Prognostics & Health Management

Part 1: Data-Driven Anomaly Detection & Diagnosis

Neil H. W. Eklund
General Electric Global Research
Industrial Artificial Intelligence Laboratory

imagination at work

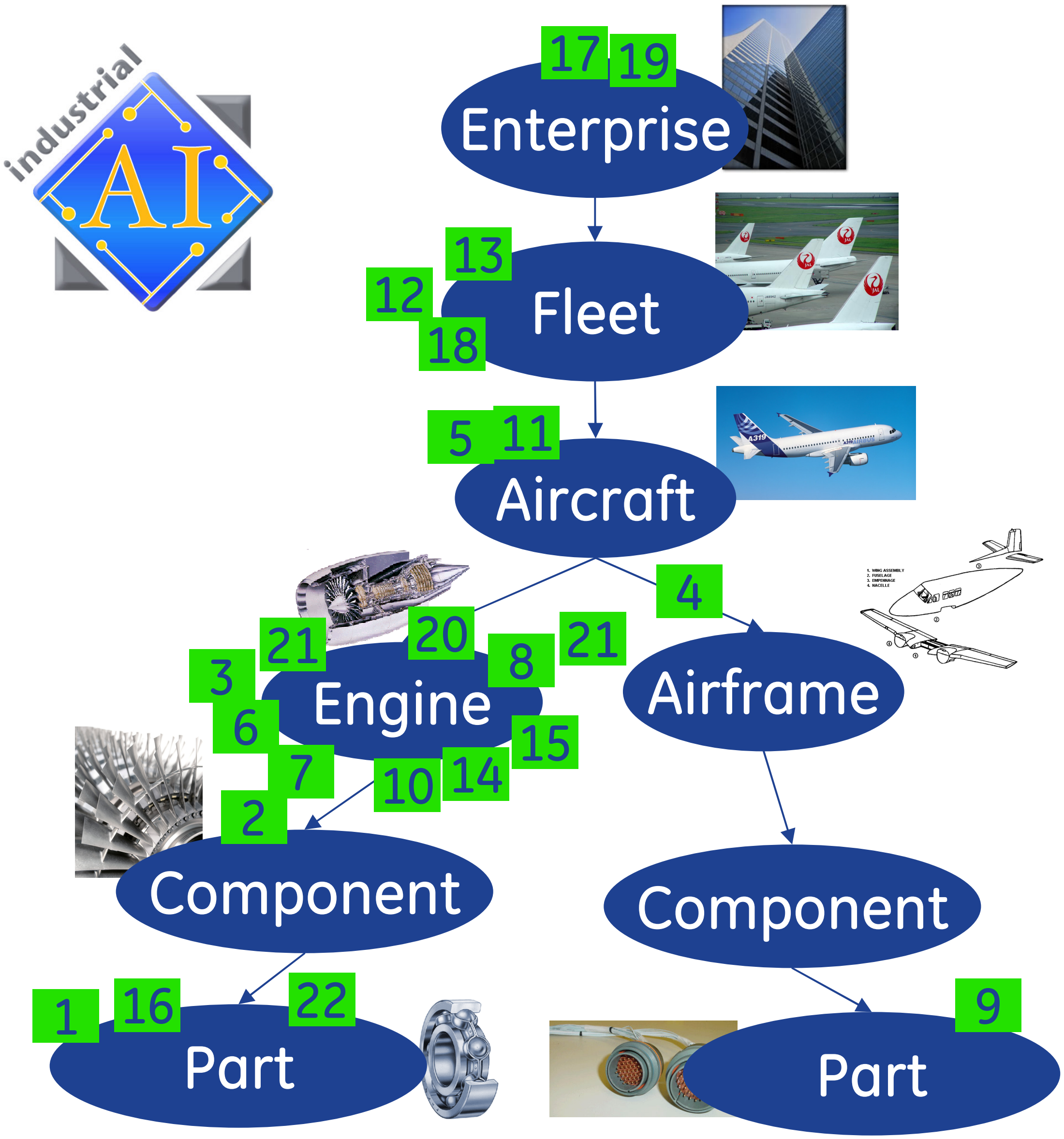
What have I left out?

Many things!

- Sensor validation
- Data conditioning
- Use of physics-based models
- Data set partitioning
  - Training
  - Test
  - Validation
- And so on…

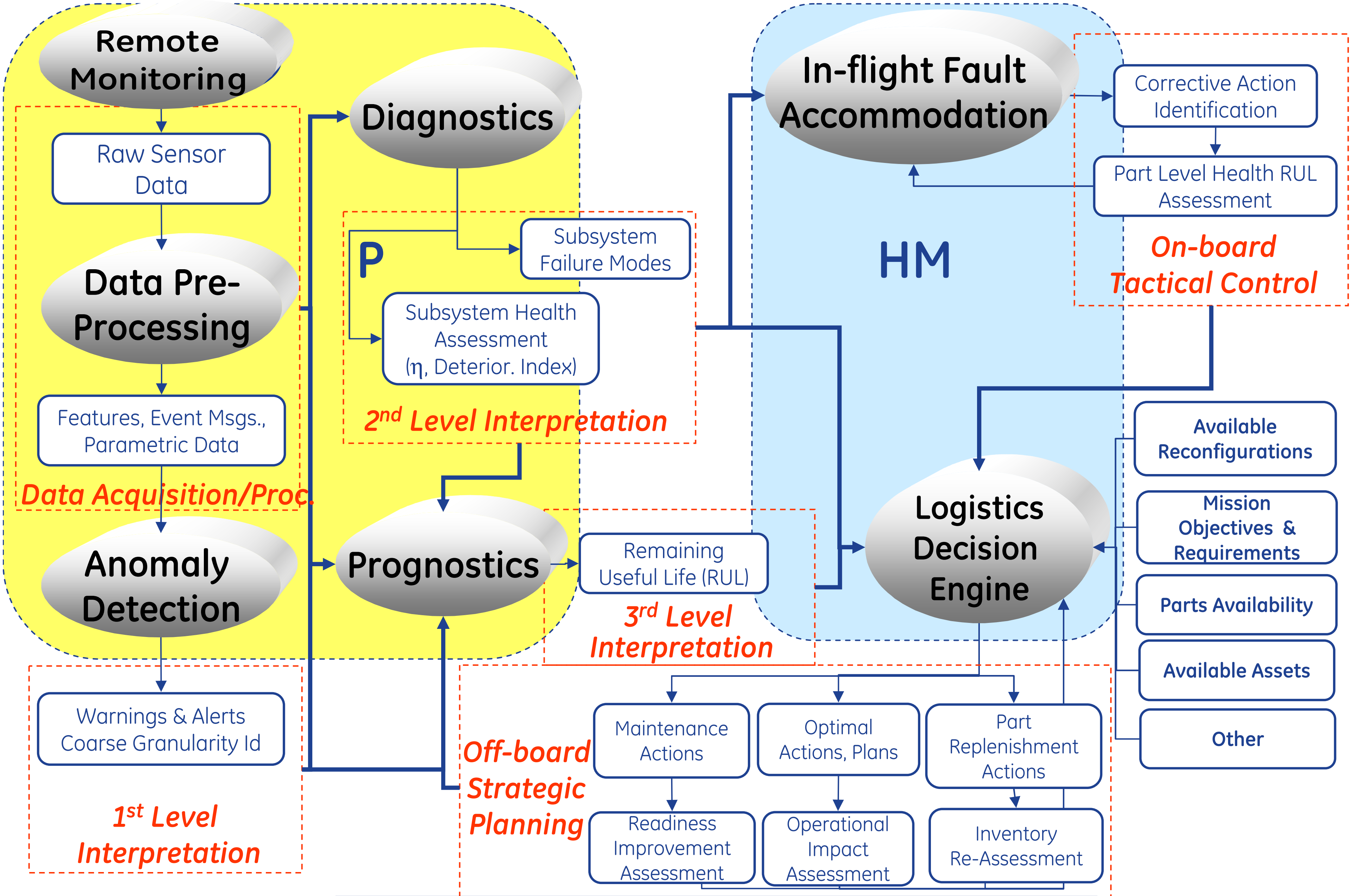
Background

iAI Lab **PHM** Projects



1. Engine ball bearing prognostics [DARPA 500k]
2. Engine Prognostics [MGPD/GEAE 765k]
3. Engine system prognostics [DARPA 300K]
4. Boeing data fusion [GE/Boeing 900k]
5. Anomaly detection for aircraft [LM 625K]
6. Baseline engine parameter estimation [GEAE 1MM]
7. Deterioration rate estimation [GEAE 300k]
8. Intelligent maintenance advisor for turbine engines [DARPA 350k]
9. Smart wire [Navair 250k]
10. Fault accommodation [NASA 950k]
11. Alert fusion [MGPD 300k]
12. Engine removal forecasting [GEAE 5MM]
13. Workscope optimization [GEAE 1.5MM]
14. Power management optimization [GEAE 1MM]
15. OPTICS [GEAE 350k]
16. Physical-based Lifing [GEAE 100k]
17. Modeling and simulation for management of PBL [LM 550k]
18. Small Commercial trend [GEAE 500k]
19. Automated workscope [GEAE 400k]
20. EGT workscope [GEAE 50k]
21. LEAP56 advanced functionality study
22. Part lifecycle management [GEAE 500k]

PHM: Functional Architecture



Remote Monitoring

Raw Sensor Data

Data Pre-Processing

Features, Event Msgs., Parametric Data

Data Acquisition/Proc.

Anomaly Detection

Warnings & Alerts Coarse Granularity Id

1st Level Interpretation

Diagnostics

P

Subsystem Failure Modes

Subsystem Health Assessment (η, Deterior. Index)

2nd Level Interpretation

Prognostics

Remaining Useful Life (RUL)

3rd Level Interpretation

HM

In-flight Fault Accommodation

Corrective Action Identification

Part Level Health RUL Assessment

On-board Tactical Control

Logistics Decision Engine

Available Reconfigurations

Mission Objectives & Requirements

Parts Availability

Available Assets

Other

Off-board Strategic Planning

Maintenance Actions

Optimal Actions, Plans

Part Replenishment Actions

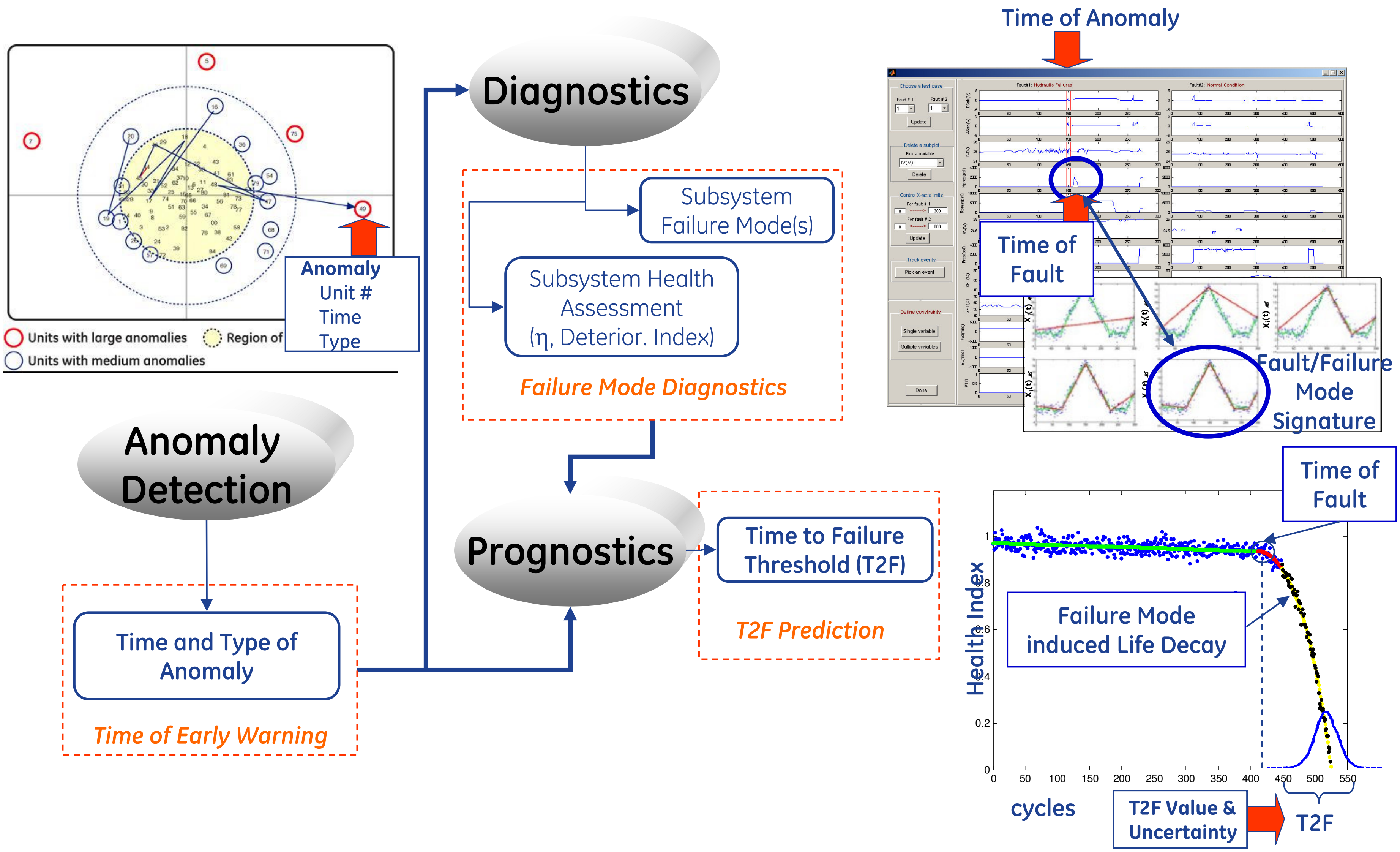
Readiness Improvement Assessment

Operational Impact Assessment

Inventory Re-Assessment

PHM Requires a Holistic Approach

Focus on Asset Health Monitoring (P):
From Anomaly Detection to Diagnostics and Prognostics



Anomaly Detection

Time and Type of Anomaly

Time of Early Warning

Diagnostics

Subsystem Failure Mode(s)

Subsystem Health Assessment (η, Deterior. Index)

Failure Mode Diagnostics

Prognostics

Time to Failure Threshold (T2F)

T2F Prediction

Anomaly
Unit #
Time
Type

Time of Anomaly

Time of Fault

Fault/Failure Mode Signature

Time of Fault

Failure Mode induced Life Decay

T2F Value & Uncertainty

T2F

Anomaly Detection

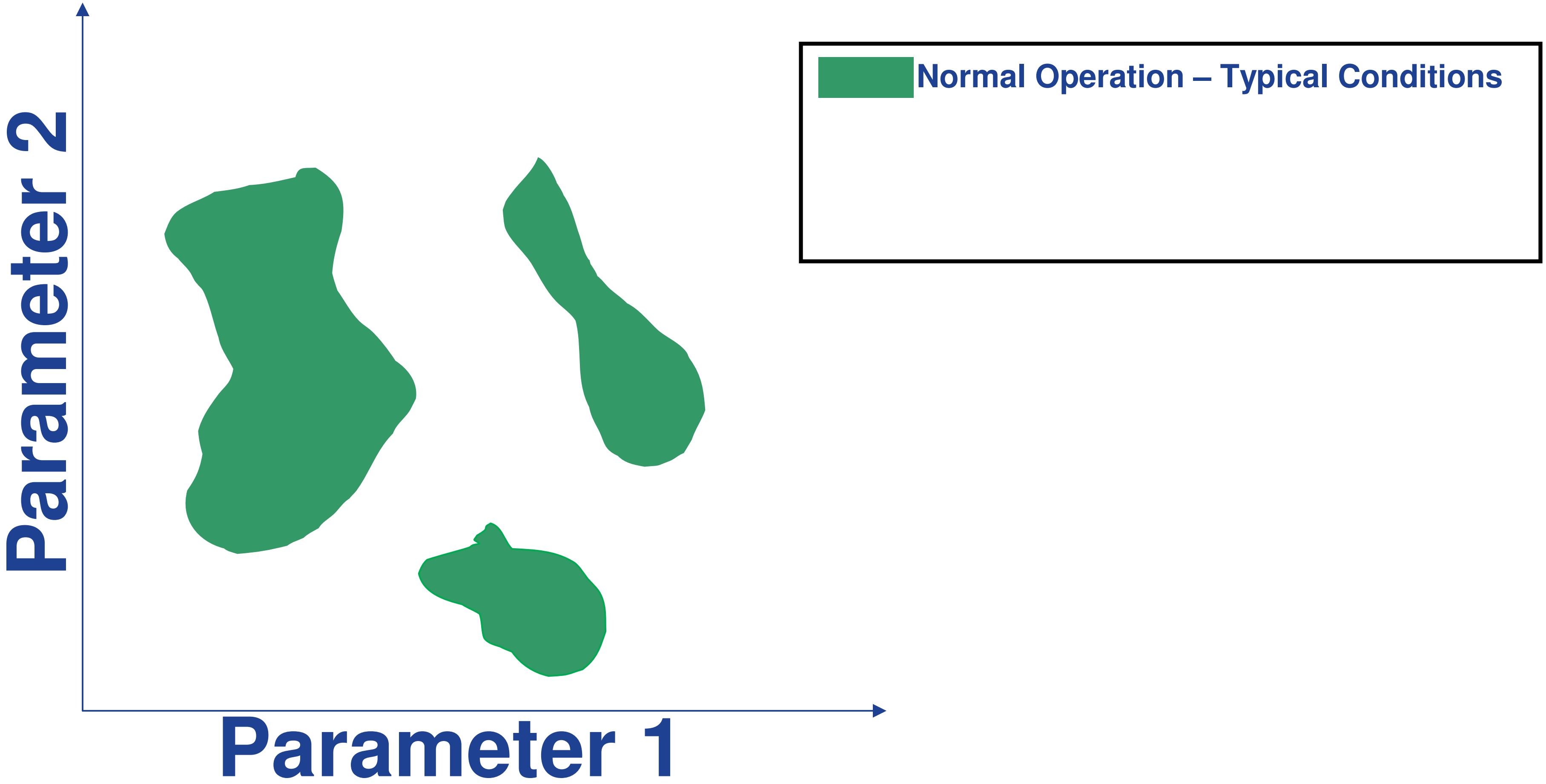
Nature of Anomaly Detection

Anomaly detection asks the question,

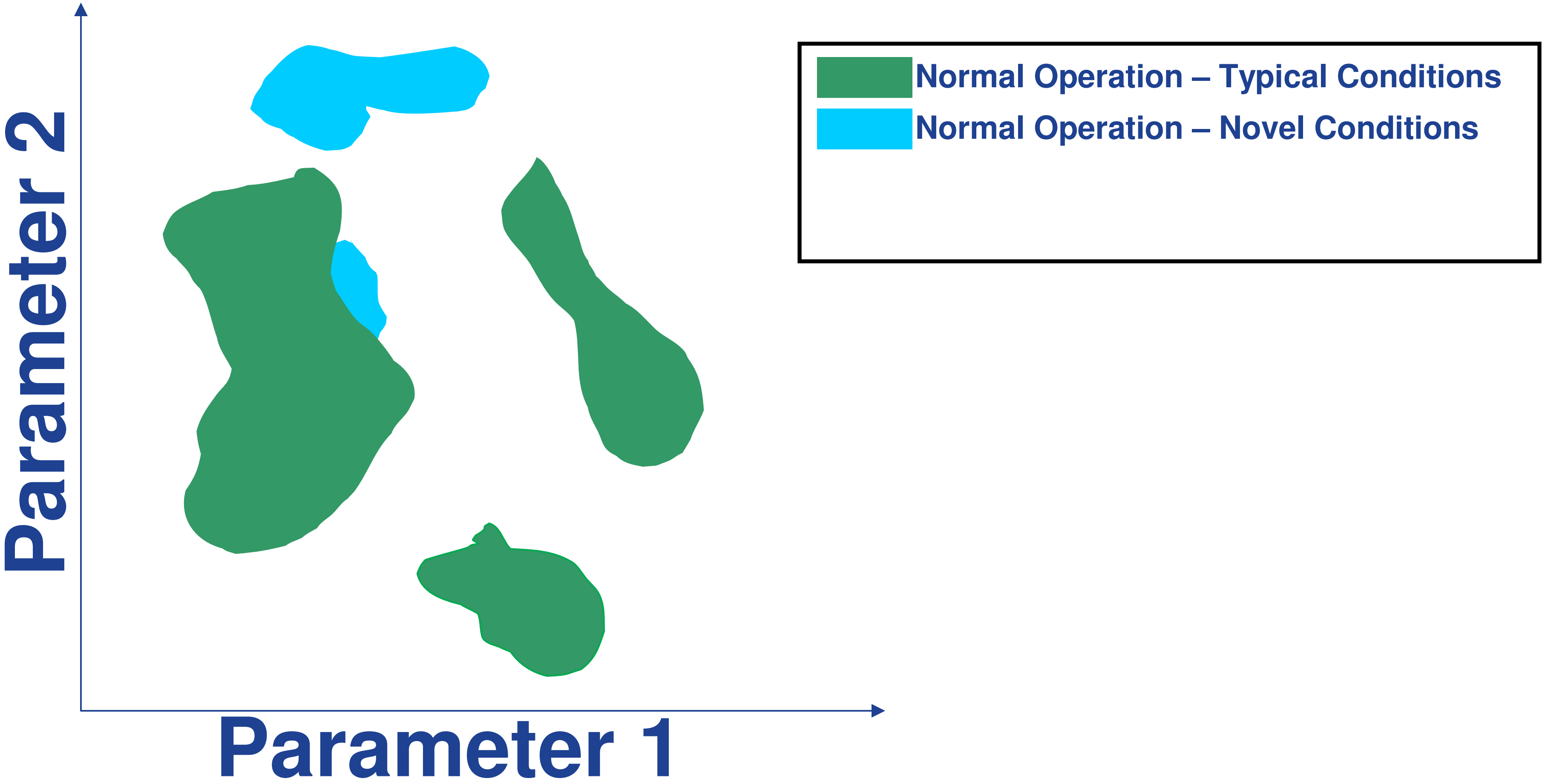
“Is my {complex | mission critical | safety-critical | expensive | highly loaded | …} system operating normally?”

It seems like such a simple question! And yet…

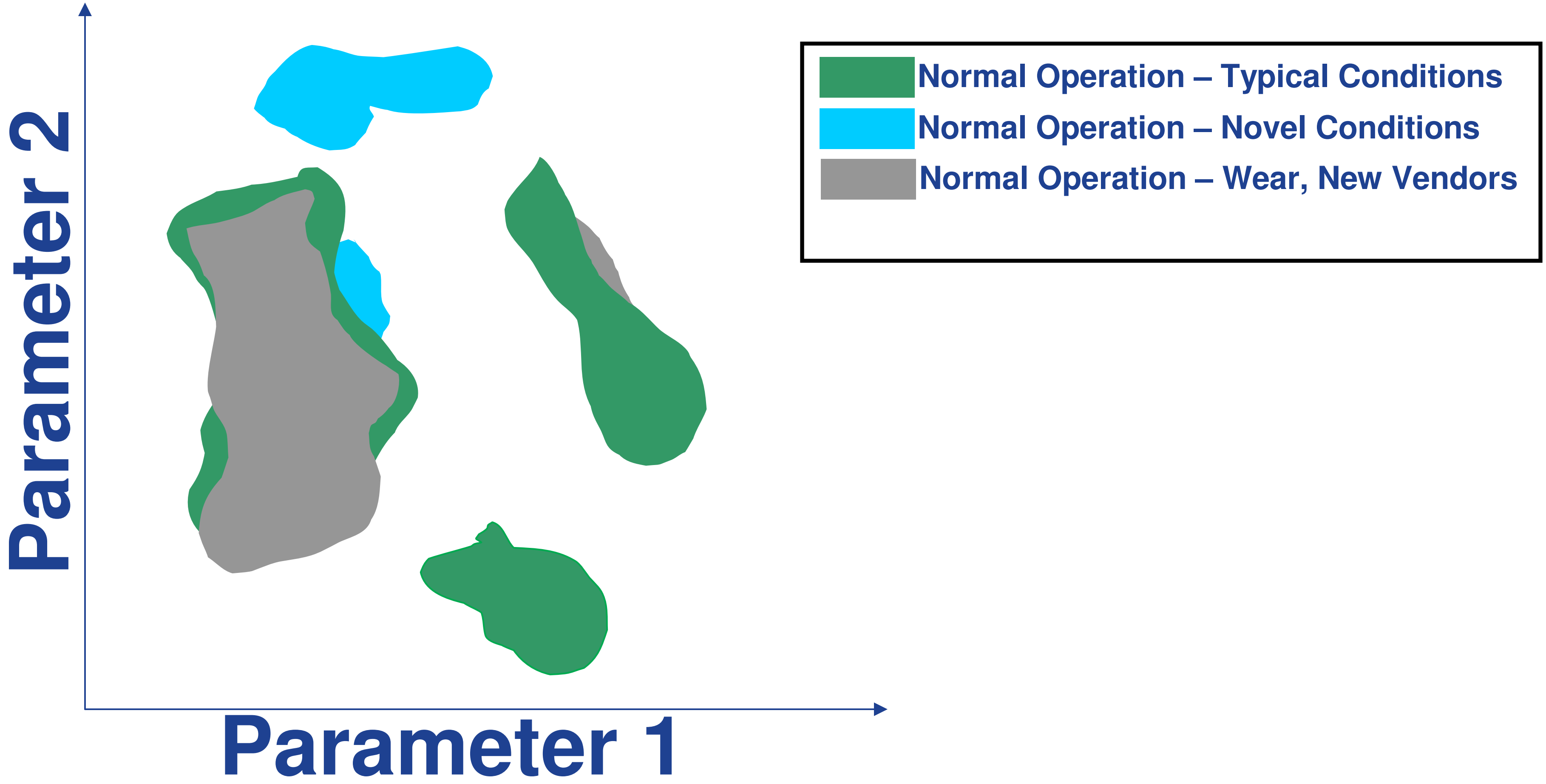
Normal Operation, Normal Conditions



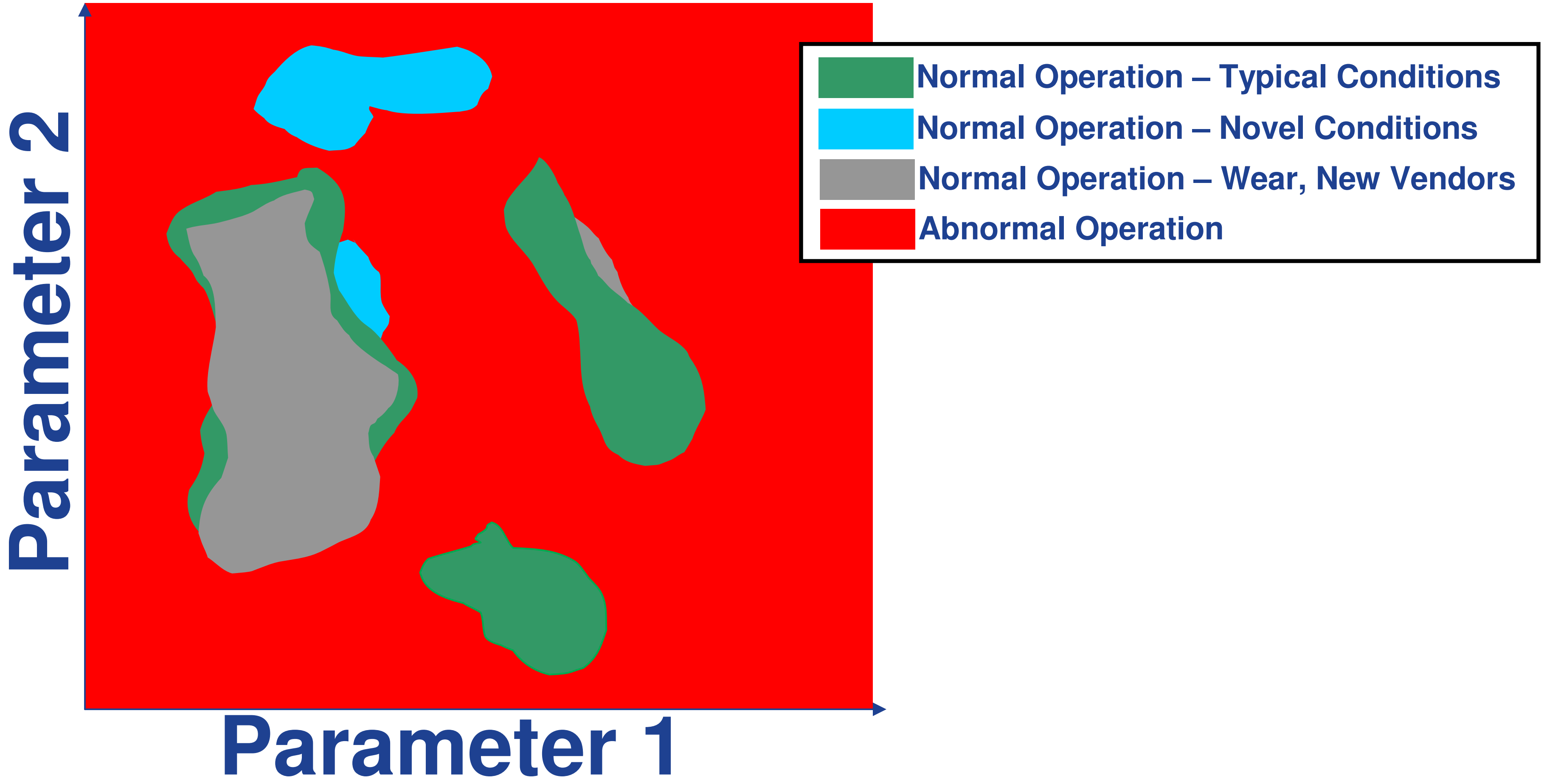
Novel Conditions


Normal Operation – Typical Conditions
Normal Operation – Novel Conditions
Parameter 2
Parameter 1

Changes in Fleet: Mixture of Operating Modes


Parameter 2
Parameter 1
🟩 Normal Operation – Typical Conditions
🟦 Normal Operation – Novel Conditions
⬜ Normal Operation – Wear, New Vendors

Anomaly Detection


Parameter 2
Parameter 1
Normal Operation – Typical Conditions
Normal Operation – Novel Conditions
Normal Operation – Wear, New Vendors
Abnormal Operation

Wacky Tricks for Anomaly Detection

Data may be

- univariate or multivariate
- parametric, nonparametric, or mixed

AD approaches differ for each combination…

Wacky Tricks for AD #1:

Rank Permutation Test

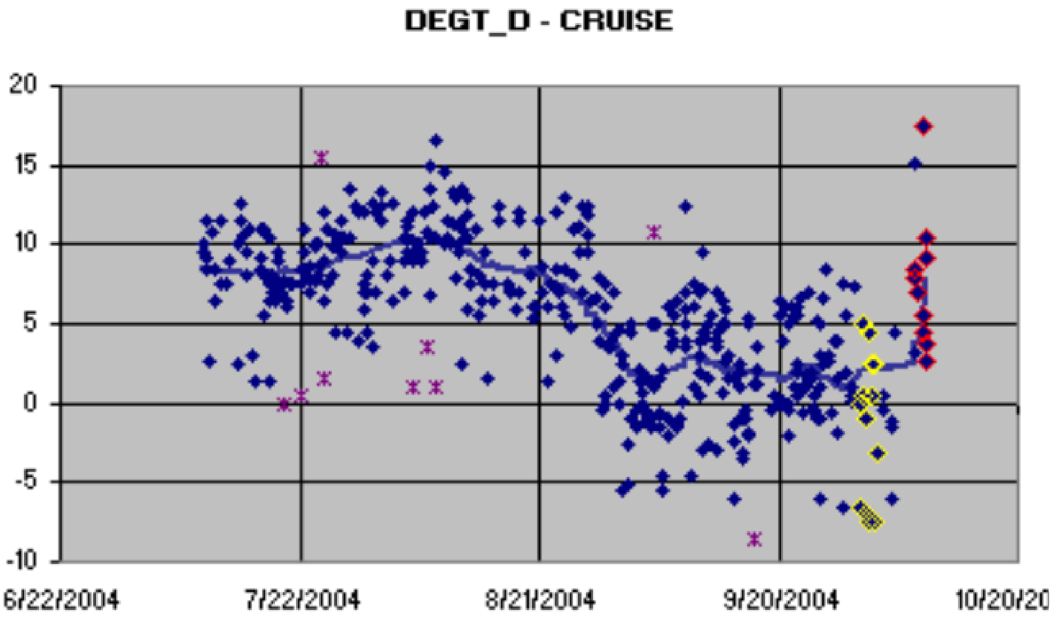
[univariate, parametric]

Univariate Parametric Data

Surprisingly common problem, e.g., aircraft EGT monitoring

Two major problems:

1) rapid detection is critical
  - safety
  - localize start of problem



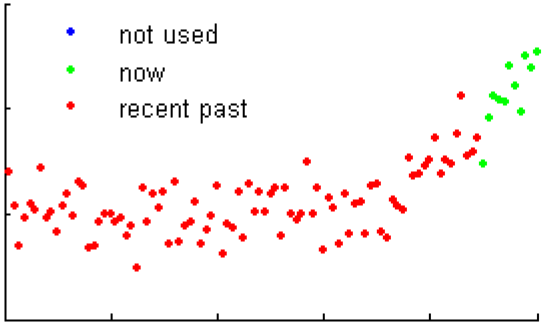
2) data is “real world”
  - ugly
    - outliers
    - non-normal distribution
    - not independent observations
    - noisy
      - engine to engine variation not accounted for
      - some flight envelope effects not accounted for

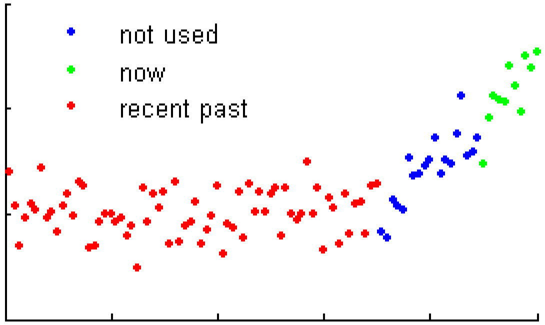
How to detect change in a way that is robust to noise & outliers?

general approach

compare "now" to the "recent past"

- now: 2, 5, 10… some small number of most recent time steps
- recent past: as many points in the past as feasible
  - more data is better (although often not much available)
  - maybe offset some points to make slow ramp more pronounced



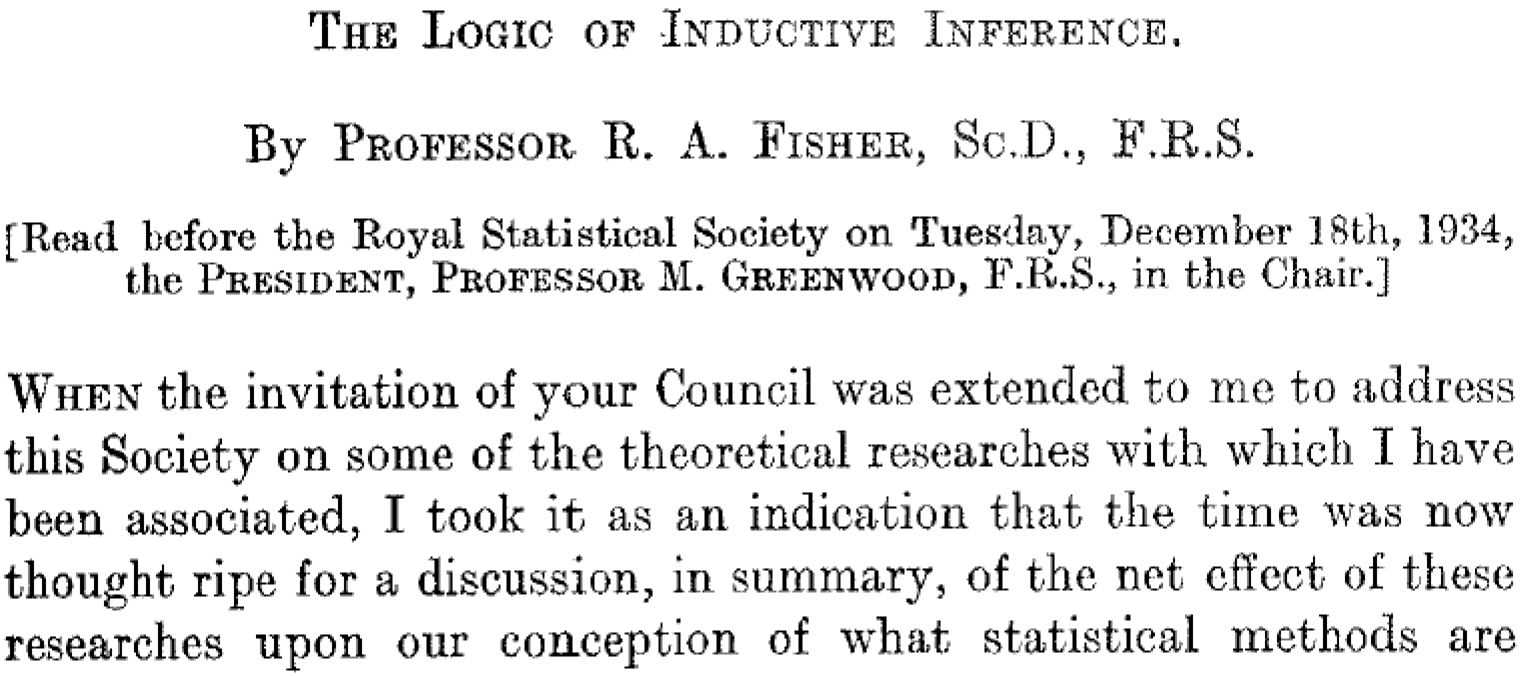


How to compare?

three methods:

- t-test
  - data violate almost all assumptions… but statisticians will often wave their hands and say, “Well, given enough data, it is OK.”
  - good standard for comparison
- Wilcoxon rank sum == Mann-Whitney U test
  - nonparametric equivalent to t-test
  - based on rank distribution
  - 99.5% asymptotic relative efficiency for normal data
    - *much better than t-test for non-normal*
  - robust to outliers
- rank permutation test

permutation test: Kickin' it old school!



THE LOGIC OF INDUCTIVE INFERENCE.

By PROFESSOR R. A. FISHER, Sc.D., F.R.S.

[Read before the Royal Statistical Society on Tuesday, December 18th, 1934, the PRESIDENT, PROFESSOR M. GREENWOOD, F.R.S., in the Chair.]

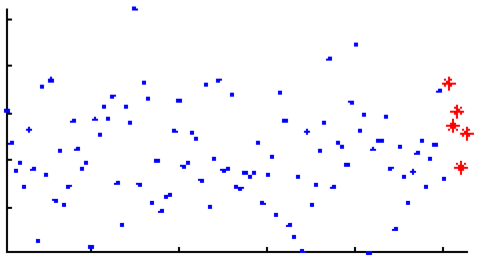
WHEN the invitation of your Council was extended to me to address this Society on some of the theoretical researches with which I have been associated, I took it as an indication that the time was now thought ripe for a discussion, in summary, of the net effect of these researches upon our conception of what statistical methods are

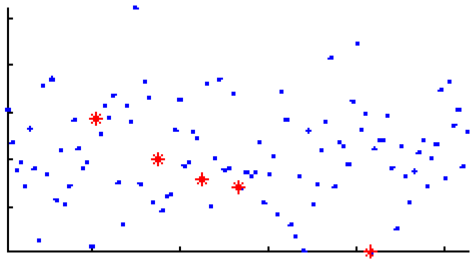
Fisher, R. A. (1935). The Logic of Inductive Inference.
Journal of the Royal Statistical Society, 98: 39-54

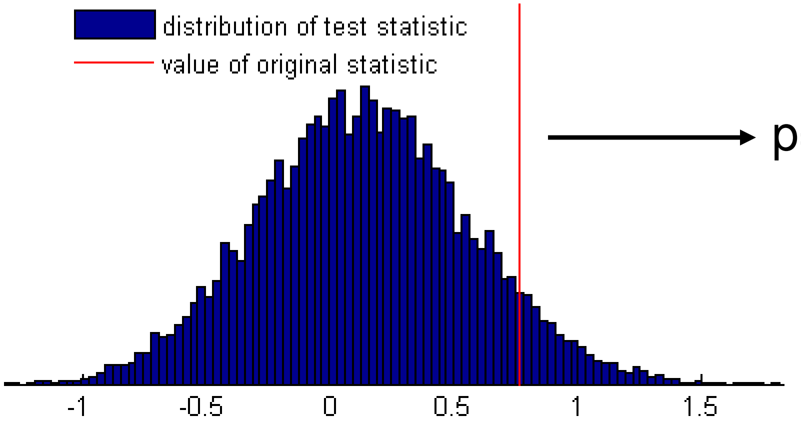
permutation test

five steps:

1. analyze the problem
   - determine a testable null hypothesis
2. choose a test statistic
   - e.g., sample mean
3. compute the test statistic for the original observations
4. permute the observations, and recalculate the test statistic; repeat
5. accept or reject null hypothesis using permutation distribution







p==0.072

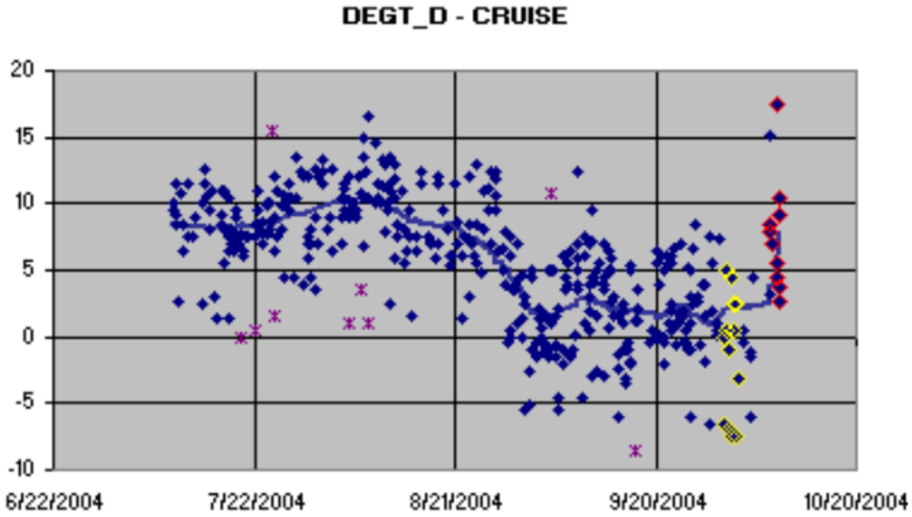
rank permutation test

Why ranks?

- to diminish the effects of outliers
- to make distribution-free
  - permits precalculation of permutation distribution
    - low memory, ***fast*** – important for on-wing applications

What is the cost of using ranks?

- slight loss of power
- **BUT**:
  - loss is very slight
  - well worth gain in robustness to ugly data



experimental design

two distributions

- normal
- fat tail, with outliers (49.5% σ=1; 49.5% σ =3; 1% σ =10)/3

three levels of difference

- [0.5 1.5 3]

two levels of “recent past”

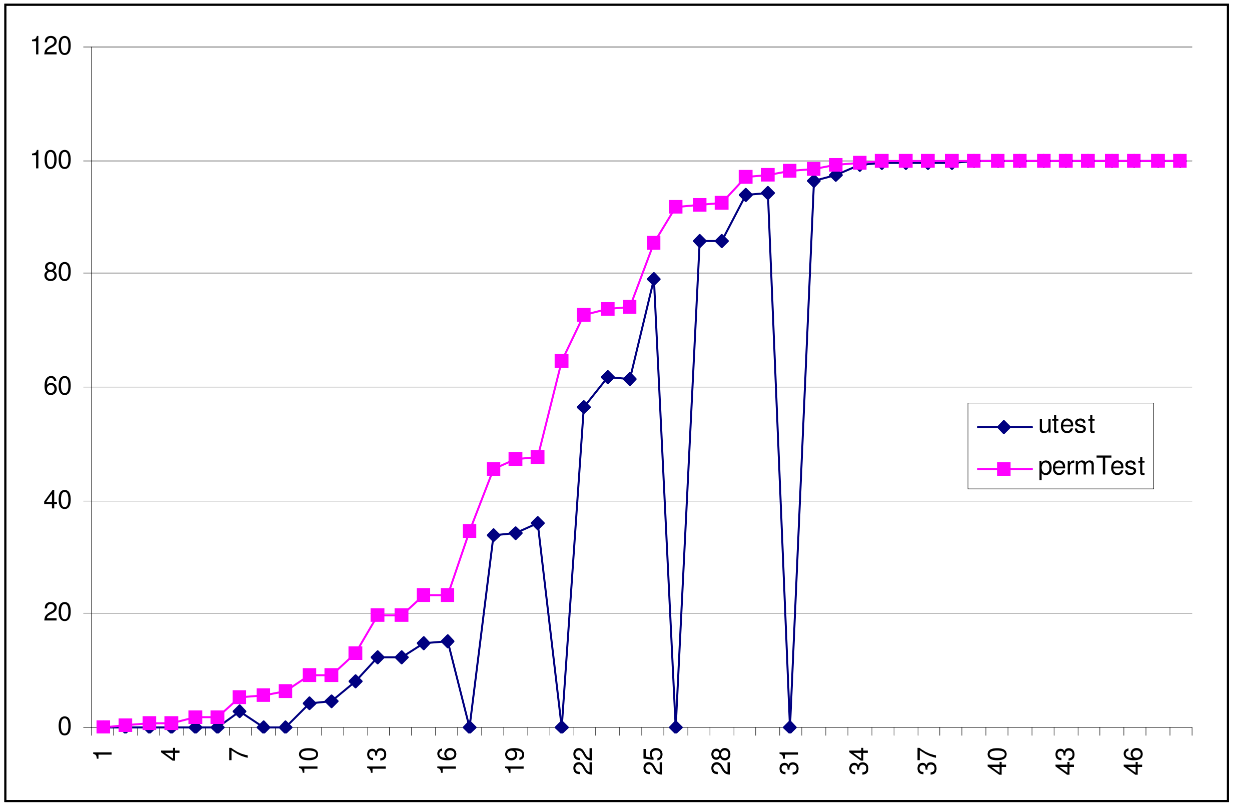
- [10 100]

four levels of “now”

- [2 5 10 100]

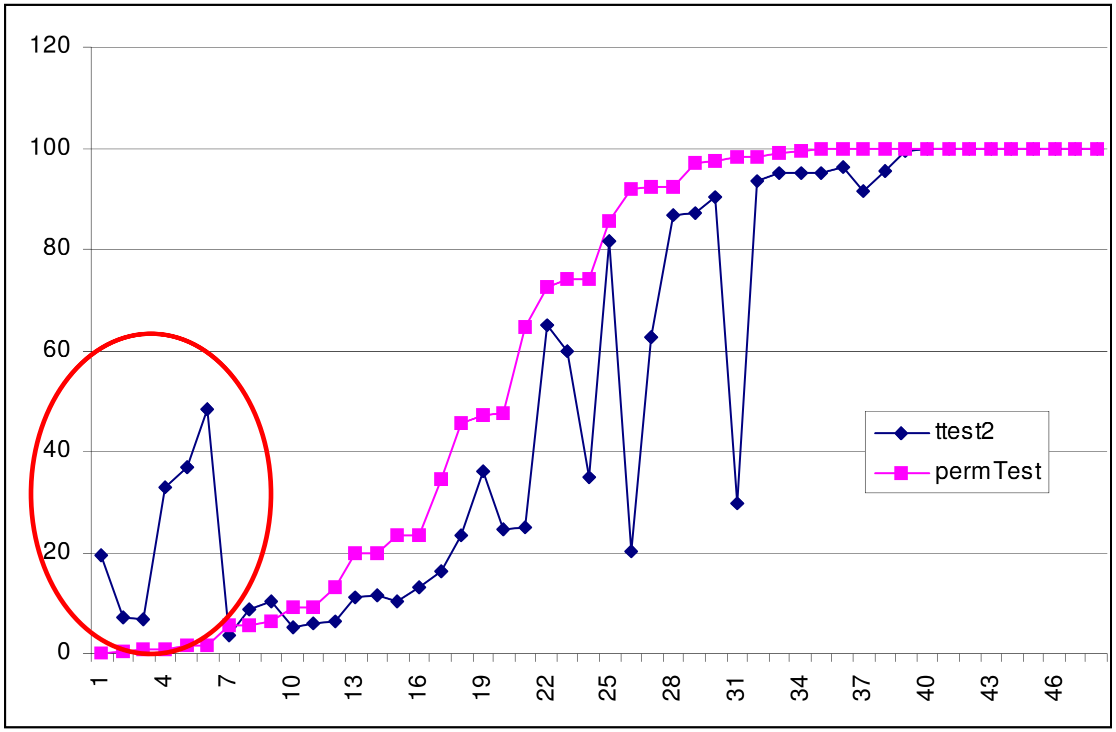
result 1: U <= permutation

statistic is percent correctly detected at 1% significance level



result 2: T (mostly) <= permutation

statistic is percent correctly detected at 1% significance level



result 2: T (sometimes) > permutation

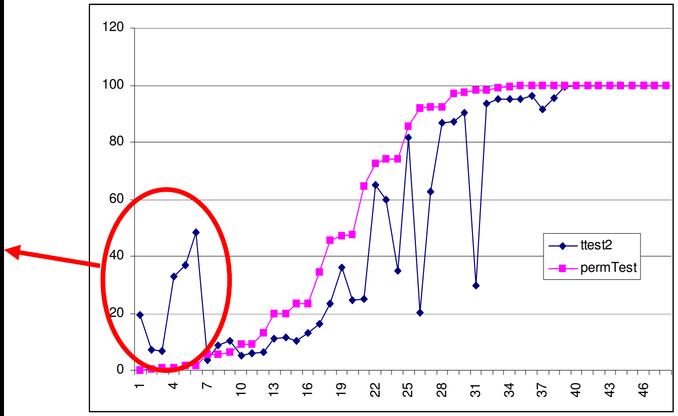
When?

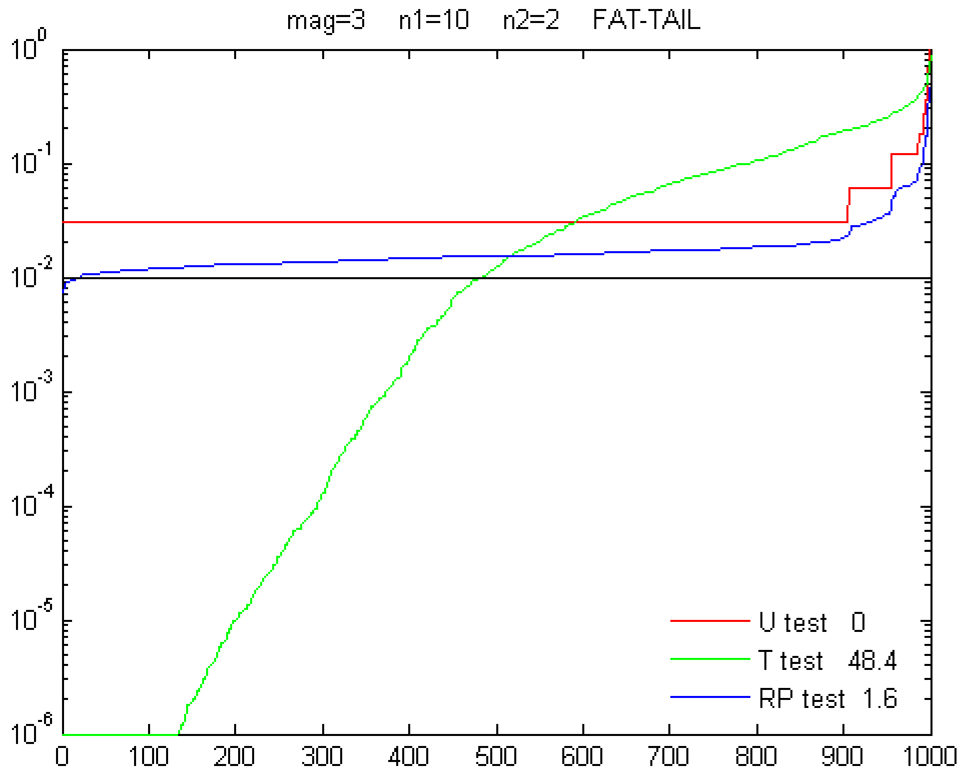
- *only* in cases where there are two comparison points n2
- *mostly* where there are 10 comparison points n1
  - or for very small difference

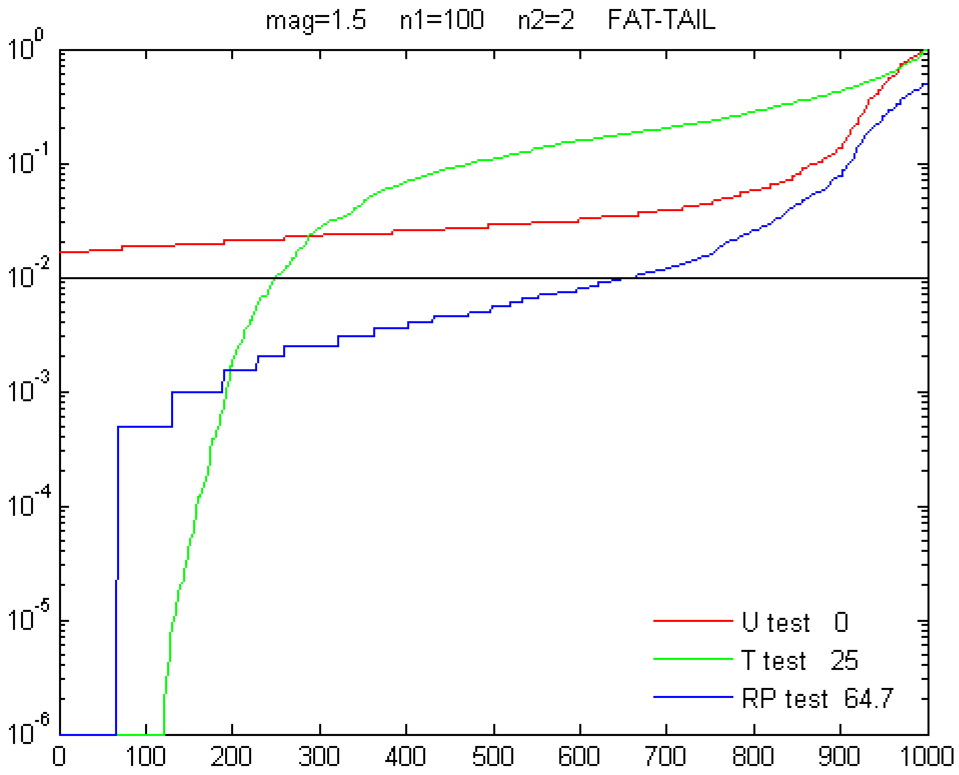
Why?

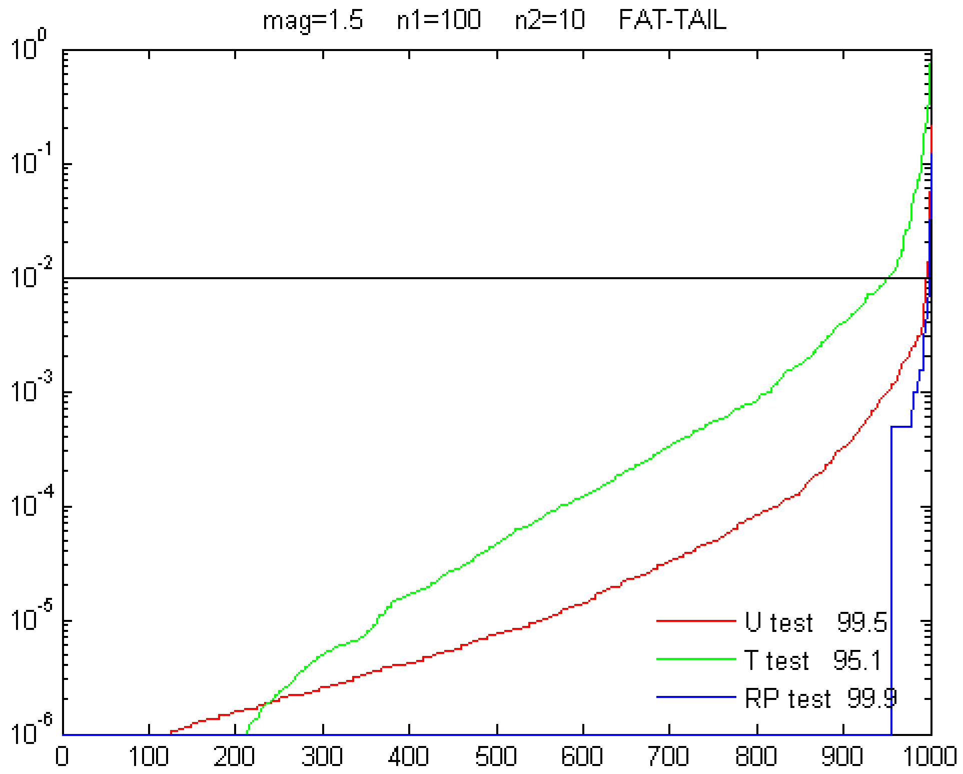
- Statistical theory is powerful stuff!
  - Permutation tests are less powerful for very small n.
  - also, 12 choose 2 is 66; 1/66=0.0152

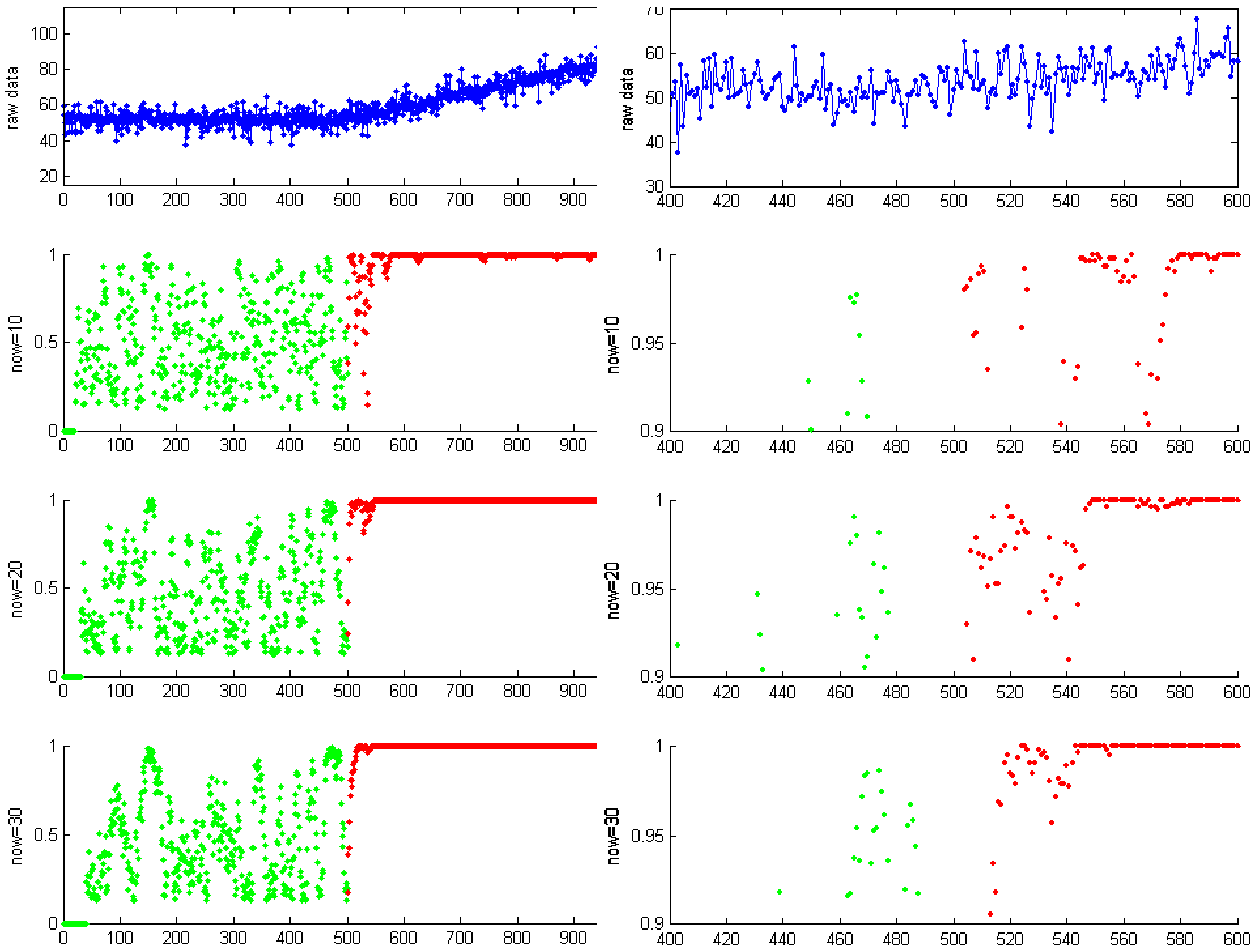
mag	n1	**n2**	dist	ttest2	permTest
3	10	**2**	fat-tail	48.4	1.6
3	10	**2**	normal	36.8	1.6
1.5	10	**2**	fat-tail	32.9	0.6
1.5	10	**2**	normal	19.5	0.1
0.5	10	**2**	normal	7.1	0.4
0.5	10	**2**	fat-tail	6.8	0.6
0.5	100	**2**	fat-tail	10.4	6.2
0.5	100	**2**	normal	8.7	5.6

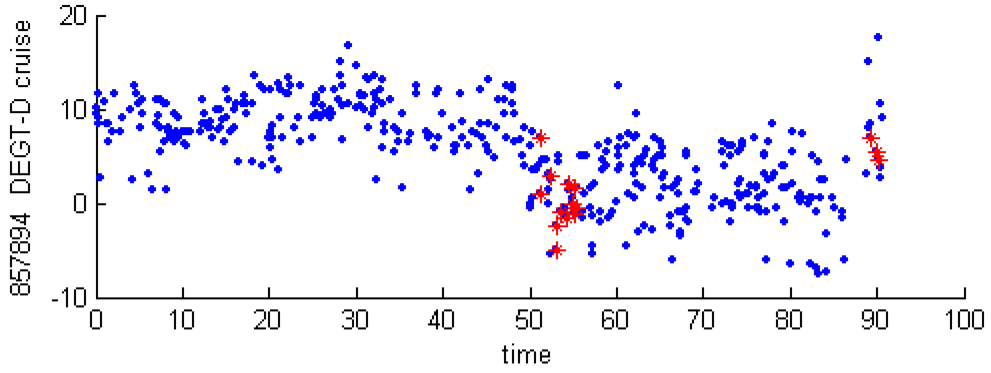


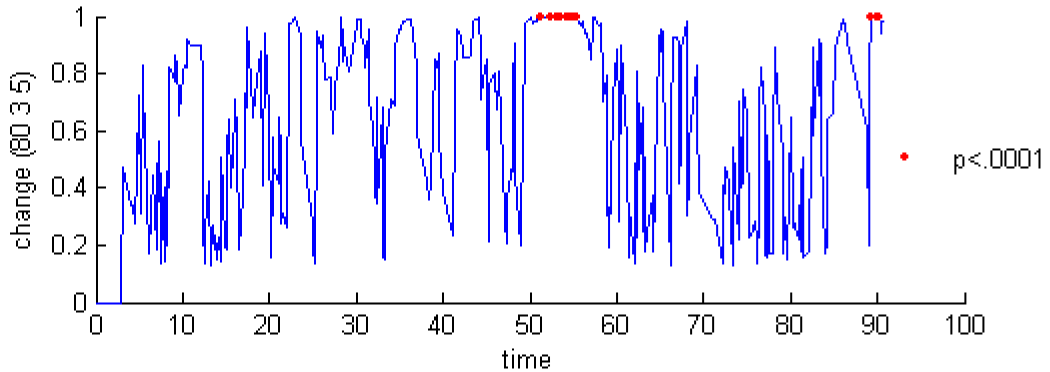

mag=3 n1=10 n2=2 FAT-TAIL
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
0
100
200
300
400
500
600
700
800
900
1000
U test 0
T test 48.4
RP test 1.6


mag=1.5 n1=100 n2=2 FAT-TAIL
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
0
100
200
300
400
500
600
700
800
900
1000
U test 0
T test 25
RP test 64.7

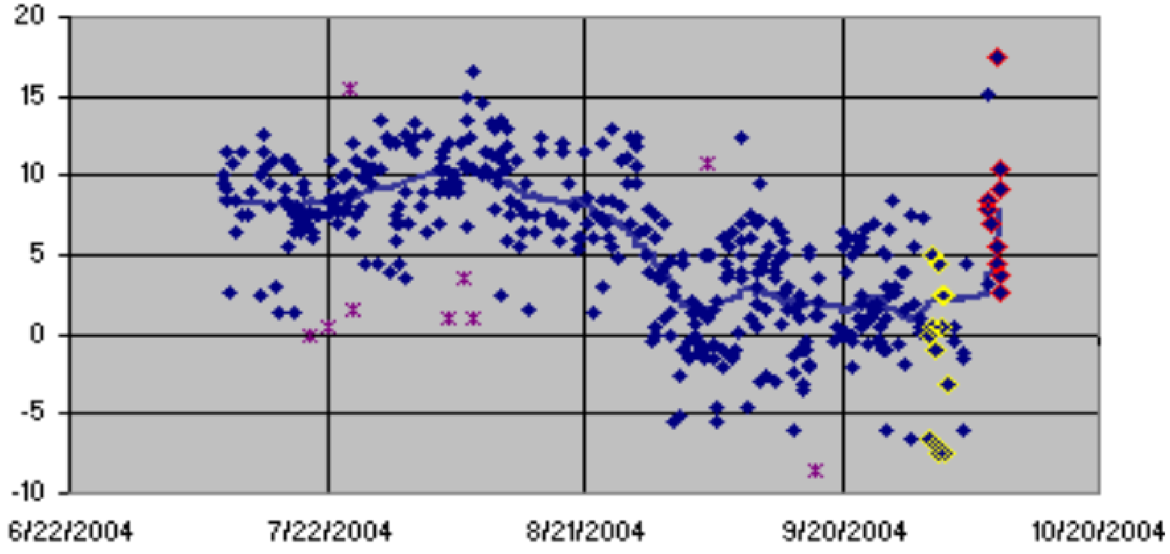

mag=1.5 n1=100 n2=10 FAT-TAIL
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
0
100
200
300
400
500
600
700
800
900
1000
U test 99.5
T test 95.1
RP test 99.9


raw data
100
80
60
40
20
0
100
200
300
400
500
600
700
800
900
raw data
70
60
50
40
30
400
420
440
460
480
500
520
540
560
580
600
now=10
1
0.5
0
0
100
200
300
400
500
600
700
800
900
now=10
1
0.95
0.9
400
420
440
460
480
500
520
540
560
580
600
now=20
1
0.5
0
0
100
200
300
400
500
600
700
800
900
now=20
1
0.95
0.9
400
420
440
460
480
500
520
540
560
580
600
now=30
1
0.5
0
0
100
200
300
400
500
600
700
800
900
now=30
1
0.95
0.9
400
420
440
460
480
500
520
540
560
580
600





DEGT_D - CRUISE



Wacky Tricks for AD #2:

Hotelling T^2

[multivariate, parametric]

Multivariate Parametric Data

Again, a very common problem…

- e.g., monitoring subsea hydraulic pumps

The Hotelling T-square statistic, t^2 , is a generalization of Student's t statistic that is used in multivariate hypothesis testing. Hotelling t^2 metric provides good sensitive to small drifts.

For a group of variables $x = (x_1, x_2, \ldots, x_p)$ with mean of $\mu = (\mu_1, \mu_2, \ldots, \mu_p)$, and covariance matrix

$$W = \sum(x - \mu)\ (x - \mu)'/(n-1)$$

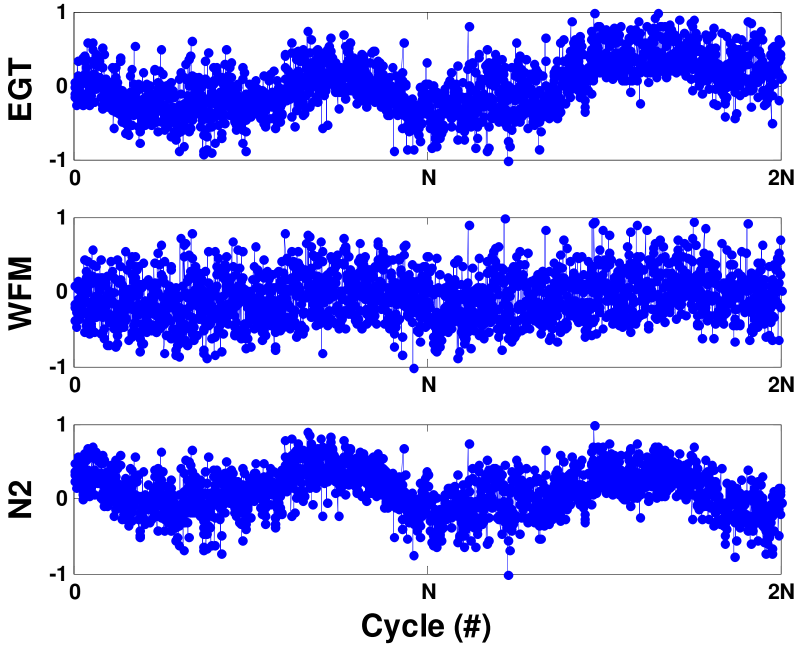
Then, the t-square statistic

$$\mathbf{t^2 = (x - \mu)'\ W^{-1}\ (x - \mu)}$$

[Looks much like squared Mahalanobis distance.]

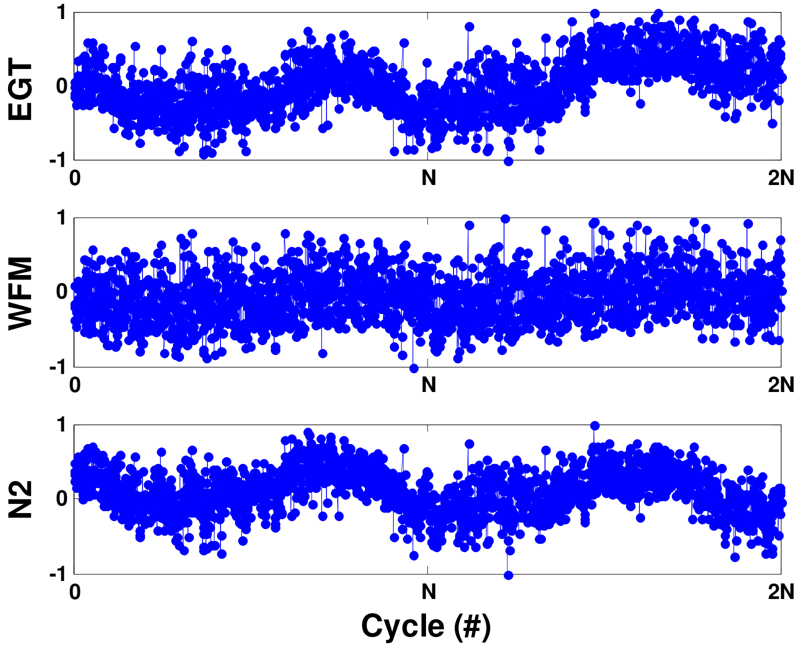
Hotelling T-square Stat: Example 1

Raw Measurement

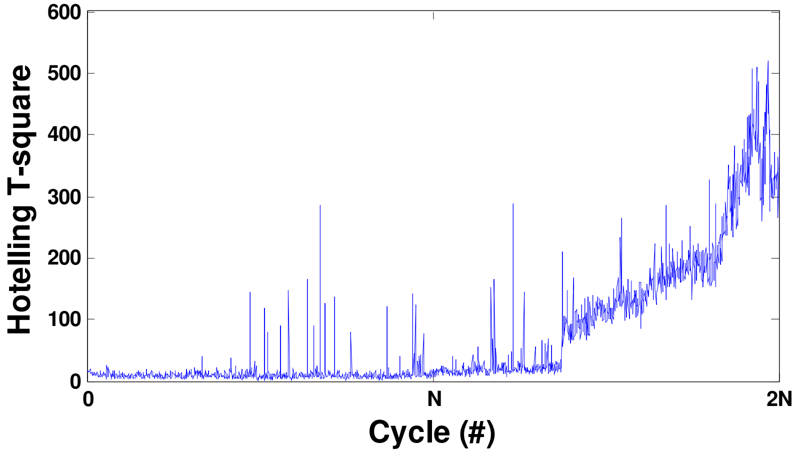


Hotelling T-square Stat: Example 1

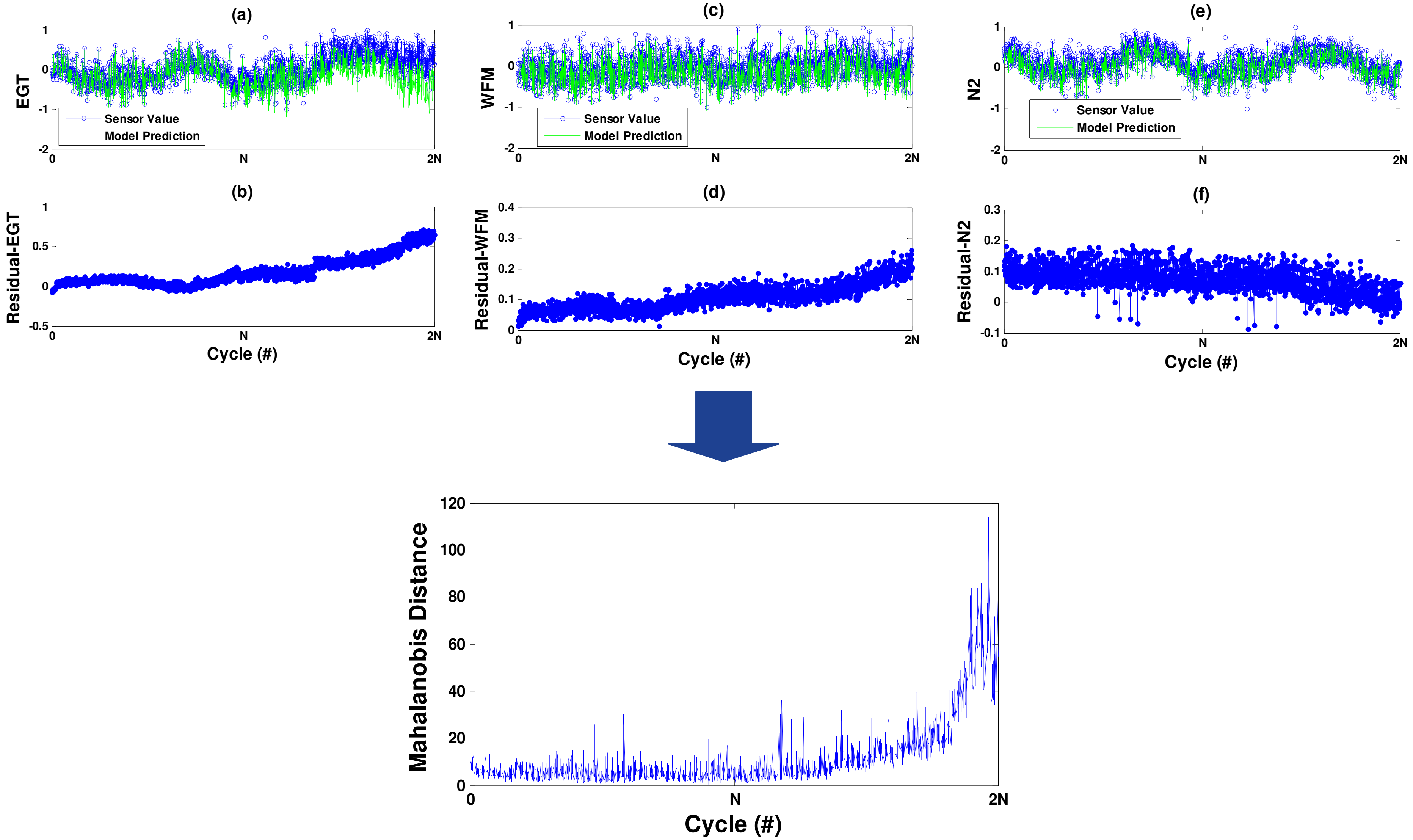
Raw Measurement



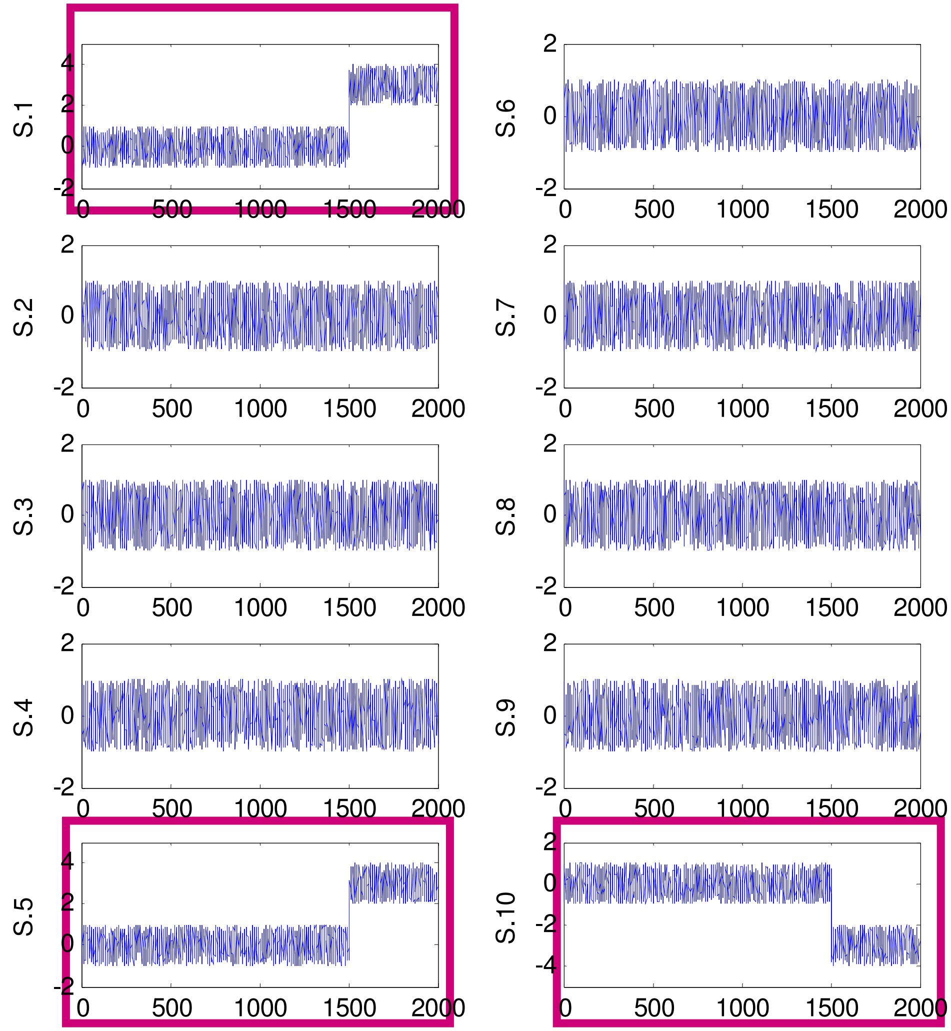
Hotelling T-square



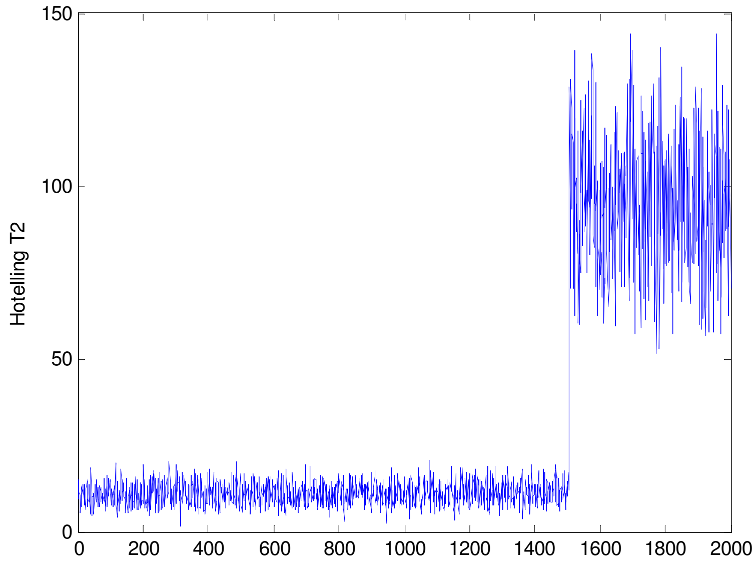
Hotelling T-square Stat: Example 2



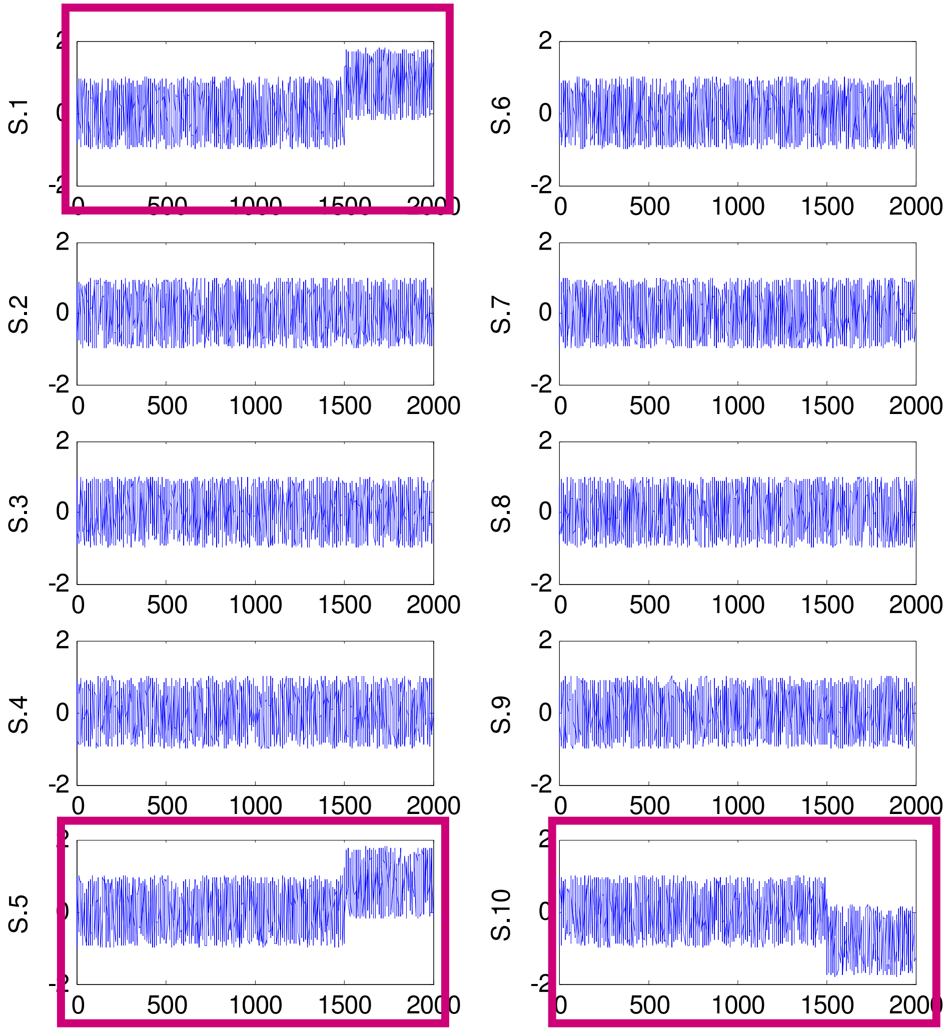
T2: Fault 1 – big Step Shift



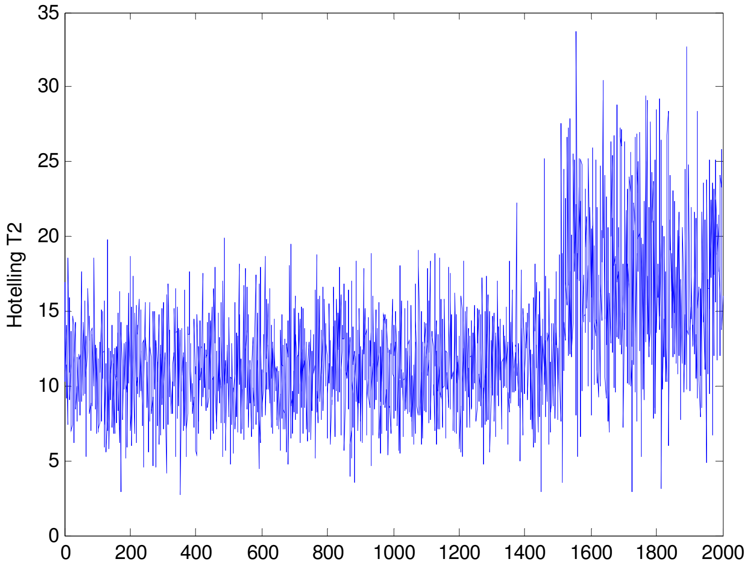
Hotelling T-square



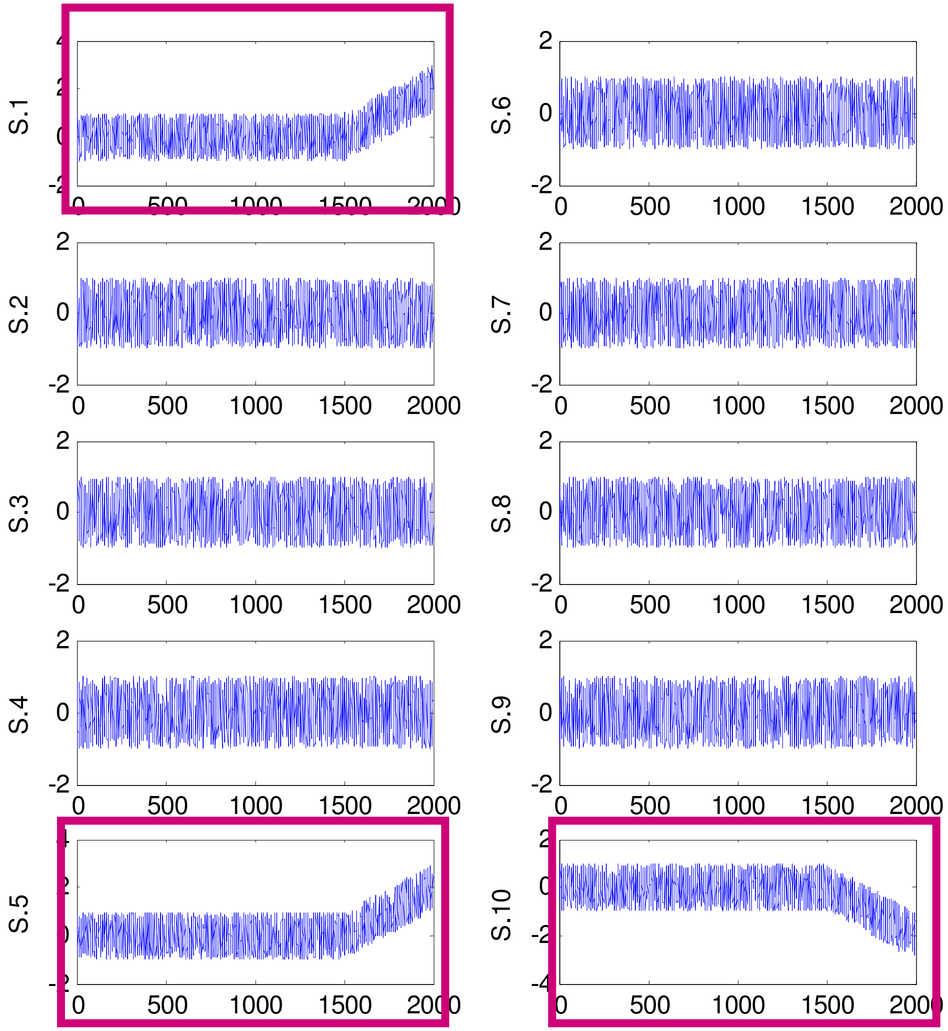
T2: Fault 2 – small Step Shift



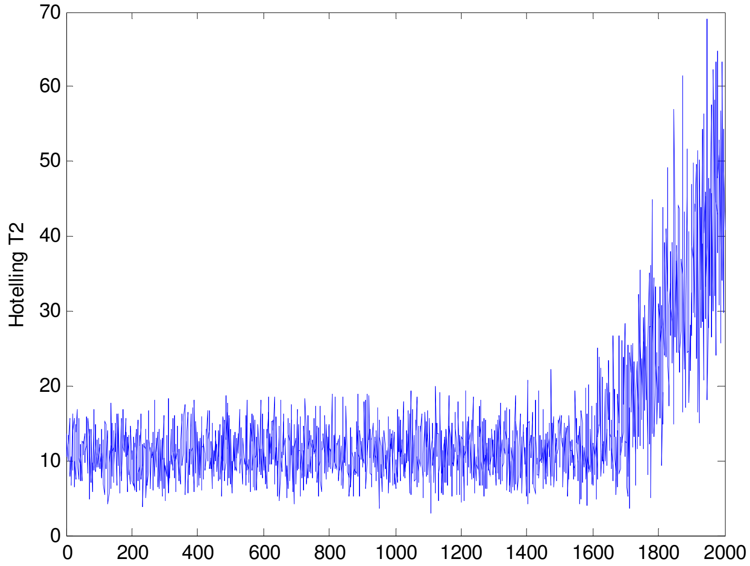
Hotelling T-square



Fault 2 – Slow Drift



Hotelling T-square



Aside #1:

Random Forest

in 30 slides or less!

Aside Outline

- bias and variance
- ensembles of classifiers
- bagging
- classification trees
- random forests

Stay with me! It makes sense in the end…

outline

- bias and variance
- ensembles of classifiers
- bagging
- classification trees
- random forests

bias and variance

bias:

- the classifier (regressor) cannot represent the true function
- i.e., the classifier (regressor) underfits the data

variance:

- variance arises when the classifier overfits the data

There is often a tradeoff between bias and variance:

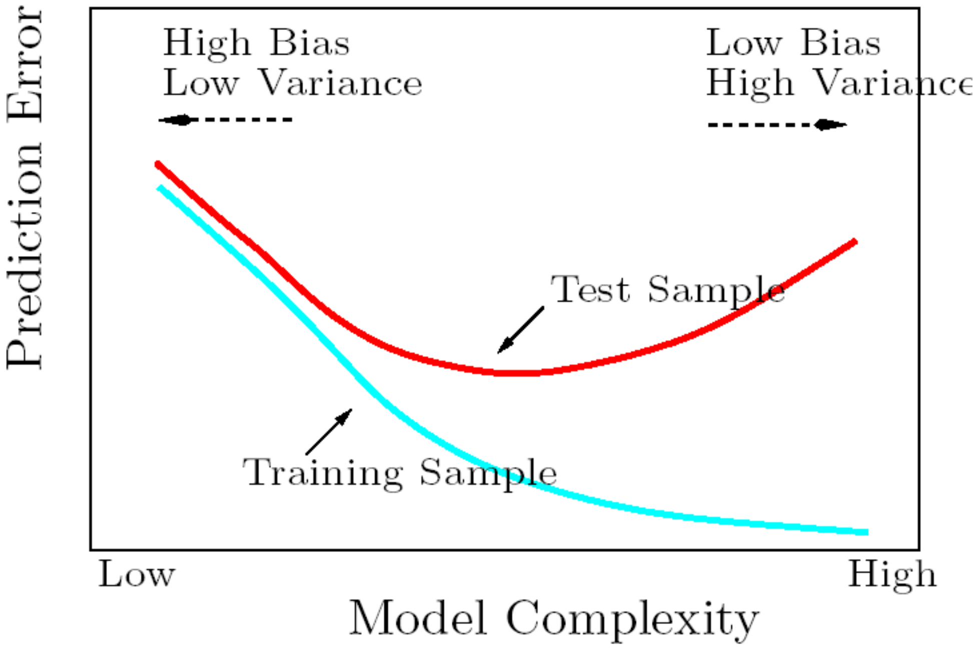


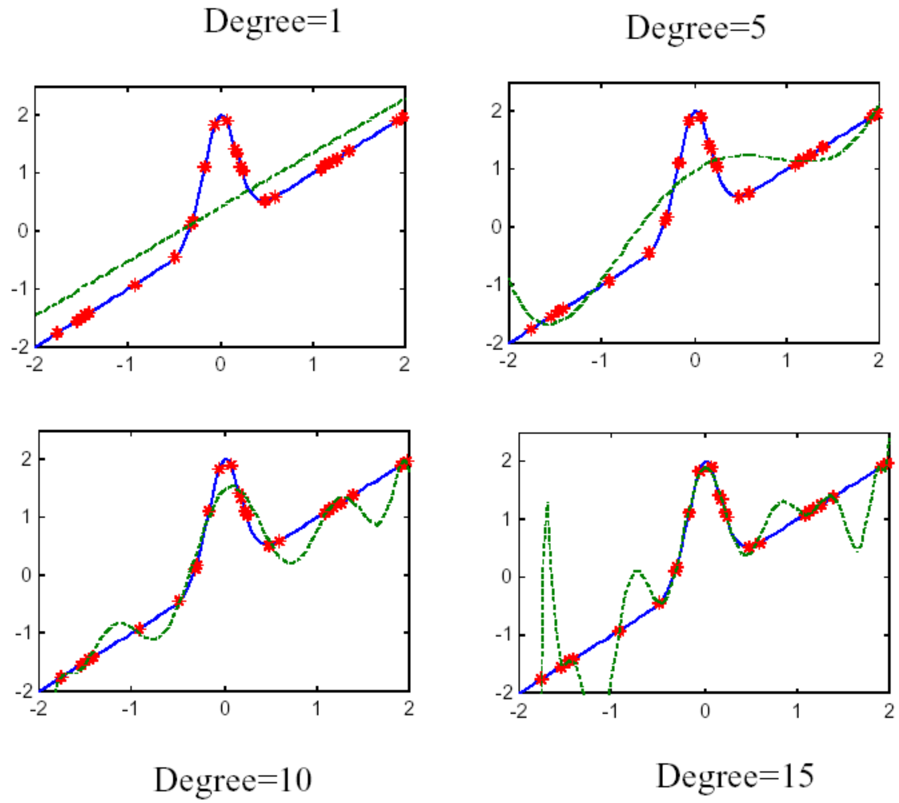
Fig. 2.11, Hasti, Tibshirani, Friedman

bias and variance example

red – experimental data
blue – underlying function
green - fit

large bias, small variance:

small bias, high variance:



Degree=1

Degree=5

Degree=10

Degree=15

outline

- bias and variance
- ensembles of classifiers
- bagging
- classification trees
- random forests

ensembles of classifiers

For any single classifier, there is typically a tradeoff between bias and variance.

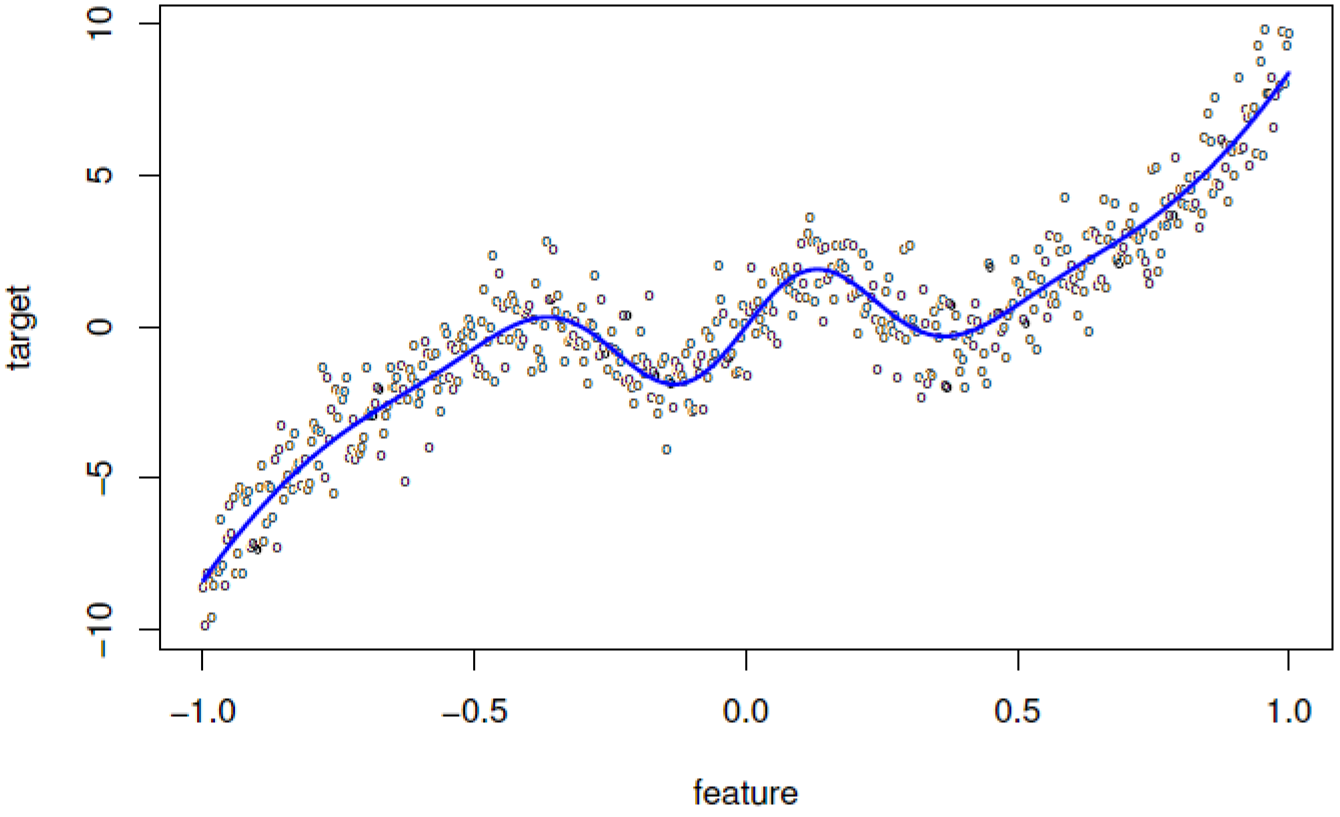
Might we achieve high accuracy by combining ensembles of high variance (i.e., uncorrelated), low bias classifiers?

- variance is reduced by combining outputs
- bias remains low

basic idea:

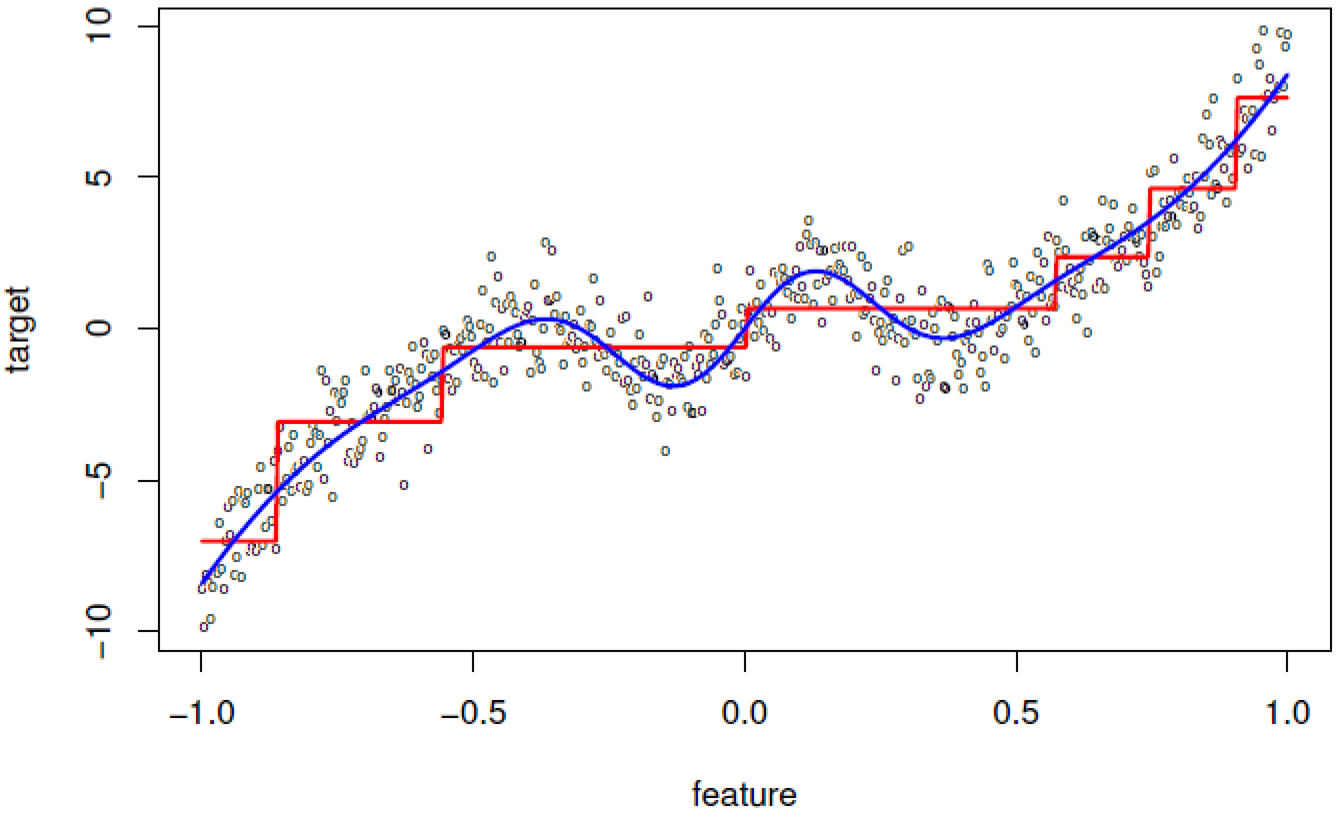
train a *set* of *diverse* classifiers (or regressors) and combine their output

blue – underlying function
black – data with noise



large bias…

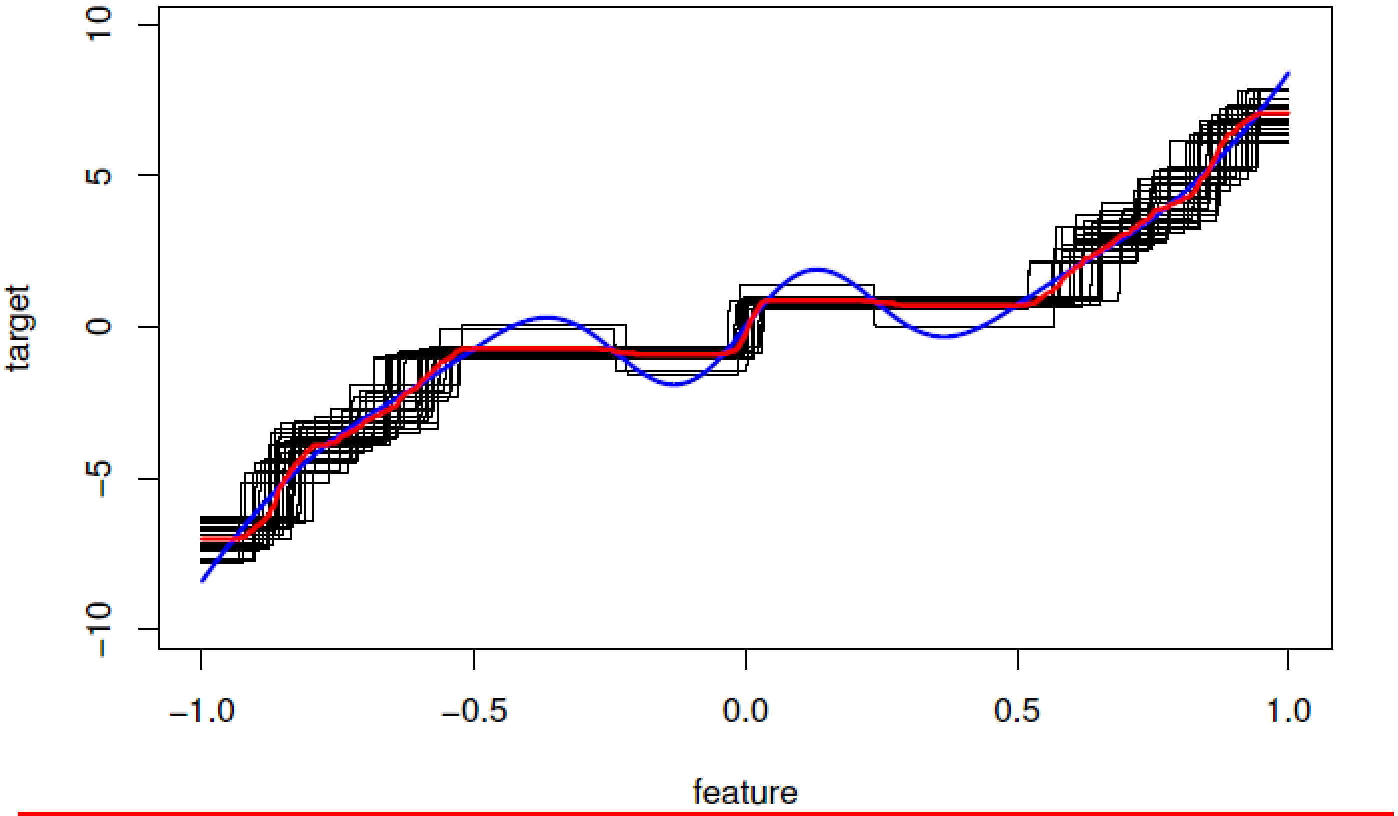
red - fit



large bias… small variance

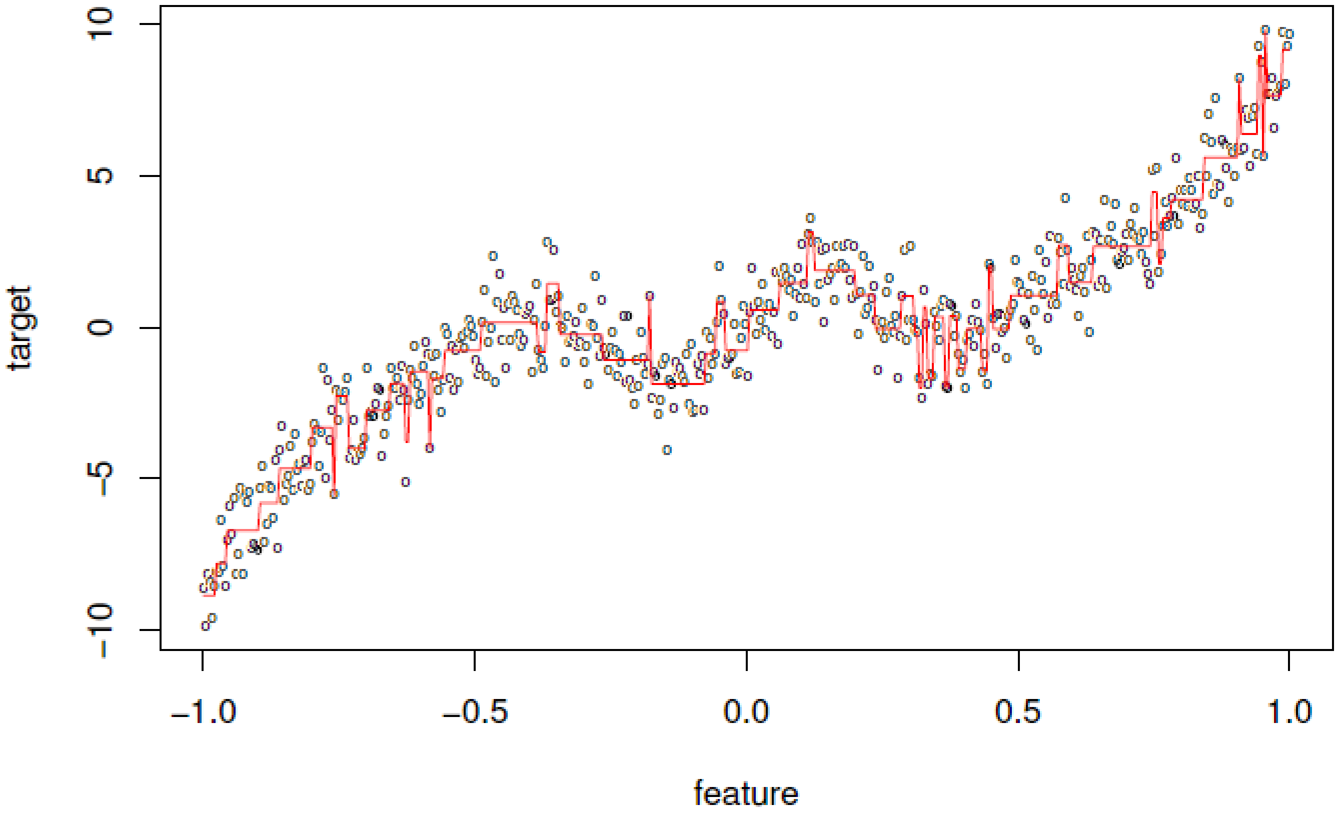
black – multiple fits

red – average of fits

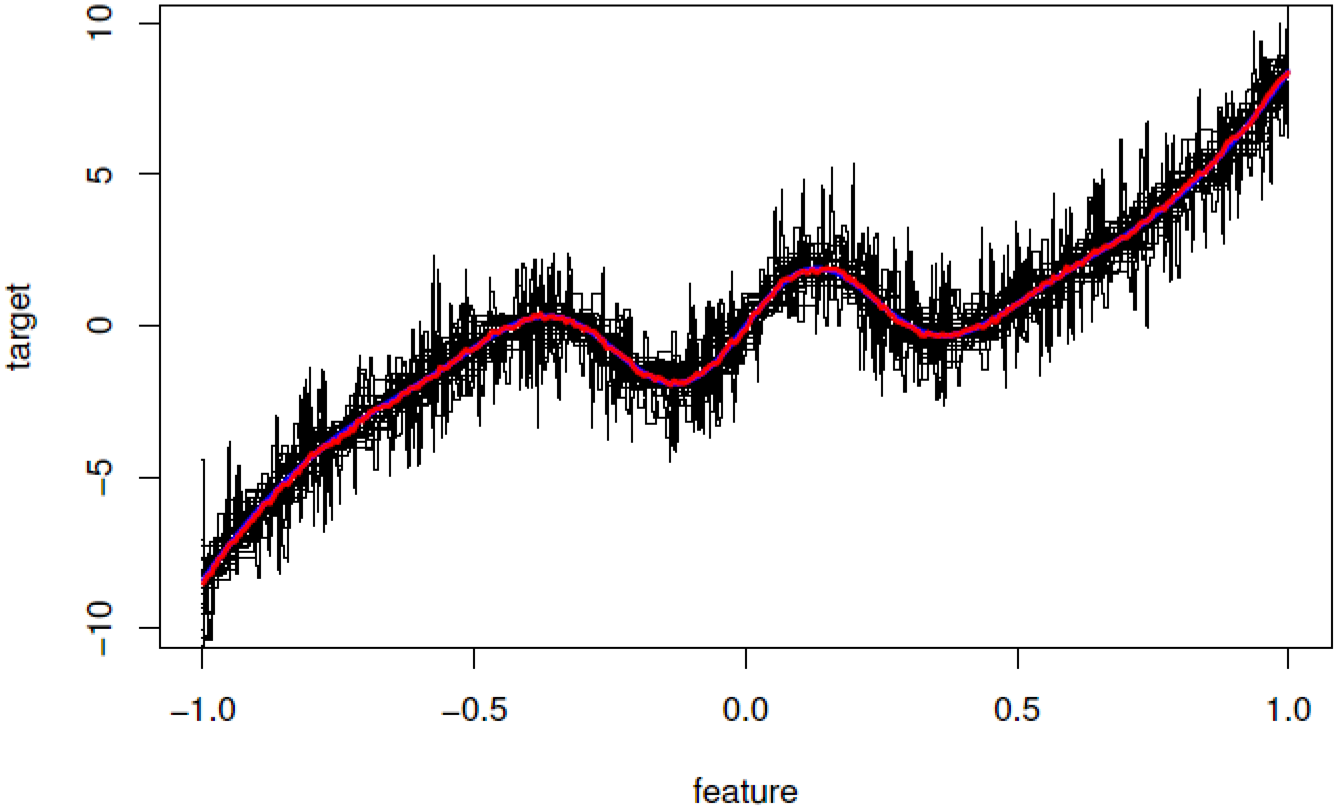


poor ensemble fit

small bias…



small bias…high variance



fantastic ensemble fit

outline

- bias and variance
- ensembles of classifiers
- bagging
- classification trees
- random forests

bagging

Bootstrap AGGregation (BAGGing)

- create multiple bootstrap samples
  - given a training set *D* of size *N*
  - generate *L* new training sets D_i also of size *N* by sampling cases uniformly from *D* with replacement
    - sampling with replacement it is likely that some examples will be repeated in each D_i
    - on average the set D_i will have 63.2% of the examples of *D*, the rest being duplicates
- train a classifier on each sample
- combine output of classifiers by voting

Good for unstable classifiers (i.e., small bias) – otherwise different classifiers aren't very diverse.

- e.g., good with decision trees
- e.g., bad with naïve Bayes

outline

- bias and variance
- ensembles of classifiers
- bagging
- classification trees
- random forests

Why tree methods?

nominal data: no metric

- descriptions that are discrete and without notion of similarity or even ordering
- examples: Yes/No; True/False; chicken/steak/pasta

rule-based vs. PDF/metric

syntactic pattern recognition vs. statistical pattern recognition

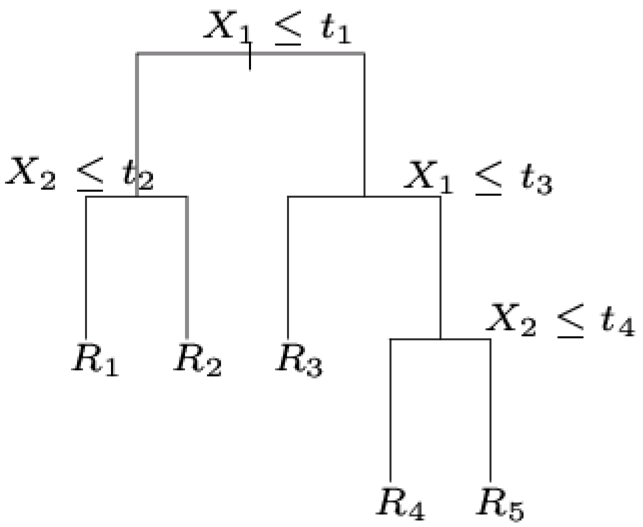
general tree method:

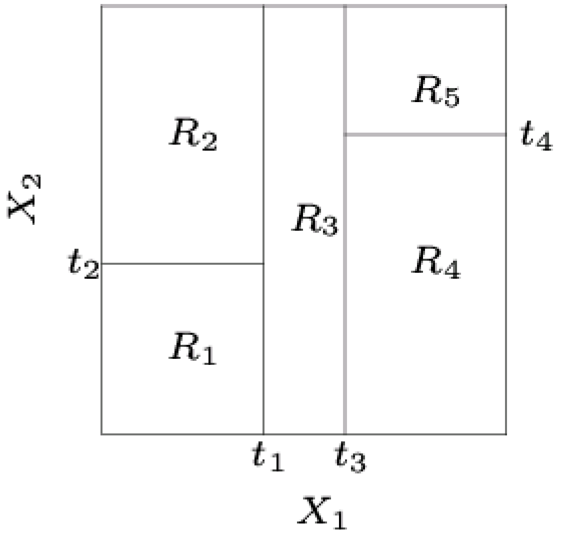
- split the feature space into a set of regions
- Regression tree (RT): Fit a regression model for each partition region
- Classification tree (CT): Assign a class label for each partition region

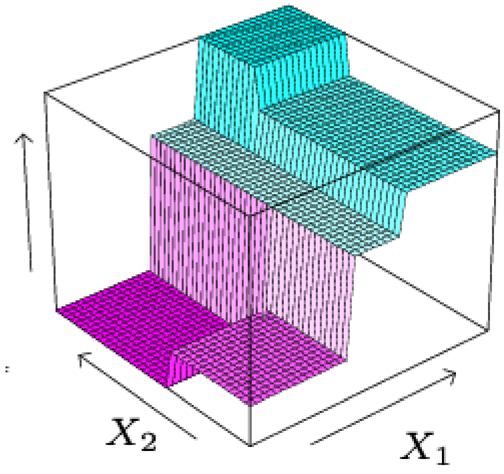
classification (regression) trees (CART, C4.5, etc.)

binary recursive partitioning

- binary: split parent node into two child nodes
  - *look at all features at each split, and choose best one*
- recursive: each child node can be treated as parent node
- partitioning: data set is partitioned into mutually exclusive subsets in each split
- *prune tree to get good generalization*







classification tree example

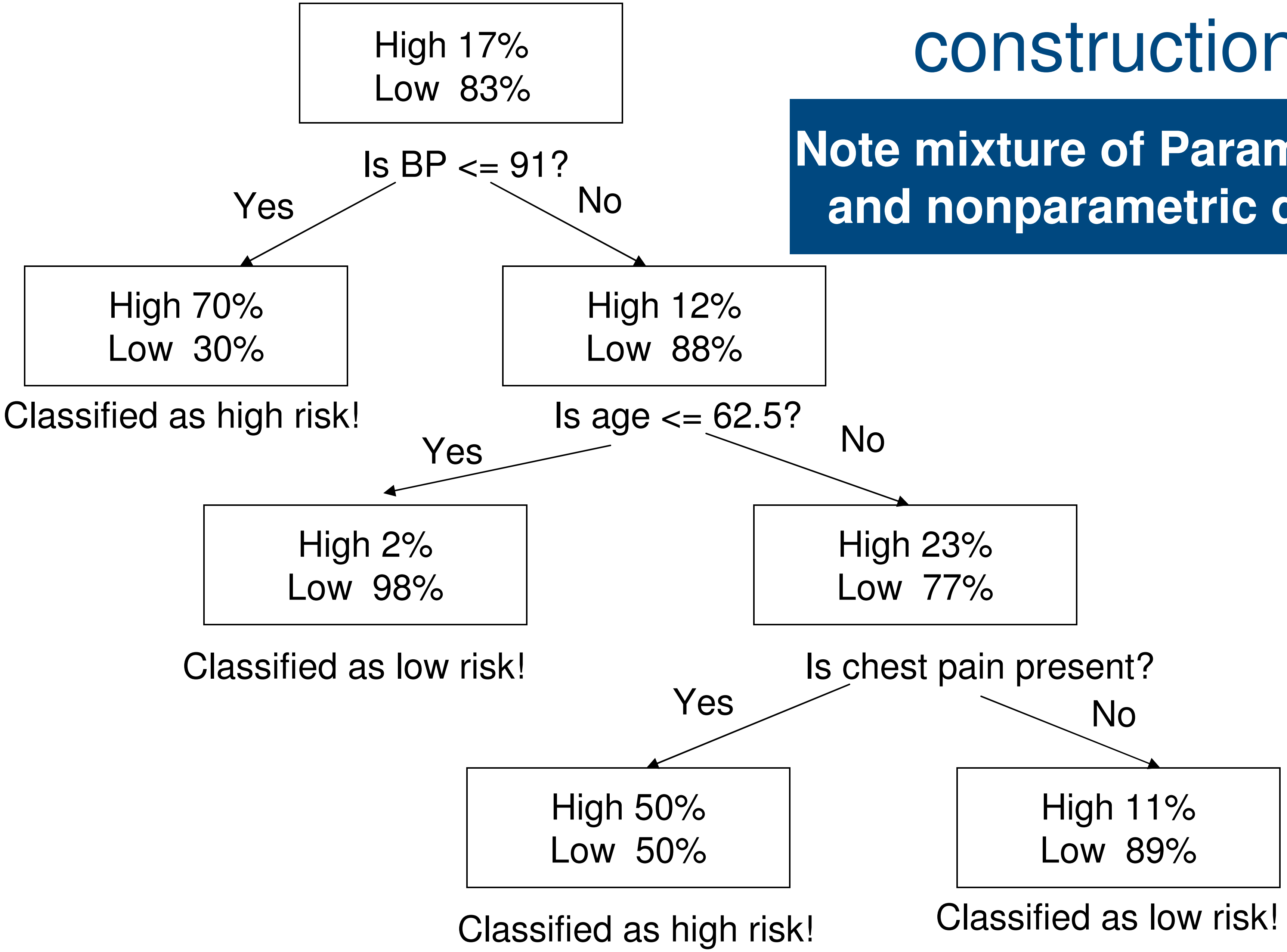
Goal: For the patients admitted into ER, to predict who is at higher risk of heart attack

Training data set:

- # of subjects = 215
- Outcome variable = High/Low Risk determined
- 19 noninvasive clinical and lab variables were used as the predictors

classification tree construction

Note mixture of Parametric and nonparametric data!



High 17%
Low 83%

Is BP <= 91?

Yes

High 70%
Low 30%

Classified as high risk!

No

High 12%
Low 88%

Is age <= 62.5?

Yes

High 2%
Low 98%

Classified as low risk!

No

High 23%
Low 77%

Is chest pain present?

Yes

High 50%
Low 50%

Classified as high risk!

No

High 11%
Low 89%

Classified as low risk!

outline

- bias and variance
- ensembles of classifiers
- bagging
- classification trees
- random forests

random forests (RF)

Bagging decision trees with “randomization injection”.

- create multiple bootstrap samples
- train a decision tree on each sample
  - *at each node, select a random subset of variables to split on*
  - *grow trees to maximum depth (i.e., no pruning)*
- combine resulting trees by voting

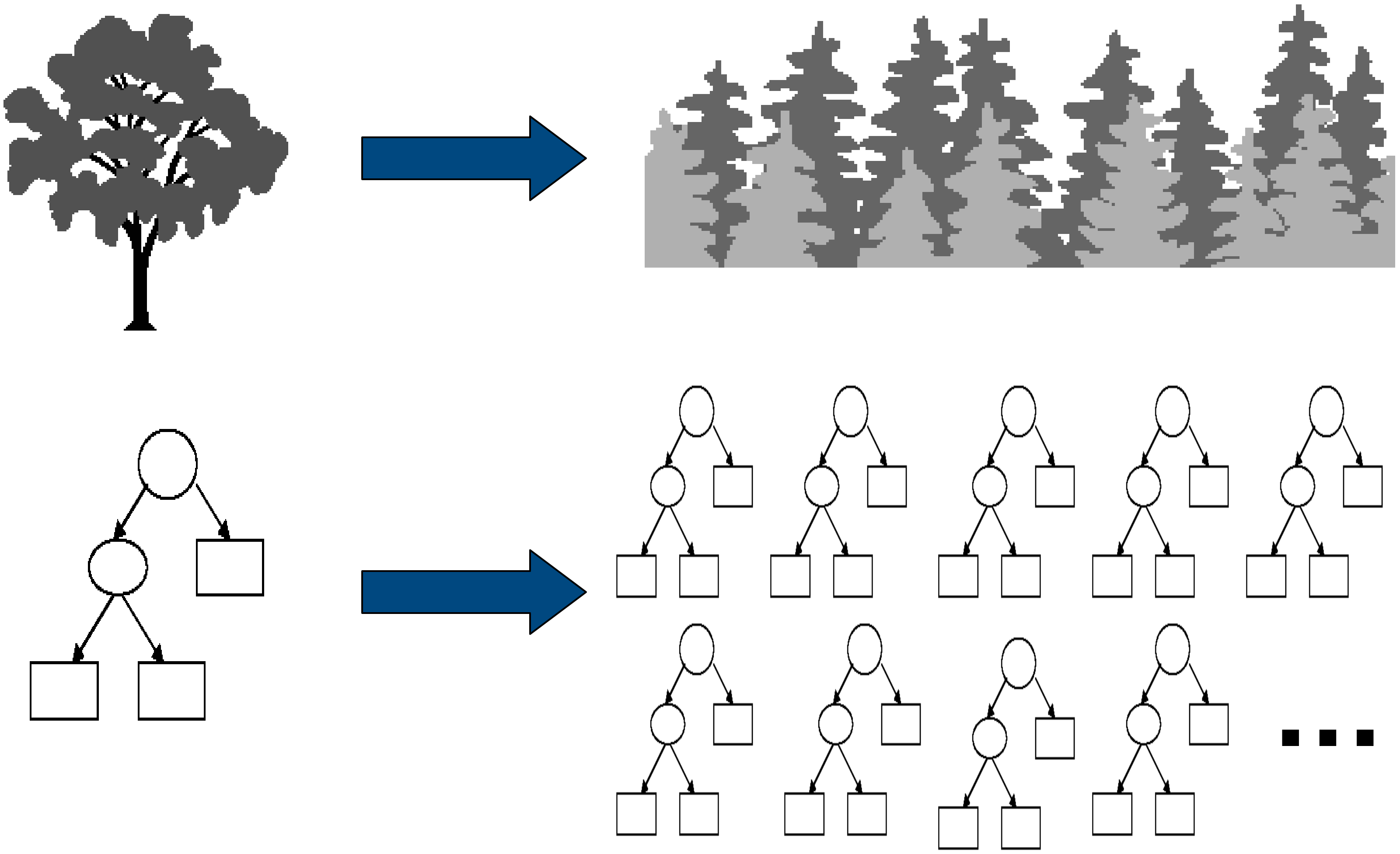
properties of RF

- test set error rates (modulo a little noise) are monotonically decreasing and converge to a limit
  - i.e., there is no overfitting as the number of trees increases

The key to accuracy is low correlation (high variance across trees) and low bias:

- to maximize variance, randomness in variable selection is introduced
- to minimize bias, trees are grown to maximum depth

RF construction



growing each tree

each tree is grown as follows:

- If the number of cases in the training set is N, sample N cases at random with replacement, from the original data. This sample will be the training set for growing the tree.
- If there are M input variables, a number m<<M is specified such that at each node, m variables are selected at random out of the M and the best split on these m is used to split the node. The value of m is held constant during the forest growing.
- Each tree is grown to the largest extent possible. There is no pruning

prediction by plurality voting

- The forest consists of N trees.
- To classify a new object from an input vector, we put the input vector down each of the trees in the forest.
- Each tree gives a classification, and we say the tree “votes” for that class.

- The forest chooses the classification having the most votes (over all the trees in the forest).
  - class prediction: each tree votes for a class; the predicted class C for an observation is the plurality:

$$\max_C \Sigma_k [f_k(\mathbf{x},\mathbf{T}) == C]$$

  - regression: predicted value is the average prediction

out-of-bag (oob) error estimate

In RF, *there is no need for cross-validation* or a separate test set to get an unbiased estimate of the test set error. It is estimated internally, during the run, as follows:

- Each tree is constructed using a different bootstrap sample from the original data. About one-third of the cases are left out of the bootstrap sample and not used in the construction of the k^{th} tree.
- Put each case left out in the construction of the k^{th} tree down the k^{th} tree to get a classification. In this way, a test set classification is obtained for each case in about one-third of the trees. At the end of the run, take j to be the class that got most of the votes every time case n was oob. The proportion of times that j is not equal to the true class of n averaged over all cases is the oob error estimate. This has proven to be unbiased in many tests.

forest error rate

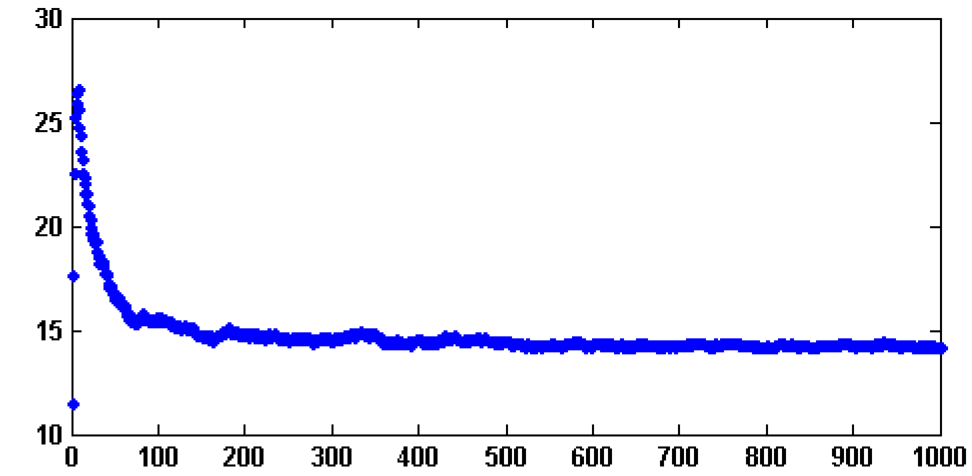
the forest error rate depends on two things:

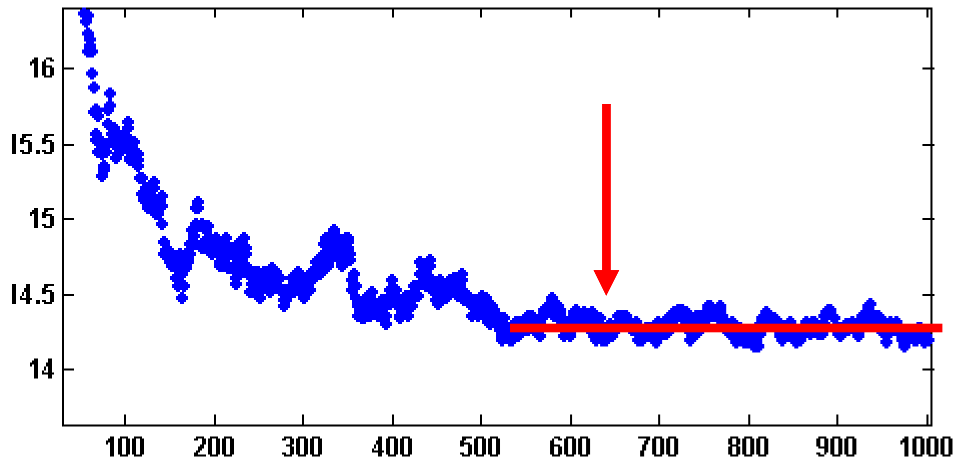
- the correlation between any two trees in the forest
  - increasing the correlation increases the forest error rate
- the strength of each individual tree in the forest
  - a tree with a low error rate is a strong classifier
  - increasing the strength of the individual trees decreases the forest error rate
- reducing m reduces both correlation and strength
- increasing m increases both
- there is an “optimal” range of m - usually quite wide

This (m) is the only adjustable parameter to which random forests is somewhat sensitive.

Using the out of bag (OOB) error rate a value of m in the range can quickly be found.

typical RF error profile





some properties of RF

- one of the most accurate machine learning approaches
    - accuracy is as good as Adaboost and sometimes better
    - some modified versions of RF (e.g., rotation forests) may be more accurate, but lack variable importance feedback
- relatively robust to outliers and noise
- faster than bagging or boosting
- gives useful internal estimates of generalization error and variable importance
- simple and easily parallelized
- robust to high dimensionality, correlation among inputs
- good for feature selection/dimensionality reduction
- very fast to calculate
    - calculate splitting criterion for only m variables at each branch

Wacky Tricks for AD #3:

RFAD

[multivariate, mixed parametric & nonparametric]

Multivariate Parametric & Nonparametric Data

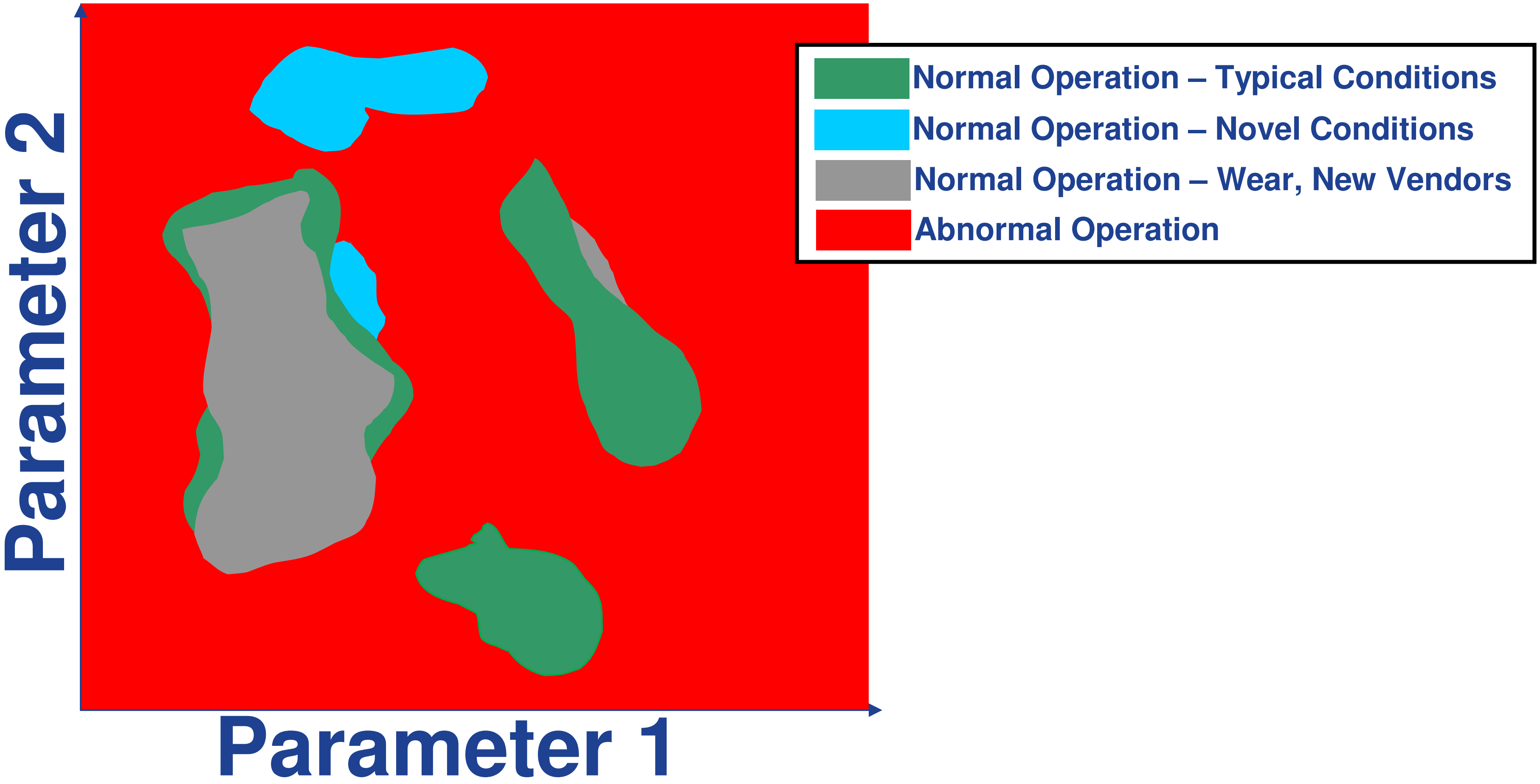
A very common problem! Sensed parameters and fault codes available in all kinds of data sets…

- aircraft airframes
- CT scanners
- turbines (engines, combined cycle, etc.)
- paper manufacturing
- financial data
- subsea oil extraction machinery
- locomotive

Problem: typical classification algorithms can't
easily accommodate mixed parametric
and nonparametric data!

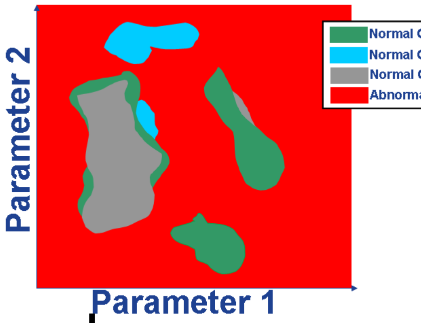
[Yeah, there are some work-arounds (I will describe one), but generally… no joy!]

Recall, the nature of Anomaly Detection


Parameter 2
Parameter 1
Normal Operation – Typical Conditions
Normal Operation – Novel Conditions
Normal Operation – Wear, New Vendors
Abnormal Operation

A two-class problem: normal operation & abnormal operation.

But how to train a classifier?

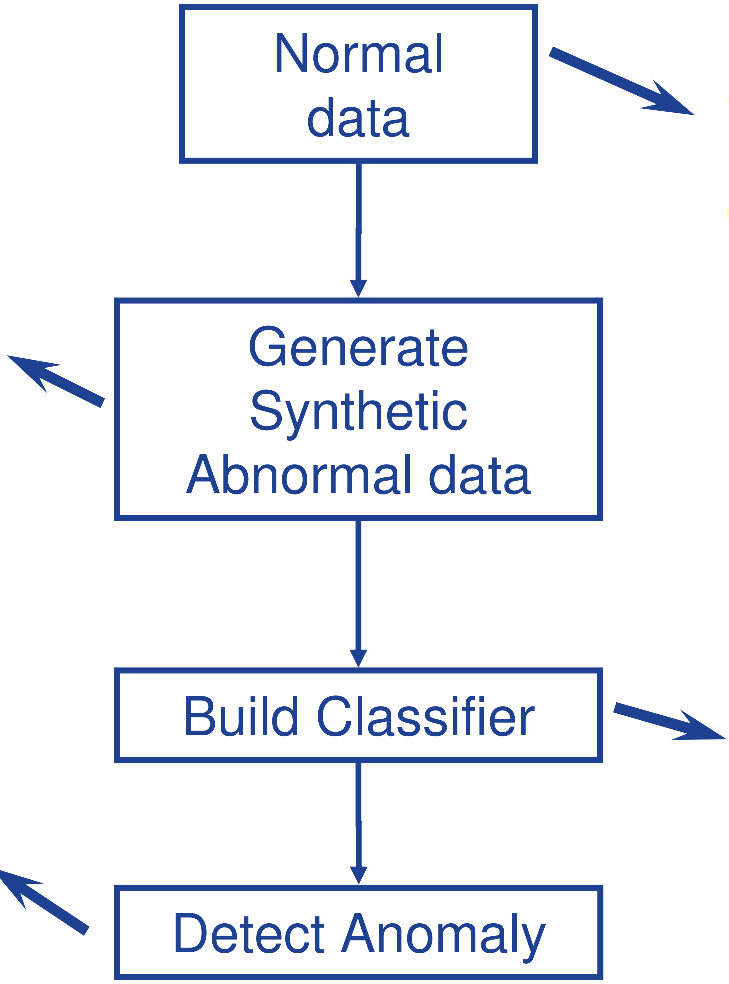


- Anomaly detection can be thought of as a two class classification problem
- However data is generally extremely unbalanced - way more normal data than abnormal
  - That’s why they call it “anomalous”!

How do you train a classifier with incredibly unbalanced data? *Make up the minority class*!

- Label all of the “normal operation” data “class 0”
- Fill the entire space with fake data, label it “class 1”
  - Literature suggests use marginal distribution
  - Experience shows a uniform distribution works better for a wide range of problems.
    - Better still, use a different class 1 realization every *n* trees or so.

Random Forest Anomaly Detection

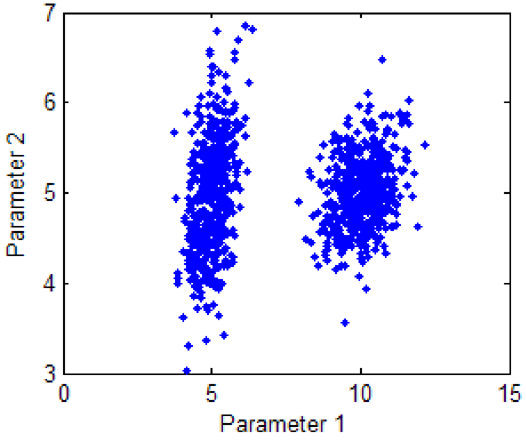


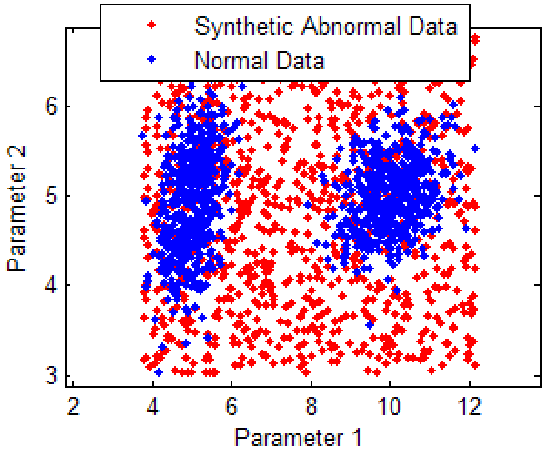
Normal data

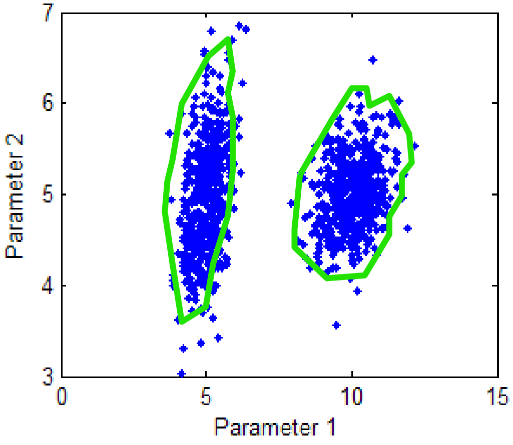
Generate Synthetic Abnormal data

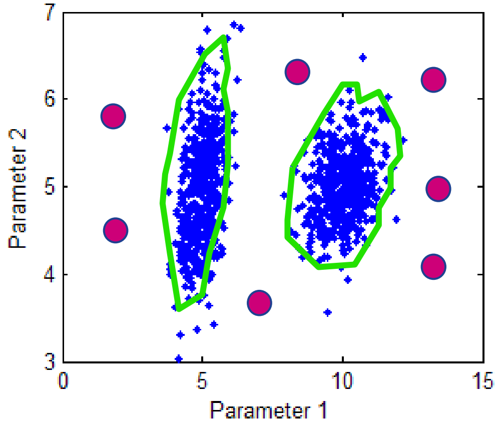
Build Classifier

Detect Anomaly









- No need to identify # of clusters
- Non-parametric decision boundary

Not necessarily RFAD

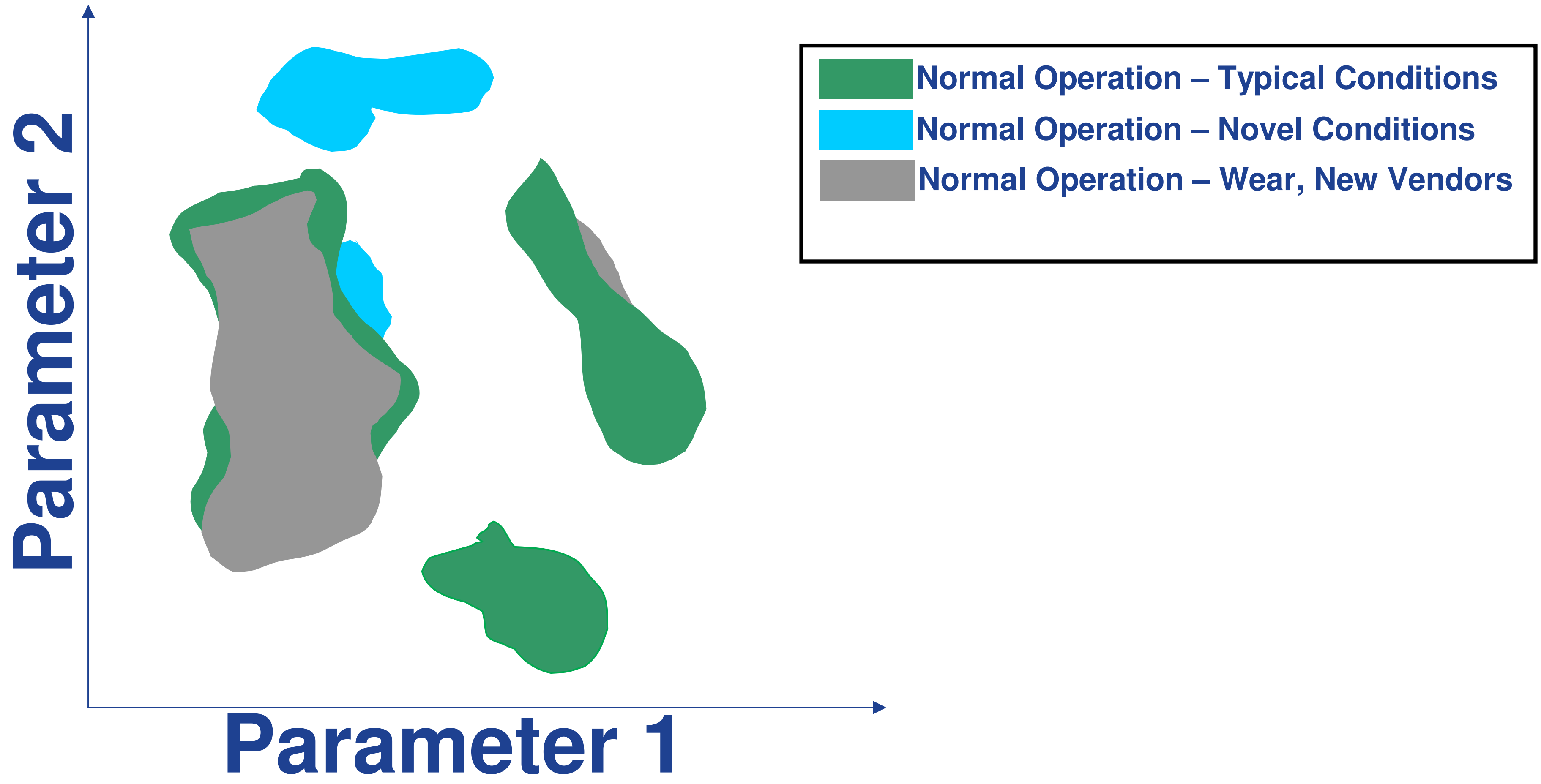
As you may have heard, RF have some wonderful properties:

- Fast
- Accurate
- Robust to noise, correlation, high dimensionality
- Trivially easy to accommodate mixed parametric and nonparametric data

However, if you have some classifier you prefer – e.g., neural networks – you can use the same fake data trick. Just not nearly as elegant…

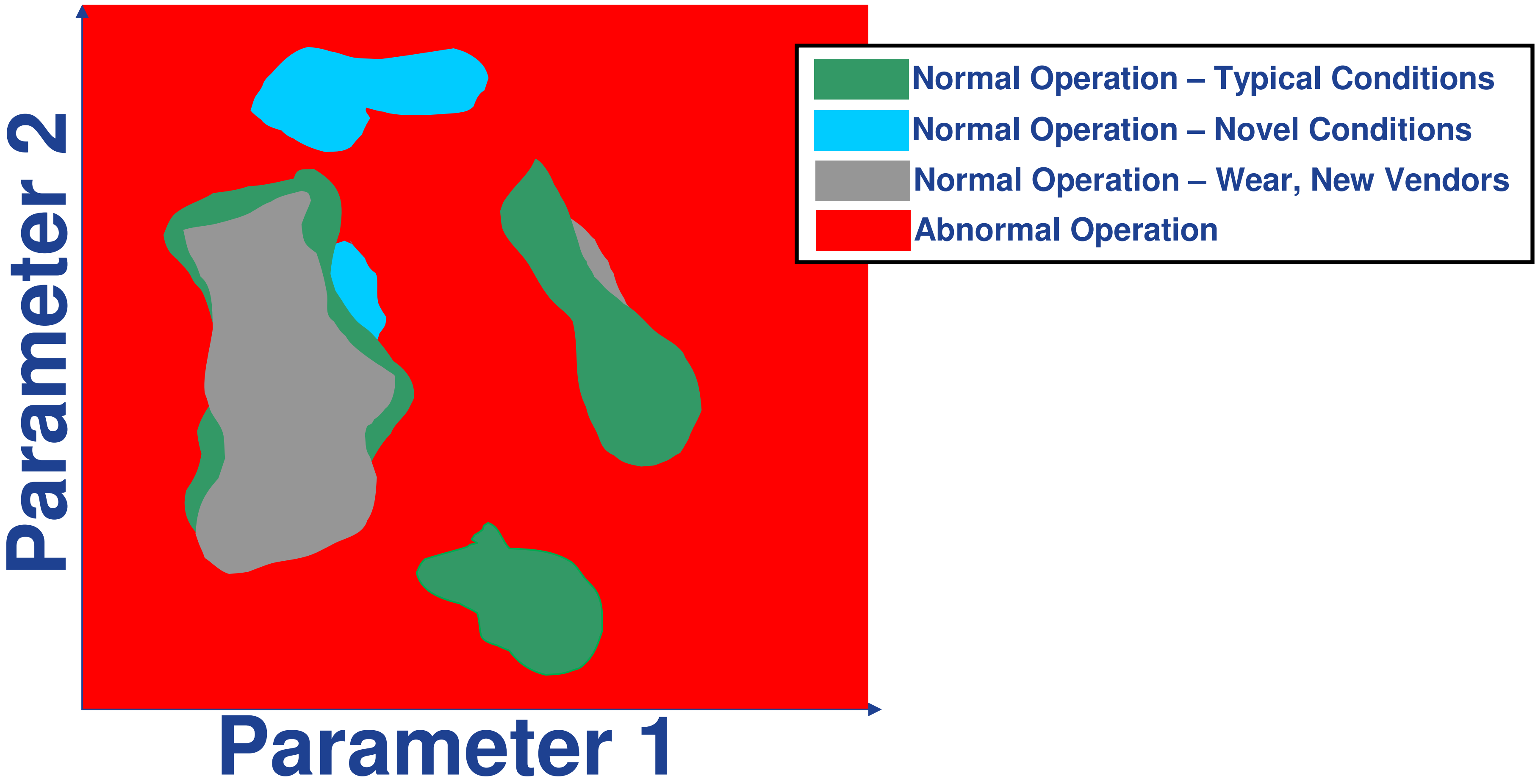
Diagnosis

Recall, Normal Operation

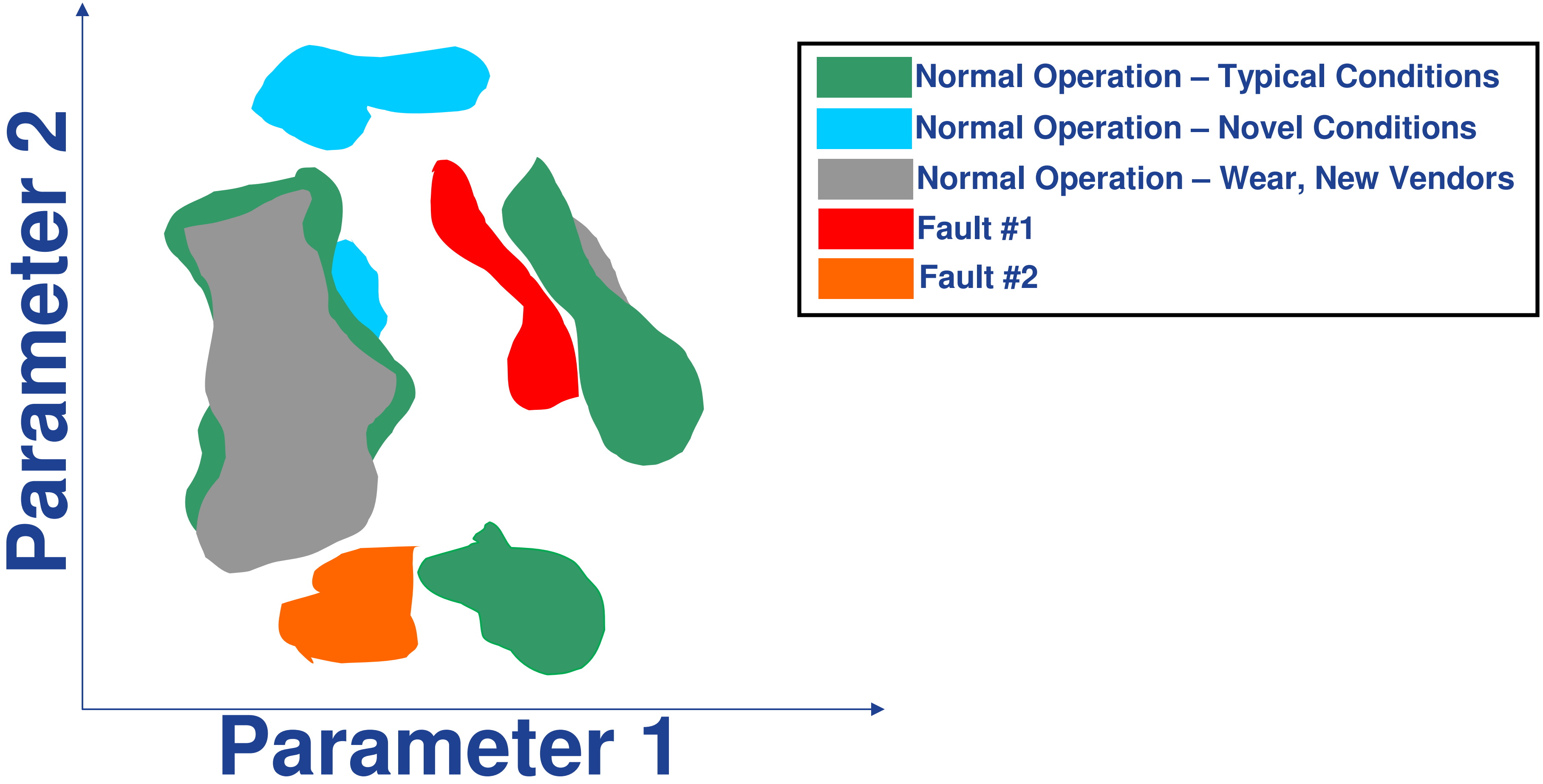


Parameter 2
Parameter 1
🟩 Normal Operation – Typical Conditions
🟦 Normal Operation – Novel Conditions
⬜ Normal Operation – Wear, New Vendors

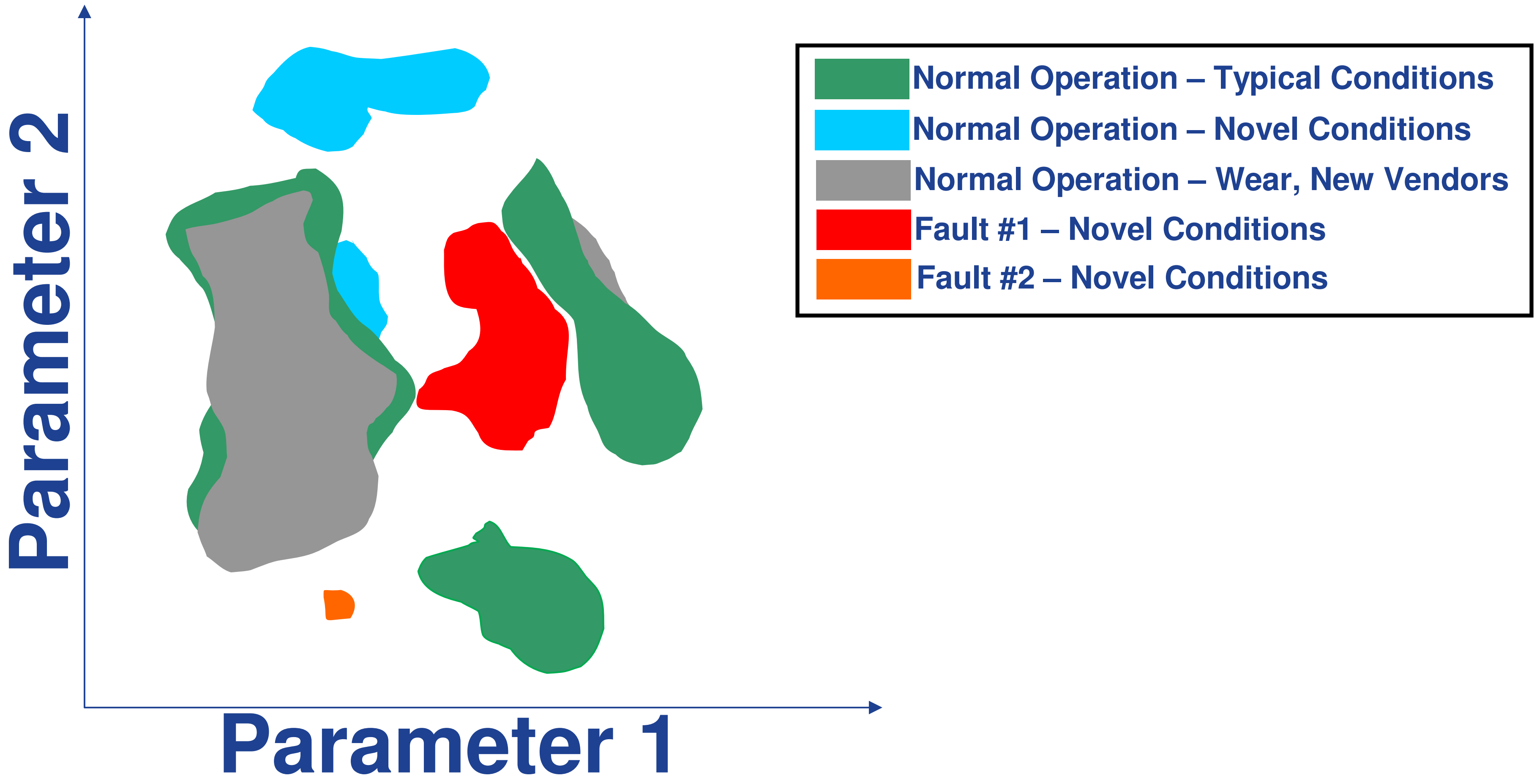
Anomaly Detection


Parameter 2
Parameter 1
Normal Operation – Typical Conditions
Normal Operation – Novel Conditions
Normal Operation – Wear, New Vendors
Abnormal Operation

Fault Identification


Parameter 2
Parameter 1
Normal Operation – Typical Conditions
Normal Operation – Novel Conditions
Normal Operation – Wear, New Vendors
Fault #1
Fault #2

Fault Identification


Parameter 2
Parameter 1
🟩 Normal Operation – Typical Conditions
🟦 Normal Operation – Novel Conditions
⬜ Normal Operation – Wear, New Vendors
🟥 Fault #1 – Novel Conditions
🟧 Fault #2 – Novel Conditions

Faults will *also* manifest differently as conditions change...

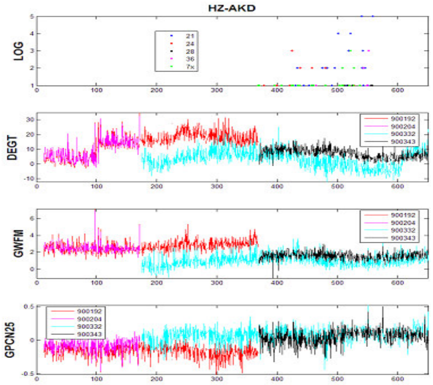
The Nature of Diagnosis

- Data driven diagnosis generally a multiclass classification problem
- Data can be real or – if you have a good model – synthetic
  - Again, the issue of very small minority class
    - But here it can’t be solved with fake data!
- Issues of observability
  - Some faults can’t be seen
  - Some faults are confounded
    - Add more sensors? Ha! Good luck…

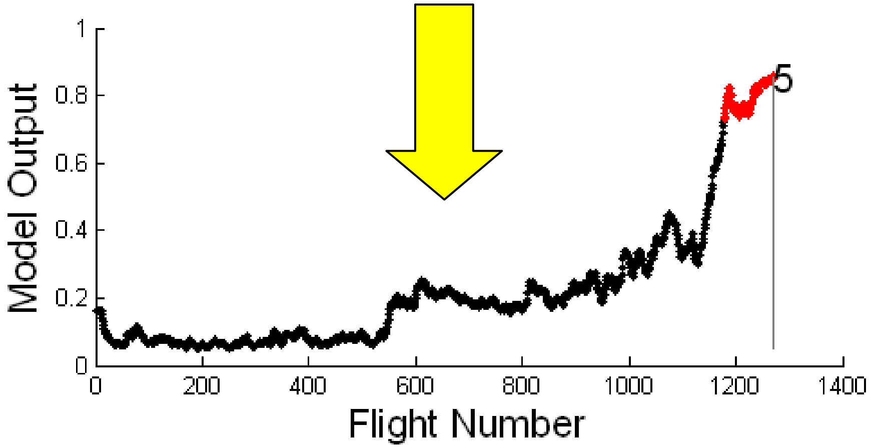
- Several key tricks for maximizing performance:
  - Feature extraction
  - Feature selection
  - Data fusion
  - Classifier fusion

Base Classification System

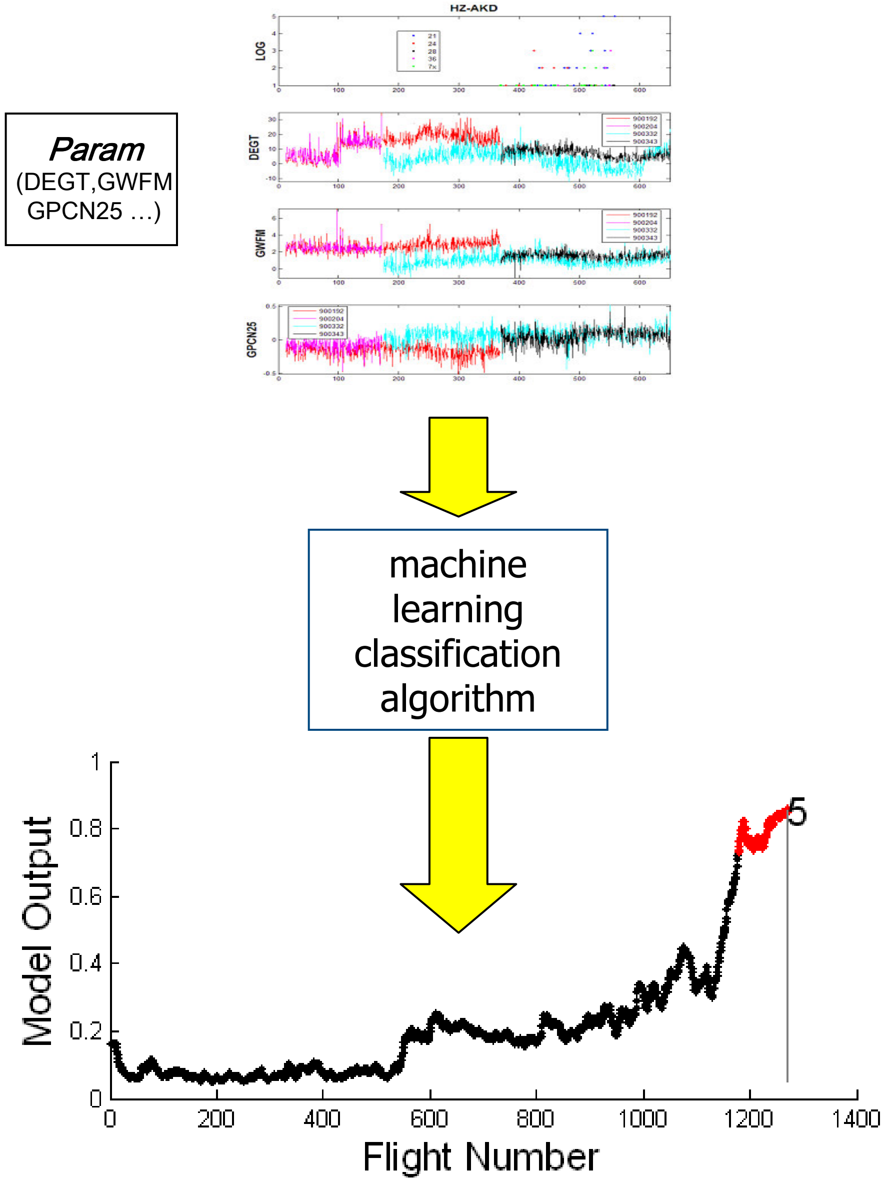
Param
(DEGT,GWFM
GPCN25 …)



machine
learning
classification
algorithm



Base Classification System



Param
(DEGT,GWFM
GPCN25 …)

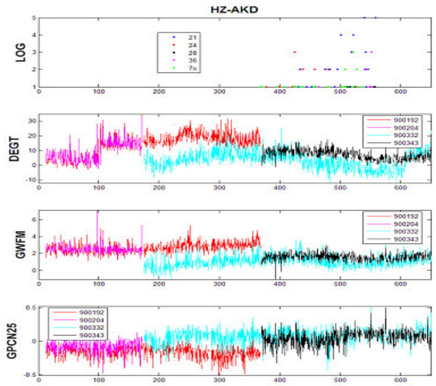
machine learning classification algorithm

The performance of this sucks! Why?

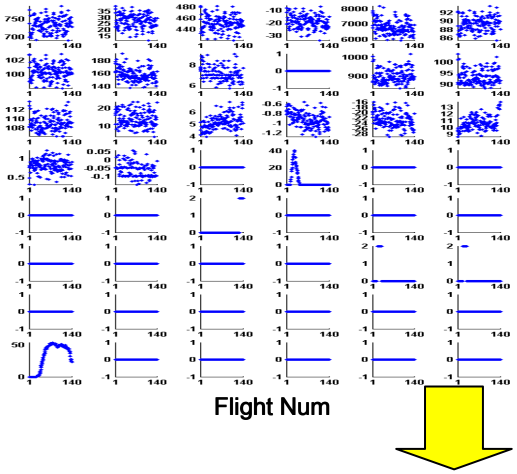
- The raw data space is never (OK, rarely) the best feature space to do classification!
- Extracted features give much better performance
  - Mean, variance, kurtosis, etc.
  - 1st, 2nd deritives
  - Ratios
  - Frequency content
  - Normalization
  - etc.

Classification w/ Feature Extraction

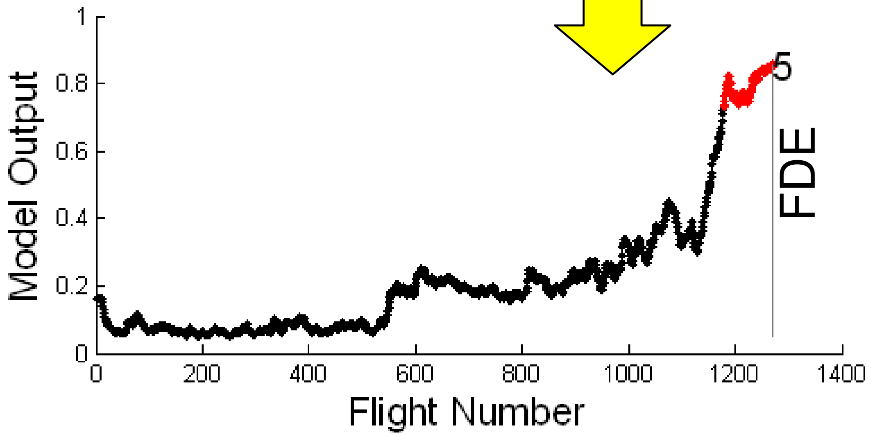
Param
(DEGT,GWFM
GPCN25 …)



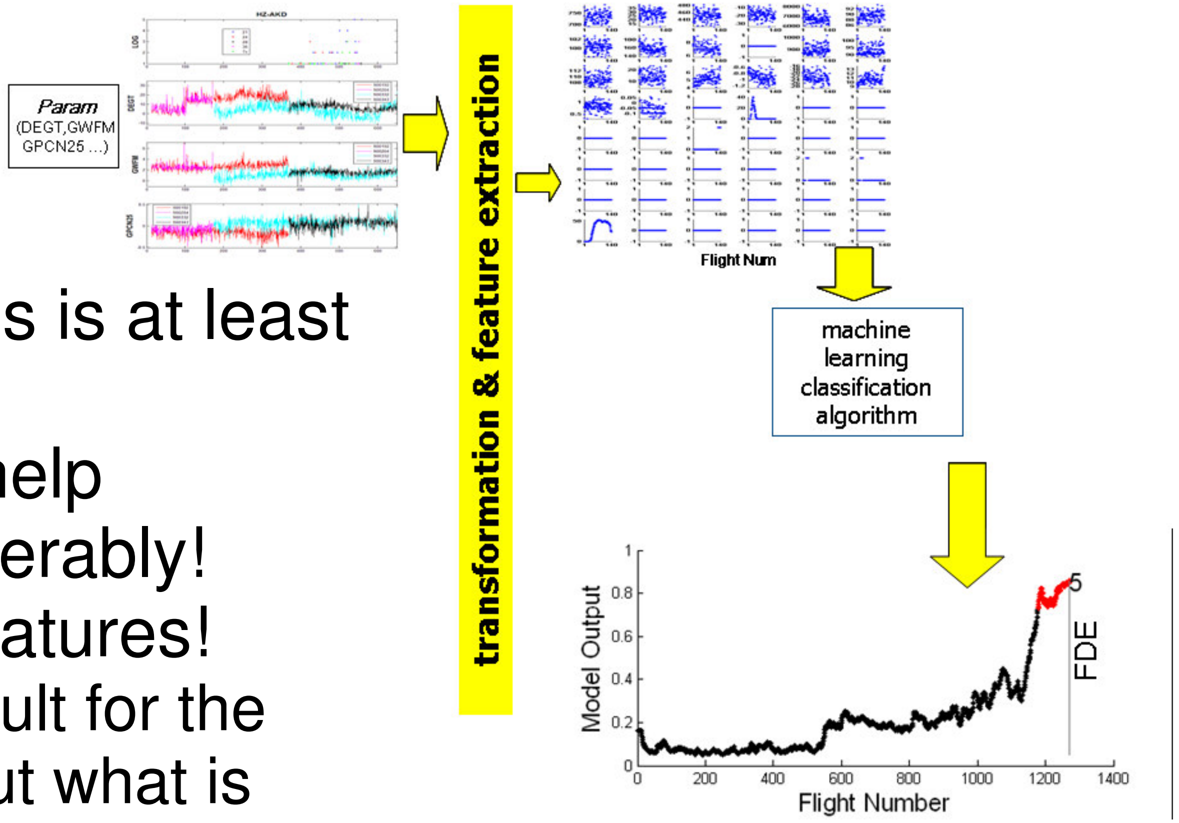
transformation & feature extraction



machine
learning
classification
algorithm



Classification w/ Feature Extraction



Param (DEGT,GWFM GPCN25 ...)

transformation & feature extraction

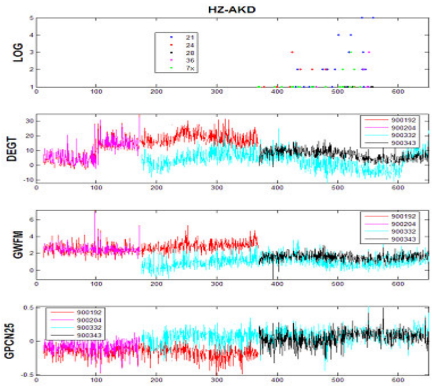
machine learning classification algorithm

The performance of this is at least tolerable. Why?

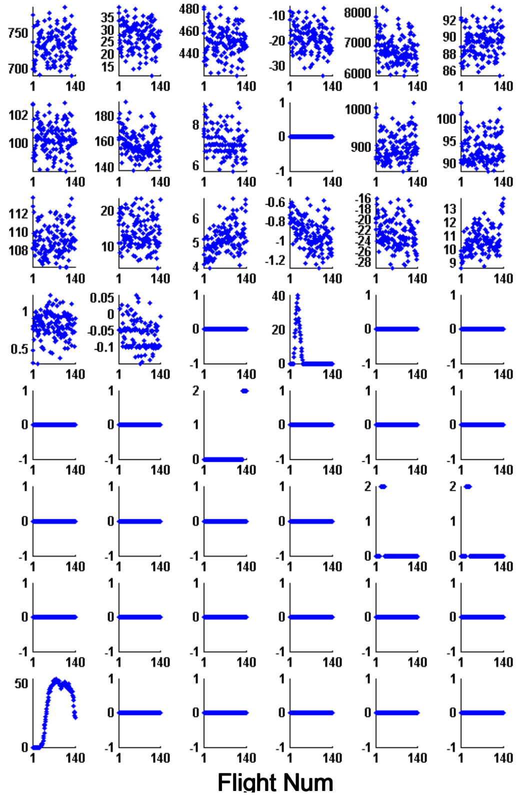
- Extracted features help performance considerably!
- But look at all the features!
  - They make it difficult for the classifier to sort out what is important
  - Moreover, they “trick” the classifier in to classifying things spuriously

Classification w/ Feature Extraction & Selection

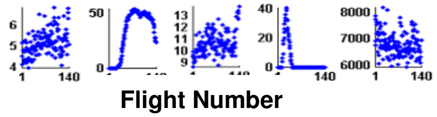
Param
(DEGT,GWFM
GPCN25 …)



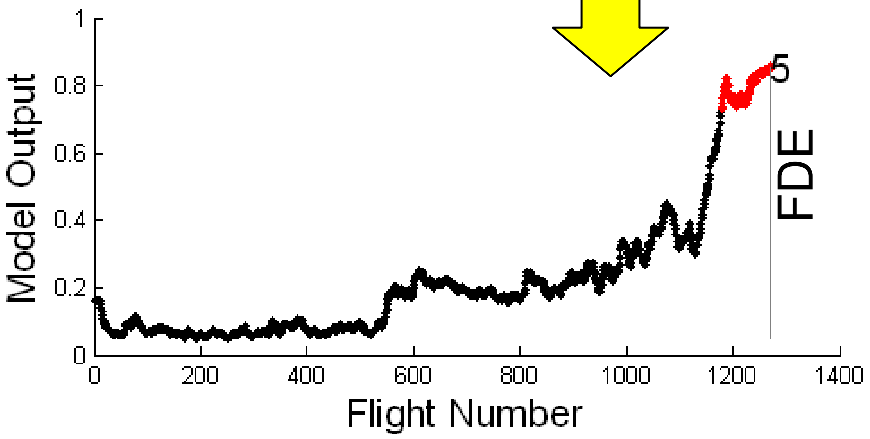
transformation & feature extraction



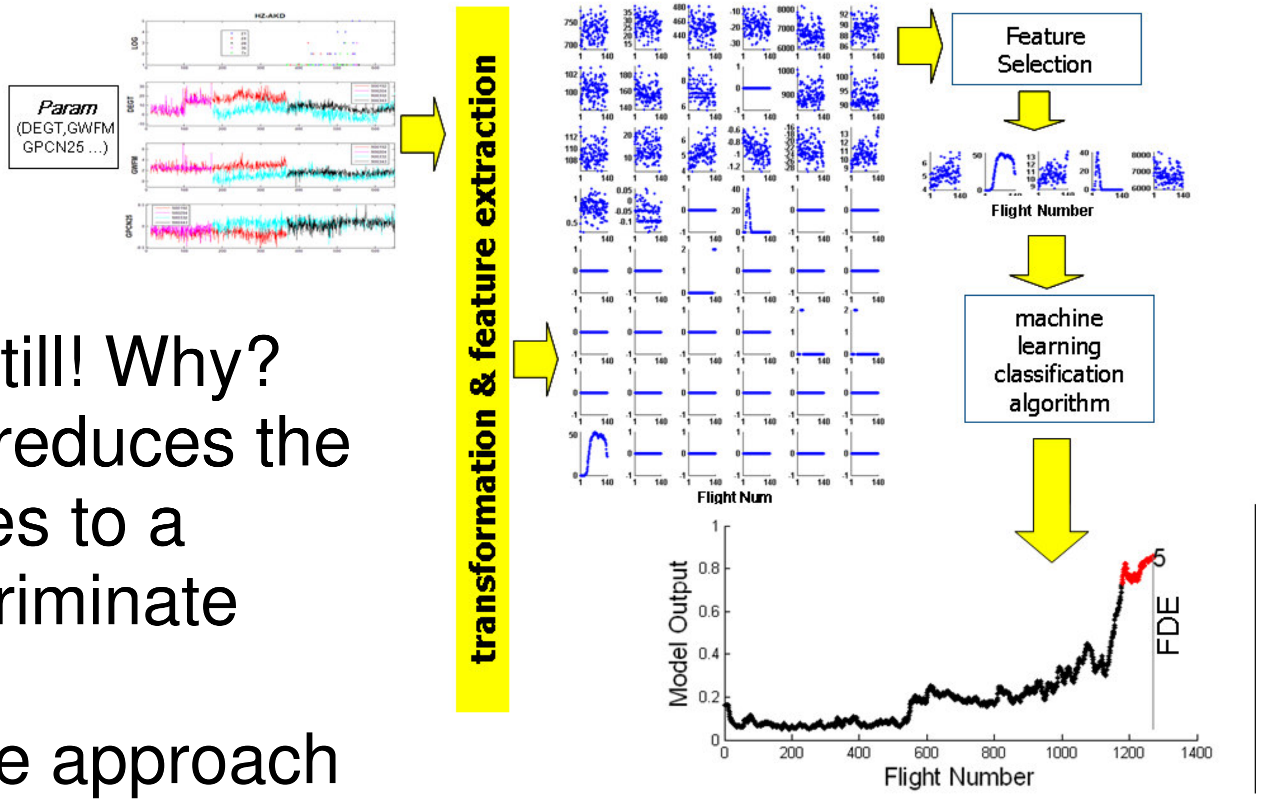
Feature Selection



machine learning classification algorithm



Classification w/ Feature Extraction & Selection



Better performance still! Why?

- Feature selection reduces the large set of features to a smaller set of discriminate features.
- This is the baseline approach that should be used to at a minimum.

Feature Selection

How is feature selection done? Many ways…

- By hand – what seems relevant
- Statistical approaches like forward/backward deletion
- Evolutionary algorithms
- Random forests!
  - If you didn’t think they were awesome enough already, they also have a built-in variable importance measure!

RF variable importance

margin of a case is the proportion of votes for the true class minus the maximum proportion of votes for the other classes

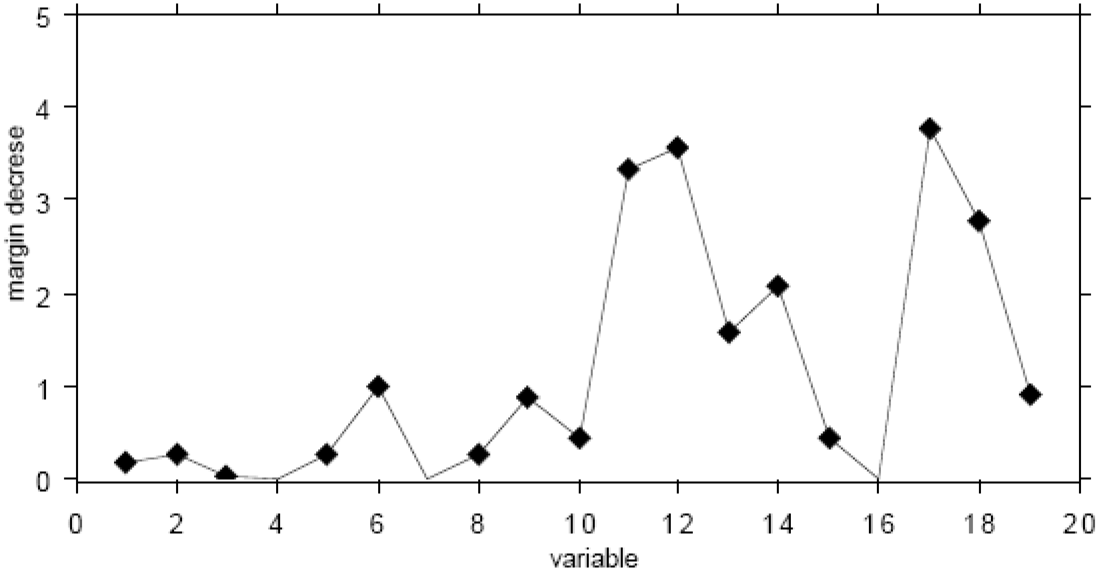
- the larger the margin, the higher the confidence of classification

margin allows definition of variable importance
to estimate the importance of the mth variable:

- take the OOB cases for the kth tree, assume that we already know the margin for those cases M_0
- randomly permute all values of the variable m
- apply the kth tree to the OOB cases with the permuted values
- compute the new margin M
- compute the difference M_0-M

variable importance is defined as the average lowering of the margin across all OOB cases and all trees in the RF

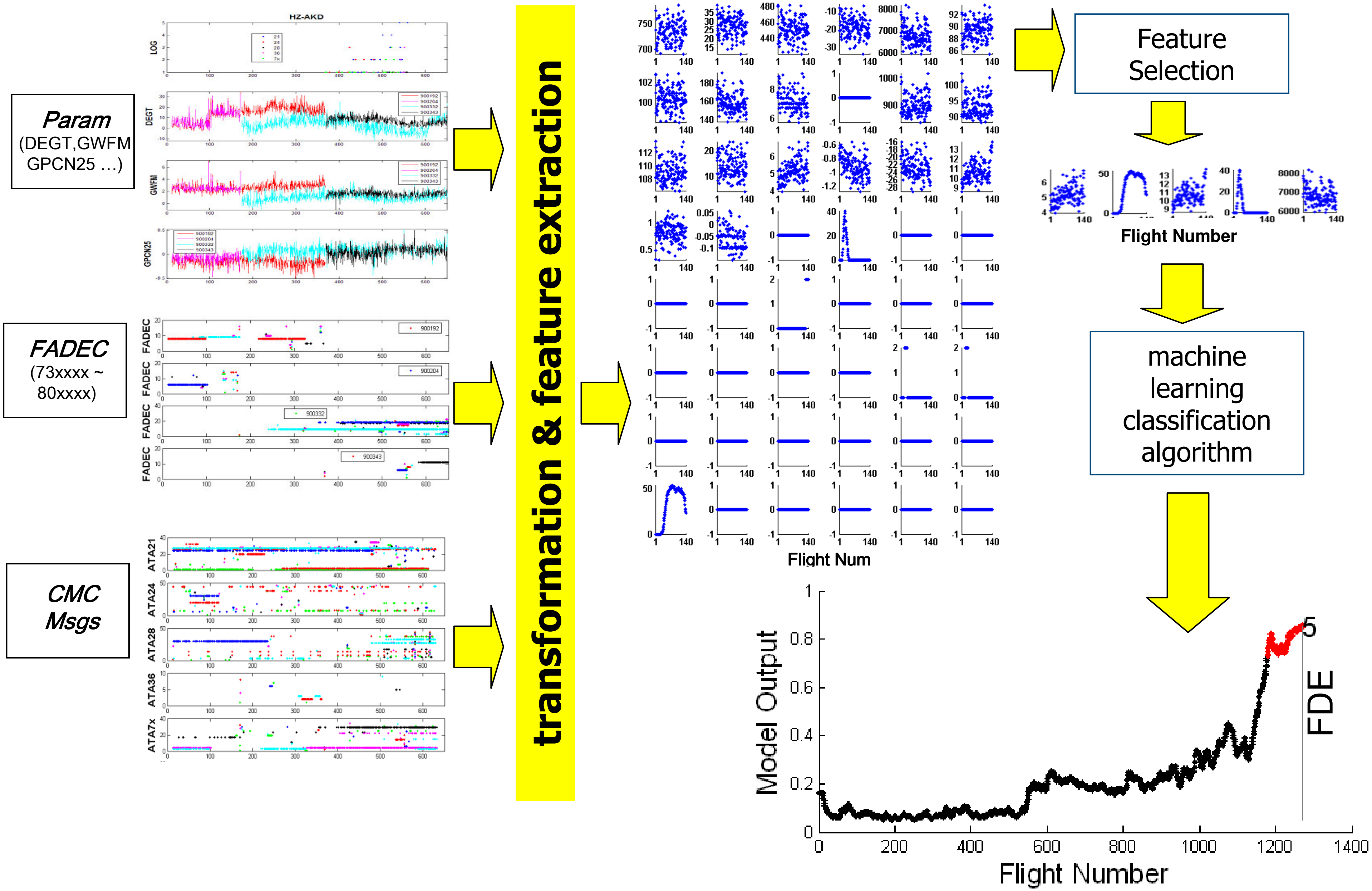
RF feature selection



variable importance can be used for dimensionality reduction

- smaller feature sets → more accurate classifiers (particularly non-RF)
- highly correlated variables “split” variable importance
  - good practice to drop one variable at a time, then recalculate margin
  - “backward deletion”

Data Fusion w/ Feature Extraction & Selection



Param
(DEGT,GWFM
GPCN25 …)

FADEC
(73xxxx ~
80xxxx)

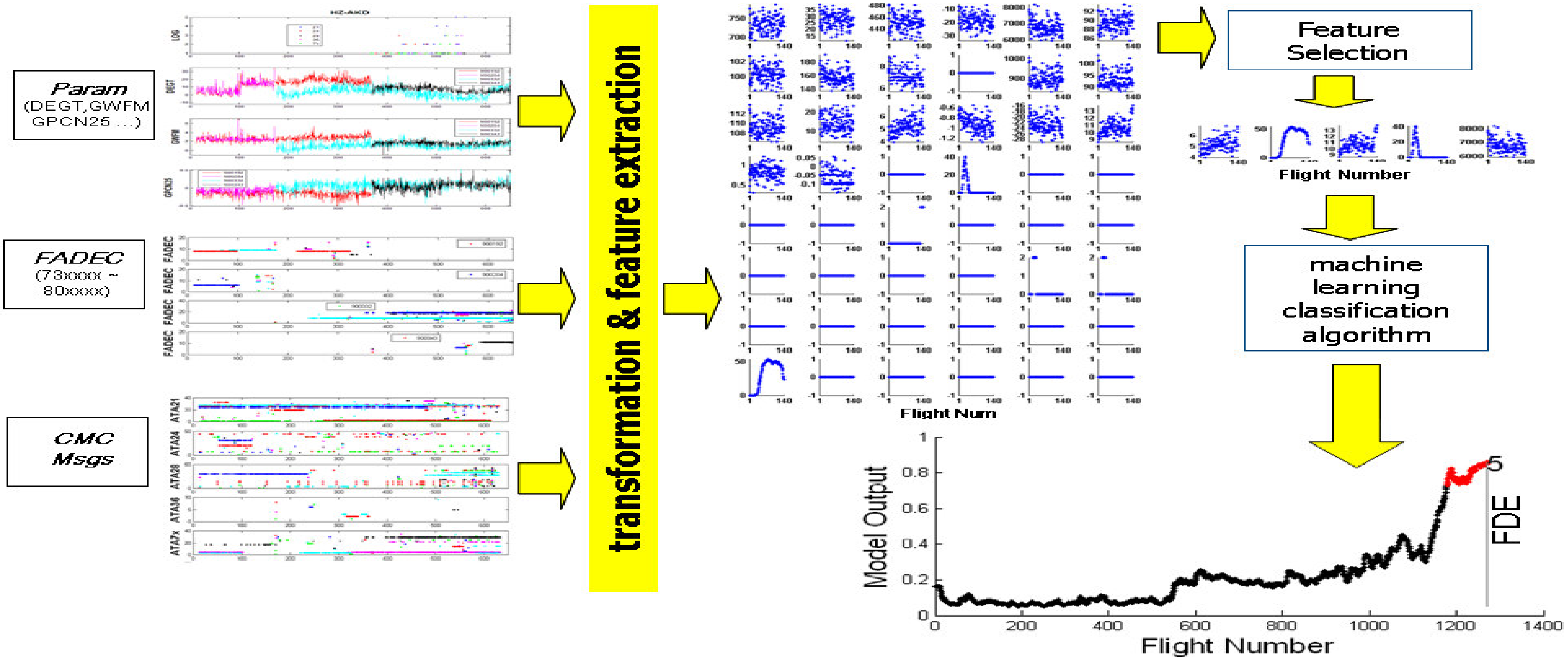
CMC
Msgs

transformation & feature extraction

Feature
Selection

machine
learning
classification
algorithm

Data Fusion w/ Feature Extraction & Selection



Param (DEGT,GWFM GPCN25 ...)

FADEC (73xxxx ~ 80xxxx)

CMC Msgs

transformation & feature extraction

Feature Selection

machine learning classification algorithm

Much more sophisticated!

- Fusion of parametric and nonparametric data allows (where there is rich nonparametric data available) *much* improved performance

Transforming nonparametric to parametric data

Some mixed P/NP systems are tough!

- **Relevant nonparametric data may only occur when alarms are tripped. They are not present for all missions, which makes it difficult to encode patterns to be combined with parametric data.**
- **Parametric data usually retain a change in their characteristics once a fault has occurred given that the measurements are taken under similar operating conditions.**

Temporal Persistence

- **Error messages may only occur in response to a change, even though the changed condition persists.**
- **We want to add an element of temporal persistence to the nonparametric information; e.g., to "remember" that the error has occurred some time in the recent past:**
  - **If the error message was a false alarm, ideally it would not be kept around for long.**
  - **If a message occurs repeatedly, we need to capture the characteristics of that repetition as well.**
  - **The influence of recent faults should be greater than the influence of faults that occurred long ago.**

Transforming non-parametric to parametric data

Decaying

- The time since message occurrence is input to a function that – over time – decreases the influence of the variable. The output diminish the influence of the message occurrence as a function of time. As the frequency of the message occurrence increases, their transformed value is higher.

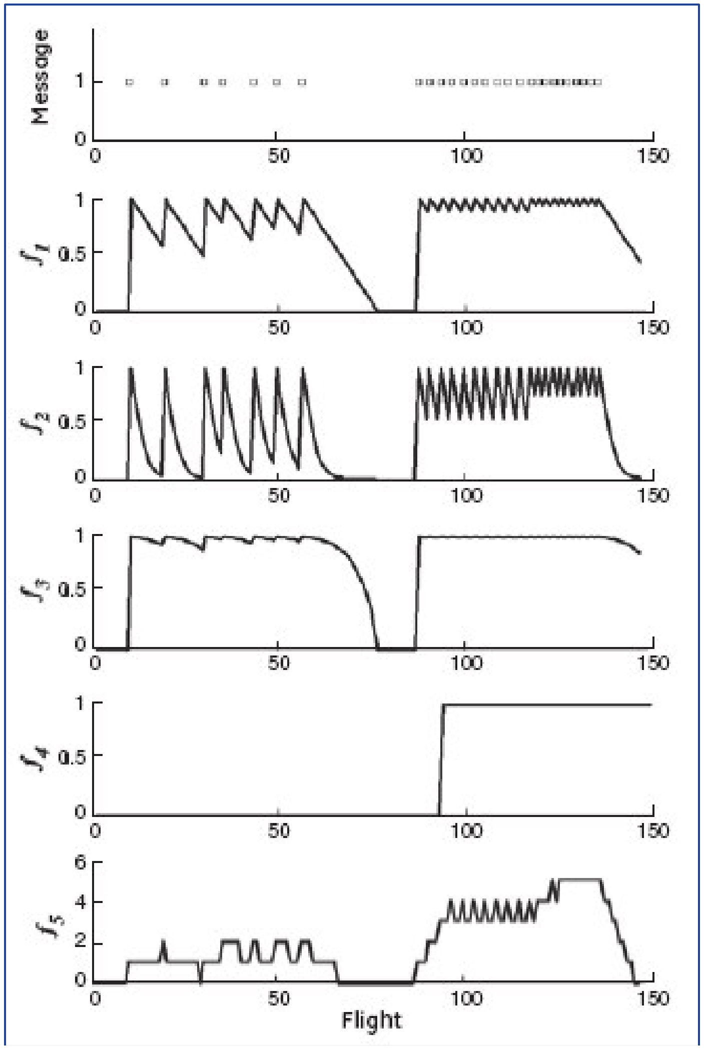
$$f_1(t) = a(t - t_0)^{\kappa} + b$$

$$f_2(t) = \frac{a}{1 + me^{\kappa(t-t_0)}} + b$$

$$f_3(t) = ae^{\kappa(t-t_0)} + b$$

$$f_4(t) = \begin{cases} 1, \sum_{t_0-t_e}^{t_0} \text{messages} > M, t = t_0, t_0+1, \ldots, t_0+t_f \\ 0, \text{otherwise} \end{cases}$$

$$f_5(t) = \sum_{t-t_e}^{t} \text{messages}$$

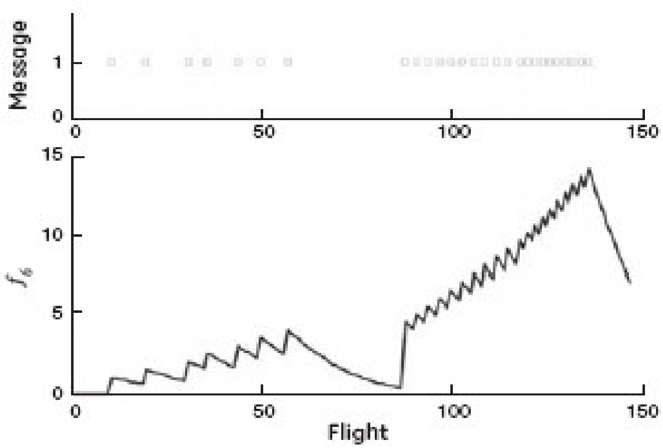


Transforming non-parametric to parametric data

Cumulative Index

- **The value of the decayed message is added to the newly occurring message, resulting in an increasing value for increased frequency**

$$f_6(t) = \frac{a}{1 + me^{n(t-t_0)}}$$

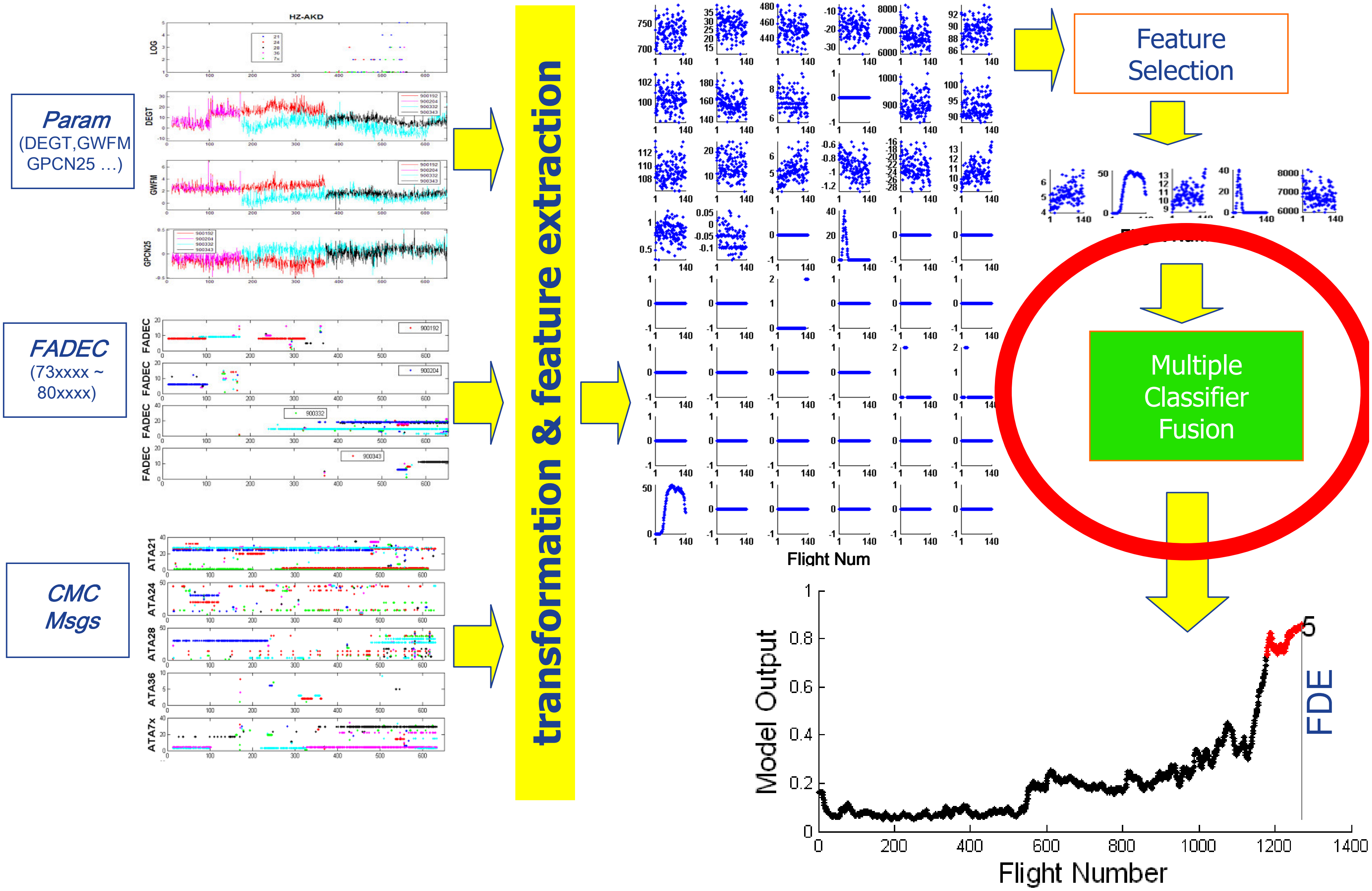


For the example:
m=1, *a*=2, *n*=0.1.

Tuning Parameters

- **Different decaying functions with different decaying time constants have different impact on conversion, affecting AD performance.**
- **Need GA or similar to tune parameters**

State-of-the-art Diagnostic System



Param
(DEGT,GWFM
GPCN25 …)

FADEC
(73xxxx ~
80xxxx)

CMC
Msgs

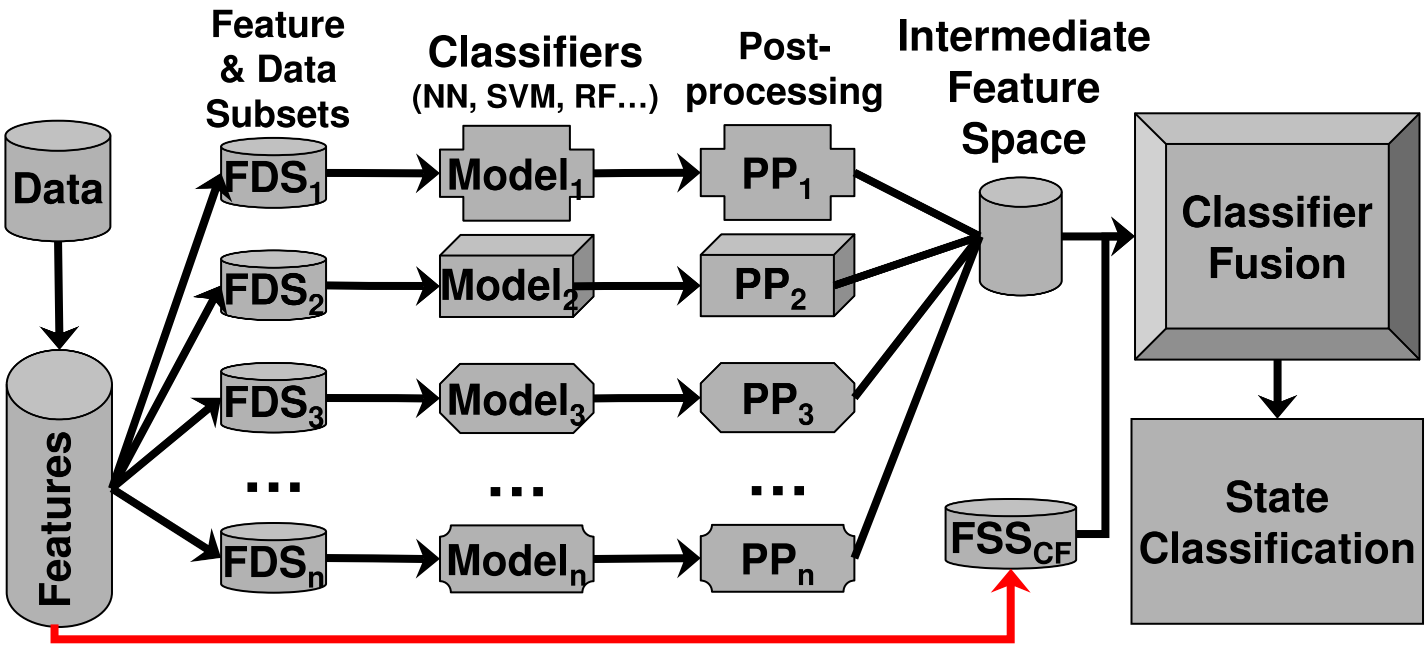
transformation & feature extraction

Feature
Selection

Multiple
Classifier
Fusion

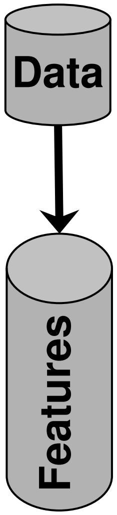
State-of-the-art Diagnostic System

- Same system! Just a more elegant figure…



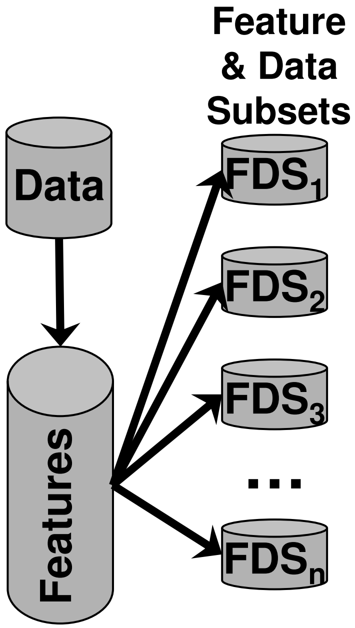
State-of-the-art Diagnostic System

- As before, many features are extracted from the sensed data.



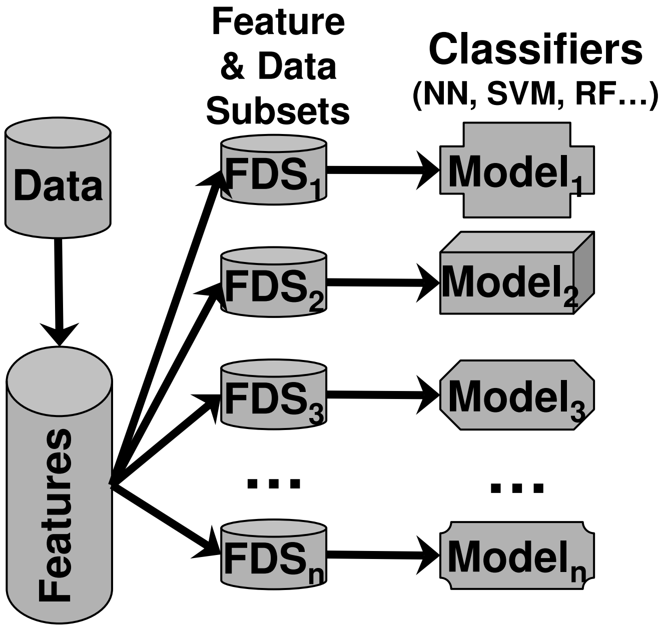
State-of-the-art Diagnostic System

- As before, many features are extracted from the sensed data.
- Partitioning the data into many not necessarily orthogonal feature subsets…



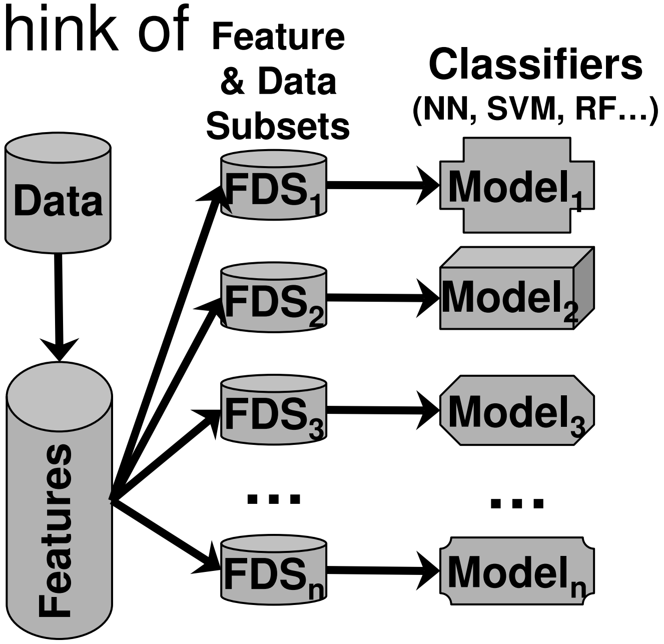
State-of-the-art Diagnostic System

- As before, many features are extracted from the sensed data.
- Partitioning the data into many not necessarily orthogonal feature subsets…
- Allows many classifiers to be developed



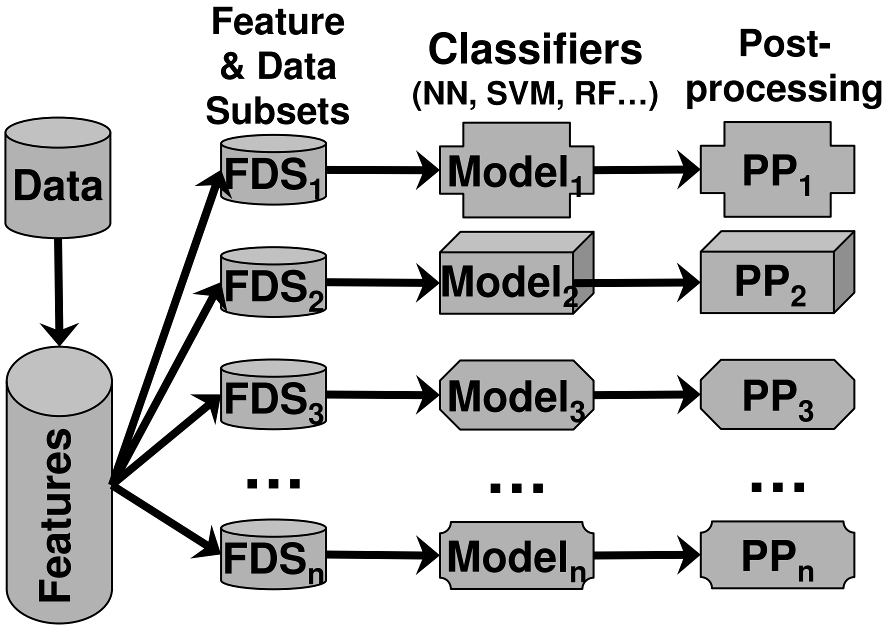
State-of-the-art Diagnostic System

- Allows many classifiers to be developed
- Want the classifiers to be *diverse*
  - If they are all making the *same* mistakes, no reason to have multiple classifiers
  - Diversity is promoted through different data subsets, different training parameters, different underlying classifiers, different regions of the input space – whatever you can think of



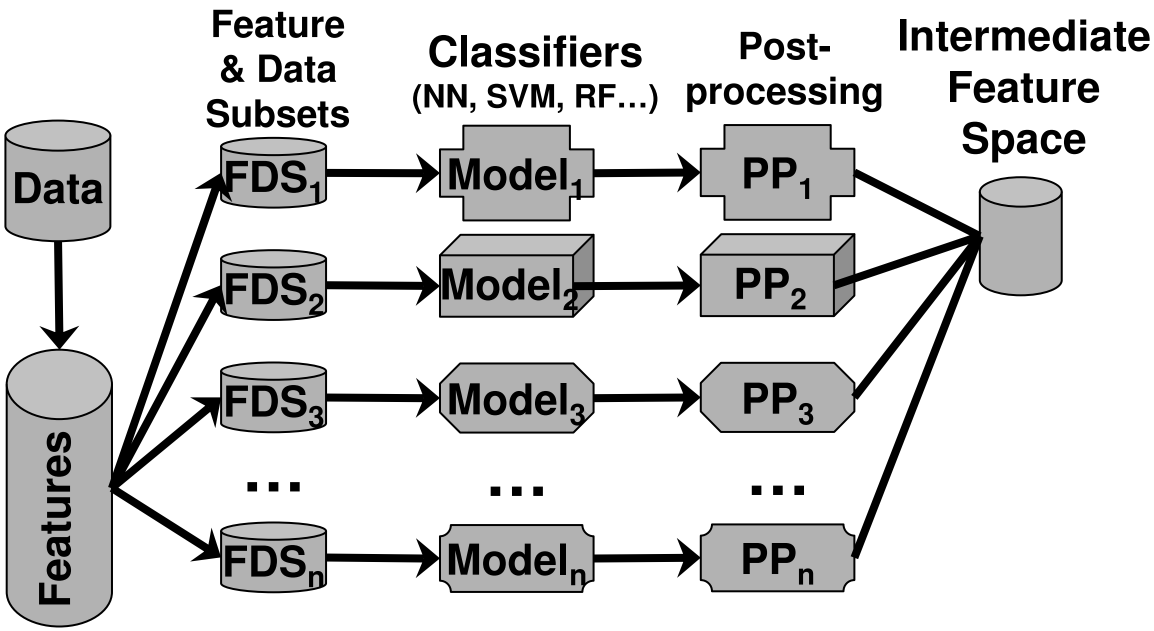
State-of-the-art Diagnostic System

- More or less trivial step
  - Make range common
  - Make direction common (“high is better” – whatever standard)



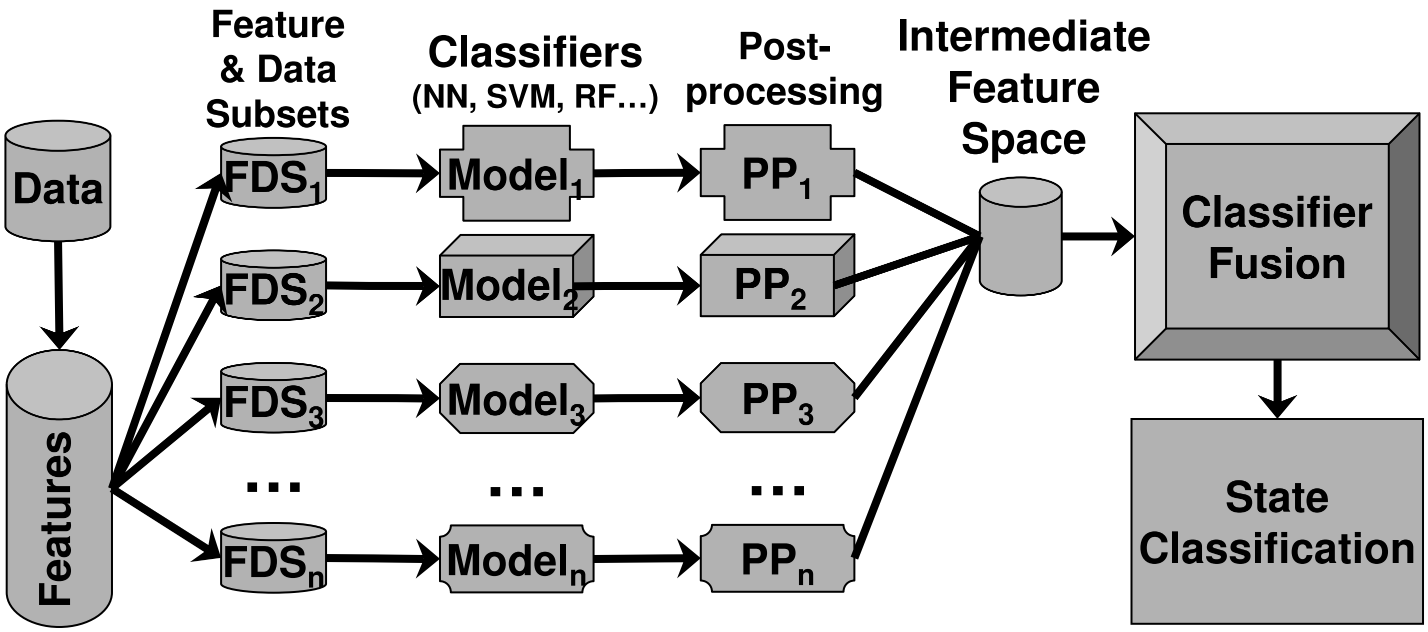
State-of-the-art Diagnostic System

- The output from the classifiers can be treated as an intermediate feature space



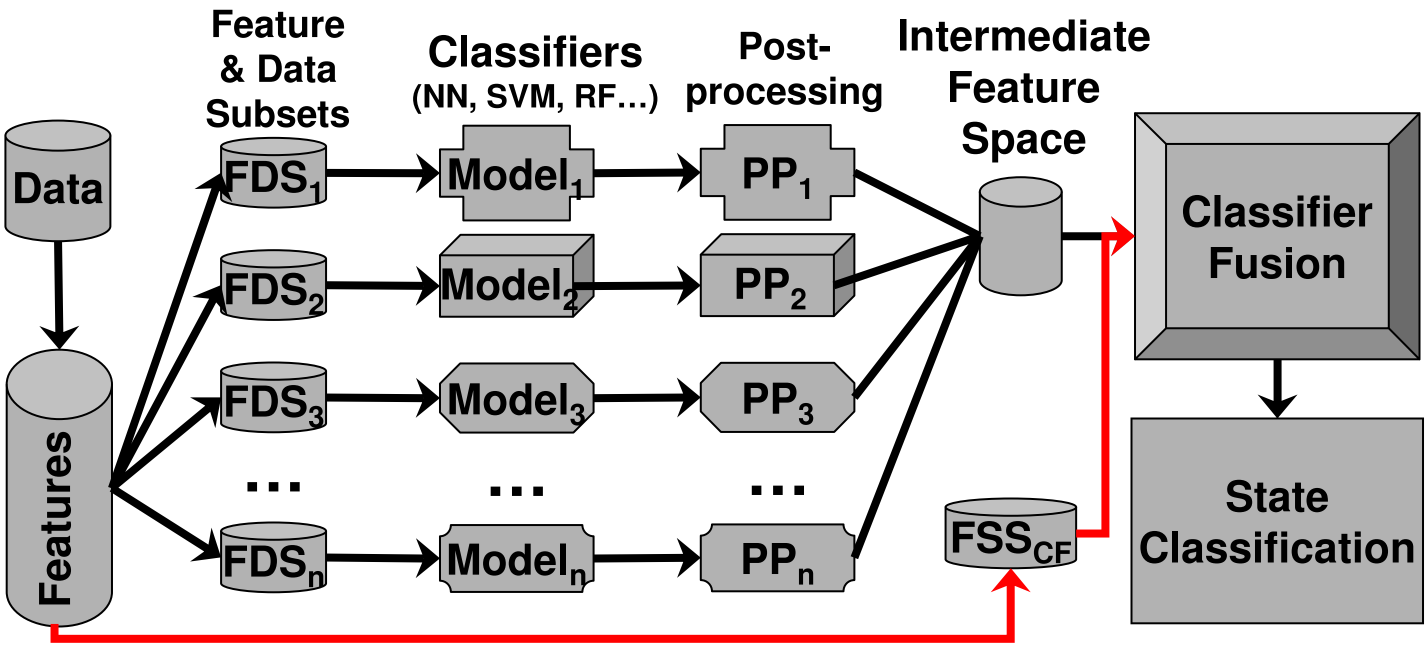
State-of-the-art Diagnostic System

- The output from the classifiers can be treated as an intermediate feature space
- And a meta-classifier can be trained to resolve the output from the classifiers
    - I’ll bet you’ll never guess what I recommend as the meta-classifier!



State-of-the-art Diagnostic System

- Pro tip: Using some of the raw features along with the output from the individual classifiers results in improved performance
  - Note that it helps tremendously to do feature selection on the intermediate feature space and the feature subset for classification



Lagniappe:

Classifier Performance Assessment & Tuning

Signal Detection Theory

- A system (human, guinea pig, computer model, etc.) responds to a stimulus by discriminating (correctly or incorrectly) between signal and noise.

- In the most simple case, there are two possible stimuli (“noise” and “signal plus noise”) and two possible categorical responses.

- After subjecting the system to a number of trials, the categorical responses are matched with the “noise” and “signal plus noise” stimuli to construct a 2×2 contingency table, which is then used to calculate HR and FAR.

- results vary with decision criterion…

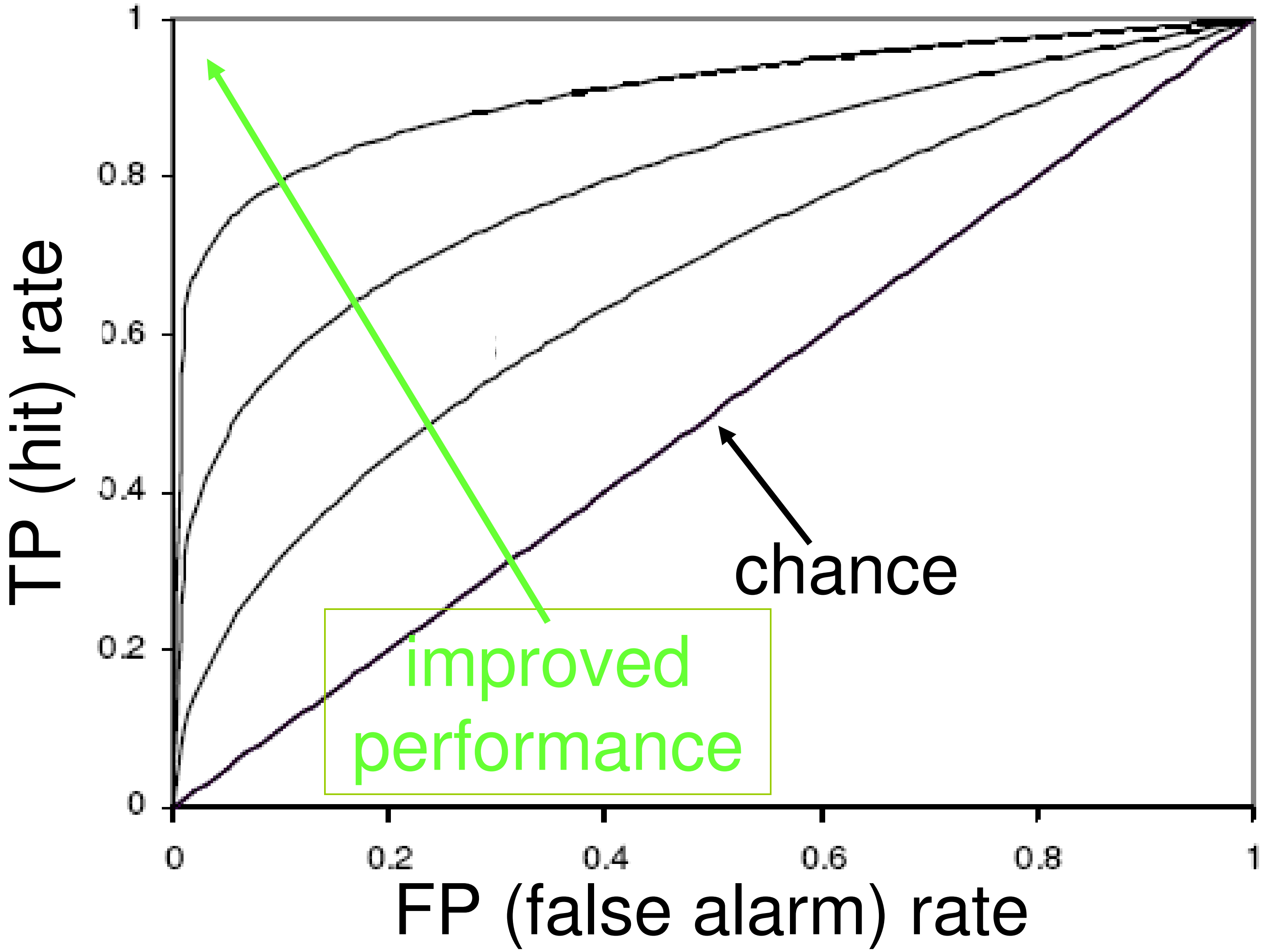
Signal Detection Theory (cont.)

- By changing the decision criterion for a response, we can construct multiple contingency tables and plot a curve of HR, FAR points based on the tables.

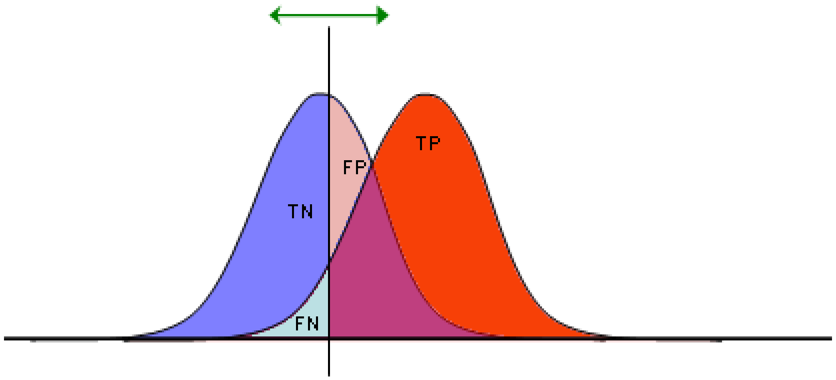
- the curve describes the system's discrimination ability
  - **Receiver Operator Characteristic**
  - **ROC curve**

- **ROC curves can be used to compare multiple classification systems and/or to select optimal decision criterion**

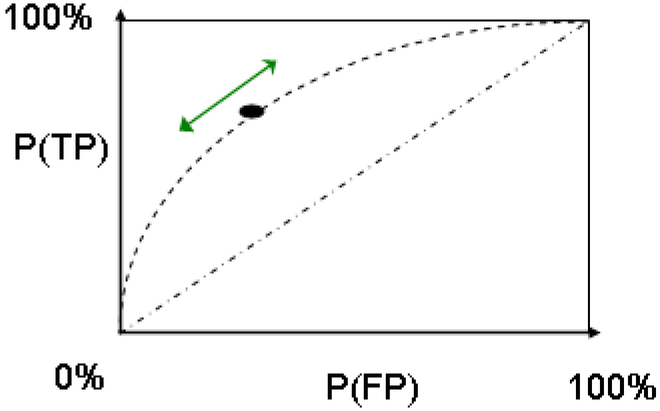
ROC curves


1
0.8
0.6
0.4
0.2
0
0
0.2
0.4
0.6
0.8
1
TP (hit) rate
FP (false alarm) rate
chance
improved performance

Interpreting ROC curves



TP	FP
FN	TN
1	1



creating an ROC curve

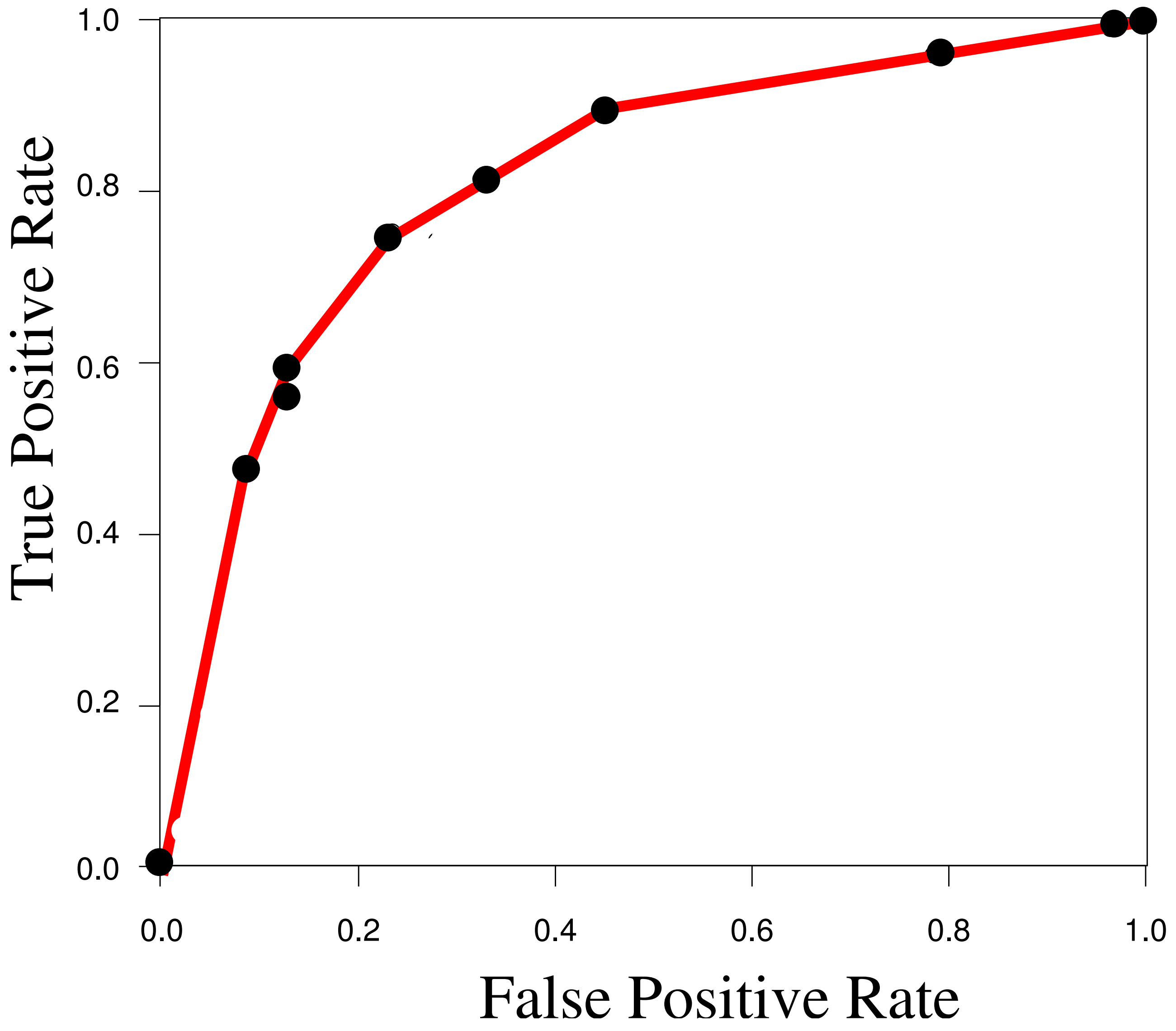
- a classifier produces a single ROC point

- if the classifier has a “sensitivity” (threshold) parameter, varying it produces a series of ROC points (confusion matrices)

- alternatively, if the classifier is produced by a learning algorithm, a series of ROC points can be generated by varying the class ratio in the training set

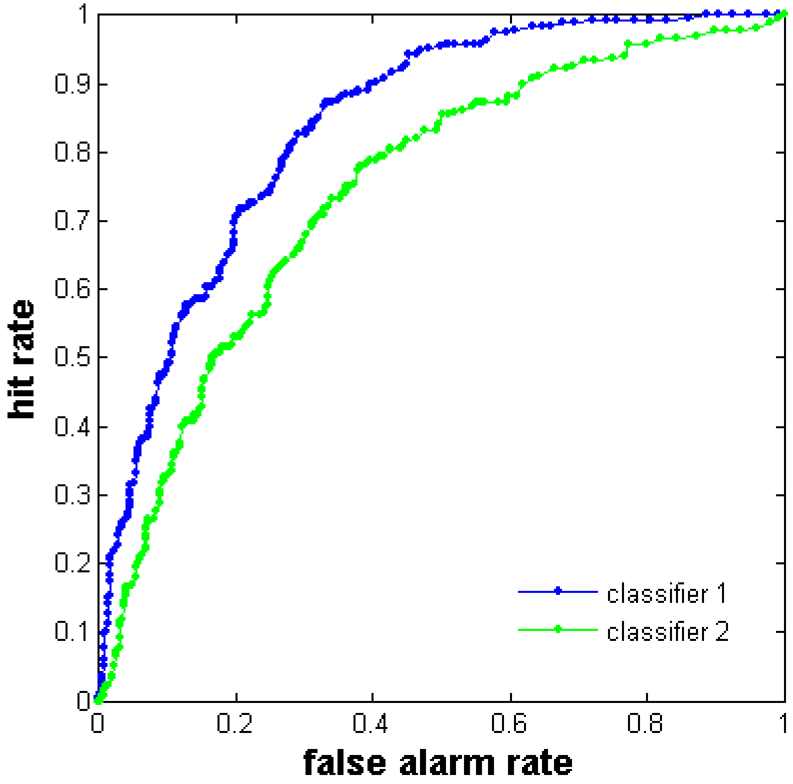
empirical ROC curve

vary the decision threshold, connect the dots



comparing two classifiers

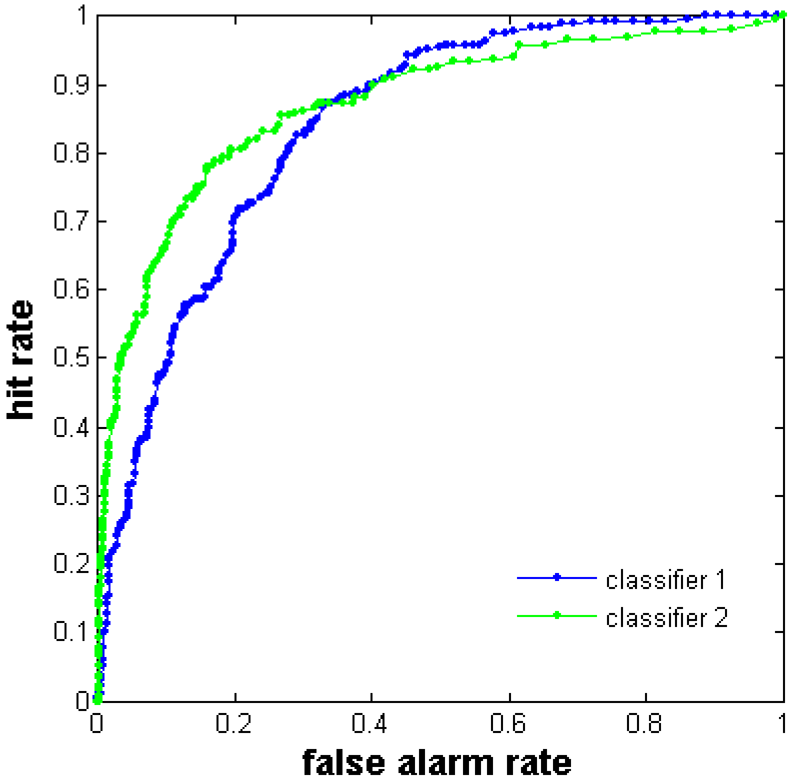
C1>C2



comparing two classifiers

C1 > C2 at high hit rate

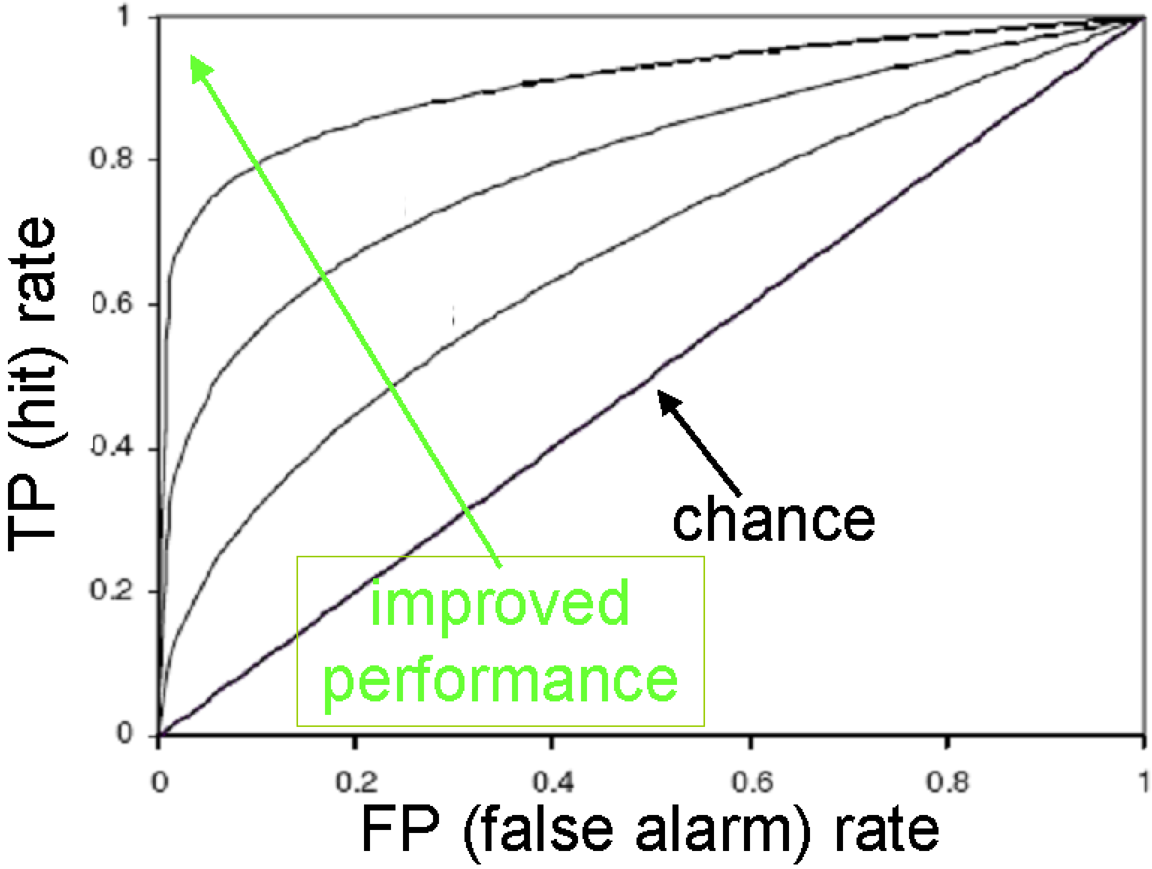
C1 < C2 at lower hit rate



distilling ROC to a scalar…

Still, sometimes the Pointy Haired Boss wants just 1 number to compare classifiers instead of a bunch of plots…

- area under the curve (AUC)
  - range between 0.5 and 1
  - 0.5 sucks
  - 1.0 is perfect
- not perfect, but better than accuracy


TP (hit) rate
FP (false alarm) rate
chance
improved performance