

Software

---

# Logistic Regression

# Legal Notices and Disclaimers

This presentation is for informational purposes only. INTEL MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS SUMMARY.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. Check with your system manufacturer or retailer or learn more at [intel.com](https://www.intel.com).

This sample source code is released under the [Intel Sample Source Code License Agreement](#).

Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries.

\*Other names and brands may be claimed as the property of others.

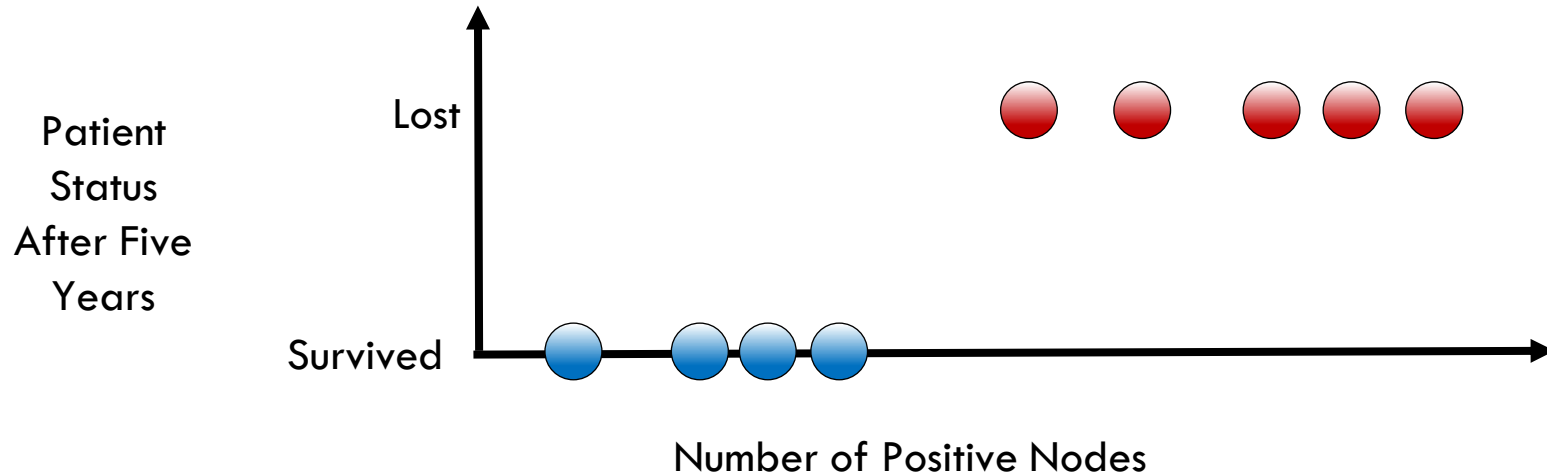
Copyright © 2021, Intel Corporation. All rights reserved.

# Learning Objectives

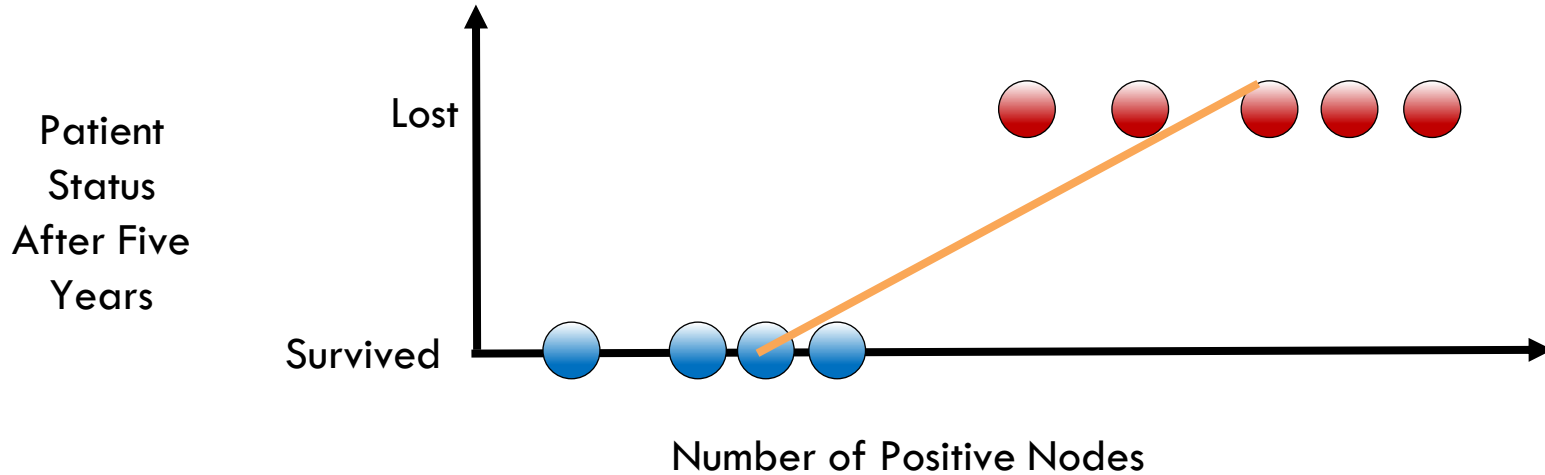
- Describe Logistic regression and how it differs from linear regression
- Identify metrics for classification errors and scenarios in which they can be used
- Apply Intel® Extension for Scikit-learn\* to leverage underlying compute capabilities of hardware

•

# Introduction to Logistic Regression

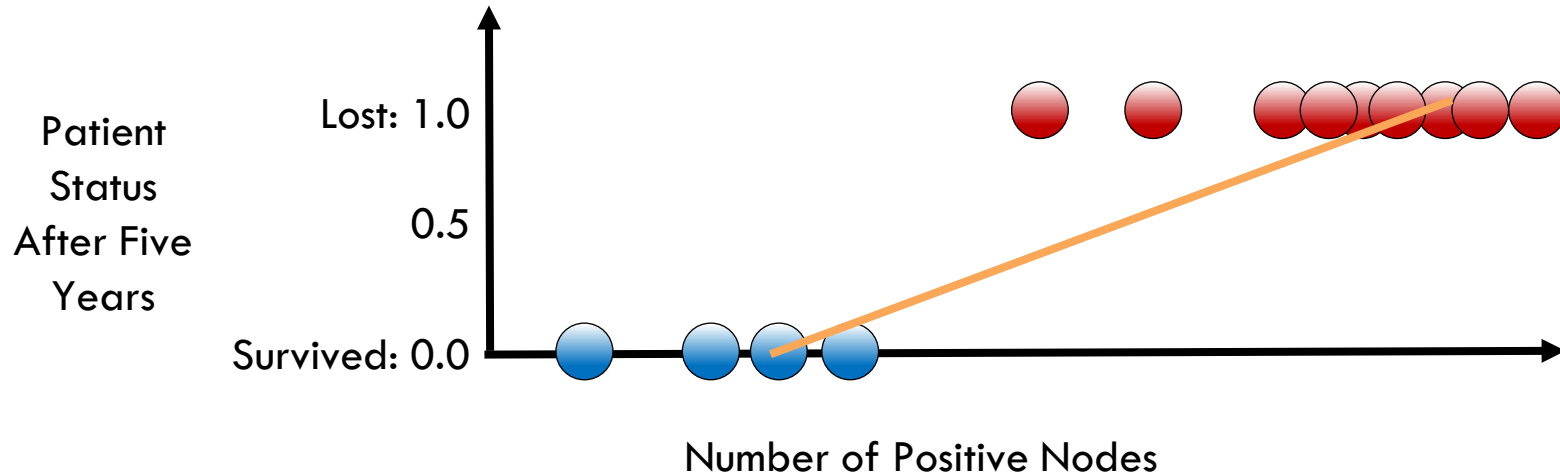


# Linear Regression for Classification?



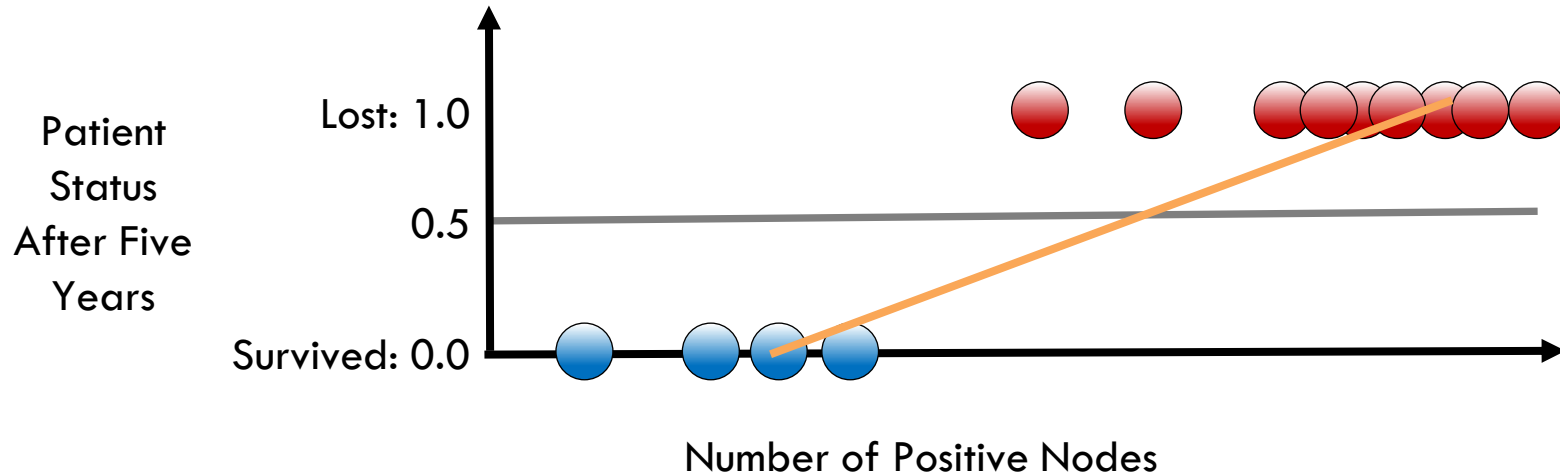
$$y_{\beta}(x) = \beta_0 + \beta_1 x + \varepsilon$$

# Linear Regression for Classification?



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

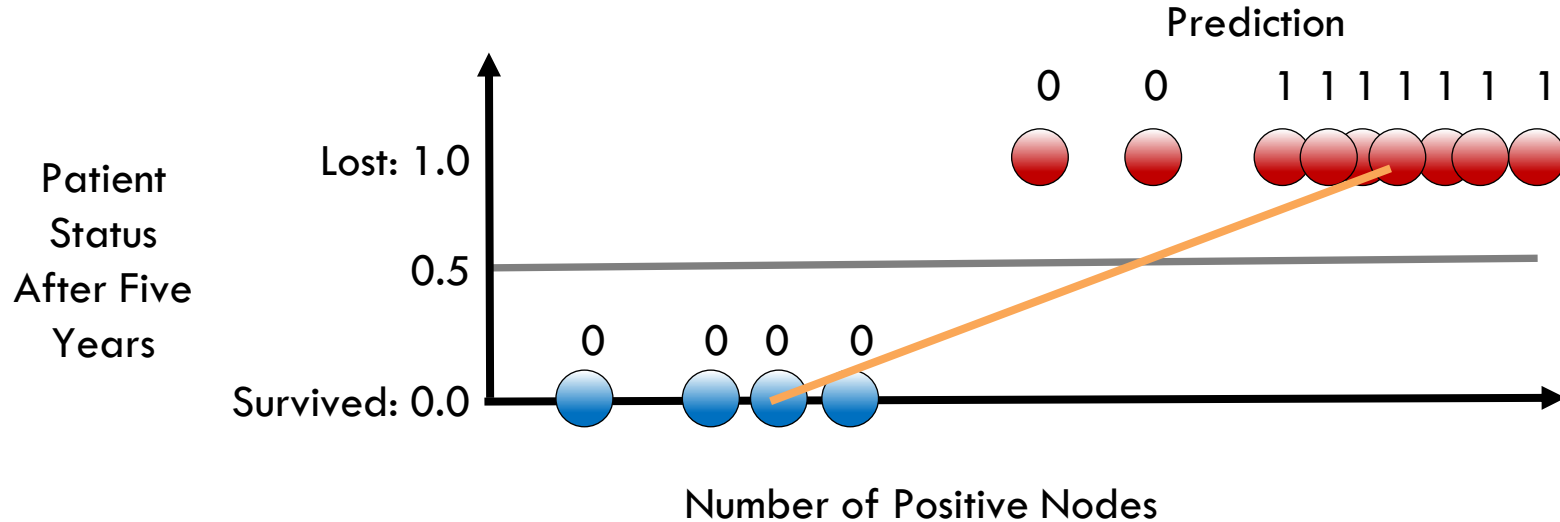
# Linear Regression for Classification?



If model result  $> 0.5$ : predict lost

If model result  $< 0.5$ : predict survived

# Linear Regression for Classification?

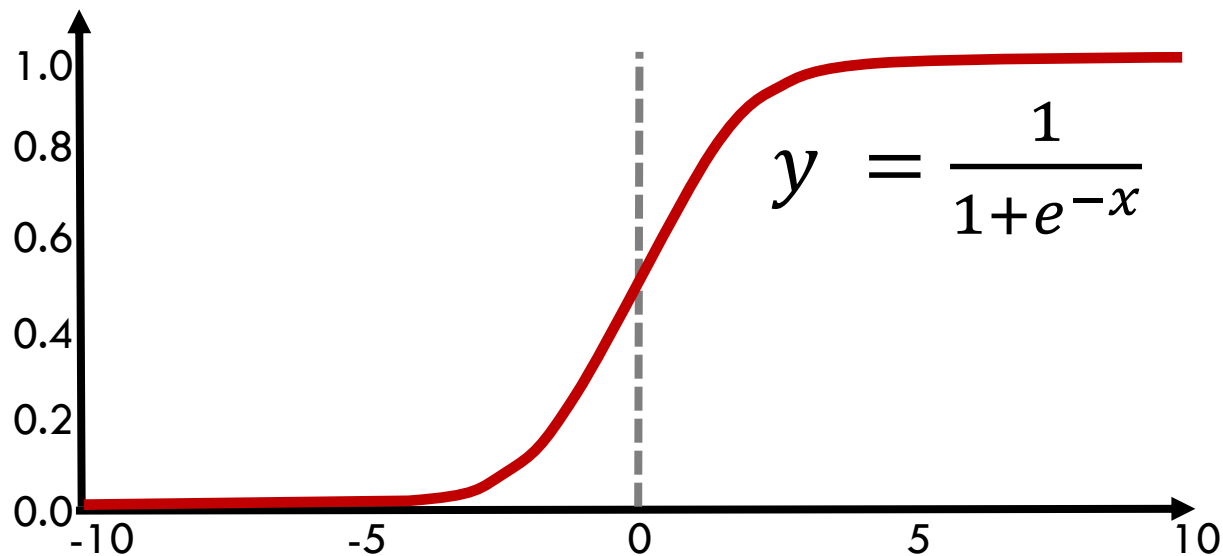


If model result  $> 0.5$ : predict lost

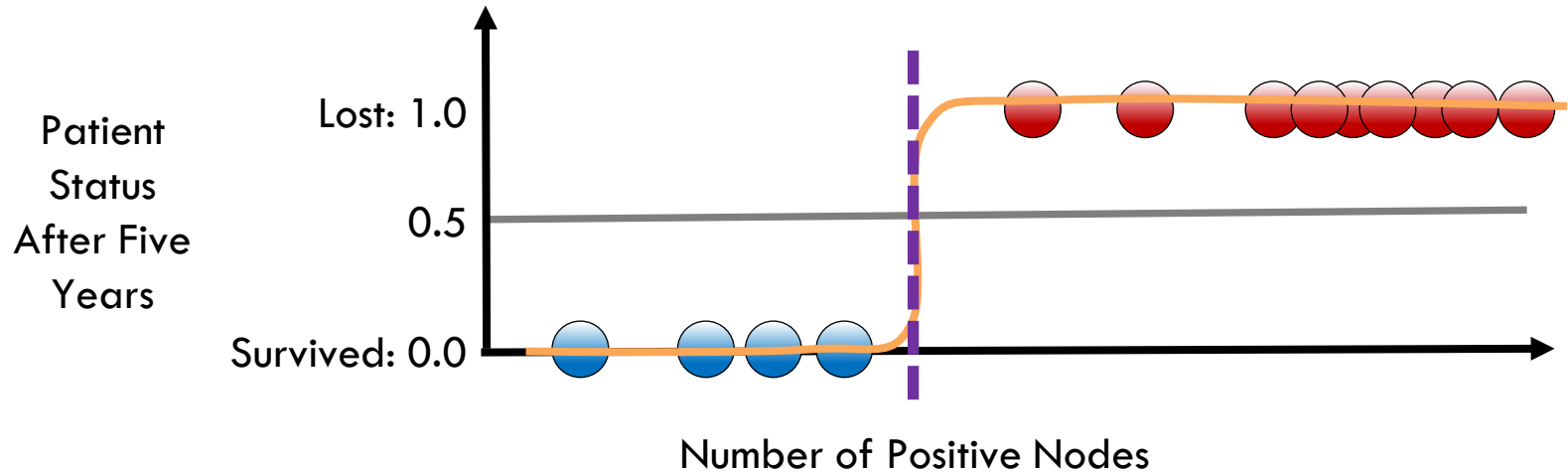
If model result  $< 0.5$ : predict survived



# What is this Function?

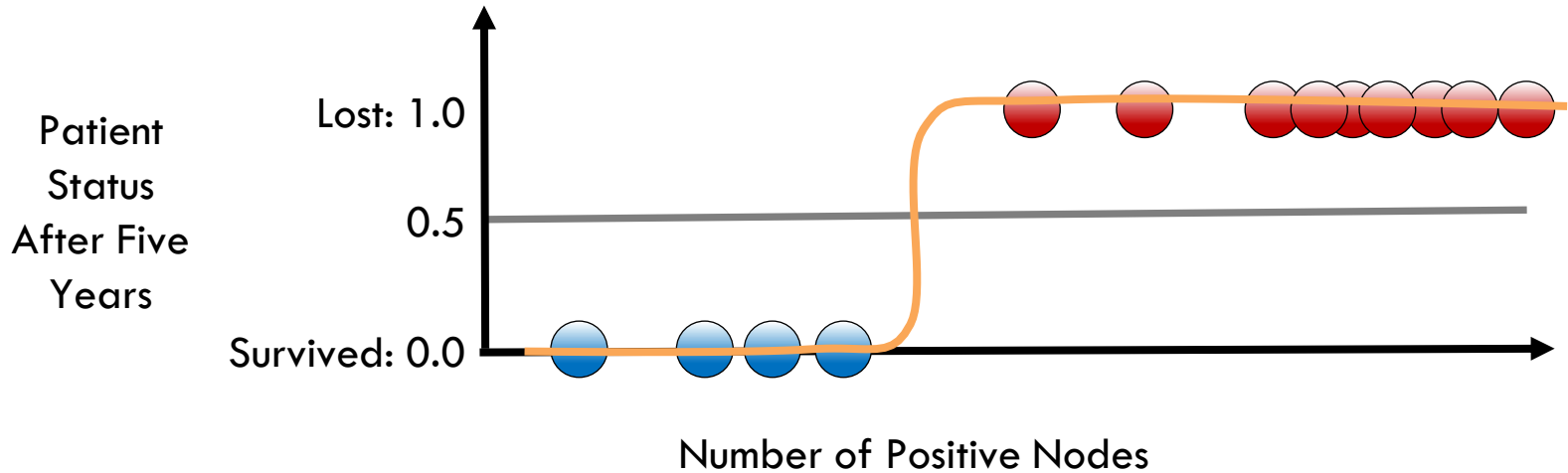


# The Decision Boundary



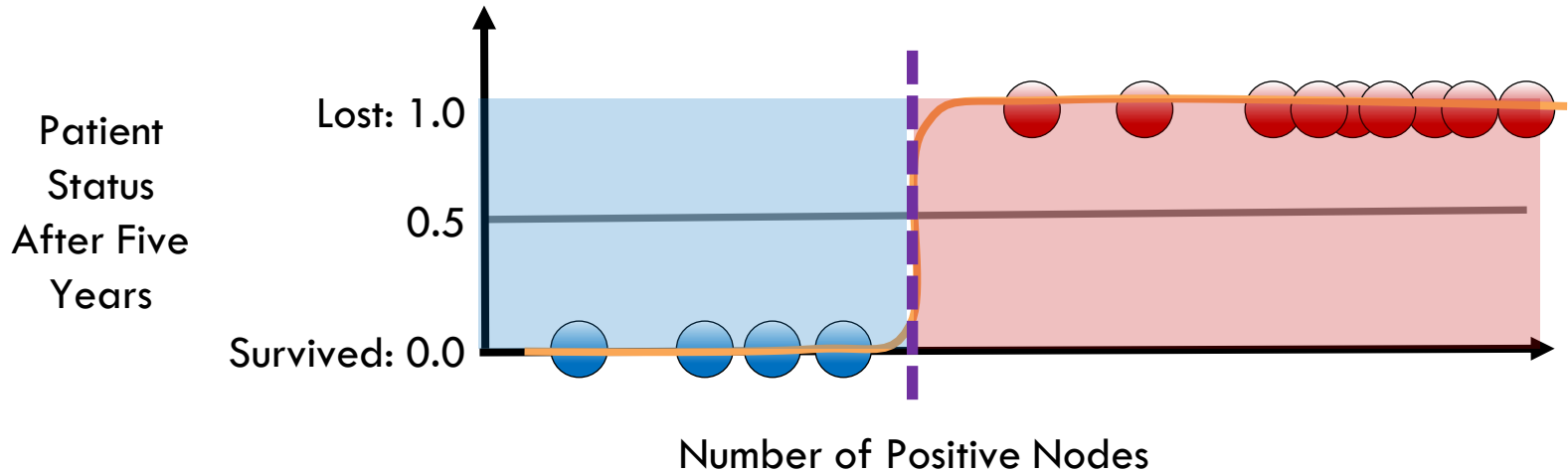
$$y_{\beta}(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x + \varepsilon)}}$$

# 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

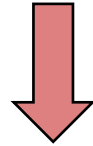
Logistic  
Function

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

# 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

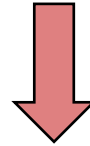
$$\frac{P(x)}{1 - P(x)} = e^{(\beta_0 + \beta_1 x)}$$



# Relationship of Logistic to Linear Regression

Logistic  
Function

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



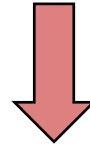
Log  
Odds

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

# Relationship of Logistic to Linear Regression

Logistic  
Function

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



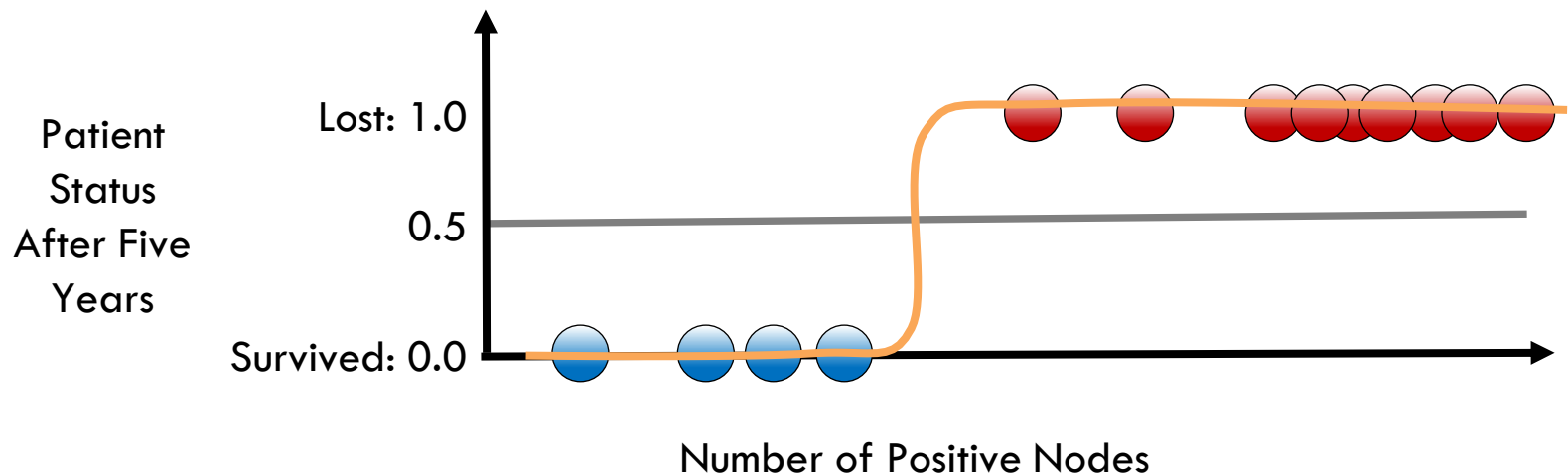
Log  
Odds

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

# Classification with Logistic Regression

One feature (nodes)

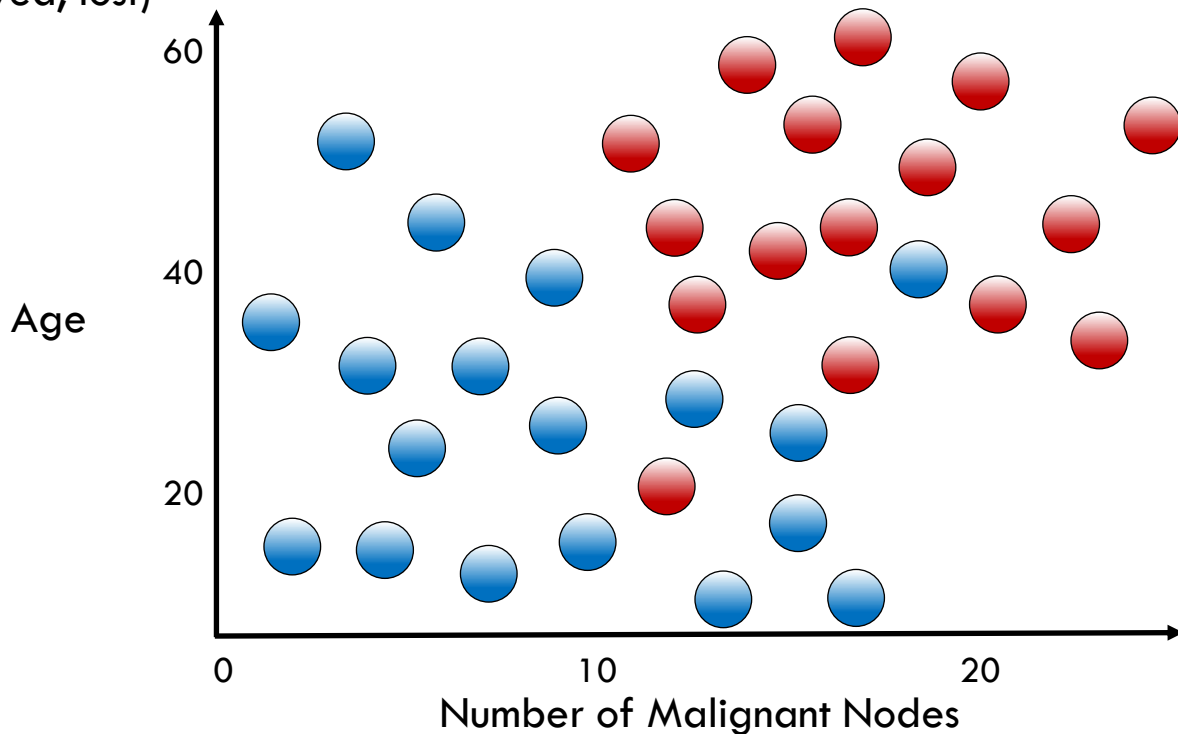
Two labels (survived, lost)



# Classification with Logistic Regression

Two features (nodes, age)

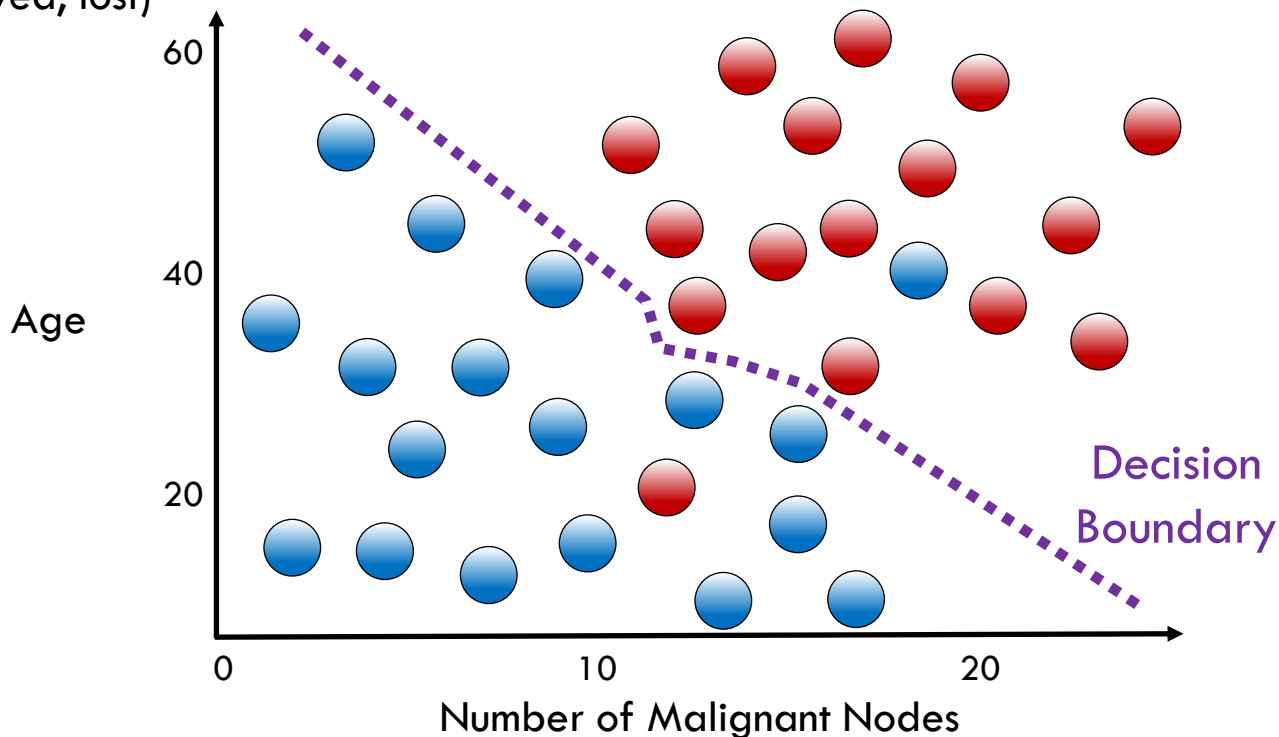
Two labels (survived, lost)



# Classification with Logistic Regression

Two features (nodes, age)

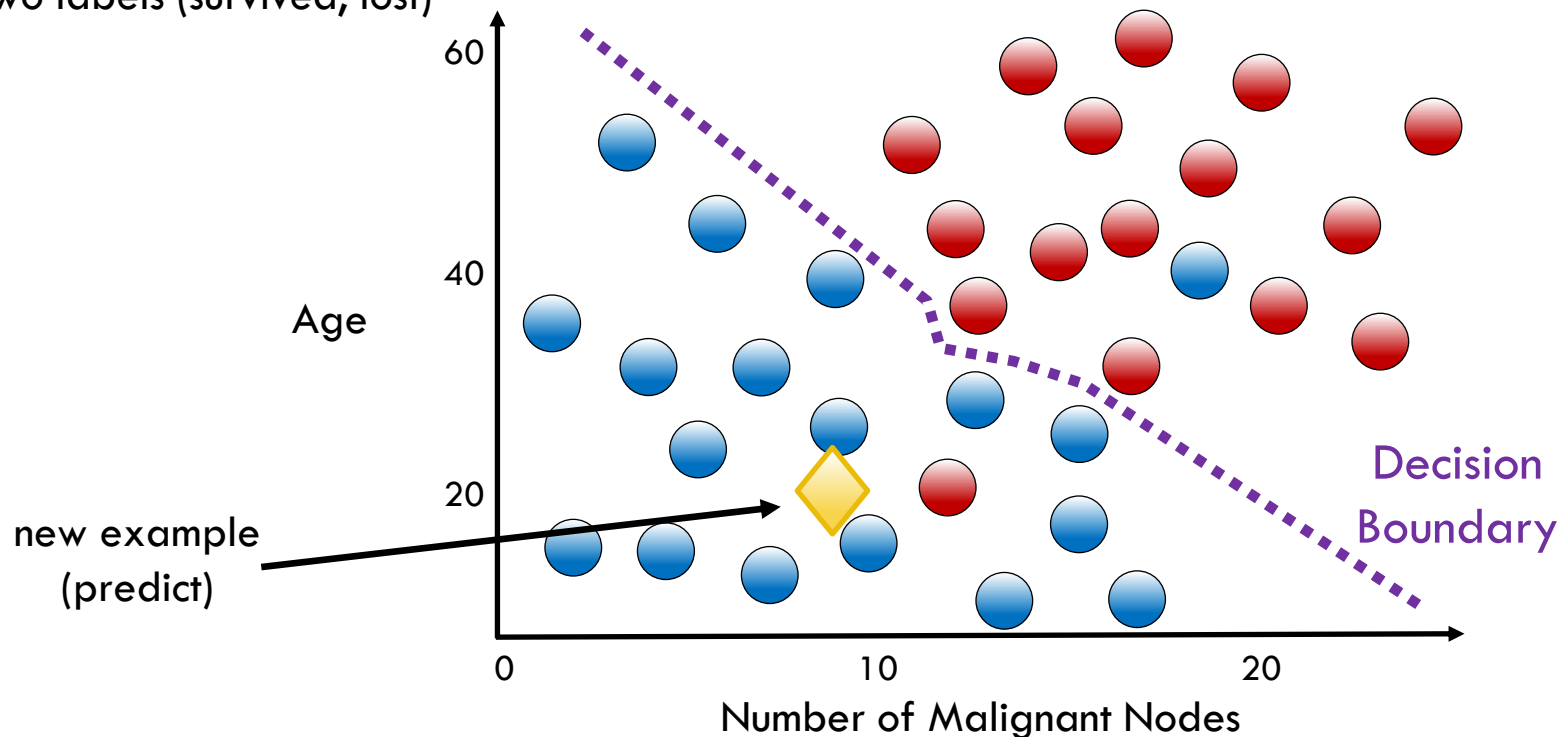
Two labels (survived, lost)



# Classification with Logistic Regression

Two features (nodes, age)

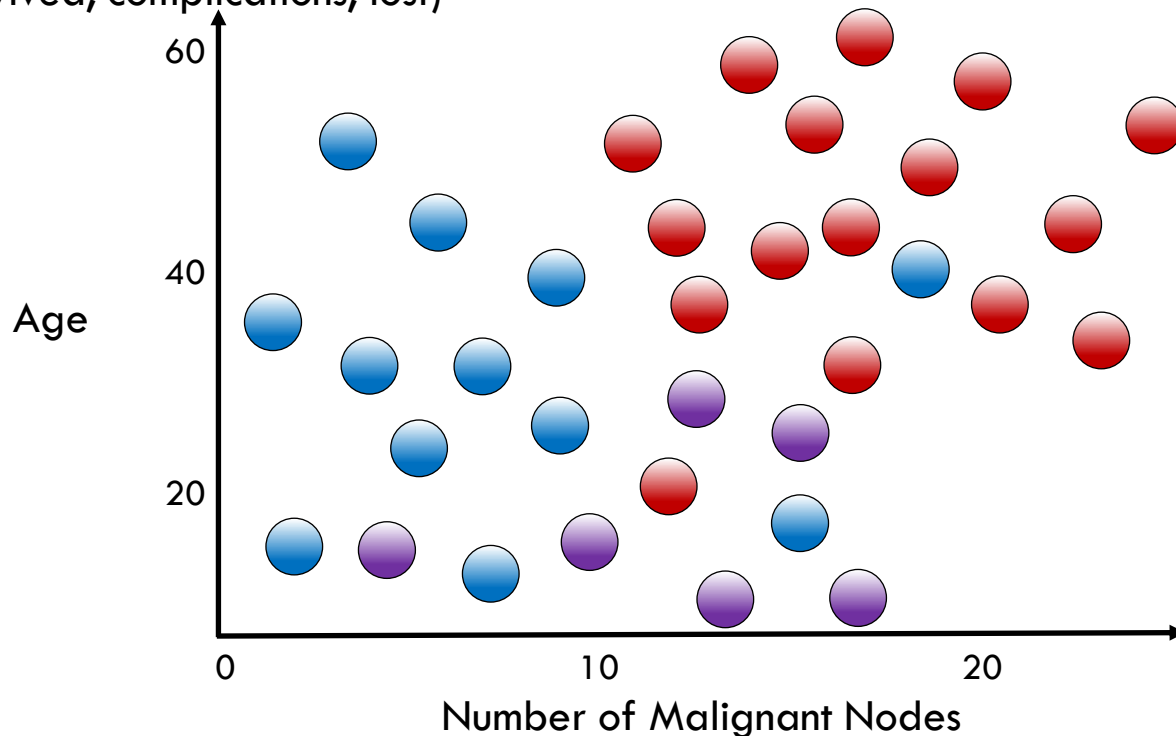
Two labels (survived, lost)



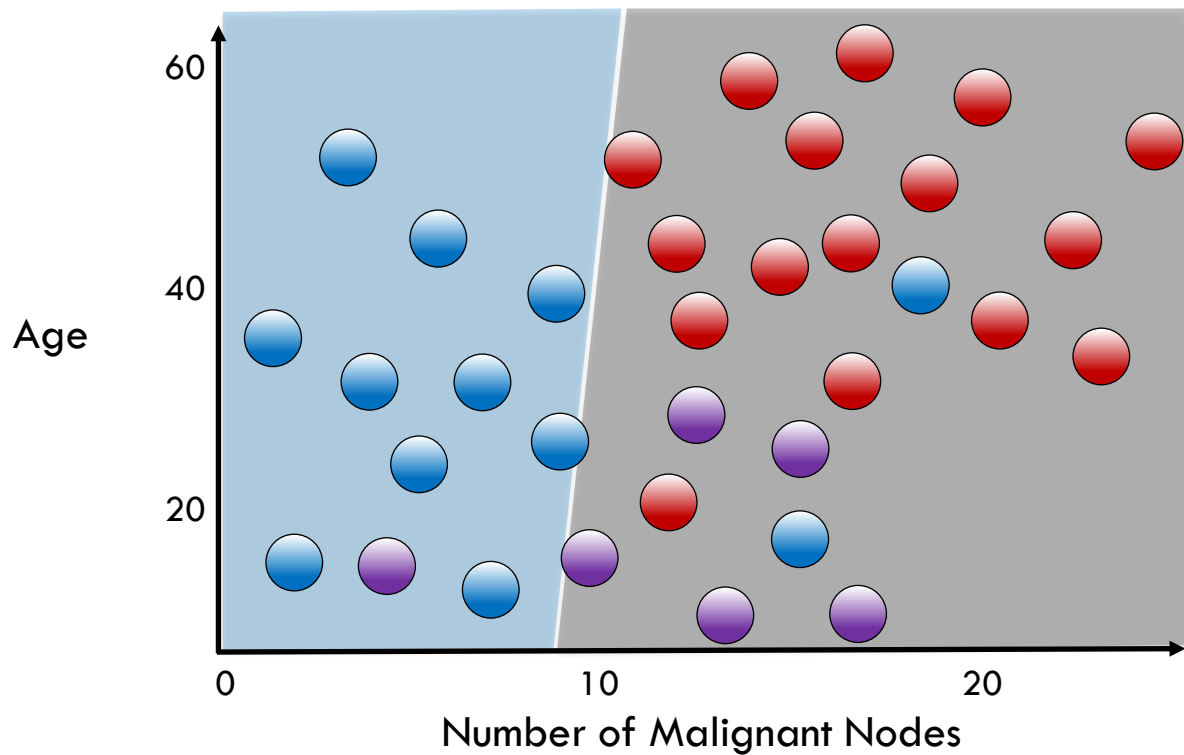
# Multiclass Classification with Logistic Regression

Two features (nodes, age)

Three labels (survived, complications, lost)

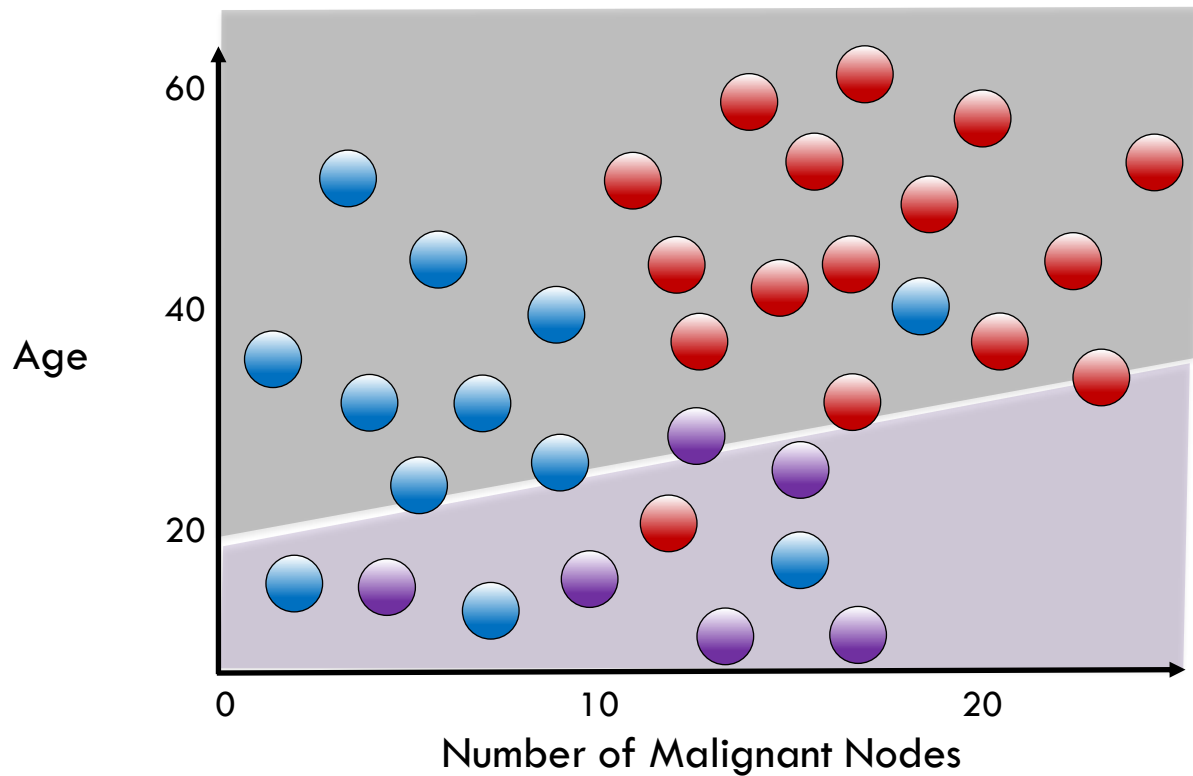


# One vs All: **Survived** vs All

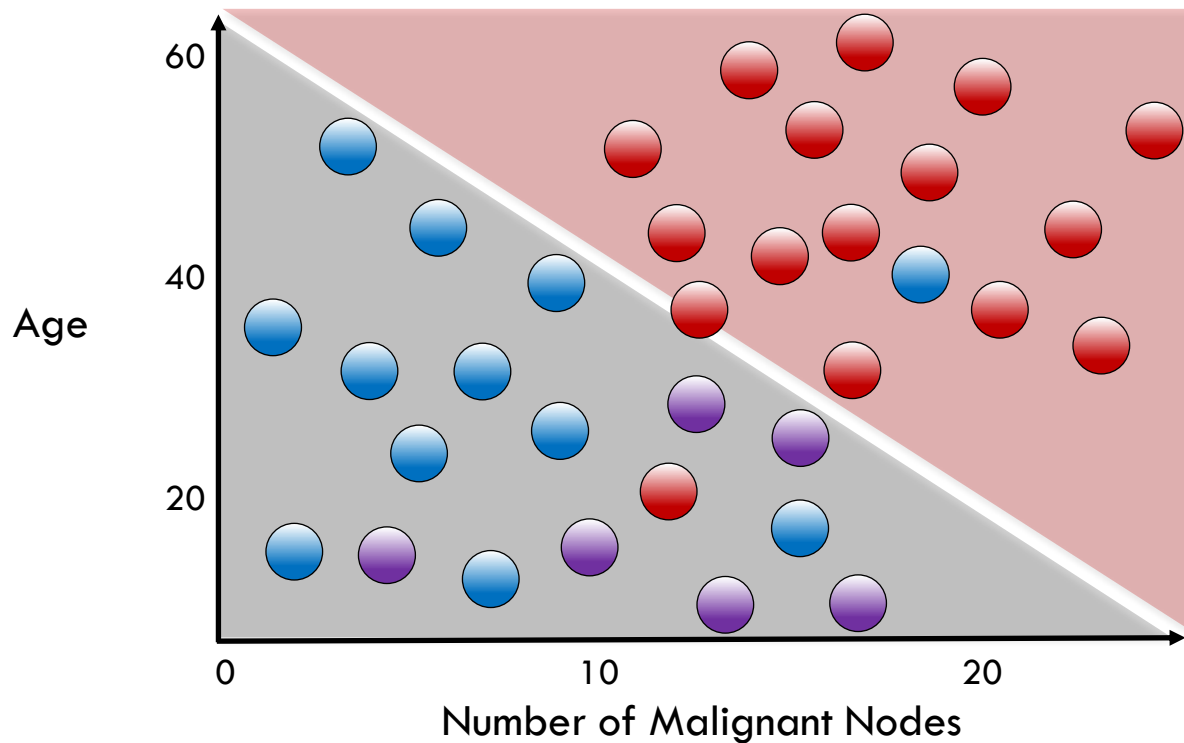




# One vs All: Complications vs All

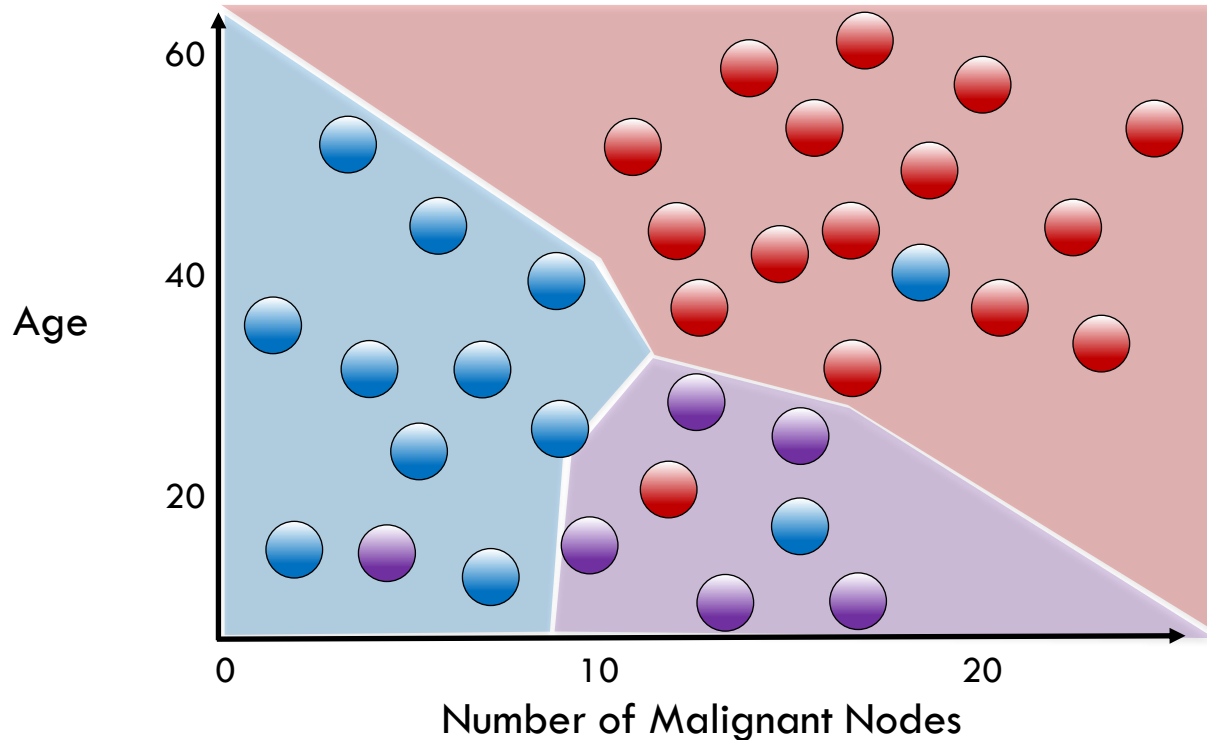


# One vs All: **Loss** vs All



# 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
```

To use the Intel® Extension for Scikit-learn\* variant of this algorithm:

- Install Intel® oneAPI AI Analytics Toolkit (AI Kit)
- Add the following two lines of code after the code above:

```
from sklearnex import patch_sklearn
patch_sklearn()
```

# 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**

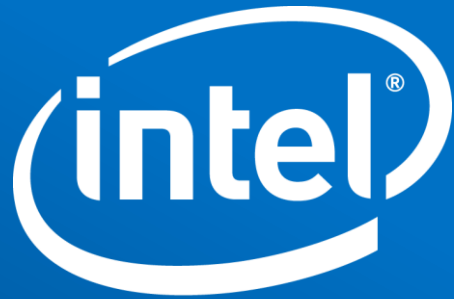
```
LR = LogisticRegression(penalty='l2', c=10.0)
```

**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**.**



Software



Software

---

# 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) ← Type II Error	
Actual Negative	False Positive (FP) ↑ Type I Error	True Negative (TN)	

# Accuracy: Predicting Correctly

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

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



# 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)

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

# 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)

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

# Specificity: Avoiding False Alarms

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

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

# Error Measurements

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

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

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

# Error Measurements

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

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

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

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

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

# Error Measurements

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

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

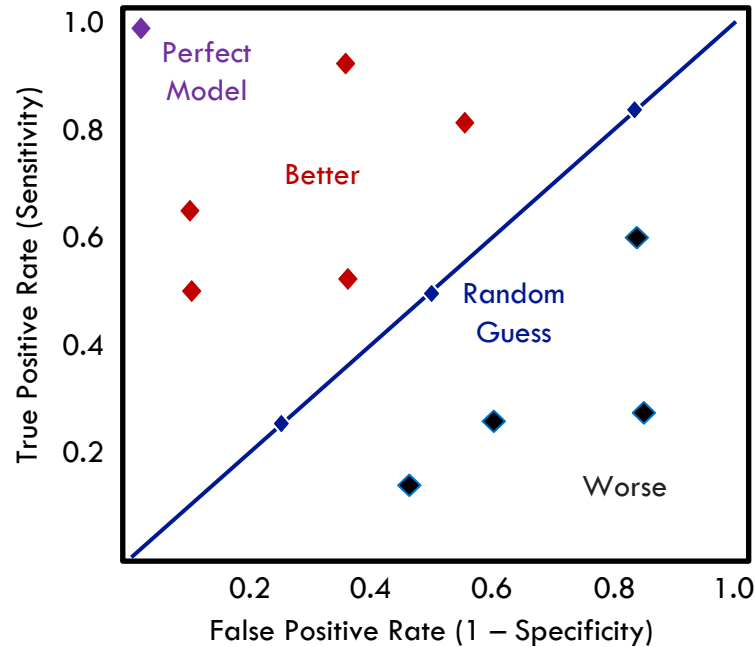
$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

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

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

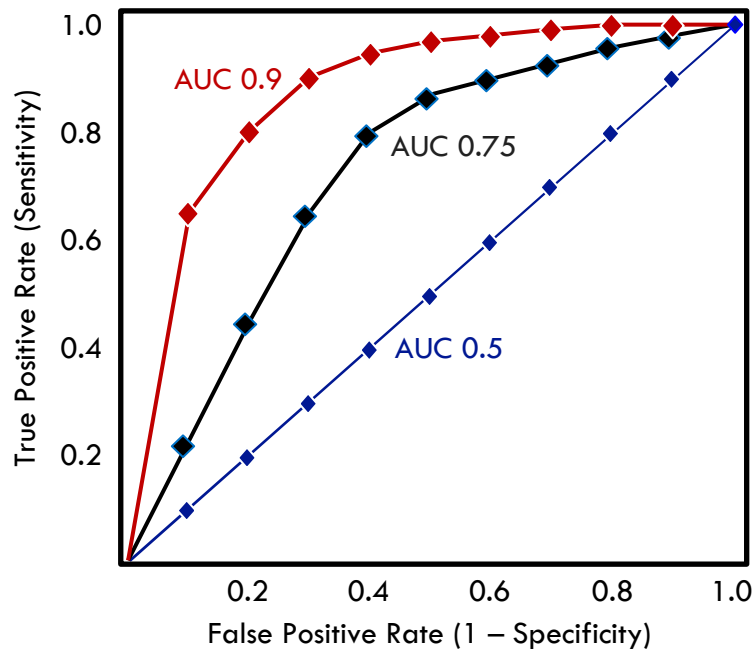
$$\text{F1} = 2 \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

# Receiver Operating Characteristic (ROC)



Evaluation of model at all possible thresholds

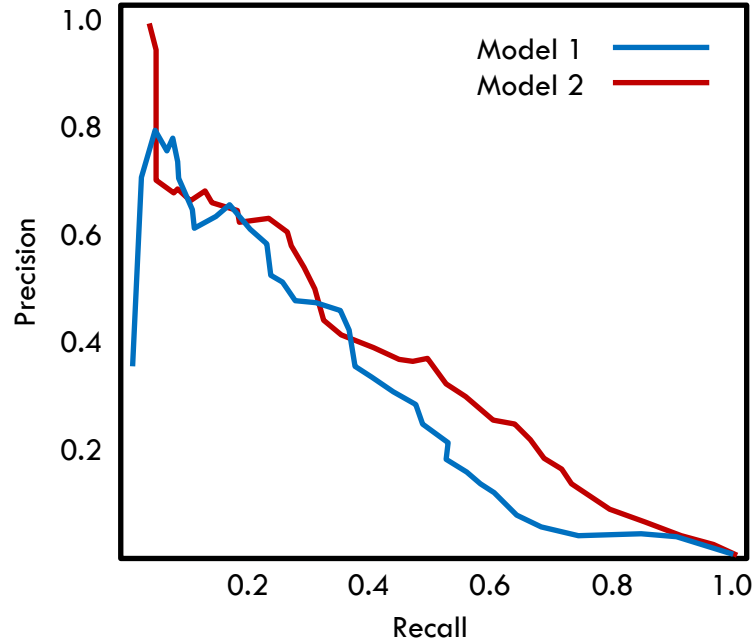
# Area Under Curve (AUC)



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

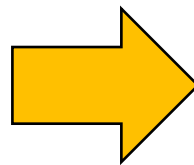
	Predicted Class 1	Predicted Class 2	Predicted Class 3
Actual Class 1	TP1		
Actual Class 2		TP2	
Actual Class 3			TP3

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

# 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

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



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
```

# 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**

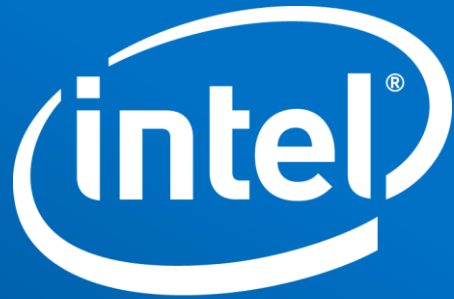
```
from sklearn.metrics import accuracy_score
```

**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
```



Software