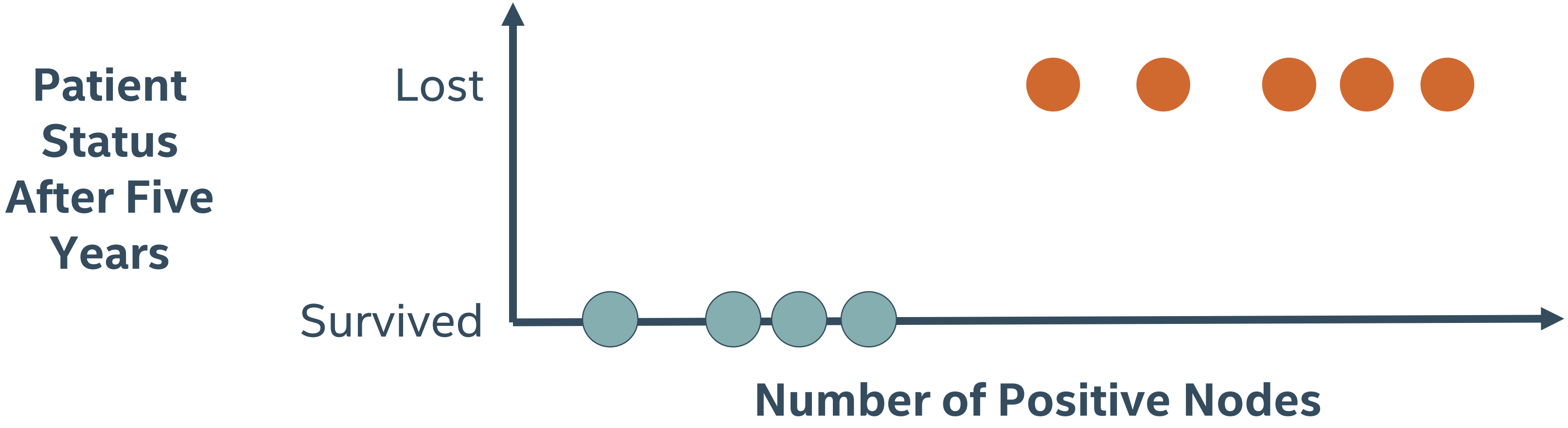
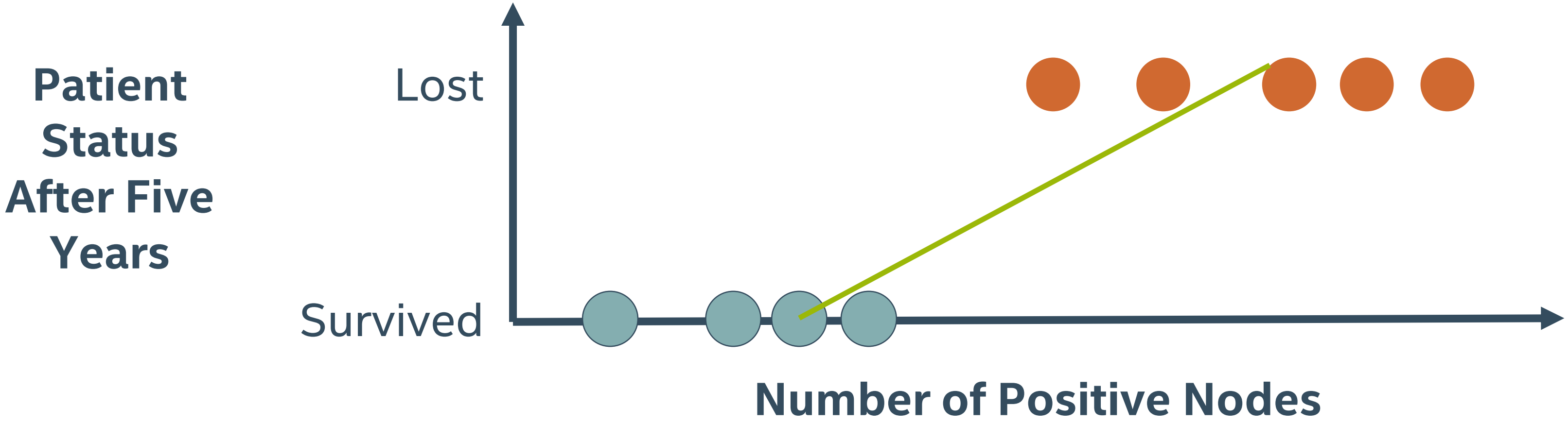
LOGISTIC REGRESSION

INTRODUCTION TO LOGISTIC REGRESSION

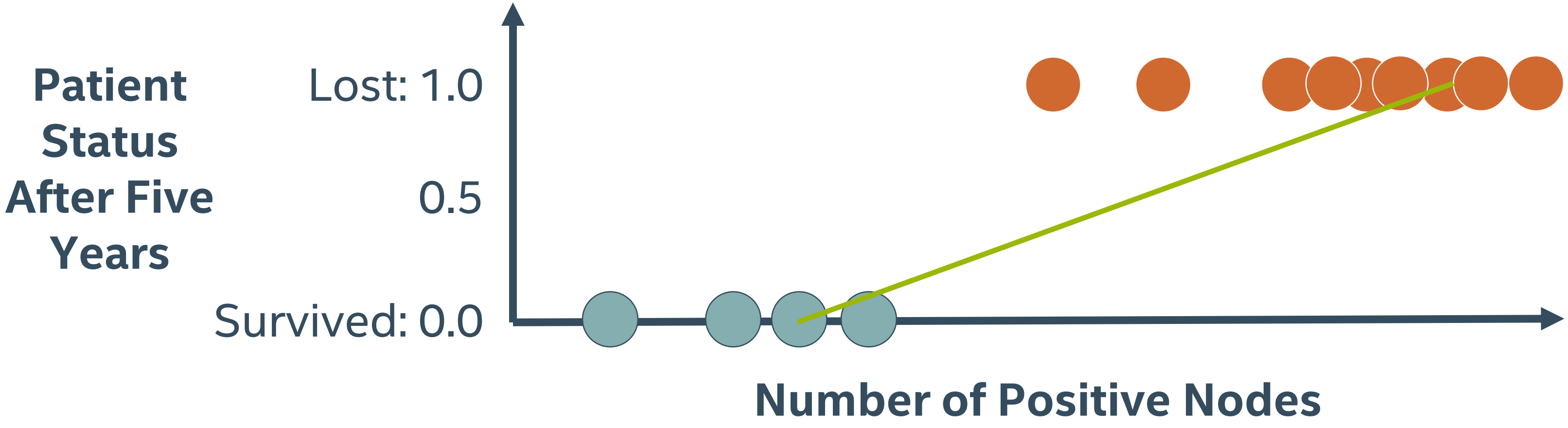


LINEAR REGRESSION FOR CLASSIFICATION?



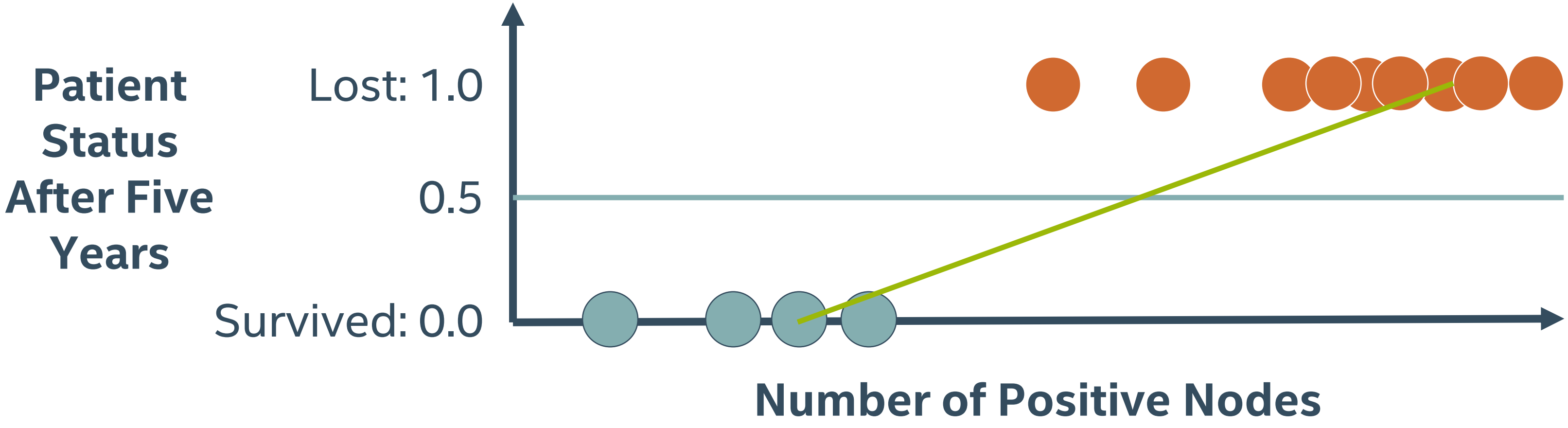
$$y_\beta(x) = \beta_0 + \beta_1 x + \varepsilon$$

LINEAR REGRESSION FOR CLASSIFICATION?



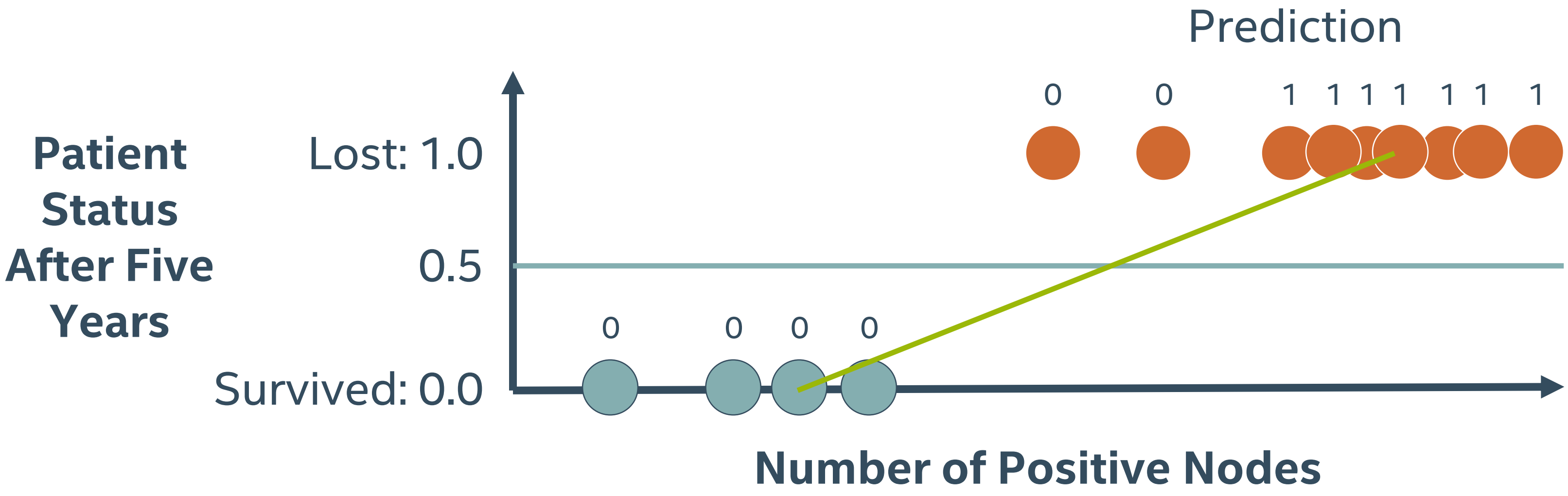
$$y_\beta(x) = \beta_0 + \beta_1 x + \varepsilon$$

LINEAR REGRESSION FOR CLASSIFICATION?



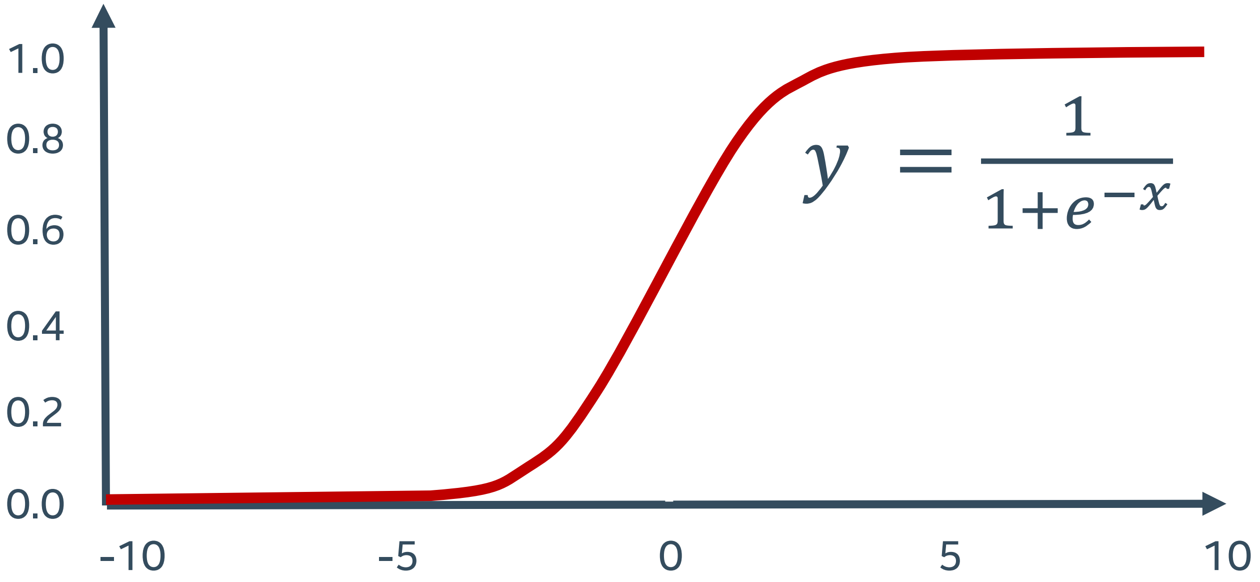
If model result > 0.5: predict lost
If model result < 0.5: predict survived

LINEAR REGRESSION FOR CLASSIFICATION?

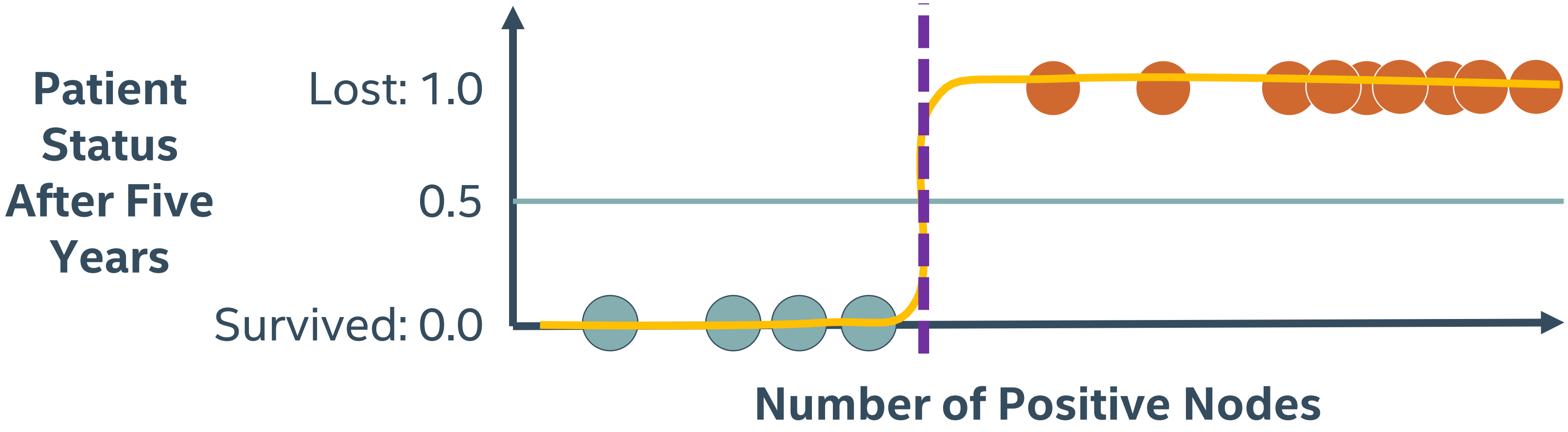


If model result > 0.5: predict lost
If model result < 0.5: predict survived

WHAT IS THIS FUNCTION?

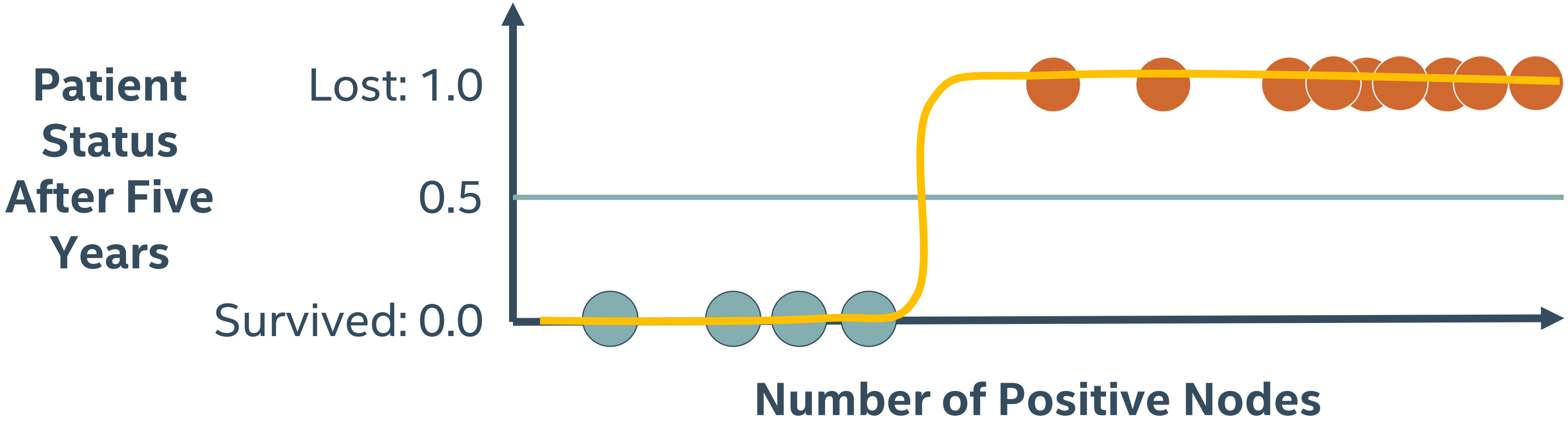


THE DECISION BOUNDARY



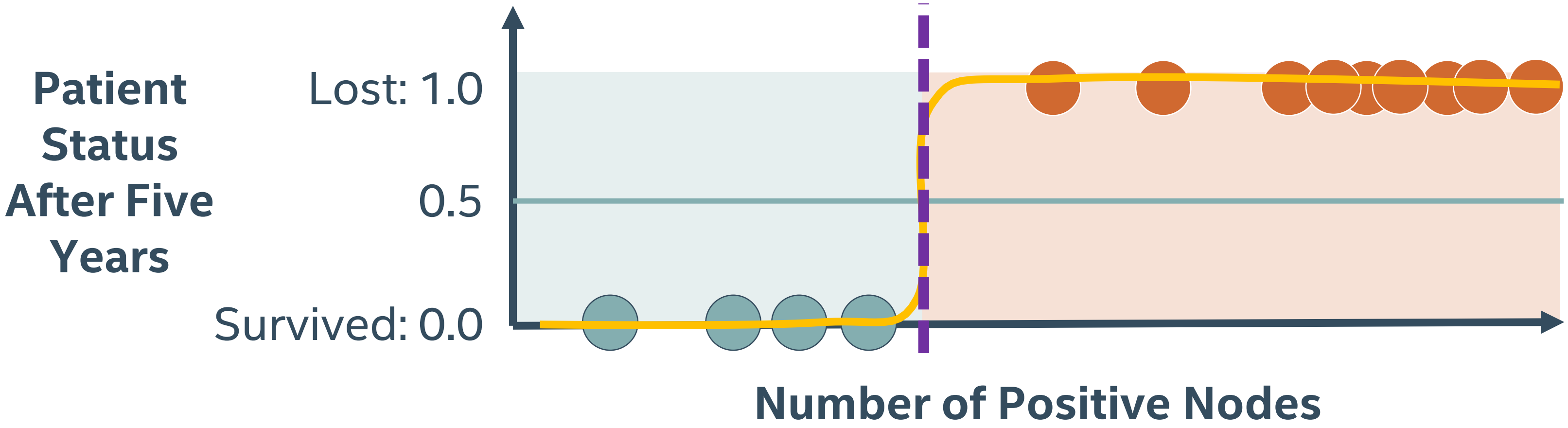
$$y_\beta(x) = \frac{1}{1+e^{-(\beta_0 + \beta_1 x + \varepsilon\ )}}$$

LOGISTIC REGRESSION



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

THE DECISION BOUNDARY



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

RELATIONSHIP OF LOGISTIC TO LINEAR REGRESSION

Logistic Function

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

RELATIONSHIP OF LOGISTIC TO LINEAR REGRESSION

Logistic Function

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

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

RELATIONSHIP OF LOGISTIC TO LINEAR REGRESSION

Logistic Function

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

RELATIONSHIP OF LOGISTIC TO LINEAR REGRESSION

Logistic Function

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

Odds Ratio

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

RELATIONSHIP OF LOGISTIC TO LINEAR REGRESSION

Logistic Function

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

Log Odds

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

RELATIONSHIP OF LOGISTIC TO LINEAR REGRESSION

Logistic Function

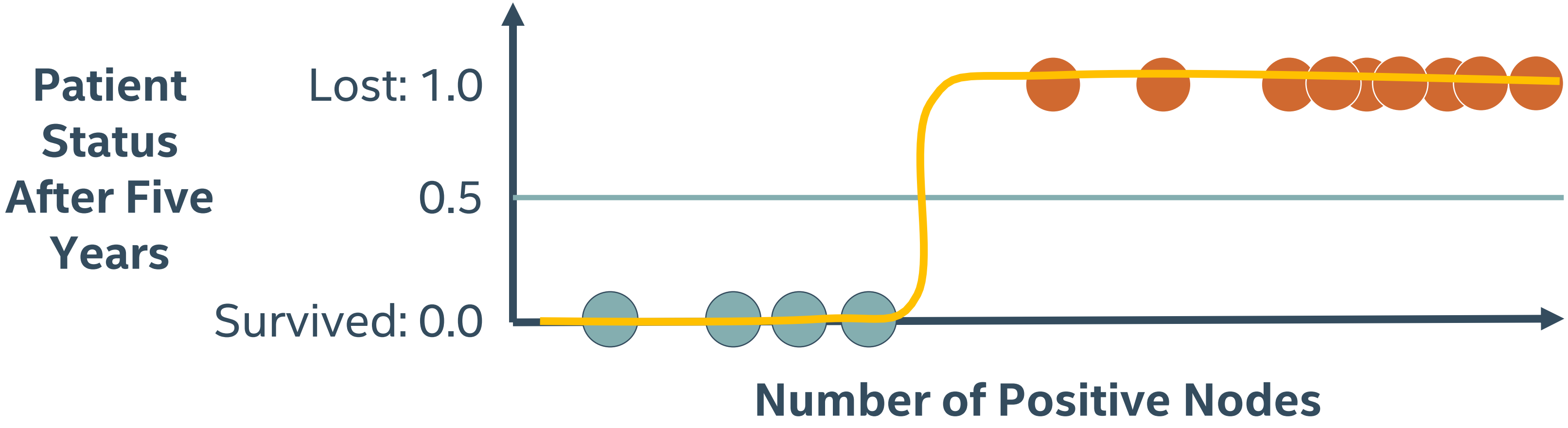
$$P(x) = \frac{e^{(\beta_0 + \beta_1 x)}}{1 + e^{(\beta_0 + \beta_1 x)}}$$

Log Odds

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

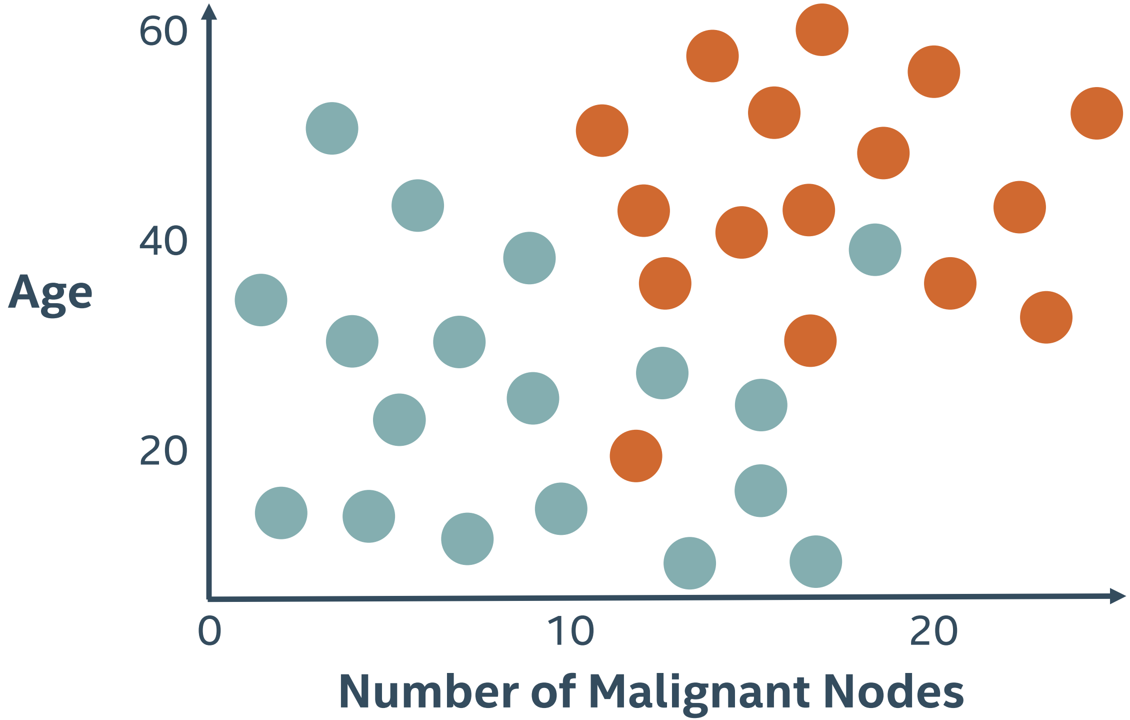
CLASSIFICATION WITH LOGISTIC REGRESSION

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



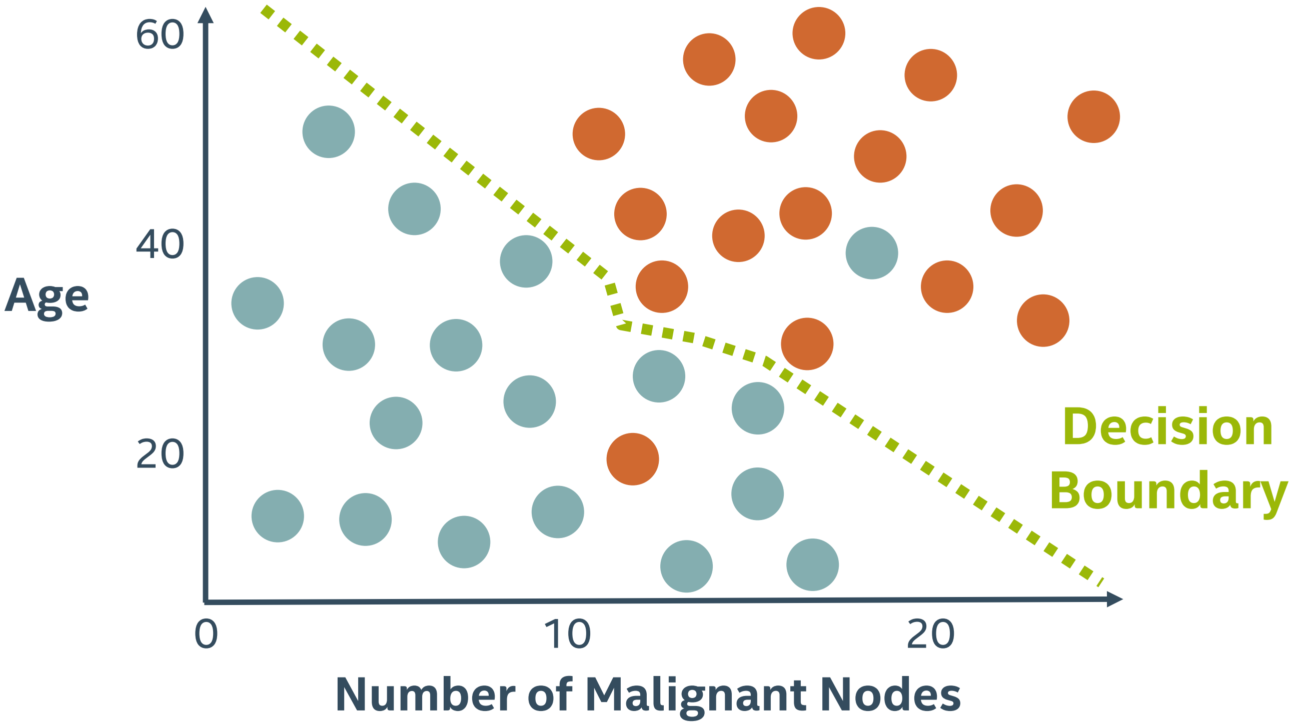
CLASSIFICATION WITH LOGISTIC REGRESSION

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



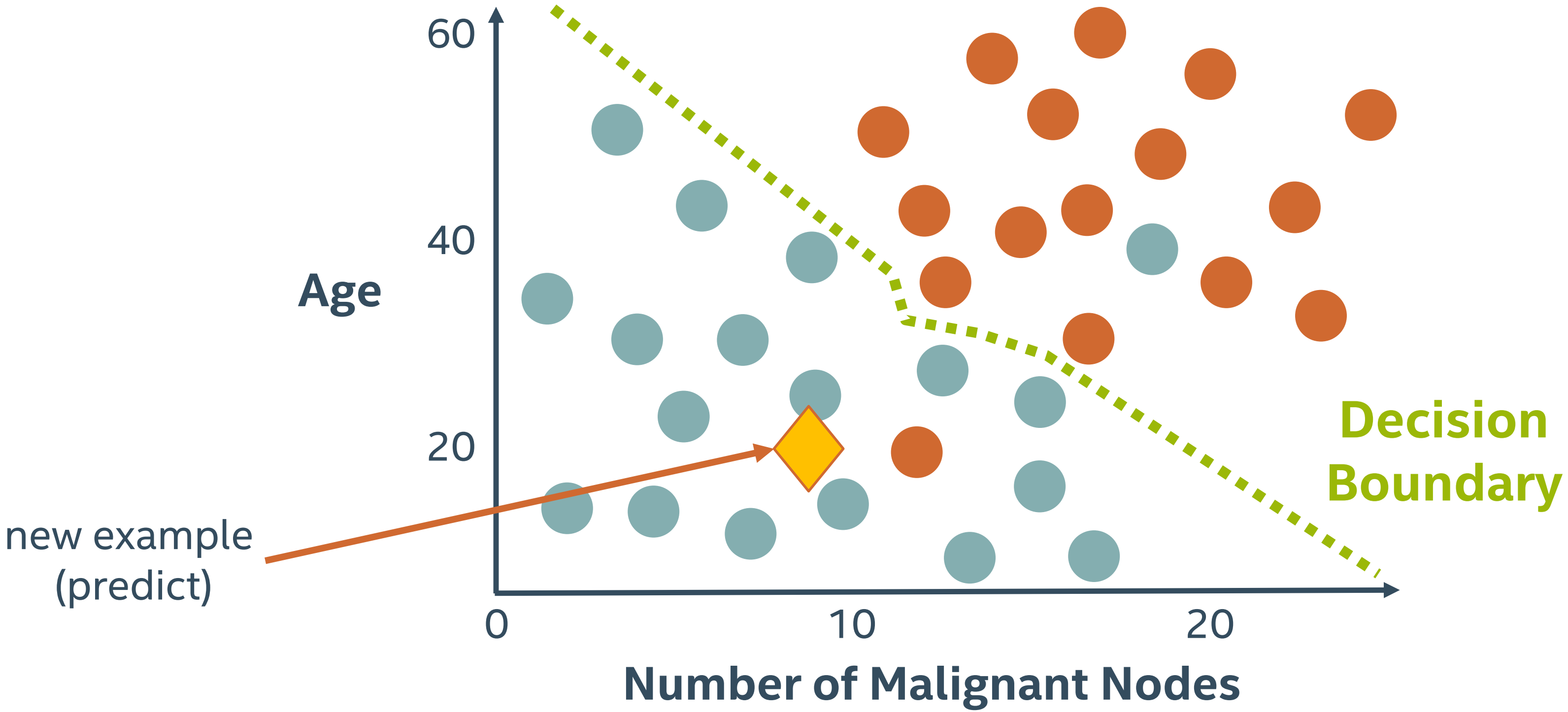
CLASSIFICATION WITH LOGISTIC REGRESSION

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



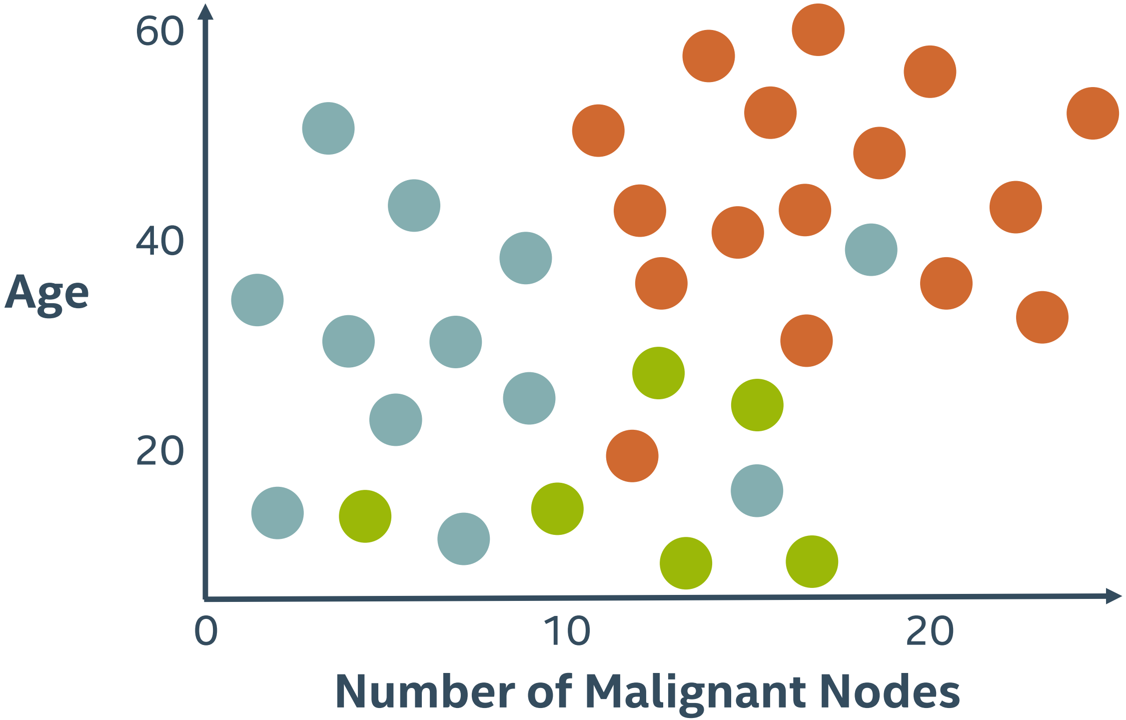
CLASSIFICATION WITH LOGISTIC REGRESSION

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

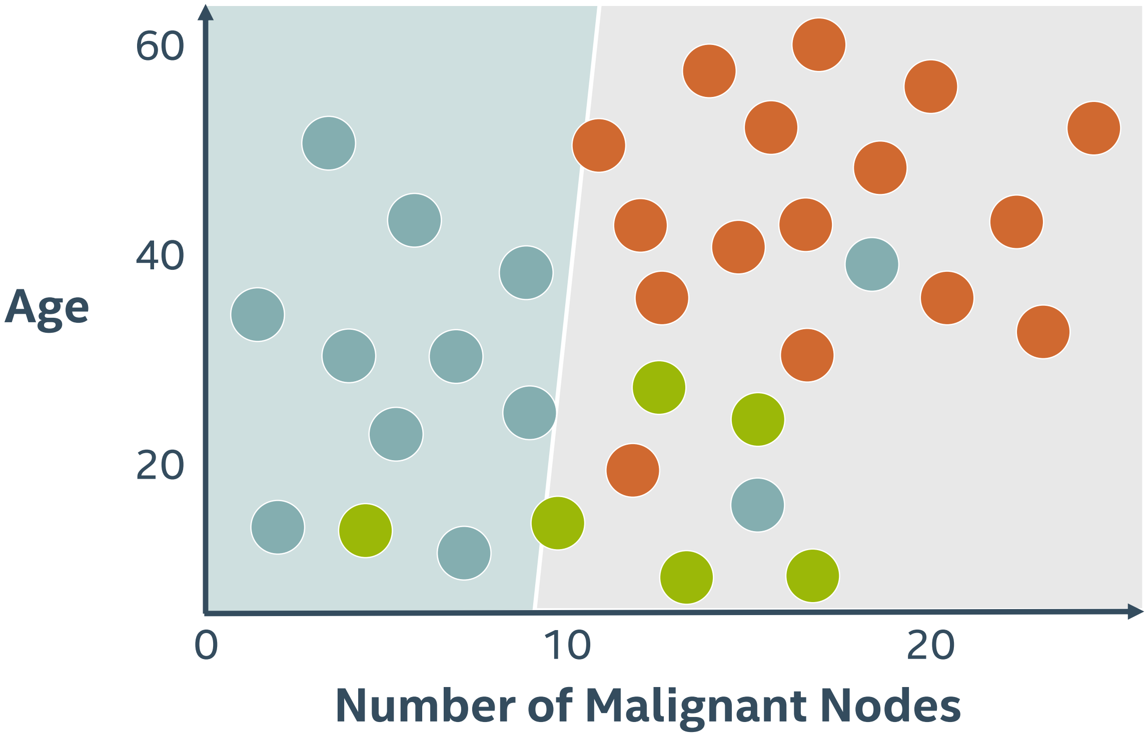


MULTICLASS CLASSIFICATION WITH LOGISTIC REGRESSION

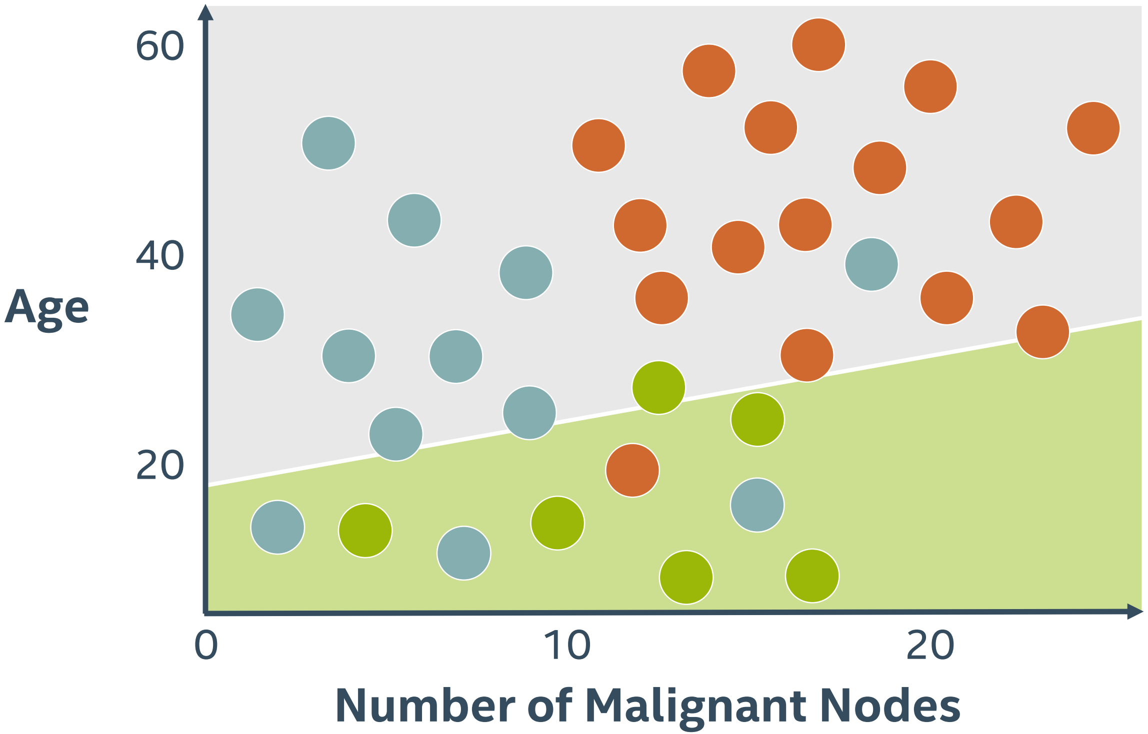
Two features (nodes, age)
Three labels (survived, complications, lost)



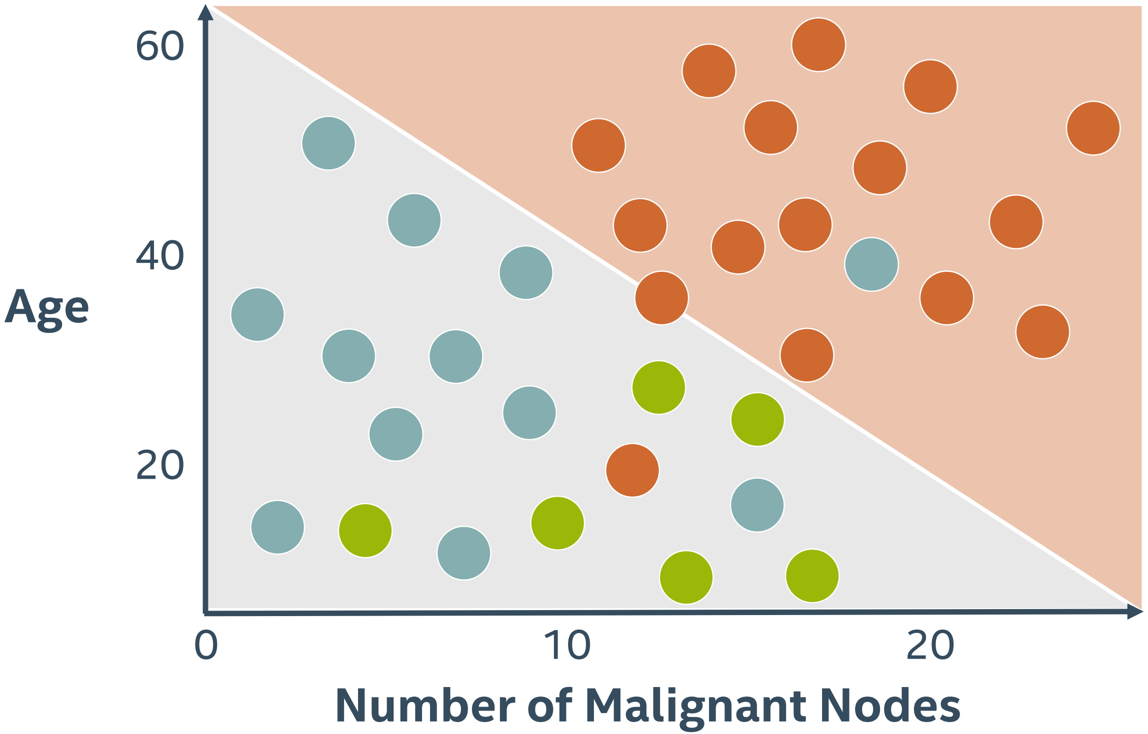
ONE VS ALL: SURVIVED VS ALL



ONE VS ALL: COMPLICATIONS VS ALL

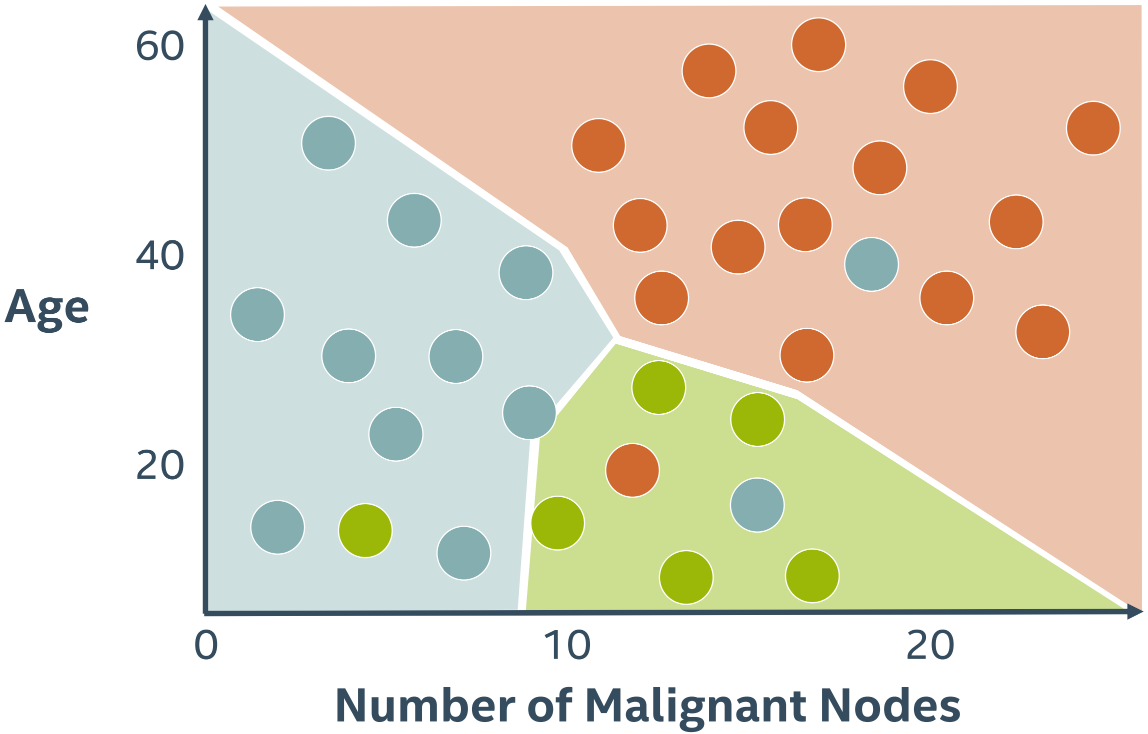


ONE VS ALL: LOSS VS ALL



MULTICLASS DECISION BOUNDARY

Assign most probable class to each region



LOGISTIC REGRESSION: THE SYNTAX

Import the class containing the classification method

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

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)
```

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)
```

regularization parameters

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)
```

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)
```

Tune regularization parameters with cross-validation: LogisticRegressionCV.

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

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
- Build a simple model that always predicts “healthy”
- Accuracy will be 99%...

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)
Actual Negative	False Positive (FP)	True Negative (TN)

Type II Error

Type I Error

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)

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

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)

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

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

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

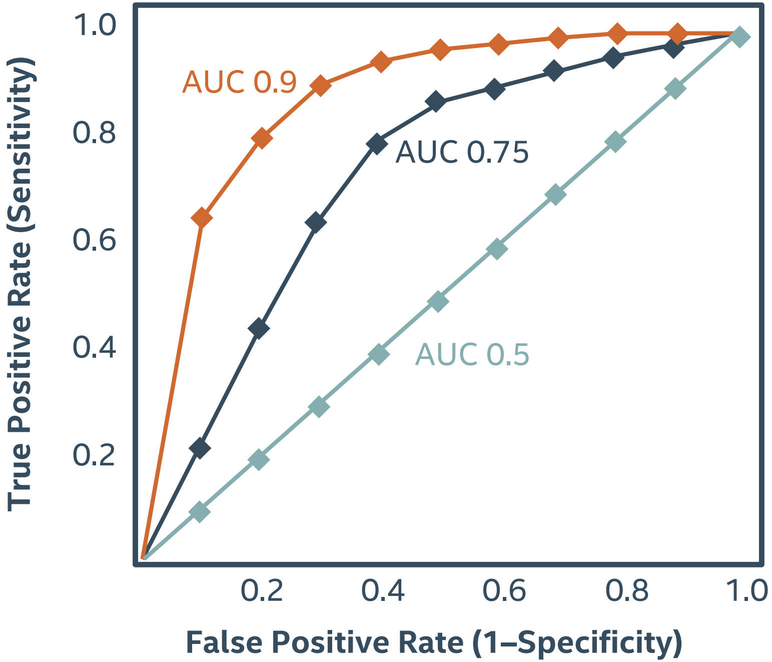
$$\text{Precision} = \frac{TP}{TP + FP}$$

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

RECEIVER OPERATING CHARACTERISTIC (ROC)

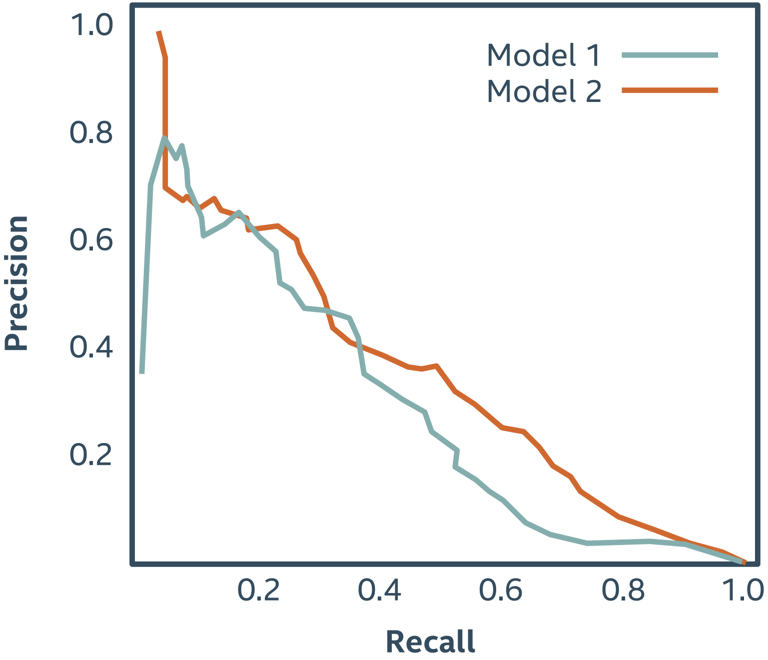
Evaluation of model at all possible thresholds

AREA UNDER CURVE (AUC)



Measures total area under ROC curve

PRECISION RECALL CURVE (PR CURVE)



Measures trade-off between precision and 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

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

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

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

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
```

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)
```

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)
```

Lots of other error metrics and diagnostic tools:

```
from sklearn.metrics import precision_score, recall_score,
    f1_score, roc_auc_score,
    confusion_matrix, roc_curve,
    precision_recall_curve
```

Nervana™ AI Academy