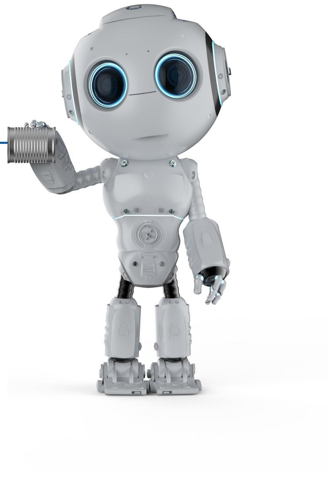
Shapley



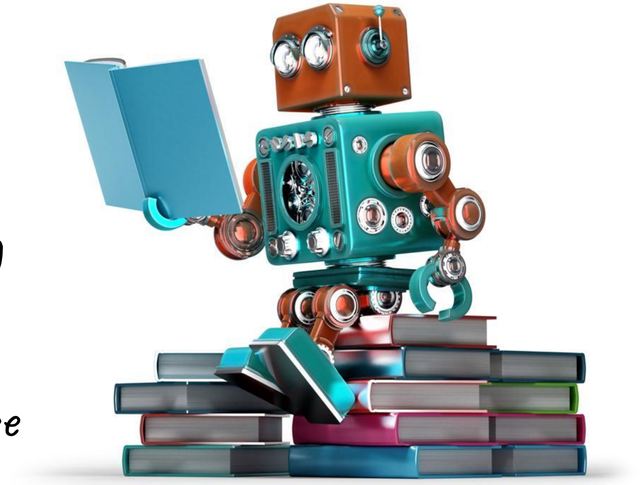
From game theory with love

day 2

Inga Strümke

inga@simula.no / inga@strumke.com

The menu - day 2, part 1

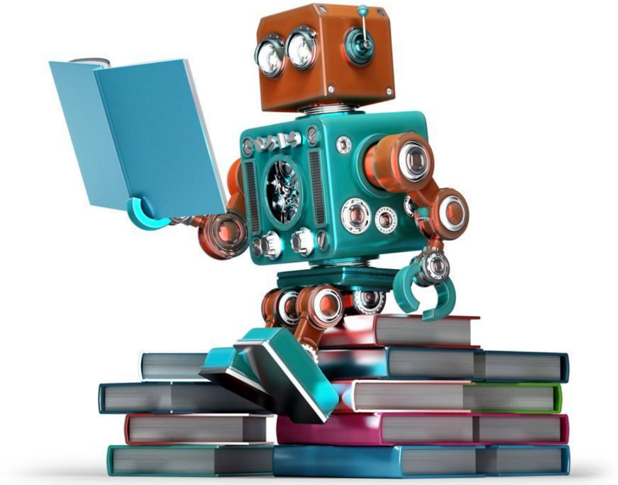


- *Quick recap (highlighting today's problem, so stay awake)*
- *Explaining a model, take 2*
- *Shapley values for ~all your ML needs: The SHAP package*
- *Hands on: SHAP on Boston Housing*
- *TreeExplainer and image SHAP*
- *Hands on: SHAP on MNIST*

Break

The menu - day 2, part 2

- *SHAP summary and relation to LIME*
- *Bonus:*
  - *Danger zone*
  - *Special cases*
  - *What are Shapley values not*
- *Woke takes and goodbye*



Recap (highlighting today's problem, so stay awake)

The Shapley value takes as input a set function $v: 2^N \rightarrow R$ which maps the input features to a single real number.

The Shapley value produces an attribution φ_i for each feature $i \in N$, that add up to $v(N)$.

The Shapley value of a feature i is given by

$$\phi_i(v) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(N - |S| - 1)!}{N!} (v(S \cup \{i\}) - v(S))$$

and all the magic happens in the characteristic function v and the **marginal contributions**, which are calculated by adding and removing features from coalitions.

Explaining a model - take 2

How to remove features from a trained model? <- severe confuse

Model building:

1. *Decide on input and output shape*
2. *Make architecture based on this*
3. *Adjust parameters to optimise something*

like XGBoost

Keywords: Parameterised and non-parameterised models

like the R^2

Two limitations

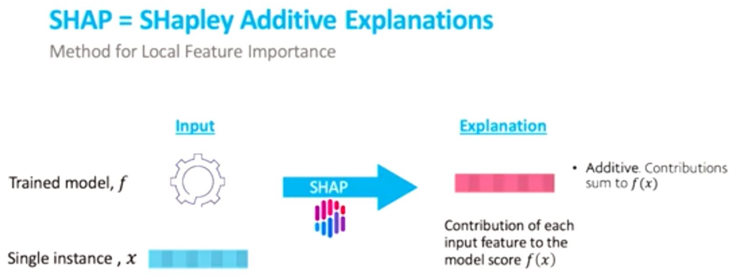
1. *Cannot remove input features from a trained, parametric model.*

    Retrain for each coalition? <- Not the same model

2. *Calculating Shapley values for N features requires 2^N evaluations of the characteristic function!* <- Exponential time (20 features => $2^{20} > 10^6$ evaluations of v)

What to do in practice?

The SHAP package



SHapley Additive exPlanations

Method for local feature importance

Input: trained model f and instance x

Output: contribution of each feature to the model output f(x)

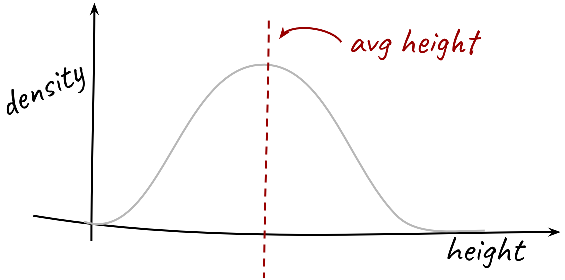
How does SHAP do this?

How SHAP works, the short story

Removing a feature is approximated by taking its expected value over a background data set

= Uninformed best guess at the feature's "actual value"

This works well in normally distributed cases where the average is a good statistic!



Just do it

Let’s get down and SHAP

Close your eyes and SHAP

Instantiate explainer on background data

Calculate SHAP values for some data

Close your eyes and SHAP

Instantiate explainer on background data

Calculate SHAP values for some data

```
explainer_bare = shap.TreeExplainer(regressor)
```

```
Setting feature_perturbation = "tree_path_dependent" because no background data was given.
```

Close your eyes and SHAP

Instantiate explainer on background data

Calculate SHAP values for some data

1. shap.force_plot on some instance i
2. shap.summary_plot on data set

   What do you see? E.g.

   a. Which feature affects the model output the most?
   b. In which direction do high crime rates (CRIM) drive the model output?
   c. Which feature has the most spread?

Close your eyes and SHAP

Explain one prediction: What drives the model prediction relative to the mean output across the data set?

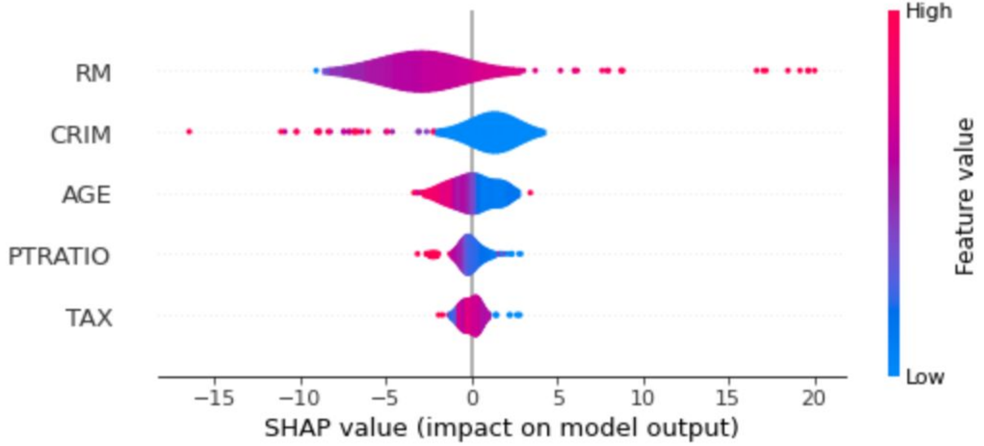
```python
# Pick an instance to look closer at
y_pred = regressor.predict(x_test)
print("Mean price: ", np.mean(y_test))
print("Mean price prediction: ", np.mean(y_pred))

print("Mean crime rate: ", np.mean(x_test.CRIM))

# i = 6: Old house where only the low crime rate helps push the price up
i=6
print(x_test.iloc[i])
print("Actual price: ", y_test.iloc[i])
print("Predicted price: ", y_pred[i])
shap.force_plot(explainer.expected_value, shap_values[i,:], x_test.iloc[i,:])
```

```
Mean price:  21.643712574850298
Mean price prediction:  21.19433
Mean crime rate:  4.1457728143712576
RM           5.91400
AGE         83.20000
TAX        304.00000
CRIM         0.31827
PTRATIO     18.40000
Name: 316, dtype: float64
Actual price:  17.8
Predicted price:  19.04583
```

Close your eyes and SHAP

Aggregated explanations: Collective explanation for the whole data set

```
shap.summary_plot(shap_values, x_test, plot_type="violin")
```



Use domain knowledge / intuition, look at the SHAP results and see if they make sense, i.e. do the worst thing you can do instead of causal inference. But ok.

shap.TreeExplainer

Only tree based models, but: Exact Shapley values! In polynomial time 😲

How to run a tree with missing input?

Encounter a split, input is missing. What to do? Answer: Go both ways. Recursively sum up the values, average the result.

Keep track of traversal to avoid repetition, et voila: Polynomial time.

```
# Initialise explainer on a background data set, the data parameter.
# (this is the data that data to ble explained is compared against when calculating SHAP values)
explainer = shap.TreeExplainer(regressor)#, data=x_train)

# Calculate SHAP values
shap_values = explainer.shap_values(x_test) #n_samples=100
```

```
Setting feature_perturbation = "tree_path_dependent" because no background data was given.
```

Image SHAP

Conceptually different from Attention Maps, GradCam & co. <- tell you about the activations inside the model

On images, SHAP is still about adding and removing features

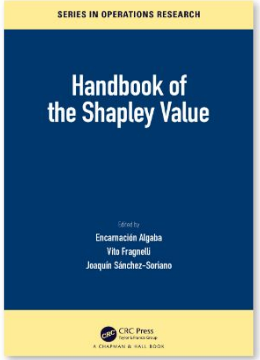
In general: Shapley values can be calculated for groups of features

Example group: $\{1,2\}$

Coalition without group: $S=\{3,4\}$

Coalition with group: $S=\{1,2,3,4\}$

Shapley values of groups and other awesomeness



Chapter

The Shapley Value as a Tool for Evaluating Groups: Axiomatization and Applications

By Ramón Flores, Elisenda Molina, Juan Tejada

Book [Handbook of the Shapley Value]

Edition	1st Edition
First Published	2019
Imprint	Chapman and Hall/CRC
Pages	25
eBook ISBN	9781351241410

Image SHAP

Conceptually different from Attention Maps, GradCam & co. <- tell you about the activations inside the model

On images, SHAP is still about adding and removing features

In general: Shapley values can be calculated for groups of features

Example group: {1,2}

Coalition without group: S={3,4}

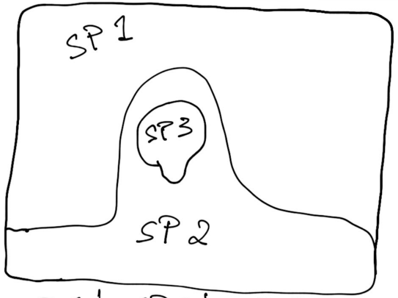
Coalition with group: S={1,2,3,4}

Pixel level would be overkill: Groups of pixels. **Super pixels!**

Coalitions of super pixels

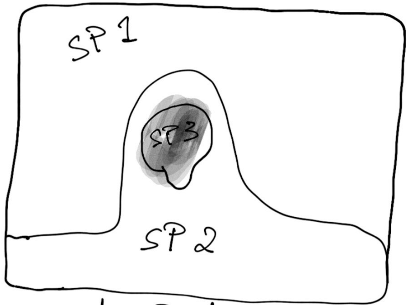
Instance

All super pixels included



SP 1	SP 2	SP 3
1	1	1

Super pixel 3 removed



SP 1	SP 2	SP 3
1	1	0

Just do it

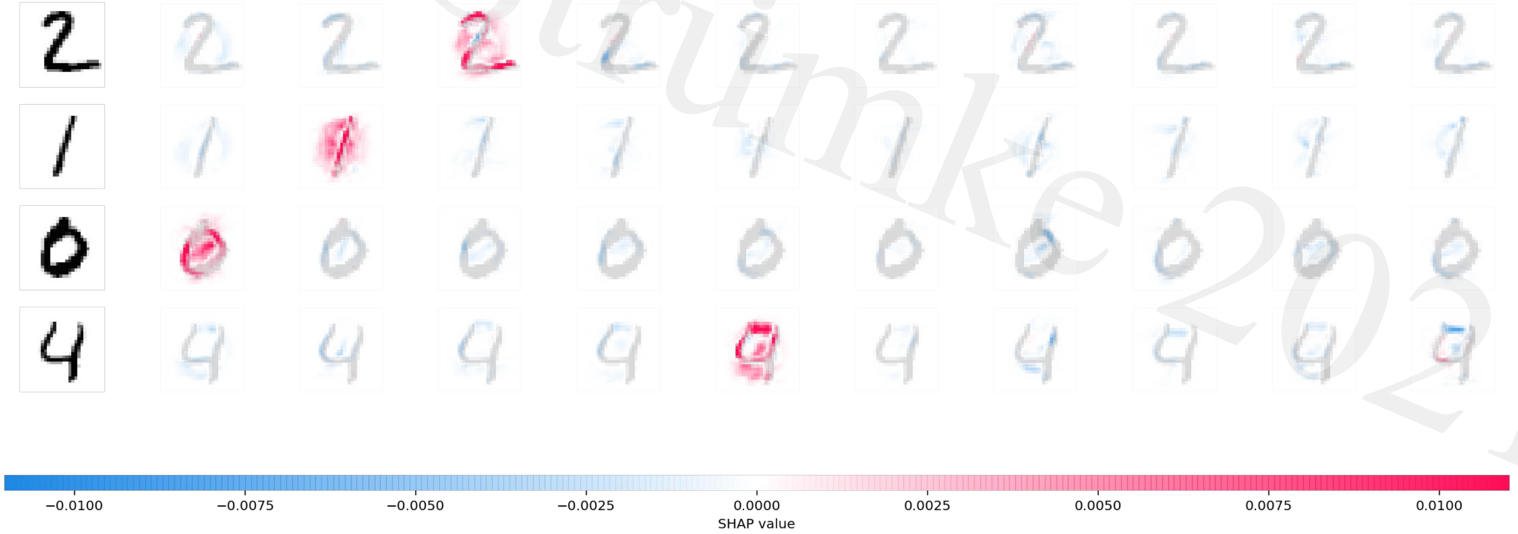
Some light image SHAP

During the break...

- *Coffee / tea*
- *Stretch*
- *(if not done: import data, train model)*
  Define background
  Create shap.DeepExplainer
  Calculate SHAP values,
  make shap.image_plot

SHAP task 2: MNIST

```
# Pick images to calculate SHAP values for
images = x_test[1:5]

# calculate SHAP values and plot them
shap_values = image_explainer.shap_values(images)
shap.image_plot(shap_values, -images)
```



Check out https://github.com/slundberg/shap for nice SHAP examples and explanations

SHAP: Take home message

1

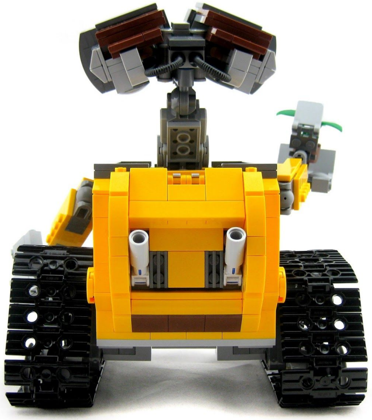
We want feature attributions

2

Shapley values offer a unique solution

3

Exact calculation is intractable



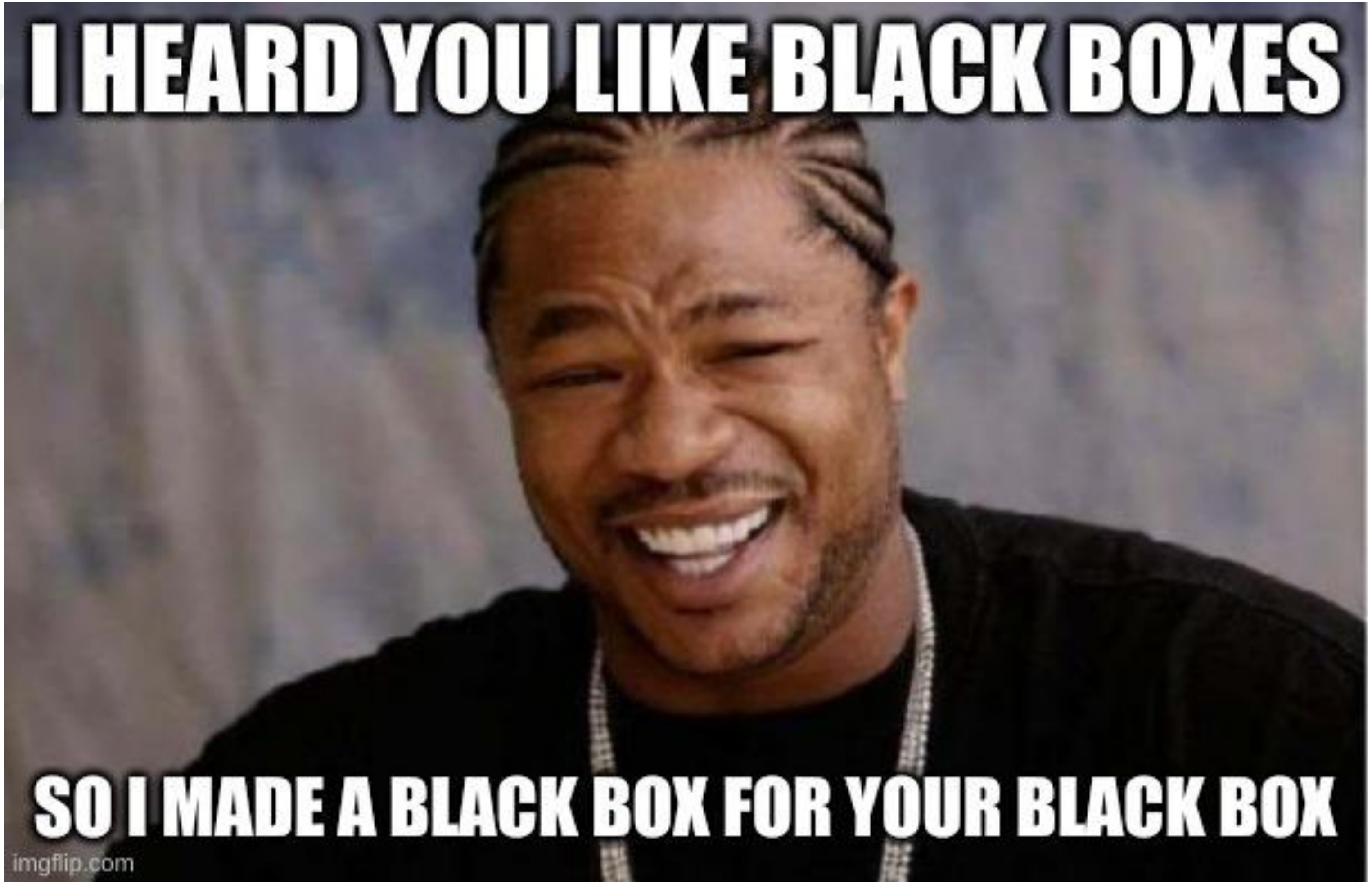
4

SHAP provides approximations: KernelExplainer, TreeExplainer, DeepExplainer, ...

Remember: Local explanation != global explanation.

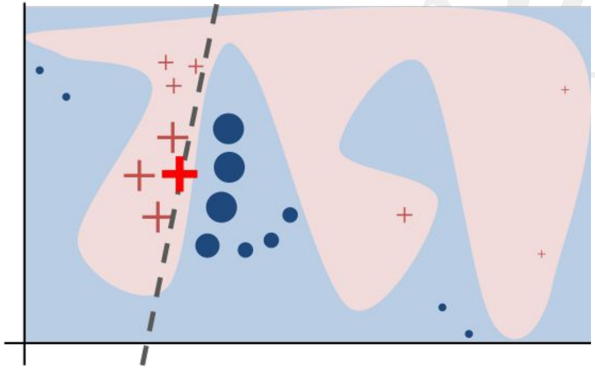
Have I explained my 🧘‍♀️ model? No, I've explained a prediction.

My sincere feeling when using SHAP



There is depth to SHAP as well

LIME: Local Interpretable Model-agnostic Explanations produces an interpretable model that locally approximates the full model. ("locally" is an interesting term, btw)



Full model

Model complexity

$$\text{explanation}(x) = \arg\min_{g \in G} L(f, g, \pi_x) + \Omega(g)$$

Family of explanation models

Loss

Local explanation model

Locality kernel

$$\pi_x(z') = \frac{(M-1)}{\binom{M}{|z'|}|z'|(M-|z'|)}$$

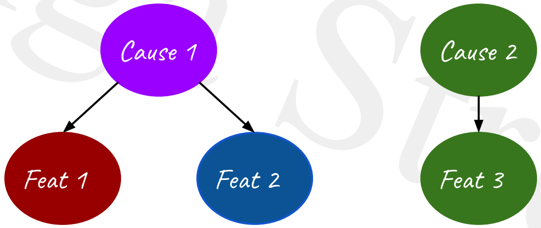
SHAP kernel

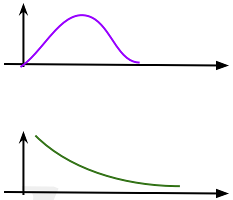
Bonus

Danger zone, special cases and woke takes

Danger zone: Correlated features

Two features created from the same cause (different effects), one from a second cause





$N(\mu=0.5, \sigma=0.1)$

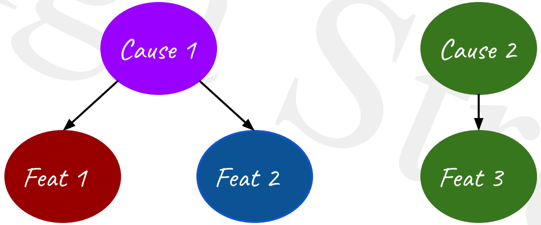
$Kumaraswamy(a=1, b=3)$

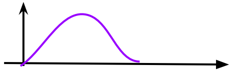
$y = f_1 + f_2 + 2 \times f_3$ *<- What should Shapley do? Attribute everything to one, or split evenly?*

(See what happens in Jupyter Notebook)

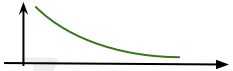
Danger zone: Correlated features

Two features created from the same cause (different effects), one from a second cause





$N(\mu=0.5, \sigma=0.1)$



Kumaraswamy(a=1, b=3)

$y = f_1 + f_2 + 2 \times f_3$ ->

```
# Create characteristic function dictionary
cf_dict = make_cf_dict(x_data, y_data, characteristic_function_r2);
```

```
# and calculate Shapley values
calc_shapley_values(x_data, y_data, cf_dict)
```

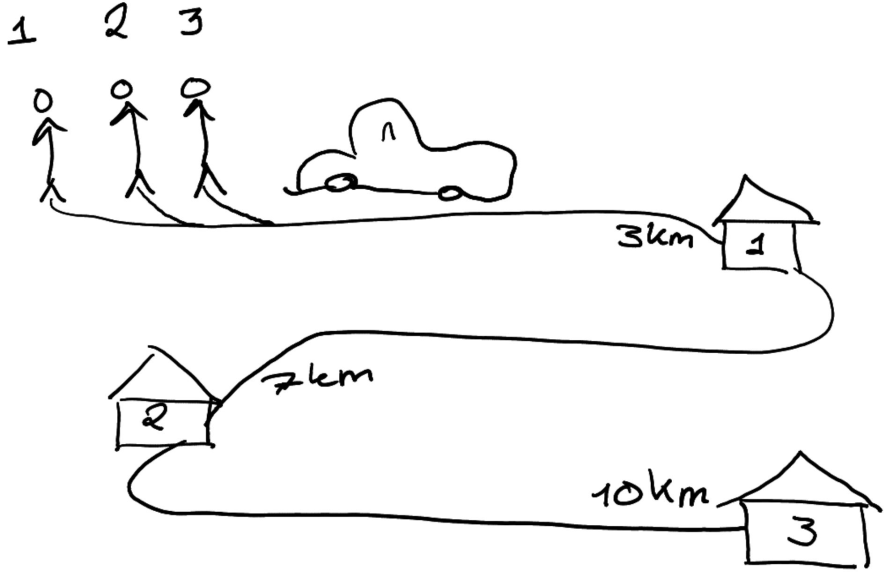
```
[0.05902964568235743, 0.05970174250118678, 0.8812686118164553]
```

What if you have 100 correlated features? => Shapley value split into 100 parts. Each feature looks unimportant.

Feature selection: Useful players may be useless features

Remember the cab scenario, with $N = [3]$ and

$v(\{1,2,3\}) = v(\{2,3\}) = v(\{1,3\}) = v(\{3\}) = 10,\ v(\{1,2\}) = v(\{2\}) = 7,\ v(\{1\}) = 3$



Feature selection: Useful players may be useless features

$v(\{1,2,3\}) = v(\{2,3\}) = v(\{1,3\}) = v(\{3\}) = 10,\ v(\{1,2\}) = v(\{2\}) = 7,\ v(\{1\}) = 3$

Shapley $\varphi = (1,3,6)$

Model selection problem: Choose a set S of features that maximises $v(S)$ while minimising the cost (complexity) $C(S)$ (increasing function of $|S|$)

Q: What is the solution?

Feature selection: Useful players may be useless features

$v(\{1,2,3\}) = v(\{2,3\}) = v(\{1,3\}) = v(\{3\}) = 10,\ v(\{1,2\}) = v(\{2\}) = 7,\ v(\{1\}) = 3$

Shapley $\varphi = (1,3,6)$

Model selection problem: Choose a set S of features that maximises $v(S)$ while minimising the cost (complexity) $C(S)$ (increasing function of $|S|$)

Q:What is the solution? A: $S=\{3\}$ (if all features have equal cost)

Model selection: Players 1 and 2 are useless features

Feature selection: Useful players may be useless features

$v(\{1,2,3\}) = v(\{2,3\}) = v(\{1,3\}) = v(\{3\}) = 10$, $v(\{1,2\}) = v(\{2\}) = 7$, $v(\{1\}) = 3$

Shapley $\varphi = (1,3,6)$

Model selection: Players 1 and 2 are useless features

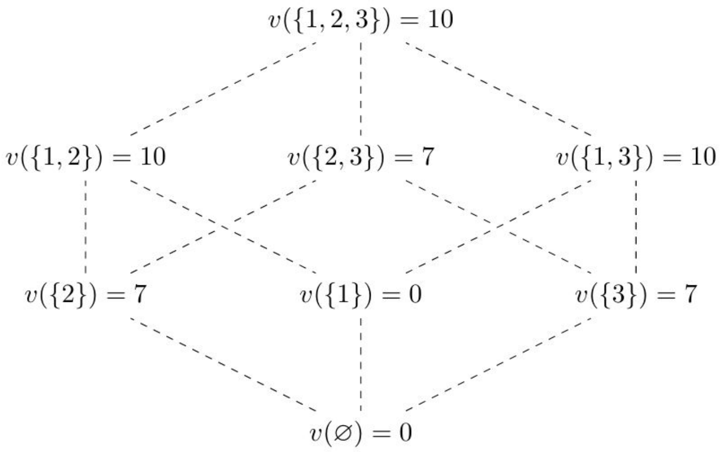
Fairness (in the Shapley sense): Players 1 and 2 are not worthless (i.e. dummy players), as they always add value in the absence of player 3

Value for model selection != Shapley values

Top players may be poor performers

Scenario: [N] = 3 and a "secret holder" payoff

Player 1 has no value alone, but unlocks the maximum performance of any team



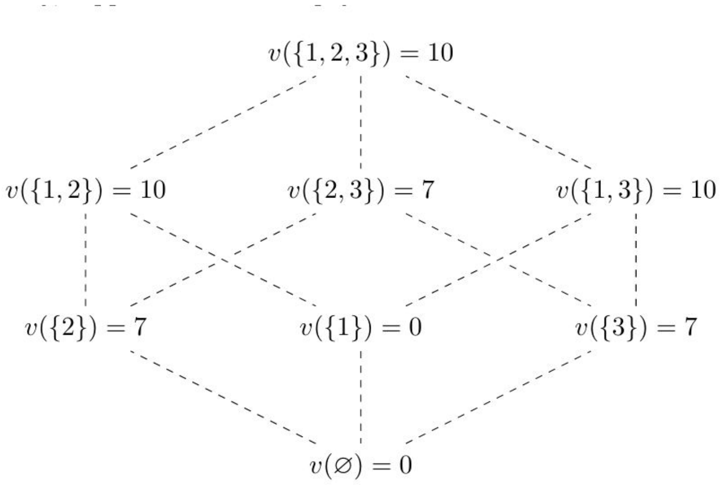
Idea by Daniel V. Fryer

Top players may be poor performers

Scenario: [N] = 3 and a "secret holder" payoff

Player 1 has no value alone, but unlocks the maximum performance of any team

φ = (2,4,4), **but should keep only {1,2} or {1,3}**




```
cf_dict = {():0,(1,):0,(2,):7,(3,):7,(1,2):10, (1,3):10, (2,3):7,(1,2,3):10}
all_players = [1,2,3]
```

```
print(calc_shapley_value(1, all_players, cf_dict))
print(calc_shapley_value(2, all_players, cf_dict))
print(calc_shapley_value(3, all_players, cf_dict))
```

```
2.0
4.0
4.0
```

Idea by Daniel V. Fryer

Q: What are Shapley values not?

Adding/removing features feels like interventions, but of course it's not <- Removing a proxy variable does not remove its effect

A: Causal inference

The Shapley values don't tell you how valuable a feature is for modelling

A: A feature selection tool

Woke takes

High Shapley value doesn't mean important for underlying process.
Low Shapley value doesn't mean unimportant for underlying process.

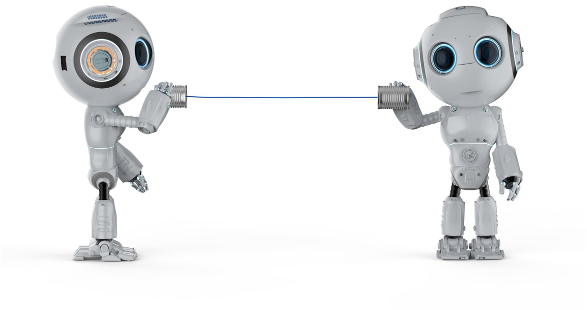
Don't use Shapley values for feature selection, keep in mind that proxy variables exist and that machine learning is correlation detection.

Shapley values are operationalised in a ~million different ways (referencing the model, approximations, data used) in different explanation contexts <- not so "unique"

No free lunch theorem <- no explanation concept can be perfect for all cases anyway

Bye now <3

Thank you <3



Want to do Shapley related research?

inga@simula.no / inga@strumke.com