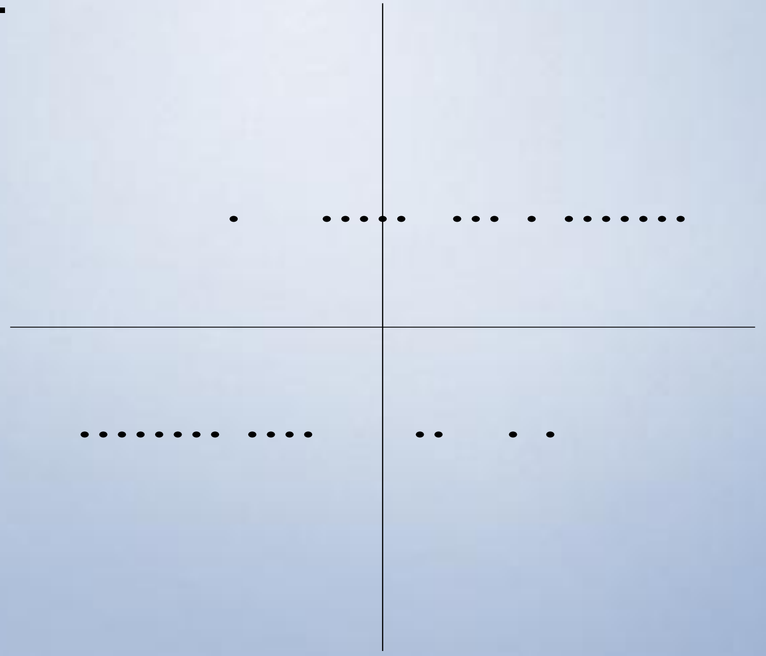
Regression

LOGISTIC REGRESSION

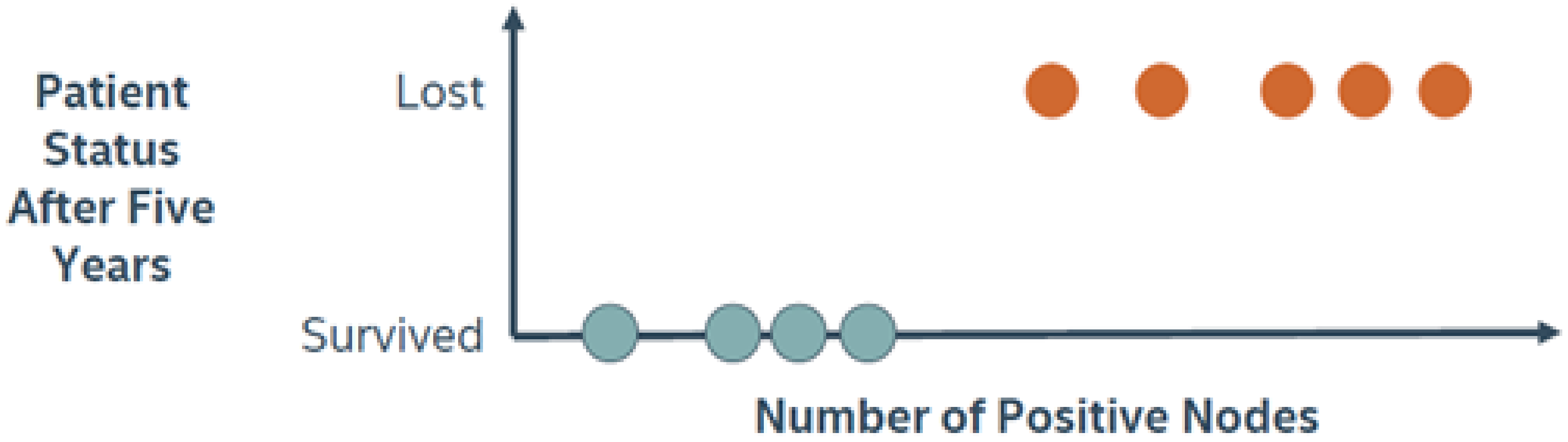
A Problem with Linear Regression

However, transforming the independent variables does not remedy all of the potential problems. What if we have a non-normally distributed dependent variable? The following example depicts the problem of fitting a regular regression line to a non-normal dependent variable).

Suppose you have a binary outcome variable. The problem of having a non-continuous dependent variable becomes apparent when you create a scatterplot of the relationship. Here, we see that it is very difficult to decipher a relationship among these variables.

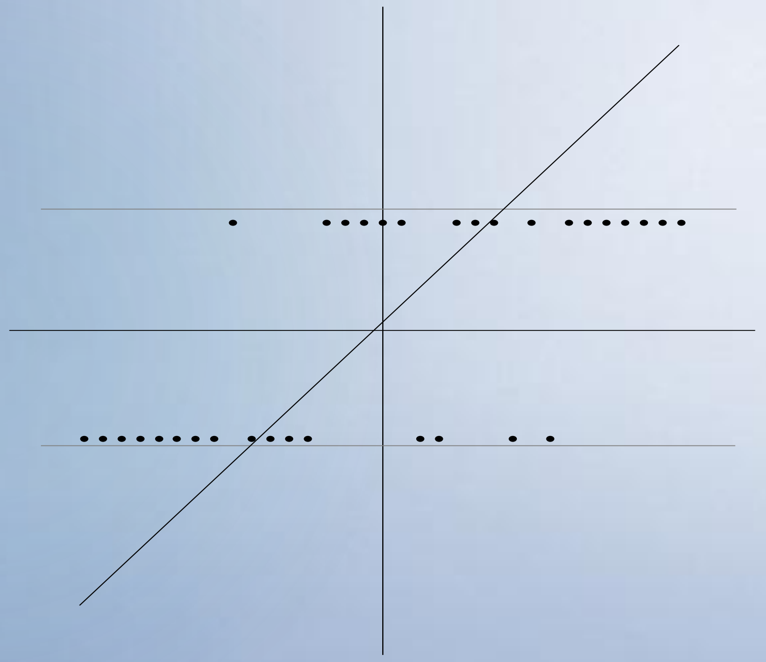


INTRODUCTION TO LOGISTIC REGRESSION



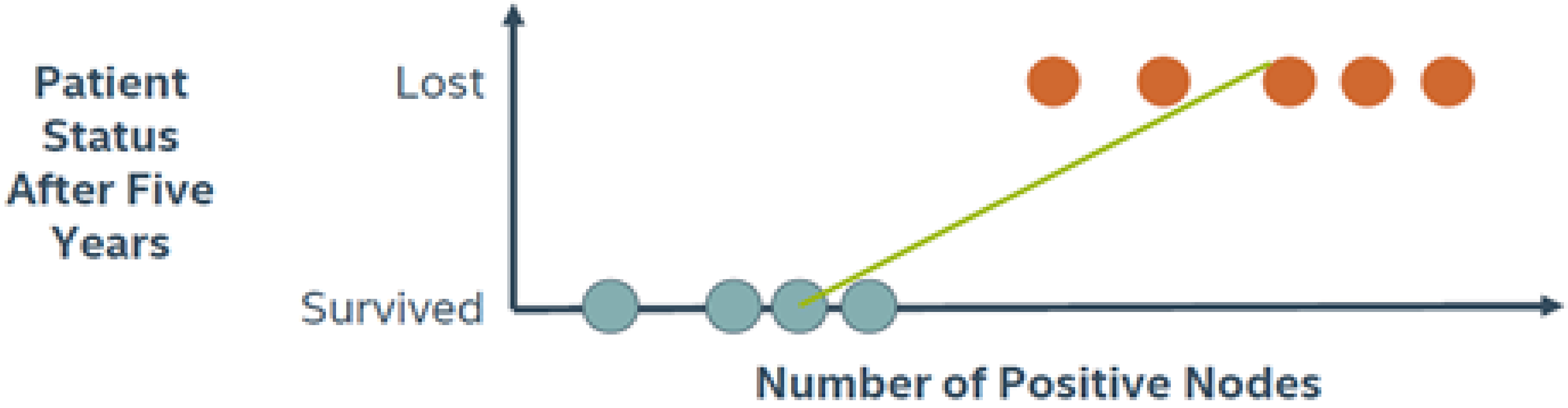
A Problem with Linear Regression

We could severely simplify the plot by drawing a line between the means for the two dependent variable levels, but this is problematic in two ways: (a) the line seems to oversimplify the relationship and (b) it gives predictions that cannot be observable values of Y for extreme values of X.



The reason this doesn't work is because the approach is analogous to fitting a linear model to the probability of the event. As you know, probabilities can only take values between 0 and 1. Hence, we need a different approach to ensure that our model is appropriate for the data.

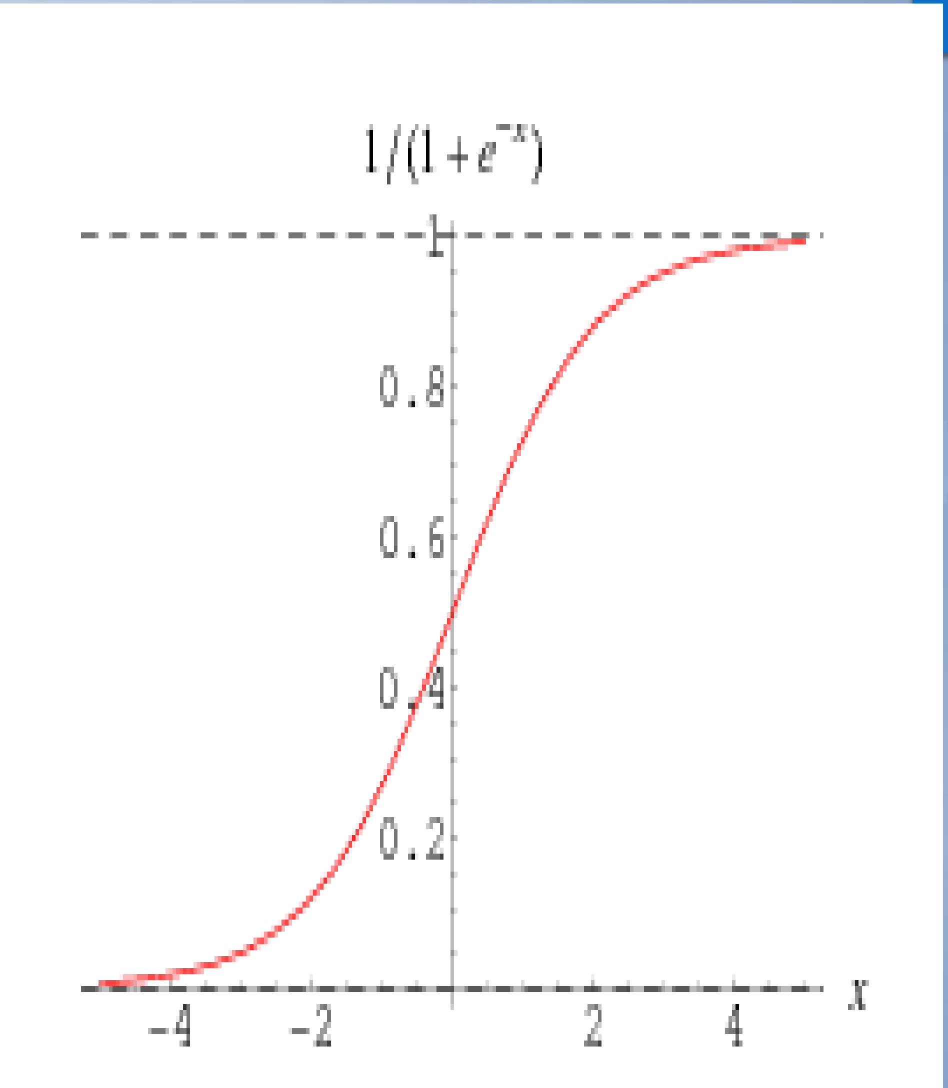
LINEAR REGRESSION FOR CLASSIFICATION?



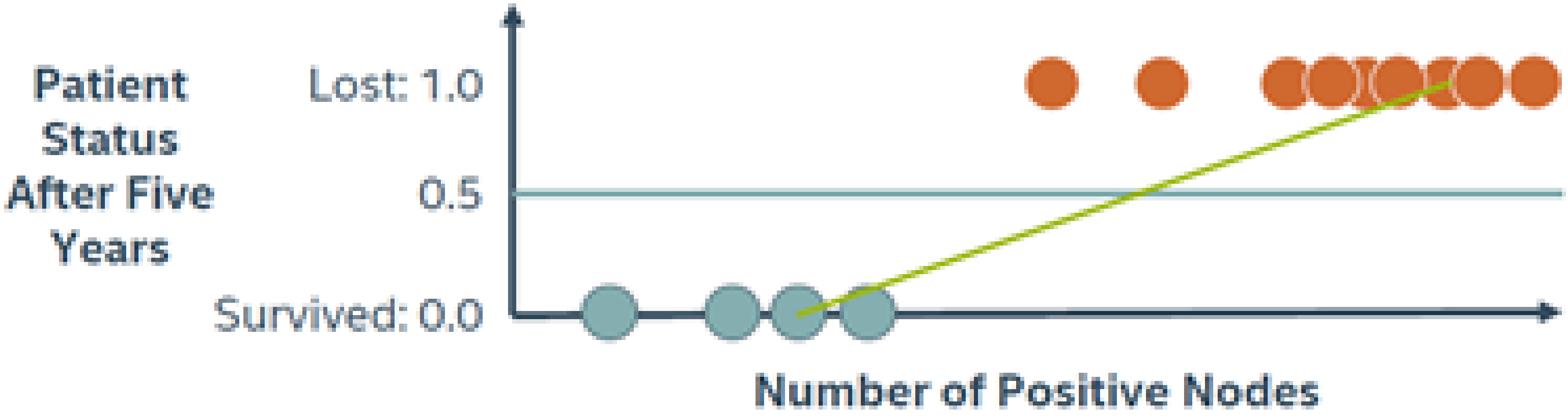
$$y_\beta(x) = \beta_0 + \beta_1 x + \varepsilon$$

A Problem with Linear Regression

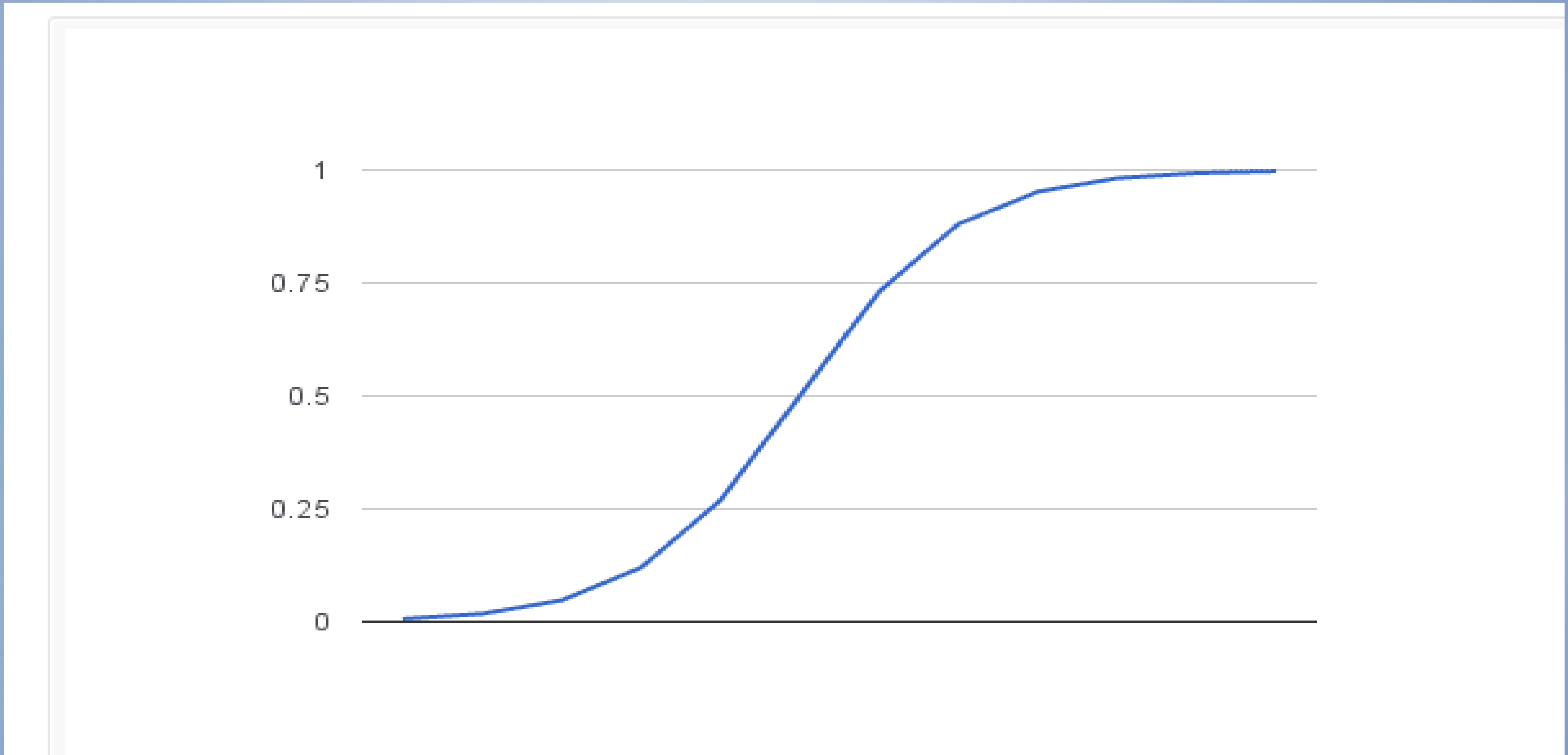
If you think about the shape of this distribution, you may posit that the function is a cumulative probability distribution. As stated previously, we can model the nonlinear relationship between X and Y by transforming one of the variables. Two common transformations that result in sigmoid functions are **probit** and **logit** transformations. In short, a probit transformation imposes a cumulative normal function on the data. But, probit functions are difficult to work with because they require integration. Logit transformations, on the other hand, give nearly identical values as a probit function, but they are much easier to work with because the function can be simplified to a linear equation.



LINEAR REGRESSION FOR CLASSIFICATION?



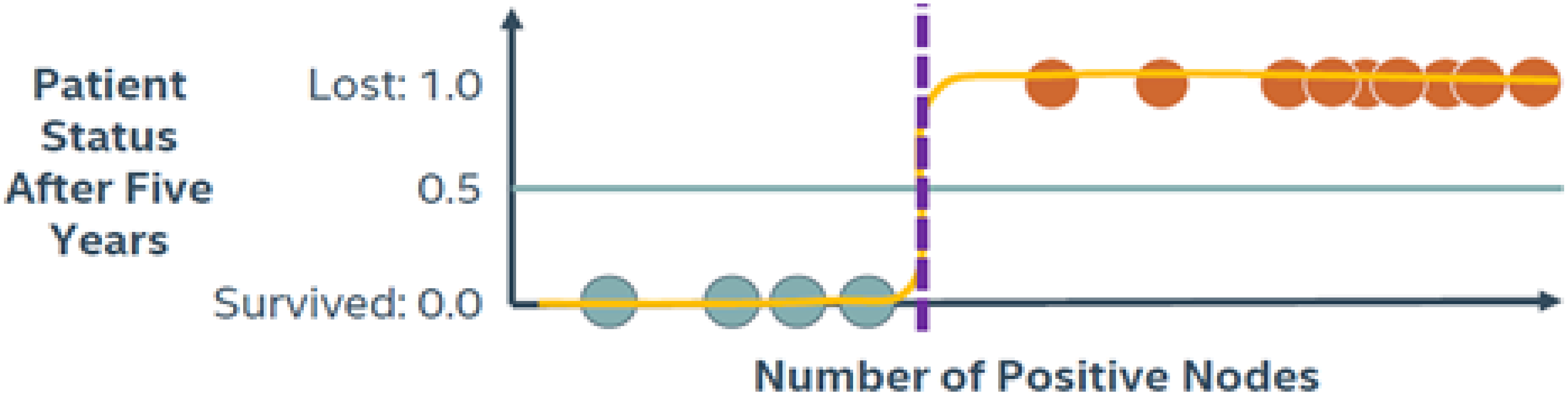
If model result > 0.5: predict lost
If model result < 0.5: predict survived


1
0.75
0.5
0.25
0

What is Logistic Regression?

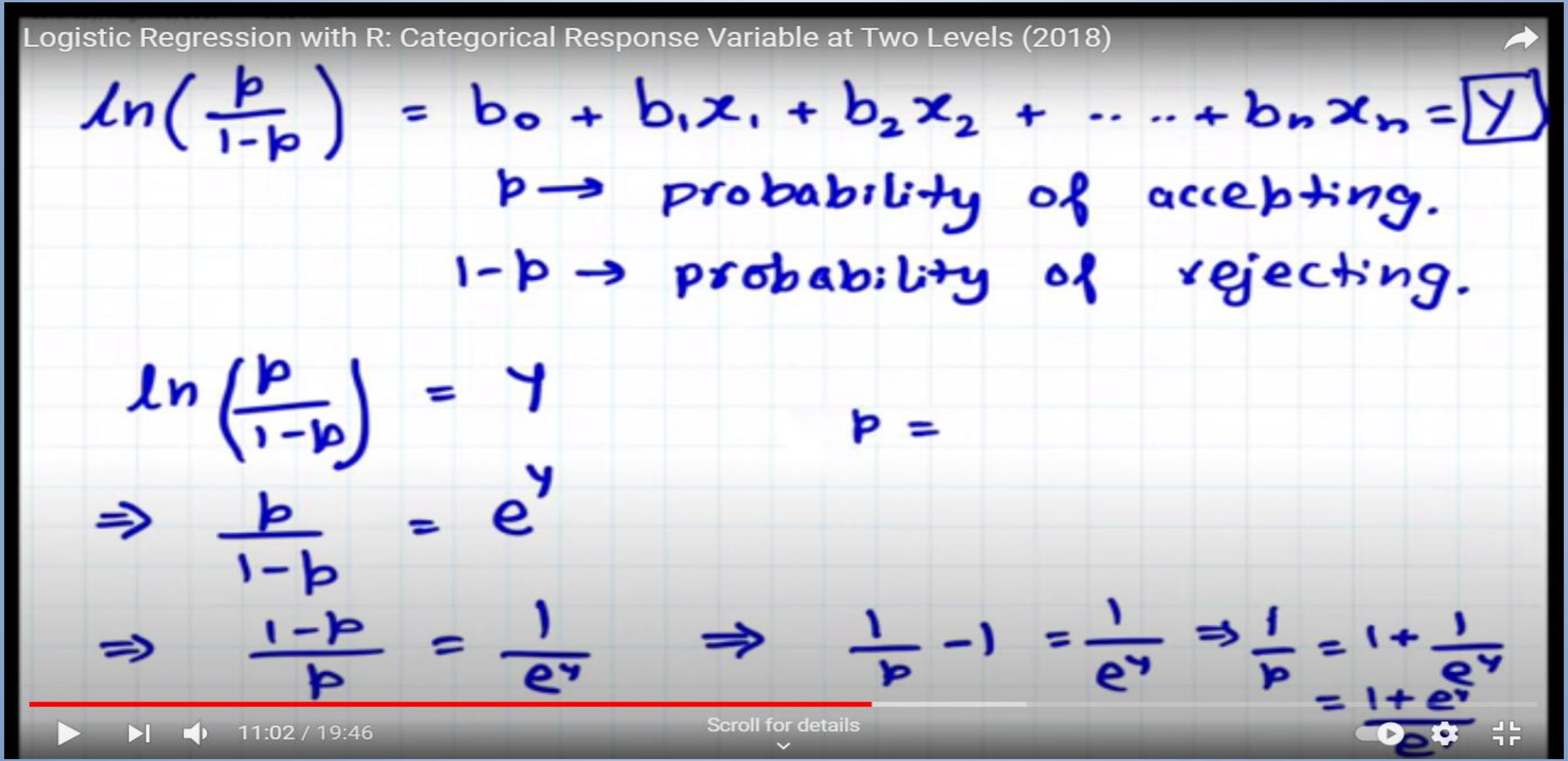
- Logistic regression is often used because the relationship between the DV (a discrete variable) and a predictor is non-linear
  - Example from the text: the probability of heart disease changes very little with a ten-point difference among people with low-blood pressure, but a ten point change can mean a drastic change in the probability of heart disease in people with high blood-pressure.

THE DECISION BOUNDARY



$$y_\beta(x) = \frac{1}{1+e^{-(\beta_0+\beta_1 x+\varepsilon)}}$$

Sigmoid Function—Logistic regression



$$\ln\left(\frac{p}{1-p}\right) = b_0 + b_1x_1 + b_2x_2 + \cdots\cdots + b_nx_n = \boxed{y}$$

$p \rightarrow$ probability of accepting.

$1-p \rightarrow$ probability of rejecting.

$$\ln\left(\frac{p}{1-p}\right) = y$$

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

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

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

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

11:08 / 19:46

RELATIONSHIP OF LOGISTIC TO LINEAR REGRESSION

Logistic Function

$$P(x) \quad = \frac{e^{(\beta_0 + \beta_1 x)}}{1 + e^{(\beta_0 + \beta_1 x)}}$$

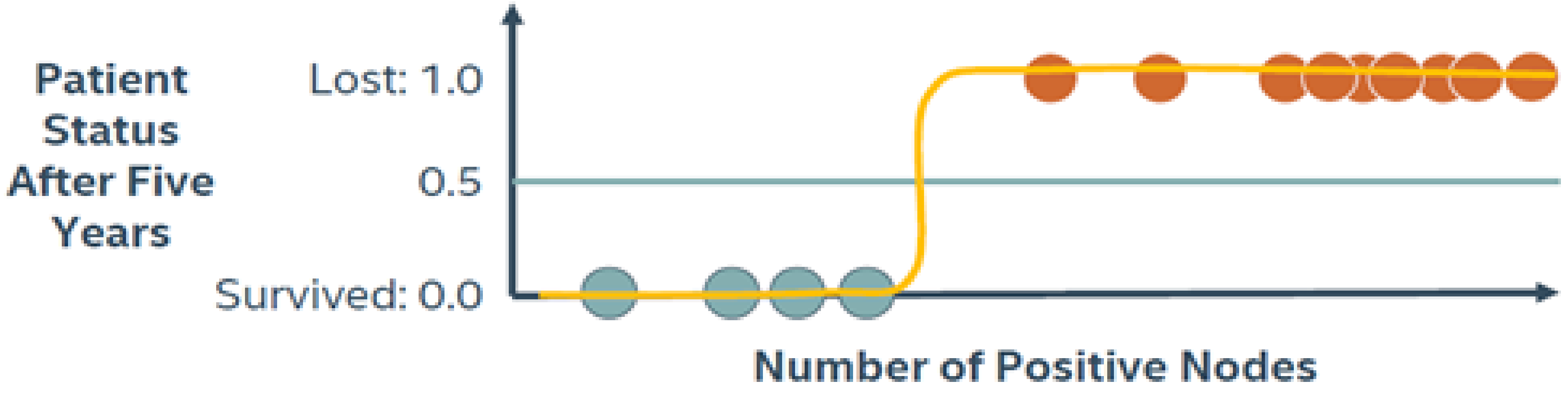
$$\Downarrow$$

Log Odds

$$log\left[\frac{P(x)}{1 - P(x)}\right] = \boxed{\beta_0 + \beta_1 x}$$

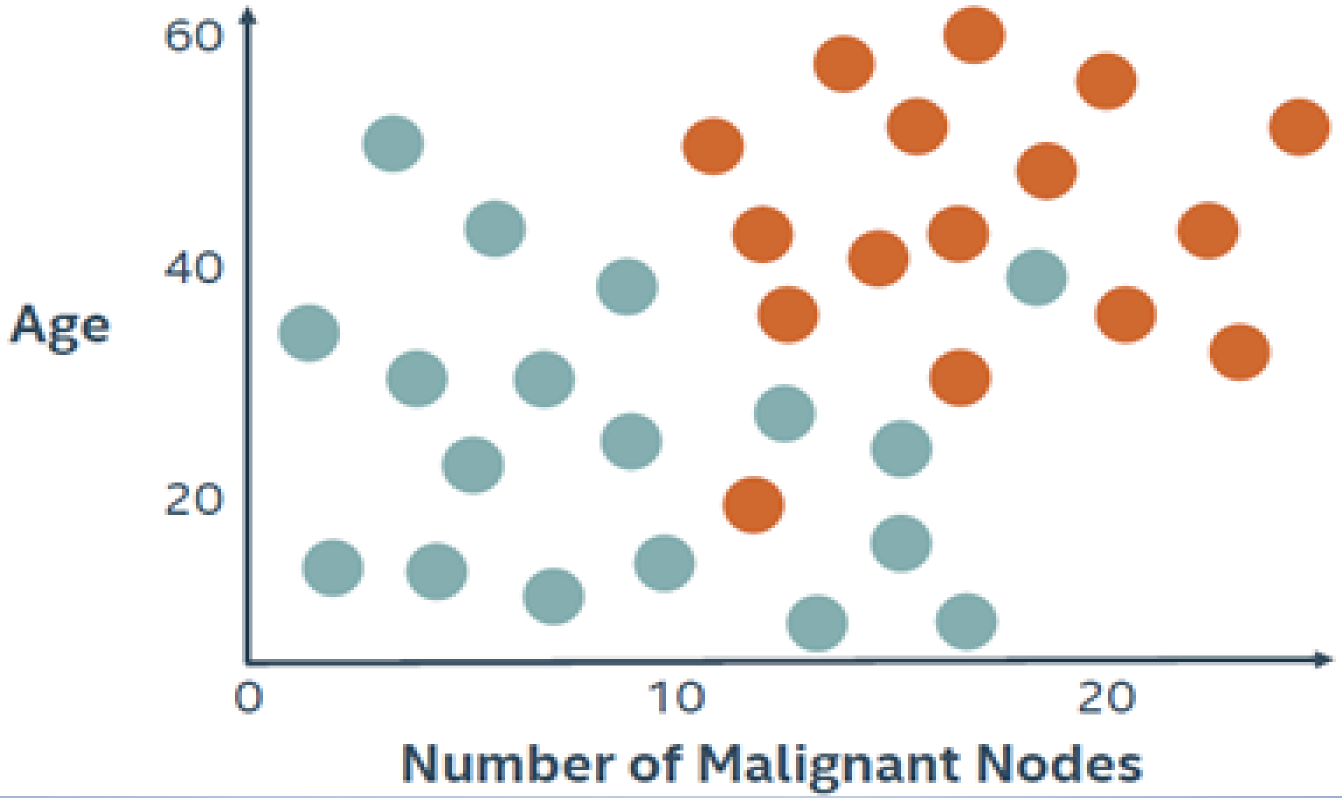
CLASSIFICATION WITH LOGISTIC REGRESSION

One feature (nodes)
Two labels (survived, lost)



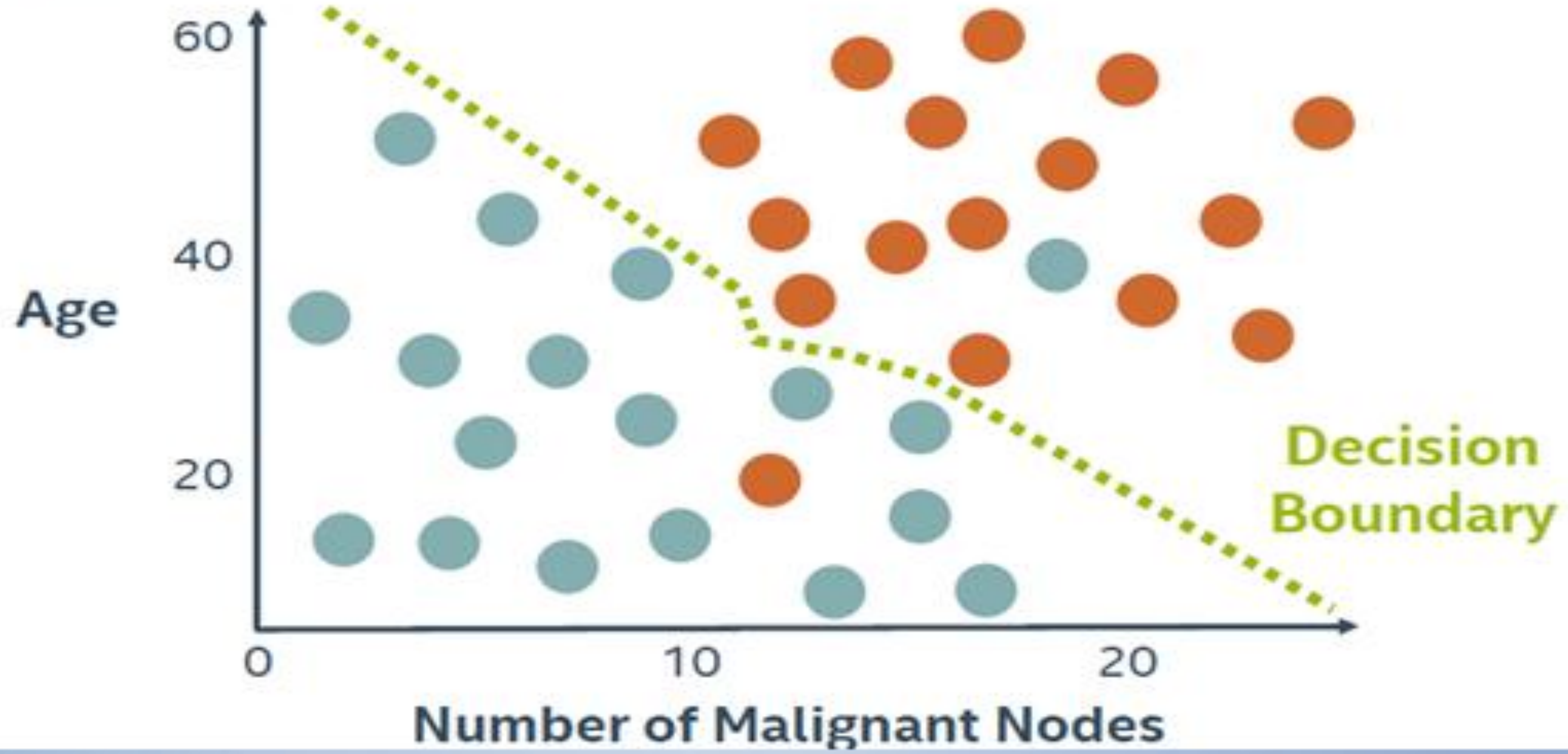
CLASSIFICATION WITH LOGISTIC REGRESSION

Two features (nodes, age)
Two labels (survived, lost)



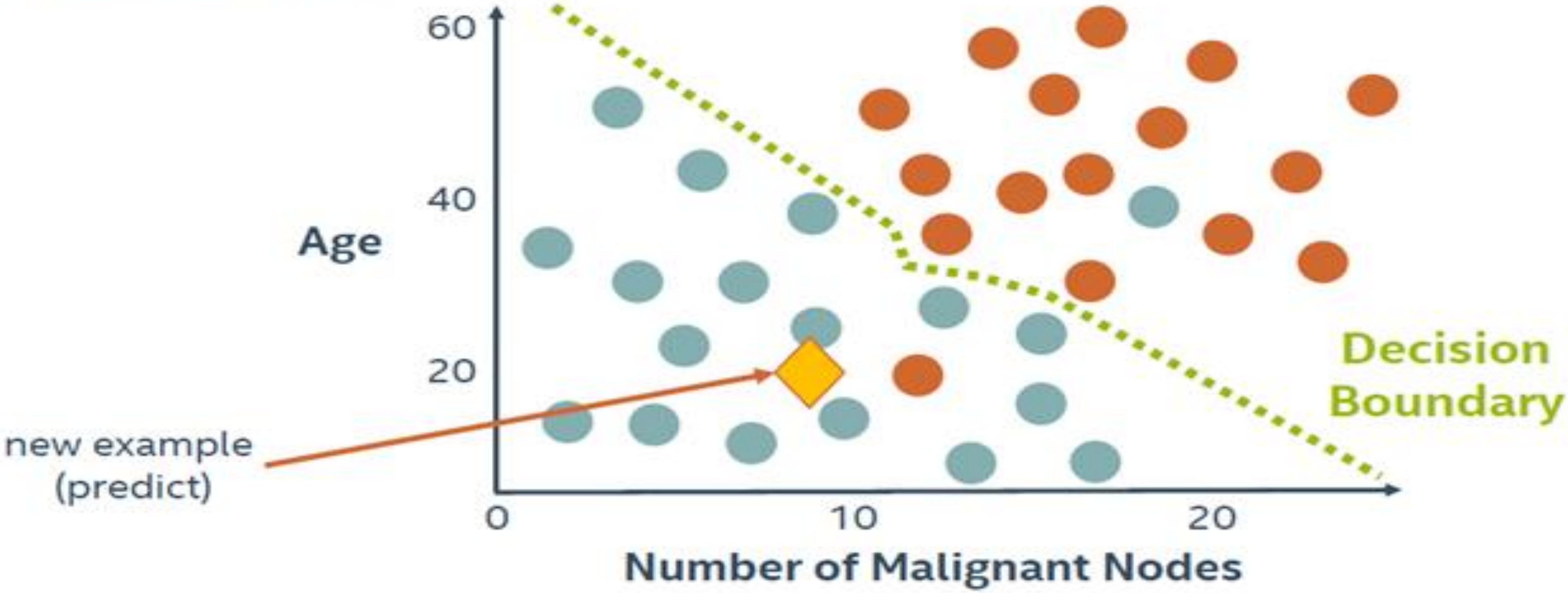
CLASSIFICATION WITH LOGISTIC REGRESSION

Two features (nodes, age)
Two labels (survived, lost)



CLASSIFICATION WITH LOGISTIC REGRESSION

Two features (nodes, age)
Two labels (survived, lost)



LOGISTIC REGRESSION: THE SYNTAX

Import the class containing the classification method

```
from sklearn.linear_model import LogisticRegression
```

Create an instance of the class

```
LR = LogisticRegression(penalty='l2', c=10.0)
```

Fit the instance on the data and then predict the expected value

```
LR = LR.fit(X_train, y_train)

y_predict = LR.predict(X_test)
```

Classification error metrics

CHOOSING THE RIGHT ERROR MEASUREMENT

- You are asked to build a classifier for leukemia
- **Training data:** 1% patients with leukemia, 99% healthy
- **Measure accuracy:** total % of predictions that are correct

CONFUSION MATRIX

	Predicted Positive	Predicted Negative
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

CONFUSION MATRIX

	Predicted Positive	Predicted Negative	
Actual Positive	True Positive (TP)	False Negative (FN)	Type II Error
Actual Negative	False Positive (FP)	True Negative (TN)	

Type I Error

Confusion Matrix : Intuition

		Prediction outcome			
		positive	negative		
Actual value	positive	TP	FN	$TP + FN$	Total Actual positive
	negative	FP	TN	$FP + TN$	Total Actual negative

ACCURACY: PREDICTING CORRECTLY

	Predicted Positive	Predicted Negative
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

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

RECALL: IDENTIFYING ALL POSITIVE INSTANCES

	Predicted Positive	Predicted Negative
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

$$\text{Recall or Sensitivity} = \frac{TP}{TP + FN}$$

True Positive Rate

$$TPR = \frac{TP}{TP + FN}$$

		Prediction outcome	
		positive	negative
Actual value	positive	TP	FN
	negative	FP	TN

Ratio of actual positive predictions over total actual p

False Negative Rate

$$TPR = \frac{TP}{TP + FN}$$

$$FNR = \frac{FN}{TP + FN}$$

		Prediction outcome	
		positive	negative
Actual value	positive	TP	FN
	negative	FP	TN

PRECISION: IDENTIFYING ONLY POSITIVE INSTANCES

	Predicted Positive	Predicted Negative
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

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

SPECIFICITY: AVOIDING FALSE ALARMS

	Predicted Positive	Predicted Negative
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

$$\text{Specificity} = \frac{TN}{FP + TN}$$

ERROR MEASUREMENTS

	Predicted Positive	Predicted Negative
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

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

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

ERROR MEASUREMENTS

	Predicted Positive	Predicted Negative
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

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

$$\text{Recall or Sensitivity} = \frac{TP}{TP + FN}$$

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

$$\text{Specificity} = \frac{TN}{FP + TN}$$

ERROR MEASUREMENTS

	Predicted Positive	Predicted Negative
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

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

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

$$\text{Recall or Sensitivity} = \frac{TP}{TP + FN}$$

$$\text{Specificity} = \frac{TN}{FP + TN}$$

$$F1 = 2\ \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

MULTIPLE CLASS ERROR METRICS

	Predicted Class 1	Predicted Class 2	Predicted Class 3
Actual Class 1	TP1		
Actual Class 2		TP2	
Actual Class 3			TP3

MULTIPLE CLASS ERROR METRICS

	Predicted Class 1	Predicted Class 2	Predicted Class 3
Actual Class 1	TP1		
Actual Class 2		TP2	
Actual Class 3			TP3

$$\text{Accuracy} = \frac{\text{TP1} + \text{TP2} + \text{TP3}}{\text{Total}}$$

Most multi-class error metrics are similar to binary versions—just expand elements as a sum

CLASSIFICATION ERROR METRICS: THE SYNTAX

Import the desired error function

```
from sklearn.metrics import accuracy_score
```

Calculate the error on the test and predicted data sets

```
accuracy_value = accuracy_score(y_test, y_pred)
```