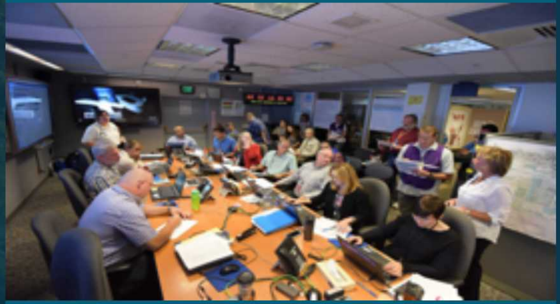
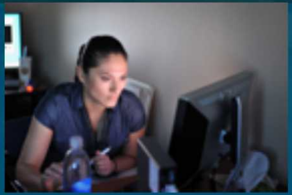
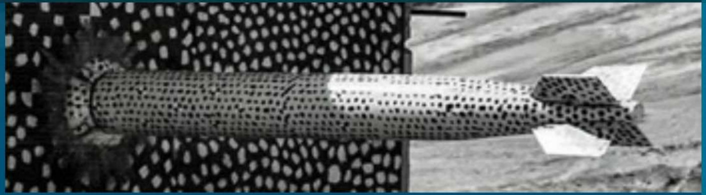
Sandia National Laboratories

SAND2019-10627C

Uncertainty Quantification and Machine Learning Classification of Mechanical Breach in Crash and Burn







PRESENTED BY

Ari Frankel, Quantitative Modeling and Analysis, 8754

alfrank@sandia.gov

Outline

Problem Definition

A Quick Introduction to Binary Classification Models

- Bernoulli events
- Logistic regression
- Quantifying accuracy of classifiers
- Artificial neural networks

Random velocity study

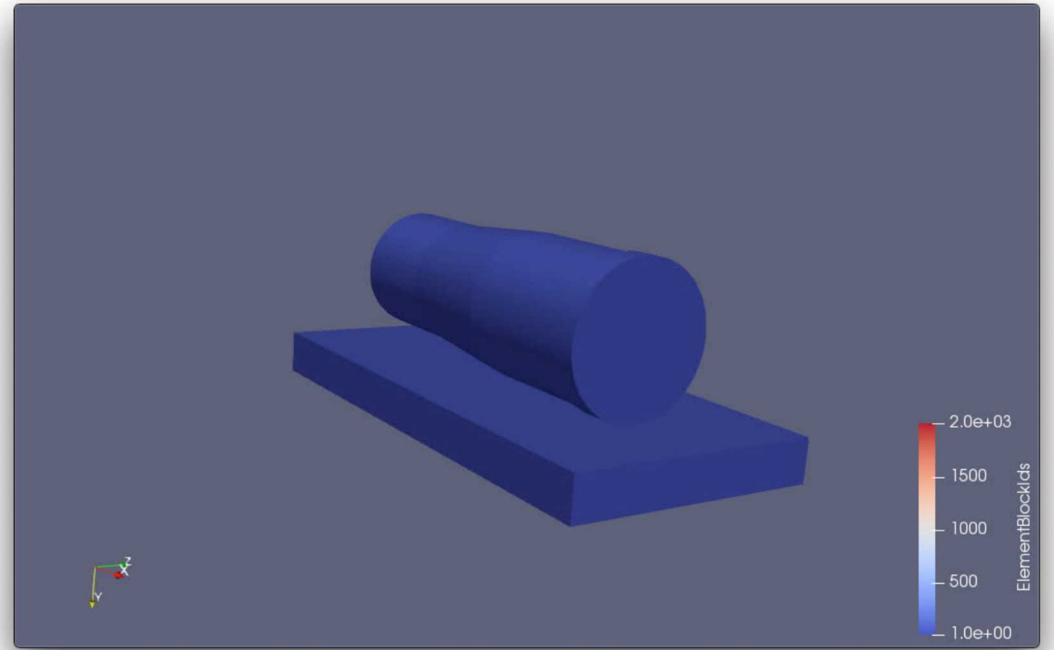
- Accuracy results
- Probability of loss of containment vs. velocity

Fixed velocity study

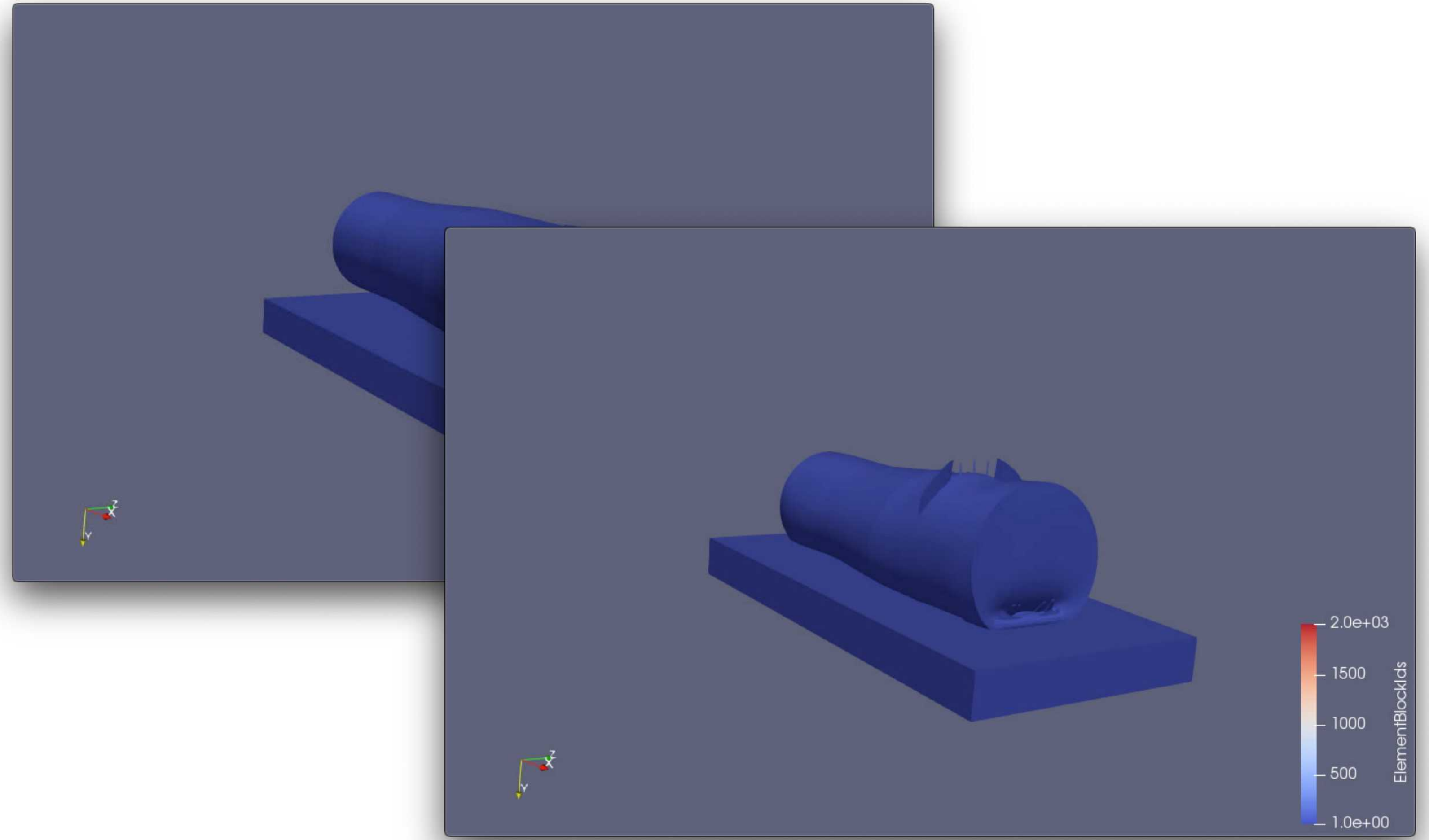
- Accuracy results
- Sensitivity analysis

Future Work

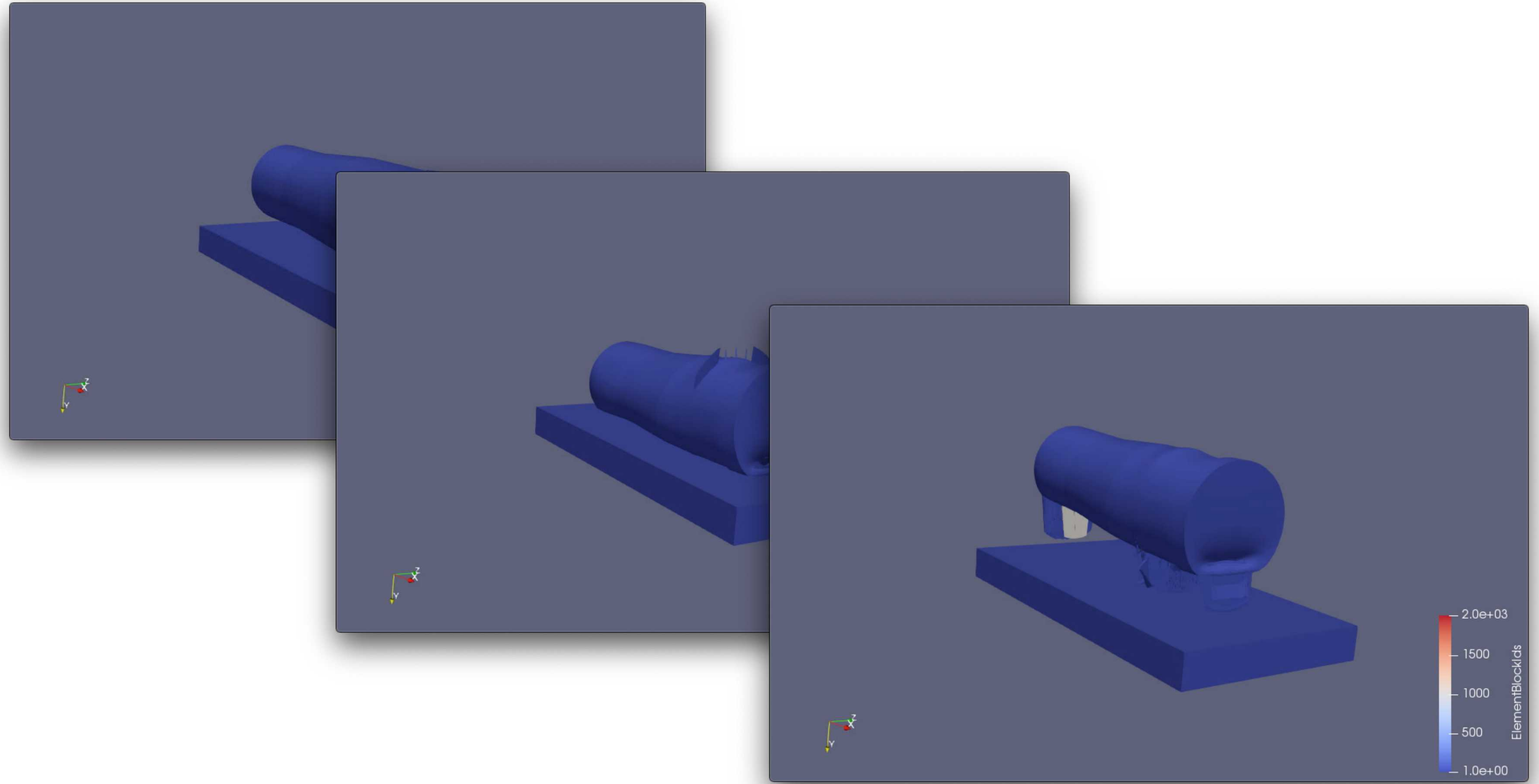
Crash and Burn



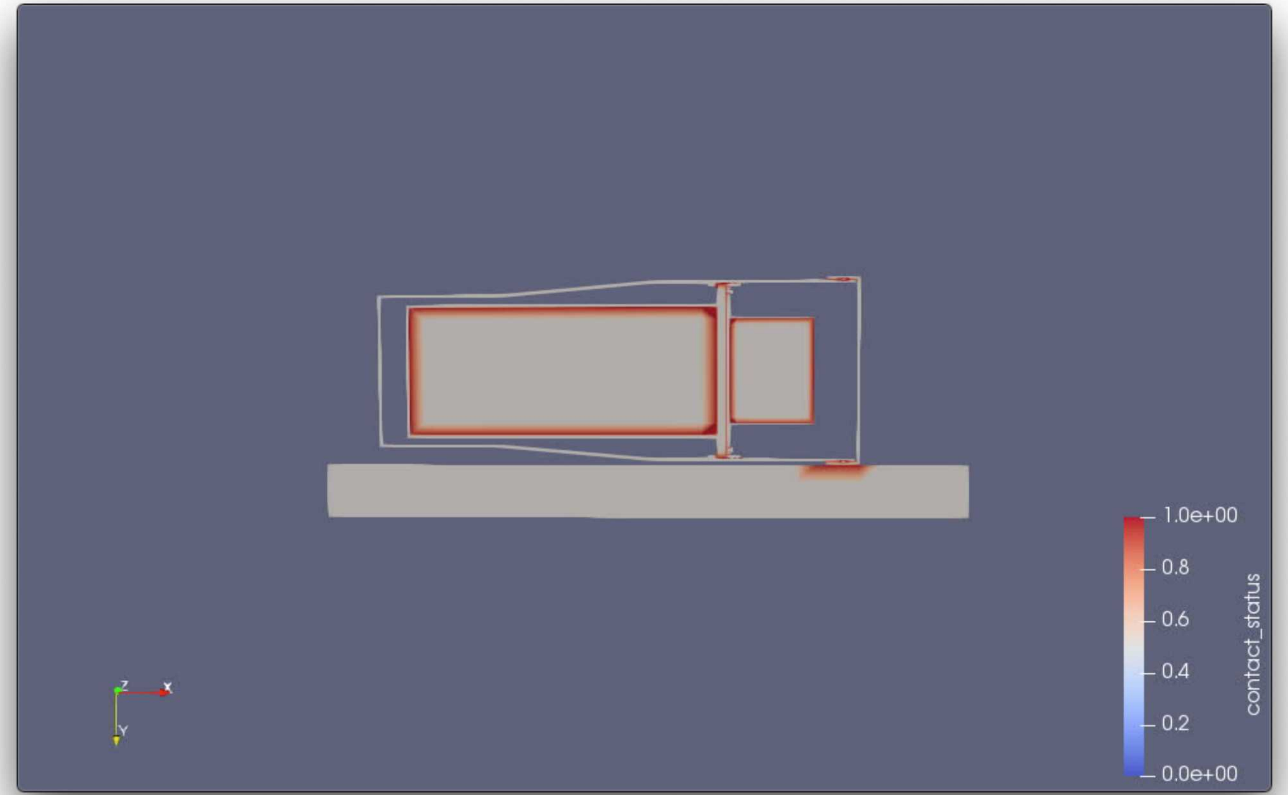
Crash and Burn



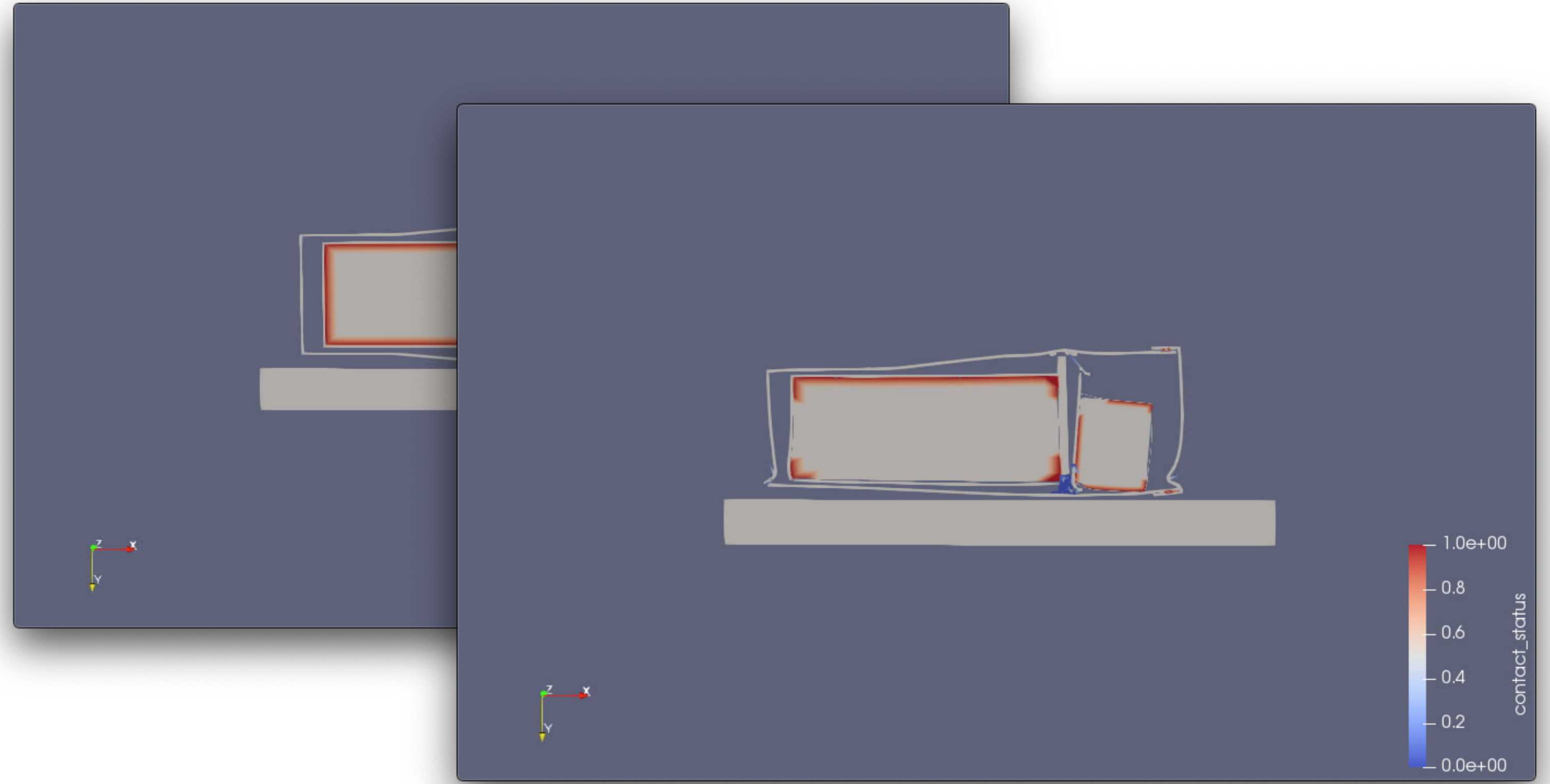
Crash and Burn



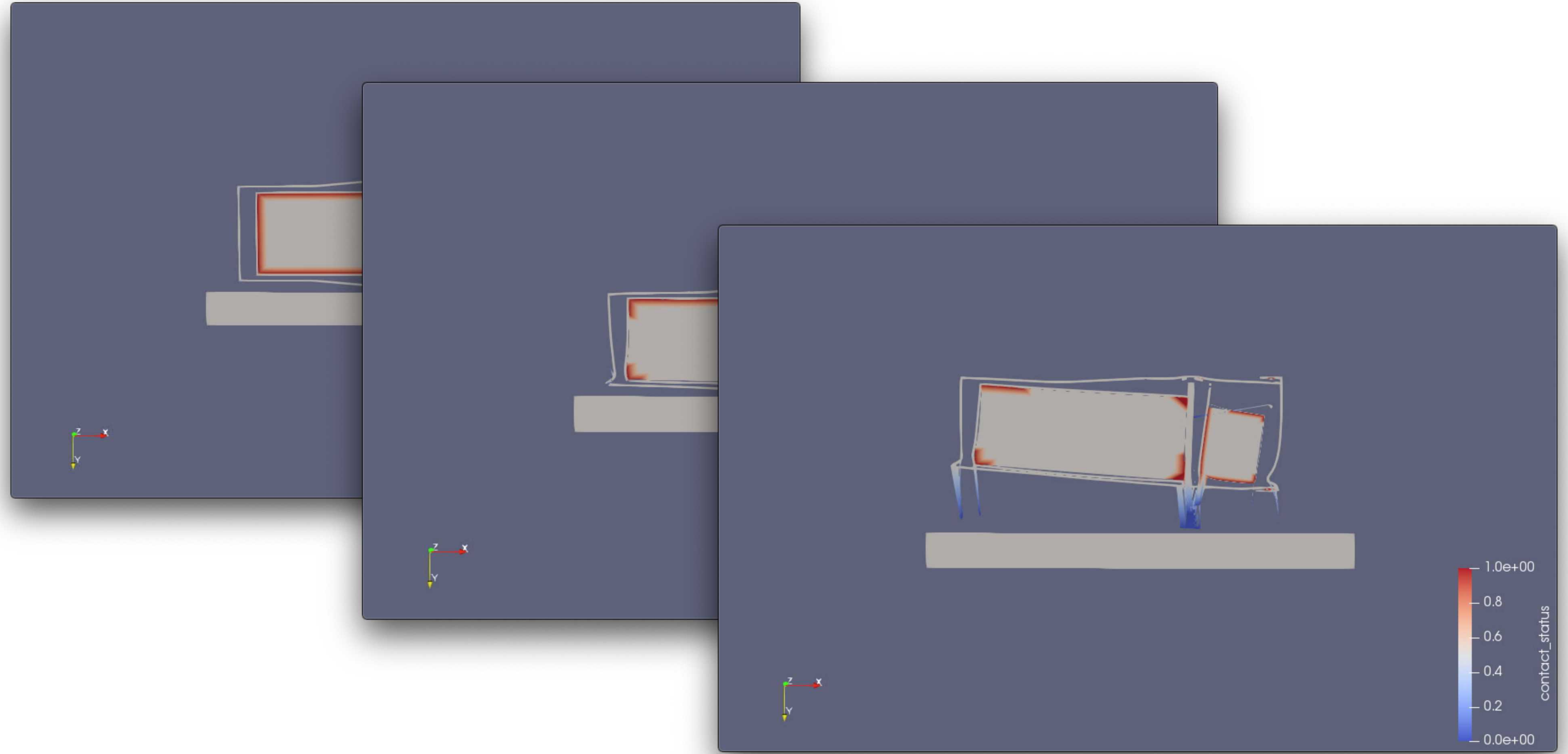
Crash and Burn



Crash and Burn



Crash and Burn



Crash: failure tracking

Track breach in the aft case, forward case, housing, and outer case

0 if intact by end of simulation

1 if breach detected

Given: material properties, impact angle and speed

Predict: 0 or 1

Methods?

Latin Hypercube Sampling?

- As usual, may require millions of samples to compute empirical reliability
- Won't give parameter sensitivity directly (without even more samples)

Polynomial Chaos Expansions?

- Can't fit a 0-1 surface- inherently discontinuous, will converge poorly

Reliability methods?

- Require a continuous function to evaluate- no single MPP in this problem
- GP surrogate will also converge poorly- no global reliability here

Classification models (probabilistic)

Want probability of failure/success given impact speed and material parameters

We want a model like PCE, linear regression, etc. we can use as a surrogate

$$m(x, \theta) = \text{probability of failure}$$

X = vector of features (Young's modulus, yield stress…)

Theta = model parameters

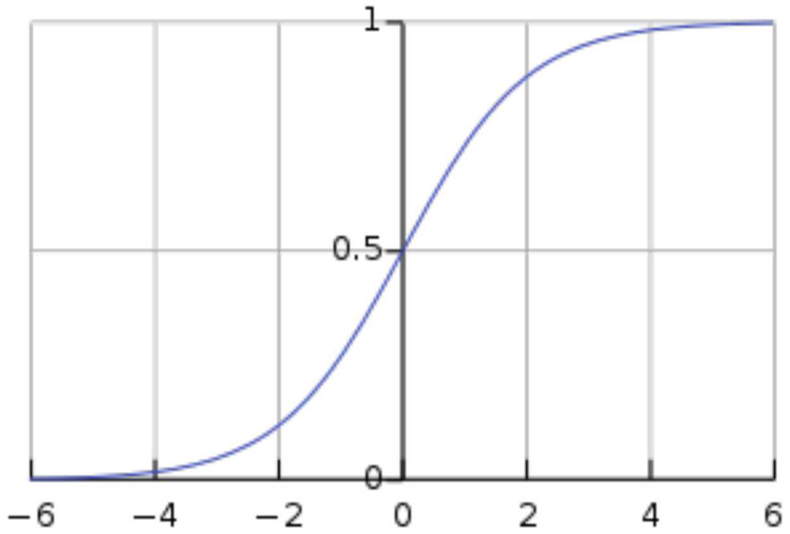
Bernoulli random variable:

$$p(y|x, \theta) = m(x, \theta)^y (1 - m(x, \theta))^{1-y}$$

$$p(y = 1|x, \theta) = m(x, \theta) \qquad p(y = 0|x, \theta) = 1 - m(x, \theta)$$

Logistic regression

$$m(x, \theta) = \frac{1}{1 + \exp(-\theta^T x)}$$



From wikipedia

Calibrating the model: maximum likelihood
N samples of data (x,y), x = features, y = {0,1}

Max $$L = \prod_{i=1}^{N} m(x_i, \theta)^{y_i} (1 - m(x_i, \theta))^{1-y_i}$$

Max $$l = \sum_{i=1}^{N} y_i \log m(x_i, \theta) + (1 - y_i) \log(1 - m(x_i, \theta))$$

Gaussian -> Minimize sum-of-squares

Bernoulli -> Maximize “cross-entropy”

Logistic regression is convex -> easy to calibrate

Logistic regression

Decision problem

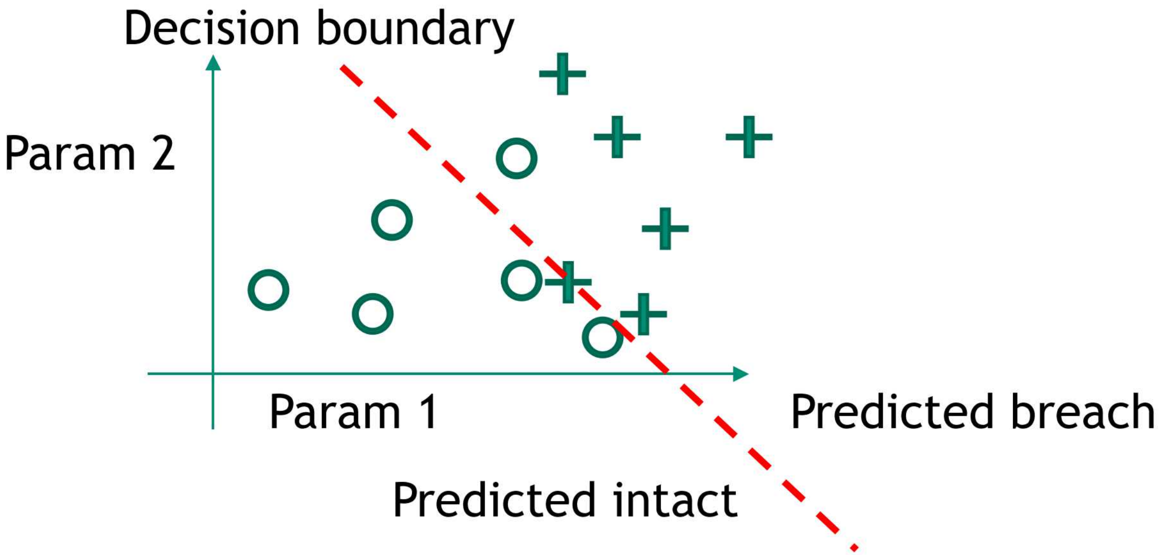
- If output is greater than 50%, we call that failure

Alternatively:

- We can pick threshold to avoid false negatives to be conservative
- E.g. if you had 10% chance of having serious illness, you'd want to get it checked out

The decision boundary at probability p is a hyperplane of form:

$$\theta^T x = \log\left(\frac{p}{1-p}\right)$$



Accuracy metrics: testing the model out

Confusion Matrix	Data says it breaches	Data says it stays intact
We predict it breaches	True positive (TP)	False positive (FP)
We predict it stays intact	False negative (FN)	True negative (TN)

Accuracy = (TP+TN)/(TP+TN+FP+FN)
How well model does overall

False negative rate = FN/(FN + TP)
Risk of falsely saying it'll pass

False positive rate = FP/(FP + TN)
Risk of falsely saying it'll breach

We may be okay with with a high FPR but want an FNR as small as possible

Logistic regression parameters

Easy to interpret!

- Big parameter values = more important to decision
- Small = little influence

Looking for sparse models

- Can add ”LASSO” or other compressed sensing models/regularization
  - Called “L1 regularization” in machine learning literature
- Useful if you don’t have a lot of data
- Forces less relevant coefficients to zero

Crash sampling study

Treat velocity and material properties as uniform random variables

For each velocity/material set, sample all angles

200 samples per angle = 1600 samples total

Property	Range
Speed	30-80m/s
Angle	{0,45,90,135,180,225,270,315} degrees
Metals E	2%
Metals yield	7%
Metals Poisson	2%

Note: metals include SS286, SS304, Al, forward mass, steel

Property	Range
Foam density	20%
Foam E	2%
Foam Poisson	2%
Foam strength	7%
Death EQPS	25%
Friction	10%

Logistic classifier

Breach rates observed in data:

- Forward case: 725/1600 failed
- Aft case: 481/1600 failed
- Housing: 423/1600 failed
- Outer case: 1372/1600 failed

LOOCV: For each of 1600 samples:

- Hold one sample out
- Train logistic classifier on the 1599 samples (scikit-learn)
- Predict the outcome of the remaining sample (If probability>50% => FAIL)
- Compare to true outcome

Logistic classifier

LOGISTIC	Accuracy	False negative rate	False positive rate
Forward case	96.63%	4.55%	2.40%
Aft case	95.69%	7.48%	2.95%
Housing	96.19%	8.51%	2.12%
Outer case	93.31%	2.33%	32.89%

Notes:

Keep in mind that in the outer case, most of the samples were “FAIL”
Hard for classifiers to balance the cases when the data is skewed towards 1 or 0

Breach rates observed in data:

Forward case: 725/1600 failed
Aft case: 481/1600 failed
Housing: 423/1600 failed
Outer case: 1372/1600 failed

Artificial neural network

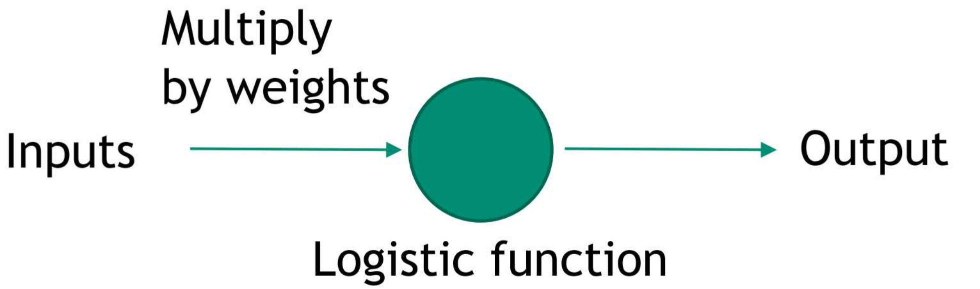
Simple logistic regression:

$$m(x,\theta) = \frac{1}{1+\exp(-\theta^T x)} \qquad \theta^T x = \log\left(\frac{p}{1-p}\right)$$

Like a linear model!

Can we make it more complicated? Capture more complex behavior?

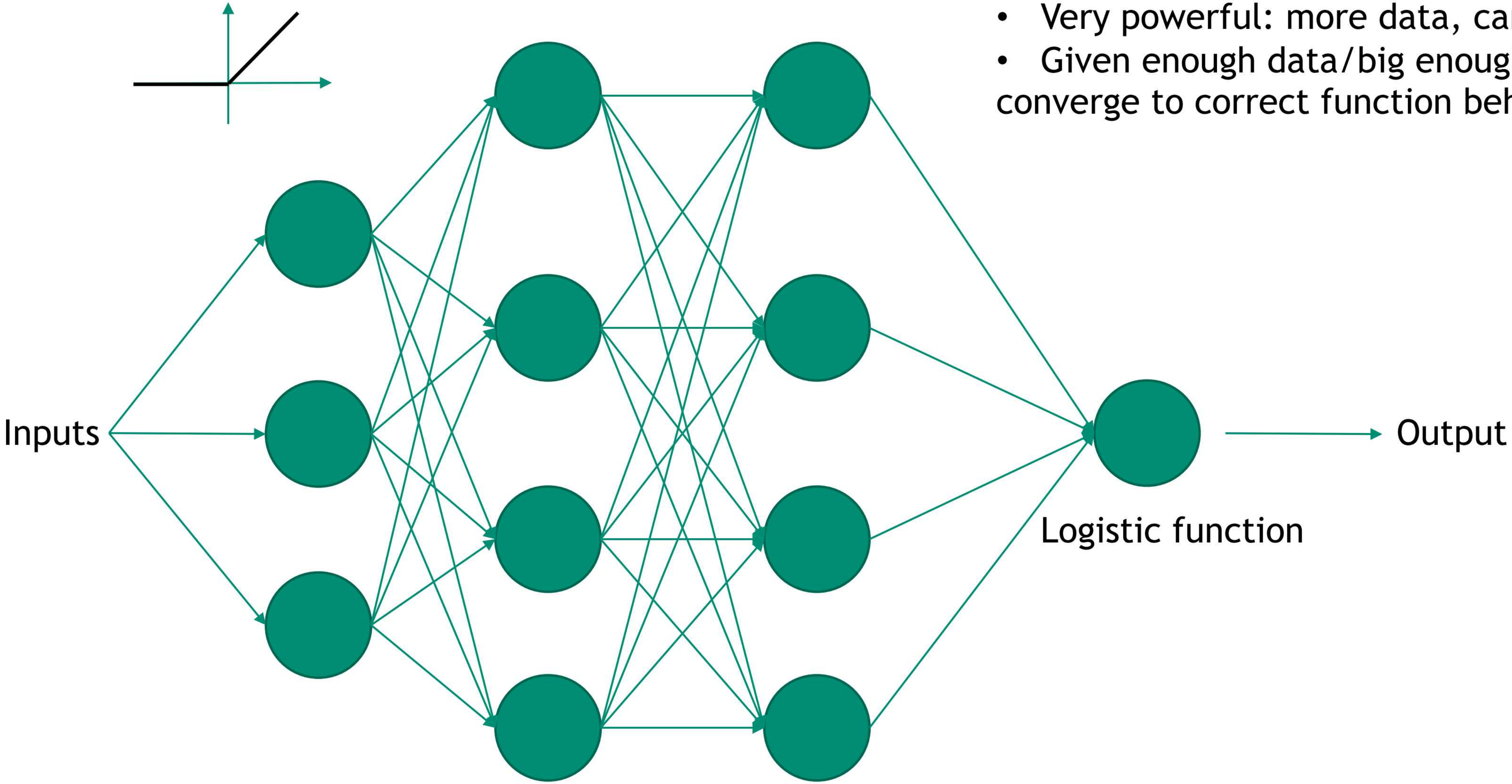
Yes… very much so



Artificial neural network

Rectified linear unit (ReLU)

- Non-convex, can be tricky to optimize/tune
- Very powerful: more data, can make bigger
- Given enough data/big enough network, will converge to correct function behavior



Mixture of linear and non-linear operations with their own tunable parameters

Neural network classifier

LOOCV: For each of 1600 samples:

- Hold one sample out
- Train NN classifier on the 1599 samples
- Predict the outcome of the remaining sample (If probability>50% => FAIL)
- Compare to true outcome

NN details:

- Scikit-learn
- Set to 3 layers of 10 nodes (did some rough tuning to get this as nearly optimal)
- Train with SGD-like algorithm (Adam)
- ReLU hidden units, logistic output

Neural network classifier

Breach rates observed in data:

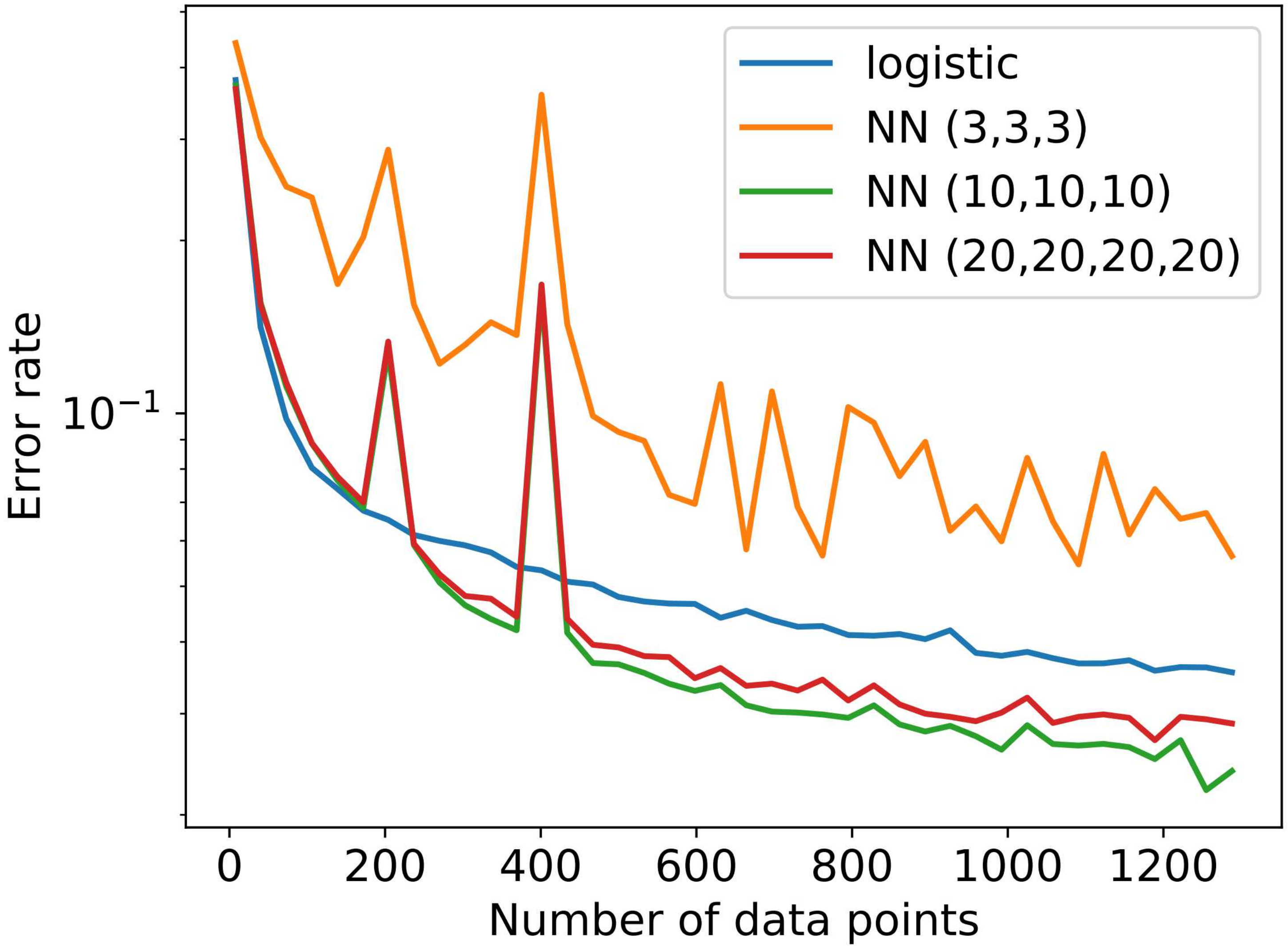
Forward case: 725/1600 failed
Aft case: 481/1600 failed
Housing: 423/1600 failed
Outer case: 1372/1600 failed

LOGISTIC	Accuracy	False negative rate	False positive rate
Forward case	96.63%	4.55%	2.40%
Aft case	95.69%	7.48%	2.95%
Housing	96.19%	8.51%	2.12%
Outer case	93.31%	2.33%	32.89%

NEURAL NETWORK	Accuracy	False negative rate	False positive rate
Forward case	97.81%	2.62%	1.83%
Aft case	95.75%	6.65%	3.22%
Housing	96.13%	6.38%	2.97%
Outer case	94.06%	2.91%	24.12%

NN is generally a little more accurate than logistic, and balances FPR/FNR better

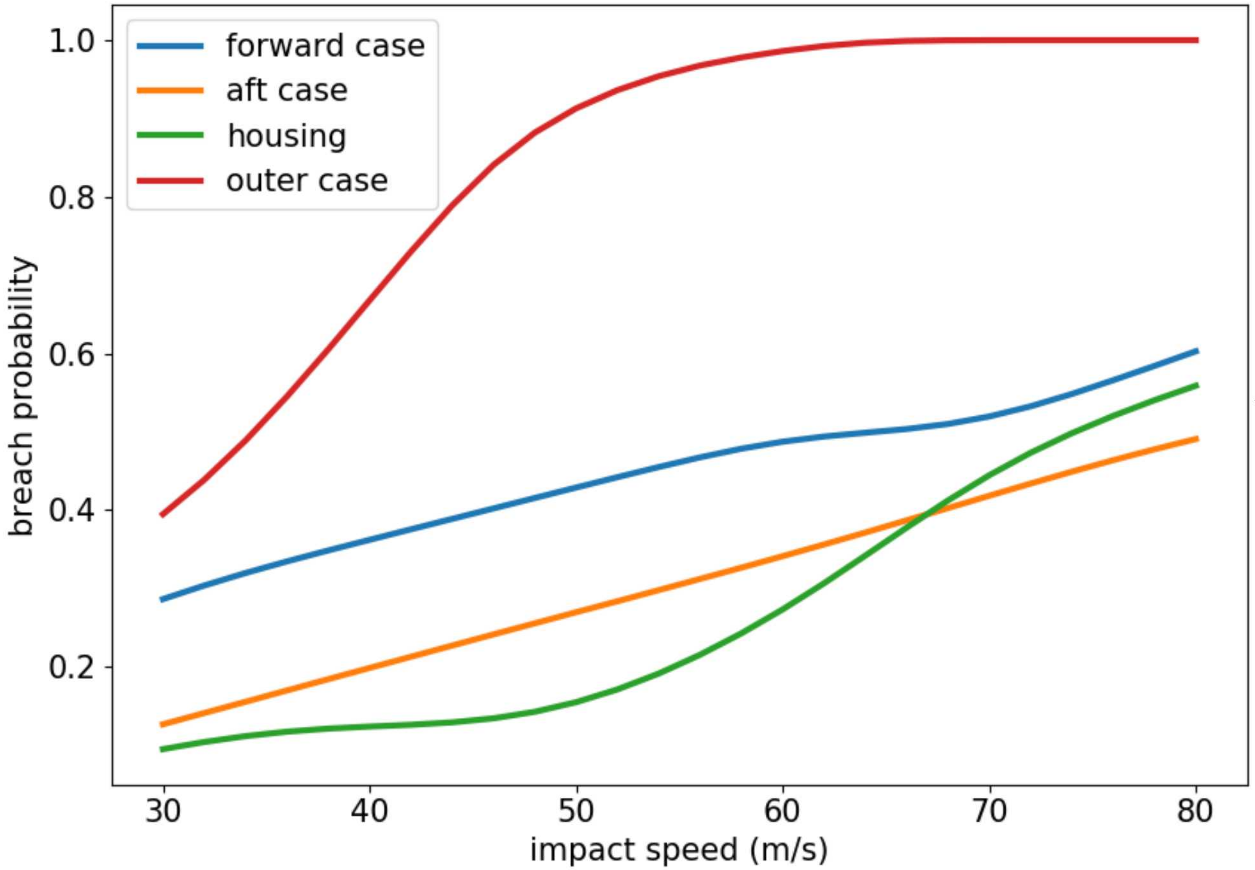
Learning Curves for Forward Case Predictions



NN: Probability of loss of containment

Sample NN over uncertain material properties for a fixed angle and speed

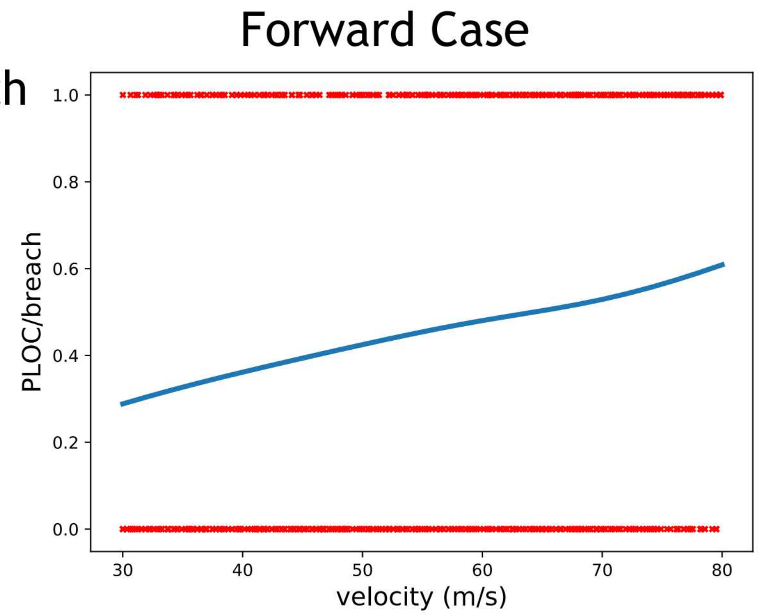
Repeat for all angles, and compute fraction that breach. 10 million samples per angle.

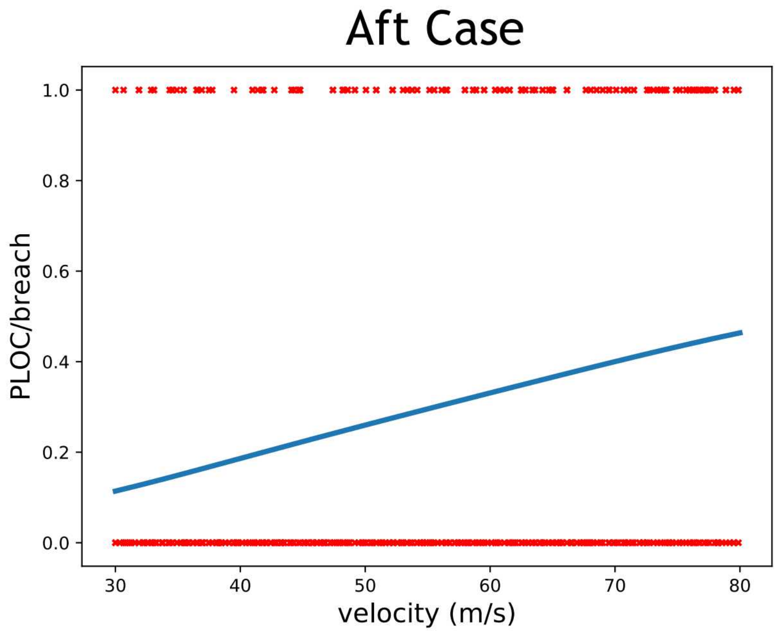


$$P(v) = \int m(v, \xi) d\xi \approx \frac{1}{N} \sum_{i=1}^{N} m(v, \xi_i)$$

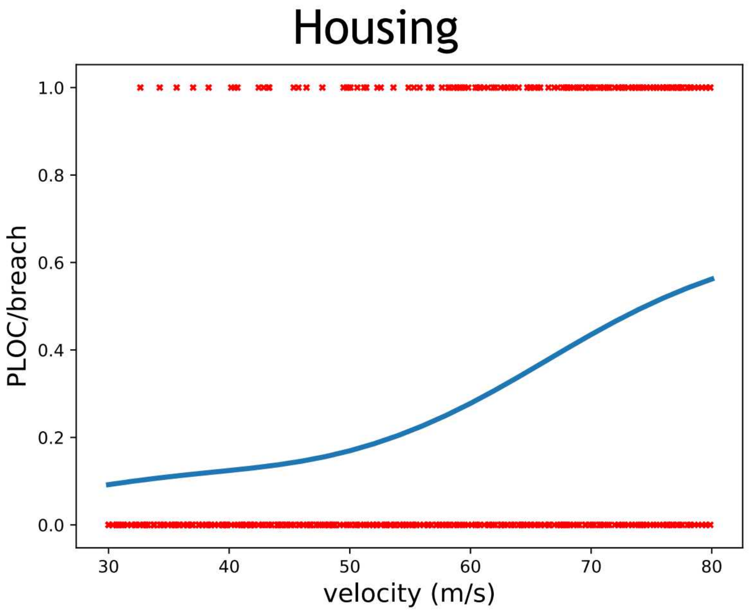
PLOC: NN versus the raw data

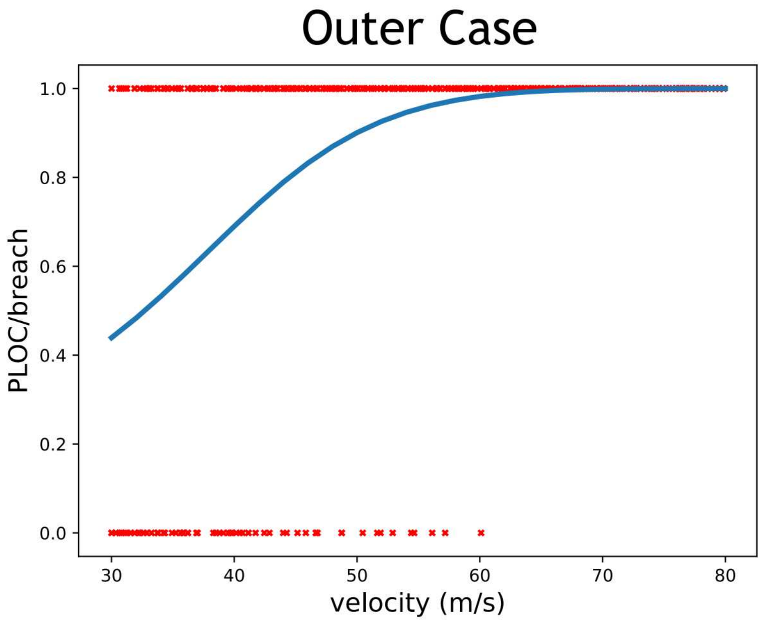
Recorded Breach





Recorded Pass





Fixed velocity study

Breach rates observed in data:

- Forward case: 688/1600 failed
- Aft case: 443/1600 failed
- Housing: 225/1600 failed
- Outer case: 1469/1600 failed

Now let's freeze the velocity at 50 m/s, 1600 new samples

LOGISTIC	Accuracy	False negative rate	False positive rate
Forward case	97.00%	1.74%	3.95%
Aft case	96.13%	9.48%	1.73%
Housing	97.31%	15.11%	0.65%
Outer case	94.69%	0.88%	54.96% (ouch)

NEURAL NETWORK	Accuracy	False negative rate	False positive rate
Forward case	97.44%	2.76%	2.41%
Aft case	95.50%	8.58%	2.94%
Housing	97.19%	15.11%	0.80%
Outer case	95.50%	1.97%	32.82%

NN takes a slight hit in accuracy compared to logistic, but still balances FNR/FPR better

Logistic regression: top 5 linear sensitivities for forward case at 50m/s

Uncertainty	Linear sensitivity index (normalized)
EQPS for element death	-0.46
Foam Poisson ratio	0.63
Aluminum Young's modulus	0.34
Friction	0.33
Steel Young's modulus	-0.29

NN: Top 5 Sobol indices of probability for forward case breach at 50m/s

Angle = 0 degrees

Uncertainty	Total effects sensitivity index
EQPS for element death	0.93
Foam Poisson ratio	0.03
Friction	0.02
Aluminum Young’s modulus	0.01
SS304 Young’s modulus	~0

Angle = 180 degrees

Uncertainty	Total effects sensitivity index
EQPS for element death	0.90
Foam Poisson ratio	0.19
Friction	0.18
SS286 Young’s modulus	0.09
Steel Young’s modulus	0.09

Conclusion

SAND2019-4786, UUR

- **With thanks to Andrew Murphy and Jay Dike for model development and expert elicitation**

Logistic regression:

- Pros: easy to interpret, easy to compute
- Cons: not very accurate

Neural network classifier:

- Pros: can be much more accurate, capture complex behavior
- Cons: difficult to optimize, not at all interpretable directly

All:

- Pros: can classify failures using surrogate models on limited data, compute sensitivities, predict probability of failure
- Cons: need to handle probability predictions with care! Analysts need to be reminded of caveats in these models

Application of these models to compute PLOC from sample data is straightforward

Future Work

Model Selection and UQ

- Use regularization/model selection to improve accuracy
- Bayesian treatment to add uncertainty bounds to predictions

Model-form error: use FPR/FNR to estimate error bounds on predicted PLOC

Adaptive simulations: draw more samples around area where model predicts failure to improve classifier accuracy

Picking the decision boundary carefully is CRUCIAL for nuclear safety

- **Ideally want to minimize the false negative rate**
- i.e. make sure we say it's safe if and only if it is actually safe