Linear Supervised Learning

M. Ndaoud

Machine Learning - Recap

Supervised Learning

Definition: Supervised learning uses labeled data to train models. The model learns from input-output pairs and can predict outcomes for new, unseen data.

Common Use Cases: Classification (spam detection, disease diagnosis) and Regression (predicting housing prices, stock trends).

Key Methods: Linear Regression, Logistic Regression, Decision Trees, Random Forests, Support Vector Machines (SVM), K-Nearest Neighbors (KNN), Neural Networks

Unsupervised Learning

Definition: Unsupervised learning works with unlabeled data. The model finds patterns, structures, or groupings without pre-existing labels or categories.

Common Use Cases: Clustering (customer segmentation, image grouping) and Dimensionality Reduction (reducing features while preserving information).

Key Methods: K-Means, Clustering, Hierarchical Clustering, Principal Component Analysis (PCA), DBSCAN (Density-Based Spatial Clustering), Association Rule Learning (e.g., Apriori, used for market basket analysis)

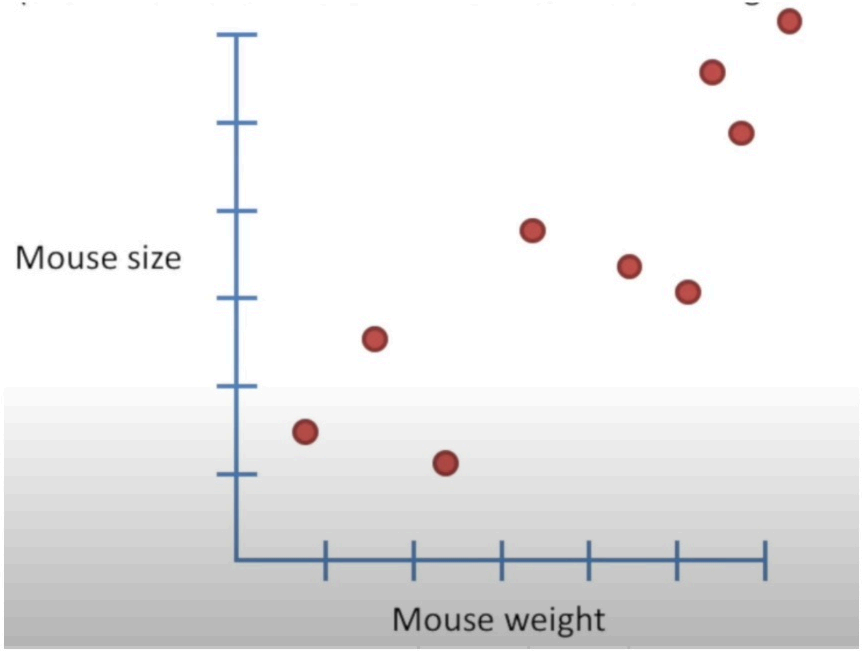
Linear Regression - Recap

Predicting continuous outcomes

Can we predict the size of a mouse based on its weight?

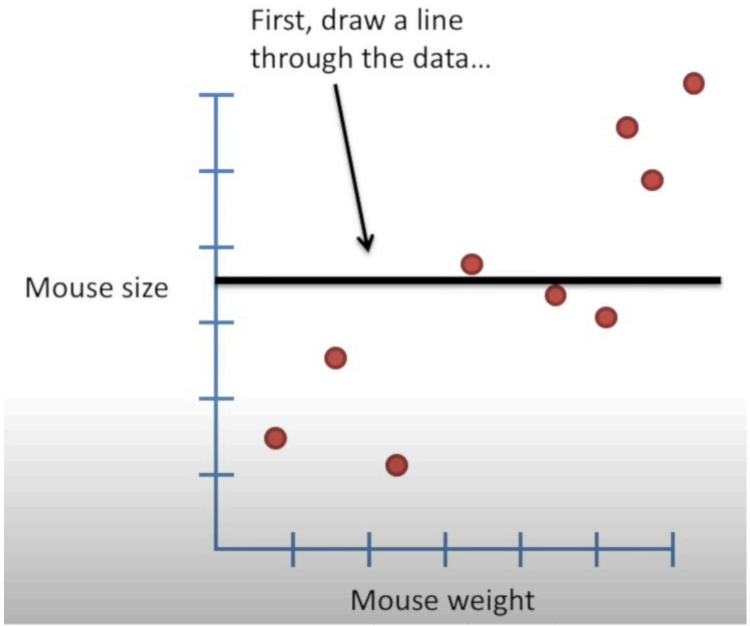
Here are a few data points we have collected on mouses.

Linear regression is about fitting a line to these data.



Linear Regression - Recap

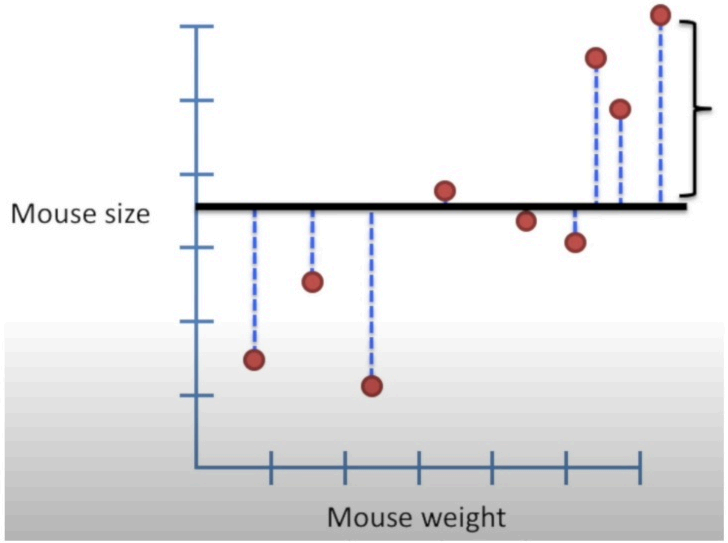
Predicting continuous outcomes



First, draw a line through the data

Linear Regression - Recap

Predicting continuous outcomes



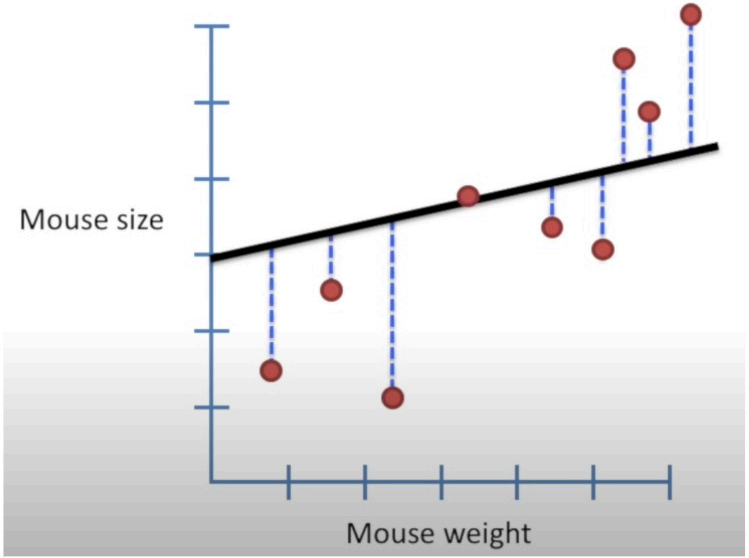
Second, calculate sum of squared residuals

Measure the distance from the line to each data point, square these distances, and add them up

Note: We square them so that negative distances do not cancel out positive distances

Linear Regression - Recap

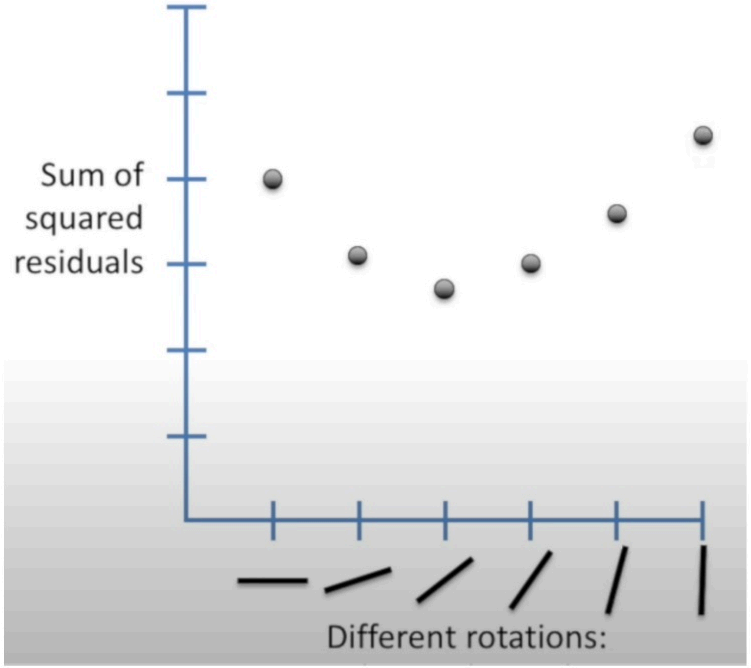
Predicting continuous outcomes



Third, rotate the line and reproduce the process

Linear Regression - Recap

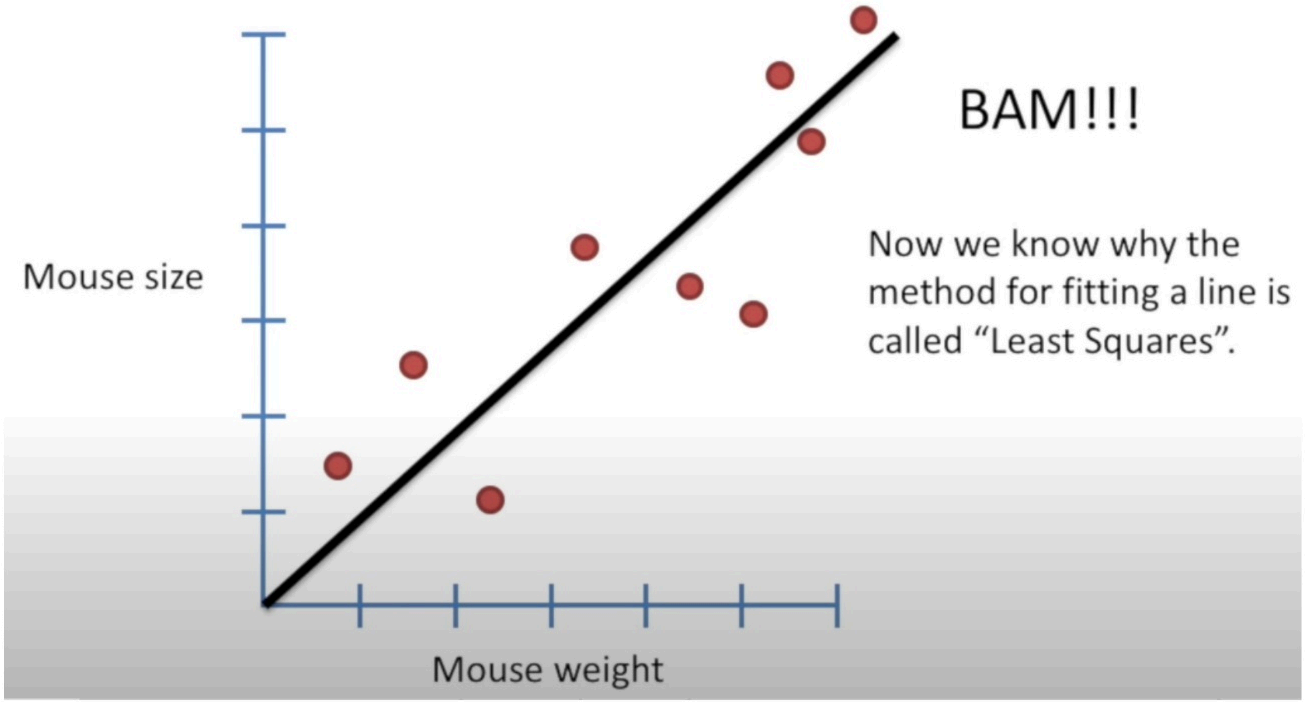
Predicting continuous outcomes



Finally, plot the results and find the rotation that has the least sum of squared residuals

Linear Regression - Recap

Predicting continuous outcomes



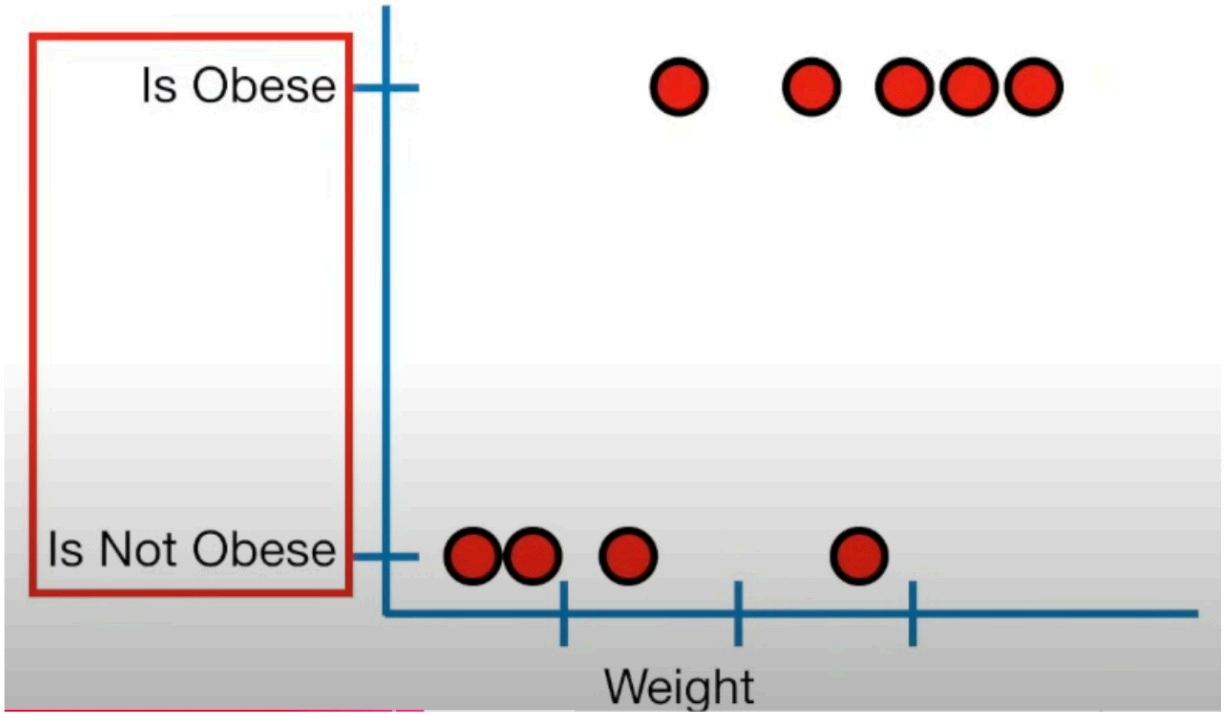
And we can predict the size of any new mouse, based on its weight !

Credits to StatQuest for these slides

This is great, but what if instead of the size, we want to predict if a mouse is obese (Y/N) based on its weight?

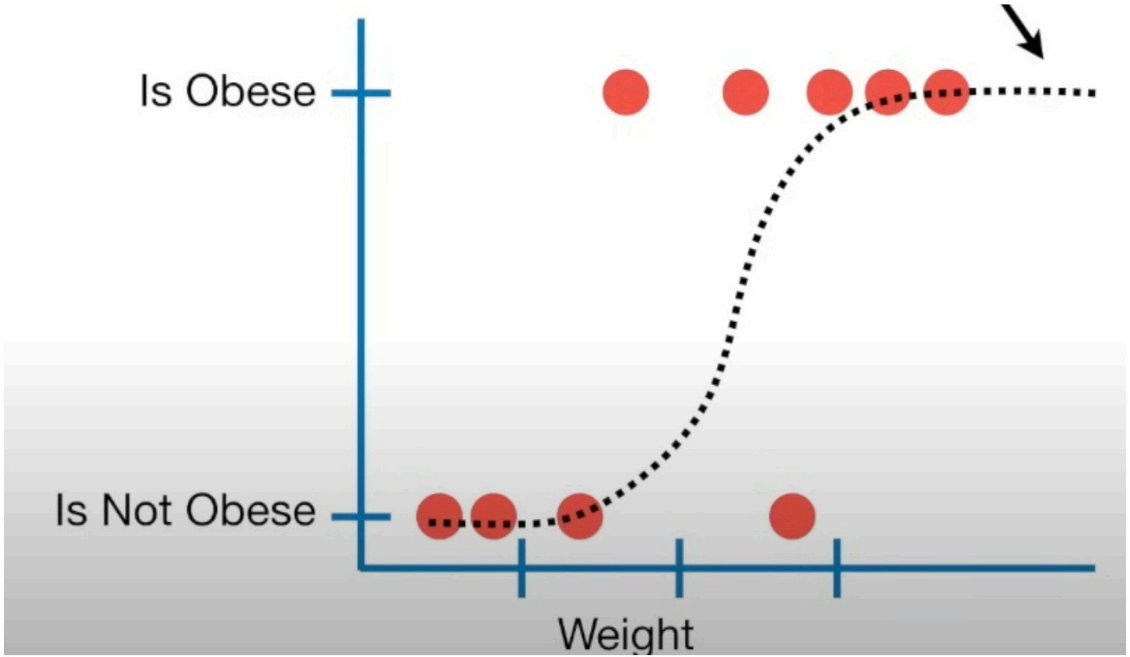
Logistic Regression - Intuition

If you are trying to fit a straight line here, and predict values based on that, you might end up with unrealistic values...

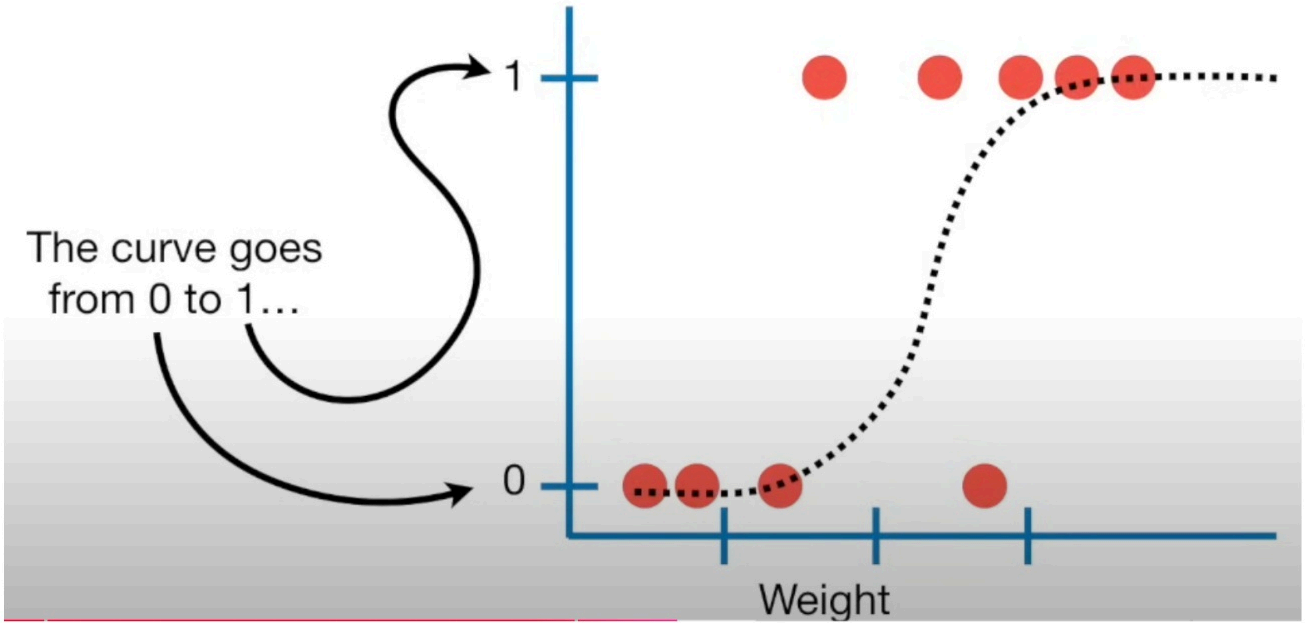


Logistic Regression - Intuition

Instead, we fit an S-curve called "Logistic Function"

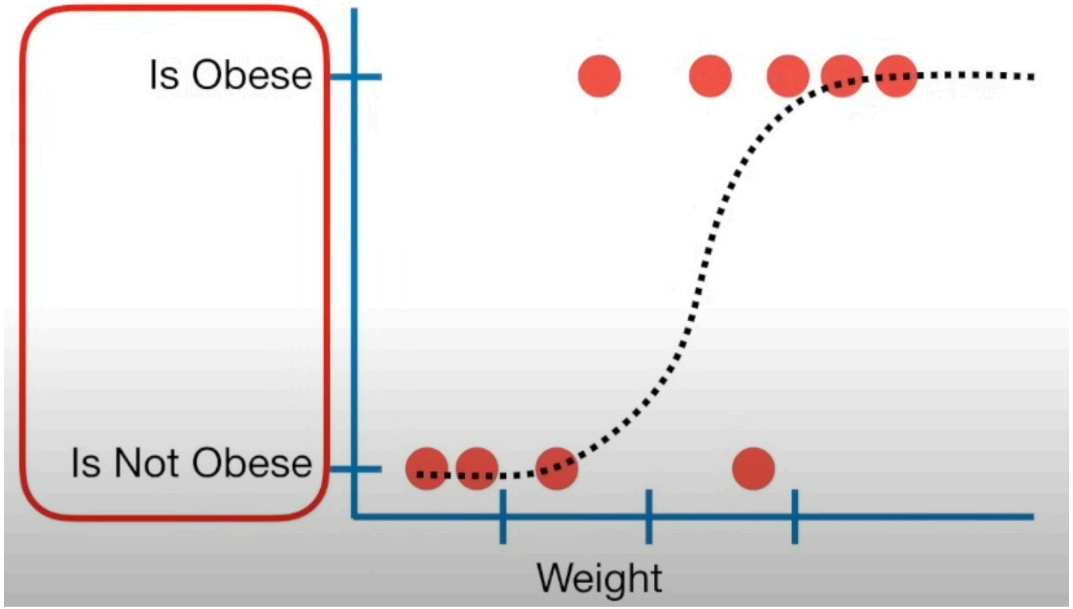


Logistic Regression - Intuition



…and represents the probability that a mouse is obese given its weight

Logistic Regression - Intuition



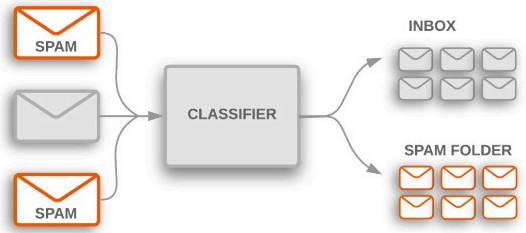
Logistic regression can be used to classify samples...

.. but also to assess what variables are useful for classifying samples

What is Classification?

Predicting the category or class to which a new observation belongs.

Classification models can be divided into two main types: binary classification, where there are only two possible classes, and multi-class classification, where there are multiple classes to choose from.



Classification Tasks

- ## Spam detection
  Classifying emails as spam or non-spam to protect users from unwanted messages.

- ## Disease prediction
  Identifying the likelihood of a person developing a specific disease based on their medical history and symptoms.

- ## Sentiment analysis
  Determining the emotional tone of text, such as reviews or social media posts, to gauge public opinion.

- ## Image recognition
  Identifying and classifying the contents of images, such as objects, people, or scenes.

- ## Fraud detection
  Identifying suspicious financial transactions or activities to prevent and detect fraud.

Classification Process

- ## Data collection and preprocessing

  Gather relevant data, clean and preprocess it to prepare it for feature engineering and model training. This may involve handling missing values, outliers, and encoding categorical variables.

- ## Feature engineering

  Identify and create relevant features from the preprocessed data that can help the model learn patterns and make accurate predictions. This may include feature selection, transformation, and dimensionality reduction techniques.

- ## Model selection and training

  Select an appropriate machine learning algorithm for the classification task and train the model using the engineered features. This involves hyperparameter tuning and optimization to achieve the best model performance.

- ## Model evaluation

  Evaluate the trained model's performance using appropriate metrics, such as accuracy, precision, recall, and F1-score. This step helps assess the model's effectiveness and identify areas for further improvement.

- ## Model deployment

  Deploy the trained and evaluated model into a production environment, where it can be used to make real-time predictions on new, unseen data. This may involve integrating the model with a web application or an API.

Popular Classification Algorithms

- ## Logistic Regression

  A statistical model used for binary classification tasks, where the goal is to predict whether an input belongs to one of two classes.

- ## Decision Trees

  A tree-based algorithm that makes predictions by recursively partitioning the input space based on feature values, creating a set of rules to make decisions.

- ## Random Forests

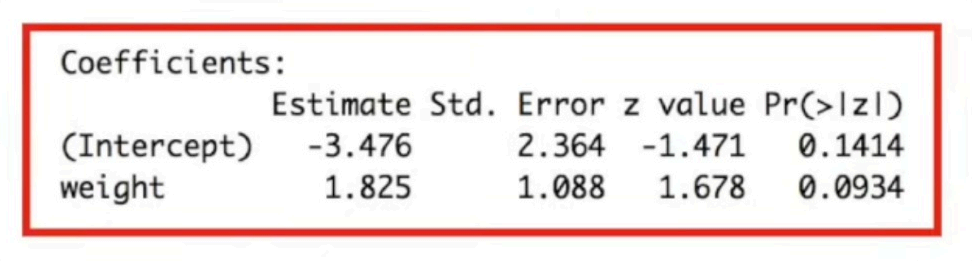
  An ensemble learning method that combines multiple decision trees to improve the accuracy and stability of the predictions, reducing the risk of overfitting.

- ## Support Vector Machines (SVMs)

  A supervised learning algorithm that finds the optimal hyperplane to separate different classes, maximizing the margin between the closest data points of each class.

- ## Neural Networks

  A machine learning model inspired by the human brain, consisting of interconnected nodes (neurons) that learn to map input data to output predictions through a process of optimization.

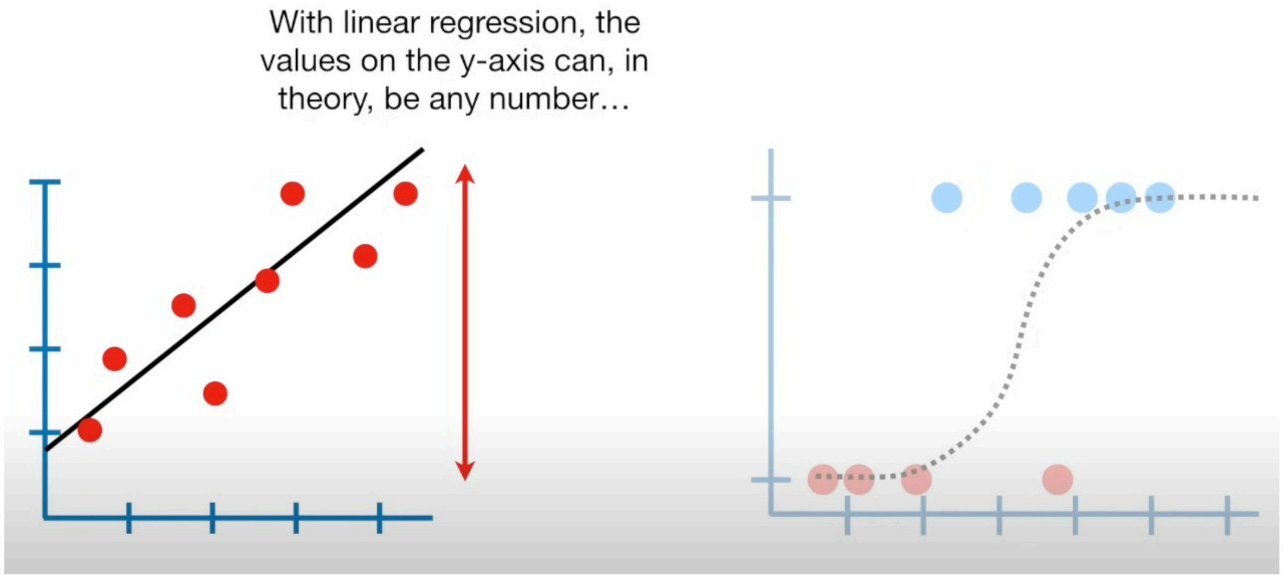


```
Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)   -3.476      2.364  -1.471   0.1414
weight         1.825      1.088   1.678   0.0934
```

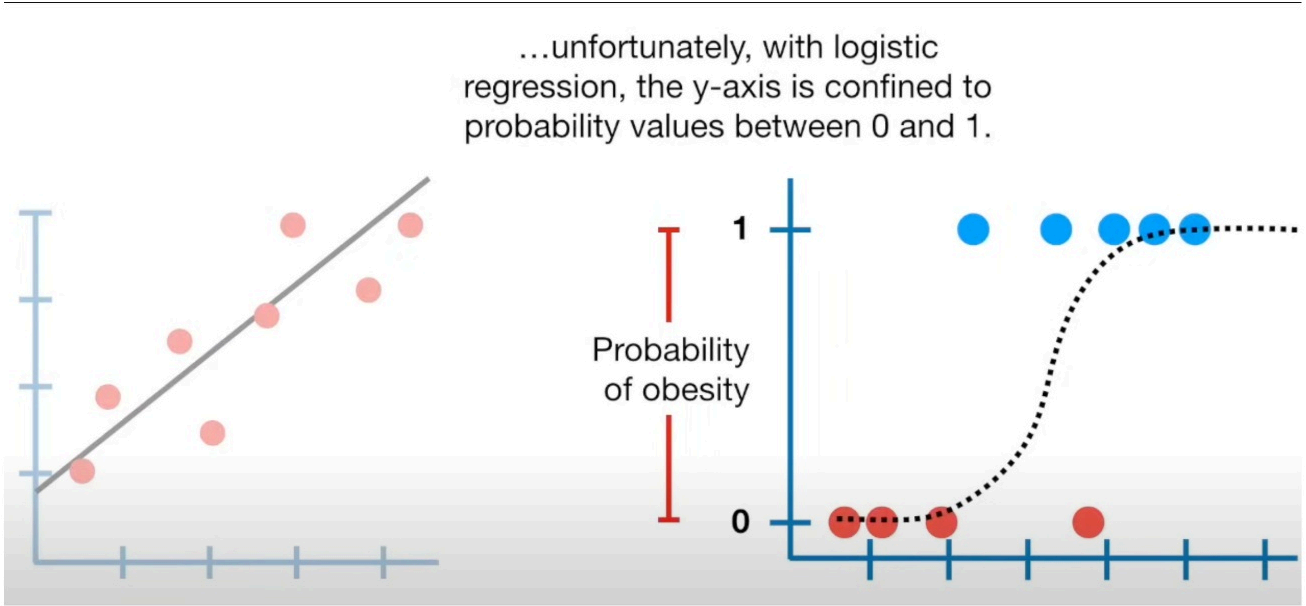
This is what you will get from your logistic regression model when you run it in your computer

I would like you to be able to understand and interpret those coefficients.

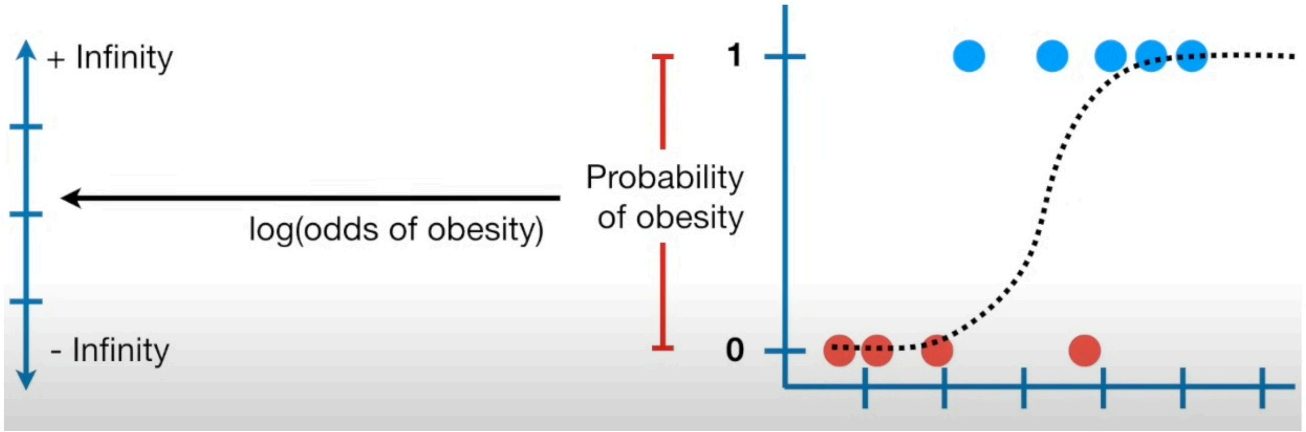
Logistic Regression - Coefficients



Logistic Regression - Coefficients



Logistic Regression - Coefficients

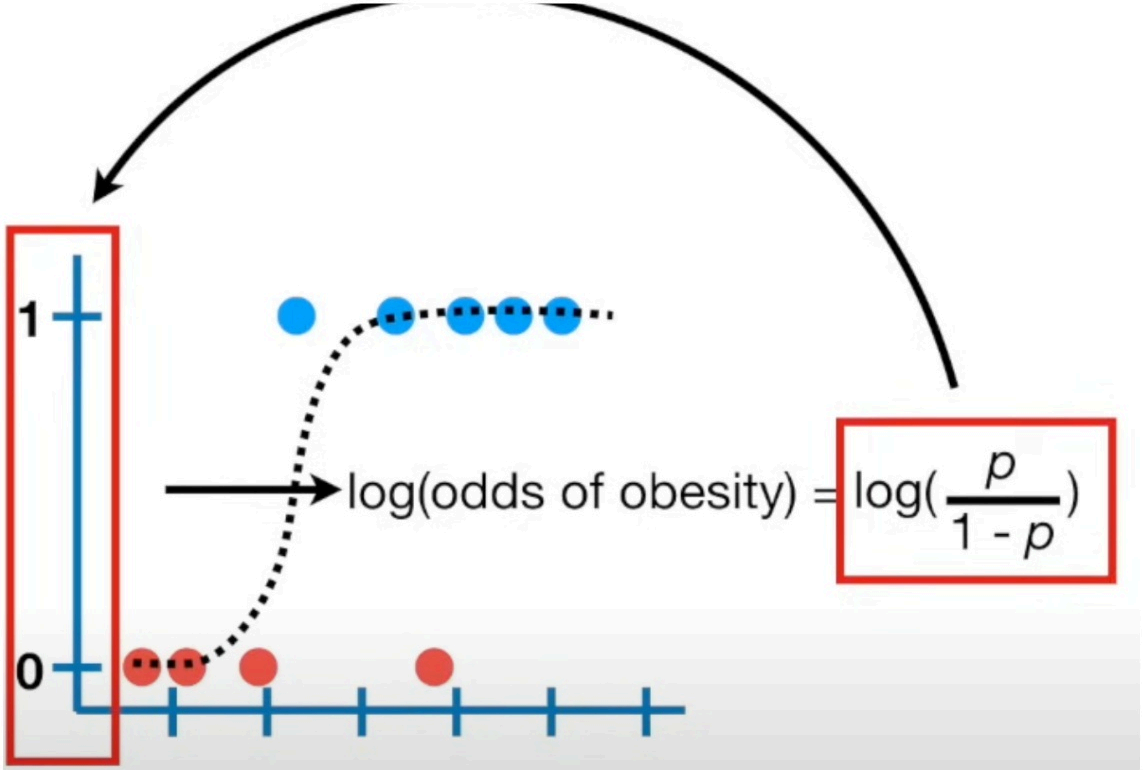


To solve the problem, we transform the probability into the log(odds of obesity)

Odds = probability of an event happening/probability of an event not happening

Taking the log(odd) transforms the problem into one where we can use a straight line

Logistic Regression - Coefficients

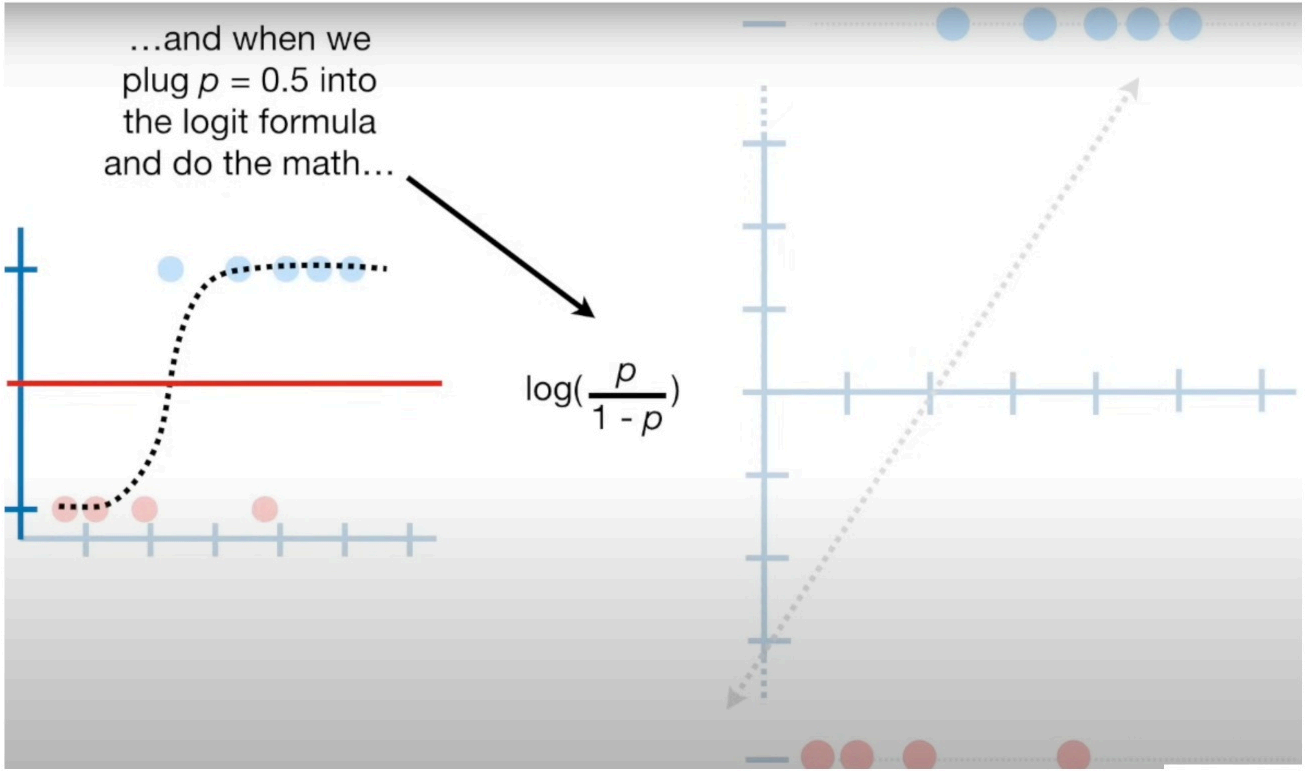


This is the pretty formula...

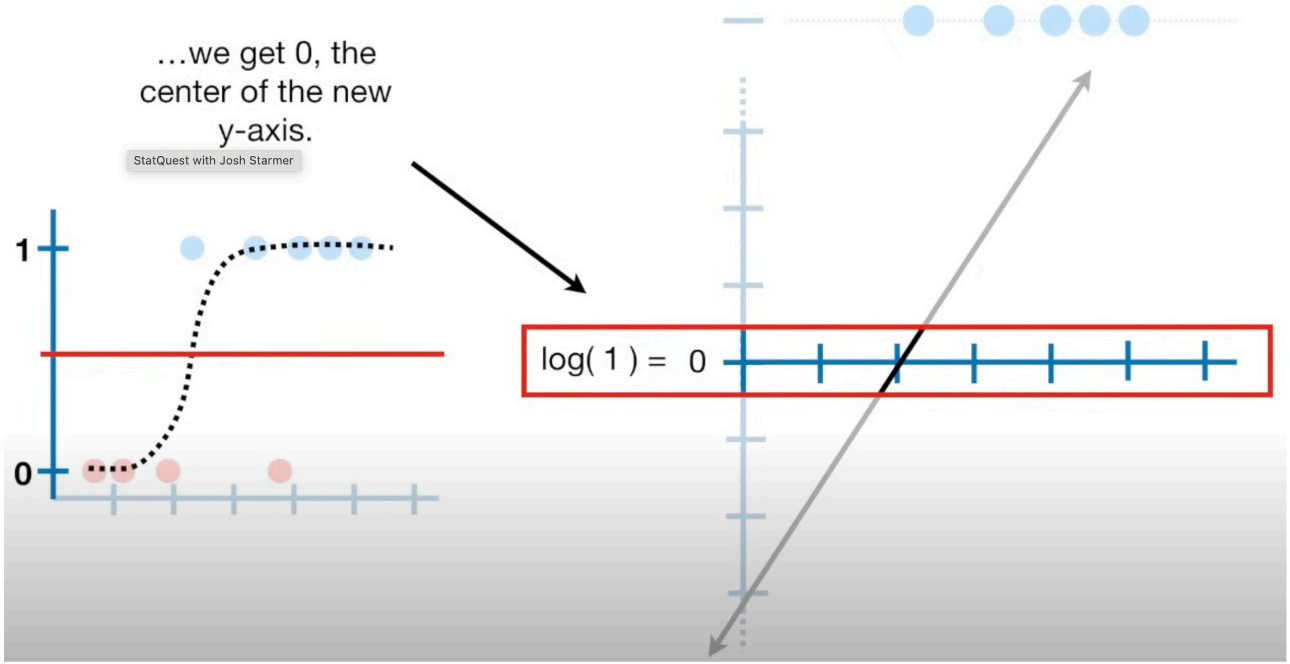
... and let's see how it works !

Logistic Regression - Coefficients

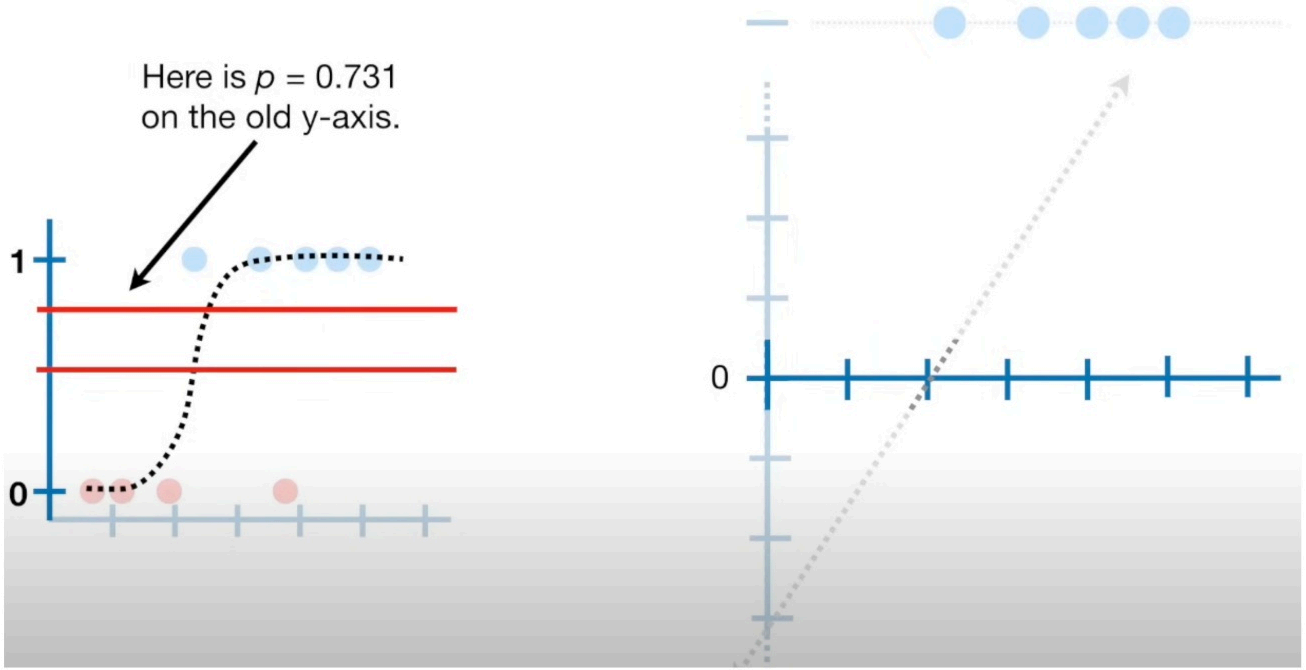
The mid point on the left corresponds to a probability of 0.5...



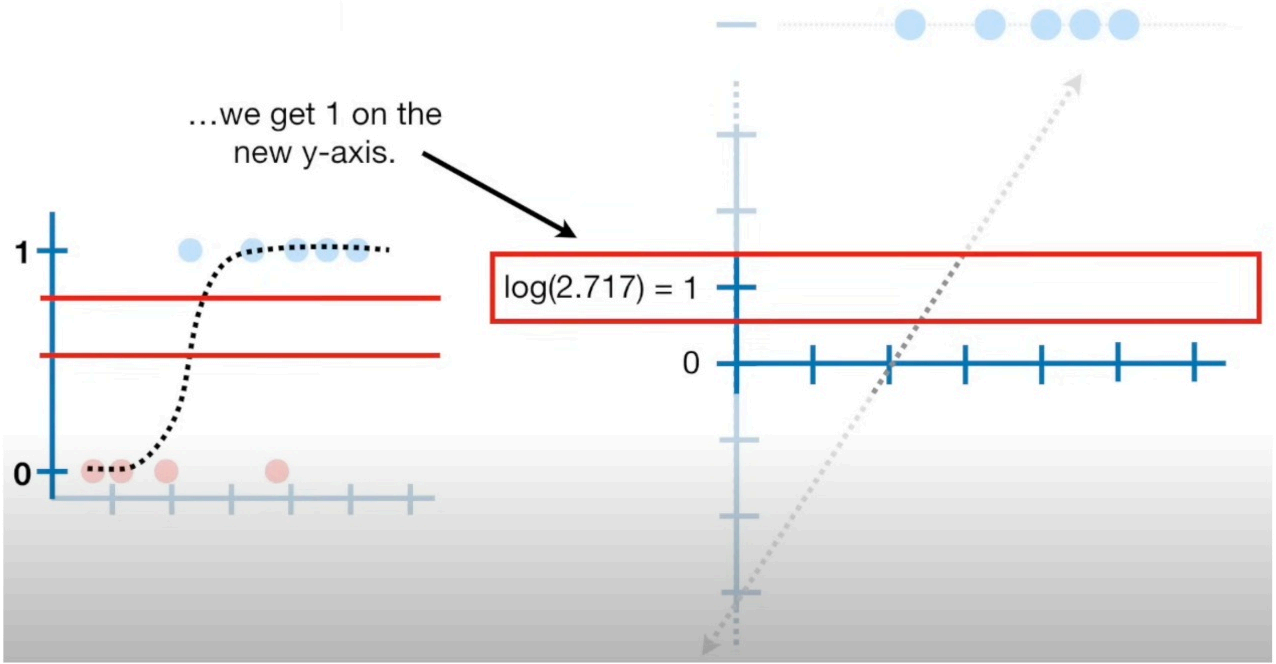
Logistic Regression - Coefficients



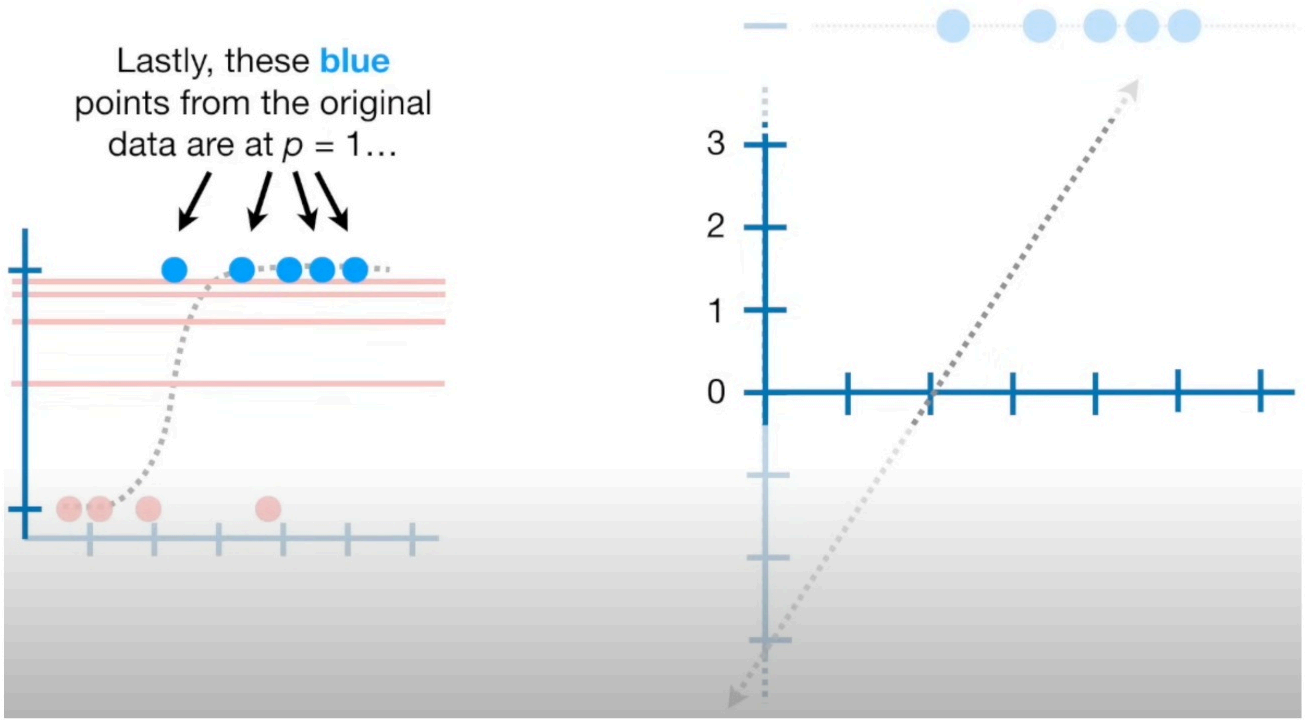
Logistic Regression - Coefficients



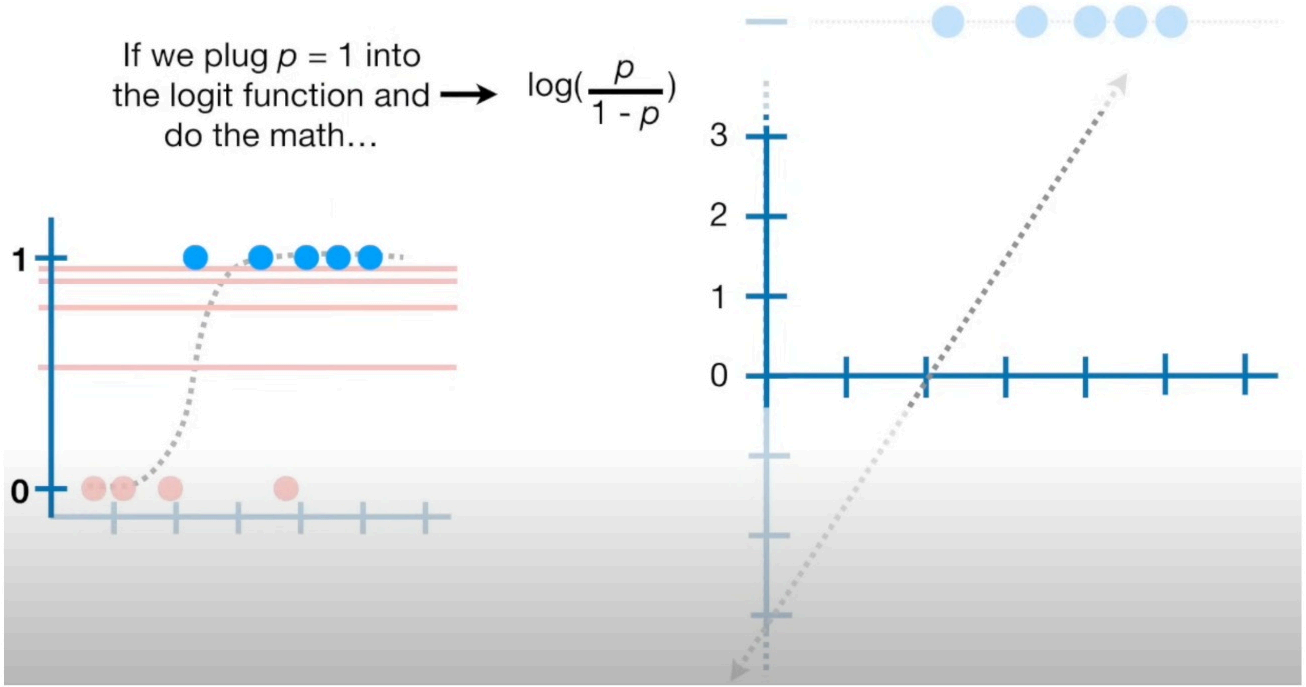
Logistic Regression - Coefficients



Logistic Regression - Coefficients



Logistic Regression - Coefficients



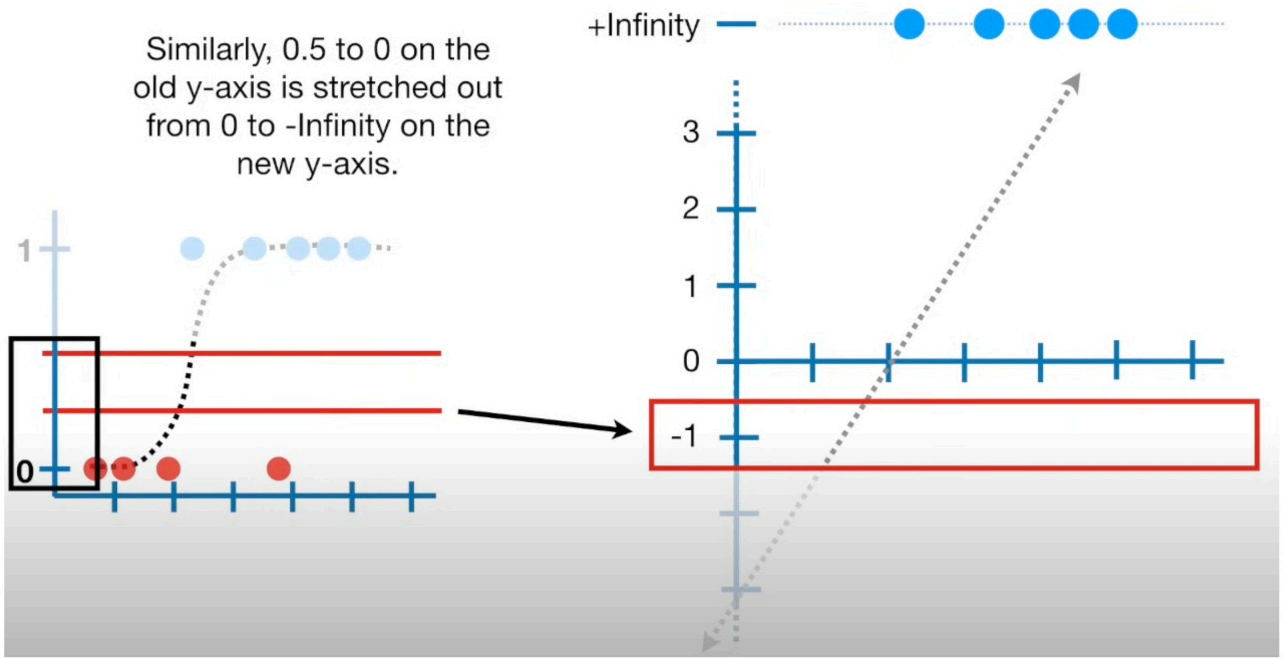
We get log(1) - log(0)

As log(0) is defined as -inf

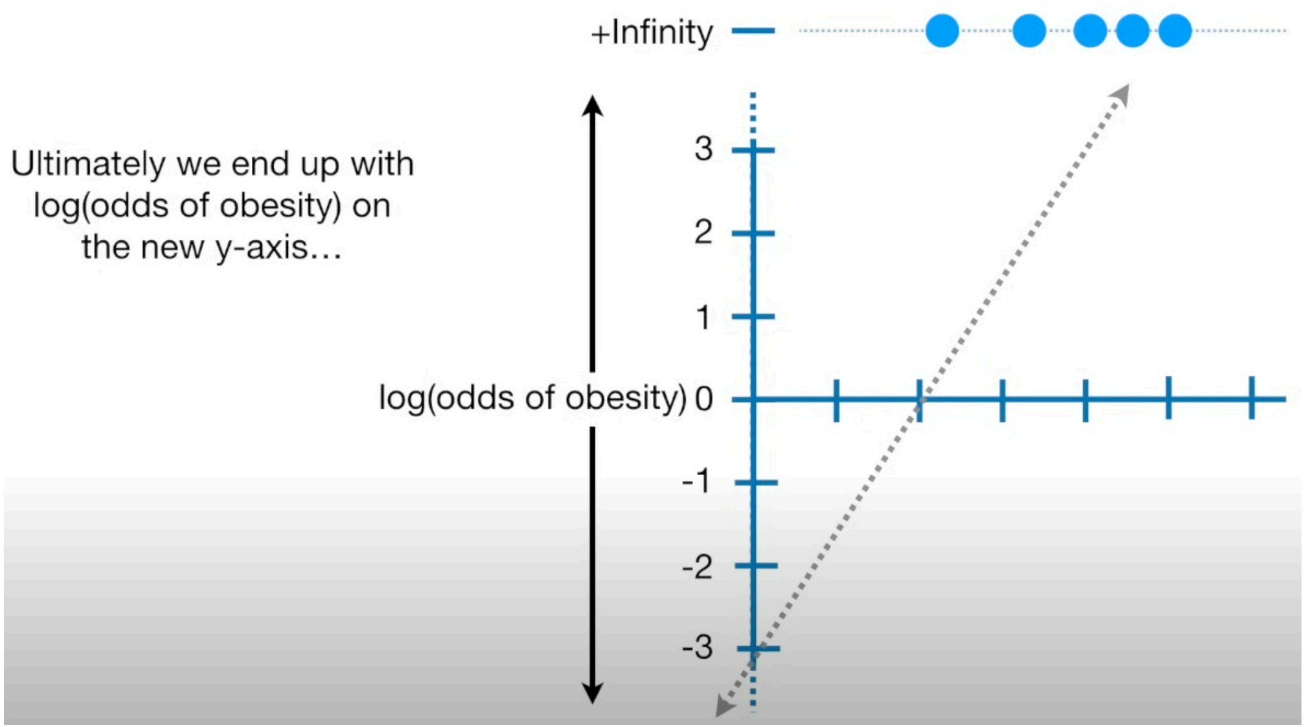
And something - (-inf) = +inf

This means the original data points labeled "obese" are now at +inf on the new Y axis

Logistic Regression - Coefficients



Logistic Regression - Coefficients



But how do we fit that line?

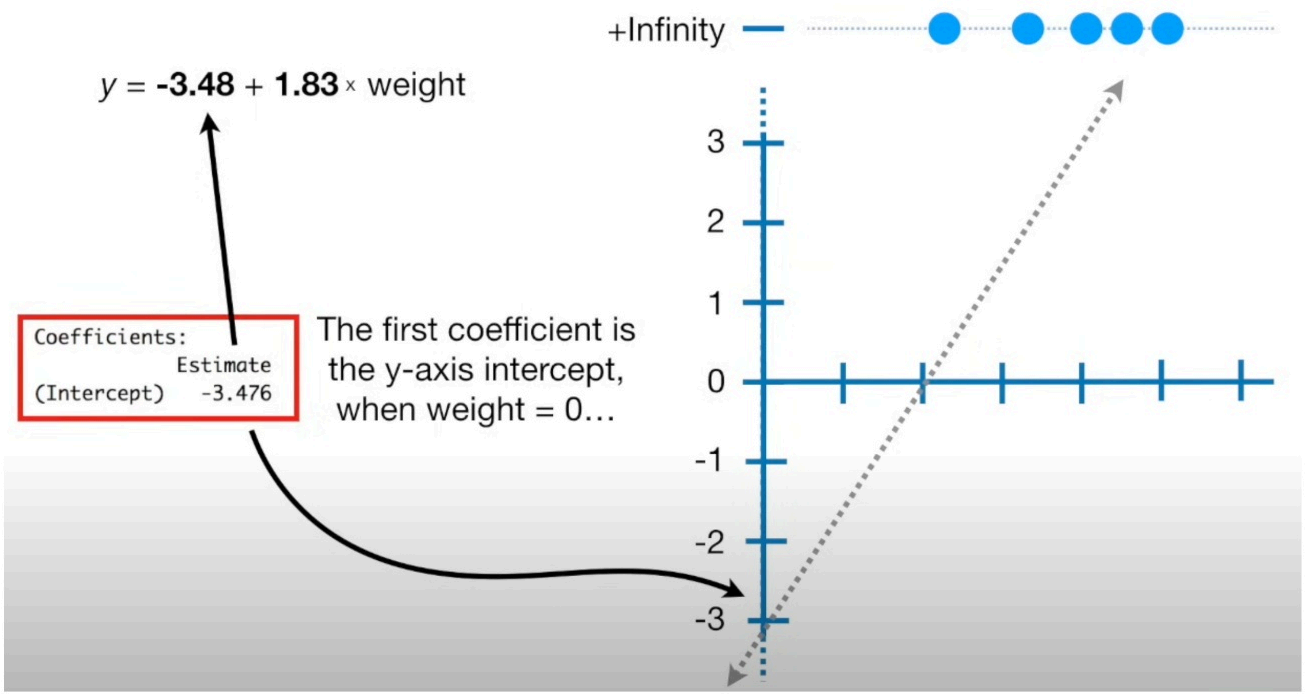
We'll see that a bit later...

Logistic Regression - Coefficients

The coefficients for the line are what you get when you do logistic regression.

```
Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)   -3.476      2.364  -1.471   0.1414
weight         1.825      1.088   1.678   0.0934
```

Logistic Regression - Coefficients

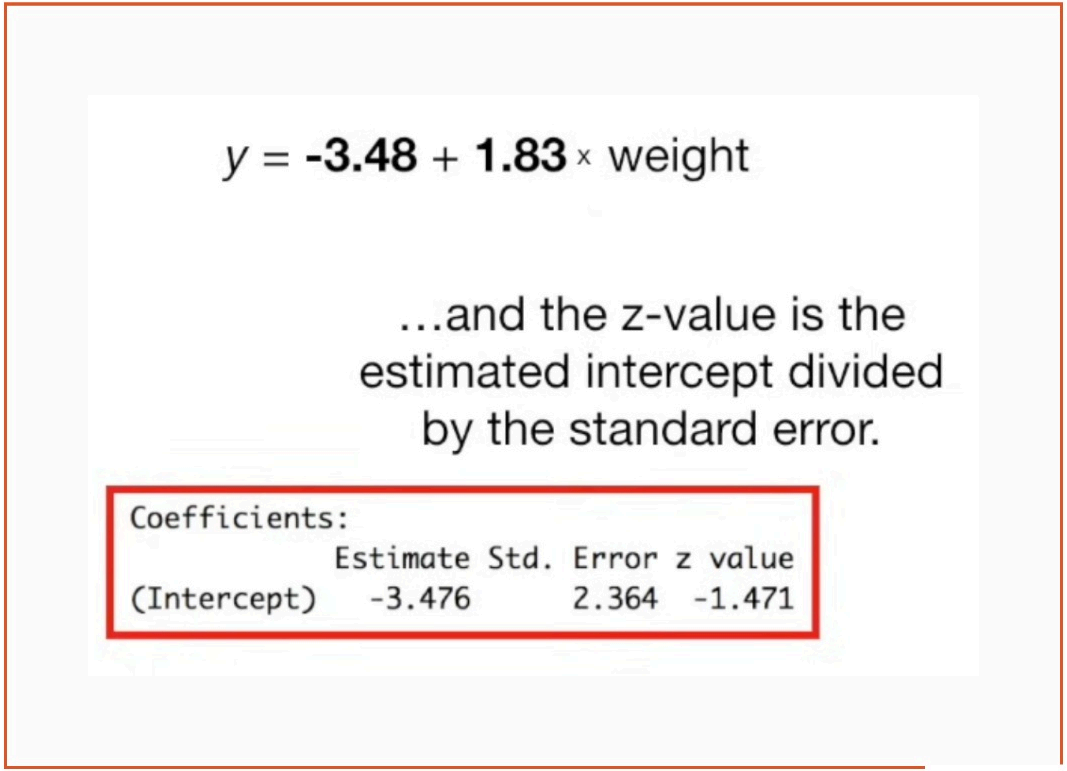


This means that when weight = 0, the log(odds of obesity) = -3.476

In other words, if you don't weight anything, the odds are against you being obese.

Logistic Regression - Coefficients

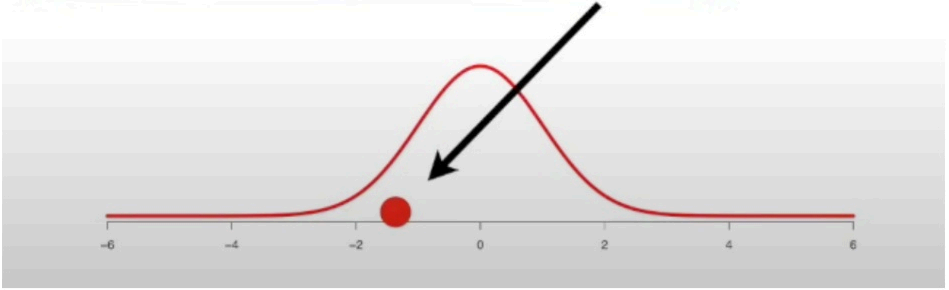
Then you have the Standard Error...



Logistic Regression - Coefficients

In other words, it's the number of standard deviations the estimated intercept is away from 0 on a standard normal curve.

```
Coefficients:
             Estimate Std. Error z value
(Intercept)    -3.476      2.364  -1.471
```

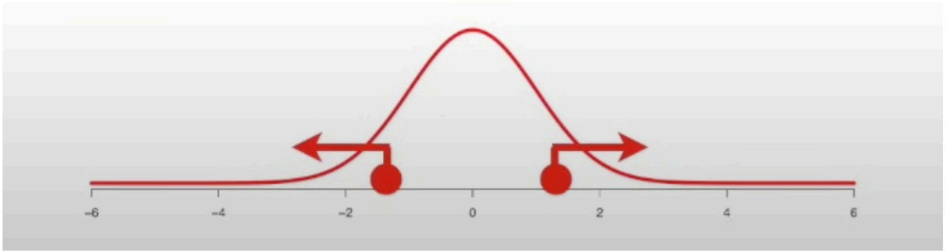


Since the estimate is less than 2 std deviations away from 0, the result is not statistically significant.

Logistic Regression - Coefficients

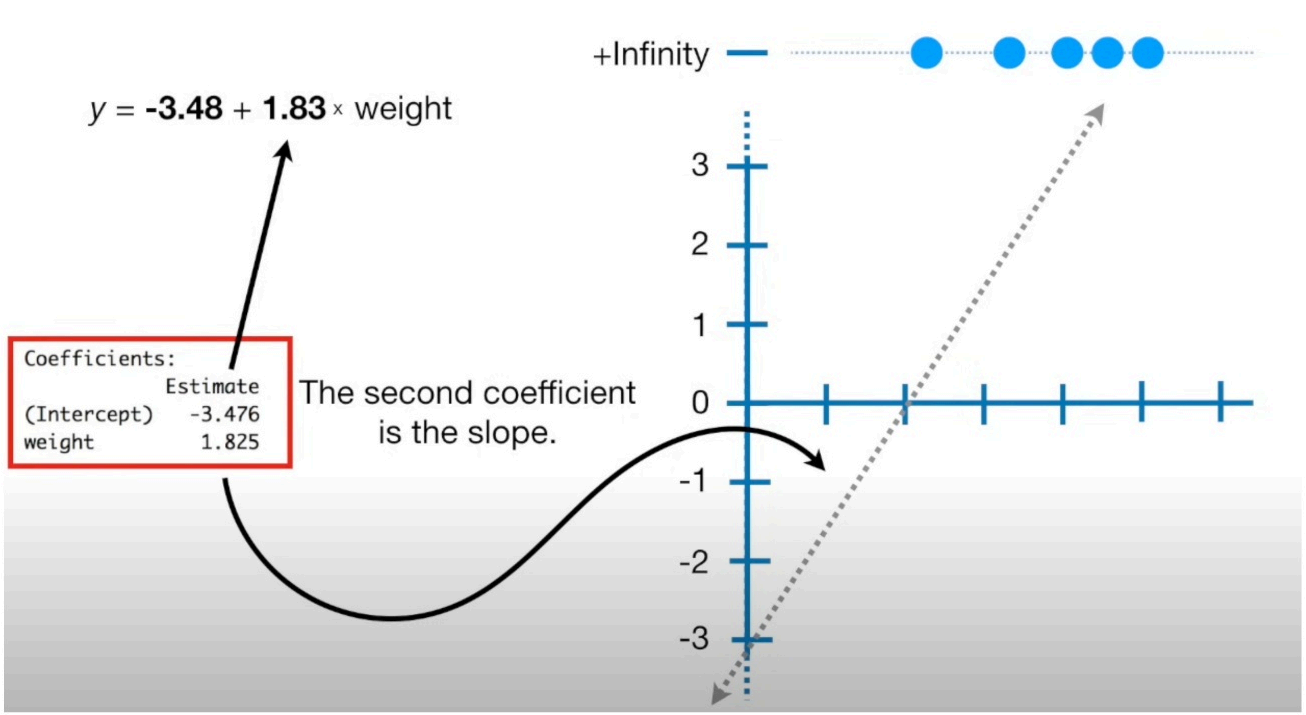
…and this is confirmed by the large p-value; the area under a standard normal curve that is further than 1.471 standard deviations away from 0.

```
Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)    -3.476      2.364  -1.471   0.1414
```



A statistically significant p-value is below 0.05

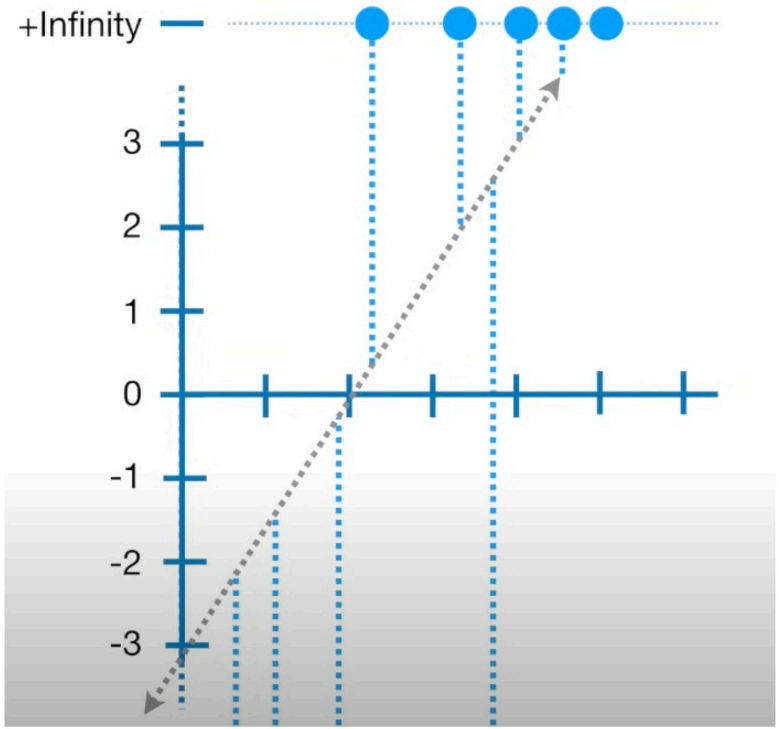
Logistic Regression - Coefficients



This means that for every one unit of weight gained, the log(odds of obesity) increases by 1.83.

Now that we know what the Coefficients mean, how do we optimize these coefficients to find the best-fitting line?

Logistic Regression - Fitting the line



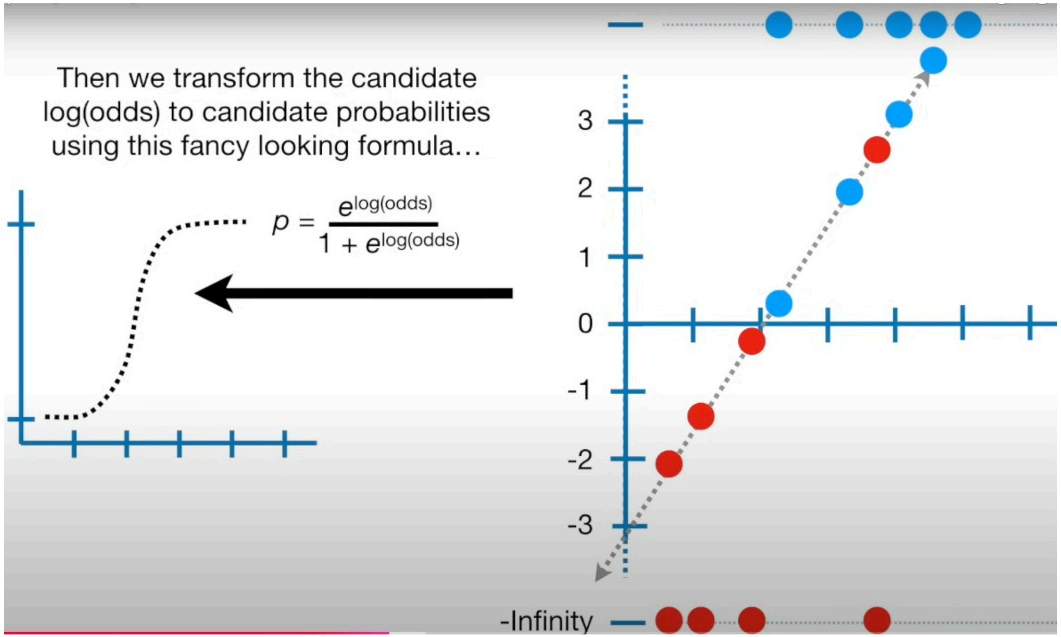
In our new y axis, we go from +inf to -inf...

...and thus we cannot use the least squares method!

We will use the Maximum Likelihood instead

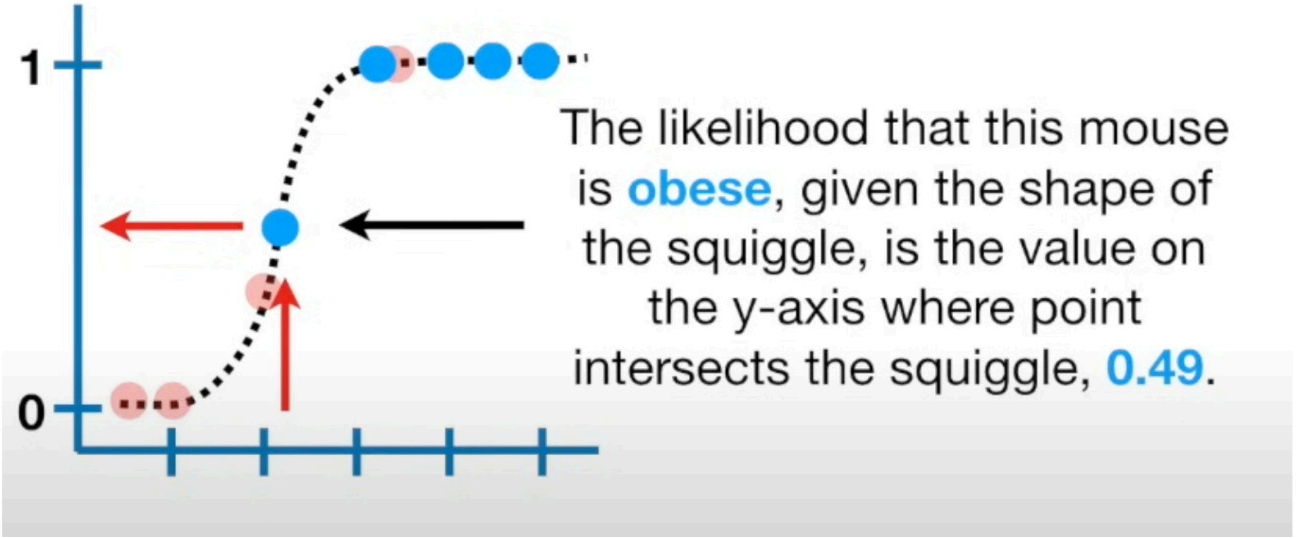
Logistic Regression - Fitting the line

First we project the data points on the line

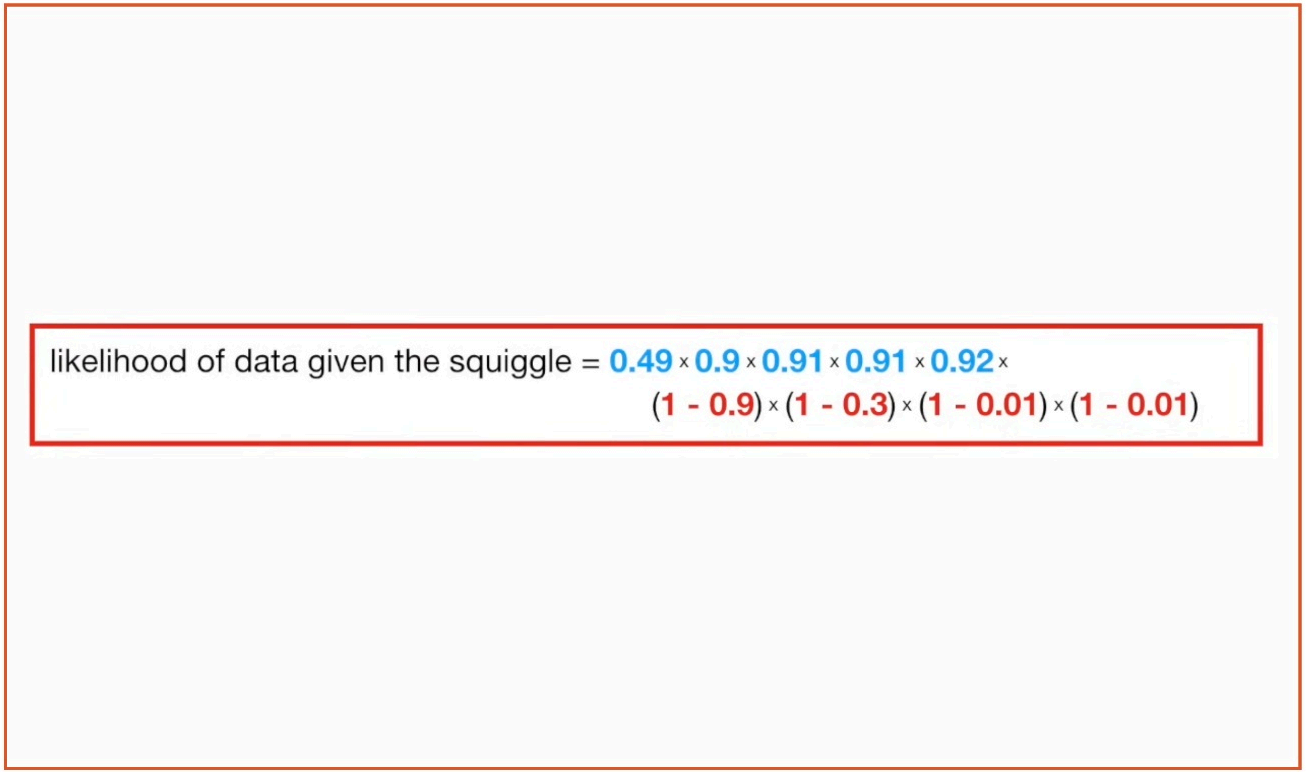


Logistic Regression - Fitting the line

Then we calculate the likelihood of each data point given the shape of the squiggle...

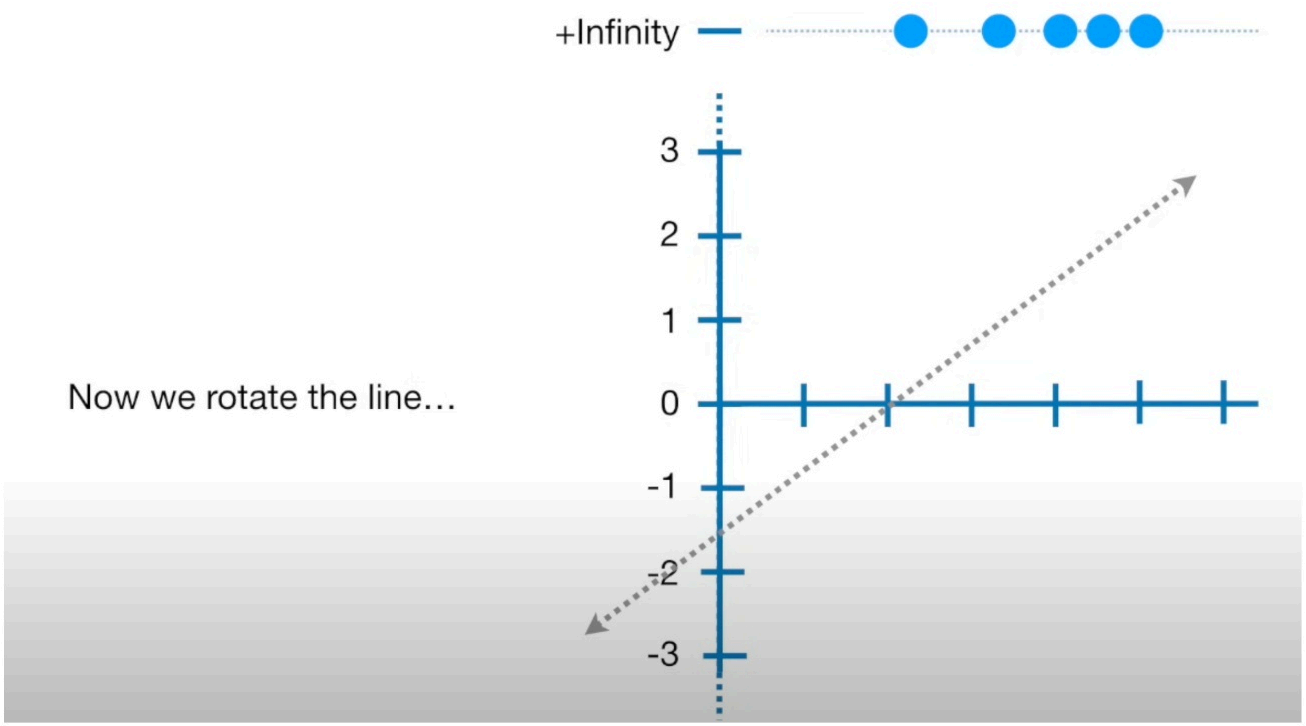


Logistic Regression - Fitting the line



...And we sum the likelihoods

Logistic Regression - Fitting the line



…and calculate the likelihoods again

…until we find the line with the maximum likelihood!

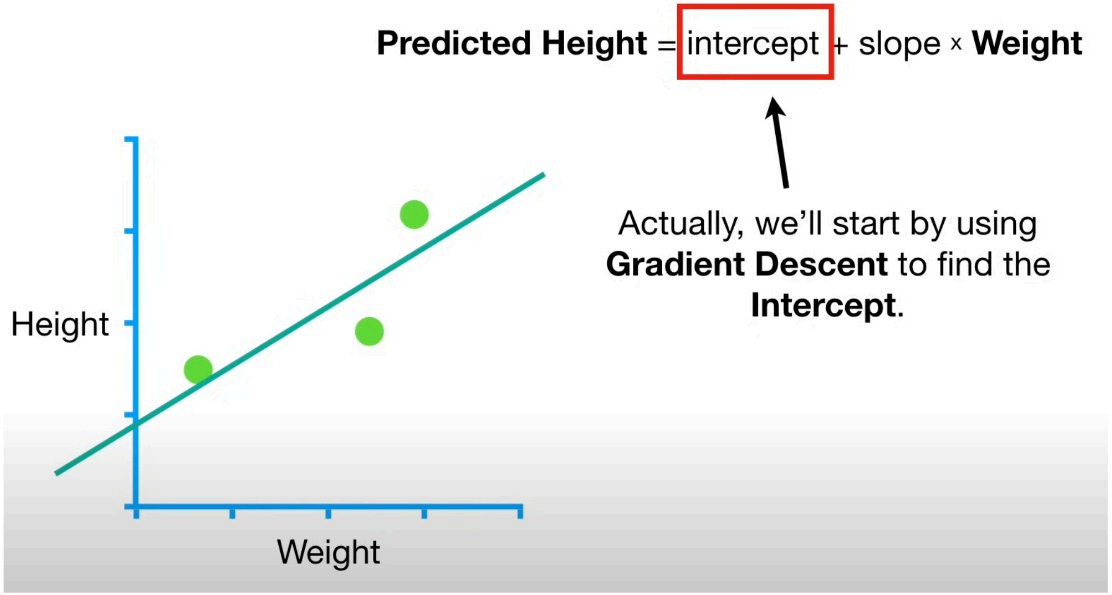
This is great!

But calculating the optimal coefficients for the curve isn't straightforward because there’s no simple formula to find them.

Gradient Descent - Intuition

Optimizing the coefficients

This is where Gradient Descent come into place

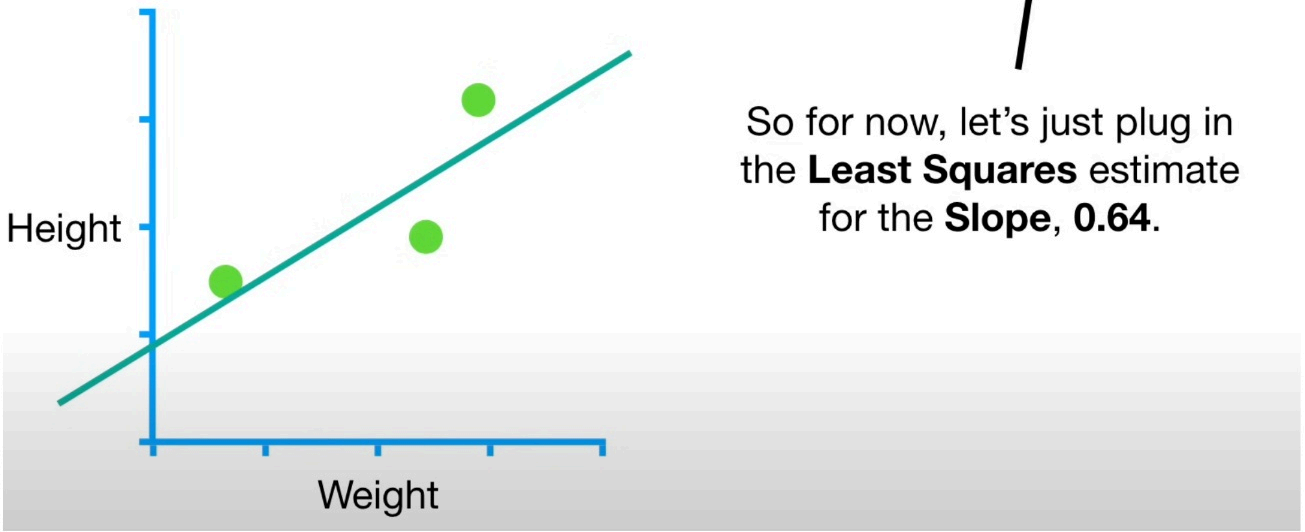


Gradient Descent - Intuition

Optimizing the coefficients

Predicted Height = intercept + 0.64 × **Weight**

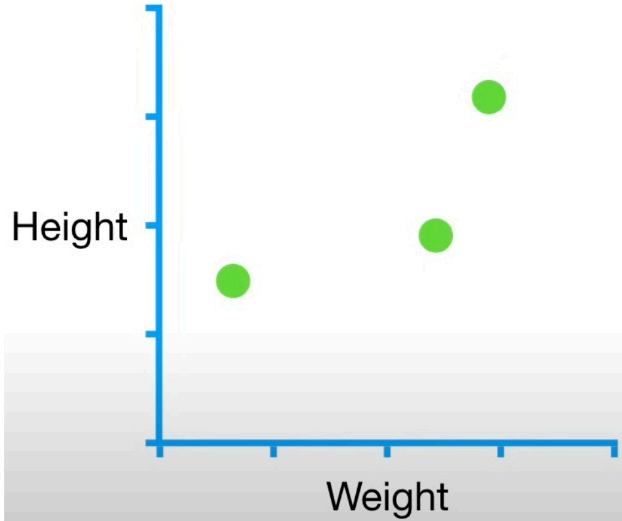
So for now, let's just plug in the **Least Squares** estimate for the **Slope**, **0.64**.



Gradient Descent - Intuition

Optimizing the coefficients

Predicted Height = intercept + 0.64 × **Weight**

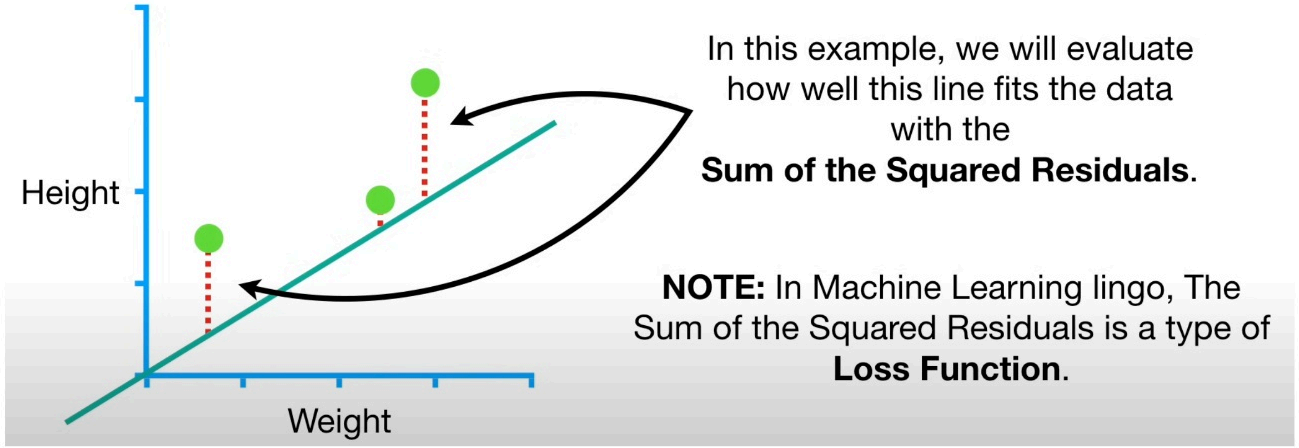


The first thing we do is pick a random value for the **Intercept**.

This is just an initial guess that gives **Gradient Descent** something to improve upon.

Gradient Descent - Intuition

Optimizing the coefficients



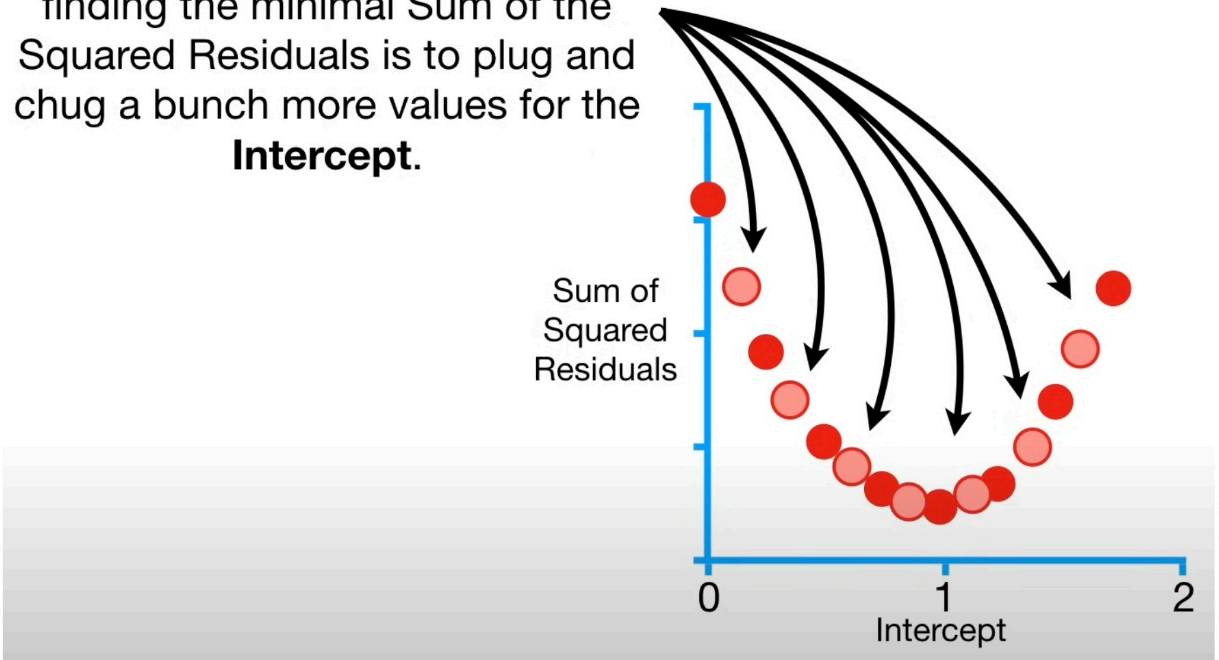
In this example, we will evaluate how well this line fits the data with the **Sum of the Squared Residuals**.

NOTE: In Machine Learning lingo, The Sum of the Squared Residuals is a type of **Loss Function**.

Gradient Descent - Intuition

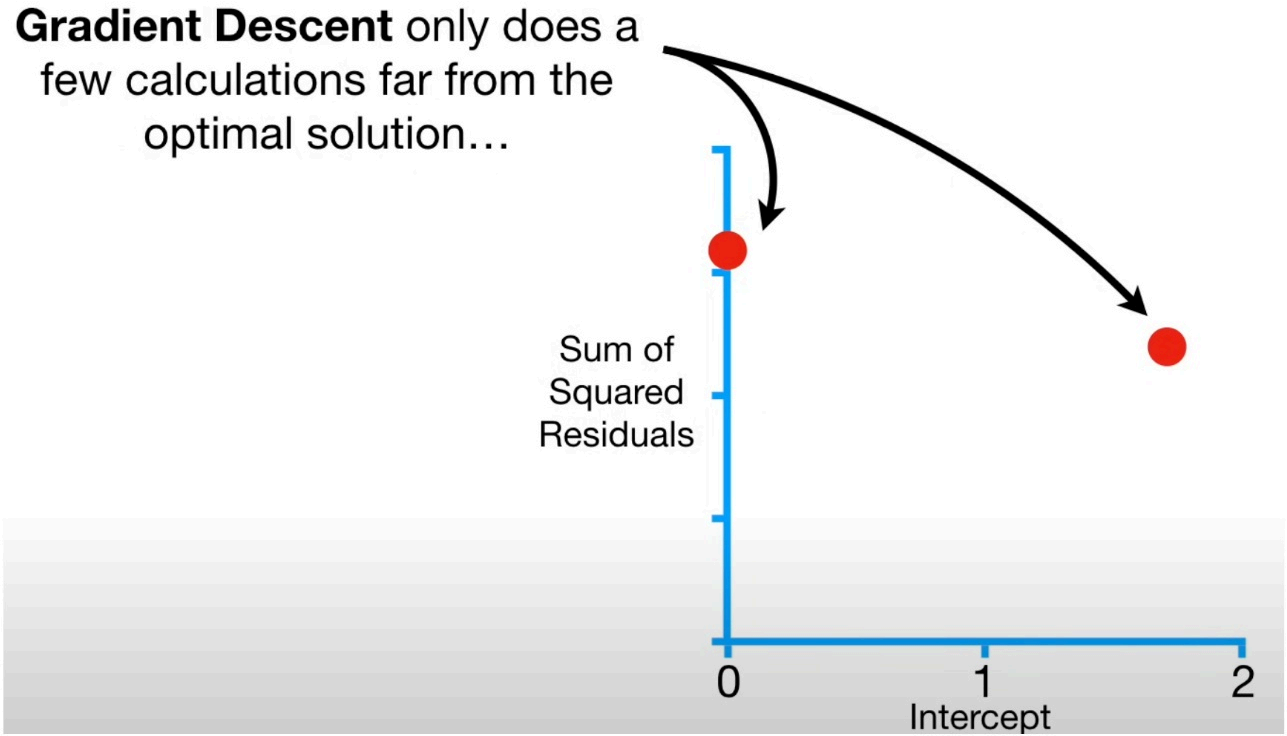
Optimizing the coefficients

A slow and painful method for finding the minimal Sum of the Squared Residuals is to plug and chug a bunch more values for the **Intercept**.



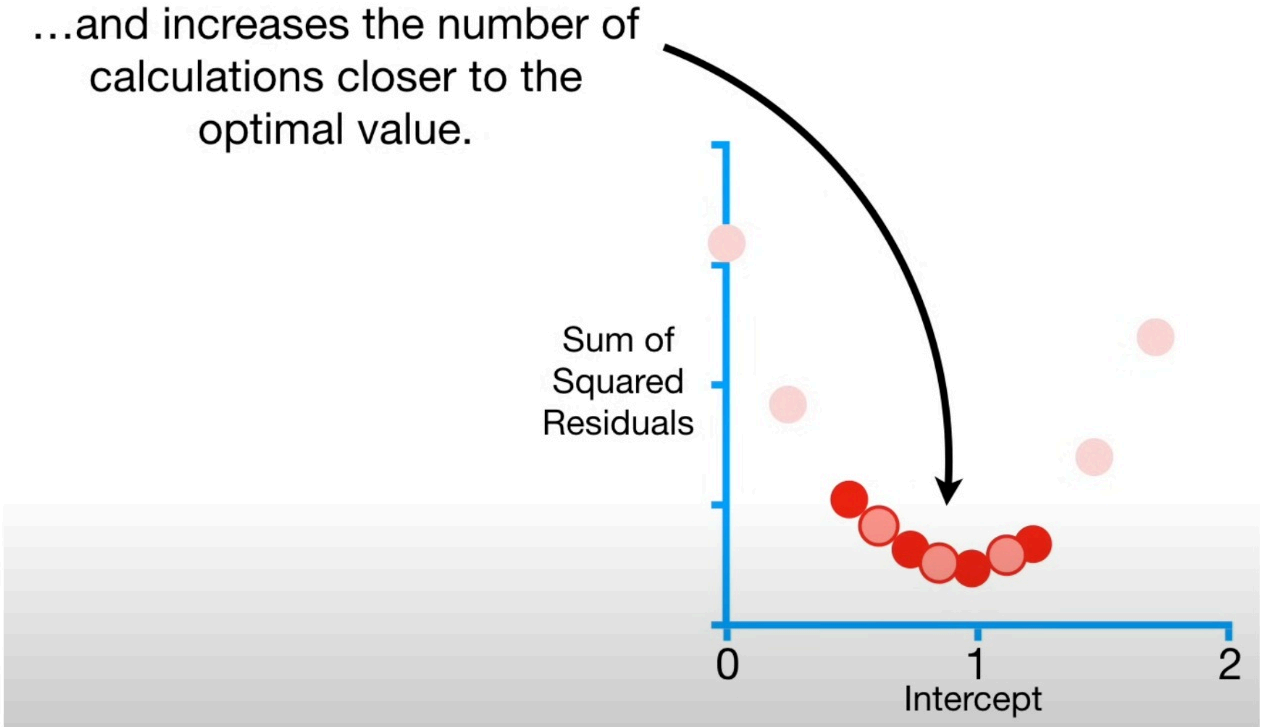
Gradient Descent - Intuition

Optimizing the coefficients



Gradient Descent - Intuition

Optimizing the coefficients



Gradient Descent identifies the optimal value by taking big steps when it is far away, and baby steps when it is close

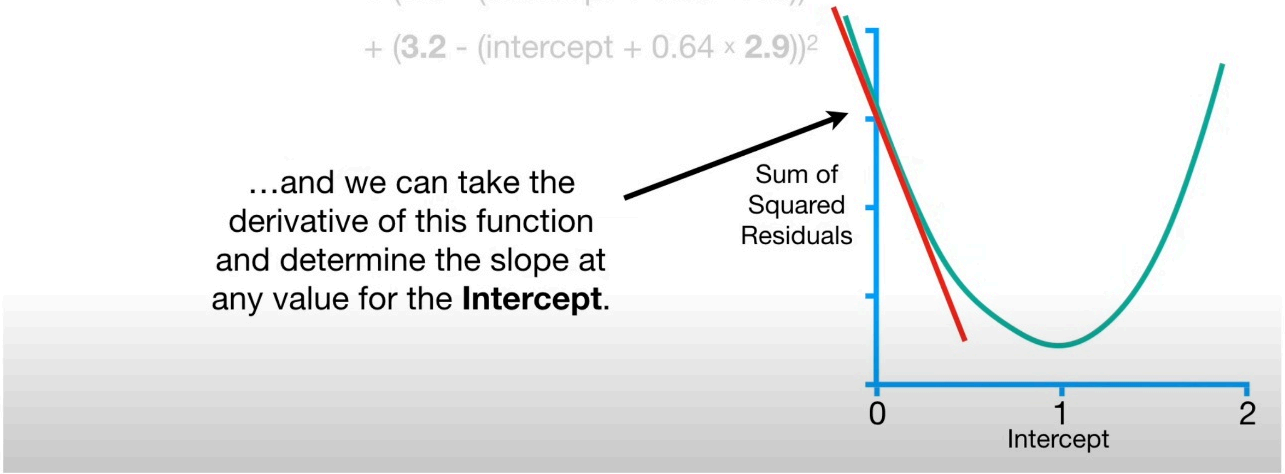
Gradient Descent - Intuition

Optimizing the coefficients

How does it work?

Once we have calculated and plotted the sum of squared residuals...

$$\text{Sum of squared residuals} = (\mathbf{1.4} - (\text{intercept} + 0.64 \times \mathbf{0.5}))^2 + (\mathbf{1.9} - (\text{intercept} + 0.64 \times \mathbf{2.3}))^2 + (\mathbf{3.2} - (\text{intercept} + 0.64 \times \mathbf{2.9}))^2$$



Gradient Descent - Intuition

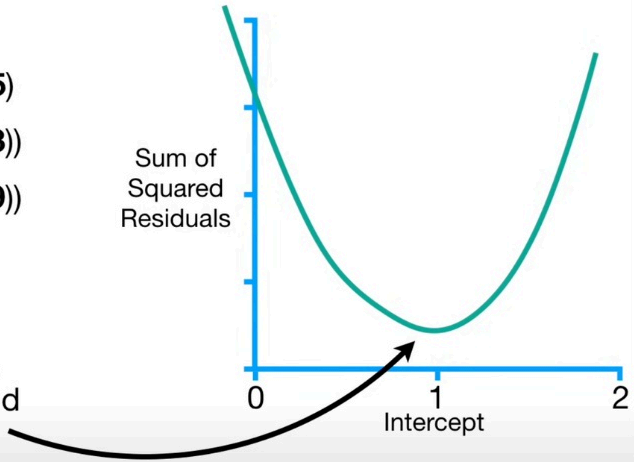
Optimizing the coefficients

$$\frac{d}{d\ intercept}\ \text{Sum of squared residuals} =$$

$$-2(\mathbf{1.4} - (\text{intercept} + 0.64 \times \mathbf{0.5})$$

$$+ -2(\mathbf{1.9} - (\text{intercept} + 0.64 \times \mathbf{2.3}))$$

$$+ -2(\mathbf{3.2} - (\text{intercept} + 0.64 \times \mathbf{2.9}))$$



Now that we have the derivative, **Gradient Descent** will use it to find where the Sum of Squared Residuals is lowest.

Gradient Descent - Intuition

Optimizing the coefficients

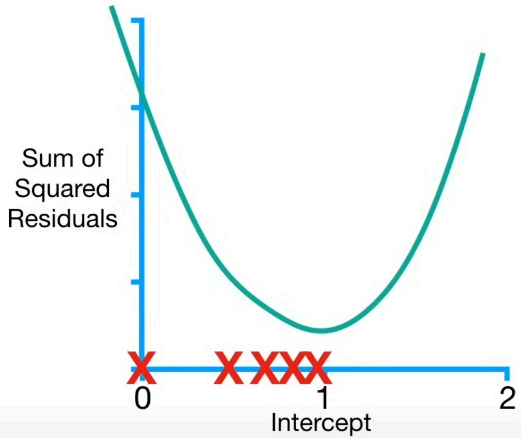
Gradient Descent finds the minimum value by taking steps from an initial guess until it reaches the best value

$$\frac{d}{d\,intercept}\text{Sum of squared residuals} =$$

$$-2(\mathbf{1.4} - (\text{intercept} + 0.64 \times \mathbf{0.5})$$

$$+ -2(\mathbf{1.9} - (\text{intercept} + 0.64 \times \mathbf{2.3}))$$

$$+ -2(\mathbf{3.2} - (\text{intercept} + 0.64 \times \mathbf{2.9}))$$

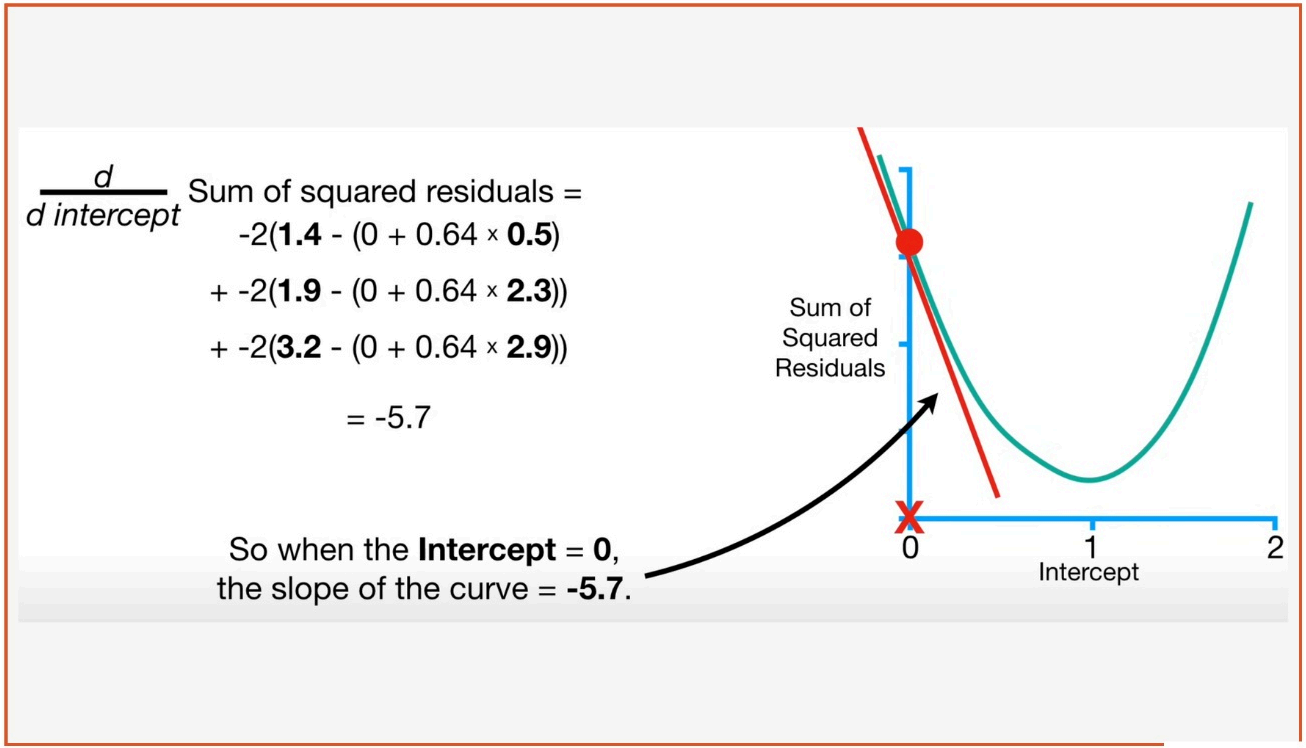


This makes **Gradient Descent** very useful when it is not possible to solve for where the derivative = **0**, and this is why **Gradient Descent** can be used in so many different situations.

Gradient Descent - Intuition

Optimizing the coefficients

Let's do Gradient Descent step by step

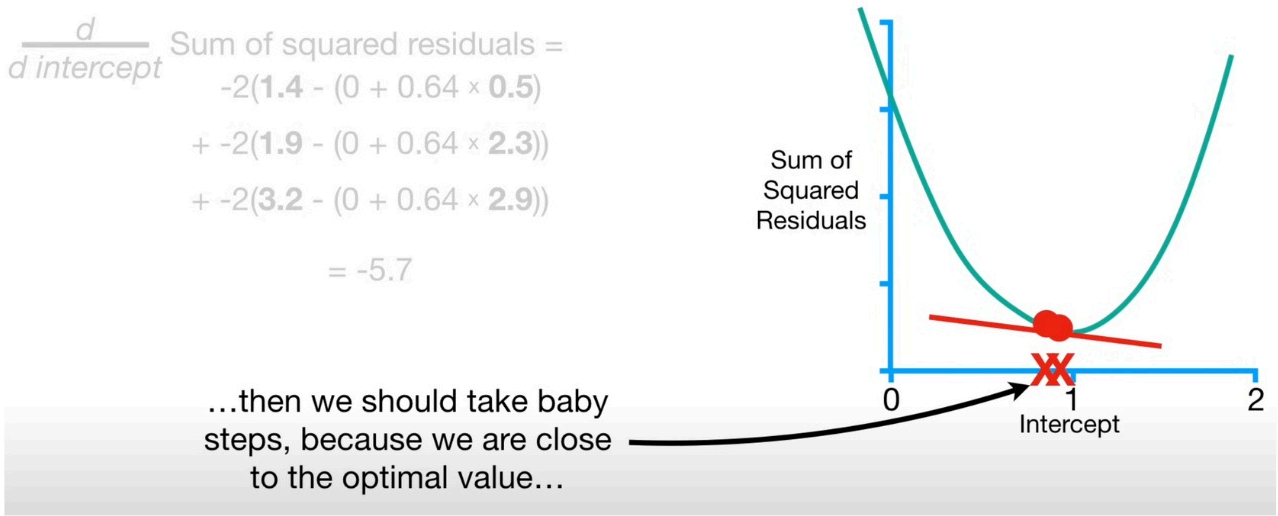


Gradient Descent - Intuition

Optimizing the coefficients

The closer we get to the optimal value, the closer the slope of the curve gets to 0.

$$\frac{d}{d\,intercept}\text{Sum of squared residuals} = -2(\mathbf{1.4} - (0 + 0.64 \times \mathbf{0.5}) + -2(\mathbf{1.9} - (0 + 0.64 \times \mathbf{2.3})) + -2(\mathbf{3.2} - (0 + 0.64 \times \mathbf{2.9})) = -5.7$$



So when we are close to 0...

Gradient Descent - Intuition

Optimizing the coefficients

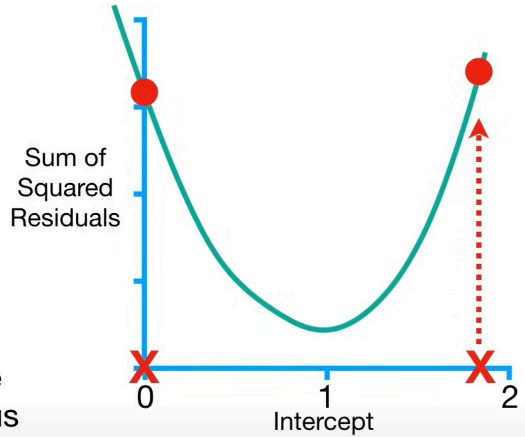
$$\frac{d}{d\ intercept}\ \text{Sum of squared residuals} =$$

$$-2(\mathbf{1.4} - (0 + 0.64 \times \mathbf{0.5}))$$

$$+ -2(\mathbf{1.9} - (0 + 0.64 \times \mathbf{2.3}))$$

$$+ -2(\mathbf{3.2} - (0 + 0.64 \times \mathbf{2.9}))$$

$$= -5.7$$



So the size of the step should be related to the slope, since it tells us if we should take a baby step or a big step, but we need to make sure the big step is not too big.

Gradient Descent - Intuition

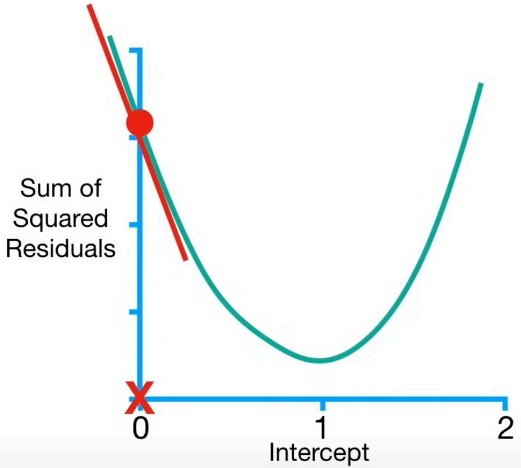
Optimizing the coefficients

Gradient Descent determines the size of the step by multiplying the slope...

$$\frac{d}{d\,intercept}\text{Sum of squared residuals} = -2(\mathbf{1.4} - (0 + 0.64 \times \mathbf{0.5})) + -2(\mathbf{1.9} - (0 + 0.64 \times \mathbf{2.3})) + -2(\mathbf{3.2} - (0 + 0.64 \times \mathbf{2.9})) = -5.7$$

Step Size = -5.7 × 0.1

…by a small number called **The Learning Rate**.



Gradient Descent - Intuition

Optimizing the coefficients

$$\frac{d}{d\,intercept}$$ Sum of squared residuals =

-2(**1.4** - (0 + 0.64 × **0.5**)

+ -2(**1.9** - (0 + 0.64 × **2.3**))

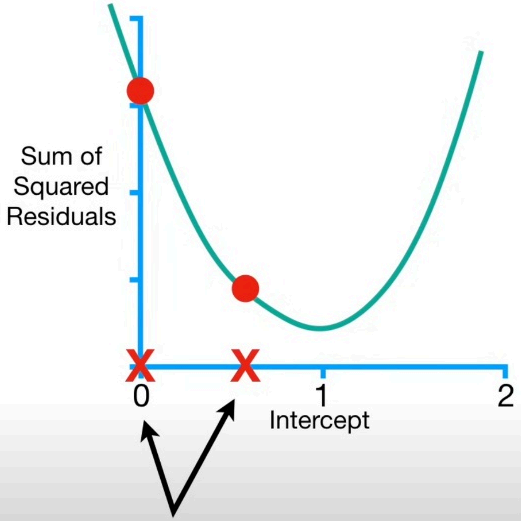
+ -2(**3.2** - (0 + 0.64 × **2.9**))

= -5.7

Step Size = -5.7 × 0.1 = -0.57

New Intercept = 0 - (-0.57) = **0.57**

BAM!!!



In one big step, we moved much closer to the optimal value for the **Intercept**.

Gradient Descent - Intuition

Optimizing the coefficients

$$\frac{d}{d\ intercept}$$ Sum of squared residuals =

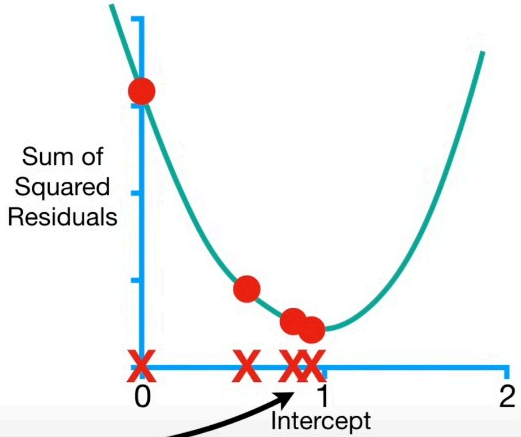
-2(**1.4** - (0.8 + 0.64 × **0.5**)

+ -2(**1.9** - (0.8 + 0.64 × **2.3**))

+ -2(**3.2** - (0.8 + 0.64 × **2.9**))

= **-0.9**

Step Size = -0.9 × 0.1 = **-0.09**



New Intercept = 0.8 - (-0.09) = **0.89**

…and the **New Intercept** = **0.89**

Each step gets smaller the closer we get to the optimal value.

We stop when the step size is smaller than 0.001.

Gradient Descent - Intuition

Optimizing the coefficients

To find the optimal values for both intercept and slope, we do the same process, but with both derivatives in respect to intercept and to slope.

NOTE: When you have two or more derivatives of the same function, they are called a **Gradient**.

Gradient Descent - Intuition

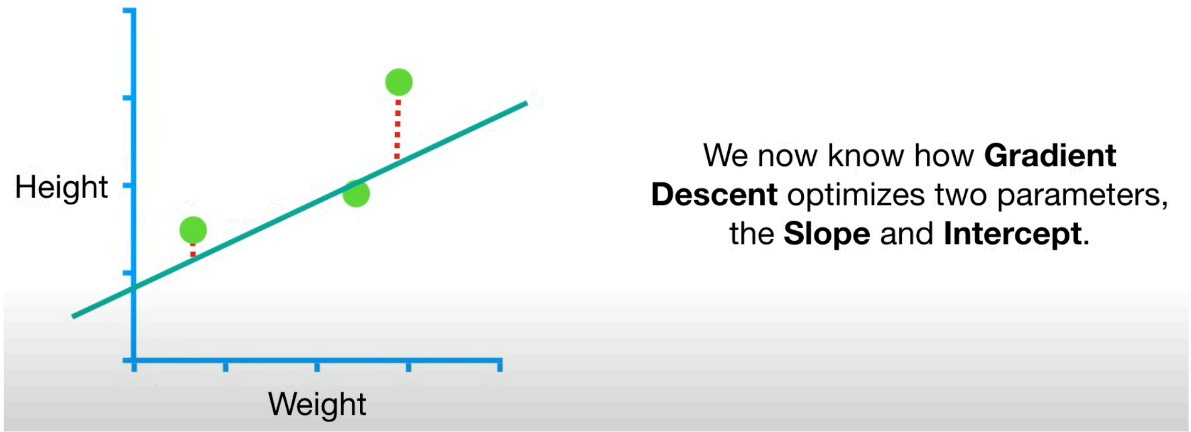
Optimizing the coefficients

We will use this **Gradient** to **descend** to lowest point in the **Loss Function**, which, in this case, is the Sum of the Squared Residuals…

And this is why this method is called **Gradient Descent**!

Gradient Descent - Intuition

Optimizing the coefficients



We now know how **Gradient Descent** optimizes two parameters, the **Slope** and **Intercept**.

If we had more parameters, we would take more derivatives, and everything else stays the same!

Gradient Descent - Intuition

Optimizing the coefficients

Step 1: Take the derivative of the **Loss Function** for each parameter in it. In fancy Machine Learning Lingo, take the **Gradient** of the **Loss Function**.

Step 2: Pick random values for the parameters.

Step 3: Plug the parameter values into the derivatives (ahem, the **Gradient**).

Step 4: Calculate the Step Sizes: **Step Size** = **Slope** x **Learning Rate**

Step 5: Calculate the New Parameters:

New Parameter = **Old Parameter** - **Step Size**

Go back to step 3 and repeat until step is very small OR you have reached the maximum number of steps

This is it!

Now we have seen 2 different methods to predict outcomes - Linear Regression and Logistic Regression.

But how do we know how good our guesses are?

Evaluation metrics

Confusion Matrix, Sensitivity, Specificity, ROC, AUC

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
...	...	...	...	...

Imagine we have a dataset of clinical measurements...

Evaluation metrics

Confusion Matrix, Sensitivity, Specificity, ROC, AUC

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
...	...	...	...	...

... and we want to apply a ML algorithm to predict whether or not someone will develop heart disease

To do this, we could rely on Logistic Regression, but also many others models like Random Forests, K-means, so how do we find the best model for this?

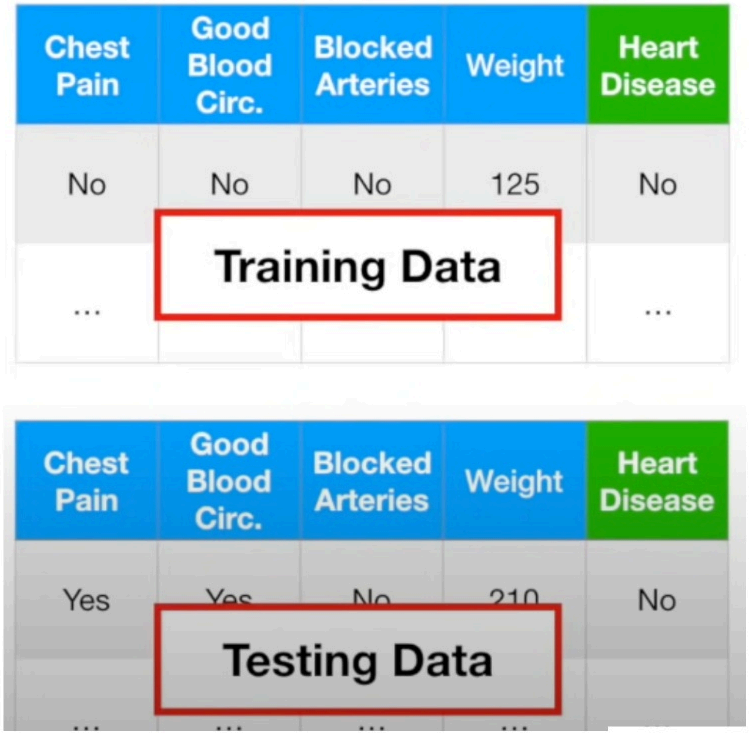
Evaluation metrics

Confusion Matrix, Sensitivity, Specificity, ROC, AUC

We start by dividing our data into a training dataset and a testing dataset

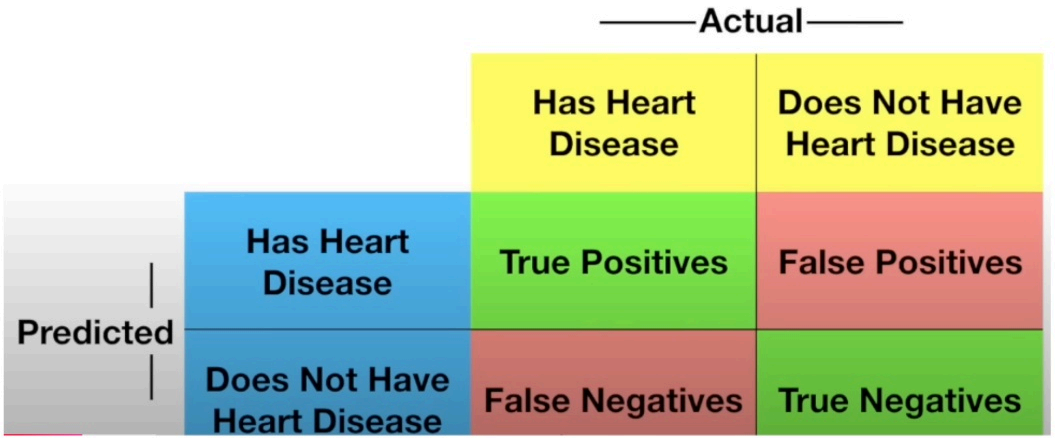
Then we train all our methods on the training dataset...

...and test each method on the testing dataset



Evaluation metrics

Confusion Matrix, Sensitivity, Specificity, ROC, AUC

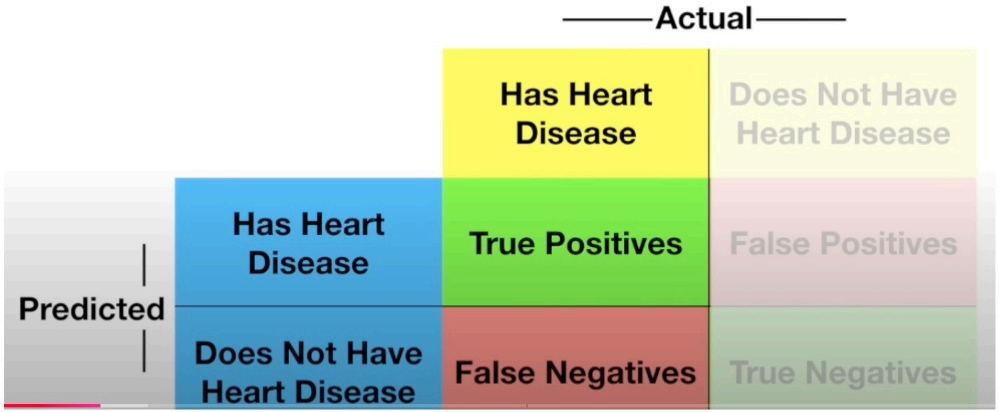


We summarize our results into a Confusion Matrix

And now we can calculate useful metrics!

Evaluation metrics

Confusion Matrix, Sensitivity, Specificity, ROC, AUC

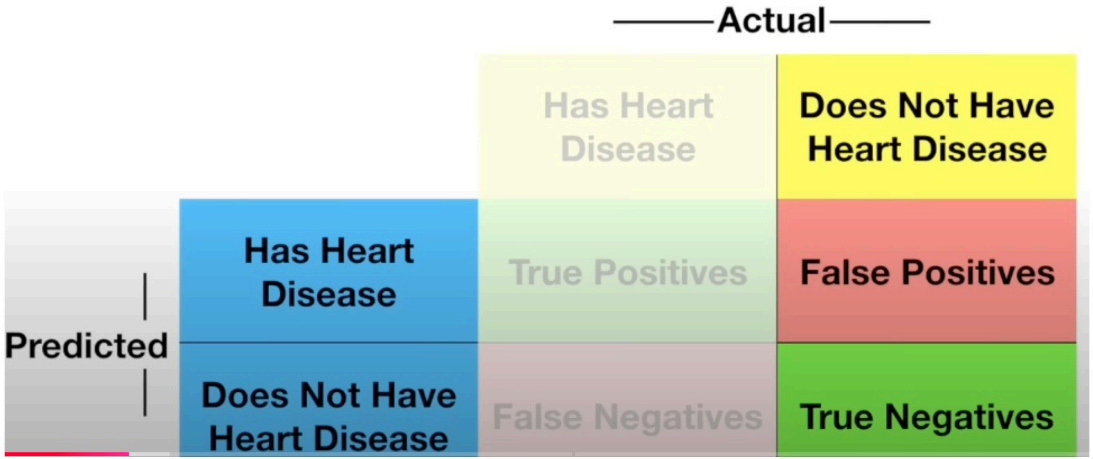


Sensitivity is the % of patients with heart disease who were correctly identified

Sensitivity = True Positives/(True Positives + False Negatives)

Evaluation metrics

Confusion Matrix, Sensitivity, Specificity, ROC, AUC



Specificity is the % of patients without heart disease who where correctly identified

Specificity = False positives/(False Positives + True Negatives)

Evaluation metrics

Confusion Matrix, Sensitivity, Specificity, ROC, AUC

Let's start by calculating **Sensitivity** for this **Logistic Regression**.

	Has Heart Disease	Does Not Have Heart Disease
Has Heart Disease	139	20
Does Not Have Heart Disease	32	112

Sensitivity = 139/171 = 0.81

81% of patients with heart disease where correctly identified with this model.

Evaluation metrics

Confusion Matrix, Sensitivity, Specificity, ROC, AUC

Now let's calculate the **Specificity**…

	Has Heart Disease	Does Not Have Heart Disease
Has Heart Disease	139	20
Does Not Have Heart Disease	32	112

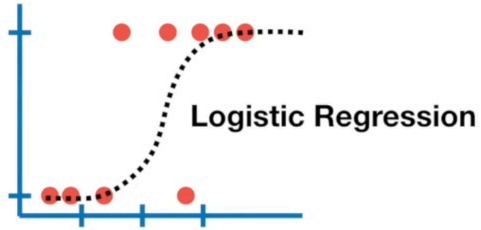
Specificity = 20/132 = 0.85

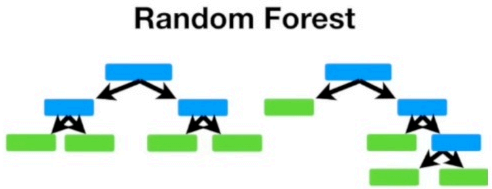
85% of patients without heart disease were correctly identified with this model.

Evaluation metrics

Confusion Matrix, Sensitivity, Specificity, ROC, AUC

Now let's compare these values...





Sensitivity = 0.81

Specificity = 0.85

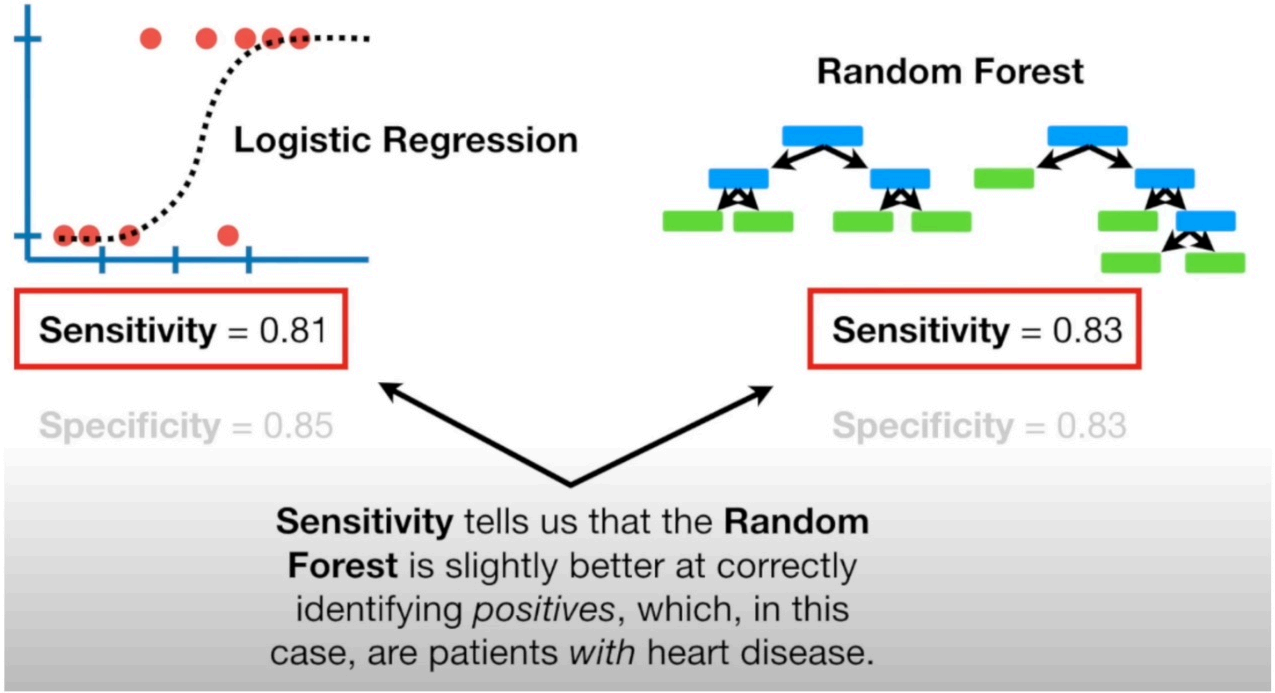
Sensitivity = 0.83

Specificity = 0.83

…to the values we calculated for the **Random Forest**.

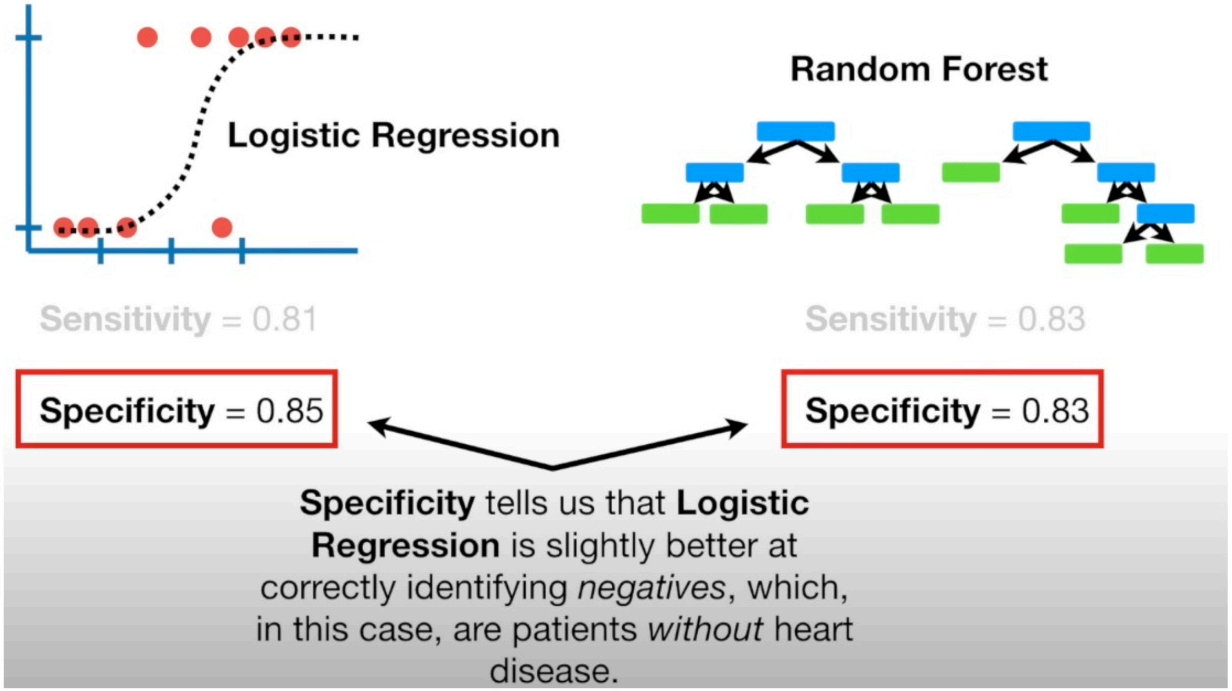
Evaluation metrics

Confusion Matrix, Sensitivity, Specificity, ROC, AUC



Evaluation metrics

Confusion Matrix, Sensitivity, Specificity, ROC, AUC

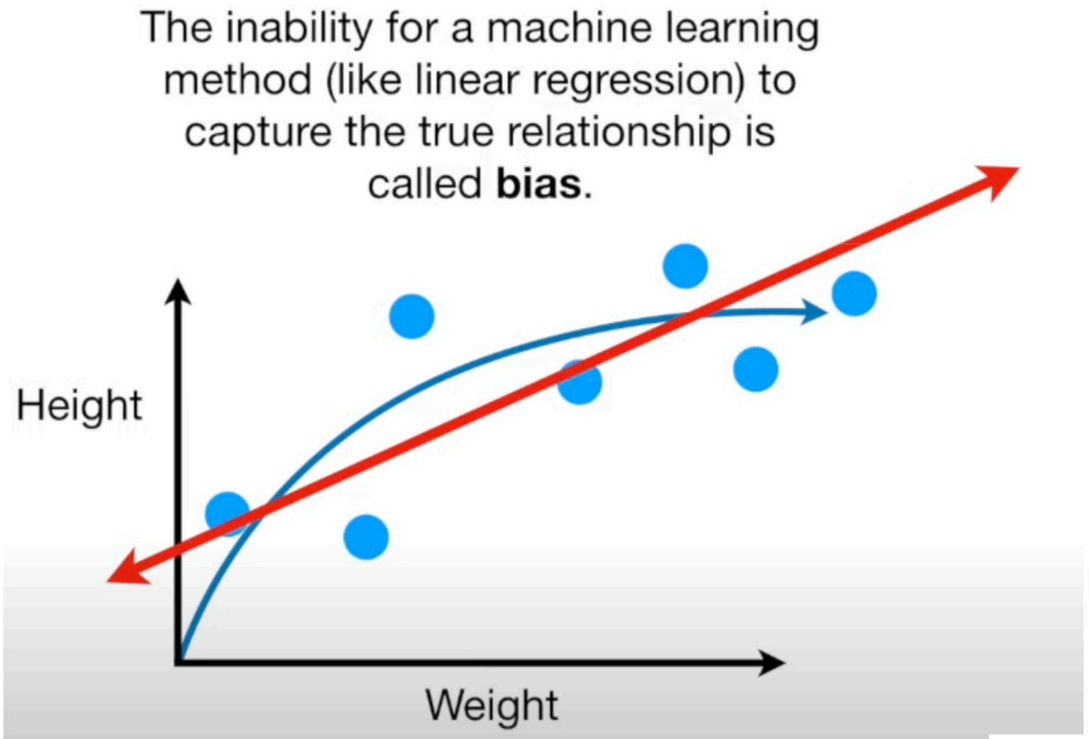


We will choose the Random Forest if correctly identifying people with heart disease is more important than correctly identifying people without heart disease.

Evaluation metrics

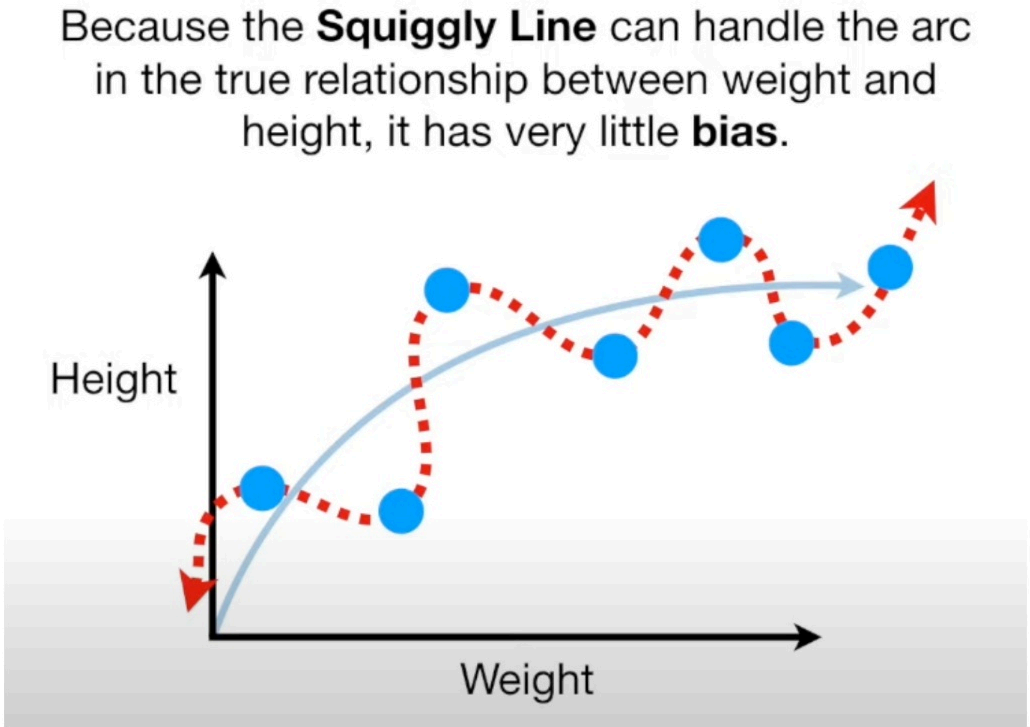
Confusion Matrix, Sensitivity, Specificity, ROC, AUC

Because linear regression produces a straight line, the model will never capture a perfect relationship



Evaluation metrics

Confusion Matrix, Sensitivity, Specificity, ROC, AUC

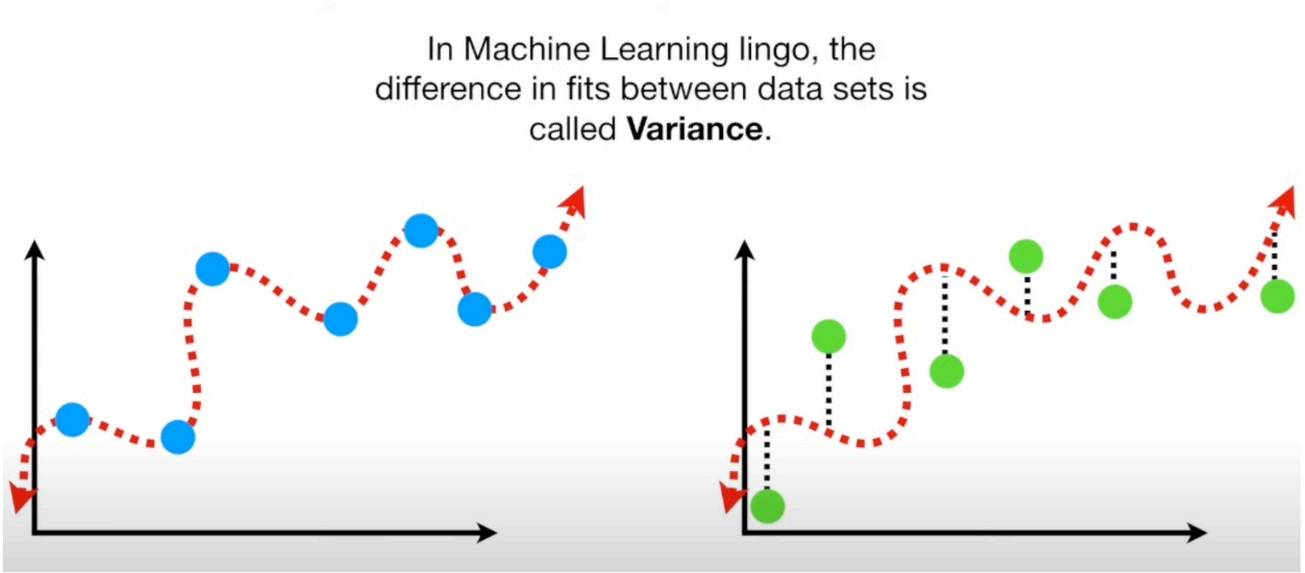


It fits perfectly this dataset...

...but it may not fit very well on other datasets

Evaluation metrics

Confusion Matrix, Sensitivity, Specificity, ROC, AUC

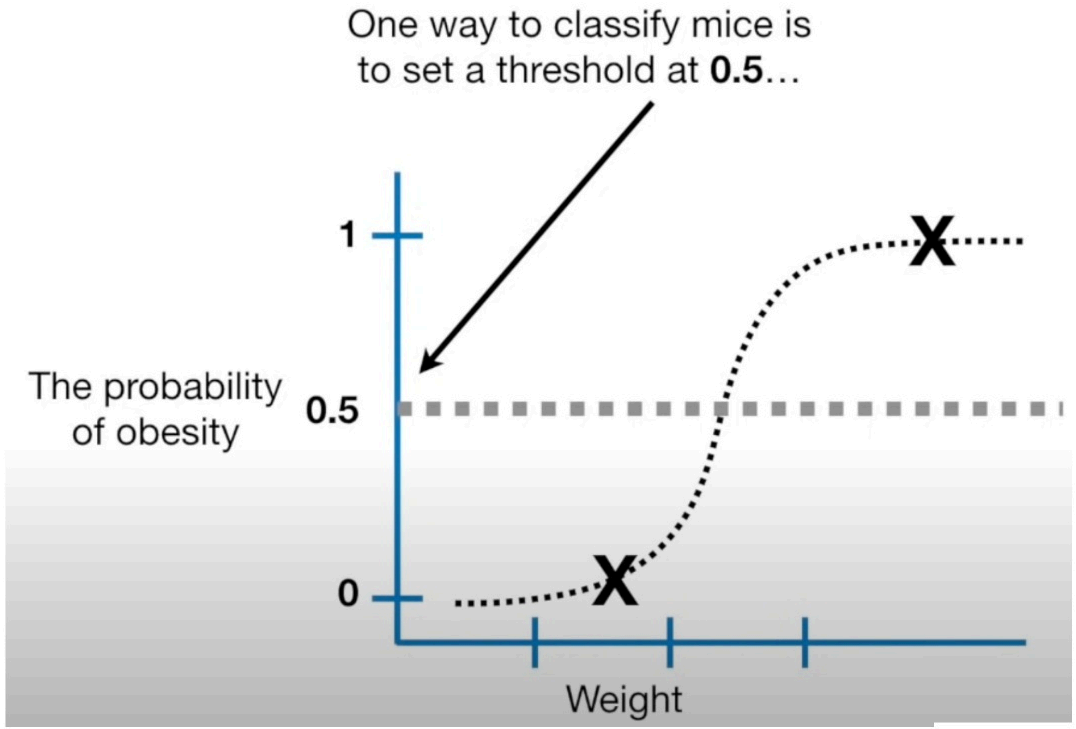


When a model has low bias but high variance, we call it **Overfitting**

Evaluation metrics

Confusion Matrix, Sensitivity, Specificity, ROC, AUC

Now let's talk about Logistic Regression again



Evaluation metrics

Confusion Matrix, Sensitivity, Specificity, ROC, AUC

But how do we know this is the right threshold?

Evaluation metrics

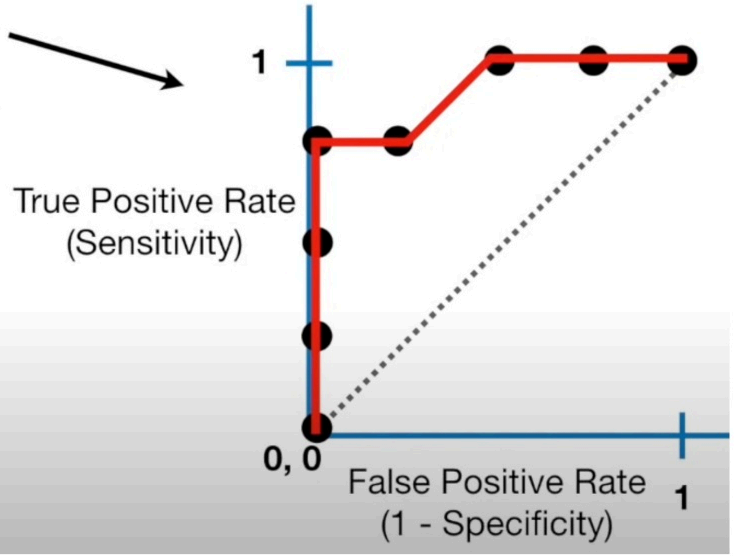
Confusion Matrix, Sensitivity, Specificity, ROC, AUC

If we made 1 confusion matrix per threshold, it would result in a lot of confusion matrices...

Evaluation metrics

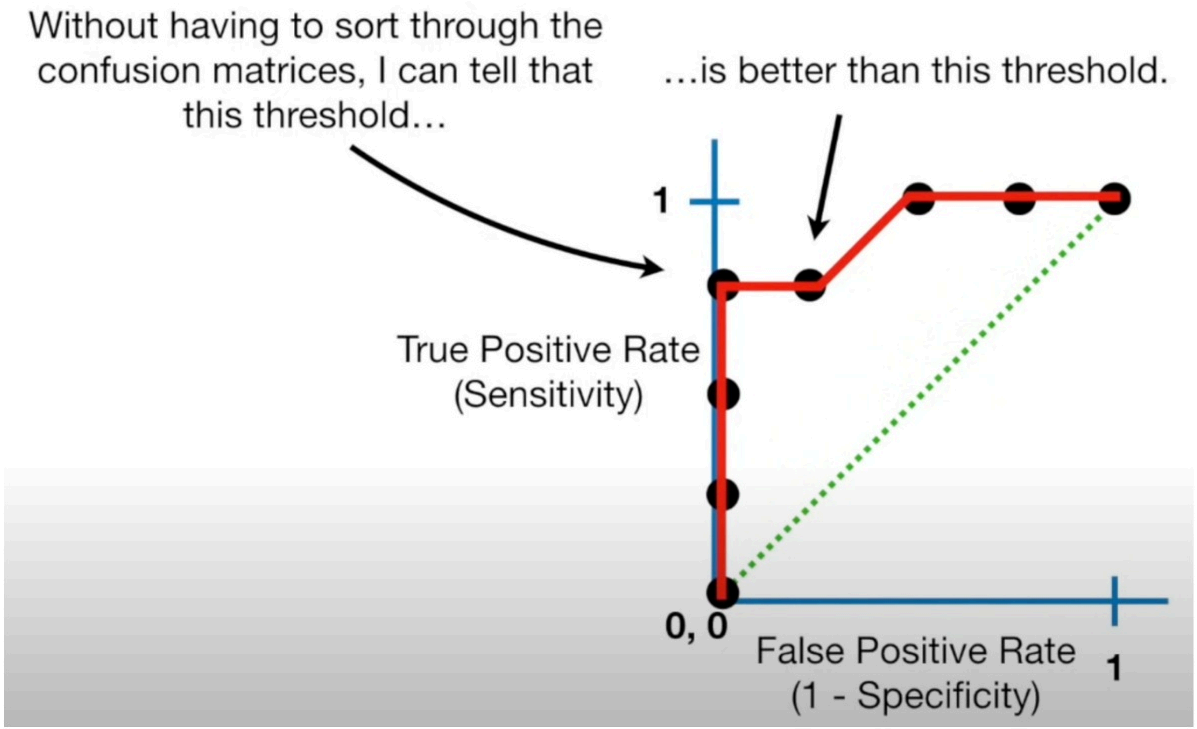
Confusion Matrix, Sensitivity, Specificity, ROC, AUC

So instead of being overwhelmed with confusion matrices, **Receiver Operator Characteristic** (**ROC**) graphs provide a simple way to summarize all of the information.



Evaluation metrics

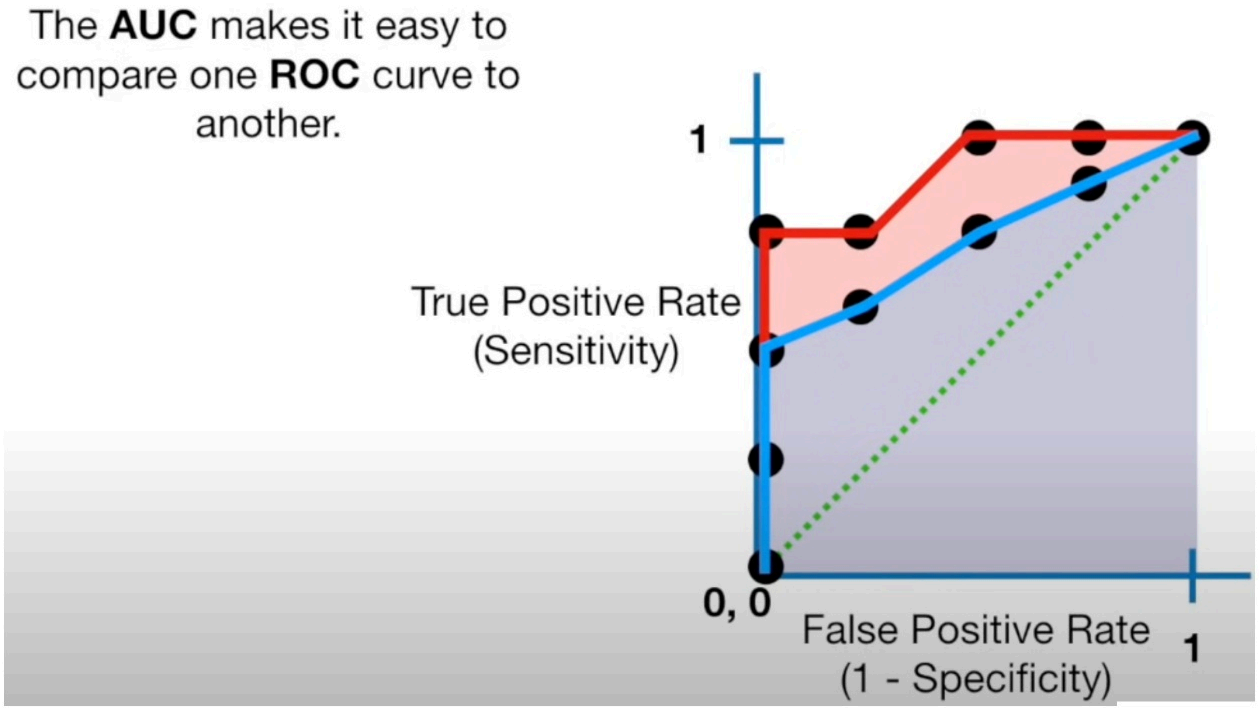
Confusion Matrix, Sensitivity, Specificity, ROC, AUC



Evaluation metrics

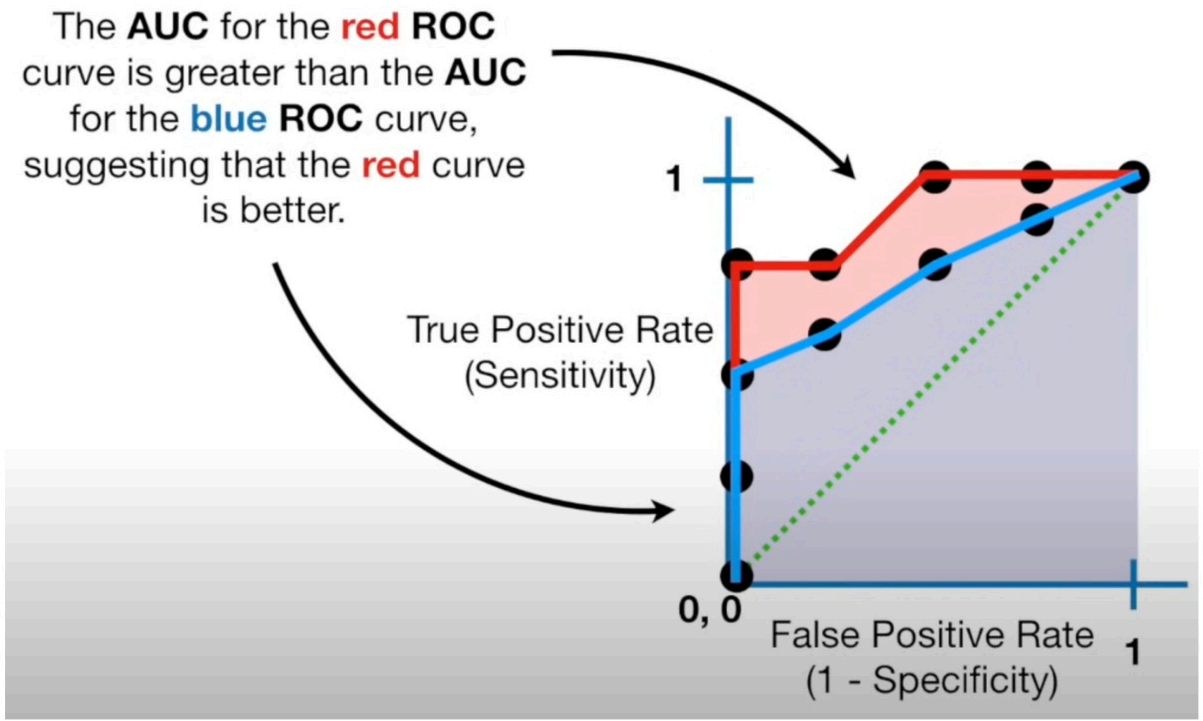
Confusion Matrix, Sensitivity, Specificity, ROC, AUC

Now that we know what ROC is, let's talk about the Area Under the Curve



Evaluation metrics

Confusion Matrix, Sensitivity, Specificity, ROC, AUC



So if the red ROC curve represented Logistic Regression and the blue one Random Forest, you would choose Logistic Regression.