

Network Module Detection from Multi-Modal Node Features with a Greedy Decision Forest for Actionable Explainable AI

Bastian Pfeifer^{a,*}, Anna Saranti^a, Andreas Holzinger^{a,b}

^a*Medical University Graz, Austria*

^b*Alberta Machine Intelligence Institute, Canada*

Abstract

Network-based algorithms are used in most domains of research and industry in a wide variety of applications and are of great practical use. In this work, we demonstrate subnetwork detection based on multi-modal node features using a new Greedy Decision Forest for better interpretability. The latter will be a crucial factor in retaining experts and gaining their trust in such algorithms in the future. To demonstrate a concrete application example, we focus in this paper on bioinformatics and systems biology with a special focus on biomedicine. However, our methodological approach is applicable in many other domains as well. Systems biology serves as a very good example of a field in which statistical data-driven machine learning enables the analysis of large amounts of multi-modal biomedical data. This is important to reach the future goal of precision medicine, where the complexity of patients is modeled on a system level to best tailor medical decisions, health practices and therapies to the individual patient. Our glass-box approach could help to uncover disease-causing network modules from multi-omics data to better understand diseases such as cancer.

Keywords: Networks, Decision Forest, Feature selection, Multi-Modal, trustworthy AI

*Corresponding author

Email address: bastian.pfeifer@medunigraz.at (Bastian Pfeifer)

Highlights

- We present a novel tree-based subnetwork detection algorithm
- Our approach supports multi-modal node features of arbitrary dimensions
- Moreover, our approach allows for multi-graphs as an input

1. Introduction

Network-based algorithms are utilized in most areas of research and industry. They find their application in virtually all application areas, from agriculture to zoology, and from social sciences to bioinformatics and especially systems biology [1], [2]. Networks are very important to help to solve many problems in decision making and knowledge discovery [3]. Such networks can represent many real-world phenomena and are technically described by graphs, which provide a unifying abstraction by which such real-world networks can be represented, explored, predicted and discovered [4]. Most importantly, such structures help to make complex phenomena re-traceable, transparent, interpretable and thus explainable to human experts. What we need in the future are context-adaptive methods, i.e. systems that construct contextual explanatory models for classes of real-world phenomena. This is a goal of so-called explainable AI [5], which is actually not a new field; rather, the problem of explainability is as old as AI itself. While the rule-based approaches of early AI were comprehensible "glass box" approaches, at least in narrow domains, their weakness lay in dealing with the uncertainties of the real world [6]. Here Actionable AI may be of help, i.e. fostering features that build trust by bringing decision-analytic perspectives into AI [7].

In this work, we focus as example application on systems biology with a particular emphasis on biomedicine. Recent developments in bioinformatics and machine learning have made it possible to analyze huge amounts of biomedical data, paving the way for future precision medicine. A grand goal of precision medicine is in modeling the complexity of patients to tailor medical decisions,

health practices and therapies to the individual patient, which calls for efficient computational methods, algorithms and tools to discover knowledge and to interactively gain insight into the data [8]. Network-based approaches help uncover disease-causing interactions between genes to better understand diseases such as cancer [9], [10]. For instance, network analyses are performed for protein-protein co-expression networks as well as for metabolomics data through metabolomic pathway analyses.

Recent evidence suggests that complex diseases, such as cancer, need to be studied in a multimodal feature space of different biological entities (multi-omics), as diverse components contribute to a single outcome [11], [12]. Multi-omics clustering, for instance, is successfully utilized to detect disease subtypes, that is patient groups with similar molecular characteristics. Recently developed multi-omics clustering methods include SNF [13], PINSplus [14], and HC-fused [15].

Another typical goal is to discover a set of disease-causing genes to efficiently monitor disease progression. These so-called biomarkers are often discovered from multi-modal omics data sets and here it is important to allow physicians to interactively intervene, read out corresponding patterns from the data and adjust their treatment accordingly. Feature selection (FS) algorithms are often used to detect such patterns. One popular family of FS algorithms is based on the random forest (RF) classifier. RFs can efficiently handle heterogeneous data types, don't need normalization of their values [16], computes feature importances and are therefore particularly suitable for the multimodal case. One example of a widely used RF-based feature selector is the Boruta algorithm [17]. The heuristic introduced by the authors quantifies the importance of a feature by the loss of accuracy of classification caused by randomly permuted probes (*shadow features*) of the original feature set. Recent work combines the Boruta algorithm with Shapley values [18]. Another RF-based feature selection algorithm is introduced in [19]. The authors developed a regularized random forest approach that penalizes the selection of a new feature for splitting a node when its information gain is similar to the features used in previous splits.

However, most of these methods treat genes/features independently or do not account for any dependencies between features. Contrarily, approaches which do account for dependencies between genes are tailored towards the detection of network modules with correlated features only (like in co-expression analyses). They typically utilize community detection algorithms [20] or unsupervised clustering. Disease-causing genes, however, may function in a multivariate manner without necessarily being correlated.

Most recently, the machine learning community has developed deep learning techniques which directly can be applied to networks comprising edge and node features of arbitrary dimensions. Graph neural networks (GNNs) [21], [22] will certainly have a big impact on systems biology research. A major difficulty with these approaches is that they are considered black-box models. The decisions they make cannot be traced back precisely, therefore medical doctors may not trust decisions made by black-box models. To address this shortcoming, first explainable GNN methods are available and are currently under strong development. Examples include GNNExplainer [23], PGExplainer [24], and GNN-LRP [25]. GNNExplainer provides *local* explanations for predictions of any graph-based model. It can be applied to node classification as well as graph classification tasks. PGExplainer is a parameterized modification of the GNNExplainer. In contrast to the GNNExplainer, it provides explanations on a model level; especially useful for graph classification tasks. The GNN-LRP approach is derived from higher-order Taylor expansions based on Layer-wise relevance (LRP). It explains the prediction by extracting paths from the input to the output of the GNN model that contributes the most to the prediction. These paths correspond to *walks* on the input graph. GNN-LRP was developed for explanations on the node level but was recently modified to also work for graph classification in a special application set-up [26].

The aforementioned methods may facilitate the discovery of disease-causing regions within the networks, but not directly can be used for network module selection purposes. Explainers on GNNs usually highlight the importance of edges, nodes or walks within the network, but they do not rank whole subnetworks by

their importance. Also, methods for explanations are often black-box models themselves, and thus may not be trustworthy either. In addition, standard GNN architectures cannot handle multi-graphs which may hinder the analysis for incomplete data, which is often the case in the biomedical domain.

At the same time, only a few FS methods are applicable on network structured input data. Some developments in this direction are realized within the R-packages `glmnet` [27] and `know-GRRF` [28]. Both methods use a regularization term to incorporate network topology into the feature selection process. The underlying algorithms, however, do not function on the network itself.

Here, we present a greedy decision forest for the selection of network modules. The proposed algorithm directly functions on a network by sampling node features by a random walk. A set of decision trees are derived from the visited nodes, forming the decision forest. A greedy process on the selection of decision trees is initiated, and the algorithm converges to a set of subnetworks. In the following we will use the term "network module" and "subnetwork" interchangeably.

The proposed methodology is a glass-box approach [29] and naturally handles multi-modal data with a high degree of interpretability, thereby perfectly suited for multi-omics types of analyses in the biomedical domain. In principle, the addressed problem could also be solved by using GNNs and explainable GNN for subnetwork detection. However, results may not be as interpretable as the glass-box approach we provide. Moreover, it encourages both the domain expert and the AI system’s designer to pursue actions for the improvement of performance and explainability.

2. Proposed approach

The core functionality of our proposed network module detection approach includes a tree building process derived from a network $G = (V, E)$. Each node $\mathbf{v} \in V$ comprises node feature vectors $f(\mathbf{v})$ of arbitrary dimensions (see Fig. 1). To build a single tree we randomly select a node $\mathbf{v} \in V$ located on the network

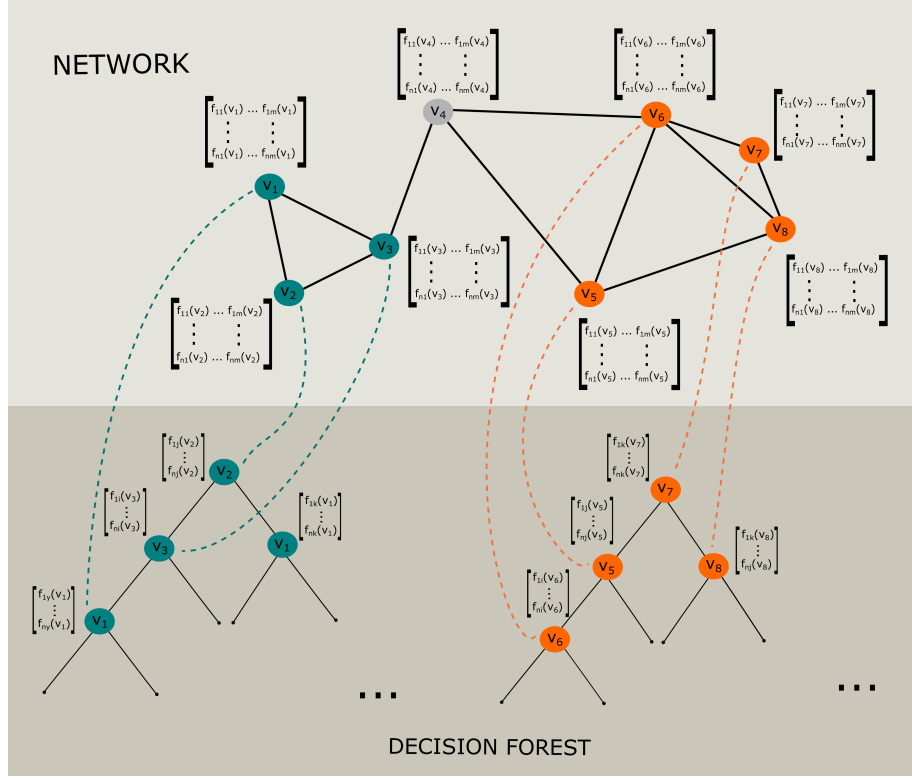


Figure 1: Deriving a Decision Forest from a network with multi-modal node feature vectors. Displayed is the network at the top and the derived decision forest at the bottom. Each node of the network comprises m node feature vectors of length n . Each random walk creates a *mtry* set of feature vectors from visited nodes, which are then used to build a decision tree.

G . From that node a random walk is initialized. The depth of this random walk is set to $\sqrt{\#V}$, which is the squared root of the total number nodes. This value is frequently used as a rule-of-thumb, and a default in R-packages such as in ranger [30] and randomForest [31]. It is the so-called *mtry* parameter to specify the number of features a decision tree is build from. In our case, features of visited nodes by the random walk will be in the *mtry* set. The above procedure is repeated until *ntree* decision trees are generated. Figure 1 illustrates this process. A network is shown comprising eight nodes. Each of these nodes has m node feature vectors of length n , where n is the number of samples and m is the

number of modalities. Two random walks of depth 4 are highlighted. Each of these walks forms a Decision Tree (blue and orange nodes). It should be noted, that the random walk may visit the exact same node multiple times. So is the case for the first random work. The derived Decision Tree contains two times the node v_1 . The corresponding features are used twice for splitting a tree node (Fig.1: blue Decision Tree). Node v_4 was not captured by the random walks.

From these decision trees we start the proposed greedy algorithm for network module selection. At each greedy step, $ntree$ decision trees are generated, but the selection of network nodes and their corresponding features depend on the outcome of previous iterations. High performing trees remain in the $ntree$ set of trees while low performing trees and their corresponding nodes are eliminated.

In the following, we will introduce necessary notations and describe the greedy algorithm in great detail.

2.1. Network module selection using a greedy decision forest

We define a Decision Tree classifier as $T(x; \Theta, X)$, where Θ consists of a set of split rules and X are the variables/features used for splitting. Given an input vector x , $T(x; \Theta, X)$ assigns a given data point to a specific class. An ensemble of Decision Tree classifier is called Decision Forest (DF). It is defined as $\{T_k(x, \Theta_k, X_k), k = 1, \dots, ntree\}$, where X_k is a set of randomly selected features from the input feature space the k -th Decision Tree T_k is based on.

Let us further assume a graph $G = (V, E)$ is specified. The nodes $v \in V$ represent the features in X and the edges $(v_i, v_j) \in E$ are reflecting any kind of dependency of these features. This graph for example could be a knowledge graph, which connects node features which have any kind of domain-specific relationship. Like in Protein-Protein networks, where functionally interacting genes are connected.

Here, we propose to restrict the samples X_k to be neighboring nodes within graph G . This regularization ensures functional related features to be located on the same Decision Tree. As a consequence, classifications made by this tree might be more reliable and interpretable for the domain expert, and at the same

time may also safeguard for overfitting.

Accordingly, the proposed Decision Forest is defined as $\{T_k(x, \Theta_k, X_k^G), k = 1, \dots, ntree\}$, where X_k^G is a set of features determined by a random walk on graph G . Starting from that Decision Forest we execute the proposed greedy steps for tree-based module selection (see Algorithm 1).

Algorithm 1: Greedy Decision Forest

```

1  Given a Decision Forest:  $\{T_k(x, \Theta_k, X_k^G)\}$ ;
2  Given a graph:  $G = (V, E)$ ;
3   $k = \{1, \dots, ntree\}, t = 1$ ;
4   $mtry_k = \sqrt{\#V}$ ;
5   $X_k^G[t = 1] = X_k^G$ ;
6  while  $t \leq niter$  do
7      for  $k \leftarrow 1$  to  $ntree$  do
8           $Perf(T_k[t]) = \text{Performance of } T_k(x; \Theta_k, X_k^G)$ ;
9          if  $Perf(T_k[t]) \leq Perf(T_k[t - 1])$  then
10              $T_k[t] = T_k[t - 1]$ ;
11              $X_k^G[t] = X_k^G[t - 1]$ ;
12         else
13              $mtry_k[t] = -$ ;
14              $X_k^G[t] = \text{RandomWalk}(G|X_k, mtry_k[t])$ ;
15         end
16     end
17     Sample ntree trees according to  $Perf\{T_k[t]\}$ ;
18      $t++$ ;
19 end

```

For each greedy step t we calculate the performance for all $k \in \{1, \dots, ntree\}$ Decision Trees (Algorithm 1: line 8). Here, we use ROC-AUC as a performance estimate, but this could be replaced by any type of performance measure such as accuracy (in the case of a balanced dataset) or mutual information [32]. In case the performance of the k -th Decision Tree $Perf(T_k[t])$ at greedy step t is lower

than the performance of the Decision Tree from the previous iteration ($t - 1$), the suggested Decision Tree and the corresponding node features are dropped. In case the Decision Tree has better performance, a random walk on a subgraph, specified by the features in $X_k[t]$ at greedy step t , is initialized (Algorithm 1: line 14). Note, the depth of this walk is now decreased by one (Algorithm 1: line 13). Thus, the proposed algorithm aims for a minimal set of tree features while not suffering sufficient loss of performance. Finally, after updating the Decision Trees, we sample a new set of n_{tree} Decision Trees according to their performance values in order to initiate a selective process (Algorithm 1: line 17). The algorithm repeats the aforementioned procedure and terminates after n_{iter} greedy steps. The selected modules are represented by the node features X_k^G at iteration $t = n_{iter}$.

The feature space X can naturally be extended to a multi-modal representation. It may simply consist of a set of matrices instead of a single matrix, representing the samples as rows and the features as columns. Figure 1 indicates this scenario. Each node is represented by a feature matrix. Each column of this matrix is a feature vector from a different modality, all associated with the same node. Consequently, the tree building process is free to pick features from any of these modalities for splitting a tree node, as long it is associated with the same network node.

Overall, Decision Forests are particularly suited for the multi-modal case, because DFs are scale invariant. There is no need to normalize data prior to execution and thus they provide an ideal framework for the analysis of heterogeneous input data. At the same time the Decision Forest is a glass-box approach making it straightforward to re-trace the contribution of each modality to the final outcome.

2.2. Network module importance scores

We compute an importance score for each of the selected modules. The proposed estimate is based on the network module performance $Perf(T_k)$ at the n_{iter} greedy step, which is the performance of the Decision Tree associated with that

specific module. However, the performance $Perf(T_k)$ of a module does not depend on the number of features used by the derived Decision Tree. Thereby, redundant features do not affect the performance in a negative way. Here, we aim to report on the smallest possible module with maximal performance. Thus, an additional importance measure is needed to account for the importance of the actual edges forming the module. Edge importance is calculated as

$$IMP_e((\mathbf{v}_i, \mathbf{v}_j) \in E) := \sum_t^{niter} \sum_k^{ntree} Perf(T_k(\{\mathbf{v}_i, \mathbf{v}_j\} \in X_k^G)[t]). \quad (1)$$

The above equation assigns the performance of a module X_k^G to the edges $(\mathbf{v}_i, \mathbf{v}_j)$ forming it. This is done for all trees grown during the greedy steps t . Consequently, we not only account for the performances but also for the number of times an edge is part of the *ntree* set. We expect less important edges to be purified during the selective greedy process.

The normalized edge importance of a module is calculated as

$$\overline{IMP_e}(X_m^G) := \frac{\sum_{i,j} IMP_e((\mathbf{v}_i, \mathbf{v}_j) \in X_m^G)}{\#(\mathbf{v}_i, \mathbf{v}_j) \in X_m^G}, \quad (2)$$

where $m \in \{1, \dots, ntree\}$ and $(\mathbf{v}_i, \mathbf{v}_j) \in E$, where $i, j \in \{1, \dots, \#V\}$. The nominator reflects the cumulative edge importance score within the m -th module X_m^G , and is normalized by the number of edges forming that module.

Finally, module importance is calculated as

$$IMP_m(X_m^G) := \overline{IMP_e}(X_m^G) + Perf(T_m), \quad (3)$$

where $Perf(T_m)$ is the performance of the m -th DT associated with the module X_m^G . We utilize the IMP_m to rank the obtained modules X_m^G after *niter* greedy steps.

2.3. Node feature importance scores

Since our proposed Greedy Decision Forest is a glass-box approach we could easily retrace from which modality the features were taken to form the selected

modules. The importance of these features can be calculated by popular tree-based importance measures, such as the Gini impurity index [33]. The Gini index at tree node \mathbf{v}^t is

$$Gini(\mathbf{v}^t) = \sum_{c=1}^C \hat{p}_c^{\mathbf{v}^t} (1 - \hat{p}_c^{\mathbf{v}^t}), \quad (4)$$

where $\hat{p}_c^{\mathbf{v}^t}$ is the proportion of samples belonging to class c at tree node \mathbf{v}^t . The Gini information gain obtained by a feature X_i for splitting node \mathbf{v}^t is the difference between the Gini impurity at node \mathbf{v}^t and the weighted average of impurities at each child node of \mathbf{v}^t . The Gini information gain is defined as

$$Gain(X_i, \mathbf{v}^t) = Gini(\mathbf{v}^t) - w_L Gini(\mathbf{v}_L^t, X_i) - w_R Gini(\mathbf{v}_R^t, X_i), \quad (5)$$

where \mathbf{v}_L^t and \mathbf{v}_R^t are the left and right child nodes of \mathbf{v}^t , respectively, and w_L and w_R are the proportions of c -class instances assigned to the left and right child nodes. At each node \mathbf{v}^t , a set of features (the *mtry* set), and the feature with the maximum $Gain(X_i, \mathbf{v}^t)$ is used for splitting the node. We calculate the importance score of a feature within a detected module X_m^G as

$$IMP_f(f(\mathbf{v}_i \in V)) = \sum_t^{niter} \sum_k^{ntree} Gain(f(\mathbf{v}_i) \in X_k^G)[t], \quad (6)$$

and is normalized as

$$\overline{IMP}_f(f(\mathbf{v}_i) \in X_m^G) = \frac{IMP_f(f(\mathbf{v}_i) \in X_m^G)}{niter \cdot ntree}, \quad (7)$$

where $f(\mathbf{v}_i)$ refers to a feature vector associated with a network node \mathbf{v}_i . Similar to the calculation of the edge importance IMP_e , we calculate the importance of a feature by accounting for the number of times it is a member of the *ntree* set of modules during the greedy process.

3. Synthetic data sets

We have exercised our approach on synthetic Barabasi networks [34]. Barabasi networks were developed to reflect real-world biological networks, where subsets

of nodes (e.g genes) are strongly connected and organized as communities. They are scale-free networks and are generated using a preferential attachment mechanism. For each simulated network we randomly selected four connected nodes $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$, and \mathbf{v}_4 , which together with their associated edges form a functional module. Each node is linked to a feature vector containing binary feature values of 1000 samples. The feature values of all nodes are uniformly distributed, while the selected nodes have the following functional relationship

$$Module(V, E) := [f(\mathbf{v}_1) \wedge f(\mathbf{v}_2)] \oplus [f(\mathbf{v}_3) \wedge f(\mathbf{v}_4)], \quad (8)$$

where $f(\mathbf{v})$ is the feature vector associated with node \mathbf{v} .

We have generated a Barabasi network comprising $\#V = 30$ nodes with a power of the preferential attachment of 1.2 using the R-package igraph [35]. The selected nodes of the functional module are $\mathbf{v}_{12}, \mathbf{v}_{15}, \mathbf{v}_{16}$, and \mathbf{v}_{30} and form a sub-graph. Figure 1 shows the resulting Barabasi Network and the top-1 module correctly verified by our approach (highlighted in red). The thickness of the edges reflects the importance of the edges determined by IMP_e .

Furthermore, Table 1 lists the selected modules sorted by their module importance scores (IMP_m). Overall, six modules were detected. As can be seen from Table 1, the first three modules have the same module performance ($Perf(T_m) = 1$). Their associated DTs perform equally well with an accuracy of $AUC = 1$. However, these modules differ with regard to their average edge importance ($\overline{IMP_e}$). Redundant node features 17 (in case of module 2) and feature 20 (in case of module 3) decrease the average edge importance scores. If we would judge the detected modules based on their module performance $Perf(T_m) = 1$ only, it may result in top-ranked modules including unimportant features. On the other hand, if we would exclusively rely on the edge importance $\overline{IMP_e}$, we may miss important features. In our example this would lead module $\{12, 16, 30\}$ to be ranked second (see Table 1). As a consequence, we miss relevant node features associated with node v_{15} . The combination of both, $\overline{IMP_e}$ and $Perf(T_m)$ provides a minimal set of most important node features.

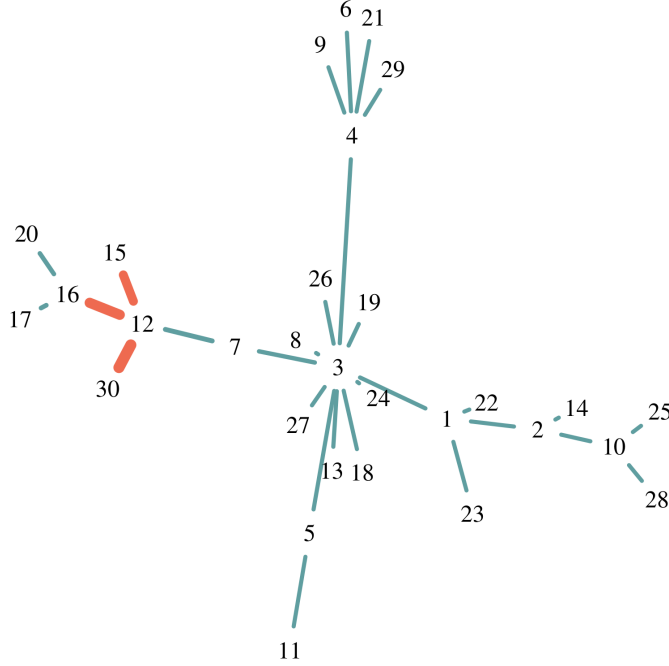


Figure 2: Simulated Barabasi network. The correctly identified subnetwork ($\{\mathbf{v}_{12}, \mathbf{v}_{15}, \mathbf{v}_{16}, \mathbf{v}_{30}\}$) is highlighted in red. The thickness of the edges is reflected by the IMP_e edge importance values. The performance of the detected module is $Perf(T_m) = 1$, and the normalized edge importance score is $\overline{IMP_e} = 0.67$. The overall importance score thus is $IMP_m = 1.67$.

In a further analysis we studied the effect of the parameter *niter* on the performance of our module selection method. We have generated a Barabasi network comprising 50 nodes and executed the module selection procedure 100 times. For each run the topology of the barabasi network is the same, but we varied the selected nodes as well as the binary feature vectors. We report on the coverage, which in our case is the number of times the module of interest is ranked first. In addition, we report on the number of unique modules detected after *niter* greedy steps, and the overall performance of the greedy Decision Forest classifier. Results are shown in Fig. 3.

The network module to be detected is consistently ranked first (*coverage* = 1)

Table 1: Detected modules from the Barabasi network shown in Figure 2.

RANK	MODULES	\overline{IMP}_e	$Perf(T_m)$	IMP_m
1	{12, 15, 16, 30}	0.67	1	1.67
2	{12, 15, 16, 17, 30}	0.57	1	1.57
3	{12, 15, 16, 20, 30}	0.56	1	1.56
4	{12, 16, 30}	0.64	0.81	1.45
5	{12, 15, 30}	0.59	0.80	1.39
6	{12, 16, 17, 30}	0.51	0.80	1.31

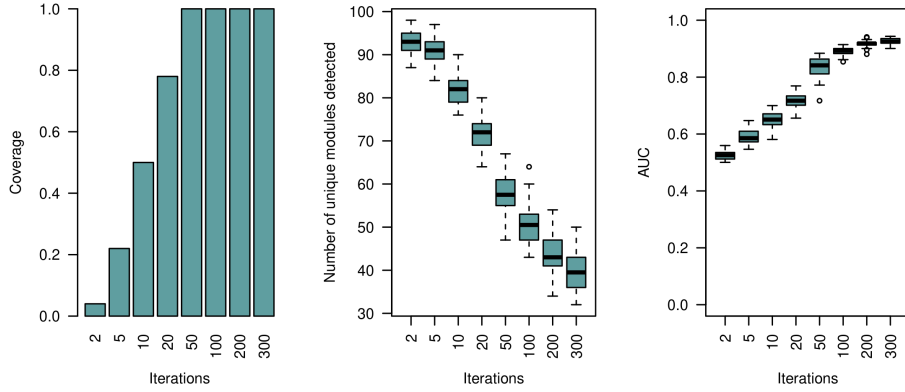


Figure 3: *Single-Modal* simulation results on Barabasi networks. We varied the number of greedy iterations ($n.iter$) and calculated the number of times the selected module is ranked first according to our proposed module importance score IMP_m (left panel). Displayed are the number of unique modules within the $n.tree$ module set after termination of the greedy process (middle panel). The performance of the greedy Decision Forest classifier is shown in the right panel. For each run the topology of the Barabasi network, the feature values, and the selected subnetwork is the same. The simulated Barabasi networks comprise 50 nodes.

after 50 greedy iterations. Also, the performance of the Decision Forest classifier stabilizes at $AUC = 0.9$ after 50 greedy runs. The number of unique modules decreases with the number of iterations and starts to converge at 200 iterations. We observed that the lower-ranked modules are forming larger graphs which all include the module ranked first as a sub-graph (similar to what can be seen

from Table 1).

Each node may include heterogeneous multi-modal features. This typically the case in integrative multi-omic studies. Possible features may include gene expression levels, micro-RNA, and DNA Methylation data for the same set of patients. In order to test the general applicability of our approach for these type of applications we define a multi-modal XOR module on a Barabasi network as follows:

$$\text{Module}(V, E) := [f_a(\mathbf{v}_1) \wedge f_b(\mathbf{v}_2)] \oplus [f_a(\mathbf{v}_3) \wedge f_b(\mathbf{v}_4)], \quad (9)$$

where $f(\mathbf{v})$ is the feature vector associated with node \mathbf{v} , and a refers to the features of the first modality and b to the second.

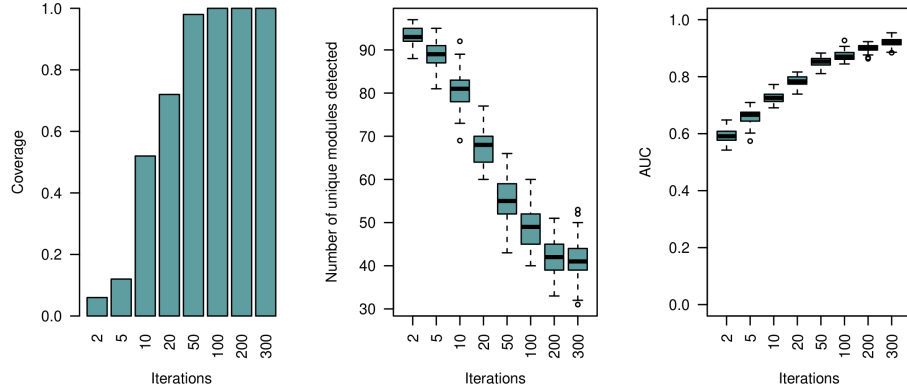


Figure 4: *Multi-Modal* simulation results on Barabasi networks. We varied the number of greedy iterations ($n.iter$) and calculated the number of times the selected module is ranked first according to our proposed module importance score IMP_m (left panel). Displayed are the number of unique modules within the $n.tree$ module set after termination of the greedy process (middle panel). The performance of the greedy Decision Forest classifier is shown in the right panel. For each run the topology of the Barabasi network, the feature values, and the selected subnetwork is the same. The simulated Barabasi networks comprise of 50 nodes.

Fig. 4 shows the results when the barabasi network structure is the same for all iterations. The results are very similar to what we have seen from the single-modal case (Fig. 3). This observation indicates that the proposed algorithm

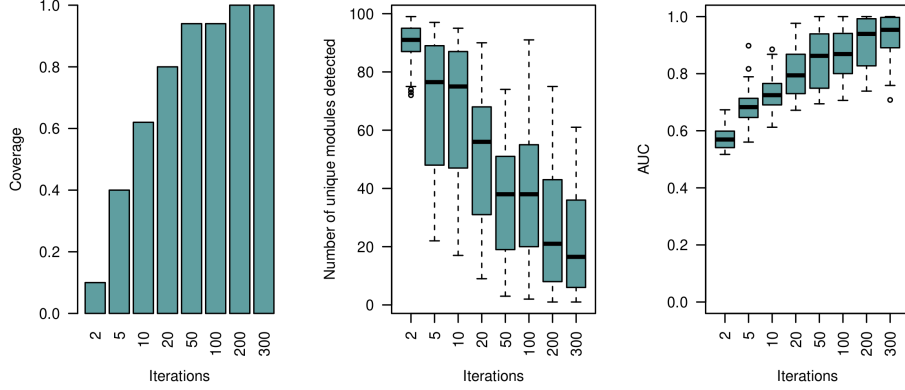


Figure 5: *Multi-Modal* simulation results on *variable* Barabasi networks. We varied the number of greedy iterations ($n.iter$) and calculated the number of times the selected module is ranked first according to our proposed module importance score IMP_m (left panel). Displayed are the number of unique modules within the $n.tree$ module set after termination of the greedy process (middle panel). The performance of the greedy Decision Forest classifier is shown in the right panel. For each run the topology of the Barabasi network, the feature values, and the selected subnetwork *varies*. The simulated Barabasi networks comprise of 50 nodes.

is fully capable of detecting important sub-networks even when the important information is distributed across multiple modalities.

In a additional investigation, we varied the topology of the barabasi networks, as well as their corresponding binary node features (Fig. 5). Results confirm that the proposed Greedy Decision Forest can be efficiently applied to multi-modal feature inputs. The coverage is comparably high and is 1 starting with 200 greedy iterations (see Fig. 5). However, both, the *AUC* performance values and the number of detected unique modules have a wider distribution (Fig. 5). We conclude, the performance of our proposed algorithms depends on the topology of the barabasi networks (see Fig.4 versus Fig. 5).

4. Application to multi-omics pan-cancer data

We showcase the applicability of our approach on a Protein-Protein Interaction Network (PPI) for the detection of disease modules. The PPI network

was retrieved from the STRING database [36]. We only kept high confidence interactions (a combined score within the upper 0.95 quantile). We filtered for relevant cancer genes as specified in [37]. This results in 13,218 relevant genes and 6,926,452 edges. We enriched each node of the PPI network by multi-modal genomic features from kidney survival and non-survival patients. Multi-Modal omic features were extracted from <http://linkedomics.org/> [38]. We selected gene expression (mRNA) and DNA Methylation data for the same set of patients.

We randomly initiated 100 random walks on the PPI network. The depth of the random walks, capturing the nodes for tree building, was set to an initial value of 30. Thus, we were interested in disease modules comprising less than 30 genes. The derived Decision Trees we then let evolve over 300 greedy iterations. The resulting DF classifier after termination of the greedy process had an overall accuracy of $AUC = 0.77$. A rather low value was expected, because the detection of network modules causing a death outcome in patient survival analysis is a hard task. In future applications we will include features from a wide range of biological entities, including medical data, which we believe will increase the overall performance.

After termination of the greedy process all detected modules/trees include about 10-15 genes. The best performing module comprises 11 genes and has a module accuracy of $AUC = 0.75$. The corresponding cumulative edge importance IMP_e is 1.59. Thus, according to our proposed module importance measure IMP_m it is ranked 6th. The top ranked module include the genes SMNDC1, CTNNBL1, CPSF4, POLR2L, LSM6, HNRNPH1, EIF4A3, MAGOHB, RBM5, HNRNPK, and CSTF2T. It has a module accuracy of $AUC = 0.70$ and a cumulative edge importance of 2.18. The detected module is shown in Figure 5. It comprises 52 edges. The importance of the edges is visually reflected by their thickness. The overall high module edge importance is mainly due to the edge connections with the HNRNPH1 gene. Its multi-variate interactions with the CPSF4, POLR2L, and LSM6 genes have by far the highest edge importance scores in that specific module. HNRNPH1 is a multi-functional protein

and is involved many mechanisms within the nucleus. It is a known cancer relevant protein and has been previously reported to influence tumor formation and growth [39].

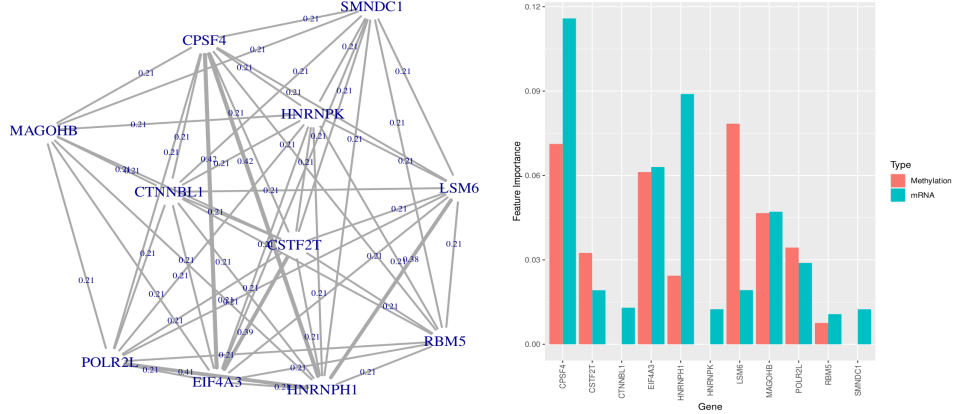


Figure 6: *Disease module for classification of kidney cancer survival and non-survival patients*

The gene expression features of the HNRNPH1 and the CPSF4 genes have the highest importance values, which may further point to a special disease effect caused by the interaction between these two genes. The highest impurity score is reported for the CPSF4 gene. An interesting observation can be made for the LSM6 gene. The methylation of that gene seems to play an important role within the detected module due to its high importance scores. The corresponding low gene expression importance scores may indicate that the LSM6 gene provides some important regulative capabilities within the disease causing module.

5. Discussion

In this work, we have proposed a tree-based network module selection algorithm. It can be applied to the scenario where each node is linked to a specific feature vector and any kind of dependency between these features are reflected by the edges of a given network. We have tested our approach on synthetically generated Barabasi Networks. We could show that our approach can be

naturally applied to the case of multi-modal input features, where each node comprises feature vectors from multiple modalities. Also, our method can be applied to multi-graphs, which is an important feature since missing data is a well known bottleneck within the biomedical domain.

We showcase the applicability of our approach on a protein-protein interaction networks, where each node is enriched by multi-omics features, to detect disease causing modules. We believe we have developed a useful framework which may find wide applications in the future, in all areas of research, but especially within the bio-medical domain. This is because the greedy decision forest for module selection is a glass-box approach and thereby decisions it makes, or modules it detects, are traceable by design. Deep Learning methods on graphs may solve similar tasks in the future, but are black-box models per design. Explainable AI methods are further needed to uncover the underlying decision path. In fact, our proposed greedy Decision Forest may act as a baseline for further developments in that field solving similar problems by the deep learning community.

Also, our methodology might be modified to a explainable GNN method. Instead of increasing the performance of the DF classifier with regard to a target vector, our method may be adjusted to converge to the predicted outcomes of a GNN classifier. The glasss-box greedy DF classifier would serve as a global explainer for black-box models, while the Decision Trees could provide local explanations. This is similar to what we have discussed in [12], and will be part of future work.

However, several technical improvements to the proposed algorithm could be made. First, the mechanism to explore the network structure can be more sophisticated. Here, we have adopted a standard random walk algorithm applicable on unweighted networks, as implemented within the R-package igraph. In future work, we plan to employ a random walk which can be applied also to weighted graphs. The PPI network retrieved from the STRING database contains information about the certainty about a functional relationship between two proteins. It is reflected by a "combined score" value, which may change

depending on future biological experiments and research findings. Therefore, a weighted random walk could be utilized to explore paths within the PPI network using edge specific certainty scores. Second, the *niter* (number of greedy steps) is a crucial parameter. We not yet provide a stopping criteria. It is similar to the problem of choosing the right number of trees within a Random Forest. We plan to work on a mechanism which monitors convergence during the greedy runs while improving the sampling strategy to accelerate exploration.

Furthermore, we aim for testing the greedy Decision Forest using other than the AUC-ROC performance measure [16], or even a combination thereof by voting. Because of possible complications with the AUC-ROC on unbalanced data sets [40], we advise to apply our algorithm on balanced data sets, e.g. accommodated by up-sampling.

Finally, the glass-box nature of this algorithm enables actionable explainability; this is not a necessary condition, since there are other methods that aid actions, like Layer-wise Relevance Propagation [41]. Nevertheless, it is a sufficient one, since it guides the domain expert and the AI system’s designer in a transparent way towards actively improving the dataset the model, and the explainable method itself. This is going to be a major topic in future experiments that use this algorithm.

6. Conclusion

We propose a novel approach for selecting network modules from multi-modal node feature spaces. Our approach can be applied to a variety of possible applications where network structured data form the input and the corresponding nodes are associated with multi-modal numerical feature values. Disease module detection in PPI networks is a typical example of such an application. This work represents a contribution to the field of explainable AI because our greedy decision forest is easy to interpret. All decision paths are understandable, explainable and interpretable by a human domain expert and any contribution of features from different modalities can be easily retrieved. In future

work, we will extend our framework for explicit patient survival analyses, taking into account survival status as well as survival times (e.g. using tree-based risk prediction models). Furthermore, the analysis of mixed input data will be supported, whereby both categorical and numerical node features of arbitrary dimensions can be considered. Furthermore, this method is also applicable in other domains, e.g. in smart forestry or climate protection, i.e. everywhere where risk prediction models are needed.

Code availability

We have implemented our approach within the R-package DFNET, freely available on GitHub (<https://github.com/pievos101/DFNET>). DFNET interfaces with the R-package ranger, which enables efficient multi-core computation. Networks are internally organized and visualized using the R-package igraph.

Abbreviations

- DT = Decision Tree
- DF = Decision Forest
- RF = Random Forest
- GDF = Greedy Decision Forest
- FS = Feature Selection
- GNN = Graph Neural Network
- AUC = Area Under the Curve
- ROC = Receiver Operating Characteristic
- BN = Barabasi Network
- PPI = Protein-Protein Interaction
- ML = Machine Learning
- DL = Deep Learning

Acknowledgments

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. Parts of this work have been funded by the Austrian Science Fund (FWF), Project: P-32554 “explainable Artificial Intelligence”.

References

- [1] J. Snider, M. Kotlyar, P. Saraon, Z. Yao, I. Jurisica, I. Stagljär, Fundamentals of protein interaction network mapping, *Molecular systems biology* 11 (12) (2015) 848. doi:10.15252/msb.20156351.
- [2] M. Dehmer, F. Emmert-Streib, S. Pickl, A. Holzinger, *Big Data of Complex Networks*, CRC Press Taylor and Francis Group, Boca Raton, London, New York, 2016.
- [3] W. J. Kickert, E.-H. Klijn, J. F. Koppenjan, *Managing complex networks: Strategies for the public sector*, Sage, London, 1997.
- [4] S. Sakr, A. Bonifati, H. Voigt, A. Iosup, *Communications of the ACM* 64 (9) (2021) 62–71. doi:10.1145/3434642.
- [5] A. B. Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-López, D. Molina, R. Benjamins, R. Chatila, F. Herrera, Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai, *Information Fusion* 58 (2020) 82–115. doi:10.1016/j.inffus.2019.12.012.
- [6] A. Holzinger, From machine learning to explainable ai, in: *2018 World Symposium on Digital Intelligence for Systems and Machines (IEEE DISA)*, IEEE, 2018, pp. 55–66. doi:10.1109/DISA.2018.8490530.

- [7] I. Linkov, S. Galaitsi, B. D. Trump, J. M. Keisler, A. Kott, Cybertrust: From explainable to actionable and interpretable artificial intelligence, *Computer* 53 (9) (2020) 91–96. doi:10.1109/MC.2020.2993623.
- [8] A. Holzinger, Trends in interactive knowledge discovery for personalized medicine: Cognitive science meets machine learning, *IEEE Intelligent Informatics Bulletin* 15 (1) (2014) 6–14.
- [9] T. Ideker, R. Sharan, Protein networks in disease, *Genome research* 18 (4) (2008) 644–652. doi:10.1101/gr.071852.107.
- [10] C. Jean-Quartier, F. Jeanquartier, I. Jurisica, A. Holzinger, In silico cancer research towards 3r, *Springer/Nature BMC cancer* 18 (1) (2018) 408. doi:10.1186/s12885-018-4302-0.
- [11] A. Holzinger, B. Haibe-Kains, I. Jurisica, Why imaging data alone is not enough: Ai-based integration of imaging, omics, and clinical data, *European Journal of Nuclear Medicine and Molecular Imaging* 46 (13) (2019) 2722–2730. doi:10.1007/s00259-019-04382-9.
- [12] A. Holzinger, B. Malle, A. Saranti, B. Pfeifer, Towards multi-modal causability with graph neural networks enabling information fusion for explainable ai, *Information Fusion* 71 (7) (2021) 28–37. doi:10.1016/j.inffus.2021.01.008.
- [13] B. Wang, A. M. Mezlini, F. Demir, M. Fiume, Z. Tu, M. Brudno, B. Haibe-Kains, A. Goldenberg, Similarity network fusion for aggregating data types on a genomic scale, *Nature methods* 11 (3) (2014) 333–337.
- [14] N. D. Nguyen, D. Wang, Multiview learning for understanding functional multiomics, *PLOS Computational Biology* 16 (4) (2020) e1007677.
- [15] B. Pfeifer, M. G. Schimek, A hierarchical clustering and data fusion approach for disease subtype discovery, *Journal of Biomedical Informatics* 113 (2021) 103636.

- [16] A. Zheng, A. Casari, Feature engineering for machine learning: principles and techniques for data scientists, " O'Reilly Media, Inc.", 2018.
- [17] M. B. Kursa, W. R. Rudnicki, et al., Feature selection with the boruta package, *J Stat Softw* 36 (11) (2010) 1–13.
- [18] E. Keany, BorutaShap : A wrapper feature selection method which combines the Boruta feature selection algorithm with Shapley values. (Nov. 2020). doi:10.5281/zenodo.4247618.
URL <https://doi.org/10.5281/zenodo.4247618>
- [19] H. Deng, G. Runger, Gene selection with guided regularized random forest, *Pattern Recognition* 46 (12) (2013) 3483–3489.
- [20] S. Choobdar, M. E. Ahsen, J. Crawford, M. Tomasoni, T. Fang, D. Lamparter, J. Lin, B. Hescott, X. Hu, J. Mercer, et al., Assessment of network module identification across complex diseases, *Nature methods* 16 (9) (2019) 843–852.
- [21] T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, *arXiv preprint arXiv:1609.02907*.
- [22] K. Xu, W. Hu, J. Leskovec, S. Jegelka, How powerful are graph neural networks?, *arXiv preprint arXiv:1810.00826*.
- [23] R. Ying, D. Bourgeois, J. You, M. Zitnik, J. Leskovec, Gnnexplainer: Generating explanations for graph neural networks, *Advances in neural information processing systems* 32 (2019) 9240.
- [24] D. Luo, W. Cheng, D. Xu, W. Yu, B. Zong, H. Chen, X. Zhang, Parameterized explainer for graph neural network, *arXiv preprint arXiv:2011.04573*.
- [25] T. Schnake, O. Eberle, J. Lederer, S. Nakajima, K. Schütt, K. Müller, G. Montavon, Higher-order explanations of graph neural networks via relevant walks, *arXiv: 2006.03589*.

- [26] H. Chereda, A. Bleckmann, K. Menck, J. Perera-Bel, P. Stegmaier, F. Auer, F. Kramer, A. Leha, T. Beißbarth, Explaining decisions of graph convolutional neural networks: patient-specific molecular subnetworks responsible for metastasis prediction in breast cancer, *Genome medicine* 13 (1) (2021) 1–16.
- [27] L. Chen, H. Liu, J.-P. A. Kocher, H. Li, J. Chen, glmgraph: an r package for variable selection and predictive modeling of structured genomic data, *Bioinformatics* 31 (24) (2015) 3991–3993.
- [28] X. Guan, G. Runger, L. Liu, Dynamic incorporation of prior knowledge from multiple domains in biomarker discovery, *BMC bioinformatics* 21 (2) (2020) 1–10.
- [29] A. Holzinger, M. Plass, K. Holzinger, G. C. Crisan, C.-M. Pintea, V. Palade, A glass-box interactive machine learning approach for solving np-hard problems with the human-in-the-loop, arXiv:1708.01104.
- [30] M. N. Wright, A. Ziegler, ranger: A fast implementation of random forests for high dimensional data in c++ and r, arXiv preprint arXiv:1508.04409.
- [31] A. Liaw, M. Wiener, et al., Classification and regression by randomforest, *R news* 2 (3) (2002) 18–22.
- [32] D. J. MacKay, D. J. Mac Kay, Information theory, inference and learning algorithms, Cambridge university press, 2003.
- [33] I. H. Witten, E. Frank, M. A. Hall, C. Pal, Data mining: Practical machine learning tools and techniques. san francisco, Morgan Kaufmann.
- [34] A.-L. Barabasi, Z. N. Oltvai, Network biology: understanding the cell’s functional organization, *Nature reviews genetics* 5 (2) (2004) 101–113.
- [35] G. Csardi, T. Nepusz, The igraph software package for complex network research, *InterJournal Complex Systems* (2006) 1695.
URL <https://igraph.org>

- [36] C. v. Mering, M. Huynen, D. Jaeggi, S. Schmidt, P. Bork, B. Snel, String: a database of predicted functional associations between proteins, *Nucleic acids research* 31 (1) (2003) 258–261.
- [37] R. Schulte-Sasse, S. Budach, D. Hnisz, A. Marsico, Integration of multi-omics data with graph convolutional networks to identify new cancer genes and their associated molecular mechanisms, *Nature Machine Intelligence* 3 (6) (2021) 513–526.
- [38] S. V. Vasaiakar, P. Straub, J. Wang, B. Zhang, Linkedomics: analyzing multi-omics data within and across 32 cancer types, *Nucleic acids research* 46 (D1) (2018) D956–D963.
- [39] Y. Li, J. Bakke, D. Finkelstein, H. Zeng, J. Wu, T. Chen, Hnrnp1 is required for rhabdomyosarcoma cell growth and survival, *Oncogenesis* 7 (1) (2018) 1–13.
- [40] A. M. Carrington, P. W. Fieguth, H. Qazi, A. Holzinger, H. H. Chen, F. Mayr, D. G. Manuel, A new concordant partial auc and partial c statistic for imbalanced data in the evaluation of machine learning algorithms, *Springer/Nature BMC Medical Informatics and Decision Making* 20 (1) (2020) 1–12. doi:10.1186/s12911-019-1014-6.
- [41] S.-K. Yeom, P. Seegerer, S. Lapuschkin, A. Binder, S. Wiedemann, K.-R. Müller, W. Samek, Pruning by explaining: A novel criterion for deep neural network pruning, *Pattern Recognition* 115 (2021) 107899.