In applied machine learning, there are opinions that we need to choose between interpretability and accuracy. However in field of the Interpretable Machine Learning, there are more and more new ideas for explaining black-box models. One of the best known method for local explanations is SHapley Additive exPlanations (SHAP).

The SHAP method is used to calculate influences of variables on the particular observation. This method is based on Shapley values, a technique borrowed from the game theory. SHAP was introduced by Scott M. Lundberg and Su-In Lee in [A Unified Approach to Interpreting Model Predictions](http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions) NIPS paper. Originally it was implemented in the Python library [shap](https://github.com/slundberg/shap).

The R package [shapper](https://cran.r-project.org/package=shapper) is a port of the Python library shap. In this post we show the functionalities of shapper. The examples are provided on *titanic\_train* data set for classification.

While *shapper* is a port for Python library *shap*, there are also pure R implementations of the SHAP method, e.g. [iml](https://github.com/christophM/iml) or [shapleyR](https://github.com/redichh/ShapleyR).

**Installation**

The *shapper* wraps up the Python library, therefore installation requires a bit more effort than installation of an ordinary R package.

**Install the R package shapper**

First of all we need to install *shapper*, this may be the stable release from CRAN

install.packages("shapper")

or the developer version form GitHub.

devtools::install\_github("ModelOriented/shapper")

**Install the Python library shap**

Before you run *shapper*, make sure that you have installed Python.

Python library *shap* is required to use *shapper*. The *shap* can be installed both by Python or R. To install it through R, you an use function *install\_shap()* from the *shapper* package.

library("shapper")

install\_shap()

If you experience any problems related to the installation of Python libraries or evaluation of Python code, see the [reticulate](https://rstudio.github.io/reticulate/) documentation. The *shapper* access Python within *reticulate*, therefore the solution to the problem is likely to be in there ;-).

**Would you survive sinking of the RMS Titanic?**

The example usage is presented on the *titanic\_train* dataset from the R package [titanic](https://cran.r-project.org/package=titanic). We will predict the *Survived* status. The other variables used by the model are: *Pclass, Sex, Age, SibSp, Parch, Fare* and *Embarked*.

library("titanic")

titanic <- titanic\_train[,c("Survived", "Pclass", "Sex", "Age", "SibSp", "Parch", "Fare", "Embarked")]

titanic$Survived <- factor(titanic$Survived)

titanic$Sex <- factor(titanic$Sex)

titanic$Embarked <- factor(titanic$Embarked)

titanic <- na.omit(titanic)

head(titanic)

## Survived Pclass Sex Age SibSp Parch Fare Embarked

## 1 0 3 male 22 1 0 7.2500 S

## 2 1 1 female 38 1 0 71.2833 C

## 3 1 3 female 26 0 0 7.9250 S

## 4 1 1 female 35 1 0 53.1000 S

## 5 0 3 male 35 0 0 8.0500 S

## 7 0 1 male 54 0 0 51.8625 S

**Let’s build a model**

Let’s see what are our chances assessed by the random forest model.

library("randomForest")

set.seed(123)

model\_rf <- randomForest(Survived ~ . , data = titanic)

model\_rf

##

## Call:

## randomForest(formula = Survived ~ ., data = titanic)

## Type of random forest: classification

## Number of trees: 500

## No. of variables tried at each split: 2

##

## OOB estimate of error rate: 18.63%

## Confusion matrix:

## 0 1 class.error

## 0 384 40 0.09433962

## 1 93 197 0.32068966

**Prediction to be explained**

Let’s assume that we want to explain the prediction of a particular observation (male, 8 years old, traveling 1-st class embarked at C, without parents and siblings.

new\_passanger <- data.frame(

Pclass = 1,

Sex = factor("male", levels = c("female", "male")),

Age = 8,

SibSp = 0,

Parch = 0,

Fare = 72,

Embarked = factor("C", levels = c("","C","Q","S"))

)

Model prediction for this observation is .558 for survival.

predict(model\_rf, new\_passanger, type = "prob")

## 0 1

## 1 0.442 0.558

## attr(,"class")

## [1] "matrix" "votes"

**Here shapper starts**

To use the function *shap()* function (alias for *individual\_variable\_effect()*) we need four elements

* a model,
* a data set,
* a function that calculated scores (predict function),
* an instance (or instances) to be explained.

The *shap()* function can be used directly with these four arguments, but for the simplicity here we are using the [DALEX](https://github.com/pbiecek/DALEX/) package with preimplemented *predict* functions.

library("DALEX")

exp\_rf <- explain(model\_rf, data = titanic[,-1])

The explainer is an object that wraps up a model and meta-data. Meta data consists of, at least, the data set used to fit model and observations to explain.

And now it’s enough to generate SHAP attributions with explainer for RF model.

library("shapper")

ive\_rf <- shap(exp\_rf, new\_observation = new\_passanger)

ive\_rf

## Pclass Sex Age SibSp Parch Fare Embarked \_id\_ \_ylevel\_ \_yhat\_

## 1 1 male 8 0 0 72 C 1 0.558

## 1.1 1 male 8 0 0 72 C 1 0.558

## 1.2 1 male 8 0 0 72 C 1 0.558

## 1.3 1 male 8 0 0 72 C 1 0.558

## 1.4 1 male 8 0 0 72 C 1 0.558

## 1.5 1 male 8 0 0 72 C 1 0.558

## \_yhat\_mean\_ \_vname\_ \_attribution\_ \_sign\_ \_label\_

## 1 0.3672941 Pclass 0.070047752 + randomForest

## 1.1 0.3672941 Sex -0.154519708 - randomForest

## 1.2 0.3672941 Age 0.143046212 + randomForest

## 1.3 0.3672941 SibSp 0.003154522 + randomForest

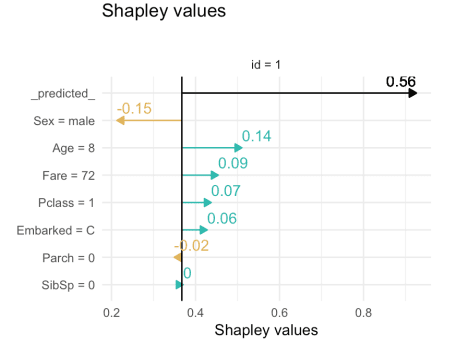
## 1.4 0.3672941 Parch -0.018111585 - randomForest

## 1.5 0.3672941 Fare 0.086728705 + randomForest

**Plotting results**

To generate a plot of Shapley values you can simply pass an object of class importance\_variable\_effect to a plot() function. Since we are interested in the class Survived = 1 we may add additional parameter that filter only selected classes.

plot(ive\_rf)



Labels on y-axis show values of variables for this particular observation. Black arrows show predictions of model, in this case, probabilities of each status. Other arrows show effect of each variable on this prediction. Effects may be positive or negative and they sum up to the value of prediction.

On this plot we can see that model predicts that the passenger will survive. Changes are higher due to young age and 1st class, only the Sex = male decreases chances of survival for this observation.

**More models**

It is useful to contrast prediction of two models. Here we will show how to use shapper for such contrastive explanations.

We will compare randomForest with svm implemented in the e1071.

library("e1071")

model\_svm <- svm(Survived~. , data = titanic, probability = TRUE)

model\_svm

##

## Call:

## svm(formula = Survived ~ ., data = titanic, probability = TRUE)

##

##

## Parameters:

## SVM-Type: C-classification

## SVM-Kernel: radial

## cost: 1

## gamma: 0.1

##

## Number of Support Vectors: 338

This model predict 32.5% chances of survival.

attr(predict(model\_svm, newdata = new\_passanger, probability = TRUE), "probabilities")

## 0 1

## 1 0.6748768 0.3251232

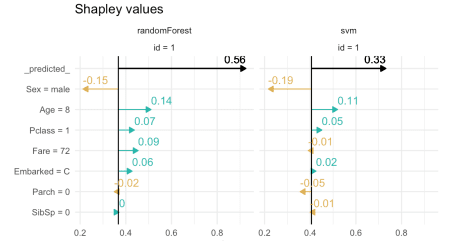
Again, we create an explainer that wraps model, data and the predict function.

exp\_svm <- explain(model\_svm, data = titanic[,-1])

ive\_svm <- shap(exp\_svm, new\_passanger)

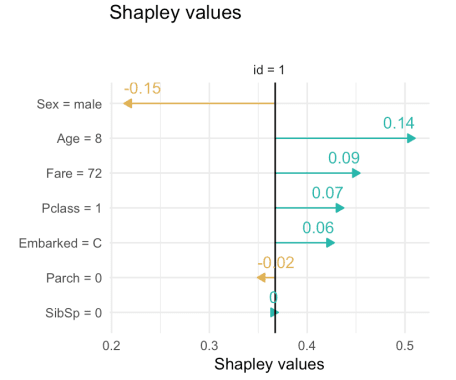
Shapley values plot may be modified. To show more than one model you can pass more individual\_variable\_plot objects.

plot(ive\_rf, ive\_svm)



To see only attributions use option *show\_predcited = FALSE*.

plot(ive\_rf, show\_predcited = FALSE)



**More**

Documentation and more examples are available at <https://modeloriented.github.io/shapper/>. The stable version of the package is on CRAN, the development version is on GitHub (<https://github.com/ModelOriented/shapper>). Shapper is a part of the [DALEX](https://github.com/pbiecek/DALEX/) universe.