

Software

# Introduction to

---

# Supervised Learning

# Legal Notices and Disclaimers

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

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

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

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

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

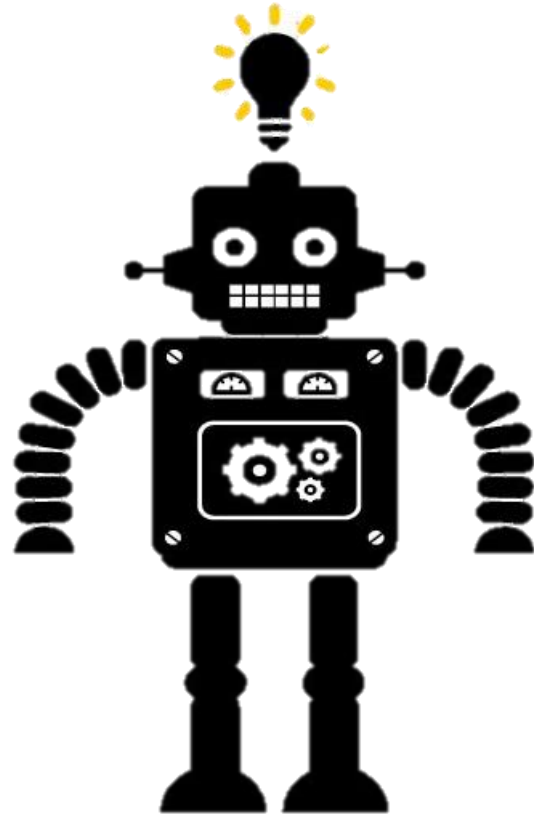
Copyright © 2021, Intel Corporation. All rights reserved.

# Learning Objectives

- Explain supervised learning and how it can be applied to regression and classification problems
- Apply K-Nearest Neighbor (KNN) algorithm for classification
- Apply Intel® Extension for Scikit-learn\* to leverage underlying compute capabilities of hardware

# What is Machine Learning?

Machine learning allows computers to learn and infer from data.



# Machine Learning in Our Daily Lives

Spam Filtering

# Machine Learning in Our Daily Lives

Spam Filtering

Web Search

# Machine Learning in Our Daily Lives

Spam Filtering

Web Search

Postal Mail Routing

# Machine Learning in Our Daily Lives

Spam Filtering

Web Search

Postal Mail Routing

Fraud Detection

Movie  
Recommendations

Vehicle Driver  
Assistance

Web Advertisements

Social Networks

Speech Recognition



# Types of Machine Learning

Supervised

data points have known outcome

# Types of Machine Learning

Supervised

data points have known outcome

Unsupervised

data points have unknown outcome

# Types of Machine Learning

**Supervised**

data points have known outcome

**Unsupervised**

data points have unknown outcome

# Types of Supervised Learning

Regression

outcome is continuous (numerical)

# Types of Supervised Learning

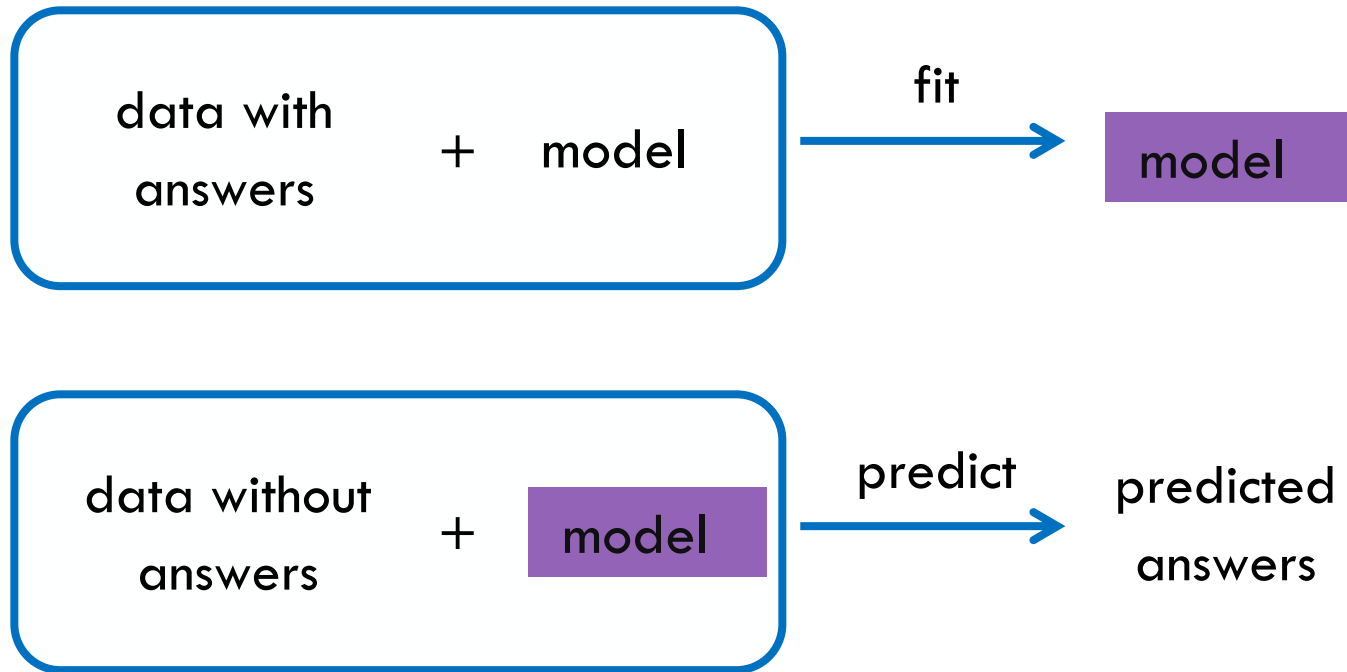
Regression

outcome is continuous (numerical)

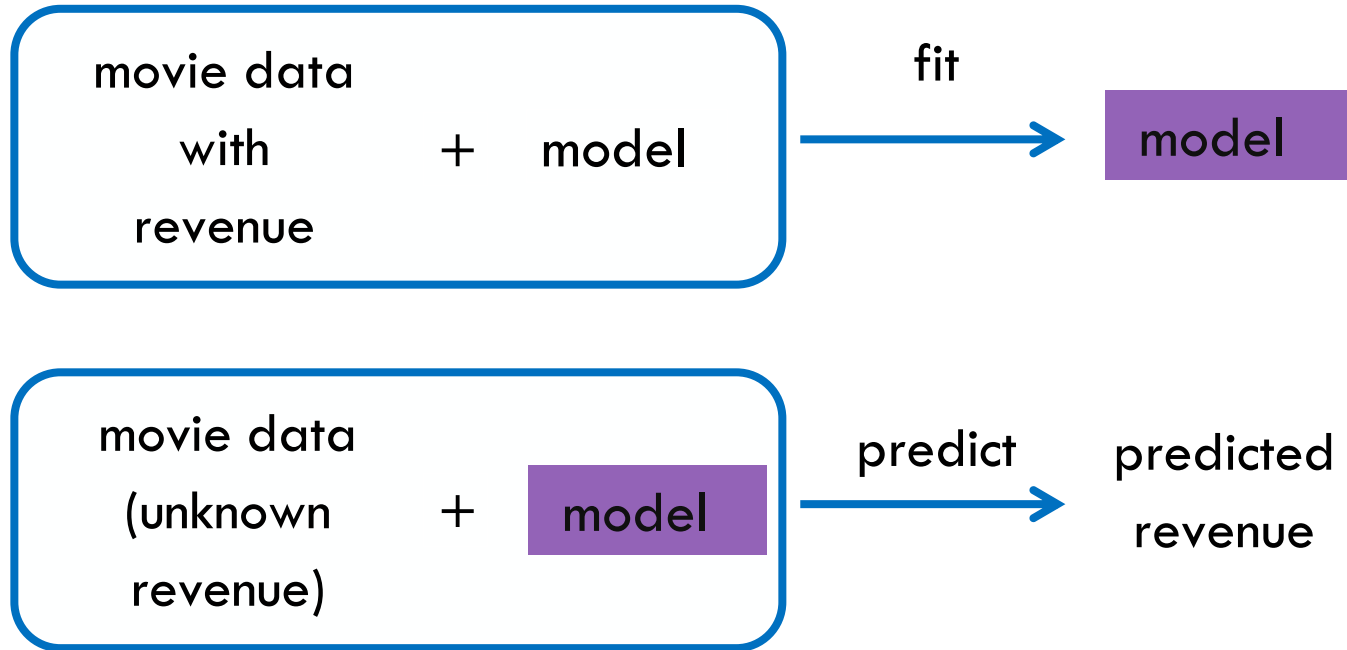
Classification

outcome is a category

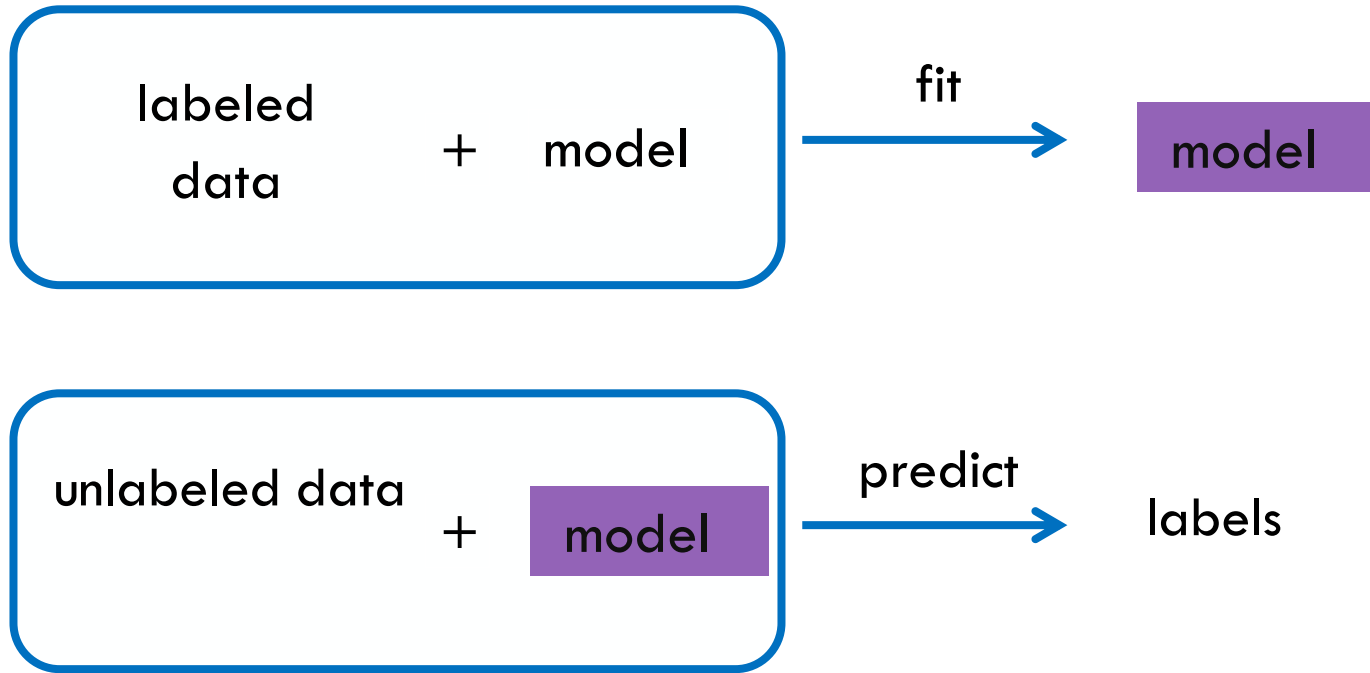
# Supervised Learning Overview



# Regression: Numeric Answers

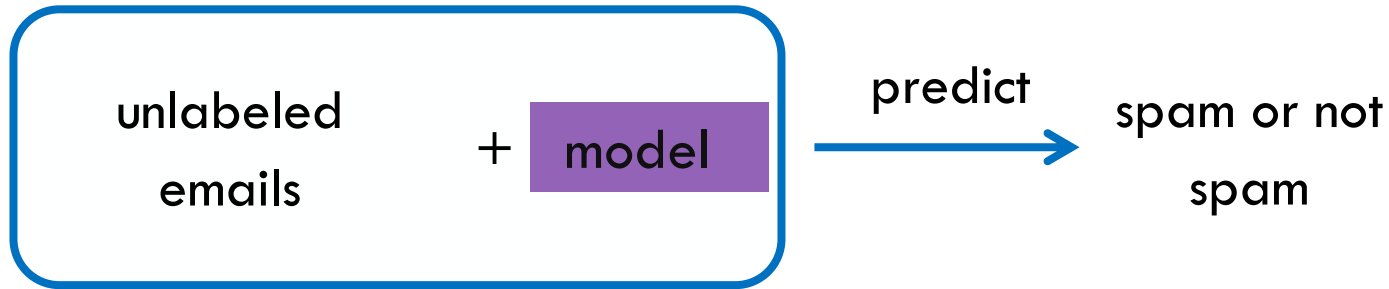
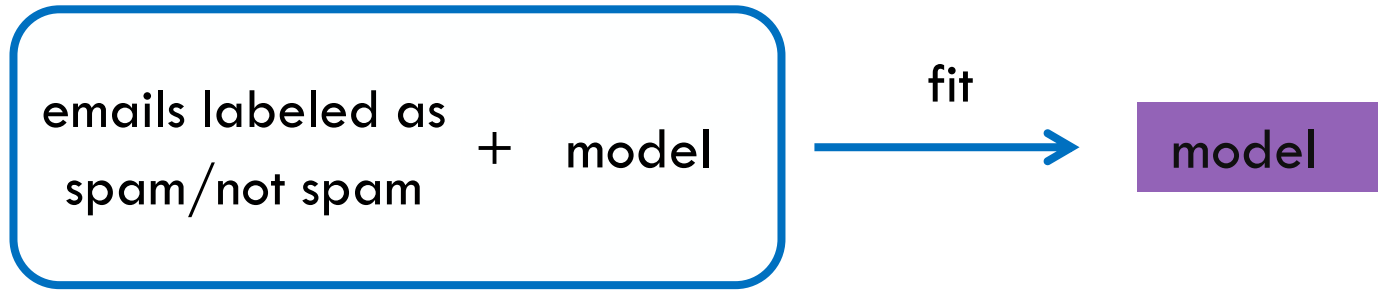


# Classification: Categorical Answers





# Classification: Categorical Answers



# Machine Learning Vocabulary

- **Target:** predicted category or value of the data  
(column to predict)

# Machine Learning Vocabulary

sepal length	sepal width	petal length	petal width	species
6.7	3.0	5.2	2.3	virginica
6.4	2.8	5.6	2.1	virginica
4.6	3.4	1.4	0.3	setosa
6.9	3.1	4.9	1.5	versicolor
4.4	2.9	1.4	0.2	setosa
4.8	3.0	1.4	0.1	setosa
5.9	3.0	5.1	1.8	virginica
5.4	3.9	1.3	0.4	setosa
4.9	3.0	1.4	0.2	setosa
5.4	3.4	1.7	0.2	setosa

# Machine Learning Vocabulary

Target


sepal length	sepal width	petal length	petal width	species
6.7	3.0	5.2	2.3	virginica
6.4	2.8	5.6	2.1	virginica
4.6	3.4	1.4	0.3	setosa
6.9	3.1	4.9	1.5	versicolor
4.4	2.9	1.4	0.2	setosa
4.8	3.0	1.4	0.1	setosa
5.9	3.0	5.1	1.8	virginica
5.4	3.9	1.3	0.4	setosa
4.9	3.0	1.4	0.2	setosa
5.4	3.4	1.7	0.2	setosa

# Machine Learning Vocabulary

- **Target:** predicted category or value of the data  
(column to predict)
- **Features:** properties of the data used for prediction  
(non-target columns)

# Machine Learning Vocabulary

Features



sepal length	sepal width	petal length	petal width	species
6.7	3.0	5.2	2.3	virginica
6.4	2.8	5.6	2.1	virginica
4.6	3.4	1.4	0.3	setosa
6.9	3.1	4.9	1.5	versicolor
4.4	2.9	1.4	0.2	setosa
4.8	3.0	1.4	0.1	setosa
5.9	3.0	5.1	1.8	virginica
5.4	3.9	1.3	0.4	setosa
4.9	3.0	1.4	0.2	setosa
5.4	3.4	1.7	0.2	setosa

# Machine Learning Vocabulary

- **Target:** predicted category or value of the data  
(column to predict)
- **Features:** properties of the data used for prediction  
(non-target columns)
- **Example:** a single data point within the data (one row)

# Machine Learning Vocabulary

Example →

sepal length	sepal width	petal length	petal width	species
6.7	3.0	5.2	2.3	virginica
6.4	2.8	5.6	2.1	virginica
4.6	3.4	1.4	0.3	setosa
6.9	3.1	4.9	1.5	versicolor
4.4	2.9	1.4	0.2	setosa
4.8	3.0	1.4	0.1	setosa
5.9	3.0	5.1	1.8	virginica
5.4	3.9	1.3	0.4	setosa
4.9	3.0	1.4	0.2	setosa
5.4	3.4	1.7	0.2	setosa



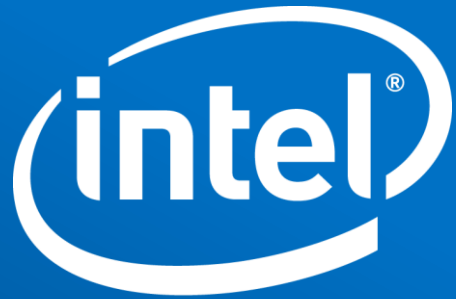
# Machine Learning Vocabulary

- **Target:** predicted category or value of the data  
(column to predict)
- **Features:** properties of the data used for prediction  
(non-target columns)
- **Example:** a single data point within the data (one row)
- **Label:** the target value for a single data point

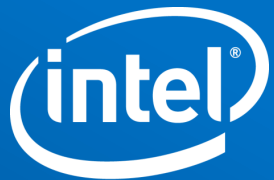
# Machine Learning Vocabulary

Label

sepal length	sepal width	petal length	petal width	species
6.7	3.0	5.2	2.3	virginica
6.4	2.8	5.6	2.1	virginica
4.6	2.4	1.4	0.2	setosa
6.9	3.1	4.9	1.5	versicolor
4.4	2.9	1.4	0.2	setosa
4.8	3.0	1.4	0.1	setosa
5.9	3.0	5.1	1.8	virginica
5.4	3.9	1.3	0.4	setosa
4.9	3.0	1.4	0.2	setosa
5.4	3.4	1.7	0.2	setosa



Software



Software

---

# K – Nearest Neighbors

# What is Classification?

A flower shop wants to guess a customer's purchase from similarity to most recent purchase.



# What is Classification?

Which flower is a customer most likely to purchase based on similarity to previous purchase?



?



# What is Classification?

Which flower is a customer most likely to purchase based on similarity to previous purchase?

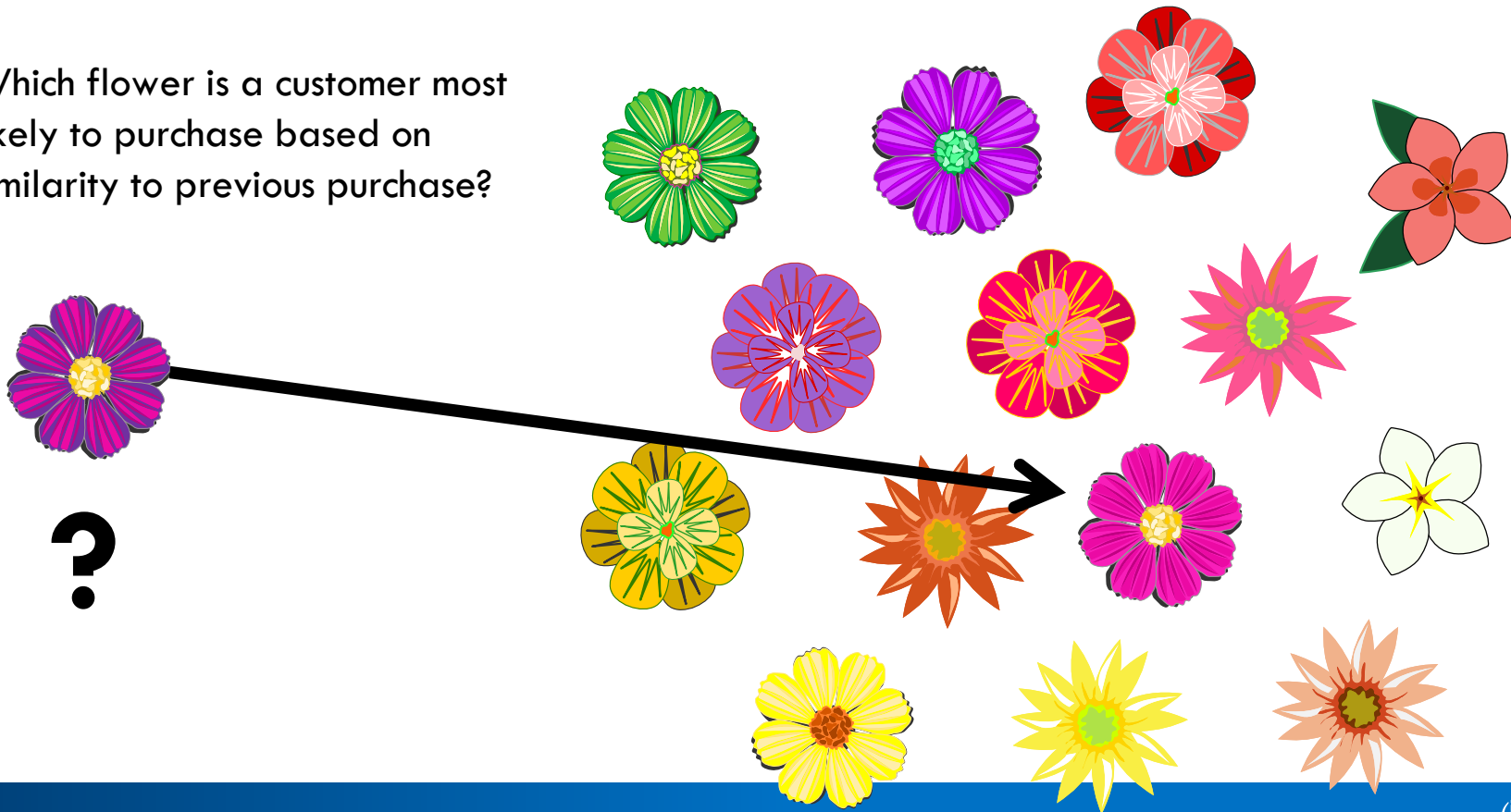


?



# What is Classification?

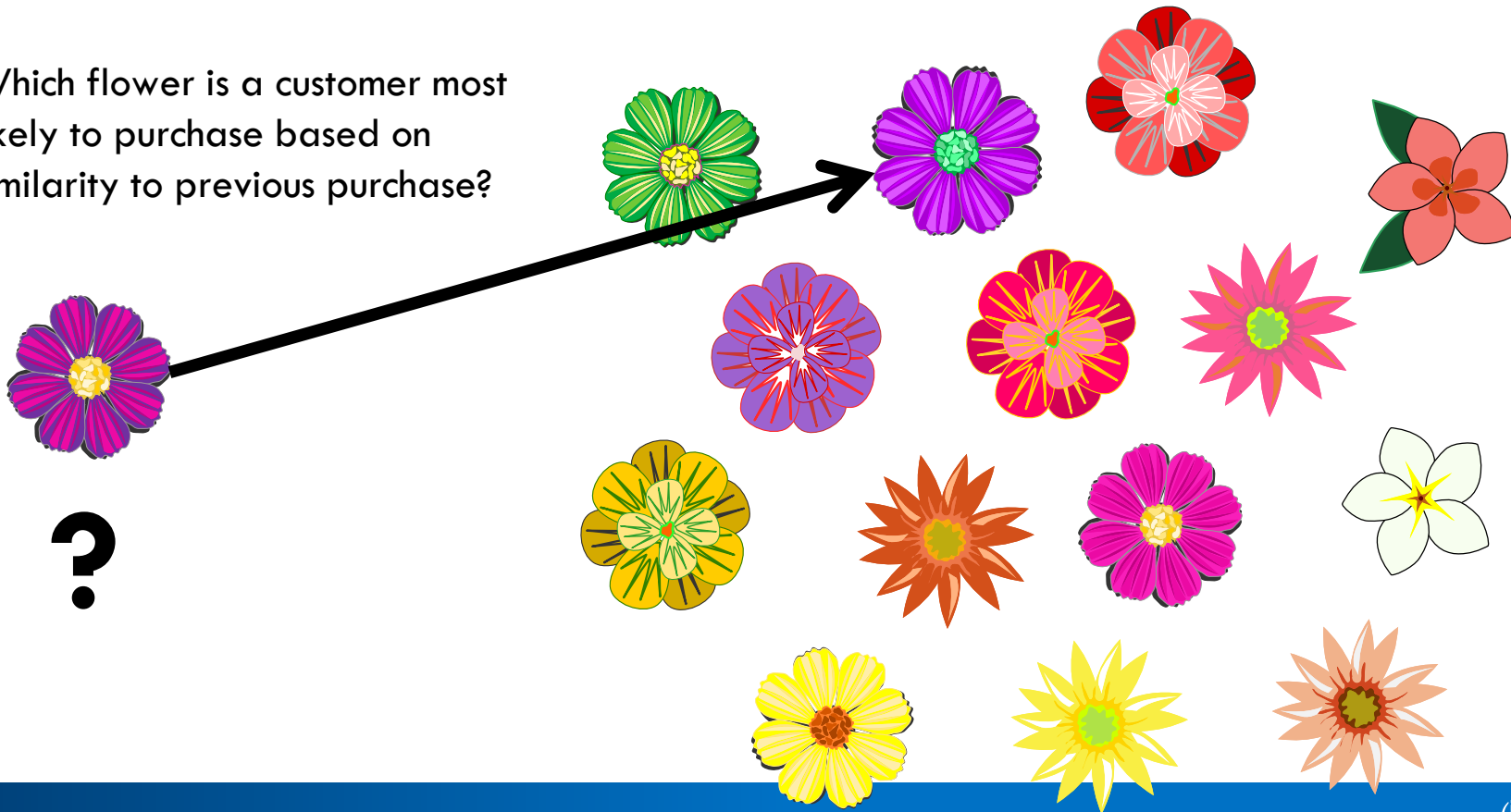
Which flower is a customer most likely to purchase based on similarity to previous purchase?





# What is Classification?

Which flower is a customer most likely to purchase based on similarity to previous purchase?



# What is Needed for Classification?

- Model data with:
  - Features that can be quantitated

# What is Needed for Classification?

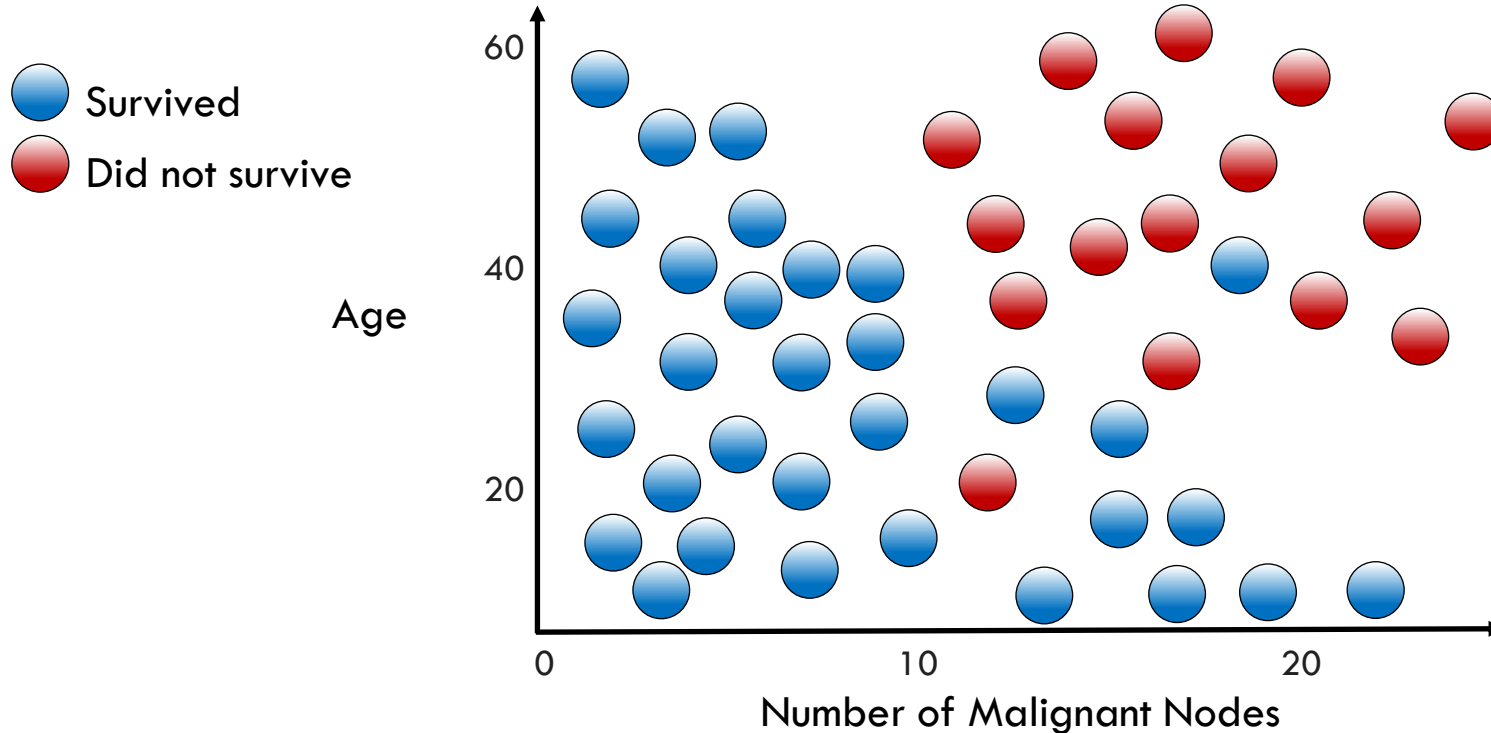
- Model data with:
  - Features that can be quantitated
  - Labels that are known

# What is Needed for Classification?

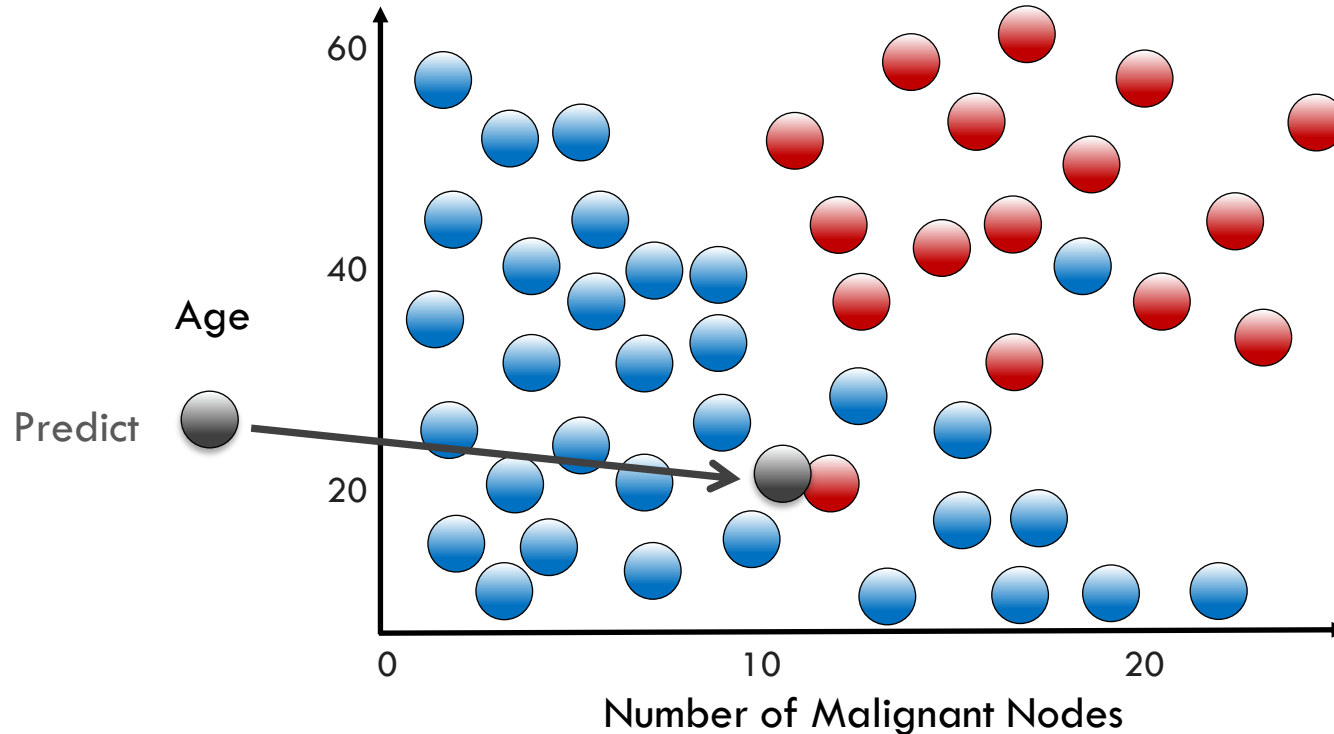
- Model data with:
  - Features that can be quantitated
  - Labels that are known
- Method to measure similarity

# K Nearest Neighbors Classification



# K Nearest Neighbors Classification

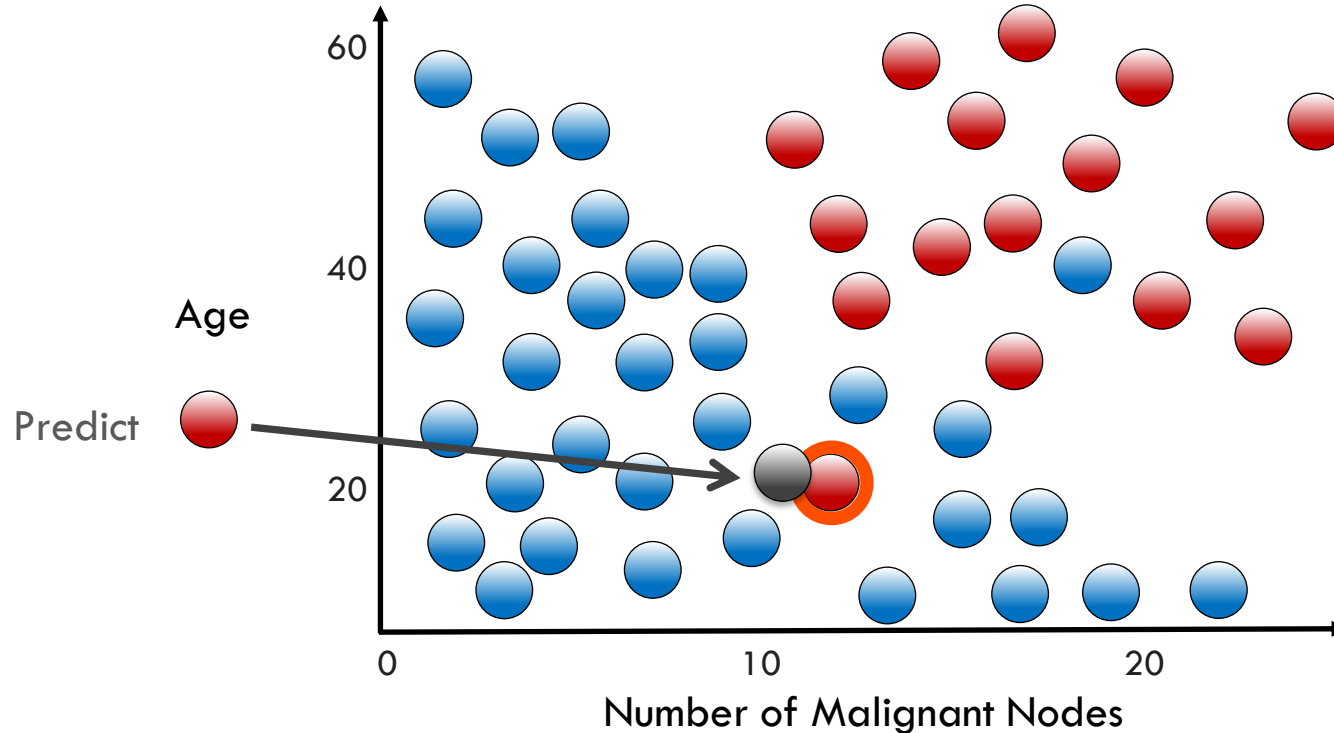


# K Nearest Neighbors Classification





# K Nearest Neighbors Classification

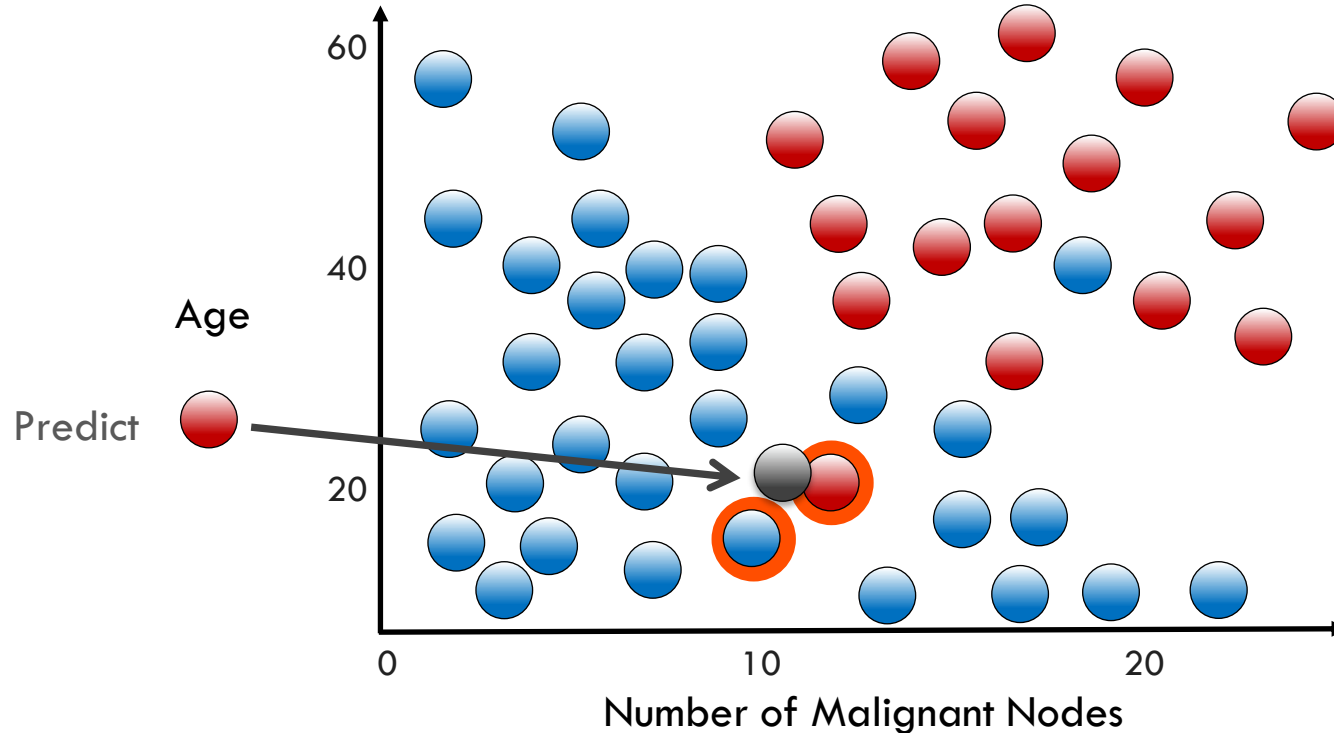
Neighbor Count ( $K = 1$ ):  0  1







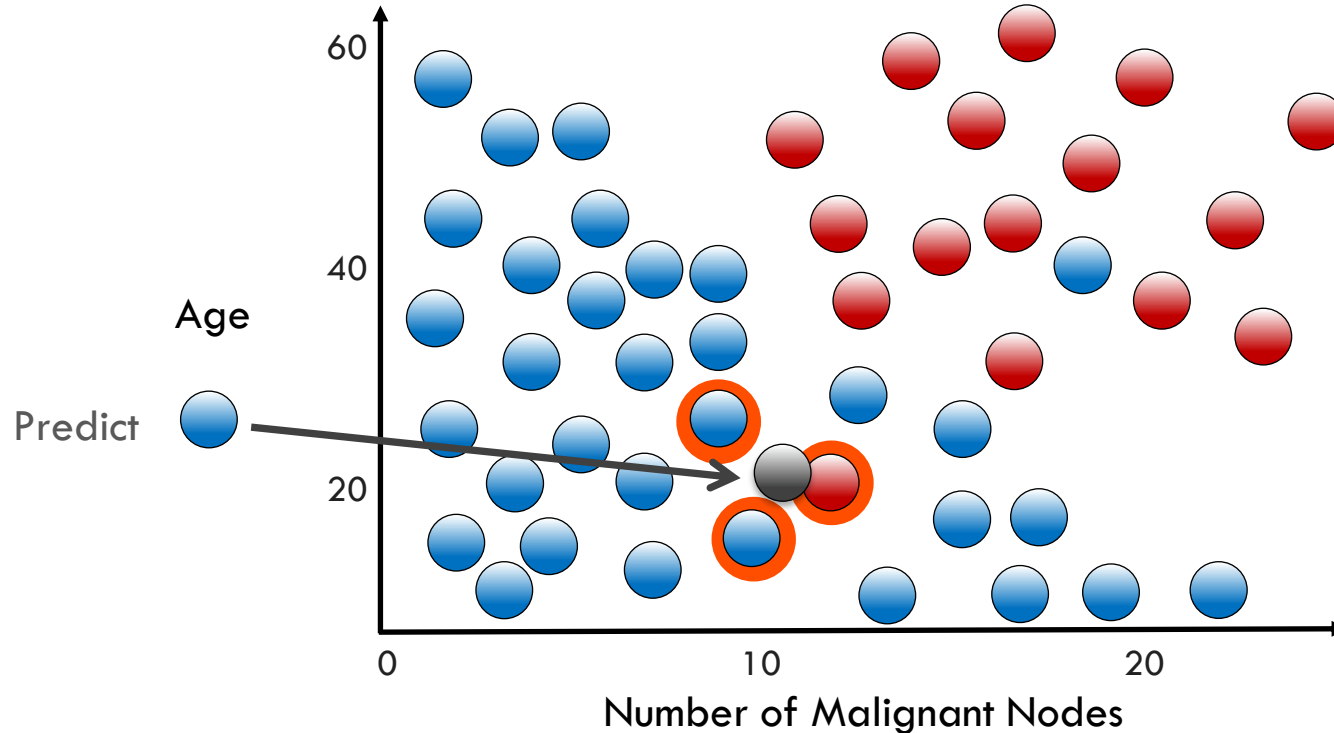
# K Nearest Neighbors Classification

Neighbor Count ( $K = 2$ ):  1  1





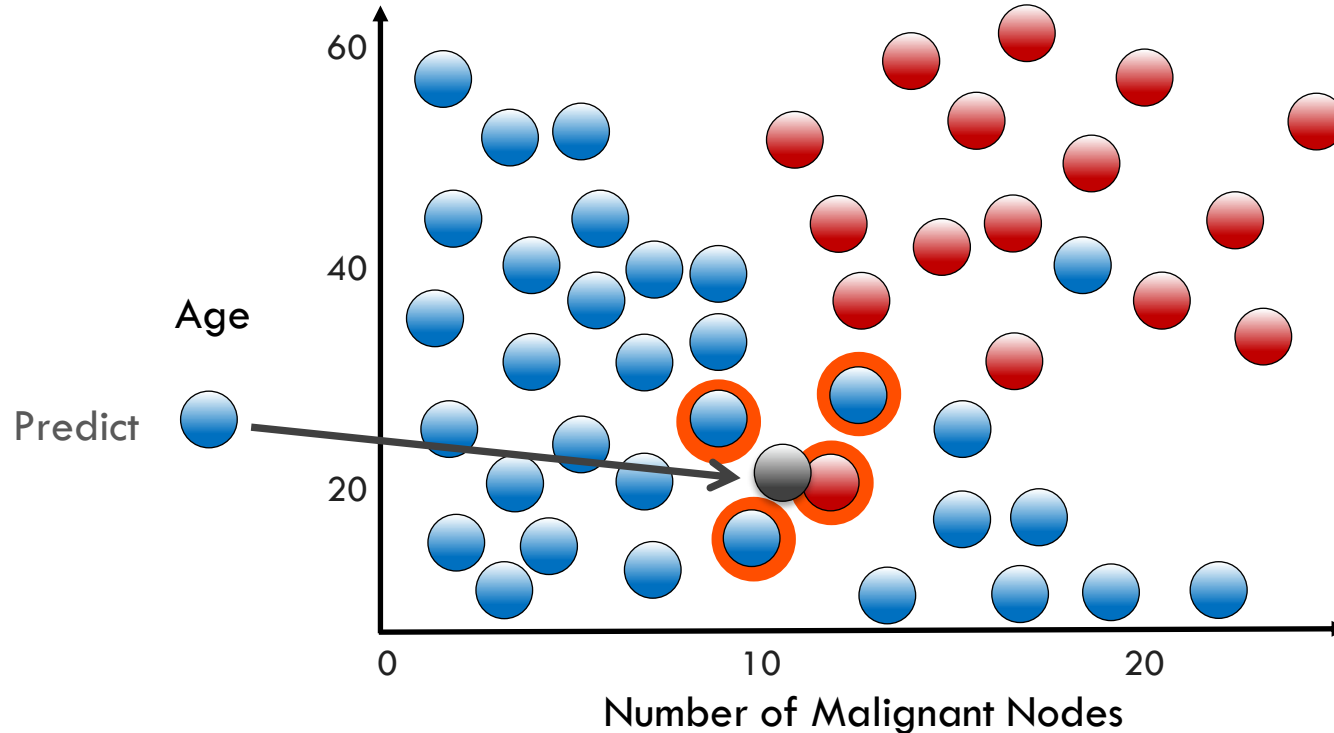
# K Nearest Neighbors Classification

Neighbor Count (K = 3):  2  1



# K Nearest Neighbors Classification

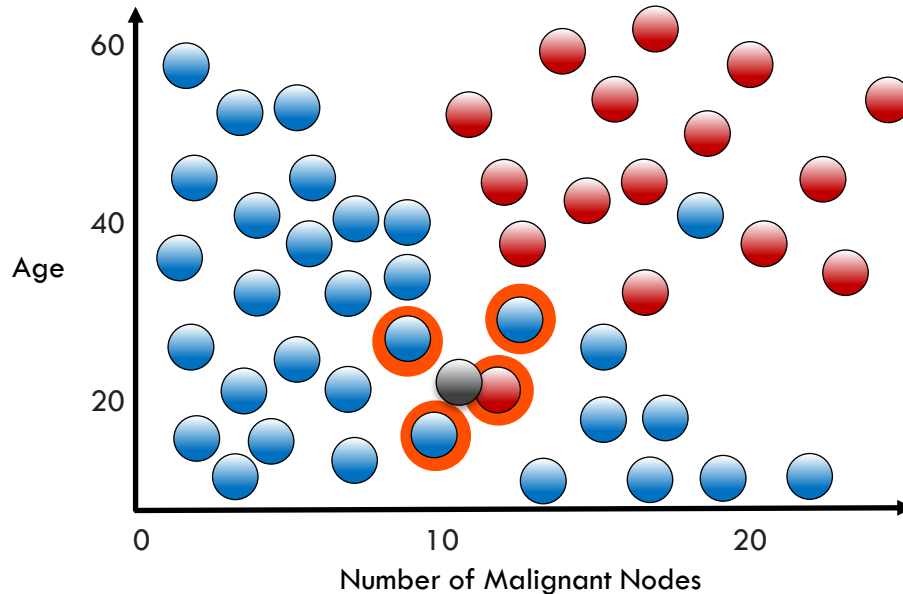
Neighbor Count (K = 4):  3  1



# What is Needed to Select a KNN Model?

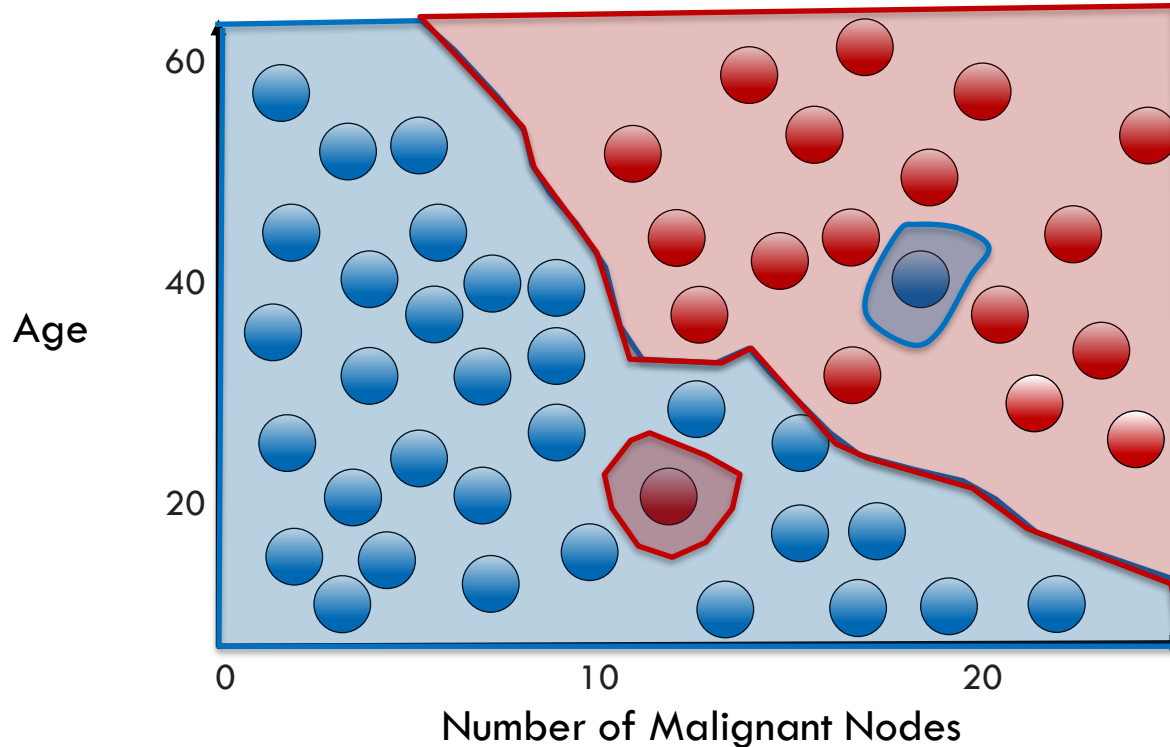
# What is Needed to Select a KNN Model?

- Correct value for 'K'
- How to measure closeness of neighbors?



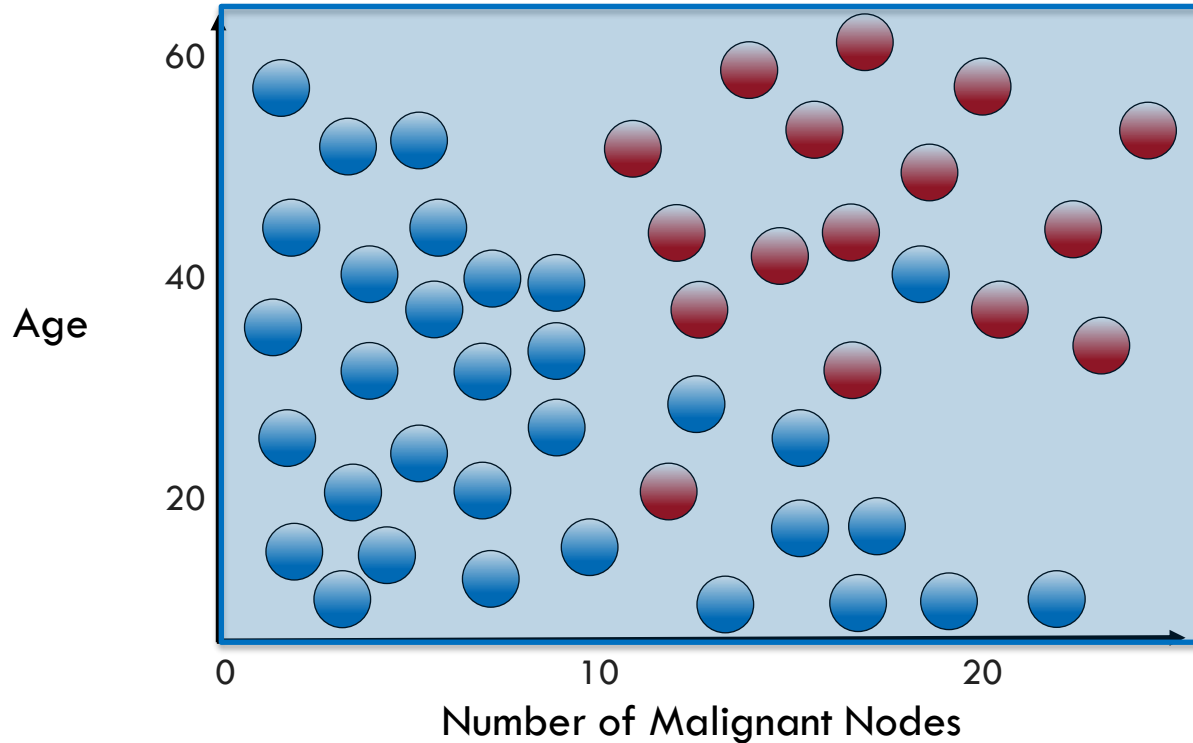
# K Nearest Neighbors Decision Boundary

$K = 1$

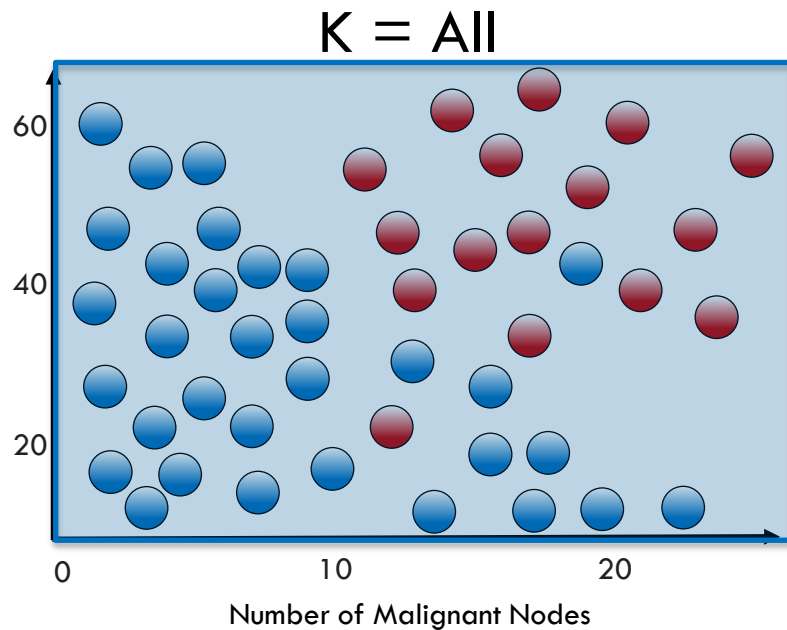
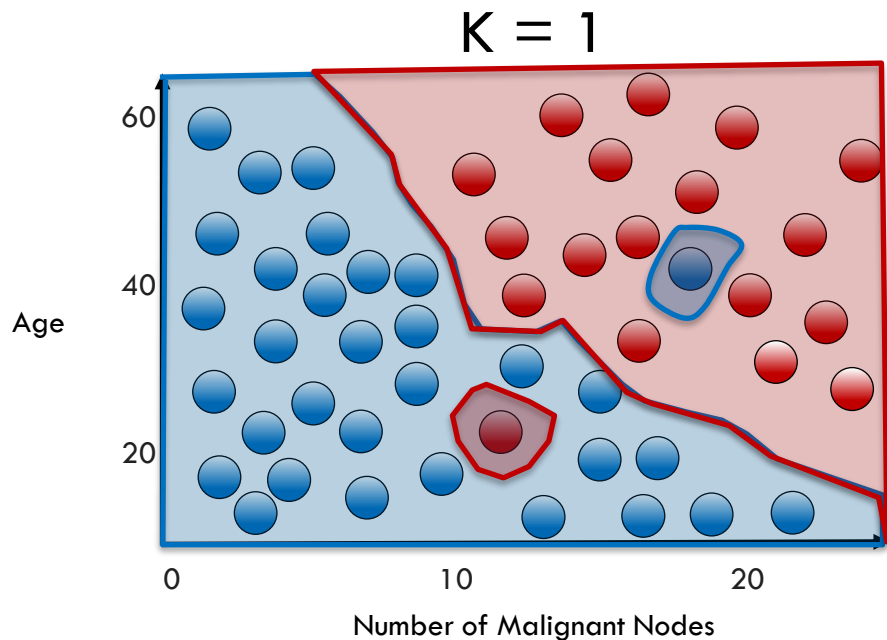


# K Nearest Neighbors Decision Boundary

$K = \text{All}$

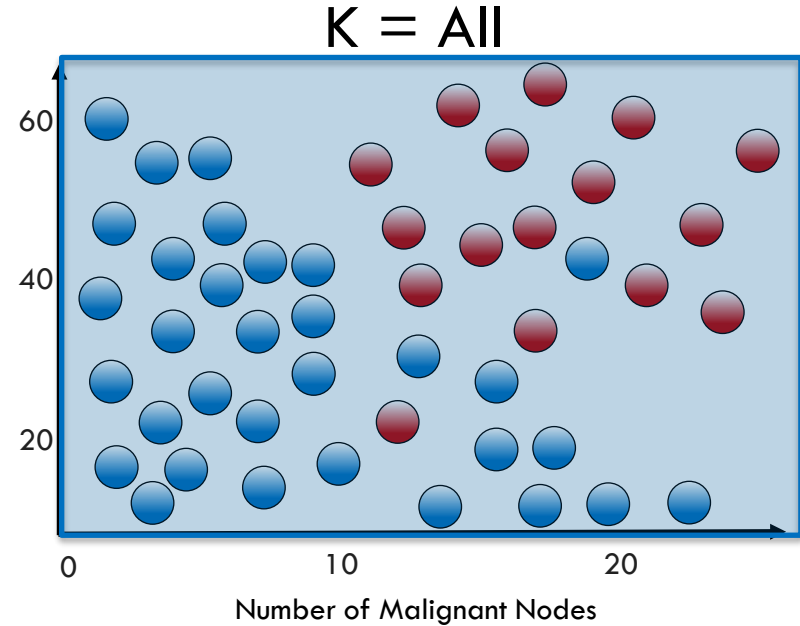
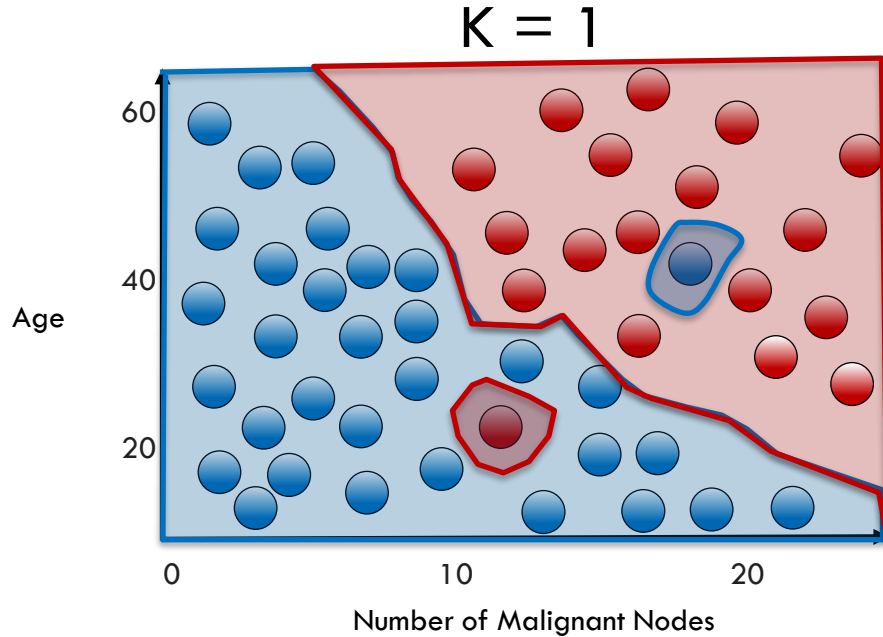


# Value of 'K' Affects Decision Boundary



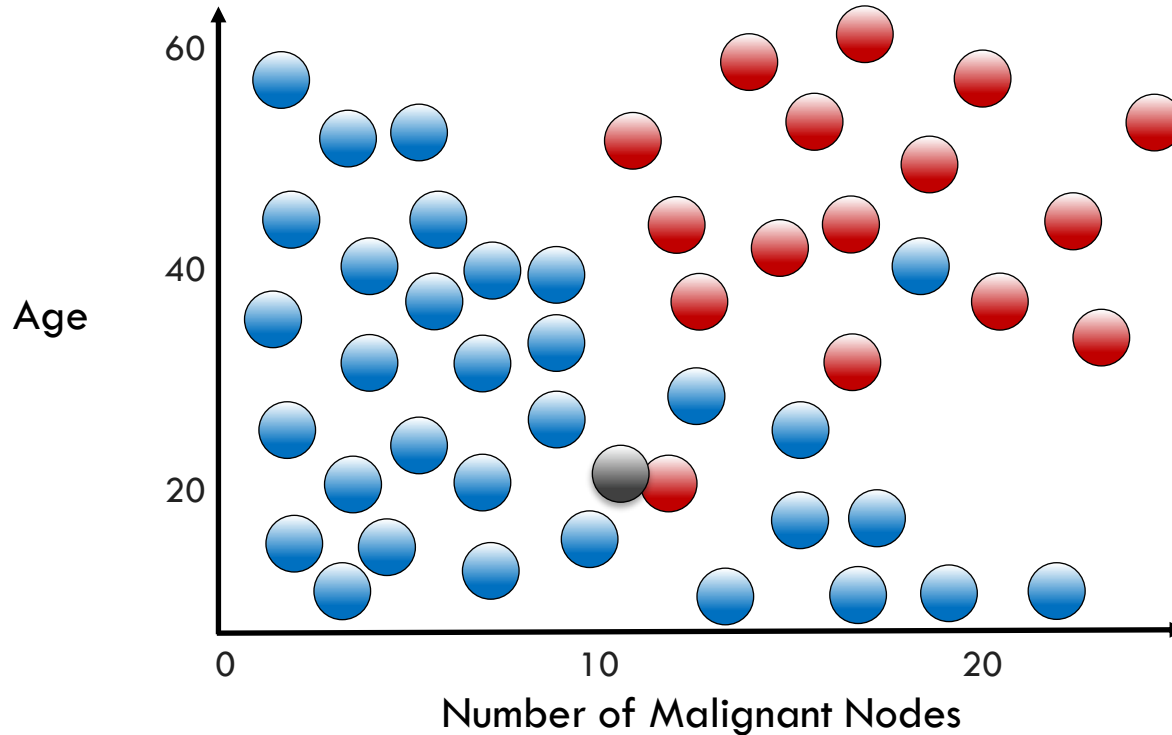


# Value of 'K' Affects Decision Boundary

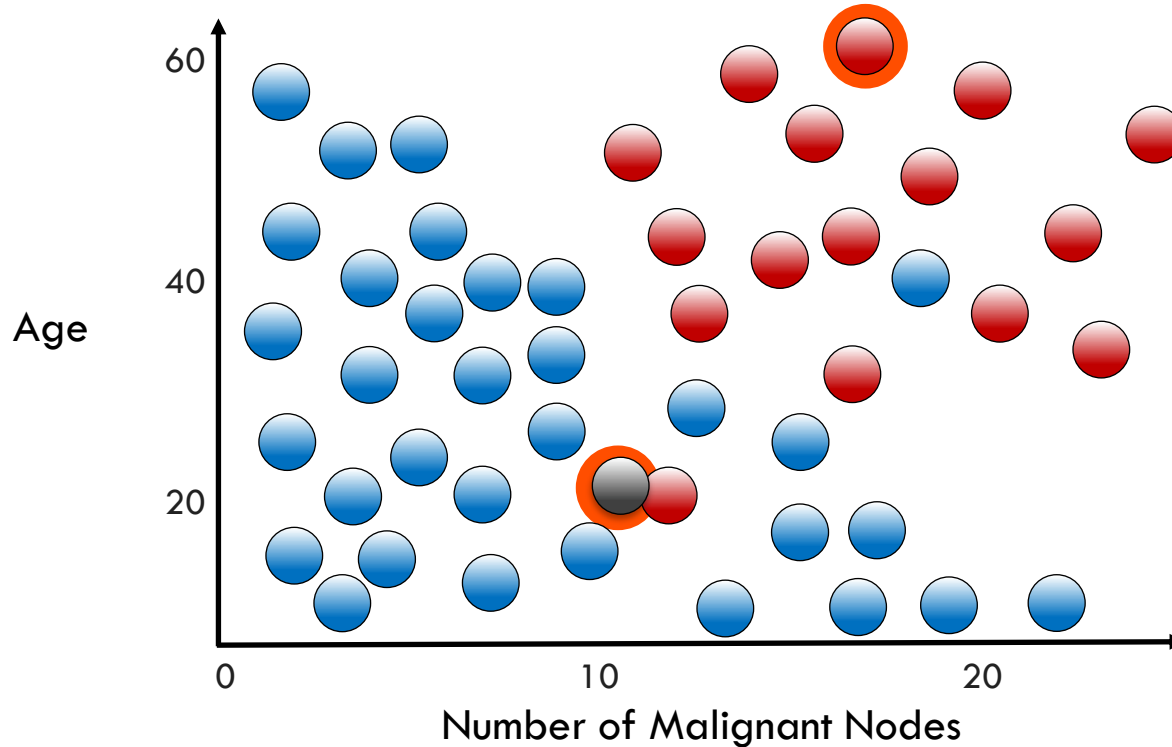


Methods for determining 'K' will be discussed in next lesson

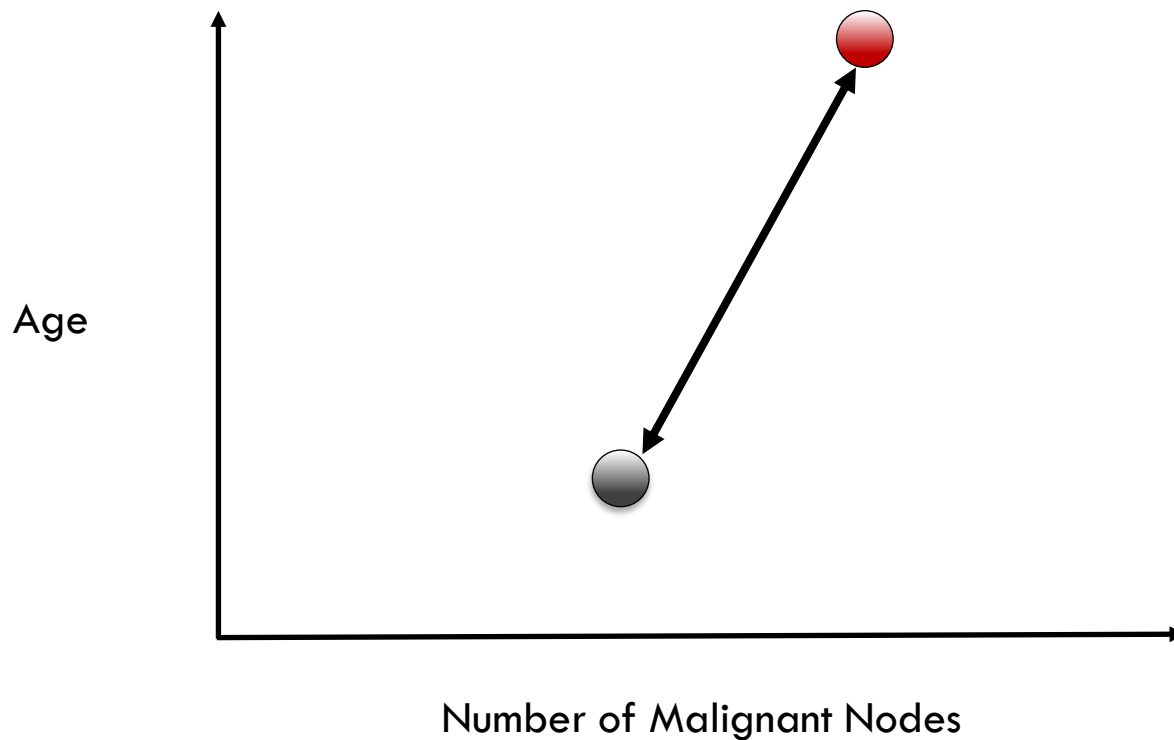
# Measurement of Distance in KNN



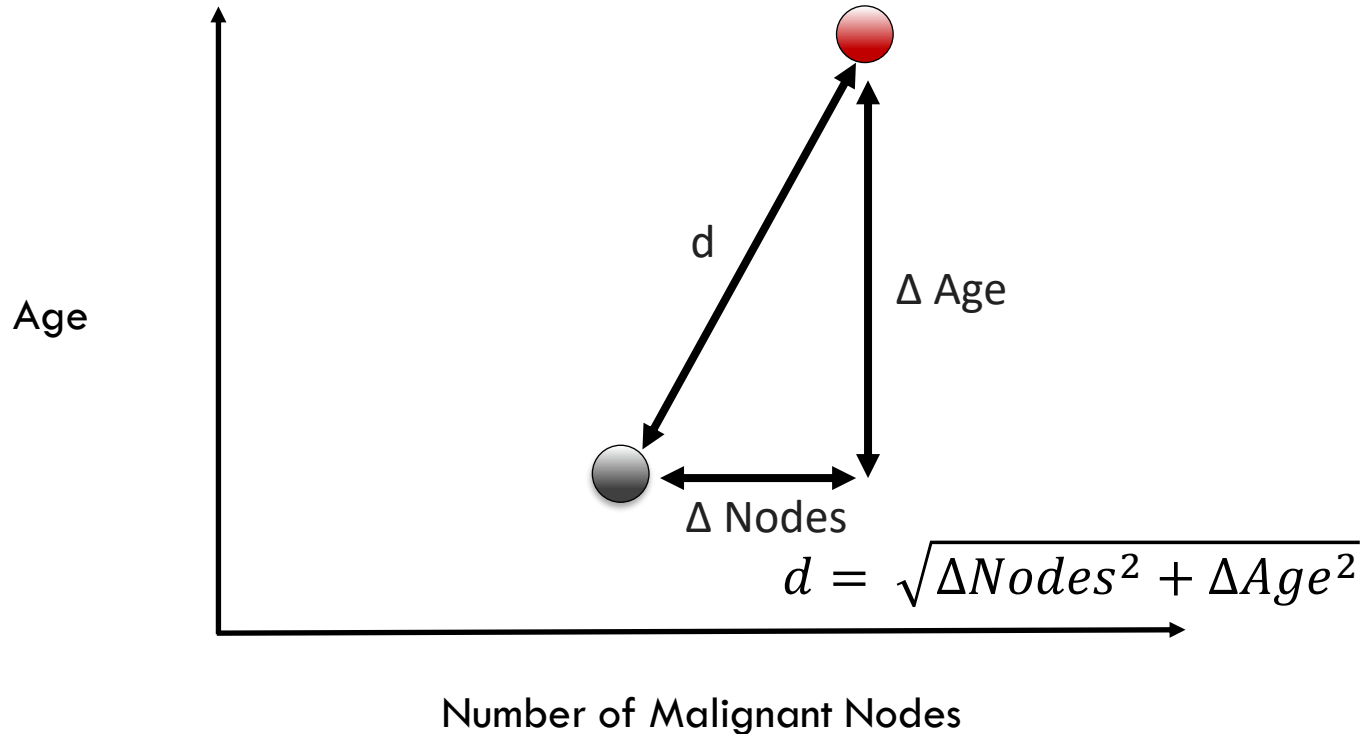
# Measurement of Distance in KNN



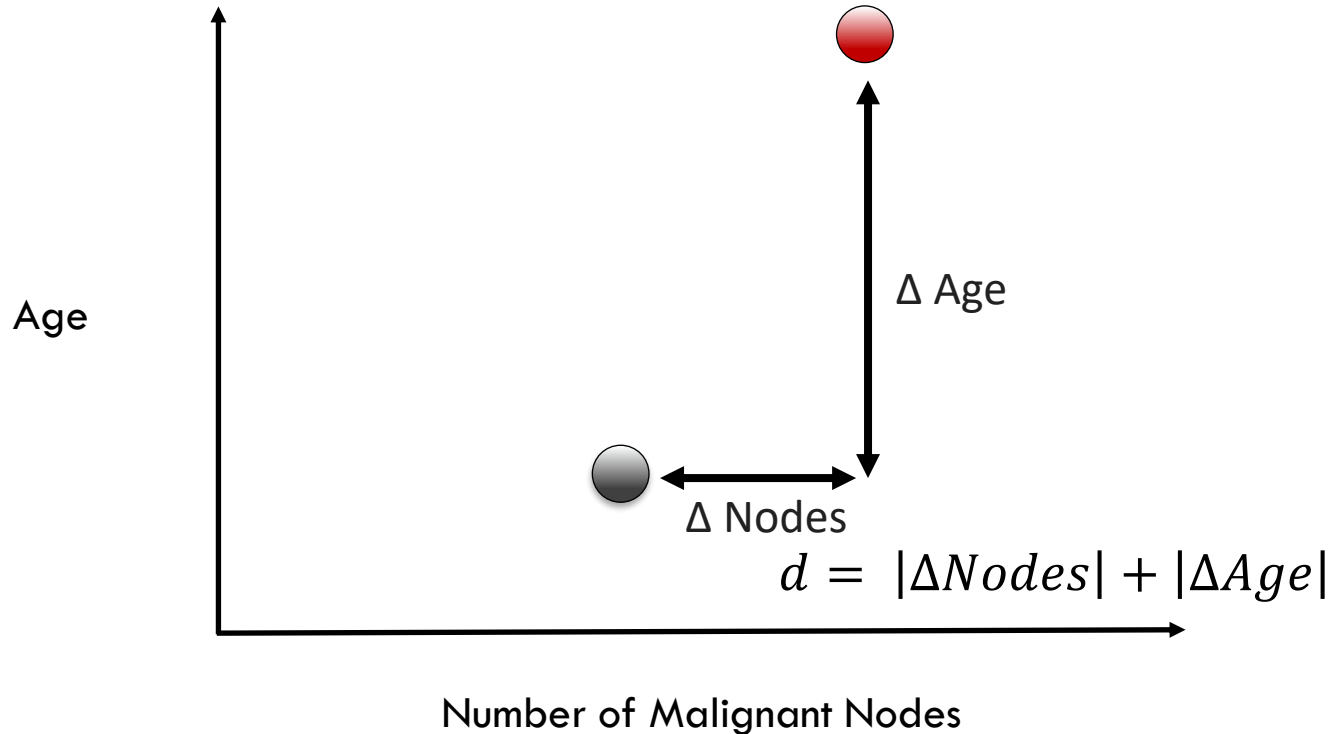
# Euclidean Distance



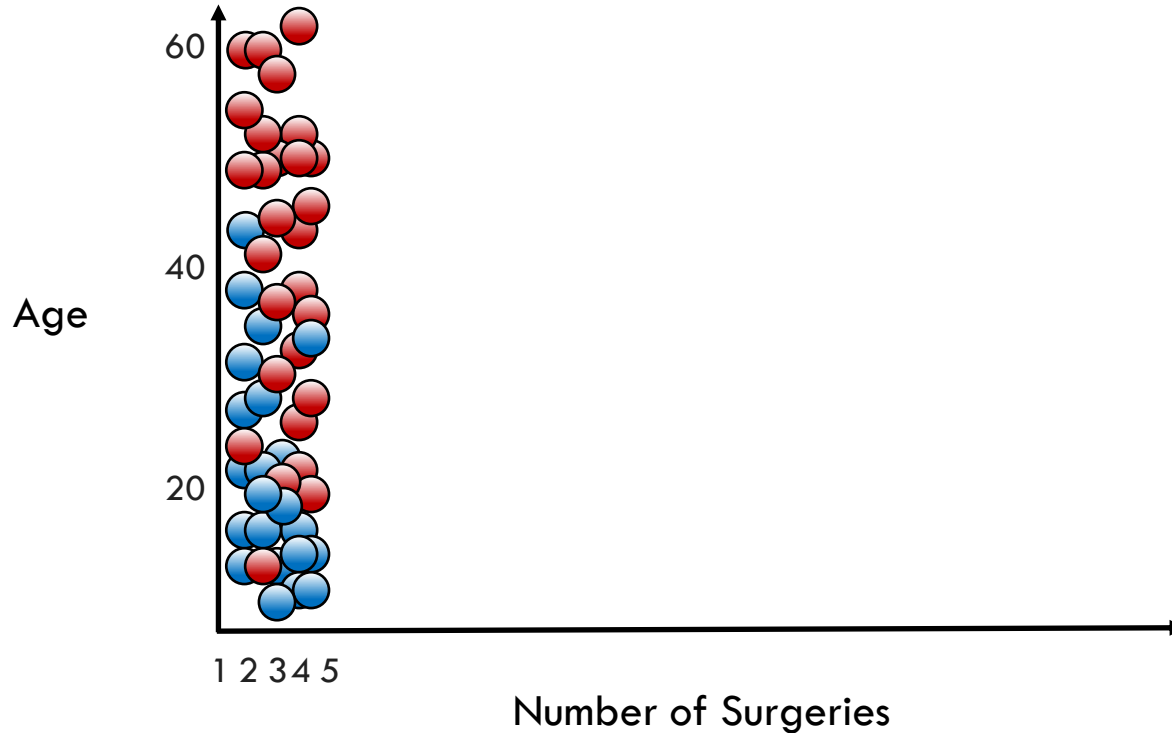
# Euclidean Distance (L2 Distance)



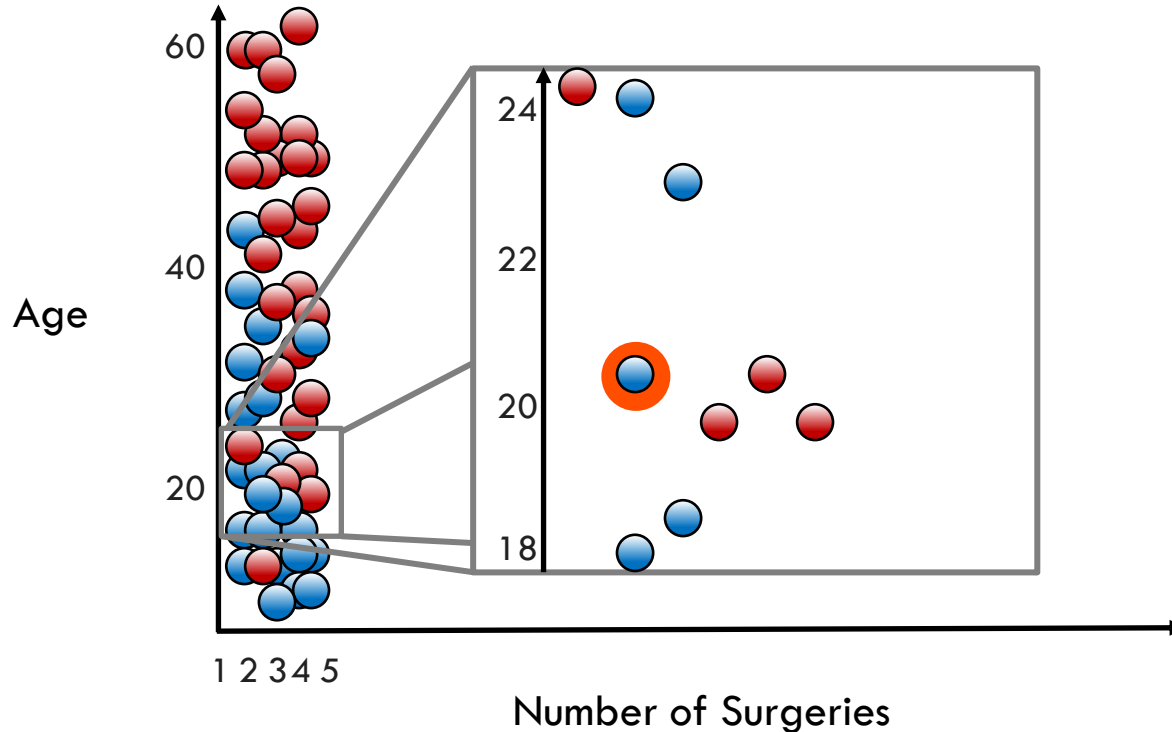
# Manhattan Distance (L1 or City Block Distance)



# Scale is Important for Distance Measurement

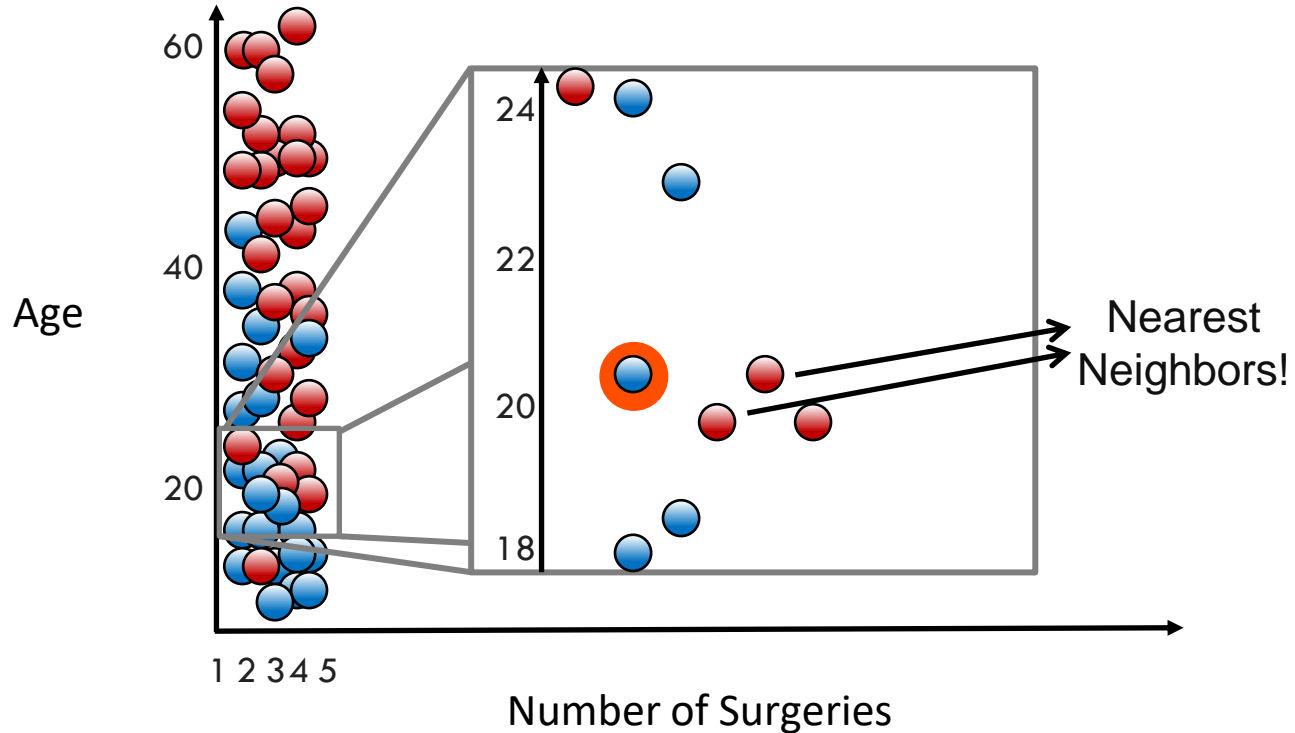


# Scale is Important for Distance Measurement



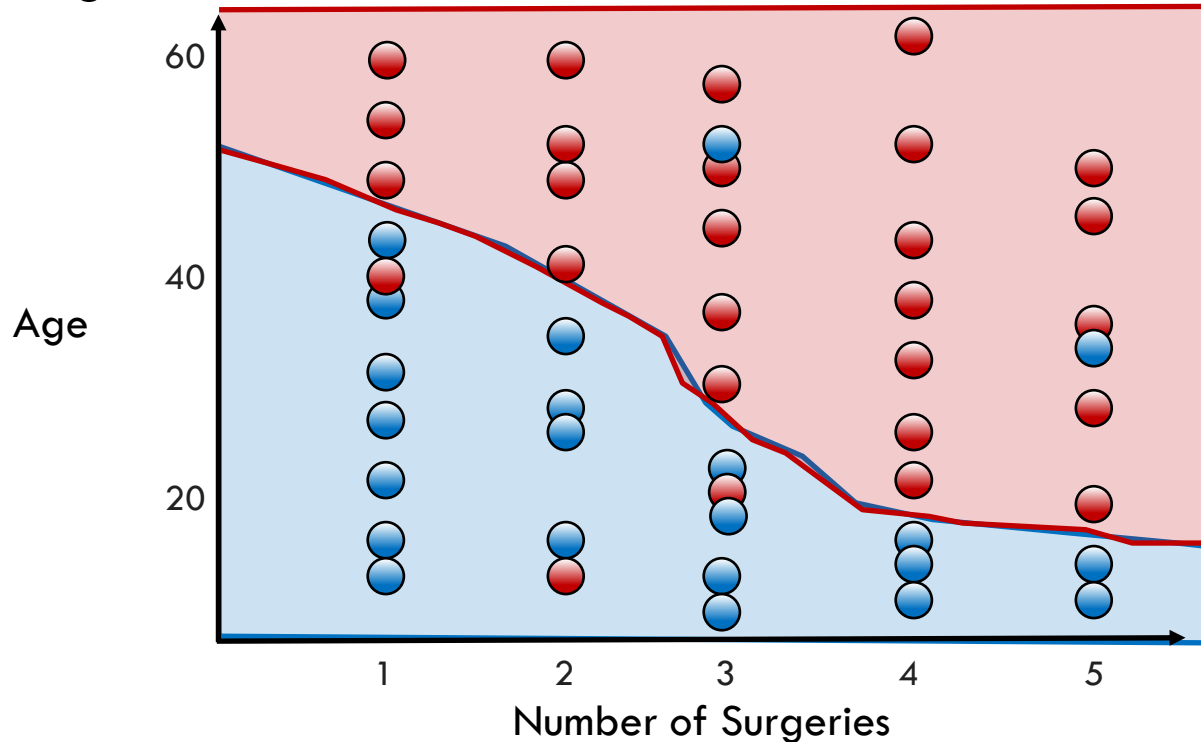


# Scale is Important for Distance Measurement



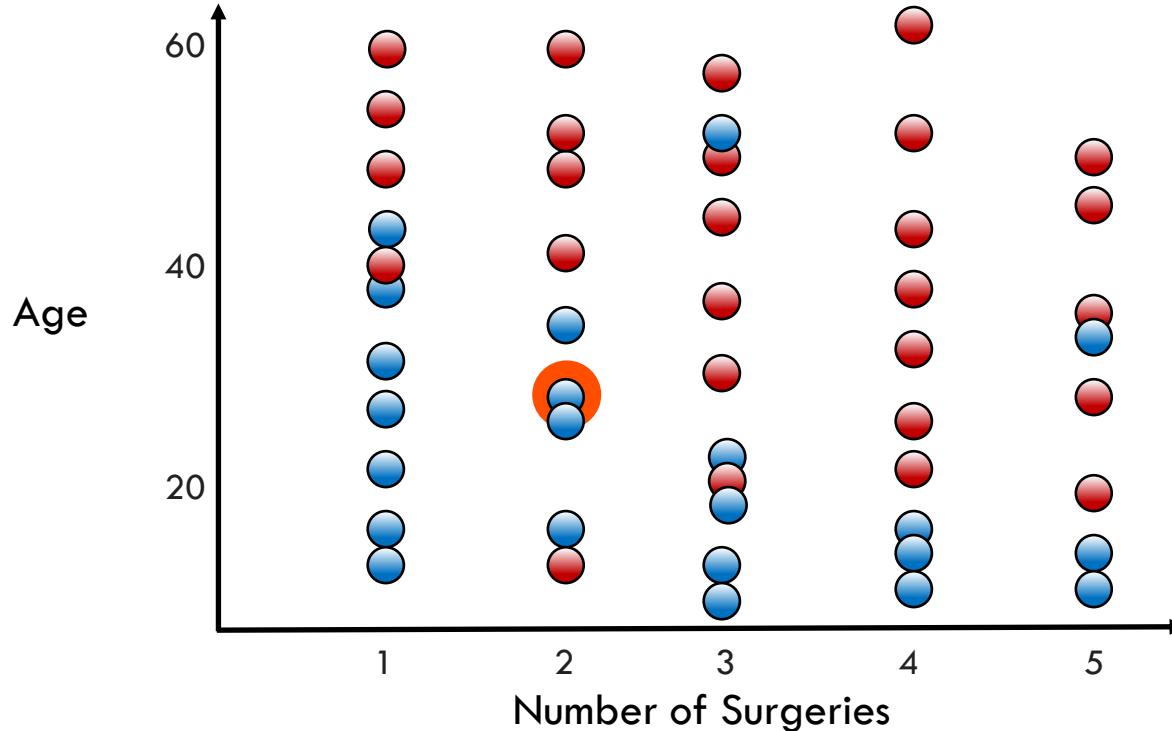
# Scale is Important for Distance Measurement

"Feature Scaling"



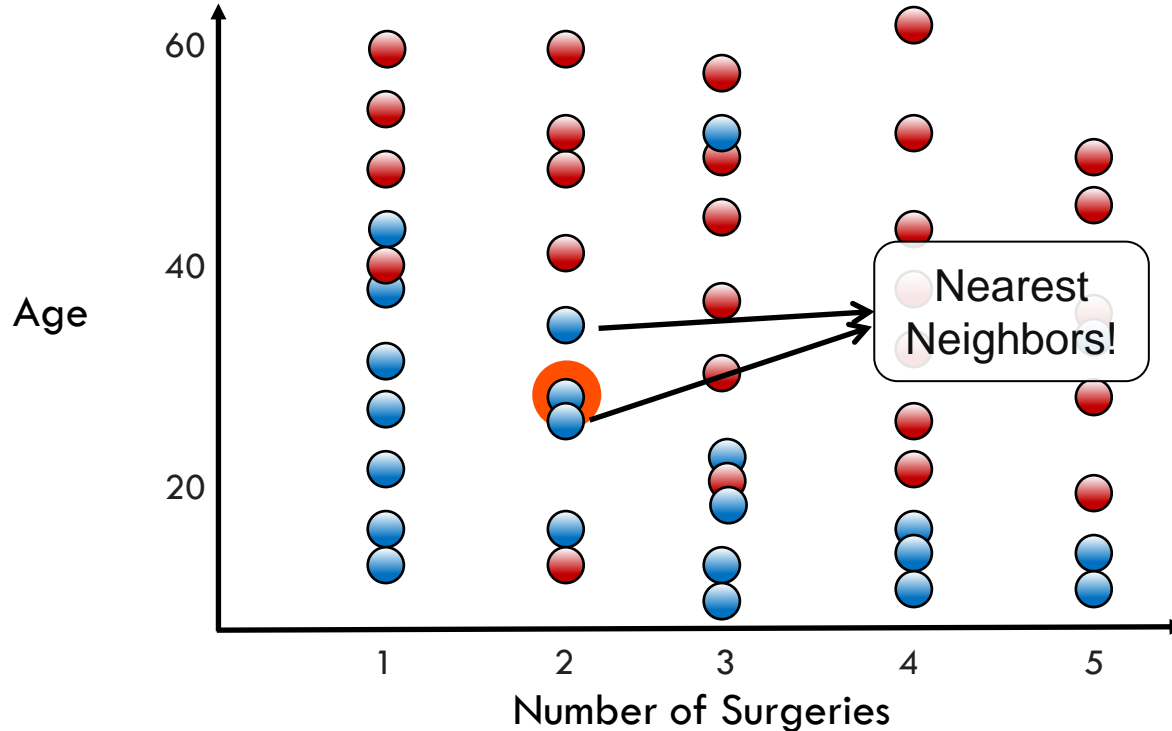
# Scale is Important for Distance Measurement

"Feature Scaling"



# Scale is Important for Distance Measurement

"Feature Scaling"



# Comparison of Feature Scaling Methods

- **Standard Scaler:** mean center data and scale to unit variance
- **Minimum-Maximum Scaler:** scale data to fixed range (usually 0–1)
- **Maximum Absolute Value Scaler:** scale maximum absolute value

# Feature Scaling: The Syntax

**Import the class containing the scaling method**

```
from sklearn.preprocessing import StandardScaler
```

# Feature Scaling: The Syntax

**Import the class containing the scaling method**

```
from sklearn.preprocessing import StandardScaler
```

**Create an instance of the class**

```
StdSc = StandardScaler()
```

# Feature Scaling: The Syntax

**Import the class containing the scaling method**

```
from sklearn.preprocessing import StandardScaler
```

**Create an instance of the class**

```
StdSc = StandardScaler()
```

**Fit the scaling parameters and then transform the data**

```
StdSc = StdSc.fit(X_data)
```

```
X_scaled = StdSc.transform(X_data)
```



# Feature Scaling: The Syntax

**Import the class containing the scaling method**

```
from sklearn.preprocessing import StandardScaler
```

**Create an instance of the class**

```
StdSc = StandardScaler()
```

**Fit the scaling parameters and then transform the data**

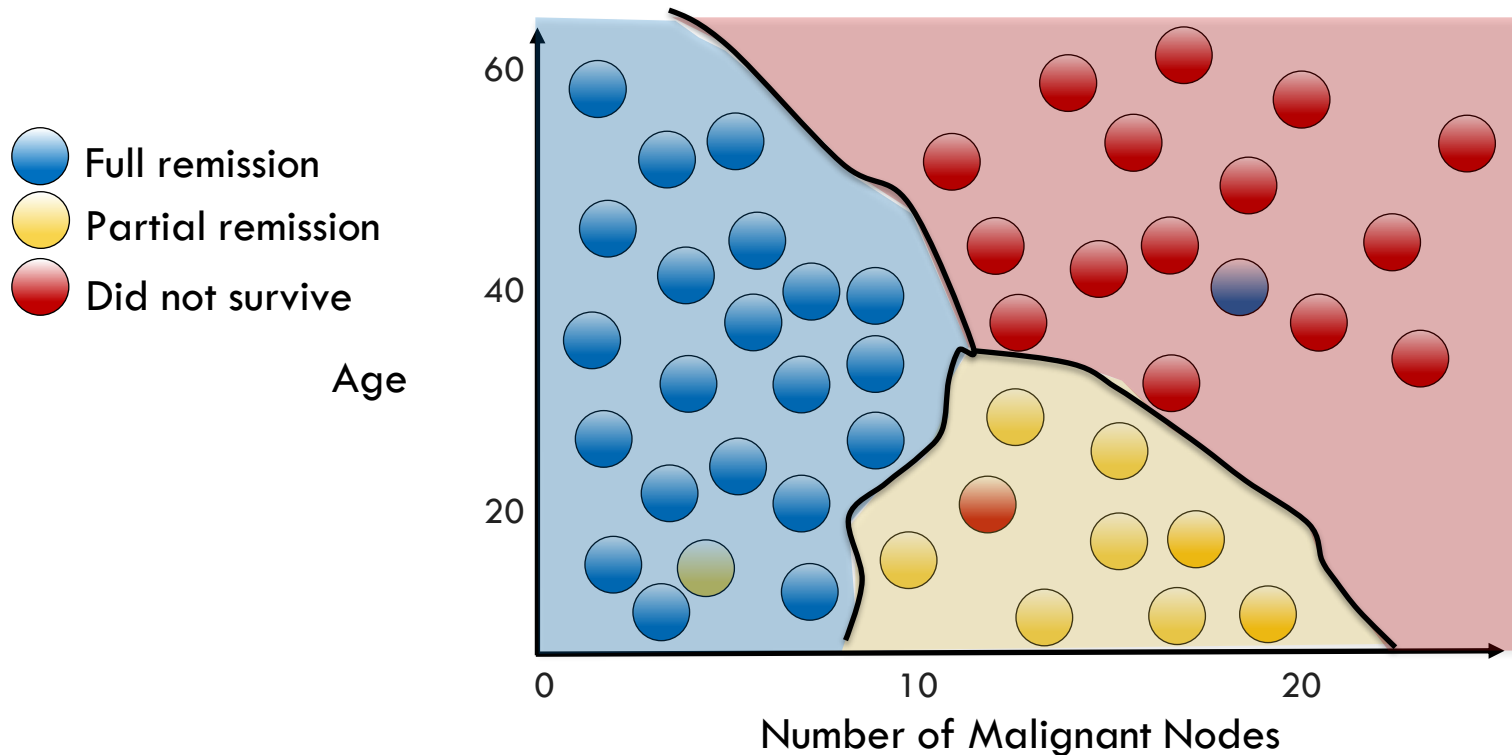
```
StdSc = StdSc.fit(X_data)
```

```
X_scaled = StdSc.transform(X_data)
```

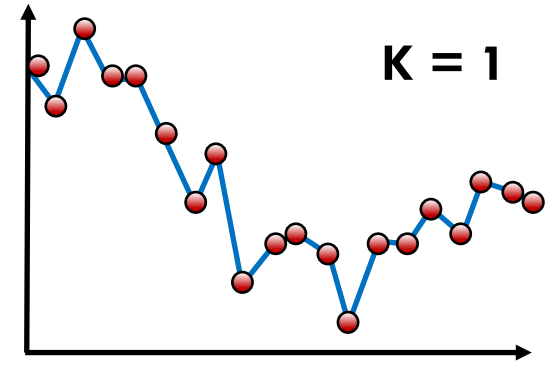
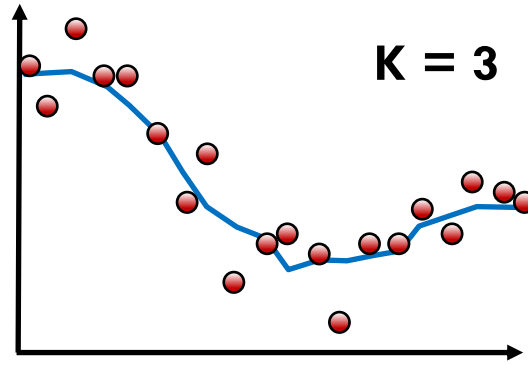
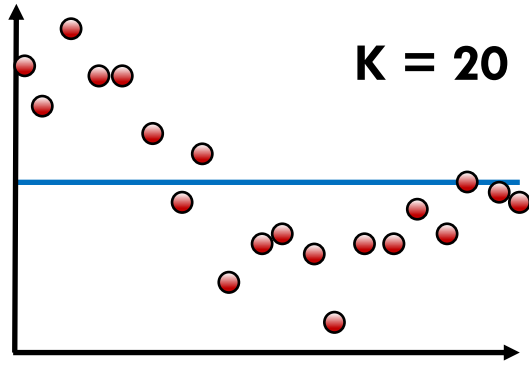
**Other scaling methods exist: [MinMaxScaler](#), [MaxAbsScaler](#).**

# Multiclass KNN Decision Boundary

$K = 5$



# Regression with KNN



# Characteristics of a KNN Model

- Fast to create model because it simply stores data
- Slow to predict because many distance calculations
- Can require lots of memory if data set is large

# K Nearest Neighbors: The Syntax

**Import the class containing the classification method**

```
from sklearn.neighbors import KNeighborsClassifier
```

# K Nearest Neighbors: The Syntax

**Import the class containing the classification method**

```
from sklearn.neighbors import KNeighborsClassifier
```

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

- Install [Intel® oneAPI AI Analytics Toolkit \(AI Kit\)](#)

**Add the following two lines of code after the above code:**

```
import patch_sklearn  
patch_sklearn()
```

# K Nearest Neighbors: The Syntax

**Import the class containing the classification method**

```
from sklearn.neighbors import KNeighborsClassifier
```

**Create an instance of the class**

```
KNN = KNeighborsClassifier(n_neighbors=3)
```

# K Nearest Neighbors: The Syntax

**Import the class containing the classification method**

```
from sklearn.neighbors import KNeighborsClassifier
```

**Create an instance of the class**

```
KNN = KNeighborsClassifier(n_neighbors=3)
```

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

```
KNN = KNN.fit(X_data, y_data)
```

```
y_predict = KNN.predict(X_data)
```



# K Nearest Neighbors: The Syntax

Import the class containing the classification method

```
from sklearn.neighbors import KNeighborsClassifier
```

Create an instance of the class

```
KNN = KNeighborsClassifier(n_neighbors=3)
```

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

```
KNN = KNN.fit(X_data, y_data)
```

```
y_predict = KNN.predict(X_data)
```

The **fit** and **predict/transform** syntax will show up throughout the course.

# K Nearest Neighbors: The Syntax

**Import the class containing the classification method**

```
from sklearn.neighbors import KNeighborsClassifier
```

**Create an instance of the class**

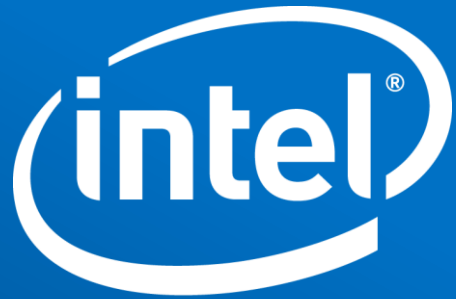
```
KNN = KNeighborsClassifier(n_neighbors=3)
```

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

```
KNN = KNN.fit(X_data, y_data)
```

```
y_predict = KNN.predict(X_data)
```

**Regression can be done with **KNeighborsRegressor**.**



Software