



Introduction to Data Science

(Lecture 3)

Dr. Mohammad Pourhomayoun

Assistant Professor

Computer Science Department

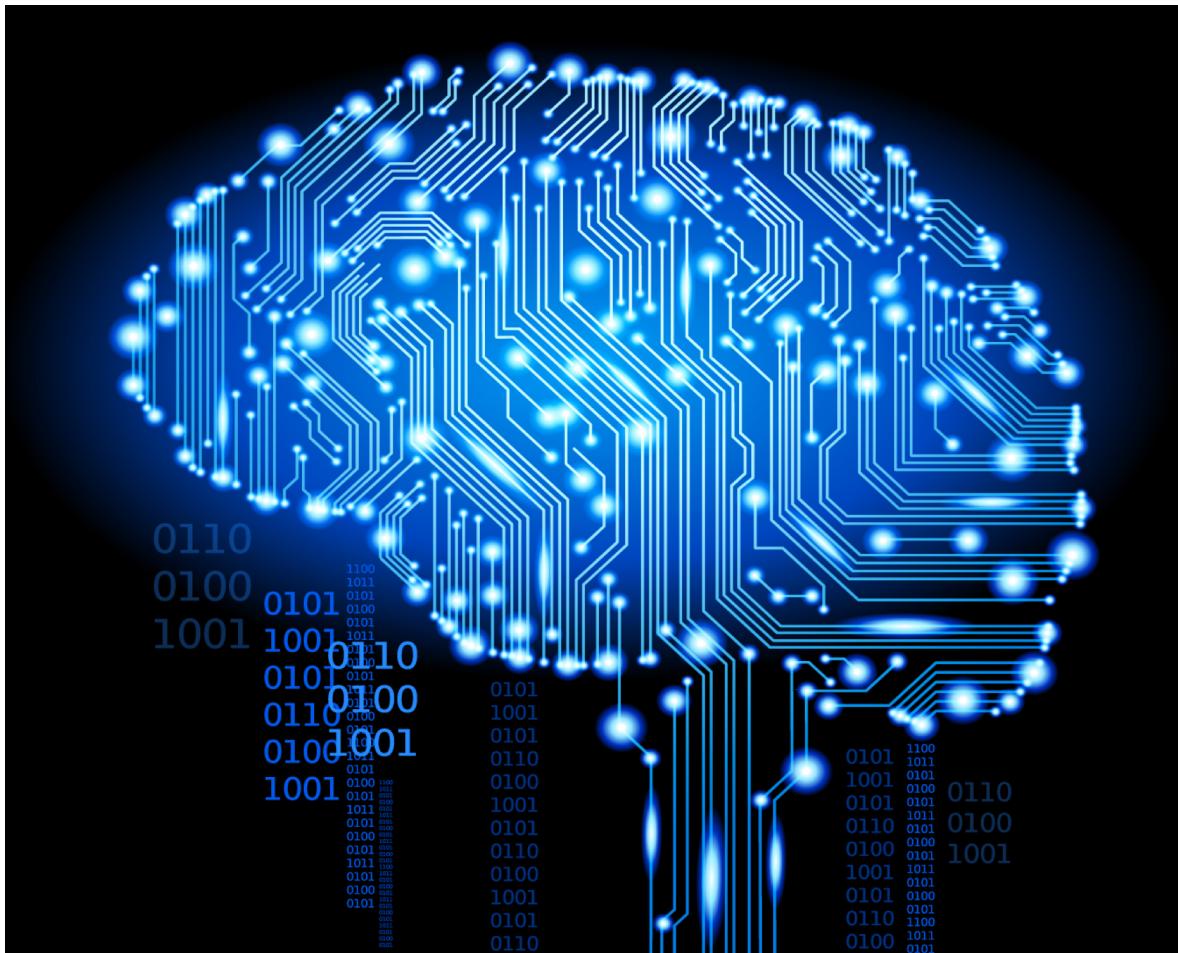
California State University, Los Angeles





Data Analytics and Machine Learning

What is Machine Learning?



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.



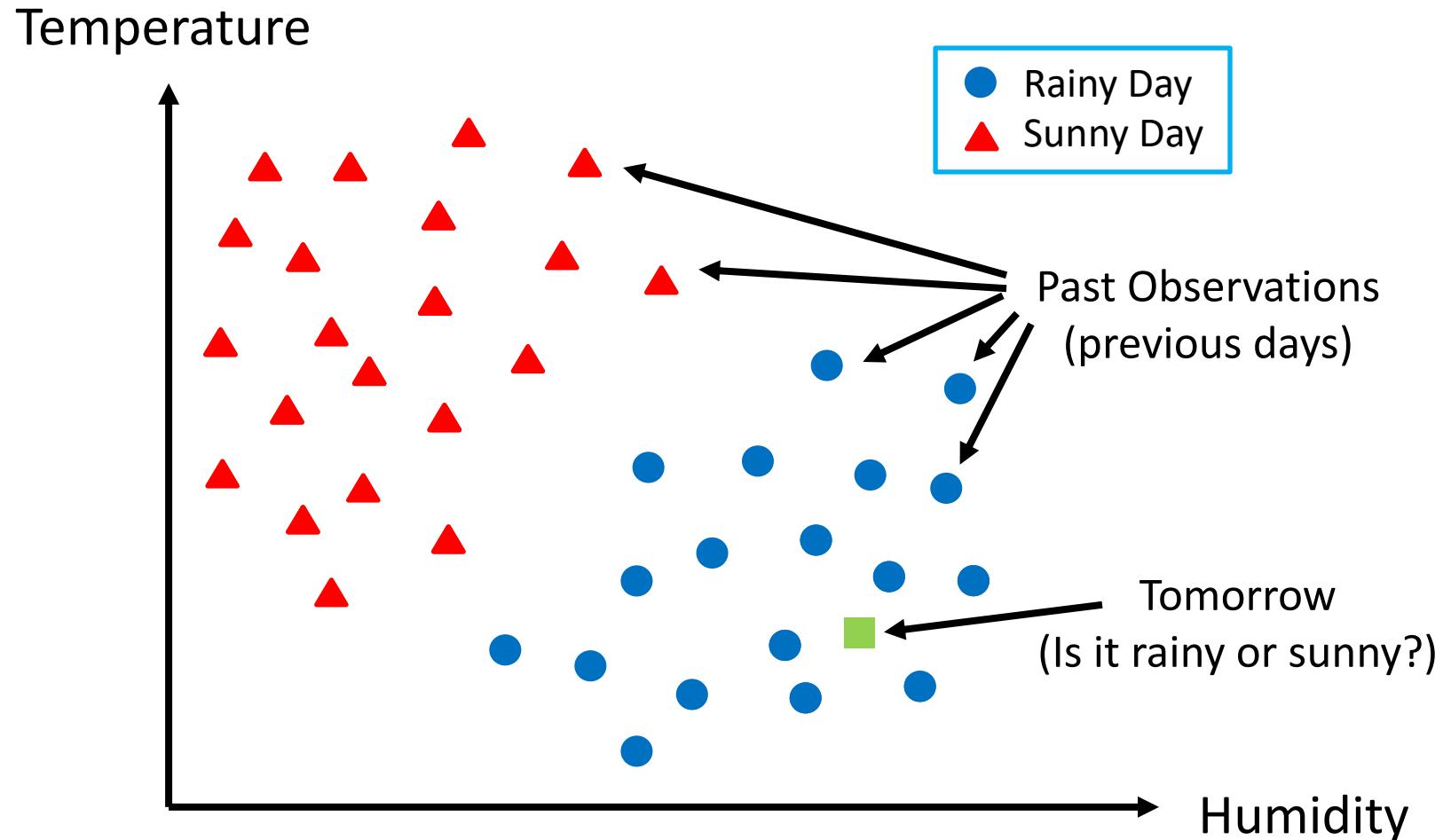
Example: Weather Forecasting

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

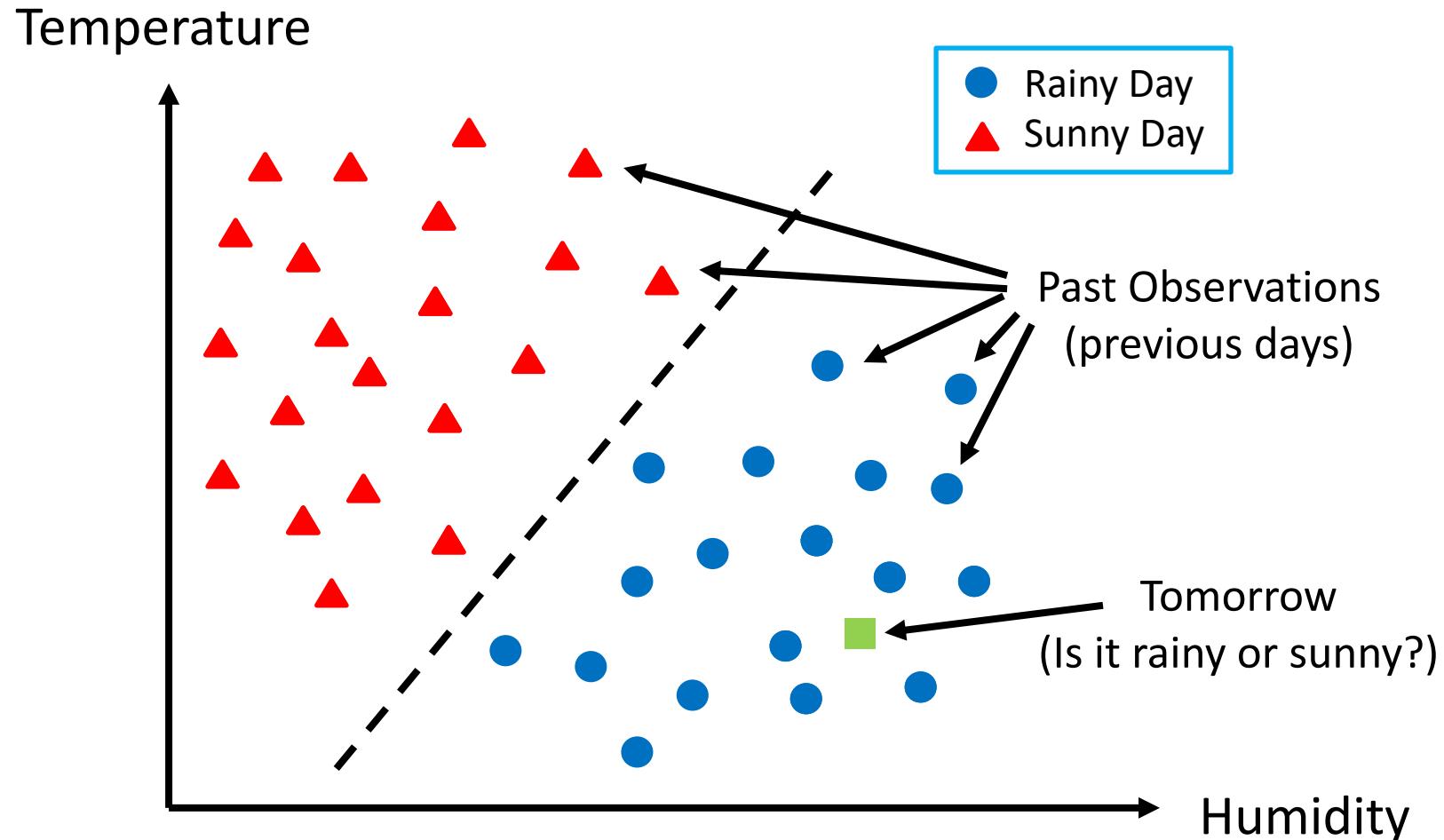
- **Questions:** Now, If we know the Temperature and Humidity of tomorrow, can we predict tomorrow's outlook (predict whether tomorrow is rainy or sunny)?

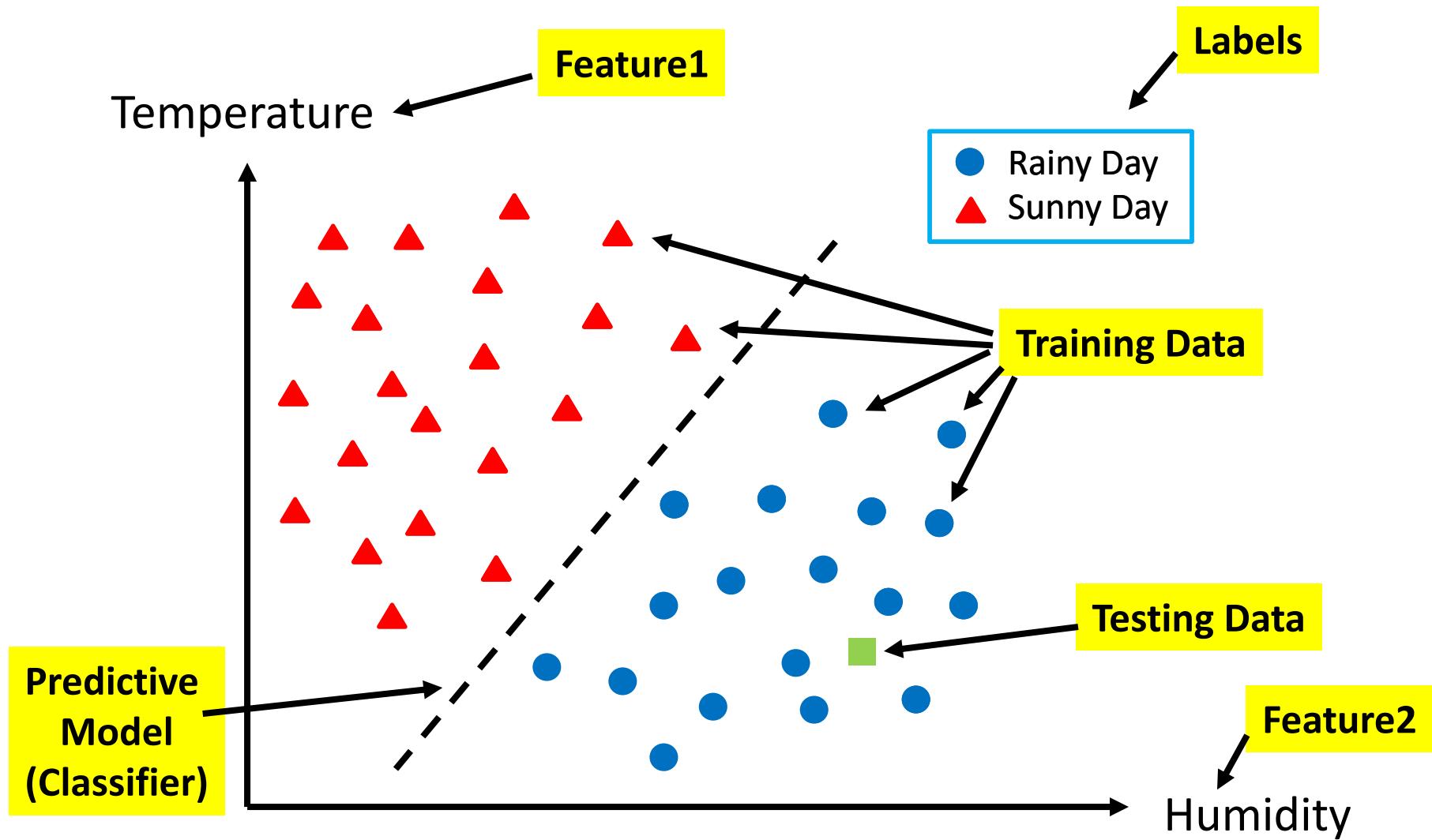


Example: Weather Forecasting

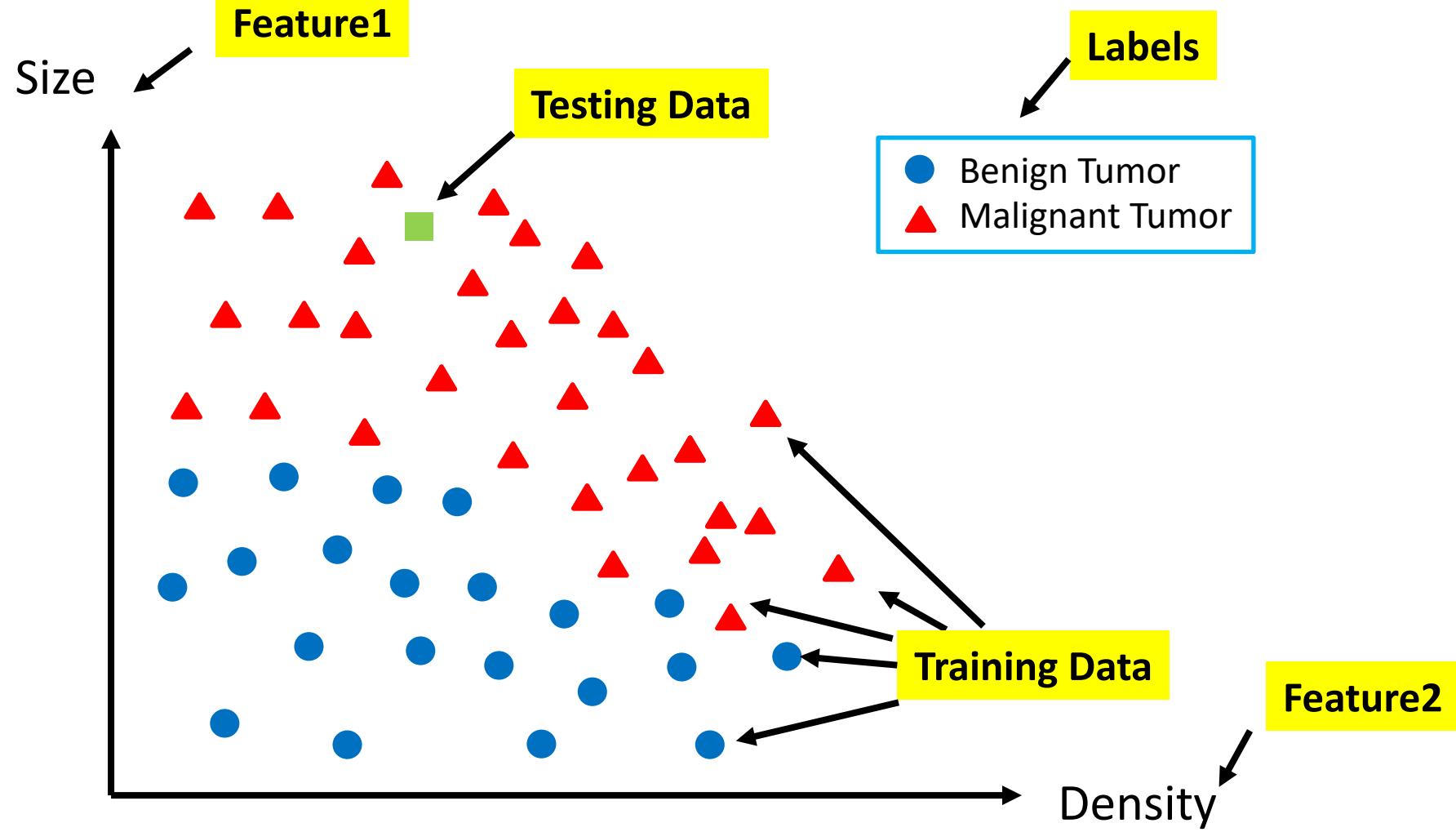


Example: Weather Forecasting





Example: Predicting Cancer



More Terminology

- **Training Stage (Modeling):** Building a predictive model based on the training dataset (past data).
 - The model does not have to be **perfect**. As long as it is close, it is useful.
 - We should tolerate randomness and **mistakes**.
- **Testing Stage (Prediction):** Applying the trained model to forecast what is going to happen in future (on future testing data)





Machine Learning Settings

Common Learning Settings

- Supervised learning
- Unsupervised learning
- Semi-supervised learning
- Reinforcement learning
- Transfer learning
- Active learning



- **Supervised learning:** Learning from labeled observations.
 - In training stage, the algorithm is presented with features and their known labels, and the goal is to train a model that maps future inputs to new labels.
 - E.g. Neural Networks (ANN), K-Nearest Neighbors classifier (KNN), Decision Tree classifier, Linear Regression, Logistic Regression, Polynomial Regression, Random Forest, ...
- **Unsupervised learning:** Learning from unlabeled observations.
 - The algorithm is presented **Only** with features! The goal is to Discover hidden patterns and structure from features alone. It is like a Data Exploration to find hidden patterns.
- **Semi-supervised learning:** Labels are provided only for a part of the training data. The remaining data is unlabeled.
- **Reinforcement learning:** Learning from an *agent* taking *actions* in an *environment* so as to maximize a long-term *reward*. E.g. Game Theory, Control Theory.
- **Transfer learning:** Learning from a dataset while solving a problem, and then applying the **extracted knowledge** to a different but related dataset/problem.
- **Active learning:** Similar to Semi-Supervised Learning, but the algorithm is able to interactively query the user or some other information source to obtain the labels as needed.



Common Learning Settings

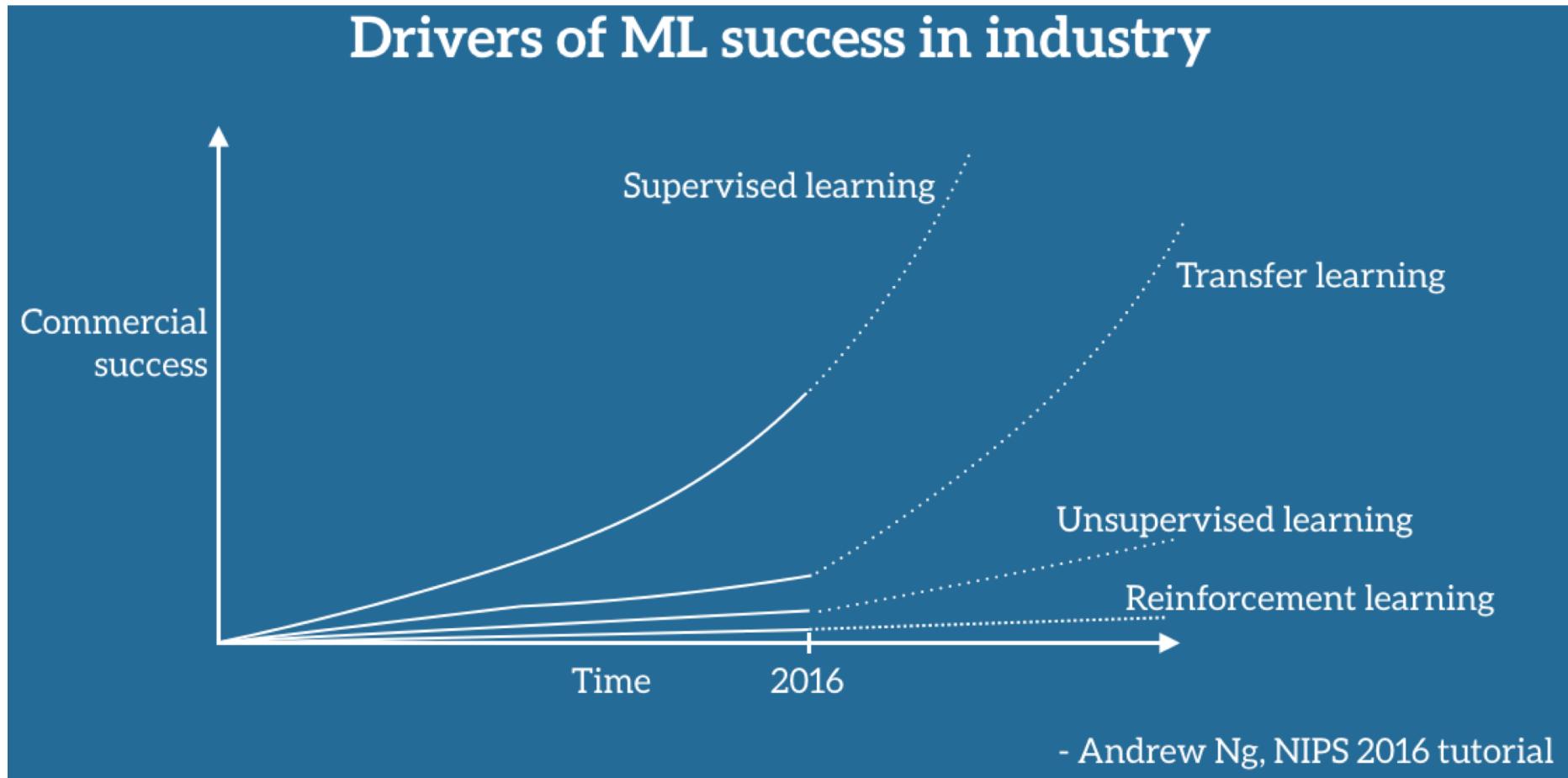
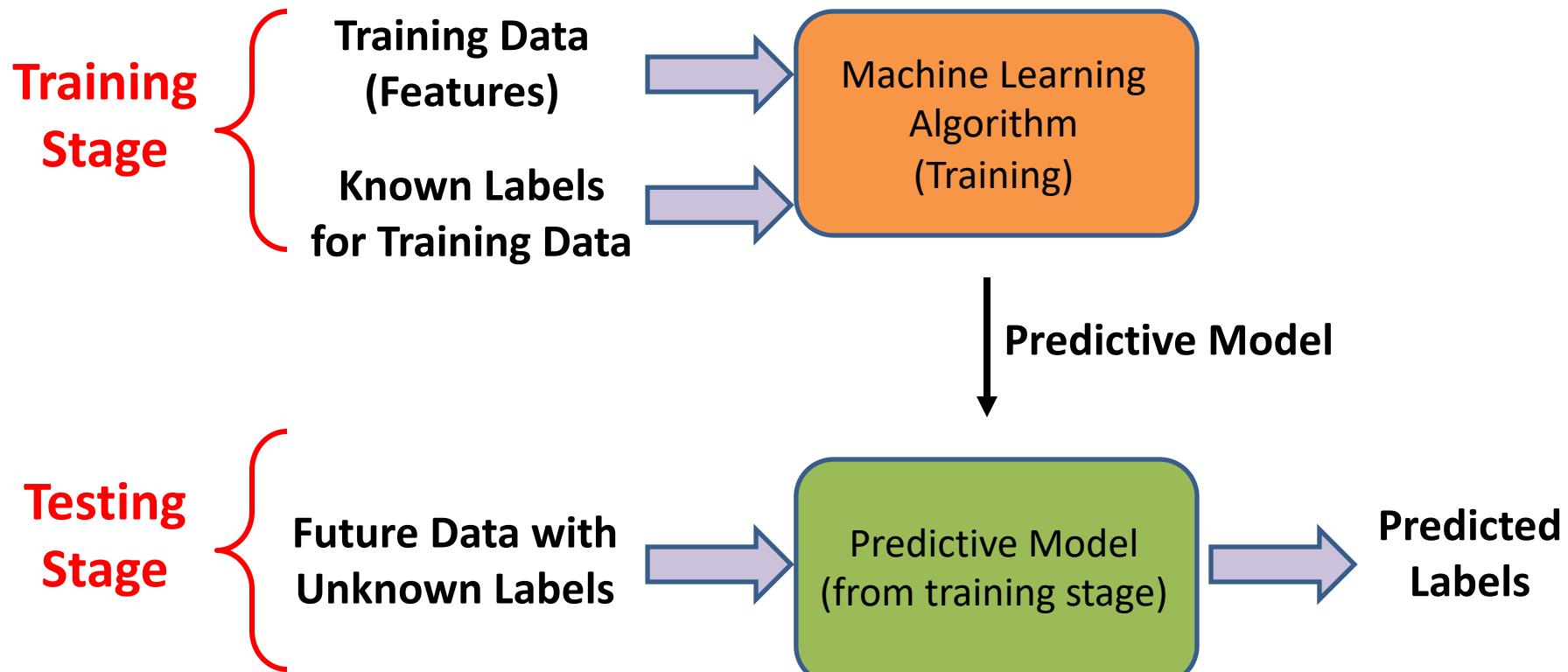


Figure Reference: Andrew Ng



Supervised Learning: Learning from labeled Data

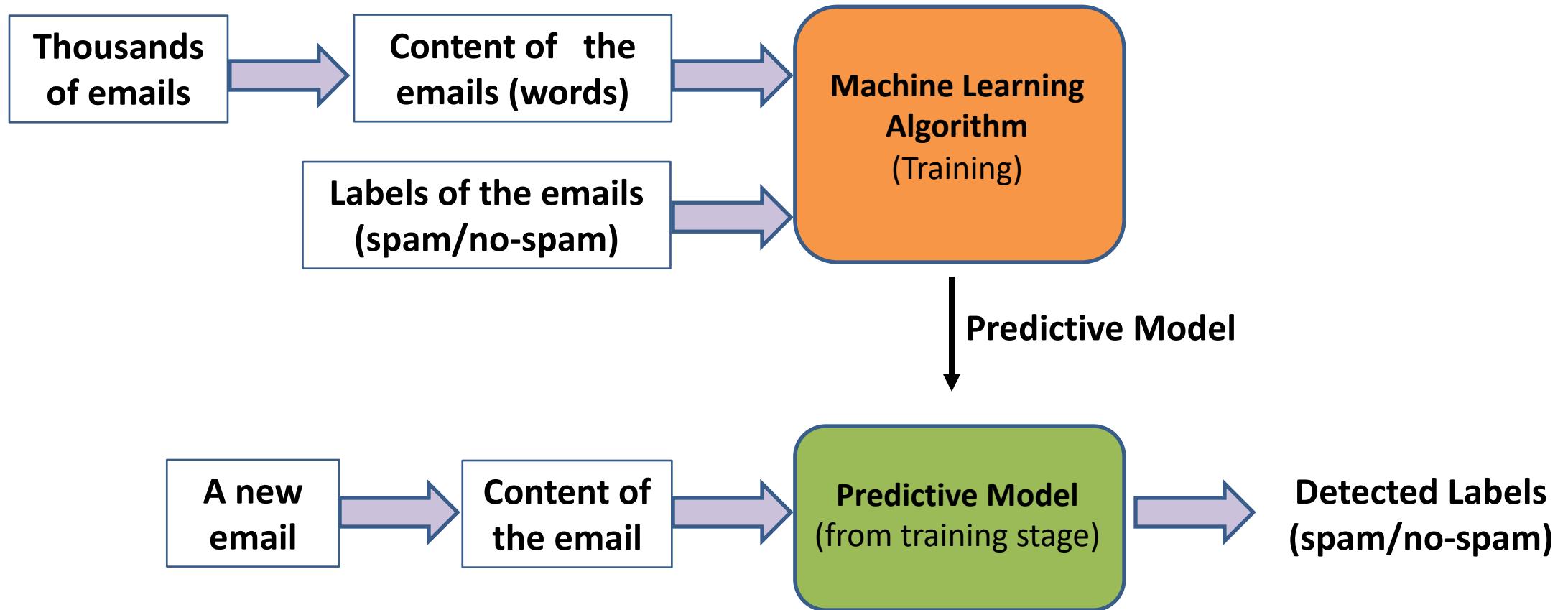


Example for Supervised Learning: Spam Detection

- The algorithm will learn from **previous data** (previous emails) that what combination of words indicate suspicious emails.



Example for Supervised Learning: Spam Detection

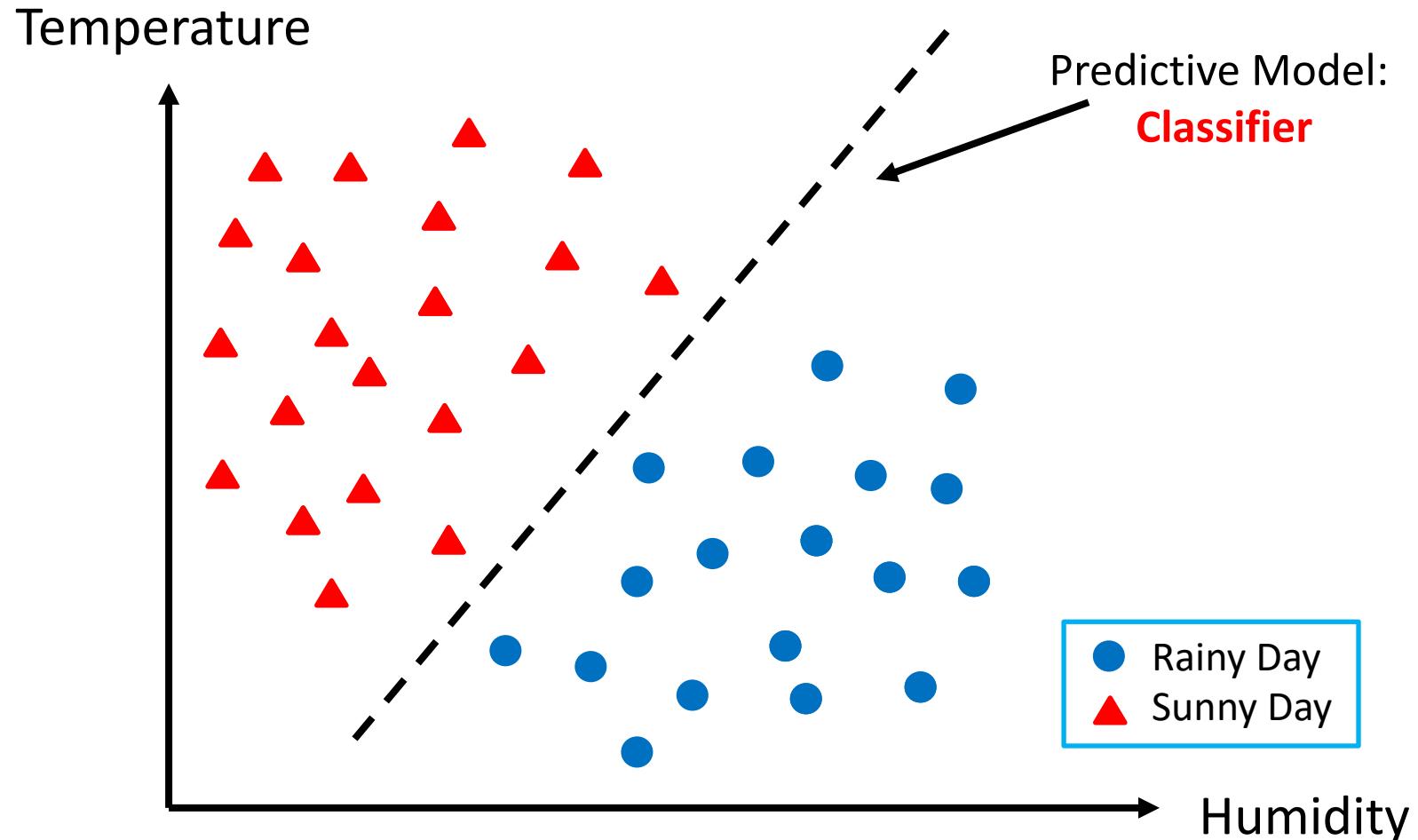


Two Important Approaches of Supervised Learning:

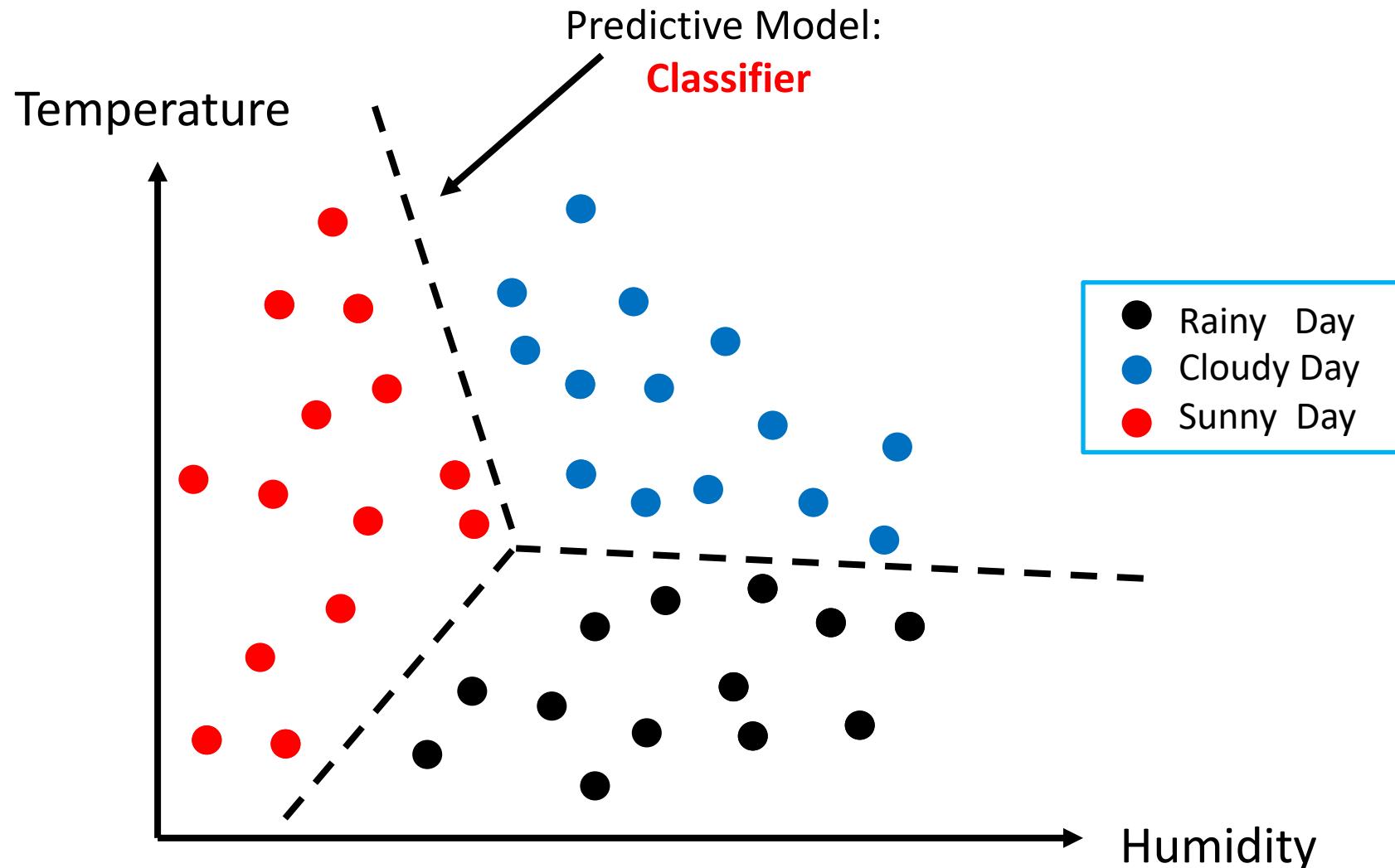
- **Classification:** Predict a discrete valued output for each observation.
 - Labels are **discrete (categorical)**
 - Labels can be binary (e.g., rainy/sunny, spam/non-spam,) or non-binary (e.g., rainy/sunny/cloudy, object recognition (100classes))
- **Regression:** Predict a continuous valued output for each observation.
 - Labels are **continuous (numeric)**, e.g., stock price, housing price
 - Can define ‘closeness’ when comparing prediction with true values



Classification Example: Binary Label



Classification Example: Multiple Label



Classification Example: Multiple Label

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

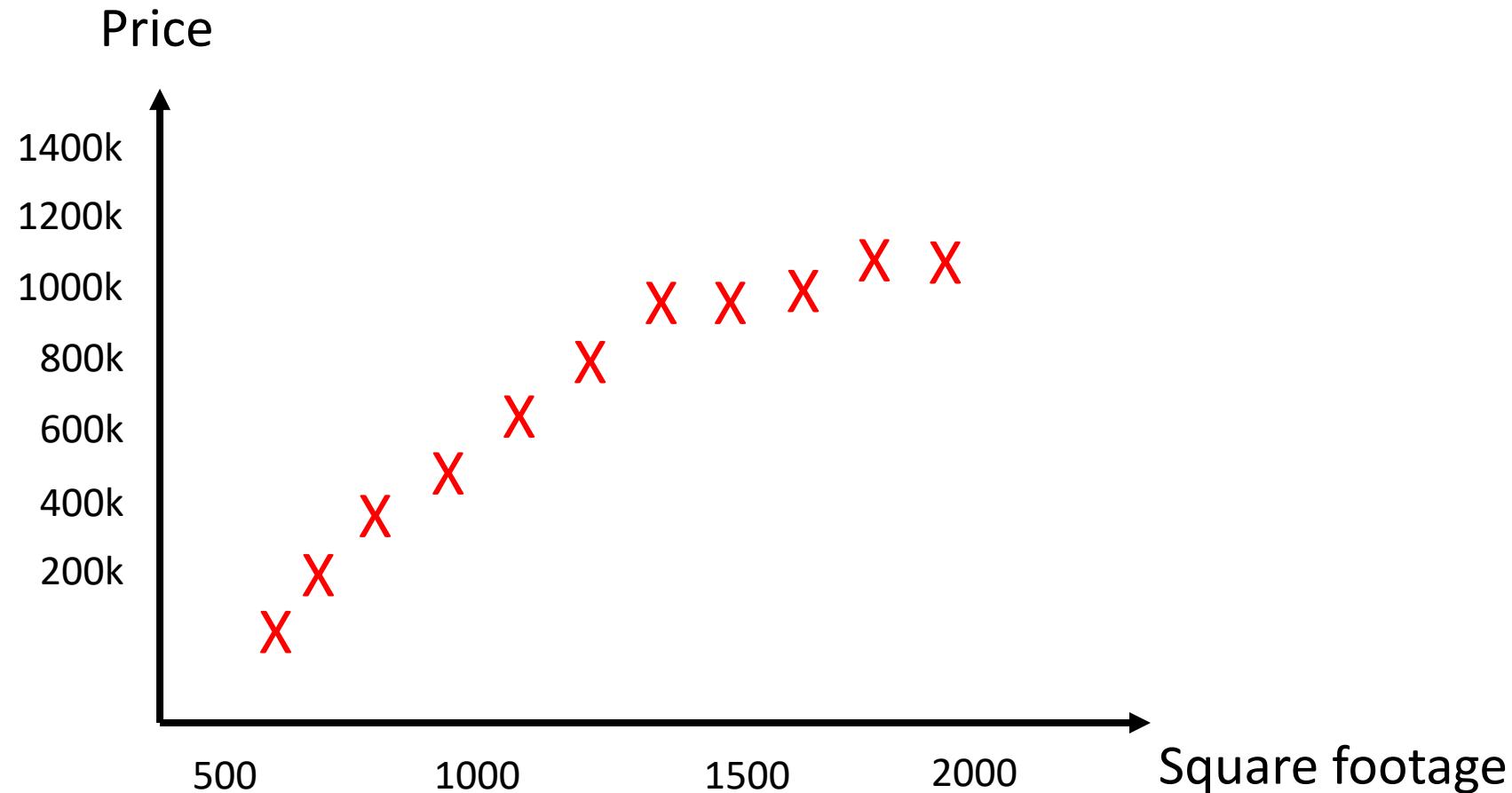


Two Important Approaches of Supervised Learning:

- **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



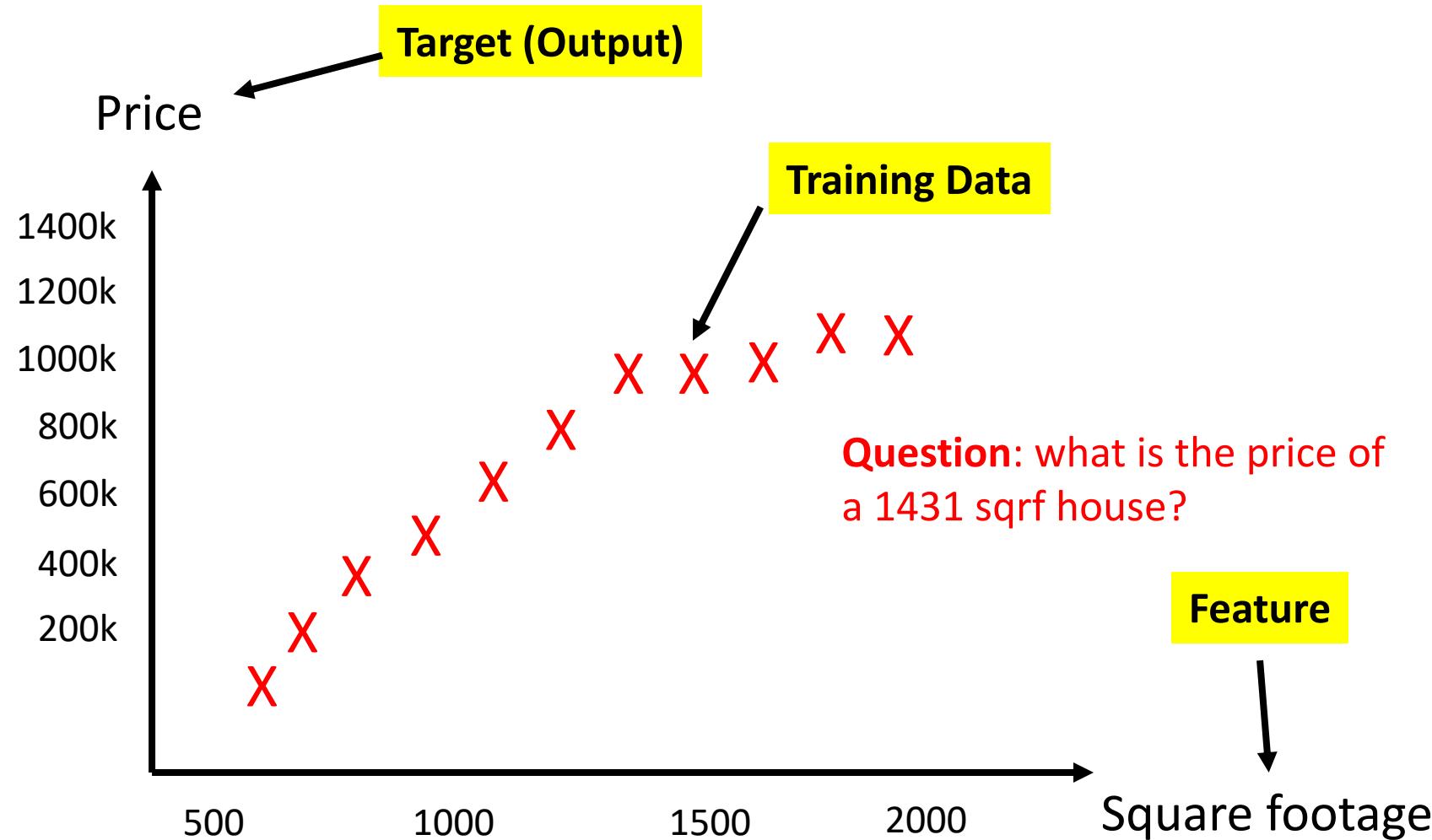
Regression Example: Housing Price



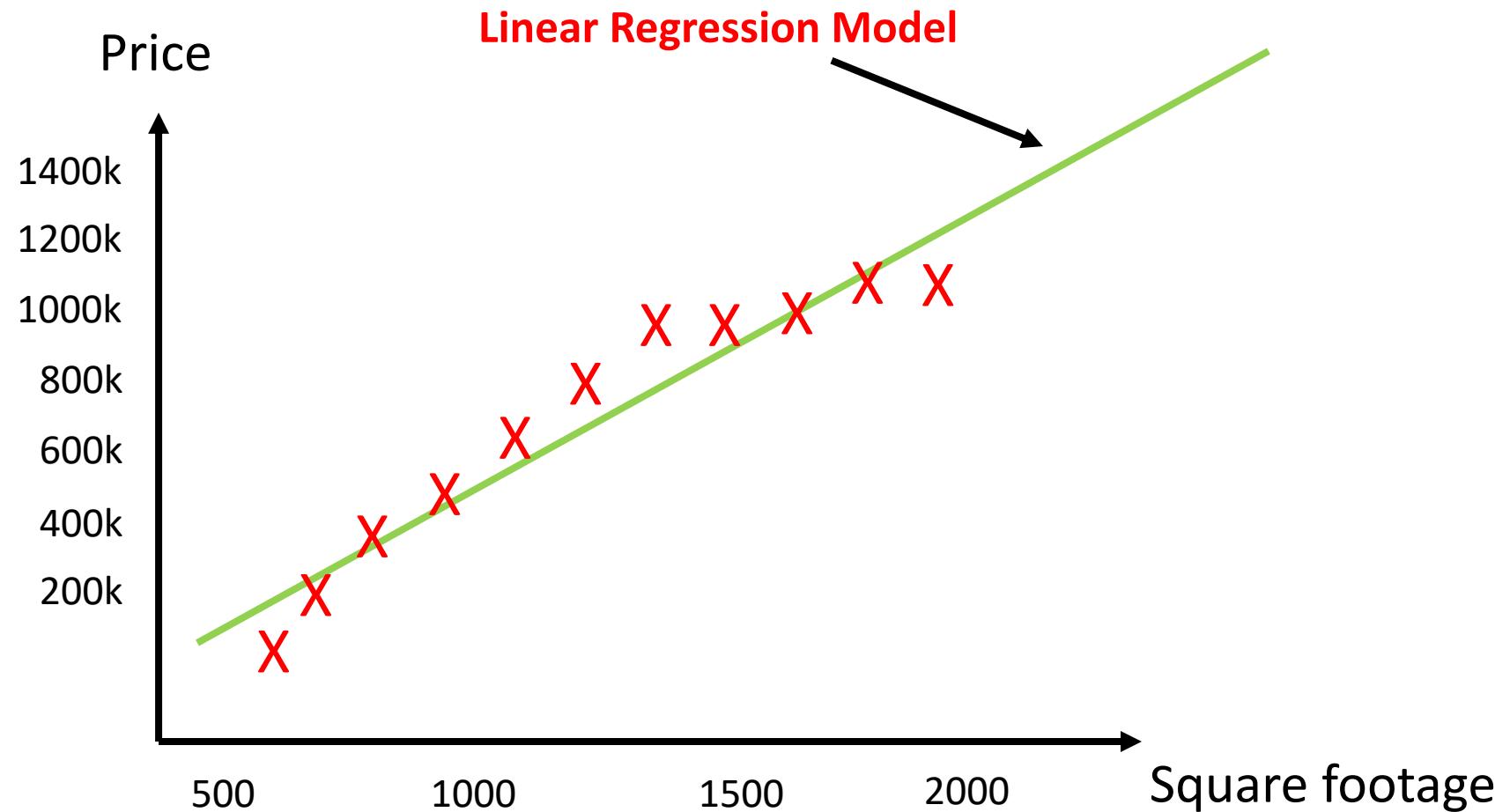
Regression Example: Housing Price



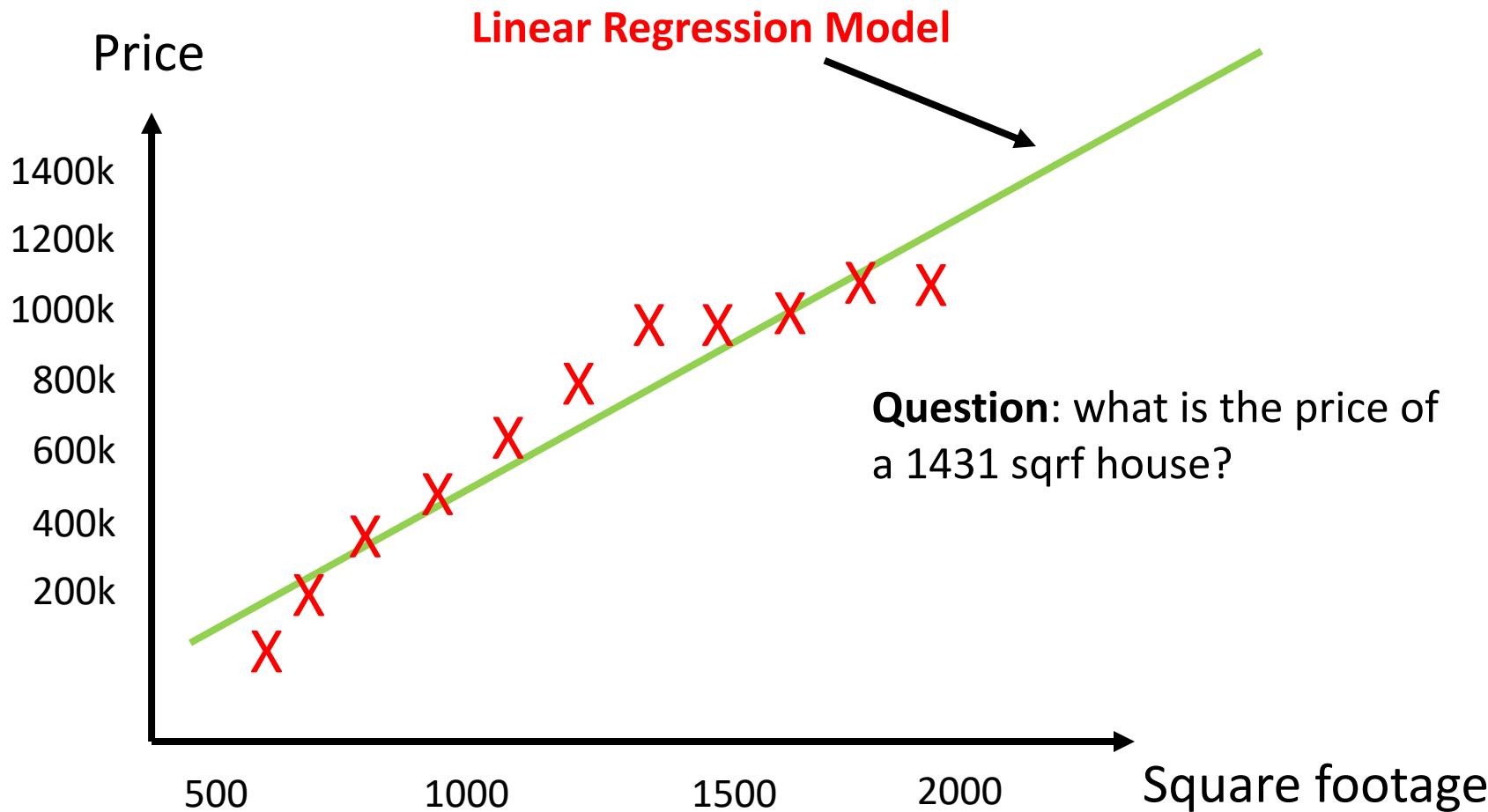
Regression Example: Housing Price



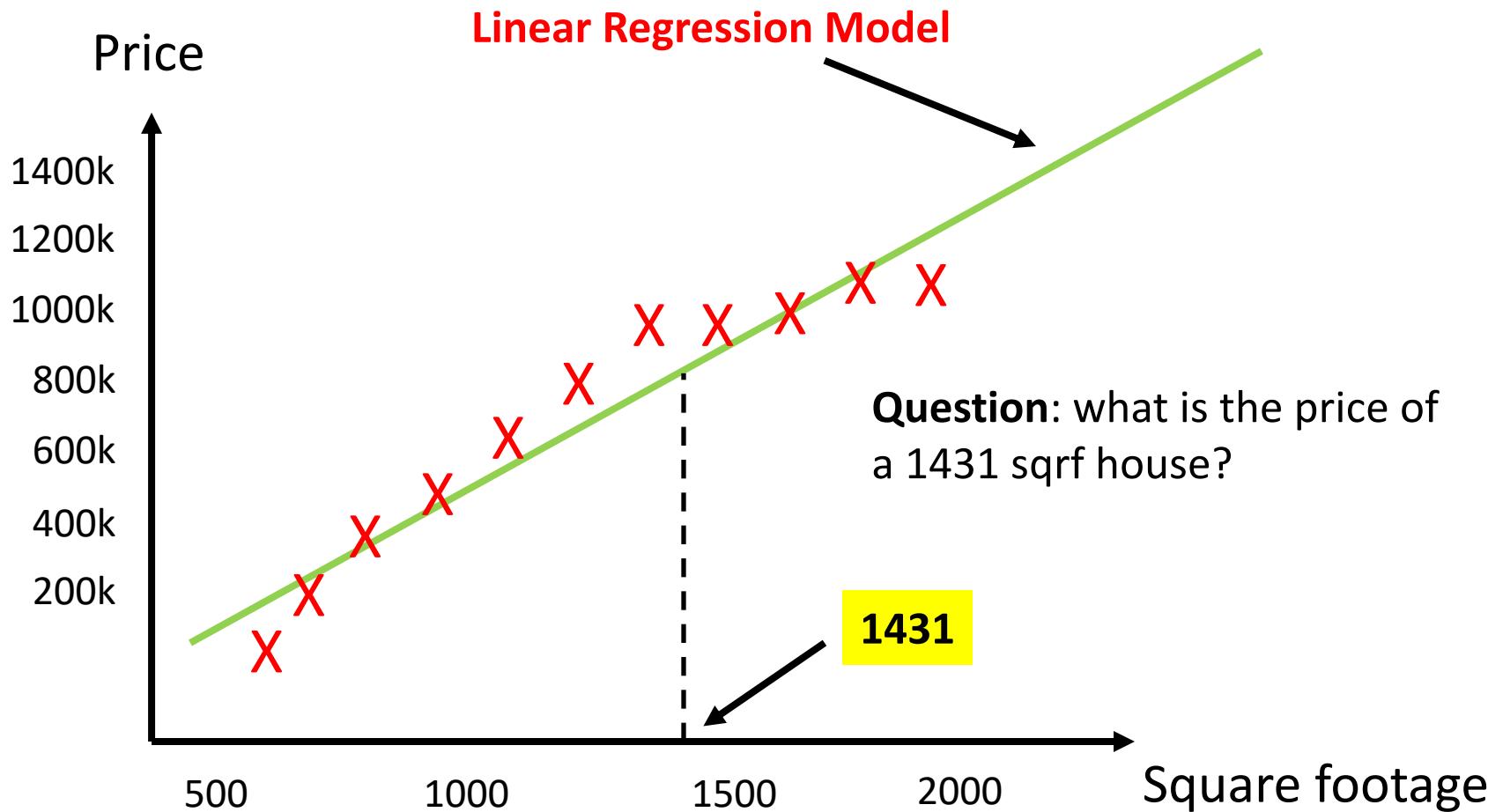
Regression Example: Housing Price



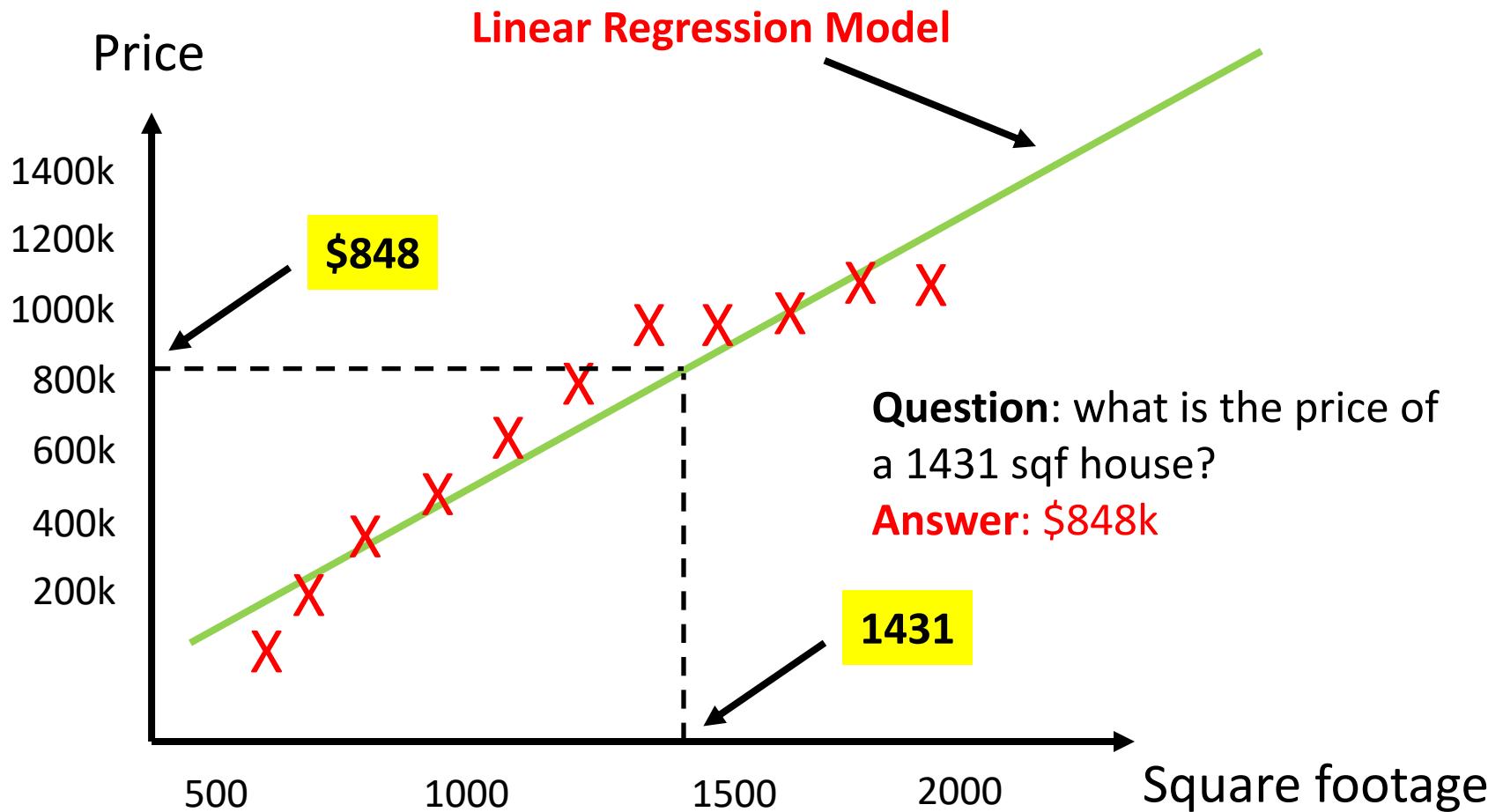
Regression Example: Housing Price



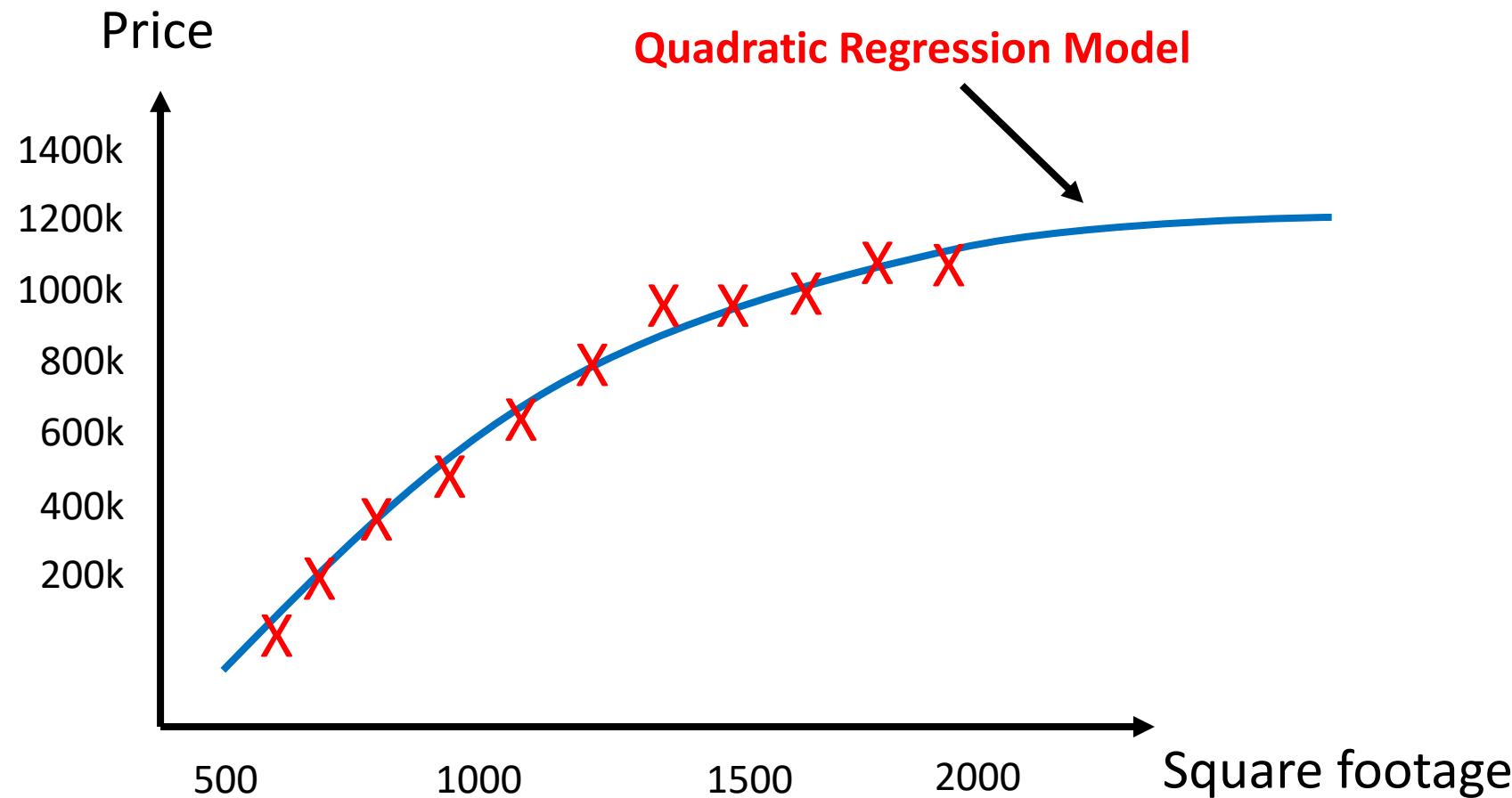
Regression Example: Housing Price



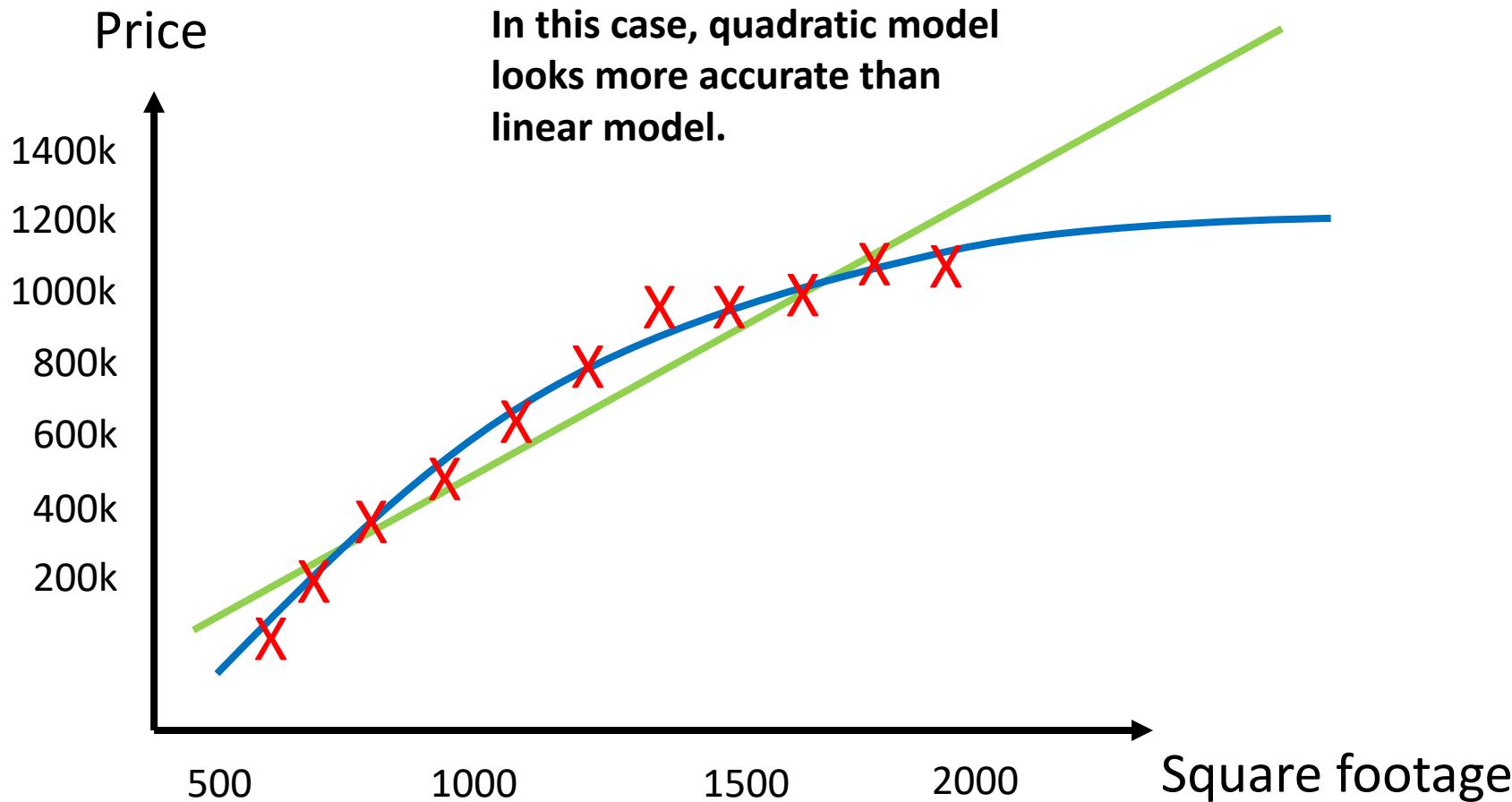
Regression Example: Housing Price



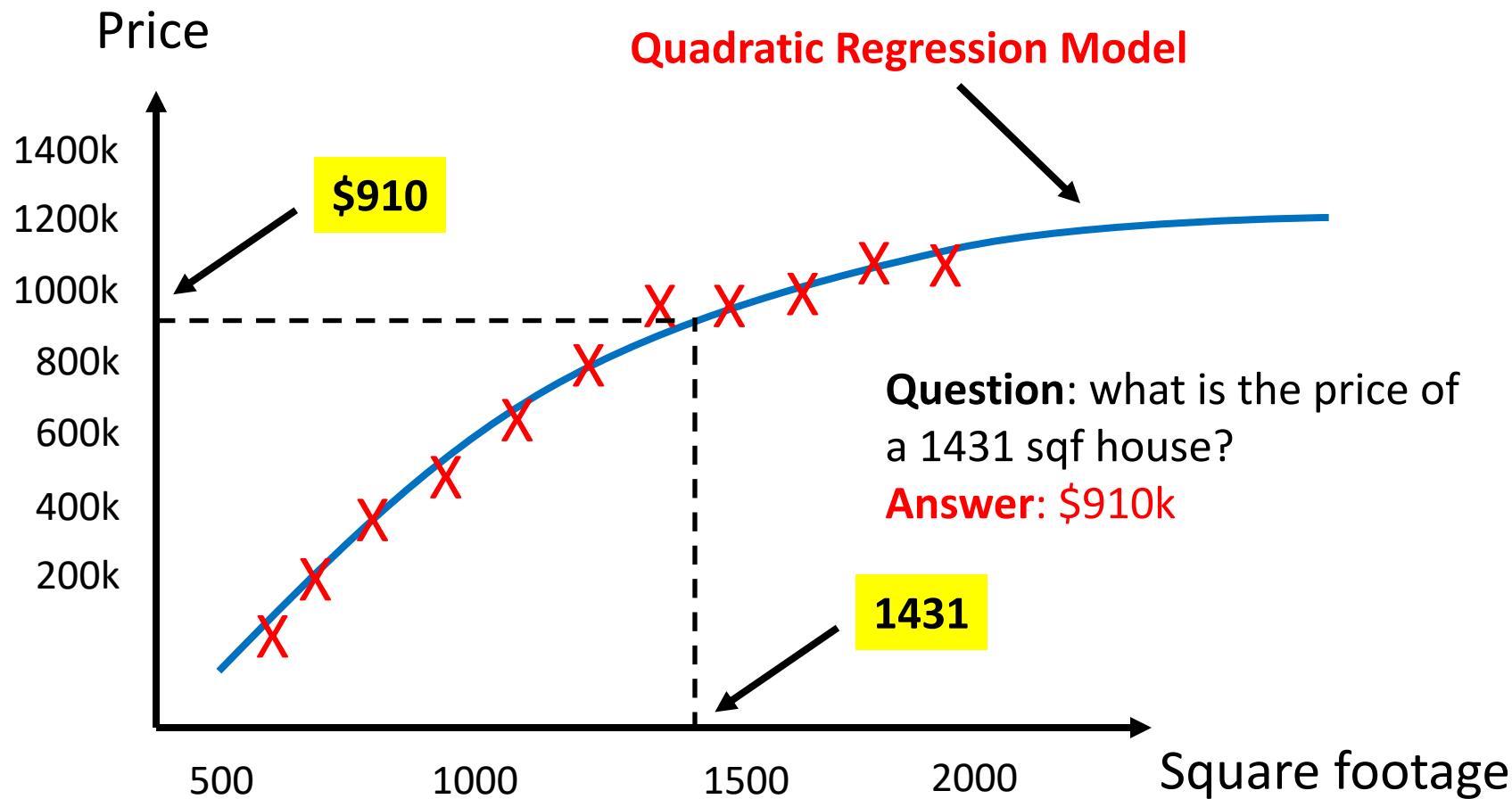
Regression Example: Housing Price



Regression Example: Housing Price



Regression Example: Housing Price

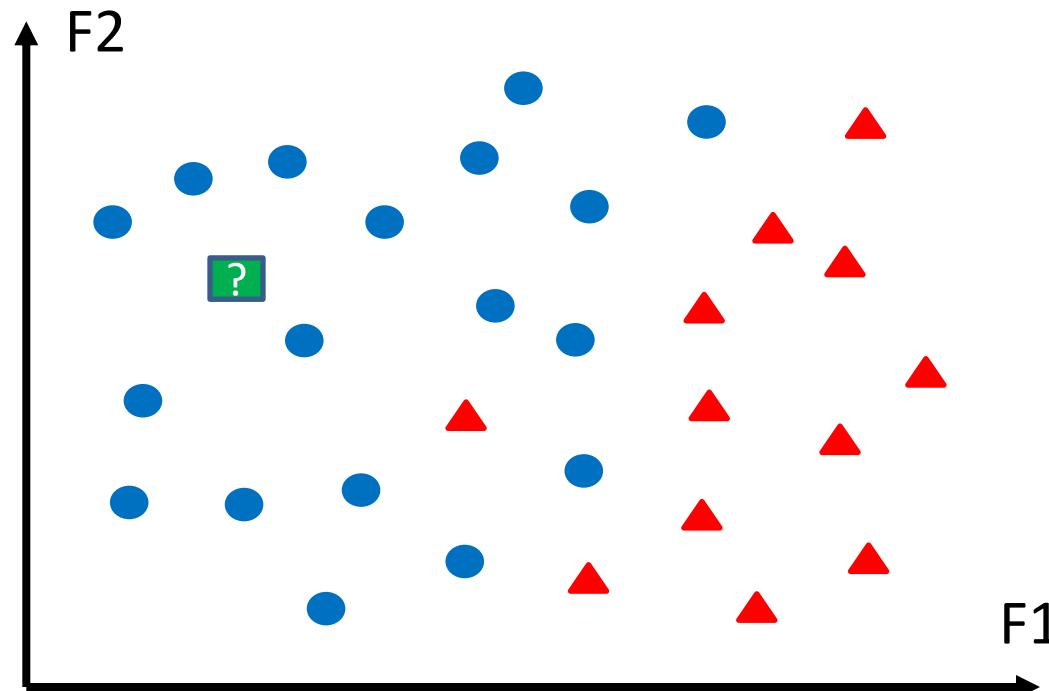




K-Nearest Neighbor Classifier (KNN Classifier)

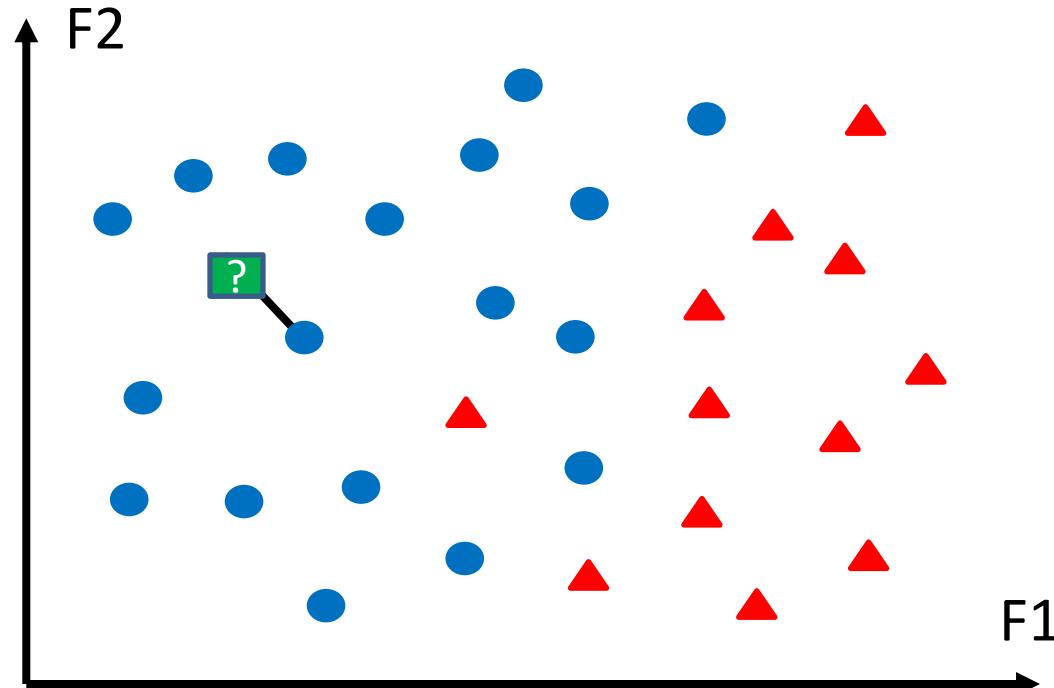
Nearest Neighbor Classification

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



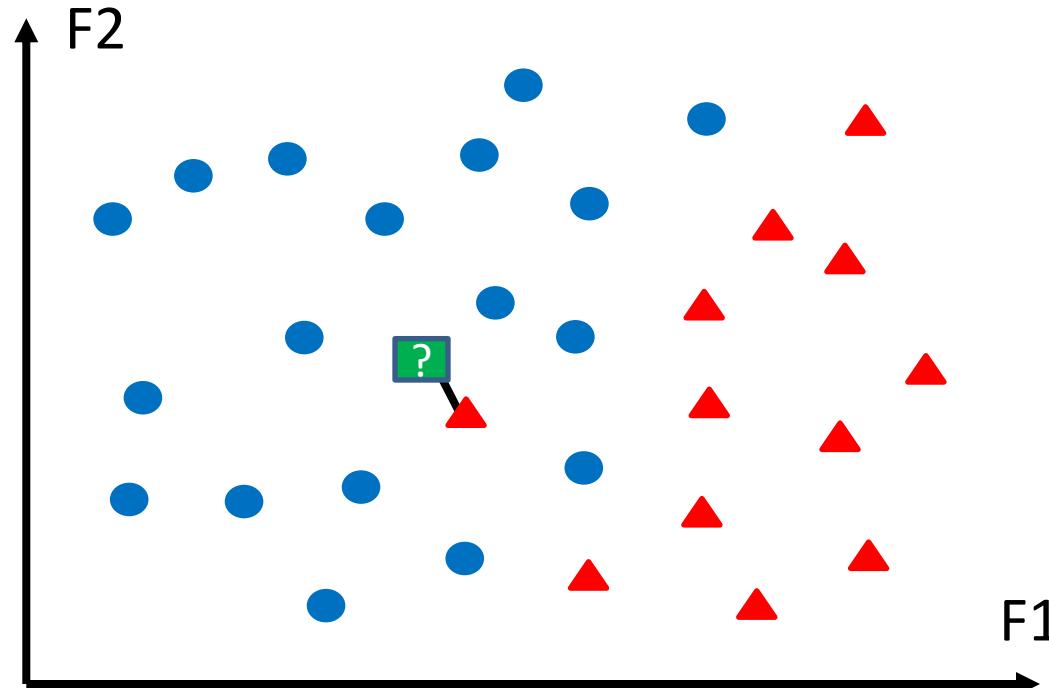
Nearest Neighbor Classification

- 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?



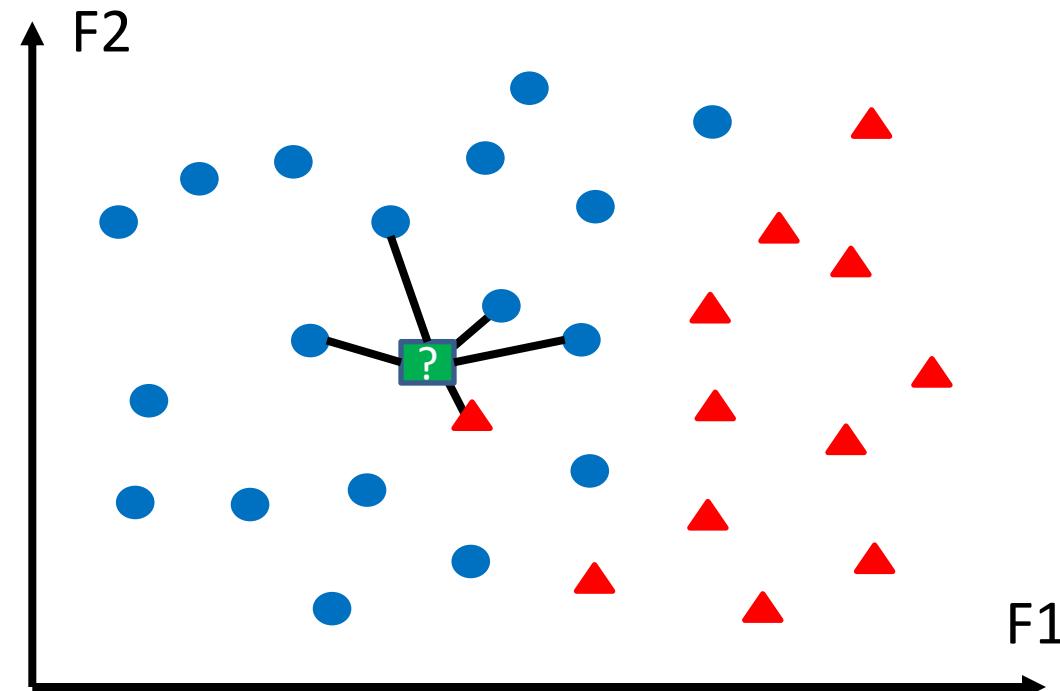
Nearest Neighbor Classification

- 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.



KNN Classification

- K-Nearest Neighbor (KNN) classifier classifies objects based on majority of K closest training samples 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, 4 features and 3 classes.

	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** (input): The vector of all features for **one single** observation: $\mathbf{x} \in \mathbb{R}^D$
 - **Label** (output): Value of label for one observation $y \in \{0,1,2, \dots, C\}$
 - **Training dataset**: $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_N, y_N)\}$
 - **Special case**: binary classification
 - 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:* $\{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$:
 N data samples used for training.

sepal length	sepal width	petal length	petal width	Label	
x_1 → 5.3	3.7	1.5	0.2	setosa	→ y_1
x_2 → 5	3	2	0.2	setosa	→ y_2
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

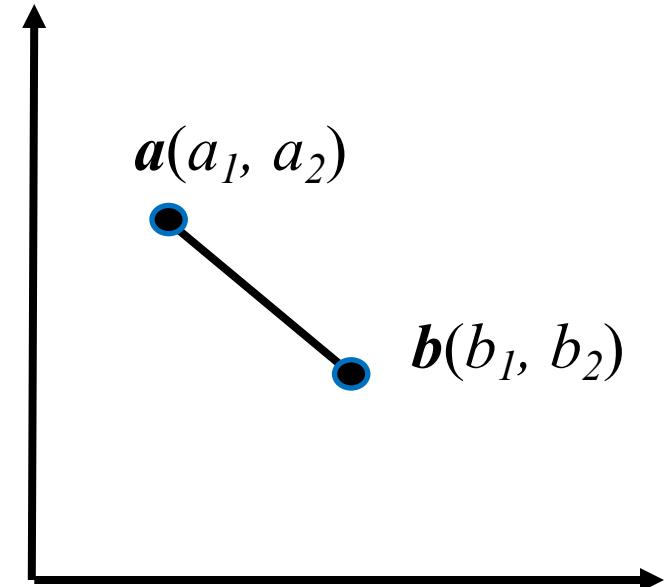
- Training dataset: $\{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$ with known label.
- Now, we have a new sample with unknown label: $(x, y=?)$

	sepal length	sepal width	petal length	petal width	Label	
x_1	5.3	3.7	1.5	0.2	setosa	y_1
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	
x_N	7.9	3.8	6.4	2	virginica	y_N
x	7	3.9	5.9	1.3	???	$y=?$

Euclidean Distance

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

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



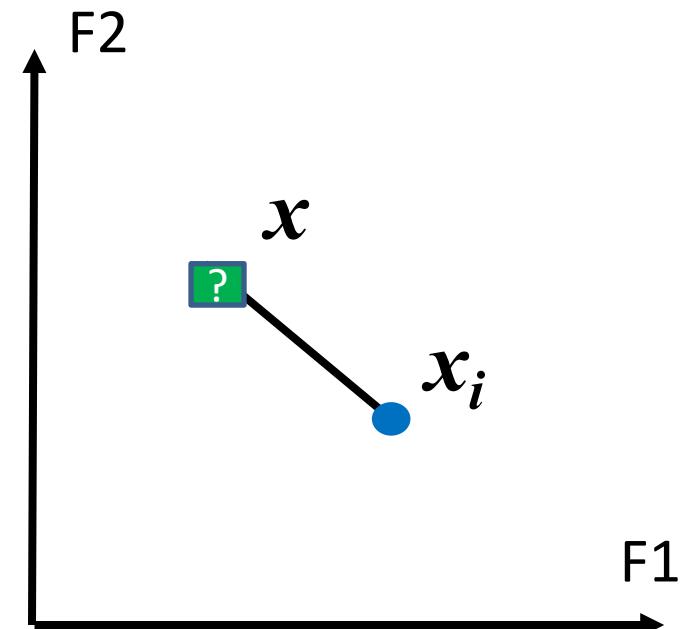
- *Euclidean Distance* for D-dimensional Space:

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

1-Nearest Neighbor Classification

- *Training dataset:* $\{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$
- Suppose we have a new sample with unknown label: $(x, y=?)$
- “1-NN algorithm” classifies new samples based on the closest training samples in the feature space.
- *The Distance* between x and a known training point x_i (Distance between 2 D-dimensional points in D-dimensional space):

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



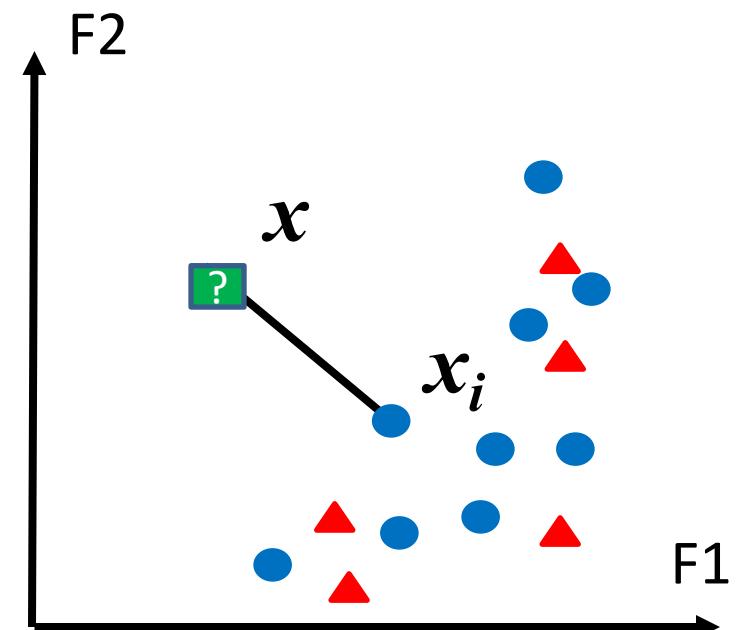
1-Nearest Neighbor Classification

- Training dataset: $\{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$
- Suppose we have a new sample with unknown label: $(x, y=?)$
- **Euclidean Distance** between x and a known training point x_i :

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

- Nearest Neighbor to x :

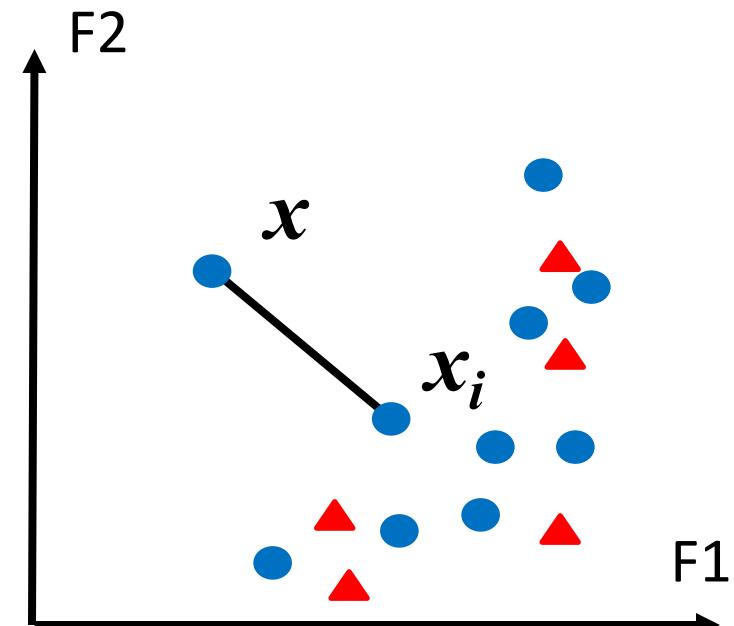
$$\begin{aligned} NN(x) &= \arg \min_{i \in \{1, \dots, N\}} (Dis(x_i, x)) \\ &= \arg \min_{i \in \{1, \dots, N\}} (\|x_i - x\|^2) \\ &= \arg \min_{i \in \{1, \dots, N\}} (\sum_{d=1}^D (x_{id} - x_d)^2) \end{aligned}$$



1-Nearest Neighbor Classification

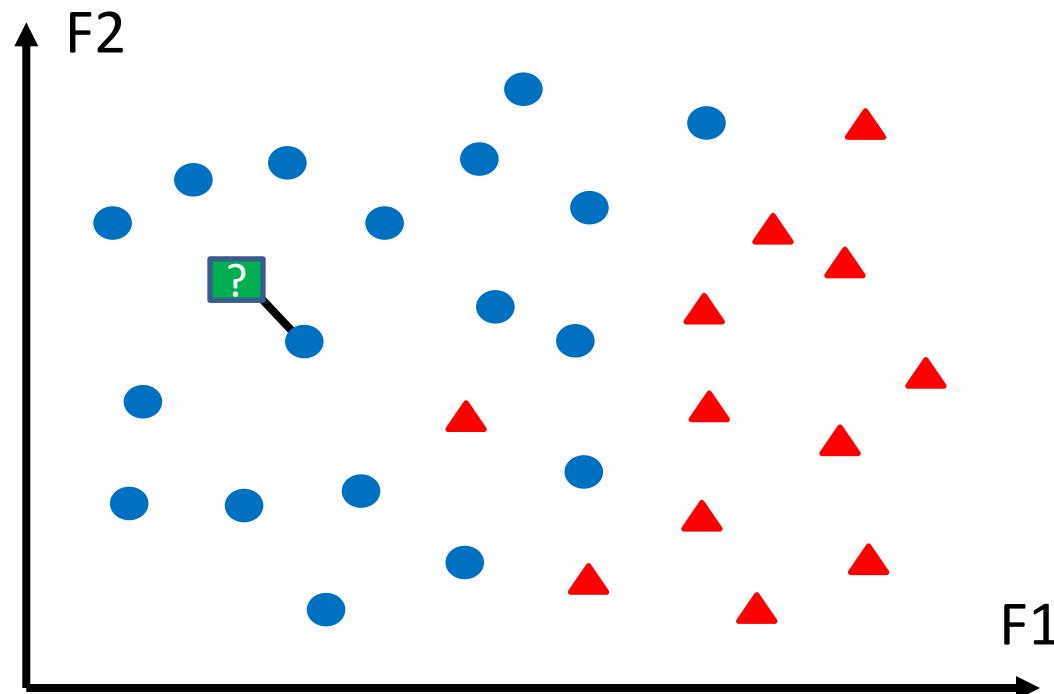
- Nearest Neighbor to x : $NN(x) = \arg \min_{i \in \{1, \dots, N\}} (\|x_i - x\|_2^2)$
- After finding the Nearest Neighbor (NN) of x , the label of x will be determined as the label of its NN:

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



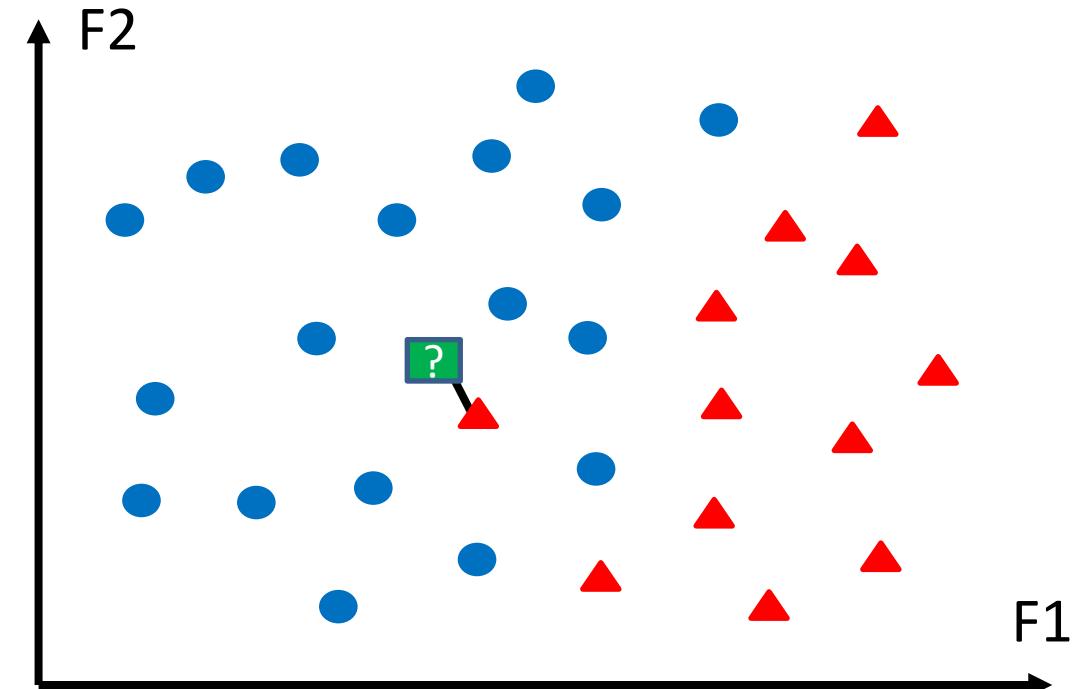
Nearest Neighbor Classification

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



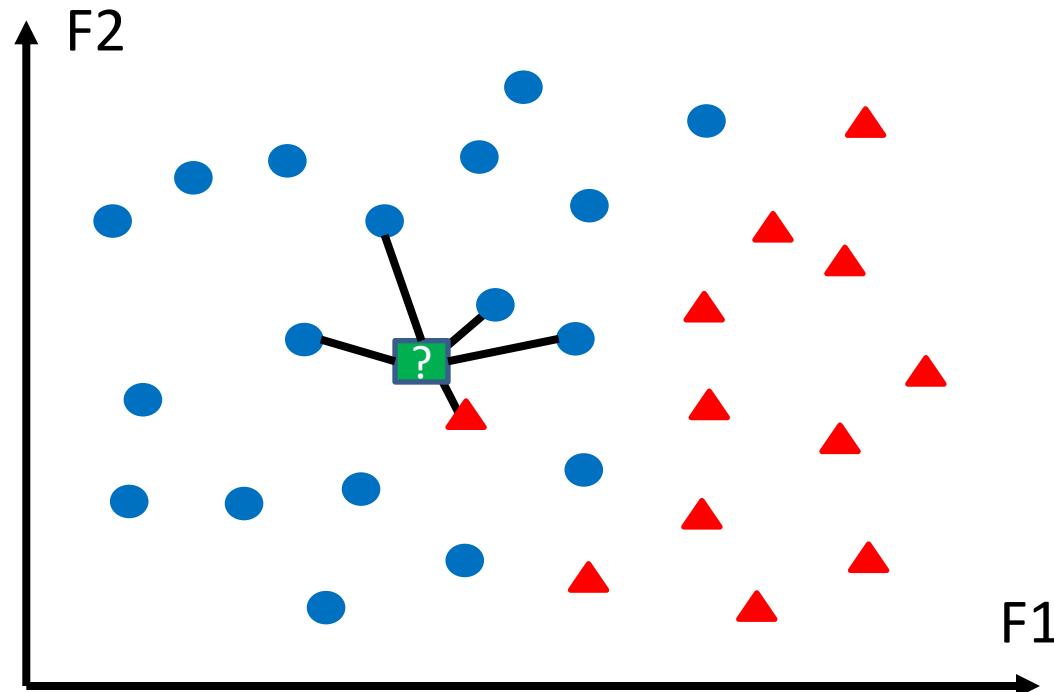
Nearest Neighbor Classification

- 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: $NN_1(\mathbf{x})$
- 2nd-Nearest Neighbor: $NN_2(\mathbf{x})$
- 3rd-Nearest Neighbor: $NN_3(\mathbf{x})$
 :
- Kth-Nearest Neighbor: $NN_K(\mathbf{x})$

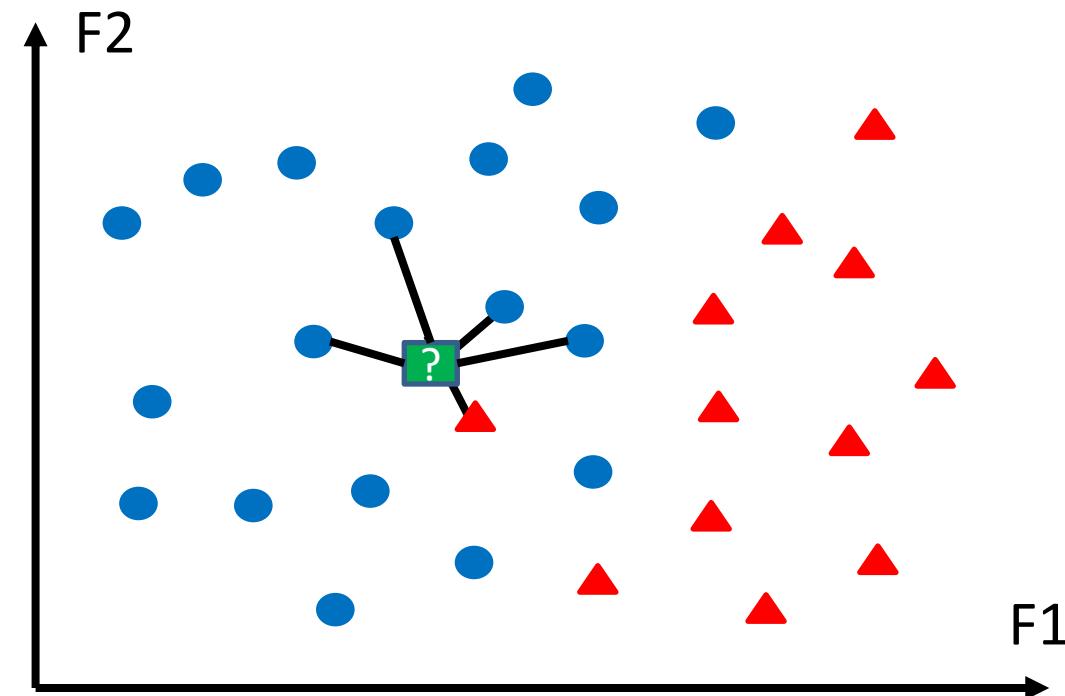
- The set of K-Nearest Neighbors: $KNN(\mathbf{x}) = \{ NN_1(\mathbf{x}), NN_2(\mathbf{x}), \dots, NN_K(\mathbf{x}) \}$
- **Classification rule: Voting**: Select the Label with the majority in $KNN(\mathbf{x})$.



K-Nearest Neighbor Classification

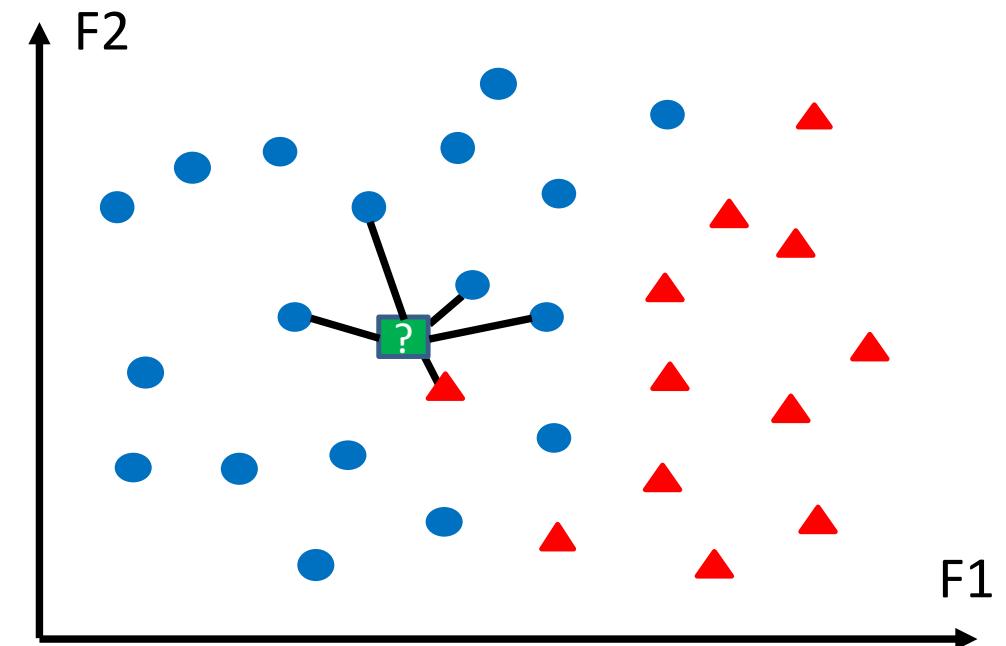
- K-Nearest Neighbor (KNN) classifier algorithm classifies objects based on K closest training samples in the feature space, e.g. K=5.

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



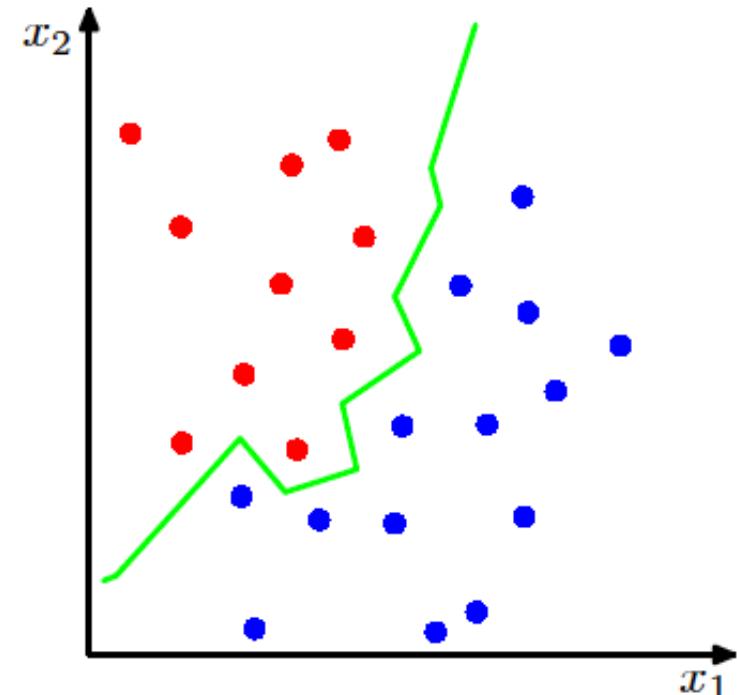
How to Select K?

- **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 decision boundary that partitions the space into different regions.



Advantages and Disadvantages

- **Advantages of using KNN Classifier:**

- Easy and Simple to implement
- Low Computational Complexity

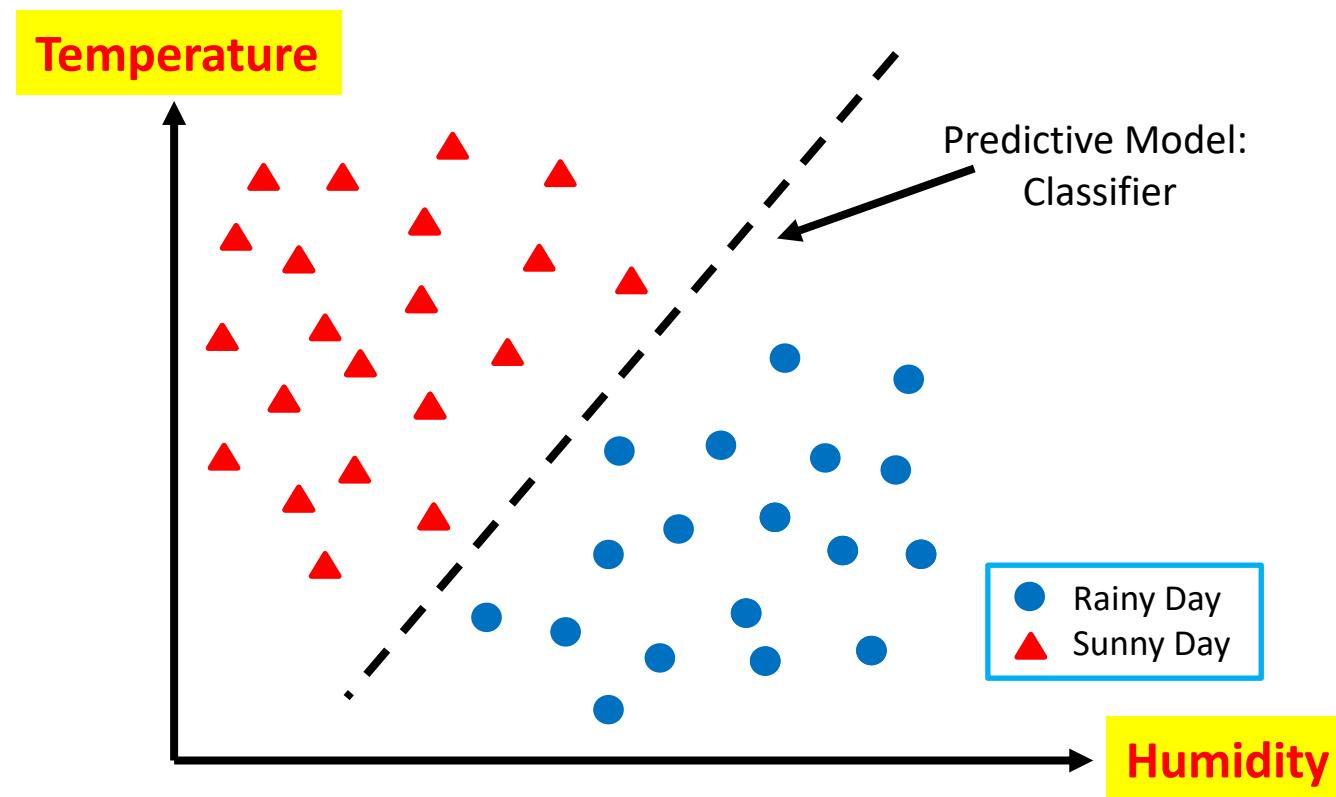
- **Disadvantages:**

- 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.



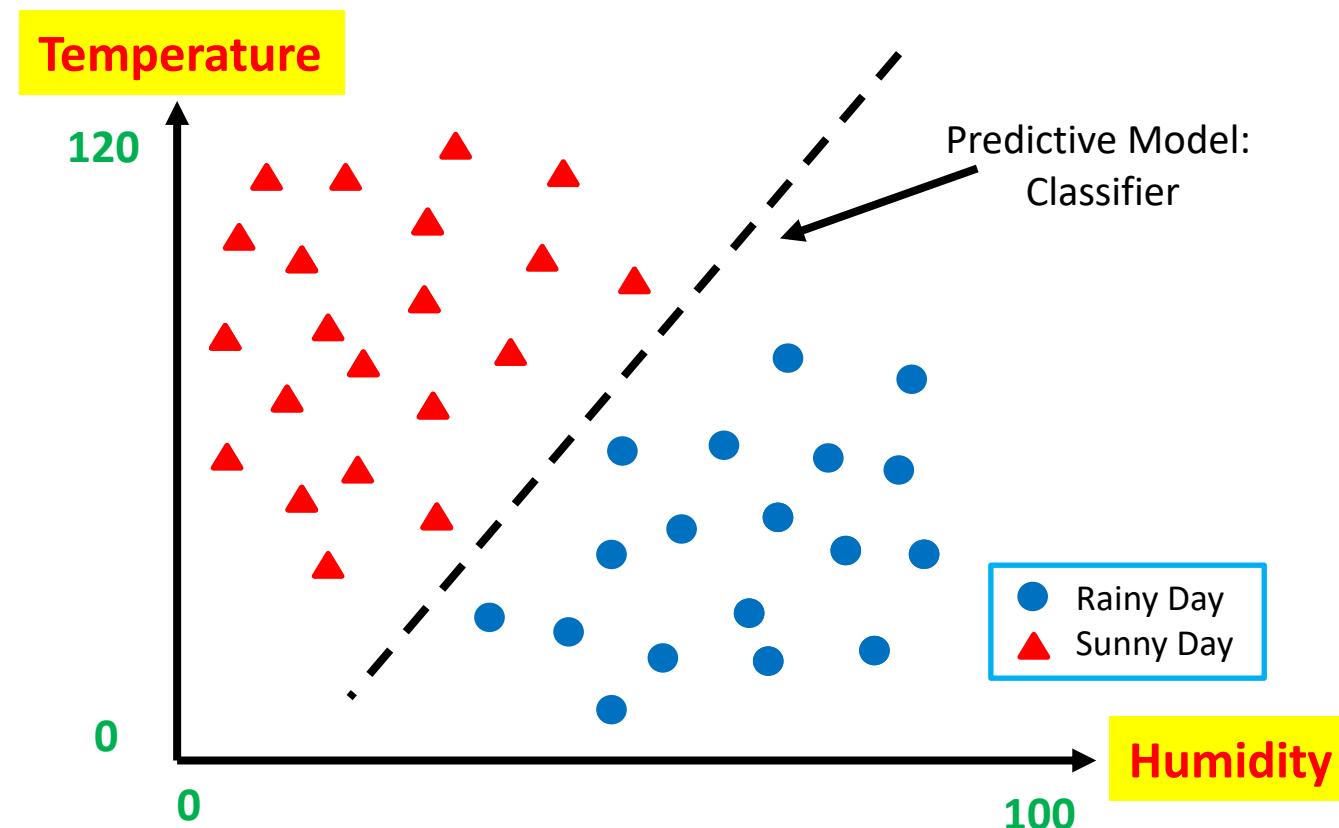
A practical hint about KNN

- 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!



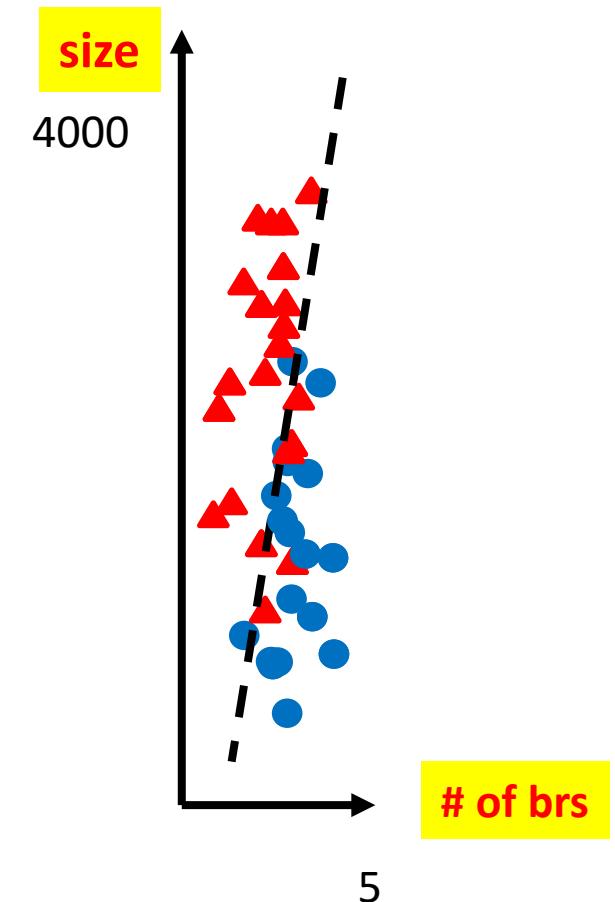
Another Example!

- 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 feet²).

Feature2 = number of bedrooms (1-5).

$$\begin{aligned}NN(x) &= \arg \min_{i \in \{1, \dots, N\}} (\|x_i - x\|_2^2) \\&= \arg \min_{i \in \{1, \dots, N\}} [(\underbrace{x_{i1} - x_1}_{\text{Negligible}})^2 + (\underbrace{x_{i2} - x_2}_{\text{Significant}})^2]\end{aligned}$$



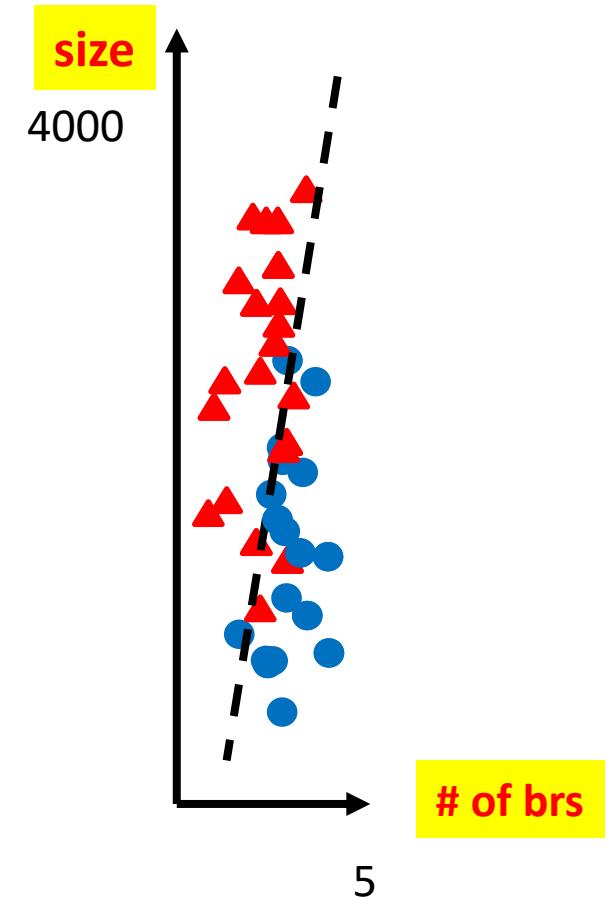
Another Example!

- 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 feet²).

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 feet²).

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 feet²).

Feature2 = number of bedrooms (1-5).

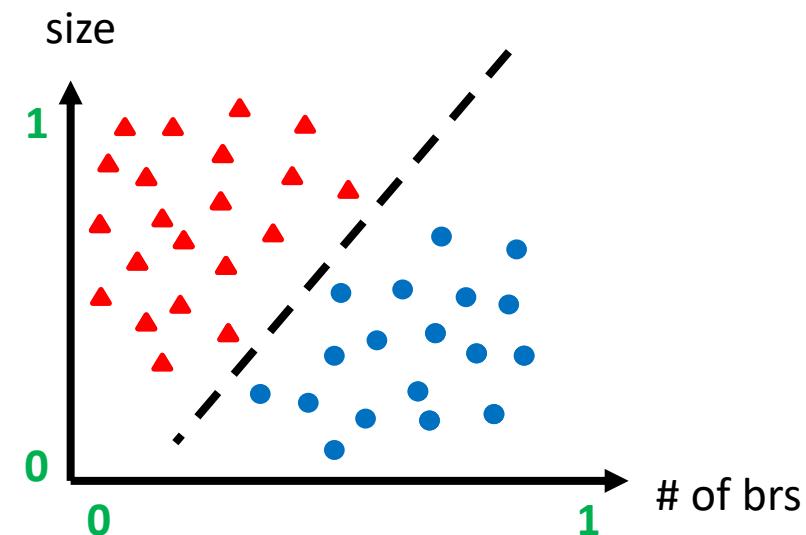
Easy way to scale features:

F1_Scaled = size / 4000

F2_Scaled = number of bedrooms / 5

($0 \leq F1_Scaled \leq 1$)

($0 \leq F2_Scaled \leq 1$)



A more advanced technique for normalization

- We can **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 d th feature of n th 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 d th feature of n th data sample):

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

- Scale the feature accordingly

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





Thank You!

Questions?