Toolbox for Machine Learning in julia

Edoardo Barp, Gergö Bohner, Valentin Churvay, Harvey Devereux, Thibaut Lienart, Franz J Király, Mohammed Nook, Annika Stechemesser, Sebastian Vollmer; Mike Innes in partnership with Julia Computing

KEY FEATURES (under development)

UNIFIED MODELLING INTERFACE DESIGN
access to a wide class of models, unified syntax

MODEL TUNING, PIPELINING and COMPOSITION
model abstracted tuning & composition interface

MODEL VALIDATION and MODEL EVALUATION
automated user workflows, benchmarking

MLJ leveraging Julia for fast, efficient integration

JULIA ECOSYSTEM: STATUS QUO

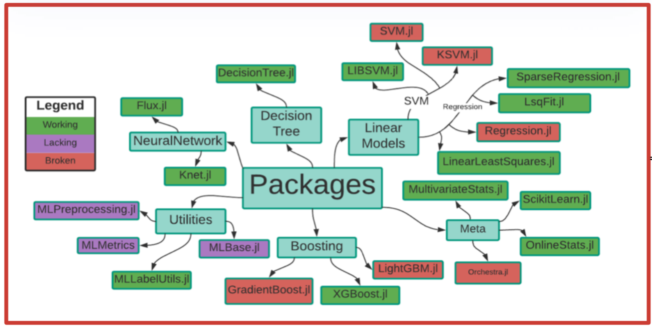


Figure: review of extant machine learning functionality and meta-functionality
"Meta" category = existing ML toolbox meta-packages in Julia

MACHINE LEARNING TOOLBOXES

Figure: logos of (the most?) popular open source machine learning toolboxes.
Left-to-right: Weka, python/sklearn, R/caret, R/mlr, Shogun

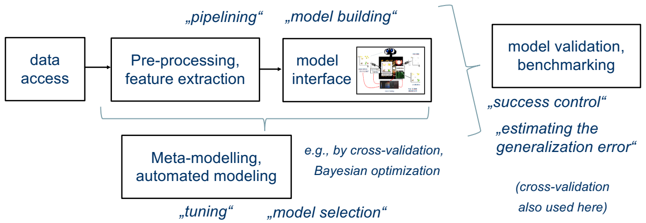
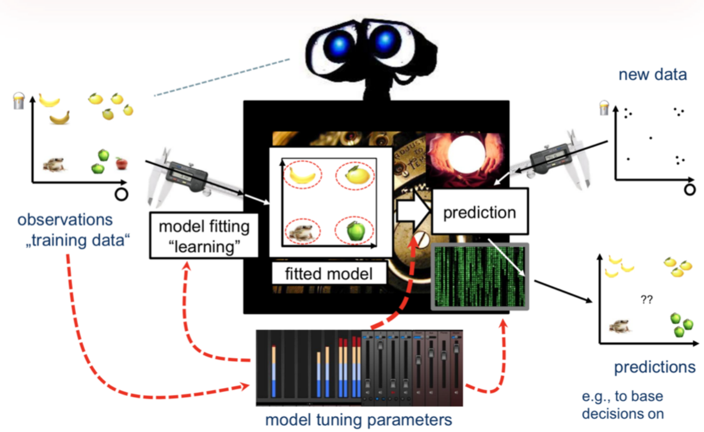
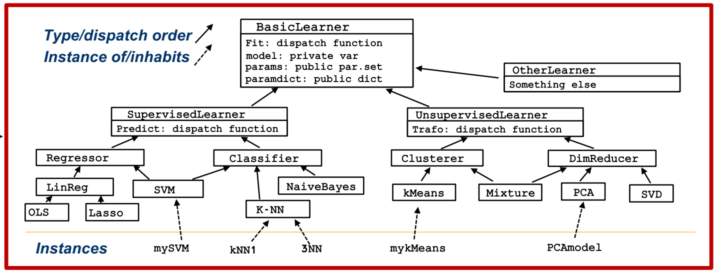


Figure: machine learning toolbox consensus workflow = abstraction layers

INTERFACE: ABSTRACTION & DISPATCH





Top: the "learner/estimator" abstraction, for supervised learning.
All strategies must implement fit, predict, model, parameter interfaces
Bottom: stylized type schema, dispatch functions, and type dispatch order
for modelling strategies implemented or interfaced by mlj

USER INTERACTION & SYNTAX: MLR INSPIRED

```
# Simple model fitting
task = Task(task_type="classification",
targets=[:y], data=data)
lrn = RandomForest( Dict("nsubfeatures"=>2,
"ntrees"=>10))
lrn_res = fit!(lrn, task)
predict!(lrn_res,df_new)
```

```
# Simple benchmarking - rapid trialing
lrns = [ModelLearner(:LDA),
ModelLearner(:RandomForest))
tasks = list(IrisTask, SonarTask)
resampling = CV(k=5)
measures = list(Acc(), MMCE())
bmr = benchmark(lrns, tasks, rdesc, measures =
measures)
```

```
# Tuning an SVM learner over a parameter grid
ps = ParametersSet([
    ContinuousParameter(name = "cost",lower =
-4,
        upper = 1,transform = x->10^x
    ),
    DiscreteParameter( name = "svmtype",
values = [SVC()]),
    DiscreteParameter(name = "kernel", values
= [Kernel.Poly]),
    ContinuousParameter(
        name = "coef0",lower = -4,upper =
1,transform = x->10^x
    )]])
svm = libsvmModel() # we pick a learner
```

```
svm_tuned = GridTunedModel(svm, ps ,CV(k=5),
MMCE())
 # userfrien dly
 listLearners(task) # all applicable
 Learner
 listMetrics(tasks) # all applicable
 Metrics
```

CORE API DESIGN & ABSTRACTIONS

TASK

```
immutable Task{T}
    _type::T
    targets::Symbol
    features::Array{Symbol}
    data::DataFrame
end

immutable RegressionTask end
immutable ClassificationTask end

abstract type BaseModel end
abstract type BaseModelFit{T<:BaseModel} end
```

MODEL

```
# model fitting returns modelFit struct type encoding fitted
instance
struct ModelFit{T} <: BaseModelFit{T}
    model :: T
    fit_result
end
model(modelFit::ModelFit) = modelFit.model # Accessor
function, infers type
predict(modelFit::BaseModelFit, Xnew) =
predict(model(modelFit), modelFit, Xnew)
```

TUNING

```
#  to add tuning to the model, this should result in a
composite that inherits
#  the initial model, as well as adds a tuning method, metric
and resampling
struct TunedModel{T<:BaseModel} <: BaseModel
    model :: T
    metric
    resampling
    tuning :: BaseTuning
end
# Accessor functions (for compile-time lookup gain)
model(tunedModel::TunedModel) = tunedModel.model
tuning(tunedModel::TunedModel) = tunedModel.tuning
# skeleton of tuning
function simple_tuning(model::BaseModel,
tuning::SimpleGridTuning,task::Task)
    tuning_result = [fit(typeof(model)(parameters), X, y) for
parameters in tuning.grid]
end
struct TunedModelFit{T} <: BaseModelFit{T}
    model :: T
    fit_result
    tuning :: BaseTuning
    tuning_result
end
```

NEXT STEPS: JOIN THE PROJECT!

We are looking for collaborators @ the Alan Turing Institute!

- **Finalising API design and user interaction patterns!**
- **Backend improvement! (Scheduling, Dagger, JuliaDB, Queryverse)**
- **Store learner meta info in METADATA.JL fashion (ideally open.ml compatable)**
- **Feature Improvement**
  - **Bootstrapping from Sklearn and mlr by wrapping with task info**
  - **Pipelining an composition meta-interface**
  - **Implementation of unsupported learners, e.g., deep learning**

channel #mlj on julialang.slack.com

julia@turing.ac.uk

https://github.com/alan-turing-institute/mlj