IBM **Research AI**

Automated Machine Learning and Data Science (AMLDS)

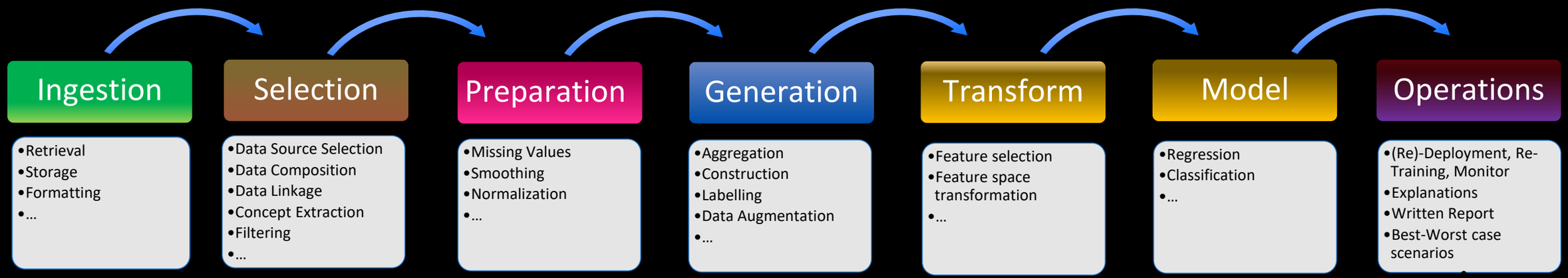
https://ibm.biz/auto-ml

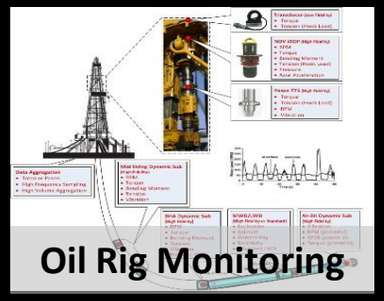
Automated Feature Engineering for Predictive Modeling

Udayan Khurana, Horst Samulowitz, Fatemeh Nargesian (University of Toronto),
Tejaswini Pedapati, Elias Khalil (Georgia Tech), Gregory Bramble, Deepak Turaga, Peter Kirchner

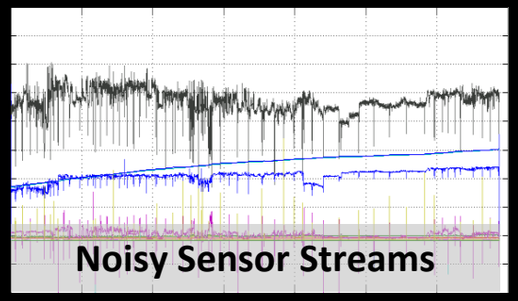
Saket Sathe @ ICDM
"*Kernel-Based Feature Extraction For Collaborative Filtering*"
[also has a very nice paper at KDD-17: "*Similarity Forests*"]

Data Science Workflow

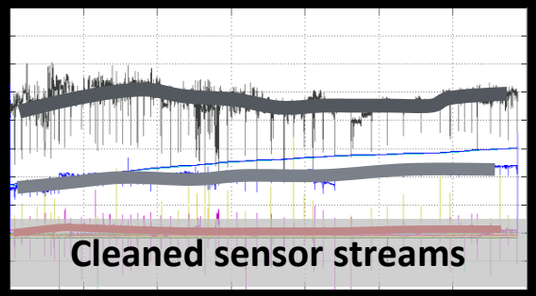




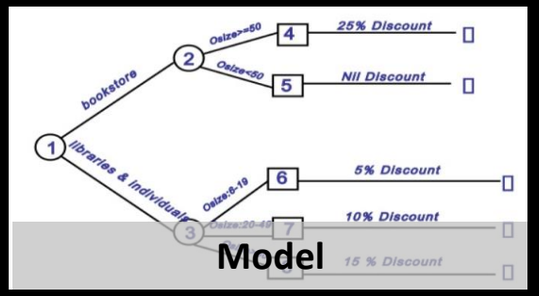
Oil Rig Monitoring



Noisy Sensor Streams

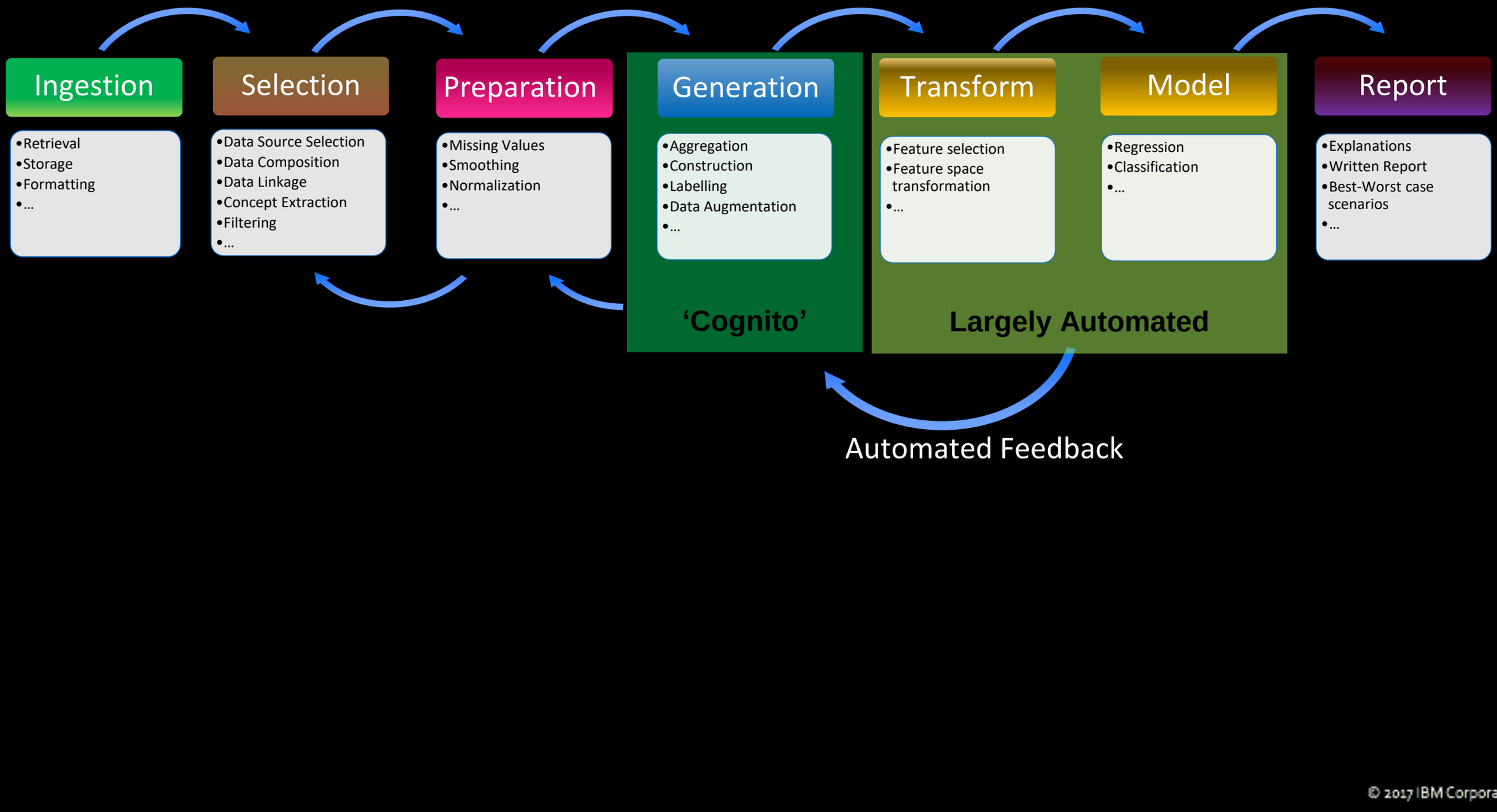


Cleaned sensor streams



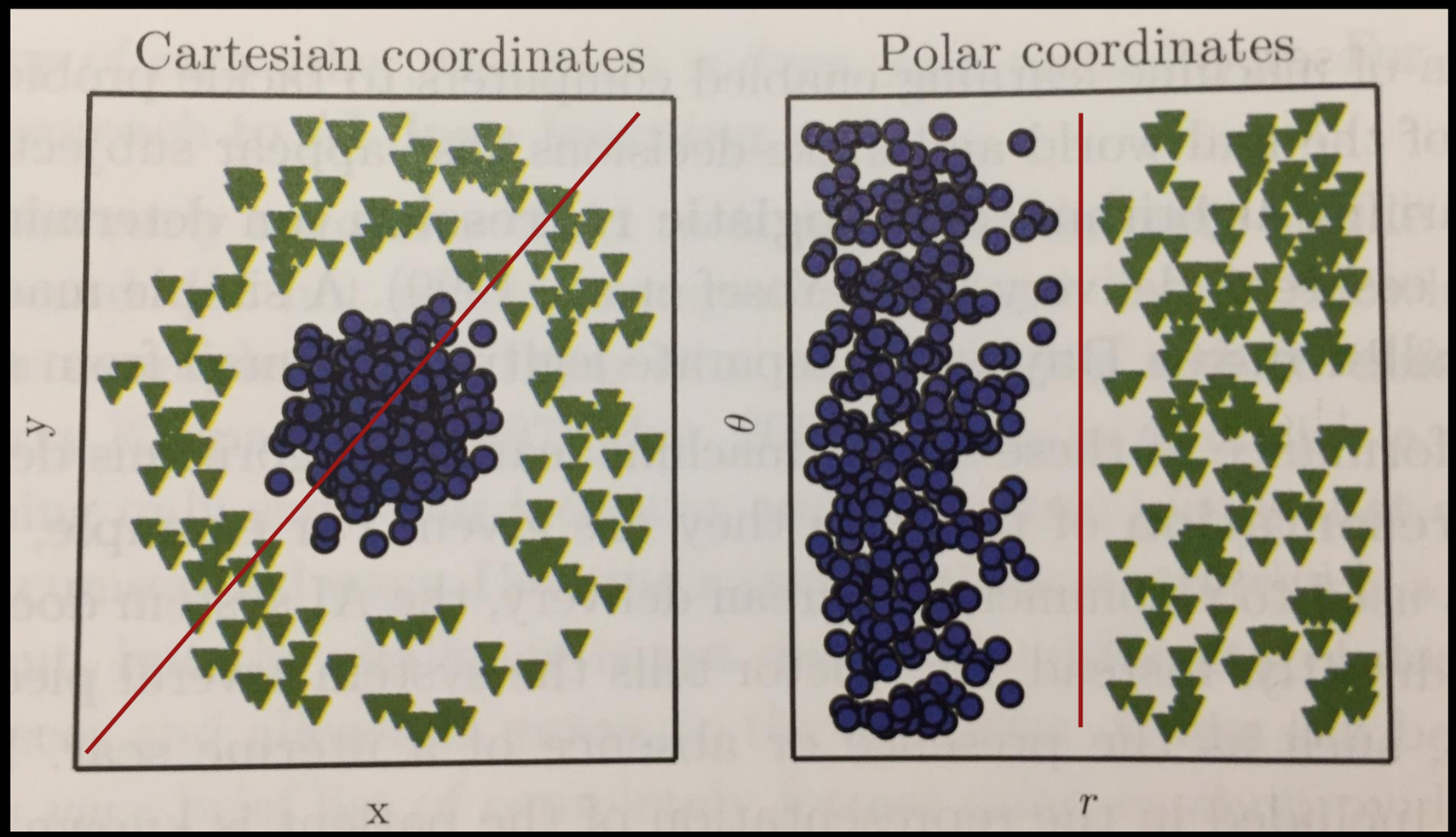
Model

Data Science Workflow – and its Automation



Feature Representation

- Why does feature representation matter?
  - Consider building a classifier using a straight line



picture source: Deep Learning by Goodfellow et al.

Feature Engineering Example

- Problem: Kaggle DC bikeshare rental prediction
- Regression with Random Forest Regressor
- Original features (**relative abs. error = 0.61**) :

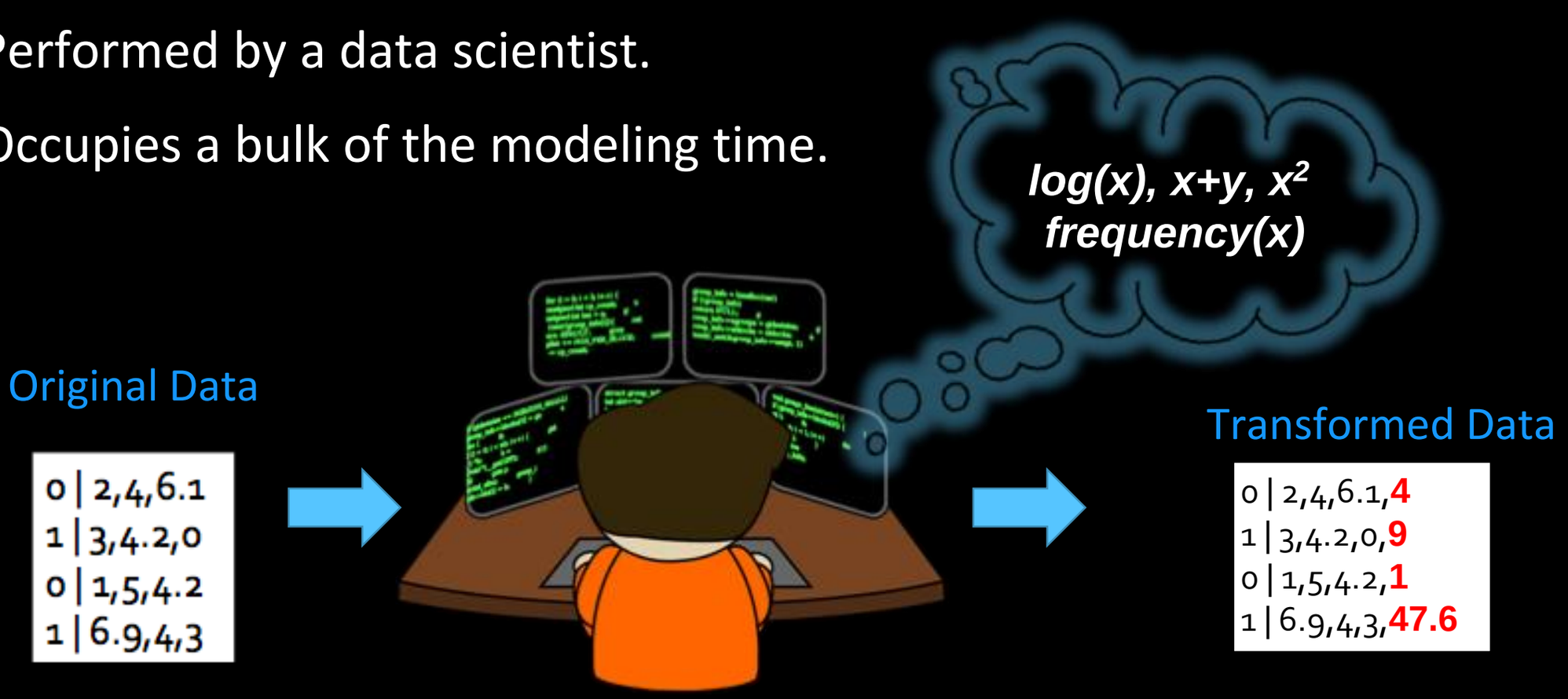
datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	count
2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0	16
2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0	40
2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0	32

- Added features (**relative abs. error = 0.20**) :

year[datetime]	hrofday[datetime]	Log[humidity]	Log[windspeed]	RC[hrofday[datetime]]	RC[Log[hrofday[datetime]]]	ZSCORE[datetime]
2011	0	4.394449	0	0	0	-1.711228
2011	1	4.382027	0	1	0	-1.711030
2011	2	4.382027	0	0.5	1.442695	-1.710832

Overview: Feature Engineering

- Feature Engineering describes the transformation of a given dataset's feature space:
  - In order to improve learning accuracy.
  - Through generating new features, removing unnecessary ones.
  - Performed by a data scientist.
  - Occupies a bulk of the modeling time.



Ways of Feature Engineering

(1) Data scientist's **expertise** gives a hunch on which transformations to try.

log(x), x+y, x^2
frequency(x)

Original Data

0	2,4,6.1
1	3,4.2,0
0	1,5,4.2
1	6.9,4,3

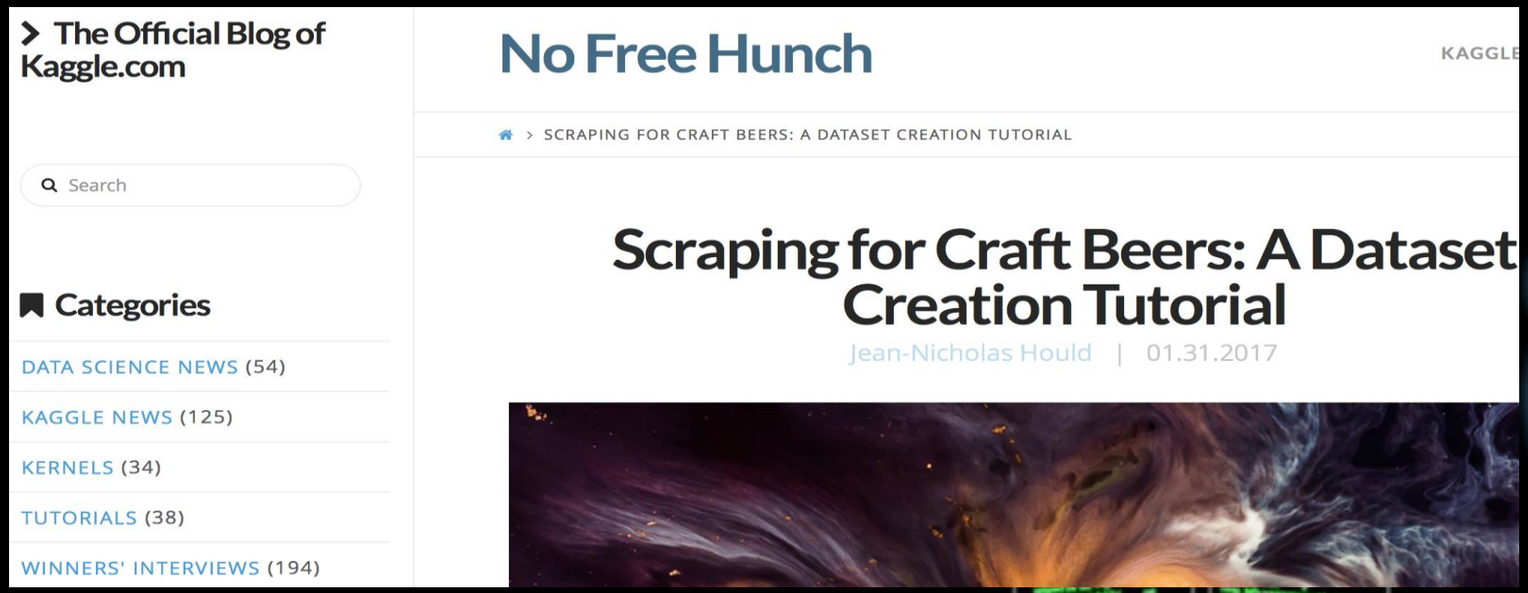
Transformed Data

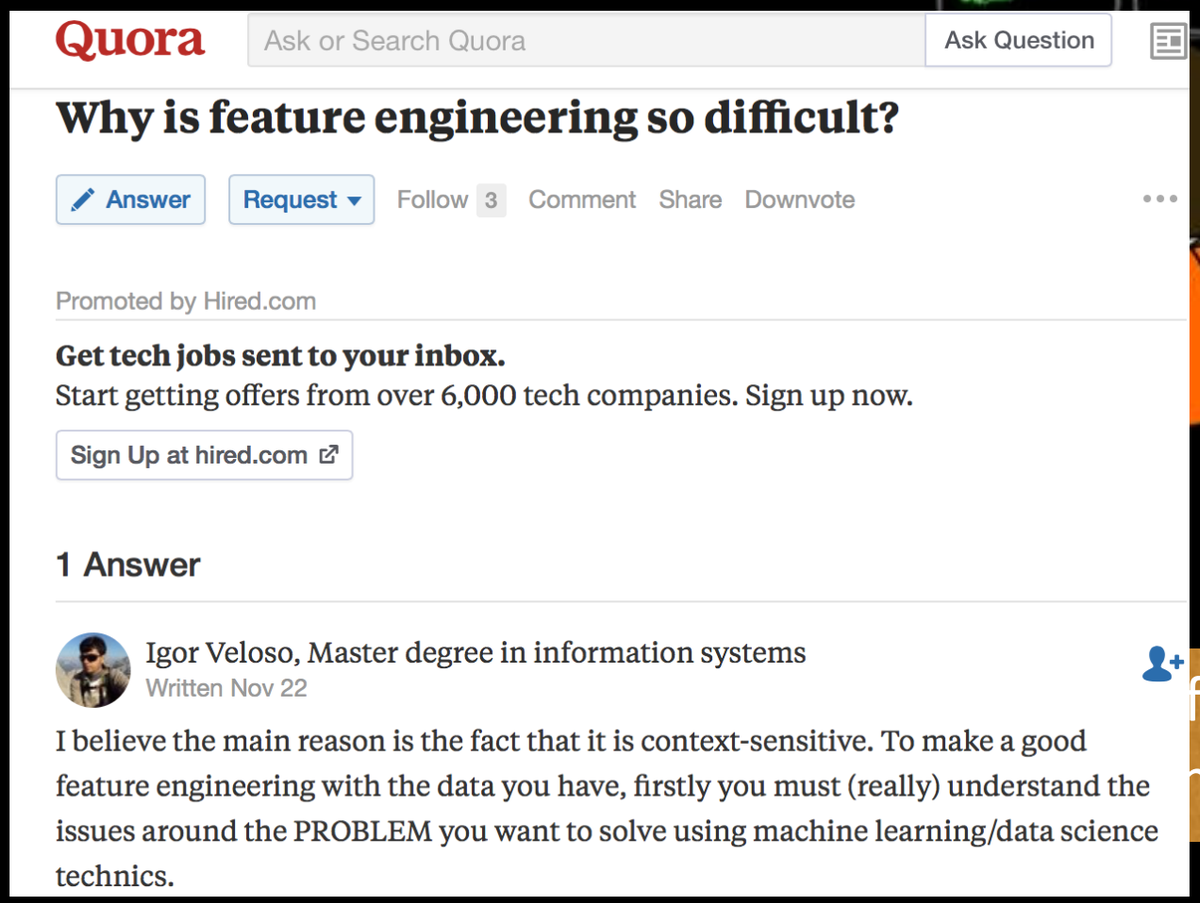
0	2,4,6.1,**4**
1	3,4.2,0,**9**
0	1,5,4.2,**1**
1	6.9,4,3,**47.6**

(2) Performed iteratively through **trial and error.**

Model Building and Validation

Ways of Feature Engineering





log(x), x+y, x^2
frequency(x)

Transformed Data

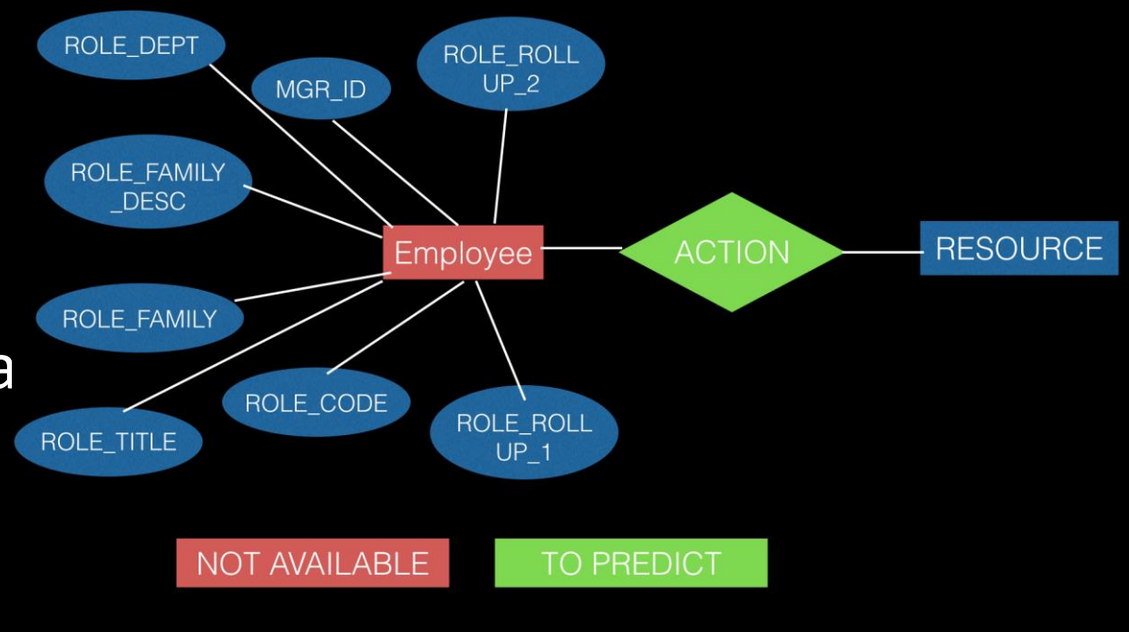
0 | 2,4,6.1,**4**
1 | 3,4.2,0,**9**
0 | 1,5,4.2,**1**
1 | 6.9,4,3,**47.6**

formed iteratively
trial and error.

Model Building and Validation

How to get a good hunch?

- Make a theoretical model
  - Consult an expert in the domain.
  - Build hypothesis and verify them with data
  - Come up with data enhancement options



Entity Relationship diagram for Amazon Resource Kaggle challenge dataset

- Limitations
  - Dependent on human effort
  - Expensive and time consuming
  - Data and theory are different
  - Dataset may not be descriptive

Feature Engineering and its Automation

- Introduction
- Problem Definition and Complexity
- Performance driven methods
  - Hierarchical function based approach
  - Reinforcement Learning based policy-learning

    ["Feature Engineering for Classification using Reinforcement Learning", AAAI'2018]
- Learning Feature Engineering

  ["Learning Feature Engineering for Classification, IJCAI'2017]
- Combined Demo: "Dataset Evolver"

  https://www.youtube.com/watch?v=4T8KaeOn-2Y
- Automated Model Selection
- Automated Neural Network Composition

Problem Definition

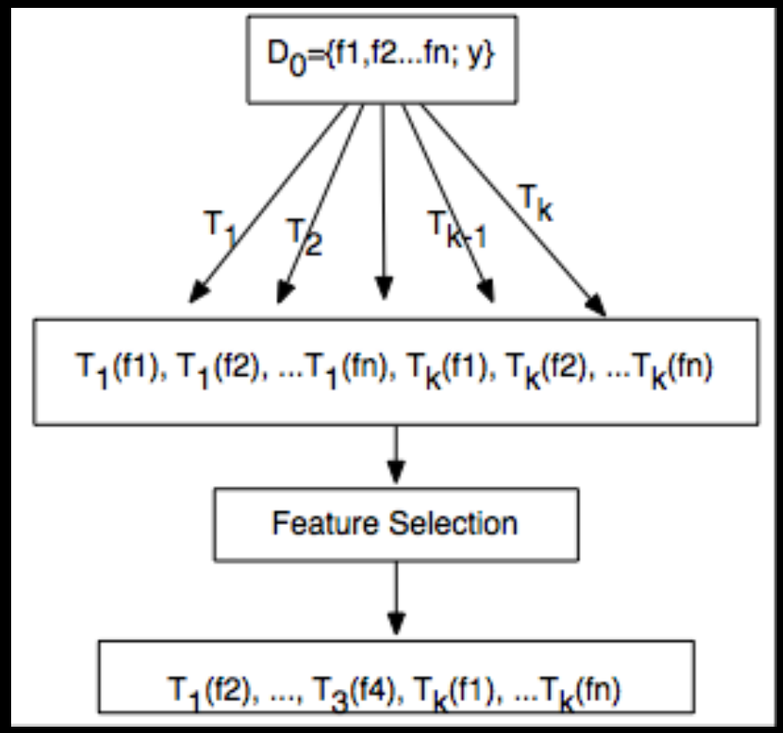
- Given a predictive modeling task:
  - Set of *m* features: **F =** {$f_1, f_2 \ldots f_m$}
  - target vector: ***y***
  - a modeling algorithm, M
    - $A_M(F,y)$ reflects model performance
    - ***k*** transform functions: $t_1, t_2, \ldots t_k$
- A sequence of transformations **s** = $t^{(1)}(t^{(2)} \ldots (f_i))$
- *Problem:* Find a set of sequences of transformations **S**={$s_1,..s_r$}
  - $\mathbf{F_{new}}$ = **F'** + **S**, where **F'** is a subset of **F**
  - $\text{argmax}_{(\mathbf{S})}\ P_M(\mathbf{F_{new}}, \boldsymbol{y})$

Complexity and Brute force

- For k features and r_1 = unary transforms, d = depth
  - $s_1 = (k * r_1)^{d+1}$
  - For k=10, r_1 =10, d=5; $s_1 = 10^{12}$
- For r_2 binary transforms
  - $s_2 = (C_{(k,2)} * r_2))^{d+1}$
  - For k=10, r_2 =10, d=5; $s_2 = 10^{15}$
- For each case, verification involves training and evaluating a model
- It is clearly computationally infeasible to verify all possibilities

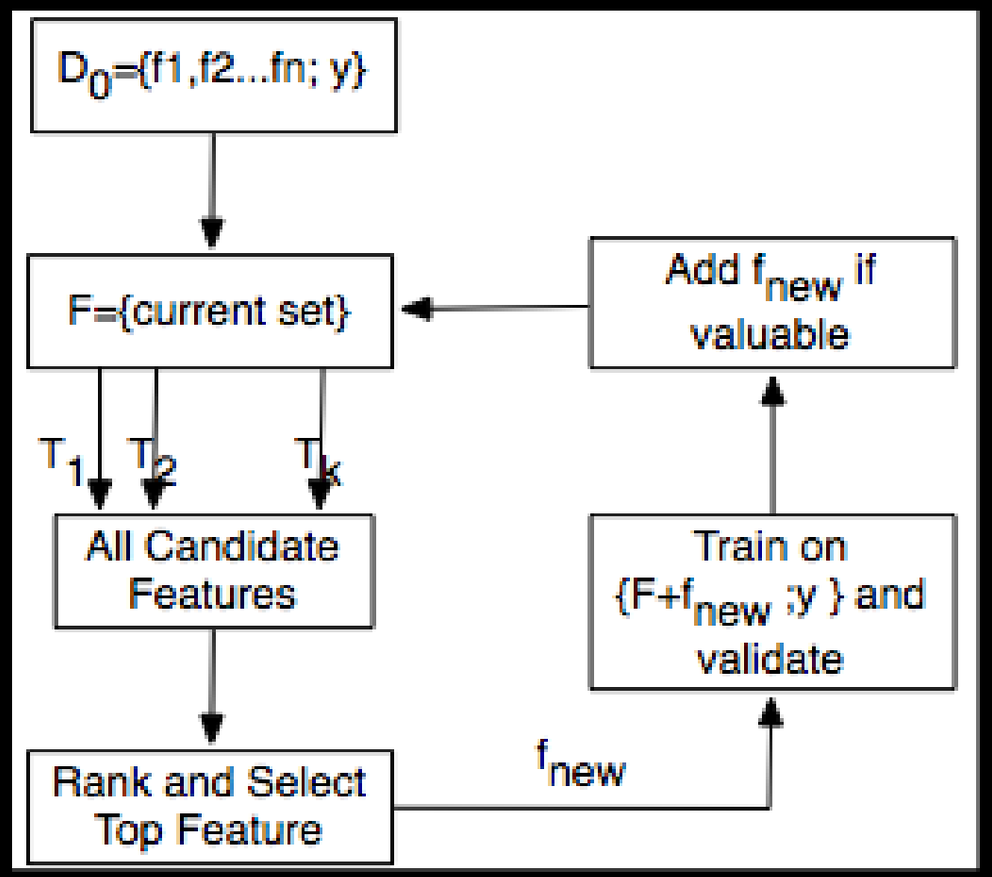
Existing approaches…

- *Expand-Reduce*
- DSM [Kanter et al. DSAA 2015], OneBM [arXiv 2017]
  - Applies all transforms at once {$f_1, f_2 \ldots f_m$} x {$t_1, t_2, \ldots t_k$} => (m x k) features
  - Followed by a *feature selection (FS)* step and parameter tuning
  - Positive: One modeling step (excluding FS)
  - Limitation: Doesn't consider compositions of functions
  - Limitation: FS is a performance bottleneck due to large (m x k) features



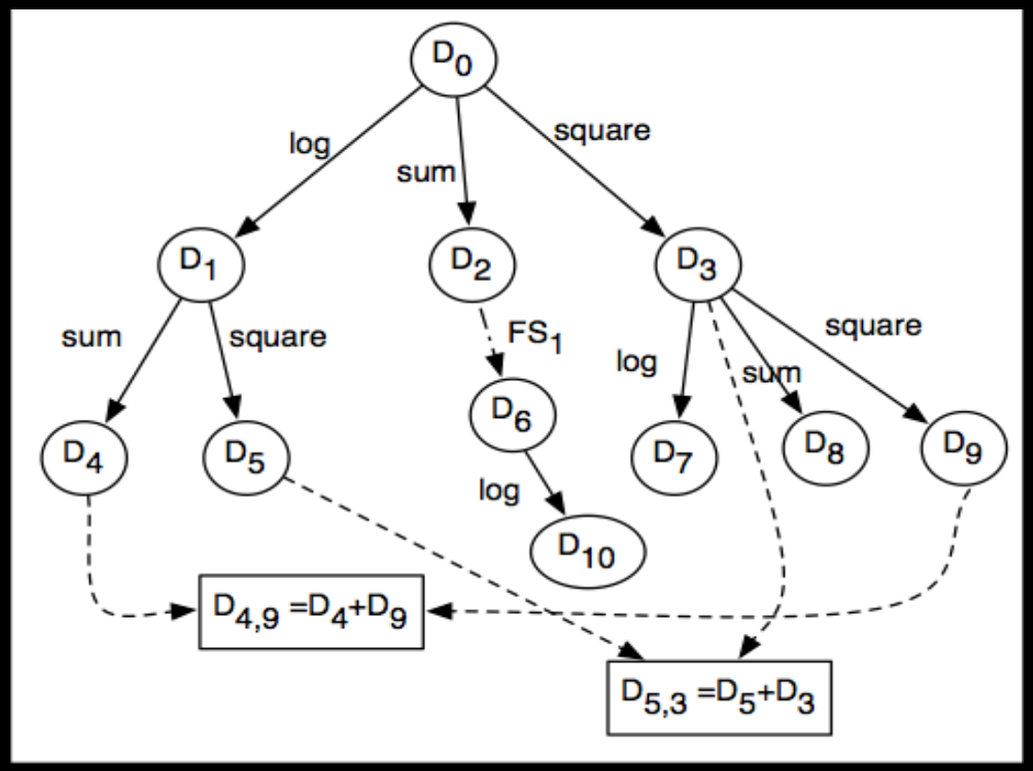
Existing approaches…

- *Evolution-centric*
  - ExploreNet [Katz et al. ICDM 2016]
  - Adds one feature at a time and performs model building and verification.
  - Runs in a greedy manner until time runs out
  - Positive: More scalable than expand-reduce method
  - Limitation: Extremely Slow



Hierarchical organization of transformations

- Transformation **T** applied to feature set applies function to all valid input features and appends new columns to the existing ones
- Exploration using search strategy guided by performance accuracy under a constrained budget



Example of a Transformation Graph, which is a directed acyclic graph. The start node D_0 corresponds to the given dataset; that and the hierarchical nodes are circular. The sum nodes are rectangular. In this example, we can see three transformations, log, sum, and square, as well as a feature selection operator FS_1.

Khurana et al.: Cognito: Automated Feature Engineering for Supervised Learning [ICDM '16]

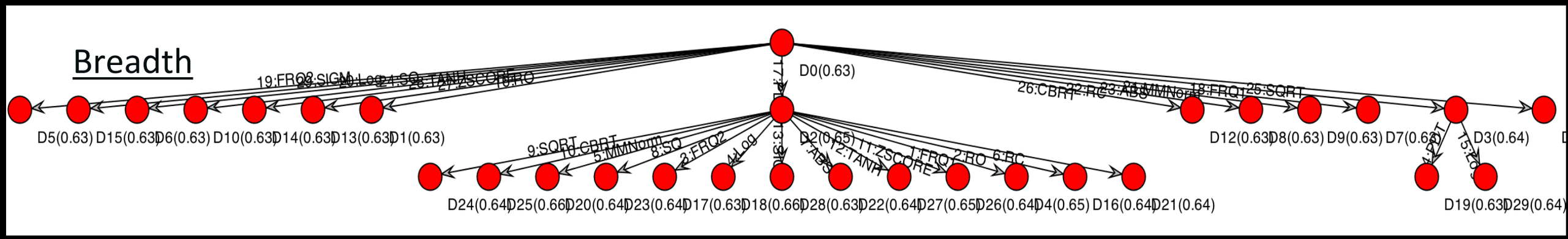
Selected factors influential in policy decisions

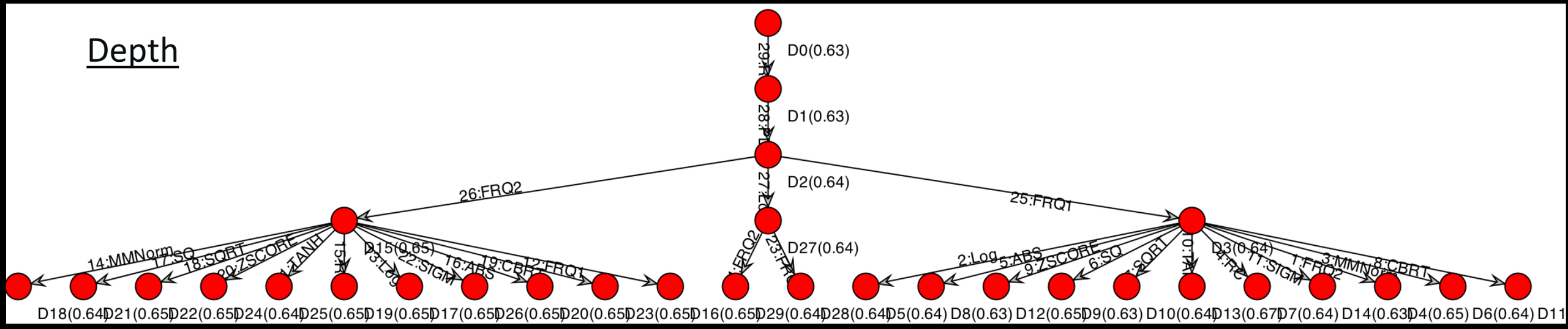
- Node n ’s Accuracy: Higher accuracy of a node incentivizes further exploration from that node, compared to others.
- Transformation, *t* ’s average performance until G_i : Using t ’s mean performance in the transformation graph so far, we compare potential gains from it, compared to others
- Frequency of a transform in the path from root node to *n* : can be used to discount the potential gains from application of t if it has already been applied to a descendant of n .
- Accuracy gain for *n’s parents* : While n ’s accuracy itself is a factor, so is the gain from its parent (and the same for its parent), indicating the focus on more promising regions of the graph.
- Node Depth: A higher value is used to penalize the relative complexity of the transformation sequence (overfitting).
- Remaining budget fraction: The budget is measured by the number of maximum nodes allowed for the graph. At each step, the fraction of remaining budget is a factor in determining the trade-off in exploration versus exploitation.
- Ratio of feature counts in *n* to the original dataset: This indicates the bloated factor of the dataset.
- Is transformation a feature selector or not (augmenter)?

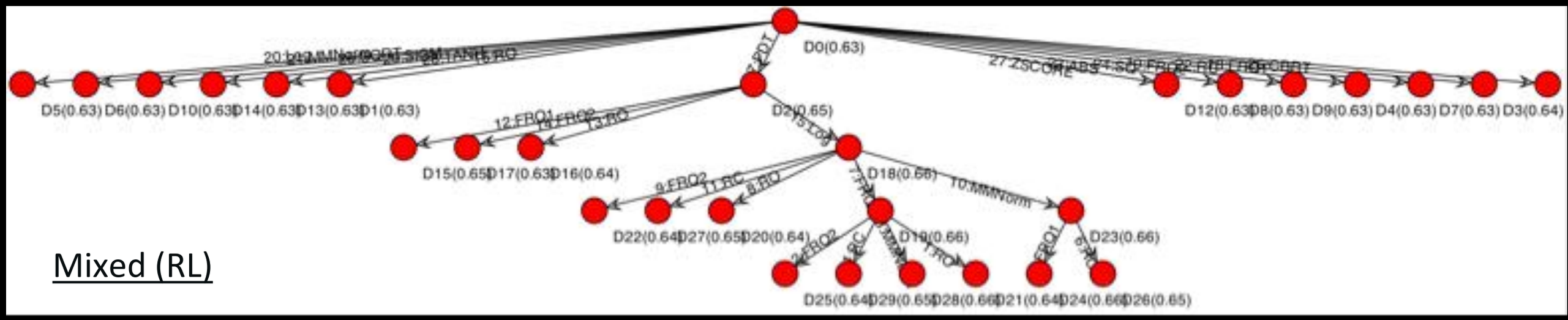
Hierarchical organization of transformations

- Emulates a human trial and error process
- Performance Driven Traversal Strategies:
  - Depth Oriented
  - Breadth Oriented
  - Mixed (budgeted)
  - Reinforcement learning-based (next)
- Advantages:
  - Allows composition of transforms
  - Batching improves performance
  - Data-level transformations are logical blocks for measuring performance
- Demo:
  - **Cognito:** https://www.youtube.com/watch?v=hJlG0mvynDo

Examples of different policies







Mixed (RL)

Policy Learning with Reinforcement Learning

- Strategy learned with experience over various datasets.
- Consider it is a *Markov Decision Process (MDP)*.
- A state (snapshot) of TG is a state of MDP.
  - A state is described by an array of TG factors and remaining budget.
  - We learn transitions from one state to another
- Objective in learning:
  - Short term goal: Balance exploration and exploitation.
  - Final Goal: maximize the final delta in accuracy

RL Modeling

- Immediate reward

$$r_i = \max_{n' \in \theta(G_{i+1})} A(n_{i+1}) - \max_{n \in \theta(G_i)} A(n_i)$$

- Cumulative reward

$$R(s_i) = \sum_{j=0}^{B_{max}} \gamma^i . r_{i+j}$$

- Q-function

$$Q(s,c) = r(s,c) + \gamma R^{\Pi}(\delta(s,c))$$

- Optimal Policy:

$$\Pi^*(s) = \arg\max_c [Q(s,c)]$$

- Approximation of Q-function:

$$Q(s,c) = w^c . f(s)$$

- Update rule for w_c:

$$w^{c_j} \leftarrow w^{c_j} + \alpha . (r_j + \gamma . \max_{n',t'} Q(g',c') - Q(g,c)) . f(g,b)$$

System Overview

Preprocessing

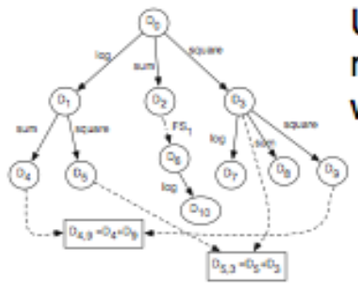
Preparation
Type inference, format conversion, missing values completion

EDA
Explores correlations, feature ranks, outliers, redundant columns, infers entities

Tagging
Tags column based upon inferences, types and correlations

Feature Engineering

Explores feature space in Transformation Graph combination of construction and selection options through reinforcement learning.



Utilizes off the shelf model selection to work with a few good models.

Employs sampling and estimation to scale by further pruning the graph

DataStore
Non-redundant storage of data versions

Model Cache

Post Feature Engineering Exploration

Trains models on a diverse set nodes based on exploration of transformation graphs. Ensembles are constructed based on combinations of such models and the best overall is suggested for deployment.

{ ○ ○ ○ ○ ○ } ***Nodes***

X

{ RF LR DT NB } ***Learning Algos***

M_1 M_2 M_3 ..... M_k ***Models***

E_1 E_r ***Ensembles***

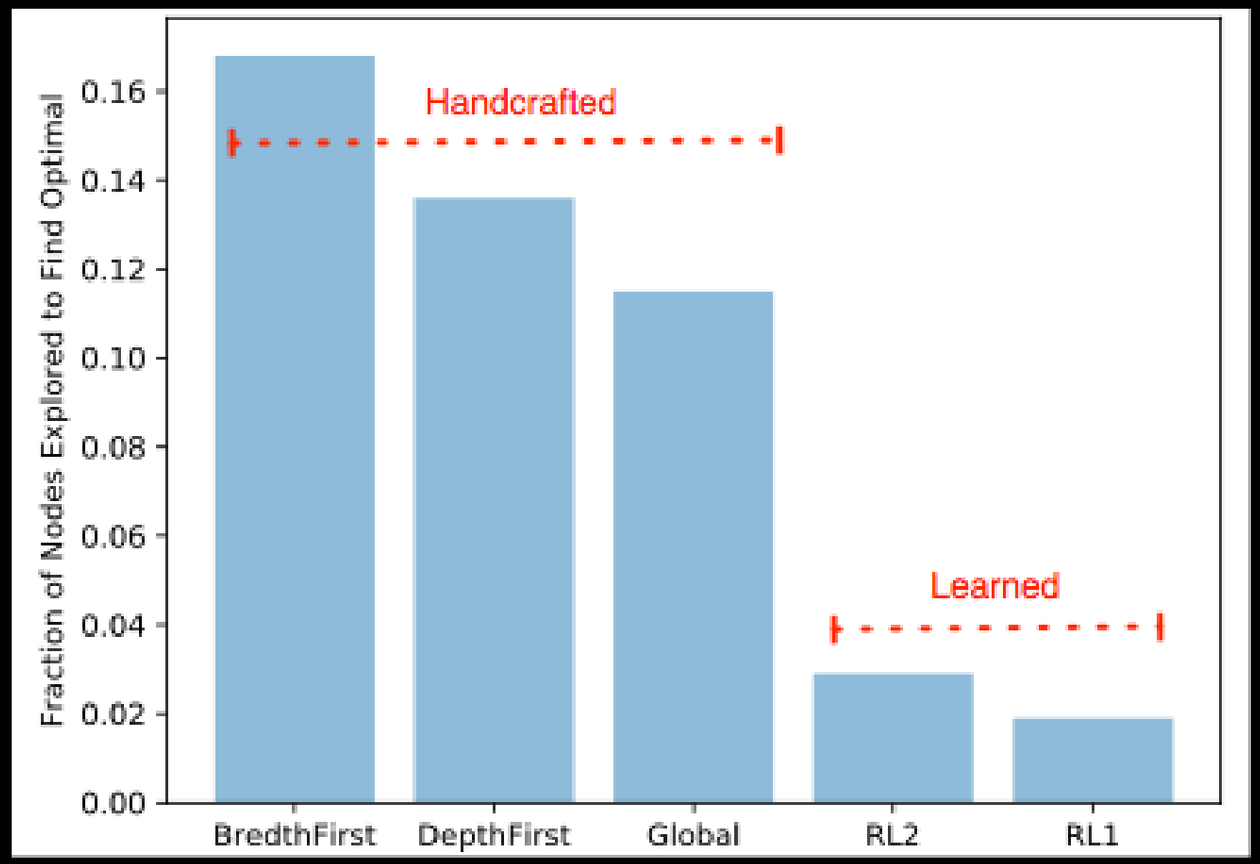
Results

Dataset	Source	C/R	Rows	Features	Base	Cognito RL	DSM FE	Random FE	Cognito Rule
Higgs Boson	UCIrvine	C	50000	28	0.718	**0.729**	0.682	0.699	0.702
Amazon Employee	Kaggle	C	32769	9	0.712	**0.806**	0.744	0.740	0.714
PimaIndian	UCIrvine	C	768	8	0.721	**0.756**	0.712	0.709	0.732
SpectF	UCIrvine	C	267	44	0.686	0.788	**0.790**	0.748	0.780
SVMGuide3	LibSVM	C	1243	21	0.740	**0.776**	0.711	0.753	**0.766**
German Credit	UCIrvine	C	1001	24	0.661	**0.724**	0.680	0.655	0.662
Bikeshare DC	Kaggle	R	10886	11	0.393	**0.798**	0.693	0.381	0.790
Housing Boston	UCIrvine	R	506	13	0.626	**0.680**	0.621	0.637	0.652
Airfoil	UCIrvine	R	1503	5	0.752	**0.801**	0.759	0.752	0.794
AP-omentum-ovary	OpenML	C	275	10936	0.615	**0.820**	0.725	0.710	0.758
Lymphography	UCIrvine	C	148	18	0.832	**0.895**	0.727	0.680	0.849
Ionosphere	UCIrvine	C	351	34	0.927	**0.941**	0.939	0.934	**0.941**
Openml_618	OpenML	R	1000	50	0.428	**0.587**	0.411	0.428	0.532
Openml_589	OpenML	R	1000	25	0.542	**0.689**	0.650	0.571	0.644
Openml_616	OpenML	R	500	50	0.343	**0.559**	0.450	0.343	0.450
Openml_607	OpenML	R	1000	50	0.380	**0.647**	0.590	0.411	0.629
Openml_620	OpenML	R	1000	25	0.524	**0.683**	0.533	0.524	0.583
Openml_637	OpenML	R	500	50	0.313	**0.585**	0.581	0.313	0.582
Openml_586	OpenML	R	1000	25	0.547	**0.704**	0.598	0.549	0.647
Credit Default	UCIrvine	C	30000	25	0.797	**0.831**	0.802	0.766	0.799
Messidor_features	UCIrvine	C	1150	19	0.691	**0.752**	0.703	0.655	0.762
Wine Quality Red	UCIrvine	C	999	12	0.380	**0.387**	0.344	0.380	0.386
Wine Quality White	UCIrvine	C	4900	12	0.677	**0.722**	0.654	0.678	0.704
SpamBase	UCIrvine	C	4601	57	0.955	**0.961**	0.951	0.937	0.959

Comparing accuracy between base dataset (no FE), Our FE, DSM inspired FE, Random FE, and Cognito using 24 datasets. Performance here is unweighted average FScore for classification and (1 rel. absolute error) for regression.

Results

Various Search Policies

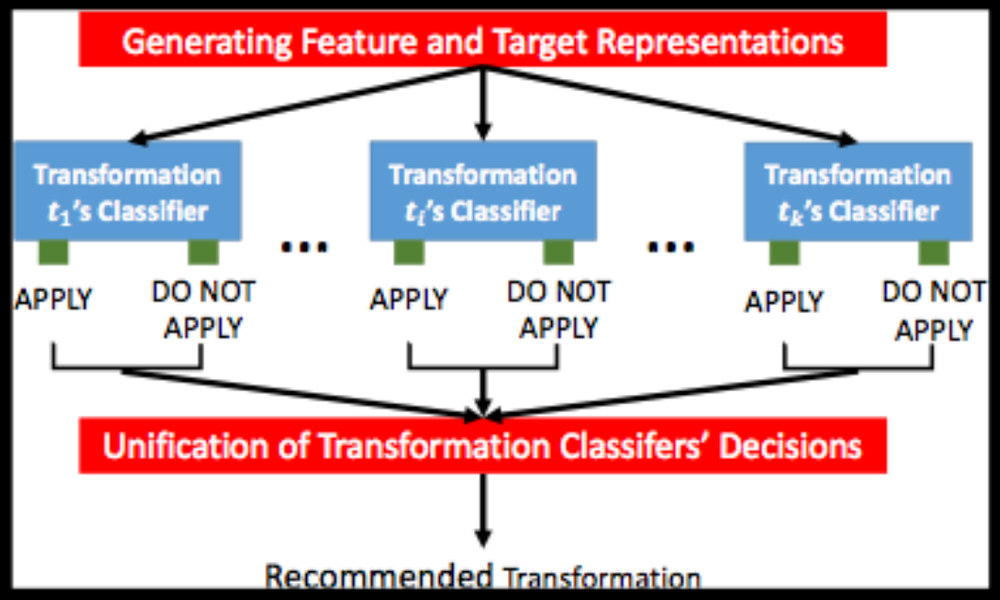


Predicting Transformations

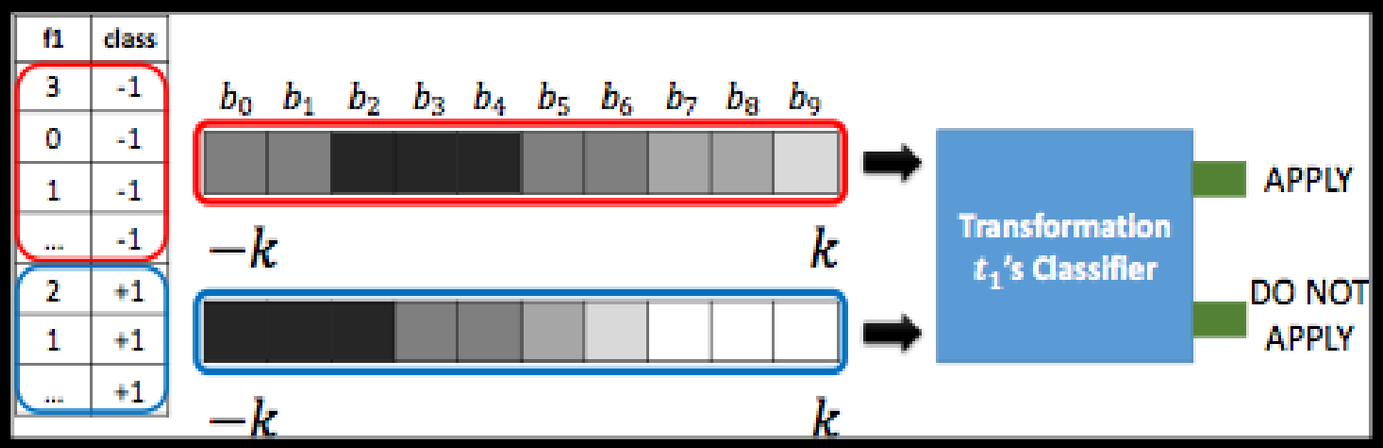
- Do we really need to trial and error?
  - Can’t we just see and tell which transforms are useful?
- Patterns not visible to the human eye
  - Use machine (learning) to crunch the patterns?
- Challenges in learning across multiple datasets:
  - Datasets have varying shapes
  - Datasets represent different problems
- We present LFE (Learning Feature Engineering):
  - Novel representation of data using data sketches
  - Predict the most useful transform for each feature
  - Learn across thousands of open source classification datasets

Learning-based Feature Engineering

- Learn correlations between feature distributions, target distributions and transformations
  - Build meta-models to predict good transformations through past observation
  - Generalize over 1000s of datasets across a variety of domains
- *Main Challenge:* Features vectors are of different sizes
- Solution: Quantile Sketch Array to capture the essential character of a feature.



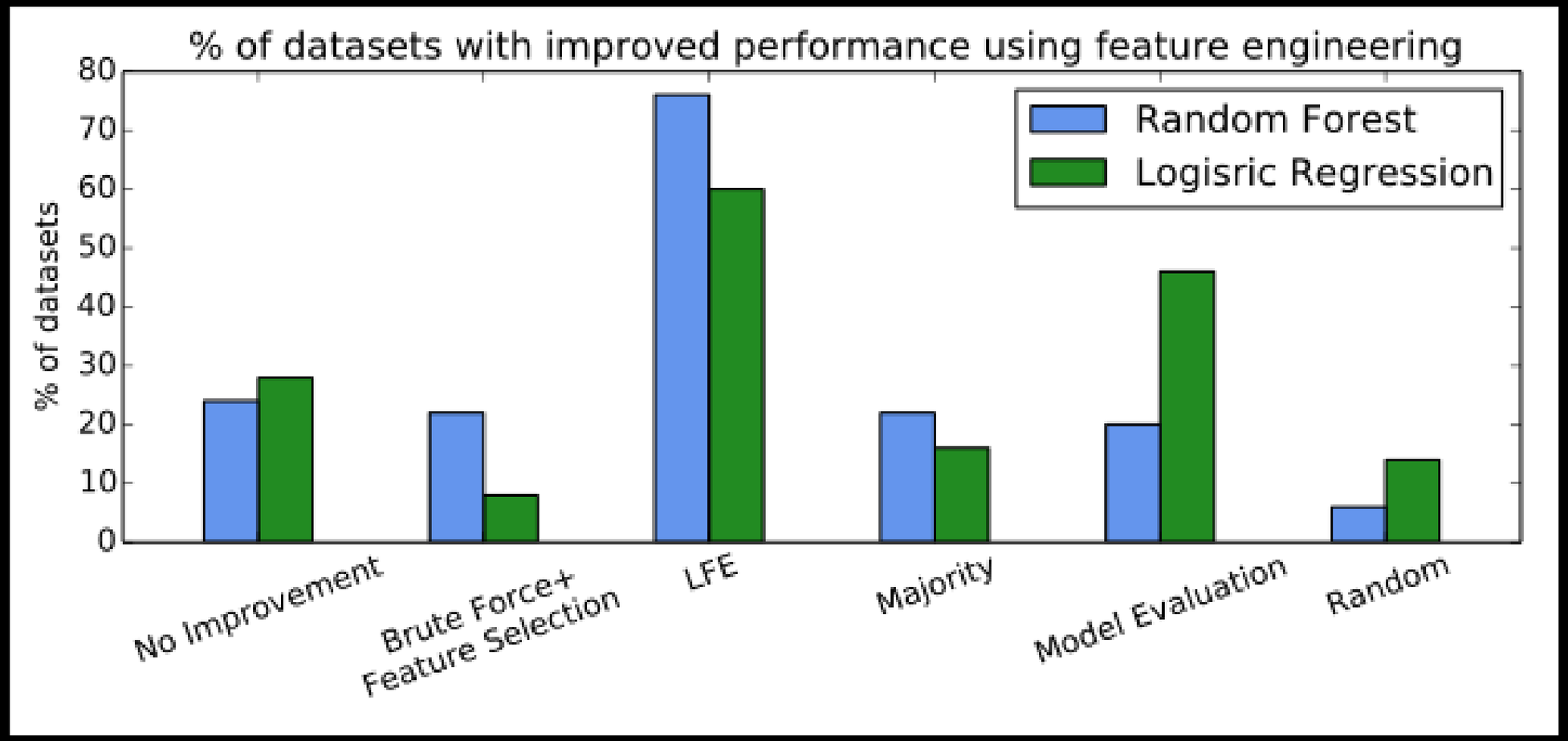
Multiple classifiers are consulted to and strongest recommendation is considered to apply a particular transforms for a feature.



An example of feature representation using *quantile sketch array (QSA)*. The feature *f1*'s values are binned into 10 equiwidth bins, separately for classes −1 and +1. The two resulting vectors are then concatenated and fed into the trans- formation t_i's classifier, which in turn recommends for or against applying t_i on *f1*.

Nargesian e al. Feature Engineering for Classification [IJCAI 17]

Experiments



The percentage of datasets, from a sample of 50, for which a feature engineering approach results in performance improvement (measured by F1 score of 10 fold cross validation for Random Forest and Logistic Regression)

Experiments

Dataset	#Numerical Features	#Data Points	Base Dataset	Majority	Brute-Force	Random (10 runs)	Evaluation based	unary LFE	binary LFE
AP-omentum-lung	10936	203	0.883	0.915	0.925	0.908	-	**0.929**	0.904
AP-omentum-ovary	10936	275	0.724	0.775	0.801	0.745	0.788	**0.811**	0.775
autos	48	4562	0.946	0.95	0.944	0.929	0.954	**0.96**	0.949
balance-scale	8	369	0.884	0.916	0.892	0.881	0.882	**0.919**	0.884
convex	784	50000	0.82	0.5	**0.913**	0.5	-	0.819	0.821
credit-a	6	690	0.753	0.647	0.521	0.643	0.748	**0.771**	0.771
dbworld-bodies	2	100	0.93	0.939	0.927	0.909	0.921	**0.961**	0.923
diabetes	8	768	0.745	0.694	0.737	0.719	0.731	**0.762**	0.749
fertility	9	100	0.854	0.872	0.861	0.832	0.833	**0.873**	0.861
gisette	5000	2100	0.941	0.601	0.741	0.855	-	**0.942**	0.933
hepatitis	6	155	0.747	0.736	0.753	0.727	**0.814**	0.807	0.831
higgs-boson-subset	28	50000	0.676	0.584	0.661	0.663	-	**0.68**	0.677
ionosphere	34	351	0.931	0.918	0.912	0.907	0.913	**0.932**	0.925
labor	8	57	0.856	0.827	0.855	0.806	0.862	**0.896**	0.896
lymph	10936	138	0.673	0.664	0.534	0.666	0.727	**0.757**	0.719
madelon	500	780	0.612	0.549	0.585	0.551	0.545	**0.617**	0.615
megawatt1	37	253	0.873	0.874	0.882	0.869	0.877	**0.894**	0.885
pima-indians-subset	8	768	0.74	0.687	**0.751**	0.726	0.735	0.745	0.76
secom	590	470	0.917	0.917	0.913	0.915	0.915	**0.918**	0.915
sonar	60	208	**0.808**	0.763	0.468	0.462	0.806	0.801	0.783
spambase	57	4601	**0.948**	0.737	0.39	0.413	0.948	0.947	0.947
spectf-heart	43	80	0.941	**0.955**	0.881	0.942	0.955	**0.955**	0.956
twitter-absolute	77	140707	**0.964**	0.866	0.946	0.958	0.963	0.964	0.964
Feature Engineering and Model Evaluation Time (seconds)		**geomean**	2.66	11.06	48.54	69.57	403.81	18.28	44.58
		average	19.23	1219.34	13723.51	2041.52	10508.75	97.90	188.17

Statistics of Datasets and F1 Score of LFE and Other Feature Engineering Approaches with 10-fold Cross Validation of Random Forest.

Live Demo [Cognito + LFE]

jupyter ForesightTutorial_CADS_LONDON Last Checkpoint: 10/23/2017 (autosaved)

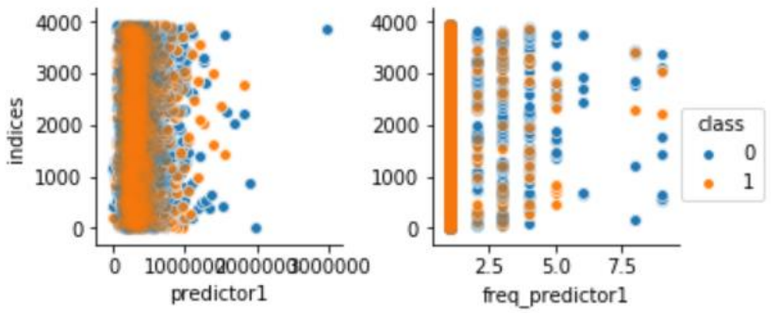
Logout

File Edit View Insert Cell Kernel Widgets Help

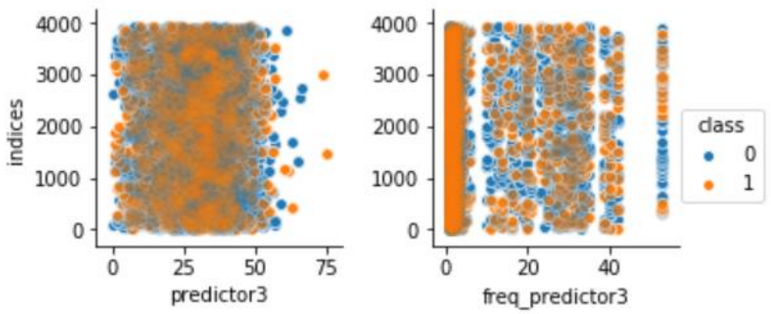
Trusted | Python 3

Code

```
In [10]: explorer.next(vis=True)
```




```
In [11]: explorer.next(vis=True)
```



jupyter Cognito-biking-example Last Checkpoint: 10/24/2017 (unsaved changes)

Logout

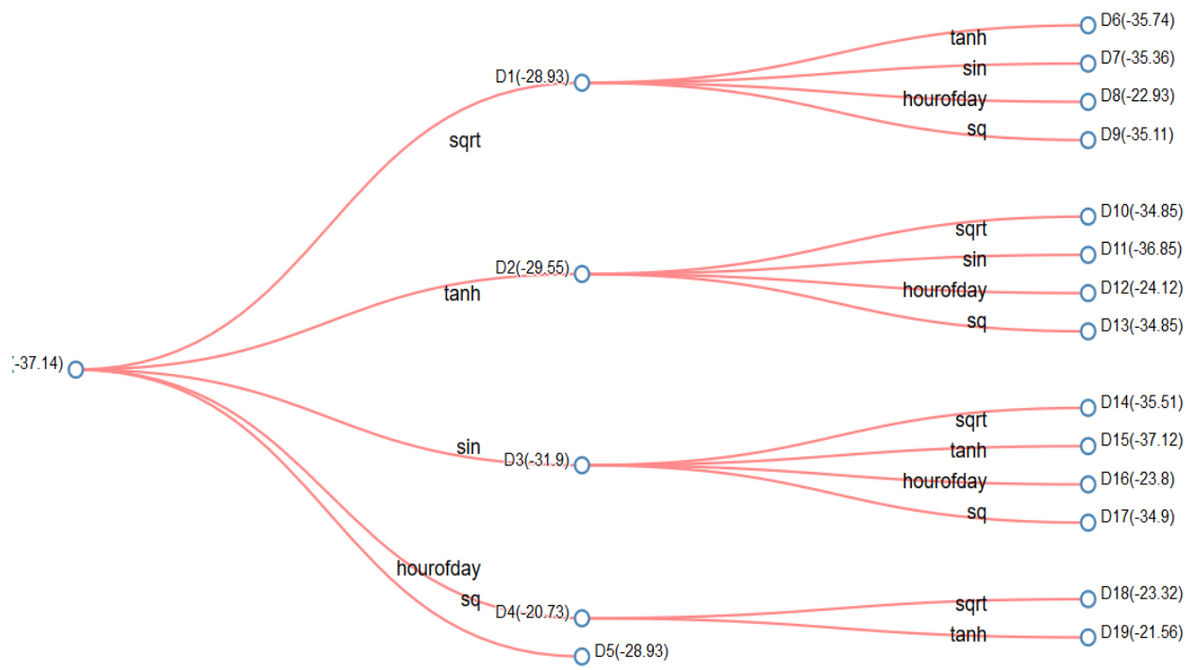
File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

Code

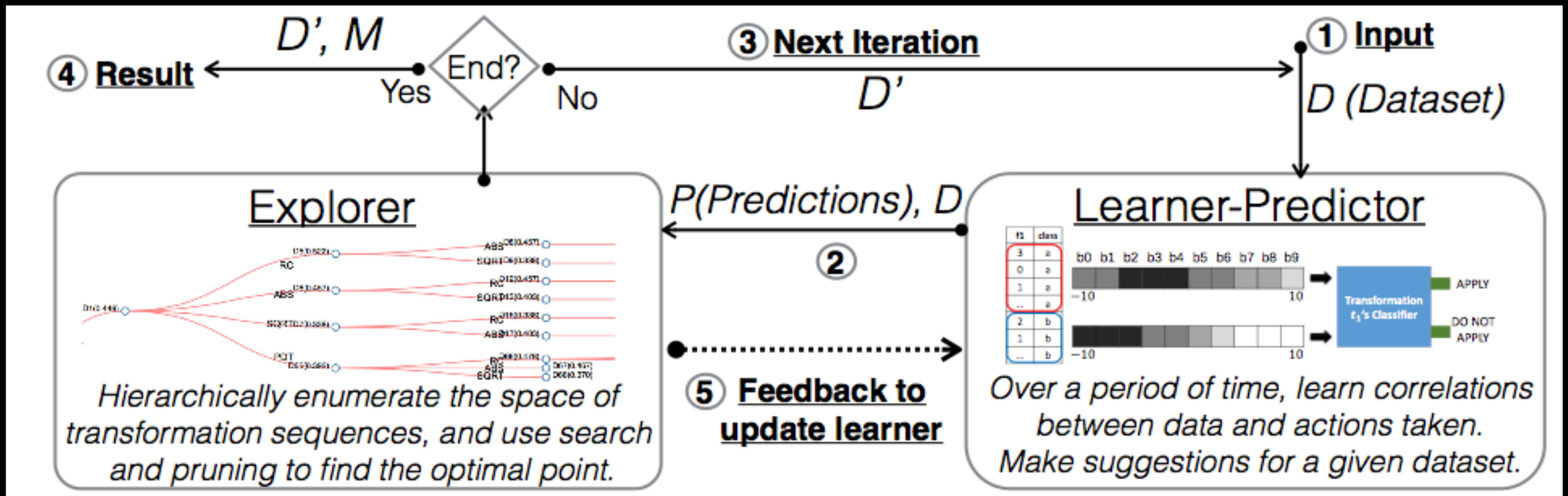
```
In [8]: %%d3
print('Using d3 visualization inline notebook')
```

print('Using d3 visualization inline notebook')



Combined: Explorer+Predictor

- Coming soon:



Automating Feature Engineering
Udayan Khurana, Fatemeh Nargesian, Horst Samulowitz, Elias Khalil, Deepak Turaga
NIPS workshop on Artificial Intelligence for Data Science, 2016

How to deal with model and data dependency?

Data Allocation using Upper Bounds (DAUB)

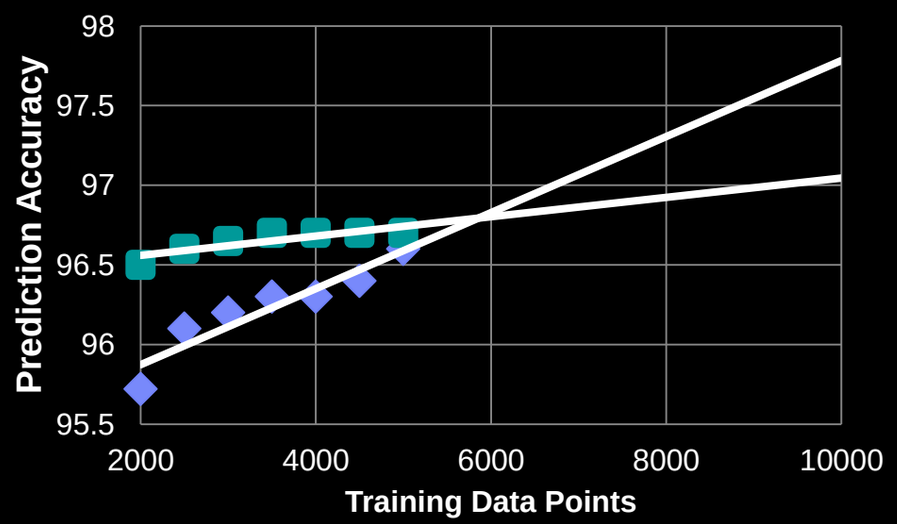
- Question: How can one robustly project the accuracy at **n** data points to the expected accuracy at **N** data points?

- Proposal: Apply principle of "Optimism under Uncertainty"

  - **DAUB**'s upper bound is based on first order Taylor expansion of unknown reward function f(N)

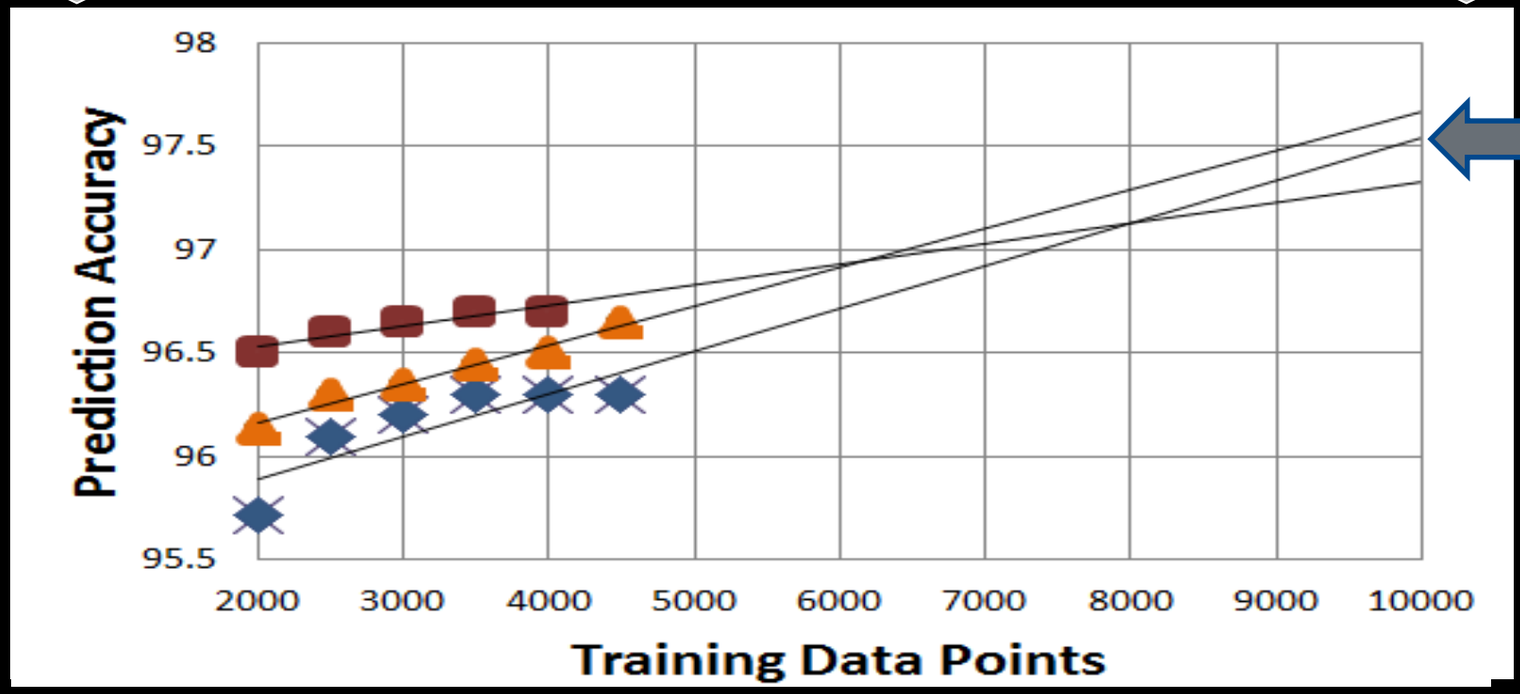
  - Using discrete derivative f'(n,s) = (f(n) – f(n-s)) / s where s a natural number

  - Allocate more training data to approach with best expected performance



Bandit-based Algorithm – An Example

Prediction Accuracy versus #data points



Upper bound estimate on performance

Theoretical Support for DAUB

- **Using following assumptions:**
  1. Learning curves are monotone
  2. Diminishing returns
  3. Access to *true* accuracy/cost (instead of *observed* quantities)
     - 1+2 can be enforced in practice, but 3. would be too expensive in practice – however as number of allocated data points increases observed accuracy converges to true accuracy

- **Analysis of idealized DAUB = DAUB***
  - Training data allocation sequence chosen by DAUB* is essentially optimal

Bounded Regret of DAUB*

- Learner **f** is called **D-optimal** iff **f(N) >= f*(N) – D**
- *Cost(f)* = computational cost of training **f** on **N** data points
- *D-Regret(f)* = cost spent on **D-suboptimal** learner **f**

Theorem[#]: If DAUB* selects f’ and f is any D-suboptimal learner, then

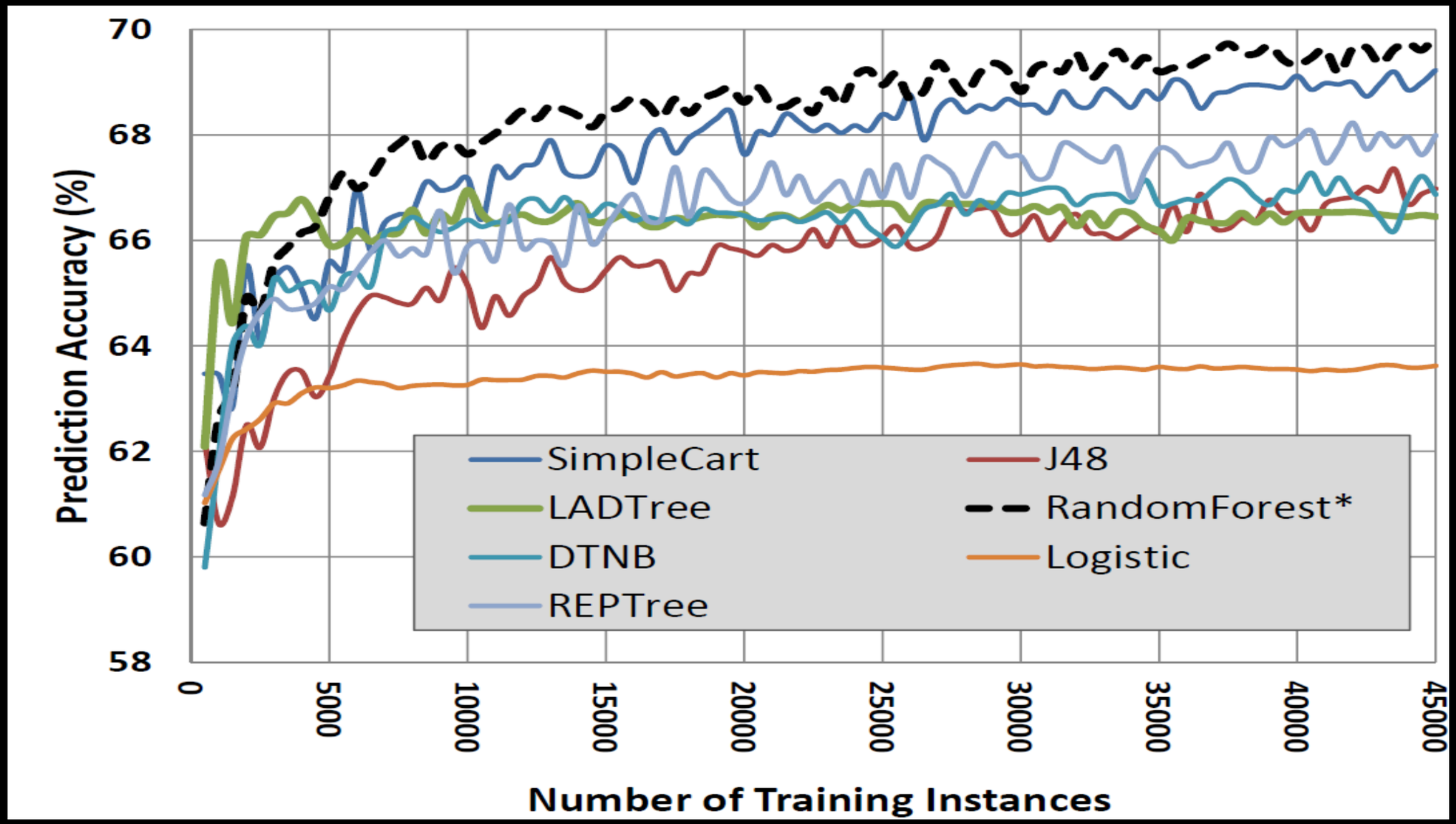
1. f’ is D-optimal
2. D-regret(f) is sub-linear in cost(f’)
3. The bound on D-regret(f) is tight up to a constant factor

[#for more details on assumptions etc. please refer to the paper “]

Things to consider in practice

- **Learning curves are not always monotone, but….**
  - Simple to employ techniques that ensure monotonicity (e.g. monotone regression)

- **Ways to improve accuracy estimate**
  - **Projected accuracy can be very inaccurate (especially in the beginning)**
    - In particular it can be often above 100% (e.g., 294%), one could cap at 100%, but loses ranking information
  - **Use Training Performance to improve estimate**
    - Assuming that training and test data come from the same distribution the training performance (accuracy at sample size) can serve as upper bound!
    - Assumption: Training Error < Validation Error
    - Upper Bound Estimate = choose minimum of projected accuracy and training accuracy



		Full Training		DAUB				
Dataset	Application Area	Allocation	Time (s)	Iterations	Allocation	Time (s)	Speedup	Loss
Buzz	social media	1,578k	56,519	57	302k	5,872	10x	0.0%
Cover Type	forestry	1,578k	43,578	13	160k	3,848	11x	1.1%
HIGGS	signal processing	1,578k	49,905	56	372k	2,001	25x	0.0%
Million Songs	music	1,578k	115,911	53	333k	17,208	7x	0.6%
SUSY	high-energy physics	1,578k	26,438	31	214k	837	31x	0.9%
Vehicle Sensing	vehicle management	1,578k	68,139	50	296k	5,603	12x	0.0%

Feature Engineering with Neural Networks

FE with Deep Learning

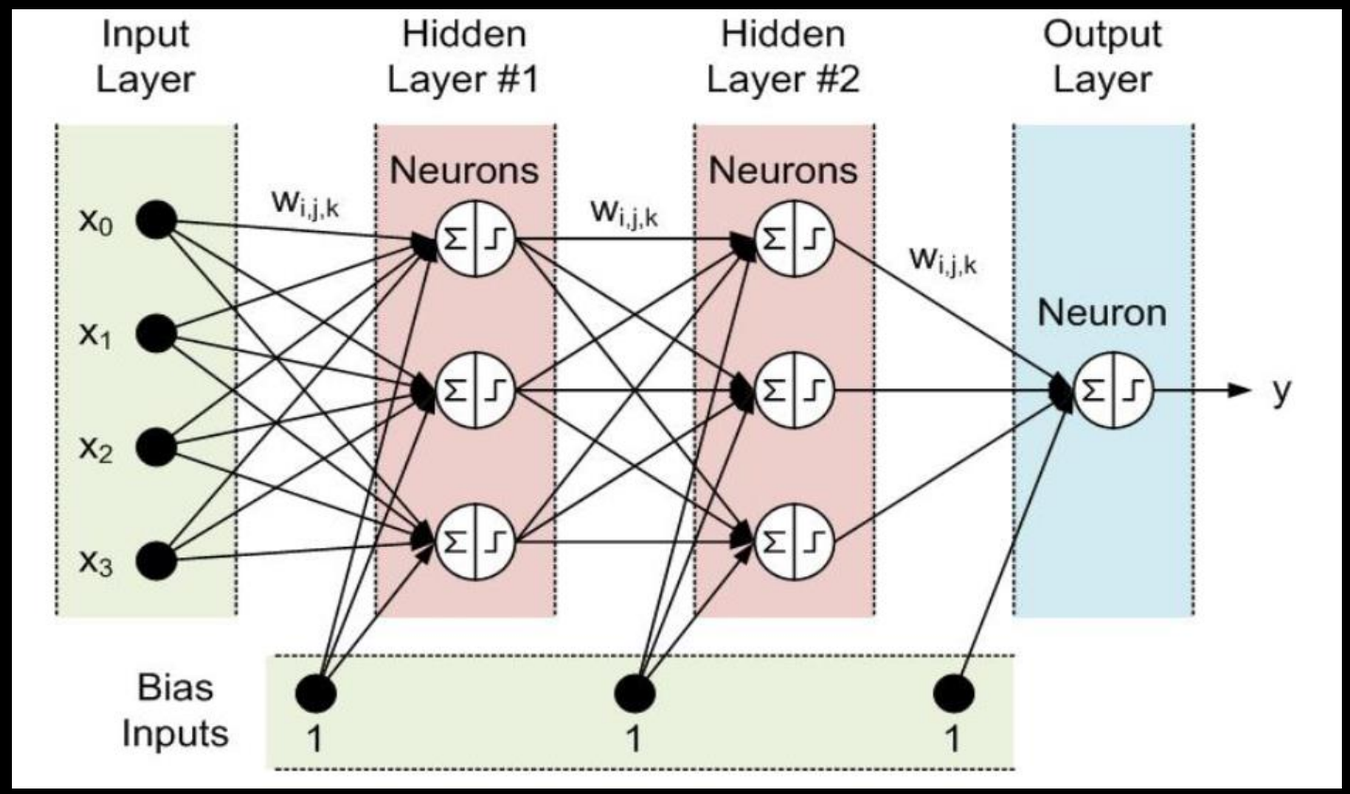
- Deep nets perform feature engineering
- Incredibly successful in several domains

- However,
  - Need extensive amount of data
  - Lack interpretability
  - Not known to work generally in all domains
  - Need architectural configuration for a new problem

Automated Composition of Neural Networks

Automated Composition of Neural Networks

- Problem in Deep Learning is not so much the large variety of analytics, but enormous configuration space of a single Deep Neural Network (DNN)
- Wide range of settings and combinations to choose from:
  - Learning Rate
  - Number of layers
  - Number of nodes per layer
  - Activation function per layer
  - Pre-Train yes or no
  - Drop-Out rate
  - …

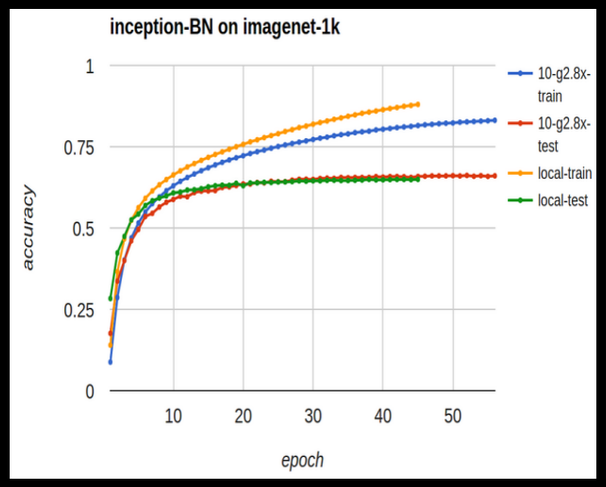


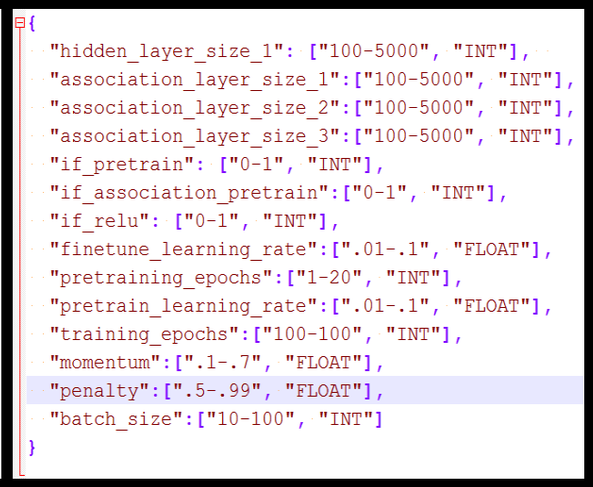
- *Ideas:*
  - Combine learning curve estimation procedure with hyper-parameter optimization
  - Apply mathematical optimization

["*An effective algorithm for hyperparameter optimization of neural networks*", IBM Journal of Research and Development, 2017]

Bandit-Based approach applied to NN search

- Use the number of epochs vs performance to estimate performance of a NN configuration
  - Use slope of learning curve to estimate performance on full training

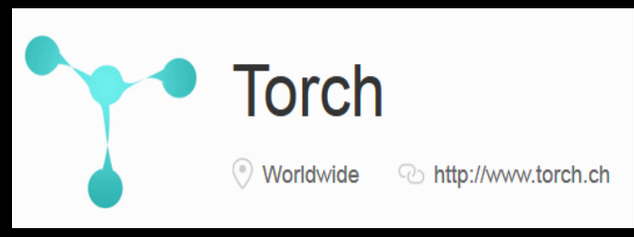




```
{
  "hidden_layer_size_1": ["100-5000", "INT"],
  "association_layer_size_1":["100-5000", "INT"],
  "association_layer_size_2":["100-5000", "INT"],
  "association_layer_size_3":["100-5000", "INT"],
  "if_pretrain": ["0-1", "INT"],
  "if_association_pretrain":["0-1", "INT"],
  "if_relu": ["0-1", "INT"],
  "finetune_learning_rate":[".01-.1", "FLOAT"],
  "pretraining_epochs":["1-20", "INT"],
  "pretrain_learning_rate":[".01-.1", "FLOAT"],
  "training_epochs":["100-100", "INT"],
  "momentum":[".1-.7", "FLOAT"],
  "penalty":[".5-.99", "FLOAT"],
  "batch_size":["10-100", "INT"]
}
```

Keras

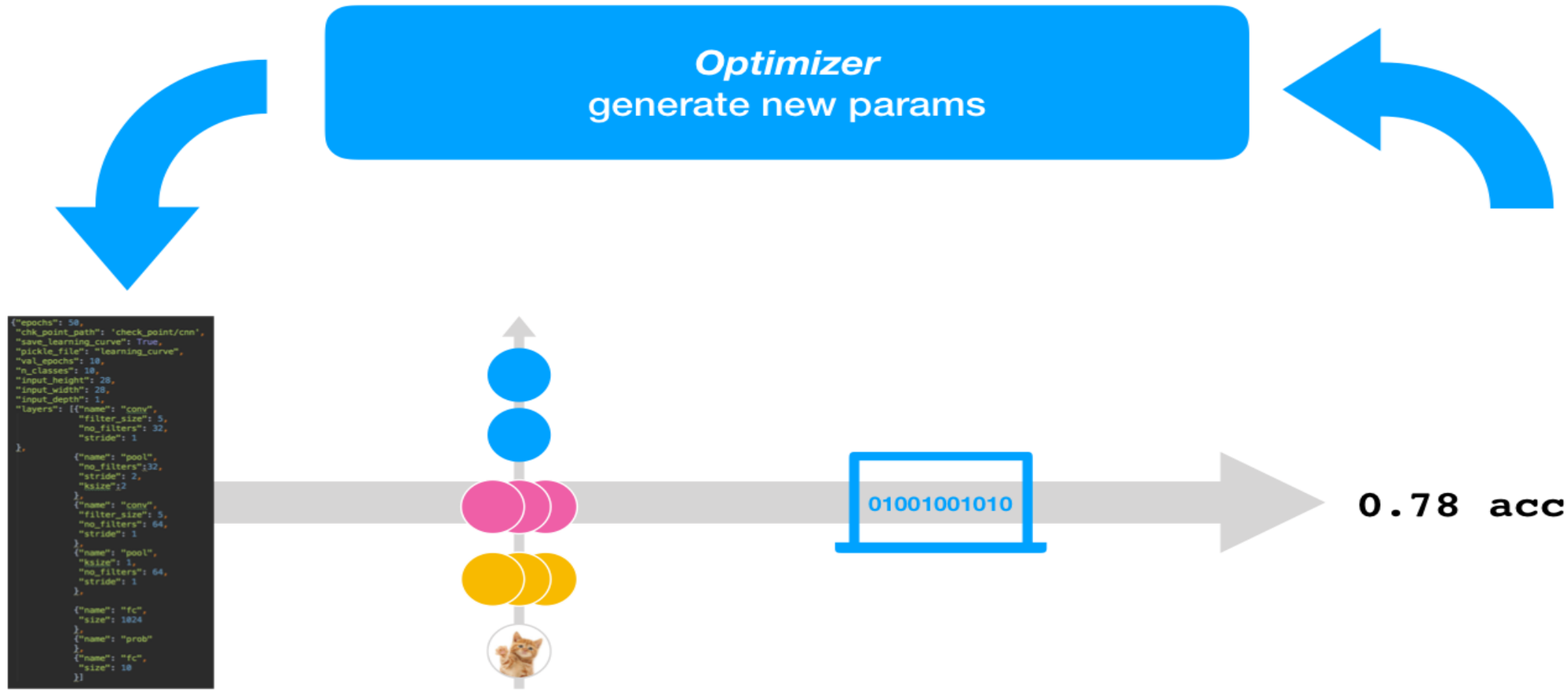
TensorFlow ™



- Support of arbitrary framework (e.g., THEANO, TORCH, TENSORFLOW) through wrapper interface
  1. Start with a default network, parameters and ranges are specified in JSON
  2. Perform hyper-parameter optimization (how many nodes per layer, activation function, learning rate, drop out, etc.) but only allow limited number of epochs
  3. Perform DAUB estimation on new configurations and allocate more epochs based on estimated performance
- Deploy on GPU cluster

Automated Composition of Neural Networks

Idea — Optimize NN Model architectures and parametrization to solve a specific 'cognitive' task



parameters
architecture
learning rate
kernel size
filters

generate
Neural Net

train model
on all data

evaluate model