Type-Driven Automated Learning with LALE

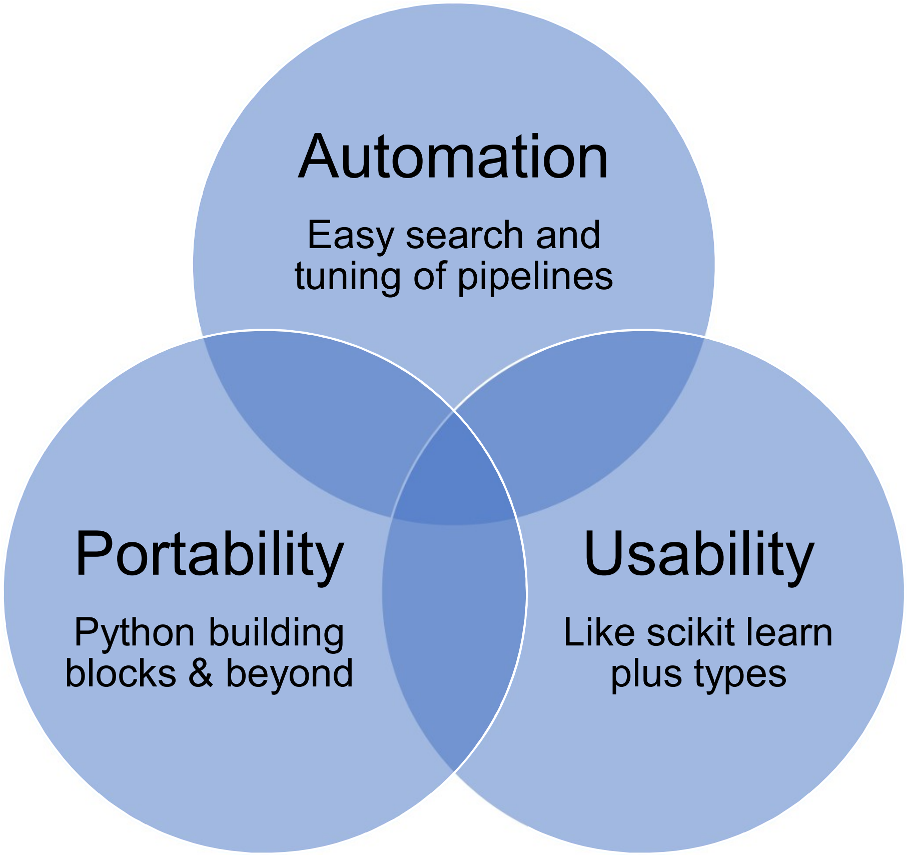
Martin Hirzel, Kiran Kate, Avi Shinnar, Subhrajit Roy, Pari Ram, and Guillaume Baudart

Wednesday 29 May 2019

Global Data Scientist Profession
1/2 Day Conference

Value Proposition

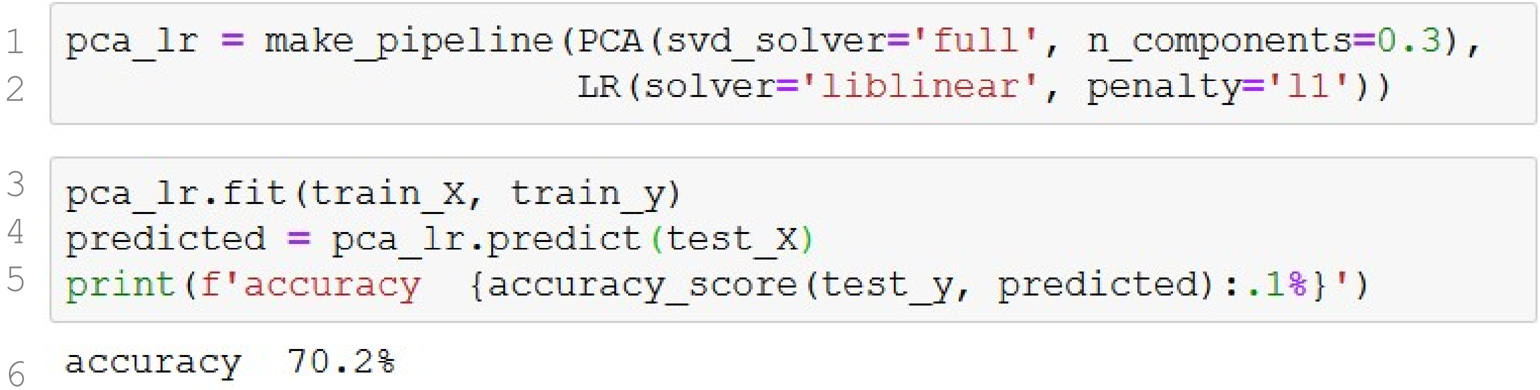
Augment, but don't replace, the data scientist.



Automation
Easy search and tuning of pipelines
Portability
Python building blocks & beyond
Usability
Like scikit learn plus types

Manual ML with Sklearn

Prior work: scikit learn, popular machine learning package



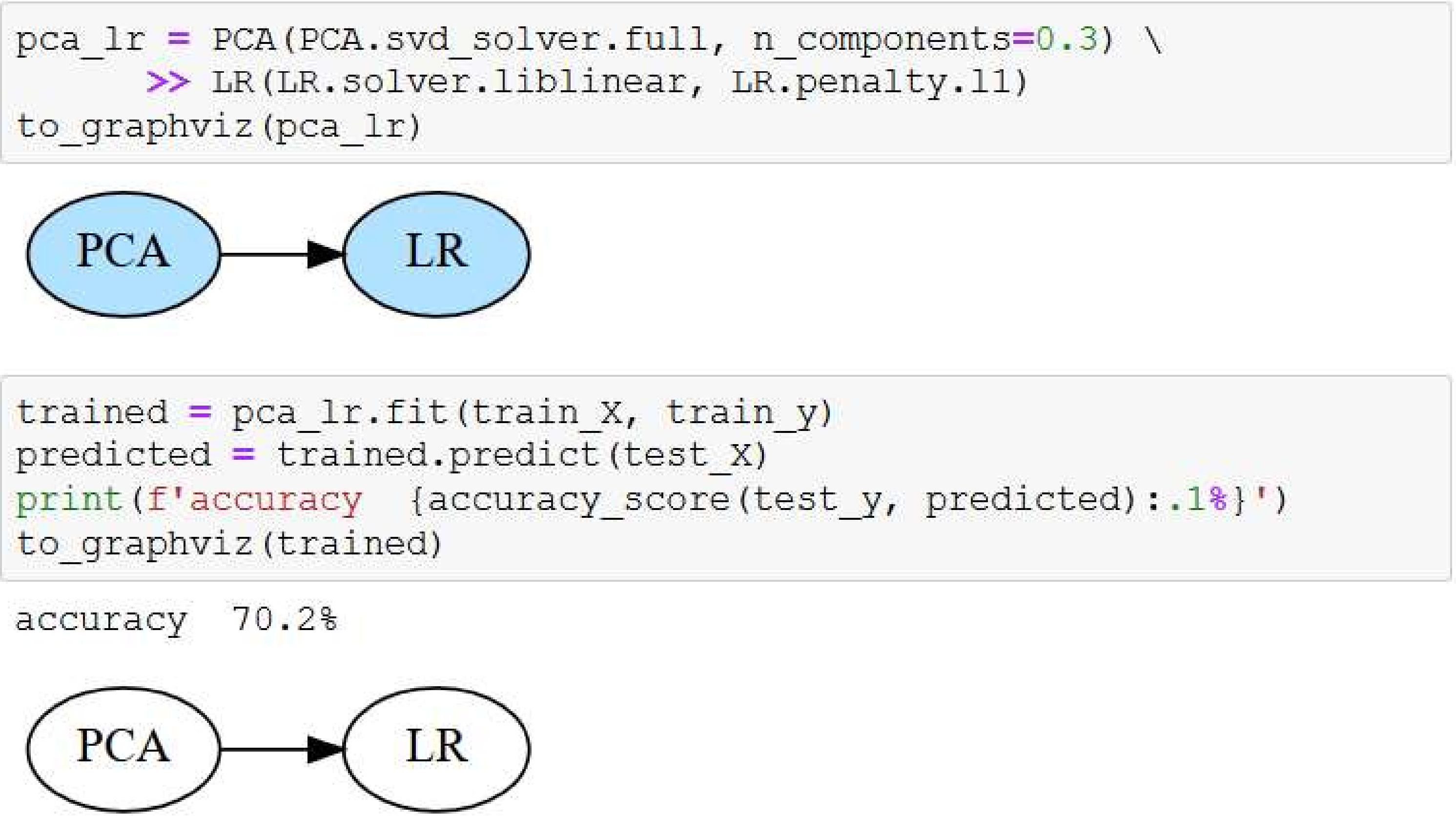
```
pca_lr = make_pipeline(PCA(svd_solver='full', n_components=0.3),
                       LR(solver='liblinear', penalty='l1'))
```

```
pca_lr.fit(train_X, train_y)
predicted = pca_lr.predict(test_X)
print(f'accuracy  {accuracy_score(test_y, predicted):.1%}')
```

```
accuracy  70.2%
```

Manual ML with LALE

Our work: Language for Automated Learning Exploration



```
pca_lr = PCA(PCA.svd_solver.full, n_components=0.3) \
       >> LR(LR.solver.liblinear, LR.penalty.l1)
to_graphviz(pca_lr)
```

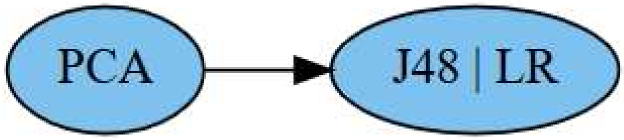
```
trained = pca_lr.fit(train_X, train_y)
predicted = trained.predict(test_X)
print(f'accuracy  {accuracy_score(test_y, predicted):.1%}')
to_graphviz(trained)
```

```
accuracy  70.2%
```

Automated ML with LALE

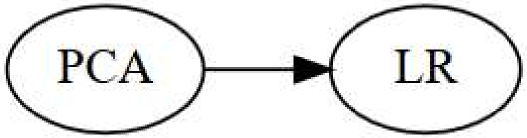
Combined algorithm selection and hyperparameter tuning

```
planned = PCA >> (J48 | LR)
to_graphviz(planned)
```



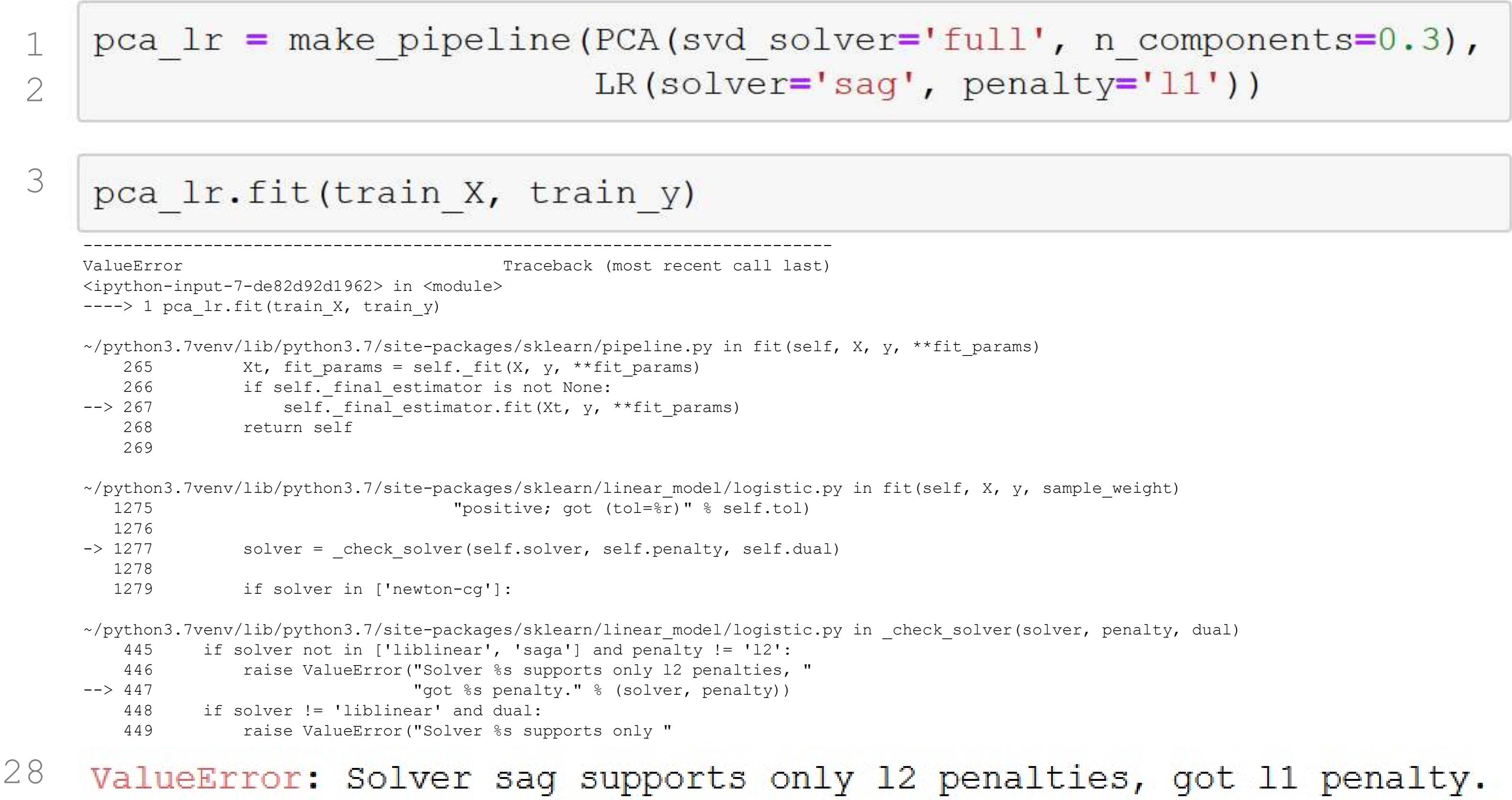
```
hyperopt_classifier = HyperoptClassifier(planned, max_evals=5)
best_found = hyperopt_classifier.fit(train_X, train_y)
predicted = best_found.predict(test_X)
print(f'accuracy  {accuracy_score(test_y, predicted):.1%}')
to_graphviz(best_found)
```

```
accuracy  96.4%
```



Constraints in Manual ML

Conditional hyperparameters



```
pca_lr = make_pipeline(PCA(svd_solver='full', n_components=0.3),
                       LR(solver='sag', penalty='l1'))
```

```
pca_lr.fit(train_X, train_y)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-7-de82d92d1962> in <module>
----> 1 pca_lr.fit(train_X, train_y)

~/python3.7venv/lib/python3.7/site-packages/sklearn/pipeline.py in fit(self, X, y, **fit_params)
    265         Xt, fit_params = self._fit(X, y, **fit_params)
    266         if self._final_estimator is not None:
--> 267             self._final_estimator.fit(Xt, y, **fit_params)
    268         return self
    269 

~/python3.7venv/lib/python3.7/site-packages/sklearn/linear_model/logistic.py in fit(self, X, y, sample_weight)
   1275                              "positive; got (tol=%r)" % self.tol)
   1276 
-> 1277         solver = _check_solver(self.solver, self.penalty, self.dual)
   1278 
   1279         if solver in ['newton-cg']:

~/python3.7venv/lib/python3.7/site-packages/sklearn/linear_model/logistic.py in _check_solver(solver, penalty, dual)
    445     if solver not in ['liblinear', 'saga'] and penalty != 'l2':
    446         raise ValueError("Solver %s supports only l2 penalties, "
--> 447                          "got %s penalty." % (solver, penalty))
    448     if solver != 'liblinear' and dual:
    449         raise ValueError("Solver %s supports only "

ValueError: Solver sag supports only l2 penalties, got l1 penalty.
```

Constraints in AutoML

Problem: Some automated iterations raise exceptions

Solution 1: Unconstrained search space

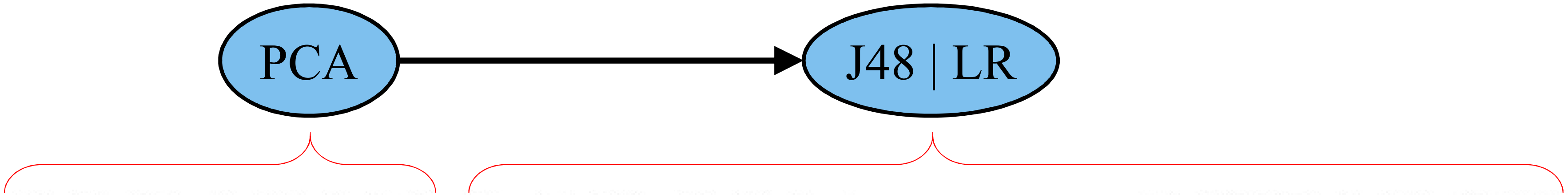
- $\{S{:}[\mathit{linear},\mathit{sag},\mathit{lbfgs}],\ P{:}\ [\mathit{l1},\mathit{l2}]\}$
- Catch exception
- Return made-up loss `np.float.max`

Solution 2: Constrained search space

- $\{S{:}[\mathit{linear},\mathit{sag},\mathit{lbfgs}],\ P{:}\ [\mathit{l1},\mathit{l2}]\}$ **and** (**if** $S{:}\ [\mathit{sag},\mathit{lbfgs}]$ **then** $P{:}\ [\mathit{l2}]$)
- No exceptions
- No made-up loss

GridSearchCV Search Space

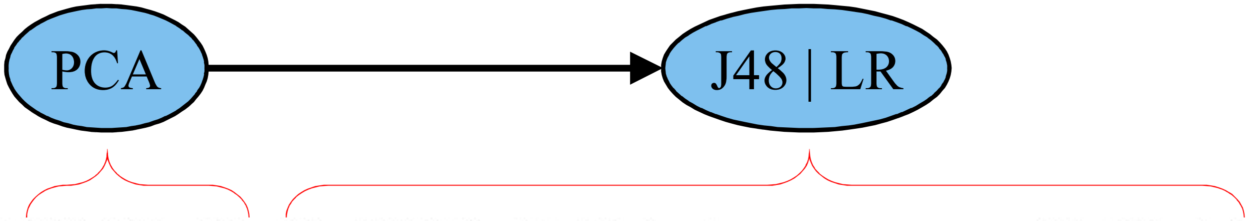
AutoML included with Sklearn



$$\begin{array}{lllll}
 & \mathrm{dict}\{N\colon [0.21, 0.65, 0.84], & D\colon [\mathit{J48}], R\colon [\mathit{false}], & C\colon [0.07, 0.30, 0.89] & \} \\
\vee & \mathrm{dict}\{N\colon [0.21, 0.65, 0.84], & D\colon [\mathit{J48}], R\colon [\mathit{true}, \mathit{false}], & C\colon [0.25] & \} \\
\vee & \mathrm{dict}\{N\colon [\mathit{mle}], & D\colon [\mathit{J48}], R\colon [\mathit{false}], & C\colon [0.07, 0.30, 0.89] & \} \\
\vee & \mathrm{dict}\{N\colon [\mathit{mle}], & D\colon [\mathit{J48}], R\colon [\mathit{true}, \mathit{false}], & C\colon [0.25] & \} \\
\vee & \mathrm{dict}\{N\colon [0.21, 0.65, 0.84], & D\colon [\mathit{LR}], \ S\colon [\mathit{linear}], & P\colon [\mathit{l1}, \mathit{l2}] & \} \\
\vee & \mathrm{dict}\{N\colon [0.21, 0.65, 0.84], & D\colon [\mathit{LR}], \ S\colon [\mathit{linear}, \mathit{sag}, \mathit{lbfgs}], & P\colon [\mathit{l2}] & \} \\
\vee & \mathrm{dict}\{N\colon [\mathit{mle}], & D\colon [\mathit{LR}], \ S\colon [\mathit{linear}], & P\colon [\mathit{l1}, \mathit{l2}] & \} \\
\vee & \mathrm{dict}\{N\colon [\mathit{mle}], & D\colon [\mathit{LR}], \ S\colon [\mathit{linear}, \mathit{sag}, \mathit{lbfgs}], & P\colon [\mathit{l2}] & \}
\end{array}$$

SMAC Search Space

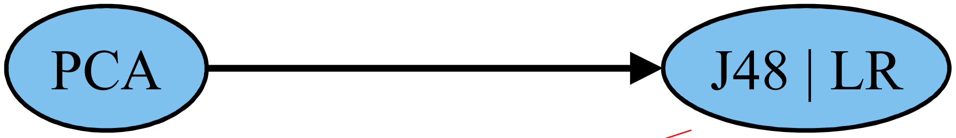
Sequential Model-based Algorithm Configuration



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

Hyperopt Search Space

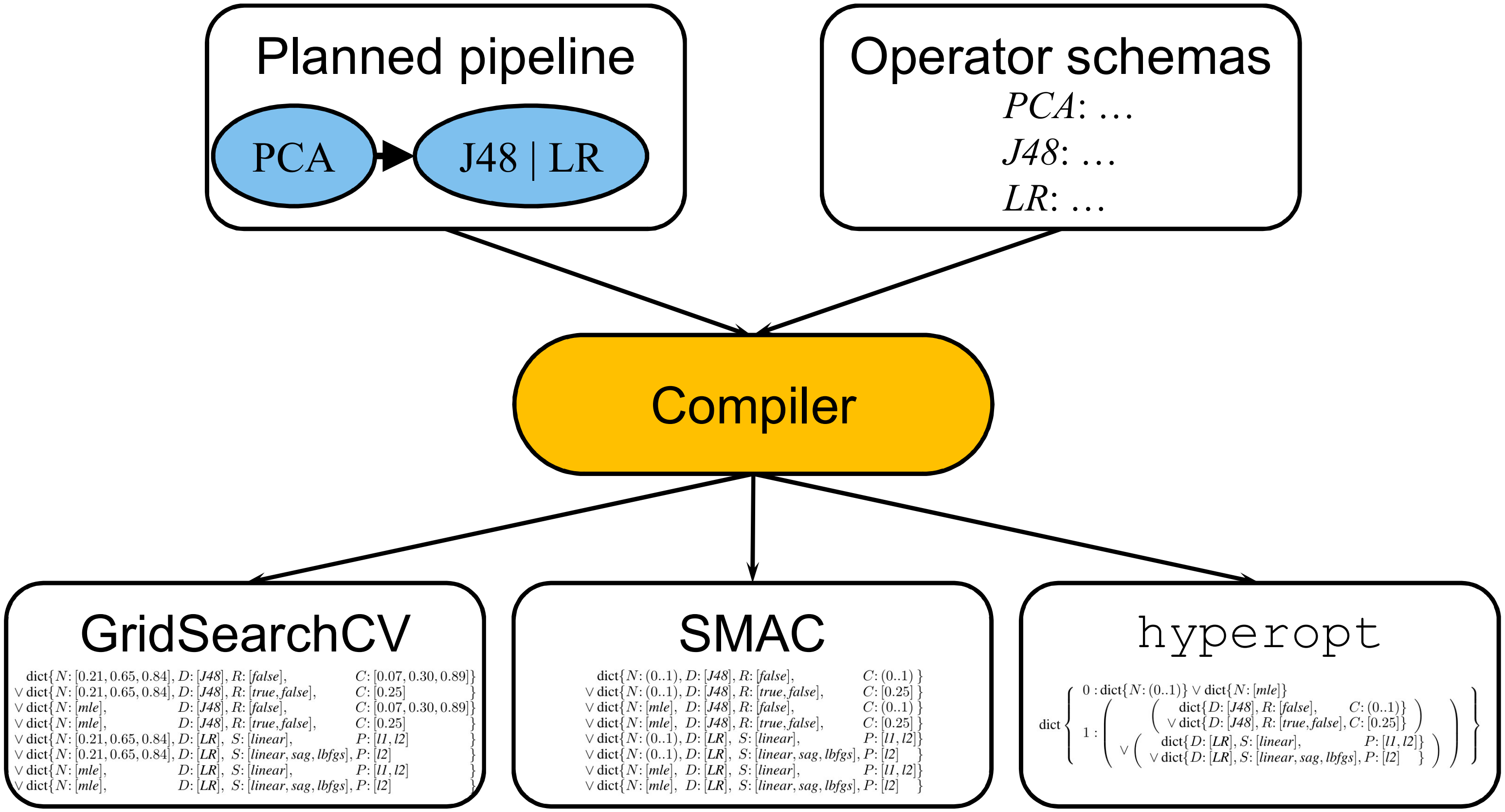
Supports parallel search



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

Types as Search Spaces

LALE auto-generates search spaces for AutoML tools



GridSearchCV

$$\begin{array}{lllll}
 & \text{dict}\{N\colon[0.21,0.65,0.84], & D\colon[\mathit{J48}], R\colon[\mathit{false}], & C\colon[0.07,0.30,0.89] & \} \\
\vee & \text{dict}\{N\colon[0.21,0.65,0.84], & D\colon[\mathit{J48}], R\colon[\mathit{true},\mathit{false}], & C\colon[0.25] & \} \\
\vee & \text{dict}\{N\colon[\mathit{mle}], & D\colon[\mathit{J48}], R\colon[\mathit{false}], & C\colon[0.07,0.30,0.89] & \} \\
\vee & \text{dict}\{N\colon[\mathit{mle}], & D\colon[\mathit{J48}], R\colon[\mathit{true},\mathit{false}], & C\colon[0.25] & \} \\
\vee & \text{dict}\{N\colon[0.21,0.65,0.84], & D\colon[\mathit{LR}], S\colon[\mathit{linear}], & P\colon[\mathit{l1},\mathit{l2}] & \} \\
\vee & \text{dict}\{N\colon[0.21,0.65,0.84], & D\colon[\mathit{LR}], S\colon[\mathit{linear},\mathit{sag},\mathit{lbfgs}], & P\colon[\mathit{l2}] & \} \\
\vee & \text{dict}\{N\colon[\mathit{mle}], & D\colon[\mathit{LR}], S\colon[\mathit{linear}], & P\colon[\mathit{l1},\mathit{l2}] & \} \\
\vee & \text{dict}\{N\colon[\mathit{mle}], & D\colon[\mathit{LR}], S\colon[\mathit{linear},\mathit{sag},\mathit{lbfgs}], & P\colon[\mathit{l2}] & \}
\end{array}$$

SMAC

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

hyperopt

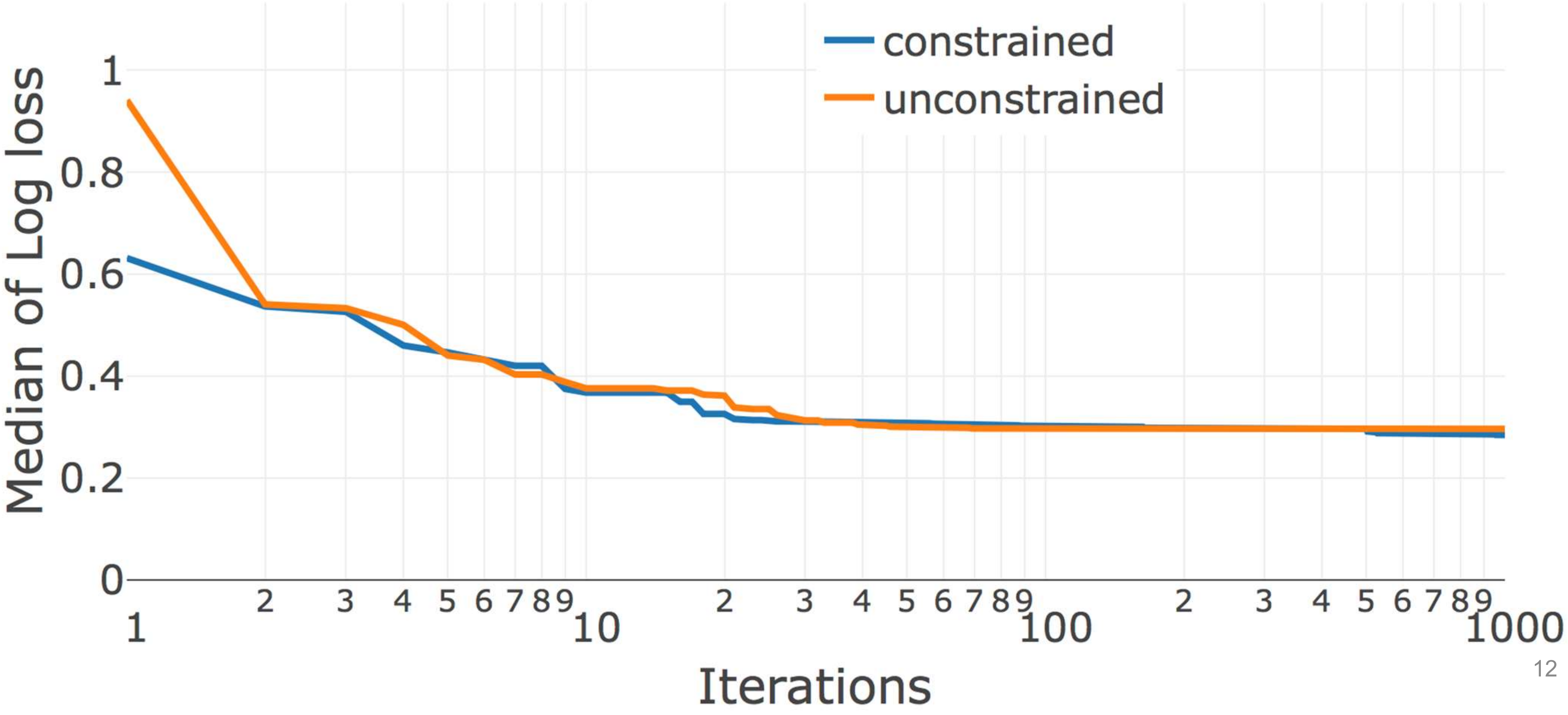
$$\text{dict}\left\{\begin{array}{l}
0: \text{dict}\{N\colon(0..1)\} \vee \text{dict}\{N\colon[\mathit{mle}]\} \\
1: \left(\begin{array}{l}
\left(\begin{array}{l} \text{dict}\{D\colon[\mathit{J48}], R\colon[\mathit{false}], C\colon(0..1)\} \\ \vee\,\text{dict}\{D\colon[\mathit{J48}], R\colon[\mathit{true},\mathit{false}],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

Search Convergence (1/3)

LR | KNN

Car dataset

hyperopt

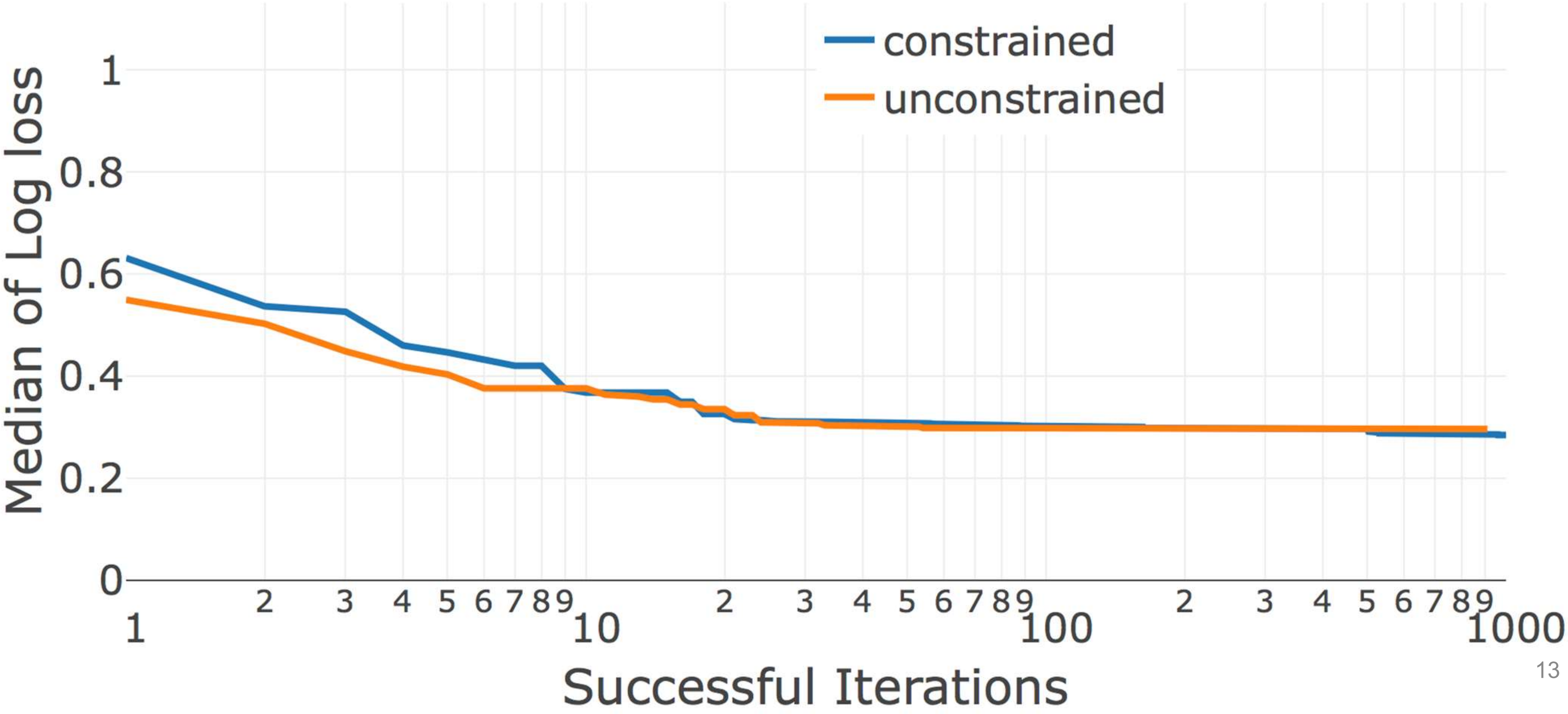


Search Convergence (2/3)

LR | KNN

Car dataset

hyperopt

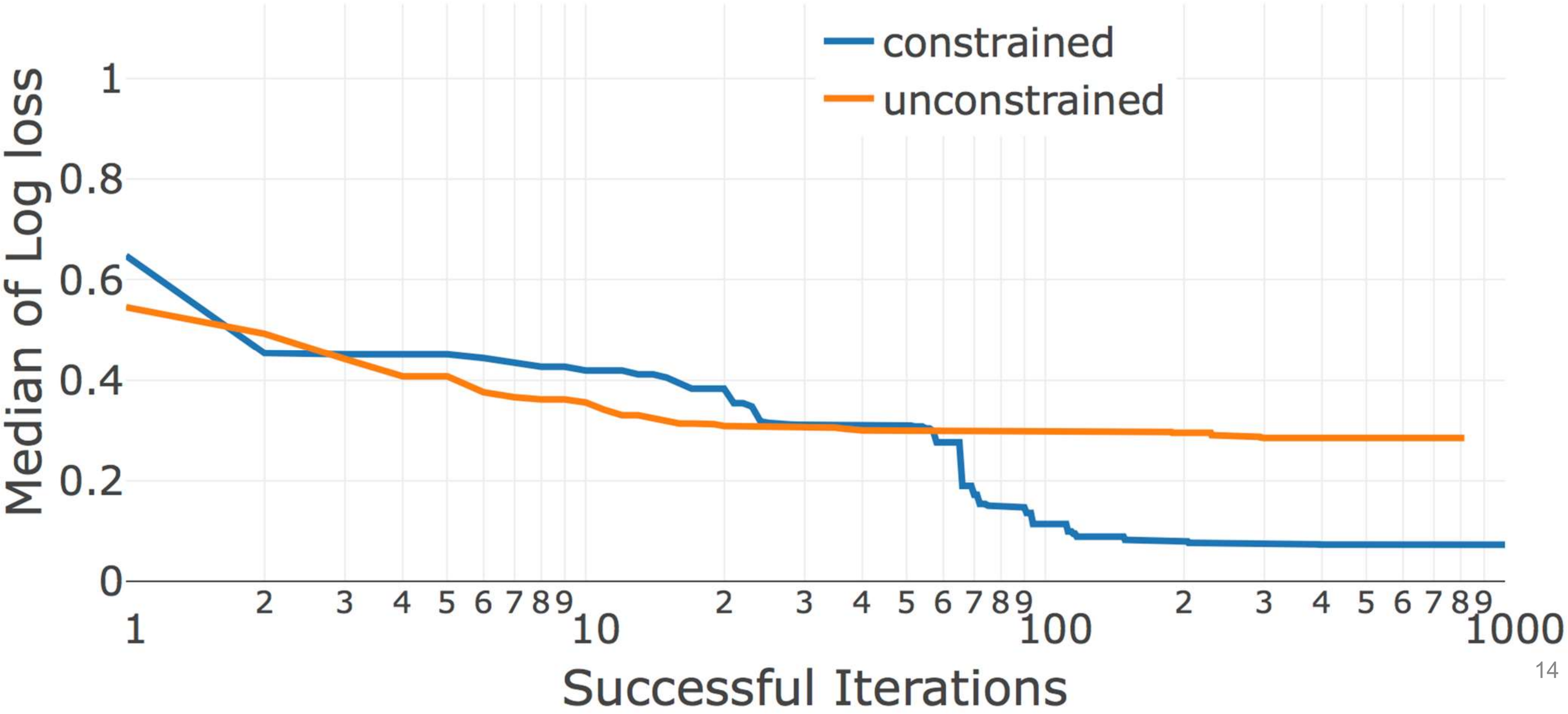


Search Convergence (3/3)

J48 | LR | KNN

Car dataset

hyperopt

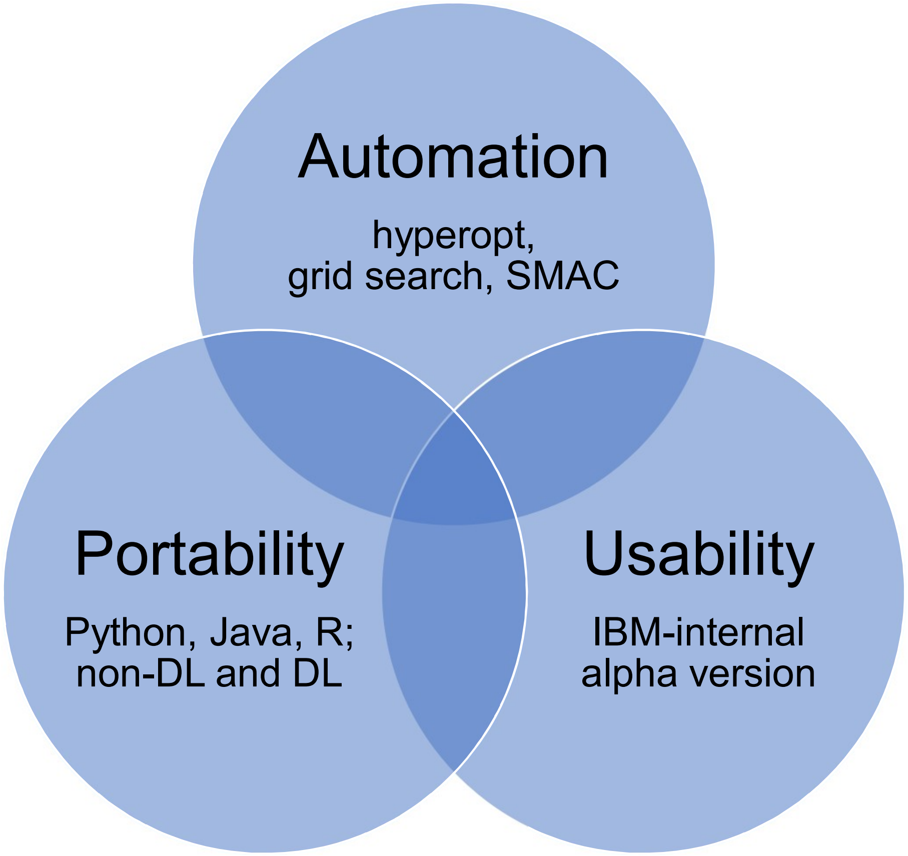


Portability

Modality	Dataset	Pipeline (**bold: best found choice**)
Text	Movie reviews (sentiment analysis)	(**BERT** | TFIDF) >> (**LR** | MLP | KNN | SVC | PAC)
Table	Car (structured with categorical features)	**J48** | ArulesCBA | LR | KNN
Images	CIFAR-10 (image classification)	**ResNet50**
Time-series	Epilepsy (seizure classification)	**WindowTransformer** >> (**KNN** | XGBoost | LR) >> **Voting**

Status

https://github.ibm.com/aimodels/lale



Automation

hyperopt,
grid search, SMAC

Portability

Python, Java, R;
non-DL and DL

Usability

IBM-internal
alpha version