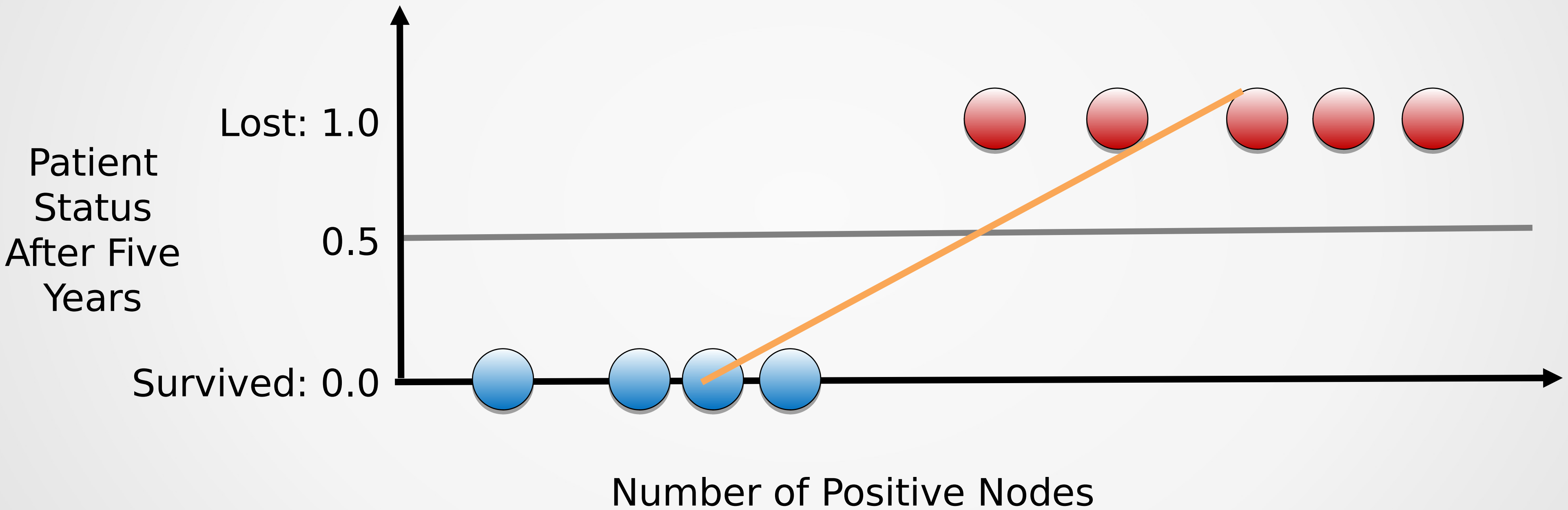
Logistic Regression And Classification Error Metrics

Learning Objectives

After completing this lecture, you will be able to:-

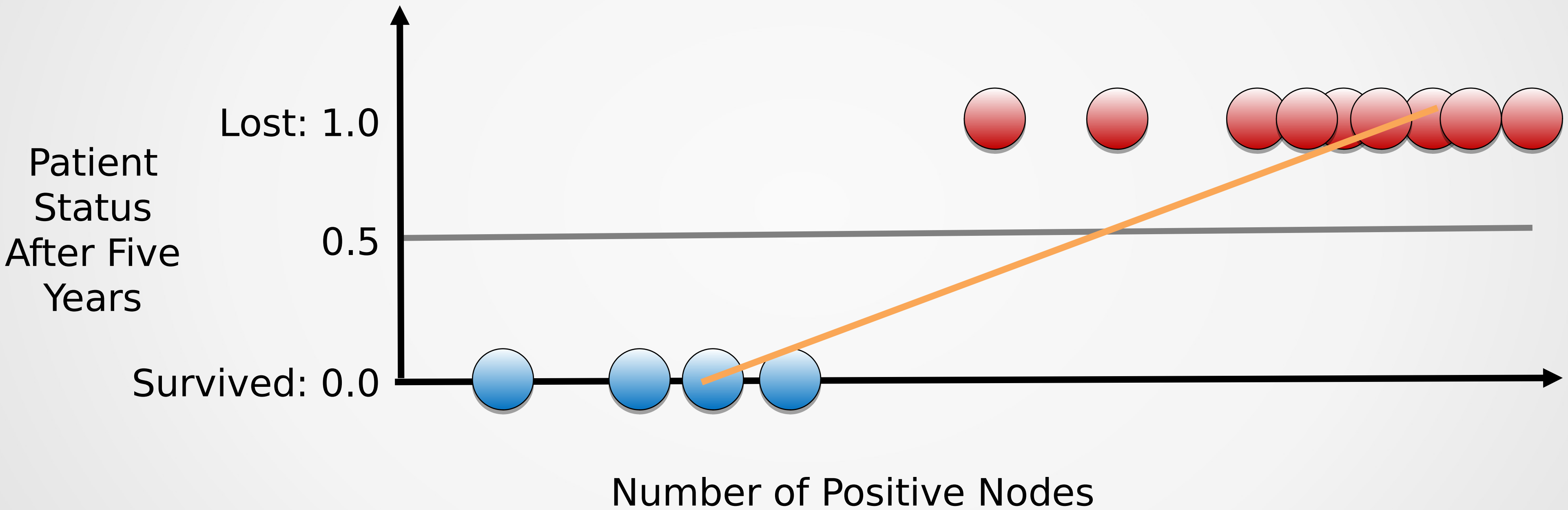
- Describe logistic regression
- Implement logistic function based optimization for any regression or classification problem
- Define and calculate the basic classification error metrics
- Utilize the advanced (compound) classification error metrics

Introduction to Logistic Regression



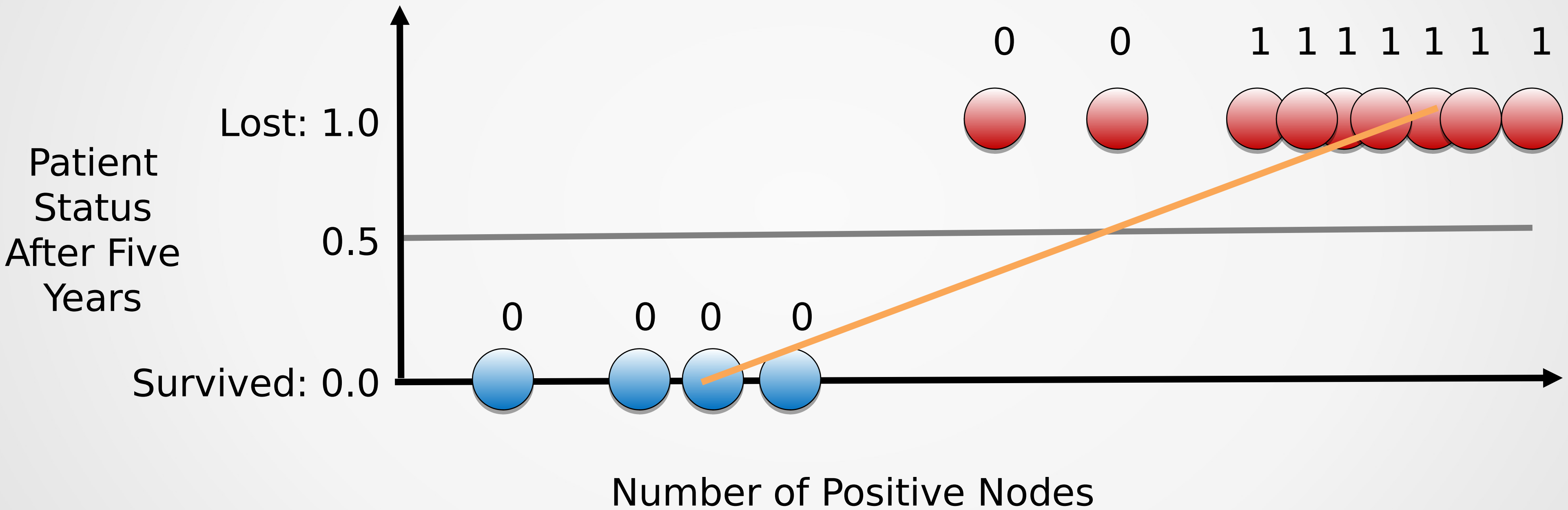
$$y_\beta(x) = \beta_0 + \beta_1 x + \varepsilon$$

Introduction to Logistic Regression



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

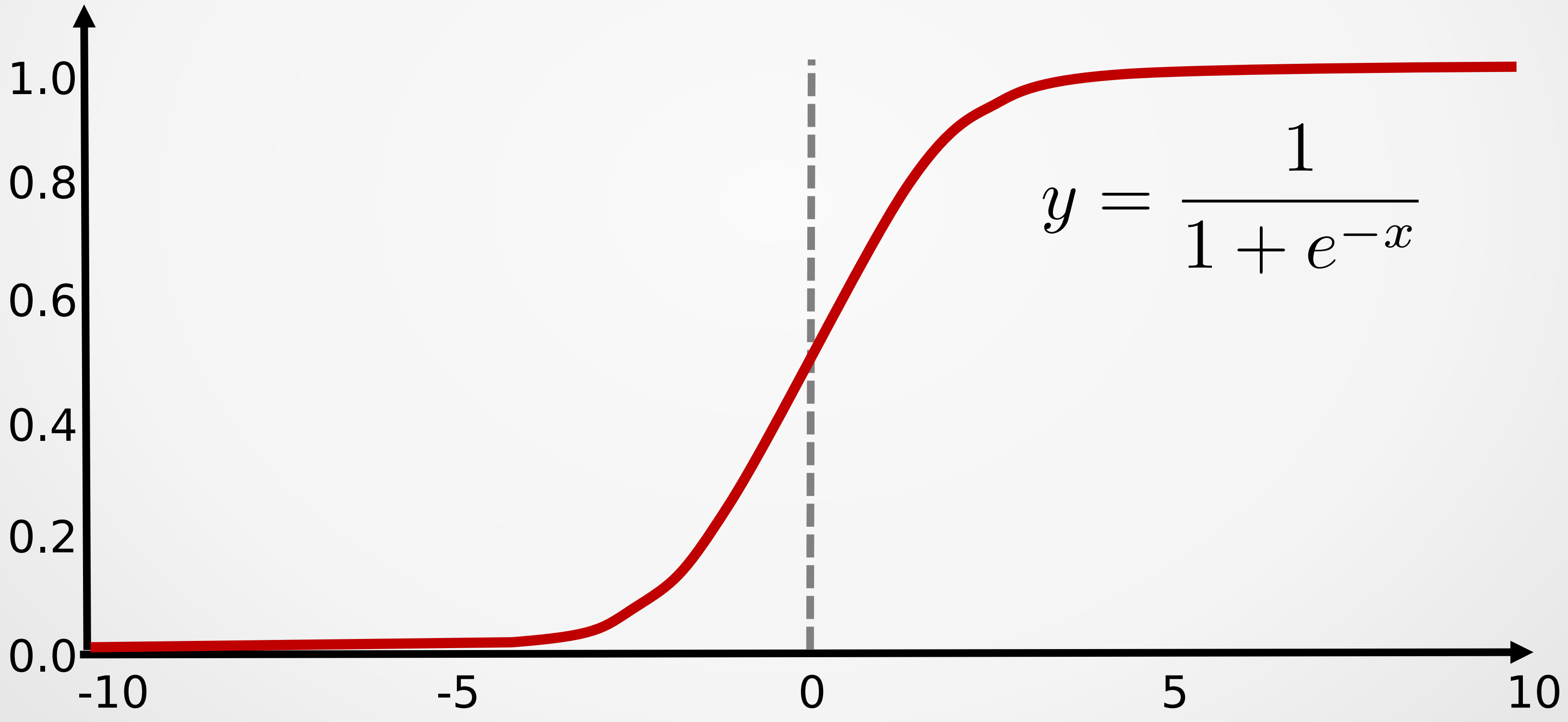
Introduction to Logistic Regression



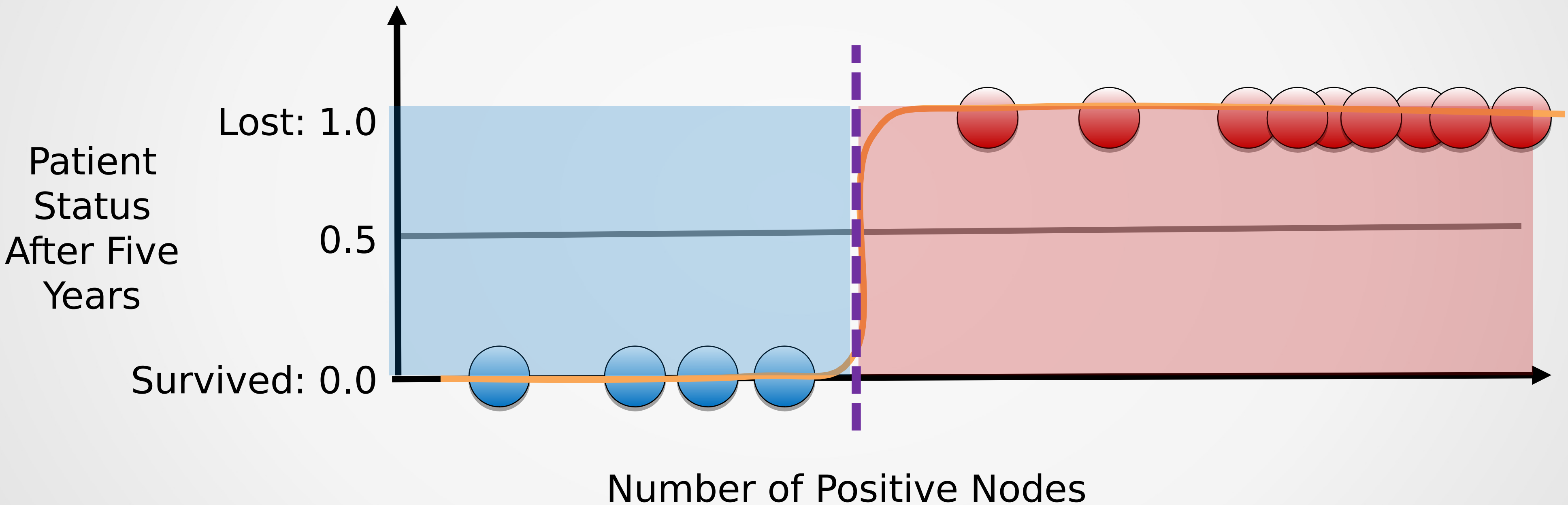
$$y_\beta(x) = \beta_0 + \beta_1 x + \varepsilon$$

Introduction to Logistic Regression

What is this function?



Classification with Logistic Regression



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

Relating to Linear Regression

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

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

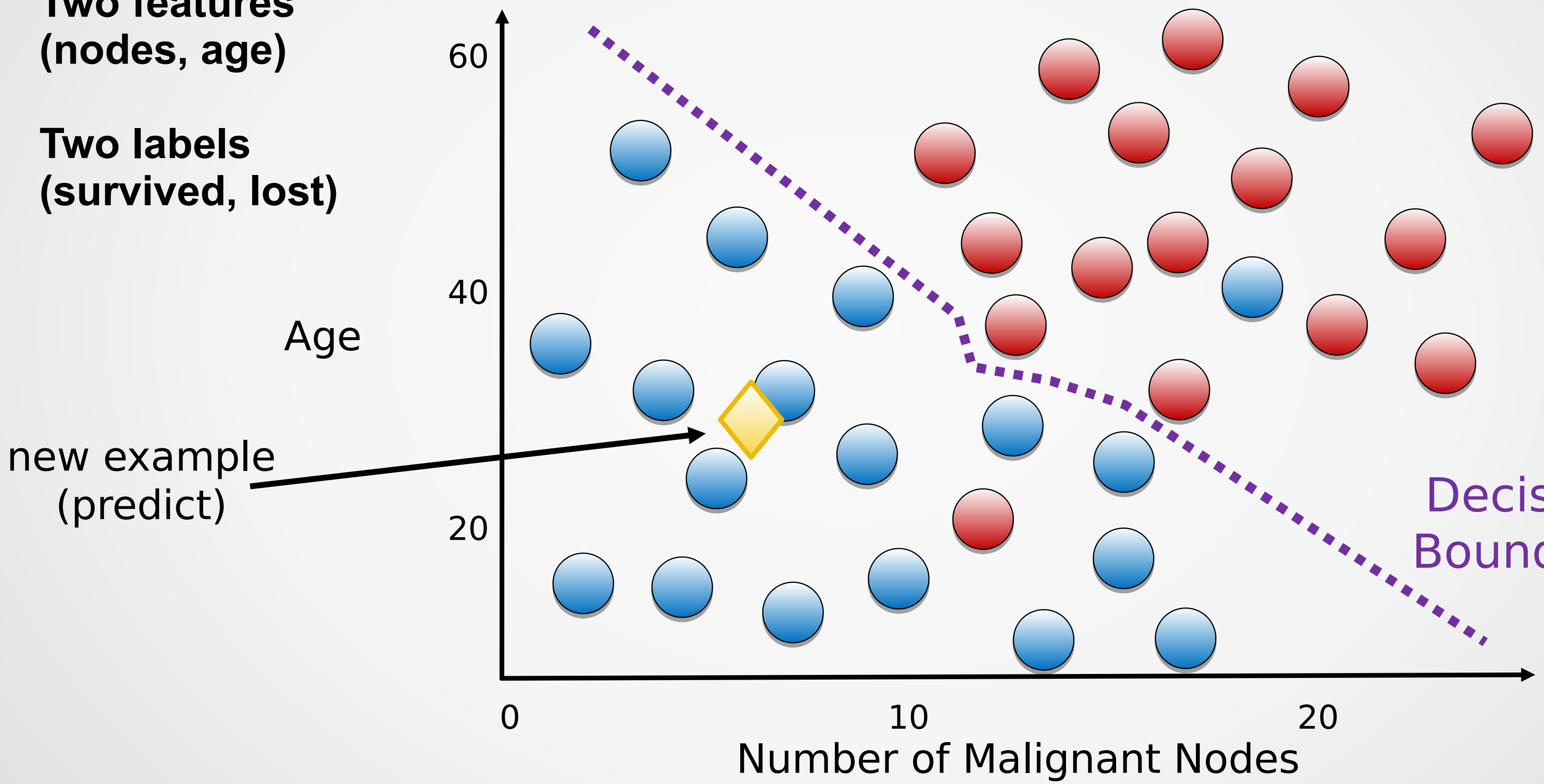
Odds Ratio
$$\frac{P(x)}{1 - P_x} = e^{(\beta_0 + \beta_1 x)}$$

Log Odds
$$\log\left(\frac{P(x)}{1 - P_x}\right) = \beta_0 + \beta_1 x$$

Classification with Logistic Regression

Two features (nodes, age)

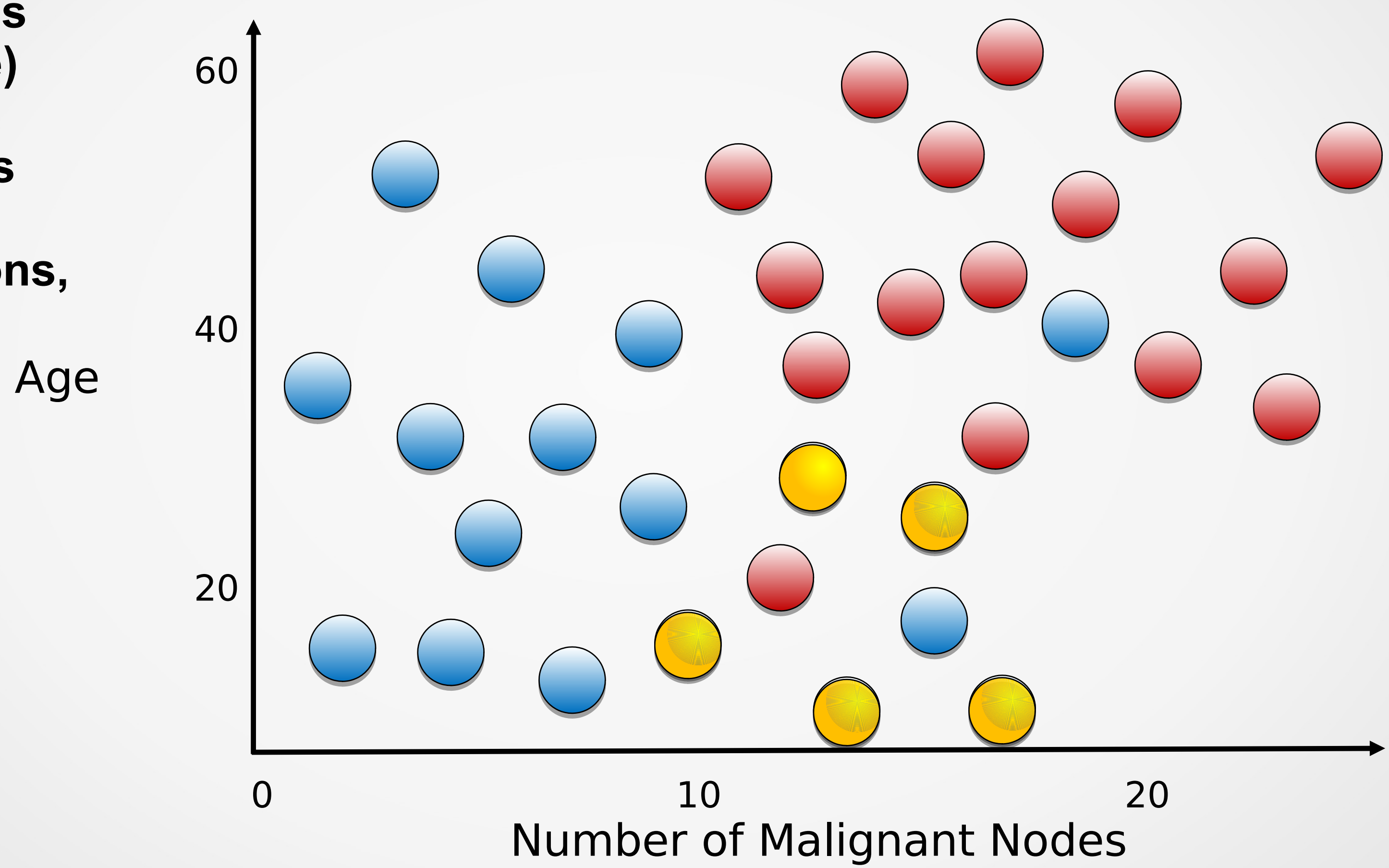
Two labels (survived, lost)



Classification with Logistic Regression

Two features (nodes, age)

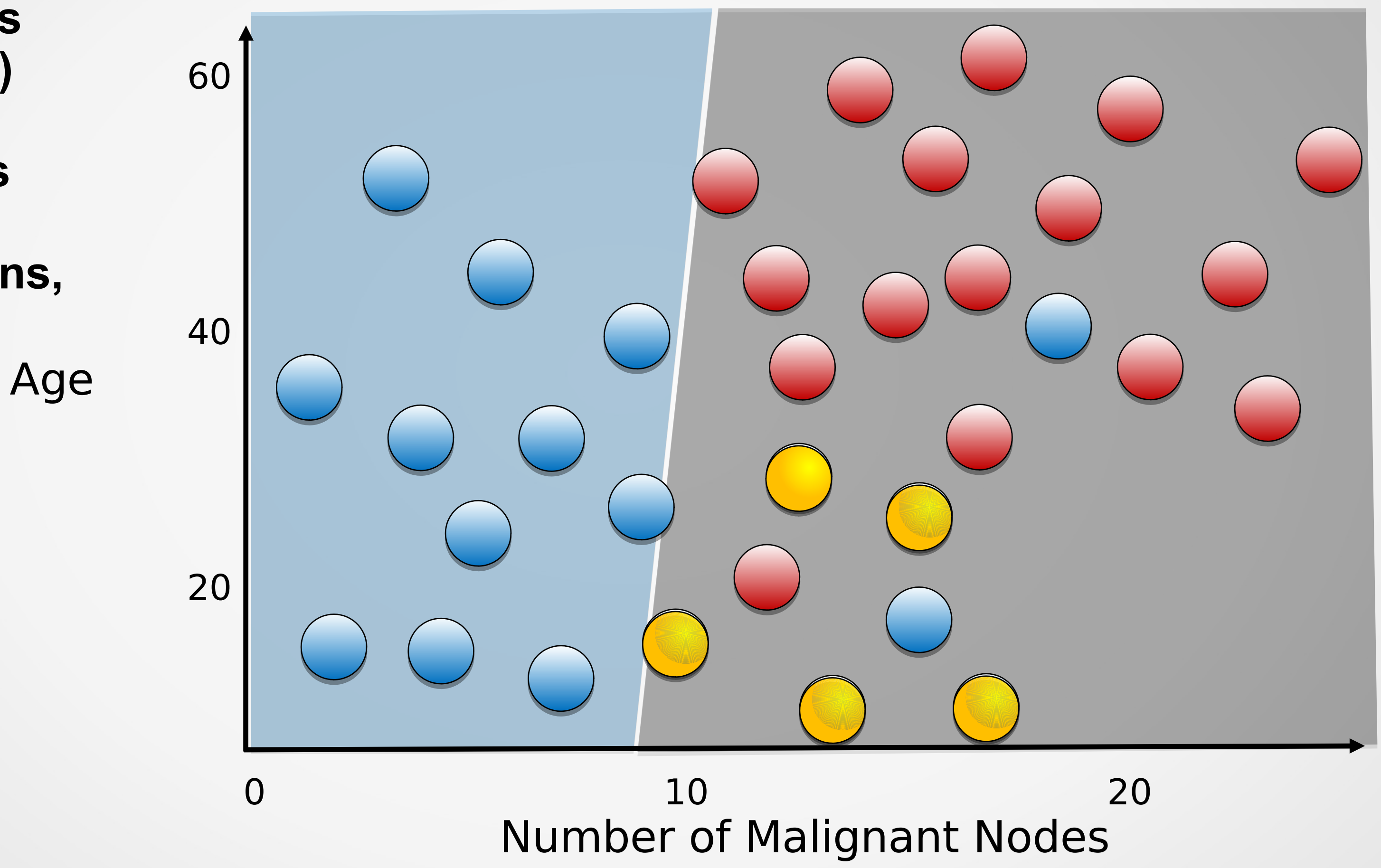
Three labels (survived, complications, lost)



Classification with Logistic Regression

Two features (nodes, age)

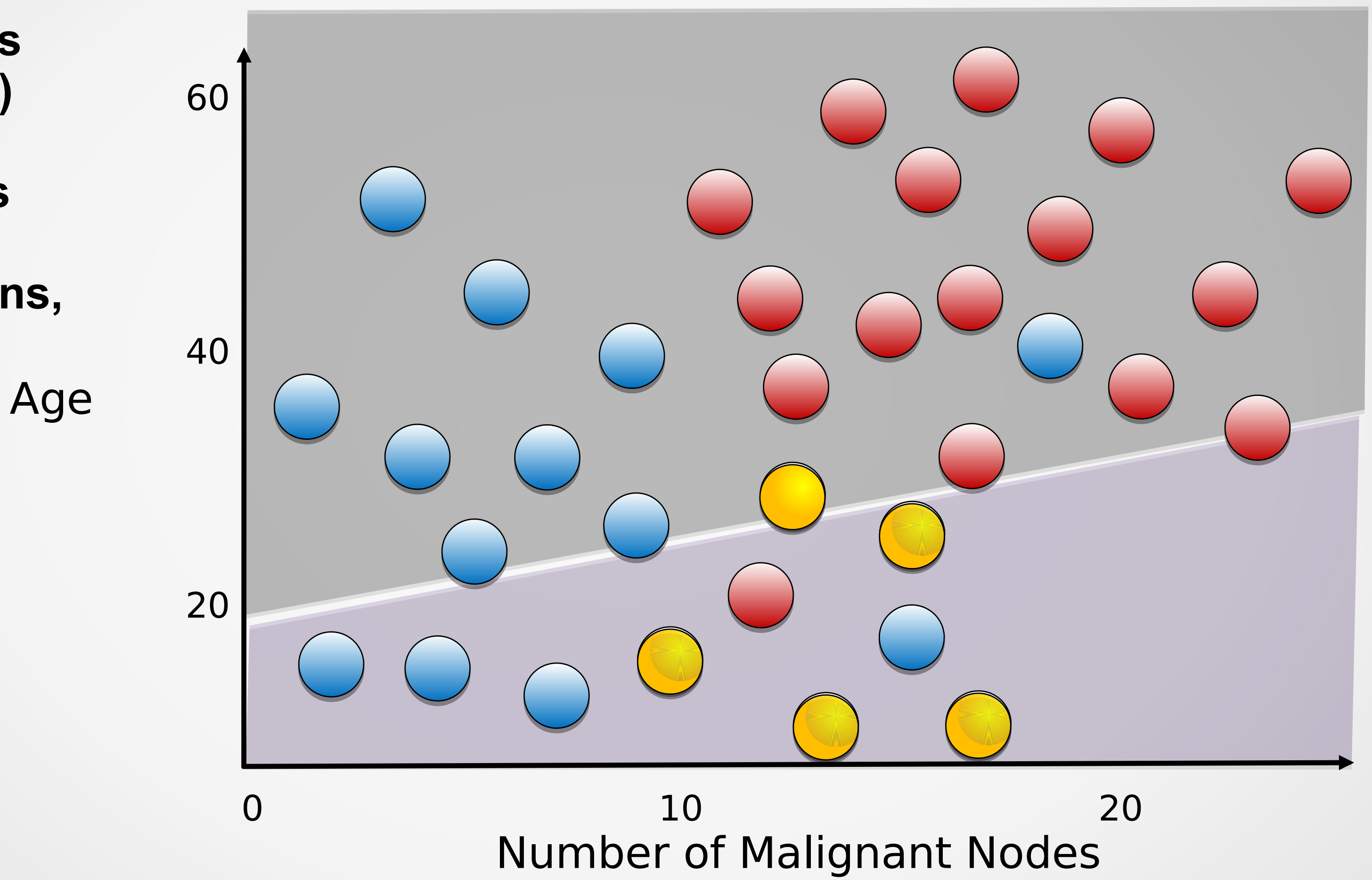
Three labels (survived, complications, lost)



Classification with Logistic Regression

Two features (nodes, age)

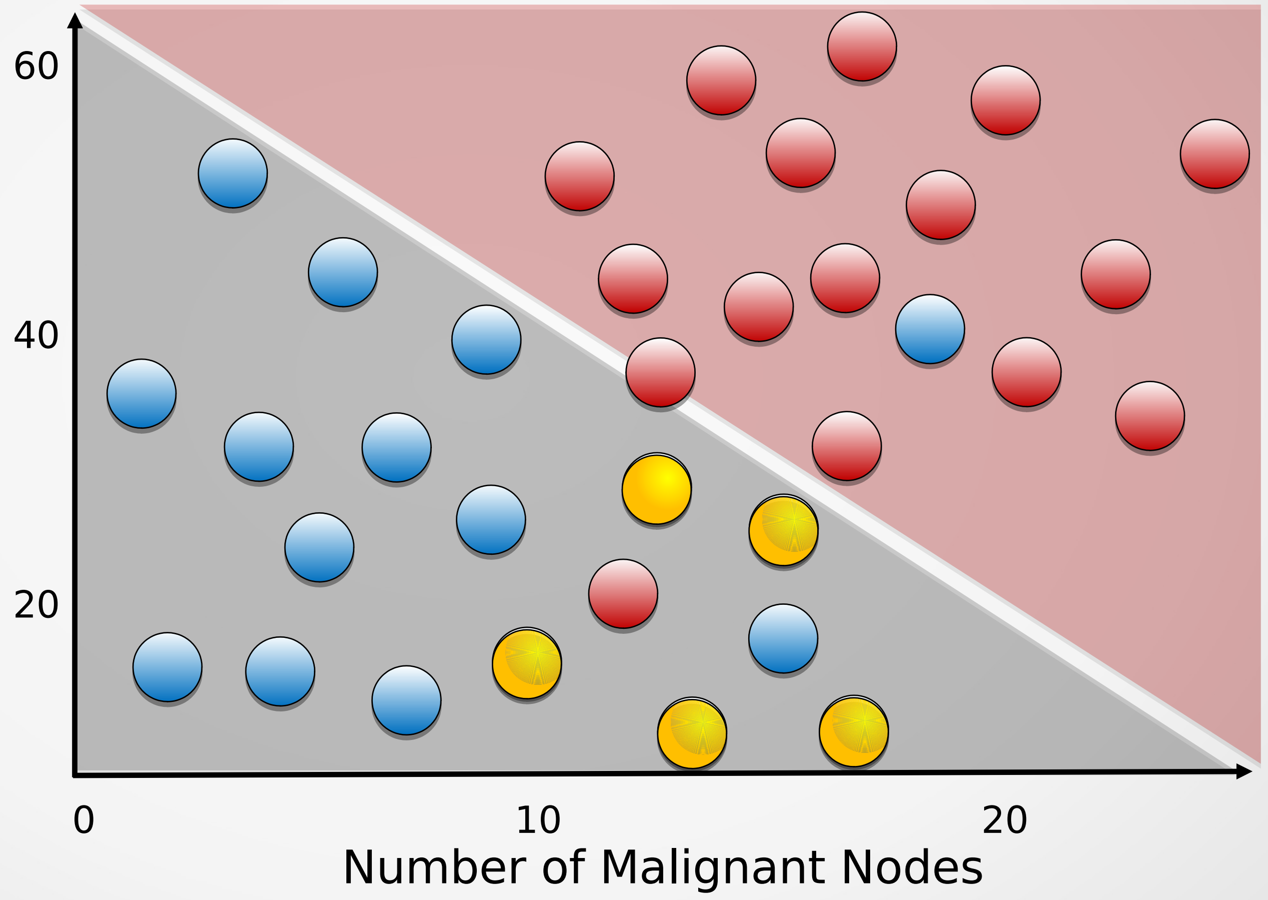
Three labels (survived, complications, lost)



Classification with Logistic Regression

Two features (nodes, age)

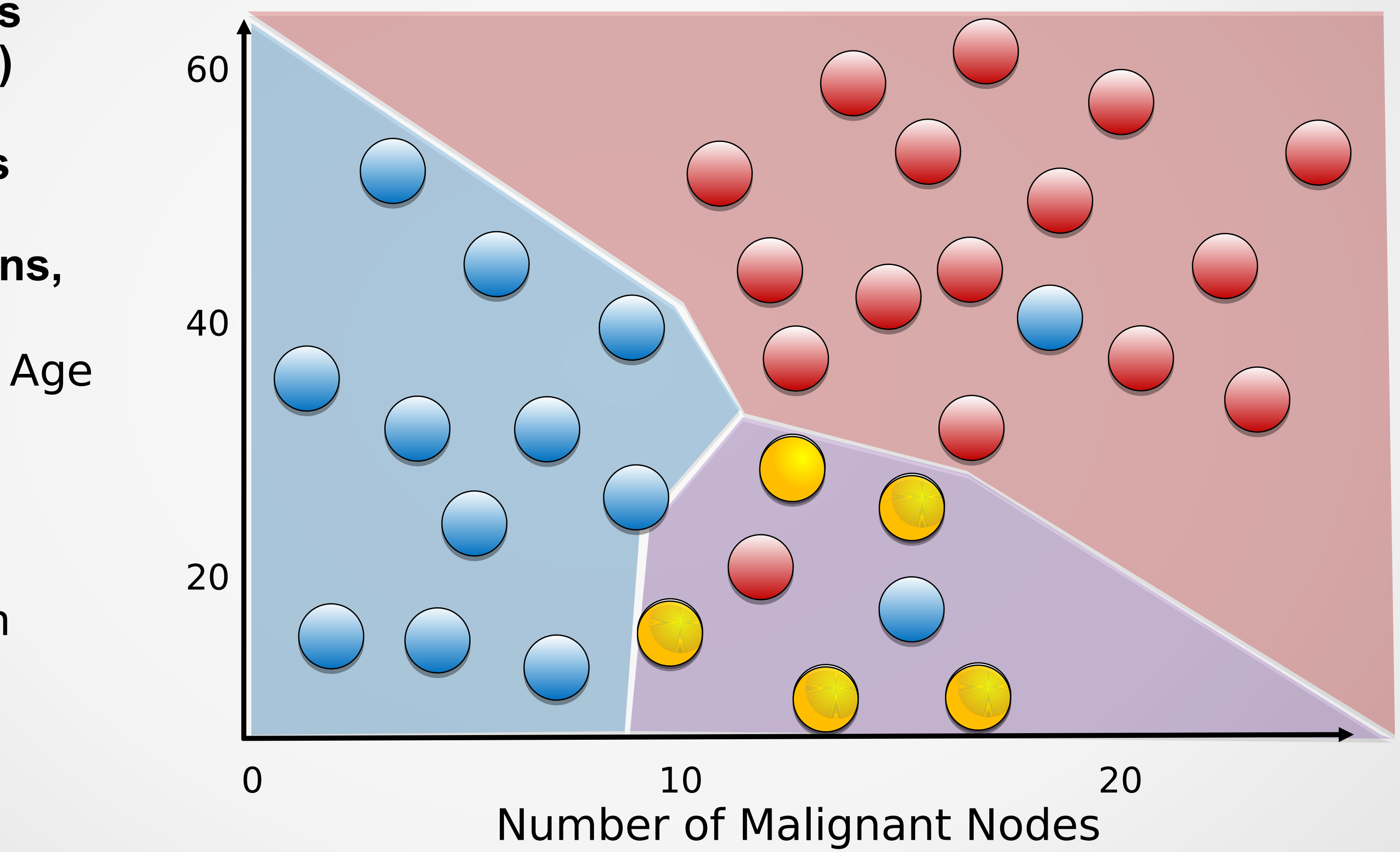
Three labels (survived, complications, lost)



Classification with Logistic Regression

Two features (nodes, age)

Three labels (survived, complications, lost)



Assign most probable class to each region

Logistic Regression 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 using `LogisticRegressionCV`

Classification Error Metrics

- Task: 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)

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)

$$\text{Recall (sensitivity)} = \frac{TP}{TP + FN}$$

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)

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

Confusion Matrix Error Measurements

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

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

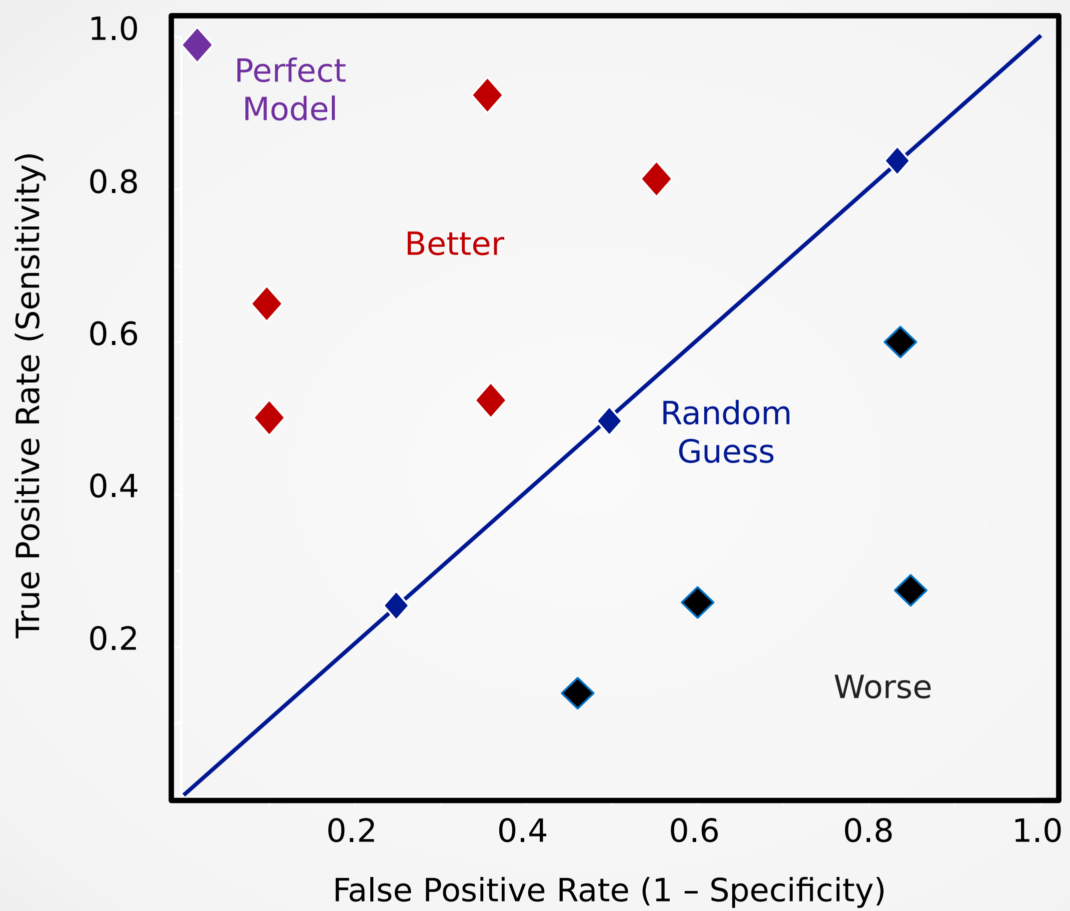
$$\text{Recall or Sensitivity} = \frac{TP}{TP + FN}$$

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

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

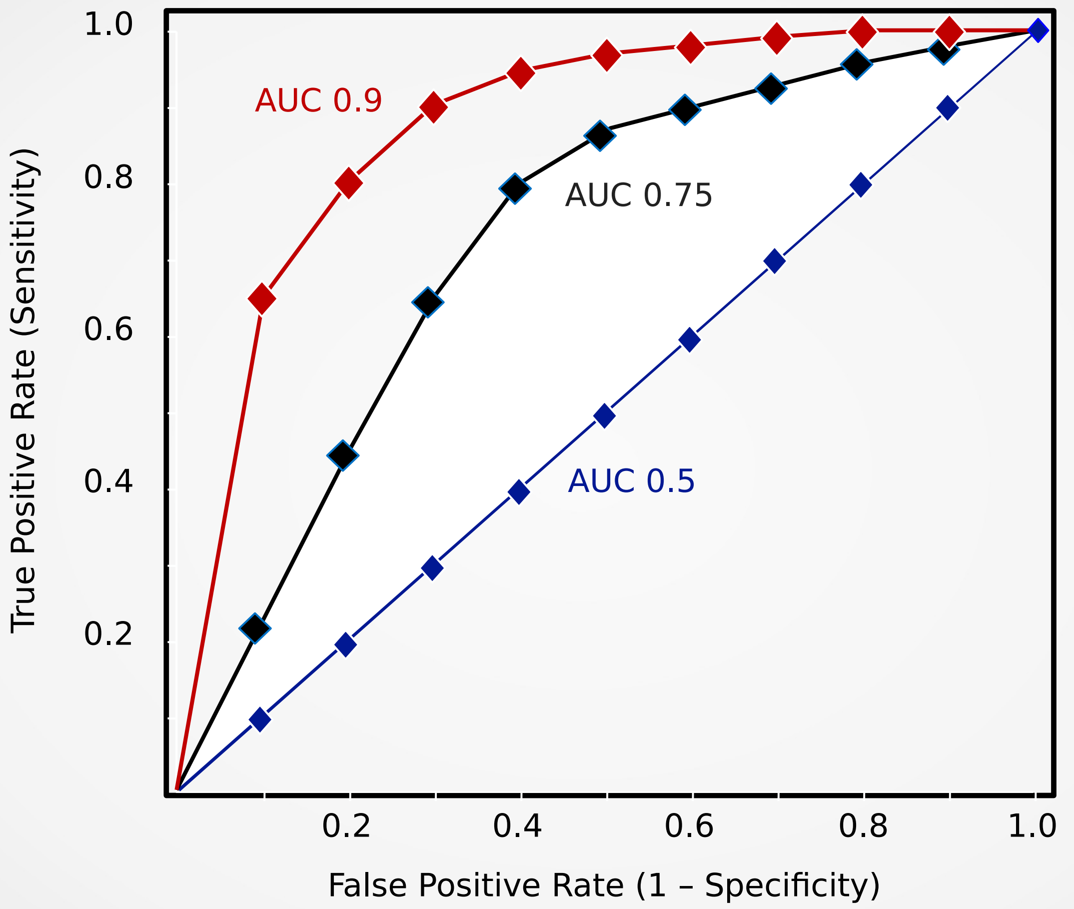
$$\text{Specificity} = \frac{TN}{FP + TN}$$

Receiver Operating Characteristic (ROC)



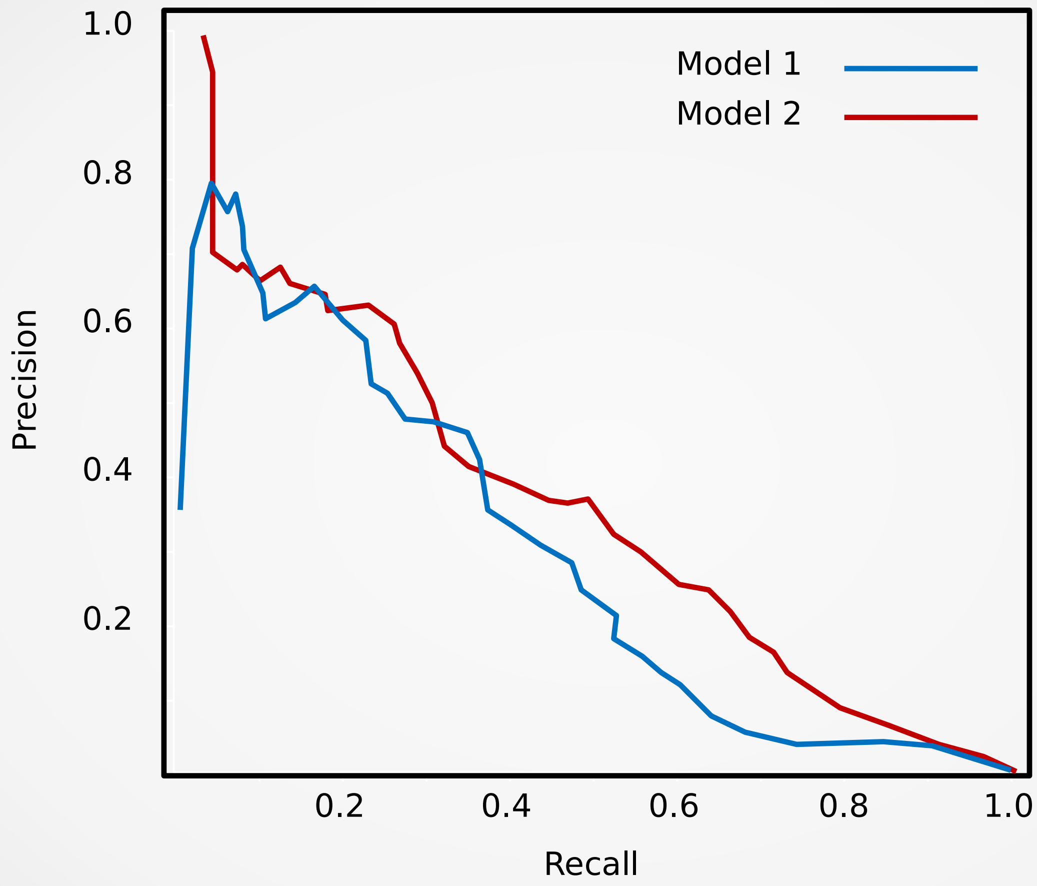
Evaluation of model at all possible thresholds

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

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

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

Classification Error Metrics 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)
```

End of Lecture

Many thanks to Intel Software for providing a variety of resources for this lecture series