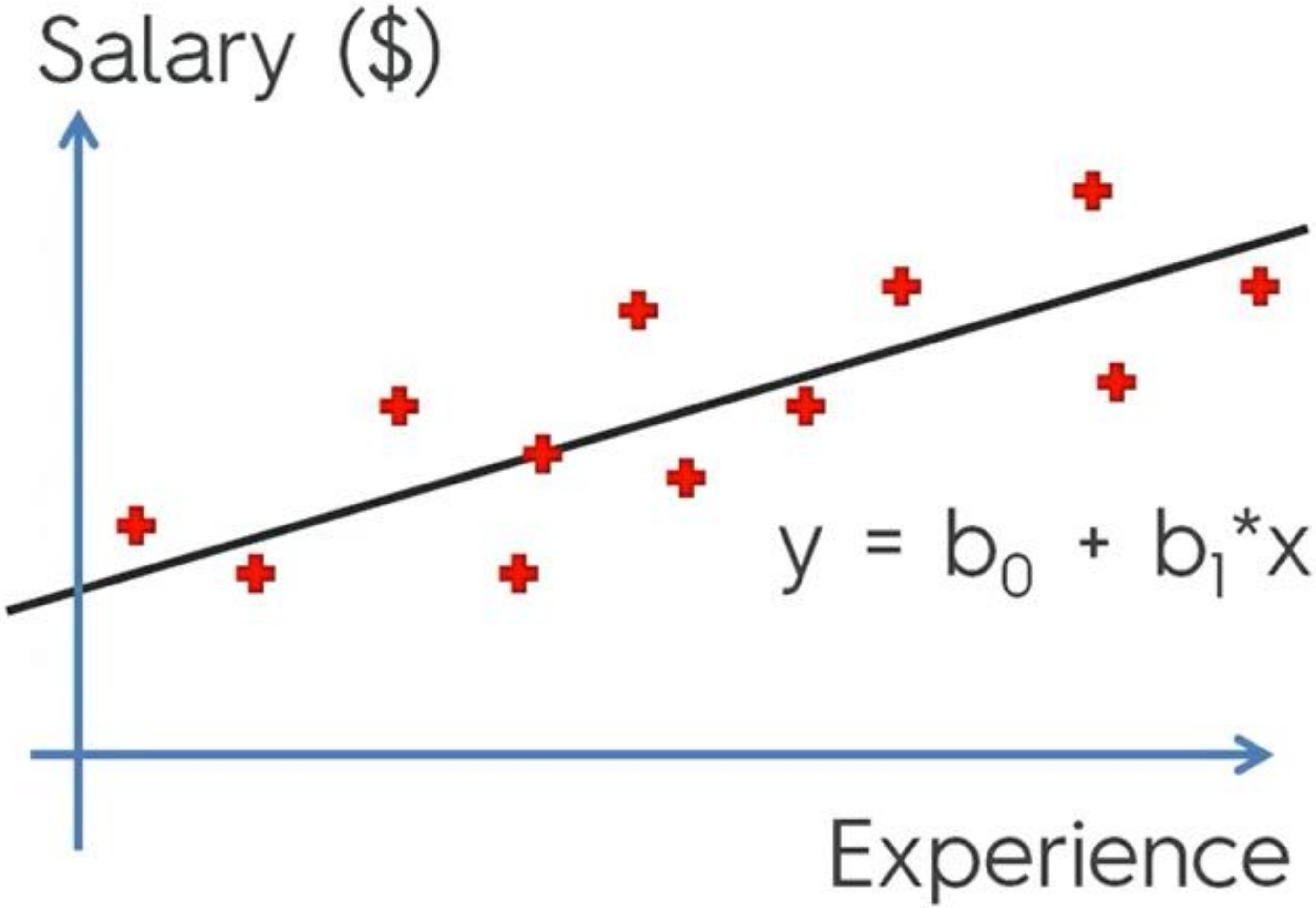
- **Simple:**

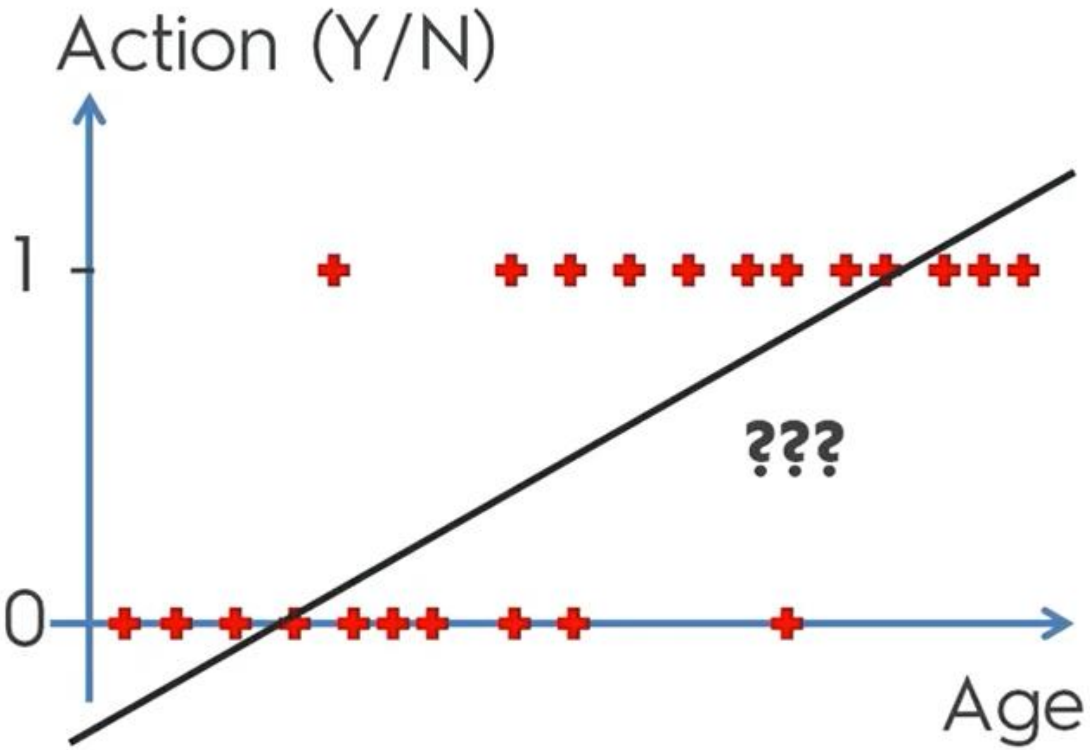
  $$y = b_0 + b_1 * x$$

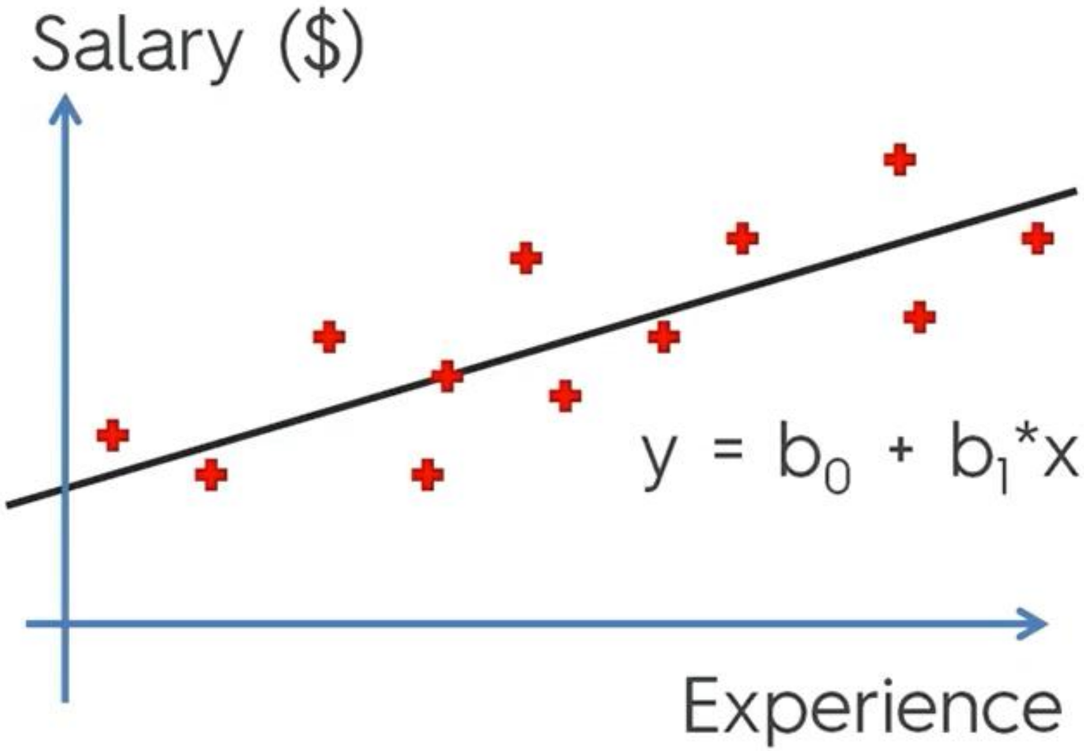
- **Multiple:**

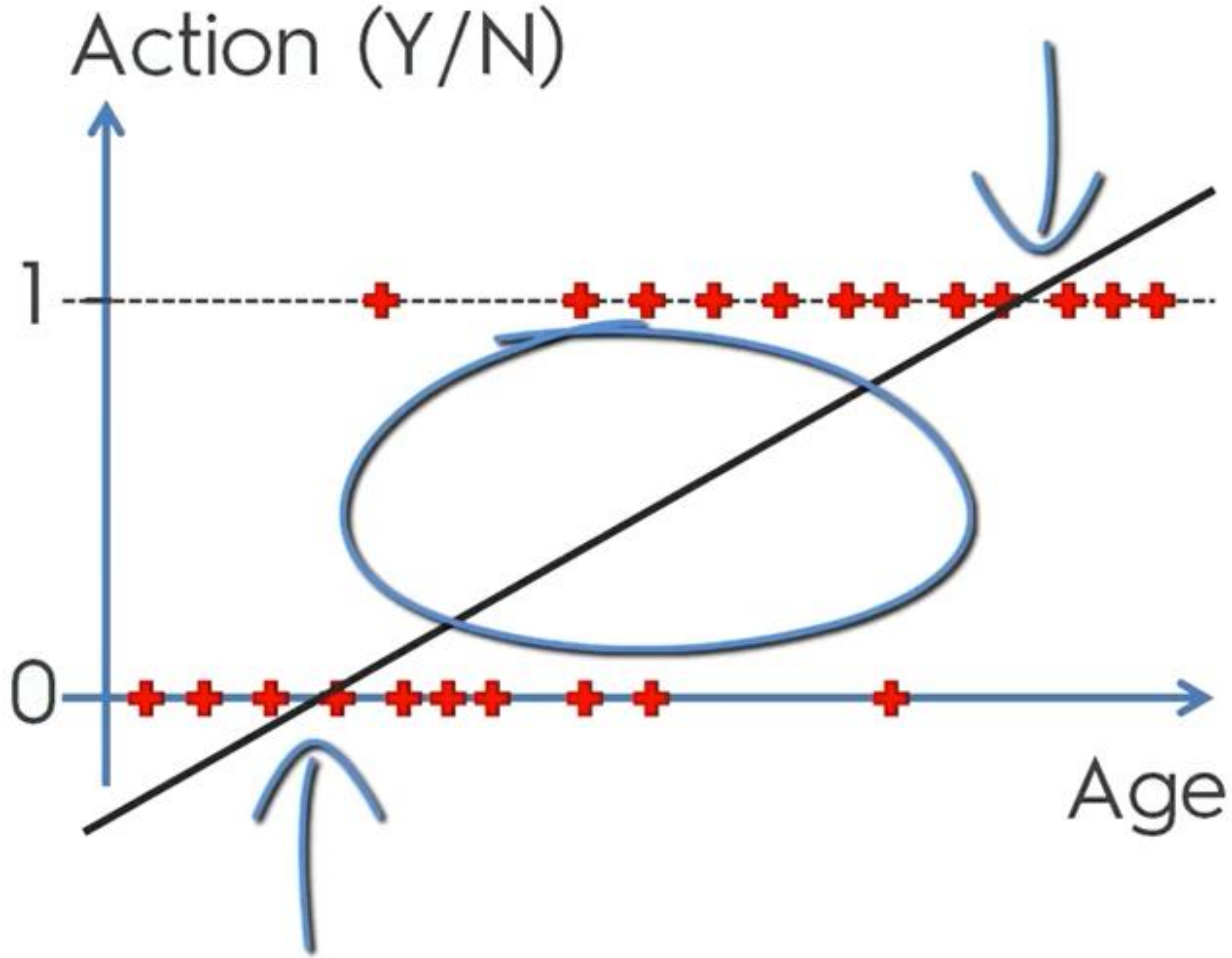
  $$y = b_0 + b_1 * x_1 + \ldots + b_n * x_n$$

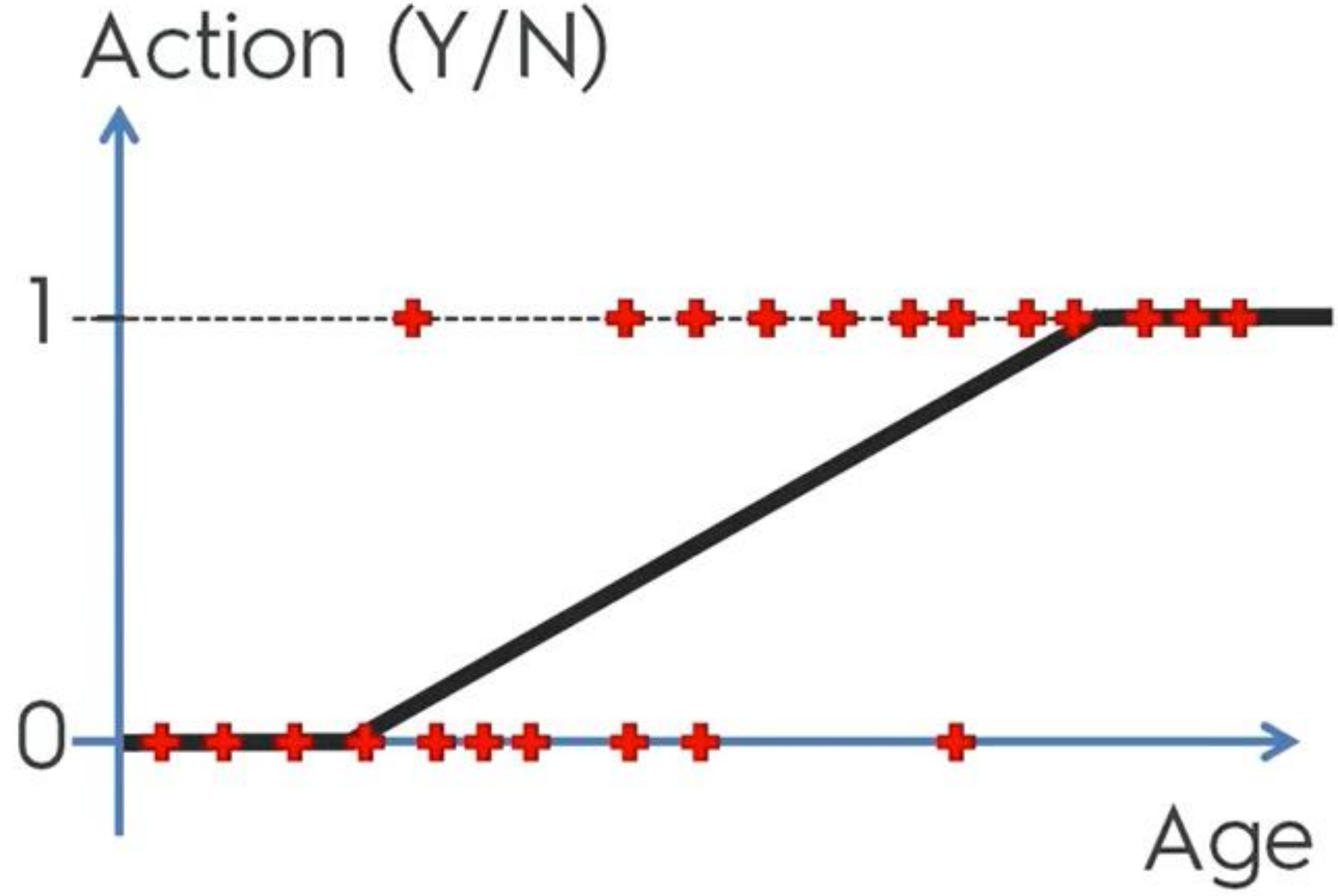
We know this:




Action (Y/N)
1
0
???
Age


Salary ($)
$y = b_0 + b_1 * x$
Experience


Action (Y/N)
1
0
Age

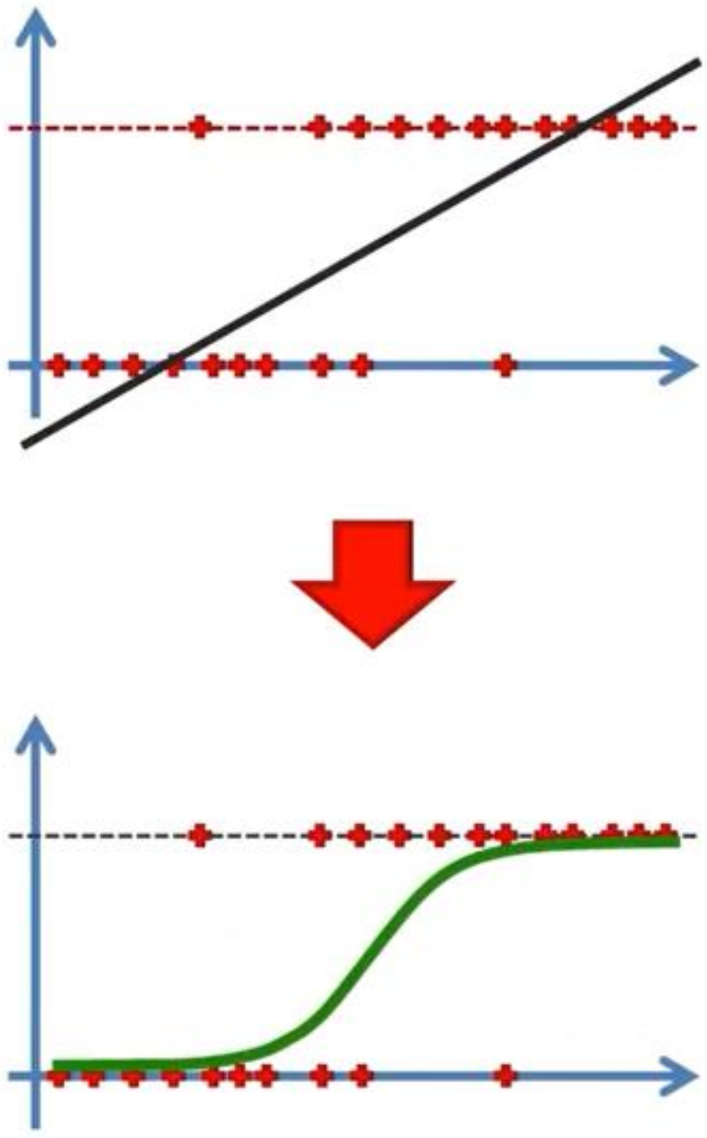

Action (Y/N)
1
0
Age

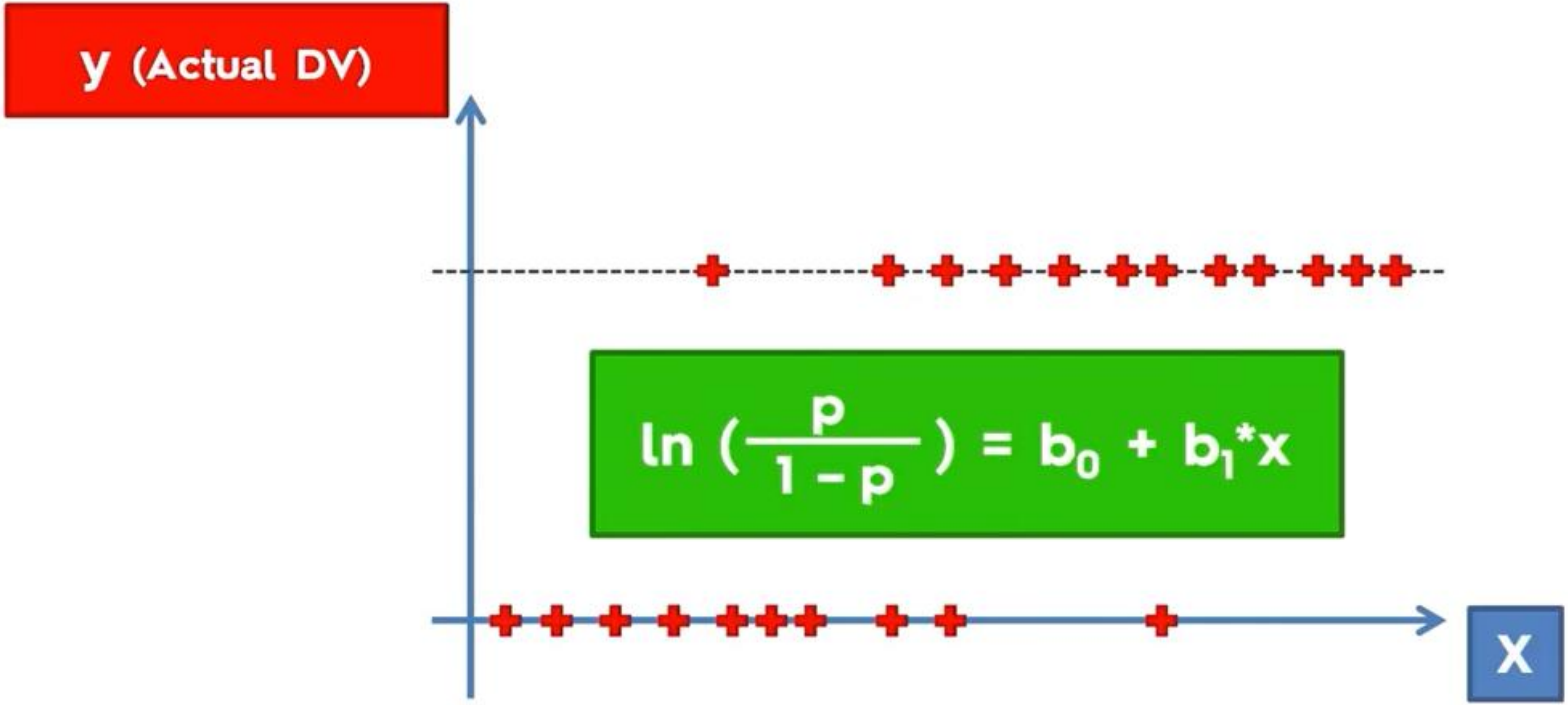
$$y = b_0 + b_1 * x$$

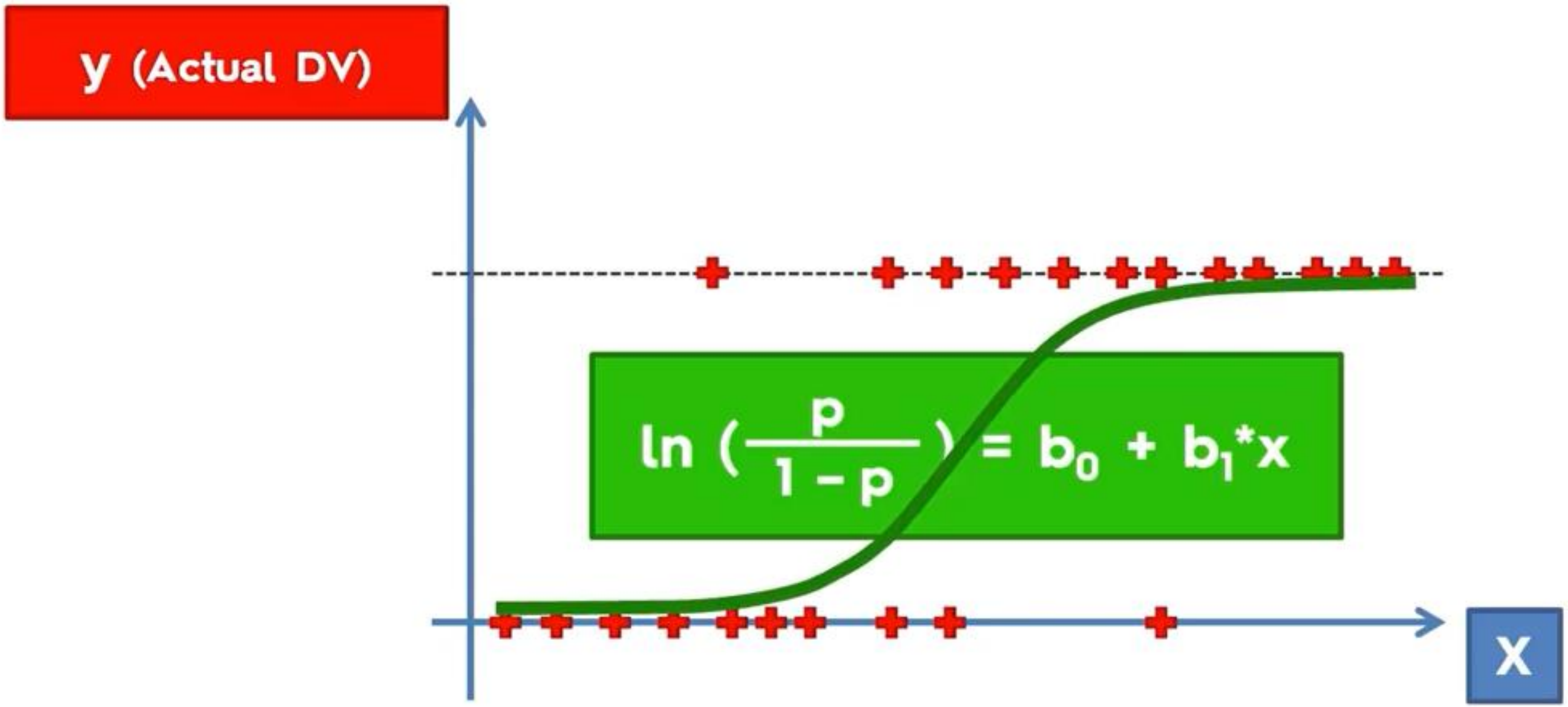
Sigmoid Function

$$p = \frac{1}{1 + e^{-y}}$$

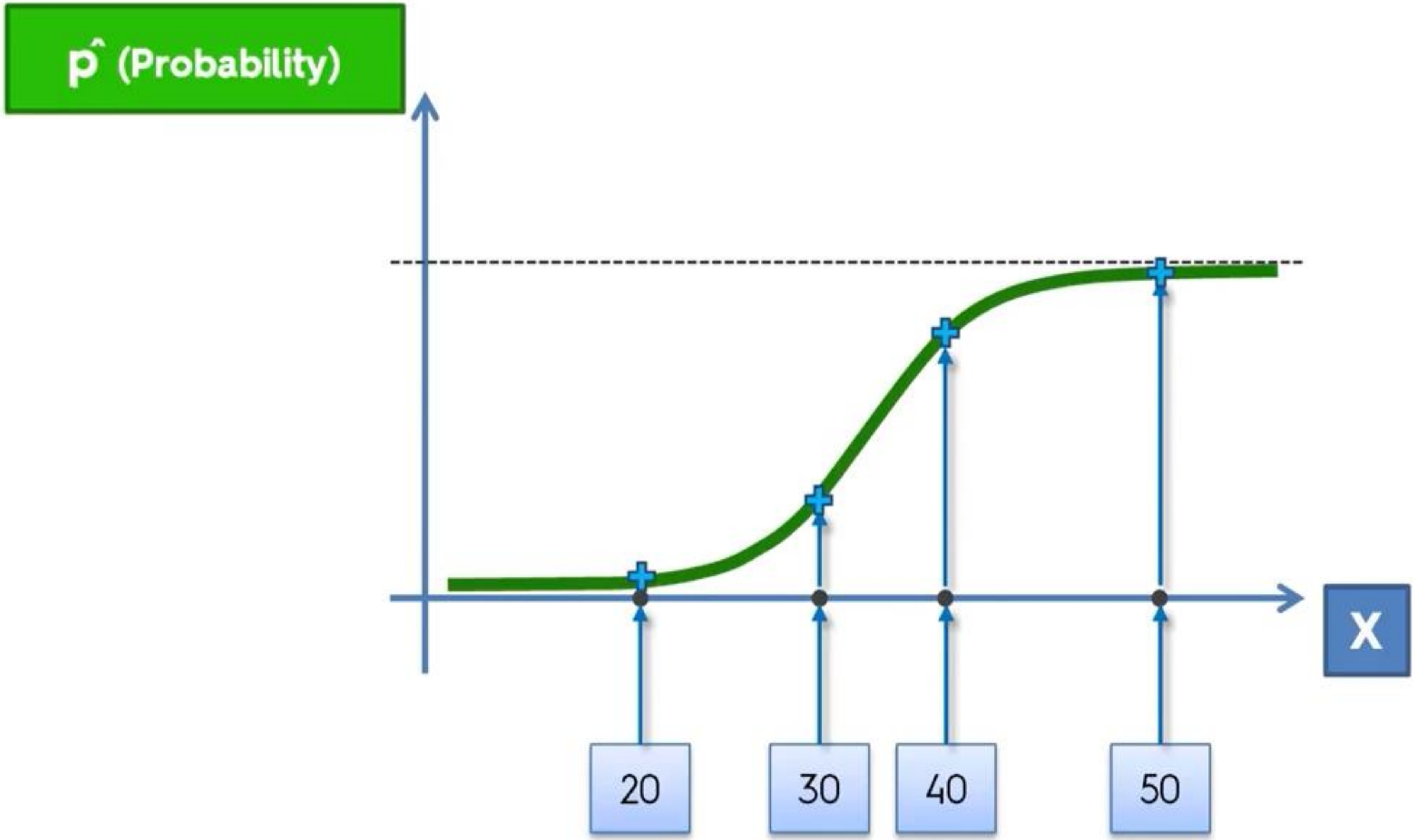
$$\ln\left(\frac{p}{1-p}\right) = b_0 + b_1 * x$$

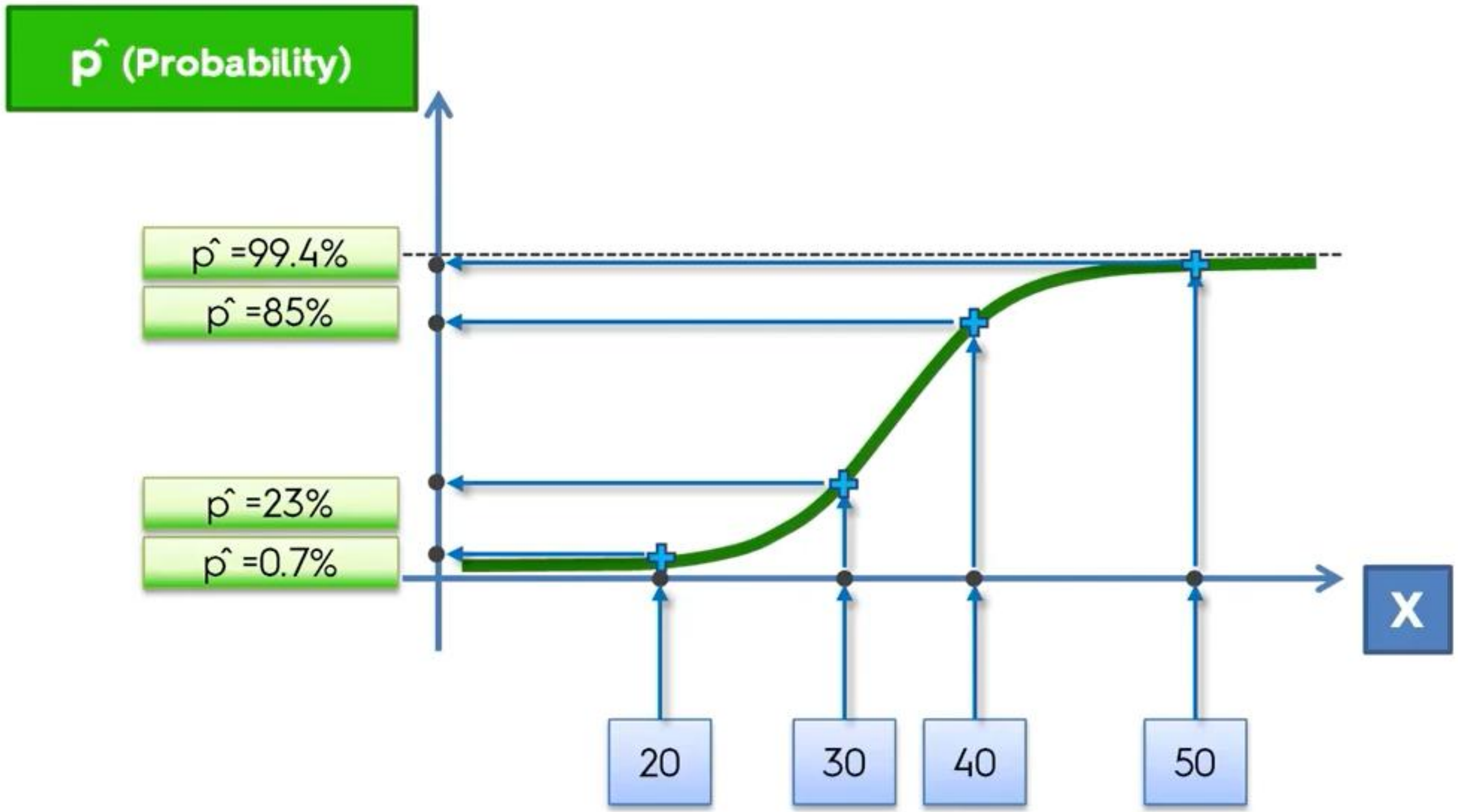


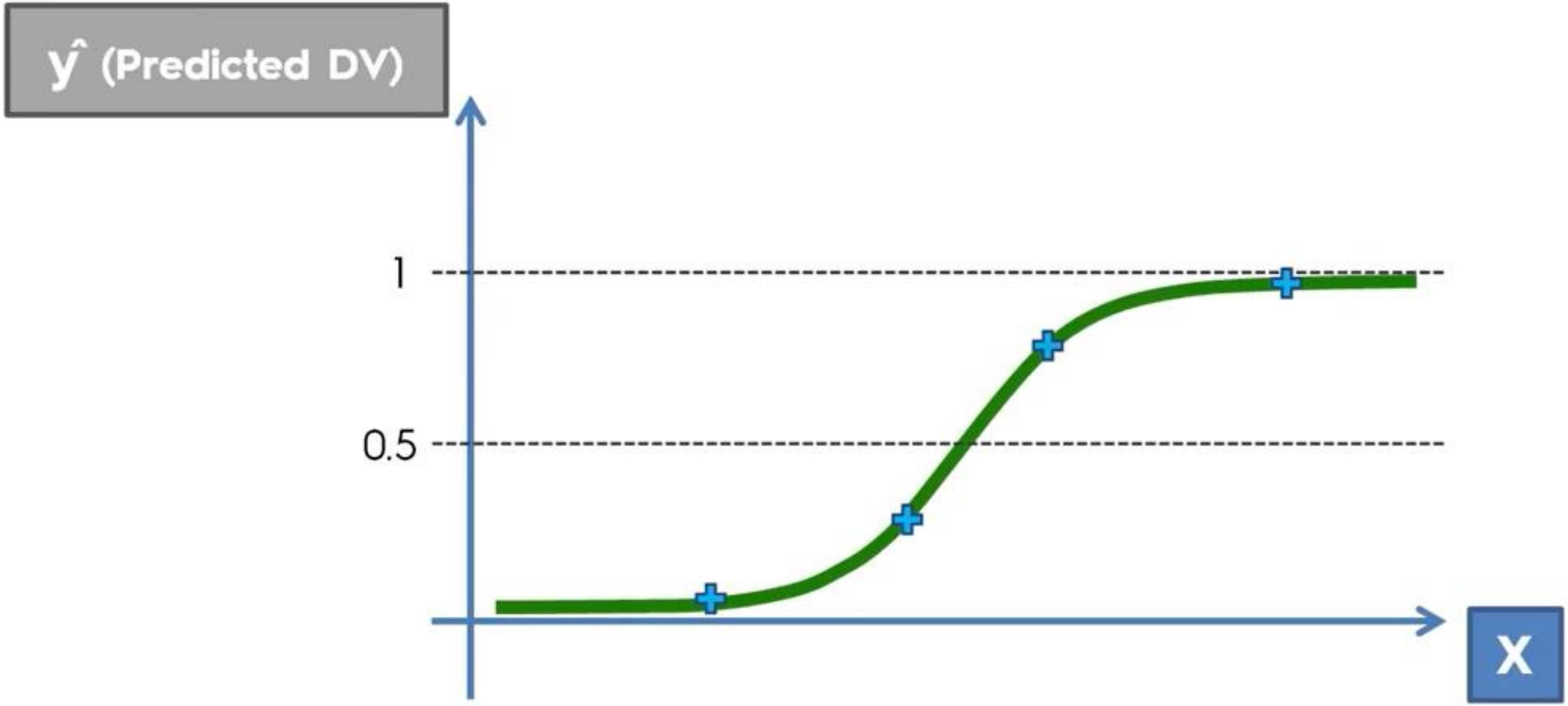

y (Actual DV)
$\ln\left(\frac{p}{1-p}\right) = b_0 + b_1 * x$
x

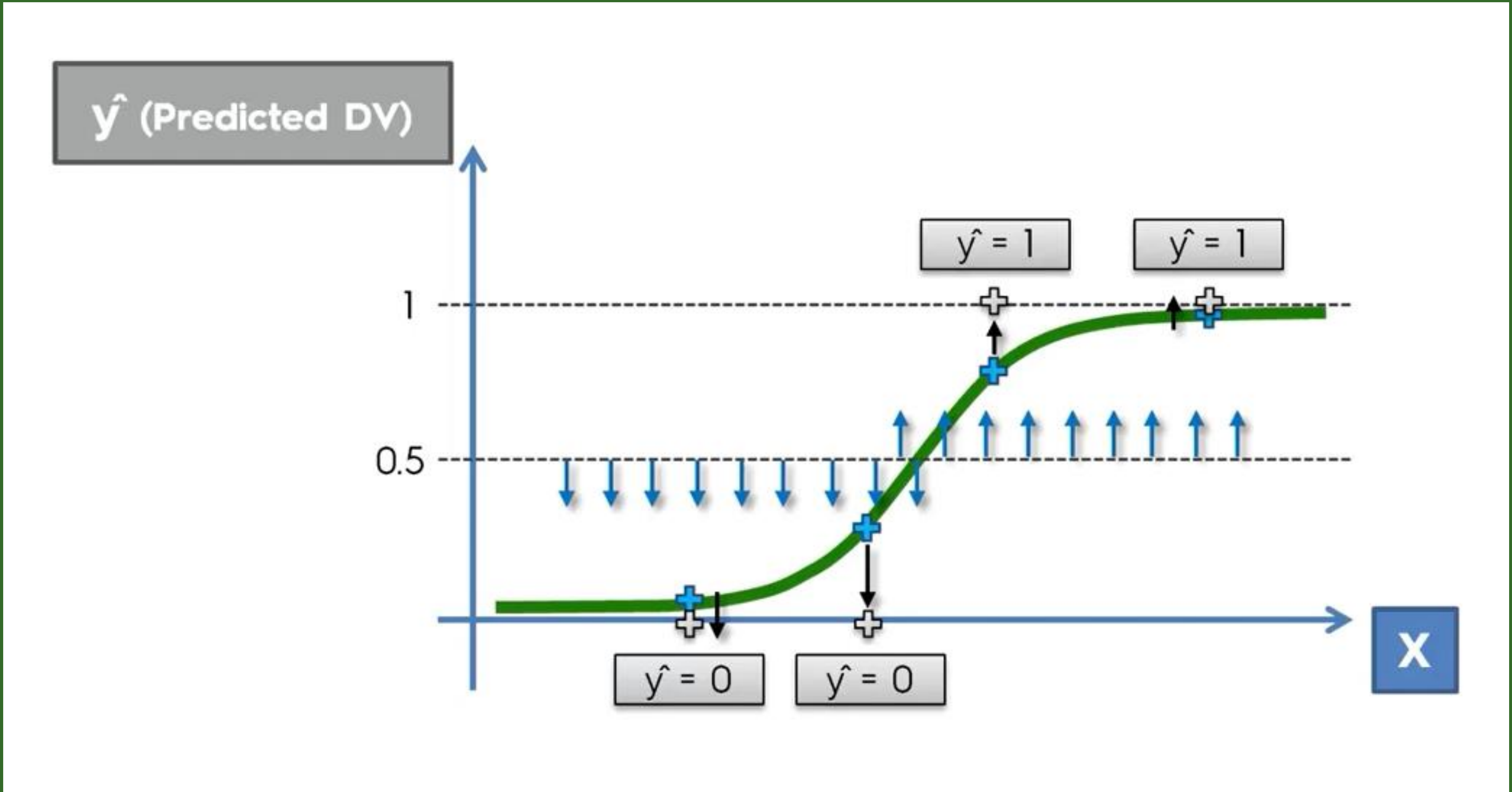


y (Actual DV)
$\ln\left(\frac{p}{1-p}\right) = b_0 + b_1*x$
X


p̂ (Probability)
X
20
30
40
50


p̂ (Probability)
p̂ =99.4%
p̂ =85%
p̂ =23%
p̂ =0.7%
X
20
30
40
50


ŷ (Predicted DV)
1
0.5
X



ŷ (Predicted DV)
ŷ = 1
ŷ = 1
1
0.5
ŷ = 0
ŷ = 0
X

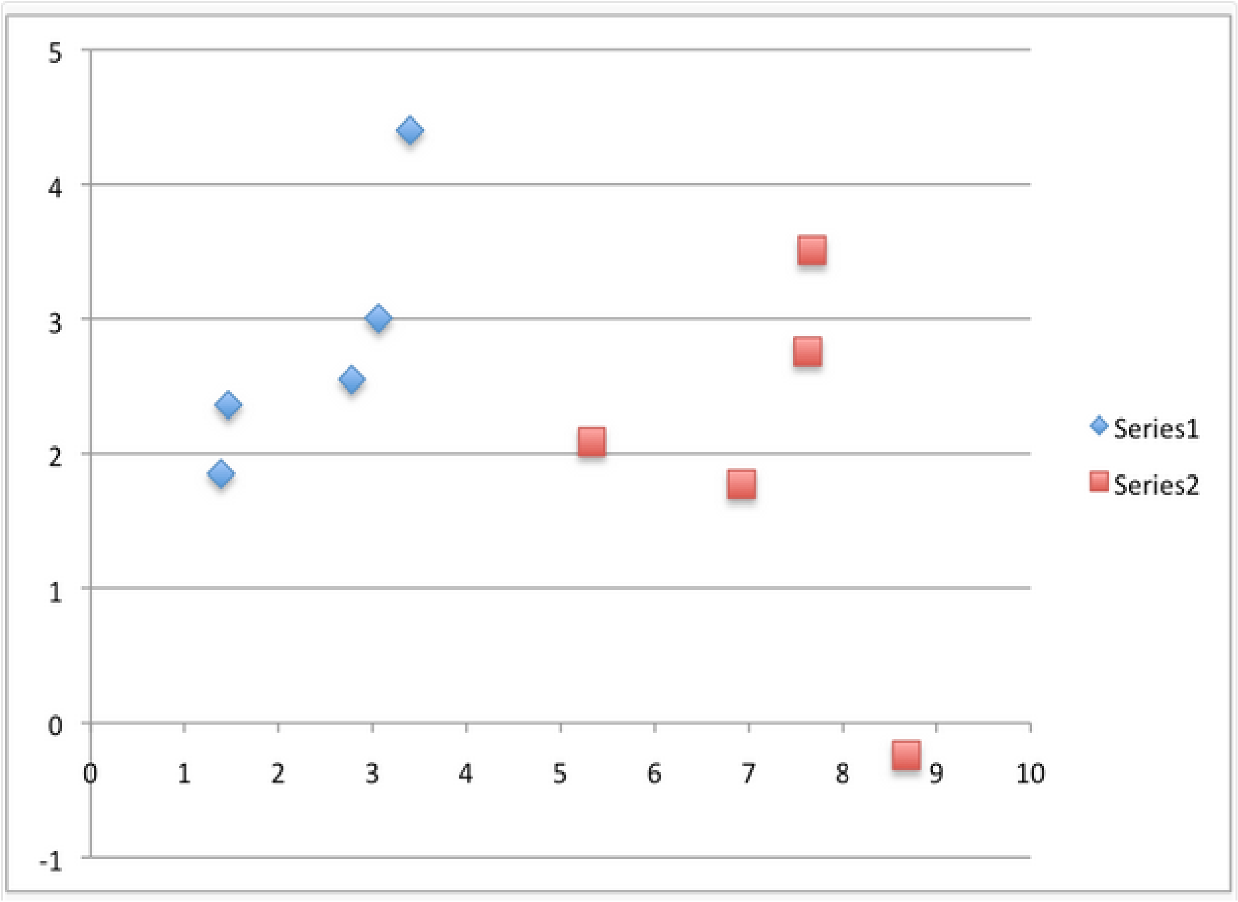
Dataset

- Dataset has two input variables (X1 and X2) and one output variable (Y).
- The input variables are real-valued random numbers drawn from a Gaussian distribution.
- The output variable has two values, making the problem a binary classification problem.

Input X1	Input X2	Actual Output Y
2.7810836	2.550537003	0
1.465489372	2.362125076	0
3.396561688	4.400293529	0
1.38807019	1.850220317	0
3.06407232	3.005305973	0
7.627531214	2.759262235	1
5.332441248	2.088626775	1
6.922596716	1.77106367	1
8.675418651	-0.2420686549	1
7.673756466	3.508563011	1

Plot of the Dataset

- we can easily draw a line to separate the classes. we are going to do with the logistic regression model.



Input X1	Input X2	Actual Output Y
2.7810836	2.550537003	0
1.465489372	2.362125076	0
3.396561688	4.400293529	0
1.38807019	1.850220317	0
3.06407232	3.005305973	0
7.627531214	2.759262235	1
5.332441248	2.088626775	1
6.922596716	1.77106367	1
8.675418651	-0.2420686549	1
7.673756466	3.508563011	1

Logistic Function

- The logistic function is the heart of the logistic regression technique.
- The logistic function is defined as:

$$\text{transformed} = 1 / (1 + e^{-x})$$

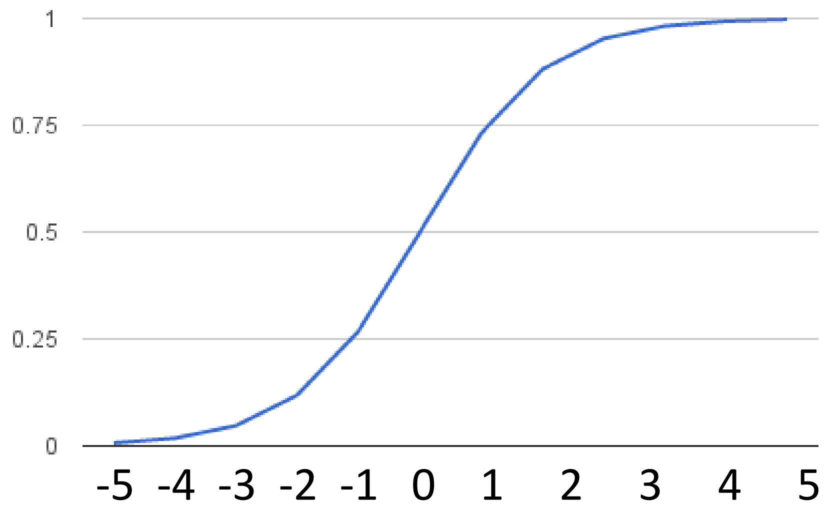
Where e is the numerical constant Euler's number and x is a input we plug into the function.

Logistic Function

- Let's plug in a series of numbers from -5 to +5 and see how the logistic function transforms them.
- You can see that all of the inputs have been transformed into the range [0, 1]
- The smallest negative numbers resulted in values close to zero
- The larger positive numbers resulted in values close to one.
- See that 0 transformed to 0.5 or the midpoint of the new range.

X	Transformed
-5	0.00669285092
-4	0.01798620996
-3	0.04742587318
-2	0.119202922
-1	0.2689414214
0	0.5
1	0.7310585786
2	0.880797078
3	0.9525741268
4	0.98201379
5	0.9933071491

transformed = 1 / (1 + e^-x)



Logistic Regression Model

- The logistic regression model takes real-valued inputs and makes a prediction

Prediction

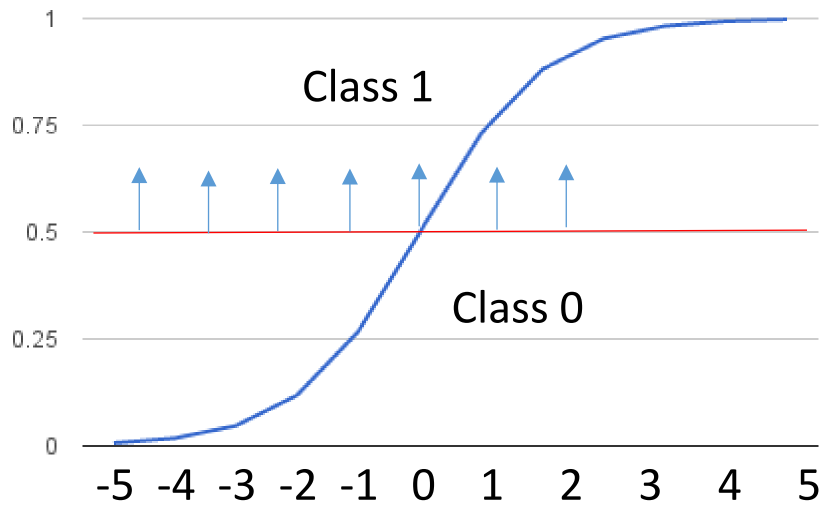
If **transformed** ≥ 0.5

y_pred = class 1

otherwise

y_pred = class 0

transformed = 1 / (1 + e^-x)



Logistic Regression Model

- For this dataset, the logistic regression has three coefficients:

$$\mathbf{x = b0 + b1{*}x_1 + b2{*}x_2}$$

- The job of the learning algorithm will be to discover the best values for the coefficients (b0, b1 and b2) based on the training data.

- The output is transformed into a probability using the logistic function:

$$\mathbf{transformed = 1 / (1 + e\text{^}(-x))}$$

Input X_1	Input X_2	A_Output Y
2.7810836	2.550537003	0
1.465489372	2.362125076	0
3.396561688	4.400293529	0
1.38807019	1.850220317	0
3.06407232	3.005305973	0
7.627531214	2.759262235	1
5.332441248	2.088626775	1
6.922596716	1.77106367	1
8.675418651	-0.2420686549	1
7.673756466	3.508563011	1

Calculate Prediction

Let's start off by assigning 0.0 to each coefficient and calculating the probability of the first training instance that belongs to class 0.

b0 = 0.0; b1 = 0.0; b2 = 0.0

The first training instance is: **x_1=2.7810836**, **x_2=2.550537003**, **x=0**

transformed = 1 / (1 + e^(-x))

Using the above equation - calculate a prediction

transformed = 1 / (1 + e^(-(b0 + b1*x_1 + b2*x_2)))

= 1 / (1 + e^(-(0.0 + 0.0*2.7810836 + 0.0*2.550537003)))

transformed = 0.5

Calculate the new coefficient values using a simple update equation

$b(new) = b(old) + alpha * (x - transformed) * transformed * (1 - transformed) * x_i$

Let's update the coefficients using the prediction (0.5) and coefficient values (0.0) from the previous section.

b0 = 0 + 0.3 * (0 – 0.5) * 0.5 * (1 – 0.5) * 1.0 = -0.0375

b1 = 0 + 0.3 * (0 – 0.5) * 0.5 * (1 – 0.5) * 2.7810836 = -0.104290635

b2 = 0 + 0.3 * (0 – 0.5) * 0.5 * (1 – 0.5) * 2.550537003 = -0.09564513761

Calculate output using new b0, b1 and b2

$x = b0 + b1*x_1 + b2*x_2$

$= -0.0375 - 0.104290635 * 2.7810836 - 0.09564513761 * 2.550537003$

$= -0.5565$

transformed $= 1 / (1 + e^{-x})$

$= 1/ (1+ e^{-(-0.5565)})$

$= 0.364$

If transformed ≥ 0.5

y_pred = class-1

otherwise

y_pred = class-0

Repeat the Process

- We can repeat this process and update the model for each training instance in the dataset.
- A single iteration through the training dataset is called an epoch. It is common to repeat the stochastic gradient descent procedure for a fixed number of epochs.
- At the end of epoch you can calculate error values for the model. Because this is a classification problem, it would be nice to get an idea of how accurate the model is at each iteration.
- The graph below show a plot of accuracy of the model over 10 epochs.

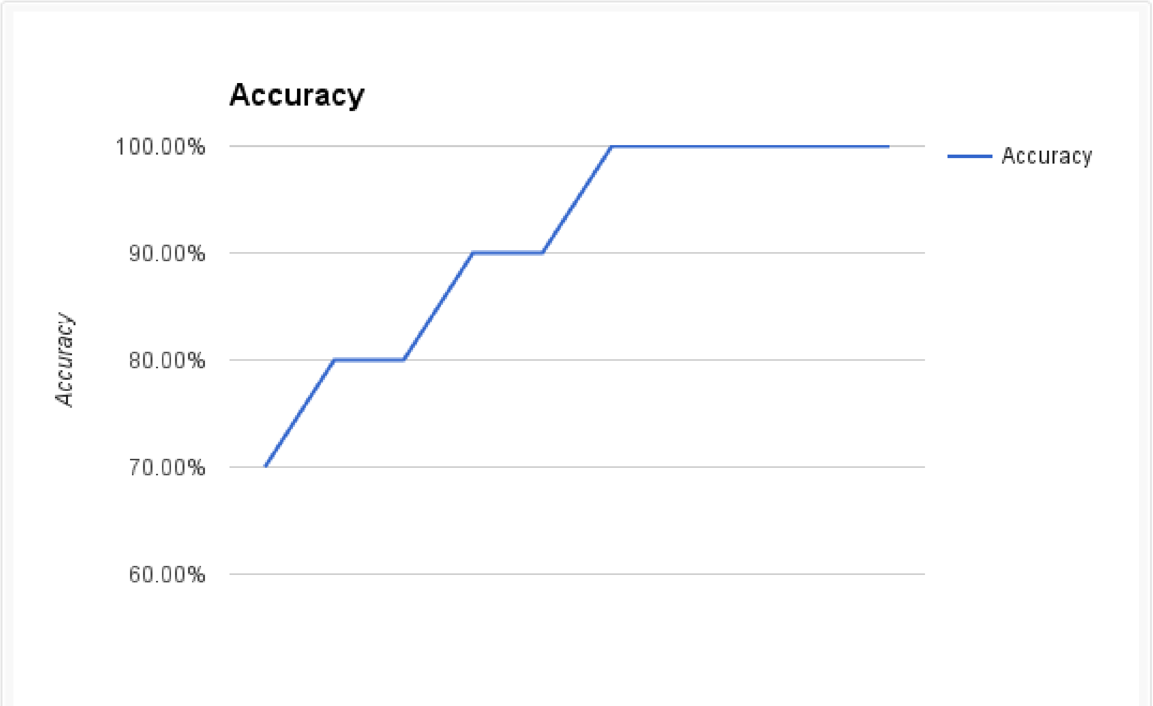
Input X1	Input X2	Actual Output Y	transformed = $1 / (1 + e^{-output})$	P_output = if p(class) < 0.5) then 0 else 1
2.7810836	2.550537003	0	0.2987569857	0
1.465489372	2.362125076	0	0.145951056	0
3.396561688	4.400293529	0	0.08533326531	0
1.38807019	1.850220317	0	0.2197373144	0
3.06407232	3.005305973	0	0.2470590002	0
7.627531214	2.759262235	1	0.9547021348	1
5.332441248	2.088626775	1	0.8620341908	1
6.922596716	1.77106367	1	0.9717729051	1
8.675418651	-0.2420686549	1	0.9992954521	1
7.673756466	3.508563011	1	0.905489323	1

output = b0 + b1*x1 + b2*x2

Compare actual & prediction

- You can see that the model very quickly achieves 100% accuracy on the training dataset.

- The coefficients calculated after 10 epochs of stochastic gradient descent are:



b0 = -0.4066054641

b1 = 0.8525733164

b2 = -1.104746259

Calculate Accuracy

- Finally, we can calculate the accuracy for the model on the training dataset:

accuracy = (correct predictions / num predictions made) * 100

accuracy = (10 /10) * 100

accuracy = 100%

Thank you