
MLJ

A machine learning toolbox for julia

Anthony Blaom et al.

The Alan Turing Institute is the national centre for data science, headquartered at the British Library.



THE UNIVERSITY
of EDINBURGH

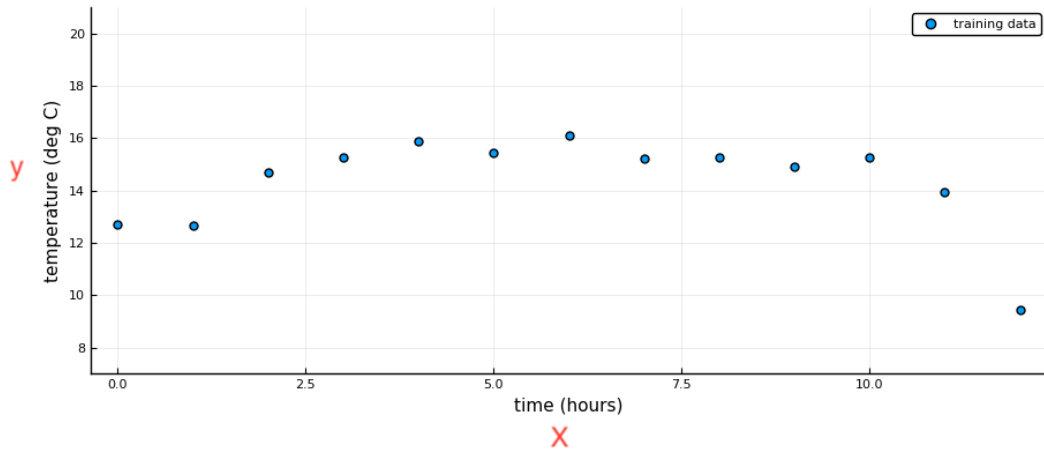




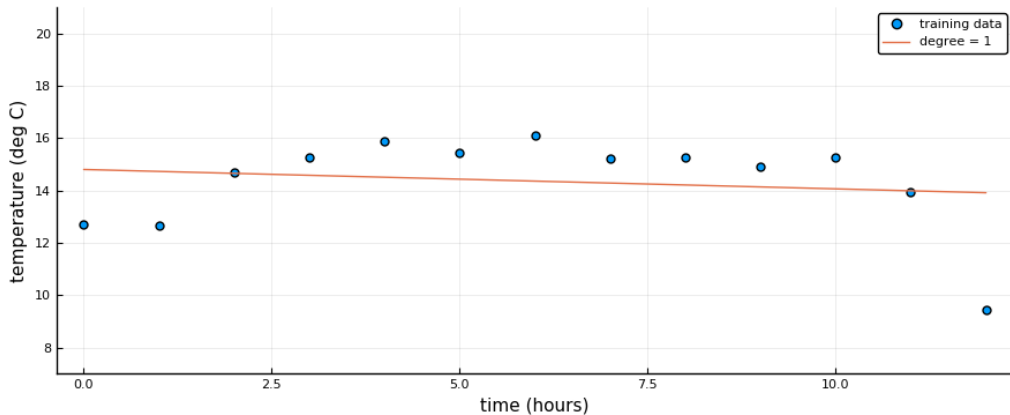
Supervised Learning

Learning to **predict** some target variable **y** from a knowledge of some other variables **X** (the *input features*).

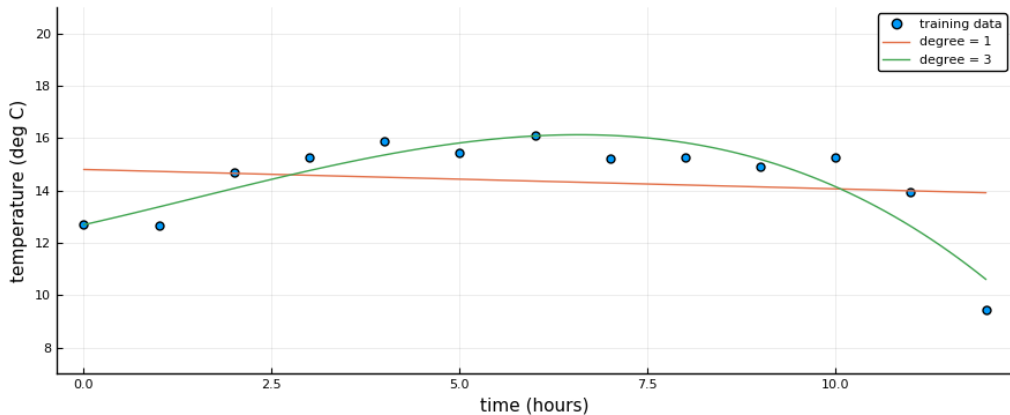
Supervised Learning



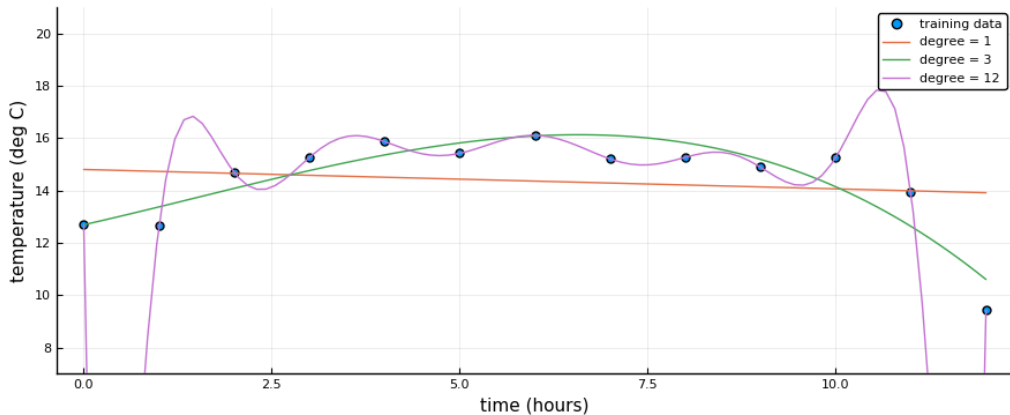
Supervised Learning



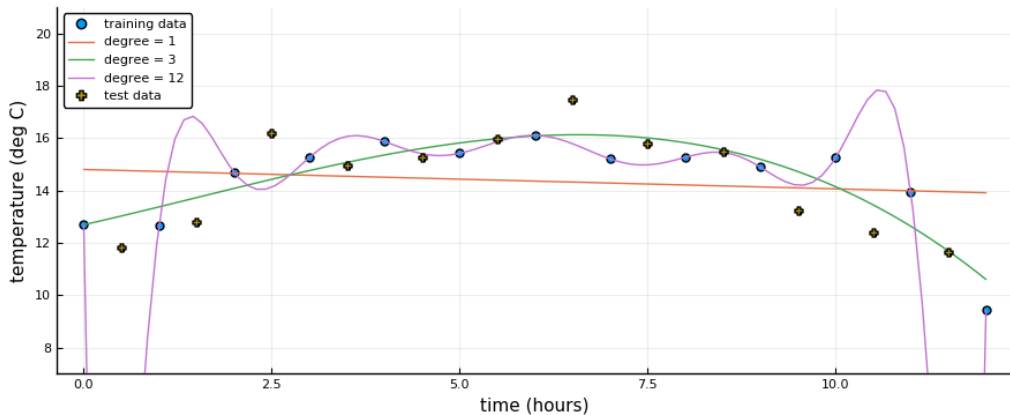
Supervised Learning



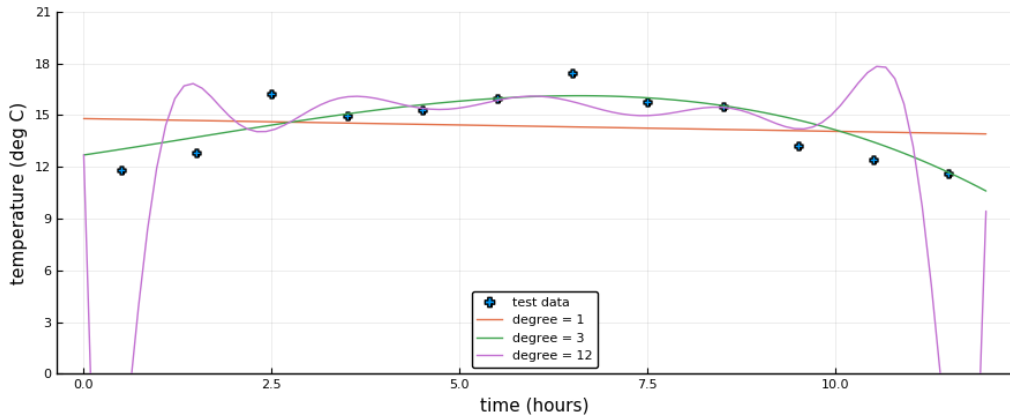
Supervised Learning



Supervised Learning

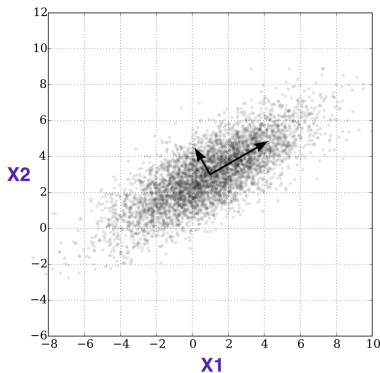


Supervised Learning



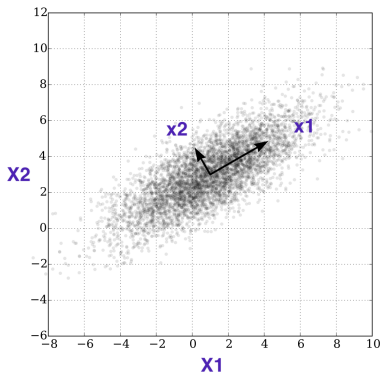
Unsupervised Learning

Learning data **transformations**, e.g., dimension reduction

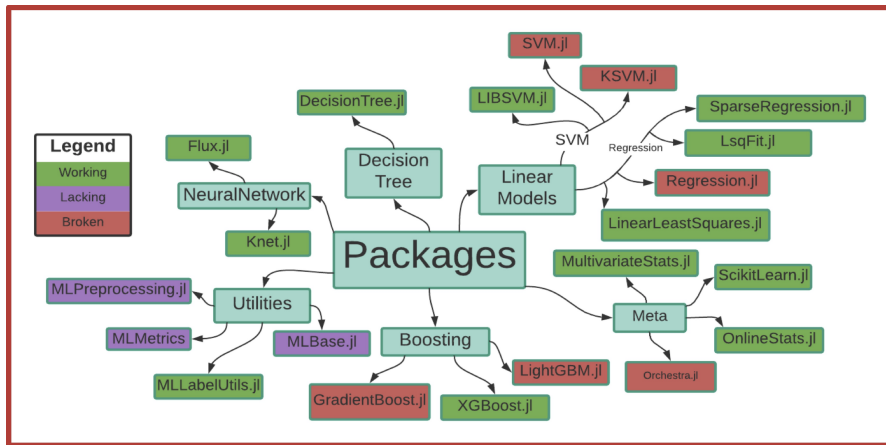


Unsupervised Learning

Learning data **transformations**, e.g., dimension reduction



A plethora of models



Machine learning toolboxes

A machine learning toolbox:

Machine learning toolboxes

A machine learning toolbox:

- Provides a **uniform interface** for **fitting**, **evaluating**, **tuning** and **benchmarking** models.

Machine learning toolboxes

A machine learning toolbox:

- Provides a **uniform interface** for **fitting**, **evaluating**, **tuning** and **benchmarking** models.
- Provides common **preprocessing tasks** (such as data cleaning and type coercion)

Machine learning toolboxes

A machine learning toolbox:

- Provides a **uniform interface** for **fitting**, **evaluating**, **tuning** and **benchmarking** models.
- Provides common **preprocessing tasks** (such as data cleaning and type coercion)
- Allows for model **composition** (aka pipelining)

Toolboxes in other ecosystems

		將軍
		

Goals for MLJ

Goals for MLJ

- Want **usability**, interoperability, extensibility and reproducibility

Goals for MLJ

- Want **usability**, interoperability, extensibility and reproducibility
- Want avoid common **pain-points**:
 - Identifying all models that solve a given task

Goals for MLJ

- Want **usability**, interoperability, extensibility and reproducibility
- Want avoid common **pain-points**:
 - Identifying all models that solve a given task
 - Routine operations requiring a lot of code

Model search and tasks

```
using MLJ
```

```
models()
```

Dict{Any,Any} with 9 entries:

```
"MultivariateStats" => Any["ICA", "RidgeRegressor", "KernelPCA", "PCA"]
"MLJ"                => Any["MLJ.Constant.DeterministicConstantRegressor", "MLJ.Constant.DeterministicConstantClassifier"]
"DecisionTree"       => Any["DecisionTreeRegressor", "DecisionTreeClassifier"]
"ScikitLearn"        => Any["SVMLRegressor", "SVMNuClassifier", "ElasticNet", "ElasticNetClassifier"]
"LIBSVM"             => Any["EpsilonSVR", "LinearSVC", "NuSVR", "NuSVC", "SVC", "SVCClassifier"]
"Clustering"         => Any["KMeans", "KMedoids"]
"GLM"                => Any["OLSRegressor", "GLMCountRegressor"]
"NaiveBayes"         => Any["GaussianNBClassifier", "MultinomialNBClassifier"]
"XGBoost"            => Any["XGBoostCount", "XGBoostRegressor", "XGBoostClassifier"]
```

Model search and tasks

```
task = load_boston()  
models(task)
```

Dict{Any,Any} with 6 entries:

```
"MultivariateStats" => Any["RidgeRegressor"]  
"MLJ"                => Any["MLJ.Constant.DeterministicConstantRegressor", "ML...  
"DecisionTree"       => Any["DecisionTreeRegressor"]  
"ScikitLearn"        => Any["SVMLRegressor", "ElasticNet", "ElasticNetCV", "SV...  
"LIBSVM"             => Any["EpsilonSVR", "NuSVR"]  
"XGBoost"            => Any["XGBoostRegressor"]
```


Quick performance evaluation

```
@load DecisionTreeRegressor # load code

tree_ = DecisionTreeRegressor(n_subfeatures=3)
tree = machine(tree_, task)
evaluate!(tree,
           resampling=Holdout(fraction_train=0.7),
           measure=[rms, mav])
```

Quick performance evaluation

```
@load DecisionTreeRegressor # load code

tree_ = DecisionTreeRegressor(n_subfeatures=3)
tree = machine(tree_, task)
evaluate!(tree,
           resampling=Holdout(fraction_train=0.7),
           measure=[rms, mav])

(MLJ.rms = 8.795939100833767,
 MLJ.mav = 5.785953164160401,)
```

Meta-algorithms as model wrappers

```
forest_ = EnsembleModel(atom=tree_, n=10)
```

Meta-algorithms as model wrappers

```
forest_ = EnsembleModel(atom=tree_, n=10)

r1 = range(forest_, :bagging_fraction, lower=0.4, upper=1.0);
r2 = range(forest_, :(atom.n_subfeatures), lower=1, upper=12)

self_tuning_forest_ = TunedModel(model=forest_,
                                tuning=Grid(),
                                resampling=CV(),
                                ranges=[r1,r2],
                                measure=rms)
```

Meta-algorithms as model wrappers

```
self_tuning_forest = machine(self_tuning_forest_, task)
```

```
evaluate!(self_tuning_forest,  
          resampling=CV(),  
          measure=[rms,rmslp1])
```

```
(MLJ.rms = [2.91827, 3.40544, 4.60971, 4.54709, 8.12081, 3.79819],  
 MLJ.rmslp1 = [0.148546, 0.119118, 0.148812, 0.134863, 0.345141, 0.221093],)
```

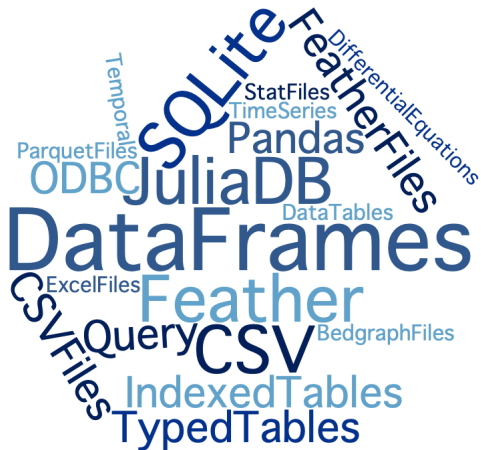
Goals for MLJ

- Want **usability**, interoperability, extensibility and reproducibility
- Want avoid common **pain-points**:
 - Identifying all models that solve a given task
 - Routine operations requiring a lot of code

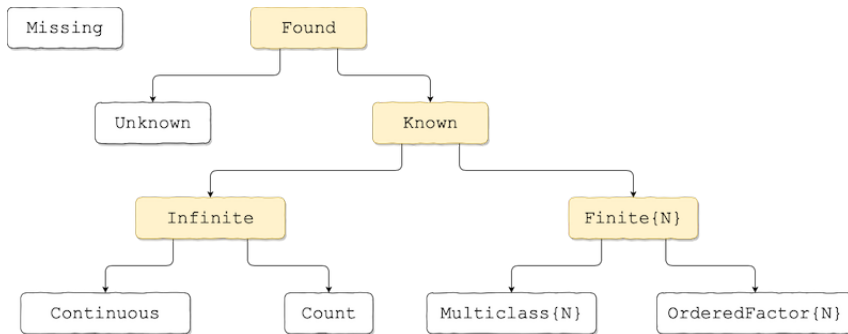
Goals for MLJ

- Want **usability**, interoperability, extensibility and reproducibility
- Want avoid common **pain-points**:
 - Identifying all models that solve a given task
 - Routine operations requiring a lot of code
 - Passage from data source to algorithm-specific data format

Use any tabular data format



Scientific types



Categorical data

categorical \neq integer

Categorical data

categorical \neq integer

data = [1, 2, 2, 2, 1, 2, 1, 1, 3, 2]

train = [1, 2, 2, 2, 1] eval = [1, 3, 2]

Categorical data

categorical \neq integer

data = [1, 2, 2, 2, 1, 2, 1, 1, 3, 2]

train = [1, 2, 2, 2, 1] eval = [1, 3, 2]

MLJ expects `CategoricalArray.CategoricalValue` for categoricals.

Goals for MLJ

- Want **usability**, interoperability, extensibility and reproducibility
- Want avoid common **pain-points**:
 - Identifying all models that solve a given task
 - Routine operations requiring a lot of code
 - Passage from data source to algorithm-specific data format

Goals for MLJ

- Want **usability**, interoperability, extensibility and reproducibility
- Want avoid common **pain-points**:
 - Identifying all models that solve a given task
 - Routine operations requiring a lot of code
 - Passage from data source to algorithm-specific data format
 - Probabilistic predictions (evaluation, inconsistent representations, ...)

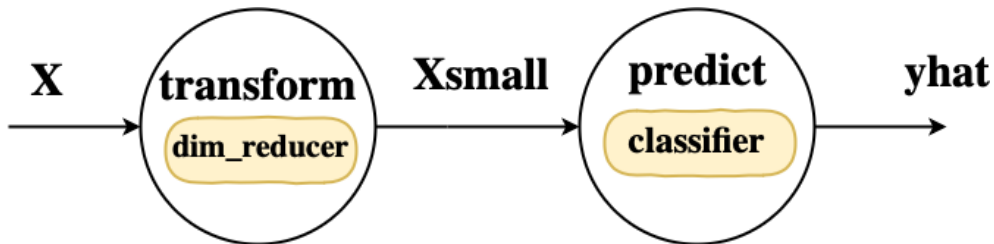
Goals for MLJ

- Want **usability**, interoperability, extensibility and reproducibility
- Want avoid common **pain-points**:
 - Identifying all models that solve a given task
 - Routine operations requiring a lot of code
 - Passage from data source to algorithm-specific data format
 - Probabilistic predictions: evaluation, inconsistent representations
 - Limitations of **model composition** API

Goals for MLJ

- Want **usability**, interoperability, extensibility and reproducibility
- Want avoid common **pain-points**:
 - Identifying all models that solve a given task
 - Routine operations requiring a lot of code
 - Passage from data source to algorithm-specific data format
 - Probabilistic predictions: evaluation, inconsistent representations
 - Limitations of **model composition** API — barrier to innovation!

Model composition (aka pipelining)



Data science competitions (kaggle)

IEEE-CIS Fraud Detection
Can you detect fraud from customer transactions?

IEEE Computational Intelligence Society 720 teams · 2 months to go (15 months to go until event deadline)

Overview Data Kernels Discussions **Leaderboard** Rules [Join Competition](#)

Public Leaderboard Private Leaderboard

This leaderboard is calculated with approximately 20% of the test data.
The final results will be based on the other 80%, so the final standings may be different.

[Raw Data](#) [Refresh](#)

■ In the money ■ Gold ■ Silver ■ Bronze

#	Team Name	Kernel	Team Members	Score @	Entries	Last
1	MingLuo			0.9482	5	2h
2	Michael Jahner			0.9478	10	5h
3	TUUV			0.9466	20	1h
4	[oda.ai] SinisterThree			0.9463	19	6h
5	Joko Pedro Peinado			0.9460	12	1d
6	3 LLamas			0.9455	10	6h
7	AL			0.9433	11	1d
8	THLUD			0.9432	7	7h
9	Aleksandr Keeslopov			0.9431	20	4h
10	Li-Der			0.9431	19	1h
11	Raghavendra Singh			0.9430	17	2h
12	ClaytonM			0.9429	3	1d
13	Team Data			0.9428	15	2h
14	Patrick Chan			0.9427	11	2h
15	less			0.9425	11	4h
16	AndreaToscher			0.9424	10	10h

Recursion Cellular Image Classification
Get Signal: Disentangling biological signal from experimental noise in cellular images

Recursion Pharmaceuticals 459 teams · 2 months to go (13 days to go until margin deadline)

Overview Data Kernels Discussion **Leaderboard** Rules [Join Competition](#)

Public Leaderboard Private Leaderboard

This leaderboard is calculated with approximately 22% of the test data.
The final results will be based on the other 78%, so the final standings may be different.

[Raw Data](#) [Refresh](#)

■ In the money ■ Gold ■ Silver ■ Bronze

#	Team Name	Kernel	Team Members	Score @	Entries	Last
1	[oda.ai] OndreaLentini			0.884	41	1d
2	gold diggers			0.804	81	10h
3	yu4u			0.690	13	16m
4	[attention heads] v-shmyklo			0.684	61	1d
5	David			0.672	34	5h
6	tascj			0.668	9	6d
7	napichai			0.634	60	3h
8	Double strand			0.620	38	10h
9	mandarinente			0.620	34	7h
10	Road to NeutIPS			0.616	46	2d
11	Kirill Brodskiy (jihad rak)			0.598	5	3d
12	-			0.596	25	10h
13	jianning			0.573	41	10h
14	MikhailPuplov			0.564	5	10m
15	Konstantin I. Novitskiy			0.490	16	1d

Jigsaw Unintended Bias in Toxicity Classification
Detect toxicity across a diverse range of conversations

Kaggle/Conversation AI 2,640 teams · 1 day ago

Overview Data Kernels Discussion **Leaderboard** Rules [Late Submission](#)

Public Leaderboard Private Leaderboard

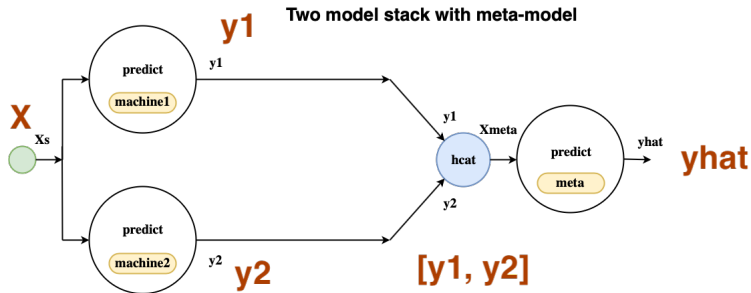
This is a Kaggle Competition with two stages. The public leaderboard represents scores on the stage 1 test set. Your final private leaderboard score and ranking will be determined in stage 2, when selected kernels are re-run on a withheld private test set. For more information, review the details provided on the Description page.
This competition has completed. This leaderboard reflects the final standings.

[Refresh](#)

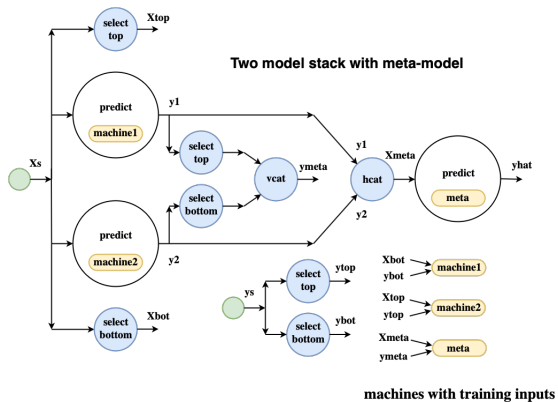
■ In the money ■ Gold ■ Silver ■ Bronze

#	Team Name	Kernel	Team Members	Score @	Entries	Last
1	2622 [oda.ai] Toxicology			0.94734	2	1d
2	2449 Linerobot (RIG RIG)			0.94720	2	1d
3	2584 F.H.S.D.Y			0.94707	2	23d
4	2150 COMBAT WOMBAT			0.94706	2	1d
5	2624 veccor			0.94683	2	1d
6	2409 yunai r			0.94678	2	1d
7	2470 [DSI] [kaggle-jai] PPPAP			0.94660	2	1d
8	2298 Qishan Ha			0.94660	2	1d
9	2416 KazikKan			0.94650	2	1d
10	2399 Harness the beasts (胡图巴图)			0.94649	2	23d
11	2331 tess, helen			0.94635	2	1d
12	2056 AAA Team			0.94634	2	1d
13	2403 zhongtian			0.94633	2	1d

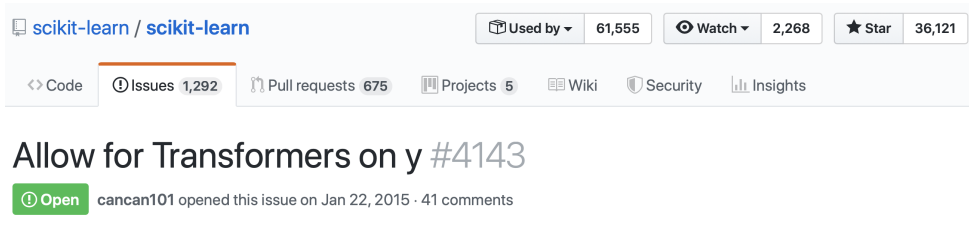
More complicated example



More complicated example



Target transformations



The screenshot shows the GitHub interface for the `scikit-learn` repository. The repository name is `scikit-learn / scikit-learn`. Statistics for the repository are displayed: 61,555 users used by, 2,268 watchers, and 36,121 stars. The navigation bar includes links for Code, Issues (1,292), Pull requests (675), Projects (5), Wiki, Security, and Insights. The 'Issues' tab is currently selected and highlighted with an orange bar. Below the navigation bar, the title of the selected issue is 'Allow for Transformers on y #4143'. A green button with an exclamation mark icon and the text 'Open' is next to the issue title. Below the button, the text indicates that the issue was opened by `cancan101` on Jan 22, 2015, and has 41 comments.

scikit-learn / scikit-learn

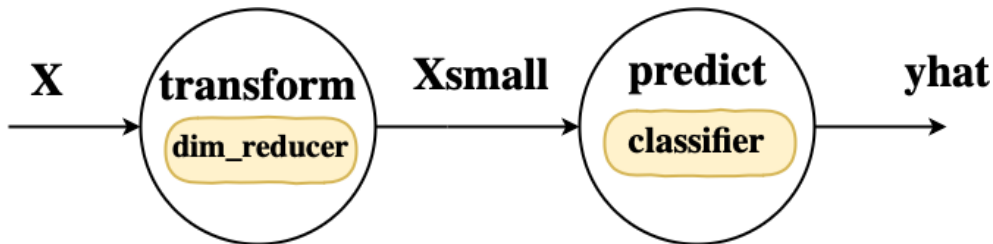
Used by 61,555 Watch 2,268 Star 36,121

<> Code Issues 1,292 Pull requests 675 Projects 5 Wiki Security Insights

Allow for Transformers on y #4143

Open `cancan101` opened this issue on Jan 22, 2015 · 41 comments

Model composition (aka pipelining)



Model composition (aka pipelining)



Compact syntax for linear pipeline

```
composite_ = @pipeline dim_reducer_ classifier_
```


Compact syntax for linear pipeline

```
composite_ = @pipeline dim_reducer_ classifier_
```

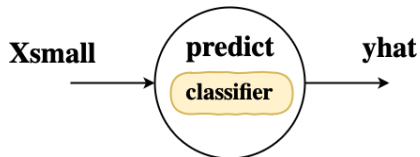
Does not generalize!

The dimension reducer



```
dim_reducer_ = PCA()  
dim_reducer = machine(dim_reducer_, X)  
fit!(dim_reducer)  
Xsmall = transform(dim_reducer, X);
```

The classifier



```
classifier_ = SVC()  
classifier = machine(classifier_, Xsmall, y)  
fit!(classifier)  
 $\hat{y}$  = predict(classifier, Xsmall)
```

Summary of unstreamlined workflow

```
dim_reducer_ = PCA()
dim_reducer = machine(dim_reducer_, X)
fit!(dim_reducer)
Xsmall = transform(dim_reducer, X);

classifier_ = SVC()
classifier = machine(classifier_, Xsmall, y)
fit!(classifier)
ŷ = predict(classifier, Xsmall)
```

Refactoring as **learning network**: Step 1

```
X = source(X)
y = source(y)

dim_reducer_ = PCA()
dim_reducer = machine(dim_reducer_, X)
fit!(dim_reducer)
Xsmall = transform(dim_reducer, X);

classifier_ = SVC()
classifier = machine(classifier_, Xsmall, y)
fit!(classifier)
ŷ = predict(classifier, Xsmall)
```

Refactoring as **learning network**: Step 2

```
X = source(X)
y = source(y)

dim_reducer_ = PCA()
dim_reducer = machine(dim_reducer_, X)
Xsmall = transform(dim_reducer, X);

classifier_ = SVC()
classifier = machine(classifier_, Xsmall, y)
ŷ = predict(classifier, Xsmall)
```

Training a learning network

```
X = source(X)
y = source(y)

dim_reducer_ = PCA()
dim_reducer = machine(dim_reducer_, X)
Xsmall = transform(dim_reducer, X);

classifier_ = SVC()
classifier = machine(classifier_, Xsmall, y)
ŷ = predict(classifier, Xsmall)

fit!(ŷ)
```

Prediction in a learning network

```
 $\hat{y}$ (rows=3:4)
```

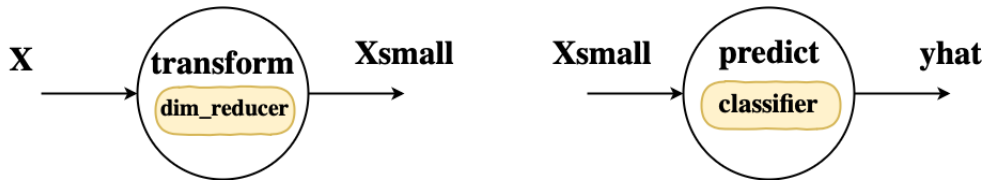
```
2-element Array{CategoricalString{UInt8},1}:  
  "versicolor"  
  "versicolor"
```


Prediction in a learning network

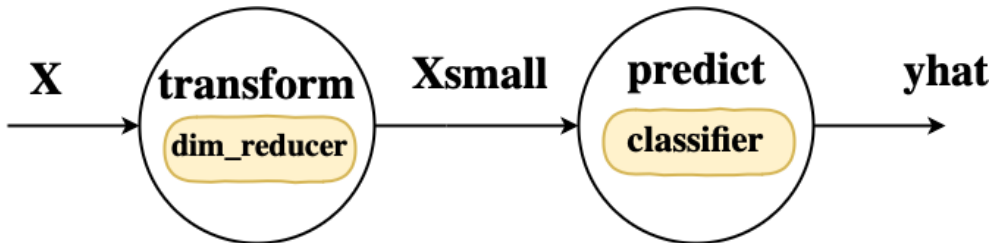
```
Xnew = (SepalLength = [4.0, 5.2],  
        SepalWidth = [3.2, 3.0],  
        PetalLength = [1.2, 1.5],  
        PetalWidth = [0.1, 0.4],)  
 $\hat{y}(X_{\text{new}})$ 
```

```
2-element Array{CategoricalString{UInt8},1}:  
 "setosa"  
 "setosa"
```

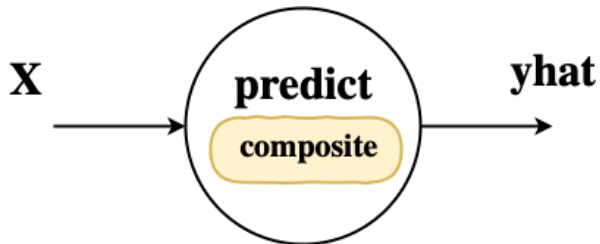
Summarizing



Summarizing



Still a need stand-alone model!



Macro to the rescue

```
composite = @from_network Composite(pca=dim_reducer_, svc=classifier_) <= (X, y,  $\hat{y}$ )
```

Composite is now a model like any other

```
composite_ = @from_network Composite(pca=dim_reducer_, svc=classifier_) <= (X, y,  $\hat{y}$ )

composite = machine(composite_, X2, y2)
fit!(composite)
predict(composite, Xnew)
```

Goals for MLJ

- Want **usability**, interoperability, extensibility and reproducibility
- Want avoid common **pain-points**:
 - Identifying all models that solve a given task
 - Routine operations requiring a lot of code
 - Passage from data source to algorithm-specific data format
 - Probabilistic predictions: evaluation, inconsistent representations
 - Limitations of **model composition** API

Goals for MLJ

- Want **usability**, interoperability, extensibility and reproducibility
- Want avoid common **pain-points**:
 - Identifying all models that solve a given task
 - Routine operations requiring a lot of code
 - Passage from data source to algorithm-specific data format
 - Probabilistic predictions: evaluation, inconsistent representations
 - Limitations of **model composition** API — barrier to innovation!

Goals for MLJ

- Want **usability**, interoperability, extensibility and reproducibility
- Want avoid common **pain-points**:
 - Identifying all models that solve a given task
 - Routine operations requiring a lot of code
 - Passage from data source to algorithm-specific data format
 - Probabilistic predictions: evaluation, inconsistent representations
 - Limitations of **model composition** API — barrier to innovation!
- Hope that project adds some focus to Julia ML development more generally

Road map

Road map

Enhancing functionality: Adding models

- Wrap the scit-learn (python/C) models (Z. Nugent, D. Arenas)
- **Flux.jl** deep learning (A. Shridhar)
- **Turing.jl** probabilistic programming (M. Trapp)
- **Geostats.jl** (J. Hoffmann)
- Data cleaning? Feature engineering (featuretools?)

Road map

Enhancing functionality: Adding models

- Wrap the scit-learn (python/C) models (Z. Nugent, D. Arenas)
- **Flux.jl** deep learning (A. Shridhar)
- **Turing.jl** probabilistic programming (M. Trapp)
- **Geostats.jl** (J. Hoffmann)
- Data cleaning? Feature engineering (featuretools?)

Road map

Enhancing core functionality

- Systematic benchmarking
- More comprehensive performance evaluation
- Tuning using Bayesian optimization
- Tuning using gradient descent and AD
- Iterative model control

Road map

Broadening scope

- Extend or supplement LossFunctions.jl
- Add sparse data support (NLP)
- Time series

Road map

Scalability

- Online learning support and distributed data
- DAG scheduling (J. Samaroo)
- Automated estimates of cpu/memory requirements

github.com/alan-turing-institute/**MLJ.jl**

Resources for this talk: examples/JuliaCon2019/

Core design: Anthony Blaom, Franz Kiraly, Sebastian Vollmer

Lead contributor: Anthony Blaom

Julia language consultants: Mike Innes, Avik Sengupta

Other contributors, past and present: Dilum Aluthge, Diego Arenas, Edoardo Barp, Gergő Bohner, Michael K. Borregaard, Valentin Churavy, Harvey

Devereux, Mosè Giordano, Thibaut Lienart, Mohammed Nook, Piotr Oleśkiewicz, Julian Samaroo, Ayush Shridar, Yiannis Simillides, Annika Stechemesser

turing.ac.uk
@turinginst