Lale: Type-Driven Auto-ML with Scikit-Learn

Talk at ICML Expo, 12 July 2020

Guillaume Baudart, Martin Hirzel, Kiran Kate,
Parikshit Ram, and Avraham Shinnar

IBM Research AI

In [1]:

```
1 import pandas as pd
2 import lale.datasets
3 (train_X, train_y), (test_X, test_y) = lale.datasets.california_housing_df()
4 pd.concat([train_X.head(), train_y.head()], axis=1)
```

Out[1]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	target
0	3.2596	33.0	5.017657	1.006421	2300.0	3.691814	32.71	-117.03	1.030
1	3.8125	49.0	4.473545	1.041005	1314.0	1.738095	33.77	-118.16	3.821
2	4.1563	4.0	5.645833	0.985119	915.0	2.723214	34.66	-120.48	1.726
3	1.9425	36.0	4.002817	1.033803	1418.0	3.994366	32.69	-117.11	0.934
4	3.5542	43.0	6.268421	1.134211	874.0	2.300000	36.78	-119.80	0.965

Example Dataset

- sklearn California housing
- pandas dataframe
- schema for error-checking
- numeric features
  ➔ no cleaning / encoding
- numeric target
  ➔ use regression model

In [1]:

```
import pandas as pd
import lale.datasets
(train_X, train_y), (test_X, test_y) = lale.datasets.california_housing_df()
pd.concat([train_X.head(), train_y.head()], axis=1)
```

Out[1]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	target
0	3.2596	33.0	5.017657	1.006421	2300.0	3.691814	32.71	-117.03	1.030
1	3.8125	49.0	4.473545	1.041005	1314.0	1.738095	33.77	-118.16	3.821
2	4.1563	4.0	5.645833	0.985119	915.0	2.723214	34.66	-120.48	1.726
3	1.9425	36.0	4.002817	1.033803	1418.0	3.994366	32.69	-117.11	0.934
4	3.5542	43.0	6.268421	1.134211	874.0	2.300000	36.78	-119.80	0.965

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()
```

Importing Operators

- most directly from sklearn, also lale, xgboost
- wrap_imported_operators attaches schemas for type-driven Auto-ML

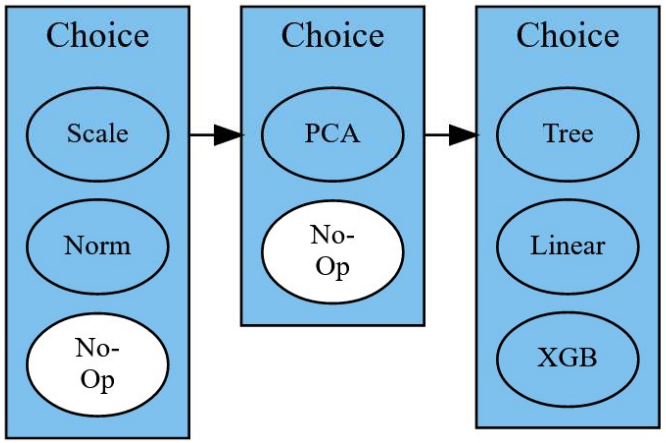
	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	target
0	3.2596	33.0	5.017657	1.006421	2300.0	3.691814	32.71	-117.03	1.030
1	3.8125	49.0	4.473545	1.041005	1314.0	1.738095	33.77	-118.16	3.821
2	4.1563	4.0	5.645833	0.985119	915.0	2.723214	34.66	-120.48	1.726
3	1.9425	36.0	4.002817	1.033803	1418.0	3.994366	32.69	-117.11	0.934
4	3.5542	43.0	6.268421	1.134211	874.0	2.300000	36.78	-119.80	0.965

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()
```



Planned Pipeline

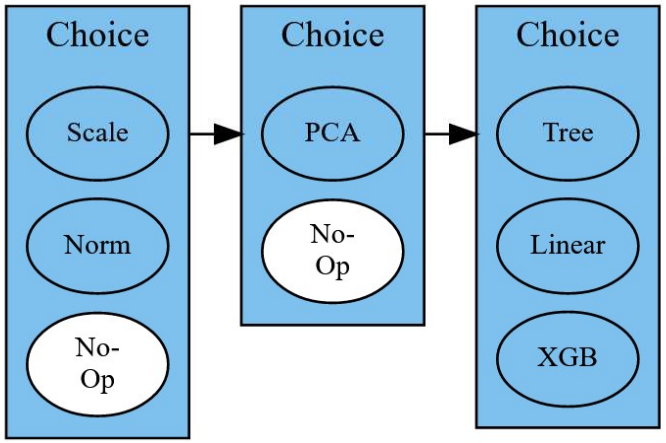
- pipe combinator **>>**
  for dataflow edges
- or combinator **|**
  for auto algorithm selection
- elided init args `XGB(...)`
  for hyperparameter tuning

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()
```



In [4]:

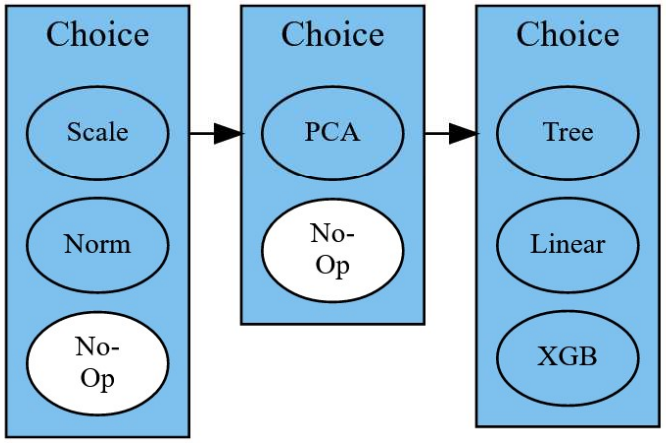
```
from lale.lib.lale import Hyperopt
import sklearn.metrics
r2 = sklearn.metrics.make_scorer(sklearn.metrics.r2_score)
trained_pipeline = planned_pipeline.auto_configure(
    train_X, train_y, optimizer=Hyperopt,
    scoring=r2, max_opt_time=10*60, max_eval_time=60, cv=3)
```

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

auto_configure

- type-driven:
  uses operator schemas
- creates search space for
  optimizer (here: Hyperopt)

```
planned_pipeline.visualize()
```



In [4]:

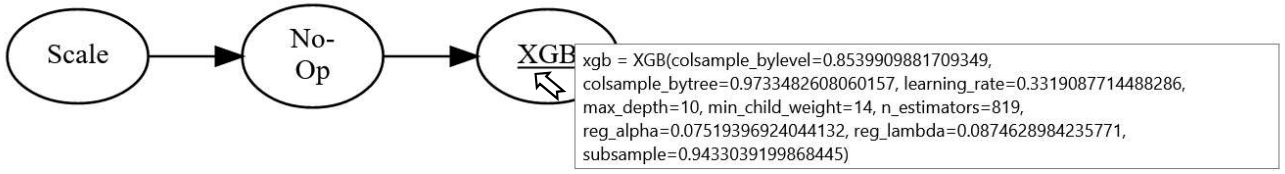
```
from lale.lib.lale import Hyperopt
import sklearn.metrics
r2 = sklearn.metrics.make_scorer(sklearn.metrics.r2_score)
trained_pipeline = planned_pipeline.auto_configure(
    train_X, train_y, optimizer=Hyperopt,
    scoring=r2, max_opt_time=10*60, max_eval_time=60, cv=3)
```

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

In [5]:

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

```
score: 0.83
```



Trained Pipeline

- concrete operators
- concrete hyperparameters (tooltip on mouse-over)
- scikit-learn compatible
- links to documentation

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]
```

In [5]:

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

```
R2 score: 0.83
```

In [6]:

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

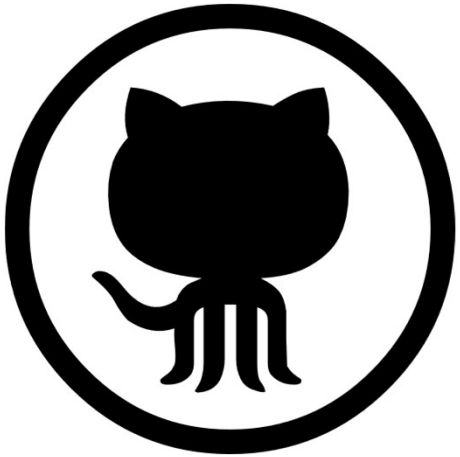
```
from lale.lib.sklearn.standard_scaler import Scale
from lale.lib.lale import NoOp
from lale.lib.xgboost.xgb_regressor import XGB
import lale
lale.wrap_imported_operators()

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
```

pretty_print

- consistent syntax for input and output or Auto-ML
- consistent syntax for manual ML and Auto-ML

Learn more

“**Lale: Consistent Automated Machine Learning**”, Guillaume Baudart, Martin Hirzel, Kiran Kate, Parikshit Ram, and Avraham Shinnar. KDD Workshop on Automation in Machine Learning (AutoML@KDD), August 2020.	• search space generation • error checking • higher-order operators • pipeline grammars
“**Mining Documentation to Extract Hyperparameter Schemas**”, Guillaume Baudart, Peter Kirchner, Martin Hirzel, and Kiran Kate. ICML Workshop on Automated Machine Learning (AutoML@ICML), July 2020.	• input: Python docstring • output: JSON schema



github.com/ibm/lale