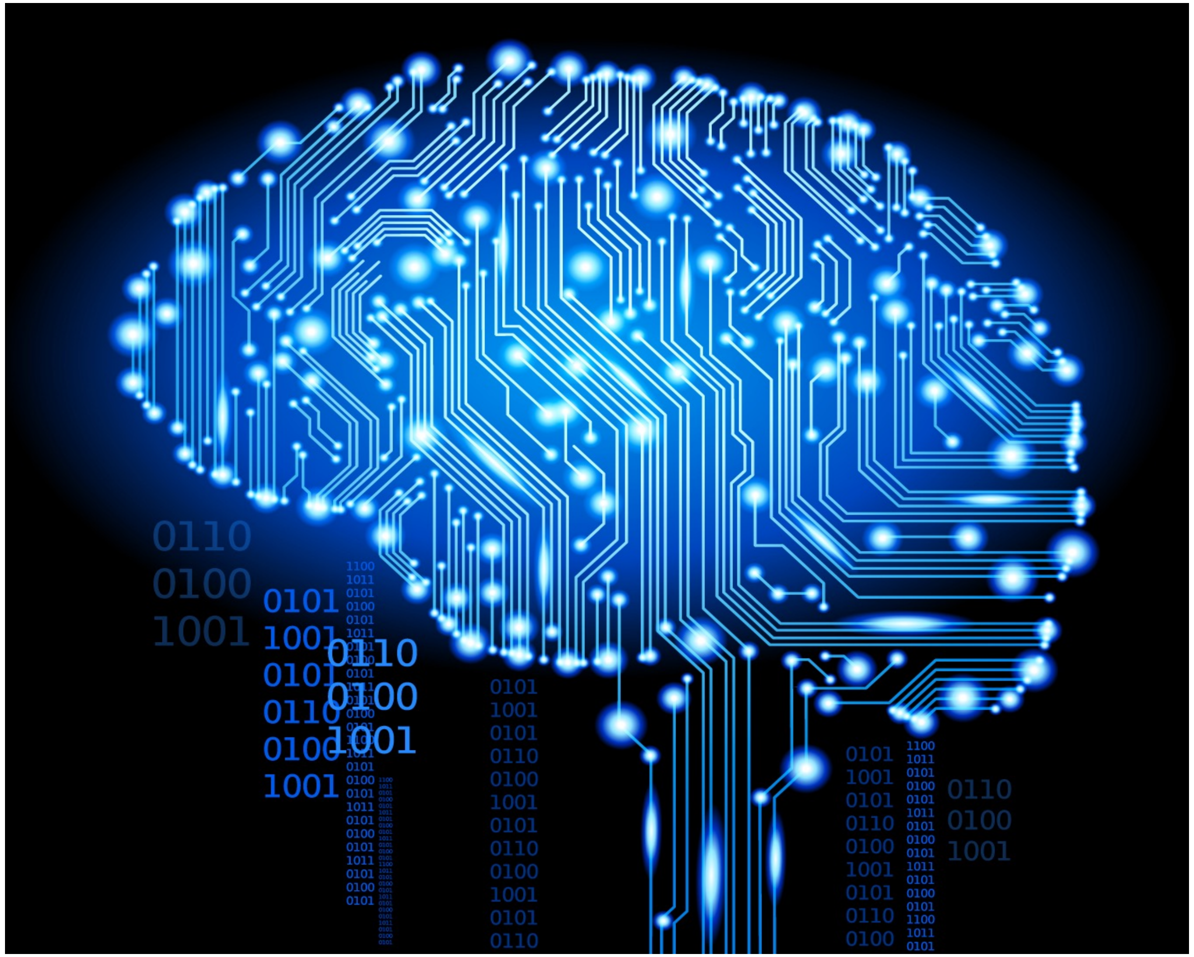
Introduction to Data Science

(Lecture 3)

Dr. Mohammad Pourhomayoun
Associate Professor
Computer Science Department
California State University, Los Angeles

Data Analytics
and Machine Learning

Review: What is Machine Learning?



Review: What is Machine Learning?

- **A Definition:** Designing and constructing algorithms or methods that give computers the ability to learn from past data, without being explicitly programmed, and then make predictions on future data.

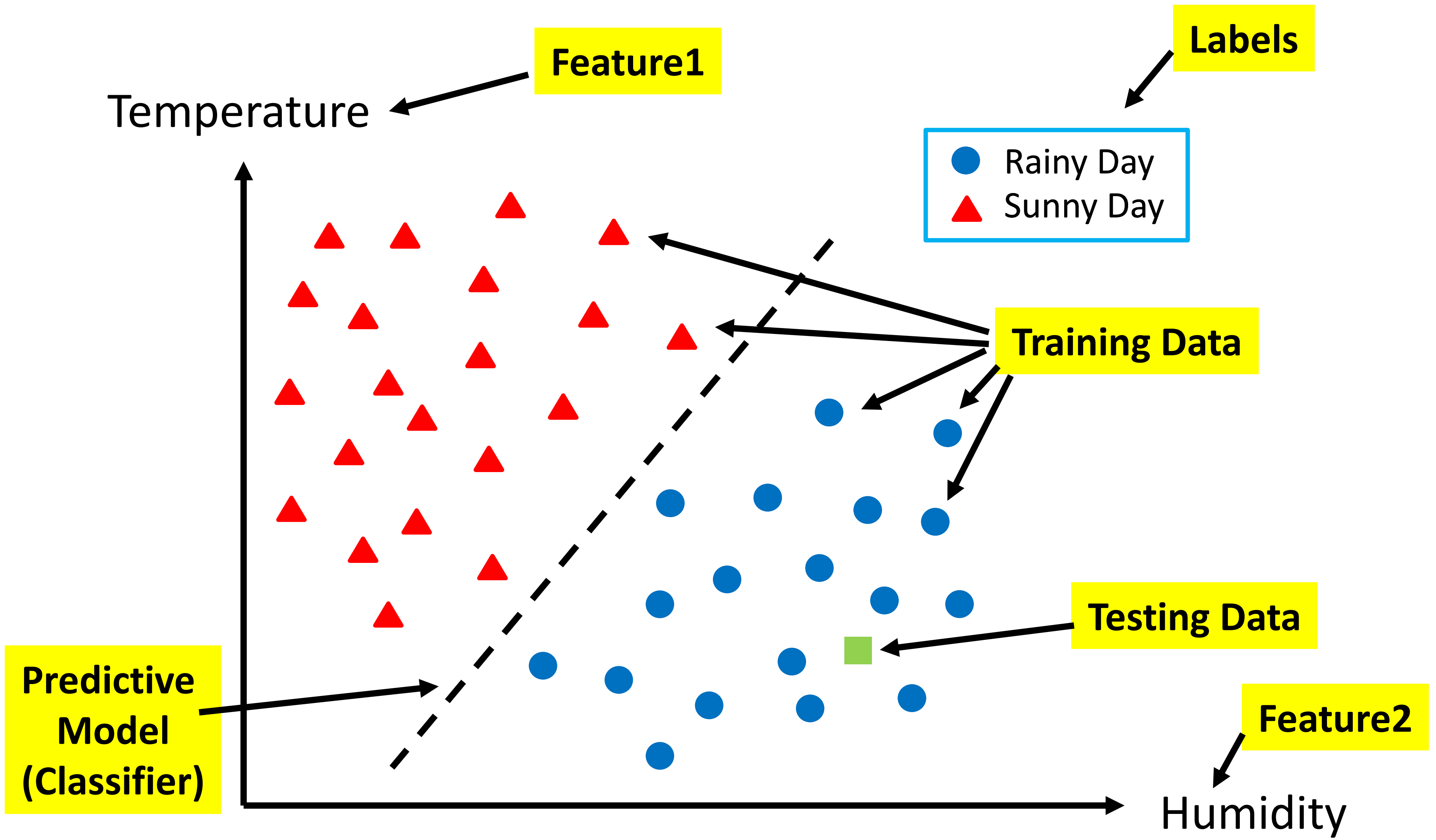
- **Another Definition:** A set of algorithms that can automatically detect and extract patterns in past data, and then use the extracted patterns to predict on future data, or to perform other kinds of decision making.

Example: Weather Forecasting

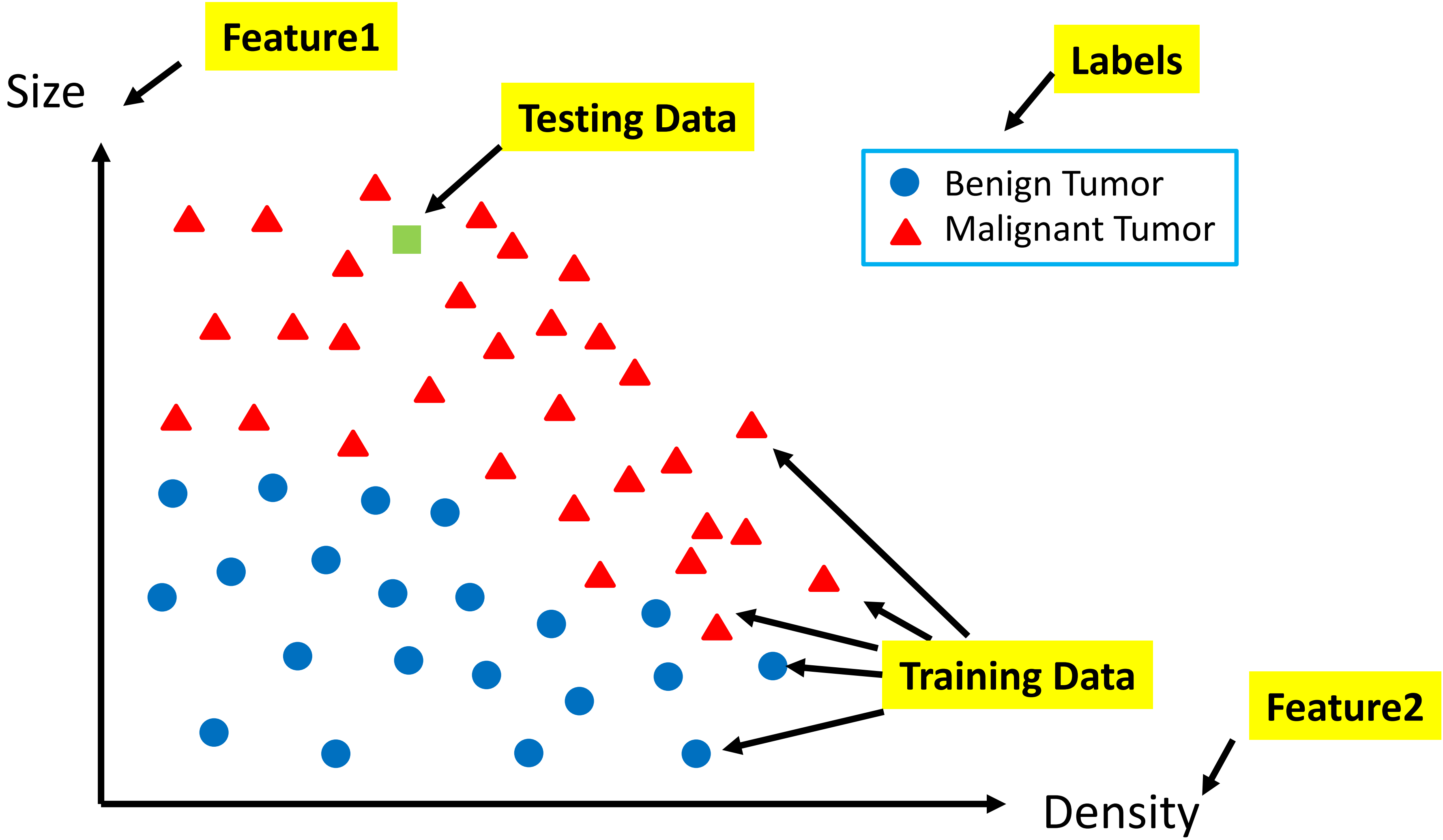
- Suppose that we have the Temperature and Humidity of the **past 30 days**.
- We also know whether those days were **Sunny** or **Rainy**.

- Questions: Now, If we know the **Temperature** and **Humidity** of **tomorrow**, can we **predict** tomorrow's outlook (predict whether tomorrow is rainy or sunny)?

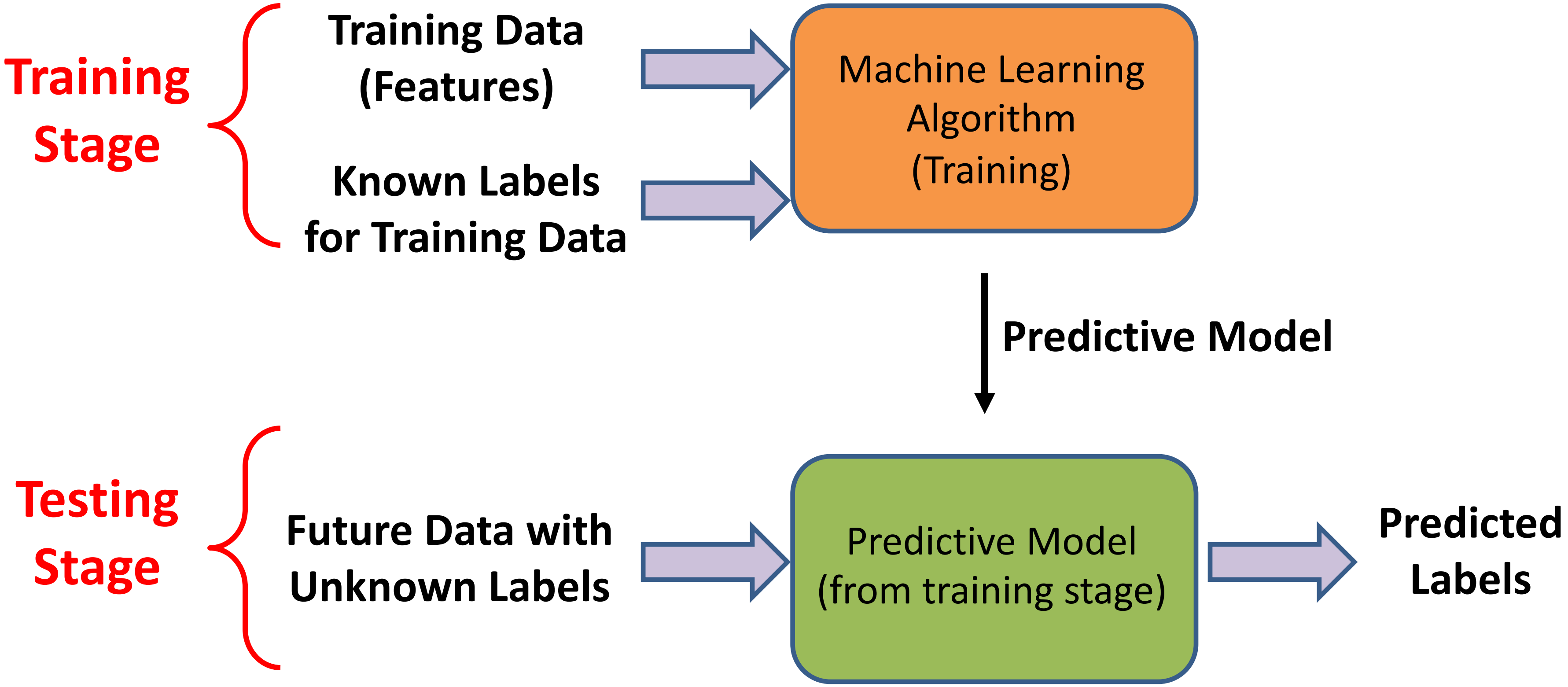
Example: Weather Forecasting



Example: Predicting Cancer



Supervised Learning: Learning from labeled Data



Two Important Approaches of Supervised Learning:

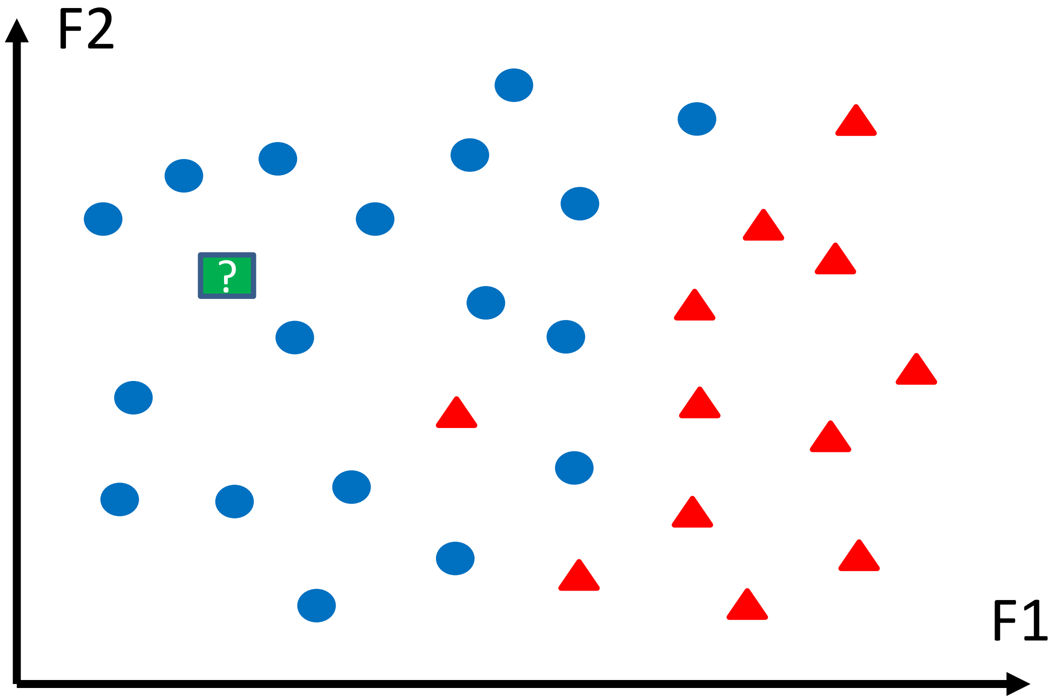
- **Classification**: Predict a **discrete** valued output for each observation.
  - Labels are **discrete** (**categorical**)
  - Labels can be binary (e.g., rainy/sunny, spam/non-spam,) or non-binary (e.g., rainy/sunny/cloudy, object recognition (100classes))

- **Regression:** Predict a **continuous** valued output for each observation.
  - Labels are **continuous** (**numeric**), e.g., stock price, housing price
  - Can define ‘closeness’ when comparing prediction with true values

K-Nearest Neighbor Classifier
(KNN Classifier)

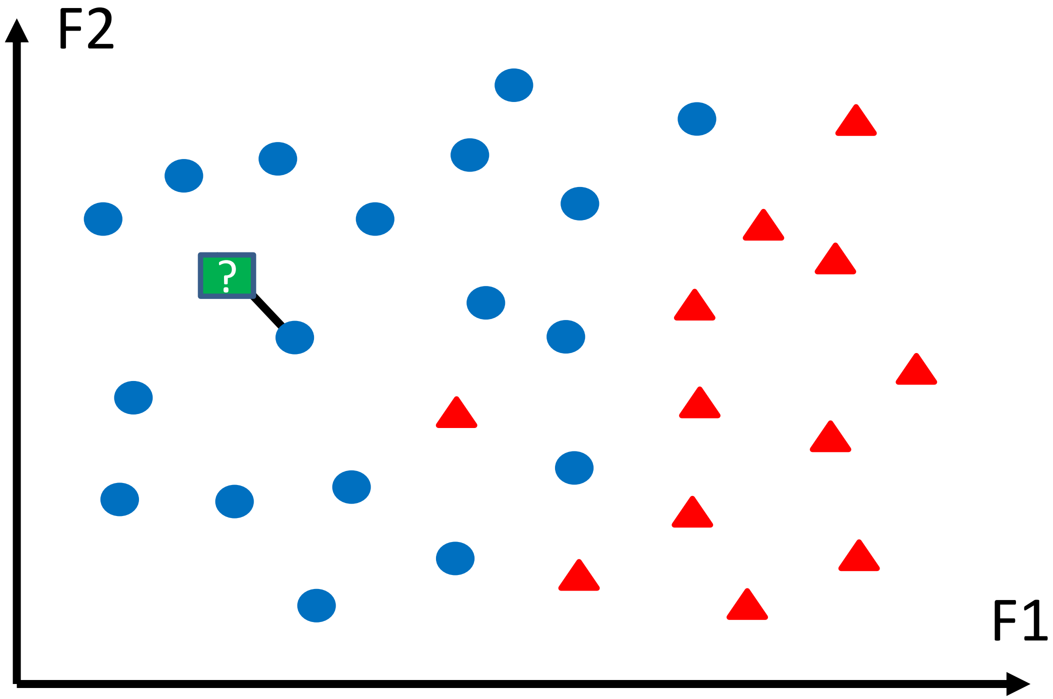
Nearest Neighbor Classification

- A simple classification algorithm that classifies objects based on the closest training samples in the feature space.



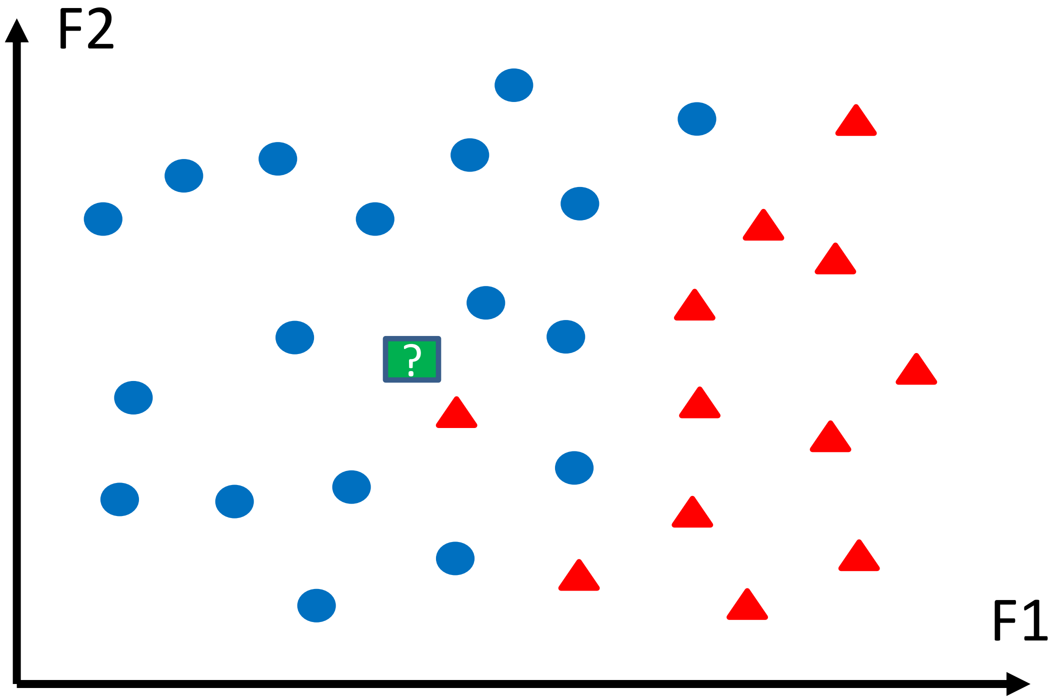
Nearest Neighbor Classification

- In the example below, the nearest neighbor (the closest training sample) is blue, so we can conclude that the unknown sample is blue.
- But, is this always true?



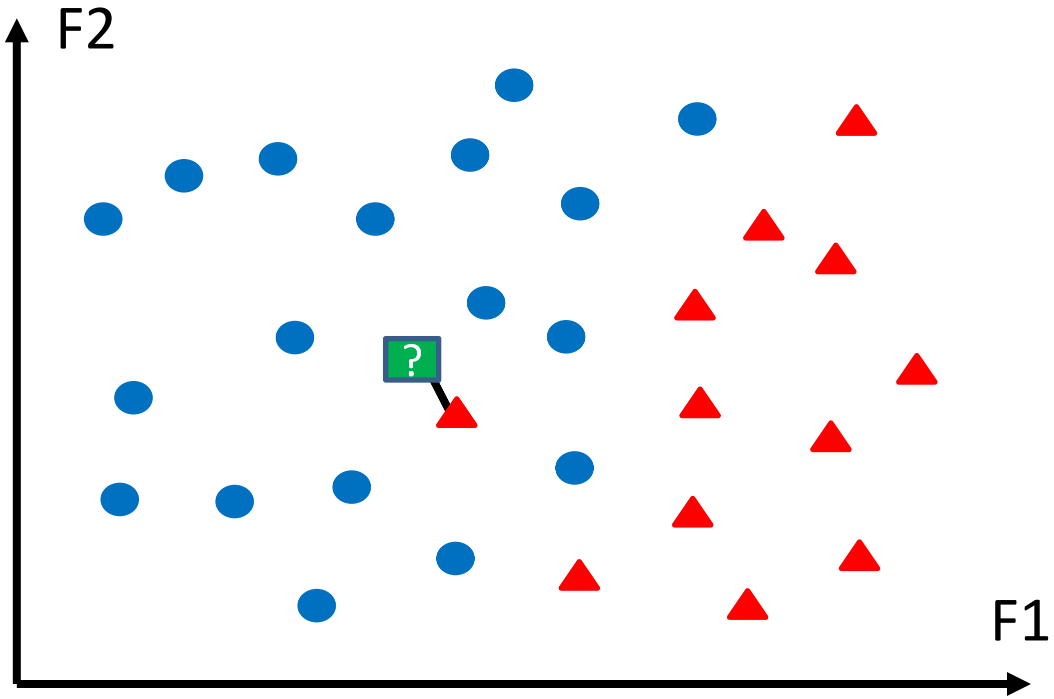
Nearest Neighbor Classification

- Making decision only based on **one sample** is risky, because it is **not reliable** and it is very **vulnerable** to noise or mistakes in training set.



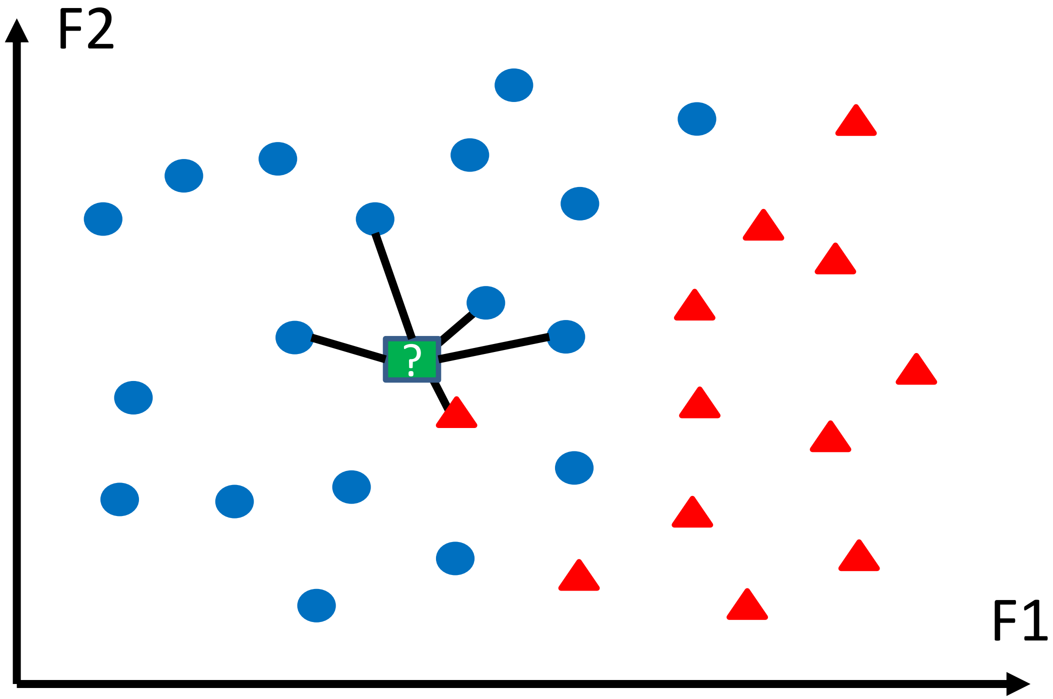
Nearest Neighbor Classification

- Making decision only based on **one sample** is risky, because it is **not reliable** and it is very **vulnerable** to noise or mistakes in training set.



KNN Classification

- K-Nearest Neighbor (KNN) classifier classifies objects based on **<u>majority of K closest training samples</u>** in the feature space, e.g. K=5.



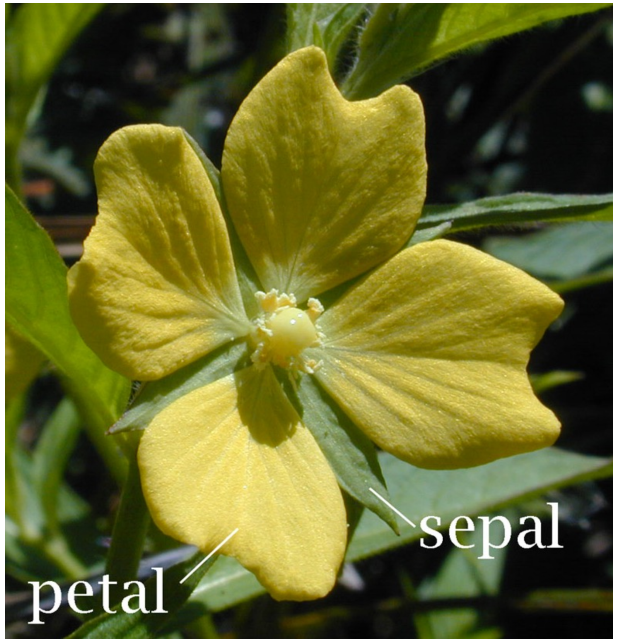
Example: Flower Classification

- **Recognizing flowers**
  - **Labels (3 classes)**: setosa, versicolor, virginica



Example: Flower Classification

- **Recognizing flowers**
  - **Labels (classes)**: setosa, versicolor, virginica
  - **Features**: sepal length, sepal width, petal length, petal width



Feature Table

- **Training Feature Table:**
  - In this example we have *6 training data samples (6 different flowers)*, *4 features* and label has *3 classes (3 different types of Iris)*.

	sepal length	sepal width	petal length	petal width	Label
1	5.3	3.7	1.5	0.2	setosa
2	5	3	2	0.2	setosa
3	7.0	3.2	4.7	1.4	versicolor
4	6.4	3.2	4.5	1.5	versicolor
5	6.3	2.7	4.9	1.8	virginica
6	7.9	3.8	6.4	2	virginica

Definition

- Assuming that we have N data samples (observations), D features, and C classes:
  - ***Feature Vector*** x (input): The vector of all features for **one single** observation: $\boldsymbol{x} \in \mathbb{R}^D$
  - ***Label*** y (output): Value of label for one observation $y \in \{0,1,2,\dots,C\}$
  - ***Training dataset***: $\{(\boldsymbol{x}_1,y_1), (\boldsymbol{x}_2,y_2),\dots, (\boldsymbol{x}_N,y_N)\}$
  - ***Special case***: binary classication
    - Number of classes: $C = 2$
    - Labels: $(0,1)$

	sepal length	sepal width	petal length	petal width	Label
1	5.3	3.7	1.5	0.2	setosa
2	5	3	2	0.2	setosa
3	7.0	3.2	4.7	1.4	versicolor
4	6.4	3.2	4.5	1.5	versicolor
5	6.3	2.7	4.9	1.8	virginica
6	7.9	3.8	6.4	2	virginica

Feature Table

- *Training dataset*: $\{(\mathbf{x}_1,y_1), (\mathbf{x}_2,y_2),\ldots, (\mathbf{x}_N,y_N)\}$: N data samples used for training.

	sepal length	sepal width	petal length	petal width	Label	
$\mathbf{x}_1$	5.3	3.7	1.5	0.2	setosa	y_1
$\mathbf{x}_2$	5	3	2	0.2	setosa	y_2
$\mathbf{x}_3$	7.0	3.2	4.7	1.4	versicolor	y_3
	6.4	3.2	4.5	1.5	versicolor	
	6.3	2.7	4.9	1.8	virginica	
	7.9	3.8	6.4	2	virginica	

1-Nearest Neighbor Classification

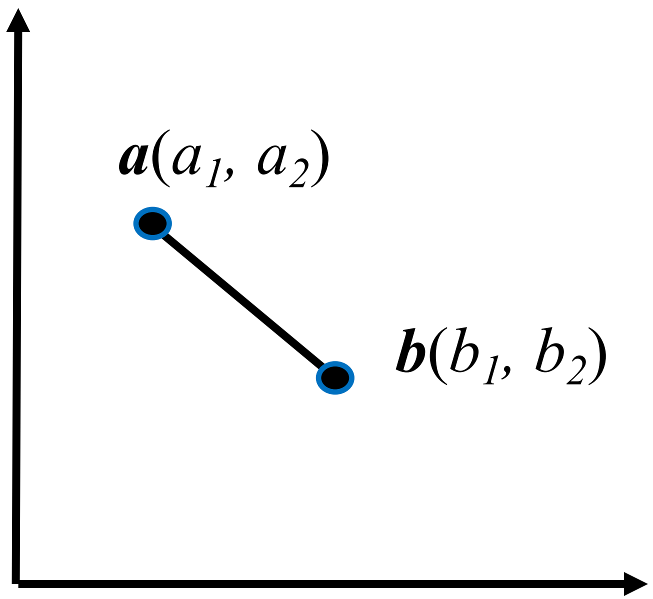
- *Training dataset*: $\{(\boldsymbol{x_1},y_1), (\boldsymbol{x_2},y_2),\ldots, (\boldsymbol{x_N},y_N)\}$ with known label.
- Now, we have a new sample with unknown label: $(\boldsymbol{x}\ , y=?)$

	sepal length	sepal width	petal length	petal width	Label	
$\boldsymbol{x_1}$	5.3	3.7	1.5	0.2	setosa	y_1
$\boldsymbol{x_2}$	5	3	2	0.2	setosa	y_2
	7.0	3.2	4.7	1.4	versicolor	
:	6.4	3.2	4.5	1.5	versicolor	:
	6.3	2.7	4.9	1.8	virginica	
$\boldsymbol{x_N}$	7.9	3.8	6.4	2	virginica	y_N
$\boldsymbol{x}$	7	3.9	5.9	1.3	???	$y=?$

Euclidean Distance

- ***Euclidean Distance*** between 2 points in 2-dimensional Space:

$$\boldsymbol{Dis(a\,,\,b)} = \|\boldsymbol{a} - \boldsymbol{b}\| \quad = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2}$$



- ***Euclidean Distance*** for D-dimensional Space:

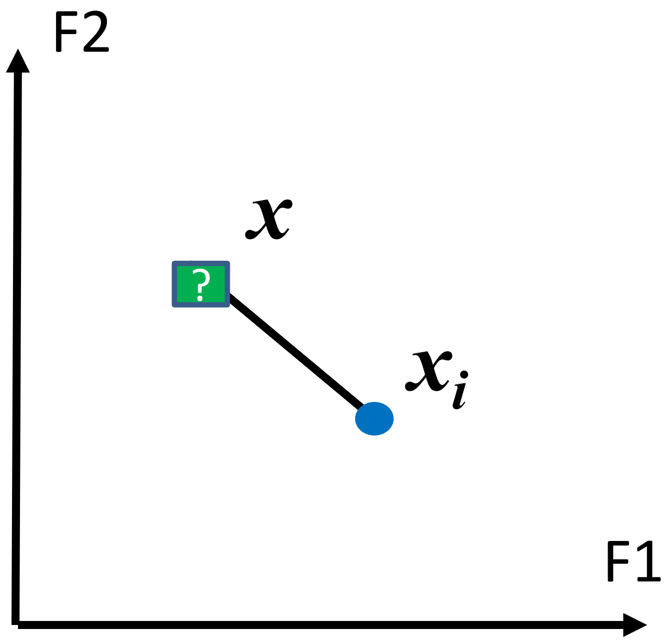
$$\boldsymbol{Dis(x_i\,,\,x)} = \|\boldsymbol{x_i} - \boldsymbol{x}\| \quad = \sqrt{\sum_{d=1}^{D}(x_{id} - x_d)^2}$$

1-Nearest Neighbor Classification

- *Training dataset*: $\{(\boldsymbol{x_1}, y_1), (\boldsymbol{x_2}, y_2), \ldots, (\boldsymbol{x_N}, y_N)\}$
- Suppose we have a new sample with unknown label: $(\boldsymbol{x}, y=?)$
- "1-NN algorithm" classifies new samples based on the closest training samples in the feature space.

- *The **Distance*** between $\boldsymbol{x}$ and a known training point $\boldsymbol{x_i}$ (Distance between 2 D-dimensional points in D-dimensional space):

$$Dis(\boldsymbol{x_i}, \boldsymbol{x}) = \|\boldsymbol{x_i} - \boldsymbol{x}\| = \sqrt{\sum_{d=1}^{D} (x_{id} - x_d)^2}$$



1-Nearest Neighbor Classification

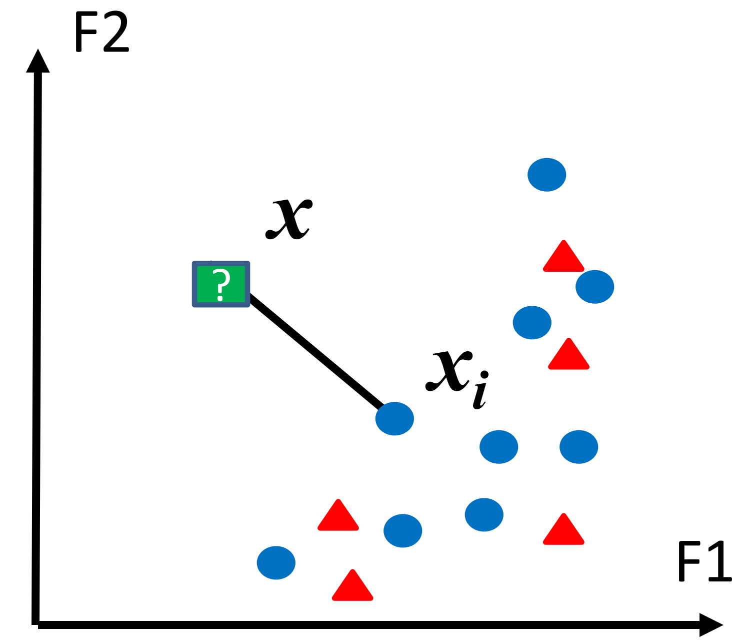
- *Training dataset*: $\{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_N, y_N)\}$
- Suppose we have a new sample with unknown label: $(\boldsymbol{x}, y=?)$

- ***Euclidean Distance*** between $\boldsymbol{x}$ and a known training point $\boldsymbol{x}_i$:

$$\boldsymbol{Dis}(\boldsymbol{x}_i, \boldsymbol{x}) = \|\boldsymbol{x}_i - \boldsymbol{x}\|_2 = \sqrt{\sum_{d=1}^{D} (x_{id} - x_d)^2}$$

- Nearest Neighbor to $\boldsymbol{x}$ (*we can call it* $\boldsymbol{NN}(\boldsymbol{x})$):

$$\boldsymbol{NN}(\boldsymbol{x}) = \arg\min_{i \in \{1,\ldots,N\}} \left(\boldsymbol{Dis}(\boldsymbol{x}_i, \boldsymbol{x})\right)$$
$$= \arg\min_{i \in \{1,\ldots,N\}} \left(\|\boldsymbol{x}_i - \boldsymbol{x}\|^2\right)$$
$$= \arg\min_{i \in \{1,\ldots,N\}} \left(\sum_{d=1}^{D} (x_{id} - x_d)^2\right)$$



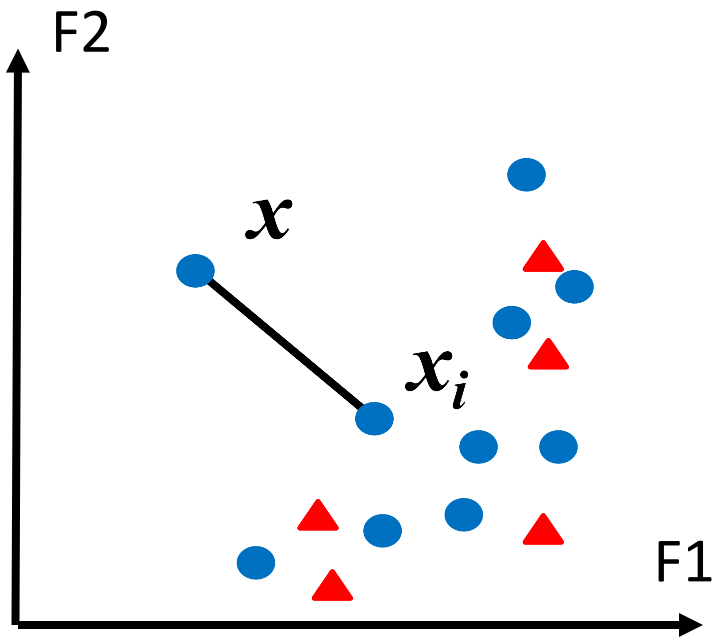
1-Nearest Neighbor Classification

- Nearest Neighbor to $\boldsymbol{x}$:

$$\boldsymbol{NN(x)} = arg\ min_{i\in\{1,..,N\}}\left(||\boldsymbol{x_i} - \boldsymbol{x}||_2^2\right)$$

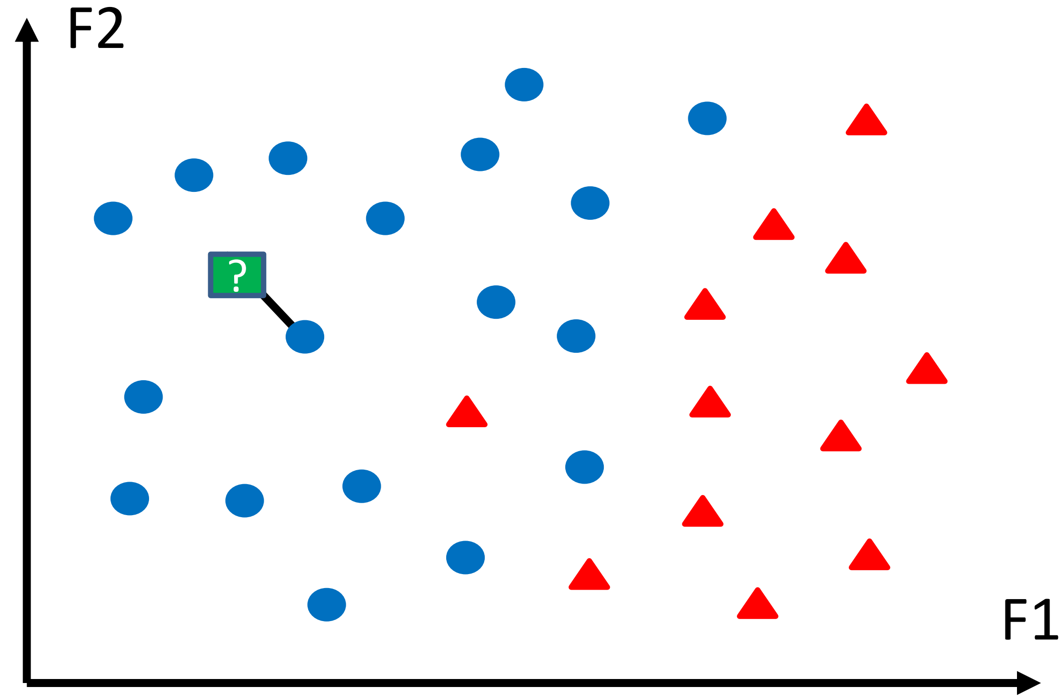
- After finding the Nearest Neighbor (NN) of $\boldsymbol{x}$, the label of $\boldsymbol{x}$ will be determined as the label of its NN:

$$y = y_{NN(\boldsymbol{x})}$$



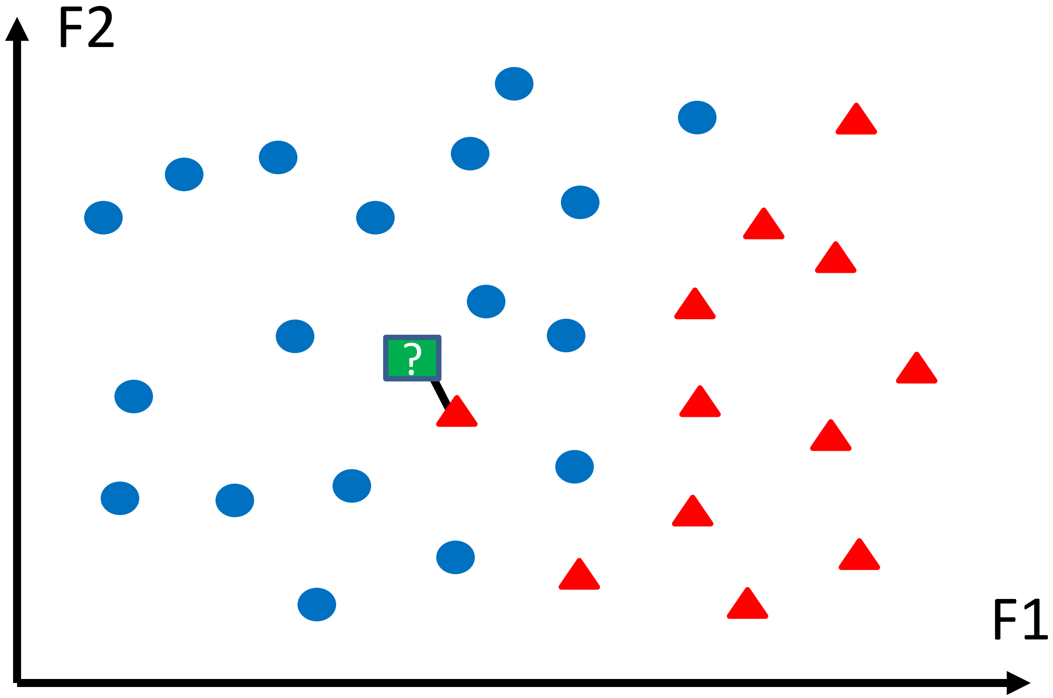
Nearest Neighbor Classification

- Nearest Neighbor: In the example below we can conclude that the unknown sample is blue. But, is this always true?



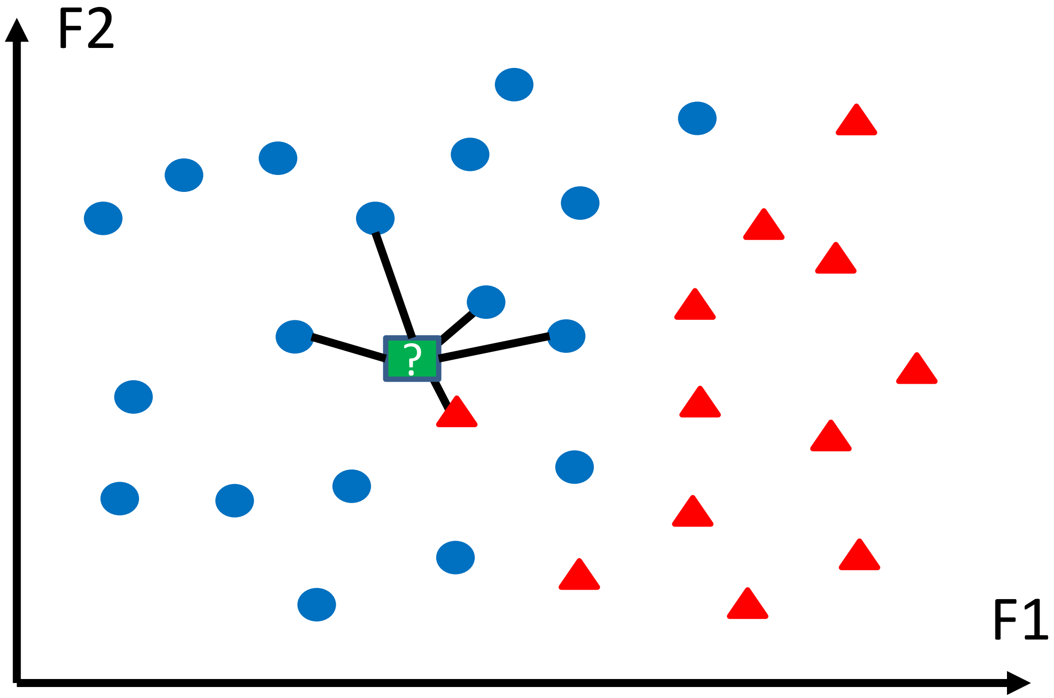
Nearest Neighbor Classification

- In this example, although this sample is in blue area, and seems to be blue, but because it happens to be close to a single red sample, it can be detected as red!!?
- Thus, making decision only based on one sample is risky!



K-Nearest Neighbor Classification

- K-Nearest Neighbor (KNN) classifier classifies objects based on **K closest** training samples in the feature space, e.g. K=5.



KNN Classification

- 1st-Nearest Neighbor: $NN_1(\boldsymbol{x})$
- 2nd-Nearest Neighbor: $NN_2(\boldsymbol{x})$
- 3rd-Nearest Neighbor: $NN_3(\boldsymbol{x})$

  :

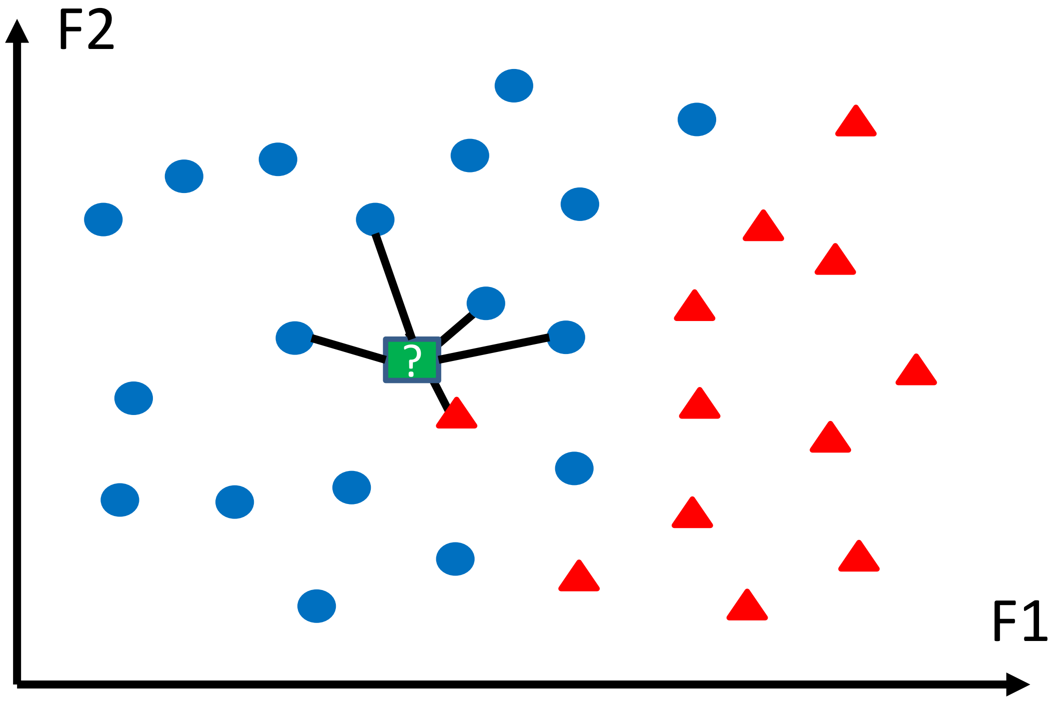
- Kth-Nearest Neighbor: $NN_K(\boldsymbol{x})$

- The set of K-Nearest Neighbors: $KNN(\boldsymbol{x}) = \{ NN_1(\boldsymbol{x}), NN_2(\boldsymbol{x}), \ldots, NN_K(\boldsymbol{x}) \}$
- **Classification rule**: **Voting**: Select the Label with the majority in $KNN(\boldsymbol{x})$.

K-Nearest Neighbor Classification

- K-Nearest Neighbor (KNN) classifier algorithm classifies objects based on K closest training samples in the feature space, e.g. K=5.

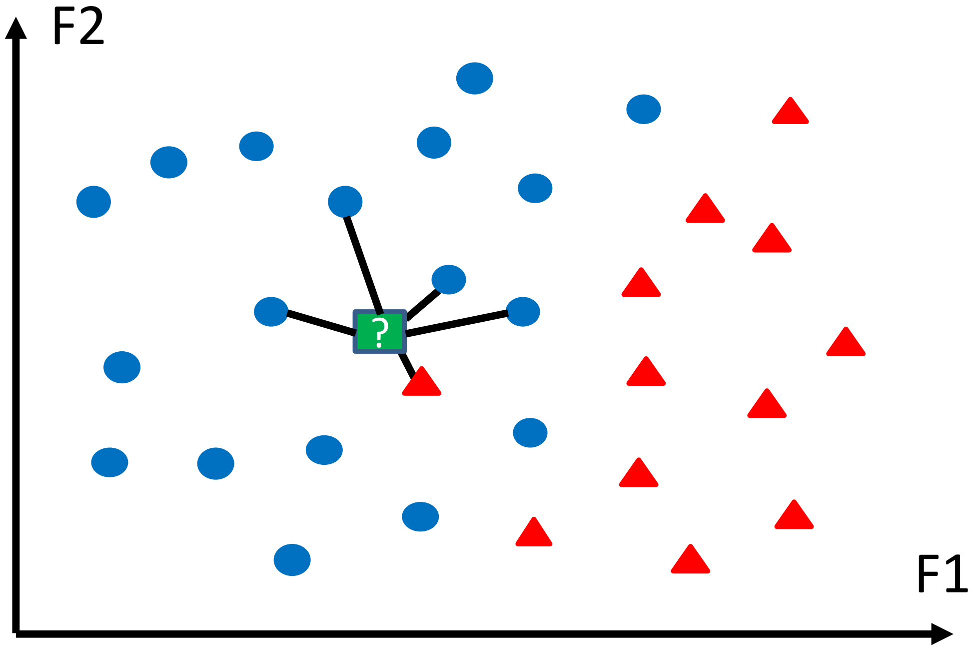
Out of 5 NN:
4 are blue, 1 is red.
Thus, our prediction is blue!



How to Select K?

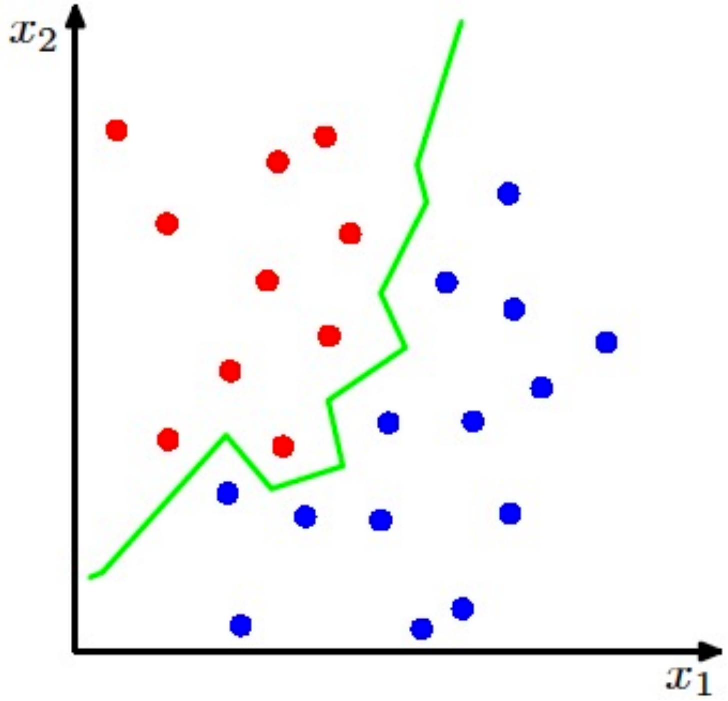
- **Benefits of a large k:**
  - Ignoring the effect of outliers
  - Better decision making

- **Benefits of a small k:**
  - Simple model
  - Low computational complexity
  - No Bias towards popular labels



Decision boundary

- For every possible point in the space, we can determine its label using the KNN rule. This gives a <u>**decision boundary**</u> that partitions the space into different regions.



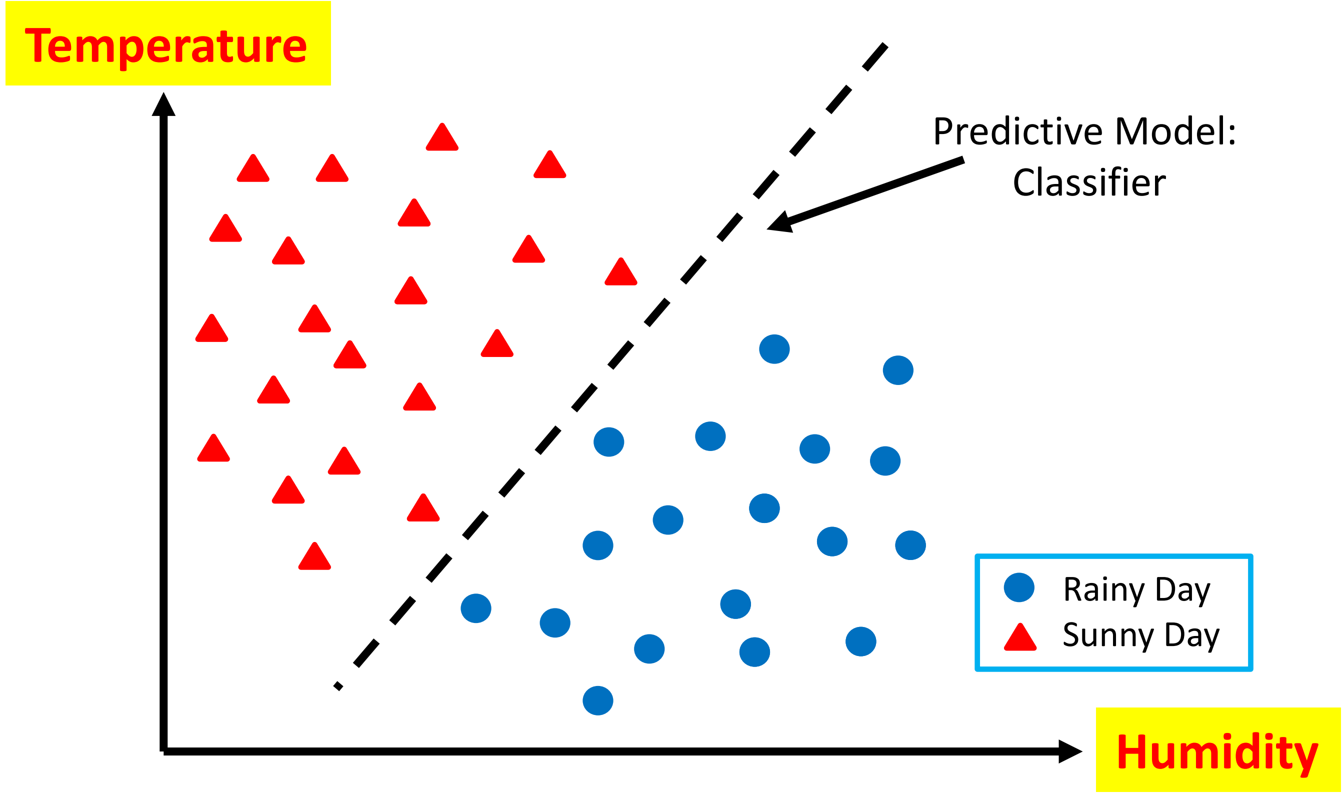
Advantages and Disadvantages

- **Advantages of using KNN Classifier:**
  - Easy and Simple to implement
  - Low Computational Complexity

- **Disadvantages:**
  - Computationally intensive for large-scale problems: Inefficient for Big Data (because we need to find millions of distances, and then find the shortest one each time)
  - Choosing the best K is challenging.

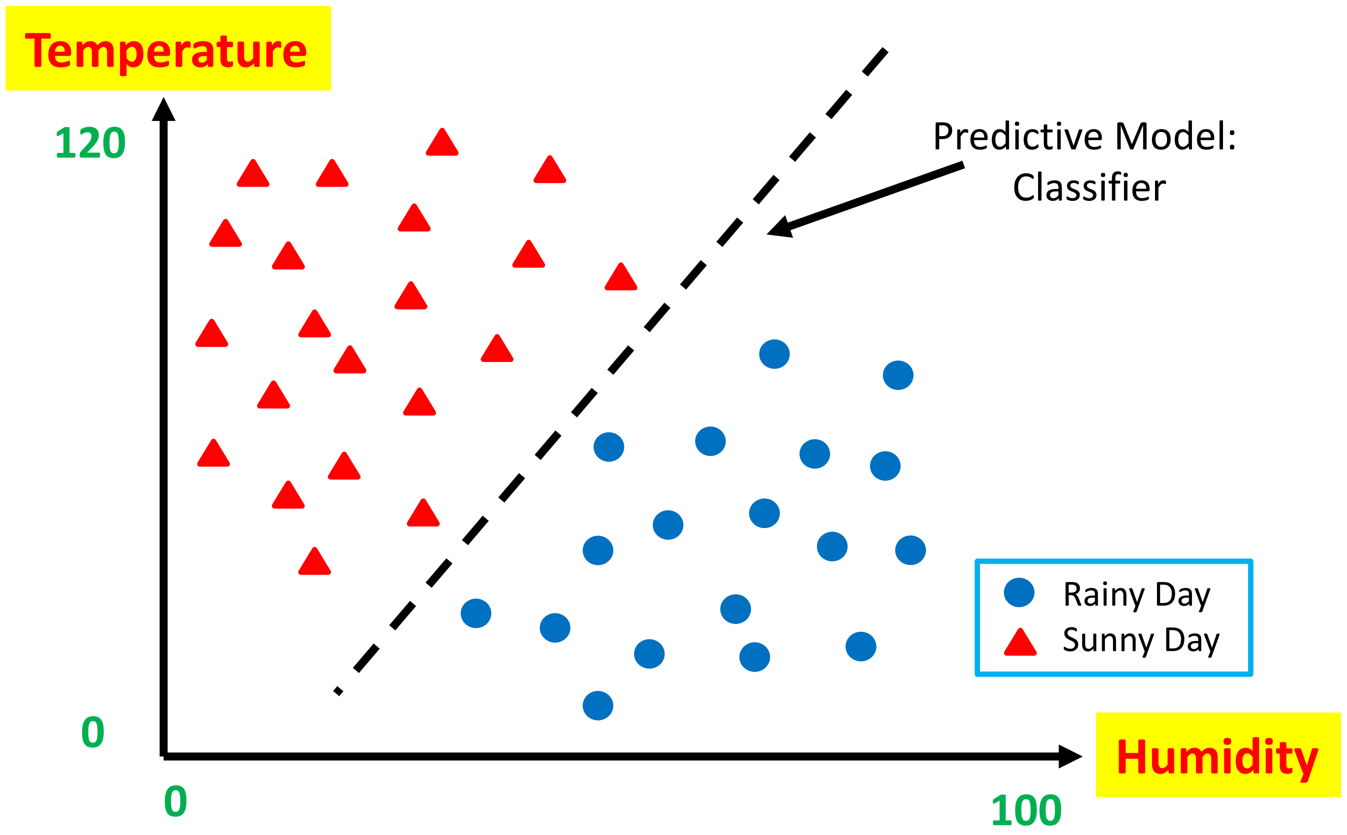
A practical hint about KNN

- Distances depend on the **unit** of the feature!



A practical hint about KNN

- In the example below, both features are ***almost*** in the same range. So, we don't have any problem!



Another Example!

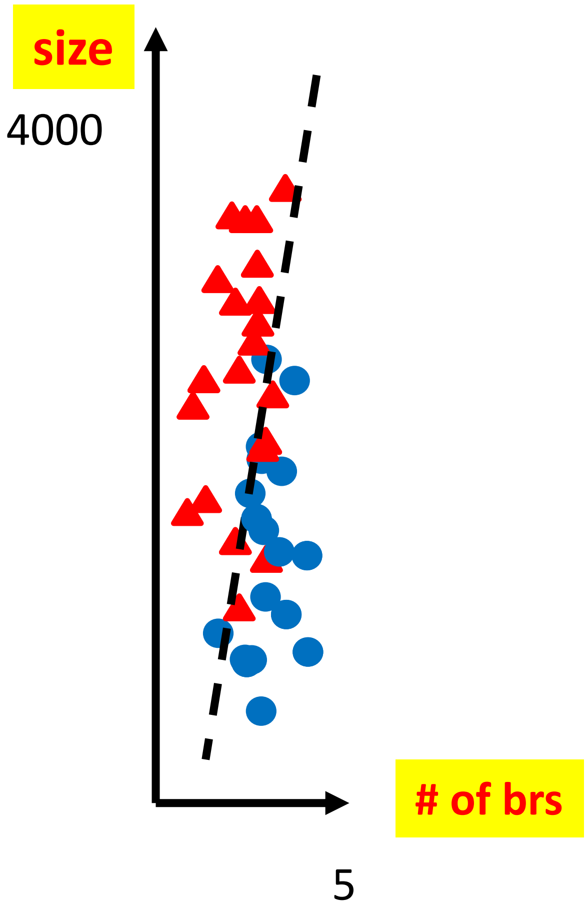
- Predicting if a house will be sold within 30 days or not based on **size of the house** and **number of bedrooms**!

Feature1 = size (0-4000 sqr feet^2).

Feature2 = number of bedrooms (1-5).

$$NN(\boldsymbol{x}) = arg\ min_{i\in\{1,..,N\}} \left(\|\boldsymbol{x_i} - \boldsymbol{x}\|_{\boldsymbol{2}}^{\boldsymbol{2}}\right)$$

$$= arg\ min_{i\in\{1,..,N\}} \left[\underbrace{(x_{i1} - x_1)^2}_{\text{Negligible}} + \underbrace{(x_{i2} - x_2)^2}_{\text{Significant}}\right]$$



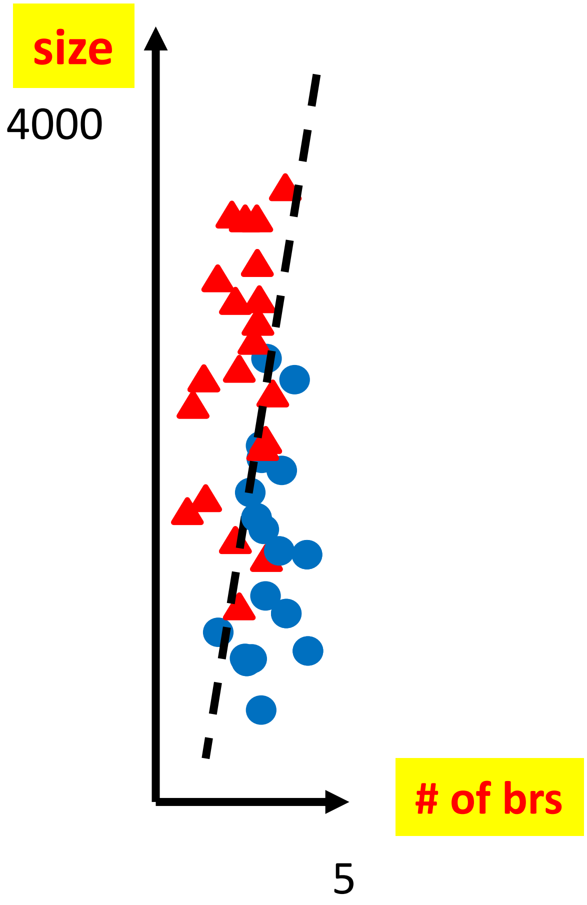
Another Example!

- Predicting if a house will be sold within 30 days or not based on **size of the house** and **number of bedrooms**!

Feature1 = size (0-4000 sqr feet^2).

Feature2 = number of bedrooms (1-5).

- **Thus, We need to Scale (Normalize) the features before training/testing.**



Example

- Predicting if a house will be sold within 30 days or not based on **size of the house** and **number of bedrooms**!

Feature1 = size (0-4000 sqr feet^2).

Feature2 = number of bedrooms (1-5).

An easy way to scale features:

F1_Scaled = size / max(size)

F2_Scaled = number of bedrooms / max(number of bedrooms)

- After normalization:

(0 <= F1_Scaled <= 1)

(0 <= F2_Scaled <= 1)

Example

- Predicting if a house will be sold within 30 days or not based on **size of the house** and **number of bedrooms**!

- Before Scaling:

	Number of Bedrooms	Size (sqr footage)	Label
1	1	1300	Sold
2	3	2400	Not-Sold
3	3	2270	Not-Sold
4	2	1450	Sold
5	2	1400	Sold
6	4	2900	Not-Sold

Example

- Predicting if a house will be sold within 30 days or not based on **size of the house** and **number of bedrooms**!

- After Scaling:

	Number of Bedrooms	Size (sqr footage)	Label
1	1/4	1300/2900	Sold
2	3/4	2400/2900	Not-Sold
3	3/4	2270/2900	Not-Sold
4	2/4	1450/2900	Sold
5	2/4	1400/2900	Sold
6	4/4	2900/2900	Not-Sold

Example

- Predicting if a house is going to be sold in 30 days based on **size of the house** and **number of bedrooms**!

Feature1 = size (0-4000 sqr feet^2).

Feature2 = number of bedrooms (1-5).

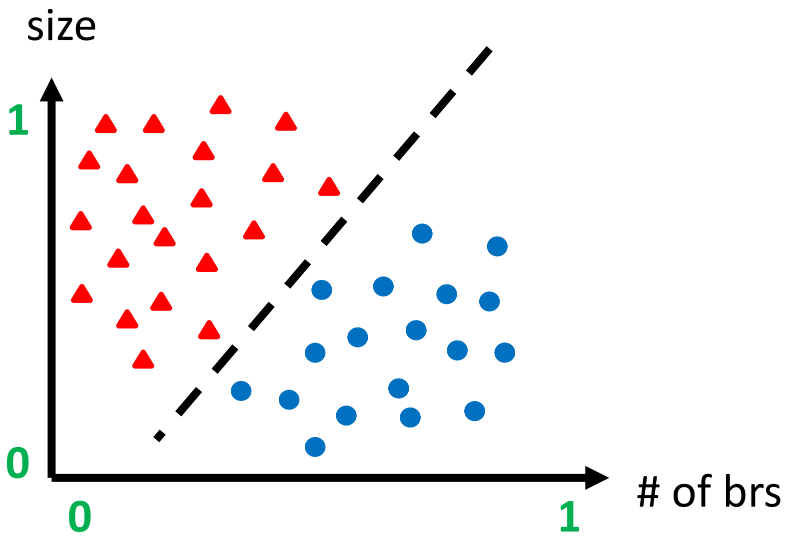
Easy way to scale features:

F1_Scaled = size / 4000

F2_Scaled = number of bedrooms / 5

(0 <= F1_Scaled <= 1)

(0 <= F2_Scaled <= 1)



A more advanced technique for normalization

- We can also **Normalize** data samples to have **zero mean** and **unit standard deviation** in each dimension (i.e. for each feature). In other word, we should normalize **each column of the feature table individually**.

A more advanced technique for normalization (Optional)

- We can **Normalize** data samples to have **zero mean** and **unit standard deviation** in each dimension (for each feature). In other word, we should normalize **each column of the feature table**.

- Compute the mean and standard deviation for each feature (x_{nd} is the dth feature of nth data sample):

$$mean(x_d) = \bar{x}_d = \frac{1}{N}\sum_{n=1}^{N} x_{nd}$$

$$std(x_d) = s_d = \sqrt{\frac{1}{N-1}\sum_{n=1}^{N}(x_{nd} - \bar{x}_d)^2}$$

A more advanced technique for normalization (Optional)

- We can **Normalize** data samples to have **zero mean** and **unit standard deviation** in each dimension (for each feature). In other word, we should normalize **each column of the feature table**.

- Compute the mean and standard deviation for each feature (x_{nd} is the dth feature of nth data sample):

$$mean(x_d) = \bar{x}_d = \frac{1}{N}\sum_{n=1}^{N} x_{nd} \qquad std(x_d) = s_d = \sqrt{\frac{1}{N-1}\sum_{n=1}^{N}(x_{nd} - \bar{x}_d)^2}$$

- Scale the feature accordingly

$$x_{nd} \leftarrow \left(\frac{x_{nd} - \bar{x}_d}{s_d}\right)$$

Thank You!

Questions?