Efficient and simple prediction explanations with *groupShapley*

A practical perspective

Martin Jullum (jullum@nr.no)
Annabelle Redelmeier
Kjersti Aas

XAI.it workshop, AIxIA 2021
November 29th 2021

Prediction explanation

- Assume a model $f(\boldsymbol{x}) \in \mathbb{R}$ that predicts some unknown outcome based on a set of features $\boldsymbol{x} = (x_1, \ldots, x_M)$
- We apply the predictive model for a specific input $\boldsymbol{x} = \boldsymbol{x}^*$, reaching a certain prediction $f(\boldsymbol{x}^*)$
- Individual prediction explanation
  - Want to understand how the different features, or types of features affect this specific prediction value $f(\boldsymbol{x}^*)$
  - I.e. explain the predicted outcome in terms of the input $\boldsymbol{x} = \boldsymbol{x}^*$ (local explanation)

- Frameworks…
  - LIME
  - Anchors
  - Counterfactual explanations
  - Explanation Vectors
  - PredDiff
  - **Shapley values**

Shapley values

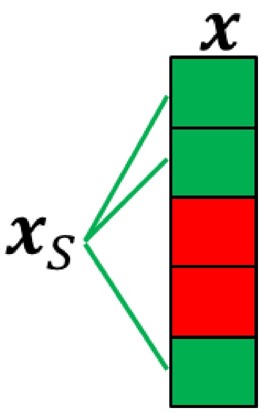
- General concept
  - Stems from cooperative game theory (Shapley, 1953)
  - Used to distribute the total payoff to the players
  - Explicit formula for the "fair" payment to every player j:

$$\phi_j = \sum_{\text{all } S \text{ without } j} w(|S|)\left(v(S \cup \{j\}) - v(S)\right),$$

w is a certain weight function,
$v(S)$ is the payoff with only players in subset S

  - Several mathematical optimality properties

- For prediction explanation
  - Players = features $(x_1, \dots, x_M)$
  - Payoff = prediction outcome $(f(\boldsymbol{x}^*))$
  - Contribution function: $v(S) = E[f(\boldsymbol{x})|\boldsymbol{x}_S = \boldsymbol{x}_S^*]$
  - Rough interpretation of ϕ_j
    - The prediction change caused by observing x_j



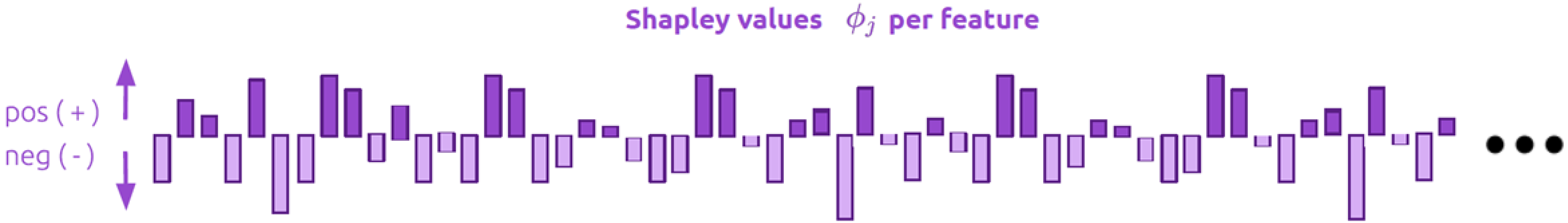
Bottlenecks

$$\phi_j = \sum_{\text{all } S \text{ without } j} w(|S|)\left(v(S \cup \{j\}) - v(S)\right)$$

$$\begin{aligned}
M = 5 &\Rightarrow 2^M = 32 \\
M = 10 &\Rightarrow 2^M = 1024 \\
M = 20 &\Rightarrow 2^M = 1048676 \\
M = 40 &\Rightarrow 2^M > 10^{12} \\
M = 100 &\Rightarrow 2^M > 10^{30} \\
M = 1000 &\Rightarrow 2^M > 10^{301}
\end{aligned}$$

1. The sum in the Shapley value formula is of size 2^M, growing exponentially in the number of features

2. How can we visualize, interpret and extract knowledge from 100s or 1000s of Shapley values?



- Typically: the sum of many small ϕ_j > sum of the few large ones
- Many highly dependent features complicates the interpretation

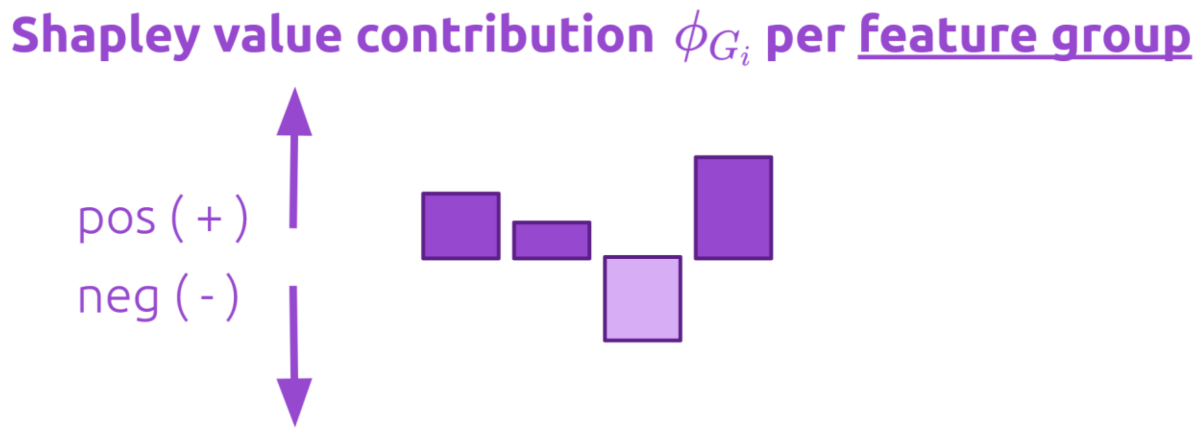
groupShapley

- Fundamentally very simple approach
  - Divide the M features into a small number of G disjoint groups $\{G_1, \ldots, G_G\}$.
  - Replace the feature subsets S in the Shapley formula by group subsets T:

$$\phi_{G_i} = \sum_{\text{all } T \text{ without } G_i} w(|T|)\left(v(T \cup G_i) - v(T)\right)$$

  - The scores are still Shapley values, so all mathematical properties are kept (on group level)

- What about the bottlenecks?
  - $2^G \ll 2^M \Rightarrow$ computationally tractable
  - G small $\Rightarrow$ easy to visualize



How to group the features?

- Crucial to group features based on the desired explanation
- Grouping based on feature dependence
  - Highly dependent features grouped together, using e.g. a clustering method.
  - Easier to study theoretically
  - Often difficult to extract knowledge from in practice
- Grouping based on application/feature knowledge
  - Group features of similar type or general category
  - Gives directly meaningful interpretations of computed groupShapley values
  - May perform multiple explanations with different groupings for increased understanding
- We advocate grouping based on feature knowledge in practical applications

Practical example 1: Car insurance

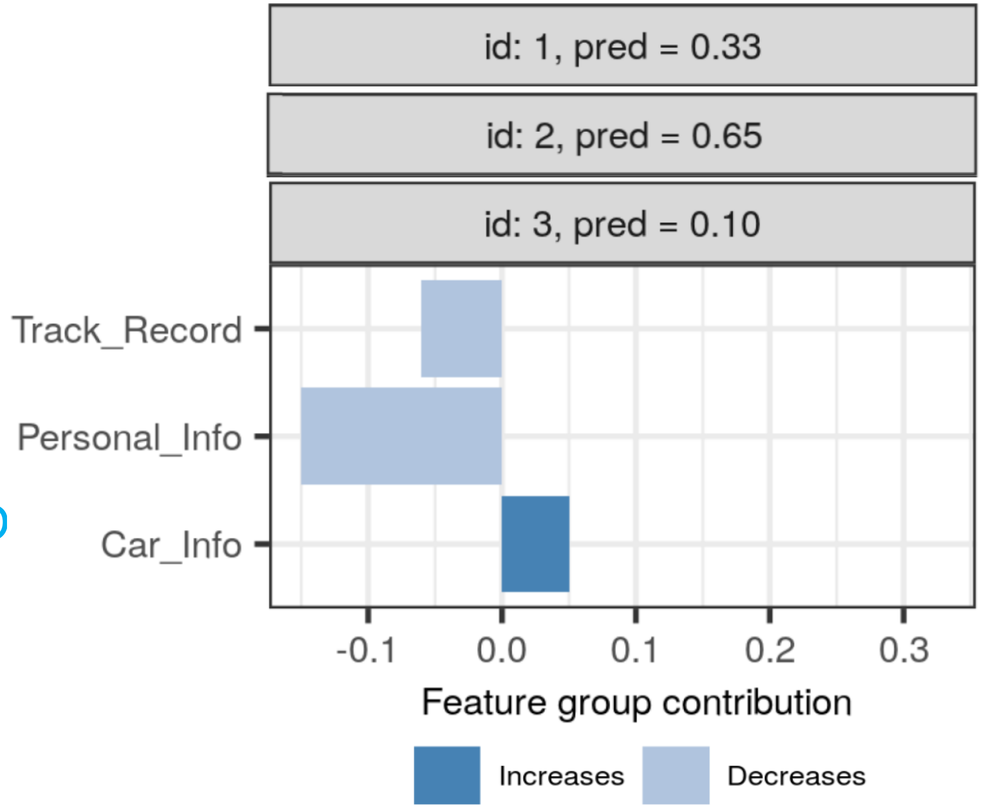
- US Car insurance dataset
  - 10 302 customers with records of crash/no crash + 21 features
  - Fit a random forest model with 500 trees to predict crash based on the 21 features

  - 3 feature groups based on type
    - Track record (4 features): # claims last 5 years, # licence record points, previous licence revokes, time as customer
    - Personal information (13 features): age of driver, education level, # children, job type, # driving children, marital status, gender, distance to work +++
    - Car information (4 features) value of car, age of car, type of car, whether car is red

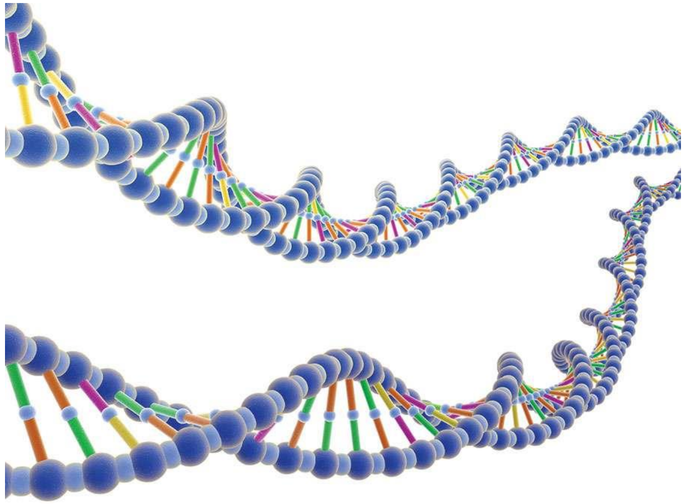
Practical example 1: Car insurance

- We apply the model to 3 individuals

1. 1 claim last 5 years, 3 licence record points.
   Single mother of 4 (2 driving).
   Driving a SUV, 27 miles to work.
2. Got licence revoked and 10 licence record points.
   37 year old father of 2 (1 driving).
3. 3 claims last 5 years, no licence record points
   60 year old married doctor with no children, with a PhD
   Red sports car.



Practical example 2: Gene data

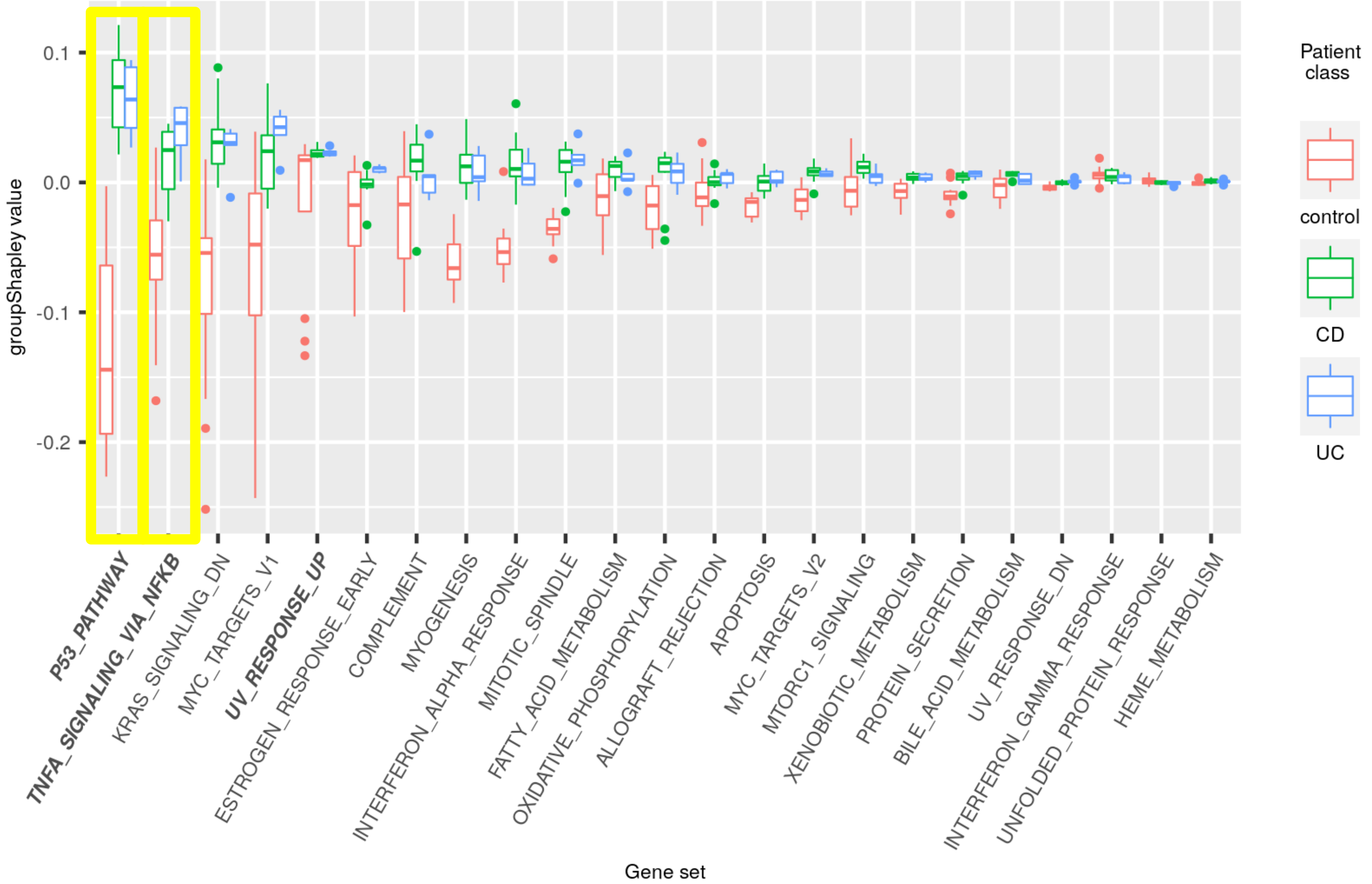


- Disease classification with high dimensional gene data
  - 127 patients where 85 are diseased with either Crohn's disease (CD) or Ulcerative colitis (UC) + 42 healthy controls.
  - 4 834 genes (after pre-processing)
  - Using 100 random individuals, we fit a Lasso penalized linear regression model to predict P(diseased with either CD or UC) based on the patient's genes

- Feature groups
  - Use the so-called Hallmark gene set to group the features (genes) into 23 different groups commonly used in gene set enrichment analysis
  - The Hallmark gene set "conveys a specific biological state or process" (Liberzon et al., 2015)

Practical example 2: Gene data

- Compute groupShapley values for the remaining 27 patients
- Make separate groupShapley boxplots for UC, CD and controls
- Can we identify genetical similarities and differences for UC and CD?
- Note: Model not trained to separate UC and CD



Concluding remarks

- Other use cases
  - Classification with time series data (see paper)
  - Explain original categorical features by grouping one-hot-encoded features
  - Explain image classification by grouping pixels into superpixels
  - Explain models with large number of feature-engineered variables based on original base features

- Implementation
  - Easy to apply in practice with the ***shapr*** R-package*
  - Code snippet for Car insurance example

```r
#### This code assumes "x_train", "x_test", and ####
#### "model" are pre-defined ####

library(shapr)
#### 1 Define groups ####
group_Names = list(Personal_Info = c("AGE", "EDUCATION", "HOMEKIDS", "HOME_VAL",
                                     "OCCUPATION","TRAVTIME","KIDSDRIV","MSTATUS",
                                     "PARENT1","GENDER","URBANICITY", "YOJ"),
                   Car_Info = c("BLUEBOOK", "CAR_AGE", "CAR_TYPE", "RED_CAR"),
                   Track_Record = c("CLM_FREQ", "MVR_PTS", "REVOKED", "TIF"))

# Prepare the explanation framework
explainer_group <- shapr::shapr(x_train,
                                model,
                                group = group_Names)

# Compute groupShapley values
explanation_group <- shapr::explain(x_test,
                                    approach = "ctree",
                                    explainer = explainer_group,
                                    prediction_zero = mean(x_train$CLAIM_FLAG))

# Plot the Shapley values
plot(explanation_group)
```

*groupShapley is currently only available in the GitHub version of *shapr*: https://github.com/NorskRegnesentral/shapr
Will be included in the next CRAN release