

You like scikit-learn? You like Stan? You love scikit-stan!

Alexey Izmailov^{1,2}

Mentored by Brian Ward²

Affiliation 1: Department of Applied Mathematics, Brown University, Providence RI

Affiliation 2: Center for Computational Mathematics, Flatiron Institute, New York NY

Abstract

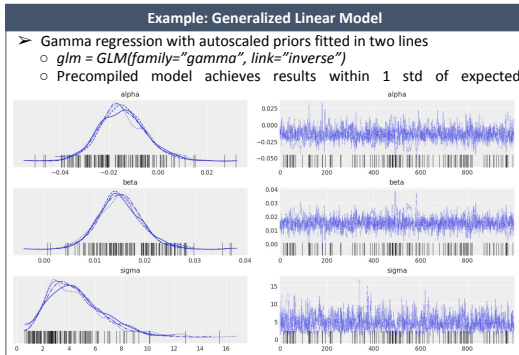
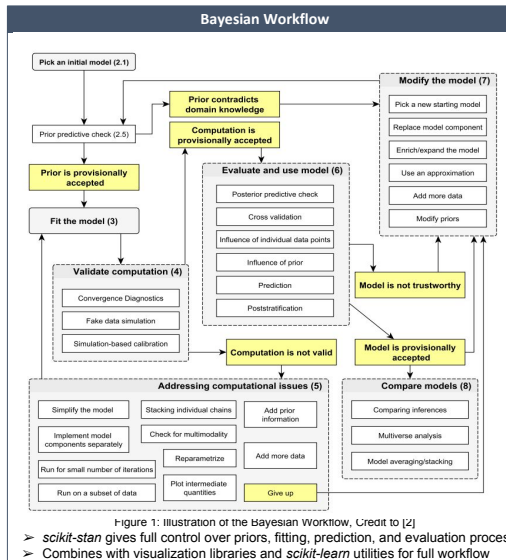
scikit-stan is a Python library of pre-compiled Bayesian models that adheres to the *scikit-learn* model philosophy and workflow. As such, this package provides a familiar API for fitting models, generating predictions, and scoring outcomes via a robust Stan backend. These design choices ensure that efficient probabilistic models are seamlessly integrated with the vast *scikit-learn* ecosystem while improving Stan's accessibility and outreach.

Introduction

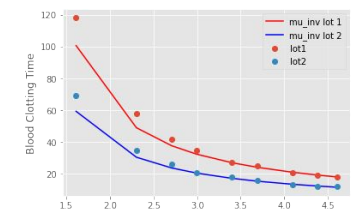
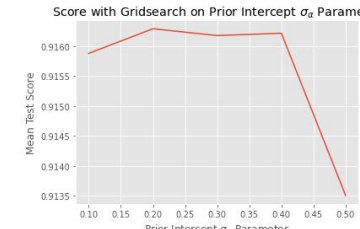
- Since its 2012 release, Stan has demonstrated novel state-of-the-art algorithms with top, cutting-edge performance for Bayesian methods and garnered over 100k users as of June 2022
 - Wrappers *RStan*, and *CmdStanPy* have introduced Bayesian methods to programming communities alongside libraries of models such as *rstanarm*, *brms*, and industrial packages like Facebook's *Prophet*
- scikit-learn* is a classic Python library with an elegant API and off-the-shelf models embedded in a mature ecosystem of modular operations and natural compositions
- This promising project improves Stan's accessibility via a familiar Python style and reduces required devtime by many provided true pre-compiled Stan models as a component crucial for the Bayesian workflow

scikit-learn API Matching

- Stan is versatile and supports several inference algorithms:
 - NUTS-HMC for sampling the posterior
 - L-BFGS for performing an MLE (point estimate) of model parameters
 - ADVI for variational inference of the posterior
- scikit-stan* models can perform any of the above inference methods to perform the role of a *scikit-learn* Estimator:
 - Estimator initialization and learning are separated [1] between object instantiation and `fit(Xtrain, ytrain)`
 - Extend to a *predictor* with a `predict(Xtest)` method to generate quantities with a fitted model based on new data
 - Inspect performance with `.score(Xtest, ytest)`
- One-to-one matching of *scikit-learn* class methods and functionality - models satisfy their respective validation suites



Example Applications

- Fitting results of package Gamma GLM on 9 data points from blood clotting data [4]
 
- Integrates with *scikit-learn* optimization, such as GridSearchCV
 - Performs hyperprior optimization out of the box:
 - Gridsearch on σ_{α} , the intercept prior's error scale in $y = \alpha_j + \beta x, \alpha_j \sim N(\mu_{\alpha}, \sigma_{\alpha}^2), j = 1, 2, \dots, 85$
- Score with Gridsearch on Prior Intercept σ_{α} Parameter
 
- Optimize over Radon household data from [5] to brute force $\sigma_{\alpha} \in [0.21, 0.38]$

References

- [1] *API design for machine learning software: experiences from the scikit-learn project*, L. Buitinck et al., 2013
- [2] *Bayesian workflow*, Gelman et al., 2020
- [3] *Stan: A Probabilistic Programming Language*, Carpenter et al., 2017
- [4] *Generalized Linear Models* - McCullagh & Nelder, 1989
- [5] *A Primer on Bayesian Multilevel Modeling using PyStan* - Fonnesbeck

Acknowledgements

This work was graciously supported by the Simons Foundation. The package was only possible with Brian Ward's fantastic mentoring and patience. Thank you for all the good you've taught me!

Gratitude to Bob Carpenter for his astute insights and memorable mentorship.

