

# Introduction to Logistic Regression

## Recall: Linear Regression

**Goal:** Predict a numeric target (regression).

**How to make predictions:** A weighted sum of predictors.

$$\hat{f}(\vec{x}) = \beta_0 + \beta_1 x^{(1)} + \beta_2 x^{(2)} + \dots + \beta_k x^{(k)}$$

**How to find coefficients:** Minimize the RSS/MSE on the available training data.

# Binary Classification Problems

**Classification:** Predicting whether an observation is in a certain category. That is, we are working with a categorical response variable.

**Binary Classification:** Only two possible categories

Eg. “Is it going to rain today?”

# Binary Classification Problems

Just guessing “yes” or “no” is not necessarily the best approach, especially if there is no perfect rule.

A better approach would take noise into account, and not just give a binary answer.

What we want are probabilities, meaning we need to fit a stochastic model!

# Binary Classification Problems

In probability lingo, we want to find  $\Pr(Y|X)$ , the conditional distribution of the response  $Y$ , given the input variables  $X$ .

**Benefits:** can tell how precise our predictions should be.

Eg. our model outputting a 51% chance of rain and it not raining, is better than outputting 99% chance of rain and it not raining.

# Binary Classification Example

Let's build a model to predict the probability that a penguin is of the Gentoo species based off of its body mass.

	body_mass_g	species
149	5700.0	Gentoo
306	3650.0	Chinstrap
88	3300.0	Adelie
315	3325.0	Chinstrap
253	5000.0	Gentoo

How can we make predictions based off of this variable?

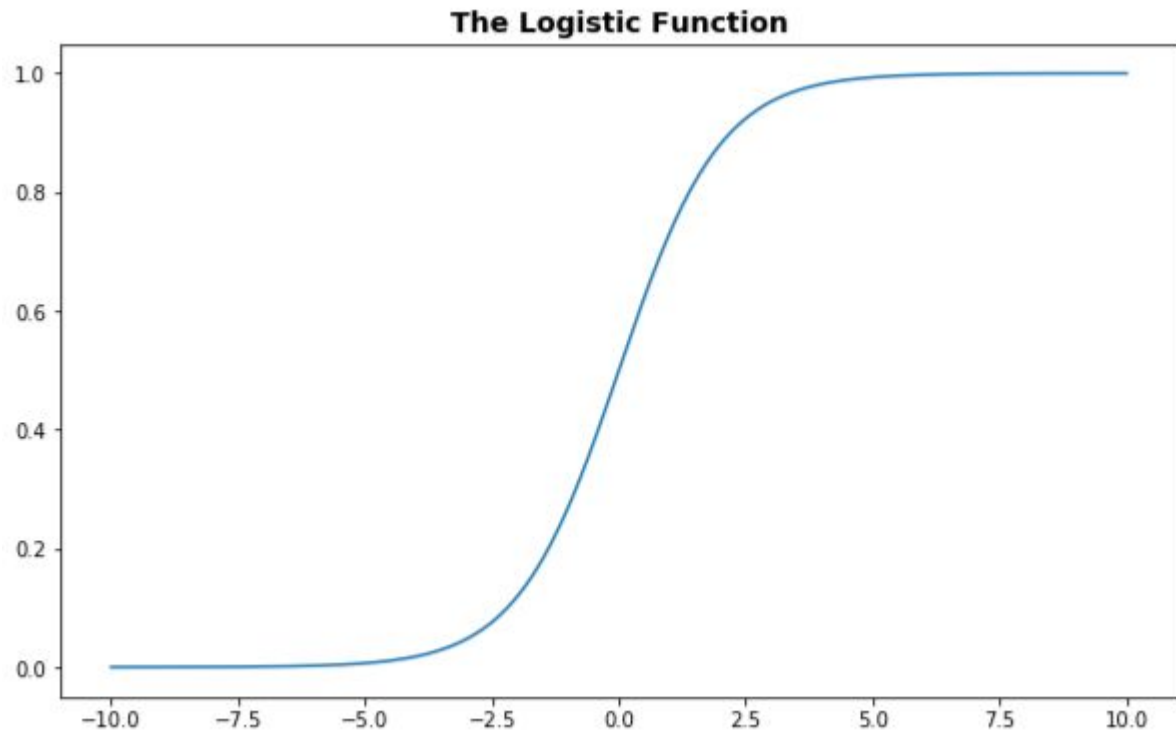
## Idea 1 (the wrong idea): Use a Linear Model

$$P = \beta_0 + \beta_1 \cdot x_1 + \cdots + \beta_k \cdot x_k$$

Here, we are saying that the probability (of True) is a linear function of the predictor variables.

### **Problems:**

- Probabilities must be in  $[0,1]$ , and there is no guarantee of that here
- We won't get "diminishing returns" — changing  $P$  by the same amount requires a bigger change in  $x$  when  $P$  is already large (or small) than when  $P$  is close to  $1/2$ .



$$p(x) = \frac{1}{1 + e^{-x}}$$



# Better Idea: Logistic Regression

## Step 1:

Fit a linear function

$$y = \beta_0 + \beta \cdot \vec{x}$$

## Step 2:

Squash the result into  $[0,1]$  by feeding the output into the logistic function

$$P(\vec{x}) = \frac{1}{1 + e^{-(\beta_0 + \beta \cdot \vec{x})}}$$

## Logistic Regression

$$P(\vec{x}) = \frac{1}{1 + e^{-(\beta_0 + \beta \cdot \vec{x})}}$$

Now, we predict “True” when  $p \geq 0.5$  and “False” if  $p < 0.5$

$$P(\vec{x}) \geq 0.5 \Rightarrow \beta_0 + \beta \cdot \vec{x} \geq 0$$

$$P(\vec{x}) < 0.5 \Rightarrow \beta_0 + \beta \cdot \vec{x} < 0$$

# Binary Classification Example

$$P(\text{species} = \text{Gentoo}) = \frac{1}{1 + \exp(-[-26.92 + 0.00604 \cdot (\text{body mass})])}$$

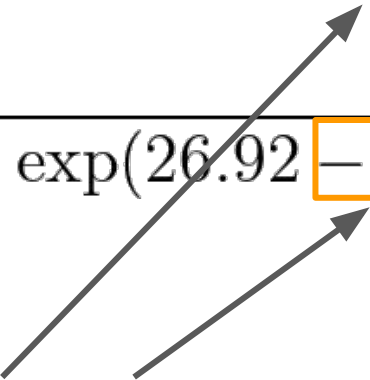
## Binary Classification Example

$$P(\text{species} = \text{Gentoo}) = \frac{1}{1 + \exp(-[-26.92 + 0.00604 \cdot (\text{body mass})])}$$

$$P(\text{species} = \text{Gentoo}) = \frac{1}{1 + \exp(26.92 - 0.00604 \cdot (\text{body mass}))}$$

# Binary Classification Example

$$P(\text{species} = \text{Gentoo}) = \frac{1}{1 + \exp(-[-26.92 + 0.00604 \cdot (\text{body mass})])}$$

$$P(\text{species} = \text{Gentoo}) = \frac{1}{1 + \exp(26.92 - 0.00604 \cdot (\text{body mass}))}$$


As body mass increases, we predict a higher probability of the species being Gentoo.

# Logistic Regression

What is our decision boundary?

$$\beta_0 + \beta \cdot \vec{x} = 0$$

This determines a hyperplane in the predictor space (i.e., a point for one predictor, a line for 2 predictors, a plane for 3 predictors, etc.)

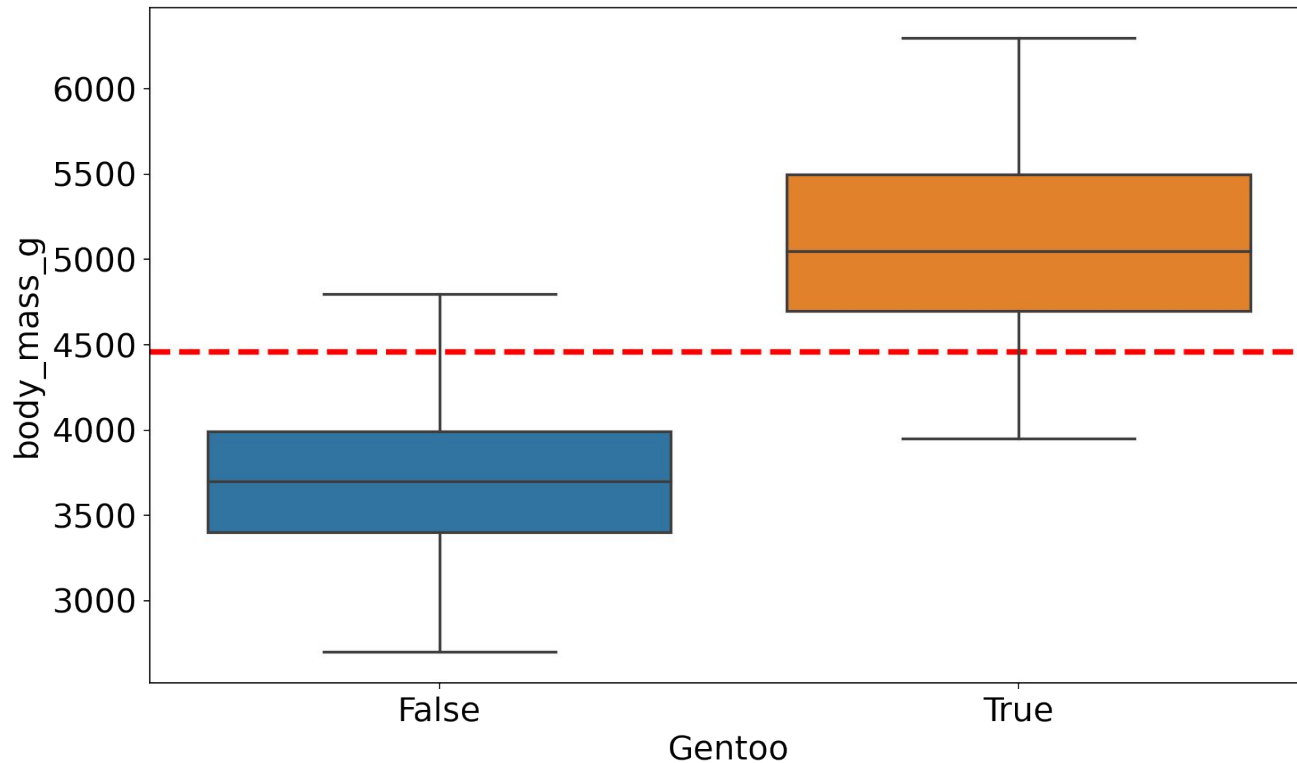
Also, the probability monotonically increases (or decreases) as points move farther away from this hyperplane

## Binary Classification Example

$$P(\text{species} = \text{Gentoo}) = \frac{1}{1 + \exp(-[-26.92 + 0.00604 \cdot (\text{body mass})])}$$

Decision boundary:  $-26.92 + 0.00604 \cdot (\text{body mass}) = 0$   
 $\Rightarrow \text{body mass} = 26.92/0.00604 = 4459$

# Binary Classification Example





# Metrics for Classification

We can understand a lot of metrics through a **confusion matrix** which shows the actual labels vs. predicted labels.

		Predicted	
		negative	positive
Actual	negative	True Negatives (TN)	False Positives (FP)
	positive	False Negatives (FN)	True Positives (TP)

# Metrics for Classification

We can understand a lot of metrics through a **confusion matrix** which shows the actual labels vs. predicted labels.

		Predicted	
		Not Gentoo	Gentoo
Actual	Not Gentoo	51	3
	Gentoo	2	28

# Metrics for Classification

**Accuracy:** What percentage of the time were the predictions correct?

$$\text{accuracy} = \frac{\text{correct predictions}}{\text{total predictions}}$$

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

# Metrics for Classification

		Predicted	
		Not Gentoo	Gentoo
Actual	Not Gentoo	51	3
	Gentoo	2	28

Accuracy: 0.94

$$\text{accuracy} = \frac{\text{correct predictions}}{\text{total predictions}}$$

# Metrics for Classification

**Precision:** Out of those predicted to be in the positive class, what percentage actually are?

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

# Metrics for Classification

		Predicted	
		Not Gentoo	Gentoo
Actual	Not Gentoo	51	3
	Gentoo	2	28

Precision: 0.903

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

# Metrics for Classification

**Recall:** Out of those observation actually in the positive class, what percentage actually are predicted correctly?

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

# Metrics for Classification

		Predicted	
		Not Gentoo	Gentoo
Actual	Not Gentoo	51	3
	Gentoo	2	28

Recall: 0.933

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$



# Metrics for Classification

**F1 Score:** The (harmonic) mean of precision and recall.

$$F1 = \frac{2 \cdot TP}{2 \cdot TP + FP + FN} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

To have a good F1 score, a model must have a good precision and recall score.

# Metrics for Classification

		Predicted	
		Not Gentoo	Gentoo
Actual	Not Gentoo	51	3
	Gentoo	2	28

F1: 0.918

$$F1 = \frac{2 \cdot TP}{2 \cdot TP + FP + FN}$$

# Multiclass Classification

**Question:** What if we have more than one possible target class?  
(Eg. 3 species of penguins)?

# Multiclass Classification

**Question:** What if we have more than one possible target class? (Eg. 3 species of penguins)?

The usual way to handle this is to train a separate classifier for each possible value of the target class and then predict the one with the highest predicted probability. This is called a **one-vs-the rest** approach.

# Multiclass Classification

**Question:** What if we have more than one possible target class? (Eg. 3 species of penguins)?

The usual way to handle this is to train a separate classifier for each possible value of the target class and then predict the one with the highest predicted probability. This is called a **one-vs-the rest** approach.

For logistic regression, there is also a [multinomial approach](#).

# Multiclass Example

If we want to predict a penguin's species using body mass and flipper length, we can fit a separate logistic regression model for each species.

	body_mass_g	flipper_length_mm
29	3325.0	195.0
305	3950.0	210.0
258	5850.0	230.0
318	4050.0	203.0
121	4300.0	195.0

# Multiclass Example

If we want to predict a penguin's species using body mass and flipper length, we can fit a separate logistic regression model for each species.

	body_mass_g	flipper_length_mm	adelie_prob	chinstrap_prob	gentoo_prob
29	3325.0	195.0	0.420950	0.569079	0.009970
305	3950.0	210.0	0.049712	0.644680	0.305608
258	5850.0	230.0	0.000016	0.000504	0.999480
318	4050.0	203.0	0.251547	0.522503	0.225950
121	4300.0	195.0	0.756711	0.148090	0.095198

# Multiclass Example

If we want to predict a penguin's species using body mass and flipper length, we can fit a separate logistic regression model for each species.

	body_mass_g	flipper_length_mm	adelie_prob	chinstrap_prob	gentoo_prob	prediction
29	3325.0	195.0	0.420950	0.569079	0.009970	Chinstrap
305	3950.0	210.0	0.049712	0.644680	0.305608	Chinstrap
258	5850.0	230.0	0.000016	0.000504	0.999480	Gentoo
318	4050.0	203.0	0.251547	0.522503	0.225950	Chinstrap
121	4300.0	195.0	0.756711	0.148090	0.095198	Adelie



# Multiclass Classification - Metrics

We can still compute accuracy the same way.

For precision, recall, and F1, we can compute these by target category.

You might also look at some kind of averaged score across target classes.

# Multiclass Example

		Predicted		
		Adelie	Chinstrap	Gentoo
Actual	Adelie	34	3	0
	Chinstrap	6	8	3
	Gentoo	0	0	30

# Multiclass Example

	precision	recall	f1-score	support
Adelie	0.85	0.92	0.88	37
Chinstrap	0.73	0.47	0.57	17
Gentoo	0.91	1.00	0.95	30
accuracy			0.86	84
macro avg	0.83	0.80	0.80	84
weighted avg	0.85	0.86	0.84	84

# Multiclass Example

	precision	recall	f1-score	support
Adelie	0.85	0.92	0.88	37
Chinstrap	0.73	0.47	0.57	17
Gentoo	0.91	1.00	0.95	30
accuracy			0.86	84
macro avg	0.83	0.80	0.80	84
weighted avg	0.85	0.86	0.84	84

unweighted average  
across all categories

# Multiclass Example

	precision	recall	f1-score	support
Adelie	0.85	0.92	0.88	37
Chinstrap	0.73	0.47	0.57	17
Gentoo	0.91	1.00	0.95	30
accuracy			0.86	84
macro avg	0.83	0.80	0.80	84
weighted avg	0.85	0.86	0.84	84



weighted by number of  
observations per category

$$P(\vec{x}) = \frac{1}{1 + e^{-(\beta_0 + \beta \cdot \vec{x})}}$$

**Question:** How do we determine the value of the coefficients?

**Warning:** Math Incoming

# Logistic Regression

How do we determine the parameter values? By maximizing the “likelihood function”.

$$L(\vec{\beta}) = \prod_{i=1}^n p(x_i)^{y_i} (1 - p(x_i))^{1-y_i}$$

# Logistic Regression

How do we determine the parameter values? By maximizing the “likelihood function”.

$$L(\vec{\beta}) = \prod_{i=1}^n p(x_i)^{y_i} (1 - p(x_i))^{1-y_i}$$

If the observation is actually in class 1 (True/ $y_i=1$ ), then we get “credit” for the predicted probability of being in class 1.



# Logistic Regression

How do we determine the parameter values? By maximizing the “likelihood function”.

$$L(\vec{\beta}) = \prod_{i=1}^n p(x_i)^{y_i} (1 - p(x_i))^{1-y_i}$$

If the observation is actually in class 0 (False/ $y_i = 0$ ), then we get “credit” for the predicted probability of being in class 0 (which is 1 - probability assigned to class 1).

# Logistic Regression

How do we determine the parameter values? By maximizing the “likelihood function”.

$$L(\vec{\beta}) = \prod_{i=1}^n p(x_i)^{y_i} (1 - p(x_i))^{1-y_i}$$

Overall, we do better by assigning high probabilities to the correct classes.

# Logistic Regression

Let's say we are trying to build a model to predict rain/no rain, and this is our training data (with some unspecified predictor variables).

We have two possible models we will choose from, whose predicted probabilities are shown in the next two slides.

Day	True Result
1	Rain
2	No Rain
3	No Rain
4	Rain
5	No Rain

# Logistic Regression - Model A

Day	True Result	Predicted Probability of Rain	Predicted Probability of No Rain	Prediction
1	Rain	0.78	0.22	Rain
2	No Rain	0.15	0.85	No Rain
3	No Rain	0.62	0.38	Rain
4	Rain	0.95	0.05	Rain
5	No Rain	0.25	0.75	No Rain

$$\text{Accuracy} = 4/5 = 80\%$$

# Logistic Regression - Model B

Day	True Result	Predicted Probability of Rain	Predicted Probability of No Rain	Prediction
1	Rain	0.55	0.45	Rain
2	No Rain	0.48	0.52	No Rain
3	No Rain	0.45	0.55	No Rain
4	Rain	0.60	0.40	Rain
5	No Rain	0.39	0.61	No Rain

Accuracy =  $5/5 = 100\%$

# Evaluating Models

From the point of view of accuracy, Model B is superior.

However, this is not how logistic regression models are fit.

Logistic regression models maximize *likelihood*, which rewards high confidence when you're right, and lower confidence when your wrong.

It is better to be wrong sometimes, but highly confident when you're right than to be correct all of the time but with low confidence.

# Logistic Regression

Recall that likelihood is calculated by multiplying together the predicted probabilities for the correct class.

$$L(\vec{\beta}) = \prod_{i=1}^n p(x_i)^{y_i} (1 - p(x_i))^{1-y_i}$$

# Logistic Regression - Model A

Day	True Result	Predicted Probability of Rain	Predicted Probability of No Rain	Prediction
1	Rain	0.78	0.22	Rain
2	No Rain	0.15	0.85	No Rain
3	No Rain	0.62	0.38	Rain
4	Rain	0.95	0.05	Rain
5	No Rain	0.25	0.75	No Rain

$$\text{Likelihood} = (0.78) * (0.85) * (0.38) * (0.95) * (0.75) = 0.180$$



# Logistic Regression - Model B

Day	True Result	Predicted Probability of Rain	Predicted Probability of No Rain	Prediction
1	Rain	0.55	0.45	Rain
2	No Rain	0.48	0.52	No Rain
3	No Rain	0.45	0.55	No Rain
4	Rain	0.60	0.40	Rain
5	No Rain	0.39	0.61	No Rain

$$\text{Likelihood} = (0.55) * (0.52) * (0.55) * (0.60) * (0.61) = 0.058$$

# Evaluating Models

This shows that model A is superior, when using the likelihood metric.

When fitting a logistic regression model, we are searching for the coefficient/weight values that produce the highest possible value for likelihood.

# Logistic Regression - How to Find Coefficients

Since products of small things can get quite small, we usually maximize the *log* likelihood instead of the likelihood.

$$L(\vec{\beta}) = \prod_{i=1}^n p(x_i)^{y_i} (1 - p(x_i))^{1-y_i}$$



$$\log L(\vec{\beta}) = \sum_{i=1}^n (y_i \cdot \log(p(x_i)) + (1 - y_i) \cdot \log(1 - p(x_i)))$$

# Logistic Regression

When finding the parameter values, it is usually impossible to find the true optimal solution, so instead we must rely on a numerical approximation.

The default one used by scikit-learn is the L-BFGS (Limited-Memory Broyden–Fletcher–Goldfarb–Shanno algorithm), but there are others.

This means that we will not necessarily find the optimal coefficients, but under most conditions will get an acceptable solution.

Note: By default, scikit-learn is optimizing a slight variant of the log-likelihood, so if you compare the coefficients from scikit-learn to other libraries like statsmodels, you'll get different results. We'll learn more about this soon.