



2

Aprendizado Supervisionado



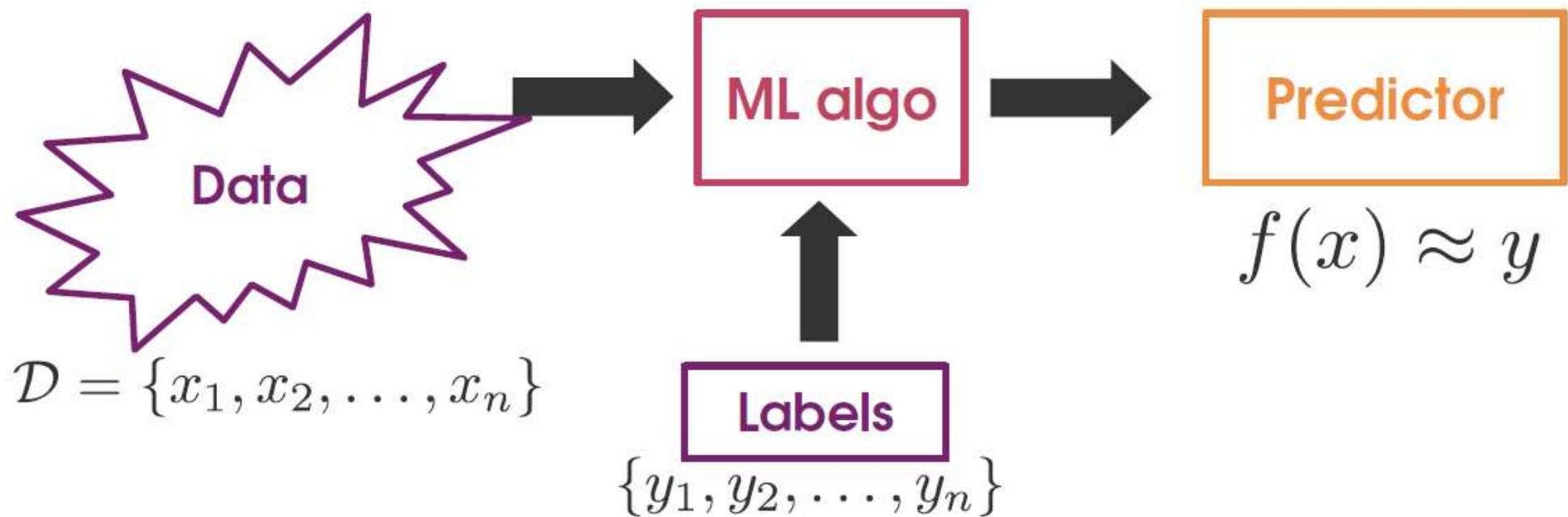
2.1. Definição do Problema



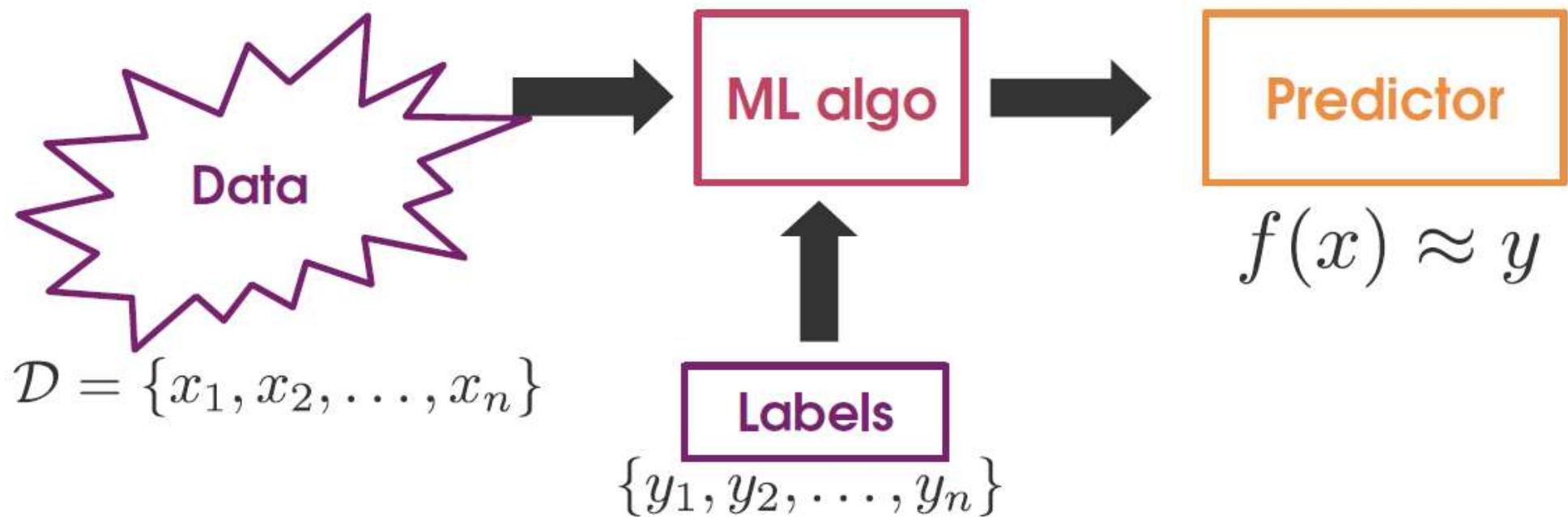
2.1. Definição do Problema

- Dado um conjunto S composto por n amostras $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$, sendo cada amostra $\mathbf{x}_i \in \mathbb{R}^m$ descrita por m atributos $x_{i1}, x_{i2}, \dots, x_{im}$ e acompanhada de um rótulo y_i , obter um modelo que determine, com a melhor acurácia possível, o valor do rótulo \hat{y}_i de uma nova amostra $\mathbf{x}_i = \{ \hat{x}_{i1}, \hat{x}_{i2}, \dots, \hat{x}_{im} \}$ não pertencente a S .

2.1. Definição do Problema



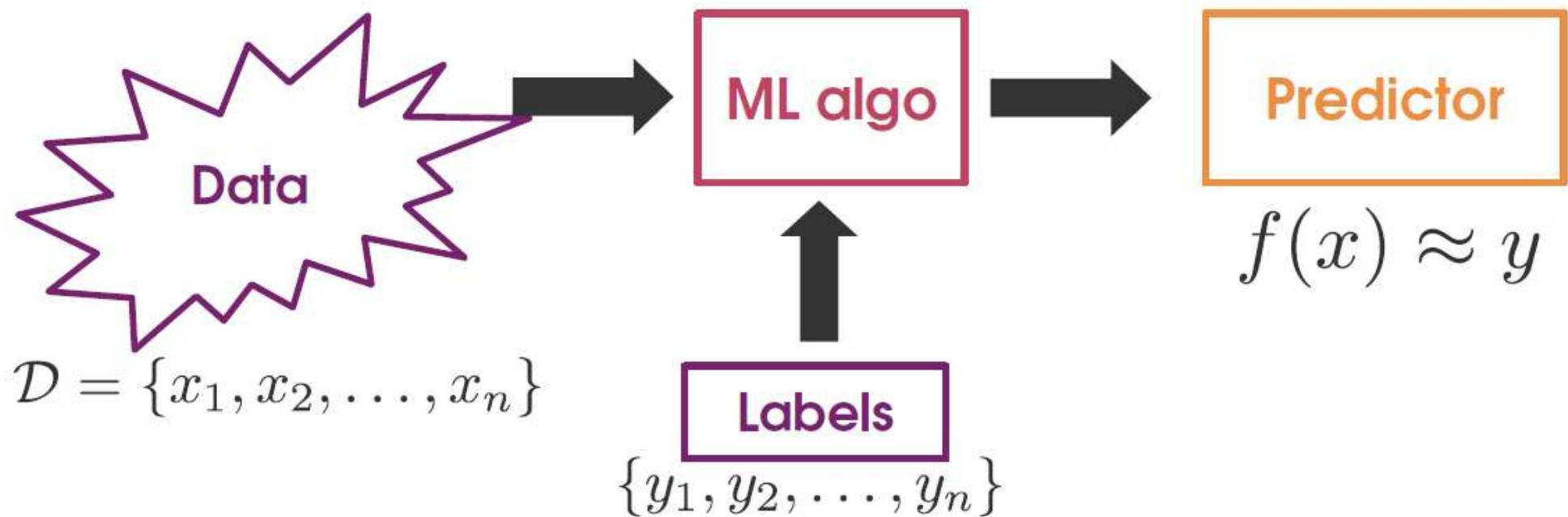
2.1. Definição do Problema



Binary classification

$$y_i \in \{0, 1\}$$

2.1. Definição do Problema



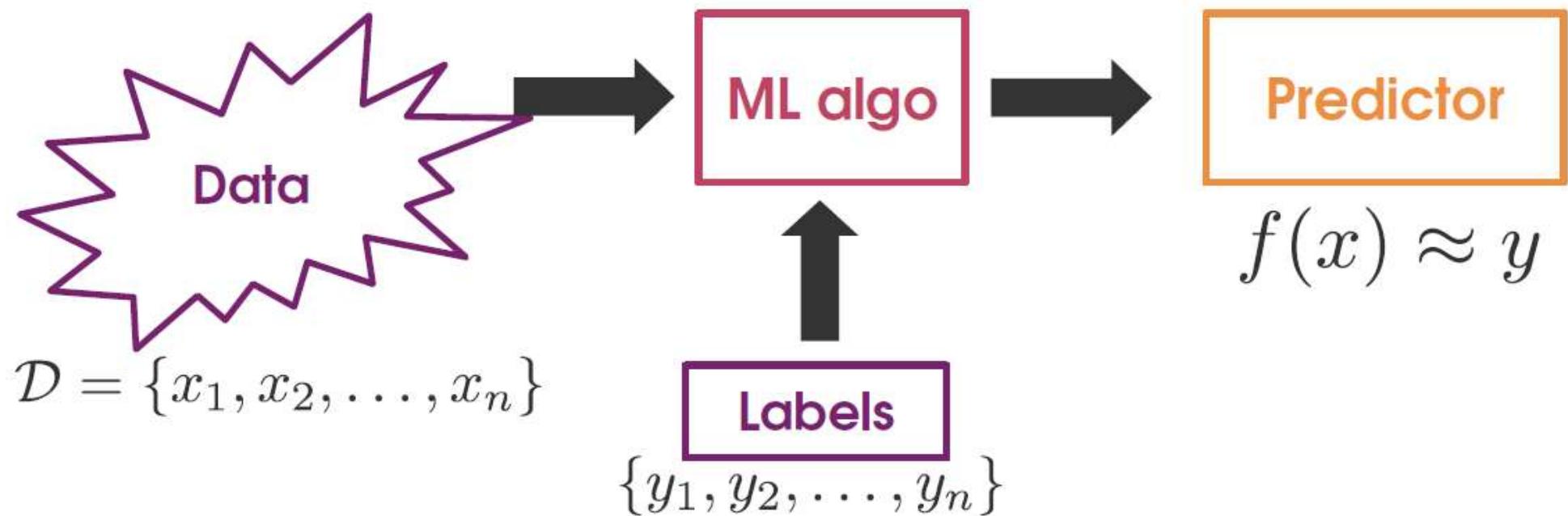
Binary classification

$$y_i \in \{0, 1\}$$

Multi-class classification

$$y_i \in \{0, 1, \dots, k\}$$

2.1. Definição do Problema



Binary classification

$$y_i \in \{0, 1\}$$

Multi-class classification

$$y_i \in \{0, 1, \dots, k\}$$

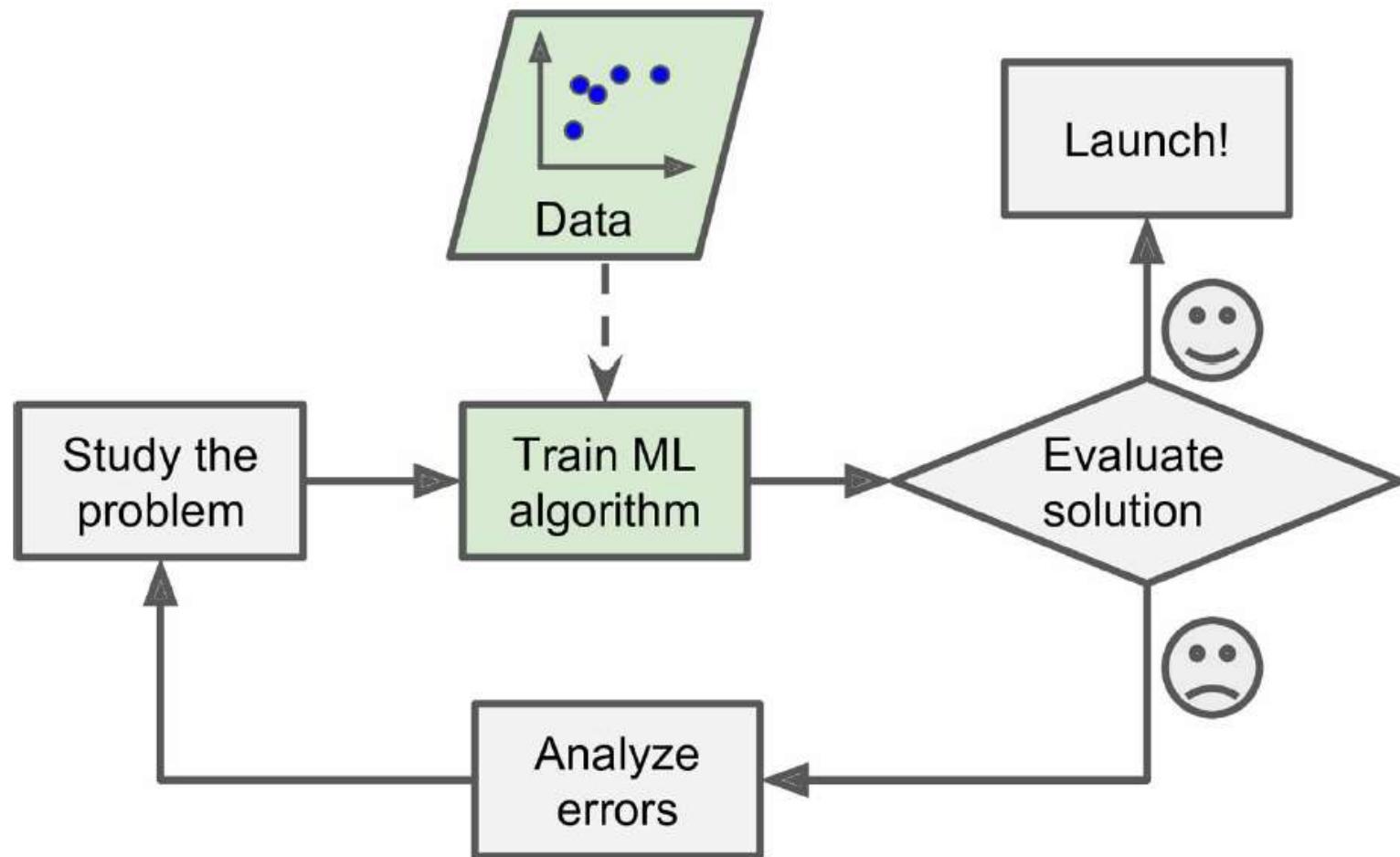
Regression

$$y_i \in \mathbb{R}$$

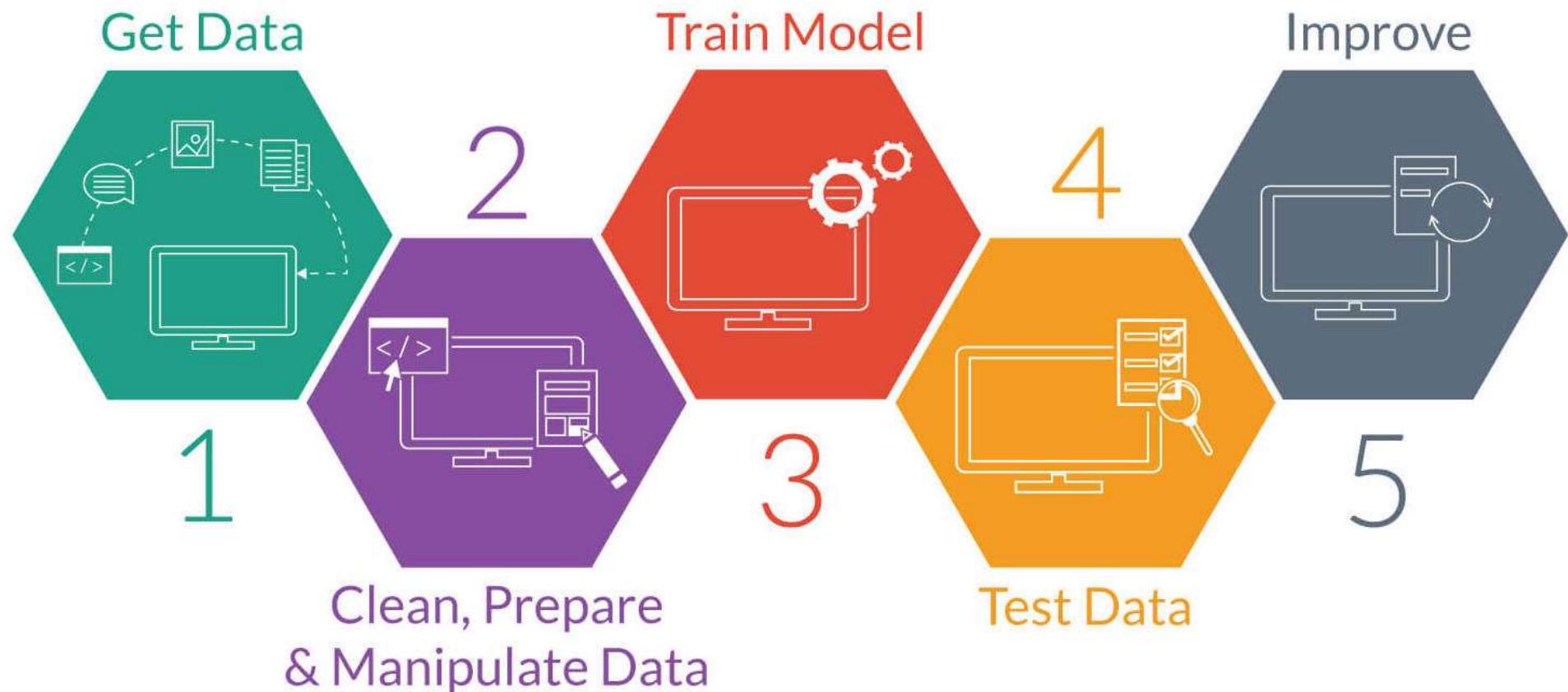


2.2. Workflow do Aprendizado Supervisionado

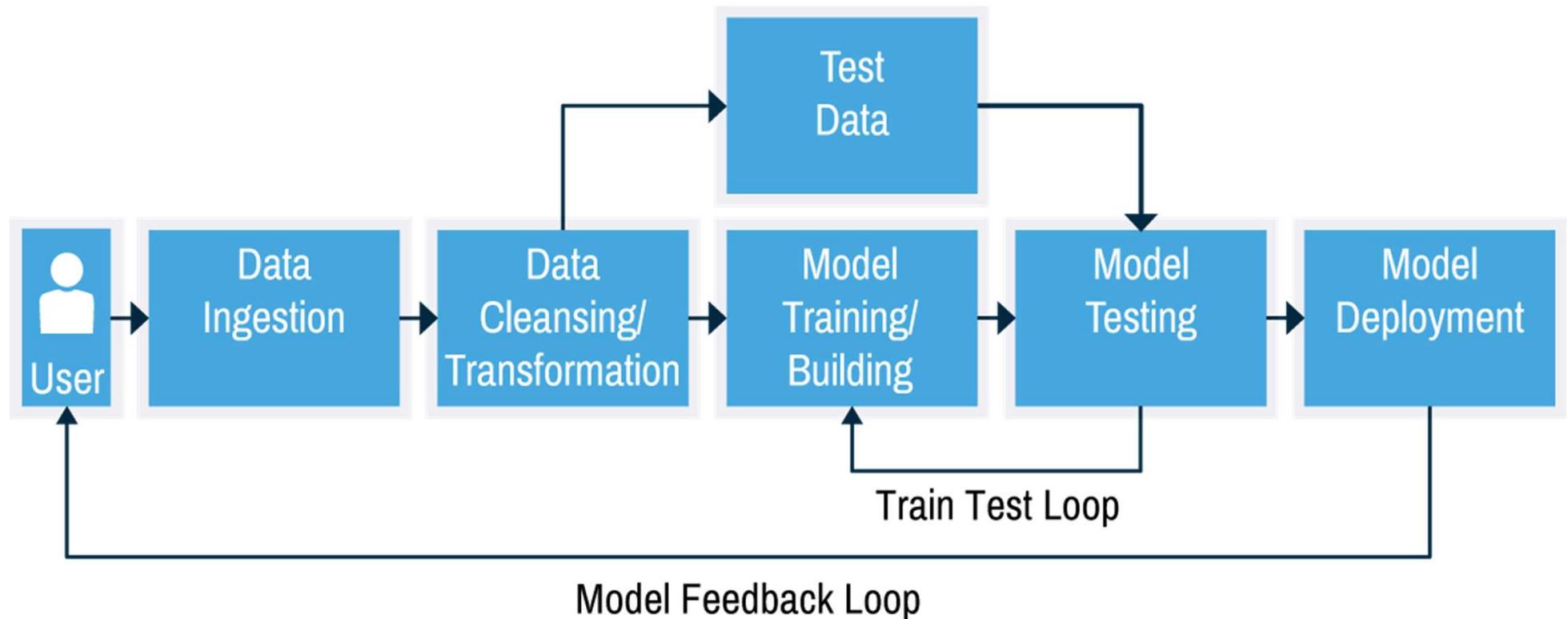
2.2. Workflow do Aprendizado Supervisionado



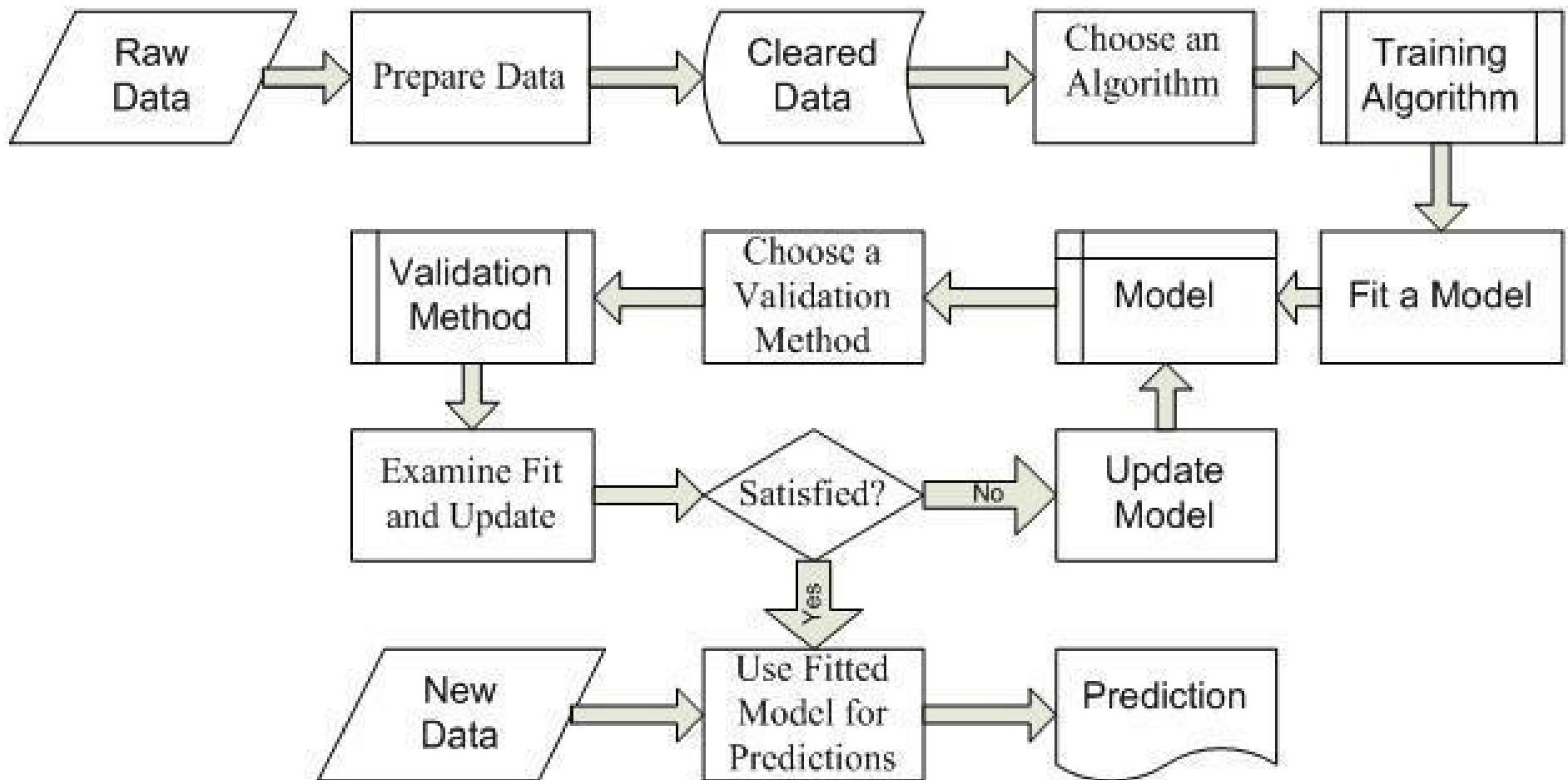
2.2. Workflow do Aprendizado Supervisionado



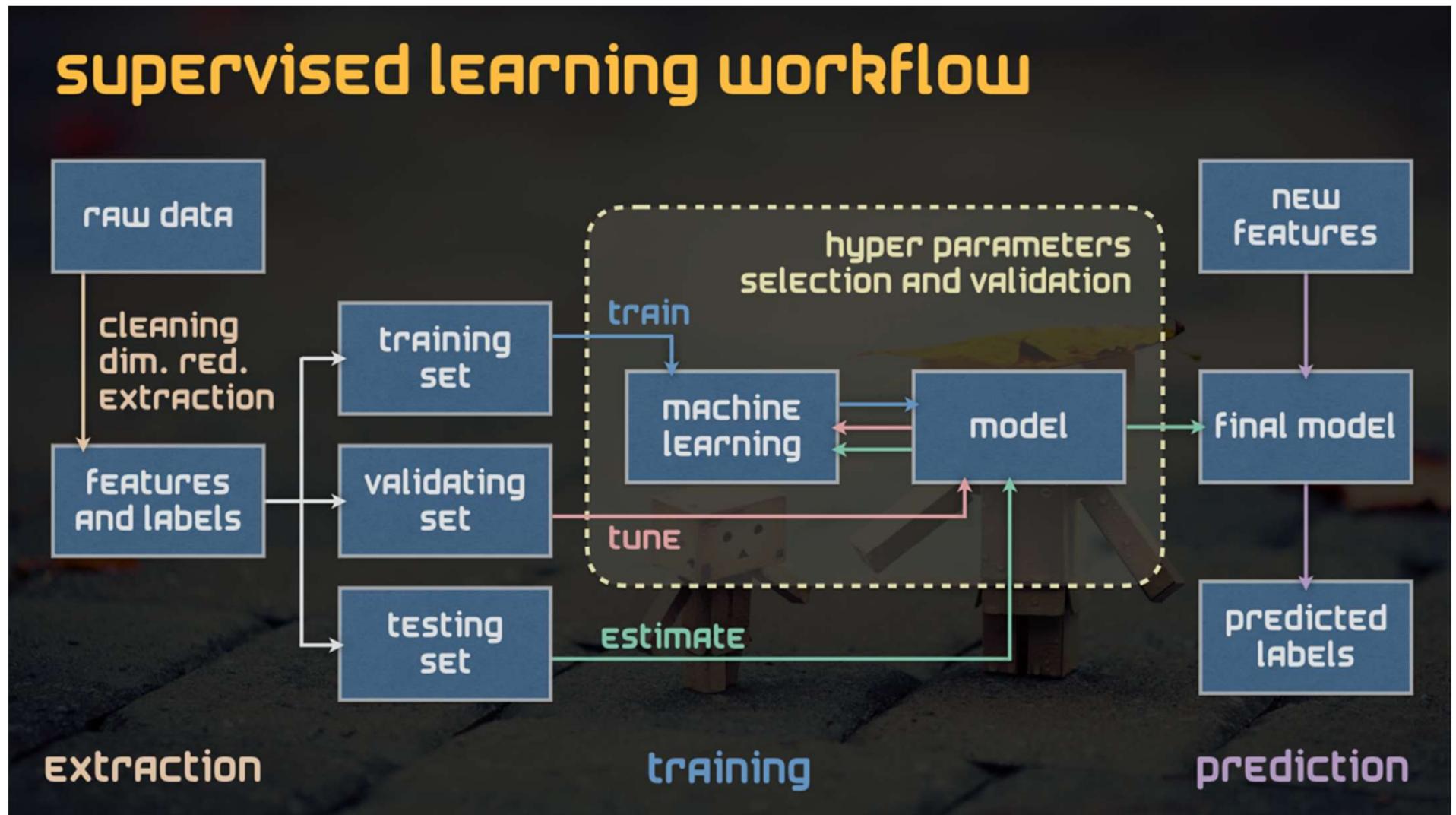
2.2. Workflow do Aprendizado Supervisionado



2.2. Workflow do Aprendizado Supervisionado

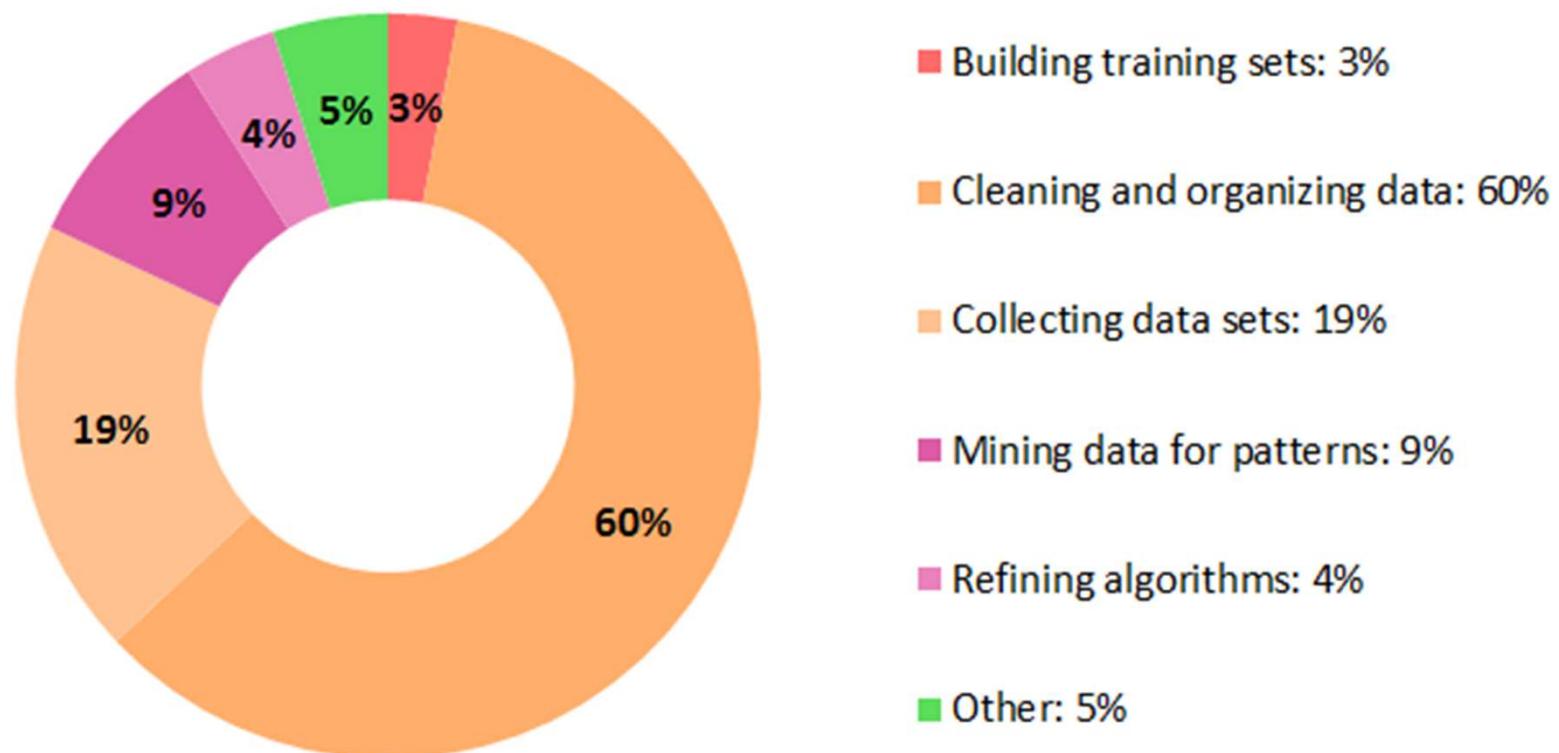


2.2. Workflow do Aprendizado Supervisionado



2.2. Workflow do Aprendizado Supervisionado

A maior parte do esforço humano é consumida no pré-processamento dos dados !!!





2.3. Preparação dos Dados



2.3. Preparação dos Dados

Preparação dos Dados

- Coleta e carga dos dados
- Entendimento dos dados
- “Limpeza” dos dados
- Seleção de atributos preditores
- Ajuste de escala
- Balanceamento de classes



2.3. Preparação dos Dados

Preparação dos Dados

- ❑ Coleta e carga dos dados →
- ❑ Entendimento dos dados
- ❑ “Limpeza” dos dados
- ❑ Seleção de atributos preditores
- ❑ Ajuste de escala
- ❑ Balanceamento de classes

```
import pandas as pd
```

IMPORTING DATA

`pd.read_csv(filename)` - From a CSV file

`pd.read_table(filename)` - From a delimited text file (like TSV)

`pd.read_excel(filename)` - From an Excel file

`pd.read_sql(query, connection_object)` -
Reads from a SQL table/database

`pd.read_json(json_string)` - Reads from a JSON formatted string, URL or file.

`pd.read_html(url)` - Parses an html URL, string or file and extracts tables to a list of dataframes

`pd.read_clipboard()` - Takes the contents of your clipboard and passes it to `read_table()`

`pd.DataFrame(dict)` - From a dict, keys for columns names, values for data as lists



2.3. Preparação dos Dados

Datasets nativos do *scikit-learn* :

scikit-learn comes with a few small standard datasets that do not require to download any file from some external website.

<code>load_boston</code> ([return_X_y])	Load and return the boston house-prices dataset (regression).
<code>load_iris</code> ([return_X_y])	Load and return the iris dataset (classification).
<code>load_diabetes</code> ([return_X_y])	Load and return the diabetes dataset (regression).
<code>load_digits</code> ([n_class, return_X_y])	Load and return the digits dataset (classification).
<code>load_linnerud</code> ([return_X_y])	Load and return the linnerud dataset (multivariate regression).
<code>load_wine</code> ([return_X_y])	Load and return the wine dataset (classification).
<code>load_breast_cancer</code> ([return_X_y])	Load and return the breast cancer wisconsin dataset (classification).

These datasets are useful to quickly illustrate the behavior of the various algorithms implemented in the scikit. They are however often too small to be representative of real world machine learning tasks.



2.3. Preparação dos Dados

Convertendo um **dataset** nativo em **dataframe** do Pandas:

```
import pandas as pd # conventional alias
from sklearn.datasets import load_boston

dataset = load_boston()
df = pd.DataFrame(dataset.data, columns=dataset.feature_names)
df['target'] = dataset.target
```

The result of `load_boston()` is a map-like object with four components: `['target', 'data', 'DESCR', 'feature_names']`:

- `dataset['target']` - 1D numpy array of target attribute values
- `dataset['data']` - 2D numpy array of attribute values
- `dataset['feature_names']` - 1D numpy array of names of the attributes
- `dataset['DESCR']` - text description of the dataset



2.3. Preparação dos Dados

Boston House Prices Dataset:

Number of Instances: 506

Number of Attributes: 13 numeric/categorical predictive

Attribute Information (in order):

- CRIM per capita crime rate by town
- ZN proportion of residential land zoned for lots over 25,000 sq.ft.
- INDUS proportion of non-retail business acres per town
- CHAS Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
- NOX nitric oxides concentration (parts per 10 million)
- RM average number of rooms per dwelling
- AGE proportion of owner-occupied units built prior to 1940
- DIS weighted distances to five Boston employment centres
- RAD index of accessibility to radial highways
- TAX full-value property-tax rate per \$10,000
- PTRATIO pupil-teacher ratio by town
- B $1000(Bk - 0.63)^2$ where Bk is the proportion of blacks by town
- LSTAT % lower status of the population
- MEDV Median value of owner-occupied homes in \$1000's



2.3. Preparação dos Dados

Preparação dos Dados

- ❑ Coleta e carga dos dados
- ❑ Entendimento dos dados
 - Quais informações estão disponíveis para o projeto?
 - Qual o significado de cada informação?
 - Quais delas são relevantes para o objetivo desejado?
 - Há outras informações relevantes além destas?
 - Há dados históricos disponíveis? Quais? Desde quando?
 - Visualizar correlações e potencial preditivo
- ❑ “Limpeza” dos dados
- ❑ Seleção de atributos preditores
- ❑ Ajuste de escala
- ❑ Balanceamento de classes



2.3. Preparação dos Dados

Preparação dos Dados

- ❑ Coleta e carga dos dados
- ❑ Entendimento dos dados
- ❑ “Limpeza” dos dados
 - tratar dados faltantes
 - tratar valores especiais (data/hora, latitude/longitude, etc.)
 - tratar atributos categóricos ordinais
 - tratar atributos categóricos não-ordinais
- ❑ Seleção de atributos preditores
- ❑ Ajuste de escala
- ❑ Balanceamento de classes



2.3. Preparação dos Dados

“Limpeza” dos Dados – valores faltantes

```
Definition : Imputer(self, missing_values="NaN", strategy="mean", axis=0, verbose=0, copy=True)
```

The imputation strategy.

- If "mean", then replace missing values using the mean along the axis.
- If "median", then replace missing values using the median along the axis.
- If "most_frequent", then replace missing using the most frequent value along the axis.

```
>>> import numpy as np
>>> from sklearn.preprocessing import Imputer
>>> imp = Imputer(missing_values='NaN', strategy='mean', axis=0)
>>> imp.fit([[1, 2], [np.nan, 3], [7, 6]])
Imputer(axis=0, copy=True, missing_values='NaN', strategy='mean', verbose=0)
>>> X = [[np.nan, 2], [6, np.nan], [7, 6]]
>>> print(imp.transform(X))
[[ 4.          2.        ]
 [ 6.          3.666...]
 [ 7.          6.        ]]
```



2.3. Preparação dos Dados

“Limpeza” dos Dados – valores especiais

Exemplo: datas no formato KSP

- Data KSP = AAAA + $\frac{\text{dias_iniciando_Jan_1} - 0.5}{365 + 1\text{ se ano bissexto}}$
 - ▶ 01/01/2005 é: $2005 + (1 - 0.5) / 365 = 2005.0014$
 - ▶ 31/03/2005 é: $2005 + (90 - 0.5) / 365 = 2005.2452$
 - ▶ 30/06/2005 é: $2005 + (181 - 0.5) / 365 = 2005.4945$
- Pode ser estendida para incluir horas, minutos, segundos, etc.



2.3. Preparação dos Dados

“Limpeza” dos Dados – atributos categóricos

- **Atributos categóricos ordinais** (e.g., *grau_de_satisfação*) podem ser convertidos para números preservando a ordem natural
 - ▶ Muito Satisfeito ⇒ 0.8
 - ▶ Satisfeito ⇒ 0.6
 - ▶ Pouco Satisfeito ⇒ 0.4
 - ▶ Insatisfeito ⇒ 0.2
- Por que é importante preservar a ordem natural?
 - ▶ Para permitir comparações que façam sentido.
Exemplo: $grau_de_satisfação > 0.5$



2.3. Preparação dos Dados

“Limpeza” dos Dados – atributos categóricos

- **Atributos categóricos ordinais**
 - ▶ Frio ⇒ 1 / Morno ⇒ 2 / Quente ⇒ 3

Definition : LabelEncoder()

```
>>> le = preprocessing.LabelEncoder()
>>> le.fit(["frio", "frio", "morno", "quente"])
LabelEncoder()
>>> list(le.classes_)
['quente', 'frio', 'morno']
>>> le.transform(["morno", "morno", "frio"])
array([2, 2, 1]...)
>>> list(le.inverse_transform([2, 2, 1]))
['morno', 'morno', 'frio']
```



2.3. Preparação dos Dados

“Limpeza” dos Dados – atributos categóricos

- Atributos categóricos binários
 - ▶ Exemplo: Sexo = (M)asculino ,(F)eeminino
- Converter para *Atributo_0_1* com os valores 0,1
 - ▶ Sexo = M \Rightarrow Sexo_0_1 = 0
 - ▶ Sexo = F \Rightarrow Sexo_0_1 = 1
- Utilizar LabelEncoder()



2.3. Preparação dos Dados

“Limpeza” dos Dados – atributos categóricos

- **Atributos categóricos multivalorados** com pequena quantidade de possíveis valores (e.g., < 20)
 - ▶ *Religião* = Católica, Protestante, Budista, ..., Outras
 - ▶ Para cada valor v de *Religião*, criar um atributo binário R_v , que será 1 se *Religião* = v , 0 caso contrário:

The diagram illustrates the transformation of a categorical attribute. On the left, a table has columns labeled 'ID', 'Religião', and '...'. The first row contains 'Pessoa 1' and 'Budista'. The second row contains 'Pessoa 2' and 'Católico'. An arrow points to the right, indicating a transformation. On the right, a second table has columns labeled 'ID', 'R_Católica', 'R_Protestante', 'R_Budista', and '...'. The first row contains 'Pessoa 1', '0', '0', '1', and an empty cell. The second row contains 'Pessoa 2', '1', '0', '0', and an empty cell.

ID	Religião	...
Pessoa 1	Budista	
Pessoa 2	Católico	

ID	R_Católica	R_Protestante	R_Budista	...
Pessoa 1	0	0	1	
Pessoa 2	1	0	0	



2.3. Preparação dos Dados

“Limpeza” dos Dados – atributos categóricos

Often features are not given as continuous values but categorical. For example a person could have features

```
["male", "female"], ["from Europe", "from US", "from Asia"],  
["uses Firefox", "uses Chrome", "uses Safari", "uses Internet Explorer"] . Such features can be  
efficiently coded as integers, for instance ["male", "from US", "uses Internet Explorer"] could be  
expressed as [0, 1, 3] while ["female", "from Asia", "uses Chrome"] would be [1, 2, 1] .
```

```
Definition : OneHotEncoder(n_values="auto", categorical_features="all", dtype=np.float64, sparse=True, handle_unknown='error')
```

```
>>> enc = preprocessing.OneHotEncoder()  
>>> enc.fit([[0, 0, 3], [1, 1, 0], [0, 2, 1], [1, 0, 2]])  
OneHotEncoder(categorical_features='all', dtype=<... 'numpy.float64'>,  
    handle_unknown='error', n_values='auto', sparse=True)  
>>> enc.transform([[0, 1, 3]]).toarray()  
array([[ 1.,  0.,  0.,  1.,  0.,  0.,  0.,  1.]])
```



2.3. Preparação dos Dados

“Limpeza” dos Dados – atributos categóricos

Atributos multivalorados com grande número de valores

- Descartar atributos cujos valores sejam únicos para cada registro ou que não possam ser agrupados: RG, CPF, etc.
- Para outros atributos, tentar agrupar valores “naturalmente”
 - ▶ 8500 CEP's ⇒ 150 bairros ⇒ 3 ou 5 regiões
 - ▶ Profissões - agrupar em grandes áreas + “outros”
- Criar atributos binários para os valores dos agrupamentos usando o método OneHotEncoder()



2.3. Preparação dos Dados

Preparação dos Dados

- ❑ Coleta e carga dos dados
- ❑ Entendimento dos dados
- ❑ “Limpeza” dos dados
- ❑ Seleção de atributos preditores
 - Selecionar atributos com alto potencial preditivo
(fortemente correlacionados com a variável a ser estimada)
 - Descartar atributos com baixo potencial preditivo
(atrapalham mais que ajudam!)
 - Formar preditores ainda melhores combinando atributos de alto potencial que sejam fracamente correlacionados entre si.
 - Assunto extenso. Vários métodos. Veremos melhor mais adiante.
- ❑ Ajuste de escala
- ❑ Balanceamento de classes



2.3. Preparação dos Dados

Preparação dos Dados

- ❑ Coleta e carga dos dados
- ❑ Entendimento dos dados
- ❑ “Limpeza” dos dados
- ❑ Seleção de atributos preditores
- ❑ Ajuste de escala
 - Alguns algoritmos de ML obtêm melhores resultados e/ou convergem mais rápido quando os valores dos atributos estão aprox. na mesma escala e/ou apresentam dispersão mais homogênea
 - Transformações mais comuns:
 - MinMaxScaler : mapeia uniformemente $\{ \min(x_i) ; \max(x_i) \} \rightarrow \{ L_{\min} ; L_{\max} \}$
 - MaxAbsScaler : divide por $\max(\text{abs}(x_i))$, de modo que $0 \rightarrow 0$ $\max(\text{abs}(x_i)) \rightarrow 1$
 - StandardScaler : subtrai a média e divide pelo desvio padrão
- ❑ Balanceamento de classes



2.3. Preparação dos Dados

Preparação dos Dados

- ❑ Coleta e carga dos dados
- ❑ Entendimento dos dados
- ❑ “Limpeza” dos dados
- ❑ Seleção de atributos preditores
- ❑ Ajuste de escala
- ❑ Balanceamento de classes
 - Alguns algoritmos de classificação têm mau desempenho quando há severo desbalanceamento entre classes nos dados de treinamento.
 - Entretanto, técnicas de balanceamento de classes podem ser usadas.
 - Assunto extenso. Várias técnicas. Veremos melhor mais adiante.



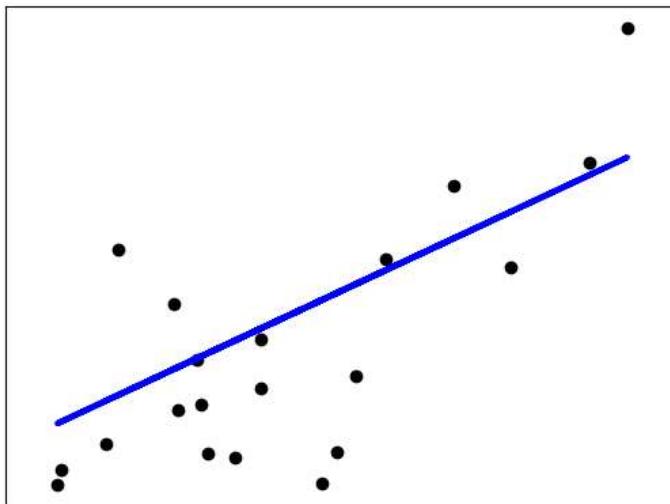
2.4. Modelos Lineares para Regressão

2.4. Modelos Lineares para Regressão

Regressão Linear (Mínimos Quadrados)

`LinearRegression` fits a linear model with coefficients $w = (w_1, \dots, w_p)$ to minimize the residual sum of squares between the observed responses in the dataset, and the responses predicted by the linear approximation. Mathematically it solves a problem of the form:

$$\min_w \|Xw - y\|_2^2$$





2.4. Modelos Lineares para Regressão

Regressão Linear (Mínimos Quadrados)

```
>>> from sklearn import linear_model
>>> reg = linear_model.LinearRegression()
>>> reg.fit ([[0, 0], [1, 1], [2, 2]], [0, 1, 2])
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
>>> reg.coef_
array([ 0.5,  0.5])
```



2.4. Modelos Lineares para Regressão

Mean Squared Error (MSE)

If \hat{y}_i is the predicted value of the i -th sample and y_i is the corresponding true value, then the mean squared error (MSE) estimated over n_{samples} is defined as

$$\text{MSE}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} (y_i - \hat{y}_i)^2$$



2.4. Modelos Lineares para Regressão

Mean Squared Error (MSE)

```
>>> from sklearn.metrics import mean_squared_error
>>> y_true = [3, -0.5, 2, 7]
>>> y_pred = [2.5, 0.0, 2, 8]
>>> mean_squared_error(y_true, y_pred)
0.375
>>> y_true = [[0.5, 1], [-1, 1], [7, -6]]
>>> y_pred = [[0, 2], [-1, 2], [8, -5]]
>>> mean_squared_error(y_true, y_pred)
0.7083...
```



2.4. Modelos Lineares para Regressão

Variance Score (R²)

If \hat{y}_i is the predicted value of the i -th sample and y_i is the corresponding true value, then the score R^2 estimated over n_{samples} is defined as

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=0}^{n_{\text{samples}}-1} (y_i - \hat{y}_i)^2}{\sum_{i=0}^{n_{\text{samples}}-1} (y_i - \bar{y})^2}$$

where $\bar{y} = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} y_i$.



2.4. Modelos Lineares para Regressão

R² Score

```
>>> from sklearn.metrics import r2_score
>>> y_true = [3, -0.5, 2, 7]
>>> y_pred = [2.5, 0.0, 2, 8]
>>> r2_score(y_true, y_pred)
0.948...
>>> y_true = [[0.5, 1], [-1, 1], [7, -6]]
>>> y_pred = [[0, 2], [-1, 2], [8, -5]]
>>> r2_score(y_true, y_pred, multioutput='variance_weighted')
...
0.938...
>>> y_true = [[0.5, 1], [-1, 1], [7, -6]]
>>> y_pred = [[0, 2], [-1, 2], [8, -5]]
>>> r2_score(y_true, y_pred, multioutput='uniform_average')
...
0.936...
>>> r2_score(y_true, y_pred, multioutput='raw_values')
...
array([ 0.965...,  0.908...])
>>> r2_score(y_true, y_pred, multioutput=[0.3, 0.7])
...
0.925...
```



2.4. Modelos Lineares para Regressão

Regressão Polinomial

$$\hat{y}(w, x) = w_0 + w_1x_1 + w_2x_2$$

$$\hat{y}(w, x) = w_0 + w_1x_1 + w_2x_2 + w_3x_1x_2 + w_4x_1^2 + w_5x_2^2$$

$$z = [x_1, x_2, x_1x_2, x_1^2, x_2^2]$$

$$\hat{y}(w, x) = w_0 + w_1z_1 + w_2z_2 + w_3z_3 + w_4z_4 + w_5z_5$$



2.4. Modelos Lineares para Regressão

Regressão Polinomial

```
>>> from sklearn.preprocessing import PolynomialFeatures
>>> import numpy as np
>>> X = np.arange(6).reshape(3, 2)
>>> X
array([[0, 1],
       [2, 3],
       [4, 5]])
>>> poly = PolynomialFeatures(degree=2)
>>> poly.fit_transform(X)
array([[ 1.,  0.,  1.,  0.,  0.,  1.],
       [ 1.,  2.,  3.,  4.,  6.,  9.],
       [ 1.,  4.,  5., 16., 20., 25.]])
```



2.4. Modelos Lineares para Regressão

Regressão Polinomial

```
>>> from sklearn.preprocessing import PolynomialFeatures
>>> from sklearn.linear_model import LinearRegression
>>> from sklearn.pipeline import Pipeline
>>> import numpy as np
>>> model = Pipeline([('poly', PolynomialFeatures(degree=3)),
...                   ('linear', LinearRegression(fit_intercept=False))])
>>> # fit to an order-3 polynomial data
>>> x = np.arange(5)
>>> y = 3 - 2 * x + x ** 2 - x ** 3
>>> model = model.fit(x[:, np.newaxis], y)
>>> model.named_steps['linear'].coef_
array([ 3., -2.,  1., -1.])
```



2.4. Modelos Lineares para Regressão

Regressão Ridge

$$\min_w \lVert Xw - y \rVert_2^2 + \alpha \lVert w \rVert_2^2$$

```
>>> from sklearn import linear_model
>>> reg = linear_model.Ridge(alpha = .5)
>>> reg.fit ([[0, 0], [0, 0], [1, 1]], [0, .1, 1])
Ridge(alpha=0.5, copy_X=True, fit_intercept=True, max_iter=None,
      normalize=False, random_state=None, solver='auto', tol=0.001)
>>> reg.coef_
array([ 0.34545455,  0.34545455])
>>> reg.intercept_
0.13636...
```



2.4. Modelos Lineares para Regressão

Regressão LASSO

$$\min_w \frac{1}{2n_{samples}} \|Xw - y\|_2^2 + \alpha \|w\|_1$$

```
>>> from sklearn import linear_model
>>> reg = linear_model.Lasso(alpha = 0.1)
>>> reg.fit([[0, 0], [1, 1]], [0, 1])
Lasso(alpha=0.1, copy_X=True, fit_intercept=True, max_iter=1000,
      normalize=False, positive=False, precompute=False, random_state=None,
      selection='cyclic', tol=0.0001, warm_start=False)
>>> reg.predict([[1, 1]])
array([ 0.8])
```



2.4. Modelos Lineares para Regressão

Regressão Elastic Net

$$\min_w \frac{1}{2n_{samples}} \|Xw - y\|_2^2 + \alpha\rho\|w\|_1 + \frac{\alpha(1-\rho)}{2}\|w\|_2^2$$

```
class sklearn.linear_model. ElasticNet (alpha=1.0, l1_ratio=0.5, fit_intercept=True, normalize=False,  
precompute=False, max_iter=1000, copy_X=True, tol=0.0001, warm_start=False, positive=False,  
random_state=None, selection='cyclic')
```

[source]

Linear regression with combined L1 and L2 priors as regularizer.

Minimizes the objective function:

```
1 / (2 * n_samples) * ||y - Xw||^2_2  
+ alpha * l1_ratio * ||w||_1  
+ 0.5 * alpha * (1 - l1_ratio) * ||w||^2_2
```

If you are interested in controlling the L1 and L2 penalty separately, keep in mind that this is equivalent to:

```
a * L1 + b * L2
```

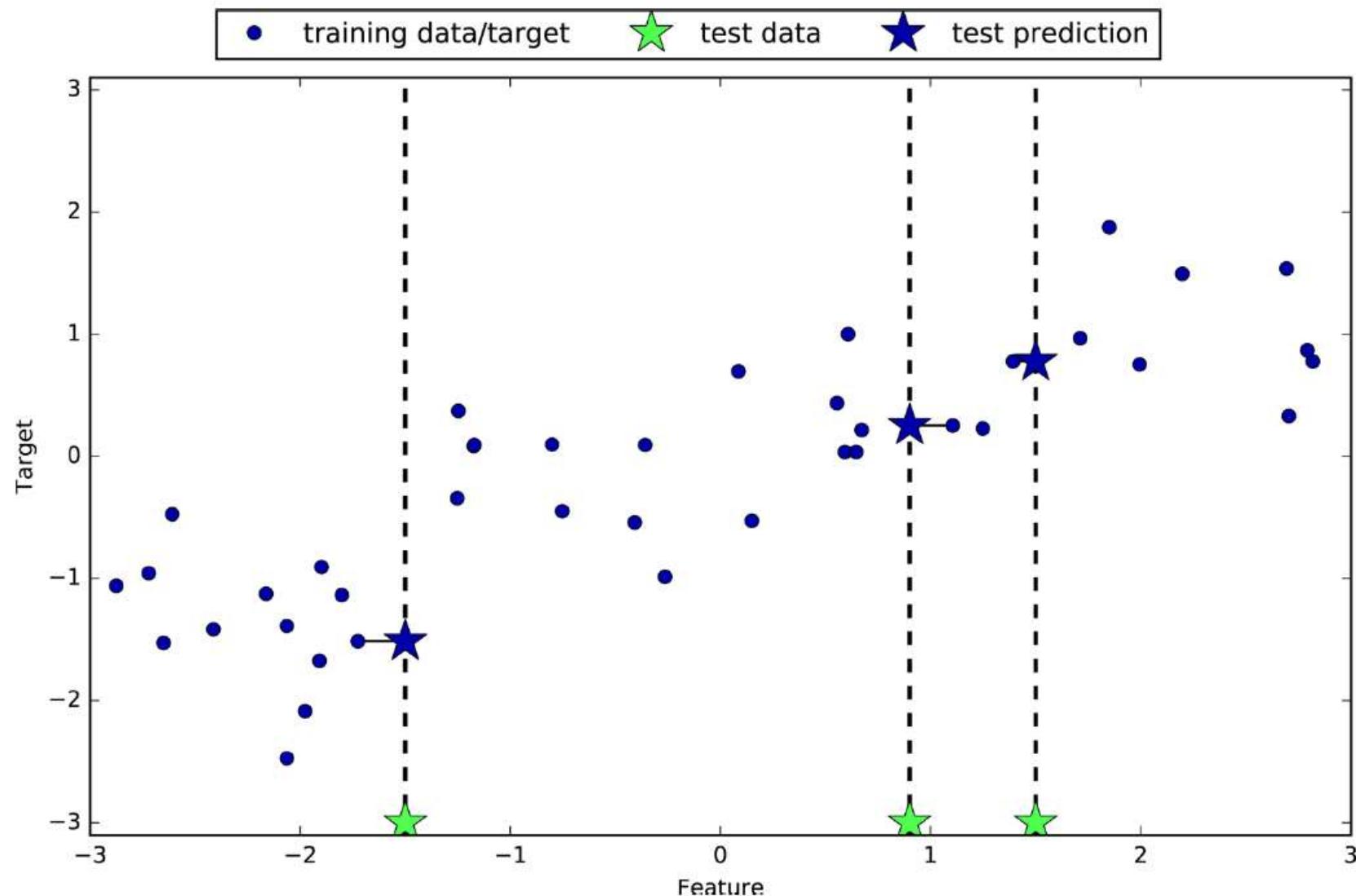
where:

```
alpha = a + b and l1_ratio = a / (a + b)
```

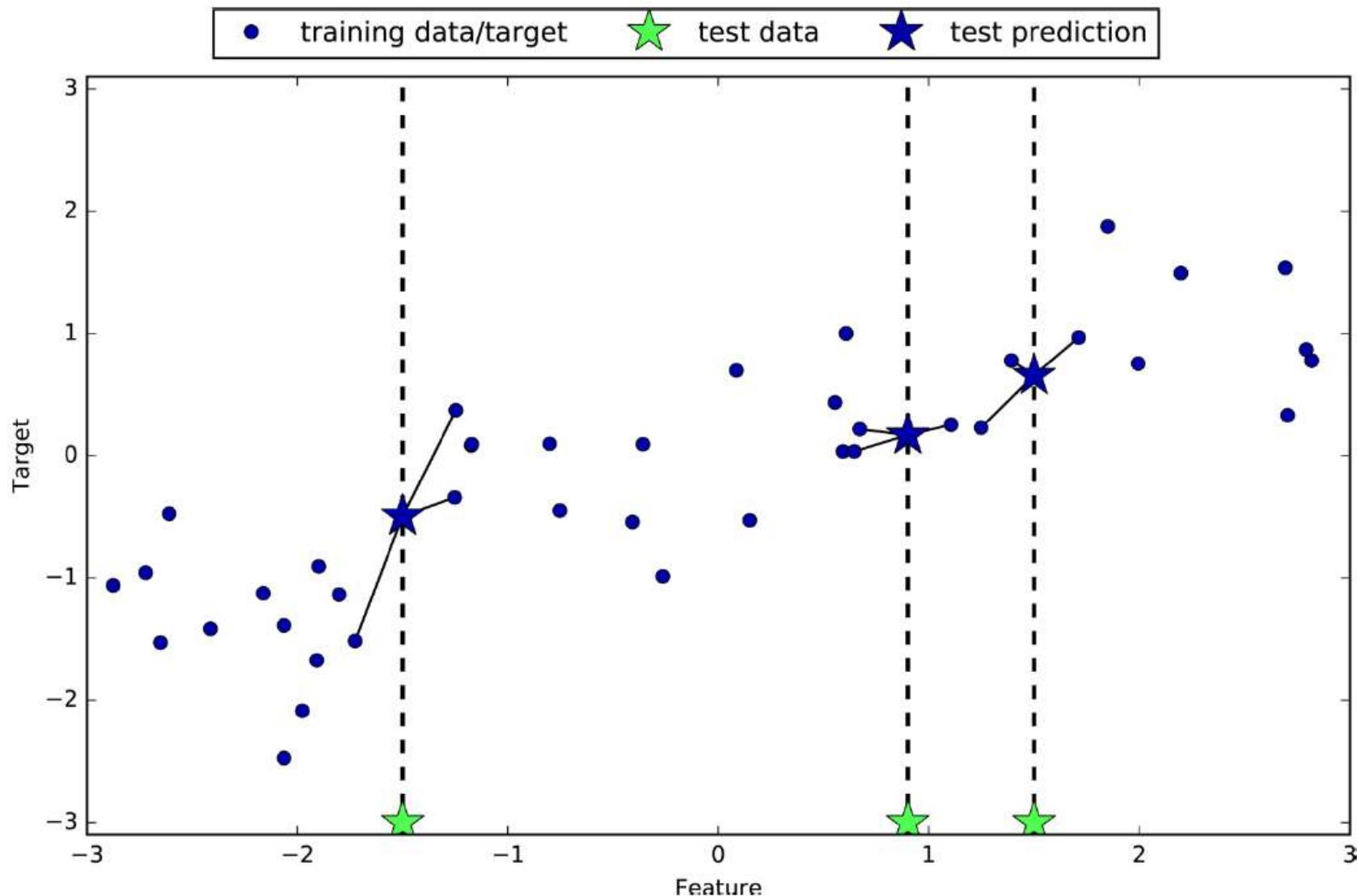


2.5. Regressão pelo Método KNN

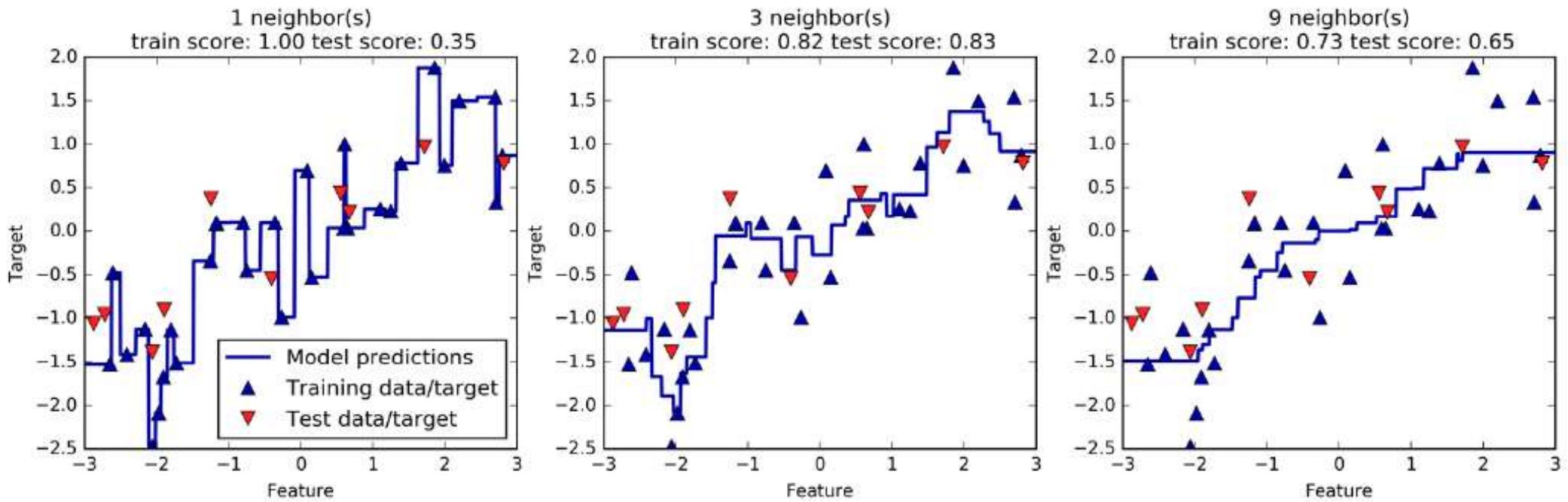
2.5. Regressão pelo Método KNN



2.5. Regressão pelo Método KNN



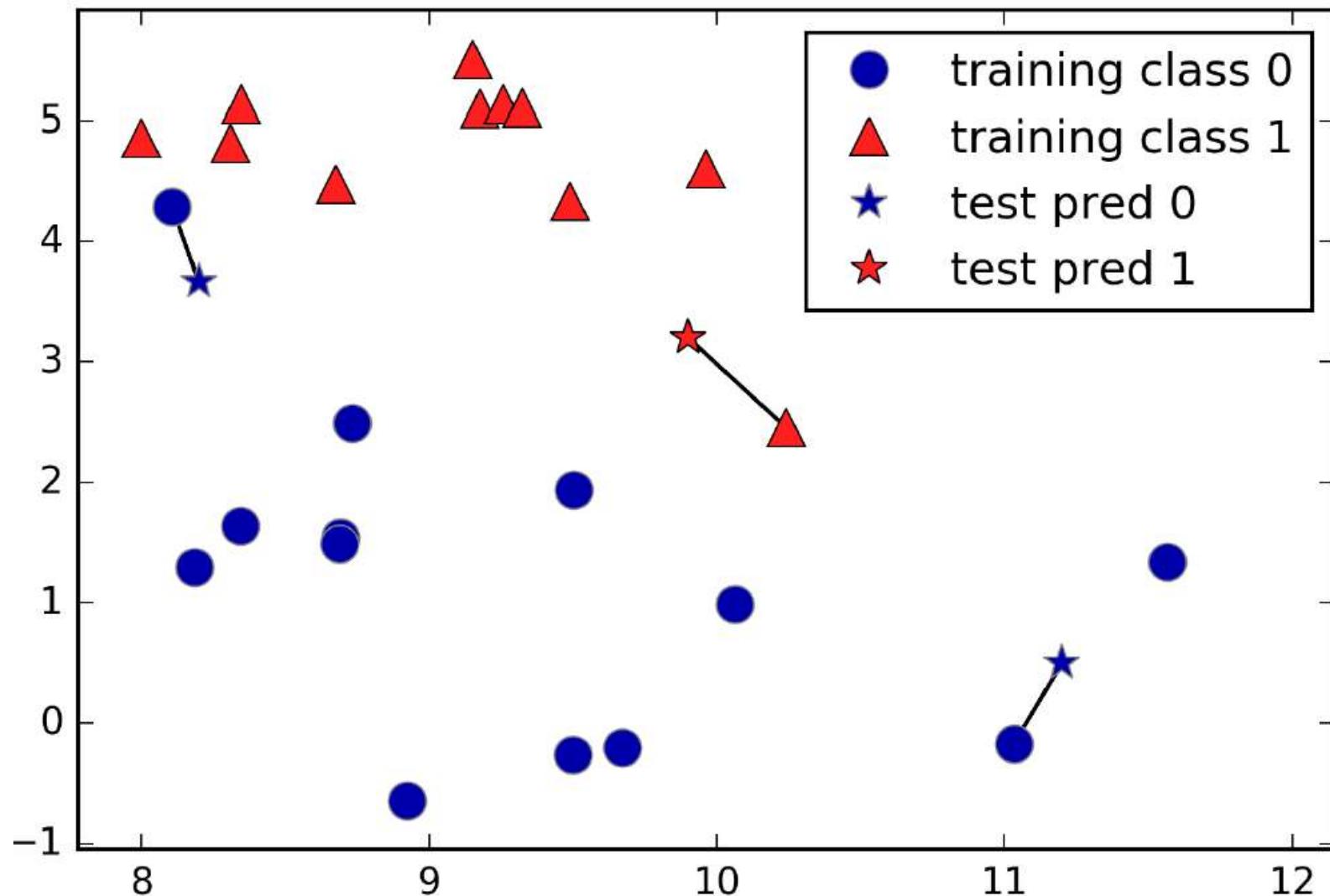
2.5. Regressão pelo Método KNN



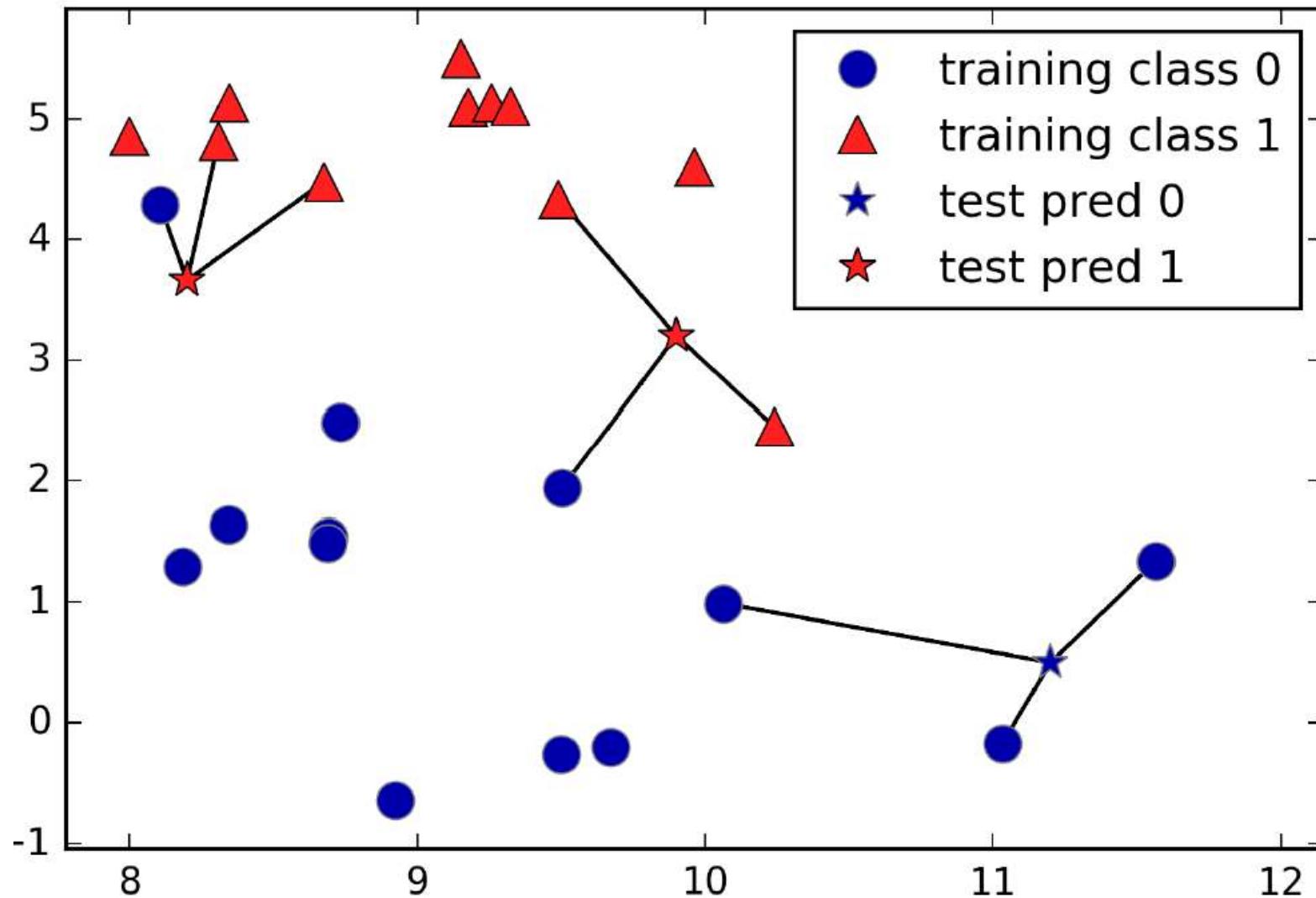


2.6. Classificação pelo Método KNN

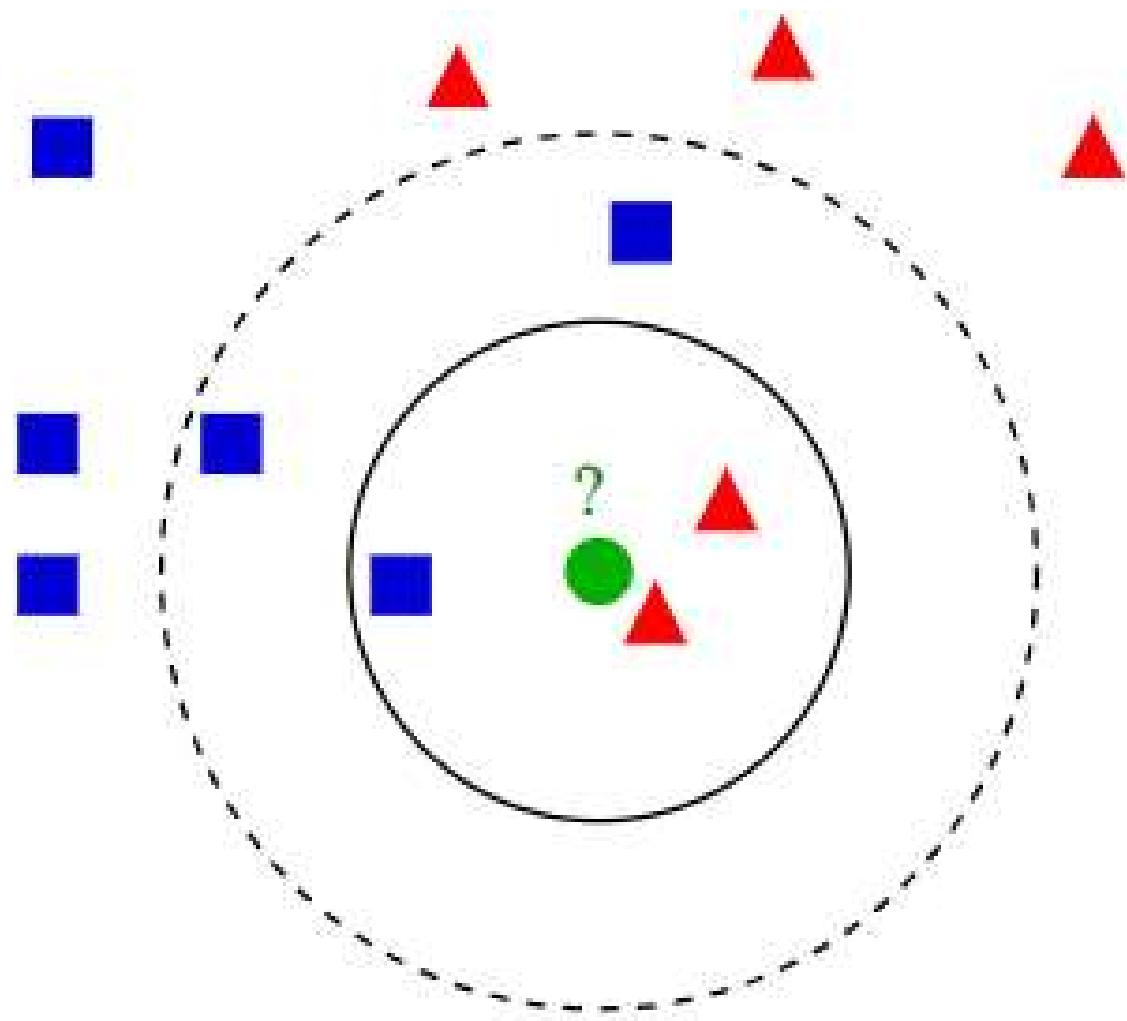
2.6. Classificação pelo Método KNN



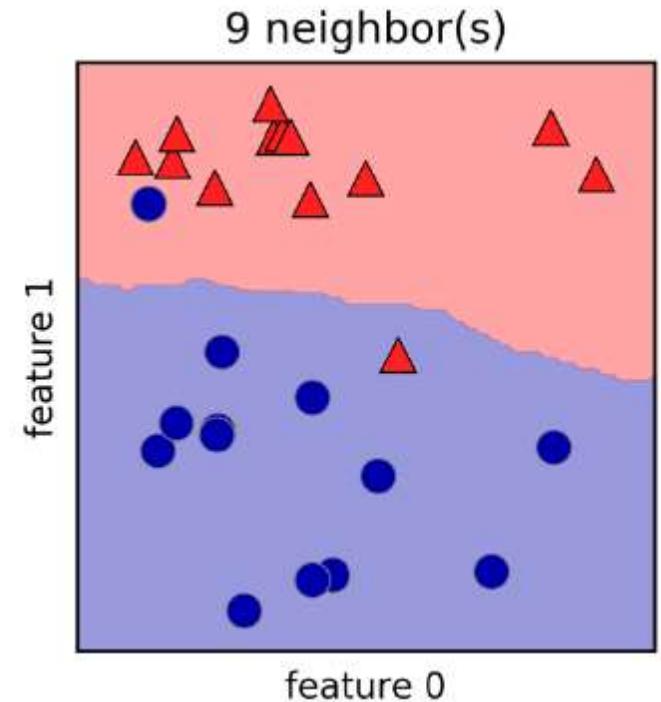
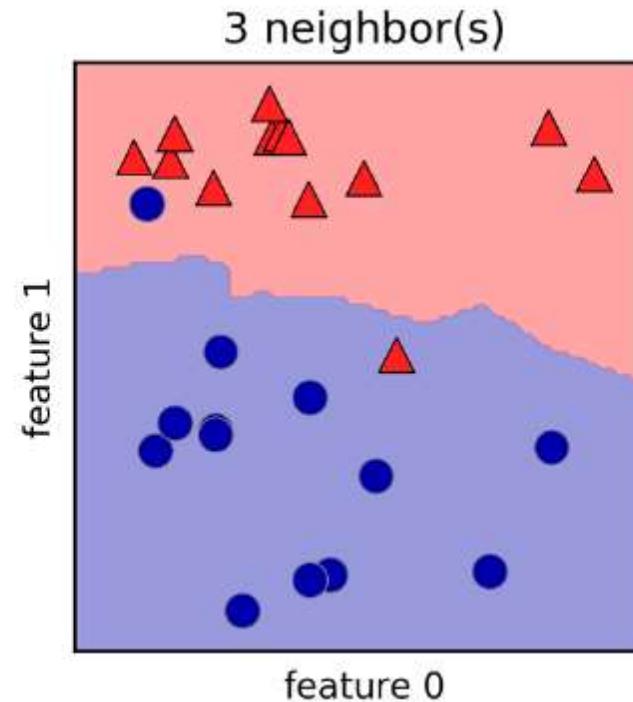
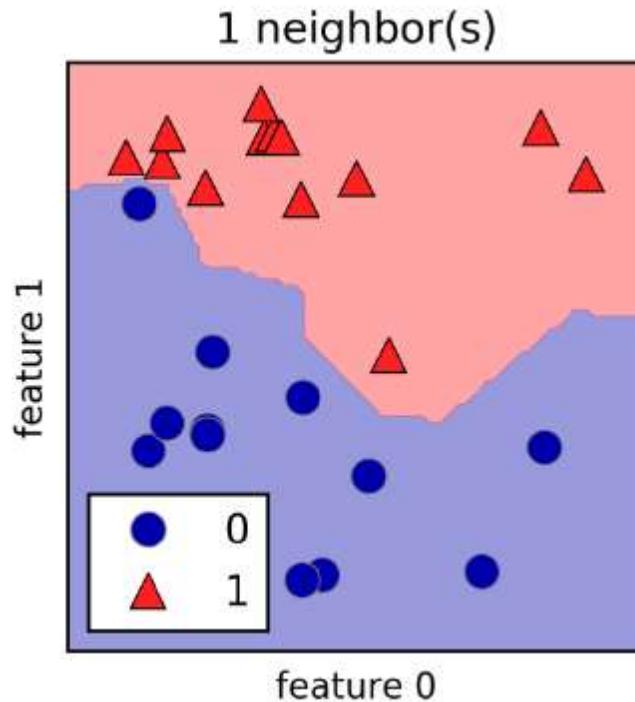
2.6. Classificação pelo Método KNN



2.6. Classificação pelo Método KNN



2.6. Classificação pelo Método KNN





2.7. Matriz de Confusão



2.7. Matriz de Confusão

		Predicted class	
		<i>P</i>	<i>N</i>
Actual Class	<i>P</i>	True Positives (TP)	False Negatives (FN)
	<i>N</i>	False Positives (FP)	True Negatives (TN)



2.7. Matriz de Confusão

		Predicted class				
		P	N			
Actual Class	P	True Positives (TP)	False Negatives (FN)			
	N	False Positives (FP)	True Negatives (TN)			
		Predicted				
		Iris-setosa	Iris-versicolor	Iris-virginica	Σ	
Actual	Iris-setosa	100.0 %	0.0 %	0.0 %	50	
	Iris-versicolor	0.0 %	88.7 %	6.4 %	50	
	Iris-virginica	0.0 %	11.3 %	93.6 %	50	
		Σ	50	53	47	150



2.7. Matriz de Confusão

```
sklearn.metrics. confusion_matrix (y_true, y_pred, labels=None, sample_weight=None)
```

Examples

```
>>> from sklearn.metrics import confusion_matrix
>>> y_true = [2, 0, 2, 2, 0, 1]
>>> y_pred = [0, 0, 2, 2, 0, 2]
>>> confusion_matrix(y_true, y_pred)
array([[2, 0, 0],
       [0, 0, 1],
       [1, 0, 2]])
```

```
>>> y_true = ["cat", "ant", "cat", "cat", "ant", "bird"]
>>> y_pred = ["ant", "ant", "cat", "cat", "ant", "cat"]
>>> confusion_matrix(y_true, y_pred, labels=["ant", "bird", "cat"])
array([[2, 0, 0],
       [0, 0, 1],
       [1, 0, 2]])
```

		Predicted class	
		P	N
Actual Class	P	True Positives (TP)	False Negatives (FN)
	N	False Positives (FP)	True Negatives (TN)

In the binary case, we can extract true positives, etc as follows:

```
>>> tn, fp, fn, tp = confusion_matrix([0, 1, 0, 1], [1, 1, 1, 0]).ravel()
>>> (tn, fp, fn, tp)
(0, 2, 1, 1)
```



2.8. Métricas de Classificação



2.8. Métricas de Classificação

$$Accuracy = \frac{T_p + T_n}{T_p + T_n + F_p + F_n}$$

$$Precision = \frac{T_p}{T_p + F_p}$$

$$Recall = \frac{T_p}{T_p + F_n}$$

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

		Predicted class	
		<i>P</i>	<i>N</i>
Actual Class	<i>P</i>	True Positives (TP)	False Negatives (FN)
	<i>N</i>	False Positives (FP)	True Negatives (TN)



2.8. Métricas de Classificação

Accuracy

```
sklearn.metrics. accuracy_score(y_true, y_pred, normalize=True, sample_weight=None)
```

$$Accuracy = \frac{T_p + T_n}{T_p + T_n + F_p + F_n}$$

```
>>> import numpy as np
>>> from sklearn.metrics import accuracy_score
>>> y_pred = [0, 2, 1, 3]
>>> y_true = [0, 1, 2, 3]
>>> accuracy_score(y_true, y_pred)
0.5
>>> accuracy_score(y_true, y_pred, normalize=False)
2
```



2.8. Métricas de Classificação

Precision

```
sklearn.metrics.precision_score(y_true, y_pred, labels=None, pos_label=1, average='binary',  
sample_weight=None)
```

$$Precision = \frac{T_p}{T_p + F_p}$$

```
>>> from sklearn.metrics import precision_score  
>>> y_true = [0, 1, 2, 0, 1, 2]  
>>> y_pred = [0, 2, 1, 0, 0, 1]  
>>> precision_score(y_true, y_pred, average='macro')  
0.22...  
>>> precision_score(y_true, y_pred, average='micro')  
0.33...  
>>> precision_score(y_true, y_pred, average='weighted')  
...  
0.22...  
>>> precision_score(y_true, y_pred, average=None)  
array([ 0.66...,  0.        ,  0.        ])
```



2.8. Métricas de Classificação

Recall

```
sklearn.metrics. recall_score(y_true, y_pred, labels=None, pos_label=1, average='binary',  
sample_weight=None)
```

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

```
>>> from sklearn.metrics import recall_score  
>>> y_true = [0, 1, 2, 0, 1, 2]  
>>> y_pred = [0, 2, 1, 0, 0, 1]  
>>> recall_score(y_true, y_pred, average='macro')  
0.33...  
>>> recall_score(y_true, y_pred, average='micro')  
0.33...  
>>> recall_score(y_true, y_pred, average='weighted')  
0.33...  
>>> recall_score(y_true, y_pred, average=None)  
array([ 1.,  0.,  0.])
```



2.8. Métricas de Classificação

F1 Score

```
sklearn.metrics. f1_score(y_true, y_pred, labels=None, pos_label=1, average='binary', sample_weight=None)
```

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

```
>>> from sklearn.metrics import f1_score
>>> y_true = [0, 1, 2, 0, 1, 2]
>>> y_pred = [0, 2, 1, 0, 0, 1]
>>> f1_score(y_true, y_pred, average='macro')
0.26...
>>> f1_score(y_true, y_pred, average='micro')
0.33...
>>> f1_score(y_true, y_pred, average='weighted')
0.26...
>>> f1_score(y_true, y_pred, average=None)
array([ 0.8,  0. ,  0. ])
```

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)

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

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)

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

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

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

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

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

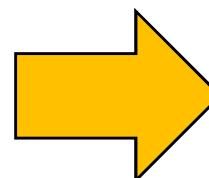
	Predicted Class 1	Predicted Class 2	Predicted Class 3
Actual Class 1	TP1		
Actual Class 2		TP2	
Actual Class 3			TP3

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

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, confusion_matrix
```

2.8. Métricas de Classificação

Resolvendo o dataset “digits” por K-NN:

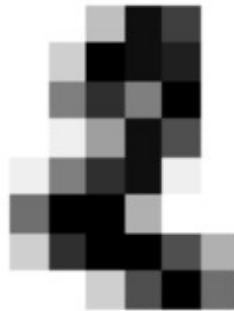
Training: 0



Training: 1



Training: 2



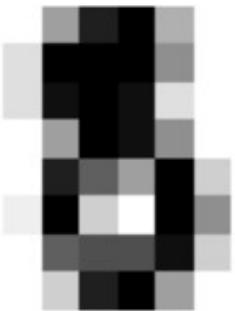
Training: 3



Prediction: 8



Prediction: 8



Prediction: 4



Prediction: 9

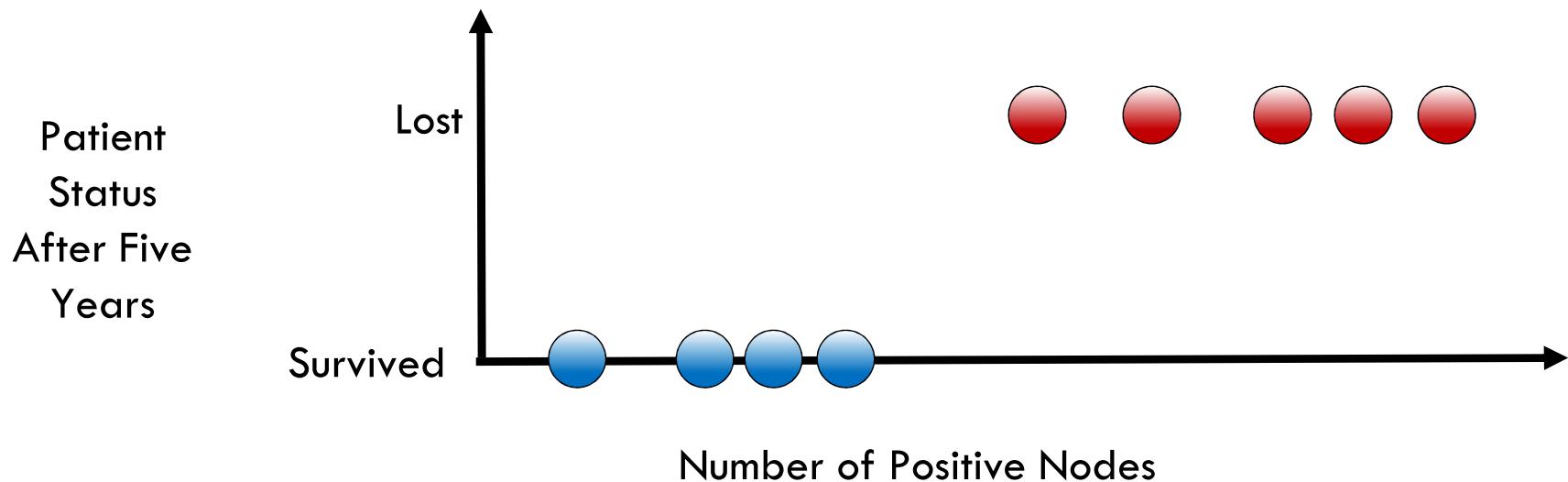




2.9. Regressão Logística

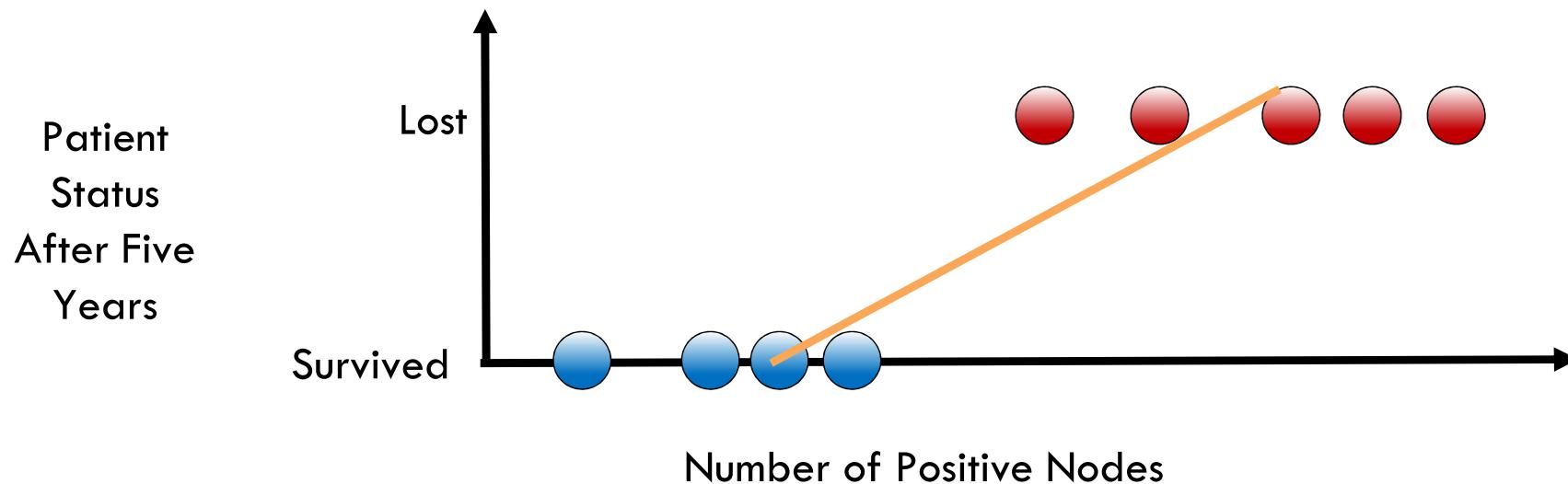
2.9. Regressão Logística

Introduction to Logistic Regression



2.9. Regressão Logística

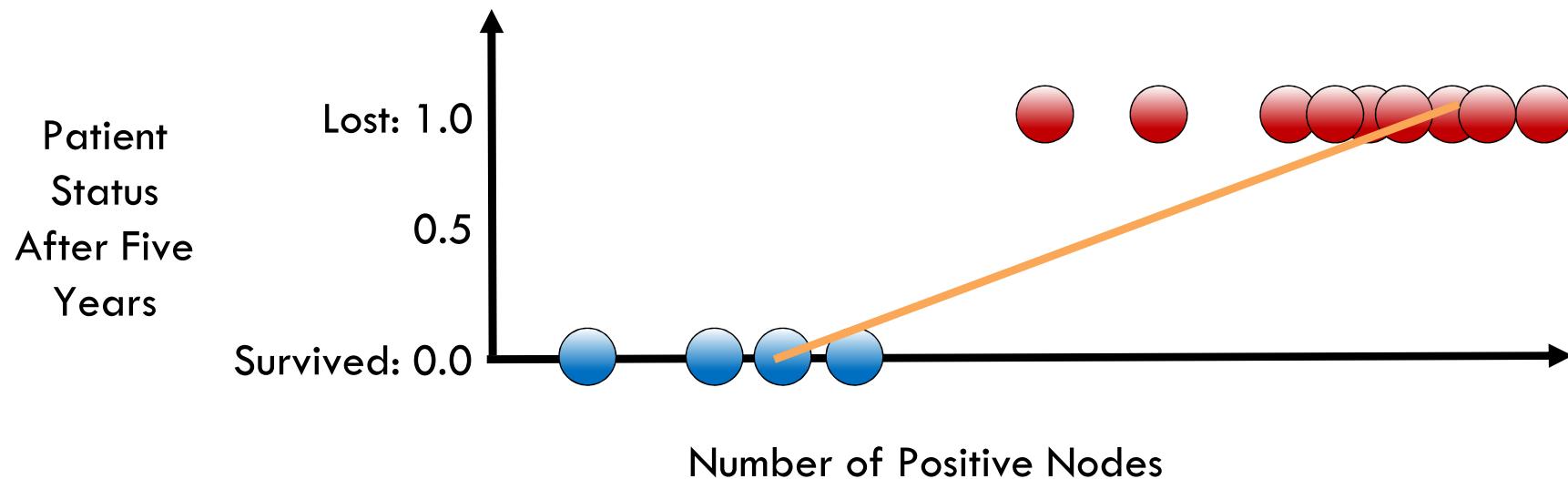
Linear Regression for Classification?



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

2.9. Regressão Logística

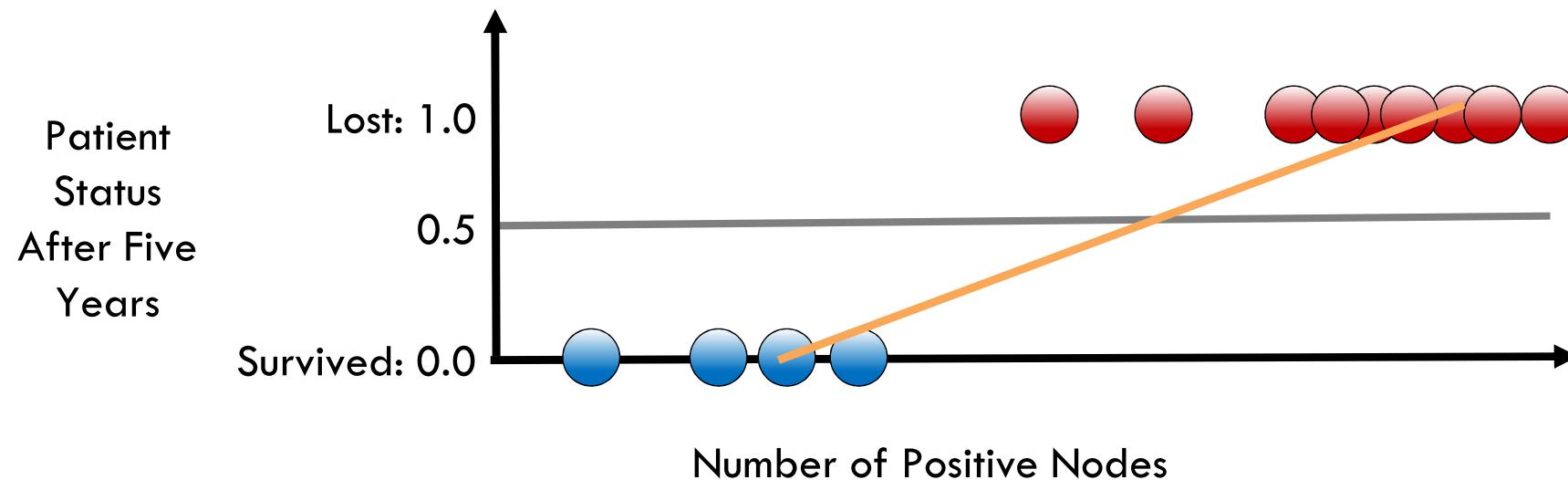
Linear Regression for Classification?



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

2.9. Regressão Logística

Linear Regression for Classification?

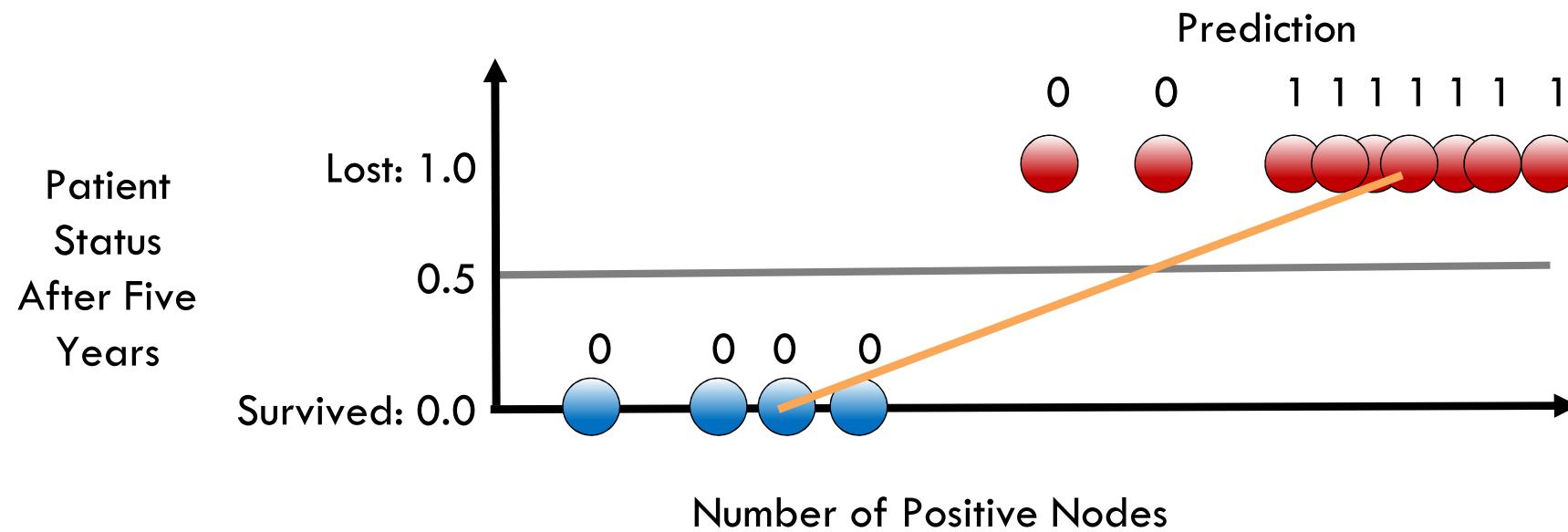


If model result > 0.5 : predict lost

If model result < 0.5 : predict survived

2.9. Regressão Logística

Linear Regression for Classification?

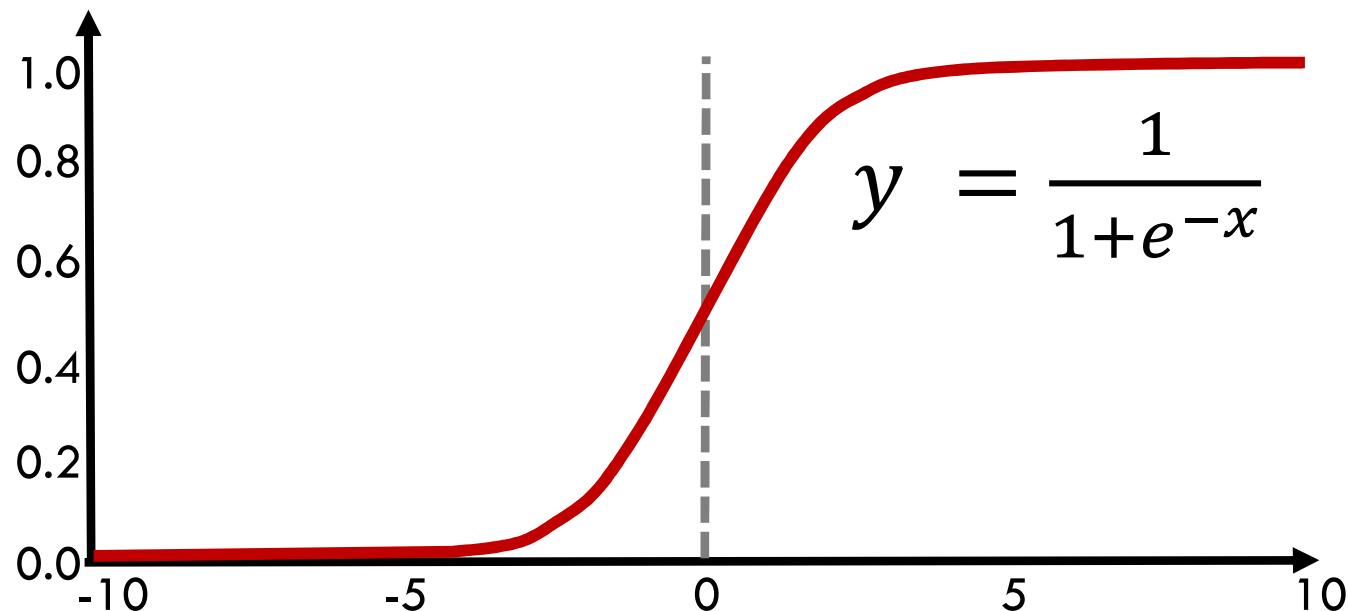


If model result > 0.5 : predict lost

If model result < 0.5 : predict survived

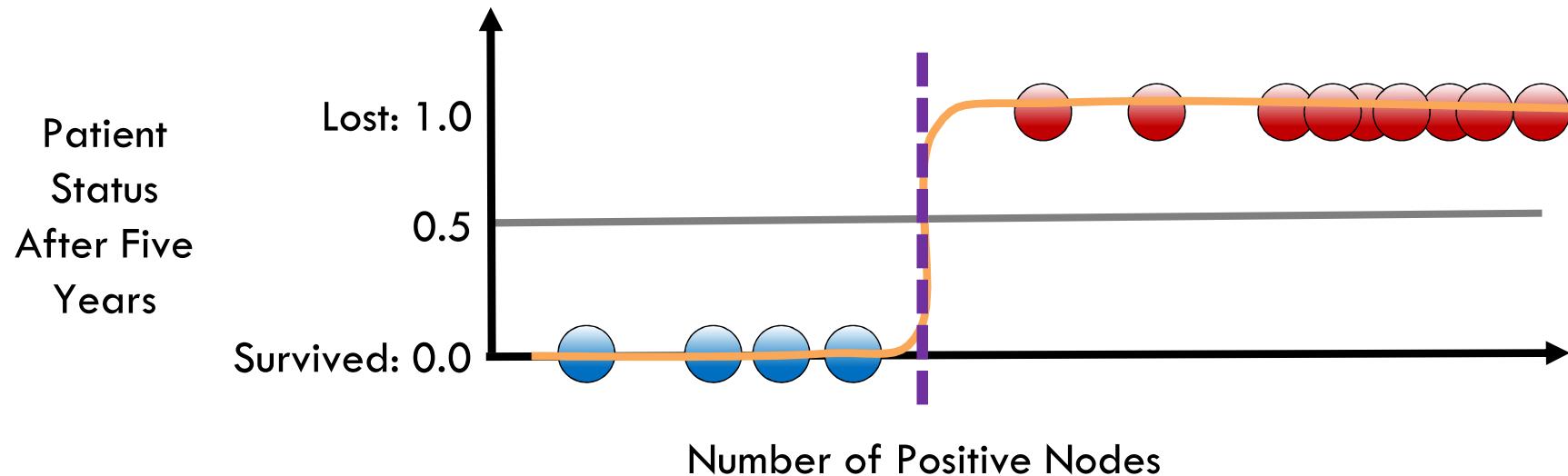
2.9. Regressão Logística

What is this Function?



2.9. Regressão Logística

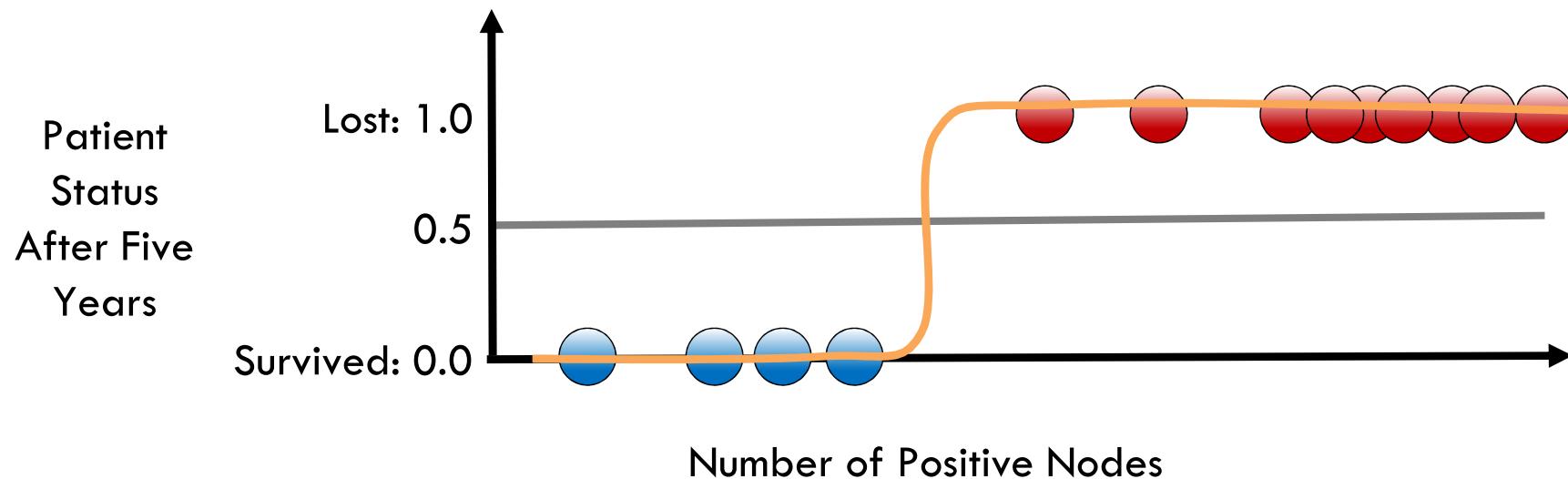
The Decision Boundary



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

2.9. Regressão Logística

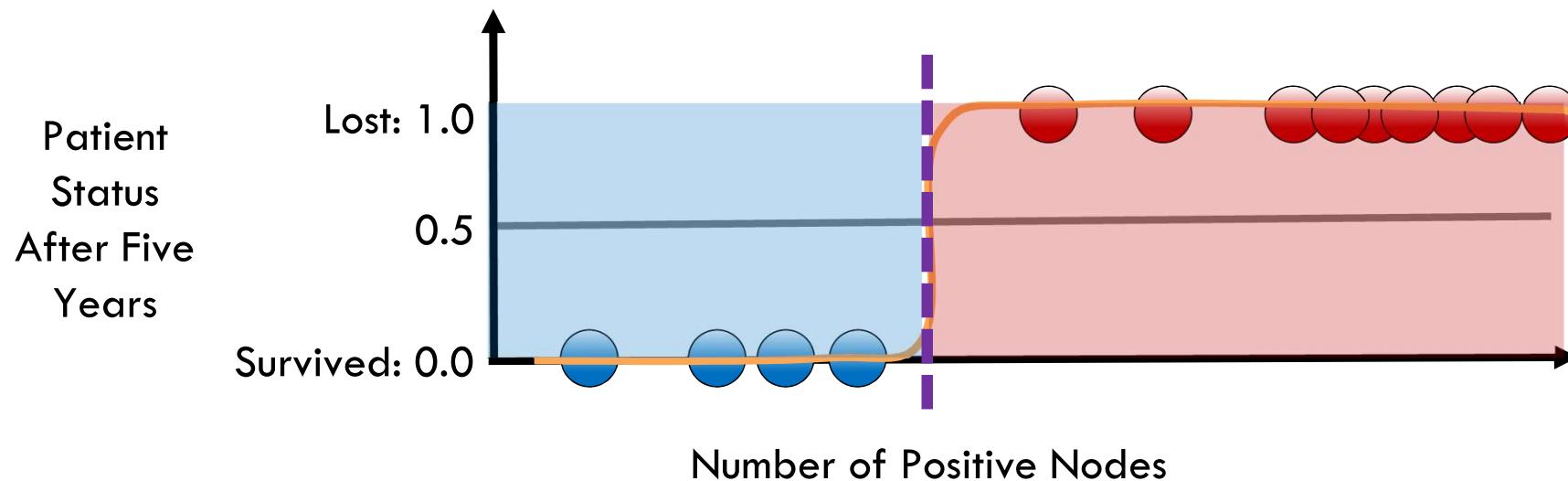
Logistic Regression



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

2.9. Regressão Logística

The Decision Boundary



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

2.9. Regressão Logística

Relationship of Logistic to Linear Regression

Logistic
Function

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

2.9. Regressão Logística

Relationship of Logistic to Linear Regression

Logistic
Function

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

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

2.9. Regressão Logística

Relationship of Logistic to Linear Regression

Logistic
Function

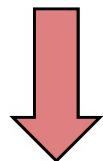
$$P(x) = \frac{e^{(\beta_0 + \beta_1 x)}}{1 + e^{(\beta_0 + \beta_1 x)}}$$

2.9. Regressão Logística

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

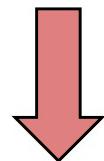
$$\frac{P(x)}{1 - P(x)} = e^{(\beta_0 + \beta_1 x)}$$

2.9. Regressão Logística

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

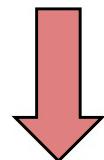
$$\log \left[\frac{P(x)}{1 - P(x)} \right] = \beta_0 + \beta_1 x$$

2.9. Regressão Logística

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

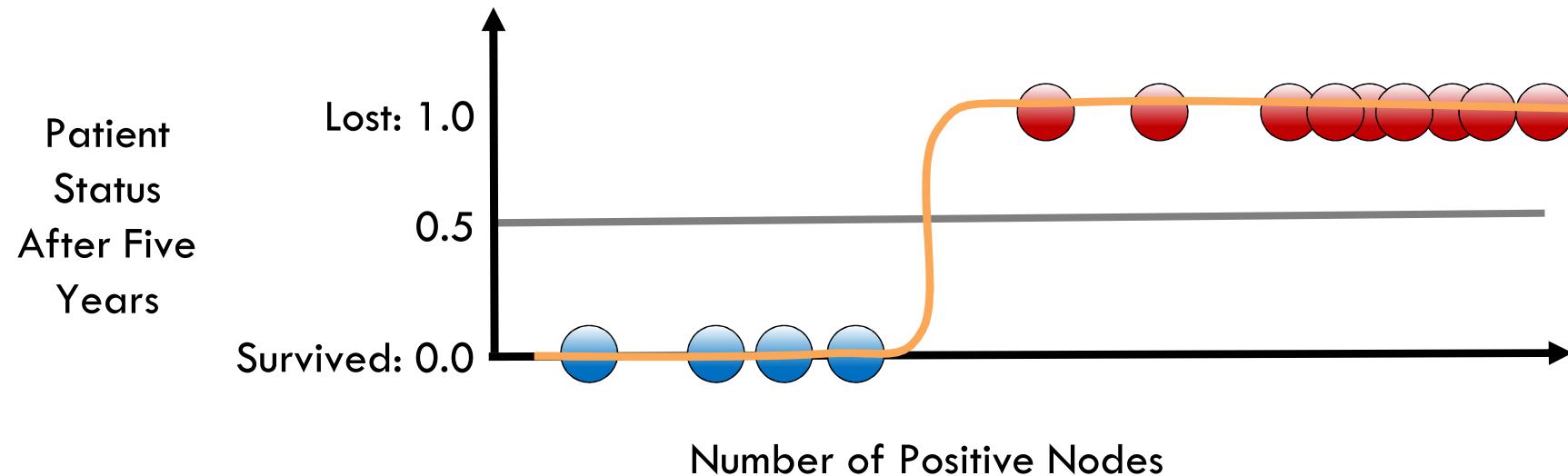
$$\log \left[\frac{P(x)}{1 - P(x)} \right] = \boxed{\beta_0 + \beta_1 x}$$

2.9. Regressão Logística

Classification with Logistic Regression

One feature (nodes)

Two labels (survived, lost)

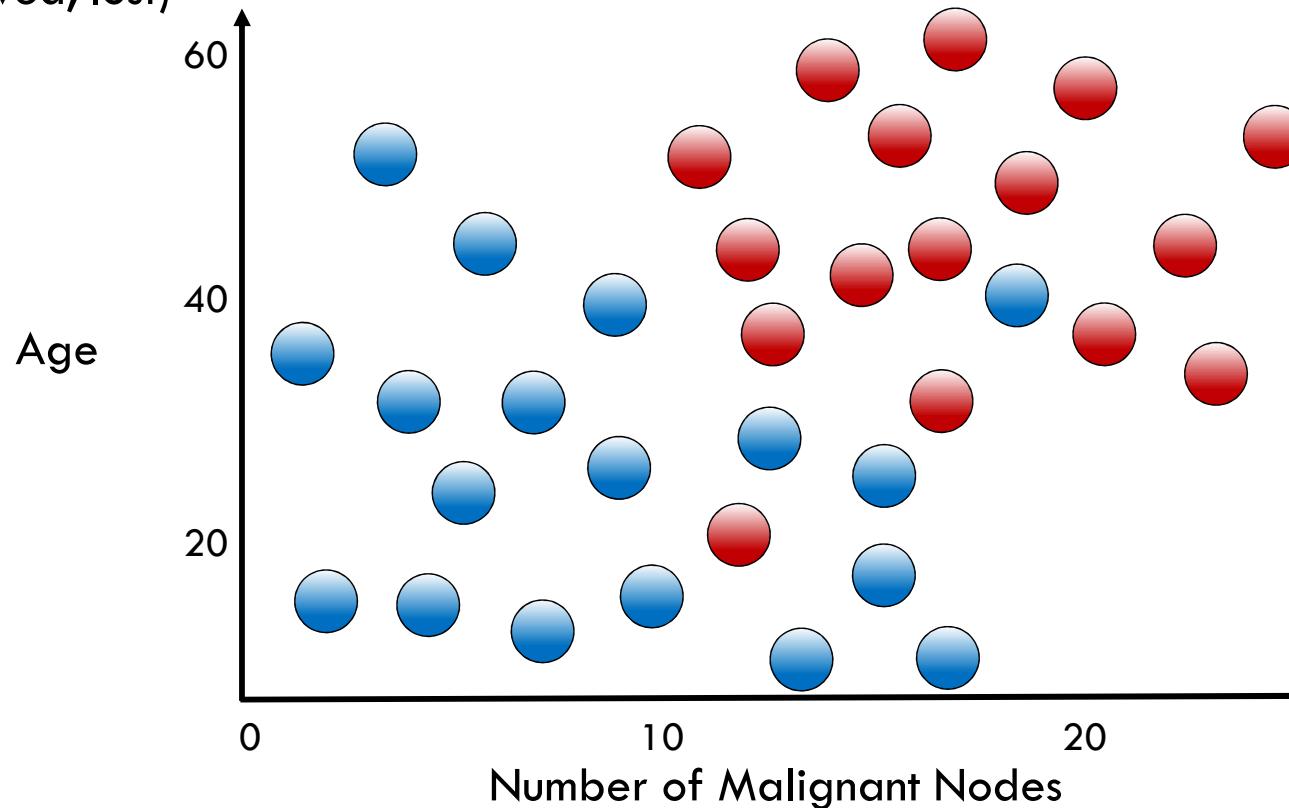


2.9. Regressão Logística

Classification with Logistic Regression

Two features (nodes, age)

Two labels (survived, lost)

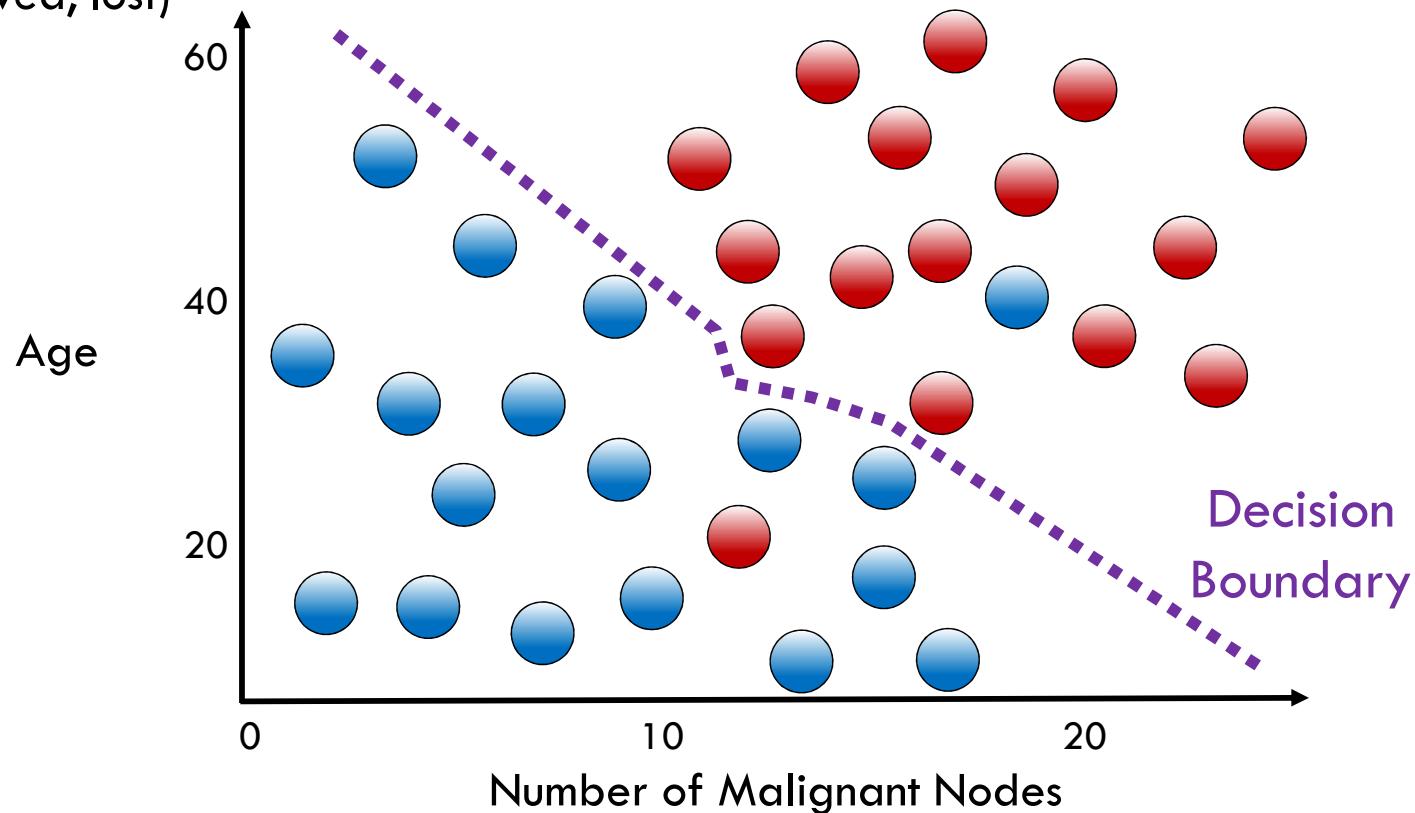


2.9. Regressão Logística

Classification with Logistic Regression

Two features (nodes, age)

Two labels (survived, lost)

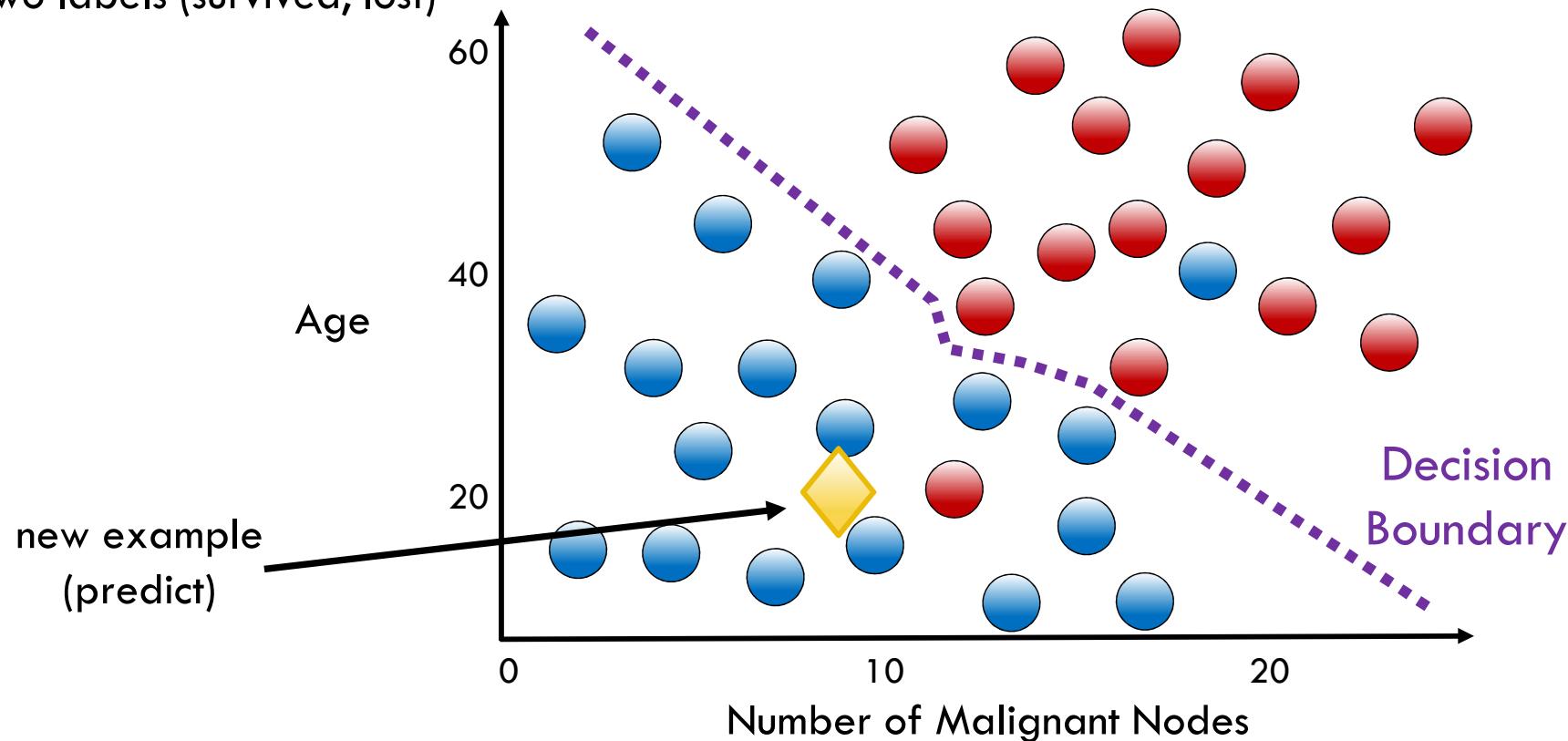


2.9. Regressão Logística

Classification with Logistic Regression

Two features (nodes, age)

Two labels (survived, lost)

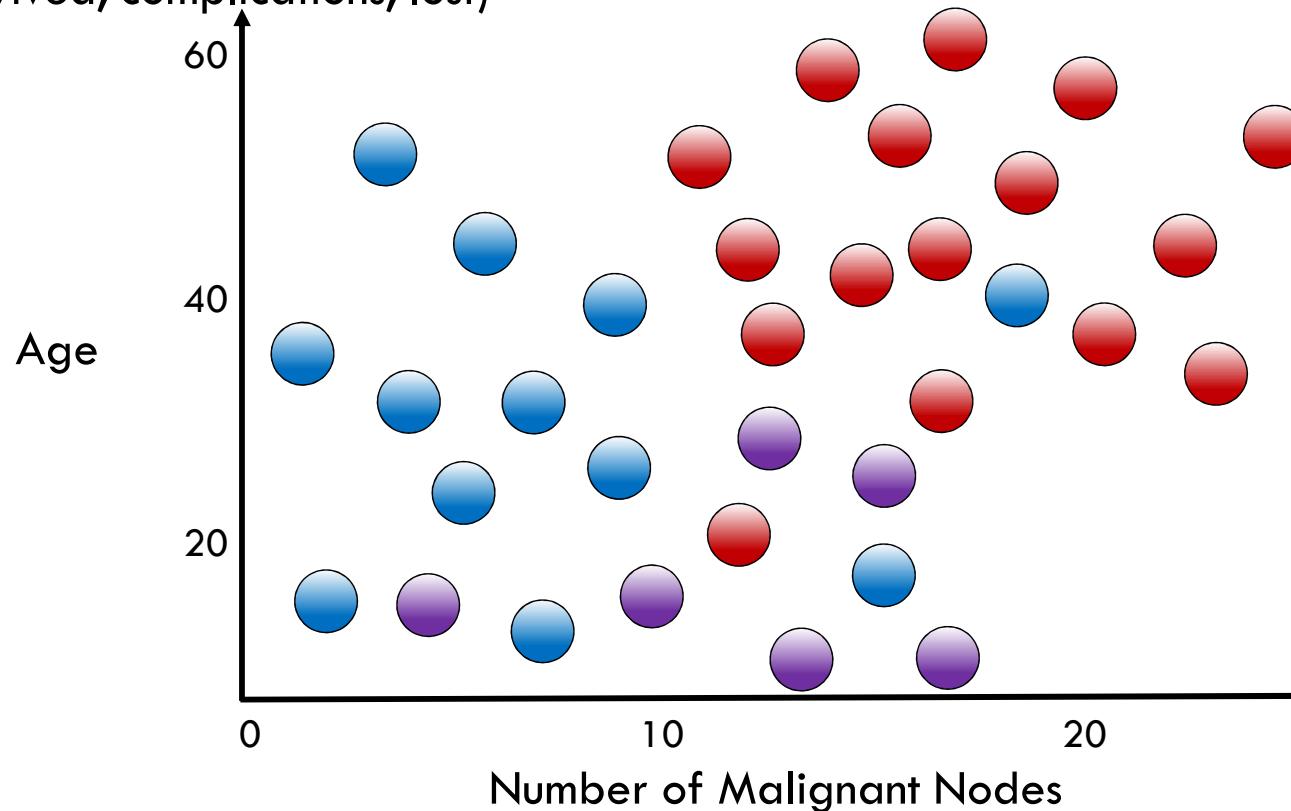


2.9. Regressão Logística

Multiclass Classification with Logistic Regression

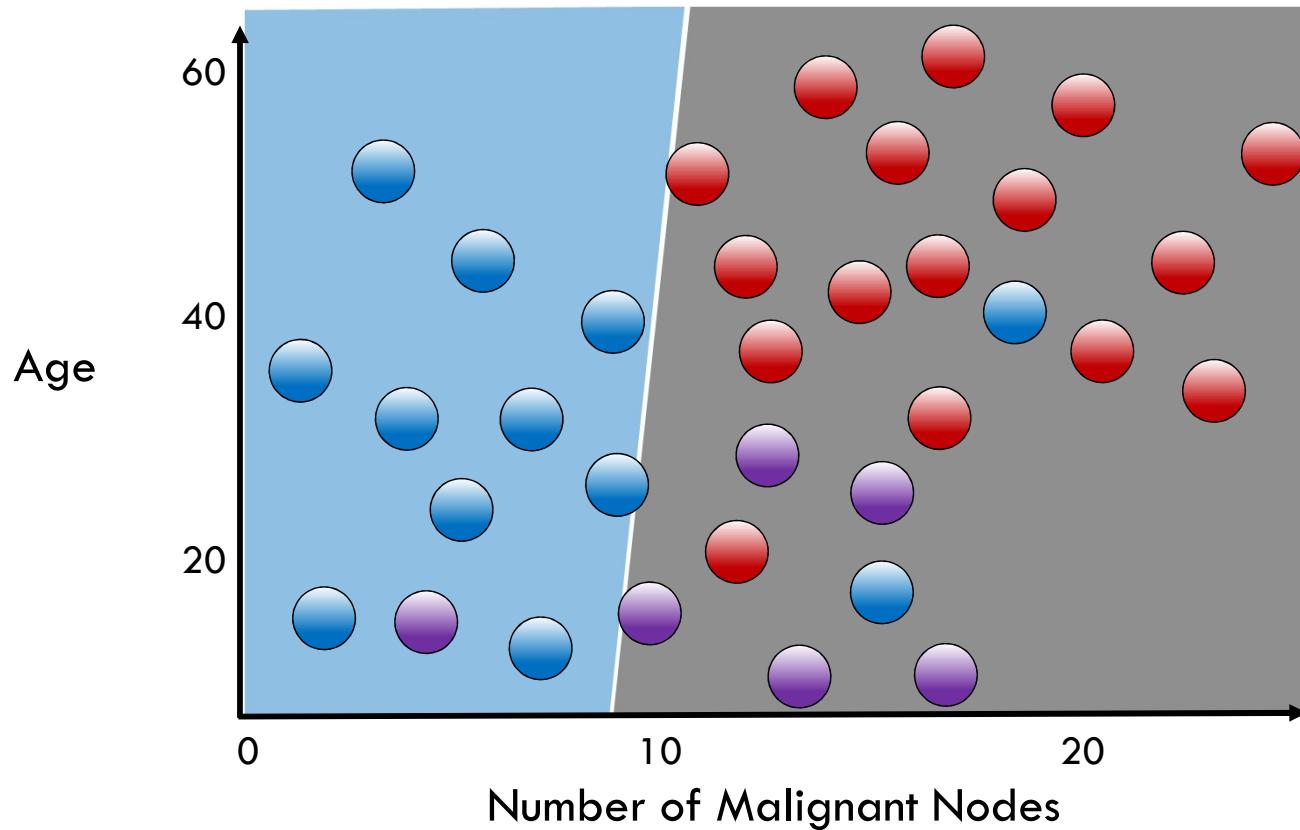
Two features (nodes, age)

Three labels (survived, complications, lost)



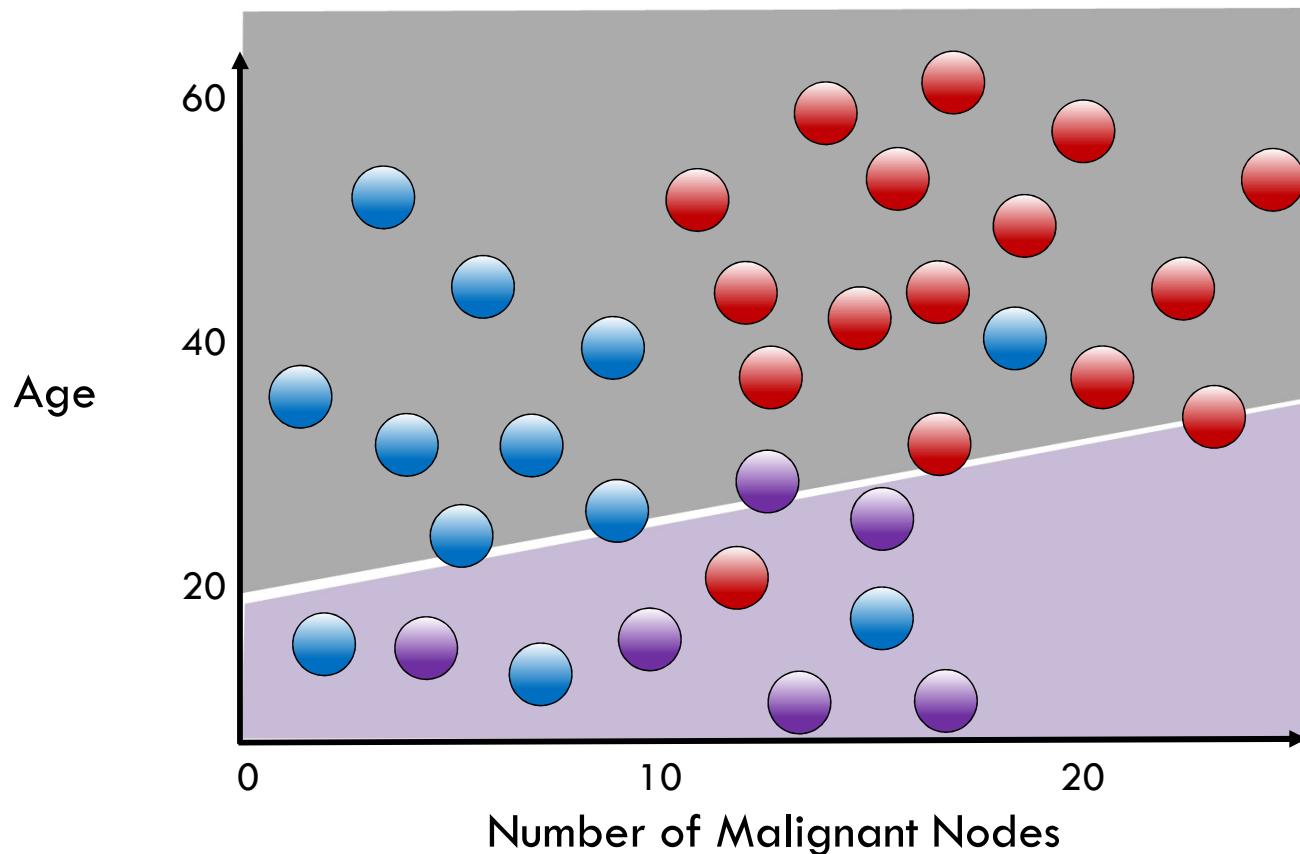
2.9. Regressão Logística

One vs All: Survived vs All



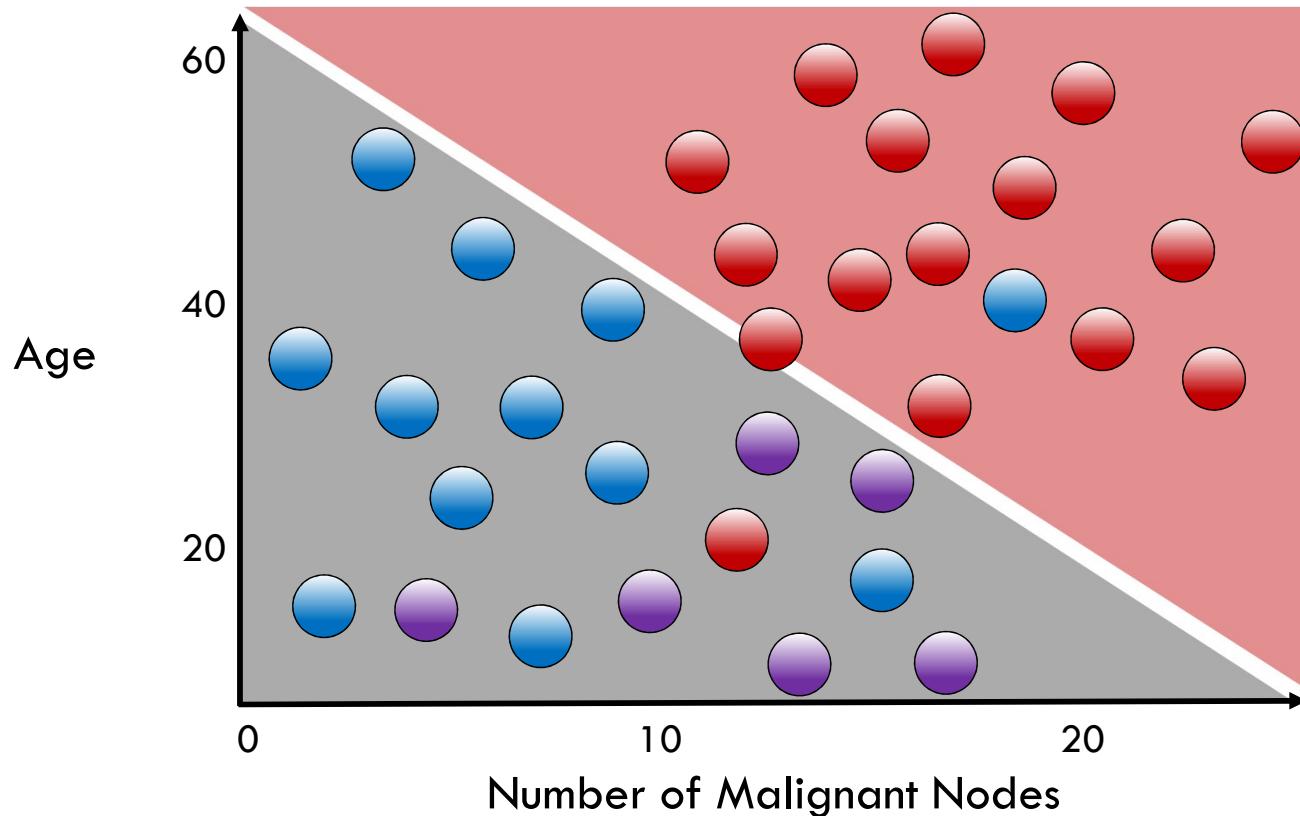
2.9. Regressão Logística

One vs All: Complications vs All



2.9. Regressão Logística

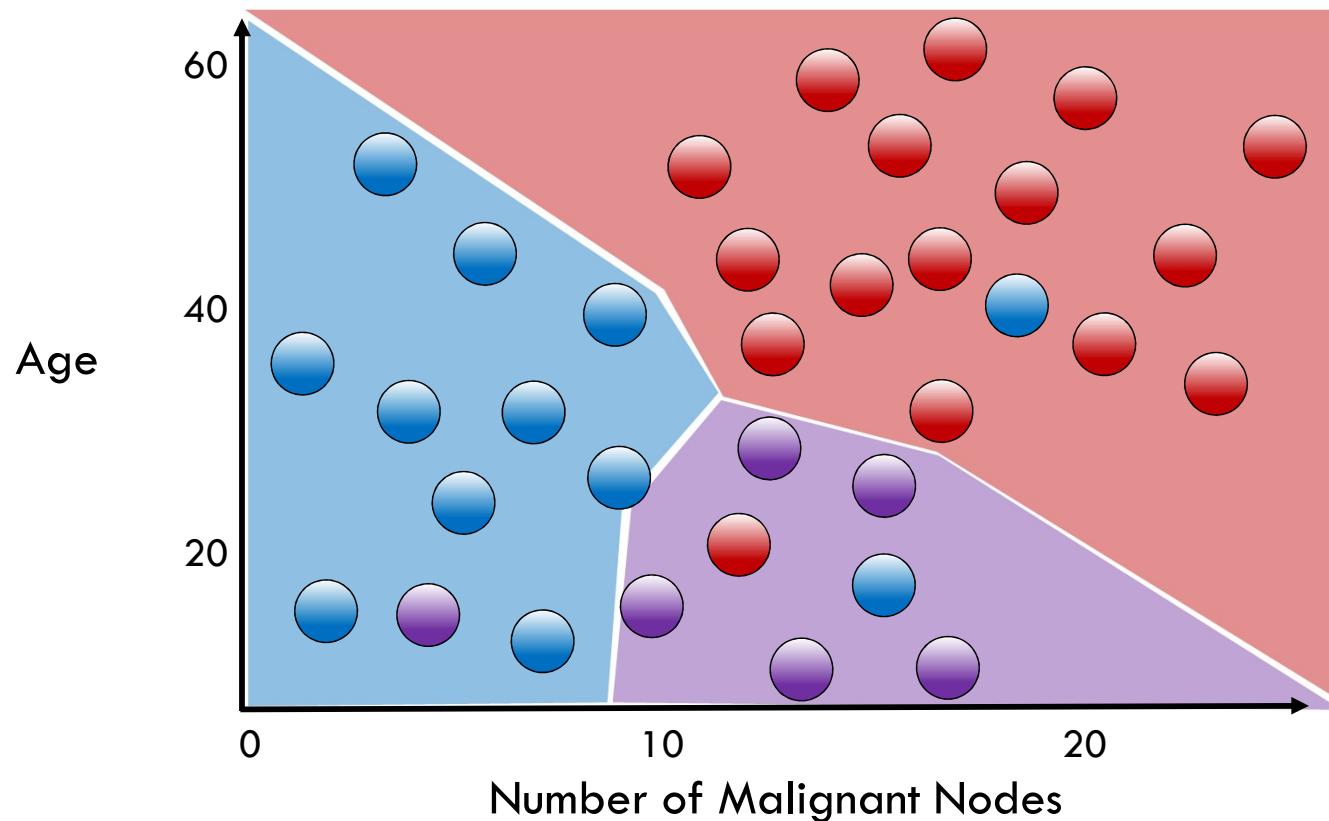
One vs All: Loss vs All



2.9. Regressão Logística

Multiclass Decision Boundary

Assign most probable class to each region



2.9. Regressão Logística

Logistic Regression: The Syntax

Import the class containing the classification method

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

2.9. Regressão Logística

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)
```

2.9. Regressão Logística

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

2.9. Regressão Logística

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)
```

2.9. Regressão Logística

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)
```

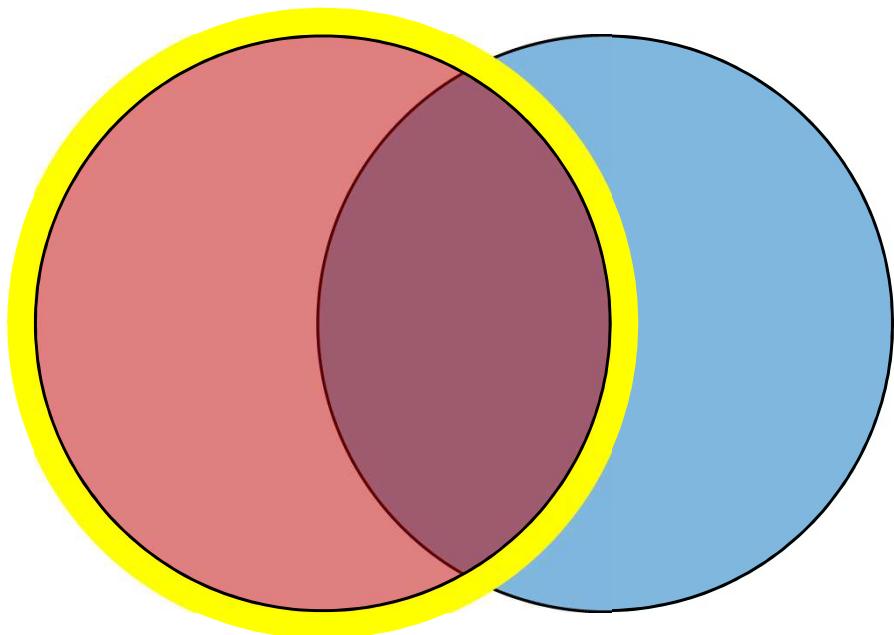
```
y_predict = LR.predict(X_test)
```

Tune regularization parameters with cross-validation: [LogisticRegressionCV](#).



2.10. Modelos Bayesianos

Probability Basics

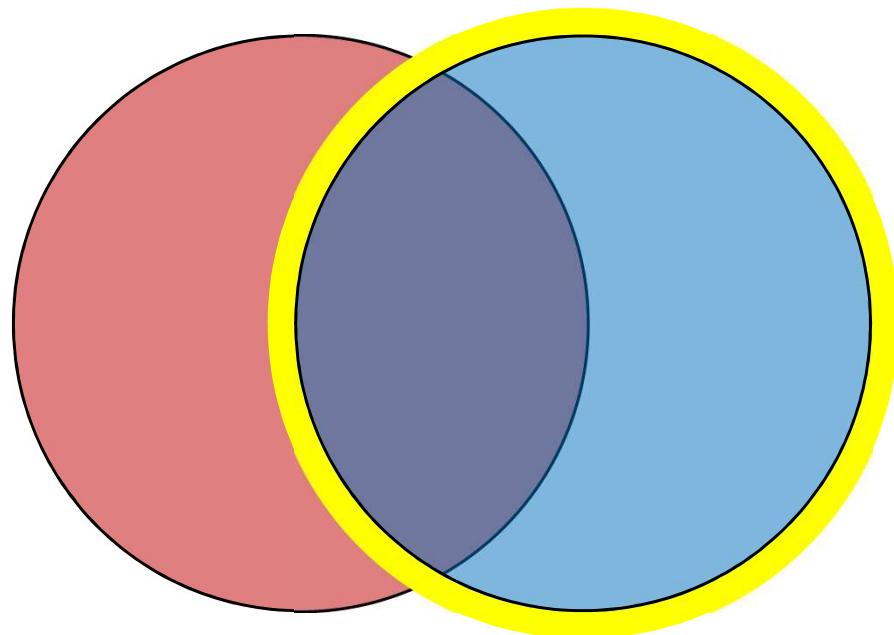


$$P(X)$$

- Single event probability:

$$P(X)$$

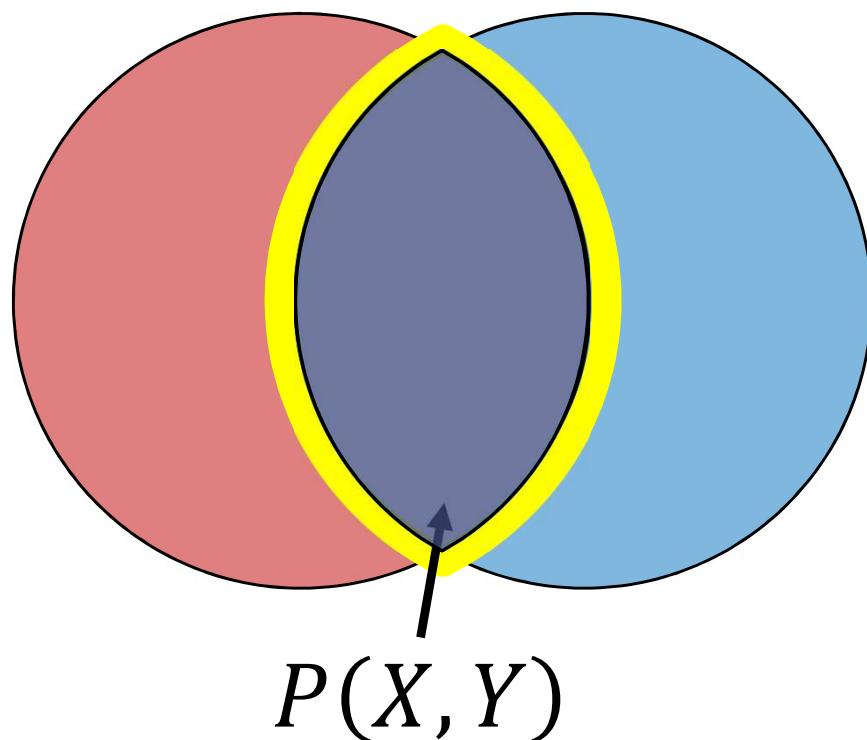
Probability Basics



$$P(X) \quad P(Y)$$

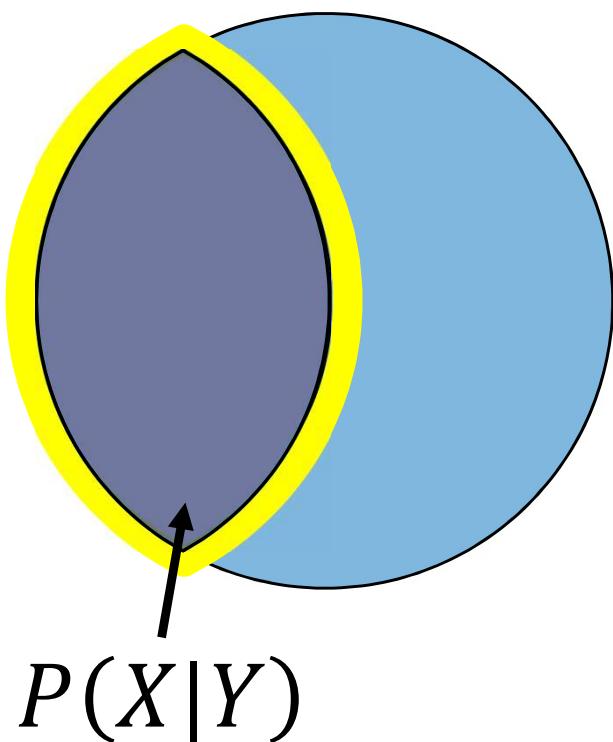
- Single event probability:
 $P(X), P(Y)$

Probability Basics



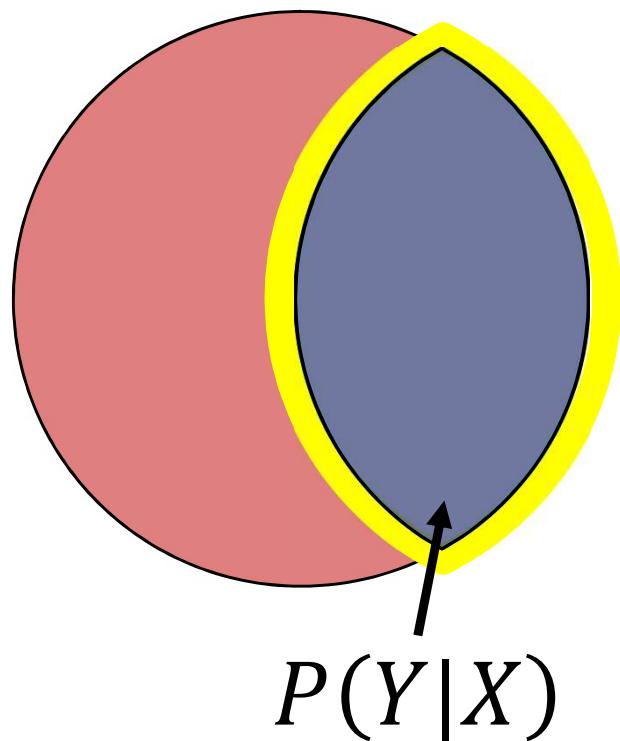
- Single event probability:
 $P(X), P(Y)$
- Joint event probability:
 $P(X, Y)$

Probability Basics



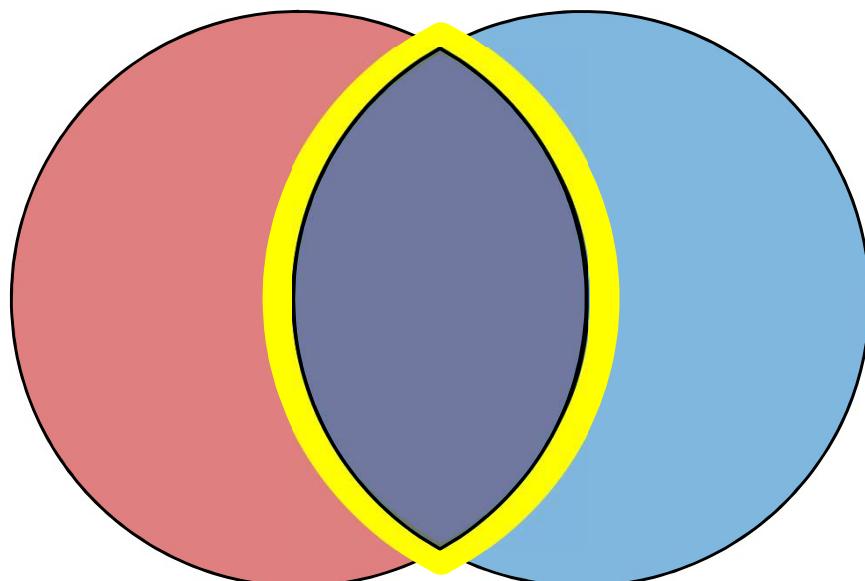
- Single event probability:
 $P(X), P(Y)$
- Joint event probability:
 $P(X, Y)$
- Conditional probability:
 $P(X|Y)$

Probability Basics



- Single event probability:
 $P(X), P(Y)$
- Joint event probability:
 $P(X, Y)$
- Conditional probability:
 $P(X|Y), P(Y|X)$

Probability Basics



- Single event probability:

$$P(X), P(Y)$$

- Joint event probability:

$$P(X, Y)$$

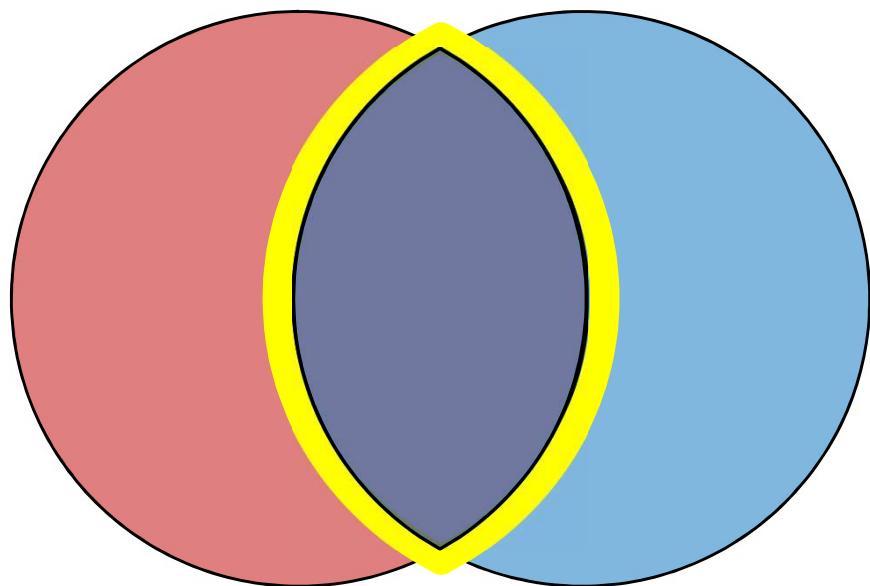
- Conditional probability:

$$P(X|Y), P(Y|X)$$

- Joint and conditional relationship:

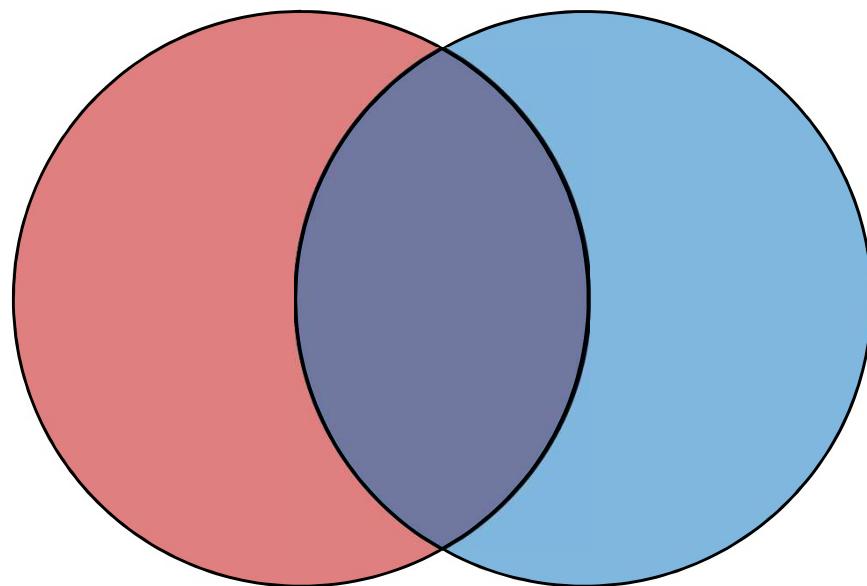
$$P(X, Y) = P(Y|X) * P(X) = P(X|Y) * P(Y)$$

Bayes Theorem Derivation



- By conditional and joint relationship:
$$P(Y|X) * P(X) = P(X|Y) * P(Y)$$

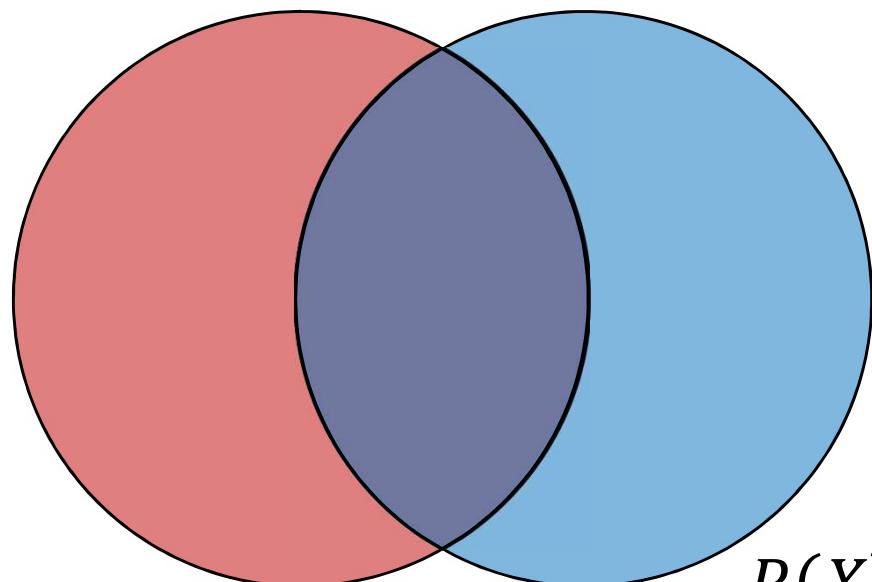
Bayes Theorem Derivation



- Use conditional and joint relationship:
$$P(Y|X) * P(X) = P(X|Y) * P(Y)$$
- To invert conditional probability:

$$P(Y|X) = \frac{P(X|Y) * P(Y)}{P(X)}$$

Bayes Theorem Derivation



- Use conditional and joint relationship:

$$P(Y|X) * P(X) = P(X|Y) * P(Y)$$

- To invert conditional probability:

$$P(Y|X) = \frac{P(X|Y) * P(Y)}{P(X)}$$

$$P(X) = \sum_Z P(X, Z) = \sum_Z P(X|Z) * P(Z)$$

Bayes Theorem

$$P(Y|X) = \frac{P(X|Y) * P(Y)}{P(X)}$$

Bayes Theorem

$$P(Y|X) = \frac{P(X|Y) * P(Y)}{P(X)}$$

$$posterior = \frac{likelihood * prior}{evidence}$$

Naïve Bayes Classification

$$P(Y|X) = \frac{P(X|Y) * P(Y)}{P(X)}$$

*likelihood * prior*

$$posterior = \frac{\textit{likelihood} * \textit{prior}}{\textit{evidence}}$$

Training Naïve Bayes

- For each class (C), calculate probability given features (X)

$$P(C|X) = P(X|C) * P(C)$$

The diagram shows the equation $P(C|X) = P(X|C) * P(C)$. Below the equation, the word "Class" is written in red and "Features" is written in blue. Two arrows point from these words to the corresponding variables in the equation: one arrow points from "Class" to C , and another arrow points from "Features" to X .

Class Features

Training Naïve Bayes: The Naïve Assumption

- For each class (C), $P(C|X) = P(X|C) * P(C)$
calculate probability given
features (X)
- Difficult to calculate joint probabilities produced by
expanding for all features $P(C|X) = P(X_1, X_2, \dots, X_n|C) * P(C)$
 $P(X_1|X_2, \dots, X_n, C) * P(X_2, \dots, X_n|C) * P(C)$
 \dots

Training Naïve Bayes: The Naïve Assumption

- For each class (C), calculate probability given features (X)
$$P(C|X) = P(X|C) * P(C)$$
- **Solution:** assume all features independent of each other
$$P(C|X) = P(X_1|C) * P(X_2|C) * P(X_n|C) * P(C)$$

Training Naïve Bayes: The Naïve Assumption

- For each class (C), calculate probability given features (X)
- **Solution:** assume all features independent of each other
- This is the "naïve" assumption

$$P(C|X) = P(X|C) * P(C)$$

$$P(C|X) = P(X_1|C) * P(X_2|C) * P(X_n|C) * P(C)$$

$$P(C|X) = P(C) \prod_{i=1}^n P(X_i|C)$$

Training Naïve Bayes

- For each class (C), calculate probability given features (X)
- Class assignment is selected based on *maximum a posteriori* (MAP) rule

$$P(C|X) = P(X|C) * P(C)$$

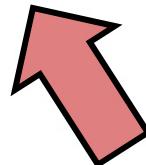
$$\underset{k \in \{1, \dots K\}}{\operatorname{argmax}} P(C_k) \prod_{i=1}^n P(X_i|C_k)$$

Training Naïve Bayes

- For each class (C), calculate probability given features (X)
- Class assignment is selected based on *maximum a posteriori* (MAP) rule

$$P(C|X) = P(X|C) * P(C)$$

$$\underset{k \in \{1, \dots K\}}{\operatorname{argmax}} P(C_k) \prod_{i=1}^n P(X_i|C_k)$$



Means select potential class
with largest value

The Log Trick

- Multiplying many values together causes computational instability (underflows)

$$\underset{k \in \{1, \dots K\}}{\operatorname{argmax}} P(\textcolor{red}{C}_k) \prod_{i=1}^n P(\textcolor{blue}{X}_i | \textcolor{red}{C}_k)$$

The Log Trick

- Multiplying many values together causes computational instability (underflows)
- Work with log values and sum the results

$$\underset{k \in \{1, \dots K\}}{\operatorname{argmax}} P(\textcolor{red}{C}_k) \prod_{i=1}^n P(\textcolor{blue}{X}_i | \textcolor{red}{C}_k)$$

$$\log(P(\textcolor{red}{C}_k)) \sum_{i=1}^n \log(P(\textcolor{blue}{X}_i | \textcolor{red}{C}_k))$$

Example: Predicting Tennis With Naïve Bayes

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Example: Training Naïve Bayes Tennis Model

$$P(\text{Play}=\text{Yes}) = 9/14 \quad P(\text{Play}=\text{No}) = 5/14$$

Create probability lookup tables based on training data

Example: Training Naïve Bayes Tennis Model

$$P(\text{Play}=\text{Yes}) = 9/14$$

Outlook	Play=Yes	Play=No
Sunny	2/9	3/5
Overcast	4/9	0/5
Rain	3/9	2/5

$$P(\text{Play}=\text{No}) = 5/14$$

Temperature	Play=Yes	Play=No
Hot	2/9	2/5
Mild	4/9	2/5
Cool	3/9	1/5

Create probability lookup tables based on training data

Example: Training Naïve Bayes Tennis Model

$$P(\text{Play}=\text{Yes}) = 9/14$$

Outlook	Play=Yes	Play=No
Sunny	2/9	3/5
Overcast	4/9	0/5
Rain	3/9	2/5

Humidity	Play=Yes	Play=No
High	3/9	4/5
Normal	6/9	1/5

$$P(\text{Play}=\text{No}) = 5/14$$

Temperature	Play=Yes	Play=No
Hot	2/9	2/5
Mild	4/9	2/5
Cool	3/9	1/5

Wind	Play=Yes	Play=No
Strong	3/9	3/5
Weak	6/9	2/5

Create probability lookup tables based on training data

Example: Predicting Tennis With Naïve Bayes

Predict outcome for the following:

$x' = (\text{Outlook}=\text{Sunny}, \text{Temperature}=\text{Cool}, \text{Humidity}=\text{High}, \text{Wind}=\text{Strong})$

$$P(\text{yes}|\text{sunny, cool, high, strong}) = P(\text{sunny|yes}) * P(\text{cool|yes}) * \\ P(\text{high|yes}) * P(\text{strong|yes}) * P(\text{yes})$$

$$P(\text{no}|\text{sunny, cool, high, strong}) = P(\text{sunny|no}) * P(\text{cool|no}) * \\ P(\text{high|no}) * P(\text{strong|no}) * P(\text{no})$$

Example: Predicting Tennis With Naïve Bayes

Predict outcome for the following:

$x'=(\text{Outlook}=\text{Sunny}, \text{Temperature}=\text{Cool}, \text{Humidity}=\text{High}, \text{Wind}=\text{Strong})$

Feature	Play=Yes	Play=No
Outlook=Sunny	2/9	3/5

Example: Predicting Tennis With Naïve Bayes

Predict outcome for the following:

$x'=(\text{Outlook}=\text{Sunny}, \text{Temperature}=\text{Cool}, \text{Humidity}=\text{High}, \text{Wind}=\text{Strong})$

Feature	Play=Yes	Play=No
Outlook=Sunny	2/9	3/5
Temperature=Cool	3/9	1/5
Humidity=High	3/9	4/5
Wind=Strong	3/9	3/5
Overall Label	9/14	5/14

Example: Predicting Tennis With Naïve Bayes

Predict outcome for the following:

$x'=(\text{Outlook}=\text{Sunny}, \text{Temperature}=\text{Cool}, \text{Humidity}=\text{High}, \text{Wind}=\text{Strong})$

Feature	Play=Yes	Play>No
Outlook=Sunny	2/9	3/5
Temperature=Cool	3/9	1/5
Humidity=High	3/9	4/5
Wind=Strong	3/9	3/5
Overall Label	9/14	5/14
Probability	0.0053	0.0206

Example: Predicting Tennis With Naïve Bayes

Predict outcome for the following:

$x'=(\text{Outlook}=\text{Sunny}, \text{Temperature}=\text{Cool}, \text{Humidity}=\text{High}, \text{Wind}=\text{Strong})$

Feature	Play=Yes	Play>No
Outlook=Sunny	2/9	3/5
Temperature=Cool	3/9	1/5
Humidity=High	3/9	4/5
Wind=Strong	3/9	3/5
Overall Label	9/14	5/14
Probability	0.0053	0.0206

Laplace Smoothing

- **Problem:** categories with no entries result in a value of "0" for conditional probability

$$P(C|X) = P(X_1|C) * P(X_2|C) * P(C)$$

Laplace Smoothing

- **Problem:** categories with no entries result in a value of "0" for conditional probability

$$P(C|X) = \frac{0}{P(X_1|C) * P(X_2|C) * P(C)}$$

Laplace Smoothing

- **Problem:** categories with no entries result in a value of "0" for conditional probability
- **Solution:** add "1" to numerator and denominator of empty categories

$$P(C|X) = \frac{0}{P(X_1|C)} * P(X_2|C) * P(C)$$

$$P(X_1|C) = \frac{1}{Count(C) + n}$$

$$P(X_2|C) = \frac{Count(X_2 \& C) + 1}{Count(C) + m}$$

Types of Naïve Bayes

Naïve Bayes Model

Data Type

Bernoulli

Binary (T/F)

Types of Naïve Bayes

Naïve Bayes Model

Data Type

Bernoulli

Binary (T/F)

Multinomial

Discrete (e.g. count)

Types of Naïve Bayes

Naïve Bayes Model

Data Type

Bernoulli

Binary (T/F)

Multinomial

Discrete (e.g. count)

Gaussian

Continuous

Combining Feature Types

Problem

- Model features contain different data types (continuous and categorical)

Combining Feature Types

Problem

- Model features contain different data types (continuous and categorical)

Solutions

- **Option 1:** Bin continuous features to create categorical ones and fit multinomial model

Combining Feature Types

Problem

- Model features contain different data types (continuous and categorical)

Solutions

- **Option 1:** Bin continuous features to create categorical ones and fit multinomial model
- **Option 2:** Fit Gaussian model on continuous features and multinomial on categorical features; combine to create "meta model" (week 10)

Distributed Computing with Naïve Bayes

- Well-suited for large data and distributed computing—limited parameters and log probabilities are a summation
- Scikit-Learn implementations contain a "partial_fit" method designed for out-of-core calculations

Naïve Bayes: The Syntax

Import the class containing the classification method

```
from sklearn.naive_bayes import BernoulliNB
```

Naïve Bayes: The Syntax

Import the class containing the classification method

```
from sklearn.naive_bayes import BernoulliNB
```

Create an instance of the class

```
BNB = BernoulliNB(alpha=1.0)
```

Naïve Bayes: The Syntax

Import the class containing the classification method

```
from sklearn.naive_bayes import BernoulliNB
```

Create an instance of the class

```
BNB = BernoulliNB(alpha=1.0)
```



Laplace smoothing
parameter

Naïve Bayes: The Syntax

Import the class containing the classification method

```
from sklearn.naive_bayes import BernoulliNB
```

Create an instance of the class

```
BNB = BernoulliNB(alpha=1.0)
```

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

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

```
y_predict = BNB.predict(X_test)
```

Naïve Bayes: The Syntax

Import the class containing the classification method

```
from sklearn.naive_bayes import BernoulliNB
```

Create an instance of the class

```
BNB = BernoulliNB(alpha=1.0)
```

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

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

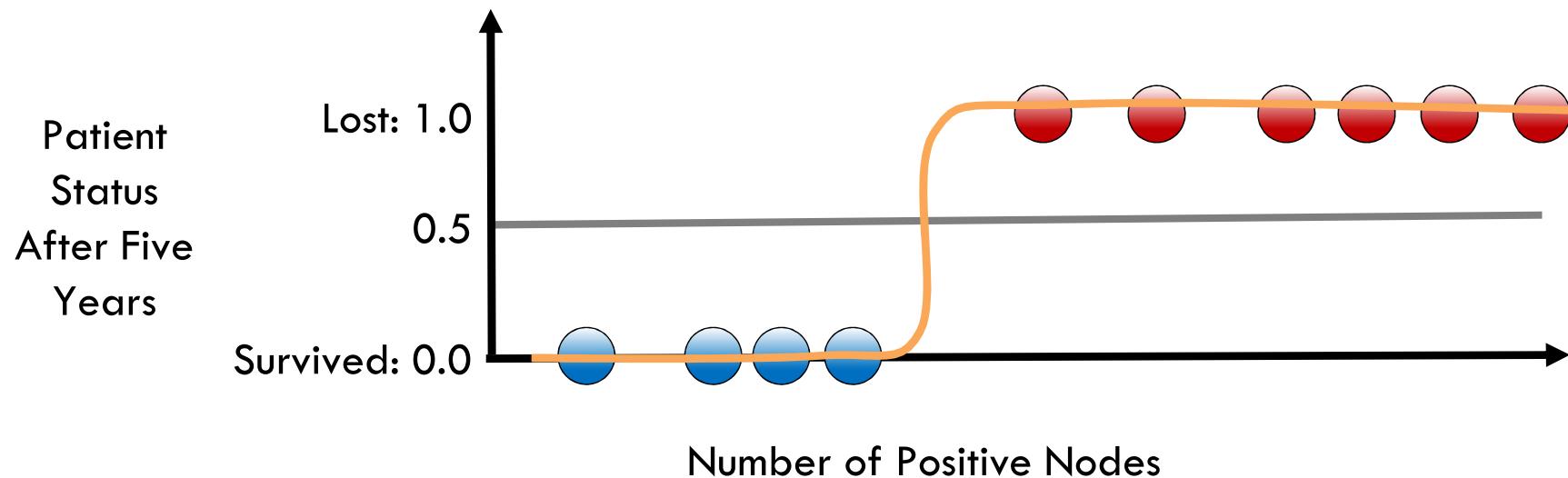
```
y_predict = BNB.predict(X_test)
```

Other naïve Bayes models: [MultinomialNB](#), [GaussianNB](#).



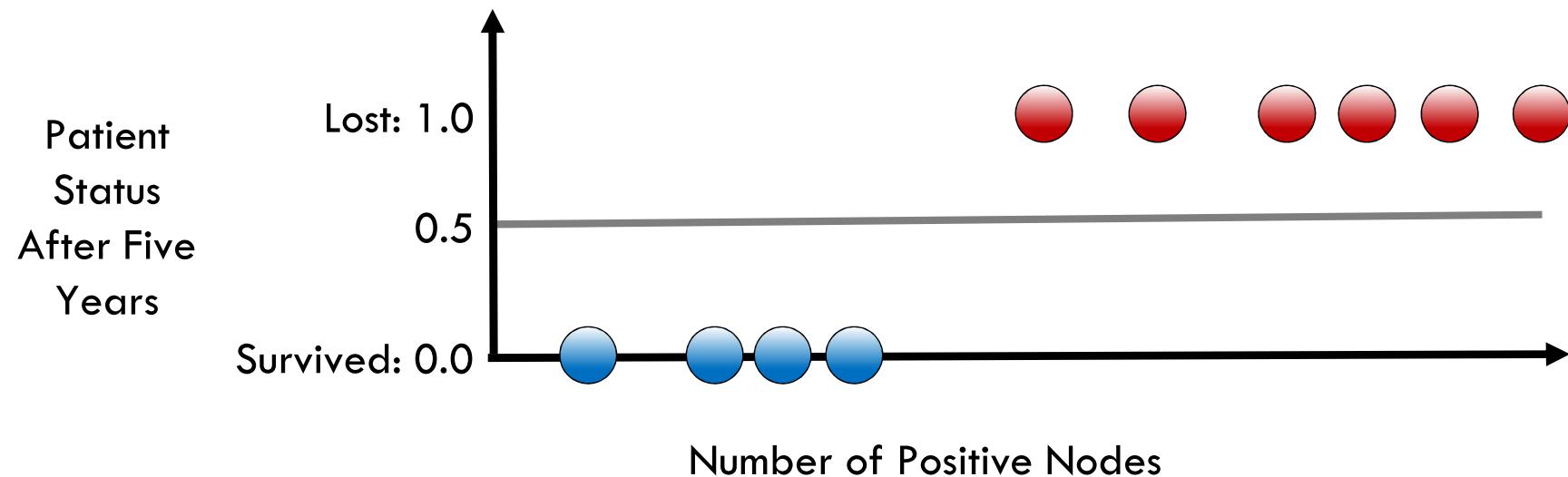
2.11. Support Vector Machines

Relationship to Logistic Regression

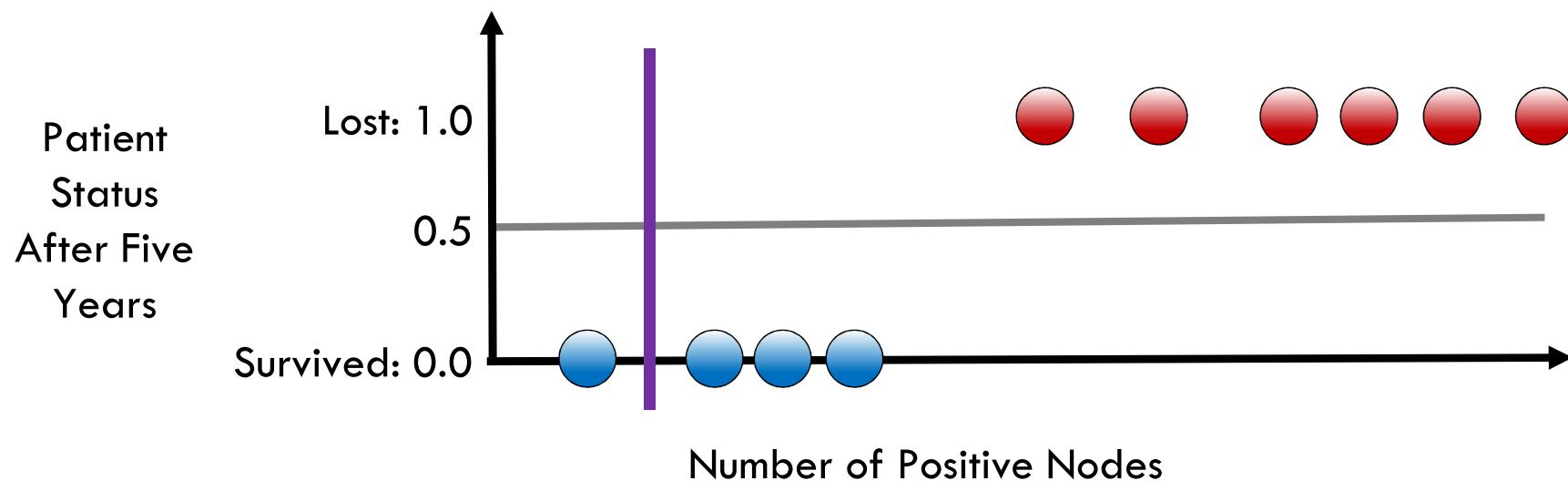


$$y_\beta(x) = \frac{1}{1+e^{-(\beta_0 + \beta_1 x + \varepsilon)}}$$

Support Vector Machines (SVM)

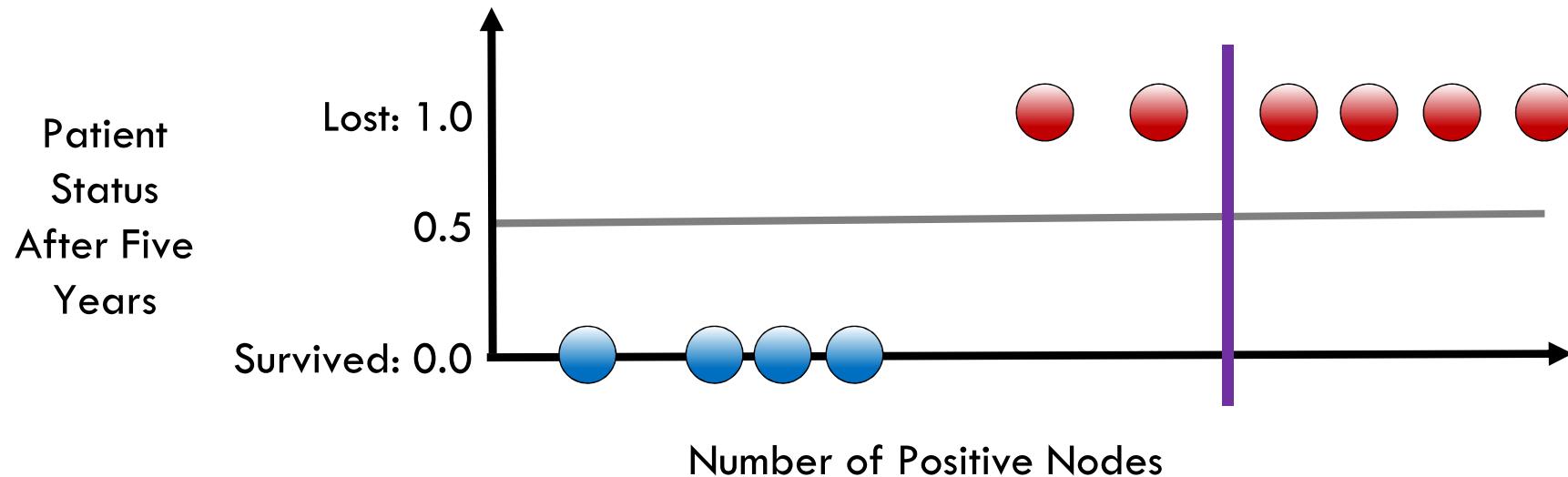


Support Vector Machines (SVM)



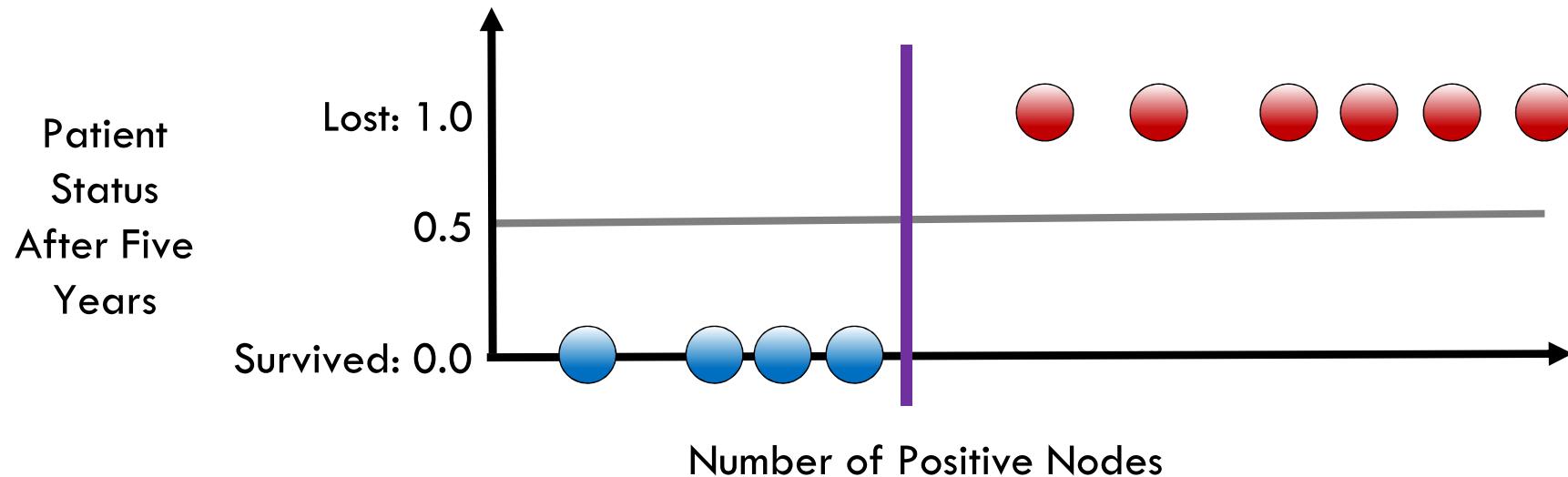
Three misclassifications

Support Vector Machines (SVM)



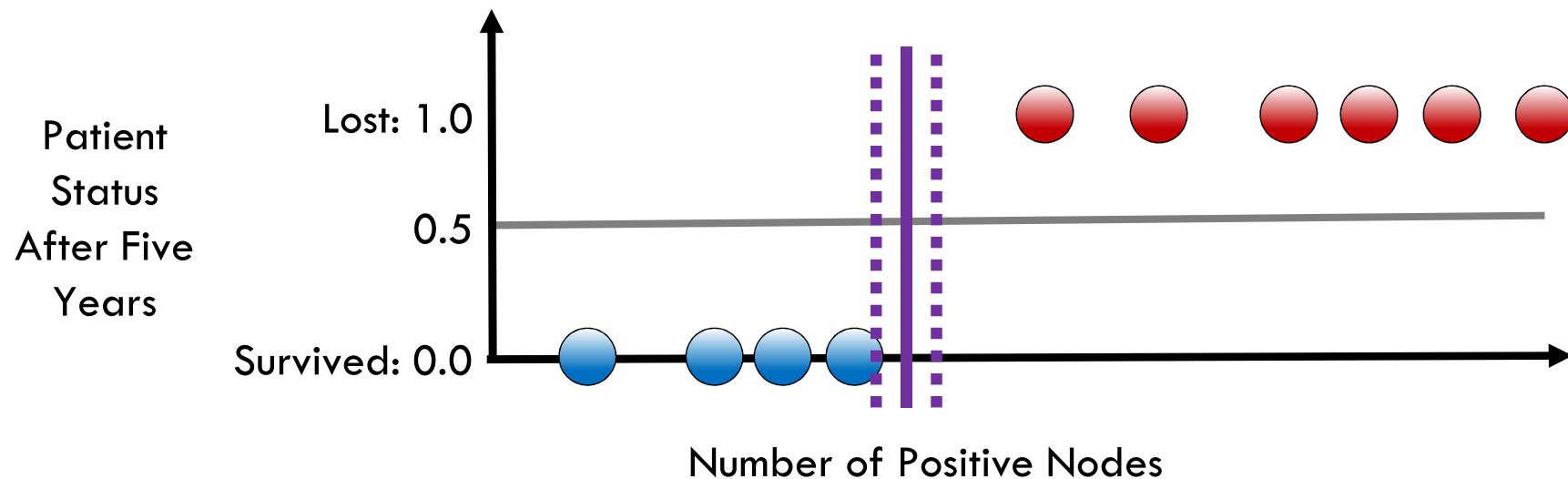
Two misclassifications

Support Vector Machines (SVM)



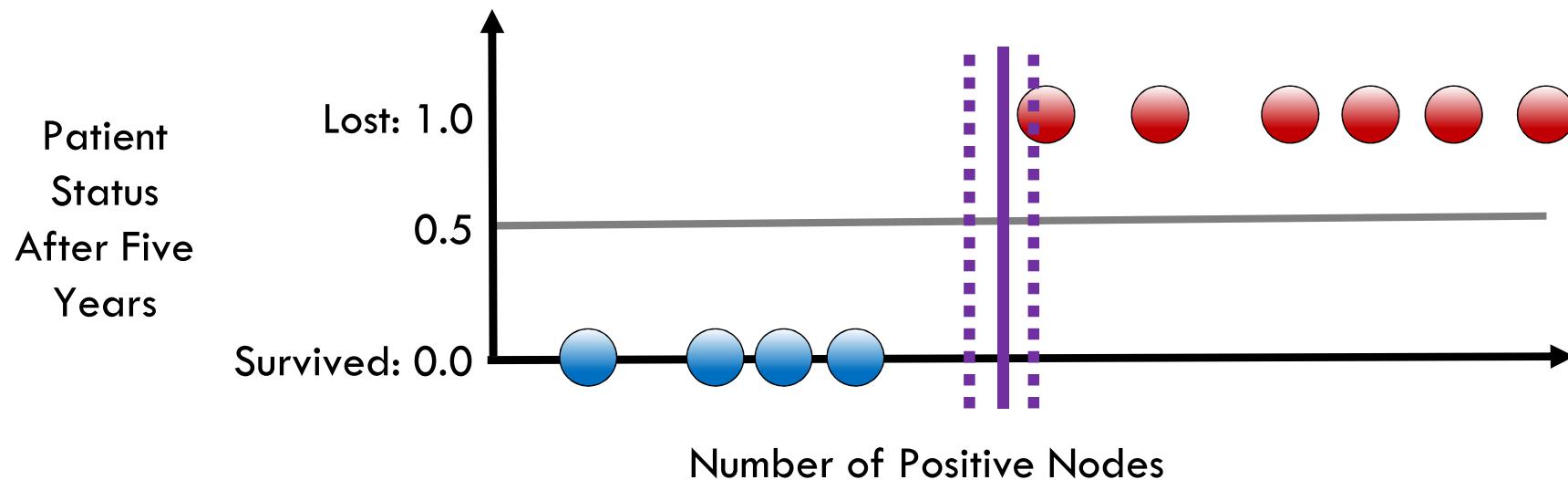
No misclassifications

Support Vector Machines (SVM)



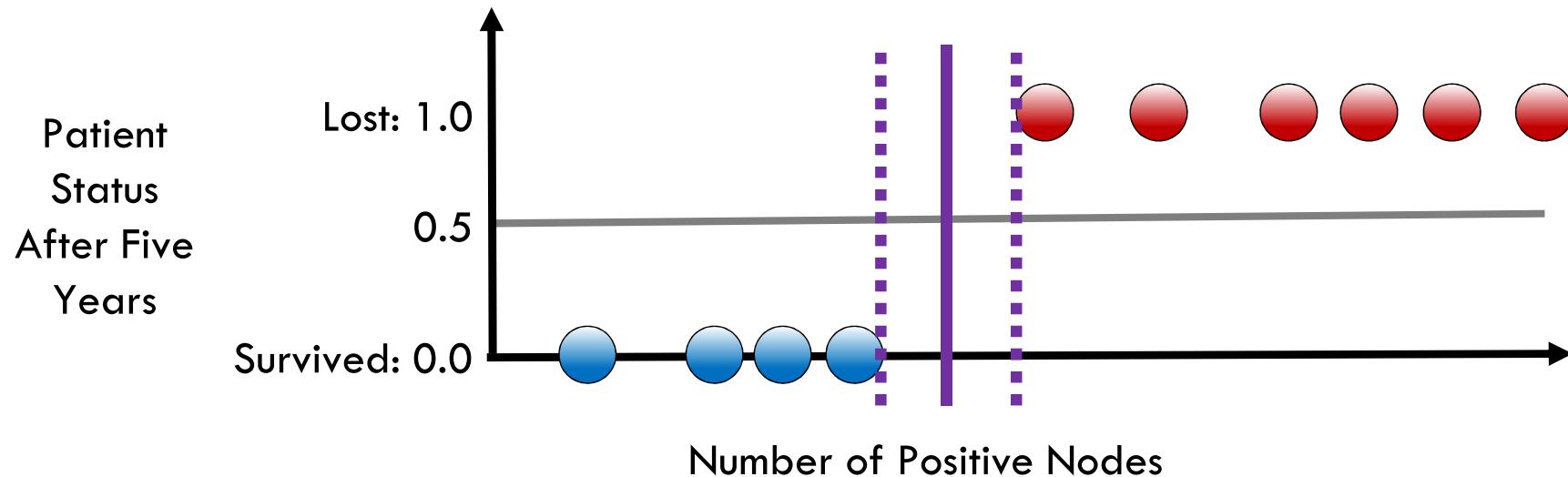
No misclassifications—but is this the best position?

Support Vector Machines (SVM)



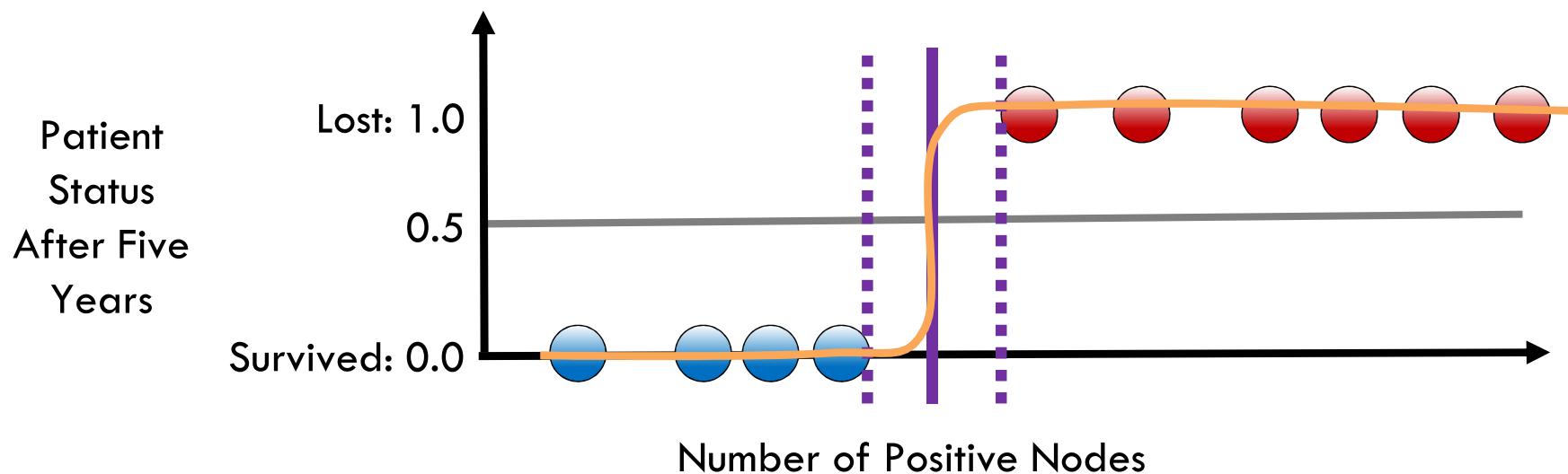
No misclassifications—but is this the best position?

Support Vector Machines (SVM)



Maximize the region between classes

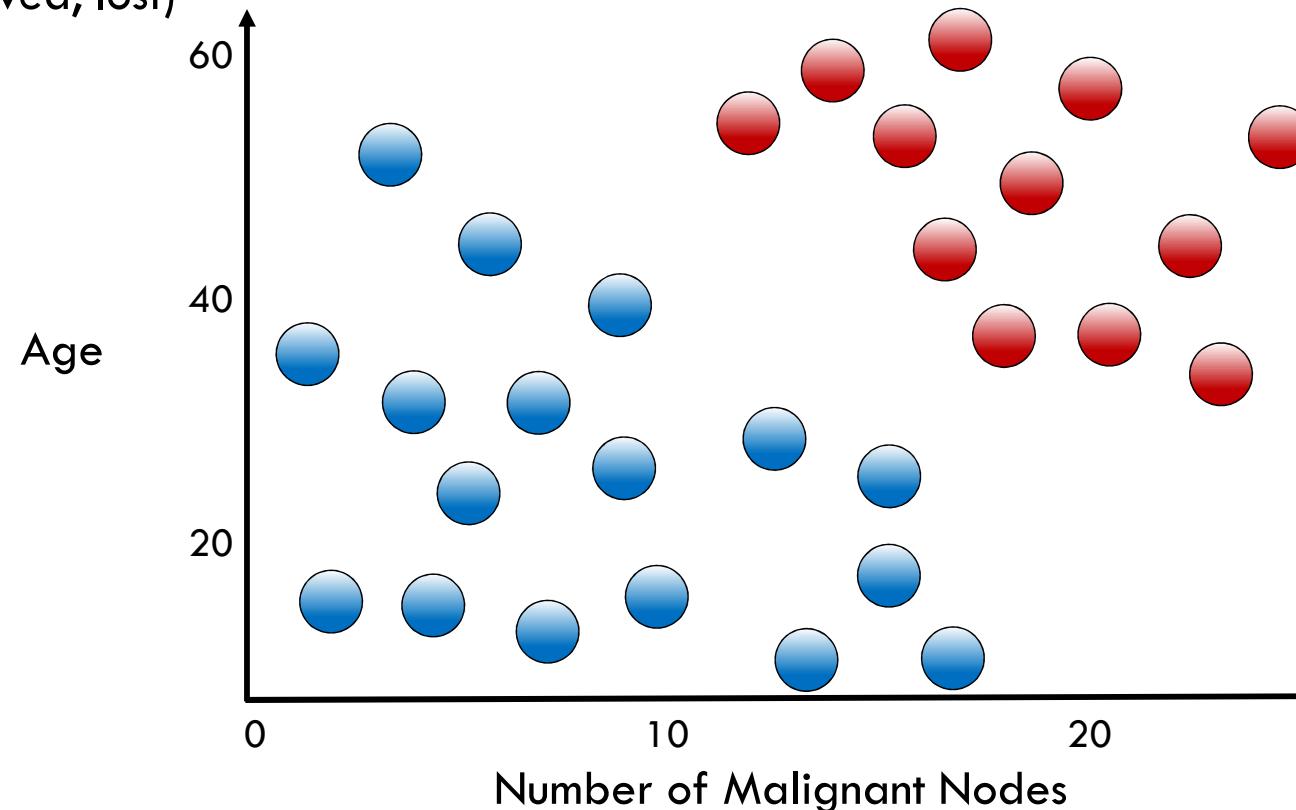
Similarity Between Logistic Regression and SVM



Classification with SVMs

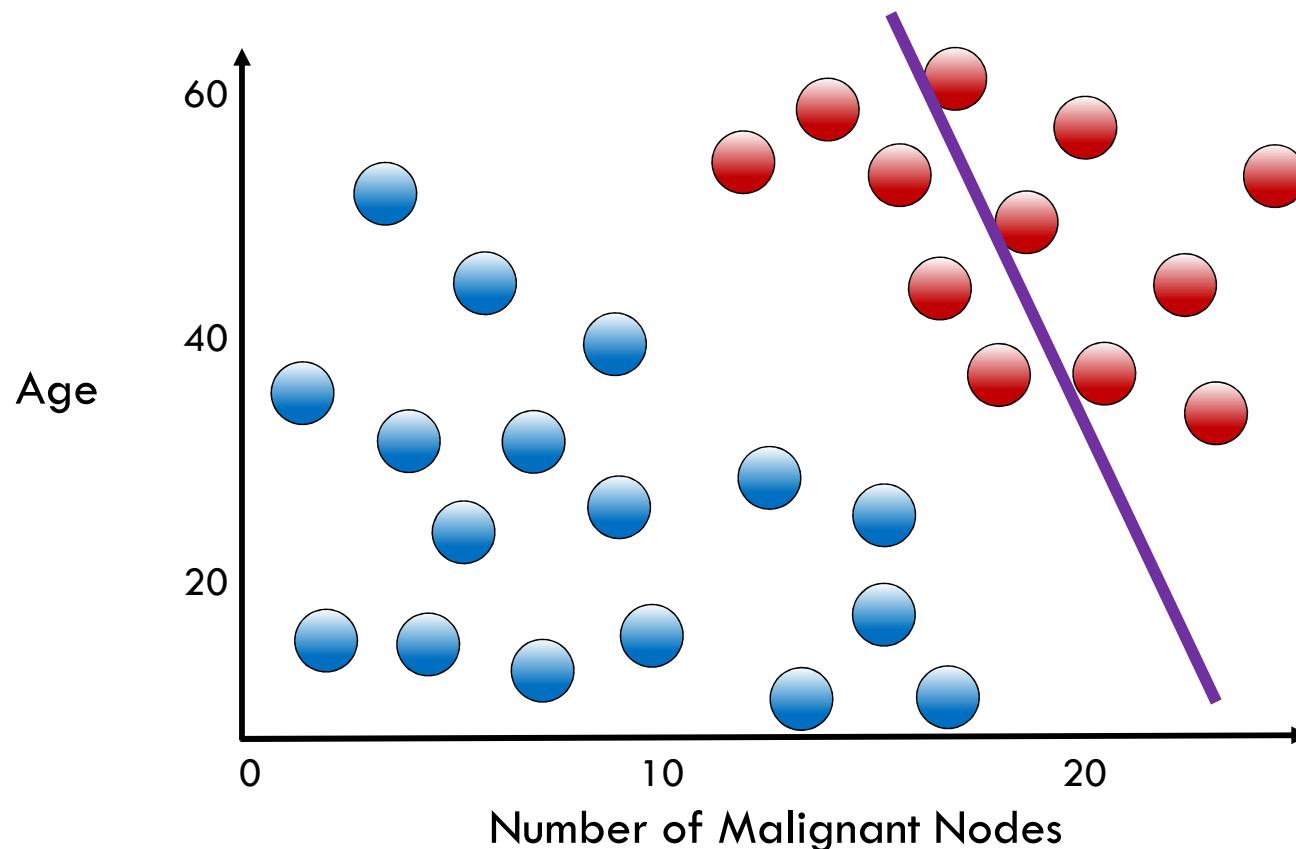
Two features (nodes, age)

Two labels (survived, lost)



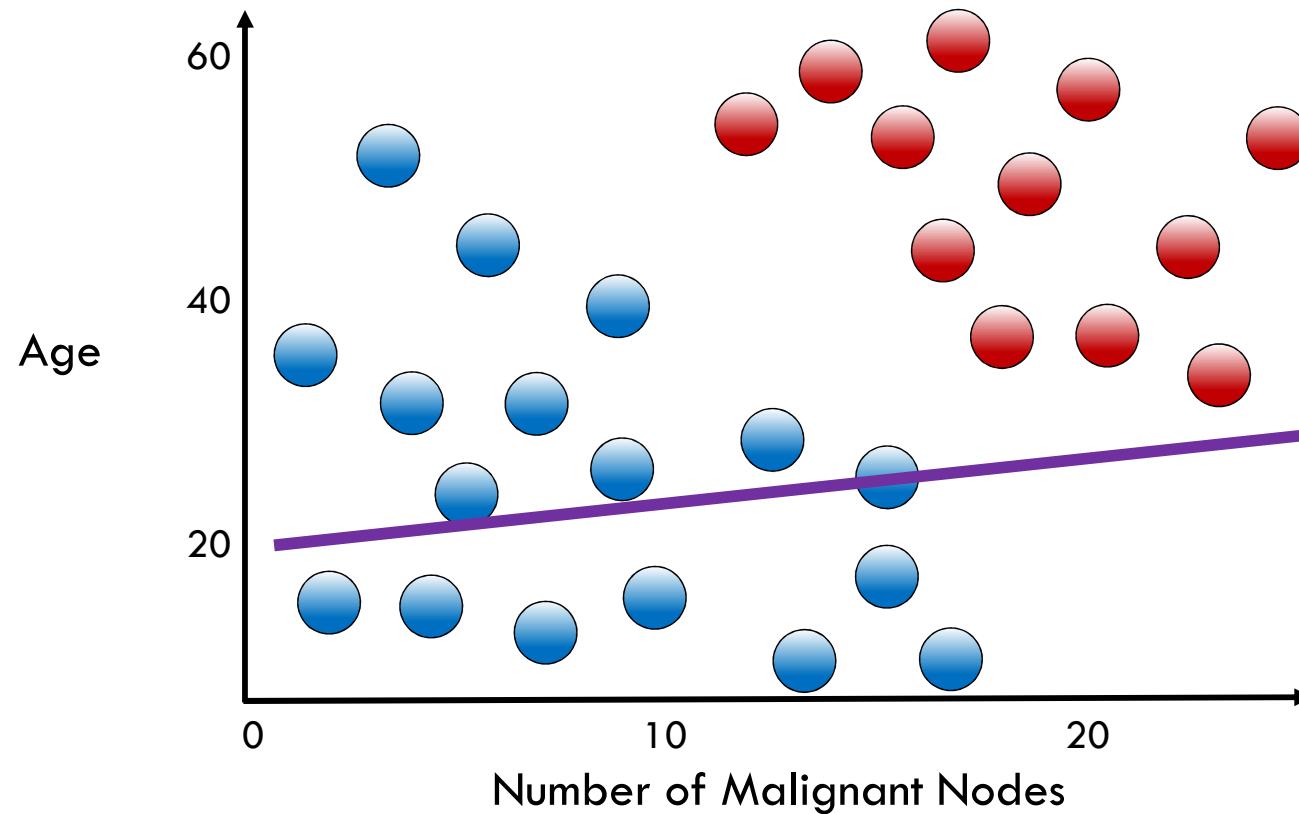
Classification with SVMs

Find the line that best separates
classes



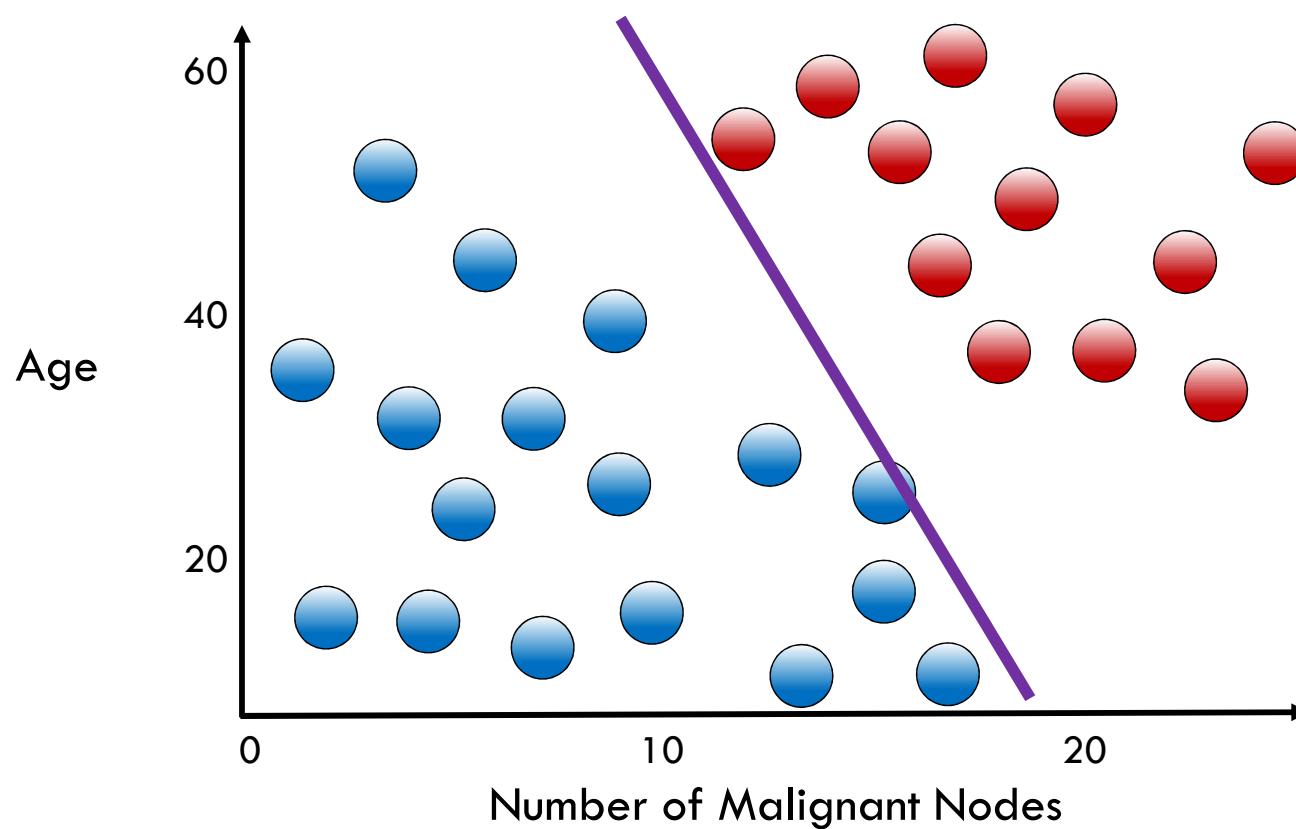
Classification with SVMs

Find the line that best separates
classes



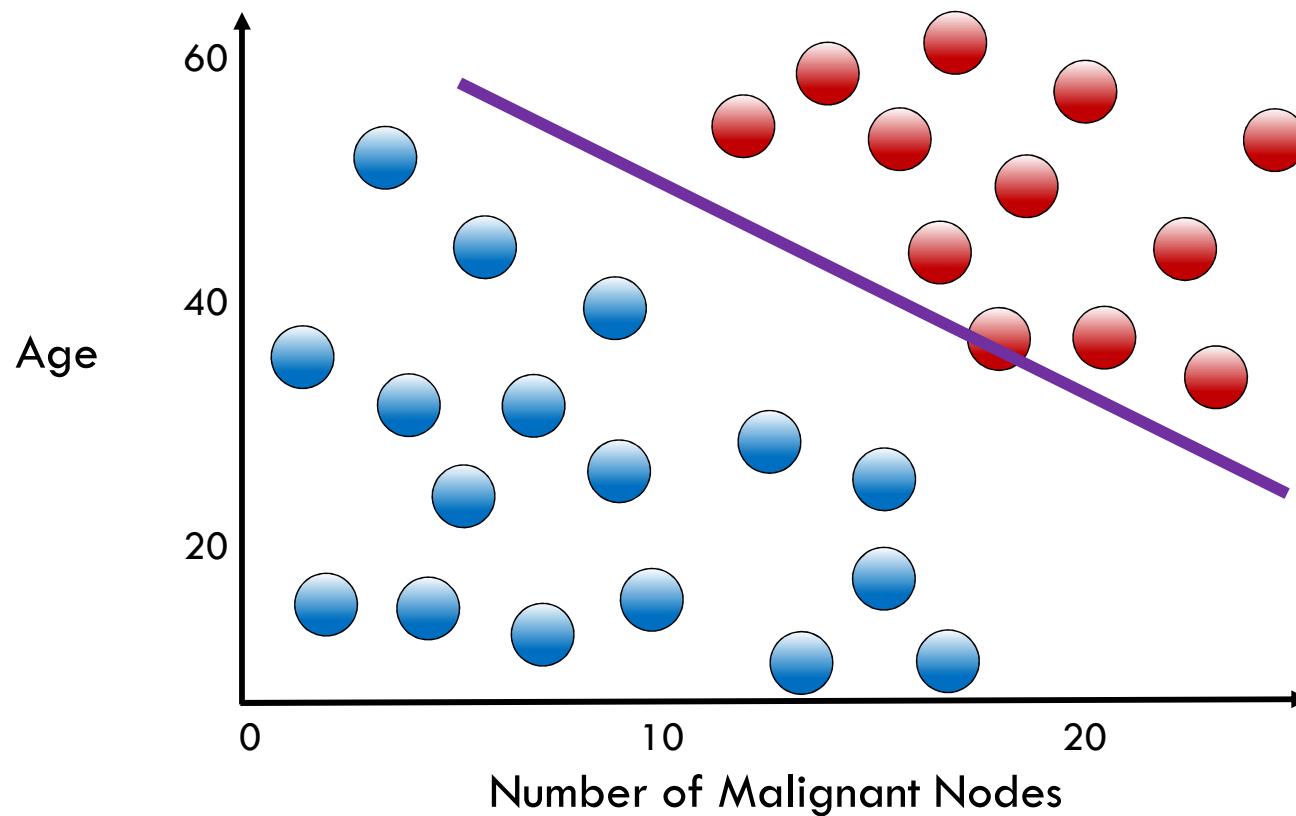
Classification with SVMs

Find the line that best separates
classes



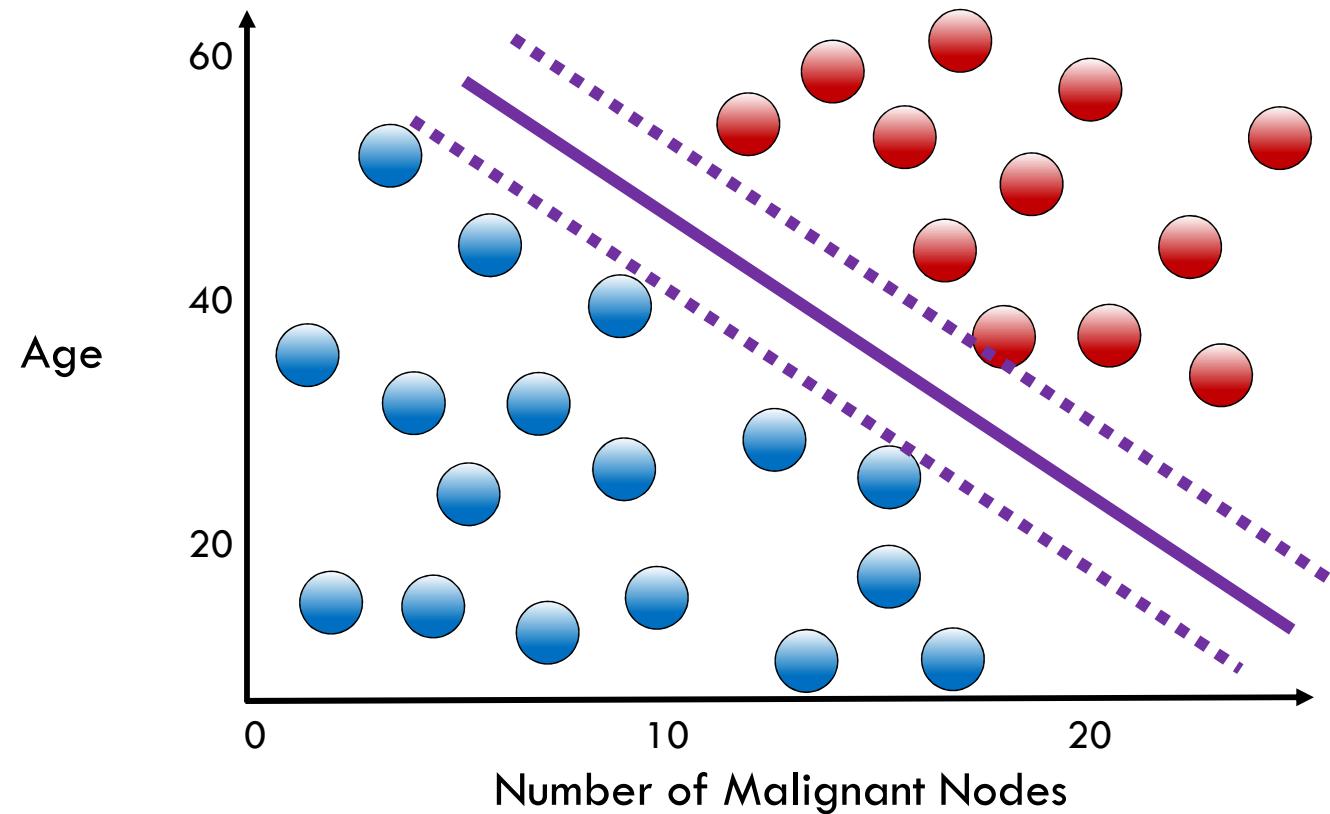
Classification with SVMs

Find the line that best separates
classes

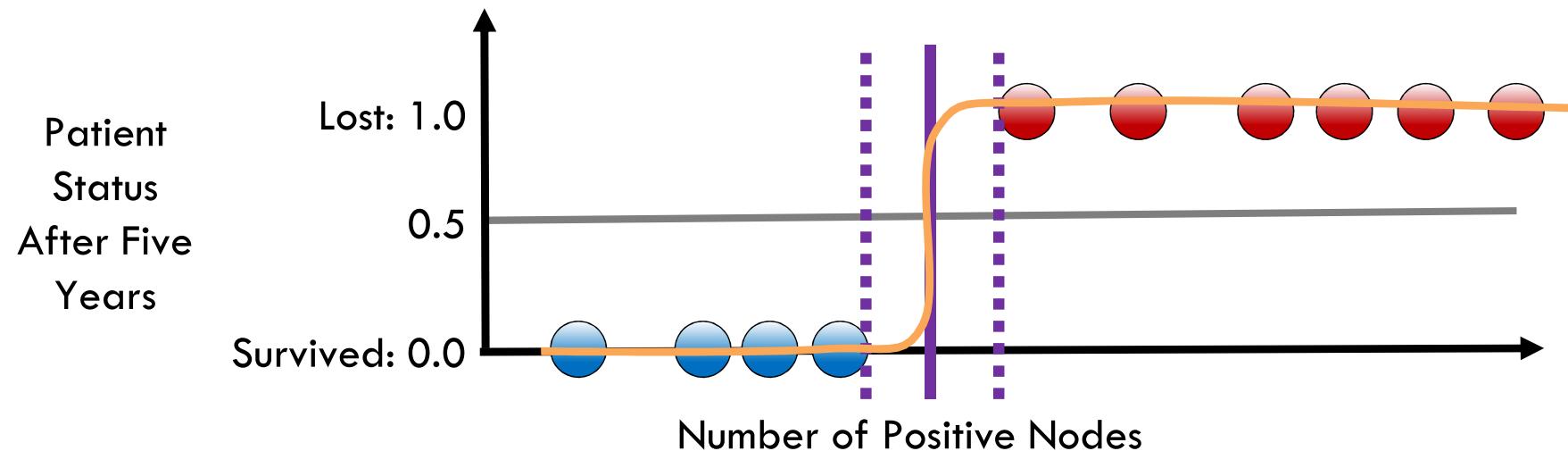


Classification with SVMs

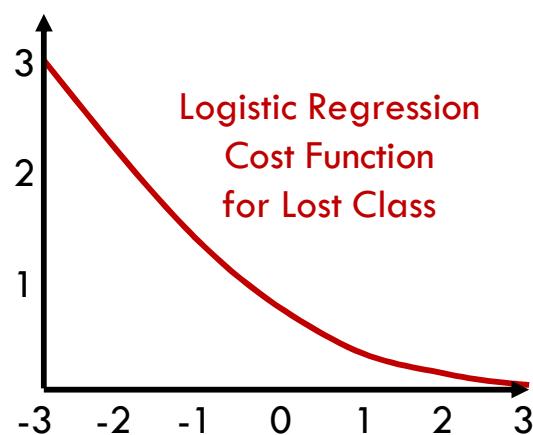
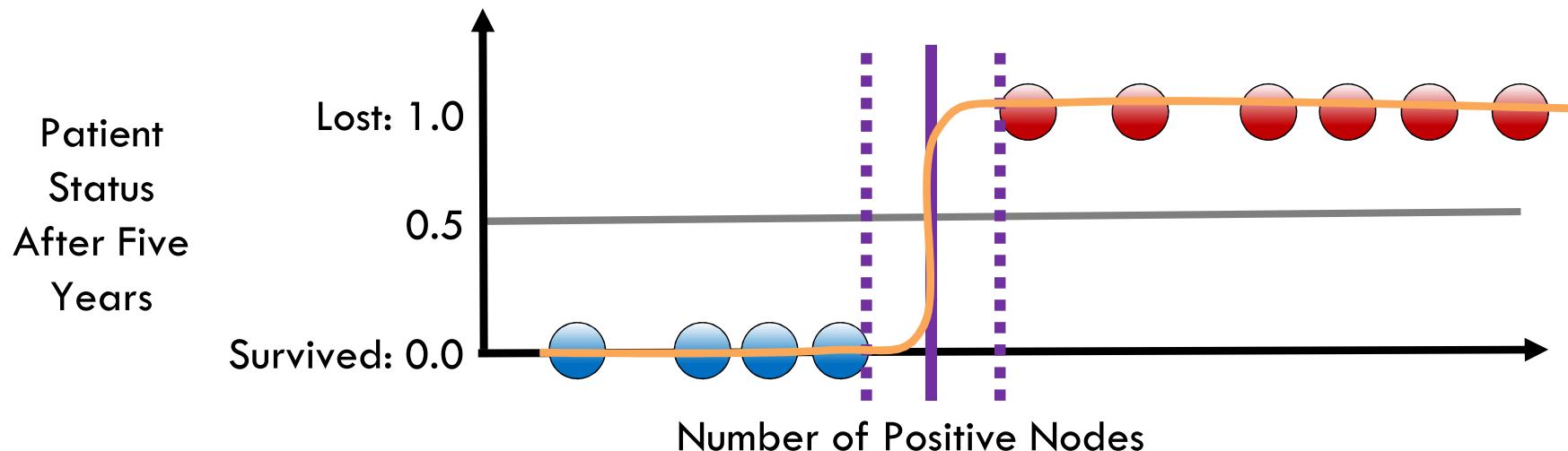
And include the largest boundary
possible



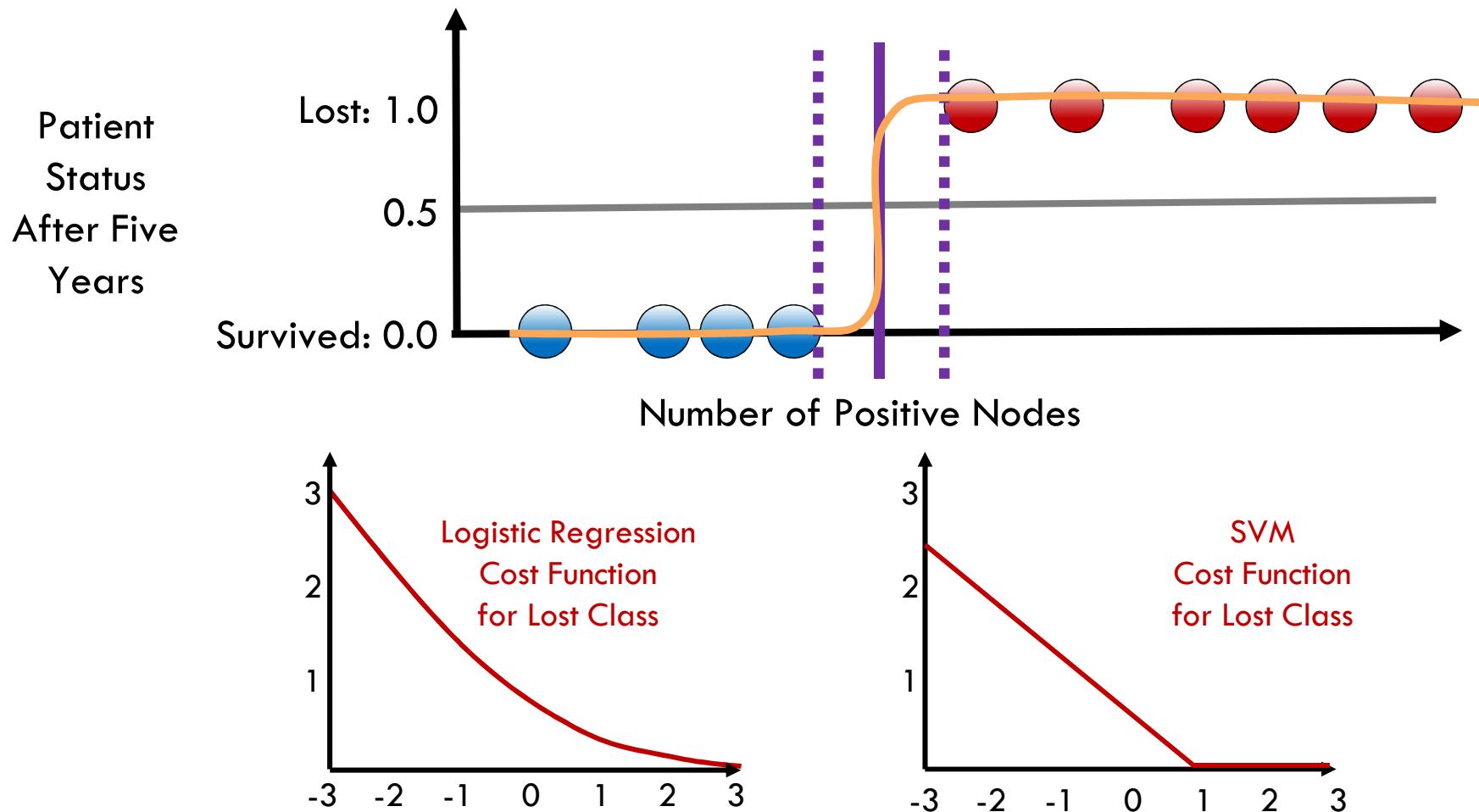
Logistic Regression vs SVM Cost Functions



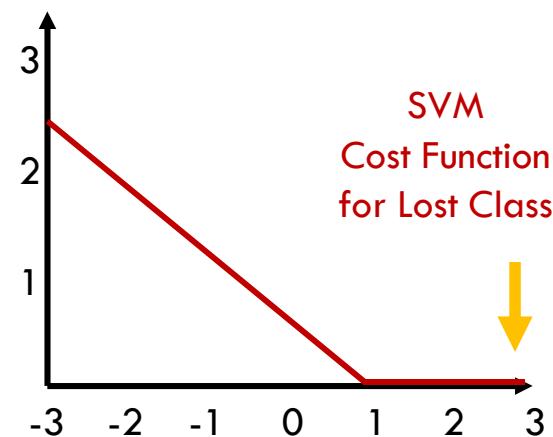
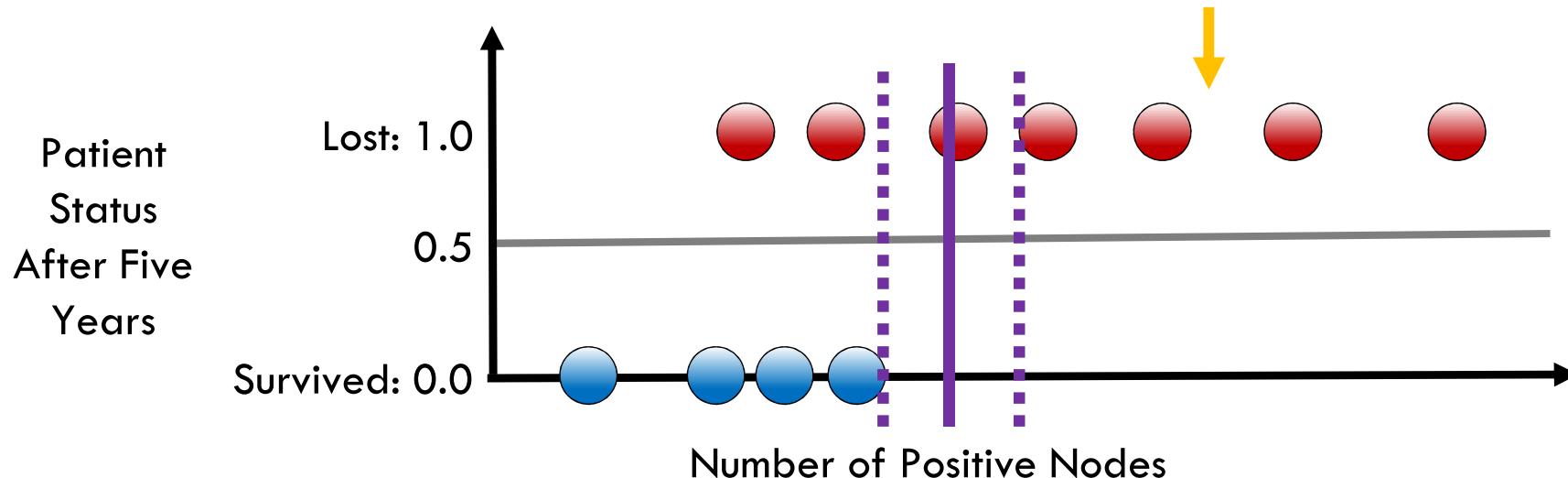
Logistic Regression vs SVM Cost Functions



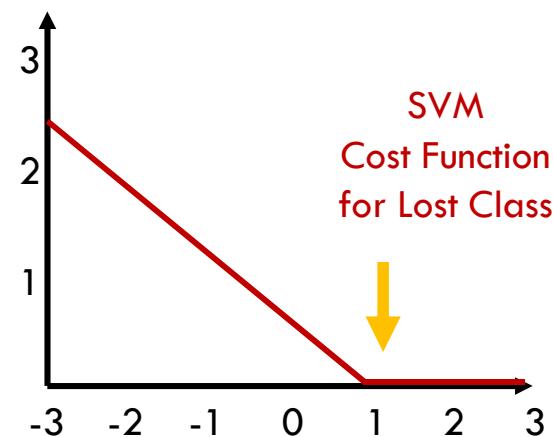
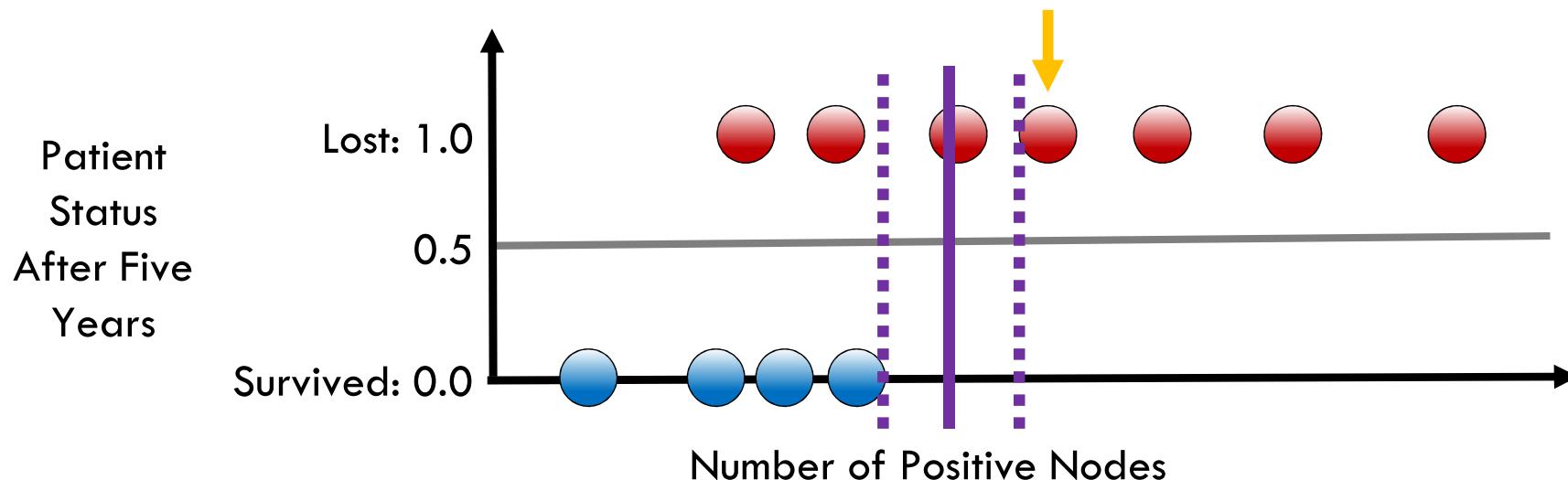
Logistic Regression vs SVM Cost Functions



The SVM Cost Function

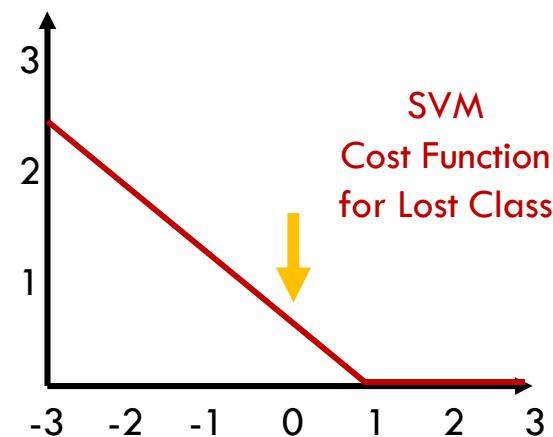
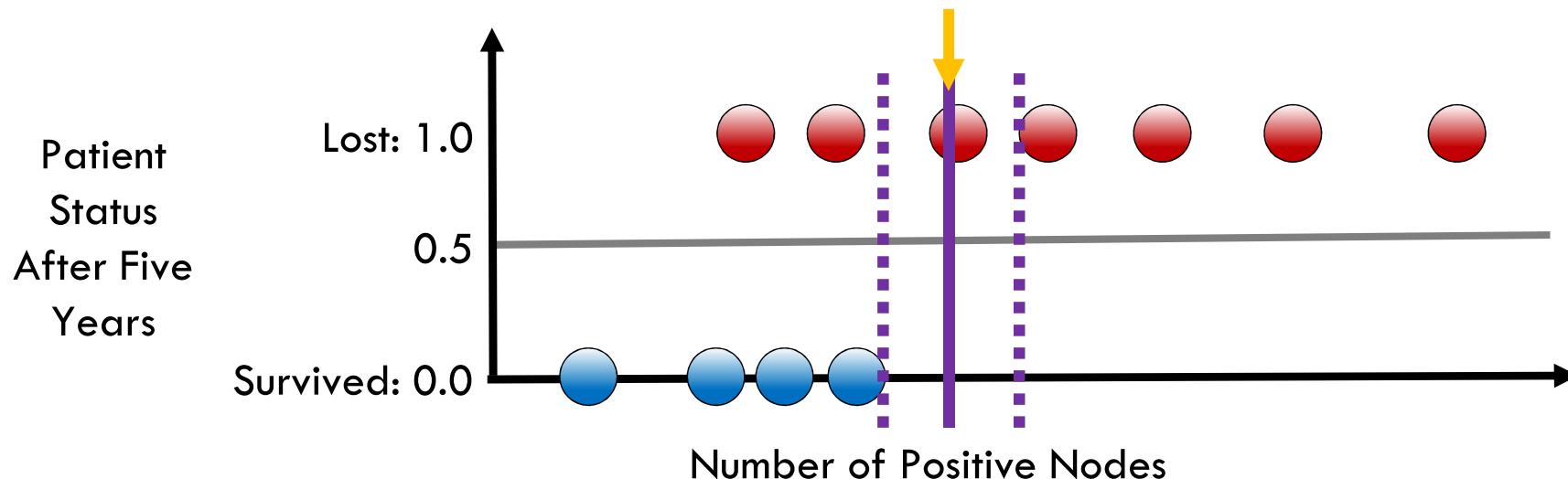


The SVM Cost Function

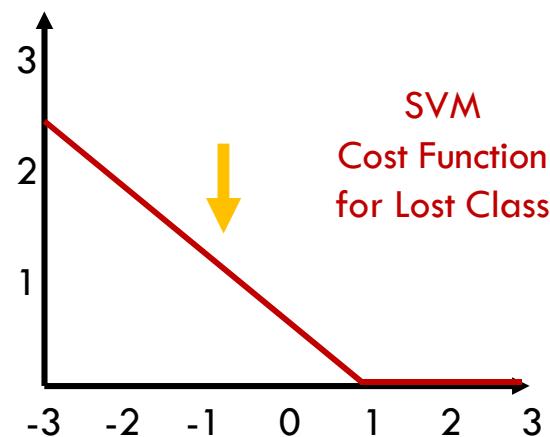
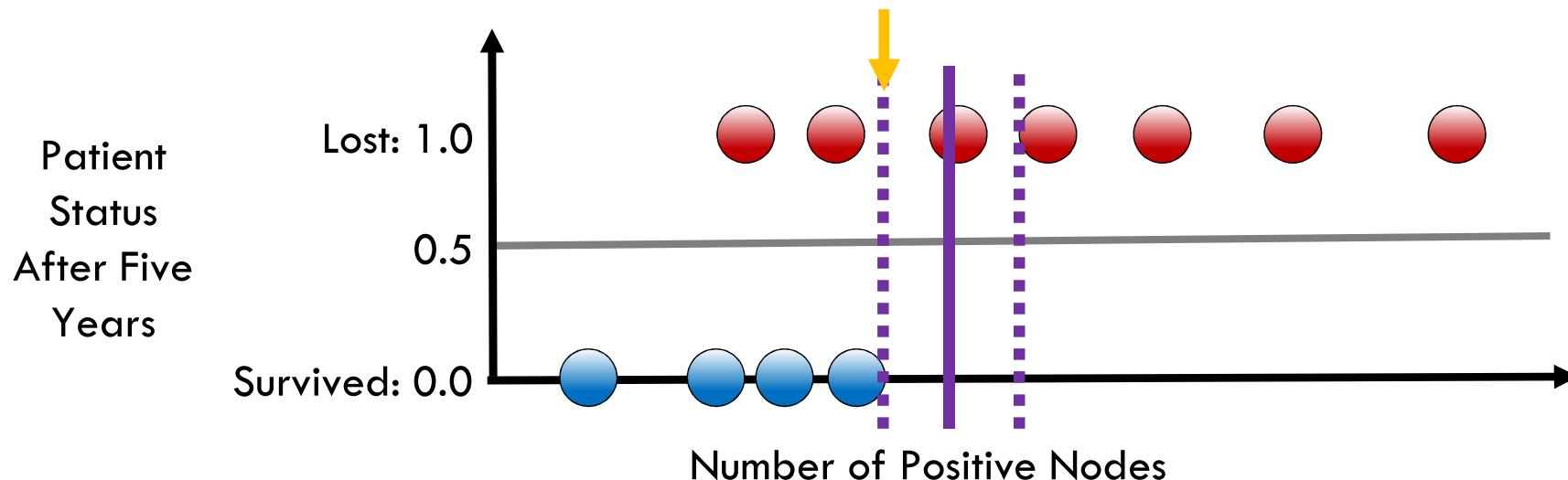


SVM
Cost Function
for Lost Class

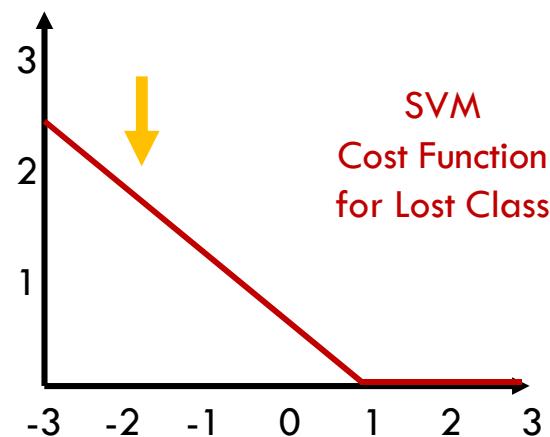
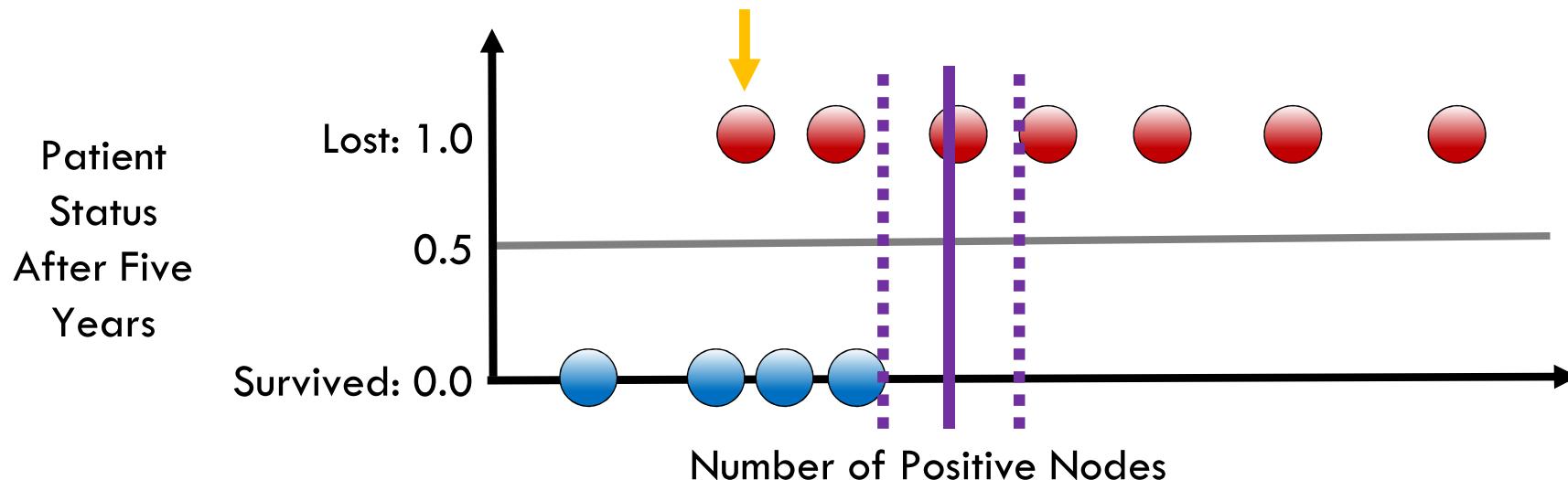
The SVM Cost Function



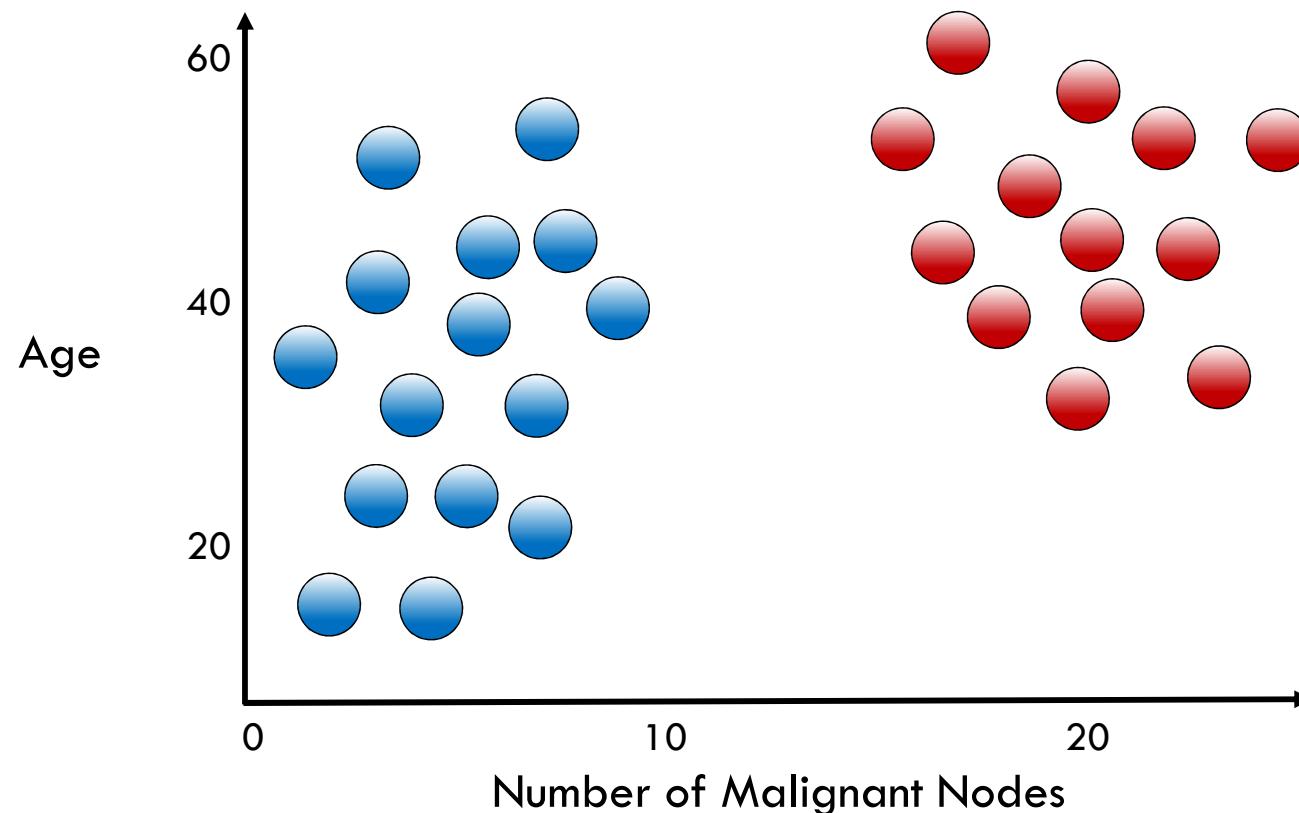
The SVM Cost Function



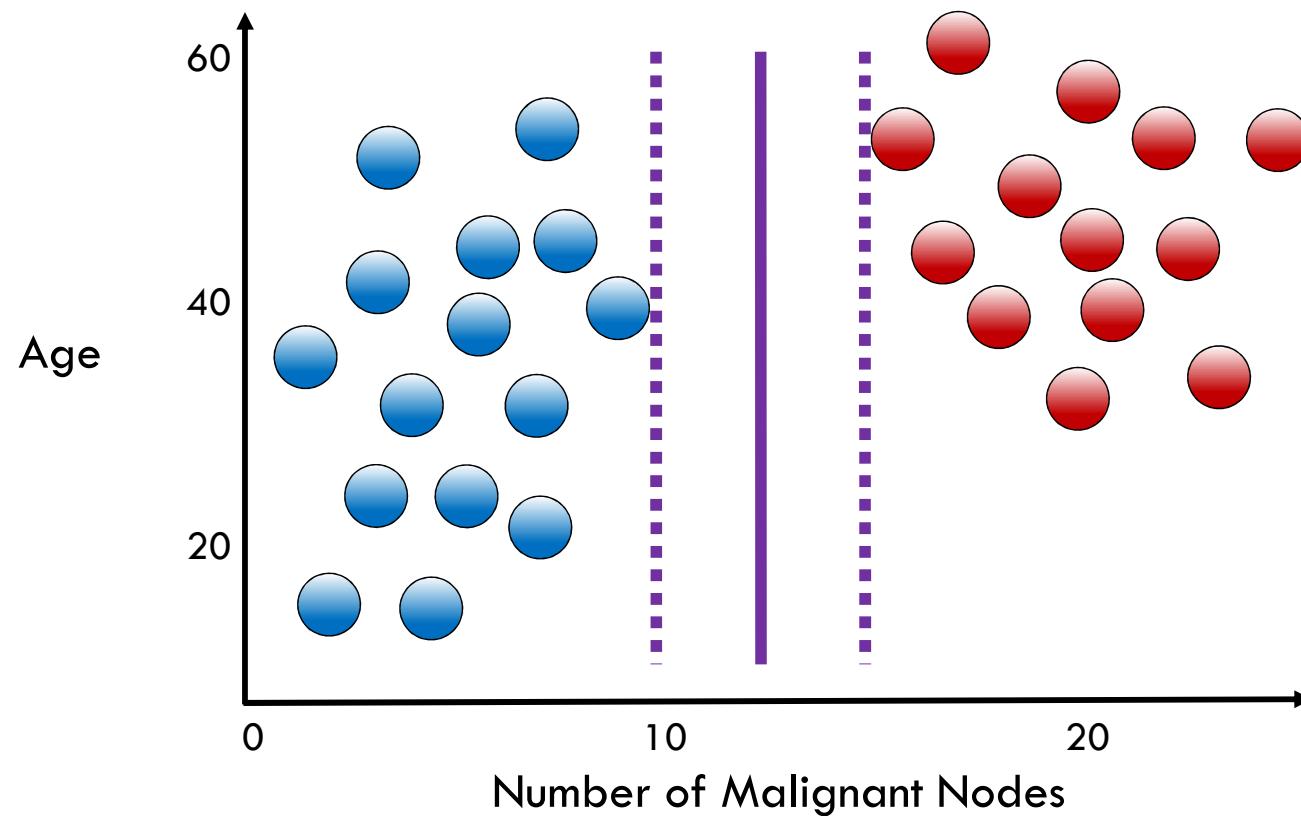
The SVM Cost Function



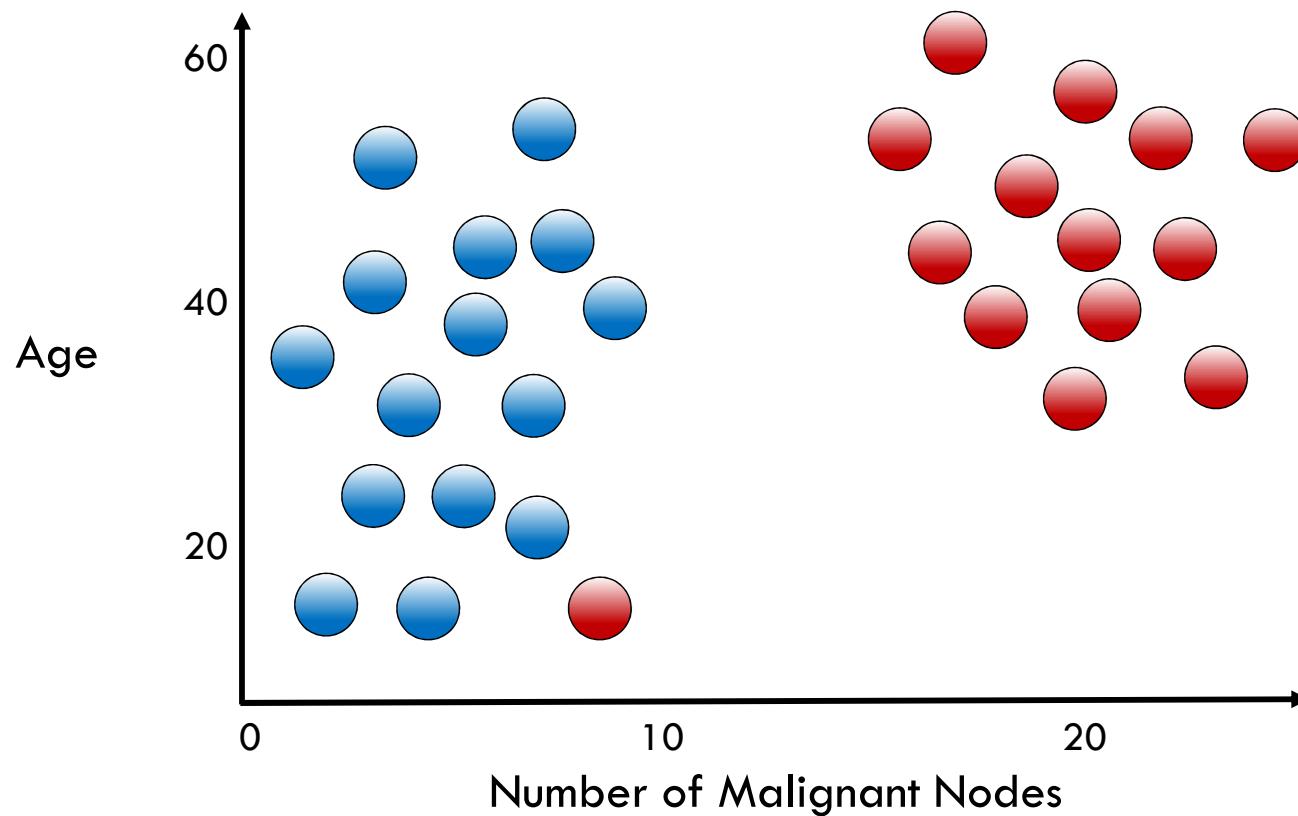
Outlier Sensitivity in SVMs



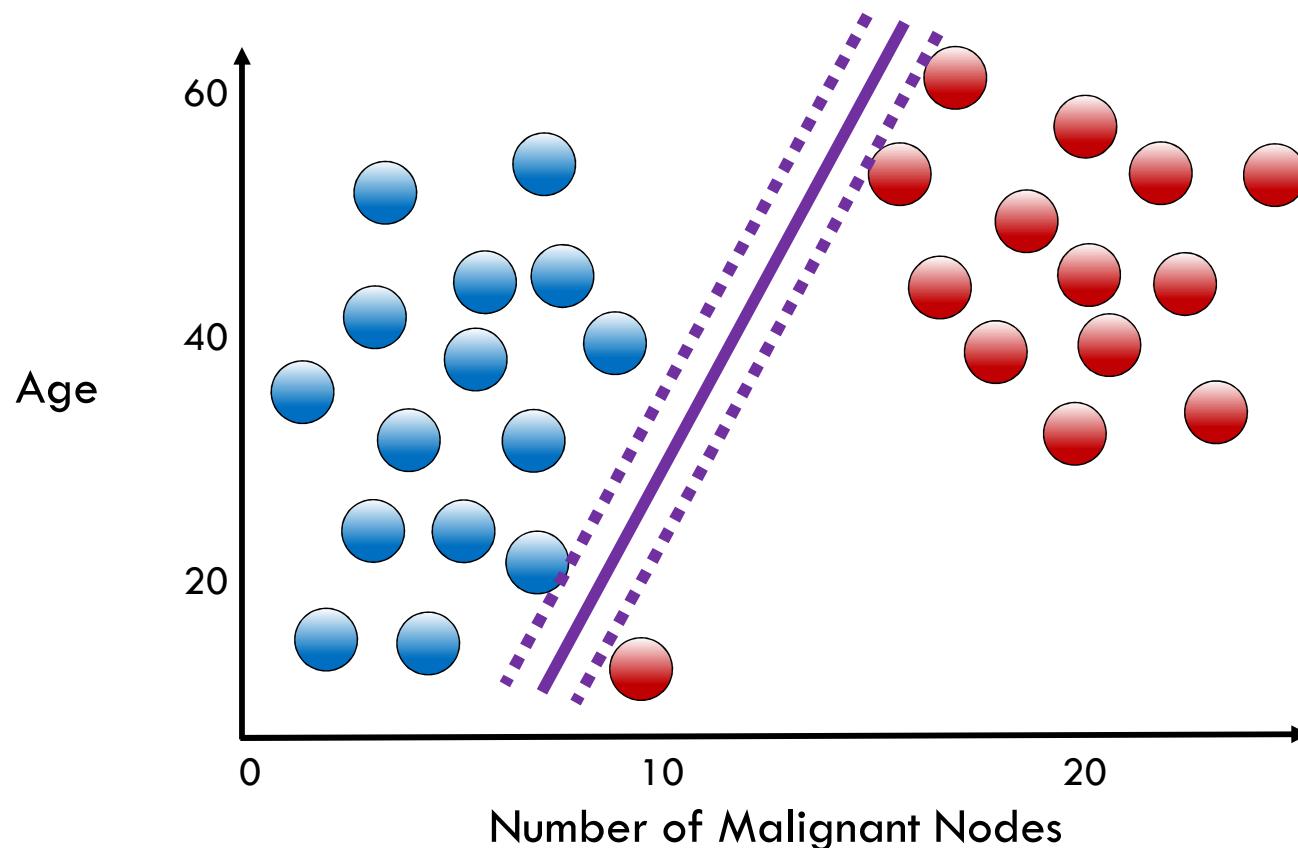
Outlier Sensitivity in SVMs



Outlier Sensitivity in SVMs

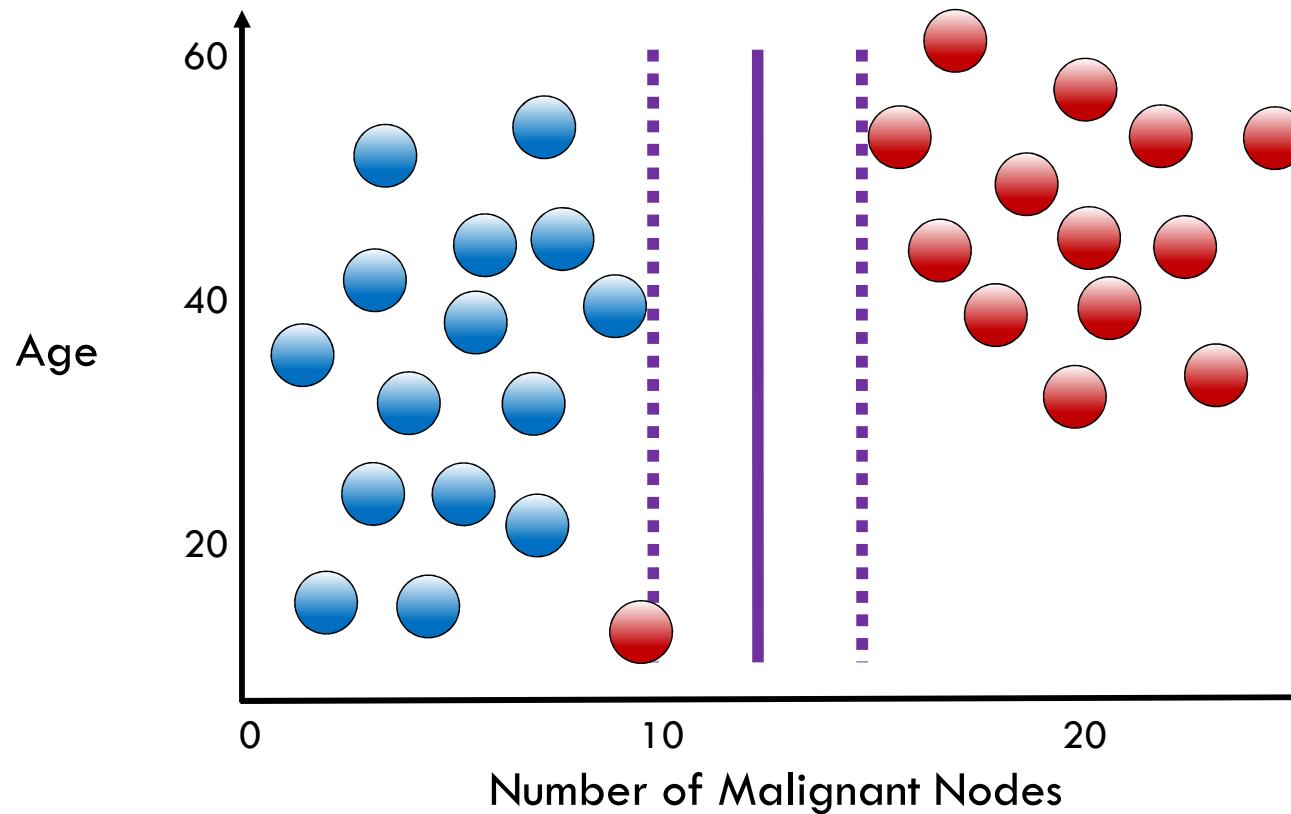


Outlier Sensitivity in SVMs



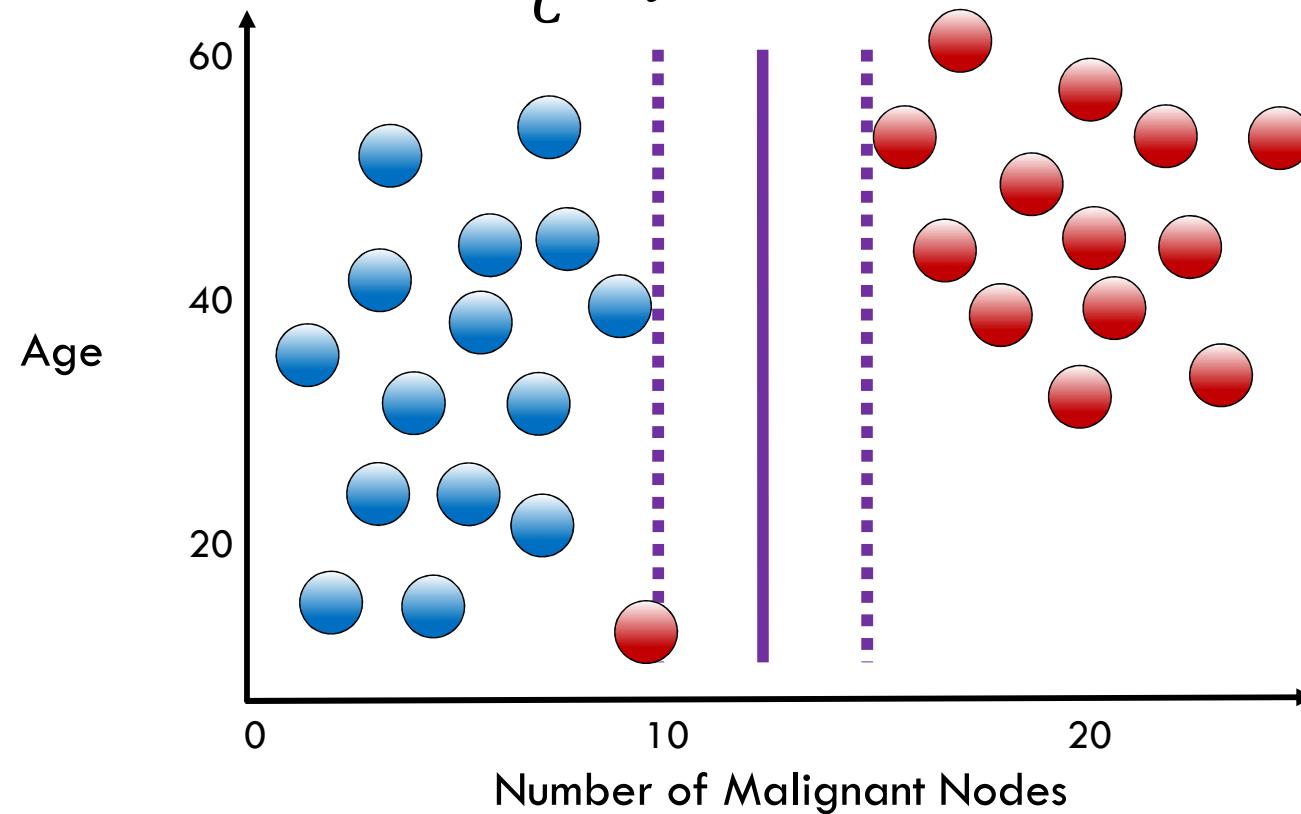
Outlier Sensitivity in SVMs

This is probably still the correct boundary



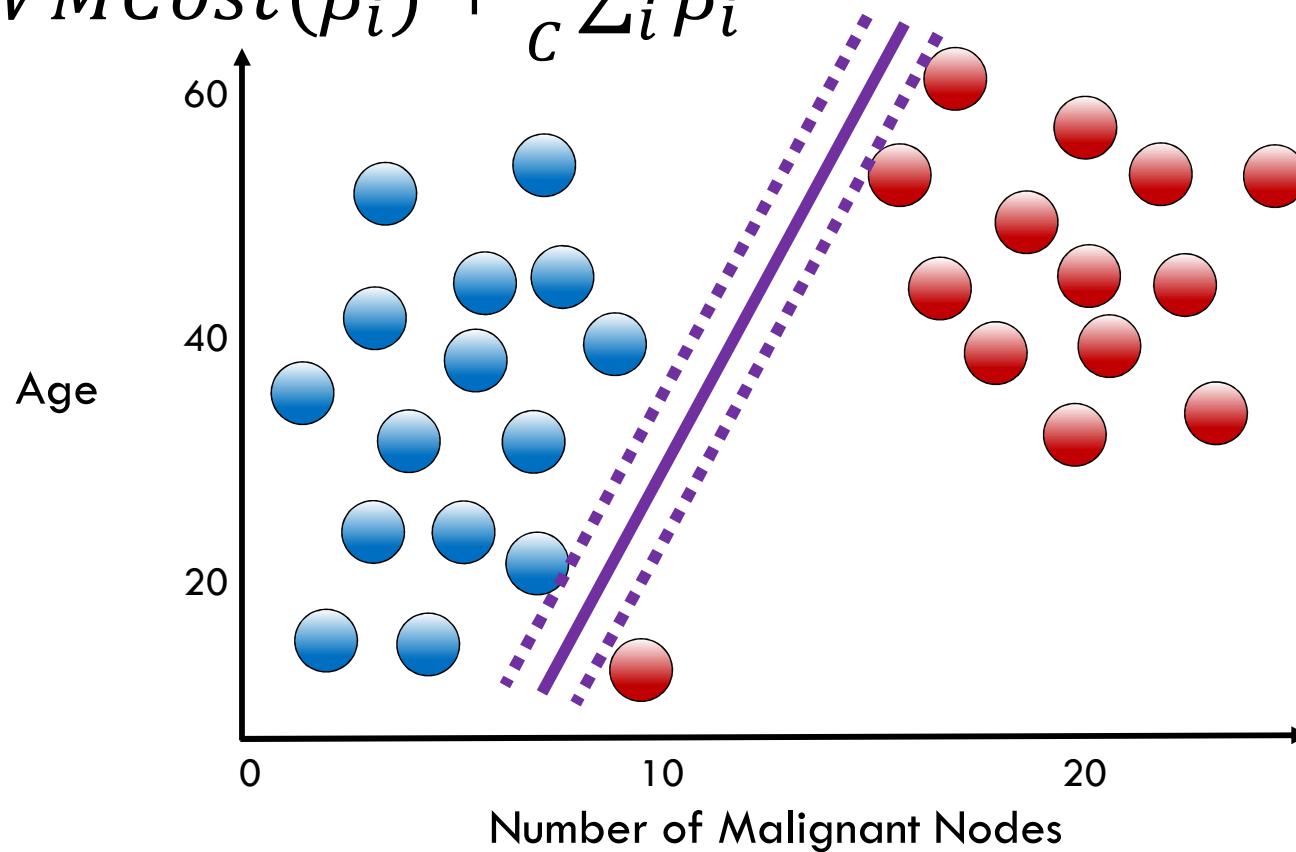
Regularization in SVMs

$$J(\beta_i) = SVMCost(\beta_i) + \frac{1}{C} \sum_i \beta_i$$



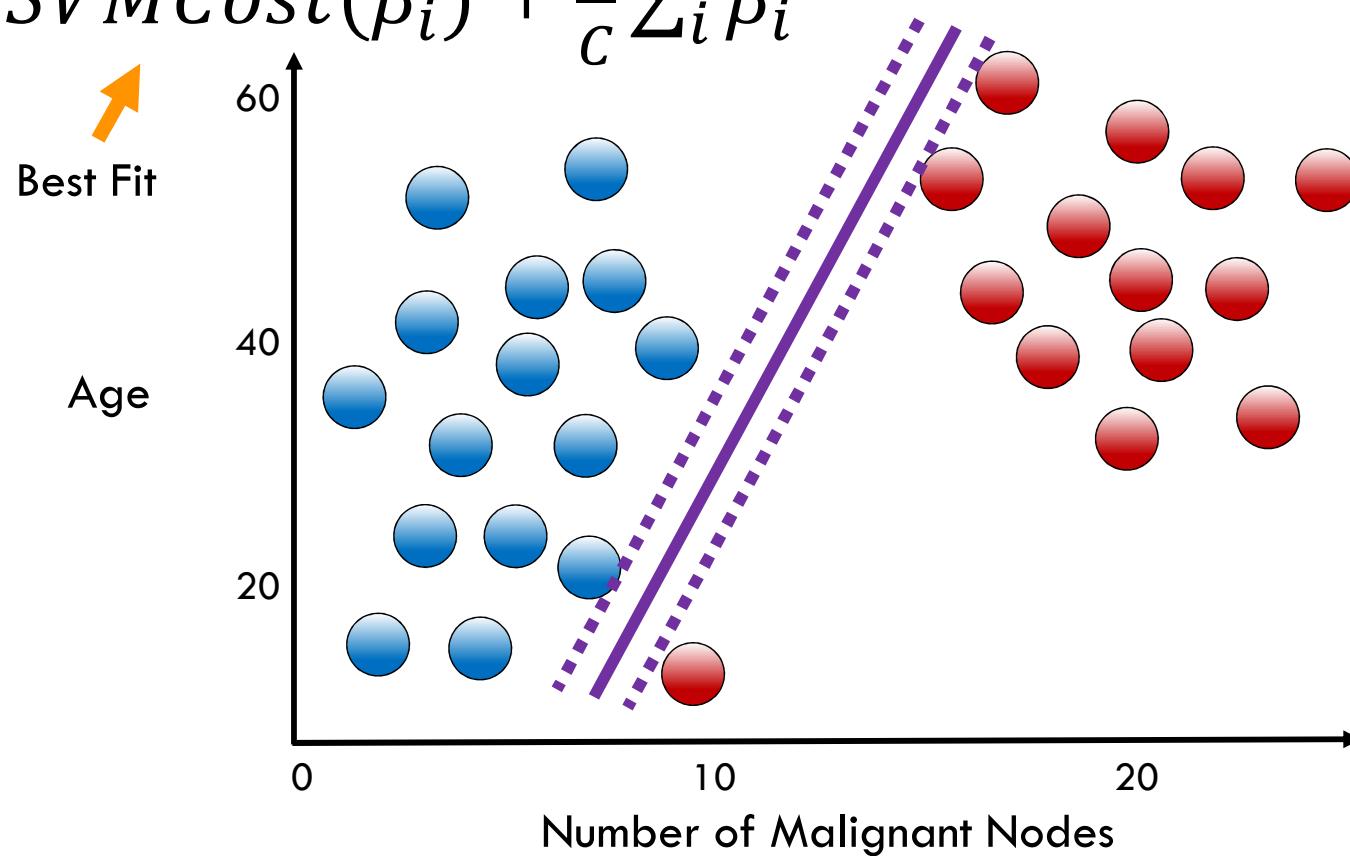
Regularization in SVMs

$$J(\beta_i) = SVMCost(\beta_i) + \frac{1}{C} \sum_i \beta_i$$



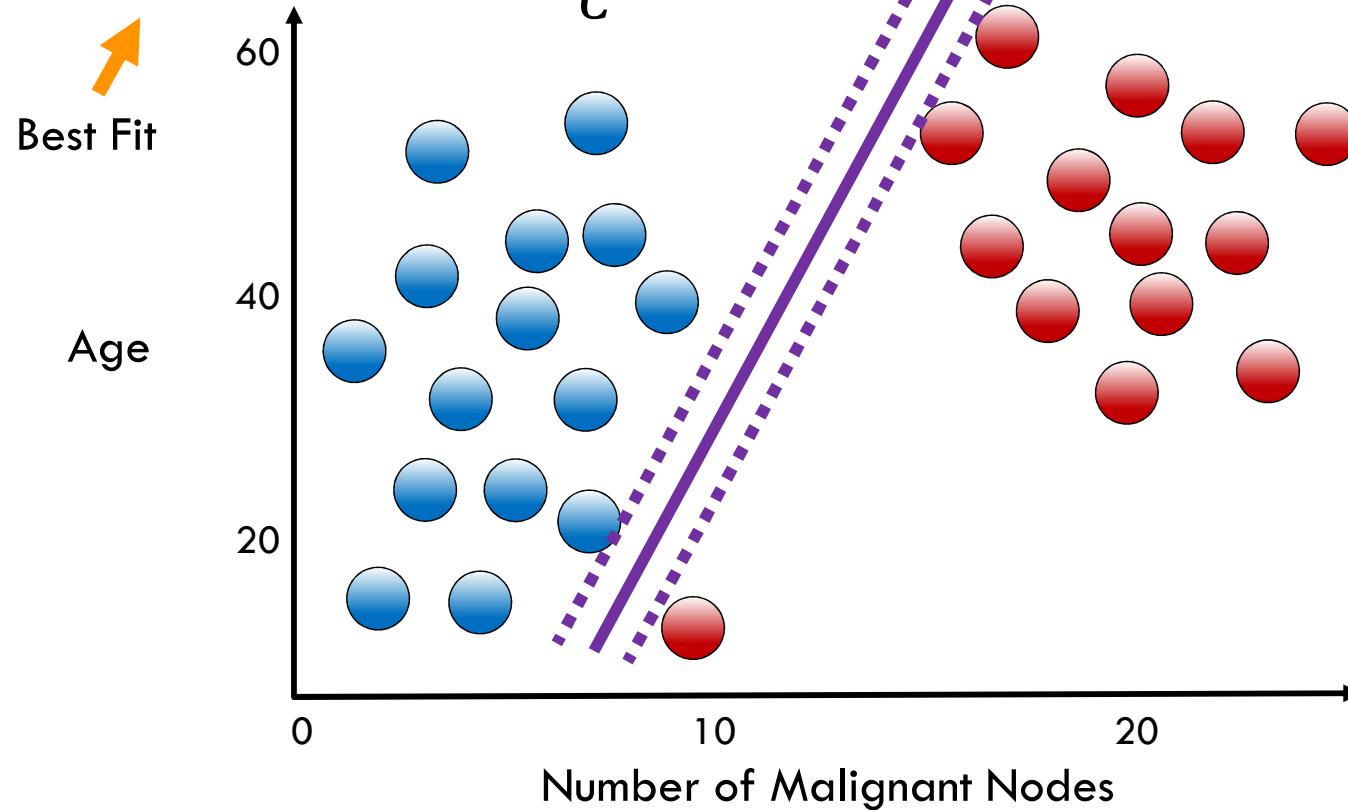
Regularization in SVMs

$$J(\beta_i) = SVMCost(\beta_i) + \frac{1}{C} \sum_i \beta_i$$



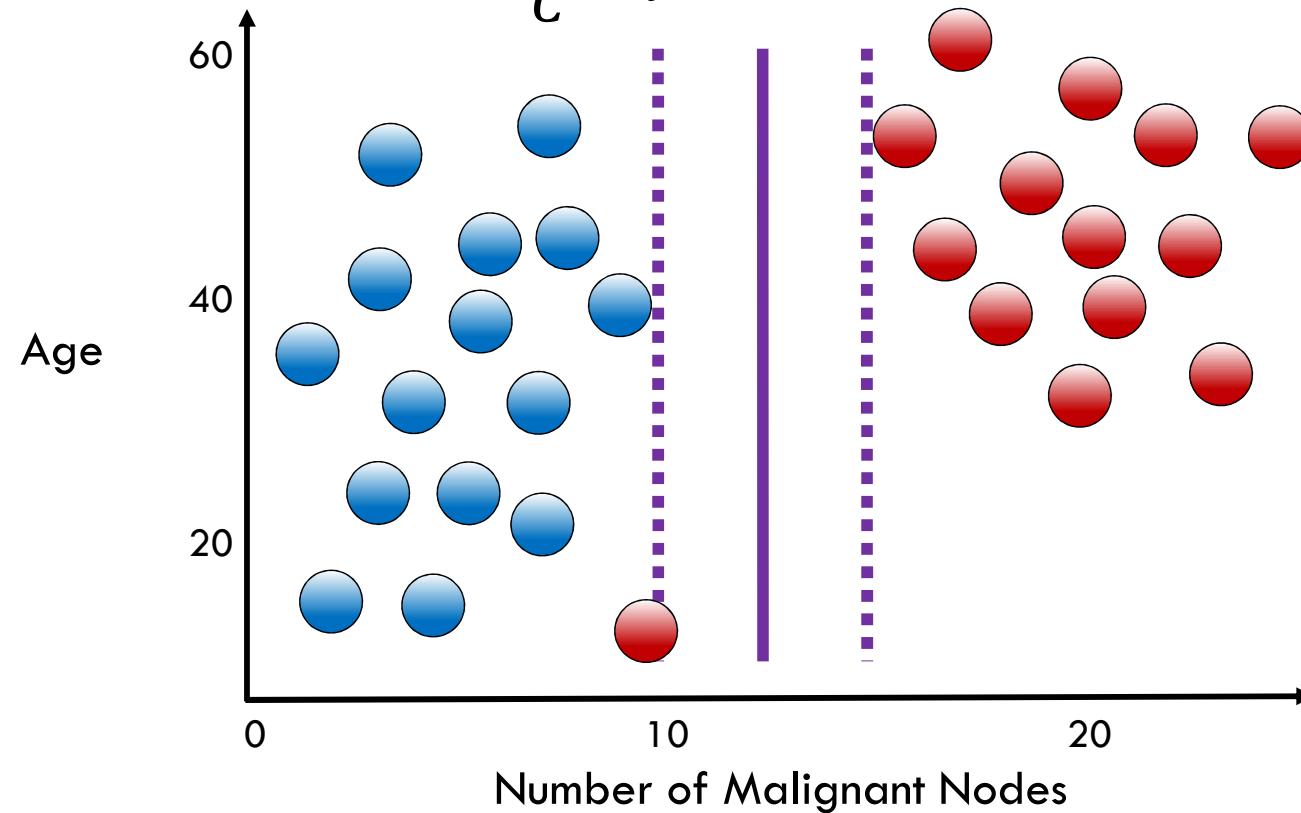
Regularization in SVMs

$$J(\beta_i) = SVMCost(\beta_i) + \frac{1}{C} \sum_i \beta_i$$



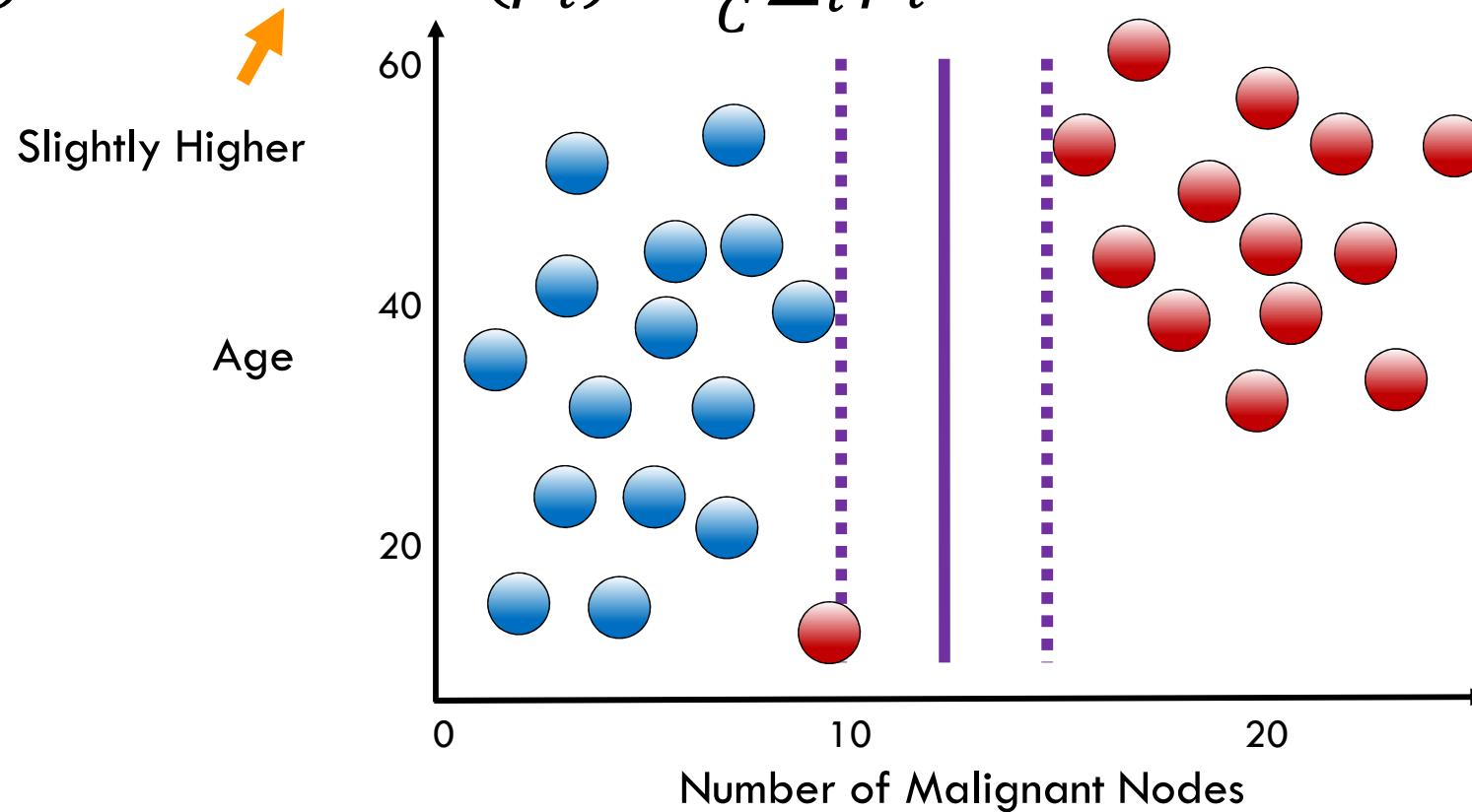
Regularization in SVMs

$$J(\beta_i) = SVMCost(\beta_i) + \frac{1}{C} \sum_i \beta_i$$



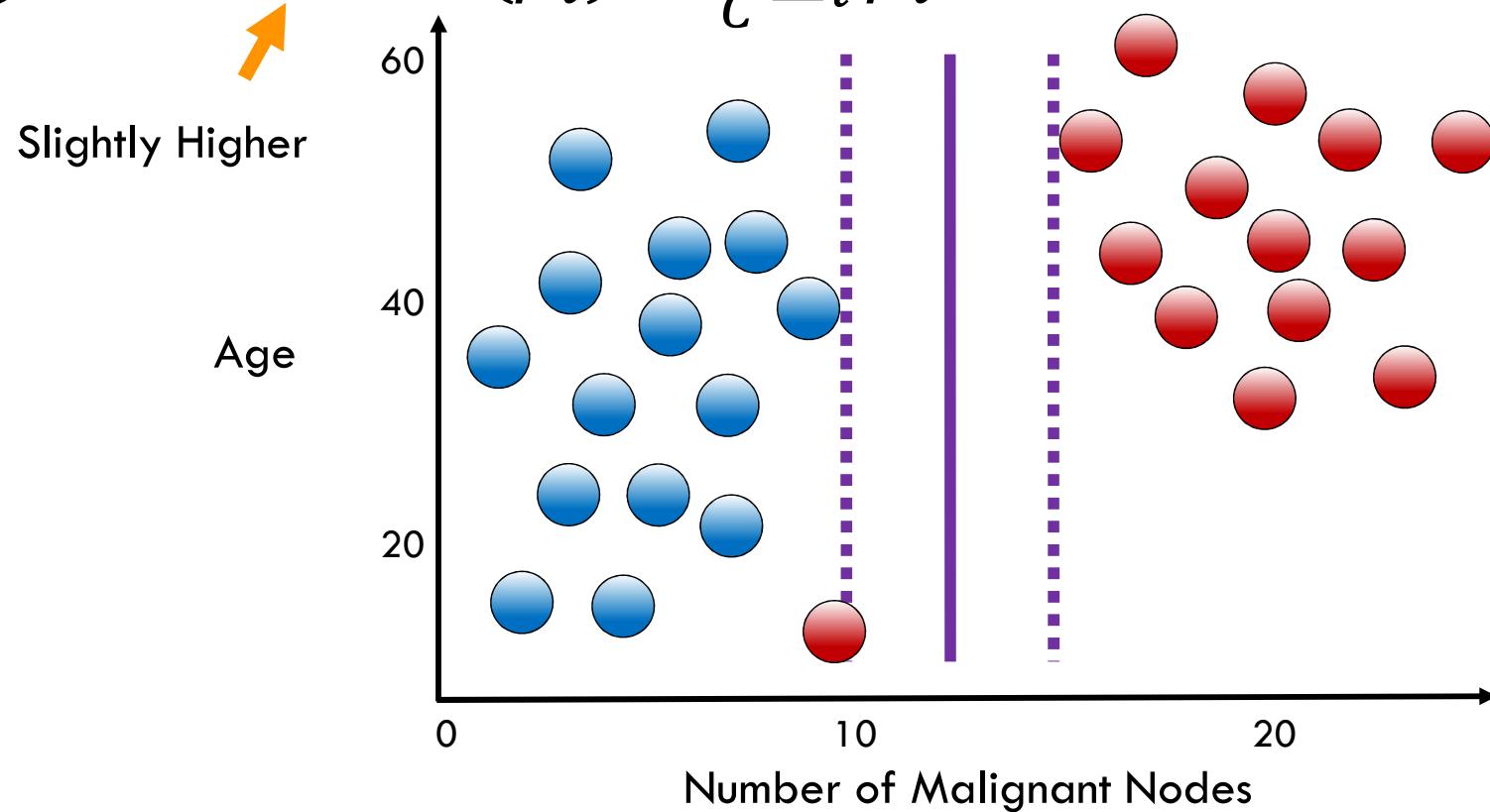
Regularization in SVMs

$$J(\beta_i) = SVMCost(\beta_i) + \frac{1}{C} \sum_i \beta_i$$



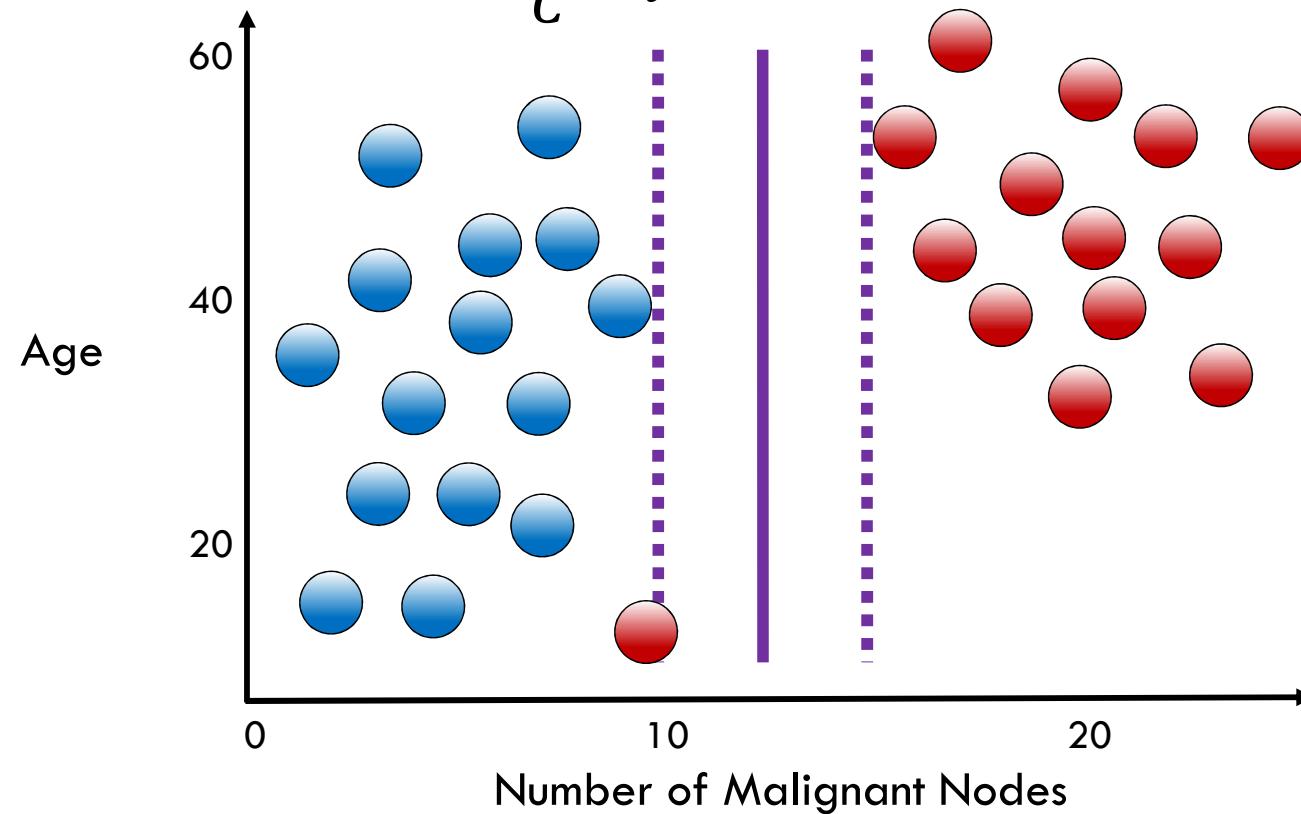
Regularization in SVMs

$$J(\beta_i) = SVMCost(\beta_i) + \frac{1}{C} \sum_i \beta_i$$



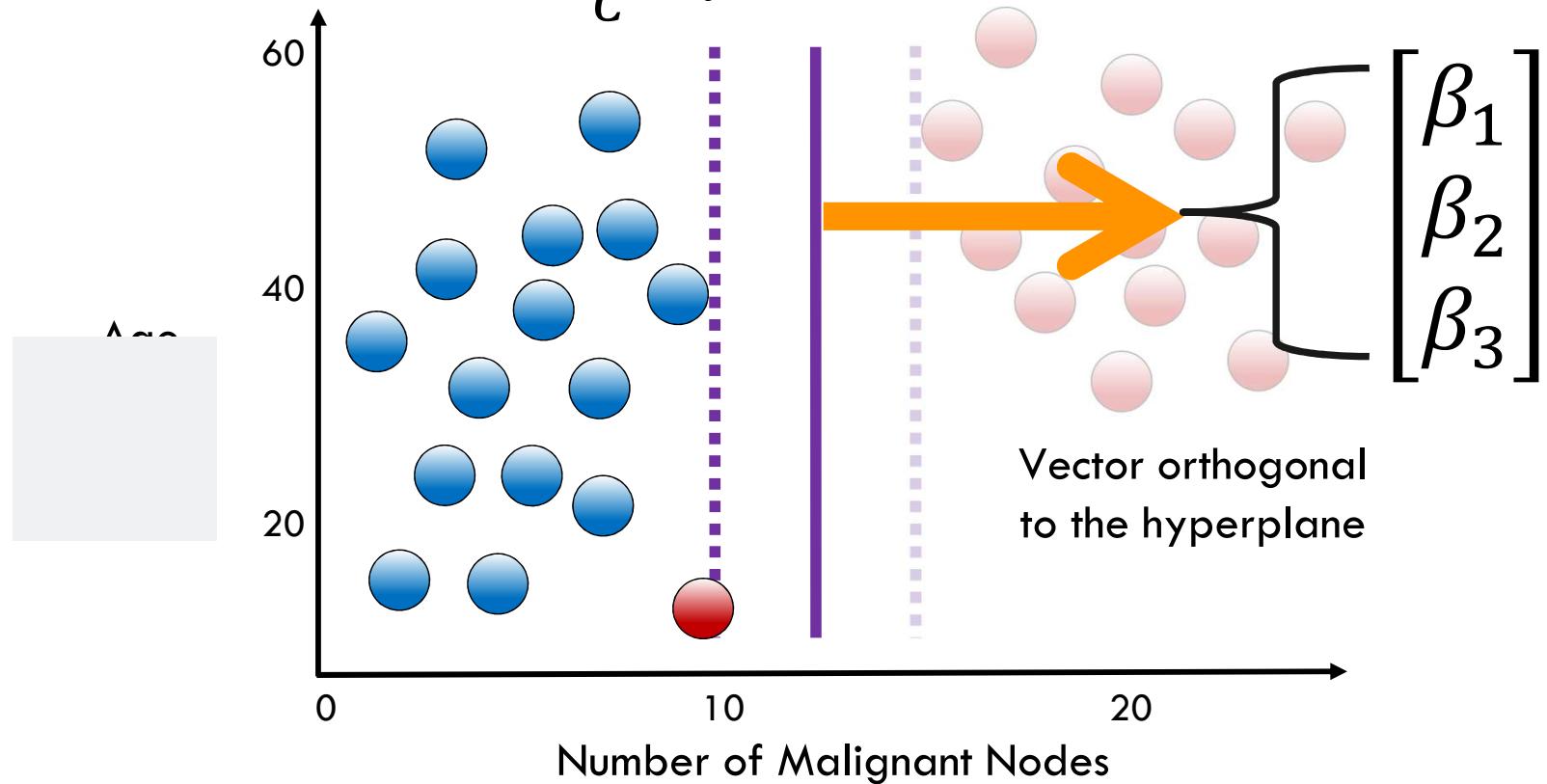
Interpretation of SVM Coefficients

$$J(\beta_i) = SVMCost(\beta_i) + \frac{1}{C} \sum_i \beta_i$$



Interpretation of SVM Coefficients

$$J(\beta_i) = SVMCost(\beta_i) + \frac{1}{C} \sum_i \beta_i$$



Linear SVM: The Syntax

Import the class containing the classification method

```
from sklearn.svm import LinearSVC
```

Linear SVM: The Syntax

Import the class containing the classification method

```
from sklearn.svm import LinearSVC
```

Create an instance of the class

```
LinSVC = LinearSVC(penalty='l2', C=10.0)
```

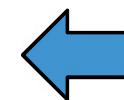
Linear SVM: The Syntax

Import the class containing the classification method

```
from sklearn.svm import LinearSVC
```

Create an instance of the class

```
LinSVC = LinearSVC(penalty='l2', C=10.0)
```



regularization
parameters

Linear SVM: The Syntax

Import the class containing the classification method

```
from sklearn.svm import LinearSVC
```

Create an instance of the class

```
LinSVC = LinearSVC(penalty='l2', C=10.0)
```

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

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

```
y_predict = LinSVC.predict(X_test)
```

Linear SVM: The Syntax

Import the class containing the classification method

```
from sklearn.svm import LinearSVC
```

Create an instance of the class

```
LinSVC = LinearSVC(penalty='l2', C=10.0)
```

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

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

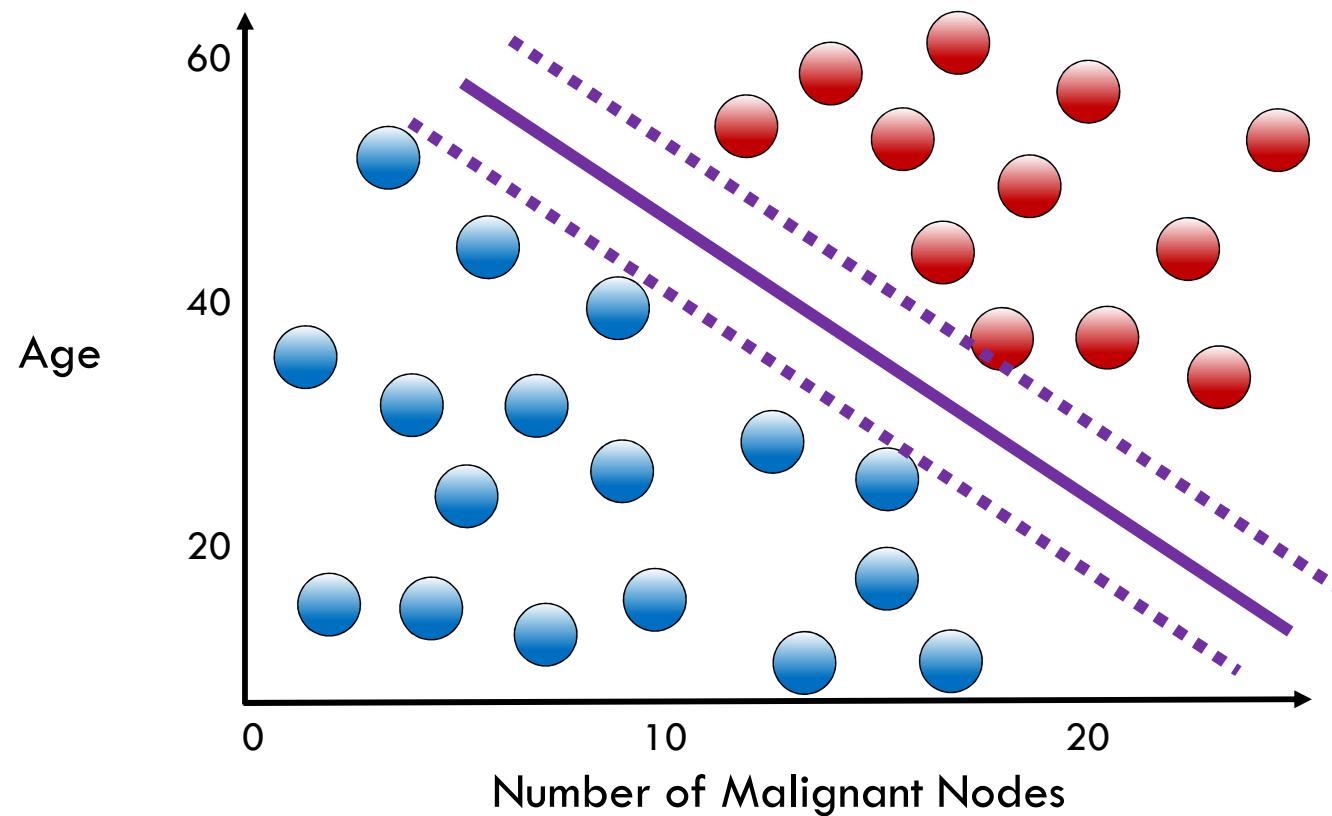
```
y_predict = LinSVC.predict(X_test)
```

Tune regularization parameters with cross-validation.



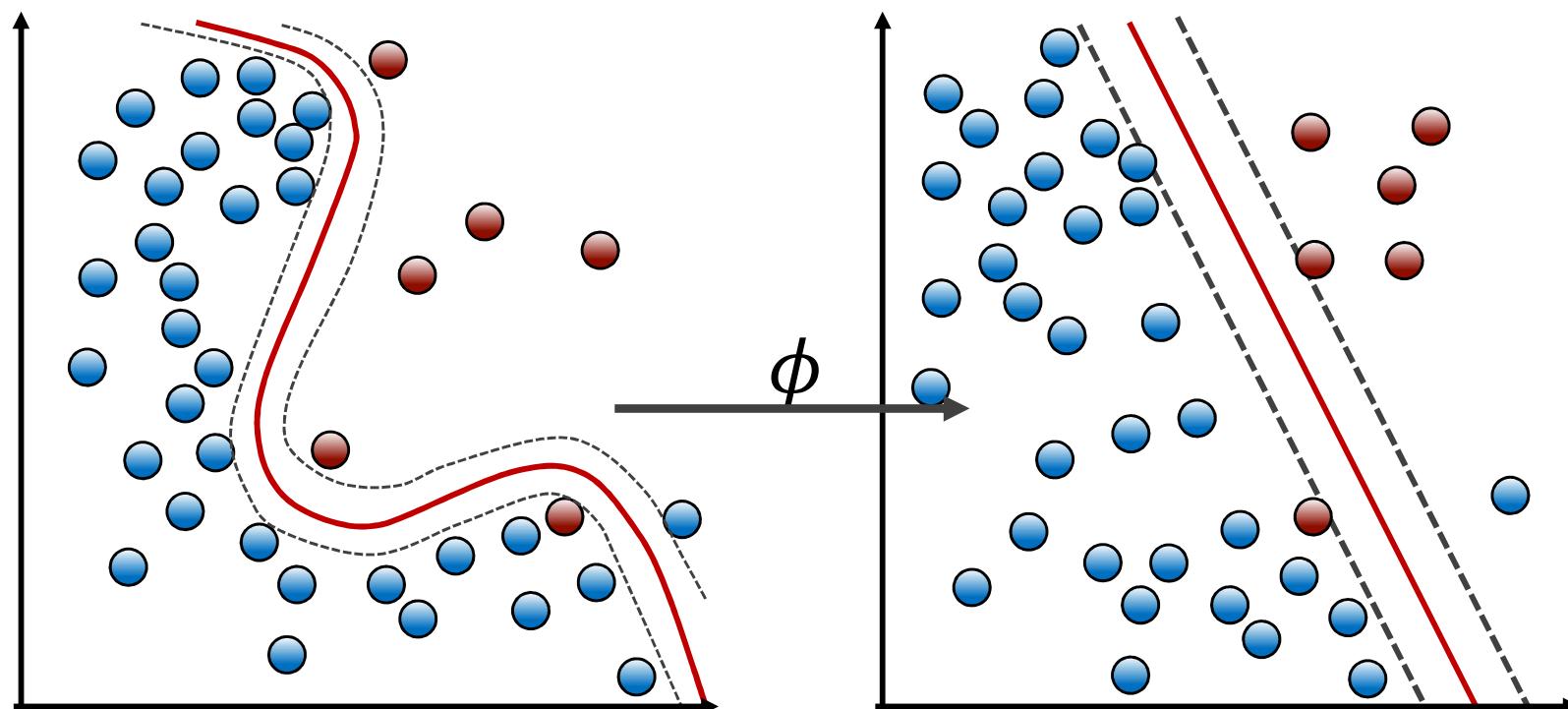
2.12. SVM com Kernel

Classification with SVMs



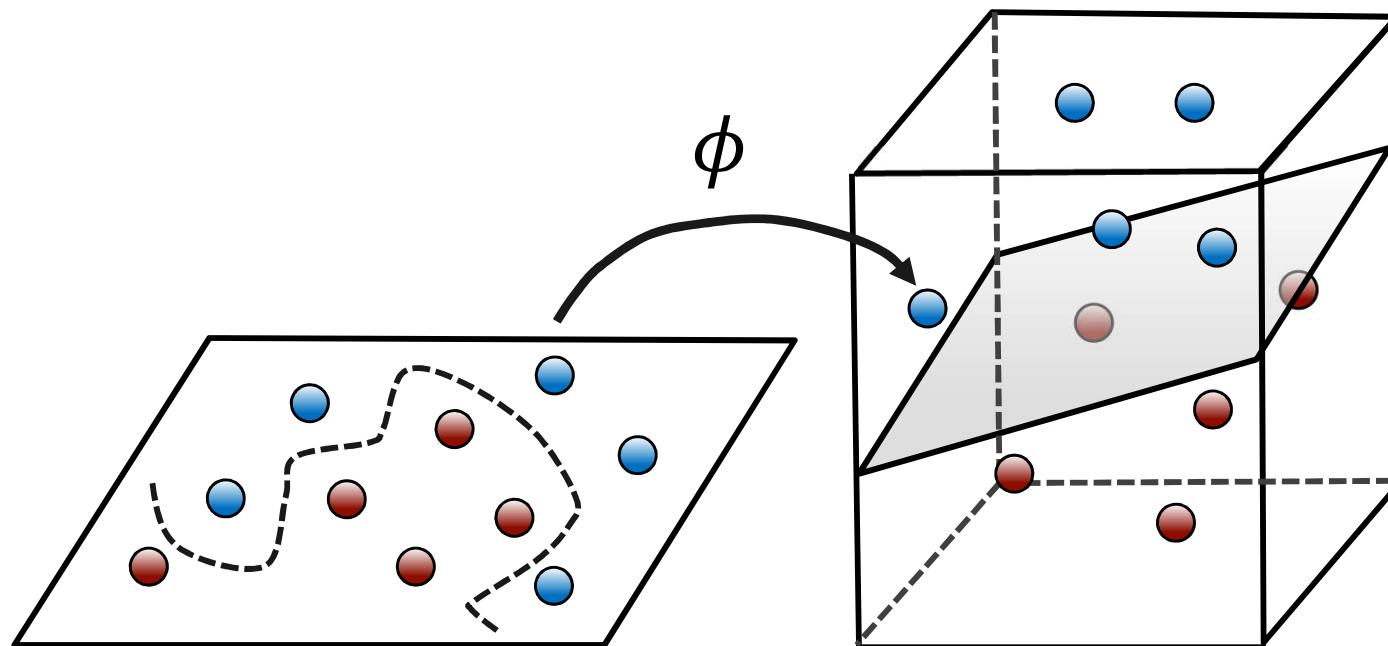
Non-Linear Decision Boundaries with SVM

Non-linear data can be made linear
with higher dimensionality



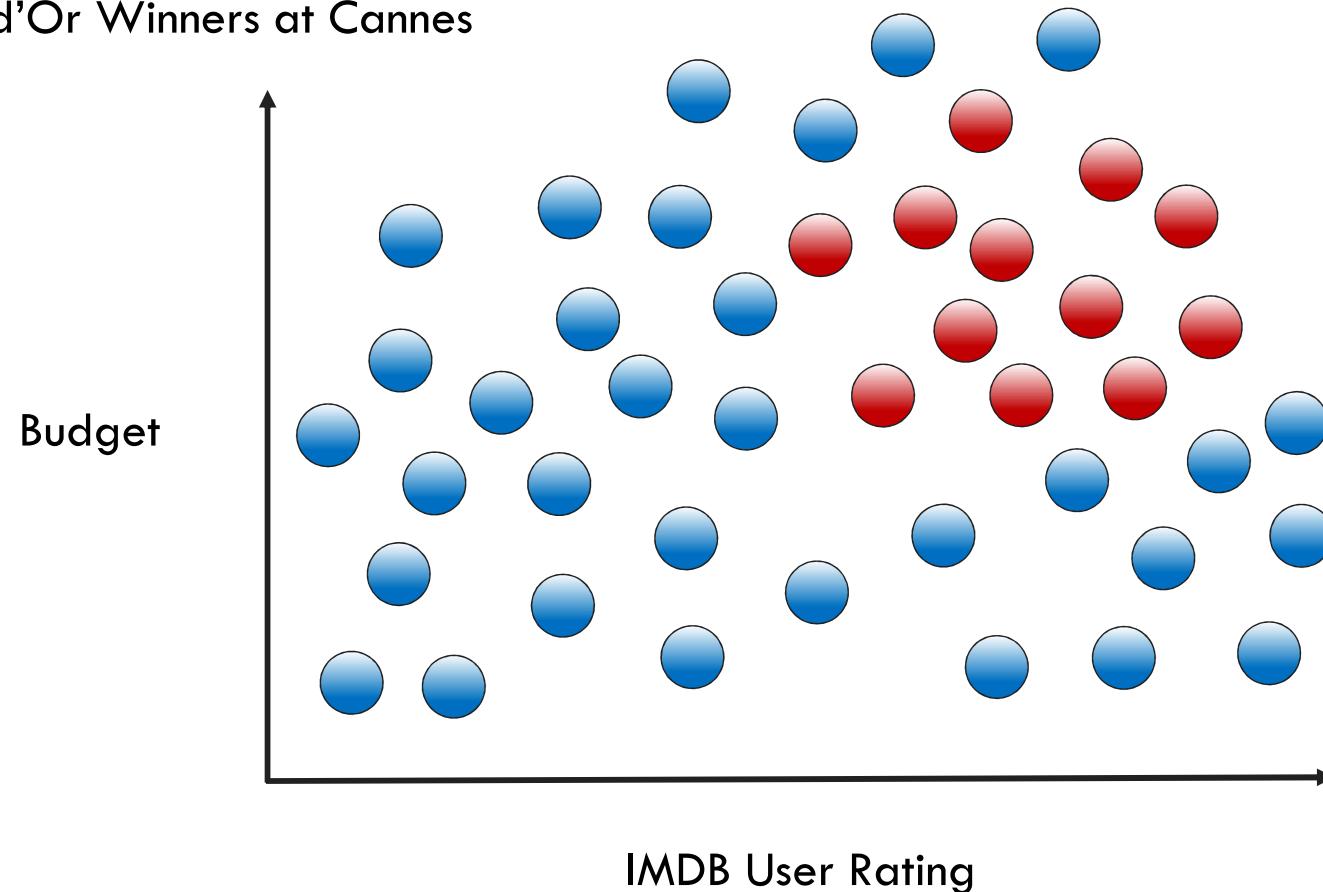
The Kernel Trick

Transform data so it is
linearly separable



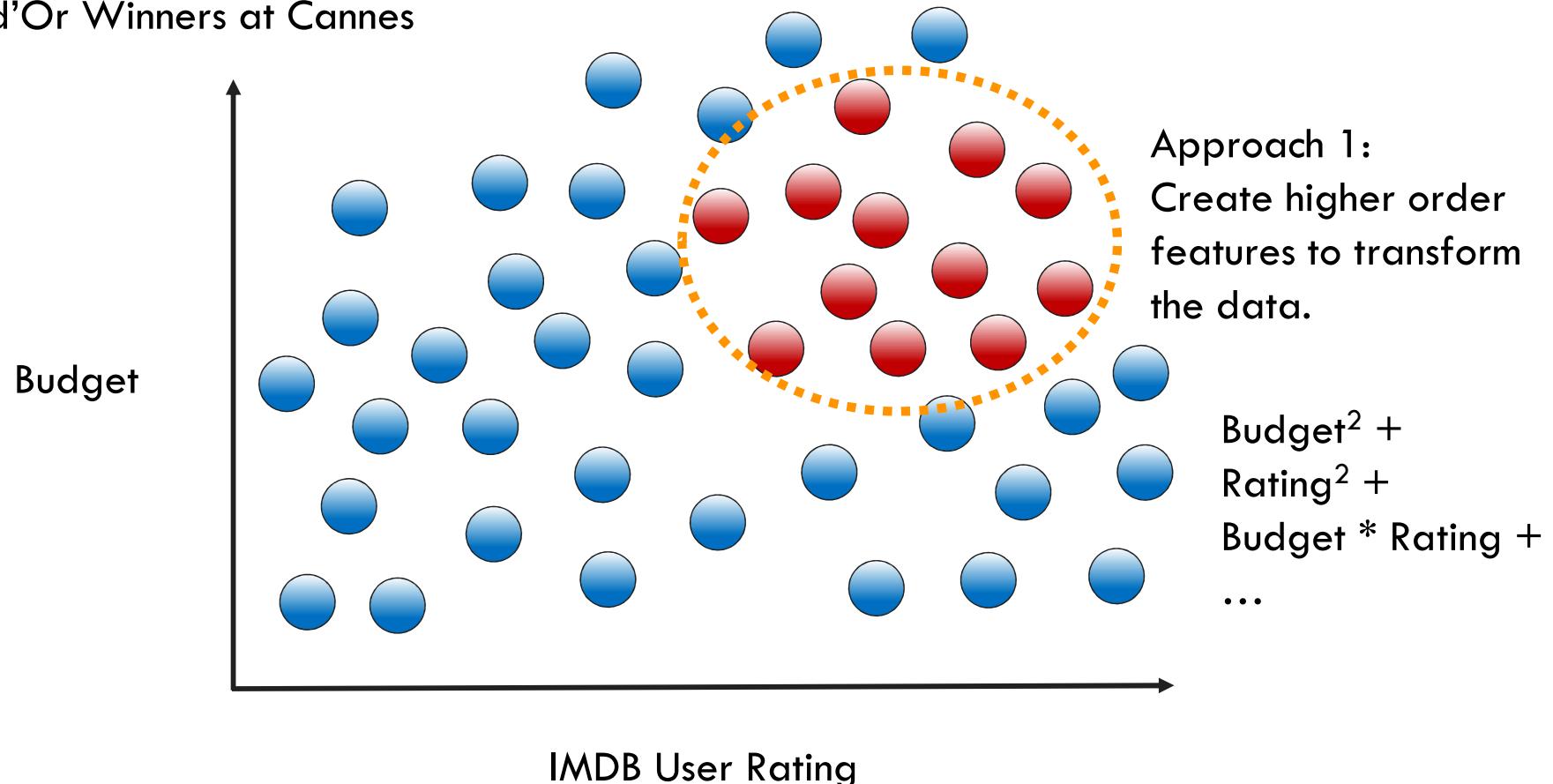
SVM Gaussian Kernel

Palme d'Or Winners at Cannes



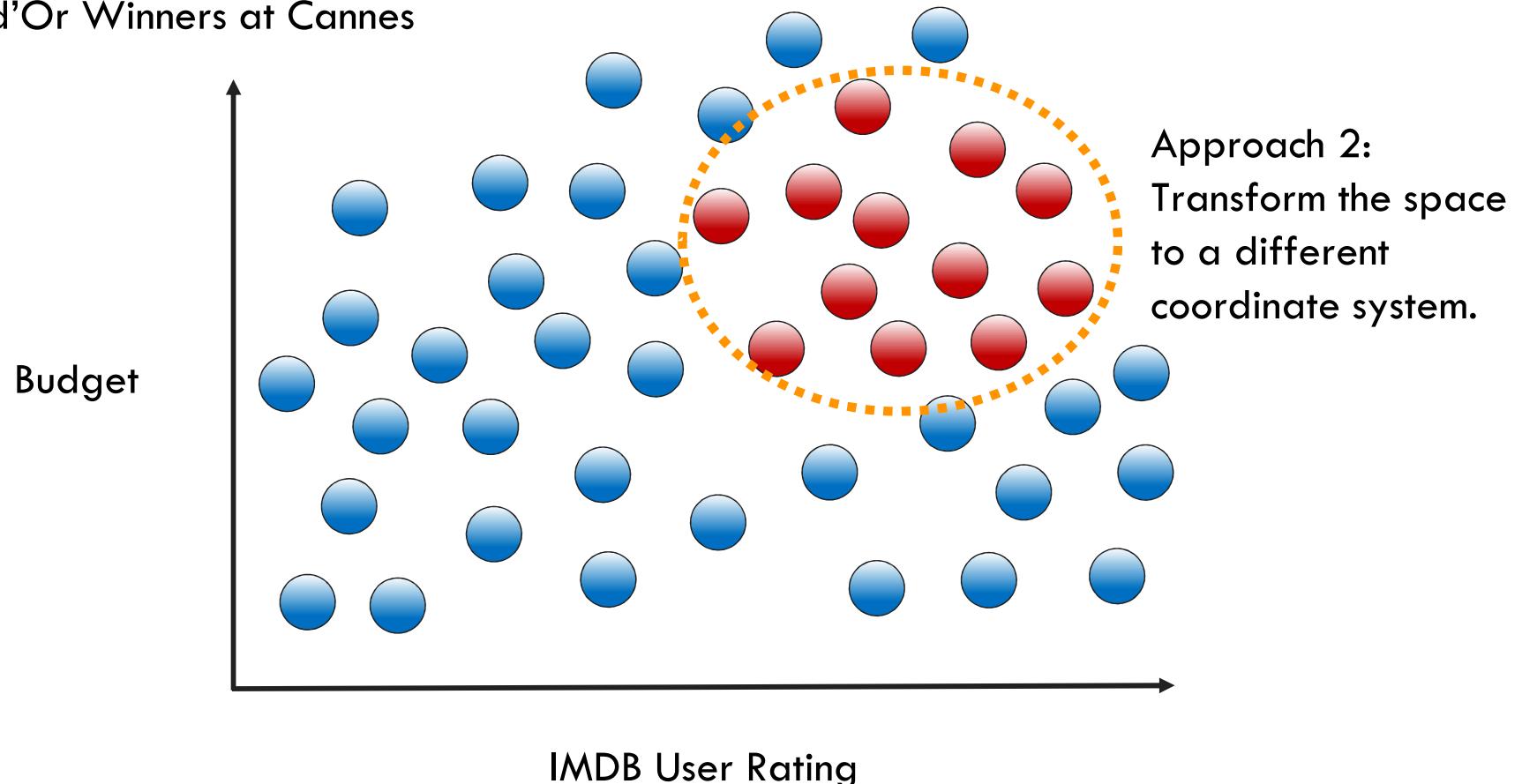
SVM Gaussian Kernel

Palme d'Or Winners at Cannes



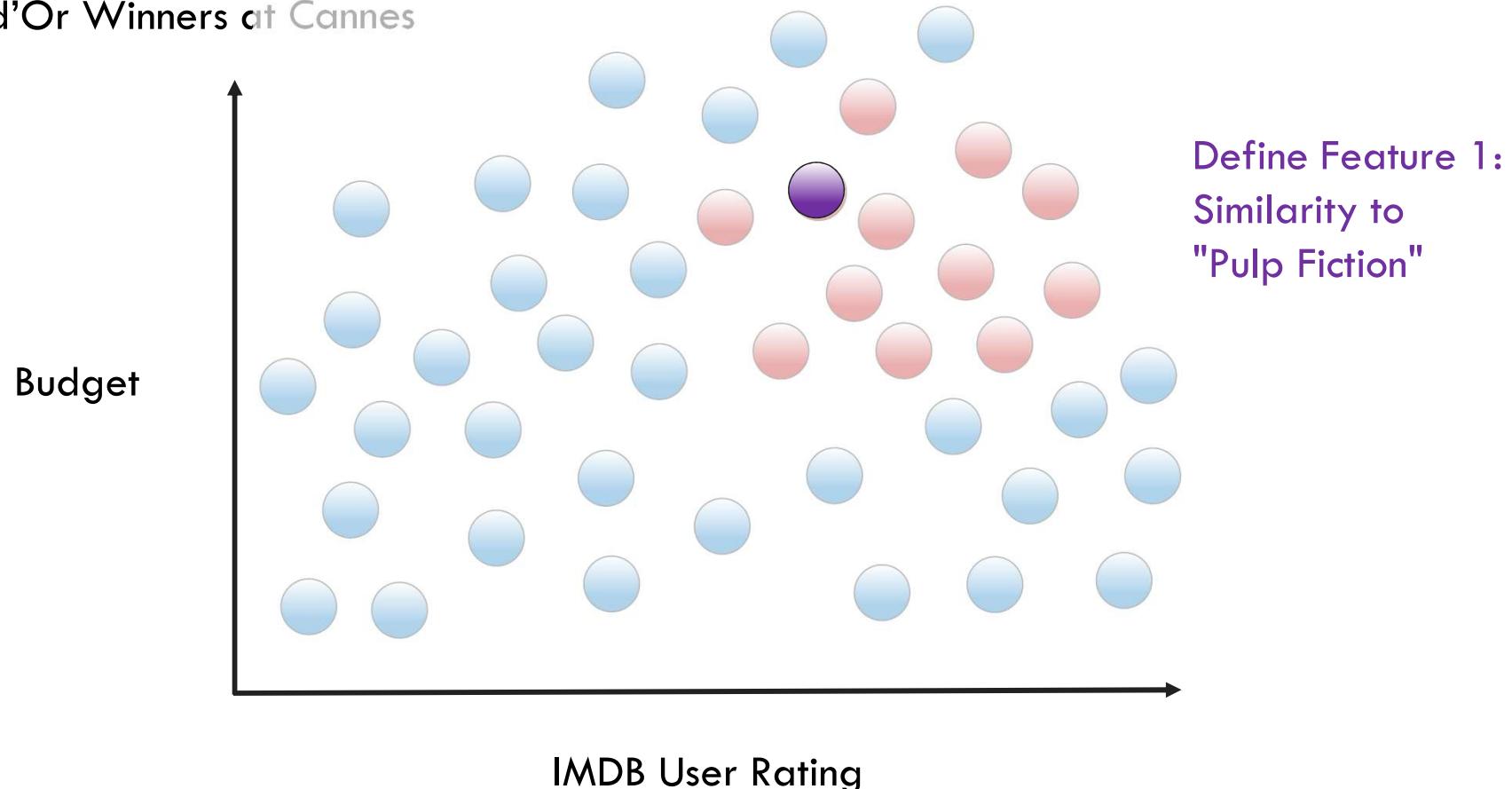
SVM Gaussian Kernel

Palme d'Or Winners at Cannes



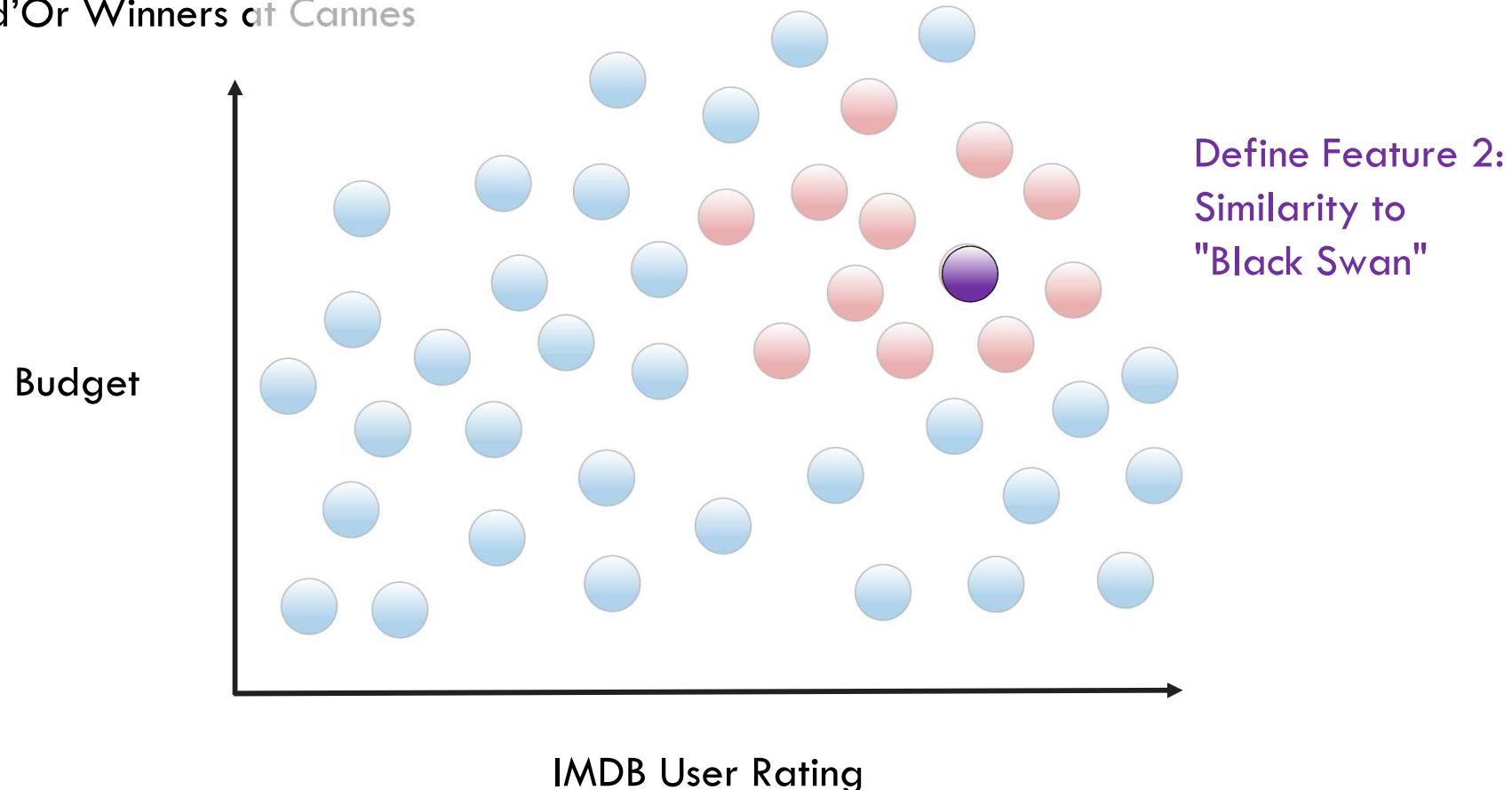
SVM Gaussian Kernel

Palme d'Or Winners at Cannes



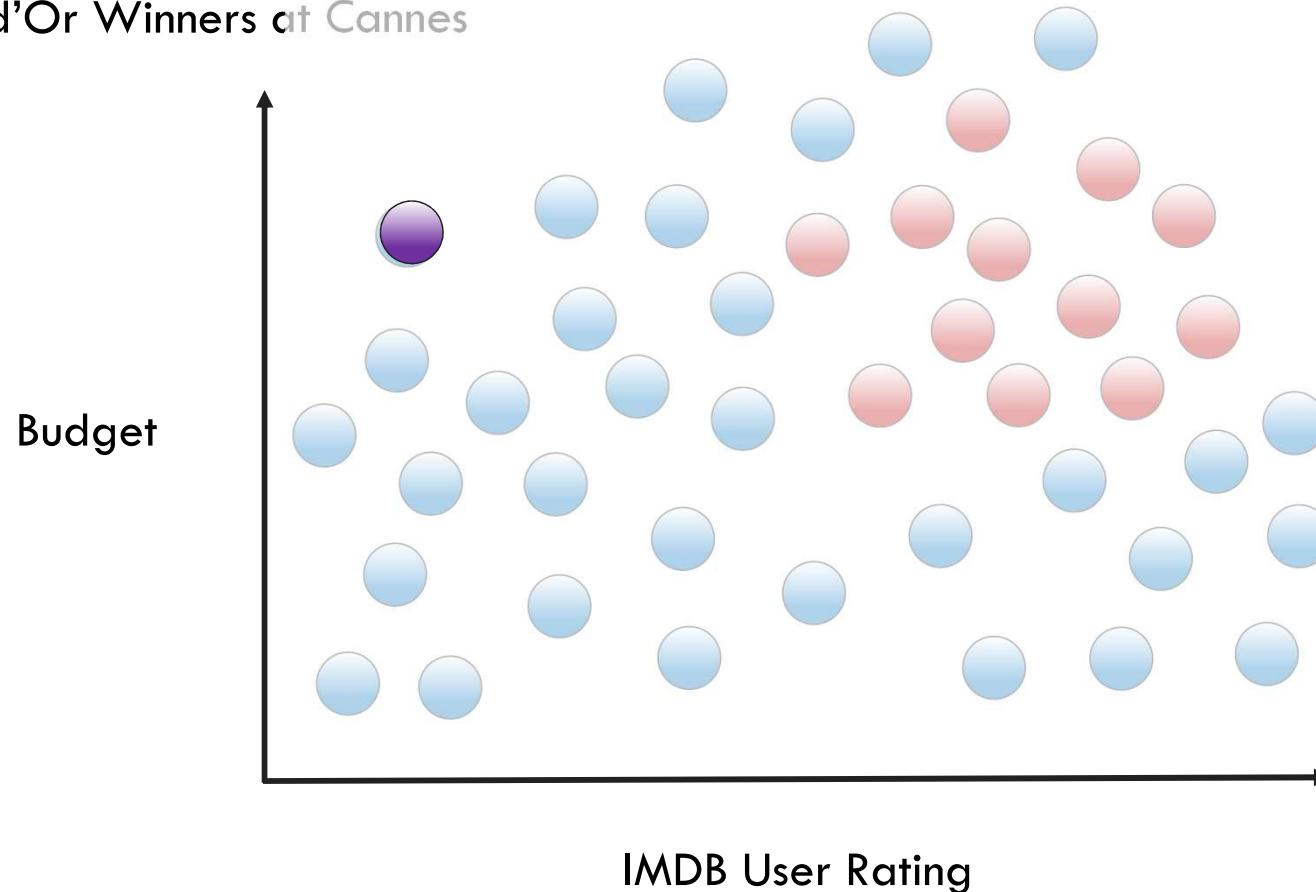
SVM Gaussian Kernel

Palme d'Or Winners at Cannes



SVM Gaussian Kernel

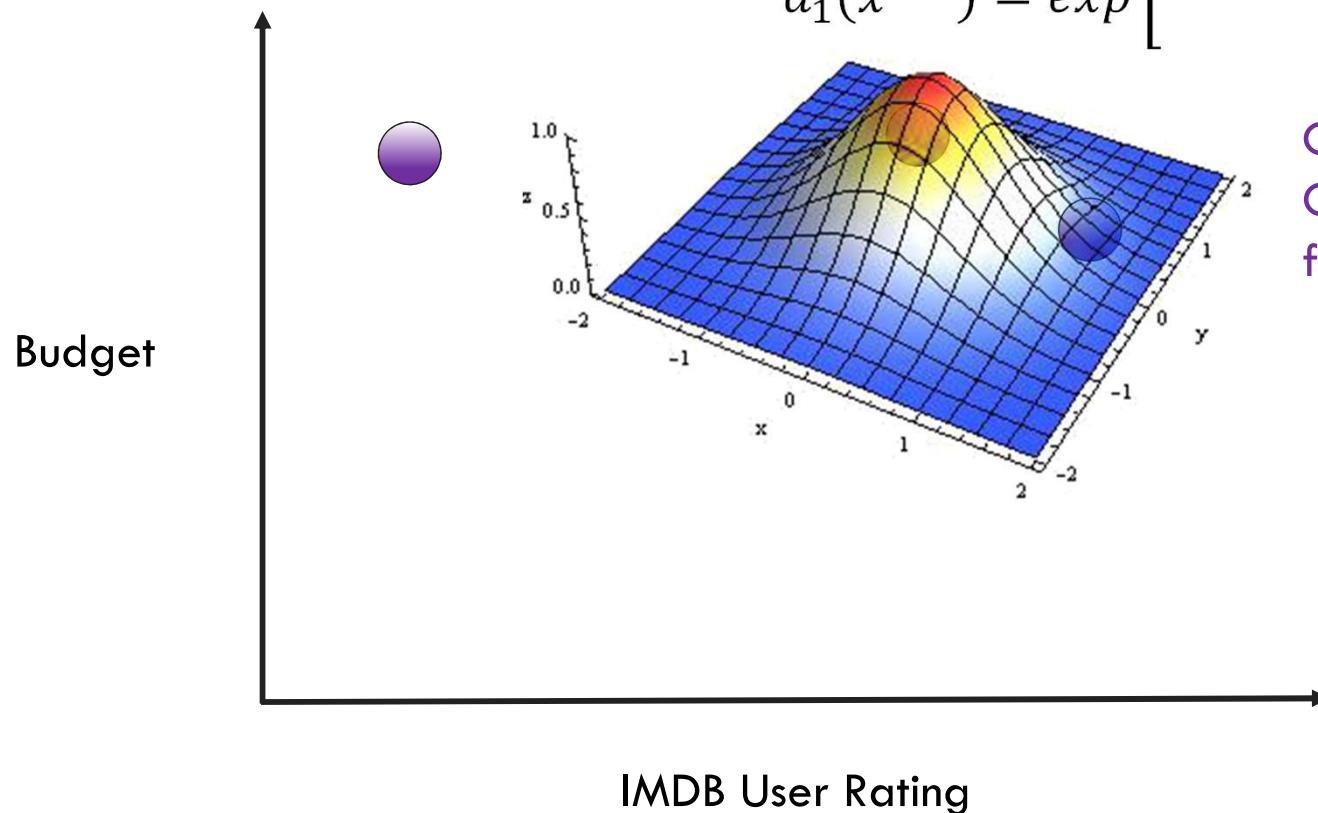
Palme d'Or Winners at Cannes



Define Feature 3:
Similarity to
"Transformers"

SVM Gaussian Kernel

Palme d'Or Winners at Cannes

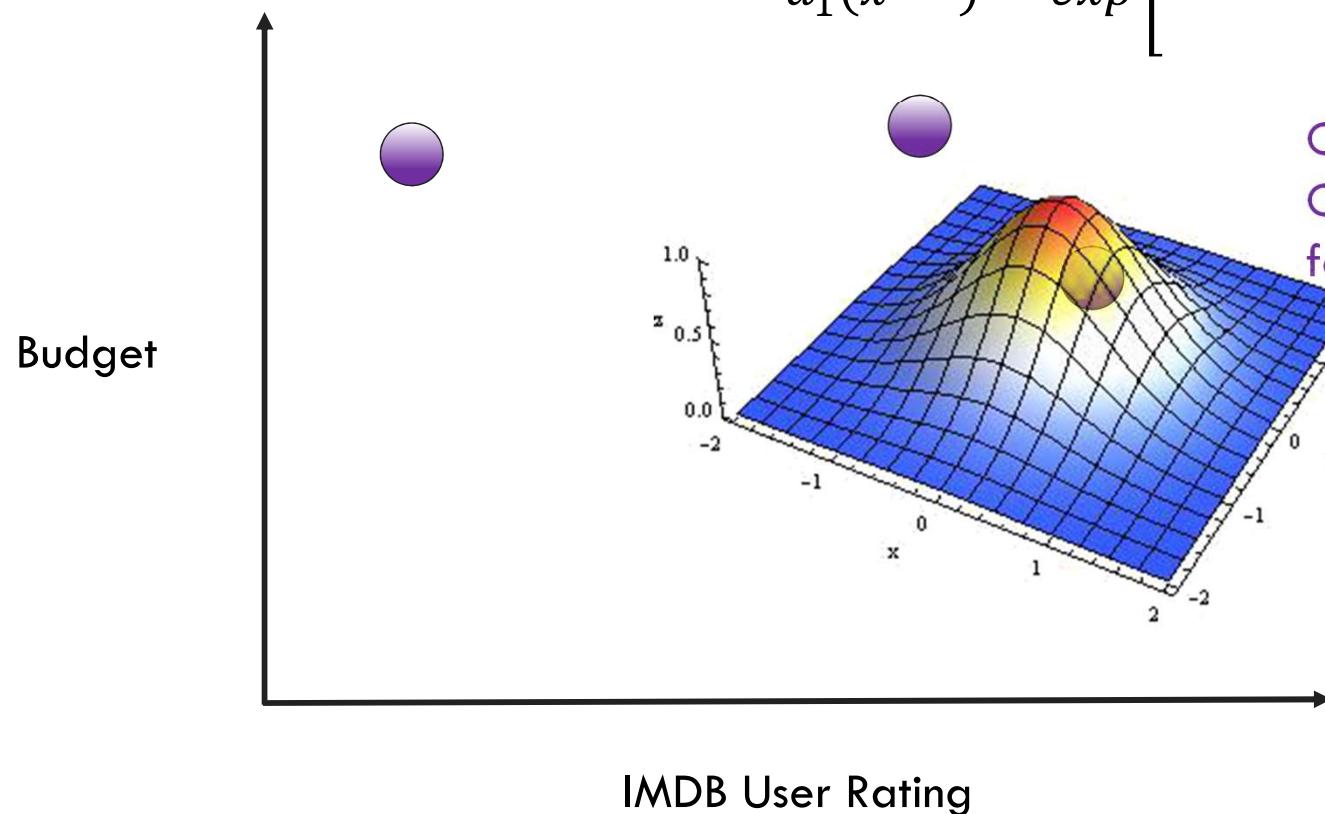


$$a_1(x^{obs}) = \exp \left[-\frac{\sum (x_i^{obs} - x_i^{Pulp Fiction})^2}{2\sigma^2} \right]$$

Create a
Gaussian function at
feature 1

SVM Gaussian Kernel

Palme d'Or Winners at Cannes

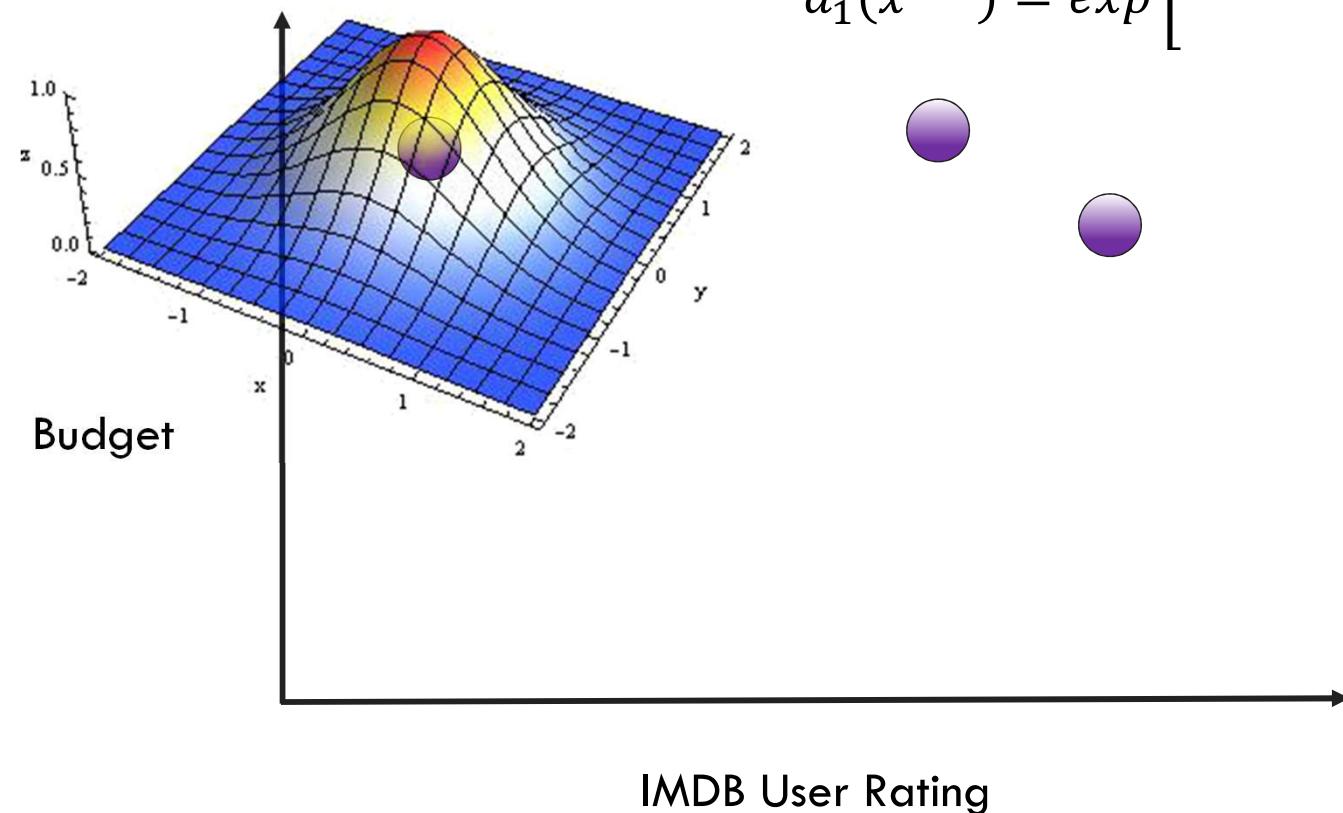


$$a_1(x_i^{obs}) = \exp \left[\frac{-\sum(x_i^{obs} - x_i^{Black Swan})^2}{2\sigma^2} \right]$$

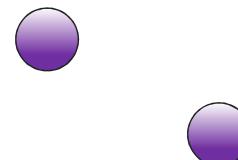
Create a
Gaussian function at
feature 2

SVM Gaussian Kernel

Palme d'Or Winners at Cannes

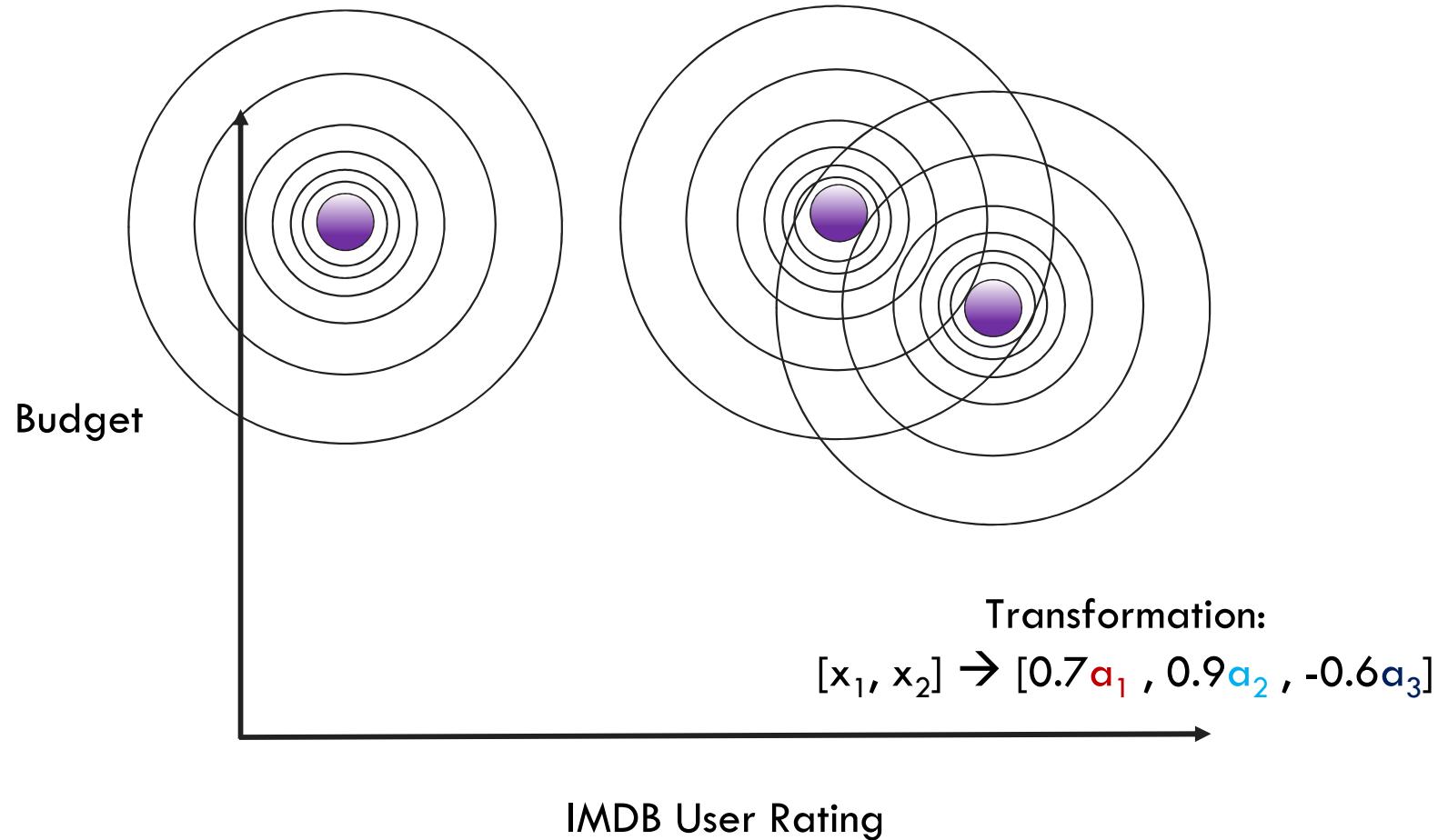


$$a_1(x^{obs}) = \exp \left[\frac{-\sum(x_i^{obs} - x_i^{Transformers})^2}{2\sigma^2} \right]$$

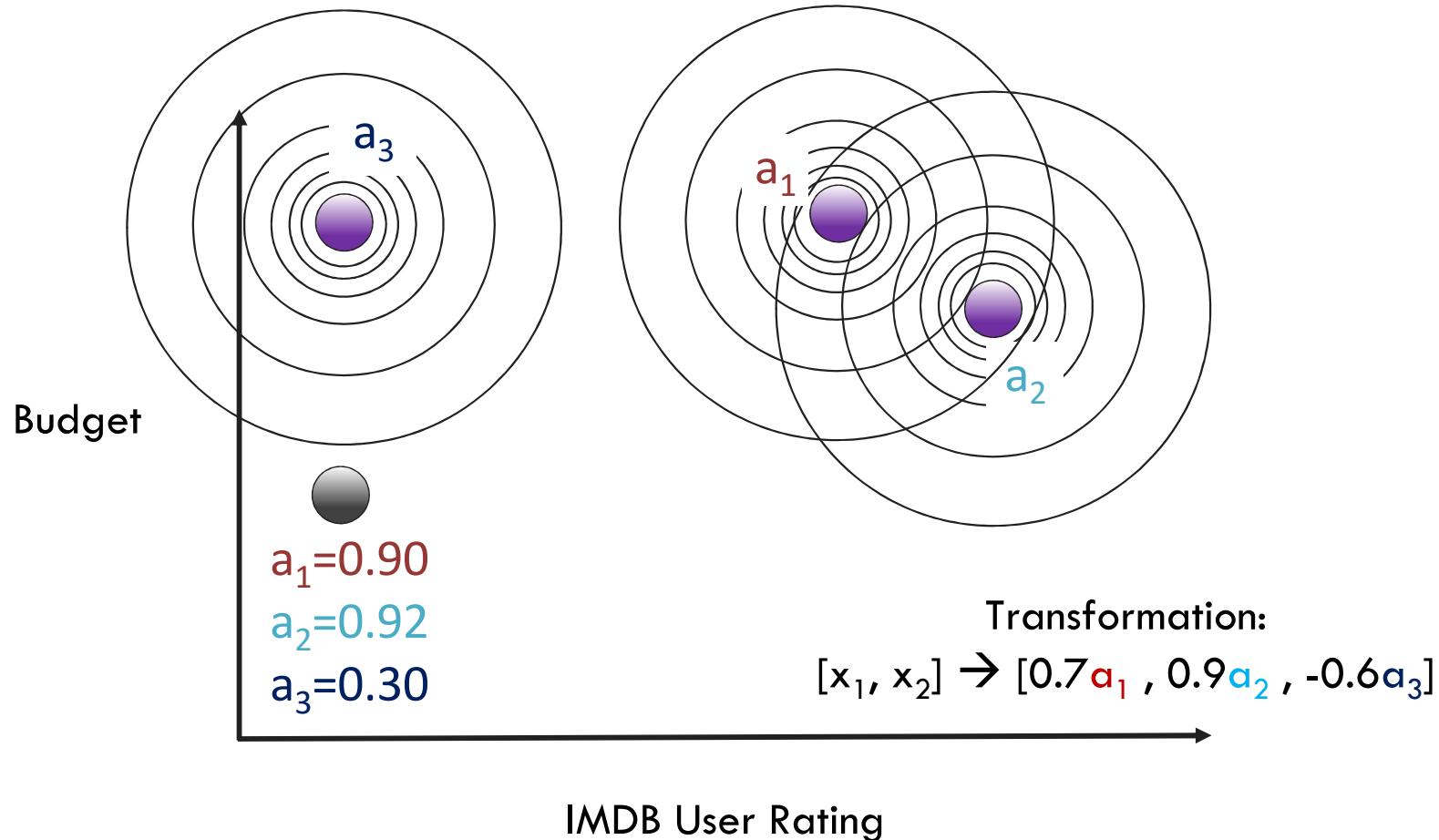


Create a
Gaussian function at
feature 3

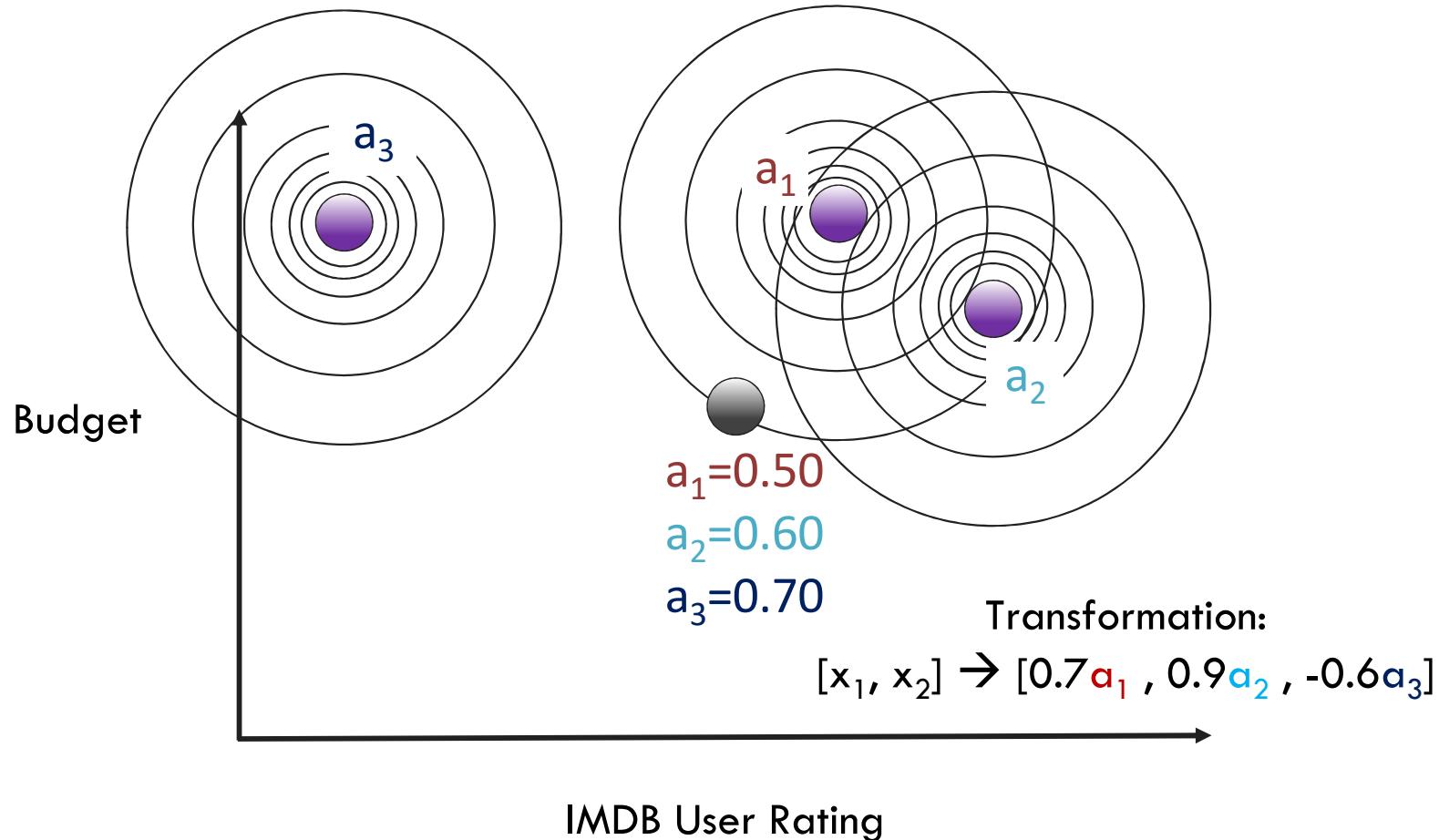
SVM Gaussian Kernel



SVM Gaussian Kernel



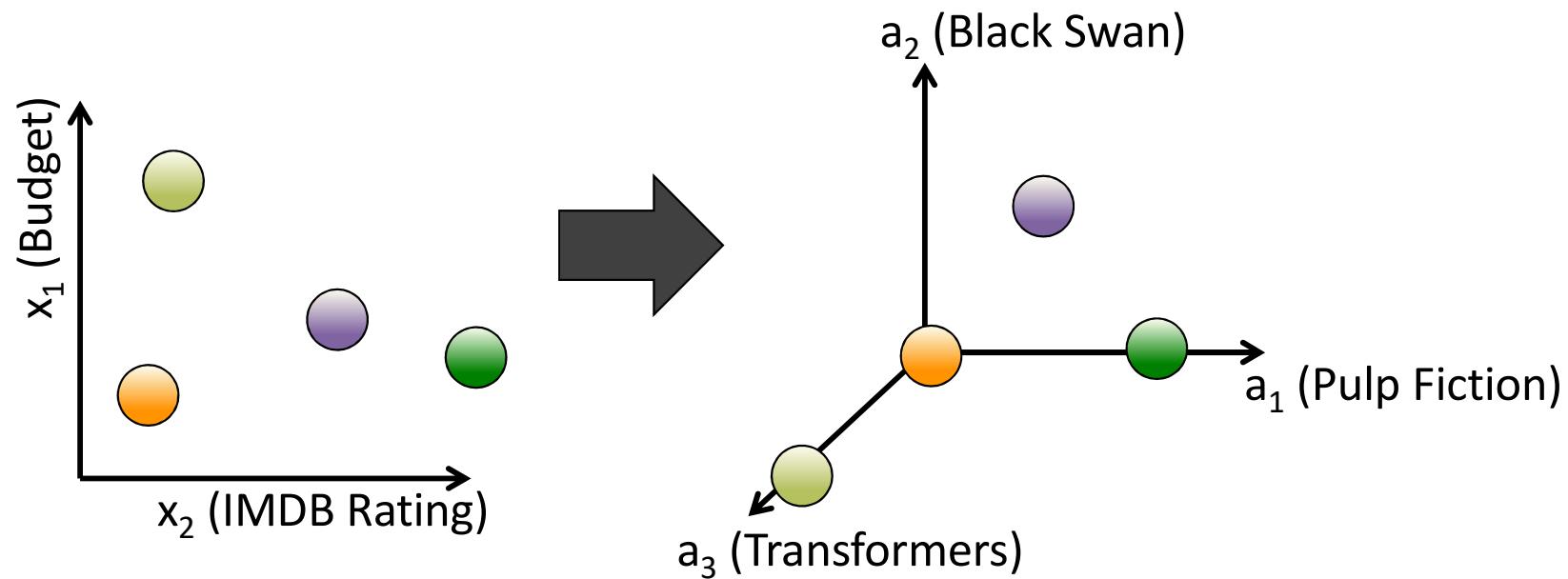
SVM Gaussian Kernel



SVM Gaussian Kernel

Transformation:

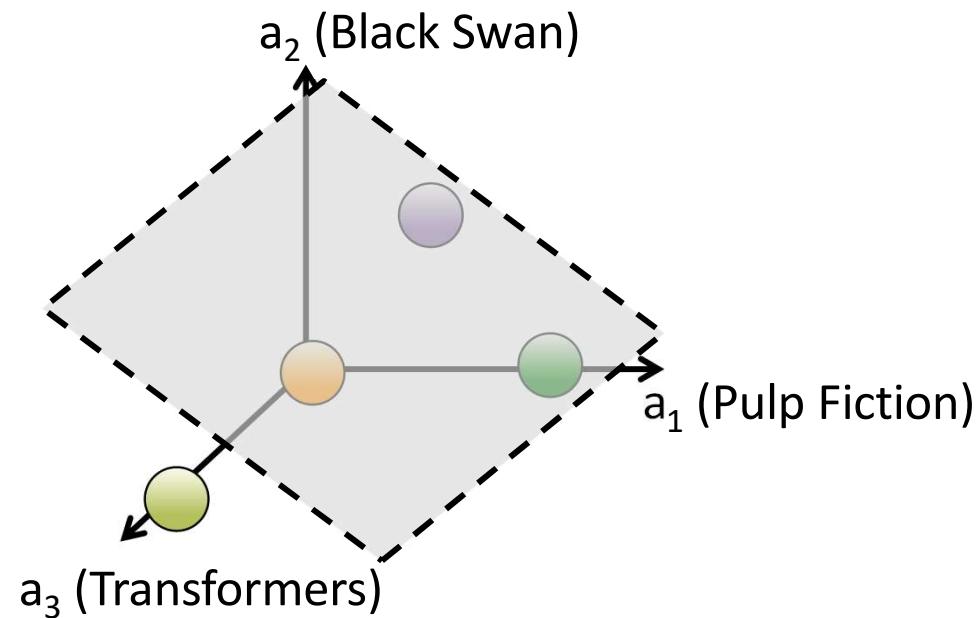
$$[x_1, x_2] \rightarrow [0.7a_1, 0.9a_2, -0.6a_3]$$



Classification in the New Space

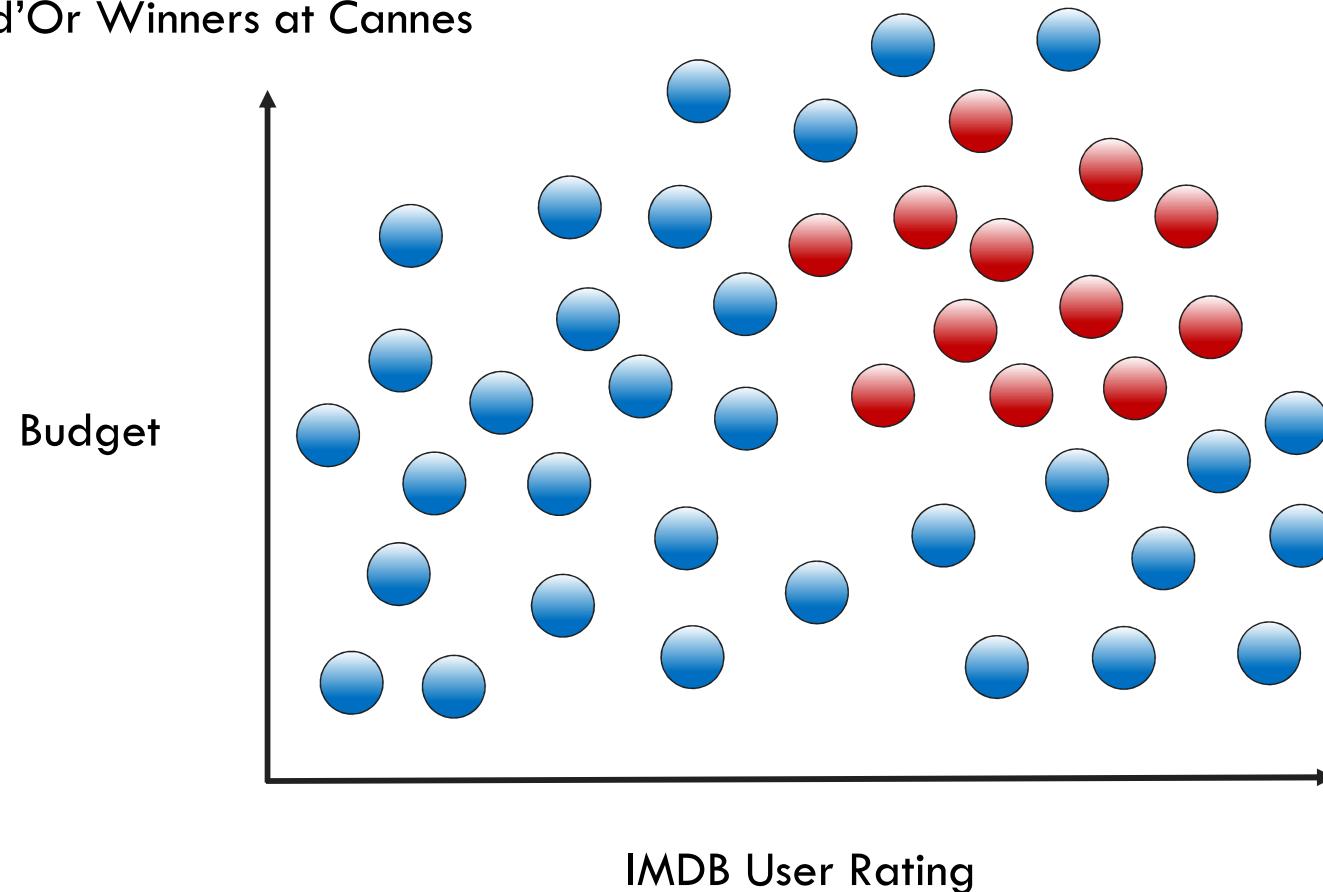
Transformation:

$$[x_1, x_2] \rightarrow [0.7a_1, 0.9a_2, -0.6a_3]$$



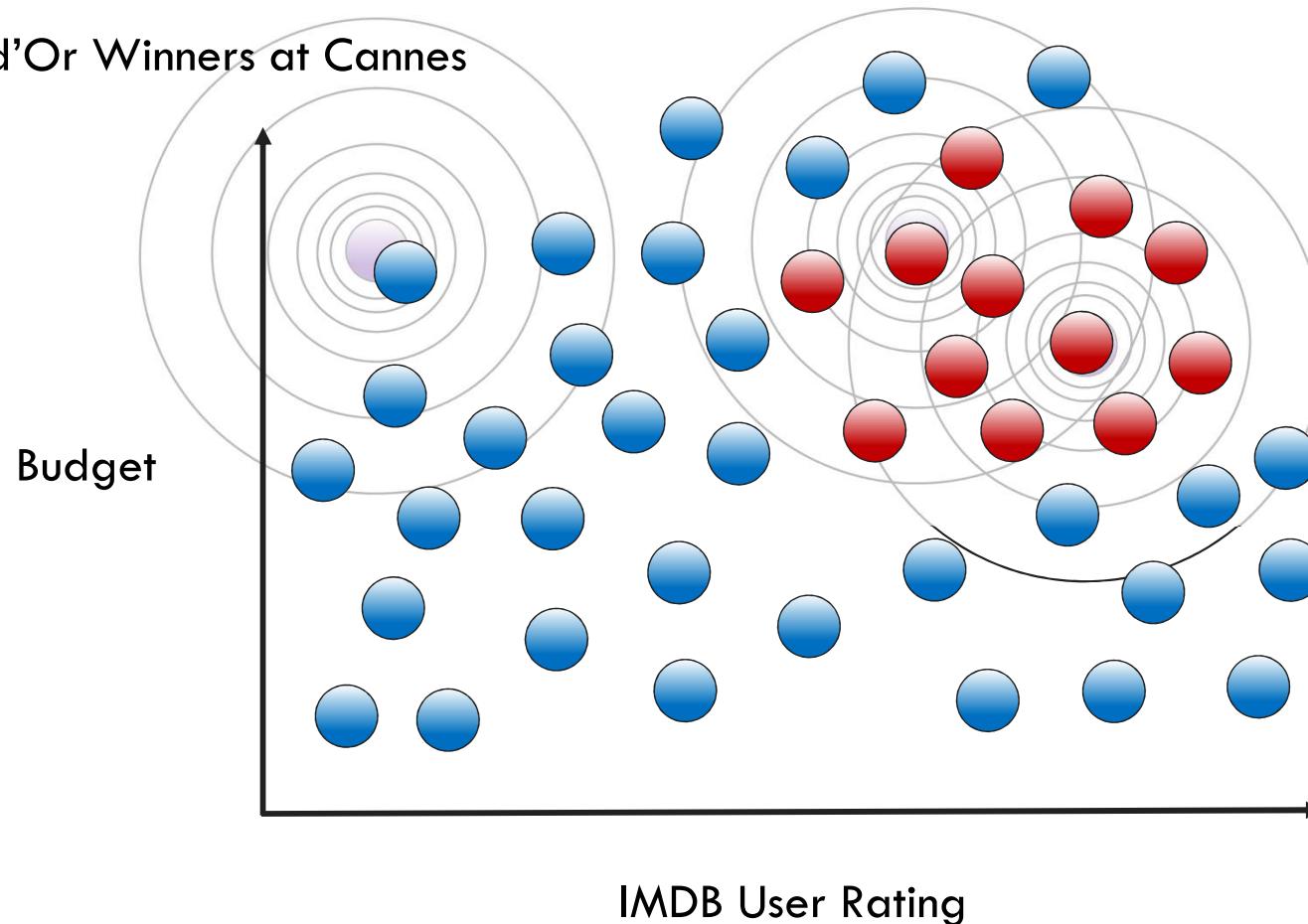
SVM Gaussian Kernel

Palme d'Or Winners at Cannes



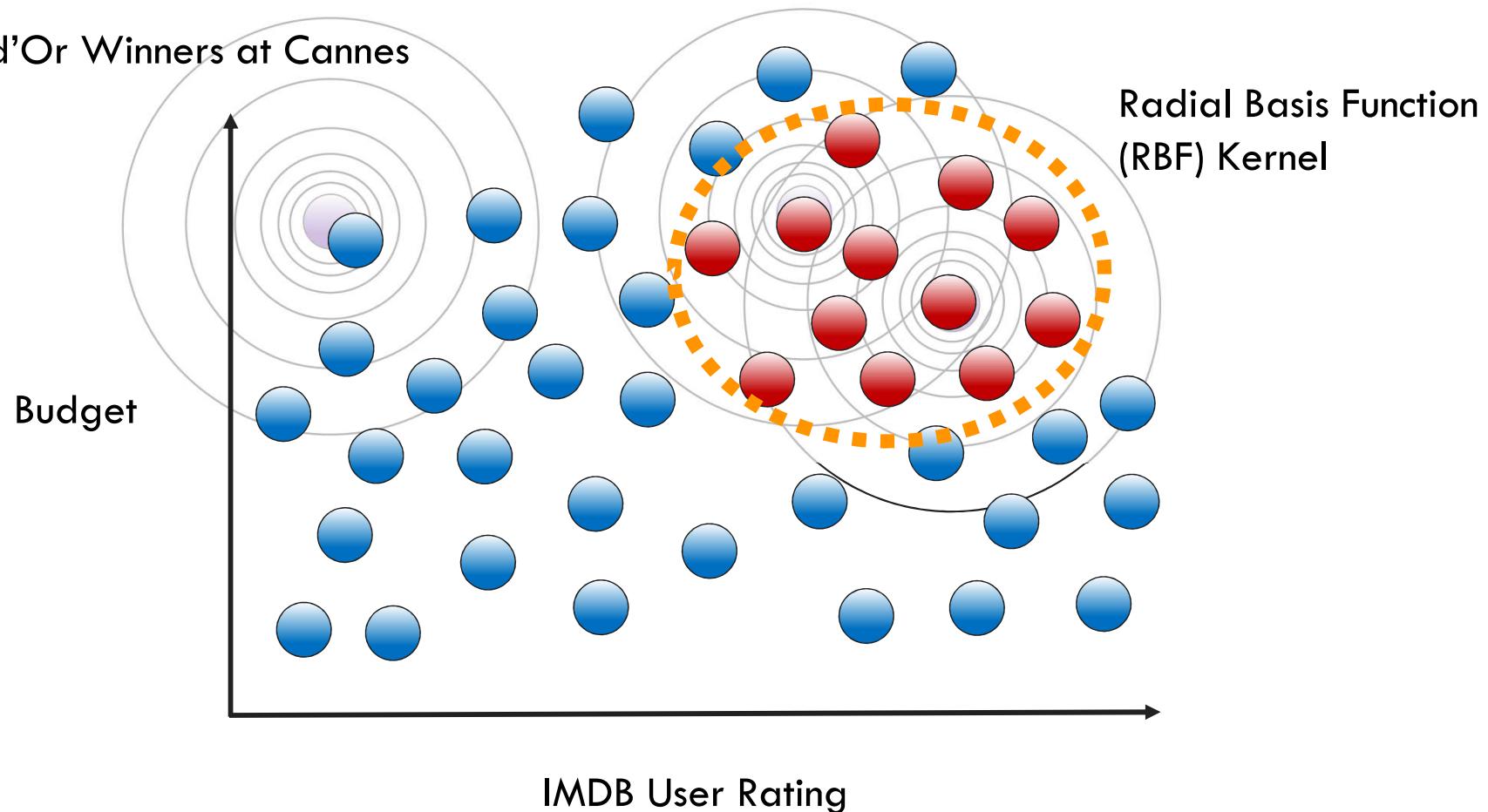
SVM Gaussian Kernel

Palme d'Or Winners at Cannes



SVM Gaussian Kernel

Palme d'Or Winners at Cannes



SVMs with Kernels: The Syntax

Import the class containing the classification method

```
from sklearn.svm import SVC
```

SVMs with Kernels: The Syntax

Import the class containing the classification method

```
from sklearn.svm import SVC
```

Create an instance of the class

```
rbfSVC = SVC(kernel='rbf', gamma=1.0, C=10.0)
```

SVMs with Kernels: The Syntax

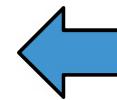
Import the class containing the classification method

```
from sklearn.svm import SVC
```

Create an instance of the class

```
rbfSVC = SVC(kernel='rbf', gamma=1.0, C=10.0)
```

set kernel and
associated
coefficient
(gamma)



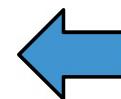
SVMs with Kernels: The Syntax

Import the class containing the classification method

```
from sklearn.svm import SVC
```

Create an instance of the class

```
rbfSVC = SVC(kernel='rbf', gamma=1.0, C=10.0)
```



"C" is penalty
associated with
the error term

SVMs with Kernels: The Syntax

Import the class containing the classification method

```
from sklearn.svm import SVC
```

Create an instance of the class

```
rbfSVC = SVC(kernel='rbf', gamma=1.0, C=10.0)
```

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

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

```
y_predict = rbfSVC.predict(X_test)
```

SVMs with Kernels: The Syntax

Import the class containing the classification method

```
from sklearn.svm import SVC
```

Create an instance of the class

```
rbfSVC = SVC(kernel='rbf', gamma=1.0, C=10.0)
```

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

```
rbfSVC = rbfSVC.fit(X_train, y_train)  
y_predict = rbfSVC.predict(X_test)
```

Tune kernel and associated parameters with cross-validation.

Feature Overload

Problem

SVMs with RBF Kernels are very slow to train
with lots of features or data

Feature Overload

Problem

SVMs with RBF Kernels are very slow to train with lots of features or data

Solution

Construct approximate kernel map with SGD using Nystroem or RBF sampler

Feature Overload

Problem

SVMs with RBF Kernels are very slow to train with lots of features or data

Solution

Construct approximate kernel map with SGD using Nystroem or RBF sampler.
Fit a linear classifier.

Faster Kernel Transformations: The Syntax

Import the class containing the classification method

```
from sklearn.kernel_approximation import Nystroem
```

Create an instance of the class

```
nystroemSVC = Nystroem(kernel='rbf', gamma=1.0,  
n_components=100)
```

Fit the instance on the data and transform

```
X_train = nystroemSVC.fit_transform(X_train)  
X_test = nystroemSVC.transform(X_test)
```

Tune kernel parameters and components with cross-validation.

Faster Kernel Transformations: The Syntax

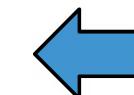
Import the class containing the classification method

```
from sklearn.kernel_approximation import Nystroem
```

Create an instance of the class

```
nystroemSVC = Nystroem(kernel='rbf', gamma=1.0,  
n_components=100)
```

multiple non-linear
kernels can be
used



Fit the instance on the data and transform

```
X_train = nystroemSVC.fit_transform(X_train)  
X_test = nystroemSVC.transform(X_test)
```

Tune kernel parameters and components with cross-validation.

Faster Kernel Transformations: The Syntax

Import the class containing the classification method

```
from sklearn.kernel_approximation import Nystroem
```

Create an instance of the class

```
nystroemSVC = Nystroem(kernel='rbf', gamma=1.0,  
n_components=100)
```

kernel and
gamma are
identical to SVC



Fit the instance on the data and transform

```
X_train = nystroemSVC.fit_transform(X_train)  
X_test = nystroemSVC.transform(X_test)
```

Tune kernel parameters and components with cross-validation.

Faster Kernel Transformations: The Syntax

Import the class containing the classification method

```
from sklearn.kernel_approximation import Nystroem
```

Create an instance of the class

```
nystroemSVC = Nystroem(kernel='rbf', gamma=1.0,  
n_components=100)
```



`n_components` is
number of
samples

Fit the instance on the data and transform

```
X_train = nystroemSVC.fit_transform(X_train)  
X_test = nystroemSVC.transform(X_test)
```

Tune kernel parameters and components with cross-validation.

Faster Kernel Transformations: The Syntax

Import the class containing the classification method

```
from sklearn.kernel_approximation import RBFsampler
```

Create an instance of the class

```
rbfSample = RBFsampler(gamma=1.0,  
                      n_components=100)
```

Fit the instance on the data and transform

```
X_train = rbfSample.fit_transform(X_train)  
X_test = rbfSample.transform(X_test)
```

Tune kernel parameters and components with cross-validation.

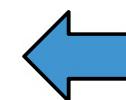
Faster Kernel Transformations: The Syntax

Import the class containing the classification method

```
from sklearn.kernel_approximation import RBFsampler
```

Create an instance of the class

```
rbfSample = RBFsampler(gamma=1.0,  
                      n_components=100)
```



RBF is only kernel
that can be used

Fit the instance on the data and transform

```
X_train = rbfSample.fit_transform(X_train)  
X_test = rbfSample.transform(X_test)
```

Tune kernel parameters and components with cross-validation.

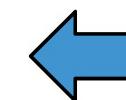
Faster Kernel Transformations: The Syntax

Import the class containing the classification method

```
from sklearn.kernel_approximation import RBFsampler
```

Create an instance of the class

```
rbfSample = RBFsampler(gamma=1.0,  
                      n_components=100)
```



parameter names
are identical to
previous

Fit the instance on the data and transform

```
X_train = rbfSample.fit_transform(X_train)  
X_test = rbfSample.transform(X_test)
```

Tune kernel parameters and components with cross-validation.

When to Use Logistic Regression vs SVC

Features

Many (~10K Features)

Few (<100 Features)

Few (<100 Features)

Data

Small (1K rows)

Medium (~10k rows)

Many (>100K Points)

Model Choice

Simple, Logistic or LinearSVC

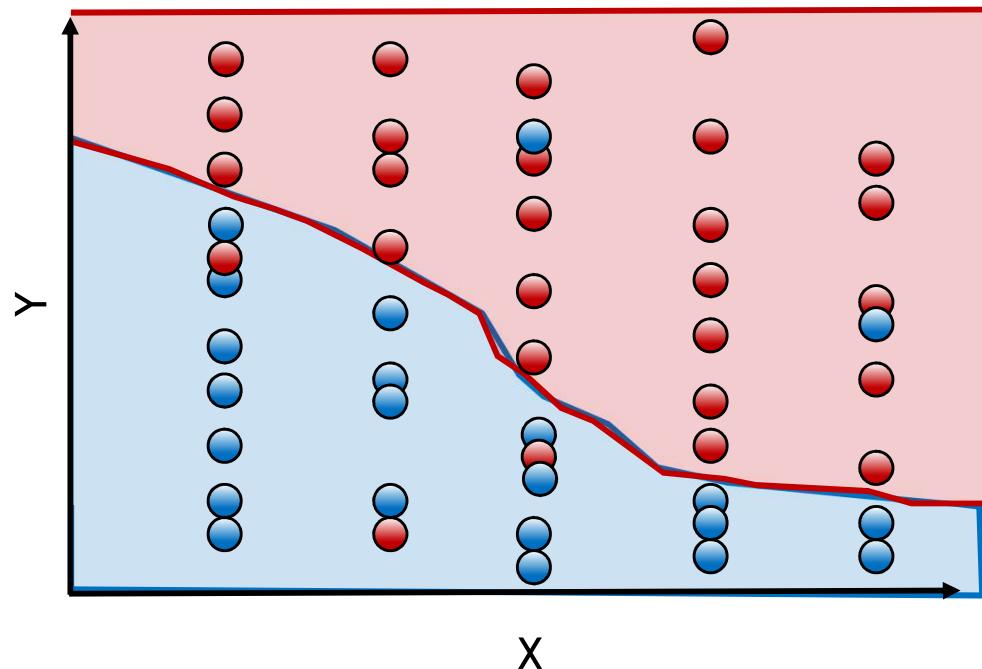
SVC with RBF

Add features, Logistic, LinearSVC or
Kernel Approx.



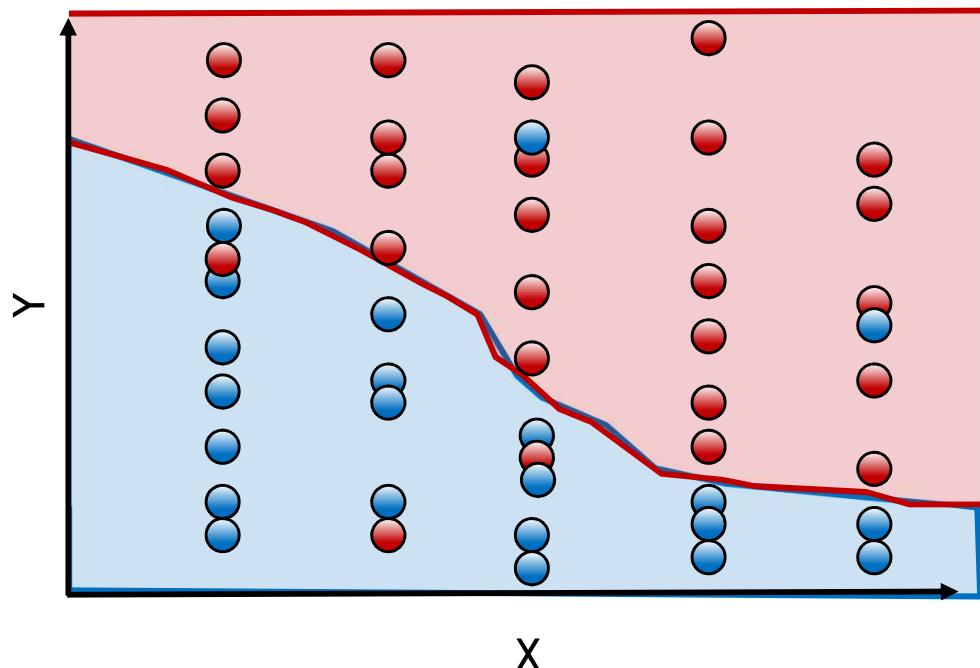
2.13. Árvores de Decisão

Overview of Classifier Characteristics



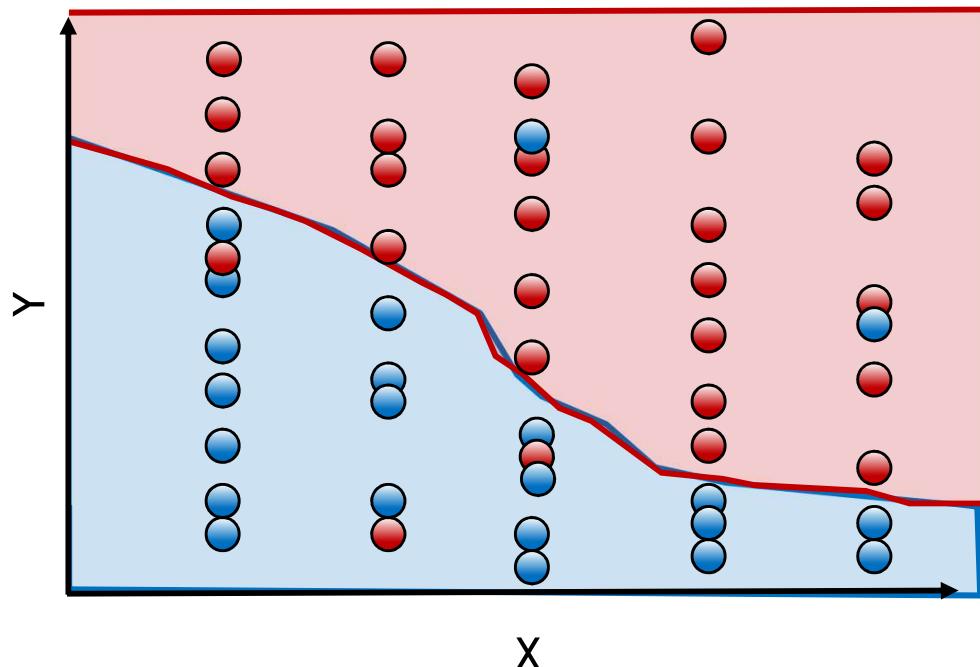
- For K-Nearest Neighbors,
training data is the model

Overview of Classifier Characteristics



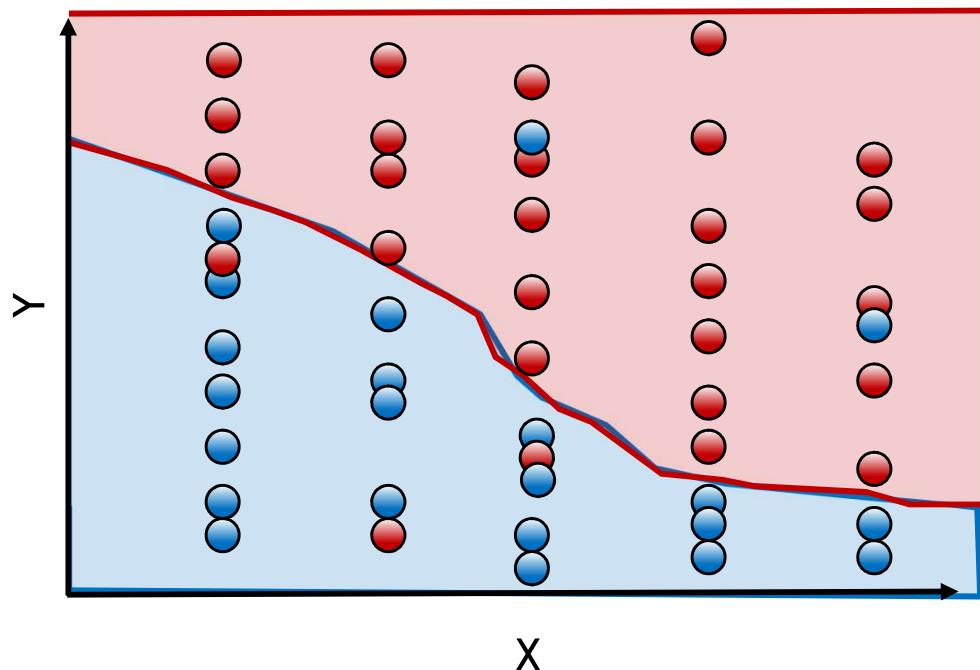
- For K-Nearest Neighbors,
training data is the model
- Fitting is fast—just store data

Overview of Classifier Characteristics



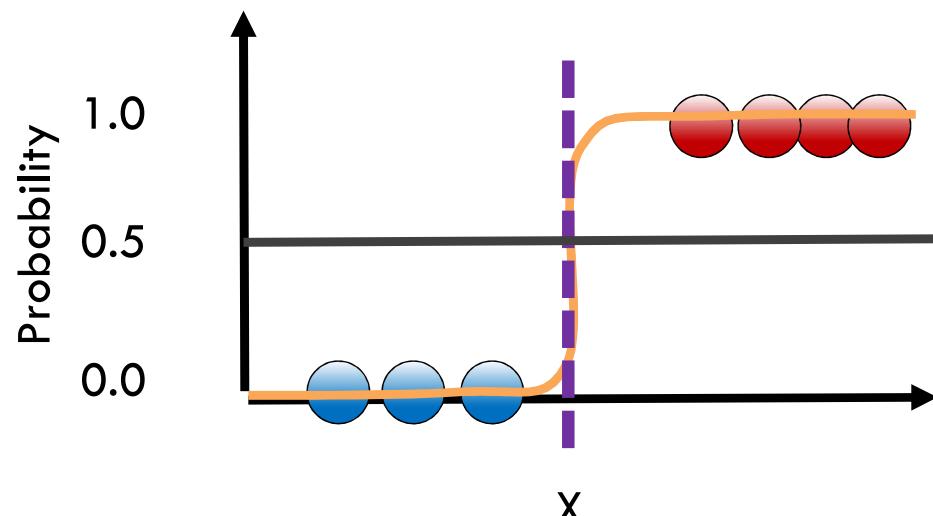
- For K-Nearest Neighbors, training data is the model
- Fitting is fast—just store data
- Prediction can be slow—lots of distances to measure

Overview of Classifier Characteristics



- For K-Nearest Neighbors,
training data is the model
- Fitting is fast—just store data
- Prediction can be slow—lots of
distances to measure
- Decision boundary is flexible

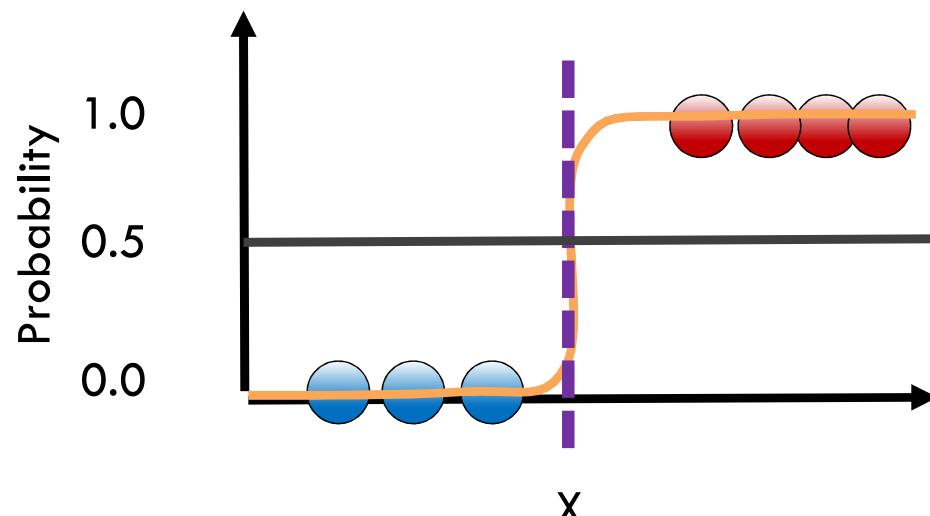
Overview of Classifier Characteristics



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

- For logistic regression, model is just parameters

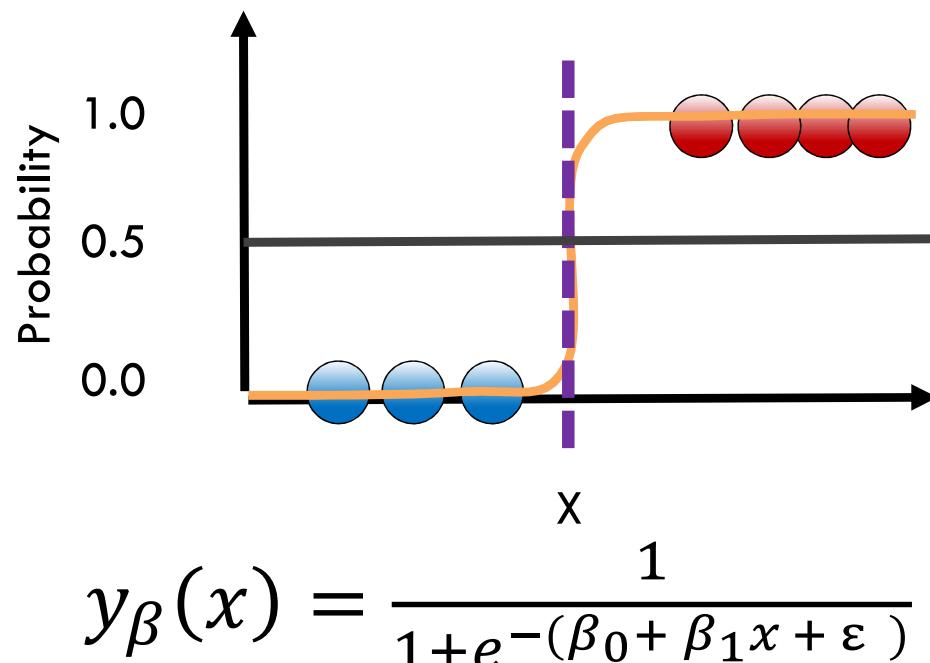
Overview of Classifier Characteristics



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

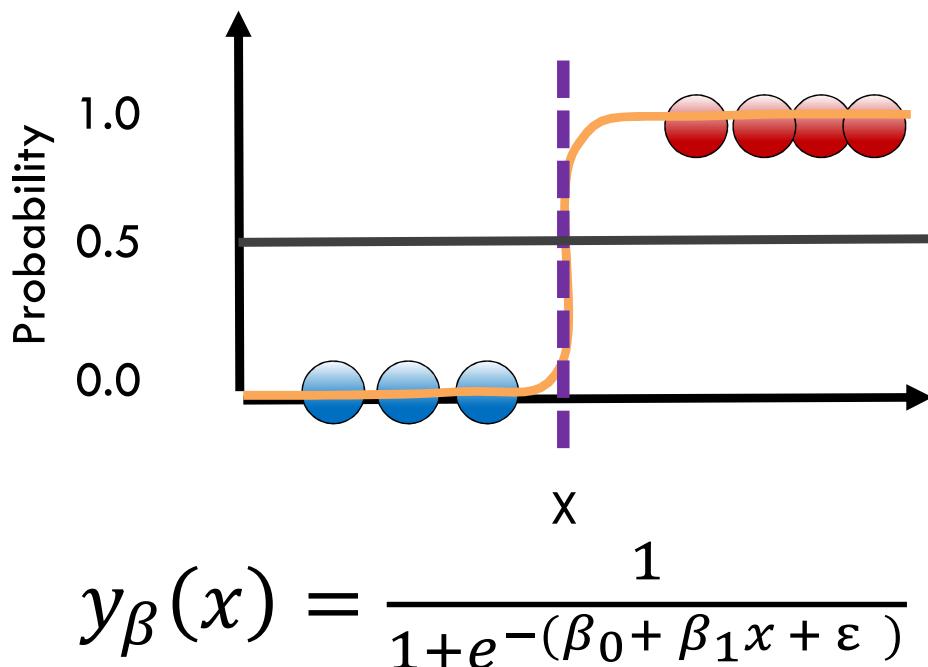
- For logistic regression, model is just parameters
- Fitting can be slow—must find best parameters

Overview of Classifier Characteristics



- For logistic regression, model is just parameters
- Fitting can be slow—must find best parameters
- Prediction is fast—calculate expected value

Overview of Classifier Characteristics



- For logistic regression, model is just parameters
- Fitting can be slow—must find best parameters
- Prediction is fast—calculate expected value
- Decision boundary is simple, less flexible

Introduction to Decision Trees

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

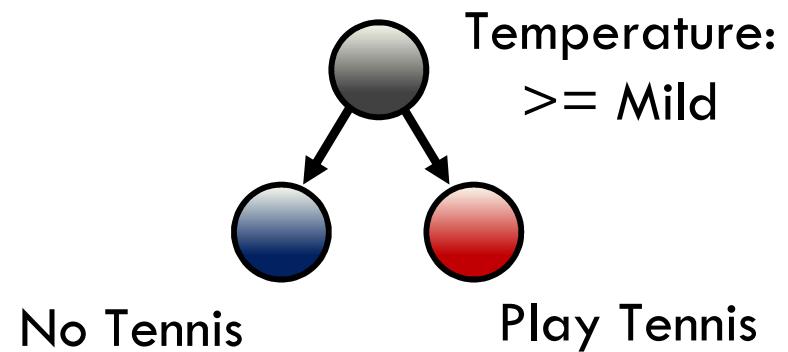
Introduction to Decision Trees

- Want to predict whether to play tennis based on temperature, humidity, wind, outlook

al

Introduction to Decision Trees

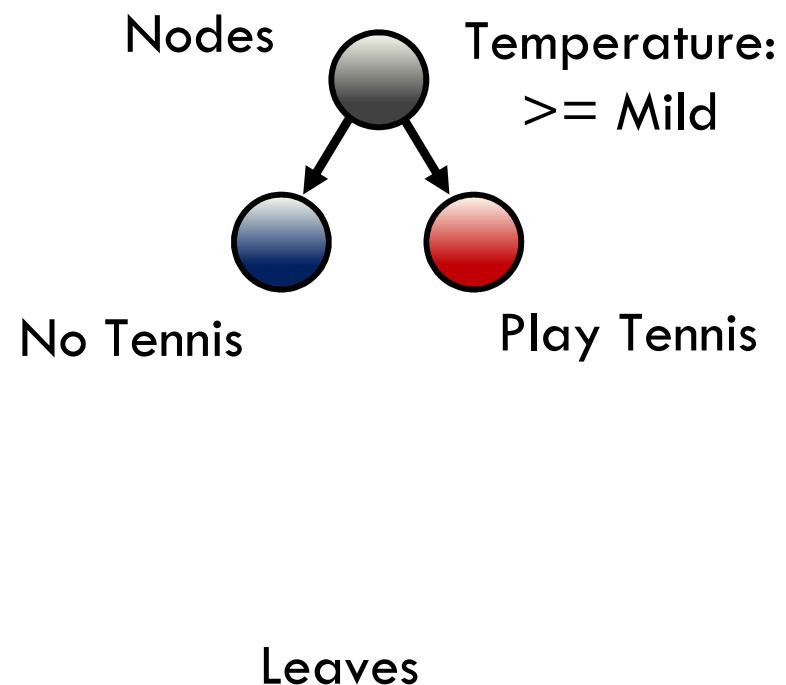
- Want to predict whether to play tennis based on temperature, humidity, wind, outlook
- Segment data based on features to predict result



al

Introduction to Decision Trees

- Want to predict whether to play tennis based on temperature, humidity, wind, outlook
- Segment data based on features to predict result

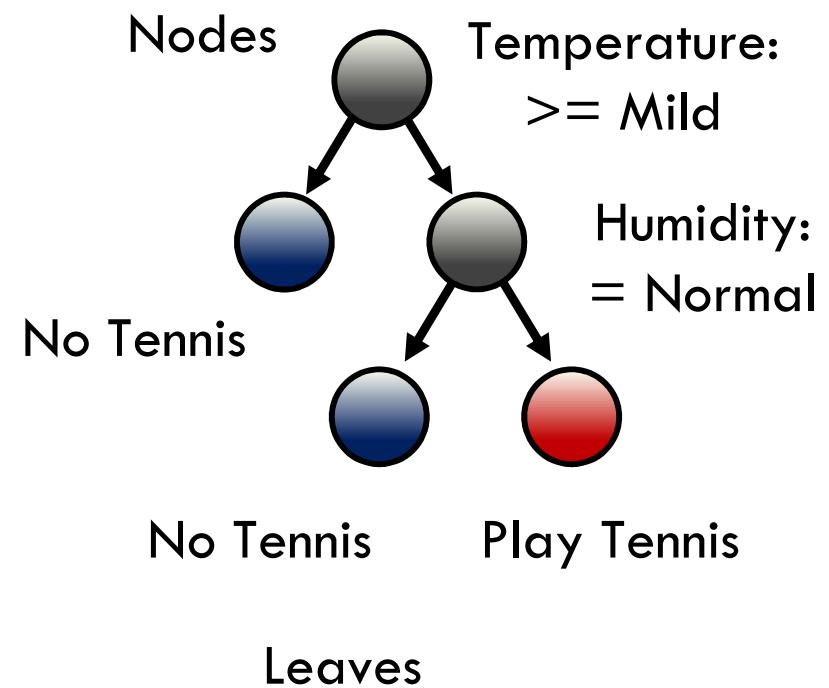


al

Introduction to Decision Trees

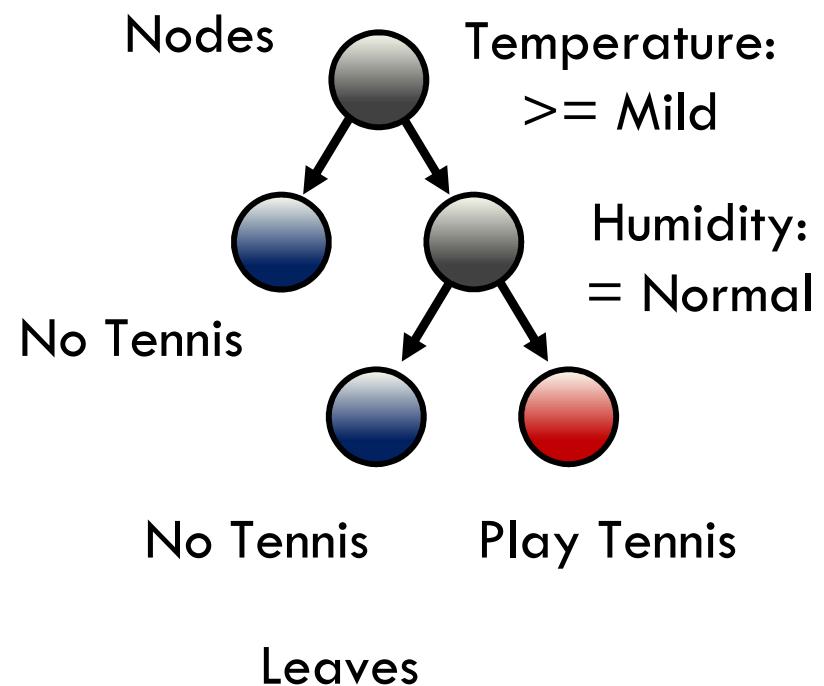
- Want to predict whether to play tennis based on temperature, humidity, wind, outlook
- Segment data based on features to predict result

al



Introduction to Decision Trees

- Want to predict whether to play tennis based on temperature, humidity, wind, outlook
- Segment data based on features to predict result
- Trees that predict categorical results are **decision trees**

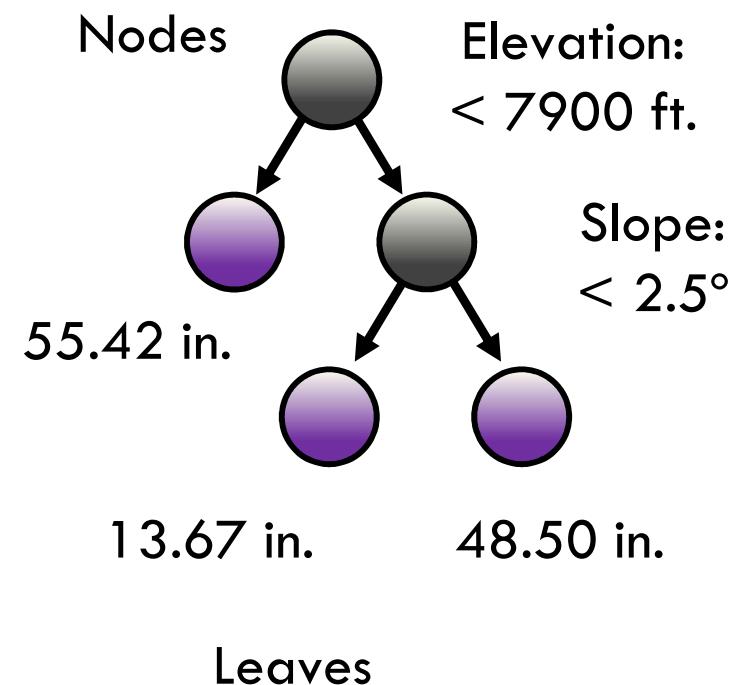


Regression Trees Predict Continuous Values

- Example: use slope and elevation in Himalayas
- Predict average precipitation (continuous value)

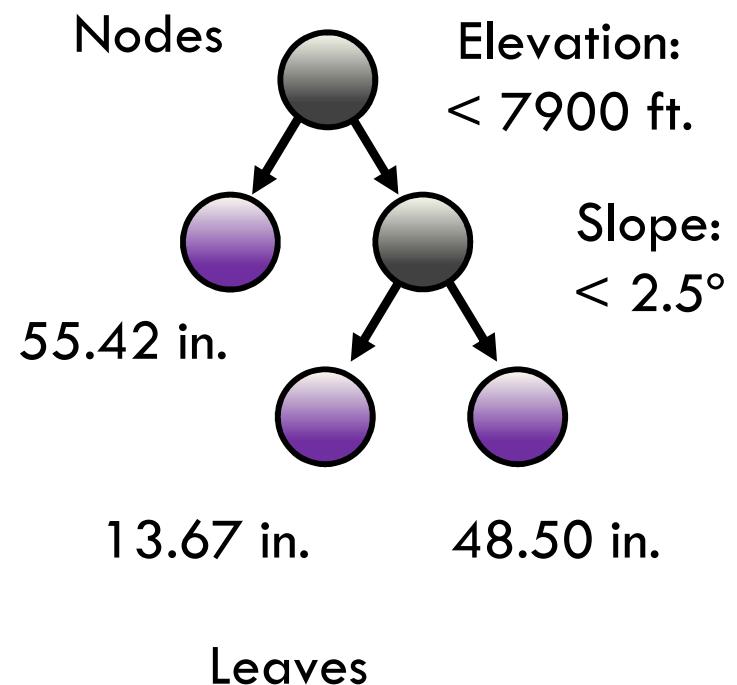
Regression Trees Predict Continuous Values

- Example: use slope and elevation in Himalayas
- Predict average precipitation (continuous value)

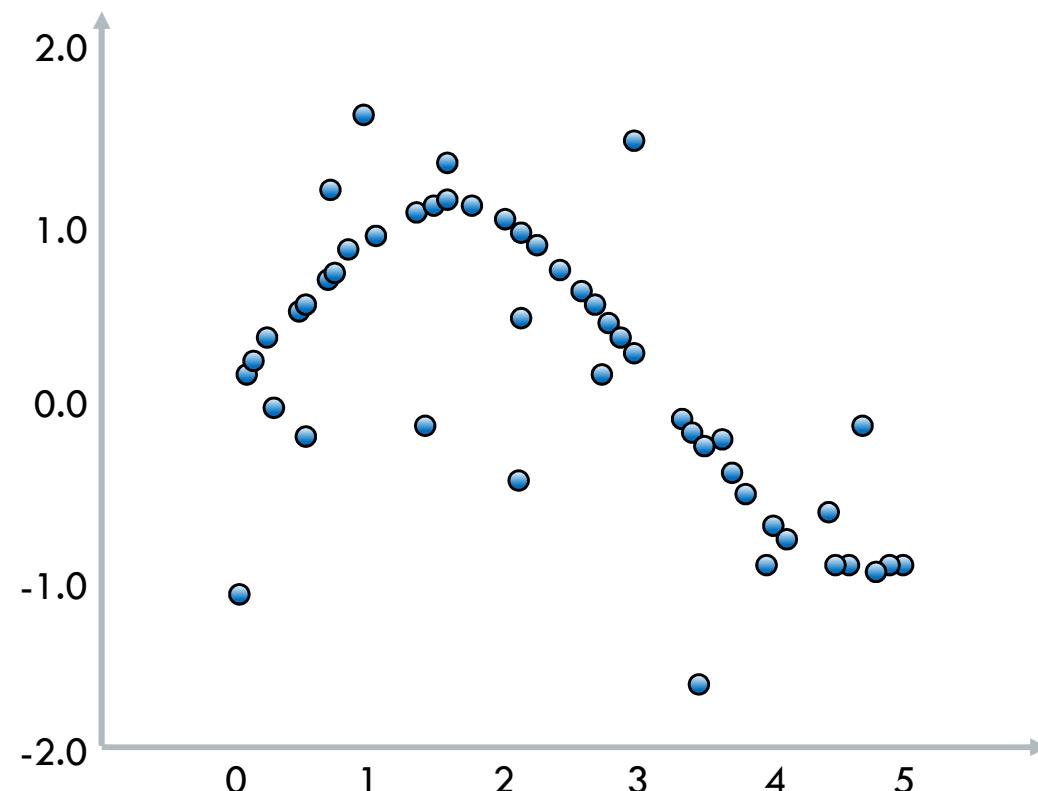


Regression Trees Predict Continuous Values

- Example: use slope and elevation in Himalayas
- Predict average precipitation (continuous value)
- Values at leaves are averages of members

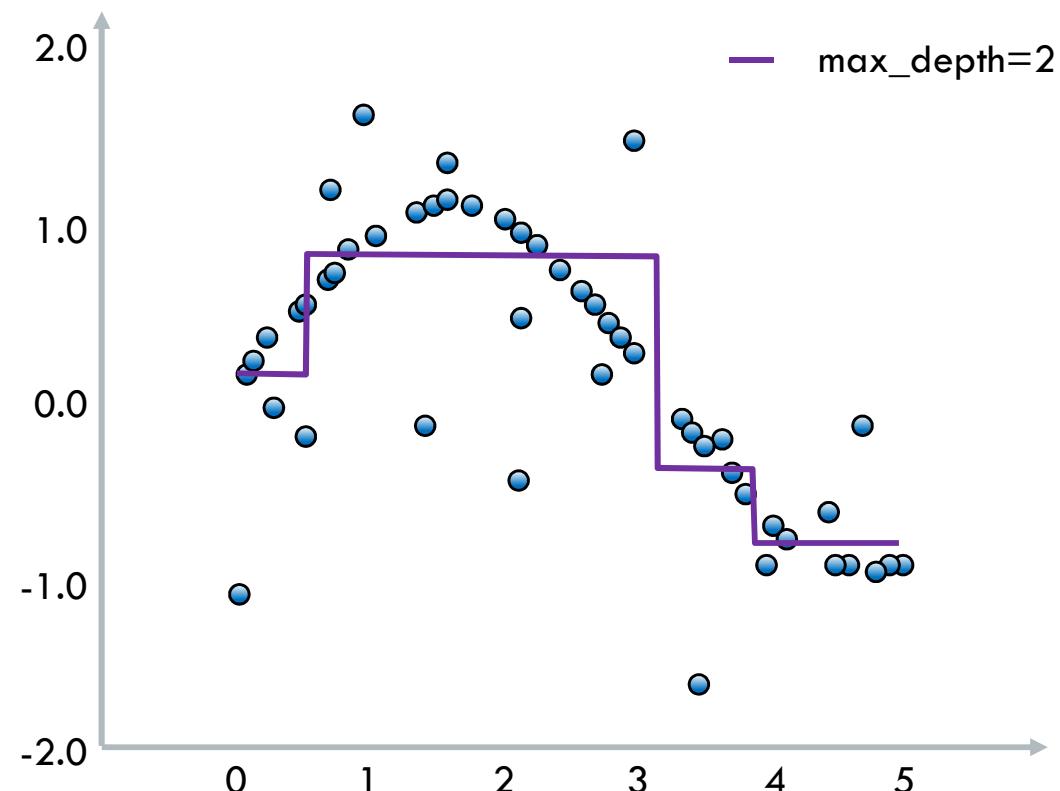


Regression Trees Predict Continuous Values



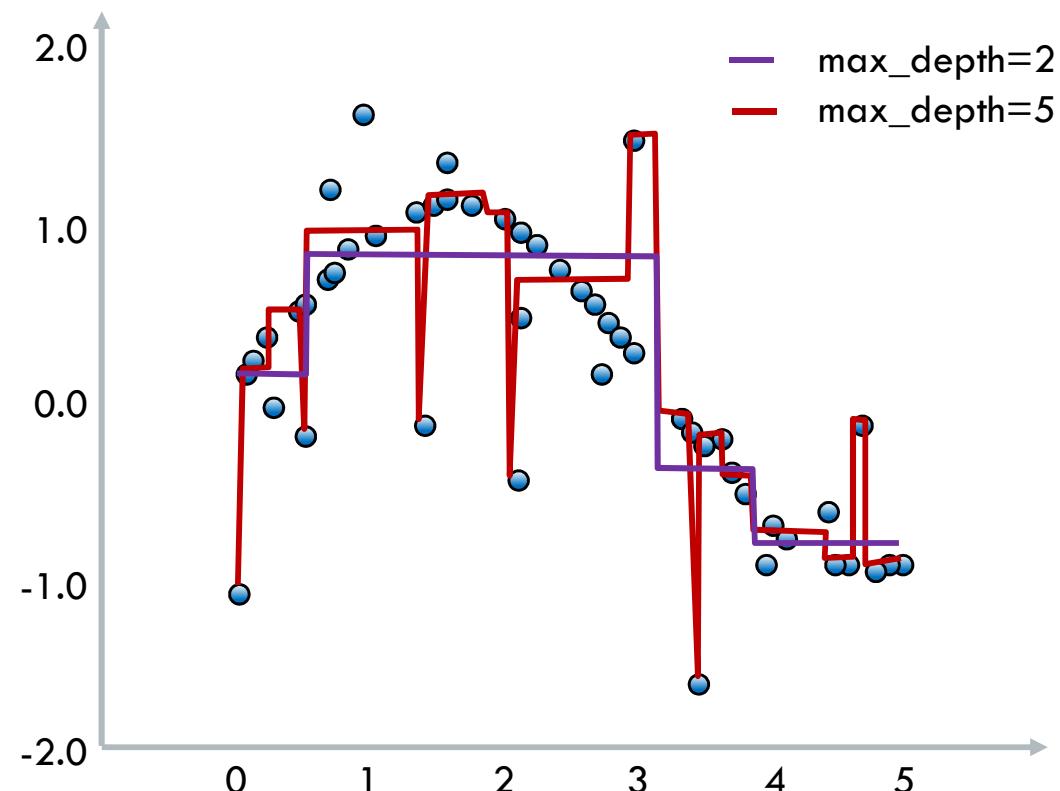
Source: http://scikit-learn.org/stable/auto_examples/tree/plot_tree_regression.html

Regression Trees Predict Continuous Values



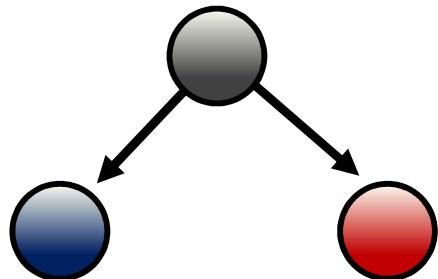
Source: http://scikit-learn.org/stable/auto_examples/tree/plot_tree_regression.html

Regression Trees Predict Continuous Values



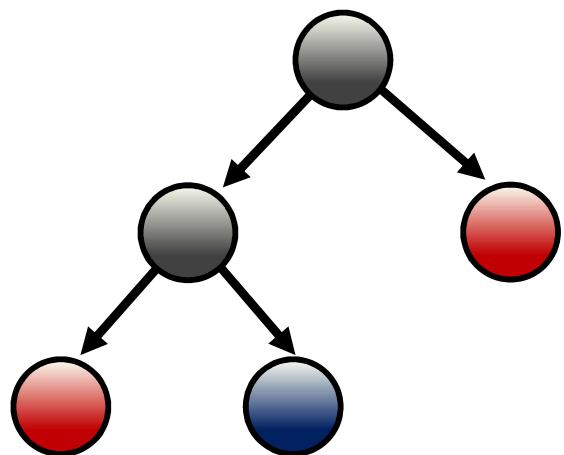
Source: http://scikit-learn.org/stable/auto_examples/tree/plot_tree_regression.html

Building a Decision Tree



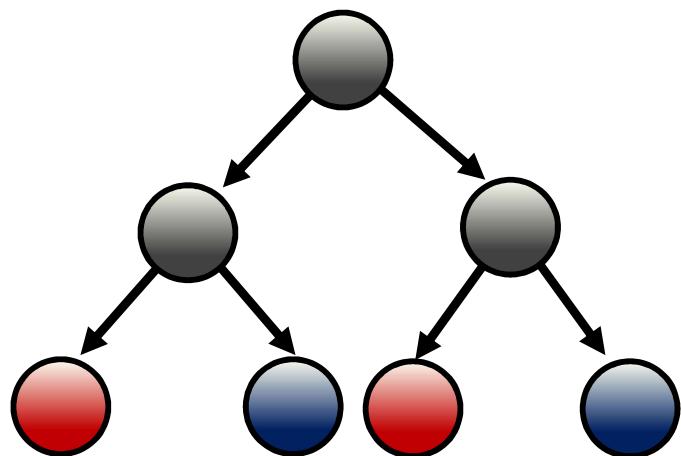
- Select a feature and split data into binary tree

Building a Decision Tree



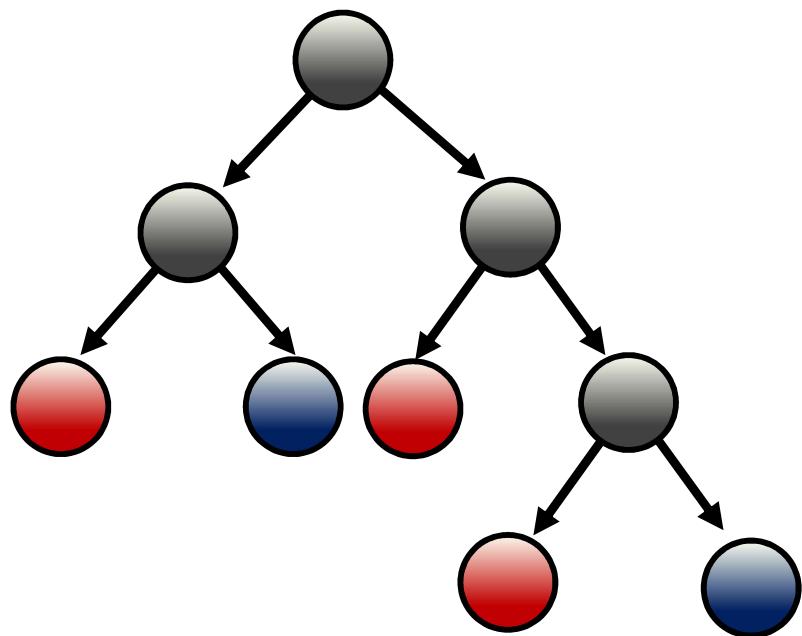
- Select a feature and split data into binary tree
- Continue splitting with available features

Building a Decision Tree



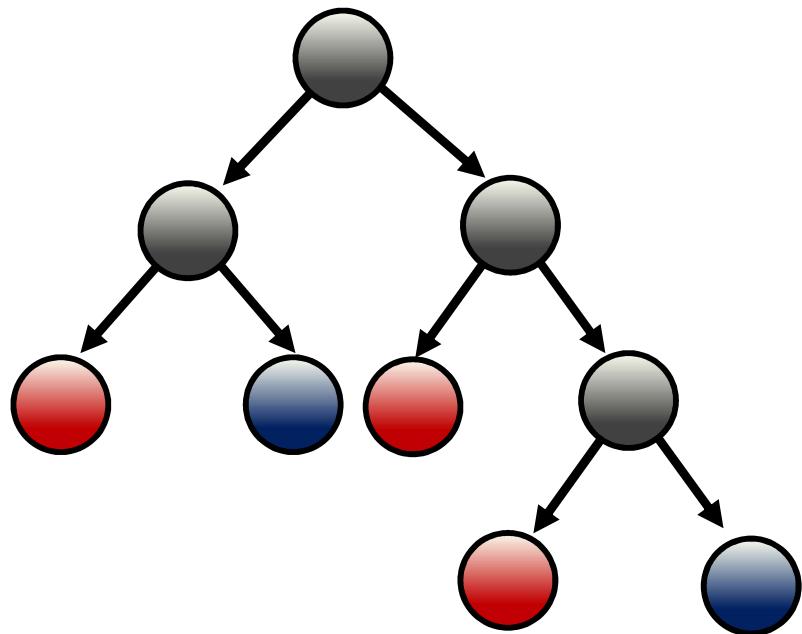
- Select a feature and split data into binary tree
- Continue splitting with available features

How Long to Keep Splitting?



ie

How Long to Keep Splitting?

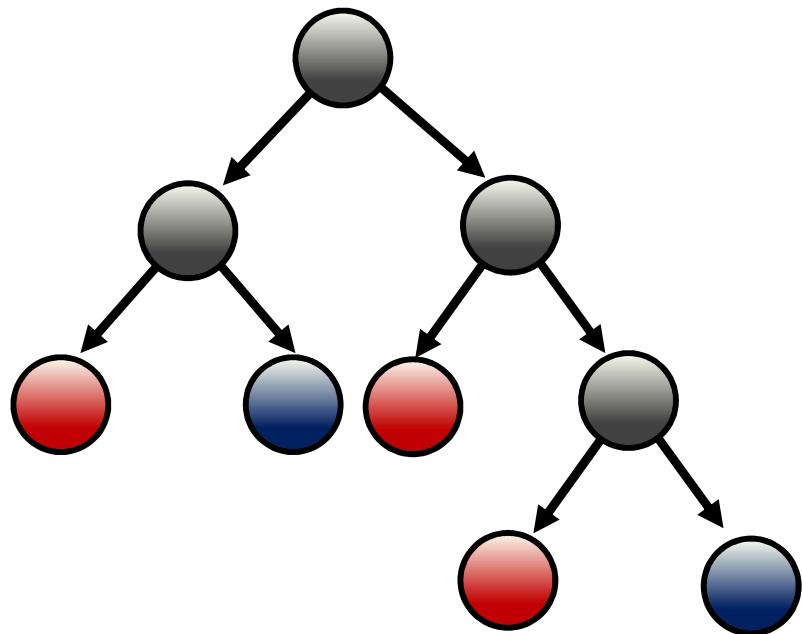


Until:

- Leaf node(s) are pure (only one class remains)

|

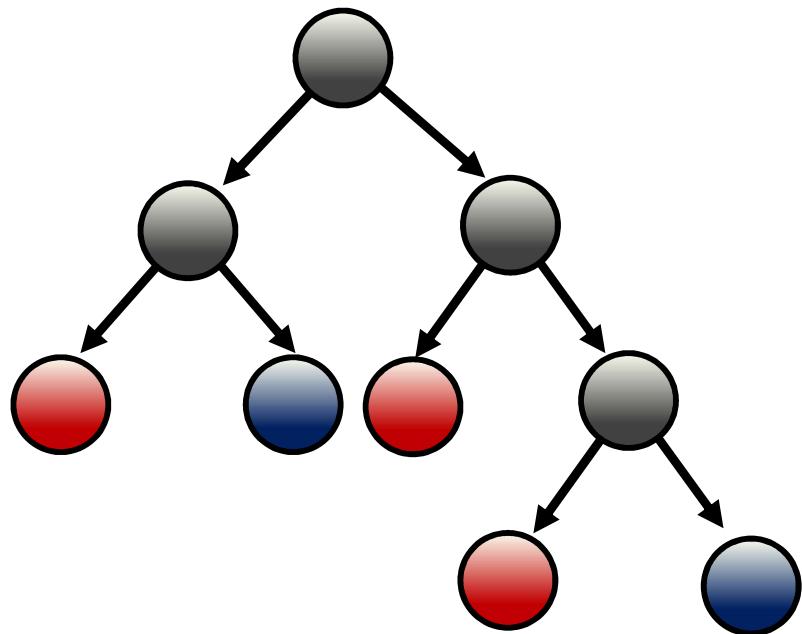
How Long to Keep Splitting?



Until:

- Leaf node(s) are pure (only one class remains)
- A maximum depth is reached

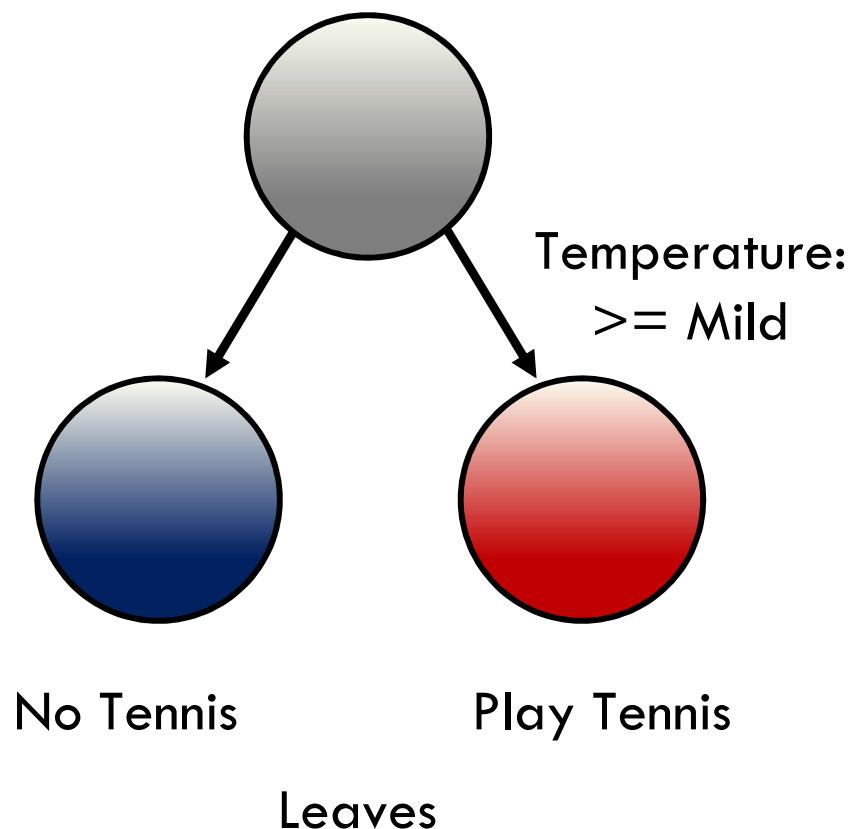
How Long to Keep Splitting?



Until:

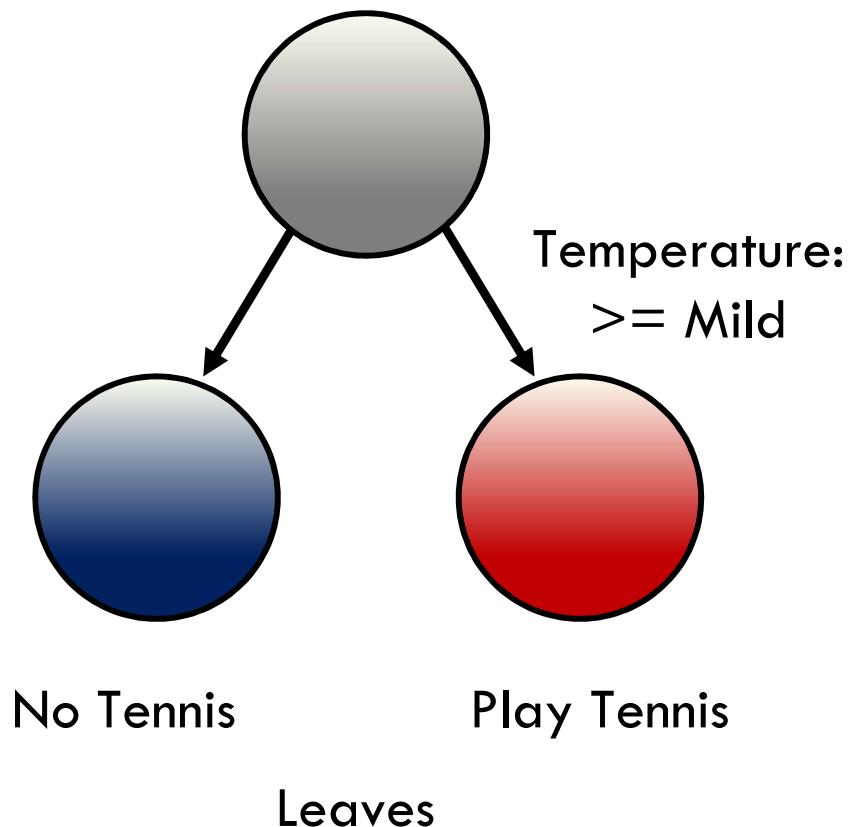
- Leaf node(s) are pure—only one class remains
- A maximum depth is reached
- A performance metric is achieved

Building the Best Decision Tree



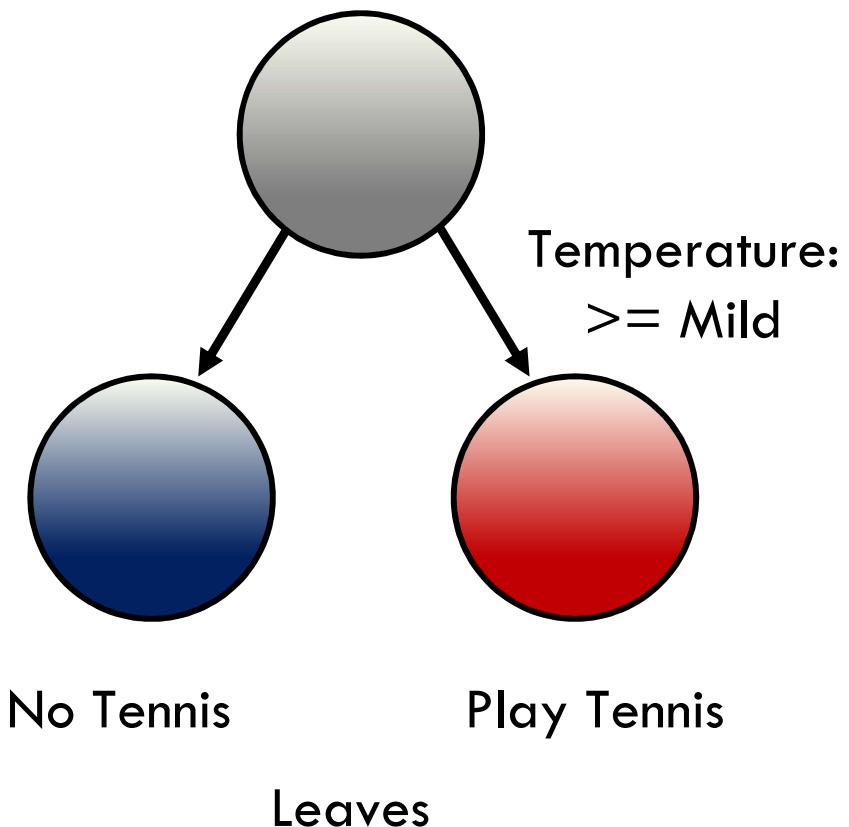
- Use greedy search: find the best split at each step

Building the Best Decision Tree



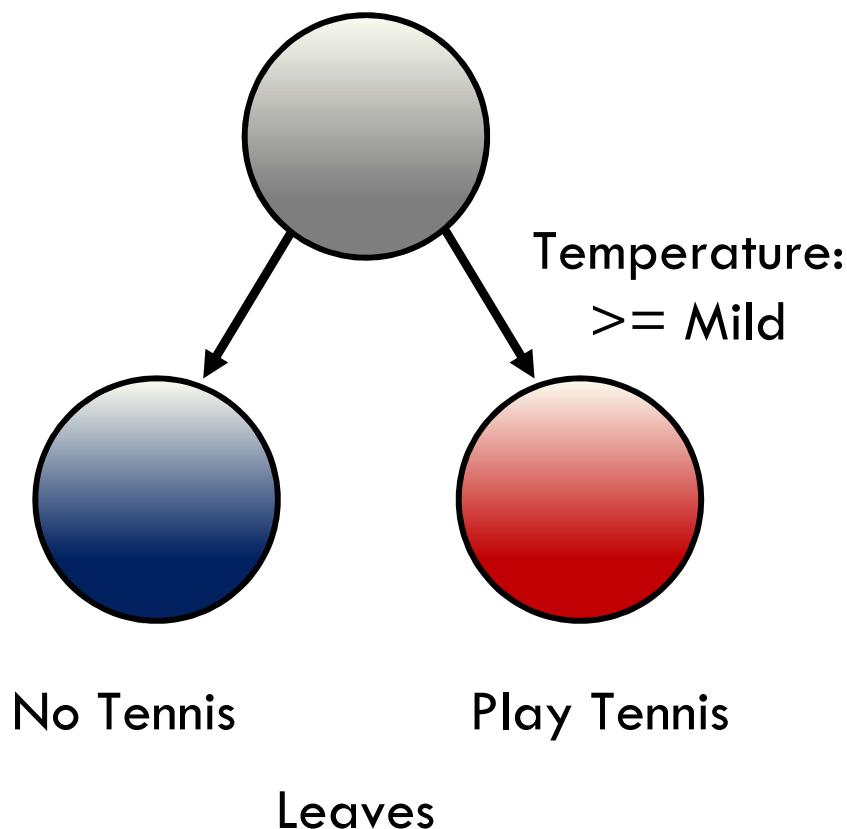
- Use greedy search: find the best split at each step
- What defines the best split?

Building the Best Decision Tree



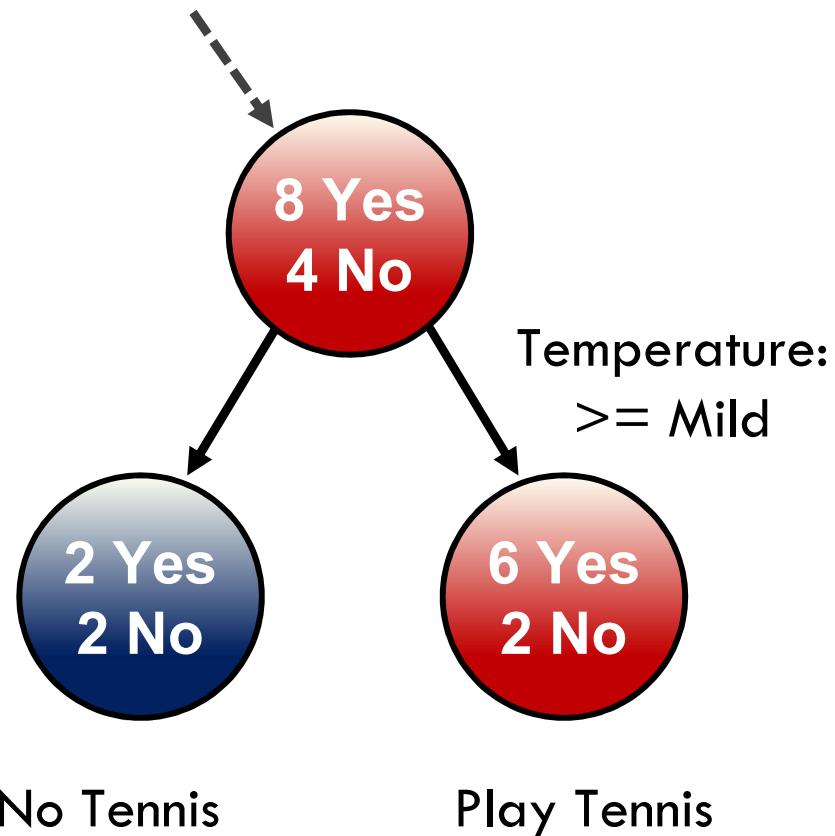
- Use greedy search: find the best split at each step
- What defines the best split?
- One that maximizes the information gained from the split

Building the Best Decision Tree



- Use greedy search: find the best split at each step
- What defines the best split?
- One that maximizes the information gained from the split
- How is information gain defined?

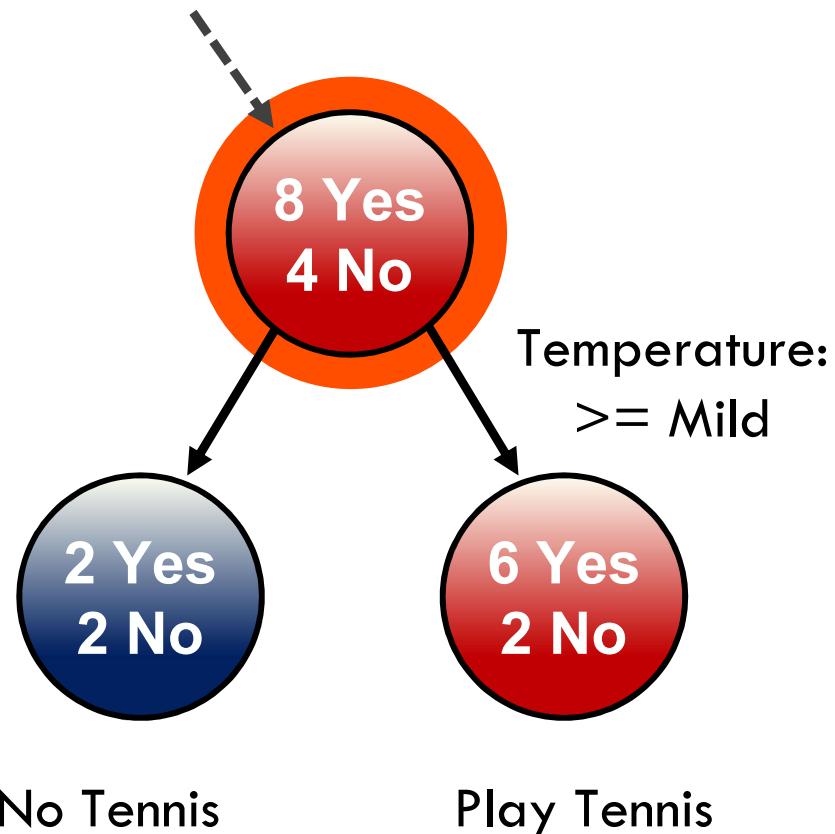
Splitting Based on Classification Error



Classification Error Equation

$$E(t) = 1 - \max_i[p(i|t)]$$

Splitting Based on Classification Error



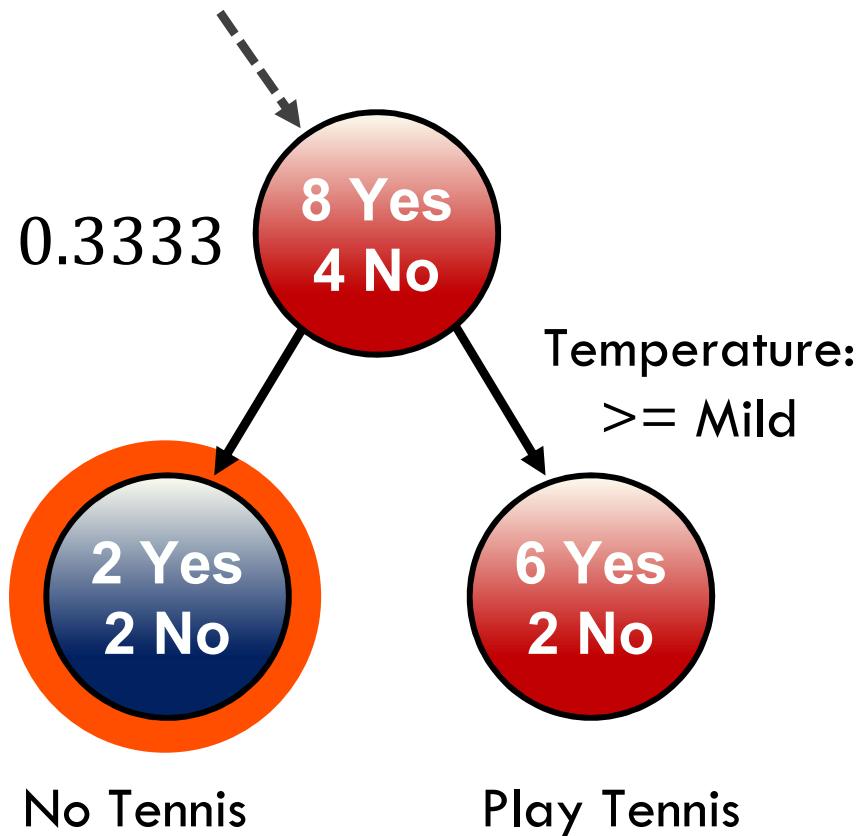
Classification Error Equation

$$E(t) = 1 - \max_i[p(i|t)]$$

Classification Error Before

$$1 - 8/12 = 0.3333$$

Splitting Based on Classification Error



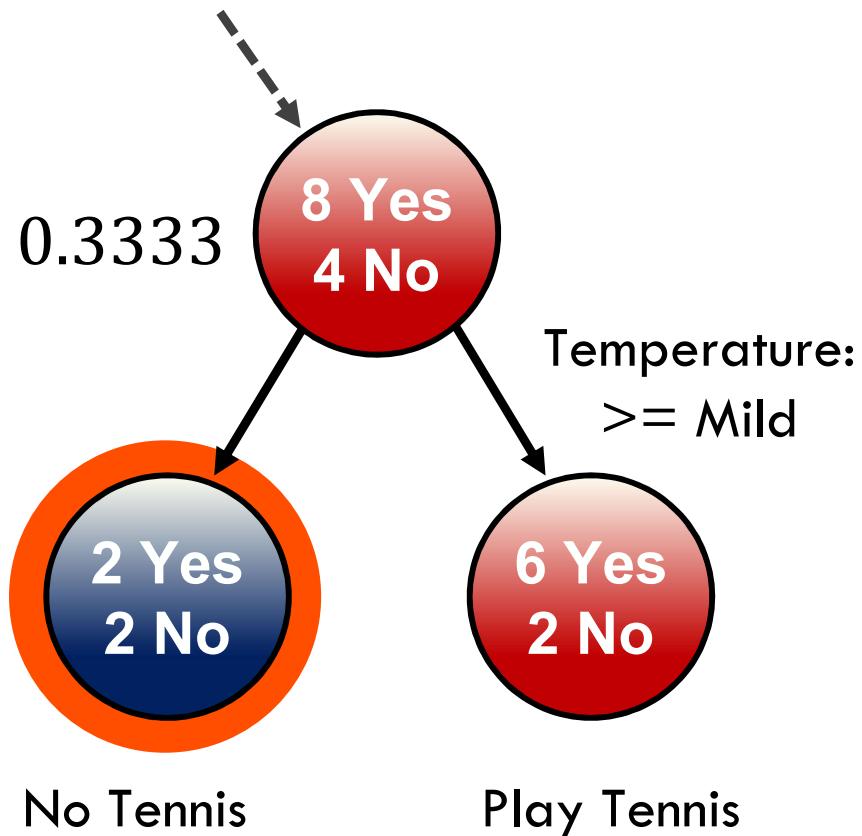
Classification Error Equation

$$E(t) = 1 - \max_i[p(i|t)]$$

Classification Error Left Side

$$1 - \frac{2}{4} = 0.5000$$

Splitting Based on Classification Error



Classification Error Equation

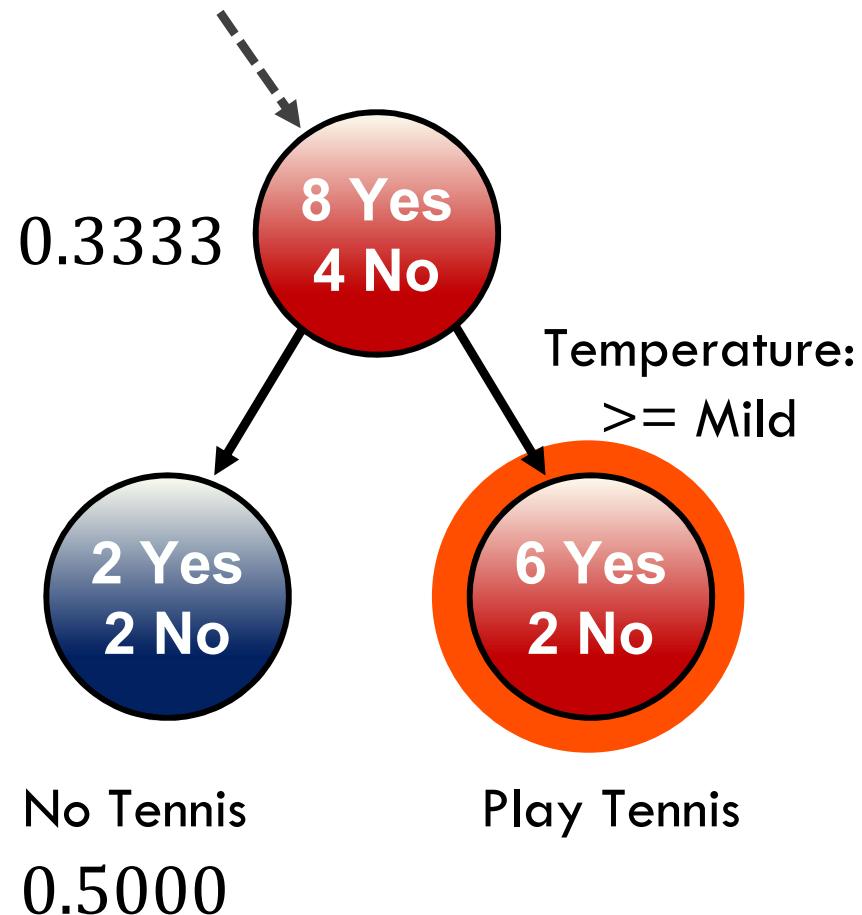
$$E(t) = 1 - \max_i[p(i|t)]$$

Classification Error Left Side

$$1 - \frac{2}{4} = 0.5000$$

Information lost on
small # of data points

Splitting Based on Classification Error



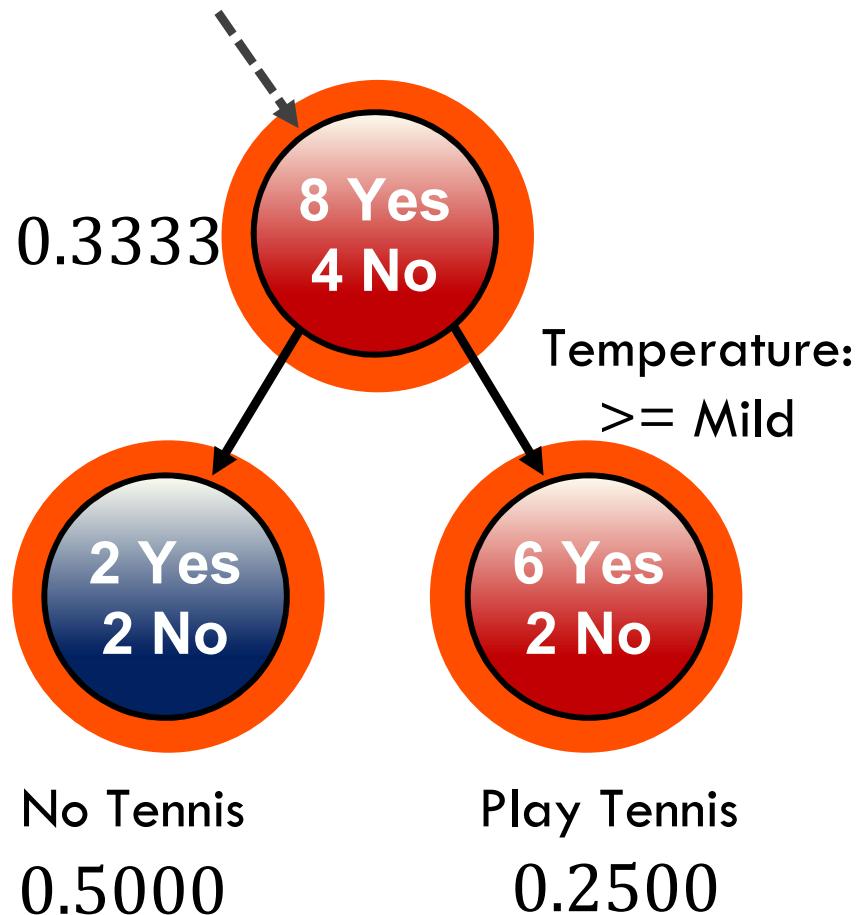
Classification Error Equation

$$E(t) = 1 - \max_i[p(i|t)]$$

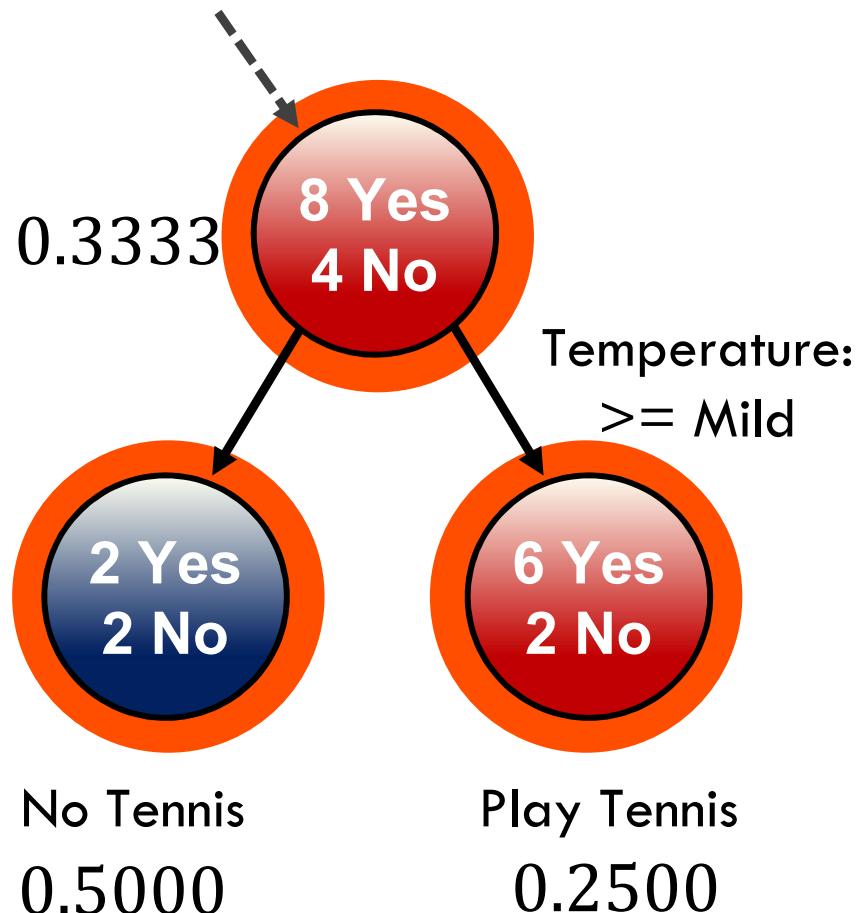
Classification Error Right Side

$$1 - \frac{6}{8} = 0.2500$$

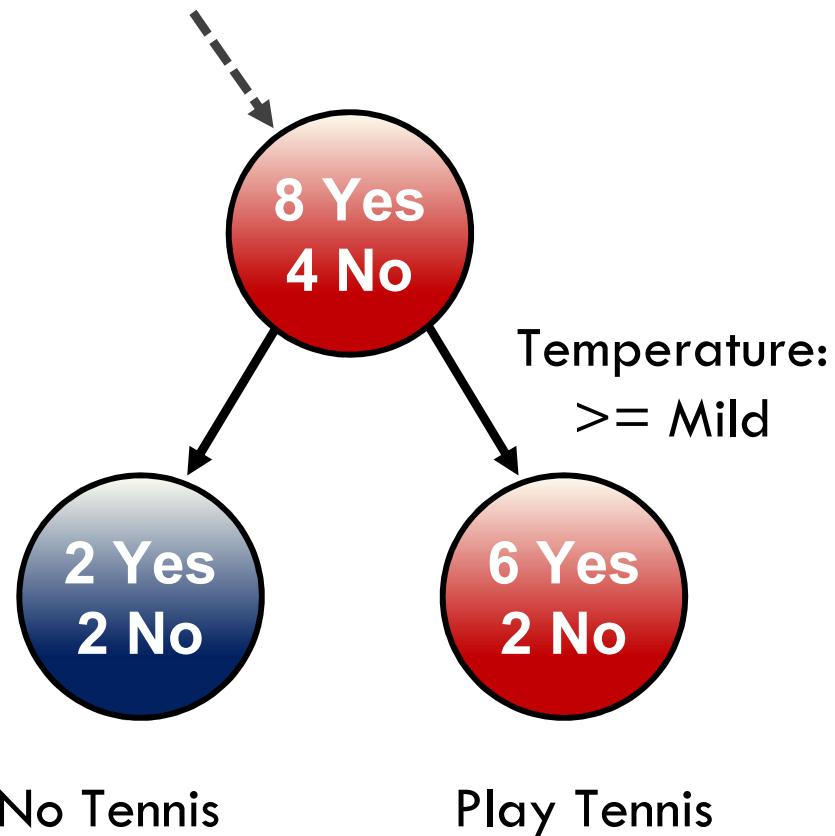
Splitting Based on Classification Error



Splitting Based on Classification Error

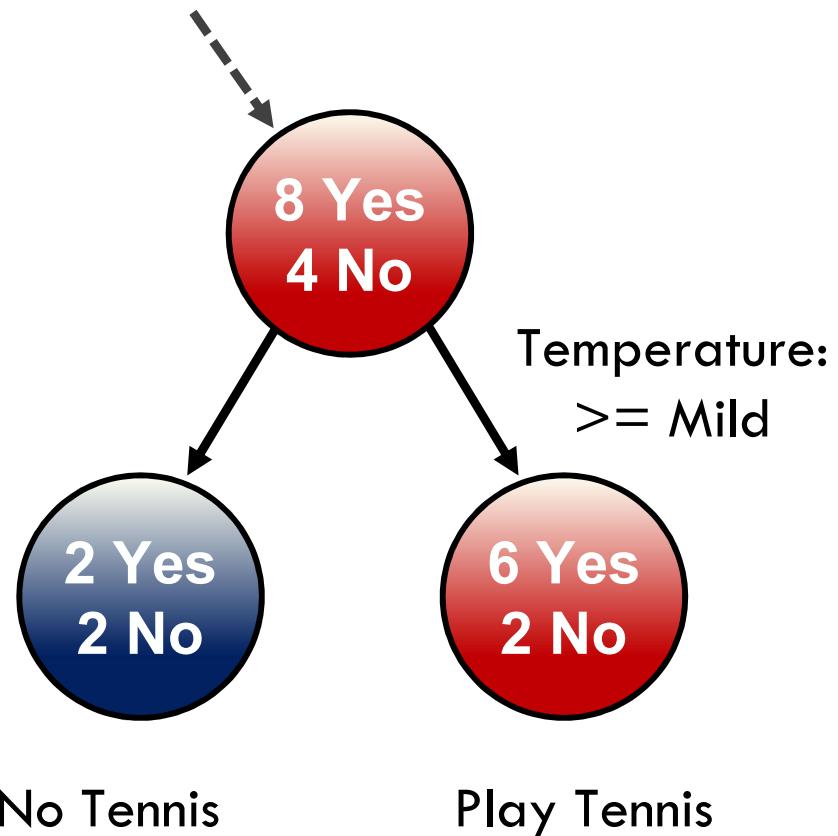


Splitting Based on Classification Error



- Using classification error, no further splits would occur
- Problem: end nodes are not homogeneous
- Try a different metric?

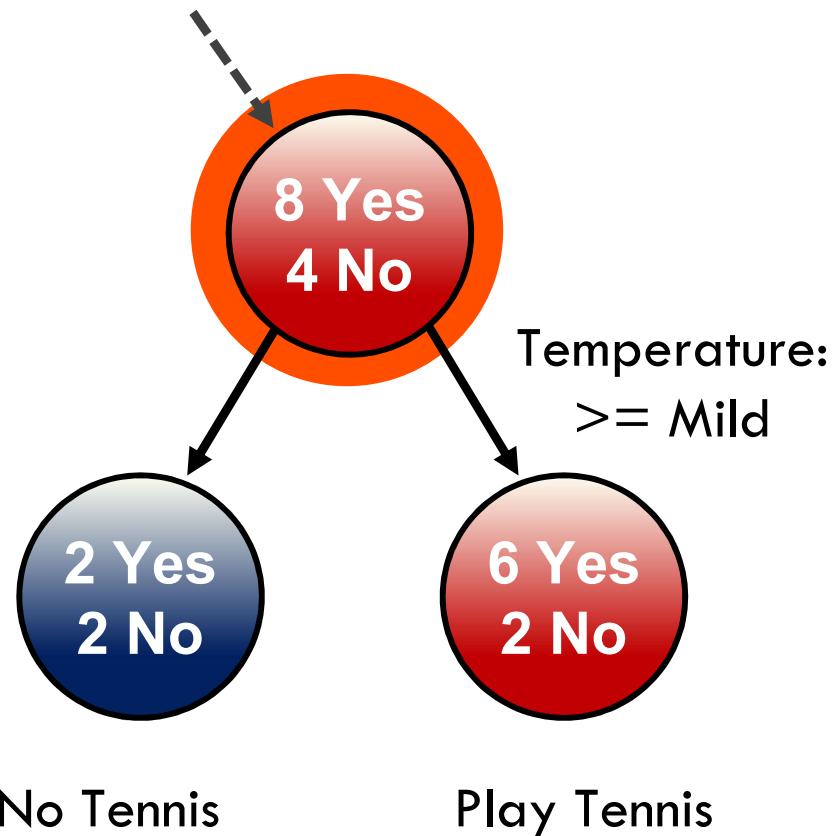
Splitting Based on Entropy



Entropy Equation

$$H(t) = - \sum_{i=1}^n p(i|t) \log_2 [p(i|t)]$$

Splitting Based on Entropy



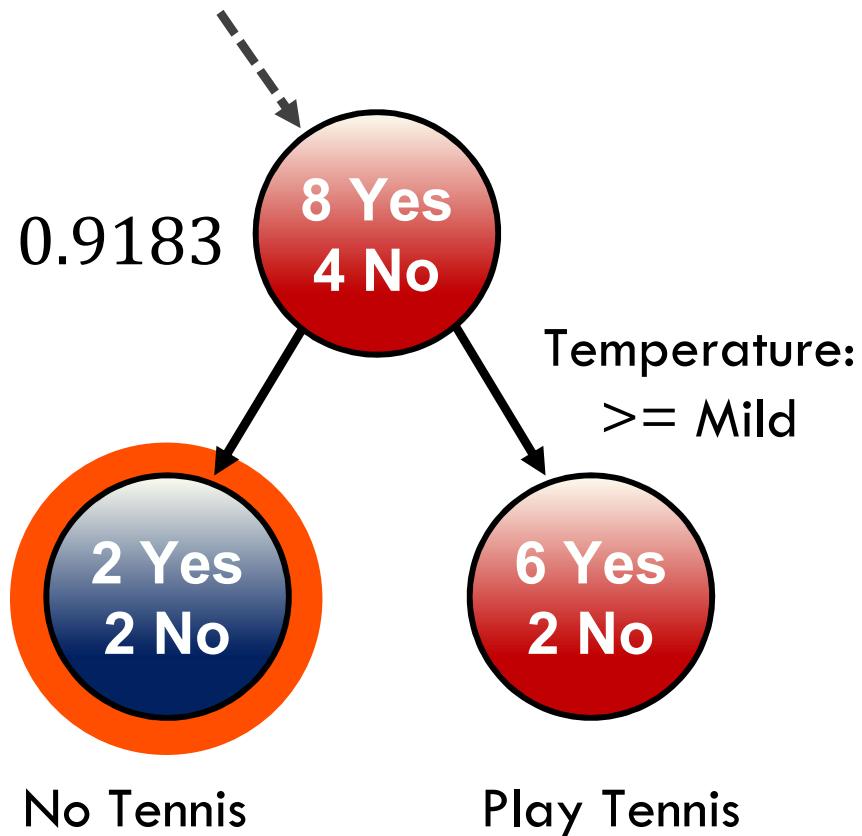
Entropy Equation

$$H(t) = - \sum_{i=1}^n p(i|t) \log_2[p(i|t)]$$

Entropy Before

$$-\frac{8}{12} \log_2(\frac{8}{12}) - \frac{4}{12} \log_2(\frac{4}{12}) = 0.9183$$

Splitting Based on Entropy



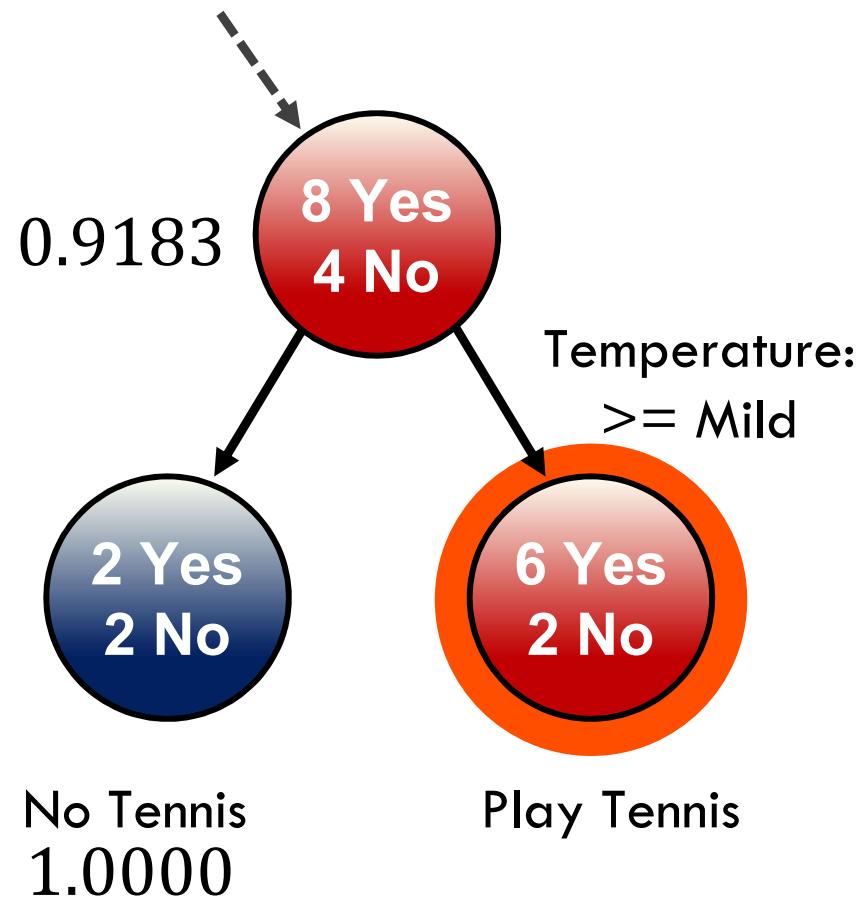
Entropy Equation

$$H(t) = - \sum_{i=1}^n p(i|t) \log_2[p(i|t)]$$

Entropy Left Side

$$-\frac{2}{4} \log_2(\frac{2}{4}) - \frac{2}{4} \log_2(\frac{2}{4}) = 1.0000$$

Splitting Based on Entropy



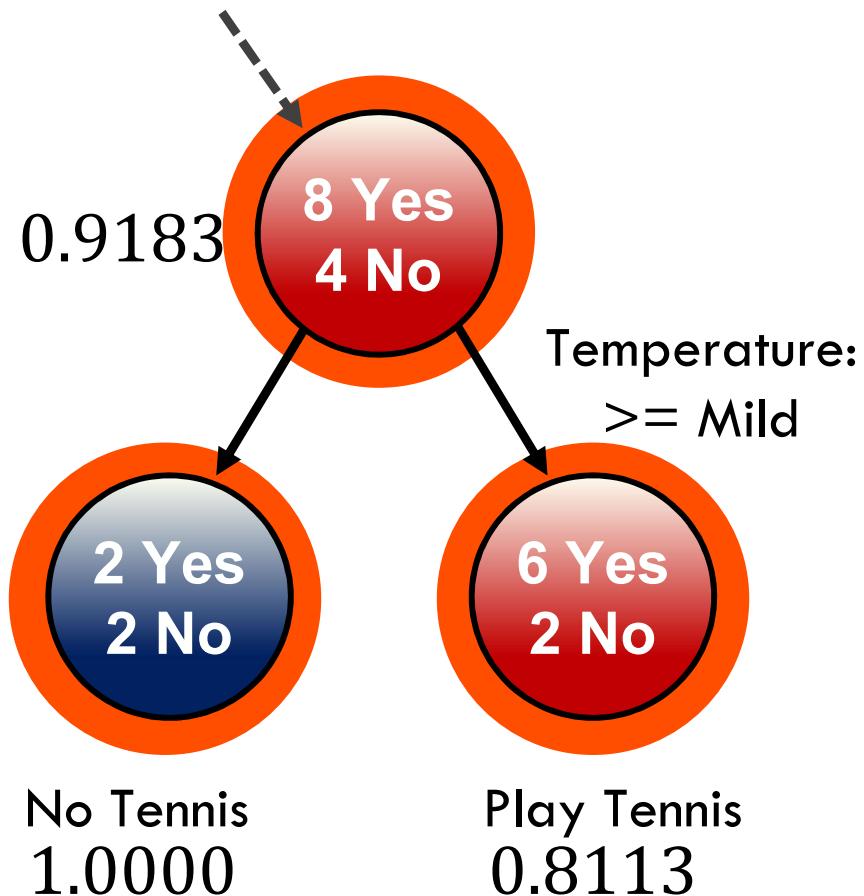
Entropy Equation

$$H(t) = - \sum_{i=1}^n p(i|t) \log_2[p(i|t)]$$

Entropy Right Side

$$-\frac{6}{8} \log_2(\frac{6}{8}) - \frac{2}{8} \log_2(\frac{2}{8}) = 0.8113$$

Splitting Based on Entropy



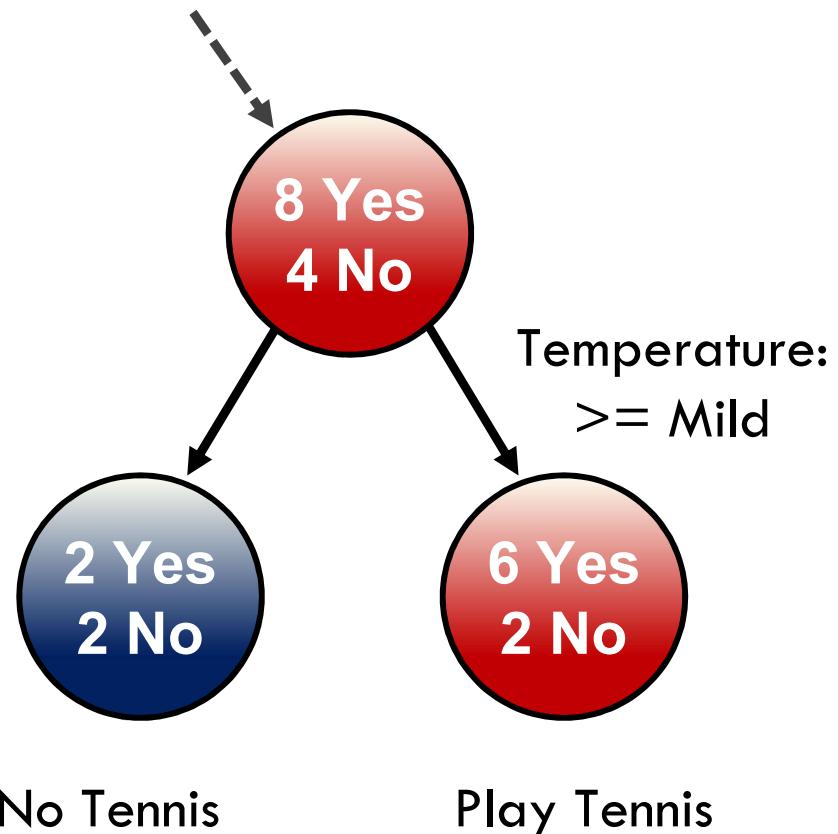
Entropy Equation

$$H(t) = - \sum_{i=1}^n p(i|t) \log_2[p(i|t)]$$

Entropy Change

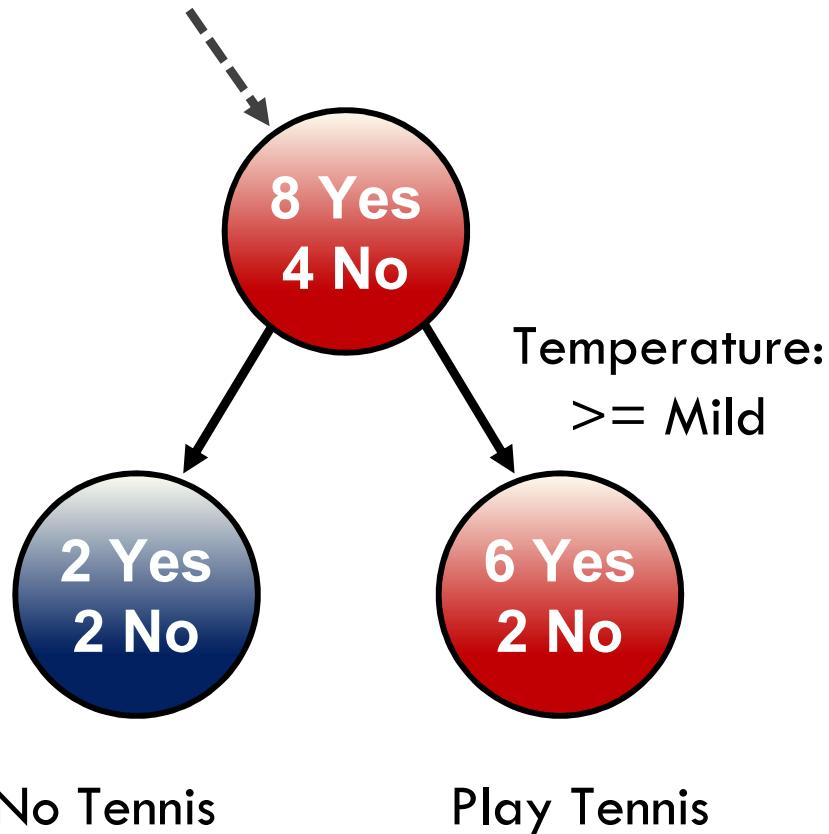
$$\begin{aligned} 0.9183 - & \frac{4}{12} * 1.0000 - \frac{8}{12} * 0.8113 \\ & = 0.0441 \end{aligned}$$

Splitting Based on Entropy



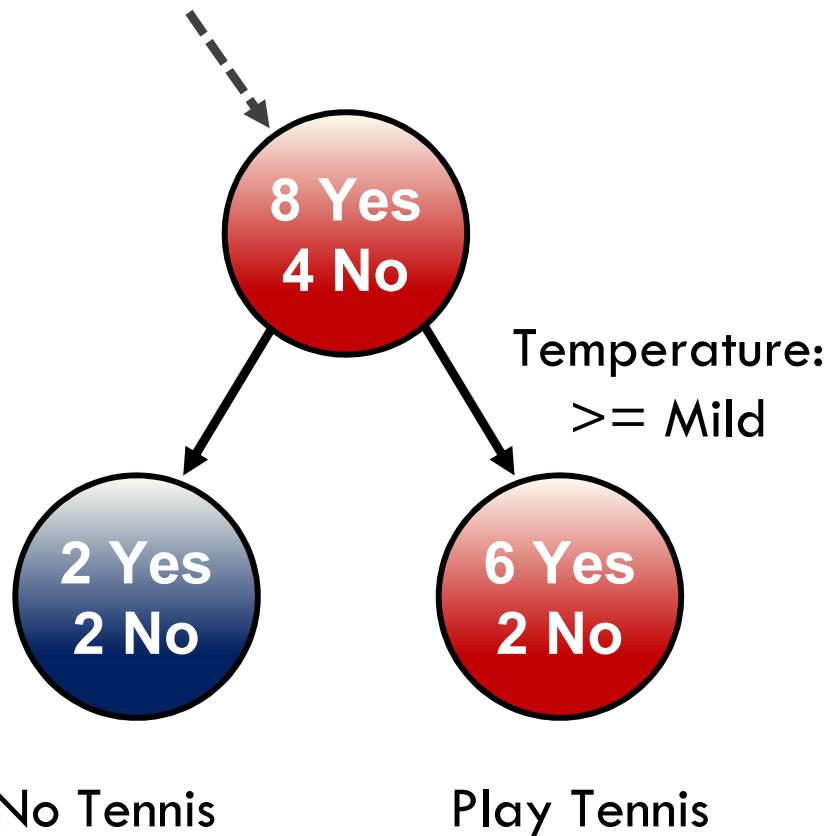
- Splitting based on entropy allows further splits to occur

Splitting Based on Entropy



- Splitting based on entropy allows further splits to occur
- Can eventually reach goal of homogeneous nodes

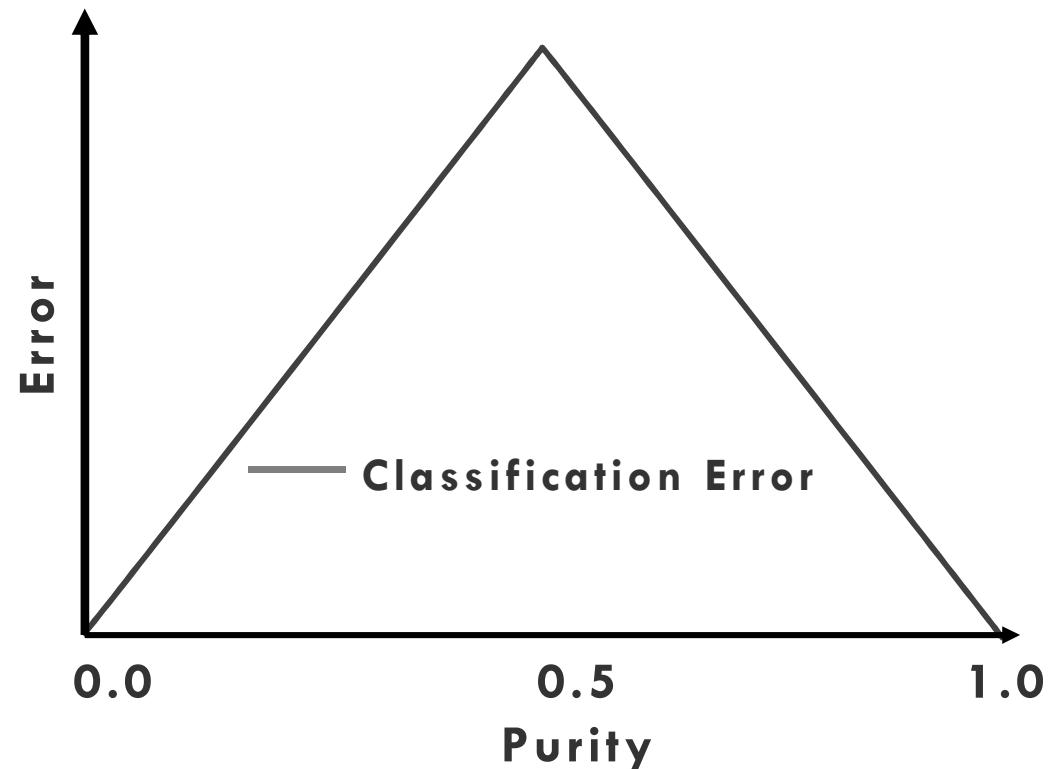
Splitting Based on Entropy



- Splitting based on entropy allows further splits to occur
- Can eventually reach goal of homogeneous nodes
- Why does this work with entropy but not classification error?

Classification Error vs Entropy

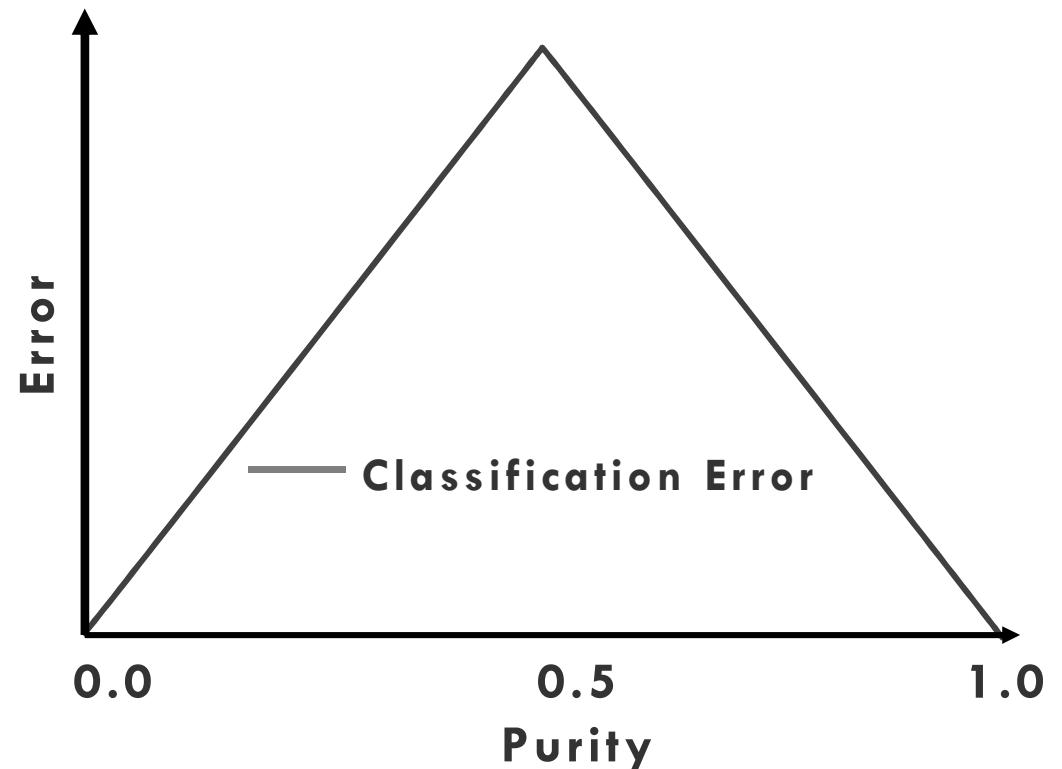
- Classification error is a flat function with maximum at center
- Center represents ambiguity—



$$E(t) = 1 - \max_i[p(i|t)]$$

Classification Error vs Entropy

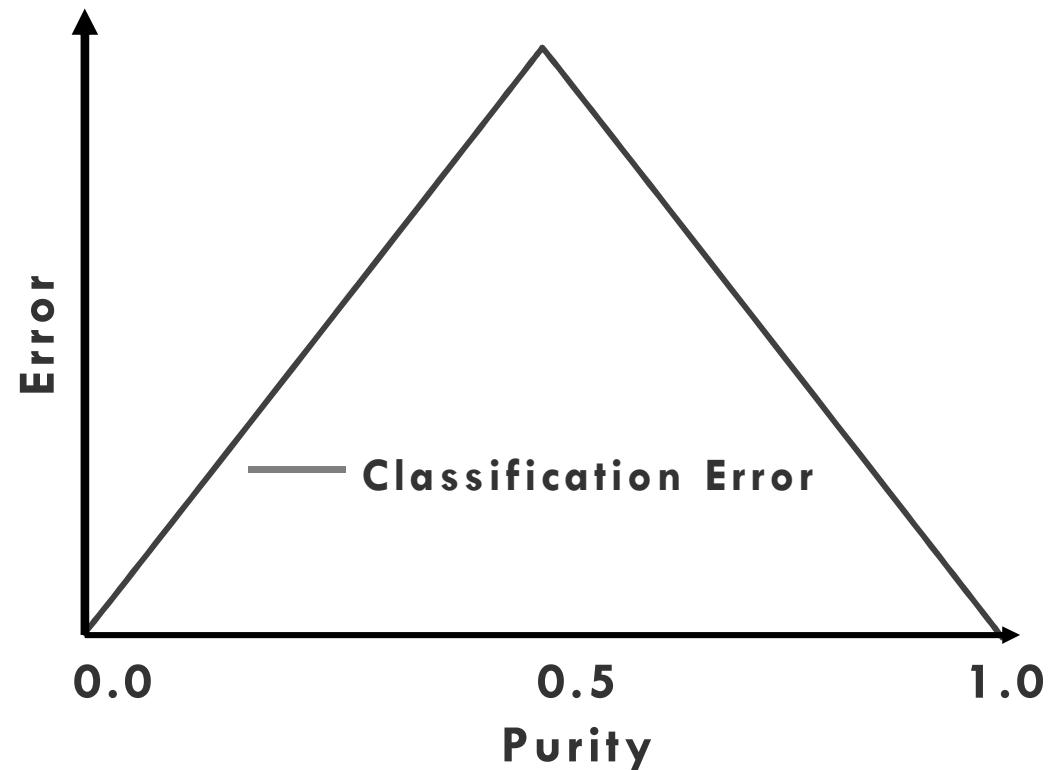
- Classification error is a flat function with maximum at center
- Center represents ambiguity—50/50 split
- Splitting metrics favor results that



$$E(t) = 1 - \max_i[p(i|t)]$$

Classification Error vs Entropy

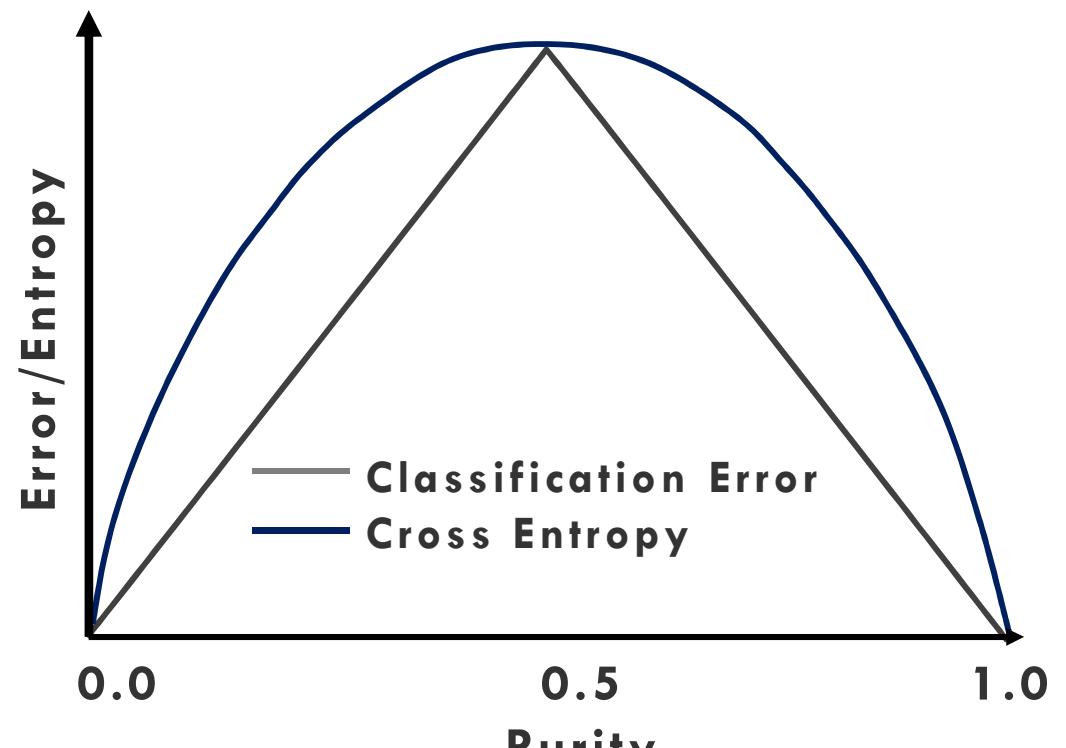
- Classification error is a flat function with maximum at center
- Center represents ambiguity—50/50 split
- Splitting metrics favor results that are furthest away from the center



$$E(t) = 1 - \max_i[p(i|t)]$$

Classification Error vs Entropy

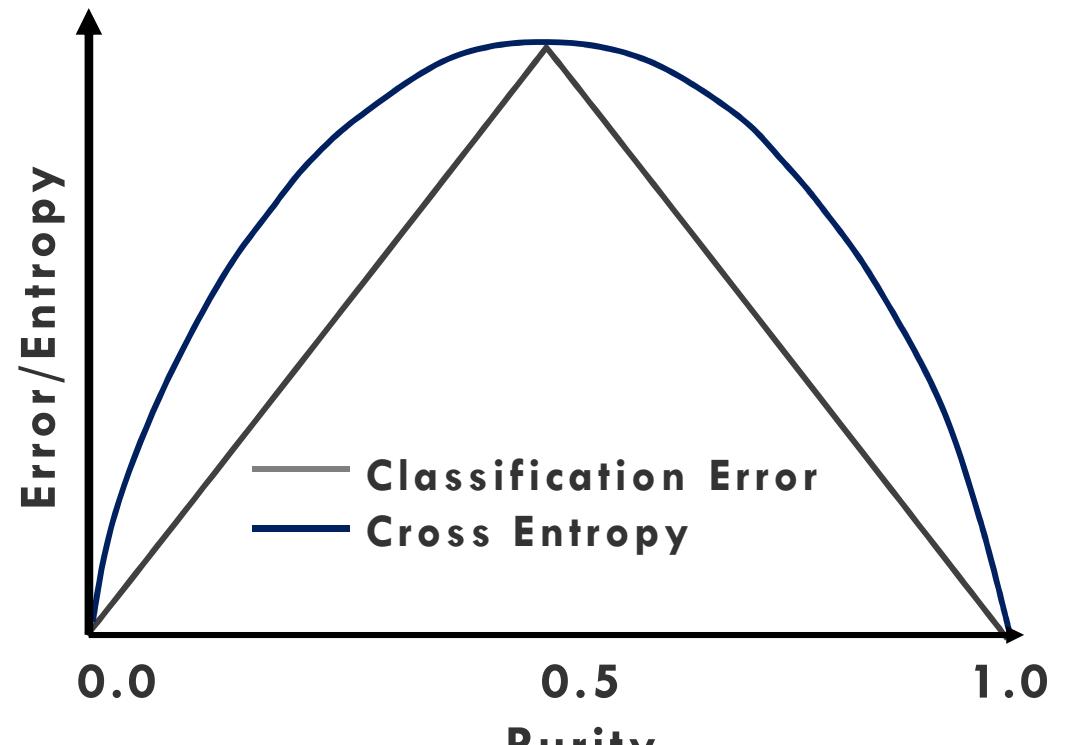
- Entropy has the same maximum but is curved



$$H(t) = - \sum_{i=1}^n p(i|t) \log_2 [p(i|t)]$$

Classification Error vs Entropy

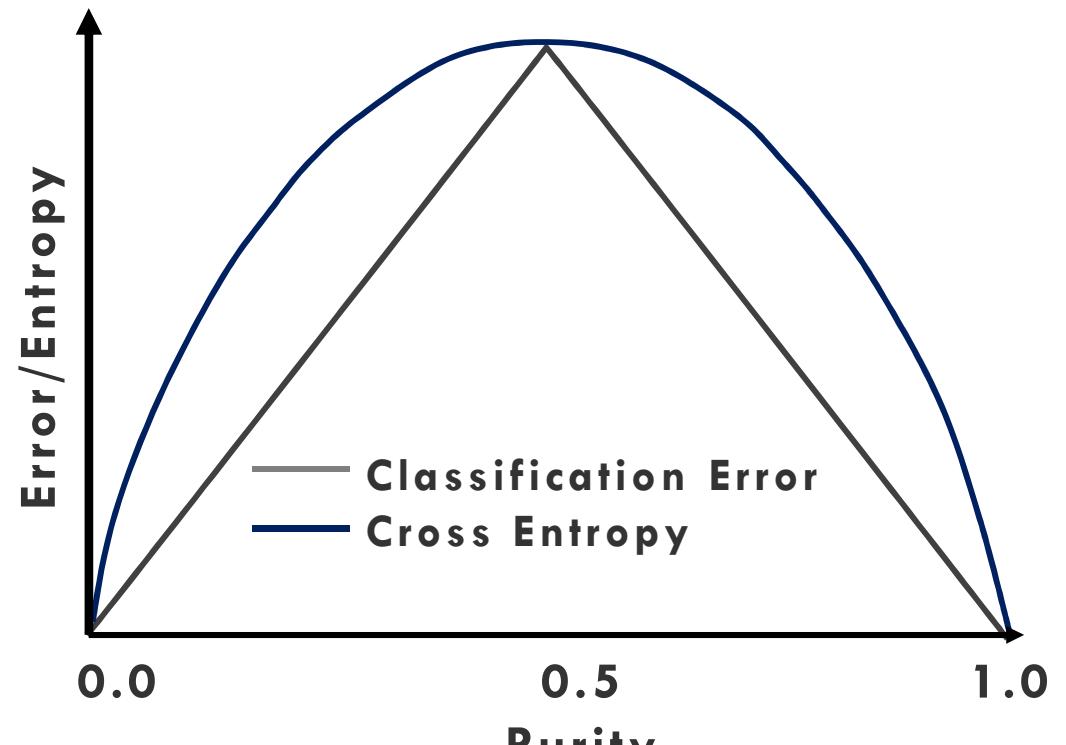
- Entropy has the same maximum but is curved
- Curvature allows splitting to continue until nodes are pure



$$H(t) = - \sum_{i=1}^n p(i|t) \log_2 [p(i|t)]$$

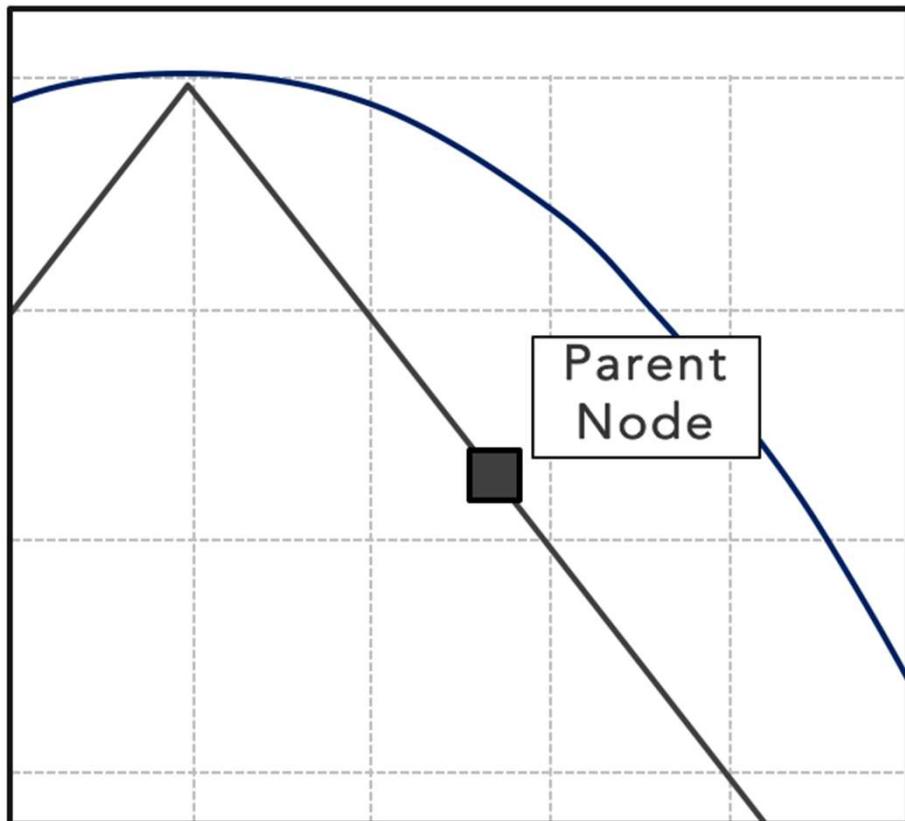
Classification Error vs Entropy

- Entropy has the same maximum but is curved
- Curvature allows splitting to continue until nodes are pure
- How does this work?



$$H(t) = - \sum_{i=1}^n p(i|t) \log_2 [p(i|t)]$$

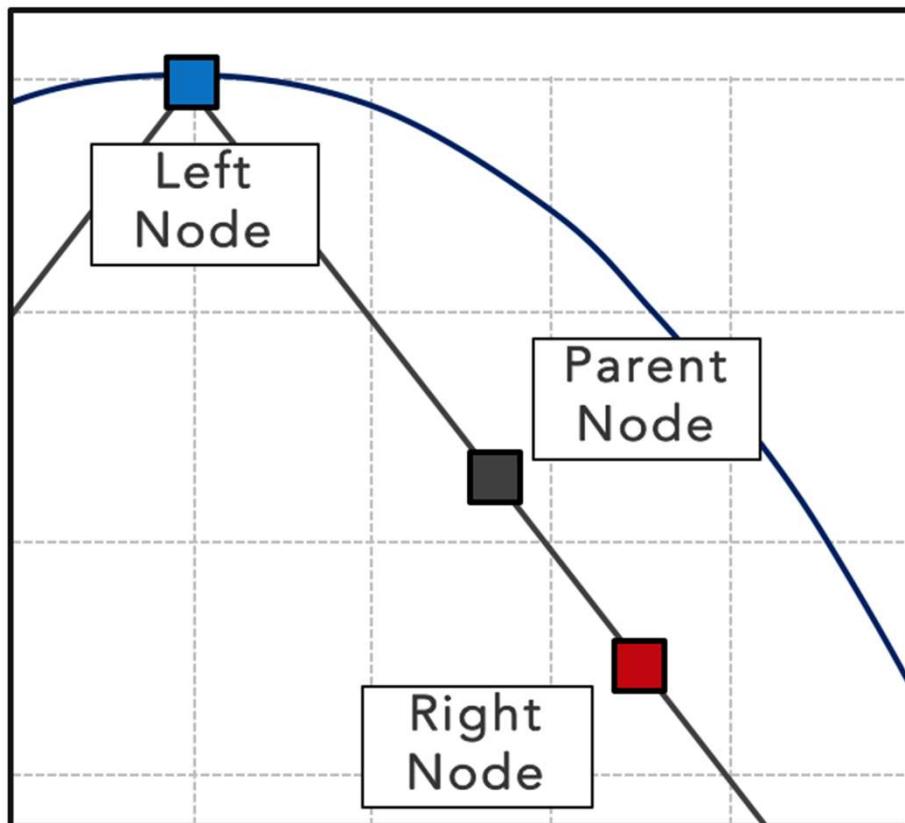
Information Gained by Splitting



- With classification error, the function is flat

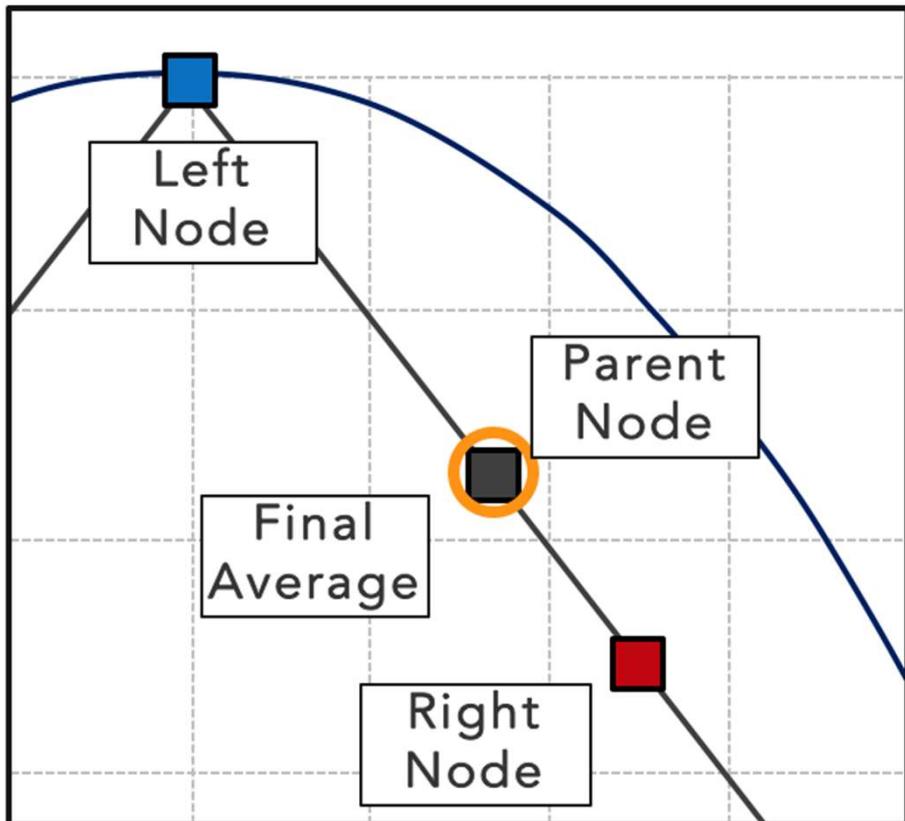
or

Information Gained by Splitting



- With classification error, the function is flat

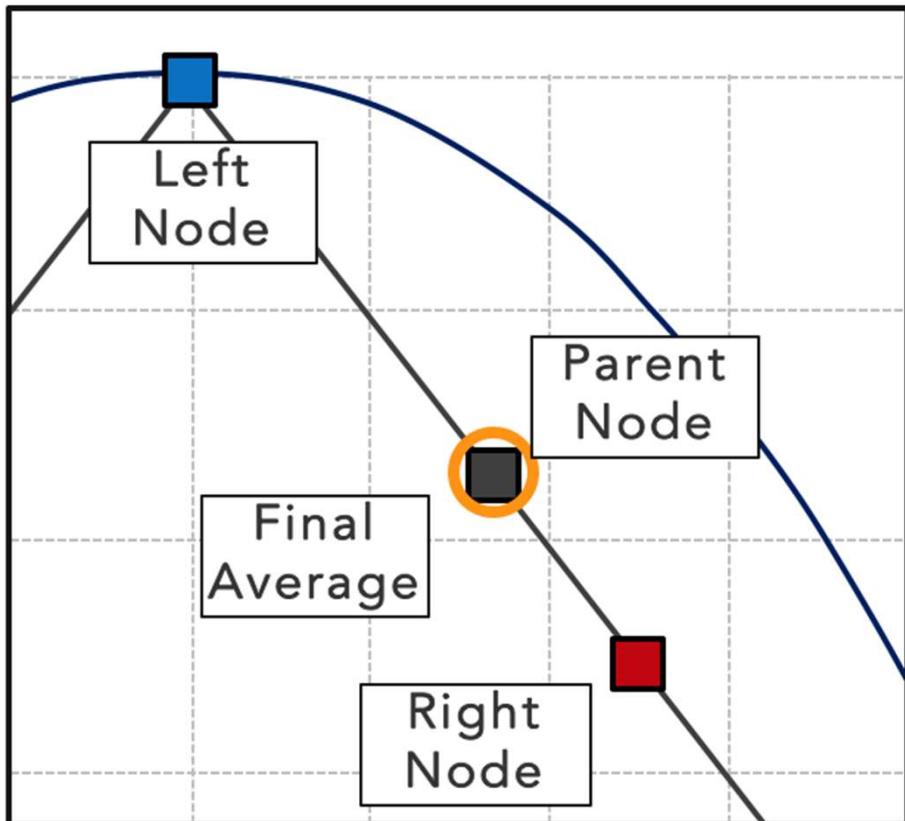
Information Gained by Splitting



- With classification error, the function is flat
- Final average classification error can be identical to parent

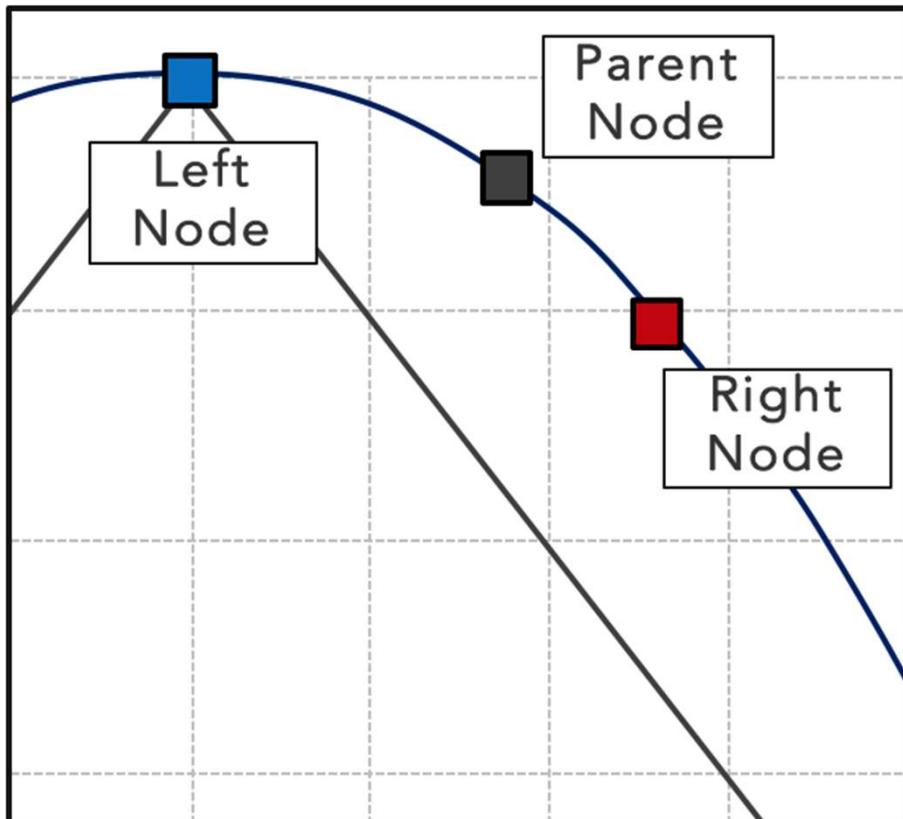
~ ~ ~ ~ ~

Information Gained by Splitting



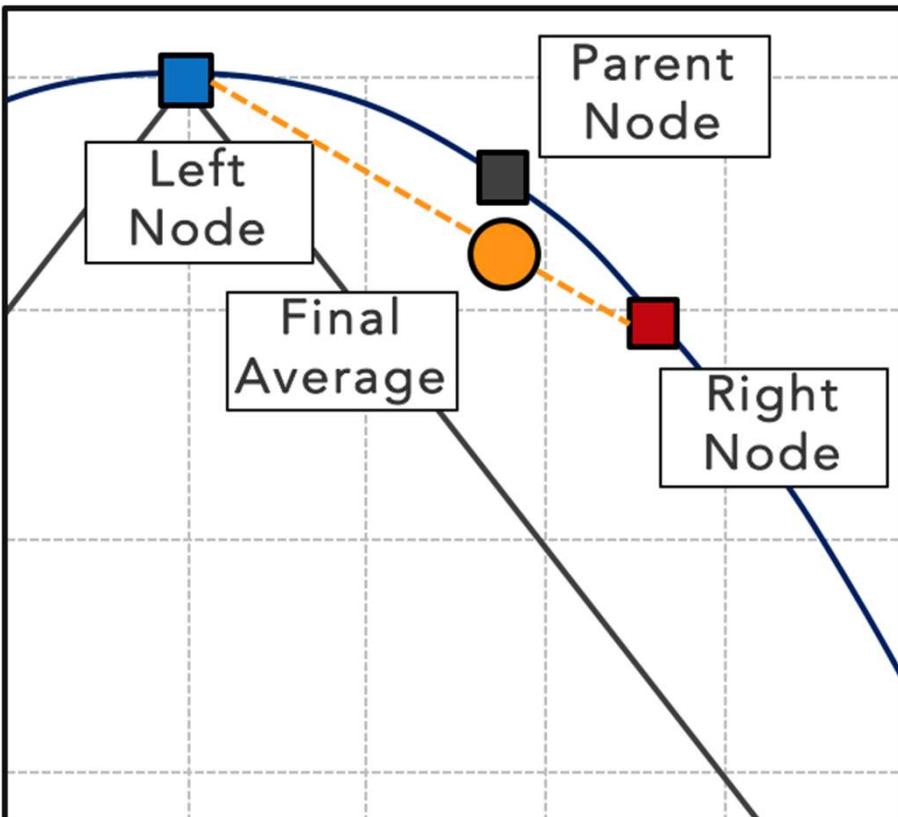
- With classification error, the function is flat
- Final average classification error can be identical to parent
- Resulting in premature stopping

Information Gained by Splitting



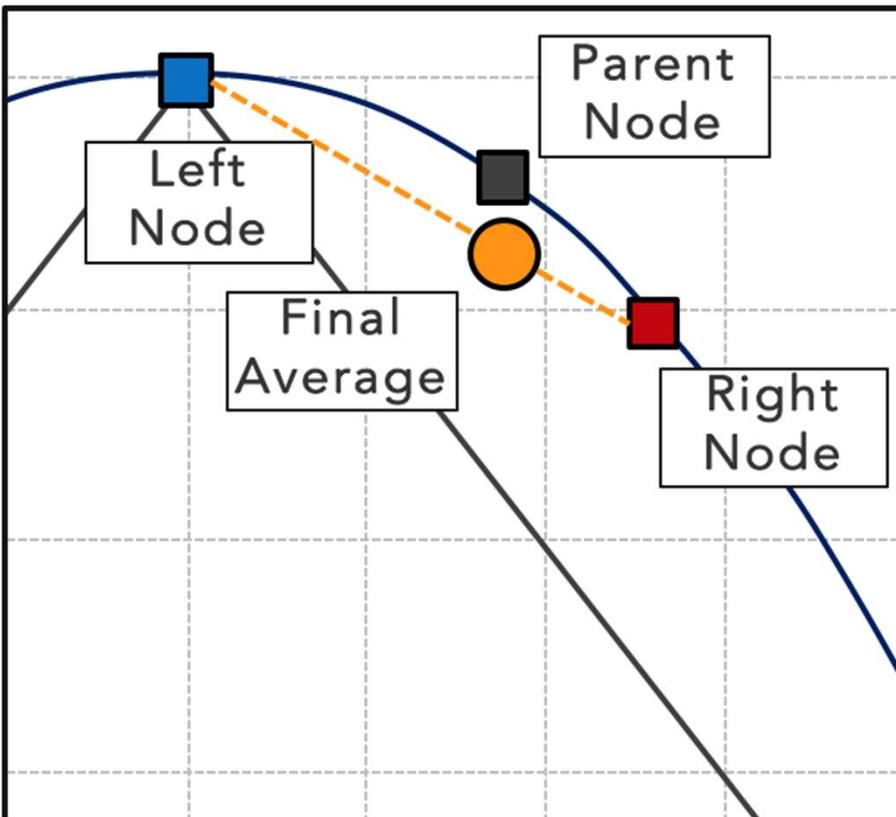
- With entropy gain, the function has a "bulge"

Information Gained by Splitting



- With entropy gain, the function has a "bulge"
- Allows average information of children to be less than parent
- Results in information gain and

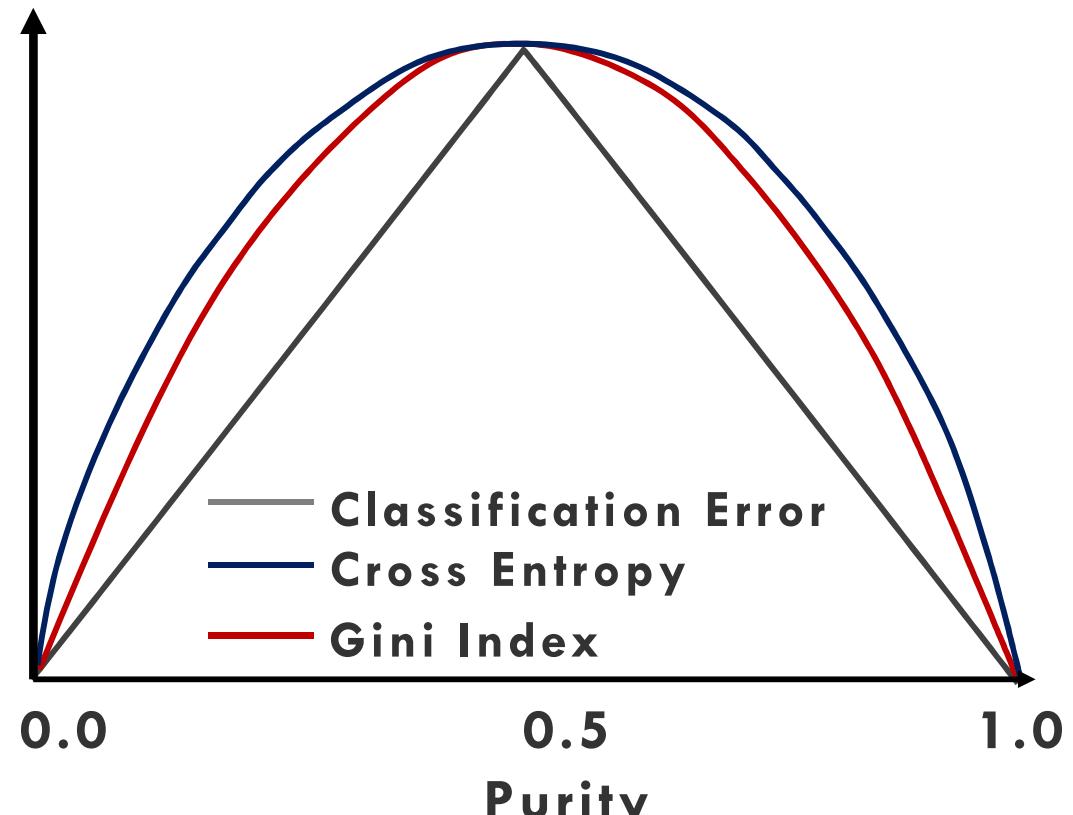
Information Gained by Splitting



- With entropy gain, the function has a "bulge"
- Allows average information of children to be less than parent
- Results in information gain and continued splitting

The Gini Index

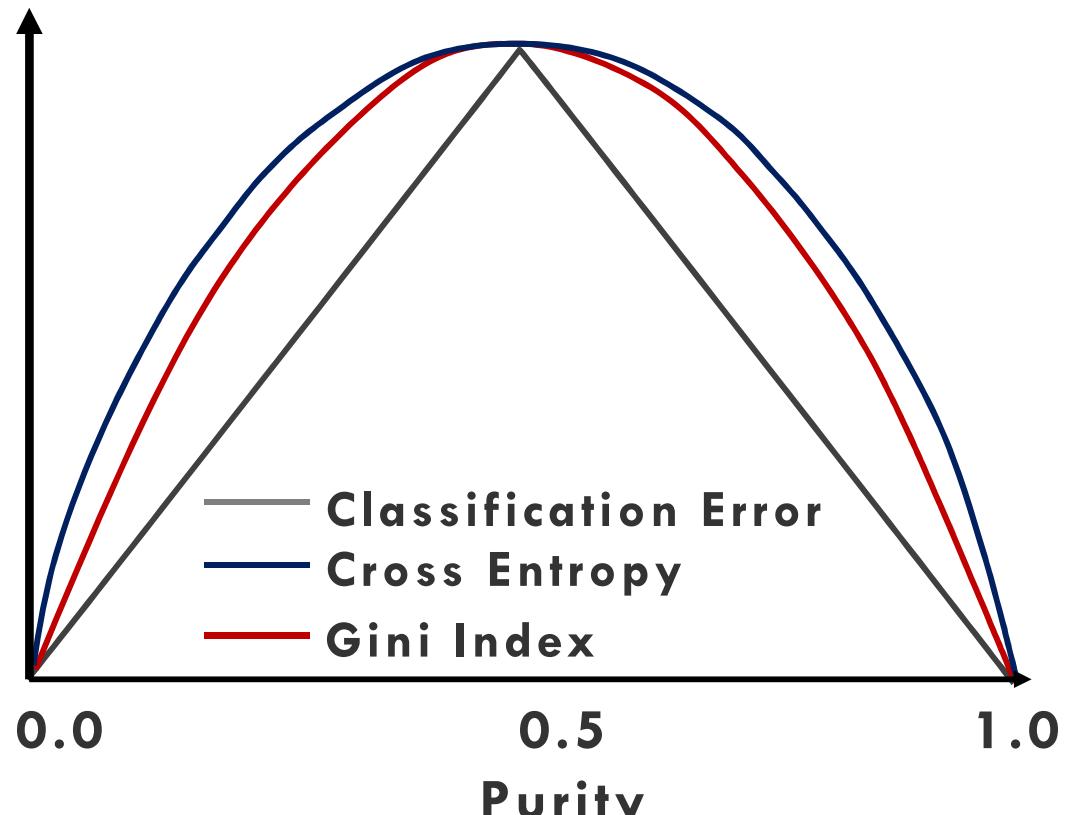
- In practice, Gini index often used for splitting



$$G(t) = 1 - \sum_{i=1}^n p(i|t)^2$$

The Gini Index

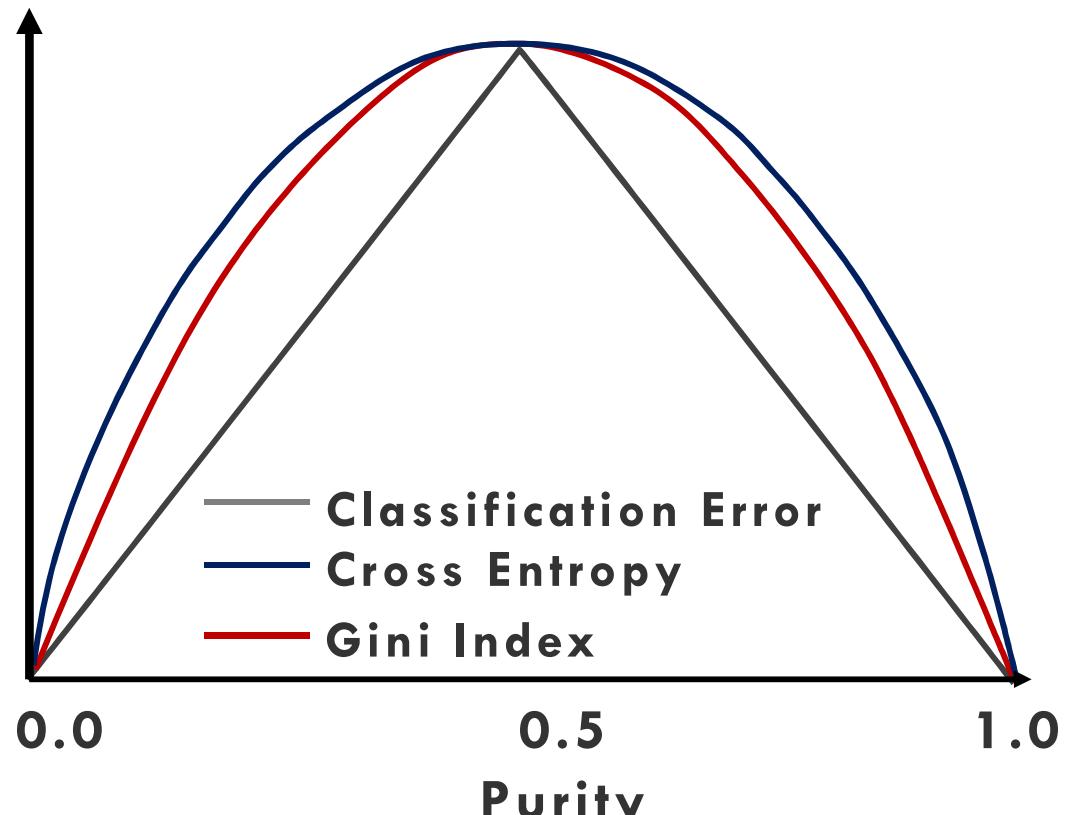
- In practice, Gini index often used for splitting
- Function is similar to entropy—has bulge



$$G(t) = 1 - \sum_{i=1}^n p(i|t)^2$$

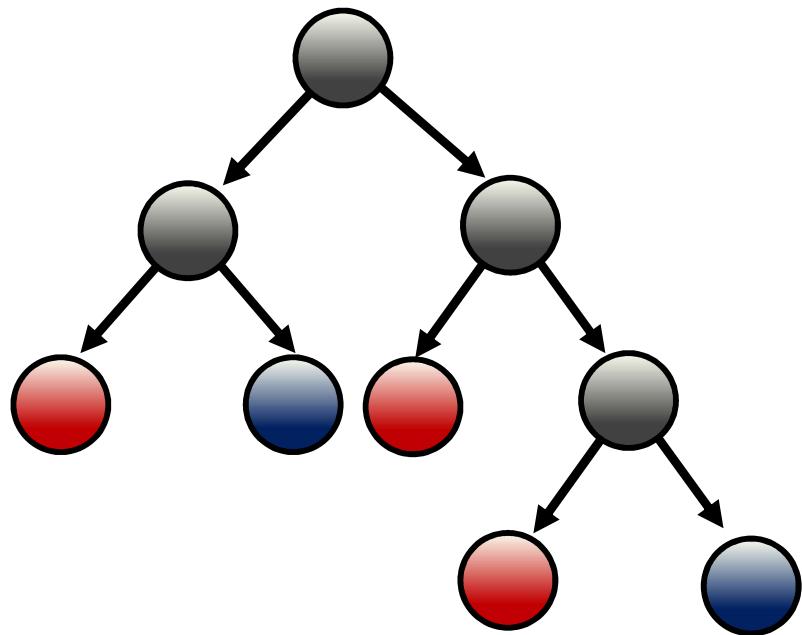
The Gini Index

- In practice, Gini index often used for splitting
- Function is similar to entropy—has bulge
- Does not contain logarithm



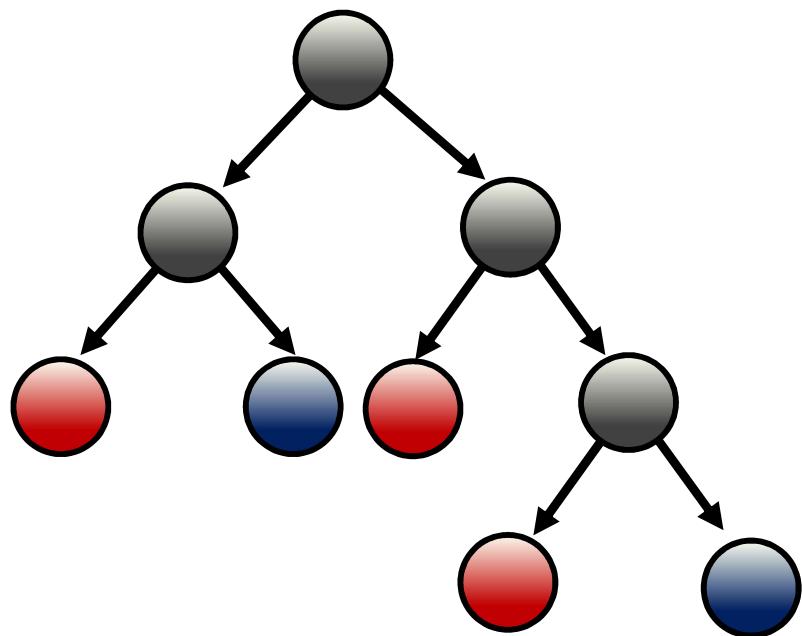
$$G(t) = 1 - \sum_{i=1}^n p(i|t)^2$$

Decision Trees are High Variance



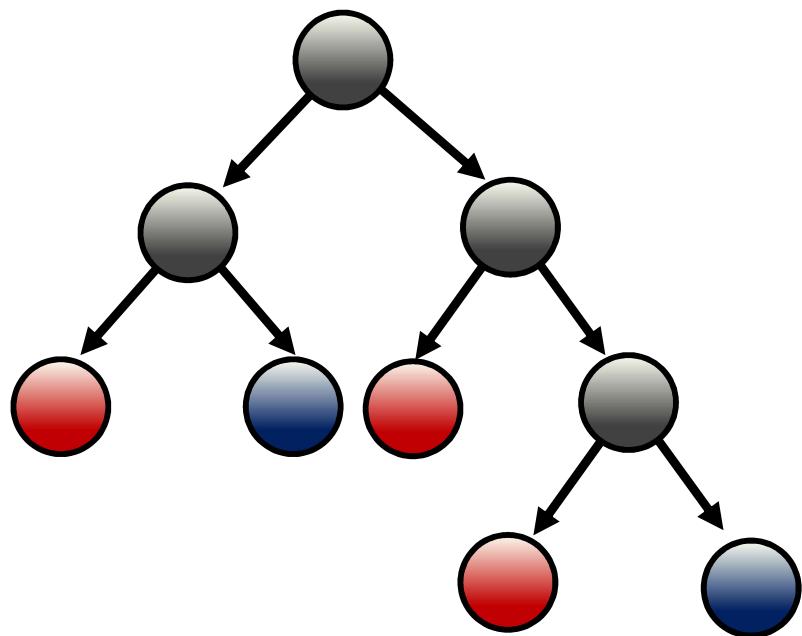
- Problem: decision trees tend to overfit

Decision Trees are High Variance



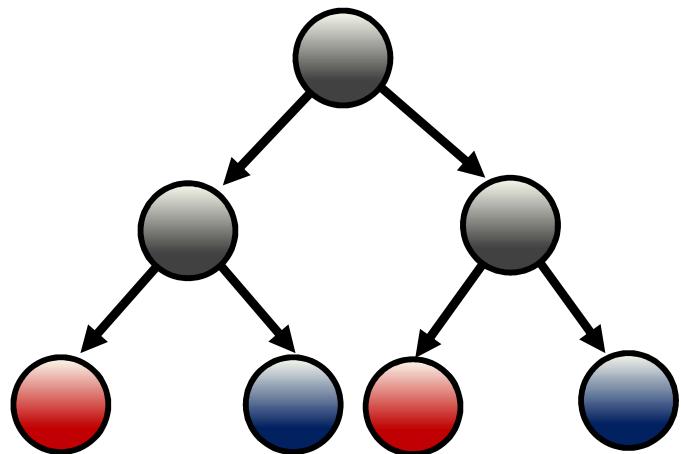
- Problem: decision trees tend to overfit
- Small changes in data greatly affect prediction--high variance
- Solution: Pruning

Decision Trees are High Variance



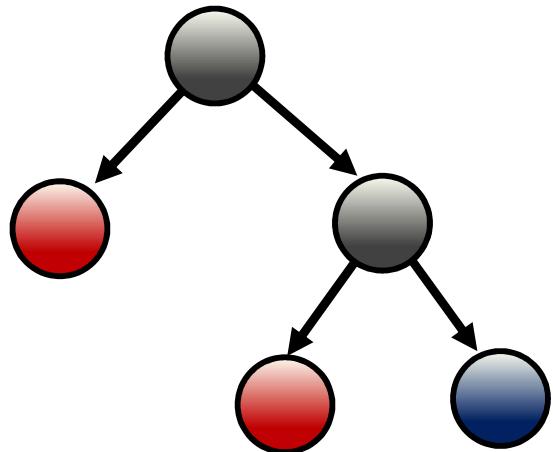
- Problem: decision trees tend to overfit
- Small changes in data greatly affect prediction--high variance
- Solution: Prune trees

Pruning Decision Trees



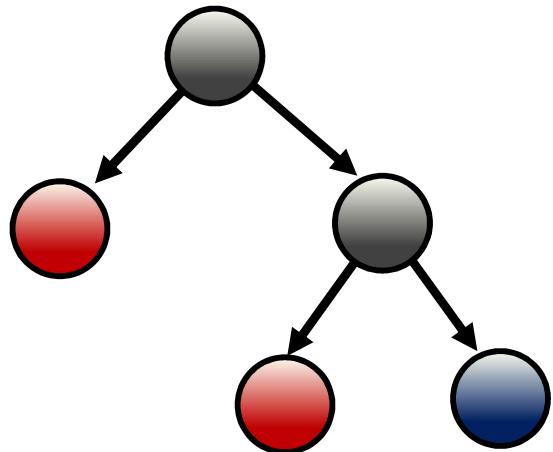
- Problem: decision trees tend to overfit
- Small changes in data greatly affect prediction--high variance
- Solution: Prune trees

Pruning Decision Trees



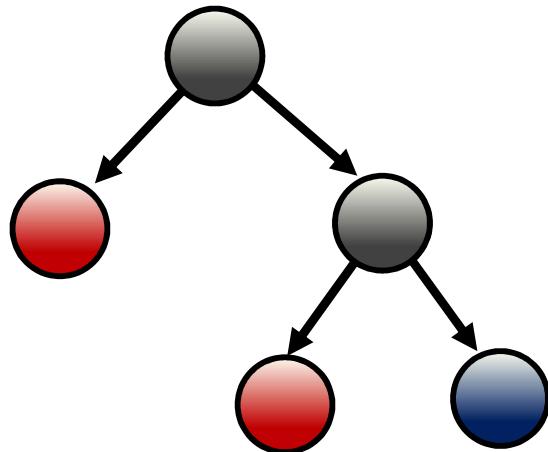
- Problem: decision trees tend to overfit
- Small changes in data greatly affect prediction--high variance
- Solution: Prune trees

Pruning Decision Trees



- How to decide what leaves to prune?

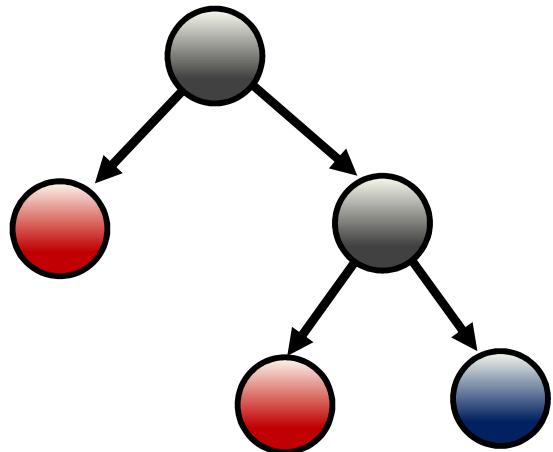
Pruning Decision Trees



- How to decide what leaves to prune?
- Solution: prune based on classification error threshold

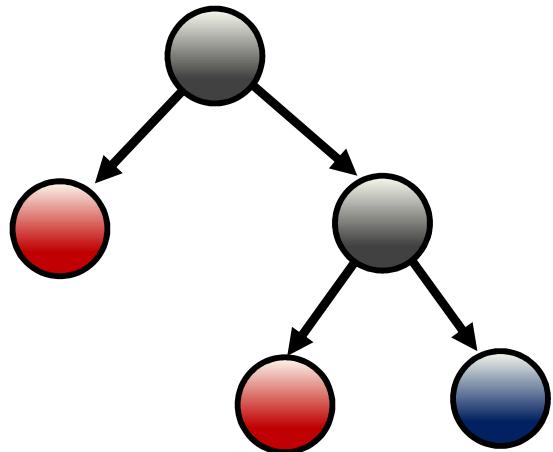
$$E(t) = 1 - \max_i[p(i|t)]$$

Strengths of Decision Trees



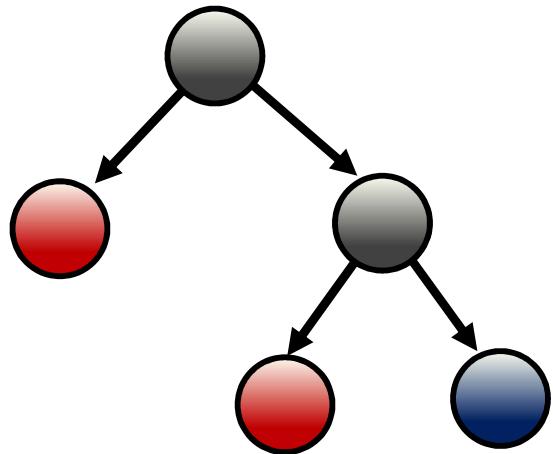
- Easy to interpret and implement—"if ... then ... else" logic

Strengths of Decision Trees



- Easy to interpret and implement—"if ... then ... else" logic
- Handle any data category—binary, ordinal, continuous

Strengths of Decision Trees



- Easy to interpret and implement—"if ... then ... else" logic
- Handle any data category—binary, ordinal, continuous
- No preprocessing or scaling required

DecisionTreeClassifier: The Syntax

Import the class containing the classification method

```
from sklearn.tree import DecisionTreeClassifier
```

DecisionTreeClassifier: The Syntax

Import the class containing the classification method

```
from sklearn.tree import DecisionTreeClassifier
```

Create an instance of the class

```
DTC = DecisionTreeClassifier(criterion='gini',  
                             max_features=10, max_depth=5)
```

DecisionTreeClassifier: The Syntax

Import the class containing the classification method

```
from sklearn.tree import DecisionTreeClassifier
```

Create an instance of the class

```
DTC = DecisionTreeClassifier(criterion='gini',  
                             max_features=10, max_depth=5)
```



tree
parameters

DecisionTreeClassifier: The Syntax

Import the class containing the classification method

```
from sklearn.tree import DecisionTreeClassifier
```

Create an instance of the class

```
DTC = DecisionTreeClassifier(criterion='gini',  
                             max_features=10, max_depth=5)
```

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

```
DTC = DTC.fit(X_train, y_train)  
y_predict = DTC.predict(X_test)
```

DecisionTreeClassifier: The Syntax

Import the class containing the classification method

```
from sklearn.tree import DecisionTreeClassifier
```

Create an instance of the class

```
DTC = DecisionTreeClassifier(criterion='gini',  
                             max_features=10, max_depth=5)
```

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

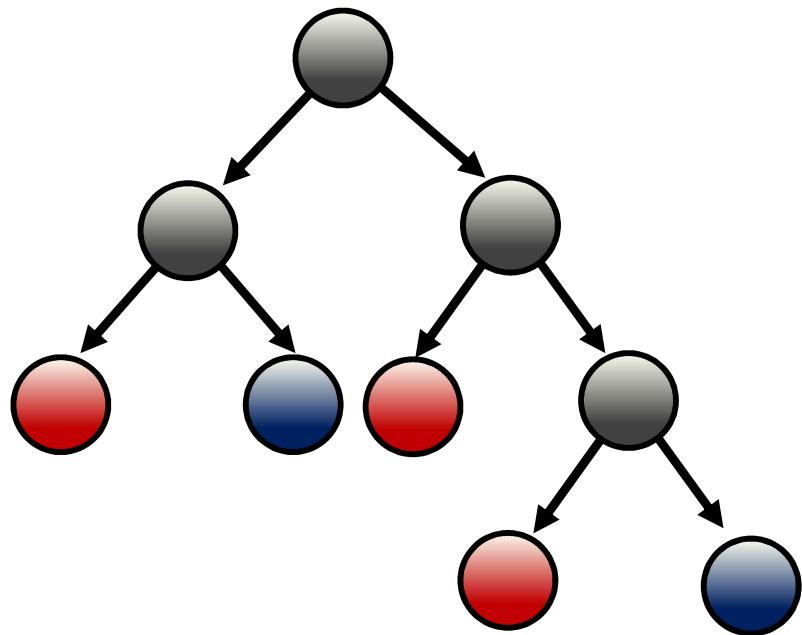
```
DTC = DTC.fit(X_train, y_train)  
y_predict = DTC.predict(X_test)
```

Tune parameters with cross-validation. Use [DecisionTreeRegressor](#) for regression.



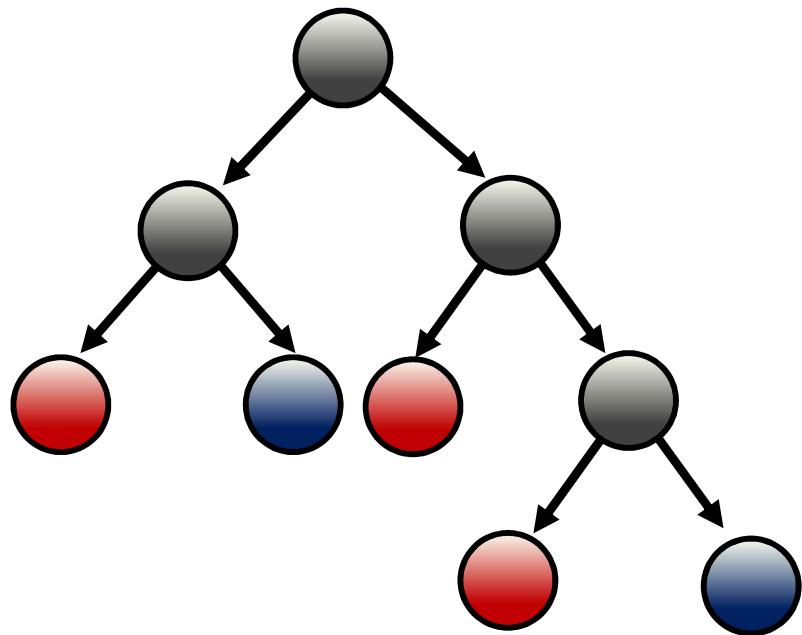
2.14. Bagging

Decision Trees are High Variance



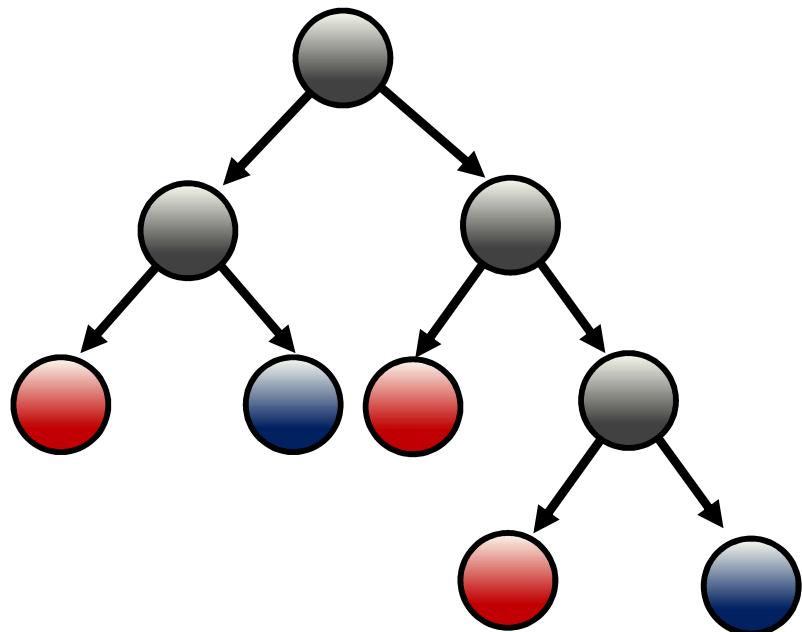
- Problem: decision trees tend to overfit

Decision Trees are High Variance



- Problem: decision trees tend to overfit
- Pruning helps reduce variance to a point

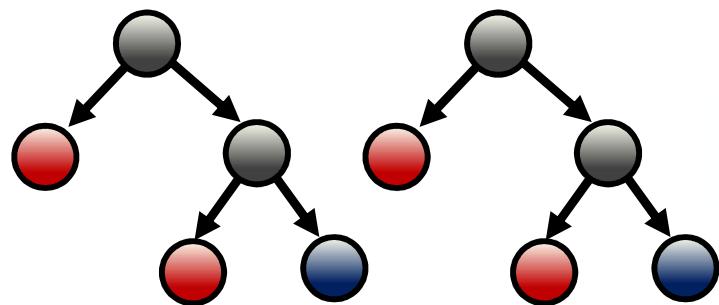
Decision Trees are High Variance



- Problem: decision trees tend to overfit
- Pruning helps reduce variance to a point
- Often not significant for model to generalize well

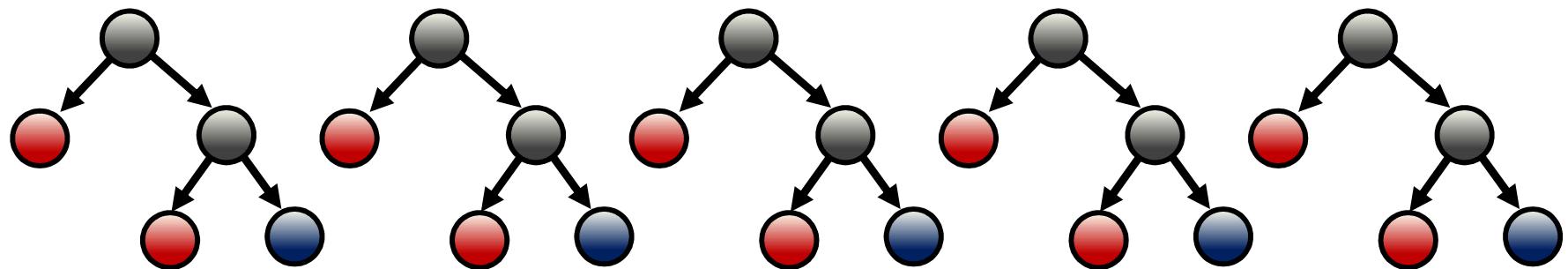
Improvement: Use Many Trees

Create many different trees



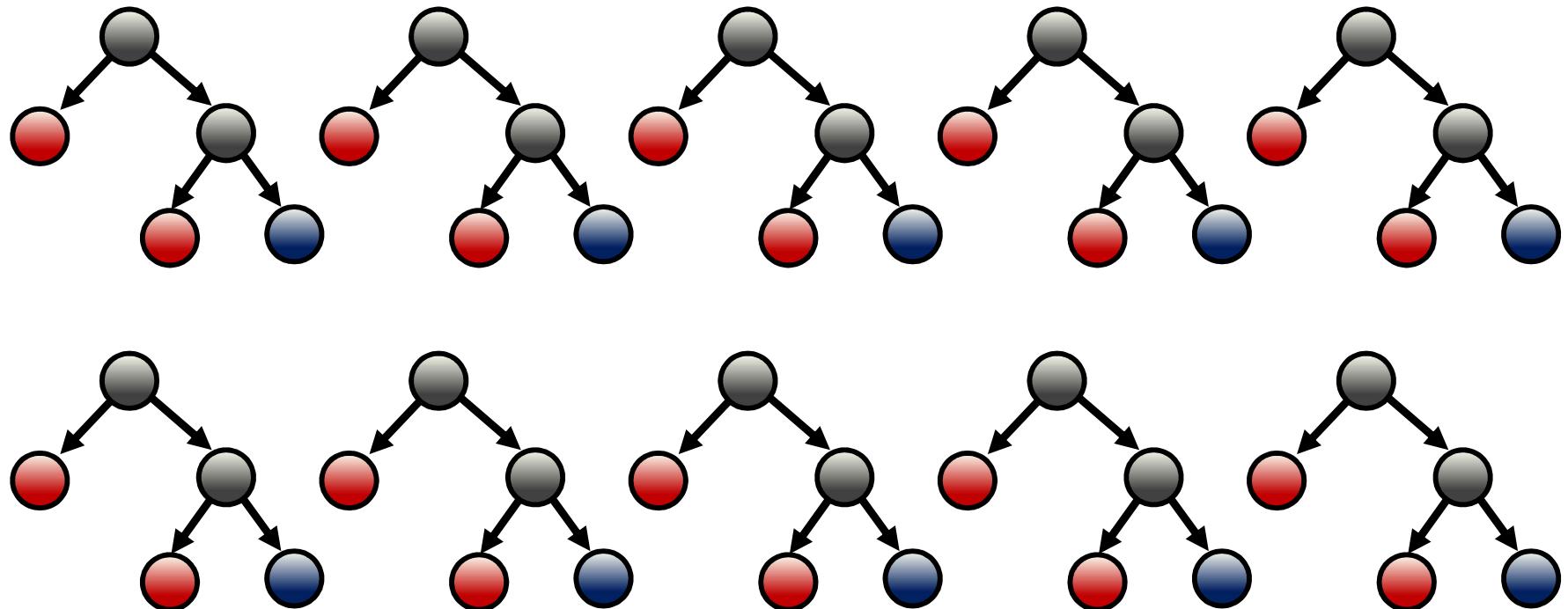
Improvement: Use Many Trees

Create many different trees



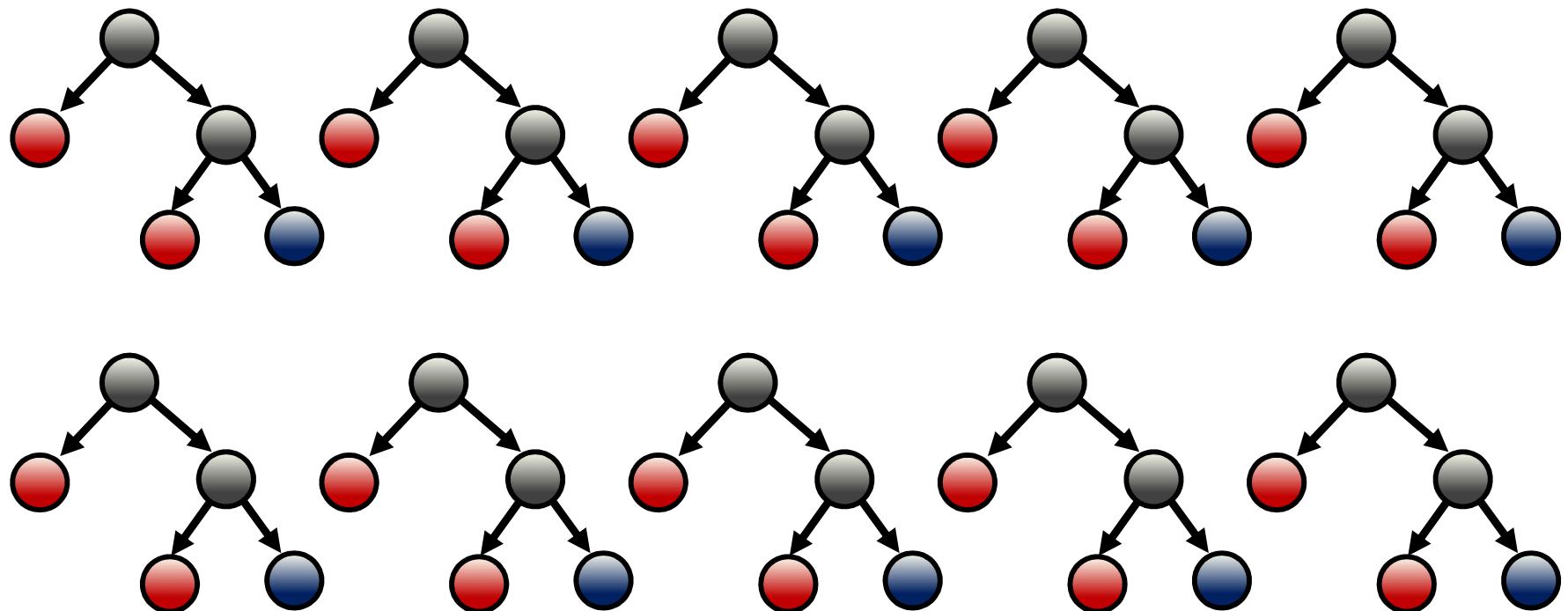
Improvement: Use Many Trees

Create many different trees



Improvement: Use Many Trees

Combine predictions to reduce variance



How to Create Multiple Trees?

Use bootstrapping: sample data with replacement

	Date	Title	Budget	DomesticTotalGross	Director	Rating	Runtime
0	2013-11-22	The Hunger Games: Catching Fire	130000000	242668047	Francis Lawrence	PG-13	146
1	2013-05-03	Iron Man 3	20000000	409013984	Shane Black	PG-13	129
2	2013-11-22	Frozen	15000000	400738009	Chris BuckJennifer Lee	PG	108
3	2013-07-03	Despicable Me 2	7600000	368061265	Pierre CoffinChris Renaud	PG	98
4	2013-06-14	Man of Steel	22500000	291045518	Zack Snyder	PG-13	143
5	2013-10-04	Gravity	10000000	274092705	Alfonso Cuaron	PG-13	91
6	2013-06-21	Monsters University	NaN	268492764	Dan Scanlon	G	107
7	2013-12-13	The Hobbit: The Desolation of Smaug	NaN	258366855	Peter Jackson	PG-13	161
8	2013-05-24	Fast & Furious 6	16000000	238879850	Justin Lin	PG-13	130
9	2013-03-08	Oz The Great and Powerful	21500000	234911825	Sam Raimi	PG	127
10	2013-05-16	Star Trek Into Darkness	19000000	228778661	J.J. Abrams	PG-13	123
11	2013-11-08	Thor: The Dark World	17000000	206582140	Alan Taylor	PG-13	120
12	2013-06-21	World War Z	19000000	202359711	Marc Forster	PG-13	116
13	2013-03-22	The Croods	13500000	187168425	Kirk De MiccoChris Sanders	PG	98
14	2013-06-28	The Heat	4300000	159582188	Paul Feig	R	117
15	2013-09-07	We're the Millers	3700000	150394119	Rawson Marshall Thurber	R	110
16	2013-12-13	American Hustle	4000000	150117807	David O. Russell	R	138
17	2013-05-10	The Great Gatsby	10500000	144840419	Baz Luhrmann	PG-13	143

How to Create Multiple Trees?

Create multiple bootstrapped samples

	Date	Title	Budget	DomesticTotalGross	Director	Rating	Runtime
0	2013-11-22	The Hunger Games: Catching Fire	1300000000	242668047	Francis Lawrence	PG-13	146
1	2013-05-03	Iron Man 3	200000000	409013984	Shane Black	PG-13	129
2	2013-11-22	Frozen	150000000	400738009	Chris BuckJennifer Lee	PG	108
3	2013-07-03	Despicable Me 2	76000000	368061265	Pierre CoffeChris Renaud	PG	98
4	2013-06-14	Man of Steel	225000000	291045518	Zack Snyder	PG-13	143
5	2013-10-04	Gravity	100000000	274092705	Alfonso Cuaron	PG-13	91
6	2013-06-21	Monsters University	Nan	268492764	Dan Scanlon	G	107
7	2013-12-13	The Hobbit: The Desolation of Smaug	258366855	200000000	Peter Jackson	PG-13	161
8	2013-06-21	Fast & Furious 6	160000000	238679850	Justin Lin	PG-13	130
9	2013-03-08	Oz The Great and Powerful	215000000	234911825	Sam Raimi	PG	127
10	2013-05-16	Star Trek Into Darkness	190000000	228778661	J.J. Abrams	PG-13	123
11	2013-11-08	Thor: The Dark World	170000000	206582140	Alan Taylor	PG-13	120
12	2013-06-21	World War Z	190000000	202359711	Marc Forster	PG-13	116
13	2013-03-22	The Croods	135000000	187168425	Kirk De MiccoChris Sanders	PG	98
14	2013-06-28	The Heat	43000000	159582188	Paul Feig	R	117
15	2013-08-07	We're the Millers	37000000	150394119	Rawson Marshall Thurber	R	110
16	2013-12-13	American Hustle	40000000	150117807	David O. Russell	R	138
17	2013-05-10	The Great Gatsby	105000000	144840419	Baz Luhrmann	PG-13	143

Date	Title	Budget	DomesticTotalGross	Director	Rating	Runtime	
0	2013-11-22	The Hunger Games: Catching Fire	190000000	24969847	Francis Lawrence	PG-13	146
1	2013-05-03	Iron Man 3	200000000	40613994	Shane Black	PG-13	129
2	2013-11-22	Frozen	150000000	400738009	Chris BuckJennifer Lee	PG	108
3	2013-07-03	Despicable Me 2	76000000	368061265	Pierre CoffeChris Renaud	PG	98
4	2013-06-14	Man of Steel	225000000	291045518	Zack Snyder	PG-13	143
5	2013-10-04	Gravity	100000000	274092705	Alfonso Cuaron	PG-13	91
6	2013-06-21	Monsters University	Nan	268492764	Dan Scanlon	G	107
7	2013-12-13	The Hobbit: The Desolation of Smaug	258366855	200000000	Peter Jackson	PG-13	161
8	2013-06-21	Fast & Furious 6	160000000	238679850	Justin Lin	PG-13	130
9	2013-03-08	Oz The Great and Powerful	215000000	234911825	Sam Raimi	PG	127
10	2013-05-16	Star Trek Into Darkness	190000000	228778661	J.J. Abrams	PG-13	123
11	2013-11-08	Thor: The Dark World	170000000	206582140	Alan Taylor	PG-13	120
12	2013-06-21	World War Z	190000000	202359711	Marc Forster	PG-13	116
13	2013-03-22	The Croods	135000000	187168425	Kirk De MiccoChris Sanders	PG	98
14	2013-06-28	The Heat	43000000	159582188	Paul Feig	R	117
15	2013-08-07	We're the Millers	37000000	150394119	Rawson Marshall Thurber	R	110
16	2013-12-13	American Hustle	40000000	150117807	David O. Russell	R	138
17	2013-05-10	The Great Gatsby	105000000	144840419	Baz Luhrmann	PG-13	143

Date	Title	Budget	DomesticTotalGross	Director	Rating	Runtime	
0	2013-11-22	The Hunger Games: Catching Fire	190000000	24969847	Francis Lawrence	PG-13	146
1	2013-05-03	Iron Man 3	200000000	40613994	Shane Black	PG-13	129
2	2013-11-22	Frozen	150000000	400738009	Chris BuckJennifer Lee	PG	108
3	2013-07-03	Despicable Me 2	76000000	368061265	Pierre CoffeChris Renaud	PG	98
4	2013-06-14	Man of Steel	225000000	291045518	Zack Snyder	PG-13	143
5	2013-10-04	Gravity	100000000	274092705	Alfonso Cuaron	PG-13	91
6	2013-06-21	Monsters University	Nan	268492764	Dan Scanlon	G	107
7	2013-12-13	The Hobbit: The Desolation of Smaug	258366855	200000000	Peter Jackson	PG-13	161
8	2013-06-21	Fast & Furious 6	160000000	238679850	Justin Lin	PG-13	130
9	2013-03-08	Oz The Great and Powerful	215000000	234911825	Sam Raimi	PG	127
10	2013-05-16	Star Trek Into Darkness	190000000	228778661	J.J. Abrams	PG-13	123
11	2013-11-08	Thor: The Dark World	170000000	206582140	Alan Taylor	PG-13	120
12	2013-06-21	World War Z	190000000	202359711	Marc Forster	PG-13	116
13	2013-03-22	The Croods	135000000	187168425	Kirk De MiccoChris Sanders	PG	98
14	2013-06-28	The Heat	43000000	159582188	Paul Feig	R	117
15	2013-08-07	We're the Millers	37000000	150394119	Rawson Marshall Thurber	R	110
16	2013-12-13	American Hustle	40000000	150117807	David O. Russell	R	138
17	2013-05-10	The Great Gatsby	105000000	144840419	Baz Luhrmann	PG-13	143

Date	Title	Budget	DomesticTotalGross	Director	Rating	Runtime	
0	2013-11-22	The Hunger Games: Catching Fire	190000000	24969847	Francis Lawrence	PG-13	146
1	2013-05-03	Iron Man 3	200000000	40613994	Shane Black	PG-13	129
2	2013-11-22	Frozen	150000000	400738009	Chris BuckJennifer Lee	PG	108
3	2013-07-03	Despicable Me 2	76000000	368061265	Pierre CoffeChris Renaud	PG	98
4	2013-06-14	Man of Steel	225000000	291045518	Zack Snyder	PG-13	143
5	2013-10-04	Gravity	100000000	274092705	Alfonso Cuaron	PG-13	91
6	2013-06-21	Monsters University	Nan	268492764	Dan Scanlon	G	107
7	2013-12-13	The Hobbit: The Desolation of Smaug	258366855	200000000	Peter Jackson	PG-13	161
8	2013-06-21	Fast & Furious 6	160000000	238679850	Justin Lin	PG-13	130
9	2013-03-08	Oz The Great and Powerful	215000000	234911825	Sam Raimi	PG	127
10	2013-05-16	Star Trek Into Darkness	190000000	228778661	J.J. Abrams	PG-13	123
11	2013-11-08	Thor: The Dark World	170000000	206582140	Alan Taylor	PG-13	120
12	2013-06-21	World War Z	190000000	202359711	Marc Forster	PG-13	116
13	2013-03-22	The Croods	135000000	187168425	Kirk De MiccoChris Sanders	PG	98
14	2013-06-28	The Heat	43000000	159582188	Paul Feig	R	117
15	2013-08-07	We're the Millers	37000000	150394119	Rawson Marshall Thurber	R	110
16	2013-12-13	American Hustle	40000000	150117807	David O. Russell	R	138
17	2013-05-10	The Great Gatsby	105000000	144840419	Baz Luhrmann	PG-13	143

How to Create Multiple Trees?

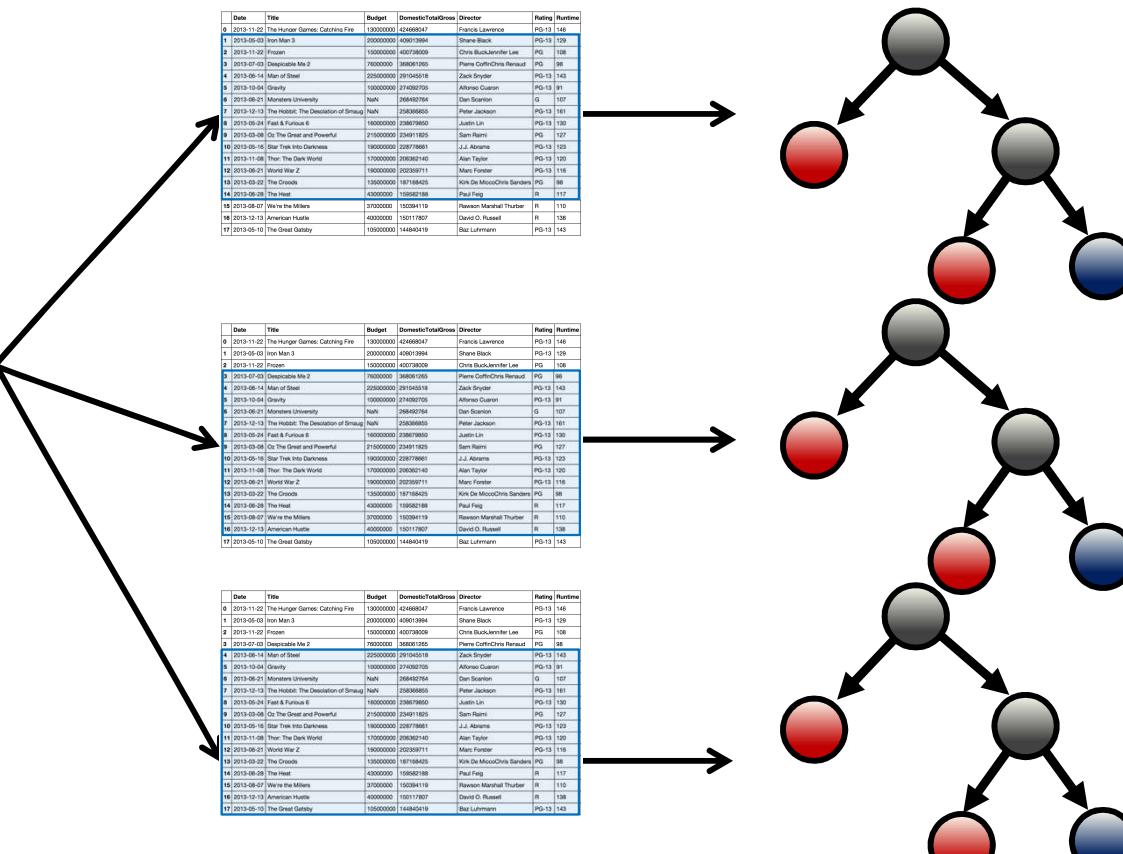
Grow decision tree from each bootstrapped sample

	Date	Title	Budget	DomesticTotalGross	Director	Rating	Runtime
0	2013-11-22	The Hunger Games: Catching Fire	130000000	242668047	Francis Lawrence	PG-13	146
1	2013-05-03	Iron Man 3	200000000	409013984	Shane Black	PG-13	129
2	2013-11-22	Frozen	150000000	400738009	Chris BuckJennifer Lee	PG	108
3	2013-07-03	Despicable Me 2	76000000	368061265	Pierre CoffeChris Renaud	PG	98
4	2013-06-14	Man of Steel	22500000	291045518	Zack Snyder	PG-13	143
5	2013-10-04	Gravity	10000000	274092705	Alfonso Cuaron	PG-13	91
6	2013-06-21	Monsters University	Nan	268492764	Dan Scanlon	G	107
7	2013-12-13	The Hobbit: The Desolation of Smaug	258366855	20000000	Peter Jackson	PG-13	161
8	2013-06-21	Fast & Furious 6	160000000	238679850	Justin Lin	PG-13	130
9	2013-03-08	Oz The Great and Powerful	215000000	234911825	Sam Raimi	PG	127
10	2013-05-16	Star Trek Into Darkness	190000000	228778661	J.J. Abrams	PG-13	123
11	2013-11-08	Thor: The Dark World	170000000	206582140	Alan Taylor	PG-13	120
12	2013-06-21	World War Z	190000000	202359711	Marc Forster	PG-13	116
13	2013-03-22	The Croods	135000000	187168425	Kirk De MiccoChris Sanders	PG	98
14	2013-06-28	The Heat	43000000	159562188	Paul Feig	R	117
15	2013-06-07	We're the Millers	37000000	150394119	Rawson Marshall Thurber	R	110
16	2013-12-13	American Hustle	40000000	150117807	David O. Russell	R	138
17	2013-05-10	The Great Gatsby	105000000	144840419	Baz Luhrmann	PG-13	143

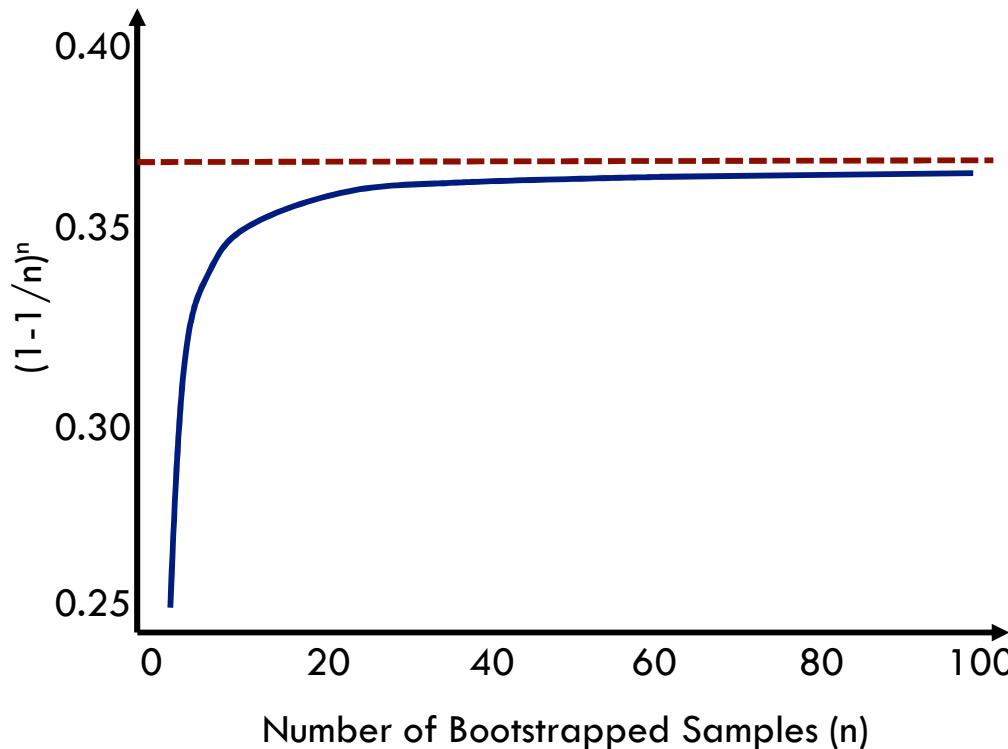
	Date	Title	Budget	DomesticTotalGross	Director	Rating	Runtime
0	2013-11-22	The Hunger Games: Catching Fire	190000000	24266847	Francis Lawrence	PG-13	146
1	2013-05-03	Iron Man 3	200000000	409013984	Shane Black	PG-13	129
2	2013-11-22	Frozen	150000000	400738009	Chris BuckJennifer Lee	PG	108
3	2013-07-03	Despicable Me 2	76000000	368061265	Pierre CoffeChris Renaud	PG	98
4	2013-06-14	Man of Steel	22500000	291045518	Zack Snyder	PG-13	143
5	2013-10-04	Gravity	100000000	274092705	Alfonso Cuaron	PG-13	91
6	2013-06-21	Monsters University	Nan	268492764	Dan Scanlon	G	107
7	2013-12-13	The Hobbit: The Desolation of Smaug	258366855	20000000	Peter Jackson	PG-13	161
8	2013-06-21	Fast & Furious 6	160000000	238679850	Justin Lin	PG-13	130
9	2013-03-08	Oz The Great and Powerful	215000000	234911825	Sam Raimi	PG	127
10	2013-05-16	Star Trek Into Darkness	190000000	228778661	J.J. Abrams	PG-13	123
11	2013-11-08	Thor: The Dark World	170000000	206582140	Alan Taylor	PG-13	120
12	2013-06-21	World War Z	190000000	202359711	Marc Forster	PG-13	116
13	2013-03-22	The Croods	135000000	187168425	Kirk De MiccoChris Sanders	PG	98
14	2013-06-28	The Heat	43000000	159562188	Paul Feig	R	117
15	2013-06-07	We're the Millers	37000000	150394119	Rawson Marshall Thurber	R	110
16	2013-12-13	American Hustle	40000000	150117807	David O. Russell	R	138
17	2013-05-10	The Great Gatsby	105000000	144840419	Baz Luhrmann	PG-13	143

	Date	Title	Budget	DomesticTotalGross	Director	Rating	Runtime
0	2013-11-22	The Hunger Games: Catching Fire	190000000	24266847	Francis Lawrence	PG-13	146
1	2013-05-03	Iron Man 3	200000000	409013984	Shane Black	PG-13	129
2	2013-11-22	Frozen	150000000	400738009	Chris BuckJennifer Lee	PG	108
3	2013-07-03	Despicable Me 2	76000000	368061265	Pierre CoffeChris Renaud	PG	98
4	2013-06-14	Man of Steel	22500000	291045518	Zack Snyder	PG-13	143
5	2013-10-04	Gravity	100000000	274092705	Alfonso Cuaron	PG-13	91
6	2013-06-21	Monsters University	Nan	268492764	Dan Scanlon	G	107
7	2013-12-13	The Hobbit: The Desolation of Smaug	258366855	20000000	Peter Jackson	PG-13	161
8	2013-06-21	Fast & Furious 6	160000000	238679850	Justin Lin	PG-13	130
9	2013-03-08	Oz The Great and Powerful	215000000	234911825	Sam Raimi	PG	127
10	2013-05-16	Star Trek Into Darkness	190000000	228778661	J.J. Abrams	PG-13	123
11	2013-11-08	Thor: The Dark World	170000000	206582140	Alan Taylor	PG-13	120
12	2013-06-21	World War Z	190000000	202359711	Marc Forster	PG-13	116
13	2013-03-22	The Croods	135000000	187168425	Kirk De MiccoChris Sanders	PG	98
14	2013-06-28	The Heat	43000000	159562188	Paul Feig	R	117
15	2013-06-07	We're the Millers	37000000	150394119	Rawson Marshall Thurber	R	110
16	2013-12-13	American Hustle	40000000	150117807	David O. Russell	R	138
17	2013-05-10	The Great Gatsby	105000000	144840419	Baz Luhrmann	PG-13	143

	Date	Title	Budget	DomesticTotalGross	Director	Rating	Runtime
0	2013-11-22	The Hunger Games: Catching Fire	190000000	24266847	Francis Lawrence	PG-13	146
1	2013-05-03	Iron Man 3	200000000	409013984	Shane Black	PG-13	129
2	2013-11-22	Frozen	150000000	400738009	Chris BuckJennifer Lee	PG	108
3	2013-07-03	Despicable Me 2	76000000	368061265	Pierre CoffeChris Renaud	PG	98
4	2013-06-14	Man of Steel	22500000	291045518	Zack Snyder	PG-13	143
5	2013-10-04	Gravity	100000000	274092705	Alfonso Cuaron	PG-13	91
6	2013-06-21	Monsters University	Nan	268492764	Dan Scanlon	G	107
7	2013-12-13	The Hobbit: The Desolation of Smaug	258366855	20000000	Peter Jackson	PG-13	161
8	2013-06-21	Fast & Furious 6	160000000	238679850	Justin Lin	PG-13	130
9	2013-03-08	Oz The Great and Powerful	215000000	234911825	Sam Raimi	PG	127
10	2013-05-16	Star Trek Into Darkness	190000000	228778661	J.J. Abrams	PG-13	123
11	2013-11-08	Thor: The Dark World	170000000	206582140	Alan Taylor	PG-13	120
12	2013-06-21	World War Z	190000000	202359711	Marc Forster	PG-13	116
13	2013-03-22	The Croods	135000000	187168425	Kirk De MiccoChris Sanders	PG	98
14	2013-06-28	The Heat	43000000	159562188	Paul Feig	R	117
15	2013-06-07	We're the Millers	37000000	150394119	Rawson Marshall Thurber	R	110
16	2013-12-13	American Hustle	40000000	150117807	David O. Russell	R	138
17	2013-05-10	The Great Gatsby	105000000	144840419	Baz Luhrmann	PG-13	143

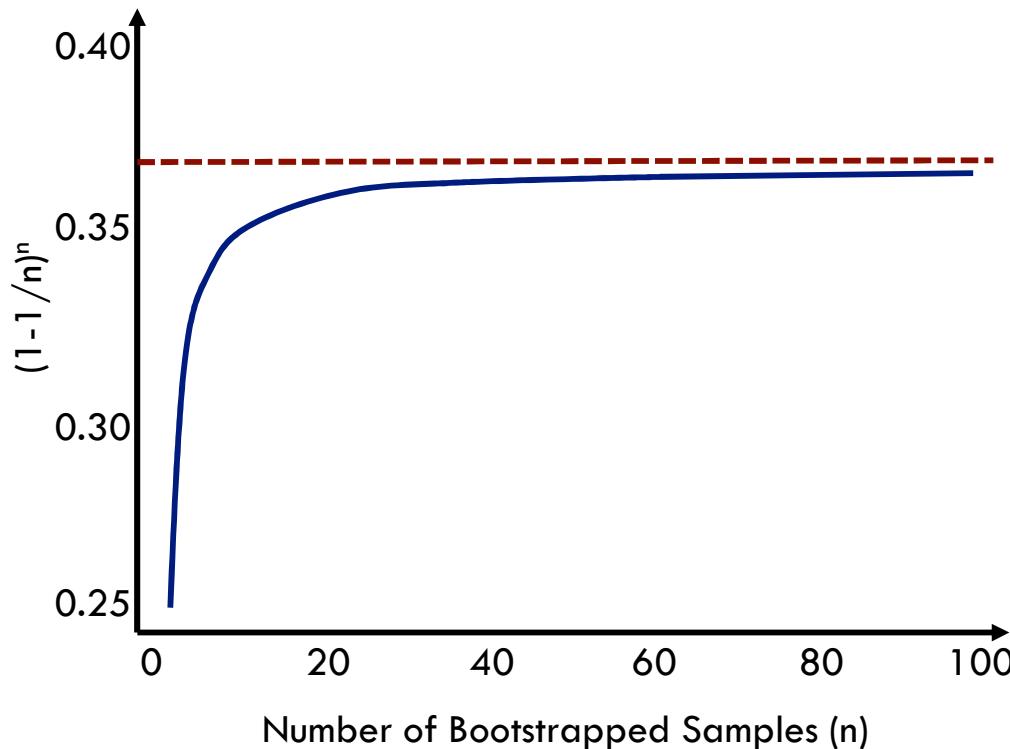


Distribution of Data in Bootstrapped Samples



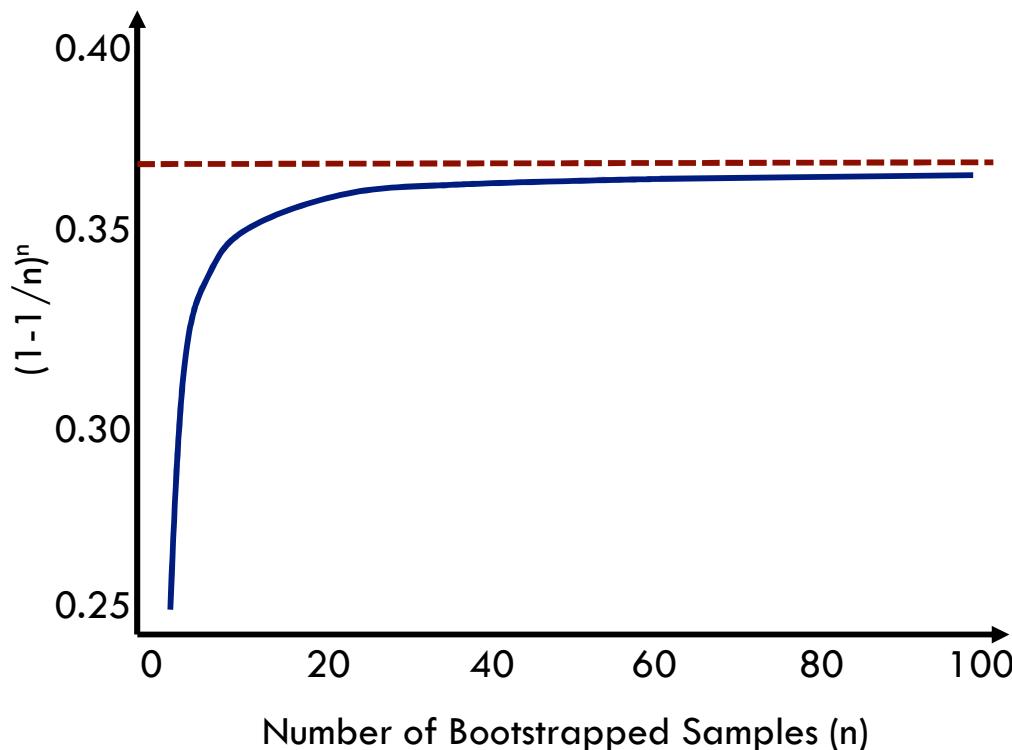
- Given a dataset, create n bootstrapped samples

Distribution of Data in Bootstrapped Samples



- Given a dataset, create n bootstrapped samples
- For a given record x ,
 $P(\text{rec } x \text{ not selected}) = (1 - 1/n)^n$

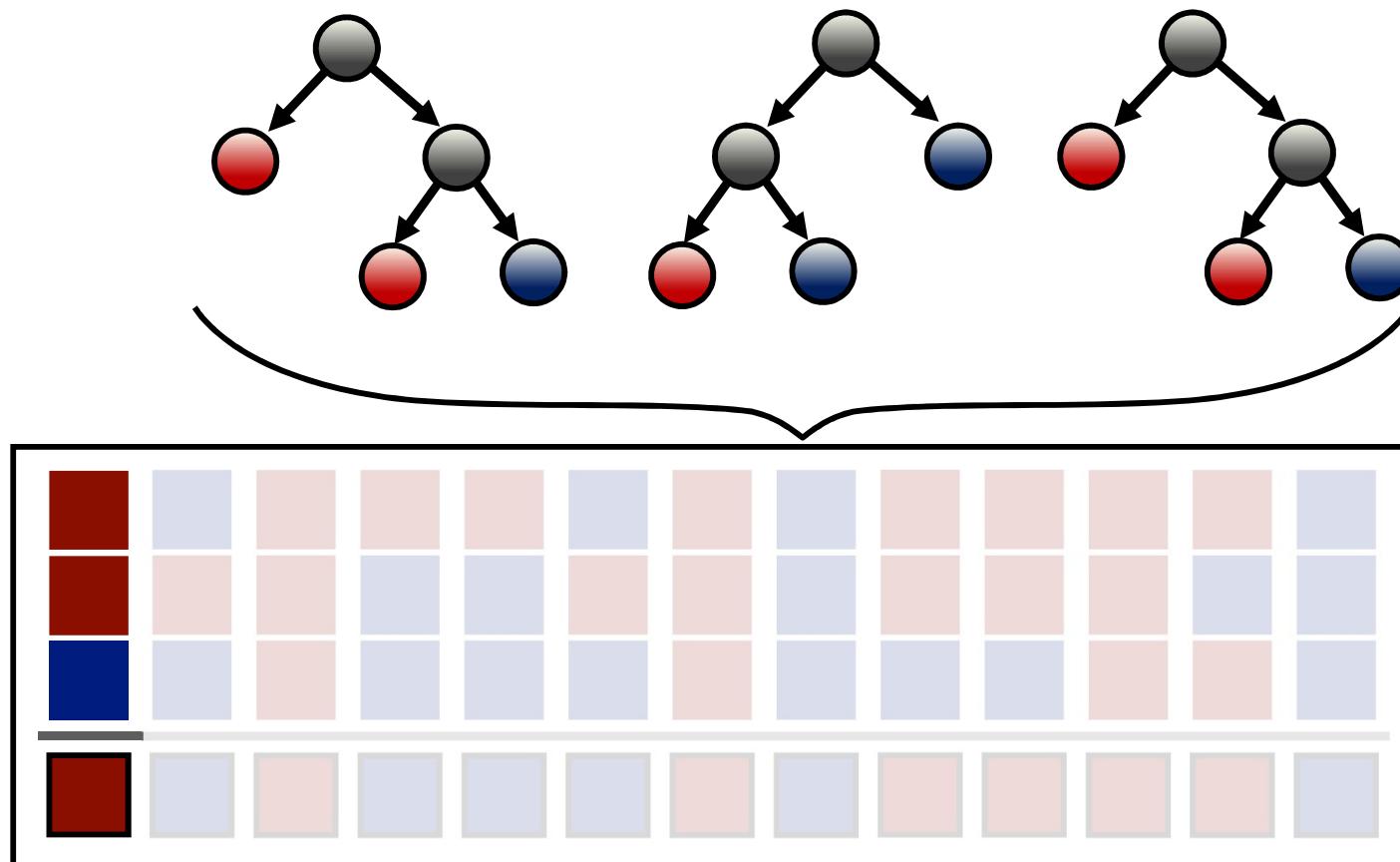
Distribution of Data in Bootstrapped Samples



- Given a dataset, create n bootstrapped samples
- For a given record x ,
 $P(\text{rec } x \text{ not selected}) = (1 - 1/n)^n$
- Each bootstrap sample contains approximately $2/3$ of the records

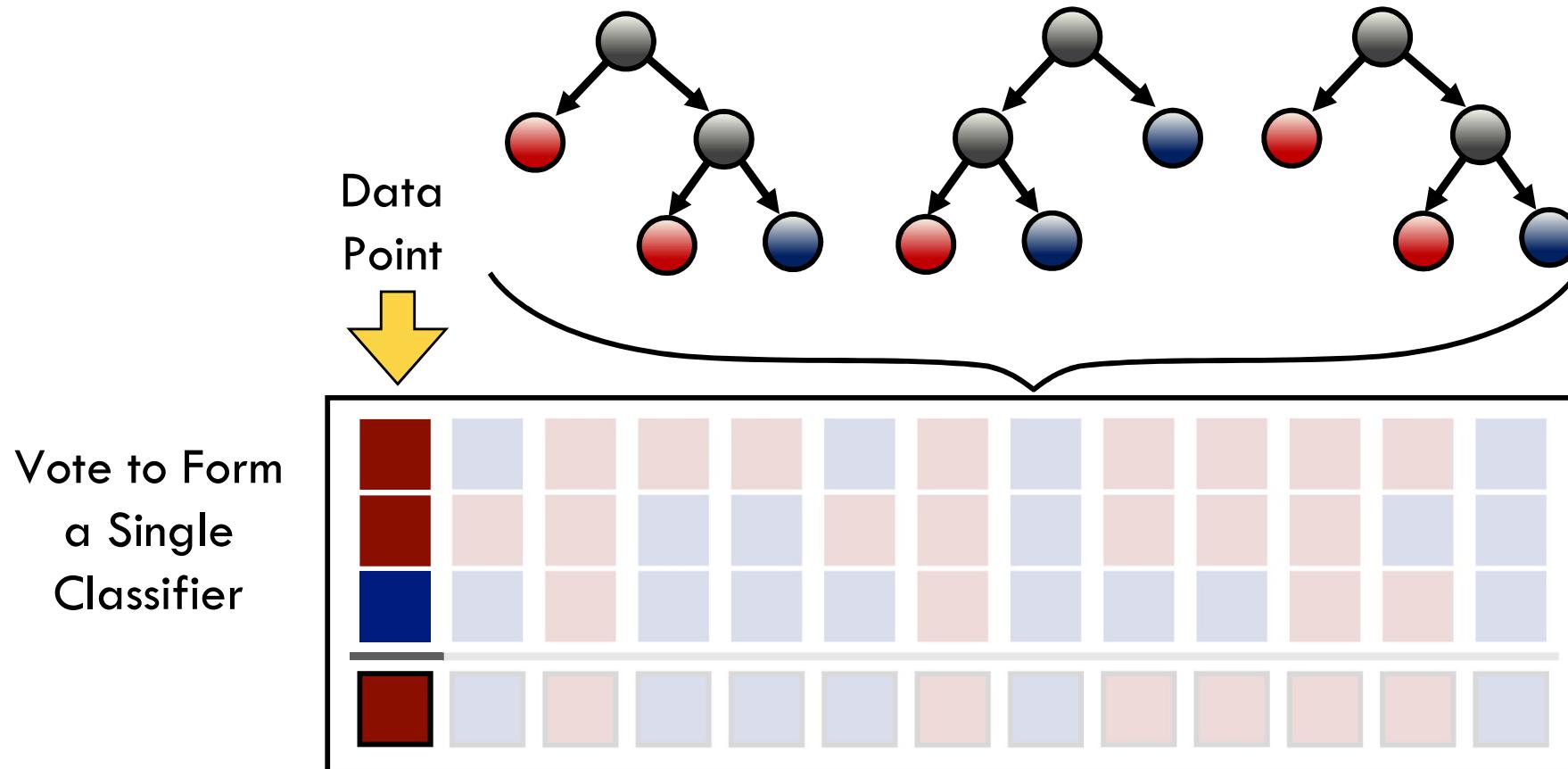
Aggregate Results

Trees vote on or average result for each data point



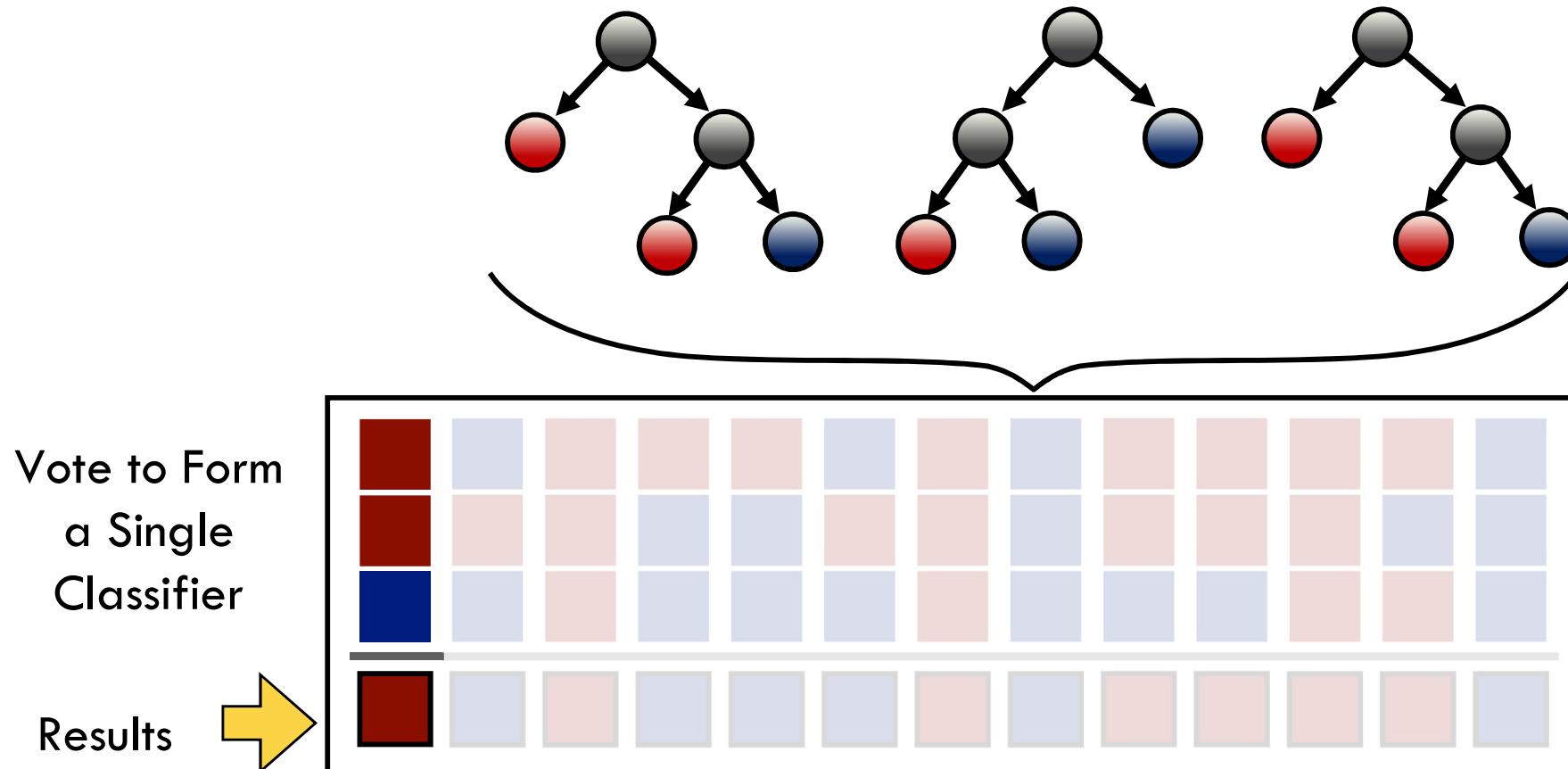
Aggregate Results

Trees vote on or average result for each data point



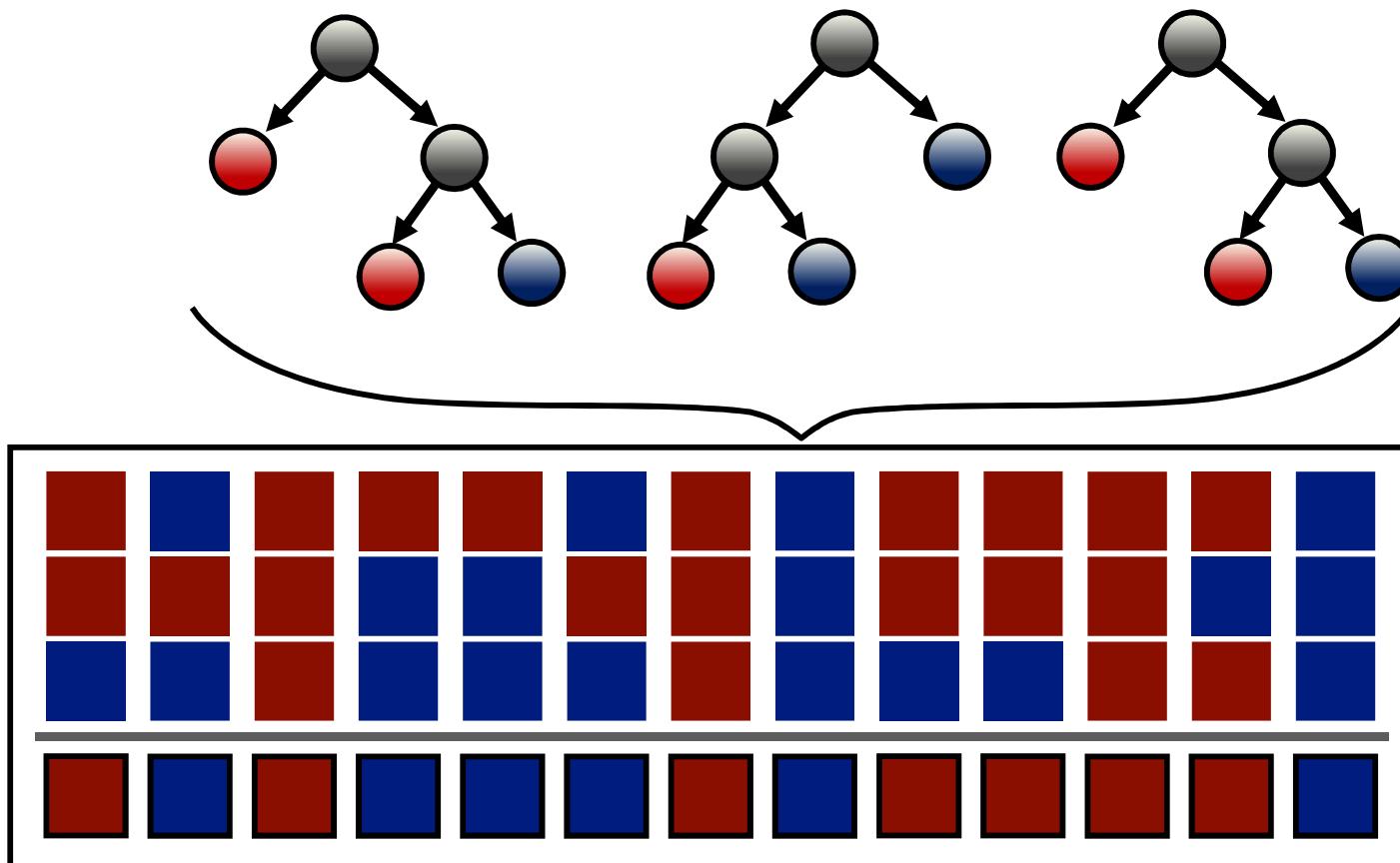
Aggregate Results

Trees vote on or average result for each data point



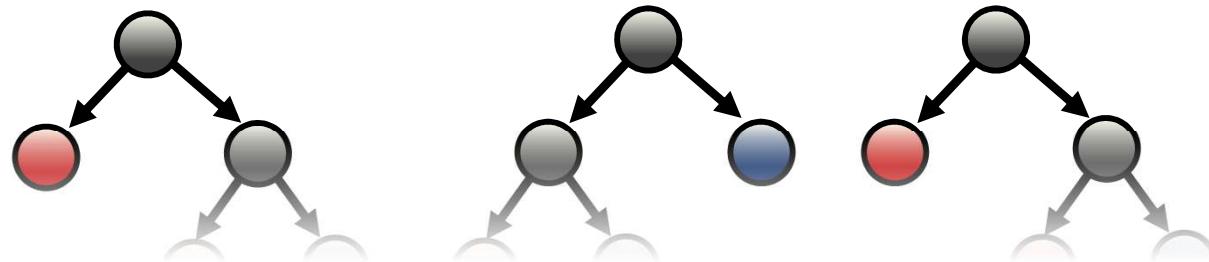
Aggregate Results

Trees vote on or average result for each data point



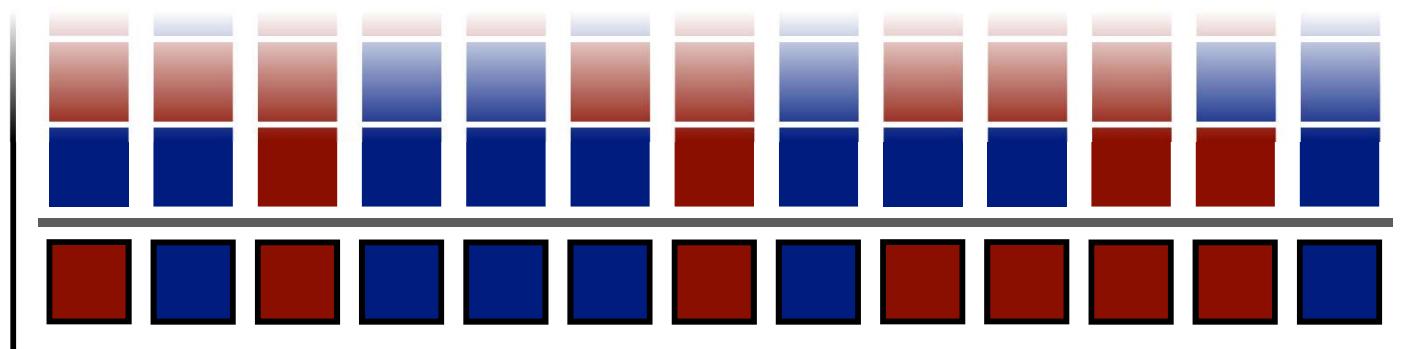
Aggregate Results

Trees vote on or average result for each data point



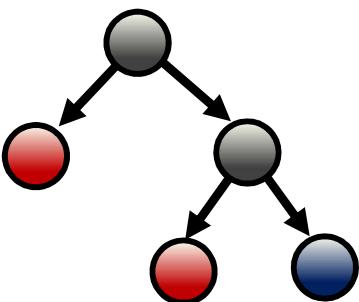
Bagging = Bootstrap Aggregating

VOTE TO FORM
a Single
Classifier



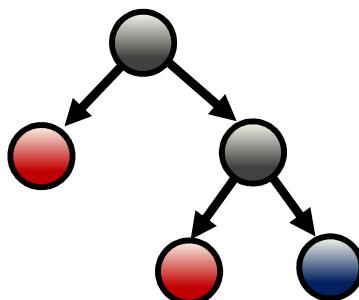
Bagging error calculations

Date	Title	Budget	DomesticTotalGross	Director	Rating	Runtime
0	2013-11-22 The Hunger Games: Catching Fire	130000000	424668047	Francis Lawrence	PG-13	146
1	2013-05-03 Iron Man 3	200000000	409013994	Shane Black	PG-13	129
2	2013-11-22 Frozen	150000000	400738009	Chris Buck/Jennifer Lee	PG	108
3	2013-07-03 Despicable Me 2	76000000	368061265	Pierre Coffin/Chris Renaud	PG	98
4	2013-06-14 Man of Steel	225000000	291045518	Zack Snyder	PG-13	143
5	2013-10-04 Gravity	100000000	274092705	Alfonso Cuaron	PG-13	91
6	2013-06-21 Monsters University	Nan	268492764	Dan Scanlon	G	107
7	2013-12-13 The Hobbit: The Desolation of Smaug	Nan	258366855	Peter Jackson	PG-13	161
8	2013-05-24 Fast & Furious 6	160000000	238679850	Justin Lin	PG-13	130
9	2013-03-08 Oz The Great and Powerful	215000000	234911825	Sam Raimi	PG	127
10	2013-05-16 Star Trek Into Darkness	190000000	228778661	J.J. Abrams	PG-13	123
11	2013-11-08 Thor: The Dark World	170000000	206362140	Alan Taylor	PG-13	120
12	2013-06-21 World War Z	190000000	202359711	Marc Forster	PG-13	116
13	2013-03-22 The Croods	135000000	187168425	Kirk De Micco/Chris Sanders	PG	98
14	2013-06-28 The Heat	43000000	159582188	Paul Feig	R	117
15	2013-08-07 We're the Millers	37000000	150394119	Rawson Marshall Thurber	R	110
16	2013-12-13 American Hustle	40000000	150117807	David O. Russell	R	138
17	2013-05-10 The Great Gatsby	105000000	144840419	Baz Luhrmann	PG-13	143



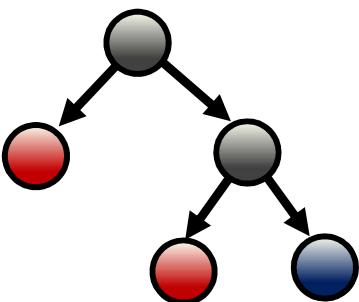
- Bootstrapped samples provide built-in error estimate for each tree
- Create tree based on subset of

Date	Title	Budget	DomesticTotalGross	Director	Rating	Runtime
0	2013-11-22 The Hunger Games: Catching Fire	130000000	424668047	Francis Lawrence	PG-13	146
1	2013-05-03 Iron Man 3	200000000	409013994	Shane Black	PG-13	129
2	2013-11-22 Frozen	150000000	400738009	Chris Buck/Jennifer Lee	PG	108
3	2013-07-03 Despicable Me 2	76000000	368061265	Pierre Coffin/Chris Renaud	PG	98
4	2013-06-14 Man of Steel	225000000	291045518	Zack Snyder	PG-13	143
5	2013-10-04 Gravity	100000000	274092705	Alfonso Cuaron	PG-13	91
6	2013-06-21 Monsters University	Nan	268492764	Dan Scanlon	G	107
7	2013-12-13 The Hobbit: The Desolation of Smaug	Nan	258366855	Peter Jackson	PG-13	161
8	2013-05-24 Fast & Furious 6	160000000	238679850	Justin Lin	PG-13	130
9	2013-03-08 Oz The Great and Powerful	215000000	234911825	Sam Raimi	PG	127
10	2013-05-16 Star Trek Into Darkness	190000000	228778661	J.J. Abrams	PG-13	123
11	2013-11-08 Thor: The Dark World	170000000	206362140	Alan Taylor	PG-13	120
12	2013-06-21 World War Z	190000000	202359711	Marc Forster	PG-13	116
13	2013-03-22 The Croods	135000000	187168425	Kirk De Micco/Chris Sanders	PG	98
14	2013-06-28 The Heat	43000000	159582188	Paul Feig	R	117
15	2013-08-07 We're the Millers	37000000	150394119	Rawson Marshall Thurber	R	110
16	2013-12-13 American Hustle	40000000	150117807	David O. Russell	R	138
17	2013-05-10 The Great Gatsby	105000000	144840419	Baz Luhrmann	PG-13	143



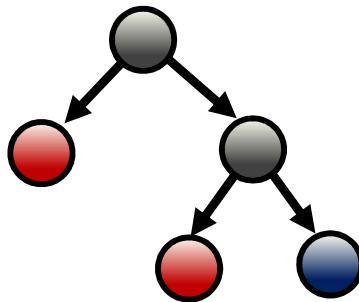
Bagging error calculations

Date	Title	Budget	DomesticTotalGross	Director	Rating	Runtime
0	2013-11-22	The Hunger Games: Catching Fire	130000000	424668047	Francis Lawrence	PG-13 146
1	2013-05-03	Iron Man 3	200000000	409013994	Shane Black	PG-13 129
2	2013-11-22	Frozen	150000000	400738009	Chris Buck/Jennifer Lee	PG 108
3	2013-07-03	Despicable Me 2	76000000	368061265	Pierre Coffin/Chris Renaud	PG 98
4	2013-06-14	Man of Steel	225000000	291045518	Zack Snyder	PG-13 143
5	2013-10-04	Gravity	100000000	274092705	Alfonso Cuaron	PG-13 91
6	2013-06-21	Monsters University	Nan	268492764	Dan Scanlon	G 107
7	2013-12-13	The Hobbit: The Desolation of Smaug	Nan	258366855	Peter Jackson	PG-13 161
8	2013-05-24	Fast & Furious 6	160000000	238679850	Justin Lin	PG-13 130
9	2013-03-08	Oz The Great and Powerful	215000000	234911825	Sam Raimi	PG 127
10	2013-05-16	Star Trek Into Darkness	190000000	228778661	J.J. Abrams	PG-13 123
11	2013-11-08	Thor: The Dark World	170000000	206362140	Alan Taylor	PG-13 120
12	2013-06-21	World War Z	190000000	202359711	Marc Forster	PG-13 116
13	2013-03-22	The Croods	135000000	187168425	Kirk De Micco/Chris Sanders	PG 98
14	2013-06-28	The Heat	43000000	159582188	Paul Feig	R 117
15	2013-08-07	We're the Millers	37000000	150394119	Rawson Marshall Thurber	R 110
16	2013-12-13	American Hustle	40000000	150117807	David O. Russell	R 138
17	2013-05-10	The Great Gatsby	105000000	144840419	Baz Luhrmann	PG-13 143



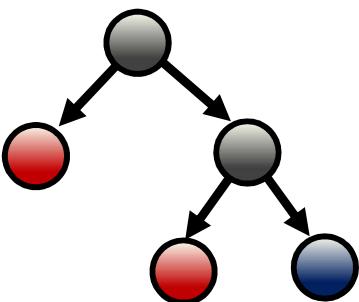
- Bootstrapped samples provide built-in error estimate for each tree
- Create tree based on subset of data
- Measure error for that tree on

Date	Title	Budget	DomesticTotalGross	Director	Rating	Runtime
0	2013-11-22	The Hunger Games: Catching Fire	130000000	424668047	Francis Lawrence	PG-13 146
1	2013-05-03	Iron Man 3	200000000	409013994	Shane Black	PG-13 129
2	2013-11-22	Frozen	150000000	400738009	Chris Buck/Jennifer Lee	PG 108
3	2013-07-03	Despicable Me 2	76000000	368061265	Pierre Coffin/Chris Renaud	PG 98
4	2013-06-14	Man of Steel	225000000	291045518	Zack Snyder	PG-13 143
5	2013-10-04	Gravity	100000000	274092705	Alfonso Cuaron	PG-13 91
6	2013-06-21	Monsters University	Nan	268492764	Dan Scanlon	G 107
7	2013-12-13	The Hobbit: The Desolation of Smaug	Nan	258366855	Peter Jackson	PG-13 161
8	2013-05-24	Fast & Furious 6	160000000	238679850	Justin Lin	PG-13 130
9	2013-03-08	Oz The Great and Powerful	215000000	234911825	Sam Raimi	PG 127
10	2013-05-16	Star Trek Into Darkness	190000000	228778661	J.J. Abrams	PG-13 123
11	2013-11-08	Thor: The Dark World	170000000	206362140	Alan Taylor	PG-13 120
12	2013-06-21	World War Z	190000000	202359711	Marc Forster	PG-13 116
13	2013-03-22	The Croods	135000000	187168425	Kirk De Micco/Chris Sanders	PG 98
14	2013-06-28	The Heat	43000000	159582188	Paul Feig	R 117
15	2013-08-07	We're the Millers	37000000	150394119	Rawson Marshall Thurber	R 110
16	2013-12-13	American Hustle	40000000	150117807	David O. Russell	R 138
17	2013-05-10	The Great Gatsby	105000000	144840419	Baz Luhrmann	PG-13 143

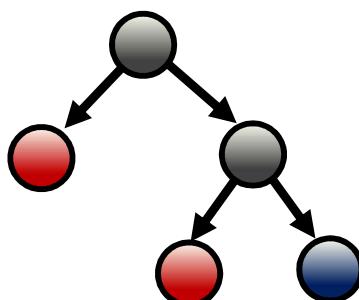


Bagging error calculations

Date	Title	Budget	DomesticTotalGross	Director	Rating	Runtime
0	2013-11-22 The Hunger Games: Catching Fire	130000000	424668047	Francis Lawrence	PG-13	146
1	2013-05-03 Iron Man 3	200000000	409013994	Shane Black	PG-13	129
2	2013-11-22 Frozen	150000000	400738009	Chris Buck/Jennifer Lee	PG	108
3	2013-07-03 Despicable Me 2	76000000	368061265	Pierre Coffin/Chris Renaud	PG	98
4	2013-06-14 Man of Steel	225000000	291045518	Zack Snyder	PG-13	143
5	2013-10-04 Gravity	100000000	274092705	Alfonso Cuaron	PG-13	91
6	2013-06-21 Monsters University	Nan	268492764	Dan Scanlon	G	107
7	2013-12-13 The Hobbit: The Desolation of Smaug	Nan	258366855	Peter Jackson	PG-13	161
8	2013-05-24 Fast & Furious 6	160000000	238679850	Justin Lin	PG-13	130
9	2013-03-08 Oz The Great and Powerful	215000000	234911825	Sam Raimi	PG	127
10	2013-05-16 Star Trek Into Darkness	190000000	228778661	J.J. Abrams	PG-13	123
11	2013-11-08 Thor: The Dark World	170000000	206362140	Alan Taylor	PG-13	120
12	2013-06-21 World War Z	190000000	202359711	Marc Forster	PG-13	116
13	2013-03-22 The Croods	135000000	187168425	Kirk De Micco/Chris Sanders	PG	98
14	2013-06-28 The Heat	43000000	159582188	Paul Feig	R	117
15	2013-08-07 We're the Millers	37000000	150394119	Rawson Marshall Thurber	R	110
16	2013-12-13 American Hustle	40000000	150117807	David O. Russell	R	138
17	2013-05-10 The Great Gatsby	105000000	144840419	Baz Luhrmann	PG-13	143



- Bootstrapped samples provide built-in error estimate for each tree
- Create tree based on subset of data
- Measure error for that tree on unused samples
- Called "Out-of-Bag" error

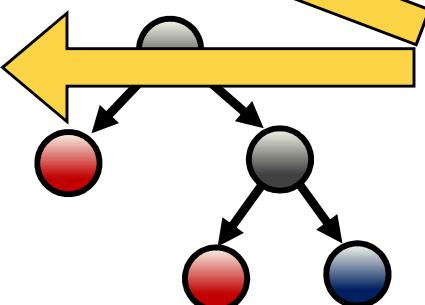
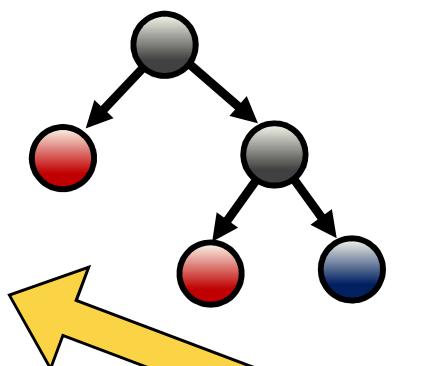


Date	Title	Budget	DomesticTotalGross	Director	Rating	Runtime
0	2013-11-22 The Hunger Games: Catching Fire	130000000	424668047	Francis Lawrence	PG-13	146
1	2013-05-03 Iron Man 3	200000000	409013994	Shane Black	PG-13	129
2	2013-11-22 Frozen	150000000	400738009	Chris Buck/Jennifer Lee	PG	108
3	2013-07-03 Despicable Me 2	76000000	368061265	Pierre Coffin/Chris Renaud	PG	98
4	2013-06-14 Man of Steel	225000000	291045518	Zack Snyder	PG-13	143
5	2013-10-04 Gravity	100000000	274092705	Alfonso Cuaron	PG-13	91
6	2013-06-21 Monsters University	Nan	268492764	Dan Scanlon	G	107
7	2013-12-13 The Hobbit: The Desolation of Smaug	Nan	258366855	Peter Jackson	PG-13	161
8	2013-05-24 Fast & Furious 6	160000000	238679850	Justin Lin	PG-13	130
9	2013-03-08 Oz The Great and Powerful	215000000	234911825	Sam Raimi	PG	127
10	2013-05-16 Star Trek Into Darkness	190000000	228778661	J.J. Abrams	PG-13	123
11	2013-11-08 Thor: The Dark World	170000000	206362140	Alan Taylor	PG-13	120
12	2013-06-21 World War Z	190000000	202359711	Marc Forster	PG-13	116
13	2013-03-22 The Croods	135000000	187168425	Kirk De Micco/Chris Sanders	PG	98
14	2013-06-28 The Heat	43000000	159582188	Paul Feig	R	117
15	2013-08-07 We're the Millers	37000000	150394119	Rawson Marshall Thurber	R	110
16	2013-12-13 American Hustle	40000000	150117807	David O. Russell	R	138
17	2013-05-10 The Great Gatsby	105000000	144840419	Baz Luhrmann	PG-13	143

Bagging error calculations

Date	Title	Budget	DomesticTotalGross	Director	Rating	Runtime
0	2013-11-22 The Hunger Games: Catching Fire	130000000	424668047	Francis Lawrence	PG-13	146
1	2013-05-03 Iron Man 3	200000000	409013994	Shane Black	PG-13	129
2	2013-11-22 Frozen	150000000	400738009	Chris Buck/Jennifer Lee	PG	108
3	2013-07-03 Despicable Me 2	76000000	368061265	Pierre Coffin/Chris Renaud	PG	98
4	2013-06-14 Man of Steel	225000000	291045518	Zack Snyder	PG-13	143
5	2013-10-04 Gravity	100000000	274092705	Alfonso Cuaron	PG-13	91
6	2013-06-21 Monsters University	Nan	268492764	Dan Scanlon	G	107
7	2013-12-13 The Hobbit: The Desolation of Smaug	Nan	258366855	Peter Jackson	PG-13	161
8	2013-05-24 Fast & Furious 6	160000000	238679850	Justin Lin	PG-13	130
9	2013-03-08 Oz The Great and Powerful	215000000	234911825	Sam Raimi	PG	127
10	2013-05-16 Star Trek Into Darkness	190000000	228779861	J.J. Abrams	PG-13	123
11	2013-11-08 Thor: The Dark World	170000000	206362140	Alan Taylor	PG-13	120
12	2013-06-21 World War Z	190000000	202359711	Marc Forster	PG-13	116
13	2013-03-22 The Croods	135000000	187168425	Kirk De Micco/Chris Sanders	PG	98
14	2013-06-28 The Heat	43000000	159582188	Paul Feig	R	117
15	2013-08-07 We're the Millers	37000000	150394119	Rawson Marshall Thurber	R	110
16	2013-12-13 American Hustle	40000000	150117807	David O. Russell	R	138
17	2013-05-10 The Great Gatsby	105000000	144840419	Baz Luhrmann	PG-13	143

Date	Title	Budget	DomesticTotalGross	Director	Rating	Runtime
0	2013-11-22 The Hunger Games: Catching Fire	130000000	424668047	Francis Lawrence	PG-13	146
1	2013-05-03 Iron Man 3	200000000	409013994	Shane Black	PG-13	129
2	2013-11-22 Frozen	150000000	400738009	Chris Buck/Jennifer Lee	PG	108
3	2013-07-03 Despicable Me 2	76000000	368061265	Pierre Coffin/Chris Renaud	PG	98
4	2013-06-14 Man of Steel	225000000	291045518	Zack Snyder	PG-13	143
5	2013-10-04 Gravity	100000000	274092705	Alfonso Cuaron	PG-13	91
6	2013-06-21 Monsters University	Nan	268492764	Dan Scanlon	G	107
7	2013-12-13 The Hobbit: The Desolation of Smaug	Nan	258366855	Peter Jackson	PG-13	161
8	2013-05-24 Fast & Furious 6	160000000	238679850	Justin Lin	PG-13	130
9	2013-03-08 Oz The Great and Powerful	215000000	234911825	Sam Raimi	PG	127
10	2013-05-16 Star Trek Into Darkness	190000000	228779861	J.J. Abrams	PG-13	123
11	2013-11-08 Thor: The Dark World	170000000	206362140	Alan Taylor	PG-13	120
12	2013-06-21 World War Z	190000000	202359711	Marc Forster	PG-13	116
13	2013-03-22 The Croods	135000000	187168425	Kirk De Micco/Chris Sanders	PG	98
14	2013-06-28 The Heat	43000000	159582188	Paul Feig	R	117
15	2013-08-07 We're the Millers	37000000	150394119	Rawson Marshall Thurber	R	110
16	2013-12-13 American Hustle	40000000	150117807	David O. Russell	R	138
17	2013-05-10 The Great Gatsby	105000000	144840419	Baz Luhrmann	PG-13	143

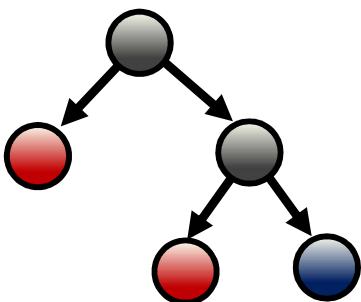


- Bootstrapped samples provide built-in error estimate for each tree
- Create tree based on subset of data
- Measure error for that tree on unused samples

- C " " " " " - C P " "

Bagging error calculations

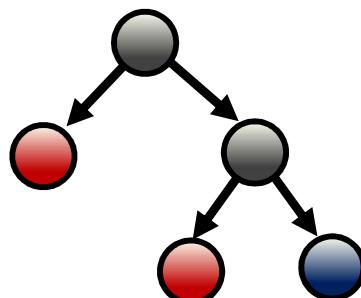
Date	Title	Budget	DomesticTotalGross	Director	Rating	Runtime
0	2013-11-22	The Hunger Games: Catching Fire	130000000	424668047	Francis Lawrence	PG-13 146
1	2013-05-03	Iron Man 3	200000000	409013994	Shane Black	PG-13 129
2	2013-11-22	Frozen	150000000	400738009	Chris Buck/Jennifer Lee	PG 108
3	2013-07-03	Despicable Me 2	76000000	368061265	Pierre Coffin/Chris Renaud	PG 98
4	2013-06-14	Man of Steel	225000000	291045518	Zack Snyder	PG-13 143
5	2013-10-04	Gravity	100000000	274092705	Alfonso Cuaron	PG-13 91
6	2013-06-21	Monsters University	Nan	268492764	Dan Scanlon	G 107
7	2013-12-13	The Hobbit: The Desolation of Smaug	Nan	258366855	Peter Jackson	PG-13 161
8	2013-05-24	Fast & Furious 6	160000000	238679850	Justin Lin	PG-13 130
9	2013-03-08	Oz The Great and Powerful	215000000	234911825	Sam Raimi	PG 127
10	2013-05-16	Star Trek Into Darkness	190000000	228778661	J.J. Abrams	PG-13 123
11	2013-11-08	Thor: The Dark World	170000000	206362140	Alan Taylor	PG-13 120
12	2013-06-21	World War Z	190000000	202359711	Marc Forster	PG-13 116
13	2013-03-22	The Croods	135000000	187168425	Kirk De Micco/Chris Sanders	PG 98
14	2013-06-28	The Heat	43000000	159582188	Paul Feig	R 117
15	2013-08-07	We're the Millers	37000000	150394119	Rawson Marshall Thurber	R 110
16	2013-12-13	American Hustle	40000000	150117807	David O. Russell	R 138
17	2013-05-10	The Great Gatsby	105000000	144840419	Baz Luhrmann	PG-13 143



- Bootstrapped samples provide built-in error estimate for each tree
- Create tree based on subset of data
- Measure error for that tree on unused samples

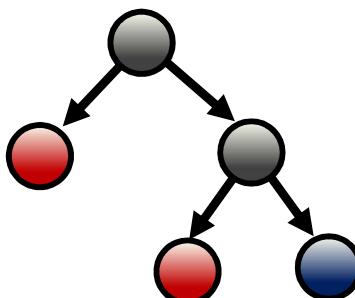
Called "Out-of-Bag" error

Date	Title	Budget	DomesticTotalGross	Director	Rating	Runtime
0	2013-11-22	The Hunger Games: Catching Fire	130000000	424668047	Francis Lawrence	PG-13 146
1	2013-05-03	Iron Man 3	200000000	409013994	Shane Black	PG-13 129
2	2013-11-22	Frozen	150000000	400738009	Chris Buck/Jennifer Lee	PG 108
3	2013-07-03	Despicable Me 2	76000000	368061265	Pierre Coffin/Chris Renaud	PG 98
4	2013-06-14	Man of Steel	225000000	291045518	Zack Snyder	PG-13 143
5	2013-10-04	Gravity	100000000	274092705	Alfonso Cuaron	PG-13 91
6	2013-06-21	Monsters University	Nan	268492764	Dan Scanlon	G 107
7	2013-12-13	The Hobbit: The Desolation of Smaug	Nan	258366855	Peter Jackson	PG-13 161
8	2013-05-24	Fast & Furious 6	160000000	238679850	Justin Lin	PG-13 130
9	2013-03-08	Oz The Great and Powerful	215000000	234911825	Sam Raimi	PG 127
10	2013-05-16	Star Trek Into Darkness	190000000	228778661	J.J. Abrams	PG-13 123
11	2013-11-08	Thor: The Dark World	170000000	206362140	Alan Taylor	PG-13 120
12	2013-06-21	World War Z	190000000	202359711	Marc Forster	PG-13 116
13	2013-03-22	The Croods	135000000	187168425	Kirk De Micco/Chris Sanders	PG 98
14	2013-06-28	The Heat	43000000	159582188	Paul Feig	R 117
15	2013-08-07	We're the Millers	37000000	150394119	Rawson Marshall Thurber	R 110
16	2013-12-13	American Hustle	40000000	150117807	David O. Russell	R 138
17	2013-05-10	The Great Gatsby	105000000	144840419	Baz Luhrmann	PG-13 143



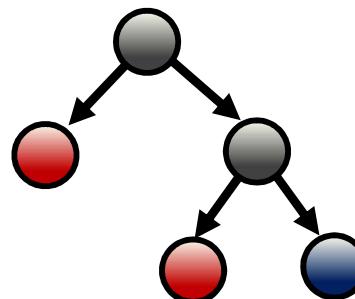
Calculation of Feature Importance

Date	Title	Budget	DomesticTotalGross	Director	Rating	Runtime
0	2013-11-22 The Hunger Games: Catching Fire	130000000	424668047	Francis Lawrence	PG-13	146
1	2013-05-03 Iron Man 3	200000000	409013994	Shane Black	PG-13	129
2	2013-11-22 Frozen	150000000	400738009	Chris Buck/Jennifer Lee	PG	108
3	2013-07-03 Despicable Me 2	76000000	368061265	Pierre Coffin/Chris Renaud	PG	98
4	2013-06-14 Man of Steel	225000000	291045518	Zack Snyder	PG-13	143
5	2013-10-04 Gravity	100000000	274092705	Alfonso Cuaron	PG-13	91
6	2013-06-21 Monsters University	Nan	268492764	Dan Scanlon	G	107
7	2013-12-13 The Hobbit: The Desolation of Smaug	Nan	258366855	Peter Jackson	PG-13	161
8	2013-05-24 Fast & Furious 6	160000000	238679850	Justin Lin	PG-13	130
9	2013-03-08 Oz The Great and Powerful	215000000	234911825	Sam Raimi	PG	127
10	2013-05-16 Star Trek Into Darkness	190000000	228778661	J.J. Abrams	PG-13	123
11	2013-11-08 Thor: The Dark World	170000000	206362140	Alan Taylor	PG-13	120
12	2013-06-21 World War Z	190000000	202359711	Marc Forster	PG-13	116
13	2013-03-22 The Croods	135000000	187168425	Kirk De Micco/Chris Sanders	PG	98
14	2013-06-28 The Heat	43000000	159582188	Paul Feig	R	117
15	2013-08-07 We're the Millers	37000000	150394119	Rawson Marshall Thurber	R	110
16	2013-12-13 American Hustle	40000000	150117807	David O. Russell	R	138
17	2013-05-10 The Great Gatsby	105000000	144840419	Baz Luhrmann	PG-13	143



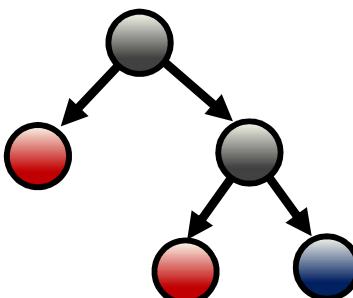
- Fitting a bagged model doesn't produce coefficients like logistic regression

Date	Title	Budget	DomesticTotalGross	Director	Rating	Runtime
0	2013-11-22 The Hunger Games: Catching Fire	130000000	424668047	Francis Lawrence	PG-13	146
1	2013-05-03 Iron Man 3	200000000	409013994	Shane Black	PG-13	129
2	2013-11-22 Frozen	150000000	400738009	Chris Buck/Jennifer Lee	PG	108
3	2013-07-03 Despicable Me 2	76000000	368061265	Pierre Coffin/Chris Renaud	PG	98
4	2013-06-14 Man of Steel	225000000	291045518	Zack Snyder	PG-13	143
5	2013-10-04 Gravity	100000000	274092705	Alfonso Cuaron	PG-13	91
6	2013-06-21 Monsters University	Nan	268492764	Dan Scanlon	G	107
7	2013-12-13 The Hobbit: The Desolation of Smaug	Nan	258366855	Peter Jackson	PG-13	161
8	2013-05-24 Fast & Furious 6	160000000	238679850	Justin Lin	PG-13	130
9	2013-03-08 Oz The Great and Powerful	215000000	234911825	Sam Raimi	PG	127
10	2013-05-16 Star Trek Into Darkness	190000000	228778661	J.J. Abrams	PG-13	123
11	2013-11-08 Thor: The Dark World	170000000	206362140	Alan Taylor	PG-13	120
12	2013-06-21 World War Z	190000000	202359711	Marc Forster	PG-13	116
13	2013-03-22 The Croods	135000000	187168425	Kirk De Micco/Chris Sanders	PG	98
14	2013-06-28 The Heat	43000000	159582188	Paul Feig	R	117
15	2013-08-07 We're the Millers	37000000	150394119	Rawson Marshall Thurber	R	110
16	2013-12-13 American Hustle	40000000	150117807	David O. Russell	R	138
17	2013-05-10 The Great Gatsby	105000000	144840419	Baz Luhrmann	PG-13	143



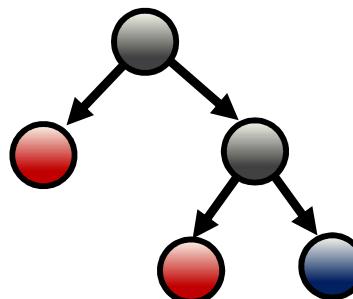
Calculation of Feature Importance

	Date	Title	Budget	DomesticTotalGross	Director	Rating	Runtime
0	2013-11-22	The Hunger Games: Catching Fire	130000000	424668047	Francis Lawrence	PG-13	146
1	2013-05-03	Iron Man 3	200000000	409013994	Shane Black	PG-13	129
2	2013-11-22	Frozen	150000000	400738009	Chris Buck/Jennifer Lee	PG	108
3	2013-07-03	Despicable Me 2	76000000	368061265	Pierre Coffin/Chris Renaud	PG	98
4	2013-06-14	Man of Steel	225000000	291045518	Zack Snyder	PG-13	143
5	2013-10-04	Gravity	100000000	274092705	Alfonso Cuaron	PG-13	91
6	2013-06-21	Monsters University	Nan	268492764	Dan Scanlon	G	107
7	2013-12-13	The Hobbit: The Desolation of Smaug	Nan	258366855	Peter Jackson	PG-13	161
8	2013-05-24	Fast & Furious 6	160000000	238679850	Justin Lin	PG-13	130
9	2013-03-08	Oz The Great and Powerful	215000000	234911825	Sam Raimi	PG	127
10	2013-05-16	Star Trek Into Darkness	190000000	228778661	J.J. Abrams	PG-13	123
11	2013-11-08	Thor: The Dark World	170000000	206362140	Alan Taylor	PG-13	120
12	2013-06-21	World War Z	190000000	202359711	Marc Forster	PG-13	116
13	2013-03-22	The Croods	135000000	187168425	Kirk De Micco/Chris Sanders	PG	98
14	2013-06-28	The Heat	43000000	159582188	Paul Feig	R	117
15	2013-08-07	We're the Millers	37000000	150394119	Rawson Marshall Thurber	R	110
16	2013-12-13	American Hustle	40000000	150117807	David O. Russell	R	138
17	2013-05-10	The Great Gatsby	105000000	144840419	Baz Luhrmann	PG-13	143



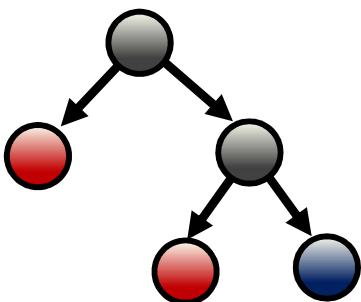
- Fitting a bagged model doesn't produce coefficients like logistic regression
- Instead, feature importances are estimated using oob error

	Date	Title	Budget	DomesticTotalGross	Director	Rating	Runtime
0	2013-11-22	The Hunger Games: Catching Fire	130000000	424668047	Francis Lawrence	PG-13	146
1	2013-05-03	Iron Man 3	200000000	409013994	Shane Black	PG-13	129
2	2013-11-22	Frozen	150000000	400738009	Chris Buck/Jennifer Lee	PG	108
3	2013-07-03	Despicable Me 2	76000000	368061265	Pierre Coffin/Chris Renaud	PG	98
4	2013-06-14	Man of Steel	225000000	291045518	Zack Snyder	PG-13	143
5	2013-10-04	Gravity	100000000	274092705	Alfonso Cuaron	PG-13	91
6	2013-06-21	Monsters University	Nan	268492764	Dan Scanlon	G	107
7	2013-12-13	The Hobbit: The Desolation of Smaug	Nan	258366855	Peter Jackson	PG-13	161
8	2013-05-24	Fast & Furious 6	160000000	238679850	Justin Lin	PG-13	130
9	2013-03-08	Oz The Great and Powerful	215000000	234911825	Sam Raimi	PG	127
10	2013-05-16	Star Trek Into Darkness	190000000	228778661	J.J. Abrams	PG-13	123
11	2013-11-08	Thor: The Dark World	170000000	206362140	Alan Taylor	PG-13	120
12	2013-06-21	World War Z	190000000	202359711	Marc Forster	PG-13	116
13	2013-03-22	The Croods	135000000	187168425	Kirk De Micco/Chris Sanders	PG	98
14	2013-06-28	The Heat	43000000	159582188	Paul Feig	R	117
15	2013-08-07	We're the Millers	37000000	150394119	Rawson Marshall Thurber	R	110
16	2013-12-13	American Hustle	40000000	150117807	David O. Russell	R	138
17	2013-05-10	The Great Gatsby	105000000	144840419	Baz Luhrmann	PG-13	143

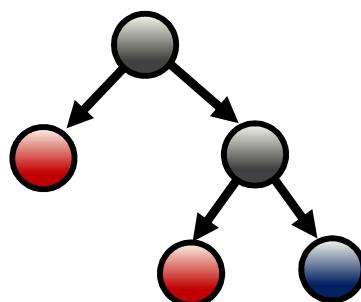


Calculation of Feature Importance

Date	Title	Budget	DomesticTotalGross	Director	Rating	Runtime
0	2013-11-22 The Hunger Games: Catching Fire	130000000	424668047	Francis Lawrence	PG-13	146
1	2013-05-03 Iron Man 3	200000000	409013994	Shane Black	PG-13	129
2	2013-11-22 Frozen	150000000	400738009	Chris Buck/Jennifer Lee	PG	108
3	2013-07-03 Despicable Me 2	76000000	368061265	Pierre Coffin/Chris Renaud	PG	98
4	2013-06-14 Man of Steel	225000000	291045518	Zack Snyder	PG-13	143
5	2013-10-04 Gravity	100000000	274092705	Alfonso Cuaron	PG-13	91
6	2013-06-21 Monsters University	Nan	268492764	Dan Scanlon	G	107
7	2013-12-13 The Hobbit: The Desolation of Smaug	Nan	258366855	Peter Jackson	PG-13	161
8	2013-05-24 Fast & Furious 6	160000000	238679850	Justin Lin	PG-13	130
9	2013-03-08 Oz The Great and Powerful	215000000	234911825	Sam Raimi	PG	127
10	2013-05-16 Star Trek Into Darkness	190000000	228778661	J.J. Abrams	PG-13	123
11	2013-11-08 Thor: The Dark World	170000000	206362140	Alan Taylor	PG-13	120
12	2013-06-21 World War Z	190000000	202359711	Marc Forster	PG-13	116
13	2013-03-22 The Croods	135000000	187168425	Kirk De Micco/Chris Sanders	PG	98
14	2013-06-28 The Heat	43000000	159582188	Paul Feig	R	117
15	2013-08-07 We're the Millers	37000000	150394119	Rawson Marshall Thurber	R	110
16	2013-12-13 American Hustle	40000000	150117807	David O. Russell	R	138
17	2013-05-10 The Great Gatsby	105000000	144840419	Baz Luhrmann	PG-13	143



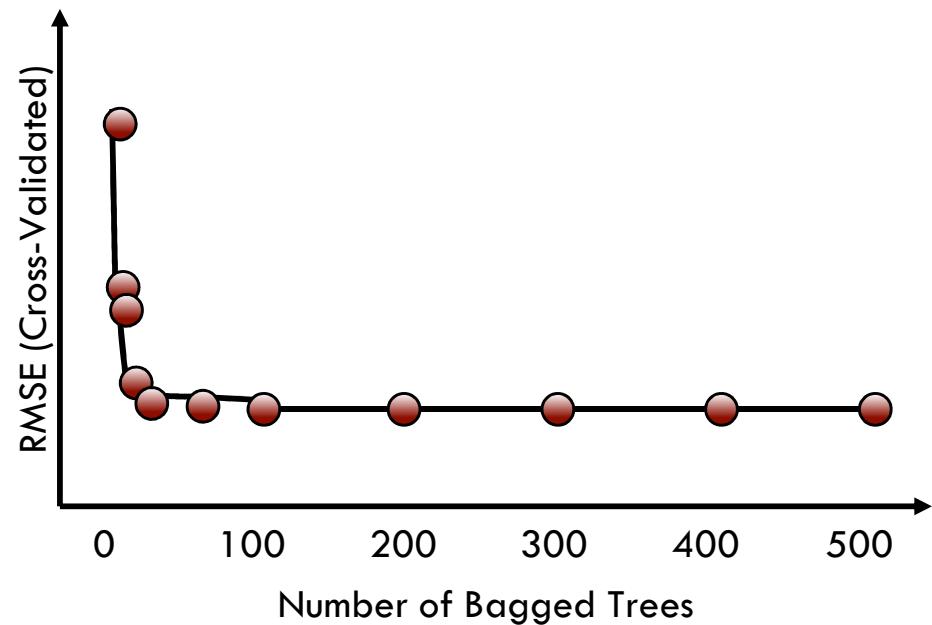
- Fitting a bagged model doesn't produce coefficients like logistic regression
- Instead, feature importances are estimated using oob error
- Randomly permute data for particular feature and measure change in accuracy



Date	Title	Budget	DomesticTotalGross	Director	Rating	Runtime
0	2013-11-22 The Hunger Games: Catching Fire	130000000	424668047	Francis Lawrence	PG-13	146
1	2013-05-03 Iron Man 3	200000000	409013994	Shane Black	PG-13	129
2	2013-11-22 Frozen	150000000	400738009	Chris Buck/Jennifer Lee	PG	108
3	2013-07-03 Despicable Me 2	76000000	368061265	Pierre Coffin/Chris Renaud	PG	98
4	2013-06-14 Man of Steel	225000000	291045518	Zack Snyder	PG-13	143
5	2013-10-04 Gravity	100000000	274092705	Alfonso Cuaron	PG-13	91
6	2013-06-21 Monsters University	Nan	268492764	Dan Scanlon	G	107
7	2013-12-13 The Hobbit: The Desolation of Smaug	Nan	258366855	Peter Jackson	PG-13	161
8	2013-05-24 Fast & Furious 6	160000000	238679850	Justin Lin	PG-13	130
9	2013-03-08 Oz The Great and Powerful	215000000	234911825	Sam Raimi	PG	127
10	2013-05-16 Star Trek Into Darkness	190000000	228778661	J.J. Abrams	PG-13	123
11	2013-11-08 Thor: The Dark World	170000000	206362140	Alan Taylor	PG-13	120
12	2013-06-21 World War Z	190000000	202359711	Marc Forster	PG-13	116
13	2013-03-22 The Croods	135000000	187168425	Kirk De Micco/Chris Sanders	PG	98
14	2013-06-28 The Heat	43000000	159582188	Paul Feig	R	117
15	2013-08-07 We're the Millers	37000000	150394119	Rawson Marshall Thurber	R	110
16	2013-12-13 American Hustle	40000000	150117807	David O. Russell	R	138
17	2013-05-10 The Great Gatsby	105000000	144840419	Baz Luhrmann	PG-13	143

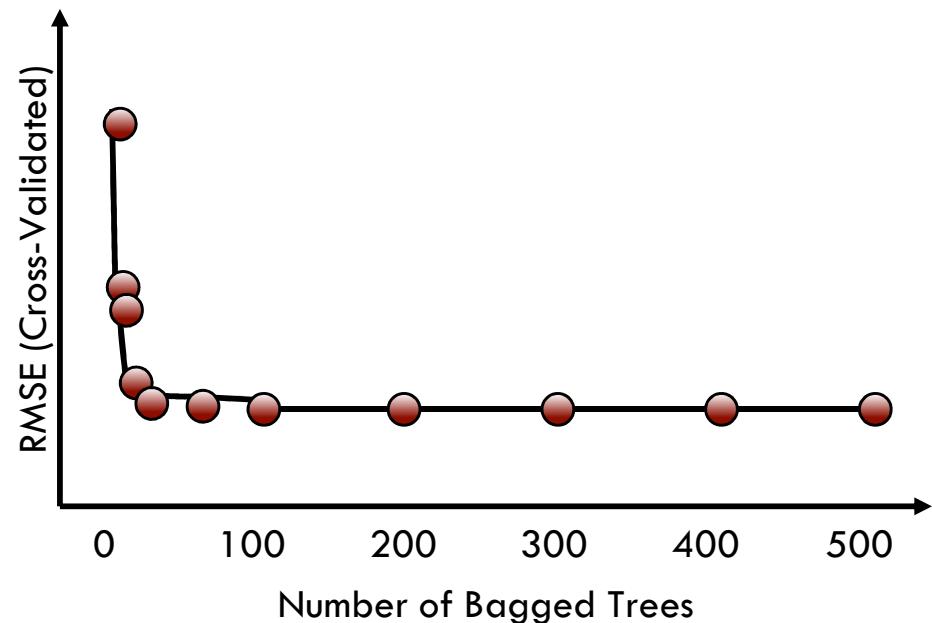
How Many Trees to Fit?

- Bagging performance improvements increase with more trees



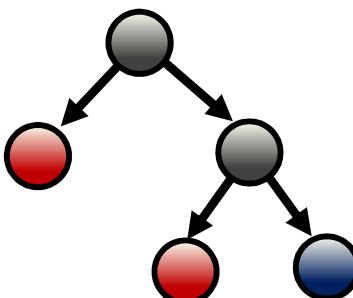
How Many Trees to Fit?

- Bagging performance improvements increase with more trees
- Maximum improvement generally reached ~ 50 trees



Strengths of Bagging

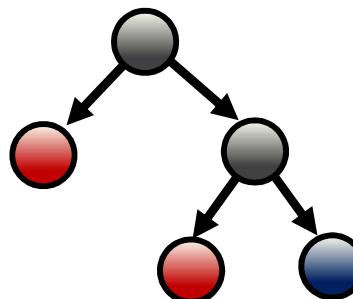
	Date	Title	Budget	DomesticTotalGross	Director	Rating	Runtime
0	2013-11-22	The Hunger Games: Catching Fire	130000000	424668047	Francis Lawrence	PG-13	146
1	2013-05-03	Iron Man 3	200000000	409013994	Shane Black	PG-13	129
2	2013-11-22	Frozen	150000000	400738009	Chris Buck/Jennifer Lee	PG	108
3	2013-07-03	Despicable Me 2	76000000	368061265	Pierre Coffin/Chris Renaud	PG	98
4	2013-06-14	Man of Steel	225000000	291045518	Zack Snyder	PG-13	143
5	2013-10-04	Gravity	100000000	274092705	Alfonso Cuaron	PG-13	91
6	2013-06-21	Monsters University	Nan	268492764	Dan Scanlon	G	107
7	2013-12-13	The Hobbit: The Desolation of Smaug	Nan	258366855	Peter Jackson	PG-13	161
8	2013-05-24	Fast & Furious 6	160000000	238679850	Justin Lin	PG-13	130
9	2013-03-08	Oz The Great and Powerful	215000000	234911825	Sam Raimi	PG	127
10	2013-05-16	Star Trek Into Darkness	190000000	228776661	J.J. Abrams	PG-13	123
11	2013-11-08	Thor: The Dark World	170000000	206362140	Alan Taylor	PG-13	120
12	2013-06-21	World War Z	190000000	202359711	Marc Forster	PG-13	116
13	2013-03-22	The Croods	135000000	187168425	Kirk De Micco/Chris Sanders	PG	98
14	2013-06-28	The Heat	43000000	159582188	Paul Feig	R	117
15	2013-08-07	We're the Millers	37000000	150394119	Rawson Marshall Thurber	R	110
16	2013-12-13	American Hustle	40000000	150117807	David O. Russell	R	138
17	2013-05-10	The Great Gatsby	105000000	144840419	Baz Luhrmann	PG-13	143



Same as decision trees:

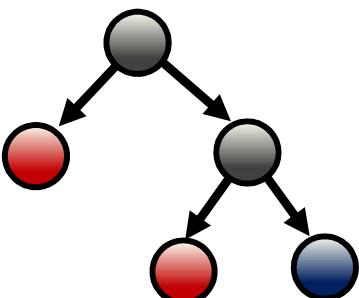
- Easy to interpret and implement

	Date	Title	Budget	DomesticTotalGross	Director	Rating	Runtime
0	2013-11-22	The Hunger Games: Catching Fire	130000000	424668047	Francis Lawrence	PG-13	146
1	2013-05-03	Iron Man 3	200000000	409013994	Shane Black	PG-13	129
2	2013-11-22	Frozen	150000000	400738009	Chris Buck/Jennifer Lee	PG	108
3	2013-07-03	Despicable Me 2	76000000	368061265	Pierre Coffin/Chris Renaud	PG	98
4	2013-06-14	Man of Steel	225000000	291045518	Zack Snyder	PG-13	143
5	2013-10-04	Gravity	100000000	274092705	Alfonso Cuaron	PG-13	91
6	2013-06-21	Monsters University	Nan	268492764	Dan Scanlon	G	107
7	2013-12-13	The Hobbit: The Desolation of Smaug	Nan	258366855	Peter Jackson	PG-13	161
8	2013-05-24	Fast & Furious 6	160000000	238679850	Justin Lin	PG-13	130
9	2013-03-08	Oz The Great and Powerful	215000000	234911825	Sam Raimi	PG	127
10	2013-05-16	Star Trek Into Darkness	190000000	228776661	J.J. Abrams	PG-13	123
11	2013-11-08	Thor: The Dark World	170000000	206362140	Alan Taylor	PG-13	120
12	2013-06-21	World War Z	190000000	202359711	Marc Forster	PG-13	116
13	2013-03-22	The Croods	135000000	187168425	Kirk De Micco/Chris Sanders	PG	98
14	2013-06-28	The Heat	43000000	159582188	Paul Feig	R	117
15	2013-08-07	We're the Millers	37000000	150394119	Rawson Marshall Thurber	R	110
16	2013-12-13	American Hustle	40000000	150117807	David O. Russell	R	138
17	2013-05-10	The Great Gatsby	105000000	144840419	Baz Luhrmann	PG-13	143



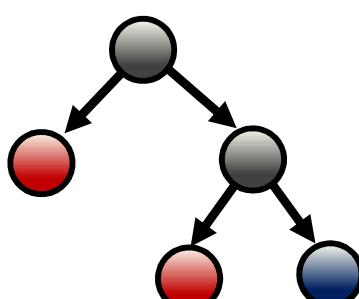
Strengths of Bagging

Date	Title	Budget	DomesticTotalGross	Director	Rating	Runtime
0	2013-11-22 The Hunger Games: Catching Fire	130000000	424668047	Francis Lawrence	PG-13	146
1	2013-05-03 Iron Man 3	200000000	409013994	Shane Black	PG-13	129
2	2013-11-22 Frozen	150000000	400738009	Chris Buck/Jennifer Lee	PG	108
3	2013-07-03 Despicable Me 2	76000000	368061265	Pierre Coffin/Chris Renaud	PG	98
4	2013-06-14 Man of Steel	225000000	291045518	Zack Snyder	PG-13	143
5	2013-10-04 Gravity	100000000	274092705	Alfonso Cuaron	PG-13	91
6	2013-06-21 Monsters University	Nan	268492764	Dan Scanlon	G	107
7	2013-12-13 The Hobbit: The Desolation of Smaug	Nan	258366855	Peter Jackson	PG-13	161
8	2013-05-24 Fast & Furious 6	160000000	238679850	Justin Lin	PG-13	130
9	2013-03-08 Oz The Great and Powerful	215000000	234911825	Sam Raimi	PG	127
10	2013-05-16 Star Trek Into Darkness	190000000	228778661	J.J. Abrams	PG-13	123
11	2013-11-08 Thor: The Dark World	170000000	206362140	Alan Taylor	PG-13	120
12	2013-06-21 World War Z	190000000	202359711	Marc Forster	PG-13	116
13	2013-03-22 The Croods	135000000	187168425	Kirk De Micco/Chris Sanders	PG	98
14	2013-06-28 The Heat	43000000	159582188	Paul Feig	R	117
15	2013-08-07 We're the Millers	37000000	150394119	Rawson Marshall Thurber	R	110
16	2013-12-13 American Hustle	40000000	150117807	David O. Russell	R	138
17	2013-05-10 The Great Gatsby	105000000	144840419	Baz Luhrmann	PG-13	143



Same as decision trees:

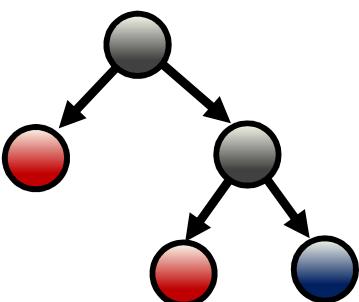
- Easy to interpret and implement
- Heterogeneous input data allowed, no preprocessing required



Date	Title	Budget	DomesticTotalGross	Director	Rating	Runtime
0	2013-11-22 The Hunger Games: Catching Fire	130000000	424668047	Francis Lawrence	PG-13	146
1	2013-05-03 Iron Man 3	200000000	409013994	Shane Black	PG-13	129
2	2013-11-22 Frozen	150000000	400738009	Chris Buck/Jennifer Lee	PG	108
3	2013-07-03 Despicable Me 2	76000000	368061265	Pierre Coffin/Chris Renaud	PG	98
4	2013-06-14 Man of Steel	225000000	291045518	Zack Snyder	PG-13	143
5	2013-10-04 Gravity	100000000	274092705	Alfonso Cuaron	PG-13	91
6	2013-06-21 Monsters University	Nan	268492764	Dan Scanlon	G	107
7	2013-12-13 The Hobbit: The Desolation of Smaug	Nan	258366855	Peter Jackson	PG-13	161
8	2013-05-24 Fast & Furious 6	160000000	238679850	Justin Lin	PG-13	130
9	2013-03-08 Oz The Great and Powerful	215000000	234911825	Sam Raimi	PG	127
10	2013-05-16 Star Trek Into Darkness	190000000	228778661	J.J. Abrams	PG-13	123
11	2013-11-08 Thor: The Dark World	170000000	206362140	Alan Taylor	PG-13	120
12	2013-06-21 World War Z	190000000	202359711	Marc Forster	PG-13	116
13	2013-03-22 The Croods	135000000	187168425	Kirk De Micco/Chris Sanders	PG	98
14	2013-06-28 The Heat	43000000	159582188	Paul Feig	R	117
15	2013-08-07 We're the Millers	37000000	150394119	Rawson Marshall Thurber	R	110
16	2013-12-13 American Hustle	40000000	150117807	David O. Russell	R	138
17	2013-05-10 The Great Gatsby	105000000	144840419	Baz Luhrmann	PG-13	143

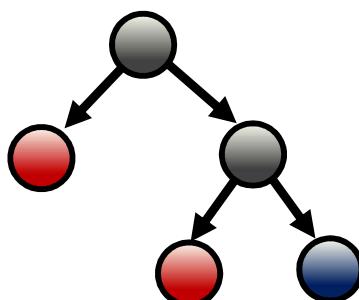
Strengths of Bagging

Date	Title	Budget	DomesticTotalGross	Director	Rating	Runtime
0	2013-11-22 The Hunger Games: Catching Fire	130000000	424668047	Francis Lawrence	PG-13	146
1	2013-05-03 Iron Man 3	200000000	409013994	Shane Black	PG-13	129
2	2013-11-22 Frozen	150000000	400738009	Chris Buck/Jennifer Lee	PG	108
3	2013-07-03 Despicable Me 2	76000000	368061265	Pierre Coffin/Chris Renaud	PG	98
4	2013-06-14 Man of Steel	225000000	291045518	Zack Snyder	PG-13	143
5	2013-10-04 Gravity	100000000	274092705	Alfonso Cuaron	PG-13	91
6	2013-06-21 Monsters University	Nan	268492764	Dan Scanlon	G	107
7	2013-12-13 The Hobbit: The Desolation of Smaug	Nan	258366855	Peter Jackson	PG-13	161
8	2013-05-24 Fast & Furious 6	160000000	238679850	Justin Lin	PG-13	130
9	2013-03-08 Oz The Great and Powerful	215000000	234911825	Sam Raimi	PG	127
10	2013-05-16 Star Trek Into Darkness	190000000	228778661	J.J. Abrams	PG-13	123
11	2013-11-08 Thor: The Dark World	170000000	206362140	Alan Taylor	PG-13	120
12	2013-06-21 World War Z	190000000	202359711	Marc Forster	PG-13	116
13	2013-03-22 The Croods	135000000	187168425	Kirk De Micco/Chris Sanders	PG	98
14	2013-06-28 The Heat	43000000	159582188	Paul Feig	R	117
15	2013-08-07 We're the Millers	37000000	150394119	Rawson Marshall Thurber	R	110
16	2013-12-13 American Hustle	40000000	150117807	David O. Russell	R	138
17	2013-05-10 The Great Gatsby	105000000	144840419	Baz Luhrmann	PG-13	143



Same as decision trees:

- Easy to interpret and implement
- Heterogeneous input data allowed, no preprocessing required



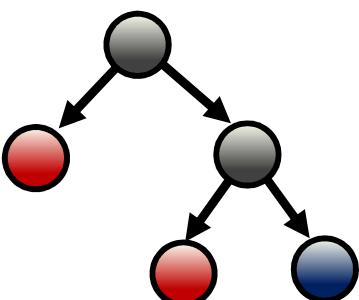
Specific to bagging:

- Less variability than decision trees

Date	Title	Budget	DomesticTotalGross	Director	Rating	Runtime
0	2013-11-22 The Hunger Games: Catching Fire	130000000	424668047	Francis Lawrence	PG-13	146
1	2013-05-03 Iron Man 3	200000000	409013994	Shane Black	PG-13	129
2	2013-11-22 Frozen	150000000	400738009	Chris Buck/Jennifer Lee	PG	108
3	2013-07-03 Despicable Me 2	76000000	368061265	Pierre Coffin/Chris Renaud	PG	98
4	2013-06-14 Man of Steel	225000000	291045518	Zack Snyder	PG-13	143
5	2013-10-04 Gravity	100000000	274092705	Alfonso Cuaron	PG-13	91
6	2013-06-21 Monsters University	Nan	268492764	Dan Scanlon	G	107
7	2013-12-13 The Hobbit: The Desolation of Smaug	Nan	258366855	Peter Jackson	PG-13	161
8	2013-05-24 Fast & Furious 6	160000000	238679850	Justin Lin	PG-13	130
9	2013-03-08 Oz The Great and Powerful	215000000	234911825	Sam Raimi	PG	127
10	2013-05-16 Star Trek Into Darkness	190000000	228778661	J.J. Abrams	PG-13	123
11	2013-11-08 Thor: The Dark World	170000000	206362140	Alan Taylor	PG-13	120
12	2013-06-21 World War Z	190000000	202359711	Marc Forster	PG-13	116
13	2013-03-22 The Croods	135000000	187168425	Kirk De Micco/Chris Sanders	PG	98
14	2013-06-28 The Heat	43000000	159582188	Paul Feig	R	117
15	2013-08-07 We're the Millers	37000000	150394119	Rawson Marshall Thurber	R	110
16	2013-12-13 American Hustle	40000000	150117807	David O. Russell	R	138
17	2013-05-10 The Great Gatsby	105000000	144840419	Baz Luhrmann	PG-13	143

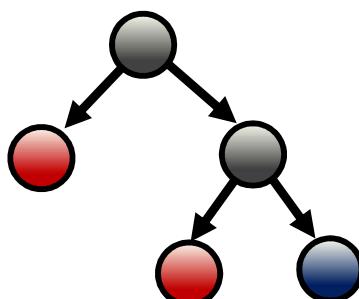
Strengths of Bagging

Date	Title	Budget	DomesticTotalGross	Director	Rating	Runtime
0	2013-11-22	The Hunger Games: Catching Fire	130000000	424668047	Francis Lawrence	PG-13 146
1	2013-05-03	Iron Man 3	200000000	409013994	Shane Black	PG-13 129
2	2013-11-22	Frozen	150000000	400738009	Chris Buck/Jennifer Lee	PG 108
3	2013-07-03	Despicable Me 2	76000000	368061265	Pierre Coffin/Chris Renaud	PG 98
4	2013-06-14	Man of Steel	225000000	291045518	Zack Snyder	PG-13 143
5	2013-10-04	Gravity	100000000	274092705	Alfonso Cuaron	PG-13 91
6	2013-06-21	Monsters University	Nan	268492764	Dan Scanlon	G 107
7	2013-12-13	The Hobbit: The Desolation of Smaug	Nan	258366855	Peter Jackson	PG-13 161
8	2013-05-24	Fast & Furious 6	160000000	238679850	Justin Lin	PG-13 130
9	2013-03-08	Oz The Great and Powerful	215000000	234911825	Sam Raimi	PG 127
10	2013-05-16	Star Trek Into Darkness	190000000	228778661	J.J. Abrams	PG-13 123
11	2013-11-08	Thor: The Dark World	170000000	206362140	Alan Taylor	PG-13 120
12	2013-06-21	World War Z	190000000	202359711	Marc Forster	PG-13 116
13	2013-03-22	The Croods	135000000	187168425	Kirk De Micco/Chris Sanders	PG 98
14	2013-06-28	The Heat	43000000	159582188	Paul Feig	R 117
15	2013-08-07	We're the Millers	37000000	150394119	Rawson Marshall Thurber	R 110
16	2013-12-13	American Hustle	40000000	150117807	David O. Russell	R 138
17	2013-05-10	The Great Gatsby	105000000	144840419	Baz Luhrmann	PG-13 143



Same as decision trees:

- Easy to interpret and implement
- Heterogeneous input data allowed, no preprocessing required



Specific to bagging:

- Less variability than decision trees
- Can grow trees in parallel

Date	Title	Budget	DomesticTotalGross	Director	Rating	Runtime
0	2013-11-22	The Hunger Games: Catching Fire	130000000	424668047	Francis Lawrence	PG-13 146
1	2013-05-03	Iron Man 3	200000000	409013994	Shane Black	PG-13 129
2	2013-11-22	Frozen	150000000	400738009	Chris Buck/Jennifer Lee	PG 108
3	2013-07-03	Despicable Me 2	76000000	368061265	Pierre Coffin/Chris Renaud	PG 98
4	2013-06-14	Man of Steel	225000000	291045518	Zack Snyder	PG-13 143
5	2013-10-04	Gravity	100000000	274092705	Alfonso Cuaron	PG-13 91
6	2013-06-21	Monsters University	Nan	268492764	Dan Scanlon	G 107
7	2013-12-13	The Hobbit: The Desolation of Smaug	Nan	258366855	Peter Jackson	PG-13 161
8	2013-05-24	Fast & Furious 6	160000000	238679850	Justin Lin	PG-13 130
9	2013-03-08	Oz The Great and Powerful	215000000	234911825	Sam Raimi	PG 127
10	2013-05-16	Star Trek Into Darkness	190000000	228778661	J.J. Abrams	PG-13 123
11	2013-11-08	Thor: The Dark World	170000000	206362140	Alan Taylor	PG-13 120
12	2013-06-21	World War Z	190000000	202359711	Marc Forster	PG-13 116
13	2013-03-22	The Croods	135000000	187168425	Kirk De Micco/Chris Sanders	PG 98
14	2013-06-28	The Heat	43000000	159582188	Paul Feig	R 117
15	2013-08-07	We're the Millers	37000000	150394119	Rawson Marshall Thurber	R 110
16	2013-12-13	American Hustle	40000000	150117807	David O. Russell	R 138
17	2013-05-10	The Great Gatsby	105000000	144840419	Baz Luhrmann	PG-13 143

BaggingClassifier: The Syntax

Import the class containing the classification method

```
from sklearn.ensemble import BaggingClassifier
```

BaggingClassifier: The Syntax

Import the class containing the classification method

```
from sklearn.ensemble import BaggingClassifier
```

Create an instance of the class

```
BC = BaggingClassifier(n_estimators=50)
```

BaggingClassifier: The Syntax

Import the class containing the classification method

```
from sklearn.ensemble import BaggingClassifier
```

Create an instance of the class

```
BC = BaggingClassifier(n_estimators=50)
```

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

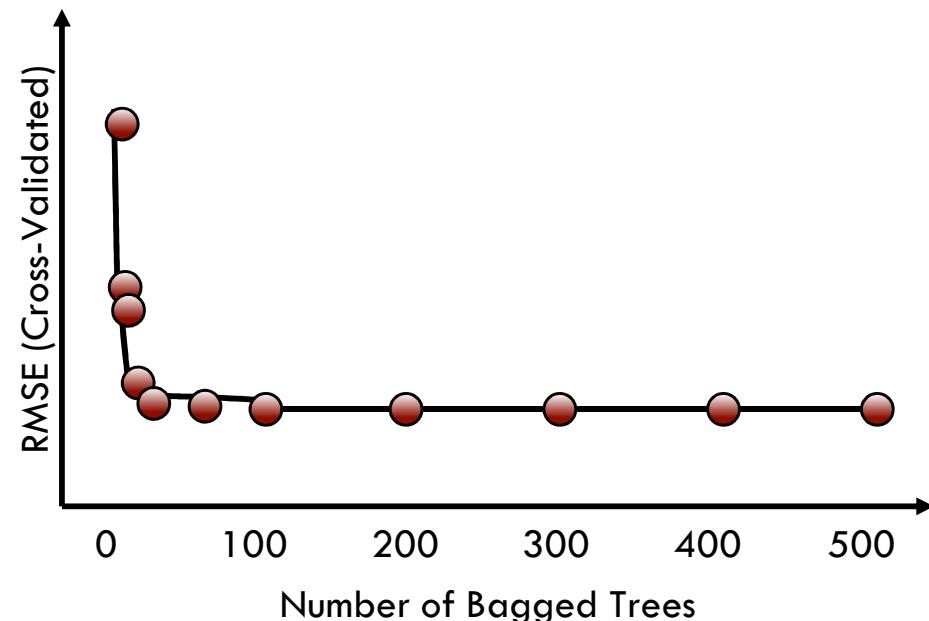
```
BC = BC.fit(X_train, y_train)  
y_predict = BC.predict(X_test)
```

Tune parameters with cross-validation. Use BaggingRegressor for regression.

Reduction in Variance Due to Bagging

- For n independent trees, each with variance σ^2 , the bagged variance is:

$$\frac{\sigma^2}{n}$$



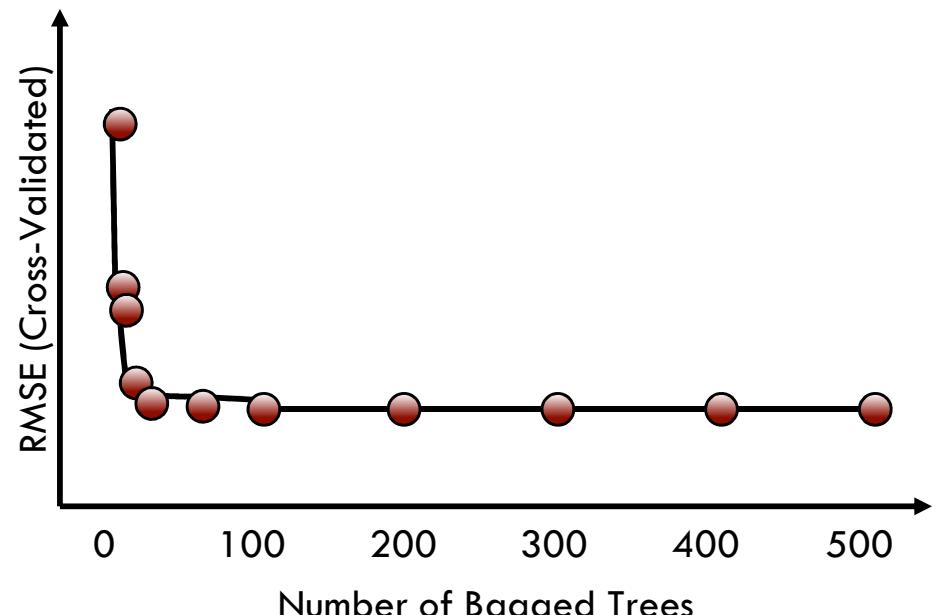
Reduction in Variance Due to Bagging

- For n independent trees, each with variance σ^2 , the bagged variance is:

$$\frac{\sigma^2}{n}$$

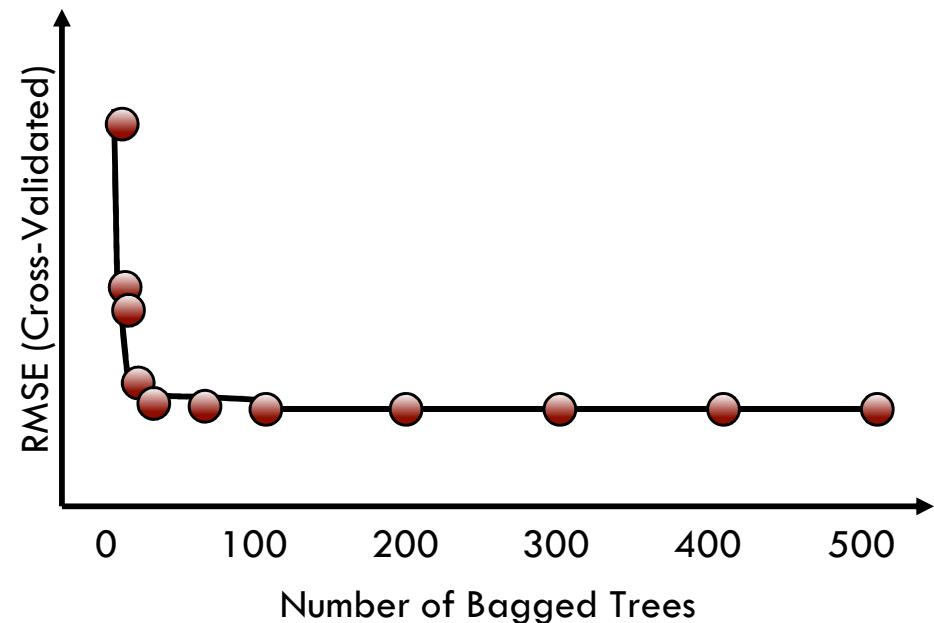
- However, bootstrap samples are correlated (ρ):

$$\rho\sigma^2 + \frac{1 - \rho}{n}\sigma^2$$



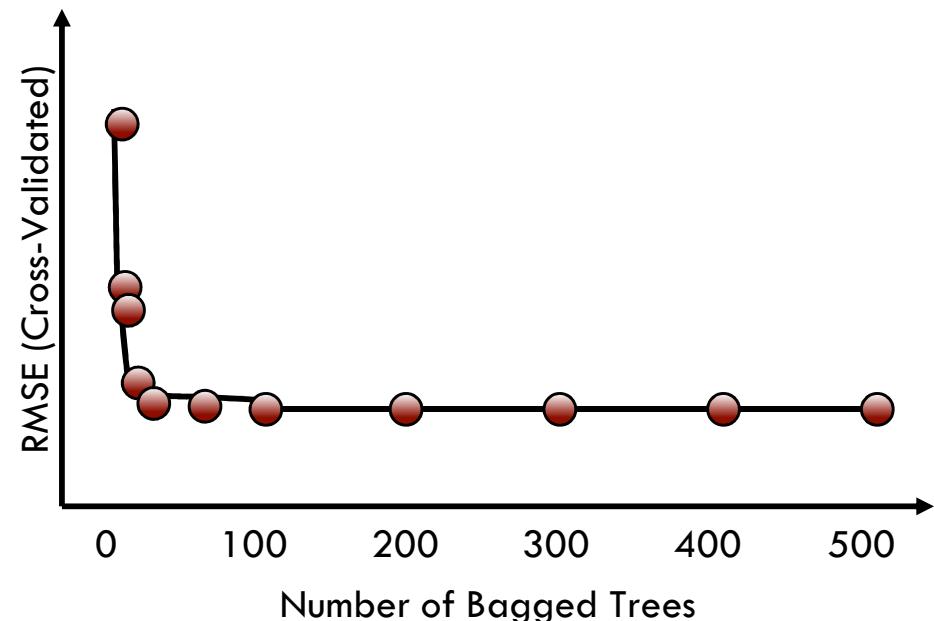
Introducing More Randomness

- Solution: further de-correlate trees



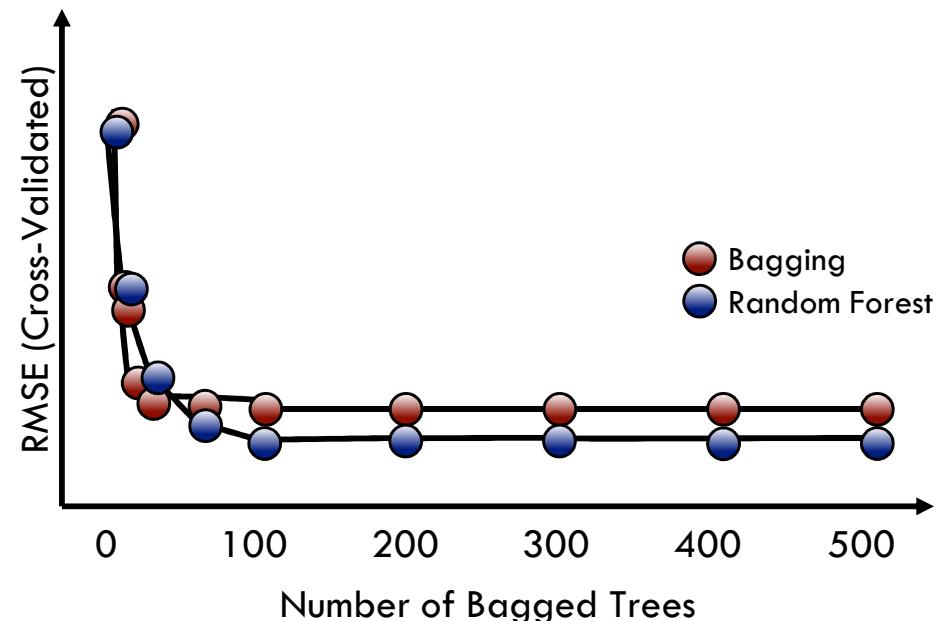
Introducing More Randomness

- Solution: further de-correlate trees
- Use random subset of features for each tree
 - Classification: \sqrt{m}
 - Regression: $m/3$



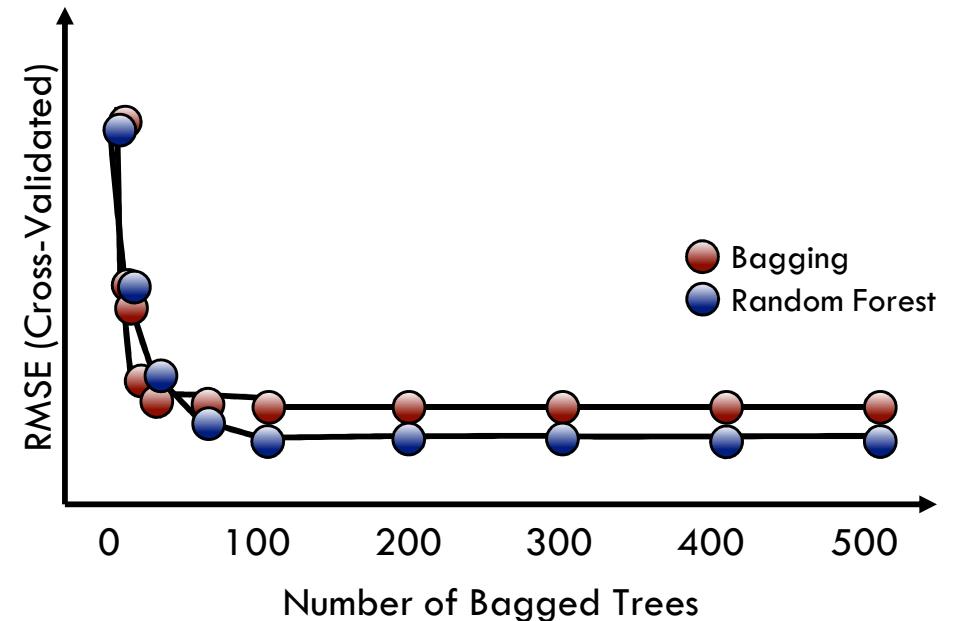
Introducing More Randomness

- Solution: further de-correlate trees
- Use random subset of features for each tree
 - Classification: \sqrt{m}
 - Regression: $m/3$
- Called "Random Forest"



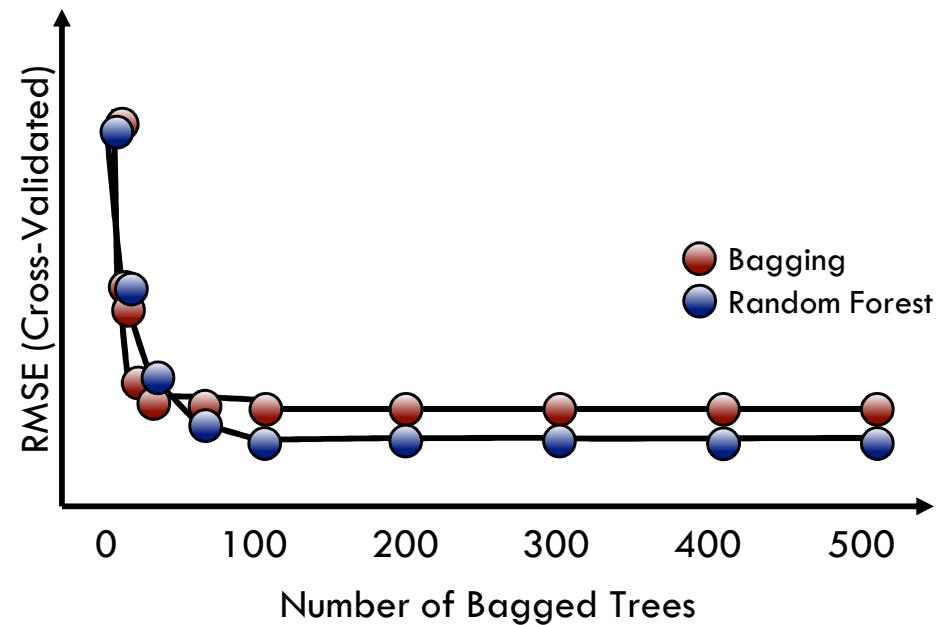
How Many Random Forest Trees?

- Errors are further reduced for Random Forest relative to Bagging



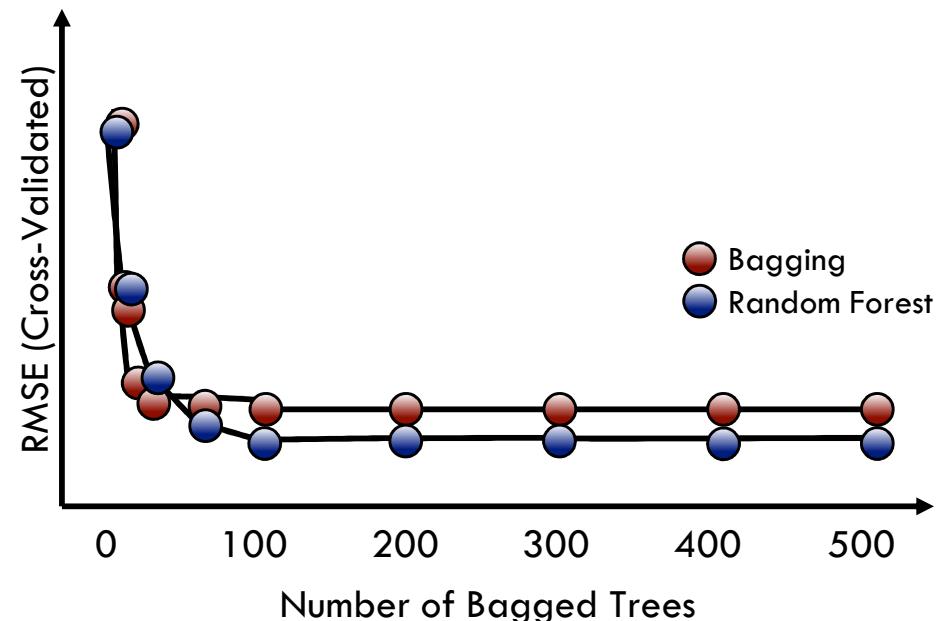
How Many Random Forest Trees?

- Errors are further reduced for Random Forest relative to Bagging
- Grow enough trees until error settles down



How Many Random Forest Trees?

- Errors are further reduced for Random Forest relative to Bagging
- Grow enough trees until error settles down
- Additional trees won't improve results



RandomForest: The Syntax

Import the class containing the classification method

```
from sklearn.ensemble import RandomForestClassifier
```

Create an instance of the class

```
RC = RandomForestClassifier(n_estimators=100, max_features=10)
```

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

```
RC = RC.fit(X_train, y_train)  
y_predict = RC.predict(X_test)
```

Tune parameters with cross-validation. Use RandomForestRegressor for regression.

Introducing Even More Randomness

- Sometimes additional randomness is desired beyond Random Forest
- Solution: select features randomly and create splits randomly—

Introducing Even More Randomness

- Sometimes additional randomness is desired beyond Random Forest
- Solution: select features randomly and create splits randomly—don't choose greedily
- Called "Extra Random Trees"

Introducing Even More Randomness

- Sometimes additional randomness is desired beyond Random Forest
- Solution: select features randomly and create splits randomly—don't choose greedily
- Called "Extra Random Trees"

ExtraTreesClassifier: The Syntax

Import the class containing the classification method

```
from sklearn.ensemble import ExtraTreesClassifier
```

Create an instance of the class

```
EC = ExtraTreesClassifier(n_estimators=100, max_features=10)
```

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

```
EC = EC.fit(X_train, y_train)  
y_predict = EC.predict(X_test)
```

Tune parameters with cross-validation. Use ExtraTreesRegressor for regression.



2.15. Boosting & Stacking

Review of Bagging

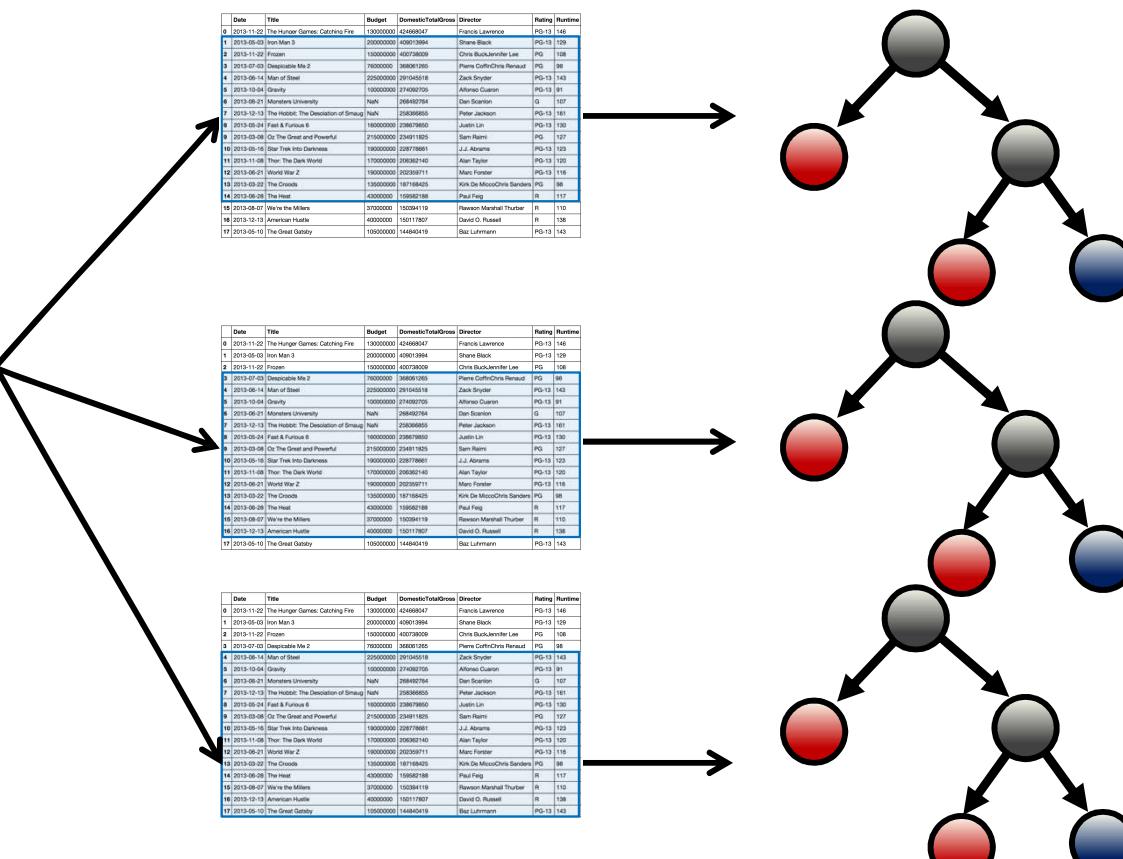
Grow decision tree from multiple bootstrapped samples

	Date	Title	Budget	DomesticTotalGross	Director	Rating	Runtime
0	2013-11-22	The Hunger Games: Catching Fire	130000000	242668047	Francis Lawrence	PG-13	146
1	2013-05-03	Iron Man 3	200000000	409013984	Shane Black	PG-13	129
2	2013-11-22	Frozen	150000000	400738009	Chris BuckJennifer Lee	PG	108
3	2013-07-03	Despicable Me 2	76000000	368061265	Pierre CoffeChris Renaud	PG	98
4	2013-06-14	Man of Steel	22500000	291045518	Zack Snyder	PG-13	143
5	2013-10-04	Gravity	10000000	274092705	Alfonso Cuaron	PG-13	91
6	2013-06-21	Monsters University	Nan	268492764	Dan Scanlon	G	107
7	2013-12-13	The Hobbit: The Desolation of Smaug	258366855	258366855	Peter Jackson	PG-13	161
8	2013-06-21	Fast & Furious 6	16000000	238679850	Justin Lin	PG-13	130
9	2013-03-08	Oz The Great and Powerful	21500000	234911825	Sam Raimi	PG	127
10	2013-05-16	Star Trek Into Darkness	19000000	228778661	J.J. Abrams	PG-13	123
11	2013-11-08	Thor: The Dark World	17000000	206582140	Alan Taylor	PG-13	120
12	2013-06-21	World War Z	19000000	202359711	Marc Forster	PG-13	116
13	2013-03-22	The Croods	13500000	187168425	Kirk De MiccoChris Sanders	PG	98
14	2013-06-28	The Heat	4300000	159562188	Paul Feig	R	117
15	2013-06-07	We're the Millers	3700000	150394119	Rawson Marshall Thurber	R	110
16	2013-12-13	American Hustle	4000000	150117807	David O. Russell	R	138
17	2013-05-10	The Great Gatsby	10500000	144840419	Baz Luhrmann	PG-13	143

	Date	Title	Budget	DomesticTotalGross	Director	Rating	Runtime
0	2013-11-22	The Hunger Games: Catching Fire	19000000	249698497	Francis Lawrence	PG-13	146
1	2013-05-03	Iron Man 3	20000000	40613994	Shane Black	PG-13	129
2	2013-11-22	Frozen	15000000	400738009	Chris BuckJennifer Lee	PG	108
3	2013-07-03	Despicable Me 2	76000000	368061265	Pierre CoffeChris Renaud	PG	98
4	2013-06-14	Man of Steel	22500000	291045518	Zack Snyder	PG-13	143
5	2013-10-04	Gravity	10000000	274092705	Alfonso Cuaron	PG-13	91
6	2013-06-21	Monsters University	Nan	268492764	Dan Scanlon	G	107
7	2013-12-13	The Hobbit: The Desolation of Smaug	258366855	258366855	Peter Jackson	PG-13	161
8	2013-06-21	Fast & Furious 6	16000000	238679850	Justin Lin	PG-13	130
9	2013-03-08	Oz The Great and Powerful	21500000	234911825	Sam Raimi	PG	127
10	2013-05-16	Star Trek Into Darkness	19000000	228778661	J.J. Abrams	PG-13	123
11	2013-11-08	Thor: The Dark World	17000000	206582140	Alan Taylor	PG-13	120
12	2013-06-21	World War Z	19000000	202359711	Marc Forster	PG-13	116
13	2013-03-22	The Croods	13500000	187168425	Kirk De MiccoChris Sanders	PG	98
14	2013-06-28	The Heat	4300000	159562188	Paul Feig	R	117
15	2013-06-07	We're the Millers	3700000	150394119	Rawson Marshall Thurber	R	110
16	2013-12-13	American Hustle	4000000	150117807	David O. Russell	R	138
17	2013-05-10	The Great Gatsby	10500000	144840419	Baz Luhrmann	PG-13	143

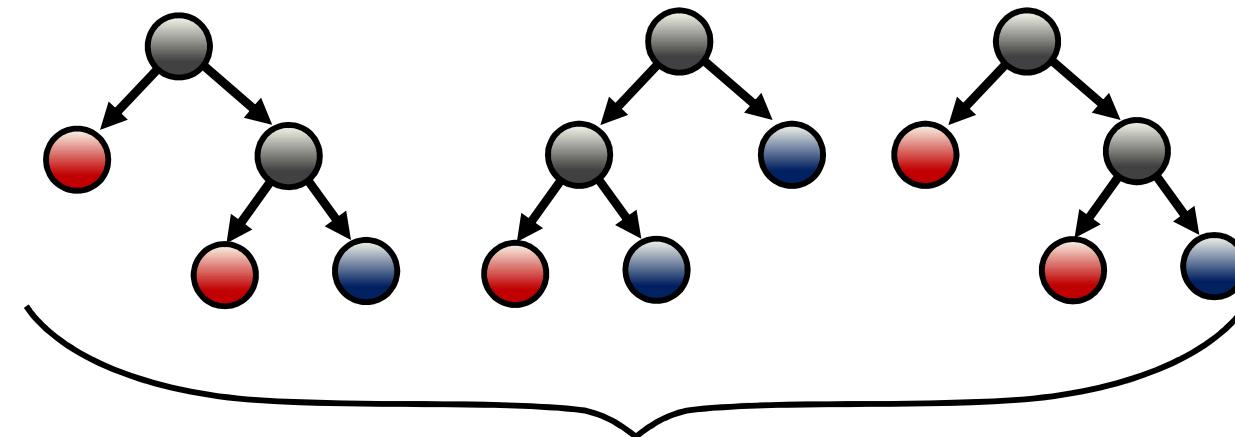
	Date	Title	Budget	DomesticTotalGross	Director	Rating	Runtime
0	2013-11-22	The Hunger Games: Catching Fire	19000000	249698497	Francis Lawrence	PG-13	146
1	2013-05-03	Iron Man 3	20000000	40613994	Shane Black	PG-13	129
2	2013-11-22	Frozen	15000000	400738009	Chris BuckJennifer Lee	PG	108
3	2013-07-03	Despicable Me 2	76000000	368061265	Pierre CoffeChris Renaud	PG	98
4	2013-06-14	Man of Steel	22500000	291045518	Zack Snyder	PG-13	143
5	2013-10-04	Gravity	10000000	274092705	Alfonso Cuaron	PG-13	91
6	2013-06-21	Monsters University	Nan	268492764	Dan Scanlon	G	107
7	2013-12-13	The Hobbit: The Desolation of Smaug	258366855	258366855	Peter Jackson	PG-13	161
8	2013-06-21	Fast & Furious 6	16000000	238679850	Justin Lin	PG-13	130
9	2013-03-08	Oz The Great and Powerful	21500000	234911825	Sam Raimi	PG	127
10	2013-05-16	Star Trek Into Darkness	19000000	228778661	J.J. Abrams	PG-13	123
11	2013-11-08	Thor: The Dark World	17000000	206582140	Alan Taylor	PG-13	120
12	2013-06-21	World War Z	19000000	202359711	Marc Forster	PG-13	116
13	2013-03-22	The Croods	13500000	187168425	Kirk De MiccoChris Sanders	PG	98
14	2013-06-28	The Heat	4300000	159562188	Paul Feig	R	117
15	2013-06-07	We're the Millers	3700000	150394119	Rawson Marshall Thurber	R	110
16	2013-12-13	American Hustle	4000000	150117807	David O. Russell	R	138
17	2013-05-10	The Great Gatsby	10500000	144840419	Baz Luhrmann	PG-13	143

	Date	Title	Budget	DomesticTotalGross	Director	Rating	Runtime
0	2013-11-22	The Hunger Games: Catching Fire	19000000	249698497	Francis Lawrence	PG-13	146
1	2013-05-03	Iron Man 3	20000000	40613994	Shane Black	PG-13	129
2	2013-11-22	Frozen	15000000	400738009	Chris BuckJennifer Lee	PG	108
3	2013-07-03	Despicable Me 2	76000000	368061265	Pierre CoffeChris Renaud	PG	98
4	2013-06-14	Man of Steel	22500000	291045518	Zack Snyder	PG-13	143
5	2013-10-04	Gravity	10000000	274092705	Alfonso Cuaron	PG-13	91
6	2013-06-21	Monsters University	Nan	268492764	Dan Scanlon	G	107
7	2013-12-13	The Hobbit: The Desolation of Smaug	258366855	258366855	Peter Jackson	PG-13	161
8	2013-06-21	Fast & Furious 6	16000000	238679850	Justin Lin	PG-13	130
9	2013-03-08	Oz The Great and Powerful	21500000	234911825	Sam Raimi	PG	127
10	2013-05-16	Star Trek Into Darkness	19000000	228778661	J.J. Abrams	PG-13	123
11	2013-11-08	Thor: The Dark World	17000000	206582140	Alan Taylor	PG-13	120
12	2013-06-21	World War Z	19000000	202359711	Marc Forster	PG-13	116
13	2013-03-22	The Croods	13500000	187168425	Kirk De MiccoChris Sanders	PG	98
14	2013-06-28	The Heat	4300000	159562188	Paul Feig	R	117
15	2013-06-07	We're the Millers	3700000	150394119	Rawson Marshall Thurber	R	110
16	2013-12-13	American Hustle	4000000	150117807	David O. Russell	R	138
17	2013-05-10	The Great Gatsby	10500000	144840419	Baz Luhrmann	PG-13	143

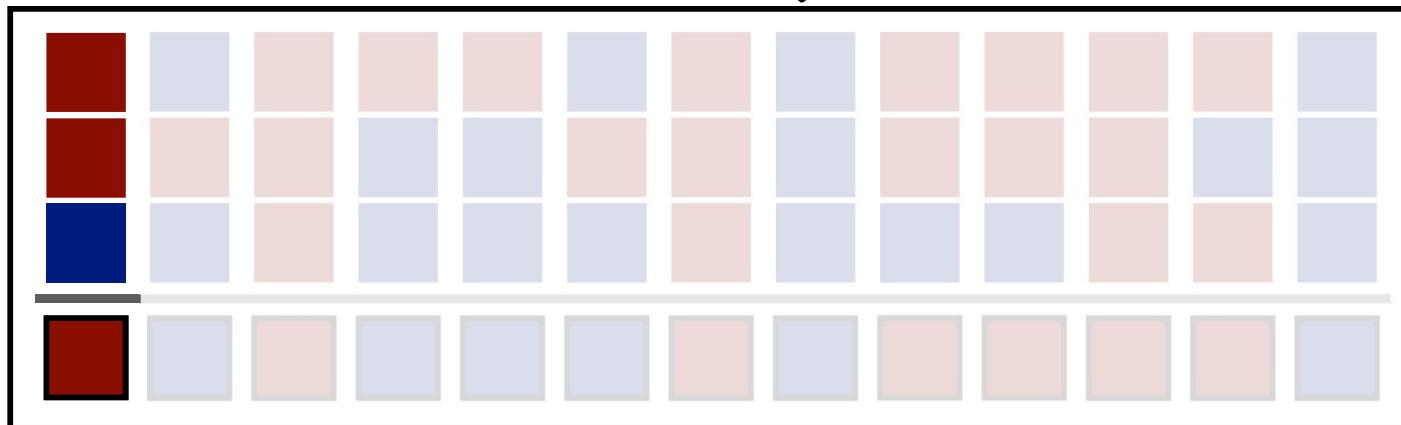


Review of Bagging

Vote on or average result from each tree for each data point

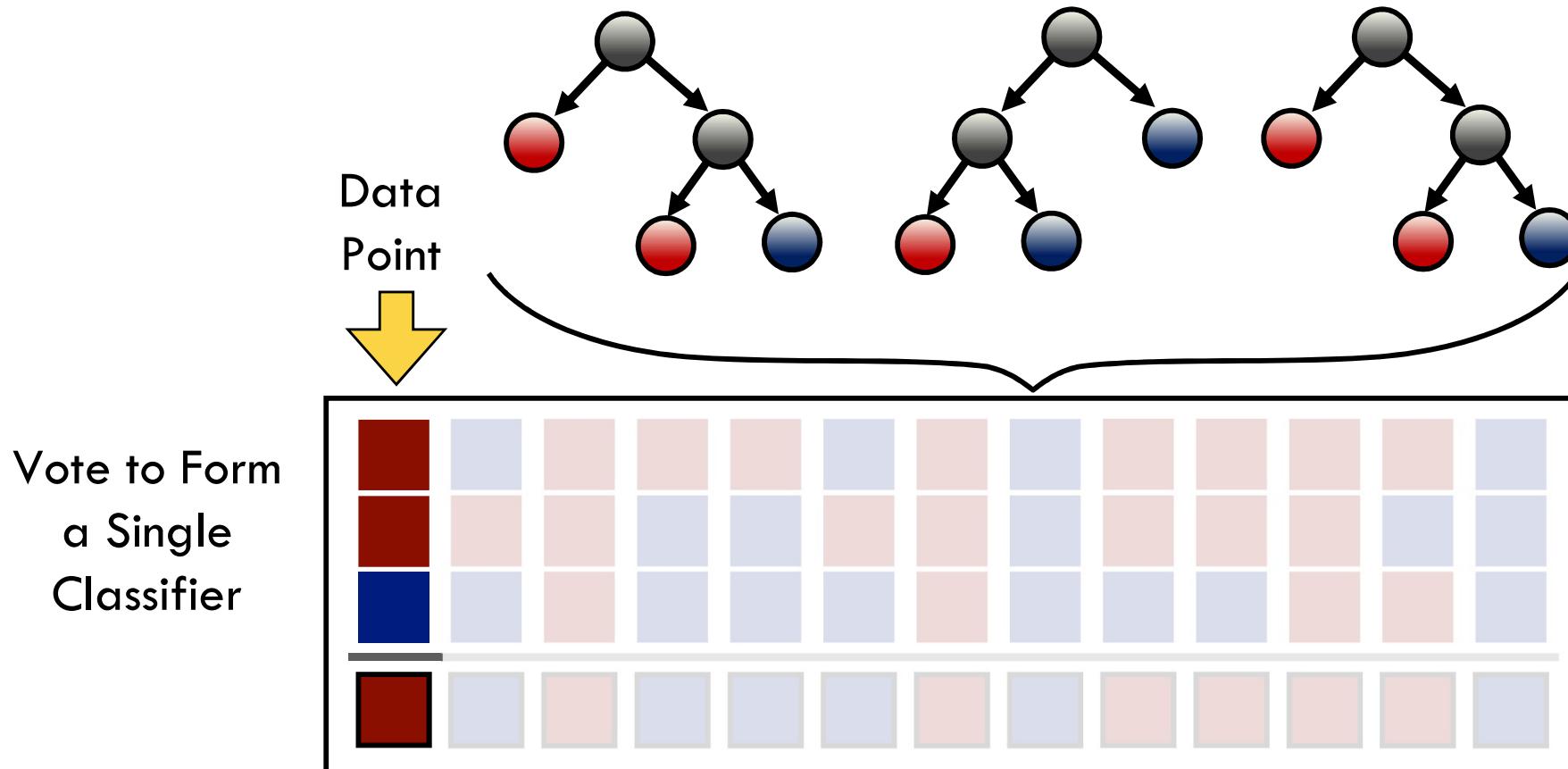


Vote to Form
a Single
Classifier



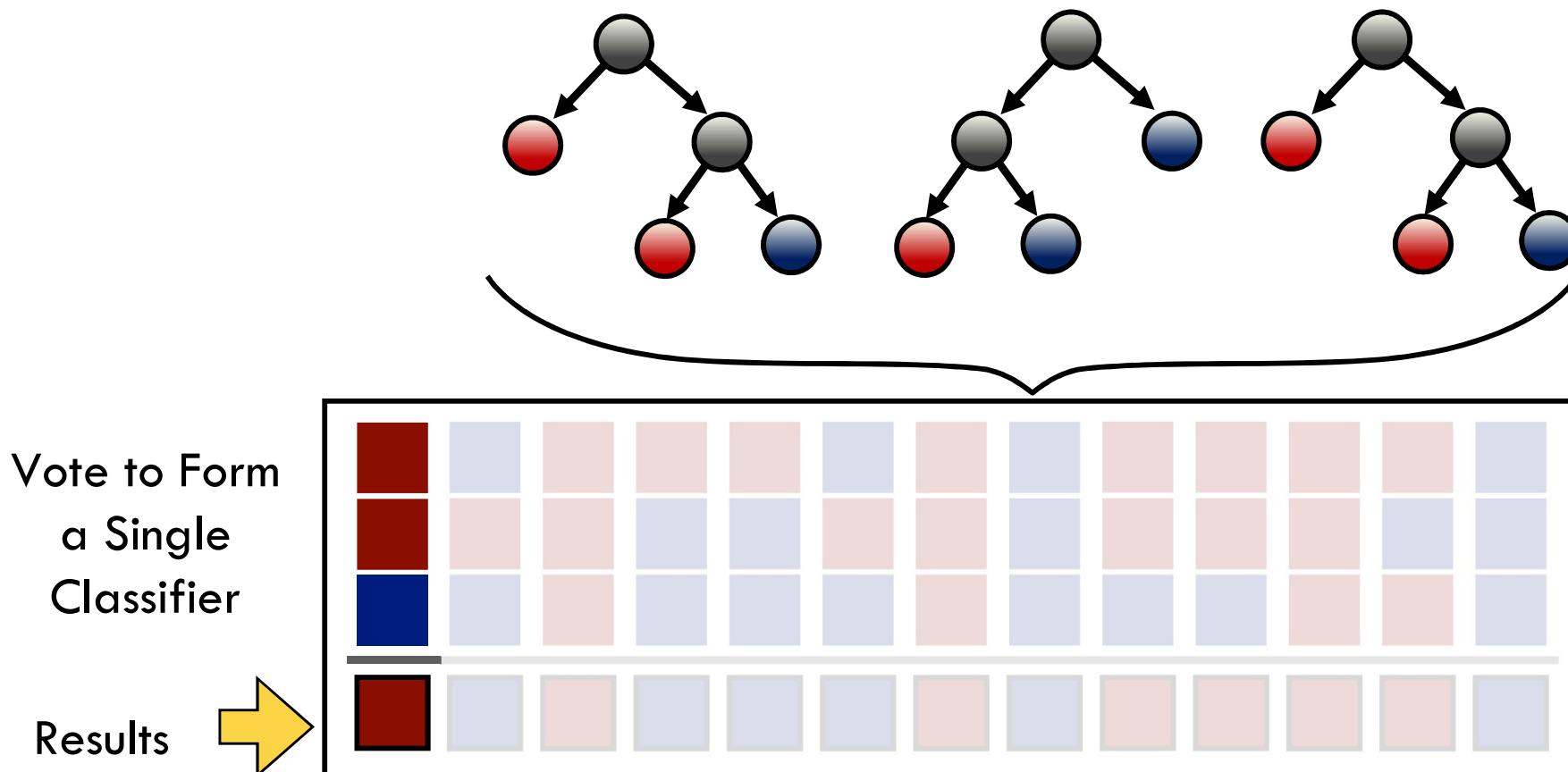
Review of Bagging

Vote on or average result from each tree for each data point



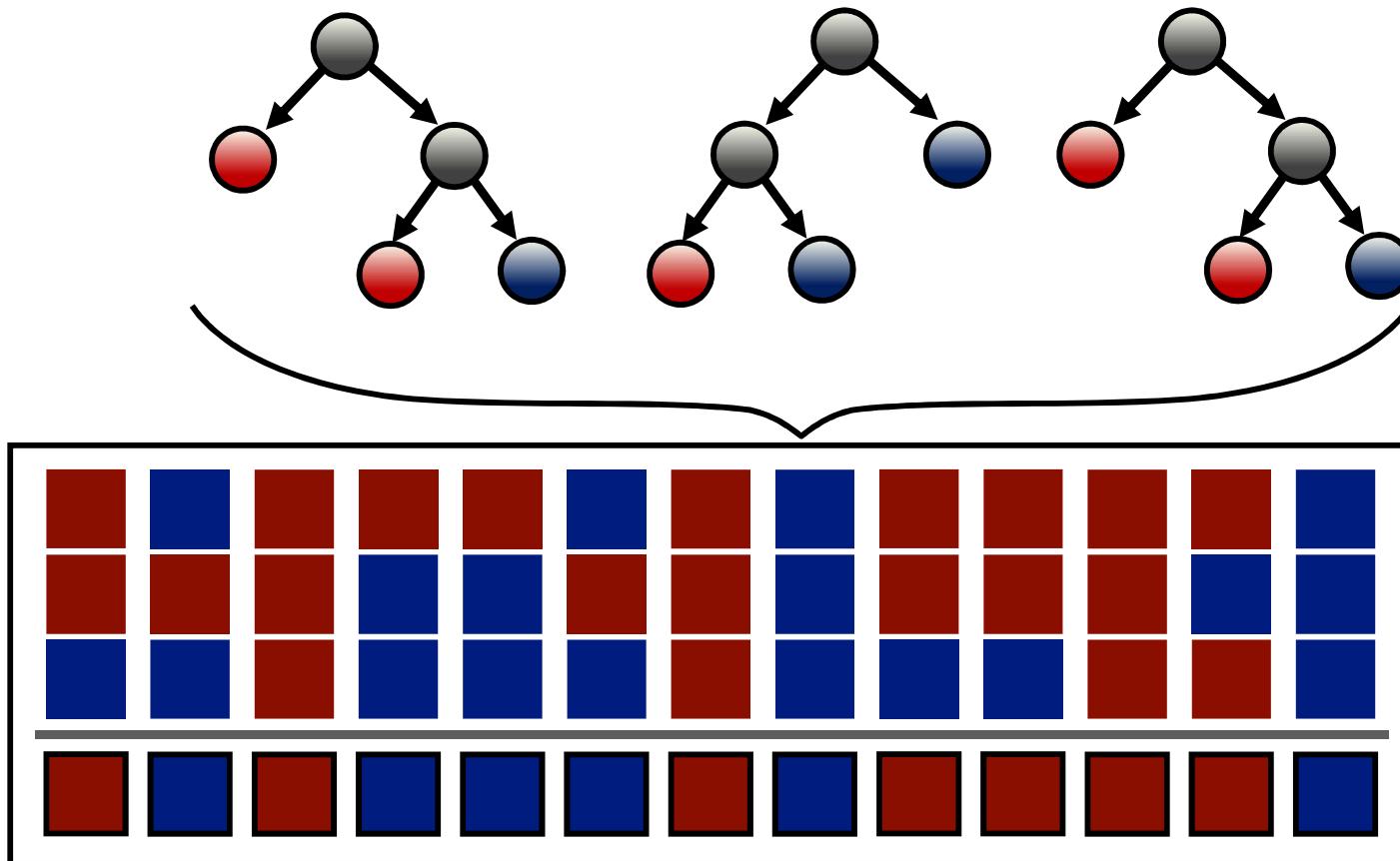
Review of Bagging

Vote on or average result from each tree for each data point



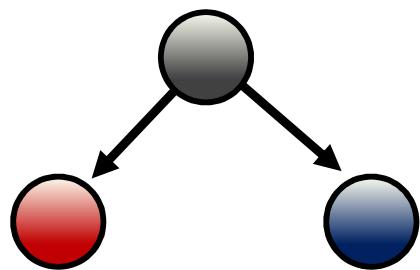
Review of Bagging

Vote on or average result from each tree for each data point



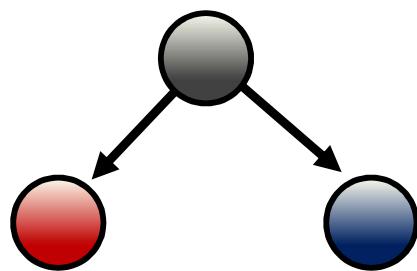
Decision Stump: the Boosting Base Learner

Temperature >50°F

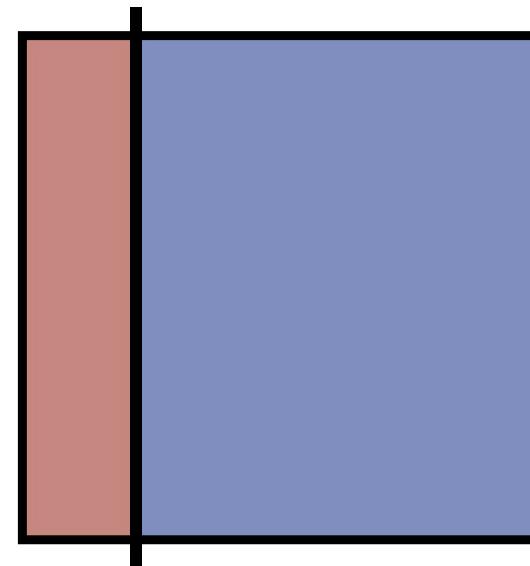


Decision Stump: the Boosting Base Learner

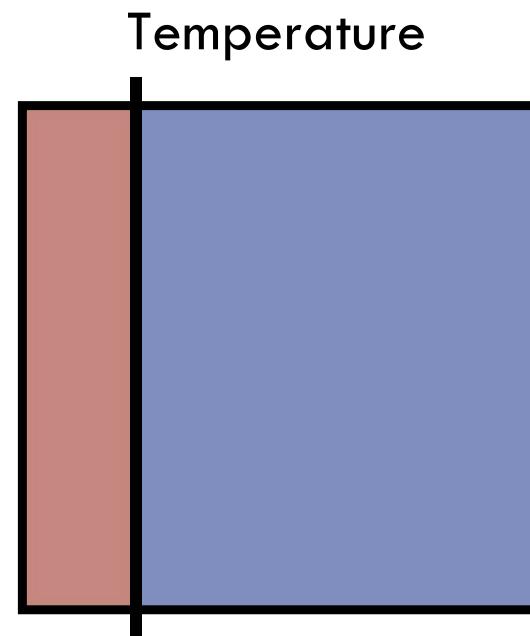
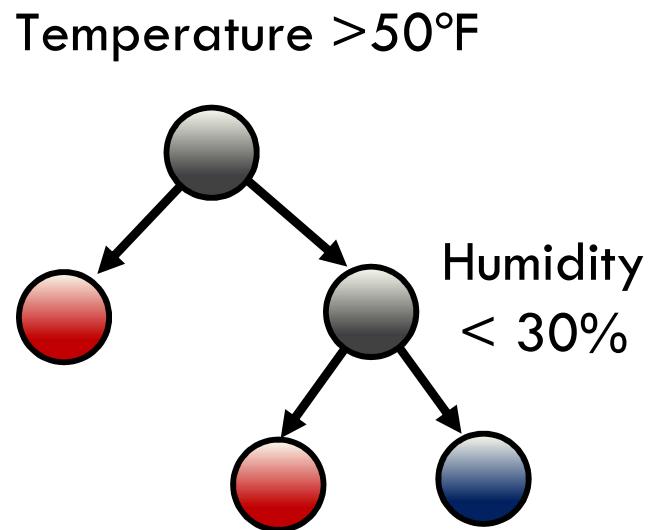
Temperature >50°F



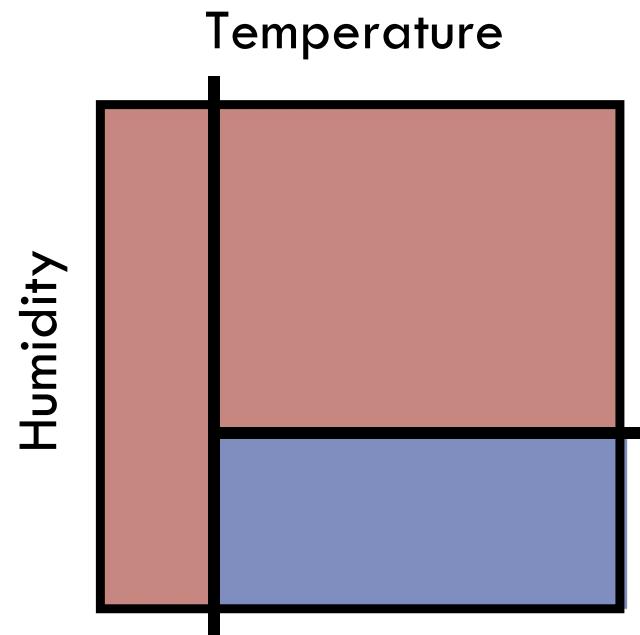
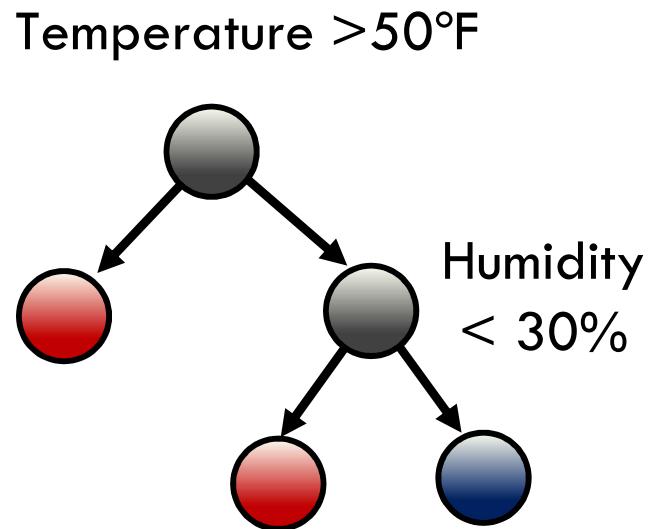
Temperature



Decision Stump: the Boosting Base Learner

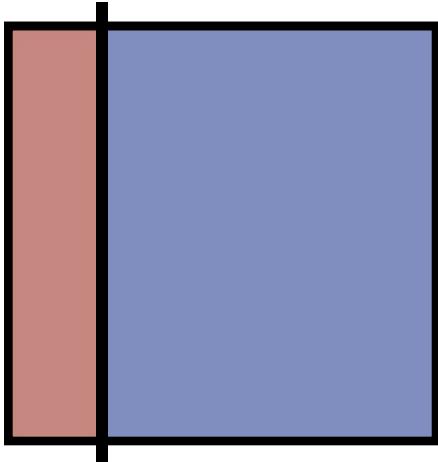


Decision Stump: the Boosting Base Learner



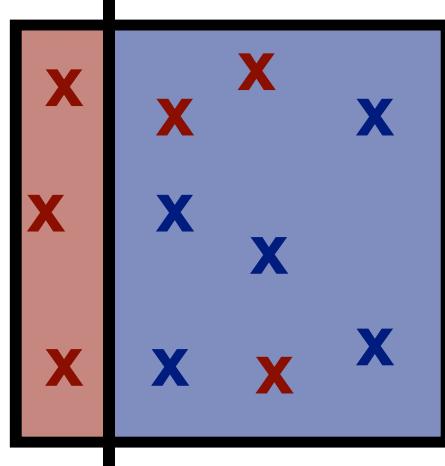
Overview of Boosting

Create
initial
decision
stump



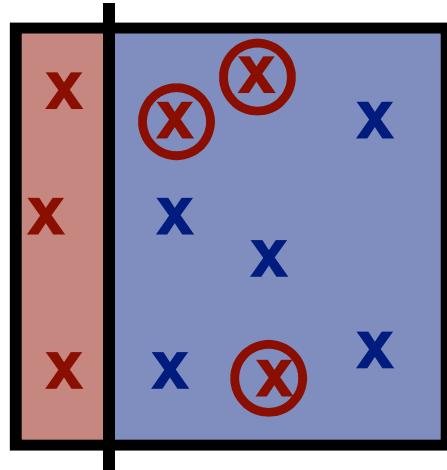
Overview of Boosting

Fit to
data
and
calculate
residuals



Overview of Boosting

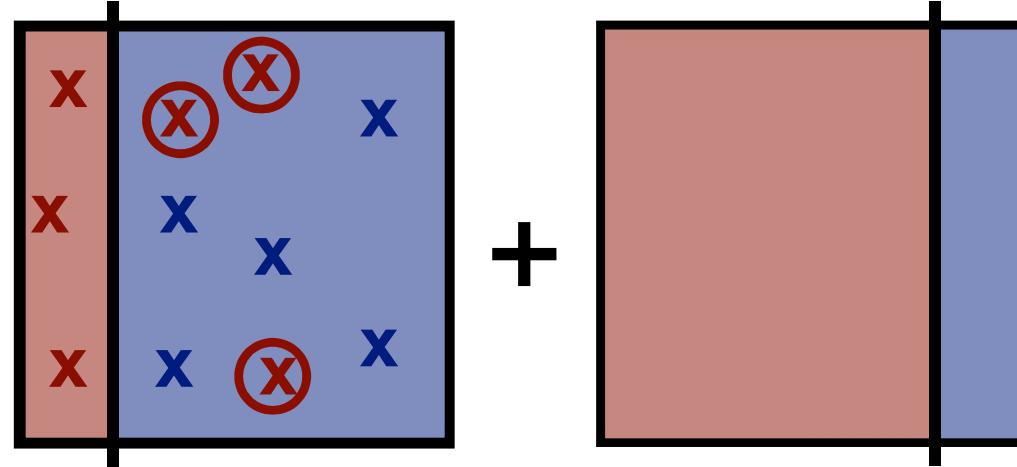
Adjust
weight
of points



Find new Overview of Boosting

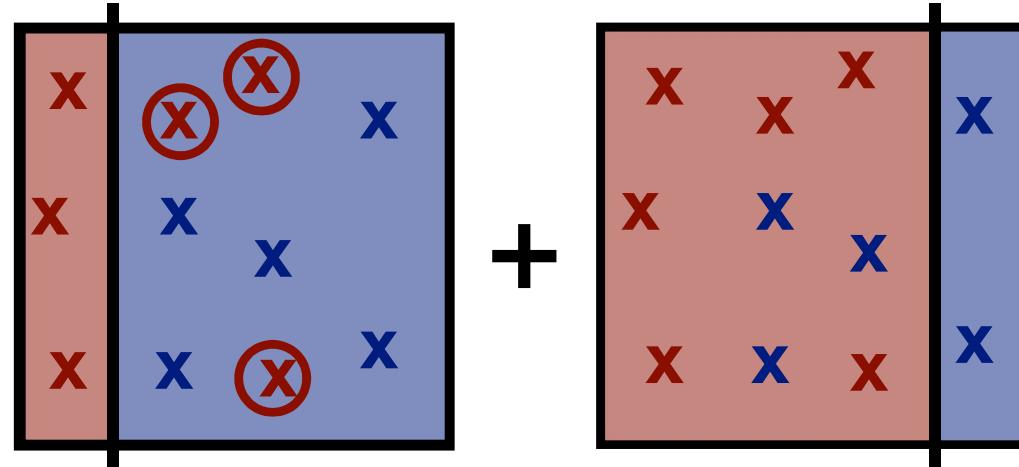
decision
stump to
fit
weighted
d

residuals



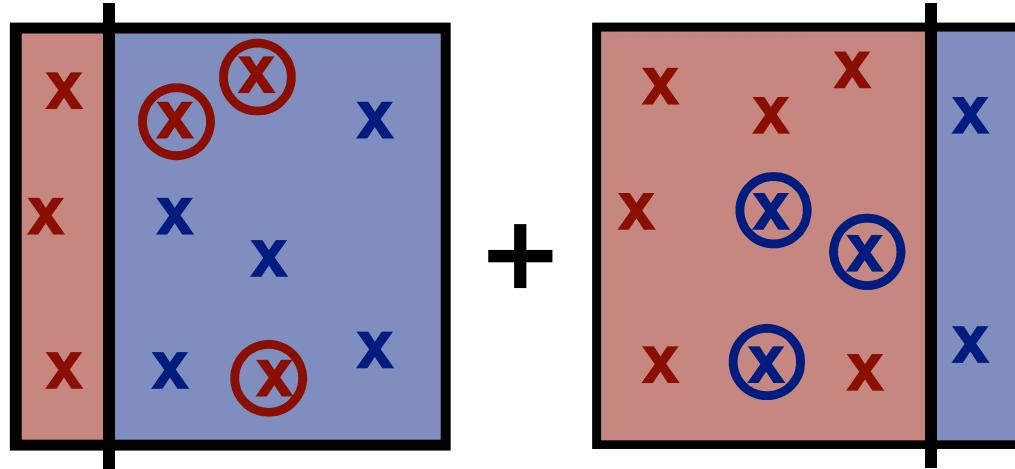
Overview of Boosting

Fit new
decision
stump to
current
residuals



Overview of Boosting

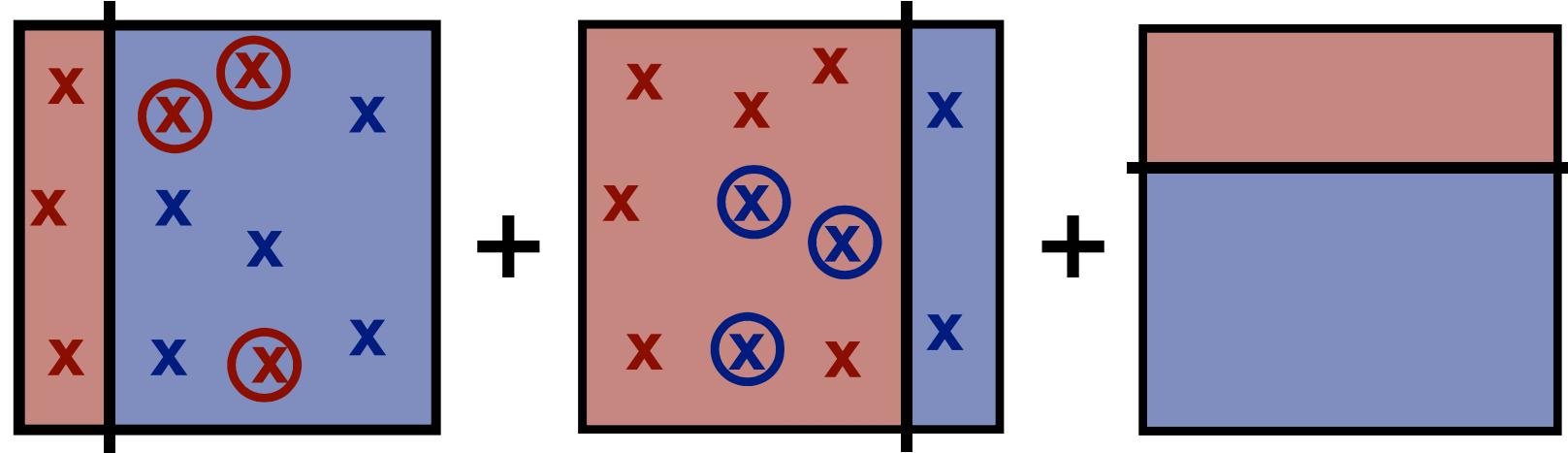
calculator
e errors
and
weight
data
points



Find new Overview of Boosting

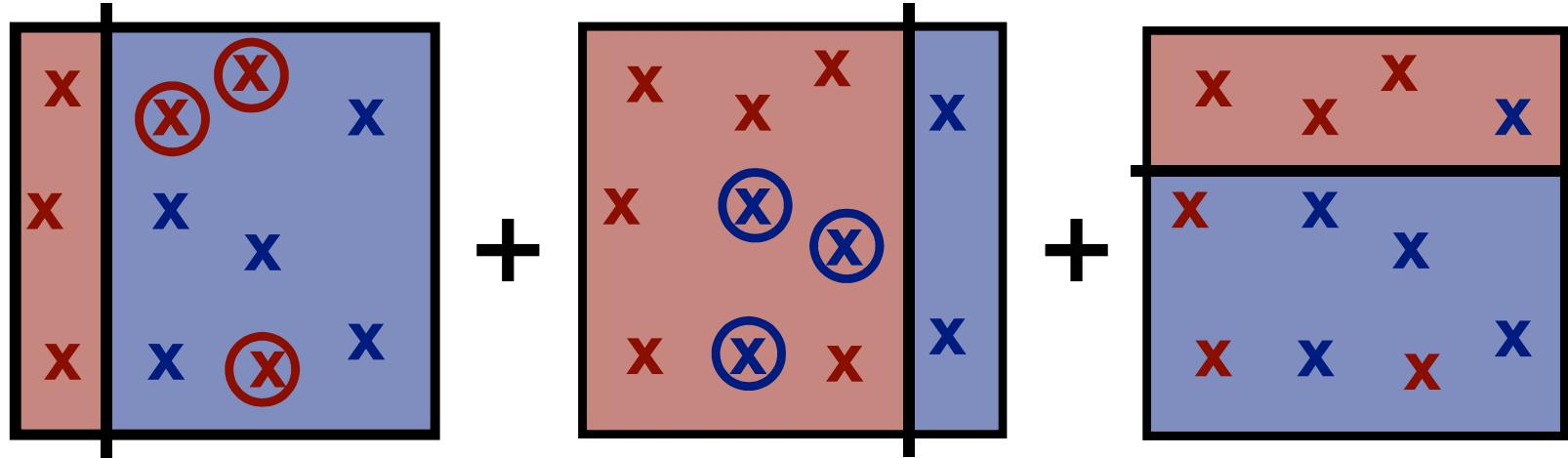
decision
stump to
fit
weighted
d

residuals

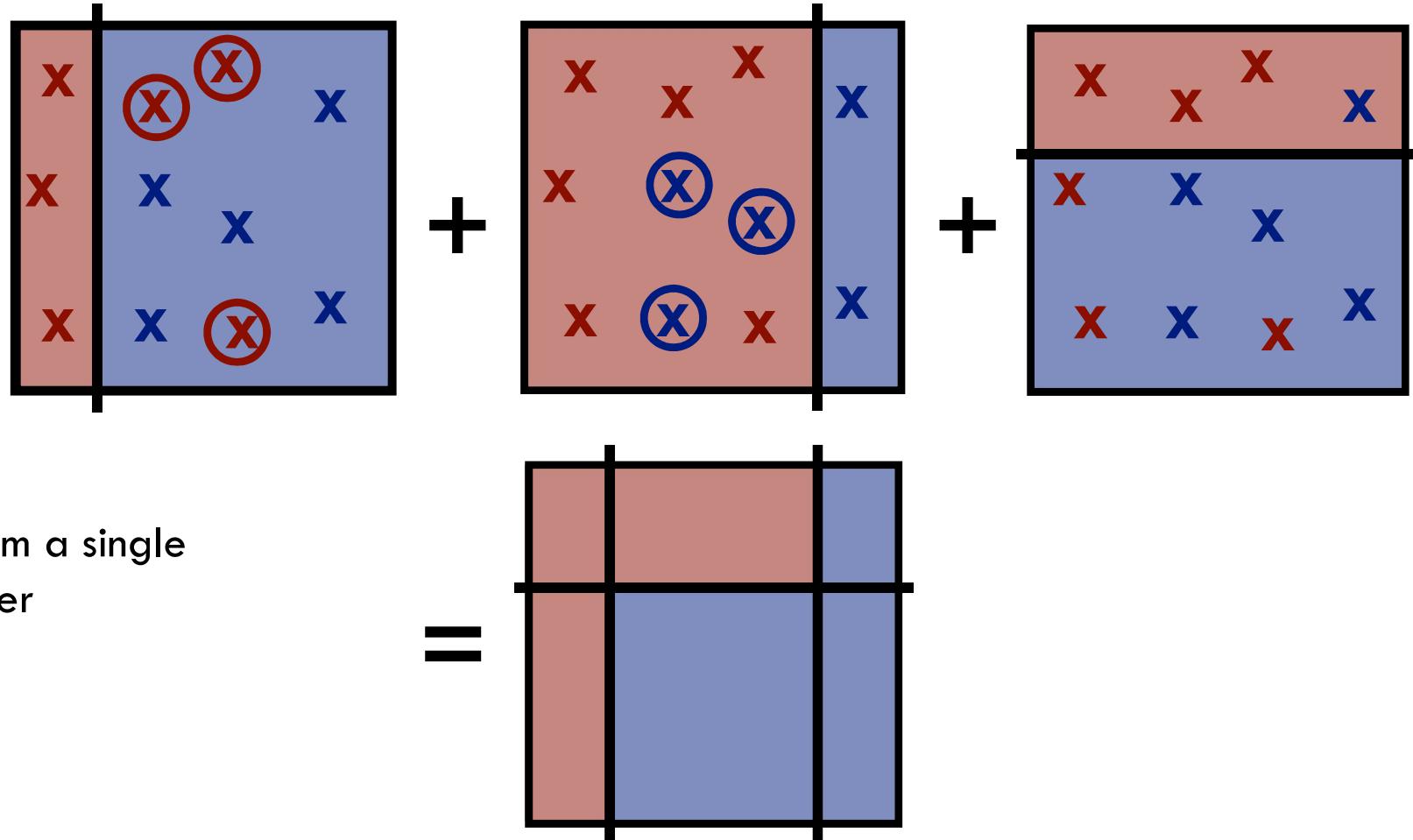


Overview of Boosting

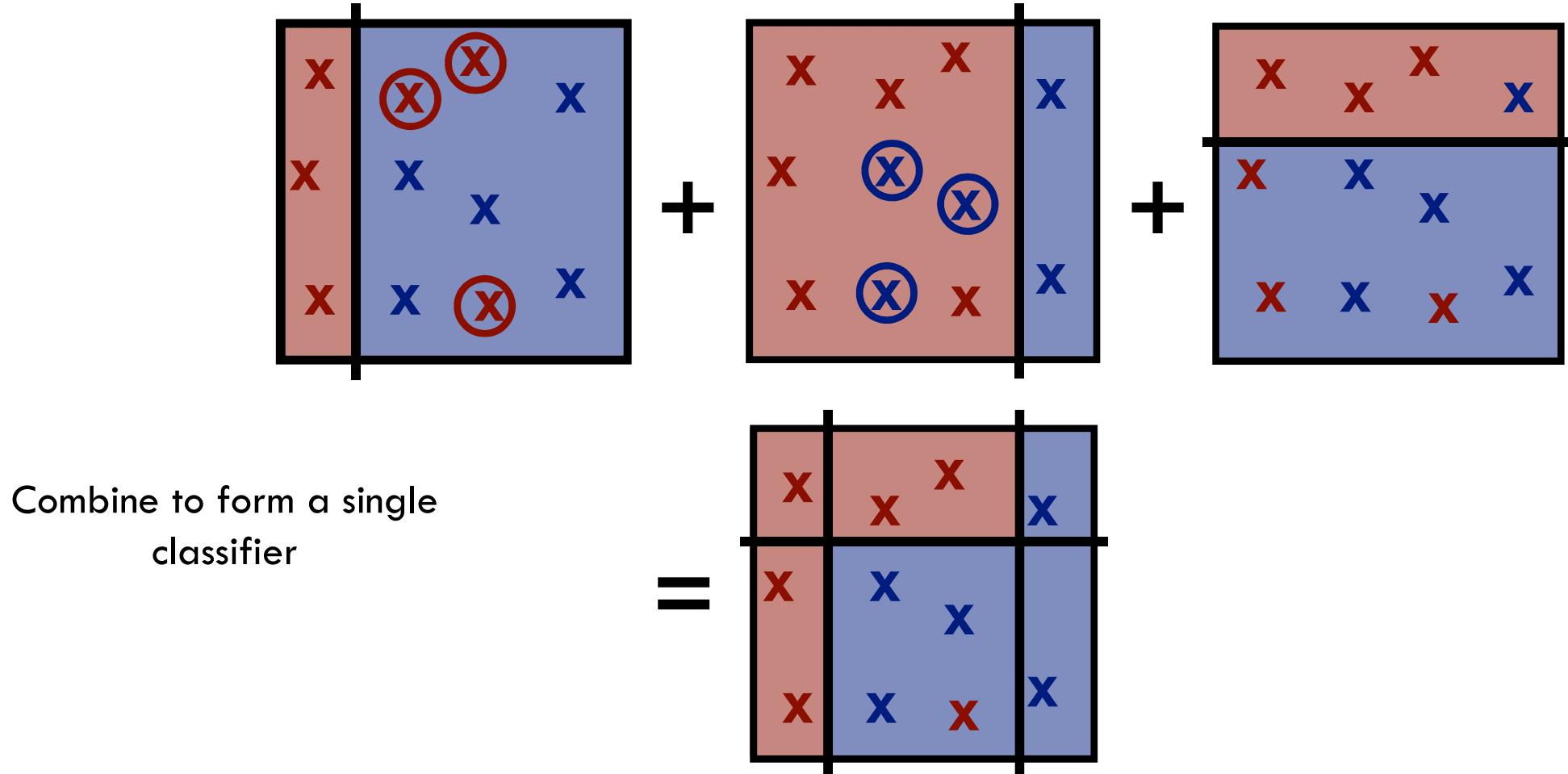
Fit new
decision
stump to
current
residuals



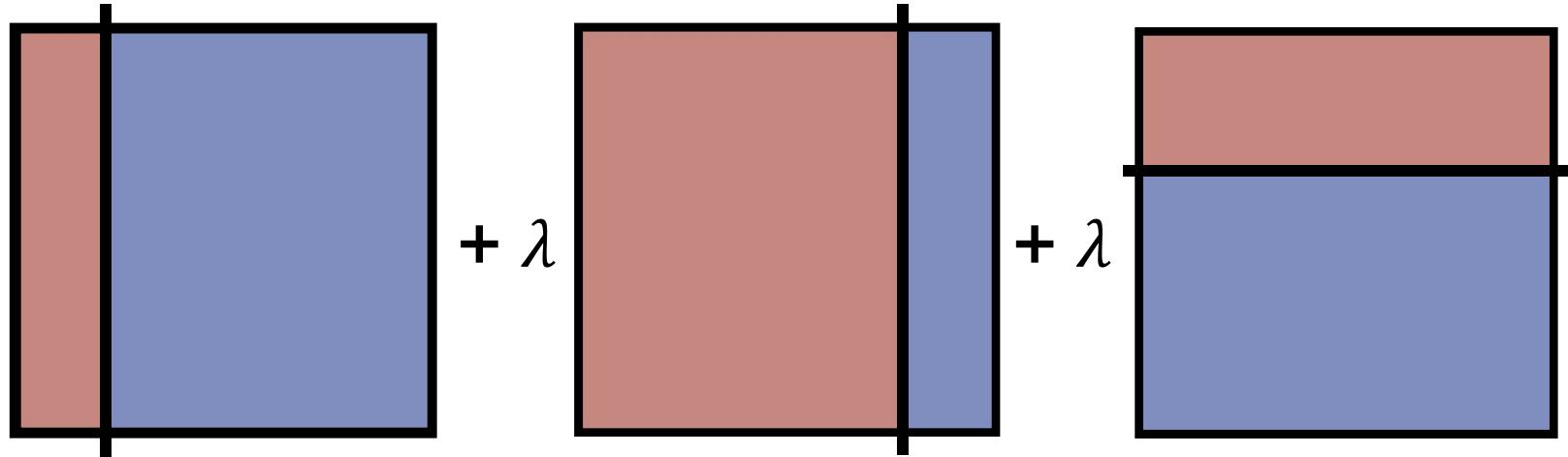
Overview of Boosting



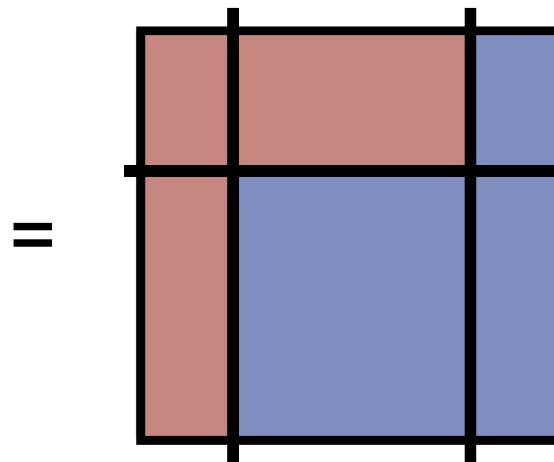
Overview of Boosting



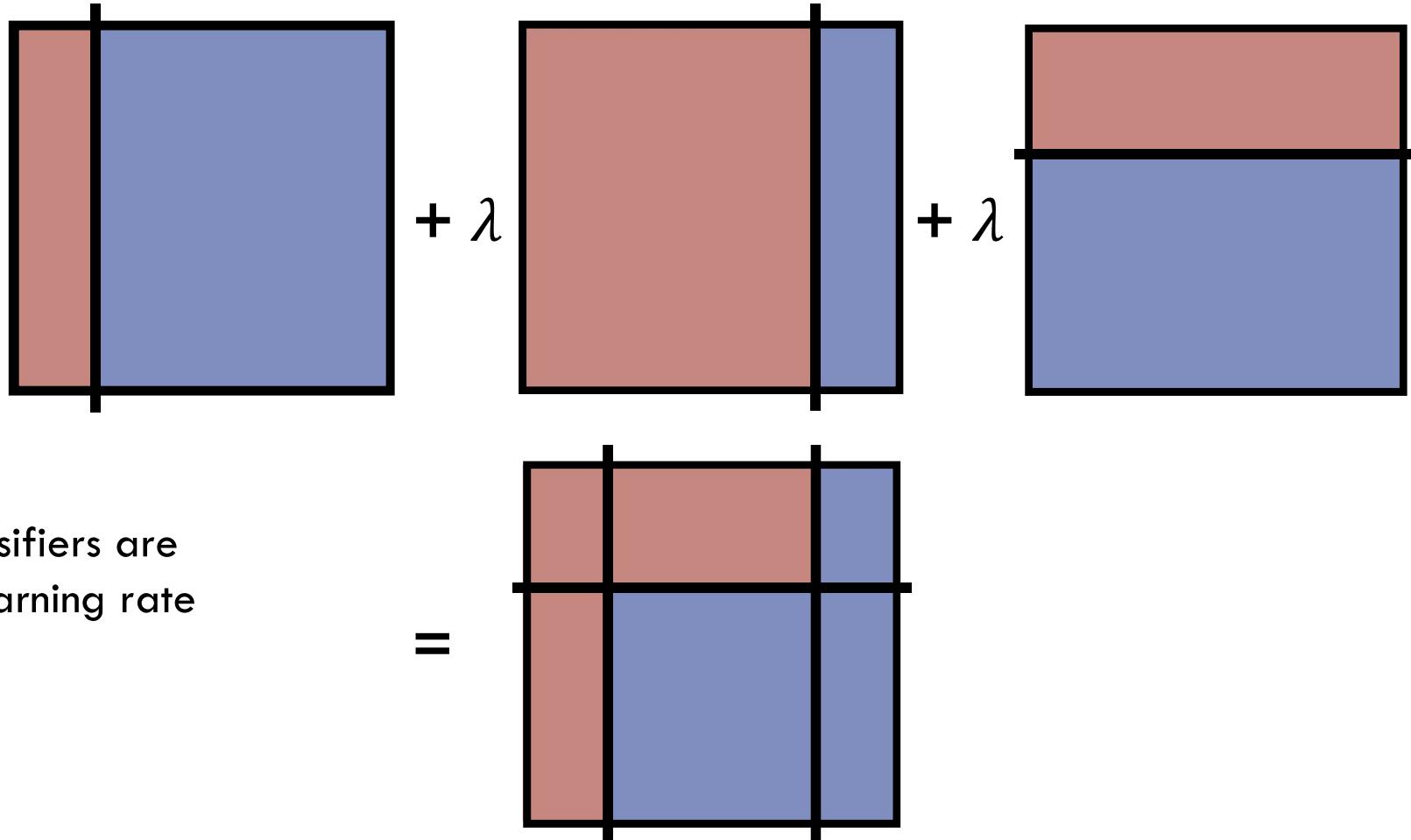
Overview of Boosting



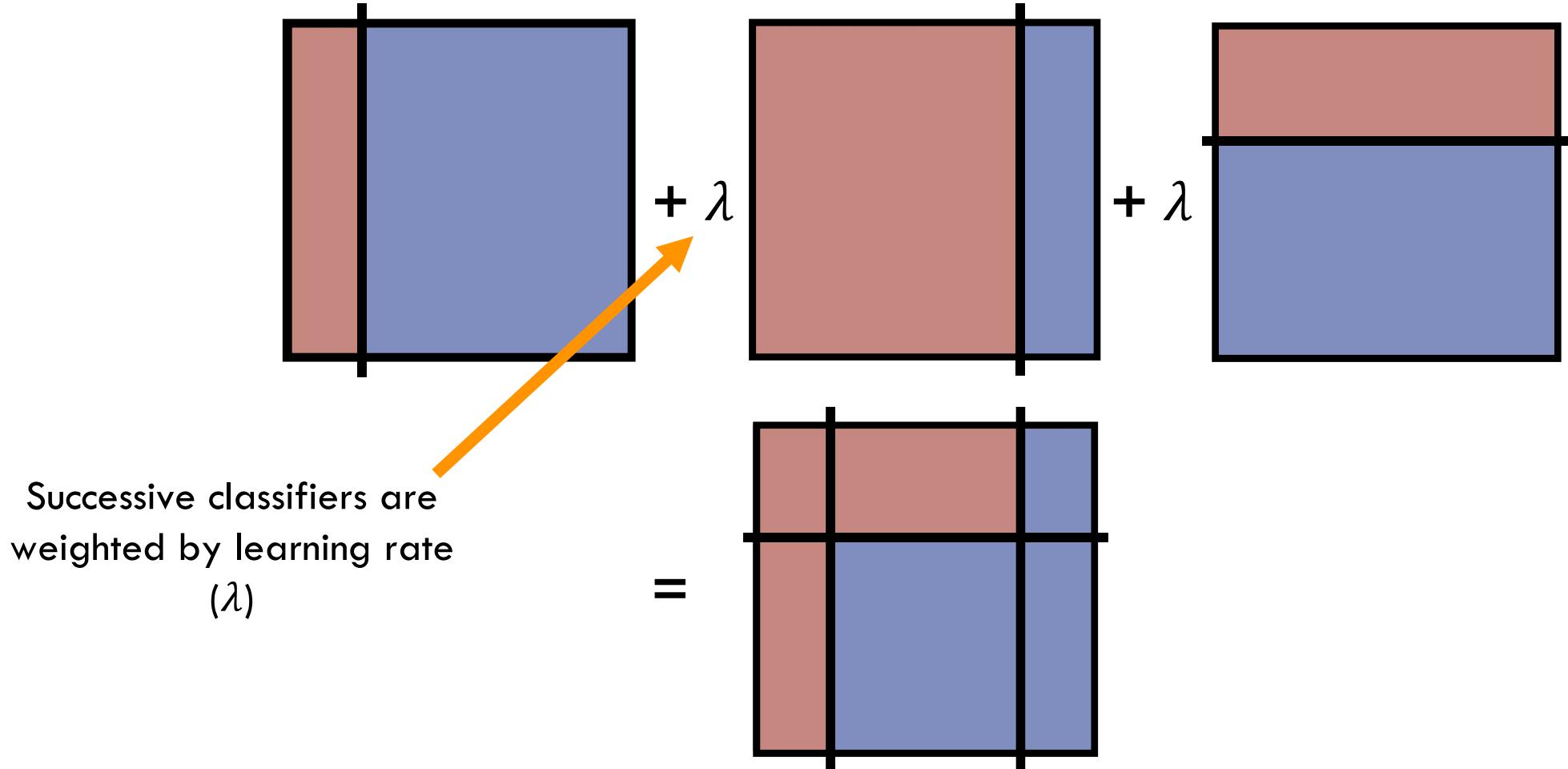
Result is weighted sum of
all classifiers



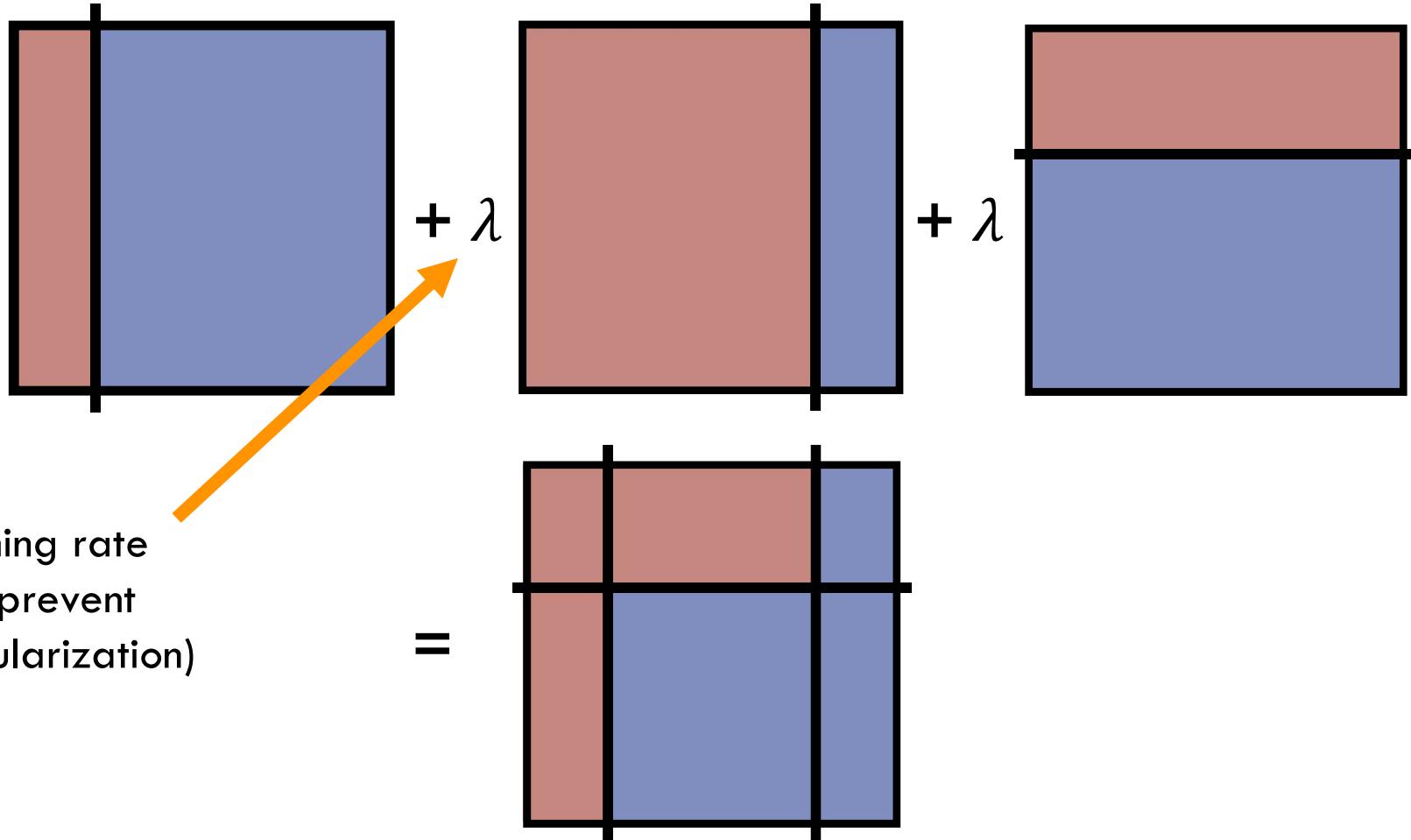
Overview of Boosting



Overview of Boosting



Overview of Boosting

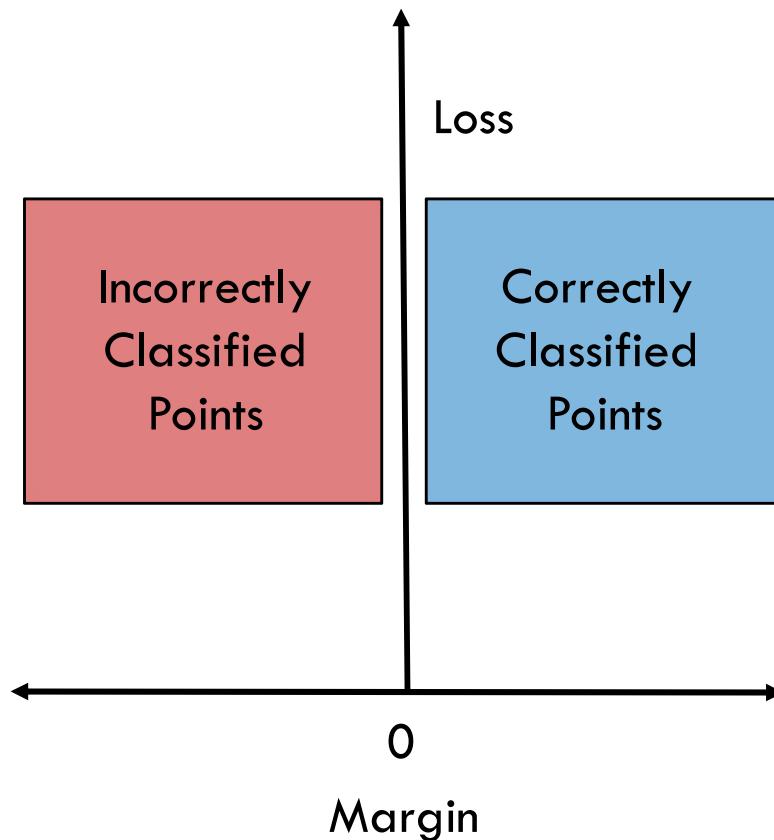


Boosting Specifics

- Boosting utilizes different loss functions
- At each stage, the margin is determined for each point

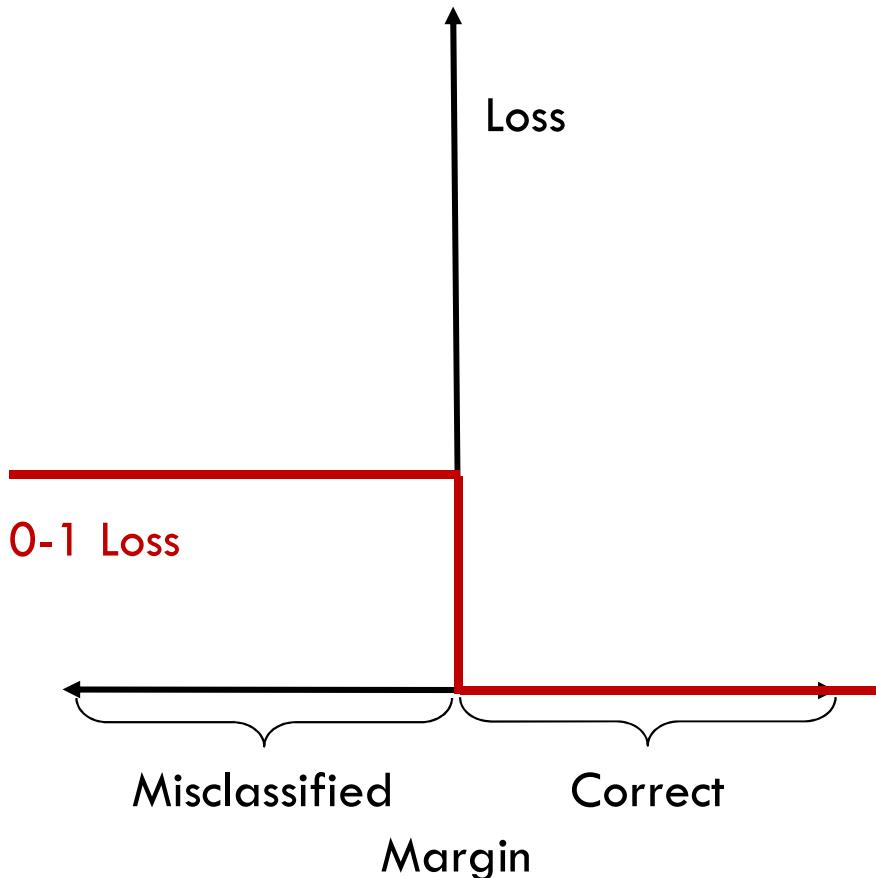
Boosting Specifics

- Boosting utilizes different loss functions
- At each stage, the margin is determined for each point
- Margin is positive for correctly classified points and negative for misclassifications
- Value of loss function is calculated from margin



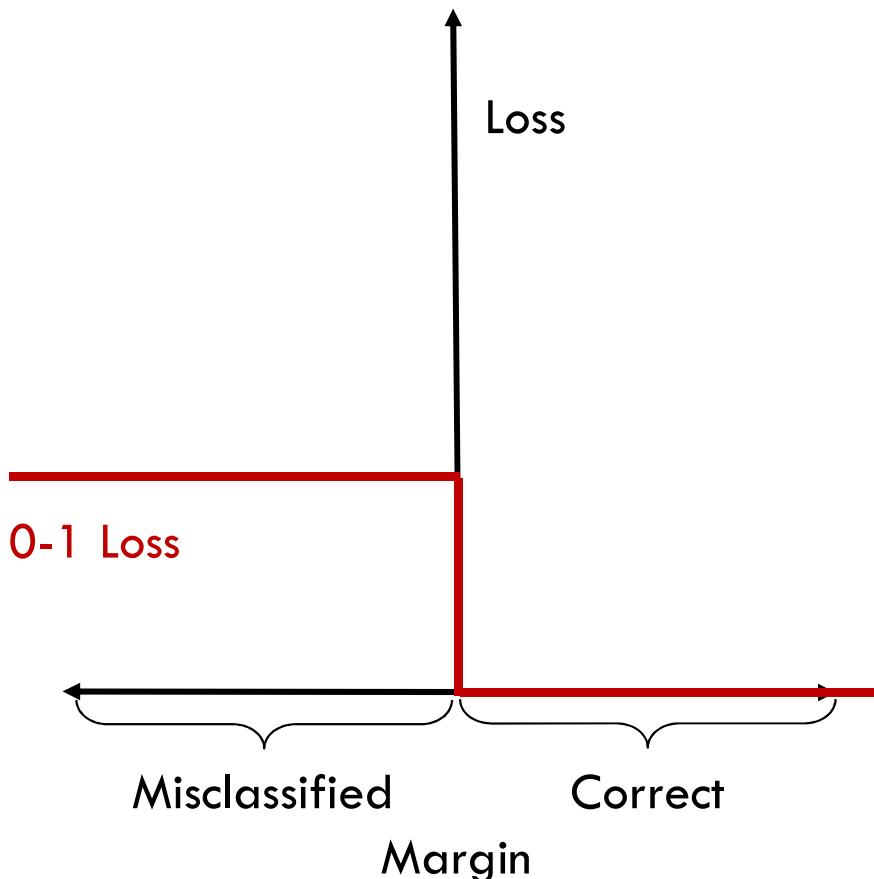
0 – 1 Loss Function

- The 0 – 1 Loss multiplies misclassified points by 1



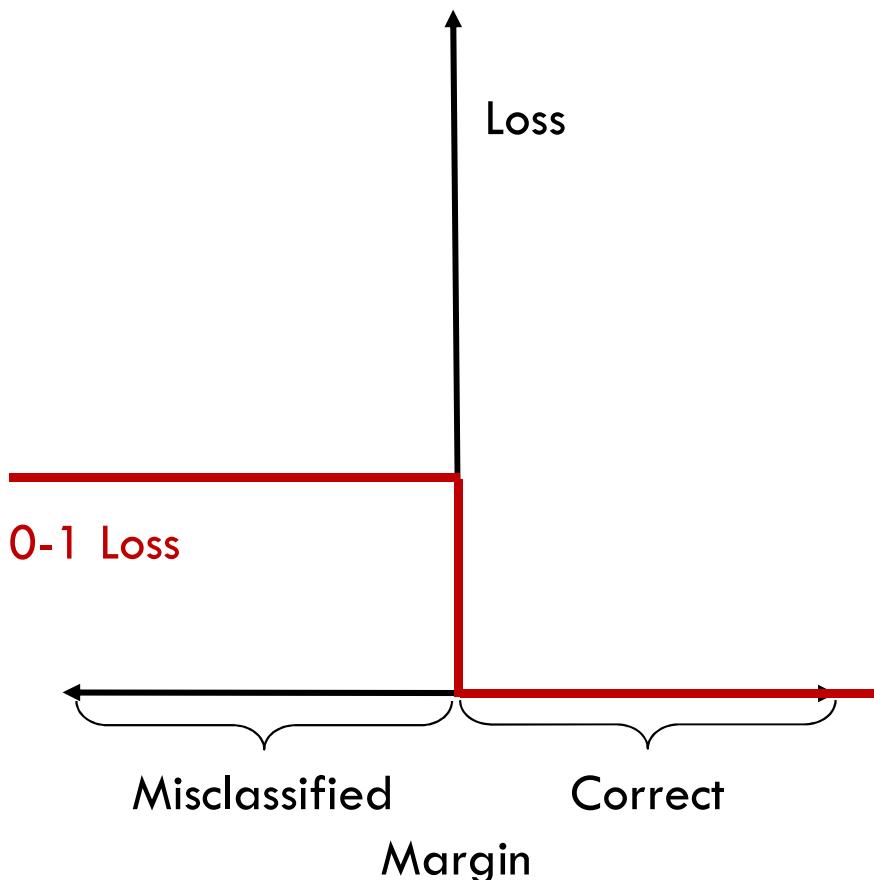
0 – 1 Loss Function

- The 0 – 1 Loss multiplies misclassified points by 1
- Correctly classified points are ignored



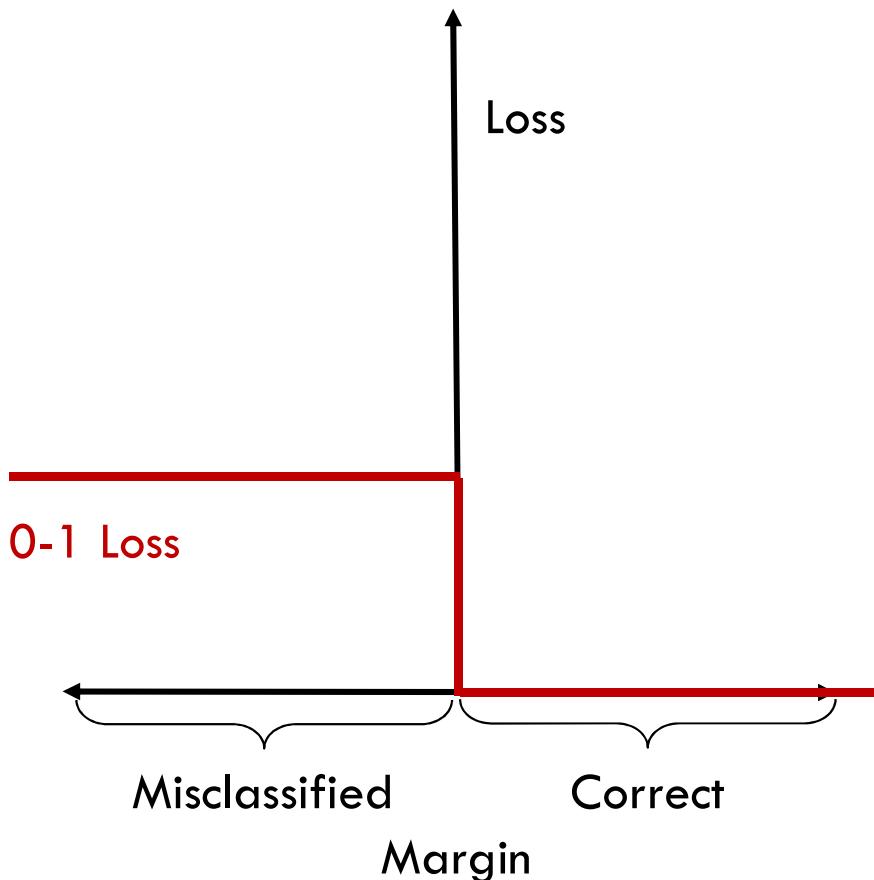
0 – 1 Loss Function

- The 0 – 1 Loss multiplies misclassified points by 1
- Correctly classified points are ignored
- Theoretical "ideal" loss function
- Difficult to optimize—non-smooth



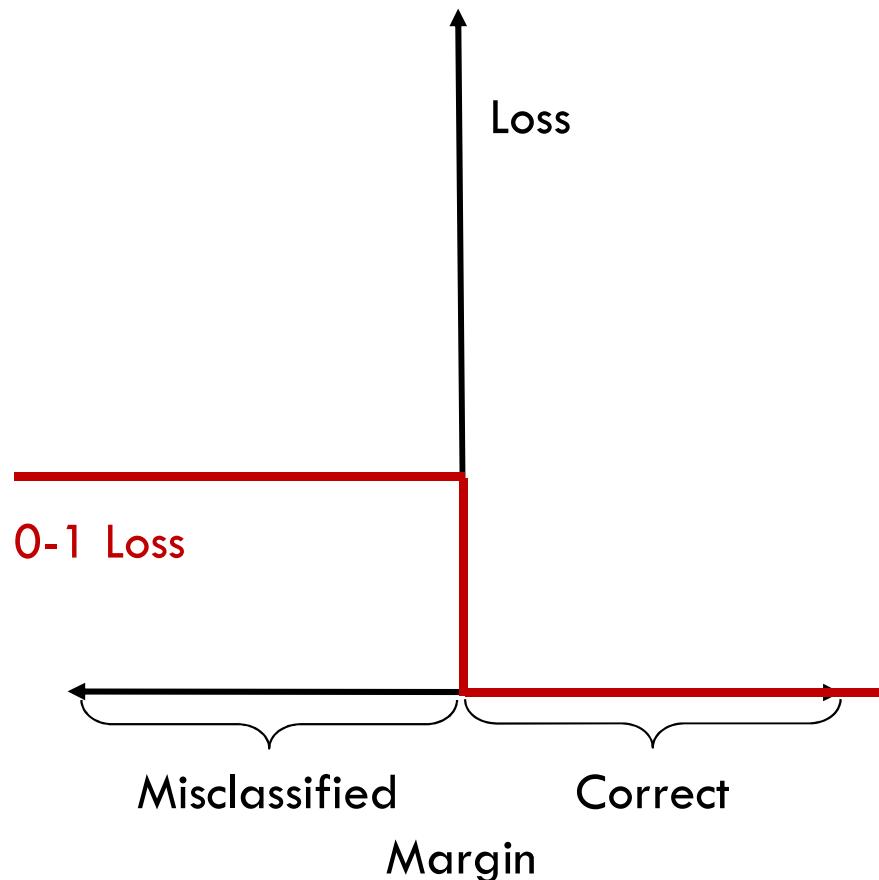
0 – 1 Loss Function

- The 0 – 1 Loss multiplies misclassified points by 1
- Correctly classified points are ignored
- Theoretical "ideal" loss function
- Difficult to optimize—non-smooth and non-convex



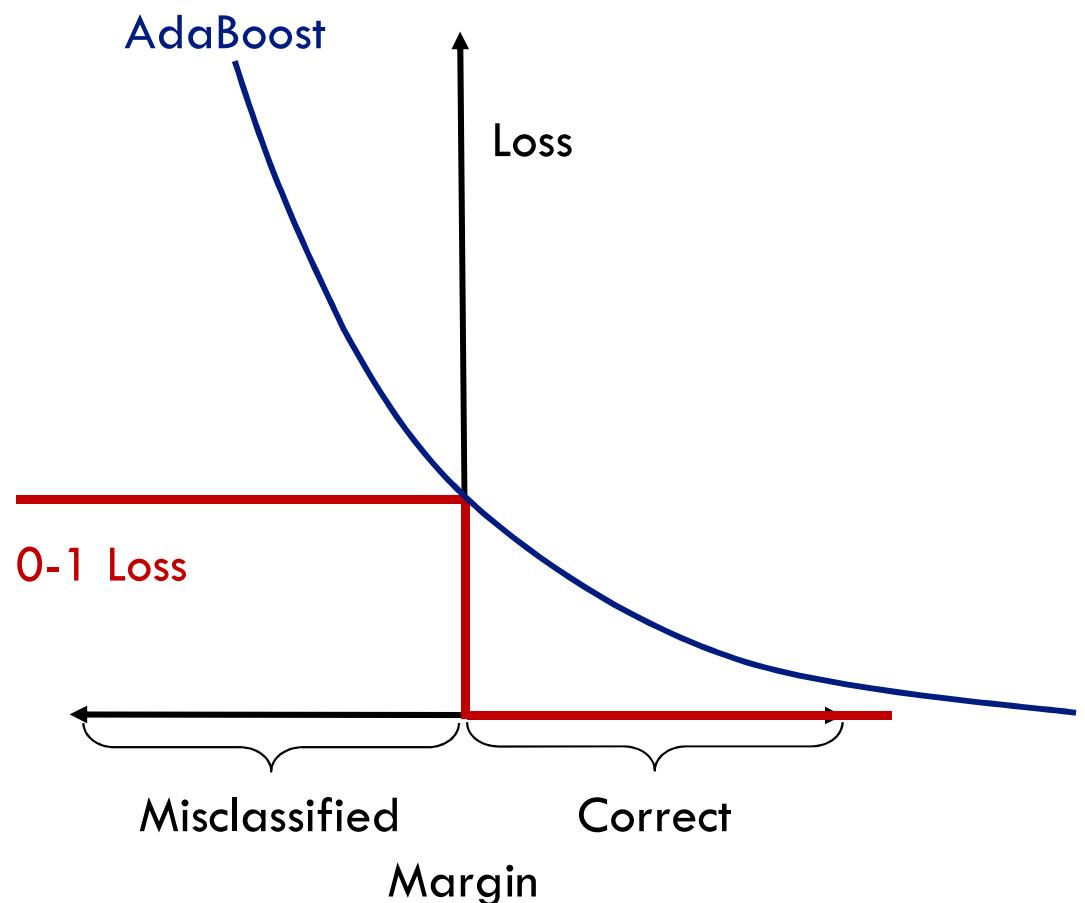
AdaBoost Loss Function

- AdaBoost = Adaptive Boosting



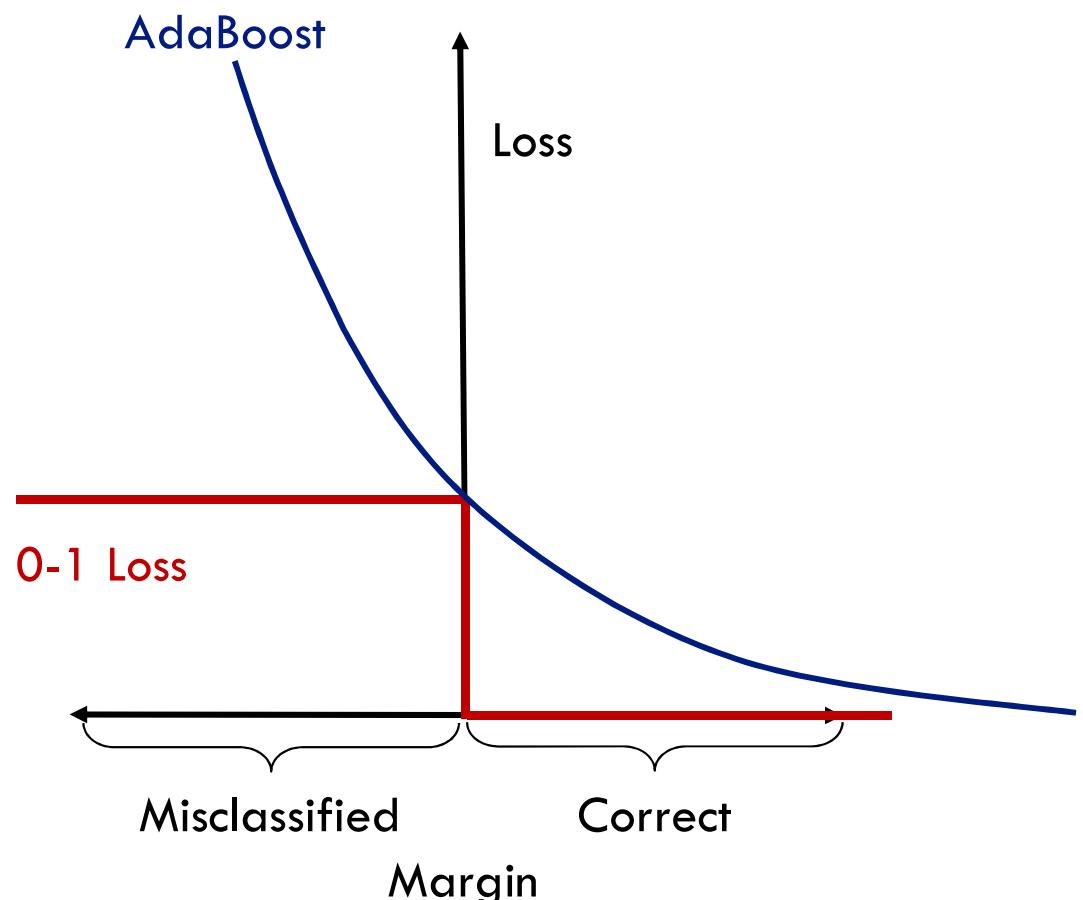
AdaBoost Loss Function

- AdaBoost = Adaptive Boosting
- Loss function is exponential:
 $e^{(-\text{margin})}$



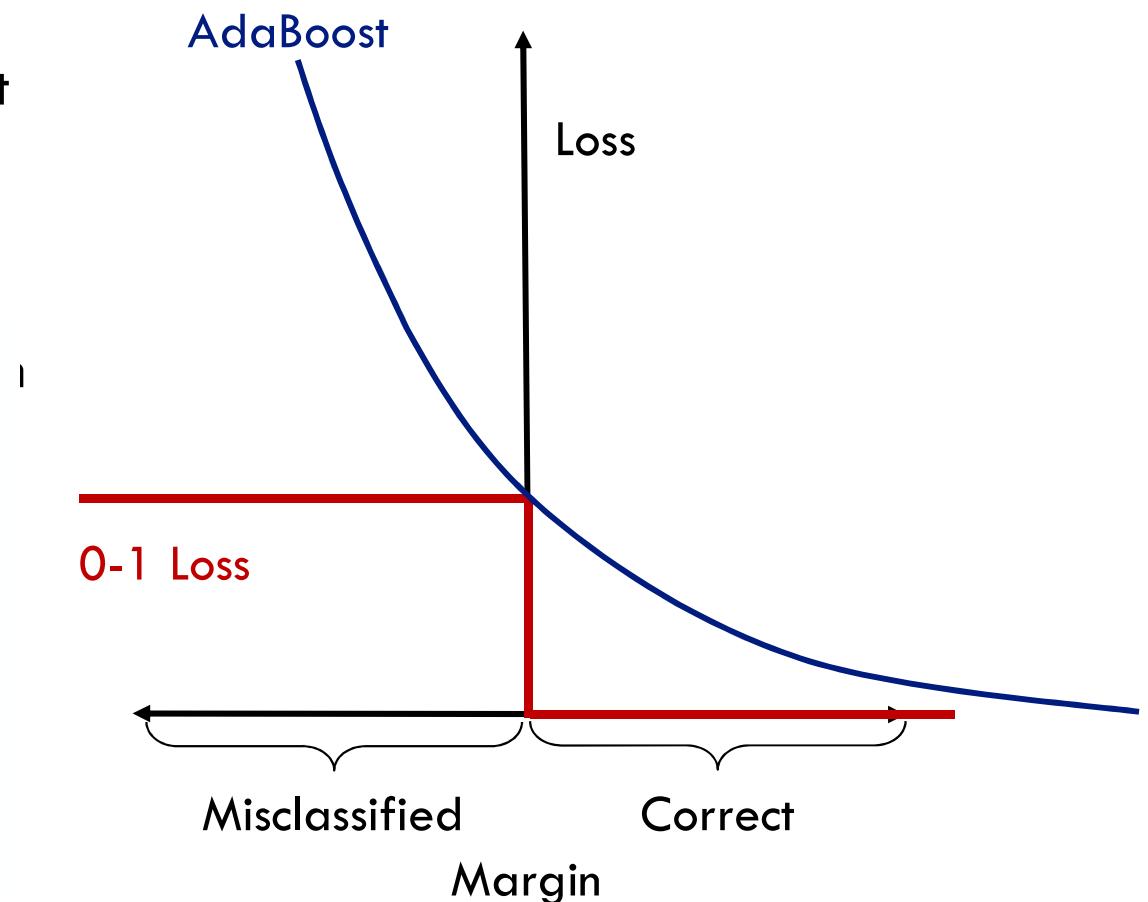
AdaBoost Loss Function

- AdaBoost = Adaptive Boosting
- Loss function is exponential:
 $e^{(-\text{margin})}$
- Makes AdaBoost more sensitive to outliers than other types of boosting



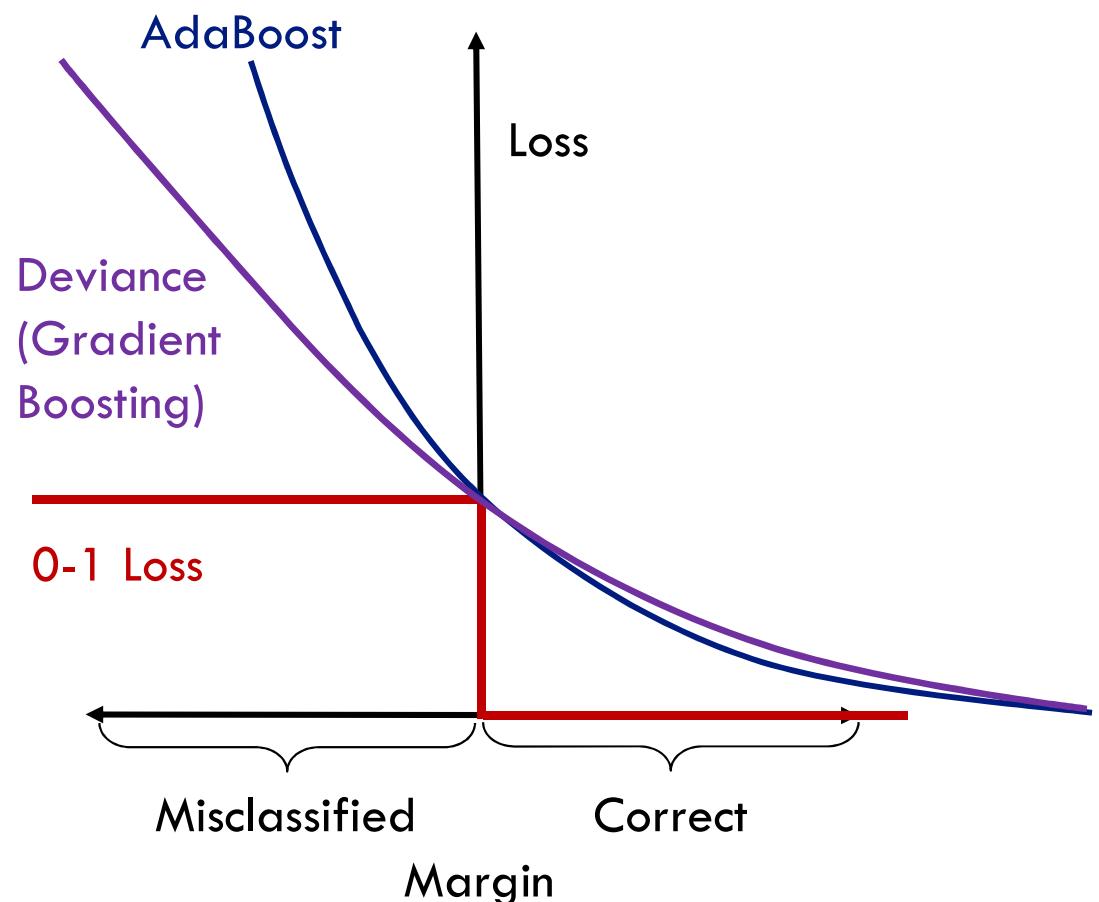
Gradient Boosting Loss Function

- Generalized boosting method that can use different loss functions
- Common implementation uses



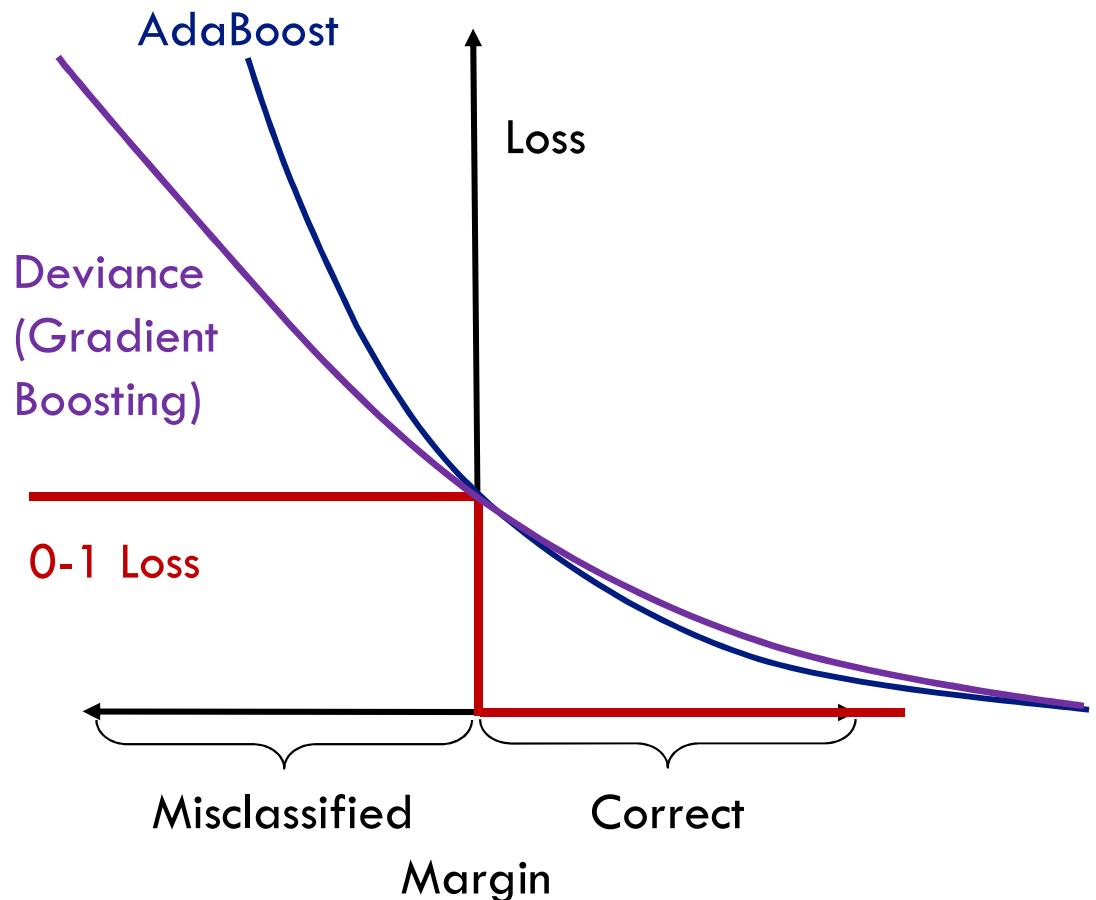
Gradient Boosting Loss Function

- Generalized boosting method that can use different loss functions
- Common implementation uses binomial log likelihood loss function (deviance):
 $\log(1 + e^{(-\text{margin})})$



Gradient Boosting Loss Function

- Generalized boosting method that can use different loss functions
- Common implementation uses binomial log likelihood loss function (deviance):
 $\log(1 + e^{(-\text{margin})})$
- More robust to outliers than AdaBoost



Bagging vs Boosting

Bagging

- Bootstrapped samples

Boosting

- Fit entire data set

Bagging vs Boosting

Bagging

- Bootstrapped samples
- Base trees created independently

Boosting

- Fit entire data set
- Base trees created successively

Bagging vs Boosting

Bagging

- Bootstrapped samples
- Base trees created independently
- Only data points considered

Boosting

- Fit entire data set
- Base trees created successively
- Use residuals from previous models

Bagging vs Boosting

Bagging

- Bootstrapped samples
- Base trees created independently
- Only data points considered
- No weighting used
- Excess trees will not overfit

Boosting

- Fit entire data set
- Base trees created successively
- Use residuals from previous models
- Up-weight misclassified points
- Beware of overfitting

Bagging vs Boosting

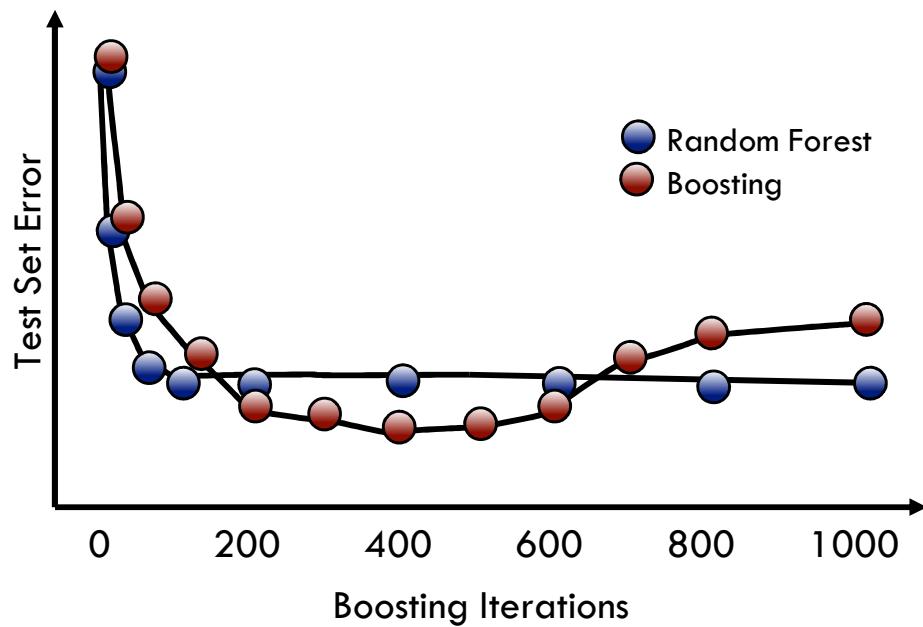
Bagging

- Bootstrapped samples
- Base trees created independently
- Only data points considered
- No weighting used
- Excess trees will not overfit

Boosting

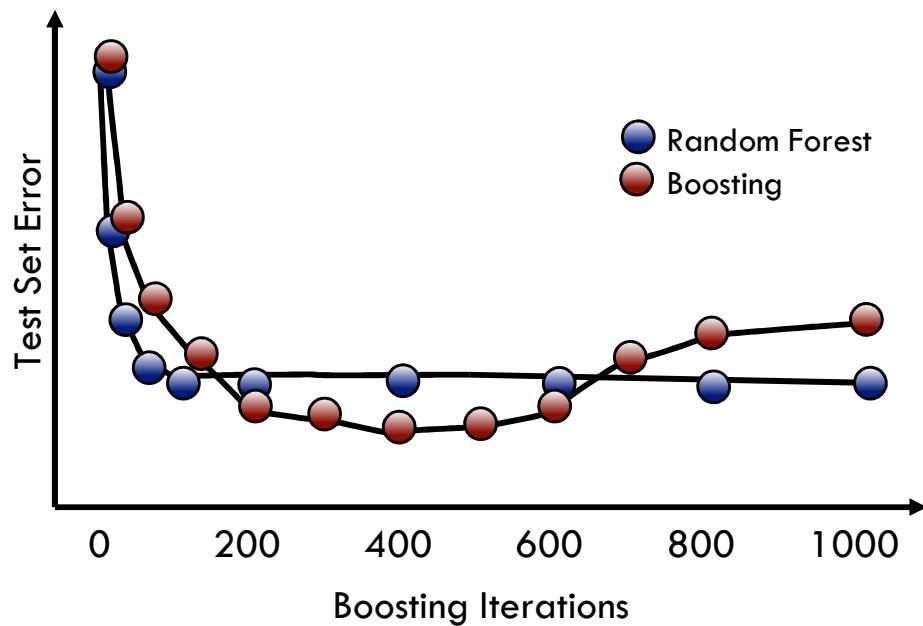
- Fit entire data set
- Base trees created successively
- Use residuals from previous models
- Up-weight misclassified points
- Beware of overfitting

Tuning a Gradient Boosted Model



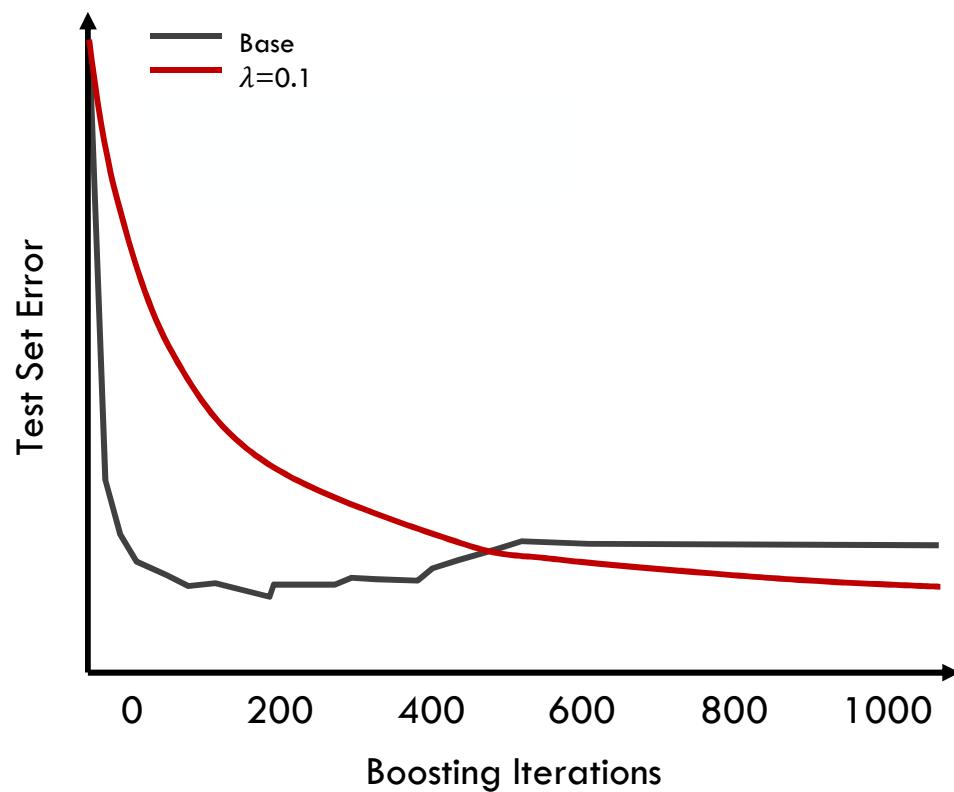
- Boosting is additive, so possible to **overfit**

Tuning a Gradient Boosted Model



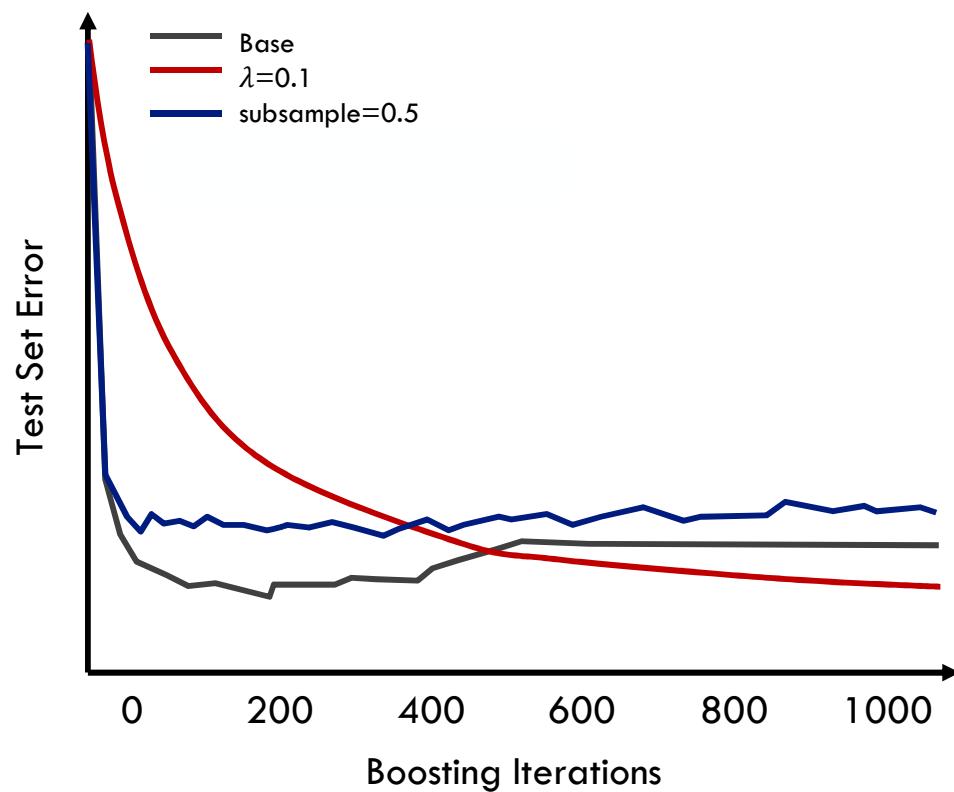
- Boosting is additive, so possible to **overfit**
- Use cross validation to set number of trees

Tuning a Gradient Boosted Model



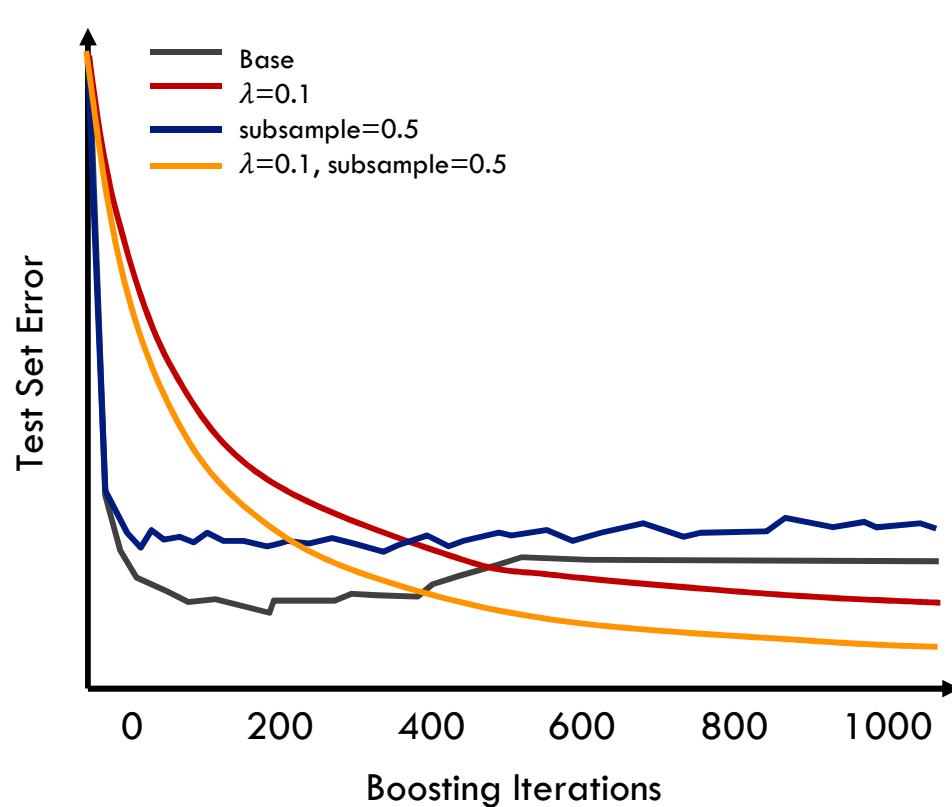
- **Learning rate (λ):** set to < 1.0 for regularization. That's also called “shrinkage”

Tuning a Gradient Boosted Model



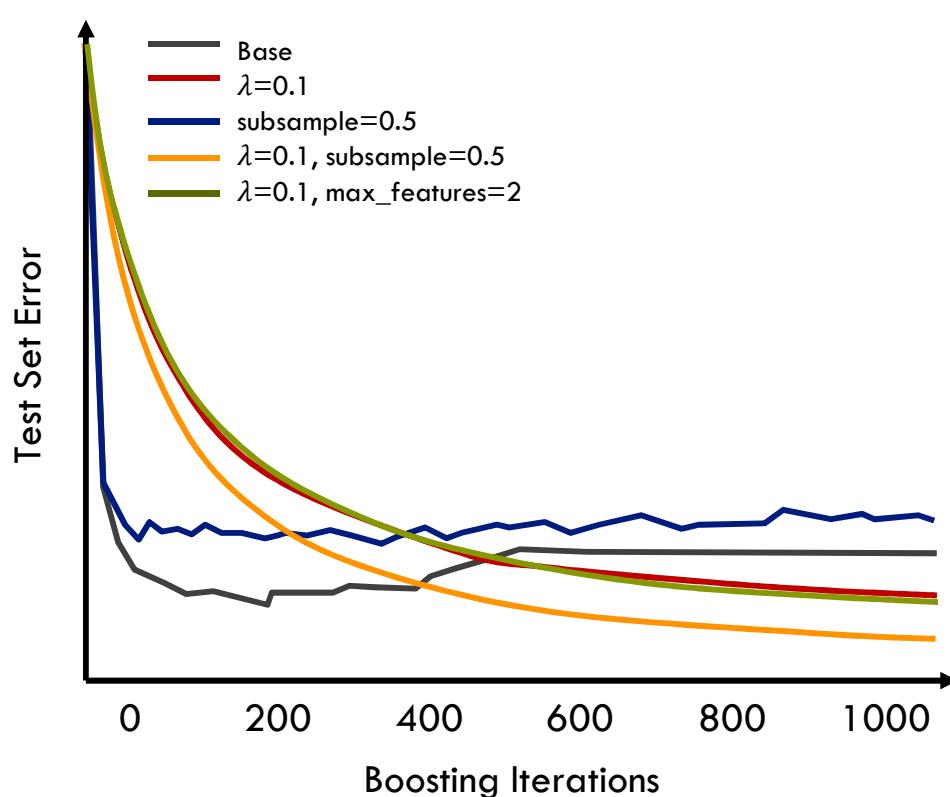
- **Learning rate (λ):** set to <1.0 for regularization. That's also called "shrinkage"
- **Subsample:** set to <1.0 to use fraction of data for base learners (stochastic gradient boosting)

Tuning a Gradient Boosted Model



- **Learning rate (λ):** set to < 1.0 for regularization. That's also called “shrinkage”
- **Subsample:** set to < 1.0 to use fraction of data for base learners (stochastic gradient boosting)

Tuning a Gradient Boosted Model



- **Learning rate (λ):** set to < 1.0 for regularization. That's also called "shrinkage"
- **Subsample:** set to < 1.0 to use fraction of data for base learners (stochastic gradient boosting)
- **Max_features:** number of features to consider in base learners when splitting.

GradientBoostingClassifier: The Syntax

Import the class containing the classification method

```
from sklearn.ensemble import GradientBoostingClassifier
```

GradientBoostingClassifier: The Syntax

Import the class containing the classification method

```
from sklearn.ensemble import GradientBoostingClassifier
```

Create an instance of the class

```
GBC = GradientBoostingClassifier(learning_rate=0.1,  
                                 max_features=1, subsample=0.5,  
                                 n_estimators=200)
```

GradientBoostingClassifier: The Syntax

Import the class containing the classification method

```
from sklearn.ensemble import GradientBoostingClassifier
```

Create an instance of the class

```
GBC = GradientBoostingClassifier(learning_rate=0.1,  
                                 max_features=1, subsample=0.5,  
                                 n_estimators=200)
```

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

```
GBC = GBC.fit (X_train, y_train)  
y_predict = GBC.predict(X_test)
```

GradientBoostingClassifier: The Syntax

Import the class containing the classification method

```
from sklearn.ensemble import GradientBoostingClassifier
```

Create an instance of the class

```
GBC = GradientBoostingClassifier(learning_rate=0.1,  
                                 max_features=1, subsample=0.5,  
                                 n_estimators=200)
```

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

```
GBC = GBC.fit (X_train, y_train)  
y_predict = GBC.predict(X_test)
```

Tune with cross-validation. Use GradientBoostingRegressor for regression.

AdaBoostClassifier: The Syntax

Import the class containing the classification method

```
from sklearn.ensemble import AdaBoostClassifier  
from sklearn.tree import DecisionTreeClassifier
```

AdaBoostClassifier: The Syntax

Import the class containing the classification method

```
from sklearn.ensemble import AdaBoostClassifier  
from sklearn.tree import DecisionTreeClassifier
```

Create an instance of the class

```
ABC = AdaBoostClassifier(base_estimator=DecisionTreeClassifier(),  
                        learning_rate=0.1, n_estimators=200)
```

AdaBoostClassifier: The Syntax

Import the class containing the classification method

```
from sklearn.ensemble import AdaBoostClassifier  
from sklearn.tree import DecisionTreeClassifier
```

Create an instance of the class

```
ABC = AdaBoostClassifier(base_estimator=DecisionTreeClassifier(),  
                        learning_rate=0.1, n_estimators=200)
```

base learner can
be set manually



AdaBoostClassifier: The Syntax

Import the class containing the classification method

```
from sklearn.ensemble import AdaBoostClassifier  
from sklearn.tree import DecisionTreeClassifier
```

Create an instance of the class

```
ABC = AdaBoostClassifier(base_estimator=DecisionTreeClassifier(),  
                        learning_rate=0.1, n_estimators=200)
```

can also set max
depth here



AdaBoostClassifier: The Syntax

Import the class containing the classification method

```
from sklearn.ensemble import AdaBoostClassifier  
from sklearn.tree import DecisionTreeClassifier
```

Create an instance of the class

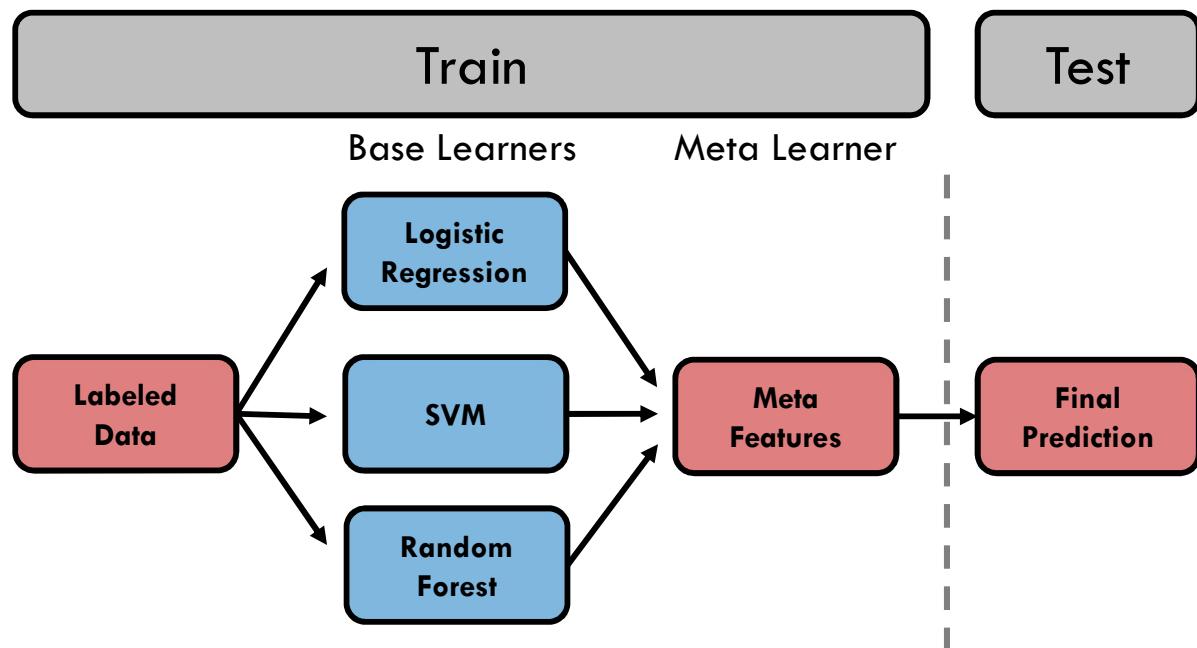
```
ABC = AdaBoostClassifier(base_estimator=DecisionTreeClassifier(),  
                        learning_rate=0.1, n_estimators=200)
```

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

```
ABC = ABC.fit(X_train, y_train)  
y_predict = ABC.predict(X_test)
```

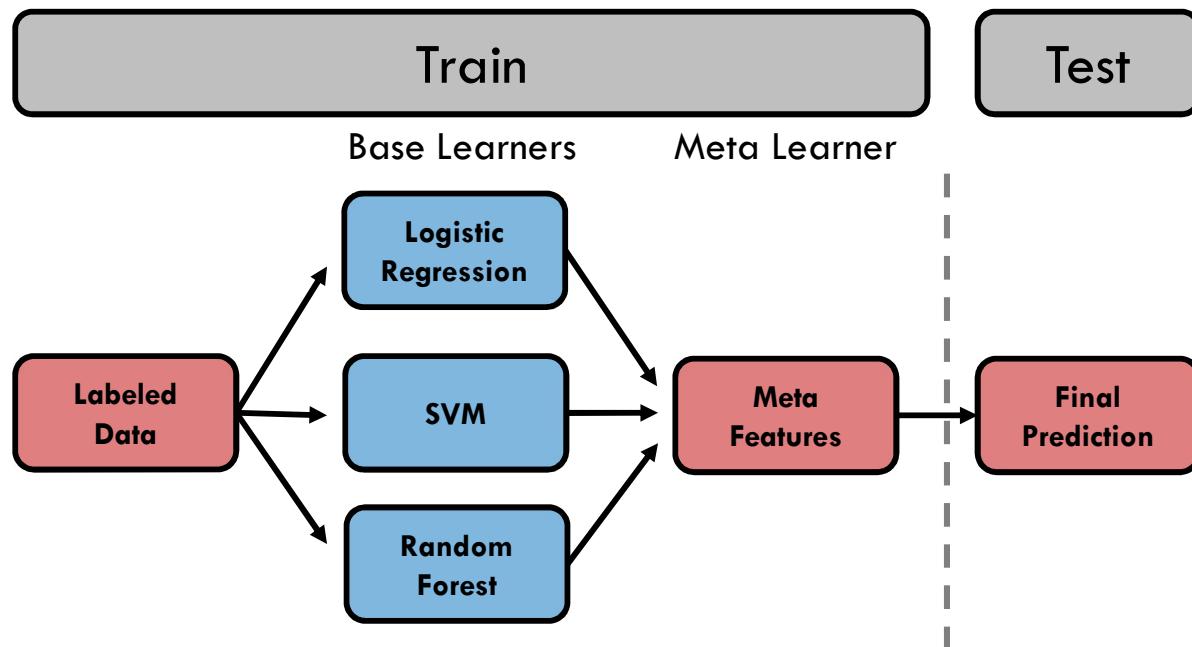
Tune parameters with cross-validation. Use AdaBoostRegressor for regression.

Stacking: Combining Heterogeneous Classifiers



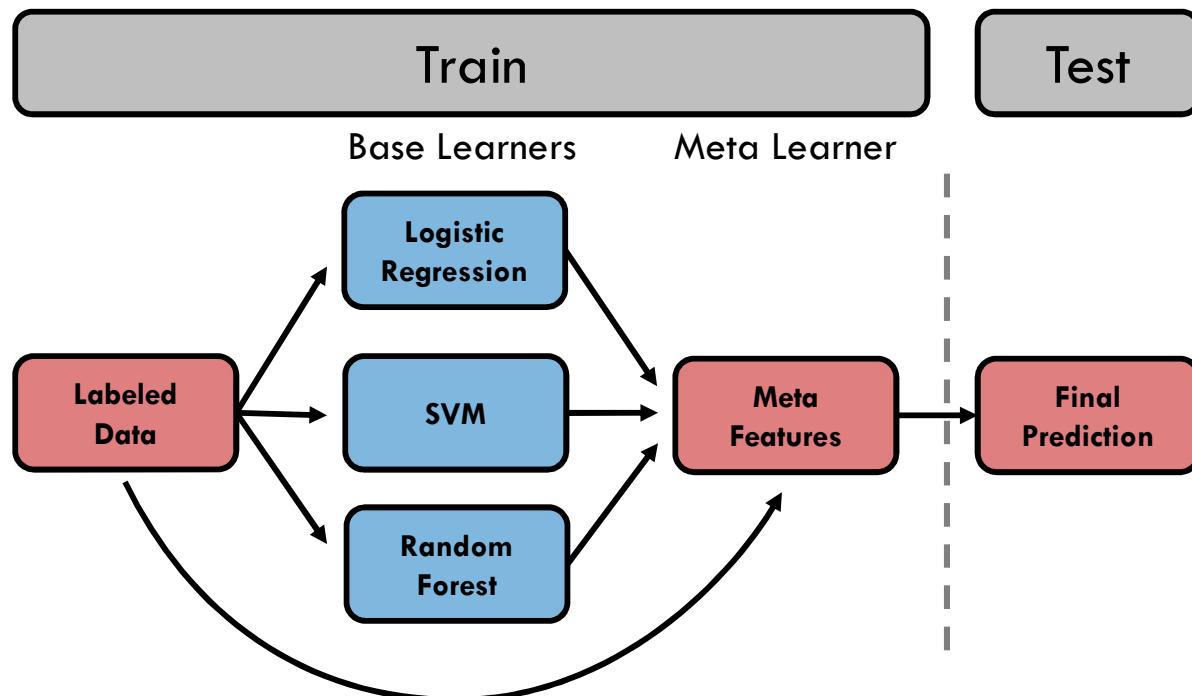
- Models of any kind combined to create stacked model

Stacking: Combining Heterogeneous Classifiers



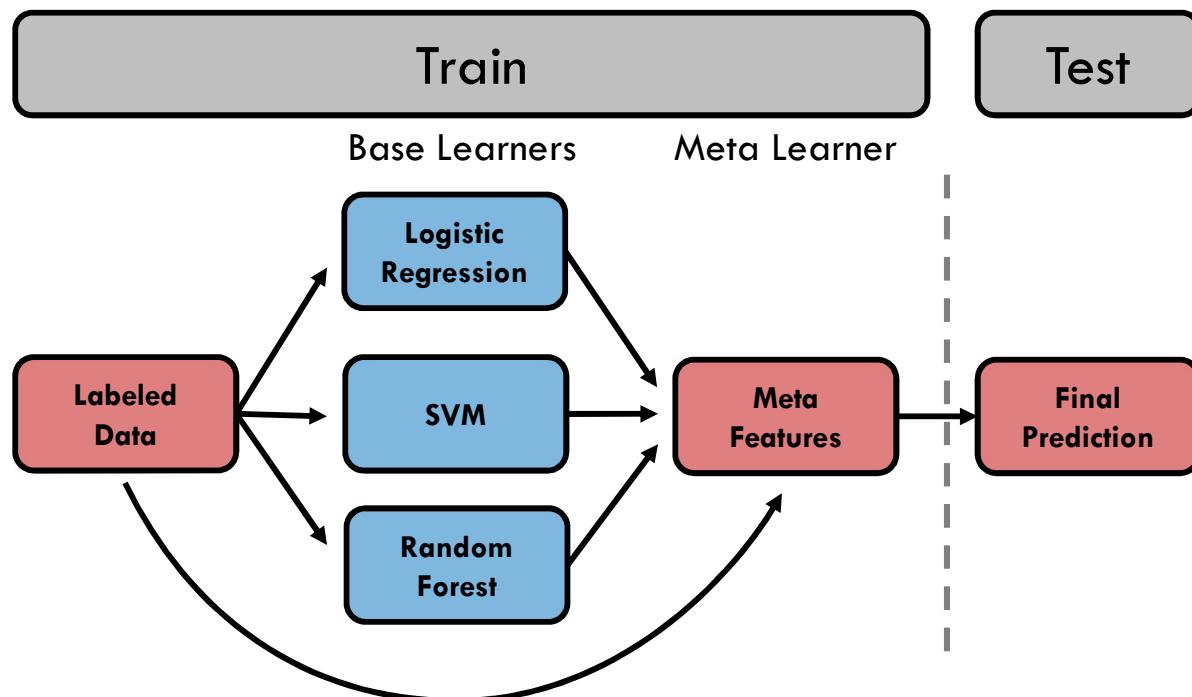
- Models of any kind combined to create stacked model
- Like bagging but not limited to decision trees

Stacking: Combining Heterogeneous Classifiers



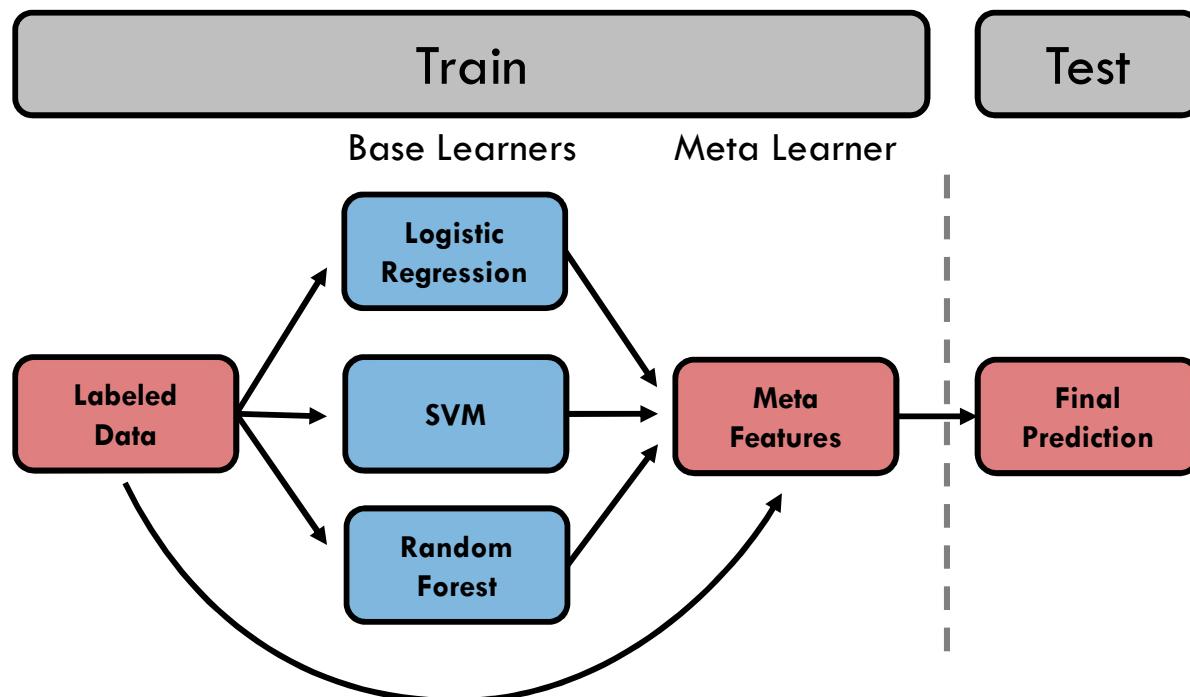
- Models of any kind combined to create stacked model
- Like bagging but not limited to decision trees
- Output of base learners creates features, can recombine with data

Stacking: Combining Heterogeneous Classifiers



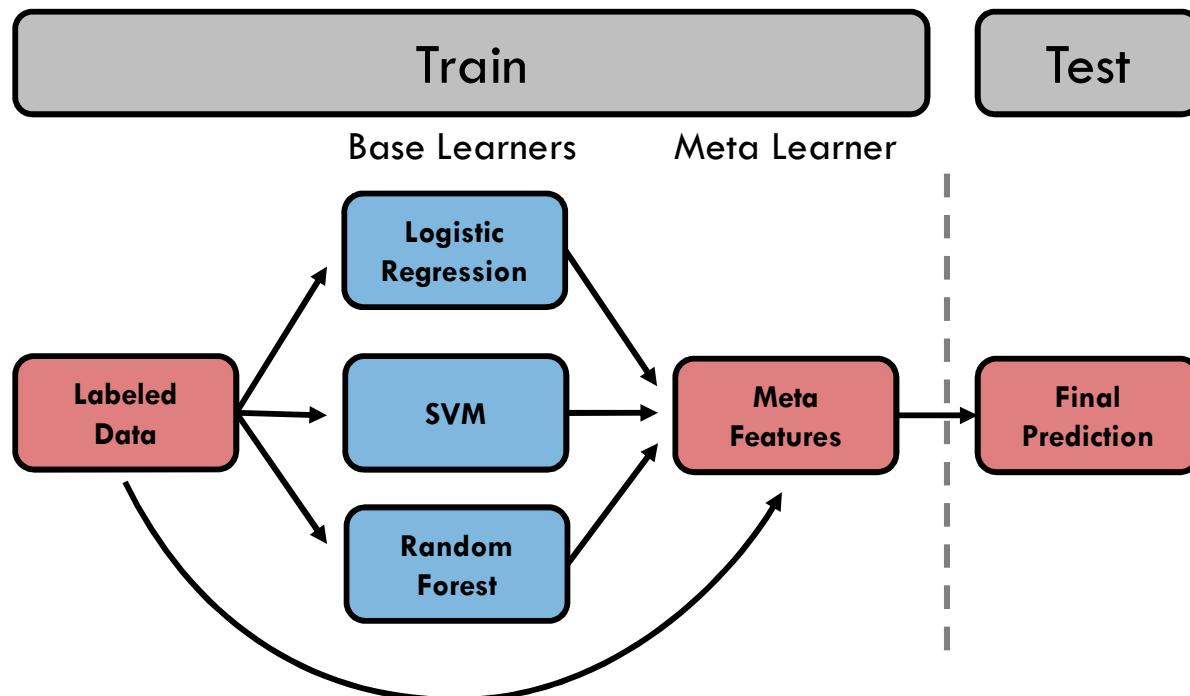
- Output of base learners can be combined via majority vote or weighted

Stacking: Combining Heterogeneous Classifiers



- Output of base learners can be combined via majority vote or weighted
- Additional hold-out data needed if meta learner parameters are used

Stacking: Combining Heterogeneous Classifiers



- Output of base learners can be combined via majority vote or weighted
- Additional hold-out data needed if meta learner parameters are used
- Be aware of increasing model complexity

VotingClassifier: The Syntax

Import the class containing the classification method

```
from sklearn.ensemble import VotingClassifier
```

VotingClassifier: The Syntax

Import the class containing the classification method

```
from sklearn.ensemble import VotingClassifier
```

Create an instance of the class

```
VC = VotingClassifier(estimator_list, voting='hard',  
                     weights=estimator_weight_list)
```

VotingClassifier: The Syntax

Import the class containing the classification method

```
from sklearn.ensemble import VotingClassifier
```

Create an instance of the class

```
VC = VotingClassifier(estimator_list, voting='hard',  
                     weights=estimator_weight_list)
```

list of fitted
models and
how to
combine



VotingClassifier: The Syntax

Import the class containing the classification method

```
from sklearn.ensemble import VotingClassifier
```

Create an instance of the class

```
VC = VotingClassifier(estimator_list, voting='hard',  
                      weights=estimator_weight_list)
```

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

```
VC = VC.fit(X_train, y_train)  
y_predict = VC.predict(X_test)
```

Tune with an ADDITIONAL LEVEL of cross-validation or hold-out set.