Towards the Assisted Design of Data Science Pipelines

Sergey Redyuk (TU Berlin, sergey.redyuk@tu-berlin.de)

Nov 24, 2022

End-to-End Management of Experimental Data Science on Biomedical Molecular Data

Sergey Redyuk, 4th year (2018-2023)

Supervisors:
Prof. Volker Markl (TU Berlin)
Prof. Uwe Ohler (MDC)
Dr. Zoi Kaoudi (TU Berlin)
Prof. Sebastian Schelter (University of Amsterdam)

https://sergred.github.io/

Goal Assist end-users in performing data science tasks that are too complex, time-consuming, or overwhelming (*automate* & *facilitate*)

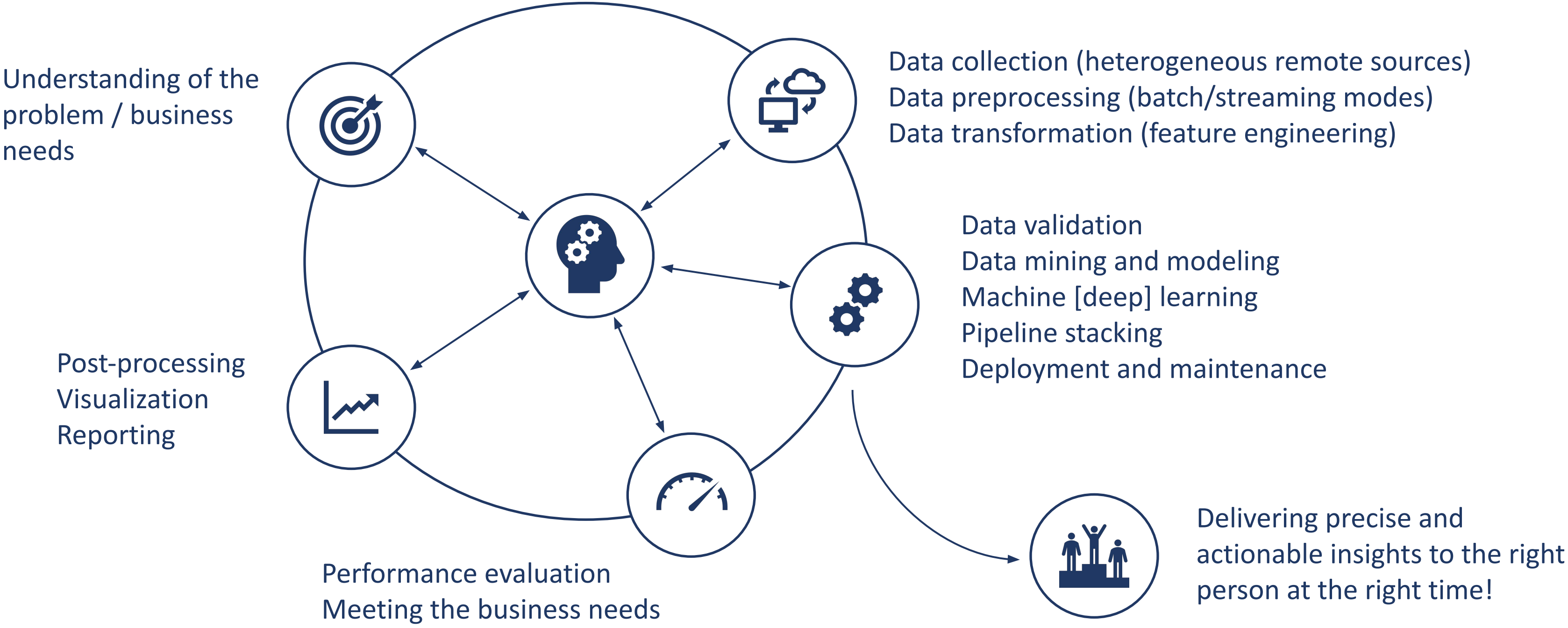
Examples

- **Automated Documentation of Data Science Experiments** - how to automatically detect and track relevant information and digital artifacts
- **Unsupervised Data Quality Validation** - how to automatically detect data quality issues without continuous manual inspection of data pipelines
- **Validating the predictions of Black-Box ML models** - how to demonstrate the effectiveness of Black-Box ML models on previously unseen production data
- **Assisted Design of Data Science Pipelines** - how to help novice-users or domain experts design efficient end-to-end DS pipelines

Selected Publications

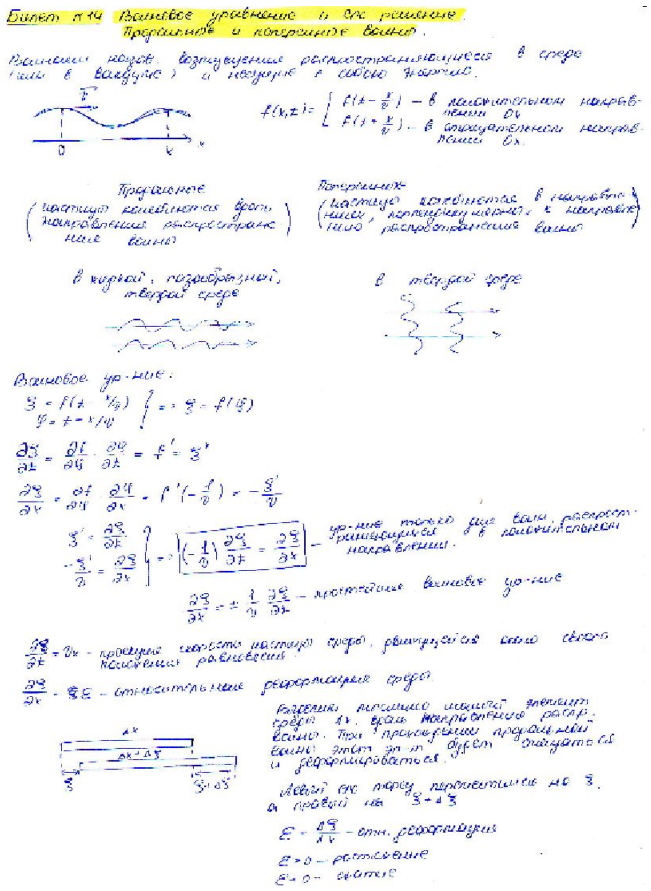
- Automated Documentation of End-to-End Experiments in Data Science, PhD Workshop, ICDE'19
- Learning to Validate the Predictions of Black Box Machine Learning Models on Unseen Data, HILDA, SIGMOD'19
- Towards Unsupervised Data Quality Validation on Dynamic Data, ETMLP, EDBT'20
- Automating Data Quality Validation for Dynamic Data Ingestion, EDBT'21
- DORIAN in Action: Assisted Design of Data Science Pipelines, VLDB'22

Data Science Processes



A Lab Notebook is … *

- Complete **record of procedures**, reagents, **data**, and **thoughts** to pass on to other researchers
- **Explanation** of why experiments were initiated, how they were performed, and the results
- **Main source for reproducibility of experiments**
- **Legal document** to prove patents and defend your data against accusations of fraud
- Scientific **legacy** in the lab

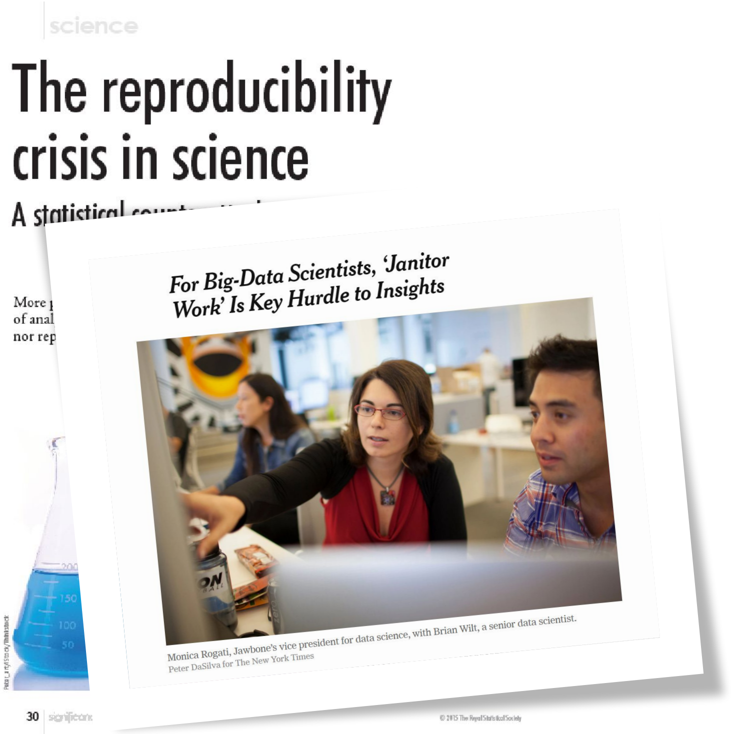


** Keeping a Lab Notebook, NIH, Office of Intramural Training and Education [https://www.training.nih.gov/assets/Lab_Notebook_508_(new).pdf]*

Automated Documentation of Data Science Experiments

80% of workload – solving technical problems

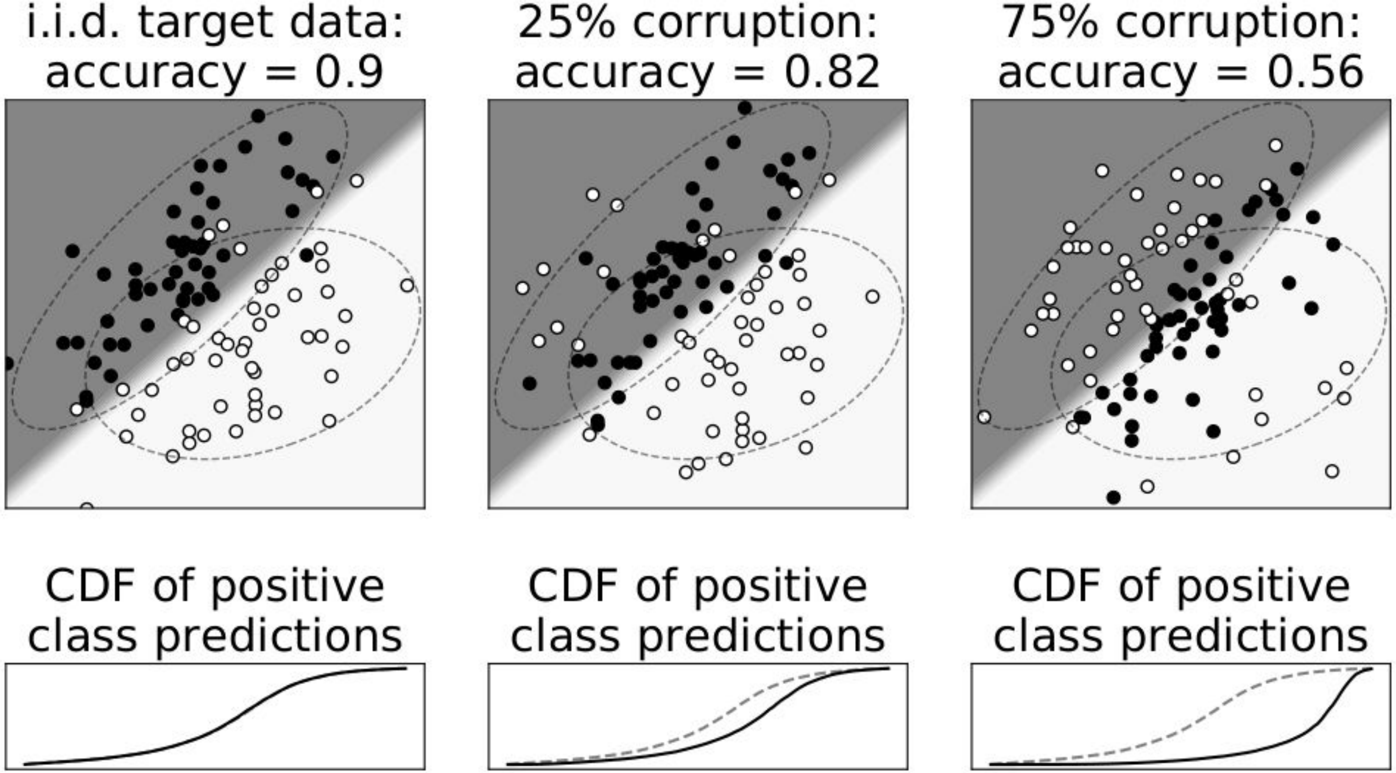
- Abundance of tools and frameworks – “glue” code, “smells”
- Multi-tenant environment
- Lack of systematic holistic approaches
- Try-Fail-Learn-Iterate paradigm



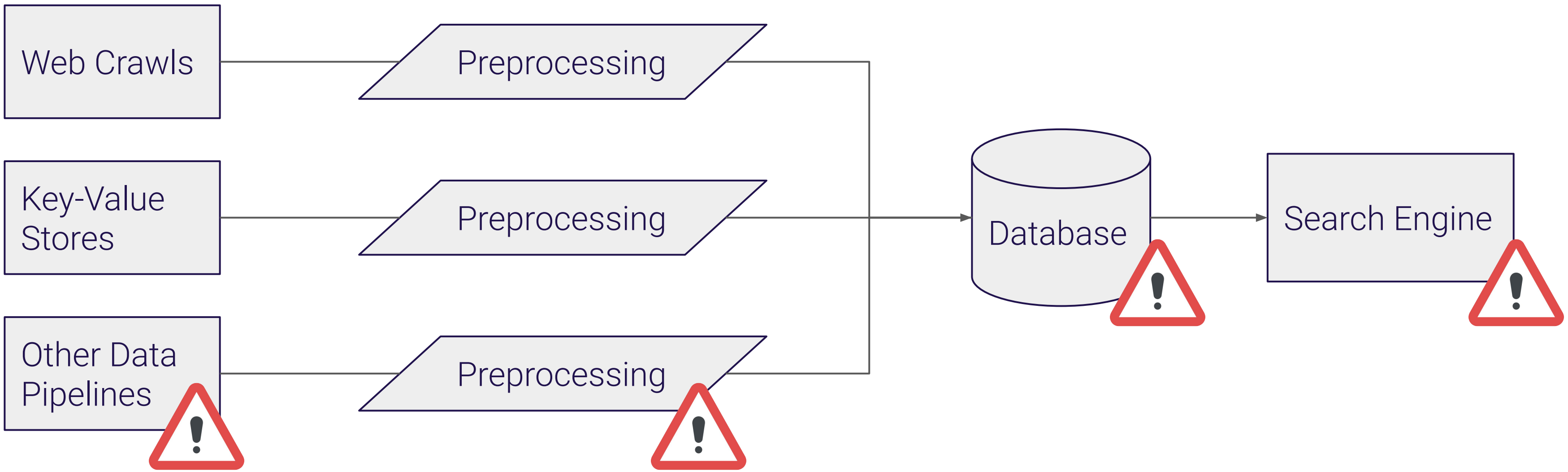
Reproducibility as the core of the scientific method is at stake

https://rss.onlinelibrary.wiley.com/doi/pdf/10.1111/j.1740-9713.2015.00827.x
https://www.nytimes.com/2014/08/18/technology/for-big-data-scientists-hurdle-to-insights-is-janitor-work.html

Validating the predictions of Black-Box ML models



Scenario I. Retail Company



Overall Challenges

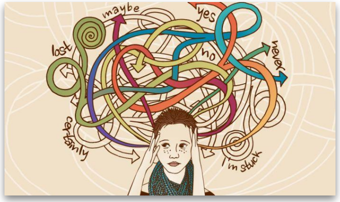
Assistance required

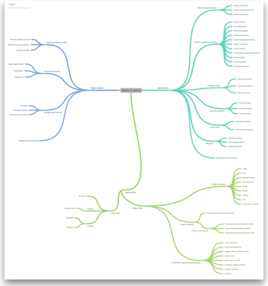
- ➔ to bring down costs
- ➔ to increase scalability of data ingestion, experimentation, pipeline design, etc.
- ➔ to reduce the time domain experts and engineers have to spend on fixing DQ issues, validating the models, aka ‘Janitor Work’
- ➔ to adapt for non-expert users

Automation is not always possible

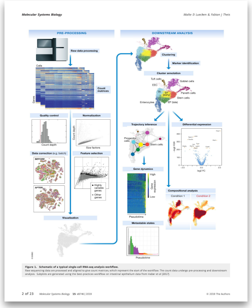
- ➔ user input needed

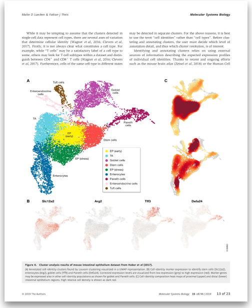
Assisted Design of DS Pipelines





- Design of DS pipelines might be **overwhelming** for **domain experts** and novice users
- Even for ML experts, **hard to keep up** with new development
- Assisting tools are **bound** to a particular context
  - Supported DS tasks
  - Supported DS operators
  - Supported evaluation processes
- What if… application domains **surpass** this context





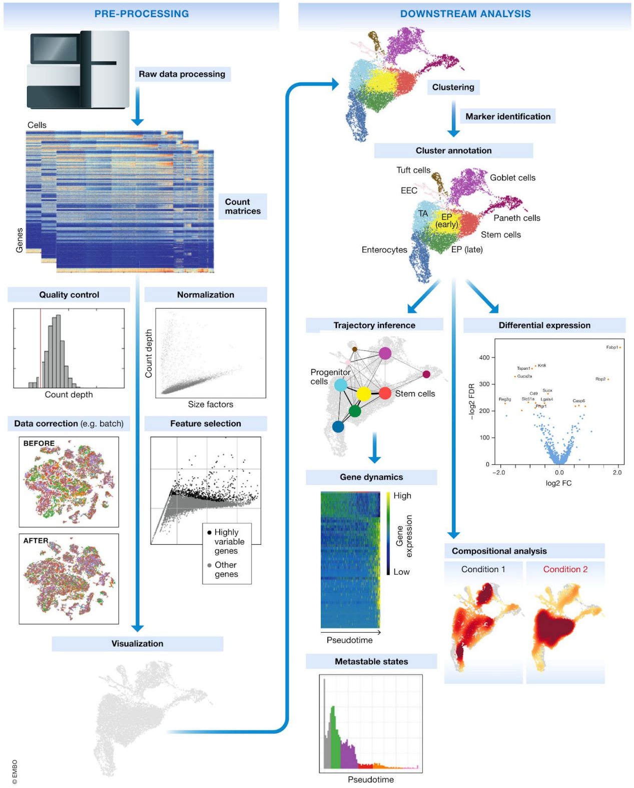


Figure 1. Schematic of a typical single-cell RNA-seq analysis workflow.
Raw sequencing data are processed and aligned to give count matrices, which represent the start of the workflow. The count data undergo pre-processing and downstream analysis. Subplots are generated using the best-practices workflow on intestinal epithelium data from Haber *et al* (2017).

While it may be tempting to assume that the clusters detected in single-cell data represent cell types, there are several axes of variation that determine cellular identity (Wagner *et al*, 2016; Clevers *et al*, 2017). Firstly, it is not always clear what constitutes a cell type. For example, while "T cells" may be a satisfactory label of a cell type to some, others may look for T-cell subtypes within a dataset and distinguish between CD4$^+$ and CD8$^+$ T cells (Wagner *et al*, 2016; Clevers *et al*, 2017). Furthermore, cells of the same cell type in different states may be detected in separate clusters. For the above reasons, it is best to use the term "cell identities" rather than "cell types". Before clustering and annotating clusters, the user must decide which level of annotation detail, and thus which cluster resolution, is of interest.

Identifying and annotating clusters relies on using external sources of information describing the expected expression profiles of individual cell identities. Thanks to recent and ongoing efforts such as the mouse brain atlas (Zeisel *et al*, 2018) or the Human Cell

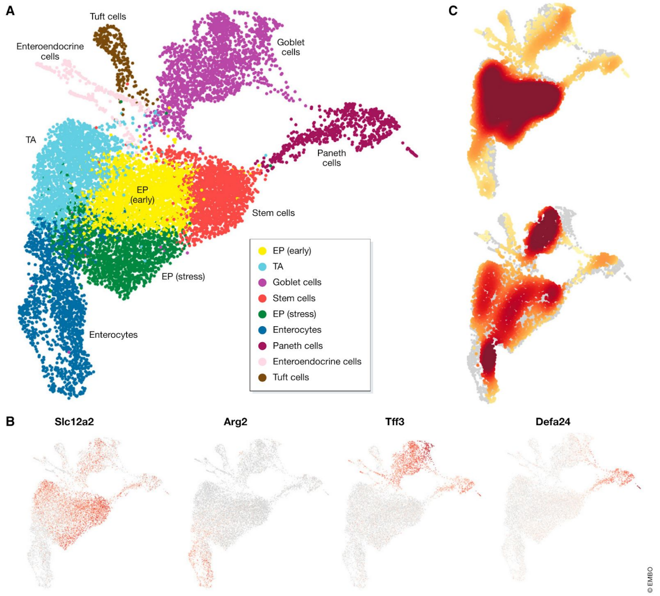


Figure 6. Cluster analysis results of mouse intestinal epithelium dataset from Haber *et al* (2017).
(A) Annotated cell-identity clusters found by Louvain clustering visualized in a UMAP representation. (B) Cell-identity marker expression to identify stem cells (Slc12a2), enterocytes (Arg2), goblet cells (Tff3) and Paneth cells (Defa24). Corrected expression levels are visualized from low expression (grey) to high expression (red). Marker genes may be expressed also in other cell-identity populations as shown for goblet and Paneth cells. (C) Cell-identity composition heat maps of proximal (upper) and distal (lower) intestinal epithelium regions. High relative cell density is shown as dark red.

Auto-Sklearn 2.0: The Next Generation

Matthias Feurer, Katharina Eggensperger, Stefan Falkner, Marius Lindauer and Frank Hutter

arXiv:2007.04074v1 [cs.LG] 8 Jul 2020

JMLR: Workshop and Conference Proceedings 64:66–74, 2016 ICML 2016 AutoML Workshop

TPOT: A Tree-based Pipeline Optimization Tool for Automating Machine Learning

Randal S. Olson olsonran@upenn.edu and Jason H. Moore jhmoore@upenn.edu

Journal of Machine Learning Research 18 (2017) 1-5 Submitted 5/16; Revised 11/16; Published 3/17

Auto-WEKA 2.0: Automatic model selection and hyperparameter optimization in WEKA

Lars Kotthoff larsko@cs.ubc.ca
Chris Thornton cwthornt@cs.ubc.ca
Holger H. Hoos hoos@cs.ubc.ca
Frank Hutter fh@cs.uni-freiburg.de
Kevin Leyton-Brown kevinlb@cs.ubc.ca
Department of Computer Science
University of British Columbia
2366 Main Mall, Vancouver, B.C. V6T 1Z4 Canada

Editor: Geoff Holmes

Abstract

WEKA is a widely used, open-source machine learning platform. Due to its intuitive interface, it is particularly popular with novice users. However, such users often find it hard to identify the best approach for their particular dataset among the many available. We describe the new version of *Auto-WEKA*, a system designed to help such users by automatically searching through the joint space of WEKA's learning algorithms and their respective hyperparameter settings to maximize performance, using a state-of-the-art Bayesian optimization method. Our new package is tightly integrated with WEKA, making it just as accessible to end users as any other learning algorithm.

Keywords: Hyperparameter Optimization, Model Selection, Feature Selection

1. The Principles Behind Auto-WEKA

The WEKA machine learning software (Hall et al., 2009) puts state-of-the-art machine learning techniques at the disposal of even novice users. However, such users do not typically know how to choose among the dozens of machine learning procedures implemented in WEKA and each procedure's hyperparameter settings to achieve good performance.

Auto-WEKA[1] addresses this problem by treating all of WEKA as a single, highly parametric machine learning framework, and using Bayesian optimization to find a strong instantiation for a given dataset. Specifically, it considers the combined space of WEKA's learning algorithms $\mathcal{A} = \{A^{(1)}, \ldots, A^{(k)}\}$ and their associated hyperparameter spaces $\Lambda^{(1)}, \ldots, \Lambda^{(k)}$ and aims to identify the combination of algorithm $A^{(j)} \in \mathcal{A}$ and hyperparameters $\lambda \in \Lambda^{(j)}$ that minimizes cross-validation loss,

$$A^*_{\lambda^*} \in \underset{A^{(j)} \in \mathcal{A}, \lambda \in \Lambda^{(j)}}{\operatorname{argmin}} \frac{1}{k} \sum_{i=1}^{k} \mathcal{L}\left(A^{(j)}_{\lambda}, \mathcal{D}^{(i)}_{\text{train}}, \mathcal{D}^{(i)}_{\text{test}}\right),$$

1. Thornton et al. (2013) first introduced Auto-WEKA and empirically demonstrated state-of-the-art performance. Here we describe an improved and more broadly accessible implementation of Auto-WEKA, focussing on usability and software design.

Designing KDD-Workflows via HTN-Planning for Intelligent Discovery Assistance

The Data Mining Advisor: Meta-learning at the Service of Practitioners

Machine Learning (2018) 107:1495–1515
https://doi.org/10.1007/s10994-018-5735-z

ML-Plan: Automated machine learning via hierarchical planning

Felix Mohr[1] · Marcel Wever[1] · Eyke Hüllermeier[1]

Received: 10 December 2017 / Accepted: 18 June 2018 / Published online: 3 July 2018

Abstract

Automated machine learning (AutoML) seeks to automatically select, compose, and parametrize machine learning algorithms, so as to achieve optimal performance on a given task (dataset). Although current approaches to AutoML have already produced impressive results, the field is still far from mature, and new techniques are still being developed. In this paper, we present ML-Plan, a new approach to AutoML based on hierarchical planning. To highlight the potential of this approach, we compare ML-Plan to the state-of-the-art frameworks Auto-WEKA, auto-sklearn, and TPOT. In an extensive series of experiments, we show that ML-Plan is highly competitive and often outperforms existing approaches.

Keywords Automated machine learning · Automated planning · Algorithm selection · Algorithm configuration · Heuristic search

1 Introduction

The demand for machine learning (ML) functionality is growing quite rapidly, and successful machine learning applications can be found in more and more sectors of science, technology, and society. Since end users in application domains are normally not machine learning experts, there is an urgent need for suitable support in terms of tools that are easy to use. Ideally, the induction of models from data, including the data preprocessing, the choice of a model class, the training and evaluation of a predictor, the representation and interpretation of results, etc., would be automated to a large extent (Lloyd et al. 2014). This has triggered the field of *automated machine learning* (AutoML).

Editors: Jesse Davis, Elisa Fromont, Derek Greene, and Bjorn Bringmaan..

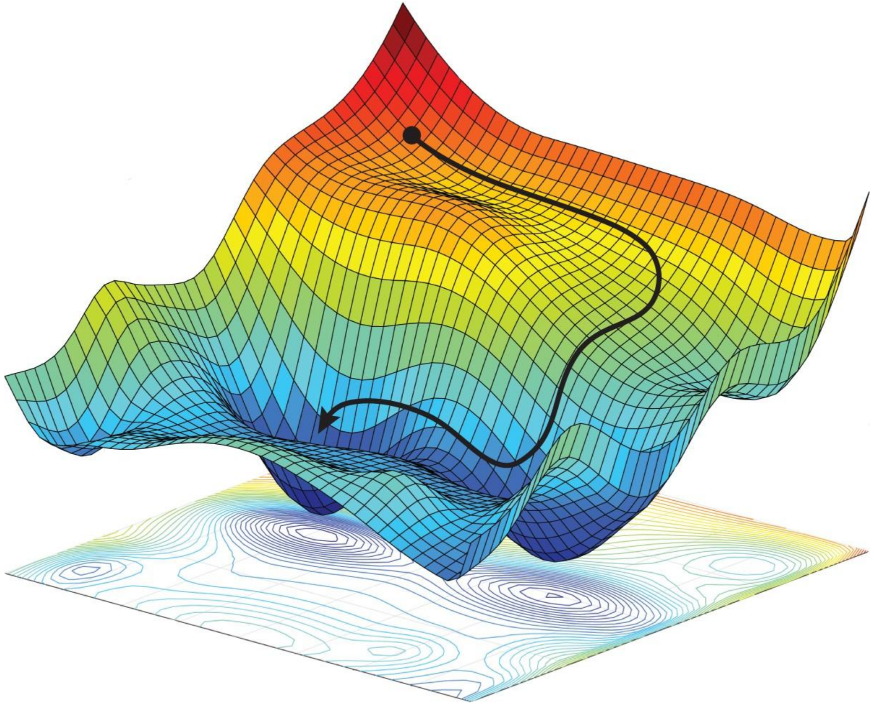
✉ Eyke Hüllermeier
eyke@uni-paderborn.de

Felix Mohr
felix.mohr@uni-paderborn.de

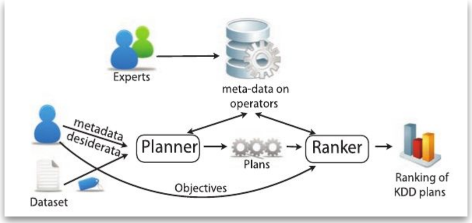
Marcel Wever
marcel.wever@uni-paderborn.de

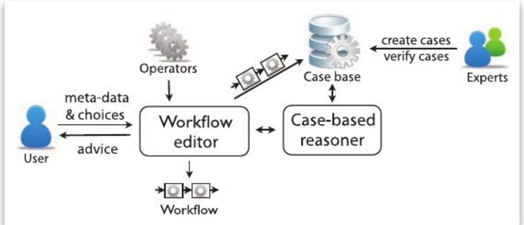
1 Paderborn University, Warburger Str. 100, 33098 Paderborn, Germany

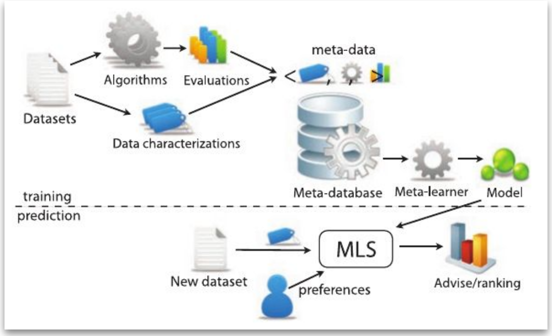
Springer



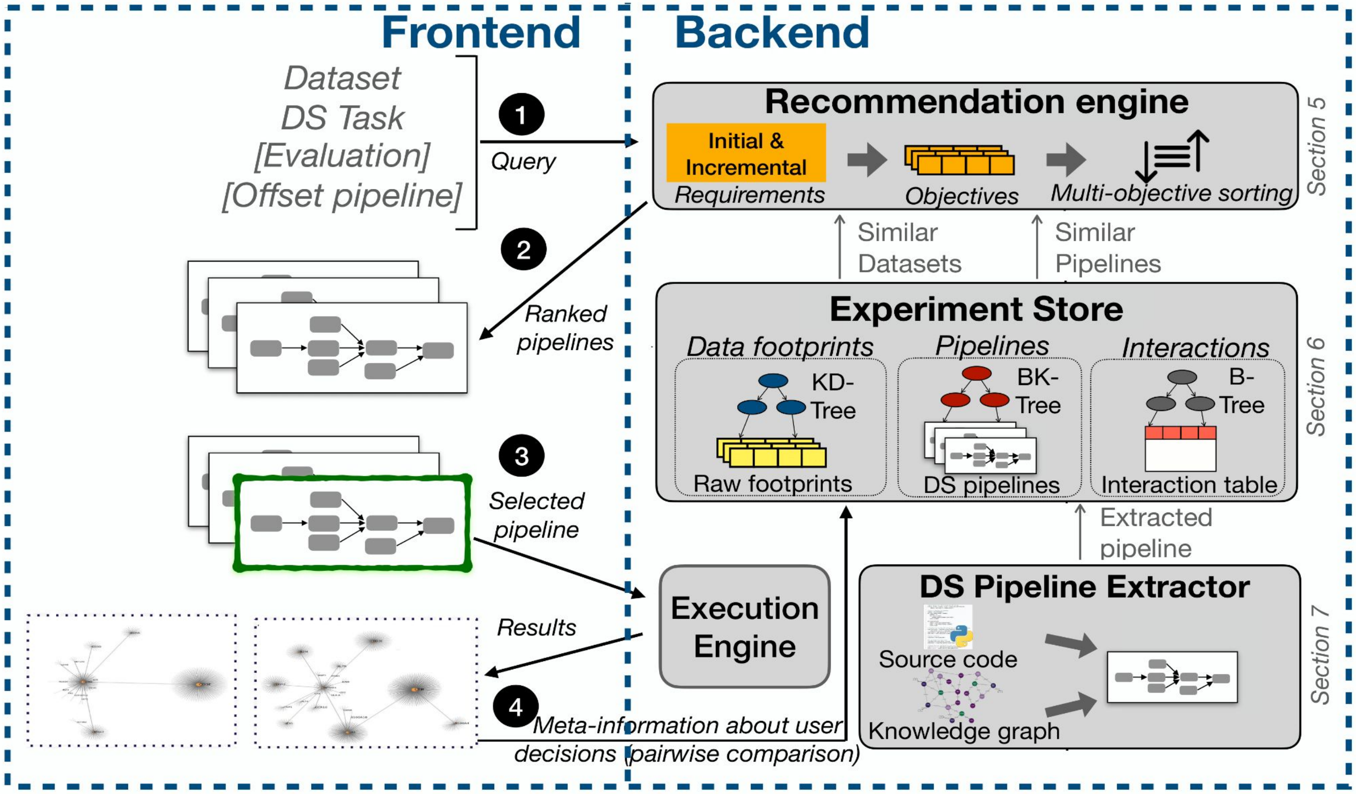

Q&A
Expert System
Expert rules
Experts
List or ranking of useful techniques


Experts
meta-data on operators
metadata
desiderata
Planner
Plans
Ranker
Ranking of KDD plans
Dataset
Objectives


Operators
Case base
create cases
verify cases
Experts
meta-data & choices
advice
User
Workflow editor
Case-based reasoner
Workflow

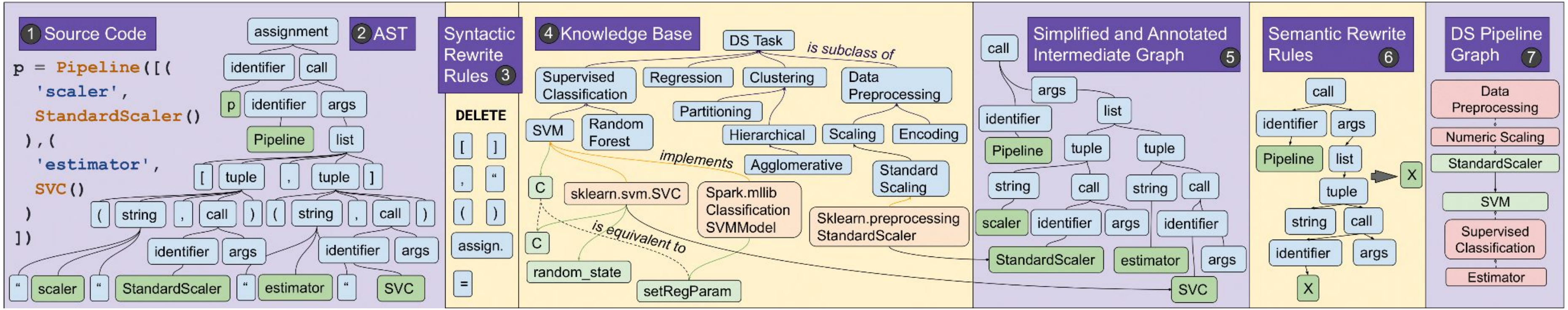

Datasets
Algorithms
Evaluations
meta-data
Data characterizations
Meta-database
Meta-learner
Model
training
prediction
New dataset
MLS
preferences
Advise/ranking

Approach

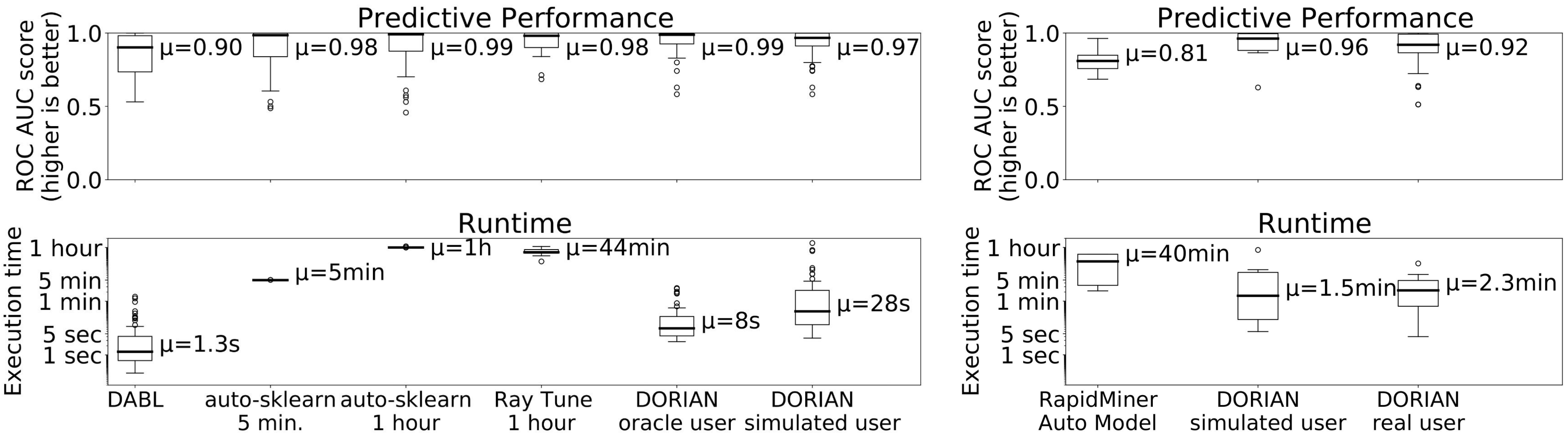


Frontend
Backend
Dataset
DS Task
[Evaluation]
[Offset pipeline]
1
Query
Recommendation engine
Initial &
Incremental
Requirements
Objectives
Multi-objective sorting
Section 5
2
Ranked
pipelines
Similar
Datasets
Similar
Pipelines
Experiment Store
Data footprints
Pipelines
Interactions
KD-
Tree
BK-
Tree
B-
Tree
Raw footprints
DS pipelines
Interaction table
Section 6
3
Selected
pipeline
Extracted
pipeline
Execution
Engine
DS Pipeline Extractor
Source code
Knowledge graph
Section 7
Results
4
Meta-information about user
decisions (pairwise comparison)

DS Pipeline Extractor



Evaluation, Classification



DORIAN in action: Assisted Design of Data Science Pipelines

- Design of DS pipelines might be overwhelming for domain experts and novice users
- Assisting tools yield limited applicability in a wide range of application domains
- DORIAN is a human-in-the-loop approach for the assisted design of DS pipelines that supports a large and growing set of DS tasks, operators, and arbitrary user-defined evaluation procedures.

Sergey Redyuk (sergey.redyuk@tu-berlin.de), Zoi Kaoudi, Sebastian Schelter, Volker Markl

References

[1] F. Serban, J. Vanschoren, J.-U. Kietz, and A. Bernstein. A survey of intelligent assistants for data analysis. ACM Computing Surveys (CSUR), 45(3):1–35, 2013.

[2] Z. Shang, E. Zgraggen, B. Buratti, F. Kossmann, P. Eichmann, Y. Chung, C. Binnig, E. Upfal, and T. Kraska. Democratizing data science through interactive curation of ml pipelines. In Proceedings of the 2019 International Conference on Management of Data, pages 1171–1188, 2019.

[3] S. Redyuk, V. Markl, and S. Schelter. Towards unsupervised data quality validation on dynamic data. In Proceedings of the Workshops of the EDBT/ICDT 2020Joint Conference, Copenhagen, Denmark, March 30, 2020, volume 2578 of CEUR Workshop Proceedings. CEUR-WS.org,2020.

[4] R. S. Olson and J. H. Moore. Tpot: A tree-based pipeline optimization tool for automating machine learning. In Workshop on automatic machine learning, pages 66–74. PMLR, 2016.

[5] M. Feurer, A. Klein, K. Eggensperger, J. T. Springenberg, M. Blum, and F. Hutter. Auto-sklearn: efficient and robust automated machine learning. In Automated Machine Learning, pages 113–134. Springer, Cham, 2019.

[6] S. Redyuk, Z. Kaoudi, V. Markl, S. Schelter (2021) Automating Data Quality Validation for Dynamic Data Ingestion. EDBT'21, Nicosia, Cyprus

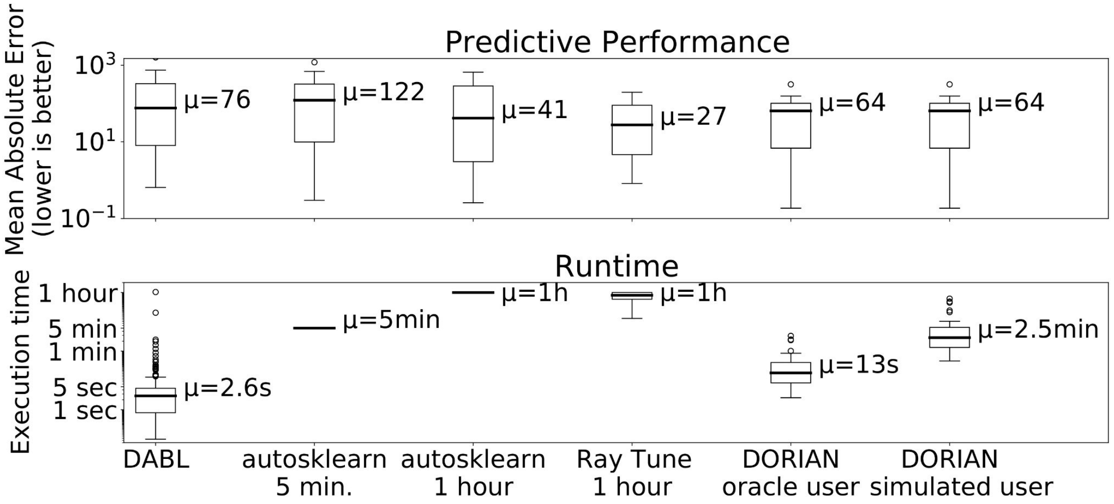
[7] Wilson, G., Bryan, J., Cranston, K., Kitzes, J., Nederbragt, L. and Teal, T.K., 2016. Good Enough Practices in Scientific Computing. PubMed

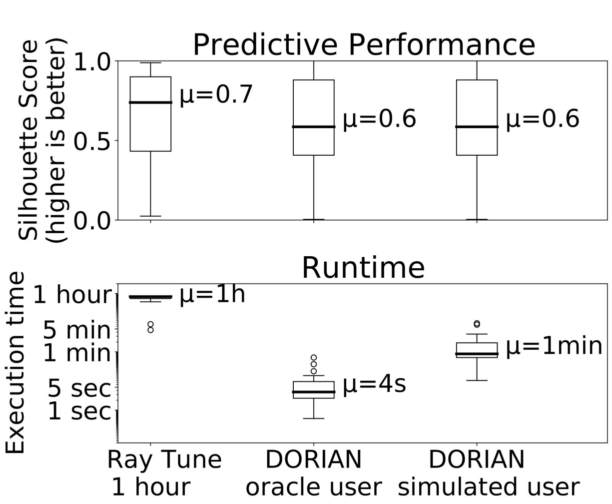
[8] Polyzotis, N., Roy, S., Whang, S.E. and Zinkevich, M., 2018. Data lifecycle challenges in production machine learning: A survey. ACM SIGMOD Record, 47(2), pp.17-28.

[9] Schelter, S., Biessmann, F., Januschowski, T., Salinas, D., Seufert, S., Szarvas, G., Vartak, M., Madden, S., Miao, H., Deshpande, A. and Zaharia, M., 2018. On Challenges in Machine Learning Model Management. Data Engineering, p.5.

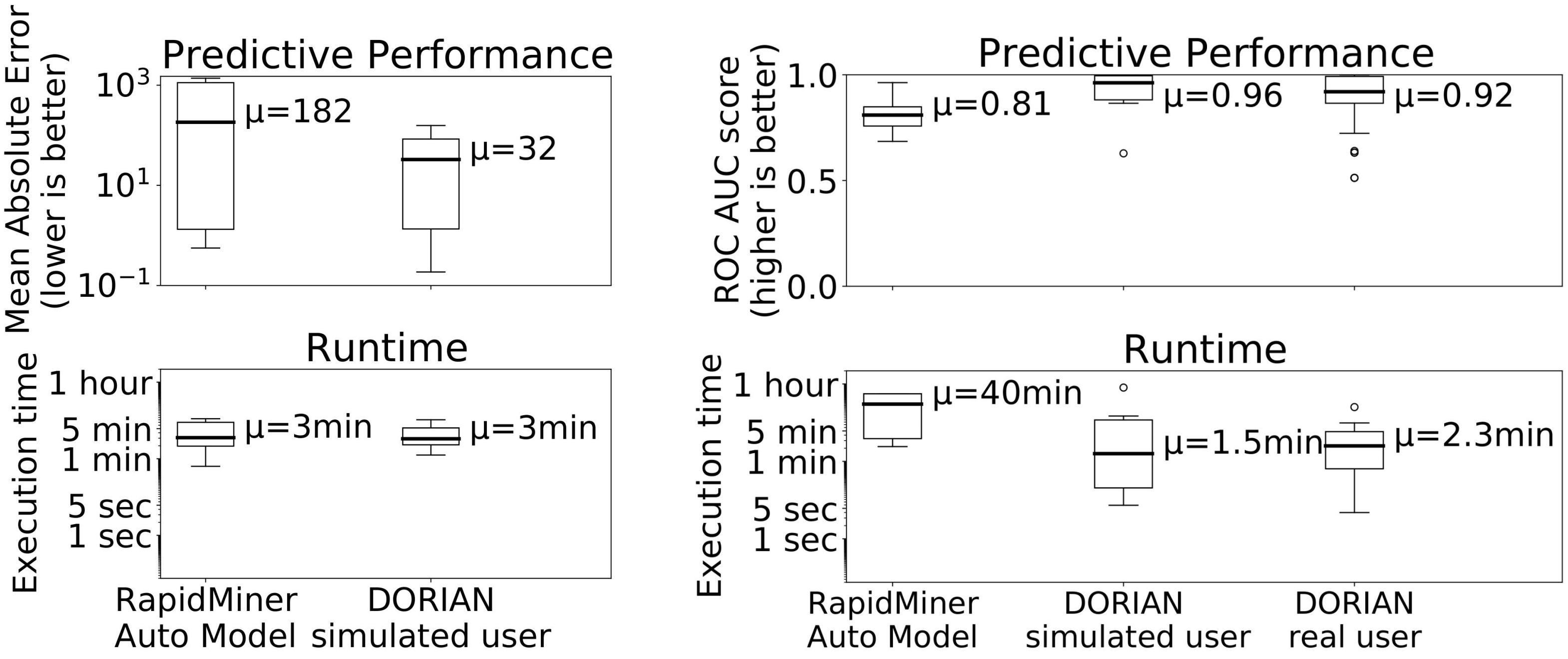
[10] Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., Chaudhary, V., Young, M., Crespo, J.F. and Dennison, D., 2015. Hidden technical debt in machine learning systems. In Advances in NIPS (pp. 2503-2511)

Evaluation, Regression & Clustering

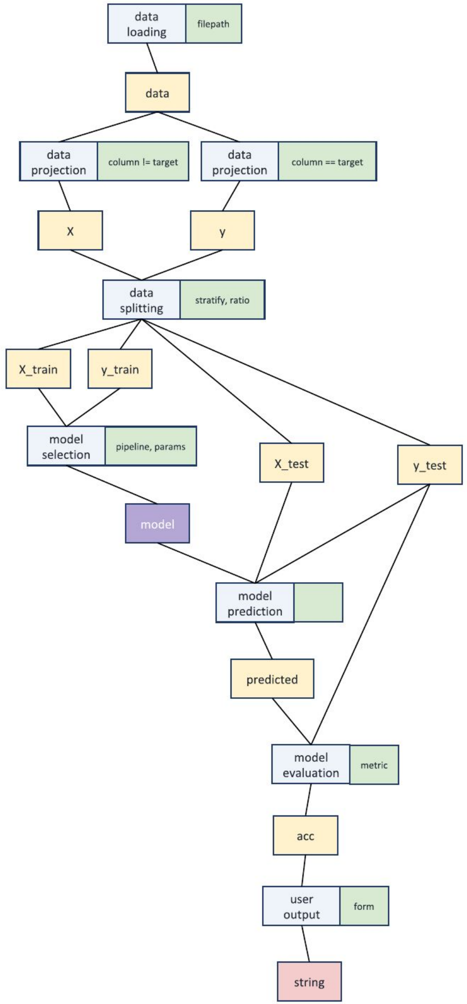




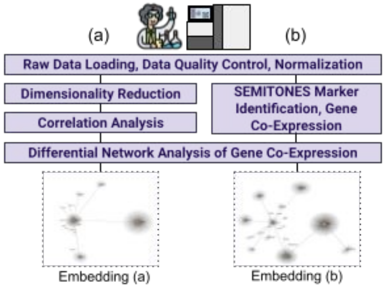
Evaluation, Regression & Clustering (manual)



```python
from sklearn.model_selection import train_test_split, GridSearchCV
...
import numpy as np

data_filepath, target = '…', 'class'
data = pd.read_csv(data_filepath)
columns = list(data)

X, y = data[[col for col in columns if col != target]], data[target]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.1)

cat_cols = ['workclass', 'occupation', 'marital_status']
num_cols = ['hours_per_week', 'age']

feature_transformation = ColumnTransformer(transformers=[
    ('cat_features', OneHotEncoder(handle_unknown='ignore'), cat_cols),
    ('scaled_numeric', StandardScaler(), num_cols)
])

pipeline = Pipeline([
    ('features', feature_transformation),
    ('learner', SGDClassifier(max_iter=1000, tol=1e-3))
])

param_grid = {
    'learner__alpha': [0.0001, 0.001, 0.01, 0.1]
}

search = GridSearchCV(pipeline, param_grid, cv=5)
model = search.fit(X_train, y_train)

predicted = model.predict(X_test)
acc = accuracy_score(y_test, predicted)
print("TRAIN. accuracy: %.4f" % (acc))
```



Design Decisions



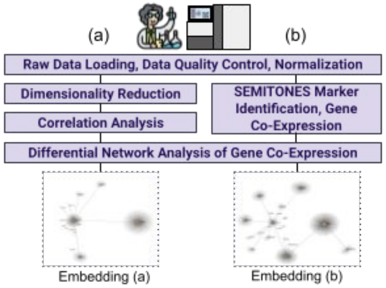
Pipeline as DAG[Operation, ConnectionPoint]

Operation as f[List[Input], List[Output]], semantically enriched black box

Hyperparameter as special Input

DS ontology to decouple Operation’s intent from implementation

Design Decisions [cont.]



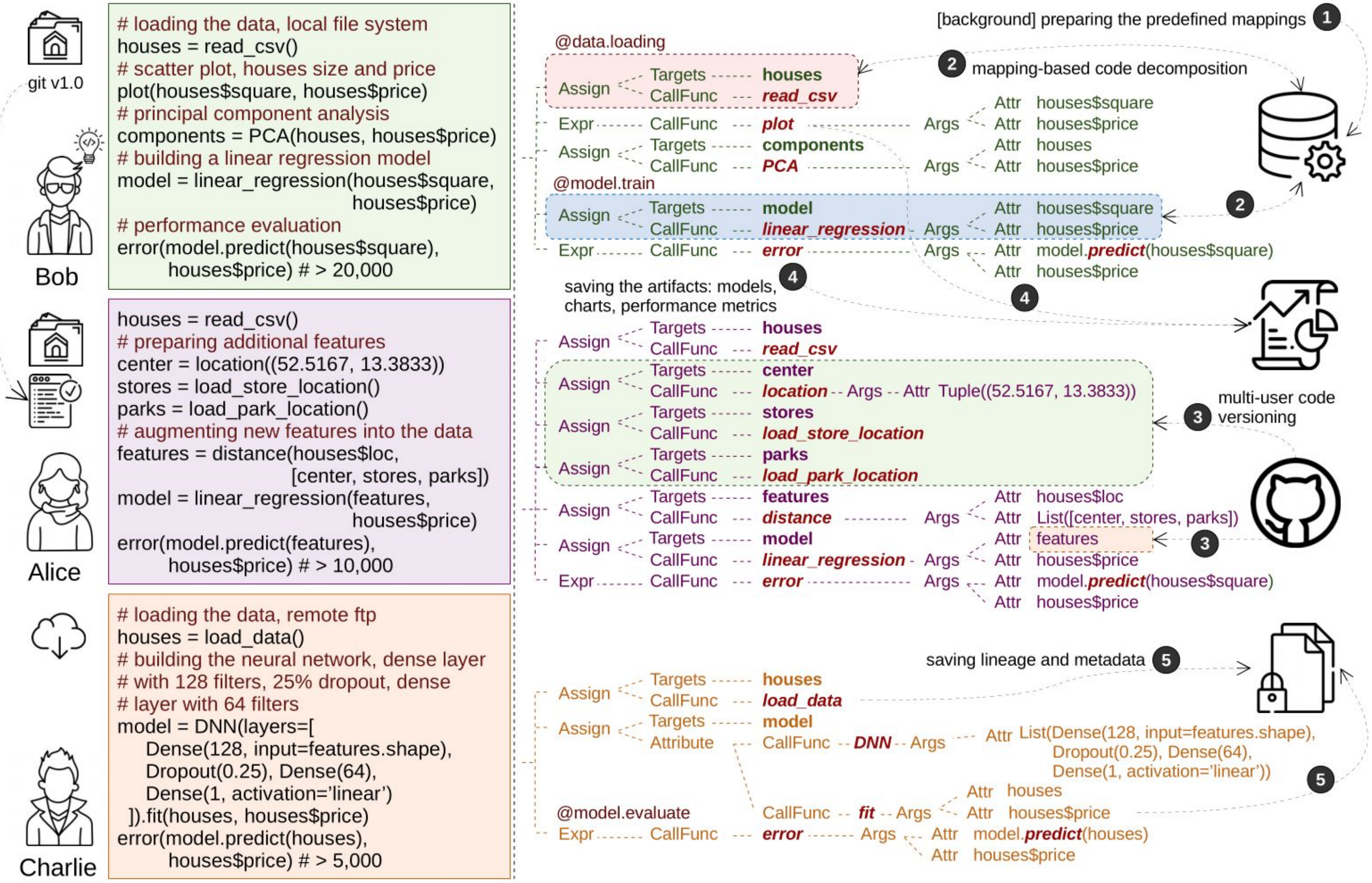
Evaluation Process supports arbitrary use cases

(pairwise comparison of candidates)

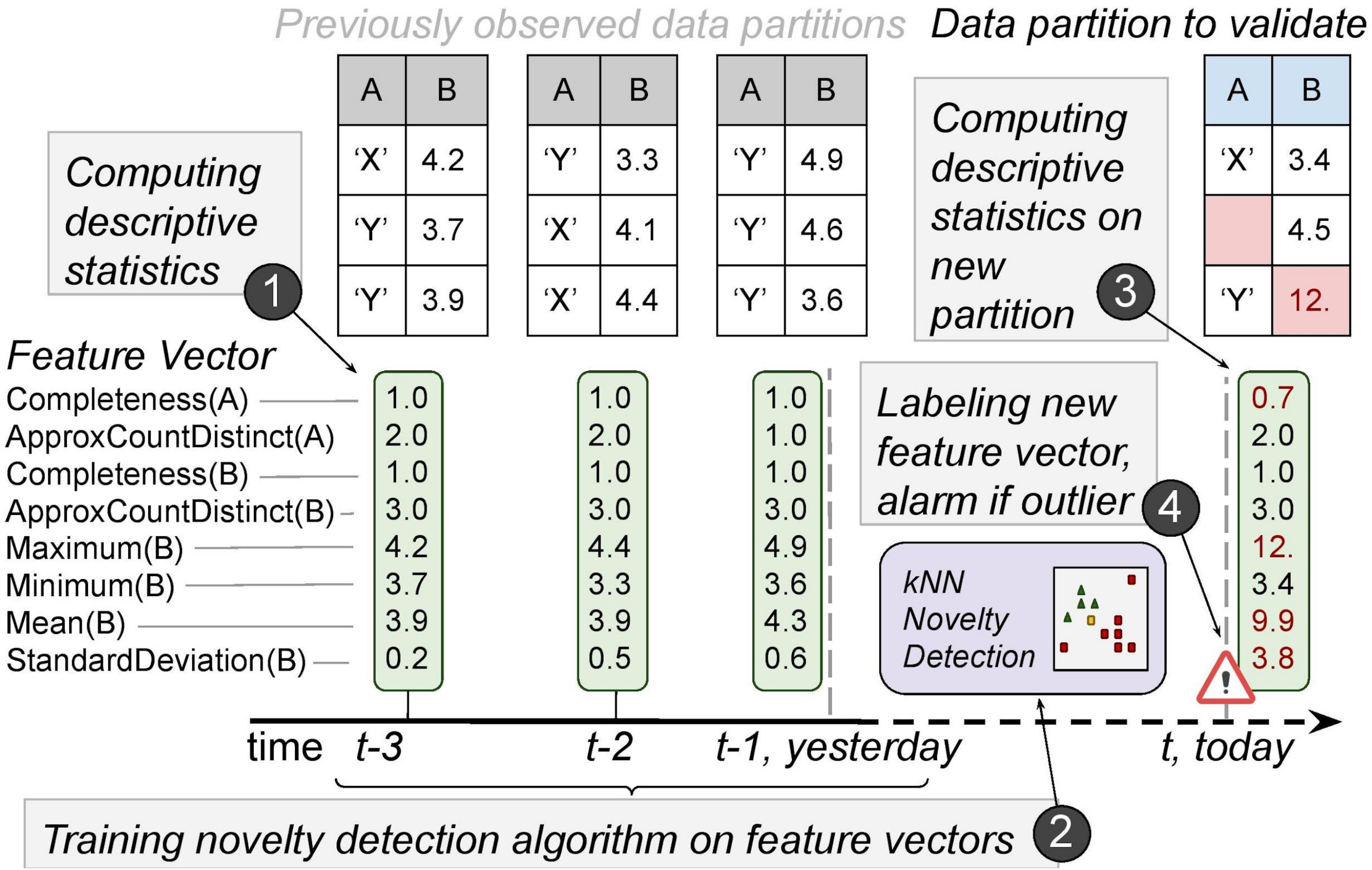
Interaction Table records user actions and preferences

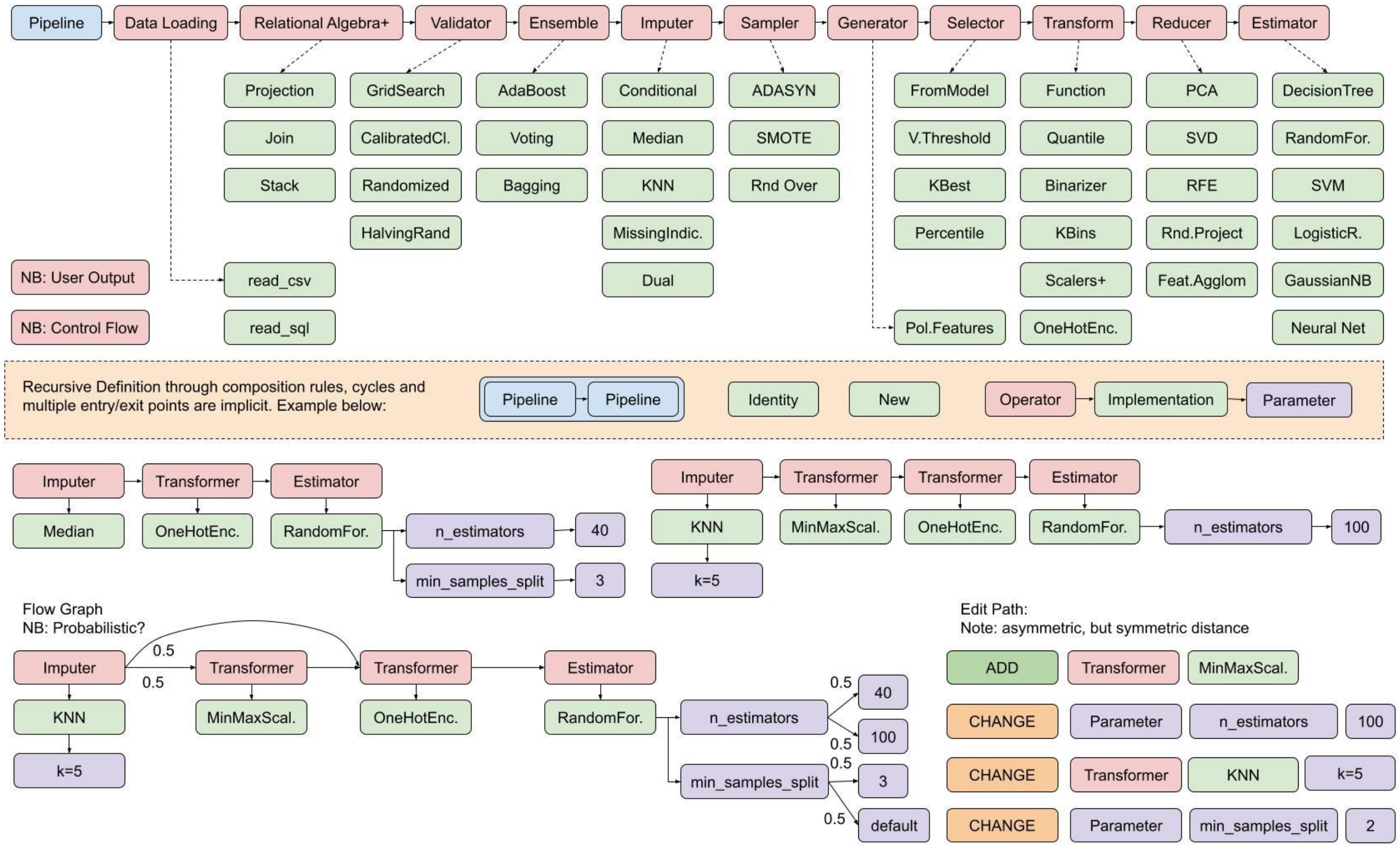
Ranking Objectives are extendable and take into account two scenarios: where the end-user does or does not specify the offset pipeline (aka Initial recommendations vs Incremental improvements)

Ranking considered a non-dominated sorting problem

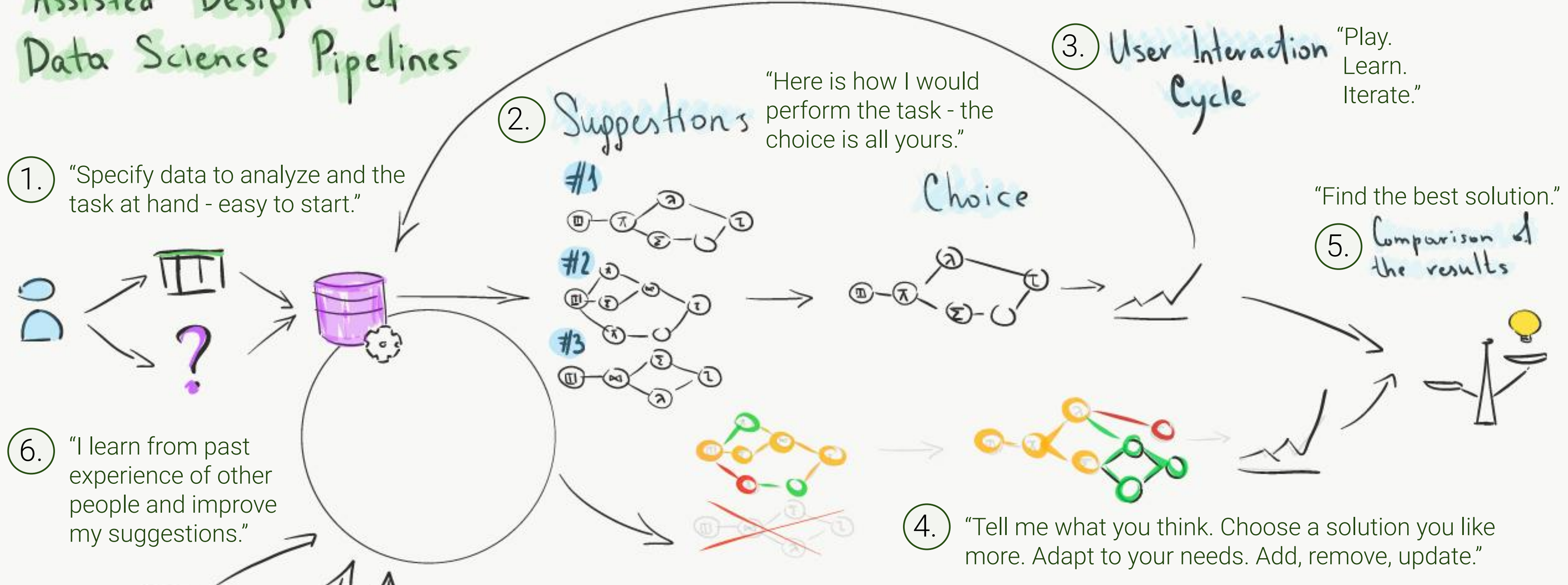


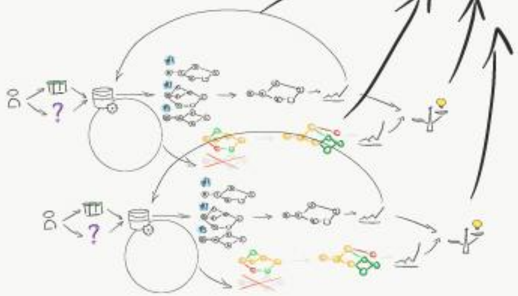
git v1.0
Bob
Alice
Charlie
# loading the data, local file system
houses = read_csv()
# scatter plot, houses size and price
plot(houses$square, houses$price)
# principal component analysis
components = PCA(houses, houses$price)
# building a linear regression model
model = linear_regression(houses$square, houses$price)
# performance evaluation
error(model.predict(houses$square), houses$price) # > 20,000
houses = read_csv()
# preparing additional features
center = location((52.5167, 13.3833))
stores = load_store_location()
parks = load_park_location()
# augmenting new features into the data
features = distance(houses$loc, [center, stores, parks])
model = linear_regression(features, houses$price)
error(model.predict(features), houses$price) # > 10,000
# loading the data, remote ftp
houses = load_data()
# building the neural network, dense layer
# with 128 filters, 25% dropout, dense
# layer with 64 filters
model = DNN(layers=[
Dense(128, input=features.shape),
Dropout(0.25), Dense(64),
Dense(1, activation='linear')
]).fit(houses, houses$price)
error(model.predict(houses), houses$price) # > 5,000
[background] preparing the predefined mappings 1
2 mapping-based code decomposition
@data.loading
Assign Targets houses
CallFunc read_csv
Expr CallFunc plot Args Attr houses$square Attr houses$price
Assign Targets components
CallFunc PCA Args Attr houses Attr houses$price
@model.train
Assign Targets model
CallFunc linear_regression Args Attr houses$square Attr houses$price
Expr CallFunc error Args Attr model.predict(houses$square) Attr houses$price
saving the artifacts: models, charts, performance metrics 4
Assign Targets houses CallFunc read_csv
Assign Targets center CallFunc location Args Attr Tuple((52.5167, 13.3833))
Assign Targets stores CallFunc load_store_location
Assign Targets parks CallFunc load_park_location
multi-user code versioning 3
Assign Targets features CallFunc distance Args Attr houses$loc Attr List([center, stores, parks])
Assign Targets model CallFunc linear_regression Args Attr features Attr houses$price
Expr CallFunc error Args Attr model.predict(houses$square) Attr houses$price
saving lineage and metadata 5
Assign Targets houses CallFunc load_data
Assign Targets model Attribute CallFunc DNN Args Attr List(Dense(128, input=features.shape), Dropout(0.25), Dense(64), Dense(1, activation='linear'))
CallFunc fit Args Attr houses Attr houses$price
@model.evaluate
Expr CallFunc error Args Attr model.predict(houses) Attr houses$price


Previously observed data partitions
Data partition to validate
Computing descriptive statistics
1
A B
'X' 4.2
'Y' 3.7
'Y' 3.9
A B
'Y' 3.3
'X' 4.1
'X' 4.4
A B
'Y' 4.9
'Y' 4.6
'Y' 3.6
Computing descriptive statistics on new partition
3
A B
'X' 3.4
4.5
'Y' 12.
Feature Vector
Completeness(A)
ApproxCountDistinct(A)
Completeness(B)
ApproxCountDistinct(B)
Maximum(B)
Minimum(B)
Mean(B)
StandardDeviation(B)
1.0 2.0 1.0 3.0 4.2 3.7 3.9 0.2
1.0 2.0 1.0 3.0 4.4 3.3 3.9 0.5
1.0 1.0 1.0 3.0 4.9 3.6 4.3 0.6
Labeling new feature vector, alarm if outlier
4
kNN Novelty Detection
0.7 2.0 1.0 3.0 12. 3.4 9.9 3.8
time t-3 t-2 t-1, yesterday t, today
Training novelty detection algorithm on feature vectors
2


Pipeline
Data Loading
Relational Algebra+
Validator
Ensemble
Imputer
Sampler
Generator
Selector
Transform
Reducer
Estimator
Projection
Join
Stack
GridSearch
CalibratedCl.
Randomized
HalvingRand
AdaBoost
Voting
Bagging
Conditional
Median
KNN
MissingIndic.
Dual
ADASYN
SMOTE
Rnd Over
FromModel
V.Threshold
KBest
Percentile
Pol.Features
Function
Quantile
Binarizer
KBins
Scalers+
OneHotEnc.
PCA
SVD
RFE
Rnd.Project
Feat.Agglom
DecisionTree
RandomFor.
SVM
LogisticR.
GaussianNB
Neural Net
NB: User Output
NB: Control Flow
read_csv
read_sql
Recursive Definition through composition rules, cycles and multiple entry/exit points are implicit. Example below:
Pipeline
Pipeline
Identity
New
Operator
Implementation
Parameter
Imputer
Transformer
Estimator
Median
OneHotEnc.
RandomFor.
n_estimators
40
min_samples_split
3
Imputer
Transformer
Transformer
Estimator
KNN
MinMaxScal.
OneHotEnc.
RandomFor.
n_estimators
100
k=5
Flow Graph
NB: Probabilistic?
Imputer
0.5
0.5
Transformer
Transformer
Estimator
KNN
MinMaxScal.
OneHotEnc.
RandomFor.
k=5
n_estimators
0.5
40
0.5
100
min_samples_split
0.5
3
0.5
default
Edit Path:
Note: asymmetric, but symmetric distance
ADD
Transformer
MinMaxScal.
CHANGE
Parameter
n_estimators
100
CHANGE
Transformer
KNN
k=5
CHANGE
Parameter
min_samples_split
2

Assisted Design of Data Science Pipelines





Data Science toolbox became popular among natural and social scientists who tend to get overwhelmed due to a large variety of available methods and parameters to choose from. This project aims to *assist* with an interactive environment where a user specifies the task at hand and receives *guidance for designing powerful data analytics* based on curated past experience of others.

reds