Machine Learning

"**Machine learning** is a subset of **artificial intelligence** that uses algorithms to **learn from data** and enables machines to improve with **experience**"

Machine Learning

Formally (by Tom Mitchell, 1997):

Study of algorithms that:

- improve their **<u>performance</u>** P
- at some **<u>task</u>** T
- with **<u>experience</u>** E

Well-defined learning task: <P,T,E>

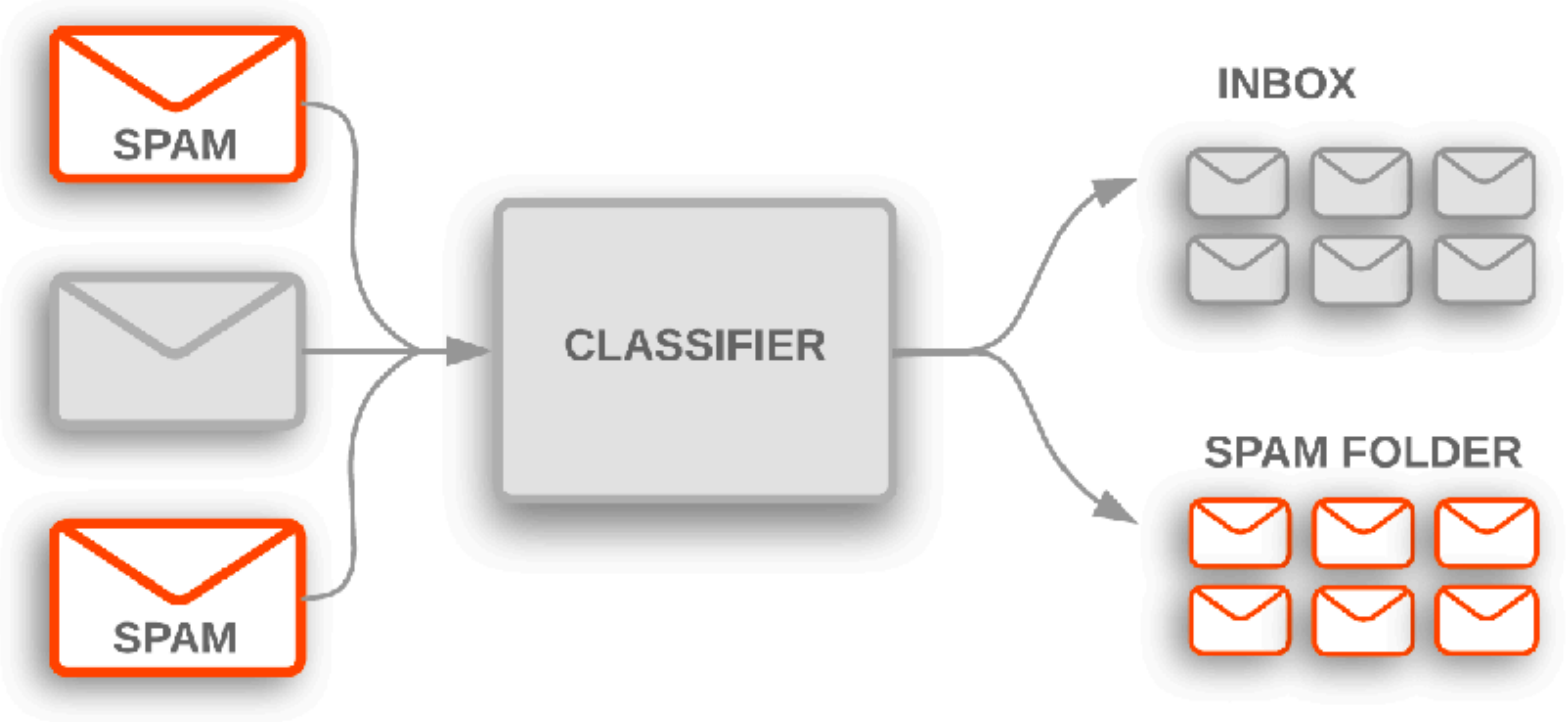
Machine Learning

Formally (by Tom Mitchell, 1997):

Study of algorithms that:

- improve their **performance** P
- at some **task** T
- with **experience** E

Well-defined learning task: <P,T,E>



Example of Machine Learning

Task: Predict the sale price of a house

Experience: Dataset with samples and one feature (square meters)

Performance: RMSE: $\frac{1}{n}\sum_{i}^{n}(y_i - \hat{y}_i)^2$

Sq. meters	Sale Price
50	250.000
75	389.000
72	340.000
60	295.000
95	440.000
55	240.000
120	800.000
87	570.000

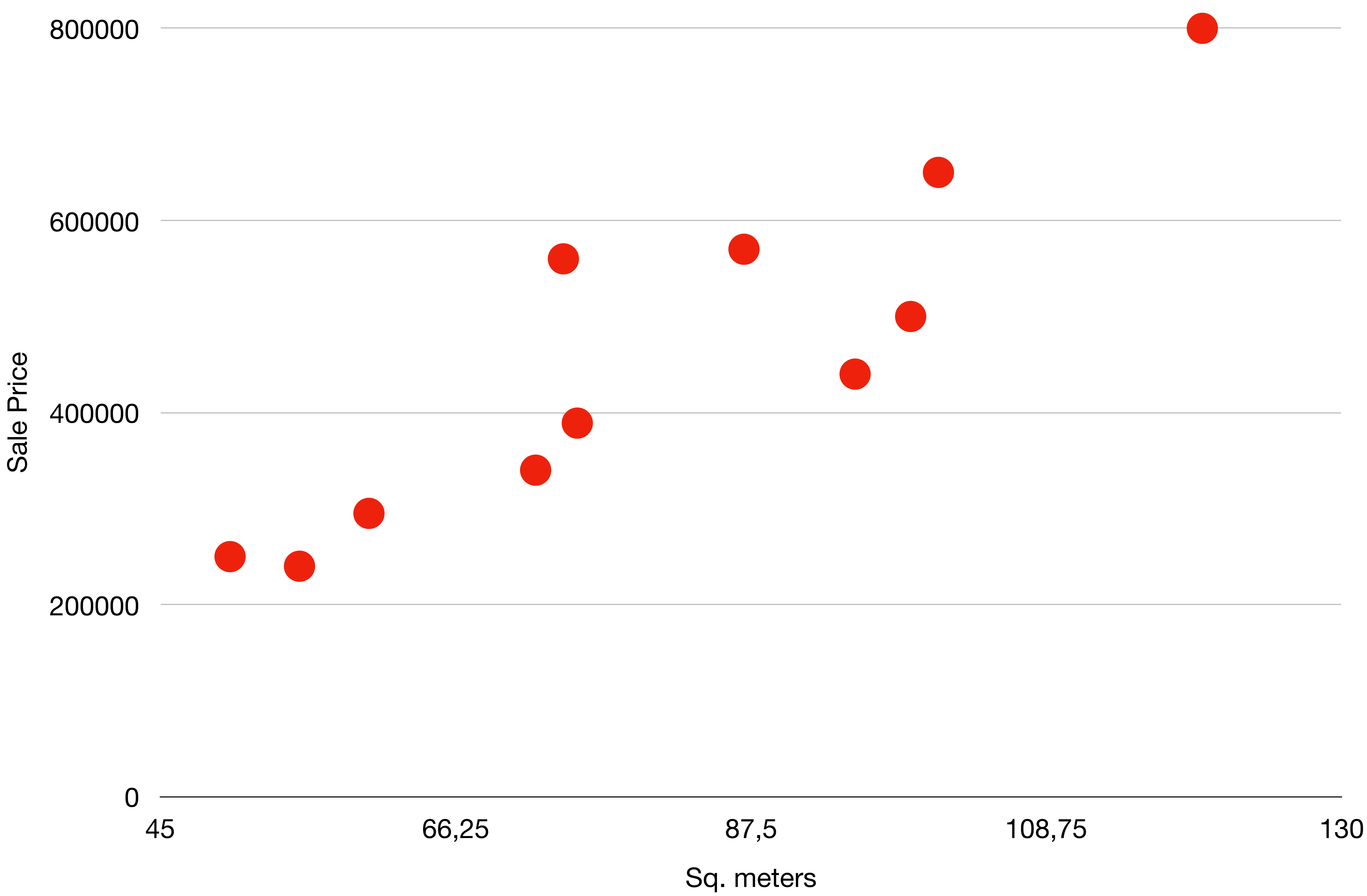
Example of Machine Learning

Now is time to **define HOW**!!! A model must be defined. It always consist of two main parts:

- **Inference** function (or regression/classification function
- **Cost** function
    - And an optimization method to minimize this cost function

Sq. meters	Sale Price
50	250.000
75	389.000
72	340.000
60	295.000
95	440.000
55	240.000
120	800.000
87	570.000

Let’s solve it with the most simple methods: Linear Regression (OLS)

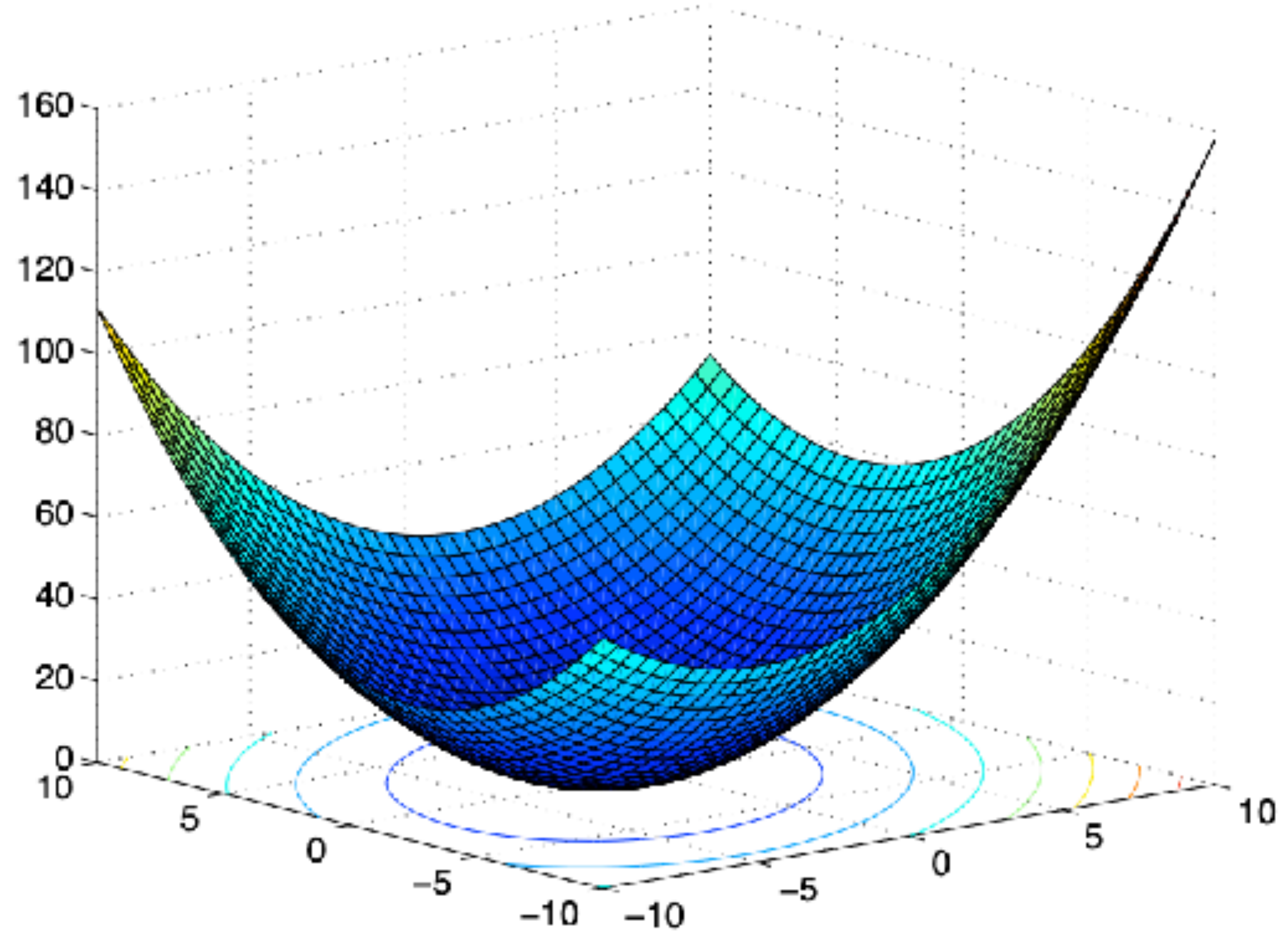


Example of Machine Learning

- We will have to define a cost function, as for instance:

$$cost = \frac{\sum_i^N (y_i - \hat{y}_i)^2}{N}$$

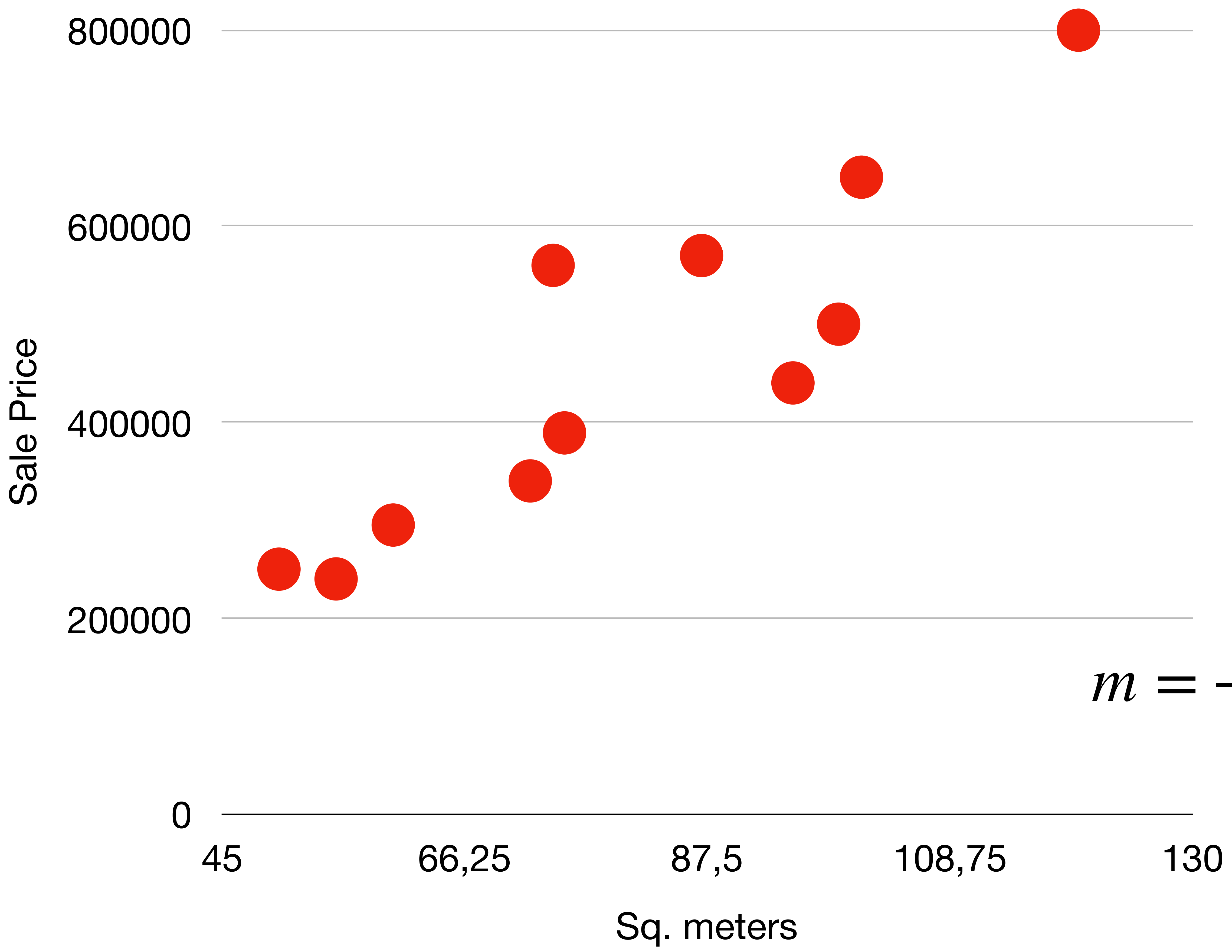
and minimize it using the training data



Example of Machine Learning

Sq. meters	Sale Price
50	250.000
75	389.000
72	340.000
60	295.000
95	440.000
55	240.000
120	800.000
87	570.000

Task: Predict sale price

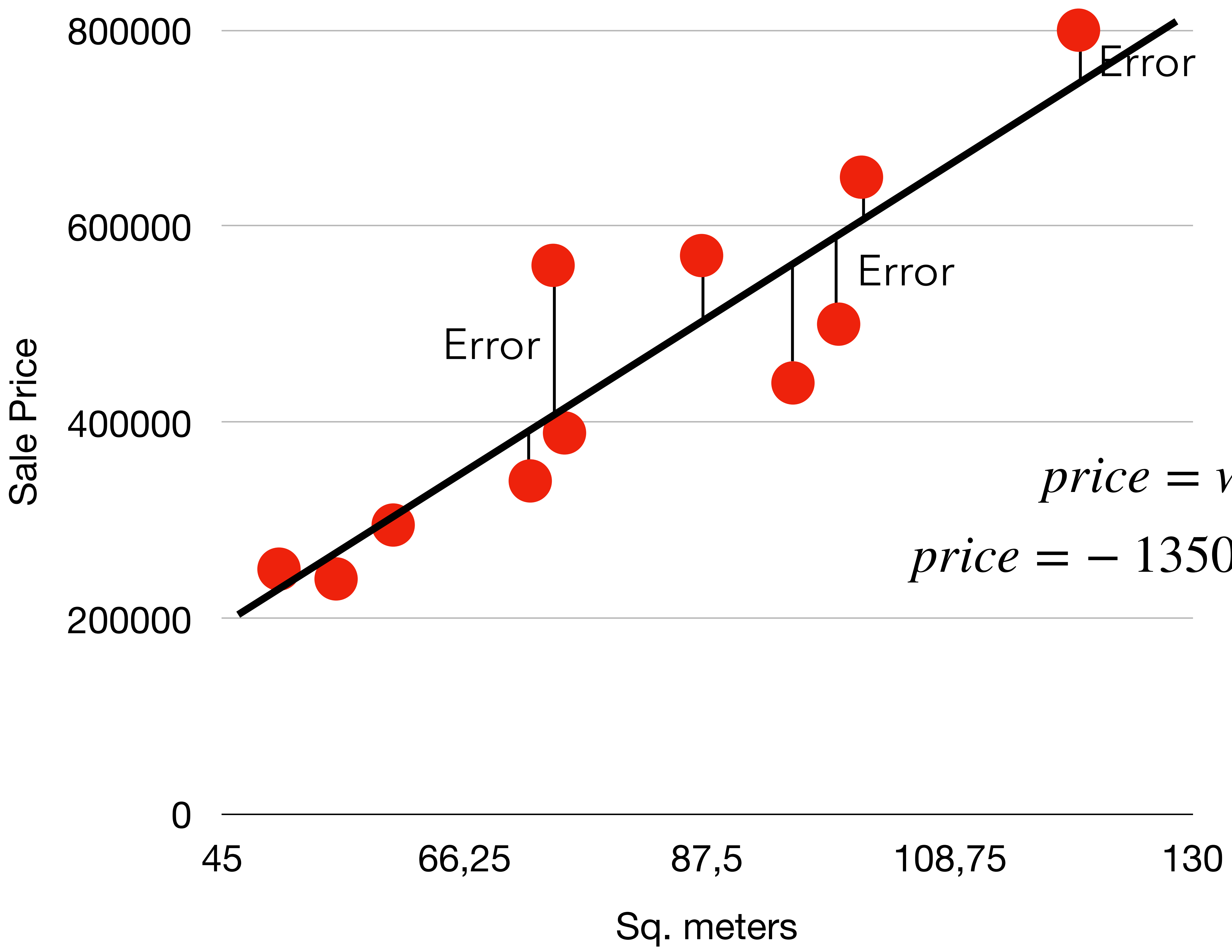


$$m = \frac{\sum_i^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_i^n x_i - \bar{x}}$$

Example of Machine Learning

Sq. meters	Sale Price
50	250.000
75	389.000
72	340.000
60	295.000
95	440.000
55	240.000
120	800.000
87	570.000

Goal: Minimize error

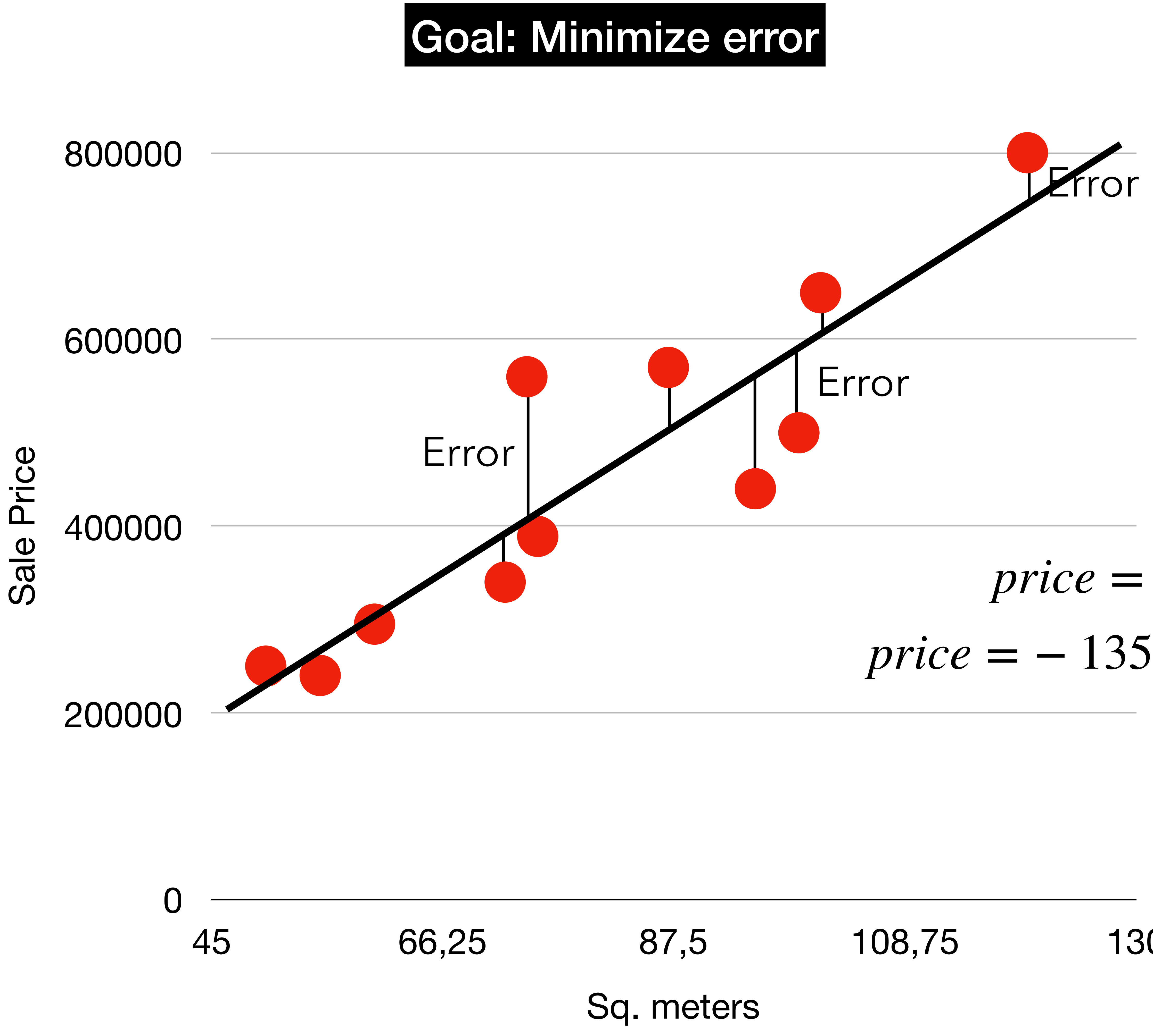


$$price = w_0 + w_1 * meters$$

$$price = -135035 + 7341 * meters$$

Example of Machine Learning

Sq. meters	Sale Price	Prediction
50	250.000	232.015
75	389.000	415.540
72	340.000	393.517
60	295.000	305.425
95	440.000	562.360
55	240.000	268.720
120	800.000	745.885
87	570.000	503.632



$$price = w_0 + w_1 * meters$$

$$price = -135035 + 7341 * meters$$

How can we improve it?

Example of Machine Learning

Sq. meters	#rooms	#baths	district	Sales in Price
50	1	1	Ciutat Vella	250.000
75	3	1	Eixample	389.000
72	2	1	Eixample	340.000
60	2	1	Sants	295.000
95	3	2	Sants	440.000
55	1	1	Eixample	240.000
120	4	2	Sarria	800.000
87	3	2	Gracia	570.000

How do we define our model?

How can we improve it?

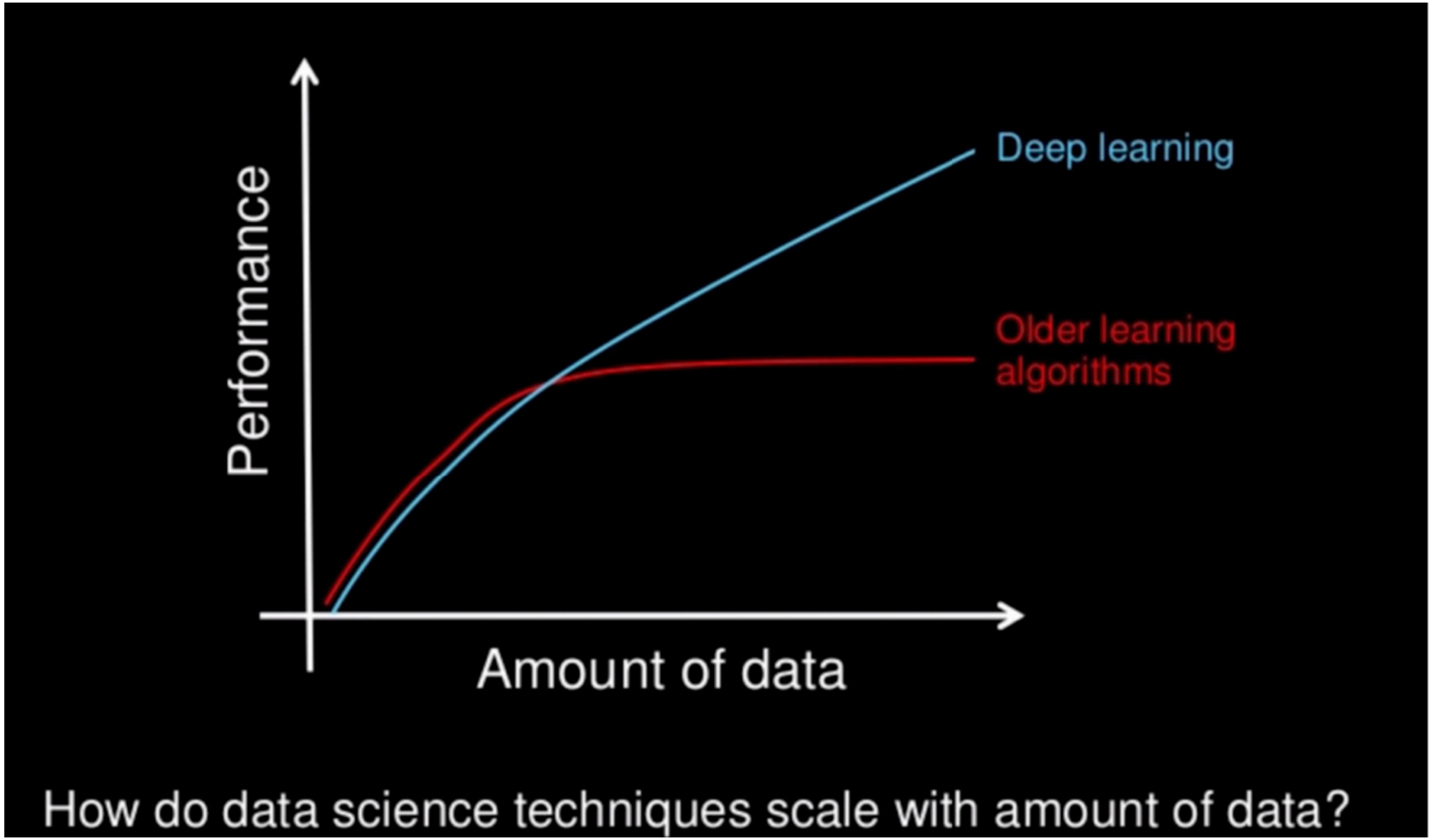
Example of Machine Learning

Sq. meters	#rooms	#baths	district	Address?	Sales in Price
50	1	1	Ciutat Vella	La Rambla, 55	250.000
75	3	1	Eixample		389.000
72	2	1	Eixample		340.000
60	2	1	Sants		295.000
95	3	2	Sants		440.000
55	1	1	Eixample		240.000
120	4	2	Sarria		800.000
87	3	2	Gracia		570.000

How do we define our model?

How can we improve it?

with better models
what about DEEP LEARNING?



No Free Lunch Theorem

Our model is a simplification of reality

⇩

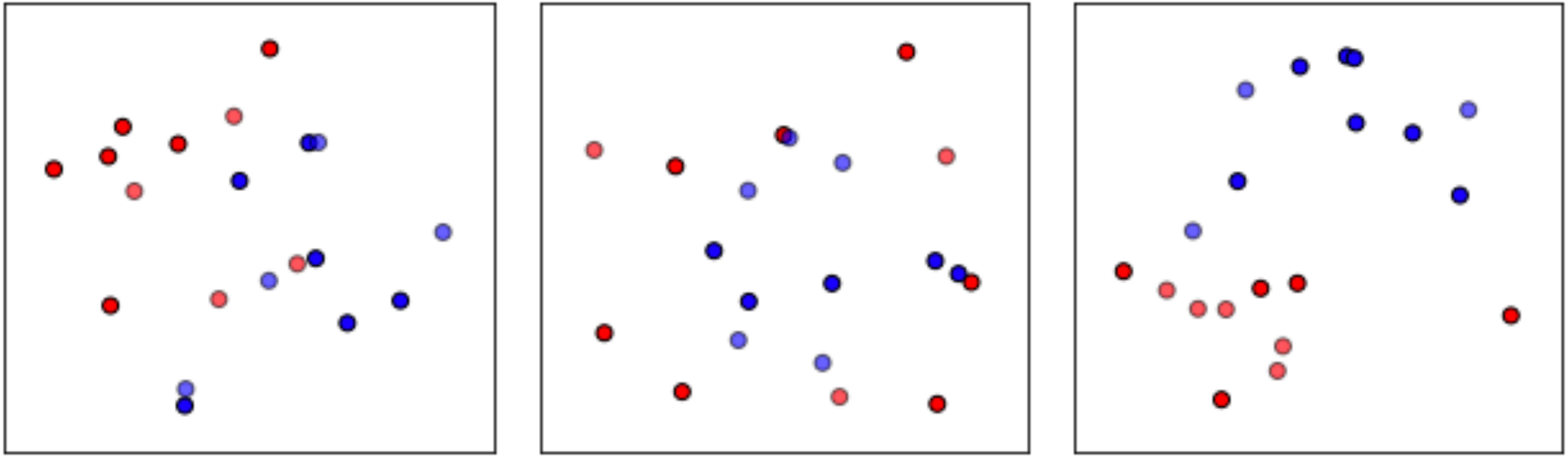
Simplification is based on assumptions (bias)

⇩

Assumptions fail in certain situations

Roughly speaking: “**There is not a model that works best for all possible situations**.”

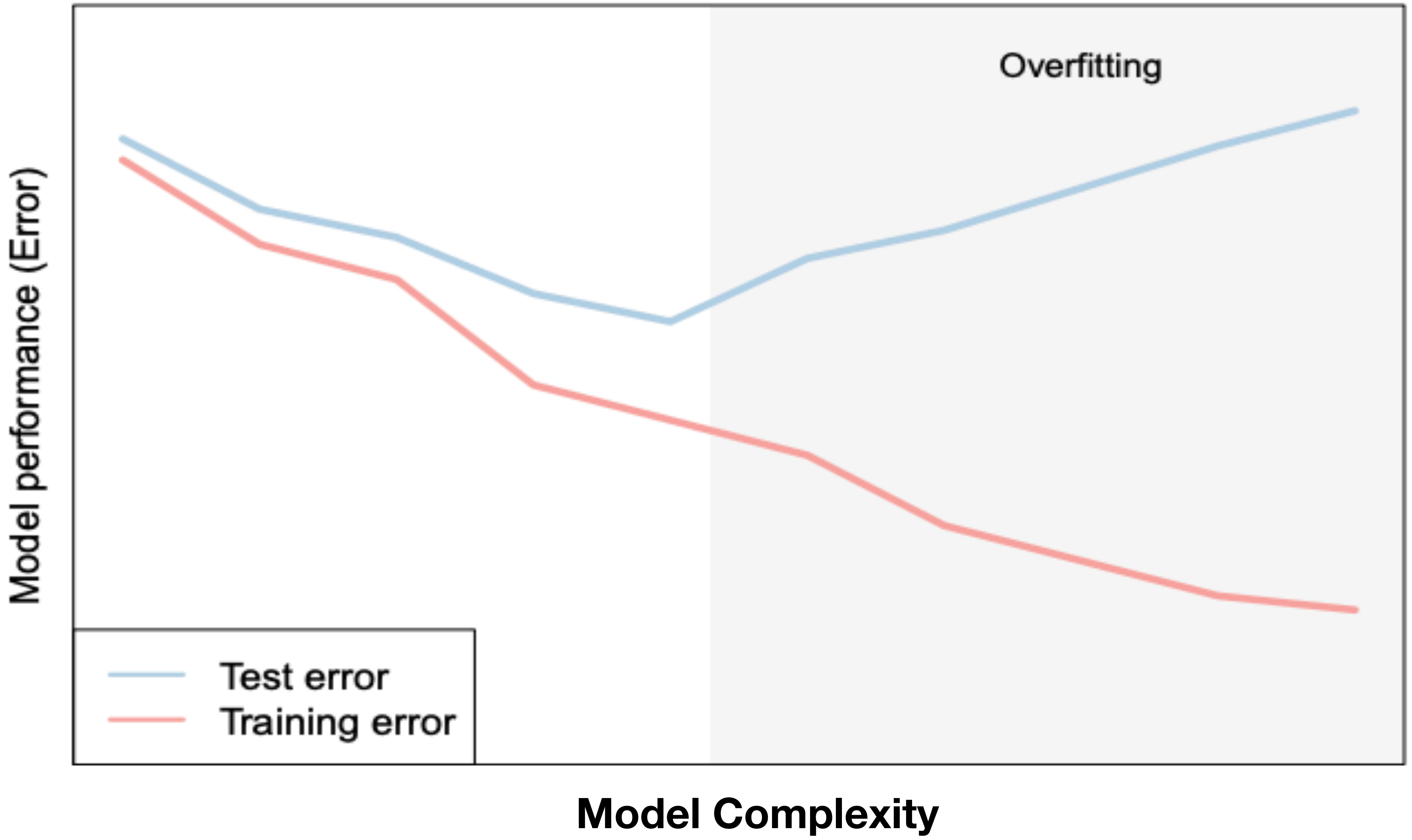
Classification Methods

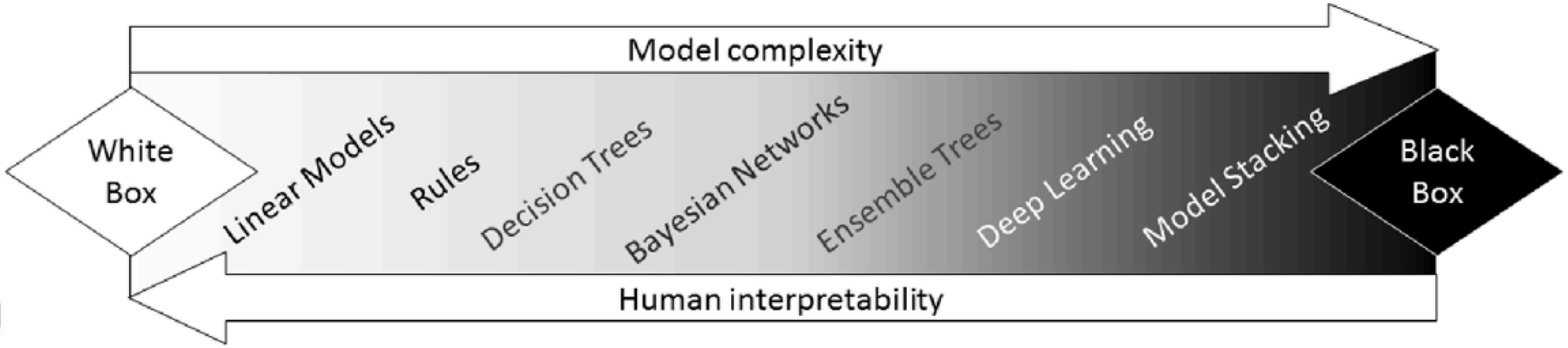


Which model to use??

Classification Methods

Evaluation


Overfitting
Model performance (Error)
Test error
Training error
Model Complexity


Model complexity
White
Box
Linear Models
Rules
Decision Trees
Bayesian Networks
Ensemble Trees
Deep Learning
Model Stacking
Black
Box
Human interpretability

Supervised Learning

Supervised Learning

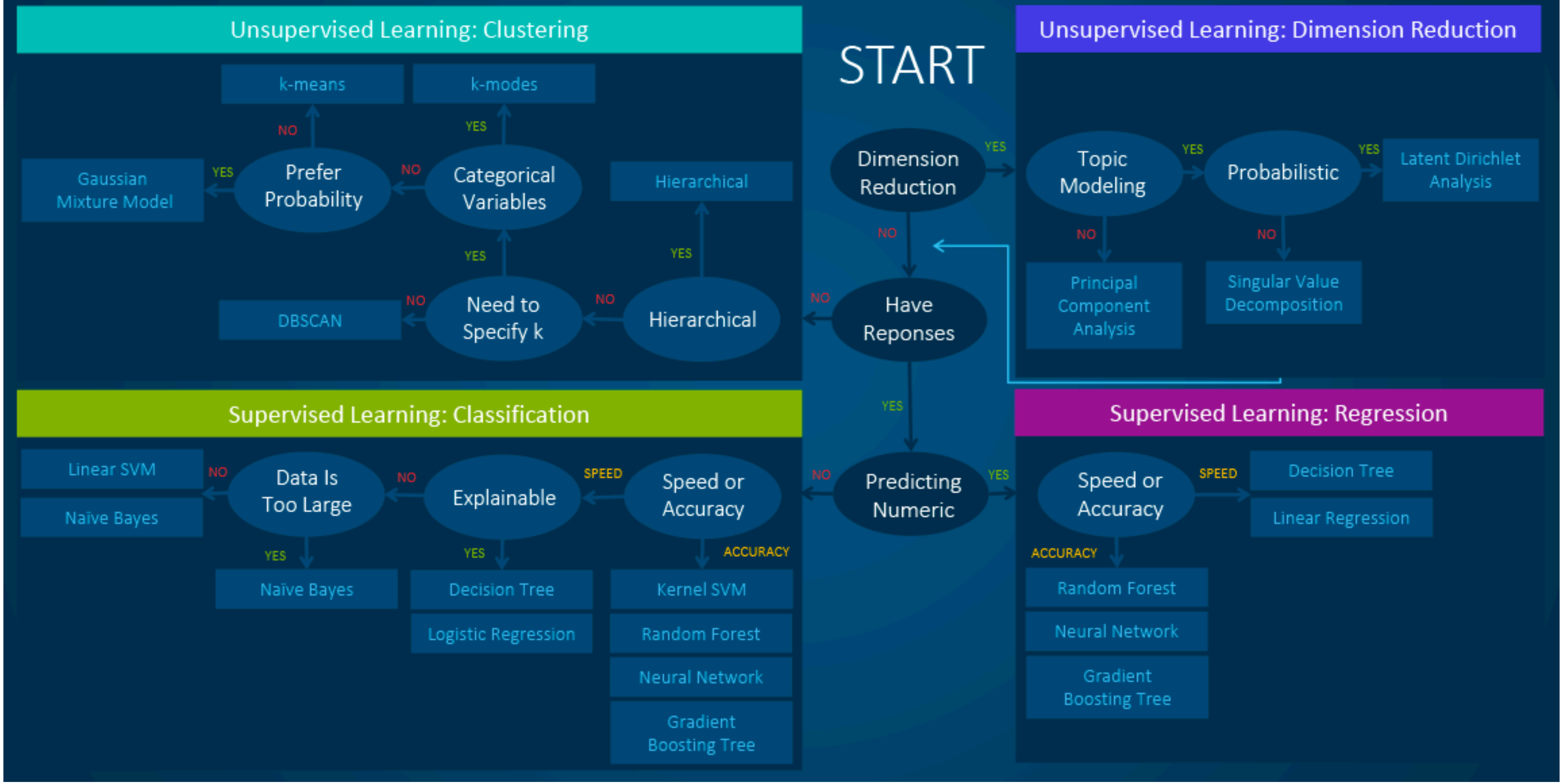
Regression →

Linear Regression	Bayesian Regression	Support Vector Regressor	Decision Trees
Lasso, Ridge & Elastic Net	Polynomial Regression	Random Forest	....
KNN Regression	Kernel Ridge Regression	Neural Networks	....

Classification →

KNN	Naive Bayes Classifier	Linear SVM's	GBM
Logistic Regression	Decision Tree	Kernel SVM	...
Random Forest	Boosting	Neural Network	...

Machine Learning Algorithms Cheat Sheet



EXERCICE

- Consider a simple classifier applied to some two-class data:
  - Starting with 5.000 features and 50 samples, find the 100 features having the largest correlation with the class label.
  - Train a classifier, such as logistic regression, using only those 100 features.

EXERCICE

- Consider a simple classifier applied to some two-class data:
  - Starting with 5.000 features and 50 samples, find the 100 features having the largest correlation with the class label.
  - Train a classifier, such as logistic regression, using only those 100 features.

Can we apply cross-validation in step 2, forgetting about step 1?

NO

This would ignore the fact that in Step 1, the procedure has already seen the labels of the training data, and made use of them. This is a form of training and must be included in the validation process.

It is easy to simulate realistic data with the class labels independent of the outcome, so that true test error =50 %, but the CV error estimate that ignores Step 1 is zero!

Example: South African Heart Disease

- **160 cases of MI** (myocardial infarction) and **302 controls** (all male in age range 15-64), from Western Cape, South Africa in early 80s
- Overall **prevalence** very high in this region: **5.1%.**
- Goal is to identify relative strengths and directions of risk factors as well as predict future cases.

Unbalaced data

- In South African data, there are **160 cases**, **302 controls** - $\tilde{\pi} = 0.35$ are cases. Yet the prevalence of MI in this region is $\pi = 0.05$.
- With case-control samples, we can estimate the regression parameters β_j accurately (if our model is correct); the constant term β_0 is incorrect.
- We can correct the estimated intercept by a simple transformation

$$\hat{\beta}_0^* = \hat{\beta}_0 + log\frac{\pi}{1-\pi} - log\frac{\tilde{\pi}}{1-\tilde{\pi}}$$

- Often cases are rare and we take them all; up to five times that number of controls is sufficient.

Classification Methods

Logistic Regression

SVM

K-NN: Distance based

Tree Based Method

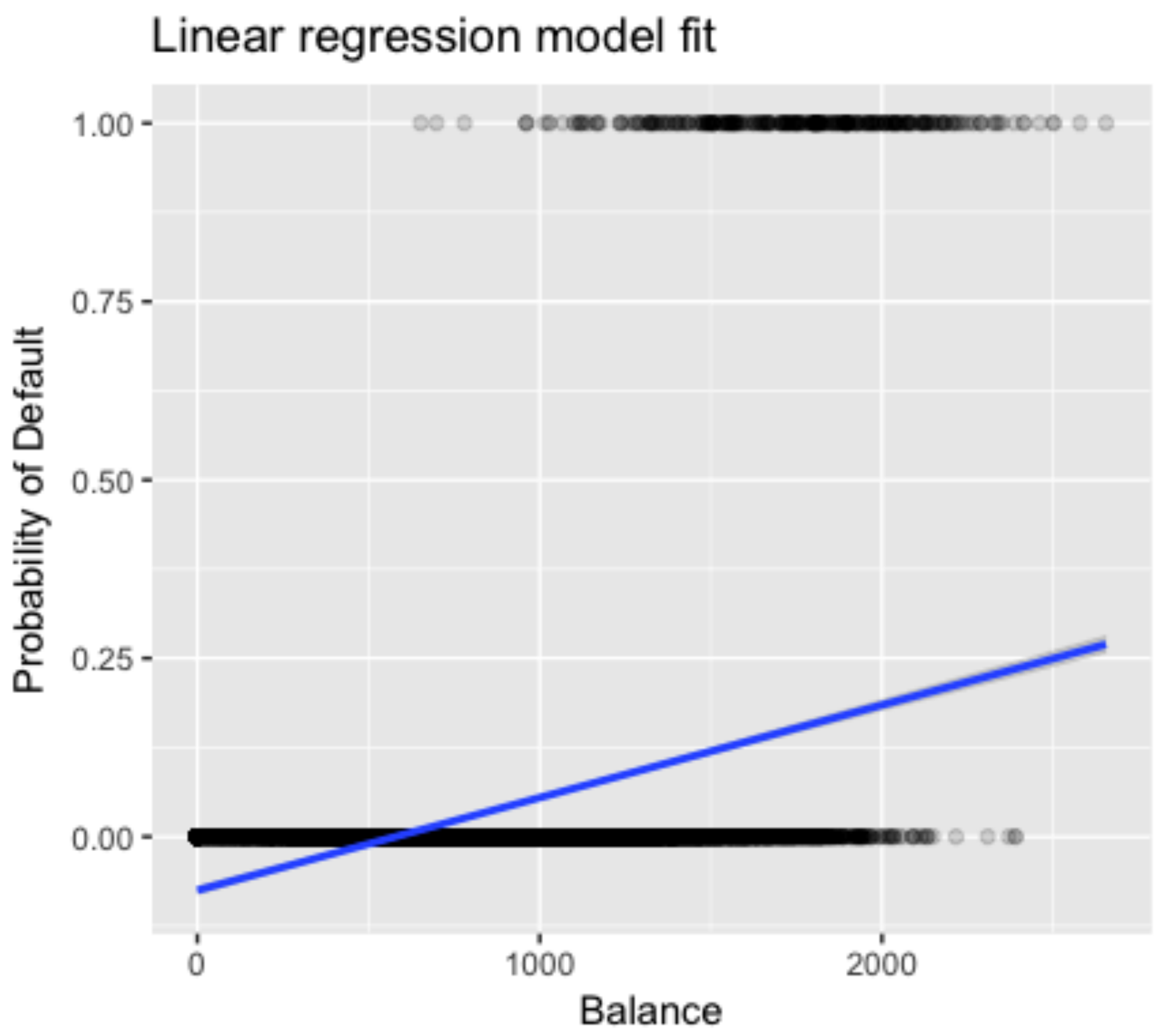
Neural Networks

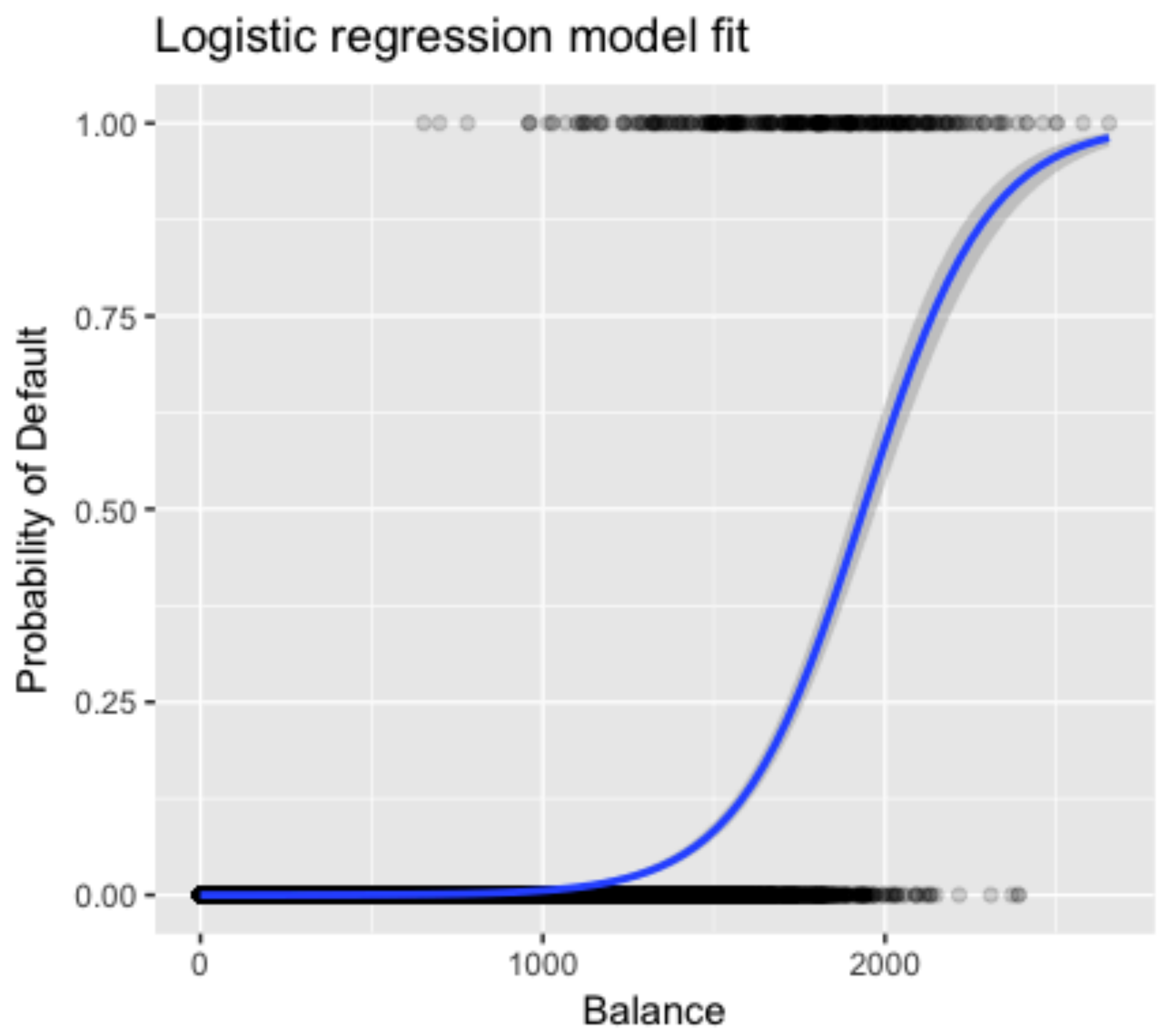
Logistic Regression

Logistic Regression

- **Logistic Regression**, despite the "regression" term in its name, **is a classification model** used in problems where the dependent (target) variable has two possible outcomes

Logistic Regression





we must model *p(X)* using a function that gives outputs between 0 and 1 for all value

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}}$$

Logistic Regression

$$\ell(\theta) = \prod_{i:y_i=1} p(x_i) \prod_{i':y_i'=0} (1 - p(x_i'))$$

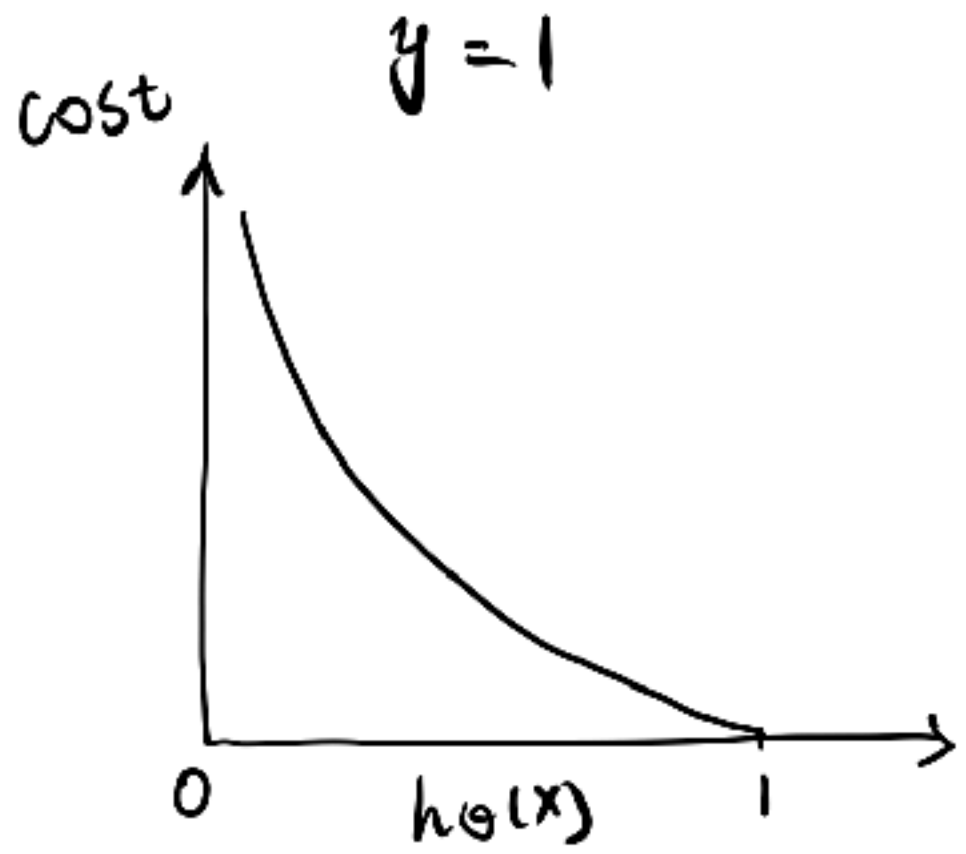
$$Cost(h_\theta(x), y) = \begin{cases} -log(h_\theta(x)) & \text{if } y = 1 \\ -log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

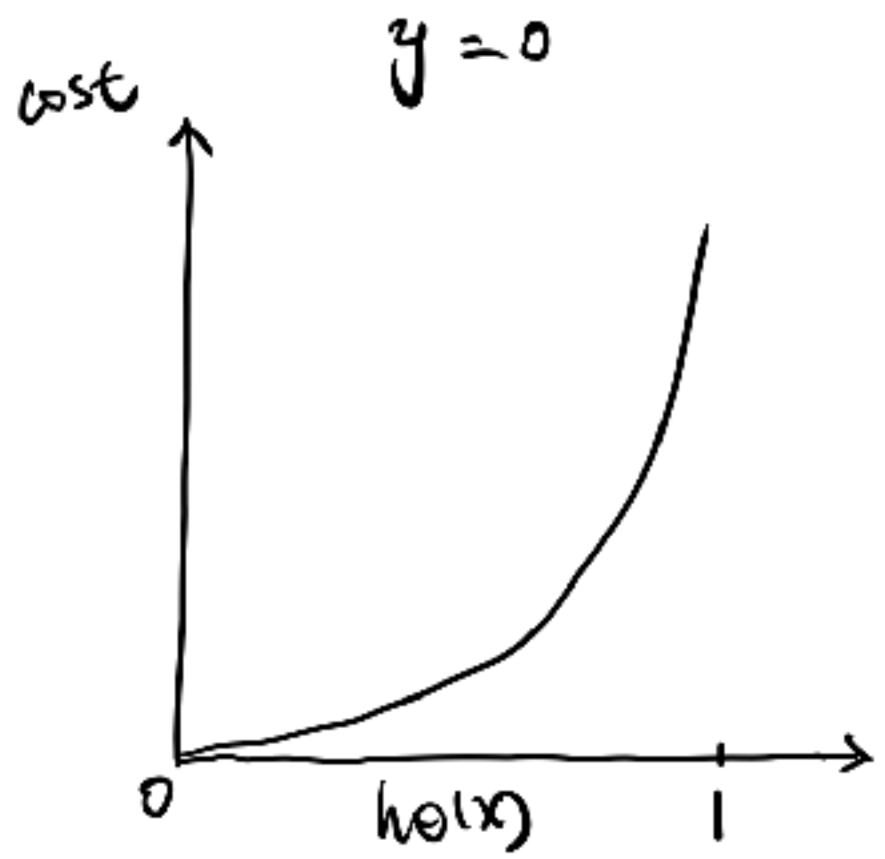
$$Cost(h_\theta(x), y) = -ylog(h_\theta(x)) - (1-y)log(h_\theta(x))$$

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} Cost(h_\theta(x^{(i)}), y^{(i)})$$

$$J(\theta) = \frac{1}{m}[\sum_{i=1}^{m} -y^{(i)} log(h_\theta(x^{(i)})) + (1-y^{(i)})log(1 - h_\theta(x^{(i)}))]$$

$$m = number\ of\ samples$$

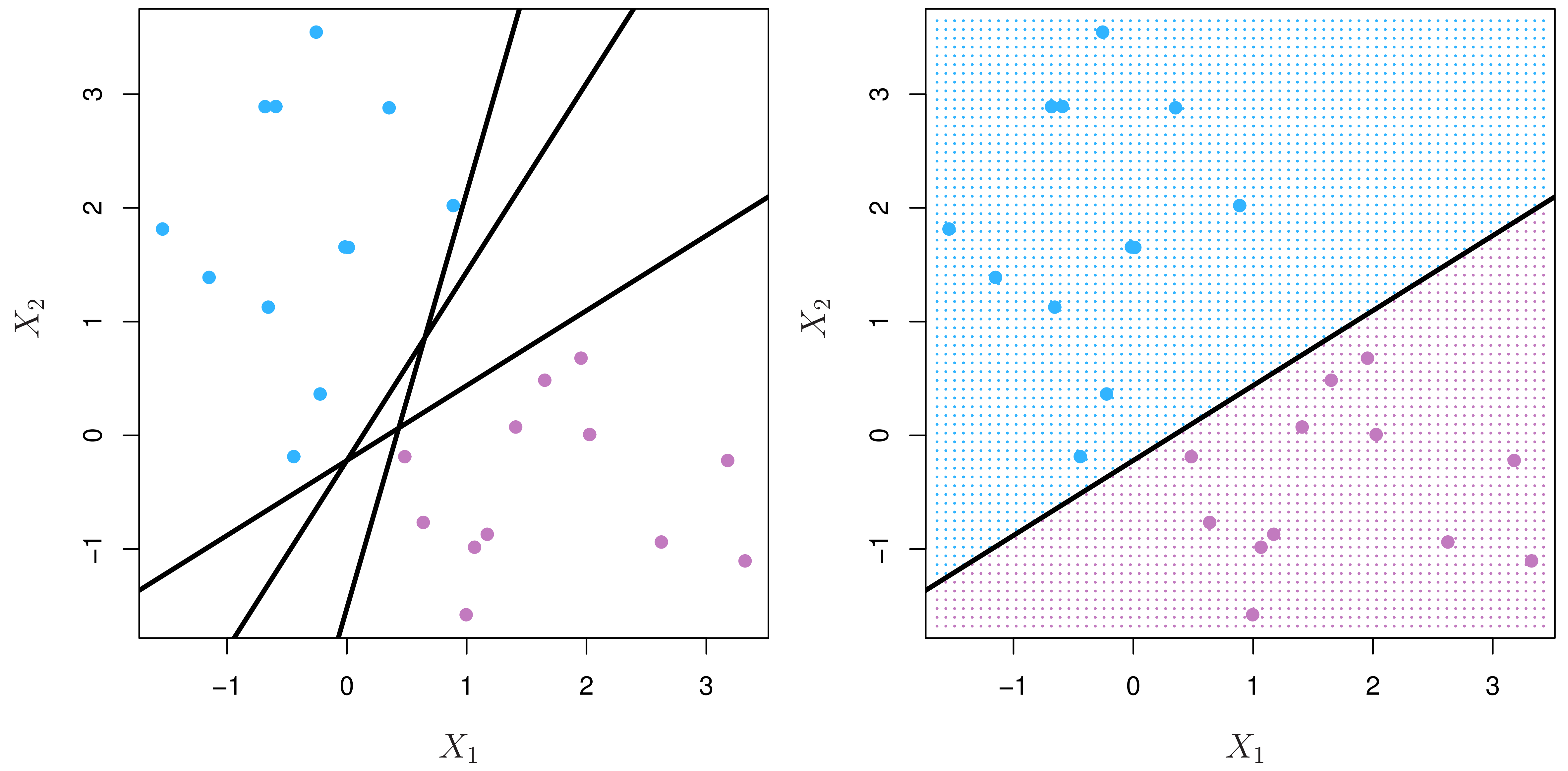




Support Vector Machines

Support Vector Machines

- Find an hyperplane that separates the classes in the features space.

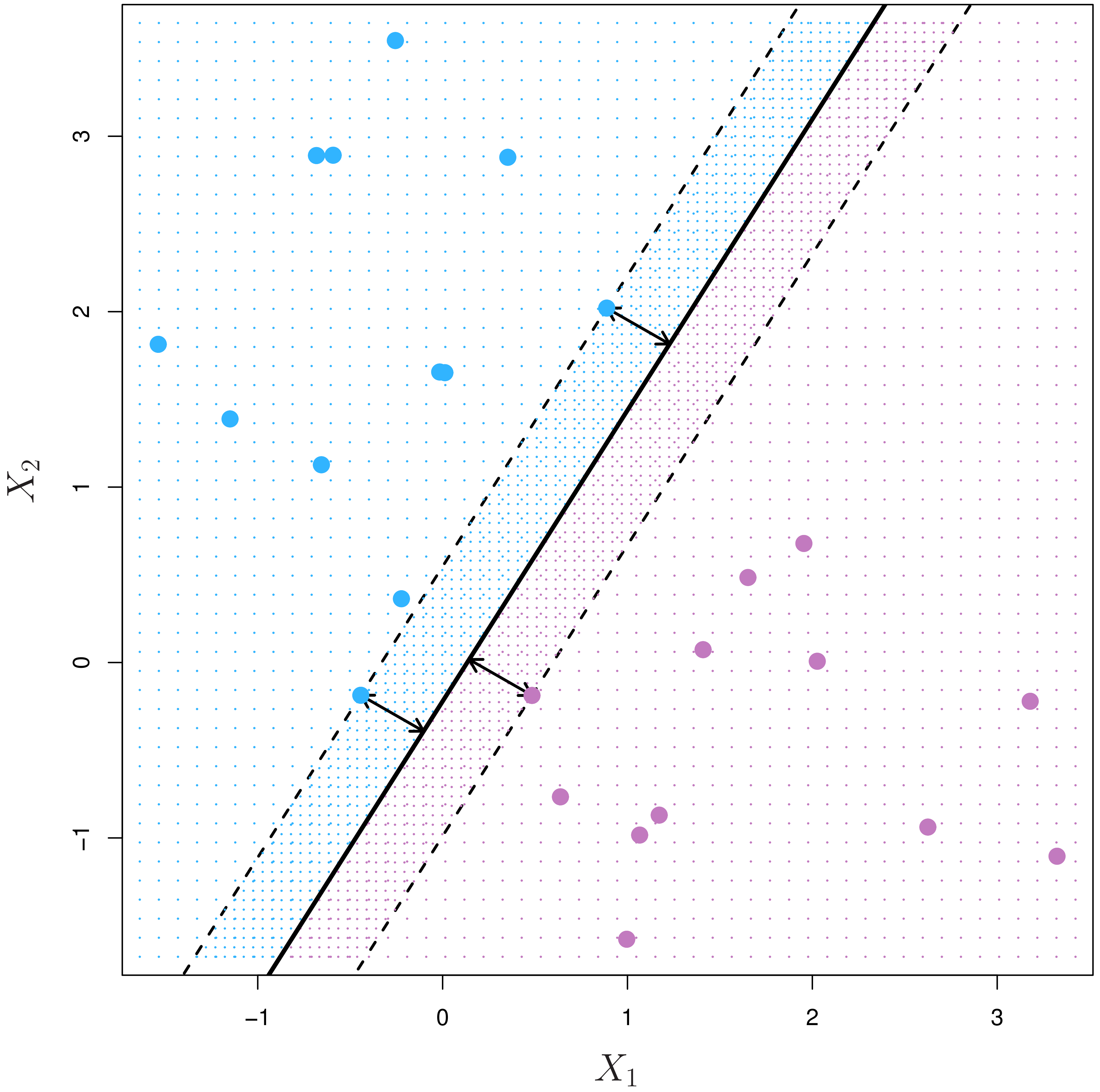


- If $f(x) = B_0 + B_1x_1 + \cdots + B_px_p$, then $f(x) > 0$ for points on one side of the hyperplane, and $f(x) < 0$ for point on the other .

- $f(x) = 0$ defines the separating hyperplane.

Support Vector Machines

- Among all separating hyperplanes, find the one that makes the biggest gap or margin between the two classes.



Constrained optimization problem

$$\text{maximize}_{\beta_0,\ldots,\beta_p} M$$

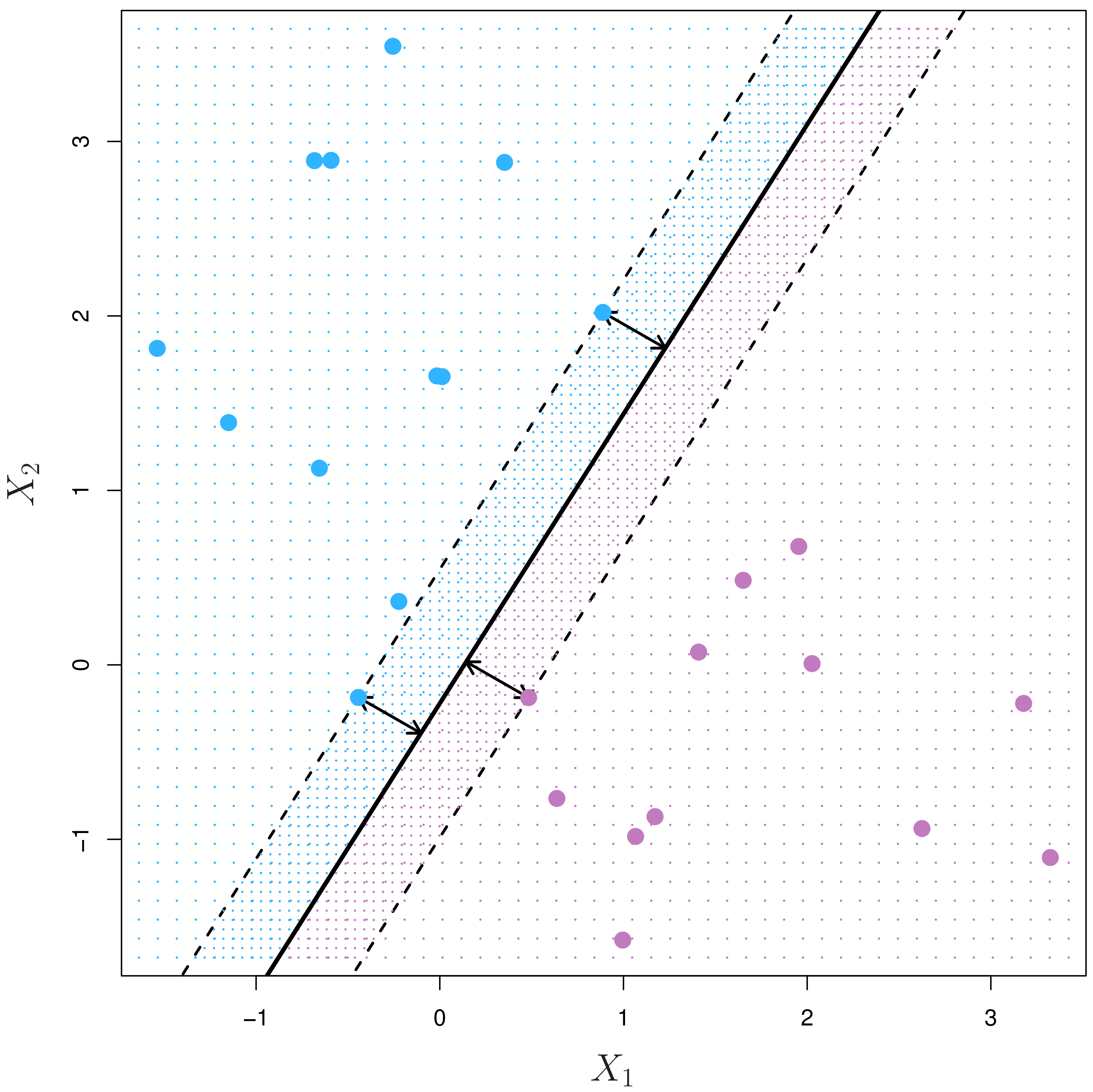
$$\text{subject to} \sum_{j=1}^{p} \beta_j^2 = 1,$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip}) \geq M$$

$$\text{for all} \quad i = 1, \ldots, N$$

Support Vector Machines

- Among all separating hyperplanes, find the one that makes the biggest gap or margin between the two classes.



The previous formulation is equivalent to:

$$minimize_{\beta_0,\ldots,\beta_p} \|\beta\|^2$$

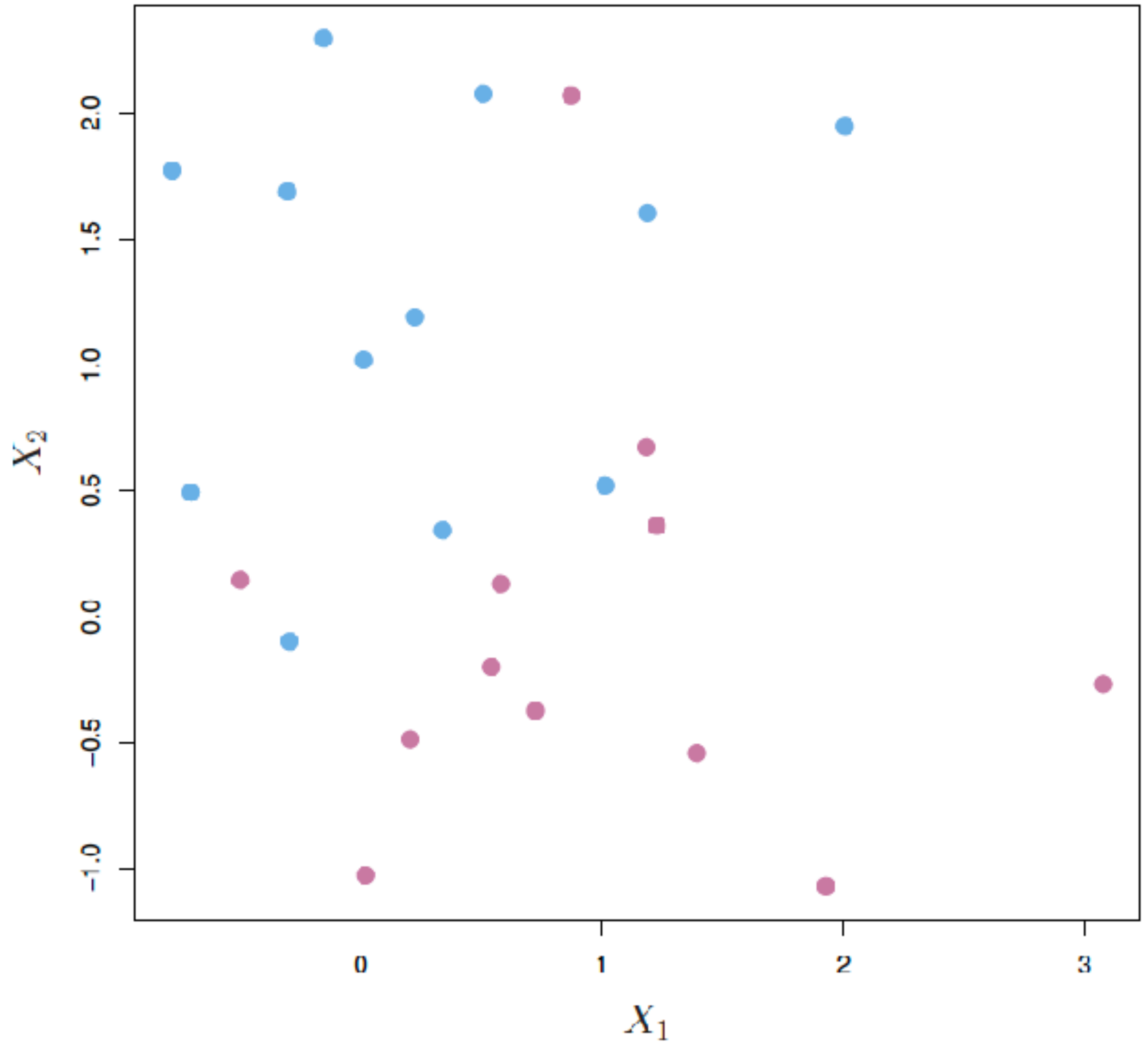
$$subject\ to\ y_i(\beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip}) \geq 1$$

$$for\ all\ \ i = 1, \ldots, N$$

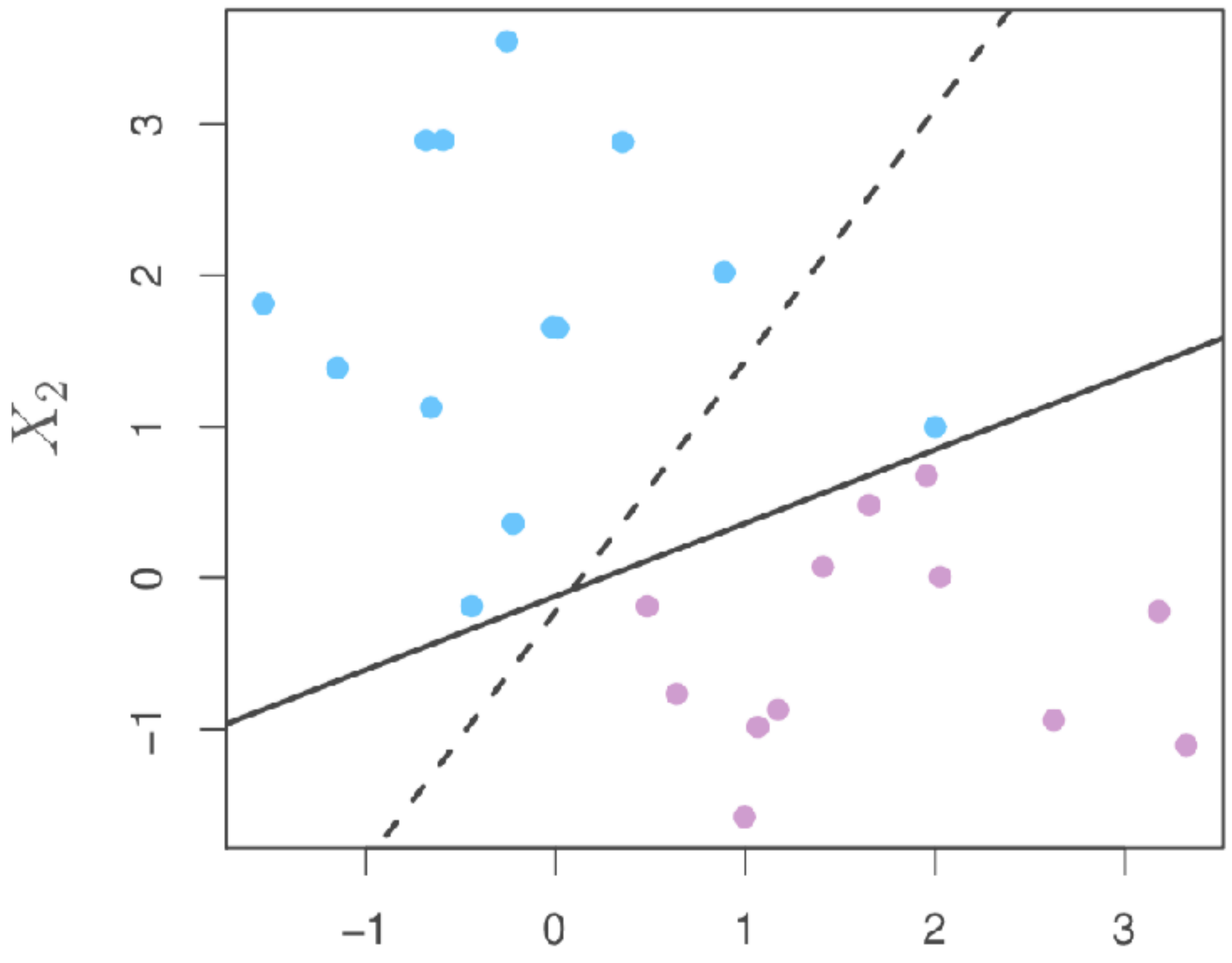
which is easy to optimize since it is differenciable a convex function

Support Vector Machines

Sometimes data is **not separabe**

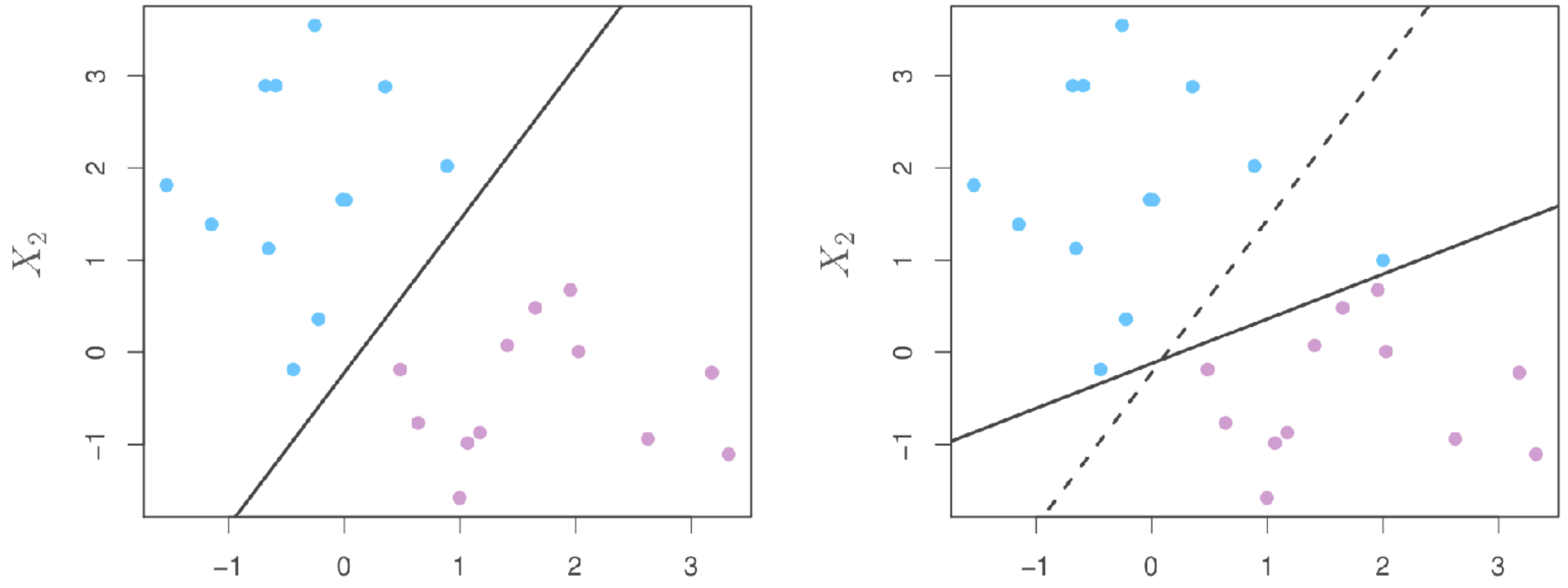


Sometimes is **separable but noisy**



Support Vector Machines

- Data is not that simple… In most of the cases data is **not separable**, and sometimes, the data is separable, but **noisy**.



- The support vector classifier **maximizes a soft margin**. Parameter C determines how hard points violating the constraint should be penalized

$$min_{\beta,\xi}\frac{1}{2}||\beta||^2 + \frac{C}{N}\sum_i \xi_i$$

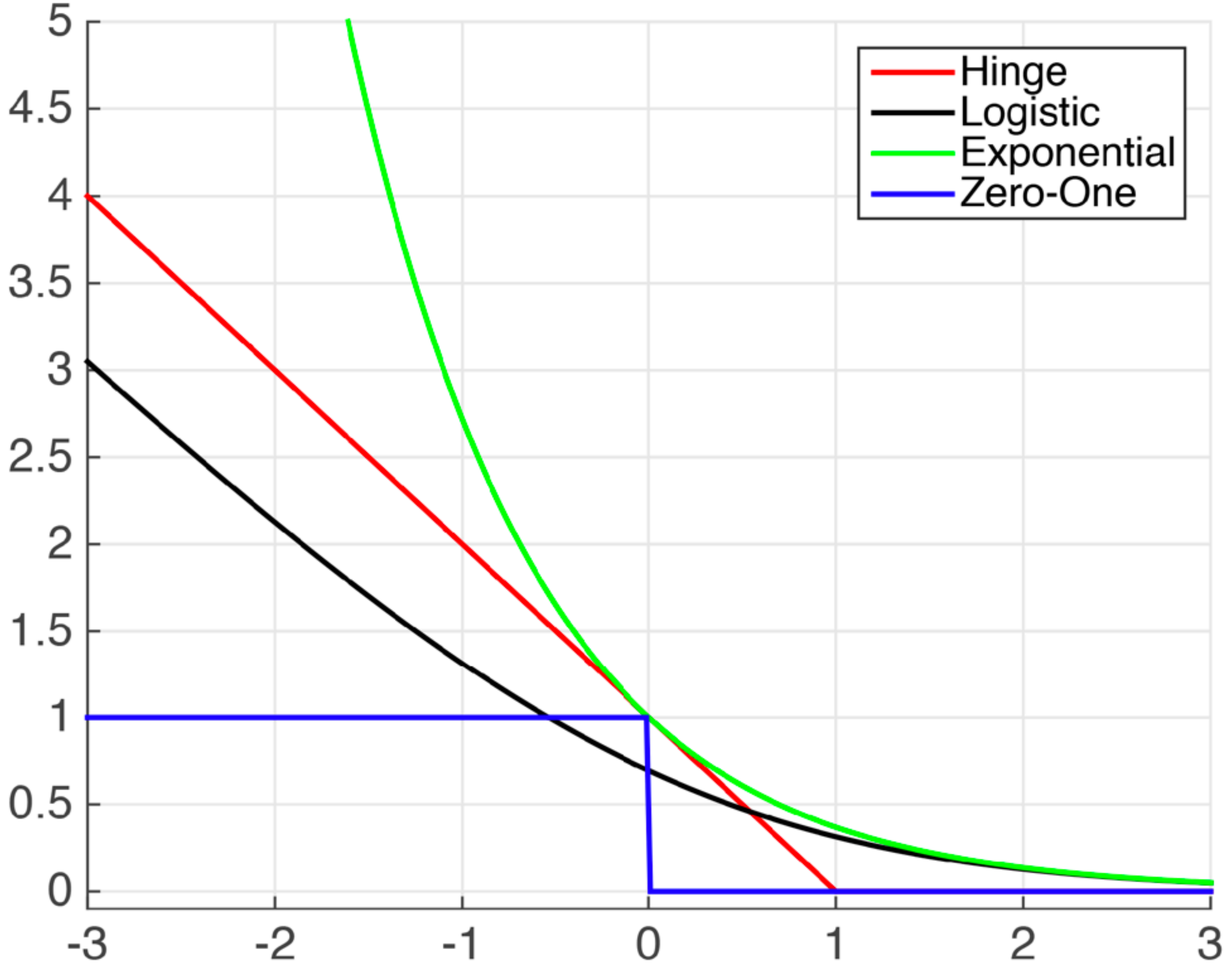
$$s.t.\, y_i(\beta x_i) \geq 1 - \xi_i, \xi \geq 0, \forall i \in \{1,...,N\}$$

Support Vector Machines

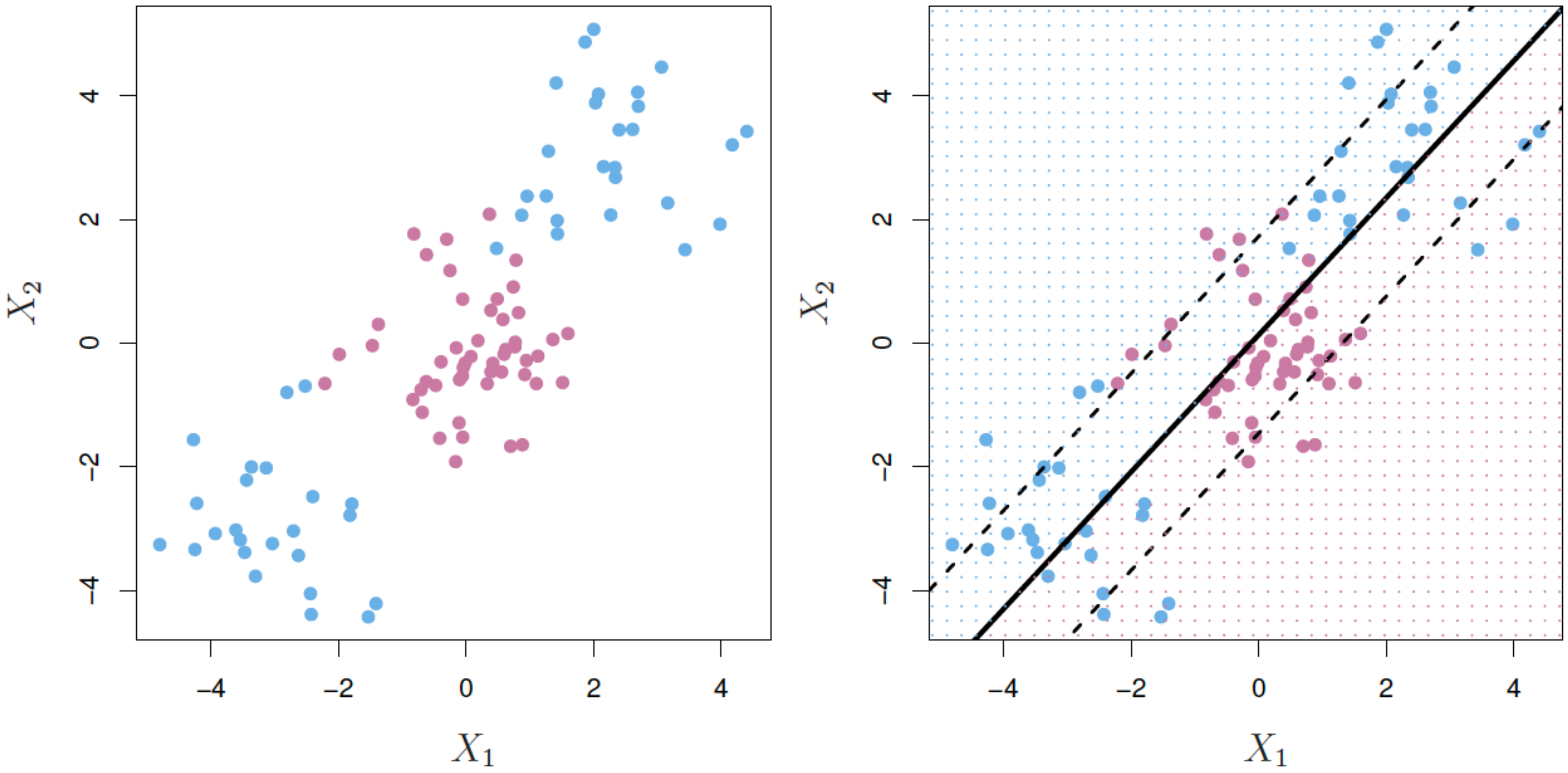
- Observe that if $y_i(\beta \cdot \mathbf{x}_i) \geq 1$, then $\xi_i = 0$ and in this case there is no contribution to the penalty, but if the margin $y_i(\beta \cdot \mathbf{x}_i) < 1$, $\xi_i > 0$ and the penalty term increases $\frac{C}{N}\xi_i$:

$$\xi_j = max(0,1 - y_i(\beta \cdot \mathbf{x_i}))$$

Loss functions

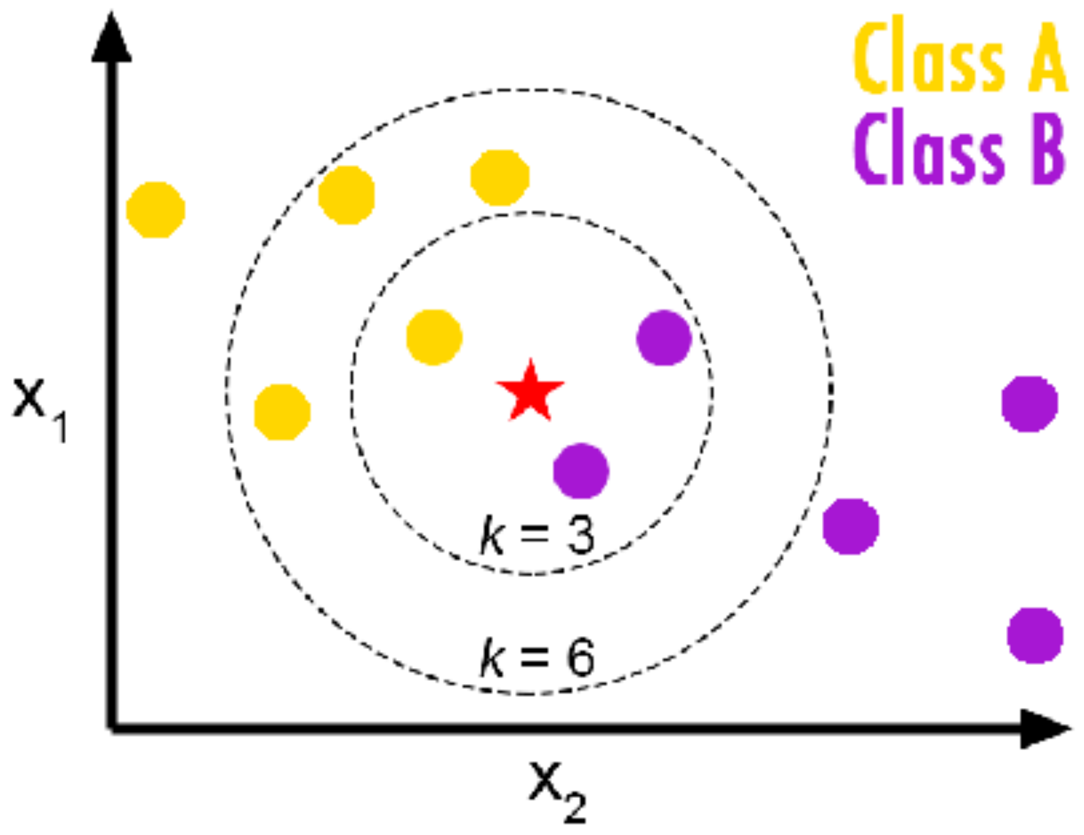


Linear Models can fail



K-Nearest Neighbor

- K-Nearest Neighbor is considered a lazy learning algorithm that classifies data sets based on their similarity with neighbors
- **Assumption**: Similar Inputs have similar outputs
- **Classification rule**: For a test input **x**, assign the most common label amongst its k most similar training inputs



★ is a new Example to be classified

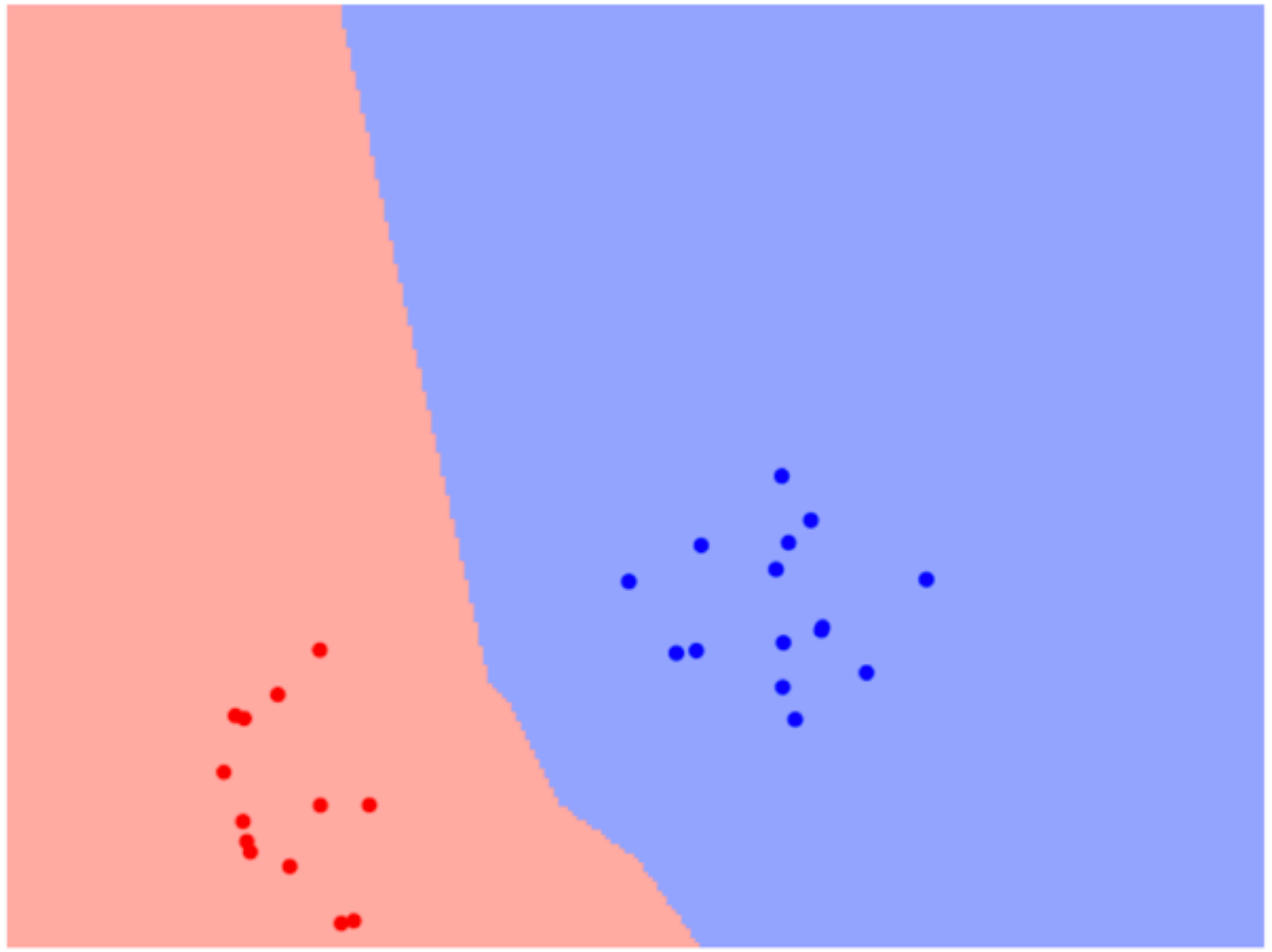
K stands for the number of neighbors to consider for the classification

K-Nearest Neighbor

K-Nearest Neighbors Demo

This interactive demo lets you explore the K-Nearest Neighbors algorithm for classification.
Each point in the plane is colored with the class that would be assigned to it using the K-Nearest Neighbors algorithm. Points for which the K-Nearest Neighbor algorithm results in a tie are colored white.

You can move points around by clicking and dragging!



Metric

L1 L2

Num classes

2 3 4 5

Num Neighbors (K)

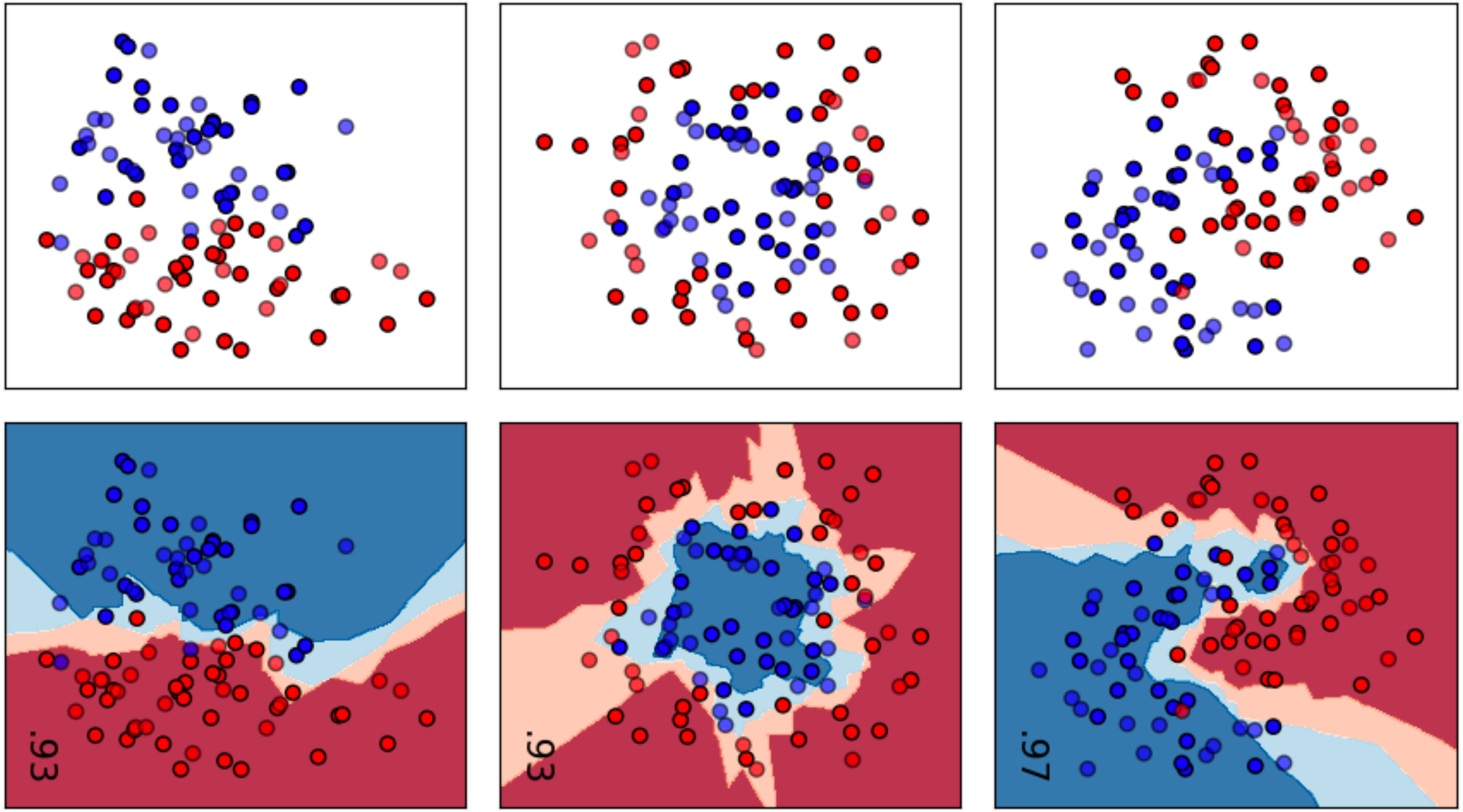
1 2 3 4 5 6 7

Num points

20 30 40 50 60

http://vision.stanford.edu/teaching/cs231n-demos/knn/

K-Nearest Neighbor

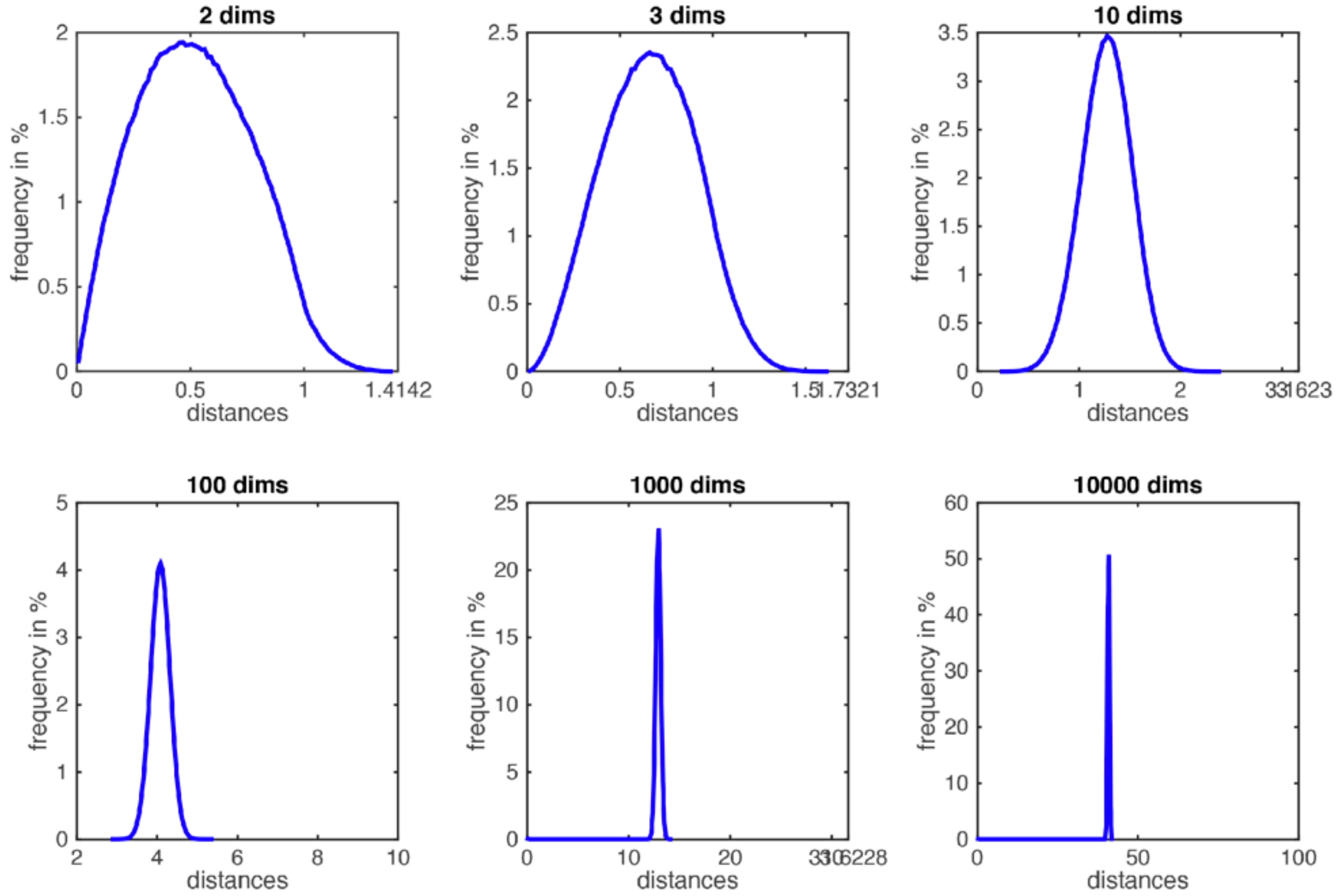


K-Nearest Neighbor

- **Choosing the right K value?**
  - To select the K that's right for your data, we run the KNN algorithm several times with different values of K and choose the K that reduces the number of errors we encounter while maintaining the algorithm's ability to accurately make predictions when it's given data it hasn't seen before.
- Some tips:
  - Small K-value = unstable/Overfitting
  - Large K-value = stable/small accuracy
  - Odd K-value to have a tiebreaker
- **Problems**:
  - Very **slow method** when the number of samples is large
  - ***Curse of dimensionality***

Curse of Dimensionality

K-NN classifier makes the assumption that similar points share similar labels. Unfortunately, **in high dimensional spaces**, points that are drawn from a probability distribution, tend to **never** be **close** together.



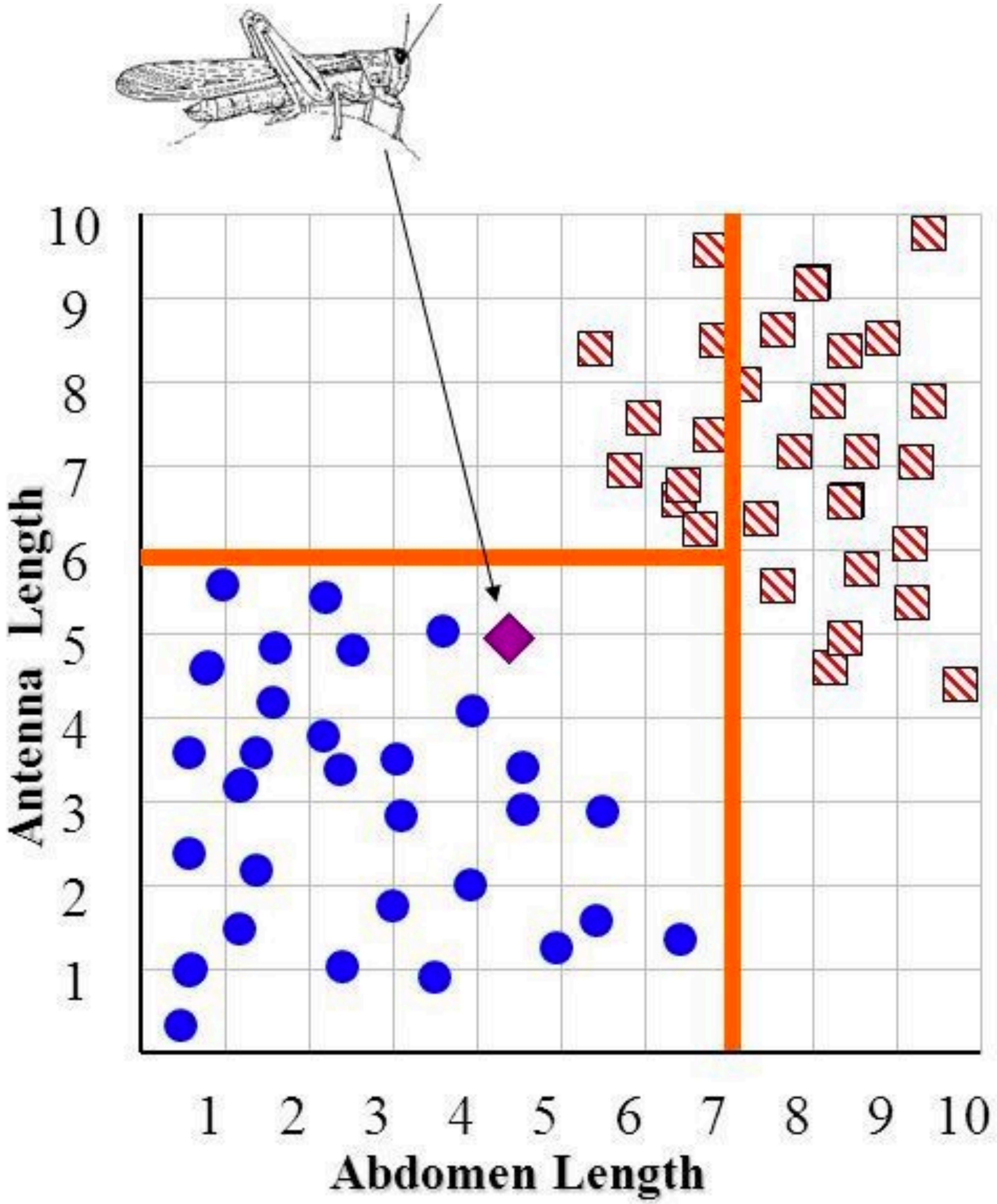
The histogram plots show the distributions of all pairwise distances between randomly distributed points within d dimensional unit squares. As the number of dimensions d grows, all distances concentrate within a very small range.

Tree Based Methods

Decision Tree based Methods

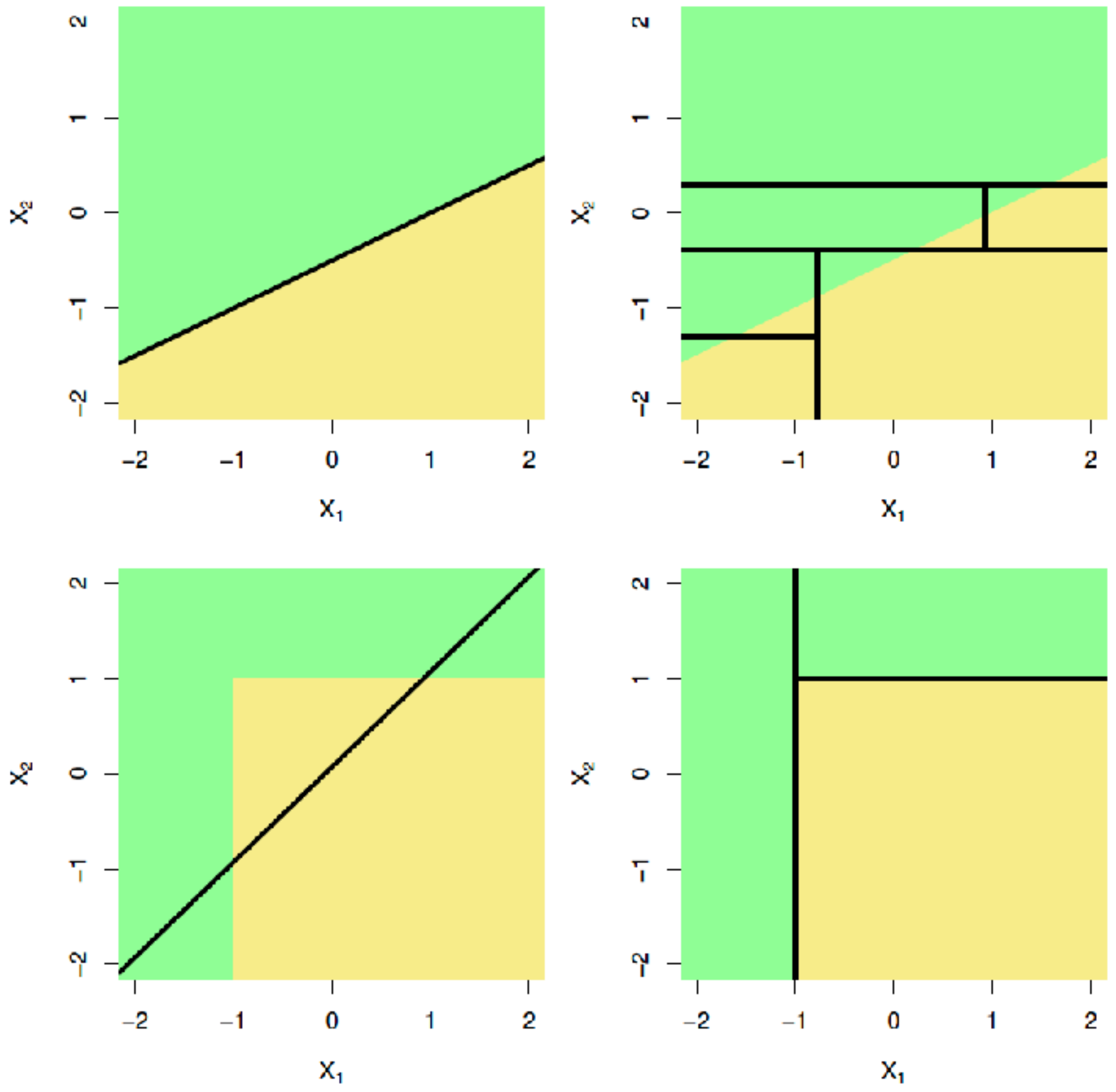
- Tree-based methods are **simple** and **useful for interpretation**
- Typically are not competitive with the best supervised approaches. However methods that grow multiple trees which are then combined to give a consensus prediction are really popular because of its performance. **Bagging**, **Boosting** and **Random Forest** are some of those well known methods.

Decision Tree



Decision Tree

- Trees vs. Linear Models



Bagging

- Build on the assumption of bias Variance

$$\underbrace{\mathbb{E}[(h_D(x)-y)^2]}_{\text{Error}} = \underbrace{\mathbb{E}[(h_D(x)-\bar{h}(x))^2]}_{\text{Variance}} + \underbrace{\mathbb{E}[(\bar{h}(x)-\bar{y}(x))^2]}_{\text{Bias}} + \underbrace{\mathbb{E}[(\bar{y}(x)-y(x))^2]}_{\text{Noise}}$$

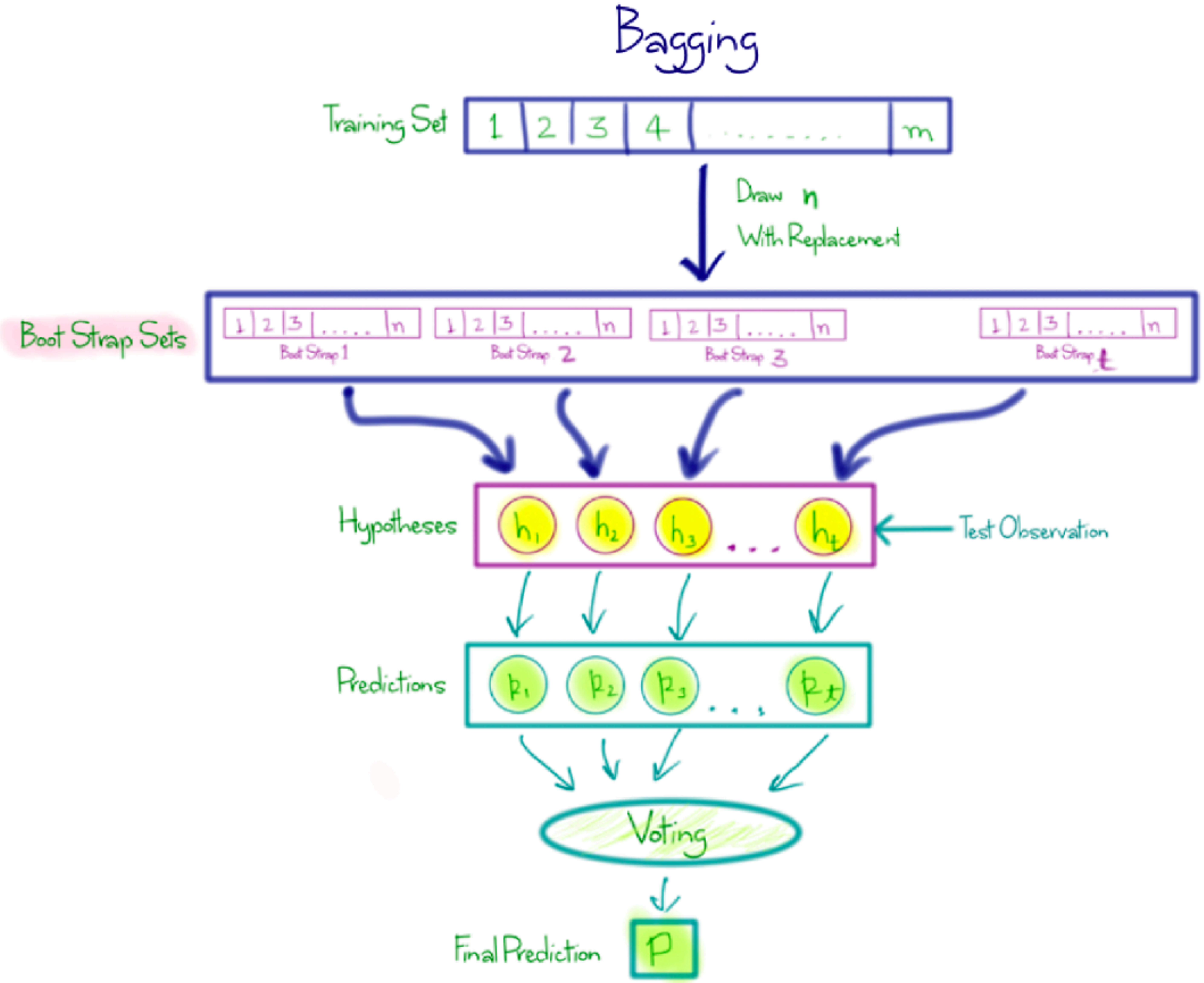
- The weak law of large numbers says (roughly) for i.i.d. random variables x_i with mean $\bar{x}$, we have

$$\frac{1}{m}\sum_{i=1}^{m} x_i \to \bar{x} \text{ as } m \to \infty$$

- Apply this to classifiers: Assume we have m training sets $D1, D2, \ldots, Dn$ drawn from P^n. Train m classifiers and avereage the result:

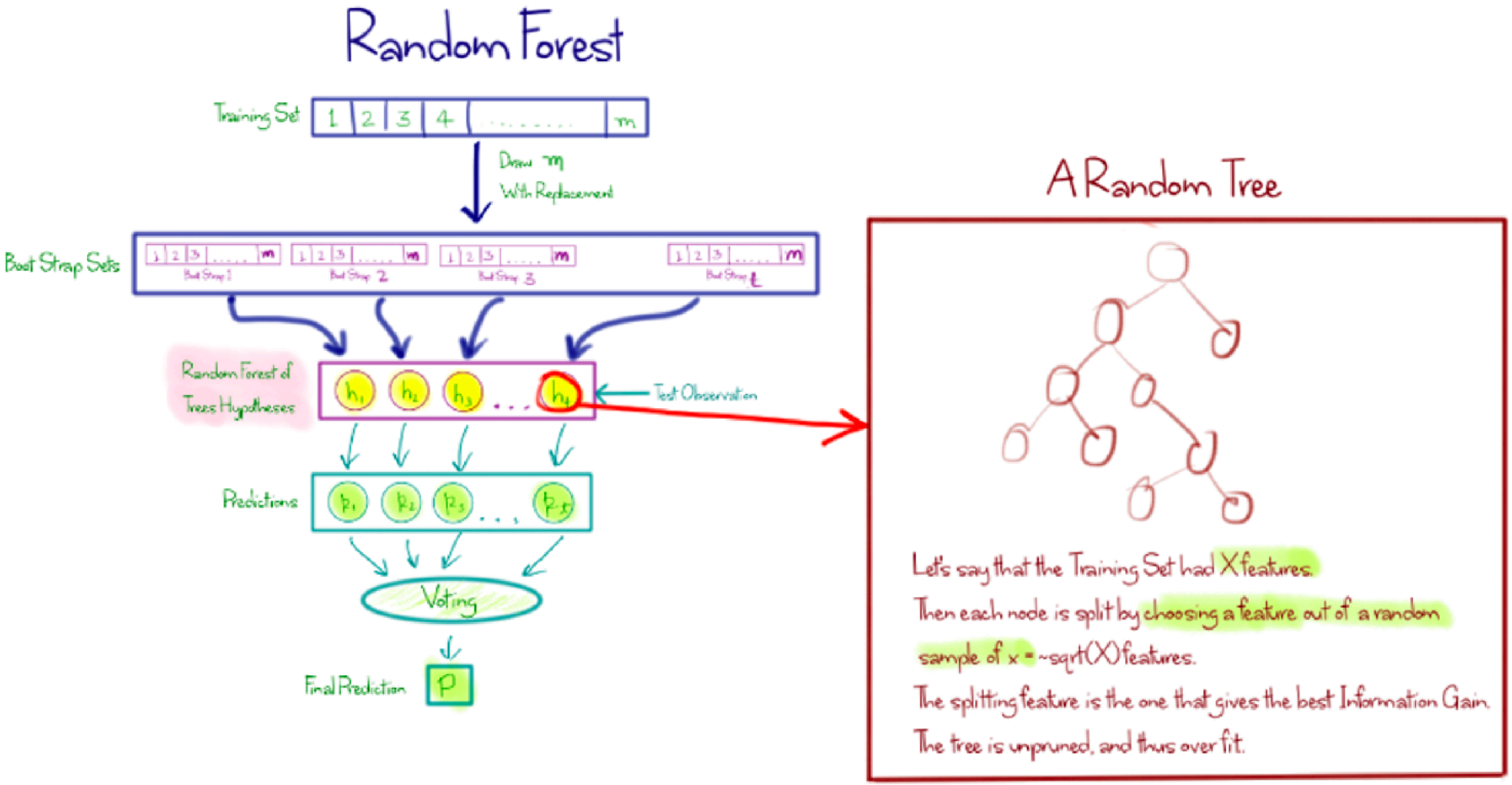
$$\hat{h} = \frac{1}{m}\sum_{i=1}^{m} h_{D_i} \to \bar{h} \qquad as\ m \to \infty$$

Bagging

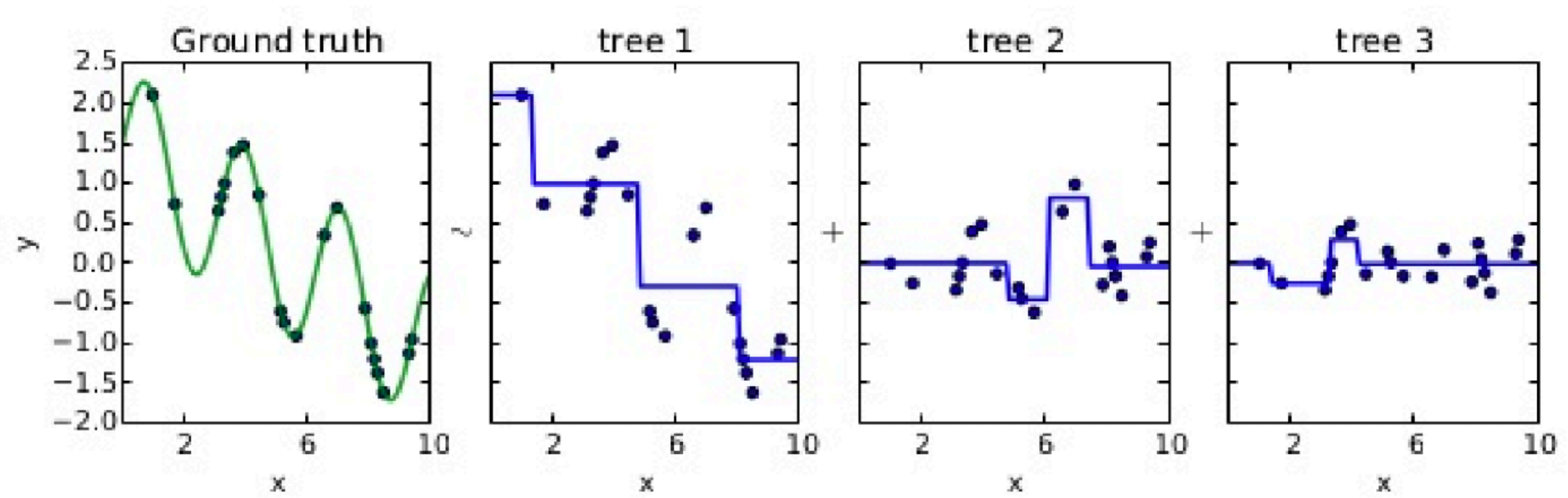


Bagging
Training Set
1
2
3
4
m
Draw n
With Replacement
Boot Strap Sets
Boot Strap 1
Boot Strap 2
Boot Strap 3
Boot Strap t
Hypotheses
Test Observation
Predictions
Voting
Final Prediction
P

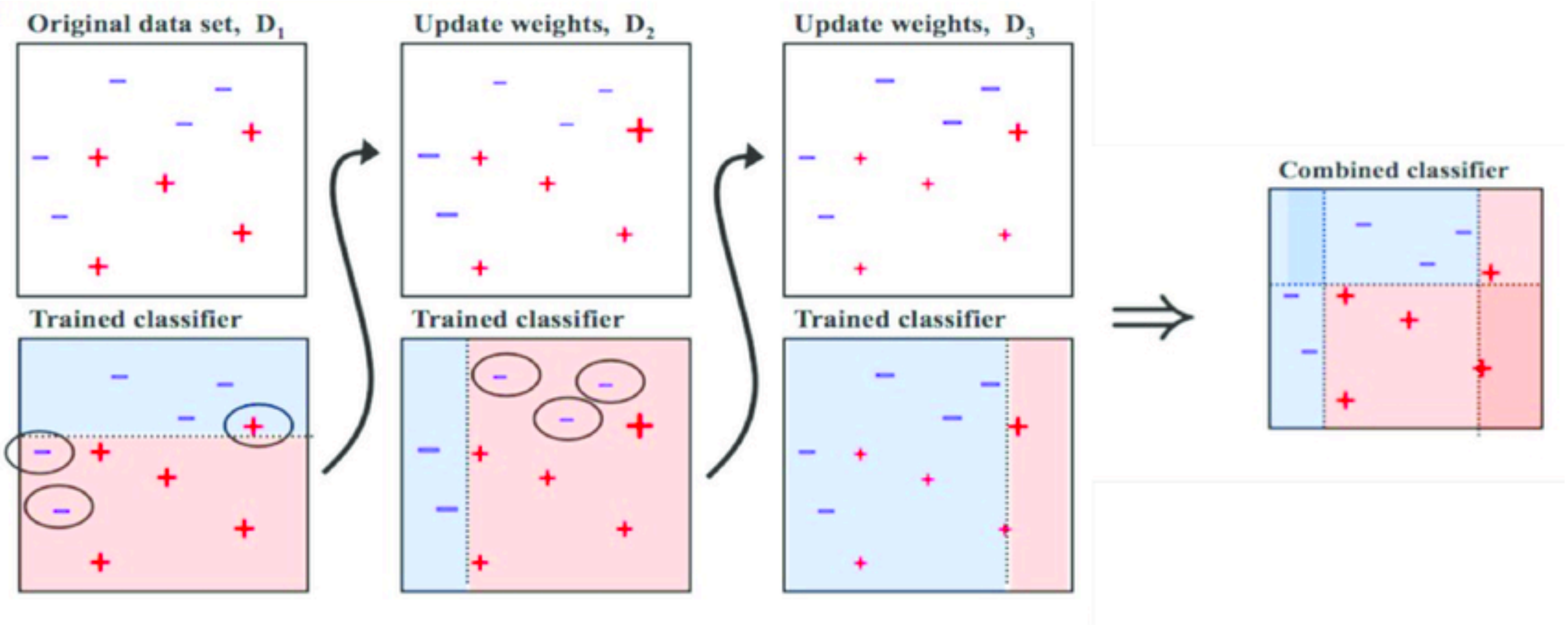
Random Forest



Boosting

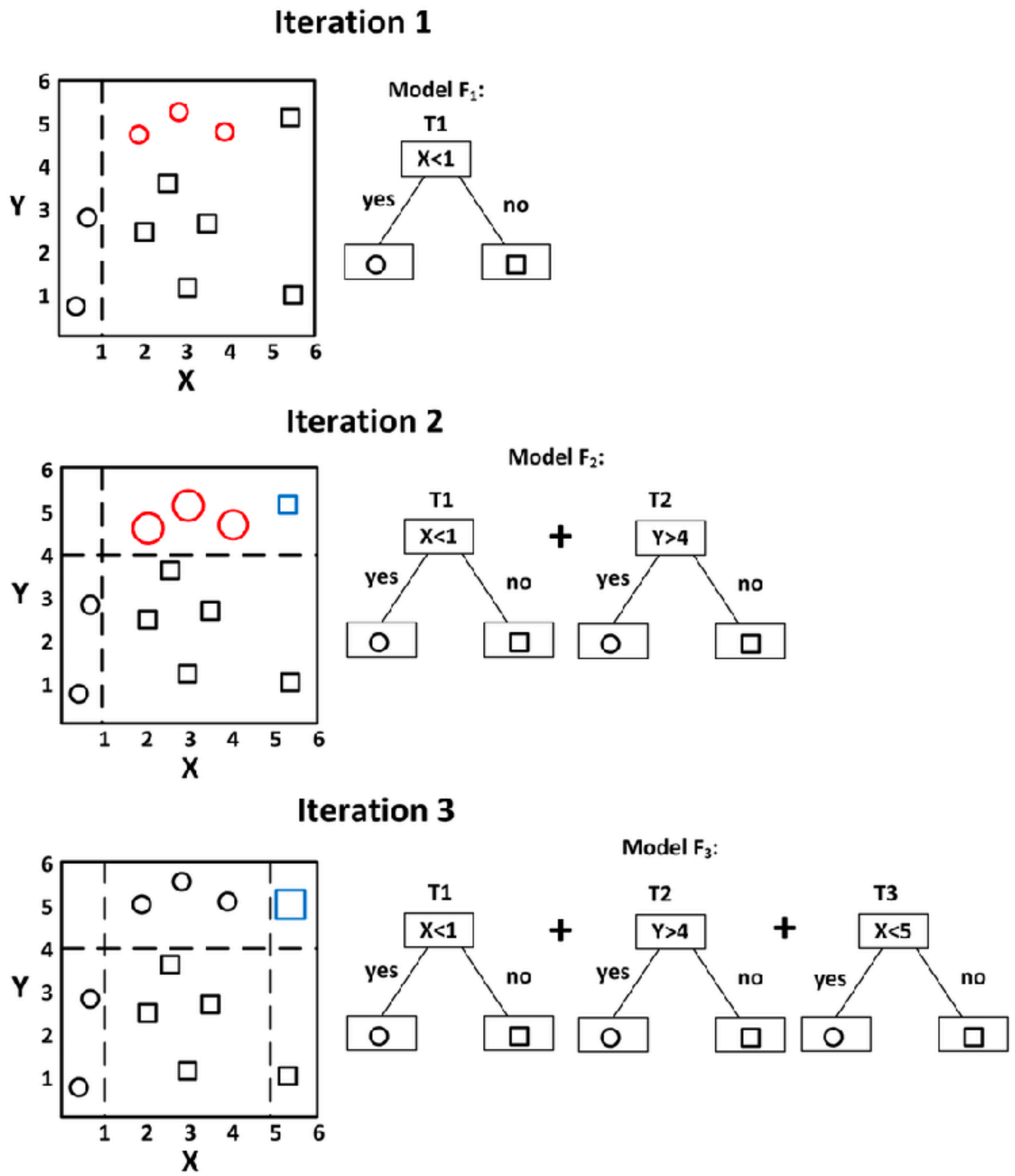

Ground truth
tree 1
tree 2
tree 3
y
x
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
2
6
10
+
+

Boosting



AdaBoost works on improving the areas where the base learner fails.

Gradient Boosting



Kernel SVM

- It can be demostrated than Linear SVM is equivalent to this equation (dual version):

$$f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i \langle x, x_i \rangle$$

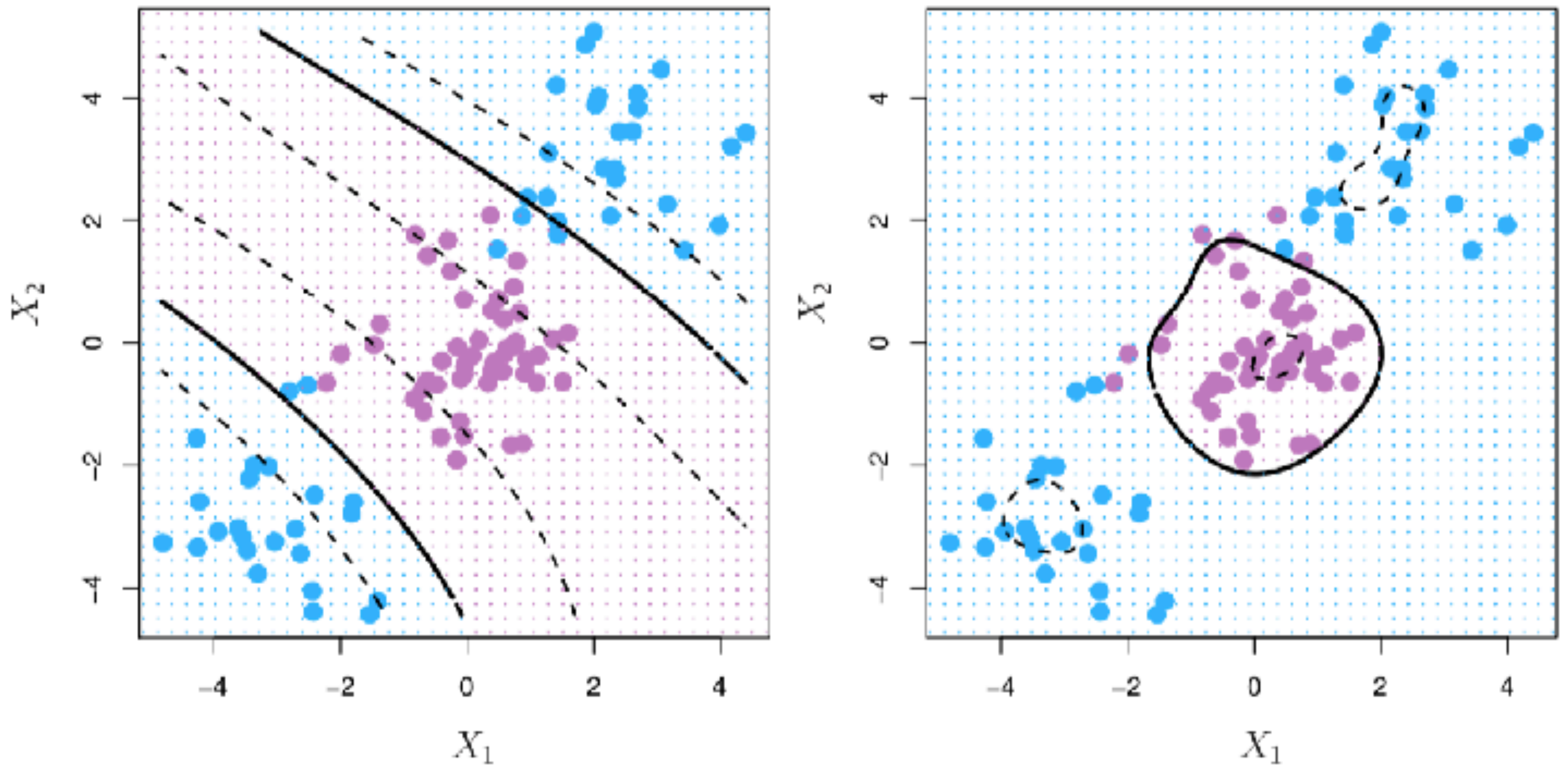
- Based on the observation than the all data terms are based exclusivly used to computed dot products it suggested to find a function $K(x_i, x_j)$, called **kernel**, that corresponds to the dot prodcut of x_i and x_j in such higher space. The Kernel SVM equations is as follows:

$$f(x) = \beta_0 + \sum_{i \in S} \hat{\alpha}_i K(x, x_i)$$

Kernel SVM

Guassian Kernel

$$K(x_i, x_{i}') = exp(-\gamma \sum_{j=1}^{p}(x_{ij} - x_{i'j})^2)$$



$$f(x) = \beta_0 + \sum_{i \in S} \hat{\alpha}_i K(x, x_i)$$