DALEXtra: Cross language model comparison

28 February 2020

Introduction

We have come to the point where machine learning predictive models usage is wider than even researchers of the past could expect. Business (Leo, Sharma, and Maddulety 2019), health care and many others relay to some extend on constantly changing models. Unfortunately constantly new appliances and need for improvement lead to more and more complicated black-boxes. That is why we seek for tools dedicated to Explainable Artificial Intelligence (XAI) that will help us to understand predictions. Good examples of such software are `DALEX` (Biecek 2018) R package or `lime` (Ribeiro, Singh, and Guestrin 2016) and `shap` (Lundberg and Lee 2017) Python libraries.

Comparison of Machine Learning models is tedious and not always a well-defined task. It is because there are many ways where such analysis could go. On top of that, fast development conducted in many directions, that were determined by variety of used programming language, made it hard to compare the behavior of models that was build using different environments. We can of course compare measurements but it is next to impossible to compare explanations. Therefore we need a unified way to analyze profiles of variables in our model (`ingredients`(Biecek et al. 2019)), the relative importance of features `lime` (Ribeiro, Singh, and Guestrin 2016) or residuals of model (`auditor`(Gosiewska and Biecek 2018)). All of mentioned above technics are a legitimate approach that can, but not always will, let us determine which model is the best.

The `DALEXtra` serves as an extension for `DALEX`(Biecek 2018) R package. One of its applications is to provide dedicated API that wraps models created using various Machine Learning libraries in order to explain them using DrWhy.ai(Przemyslaw Biecek 2018) family. That is necessary to perform Champion-Challenger analysis that will be covered in future paragraphs.

Comparison of explanations

The growing popularity of machine learning has sped up software development in that area. Developers took as their objective to make training models faster and smoother. R libraries `mlr`(Bischl et al. 2016) and `caret`(Jed Wing et al. 2019), Python `scikit-learn`(Pedregosa et al. 2011), or Java `h2o`(team 2015) made today's machine learning experts life easier than ever before. It is no longer hard to quickly make a decent model, even without sophisticated knowledge about machine learning. What is challenging nowadays is to understand what stands behind the model's decisions. This is exactly the motivation for the development of Explainable Artificial Intelligence (XAI) tools. Unfortunately due to huge diversity among frameworks used to train specific models it is hard to compare explanations of two models. Let's take for example model created using mlr and scikit-learn wrappers. Although they can use the same training algorithm (eg. randomForest), the software makes the difference. Various tools for explanations support only specific types of frameworks, and because of that it is impossible to smoothly compare every two models using them. It requires rearranging models, making them behave like the ones that are supported by the given XAI tool. Such a task requires time and what is worse, specific knowledge about how those tools work, what makes data scientist job harder. That is the motivation why explainable machine learning libraries should support a variety of frameworks on their own.

	mlr (R)	mlr3 (R)	parsnip (R)	caret (R)	mlr (Python)	keras (Python)	scikit-learn (Python)	h2o (Java)
DALEXtra	x	x	x	x	DODAM	x	x	x
interpret	-	-	-	-	-	-	x	-
fscaret	-	-	-	x	-	-	-	-
iml	x	x	-	x	-	x	-	x

	mlr (R)	mlr3 (R)	parsnip (R)	caret (R)	mlr (Python)	keras (Python)	scikit-learn (Python)	h2o (Java)
eli5	-	-	-	-	-	x	x	-
lime (R)	x	x	x	x	-	x	-	x
lime (Python)	-	-	-	-	x	x	x	x
shap	-	-	-	-	-	x	x	-
Skater								

Funnel Plot

The Funnel Plot is a new way to present various metrics scores of predictive model. It stands as a contrast to a global approach where we calculate measurements and plot them together. Instead of that, we do calculate metrics on subsets of our dataset, that are determined by variable distribution. Every numerics variable is being divided into bins, which number is stated by the user, according to empirical distribution. For categorical values, each level more frequent than cutoff will determine a different subset. Calculations can be made with any type of measurement function with a property that lower score indicates better performance (such as MSE or 1-AUC).

Code example

```
library("mlr")
library("DALEXtra")
task <- mlr::makeRegrTask(
  id = "R",
  data = apartments,
  target = "m2.price"
)
learner_lm <- mlr::makeLearner("regr.lm")
model_lm <- mlr::train(learner_lm, task)
explainer_lm <- explain_mlr(model_lm, apartmentsTest, apartmentsTest$m2.price,
                            label = "LM")
learner_rf <- mlr::makeLearner("regr.randomForest" )
model_rf <- mlr::train(learner_rf, task)
explainer_rf <- explain_mlr(model_rf, apartmentsTest, apartmentsTest$m2.price,
                            label = "RF")
funnel_plot_data <- funnel_measure(explainer_lm, explainer_rf,
                            nbins = 5,
                            measure_function = DALEX::loss_root_mean_square)
plot(funnel_plot_data)
```

Champion-challenger

Champion-challenger analysis is a prediction oriented way to present how a given model (Champion) behaves in comparison to other (possibly many Challengers). DALEXtra package allows users to create a report in an automatics way using three different already implemented sections and one default.

- **Funnel Plot** described in the previous paragraph.
- **Overall comparison** section that aims to visualize the global behavior of our model taking into consideration whole dataset. In fact, it consists of two sections

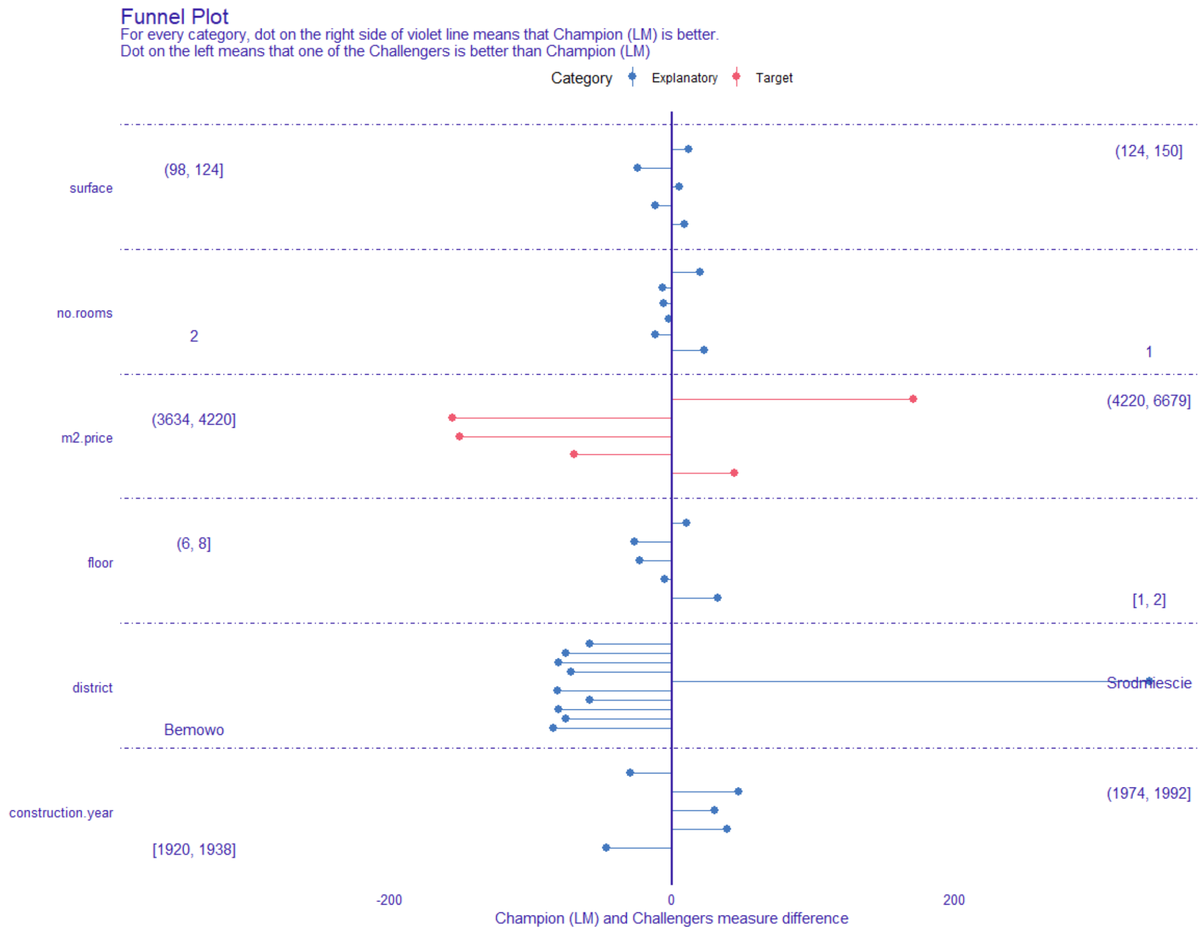

Figure 1: Funnel Plot. For every variable, dot on the right side of the violet line means that Linear Model (Champion model) is better for given subset. Dot on the left means that Random Forest model (Challenger model) is better. Values one the edges of the plot indicates subset that caused maximum deviation in measure difference for each side.

- **Radar Plot** Plots scores scaled to [0,1] compartment using radar type geometry. It represents a global approach where we are taking into consideration scores over the whole datasets. (Gosiewska and Biecek 2018)
- **Accordance Plot** for each observation it plots Champion response at OX axis and challengers responses at OY axis possibly using colors to distinguish models

- **Training-test comparison** plot presents the relation between a score that models achieved using test dataset and training dataset. Champion and Challengers are distinguished using different colors. It helps to prevent over-fitting
- **Default section** besides implemented sections we can pass any other object on which `plot` method can be used. (eg. `iBreakDown`(Gosiewska and Biecek 2019). It will be included in the report as an independent section.

Integration

R and Python frameworks for machine learning are developing rapidly. Therefore it happens that machine learning experts seek to cross them together in order to get the best model. That is the motivation for the second pillar of the DALEXtra package, availability to import a Python machine learning model and explain it using DALEX tools or do whatever the user intends to. Thanks to `reticulate`(Ushey, Allaire, and Tang 2019) package, DALEXtra can get predictions out of Python model and import them into R. All that functionalities are encapsulated in one function `explain_scikit` (accordingly `explain_keras`) where we have to just pass a pickle file with saved Python model and, if it is necessary, a .yml file that specifies an Anaconda virtual environment. A function will also extract a set of hyperparameters and save in an `explainer` object.

Code example

Please keep in mind that `system.file` function extracts data form package files.

```
explainer <- explain_scikitlearn(
system.file("extdata", "scikitlearn.pkl", package = "DALEXtra"),
yml = system.file("extdata", "testing_environment.yml", package = "DALEXtra"),
data = titanic_test[,1:17], y = titanic_test$survived)
```

Conclusions

The `DALEXtra` package is easy and intuitive to compare models even if they were built using various environments. It also provides dedicated wrappers that allow users to explain and therefore understand their models. All the above enhance makes machine learning experts' life easier. A comprehensive guide to how to use DALEXtra with python can be found in the vignette and on GitHub.

Acknowledgments

Work on this package was financially supported by the 'NCN Opus grant 2016/21/B/ST6/02176'.

References

Biecek, Przemyslaw, Hubert Baniecki, Adam Izdebski, and Katarzyna Pekala. 2019. *Ingredients: Effects and Importances of Model Ingredients.* http://CRAN.R-project.org/package=ingredients.

Biecek, Przemysław. 2018. "DALEX: explainers for complex predictive models." http://arxiv.org/abs/1806.08915.

Bischl, Bernd, Michel Lang, Lars Kotthoff, Julia Schiffner, Jakob Richter, Erich Studerus, Giuseppe Casalicchio, and Zachary M. Jones. 2016. "mlr: Machine Learning in R." *Journal of Machine Learning Research* 17 (170): 1–5. http://jmlr.org/papers/v17/15-066.html.

Gosiewska, Alicja, and Przemyslaw Biecek. 2019. "IBreakDown: Uncertainty of Model Explanations for Non-Additive Predictive Models." *arXiv Preprint arXiv:1903.11420*.

Gosiewska, Alicja, and Przemysław Biecek. 2018. "auditor: An R Package for Model-Agnostic Visual Validation and Diagnostic." *ArXiv E-Prints*. http://adsabs.harvard.edu/abs/2018arXiv180907763G.

Jed Wing, Max Kuhn. Contributions from, Steve Weston, Andre Williams, Chris Keefer, Allan Engelhardt, Tony Cooper, Zachary Mayer, et al. 2019. *Caret: Classification and Regression Training*. https://CRAN.R-project.org/package=caret.

Leo, Martin, Suneel Sharma, and K. Maddulety. 2019. "Machine Learning in Banking Risk Management: A Literature Review." *Risks* 7 (March): 29. https://doi.org/10.3390/risks7010029.

Lundberg, Scott M, and Su-In Lee. 2017. "A Unified Approach to Interpreting Model Predictions." In *Advances in Neural Information Processing Systems 30*, edited by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, 4765–74. Curran Associates, Inc. http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, et al. 2011. "Scikit-Learn: Machine Learning in Python." *Journal of Machine Learning Research* 12: 2825–30.

Przemyslaw Biecek, MI2DataLab. 2018. *Collection of Tools for Visual Exploration, Explanation and Debugging of Predictive Models*. drwhy.ai.

Ribeiro, Marco Tulio, Sameer Singh, and Carlos Guestrin. 2016. ""Why Should I Trust You?": Explaining the Predictions of Any Classifier." In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, ca, Usa, August 13-17, 2016*, 1135–44. https://doi.org/10.18653/v1/n16-3020.

team, The H2O.ai. 2015. *H2O: Scalable Machine Learning*. http://www.h2o.ai.

Ushey, Kevin, JJ Allaire, and Yuan Tang. 2019. *Reticulate: Interface to 'Python'*. https://CRAN.R-project.org/package=reticulate.