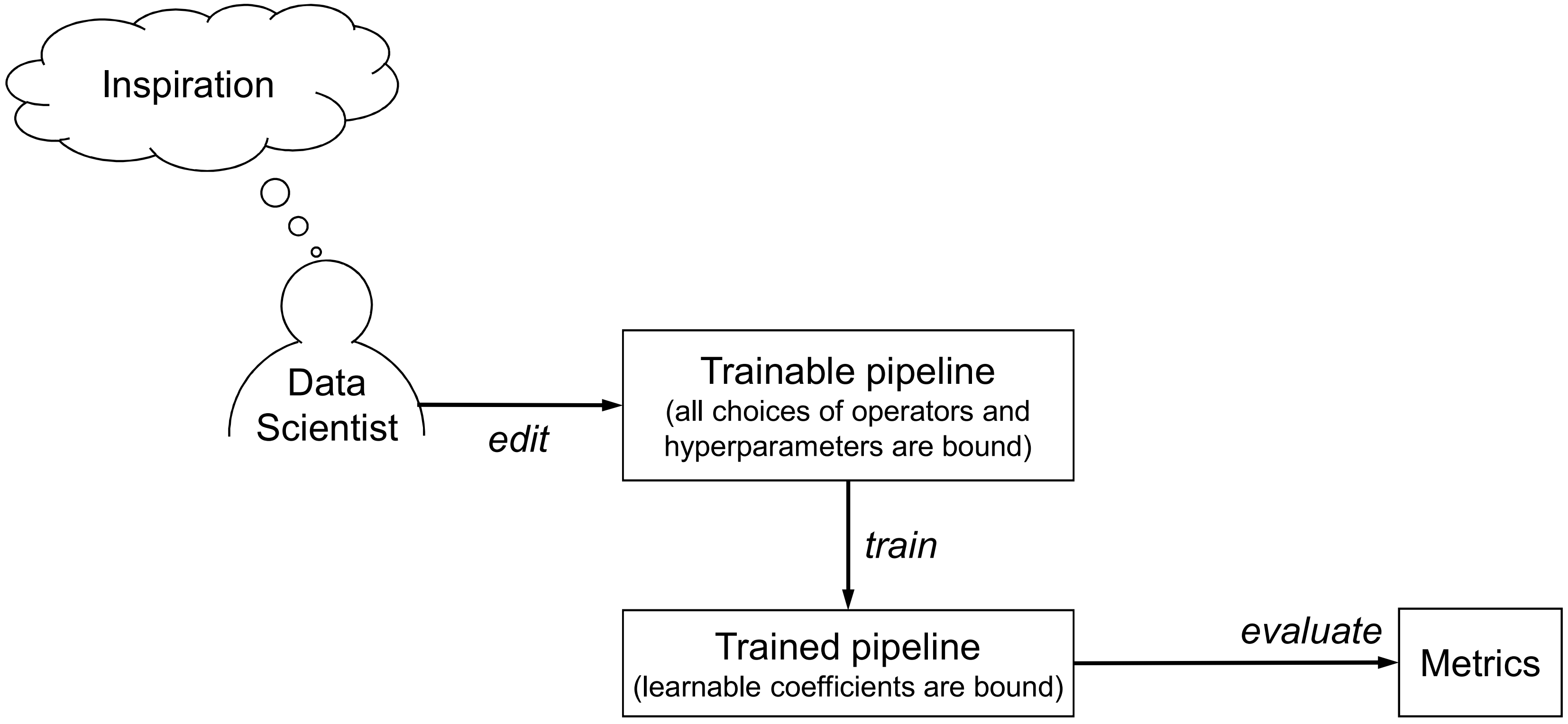
Type-Driven Automated Learning with LALE

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

Thursday 9 April 2020

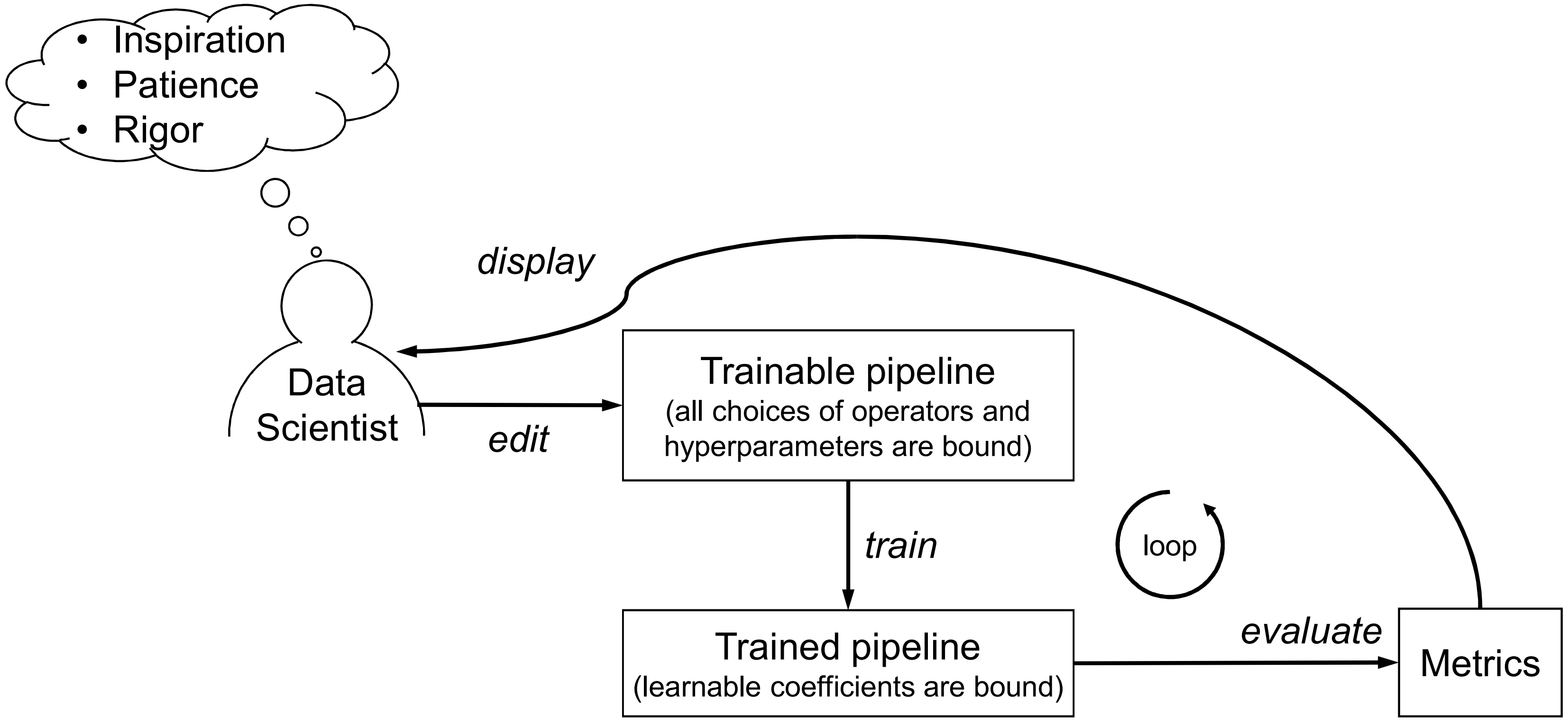
https://github.com/ibm/lale

Manual Machine Learning

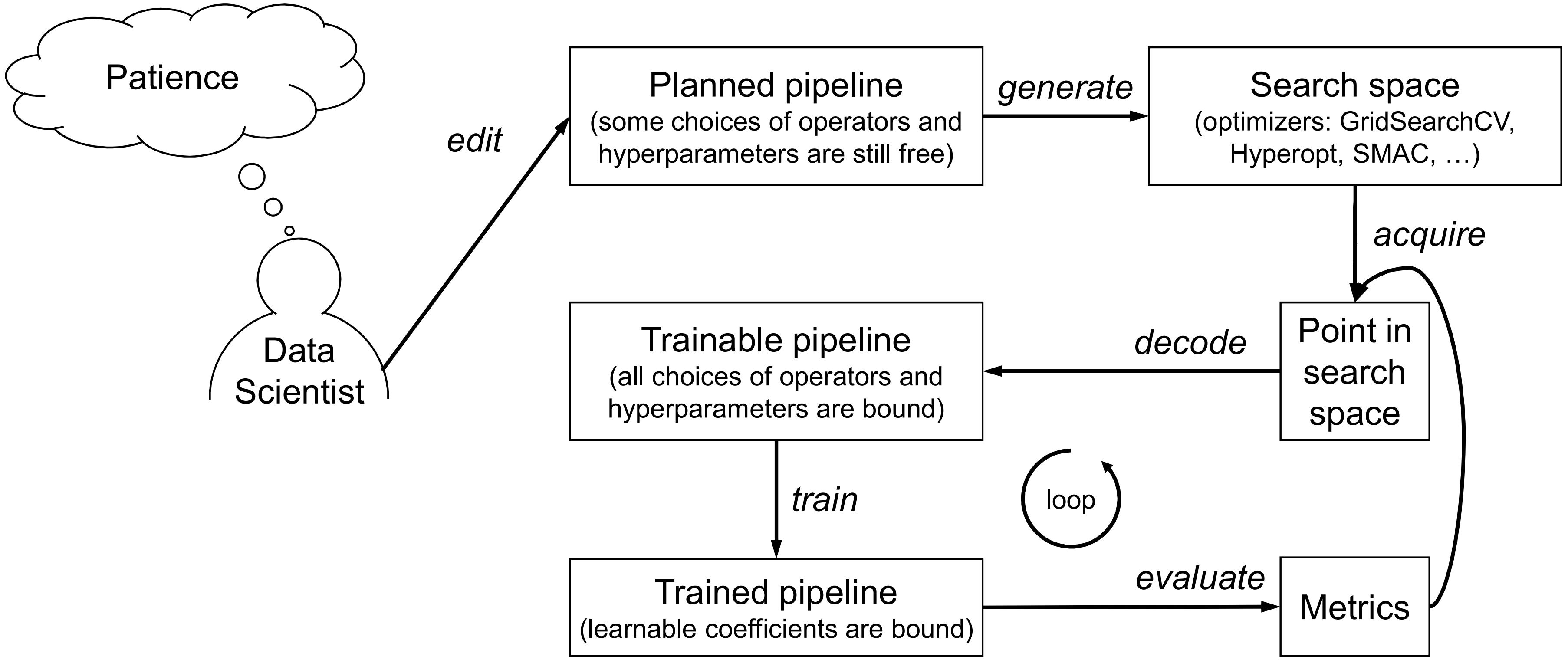


Inspiration
Data Scientist
edit
Trainable pipeline
(all choices of operators and hyperparameters are bound)
train
Trained pipeline
(learnable coefficients are bound)
evaluate
Metrics

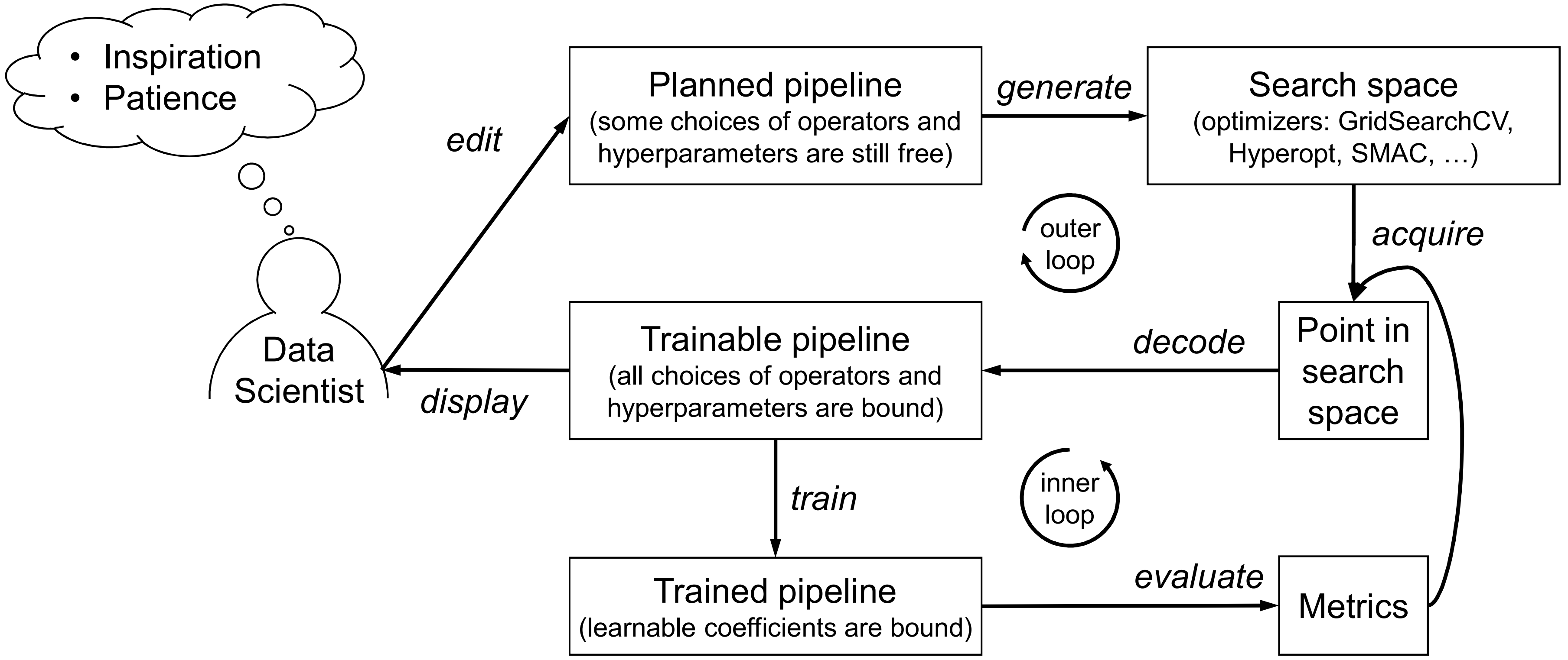
Iterated Manual Machine Learning



AutoML: Automated Machine Learning


Patience
Data Scientist
edit
Planned pipeline (some choices of operators and hyperparameters are still free)
generate
Search space (optimizers: GridSearchCV, Hyperopt, SMAC, …)
acquire
Point in search space
decode
Trainable pipeline (all choices of operators and hyperparameters are bound)
train
loop
Trained pipeline (learnable coefficients are bound)
evaluate
Metrics

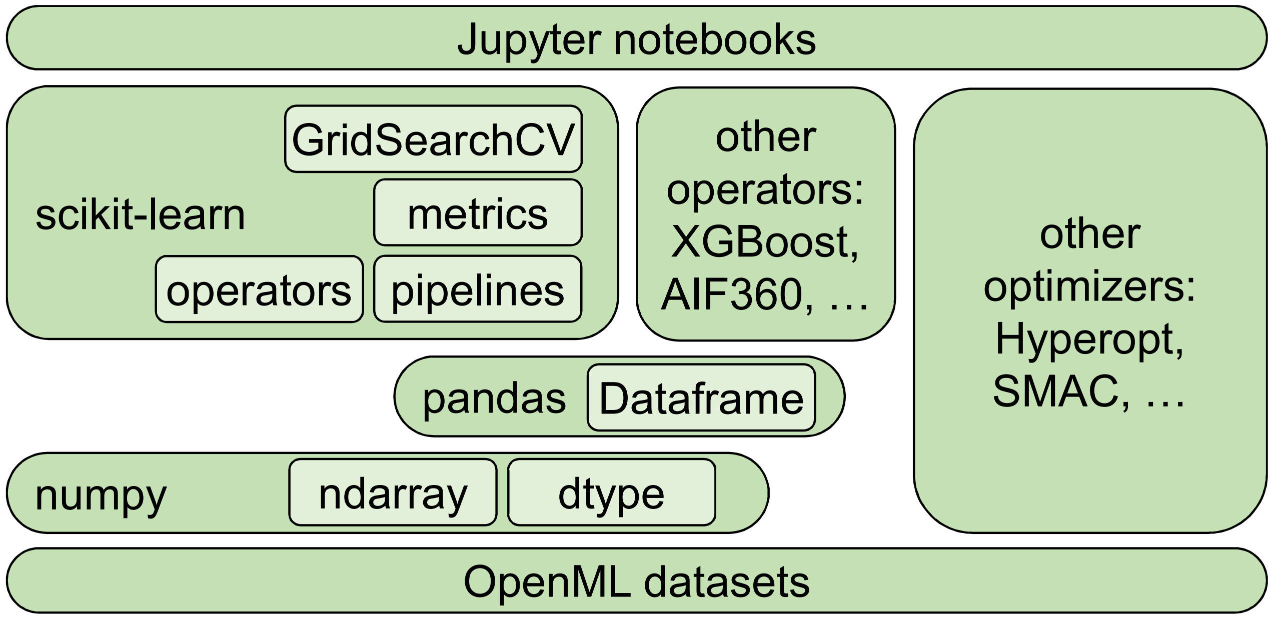
Iterated AutoML


• Inspiration
• Patience
Data Scientist
edit
Planned pipeline
(some choices of operators and hyperparameters are still free)
generate
Search space
(optimizers: GridSearchCV, Hyperopt, SMAC, …)
acquire
outer loop
Point in search space
decode
Trainable pipeline
(all choices of operators and hyperparameters are bound)
display
train
inner loop
Trained pipeline
(learnable coefficients are bound)
evaluate
Metrics

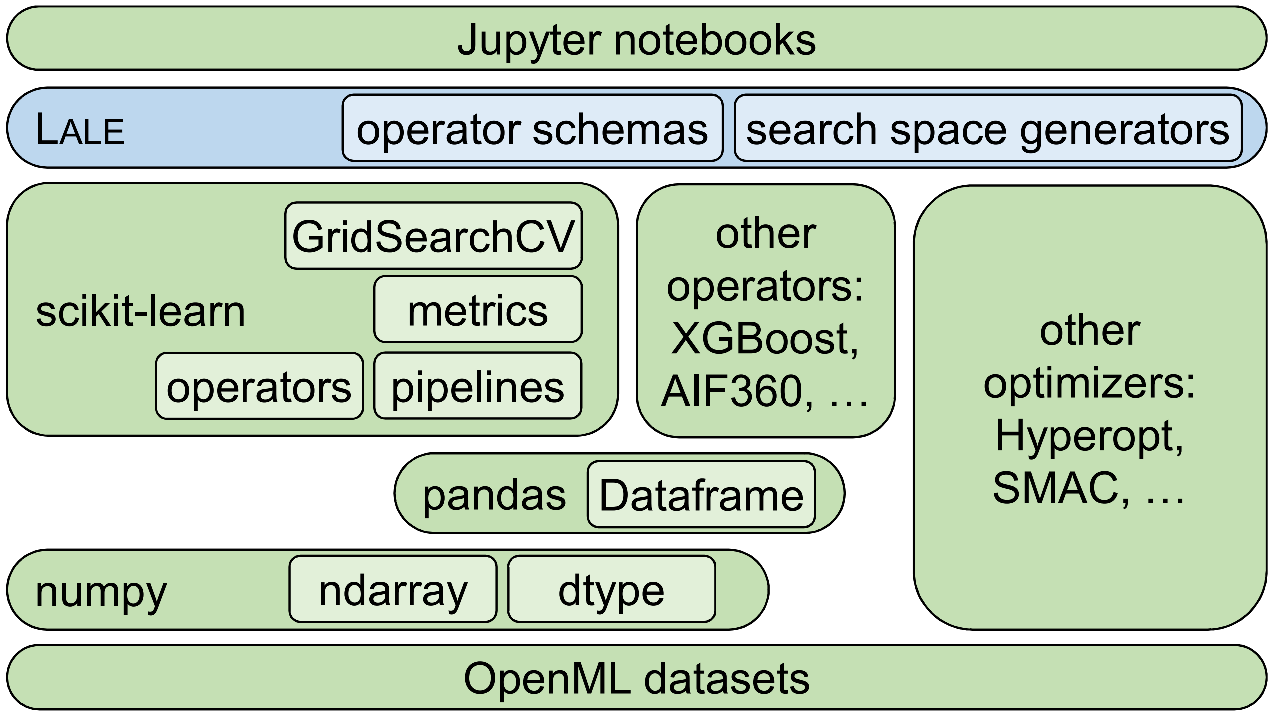
Problem Statement

- Library for semi-automated data science
- Easy-to-use automation
- Consistency of manual vs. automated experience
- Consistency across AutoML optimizers
- Display and iterative refinement
- Expert-level control for human inspiration

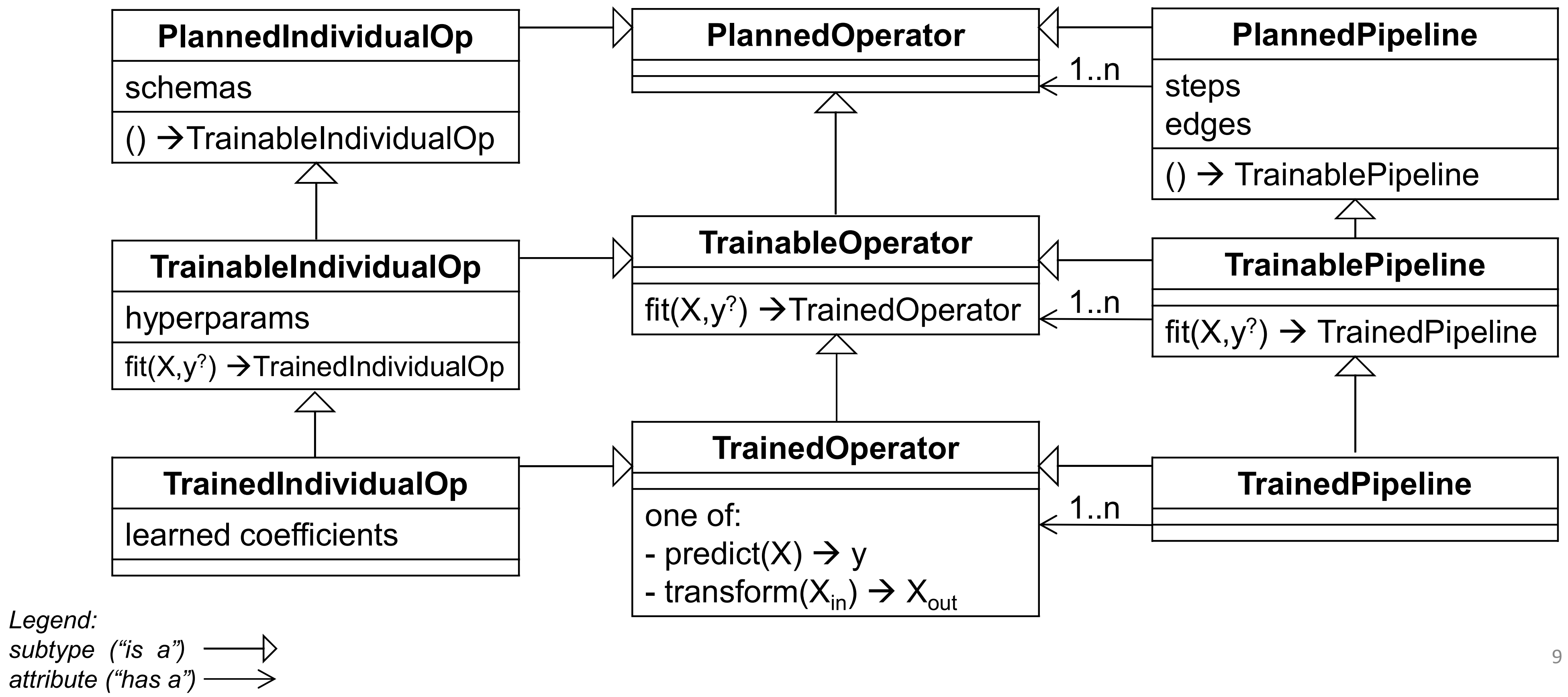
Open-Source AutoML Technologies



LALE's Place in the Open-Source Stack



Operators, Pipelines, and Lifecycle

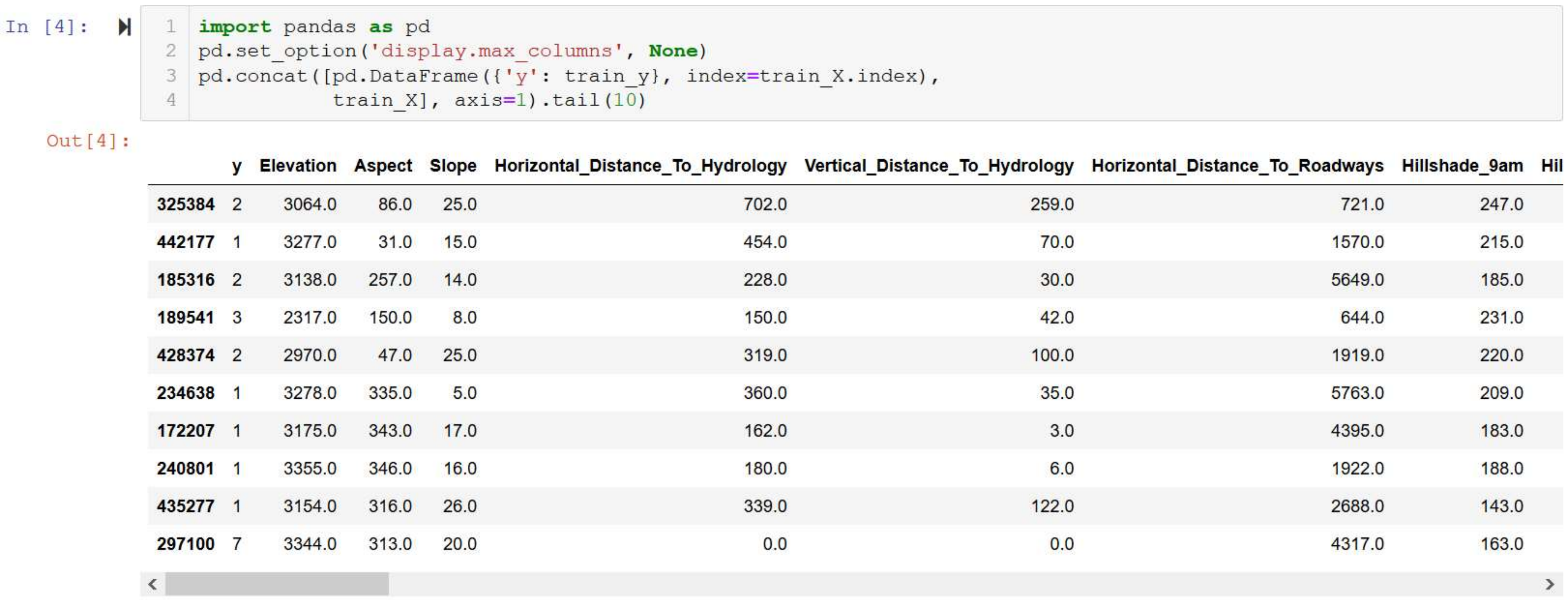


Example

- Dataset
- Manual machine learning
- Hyperparameter tuning
- Inspecting AutoML results
- Algorithm selection

https://nbviewer.jupyter.org/github/IBM/lale/blob/master/examples/talk_2020-0327-lale.ipynb

Example: Covertype Dataset



In [4]:

```
import pandas as pd
pd.set_option('display.max_columns', None)
pd.concat([pd.DataFrame({'y': train_y}, index=train_X.index),
           train_X], axis=1).tail(10)
```

Out[4]:

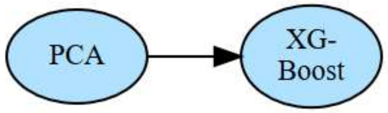
	y	Elevation	Aspect	Slope	Horizontal_Distance_To_Hydrology	Vertical_Distance_To_Hydrology	Horizontal_Distance_To_Roadways	Hillshade_9am	Hil
325384	2	3064.0	86.0	25.0	702.0	259.0	721.0	247.0	
442177	1	3277.0	31.0	15.0	454.0	70.0	1570.0	215.0	
185316	2	3138.0	257.0	14.0	228.0	30.0	5649.0	185.0	
189541	3	2317.0	150.0	8.0	150.0	42.0	644.0	231.0	
428374	2	2970.0	47.0	25.0	319.0	100.0	1919.0	220.0	
234638	1	3278.0	335.0	5.0	360.0	35.0	5763.0	209.0	
172207	1	3175.0	343.0	17.0	162.0	3.0	4395.0	183.0	
240801	1	3355.0	346.0	16.0	180.0	6.0	1922.0	188.0	
435277	1	3154.0	316.0	26.0	339.0	122.0	2688.0	143.0	
297100	7	3344.0	313.0	20.0	0.0	0.0	4317.0	163.0	

https://nbviewer.jupyter.org/github/IBM/lale/blob/master/examples/talk_2020-0327-lale.ipynb

Example: Manual Machine Learning

In [5]:
```
from sklearn.decomposition import PCA
from xgboost import XGBClassifier as XGBoost
lale.wrap_imported_operators()
```

In [6]:
```
manual_trainable = PCA(n_components=6) >> XGBoost(n_estimators=3)
manual_trainable.visualize()
```



In [7]:
```
%%time
manual_trained = manual_trainable.fit(train_X, train_y)
```

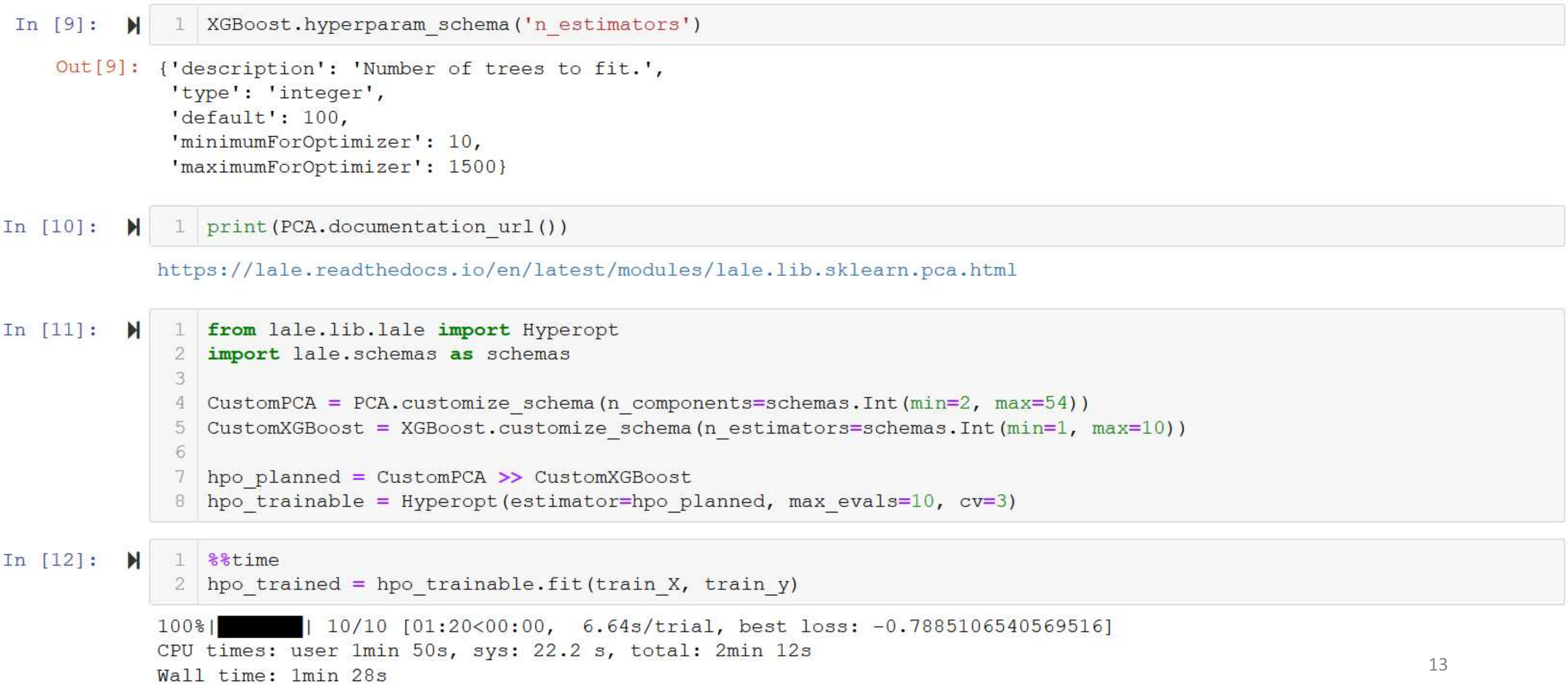
```
CPU times: user 2.34 s, sys: 1.2 s, total: 3.55 s
Wall time: 2.05 s
```

In [8]:
```
import sklearn.metrics
manual_y = manual_trained.predict(test_X)
print(f'accuracy {sklearn.metrics.accuracy_score(test_y, manual_y):.1%}')
```

```
accuracy 64.5%
```

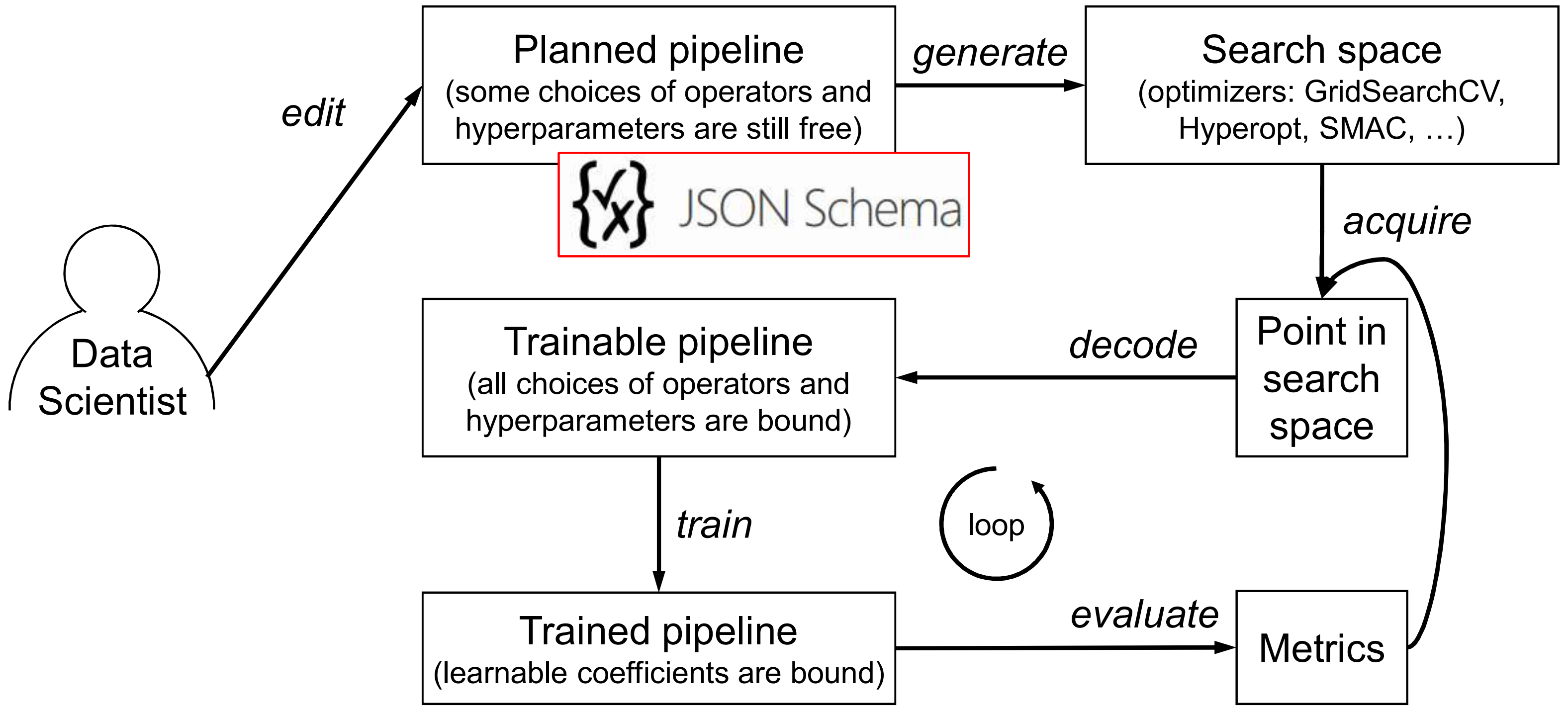
https://nbviewer.jupyter.org/github/IBM/lale/blob/master/examples/talk_2020-0327-lale.ipynb

Example: Hyperparameter Tuning



```
In [9]:  XGBoost.hyperparam_schema('n_estimators')

Out[9]:  {'description': 'Number of trees to fit.',
          'type': 'integer',
          'default': 100,
          'minimumForOptimizer': 10,
          'maximumForOptimizer': 1500}

In [10]: print(PCA.documentation_url())

         https://lale.readthedocs.io/en/latest/modules/lale.lib.sklearn.pca.html

In [11]: from lale.lib.lale import Hyperopt
         import lale.schemas as schemas

         CustomPCA = PCA.customize_schema(n_components=schemas.Int(min=2, max=54))
         CustomXGBoost = XGBoost.customize_schema(n_estimators=schemas.Int(min=1, max=10))

         hpo_planned = CustomPCA >> CustomXGBoost
         hpo_trainable = Hyperopt(estimator=hpo_planned, max_evals=10, cv=3)

In [12]: %%time
         hpo_trained = hpo_trainable.fit(train_X, train_y)

         100%|██████████| 10/10 [01:20<00:00,  6.64s/trial, best loss: -0.7885106540569516]
         CPU times: user 1min 50s, sys: 22.2 s, total: 2min 12s
         Wall time: 1min 28s
```

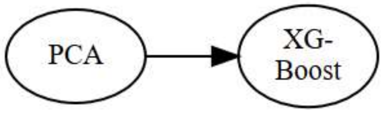
Types as Search Spaces


edit
Planned pipeline
(some choices of operators and hyperparameters are still free)
JSON Schema
generate
Search space
(optimizers: GridSearchCV, Hyperopt, SMAC, …)
acquire
Data Scientist
Trainable pipeline
(all choices of operators and hyperparameters are bound)
decode
Point in search space
train
loop
Trained pipeline
(learnable coefficients are bound)
evaluate
Metrics

https://nbviewer.jupyter.org/github/IBM/lale/blob/master/examples/talk_2020-0327-lale.ipynb

Example: Inspecting AutoML Results

In [13]:
```
hpo_y = hpo_trained.predict(test_X)
print(f'accuracy {sklearn.metrics.accuracy_score(test_y, hpo_y):.1%}')
```
```
accuracy 80.1%
```

In [14]:
```
hpo_trained.get_pipeline().visualize()
```


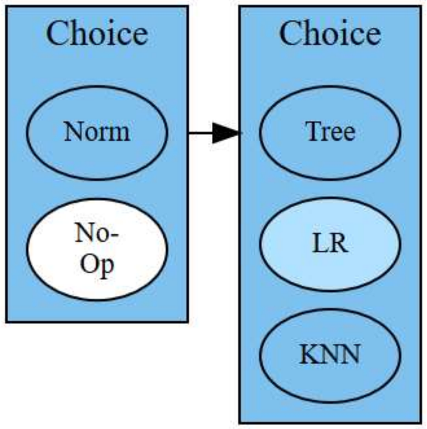
In [15]:
```
hpo_trained.get_pipeline().pretty_print(ipython_display=True)
```
```
from lale.lib.sklearn import PCA
from lale.lib.xgboost.xgb_classifier import XGBoost
import lale
lale.wrap_imported_operators()

pca = PCA(n_components=39, svd_solver='full')
xg_boost = XGBoost(colsample_bylevel=0.6016063807304212, colsample_bytree=0.7763972782064467, learning_rate=0.16389357351003786, max_depth=10, min_child_weight=5, n_estimators=5, reg_alpha=0.10485915855270356, reg_lambda=0.9268502695024392, subsample=0.4503841871781402)
pipeline = pca >> xg_boost
```

https://nbviewer.jupyter.org/github/IBM/lale/blob/master/examples/talk_2020-0327-lale.ipynb

Example: Algorithm Selection

In [19]:

```python
from sklearn.preprocessing import Normalizer as Norm
from sklearn.linear_model import LogisticRegression as LR
from sklearn.tree import DecisionTreeClassifier as Tree
from sklearn.neighbors import KNeighborsClassifier as KNN
from lale.lib.lale import NoOp
lale.wrap_imported_operators()

KNN = KNN.customize_schema(n_neighbors=schemas.Int(min=1, max=10))
transp_planned = (Norm | NoOp) >> (Tree | LR(dual=True) | KNN)
transp_planned.visualize()
```



https://nbviewer.jupyter.org/github/IBM/lale/blob/master/examples/talk_2020-0327-lale.ipynb

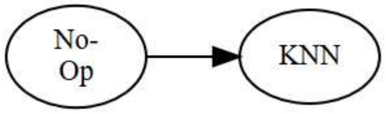
Example: Combined Algorithm Selection and Hyperparameter Tuning

In [20]:
```
%%time
transp_trained = transp_planned.auto_configure(
    train_X, train_y, optimizer=Hyperopt, cv=3, max_evals=3)
```

```
100%|██████████| 3/3 [01:48<00:00, 32.59s/trial, best loss: -0.8376392446578157]
CPU times: user 1min 50s, sys: 1.12 s, total: 1min 51s
Wall time: 1min 49s
```

In [21]:
```
transp_trained.pretty_print(ipython_display=True, show_imports=False)
transp_trained.visualize()
```

```
knn = KNN(algorithm='ball_tree', metric='manhattan', n_neighbors=9)
pipeline = NoOp() >> knn
```



In [22]:
```
%%time
transp_y = transp_trained.predict(test_X)
print(f'accuracy {sklearn.metrics.accuracy_score(test_y, transp_y):.1%}')
```

```
accuracy 86.6%
CPU times: user 50.6 s, sys: 15.6 ms, total: 50.6 s
Wall time: 50.7 s
```

Expert-Level Control

- Custom metrics
- Non-linear pipelines
- Higher-order operators
- Adding new operators

https://nbviewer.jupyter.org/github/IBM/lale/blob/master/examples/demo_aif360.ipynb

Custom Metrics

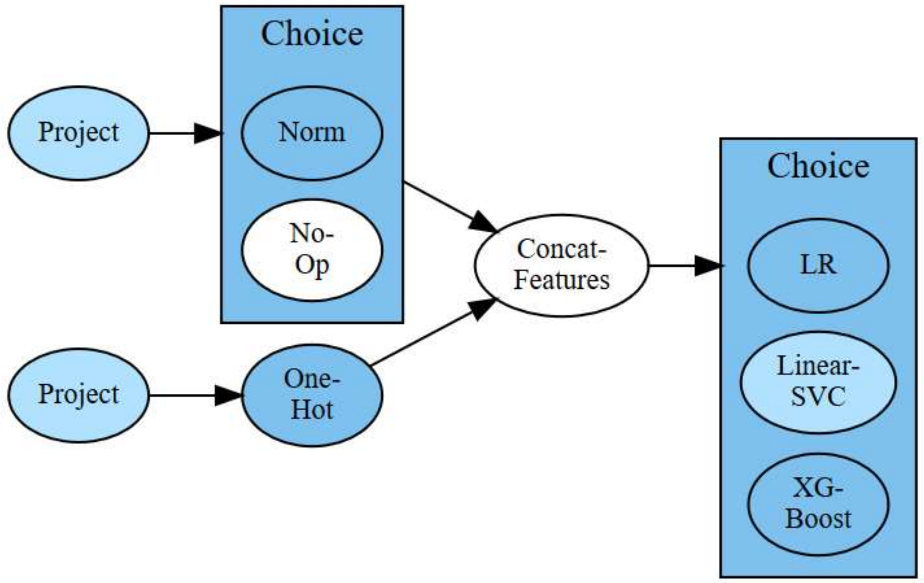
```python
def fairness_scorer(pipeline, X, y):
    from sklearn.metrics import accuracy_score
    from aif360.datasets import BinaryLabelDataset
    predictions = pipeline.predict(X)
    accuracy = accuracy_score(y, predictions)
    df = pd.concat([X, y], axis=1)
    dataset_pred = BinaryLabelDataset(favorable_label=1., unfavorable_label=2.,
                                      protected_attribute_names=['sex', 'age'],
                                      df=df, label_names=['credit'])
    fairness_metrics = aif360.metrics.BinaryLabelDatasetMetric(
        dataset_pred, unpr_groups, priv_groups)
    disparate_impact = fairness_metrics.disparate_impact()
    #Hyperopt minimizes (best_score - score_returned_by_scorer), choosing the values below based on that.
    if disparate_impact < 0.9 or 1.1 < disparate_impact:
        return -99
    else:
        return accuracy
```

```python
trained_fairer = planned_fairer.auto_configure(
    train_X, train_y, optimizer = Hyperopt, cv=3, max_evals=25, scoring=fairness_scorer, best_score=1.0)
```

https://nbviewer.jupyter.org/github/IBM/lale/blob/master/examples/talk_2020-0302-lale.ipynb

Non-Linear Pipelines

```
project_nums = Project(columns={'type': 'number'})
project_cats = Project(columns={'type': 'string'})
planned_pipeline = (
         (project_nums >> (Norm | NoOp) & project_cats >> OneHot)
     >> ConcatFeatures
     >> (LR | LinearSVC(dual=False)| XGBoost))
planned_pipeline.visualize()
```



Pipeline Combinators

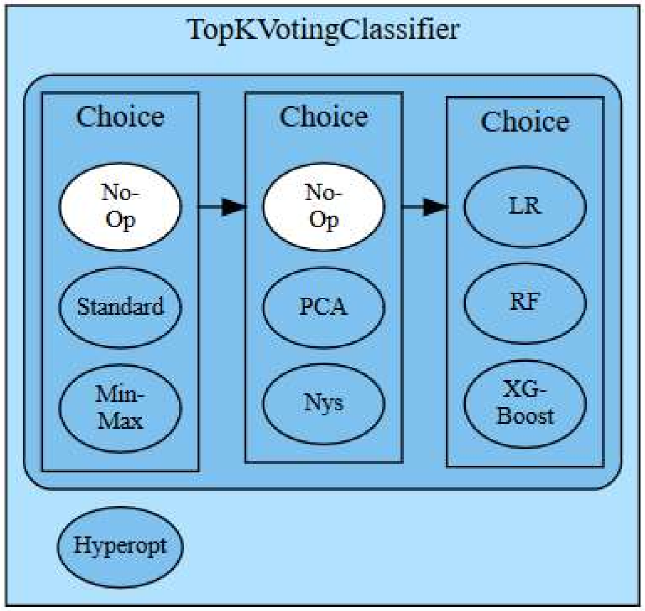
LALE features	Name	Description	Scikit-learn features
`>>` or `make_pipeline`	pipe	feed to next	`make_pipeline`
`&` or `make_union`	and	run both	`make_union` or `ColumnTransformer`
`|` or `make_choice`	or	choose one	N/A (specific to given Auto-ML tool)

https://nbviewer.jupyter.org/github/IBM/lale/blob/master/examples/demo_top_k_voting_classifier.ipynb

Higher-Order Operators (1/2)

In [4]:

```
planned_pipeline = (NoOp | Standard | MinMax) >> (NoOp | PCA | Nys) >> (LR | RF | XGBoost)
ensemble = TopKVotingClassifier(
    estimator=planned_pipeline, k=3, optimizer=Hyperopt,
    args_to_optimizer={'max_evals':25, 'scoring':'accuracy'})
ensemble.visualize()
```



In [5]:

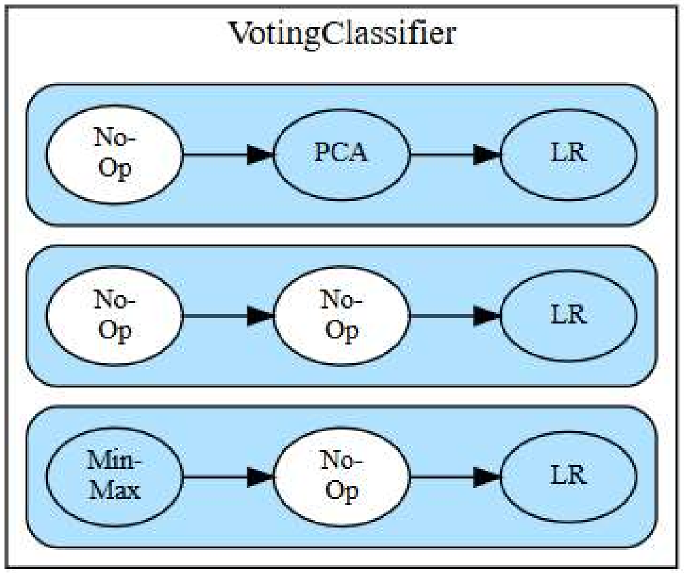
```
trained_ensemble = ensemble.fit(train_X, train_y)
```

https://nbviewer.jupyter.org/github/IBM/lale/blob/master/examples/demo_top_k_voting_classifier.ipynb

Higher-Order Operators (2/2)

In [8]:

```
best_pipeline = trained_ensemble.get_pipeline()
best_pipeline.visualize()
```



Adding New Operators

- https://nbviewer.jupyter.org/github/IBM/lale/blob/master/examples/docs_new_operators.ipynb
  1. Write class with `__init__`, `fit`, and `predict` or `transform`
  2. Write JSON schemas for hyperparameters and datasets
  3. Call `lale.operators.make_operator`
  4. Write tests for your new operator
- Currently supported operators

Package	Count	Description
lale.lib.sklearn	45	Hand-curated scikit-learn operators
lale.lib.autogen	115	Auto-extracted scikit-learn operators
lale.lib.xgboost	2	Gradient-boosted random forests
lale.lib.aif360	1	Fairness mitigator (incomplete)
… other lale.lib	25	… other packages

Type-Driven Automated Learning

- Types as search spaces
  - E.g., continuous hyperparameter range
  - E.g., categorical hyperparameter enum
- Types as documentation
  - E.g., lale.readthedocs.io/en/latest/modules/lale.lib.sklearn.pca.html
- Types for feature engineering
  - E.g., Project(columns={'type': 'string’})
- Types for error checking
  - E.g., datasets, hyperparameters, constraints

Conclusion

- Library for semi-automated data science
  - https://github.com/ibm/lale
- Suggested actions:
  - Try it out and send us feedback
  - Contribute new operators
  - Do original research (AutoML optimizers, AI fairness, fast AutoML, …)