Interactive explainable AI platform for graph neural networks

Jacqueline Beinecke[1] Anna Saranti[2,3] Alessa Angerschmid[2,3] Bastian Pfeifer[2]
Vanessa Klemt[4] Andreas Holzinger[2,3] Anne-Christin Hauschild[1]

[1]Institute for Medical Informatics, University Medicalcenter Göttingen, Germany
[2]Institute for Medical Informatics, Statistics and Documentation,
Medical University Graz, Austria
[3]Human-Centered AI Lab, University of Natural Resources and Life Sciences, Vienna, Austria
[4]Philipps University Marburg, Germany

Abstract

Lack of trust in artificial intelligence (AI) models in medicine is still the key blockage in the use of AI as clinical decision support systems (CDSS). Although AI models are already performing excellently in medicine, their black-box nature means that it is often impossible to understand why a particular decision was made. In the field of explainable AI (XAI), many good tools have already been developed to "explain" to a human expert, for example, which input features influenced a prediction. However, in the clinical domain, it is essential that these explanations lead to some degree of causal understanding by the domain expert in the context of the application domain. For this reason, we have developed an interactive XAI platform that allows the subject matter expert to ask counterfactual ("what-if") questions. Our platform allows a subject matter expert to observe how changes based on their questions affect the AI decision and the XAI explanation.

Keywords: Explainable AI, Graph Neural Networks, Counterfactuals, Human-in-the-loop, Causability

1 Introduction

In the domain of clinical decision support systems (CDSS), there is a need for trust in machine learning (ML) and artificial intelligence (AI) models. This is mainly because AI in the medical domain is highly critical since it often involves patient care or treatment development [28]. It is unacceptable for AI models to not be interpretable in such a setting since wrong predictions could cause severe consequences [7]. A lack of interpretability and, therefore, lack of trust in AI systems as CDSS typically stems from the blackbox nature of complex AI models [30].

In recent years explainable AI (XAI) models have been developed to help build trust in AI models and increase their transparency. They achieve this, for example, by calculating relevance scores for inputs based on the AI models configurations [38][1], by creating saliency maps [35] or by analysing the model down to a single neuron [8]. Several surveys and practical tutorials compare and analyse state-of-the-art XAI methods in different scenarios with different data [2] [16].
In the medical and biomedical domain, however, transparency alone is not sufficient in reaching a state of "explainable medicine". In medicine, there is a need for causability [15], the measurable extent to which an explanation from an XAI model achieved a specified level of causal understanding.
In order to connect the transparency of XAI models with the need for causability, we need interactive XAI platforms that guide domain experts through the explanations given by an XAI model. Such platforms will allow the expert to gain insight into what influenced the AI model in its decision-making process. Additionally, it can help the expert identify confounding factors that might inhibit model performance or correlated factors that are biologically unimportant. Most importantly, it is imperative to allow the expert to interactively ask counterfactual questions ("if-not", "why-not", and "what-if" questions) and change his data based on those. This is going to help the expert see how changes affect not only the AI model but also the XAI model, hence, increasing their causal understanding.

In many research domains, dependencies between parameters have an additional impact on AI models. Such dependencies are often represented in graph structures. For example, in the medical and biomedical domain, much research is being done on protein-protein interaction (PPI) graphs, especially in cancer research and subtyping, since the gene mutations caused by cancer can lead to a gain or loss of PPIs in the graphs [12]. Graph neural networks (GNN) have been specifically developed to handle graph-structured input data [44]. They have already been used on PPI graphs, for example, to predict novel cancer genes [33] or for disease subnetwork detection [24]. Many XAI models for GNNs have already been developed, such as the GNNExplainer [47], PGMExplainer [41], XGNN [48], [32]. But, as pointed out earlier, while these XAI models might increase the transparency of GNNs and help with

understanding their inner processes, there is still a lack of causability.

For this reason, we developed an interactive XAI platform for GNNs. Our interactive XAI platform will allow domain experts to understand why GNNs reach a particular decision and what features are important in the decision-making process. The interactivity will enable the expert to act based on their already gained understanding. Furthermore, the platform will let him/her see the consequences of the action, which will increase his/her causal understanding. This interactive XAI platform will pave the way for informed medical decision-making and the application of AI models as CDSS.

2 Results

2.1 Design overview of the interactive XAI platform

The aim of our interactive XAI platform is to promote human understanding of GNN predictions and to allow the domain expert to validate and improve the GNN decision-making process. For this reason, we have developed a platform that visualises the input graphs that were used to train and test the GNN, including node and edge attributes, as well as, node and edge relevances computed by XAI methods, such as GNNExplainer [47]. Through this the domain expert can gain insights into what parts of the graph were most influential in the GNNs decision-making process. The domain expert can then interactively manipulate the graph dataset based on his/her understanding and initiate a retraining or prediction of the GNN. This interactivity allows the domain expert to ask counterfactual questions and analyse the resulting effects on the GNN decision.
The platform is being hosted by the GWDG and reachable under the following domain:
`http://rshiny.gwdg.de/apps/interactive_xai_for_gnn/`
The User Interface (UI) is divided into three main windows. The entry window (Home), the selection window (Select Data) and the interaction window (Interact). The first window serves as a starting point for the user. Here the user gets a quick overview of the purpose of the platform along with a short description on how to use the platform. After reading this short introduction carefully, the user can continue to the next window and select a dataset. The user can choose between two pre-selected datasets, namely the Kidney Renal Clear Cell Carcinoma (KIRC) dataset (a real world dataset) and a smaller subset of the KIRC dataset as well as a synthetic dataset. The synthetic dataset is built out of 1000 Barabasi networks with two synthetic classes. This dataset is smaller with only 30 nodes and 29 edges per graph making it easier for the user to get familiar with our platform than with the KIRC dataset. The KIRC dataset is a larger real world dataset, taken from The Cancer Genome Atlas (TCGA) database (`https://cancergenome.nih.gov/`). A detailed description of the generation and pre-processing of these datasets can be found in the methods section (see 4.24.3). An application of our platform to the KIRC dataset will follow in section 2.4. After the user selects a dataset, it will be loaded into the UI and the interaction window will be unlocked. Figure 1 shows

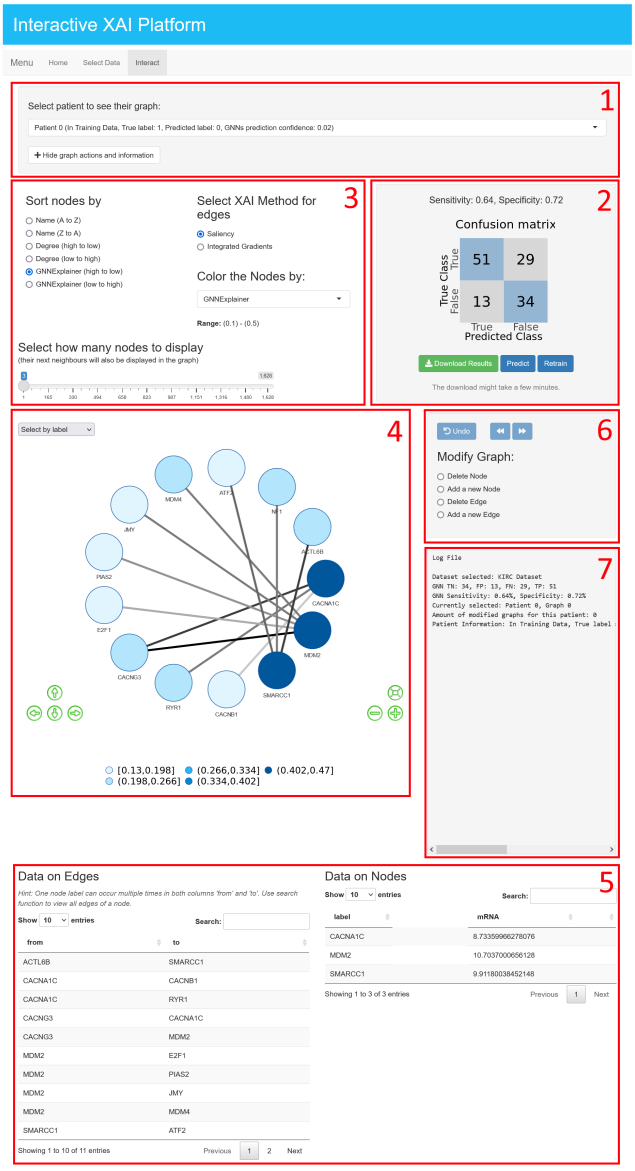

Figure 1: Interact window with its seven components. 1: Drop down selection of a patient graph, with information on the patient graphs, 2: Performance scores of GNN on test data and predict,retrain, and download button, 3: Specifications for graph visualisation (node/edge colouring, node sorting, and how many nodes to display), 4: Graph visualisation, 5: Information on nodes and edges, 6: Graph modifications (node/edge deletion/addition), 7: Log print of actions performed by the user.

the interaction window with its seven main interaction components. The first component is a drop-down

menu that allows the user to select a graph from the dataset and gives the user some first information on the graphs, namely if the graph was used in the training or in the testing of the GNN, what class the graph belongs to, what class the GNN classified the graph as, and the confidence of the GNNs prediction (e.g. probability for a certain class). This enables the user to decide if he/she wants to work on wrongly classified graphs or on training graphs to see if he/she can improve the GNNs performance. The second component shows the confusion matrix and sensitivity and specificity of the GNN on the test-set data [3], [9]; Mutual Information (MI) [21] would be even more appropriate, specially for imbalanced datasets, and will be included in future work together with its analytical description. Here the user can also enact a prediction on the selected graph or a re-training of the GNN on the dataset. Additionally, the user can download the results of his analysis. For this the data tables in the fifth component get saved alongside the calculated relevance values and the log file. Everything is zipped together and downloaded as a whole.
After the user selected a graph the third component allows the user to specify the graph visualisation. Here the user can select how to sort and colour the nodes, by degree, by a selection of relevances from XAI methods or alphabetically according to their labels. Additionally, he/she can select the XAI method used for colouring the edges and how many nodes to display in the graph visualisation. The graph visualisation can be found in the fourth interaction component and it displays the graph based on the user selections alongside a legend for the node colours.
The fifth component consists of information on the edges and nodes displayed in the graph visualisation. This allows the user to get a more detailed view on the graph components. To ask counterfactual questions, the sixth component allows the user to manipulate the graph by adding or deleting nodes or edges. The last component is a log print, that shows the user all of his performed actions.

2.2 Graph visualisation

To help the user gain insights into what parts of the graph data might be important, we have developed a graph visualisation that displays node and edge relevances from a multitude of XAI methods.
There are two attribution that were used, that calculate positive attribution values on the edges [39] and they are provided by the Captum library of Pytorch `https://captum.ai/`. The saliency method computes for each edge the absolute value of the gradient of the non-linear function that the GNN computes w.r.t. the weight of this edge (as defined by the training process). The Integrated Gradients (IG) method is a more sophisticated version of the saliency method, where instead of the absolute value, the integral of the gradient w.r.t a range of learned weight values is used. The GNNExplainer [47] computes the important subgraph G_S of the computation graph G_c [11], [43], [17] of the input graph G with the use of an optimization algorithm. The algorithm is driven by the maximization of the mutual information [21] between the tried substructure (G_S of G_c) w.r.t. the predicted label distribution. The user can choose which XAI methods to use for colouring the edges and nodes. The edges will be coloured in grey-scale based on the relevance values, while the nodes will be coloured in a blue-scale. As to not overwhelm the user by displaying the complete graph we have implemented a sliding bar that allows the user to select how many nodes to display. Alongside those nodes their next neighbours will also be displayed, to allow the user to see the interactions of his selected nodes. The amount of nodes selected will be taken from the top of the list of nodes. This might result in a subset of not relevant nodes being displayed. For that reason we, additionally, allow the user to sort this list of nodes by relevance values, degree or alphabetically. This ensures that the user can analyse the most relevant nodes or if desired the most irrelevant nodes.
Furthermore, we implemented a tooltip that gets shown to the user, when he hovers over an edge or node. If the user hovers over an edge, the tooltip will display information on which nodes the edge connects, as well as, all computed relevances for this edge, labeled by the XAI method that computed them (see Figure 2). When the user hovers over a node, the tooltip displays the node label and degree, alongside all computed relevances for this node, labeled by the XAI method that computed them (see Figure 3). In addition to the tooltip, information on the selected nodes, as well as, information on all of the displayed edges is being shown to the user in two data tables. Here the user can get more details on the nodes and edges by looking at their features.

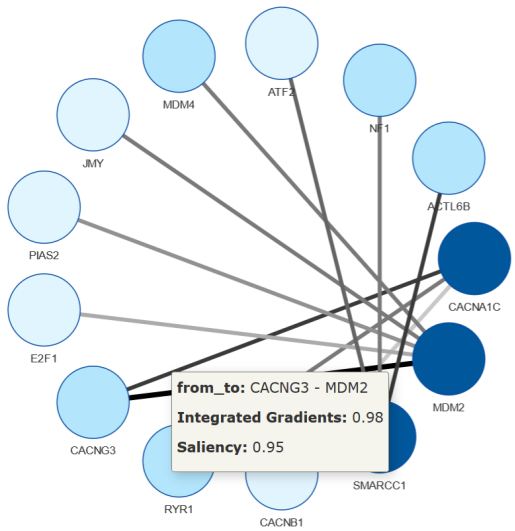

Figure 2: Edge tooltip: Hovering over an edge shows information on connected nodes and relevance values from XAI methods.

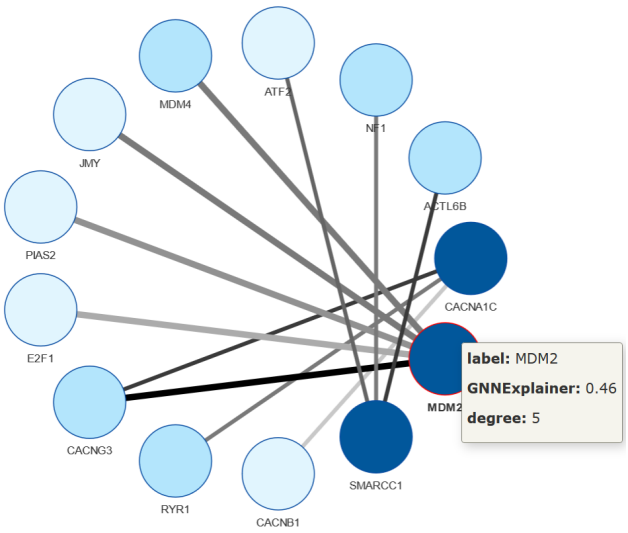

Figure 3: Node tooltip: Hovering over a node shows information on label, degree and relevance values from XAI method.

2.3 Impact of graph manipulation

Besides the graph visualisation, one of our main contributions is the interactive manipulation of the graph data and the visualisation of their impact of this manipulation. This allows the user to directly ask counterfactual questions by changing the data and answer them directly by retraining the GNN and seeing the effects of his changes in the GNN performance and the newly calculated XAI relevances.

In order for the user to ask counterfactual questions such as "What would have been the prediction if this node or edge would not have been in the graph?" or "What would have been the model performance if this node or edge would have been present in the graph?", we have implemented the deletion and addition of nodes and edges. The user can manipulate the graph by adding or deleting a multitude of nodes and edges, as well as, undoing his/her manipulations, and then starting a retraining or prediction to see if the changes resulted in an expected outcome. After predicting or retraining the current state of the graph will always be saved, allowing the user to jump between saved graphs states later on. This ensures that the user can always return to a state before a prediction or retraining, if the changes did not satisfy the user.

To identify the effects the changes have on the GNN performance the user is shown a confusion matrix along with sensitivity and specificity of the GNN on the test data. The confusion matrix allows for a direct comparison of values such as true positives, false positives, true negatives and false negatives, while sensitivity and specificity identify the proportion of correctly identified positive and negative classes. This enables a fast inspection to what degree the model performance was effected by the changes of the user. The GNN performance of the initial training and every retraining will be printed in the log-file, allowing for an easy comparison.

In addition to that, the user can observe the effects of his/her changes by identifying if the predicted label and the confidence in the predicted label have changed. Lastly, after retraining or predicting with the changed graph the user will be able to identify changes in the relevance values. This will allow him/her to ask counterfactual questions of the form "What would have been the most relevant node or edge if this very relevant node or edge would not have been in the graph?". This shows that our interactive XAI platform not only enables the user to understand what consequences his changes have on the GNN performance, but also to understand the effects on the decision-making process of the GNN. By viewing changes in the relevances of nodes and edges, the user can gain a deeper knowledge on the correlations of the nodes and edges in the data and their influence in the decision-making process of the GNN. This should result in an increase in causability, but of course this will need to be measured in future research.

2.4 Application to KIRC dataset

Bastian & Anne-Christin & Andreas: We verified a potential disease module using the GNN-SubNet Python package [24]. The resulting subnetwork consists of four proteins, namely MGAT5B, MGAT5, MGAT4B, and MGAT3 connected by five edges. We have uploaded this module to the here presented XAI Platform in order to conduct a in-depth investigation of the performance and the associated relevance scores. The module has a high sensitivity of 0.95 and a specificity of 0.31 obtained on an independent test data set (127 patients).

We have selected a patient which was classified with a high confidence score. The GNNexplainer assigned the highest relevance score to the node associated with MGATT3. The edge between MGATT3 and MGAT4B is the most important interaction with maximum Saliency value of 1. .... [Server down ...] In a first investigation we verified whether the gene MGAT3 itself, or the edge with MGAT4B is the most relevant part. [Server down] ...

3 Discussion

A vast amount of XAI methods have been developed to help understand the decision-making process of GNNs but due to a lack of causability GNNs are not yet usable as CDSS. Therefore, we developed an interactive XAI platform that increases the causability of XAI methods by allowing the user to ask counterfactual questions. By displaying the relevances computed by XAI methods directly in the graph and letting the user interact with them they can gain a better understanding of the causal relationships in the GNN decision-making process. This human-in-the-loop approach to GNN classification will pave the way for implementation of GNNs in the clinical setting.

There is a plethora of research works that deal with

the automatic generation of counterfactual explanations for graphs [25]. Most of them try to generate the counterfactuals automatically, for example with the use of generative models [19], by just concentrating on edge deletion [18] or by a search driven by a Reinforcement Learning (RL) algorithm [22]. The main goal of those methods is to from a provided graph instance to compute another that has a small graph edit distance and sparsity (modification of fewer features as possible) in the smallest possible time. Preliminary research tries to compare several graph counterfactual generation frameworks under many datasets and evaluation metrics [26], and the result shows no clear winner in all metrics. The methods listed and evaluated do not incorporate solutions where the graph counterfactual generation was driven by human decisions. The evaluation of human-in-the-loop search strategies by similar means as the automated ones is one of the long-term goals of our research.

A similar work to ours that deals with explanations of GNNs via targeted topology perturbation is presented in [37]. The platform "GNNViz" belongs also to the category of Human-in-the-Loop Machine Learning Systems (HITL ML) [14] that was created with the goal that also non-expert users will manipulate graphs and thereby influence the GNN's prediction outcome. They show with the benchmark datasets they are using that with the use of this tool label correlations, unfairness and biases can be uncovered and that a partial comparison with the results of GNNExplainer [47] can be made. The main difference with our work is that we allow and encourage much more changes in the graph as node addition and deletion, as well as nodes and edges features addition and deletion, whereas the GNNViz only considers edge addition and deletion. Furthermore, the recalculation of the prediction result seems to be equivalent to our "predict" method, but there is no functionality that corresponds to our "retrain" method.

The researchers differentiate between three main perturbation strategies: "preserve", which encompasses changes in the topology that leave the prediction unchanged, "promote", which contains all perturbations that affect the prediction results (change the predicted class) and "attack", which are the ones that generate counterfactuals. Their work is not constrained as ours on just accepting changes from the user, but with specially designed losses to compute the edges that can be added or deleted in each of those three modes. The platform sorts and suggests those edges to the user, although it is up to him or her to select that edge (exploration is allowed). Furthermore, there are also filters on the dataset that are used for graph selection and rules by which the graphs will be displayed. The researchers put emphasis on explanations of incorrect predictions, compute and recommend the egde(s) that will flip the prediction towards the correct class. It would be interesting to see how experiments with human users work and what the benefits of this platform is in terms of user acceptance, trust, usability, and causability [15]. The source code will be published upon acceptance and it is going to be an API-based web service.

There are a multitude of features, which we plan to implement in a second version update of our platform. One feature is the addition and removal of whole node and edge features. This is a useful feature from a user standpoint, because some features may be very noisy and thus the user might want to remove that feature completely. But from an implementation standpoint this could be problematic, since the removal of the feature in one graph doesn't automatically remove the feature from the other graphs in the dataset, which could in the end result in many missing values, which the GNN might potentially not be able to handle. Implementing this in such a way that the feature gets automatically removed from every graph would quickly become very complex, because we also want the user to be able to undo his actions as before. For this we would have to make sure that all values are stored somewhere, linked to the correct graph and retrievable. Since this software development direction will additionally need the initialisation of a new GNN architecture, it is going to be more complex. We will need to suggest a GNN architecture automatically by some principles based on empiricism and/or architecture search. Because of this complexity, we currently have not implemented this feature.
Another complex feature that we have currently not implemented is the uploading of personal datasets. Of course in the long run this would be a very desirable feature, since most domain-experts probably have their own datasets that they would like to analyse using our platform. But this would not only involve a large implementation workload but also a huge security risk. Since our platform is running on a university server we cannot allow just any, arbitrarily large, uploads. There would need to be a lot of security implemented around the upload and even if a dataset can be uploaded it needs to have the correct format and data types to work with our platform and the GNN in the backend. We have already identified the intricate parts of this feature and plan to integrate it in the next version update of our XAI platform. Other features that may be implemented in the future will be, selecting between different types of GNN architectures like the Graph Isomorphism Network (GIN) [45], Jumping Knowledge Network [46], implementing more XAI methods, using more datasets that the user can choose from and so on. Lastly, a user management could be implemented. This could involve a register and login/logout option for users. This would enable the possibility to store the user's graphs on the server. Thus, a registered user could retrieve changed graphs at any given time.

4 Methods

4.1 Implementations and workflows

We developed the UI using R version 4.1.0 [27] and R Shiny [29] with important packages such as visNetwork [5] and igraph [6] for the graph visualisation and

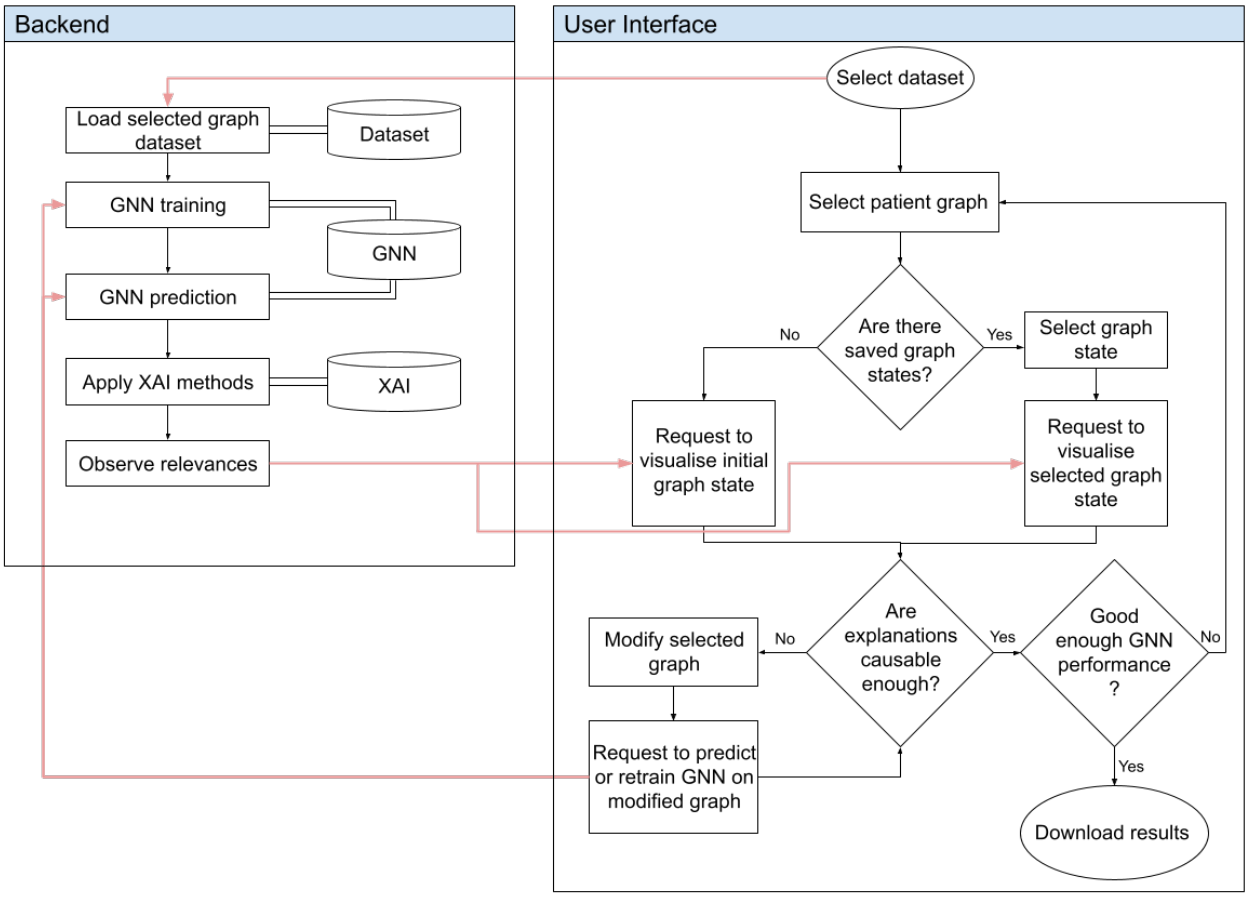

Figure 4: Typical user workflow that shows interaction of the UI (right) with the backend (left) via Flask API (red arrows)

jsonlite [23] and httr [42] for the communication via an API.

A typical user workflow is depicted in Figure 4. Each dataset consists of multiple patient graphs between which the user can switch. After selecting a patient graph the initial graph will be visualised alongside node and edge features as well as relevances from an XAI method. At this point the user can ask counterfactual questions such as "If the most relevant node would have been removed, what would the GNN prediction have been?". He or she can then answer the question by removing or adding nodes or edges and getting a new prediction for this graph from the GNN or initiating a complete retraining of the GNN on the whole dataset with this modified graph. This can be repeated until a sufficient causability is reached. Every modified graph state will be saved after a prediction or retraining. Afterwards, the user can switch to a different patient graph, if they believe the GNN performance is not good enough and continue performing modifications on that graph. Of course, the user could also try and provoke a bad performance out of curiosity on how the explanations for such cases look like. Later, the user can additionally switch between saved patient graph states, in order to allow him or her to return to a previous state, if the modifications did not result in a desired outcome. If the user finds the explanations causable enough, the GNN decision-making process, as well as the GNN performance is good enough, he or she can download the results (node and edge tables with all features and relevance values and the log files) and take screenshots of the graph to save important information.

The backend is implemented using Python. Special care has been taken to ensure the quality of the software. Several unit test-cases have been implemented covering most of the functionality of node and edge addition and deletion. Edge cases concerning degenerate situations like the smallest graph possible that Pytorch Geometric (PyG) can represent, or non admissible actions like re-entering an already existing edge were covered by dedicated Quality Management (QM) Software written in `pytest` `https://pytest.org`. Furthermore, property-based test-cases [13] were written with the use of the package `hypothesis` `https://hypothesis.readthedocs.io/en/latest/` [20]. Those test cases did not explicitly check all the specific changes that are expected to happen in the graph structure after an action, but rather the properties and constraints that have to be fulfilled. For example, removing a node makes the number of edges in the graph lower or equal to the previous value, makes the number of rows of the array that contains the node features smaller by one, and so on. Correspondingly, removing an edge should keep all

information about the nodes alone unchanged; this is an invariant that can be eloquently expressed by the code of a property-based test. Added to that, the selected nodes and edges are not predefined but randomly chosen by the framework. Property-based testing has been proven beneficial for other machine learning problems too [4], [31].

The API for communication between the backend and the UI is implemented in Python using the flask package [10].
Interactions via the API based on the user workflow are shown in Figure 4. When the user selects a dataset, a post-call will be sent to the backend to load the selected dataset. Afterwards, when the user selects a graph and graph state, a get-call will be made to the backend, that will fetch the specified graph. Lastly, when the user wants to initiate a prediction or retraining, post-calls will be sent to the backend, that will initiate the prediction or retraining in the backend. Other API calls for the graph manipulation and retrieval of the performance values are also made, but are not displayed here in order to keep a clearer overview.

4.2 Synthetic dataset generation

We have generated 1000 Barabasi networks comprising 30 nodes and 29 edges. The networks had the same topology, but with varying node feature values. The features of the nodes were randomly sampled from a normal distribution $N(0, 0.1)$. For one half of the networks we have selected two connected nodes to which we assigned values from $N(1, 0.1)$. The GNN classifier identified the two specified classes with high accuracy. However, the ultimate goal here was to detect the selected nodes as the discriminatory factors. Explainable AI methods should uncover these patterns in an algorithmic way. Based on the synthetic data set, we investigate to what extend the human-in-the-loop could improve this process.

4.3 KIRC random dataset generation

To illustrate the usability of the presented platform in we make use of the application presented in [24] for disease subnetwork discovery. The authors model each patient as a PPI network, where nodes are enriched by feature values from DNA Methylation and gene expression. The datasets were downlaoded from The Cancer Genome Atlas (TCGA) database (`https://cancergenome.nih.gov/`), which represents one of the largest collections of multi-omics data sets. It contains molecular and genetic profiles for over 33 different cancer types from 20,000 individual tumor samples [36].

The data sets were harmonized so that for every patient multi-source information was available. Furthermore, we focused on cancer-relevant genes as proposed by [34]. Genes containing missing values at least for one patient were removed from the analyses. The obtained numerical data matrices were normalized using *min-max* normalization. The protein-protein interaction network was retrieved from the STRING database [40].

One class of the networks is cancer-specific and contains multi-modal molecular footprints from KIRC cancer patients. The other class is composed by a random sample from multiple cancer types. Using graph classification with explainable GNNs we infer cancer-specific subnetworks.

5 Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper. This work does not raise any ethical issues.

6 Data availability

Both pre-processed datasets, the synthetic and the real world dataset, can be downloaded in python pickle file format (PKL) through our platform. A PKL file is a file created by pickle, a python module that enables objects to be serialised to files on disk and de-serialised back into the program at run-time. It contains a byte stream that represents the objects. In this format the datasets can be easily loaded into python and analysed further by the user. Inside the PKL file the data is saved as a list, containing every patient graph.

7 Code availability

The code for the UI and the python backend are open-source. They can be found in the following public GitHub repositories:
https://github.com/JacquelineBeinecke/xAI-Shiny-App

8 Abbreviations

- AI - Artificial intelligence
- CDSS - Clinical decision support system
- XAI - Explainable artificial intelligence
- ML - Machine learning
- PPI - Protein-protein-interaction
- GNN - Graph neural network
- UI - User interface
- KIRC - Kidney Renal Clear Cell Carcinoma
- TCGA - The cancer genome atlas

- MI - Mutual information
- IG - Integrated gradients
- PKL - Pickle (file format)

9 Acknowledgements

The Authors declare that there are no conflicts of interests. This work does not raise any ethical issues. Parts of this work have been funded by the Austrian Science Fund (FWF), Project: P-32554 explainable Artificial Intelligence. Parts of this work have received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 826078 (Feature Cloud). This publication reflects only the authors' view and the European Commission is not responsible for any use that may be made of the information it contains.
This work used the Scientific Compute Cluster at GWDG, the joint data center of Max Planck Society for the Advancement of Science (MPG) and University of Göttingen.

References

[1] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLOS ONE*, 10(7):1–46, 2015.

[2] Adrien Bennetot, Ivan Donadello, Ayoub El Qadi, Mauro Dragoni, Thomas Frossard, Benedikt Wagner, Anna Saranti, Silvia Tulli, Maria Trocan, Raja Chatila, et al. A practical tutorial on explainable ai techniques. *arXiv preprint arXiv:2111.14260*, 2021.

[3] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[4] Houssem Ben Braiek and Foutse Khomh. On testing machine learning programs. *Journal of Systems and Software*, 164:110542, 2020.

[5] Almende B.V., Benoit Thieurmel, and Titouan Robert. *visNetwork: Network Visualization using 'vis.js' Library*, 2021. R package version 2.1.0.

[6] Gabor Csardi and Tamas Nepusz. The igraph software package for complex network research. *InterJournal*, Complex Systems:1695, 2006.

[7] Mengnan Du, Ninghao Liu, and Xia Hu. Techniques for interpretable machine learning. *Communications of the ACM*, 63(1):68–77, 12 2019.

[8] Dumitru Erhan, Y. Bengio, Aaron Courville, and Pascal Vincent. Visualizing higher-layer features of a deep network. *Technical Report, Univeristé de Montréal*, 01 2009.

[9] Aurélien Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems.* O'Reilly Media, 2019.

[10] Miguel Grinberg. *Flask web development: developing web applications with python.* " O'Reilly Media, Inc.", 2018.

[11] William L Hamilton. Graph representation learning. *Synthesis Lectures on Artifical Intelligence and Machine Learning*, 14(3):1–159, 2020.

[12] Anne-Christin Hauschild, Chiara Pastrello, Max Kotlyar, and Igor Jurisica. *Protein–Protein Interaction Data, their Quality, and Major Public Databases*, page 151–192. Cambridge University Press, 2019.

[13] Fred Hebert. *Property-Based Testing with PropEr, Erlang, and Elixir: Find Bugs Before Your Users Do.* Pragmatic Bookshelf, 2019.

[14] Andreas Holzinger. Interactive machine learning for health informatics: when do we need the human-in-the-loop? *Brain Informatics*, 3(2):119–131, 2016.

[15] Andreas Holzinger, Georg Langs, Helmut Denk, Kurt Zatloukal, and Heimo Müller. Causability and explainability of artificial intelligence in medicine. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 9(4):e1312, 2019.

[16] Andreas Holzinger, Anna Saranti, Christoph Molnar, Przemyslaw Biecek, and Wojciech Samek. Explainable ai methods-a brief overview. In *International Workshop on Extending Explainable AI Beyond Deep Models and Classifiers*, pages 13–38. Springer, 2022.

[17] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[18] Ana Lucic, Maartje A Ter Hoeve, Gabriele Tolomei, Maarten De Rijke, and Fabrizio Silvestri. Cf-gnnexplainer: Counterfactual explanations for graph neural networks. In *International Conference on Artificial Intelligence and Statistics*, pages 4499–4511. PMLR, 2022.

[19] Jing Ma, Ruocheng Guo, Saumitra Mishra, Aidong Zhang, and Jundong Li. Clear: Generative counterfactual explanations on graphs. *arXiv preprint arXiv:2210.08443*, 2022.

[20] David R MacIver, Zac Hatfield-Dodds, et al. Hypothesis: A new approach to property-based testing. *Journal of Open Source Software*, 4(43):1891, 2019.

[21] David JC MacKay and David JC Mac Kay. *Information theory, inference and learning algorithms.* Cambridge university press, 2003.

[22] Danilo Numeroso and Davide Bacciu. Meg: Generating molecular counterfactual explanations for deep graph networks. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2021.

[23] Jeroen Ooms. The jsonlite package: A practical and consistent mapping between json data and r objects. *arXiv:1403.2805 [stat.CO]*, 2014.

[24] Bastian Pfeifer, Anna Saranti, and Andreas Holzinger. GNN-SubNet: Disease subnetwork detection with explainable graph neural networks. *Bioinformatics*, 38(Supplement_2):ii120–ii126, 2022.

[25] Mario Alfonso Prado-Romero, Bardh Prenkaj, Giovanni Stilo, and Fosca Giannotti. A survey on graph counterfactual explanations: Definitions, methods, evaluation. *arXiv preprint arXiv:2210.12089*, 2022.

[26] Mario Alfonso Prado-Romero and Giovanni Stilo. Gretel: Graph counterfactual explanation evaluation framework. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 4389–4393, 2022.

[27] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2021.

[28] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?": Explaining the predictions of any classifier. *arXiv preprint arXiv:1602.04938*, 2016.

[29] RStudio, Inc. *Easy web applications in R.*, 2013.

[30] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence volume*, 1:206–215, 2019.

[31] Anna Saranti, Behnam Taraghi, Martin Ebner, and Andreas Holzinger. Property-based testing for parameter learning of probabilistic graphical models. In *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*, pages 499–515. Springer, 2020.

[32] Thomas Schnake, Oliver Eberle, Jonas Lederer, Shinichi Nakajima, Kristof T Schütt, Klaus-Robert Müller, and Grégoire Montavon. Higher-order explanations of graph neural networks via relevant walks. *arXiv preprint arXiv:2006.03589*, 2020.

[33] Roman Schulte-Sasse, Stefan Budach, Denes Hnisz, and Annalisa Marsico. Integration of multiomics data with graph convolutional networks to identify new cancer genes and their associated molecular mechanisms. *Nature Machine Intelligence*, 3:1–14, 06 2021.

[34] Roman Schulte-Sasse, Stefan Budach, Denes Hnisz, and Annalisa Marsico. Integration of multiomics data with graph convolutional networks to identify new cancer genes and their associated molecular mechanisms. *Nature Machine Intelligence*, 3(6):513–526, 2021.

[35] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *CoRR*, abs/1312.6034, 2014.

[36] Indhupriya Subramanian, Srikant Verma, Shiva Kumar, Abhay Jere, and Krishanpal Anamika. Multi-omics data integration, interpretation, and its application. *Bioinformatics and biology insights*, 14:1177932219899051, 2020.

[37] Yi Sun, Abel Valente, Sijia Liu, and Dakuo Wang. Preserve, promote, or attack? gnn explanation via topology perturbation. *arXiv preprint arXiv:2103.13944*, 2021.

[38] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 3319–3328. JMLR.org, 2017.

[39] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR, 2017.

[40] Damian Szklarczyk, Annika L Gable, Katerina C Nastou, David Lyon, Rebecca Kirsch, Sampo Pyysalo, Nadezhda T Doncheva, Marc Legeay, Tao Fang, Peer Bork, et al. The string database in 2021: customizable protein–protein networks, and functional characterization of user-uploaded gene/measurement sets. *Nucleic acids research*, 49(D1):D605–D612, 2021.

[41] Minh N Vu and My T Thai. Pgm-explainer: Probabilistic graphical model explanations for graph neural networks. *arXiv preprint arXiv:2010.05788*, 2020.

[42] Hadley Wickham. *httr: Tools for Working with URLs and HTTP*, 2020. R package version 1.4.2.

[43] Lingfei Wu, Peng Cui, Jian Pei, Liang Zhao, and Le Song. Graph neural networks. In *Graph Neural Networks: Foundations, Frontiers, and Applications*, pages 27–37. Springer, 2022.

[44] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32:4–24, 2019.

[45] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[46] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *International Conference on Machine Learning*, pages 5453–5462. PMLR, 2018.

[47] Rex Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnnexplainer: Generating explanations for graph neural networks. *Advances in neural information processing systems*, 32:9240, 2019.

[48] Hao Yuan, Jiliang Tang, Xia Hu, and Shuiwang Ji. Xgnn: Towards model-level explanations of graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 430–438, 2020.