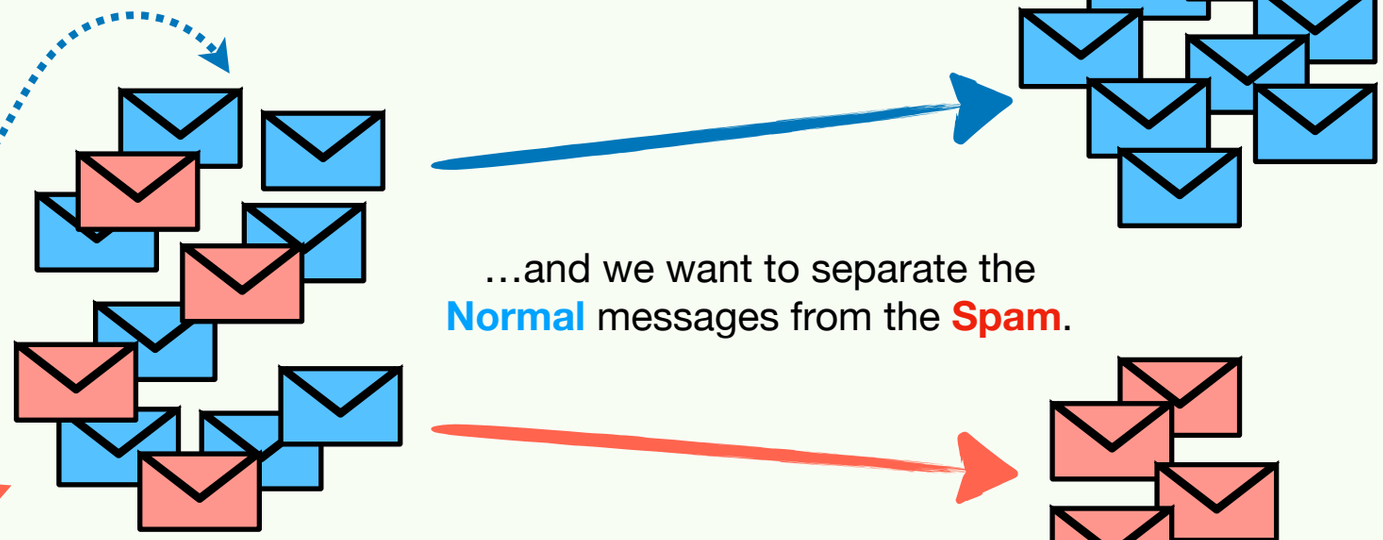
Naive Bayes: Main Ideas

1 **The Problem:** We start with a jumble of messages: some of them are **Normal** messages from friends and family…

…and some of them are **Spam**, unwanted messages that are usually scams or unsolicited advertisements…



…and we want to separate the **Normal** messages from the **Spam**.

2 **A Solution:** We can use a **Naive Bayes** classifier, one of the simplest, but surprisingly most effective machine learning methods.

3 Let's say we get a message that says **Dear Friend**, and we want to classify it as either **Normal** or **Spam**. We'll need two different equations: one for **Normal** and one for **Spam**.

4 First we multiply the **Prior Probability**, which is an initial guess, that the message is **Normal**…

…by the probabilities of seeing the words **Dear** and **Friend**, *given* (**NOTE:** the vertical bar, **|**, stands for **given**) that the message is **Normal**.

$$p(\mathbf{N}) \times p(\mathbf{Dear} \mid \mathbf{N}) \times p(\mathbf{Friend} \mid \mathbf{N})$$

$$p(\mathbf{S}) \times p(\mathbf{Dear} \mid \mathbf{S}) \times p(\mathbf{Friend} \mid \mathbf{S})$$

5 Then we compare that to the **Prior Probability**, another initial guess, that the message is **Spam**…

…multiplied by the probabilities of seeing the words **Dear** and **Friend**, given that the message is **Spam**.

6 Whichever equation gives us the larger value, which we'll call a score, tells us the final classification.

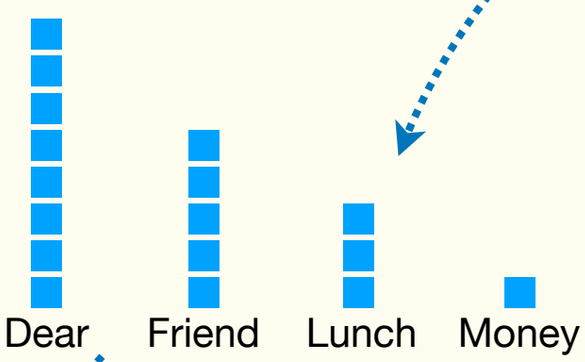
Multinomial Naive Bayes: Details Part 1

1. There are several types of **Naive Bayes** algorithms, but the most commonly used version is called **Multinomial Naive Bayes**.

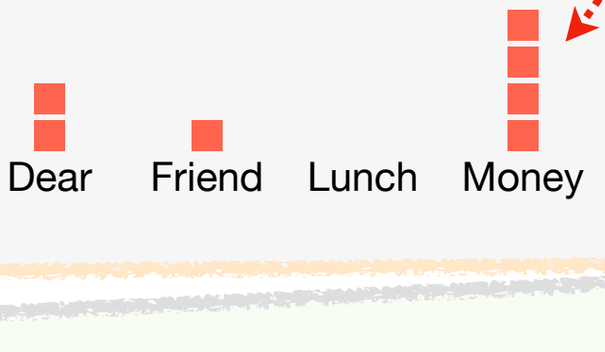
2. We start with **Training Data**: **8** messages that we know are **Normal**…

…and **4** messages that we know are **Spam**.

3. Then we make a histogram for all of the words in the **Normal** messages…



4. …and a histogram for all of the words in the **Spam** messages.



5. Now we calculate the probabilities of seeing each word given that they came from **Normal** messages.

For example, the probability that we see the word **Dear** *given* (remember, the vertical bar, **|**, stands for **given**) that we see it in **Normal** (**N**) messages…

$$p(\textbf{Dear} \mid \textbf{N}) = \frac{8}{17} = 0.47$$

…is **8**, the number of times **Dear** occurs in **Normal** messages, divided by the total number of words in the **Normal** messages, **17**…

…and we get **0.47**.

Then we do the same thing for all the other words in the **Normal** messages.

$p(\textbf{Dear} \mid \textbf{N}) = 0.47$

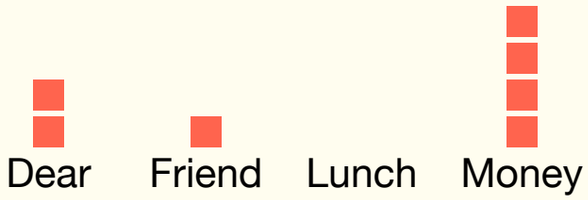
$p(\textbf{Friend} \mid \textbf{N}) = 0.29$

$p(\textbf{Lunch} \mid \textbf{N}) = 0.18$

$p(\textbf{Money} \mid \textbf{N}) = 0.06$

Multinomial Naive Bayes: Details Part 2

6 Then we calculate the probabilities of seeing each word given that they came from **Spam** messages.



For example, the probability that we see the word **Dear** given that we see it in **Spam** (**S**) messages…

$$p(\textbf{Dear} \mid \textbf{S}) = \frac{2}{7} = 0.29$$

…is **2**, the number of times **Dear** occurs in **Spam** messages divided by the total number of words in the **Spam** messages, **7**…

…and we get **0.29**.

Then we do the same thing for all the other words in the **Spam** messages.

$p(\textbf{Dear} \mid \textbf{S}) = 0.29$

$p(\textbf{Friend} \mid \textbf{S}) = 0.14$

$p(\textbf{Lunch} \mid \textbf{S}) = 0.00$

$p(\textbf{Money} \mid \textbf{S}) = 0.57$

7 Now we calculate **Prior Probabilities**. In this context, a **Prior Probability** is simply a guess that we make without looking at the words in the message about whether or not a message is **Normal** or **Spam**.

NOTE: The **Prior Probabilities** can be any pair of probabilities we want, but we usually derive them from the **Training Data**.

8 For example, because **8** of the **12** messages are **Normal**, we let the **Prior Probability** for **Normal** messages, $p(\textbf{N})$, be **8/12 = 0.67**…

$$p(\textbf{N}) = \frac{\#\text{ of Normal Messages}}{\text{Total \# of Messages}} = \frac{8}{12} = 0.67$$

9 …and because **4** of the **12** messages are **Spam**, we let the **Prior Probability** for **Spam** messages, $p(\textbf{S})$, be **4/12 = 0.33**.

$$p(\textbf{S}) = \frac{\#\text{ of Spam Messages}}{\text{Total \# of Messages}} = \frac{4}{12} = 0.33$$

Multinomial Naive Bayes: Details Part 3

10 Now that we have the **Prior Probability** for **Normal** messages…

$p(\text{N}) = 0.67$

…and the probabilities for each word occurring in **Normal** messages…

$p(\textbf{Dear} \mid \text{N}) = 0.47$

$p(\textbf{Friend} \mid \text{N}) = 0.29$

$p(\textbf{Lunch} \mid \text{N}) = 0.18$

$p(\textbf{Money} \mid \text{N}) = 0.06$

…we can calculate the overall score that the message, **Dear Friend**, is **Normal**…

$$p(\text{N}) \times p(\textbf{Dear} \mid \text{N}) \times p(\textbf{Friend} \mid \text{N}) = 0.67 \times 0.47 \times 0.29 = 0.09$$

11 …by multiplying the **Prior Probability** that the message is **Normal**…

…by the probabilities of seeing the words **Dear** and **Friend** in **Normal** messages…

…and when we do the math, we get **0.09**.

12 Likewise, using the **Prior Probability** for **Spam** messages…

$p(\text{S}) = 0.33$

…and the probabilities for each word occurring in **Spam** messages…

$p(\textbf{Dear} \mid \text{S}) = 0.29$

$p(\textbf{Friend} \mid \text{S}) = 0.14$

$p(\textbf{Lunch} \mid \text{S}) = 0.00$

$p(\textbf{Money} \mid \text{S}) = 0.57$

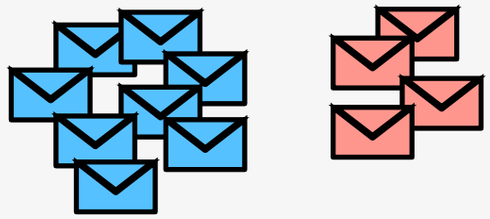
…we can calculate the overall score that the message, **Dear Friend**, is **Spam**, and when we do the math, we get **0.01**.

$$p(\text{S}) \times p(\textbf{Dear} \mid \text{S}) \times p(\textbf{Friend} \mid \text{S}) = 0.33 \times 0.29 \times 0.14 = 0.01$$

Multinomial Naive Bayes: Details Part 4

13 Now, remember the goal was to classify the message **Dear Friend** as either **Normal** or **Spam**…

…and we started with **Training Data**, **8 Normal** and **4 Spam** messages…

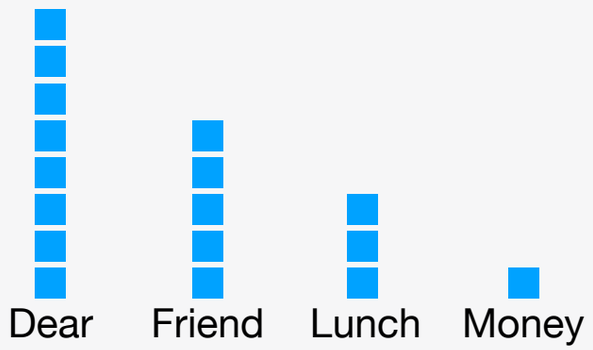


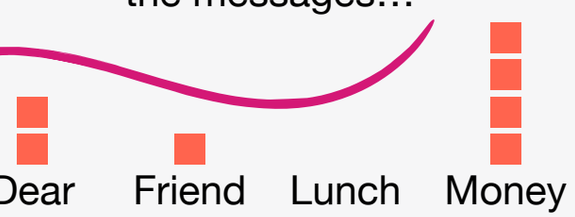
$$p(\text{N}) = \frac{\#\text{ of Normal Messages}}{\text{Total \# of Messages}} = 0.67$$

$$p(\text{S}) = \frac{\#\text{ of Spam Messages}}{\text{Total \# of Messages}} = 0.33$$

…and we calculated **Prior Probabilities**, which are just guesses that we make without looking at the contents of a message, that a message is either **Normal** or **Spam**…

…and we created histograms of the words in the messages…





$p(\textbf{Dear} \mid \text{N}) = 0.47$
$p(\textbf{Friend} \mid \text{N}) = 0.29$
$p(\textbf{Lunch} \mid \text{N}) = 0.18$
$p(\textbf{Money} \mid \text{N}) = 0.06$

…and we used the histograms to calculate probabilities…

$p(\textbf{Dear} \mid \text{S}) = 0.29$
$p(\textbf{Friend} \mid \text{S}) = 0.14$
$p(\textbf{Lunch} \mid \text{S}) = 0.00$
$p(\textbf{Money} \mid \text{S}) = 0.57$

…then, using the **Prior Probabilities** and the probabilities for each word, given that it came from either a **Normal** message or **Spam**, we calculated scores for **Dear Friend**….

$$p(\text{N}) \times p(\textbf{Dear} \mid \text{N}) \times p(\textbf{Friend} \mid \text{N}) = 0.67 \times 0.47 \times 0.29 = 0.09$$

$$p(\text{S}) \times p(\textbf{Dear} \mid \text{S}) \times p(\textbf{Friend} \mid \text{S}) = 0.33 \times 0.29 \times 0.14 = 0.01$$

…and now we can finally classify **Dear Friend**!!! Because the score for **Normal** (**0.09**) is greater than **Spam** (**0.01**), we classify **Dear Friend** as a **Normal** message.

Dear Friend 

BAM!!!

Multinomial Naive Bayes: Details Part 5

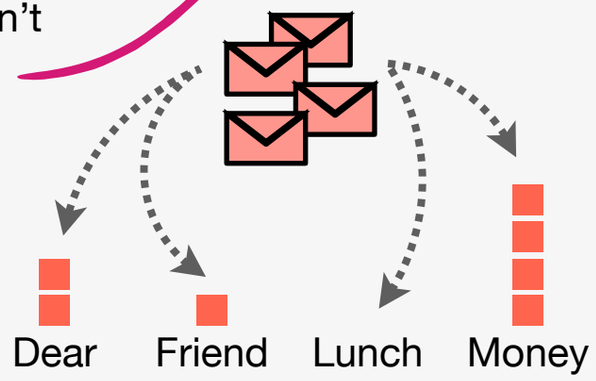
14 The reason **Naive Bayes** is ***naive***, is that it's simple: it treats all of the words independently, and this means it ignores word order and phrasing.

So, both **Dear Friend** and **Friend Dear** get the same score, **0.09**.

$p($ **N** $) \times p($ **Friend** $|$ **N** $) \times p($ **Dear** $|$ **N** $) = 0.67 \times 0.29 \times 0.47 = 0.09$

$p($ **N** $) \times p($ **Dear** $|$ **N** $) \times p($ **Friend** $|$ **N** $) = 0.67 \times 0.47 \times 0.29 = 0.09$

15 Missing data can, in theory, be a problem. Remember, in the last example, the word **Lunch** didn't occur in any of the **Spam**…



…and that means that the probability of seeing **Lunch** in **Spam** is **0**…

$p($ **Dear** $|$ **S** $) = 0.29$

$p($ **Friend** $|$ **S** $) = 0.14$

$p($ **Lunch** $|$ **S** $) = 0.00$

$p($ **Money** $|$ **S** $) = 0.57$

…and this means that any message that contains the word **Lunch** will be classified as **Normal** because the score for **Spam** will always be **0**.

16 For example, if we see this message…

Money Money Money Money Lunch

…which looks like it could be **Spam** because it has the word **Money** in it a bunch of times…

…then, when we calculate this score…

$p($ **S** $) \times p($ **Money** $|$ **S** $) \times p($ **Money** $|$ **S** $) \times p($ **Money** $|$ **S** $) \times p($ **Money** $|$ **S** $) \times p($ **Lunch** $|$ **S** $)$

…because $p($ **Lunch** $|$ **S** $) =$ **0**, and anything times **0** is **0**, we get **0**.

$= 0.33 \times 0.57 \times 0.57 \times 0.57 \times 0.57 \times 0 = 0$

Good thing there's an easy way to deal with the problem of missing data!!! Read on for the solution.

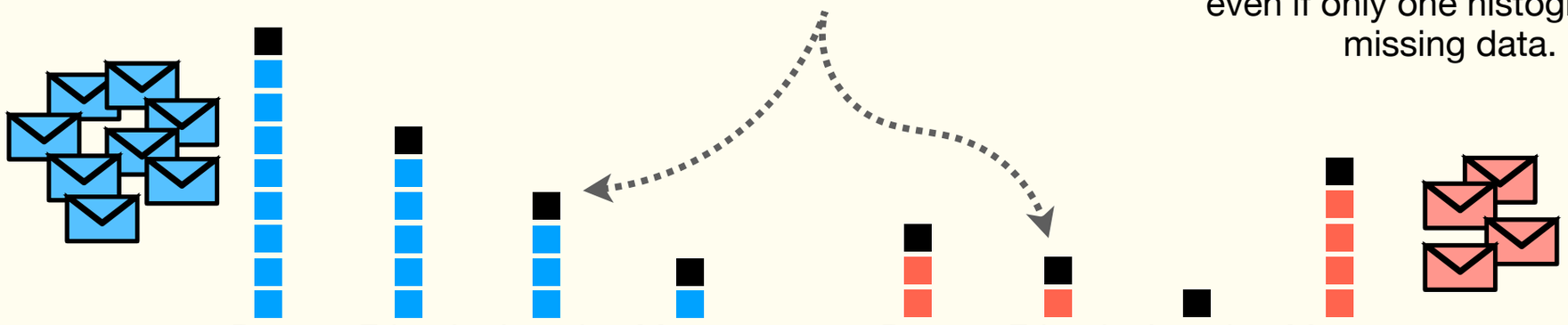
Multinomial Naive Bayes: Dealing With Missing Data

1 Missing data can pose a real problem for **Naive Bayes** or anything else based on histograms. As we saw in **Chapter 3**, we can easily have missing data if the **Training Dataset** is not large enough.

So, **Naive Bayes** eliminates the problem of missing data by adding something called a ***pseudocount*** to each word.

2 A ***pseudocount*** is just an extra value added to each word, and usually adding ***pseudocounts*** means adding **1** count to each word. Here the ***pseudocounts*** are represented by **black boxes**.

NOTE: *pseudocounts* are added to every word in *both* histograms, even if only one histogram has missing data.



3 After we add the ***pseudocounts*** to the histograms, we calculate the probabilities just like before, only this time we include the ***pseudocounts*** in the calculations.

$$p(\textbf{Dear} \mid \textbf{N}) = \frac{8+1}{17+4} = 0.43$$

$p(\textbf{Dear} \mid \textbf{N}) = 0.43$
$p(\textbf{Friend} \mid \textbf{N}) = 0.29$
$p(\textbf{Lunch} \mid \textbf{N}) = 0.19$
$p(\textbf{Money} \mid \textbf{N}) = 0.10$

$p(\textbf{Dear} \mid \textbf{S}) = 0.27$
$p(\textbf{Friend} \mid \textbf{S}) = 0.18$
$p(\textbf{Lunch} \mid \textbf{S}) = 0.09$
$p(\textbf{Money} \mid \textbf{S}) = 0.45$

4 Now the scores for this message are… **Money Money Money Money Lunch**

$$p(\textbf{N}) \times p(\textbf{Money} \mid \textbf{N})^4 \times p(\textbf{Lunch} \mid \textbf{N}) = 0.67 \times 0.10^4 \times 0.19 = 0.00001$$

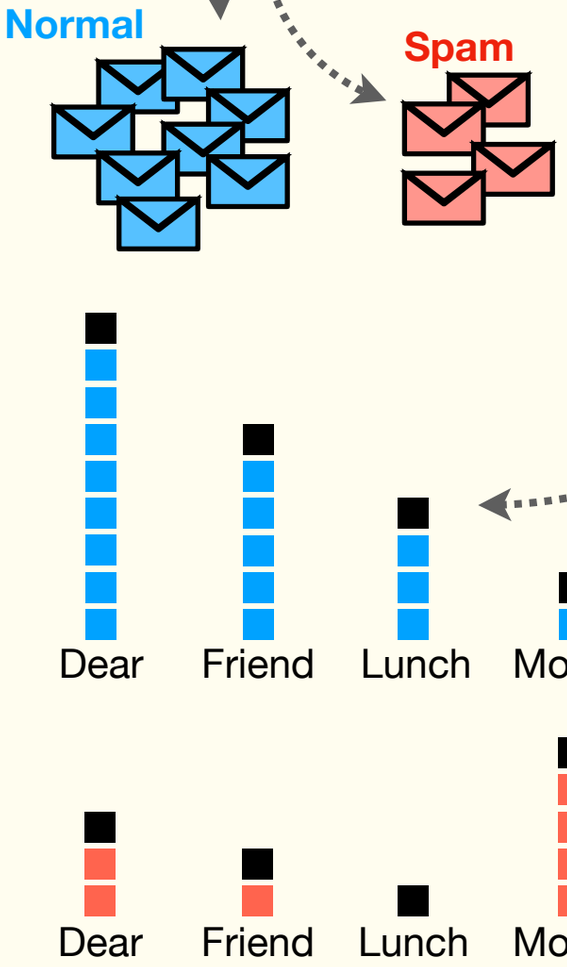
$$p(\textbf{S}) \times p(\textbf{Money} \mid \textbf{S})^4 \times p(\textbf{Lunch} \mid \textbf{S}) = 0.33 \times 0.45^4 \times 0.09 = 0.00122$$

5 Because **Money Money Money Money Lunch** has a higher score for **Spam** (**0.00122**) than **Normal** (**0.00001**), we classify it as **Spam**.

SPAM!!!

Multinomial Naive Bayes vs. Gaussian Naive Bayes

1 So far, we've shown how **Naive Bayes** works with ***discrete*** data, like individual words in a message…



…by building histograms…

…and using the histograms to calculate probabilities that are used to make classifications.

$p($ **Dear** | **N** $) = 0.43$
$p($ **Friend** | **N** $) = 0.29$
$p($ **Lunch** | **N** $) = 0.19$
$p($ **Money** | **N** $) = 0.10$

$p($ **Dear** | **S** $) = 0.27$
$p($ **Friend** | **S** $) = 0.18$
$p($ **Lunch** | **S** $) = 0.09$
$p($ **Money** | **S** $) = 0.45$

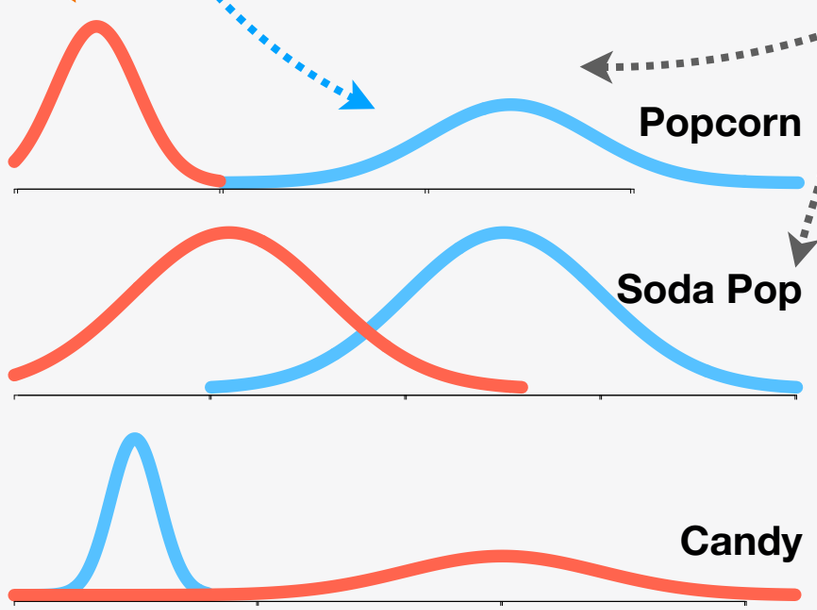
2 In contrast, when we have ***continuous*** data, like these measurements for Popcorn, Soda Pop, and Candy consumption taken from **4** people who **Love Troll 2**…

Popcorn (grams)	Soda Pop (ml)	Candy (grams)
24.3	750.7	0.2
28.2	533.2	50.5
etc.	etc.	etc.

…and from **3** people who **Do Not Love Troll 2**…

Popcorn (grams)	Soda Pop (ml)	Candy (grams)
2.1	120.5	90.7
4.8	110.9	102.3
etc.	etc.	etc.

…we can calculate the **means** and **standard deviations** of each column to draw **Gaussian** (**Normal**) curves…



…and use these curves to make classifications. Read on to learn how!

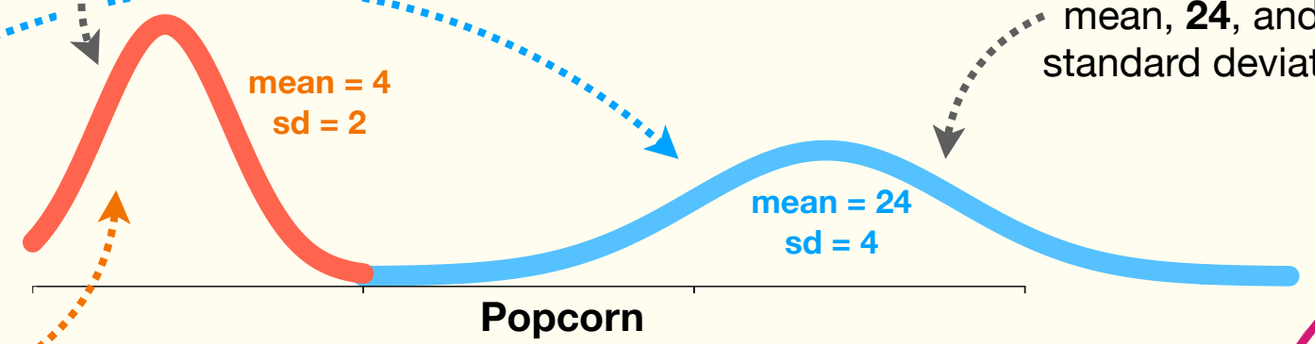
Gaussian Naive Bayes: Details Part 1

1 First we create **Gaussian** (**Normal**) curves for each **Feature** (each column in the **Training Data** that we're using to make predictions).

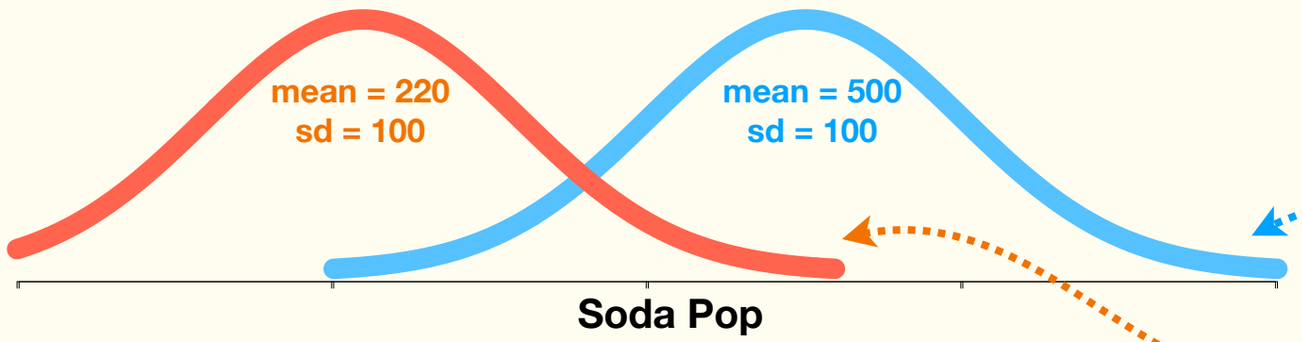
Popcorn (grams)	Soda Pop (ml)	Candy (grams)
24.3	750.7	0.2
28.2	533.2	50.5
etc.	etc.	etc.
2.1	120.5	90.7
4.8	110.9	102.3
etc.	etc.	etc.

Starting with Popcorn, for the people who **Do Not Love Troll 2**, we calculate their mean, **4**, and standard deviation (**sd**), **2**, and then use those values to draw a **Gaussian** curve.

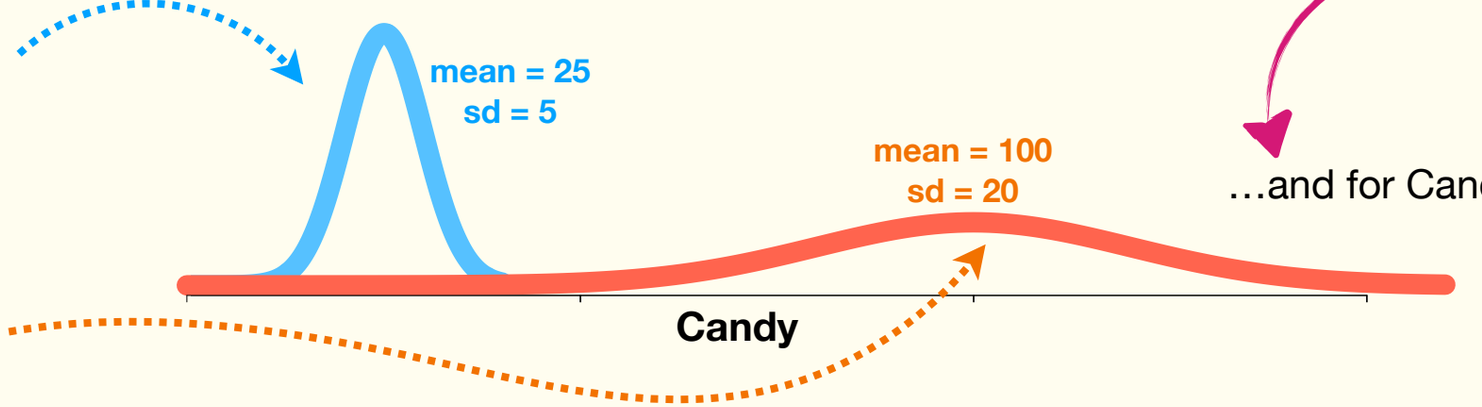
Then we draw a Gaussian curve for the people who **Love Troll 2** using their mean, **24**, and their standard deviation, **4**.



Likewise, we draw curves for Soda Pop…



…and for Candy.

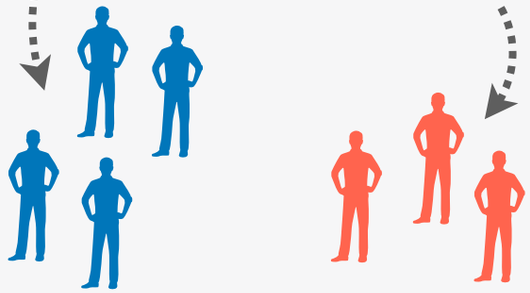


Gaussian Naive Bayes: Details Part 2

2 Now we calculate the **Prior Probability** that someone **Loves Troll 2**.

Just like for **Multinomial Naive Bayes**, this **Prior Probability** is just a guess and can be any probability we want. However, we usually estimate it from the number of people in the **Training Data**.

In this case, the **Training Data** came from **4** people who **Love Troll 2** and **3** people who **Do Not Love Troll 2**.

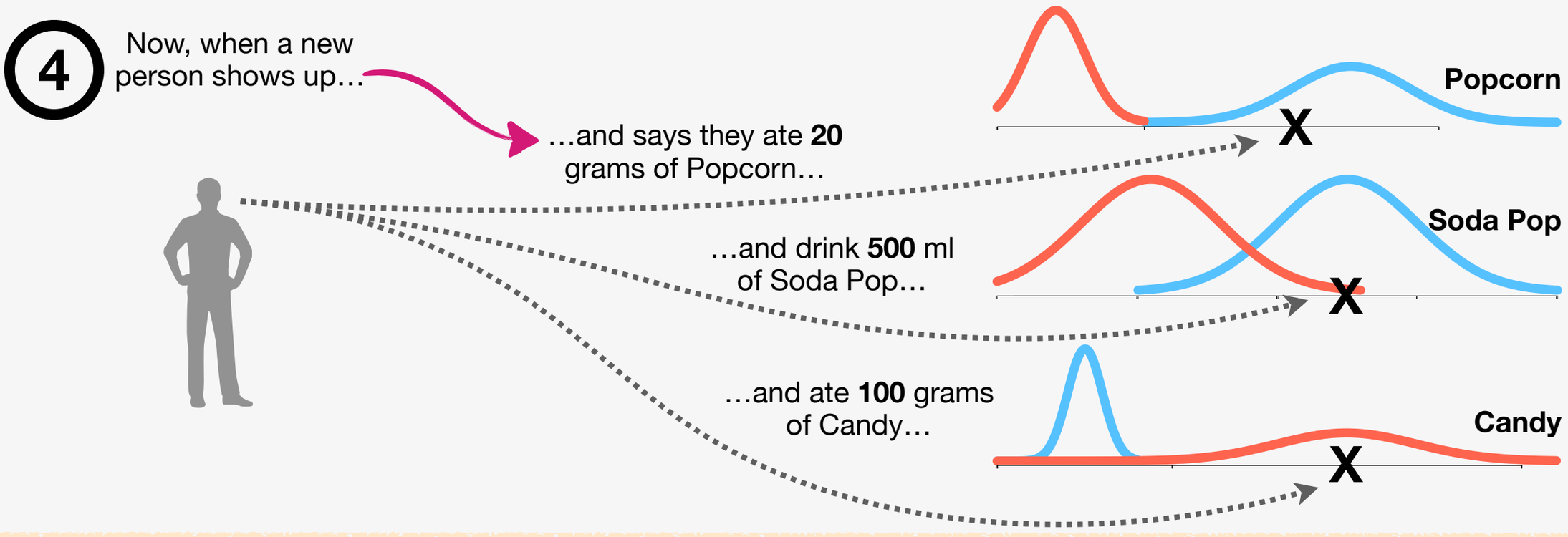


$$p(\text{Loves Troll 2}) = \frac{\text{\# of People Who Love Troll 2}}{\text{Total \# of People}} = \frac{4}{4+3} = 0.6$$

3 Then we calculate the **Prior Probability** that someone **Does Not Love Troll 2**…

$$p(\text{Does Not Love Troll 2}) = \frac{\text{\# of People Who Do Not Love Troll 2}}{\text{Total \# of People}} = \frac{3}{4+3} = 0.4$$

4 Now, when a new person shows up…



Gaussian Naive Bayes: Details Part 3

5 …we calculate the score for **Loves Troll 2** by multiplying the **Prior Probability** that they **Love Troll 2**…

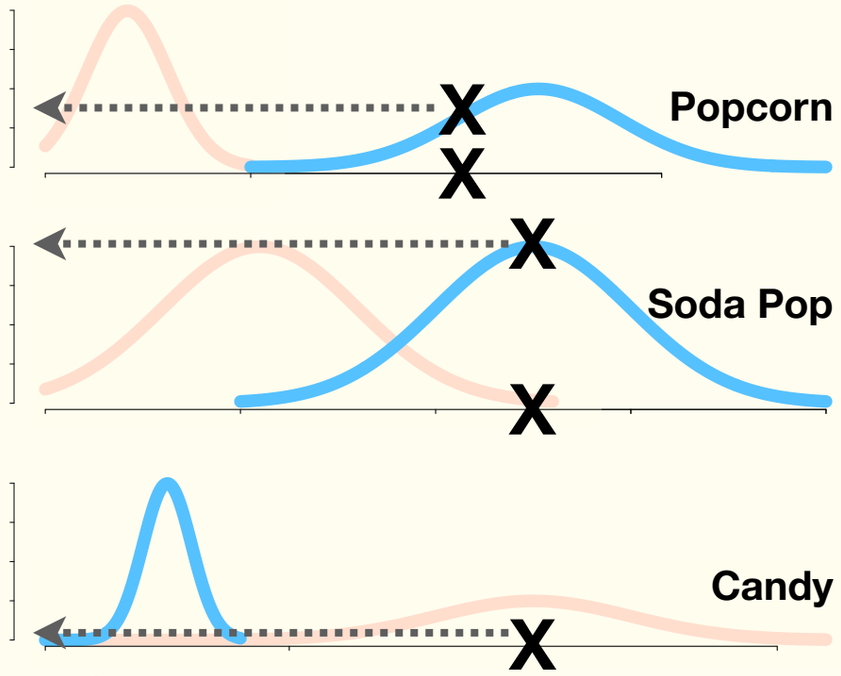
p(**Loves Troll 2**)

x *L*(**Popcorn** = 20 | **Loves**)

x *L*(**Soda Pop** = 500 | **Loves**)

x *L*(**Candy** = 100 | **Loves**)

…by the **Likelihoods**, the y-axis coordinates, that correspond to **20** grams of Popcorn, **500** ml of Soda Pop, and **100** grams of Candy, given that they **Love Troll 2**.



6 **NOTE:** When we plug in the actual numbers…

p(**Loves Troll 2**)	0.6
x *L*(**Popcorn** = 20 | **Loves**)	x 0.06
x *L*(**Soda Pop** = 500 | **Loves**)	x 0.004
x *L*(**Candy** = 100 | **Loves**)	x 0.000000000…001

…we end up plugging in a tiny number for the **Likelihood** of **Candy**, because the **y-axis coordinate** is super close to **0**…

…and computers can have trouble doing multiplication with numbers very close to **0**. This problem is called **Underflow**.

7 To avoid problems associated with numbers close to **0**, we take the **log** (usually the **natural log**, or **log base *e***), which turns the multiplication into addition…

…and turns numbers close to **0** into numbers far from **0**…

…and when we do the math, the score for **Loves Troll 2** = **-124**.

log(0.6 x 0.06 x 0.004 x a tiny number) = **log**(0.6) + **log**(0.06) + **log**(0.004) + **log**(a tiny number)

= -0.51 + -2.8 + -5.52 + -115 = -124

Gaussian Naive Bayes: Details Part 4

8 Likewise, we calculate the score for **Does Not Love Troll 2** by multiplying the **Prior Probability**…

p(**No Love**)

x L(**Popcorn** = 20 | **No Love**)

x L(**Soda Pop** = 500 | **No Love**)

x L(**Candy** = 100 | **No Love**)

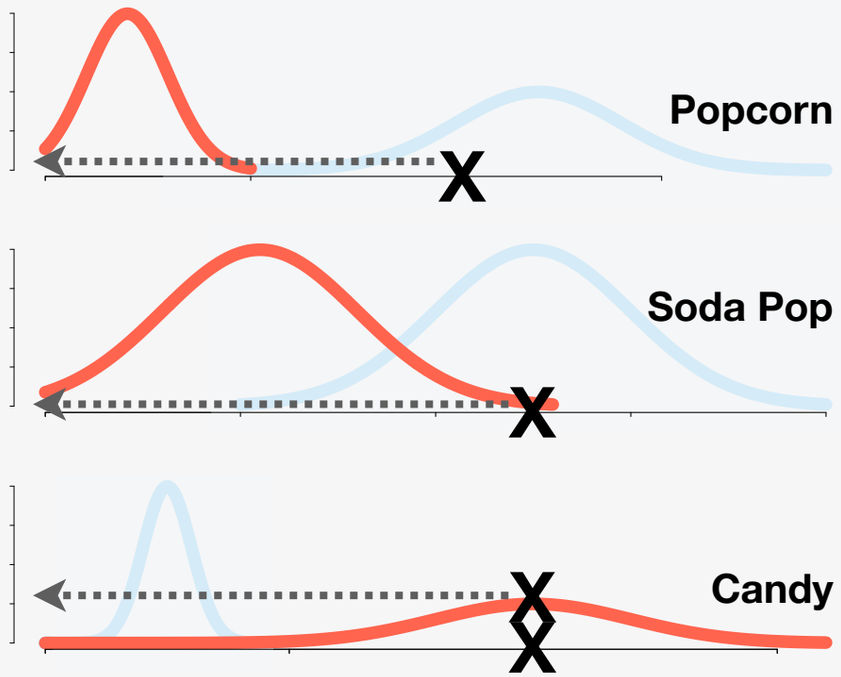
…by the **Likelihoods** from the **Does Not Love Troll 2** distributions for Popcorn, Soda Pop, and Candy. But before we do the math, we first take the **log**…

log(p(**No Love**))

+ **log**(L(**Popcorn** = 20 | **No Love**))

+ **log**(L(**Soda Pop** = 500 | **No Love**))

+ **log**(L(**Candy** = 100 | **No Love**))



…then we plug in the numbers, do the math, and get **-48**.

log(0.4) + **log**(a tiny number) + **log**(0.00008) + **log**(0.02)

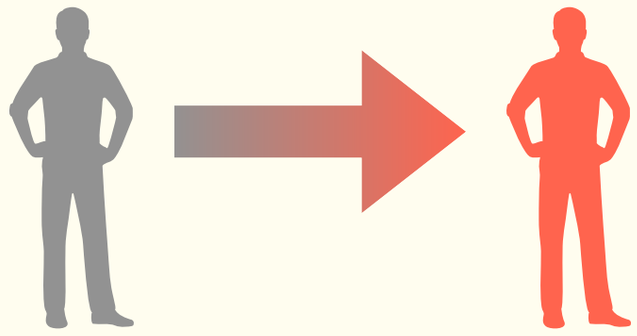
= -0.92 + -33.61 + -9.44 + -3.91 = -48

9 Lastly, because the score for **Does Not Love Troll 2** (**-48**) is greater than the score for **Loves Troll 2** (**-124**), we classify this person…

Log(**Loves Troll 2 Score**) = -124

Log(**Does Not Love Troll 2 Score**) = -48

…as someone who **Does Not Love Troll 2**.



DOUBLE BAM!!!

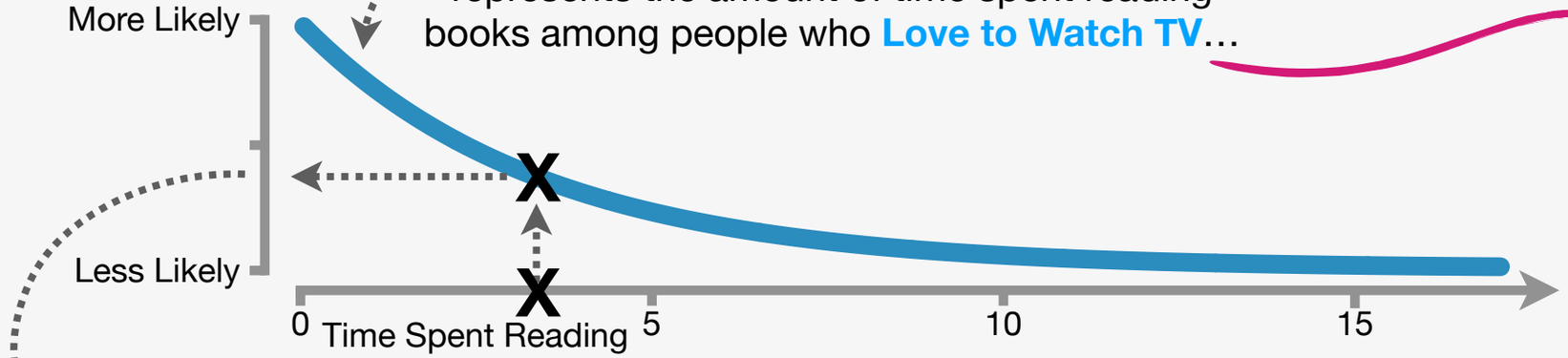
Now that we understand **Multinomial** and **Gaussian Naive Bayes**, let's answer some **Frequently Asked Questions**.

Naive Bayes: FAQ Part 1

If my continuous data are not *Gaussian*, can I still use *Gaussian Naive Bayes*?

Although the **Gaussian** (**Normal**) distribution is the most commonly used, we can use *any* statistical distribution in combination with **Naive Bayes**.

For example, if this **Exponential** distribution represents the amount of time spent reading books among people who **Love to Watch TV**…



…then we can plug the **Likelihood** — the y-axis coordinate that corresponds to the specific amount of time spent reading — into the equation for the score for **Loves To Watch TV**.

p(**Loves TV Prior Probability**)

x *L*(Reading = 3 | **Loves TV**)

However, there's one problem with using an **Exponential** distribution…

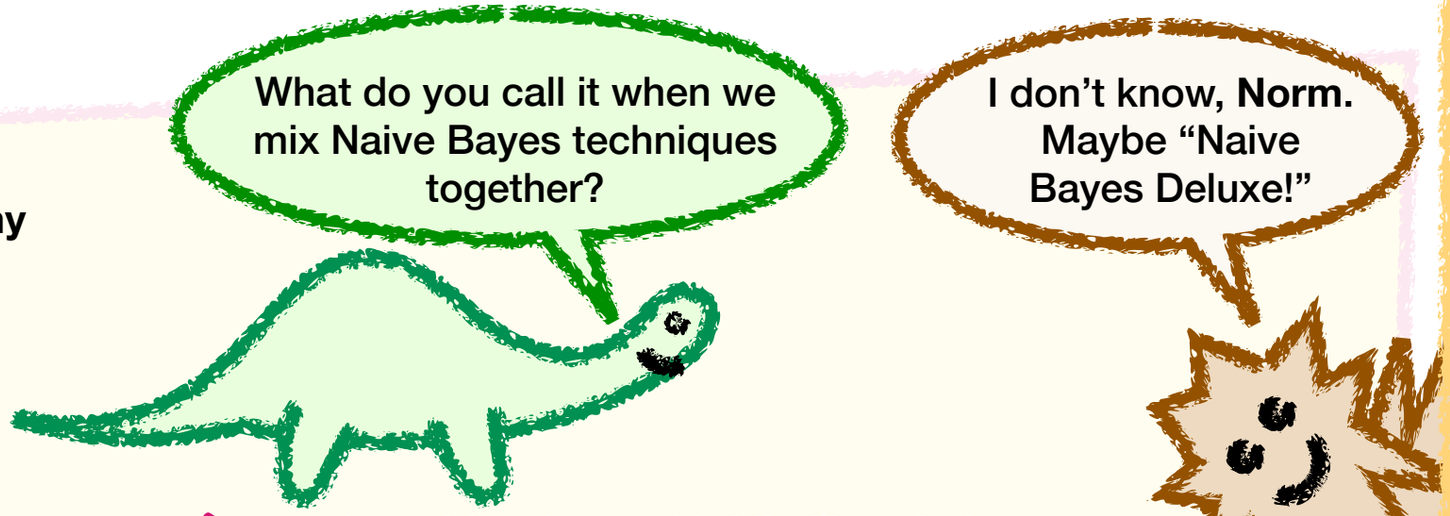
…we can't call it **Gaussian Naive Bayes** anymore!!!
Now we'll call it **Exponential Naive Bayes**.

Silly Bam. :)

Naive Bayes: FAQ Part 2

What if some of my data are *discrete* and some of my data are *continuous*? Can I still use Naive Bayes?

Yes! We can combine the histogram approach from **Multinomial Naive Bayes** with the distribution approach from **Gaussian Naive Bayes**.

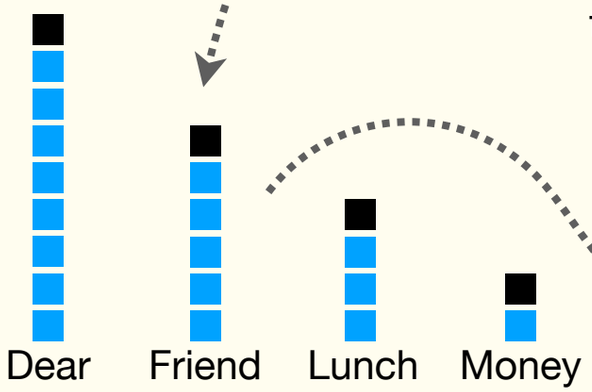


For example, if we had word counts from **Normal** messages and **Spam**, we could use them to create histograms and probabilities…

…and combine those with **Likelihoods** from **Exponential** distributions that represent the amount of time that elapses between receiving **Normal** messages and **Spam**…

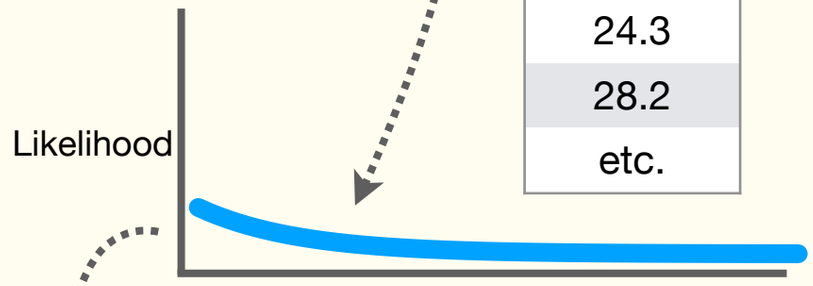
…to calculate **Normal** and **Spam** scores for the message **Dear Friend**, received after **25** seconds elapsed, using both **discrete** and **continuous** data.

Normal

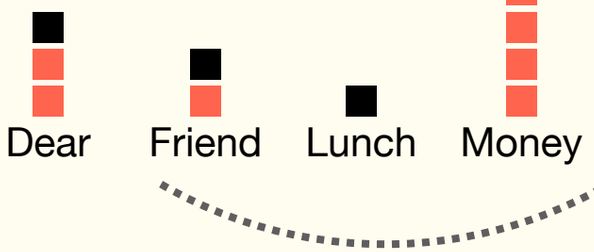


log(*p*(**Normal**))
+ **log**(*p*(**Dear** | **Normal**))
+ **log**(*p*(**Friend** | **Normal**))
+ **log**(*L*(Time = 25 | **Normal**))

Time
24.3
28.2
etc.

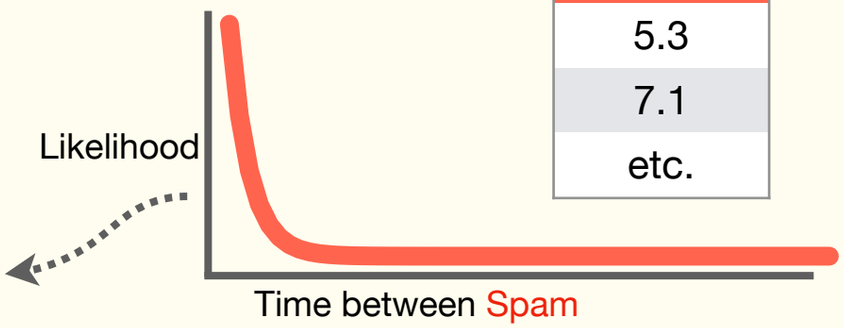


Spam



log(*p*(**Spam**))
+ **log**(*p*(**Dear** | **Spam**))
+ **log**(*p*(**Friend** | **Spam**))
+ **log**(*L*(**Time = 25** | **Spam**))

Time
5.3
7.1
etc.



TRIPLE SPAM!!!

Naive Bayes: FAQ Part 3

How is Naive Bayes related to Bayes' Theorem?

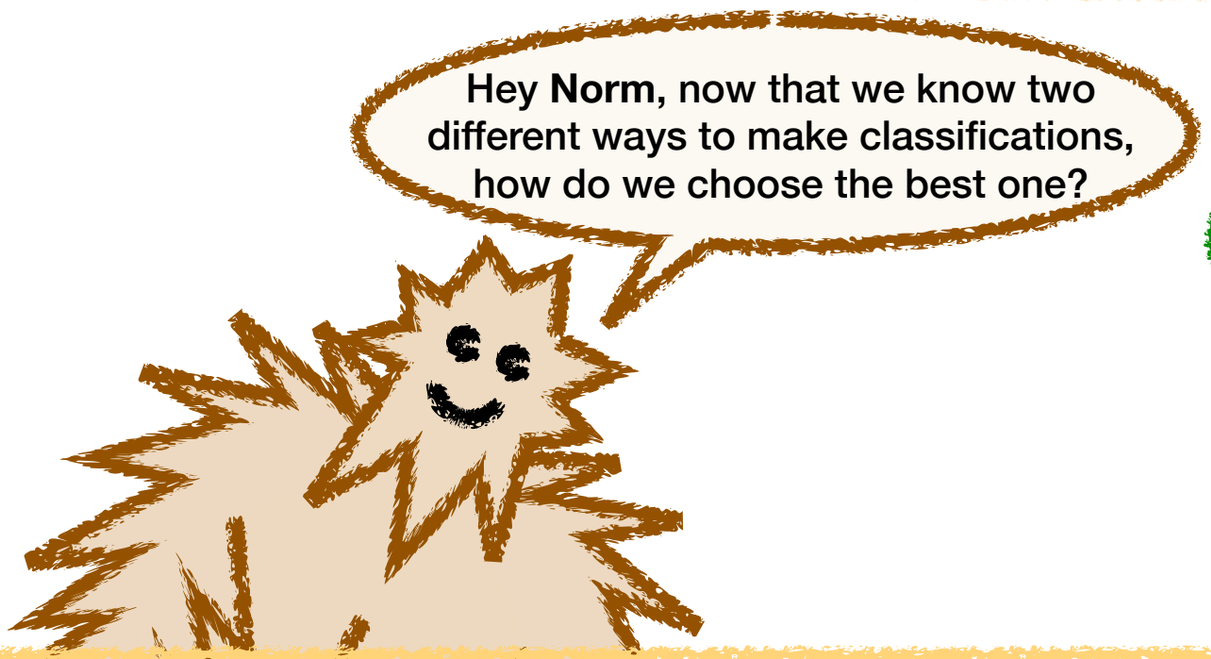
If we wanted to do extra math that wouldn't change the result, we could divide each score by the sum of the scores, and that would give us **Bayes' Theorem**.

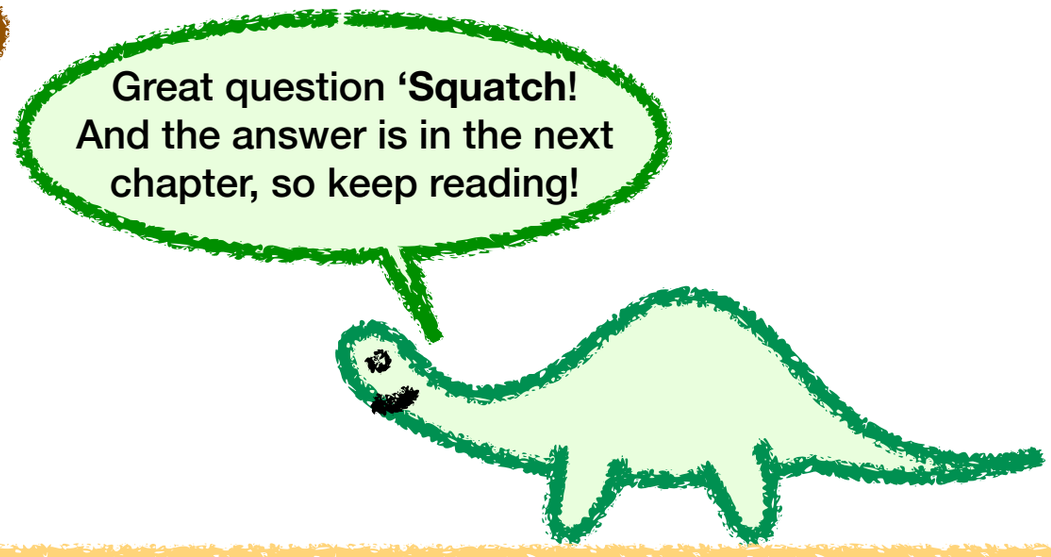
Because the denominators are the same in both equations, the results are determined entirely by the numerators. So, to save time and trouble, often the denominators are omitted.

small bam.

$$p(\text{N} \mid \textbf{Dear Friend}) = \frac{p(\text{N}) \times p(\textbf{Dear} \mid \text{N}) \times p(\textbf{Friend} \mid \text{N})}{p(\text{N}) \times p(\textbf{Dear} \mid \text{N}) \times p(\textbf{Friend} \mid \text{N}) + p(\text{S}) \times p(\textbf{Dear} \mid \text{S}) \times p(\textbf{Friend} \mid \text{S})}$$

$$p(\text{S} \mid \textbf{Dear Friend}) = \frac{p(\text{S}) \times p(\textbf{Dear} \mid \text{S}) \times p(\textbf{Friend} \mid \text{S})}{p(\text{N}) \times p(\textbf{Dear} \mid \text{N}) \times p(\textbf{Friend} \mid \text{N}) + p(\text{S}) \times p(\textbf{Dear} \mid \text{S}) \times p(\textbf{Friend} \mid \text{S})}$$





Chapter 08

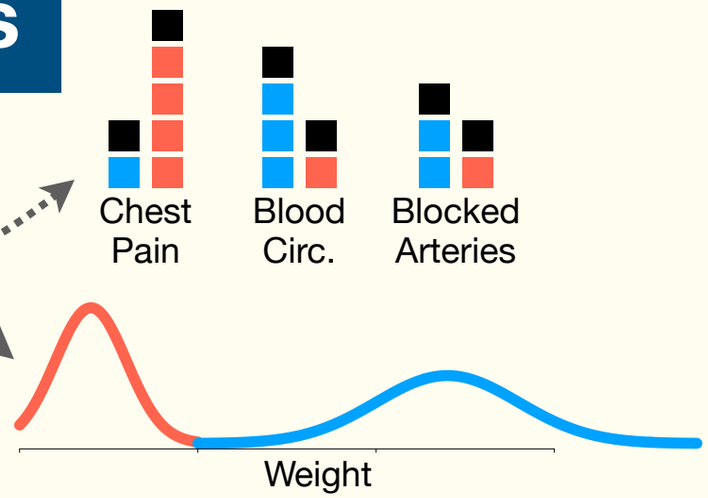
Assessing Model Performance!!!

Assessing Model Performance: Main Ideas

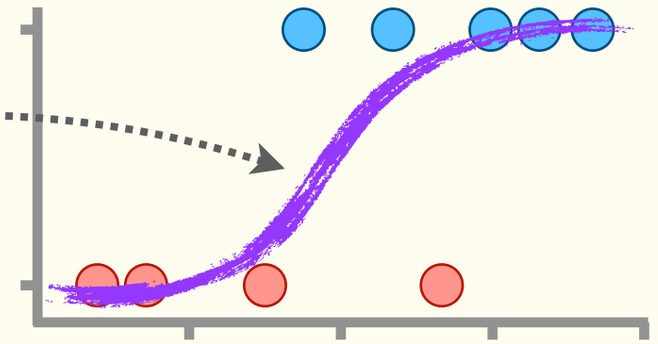
1 **The Problem:** We've got this dataset, and we want to use it to predict if someone has Heart Disease, but we don't know in advance which model will make the best predictions.

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
...	...	...	...	...

Should we choose **Naive Bayes**...



...or **Logistic Regression**?

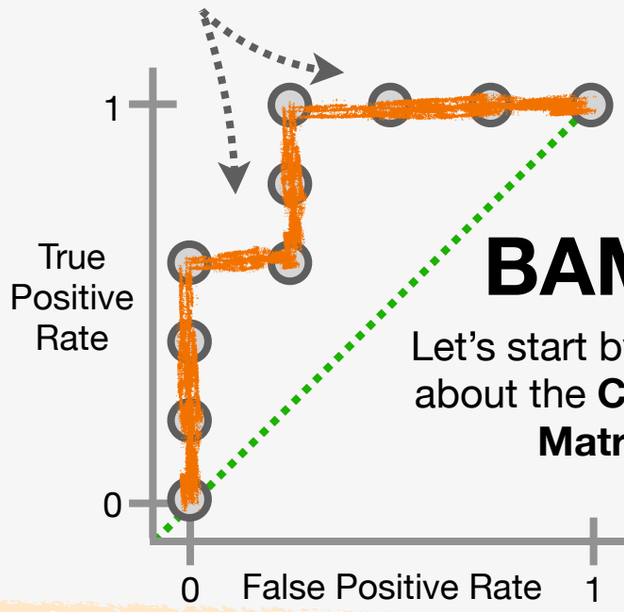


2 **A Solution:** Rather than worry too much upfront about which model will be best, we can try both and assess their performance using a wide variety of tools...

		Predicted: Has Heart Disease	Predicted: Does Not Have Heart Disease
Actual	Has Heart Disease	142	22
Actual	Does Not Have Heart Disease	29	110

...from **Confusion Matrices**, simple grids that tell us what each model did right and what each one did wrong...

...to **Receiver Operator Curves** (**ROCs**), which give us an easy way to evaluate how each model performs with different classification thresholds.



BAM!!!

Let's start by learning about the **Confusion Matrix**.

The Confusion Matrix: Main Ideas

1 So we have this dataset that we can use to predict if someone has Heart Disease, and we want to decide between using **Naive Bayes** and **Logistic Regression**.

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
...	...	...	...	...

2 We start by dividing the data into two blocks…

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
...	...	...	...	...

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	Yes	No	210	No
...	...	...	...	...

…and we use the first block to optimize both models using **Cross Validation**.

3 Then we apply the optimized **Naive Bayes** model to the second block and build a **Confusion Matrix** that helps us keep track of four things: **1)** the number of people with Heart Disease who were *correctly* predicted to have Heart Disease…

…**2)** the number of people with Heart Disease who were *incorrectly* predicted to not have Heart Disease…

…**3)** the number of people without Heart Disease who were *correctly* predicted to not have Heart Disease…

…and, lastly, **4)** the number of people without Heart Disease who were *incorrectly* predicted to have Heart Disease.

		Predicted: Has Heart Disease	Predicted: Does Not Have Heart Disease
Actual	Has Heart Disease	142	22
Actual	Does Not Have Heart Disease	29	110

4 We then build a **Confusion Matrix** for the optimized **Logistic Regression** model…

		Predicted: Yes	Predicted: No
Actual	Yes	137	22
Actual	No	29	115

…and, at a glance, we can see that **Logistic Regression** is better at predicting people ***without*** Heart Disease and **Naive Bayes** is better at predicting people ***with*** it.

Now we can pick the model based on whether we want to identify someone ***with*** or ***without*** Heart Disease.

BAM!!

The Confusion Matrix: Details Part 1

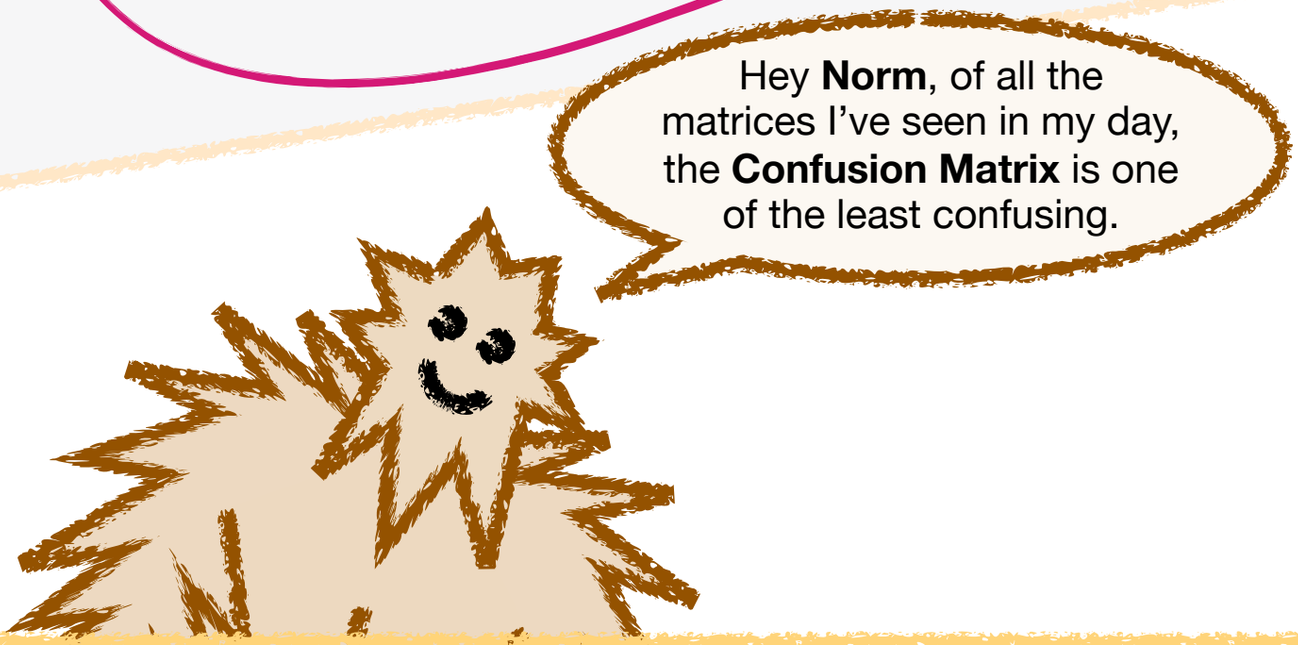
1 When the actual and predicted values are both **YES**, then we call that a **True Positive**…

…when the actual value is **YES**, but the predicted value is **NO**, then we call that a **False Negative**…

		Predicted: Yes	Predicted: No
Actual	Yes	**True Positives**	False Negatives
Actual	No	False Positives	**True Negatives**

…and when the actual value is **NO**, but the predicted value is **YES**, then we call that a **False Positive**.

…when the actual and predicted values are both **NO**, then we call that a **True Negative**…



The Confusion Matrix: Details Part 2

2 When there are only two possible outcomes, like **Yes** and **No**…

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
...	...	...	...	...

…then the corresponding **Confusion Matrix** has **2** rows and **2** columns: one each for **Yes** and **No**.

		Predicted	
		Has Heart Disease	Does Not Have Heart Disease
Actual	Has Heart Disease	**142**	22
	Does Not Have Heart Disease	29	**110**

3 When there are **3** possible outcomes, like in this dataset that has **3** choices for favorite movie, **Troll 2**, **Gore Police**, and **Cool as Ice**…

Jurassic Park III	Run for Your Wife	Out Kold	Howard the Duck	Favorite Movie
Yes	No	Yes	Yes	Troll 2
No	No	Yes	No	Gore Police
No	Yes	Yes	Yes	Cool as Ice
...	...	...	...	...

…then the corresponding **Confusion Matrix** has **3** rows and **3** columns.

		Predicted		
		Troll 2	Gore Police	Cool As Ice
Actual	Troll 2	**142**	22	..
	Gore Police	29	**110**	..
	Cool as Ice	..	..	**...**

4 In general, the size of the matrix corresponds to the number of classifications we want to predict.

Bam.

The Confusion Matrix: Details Part 3

5

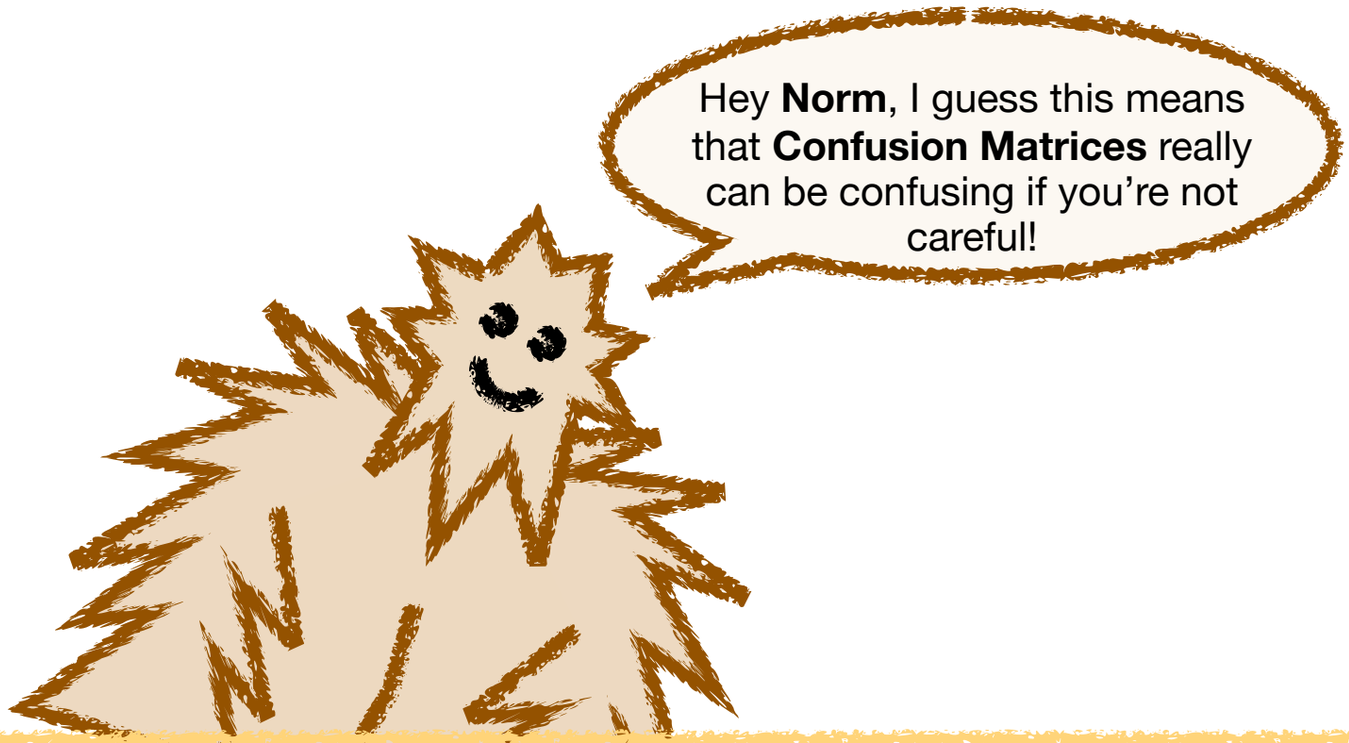
Unfortunately, there's no standard for how a **Confusion Matrix** is oriented. In many cases, the rows reflect the actual, or known, classifications and the columns represent the predictions.

		Predicted	
		Has Heart Disease	Does Not Have Heart Disease
Actual	Has Heart Disease	**True Positives**	False Negatives
	Does Not Have Heart Disease	False Positives	**True Negatives**

In other cases, the rows reflect the predictions and the columns represent the actual, or known, classifications.

		Actual	
		Has Heart Disease	Does Not Have Heart Disease
Predicted	Has Heart Disease	**True Positives**	False Positives
	Does Not Have Heart Disease	False Negatives	**True Negatives**

So, make sure you read the labels before you interpret a **Confusion Matrix**!



The Confusion Matrix: Details Part 4

Gentle Reminder:

		Predicted: Yes	Predicted: No
Actual: Yes		**TP** (True Positive)	FN (False Negative)
Actual: No		FP (False Positive)	**TN** (True Negative)

6 Lastly, in the original example, when we compared the **Confusion Matrices** for **Naive Bayes** and **Logistic Regression**…

…because both matrices contained the same number of **False Negatives** (**22** each) and **False Positives** (**29** each)…

…all we needed to do was compare the **True Positives** (**142** vs. **137**) to quantify how much better **Naive Bayes** was at predicting people *with* Heart Disease. Likewise, to quantify how much better **Logistic Regression** was at predicting people *without* Heart Disease, all we needed to do was compare the **True Negatives** (**110** vs. **115**).

Naive Bayes

Actual \ Predicted	Yes	No
Yes	**142**	22
No	29	**110**

Logistic Regression

Actual \ Predicted	Yes	No
Yes	**137**	22
No	29	**115**

7 However, what if we had ended up with different numbers of **False Negatives** and **False Positives**?

We can still see that **Naive Bayes** is better at predicting people *with* Heart Disease, but quantifying how much better is a little more complicated because now we have to compare **True Positives** (**142** vs. **139**) and **False Negatives** (**29** vs. **32**).

We still see that **Logistic Regression** is better at predicting people *without* Heart Disease, but quantifying how much better has to take **True Negatives** (**110** vs. **112**) *and* **False Positives** (**22** vs. **20**) into account.

Naive Bayes

Actual \ Predicted	Yes	No
Yes	**142**	29
No	22	**110**

Logistic Regression

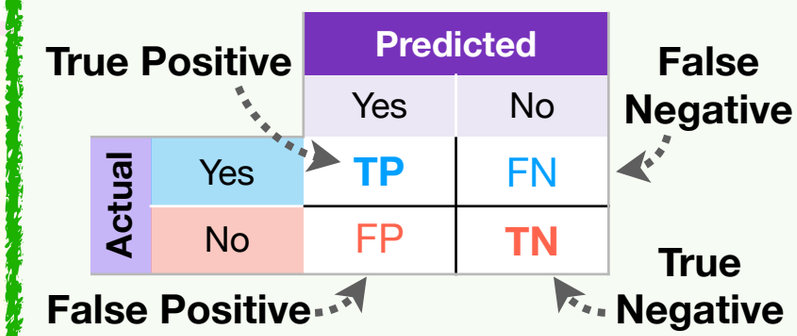
Actual \ Predicted	Yes	No
Yes	**139**	32
No	20	**112**

8 The good news is that we have metrics that include various combinations of **True** and **False Positives** with **True** and **False Negatives** and allow us to easily quantify all kinds of differences in algorithm performance. The first of these metrics are **Sensitivity** and **Specificity**, and we'll talk about those next.

BAM!!!

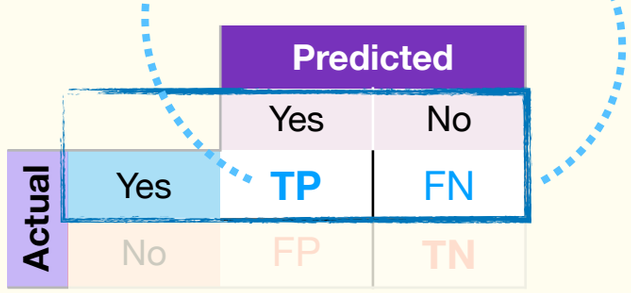
Sensitivity and Specificity: Main Ideas

Gentle Reminder:

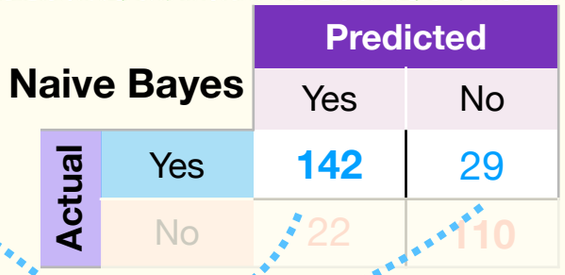


1 When we want to quantify how well an algorithm (like **Naive Bayes**) correctly classifies the *actual* **Positives**, in this case, the known people *with* Heart Disease, we calculate **Sensitivity**, which is the percentage of the actual **Positives** that were *correctly* classified.

$$\textbf{Sensitivity} = \frac{\text{True Positives}}{\text{True Positives + False Negatives}}$$



For example, using the Heart Disease data and **Confusion Matrix**, the **Sensitivity** for **Naive Bayes** is **0.83**…



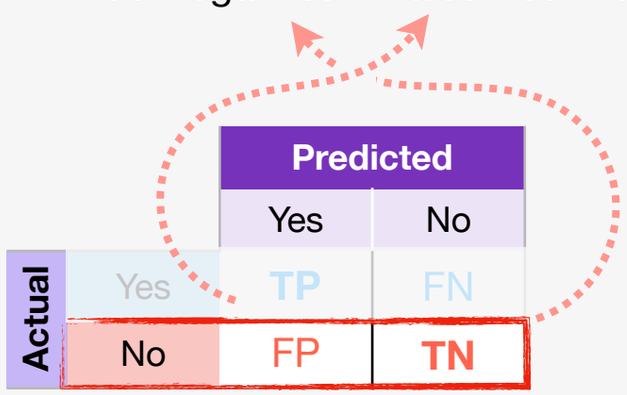
$$\text{Sensitivity} = \frac{TP}{TP + FN} = \frac{142}{142 + 29} = 0.83$$

…which means that **83%** of the people *with* Heart Disease were correctly classified.

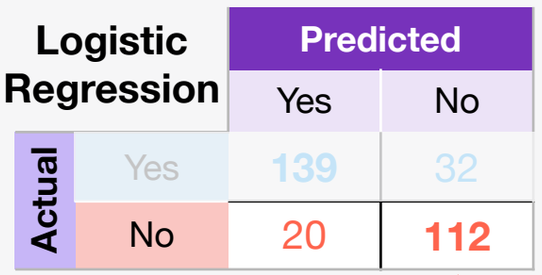
BAM!!!

2 When we want to quantify how well an algorithm (like **Logistic Regression**) correctly classifies the *actual* **Negatives**, in this case, the known people *without* Heart Disease, we calculate **Specificity**, which is the percentage of the actual **Negatives** that were *correctly* classified.

$$\textbf{Specificity} = \frac{\text{True Negatives}}{\text{True Negatives + False Positives}}$$



For example, using the **Heart Disease** data and **Confusion Matrix**, the **Specificity** for **Logistic Regression** is **0.85**…



$$\text{Specificity} = \frac{TN}{TN + FP} = \frac{115}{115 + 20} = 0.85$$

…which means that **85%** of the people *without* Heart Disease were correctly classified.

DOUBLE BAM!!!

Now let's talk about **Precision** and **Recall**.

Precision and Recall: Main Ideas

1 **Precision** is another metric that can summarize a **Confusion Matrix**. It tells us the percentage of the *predicted* **Positive** results (so, both **True** and **False Positives**) that were *correctly* classified.

$$\textbf{Precision} = \frac{\text{True Positives}}{\text{True Positives + False Positives}}$$

		Predicted: Yes	Predicted: No
Actual	Yes	**TP**	FN
Actual	No	FP	TN

For example, using the Heart Disease data and **Confusion Matrix**, the **Precision** for **Naive Bayes** is **0.87**…

Naive Bayes		Predicted: Yes	Predicted: No
Actual	Yes	**142**	29
Actual	No	22	110

$$\text{Sensitivity} = \frac{TP}{TP + FP} = \frac{142}{142 + 22} = 0.87$$

…which means that of the **164** people that we predicted to have Heart Disease, **87%** actually have it. In other words, **Precision** gives us a sense of the *quality* of the positive results. When we have high **Precision**, we have high-quality positive results.

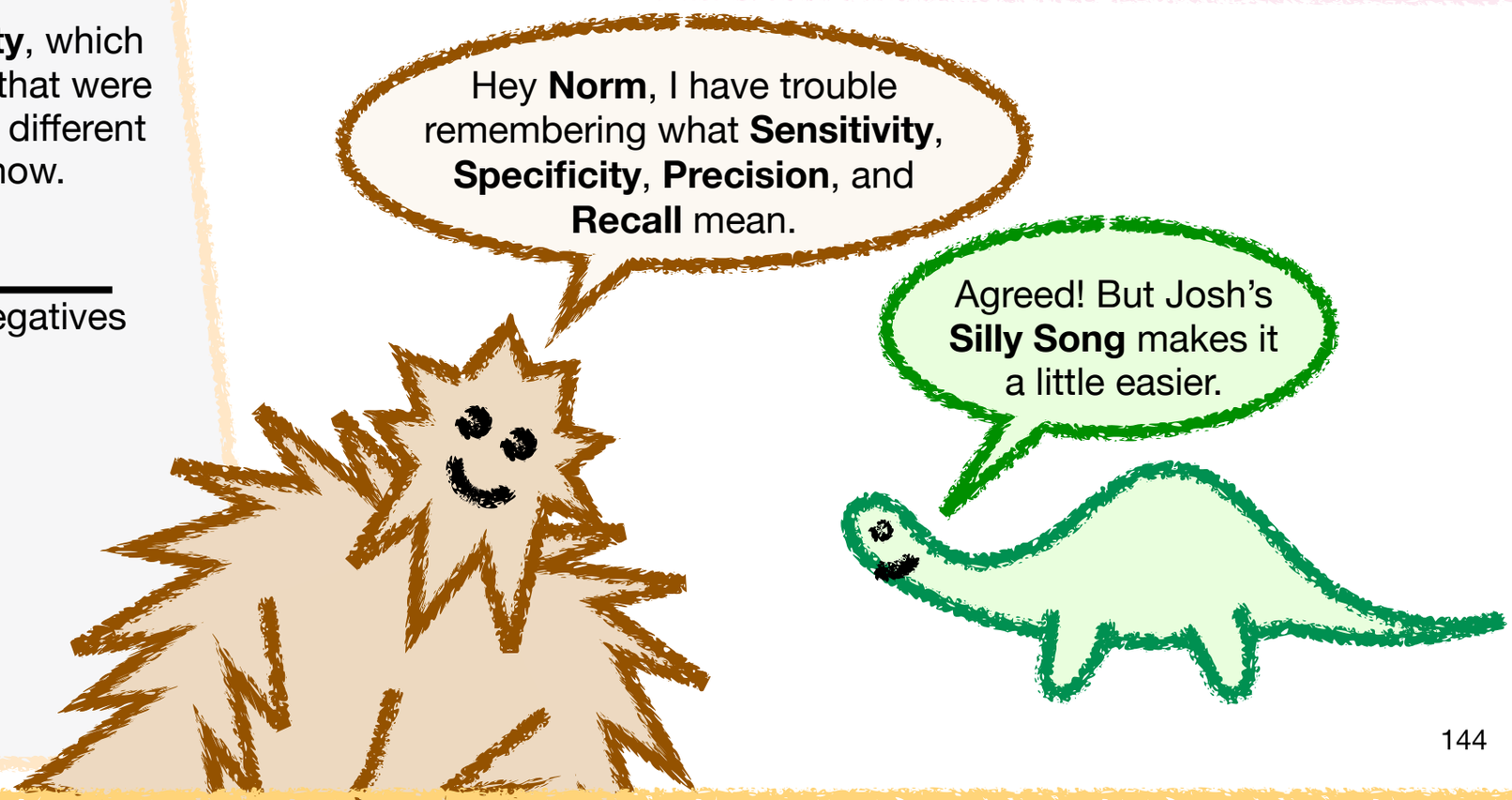
Gentle Reminder:

		Predicted: Yes	Predicted: No
Actual	Yes	**TP** (True Positive)	FN (False Negative)
Actual	No	FP (False Positive)	**TN** (True Negative)

2 **Recall** is just another name for **Sensitivity**, which is the percentage of the *actual* **Positives** that were *correctly* classified. Why do we need two different names for the same thing? I don't know.

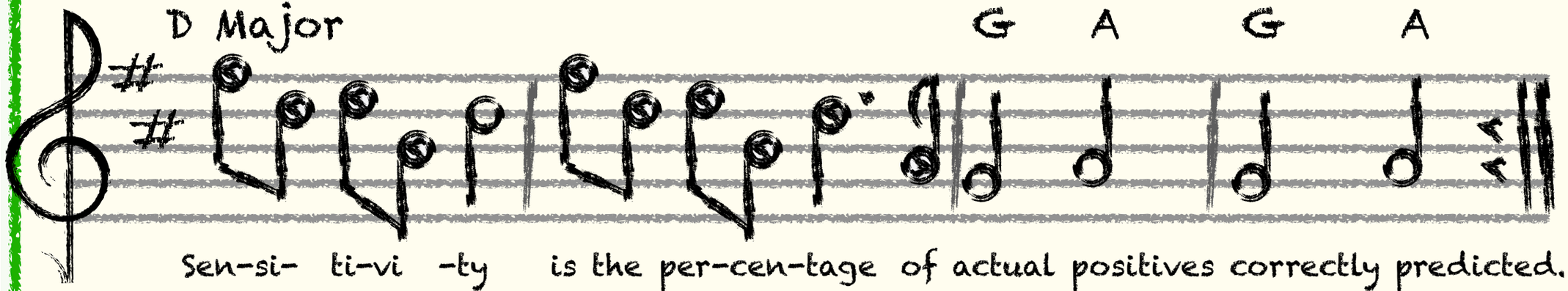
$$\textbf{Recall} = \textbf{Sensitivity} = \frac{\text{True Positives}}{\text{True Positives + False Negatives}}$$

		Predicted: Yes	Predicted: No
Actual	Yes	**TP**	FN
Actual	No	FP	TN



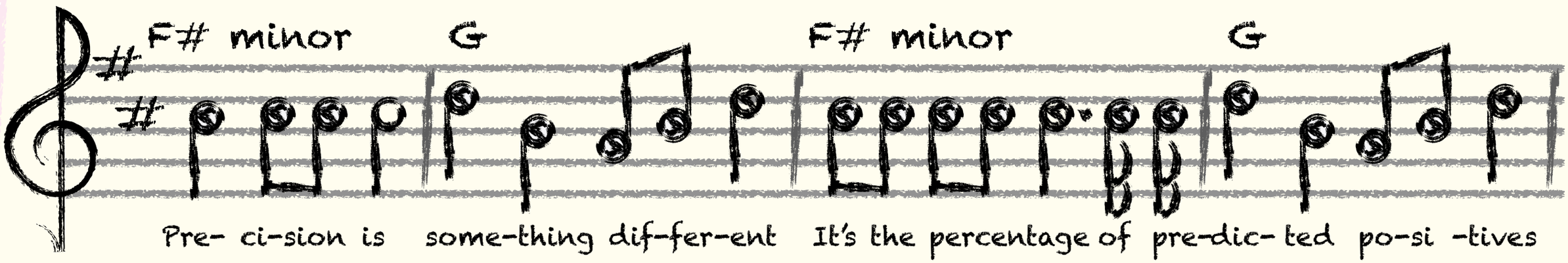
Scan, click, or tap this QR code to hear the **Silly Song**!!!

The Sensitivity, Specificity, Precision, Recall Song!!!

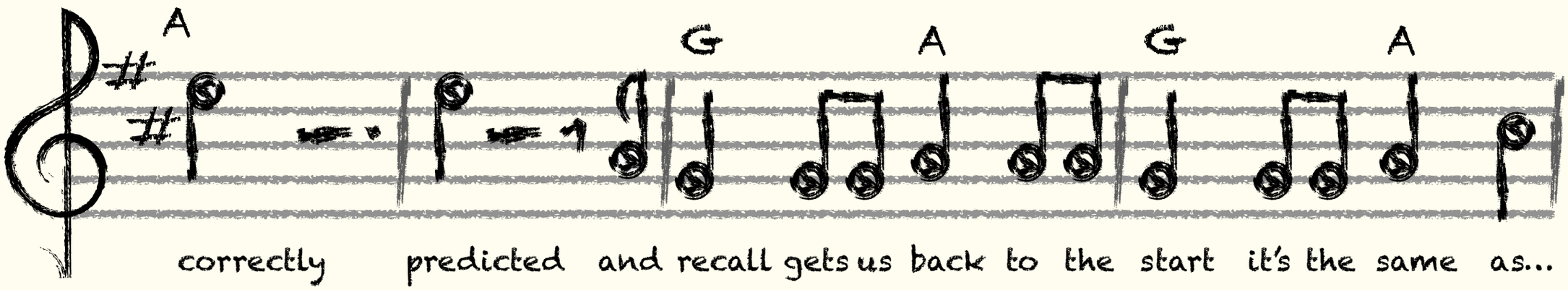


Sen-si- ti-vi -ty is the per-cen-tage of actual positives correctly predicted.

Spe-ci- fi-ci -ty is the per-cen-tage of actual negatives correctly predicted.

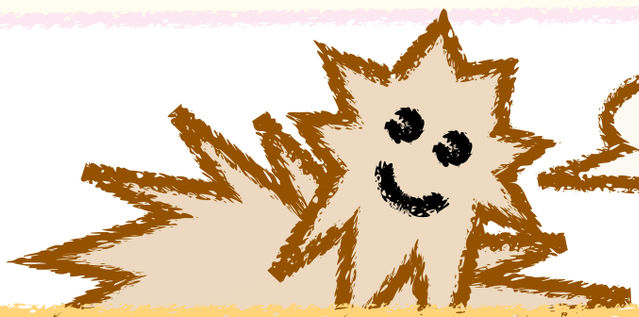


Pre- ci-sion is some-thing dif-fer-ent It's the percentage of pre-dic- ted po-si -tives



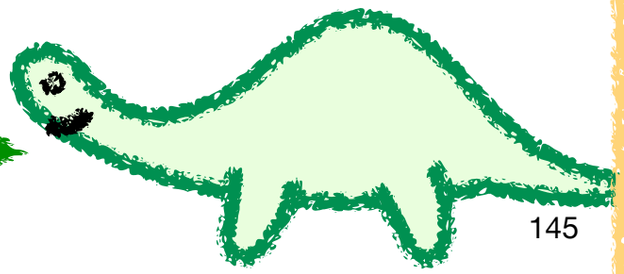
correctly predicted and recall gets us back to the start it's the same as...

Now go back to the top and repeat!!!



Hooray!!!

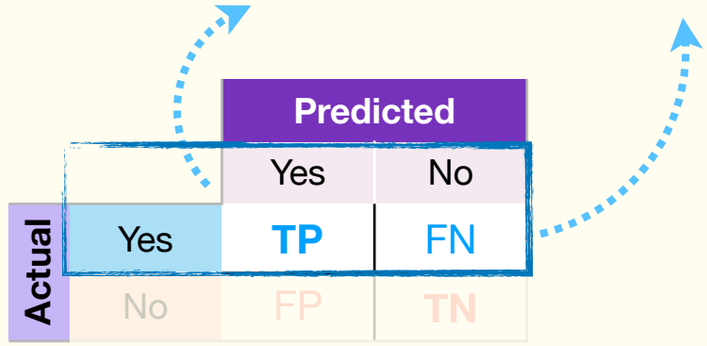
Now let's talk about the **True Positive Rate** and the **False Positive Rate**.

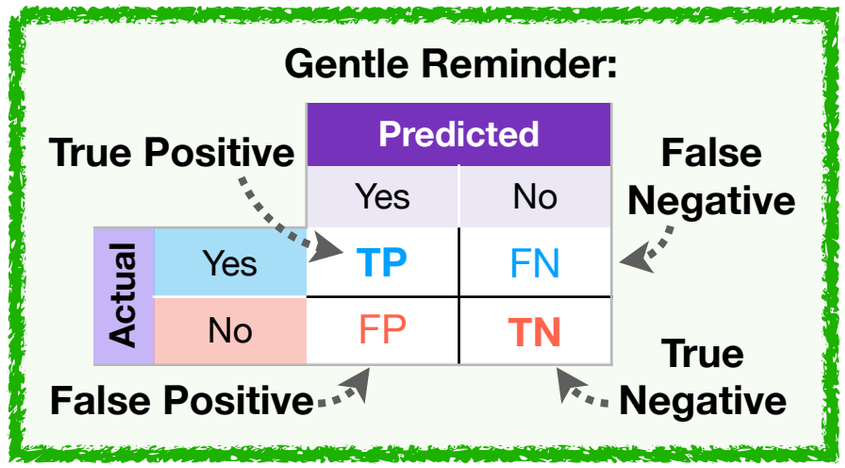


True Positive Rate and False Positive Rate: Main Ideas

1 The **True Positive Rate** is the same thing as **Recall**, which is the same thing as **Sensitivity**. I kid you not. We have **3** names for the exact same thing: the percentage of *actual* **Positives** that were *correctly* classified. **TRIPLE UGH!!!**

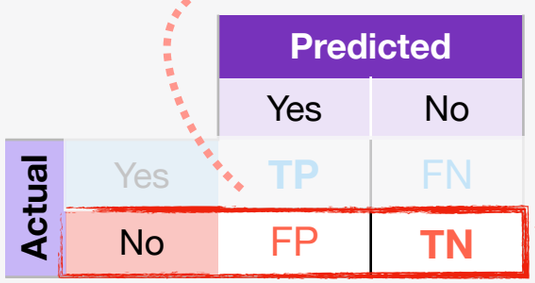
$$\textbf{True Positive Rate} = \textbf{Recall} = \textbf{Sensitivity} = \frac{\text{True Positives}}{\text{True Positives + False Negatives}}$$





2 The **False Positive Rate** tells you the percentage of *actual* **Negatives** that were *incorrectly* classified. In this case, it's the known people *without* Heart Disease who were *incorrectly* classified.

$$\textbf{False Positive Rate} = \frac{\text{False Positives}}{\text{False Positives + True Negatives}}$$



NOTE: Remember, **Specificity** is the proportion of actual **Negatives** that were *correctly* classified, thus…

False Positive Rate = 1 - **Specificity**

…and…

Specificity = 1 - **False Positive Rate**

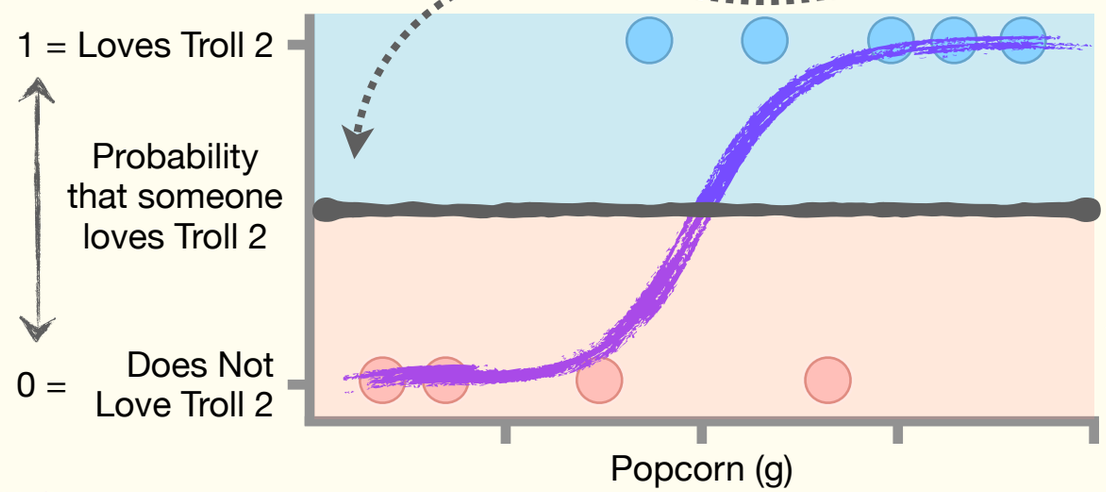
3 Okay, now we've got some fancy terminology that we can use to summarize individual confusion matrices. BAM?

To be honest, even with the **Silly Song**, I find it hard to remember all of this fancy terminology and so don't use it that much on its own. However, it's a stepping stone to something that I use all the time and find super useful: **ROC** and **Precision Recall Graphs**, so let's learn about those!

BAM!!!

ROC: Main Ideas Part 1

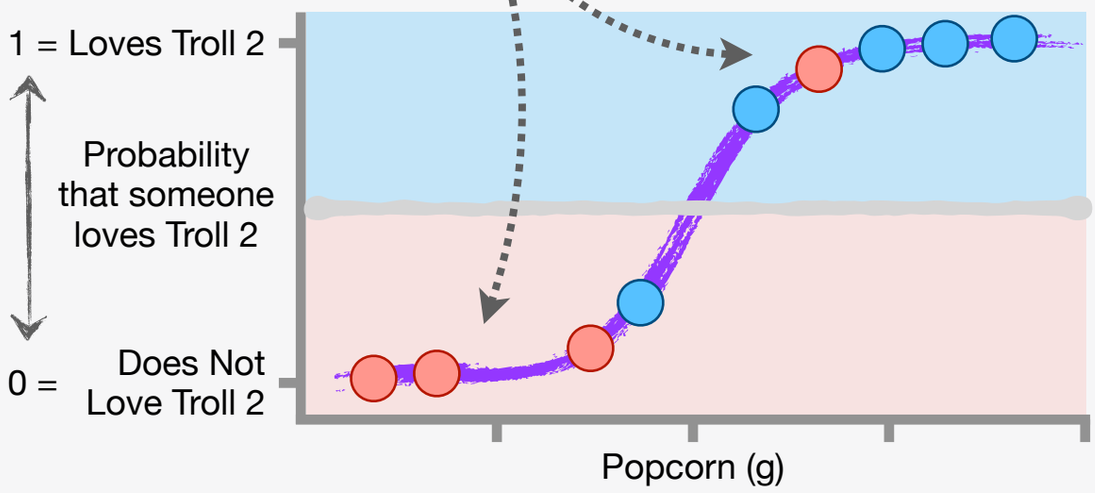
1 In **Chapter 6**, we used **Logistic Regression** to predict whether or not someone loves Troll 2, and we mentioned that usually the classification threshold is **50%**…



…which means that if the predicted probability is **> 50%**, then we classify them as someone who **Loves Troll 2**…

…and if the probability is **≤ 50%**, then we classify them as someone who **Does Not Love Troll 2**.

2 Now, using a classification threshold of **50%**, we can classify the **Training Data**…

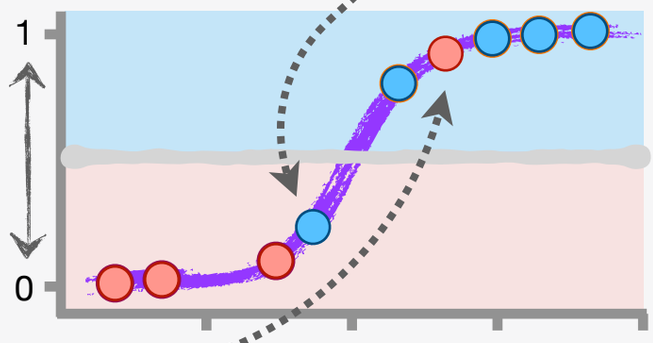


…and create this **Confusion Matrix**.

		Predicted: Yes	Predicted: No
Actual	Yes	4	1
Actual	No	1	3

NOTE: This **False Positive** comes from someone we know **Does Not Love Troll 2**, but has a predicted probability of **0.94**.

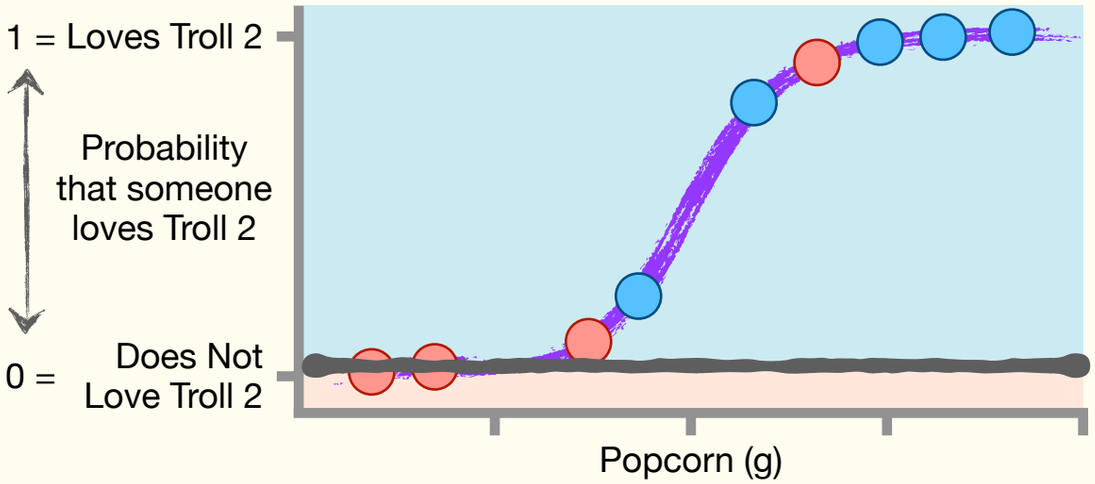
NOTE: This **False Negative** comes from someone we know **Loves Troll 2**, but has a predicted probability of **0.27**.



ROC: Main Ideas Part 2

3 Now let's talk about what happens when we use a different classification threshold for deciding if someone **Loves Troll 2** or not.

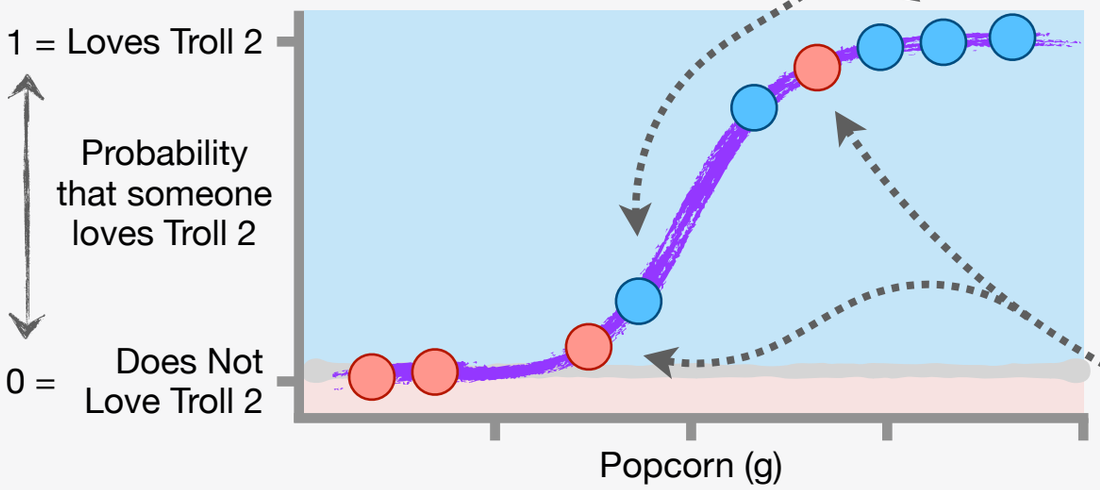
For example, if it was super important to correctly classify every single person who **Loves Troll 2**, we could set the threshold to **0.01**.



NOTE: If the idea of using a classification threshold other than **0.5** is blowing your mind, imagine we're trying to classify people with the Ebola virus. In this case, it's absolutely essential that we correctly identify every single person with the virus to minimize the risk of an outbreak. And that means lowering the threshold, even if it results in more **False Positives**.

4 When the classification threshold is **0.01**, we correctly classify everyone who **Loves Troll 2**…

…and that means there are **0 False Negatives**…

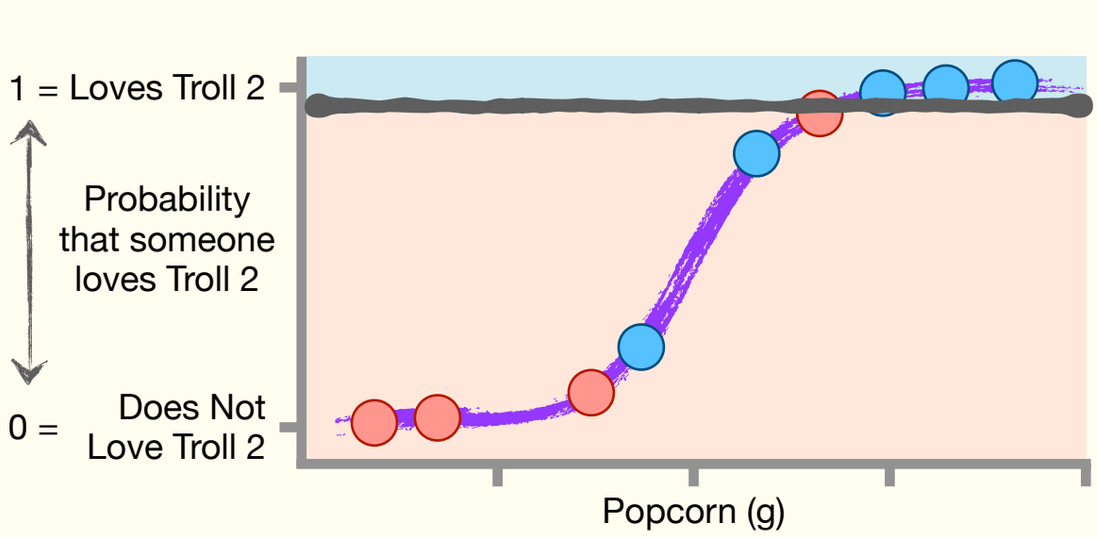


		Predicted	
		Yes	No
Actual	Yes	5	0
	No	2	2

…but we also increase the number of **False Positives** to **2** because these two people we know **Do Not Love Troll 2** are now predicted to **Love Troll 2**.

ROC: Main Ideas Part 3

5 On the other hand, if it was super important to correctly classify everyone who **Does Not Love Troll 2**, we could set the classification threshold to **0.95**…



Gentle Reminder:

		Predicted: Yes	Predicted: No
Actual	Yes	**TP**	FN
Actual	No	FP	**TN**

True Positive

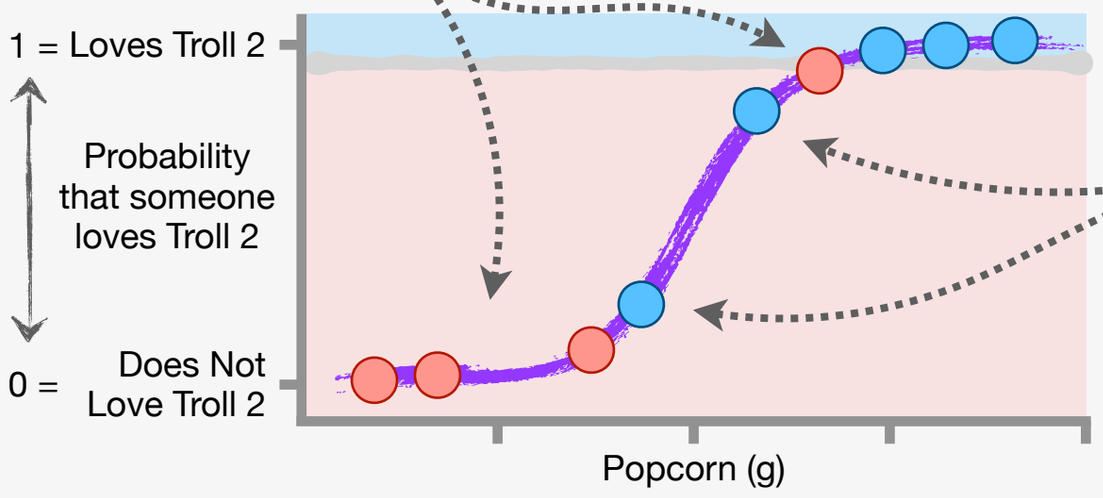
False Negative

False Positive

True Negative

6 …and now we would have **0 False Positives** because all of the people that we know **Do Not Love Troll 2** would be correctly classified…

…but now we would have **2 False Negatives** because two people that we know **Love Troll 2** would now be incorrectly classified as people who **Do Not Love Troll 2**…

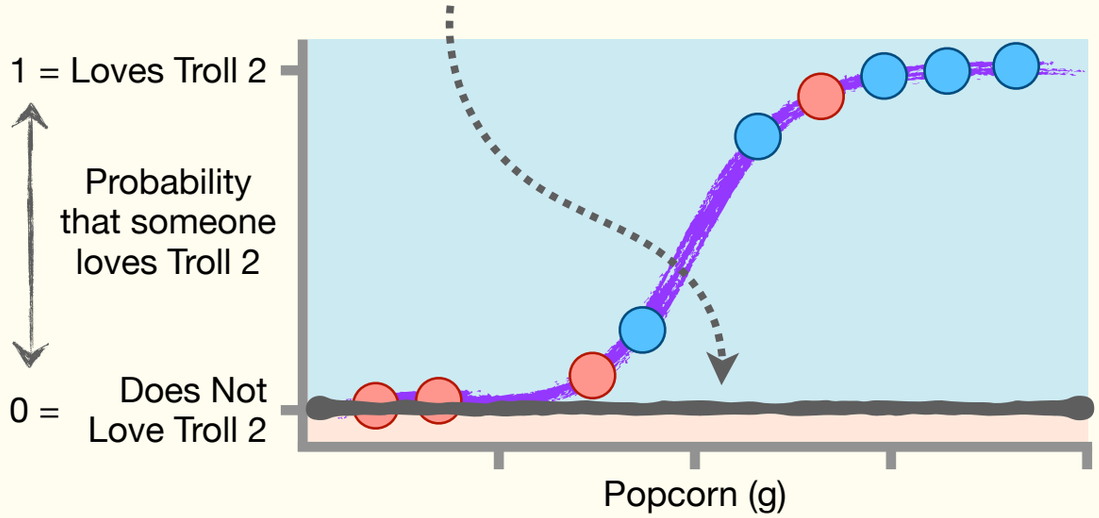


…and we would end up with this **Confusion Matrix**.

		Predicted: Yes	Predicted: No
Actual	Yes	**3**	2
Actual	No	0	**4**

ROC: Main Ideas Part 4

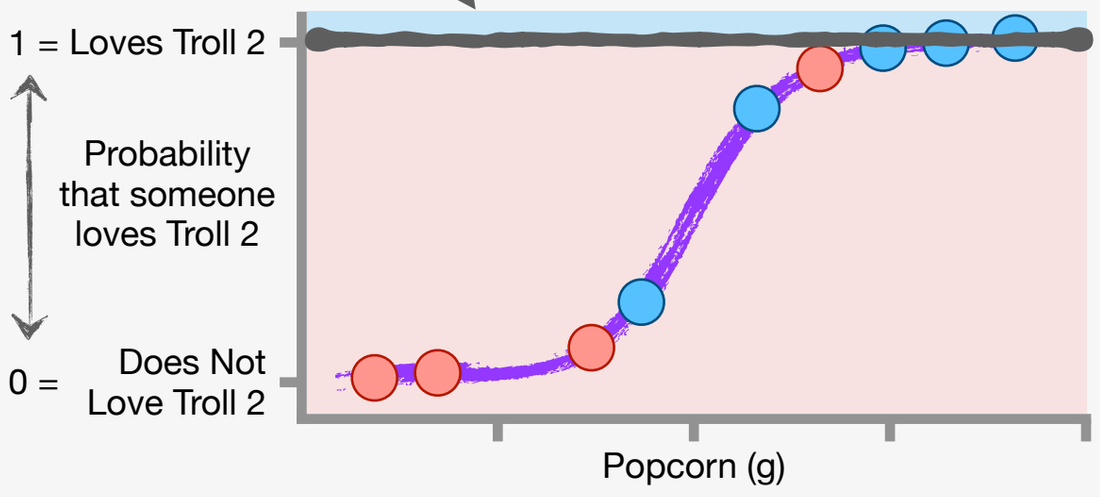
7 We could also set the classification threshold to **0**, and classify everyone as someone who **Loves Troll 2**…



…and that would give us this **Confusion Matrix**.

		Predicted: Yes	Predicted: No
Actual	Yes	5	0
Actual	No	4	0

8 Or, we could set the classification threshold to **1** and classify everyone as someone who **Does Not Love Troll 2**…

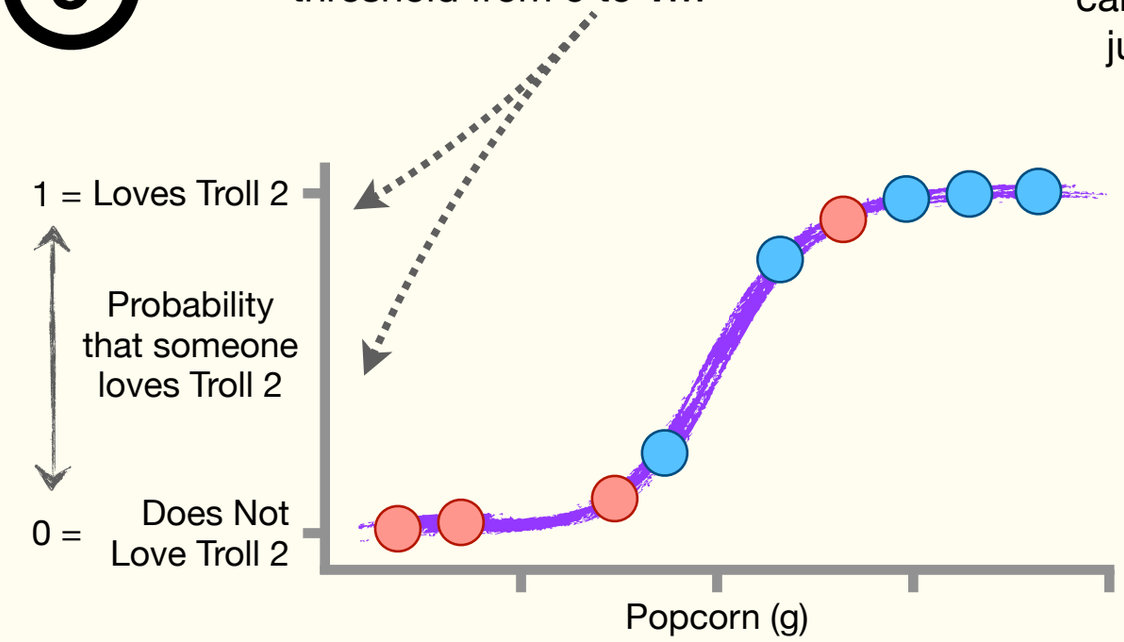


…and that would give us this **Confusion Matrix**.

		Predicted: Yes	Predicted: No
Actual	Yes	0	5
Actual	No	0	4

ROC: Main Ideas Part 5

9 Ultimately, we can try any classification threshold from **0** to **1**…



…and when we do, we end up with **10** different **Confusion Matrices** that we can choose from. (**NOTE:** The threshold under each **Confusion Matrix** is just one of many that will result in the exact same **Confusion Matrix**).

		Predicted: Yes	Predicted: No
Actual	Yes	5	0
Actual	No	4	0

Threshold = 0

		Predicted: Yes	Predicted: No
Actual	Yes	5	0
Actual	No	3	1

Threshold = 0.007

		Predicted: Yes	Predicted: No
Actual	Yes	5	0
Actual	No	2	2

Threshold = 0.0645

		Predicted: Yes	Predicted: No
Actual	Yes	5	0
Actual	No	1	3

Threshold = 0.195

		Predicted: Yes	Predicted: No
Actual	Yes	4	1
Actual	No	1	3

Threshold = 0.5

		Predicted: Yes	Predicted: No
Actual	Yes	3	2
Actual	No	1	3

Threshold = 0.87

		Predicted: Yes	Predicted: No
Actual	Yes	3	2
Actual	No	0	4

Threshold = 0.95

		Predicted: Yes	Predicted: No
Actual	Yes	2	3
Actual	No	0	4

Threshold = 0.965

		Predicted: Yes	Predicted: No
Actual	Yes	1	4
Actual	No	0	4

Threshold = 0.975

		Predicted: Yes	Predicted: No
Actual	Yes	0	5
Actual	No	0	4

Threshold = 1

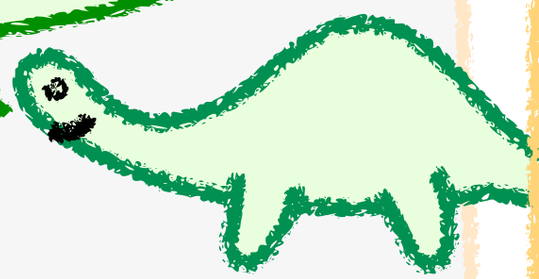
UGH!!! Trying to find the ideal classification threshold among all of these technicolor **Confusion Matrices** is tedious and annoying. Wouldn't it be awesome if we could consolidate them into one easy-to-interpret graph?

YES!!!

Well, we're in luck because that's what **ROC** graphs do!!!

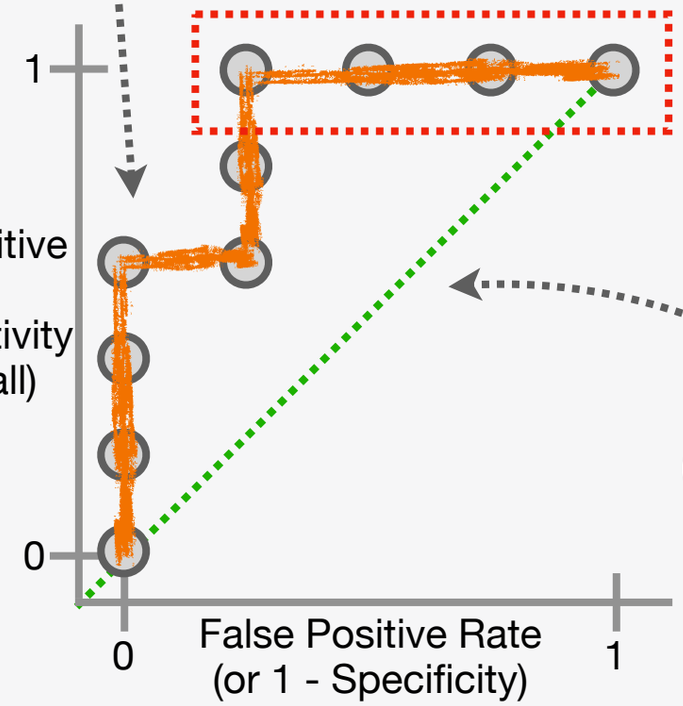
ROC: Main Ideas Part 6

ROC stands for **Receiver Operating Characteristic**, and the name comes from the graphs drawn during World War II that summarized how well radar operators correctly identified airplanes in radar signals.



10 **ROC** graphs are super helpful when we need to identify a good classification threshold because they summarize how well each threshold performed in terms of the **True Positive Rate** and the **False Positive Rate**.

a Each **gray dot** on the **ROC** graph tells us the **True Positive Rate** and the **False Positive Rate** for a specific classification threshold.



b The higher the dot is along the y-axis, the higher the percentage of actual **Positives** were *correctly* classified…

c …and the further to the left along the x-axis, the lower the percentage of actual **Negatives** that were *incorrectly* classified.

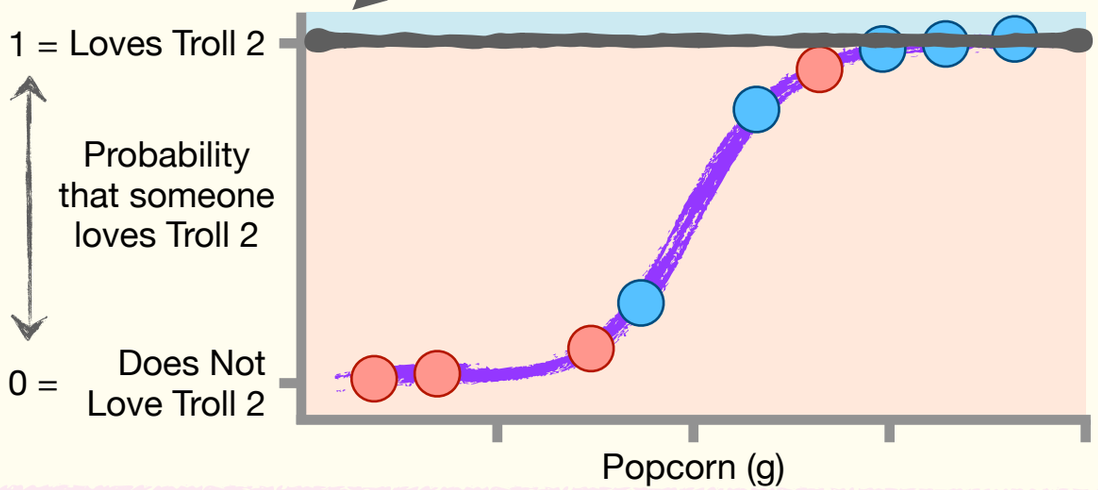
d The **diagonal line** shows where the **True Positive Rate** = **False Positive Rate**.

e At a glance, we can look at the top row of points and tell that the classification threshold that resulted in the point on the left side performed better than the others because they all have the same **True Positive Rate**, but the point on the left has a lower **False Positive Rate**.

11 Now that we understand the main ideas behind **ROC** graphs, let's dive into the details!!!

ROC: Details Part 1

1 To get a better sense of how an **ROC** graph works, let's draw one from start to finish. We'll start by using a classification threshold, **1**, that classifies everyone as someone who **Does Not Love Troll 2**…



Gentle Reminder:

		Predicted: Yes	Predicted: No
Actual	Yes	**TP** (True Positive)	FN (False Negative)
Actual	No	FP (False Positive)	**TN** (True Negative)

…and when the classification threshold is set to **1**, we end up with this **Confusion Matrix**.

		Predicted: Yes	Predicted: No
Actual	Yes	**0**	5
Actual	No	0	**4**

Threshold = 1

2 Using the values in the **Confusion Matrix**, we can calculate the **True Positive Rate**…

$$\textbf{True Positive Rate} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

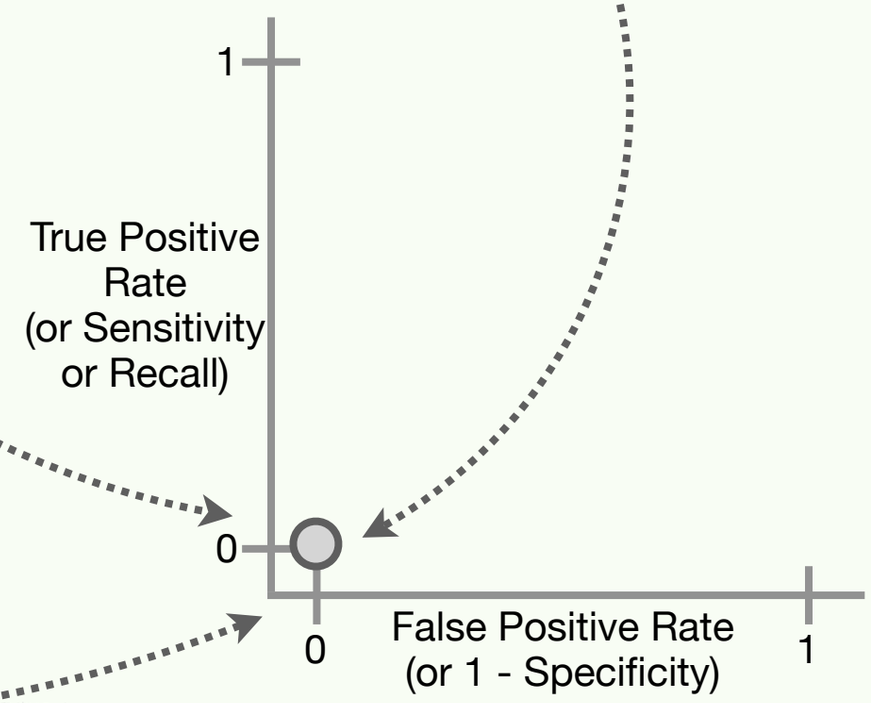
$$= \frac{0}{0 + 5} = 0$$

3 …and the **False Positive Rate**…

$$\textbf{False Positive Rate} = \frac{\text{False Positives}}{\text{False Positives} + \text{True Negatives}}$$

$$= \frac{0}{0 + 4} = 0$$

4 …and then we can plot that point, **(0, 0)**, on the **ROC** graph.

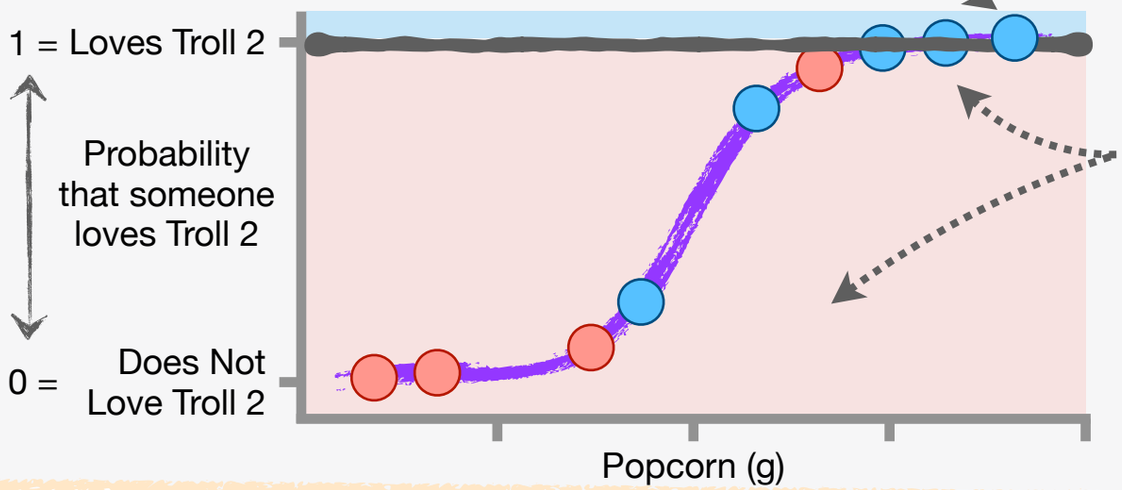


ROC: Details Part 2

Gentle Reminder:

		Predicted: Yes	Predicted: No
Actual	Yes	TP (True Positive)	FN (False Negative)
Actual	No	FP (False Positive)	TN (True Negative)

5 Now let's lower the classification threshold to **0.975**, which is just enough to classify one person as someone who **Loves Troll 2**…



…and everyone else is classified as someone who **Does Not Love Troll 2**…

…and that gives us this **Confusion Matrix**.

		Predicted: Yes	Predicted: No
Actual	Yes	1	4
Actual	No	0	4

Threshold = 0.975

6 Using the values in the **Confusion Matrix**, we can calculate the **True Positive Rate**…

$$\textbf{True Positive Rate} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

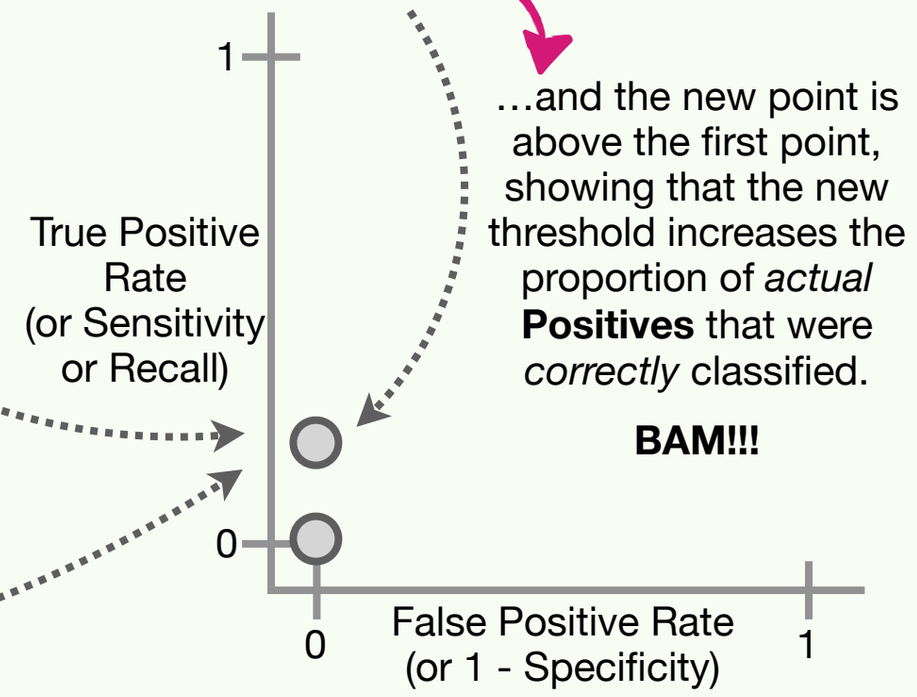
$$= \frac{1}{1+4} = 0.2$$

7 …and the **False Positive Rate**…

$$\textbf{False Positive Rate} = \frac{\text{False Positives}}{\text{False Positives} + \text{True Negatives}}$$

$$= \frac{0}{0+4} = 0$$

8 …and then we can plot that point, **(0, 0.2)**, on the **ROC** graph…



…and the new point is above the first point, showing that the new threshold increases the proportion of *actual* **Positives** that were *correctly* classified.

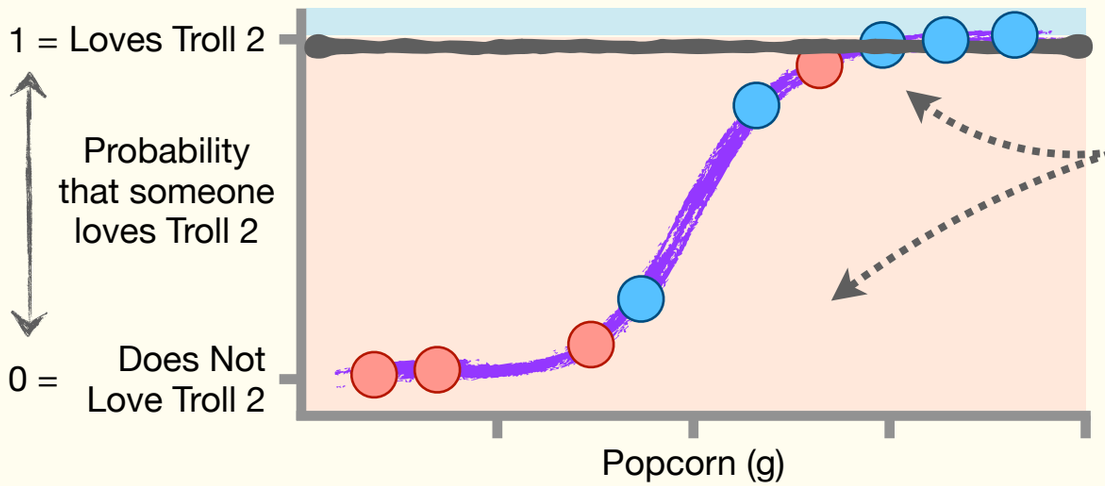
BAM!!!

ROC: Details Part 3

Gentle Reminder:

		Predicted: Yes	Predicted: No
Actual	Yes	TP (True Positive)	FN (False Negative)
Actual	No	FP (False Positive)	TN (True Negative)

9 Now let's lower the classification threshold to **0.965**, which is just enough to classify **2** people as people who **Love Troll 2**…



…and everyone else is classified as someone who **Does Not Love Troll 2**…

…and that gives us this **Confusion Matrix**.

		Predicted: Yes	Predicted: No
Actual	Yes	2	3
Actual	No	0	4

Threshold = 0.965

10 Using the values in the **Confusion Matrix**, we can calculate the **True Positive Rate**…

$$\textbf{True Positive Rate} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

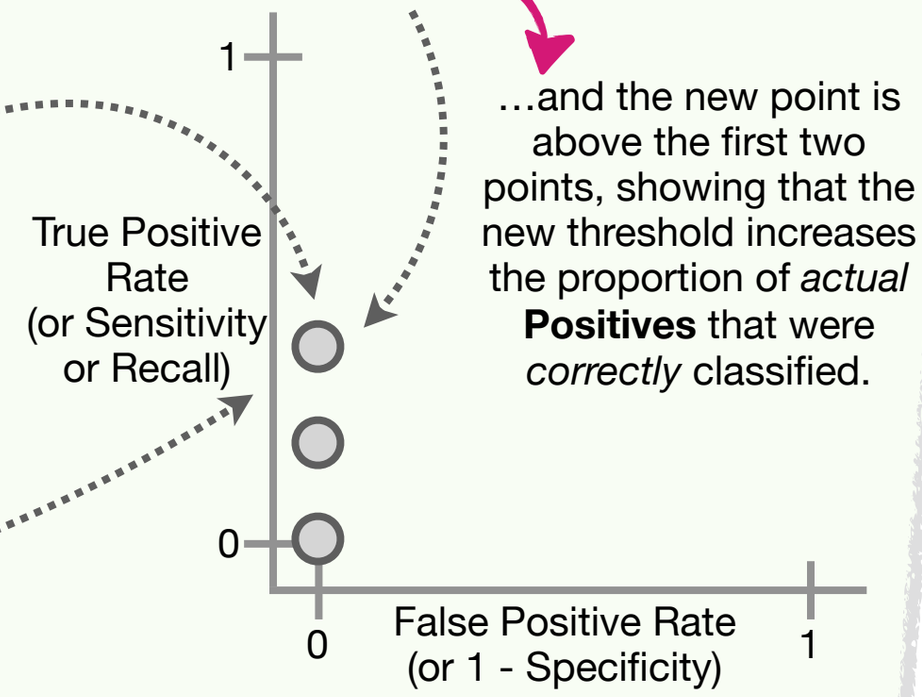
$$= \frac{2}{2+3} = 0.4$$

11 …and the **False Positive Rate**…

$$\textbf{False Positive Rate} = \frac{\text{False Positives}}{\text{False Positives} + \text{True Negatives}}$$

$$= \frac{0}{0+4} = 0$$

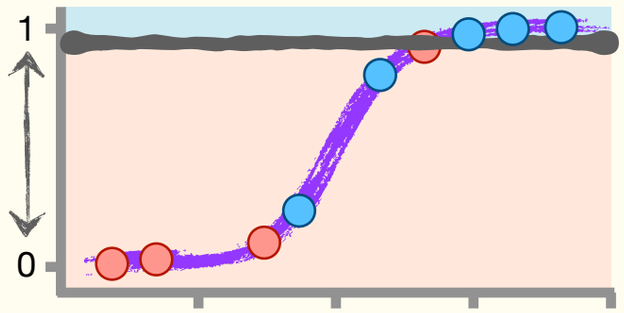
12 …and then we can plot that point, **(0, 0.4)**, on the **ROC** graph…



…and the new point is above the first two points, showing that the new threshold increases the proportion of *actual* **Positives** that were *correctly* classified.

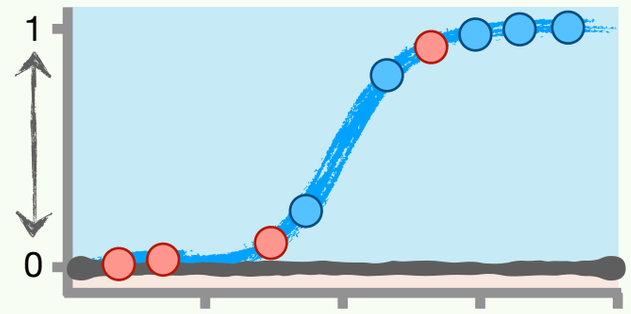
ROC: Details Part 4

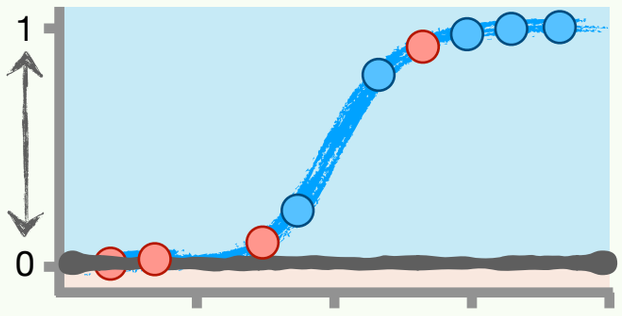
13 Likewise, for each threshold that increases the number of **Positive** classifications (in this example, that means classifying a person as someone who **Loves Troll 2**), we calculate the **True Positive Rate** and **False Positive Rate** until everyone is classified as **Positive**.

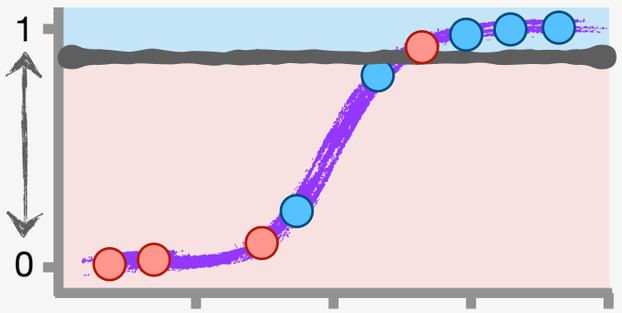


		Predicted	
		Yes	No
Actual	Yes	3	2
	No	0	4

Threshold = 0.95

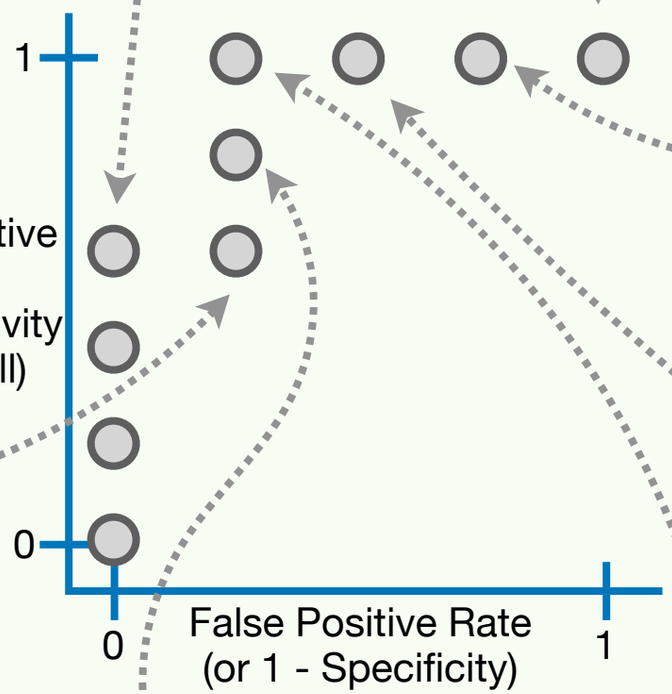


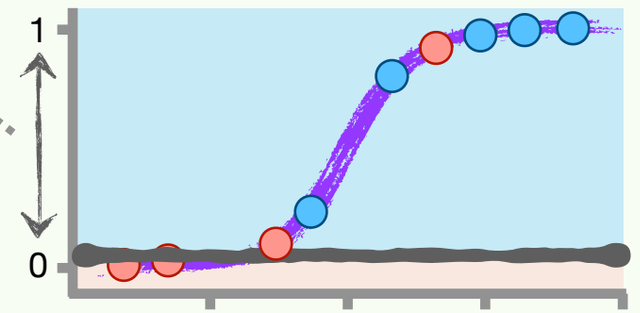


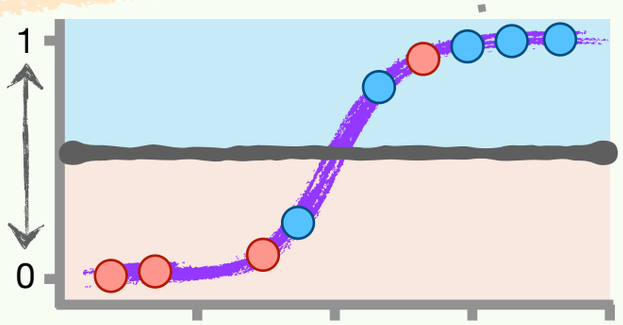


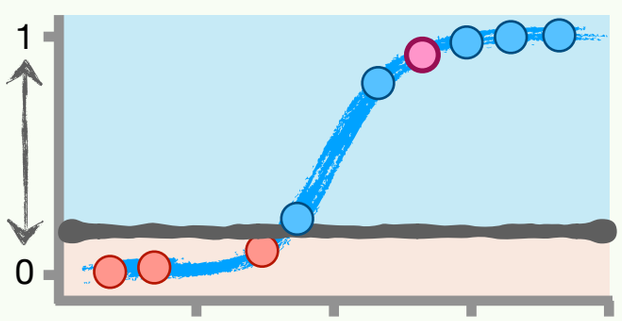
		Predicted	
		Yes	No
Actual	Yes	3	2
	No	1	3

Threshold = 0.87





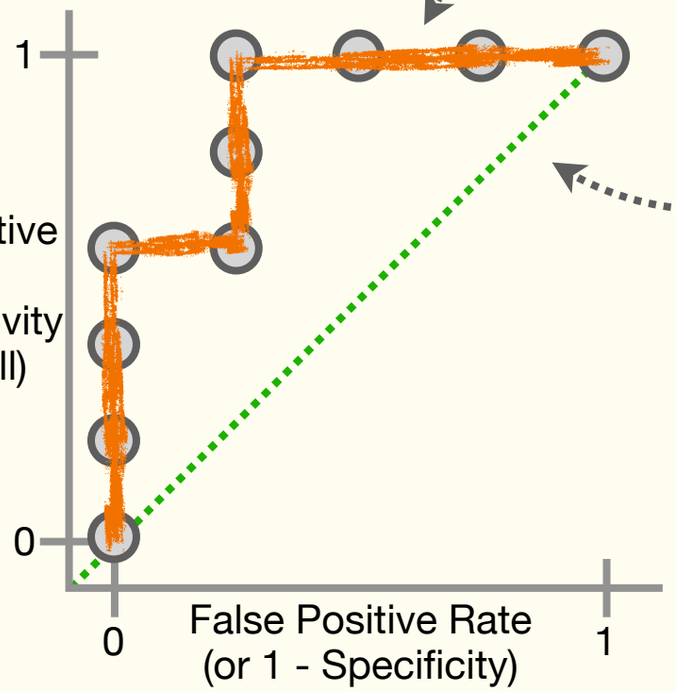




ROC: Details Part 5

14 After we finish plotting the points from each possible **Confusion Matrix**, we usually **connect the dots**…

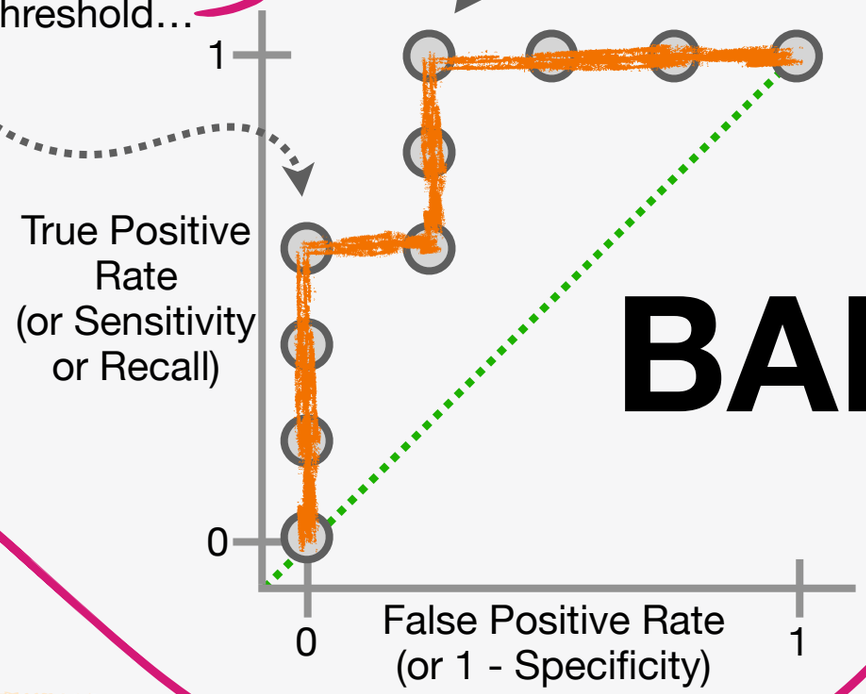
…and add a **diagonal line** that tells us when the **True Positive Rate = False Positive Rate.**



15 Now, without having to sort through a huge pile of **Confusion Matrices**, we can use the **ROC** graph to pick a classification threshold.

If we want to avoid all **False Positives**, but want to maximize the number of actual **Positives** correctly classified, we would pick this threshold…

…but if we can tolerate a few **False Positives**, we would pick this threshold because it *correctly* classifies all of the actual **Positives**.



BAM!!!

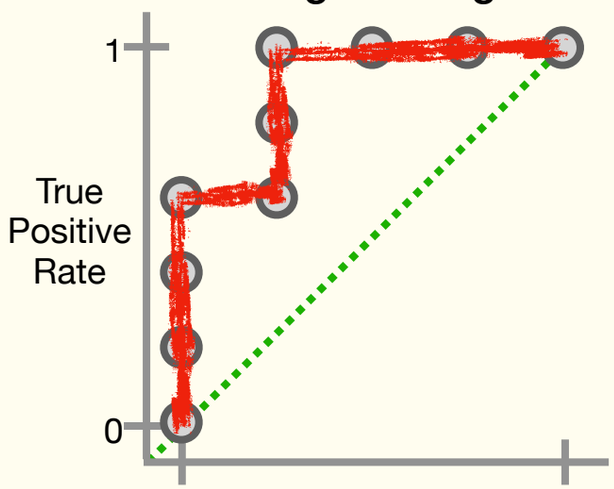
16 **ROC** graphs are great for selecting an optimal classification threshold for a model. But what if we want to compare how one model performs vs. another? This is where the **AUC**, which stands for **Area Under the Curve**, comes in handy. So read on!!!

AUC: Main Ideas

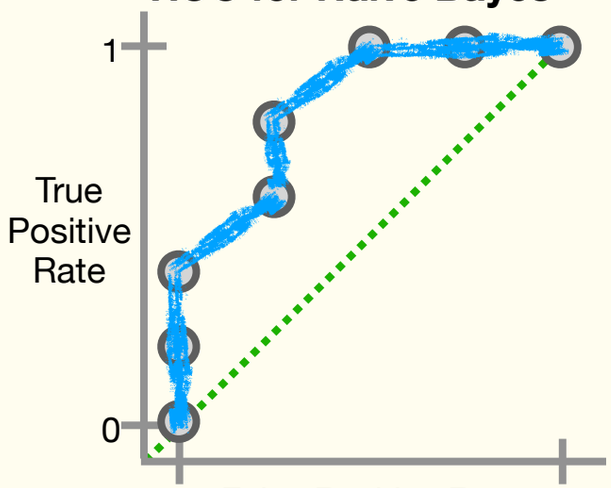
1 Now, imagine we created **Logistic Regression** and **Naive Bayes** models and tested them with the same data, and we wanted to know which model performed better.

In theory, we could compare the individual **ROC** graphs, and when we only have two models to compare, this is a pretty good option.

ROC for Logistic Regression



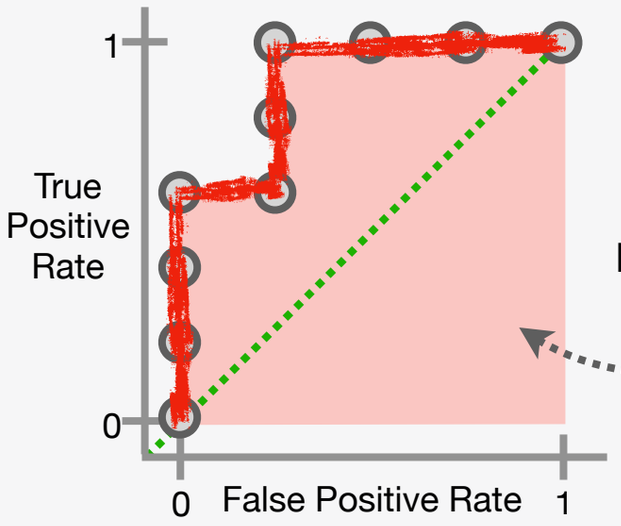
ROC for Naive Bayes



However, if we wanted to compare a bunch of models, this would be just as tedious as comparing a bunch of **Confusion Matrices**.

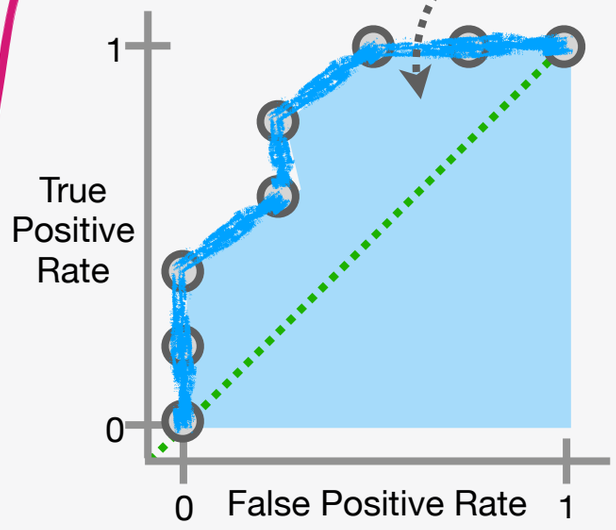
UGH!!!

2 So, instead of comparing a bunch of **ROC** graphs, one simple way to summarize them is to calculate and compare the **AUC:** the **Area Under** each **Curve**.



In this case, the **AUC** for **Logistic Regression** is **0.9**…

…and the **AUC** for **Naive Bayes** is **0.85**…



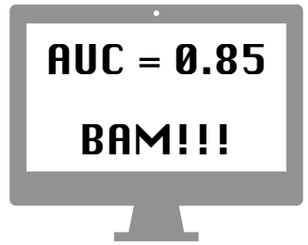
…and because **Logistic Regression** has a larger **AUC**, we can tell that, overall, **Logistic Regression** performs better than **Naive Bayes** with these data.

BAM!!!

AUC: FAQ

How do you calculate the AUC?

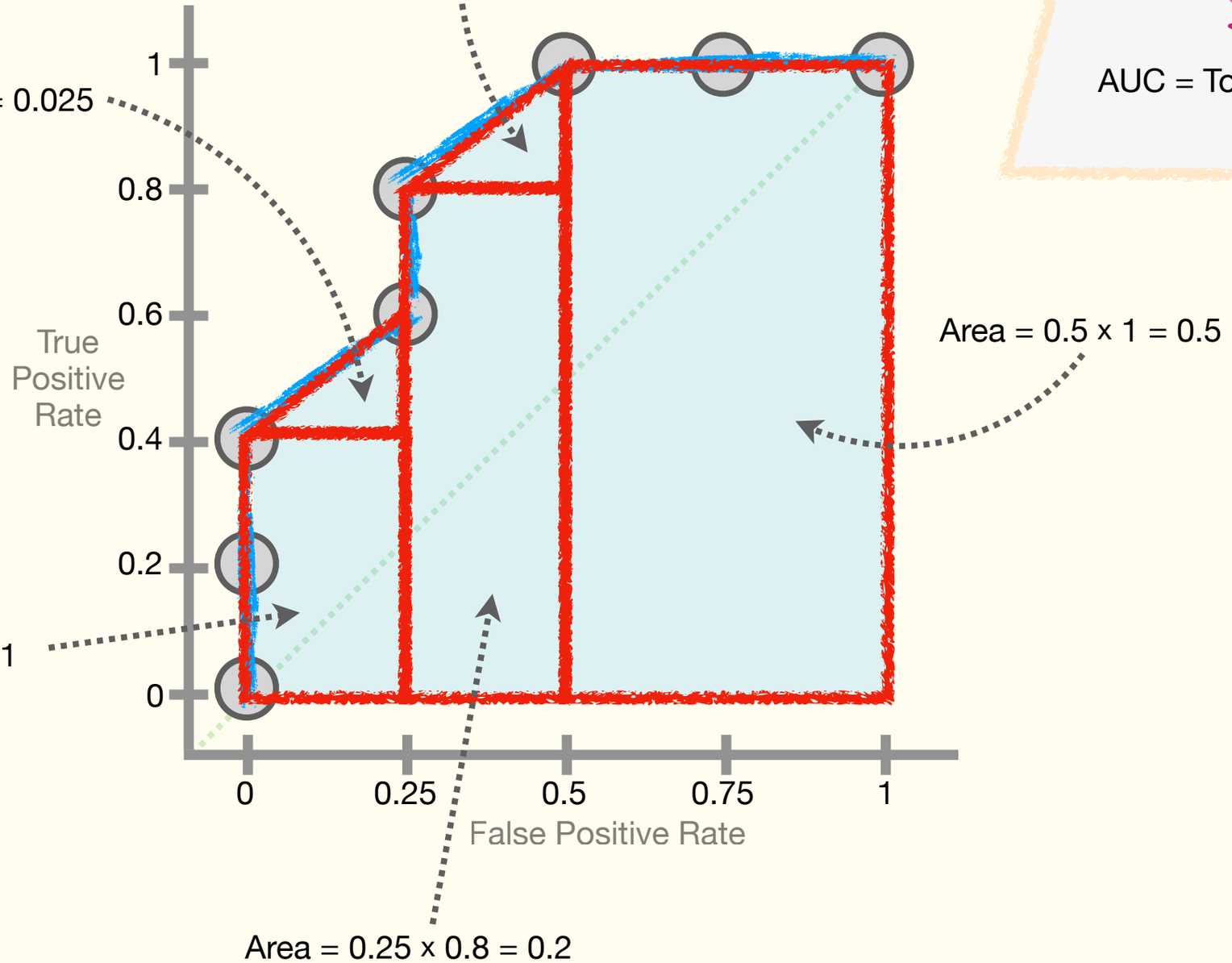
My favorite way is to get a computer to do it…



…but if you want to do it by hand, you simply divide the area into a bunch of **rectangles** and **triangles** and add the areas of each shape together.

$$\text{Area} = \frac{1}{2}\ 0.25 \times 0.2 = 0.025$$

$$\text{Area} = \frac{1}{2}\ 0.25 \times 0.2 = 0.025$$



$$\text{Area} = 0.5 \times 1 = 0.5$$

$$\text{Area} = 0.25 \times 0.4 = 0.1$$

$$\text{Area} = 0.25 \times 0.8 = 0.2$$

$$\begin{array}{r} 0.500 \\ 0.200 \\ 0.100 \\ 0.025 \\ +\ 0.025 \\ \hline \end{array}$$

$$\text{AUC} = \text{Total Area} = 0.850$$

BUT WAIT!!!

THERE'S MORE!!!