Global Sensitivity Analysis of xLPR using Metamodeling

(Machine Learning)

xLPR User Group Meeting

August 18, 2021

Background

- As part of applying xLPR to production analyses and to further validate the model, sensitivity analyses were conducted
  - ***Sensitivity studies*** can be used to assess the impacts of uncertain parameters and analysis assumptions on the results
  - ***Sensitivity analysis*** is a useful tool for identifying important uncertain model inputs that explain a large degree of the uncertainty in a quantity of interest
- Reasons to perform a sensitivity analysis:
  - Identify inputs that warrant greatest level of scrutiny, validation, and further sensitivity analysis
  - Identify inputs that are key to the results
  - Model validation
  - Improve understanding of model behavior
  - Reduction of model complexity (e.g., set “unimportant” inputs to constant values)
  - Inform advanced Monte Carlo sampling strategies (e.g., importance sampling)
- Available techniques *(see TLR-RES/DE/CIB-2021-11; ML21133A485)*:
  - One-at-a-time
  - Local partial derivatives (e.g., Adjoint Modeling)
  - Variance-based (e.g., Sobol method)
  - Linear regression
  - Metamodels

Sensitivity Analysis using Metamodels

- Why machine learning metamodeling?
  - Can handle correlated inputs
  - Accurately reflects non-monotonicity, non-linearity, and interactions
  - Importance measures reflect the whole input space
  - Several machine learning models automatically generate sensitivity metrics and down-select input variables based on information gained as part of the model fitting process
  - Fitted model can be used in place of the original model to compute quantitative sensitivity measures at lower computational cost
- Focus of this presentation: using built-in sensitivity metrics generated during fitting

Metamodeling Analysis Workflow

- Run the probabilistic code and collect results
- Implement metamodeling code
  - Import results from probabilistic code runs
  - Transform results to prepare for input to metamodel fitting (e.g., accounting for spatially sampled variables)
  - Fit the metamodel, including parameter optimization using cross-validation
  - Extract and report input importance metrics
- Evaluate
  - Examine goodness of fit metrics
  - Compare importance ranking results from alternate metamodels
  - Compare importance ranking results across different outputs of interest
- Iterate
  - Collect more inputs
  - Analyze different outputs
  - Run different discrete configurations of the probabilistic code
  - Use different metamodels / different metamodel parameters

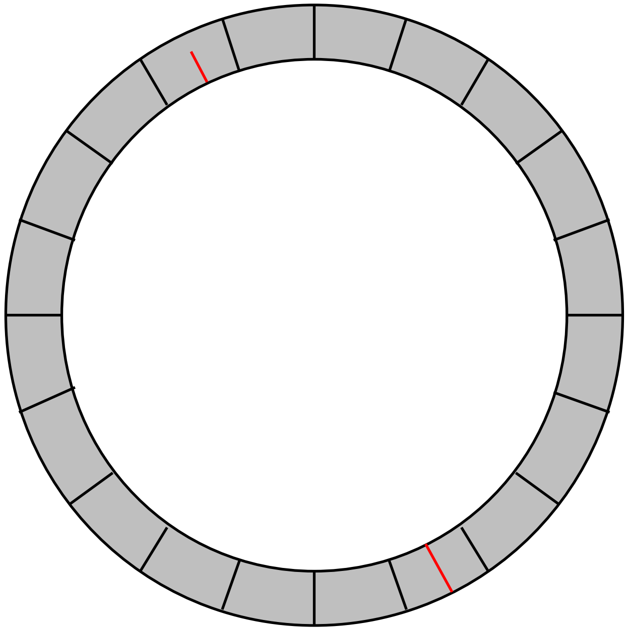
Model Implementation

- Python 3.6 using Scikit Learn Package*
- Machine learning models implemented:
  - Gradient Boosting Decision Trees
  - Random Forest Decision Trees
  - Linear Support Vector Machines
- All models used are classifiers (as opposed to regressors) because the outcomes are binary (yes/no). Regressor models would be used for scalar outputs.
- All models include metrics for feature selection / feature importance
- Initial work focused on subset of 60 inputs:
  - Inputs that are expected to have high importance
  - Distributed inputs
  - Constant inputs uniformly distributed from 0.8 to 1.2 times constant value
- Outputs analyzed:
  - Occurrence leak
  - Occurrence rupture (with and without inservice inspection (ISI))

*Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011

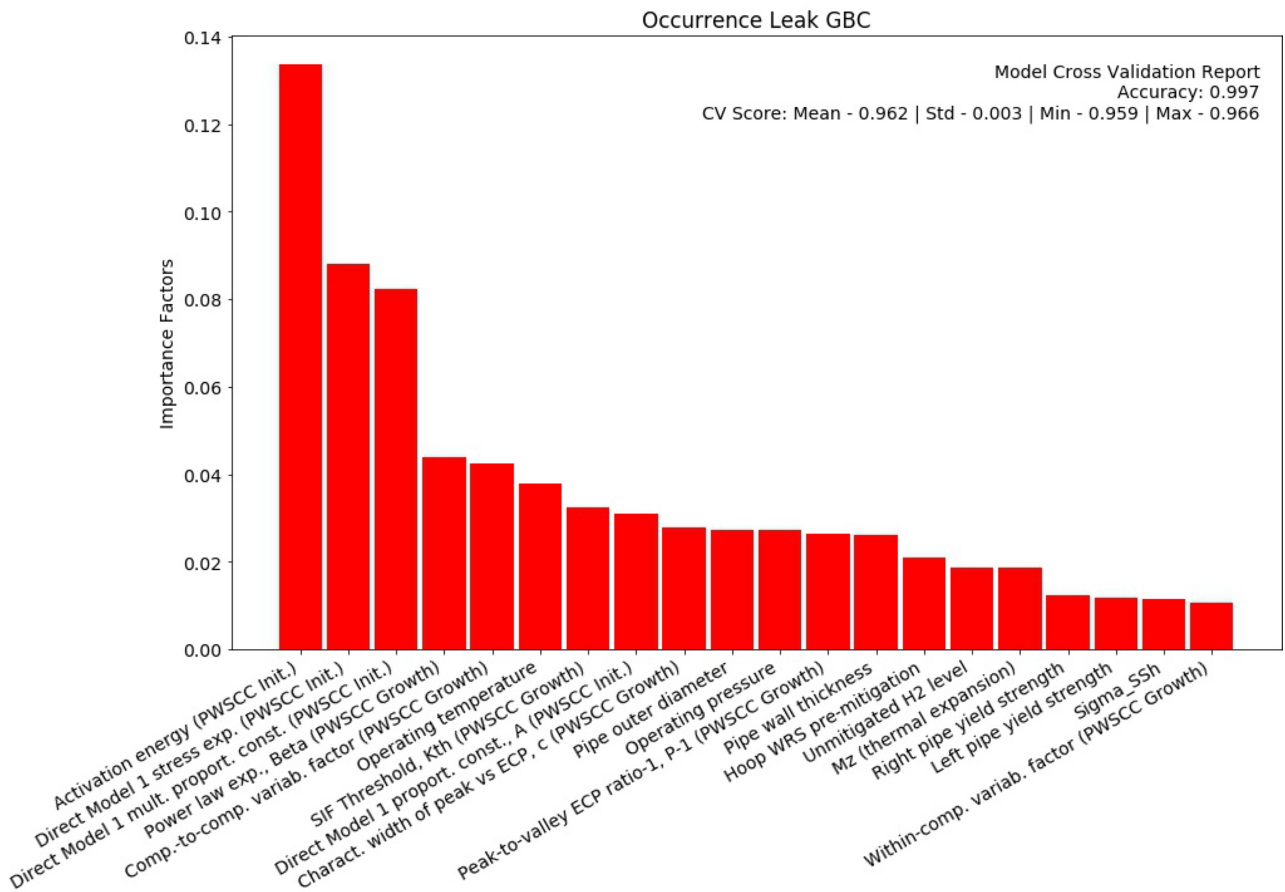
Spatially Distributed Inputs / Outputs

- Pipe section split into 19 subunits that can potentially crack
- Some inputs sampled on a subunit basis
- Some outputs also available on a subunit basis
- Aggregation methodology for subunit inputs / outputs
  - Pipe subunit inputs and outputs: Analyze each pipe subunit and crack direction separately and average feature importance metrics
  - Pipe subunit inputs and global outputs: Average input across all pipe subunits (and crack types) and perform single analysis to determine feature importance
  - This method may cause underreporting of importance metrics in comparison to alternative methods



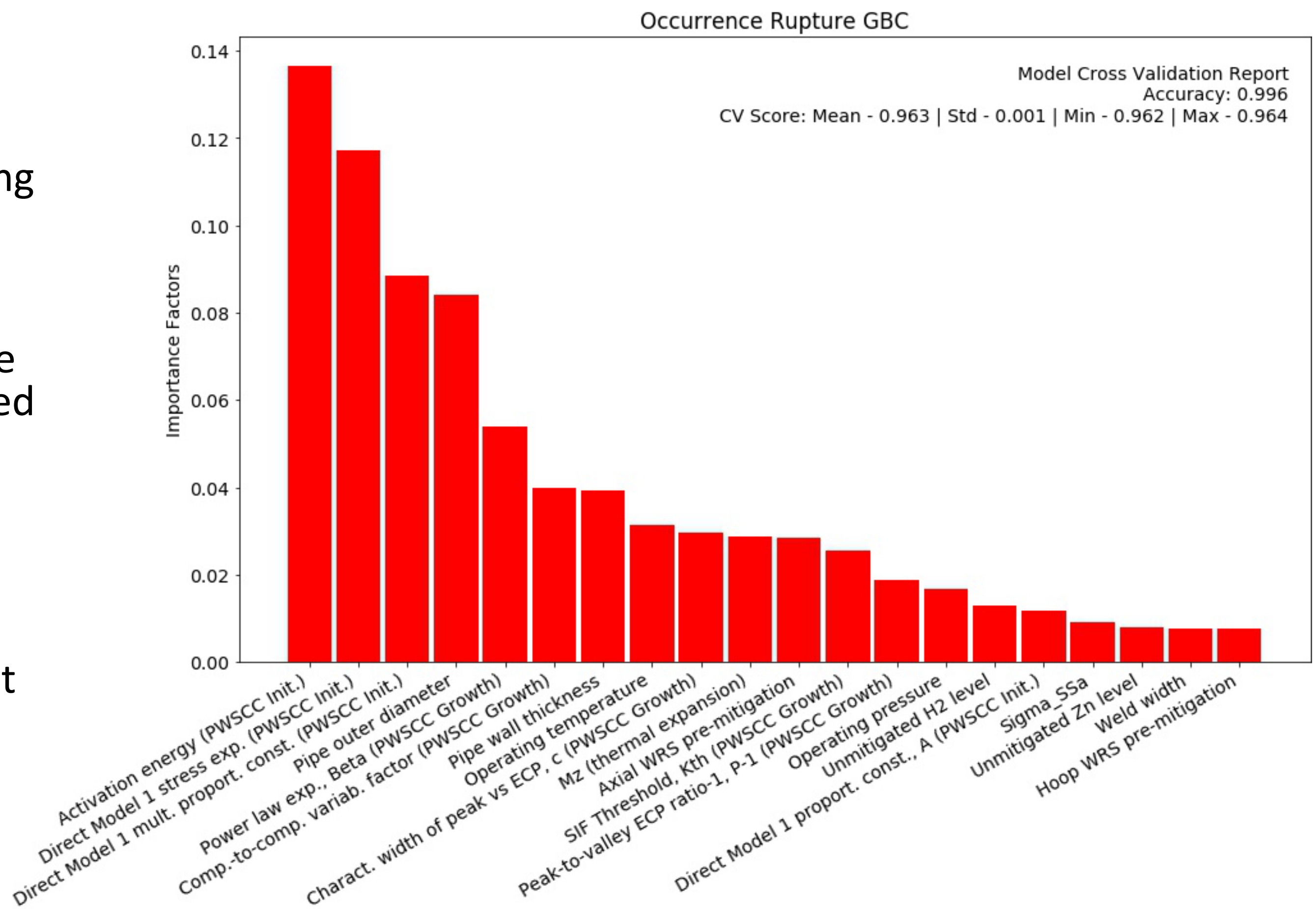
Results: Leak Output

- Output: Leak (through wall crack) in any pipe subunit
- Analyzed using Gradient Boosted Trees Classifier (GBC)
- Allows comparison between averaging subunit inputs and averaging subunit analysis outputs
- Top importance parameters for averaged subunit inputs:
  - Primary water stress-corrosion cracking (PWSCC) initiation parameters
  - PWSCC growth parameters
  - Operating Temp./Pressure
  - Pipe outside diameter / Thickness
  - Welding Residual Stresses (WRS) - Hoop
  - Pipe yield strength



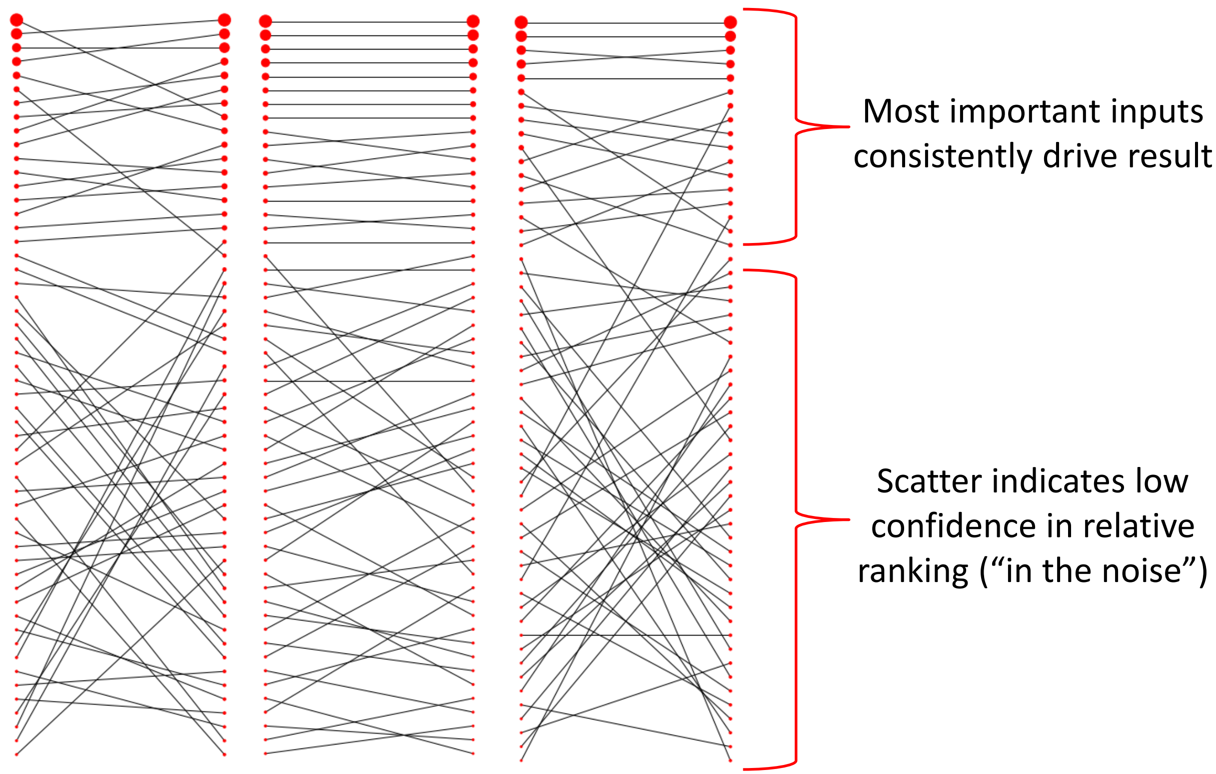
Results: Rupture Output

- Rupture full model output (not subunit basis)
- Analyzed using all three machine learning classification algorithms
- Best prediction accuracy and CV score using Gradient Boosted Trees Classifier
- General agreement between all three fitted models
- Top importance parameters consistent with leak parameters
  - PWSCC initiation
  - Axial WRS ranked above Hoop (opposite of leak)



Changes in Importance Rankings

- Importance factor results may be compared between different scenarios/cases to show changes in the relative ordering of inputs
- Useful for:
  - Comparison between alternate metamodeling approaches
  - Determining differences in sensitivity between different outputs of interest
  - Comparing runs with different model settings (e.g., different ISI intervals)



Conclusions

- Key findings
  - Relative comparisons (e.g., Axial vs. Circ, Rupture with/without ISI) are very useful for sanity checking the model
  - Relatively high confidence in the identification of highest-impact inputs but low confidence in ordering of low-impact inputs
- General challenges
  - Input distributions need to be selected carefully to get informative results
    - A default real-world analysis input set is probably not sufficient
  - Special consideration needed for inputs that are not continuous variables (e.g., settings flags)
- xLPR-specific challenges
  - Prediction of simulation-wide outcomes using subunit-level sampled values
  - Consideration of all inputs would be time-intensive (labor to extract sampled values and simulation time to adequately cover full input space)
- Potential future improvements
  - Include more inputs in the machine learning model
  - Examine other outputs of interest (e.g., leak rate jump indicator)
  - Examine alternate configurations that can't be covered automatically using input distributions
  - Use more advanced methods to improve on the relative rank importance metric (e.g., variance decomposition)