

The background of the slide is an abstract composition. It features a series of thick, dark, wavy lines that sweep across the frame from the top left towards the bottom right. These lines are layered over a light gray background that contains a grid of faint, semi-transparent numbers (0-9) scattered across it. The overall aesthetic is modern and technical, suggesting data or complex systems.

8. Model Evaluation

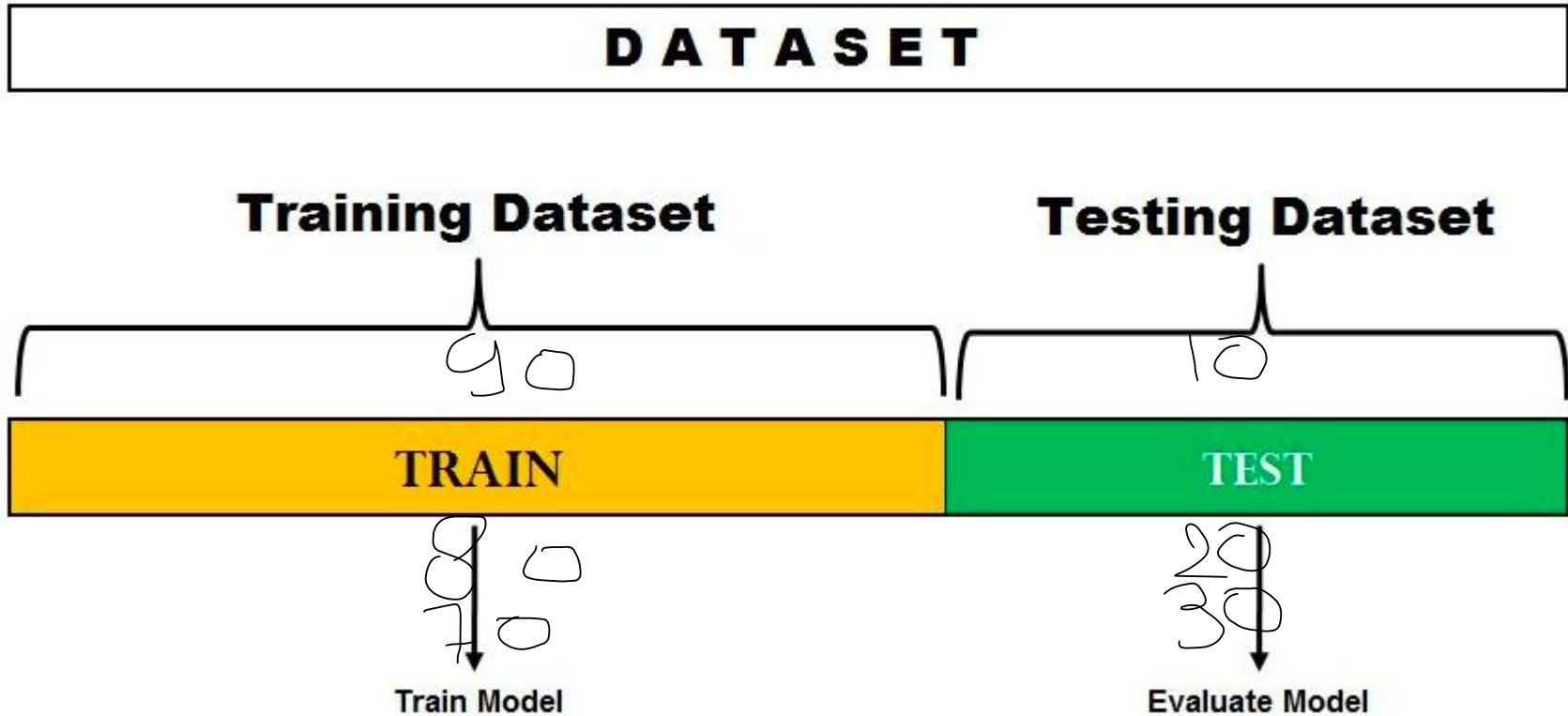
Model Evaluation

- Model Evaluation is an integral part of the model development process.
- It helps to find the best model that represents our data and how well the chosen model will work in the future.
- Evaluating model performance with the data used for training is not acceptable in data science because it can easily generate overoptimistic and overfitted models.
- There are two methods of evaluating models in data science:
 - Hold-Out
 - Cross-Validation.
- To avoid overfitting, both methods use a test set (not seen by the model) to evaluate model performance.

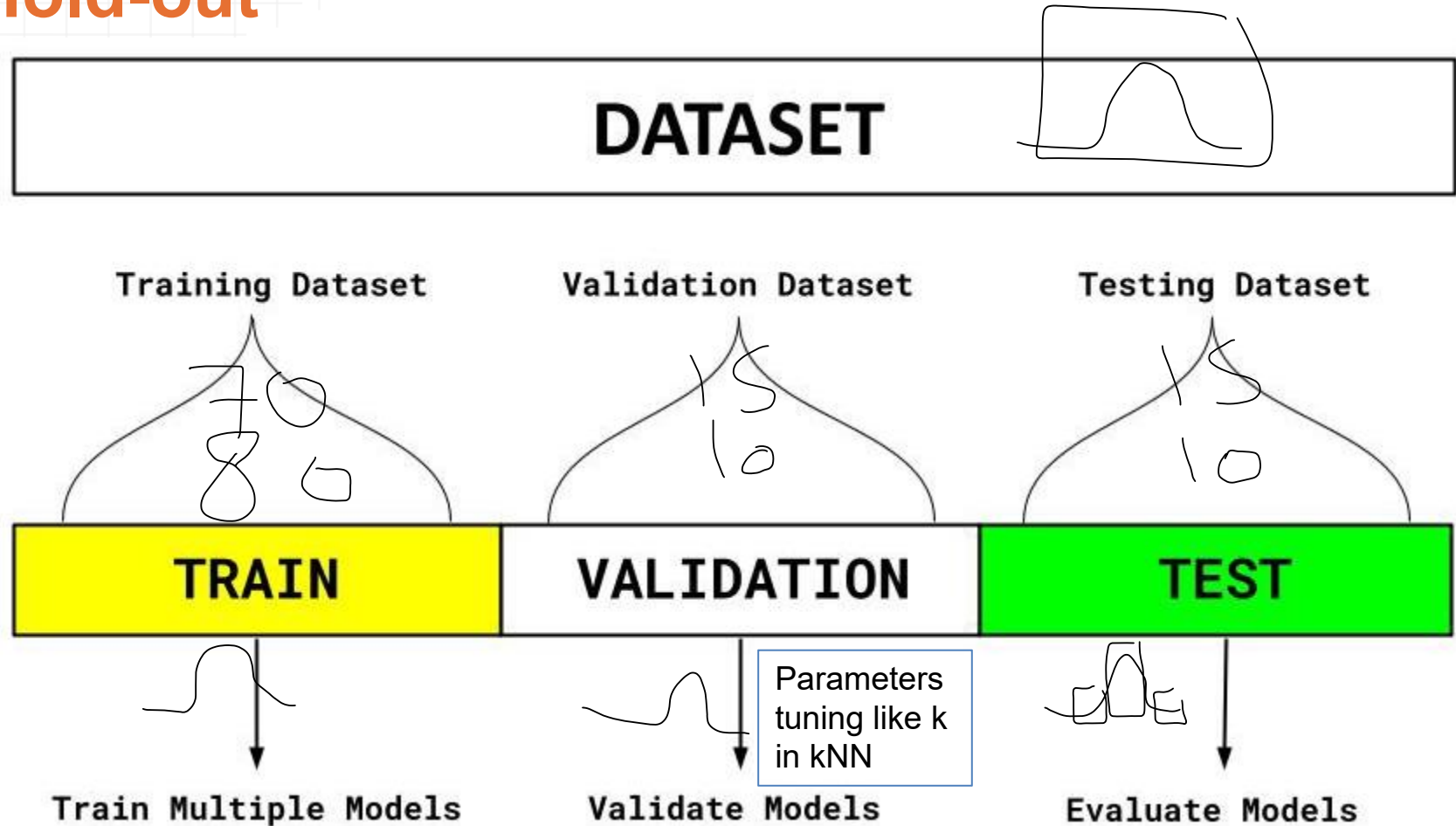
Hold-Out

- In this method, the mostly large dataset is randomly divided to three subsets:
 1. **Training set** is a subset of the dataset used to build predictive models.
 2. **Validation set** is a subset of the dataset used to **assess the performance of model built in the training phase**. It provides a test platform for fine tuning model's **parameters** and selecting the best-performing model. Not all modeling algorithms need a validation set.
 3. **Test set** or unseen examples is a subset of the dataset to assess the likely future performance of a model. If a model fit to the training set much better than it fits the test set, overfitting is probably the cause.

Hold-out



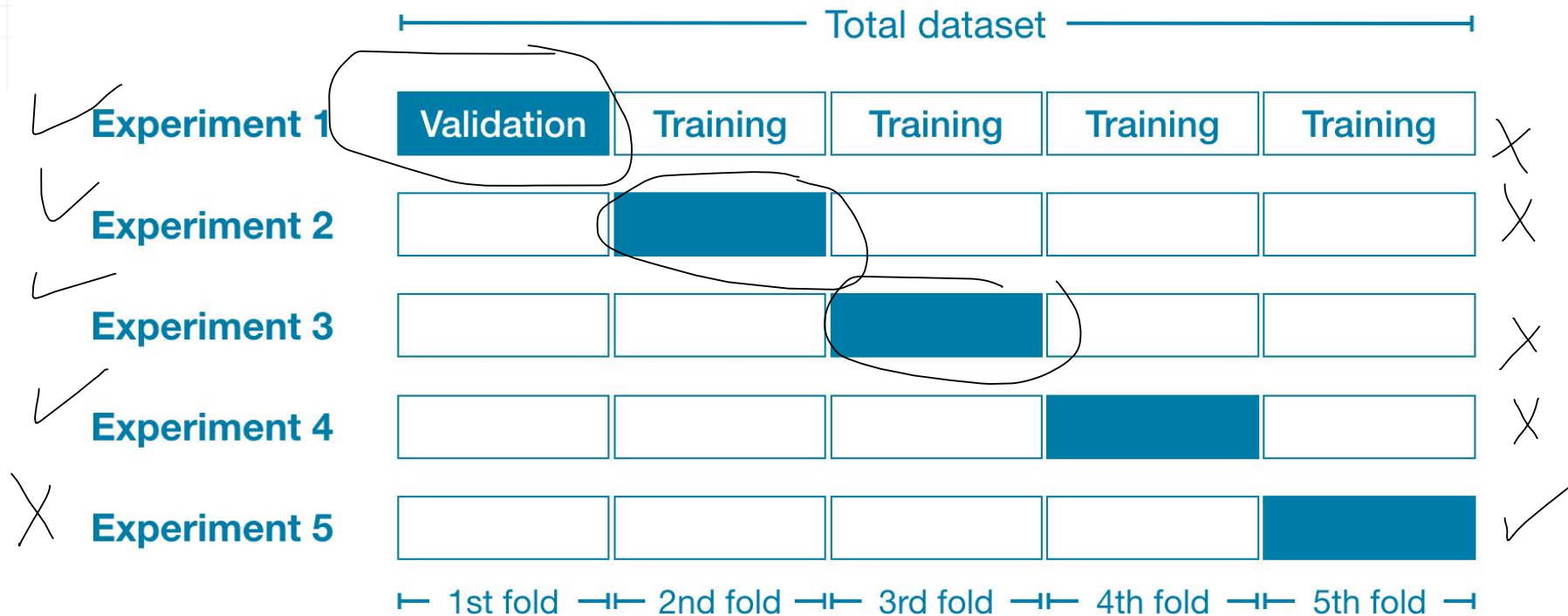
Hold-out



Cross-Validation

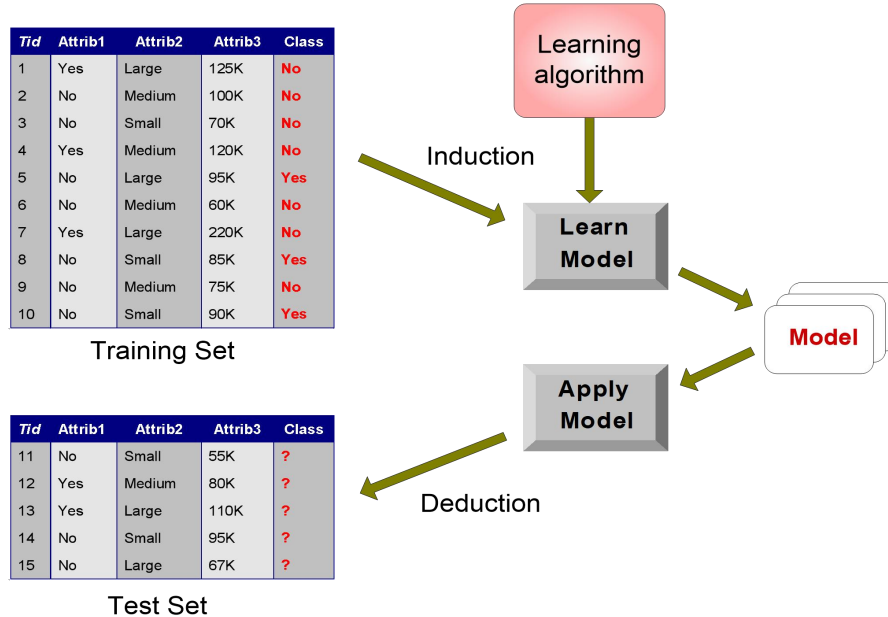
- When only a limited amount of data is available, to achieve an unbiased estimate of the model performance we use k -fold cross-validation.
- In k -fold cross-validation, we divide the data into k subsets of equal size.
- We build models k times, each time leaving out one of the subsets from training and use it as the test set. If k equals the sample size, this is called "leave-one-out".

Cross validation: 5-folds CV

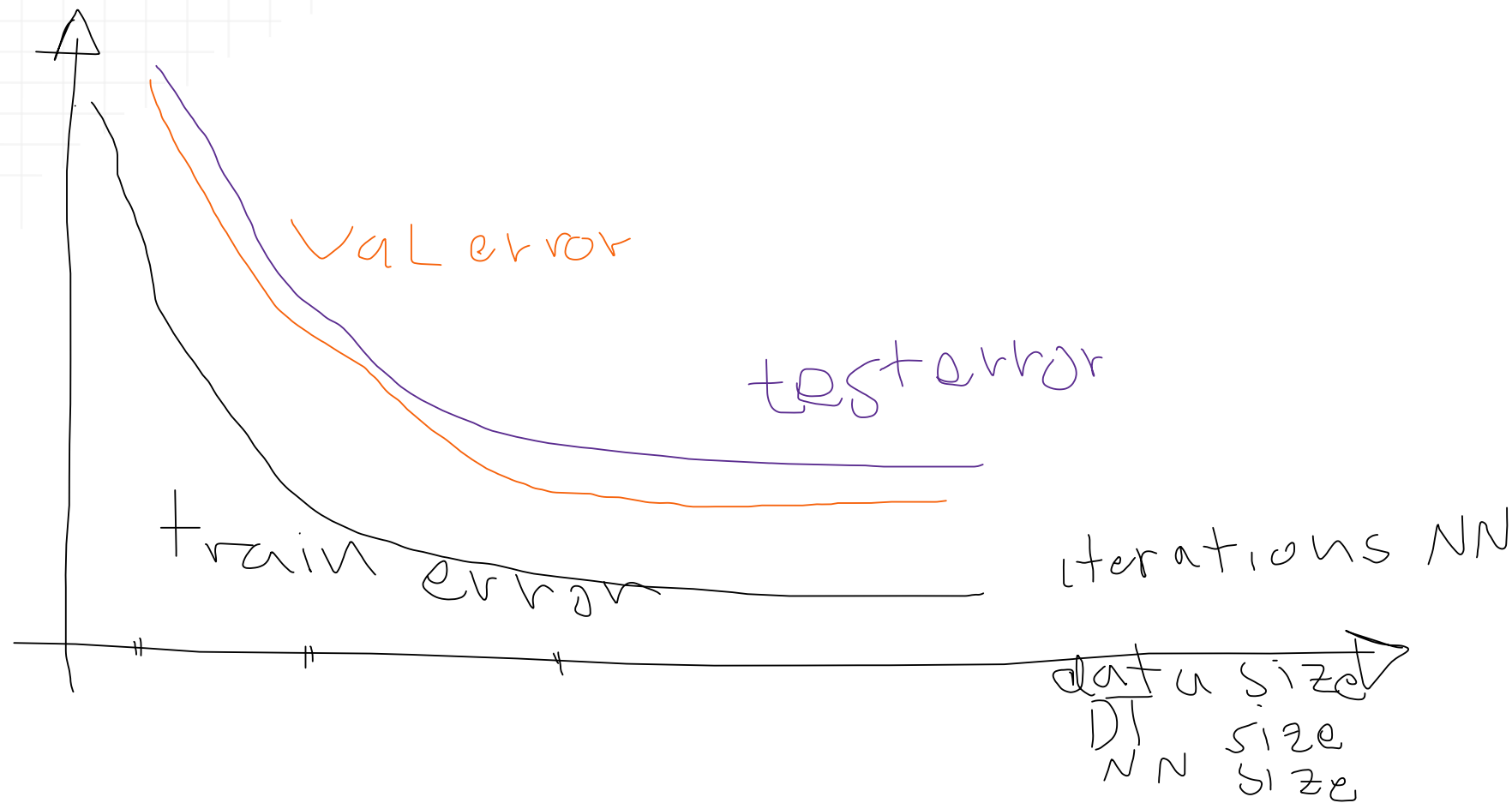


Errors

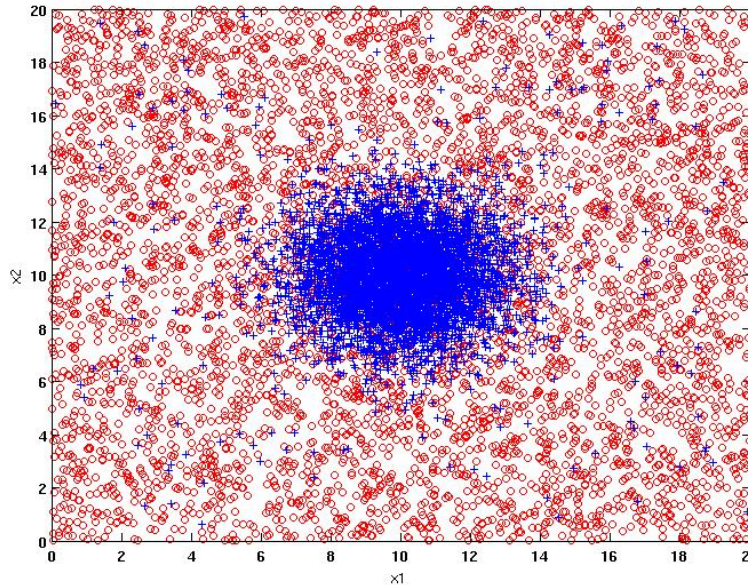
- **Training errors:** Errors committed on the training set
- **Test errors:** Errors committed on the test set
- **Generalization errors:** Expected error of a model over random selection of records from same distribution



error



Example Data Set



Two class problem:

+ : 5400 instances

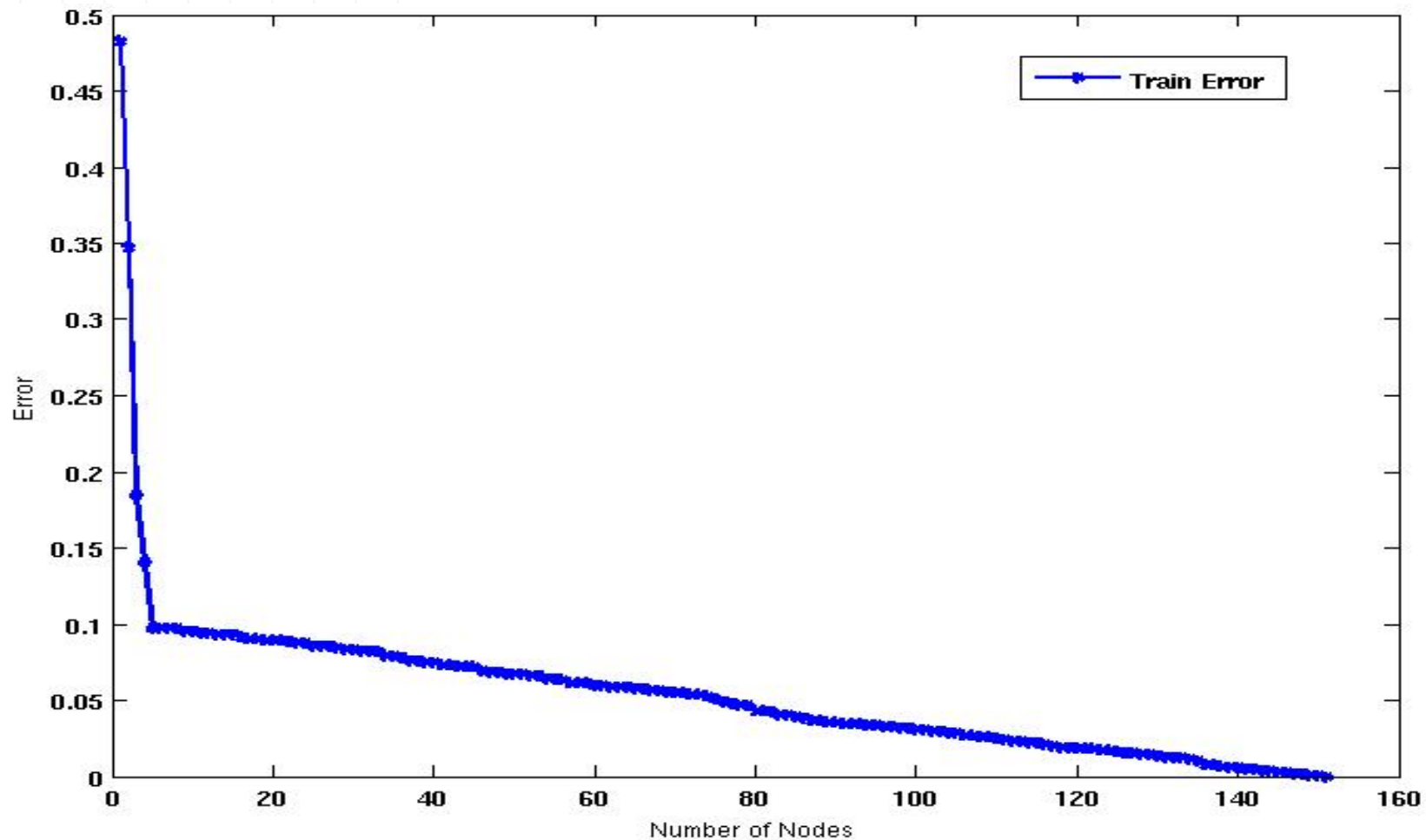
- 5000 instances generated from a Gaussian centered at (10,10)
- 400 noisy instances added

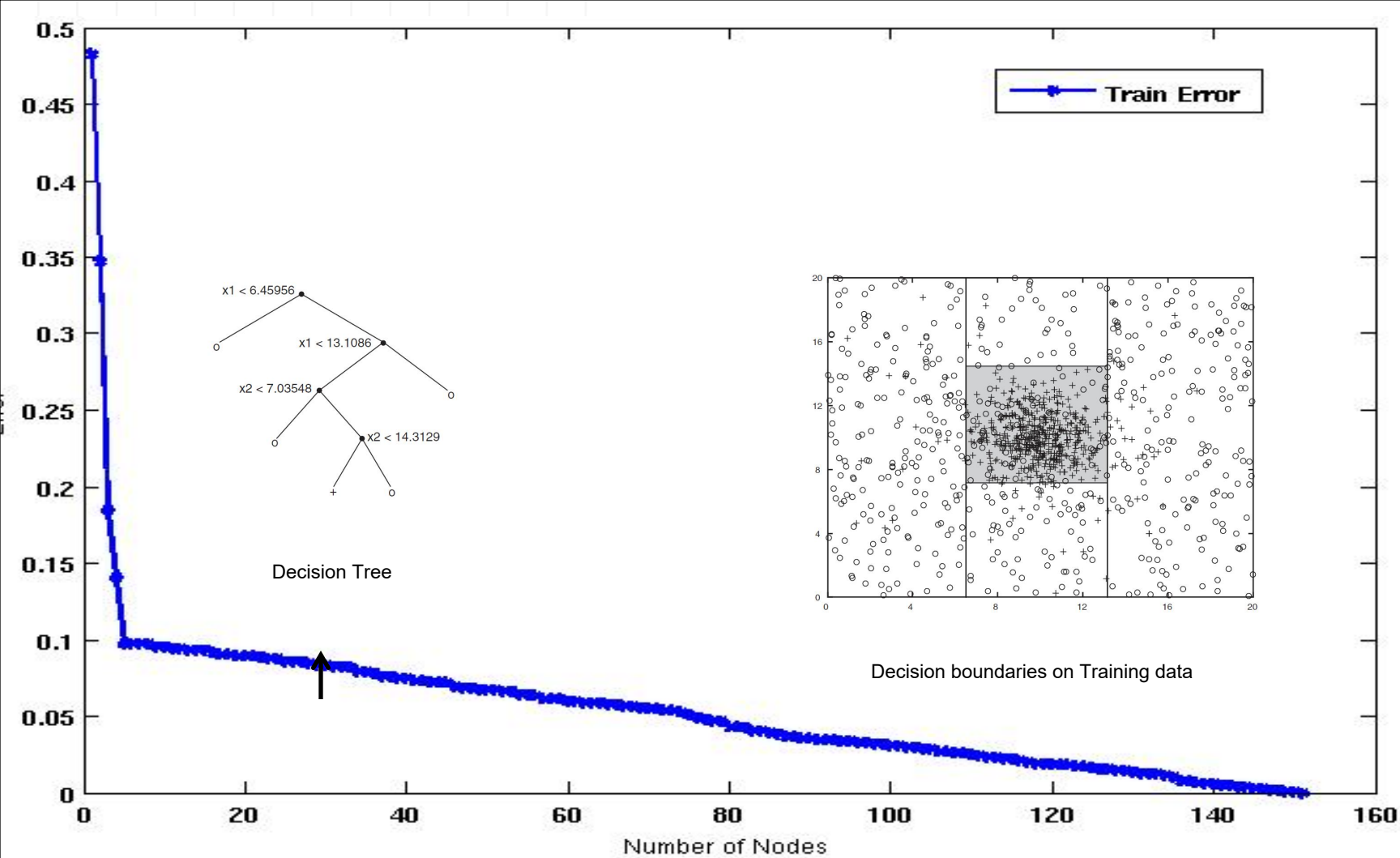
o : 5400 instances

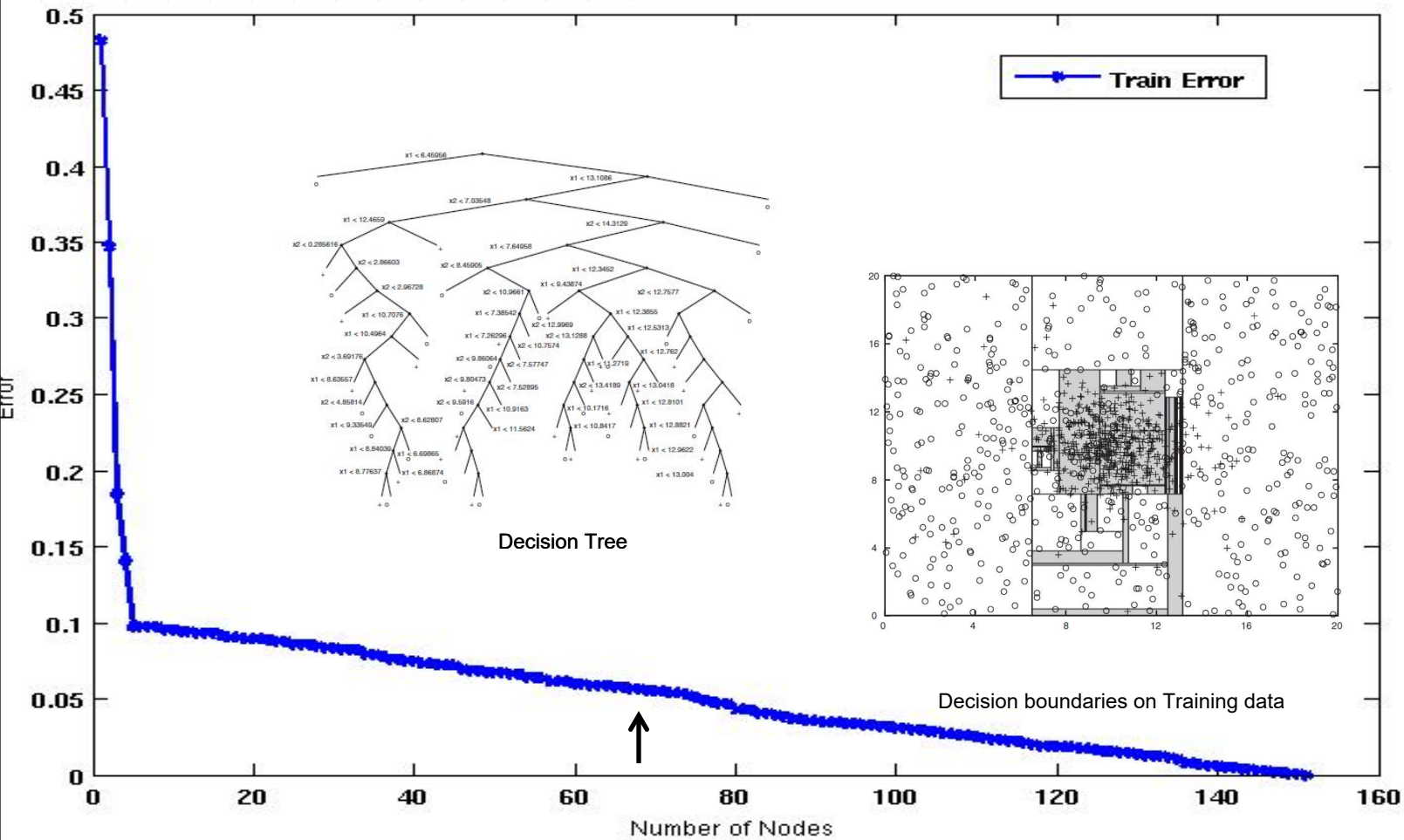
- Generated from a uniform distribution

10 % of the data used for training and 90% of the data used for testing

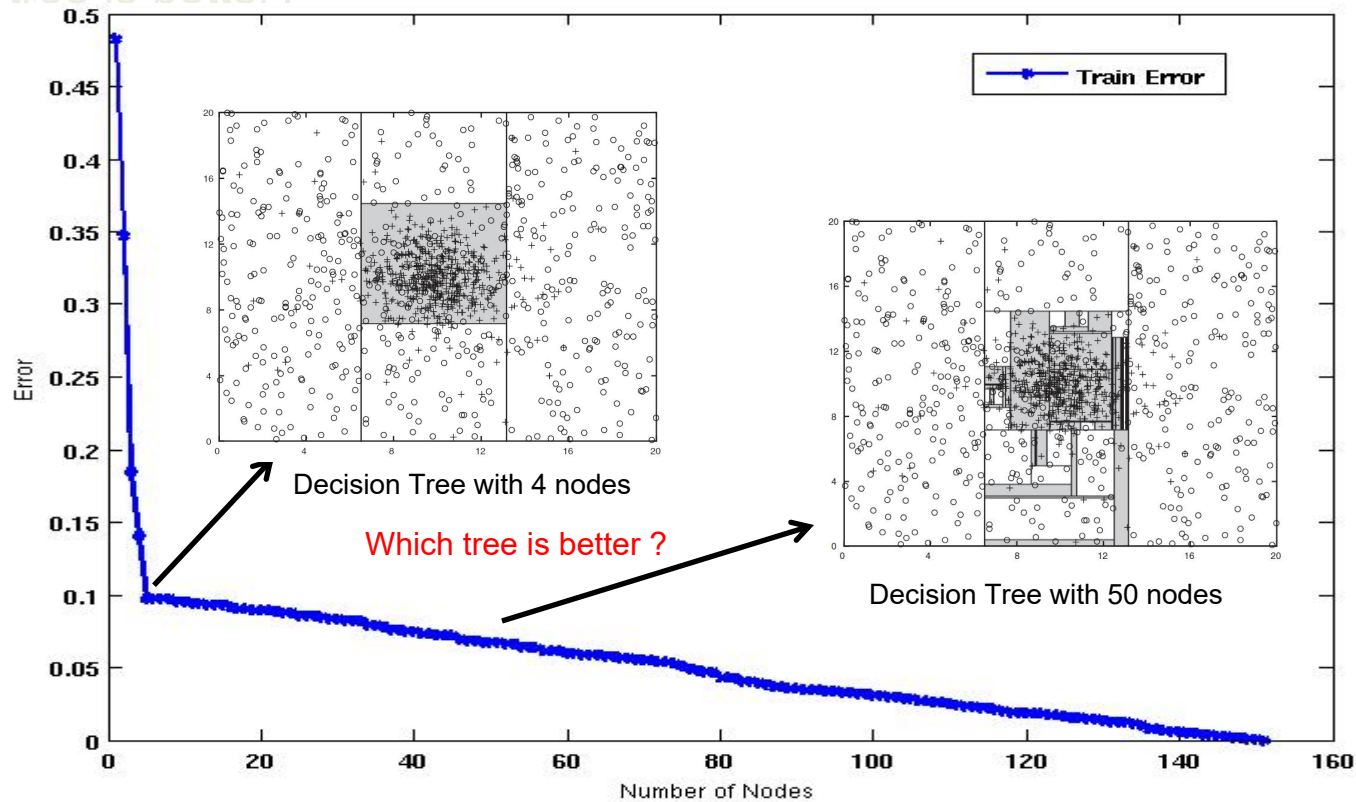
Increasing number of nodes in Decision Trees



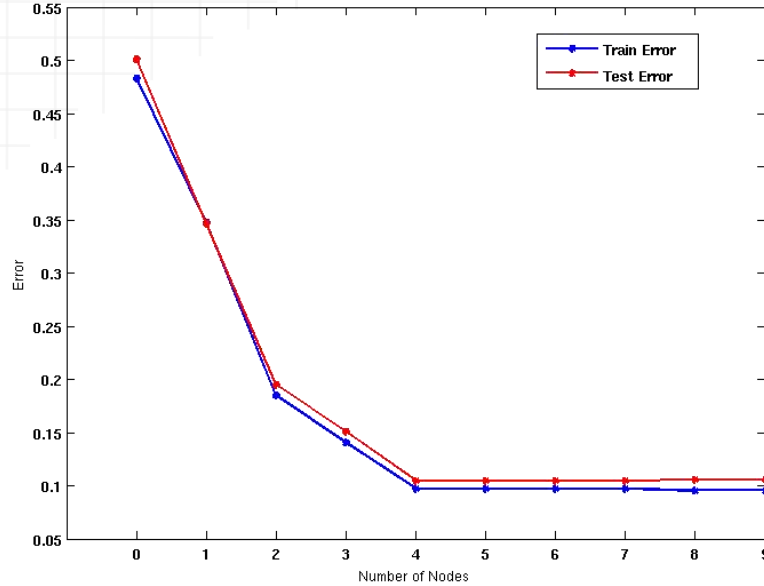




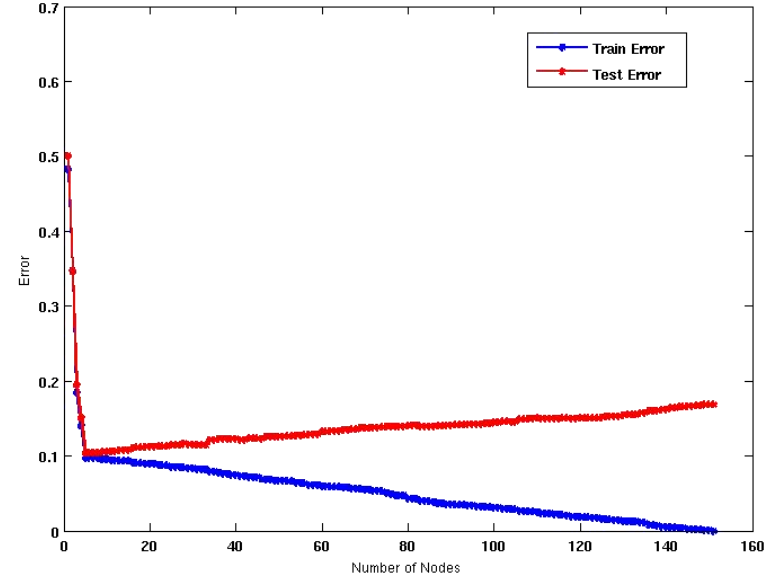
Which tree is better?



Model Underfitting and Overfitting



•As the model becomes more and more complex, test errors can start increasing even though training error may be decreasing

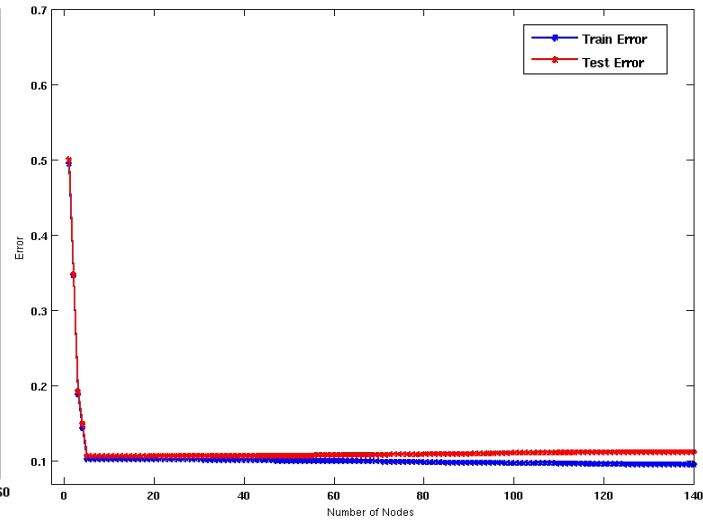
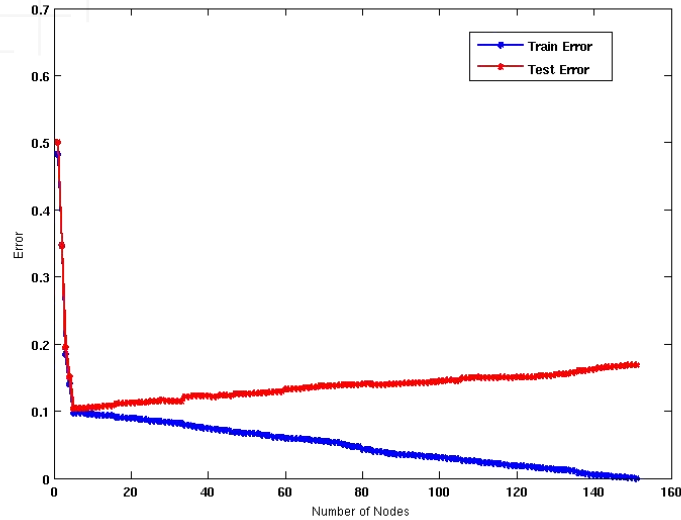


NN size
Iterations in NN
DT size

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

Overfitting: when model is too complex, training error is small but test error is large

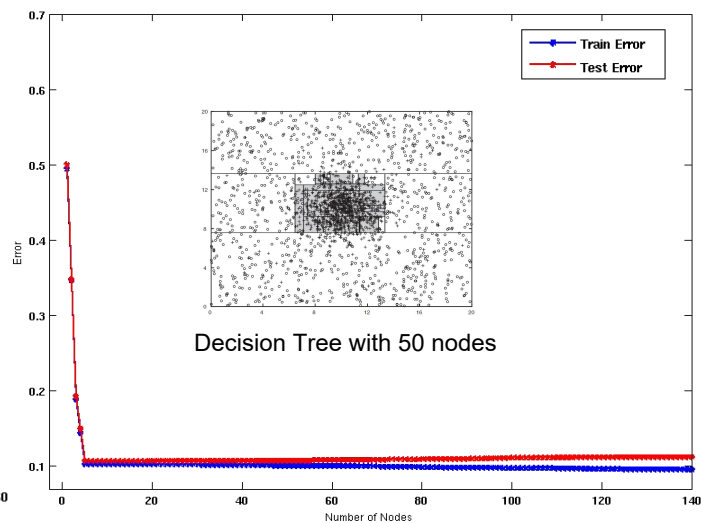
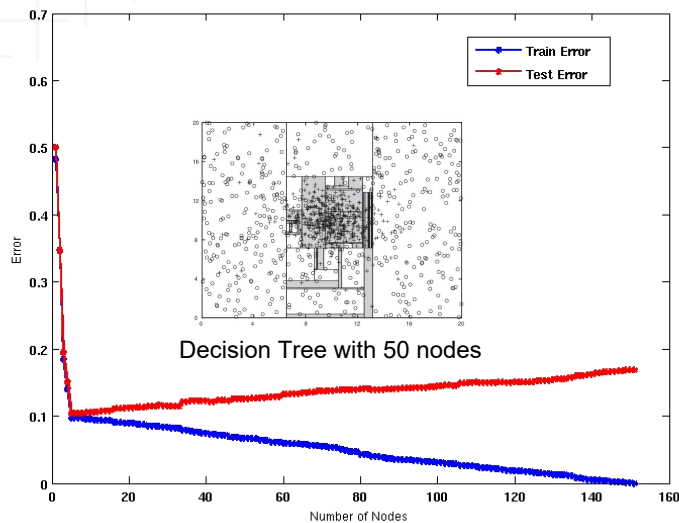
Model Overfitting – Impact of Training Data Size



Using twice the number of data instances

- Increasing the size of training data reduces the difference between training and testing errors at a given size of model

Model Overfitting – Impact of Training Data Size



Using twice the number of data instances

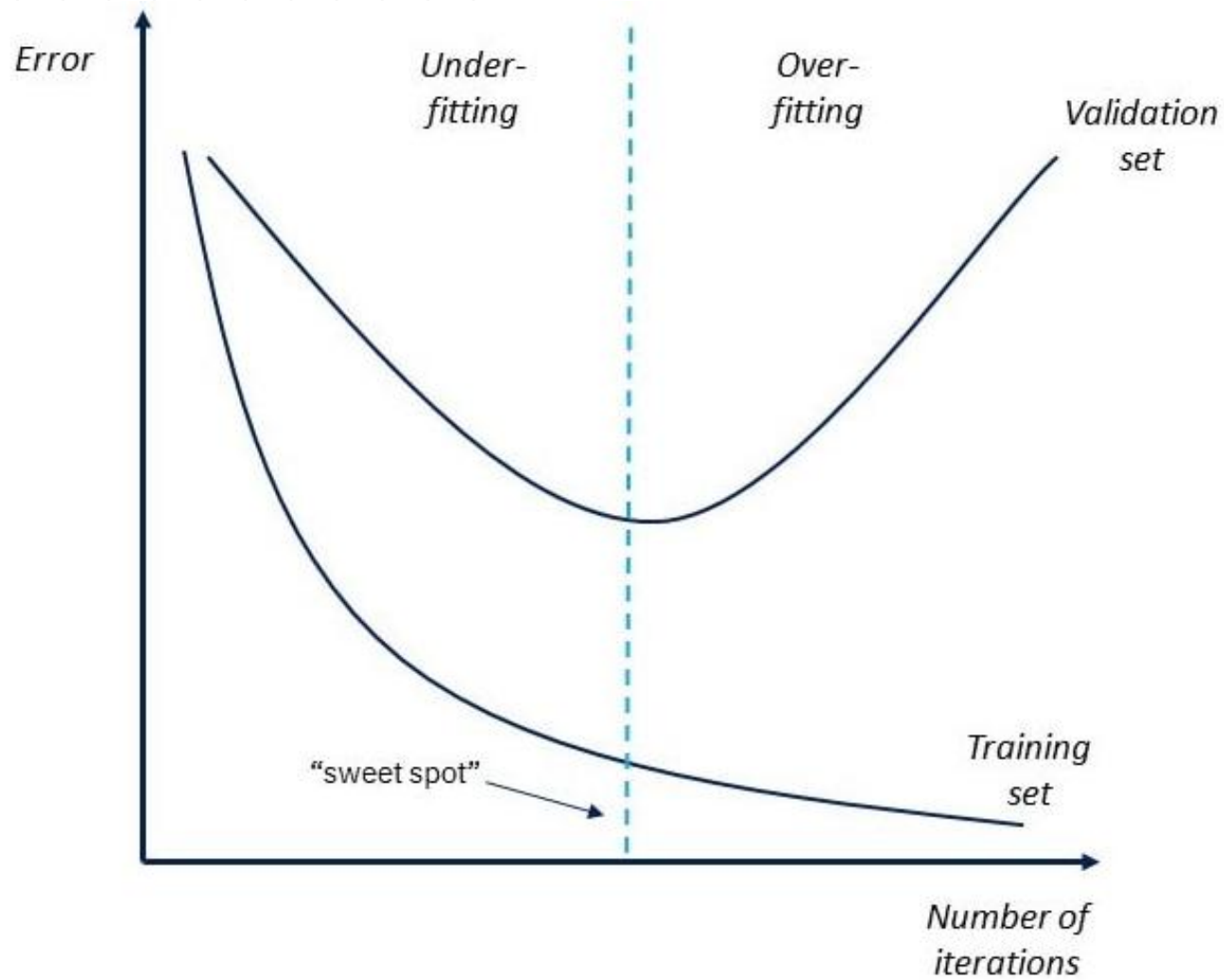
- Increasing the size of training data reduces the difference between training and testing errors at a given size of model

Reasons for Model Overfitting

- Not enough training data
- Over training (train NN with large number of iterations)

Model Selection

- Performed during model building
- Purpose is to ensure that model is not overly complex (to avoid overfitting)
- Need to estimate generalization error
 - Using Validation Set
 - Incorporating Model Complexity



Model Selection:

Using Validation Set

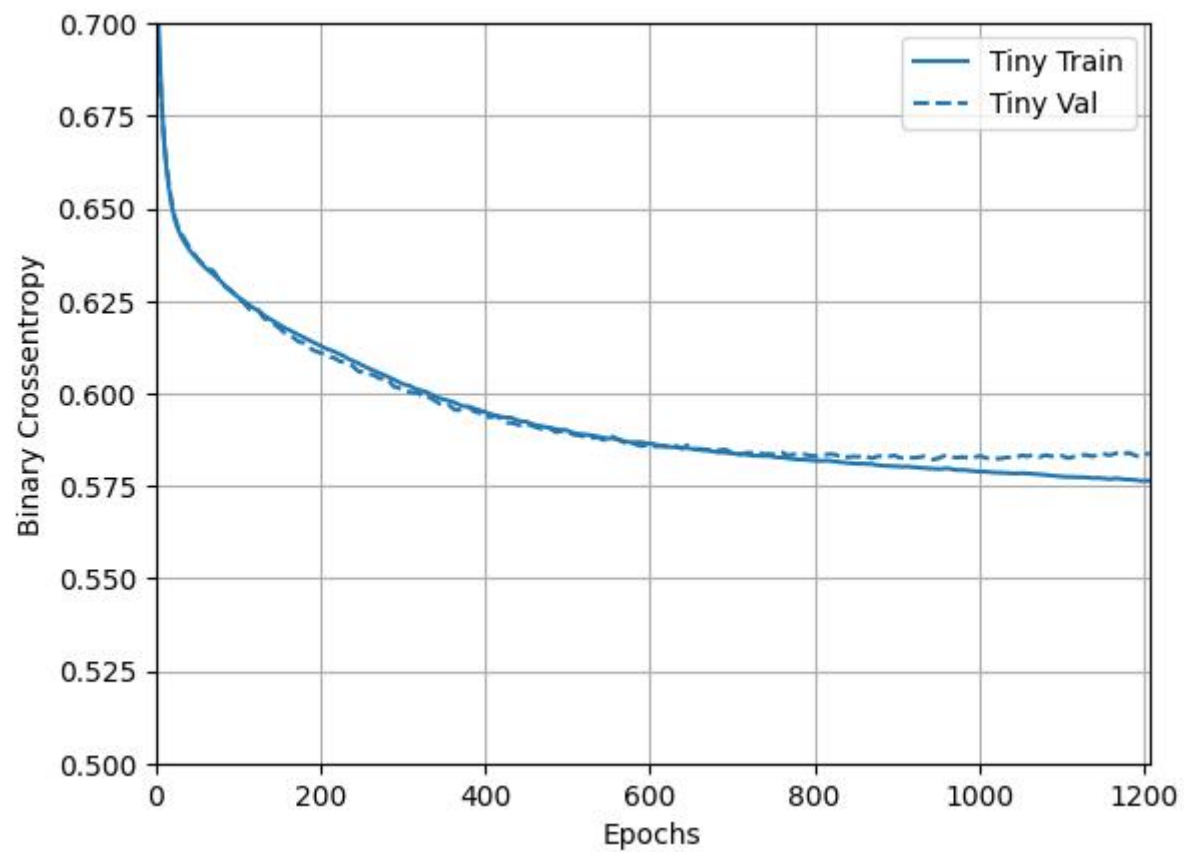
- Divide training data into two parts:
 - Training set:
 - use for model building
 - Validation set:
 - use for estimating generalization error
 - Note: validation set is not the same as test set
- Drawback:
 - Less data available for training

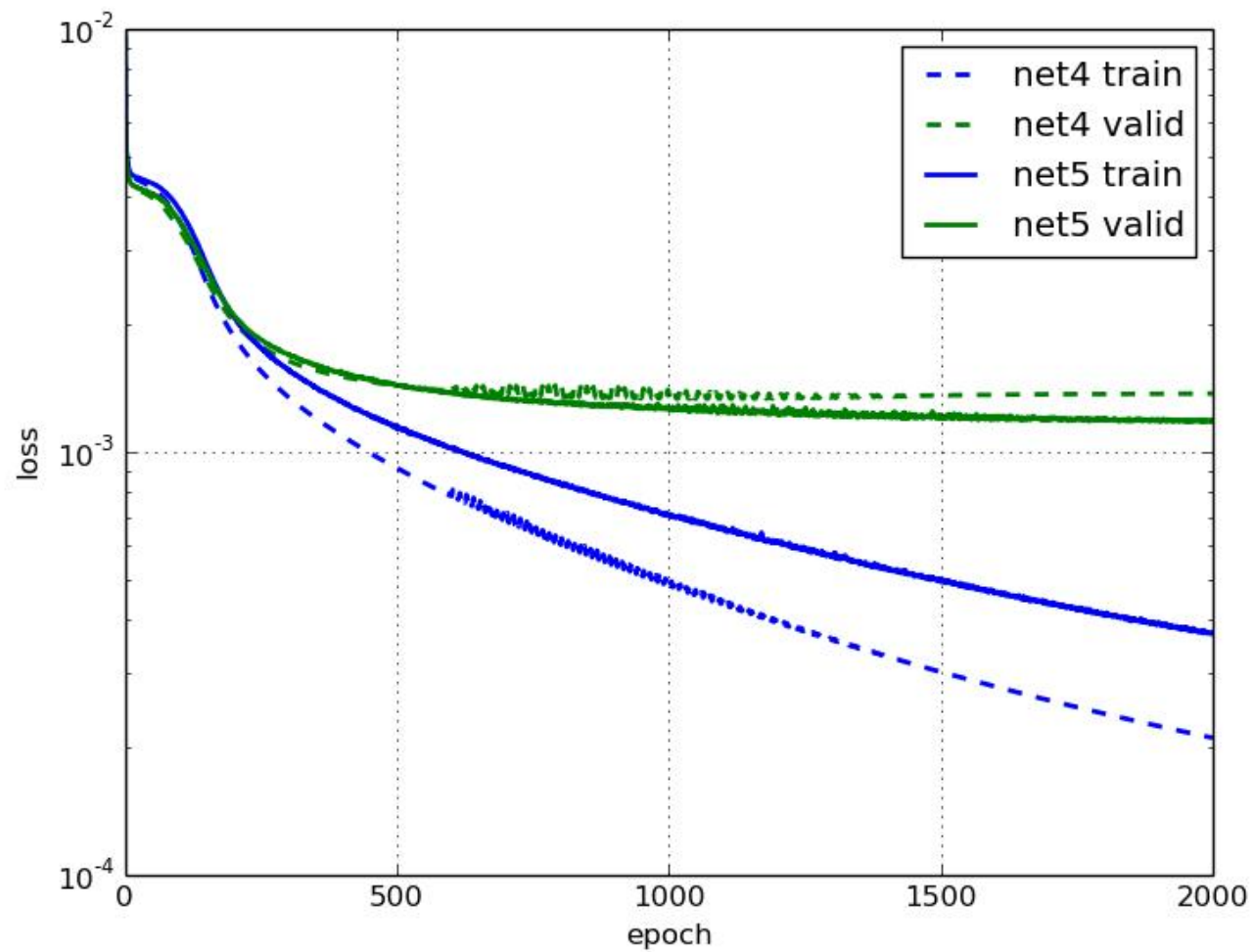
Model Evaluation

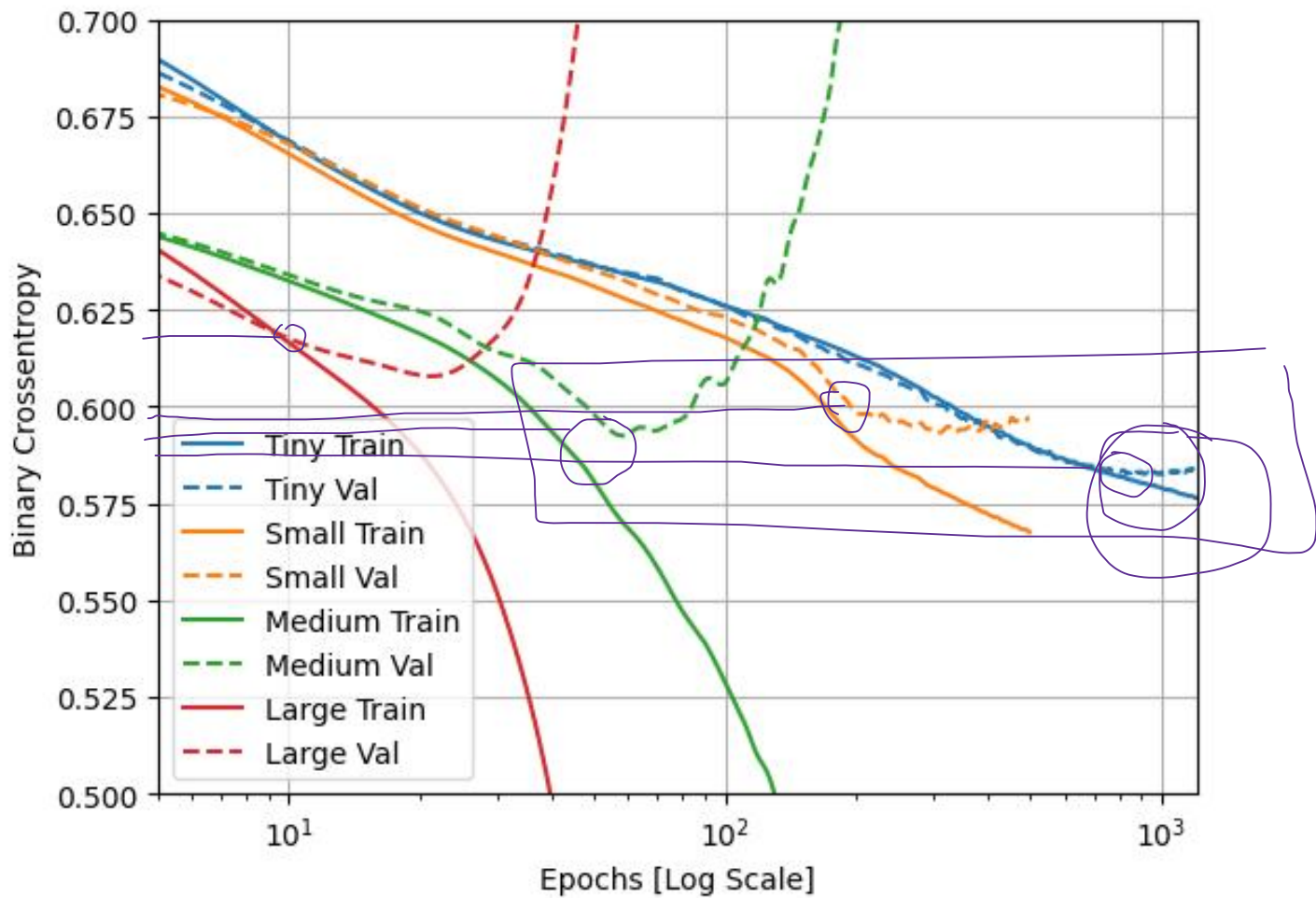
- Purpose:
 - To estimate performance of classifier on previously unseen data (test set)
- Holdout
 - Reserve $k\%$ for training and $(100-k)\%$ for testing
 - Random subsampling: repeated holdout
- Cross validation
 - Partition data into k disjoint subsets
 - k -fold: train on $k-1$ partitions, test on the remaining one
 - Leave-one-out: $k=n$

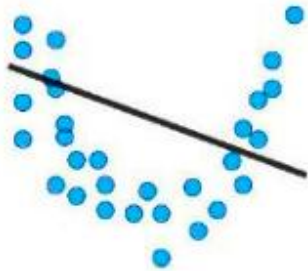
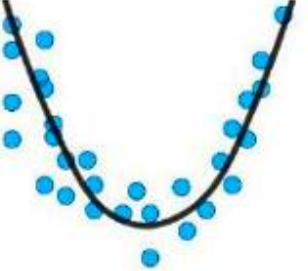
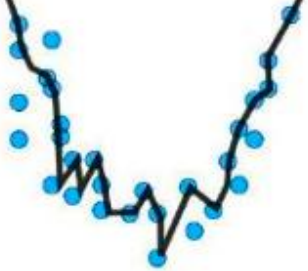
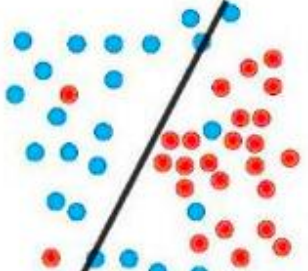
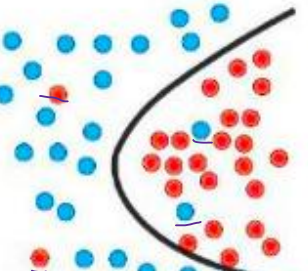
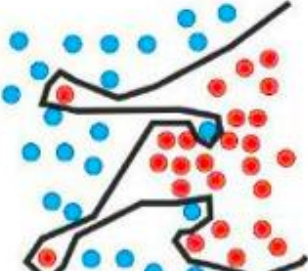
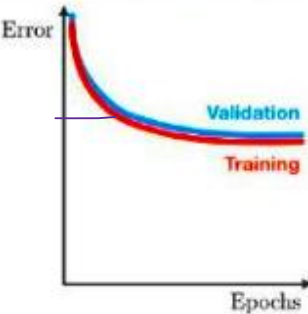
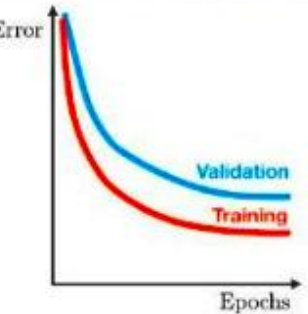
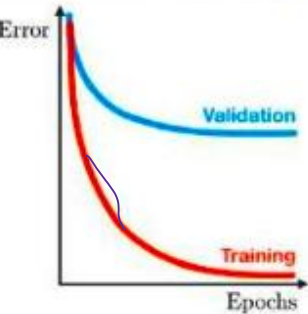
Overfit and underfit for NN

- Build 4 NNs
 - Tiny
 - Small
 - Medium
 - Large
- Draw training & validation curves for the 4 networks







	Under-fitting	Optimal-fitting	Over-fitting
✓ Regression			
✓ Classification			
✓ Deep learning			

Two types of Evaluation

- Classification Evaluation
- Regression Evaluation



Classification Evaluation

Confusion Matrix

- A confusion matrix shows the number of **correct** and **incorrect predictions** made by the classification model compared to the actual outcomes (target value) in the data.
- The matrix is $N \times N$, where N is the number of target values (classes).
- Performance of such models is commonly evaluated using the data in the matrix.
- The following table displays a 2x2 confusion matrix for two classes (Positive and Negative).

Confusion Matrix

Confusion Matrix		Target (Actual)			
		Positive	Negative		
Model Prediction	Positive	a (correct)	b (incorrect)	<i>Positive Predictive Value</i> <i>precision</i>	$a/(a+b)$
	Negative	c (incorrect)	d (correct)	<i>Negative Predictive Value</i>	$d/(c+d)$
		<i>Sensitivity</i> <i>recall</i>	<i>Specificity</i>	Accuracy = $(a+d)/(a+b+c+d)$	
		$a/(a+c)$	$d/(b+d)$		

- **Accuracy** : the proportion of the total number of predictions that were correct.
- **Positive Predictive Value** or **Precision** : the proportion of positive cases that were correctly identified.
- **Negative Predictive Value** : the proportion of negative cases that were correctly identified.
- **Sensitivity** or **Recall** : the proportion of actual positive cases which are correctly identified.
- **Specificity** : the proportion of actual negative cases which are correctly identified.

Example

Confusion Matrix		Target			
		Positive	Negative		
Model	Positive	70	20	<i>Positive Predictive Value</i>	0.78
	Negative	30	80	<i>Negative Predictive Value</i>	0.73
		<i>Sensitivity</i>	<i>Specificity</i>	Accuracy = 0.75	
		0.70	0.80		

$$N = \# \text{ of instances} = a + b + c + d = 70 + 20 + 30 + 80 = 200$$

		A		
		A	B	C
P	A	78		
	B	3		
	C	5		

Model Evaluation - Regression

		P	A
u	l2	300	250
	:		
			$\frac{\sum a}{n}$

Model Evaluation - Regression

- After building a number of different regression models, there is a wealth of criteria by which they can be evaluated and compared.
- **Root Mean Squared Error**
 - RMSE is a popular formula to measure the error rate of a regression model. However, it can only be compared between models whose errors are measured in the same units.

$$RMSE = \sqrt{\frac{\sum_{i=1}^n (p_i - a_i)^2}{n}}$$

Model Evaluation - Regression

- **Relative Squared Error**
 - Unlike RMSE, the relative squared error (RSE) can be compared between models whose errors are measured in the different units.

$$RSE = \frac{\sum_{i=1}^n (p_i - a_i)^2}{\sum_{i=1}^n (\bar{a} - a_i)^2}$$

Model Evaluation - Regression

- **Mean Absolute Error**
 - The mean absolute error (MAE) has the same unit as the original data, and it can only be compared between models whose errors are measured in the same units. It is usually similar in magnitude to RMSE, but slightly smaller.

$$MAE = \frac{\sum_{i=1}^n |p_i - a_i|}{n}$$

Model Evaluation - Regression

- **Relative Absolute Error**
 - Like RSE , the relative absolute error (RAE) can be compared between models whose errors are measured in the different units.

$$RAE = \frac{\sum_{i=1}^n |p_i - a_i|}{\sum_{i=1}^n |\bar{a} - a_i|}$$