

Data Mining & Machine Learning

CS37300

Purdue University

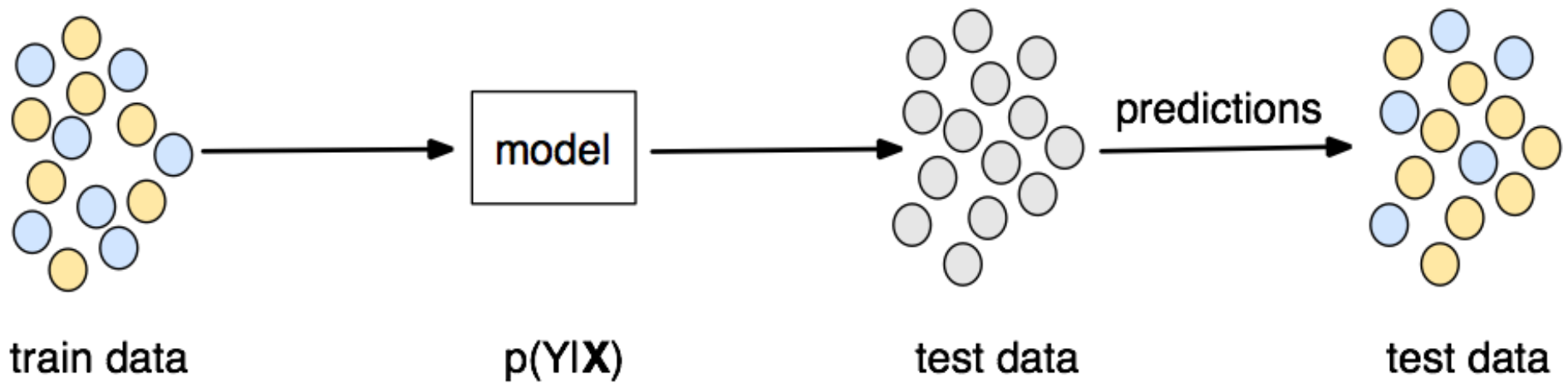
October 16, 2017

Ensemble methods

Ensemble methods

- Motivation
 - Too difficult to construct a single model that optimizes performance (why?)
- Approach
 - Construct many models on different versions of the training set and combine them during prediction
- **Goal:** *reduce bias and/or variance*

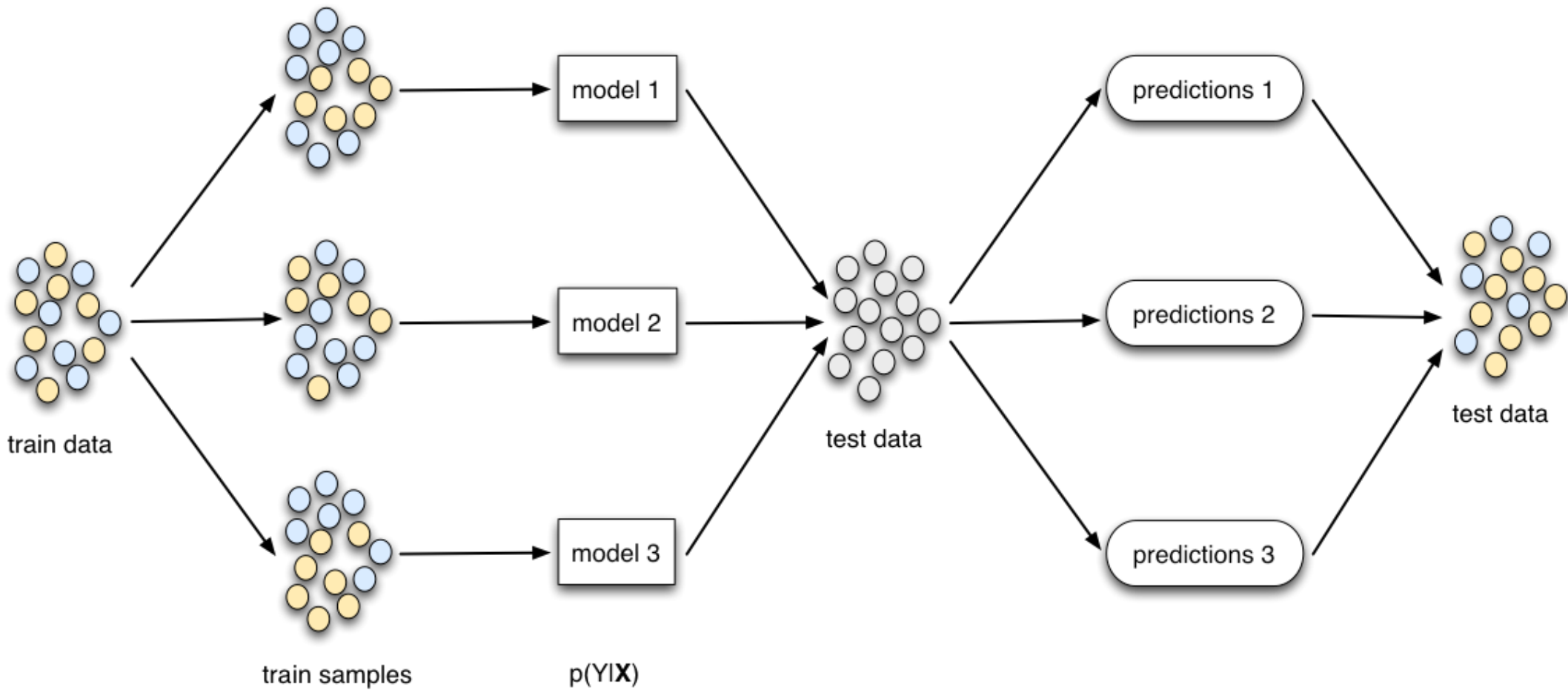
Conventional classification



X: attributes

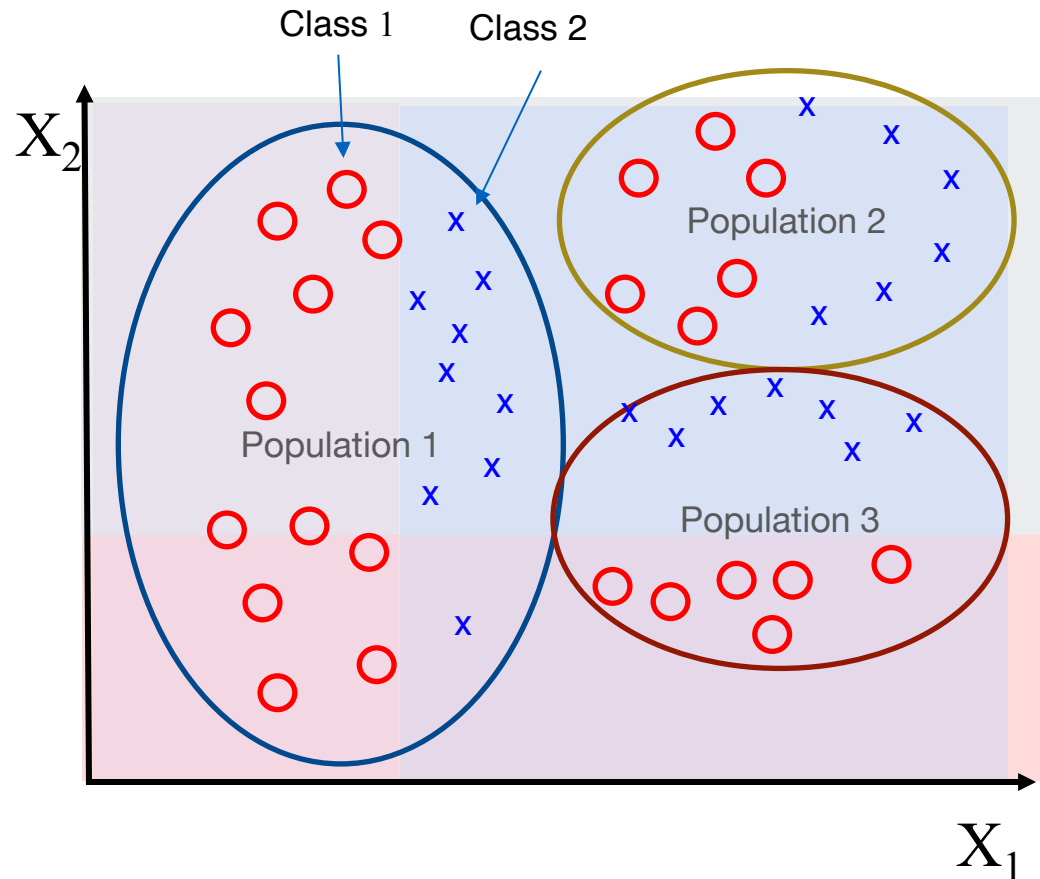
Y: class label

Ensemble classification



Understanding Ensembles

The Data



Three simple classification problems if we can break them down

If no classes, clustering problem is also simple if clustering algorithm knows shape of cluster

Why just not choose the **Best** classifier?

- As we have seen throughout the course, choosing the "best" classifier is a tricky business
 - *Why select just one of multiple similarly good hypotheses*
- Ensembles are a way to include multiple hypotheses in the decision
- Can be justified by Bias-Variance reduction:
 - *Variance*: error from sensitivity to small fluctuations in the training set
 - *Bias*: erroneous assumptions in the model

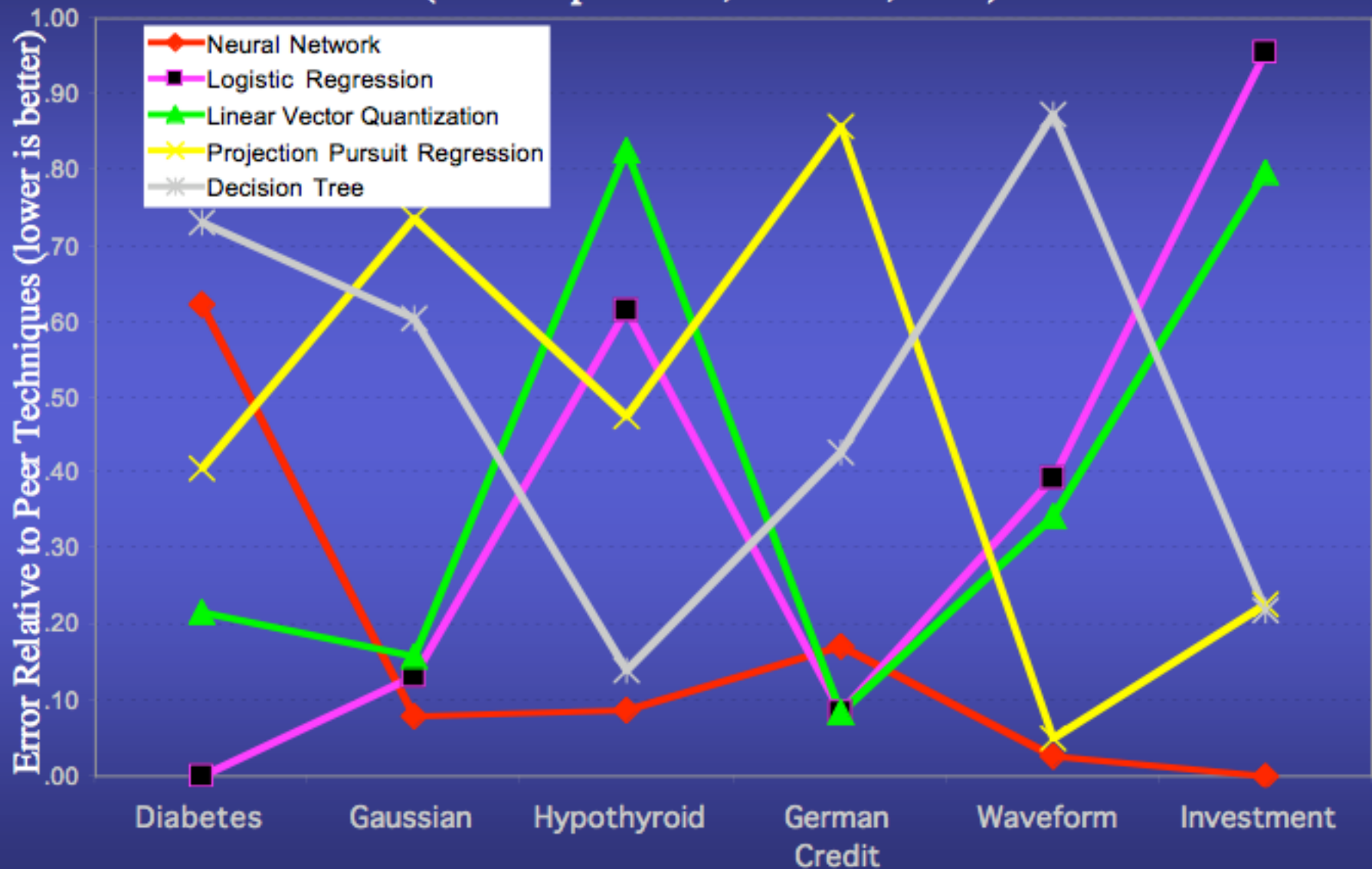
Bias-Variance Reduction

- *Variance reduction*: if the training sets are completely independent, it will always help to average an ensemble to reduce variance without affecting bias (e.g., bagging). Reduces sensitivity to individual data points
- *Bias reduction*: for simple models, average of models has much greater model complexity than single model (e.g., hyperplane classifiers, Gaussian densities).
 - Averaging models can reduce bias substantially by increasing model complexity, and control variance by fitting one component at a time (e.g., boosting)

Advantages

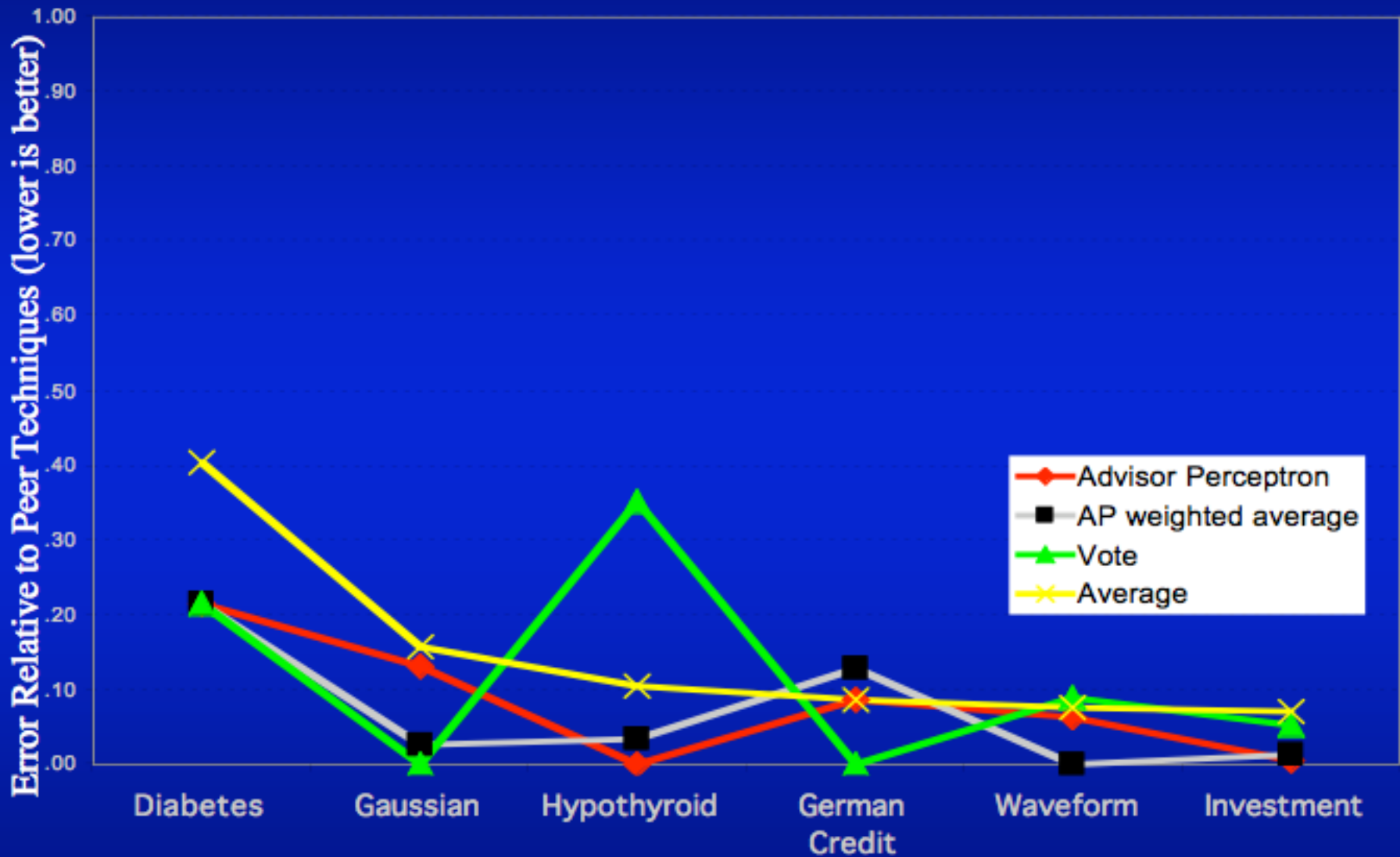
- No need to select best model
- Put all models to work, make them work together
- Ensembles are generally more accurate than any individual members:
 - As long as constituent members of the ensemble are
 - Accurate (better than random guessing)
 - Diverse (different errors on new examples)

Relative Performance Examples: 5 Algorithms on 6 Datasets (with Stephen Lee, U. Idaho, 1997)



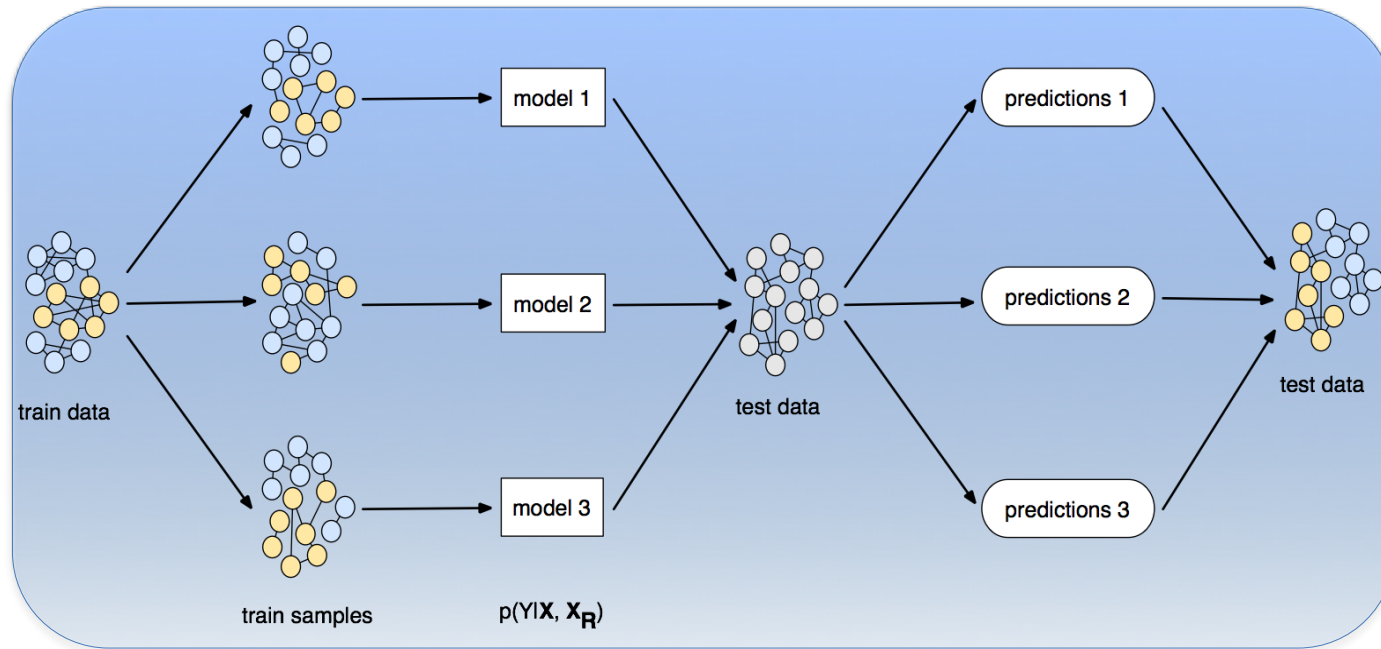
source: *Top Ten Data Mining Mistakes*, John Edler, (Edler Research)

Essentially every *ensemble* method improves performance

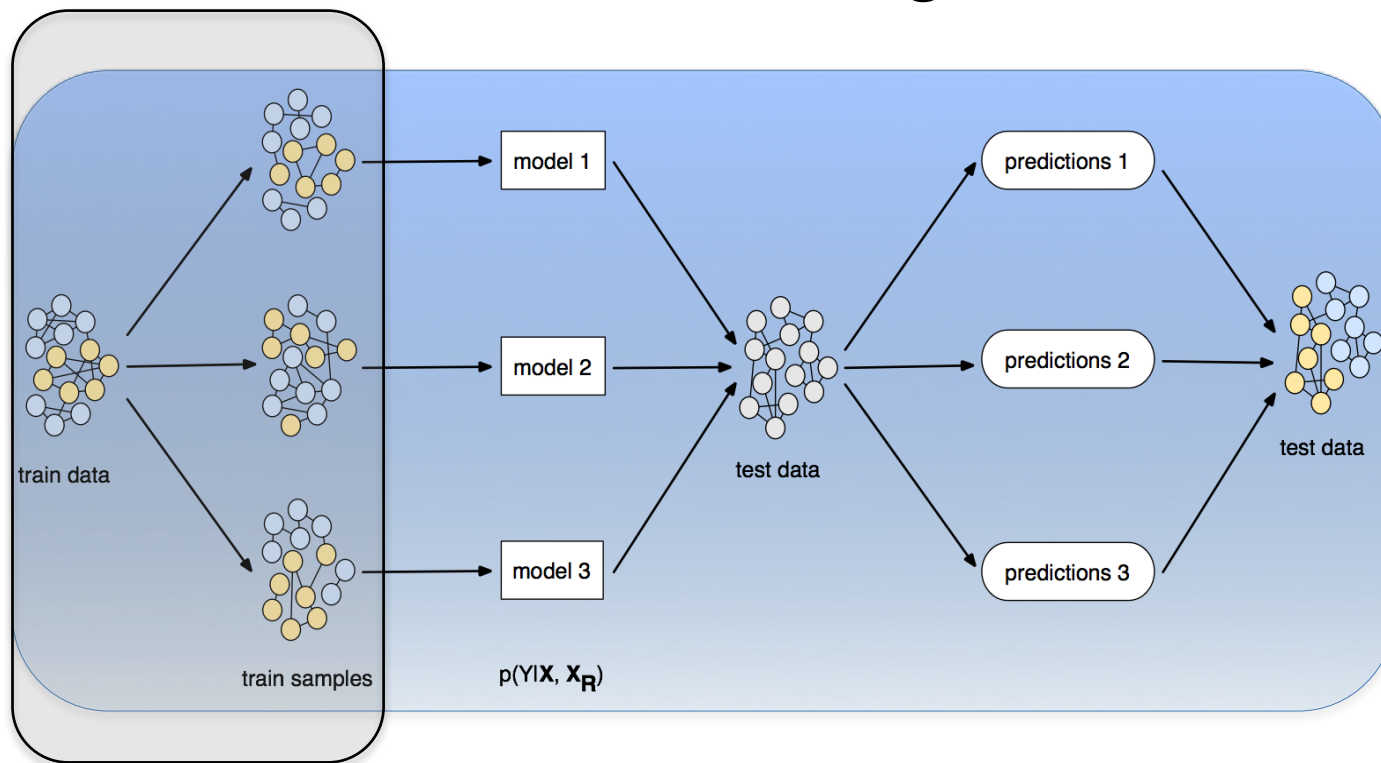


source: *Top Ten Data Mining Mistakes*, John Edler, (Edler Research)

Ensemble design



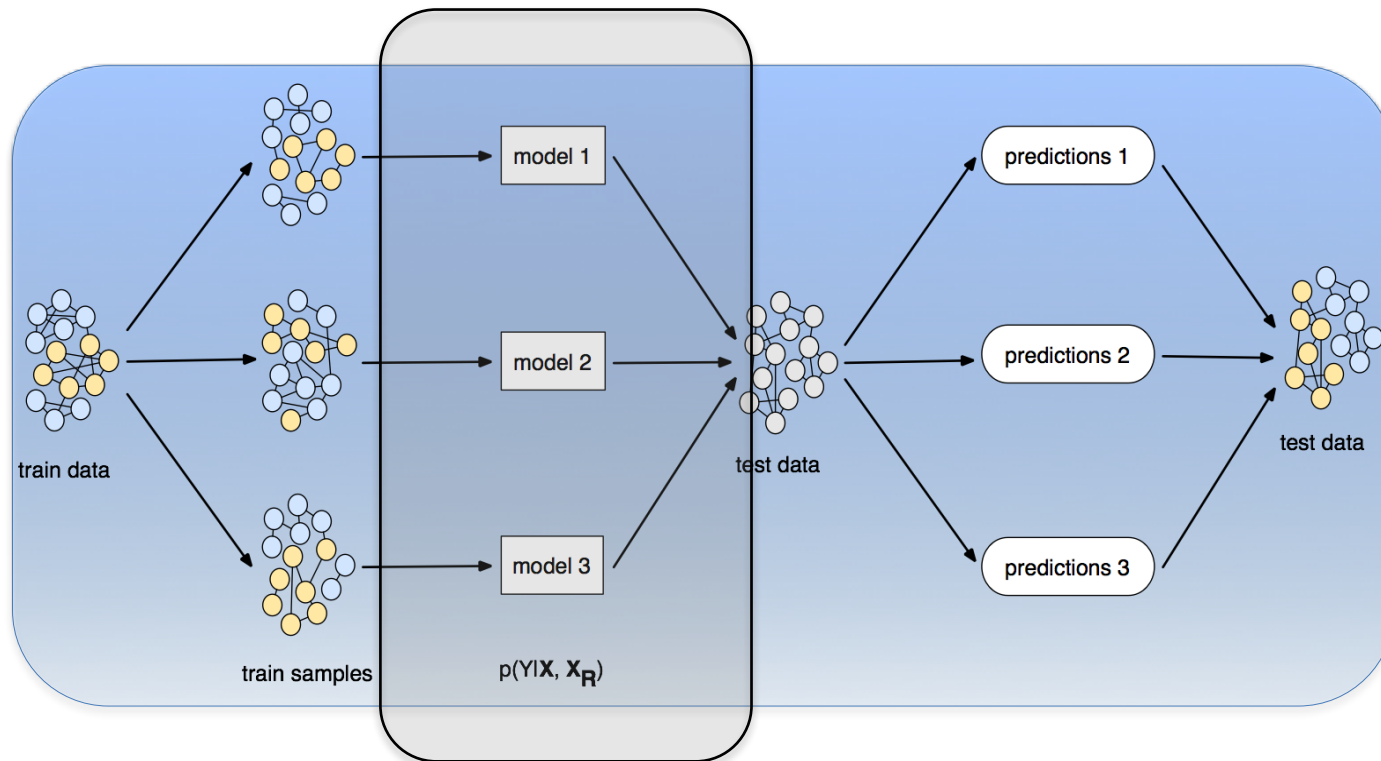
Ensemble design



TREATMENT OF INPUT DATA

- *sampling*
- *variable selection*

Ensemble design



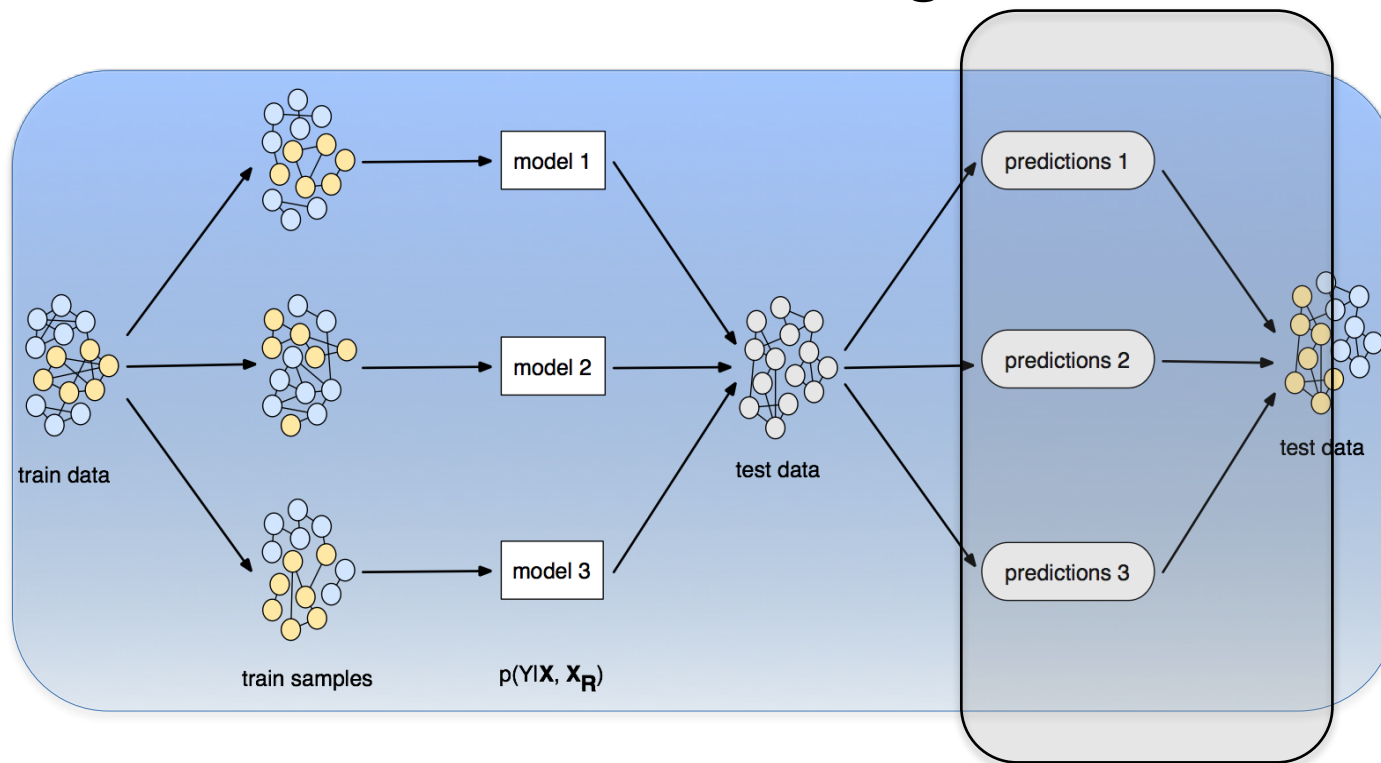
TREATMENT OF INPUT DATA

- *sampling*
- *variable selection*

CHOICE OF BASE CLASSIFIER

- *decision tree*
- *perceptron*
- ...

Ensemble design



TREATMENT OF INPUT DATA

- *sampling*
- *variable selection*

CHOICE OF BASE CLASSIFIER

- *decision tree*
- *perceptron*
- ...

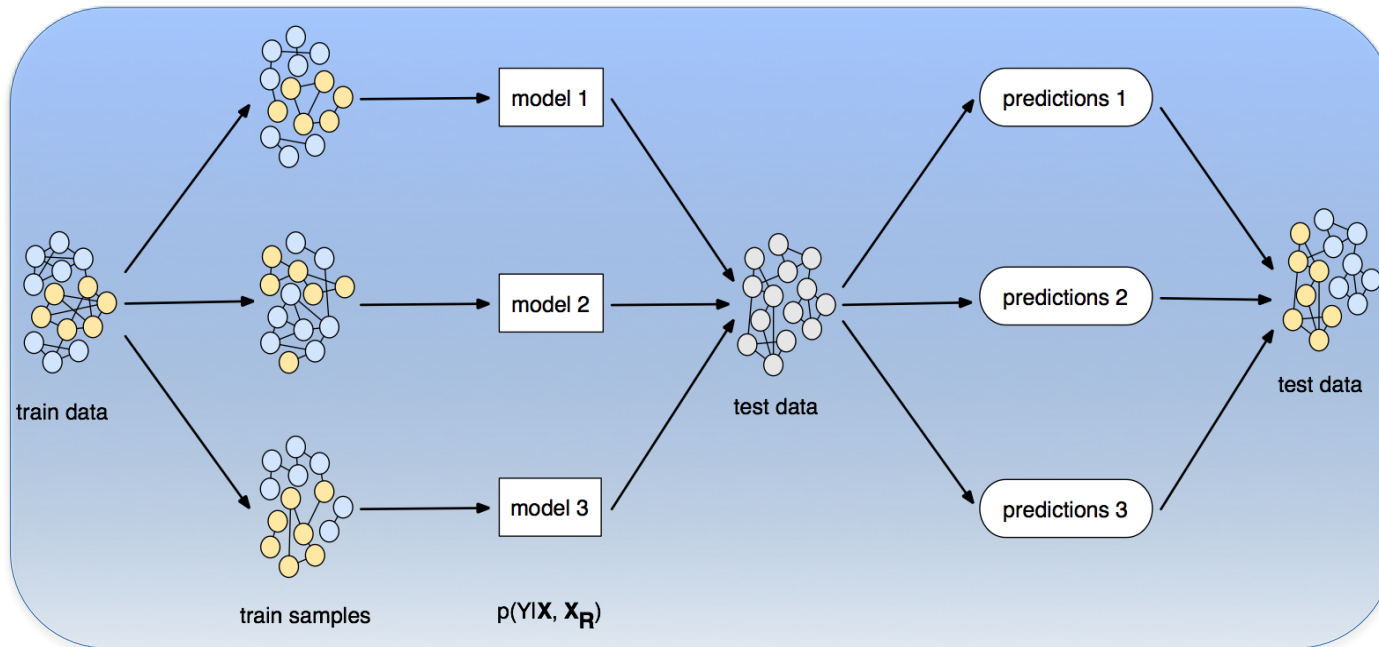
PREDICTION AGGREGATION

- *averaging*
- *weighted vote*
- ...

Bagging

- **B**ootstrap **agg**regating
- Main assumption
 - Combining many *unstable* predictors in an ensemble produces a *stable* predictor (i.e., reduces variance)
 - Unstable predictor: small changes in training data produces large changes in the model (e.g., trees)
- Model space: non-parametric, can model any function if an appropriate base model is used

Bagging



TREATMENT OF INPUT DATA

- *sample with replacement*

CHOICE OF BASE CLASSIFIER

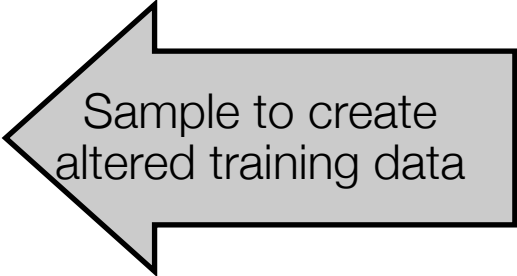
- *unstable predictor*
e.g., decision tree

PREDICTION AGGREGATION

- *averaging*

Bagging

- Given a training data set $D = \{(x_1, y_1), \dots, (x_N, y_N)\}$
- For $m=1:M$
 - Obtain a bootstrap sample D_m by drawing N instances ***with replacement*** from D
 - Learn model M_m from D_m
- To classify test instance t , apply each model M_m to t and use majority predication or average prediction
- Models have uncorrelated errors due to difference in training sets (each bootstrap sample has ~68% of D)

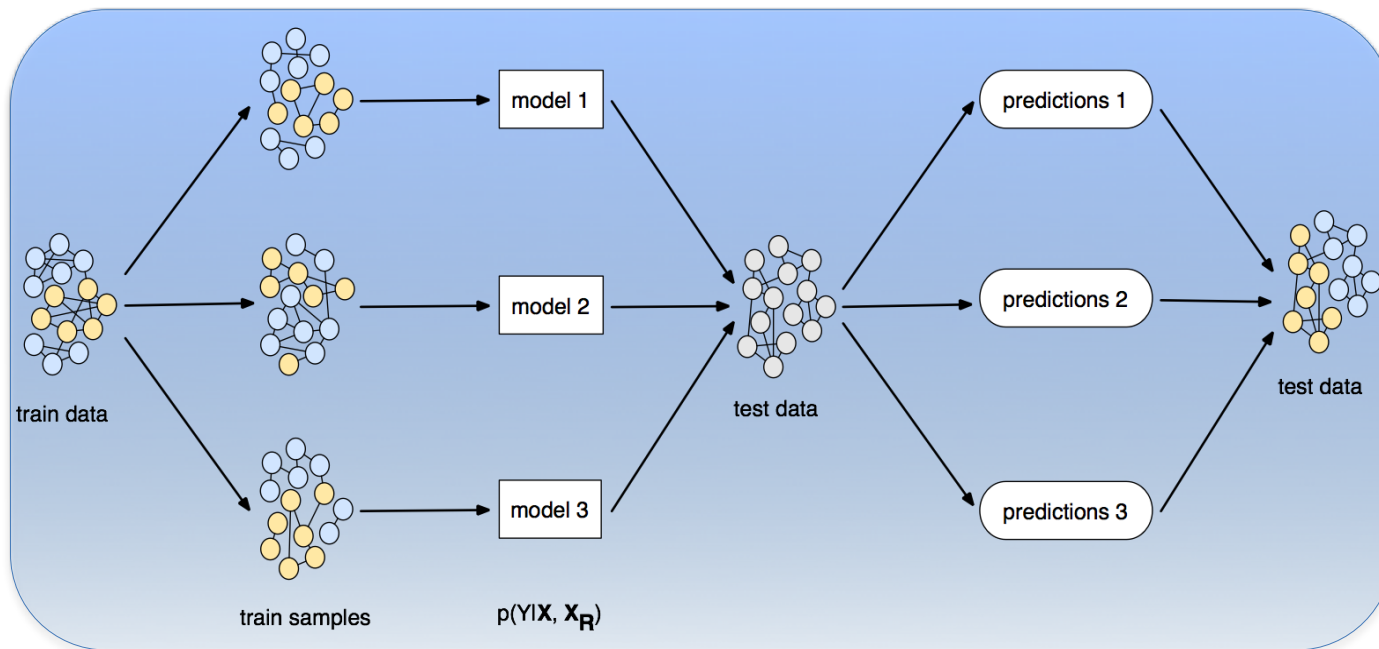


Sample to create
altered training data

Boosting

- Main assumption
 - Combining many *weak* (but stable) predictors in an ensemble produces a *strong* predictor (i.e., reduces bias)
 - Weak predictor: only weakly predicts correct class of instances (e.g., tree stumps, 1-R)
- Model space: non-parametric, can model any function if an appropriate base model is used

Boosting



TREATMENT OF INPUT DATA

- *reweight examples*

CHOICE OF BASE CLASSIFIER

- *weak predictor*
e.g., decision stump

PREDICTION AGGREGATION

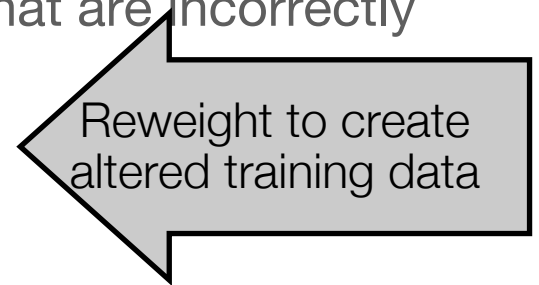
- *weighted vote*

Sequential Training: Boosting

- Classifiers trained sequentially
- Each classifier is trained given knowledge of the performance of previously trained classifiers: **focus on hard examples**
- Final classifier: weighted sum of component classifiers

General Boosting Procedure

- Assign every example in D an equal weight ($1/N$)
- For $k=1:M$
 - Learn model f_k with D_k
 - Calculate the error of M_m and up-weight the examples that are incorrectly classified to form D_{k+1}
 - Normalize weights in D_{k+1} to sum to 1
 - Set $\alpha_k = \log((1-\text{err}_k)/\text{err}_k)$
- To classify test instance t , apply each model M_k to t and take weighted vote of predictions (ie. using α_k)



Boosting Justification

- Suppose you have a *weak learner* (a classifier that uses a very simple model) that can always get correct labels with $(0.5 + \epsilon)$ probability when given a binary classification task (two labels)
- This seems like a weak assumption but...
- Can we apply this learning module many times to get a strong learner that gets close to zero error rate on the training data?
- (Freund & Shapire, 1996) theoretically showed how to do this and it actually led to an effective new learning procedure

Boosting (Adaboost) Algorithm

- First train the base classifier on all the training data with equal importance weights
- Then re-weight the training data to emphasize the misclassified (or hard) cases and train a second model
 - How to re-weight the data?
- Keep training new models on re-weighted data
- Finally, use a weighted committee of all the models for the validation data
 - How to weight the models in the committee?

Adaboost (Details)

- Input: $\{x_i, t_i\}_{i=1,\dots,n}$, $t_i \in \{-1, 1\}$ and a learning procedure f that outputs classification probabilities, Output: classifier $f^*(x)$
- Initialize example x_i weight: $w_i^{(1)} = 1/N$, $\forall i$
- For $k=1, \dots, K$

- Learn classifier f_k by minimizing over

$$\text{Err}_k = \sum_{i=1}^n w_i^{(k)} \mathbf{1}\{f_k(x_i) \neq t_i\}$$

- Compute classifier coefficient

$$\alpha_k = \frac{1}{2} \log \frac{1 - \text{Err}_k}{\text{Err}_k}$$

- Update observation weights

$$w_i^{(k+1)} = \frac{w_i^{(k)} \exp(-\alpha_k t_i f_k(x_i))}{\sum_{j=1}^n w_j^{(k)} \exp(-\alpha_k t_j f_k(x_j))}$$

- Final classifier:

$$f^*(x) = \text{sign} \left(\sum_{k=1}^K \alpha_k f_k(x) \right)$$

Adaboost (Important to Know)

- Main Issue with Adaboost?
 - Not at all robust to mislabeling (noise in labels)
 - Weights of mislabeled observations keeps growing exponentially until classifier fits the noise
- Choice of classifiers f_k ?
 - Weak learners, very simple models (e.g., shallow decision tree)
 - Otherwise Adaboost easily “overfits” data

Properties of parameters

Properties of estimators

- Let $\hat{\theta}$ be an estimate for a population parameter θ
- Using different samples D will result in different estimates $\hat{\theta}_D$
- Thus $\hat{\theta}$ is a random variable with a distribution, mean, and variance
 - We can evaluate the quality of an estimator for θ based on the properties of the sampling distribution of $\hat{\theta}$

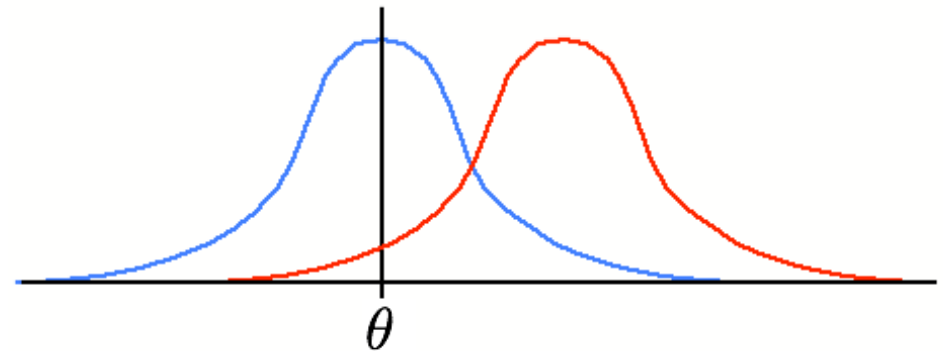
Bias

- The best estimators produce values that center around the population parameter
- The bias of an estimator is defined as:

$$Bias(\hat{\theta}) = \underbrace{E[\hat{\theta}]}_{\text{Average estimated accuracy}} - \underbrace{\theta}_{\text{True accuracy in popul.}}$$

- An estimator is unbiased if:

$$E[\hat{\theta}] - \theta = 0$$



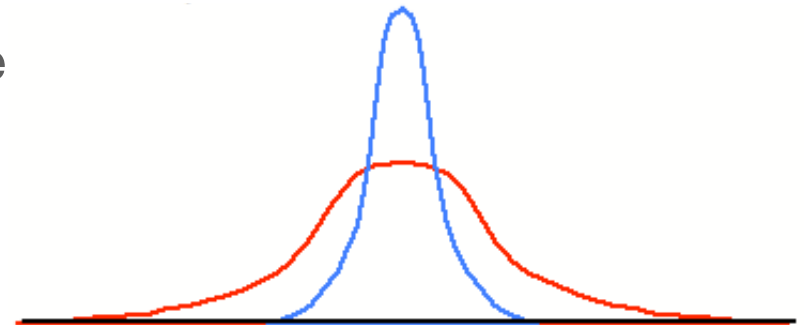
Variance

- The best estimators produce values that differ only slightly from the population parameter
- The variance of an estimator is defined as:

$$Var(\hat{\theta}) = E[(\hat{\theta} - E[\hat{\theta}])^2]$$

Accuracy on a single trial Average estimated accuracy

- Measures how sensitive the estimator is to different datasets
- Unbiased estimators with minimum variance called *best unbiased estimators*



Example

- Ignore data and declare that: $\hat{\theta} = 1.0$
- Estimate will not depend on data, thus: $Var(\hat{\theta}) = 0$
- However, in most cases this estimator will have a large bias (non-zero)

Bias-variance decomposition

- The mean-squared error (MSE) of $\hat{\theta}$ is:

$$\begin{aligned} E[(\hat{\theta} - \theta)^2] &= E[(\hat{\theta} - E[\hat{\theta}] + E[\hat{\theta}] - \theta)^2] \\ &= \underbrace{(E[\hat{\theta}] - \theta)^2}_{\text{bias}} + \underbrace{E[(\hat{\theta} - E[\hat{\theta}])^2]}_{\text{variance}} \end{aligned}$$

- MSE measures systematic bias and random variance between estimate and population value

Comparing algorithms (cont)

Bias-variance analysis of model prediction error

$$E_D[L_{sq}(t, y)] = \underbrace{E_D[(t - E_D[t])^2]}_{\text{noise}} + \underbrace{(E_D[t] - E_D[y])^2}_{\text{bias}} + \underbrace{E_D[(E_D[y] - y)^2]}_{\text{variance}}$$

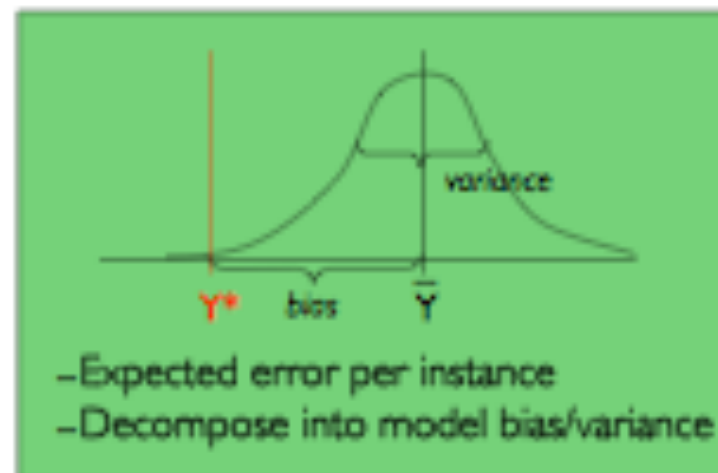
- **Noise:** loss incurred for optimal prediction $E_D[t]$ (*independent of learning algorithm*)

$t :=$ true class

$y :=$ predicted class

$E_D :=$ expectation over the data

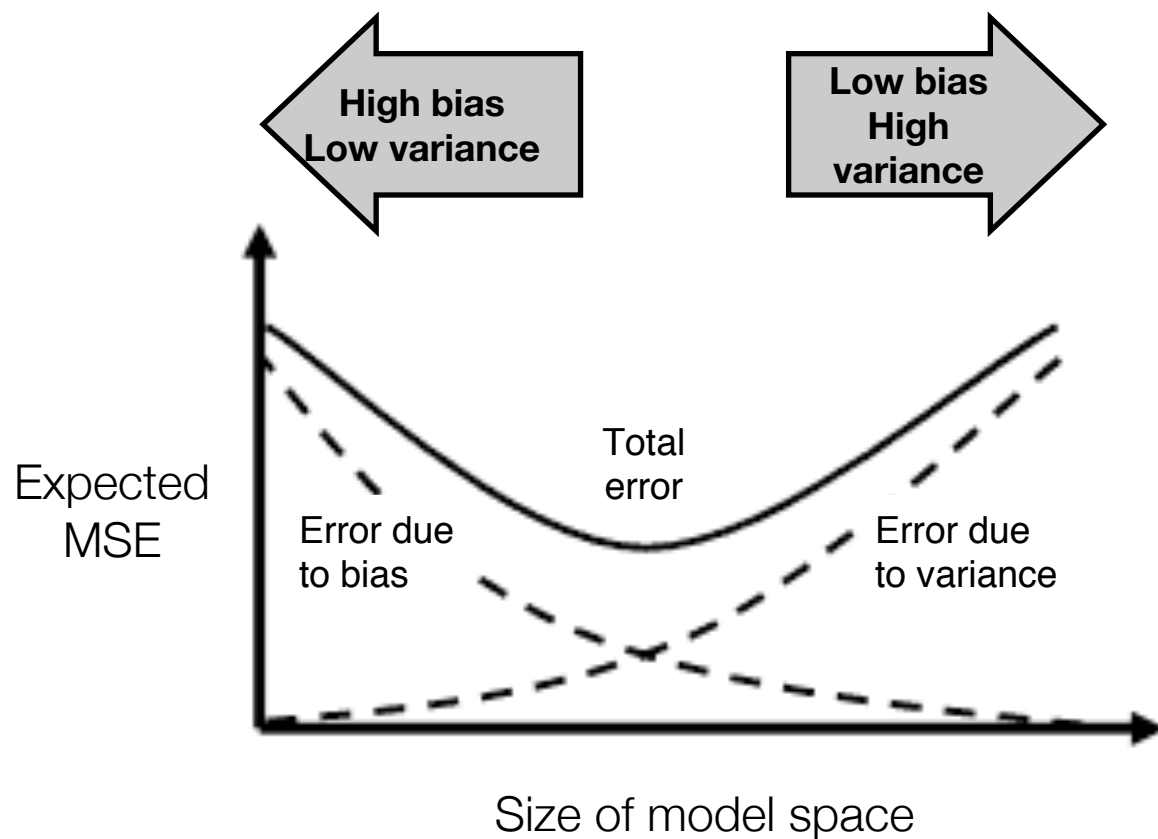
- **Bias:** loss incurred of mean prediction $E_D[y]$ relative to optimal prediction $E_D[t]$
- **Variance:** average loss of predictions compared to mean prediction $E_D[y]$



Findings

- Bias
 - Often related to size of model space
 - High bias indicates a poor match between model and concept
 - More complex models tend to have larger model space and thus lower bias
- Variance
 - Often related to size of dataset; When data is large enough to estimate parameters well then models have lower variance
 - High variance indicates a weak match between model and concept
 - More complex models usually have more parameters and thus tend to have higher variance
- Simple models can perform surprisingly well due to lower variance

Bias/variance tradeoff for learning a single model

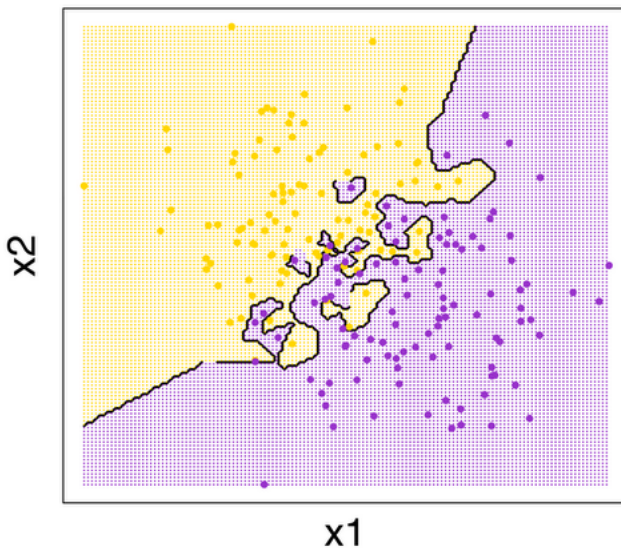


Bias-variance tradeoff:
increasing the size of the model space can **reduce bias** of the learned model, but that also tends to **increase variance**...

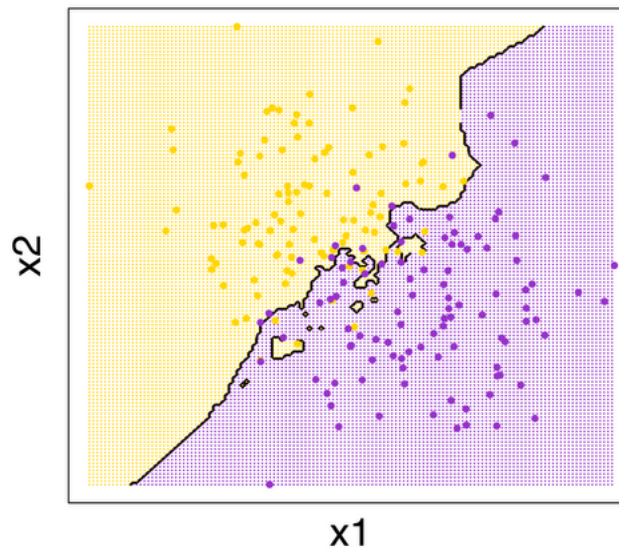
and *decreasing* the model space tends to **reduce variance** but also **increase bias**

kNN model complexity decreases as k increases

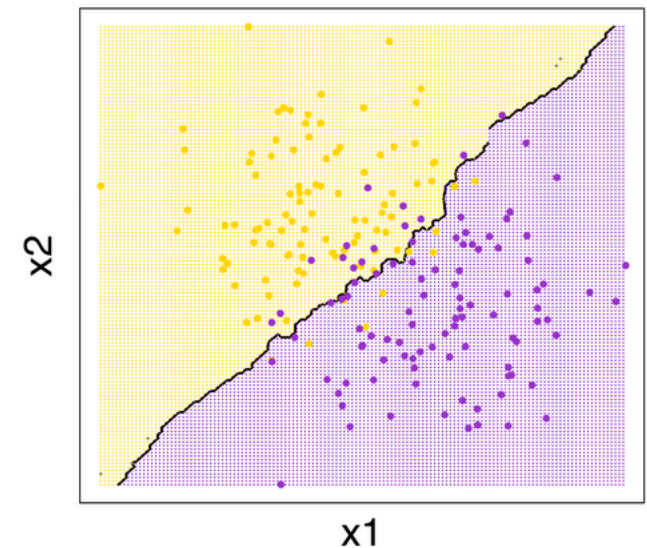
Binary kNN Classification (k=1)



Binary kNN Classification (k=5)



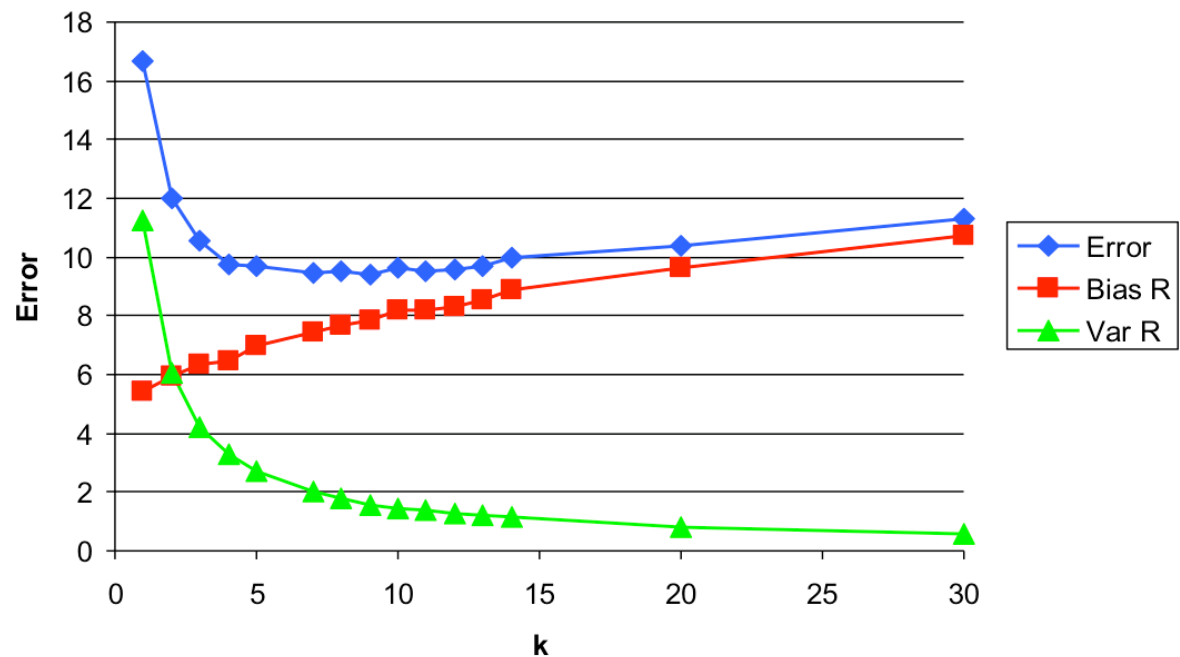
Binary kNN Classification (k=25)



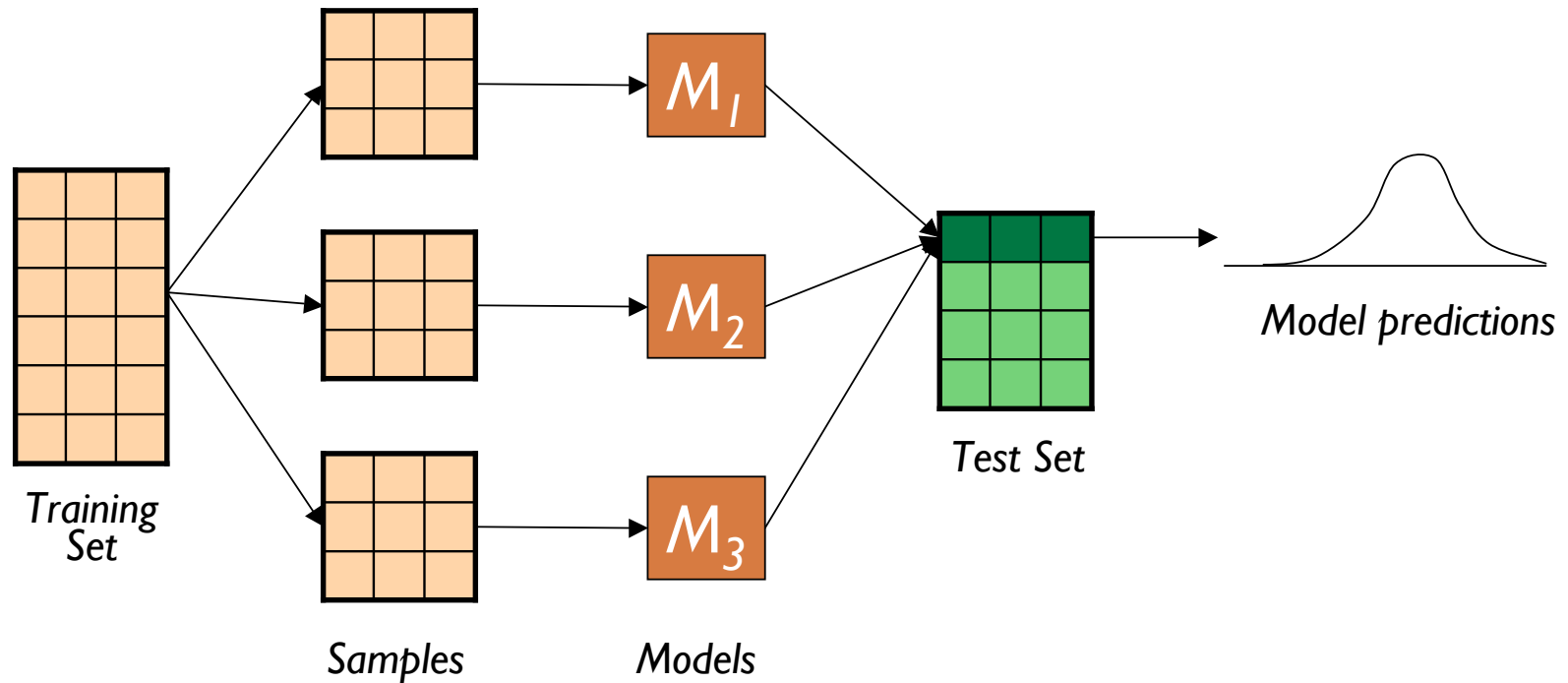
K decreases
Model complexity/space increases
Bias decreases, variance increases

K increases
Model complexity/space decreases
Bias increases, variance decreases

Bias variance as k increases in kNN



Bias-variance analysis to compare models:
to assess types of error and their impact on performance



Ensemble summary

- Bagging doesn't work so well on stable models; boosting may still help
- Boosting might hurt performance on noisy datasets; bagging doesn't have this problem
- Boosting typically helps more than bagging, but is also more common for boosting to hurt performance

More score functions

- Contingency table loss functions

- True positive (**TP**):
positive prediction that is correct
- True negative (**TN**):
negative prediction that is correct
- False positive (**FP**):
positive prediction that is incorrect
- False negative (**FN**):
negative prediction that is incorrect

		Actual	
		+	-
Predicted	+	TP	FP
	-	FN	TN

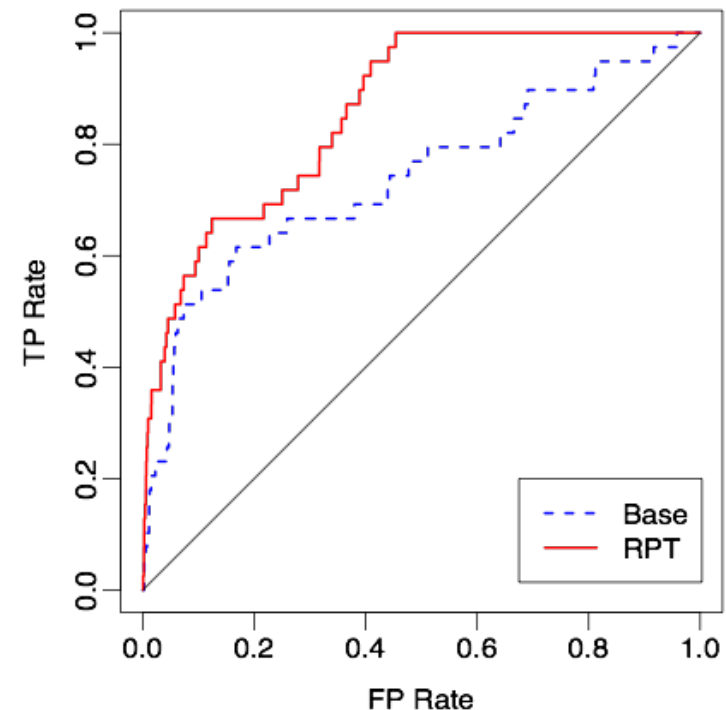
AKA: confusion matrix

- Accuracy = $(TP + TN) / (TP + TN + FP + FN)$ *% predictions that are correct*
- Recall/Sensitivity = $TP / (TP + FN)$ *% positive instances that are predicted positive*
- Precision = $TP / (TP + FP)$ *% positive predictions that are correct*
- Specificity = $TN / (TN + FP)$ *% negative instances that are predicted negative*
- F1 = $2 (P \cdot R) / (P + R)$ *harmonic mean of precision and recall*

More score functions

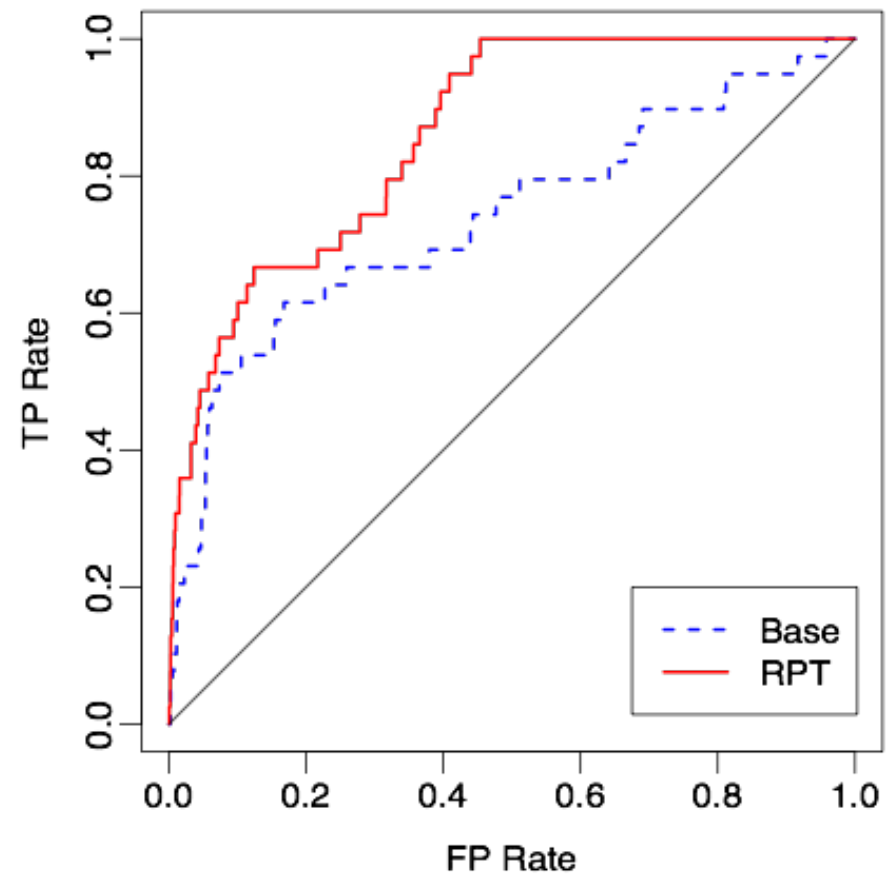
- Absolute loss
- Squared loss
 - Root mean-squared error
- Likelihood/conditional likelihood
- **Area under the ROC curve**
 - *Receiver Operating Characteristic* (ROC) curve
 - Plots the true positive rate against the false positive rate for different classification thresholds

$$S(M) = \sum_{i=1}^{N_{test}} d[f(x(i); M), y(i)]$$



ROC curves

- Evaluates performance over varying costs and class distributions
 - Can summarize with area under the curve (AUC)
 - AUC of 0.5 is random
 - AUC of 1.0 is perfect



How to compute ROC curve

$P(Y)$	True class
0.94	+
0.84	-
0.67	+
0.58	+
0.51	+
0.42	+
0.16	-
0.1	+
0.07	-

$P(Y)$	True class	Predict class
0.94	+	+
0.84	-	-
0.67	+	-
0.58	-	-
0.51	+	-
0.42	+	-
0.16	-	-
0.1	-	-
0.07	-	-

TPR = 1/4
FPR = 0/5

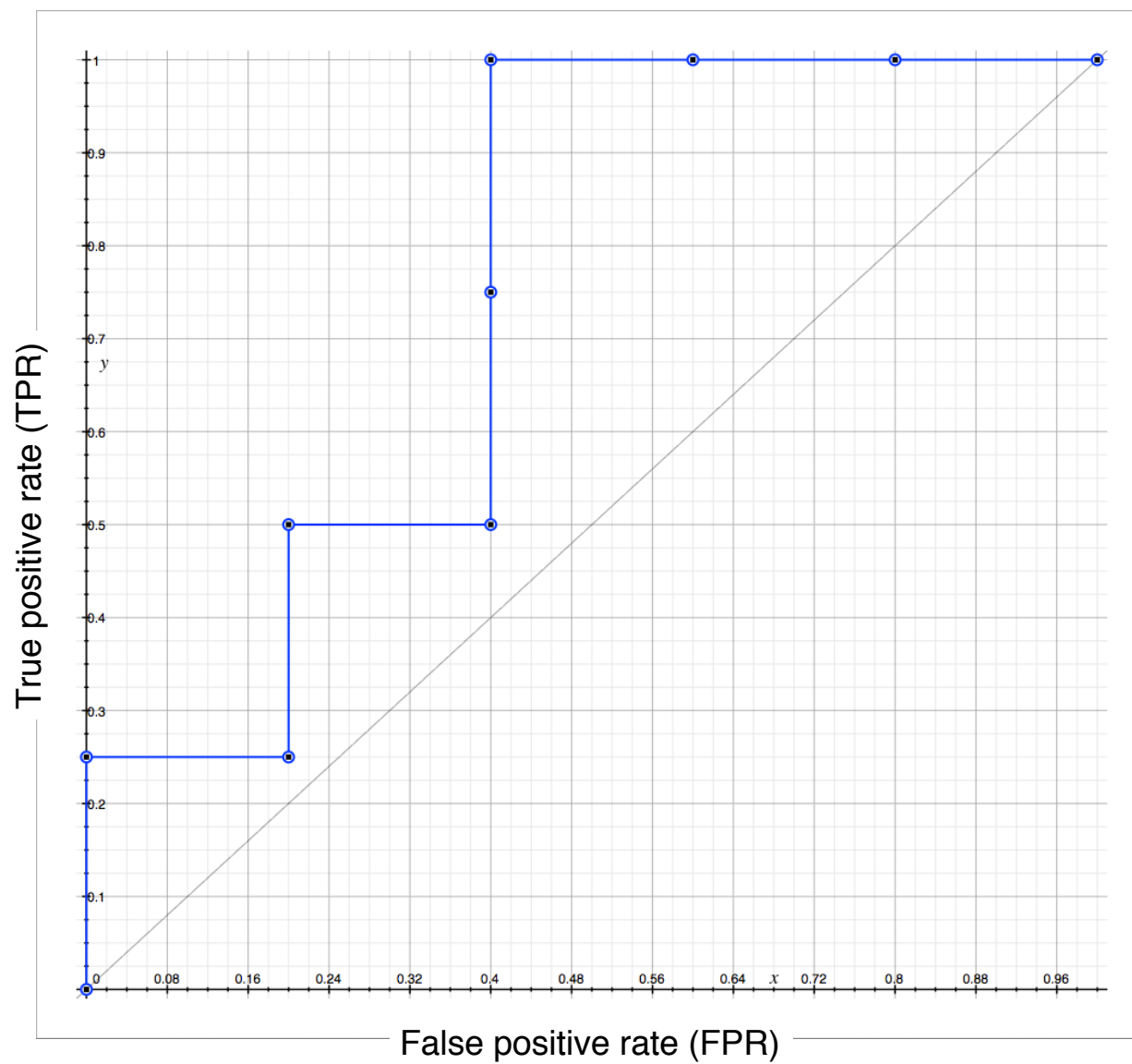
$P(Y)$	True class	Predict class
0.94	+	+
0.84	-	+
0.67	+	-
0.58	-	-
0.51	+	-
0.42	+	-
0.16	-	-
0.1	-	-
0.07	-	-

TPR = 1/4
FPR = 1/5

$P(Y)$	True class	Predict class
0.94	+	+
0.84	-	+
0.67	+	+
0.58	-	-
0.51	+	-
0.42	+	-
0.16	-	-
0.1	-	-
0.07	-	-

TPR = 2/4
FPR = 1/5

ROC curve



Issues with conventional score functions

- Assumes errors for all individuals are equally weighted
 - May want to weight recent instances more heavily
 - May want to include information about reliability of sets of measurements
- Assumes errors for all contexts are equally weighted
 - May want to weigh false positive and false negatives differently
 - May be costs associated with acting on particular classifications (e.g., marketing to individuals)

Examples

- Loan decisions
 - Cost of lending to a defaulter is much greater than the lost business of refusing loan to a non-defaulter
- Oil-slick detection
 - Cost of failing to detect an environmental disaster is far less than the cost of a false alarm
- Promotional mailing
 - Cost of sending junk mail to a household that doesn't respond is far less than the lost business of not sending it to a household that would have responded

Cost-sensitive models

- Define score function based on cost matrix that reflects the gain/loss of classifying an instance with true class y to predicted class y'
- If $f(x)$ is the predicted class and y is the true class, then define matrix of costs $C[f(x), y]$

**Original
0/1 loss**

$$S_{0/1}(M) = \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} I[f(x(i); M), y(i)]$$

$$\text{where } I(a, b) = \begin{cases} 1 & a \neq b \\ 0 & \text{otherwise} \end{cases}$$

**Cost-
sensitive**

$$S_C(M) = \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} C[f(x(i); M), y(i)]$$

		Actual	
		+	-
Predicted	+	TP	FP
	-	FN	TN

C:

		Actual	
		+	-
Predicted	+	+\$10	-\$1
	-	-\$5	0