IFOA eXplainable AI Workstream - SHAP

1. Intro to case study

1.1 Motivation

The aim of this document is to supply the reader of name of the article in The Actuary Magazine, with a practical, comprehensive example of applying SHAP to explain models' predictions. K. Gawlowski, D. Liew and C. Richard.

1.2 Intro to dataset

The dataset can be downloaded from: freMTPL2freq

Originally it was accessible through CASdatasets R package. Excerpt from the original documentation:

Description

In the two datasets freMTPL2freq, freMTPL2sev, risk features are collected for 677,991 motor third-part liability policies (observed mostly on one year). In addition, we have claim numbers by policy as well as the corresponding claim amounts. freMTPL2freq contains the risk features and the claim number while freMTPL2sev contains the claim amount and the corresponding policy ID.

Format

IDpol The policy ID (used to link with the claims dataset).
ClaimNb Number of claims during the exposure period.
Exposure The period of exposure for a policy, in years.
VehPower The power of the car (ordered values).
VehAge The vehicle age, in years.
DrivAge The driver age, in years (in France, people can drive a car at 18).
BonusMalus Bonus/malus, between 50 and 350: <100 means bonus, >100 means malus in France.
VehBrand The car brand (unknown categories).
VehGas The car gas, Diesel or regular.
Area The density value of the city community where the car driver lives in: from "A" for rural area to "F" for urban centre.
Density The density of inhabitants (number of inhabitants per square-kilometer) of the city where the car driver lives in.
Region The policy region in France (based on the 1970-2015 classification)

1.3 Basic summary statistics

Table 1: Data summary

ClaimNb	Area	VehPower	VehAge	DrivAge	BonusMalus	VehBrand	VehGas	Density	Region
1	D	5	0	55	50	B12	Regular	7.104144	R82
1	D	5	0	55	50	B12	Regular	7.104144	R82
1	B	6	2	52	50	B12	Diesel	3.988984	R22
1	B	7	0	46	50	B12	Diesel	4.330733	R72
1	B	7	0	46	50	B12	Diesel	4.330733	R72
1	E	6	2	38	50	B12	Regular	8.007367	R31

Name	data
Number of rows	678013
Number of columns	10
_______________________	
Column type frequency:	
factor	5
numeric	5
________________________	
Group variables	None

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
Area	0	1	FALSE	6	C: 191880, D: 151596, E: 137167, A: 103957
VehPower	0	1	TRUE	12	6: 148976, 7: 145401, 5: 124821, 4: 115349
VehBrand	0	1	FALSE	11	B12: 166024, B1: 162736, B2: 159861, B3: 53395
VehGas	0	1	FALSE	2	Reg: 345877, Die: 332136
Region	0	1	FALSE	22	R24: 160601, R82: 84752, R93: 79315, R11: 69791

Variable type: numeric

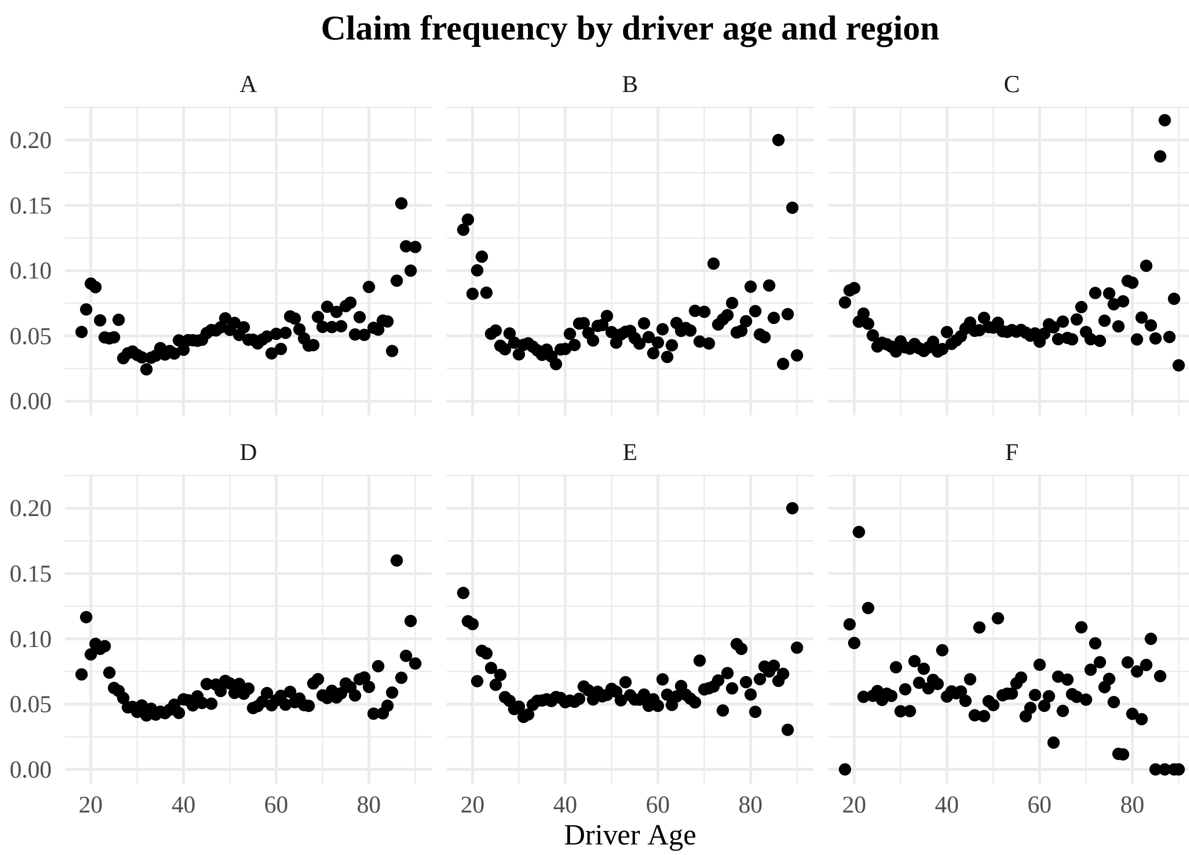
skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
ClaimNb	0	1	0.05	0.24	0	0.00	0.00	0.00	4.0	
VehAge	0	1	6.98	5.40	0	2.00	6.00	11.00	20.0	
DrivAge	0	1	45.50	14.13	18	34.00	44.00	55.00	90.0	
BonusMalus	0	1	59.76	15.64	50	50.00	50.00	64.00	230.0	
Density	0	1	5.98	1.87	0	4.52	5.97	7.41	10.2	

1.4 EDA

```r
# ```{r, echo=FALSE, message=FALSE, warning=FALSE}

data %>%
  group_by(DrivAge,Area) %>%
  summarise(Freq = mean(ClaimNb)) %>%
```

```
ggplot(aes(x = DrivAge, y=Freq))+
facet_wrap(~Area)+
geom_point()+
ggtitle("Claim frequency by driver age and region")+
xlab("Driver Age")+
ylab("")+
theme_minimal()+
theme(text=element_text(family="serif"), # Cambria Math
      plot.title = element_text(hjust = 0.5,face = "bold"))
```



2. models - fitting, output comparison

In this section we build the three models (NN, XGB, GLM) and assess their performance. The objects holding the models are pulled from the Prep_RMD folder for convenience. The code used to generate them however is supplied below in each respective section.

We examine the models' outputs using a hold out test dataset and applying a custom model_evalulation function. It lets us view: the obtained MSE or Poisson loss; prediction statistics grouped by the actual claim number and raw predictions. model_evalulation is standardized to accept all three of the models considered in the exercise. Further details along with the source code can be found in section 4.1

2.1 NN

Notes: Rebuild the NN and XGB models using the same validation split. Refine the XGB, Further tweak the NN Make the train/test split less awkward

```r
# notrun
if(FALSE){

  early_stop = callback_early_stopping(monitor = "val_loss", patience = 6)

  model1 = keras_model_sequential(input_shape = c(ncol(train))) %>%
    layer_dense(units = 64, activation = 'relu') %>%
    layer_dense(units = 64, activation = 'relu') %>%
    layer_dense(units = 64, activation = 'relu') %>%
    layer_dense(units = 1, activation = 'relu')

  model1 %>%
    compile(
      loss = "mse",
      # metrics = list("mse","poisson")
      optimizer = optimizer_rmsprop()
    ) %>%
    fit(
      callbacks = list(early_stop),
      train %>% as.matrix(),
      (data[sample_TTS,"ClaimNb"]) %>% as.matrix(),
      batch_size = 1024,
      epochs = 60,
      validation_split = 0.2,
      shuffle = TRUE ,
      sample_weight =  ((data[sample_TTS,"ClaimNb"]>0)*1+1) %>% as.matrix()
    )

}


# eval_list$model1 = model_evaluation(model1,
#                                       cbind(test,data[-sample_TTS,"ClaimNb"]),
#                                       type = "NN")

summary(model1)
```

```
## Model: "sequential"
## ________________________________________________________________________________
## Layer (type)                        Output Shape                    Param #
## ================================================================================
## dense_3 (Dense)                     (None, 64)                      3712
## ________________________________________________________________________________
## dense_2 (Dense)                     (None, 64)                      4160
## ________________________________________________________________________________
## dense_1 (Dense)                     (None, 64)                      4160
## ________________________________________________________________________________
## dense (Dense)                       (None, 1)                       65
## ================================================================================
```

```
## Total params: 12,097
## Trainable params: 12,097
## Non-trainable params: 0
## ________________________________________________________________________________
```

```
# eval_list$model1$Evaluation
```

Evaluation

```
# eval_list$model1$AvE
```

Actual vs Expected

2.2 XGB

```
# notrun
if(FALSE){

  XGB_validation = 1:10000

  dtrain = xgb.DMatrix(data = train[-XGB_validation,] %>% as.matrix(),
                       label = data[sample_TTS,]$ClaimNb[-XGB_validation])

  dvalid = xgb.DMatrix(data = train[XGB_validation,] %>% as.matrix(),
                       label = data[sample_TTS,]$ClaimNb[XGB_validation])

  watchlist <- list(train=dtrain, test=dvalid)

  m5_xgb <-
    xgboost(
      # data = data[sample_TTS,] %>% select(-ClaimNb) %>% ,
      # label = data[sample_TTS,]$ClaimNb ,

      # data = train[-XGB_validation,] %>% as.matrix() ,
      # label = data[sample_TTS,]$ClaimNb[-XGB_validation],
      data = dtrain,

      nrounds = 200,


      weight = (data[sample_TTS,"ClaimNb"]+1) %>% as.matrix(),
      objective = "reg:squarederror",
      early_stopping_rounds = 3,
      max_depth = 8,
      eta = .3,
```

```
      verbose = 2,
      watchlist = watchlist
    )

}
```

```
# eval_list$xgb5 = model_evalulation(model = m5_xgb,
#                                     data = data.frame(test,
#                                                       ClaimNb = data[-sample_TTS,"ClaimNb"]),
#                                     type = "XGB")
```

text

```
# eval_list$xgb5$Evaluation
```

text

```
# eval_list$xgb5$AvE
```

text

2.3 GLM

```
#
# eval_list$PoissonGLM = model_evalulation(model = PoissonGLM$model,
#                                          data = data[-sample_TTS,],
#                                          type = "GLM")
#
# summary(PoissonGLM)
```

text

```
# eval_list$PoissonGLM$Evaluation
```

text

```r
# eval_list$PoissonGLM$AvE
```

text

2.4 Model comparison

```r
# eval_list$PoissonGLM$AvE
```

3. XAI

3.1 Model - level SHAP

The graphs will be brought over as .png files.

```
if(FALSE){
  explainer = list()
  residuals = list()
  SHAP= list()

  predict_wrapper=function(model,new_data){
    return(model %>% predict(new_data %>% as.matrix()))
  }

  row_subset = sample(1:nrow(test),1000)

  explainer$model1 = explain(
    model = model1,
    data = test[row_subset,] %>% select(-ClaimNb),
    y = test[row_subset,]$ClaimNb,
    predict_function = predict_wrapper,
    label = "model1"
  )

  SHAP$model1 = shap(explainer$model1,
                     new_observation = test[1,-1],
                     method = "KernelSHAP"
  )

  colnames(SHAP$model1)
  p1 = plot(SHAP$model1,digits = 5,bar_width = 3)
}
```

3.2 Individual SHAP

4. Additional Information

4.1 model_evalulation utility function

Utility function for analysing model performance.

Input:
model: KERAS Neural Network, XGBoost, GLM
type: specify the type of model can be done automatically
data: named dataframe to assess models' performance on
ClaimNBadj: When passing a NN with poisson loss, an adjustment has to be made to the Claim Number

Output:
list(Evaluation = Evaluation, Predictions = Predictions, AvE = AvE, Sorted_by_MSE = Sorted_by_MSE))

```
model_evalulation = function(model,
                             type = "NN",
                             data = test,
                             ClaimNBadj = FALSE){
  if (type=="NN"){

    Evaluation = model %>% evaluate(data %>% select(-ClaimNb) %>% as.matrix() , data$ClaimNb)

    if(ClaimNBadj==TRUE){

      # rescaled Predictions (since min(ClaimNb)==0 then it's just times max(ClaimNb))
      Predictions = data.frame(Predicted = (model %>%
                                              predict(data %>%
                                                        select(-ClaimNb) %>%
                                                        as.matrix()) - 1),
                               Actual = data$ClaimNb) # *minmax$ClaimNb[1]
    }else{
      # rescaled Predictions (since min(ClaimNb)==0 then it's just times max(ClaimNb))
      Predictions = data.frame(Predicted = (model %>%
                                              predict(data %>%
                                                        select(-ClaimNb) %>%
                                                        as.matrix())),
                               Actual = data$ClaimNb) # *minmax$ClaimNb[1]
    }

  }else if(type=="GLM"){

    Predictions = data.frame(Predicted = exp(predict(model,
                                                     newdata = data %>%
                                                       select(-ClaimNb))),
                             Actual = data$ClaimNb)

    Evaluation = data.frame(#loss = NA,
      absolute_error = mean(abs(as.matrix(Predictions$Predicted - data$ClaimNb))),
      mean_squared_error = mean(as.matrix((Predictions$Predicted - data$ClaimNb)^2)))

  }

  AvE = Predictions %>% mutate(Actual = as.factor(Actual)) %>%
```

```r
    group_by(Actual) %>%
    summarise(count = n(),
              mean_pred = mean(Predicted),
              sd_pred = sd(Predicted),
              min = min(Predicted),
              max = max(Predicted),
              Q1 = quantile(Predicted,probs = 0.25),
              Q2 = quantile(Predicted,probs = 0.5),
              Q3 = quantile(Predicted,probs = 0.75),
              IQR = (Q3-Q1)/Q2,
              Negative_Pred = sum(Predicted<0))


  Sorted_by_MSE = data %>% mutate(Predictions = Predictions$Predicted,
                                  SquaredError = (Predictions - ClaimNb)^2) %>%
    arrange(-SquaredError)

  return(list(Evaluation = Evaluation,
              Predictions = Predictions,
              AvE = AvE,
              Sorted_by_MSE = Sorted_by_MSE))
}
```

4.2

Data wrangling was not the main intention of this exercise, thus the dataset in Prep_RMD/XAI_data.rda has been already processed. The script below takes the raw freMTPL data as in the source and transforms it to our needs.

```r
if (FALSE){

  data=read.csv2("freMTPL2freq.csv",sep = ",") %>%
    as_tibble() %>%
    mutate(VehPower  = factor(VehPower, order = TRUE,levels = 4:15),
           across(c(Area,Region,VehBrand,VehGas),factor)) %>%
    rowwise() %>%
    mutate(ClaimNb = as.integer(min(ClaimNb,4)),
           VehAge = as.integer(min(VehAge,20)),
           DrivAge = as.integer(min(DrivAge,90)),
           Exposure = as.numeric(min(Exposure,1)),
           Density = log(Density)) %>%
    ungroup() #%>%
  # mutate(ClaimNb = ClaimNb/Exposure) # not in Wutrich?

  # remove ID's
  data = data %>% select(-c(IDpol,Exposure))

  # save min and max of numerical columns
  # minmax = data.frame(max = apply(data %>%
  #                                 select_if(Negate(is.factor)),2,max),
  #                       min = apply(data %>%
  #                                 select_if(Negate(is.factor)),2,min)) %>%
  #   t() %>%
```

```
  #   as.data.frame()
  #
  # train/test split
  sample_TTS = sample(x = 1:nrow(data),
                      size = round(0.85 * nrow(data)),
                      replace = FALSE)

  train = data[sample_TTS,] %>% as.data.frame() #%>% as.matrix()
  test = data[-sample_TTS,] %>% as.data.frame() #%>% as.matrix()


  # Normalization of numeric variables
  data_prep = function(data){

    Normalized = apply(data %>% select(-ClaimNb) %>% select_if(Negate(is.factor)),
                       2,
                       FUN = function(x){return((x-min(x))/(max(x)-min(x)))}) %>%
      as_tibble()

    data = data %>% select_if(is.factor) %>% cbind(Normalized)

    data$VehPower = factor(data$VehPower, order = FALSE)

    # OHE of factor variables
    OHE = dummyVars("~.", data = data %>% select_if(is.factor)) %>%
      predict(newdata = data %>% select_if(is.factor)) %>%
      as_tibble()

    data = data %>% select_if(Negate(is.factor)) %>% cbind(OHE)

    data = as_tibble(data)

    return(data)
  }

  # data prep separately for train and test
  train = data_prep(train)
  test = data_prep(test)

}
```

4.3 Session information

Hardware, R and Python configuration

```
sessionInfo()
```

```
## R version 4.0.4 (2021-02-15)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 19044)
##
## Matrix products: default
```

```
##
## locale:
## [1] LC_COLLATE=English_United States.1252
## [2] LC_CTYPE=English_United States.1252
## [3] LC_MONETARY=English_United States.1252
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United States.1252
## system code page: 1250
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] xgboost_1.5.0.2  skimr_2.1.3      keras_2.7.0      tensorflow_2.7.0
##  [5] reticulate_1.18  shapper_0.1.3    DALEX_2.2.0      caret_6.0-86
##  [9] lattice_0.20-41  forcats_0.5.1    stringr_1.4.0    dplyr_1.0.7
## [13] purrr_0.3.4      readr_1.4.0      tidyr_1.1.4      tibble_3.1.6
## [17] ggplot2_3.3.5    tidyverse_1.3.1
##
## loaded via a namespace (and not attached):
##  [1] nlme_3.1-152        fs_1.5.0             lubridate_1.7.10
##  [4] httr_1.4.2          repr_1.1.3           tools_4.0.4
##  [7] backports_1.2.1     utf8_1.2.1           R6_2.5.0
## [10] rpart_4.1-15        DBI_1.1.1            colorspace_2.0-0
## [13] nnet_7.3-15         withr_2.4.1          tidyselect_1.1.0
## [16] compiler_4.0.4      cli_3.1.0            rvest_1.0.0
## [19] xml2_1.3.2          labeling_0.4.2       scales_1.1.1
## [22] tfruns_1.5.0        rappdirs_0.3.3       digest_0.6.27
## [25] rmarkdown_2.7       base64enc_0.1-3      pkgconfig_2.0.3
## [28] htmltools_0.5.1.1   highr_0.8            dbplyr_2.1.1
## [31] rlang_0.4.10        readxl_1.3.1         rstudioapi_0.13
## [34] farver_2.1.0        generics_0.1.0       jsonlite_1.7.2
## [37] ModelMetrics_1.2.2.2 magrittr_2.0.1      Matrix_1.3-2
## [40] Rcpp_1.0.6          munsell_0.5.0        fansi_0.4.2
## [43] lifecycle_1.0.0     stringi_1.5.3        whisker_0.4
## [46] pROC_1.17.0.1       yaml_2.2.1           MASS_7.3-53
## [49] plyr_1.8.6          recipes_0.1.17       grid_4.0.4
## [52] crayon_1.4.1        haven_2.3.1          splines_4.0.4
## [55] hms_1.0.0           zeallot_0.1.0        knitr_1.31
## [58] pillar_1.6.4        reshape2_1.4.4       codetools_0.2-18
## [61] stats4_4.0.4        reprex_2.0.0         glue_1.4.2
## [64] evaluate_0.14       data.table_1.14.0    modelr_0.1.8
## [67] vctrs_0.3.8         foreach_1.5.1        cellranger_1.1.0
## [70] gtable_0.3.0        assertthat_0.2.1     xfun_0.22
## [73] gower_0.2.2         prodlim_2019.11.13   broom_0.7.10
## [76] class_7.3-18        survival_3.2-7       timeDate_3043.102
## [79] iterators_1.0.13    lava_1.6.9           ellipsis_0.3.2
## [82] ipred_0.9-12
```

```
py_config()
```

```
## python:         C:/Users/gawlo/Documents/Py_projects/Anaconda/envs/TEST_ENV2/python.exe
## libpython:      C:/Users/gawlo/Documents/Py_projects/Anaconda/envs/TEST_ENV2/python38.dll
## pythonhome:     C:/Users/gawlo/Documents/Py_projects/Anaconda/envs/TEST_ENV2
```

```
## version:        3.8.5 (default, Sep  3 2020, 21:29:08) [MSC v.1916 64 bit (AMD64)]
## Architecture:   64bit
## numpy:          C:/Users/gawlo/Documents/Py_projects/Anaconda/envs/TEST_ENV2/Lib/site-packages/numpy
## numpy_version:  1.19.2
## tensorflow:     C:\Users\gawlo\DOCUME~1\PY_PRO~1\Anaconda\envs\TEST_E~1\lib\site-packages\tensorflow\
##
## NOTE: Python version was forced by use_python function
```