Introduction to explainable artificial intelligence

SHAP and GNNExplainer

Agnieszka Wojtuch[1]

[1]Department of Informatics and Chemistry,
Vysoká škola chemicko-technologická v Praze

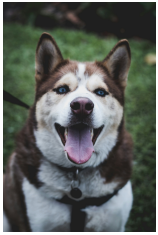
June 2025

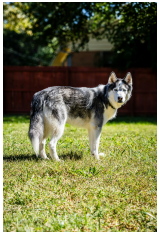
Table of Contents

Table of Contents

training data

huskies:





wolves:

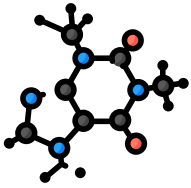


input image

explanation

correct: husky
predicted: wolf

Explainability – definition

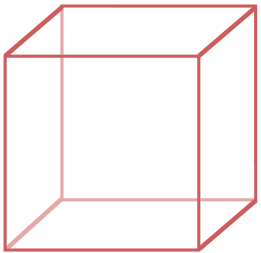


Explainability consists of methods that explain how a particular AI model works in a comprehensible way.

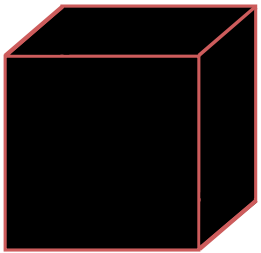
- why was this sample classified as negative? (right to explanation)
- which features does the model use? (fairness, scientific discovery)
- how this feature affects the prediction? (right to explanation, sample optimisation)

Explainability vs interpretability

interpretability

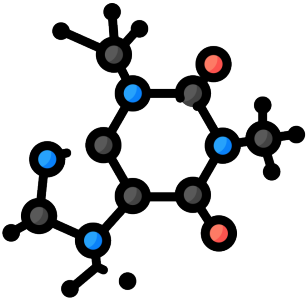


explainability

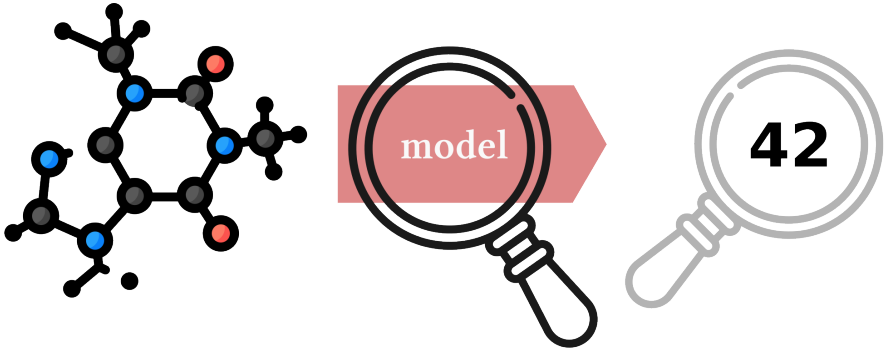


- **interpretability** – we can interpret what the model does e.g. a small decision tree
- **explainability** – someone has to explain the model's inner working to us
  e.g. a neural network

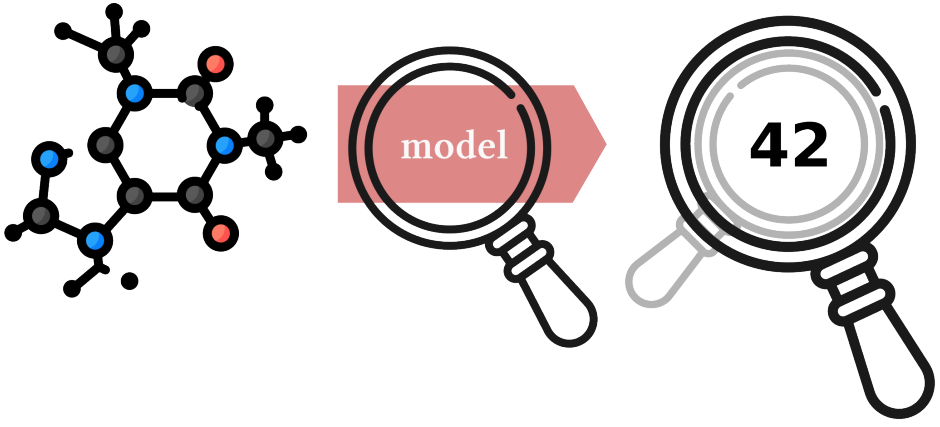
Explainability vs AI



model


model
42

Explainability vs AI

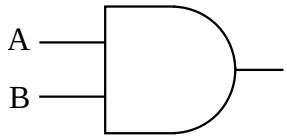


Tasks

1. understanding the model
   - model diagnostics
   - model trust
2. understanding the modelled process
   - scientific discovery
   - molecule optimisation

Table of Contents

Example 1 – Boolean AND

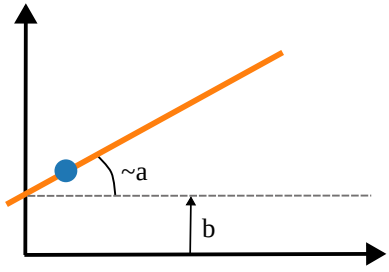


model: logic AND gate
inputs: A, B
output: 1 if both A and B are 1; otherwise 0

What is the relative importance of these features?
$A = 1, B = 1$

And in this case?
$A = 0, B = 1$

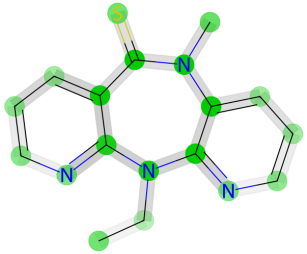
Example 2 – linear regression

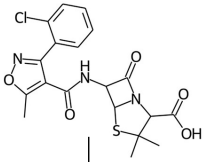


$$y = \underbrace{X \cdot a}_{\text{explanation}} + b$$

Example 3 – different explanations

1	0	0	1	0	1	1	0
1.7	-0.3	0.45	0.97	-2.1	0.03	1.25	-1.4





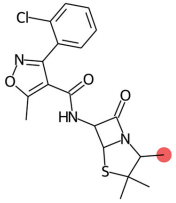


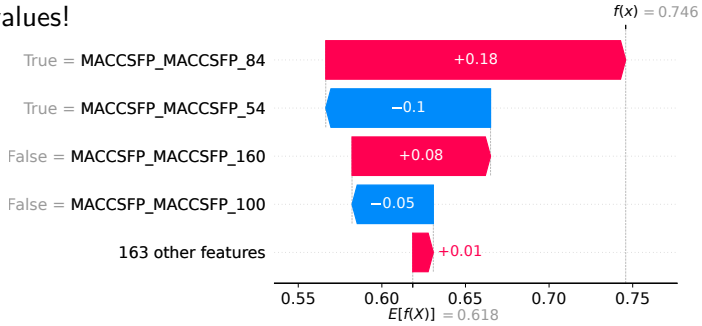
Table of Contents

SHAP – general idea

🤔 The linear regression-like explanations sum to the prediction so they're easy to interpret.

Could we use them for other models somehow?

🤦 We could use Shapley values!



- multi-player coalition game – the players work together to gain the biggest reward they can
- the reward is then divided between the players

Problem: How to split the total payout (reward) between the players in **a fair way**, i.e. taking into consideration their contribution?

Shapley showed that there exists **a unique solution** to this problem.

Efficiency

The entire reward must be split.

Symmetry

If two players contributed equally then their payout should be the same.

Dummy

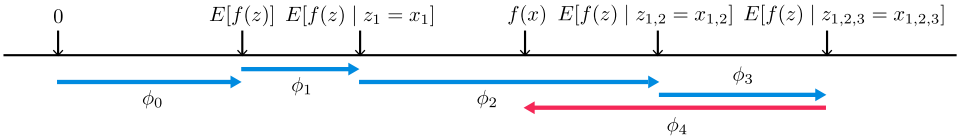
A player who has no impact on the reward value gets no reward.

Additivity

If players play one game after another then calculating their rewards after each game should give the same result as calculating their total rewards once, after the last game.

Definition: The Shapley value is the average marginal contribution of a feature value across all possible coalitions.

$$\phi_j(val) = \sum_{\underbrace{S \subseteq}_{\text{all possible coalitions}} \underbrace{\{x_1, \ldots, x_p\}}_{\text{of } p \text{ players}} \underbrace{\setminus \{x_j\}}_{\text{without player } j}} \underbrace{\frac{|S|!(p-|S|-1)!}{p!}}_{\text{weighting factor}} \underbrace{\left(\overbrace{val(S \cup \{j\})}^{\text{reward with player } j \text{ present}} - \overbrace{val(S)}^{\text{reward without player } j}\right)}_{j\text{'s contribution for coalition } S}$$



When calculating Shapley values, all possible orderings are considered and an average contribution is calculated.

Order matters when the function is nonlinear or the variables are not independent.

Order matters – nonlinearity

Nonlinearity: let's go back to our AND example. A $= 0$, B $= 1$

- if we learn about A first, then it gets the entire contribution – we already know what the output would be – and so B will get 0 contribution
- but if we learn first about B, then we get some information but we still don't know what the output would be – B gets non-zero contribution

Order matters – dependent variables

Dependent variables:

- we have four variables: A, B, C, D; B = C + D
- we predict the sum of all the variables
- if we learn about the variables in order: A, B, C, D; then C and D have no contribution – once we know A and B, we know that the sum is A + 2B
- if we learn about the variables in order: A, C, D, B, then C and D have nonzero contribution, but B has zero contribution – once we learn about A, C, D, we know that the sum is A + 2 * (C + D)

Weighting factor for a coalition of players

$$\underbrace{A\ B\ C\ E\ F\ G\ H\ I}_{\text{coalition } S}\ J\ \underbrace{K\ L\ M\ N\ O\ P}_{\text{remaining players}}$$

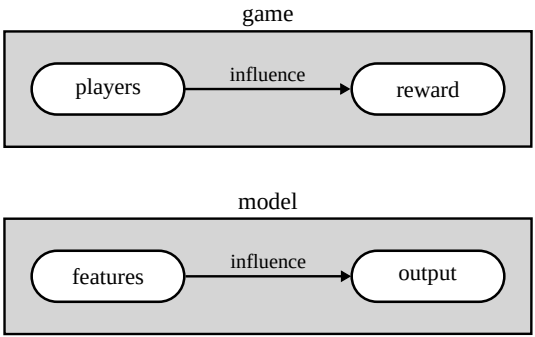
When calculating j's contribution for some coalition S we need to know how many sequences correspond to this situation:

- $|S|$ players can be ordered in $|S|!$ sequences
- the remaining players can be ordered in $(p - |S| - 1)!$ sequences
- coalition S corresponds to $|S|!(p - |S| - 1)!$ sequences

There are $p!$ sequences of p players, so each sequence has weight $\frac{1}{p!}$
Therefore, the weight for j's contribution to coalition S is:

$$\frac{|S|!(p - |S| - 1)!}{p!}$$

Shapley values → SHAP values



Missing players – missing features

$$\phi_j(val) = \sum_{S \subseteq \{x_1,\ldots,x_p\} \setminus \{x_j\}} \frac{|S|!(p-|S|-1)!}{p!} (val(S \cup \{j\}) - val(S))$$

- When we calculate Shapley values we play out an imaginary scenario in which some players are not taking part in the game.
- In AI, this corresponds to not knowing values of some features. But most models don't accept missing features!

Solution: Although one cannot calculate a prediction when a feature is missing, one can pretend not to know its value. The value of the 'missing' feature is replaced with any value from the dataset and prediction can be easily calculated.

SHAP values are not exactly the same as Shapley values.
SHAP values are the Shapley values of a **conditional expectation function of the original model**.

$$\hat{y} = f(x) \rightarrow \bar{y} = \mathbb{E}[f(x)]$$

$\tilde{y} = f(\tilde{x})$

$\bar{y} = \mathbb{E}_{|x_{1:4}}[f(x)]$

In simple English: SHAP values are the Shapley values of **a fancy version of our model**.

SHAP values – working example

Let's predict apartment price based on four features:

- 🥇, 🥈, 🥉 – floor
- 🔢 – size
- 🌳, 🪓 – park nearby?
- 😺, 😾 – are cats allowed?

(🥈, 🔢50m^2, 🌳, 😾) → 300K

What is the contribution of the cat feature 😾?

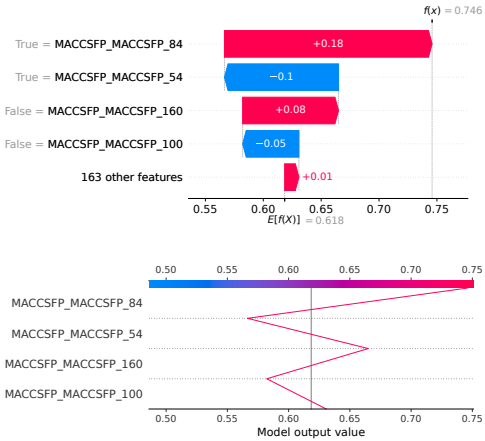
1. Sample a coalition of features: 🔢50m^2, 🌳
2. Sample the values for the missing features: 🥇
3. Prediction for the new sample: 🥇, 🔢50m^2, 🌳, 😾 → 310K
4. Replace the cat feature with a randomly drawn value: 😾 → 😺
5. Prediction for the new sample: 🥇, 🔢50m^2, 🌳, 😾 → 320K
6. The contribution of the cat feature is: 310K - 320K = -10K
7. This is a very rough estimation! Average over more samples → repeat steps 2-6.
8. Do this for every coalition → repeat steps 1-7.

$$\sum_{S \subseteq \{x_1,\ldots,x_p\} \setminus \{x_j\}} \frac{|S|!(p-|S|-1)!}{p!} \left(val(S \cup \{j\}) - val(S)\right)$$

Be careful to interpret the Shapley value correctly: the Shapley value is the **average contribution** of a feature value to the prediction **in different coalitions**.

The Shapley value is **NOT** the difference in prediction when we would remove the feature from the model.

SHAP – summary



- additive feature attribution method
- explains how to get from the base prediction (mean prediction of the model) to the actual prediction
- suitable for tabular data (e.g. fingerprints)
- both classification and regression
- model-agnostic – works with any model
- provides local explanations – explains predictions (not the entire model)
- perturbation-based – calculates explanations by perturbing (introducing small changes into), the input instance
- based on Shapley values from game theory – nice mathematical guarantees
- can be rather slow but there exist model-specific fast alternatives
- **Limitation:** prone to unrealistic data instances

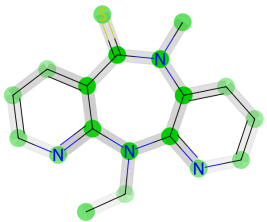
Table of Contents

GNNExplainer – general idea

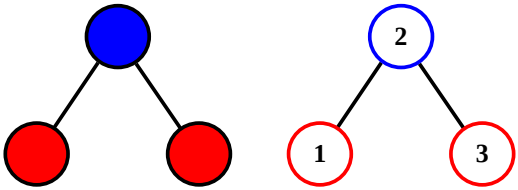
🤔 How to calculate explanation of this prediction?

🤦 Let's do what we always do – train a neural network to solve the problem for us!

💁 We want to see which nodes are important for the prediction. And edges. And features.



Graph representation recap



$$A = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

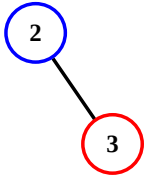
$$A = [[1, 2], [2, 1], [2, 3], [3, 2]]$$

feature order: $\mathrm{B}, \mathrm{C}, \mathrm{N}, \mathrm{O}, \mathrm{F}$

$$X = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

$$M_v = [0, 1, 1]$$

$$M_e = [0, 0, 1, 1]$$



Bernoulli distribution

Bernoulli distribution:

$$P(X = 1) = p$$
$$P(X = 0) = 1 - p$$

$$P(x_1 = 1, x_2 = 0, x_3 = 1) =$$
$$P(X = 1) \cdot P(X = 0) \cdot P(X = 1) =$$
$$p \cdot (1 - p) \cdot p$$

Multivariate Bernoulli distribution

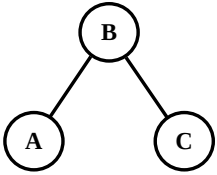
Multivariate Bernoulli distribution:

$P(A = 1) = p_A$

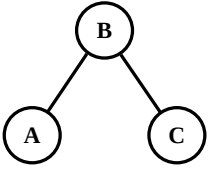
$P(B = 1) = p_B$

$P(A = 1, B = 0) = p_A \cdot (1 - p_B)$

Probability distribution over graphs

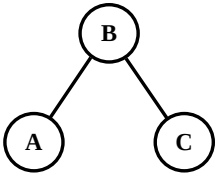


$$P(A = 1) = p_A$$
$$P(B = 1) = p_B$$
$$P(C = 1) = p_C$$

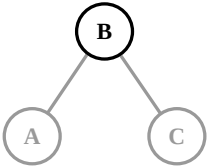


$$P(A = 1, B = 1, C = 1) = p_A \cdot p_B \cdot p_C$$

Probability distribution over graphs

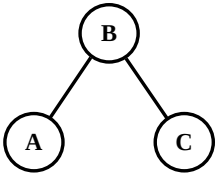


$$P(A=1) = p_A$$
$$P(B=1) = p_B$$
$$P(C=1) = p_C$$

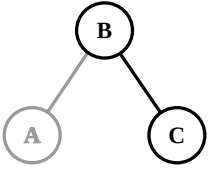


$$P(A=0, B=1, C=0) = (1-p_A)\cdot p_B \cdot (1-p_C)$$

Probability distribution over graphs



$$P(A = 1) = p_A$$
$$P(B = 1) = p_B$$
$$P(C = 1) = p_C$$



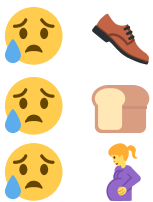
$$P(A = 0, B = 1, C = 1) = (1 - p_A) \cdot p_B \cdot p_C$$

entropy:

$$H[X] = -\sum_{x \in X} p(x) \log p(x)$$

mutual information:

$$MI[X, Y] = H[X] - H[X|Y] = H[Y] - H[Y|X]$$



GNNExplainer – idea

A proxy model is trained to generate such a mask M that defines a subgraph of the input graph which maximises the probability of class c predicted by black-box model Φ.

In other words:

- we look for a smaller graph that would give the same prediction as the input graph,
- this smaller graph is defined by masking the input graph with mask M,
- mask M is generated by a proxy model.

GNNExplainer – formulation

$$\max_{G_S} \underbrace{MI(Y, (G_S, X_S))}$$

random variable representing labels: Y

features of G_S: X_S

examine all subgraphs of G: $\max_{G_S}$

how much do we know about Y knowing (G_S, X_S)?: $MI(Y, (G_S, X_S))$

We express mutual information in terms of entropy:

$$\max_{G_S} MI\left(Y, (G_S, X_S)\right) = \overset{\substack{\text{uncertainty}\\ \text{about } Y\\ \text{(constant)}}}{\overset{\downarrow}{H(Y)}} - \underbrace{H(Y|G = G_S, X = X_S)}_{\substack{\uparrow\\ \text{uncertainty about } Y\\ \text{knowing } (G_S, X_S)}}$$

GNNExplainer – make it tractable

$$\min_{G_S} \longrightarrow \min_{\mathcal{G}} \mathbb{E}_{G_S \sim \mathcal{G}}$$

$\min_{\mathcal{G}}$
↑
we look for such distribution on graphs that minimises...

$\mathbb{E}_{G_S \sim \mathcal{G}}$
↑
...expected value conditioned on graphs drawn from this distribution

$\underbrace{H(Y|G = G_S, X = X_S)}$
↑
uncertainty about Y knowing (G_S, X_S)

GNNExplainer – the upper bound

Using Jensen's inequality ($\mathbb{E}[\varphi(X)] \leq \varphi(\mathbb{E}[X])$), we get the following upper bound:

$$\mathbb{E}_{G_S \sim \mathcal{G}} H(Y|G = G_S, X = X_S) \leq H(Y|G = \mathbb{E}_{\mathcal{G}}[G_S], X = X_S)$$

so we can do:

$$\min_{\mathcal{G}} H(Y|G = \mathbb{E}_{\mathcal{G}}[G_S], X = X_S)$$

How to estimate $\mathbb{E}_{\mathcal{G}}$?

We will parametrise the distribution over graphs, $\mathcal{G}$, as a multivariate Bernoulli distribution:

$$P_{\mathcal{G}}(G_S) = \prod_{(j,k)\in G_c} \underbrace{A_S[j,k]}_{\text{expectation on whether edge } (v_j, v_k) \text{ exists}}$$

- $P_{\mathcal{G}}$: probability defined by distribution $\mathcal{G}$
- G_S: graph sampled from $\mathcal{G}$
- $\prod_{(j,k)\in G_c}$: edge between j^{th} and k^{th} node; edges from **computational** input graph
- A_S: fractional adjacency matrix of G_S

GNNExplainer – what to learn?

Thus

$$H(Y|G = \mathbb{E}_{\mathcal{G}}[G_S], X = X_S) = H(Y|G = \underbrace{A_c \odot \sigma(M)}_{\text{expected graph}}, X = X_S)$$

- $\odot$ – element-wise multiplication
- M – mask that we need to learn
- $\sigma(\cdot)$ – sigmoid

GNNExplainer – towards the loss function

$$H[X] = -\sum_{x \in X} p(x) \log p(x)$$

$$H(Y|G=G_S, X=X_S) = -\mathbb{E}_{Y|G_S,X_S}\left[\log \underbrace{P_\Phi}_{\substack{\text{probability defined} \\ \text{by model } \Phi}} \underbrace{(Y|G=G_S, X=X_S)}_{\substack{\text{distribution of labels} \\ \text{given only a subgraph} \\ \text{and a subset of features}}}\right]$$

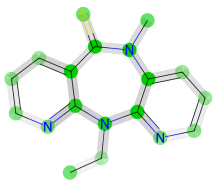
expected distribution of Y conditioned on G_S,X_S

GNNExplainer – the loss function

$$\min_{M} \underbrace{-\sum_{c=1}^{C} \mathbb{1}[y=c] \log}_{\text{cross-entropy loss}} \underbrace{P_{\Phi}(Y=y|G=A_c \odot \sigma(M), X=X_c)}_{\Phi\text{'s prediction for the masked graph}}$$

GNNExplainer – what have we learnt?

- simplifying the optimisation criterion
- modelling graphs with a probabilistic distribution (helped with tractability)
- using Jensen's inequality to get an upper bound
- tricking neural networks into returning values from $[0, 1]$ range



- suitable for graph representations
- model-agnostic – works with any graph neural-network
- uses a proxy-model – a small graph neural network is trained to predict the explanation
- explanation is a subgraph of the input graph and a subset of features that are important for the prediction
- both classification and regression
- provides local explanations
- produces graph-level explanations – graph-level tasks depend on the entire graph, e.g. activity prediction
- produces node-level explanations – node-level tasks concern a single node, e.g. predicting aromaticity of each atom
- **Limitation:** the authors assume (by using Jensen's inequality) that the function represented by GNN being explained is convex

Attributions:

- Motivation
  - dataset photographs come from Unsplash.com
  - input and explanation photographs come from *„Why Should I Trust You?" Explaining the Predictions of Any Classifier* M. T. Ribeiro et al.
- Example 3 – different explanations – an example of counterfactual explanation was adapted from *A Perspective on Explanations of Molecular Prediction Models*, Geemi P. Wellawatte, et al.
- Shapley values, The unique solution, Missing players – missing features
  - image of Lloyd Shapley – CC BY-SA 2.0 DE copyright is with Mathematisches Institut Oberwolfach, background was removed from the image
  - cards – from Flaticon.com
- Order matters – *A Unified Approach to Interpreting Model Predictions*, Scott M. Lundberg and Su-In Lee
- twemoji icons – The Twitter emoji graphics provided by twemojis are licensed under CC-BY 4.0. Copyright ©2019, Twitter, Inc. and other contributors.