Lale: Consistent Automated Machine Learning

Guillaume Baudart, Martin Hirzel,
Kiran Kate, Pari Ram, and Avi Shinnar

AutoML Workshop @KDD’20, Aug. 24

https://github.com/ibm/lale

https://arxiv.org/abs/2007.01977

Spectrum of Automation

		What is being configured?		
		Operator choice	Hyperparameter configuration	Pipeline topology
How is it being configured?	Manual	scikit-learn		
	Automated (with control)	sklearn.model_selection.GridSearchCV, SMAC, hyperopt		AlphaD3M
	Automated (black-box)	auto-sklearn, hyperopt-sklearn		TPOT

Existing AutoML tools …

- … automatically find good configurations ☺
- … are open-source ☺
- … are scikit-learn-based ☺
- … support only part of the automation spectrum
- … use inconsistent programming models

Programming Model Requirements

- Support entire automation spectrum in a consistent way
- Be consistent across tools (hyperopt, GridSearchCV, SMAC)
  ➔ Search space generation
- Extend established abstractions (scikit-learn, JSON schema)

Operator Choice

Manual

```
pipeline = J48()
trained = pipeline.fit(X, y)
```

Automated (aka. algorithm selection)

```
pipeline = J48() | LR()
trained = pipeline.auto_configure(X, y, optimizer=GridSearchCV)
```

Hyperparameter Configuration

Manual

```
pipeline = J48(R=False, C=0.3)
trained = pipeline.fit(X, y)
```

```
J48: { allOf: [
  { type: object,
    properties: {
      R: { description: "Use reduced error pruning", type: boolean },
      C: { description: "Pruning confidence threshold",
           type: number, minimum: 0.0, maximum: 1.0,
           maximumForOptimizer: 0.5, distribution: uniform }}},
  { description: "Setting confidence makes no sense for R",
    anyOf: [
      { not: { type: object, properties: {R: {enum: [true]}}}},
      { type: object, properties: {C: {enum: [0.25]}}}]}]}
```

Automated (aka. hyperparameter tuning)

```
pipeline = J48
trained = pipeline.auto_configure(X, y, optimizer=SMAC)
```

Pipeline Composition

Manual

```
prep_num = Project(columns={'type':'number'}) >> PCA()
prep_str = Project(columns={'type':'string'}) >> OneHotEncoder()
pipeline = (prep_num & prep_str) >> ConcatFeatures >> LR()
trained = pipeline.fit(X, y)
```

Automated (aka. topology search)

```
g.start = g.prep >> (J48 | LR)
g.prep = NoOp | (g.prep >> g.prep1)
g.prep1 = StandardScaler | Normalizer | PolynomialFeatures | PCA
trained = g.unfold(3).auto_configure(X, y, optimizer=Hyperopt)
```

Higher-Order Operators

Manual

```
tree = DecisionTreeClassifier(max_depth=1)
clf = AdaBoostClassifier(base_estimator=tree, n_estimators=10)
pipeline = StandardScaler() >> clf
trained = pipeline.fit(X, y)
```

Automated (aka. topology search)

```
clf = AdaBoostClassifier(base_estimator=DecisionTreeClassifier)
pipeline = (StandardScaler | PCA | NoOp) >> clf
trained = pipeline.auto_configure(X, y, optimizer=Hyperopt)
```

Demonstration

In [2]:

```
from sklearn.preprocessing import StandardScaler as Scale
from sklearn.preprocessing import Normalizer as Norm
from lale.lib.lale import NoOp
from sklearn.decomposition import PCA
from sklearn.tree import DecisionTreeRegressor as Tree
from sklearn.linear_model import LinearRegression as Linear
from xgboost import XGBRegressor as XGB
lale.wrap_imported_operators()
```

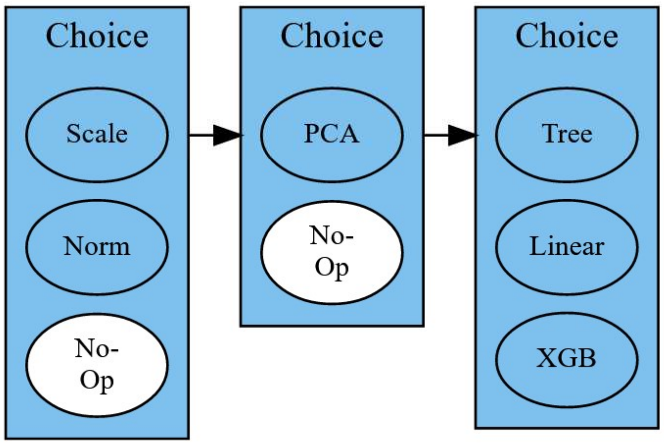
Demonstration

In [2]:

```
from sklearn.preprocessing import StandardScaler as Scale
from sklearn.preprocessing import Normalizer as Norm
from lale.lib.lale import NoOp
from sklearn.decomposition import PCA
from sklearn.tree import DecisionTreeRegressor as Tree
from sklearn.linear_model import LinearRegression as Linear
from xgboost import XGBRegressor as XGB
lale.wrap_imported_operators()
```

In [3]:

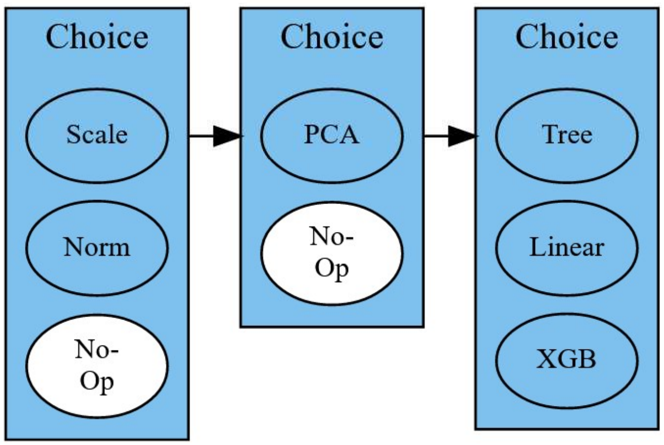
```
planned_pipeline = (Scale | Norm | NoOp) >> (PCA | NoOp) >> (Tree | Linear | XGB)
planned_pipeline.visualize()
```



Demonstration

In [3]:

```
planned_pipeline = (Scale | Norm | NoOp) >> (PCA | NoOp) >> (Tree | Linear | XGB)
planned_pipeline.visualize()
```



In [4]:

```
from lale.lib.lale import Hyperopt
import sklearn.metrics
r2 = sklearn.metrics.make_scorer(sklearn.metrics.r2_score)
trained_pipeline = planned_pipeline.auto_configure(
    train_X, train_y, optimizer=Hyperopt,
    scoring=r2, max_opt_time=10*60, max_eval_time=60, cv=3)
```

```
100%|██████████| 50/50 [10:02<00:00, 12.05s/trial, best loss: -0.8107292214446913]
```

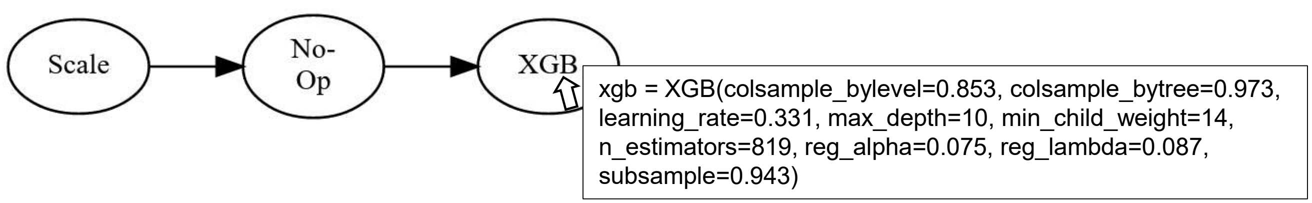
Demonstration

In [4]:
```python
from lale.lib.lale import Hyperopt
import sklearn.metrics
r2 = sklearn.metrics.make_scorer(sklearn.metrics.r2_score)
trained_pipeline = planned_pipeline.auto_configure(
    train_X, train_y, optimizer=Hyperopt,
    scoring=r2, max_opt_time=10*60, max_eval_time=60, cv=3)
```

```
100%|██████████| 50/50 [10:02<00:00, 12.05s/trial, best loss: -0.8107292214446913]
```

In [5]:
```python
print(f'R2 score: {r2(trained_pipeline, test_X, test_y):.2f}')
trained_pipeline.visualize()
```

```
R2 score: 0.83
```

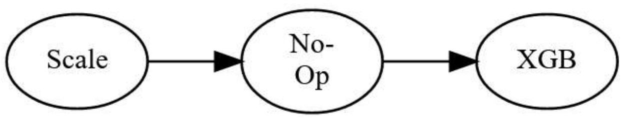


Demonstration

In [5]:

```python
print(f'R2 score: {r2(trained_pipeline, test_X, test_y):.2f}')
trained_pipeline.visualize()
```

```
R2 score: 0.83
```



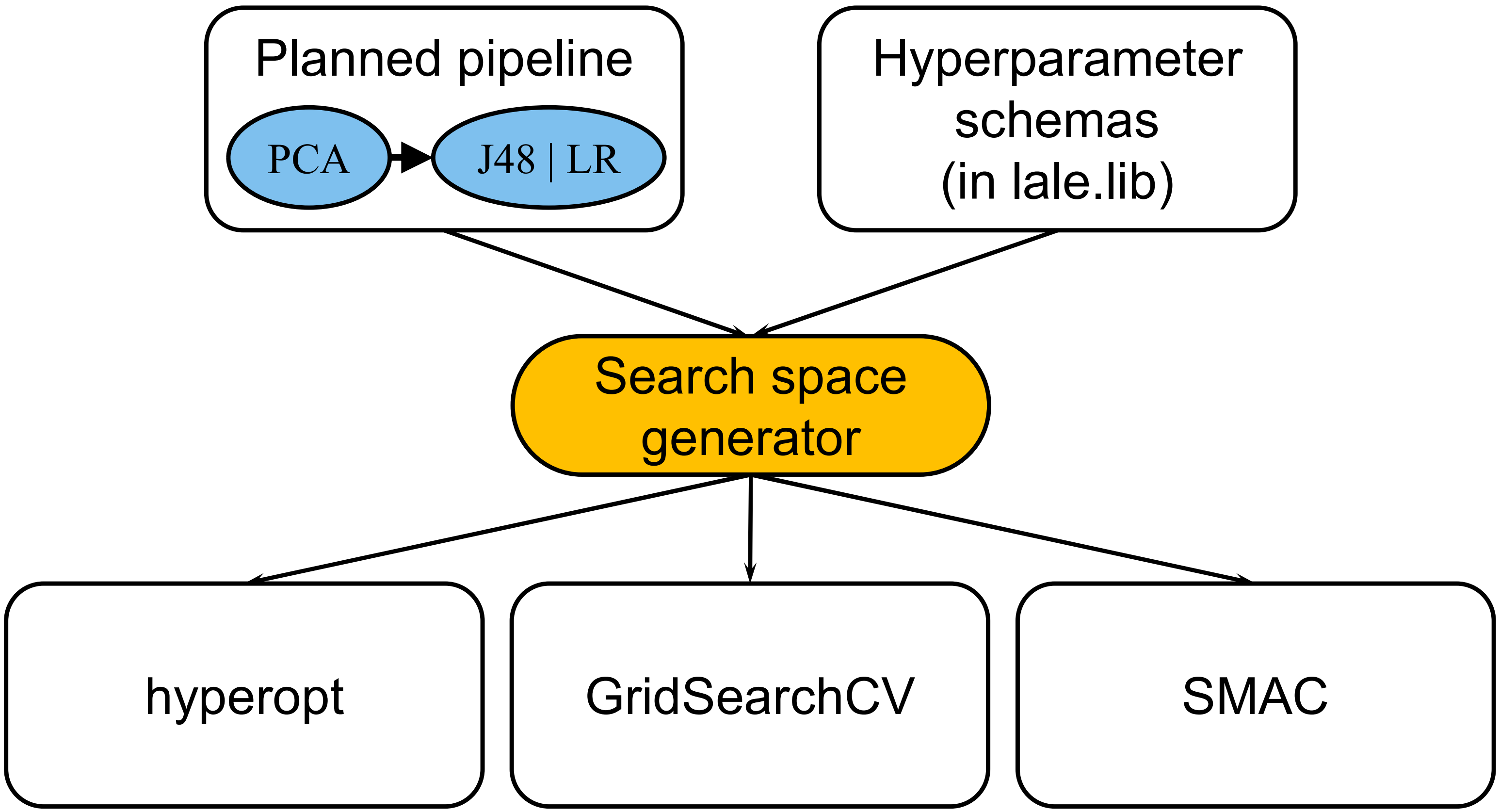
In [6]:

```python
trained_pipeline.pretty_print(ipython_display=True)
```

```python
from lale.lib.sklearn.standard_scaler import Scale
from lale.lib.lale import NoOp
from lale.lib.xgboost.xgb_regressor import XGB
import lale
lale.wrap_imported_operators()

xgb = XGB(colsample_bylevel=0.8539909881709349, colsample_bytree=0.9733482608060157, learning_rate=0.3319087714488286, max_depth=10, min_child_weight=14, n_estimators=819, reg_alpha=0.07519396924044132, reg_lambda=0.0874628984235771, subsample=0.9433039199868445)
pipeline = Scale() >> NoOp() >> xgb
```

Consistency Across Existing Tools



Normalize

Hyperparameter schemas (in lale.lib)

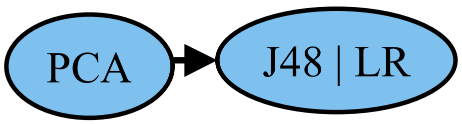
$$
\begin{aligned}
PCA &: \text{dict}\{N\!: (0..1) \vee [mle]\}\\
J48 &: \text{dict}\{R\!: [true, false], C\!: (0..0.5)\} \wedge\\
&\quad (\text{dict}\{R\!: [true]\} \Rightarrow \text{dict}\{C\!: [0.25]\})\\
LR &: \text{dict}\{S\!: [linear, sag, lbfgs], P\!: [l1, l2]\} \wedge\\
&\quad (\text{dict}\{S\!: [sag, lbfgs]\} \Rightarrow \text{dict}\{P\!: [l2]\})
\end{aligned}
$$

Normalized schemas

$$
\begin{aligned}
PCA &: \text{dict}\{N\!: (0..1)\} \vee \text{dict}\{N\!: [mle]\}\\
J48 &: \text{dict}\{R\!: [false], C\!: (0..0.5)\} \vee \text{dict}\{R\!: [true, false], C\!: [0.25]\}\\
LR &: \text{dict}\{S\!: [linear], P\!: [l1, l2]\} \vee \text{dict}\{S\!: [linear, sag, lbfgs], P\!: [l2]\}
\end{aligned}
$$

Combine

Planned pipeline



Normalized schemas

$$\begin{aligned}
&\mathit{PCA}\!: \mathrm{dict}\{N\!: (0..1)\} \vee \mathrm{dict}\{N\!: [\mathit{mle}]\}\\
&\mathit{J48}\;\,: \mathrm{dict}\{R\!: [\mathit{false}], C\!: (0..0.5)\} \vee \mathrm{dict}\{R\!: [\mathit{true}, \mathit{false}], C\!: [0.25]\}\\
&\mathit{LR}\;\;\,: \mathrm{dict}\{S\!: [\mathit{linear}], P\!: [\mathit{l1}, \mathit{l2}]\} \vee \mathrm{dict}\{S\!: [\mathit{linear}, \mathit{sag}, \mathit{lbfgs}], P\!: [\mathit{l2}]\}
\end{aligned}$$

Combined search space

$$\mathrm{dict}\left\{\begin{array}{l}
0\!: \mathrm{dict}\{N\!: (0..1)\} \vee \mathrm{dict}\{N\!: [\mathit{mle}]\}\\
1\!: \left(\begin{array}{l}
\left(\begin{array}{l}\mathrm{dict}\{D\!: [\mathit{J48}], R\!: [\mathit{false}], C\!: (0..0.5)\} \vee\\ \mathrm{dict}\{D\!: [\mathit{J48}], R\!: [\mathit{true}, \mathit{false}], C\!: [0.25]\}\end{array}\right) \vee\\
\left(\begin{array}{l}\mathrm{dict}\{D\!: [\mathit{LR}], S\!: [\mathit{linear}], P\!: [\mathit{l1}, \mathit{l2}]\} \vee\\ \mathrm{dict}\{D\!: [\mathit{LR}], S\!: [\mathit{linear}, \mathit{sag}, \mathit{lbfgs}], P\!: [\mathit{l2}]\}\end{array}\right)
\end{array}\right)
\end{array}\right\}$$

hyperopt

Flatten

Combined search space

$$\text{dict}\left\{\begin{array}{l} 0\colon \text{dict}\{N\colon (0..1)\} \vee \text{dict}\{N\colon [\mathit{mle}]\} \\ 1\colon \left(\begin{pmatrix} \text{dict}\{D\colon [\mathit{J48}], R\colon [\mathit{false}], C\colon (0..0.5)\} \vee \\ \text{dict}\{D\colon [\mathit{J48}], R\colon [\mathit{true}, \mathit{false}], C\colon [0.25]\} \end{pmatrix} \vee \begin{pmatrix} \text{dict}\{D\colon [\mathit{LR}], S\colon [\mathit{linear}], P\colon [\mathit{l1}, \mathit{l2}]\} \vee \\ \text{dict}\{D\colon [\mathit{LR}], S\colon [\mathit{linear}, \mathit{sag}, \mathit{lbfgs}], P\colon [\mathit{l2}]\} \end{pmatrix}\right) \end{array}\right\}$$

Flat search space

$$\begin{array}{l} \ \text{dict}\{N\colon (0..1), D\colon [\mathit{J48}], R\colon [\mathit{false}], C\colon (0..0.5)\} \\ \vee\ \text{dict}\{N\colon (0..1), D\colon [\mathit{J48}], R\colon [\mathit{true}, \mathit{false}], C\colon [0.25]\} \\ \vee\ \text{dict}\{N\colon [\mathit{mle}], D\colon [\mathit{J48}], R\colon [\mathit{false}], C\colon (0..0.5)\} \\ \vee\ \text{dict}\{N\colon [\mathit{mle}], D\colon [\mathit{J48}], R\colon [\mathit{true}, \mathit{false}], C\colon [0.25]\} \\ \vee\ \text{dict}\{N\colon (0..1), D\colon [\mathit{LR}], S\colon [\mathit{linear}], P\colon [\mathit{l1}, \mathit{l2}]\} \\ \vee\ \text{dict}\{N\colon (0..1), D\colon [\mathit{LR}], S\colon [\mathit{linear}, \mathit{sag}, \mathit{lbfgs}], P\colon [\mathit{l2}]\} \\ \vee\ \text{dict}\{N\colon [\mathit{mle}], D\colon [\mathit{LR}], S\colon [\mathit{linear}], P\colon [\mathit{l1}, \mathit{l2}]\} \\ \vee\ \text{dict}\{N\colon [\mathit{mle}], D\colon [\mathit{LR}], S\colon [\mathit{linear}, \mathit{sag}, \mathit{lbfgs}], P\colon [\mathit{l2}]\} \end{array}$$

SMAC

Discre-tize

Flat search space

$$
\begin{aligned}
&\text{dict}\{N\colon (0..1), D\colon [\mathit{J48}], R\colon [\mathit{false}], C\colon (0..0.5)\}\\
\vee\ &\text{dict}\{N\colon (0..1), D\colon [\mathit{J48}], R\colon [\mathit{true}, \mathit{false}], C\colon [0.25]\}\\
\vee\ &\text{dict}\{N\colon [\mathit{mle}], D\colon [\mathit{J48}], R\colon [\mathit{false}], C\colon (0..0.5)\}\\
\vee\ &\text{dict}\{N\colon [\mathit{mle}], D\colon [\mathit{J48}], R\colon [\mathit{true}, \mathit{false}], C\colon [0.25]\}\\
\vee\ &\text{dict}\{N\colon (0..1), D\colon [\mathit{LR}], S\colon [\mathit{linear}], P\colon [\mathit{l1}, \mathit{l2}]\}\\
\vee\ &\text{dict}\{N\colon (0..1), D\colon [\mathit{LR}], S\colon [\mathit{linear}, \mathit{sag}, \mathit{lbfgs}], P\colon [\mathit{l2}]\}\\
\vee\ &\text{dict}\{N\colon [\mathit{mle}], D\colon [\mathit{LR}], S\colon [\mathit{linear}], P\colon [\mathit{l1}, \mathit{l2}]\}\\
\vee\ &\text{dict}\{N\colon [\mathit{mle}], D\colon [\mathit{LR}], S\colon [\mathit{linear}, \mathit{sag}, \mathit{lbfgs}], P\colon [\mathit{l2}]\}
\end{aligned}
$$

↓

Discretized search space

$$
\begin{aligned}
&\text{dict}\{N\colon [0.50, 0.01], D\colon [\mathit{J48}], R\colon [\mathit{false}], C\colon [0.25, 0.01]\}\\
\vee\ &\text{dict}\{N\colon [0.50, 0.01], D\colon [\mathit{J48}], R\colon [\mathit{true}, \mathit{false}], C\colon [0.25]\}\\
\vee\ &\text{dict}\{N\colon [\mathit{mle}], D\colon [\mathit{J48}], R\colon [\mathit{false}], C\colon [0.25, 0.01]\}\\
\vee\ &\text{dict}\{N\colon [\mathit{mle}], D\colon [\mathit{J48}], R\colon [\mathit{true}, \mathit{false}], C\colon [0.25]\}\\
\vee\ &\text{dict}\{N\colon [0.50, 0.01], D\colon [\mathit{LR}], S\colon [\mathit{linear}], P\colon [\mathit{l1}, \mathit{l2}]\}\\
\vee\ &\text{dict}\{N\colon [0.50, 0.01], D\colon [\mathit{LR}], S\colon [\mathit{linear}, \mathit{sag}, \mathit{lbfgs}], P\colon [\mathit{l2}]\}\\
\vee\ &\text{dict}\{N\colon [\mathit{mle}], D\colon [\mathit{LR}], S\colon [\mathit{linear}], P\colon [\mathit{l1}, \mathit{l2}]\}\\
\vee\ &\text{dict}\{N\colon [\mathit{mle}], D\colon [\mathit{LR}], S\colon [\mathit{linear}, \mathit{sag}, \mathit{lbfgs}], P\colon [\mathit{l2}]\}
\end{aligned}
$$

→ Grid-Search-CV

177 Operators with Schemas

Package	Count	Description
lale.lib.sklearn	46	Hand-curated scikit-learn operators (subset of lale.lib.autogen)
lale.lib.autogen	115	Auto-extracted scikit-learn operators
lale.lib.aif360	4	Fairness mitigator
lale.lib.autoai_libs	21	Data cleansing and feature engineering
github.com/lale/lale-gpl	2	J48 from Weka and ARulesCBA from R
lale.lib.imblearn	12	Class imbalance handling
lale.lib.lale	15	Optimizers and utility operators
lale.lib.lightgbm	2	Gradient-boosted random forests
lale.lib.pytorch	2	BERT and ResNet
lale.lib.spacy	1	Glove
lale.lib.tensorflow	1	Universal sentence encoder
lale.lib.xgboost	2	Gradient-boosted random forests

Results

- Comparison against auto-sklearn
  - 4 Lale pipelines, including grammars and higher-order operators
  - Competitive results on 15 OpenML datasets
- Case studies with other modalities
  - Text, images, time-series
- Effects of side constraints on convergence
  - Pruning the search space beats catching exceptions from trials
- See paper for details

	$100 * (accuracy/\text{AUTOSKL} - 1)$			
DATASET	LALE-PIPE	LALE-TPOT	LALE-AD3M	LALE-ADB
australian	0.41	0.93	2.06	1.13
blood	-2.08	-0.52	-4.05	-1.04
breast-cancer	-2.59	-2.31	-4.90	-2.88
car	-1.13	-0.25	-6.70	-1.09
credit-g	-2.29	-3.24	-2.37	-0.71
diabetes	0.61	-0.82	1.12	-1.33
hill-valley	-0.20	0.55	-2.66	0.55
jungle-chess	2.54	0.96	-15.80	1.53
kc1	-0.38	-0.38	-0.21	-0.58
kr-vs-kp	-0.36	-0.27	-2.87	-0.19
mfeat-factors	-1.14	-1.54	-1.17	-1.20
phoneme	-1.39	-0.83	-15.20	-0.22
shuttle	14.51	14.50	14.45	14.56
spectf	-0.78	0.59	-4.90	0.59
sylvine	-0.45	-1.07	-4.31	-0.29

Spectrum of Automation, Revisited

		What is being configured?		
		Operator choice	Hyperparameter configuration	Pipeline topology
How is it being configured?	Manual	scikit-learn		
	Automated (with control)	sklearn.model_selection.GridSearchCV SMAC, hyperopt		AlphaD3M
	Automated (black-box)	auto-sklearn, hyperopt-sklearn		TPOT

LALE

Conclusion

- https://github.com/ibm/lale
- Use it
- Star it
- Contribute
  - Operators
  - Optimizers
  - …