CDS503: Machine Learning

Week 6: Designing Machine Learning Experiments

DR. JASY LIEW SUET YAN

SCHOOL OF COMPUTER SCIENCES

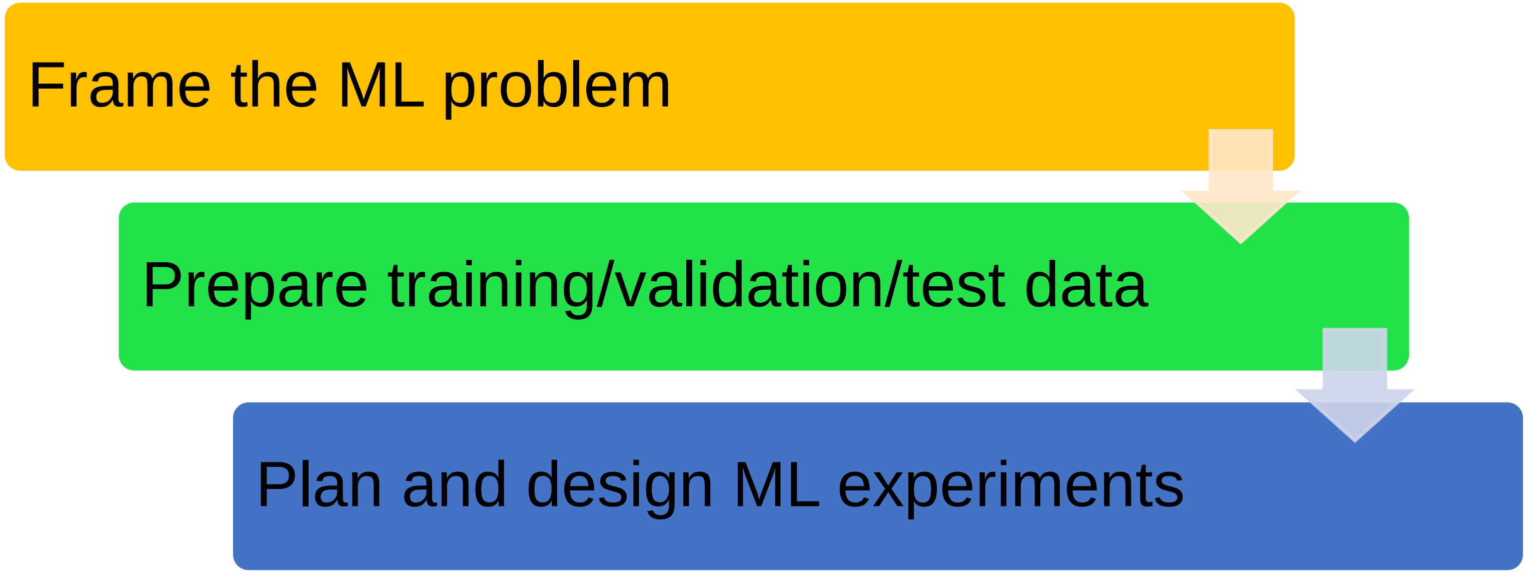
UNIVERSITI SAINS MALAYSIA (USM)

Outline

> *“Numbers have an important story to tell. They rely on you to give them a voice.”—Stephen Few*

- Designing machine learning experiments
- Model selection
- Bias-variance tradeoff
- ML diagnostic

ML Experiment Design



Frame the ML problem

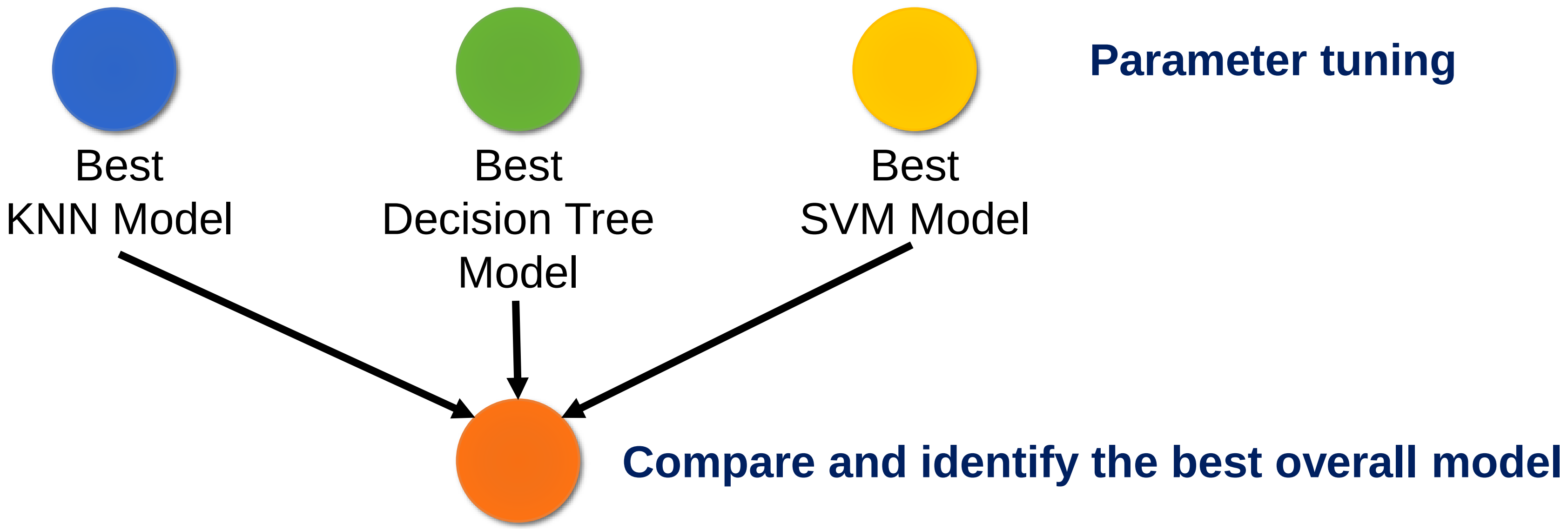
Prepare training/validation/test data

Plan and design ML experiments

- Select learning algorithm
- Tune parameters
- Select features
- Sample training and test data
- Test generalization error

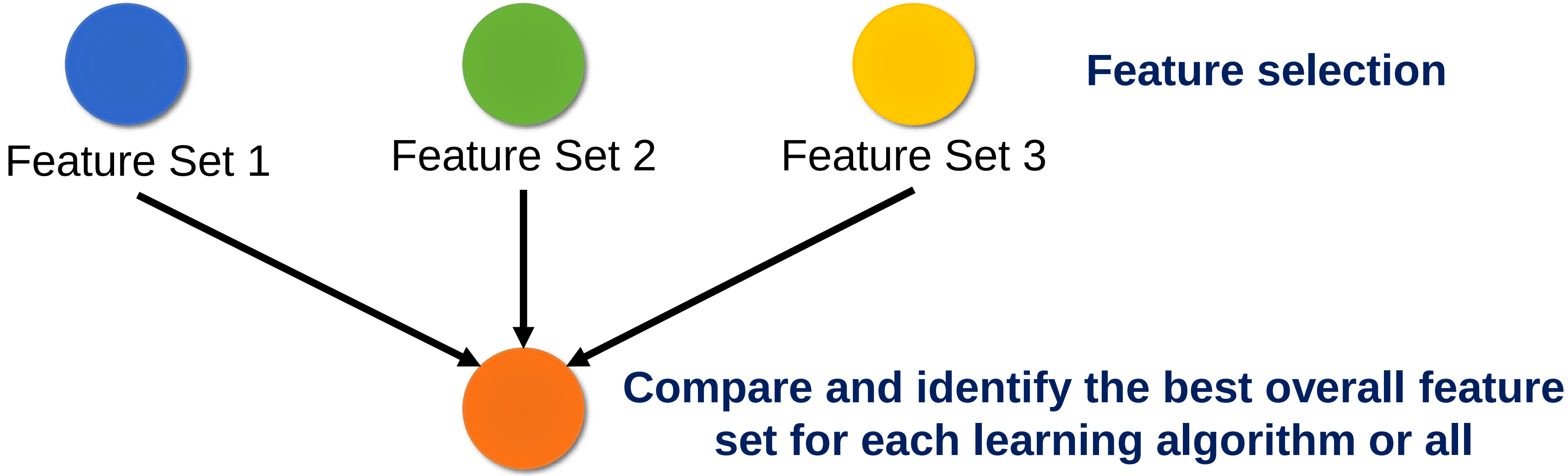
ML Experiment Design

- **Example 1:** Find the best parameter setting configuration for each learning algorithm. Compare the “best” model from each learning algorithm.



ML Experiment Design

- **Example 2:** Find the best set of features across different learning algorithms. Do this “best” set of features consistently result in the best performance across different learning algorithms?


Feature Set 1
Feature Set 2
Feature Set 3
Feature selection
Compare and identify the best overall feature set for each learning algorithm or all

Data Description

- Breast cancer data set: Medical data from 681 women (instances) who has potentially cancerous tumors (12 attributes)

Class attribute (1)
1: Tumor is malignant (238)
0: Tumor is benign (443)

Other attributes (2)
PID: Patient ID
Date: Diagnosis Date

Predictor attributes (9)
Adhes - marginal adhesion
BNucl - bare nuclei
Chrom - bland chromatin
Epith - epithelial cell size
Mitos – mitoses
NNucl - normal nucleoli
Thick - clump thickness
UShap - cell shape uniformity
USize - cell size uniformity

* A predictor is assigned the value 1 if it is normal and the value 10 if it is most abnormal

Prepare Data

Remove attributes. Not useful predictors.

PID	Date	Adhes	BNucl	Chrom	Epith	Mitos	NNucl	Thick	UShape	USize	Class
1	01/03/2007	1	1	3	2	1	1	5	1	1	0
2	12/12/2005	5	10	3	7	1	2	5	4	4	0
3	14/08/2016	1	2	3	2	1	1	3	1	1	0
4	02/02/2001	1	4	3	3	1	7	6	8	8	0
5	14/11/2014	3	1	3	2	1	1	4	1	1	0
6	22/09/2011	8	10	9	7	1	7	8	10	10	1
7	18/05/2015	7	8	8	9	2	8	8	6	6	1
8	27/04/2011	5	9	9	10	2	6	7	9	9	1
9	19/02/2003	8	6	8	3	1	3	5	10	10	1
10	25/07/2011	10	5	6	6	4	4	10	7	7	1

Class attribute

Predictor attributes

Model Selection

- Standard technique to evaluate a hypothesis

Adhes	BNucl	Chrom	Epith	Mitos	NNucl	Thick	UShape	USize	Class
1	1	3	2	1	1	5	1	1	0
5	10	3	7	1	2	5	4	4	0
1	2	3	2	1	1	3	1	1	0
1	4	3	3	1	7	6	8	8	0
3	1	3	2	1	1	4	1	1	0
8	10	9	7	1	7	8	10	10	1
7	8	8	9	2	8	8	6	6	1
5	9	9	10	2	6	7	9	9	1
8	6	8	3	1	3	5	10	10	1
10	5	6	6	4	4	10	7	7	1

Training set (70%)
Learn parameter
from training data
(hypothesis function)

Split dataset

- *Should be randomly sorted*

Test set (30%)
Compute test set error

Model Selection

- Train / validation / test sets

Adhes	BNucl	Chrom	Epith	Mitos	NNucl	Thick	UShape	USize	Class
1	1	3	2	1	1	5	1	1	0
5	10	3	7	1	2	5	4	4	0
1	2	3	2	1	1	3	1	1	0
1	4	3	3	1	7	6	8	8	0
3	1	3	2	1	1	4	1	1	0
8	10	9	7	1	7	8	10	10	1
7	8	8	9	2	8	8	6	6	1
5	9	9	10	2	6	7	9	9	1
8	6	8	3	1	3	5	10	10	1
10	5	6	6	4	4	10	7	7	1

Training set (60%)
Learn parameter
from training data
(hypothesis function)

Split dataset

- *Should be randomly sorted*

Validation set (20%)
Tune parameters (hyperparameters)

Test set (20%)
Compute test set error

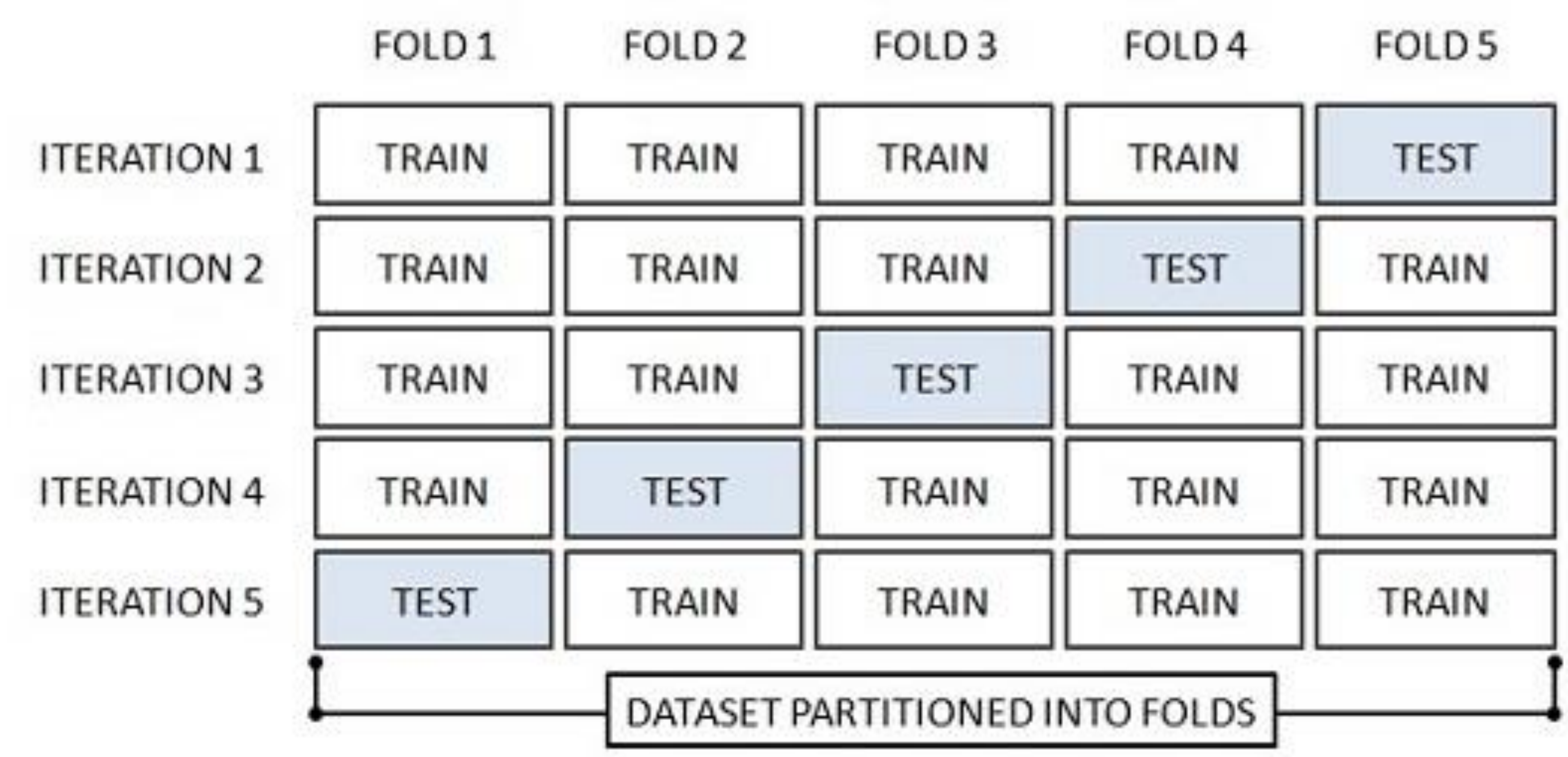
Model Selection

- Calculate 3 separate error values
  - **Training error:** Optimize the parameters in hypothesis function using training set
  - **Validation error:** Find the best hyperparameters with the least error using validation set
  - **Test/Generalization error:** Estimate the generalization error using the test set

Test Options

- Set up train and test sets
  - **Percentage split**: Splits the data and separates x% of the data for training and the rest for testing
  - **Supplied test set**: Prepare own external file as training set
  - **K-fold cross validation**: Data set is divided into *k* subsets. Each time, one of the *k* subsets is used as the test set and the other *k-1* subsets are put together to form a training set. Then the average error across all *k* trials is computed.

Cross Validation



	FOLD 1	FOLD 2	FOLD 3	FOLD 4	FOLD 5
ITERATION 1	TRAIN	TRAIN	TRAIN	TRAIN	TEST
ITERATION 2	TRAIN	TRAIN	TRAIN	TEST	TRAIN
ITERATION 3	TRAIN	TRAIN	TEST	TRAIN	TRAIN
ITERATION 4	TRAIN	TEST	TRAIN	TRAIN	TRAIN
ITERATION 5	TEST	TRAIN	TRAIN	TRAIN	TRAIN

DATASET PARTITIONED INTO FOLDS

5-fold cross validation

Confusion Matrix

N = 10	Predicted Class			Total
Actual Class		1	0	
	1: Malignant	**3 (TP)**	**2 (FN)**	5
	0: Benign	**1 (FP)**	**4 (TN)**	5
Total		4	6	10

1: Malignant (Positive Class)
0: Benign (Negative Class)

True Positives (TP): Actual class of the data point was TRUE and the predicted is also TRUE (positive class)
Ex: The case where a tumor is malignant and the model classifying the tumor as malignant

True Negatives (TN): Actual class of the data point was FALSE and the predicted is also FALSE (negative class)
Ex: The case where the tumor is benign and the model classifying the tumor as benign

False Positives (FP): Actual class of the data point was FALSE and the predicted is TRUE.
Ex: A tumor being benign and the model classifying the tumor as malignant

False Negatives (FN): Actual class of the data point was TRUE and the predicted is FALSE.
Ex: Tumor is malignant and the model classifying the tumor as benign

Accuracy

Good measure when the classes in the data are nearly balanced.

Malignant = 5

Benign = 5

N = 10		Predicted Class		Total
		1	0	
Actual Class	1: Malignant	**3 (TP)**	**2 (FN)**	5
	0: Benign	**1 (FP)**	**4 (TN)**	5
Total		4	6	10

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

Accuracy

= (3 + 4) / (3 + 4 + 1 + 2)

= 7 / 10

= 0.7

Accuracy

Accuracy is 80% even though the classifier assigned all 10 instances as BENIGN (0)
Malignant = 2
Benign = 8

* NEVER be used as a measure when classes in the data are a majority of one class

N = 10	Predicted Class			Total
Actual Class		1	0	
	1: Malignant	**0 (TP)**	**2 (FN)**	2
	0: Benign	**0 (FP)**	**8 (TN)**	8
Total		0	10	10

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

Accuracy
= (0 + 8) / (0 + 8 + 0 + 2)
= 8 / 10
= 0.8

Precision

- Measures how good is the model at whatever it predicted
  - Example: Proportion of tumors predicted as malignant, which are actually malignant

$$\text{Precision} = \frac{TP}{TP+FP}$$

N = 10	Predicted Class			Total
Actual Class		1	0	
	1: Malignant	**0 (TP)**	**2 (FN)**	2
	0: Benign	**1 (FP)**	**7 (TN)**	8
Total		1	9	10

Precision
= 0 / (0 + 1)
= 0 / 1
= 0

Recall

- Measures how good is the model at picking the correct items
  - Example: Proportion of actual malignant tumors being predicted by the algorithm as being malignant

$$\text{Recall} = \frac{TP}{TP+FN}$$

Recall
= 0 / (0 + 2)
= 0 / 2
= 0

N = 10		Predicted Class		Total
Actual Class		1	0	
	1: Malignant	**0 (TP)**	**2 (FN)**	2
	0: Benign	**1 (FP)**	**7 (TN)**	8
Total		1	9	10

F-Measure (F1)

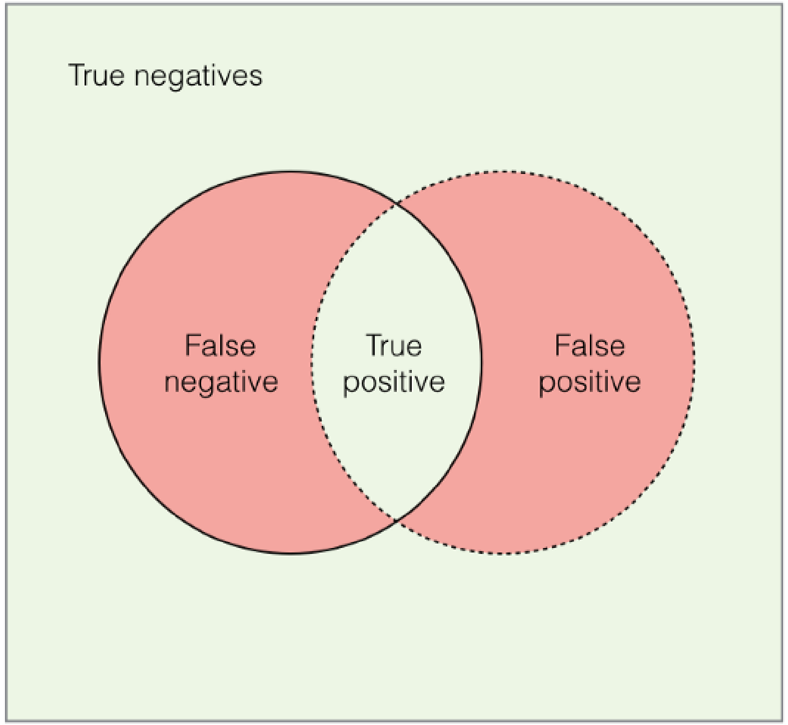
- Harmonic mean of precision and recall
- A single score that represents both precision and recall

$$\text{F1} = \frac{2 * Precision * Recall}{Precision + Recall}$$

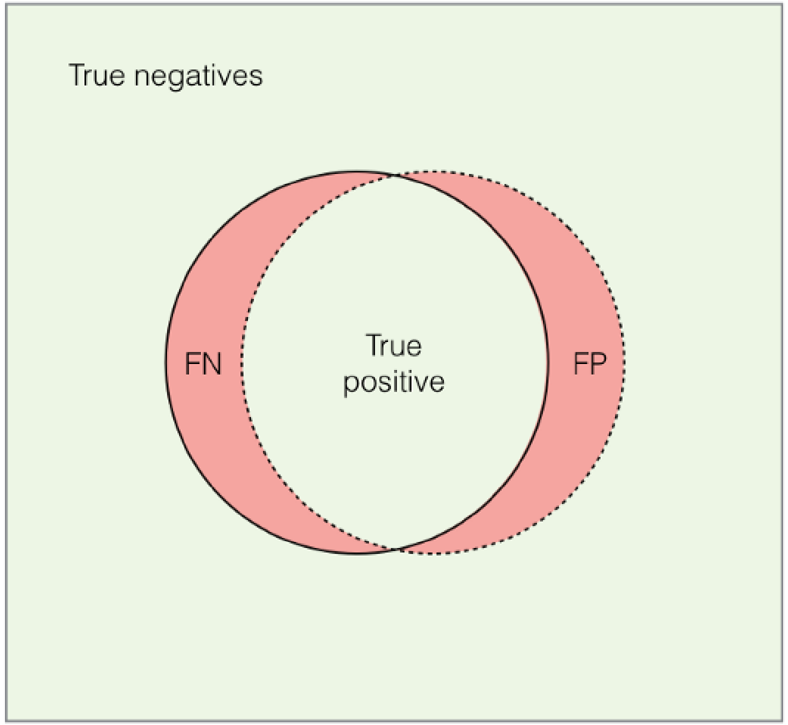
Say precision = 0.4, recall = 0.7
F1 = (2 * 0.4 * 0.7) / (0.4 + 0.7)
= 0.56 / 1.1
= 0.51

Precision and Recall

Low Recall, Low Precision

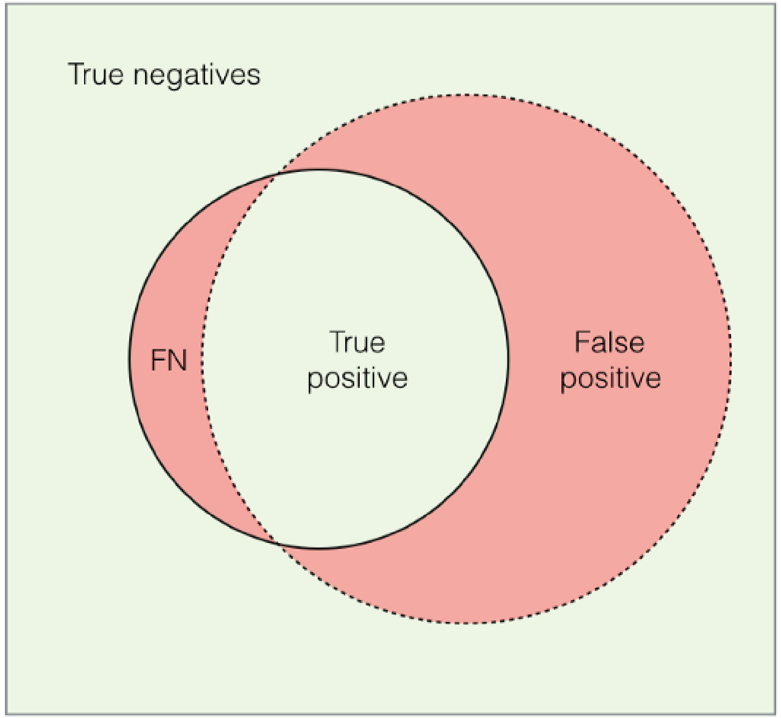


High Recall, High Precision

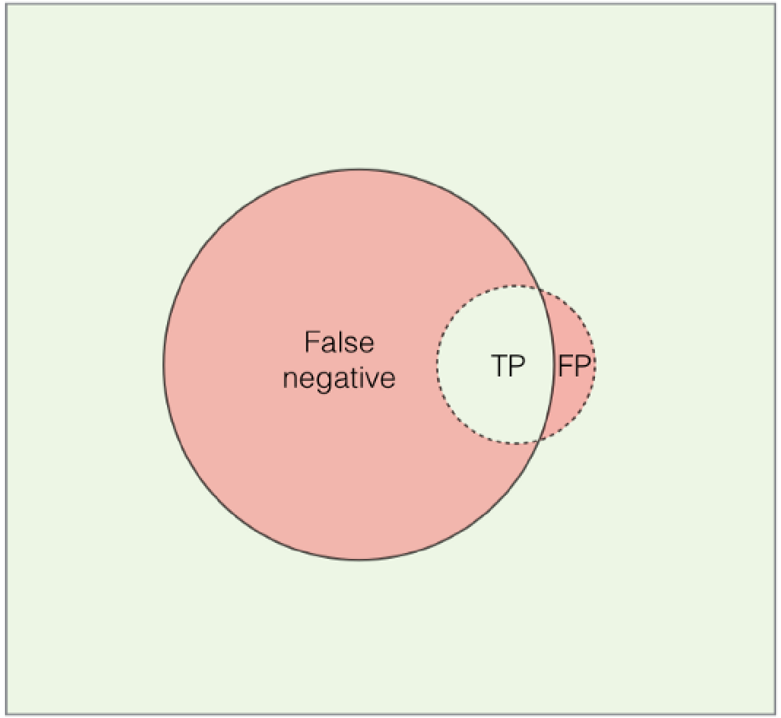


Precision and Recall

High Recall, Low Precision



Low Recall, High Precision



Precision and Recall

- Suppose we want to predict cancer only if very confident (avoid false positives)

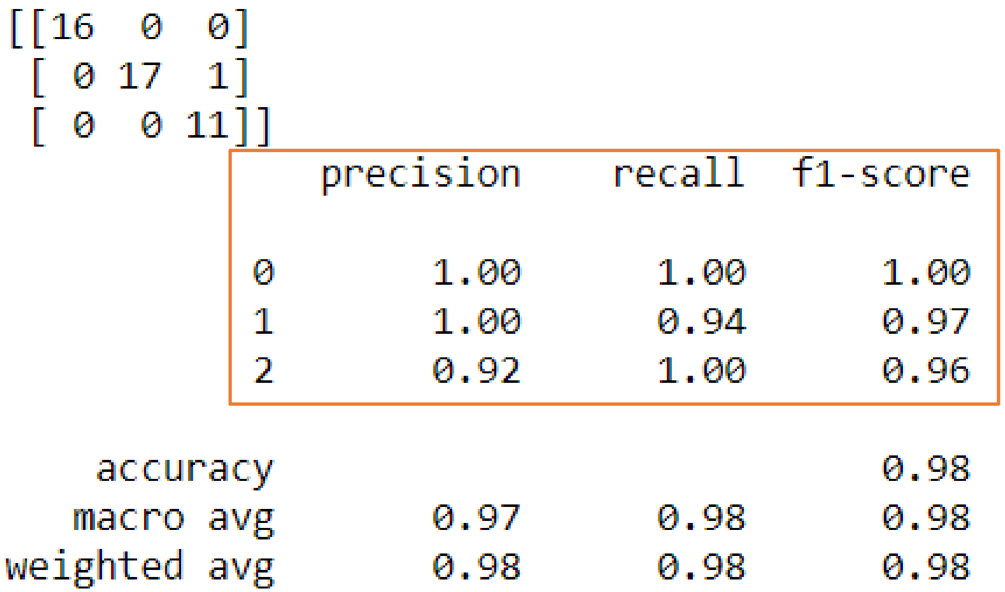
Higher precision, lower recall

- Suppose we want to avoid missing too many cases of cancer (avoid false negatives)

Higher recall, lower precision

Precision and Recall

Confusion Matrix
(Multiclass Classification)



```
[[16  0  0]
 [ 0 17  1]
 [ 0  0 11]]
              precision    recall  f1-score

           0       1.00      1.00      1.00
           1       1.00      0.94      0.97
           2       0.92      1.00      0.96

    accuracy                           0.98
   macro avg       0.97      0.98      0.98
weighted avg       0.98      0.98      0.98
```

For each class

Macro F1
= (1.00 + 0.97 + 0.96) / 3
= 0.98

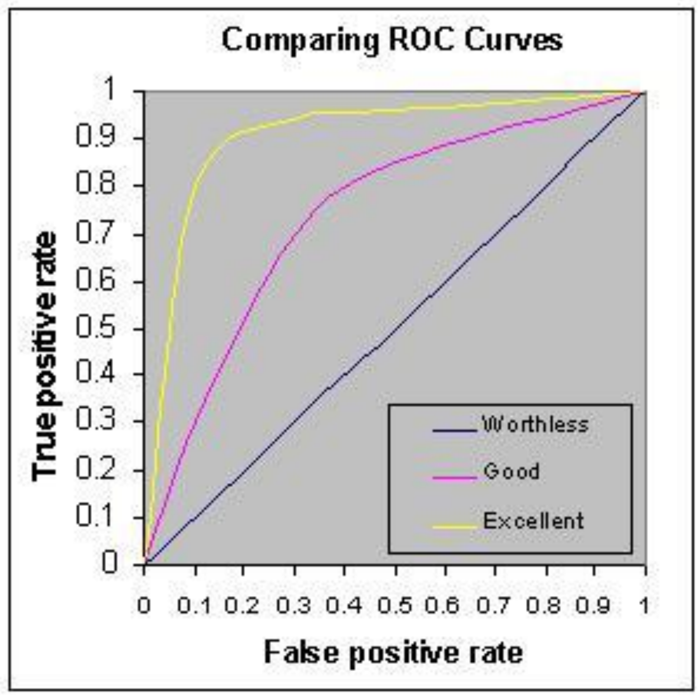
Weighted F1
= (1.00 * 50 + 0.97 * 50 + 0.96 * 50) / 150
= 0.98

Average across all classes

- Macro avg: Equal weights assigned to each class
- Weighted avg = Assign weights to each class based on number of samples

Area Under ROC

- ROC: Receiver Operating Characteristic
- Represent model’s ability to discriminate between positive and negative classes (for binary classification)



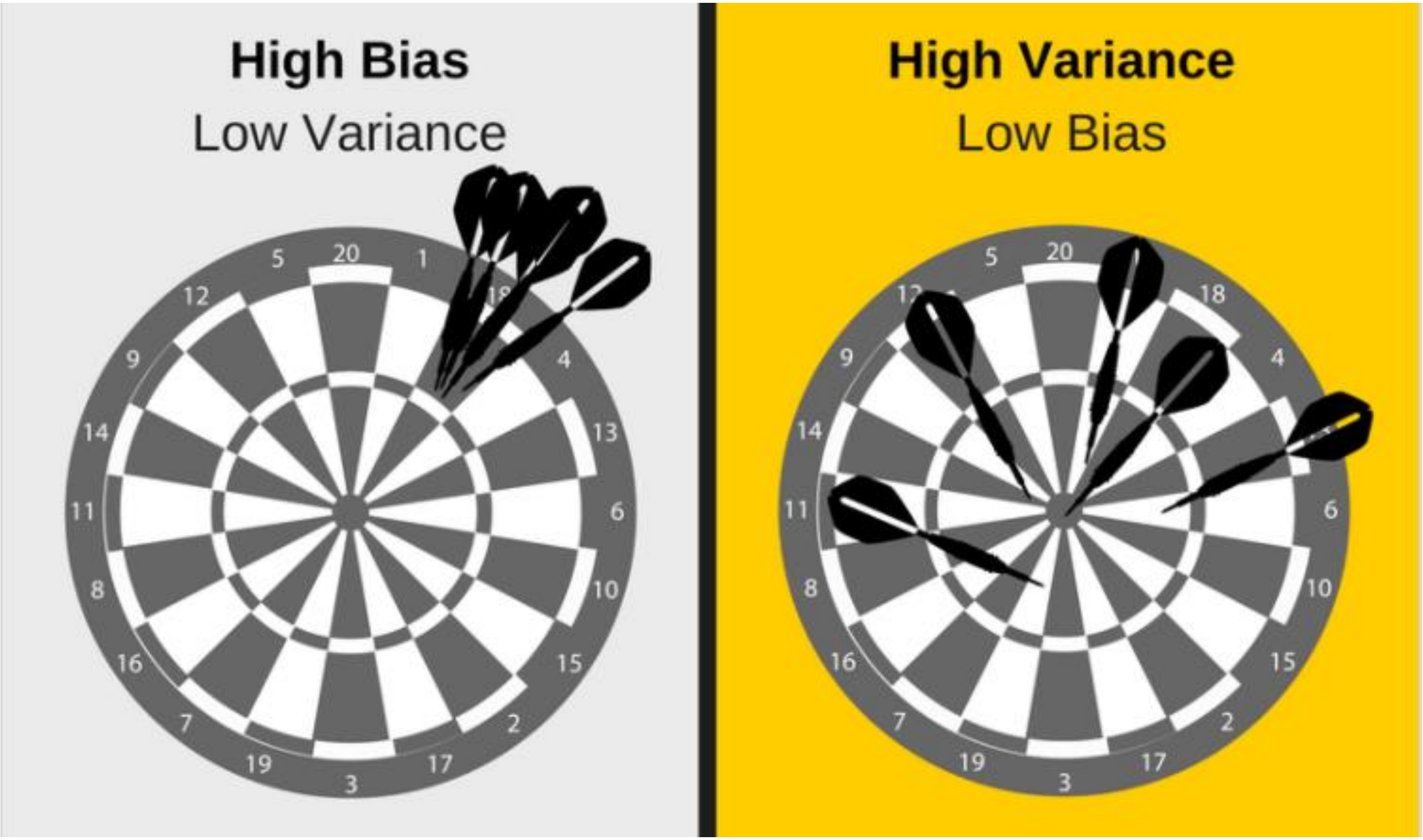
ROC area of 1.0 represents a model that made all predictions perfectly
ROC area of 0.5 represents a model as good as random

- X-axis: 1 – specificity (false positive rate = FP/(FP+TN))
- Y-axis: sensitivity (true positive rate = TP/(TP+FN))

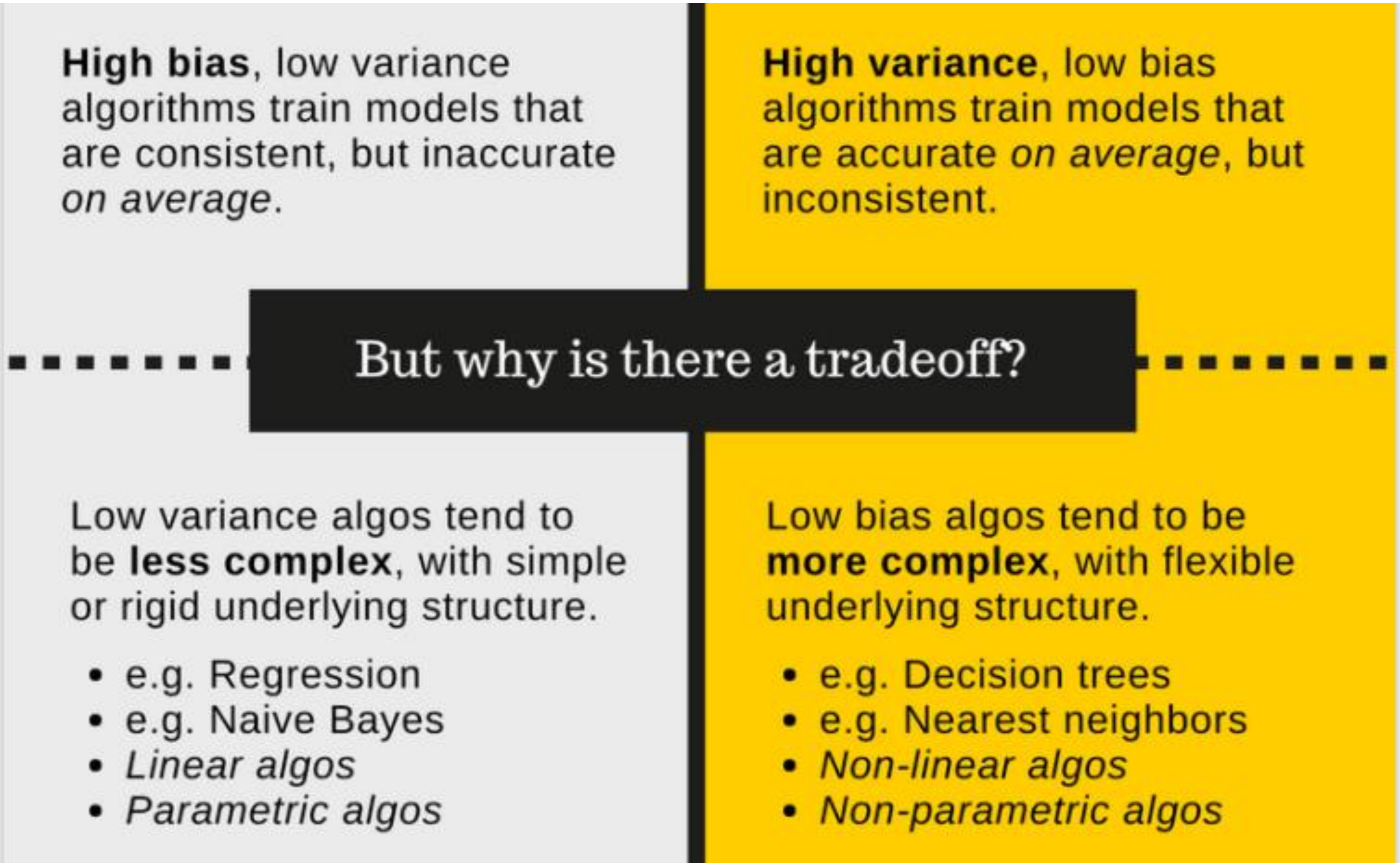
Bias-Variance Tradeoff

- **Bias**
  - How removed a model's predictions are from correctness
  - Occurs when an algorithm has limited flexibility to learn the true signal from a data set
- **Variance**
  - Degree to which these predictions vary between model iterations
  - Refers to algorithm’s sensitivity to specific sets of training data

Bias-Variance Tradeoff



Bias-Variance Tradeoff



High bias, low variance algorithms train models that are consistent, but inaccurate *on average*.

High variance, low bias algorithms train models that are accurate *on average*, but inconsistent.

But why is there a tradeoff?

Low variance algos tend to be **less complex**, with simple or rigid underlying structure.

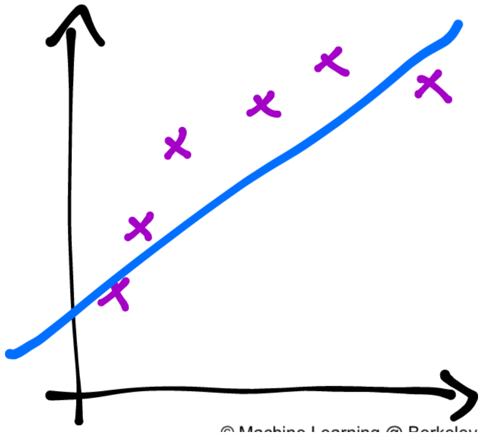
- e.g. Regression
- e.g. Naive Bayes
- *Linear algos*
- *Parametric algos*

Low bias algos tend to be **more complex**, with flexible underlying structure.

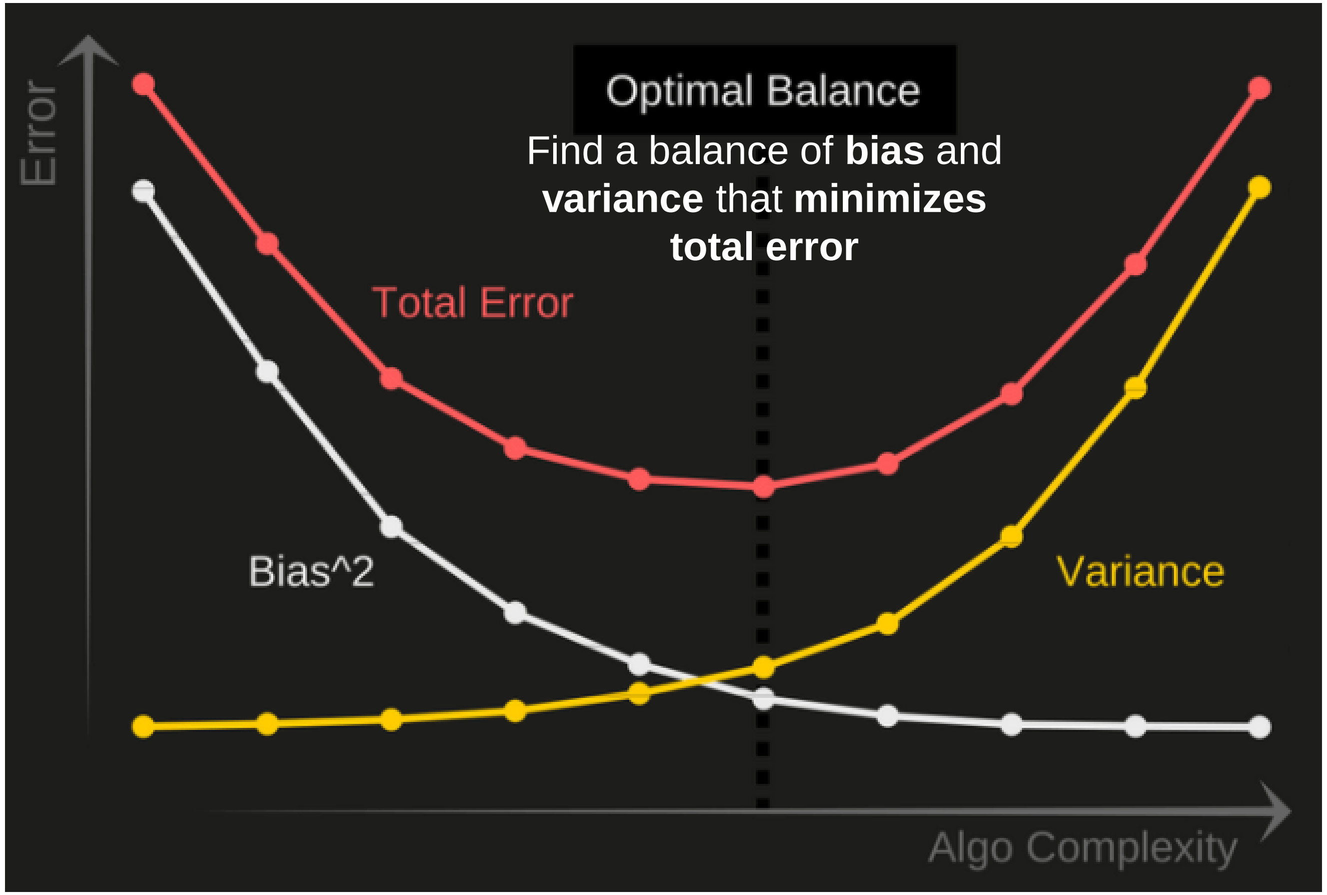
- e.g. Decision trees
- e.g. Nearest neighbors
- *Non-linear algos*
- *Non-parametric algos*

Bias-Variance Tradeoff

Algorithms that are **not complex enough** (**high BIAS**) produce **UNDERFIT** models that cannot learn the signal from data



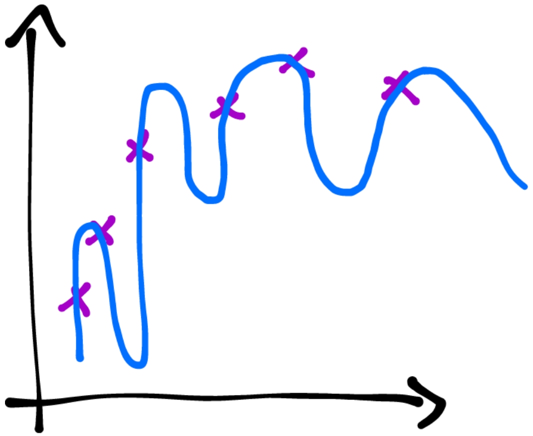
© Machine Learning @ Berkeley



Optimal Balance

Find a balance of **bias** and **variance** that **minimizes total error**

Algorithms that are **too complex (high VARIANCE)** produce **OVERFIT** models that memorize noise instead of the signal

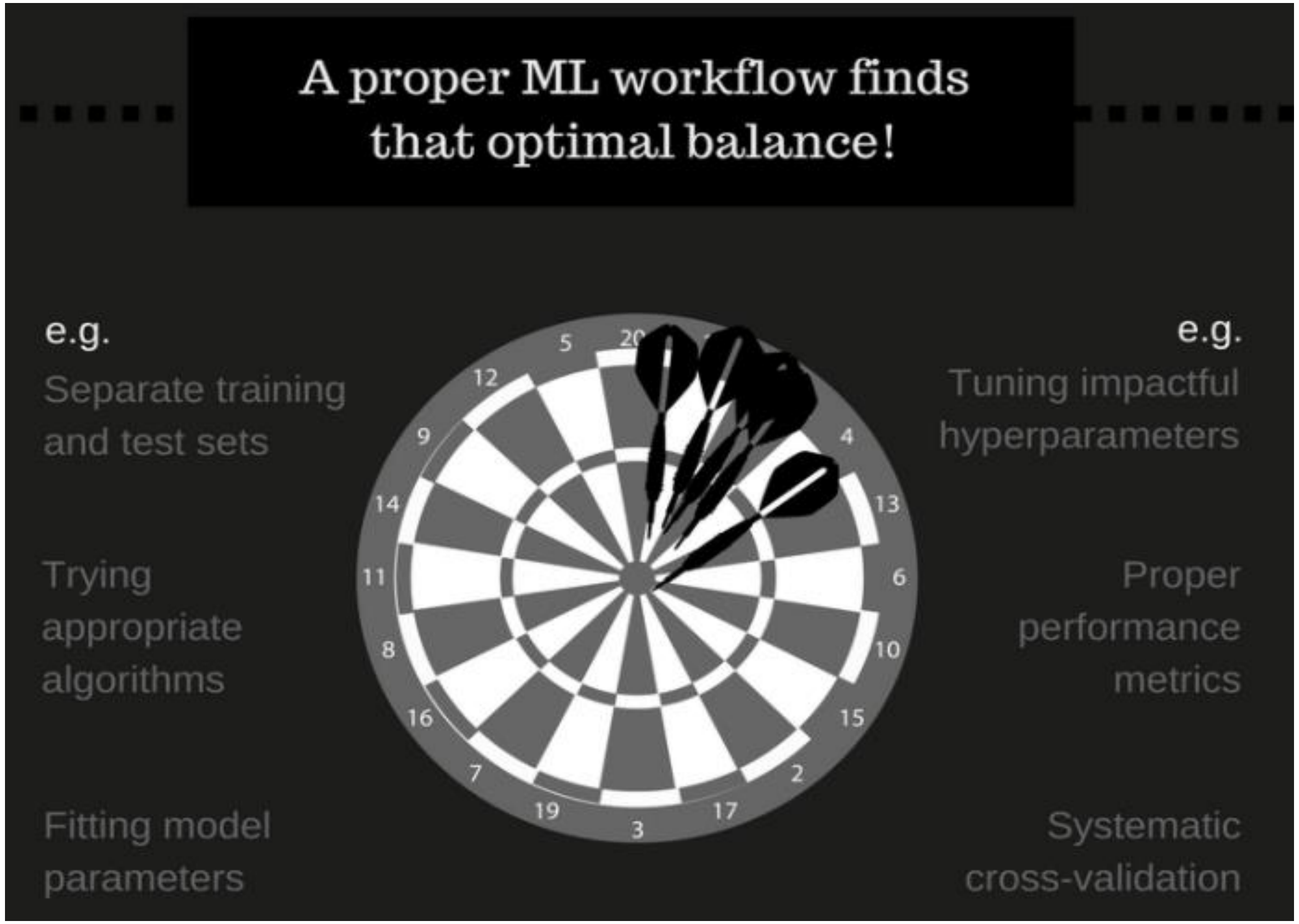


Bias-Variance Tradeoff

Separate training and test sets

Trying appropriate algorithms

Fitting model parameters



A proper ML workflow finds that optimal balance!

e.g.
Separate training and test sets

Trying appropriate algorithms

Fitting model parameters

e.g.
Tuning impactful hyperparameters

Proper performance metrics

Systematic cross-validation

Tuning impactful hyperparameter

Proper performance metrics

Systematic cross-validation

ML Diagnostic

- A test to gain insight what is or is not working with a learning algorithm
- Gain guidance as to how best to improve the performance of the learning algorithm (deciding what to try next)
- Perform **error analysis**
  - Manually examine misclassified instances
  - Use a single real number performance metric based on validation set

ML Diagnostic

- Choosing what to try next
  - Get more training data
  - Try smaller sets of features
  - Try getting additional features
  - Try adding polynomial features (increase the complexity of hypothesis function)

ML Diagnostic

- Choosing what to try next

Get more training data	Fixes high variance
Try smaller sets of features	Fixes high variance
Try getting additional features	Fixes high bias
Try adding polynomial features	Fixes high bias