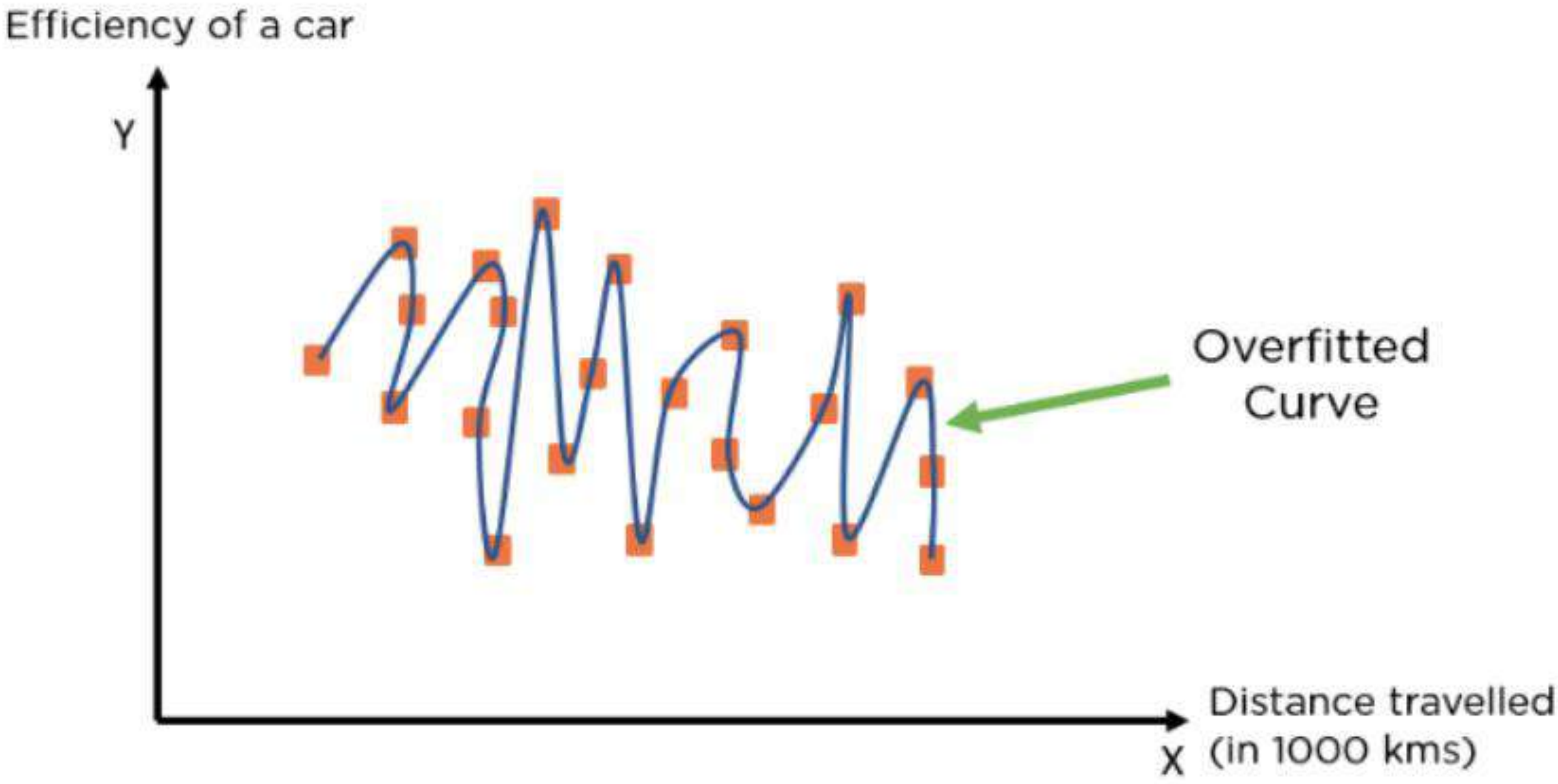
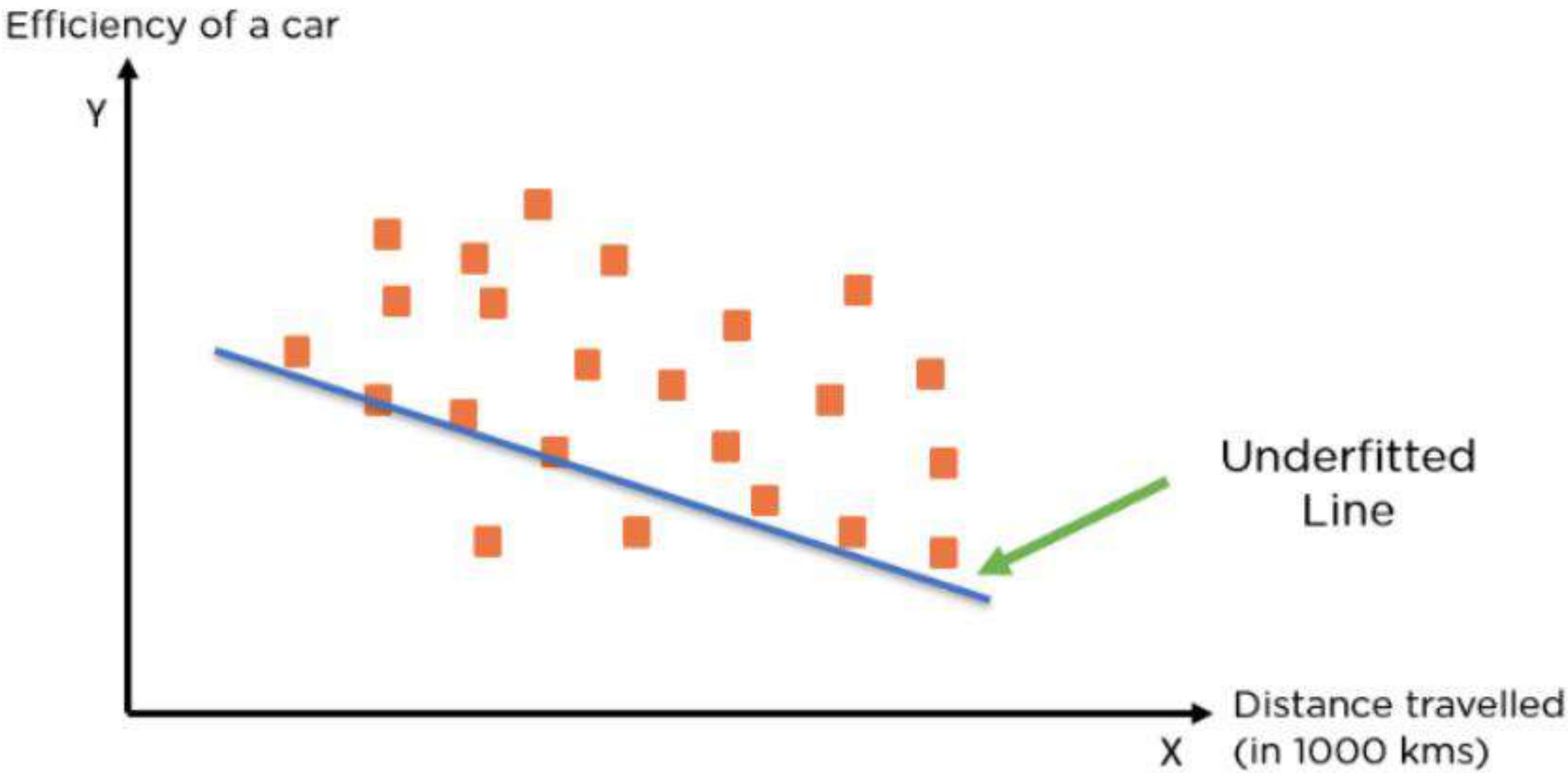
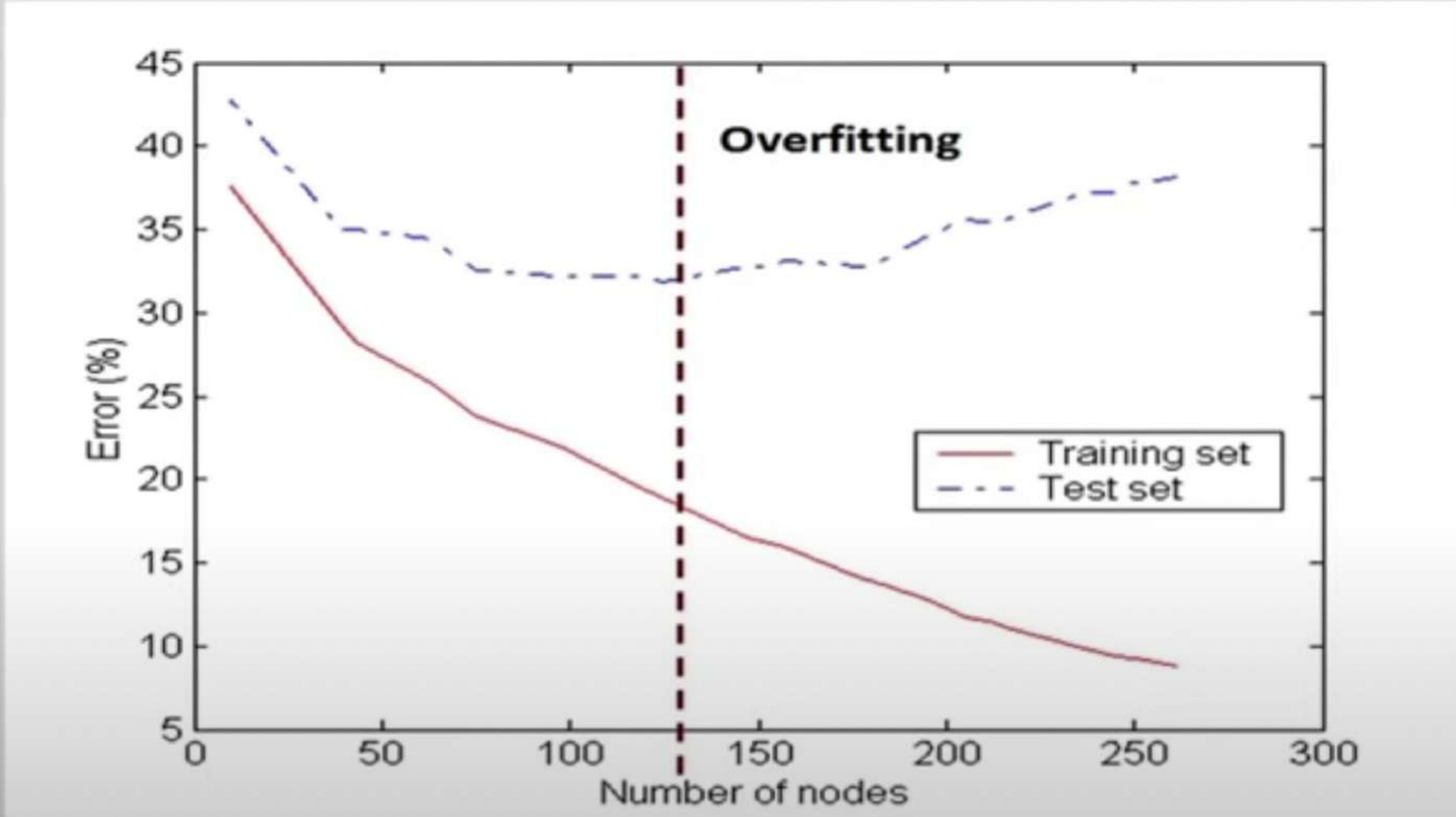

Efficiency of a car
Y
Overfitted
Curve
Distance travelled
(in 1000 kms)
X



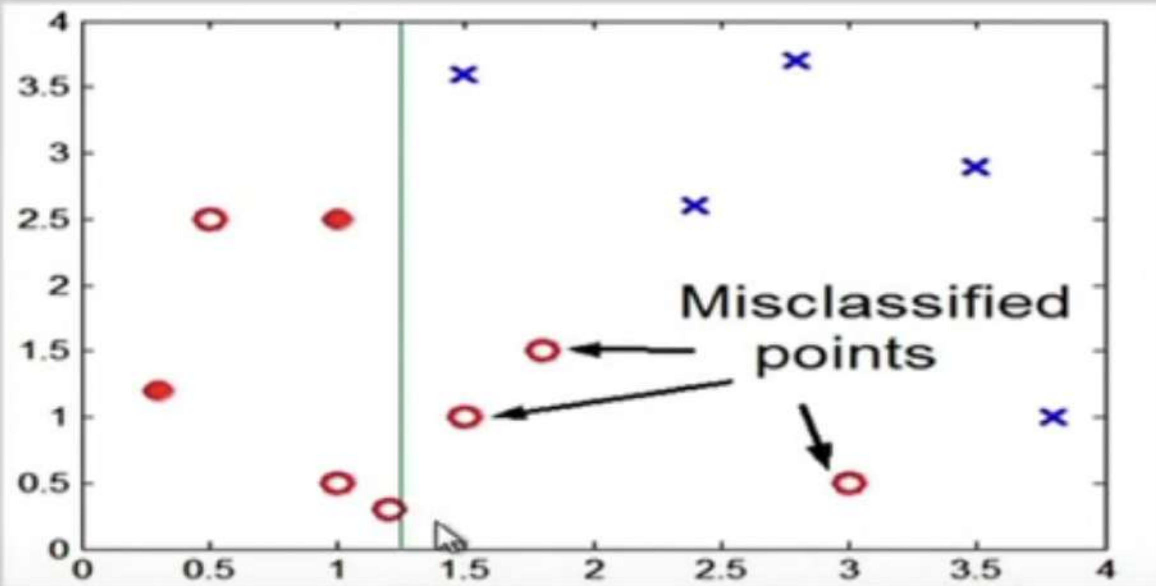
Efficiency of a car
Y
Underfitted
Line
X
Distance travelled
(in 1000 kms)

Underfitting and Overfitting



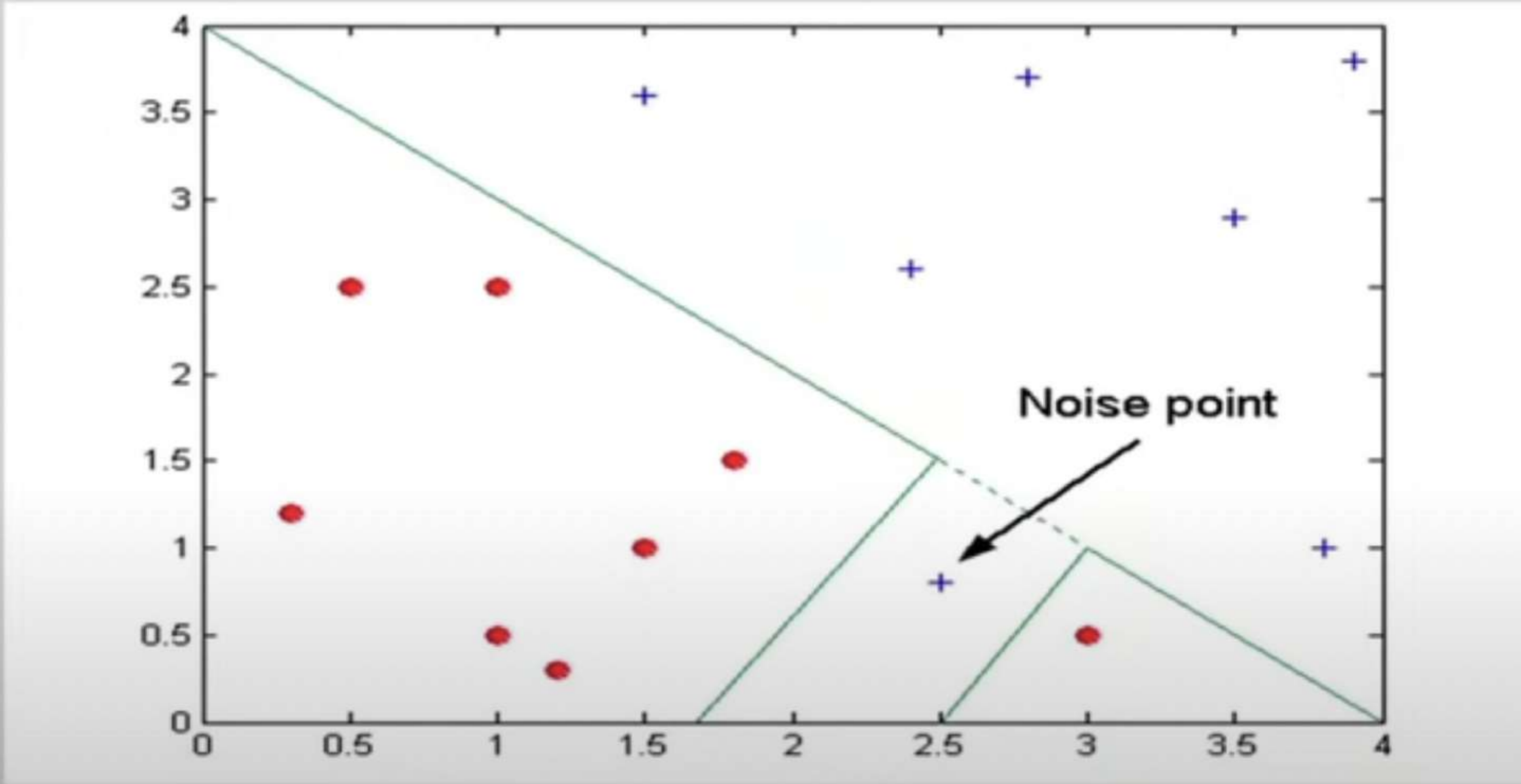
Underfitting: when model is too simple, both training and test errors are large

Overfitting due to Insufficient Examples

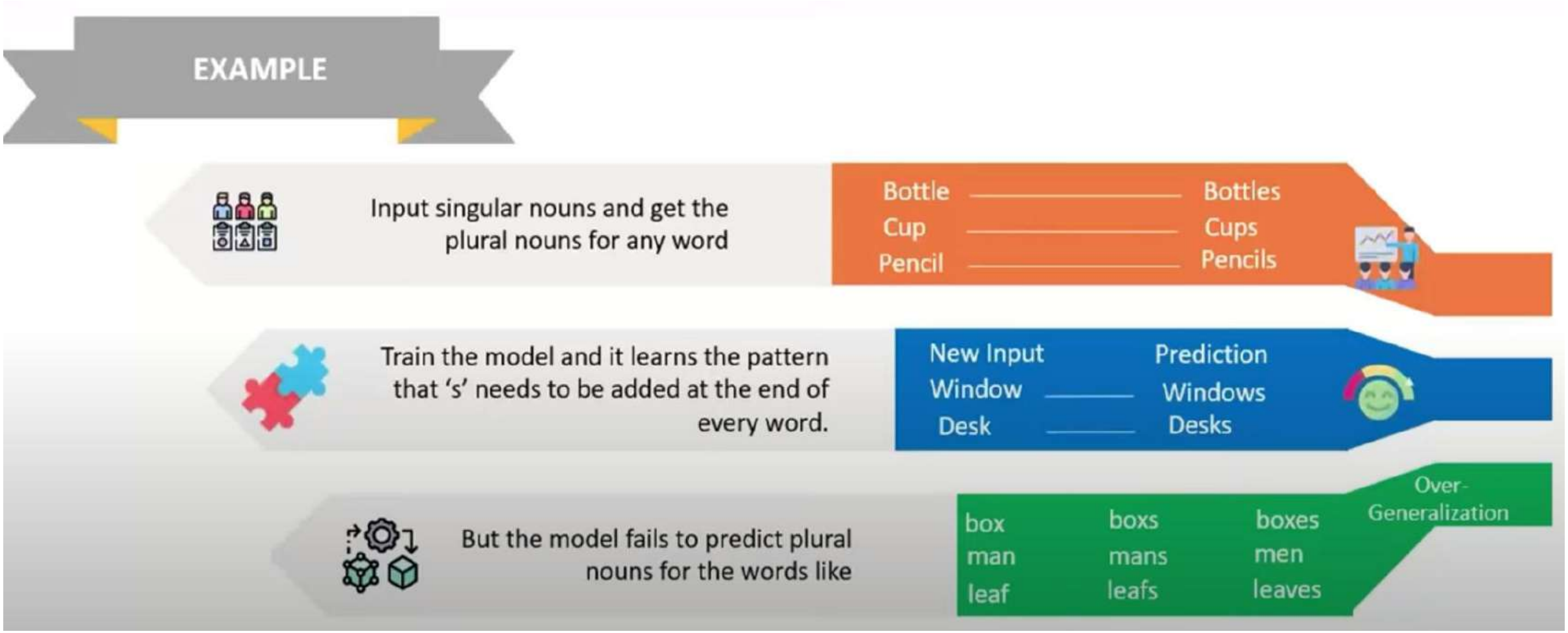


Lack of data points makes it difficult to predict correctly the class labels of that region

Overfitting due to Noise



Decision boundary is distorted by noise point


EXAMPLE
Input singular nouns and get the plural nouns for any word
Bottle
Bottles
Cup
Cups
Pencil
Pencils
Train the model and it learns the pattern that ‘s’ needs to be added at the end of every word.
New Input
Prediction
Window
Windows
Desk
Desks
But the model fails to predict plural nouns for the words like
box
boxs
boxes
man
mans
men
leaf
leafs
leaves
Over-Generalization

Reasons for Overfitting

- Data used for training is not cleaned and contains noise (garbage values) in it
- The model has a high variance
- The size of the training dataset used is not enough
- The model is too complex

Reasons for Underfitting

- Data used for training is not cleaned and contains noise (garbage values) in it
- The model has a high bias
- The size of the training dataset used is not enough
- The model is too simple

Notes on Overfitting

- **overfitting** happens when a model is capturing idiosyncrasies of the data rather than generalities.
  - Often caused by too many parameters relative to the amount of training data.
  - E.g. an order-*N* polynomial can intersect any *N+1* data points

Pre-Pruning (Early Stopping)

- Typical stopping conditions for a node:
  - Stop if all instances belong to the same class
  - Stop if all the attribute values are the same
- More restrictive conditions:
  - Stop if number of instances is less than some user-specified threshold
  - Stop if class distribution of instances are independent of the available features (e.g., using χ^2 test)
  - Stop if expanding the current node does not improve impurity measures (e.g., Gini or information gain).

Reduced-error Pruning

- A post-pruning, cross validation approach
  - Partition training data into "grow" set and "validation" set.
  - Build a complete tree for the "grow" data
  - Until accuracy on validation set decreases, do:

    For each non-leaf node in the tree

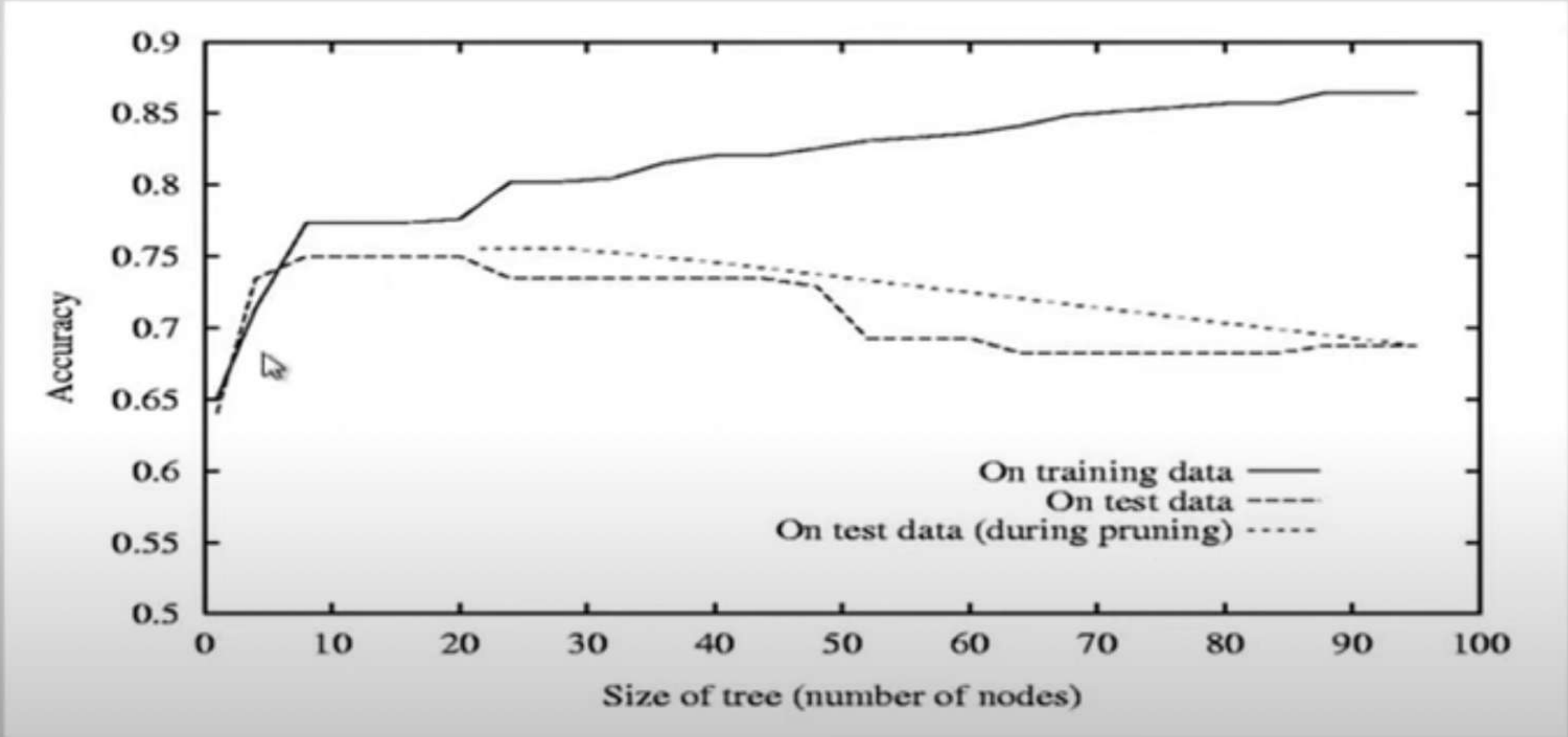
    Temporarily prune the tree below; replace it by majority vote

    Test the accuracy of the hypothesis on the validation set

    Permanently prune the node with the greatest increase in accuracy on the validation test.
- Problem: Uses less data to construct the tree
- Sometimes done at the rules level

General Strategy: Overfit and Simplify

Reduced Error Pruning



Model Selection & Generalization

- Learning is an ill-posed problem; data is not sufficient to find a unique solution
- The need for inductive bias, assumptions about H
- Generalization: How well a model performs on new data
- Overfitting: H more complex than C or f
- Underfitting: H less complex than C or f

Triple Trade-Off

- There is a trade-off between three factors:
  - Complexity of H, *c* (H),
  - Training set size, *N*,
  - Generalization error, *E* on new data

- As *N* *increases*-, *E* decreases
- As *c* (H) *increases*-, first *E* *decreases* and then *E*- *increases*
- As *c* (H)- *increases*, the training error *decreases* for some time and then stays constant (frequently at 0)

Notes on Overfitting

- Overfitting results in decision trees that are more complex than necessary
- Training error no longer provides a good estimate of how well the tree will perform on previously unseen records

Dealing with Overfitting

- Use more data
- Use a tuning set
- **Regularization**
- Be a Bayesian

Avoid Overfitting

- How can we avoid overfitting a decision tree?
  - Prepruning: Stop growing when data split not statistically significant
  - Postpruning: Grow full tree then remove nodes

- Methods for evaluating subtrees to prune:
  - Minimum description length (MDL):

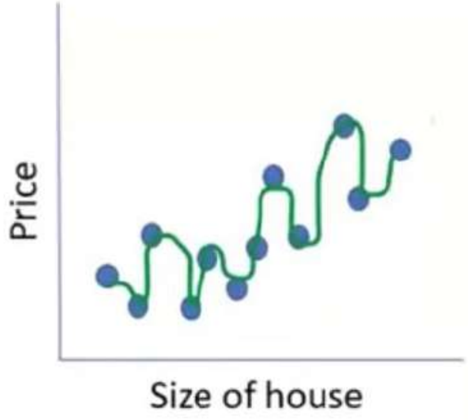
  Minimize: size(tree) + size(misclassifications(tree))
  - Cross-validation

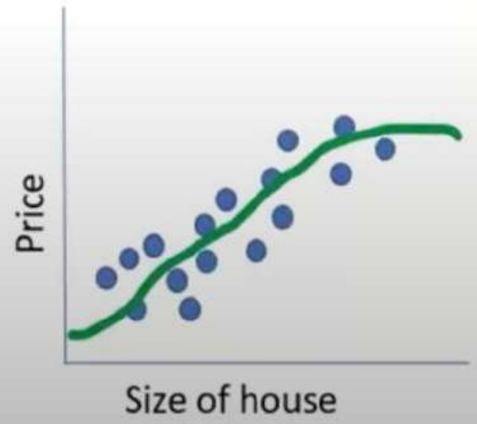
To avoid this condition regularization is used. Regularization is a technique used for tuning the function by adding an additional penalty term in the error function. The additional term controls the excessively fluctuating function such that the coefficients don't take extreme values. This technique of keeping a check or reducing the value of error coefficients are called shrinkage methods or weight decay in case of neural networks.

Regularization

- In a linear regression model overfitting is characterized by large weights.

	M = 0	M = 1	M = 3	M = 9
w_0	0.19	0.82	0.31	0.35
w_1		-1.27	7.99	232.37
w_2			-25.43	-5321.83
w_3			17.37	48568.31
w_4				-231639.30
w_5				640042.26
w_6				-1061800.52
w_7				1042400.18
w_8				-557682.99
w_9				125201.43





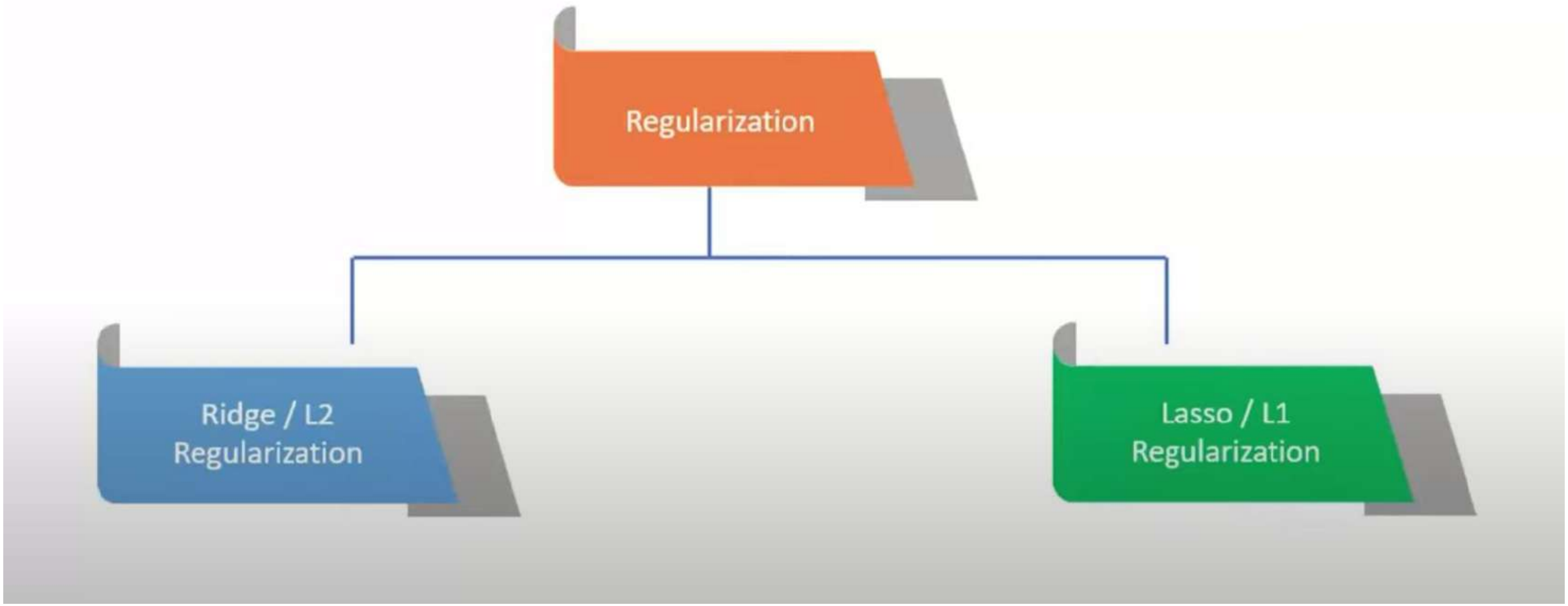
Price = $\beta_0 + \beta_1 * \text{size} + \beta_2 * \text{size}^2 + \beta_3 * \text{size } 3 + \beta_4 * \text{size } 4$

↓

reduce β_3 and β_4 close to zero

↓

Price = $\beta_0 + \beta_1 * \text{size} + \beta_2 * \text{size}^2$


Regularization
Ridge / L2
Regularization
Lasso / L1
Regularization

Which Technique To Use?

Ridge

Lot of features
In the dataset and
all features have
small coefficients.

Lasso

Small number of
features and few
features have high
coefficient value.

Mean Squared Error

$$mse = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - h_\theta(x_i)\right)^2$$

$$h_\theta(x_i) = \theta_0 + \theta_1 x_1 + \theta_2 x_2^2 + \theta_3 x_3^3$$

Lasso Regularization

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (y_{i\,(actual} - y_{i\,(predicted)})^2$$

$$Loss = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{P} |\beta_j|$$

Penalty term regularizes the coefficients

λ = Tuning parameter

Ridge Regularization

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(y_{i\,(actual} - y_{i\,(predicted)}\right)^2$$

$$Loss = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda\sum_{j=1}^{P}\beta_j{}^2$$

Penalty term regularizes the coefficients

λ = Tuning parameter

Penalize large weights in Linear Regression

- Introduce a penalty term in the loss function.

$$E(\vec{w}) = \frac{1}{2}\sum_{n=0}^{N-1}\{t_n - y(x_n, \vec{w})\}^2$$

Regularized Regression

1. (L2-Regularization or Ridge Regression)

$$E(\vec{w}) = \frac{1}{2}\sum_{n=0}^{N-1}(t_n - y(x_n, \vec{w}))^2 + \frac{\lambda}{2}\|\vec{w}\|^2$$

1. L1-Regularization

$$E(\vec{w}) = \frac{1}{2}\sum_{n=0}^{N-1}(t_n - y(x_n, \vec{w}))^2 + \lambda|\vec{w}|_1$$