The Journal of Open Source Software

Human-Learn: Human Benchmarks in a Scikit-Learn Compatible API

Vincent D. Warmerdam[1]

1 Personal

DOI: 10.21105/joss.03448

Software

- Review
- Repository
- Archive

Editor: Mehmet Hakan Satman

Reviewers:

- @desilinguist
- @ahurriyetoglu

Submitted: 03 May 2021
Published: 27 July 2021

Summary

This package contains scikit-learn compatible tools that make it easier to construct and benchmark rule-based systems designed by humans. There are tools to turn Python functions into scikit-learn compatible components and interactive jupyter widgets that allow the user to draw models. One can also use it to design rules on top of existing models that, for example, can trigger a classifier fallback when outliers are detected.

Statement of need

There's been a transition from rule-based systems to ones that use machine learning. Initially, systems converted data to labels by applying rules.

Figure 1: Rule Based Systems.

Recently, it has become much more fashionable to take data with labels and to use machine-learning algorithms to figure out appropriate rules.

Figure 2: Machine Learning Based Systems.

We started wondering if we might have lost something in this transition. Machine learning is a general tool, but it is capable of making bad decisions. Decisions that are very hard to debug too. Tools like SHAP (Lundberg & Lee, 2017) and LIME (Ribeiro et al., 2016) try to explain why algorithms make certain decisions in hindsight, but even with the benefit of hindsight, it is tough to understand what is happening.

At the same time, it is also true that many classification problems can be done by natural intelligence. This package aims to make it easier to turn the act of exploratory data analysis into a well-understood model. These "human" models are very explainable from the start. If nothing else, they can serve as a simple benchmark representing domain knowledge which is a great starting point for any predictive project.

Features

Human-learn can be installed via pip.

```
pip install human-learn
```

The library features components to easily turn Python functions into scikit-learn compatible components (Buitinck et al., 2013).

```python
import numpy as np
from hulearn.classification import FunctionClassifier

def fare_based(dataf, threshold=10):
    """
    The assumption is that folks who paid more are wealthier and are more
    likely to have recieved access to lifeboats.
    """
    return np.array(dataf['fare'] > threshold).astype(int)

# The function is now turned into a scikit-learn compatible classifier.
mod = FunctionClassifier(fare_based)
```

Besides the `FunctionClassifier`, the library also features a `FunctionRegressor` and a `FunctionOutlierDetector`. These can all take a function and turn the keyword parameters into grid-searchable parameters.

```python
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import *

# The GridSearch object can now "grid-search" over this model.
grid = GridSearchCV(mod,
                    cv=2,
                    param_grid={'threshold': np.linspace(0, 100, 30)},
                    scoring={'accuracy': make_scorer(accuracy_score),
                             'precision': make_scorer(precision_score),
                             'recall': make_scorer(recall_score)},
                    refit='accuracy')
grid.fit(X, y)
```

These function-based models can be very powerful because they allow the user the define rules for situations for which there is no data available. In the case of financial fraud, if a child has above median income, this should trigger risk. Machine learning models cannot learn if there is no data but rules can be defined even if, in this case, a child with above median income doesn't appear in the training data. An ideal use-case for this library is to combine rule based systems with machine learning based systems. An example of this is shown in Figure 3.

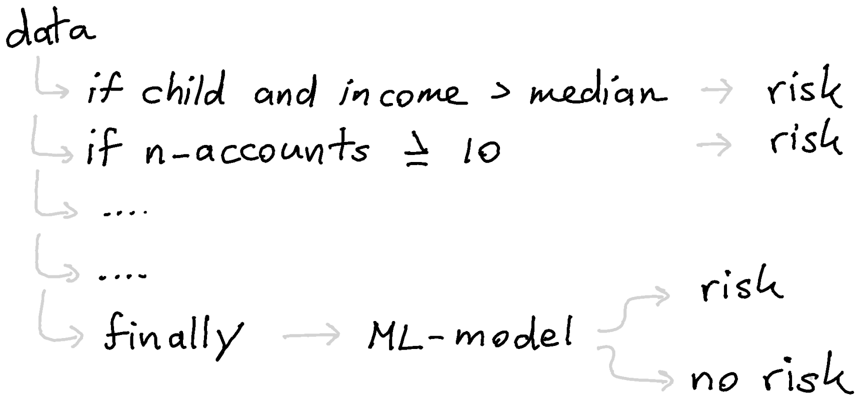

Figure 3: A rule based systems that resorts to ML when rules don't cover the example.

This example also demonstrates the main difference between this library and Snokel (Ratner et al., 2017). This library offers methods to turn domain knowledge immediately into models, as opposed to labelling-functions.

Human-learn also hosts interactive widgets, made with Bokeh, that might help construct models from Jupyter as well. An example of a drawn widget is shown below in figure Figure 4.

```
from hulearn.experimental.interactive import InteractiveCharts

df = load_penguins()
clf = InteractiveCharts(df, labels="species")

# It is best to add charts in their own seperate notebook cells
clf.add_chart(x="bill_length_mm", y="bill_depth_mm")
```

This interface allows the user to draw machine learning models. They can be used for classification, outlier detection, labeling tasks, or general data exploration. The snippet below demonstrates how to define a classifier based on the drawings.

```
from hulearn.classification import InteractiveClassifier

# This classifier uses a point-in-poly method to convert the drawn
# data from `clf` into a scikit-learn classifier.
model = InteractiveClassifier(json_desc=clf.data())
```

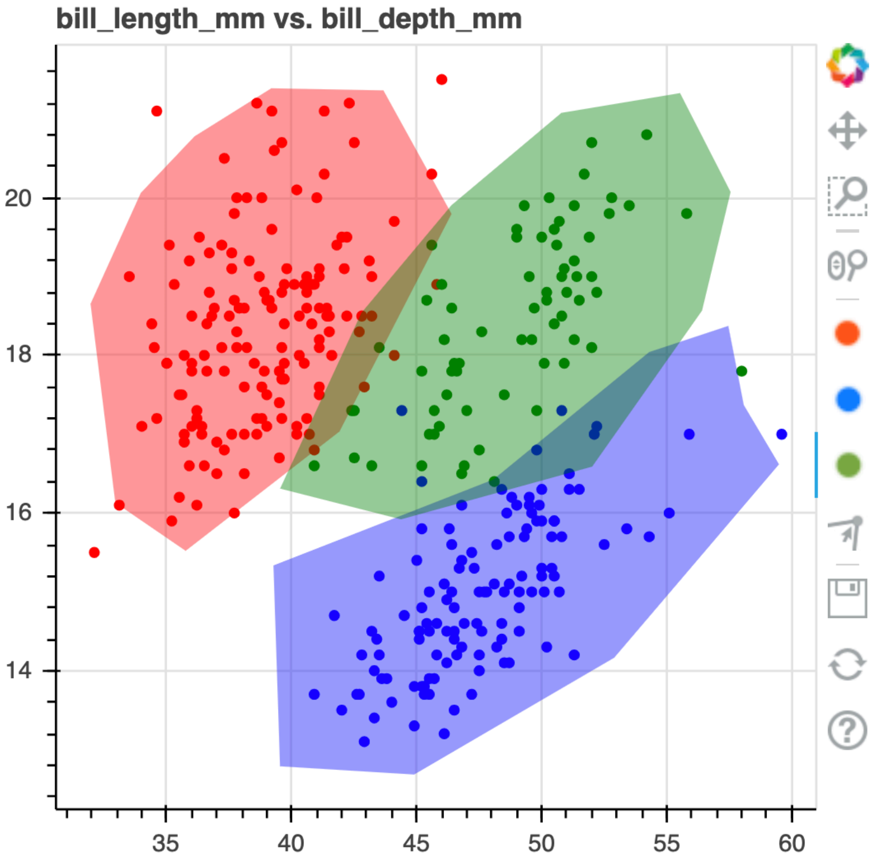

Figure 4: A screenshot of the drawing widget.

Acknowledgements

This project was developed in my spare time while being employed at Rasa. They have been very supportive of me working on my own projects on the side, and I would like to recognize them for being a great employer.

I also want to acknowledge that I'm building on the shoulders of giants. The popular drawing widget in this library would not have been possible without the wider Bokeh (Bokeh Development Team, 2018), Jupyter (Kluyver et al., 2016) and scikit-learn (Pedregosa et al., 2011) communities.

There have also been small contributions on Github from Joshua Adelman, Kay Hoogland, and Gabriel Luiz Freitas Almeida.

References

Bokeh Development Team. (2018). *Bokeh: Python library for interactive visualization*. https://bokeh.pydata.org/en/latest/

Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., VanderPlas, J., Joly, A., Holt, B.,

& Varoquaux, G. (2013). API design for machine learning software: Experiences from the scikit-learn project. *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 108–122.

Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J., Grout, J., Corlay, S., Ivanov, P., Avila, D., Abdalla, S., Willing, C., & team, J. development. (2016). Jupyter notebooks - a publishing format for reproducible computational workflows. In F. Loizides & B. Scmidt (Eds.), *Positioning and power in academic publishing: Players, agents and agendas* (pp. 87–90). IOS Press. https://eprints.soton.ac.uk/403913/

Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in neural information processing systems 30* (pp. 4765–4774). Curran Associates, Inc. http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Ratner, A., Bach, S. H., Ehrenberg, H., Fries, J., Wu, S., & Ré, C. (2017). Snorkel. *Proceedings of the VLDB Endowment*, *11*(3), 269–282. https://doi.org/10.14778/3157794.3157797

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "why should I trust you?": Explaining the predictions of any classifier. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, 1135–1144. https://doi.org/10.18653/v1/n16-3020