

JuliaCon 2024

End-to-End AI (E2EAI) with Julia, KOs, and Argo Workflow

Presentor: Paulito Palmes, IBM Research

Collaborators: SUNRISE-6G EU Partners

Date: July 11, 2024



Funded by the
European Union



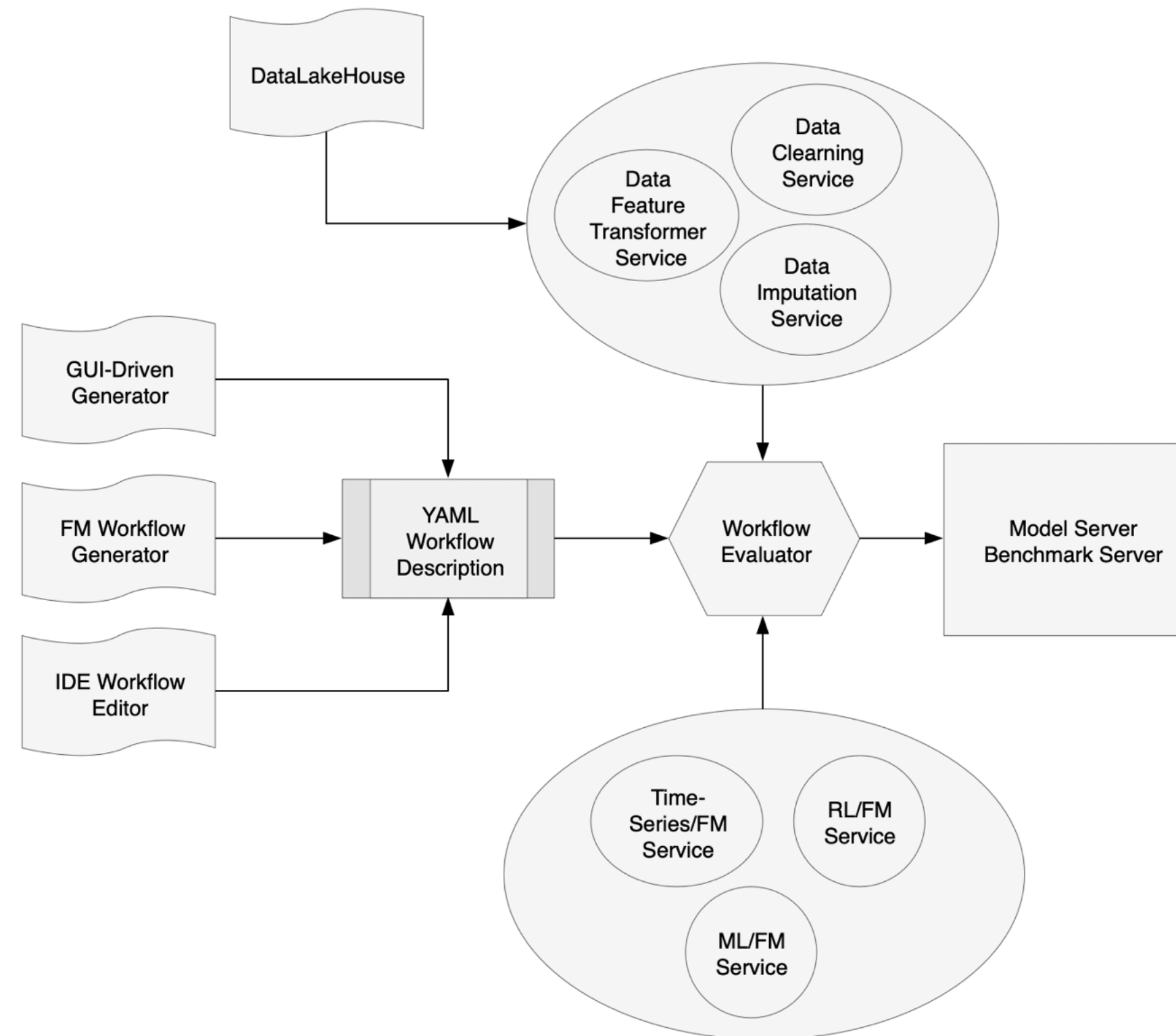
OUTLINE

- The Motivations Behind E2EAI (End-to-End AI)
- Components of E2EAI
- The Julia AI/ML Solution Use-case
- The Future

The Motivations Behind E2EAI

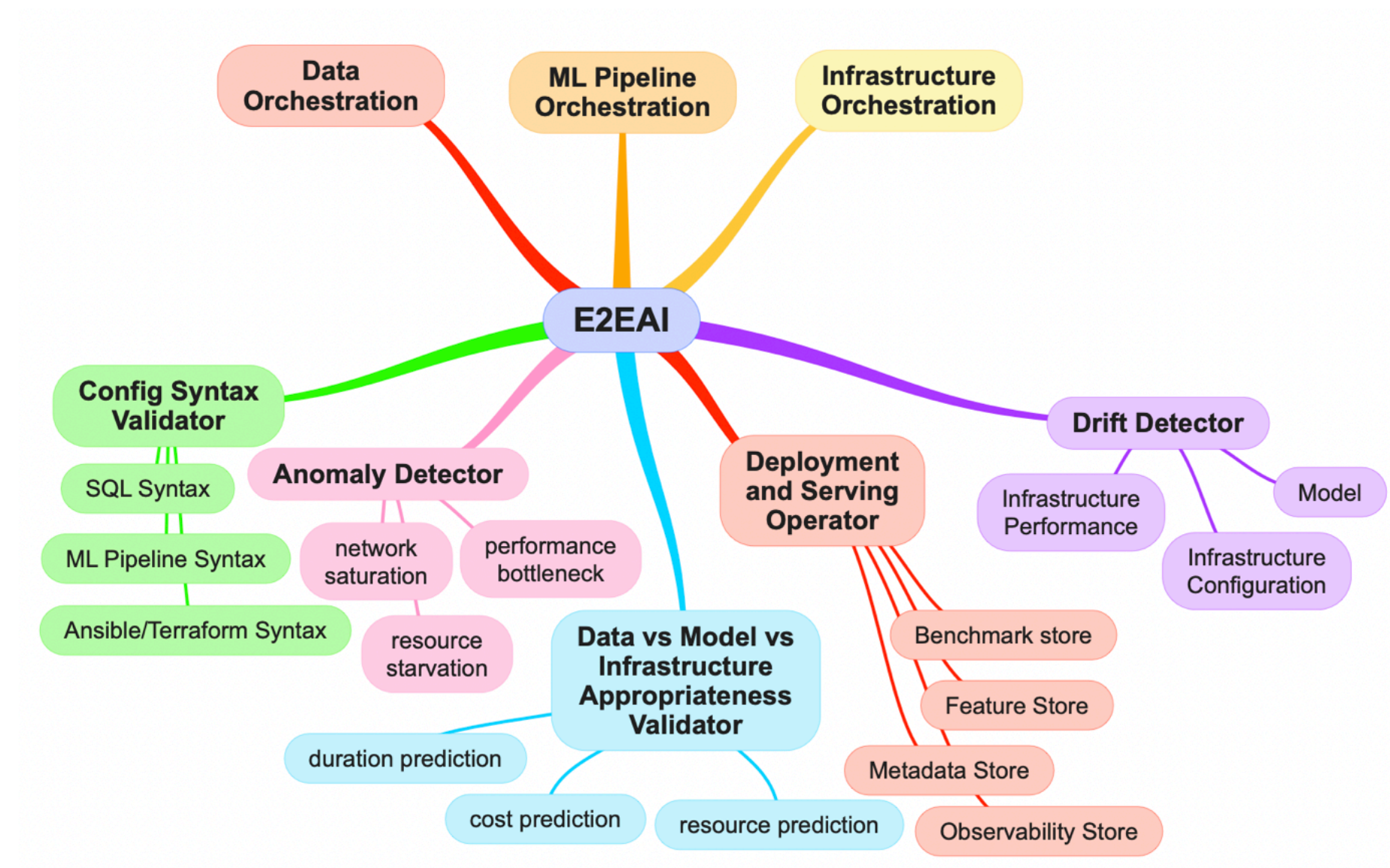
- currently, IaC and MLOPs are treated separately in deploying AI solutions
- issues with no tight integration:
 - difficult to identify optimal infrastructure
 - difficult to predict resource viability and feasibility
 - difficult to infer the cost of deployment
 - difficult to identify performance bottlenecks and root-cause analysis

End-to-End AI (E2EAI)



- E2EAI is a unified framework tightly integrating MLOps and IaC
 - single yaml file: Infrastructure + ML Pipeline + LifeCycle Management
 - reliance on yaml workflow templates imply zero to minimal coding
 - collection of yamls can be used as inputs to LLM for intent-driven E2EAI

Components of E2EAI



- SUNRISE-6G
 - Sustainable federation of Research Infrastructures for Scaling-up Experimentation in 6G
 - H2020 EU Project (3 years)

The Julia AI/ML Solution Use-case

- AutoMLPipeline workflow
- Integrating AutoMLPipeline in E2EAI



Load ML pipeline preprocessing components and models

```
[14]: using AutoMLPipeline;
import PythonCall; const PYC=PythonCall; warnings = PYC.pyimport("warnings"); warnings.filterwarnings("ignore")

#### Decomposition
pca = skoperator("PCA"); fa = skoperator("FactorAnalysis"); ica = skoperator("FastICA")
#### Scaler
rb = skoperator("RobustScaler"); pt = skoperator("PowerTransformer"); norm = skoperator("Normalizer")
mx = skoperator("MinMaxScaler"); std = skoperator("StandardScaler")
#### categorical preprocessing
ohe = OneHotEncoder()
#### Column selector
catf = CatFeatureSelector(); numf = NumFeatureSelector(); disc = CatNumDiscriminator()
#### Learners
rf = skoperator("RandomForestClassifier"); gb = skoperator("GradientBoostingClassifier"); lsvc = skoperator("LinearSVC")
svc = skoperator("SVC"); mlp = skoperator("MLPClassifier")
ada = skoperator("AdaBoostClassifier"); sgd = skoperator("SGDClassifier")
skrf_reg = skoperator("RandomForestRegressor"); skgb_reg = skoperator("GradientBoostingRegressor")
jrf = RandomForest(); tree = PrunedTree()
vote = VoteEnsemble(); stack = StackEnsemble(); best = BestLearner();
```



Prepare dataset for classification

```
[15]: # Make sure that the input feature is a dataframe and the target output is a 1-D vector.
      using AutoMLPipeline
      profbdata = getprofb()
      X = profbdata[:,2:end]
      Y = profbdata[:,1] |> Vector;
      head(x)=first(x,10)
      head(profbdata)
```

[15]: 10x7 DataFrame

Row	Home.Away	Favorite_Points	Underdog_Points	Pointspread	Favorite_Name	Underdog_name	Year
	String7	Int64	Int64	Float64	String3	String3	Int64
1	away	27	24	4.0	BUF	MIA	89
2	at_home	17	14	3.0	CHI	CIN	89
3	away	51	0	2.5	CLE	PIT	89
4	at_home	28	0	5.5	NO	DAL	89
5	at_home	38	7	5.5	MIN	HOU	89
6	at_home	34	20	6.0	DEN	KC	89
7	away	31	21	6.0	LAN	ATL	89
8	at_home	24	27	2.5	NYJ	NE	89
9	away	16	13	1.5	PHX	DET	89
10	at_home	40	14	3.5	LAA	SD	89



Pipeline to transform categorical features to one-hot encoding

```
[16]: pohe = catf |> ohe
      tr = fit_transform!(pohe,X,Y)
      head(tr)
```

[16]: 10x56 DataFrame

Row	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	x11	x12	x13	x14	x15	x16	x17	x18	x19	x20	x21	x22	x23	x24	x25	x26	x27	x28	x29	x30
	Float64	Float64	Float64	Float64	Float64	Float64	Float64	Float64	Float64	Float64	Float64	Float64	Float64	Float64	Float64	Float64	Float64	Float64	Float64	Float64	Float64	Float64	Float64	Float64	Float64	Float64	Float64	Float64	Float64	Float64
1	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	
2	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
7	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
10	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0



Pipeline to transform numerical features to pca and ica and concatenate them

```
[17]: pdec = (numf |> pca) + (numf |> ica)
      tr = fit_transform!(pdec,X,Y)
      head(tr)
```

[17]: 10×8 DataFrame

Row	x1	x2	x3	x4	x1_1	x2_1	x3_1	x4_1
	Float64	Float64	Float64	Float64	Float64	Float64	Float64	Float64
1	2.47477	7.87074	-1.10495	0.902431	0.079562	0.433191	1.21188	0.696414
2	-5.47113	-3.82946	-2.08342	1.00524	-0.566705	-0.851222	1.16879	0.177761
3	30.4068	-10.8073	-6.12339	0.883938	2.99276	-1.91161	1.1005	1.30006
4	8.18372	-15.507	-1.43203	1.08255	0.956083	-1.7026	1.18445	-0.49928
5	16.6176	-6.68636	-1.66597	0.978243	1.66491	-0.881385	1.19645	0.18067
6	10.2588	5.22112	0.0731649	0.928496	0.909602	0.448153	1.24404	0.328416
7	7.13435	5.60902	0.368661	0.939797	0.601462	0.511014	1.24843	0.232904
8	-1.16369	10.3011	-2.15564	0.86957	-0.334966	0.449637	1.18639	1.07029
9	-6.38764	-4.92017	-3.57339	0.986345	-0.673127	-1.20918	1.13179	0.498675
10	17.0567	0.672	-3.29448	0.879581	1.57446	-0.486426	1.16658	1.04515



More complex pipeline with robust scaling and power transform

```
[18]: ppt = (numf |> rb |> ica) + (numf |> pt |> pca)
      tr = fit_transform!(ppt,X,Y)
      head(tr)
```

[18]: 10×8 DataFrame

Row	x1	x2	x3	x4	x1_1	x2_1	x3_1	x4_1
	Float64	Float64	Float64	Float64	Float64	Float64	Float64	Float64
1	1.21202	-0.435836	0.694313	-0.081249	-0.64552	1.40289	-0.0284468	0.111773
2	1.16944	0.853312	0.182165	0.560799	-0.832404	0.475629	-1.14881	-0.01702
3	1.10269	1.89116	1.28493	-3.01142	1.54491	1.65258	-1.35967	-2.57866
4	1.1853	1.69734	-0.502448	-0.962693	1.32065	0.563565	-2.05839	-0.74898
5	1.19721	0.87072	0.171547	-1.67093	1.1223	1.45555	-0.88864	-0.776195
6	1.24399	-0.454814	0.321226	-0.90894	0.277462	1.70936	0.00130938	0.0768767
7	1.24826	-0.515726	0.227463	-0.599871	0.0977821	1.58007	-0.0364638	0.258464
8	1.18674	-0.450652	1.07069	0.331088	-1.31815	1.27463	0.00789964	-0.0553192
9	1.13287	1.21125	0.504449	0.663221	-1.29056	0.326316	-1.31916	-0.511818
10	1.16761	0.474565	1.0358	-1.58347	0.318224	1.76616	-0.28608	-1.02674



Evaluating complex pipeline with RandomForest learner

```
[19]: prf = (catf |> ohe) + (numf |> rb |> fa) + (numf |> pt |> pca) |> rf  
      crossvalidate(prf,X,Y,"accuracy_score")
```

```
fold: 1, 0.5522388059701493  
fold: 2, 0.7014925373134329  
fold: 3, 0.7058823529411765  
fold: 4, 0.6417910447761194  
fold: 5, 0.6865671641791045  
fold: 6, 0.7164179104477612  
fold: 7, 0.6716417910447762  
fold: 8, 0.6911764705882353  
fold: 9, 0.6417910447761194  
fold: 10, 0.7313432835820896  
errors: 0
```

```
[19]: (mean = 0.6740342405618964, std = 0.051874840276292904, folds = 10, errors = 0)
```



Evaluating complex pipeline with Linear SVM learner

```
[30]: plsvc = (catf |> ohe) + (numf |> rb |> fa) + (numf |> pt |> pca) |> lsvc  
crossvalidate(plsvc,X,Y,"accuracy_score")
```

```
fold: 1, 0.7014925373134329  
fold: 2, 0.6268656716417911  
fold: 3, 0.7647058823529411  
fold: 4, 0.7164179104477612  
fold: 5, 0.7014925373134329  
fold: 6, 0.6268656716417911  
fold: 7, 0.7611940298507462  
fold: 8, 0.7058823529411765  
fold: 9, 0.8208955223880597  
fold: 10, 0.7910447761194029  
errors: 0
```

```
[30]: (mean = 0.7216856892010537, std = 0.06423880195972397, folds = 10, errors = 0)
```



Parallel search of the best ML pipeline

```
[21]: using Random, DataFrames, Distributed
      nprocs() == 1 && addprocs()
      @everywhere using DataFrames; @everywhere using AutoMLPipeline
      @everywhere begin
        import PythonCall; const PYC=PythonCall; warnings = PYC.pyimport("warnings"); warnings.filterwarnings("ignore")
      end
      @everywhere begin
        profbdata = getprofb(); X = profbdata[:,2:end]; Y = profbdata[:,1] |> Vector;
      end
      @everywhere begin
        jrf = RandomForest(); ohe = OneHotEncoder(); catf = CatFeatureSelector(); numf = NumFeatureSelector()
        tree = PrunedTree(); ada = skoperator("AdaBoostClassifier"); disc = CatNumDiscriminator()
        sgd = skoperator("SGDClassifier"); std = skoperator("StandardScaler"); lsvc = skoperator("LinearSVC")
      end

      learners = @sync @distributed (vcat) for learner in [jrf,ada,sgd,lsvc,tree]
        pcmc = disc |> ((catf |> ohe) + (numf |> std)) |> learner
        println(learner.name[1:end-4])
        mean,sd,_ = crossvalidate(pcmc,X,Y,"accuracy_score",3)
        DataFrame(name=learner.name[1:end-4],mean=mean,sd=sd)
      end;
```

```
From worker 3:      AdaBoostClassifier
From worker 6:      prunetree
From worker 2:      rf
From worker 5:      LinearSVC
From worker 4:      SGDClassifier
From worker 2:      fold: 1, 0.7098214285714286
From worker 6:      fold: 1, 0.5982142857142857
From worker 4:      fold: 1, 0.6741071428571429
From worker 5:      fold: 1, 0.6830357142857143
From worker 2:      fold: 2, 0.6428571428571429
From worker 4:      fold: 2, 0.6607142857142857
From worker 6:      fold: 2, 0.5982142857142857
From worker 5:      fold: 2, 0.7321428571428571
From worker 3:      fold: 1, 0.7410714285714286
From worker 2:      fold: 3, 0.65625
From worker 2:      errors: 0
From worker 6:      fold: 3, 0.5758928571428571
From worker 6:      errors: 0
From worker 4:      fold: 3, 0.6830357142857143
From worker 4:      errors: 0
From worker 5:      fold: 3, 0.6875
From worker 5:      errors: 0
```



Best Pipeline

```
[22]: @show sort!(learners, :mean, rev=true);
```

sort!(learners, :mean, rev = true) = 5×3 DataFrame

Row	name String	mean Float64	sd Float64
1	LinearSVC	0.700893	0.0271552
2	AdaBoostClassifier	0.686012	0.0501777
3	SGDClassifier	0.672619	0.0112349
4	rf	0.669643	0.0354342
5	prunetree	0.590774	0.0128873





E2EAI Application



Infrastructure Creation Automation

inventory.yaml

```
# infra
kind: Cluster
metadata:
  name: e2eai-cluster
spec:
  hosts:
    - ssh:
        address: 10.10.7.10
        user: root
        port: 22
        role: controller
    - ssh:
        address: 10.10.7.11
        user: root
        port: 22
        role: worker
    - ssh:
        address: 10.10.7.12
        user: root
        port: 22
        role: worker
    - ssh:
        address: 10.10.7.13
        role: worker
        user: root
        role: worker
  ---
```

- ❖ Identify IP addresses of Controllers and Workers and edit inventory.yaml
- ❖ Run the **ssh-copy-id** as root to Controller and Workers for passwordless authentication by exporting public key
- ❖ Run the **iac.sh** script to automate infrastructure creation
- ❖ Automation tested in Debian/Ubuntu and Redhat

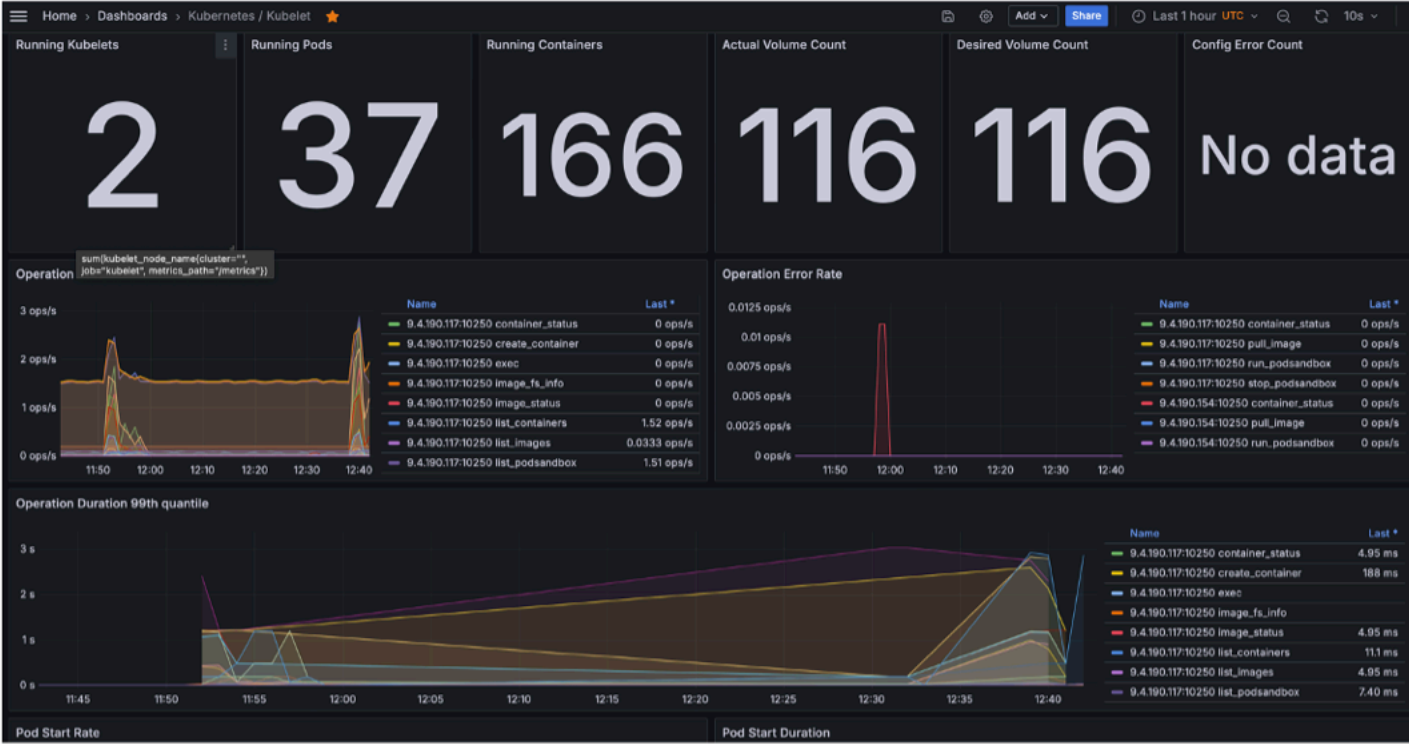
kubectl get nodes

NAME	STATUS	ROLES	AGE	VERSION
e2eai1.myapp.mydomain.com	Ready	<none>	41d	v1.29.3+k0s
e2eai2.myapp.mydomain.com	Ready	<none>	41d	v1.29.3+k0s
e2eai3.myapp.mydomain.com	Ready	<none>	41d	v1.29.3+k0s



E2EAI Cluster Observability

Cluster General Information



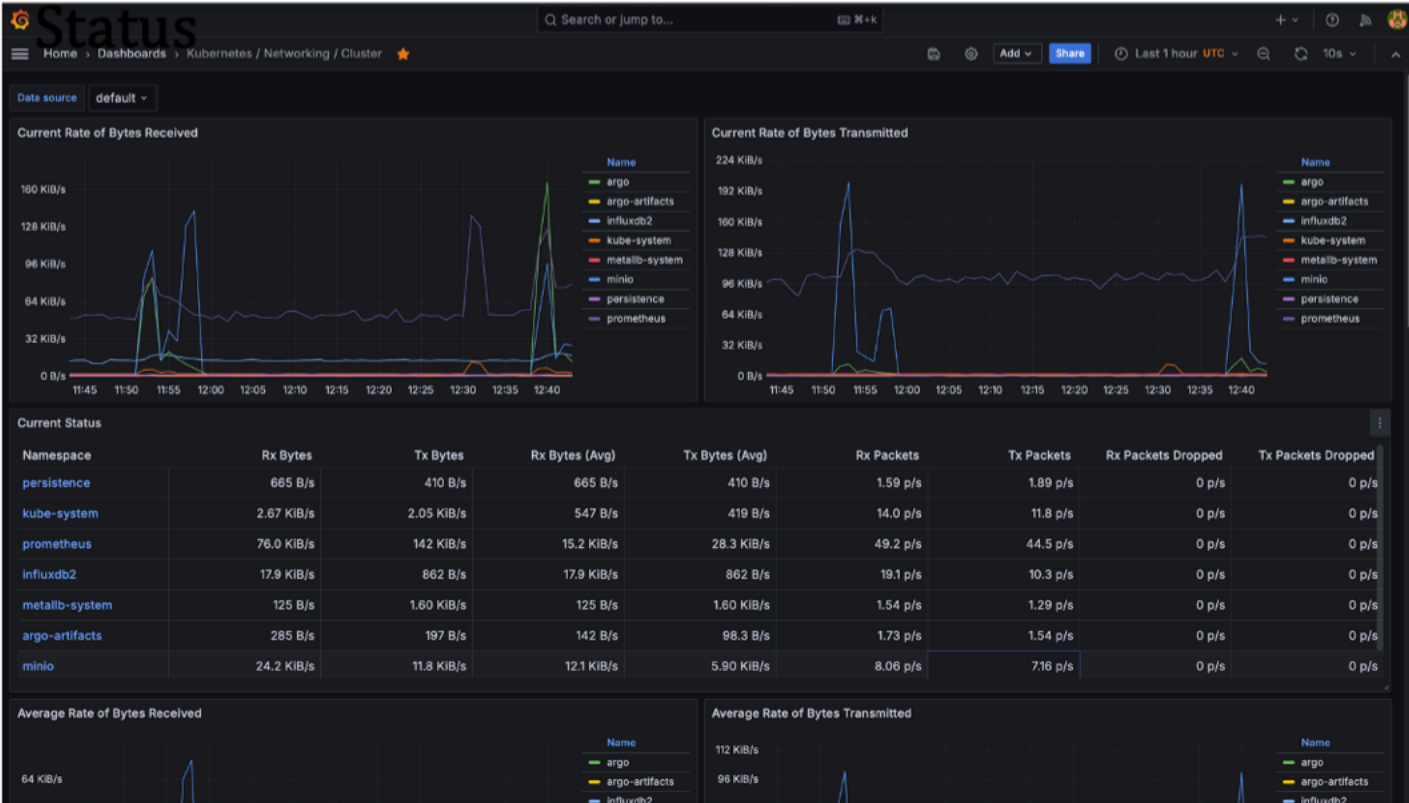
Cluster Hardware Utilization



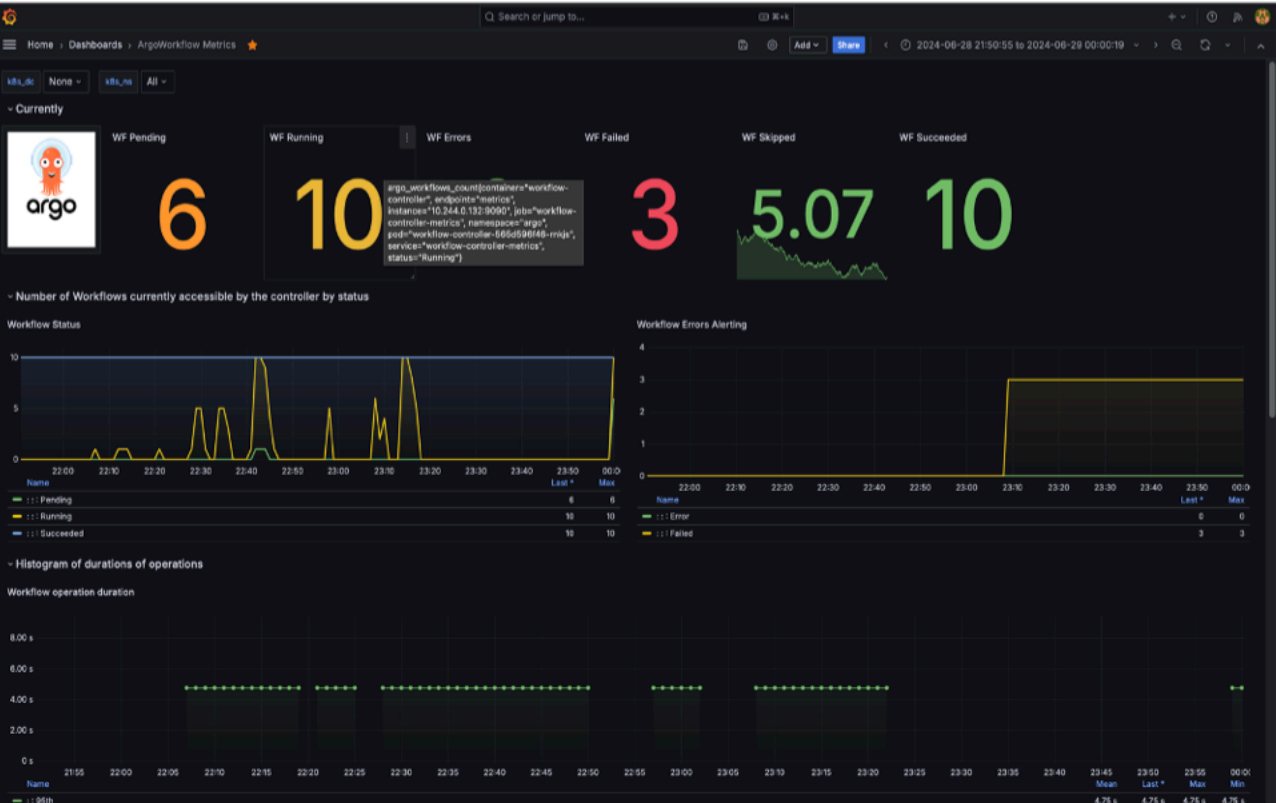
Workflow Queue



Network Cluster



Workflow Status



Workflow Experiment Duration



AI as a Service: Zero Coding Using Workflow Template

Cluster Workflow Templates

+ CREATE NEW CLUSTER WORKFLOW TEMPLATE

NAME	CREATED
argpassing-template	15d18h ago
cluster-workflow-template-inner-dag	32d13h ago
cluster-workflow-template-inner-steps	32d13h ago
cluster-workflow-template-random-fail-template	32d13h ago
cluster-workflow-template-submittable	32d13h ago
cluster-workflow-template-whalesay-template	32d13h ago
parallel-cluster-workflow-template	32d13h ago

Cluster scoped Workflow templates are reusable templates you can create new workflows from. You can find manifests [in the e](#)

Cluster Workflow Templates / bicomplex-amp-template

CLUSTER WORKFLOW TEMPLATE DETAILS

+ SUBMIT UPDATE DELETE

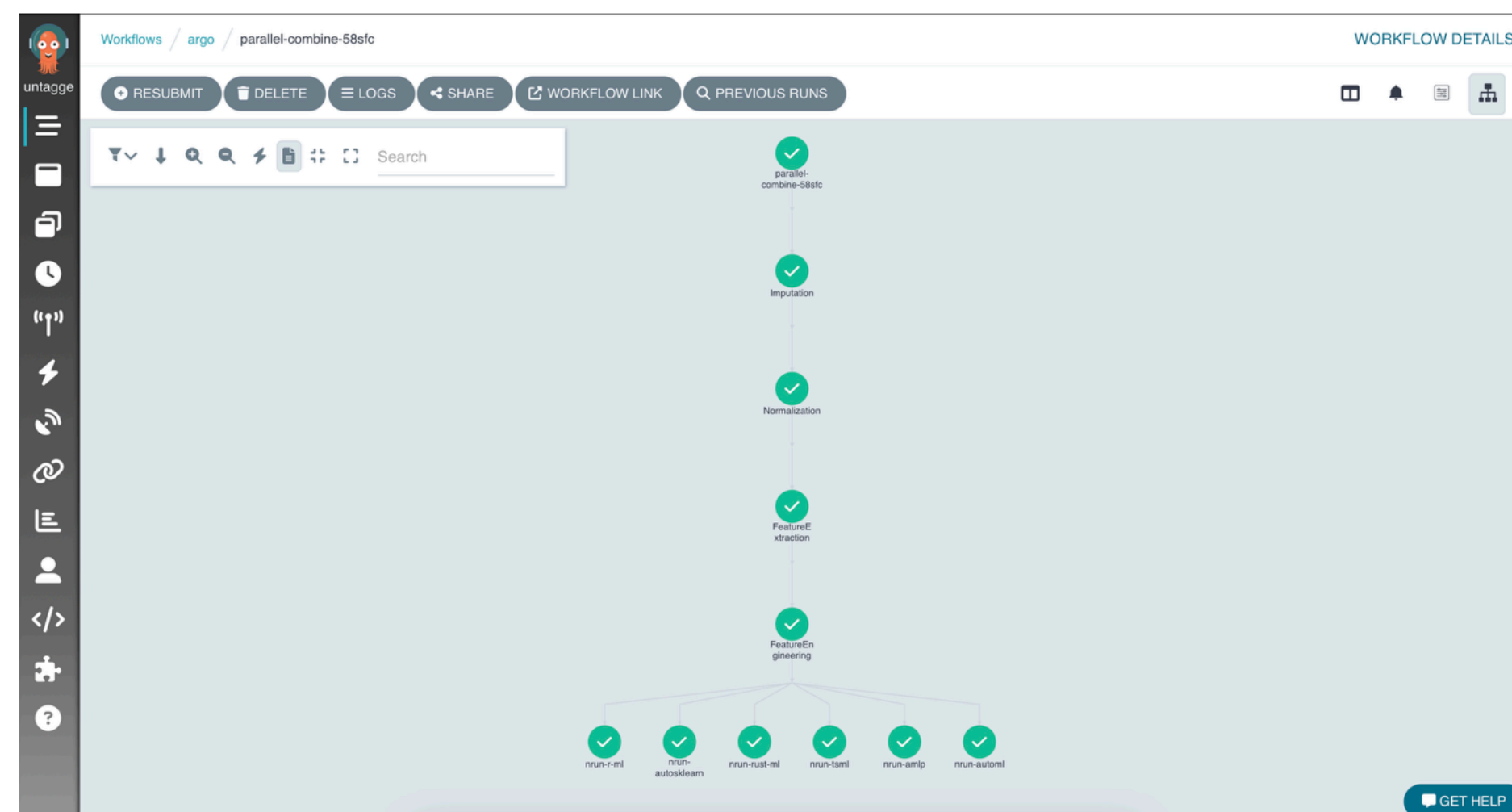
MANIFESTSPECMETADATWORKFLOW METADATA

JSON/YAML METADATA SPEC

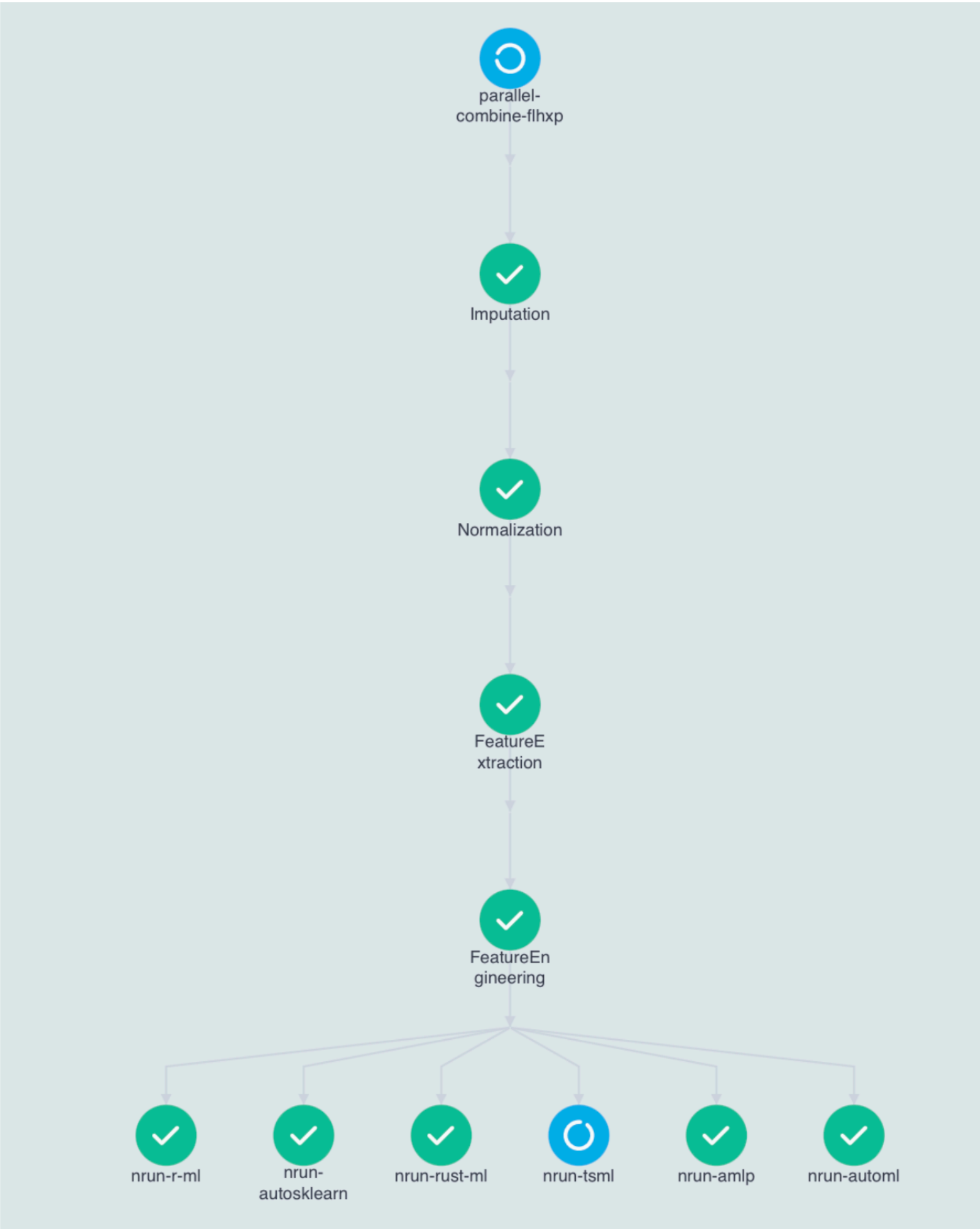
```
1 metadata:
2   name: bicomplex-amp-template
3   uid: 546a2486-bab6-4a9c-95c6-d6ce43010614
4   resourceVersion: '1267908'
5   generation: 1
6   creationTimestamp: '2024-06-30T14:07:05Z'
7   managedFields:
8     - manager: argo
9       operation: Update
10      apiVersion: argoproj.io/v1alpha1
11      time: '2024-06-30T14:07:05Z'
12      fieldType: FieldsV1
13      fieldsV1:
14        f:spec: {}
15  spec:
16    templates:
17      - name: run-main
18        inputs:
19          parameters:
20            - name: input
21            - name: predictiontype
22            - name: folds
23            - name: workers
24          outputs: {}
25          metadata: {}
26        steps:
27          - name: low-complexity
28            template: run-amp
29            arguments:
30              parameters:
31                - name: complexity
32                value: low
33                - name: input
```

Basic completion for YAML. Switch to JSON for full auto-completion. [Learn how to get auto-completion in your IDE.](#)

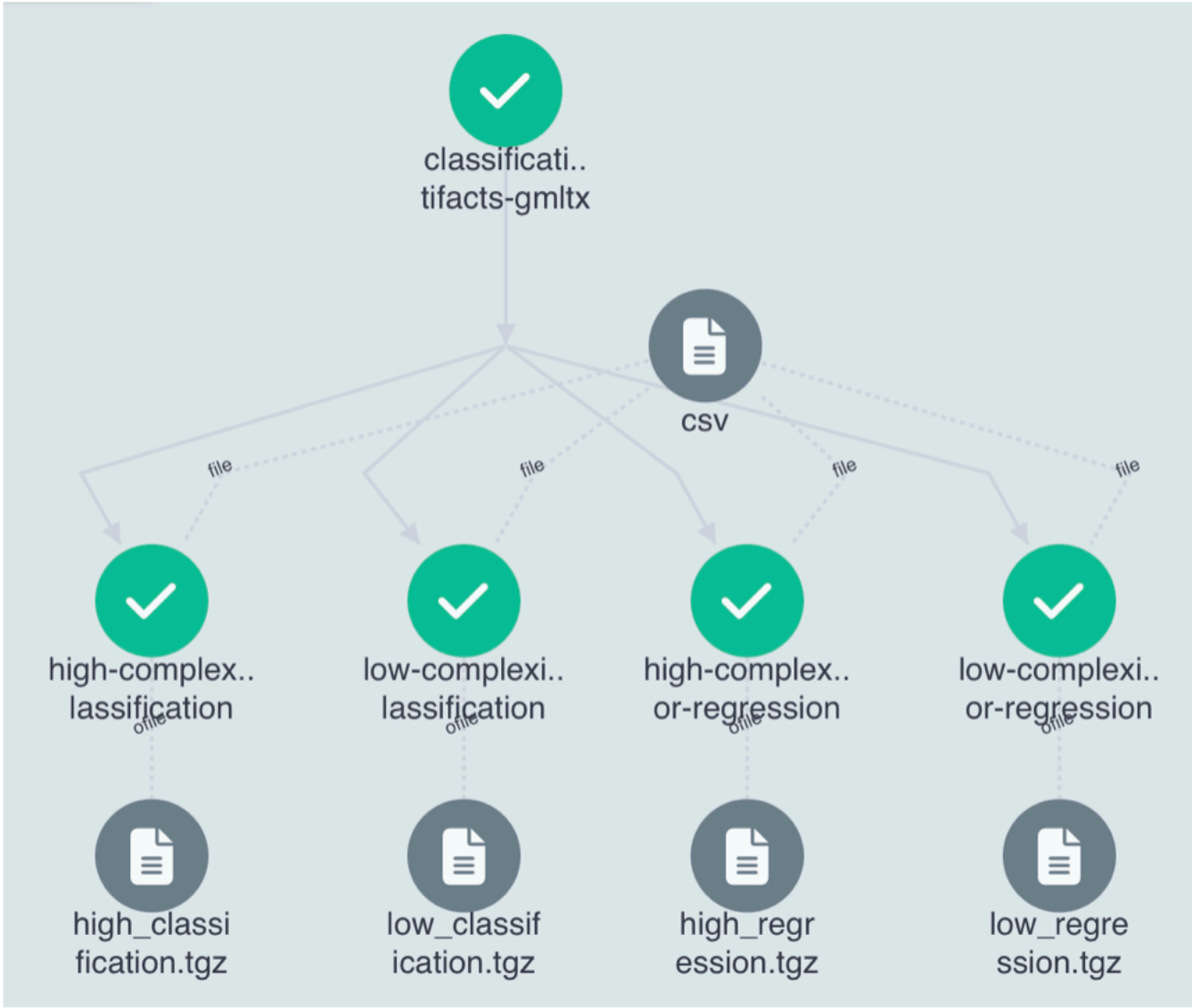
GET HELP



Explicit ML Pipeline

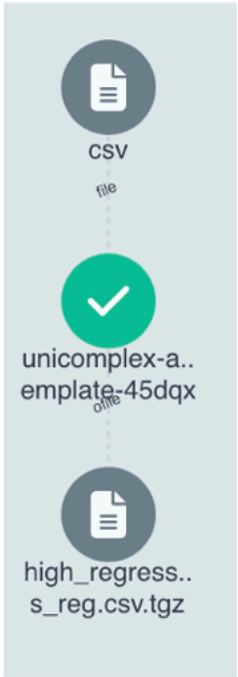


Optimal Pipeline Discovery by AutoML



Low vs High Pipeline Complexity

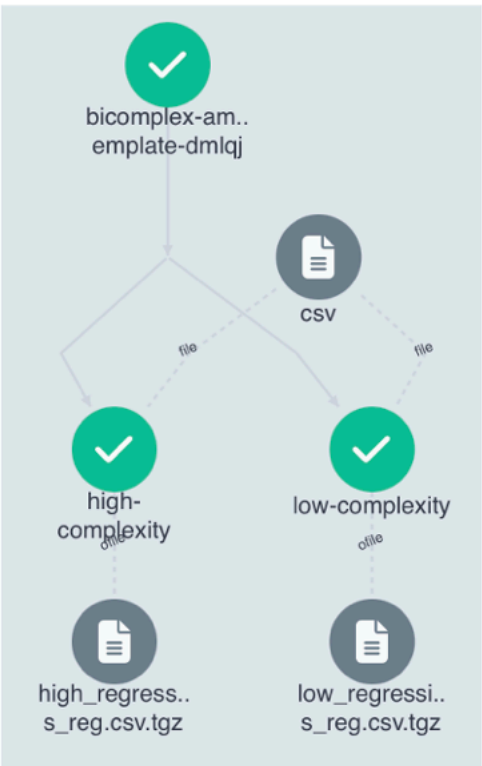
```
argo -n argo submit \  
  --from clusterworkflowtemplate/unicomplex-amlp-template \  
  -p workers=3 -p complexity=high -p input=diabetes_reg.csv \  
  -p predictiontype=regression
```



Row	Description String	mean Float64	sd Float64	Pipeline Pipeline
1	(std > ic...	2896.7...	272.50...	Pipelin...
2	(rb > pca...	2922.9...	7.87...	Pipelin...
3	(noop > f...	2925.5...	117.03...	Pipelin...
4	(rb > noo...	2926.3...	127.00...	Pipelin...
5	(rb > noo...	2934.8...	138.31...	Pipelin...
6	(mx > pca...	2949.7...	451.59...	Pipelin...
7	(std > ic...	2951.8...	292.75...	Pipelin...
8	(rb > pca...	2952.1...	520.47...	Pipelin...
...
570	(pt > pca...	2.1...	2.99...	Pipelin...
571	(pt > ica...	1.8...	2.67...	Pipelin...
572	(pt > noo...	1.5...	2.13...	Pipelin...
573	(std > ic...	1.9...	Inf ...	Pipelin...
574	(pt > noo...	5.7...	Inf ...	Pipelin...
575	(pt > noo...	5.4...	Inf ...	Pipelin...
576	(mx > ica...	Inf ...	NaN ...	Pipelin...

561 rows omitted
best model: (std |> ica) + (mx |> pca) |> lars
mean ± sd: 2896.77 ± 272.51

```
argo -n argo submit \  
  --from clusterworkflowtemplate/bicomplex-amlp-template \  
  -p workers=3 -p input=diabetes_reg.csv \  
  -p predictiontype=regression
```



Row	Description String	mean Float64	sd Float64	Pipeline Pipeline
1	(noop > n...	2928.2...	185.02...	Pipelin...
2	(noop > i...	2930.7...	324.98...	Pipelin...
3	(std > pc...	2946.3...	103.04...	Pipelin...
4	(std > pc...	2950.4...	91.59...	Pipelin...
5	(rb > pca...	2955.1...	214.55...	Pipelin...
6	(noop > i...	2957.3...	57.15...	Pipelin...
7	(mx > pca...	2958.8...	100.88...	Pipelin...
8	(mx > noo...	2960.1...	13.51...	Pipelin...
...
570	(noop > i...	3.4...	4.88...	Pipelin...
571	(mx > pca...	9.6...	1.37...	Pipelin...
572	(rb > ica...	3.6...	5.18...	Pipelin...
573	(std > ic...	2.5...	Inf ...	Pipelin...
574	(pt > pca...	Inf ...	NaN ...	Pipelin...
575	(pt > pca...	Inf ...	NaN ...	Pipelin...
576	(norm > n...	Inf ...	NaN ...	Pipelin...

561 rows omitted
best model: (noop |> noop) + (mx |> pca) |> lars
mean ± sd: 2928.2 ± 185.03

Row	Description String	mean Float64	sd Float64	Pipeline Pipeline
1	(std > ic...	2999.03	9.04...	Pipelin...
2	(noop > i...	3011.38	161.39...	Pipelin...
3	(std > pc...	3012.85	8.36...	Pipelin...
4	(norm > n...	3018.28	332.26...	Pipelin...
5	(mx > noo...	3034.49	331.61...	Pipelin...
6	(pt > pca...	3040.44	45.22...	Pipelin...
7	(rb > pca...	3047.21	281.66...	Pipelin...
8	(norm > f...	3048.61	333.40...	Pipelin...
...
18	(rb > ica...	3217.64	104.95...	Pipelin...
19	(norm > i...	3248.68	10.58...	Pipelin...
20	(pt > ica...	3352.62	147.94...	Pipelin...
21	(noop > f...	3650.02	254.86...	Pipelin...
22	(noop > n...	3742.79	759.00...	Pipelin...
23	(mx > fa)	3802.83	185.58...	Pipelin...
24	(noop > p...	3861.06	526.55...	Pipelin...

9 rows omitted
best model: (std |> ica) |> sgd
mean ± sd: 2999.03 ± 9.05



Low Complexity Pipeline for Classification

```
From worker 3: fold: 1, 0.8625041025041024
From worker 3: fold: 2, 0.9363636363636364
From worker 3: errors: 0
From worker 5: fold: 2, 0.9732905982905983
From worker 5: errors: 0

24x4 DataFrame
Row  Description  mean      sd      Pipeline
String Float64  Float64  Pipeline
1  (norm |> i...  0.9559...  0.0135...  Pipelin...
2  (std |> ic...  0.9450...  0.0253...  Pipelin...
3  (mx |> pca...  0.9422...  0.0234...  Pipelin...
4  (norm |> f...  0.9304...  0.0264...  Pipelin...
5  (pt |> noo...  0.9279...  0.0641...  Pipelin...
6  (mx |> fa)...  0.9265...  0.0139...  Pipelin...
7  (rb |> noo...  0.9216...  0.0295...  Pipelin...
8  (rb |> ica...  0.9208...  0.0732...  Pipelin...
:      :      :      :      :
18  (mx |> ica...  0.8753...  0.0226...  Pipelin...
19  (rb |> fa)...  0.8592...  0.0271...  Pipelin...
20  (noop |> p...  0.8459...  0.0307...  Pipelin...
21  (noop |> f...  0.8429...  0.0536...  Pipelin...
22  (pt |> fa)...  0.8351...  0.0251...  Pipelin...
23  (noop |> n...  0.8335...  0.1480...  Pipelin...
24  (norm |> n...  0.6651...  0.0126...  Pipelin...
      9 rows omitted

best model: (norm |> ica) |> sgd
mean ± sd: 0.96 ± 0.01
```



High Complexity Pipeline for Classification

```
From worker 5: fold: 2, 0.9623015873015873
From worker 5: errors: 0
From worker 5: fold: 1, 0.9310144927536231
From worker 5: fold: 2, 0.9655172413793104
From worker 5: errors: 0
576x4 DataFrame
Row      Description      mean      sd      Pipeline
      String      Float64  Float64  Pipeline
-----
1  (norm |> f...  0.9901...  0.0138...  Pipelin...
2  (rb |> noo...  0.9895...  0.0147...  Pipelin...
3  (mx |> fa)...  0.9876...  0.0174...  Pipelin...
4  (std |> pc...  0.9871...  0.0006...  Pipelin...
5  (pt |> pca...  0.9871...  0.0181...  Pipelin...
6  (noop |> i...  0.9871...  0.0006...  Pipelin...
7  (mx |> noo...  0.9871...  0.0006...  Pipelin...
8  (std |> pc...  0.9871...  0.0014...  Pipelin...
:      :      :      :      :
570  (norm |> p...  0.8932...  0.0393...  Pipelin...
571  (rb |> ica...  0.8916...  0.0735...  Pipelin...
572  (mx |> ica...  0.8913...  0.0078...  Pipelin...
573  (noop |> n...  0.8891...  0.0052...  Pipelin...
574  (rb |> fa)...  0.8888...  0.0    ...  Pipelin...
575  (mx |> pca...  0.8836...  0.0238...  Pipelin...
576  (noop |> i...  0.8671...  0.0199...  Pipelin...
      561 rows omitted
best model: (norm |> fa) + (mx |> pca) |> rbfsvc
mean ± sd: 0.99 ± 0.01
```



Sample AutoML Workflow Template

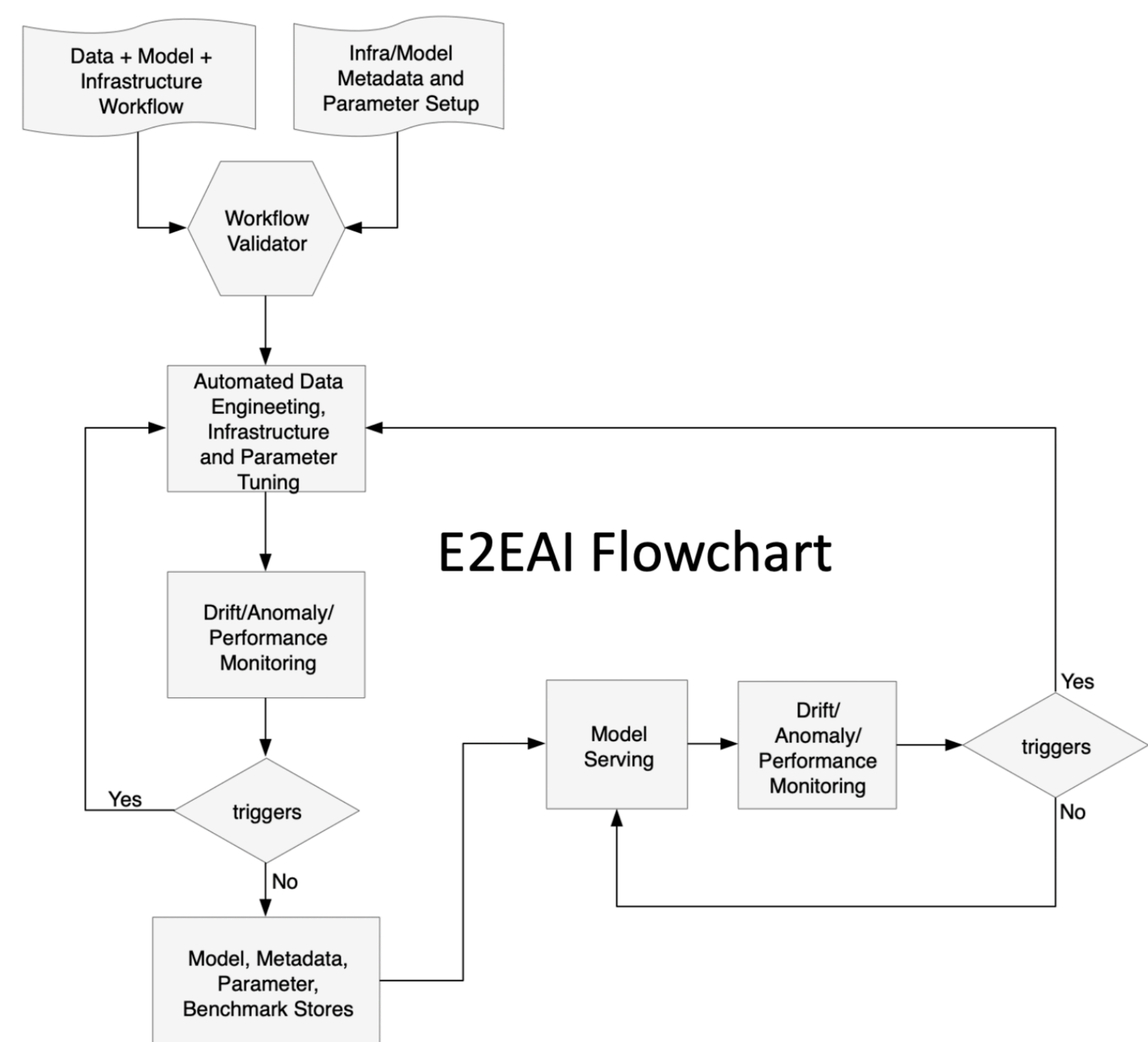
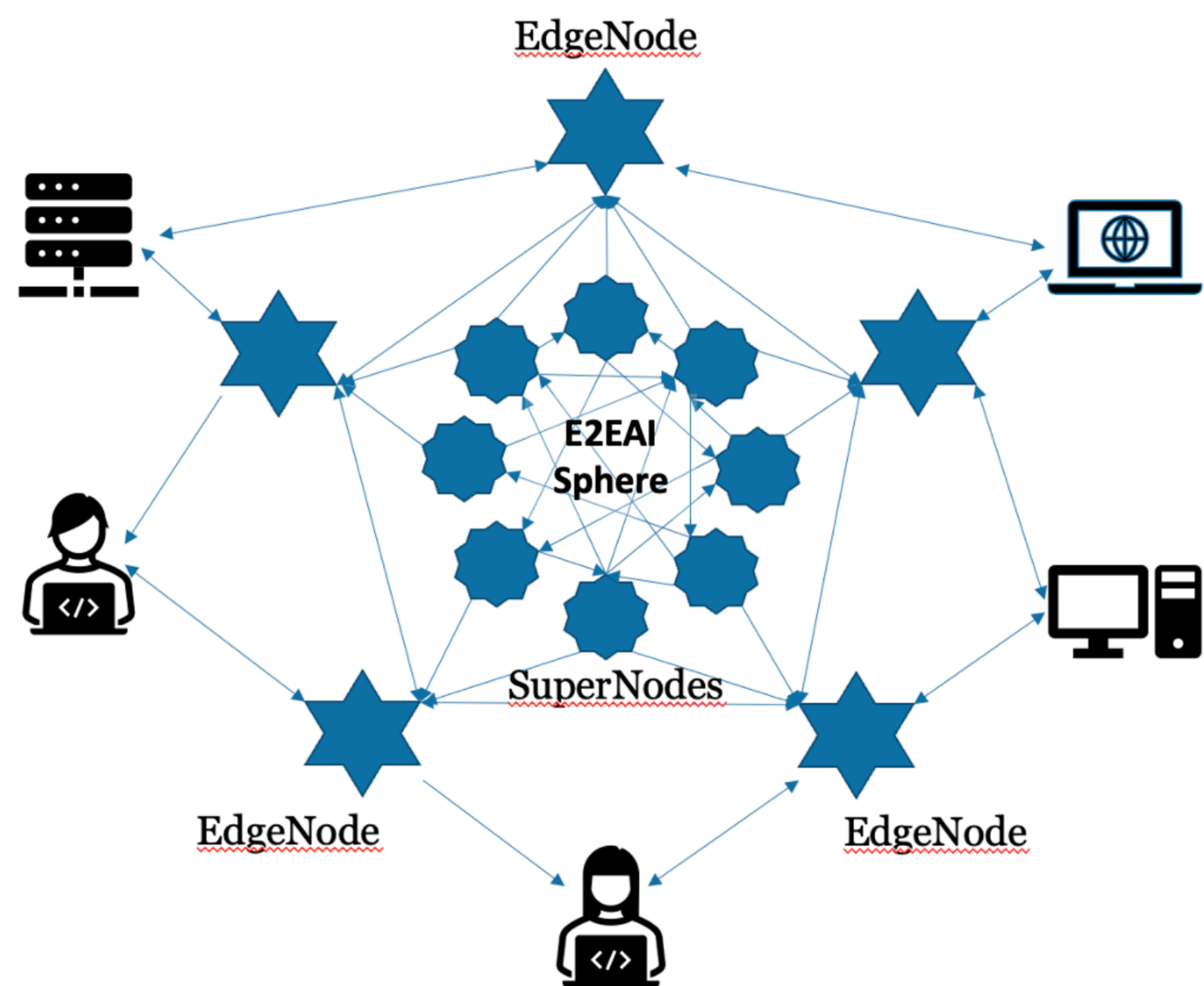
```
argo -n argo submit --from clusterworkflowtemplate/automl -p workers=6 -p input=sensor.csv -p predictiontype=classification
```

```
apiVersion: argoproj.io/v1alpha1
kind: ClusterWorkflowTemplate
metadata:
  name: automl
spec:
  entrypoint: run-automl
  arguments:
    parameters:
      - name: complexity
        value: low
      - name: input
        value: data_reg.csv
      - name: predictiontype
        value: regression
      - name: folds
        value: 10
      - name: workers
        value: 10
  metrics:
    prometheus:
      - name: exec_duration_gauge
        labels:
          - key: name
            value: "automl_{{inputs.parameters.complexity}}_{{inputs.parameters.input}}"
        help: "Duration gauge by name"
        gauge:
          value: "{{workflow.duration}}"
```

```
templates:
  - name: run-amlp
    inputs:
      parameters:
        - name: complexity
        - name: input
        - name: predictiontype
        - name: folds
        - name: workers
      artifacts:
        - name: file
          path: /inputfile
          s3:
            key: csv/
    container:
      image: automl/automlpipeline:latest
      command:
      args:
        - "-t {{inputs.parameters.predictiontype}}"
        - "-c {{inputs.parameters.complexity}}"
        - "-f {{inputs.parameters.folds}}"
        - "-w {{inputs.parameters.workers}}"
        - "-o /outputfile"
        - "/inputfile/{{inputs.parameters.input}}"
    outputs:
      artifacts:
        - name: ofile
          path: /outputfile
          s3:
            key: "output/{{inputs.parameters.complexity}}_{{inputs.parameters.input}}.tgz"
```



The Future



- Unified Control Plane and Intent-Driven E2EAI

Acknowledgement



Funded by the
European Union



This work has been funded by the SUNRISE-6G project, Grant number 101139257, co-funded by the European Union and Smart Networks and Services Joint Undertaking (SNS JU).

DISCLAIMER

"Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union and Smart Networks and Services Joint Undertaking (SNS JU). Neither the European Union nor the granting authority can be held responsible for them."

THANK YOU!

