AAET
ADVANCED ANALYTICS & EMERGING TECHNOLOGIES

# Petrophysics-Driven Well Log Quality Control Using Machine Learning

Michael Ashby, Natalie Berestovsky, Ingrid Tobar

September 17-19, 2019

ANADARKO PETROLEUM CORPORATION

# Petrophysics-driven Well Log Quality Control Using ML

## BUSINESS PROBLEM

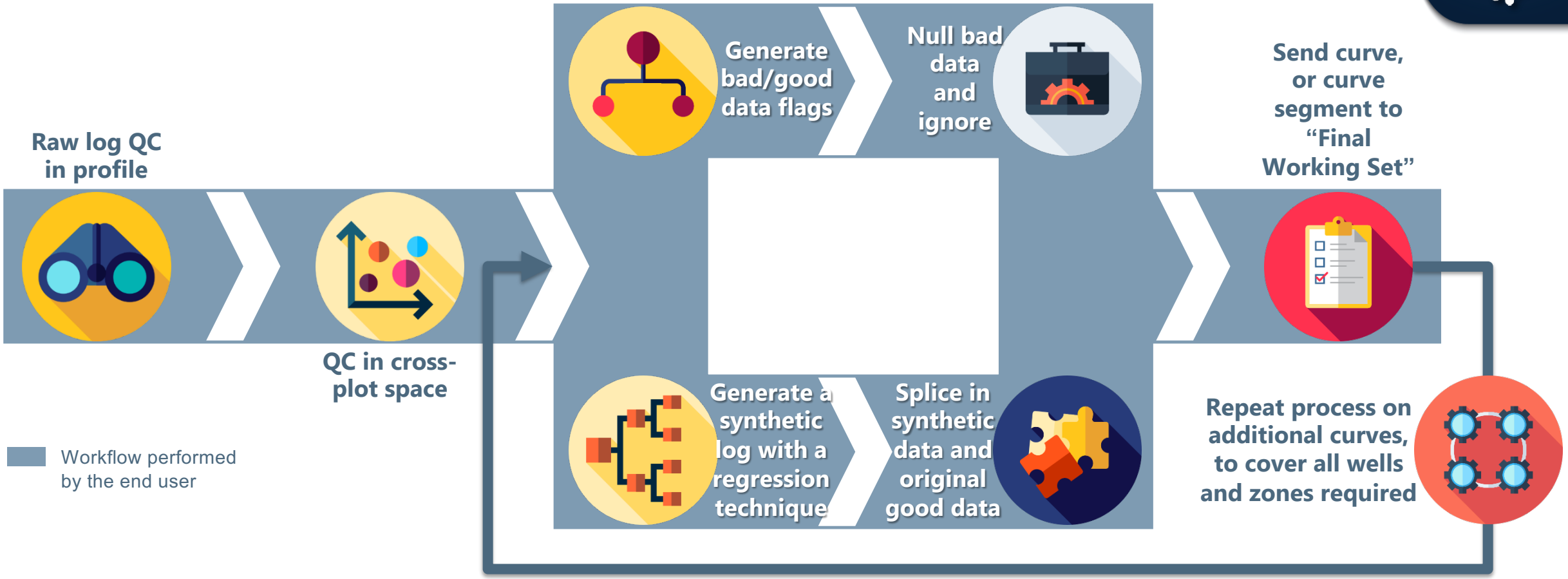
Petrophysicists rely on well log data to derive valuable information about reservoirs

- ***Well log data is not always optimal*** due to
  - Poor borehole conditions,
  - Acquisition errors, and
  - Tool failures
- Accuracy of log interpretation depends on ***time spent*** on data quality control (QC) and conditioning
- Reducing time on data QC enables petrophysicists to proceed more quickly to log interpretation

## SOLUTION

Our team developed and implemented a tool that integrates ***insights from petrophysics*** with ***data science techniques***, significantly reducing the amount of time dedicated to log QC and editing.

# Log QC Manual Process



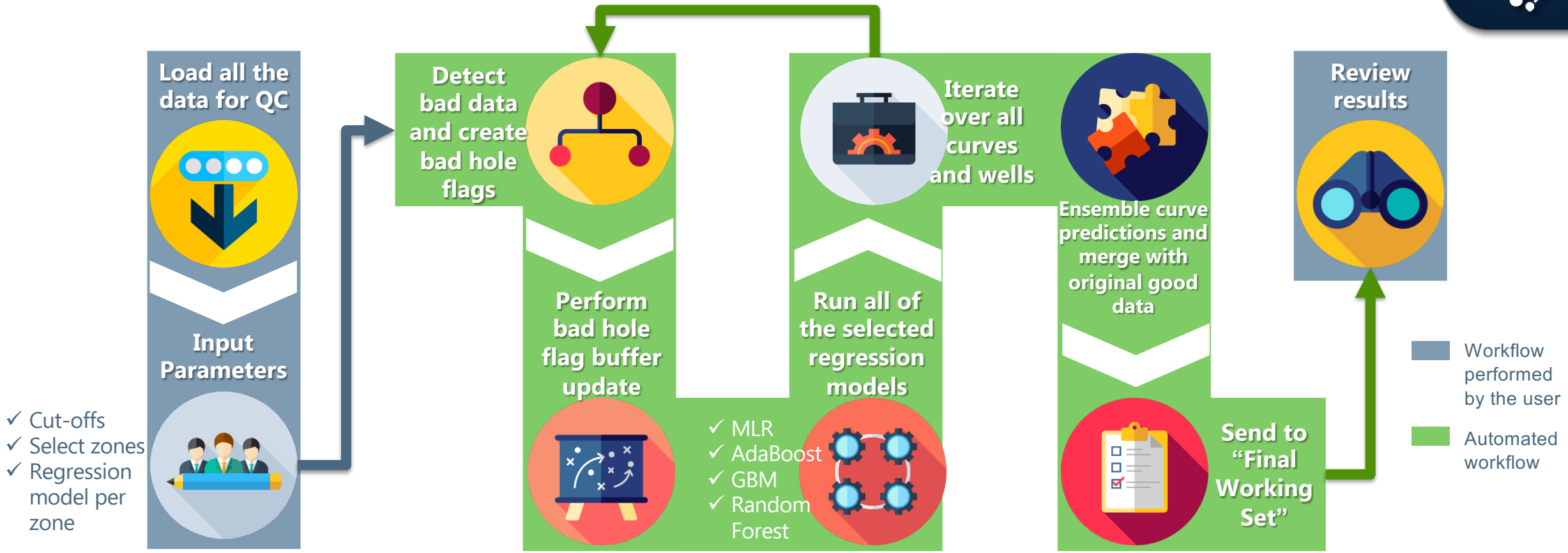

## Advantages

- Addresses issues with log data in advance
- Reproducible results, if the workflow is documented

## Disadvantages

- Very manual and time consuming process
- User must repeat the same process for each zone, each well and each curve

# Log QC Automated Process (Single Well)




## Advantages

- Automated bad hole detection, flagging and bad hole flag buffer can be applied to multiple curves/wells
- Can execute multiple regressions simultaneously and ensemble the best results from all the models

## Disadvantages

- Each well is reconstructed independently
  - *Currently working on a multi-well solution*

# Algorithm Selection

| TYPE | NAME | DESCRIPTION | ADVANTAGES | DISADVANTAGES |
|---|---|---|---|---|
| Linear | Linear regression | **The "best fit" line** through all data points. Predictions are numerical. | Easy to understand -- you clearly see what the biggest drivers of the model are. | X Sometimes too simple to capture complex relationships between variables.<br>X Tendency for the model to "overfit". |
| Linear | Logistic regression | The adaptation of **linear regression** to problems of classification (e.g., yes/no questions, groups, etc.) | Also easy to understand. | X Sometimes too simple to capture complex relationships between variables.<br>X Tendency for the model to "overfit". |
| Tree-based | Decision tree | A graph that uses a **branching method** to match all possible outcomes of a decision. | Easy to understand and implement. | X Not often used on its own for prediction because it's also often too simple and not powerful enough for complex data. |
| Tree-based | Random Forest | Takes the average of many decision trees, each of which is made with a sample of the data. Each tree is weaker than a full decision tree, but **by combining them we get better overall performance.** | A sort of "wisdom of the crowd". Tends to result in very high quality models. Fast to train. | X Can be slow to output predictions relative to other algorithms.<br>X Not easy to understand predictions. |
| Tree-based | Gradient Boosting | Uses even weaker decision trees, that are increasingly **focused on "hard" examples.** | High-performing. | X A small change in the feature set or training set can create radical changes in the model.<br>X Not easy to understand predictions. |
| Neural networks | Neural networks | Mimics the behavior of the brain. Neural networks are interconnected neurons that pass messages to each other. Deep learning uses several **layers of neural networks put one after the other.** | Can handle extremely complex tasks - no other algorithm comes close in image recognition. | X Very, very slow to train, because they have so many layers. Require a lot of power.<br>X Almost impossible to understand predictions. |

Source: © 2017 Dataiku, Inc. www.dataiku.com

## Explored:

- ✓ Random Forest 2
- ✓ Bayesian Ridge
- ✓ NN Regression
- ✓ AdaBoost 1
- ✓ SVM
- ✓ Lasso
- ✓ Gradient Boost Machine (GBM) 3
- ✓ XG Boost
- ✓ Light GBM

## Selected:

- ❖ AdaBoost
- ❖ Random Forest
- ❖ Gradient Boost Machine (GBM)
- ❖ Multilinear Regression (MLR)

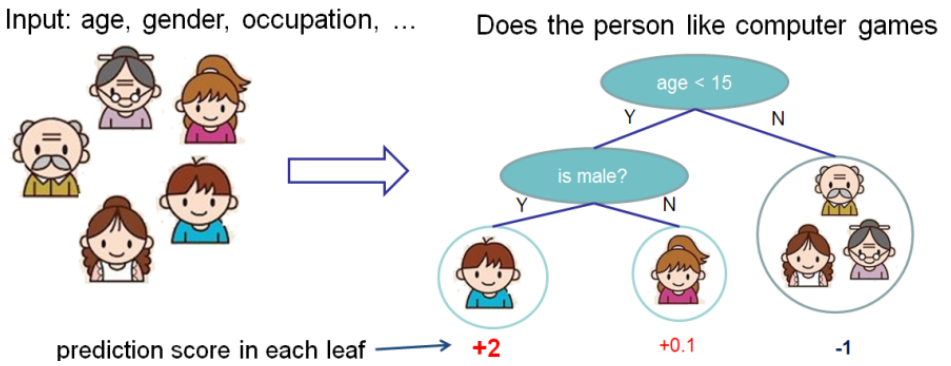
*Added MLR as a simple, computationally less intensive benchmark to compare results against*

# Random Forest

- Random Forest (or decision forest) is an ensemble learning method based on **averaging decisions** across multiple decision trees
- Alone, a single decision tree is prone to overfitting
- Random forests generate many decision trees, and each tree is trained on a random subset of the data in a process known as bootstrap aggregating, or 'bagging'
- While each individual tree is likely to overfit the training data that it has been given, the average across all of the trees is expected to correct this tendency to overfit

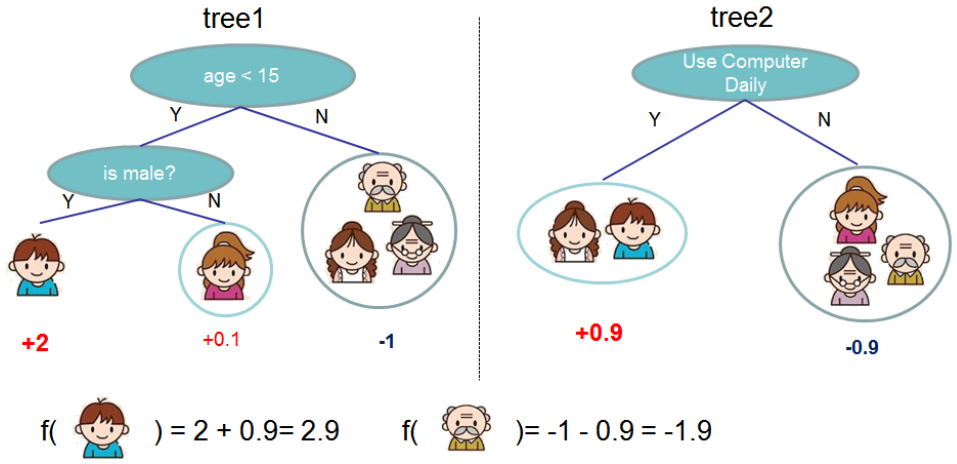
Source: Brieman 2001; Hastie, Tibshirani, and Friedman 2009

**Example:**
Predict whether someone likes computer games




### Single Tree Model

- Classify the subjects into different leaves, and assign them the score on the corresponding leaf
- Usually, a single tree is not strong enough to be used in practice
- What is actually used is the tree ensemble model, which sums the prediction of multiple trees together




### Ensemble Random Forest Model

- This is an ensemble of two trees:
  1. Prediction score based on age
  2. Prediction score based on daily computer use
- Prediction scores of each individual tree are added to get the final score
- The two trees try to complement each other

Source: XGBoost, http://xgboost.readthedocs.io/en/latest/model.html

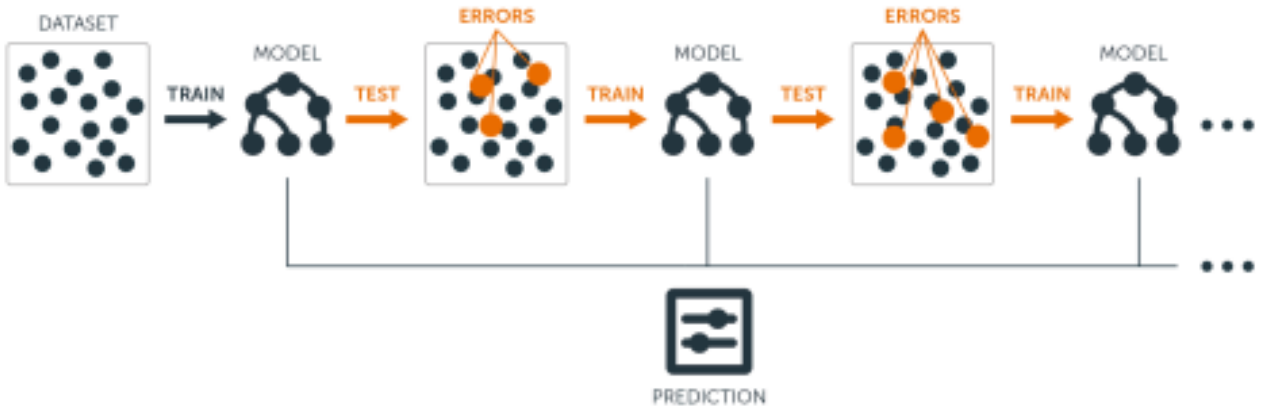
# Boosting Algorithms

- Includes **Gradient Boosting (GBM)** and **AdaBoost**
- Decision trees, while prone to overfitting, are essential building blocks to many machine learning algorithms
- While the Random Forest algorithm uses a 'bagging' approach to train many trees on random subsets of the data, 'boosting' algorithms take a more direct approach when sub-setting data and training trees
- To minimize prediction error, a boosting algorithm generates a series of **weak learners** –decision trees that perform at least slightly better than random chance– and combines them to generate a **strong learner**

Source: Drucker, 1997

- Weak learners are trained **iteratively**, so that the goal for each learner is to predict data points that the previous tree had difficulty predicting
- Each subsequent learner '**boosts**' the accuracy of the previous learners

**Iterative Process of Boosting Algorithms**



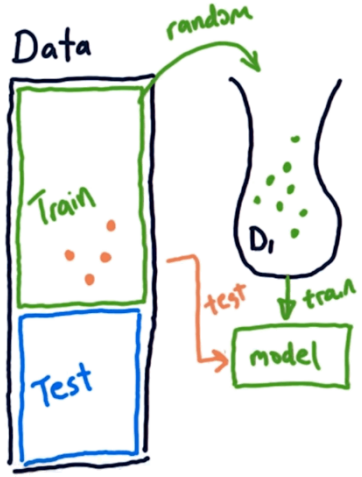

Source: Introduction to Boosted Trees, https://blog.bigml.com/2017/03/14/introduction-to-boosted-trees/

# AdaBoost

- Each consecutive decision tree is **preferentially trained** on data that was difficult for previous trees to accurately predict
- Data subsets are generated by first assigning each observation a probability of being sampled
  - This probability is determined by how difficult it is for a decision tree to predict the observation, so that more difficult observations have higher probabilities of being sampled
  - Decision trees are intentionally trained on points that are difficult to predict
- Sample weights –the probabilities used to determine which observations from the training data are sampled– are updated with each **iteration**
- The prediction returned for any set of observations is the weighted median prediction across all of the decision trees
  - Medians are weighted by the confidence each decision tree has in the accuracy of its prediction

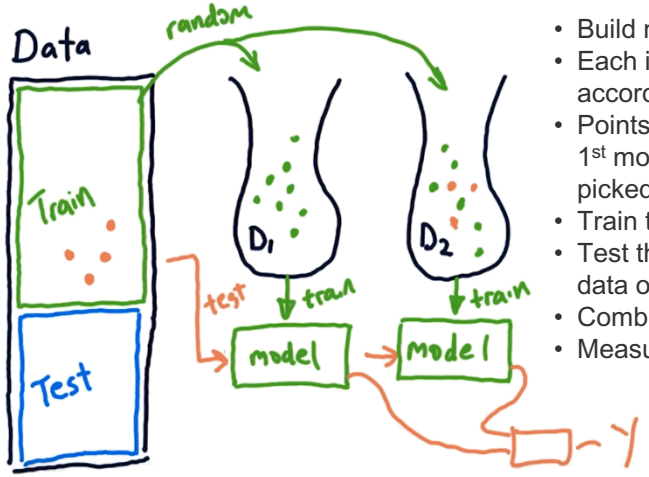
Source: Drucker, 1997

**Train first model**



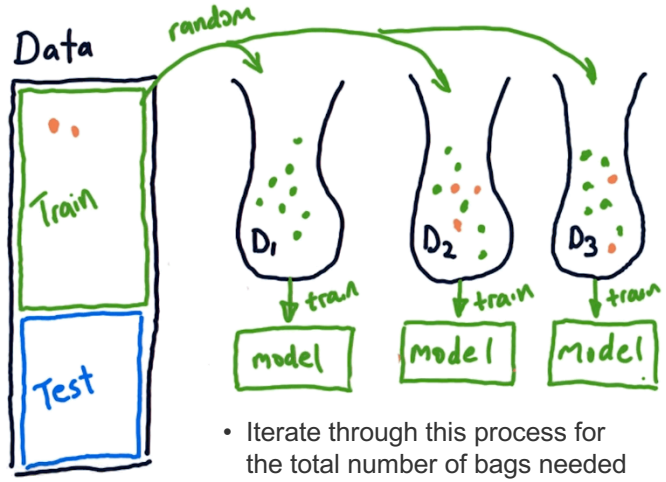

- Build first bag of data selecting randomly from training data
- Train the model
- Take all training data and use it to test the model
- Some points are not well predicted (**error**)

**Train next model**




- Build next bag (random selection)
- Each instance weighted according to **error** from 1st model
- Points with significant **error** from 1st model are more likely to get picked for this bag
- Train the next model
- Test the system using training data on both model instances
- Combine the outputs
- Measure **error** across all the data

**Iterate through *n* number of bags**




- Iterate through this process for the total number of bags needed

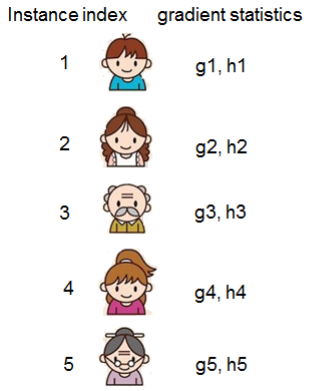
Source: Udacity course "Machine Learning for Trading", https://www.youtube.com/watch?time_continue=52&v=GM3CDQfQ4sw

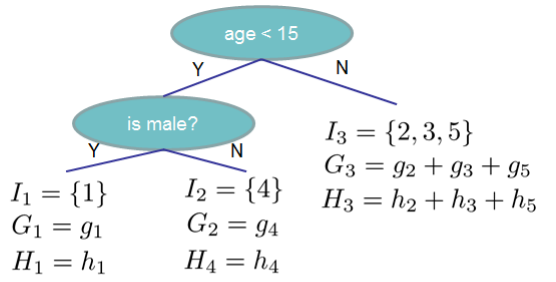
# Gradient Boosting (GBM)

- Trains **first tree on the observed data**, and trains each **remaining tree on the residual error** between the first tree's predicted values and the observations in the training data
  - Instead of creating multiple decision trees and training each tree on the observed data (AdaBoost)
- New trees are **parameterized** to minimize the residual error using gradient descent
- Builds a hierarchical model where each subsequent tree aims to decrease the residual prediction error
  - Instead of building a suite of trees that are able to make accurate predictions in concert (AdaBoost)
- Since each tree is part of this hierarchical model the prediction returned is simply a sum of predictions across all trees
  - Instead of a weighted 'voting' system (AdaBoost)

Source: Friedman, 2001

**Continuing the first example:**
Predict whether someone likes computer games

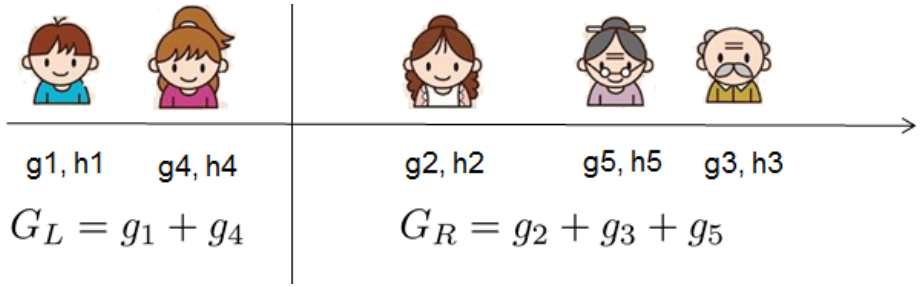



$$Obj = -\sum_j \frac{G_j^2}{H_j + \lambda} + 3\gamma$$

The smaller the score is, the better the structure is

$$Gain = \frac{1}{2}\left[\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda}\right] - \gamma$$




- Left to right scan is sufficient to calculate the structure score of all possible split solutions and find the best split efficiently

Source: XGBoost, http://xgboost.readthedocs.io/en/latest/model.html

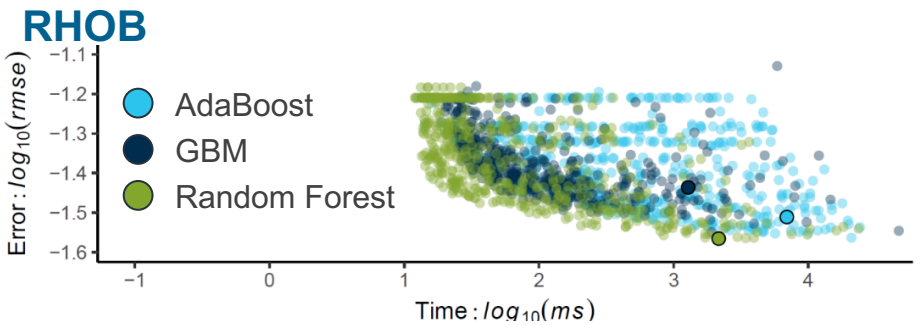
### Calculate Structure Score

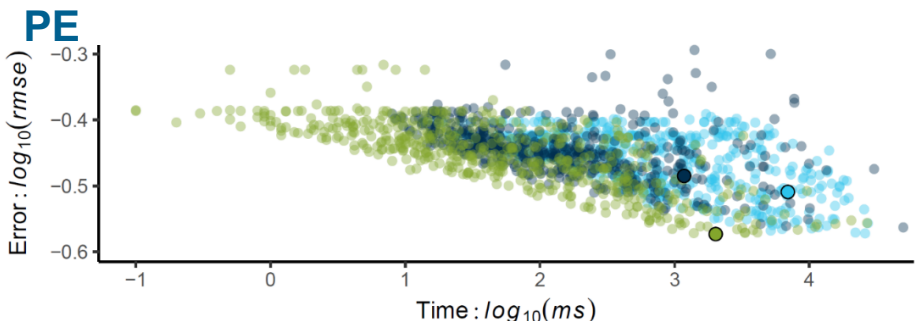
- Define an Objective Function (*Obj*)
- Push statistics $g_i$ and $h_i$ to the leaves they belong to
- Sum statistics together
- Use *Obj* to calculate how good the tree is
- This score is an impurity measure and takes into account model complexity

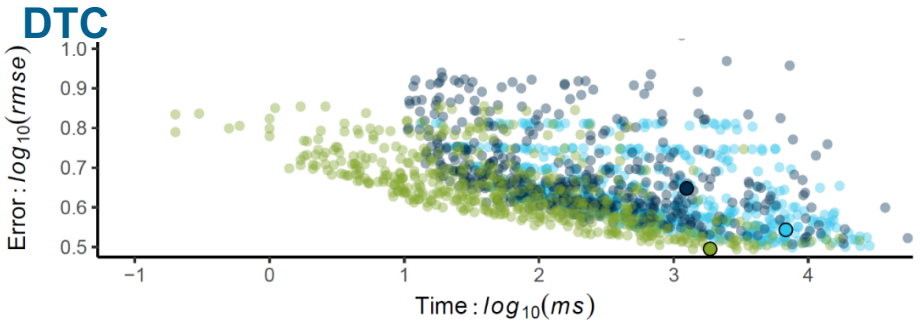
### Learn Tree Structure

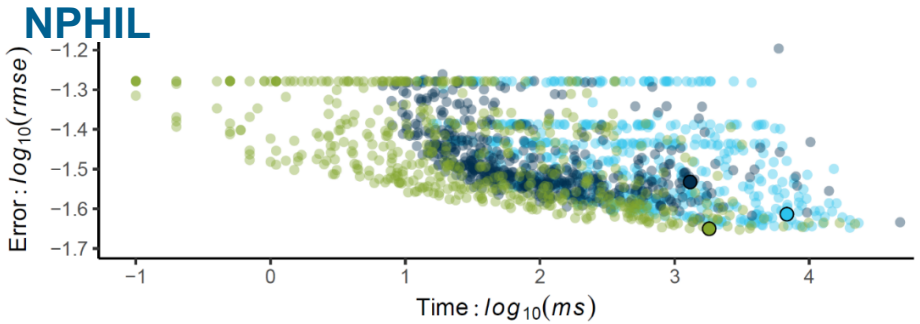
- Split a leaf into two leaves and the score is given by the *Gain* formula
- If the gain is smaller than $\gamma$ it would be best not to add that branch (tree-based model pruning technique)
- Place all the instances in sorted order

# Hyperparameter Tuning and Results




- Graphs show error (RMSE) vs processing time (milliseconds) for each model/curve combination using different hyperparameter settings
- Outlined circles highlight error/processing time for the default hyperparameter settings
- Random forest typically achieves the lowest error in the fastest processing time

## Relative RMSE

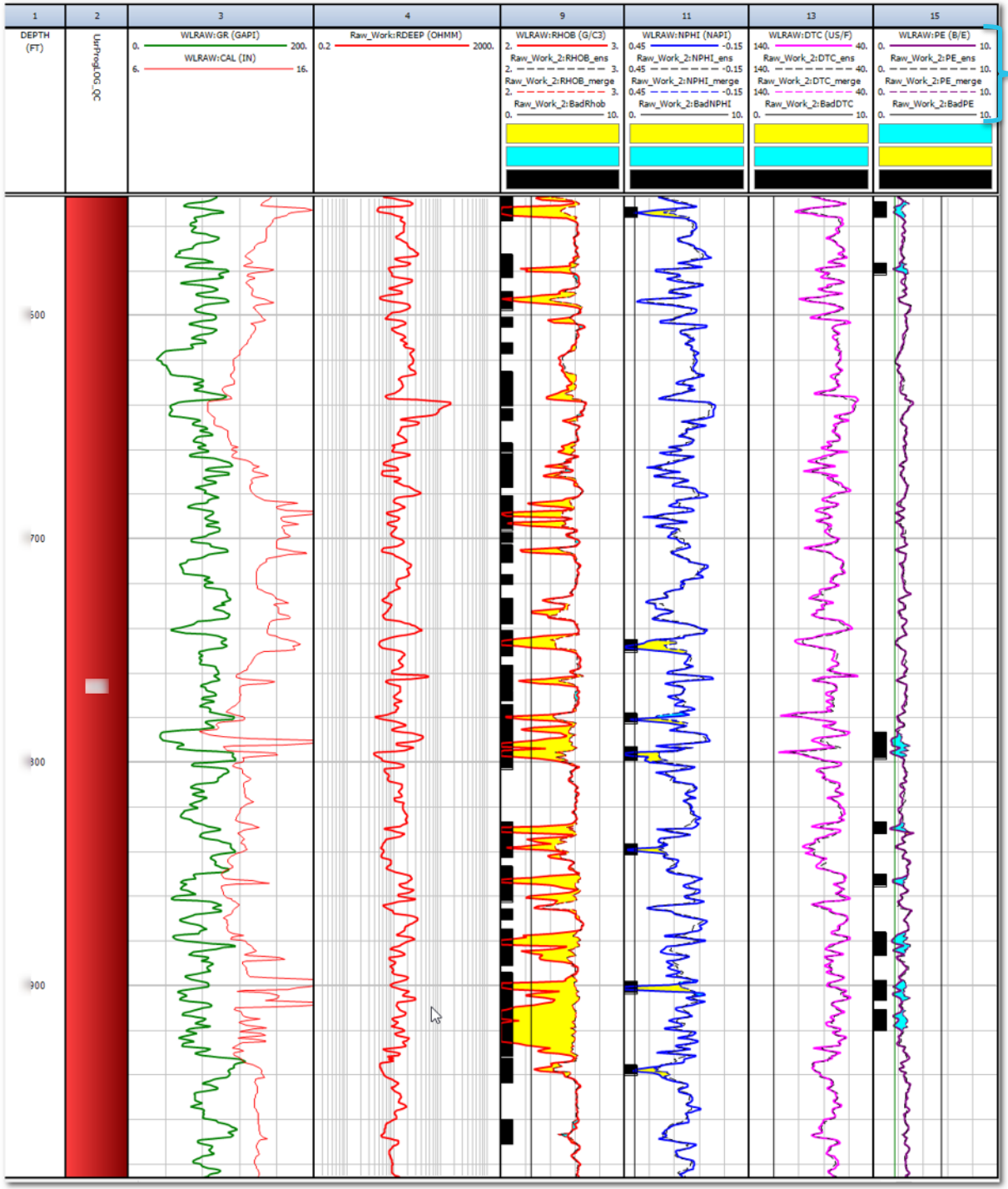
### *Default Settings*

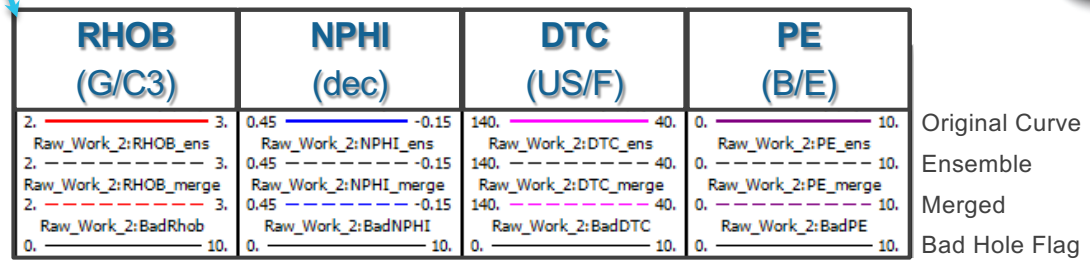
| | NPHIL | RHOB | DTC | PE |
|---|---|---|---|---|
| **AdaBoost** | 1.0 ± 3.13E-04 | 1.0 ± 3.01E-04 | 1.0 ± 2.12E-02 | 1.0 ± 3.40E-03 |
| **Gradient Boosting** | 1.0 ± 4.16E-04 | 1.0 ± 4.57E-04 | 1.0 ± 5.18E-02 | 1.0 ± 4.43E-03 |
| **Random Forest** | 1.0 ± 3.82E-04 | 1.0 ± 4.65E-04 | 1.0 ± 2.57E-02 | 1.0 ± 3.06E-03 |

### *Minimized Error*

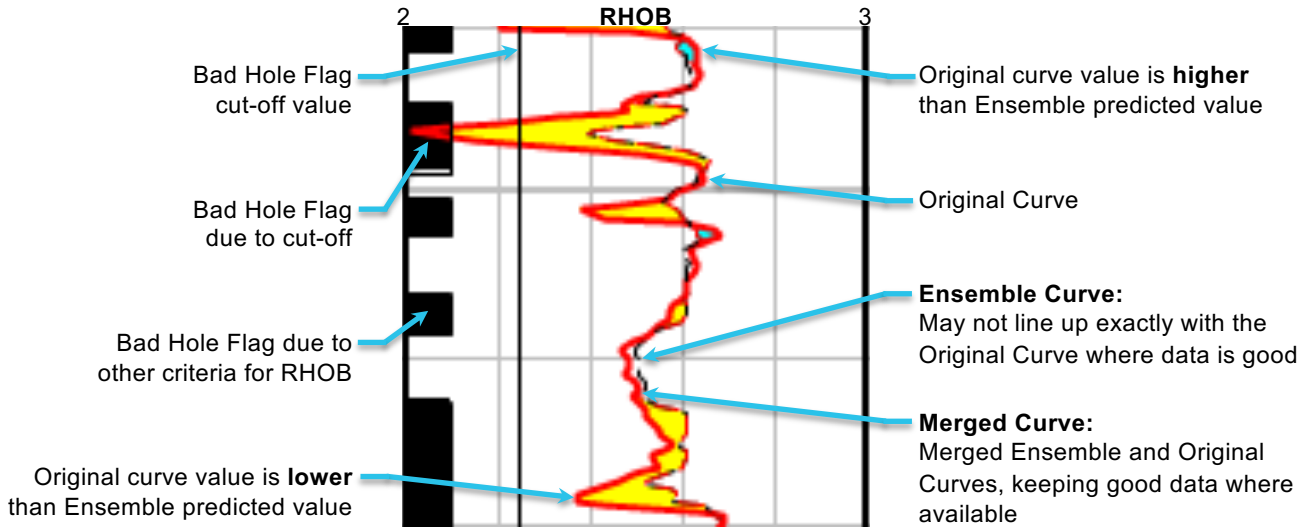
| | NPHIL | RHOB | DTC | PE |
|---|---|---|---|---|
| **AdaBoost** | 0.936 ± 3.66E-04 | 0.896 ± 4.26E-04 | 0.918 ± 3.36E-02 | 0.878 ± 2.85E-03 |
| **Gradient Boosting** | 0.803 ± 2.90E-04 | 0.776 ± 4.49E-04 | 0.730 ± 2.58E-02 | 0.843 ± 3.28E-03 |
| **Random Forest** | 1.005 ± 3.62E-04 | 0.998 ± 5.04E-04 | 0.995 ± 3.29E-02 | 1.000 ± 4.02E-03 |

# Single well reconstruction, with multiple curves



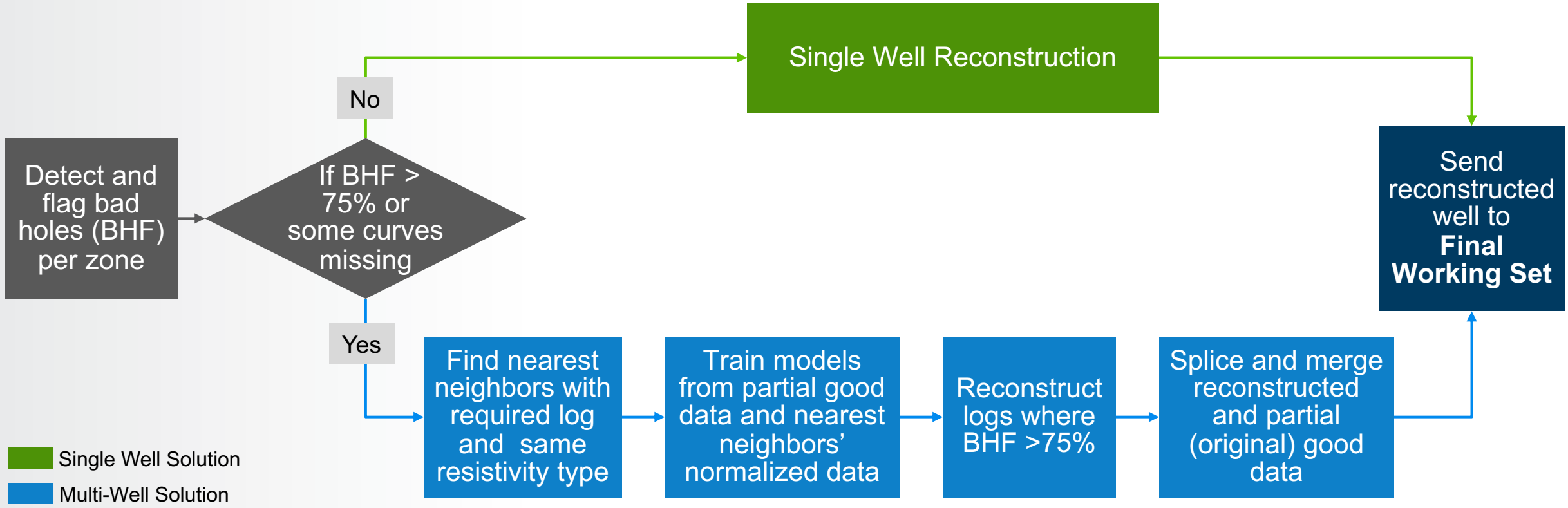

- **Original Curve:** As is, before reconstruction
- **Ensemble:** Curve predictions (calculated by one or multiple methods: MLR, ADA, GBM, RF), and assembled into a single curve
- **Merged:** Merged ensemble and original curves, where ensemble curve predictions replace bad hole sections, and good/valid original curve data remains in place.



# Approach

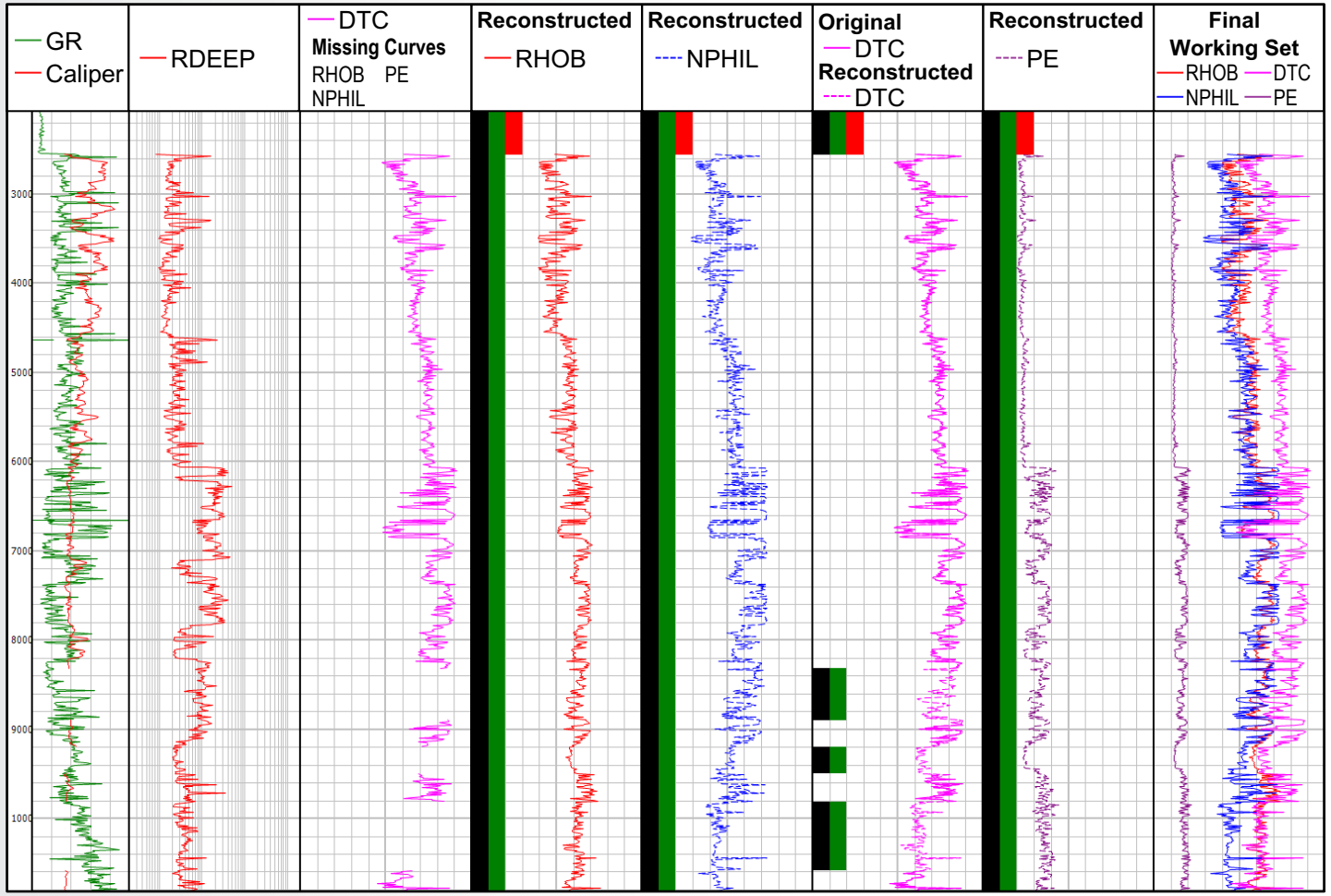
## MULTI-WELL IMPLEMENTATION

The tool operates on multiple wells simultaneously, training models from partial good data and nearest neighbors' normalized data to predict values at bad hole flag areas across all of the wells in the selection



# Approach

## LOG QC MULTI-WELL IN ACTION




**Legend**

- **Bad hole flag**
  Caused by entire curves missing (RHOB, NPHIL, PE) or missing segments of the curve (DTC)
- **Multi-well Flag**
  Sections that were reconstructed by multi-well using partial raw data from this well (GR, RDEEP, DTC) and available good data from neighboring wells
- **No Correction Flag**
  Sections that cannot be reconstructed by multi-well due to insufficient data (RDEEP curve missing at top)

### ADVANTAGES

- ✓ Uses single well solution when possible (defaults to multi-well only when necessary)
- ✓ Larger dataset to use for reconstruction
- ✓ Create synthetic data from neighbor wells

### DISADVANTAGES

- ➢ Can be slow
- ➢ Reconstruction is entirely dependent on quality and distance of offset wells
- ➢ Requires basin-wide, expert-derived cutoffs

# Validation and scaling

## VALIDATION

- ✓ Conducted cross-validation (90/10)
  - ✓ 300 wells from Uinta Basin Green River
  - ✓ 5,800 well from Delaware Basin

| Curve | # of Wells | % Mean Error* |
|---|---|---|
| RHOB | 92 | 5.3 |
| NPHIL | 78 | 9.8 |
| DTC | 64 | 6.8 |
| PE | 6 | 8.2 |

| Curve | # of Wells | % Mean Error* |
|---|---|---|
| RHOB | 491 | 3.4 |
| NPHIL | 291 | 7.5 |
| DTC | 484 | 4.4 |
| PE | 275 | 6.1 |

*Absolute difference relative to log range

## SCALING

- Ran on ~8000 logs from Delaware Basin
- Successfully corrected ~7900 log
- Executed on GCP
- Took 24 hours

# Addressing Conventional Log Data QC Challenges

| Conventional Log Data QC Challenges | Our Solution |
| --- | --- |
| **Time Consuming**<br>• Bad hole detection is time consuming<br>• Bad hole data must be identified for multiple curves across multiple zones | **Time Saving**<br>• The tool predicts values at bad hole flag areas |
| **Insufficient Information**<br>• Some logs do not contain sufficient information | **Efficient Reconstruction**<br>• The tool performs well QC and reconstruction efficiently and accurately |
| **Manual Approach**<br>• Some curves have been digitized from old paper logs and require extensive QC and manual editing | **Machine-Assisted Approach**<br>• The tool allows users to QC, correct, and reconstruct large volumes of well logs |
| **Inadequate Data Usage**<br>• Filtering out too much data to only go with highest quality logs<br>• Making interpretations with poor quality data | **Increasing Data Density**<br>• Expand amount of data that is being processed for petrophysical interpretation<br>• Increase amount of available and interpretable data by an order of magnitude |