

This file contains a brief tutorial on how to install and use the ONAM package.

Install package:

```
install.packages("devtools")  
library(devtools)  
devtools::install_github("Koehlibert/ONAM")
```

If this is the first time using Keras or TensorFlow in R, you need to run `keras::install_keras()`.

The following packages are required for importing the data, machine learning and visualization:

```
library(ONAM)  
library(mlbench)  
library(MASS)  
library(gbm)  
library(e1071)  
library(caret)  
library(xgboost)  
library(dplyr)
```

We demonstrate the use of the proposed algorithm by decomposing (i) the results of a gradient boosting machine for the prediction of diabetes risk and (ii) the results of an XGBoost model for the prediction of housing prices.

Gradient boosting machine for the prediction of diabetes risk using the Pima Indians Diabetes Database

Load the data:

```
data(PimaIndiansDiabetes)
diabetes_data <- PimaIndiansDiabetes %>%
  mutate(diabetes = ifelse(diabetes == "pos", 1, 0)) %>%
  mutate(across(!diabetes, as.numeric)) %>%
  filter(mass > 0)

head(diabetes_data)
```

##	pregnant	glucose	pressure	triceps	insulin	mass	pedigree	age	diabetes
## 1	6	148	72	35	0	33.6	0.627	50	1
## 2	1	85	66	29	0	26.6	0.351	31	0
## 3	8	183	64	0	0	23.3	0.672	32	1
## 4	1	89	66	23	94	28.1	0.167	21	0
## 5	0	137	40	35	168	43.1	2.288	33	1
## 6	5	116	74	0	0	25.6	0.201	30	0

The data consist of 757 observations of multiple medical features as well as the diabetes status.

There are 266 diabetes cases and 491 controls in the data.

We demonstrate how to use ONAM to explain a gradient boosting machine for the prediction of diabetes status.

First, we fit a gradient boosting machine for binary classification.

```
# Train the GBM model (binary classification)
gbm_model <- gbm(diabetes ~ .,
  data = diabetes_data,
  distribution = "bernoulli",
  n.trees = 500,
  interaction.depth = 3,
  shrinkage = 0.01,
  cv.folds = 5)
summary(gbm_model, plotit = F)
```

##		var	rel.inf
## glucose	glucose	44.687850	
## mass	mass	19.620205	
## age	age	12.942128	
## pedigree	pedigree	9.985040	
## pregnant	pregnant	4.078983	
## insulin	insulin	3.625236	
## pressure	pressure	3.326441	
## triceps	triceps	1.734118	

Assessment of model performance:

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##           0 468 126
##           1  23 140
##
##           Accuracy : 0.8032
##           95% CI : (0.773, 0.8309)
##           No Information Rate : 0.6486
##           P-Value [Acc > NIR] : < 2.2e-16
##
##           Kappa : 0.5262
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##           Sensitivity : 0.9532
##           Specificity : 0.5263
##           Pos Pred Value : 0.7879
##           Neg Pred Value : 0.8589
##           Prevalence : 0.6486
##           Detection Rate : 0.6182
##           Detection Prevalence : 0.7847
##           Balanced Accuracy : 0.7397
##
##           'Positive' Class : 0
##
```

Fit ONAM model to decompose gbm predictions

We illustrate the use of the proposed algorithm by decomposing the predictions of the fitted gbm model:

```
Diabetes_formula <- diabetes ~ mod(glucose) + mod(insulin) +  
  mod(pregnant) + mod(pressure) + mod(triceps) +  
  mod(pedigree) + mod(mass) + mod(age) +  
  mod(glucose, insulin) + mod(age, mass) + mod(.)  
list_of_mods_Diabetes <- list(mod = ONAM::get_submodel)  
gbm_expl <- onam(  
  Diabetes_formula,  
  list_of_deep_models_Diabetes,  
  diabetes_data,  
  gbm_model,  
  prediction_function = function(model, data) {  
    predict(model, data, n.trees = 500, type = "response")  
  },  
  target = "binary",  
  n_ensemble = 20,  
  progresstext = TRUE,  
  epochs = 500,  
  verbose = 0  
)
```

```
#generate and save predictions  
gbm_res <- predict(gbm_expl)  
saveRDS(gbm_res, "gbm_res.RDS")
```

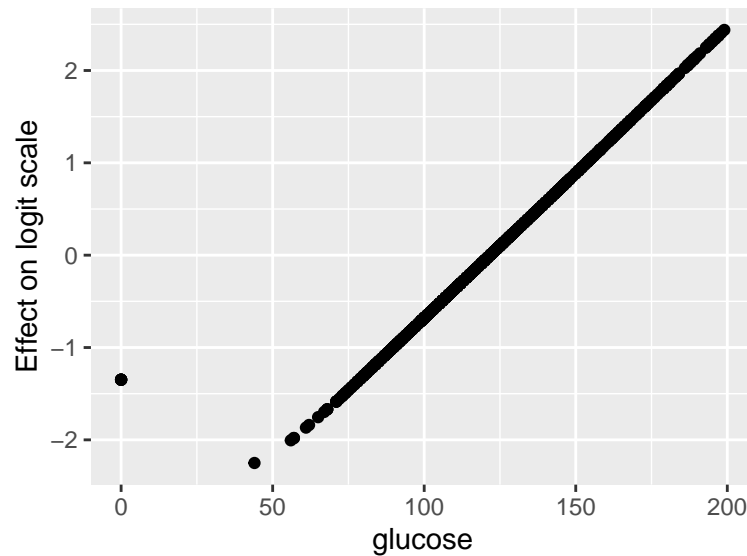
```
##  
## Call:  
## onam(formula = Diabetes_formula, list_of_deep_models = list_of_deep_models_Diabetes,  
##       data = diabetes_data, model = gbm_model, prediction_function = function(model,  
##       data) {  
##         predict(model, data, n.trees = 500, type = "response")  
##       }, target = "binary", n_ensemble = 10, epochs = 500, progresstext = TRUE,  
##       verbose = 0)  
##  
## Correlation of onam probabilities with original model predicted probabilities: 0.9889  
## Number of ensemble members: 10  
## I_1: 0.9726; I_2: 0.0101  
## Degree of interpretability: 0.9827
```

Visualization

All effects are presented on the logit scale.

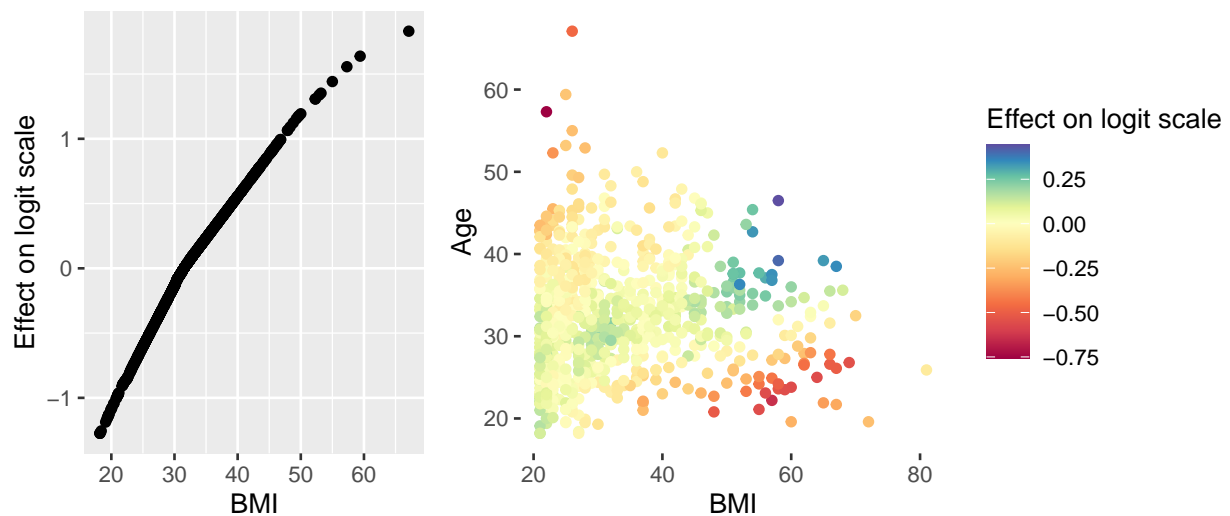
A higher plasma glucose concentration is associated with a higher risk of diabetes:

```
plot_main_effect(gbm_res, "glucose")
```



A higher BMI is associated with a higher risk of diabetes, but the increase is lower in people of younger age:

```
plot_main_effect(gbm_res, "mass") + xlab("BMI")  
  
plot_inter_effect(gbm_res, "age", "mass", interpolate = FALSE) +  
  xlab("BMI") + ylab("Age")
```



XGBoost model for the prediction of housing prices using the Boston Housing Data

Load the data:

```
data("Boston")
```

```
head(Boston)
```

```
##      crim zn  indus chas   nox    rm  age    dis rad tax ptratio  black lstat
## 1 0.00632 18   2.31    0 0.538 6.575 65.2 4.0900   1 296    15.3 396.90  4.98
## 2 0.02731  0   7.07    0 0.469 6.421 78.9 4.9671   2 242    17.8 396.90  9.14
## 3 0.02729  0   7.07    0 0.469 7.185 61.1 4.9671   2 242    17.8 392.83  4.03
## 4 0.03237  0   2.18    0 0.458 6.998 45.8 6.0622   3 222    18.7 394.63  2.94
## 5 0.06905  0   2.18    0 0.458 7.147 54.2 6.0622   3 222    18.7 396.90  5.33
## 6 0.02985  0   2.18    0 0.458 6.430 58.7 6.0622   3 222    18.7 394.12  5.21
##   medv
## 1 24.0
## 2 21.6
## 3 34.7
## 4 33.4
## 5 36.2
## 6 28.7
```

The dataset contains information on 506 census tracts. We use XGBoost to fit a prediction model for median housing price per census tract.

```
train_Boston <- Boston %>% dplyr::select(-medv)
label_Boston <- Boston %>% dplyr::select(medv) %>% unlist()

xgb_train_Boston <- xgb.DMatrix(as.matrix(train_Boston),
                                label = label_Boston)

xgb_mod <-
  xgboost(
    xgb_train_Boston,
    nrounds = 10000,
    verbose = 0,
    objective = "reg:squarederror",
    eval_metric = "rmse"
  )
summary(xgb_mod)
```

```
##           Length Class           Mode
## handle           1 xgb.Booster.handle externalptr
## raw             7318563 -none-      raw
## niter            1 -none-      numeric
## evaluation_log    2 data.table    list
## call             15 -none-      call
## params            3 -none-      list
## callbacks         1 -none-      list
## feature_names     13 -none-      character
## nfeatures         1 -none-      numeric
```

```
preds <- predict(xgb_mod, xgb_train_Boston)
pricing_cor <- cor(preds, Boston$medv)
```

Correlation between housing prices and predictions: 1

Fit ONAM model to decompose XGBoost predictions

We illustrate the use of the proposed algorithm by decomposing the predictions of the fitted XGBoost model:

```
Boston_formula <-
  medv ~ mod(crim) + mod(zn) + mod(indus) + mod(chas) +
  mod(nox) + mod(rm) + mod(age) + mod(dis) + mod(rad) +
  mod(tax) + mod(ptratio) + mod(black) + mod(lstat) +
  mod(crim, dis) + mod(rm, age) + mod(zn, indus) +
  mod(.)

list_of_deep_models_Boston <- list(mod = ONAM::get_submodel)

categorical_features <- c("chas")

xgb_expl <-
  onam(
    Boston_formula,
    list_of_deep_models_Boston,
    Boston,
    # target = "binary",
    model = xgb_mod,
    model_data = xgb_train_Boston,
    categorical_features = categorical_features,
    n_ensemble = 20,
    epochs = 1000,
    progresstext = TRUE,
    verbose = 0
  )

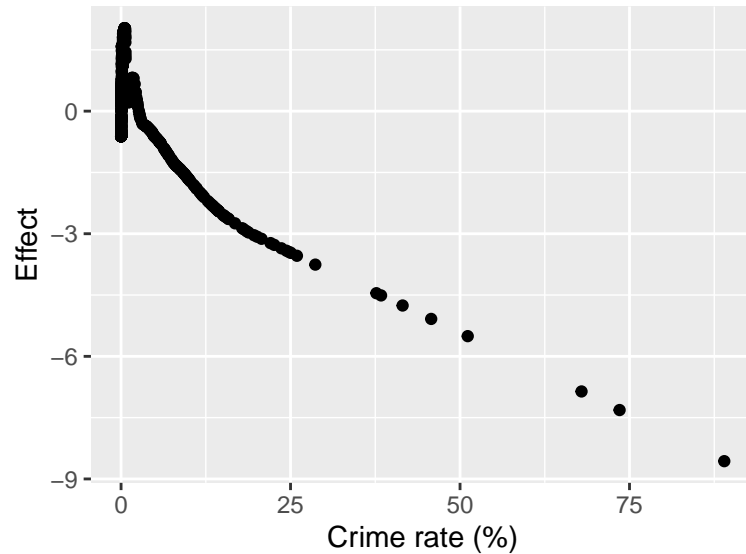
xgb_res <- predict(xgb_expl)
saveRDS(xgb_res, "xgb_res.RDS")
```

```
##
## Call:
## onam(formula = Boston_formula, list_of_deep_models = list_of_deep_models_Boston,
##       data = Boston, model = xgb_mod, model_data = xgb_train_Boston,
##       categorical_features = categorical_features, n_ensemble = 20,
##       epochs = 1000, progresstext = TRUE, verbose = 0)
##
## Correlation of model prediction with outcome variable: 0.9942
## Number of ensemble members: 20
## I_1: 0.924; I_2: 0.0415
## Degree of interpretability: 0.9656
```

Visualization

Higher crime rate is associated with lower housing prices:

```
plot_main_effect(xgb_res, "crim") + xlab("Crime rate (%)")
```



Lower number of rooms is associated with lower housing prices, but less so in (areas with) older houses:

```
plot_main_effect(xgb_res, "rm") + xlab("Av. number of rooms per dwelling")
plot_inter_effect(xgb_res, "rm", "age") +
  xlab("Av. number of rooms per dwelling") +
  ylab("% of units build prior to 1940")
```

