CAS RPM Boston, March 2019

Automated Machine Learning for Insurance Applications

Eliade Micu, PhD, FCAS

Director, Research and Development, Clyde Analytics

Presentation Outline

Automated Machine Learning

- Motivation
- Hyperparameter Optimization
- Evolutionary Algorithms
- Feature Creation and Selection
- Component Algorithms

Insurance Applications

- Problem Description
- Results

Conclusions

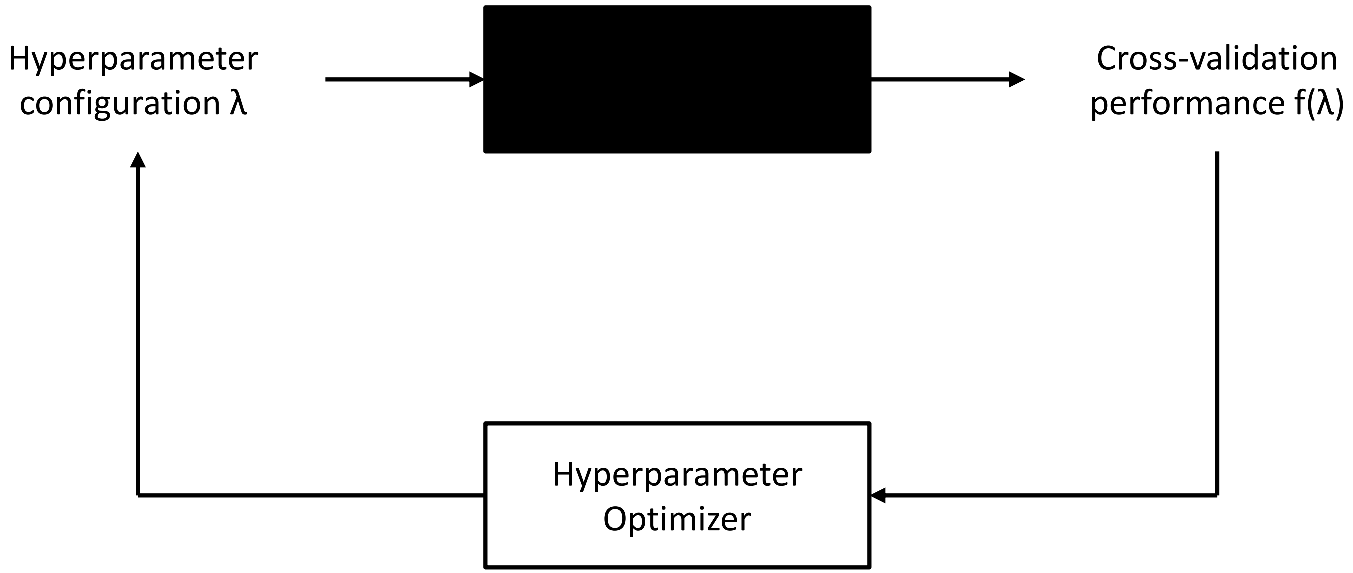
Motivation for Automated Machine Learning

- Demand for machine learning experts has outpaced the supply
- Human machine learning experts perform the following tasks:
  - Preprocess and clean the data
  - *Construct and select appropriate features*
  - *Select an appropriate model family*
  - *Optimize model hyperparameters*
  - *Postprocess machine learning models*
  - Critically analyze the results obtained
- Need user-friendly machine learning software that can be used by non-experts
- Automated Machine Learning attempts to optimize the *inner loop*

Hyperparameter Optimization

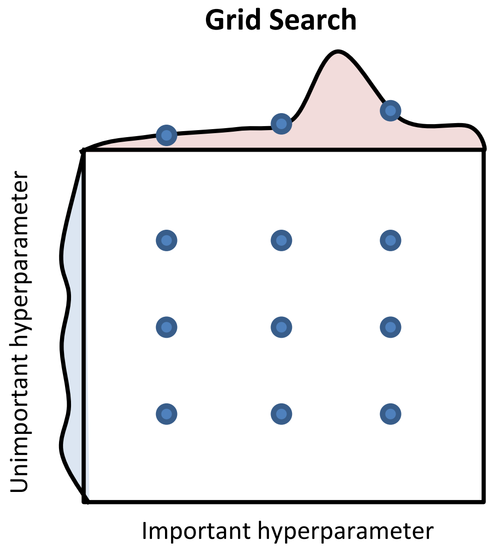
- Hyperparameter types:
  - Continuous (e.g. learning rate for gradient boosting), integer(e.g. number of trees in ensemble)
  - Categorical: unordered, finite domain, e.g. {GBM, Neural Network, Random Forrest}
- Hyperparameter space has structure:
  - Top level parameter A selects algorithm
  - Learning rate parameter λ is active only if A = GBM
  - λ is a conditional hyperparameter with parent A
- Hyperparameters give rise to a structured space of algorithms:
  - Many configurations (e.g. 10^{10})
  - Configurations often yield qualitatively different behavior

Blackbox Hyperparameter Optimization



Optimization Strategy: Random Search

- Select configurations of hyperparameters to test uniformly at random:
  - Completely uninformed
  - Performs global search, will not get stuck in a local optimum
  - Better than grid search



Optimization Strategy: Stochastic Methods

- Stochastic local search:
  - Combines **intensification** and **diversification** steps
  - **Intensification:** gradient descent
  - **Diversification:** restarts, random steps, perturbations
  - Example: Simulated Annealing

- Population based methods:
  - Search is both local and global via the population
  - Maintain population fitness and diversity
  - Examples: Genetic Algorithms, Evolutionary Strategies

Evolutionary Computing

- **Draws inspiration from natural evolution:**

Evolution	Problem Solving
Environment	Problem
Individuals	Candidate solutions
Survival fitness	Solution quality

- **Darwinian Evolution:**
  - Population consists of diverse set of individuals
  - Combinations of traits that are better adapted tend to increase representation in population:

    Individuals are "units of selection"

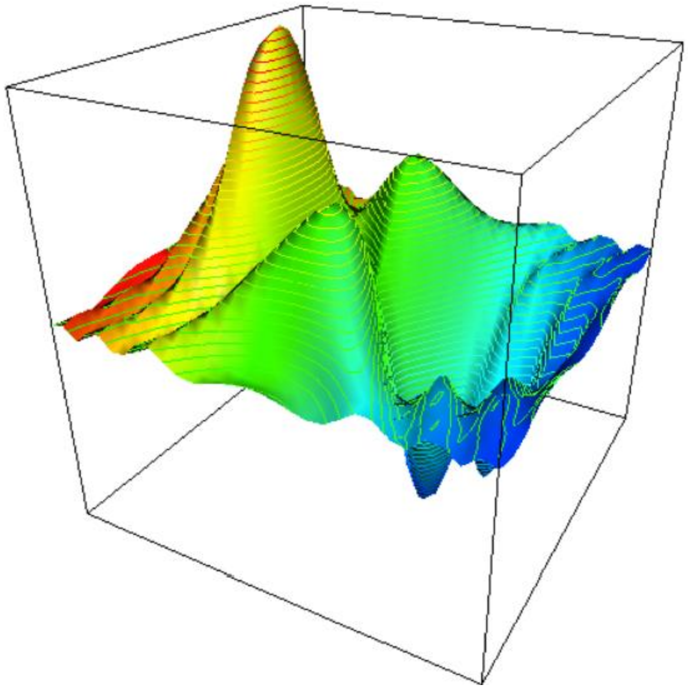
  - Variations occur through random changes yielding constant source of diversity, coupled with selection:

    Population is the "unit of evolution"

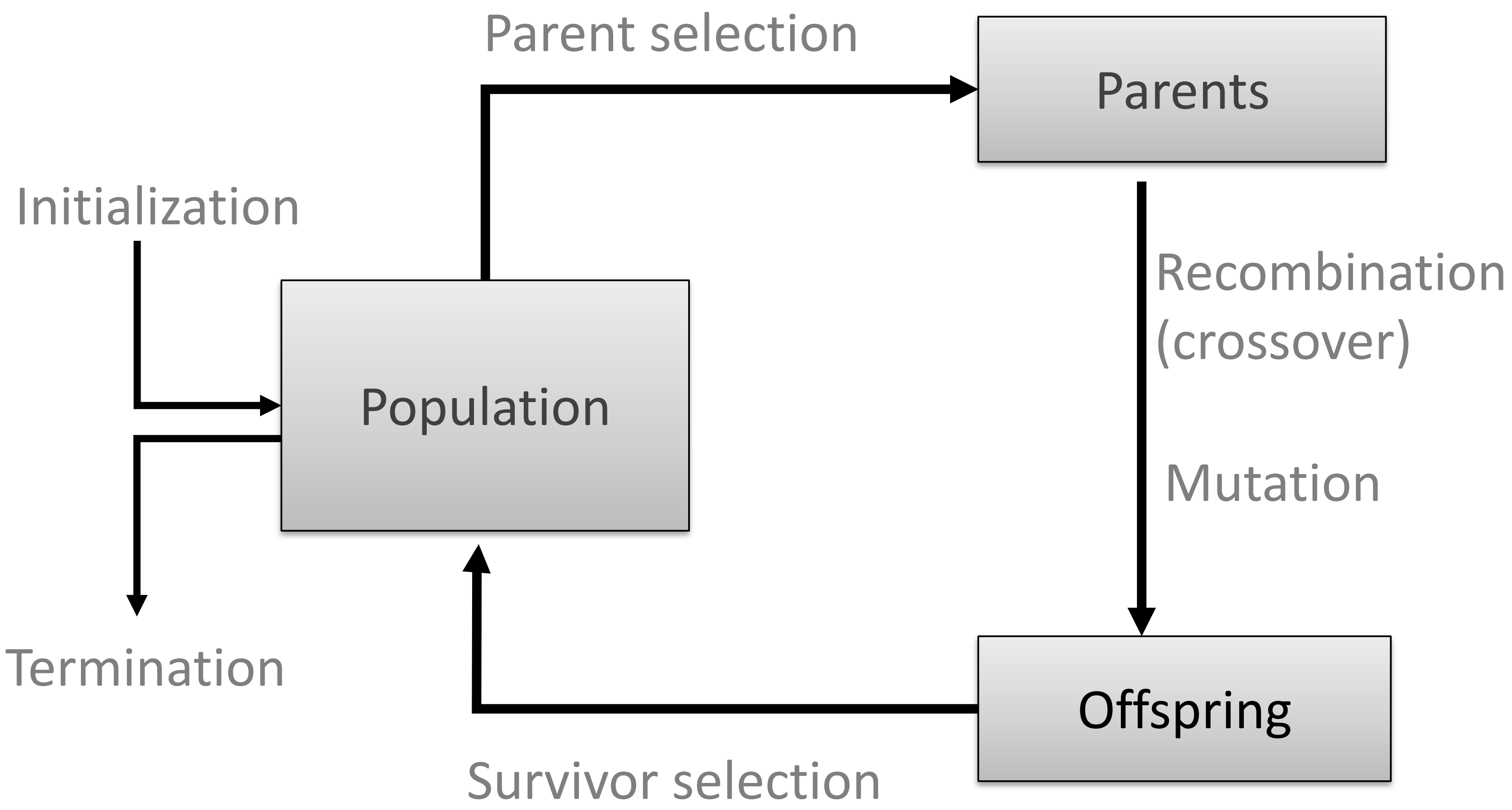
  - There is no "guiding force"

Adaptive Landscape Metaphor (Wright, 1932)

- Population with *n* traits exists in a *n*+1-dimensional space (landscape) with height corresponding to fitness
- Each different individual (phenotype) represents a single point on the landscape
- Population is therefore a “cloud” of points, moving on the landscape over time as it evolves: adaptation
- Selection “pushes” population up the landscape
- Problem is “multimodal”
- Genetic drift:
  - Highly fit individuals may be lost
  - Can cause the population to “melt down” hills, thus crossing valleys and leaving local optima



General Scheme of Evolutionary Algorithms

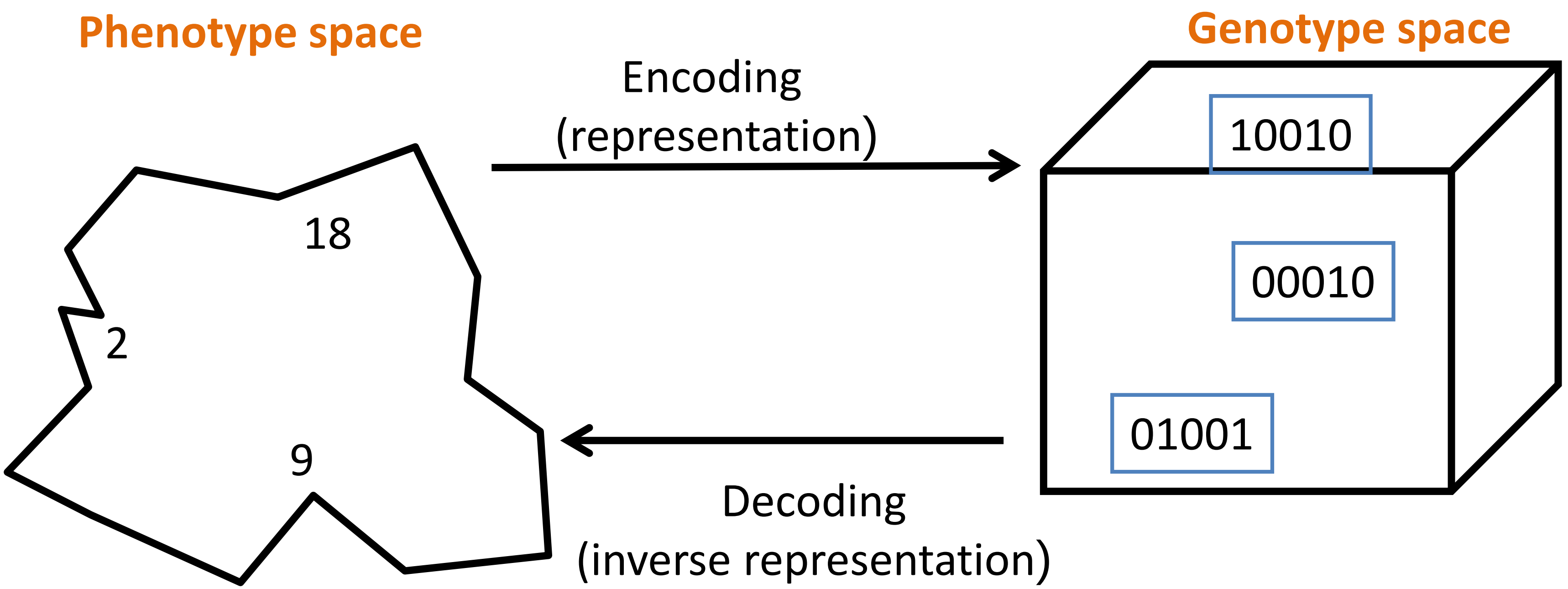


Dialects of Evolutionary Computing

- **Generally differ on candidate solution representation:**
  - Genetic Algorithms (GAs): strings over a finite alphabet
  - Evolution Strategies (EAs): real-valued vectors
  - Classical Evolutionary Programming (EP): finite-state machines
  - Genetic Programming (GP): parse trees
- **One representation may be preferable if it matches problem representation better:**
  - Checkers-playing program: parse trees or finite state machines (EP or GP)
  - Satisfiability problem on *n* variables: bit-strings of length *n* (GA)
- **Variation operators (recombination and mutation) are representation specific**
- **Selection process only takes fitness into account, so independent of representation**

Components of Evolutionary Algorithms

- **Representation (definition of individuals):**
  - Mapping from original objects (phenotypes) to EA objects (genotypes)
  - Whole search takes place in the genotype space
  - Solution is obtained by decoding the best genotype after termination
  - Example: for integer optimization problems, map each integer into its base 2 representation:



Components of Evolutionary Algorithms (cont.)

- **Evaluation (fitness) function:**
  - Represents the task to solve, the requirements to adapt to (can be seen as “the environment”)
  - Enables selection (provides basis for comparison)
  - Assigns a single real-valued fitness to each genotype, which forms the basis for selection, so the more discrimination (different values) the better
- **Population:**
  - Holds the candidate solutions of the problem as individuals (genotypes)
  - Multiset of individuals, i.e. repetitions are possible
  - Population is the basic unit of evolution, i.e., the population is evolving, not the individuals
  - Selection operators act on population level
  - Selection operators usually take whole population into account i.e., reproductive probabilities are relative to current generation
  - Diversity of a population refers to the number of different fitnesses / phenotypes / genotypes present
  - Variation operators act on individual level

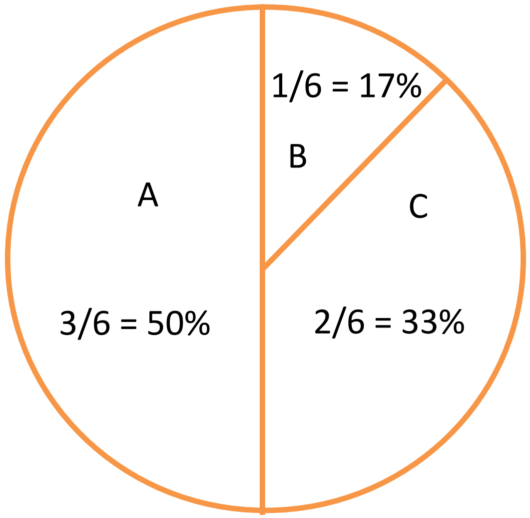
Components of Evolutionary Algorithms (cont.)

- **Parent selection mechanism:**
  - Identifies individuals to become parents
  - Pushes population towards higher fitness
  - Enables selection (provides basis for comparison)
  - Usually stochastic, high quality solutions more likely to be selected than low quality solutions (not guaranteed); even worst fit individual has non-zero probability of being selected
  - Stochastic nature aids in escaping from local optima
  - Example: Roulette wheel selection:

Fitness(A) = 3

Fitness(B) = 1

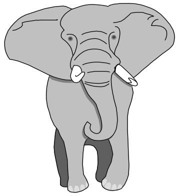
Fitness(C) = 2



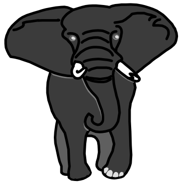
Components of Evolutionary Algorithms (cont.)

- **Variation Operators**
  - Generate new candidate solutions
  - Mutation: causes small, random variance, acts on one genotype and returns another

Before 1 1 1 1 1 1 1



After 1 1 1 0 1 1 1

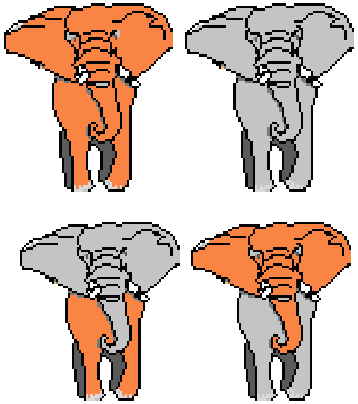


  - Crossover: merges information from parents into offspring

cut | cut

Parents 1 1 1 | 1 1 1 1   0 0 0 | 0 0 0 0

Offspring 1 1 1 0 0 0 0   0 0 0 1 1 1 1



Components of Evolutionary Algorithms (cont.)

- **Survivor selection mechanism (replacement):**
  - Most EAs use fixed population size so need a way of going from parents + offspring to next generation
  - Often deterministic (while parent selection is usually stochastic)
    - Fitness based : rank parents + offspring and take best
    - Age based: make as many offspring as parents and delete all parents
  - Sometimes a combination of stochastic and deterministic (elitism)
- **Initialization:**
  - Usually done at random
  - Can include existing solutions, or use problem-specific heuristics, to “seed” the population
- **Termination condition:**
  - Checked every generation
  - Reaching some maximum allowed number of generations
  - Reaching some minimum level of diversity
  - Reaching some specified number of generations without fitness improvement

Example of Evolutionary Cycle

- **Maximize f(x) = x^2 over the integers 0…31**
  - Use 5-bit binary encoding of integers (phenotypes) into bit-strings (genotypes)
  - Roulette-wheel parent selection (proportional to fitness function value)
  - Replace entire population with the offspring

String No.	Initial Population	Value of x	Fitness $f(x) = x^2$	Probability	Expected Count	Actual Count
1	0 1 1 0 1	13	169	0.14	0.58	1
2	1 1 0 0 0	24	576	0.49	1.97	2
3	0 1 0 0 0	8	64	0.06	0.22	0
4	1 0 0 1 1	19	361	0.31	1.23	1
Sum			1170		4.00	4
Average			293		1.00	1
Max			576		1.97	2

Example of Evolutionary Cycle (cont.)

- **Crossover and offspring evaluation:**

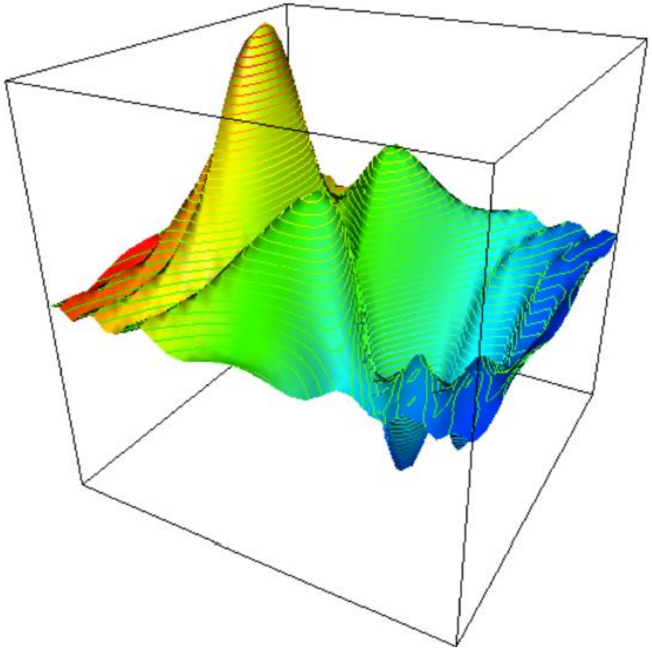
String No.	Mating Pool	Crossover Point	Offspring after xover	x Value	Fitness $f(x) = x^2$
1	0 1 1 0 | 1	4	0 1 1 0 0	12	144
2	1 1 0 0 | 0	4	1 1 0 0 1	25	625
2	1 1 | 0 0 0	2	1 1 0 1 1	27	729
4	1 0 | 0 1 1	2	1 0 0 0 0	16	256
Sum					1754
Average					439
Max					729

- **Mutation and offspring evaluation:**

String No.	Offspring after xover	Offspring after Mutation	x Value	Fitness $f(x) = x^2$
1	0 1 1 0 0	1 1 1 0 0	26	676
2	1 1 0 0 1	1 1 0 0 1	25	625
2	1 1 0 1 1	1 1 0 1 1	27	729
4	1 0 0 0 0	1 0 1 0 0	18	324
Sum				2354
Average				588.5
Max				729

Numerical Optimization: Differential Evolution

- **Differential Evolution:**
  - Storn & Price, 1997
  - Designed to deal with multimodal objective functions, not necessarily continuous or differentiable
  - Population members: n-dimensional real vectors, objective function assumed to be minimized
  - **Differential Mutation**: add a perturbation vector to an existing one
  - Initially designed for unconstrained optimization, can be extended to handle inequality constraints

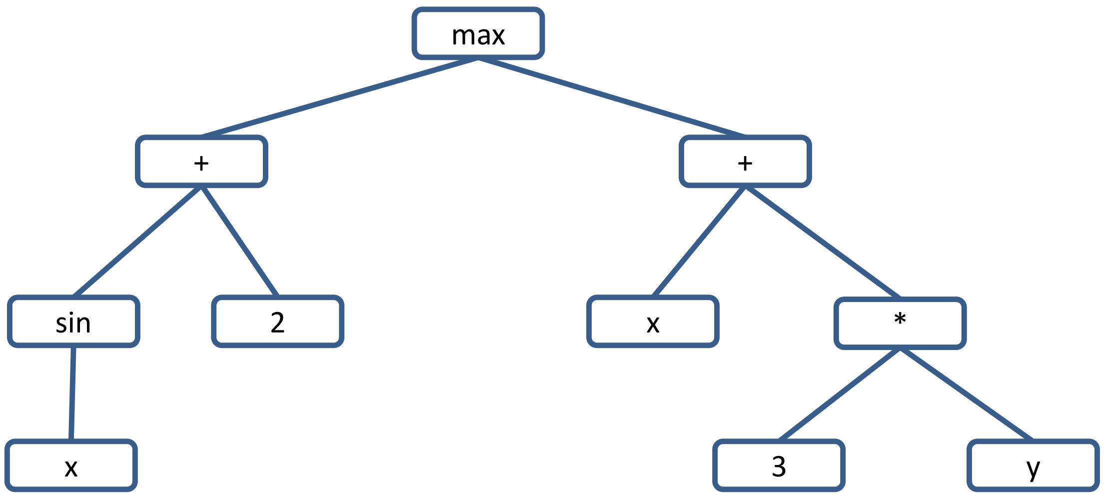


Differential Evolution Description

- **Problem Setup:**
  - Function $f: \mathbb{R}^k \rightarrow \mathbb{R}$ to be minimized
  - Box constraints on the arguments: $x_j \in [a_j, b_j]$ for $j = 1, \dots, k$
- **Population Initialization:**
  - Random: $x_{ij} = a_j + \mathrm{rand}_j[0,1) \cdot (b_j - a_j)$, $j = 1, \dots, k$; $i = 1, \dots, \mathrm{Np}$, where Np = population size
  - If any inequality constraints present, force initial members to be in the feasible region
- **Crossover:**
  - Add a perturbation vector to each base vector: $\mathbf{v}_i = \mathbf{x}_i + \mathrm{F} \cdot (\mathbf{x}_{\mathrm{r}1} - \mathbf{x}_{\mathbf{r}2})$, $i = 1, \dots, \mathrm{Np}$
  - Generate target vector: $\mathbf{u}_{ij} = \begin{cases} \mathbf{v}_{ij} \text{ if } \mathrm{rand}_j[0,1) \leq \mathrm{Cr} \\ \mathbf{x}_{ij} \text{ otherwise} \end{cases}, j = 1, \dots, k.$
- **Selection:** replace $\mathbf{x}_i$ with $\mathbf{u}_i$ in the population if $f(\mathbf{u}_i) \leq f(\mathbf{x}_i)$, keep $\mathbf{x}_i$ otherwise
- **Typical parameter values:** $\mathrm{F} \in [0.5, 1.0]$, $\mathrm{Cr} \in [0.8, 1.0]$, $\mathrm{Np} = 10 \cdot k$

Feature Generation by Genetic Programming

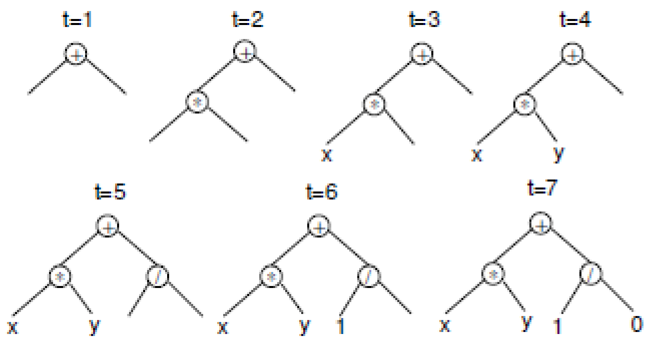
- **Differences from other EA strands:**
  - Positioned as machine learning (as opposed to optimization): seek models with maximum fit
  - Uses **parse trees** as chromosomes (for arithmetic expressions, formulas in predicate logic, or code written in a given programming language)
  - Universe: **set of functions** F = {+, -, *, /, sin, min, max, if, <=, <, >=, >} and **set of terminals** $T = \mathbb{R} \cup \{x, y\}$
  - Example expression: max(sin(x) + 2, x + 3 * y)



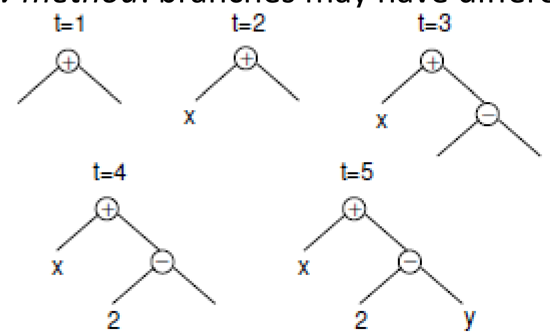
Genetic Programming (cont.)

- **Initialization: ramped half-and-half, combination of full and grow**

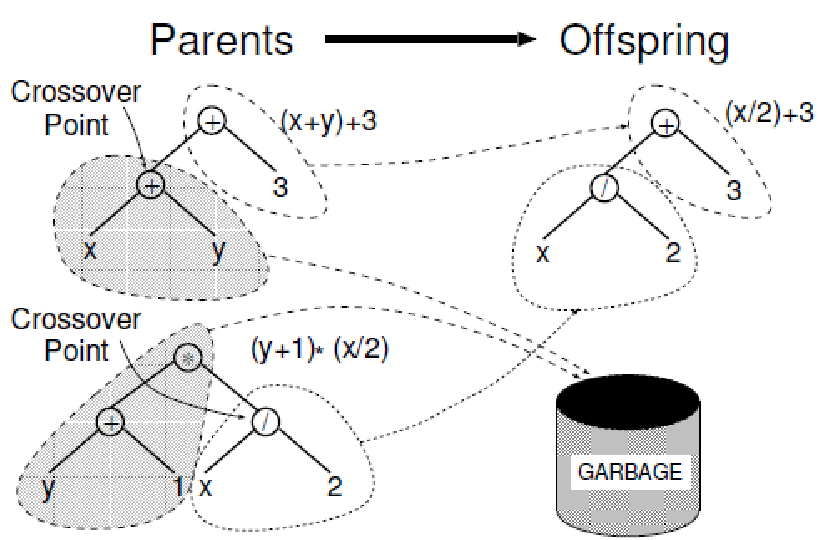
Full method: each tree branch has equal length



Grow method: branches may have different lengths

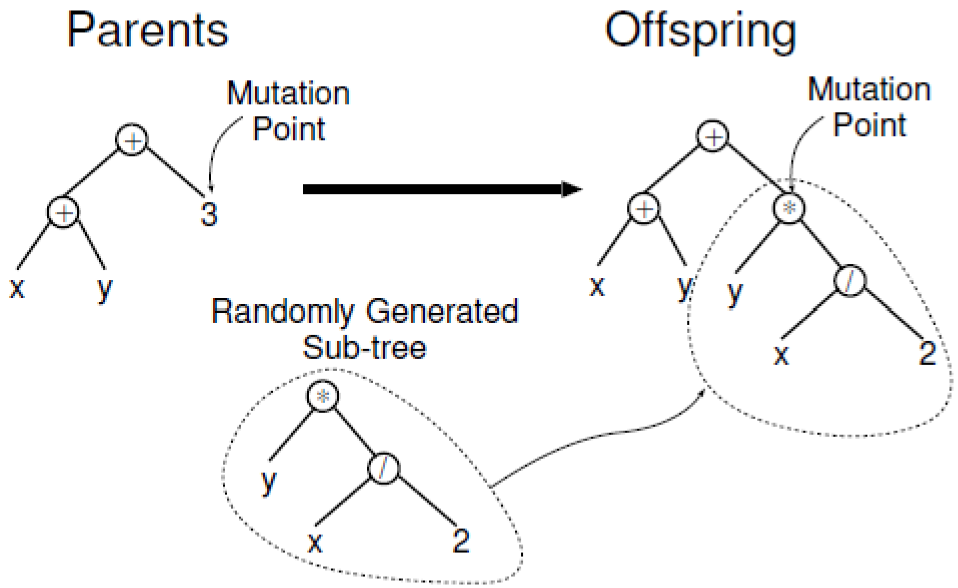


- **Crossover:**



Genetic Programming (cont.)

- **Mutation:**



- **Fitness Function (Symbolic Regression example):**
  - Given a set of n observations $(x_1, y_1, z_1), \ldots, (x_n, y_n, z_n)$ find a function $f(x, y)$ that approximates z
  - Minimize $err(f) = \sum_{i=1}^{n}(f(x_i, y_i) - z_i)^2$

Symbolic Regression Example

- 5M records, PPA data for 5 main coverages
- Model response: loss ratio; Model weights: earned fitted pure premium
- Function Set = {+, -, *, /, exp, abs, if}, Terminals = 20 numerical predictors + real constants
- Fitness function: Gini, Population size: 100, Evolution steps: 200
- Model trained on 60% of data chosen at random, validated on remaining 40%
- Sample expression (Individual #1, best Gini):

SAFE_EVAL(EXP((LOYALTY_MOD - 8.98)/7.769 - ((coll_vrg_curr - 23.247)/9.929 - (ab_vrg_curr - 32.32)/6.299)) * (EXP((LOYALTY_MOD - 8.98)/7.769 - ((dc_vrg_curr - 23.247)/9.929 - (ab_vrg_curr - 32.32)/6.299)) * EXP(0.742900070070423 - (TOT_VEH - 1.813)/0.947)))

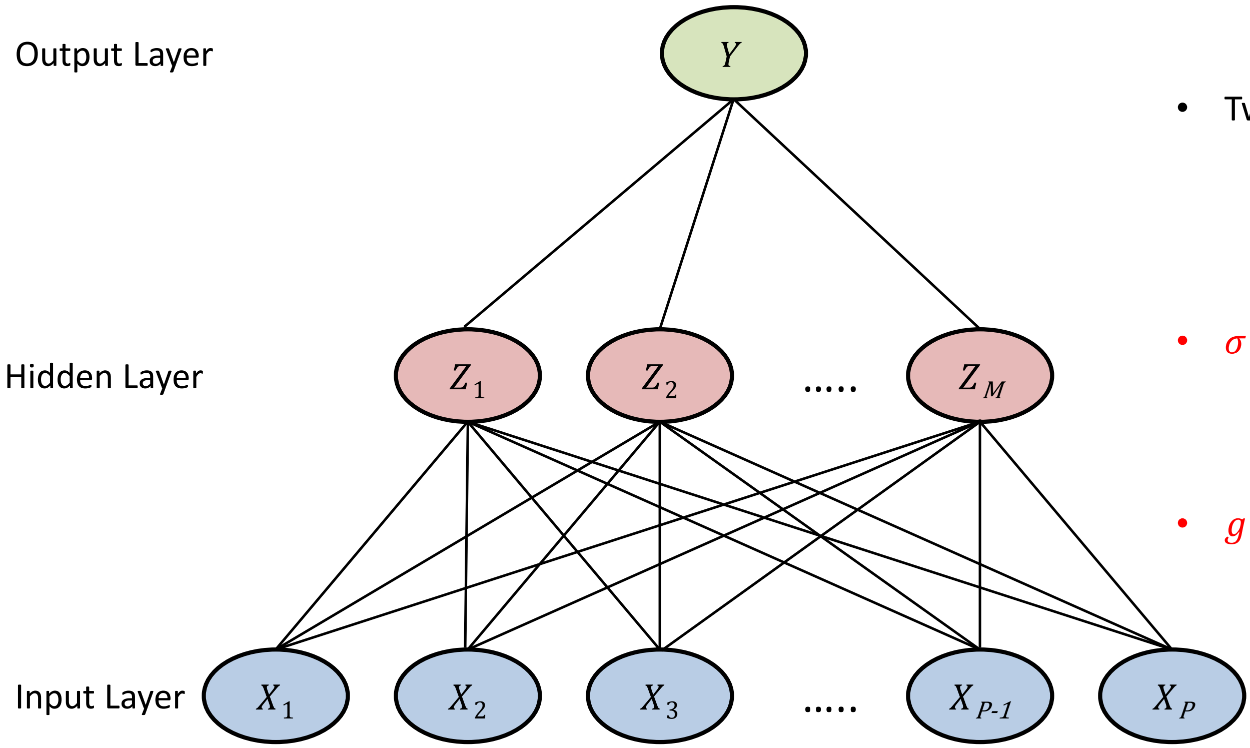
Individual (Top 10 Elite)	Lift	Validation Lift	Gini	Validation Gini	Correlation
1	2.00	2.03	0.106	0.106	99.37%
2	2.00	2.05	0.105	0.104	99.12%
3	2.05	2.07	0.105	0.105	98.39%
4	2.05	2.07	0.105	0.105	98.38%
5	2.05	2.07	0.105	0.105	98.39%
6	2.05	2.07	0.105	0.105	98.36%
7	2.05	2.07	0.105	0.105	98.37%
8	2.05	2.07	0.105	0.105	98.38%
9	1.89	1.97	0.103	0.107	97.51%
10	1.91	1.83	0.101	0.100	98.00%

AutoML Component Algorithms

- Penalized GLMs (ridge, lasso, elastic net)
- Neural Networks
- Ensembles:
  - Combines weak base learners that come from the same class, such as trees
  - Bagging: averaging predictions of weak learners trained independently on subsets of the data
  - Boosting: summing predictions of weak learners trained sequentially on modified versions of the data
- Stacked models (**Super Learners**):
  - Combines strong, diverse sets of learners together
  - Trains a second-level “metalearner” to find the optimal combination of the base learners

Neural Networks

- Hyperparameters: M (number of nodes in hidden layer), σ, g, λ (regularization strength), n (number of hidden layers)

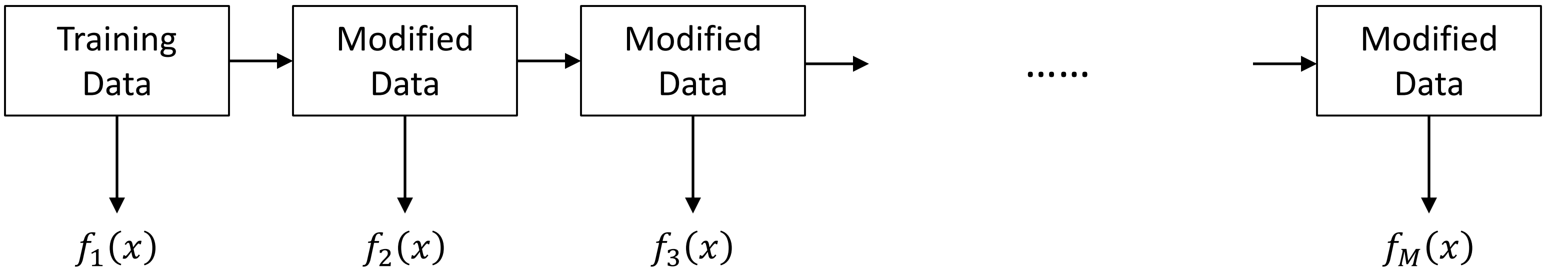


- Two-stage regression or classification:
  - $Z_m = \sigma(\gamma_{0m} + \gamma_m^T X), m = 1, \ldots, M$
  - $Y = f(X) = g^{-1}(\beta_0 + \beta^T Z)$
- σ is the **activation function**:
  - Sigmoid
  - Hyperbolic tangent
- g is the **link function**
  - Log or identity for regression
  - Logit for classification

Fitting Neural Networks

- ***Model parameters (complete set of network weights θ):***
  - $\{\gamma_{0m}, \gamma_m : m = 1, \dots, M\}$ $\rightarrow M(P+1)$ weights
  - $\{\beta_0, \beta\}$ $\rightarrow M+1$ weights
  - Neural networks can approximate any continuous function with an arbitrary degree of precision by increasing M
- ***Error function:***
  - $R(\theta) = \sum_{i=1}^{n}(y_i - f(x_i))^2$ - sum of squared errors or deviance (regression)
  - $R(\theta) = -\sum_{i=1}^{n} y_i \log f(x_i)$ - cross-entropy (classification)
- ***Optimize penalized error $R(\theta) + \lambda \cdot P(\theta)$ to prevent overfitting:***
  - $P(\theta) = \sum_m \beta_m^2 + \sum_{ml} \gamma_{ml}^2$ - quadratic
  - $P(\theta) = \sum_m |\beta_m| + \sum_{ml} |\gamma_{ml}|$ - linear
  - $P(\theta) = \sum_m \frac{\beta_m^2}{1+\beta_m^2} + \sum_{ml} \frac{\gamma_{ml}^2}{1+\gamma_{ml}^2}$ - elimination
- ***GLMs (and penalized GLMs) are a special case of neural network:***
  - Deviance as the error function
  - Identity as the activation function
  - One node in the hidden layer

Gradient Boosting for Regression Trees



- Hyperparameters: M (number of component trees), k (size of component trees), λ (learning rate)
- For $m = 1, 2, \dots, M$ do:
  - For each observation $i = 1,2, \dots, n$ compute *pseudo residuals:*

$$r_{im} = y_i - f_{m-1}(x_i)$$

  - Fit “weak” learner (regression tree with k terminal nodes) to r_{im} giving regions $R_1, R_2, \dots, R_k$
  - For $j = 1,2, \dots, k$:

    α_j = observed average for region R_j

  - Update $f_m(x) = f_{m-1}(x) + \lambda \cdot \sum_{j=1}^{k} \alpha_j \cdot I(x \in R_j)$
- Final model: $\hat{f}(x) = f_M(x)$

Super Learner Algorithm

- Set up the ensemble:
  - Specify a list of L base algorithms (with a specific set of hyperparameters for each)
  - Specify a **metalearning** algorithm, e.g. GLM with positive weights, GBM, NN, etc.
- Train the ensemble:
  - Train each of the L base algorithms on the training set
  - Perform k-fold cross-validation on each of these learners and collect the cross-validated predicted values
  - Combine the N cross-validated predicted values from each of the L algorithms into a N x L matrix, to create the **level-one** data (N = number of rows in the training set)
  - Train the metalearning algorithm on the level-one data, with the same response as the L base algorithms
- Predict on new data:
  - Generate predictions from the L base learners
  - Feed those predictions into the metalearner to generate the ensemble prediction.

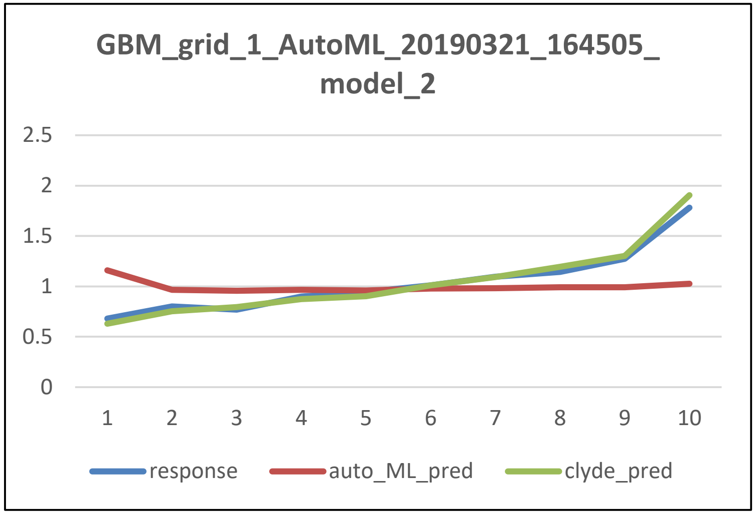
Insurance Application

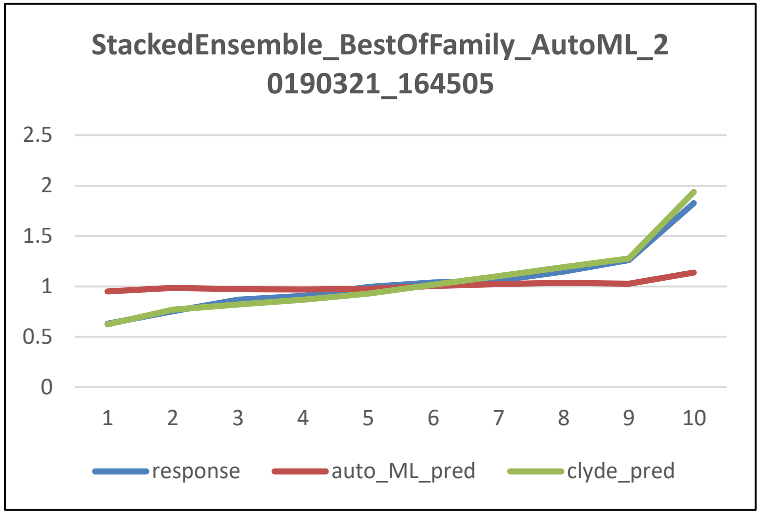
- **$H_2$0.ai** machine learning framework:
  - "Open source, in-memory, distributed, fast, and scalable"
  - Core written in Java, can be used from R or Python
  - AutoML component algos: Penalized GLMs (elastic net), Random Forests, Extremely Randomized Trees, GBM, Multi-layer NN (deep learning), Stacked Ensembles
  - Candidate models are scored using 5-fold cross validation deviance
- Human expert:
  - Component algos: GLMs, customized versions of single-layer NN and boosted trees
  - Pipeline: GLM, followed by single-layer NN on GLM residuals, followed by boosted trees on NN residuals
- PPA COLL dataset, 7M records, 35 predictors:
  - 60/40 train/validation split
  - Model Weights: $EEXP \cdot pred\_GLM$
  - Model Response: $Observed\_Loss \, / \, (EEXP \cdot pred\_GLM)$

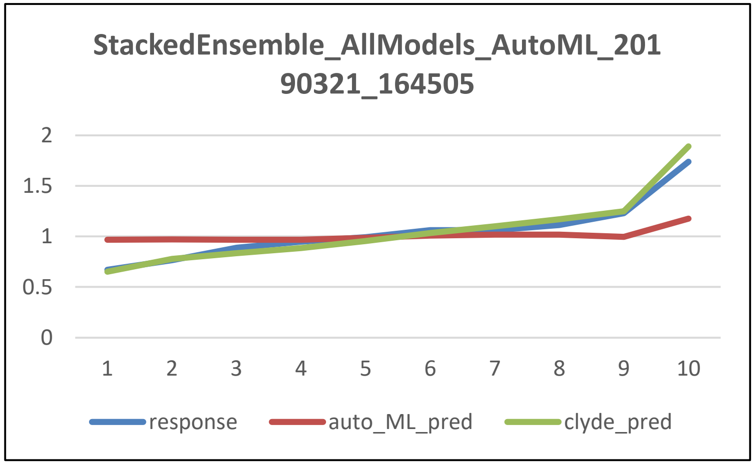
AutoML Leaderboard

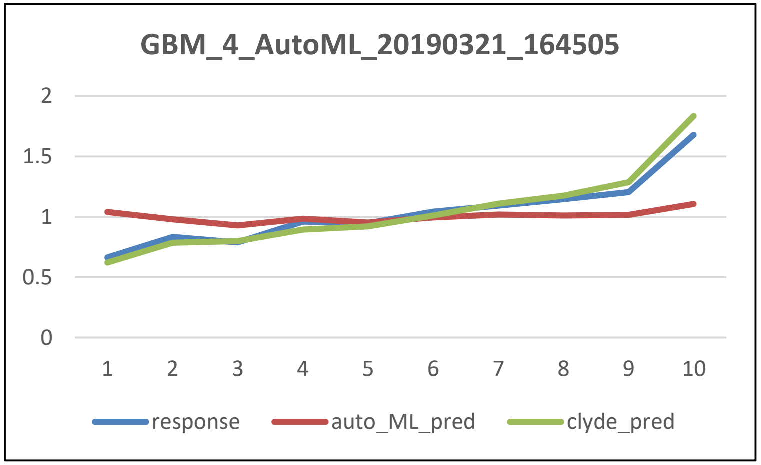
Model	Lift Train	Gini Train	Lift Valid	Gini Valid
XRT_1_AutoML_20190321_164505	538.99	0.65	1.26	0.05
DRF_1_AutoML_20190321_164505	1111.97	0.63	1.20	0.04
GBM_grid_1_AutoML_20190321_164505_model_2	35.22	0.49	1.88	0.12
GBM_5_AutoML_20190321_164505	28.05	0.37	1.42	0.08
GBM_grid_1_AutoML_20190321_164505_model_7	21.63	0.32	1.26	0.06
StackedEnsemble_BestOfFamily_AutoML_20190321_164505	6.12	0.29	2.20	0.12
StackedEnsemble_AllModels_AutoML_20190321_164505	5.47	0.27	2.21	0.13
GBM_4_AutoML_20190321_164505	6.49	0.21	2.11	0.13
GBM_grid_1_AutoML_20190321_164505_model_8	3.99	0.20	1.31	0.07
GBM_3_AutoML_20190321_164505	4.18	0.18	2.27	0.13
GBM_2_AutoML_20190321_164505	3.84	0.17	2.32	0.13
GBM_1_AutoML_20190321_164505	3.35	0.16	2.54	0.13
GBM_grid_1_AutoML_20190321_164505_model_6	3.84	0.14	1.20	0.11
GLM_grid_1_AutoML_20190321_164505_model_1	2.25	0.12	2.16	0.12
DeepLearning_grid_1_AutoML_20190321_164505_model_8	1.30	0.05	1.24	0.04
DeepLearning_grid_1_AutoML_20190321_164505_model_1	1.39	0.04	1.31	0.04
DeepLearning_grid_1_AutoML_20190321_164505_model_3	1.07	0.01	1.15	0.02
DeepLearning_grid_1_AutoML_20190321_164505_model_2	1.08	0.00	1.21	0.02
DeepLearning_grid_1_AutoML_20190321_164505_model_7	1.07	0.00	1.09	0.01
DeepLearning_1_AutoML_20190321_164505	0.71	0.00	0.80	0.01
Clyde_NN	3.01	0.17	2.35	0.14
Clyde_NN_Boosted_Tree	2.85	0.17	2.50	0.14

AutoML Double Lift Test

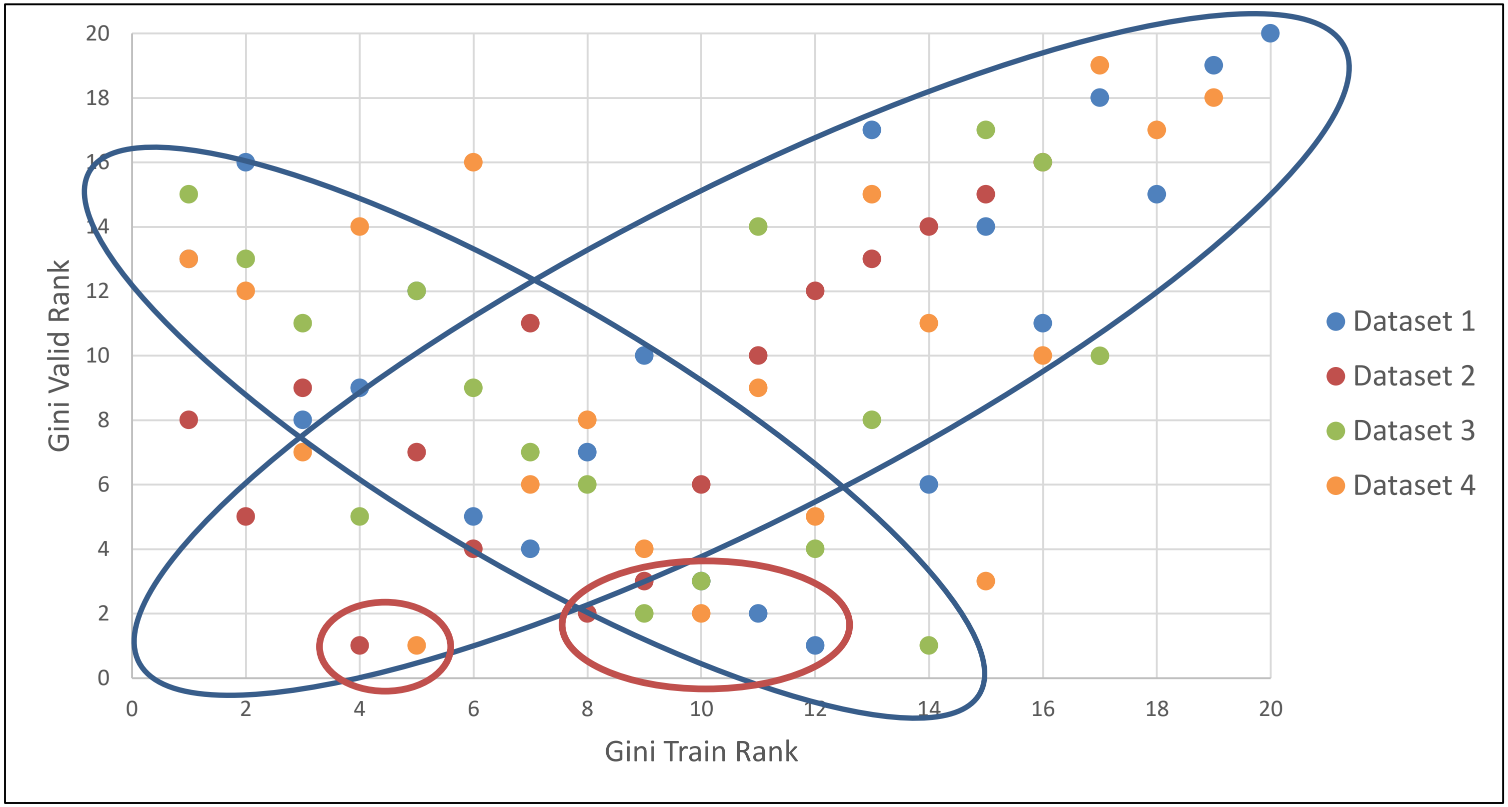








AutoML Generalization Performance



Sample	Gini Rank Correlation	Max Gini Valid	Line of Business	Model Response
Dataset 1	38.50%	0.13	PPA COLL	Loss Ratio
Dataset 2	68.24%	0.54	Comm Prop	Pure Prem
Dataset 3	-5.64%	0.07	HO All Perils	Loss Ratio
Dataset 4	46.75%	0.13	PPA All Coverages	Loss Ratio

Conclusions

- AutoML can successfully perform the following tasks:
  - *Construct and select appropriate features*
  - *Select an appropriate model family*
  - *Optimize model hyperparameters*
  - *Postprocess machine learning models*
- Relatively easy to use out of the box, decent default settings for some algorithms, such as GBM
- Produced (some) models with good performance
- Human expert still needed to inspect results for reasonableness and select the final model
- AutoML generates a large number of hypotheses, danger of “overfitting the validation data”
- AutoML performance depends on difficulty of problem, e.g. ground-up vs. residual analysis
- When in doubt, and with no prior knowledge about the domain, select a “middle-performing” model, not top models, to ensure better expected generalization performance
- Human experience plus customized algorithms in a custom pipeline can outperform AutoML