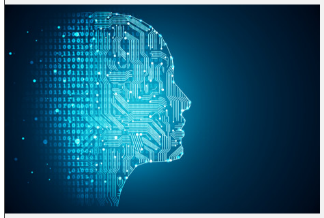


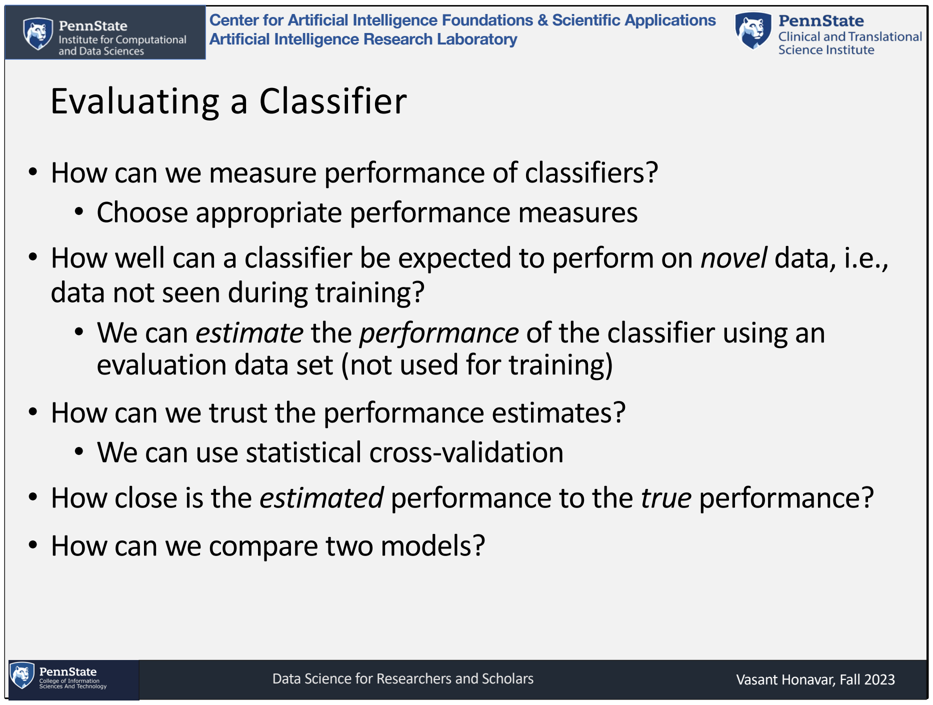
Data Science for Researchers and Scholars

Vasant G. Honavar
Dorothy Foehr Huck and J. Lloyd Huck Chair in Biomedical Data Sciences and Artificial Intelligence
Professor of Data Sciences, Informatics, Computer Science and Engineering, Bioinformatics & Genomics, Public Health Sciences and Neuroscience
Director, Center for Artificial Intelligence Foundations and Scientific Applications
Associate Director, Institute for Computational and Data Sciences
Pennsylvania State University

vhonavar@psu.edu
http://faculty.ist.psu.edu/vhonavar
http://ailab.ist.psu.edu

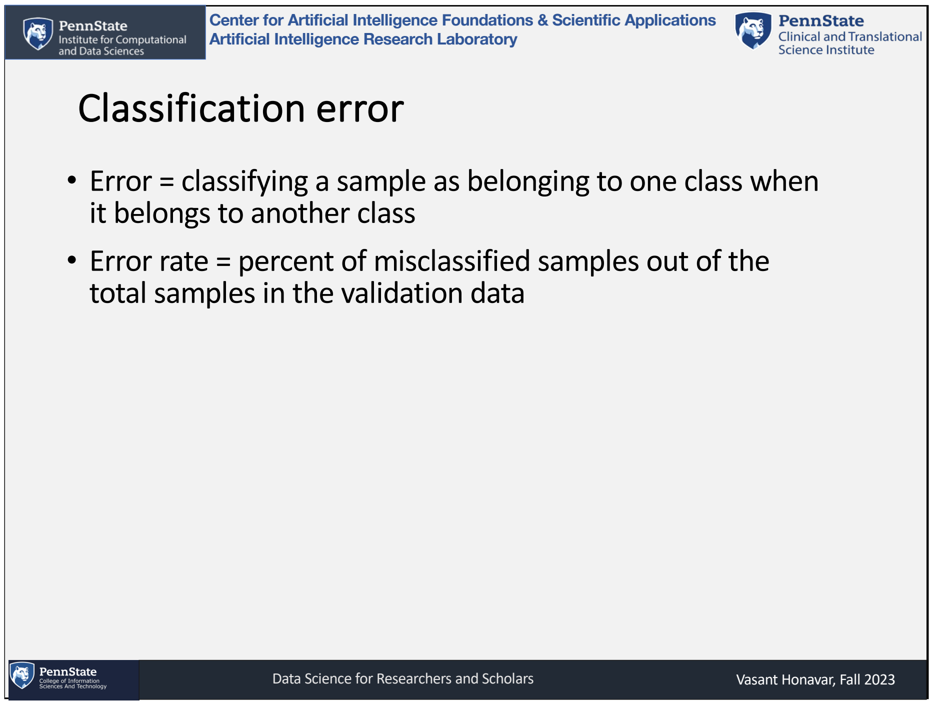
Evaluating Predictive models

- To know how well a predictive model can be expected to perform when it is put to use
- To choose the best model from among a set of alternatives



Evaluating a Classifier

- How can we measure performance of classifiers?
  - Choose appropriate performance measures
- How well can a classifier be expected to perform on *novel* data, i.e., data not seen during training?
  - We can *estimate* the *performance* of the classifier using an evaluation data set (not used for training)
- How can we trust the performance estimates?
  - We can use statistical cross-validation
- How close is the *estimated* performance to the *true* performance?
- How can we compare two models?



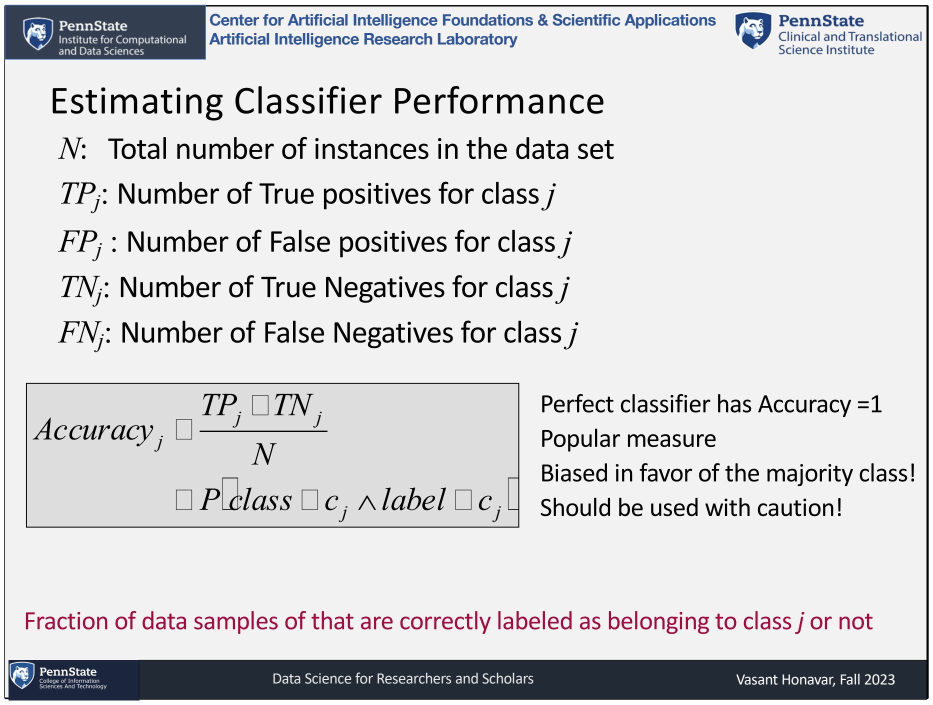
Classification error

- Error = classifying a sample as belonging to one class when it belongs to another class
- Error rate = percent of misclassified samples out of the total samples in the validation data

Naïve Baseline

Naïve baseline: classify all samples as belonging to the most prevalent (majority) class

- We hope to do better than the naïve baseline
- When the goal is to identify high-value but rare outcomes, we may do well by doing worse than the naïve baseline in terms of accuracy



Estimating Classifier Performance

N: Total number of instances in the data set

TP_j: Number of True positives for class j

FP_j : Number of False positives for class j

TN_j: Number of True Negatives for class j

FN_j: Number of False Negatives for class j

$$Accuracy_j = \frac{TP_j + TN_j}{N}$$
$$= P(class = c_j \wedge label = c_j)$$

Perfect classifier has Accuracy =1
Popular measure
Biased in favor of the majority class!
Should be used with caution!

Fraction of data samples of that are correctly labeled as belonging to class *j* or not

Classifier Learning -- Measuring Performance

Class Label $\downarrow$ $\rightarrow$	C_1	$\neg C_1$
C_1	TP= 55	FP=5
$\neg C_1$	FN=10	TN=30

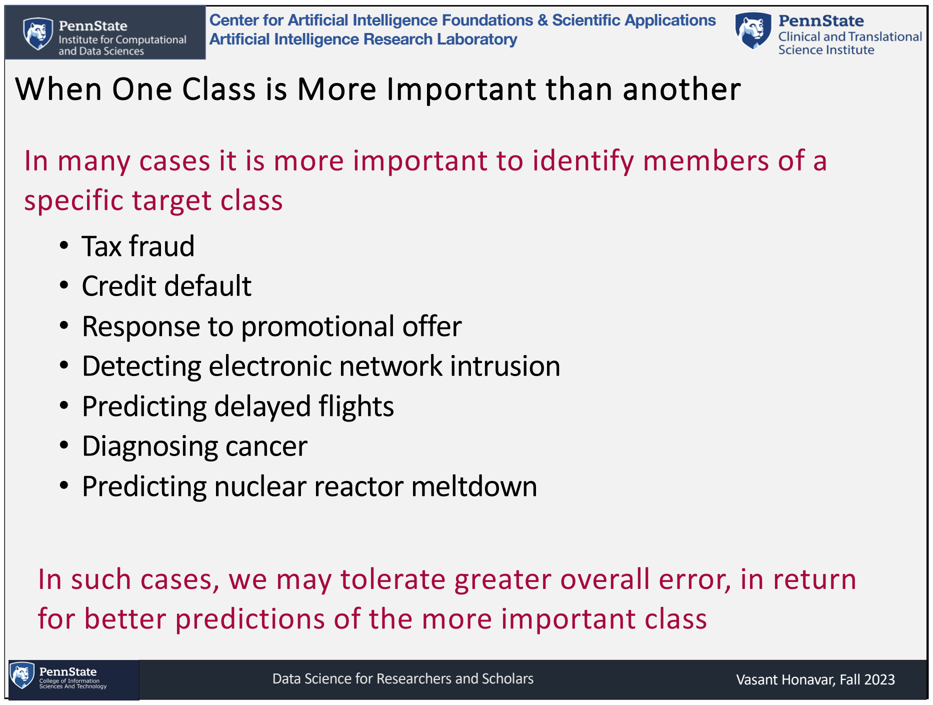
$$N = TP + FN + TN + FP = 100$$

$$sensitivity_1 = \frac{TP}{TP + FN} = \frac{55}{55 + 10} = \frac{55}{65}$$

$$specificity_1 = \frac{TP}{TP + FP} = \frac{55}{55 + 5} = \frac{55}{60}$$

$$accuracy_1 = \frac{TP + TN}{N} = \frac{55 + 30}{100} = \frac{85}{100}$$

$$falsealarm_1 = \frac{FP}{TN + FP} = \frac{5}{30 + 5} = \frac{5}{35}$$



When One Class is More Important than another

In many cases it is more important to identify members of a specific target class

- Tax fraud
- Credit default
- Response to promotional offer
- Detecting electronic network intrusion
- Predicting delayed flights
- Diagnosing cancer
- Predicting nuclear reactor meltdown

In such cases, we may tolerate greater overall error, in return for better predictions of the more important class

Measuring Classifier Performance: Sensitivity

$$
\begin{aligned}
Sensitivity_j &= \frac{TP_j}{TP_j + FN_j} \\
&= \frac{Count(label = c_j \wedge class = c_j)}{Count(class = c_j)} \\
&= P(label = c_j \mid class = c_j)
\end{aligned}
$$

Fraction of samples of class *j* that are correctly detected

Perfect classifier → Sensitivity = 1

Probability of correctly labeling members of the target class

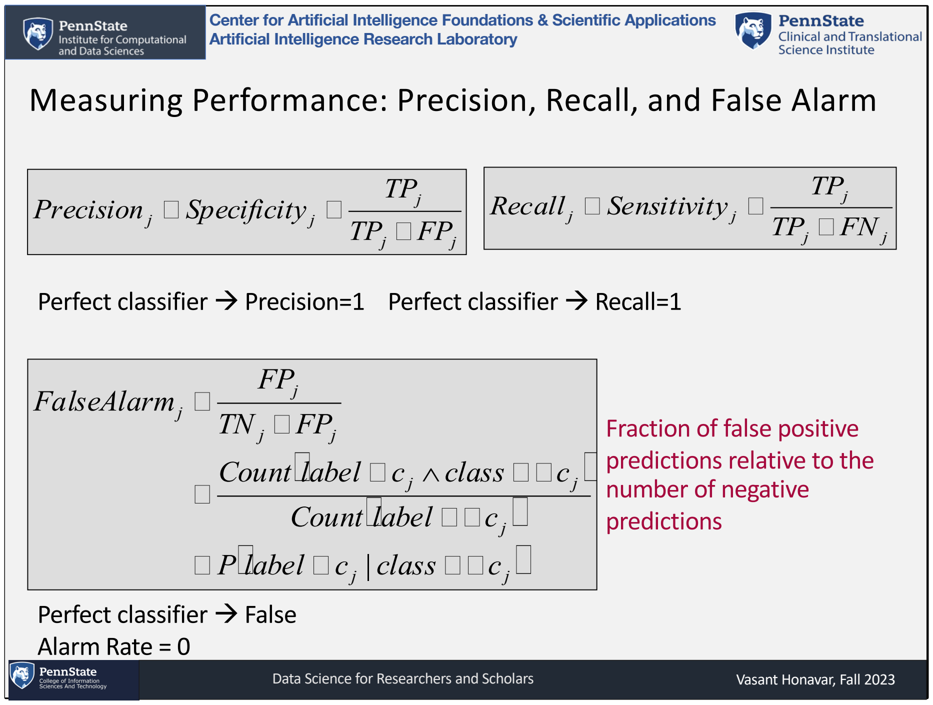
Also called recall or hit rate

Measuring Classifier Performance: Specificity

$$\begin{aligned} Specificity_j &= \frac{TP_j}{TP_j + FP_j} \\ &= \frac{Count(label = c_j \wedge class = c_j)}{Count(label = c_j)} \\ &= P(class = c_j \mid label = c_j) \end{aligned}$$

Fraction of positively labeled samples that are in fact positive

Perfect classifier → Specificity = 1
Also called precision
Probability that a positive prediction is correct



Measuring Performance: Precision, Recall, and False Alarm

$$Precision_j = Specificity_j = \frac{TP_j}{TP_j + FP_j}$$

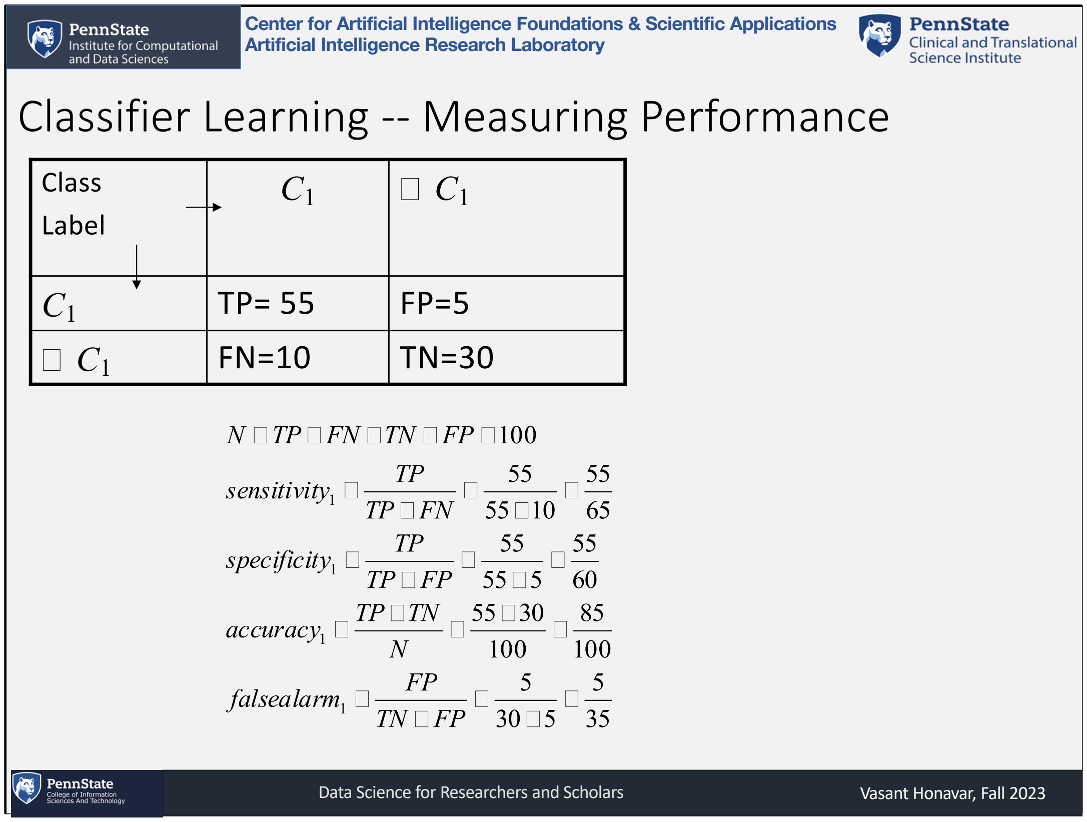
$$Recall_j = Sensitivity_j = \frac{TP_j}{TP_j + FN_j}$$

Perfect classifier → Precision=1   Perfect classifier → Recall=1

$$FalseAlarm_j = \frac{FP_j}{TN_j + FP_j}$$
$$= \frac{Count(label = c_j \wedge class = \neg c_j)}{Count(label = \neg c_j)}$$
$$= P(label = c_j \mid class = \neg c_j)$$

Fraction of false positive predictions relative to the number of negative predictions

Perfect classifier → False Alarm Rate = 0



Classifier Learning -- Measuring Performance

Class Label	C_1	$\neg C_1$
C_1	TP= 55	FP=5
$\neg C_1$	FN=10	TN=30

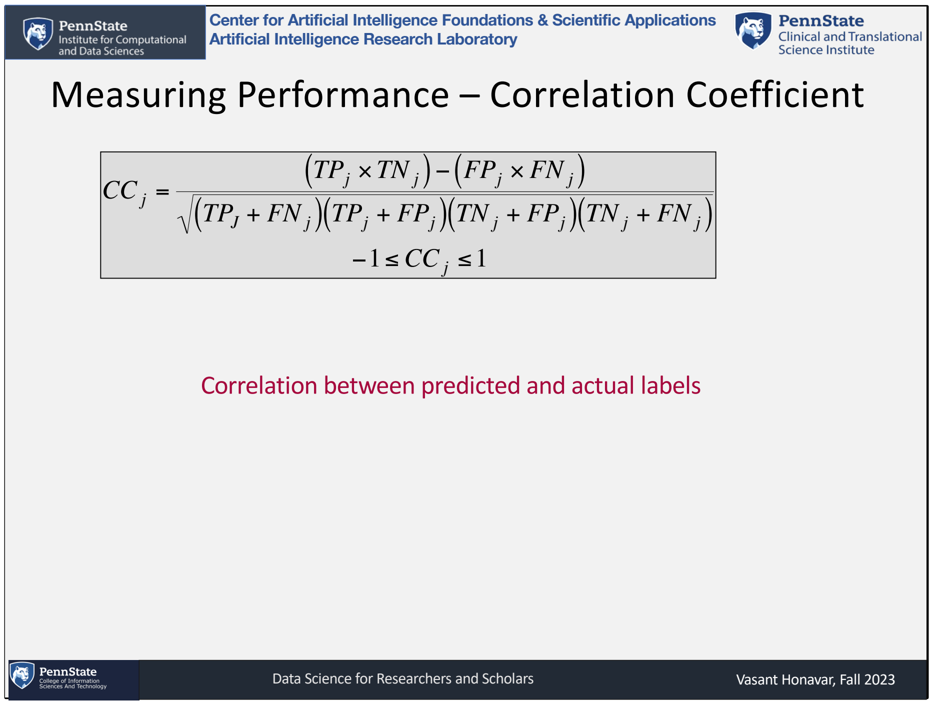
$$N = TP + FN + TN + FP = 100$$

$$sensitivity_1 = \frac{TP}{TP + FN} = \frac{55}{55 + 10} = \frac{55}{65}$$

$$specificity_1 = \frac{TP}{TP + FP} = \frac{55}{55 + 5} = \frac{55}{60}$$

$$accuracy_1 = \frac{TP + TN}{N} = \frac{55 + 30}{100} = \frac{85}{100}$$

$$falsealarm_1 = \frac{FP}{TN + FP} = \frac{5}{30 + 5} = \frac{5}{35}$$

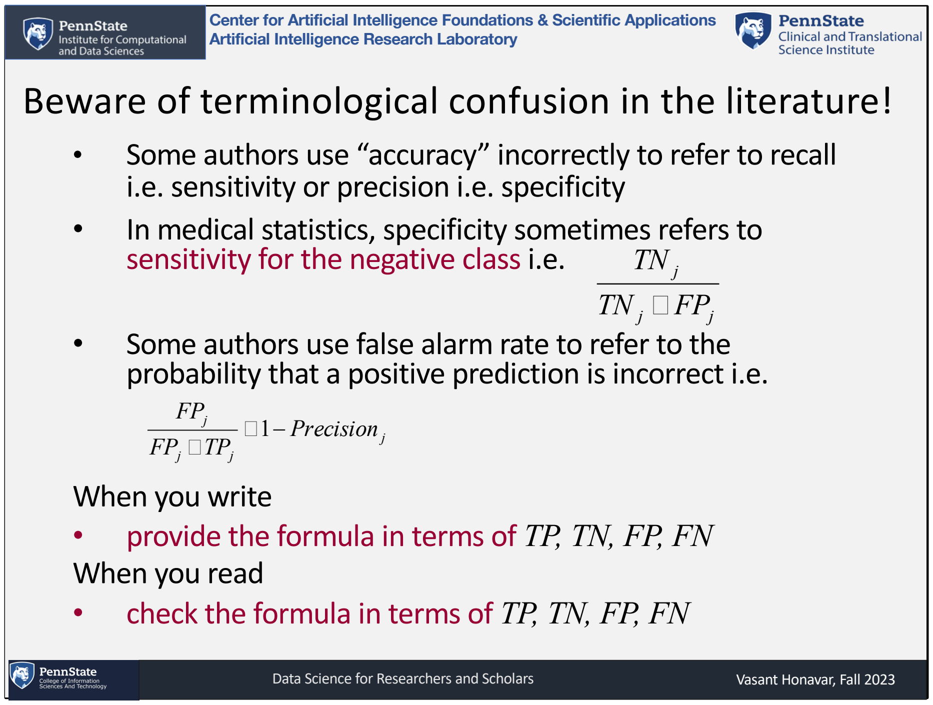


Measuring Performance – Correlation Coefficient

$$CC_j = \frac{(TP_j \times TN_j) - (FP_j \times FN_j)}{\sqrt{(TP_J + FN_j)(TP_j + FP_j)(TN_j + FP_j)(TN_j + FN_j)}}$$

$$-1 \leq CC_j \leq 1$$

Correlation between predicted and actual labels



Beware of terminological confusion in the literature!

- Some authors use “accuracy” incorrectly to refer to recall i.e. sensitivity or precision i.e. specificity
- In medical statistics, specificity sometimes refers to sensitivity for the negative class i.e. $\frac{TN_j}{TN_j + FP_j}$
- Some authors use false alarm rate to refer to the probability that a positive prediction is incorrect i.e.

$$\frac{FP_j}{FP_j + TP_j} = 1 - Precision_j$$

When you write

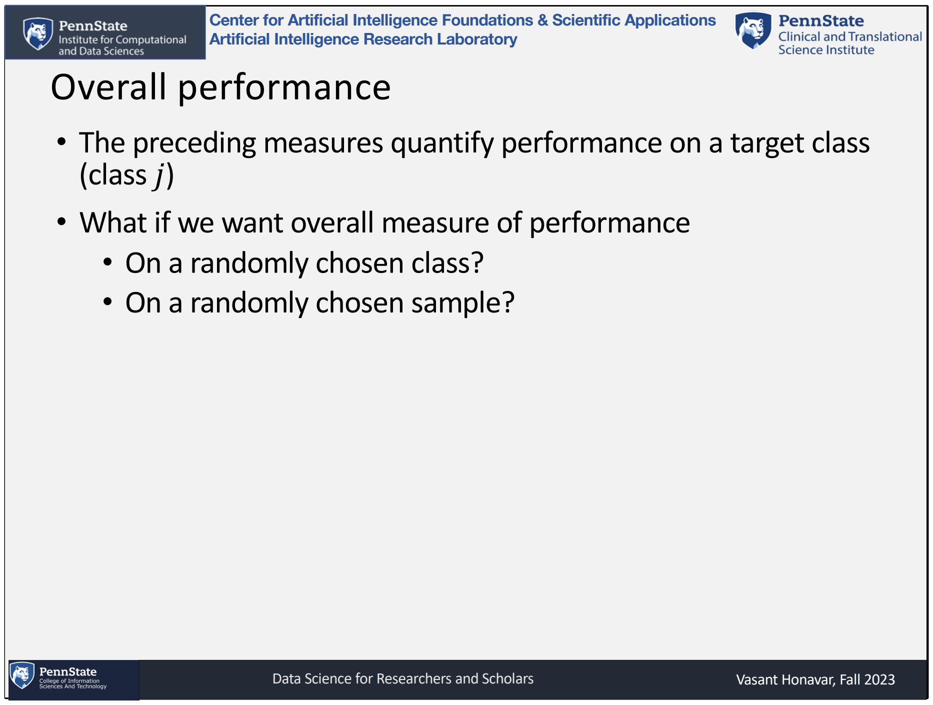
- provide the formula in terms of *TP, TN, FP, FN*

When you read

- check the formula in terms of *TP, TN, FP, FN*

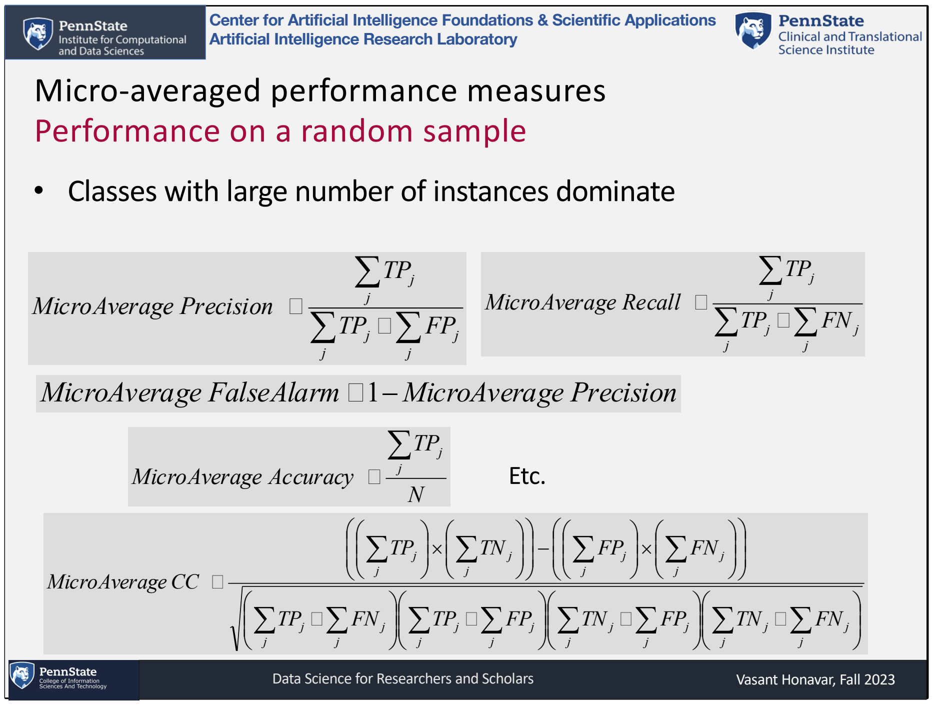
Measuring Classifier Performance

- TP, FP, TN, FN provide the relevant information
- No single measure tells the whole story
- A classifier with 98% accuracy can be useless if 98% of the population does not have cancer and the 2% that do are misclassified by the classifier
- Use of multiple measures recommended
- Beware of terminological confusion!



Overall performance

- The preceding measures quantify performance on a target class (class *j*)
- What if we want overall measure of performance
  - On a randomly chosen class?
  - On a randomly chosen sample?



Micro-averaged performance measures
Performance on a random sample

- Classes with large number of instances dominate

$$MicroAverage\ Precision = \frac{\sum_j TP_j}{\sum_j TP_j + \sum_j FP_j}$$

$$MicroAverage\ Recall = \frac{\sum_j TP_j}{\sum_j TP_j + \sum_j FN_j}$$

$$MicroAverage\ FalseAlarm = 1 - MicroAverage\ Precision$$

$$MicroAverage\ Accuracy = \frac{\sum_j TP_j}{N}$$

Etc.

$$MicroAverage\ CC = \frac{\left(\left(\sum_j TP_j\right)\times\left(\sum_j TN_j\right)\right)-\left(\left(\sum_j FP_j\right)\times\left(\sum_j FN_j\right)\right)}{\sqrt{\left(\sum_j TP_j + \sum_j FN_j\right)\left(\sum_j TP_j + \sum_j FP_j\right)\left(\sum_j TN_j + \sum_j FP_j\right)\left(\sum_j TN_j + \sum_j FN_j\right)}}$$

Macro-averaged performance measures
Performance on a random class

Macro averaging gives equal importance to each of the *M* classes

$$MacroAverage\ Sensitivity \ \square \frac{1}{M}\sum_{j} Sensitivity_j$$

$$MacroAverageCorrelationCoeff \ \square \frac{1}{M}\sum_{j} CorrelationCoeff_j$$

$$MacroAverage\ Specificity \ \square \frac{1}{M}\sum_{j} Specificity_j$$

Cutoff for classification

Most machine learning algorithms classify via a 2-step process:

For each sample,

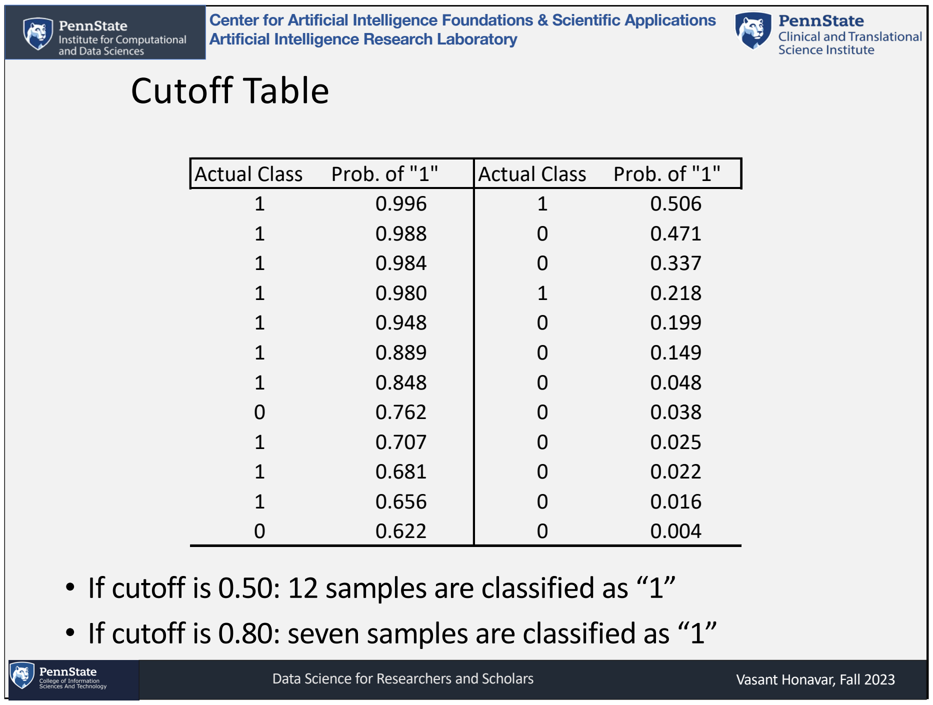
1. Compute **probability of belonging to class "1"**
2. Compare to cutoff value, and classify accordingly

- Default cutoff value is 0.50 in the two-class case (if the prior probability of two classes is same)

  If probability of sample belonging to class 1 $\geq$ 0.50, classify as "1"

  If probability of sample belonging to class 1 $<$ 0.50, classify as "0"

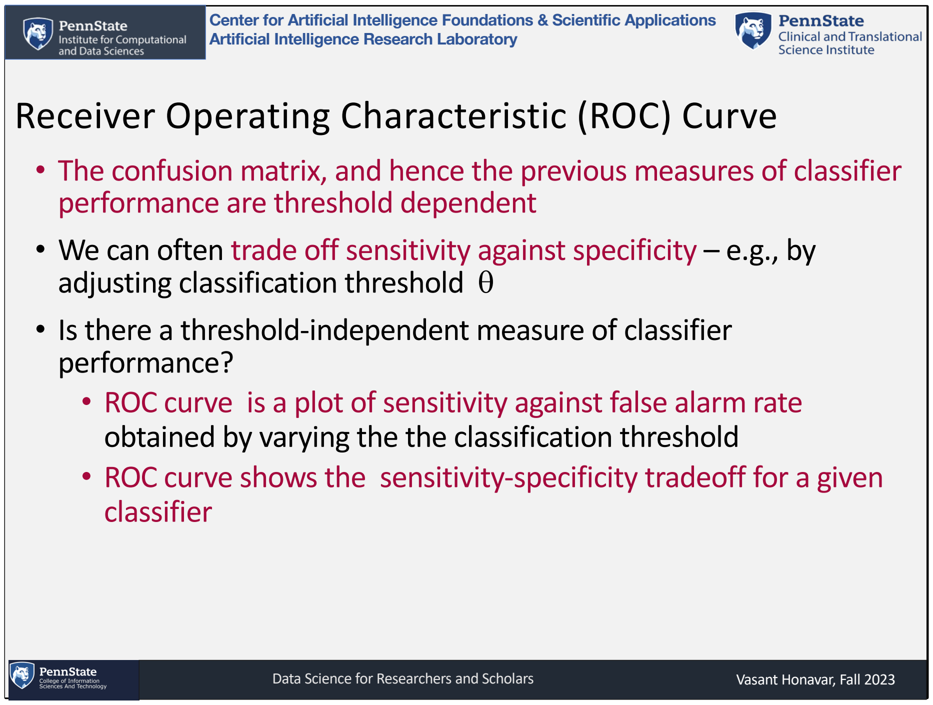
- Can use different cutoff values for trading off one measure against another (more on this later)



Cutoff Table

Actual Class	Prob. of "1"	Actual Class	Prob. of "1"
1	0.996	1	0.506
1	0.988	0	0.471
1	0.984	0	0.337
1	0.980	1	0.218
1	0.948	0	0.199
1	0.889	0	0.149
1	0.848	0	0.048
0	0.762	0	0.038
1	0.707	0	0.025
1	0.681	0	0.022
1	0.656	0	0.016
0	0.622	0	0.004

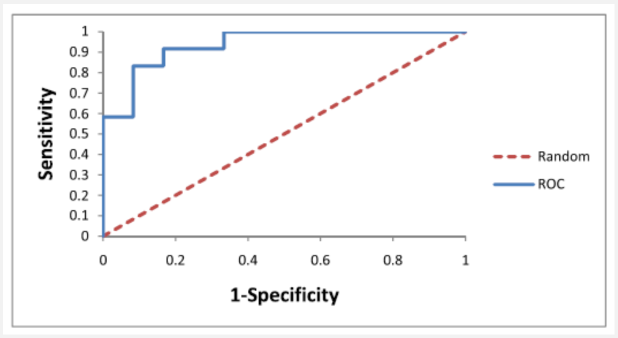
- If cutoff is 0.50: 12 samples are classified as “1”
- If cutoff is 0.80: seven samples are classified as “1”



Receiver Operating Characteristic (ROC) Curve

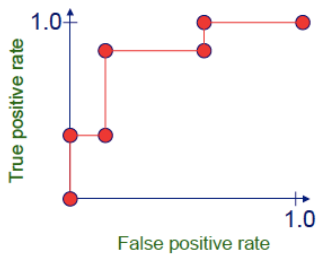
- The confusion matrix, and hence the previous measures of classifier performance are threshold dependent
- We can often trade off sensitivity against specificity – e.g., by adjusting classification threshold θ
- Is there a threshold-independent measure of classifier performance?
  - ROC curve is a plot of sensitivity against false alarm rate obtained by varying the the classification threshold
  - ROC curve shows the sensitivity-specificity tradeoff for a given classifier

ROC Curve

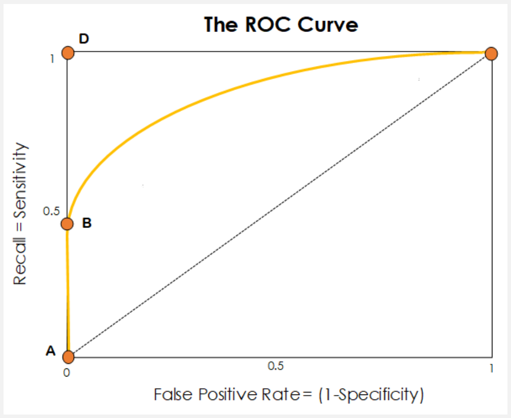

Sensitivity
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.2
0.4
0.6
0.8
1
1-Specificity
Random
ROC

Computing the ROC curve

instance	confidence positive		correct class
Ex 9	.99		+
Ex 7	.98	TPR= 2/5, FPR= 0/5	+
Ex 1	.72	TPR= 2/5, FPR= 1/5	-
Ex 2	.70		+
Ex 6	.65	TPR= 4/5, FPR= 1/5	+
Ex 10	.51		-
Ex 3	.39	TPR= 4/5, FPR= 3/5	-
Ex 5	.24	TPR= 5/5, FPR= 3/5	+
Ex 4	.11		-
Ex 8	.01	TPR= 5/5, FPR= 5/5	-

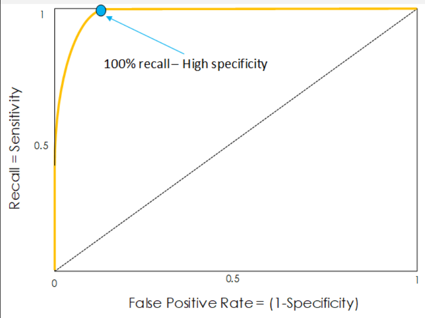


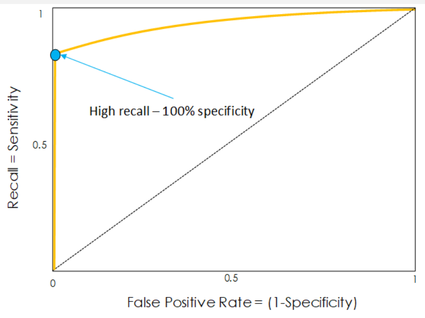
How to use an ROC curve?



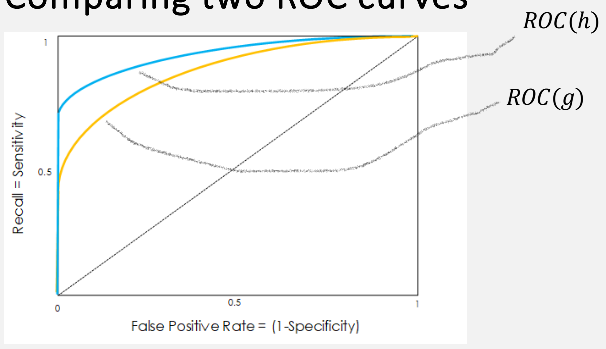
- Each point on the ROC curve corresponds to a specific tradeoff between sensitivity and false positive rate
- The right tradeoff is application specific

Trading off sensitivity against false positive rate



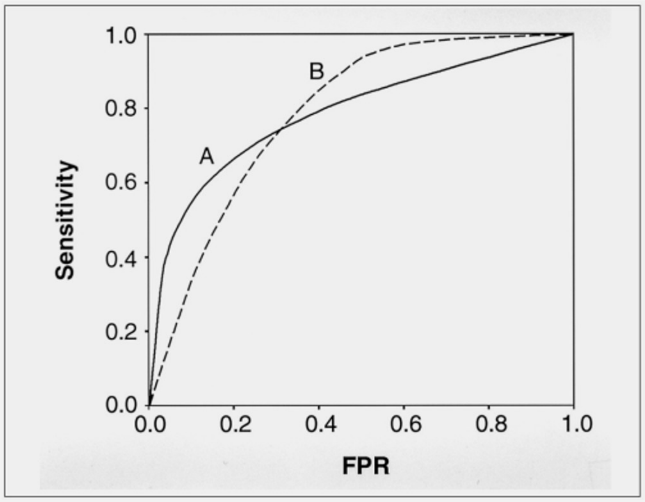


Comparing two ROC curves



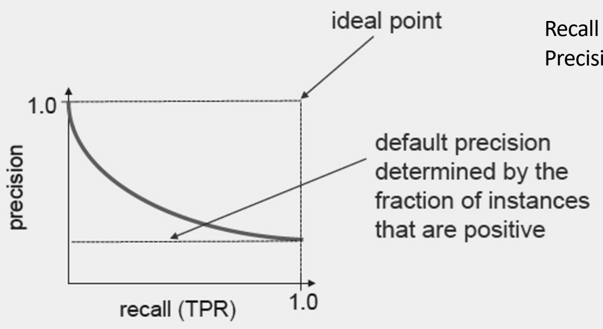
- A classifier *h* is better than another classifier *g* if ROC(*h*) dominates the ROC(*g*)
- ROC(*h*) dominates ROC(*g*) → AreaROC(*h*) > AreaROC(*g*)

Comparing ROC curves

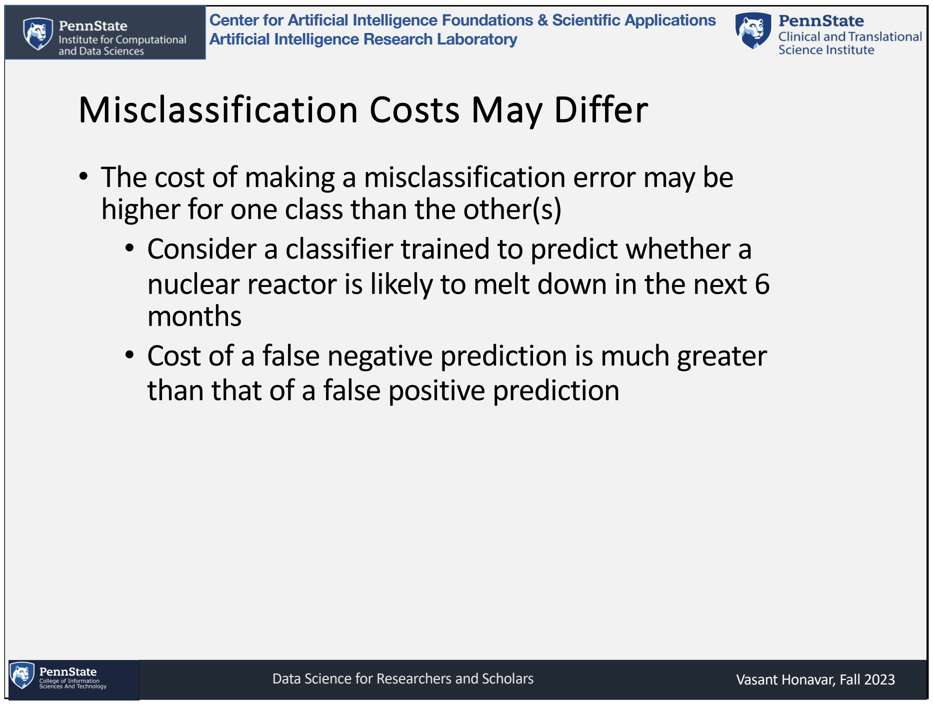


Precision/recall curves

A *precision/recall curve* plots the precision vs. recall (TP-rate) as a threshold on the confidence of an instance being positive is varied

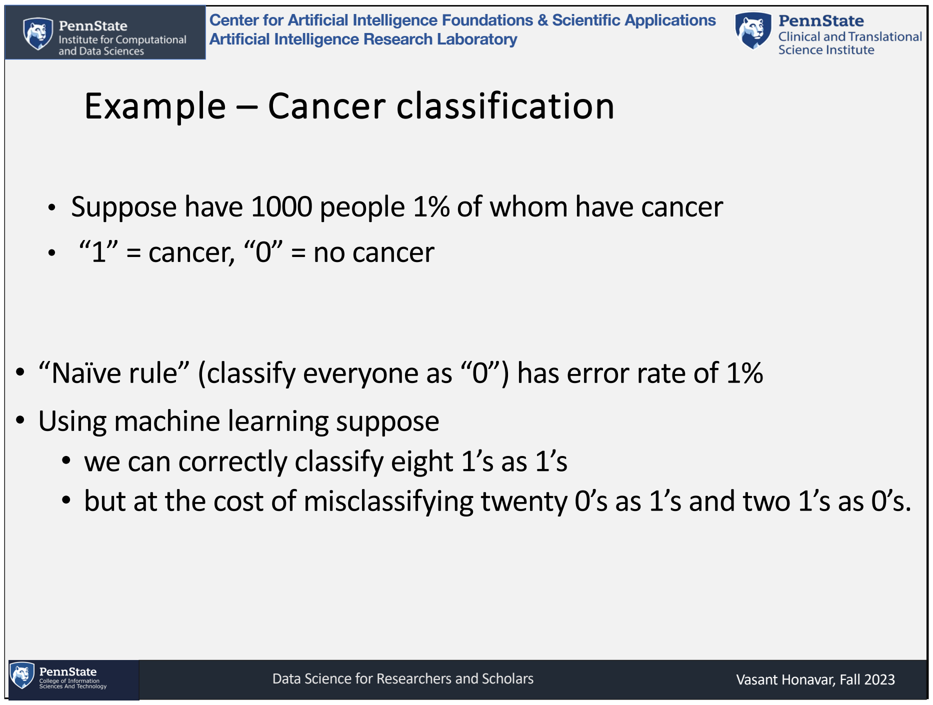


Recall = Sensitivity = TPR
Precision = Specificity



Misclassification Costs May Differ

- The cost of making a misclassification error may be higher for one class than the other(s)
  - Consider a classifier trained to predict whether a nuclear reactor is likely to melt down in the next 6 months
  - Cost of a false negative prediction is much greater than that of a false positive prediction



Example – Cancer classification

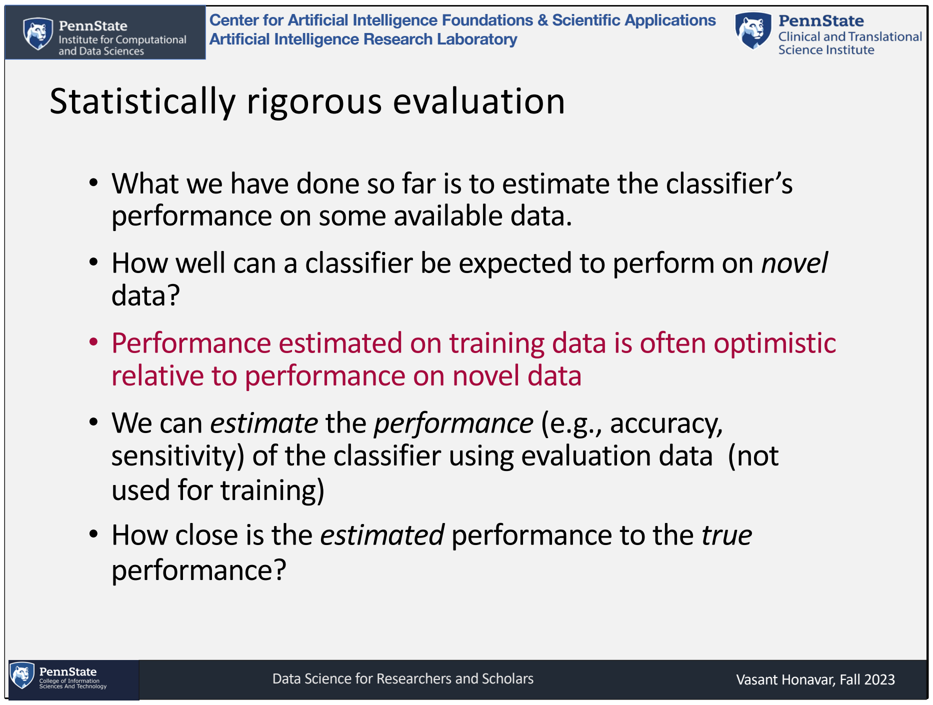
- Suppose have 1000 people 1% of whom have cancer
- "1" = cancer, "0" = no cancer

- "Naïve rule" (classify everyone as "0") has error rate of 1%
- Using machine learning suppose
  - we can correctly classify eight 1's as 1's
  - but at the cost of misclassifying twenty 0's as 1's and two 1's as 0's.

Confusion Matrix

	Predict as 1	Predict as 0
Actual 1	8	2
Actual 0	20	970

Error rate = (2+20) = 2.2% (higher than naïve rate)



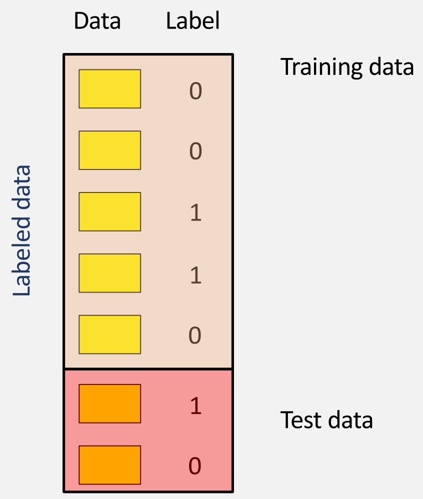
Statistically rigorous evaluation

- What we have done so far is to estimate the classifier's performance on some available data.
- How well can a classifier be expected to perform on *novel* data?
- Performance estimated on training data is often optimistic relative to performance on novel data
- We can *estimate* the *performance* (e.g., accuracy, sensitivity) of the classifier using evaluation data (not used for training)
- How close is the *estimated* performance to the *true* performance?

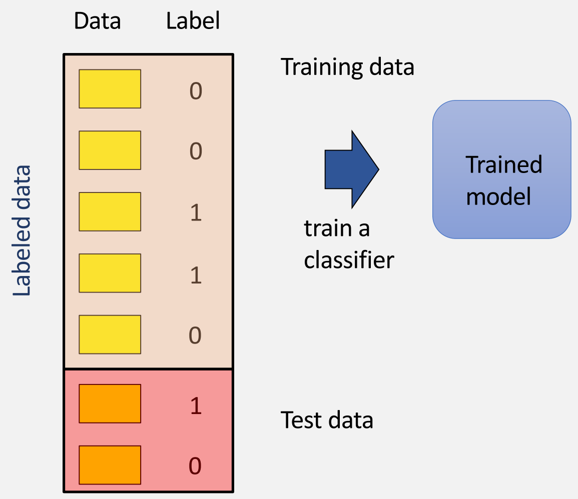
Evaluation of a classifier with limited data

- <u>Holdout method</u> – use part of the data for training, and the rest for testing
- We may be lucky or unlucky – training data or test data may not be *representative*
- <u>Solution</u> – Run multiple experiments with disjoint training and test data sets in which each class is represented in roughly the same proportion as in the entire data set

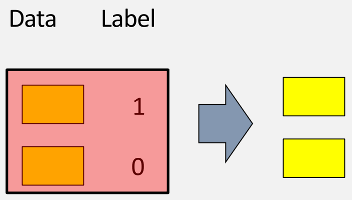
Statistical evaluation of classifiers



Classifier evaluation

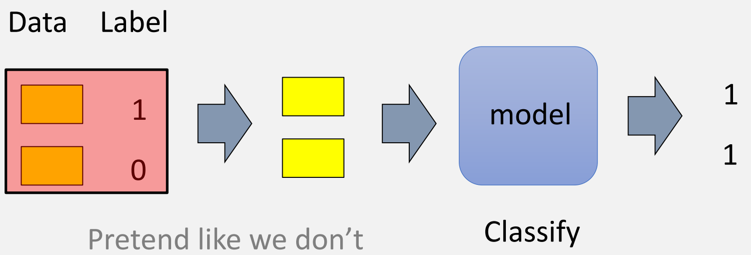


Classifier evaluation

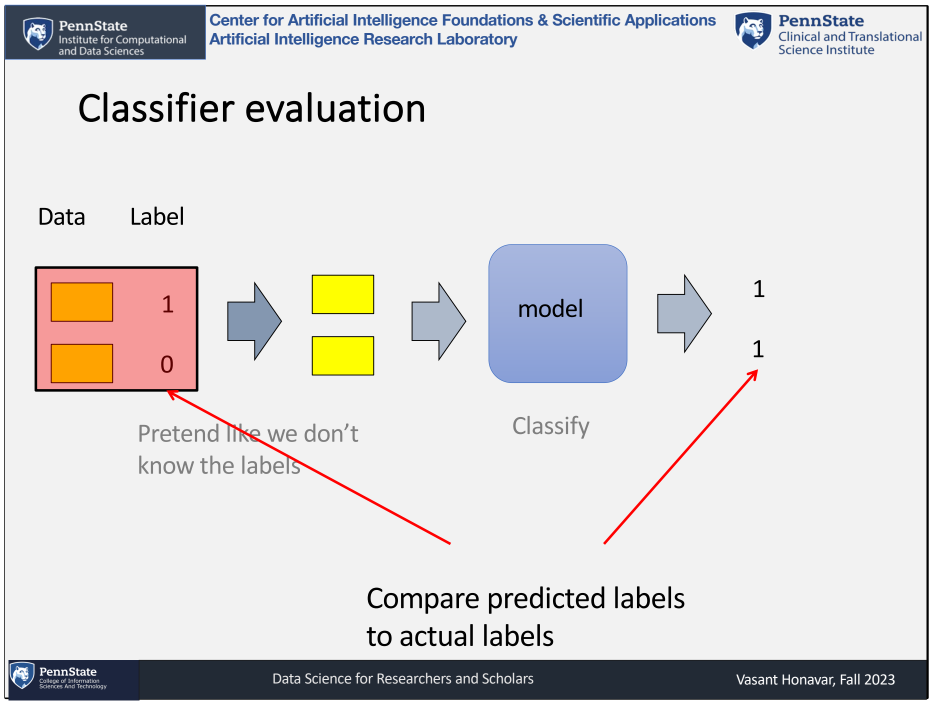


Pretend like we don't know the labels

Classifier evaluation



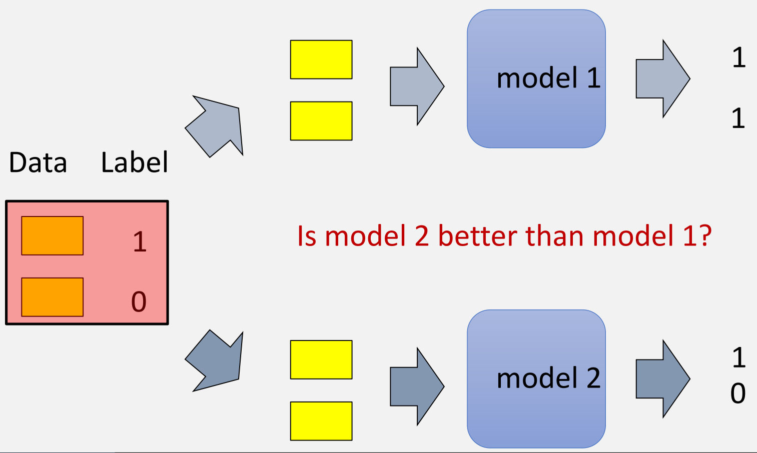
Pretend like we don't know the labels

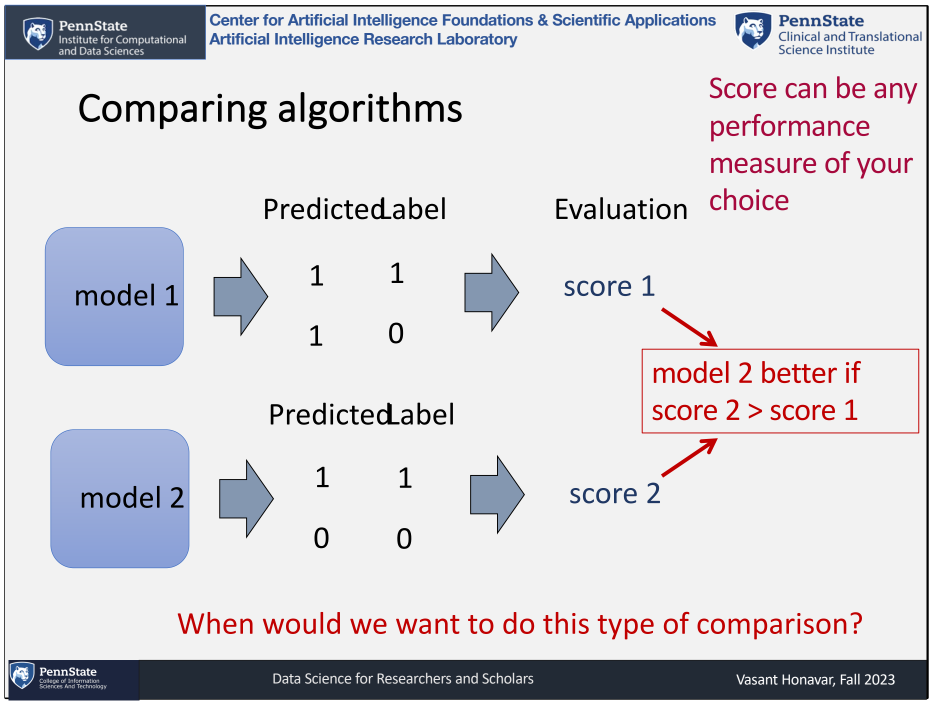


Classifier evaluation

Data
Label
1
0
model
1
1
Pretend like we don't know the labels
Classify
Compare predicted labels to actual labels

Comparing trained models





PennState Institute for Computational and Data Sciences

Center for Artificial Intelligence Foundations & Scientific Applications
Artificial Intelligence Research Laboratory

PennState Clinical and Translational Science Institute

Comparing algorithms

Score can be any performance measure of your choice

Predicted Label
Evaluation
model 1
1 1
1 0
score 1
model 2 better if score 2 > score 1
Predicted Label
model 2
1 1
0 0
score 2

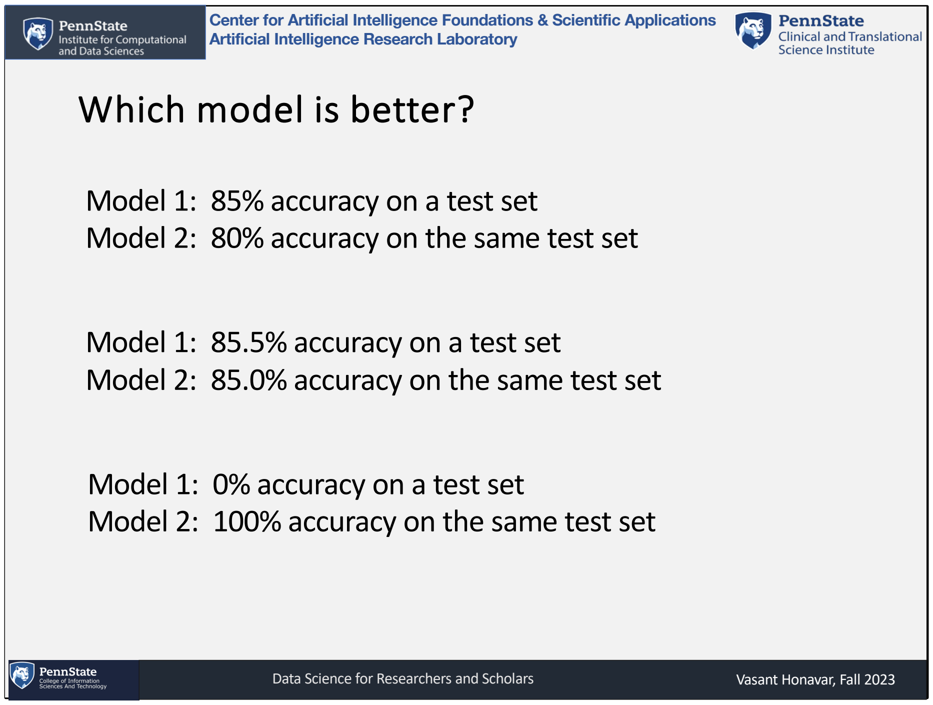
When would we want to do this type of comparison?

PennState College of Information Sciences and Technology

Data Science for Researchers and Scholars

Vasant Honavar, Fall 2023

- comparing different learning algorithms
- comparing different hyperparameters
- comparing different pre-processing techniques
- figuring out who has the best algorithm
- …



Which model is better?

Model 1: 85% accuracy on a test set
Model 2: 80% accuracy on the same test set

Model 1: 85.5% accuracy on a test set
Model 2: 85.0% accuracy on the same test set

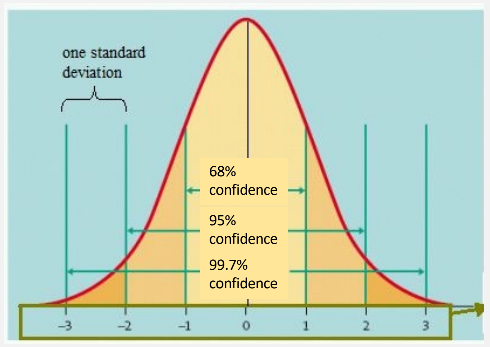
Model 1: 0% accuracy on a test set
Model 2: 100% accuracy on the same test set

Comparing scores: significance

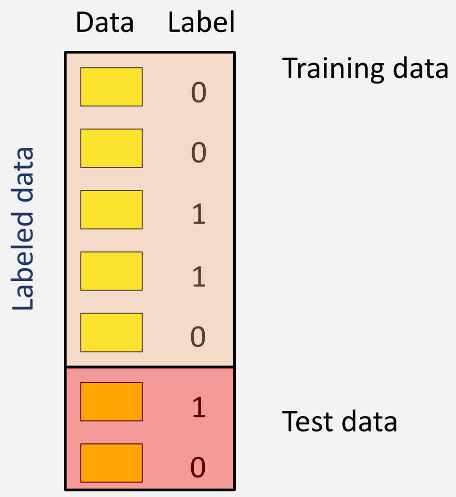
- Just comparing scores on one data set isn't enough!
- We don't particularly care that model 2 is better than model 1 on the test data set that we happened to choose by chance
- We want to know whether model 2 is better than model 1 in general
- How can we be confident that the difference is real and not just due to random chance?

Distribution of performance measure

- We need the distribution of scores!
- How can we get it?

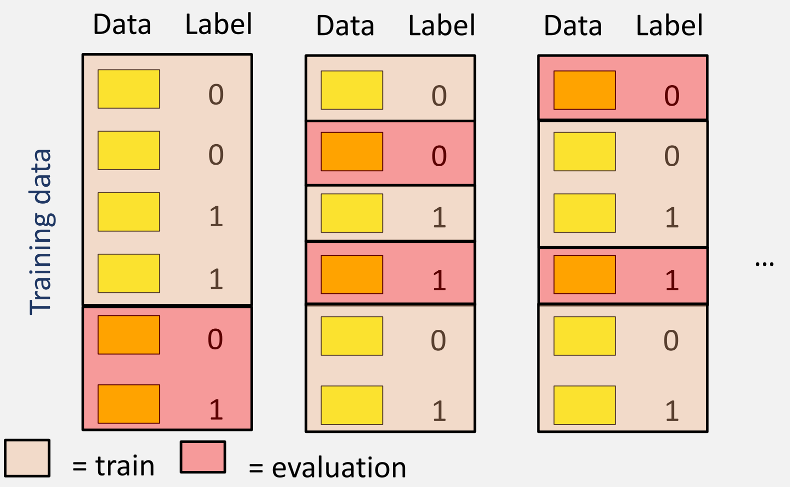


Repeated experimentation

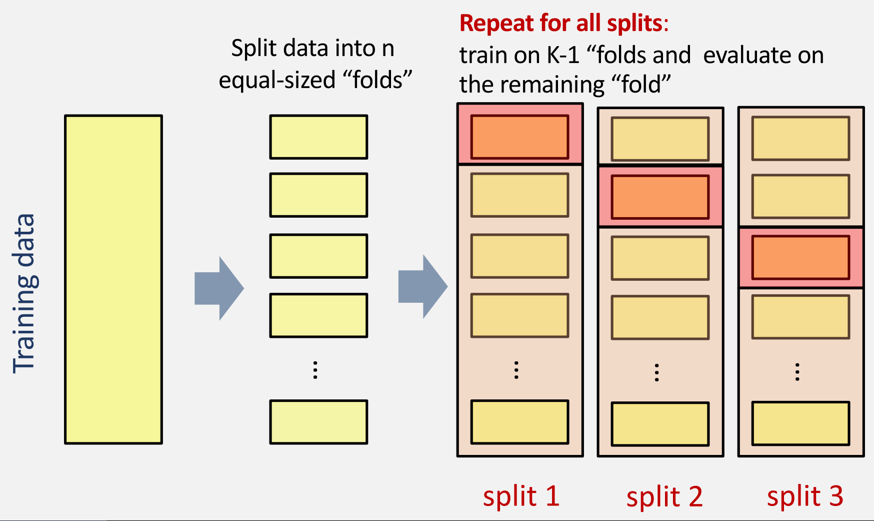


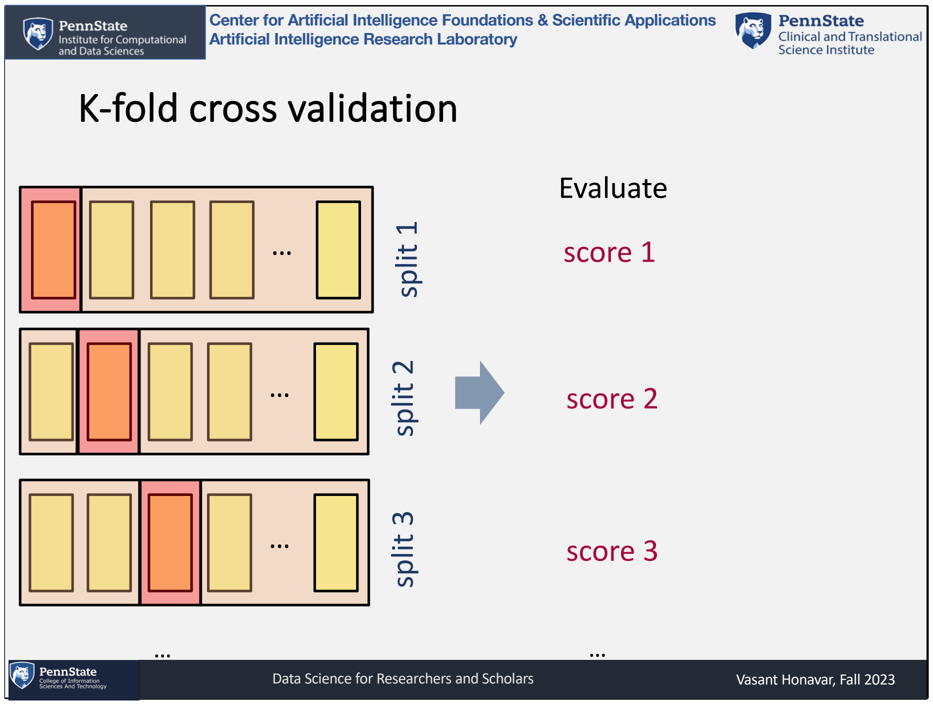
Instead of one evaluation with a particular split of training and test data, run multiple evaluations, with different splits of training and test data

Repeated experimentation

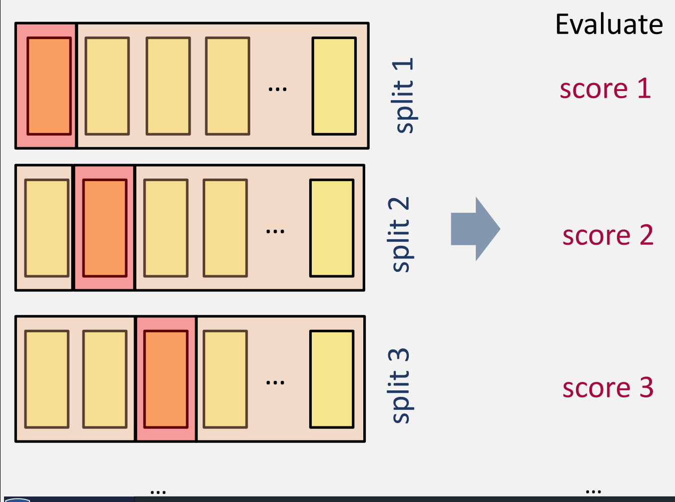


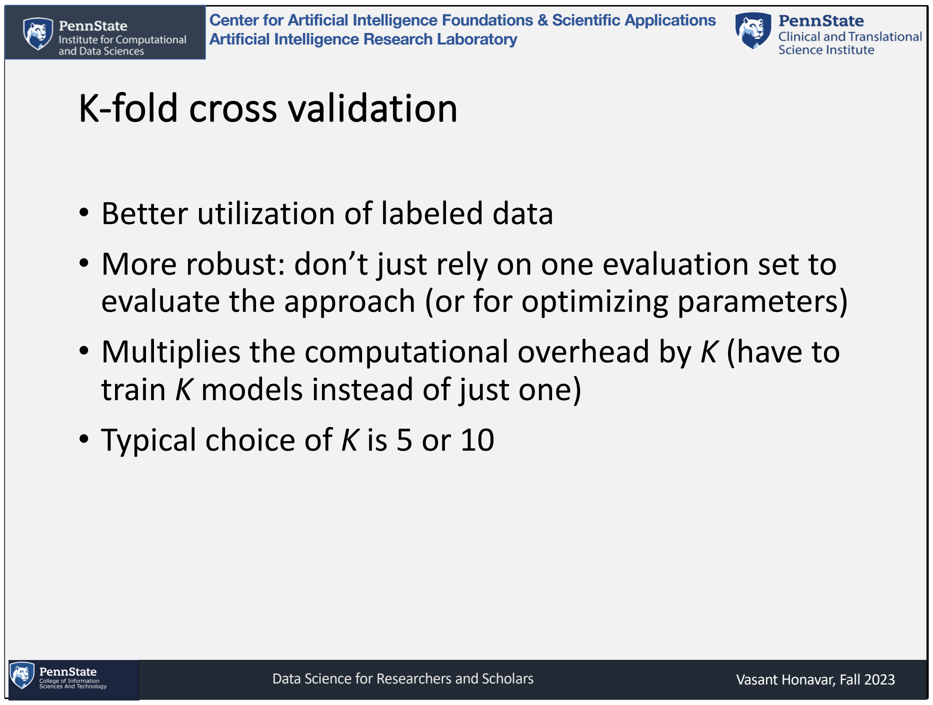
K-fold cross validation


Training data
Split data into n equal-sized "folds"
Repeat for all splits:
train on K-1 "folds and evaluate on the remaining "fold"
⋮
split 1
split 2
split 3



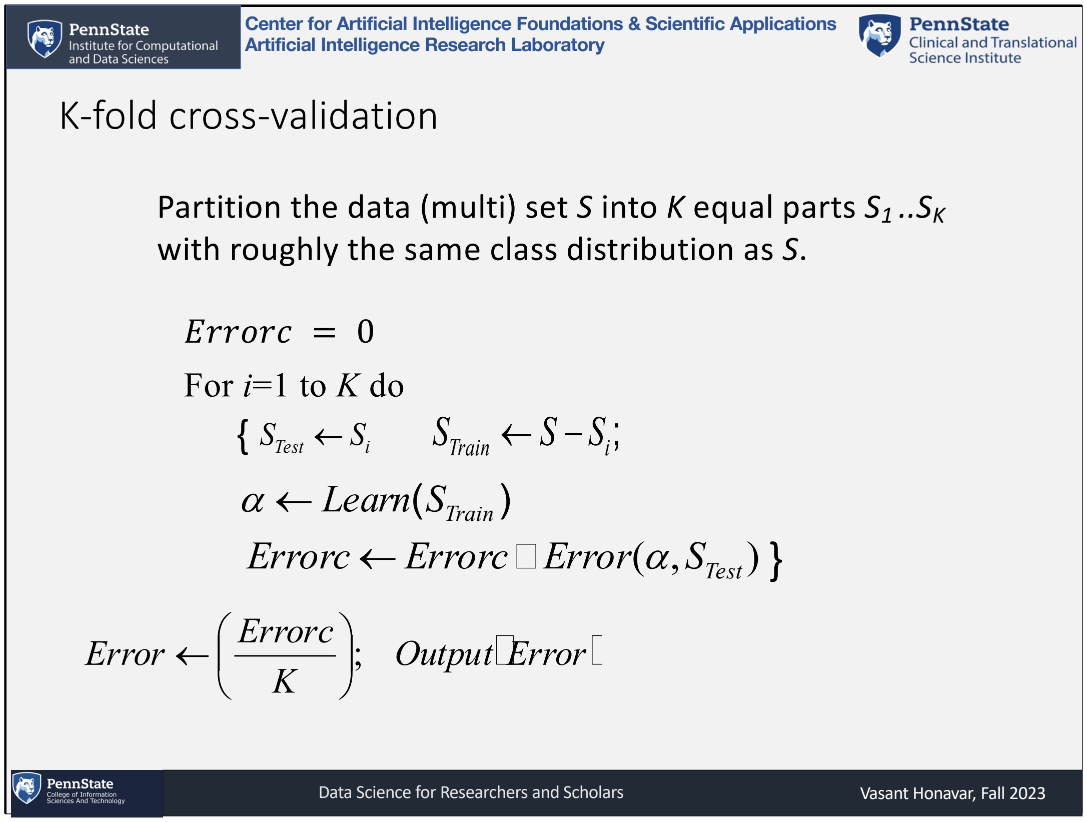
K-fold cross validation





K-fold cross validation

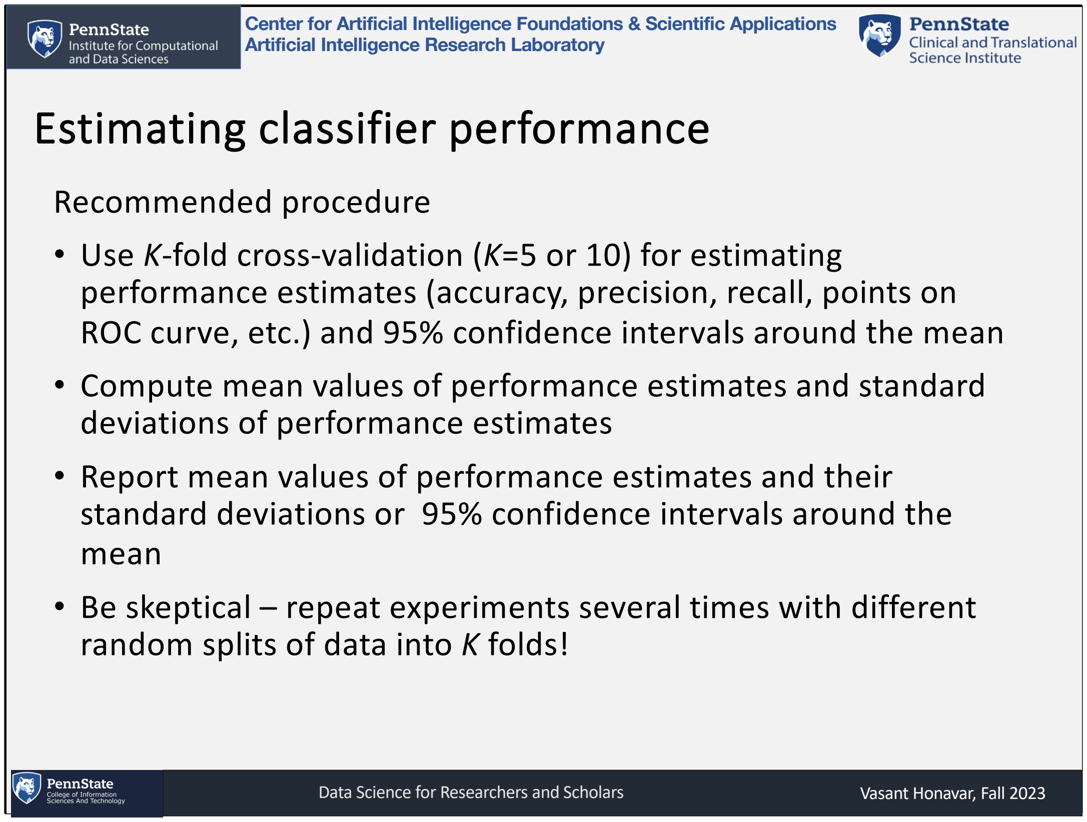
- Better utilization of labeled data
- More robust: don’t just rely on one evaluation set to evaluate the approach (or for optimizing parameters)
- Multiplies the computational overhead by *K* (have to train *K* models instead of just one)
- Typical choice of *K* is 5 or 10



K-fold cross-validation

Partition the data (multi) set *S* into *K* equal parts $S_1 .. S_K$ with roughly the same class distribution as *S*.

```
$Errorc = 0$
For $i$=1 to $K$ do
    { $S_{Test} \leftarrow S_i \quad S_{Train} \leftarrow S - S_i;$
    $\alpha \leftarrow Learn(S_{Train})$
    $Errorc \leftarrow Errorc \square Error(\alpha, S_{Test})$ }
$Error \leftarrow \left(\frac{Errorc}{K}\right); \quad Output \square Error \square$
```



Estimating classifier performance

Recommended procedure

- Use *K*-fold cross-validation (*K*=5 or 10) for estimating performance estimates (accuracy, precision, recall, points on ROC curve, etc.) and 95% confidence intervals around the mean
- Compute mean values of performance estimates and standard deviations of performance estimates
- Report mean values of performance estimates and their standard deviations or 95% confidence intervals around the mean
- Be skeptical – repeat experiments several times with different random splits of data into *K* folds!

ROC and precision/recall curves

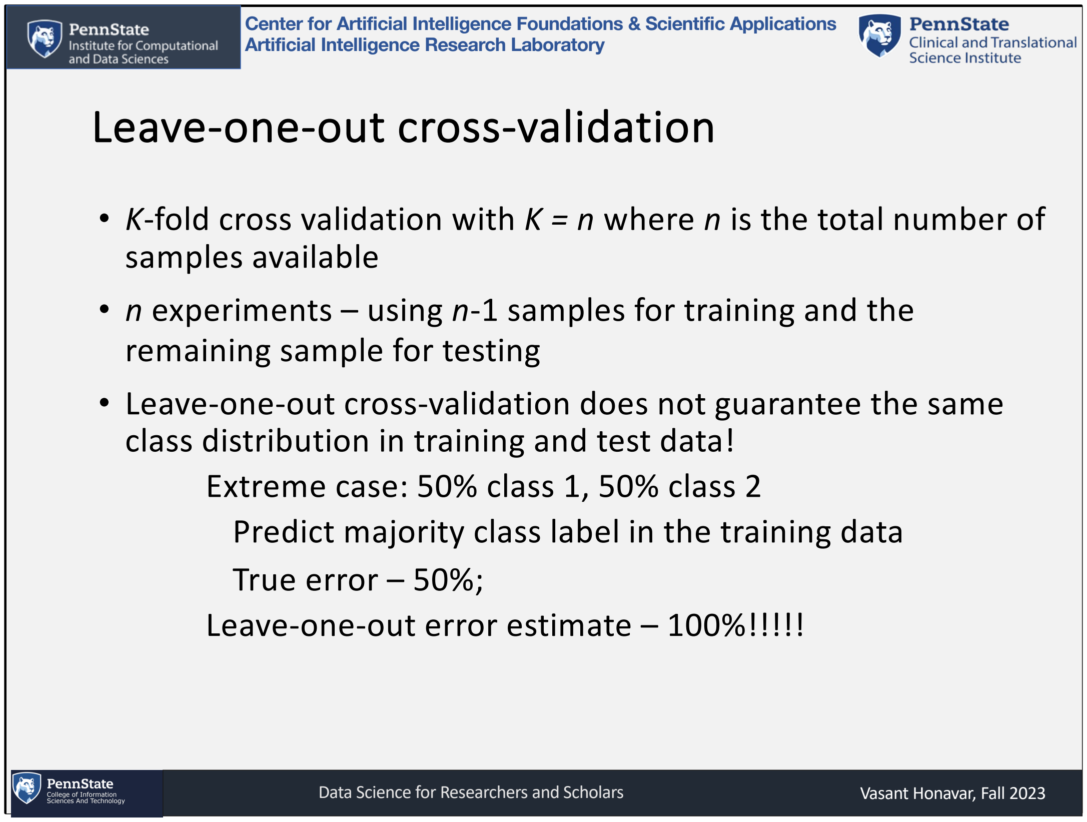
In the case of binary (2-class) classification

- Assume that the thresholds are comparable across folds.
- Pool the predictions across the K folds.
- Vary the prediction threshold and plot the ROC

In the case of multi-class classification, compute an ROC for each class against the rest (one versus all) using a procedure analogous to the above

Leave-one-out cross validation

- *K*-fold cross validation where *K* = number of samples
- aka “jackknifing”
- pros/cons?
- when would we use this?



Leave-one-out cross-validation

- *K*-fold cross validation with *K* = *n* where *n* is the total number of samples available
- *n* experiments – using *n*-1 samples for training and the remaining sample for testing
- Leave-one-out cross-validation does not guarantee the same class distribution in training and test data!

    Extreme case: 50% class 1, 50% class 2

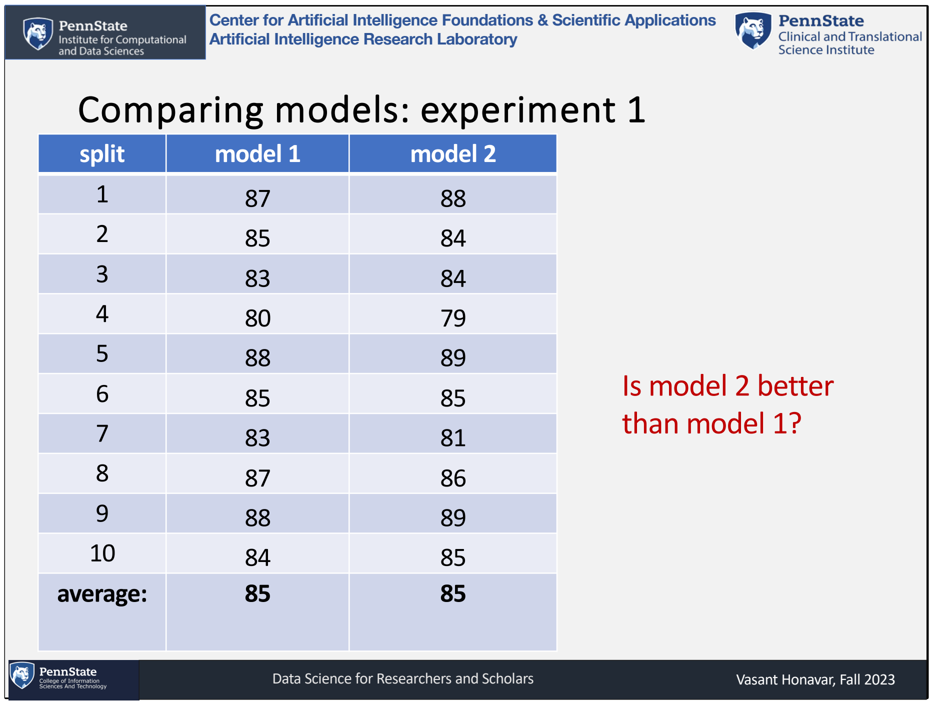
    Predict majority class label in the training data

    True error – 50%;

    Leave-one-out error estimate – 100%!!!!!

Leave-one-out cross validation

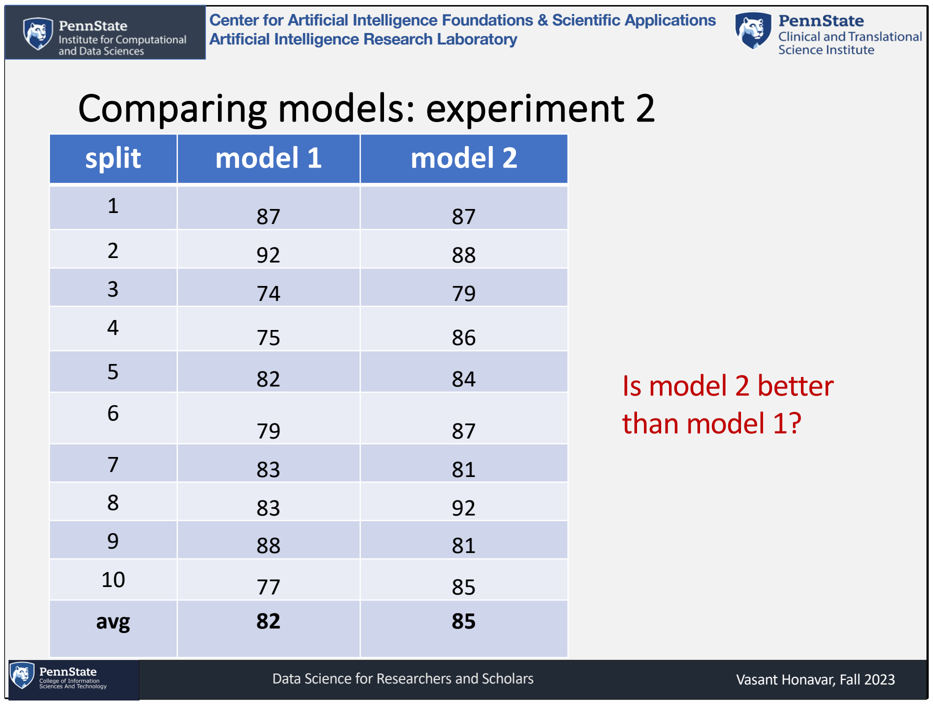
- Can be very expensive if training is slow and/or if there are a large number of examples
- Useful in domains with limited training data
    - maximizes the data we can use for training
- Some classifiers permit the estimation of leave-1-out performance measure without training and testing K models



Comparing models: experiment 1

split	model 1	model 2
1	87	88
2	85	84
3	83	84
4	80	79
5	88	89
6	85	85
7	83	81
8	87	86
9	88	89
10	84	85
average:	**85**	**85**

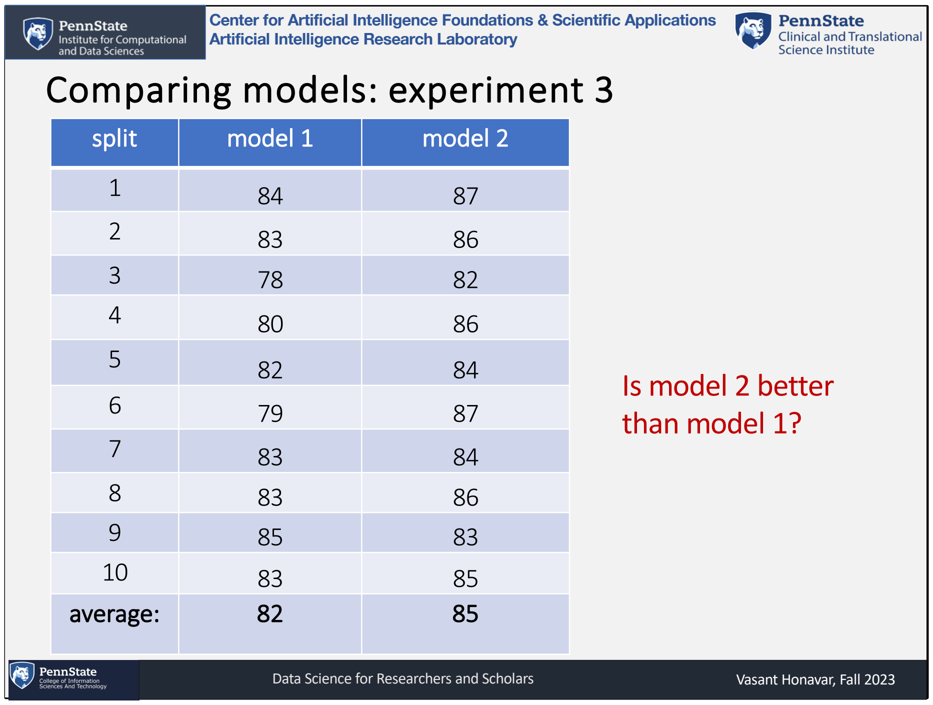
Is model 2 better than model 1?



Comparing models: experiment 2

split	model 1	model 2
1	87	87
2	92	88
3	74	79
4	75	86
5	82	84
6	79	87
7	83	81
8	83	92
9	88	81
10	77	85
avg	**82**	**85**

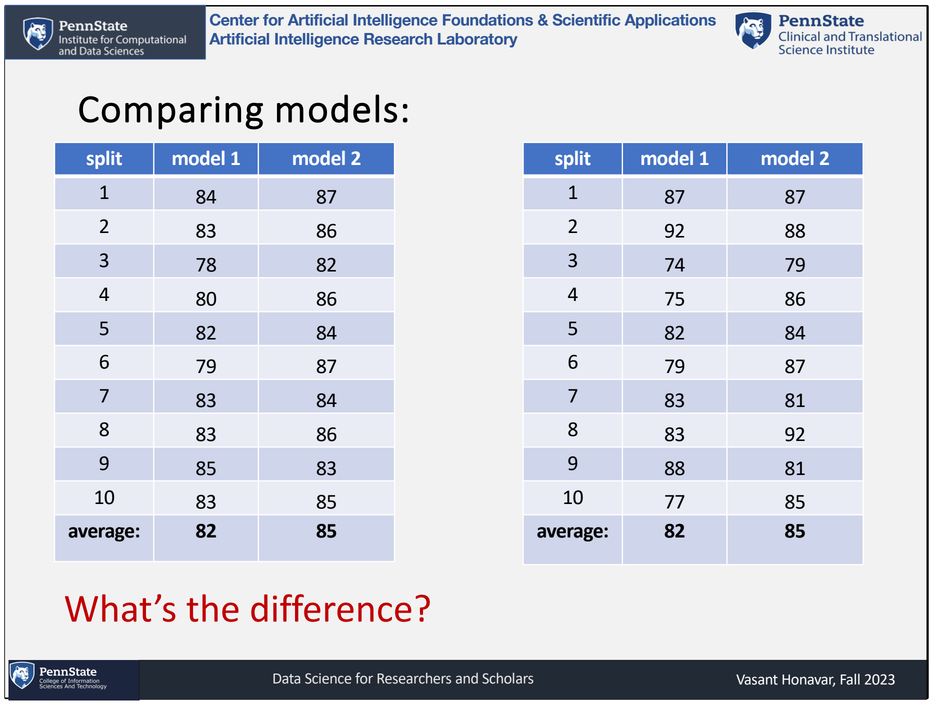
Is model 2 better than model 1?



Comparing models: experiment 3

split	model 1	model 2
1	84	87
2	83	86
3	78	82
4	80	86
5	82	84
6	79	87
7	83	84
8	83	86
9	85	83
10	83	85
average:	**82**	**85**

Is model 2 better than model 1?

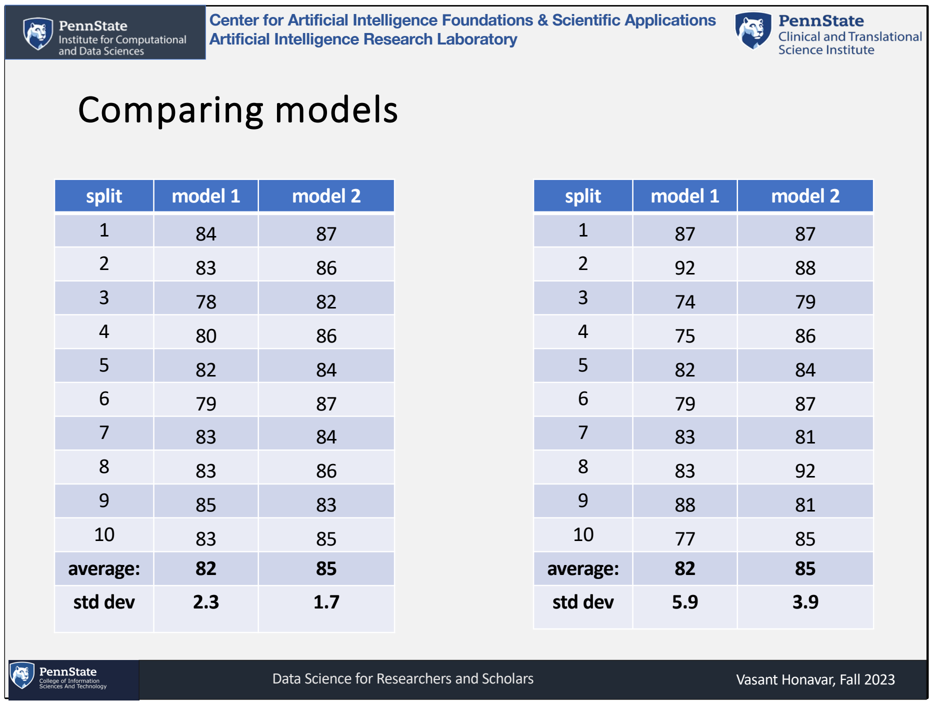


Comparing models:

split	model 1	model 2
1	84	87
2	83	86
3	78	82
4	80	86
5	82	84
6	79	87
7	83	84
8	83	86
9	85	83
10	83	85
average:	**82**	**85**

split	model 1	model 2
1	87	87
2	92	88
3	74	79
4	75	86
5	82	84
6	79	87
7	83	81
8	83	92
9	88	81
10	77	85
average:	**82**	**85**

What's the difference?

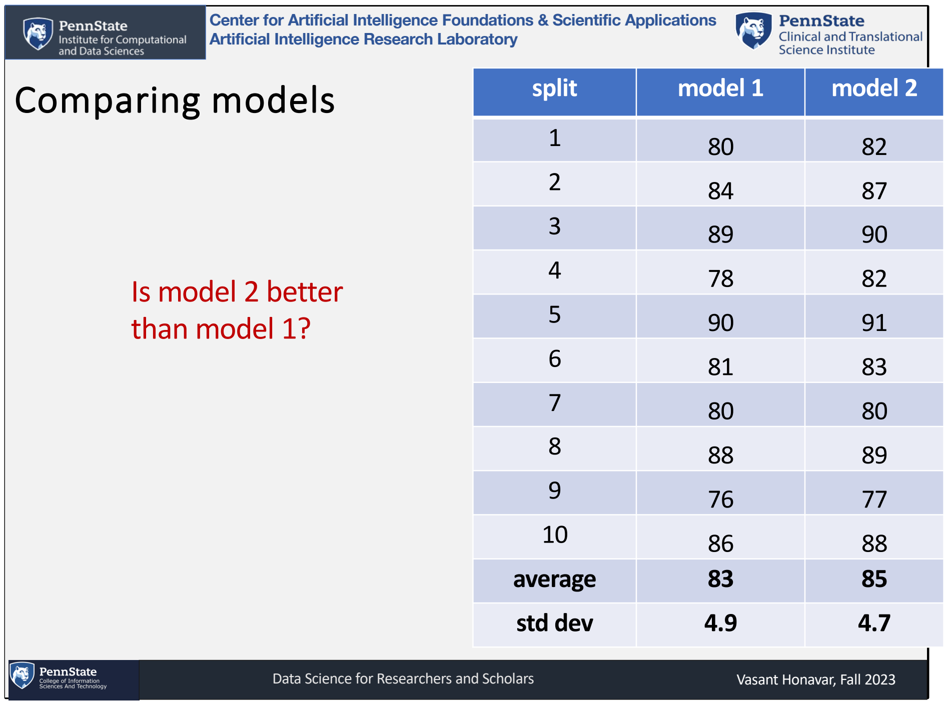


Comparing models

split	model 1	model 2
1	84	87
2	83	86
3	78	82
4	80	86
5	82	84
6	79	87
7	83	84
8	83	86
9	85	83
10	83	85
average:	**82**	**85**
std dev	**2.3**	**1.7**

split	model 1	model 2
1	87	87
2	92	88
3	74	79
4	75	86
5	82	84
6	79	87
7	83	81
8	83	92
9	88	81
10	77	85
average:	**82**	**85**
std dev	**5.9**	**3.9**

Even though the averages are same, the variance is different!



PennState Institute for Computational and Data Sciences

Center for Artificial Intelligence Foundations & Scientific Applications
Artificial Intelligence Research Laboratory

PennState Clinical and Translational Science Institute

Comparing models

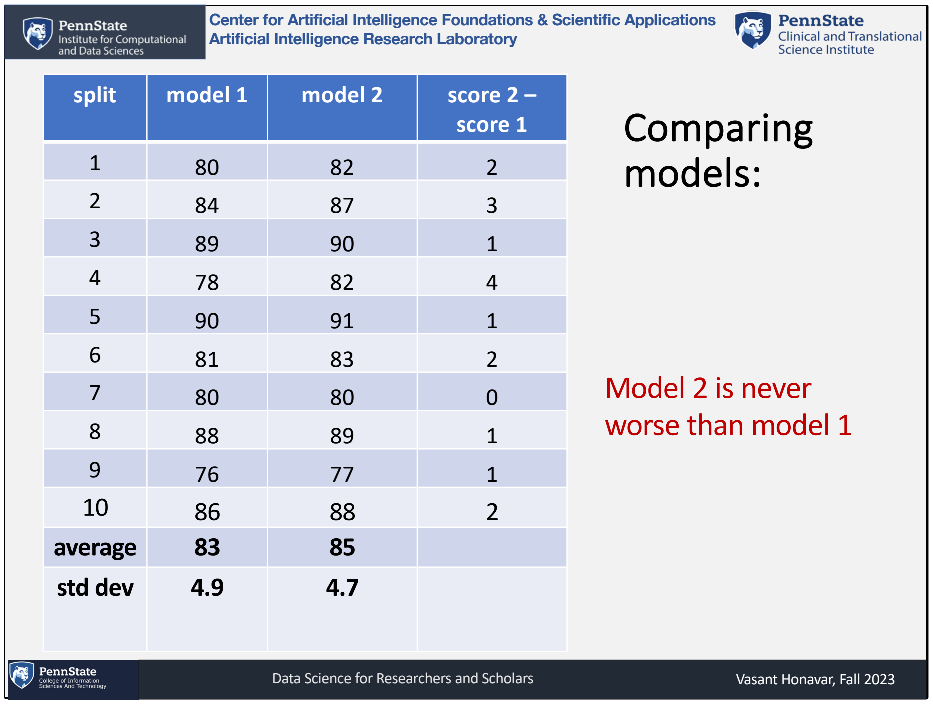
Is model 2 better than model 1?

split	model 1	model 2
1	80	82
2	84	87
3	89	90
4	78	82
5	90	91
6	81	83
7	80	80
8	88	89
9	76	77
10	86	88
average	**83**	**85**
std dev	**4.9**	**4.7**

PennState College of Information Sciences And Technology

Data Science for Researchers and Scholars

Vasant Honavar, Fall 2023



Comparing models:

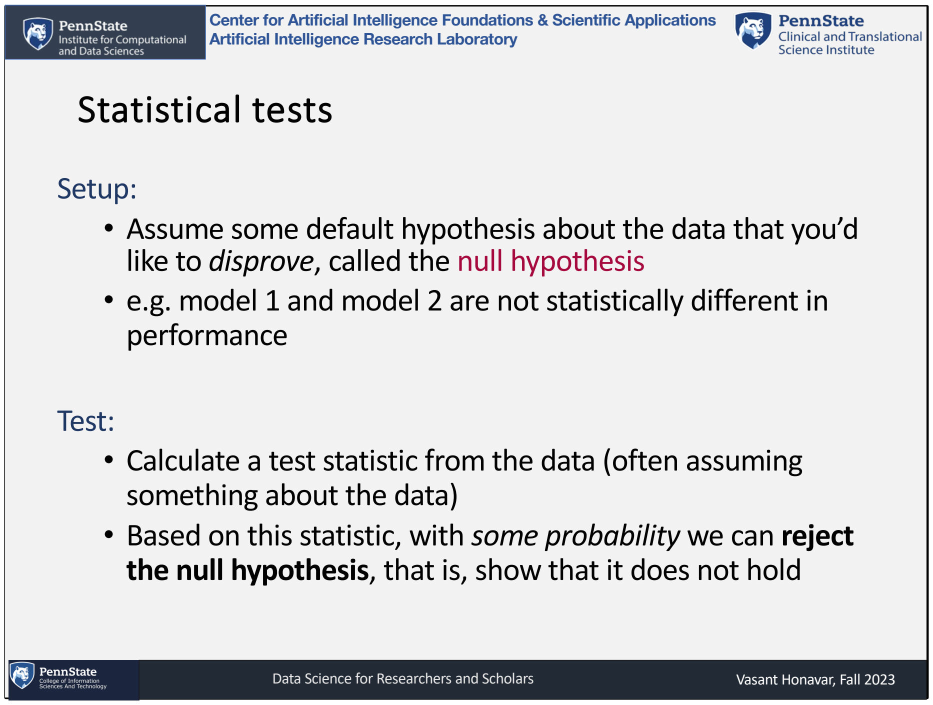
split	model 1	model 2	score 2 – score 1
1	80	82	2
2	84	87	3
3	89	90	1
4	78	82	4
5	90	91	1
6	81	83	2
7	80	80	0
8	88	89	1
9	76	77	1
10	86	88	2
average	**83**	**85**	
std dev	**4.9**	**4.7**	

Model 2 is never worse than model 1

Comparing models

split	model 1	model 2	model 2 – model 1
1	80	82	2
2	84	87	3
3	89	90	1
4	78	82	4
5	90	91	1
6	81	83	2
7	80	80	0
8	88	89	1
9	76	77	1
10	86	88	2
average:	**83**	**85**	
std dev	**4.9**	**4.7**	

How do we decide if model 2 is better than model 1?



Statistical tests

Setup:

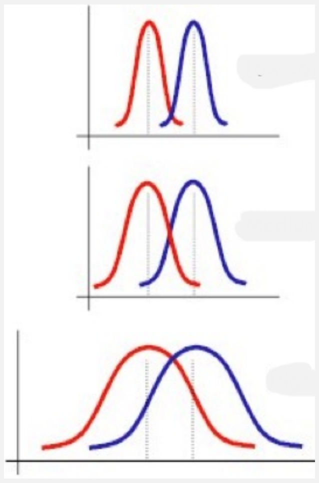
- Assume some default hypothesis about the data that you'd like to *disprove*, called the null hypothesis
- e.g. model 1 and model 2 are not statistically different in performance

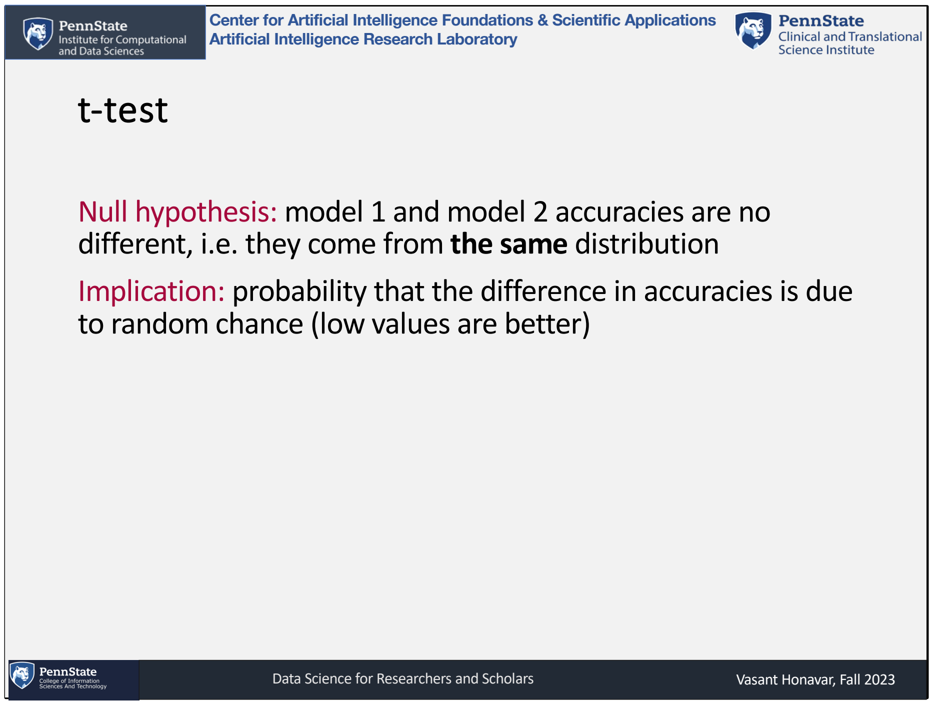
Test:

- Calculate a test statistic from the data (often assuming something about the data)
- Based on this statistic, with *some probability* we can **reject the null hypothesis**, that is, show that it does not hold

t-test

- Tests whether or not two samples come from the same underlying distribution
- In our case, the two distributions of interest are the distributions of performance of two models e.g., on identical K-fold cross-validation runs

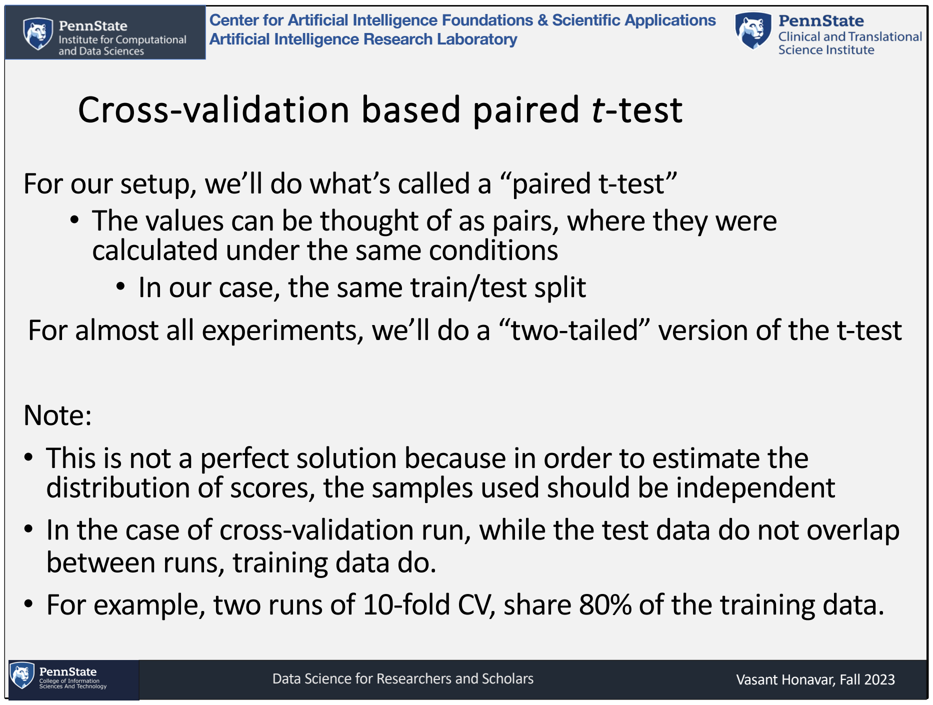




t-test

Null hypothesis: model 1 and model 2 accuracies are no different, i.e. they come from **the same** distribution

Implication: probability that the difference in accuracies is due to random chance (low values are better)



PennState Institute for Computational and Data Sciences

Center for Artificial Intelligence Foundations & Scientific Applications
Artificial Intelligence Research Laboratory

PennState Clinical and Translational Science Institute

Cross-validation based paired *t*-test

For our setup, we'll do what's called a "paired t-test"

- The values can be thought of as pairs, where they were calculated under the same conditions
  - In our case, the same train/test split

For almost all experiments, we'll do a "two-tailed" version of the t-test

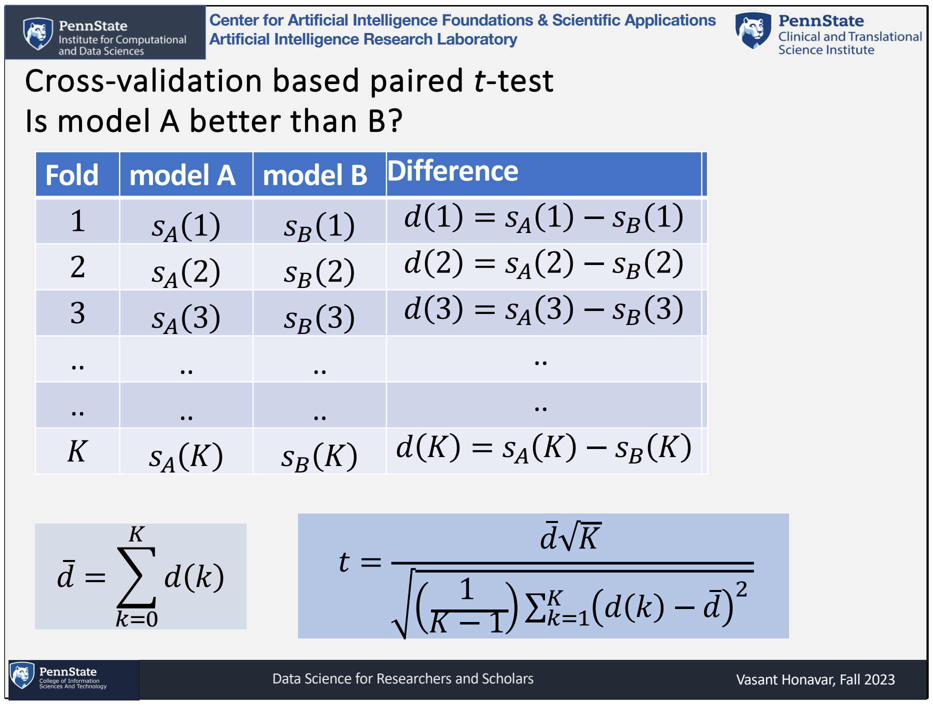
Note:

- This is not a perfect solution because in order to estimate the distribution of scores, the samples used should be independent
- In the case of cross-validation run, while the test data do not overlap between runs, training data do.
- For example, two runs of 10-fold CV, share 80% of the training data.

PennState College of Information Sciences And Technology

Data Science for Researchers and Scholars

Vasant Honavar, Fall 2023



Cross-validation based paired *t*-test
Is model A better than B?

Fold	model A	model B	Difference
1	$s_A(1)$	$s_B(1)$	$d(1) = s_A(1) - s_B(1)$
2	$s_A(2)$	$s_B(2)$	$d(2) = s_A(2) - s_B(2)$
3	$s_A(3)$	$s_B(3)$	$d(3) = s_A(3) - s_B(3)$
..	..	..	..
..	..	..	..
K	$s_A(K)$	$s_B(K)$	$d(K) = s_A(K) - s_B(K)$

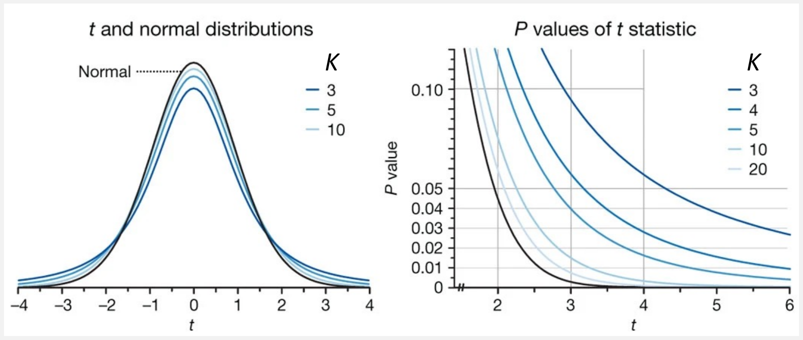
$$\bar{d} = \sum_{k=0}^{K} d(k)$$

$$t = \frac{\bar{d}\sqrt{K}}{\sqrt{\left(\frac{1}{K-1}\right)\sum_{k=1}^{K}\left(d(k) - \bar{d}\right)^2}}$$

p-value

- The result of a statistical test is often a p-value
- p-value: the probability that the null hypothesis holds
  - Specifically, if we re-ran this experiment multiple times (say on different data) what is the probability that we would reject the null hypothesis incorrectly (i.e. the probability we'd be wrong)
  - High p-values are bad. Low p-values are good.
- Common values to consider "significant":
  - 0.05 (95% confident)
  - 0.01 (99% confident)
  - 0.001 (99.9% confident)

t statistic and p-values for different values of K



For a given t, you can read off the corresponding p-value
Or use a p-value calculator which for a given t and $K-1$, returns p

Figure source: Nature Methods volume 10, pages 1041–1042 (2013)

Comparing models

Is model B better than model A?

Compute the *t*-statistic and the *p*-value!

split	model A	model B	Difference
1	80	82	2
2	84	87	3
3	89	90	1
4	78	82	4
5	90	91	1
6	81	83	2
7	80	80	0
8	88	89	1
9	76	77	1
10	86	88	2

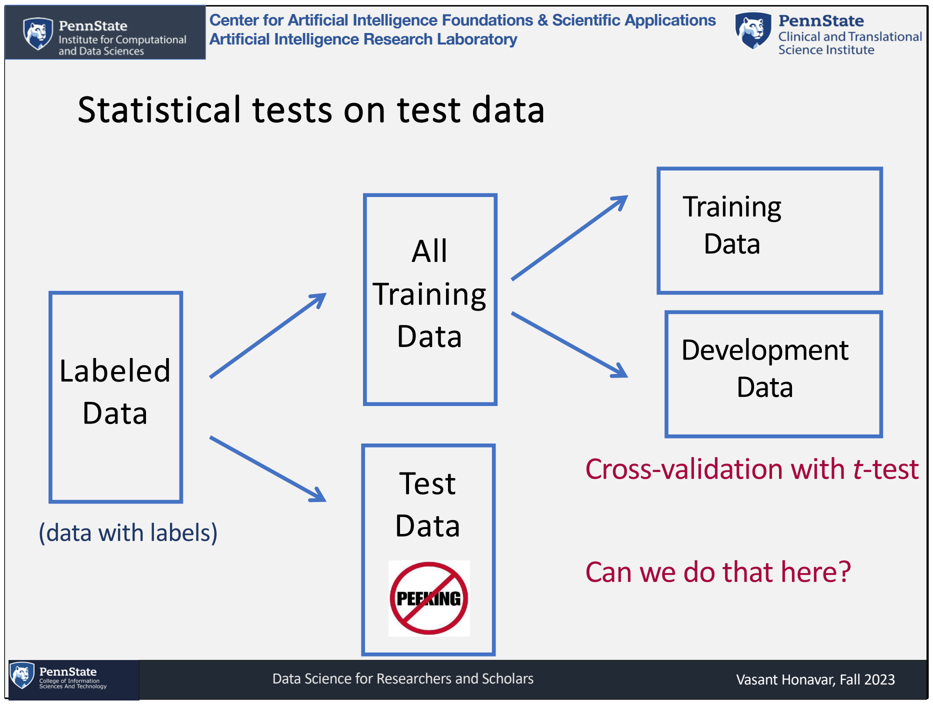
$\bar{d}$=1.7

$$t = \frac{\bar{d}\sqrt{K}}{\sqrt{\left(\frac{1}{K-1}\right)\sum_{k=1}^{K}\left(d(k)-\bar{d}\right)^2}}$$

$$t = \frac{1.7\sqrt{10}}{\sqrt{(1/9)(13.7)}} = 4.3572$$

p-value= 0.001831

Model 2 is statistically significantly better than 1 at p-value < 0.01



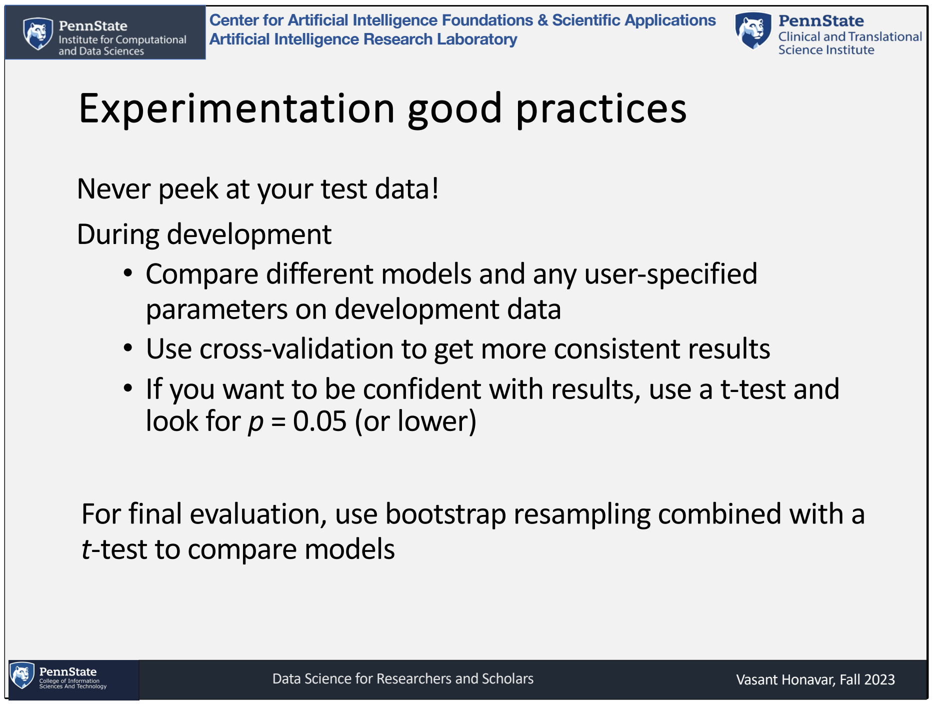
Statistical tests on test data

Labeled Data
(data with labels)
All Training Data
Test Data
PEEKING
Training Data
Development Data

Cross-validation with *t*-test

Can we do that here?

No… the problem is that we only have one test set and we can’t resample, etc. because then we’ll have looked at the test data!



Experimentation good practices

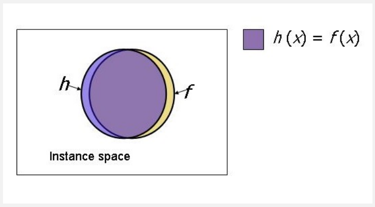
Never peek at your test data!

During development

- Compare different models and any user-specified parameters on development data
- Use cross-validation to get more consistent results
- If you want to be confident with results, use a t-test and look for $p = 0.05$ (or lower)

For final evaluation, use bootstrap resampling combined with a *t*-test to compare models

How close is the estimated error to true error?



The *true* error of a hypothesis *h* with respect to a target function *f* and an instance distribution *D* is

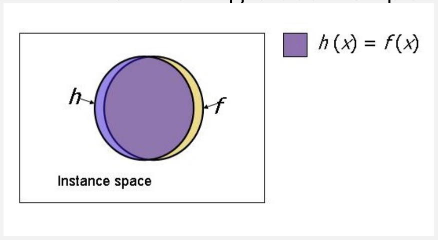
$$Error_D(h) \equiv \Pr_{x \in D}\left[f(x) \neq h(x) \right]$$

The sample error of a binary classifier *h* with respect to a target function *f* and an instance distribution D is

$$Error_S(h) \equiv \frac{1}{|S|} \sum_{x \in S} \delta(f(x) \neq h(x))$$

$$\delta(a,b) \equiv 1 \text{ iff } a \neq b; \delta(a,b) \equiv 0 \text{ otherwise}$$

Estimating classifier performance



$$Domain(X) = \{a,b,c,d\}$$

$$D(X) = \left\{\frac{1}{8},\frac{1}{2},\frac{1}{8},\frac{1}{4}\right\}$$

$$\begin{array}{ccccc} x & a & b & c & d \\ h(x) & 0 & 1 & 1 & 0 \\ f(x) & 1 & 1 & 0 & 0 \end{array}$$

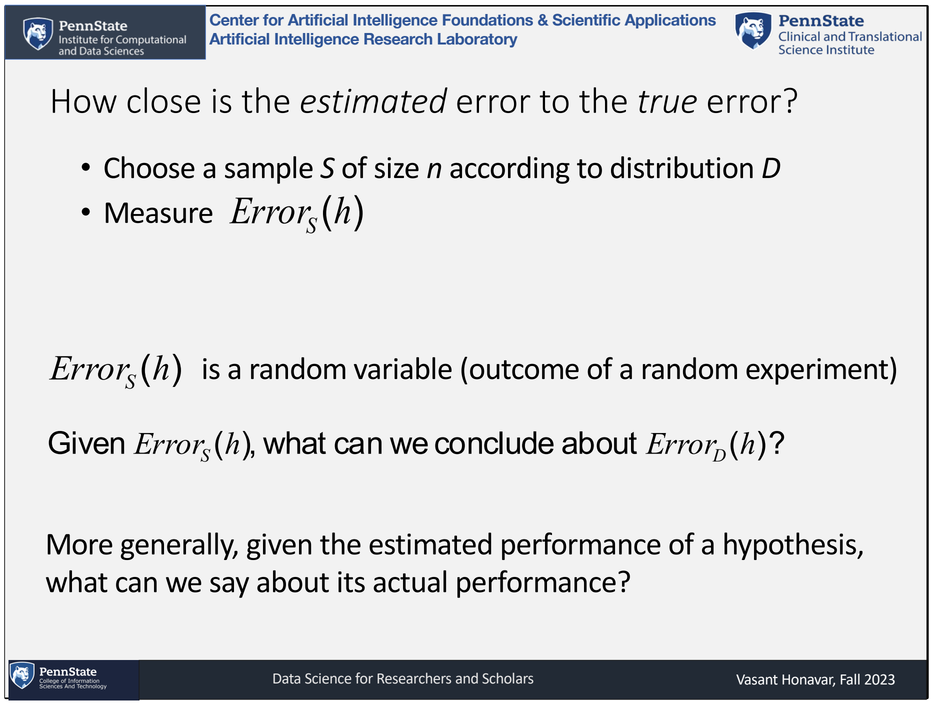
$$error_D(h) = \Pr_D[h(x) \neq f(x)]$$
$$= D(X = a) + D(X = c)$$
$$= \frac{1}{8} + \frac{1}{8} = \frac{1}{4}$$

Evaluating the performance of a classifier

- Sample error estimated from training data is an *optimistic* estimate $Bias \equiv E[Error_S(h)] - Error_D(h)$

- For an *unbiased* estimate, *h* must be evaluated on an independent sample *S* (which is not the case if *S* is the training set!)
- Even when the estimate is unbiased, it can *vary* across samples!
- If *h* misclassifies 8 out of 100 samples $Error_S(h) = \frac{8}{100} = 0.08$

How close is the sample error to the true error?



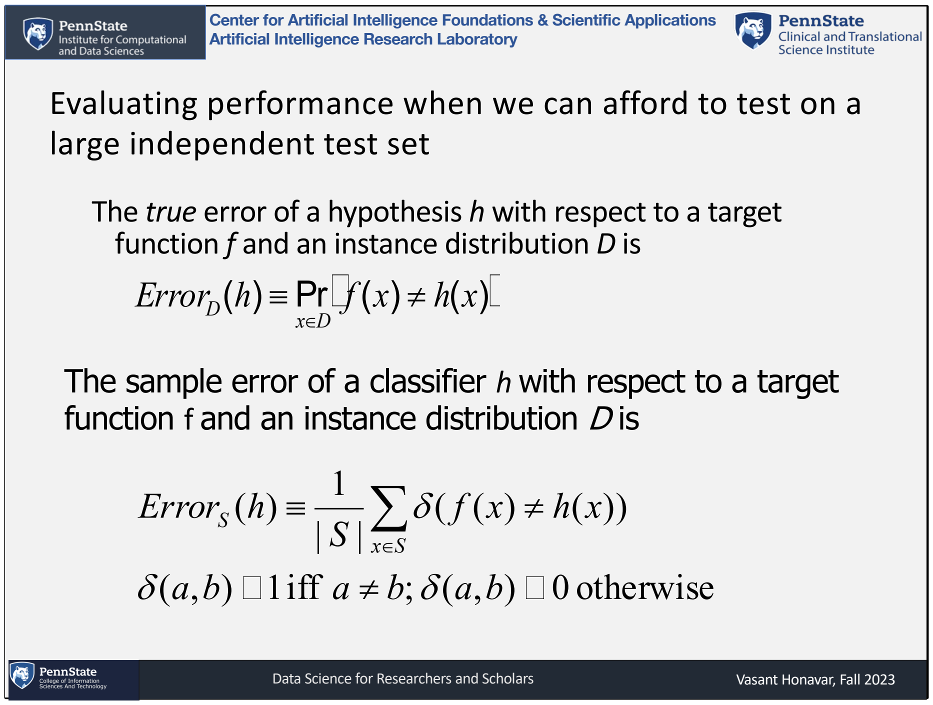
How close is the *estimated* error to the *true* error?

- Choose a sample *S* of size *n* according to distribution *D*
- Measure $Error_S(h)$

$Error_S(h)$ is a random variable (outcome of a random experiment)

Given $Error_S(h)$, what can we conclude about $Error_D(h)$?

More generally, given the estimated performance of a hypothesis, what can we say about its actual performance?



PennState Institute for Computational and Data Sciences

Center for Artificial Intelligence Foundations & Scientific Applications
Artificial Intelligence Research Laboratory

PennState Clinical and Translational Science Institute

Evaluating performance when we can afford to test on a large independent test set

The *true* error of a hypothesis *h* with respect to a target function *f* and an instance distribution *D* is

$$Error_D(h) \equiv \Pr_{x \in D}\left[f(x) \neq h(x)\right]$$

The sample error of a classifier *h* with respect to a target function f and an instance distribution *D* is

$$Error_S(h) \equiv \frac{1}{|S|}\sum_{x \in S}\delta(f(x) \neq h(x))$$

$$\delta(a,b) = 1 \text{ iff } a \neq b; \delta(a,b) = 0 \text{ otherwise}$$

PennState College of Information Sciences And Technology

Data Science for Researchers and Scholars

Vasant Honavar, Fall 2023

Evaluating Classifier performance

$$Bias \equiv E[Error_S(h)] - Error_D(h)$$

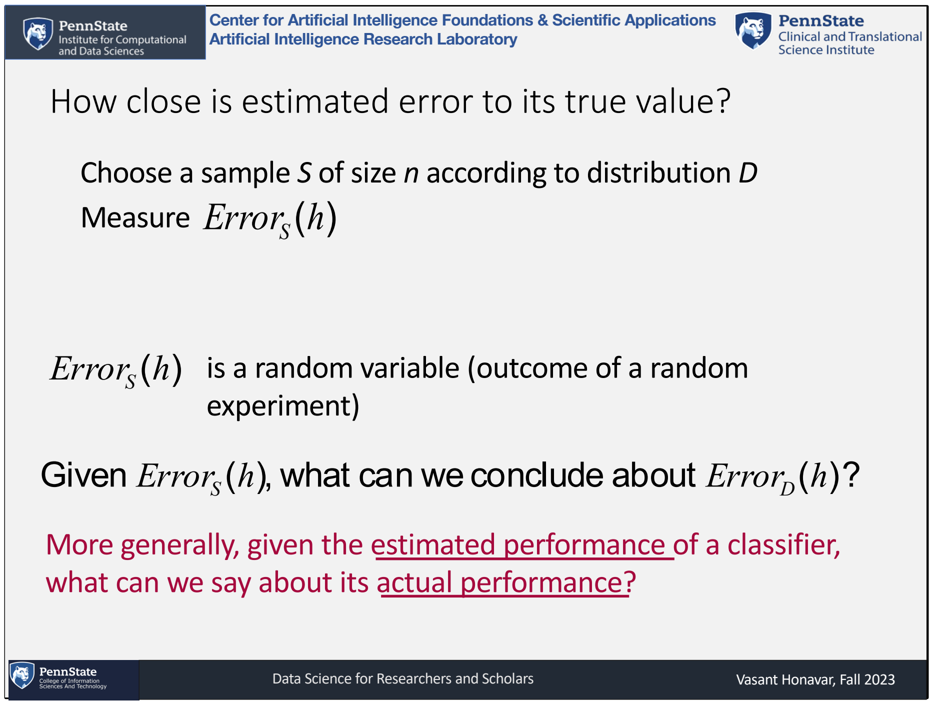
Sample error estimated from training data is an *optimistic* estimate

For an *unbiased* estimate, h must be evaluated on an independent sample S (which is not the case if S is the training set!)

Even when the estimate is unbiased, it can *vary* across samples!

If *h* misclassifies 8 out of 100 samples $Error_S(h) = \frac{8}{100} = 0.08$

How close is the sample error to the true error?



How close is estimated error to its true value?

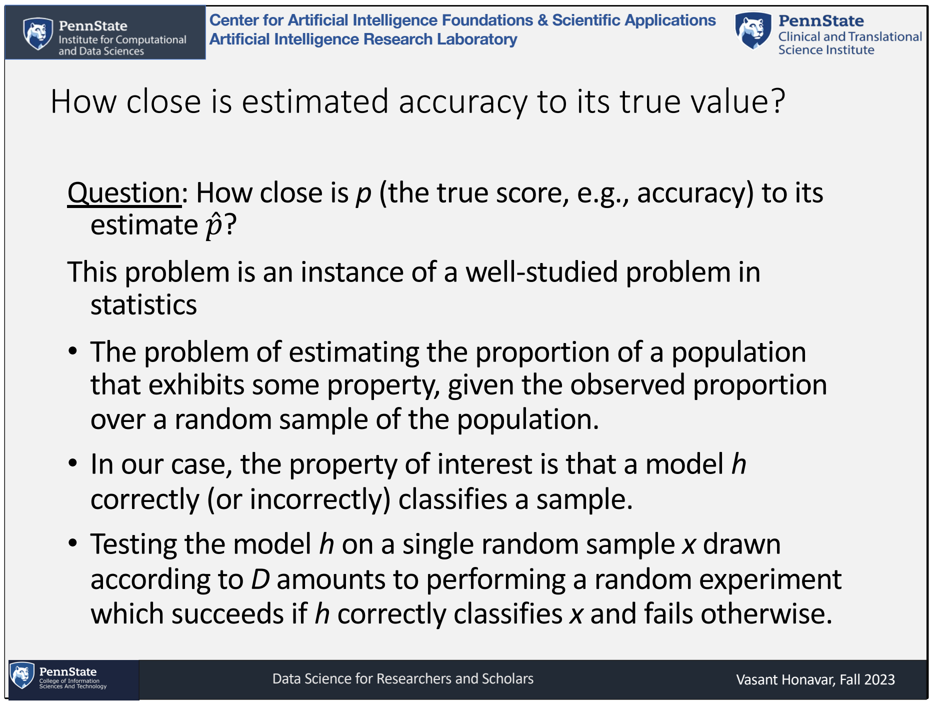
Choose a sample *S* of size *n* according to distribution *D*

Measure $Error_S(h)$

$Error_S(h)$ is a random variable (outcome of a random experiment)

Given $Error_S(h)$, what can we conclude about $Error_D(h)$?

More generally, given the estimated performance of a classifier, what can we say about its actual performance?

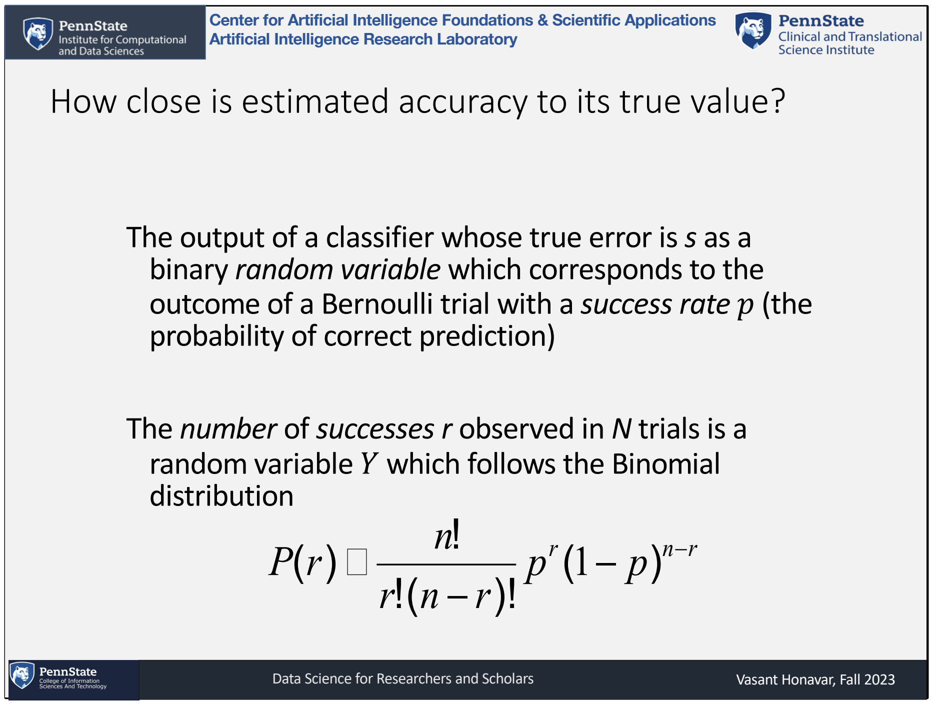


How close is estimated accuracy to its true value?

Question: How close is *p* (the true score, e.g., accuracy) to its estimate $\hat{p}$?

This problem is an instance of a well-studied problem in statistics

- The problem of estimating the proportion of a population that exhibits some property, given the observed proportion over a random sample of the population.
- In our case, the property of interest is that a model *h* correctly (or incorrectly) classifies a sample.
- Testing the model *h* on a single random sample *x* drawn according to *D* amounts to performing a random experiment which succeeds if *h* correctly classifies *x* and fails otherwise.



How close is estimated accuracy to its true value?

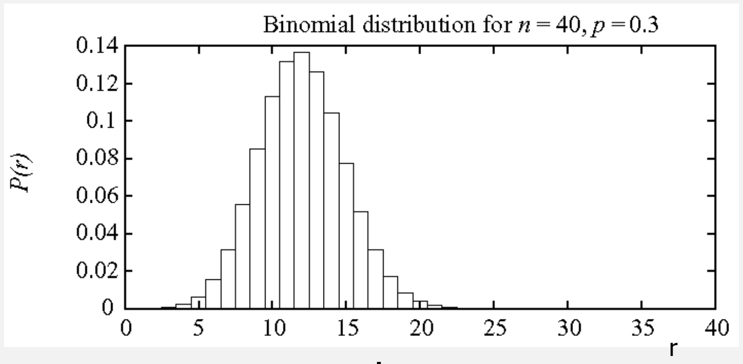
The output of a classifier whose true error is *s* as a binary *random variable* which corresponds to the outcome of a Bernoulli trial with a *success rate* p (the probability of correct prediction)

The *number* of *successes r* observed in *N* trials is a random variable Y which follows the Binomial distribution

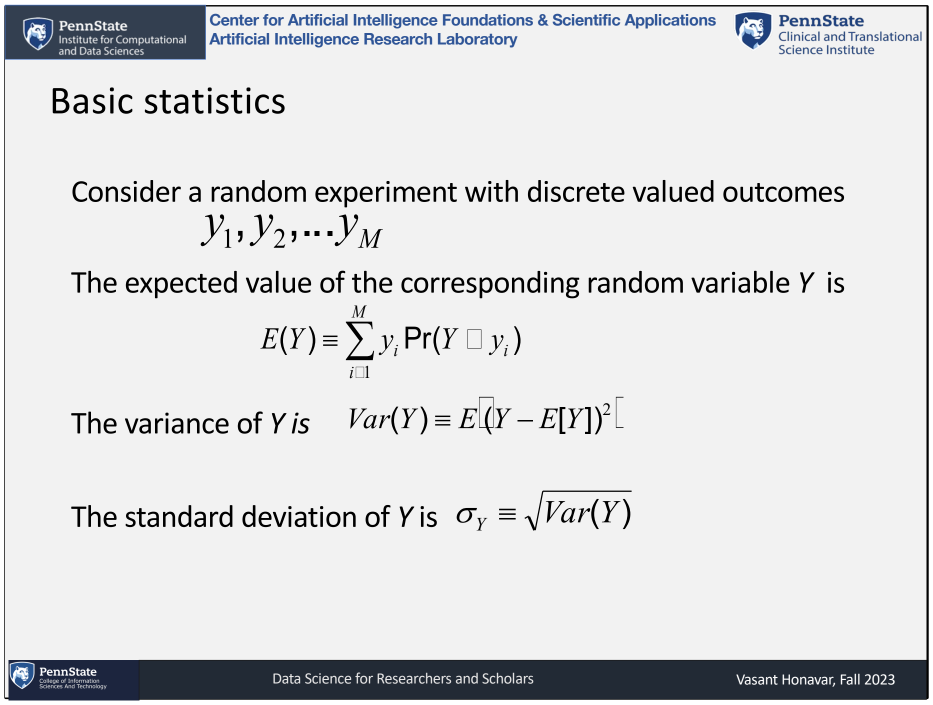
$$P(r) \square \frac{n!}{r!(n-r)!} p^r (1-p)^{n-r}$$

$Error_S(h)$ is a Random Variable

Probability of observing r misclassified examples in a sample of size n:



$$P(r) \square \frac{n!}{r!(n-r)!} p^r (1-p)^{n-r}$$



Basic statistics

Consider a random experiment with discrete valued outcomes

$$y_1, y_2, \ldots y_M$$

The expected value of the corresponding random variable *Y* is

$$E(Y) \equiv \sum_{i=1}^{M} y_i \Pr(Y = y_i)$$

The variance of *Y is* $Var(Y) \equiv E\left[(Y - E[Y])^2\right]$

The standard deviation of *Y* is $\sigma_Y \equiv \sqrt{Var(Y)}$

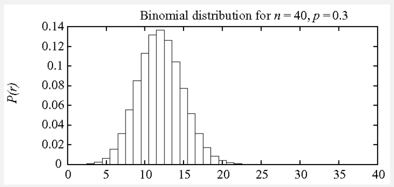
How close is estimated accuracy to its true value?

- The *mean* of a Bernoulli trial with success rate *p* = *p*
- *Variance* = *p* (*1*-*p*)

If *N trials* are taken from the same Bernoulli process, the observed success rate $\hat{p}$ has the same mean *p* and variance $\frac{p(1-p)}{N}$

For large *N*, the distribution of $\hat{p}$ follows a Gaussian distribution

Binomial Probability Distribution



$$P(r) = \frac{n!}{r!(n-r)!} p^r (1-p)^{n-r}$$

Probability $P(r)$ of r heads in n coin flips, if p = Pr(heads)

- Expected, or mean value of X, $E[X]$, is

$$E[X] \equiv \sum_{i=0}^{N} iP(i) = np$$

- Variance of X is

$$Var(X) \equiv E[(X - E[X])^2] = np(1-p)$$

- Standard deviation of X, σ_X, is

$$\sigma_X \equiv \sqrt{E[(X - E[X])^2]} = \sqrt{np(1-p)}$$

Estimators, Bias, Variance, Confidence Interval

$$Error_S(h) \square \frac{r}{n}$$

$$Error_D(h) \square p$$

$$\sigma_{Error_S(h)} \square \sqrt{\frac{p(1-p)}{n}}$$

$$\sigma_{Error_S(h)} \square \sqrt{\frac{Error_D(h)(1-Error_D(h))}{n}}$$

$$\sigma_{Error_S(h)} \approx \sqrt{\frac{Error_S(h)(1-Error_S(h))}{n}}$$

An $N\%$ confidence interval for some parameter p that is the interval which is expected with probability $N\%$ to contain p

Normal distribution approximates binomial

$Error_S(h)$ follows a Binomial distribution, with

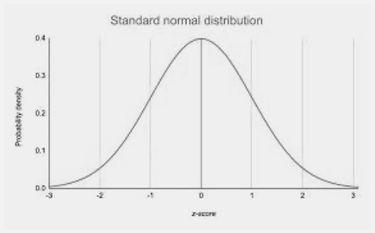
- mean $\square_{Error_S(h)} \square Error_D(h)$
- standard deviation

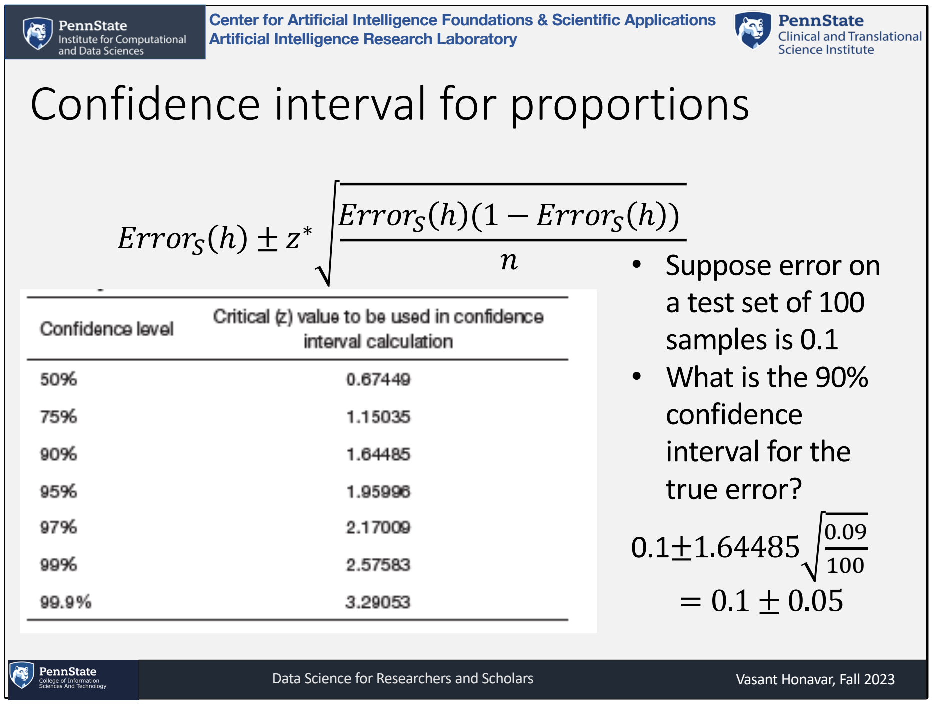
$$\sigma_{Error_S(h)} \approx \sqrt{\frac{Error_S(h)(1 - Error_S(h))}{n}}$$

We can approximate this by a Normal distribution with the same mean and variance when $np(1-p) \geq 5$

Say $p = 0.1$ Then $n \geq \frac{5}{(0.1)(0.9)} \approx 55$

For typical values of p (classification error) and n (test set size), $np(1-p) \geq 5$





Confidence interval for proportions

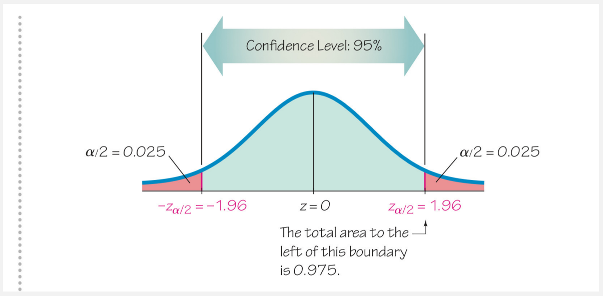
$$Error_S(h) \pm z^* \sqrt{\frac{Error_S(h)(1 - Error_S(h))}{n}}$$

Confidence level	Critical (z) value to be used in confidence interval calculation
50%	0.67449
75%	1.15035
90%	1.64485
95%	1.95996
97%	2.17009
99%	2.57583
99.9%	3.29053

- Suppose error on a test set of 100 samples is 0.1
- What is the 90% confidence interval for the true error?

$$0.1 \pm 1.64485 \sqrt{\frac{0.09}{100}}$$
$$= 0.1 \pm 0.05$$

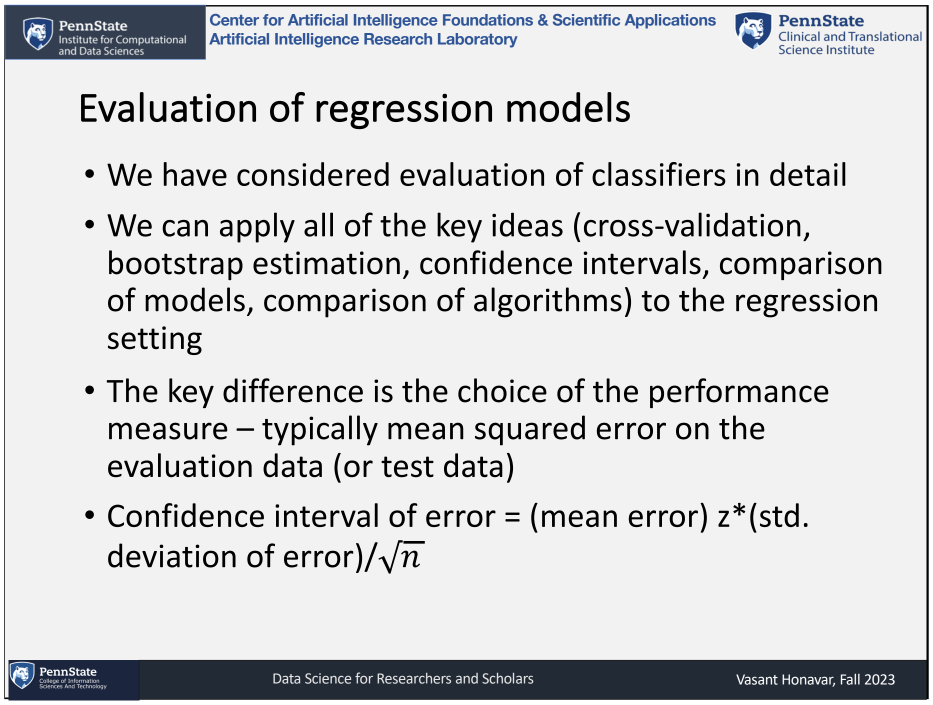
One sided confidence interval



Sometimes we are interested in the confidence associated with the upper bound on error.

In the above example, we can be 97.5% confident that the error is not greater than

$$Error_S(h) + z^* \sqrt{\frac{Error_S(h)(1 - Error_S(h))}{n}}$$

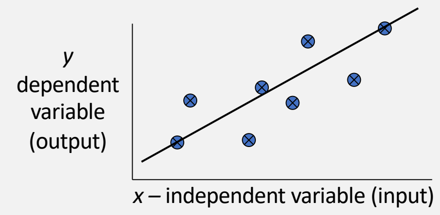


Evaluation of regression models

- We have considered evaluation of classifiers in detail
- We can apply all of the key ideas (cross-validation, bootstrap estimation, confidence intervals, comparison of models, comparison of algorithms) to the regression setting
- The key difference is the choice of the performance measure – typically mean squared error on the evaluation data (or test data)
- Confidence interval of error = (mean error) z*(std. deviation of error)/$\sqrt{n}$

Regression

- For classification the output is nominal
- In regression the output is continuous
- Linear regression is perhaps the simplest approach
  - Fit data with the best hyper-plane (line when the function is defined with respect to a single variable) which "goes through" the points


y
dependent
variable
(output)
x – independent variable (input)