Εθνικο Μετσοβιο Πολυτεχνειο

Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων

Εργαστηριο Συστηματων Τεχνητης Νοημοσυνης και Μαθησης

Counterfactual Explanations for Graph Neural Networks

Diploma Thesis

by

Charalampos Koilakos

Επιβλέπων: Γεώργιος Στάμου
Καθηγητής Ε.Μ.Π.

Αθήνα, Μάρτιος 2025

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Πληροφορικής
Εργαστήριο Συστημάτων Τεχνητής Νοημοσύνης και Μάθησης

Counterfactual Explanations for Graph Neural Networks

Diploma Thesis

by

Charalampos Koilakos

Επιβλέπων: Γεώργιος Στάμου
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 26η Μαρτίου, 2025.

.....	.....	.....
Γεώργιος Στάμου	Αθανάσιος Βουλόδημος	Ανδρέας-Γεώργιος Σταφυλοπάτης
Καθηγητής Ε.Μ.Π.	Επίκουρος Καθηγητής Ε.Μ.Π.	Ομότιμος Καθηγητής Ε.Μ.Π.

Αθήνα, Μάρτιος 2025

.....

Χαραλαμπος Κοιλακος
Διπλωματούχος Ηλεκτρολόγος Μηχανικός
και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Οι Εξηγήσεις με Αντιπαραδείγματα έχουν προσελκύσει ιδιαίτερα μεγάλο ενδιαφέρον στο πεδίο της εξηγήσιμης τεχνητής νοημοσύνης, καθώς προσφέρουν εύστοχα παραδείγματα για το πως μικρές τροποποιήσεις σε μια είσοδο μπορούν να μεταβάλλουν την πρόβλεψη οποιουδήποτε μοντέλου. Σε αντίθεση με τις παραδοσιακές post-hoc μεθόδους επεξήγησης, οι οποίες φαίνεται απλώς να αναδεικνύουν τα κύρια χαρακτηριστικά που οδηγούν σε μια πρόβλεψη, οι αντιπαραδειγματικές εξηγήσεις εστιάζουν στις ελάχιστες απαραίτητες τροποποιήσεις που θα αλλάξουν το τελικό αποτέλεσμα, παρέχοντας έτσι ένα ισχυρό εργαλείο κατανόησης του μοντέλου και διευκόλυνσης των χρηστών. Η σημασία αυτών των χαρακτηριστικών γίνεται ακόμη πιο εμφανής σε δεδομένα με γράφους, όπου οι δομικές σχέσεις και οι προκαταλήψεις συχνά καθιστούν ακόμη πιο αδιαφανείς τις μεθόδους πρόβλεψης ενός μοντέλου.

Στην παρούσα εργασία, επιδιώκουμε την διερεύνηση εξηγήσεων με αντιπαραδείγματα για Νευρωνικά Δίκτυα με Γράφους (GNNs) μέσω τροποποιήσεων στις ακμές. Προτείνουμε ένα πλαίσιο "explainer" που μαθαίνει μια "μάσκα ακμών", καθοδηγούμενο από μια πολύ-παραγοντική συνάρτηση απωλειών. Αφαιρώντας επιλεκτικά συγκεκριμένες ακμές, η μέθοδός μας εντοπίζει τις ελάχιστες διαταραχές που απαιτούνται για να μεταβληθεί η πρόβλεψη οποιουδήποτε μοντέλου, οδηγώντας σε αντιπαραδειγματικές εξηγήσεις οι οποίες είναι ταυτόχρονα ακριβείς και ερμηνεύσιμες. Η συνάρτηση απωλειών περιλαμβάνει πολλαπλούς στόχους, επιτυγχάνοντας μια ισορροπία ανάμεσα στη διατήρηση της αρχικής πρόβλεψης, την ελαχιστοποίηση της διαταραχής και τη συνοχή της εξήγησης, τόσο σε επίπεδο γράφου όσο και κόμβου. Για κάθε διαφορετική περίπτωση, προσδιορίζουμε τα βασικά στοιχεία της δομής του γράφου και σχεδιάζουμε έναν κατάλληλο, ανεξάρτητο από το εκάστοτε μοντέλο, περιορισμό, επιτρέποντάς μας να δημιουργήσουμε ακόμη πιο συμπυκνωμένα και ακριβή αντιπαραδείγματα. Η αξιολόγηση και οι δοκιμές σε τυπικά σύνολα δεδομένων καταδεικνύουν ότι η προσέγγισή μας παράγει ουσιαστικές και εφαρμόσιμες αντιπαραδειγματικές εξηγήσεις, λαμβάνοντας υπόψη τόσο την ελαχιστοποίηση όσο και την ανθεκτικότητα. Επιπλέον, οι διαταραχές σε επίπεδο ακμής προσφέρουν σημαντική πληροφόρηση σχετικά με κρίσιμες δομικές σχέσεις του γράφου, παρέχοντας μια βαθύτερη κατανόηση για το πώς οι ανεπτυγμένες αναπαραστάσεις επηρεάζουν τις προβλέψεις ενός μοντέλου. Τα ενθαρρυντικά ευρήματα του προτεινόμενου "explainer" υπογραμμίζουν τη δυνατότητα βελτίωσης της ερμηνευσιμότητας, της αξιοπιστίας και της διαφάνειας σε εφαρμογές γραφημάτων, καλύπτοντας ένα ευρύ φάσμα εφαρμογών.

Λέξεις-κλειδιά — Εξηγήσεις με Αντιπαράδειγμα, Εξηγήσιμη τεχνητή νοημοσύνη, Νευρωνικά Δίκτυα με Γράφους, Μάσκα Ακμών,

Abstract

Counterfactual explanations are gaining increasing attention in the field of interpretable machine learning, as they provide insightful examples of how slight modifications to an input instance can alter the prediction of any model. Unlike traditional post hoc explanation methods that highlight important features driving a prediction, counterfactual explanations provide minimal perturbations required to change the outcome, thus offering a powerful tool for model clarification and user guidance. The importance of such characteristics becomes even more apparent in graph-based domains, where structural relationships and biases often obscure the predictive methods of a model even more.

In this thesis, we explore counterfactual explanations for Graph Neural Networks (GNNs) through edge-level perturbations. We propose an explainer framework that learns an edge mask guided by a multi-factor loss function. By selecting which edges to remove, our method identifies the smallest perturbations required to alter any model's prediction, leading to counterfactual explanations that are as accurate as they are intuitively interpretable. The loss function integrates multiple objectives, balancing fidelity to the original prediction, perturbation minimization, and explanation coherence across both graph and node-level classification tasks. For each different task, we determine the key aspects of the graph structure and design a suitable, model-agnostic constraint that allows us to make counterfactual examples even more minimal and precise. Empirical evaluations and testing across benchmark datasets demonstrate that our approach generates meaningful and actionable counterfactuals, while accounting for minimality and robustness. Moreover, the edge perturbations provide insight into critical structural relationships within the graph, providing a deeper understanding of how the learned representations influence any model's predictions. The promising results of our explainer highlight the potential to improve interpretability, trustworthiness, and transparency in graph applications across multiple domains and tasks.

Keywords — Counterfactual Explanations, Graph Neural Networks, Perturbation, Edge Mask,

Ευχαριστίες

Η ακαδημαϊκή και επαγγελματική πορεία ενός ατόμου δεν καθορίζεται μόνο από τα προσωπικά του στοιχεία, αλλά και από το περιβάλλον του. Γι'αυτό λοιπόν, θα ήθελα να ευχαριστήσω τον επιβλέποντα μου, κ. Στάμου Γεώργιο για την καθοδήγηση αλλά και για την ευκαιρία να εκπονήσω την διπλωματική μου εργασία στο εργαστήριο Συστημάτων Τεχνητής Νοημοσύνης και Μάθησης. Θα ήθελα επίσης να ευχαριστήσω τον Ορφέα-Μενή Μαστρομιχαλάκη για την καθοδήγηση, τις ιδέες, και την άριστη συνεργασία κατά την διάρκεια της εκπόνησης αυτής της εργασίας.

Παράλληλα, θα ήθελα να ευχαριστήσω και ανθρώπους από το προσωπικό μου περιβάλλον. Αρχικά τους γονείς μου Αναστάσιο και Ρίτα και τον αδερφό μου Μιχάλη για την στήριξη τους όλα τα χρόνια της φοίτησης μου. Επιπλέον θα ήθελα να ευχαριστήσω τους φίλους μου και συμφοιτητές μου καθώς και την κοπέλα μου Άννα που βρίσκονταν μαζί μου σε κάθε βήμα και αποφοιτήσαμε από ένα πολύ σημαντικό κεφάλαιο μαζί.

Χαράλαμπος Κοιλάκος, Μάρτιος 2025

Contents

List of Figures

Chapter 0

Εκτεταμένη Περίληψη στα Ελληνικά

0.1 Θεωρητικό Υπόβαθρο

Οι πρόσφατες εξελίξεις στη μηχανική μάθηση έχουν φέρει σημαντικές προόδους σε πολλά πεδία, από την ανάλυση εικόνων και τη ρομποτική μέχρι την επεξεργασία φυσικής γλώσσας και τα μεγάλα γλωσσικά μοντέλα (LLMS). Ένα ακόμα ιδιαίτερο ερευνητικό πεδίο είναι η μελέτη δεδομένων γράφων με περίπλοκες σχέσεις και δομή. Τα Νευρωνικά Δίκτυα Γράφων αναδεικνύονται ως μια καινοτόμος μεθοδολογία ικανή να αντιληφθεί πολύπλοκες συσχετίσεις και να παράγει αξιόπιστες προβλέψεις, παρά τους υπάρχοντες περιορισμούς σχετικά με την περιπλοκότητα των δεδομένων.

Για να αντιμετωπιστούν οι προκλήσεις που θέτουν τα μοντέλα "μαύρου κουτιού", αναπτύσσονται μέθοδοι ερμηνευσιμότητας (Explainable AI), συμπεριλαμβανομένων των Εξηγήσεων με Αντιπαραδείγματα. Οι τελευταίες, αξιοποιούνται ώστε να καταδείξουν ποίες αλλαγές θα ήταν αρκετές για να τροποποιηθεί η έξοδος ενός μοντέλου, αναγνωρίζοντας έτσι τα καθοριστικά στοιχεία για την εκάστοτε πρόβλεψη. Μέσα σε αυτό το πλαίσιο, παρουσιάζουμε έναν εξηγητή (explainer) που στοχεύει να απαντήσει στο ερώτημα «Ποια χαρακτηριστικά εισόδου πρέπει να αλλάξουμε ώστε το Νευρωνικό Δίκτυο να παράγει μια διαφορετική πρόβλεψη;». Πρακτικά, ο αλγόριθμός μας δημιουργεί μια "μάσκα" ακμών, βασιζόμενος σε πρότερη γνώση του εκάστοτε προβλήματος, και διαγράφει μόνο τις απαραίτητες ακμές έτσι ώστε να ανατραπεί η αρχική πρόβλεψη. Παράλληλα, ενσωματώνει μια πολυπαραγοντική συνάρτηση κόστους που επιδιώκει την αντιστροφή της εκάστοτε πρόβλεψης, την ελαχιστοποίηση των διαγραφών και οποιουσδήποτε επιπλέον περιορισμούς υπαγορεύει το πρόβλημα.

Η μέθοδός μας δεν εξαρτάται από την εσωτερική δομή και τις παραμέτρους του μοντέλου (model agnostic) και μπορεί να εφαρμοστεί τόσο σε ταξινομήσεις κόμβων όσο και σε ταξινομήσεις γράφων. Για να αξιολογήσουμε την αποτελεσματικότητά της, πειραματιστήκαμε με τρία αντιπροσωπευτικά σύνολα δεδομένων (ένα για ταξινόμηση κόμβων και δύο για ταξινόμηση γραφημάτων) και χρησιμοποιήσαμε δύο βασικούς μετρικές: το ποσοστό αλλαγής ετικέτας (prediction flip) και την απόσταση επεξεργασίας γραφήματος (graph edit distance). Τα αποτελέσματα του explainer μας, δείχνουν ότι παράγoyme Εξηγήσεις με Αντιπαραδείγματα για την πλειοψηφία των δειγμάτων σε κάθε σύνολο δεδομένων, διατηρώντας παράλληλα χαμηλό το επίπεδο αλλαγών, και διασφαλίζοντας ότι οι παρεμβάσεις είναι ουσιαστικές και εφαρμόσιμες.

Σε αυτή την περίληψη θα ξεκινήσουμε παρουσιάζοντας όλο το θεωρητικό υπόβαθρο για βασικές έννοιες της μηχανικής μάθησης. Έπειτα θα εμβαθύνουμε στους στα δεδομένα με γράφους και τις ιδιαιτερότητες τους ενώ παράλληλα θα εξηγήσουμε πώς οι εξηγήσεις με αντιπαραδείγματα είναι ένα σημαντικό μέσο για τον εκδημοκρατισμό της Μηχανικής Μάθησης. Επιπλέον θα παρουσιάσουμε το θεωρητικό υπόβαθρο της μεθόδου που χρησιμοποιήσαμε καθώς και τα αναλυτικά αποτελέσματα που προέκυψαν από τα πειράματα μας.

0.1.1 Εκπαίδευση Νευρωνικών Δικτύων

Σε αυτή την παράγραφο θα παρουσιάσουμε τις βασικές έννοιες της εκπαίδευσης ενός νευρωνικού δικτύου, που είναι σημαντικές για την μετέπειτα θεμελίωση του explainer μας.

Τα νευρωνικά δίκτυα αποτελούν μία από τις πλέον βασικές προσεγγίσεις στη μηχανική μάθηση, αξιοποιώντας διαδοχικά στάδια (layers) για την επεξεργασία και την ανάλυση δεδομένων. Στην απλούστερη μορφή του, ένας νευρώνας (neuron) υπολογίζει ένα βάρη-σταθμισμένο άθροισμα των εισόδων του και προσθέτει μια παράμετρο μετατόπισης (bias):

$$z_i^{(l)} = \sum_j w_{ij}^{(l)} a_j^{(l-1)} + b_i^{(l)},$$

όπου $w_{ij}^{(l)}$ είναι το βάρος από τον j-οστό νευρώνα της προηγούμενης στοιβάδας $(l-1)$ στον i-οστό νευρώνα της τρέχουσας στοιβάδας (l), και $a_j^{(l-1)}$ η έξοδος του j-οστού νευρώνα στην προηγούμενη στοιβάδα. Ακολουθεί η εφαρμογή μιας συνάρτησης ενεργοποίησης (activation function), ώστε να προκύψει η τελική έξοδος $a_i^{(l)}$.

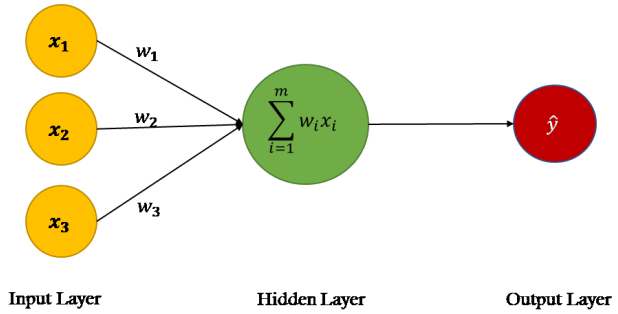


Figure 0.1.1: Ένας απλός νευρώνας. [39]

Η απόδοση ενός νευρωνικού δικτύου αξιολογείται μέσω της συνάρτησης κόστους, που μετρά την απόκλιση μεταξύ των προβλέψεων του μοντέλου και των πραγματικών τιμών. Σε regression tasks, συνηθίζεται η χρήση της Mean Squared Error (MSE):

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2,$$

η οποία τιμωρεί ιδιαίτερα τις μεγαλύτερες αποκλίσεις μεταξύ πρόβλεψης $\hat{y}_i$ και πραγματικής τιμής y_i. Για την εκτίμηση μέσης απόλυτης απόκλισης (mean absolute error), μπορεί να χρησιμοποιηθεί η $L1$ loss function:

$$L1 = \frac{\sum_{i=1}^{n} |y_i - f(x_i)|}{N},$$

ενώ για προβλήματα ταξινόμησης συνηθίζεται η Cross-Entropy loss:

$$CE = -\frac{1}{N}\sum_{i=1}^{N}\Big[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)\Big],$$

η οποία προέρχεται από τη θεωρία της πληροφορίας και "τιμωρεί" αυστηρά τις περιπτώσεις όπου το μοντέλο αποδίδει υψηλή πιθανότητα σε λανθασμένη κλάση. Στην παρούσα εργασία θα χρησιμοποιήσουμε την CE loss καθώς και στοιχεία από την L1 προκειμένου να αλλάξουμε την πρόβλεψη ενός μοντέλου.

Η ελαχιστοποίηση της παραπάνω loss function πραγματοποιείται συνήθως με αλγορίθμους βασισμένους στη μέθοδο Gradient Descent. Σε κάθε βήμα εκπαίδευσης, υπολογίζουμε την παράγωγο του σφάλματος (gradient) και ενημερώνουμε αντίστοιχα τα βάρη:

$$\theta' = \theta - \epsilon \nabla_\theta L(\theta),$$

όπου ϵ είναι ο ρυθμός εκμάθησης (learning rate). Για να υπολογιστούν αυτές οι παράγωγοι οικονομικά και με ακρίβεια, χρησιμοποιείται η μέθοδος της backpropagation. Σε αυτήν, το δίκτυο "πορεύεται προς τα πίσω": από την έξοδο (output layer) ως την είσοδο (input layer), υπολογίζοντας πώς οι αλλαγές σε κάθε βάρος $w_{ij}^{(l)}$ και μετατόπιση $b_i^{(l)}$ επηρεάζουν τη συνολική απώλεια. Μέσω συνεχών ενημερώσεων, τα νευρωνικά δίκτυα βελτιώνουν διαδοχικά τις προβλέψεις τους, κάνοντας τις παραμέτρους τους όλο και πιο ακριβείς ως προς τα δεδομένα εκπαίδευσης.

Συνοψίζοντας, η ικανότητα ενός νευρωνικού δικτύου να μαθαίνει από δεδομένα βασίζεται:

- Στον ορισμό κατάλληλων **συναρτήσεων ενεργοποίησης** για τους νευρώνες.
- Στην επιλογή μιας **loss function** που αντικατοπτρίζει τις ανάγκες του προβλήματος.
- Στη χρήση αποτελεσματικών αλγορίθμων (**Gradient Descent**) για την ελαχιστοποίηση των απωλειών.
- Στον υπολογισμό των παραγώγων μέσω **backpropagation**, ώστε το δίκτυο να επικαιροποιεί τις παραμέτρους του με συστηματικό και οικονομικό τρόπο.

Με αυτόν τον τρόπο, τα νευρωνικά δίκτυα μπορούν να εκπαιδεύονται σε πληθώρα σεναρίων, επιτυγχάνοντας υψηλή ακρίβεια και ευελιξία σε προβλήματα πρόβλεψης και ταξινόμησης.

0.1.2 Ιδιότητες Γράφων

Οι γράφοι συνιστούν έναν ιδιαίτερα ευέλικτο τρόπο αναπαράστασης περίπλοκων σχέσεων μεταξύ δεδομένων που συνδέονται δομικά σε κάποιον χώρο. Ένας γράφος G περιγράφεται από το σύνολο των κόμβων (nodes) V και το σύνολο των ακμών του (edges) E:

$$G(V, E) = \{(u, v) : u, v \in V, (u, v) \in E\},$$

όπου οι ακμές μπορούν να μην έχουν κατεύθυνση (undirected) ή να είναι κατευθυνόμενες (directed), καθώς και ζυγισμένες (weighted) ή και χωρίς βάρος (unweighted). Μια διαδεδομένη μαθηματική αναπαράσταση για έναν οποιονδήποτε γράφο αποτελεί η **μήτρα γειτνίασης** (adjacency matrix) $A \in \mathbb{R}^{n \times n}$, όπου κάθε στοιχείο a_{ij} ορίζεται ως:

$$a_{ij} = \begin{cases} 1, & \text{if } \{v_i, v_j\} \in E \\ 0, & \text{otherwise} \end{cases}$$

Σε μη κατευθυνόμενους γράφους, η μήτρα γειτνίασης είναι συμμετρική και τα στοιχεία a_{ij} παίρνουν την τιμή 1 εάν υπάρχει ακμή και 0 εάν δεν υπάρχει. Για ζυγισμένα γραφήματα, η τιμή του a_{ij} αντιστοιχεί στο βάρος της ακμής μεταξύ v_i και v_j.

Όσον αφορά τους κόμβους, ένα από τα βασικά χαρακτηριστικά μεγέθη είναι ο **βαθμός (degree)** ενός κόμβου v, ο οποίος σε γράφους χωρίς κατεύθυνση ισούται με το πλήθος των ακμών που τον συνδέουν με γειτονικούς κόμβους. Μαθηματικά, αυτή η σχέση μπορεί να εκφραστεί ως:

$$\deg(v) = \sum_{u=1}^{N} a_{vu},$$

όπου $a_{vu} = 1$ εφόσον υπάρχει ακμή μεταξύ του κόμβου v και του κόμβου u. Κόμβοι με υψηλό βαθμό τείνουν να έχουν σημαντικό ρόλο στη διάδοση πληροφορίας σε έναν γράφο καθώς συνδέονται με τους περισσότερους κόμβους και μεταδίδουν την πληροφορία τους στην πλειοψηφία του γράφου.

Παράλληλα, μια από τις πιο διαδεδομένες μετρικές για την αξιολόγηση της σημασίας μιας ακμής είναι η **edge betweenness**, η οποία εκφράζει πόσο συχνά μια συγκεκριμένη ακμή βρίσκεται σε συντομότερες διαδρομές μεταξύ ζευγών κόμβων:

$$\text{BetweennessCentrality}(e) = \sum_{s \neq t \in V} \frac{\sigma_{s,t}(e)}{\sigma_{s,t}},$$

όπου s και t είναι διαφορετικοί κόμβοι, $\sigma_{s,t}$ ο συνολικός αριθμός συντομότερων διαδρομών από τον κόμβο s στον κόμβο t και $\sigma_{s,t}(e)$ ο αριθμός αυτών που περνούν από την ακμή e. Μια ακμή με υψηλή τιμή edge betweenness συχνά λειτουργεί ως "γέφυρα" μεταξύ διαφορετικών περιοχών του γράφου και η ύπαρξή της είναι καθοριστική για την διάδοση της πληροφορίας σε διαφορετικές γειτονιές του γράφου..

Η ευελιξία των γραφημάτων στην αναπαράσταση διαφορετικών ειδών δεδομένων (κοινωνικά, βιολογικά, κ.ά.) οδήγησε στην ανάπτυξη των **Νευρωνικών δικτύων με Γράφους**, τα οποία αξιοποιούν τόσο την τοπολογία του γράφου όσο και τα χαρακτηριστικά (features) των κόμβων/ακμών. Σε αντίθεση με τις παραδοσιακές μορφές δεδομένων (π.χ. πίνακες, εικόνες), οι δομές με γράφους εστιάζουν στη συνδεσιμότητα και τις σχέσεις μεταξύ οντοτήτων, καθιστώντας αναγκαίες εξειδικευμένες αρχιτεκτονικές και αλγόριθμους για την αποτελεσματική εκπαίδευση και εξαγωγή συμπερασμάτων.

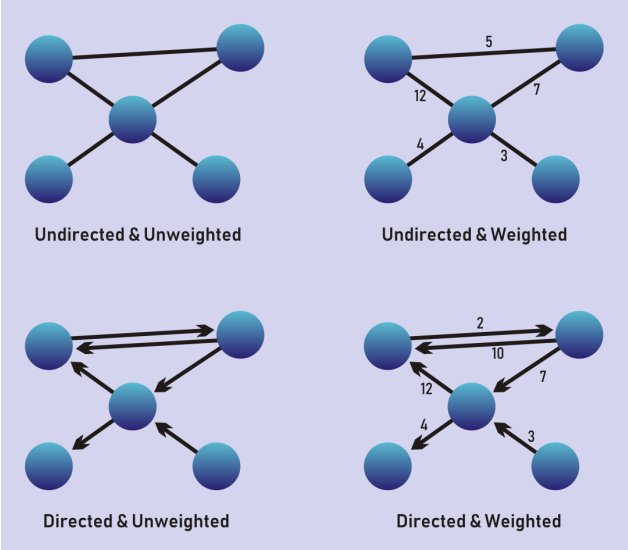


Figure 0.1.2: Διαφορετικοί πιθανοί τύποι γράφου [5]

0.1.3 Νευρωνικά Δίκτυα Γράφων

Αφού περιγράψαμε τα βασικά χαρακτηριστικά που ξεχωρίζουν τα δεδομένα με γράφους, ήρθε η ώρα να θεμελιώσουμε τα Νευρωνικά Δίκτυα με Γράφους, τα οποία μετέπειτα θα προσπαθήσουμε να ερμηνεύσουμε. Επιπλέον θα παρουσιάσουμε συνοπτικά την αρχιτεκτονική μερικών μοντέλων που θα χρησιμοποιηθούν στην παρούσα εργασία.

Δεδομένου ότι η δομή ενός γράφου υπάρχει φυσικά παντού γύρω μας, δημιουργήθηκαν νευρωνικά δίκτυα συγκεκριμένου τύπου, προκειμένου να μπορέσουν να εκπαιδευτούν σε αυτού του είδους τα δεδομένα. Τα γραφήματα αποτελούν μη ευκλείδεια μορφή δεδομένων και επομένων τα Νευρωνικά Δίκτυα Γράφων (GNNs) πρέπει να μπορούν να συλλάβουν τις πολύπλοκες και πολυδιάστατες σχέσεις μεταξύ των δεδομένων. Τα δίκτυα αυτά, έχουν αναπτυχθεί ιδιαίτερα τα τελευταία χρόνια και χρησιμοποιούνται ευρέως σε κοινωνικά, οικονομικά domains καθώς και στην μοριακή βιολογία για την αναζήτηση φαρμάκων.

Τα Νευρωνικά Δίκτυα με Γράφους χρησιμοποιούνται, όπως είπαμε, σε πολλά διαφορετικά είδη προβλημάτων. Παρόλα αυτά, μπορούμε να εντοπίσουμε μερικές γενικές κατηγορίες ανάλογα με το κομμάτι του γράφου στο οποίο θέλουμε να κάνουμε προβλέψεις.

- **Προβλέψεις σε επίπεδο κόμβων (Node-level tasks):** Σε αυτές τις περιπτώσεις, το μοντέλο καλείται να προβλέψει τις ιδιότητες κάθε κόμβου ξεχωριστά, σε έναν μεγάλο γράφο. Πρακτικά, το μοντέλο λαμβάνει ως είσοδο έναν κόμβο και οφείλει είτε να του αποδώσει μια κλάση είτε να υπολογίσει μια αριθμητική τιμή για κάθε κόμβο. Η πιο συνηθισμένη εργασία σε αυτό το επίπεδο είναι η ταξινόμηση κόμβων (supervised node classification), με την οποία θα ασχοληθούμε και σε αυτή την εργασία.
- **Προβλέψεις σε επίπεδο ακμών (Edge-level tasks):** Απαιτούν από το μοντέλο να προβλέψει κάποιο χαρακτηριστικό που αντιστοιχεί σε κάθε ακμή ενός γράφου. Στην πράξη, είναι άμεσα ανάλογες με αυτές των κόμβων, με τα πιο διαδεδομένα παραδείγματα να είναι η ταξινόμηση ακμών (edge classification) η πρόβλεψη ύπαρξης ακμής (link prediction).
- **Προβλέψεις σε επίπεδο γραφήματος (Graph-level tasks):** Σε αυτή την κατηγορία, το σύνολο δεδομένων δεν αποτελείται πλέον από έναν ενιαίο μεγάλο γράφο, αλλά από πολλά επιμέρους γραφήματα, και το μοντέλο πρέπει να αποδώσει είτε μια κλάση είτε κάποιο αριθμητικό χαρακτηριστικό για κάθε γράφημα. Αυτό επιτυγχάνεται με την ανίχνευση προτύπων σε επίπεδο ολόκληρου γραφήματος και τη συμπύκνωση της πληροφορίας σε μία μόνο είσοδο κάθε φορά. Στο πλαίσιο αυτής της εργασίας, θα εστιάσουμε στο πιο σύνηθες task, την ταξινόμηση γραφημάτων (graph classification).

Επιπλέον τα Νευρωνικά Δίκτυα με Γράφους μπορούν να ταξινομηθούν ανάλογα ανάλογα με την αρχιτεκτονική που ακολουθούν σε συνελικτικά, επαναλαμβανόμενα, αυτοκωδικοποιητές και χωροχρονικά και ανάλογα με τον τρόπο εκπαίδευσης σε επιβλεπόμενα, μη επιβλεπόμενα και μερικώς επιβλεπόμενα. Παρακάτω θα παρουσιάσουμε

δύο ευρέως γνωστές αρχιτεκτονικές, που χρησιμοποιήθηκαν σε αυτήν την εργασία. Συγκεκριμένα οι δύο αυτές αρχιτεκτονικές εντάσονται στην κατηγορία των Φασματικών Νευρωνικών Δικτύν με Γράφους (**Spectral-based ConvGNNs**

Τα δίκτυα που ανήκουν σε αυτή την κατηγορία έχουν ως αφετηρία τη φασματική θεωρία γραφημάτων (spectral graph theory), ορίζοντας τη συνέλιξη (convolution) στο φασματικό πεδίο του γράφου μέσω της ιδιοδιάσπασης του λαπλασιανού πίνακα. Ο λαπλασιανός πίνακας L παρέχει κρίσιμες πληροφορίες για τη δομή του γράφου, ενώ ο μετασχηματισμός Fourier αξιοποιείται για την προβολή του γράφου σε ένα ορθοκανονικό χώρο. Θεωρώντας έναν συμμετρικό, κανονικοποιημένο λαπλασιανό $L = U\Lambda U^T$, όπου το U είναι ο πίνακας ιδιοδιανυσμάτων ταξινομημένων με βάση τις ιδιοτιμές που προκύπτουν από τη διάσπαση του κανονικοποιημένου λαπλασιανού, και το Λ είναι ένας διαγώνιος πίνακας με τις ιδιοτιμές, έχουμε ένα σήμα γραφήματος x που μεταφέρεται στο φασματικό πεδίο μέσω $\hat{x} = U^T x$. Στη συνέχεια, ο τελεστής της συνέλιξης πραγματοποιείται χρησιμοποιώντας ένα φίλτρο g_θ ορισμένο ως:

$$g_\theta * x = U g_\theta(\Lambda) U^T x$$

Αυτή η προσέγγιση αντλεί έμπνευση από τις κλασικές μεθόδους επεξεργασίας σήματος, όπου η συνέλιξη στον χώρο αντιστοιχεί σε στοιχειώδη πολλαπλασιασμό (element-wise multiplication) στον συχνοτικό χώρο. Εντούτοις, λόγω του υψηλού υπολογιστικού κόστους που συνεπάγεται η ιδιοδιάσπαση του λαπλασιανού πίνακα, προτάθηκαν νέες μέθοδοι που χρησιμοποιούν αποδοτικότερα φίλτρα, τις οποίες θα δούμε παρακάτω.

Το **Chebyshev Spectral CNN (ChebNet)** [10] ανήκει στην κατηγορία των φασματικών νευρωνικων δικτυων γραφων και προσεγγίζει αποτελεσματικά τα φασματικά φίλτρα χωρίς να απαιτείται κοστοβόρα ιδιοδιάσπαση του λαπλασιανού πίνακα. Η βασική του ιδέα βασίζεται στην ανάπτυξη του φίλτρου g_θ σε πολυώνυμα Chebyshev, με αποτέλεσμα να εντοπίζεται χωρικά η συνέλιξη σε ένα πεπερασμένο γειτονικό περιβάλλον. Συγκεκριμένα, ο τελεστής της συνέλιξης προσεγγίζεται από την εξίσωση:

$$g_\theta * x \approx \sum_{k=0}^{K} \theta_k T_k(\tilde{L}) x$$

όπου T_k είναι το πολυώνυμο Chebyshev βαθμού k, θ_k είναι οι εκπαιδεύσιμες παράμετροι και $\tilde{L} = \frac{2}{\lambda_{\max}} L - I$ είναι ο ανακλιμακωμένος λαπλασιανός πίνακας, με $\lambda_{\max}$ να είναι η μέγιστη ιδιοτιμή του L. Το πολυώνυμο Chebyshev τάξης k μπορεί να υπολογιστεί σύμφωνα με τον παρακάτω τύπο:

$$T_k(\tilde{L}) = U T_k(\tilde{\Lambda}) U^T$$

Αυτή η διατύπωση μειώνει αισθητά το υπολογιστικό κόστος αποφεύγοντας τον ρητό μετασχηματισμό Fourier, ενώ παράλληλα διατηρεί μια τοπική περιοχή επιρροής (localized receptive field), απαραίτητη για τη σύλληψη τοπικών δομών του γράφου. Το συγκεκριμένο δίκτυο έχει βρει εφαρμογή σε διάφορους τομείς, όπως δίκτυα αισθητήρων, κοινωνικά γραφήματα και μοριακή χημεία.

Στη συνέχεια, το **Graph Convolutional Network (GCN)** απλοποιεί περαιτέρω το φασματικό τελεστή συνέλιξης [19] που χρησιμοποιείται στο Chebyshev δίκτυο, εφαρμόζοντας ουσιαστικά μια προσέγγιση πρώτης τάξης των πολυωνύμων Chebyshev με επιβολή $K = 1$ και $\lambda_{\max} = 2$. Η αντίστοιχη συνέλιξη ορίζεται ως:

$$x *_G g_\theta = \theta \left(I_n + D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \right) x$$

όπου $D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$ είναι μια διαφορετική αναπαράσταση του λαπλασιανού πίνακα που χρησιμοποιεί τον πίνακα βαθμών D και τη μήτρα γειτνίασης A. Εισάγοντας μια κανονικοποίηση, μπορούμε να γράψουμε το κ'αθε επίπεδο (layer) του GCN ως:

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-1/2} \tilde{A} \tilde{D}^{-1/2} H^{(l)} W^{(l)} \right)$$

όπου σ είναι μια συνάρτηση ενεργοποίησης (activation function), $H^{(l)}$ είναι η μήτρα χαρακτηριστικών στο επίπεδο l και $W^{(l)}$ η μαθηματικά εκπαιδεύσιμη παράμετρος (trainable weight matrix). Αυτή η προσέγγιση στην

πράξη αυξάνει τη σταθερότητα των υπολογισμών και επιταχύνει τη σύγκλιση κατά την εκπαίδευση. Λόγω της απλότητας, της υπολογιστικής απόδοσης και της ικανότητάς του να αποτυπώνει τις τοπικές δομές ενός γράφου, το GCN έχει υιοθετηθεί ευρέως σε ποικίλα πεδία εφαρμογών. Στο πλαίσιο της παρούσας εργασίας, το Graph Convolutional Neural Network αποτελεί το βασικό μοντέλο στο οποίο εστιάζουμε, καθώς θα κληθούμε να εξηγήσουμε τις προβλέψεις του σε διάφορα προβλήματα.

0.1.4 Εξηγήσεις με Αντιπαραδείγματα

Σε αυτό το σημείο, ήρθε η ώρα να επεκταθούμε πάνω στο θέμα των Εξηγήσεων με Αντιπαραδείγματα. Αρχικά θα παρουσιάσουμε τους λόγους για τους οποίους οι εξηγήσεις αυτές είναι σημαντικές για τον τομέα της Τεχνητής Νοημοσύνης. Έπειτα θα αναλύσουμε τις διαφορετικές μεθόδους και τα προβλήματα που λύνουν οι Εξηγήσεις με Αντιπαράδειγμα ενώ θα αναφέρουμε και την μαθηματική διατύπωσή τους.

Οι σύγχρονες τεχνικές Βαθιάς Μάθησης διακρίνονται για τις υψηλές επιδόσεις τους σε τομείς όπως η βιολογία, τα κοινωνικά δίκτυα και τα οικονομικά συστήματα, ωστόσο συχνά χαρακτηρίζονται από έλλειψη διαφάνειας που δυσκολεύει την κατανόηση του τρόπου λειτουργίας τους. Ειδικότερα, τα Νευρωνικά Δίκτυα Γράφων (GNNs) παρέχουν σημαντικές δυνατότητες επεξεργασίας δεδομένων με σύνθετη δομή, αλλά κληρονομούν και ενδέχεται να ενισχύσουν τις προϋπάρχουσες μεροληψίες του συνόλου δεδομένων. Καθώς βασίζονται σε πολλαπλές συναρτήσεις ενεργοποίησης και πολλαπλά κρυφά στάδια, παραμένουν "μαύρα κουτιά" για πολλούς χρήστες, με αποτέλεσμα να καθίσταται δυσχερής η διερεύνηση των παραγόντων που οδηγούν στην εκάστοτε πρόβλεψη.

Οι Εξηγήσεις με Αντιπαραδείγματα (counterfactual explanations) έρχονται να αντιμετωπίσουν τις παραπάνω προκλήσεις, προσφέροντας μια δράσιμη και κατανοητή οπτική στην ερμηνεία μοντέλων. Σε αντίθεση με άλλες μεθόδους, δεν περιορίζονται στην ερώτηση "ποιοι ήταν οι παράγοντες που οδήγησαν στην συγκεκριμένη απόφαση;" αλλά επεκτείνονται στο "πώς μπορούμε να τροποποιήσουμε το αποτέλεσμα προς όφελος μας με τις ελάχιστες δυνατές αλλαγές;". Αυτές οι εξηγήσεις αποδεικνύονται πολύτιμες σε περιπτώσεις όπου απαιτούνται διαφάνεια, αμεροληψία και πρακτικότητα.

Οι Εξηγήσεις με αντιπαραδείγματα σε δεδομένα γράφων μπορούν να ταξινομηθούν ανάλογα με το σημείο εστίασής τους (δηλαδή αν στοχεύουν σε επίπεδο ολόκληρου του μοντέλου ή ενός μόνο δείγματος) αλλά και ανάλογα με την προσέγγιση που ακολουθούν για να παράξουν τις εξηγήσεις [15, 41, 2].

Σε επίπεδο μοντέλου (Model-Level): Οι προσεγγίσεις αυτές στοχεύουν στην παραγωγή αντιπαραδειγμάτων που αντικατοπτρίζουν τη συνολική λογική του Νευρωνικού Δικτύου. Προσφέρουν ένα ευρύτερο σύνολο κανόνων ή παραδειγμάτων, ικανά να διαφωτίσουν τον τρόπο με τον οποίο το μοντέλο συνολικά φτάνει στις προβλέψεις του, ανεξάρτητα από την εκάστοτε είσοδο [20].

Σε επίπεδο μεμονωμένου δείγματος εισόδου (Instance-Level): Αντίθετα, αυτές οι μέθοδοι εντοπίζουν αντιπαραδείγματα για κάθε μια από τις εισόδους δεδομένων ξεχωριστά. Για μια εργασία ταξινόμησης γραφήματος, αυτό μεταφράζεται σε αλλαγή της πρόβλεψης για ένα συγκεκριμένο γράφημα, ενώ στη ταξινόμηση κόμβων, για έναν συγκεκριμένο κόμβο.

Επιπλέον, η ανάλυση των προσεγγίσεων σε υψηλό επίπεδο μάς επιτρέπει να διακρίνουμε τις εξής κατηγορίες:

- **Αναζητητικές (Search-Based) μέθοδοι:** Ξεκινούν από ένα αρχικό υποψήφιο αντιπαράδειγμα x' και το βελτιστοποιούν ωσότου αλλάξει η πρόβλεψη του αρχικού μοντέλου. Χρησιμοποιούν κριτήρια αναζήτησης εντός του συνόλου δεδομένων, αξιοποιώντας gradient-based τεχνικές όταν το μοντέλο είναι διαφορίσιμο ή μεθόδους όπως εξελικτικούς αλγορίθμους όταν δεν είναι.
- **Ευρετικές (Heuristic-Based) μέθοδοι:** Χρησιμοποιούν πολιτικές τροποποίησης του γραφήματος, βασισμένες σε συγκεκριμένες δομικές και στατιστικές τεχνικές, ώστε να επιτύχουν το επιθυμητό αποτέλεσμα. Οι αλλαγές γίνονται σταδιακά και κατευθυνόμενα (π.χ. με greedy αλγόριθμους), έως ότου φτάσουν στο επιθυμητό αντιπαράδειγμα.
- **Μαθησιακές (Learning-Based) μέθοδοι:** Εκπαιδεύουν ένα δεύτερο μοντέλο για την παραγωγή αντιπαραδειγμάτων, συνήθως αξιοποιώντας μια counterfactual loss function για τη δημιουργία υποψήφιων τροποποιήσεων.

Επιπλέον, οι μαθησιακές προσεγγίσεις χωρίζονται σε τρεις υποκατηγορίες:

- **Perturbation:** Διαγράφουν στοιχεία από την μήτρας γειτνίασης ή την μήτρα χαρακτηριστικών (feature matrix) έως ότου προκύψει ένα έγκυρο αντιπαράδειγμα [64, 8].

- **Ενισχυτική μάθηση (Reinforcement Learning):** Χρησιμοποιούν πράκτορες (agents) που αναζητούν αντιπαραδείγματα μέσω μιας συνάρτησης ανταμοιβής (reward), την οποία ορίζει ο χρήστης [38].

- **Generative Methods:** Βασίζονται σε γεννητικά μοντέλα ή variational autoencoders (VAE) για τη δημιουργία αντιπαραδειγμάτων σε έναν εκπαιδευμένο λανθάνοντα χώρο [48, 26].

Τέλος, σημαντικό ρόλο παίζει και ο βαθμός πρόσβασης που έχει μια μέθοδος στο εσωτερικό του μοντέλου:

- **Model-Agnostic:** Οι μέθοδοι αυτές δεν απαιτούν πρόσβαση στις εσωτερικές παραμέτρους ή στα βάρη του μοντέλου, στηριζόμενες αποκλειστικά σε ζεύγη εισόδου-εξόδου.

- **Model-Specific:** Χρησιμοποιούν τα βάρη ή άλλα εσωτερικά στοιχεία του μοντέλου, ώστε να εξηγήσουν καλύτερα γιατί προκύπτουν συγκεκριμένες αποφάσεις.

Στην πράξη, οι διαχωρισμοί αυτοί συχνά επικαλύπτονται, και πολλές μέθοδοι αξιοποιούν ιδέες από παραπάνω από μία κατηγορίες, προσφέροντας ευέλικτες και αποτελεσματικές λύσεις σε ένα ευρύ φάσμα προβλημάτων. Η μέθοδος που θα αναπτύξουμε στην επόμενη παράγραφο ανήκει στις pertubation, model agnostic instance level τεχνικές. Αυτό σημαίνει πως θα προσπαθήσουμε να παράγουμε αντιπαραδείγματα για όλες τις εισόδους των δεδομένων μας ξεχωριστά, χωρίς να χρησιμοποιούμε τις εσωτερικές αναπαραστάσεις του δικτύου αλλά αντιμετωπίζοντας το σαν black-box.

Ένα **αντιπαράδειγμα (Counterfactual Example)** είναι μια παραλλαγμένη εκδοχή $\mathbf{x}'$ μιας αρχικής εισόδου $\mathbf{x}$ που οδηγεί ένα οποιοδήποτε black box μοντέλο Φ σε διαφορετική πρόβλεψη [14]. Συγκεκριμένα, αν το μοντέλο Φ αποτυπώνει δεδομένα $\mathbf{x} \in \mathbb{R}^d$ σε μια πρόβλεψη $\hat{y} = \Phi(\mathbf{x})$, τότε για ένα αντιπαράδειγμα $\mathbf{x}'$ ισχύει:

$$\Phi(\mathbf{x}) \neq \Phi(\mathbf{x}').$$

Με άλλα λόγια, οι Εξηγήσεις με Αντιπαραδείγματα προκύπτουν όταν η νέα είσοδος $\mathbf{x}'$ οδηγεί το μοντέλο πάντα σε διαφορετική πρόβλεψη σε σχέση με την αρχική είσοδο $\mathbf{x}$. Στη περίπτωση της ταξινόμησης γραφημάτων, όπου $x = G$ τότε και το αντιπαράδειγμα είναι γράφημα $x' = G'$, και η βασική απαίτηση είναι να αλλάξει η προβλεπόμενη κλάση διατηρώντας παράλληλα τις αλλαγές στο ελάχιστο.

Ορίζοντας μια συνάρτηση απόστασης $\mathcal{D}$ σχετική με το εκάστοτε πρόβλημα, απαιτείται να ισχύει:

$$\mathcal{D}(x, x') \leq t,$$

όπου t είναι ένα κατάλληλο όριο (threshold) που διασφαλίζει την ελαχιστοποίηση των αλλαγών. Στο πλαίσιο της δημιουργίας εξηγήσεων με αντιπαραδείγματα σε δεδομένα γράφων, η διαδικασία μπορεί να διατυπωθεί ως μια ελαχιστοποίηση της παρακάτω συνάρτησης κόστους, η οποία ισορροπεί την απόσταση $\mathcal{D}(\mathbf{x}, \mathbf{x}')$ και την απώλεια $\mathcal{L}$ μεταξύ $\Phi(\mathbf{x}')$ και $\Phi(\mathbf{x})$:

$$\arg\min_{\mathbf{x}'} \mathcal{L}\big(\Phi(\mathbf{x}'), \Phi(\mathbf{x})\big) + \lambda \cdot \mathcal{D}(\mathbf{x}', \mathbf{x}),$$

όπου $\mathcal{L}$ είναι μια διαφορίσιμη συνάρτηση (συχνά επιλέγεται η cross-entropy loss), ενώ λ είναι μια υπερπαράμετρος που καθορίζει το βάρος ανάμεσα στην αλλαγή της πρόβλεψης και στην εγγύτητα με την αρχική είσοδο $\mathbf{x}$ [15, 41]. Στην πράξη, στην παραπάνω συνάρτηση κόστους μπορούν να προστεθούν επιπλέον όροι, ώστε να επιτυγχάνεται δράσιμη καθοδήγηση (actionability) ή κάποιος συγκεκριμένος στόχος. Αυτή ακριβώς την πρακτική αξιοποιούμε και στην παρούσα εργασία, για να εξασφαλίσουμε ότι οι Εξηγήσεις με Αντιπαραδείγματα είναι όχι μόνο εφικτές, αλλά και ουσιαστικές.

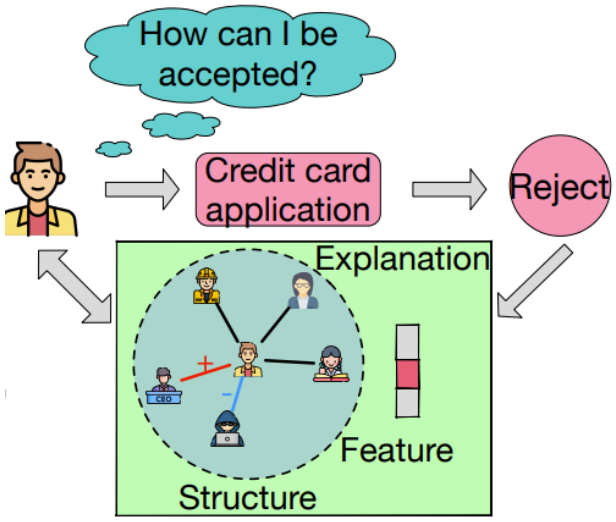


Figure 0.1.3: Παράδειγμα που αναδεικνύει την χρησιμότητα των Εξηγήσεων με Αντιπαραδείγματα. [15]

0.2 Προτεινόμενη Μεθοδολογία

Ξεκινώντας έχουμε τον καθορισμό μιας αρχιτεκτονικής υψηλού επιπέδου, η οποία χρησιμοποιείται ευρέως στην βιβλιογραφία, ικανής να παράγει Εξηγήσεις με Αντιπαραδείγματα. Ξεκινάμε ορίζοντας το σύνολο δεδομένων μας, έναν black-box ταξινομητή Νευρωνικών Δικτύων Γράφων (GNN) Φ, καθώς και την κλάση μας που θα παράγει τις εξηγήσεις (explainer). Αρχικά, εκπαιδεύουμε το GNN μοντέλο πάνω στο σύνολο δεδομένων για την εκάστοτε εργασία ταξινόμησης (είτε γράφου είτε κόμβου). Αφού το μοντέλο παράξει τις προβλέψεις $\Phi(x)$ για οποιοδήποτε δείγμα εισόδου, οι εσωτερικές παράμετροί του (βάρη και μετατοπίσεις) παραμένουν σταθερές, ώστε να διασφαλιστεί η συνέπεια των πειραμάτων.

Στη συνέχεια, η κλάση του εξηγητή λαμβάνει ως είσοδο:

1. Την αρχική είσοδο $\mathbf{x}$.

2. Την πρόβλεψη του GNN για αυτό το δείγμα, $\Phi(\mathbf{x})$.

3. Οποιαδήποτε χρήσιμη πρότερη γνώση που προκύπτει από επεξεργασία του συνόλου δεδομένων.

Ο explainer δημιουργεί στη συνέχεια ένα υποψήφιο αντιπαράδειγμα $\mathbf{x}'$ μέσα από επαναλαμβανόμενη αλληλεπίδραση το GNN, σε μια διαδικασία που θα αναλυθεί στην επόμενη παράγραφο. Τέλος, αυτό το νέο παράδειγμα $\mathbf{x}'$ εισάγεται εκ νέου στο μοντέλο GNN, με στόχο να παραχθεί διαφορετική ετικέτα πρόβλεψης:

$$\Phi(\mathbf{x}) \neq \Phi(\mathbf{x}').$$

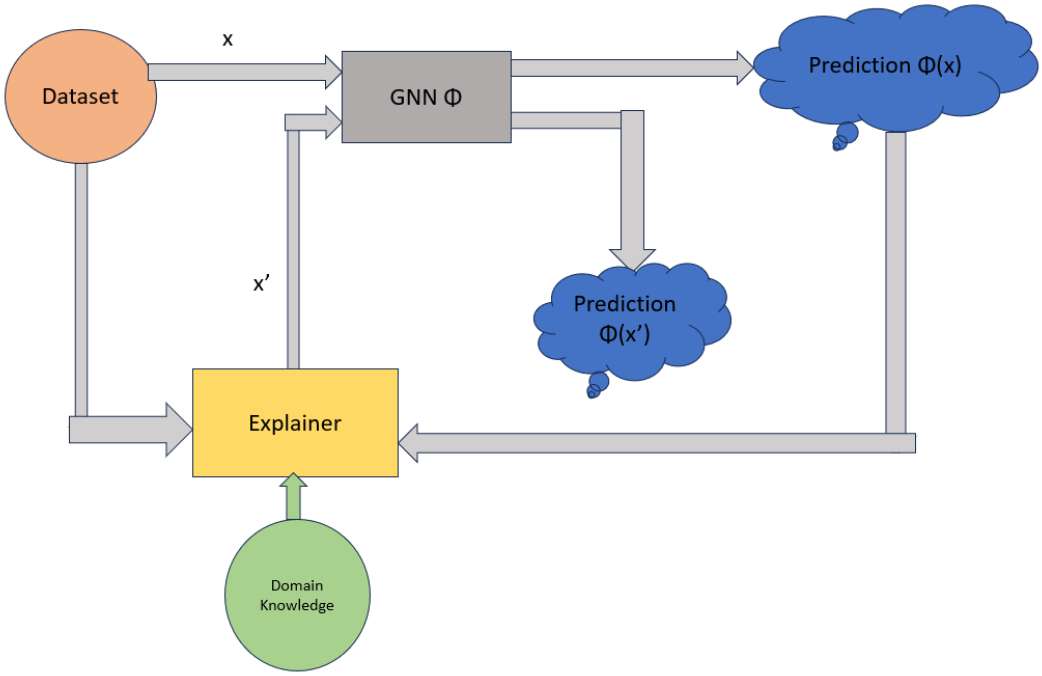


Figure 0.2.1: Η high level αρχιτεκτονική για την παραγωγή αντιπαραδειγμάτων. Το GNN μοντέλο αρχικά παράγει μια πρόβλεψη για οποιαδήποτε είσοδο από το σύνολο δεδομένων. Έπειτα ο explainer χρησιμοποιεί αυτή την πρόβλεψη καθώς και πρότερη γνώση για την παραγωγή αντιπαραδειγμάτων.

0.2.1 Περιγραφή Μεθόδου Διαγραφής Ακμών

Η κεντρική ιδέα του explainer βασίζεται στην ελάχιστη τροποποίηση ενός γράφου προκειμένου να αλλάξει η πρόβλεψη ενός GNN μοντέλου. Οι τροποποιήσεις αυτές, στην περίπτωση μας, αφορούν την αλλαγή της τοπολογίας του γράφου μέσω της διαγραφής ακμών. Για να επιτευχθεί αυτό, ο explainer εφαρμόζει μια *soft masking* τεχνική στις ακμές του γραφήματος, επιτρέποντας τη σταδιακή (και διαφορίσιμη) εξασθένιση ή διατήρηση των ακμών, μέχρι τελικά να γίνει η επιλογή για το ποιές ακμές θα διαγραφούν.

Όσον αφορά την διαδικασία του soft ή continuous masking αρχικά ορίζονται δύο πίνακες ή μάσκες, Z και M, που έχουν τις ίδιες διαστάσεις με τη μήτρα γειτνίασης A και τον πίνακα ακμών (edge index). Αρχικά, η Z αρχικοποιείται στο μηδέν ($z_i = 0$), ενώ στη συνέχεια τροφοδοτείται στη σιγμοειδή συνάρτηση:

$$m_i = \sigma(z_i) = \frac{1}{1 + e^{-z_i}},$$

παρέχοντας έτσι τις τιμές της μάσκας M, $m_i \in [0, 1]$. Αυτό μας επιτρέπει να αλλάζουμε την τιμή του βάρους κάθε ακμής (edge weight) προοδευτικά, χωρίς να απαιτείται διακριτή διαγραφή κάθε φορά (μη-διαφορίσιμη διαδικασία). Αρχικοποιώντας όλες τις τιμές z_i στο μηδέν, η sigmoid παρέχει μια ουδέτερη πιθανότητα 0.5 για κάθε ακμή, επιτρέποντας ίσες πιθανότητες διατήρησης ή διαγραφής.

Προκειμένου οι τιμές της μάσκας μας να μπορέσουν να ανανεωθούν σε κάθε επανάληψη της διαδικασίας χρησιμοποιούμε την εξής συνάρτηση κόστους:

$$\mathcal{L}(Z) \;=\; \mathcal{L}_{\text{pred}}\big(\Phi(G; M(Z)), \hat{y}\big) \;+\; \lambda \cdot \mathcal{L}_{\text{dist}}\big(M(Z)\big),$$

όπου:

- $\mathcal{L}_{\text{pred}}$ είναι ένας όρος cross-entropy loss, $-\log P_\Phi\big(\hat{y} \mid G; M(Z)\big)$, που ωθεί το μοντέλο να προβλέψει την επιθυμητή κλάση $\hat{y}$.

- $\mathcal{L}_{\text{dist}}$ είναι ένας όρος ποινής (penalty) που ενθαρρύνει την διαγραφή όσο το δυνατόν λιγότερων ακμών.

$$\mathcal{L}_{\text{distance}}(M) = \sum_{e \in E} (1 - m_e)$$

Μετά από μια εμπρόσθια διέλευση (forward pass) στο μοντέλο GNN, ακολουθεί οπίσθια διάδοση (backpropagation) που ενημερώνει τη μάσκα Z ανάλογα με την πρόβλεψη που παρείχε το μοντέλο στον γράφο με τα καινούργια edge weights:

$$Z^{(t+1)} = Z^{(t)} - \alpha \, \nabla_Z \, \mathcal{L}^{(t)},$$

όπου α είναι ο ρυθμός εκμάθησης (learning rate). Η διαδικασία αυτή επαναλαμβάνεται για έναν δεδομένο αριθμό επαναλήψεων.

Αφότου ολοκληρωθούν αυτοί βρόχοι εκπαίδευσης, η μάσκα $M(Z)$ περιέχει συνεχείς τιμές $m_e \in [0, 1]$. Σε εκείνο το στάδιο δεν έχουν ακόμα διαγραφεί ακμές αλλά μόνο εξασθενήσει η ισχυροποιηθεί τα βάρη τους. Για να οριστικοποιηθεί η διαγραφή, επιβάλλουμε μια δυαδική μάσκα M_{binary} με χρήση ενός ορίου τ:

$$m_e^{\text{binary}} = \begin{cases} 1, & m_e > \tau \\ 0, & \text{αλλιώς} \end{cases}$$

Με αυτόν τον τρόπο, κάθε ακμή με $m_e > \tau$ διατηρείται, ενώ οι υπόλοιπες διαγράφονται. Αντί να ορίσουμε $\tau = 0.5$ όπως συχνά συναντάμε στην βιβλιογραφία, ο explain αναζητά τη μικρότερη τιμή τ που οδηγεί σε αλλαγή κλάσης (flip) της πρόβλεψης του GNN, διαγράφοντας τον ελάχιστο απαραίτητο αριθμό ακμών. Αφού δημιουργήσουμε τον καινούργιο γράφο με τις διαγεγραμμένες ακμές να λείπουν, τον δίνουμε ως είσοδο στο GNN μοντέλο προκειμένου να κάνει πρόβλεψη. Αν η πρόβλεψη είναι διαφορετική από την αρχική έχουμε παράξει ένα επιτυχές αντιπαράδειγμα. Καθώς αυξάνει σταδιακά το τ, και εφαρμόζεται αυτή η διαδικασία, η αναζήτηση σταματά όταν διαπιστώσει ότι η πρόβλεψη $\Phi(G')$ είναι διαφορετική για πρώτη φορά. Δοκιμάζοντας τις τιμές $\tau = [0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5]$ με αυτή τη σειρά σιγουρεύουμε πως αν βρεθεί αντιπαράδειγμα, θα είναι με τις ελάχιστες δυνατές αλλαγές στην τοπολογία του γράφου.

Χάρη στην συνεχή μάσκα, ο αρχικός γράφος μένει σταθερός καθ' όλη τη διαδικασία (τα βάρη των ακμών μεταβάλλονται), βελτιώνοντας έτσι την υπολογιστική απόδοση σε σύγκριση με το να ξαναχτίζαμε εξ ολοκλήρου το γράφημα σε κάθε βήμα. Επιπλέον, επειδή ο ορισμός της μάσκας βασίζεται σε διαφορίσιμες συναρτήσεις (sigmoid), διατηρείται η δυνατότητα ροής παραγώγων (gradient flow) και συνεπώς η βελτιστοποίηση είναι αποτελεσματική για μεγάλα γραφήματα. Τελικά, το παραγόμενο αντιπαράδειγμα είναι το ελάχιστο δυνατό, αφού διαγράφει μόνο τις απαραίτητες ακμές για να επιτευχθεί αλλαγή πρόβλεψης.

0.2.2 Εισαγόμενοι Περιορισμοί στην Συνάρτηση Κόστους

Συνεχίζουμε στην πραγματική συνεισφορά αυτής της εργασίας, που είναι η εισαγωγή περιορισμών, στην συνάρτηση κόστους, που περιλαμβάνουν πρότερη γνώση. Η προσθήκη επιπλέον όρων στη συνάρτηση κόστους στοχεύει στην κατευθυνόμενη διαγραφή ακμών από τον explainer με σκοπό πιο στοχευμένες εξηγήσεις. Η εισαγωγή τέτοιων όρων δεν είναι απαραίτητο πως θα έχουν θετική επίδραση καθώς πέρα από ενδελεχή πειράματα, πρέπει αυτοί οι περιορισμοί να αντικατοπτρίζουν πραγματικές ιδιότητες του συνόλου δεδομένων μας. Στην συγκεκριμένη εργασία εντοπίσαμε δύο τοπολογικές μετρικές που εμφανίζονται στα δεδομένα που θα παρουσιάσουμε παρακάτω. Συγκεκριμένα, ορίζονται δύο παραδείγματα περιορισμών που αξιοποιούν ιδιότητες γράφων, δίνοντας τη δυνατότητα στο μοντέλο να επικεντρώνεται σε ακμές με υψηλή σημασία.

1. Node Degree Centrality Constraint. Στόχος είναι να προτιμάται η *αφαίρεση* ακμών που συνδέονται με κόμβους-κέντρα (hub nodes), με υψηλή κεντρικότητα βαθμού. Αρχικά, για κάθε κόμβο $v \in V$, υπολογίζουμε τον βαθμό:

$$\text{degree}(v_i) = \sum_j \mathbb{1}[(v_i, v_j) \in E],$$

και στην συνέχεια εντοπίζεται ο κόμβος $v_{\text{max\_deg}}$ με τον μεγαλύτερο βαθμό και συγκεντρώνονται όλες οι ακμές που τον συνδέουν με άλλους κόμβους:

$$E_{\text{hub}} = \{ (v_i, v_j) \in E \mid v_i = v_{\text{max\_deg}} \ \vee \ v_j = v_{\text{max\_deg}} \}.$$

Έπειτα, ορίζεται ένας νέος όρος ποινής (penalty):

$$\mathcal{L}_{\text{degree}} = \sum_{e \in E_{\text{hub}}} m_e,$$

που τιμωρεί την *διατήρηση* αυτών των ακμών ($m_e \approx 1$), ενθαρρύνοντας δηλαδή την αφαίρεσή τους. Η συνολική συνάρτηση κόστους του εξηγητή γίνεται:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{pred}} \ + \ \lambda_{\text{dist}} \cdot \mathcal{L}_{\text{dist}} \ + \ \lambda_{\text{degree}} \cdot \mathcal{L}_{\text{degree}},$$

όπου

- $\mathcal{L}_{\text{pred}}$: Ενθαρρύνει την αλλαγή της πρόβλεψης του GNN στην επιθυμητή κλάση.
- $\mathcal{L}_{\text{dist}}$: Τιμωρεί τη μαζική διαγραφή ακμών, επιδιώκοντας να διατηρηθούν όσες περισσότερες γίνεται.
- $\mathcal{L}_{\text{degree}}$: Προτρέπει την αφαίρεση ακμών που συνδέονται με τον κόμβο μέγιστου βαθμού.

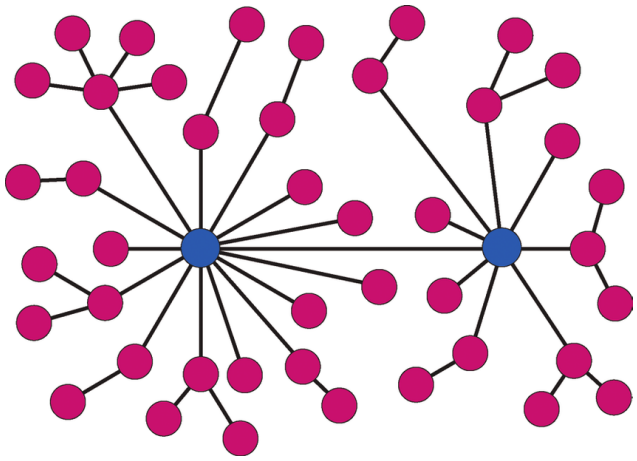


Figure 0.2.2: Παράδειγμα ενός γράφου που περιέχει δύο κόμβους με υψηλότερο βαθμό σε σχέση με τους υπόλοιπους.

2. Edge Betweenness Centrality Constraint. Αντίστοιχα, η μετρική edge betweenness υποδεικνύει πόσο συχνά μια ακμή βρίσκεται σε συντομότερα μονοπάτια μεταξύ ζευγών κόμβων. Έτσι, ορίζεται:

$$C_B(e) = \sum_{v,\, u \in V} \frac{\sigma(v, u \mid e)}{\sigma(v, u)},$$

όπου $\sigma(v, u \mid e)$ είναι ο αριθμός συντομότερων διαδρομών από v προς u που διέρχονται από την ακμή e, και $\sigma(v, u)$ ο συνολικός αριθμός συντομότερων διαδρομών. Επιλέγονται οι ακμές με τιμή $C_B(e)$ πάνω από ένα συγκεκριμένο όριο (π.χ. 90% στα πειράματα μας):

$$E_{\text{bet}} = \{\, e \in E \mid C_B(e) > Q_{0.9}(C_B) \}.$$

Η συνάρτηση ποινής ορίζεται ως:

$$\mathcal{L}_{\text{bet}} = \sum_{e \in E_{\text{bet}}} m_e,$$

και εισάγεται στη συνολική συνάρτηση κόστους:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{pred}} \ + \ \lambda_{\text{dist}} \cdot \mathcal{L}_{\text{dist}} \ + \ \lambda_{\text{bet}} \cdot \mathcal{L}_{\text{bet}},$$

όπου ο όρος λ_{bet} ελέγχει τη βαρύτητα του περιορισμού. Με άλλα λόγια, όσο μεγαλύτερο το λ_{bet}, τόσο περισσότερο ωθείται το μοντέλο να διαγράψει ακμές υψηλού betweenness.

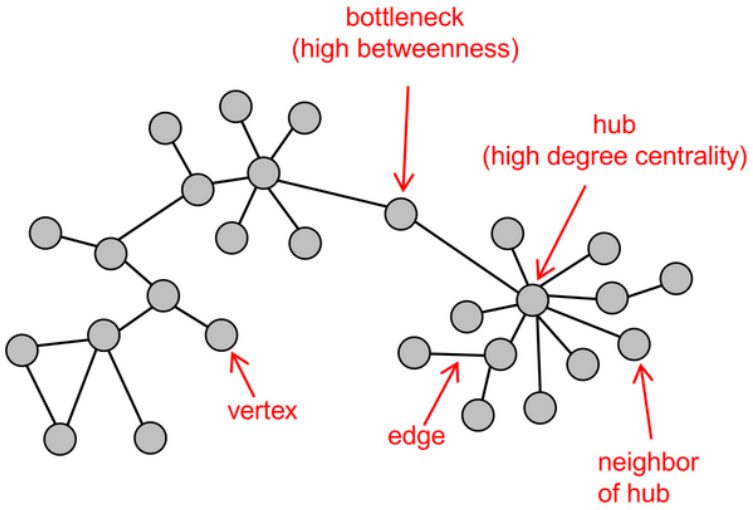


Figure 0.2.3: Παράδειγμα γράφου με ακμή με υψηλό edge betweenness score και κόμβο με υψηλό βαθμό. [34]

Οι συγκεκριμένοι περιορισμοί αποτελούν δομικά παραδείγματα, βασιζόμενα αμιγώς στην τοπολογία (structure) του γράφου. Το πλαίσιο εκπαίδευσης του explainer, ωστόσο, επιτρέπει την εισαγωγή οποιουδήποτε επιπλέον όρου στη συνάρτηση κόστους για να προτιμάται η αφαίρεση ακμών με συγκεκριμένα χαρακτηριστικά. Έτσι, διαμορφώνεται μια ευέλικτη μεθοδολογία που μπορεί να προσαρμοστεί σε διαφορετικές απαιτήσεις, ανάλογα με το εκάστοτε πεδίο γνώσης και τους στόχους της ερμηνείας.

0.3 Πειράματα και Αποτελέσματα

Σε αυτό το κομμάτι της εργασίας θα παρουσιάσουμε το περιβάλλον των πειραμάτων που εκτελέσαμε. Αρχικά θα περιγράψουμε συνοπτικά τα σύνολα δεδομένων που χρησιμοποιήθηκαν καθώς και τις μετρικές που αξιολόγησαν τις μεθόδους μας. Έπειτα θα περάσουμε στα αναλυτικά αποτελέσματα καθώς και στην οπτικοποίηση του trade-off μεταξύ της παραγωγής εξηγήσεων για πολλές εισόδους και τον αριθμό των διαγραφών που πρέπει να εφαρμόσουμε προκειμένου να τις πετύχουμε.

0.3.1 Επισκόπηση των Πειραμάτων

Για τα πειράματα μας, χρησιμοποιήσαμε δύο σύνολα δεδομένων για την ταξινόμηση γραφημάτων, καθώς και ένα για την ταξινόμηση κόμβων.

REDDIT-BINARY

Το συγκεκριμένο σύνολο δεδομένων χρησιμοποιείται κυρίως σε εργασίες ταξινόμησης γράφων. Κάθε γράφημα αναπαριστά ένα νήμα συζήτησης (thread) στο Reddit, όπου οι κόμβοι αντιστοιχούν σε χρήστες και δύο κόμβοι συνδέονται με ακμή αν ο ένας χρήστης απάντησε σε σχόλιο του άλλου. Ο στόχος είναι να ταξινομηθεί ολόκληρο το γράφημα ως ένα νήμα συζήτησης βασισμένο σε διάλογο ή ένα νήμα ερωτήσεων/απαντήσεων.

IMDB-BINARY

Το IMDB-BINARY είναι ένα σύνολο δεδομένων από τη βιομηχανία του κινηματογράφου, αποτελούμενο από τα ego-networks 1.000 ηθοποιών που έχουν συμμετάσχει σε ταινίες που βρίσκονται στο IMDB. Σε κάθε γράφημα, οι κόμβοι αντιστοιχούν σε ηθοποιούς, ενώ μια ακμή υφίσταται όταν δύο ηθοποιοί εμφανίζονται μαζί σε μια ταινία. Αν το ego-network προέρχεται από μια ταινία δράσης (Action), το γράφημα επισημαίνεται με την ετικέτα "Action". Αντίθετα, αν προκύπτει από μια ταινία ρομαντικού περιεχομένου (Romance), σημειώνεται ως "Romance". Συνεπώς, η εργασία ταξινόμησης συνίσταται στο να προσδιορίσουμε αν ένα άγνωστο ego-network προέρχεται από μια ταινία δράσης ή ρομαντική.

	Reddit-Binary	**IMDB-Binary**
# classes	2	2
# graphs	2000	1,000
# node features	0	0
# edge features	0	0
Avg # nodes	429.6	19.8
Avg # edges	497.75	96.5

Table 1: Τα χαρακτηριστικά των δύο συνόλων δεδομένων μας για την ταξινόμηση γράφων. Παρατηρούμε πως το Reddit-Binary περιέχει μεγαλύτερους και πιο πυκνούς γράφους, ενώ το IMDB-Binary περιέχει μικρότερους γράφους. Και τα δύο σύνολο δεδομένων βασίζονται στην τοπολογία τους καθώς δεν διαθέτουν χαρακτηριστικά.

BA-Shapes

Για την εργασία ταξινόμησης κόμβων, αξιοποιήσαμε το συνθετικό σύνολο δεδομένων BA-shapes. Το συγκεκριμένο σύνολο δεδομένων δημιουργείται προσθέτοντας ένα μοτίβο (motif) σε τυχαίους κόμβους ενός βασικού γραφήματος Barabasi-Albert (BA). Συγκεκριμένα, επιλέξαμε ένα τυχαίο γράφημα Barabasi με 1000 κόμβους και προσθέσαμε 400 μοτίβα σε τυχαίους διαφορετικούς κόμβους της βάσης. Κάθε κόμβος ταξινομείται σε μία από τέσσερις ετικέτες: "Non-House" για κόμβους εκτός του μοτίβου, καθώς και "House-Top", "House-Middle" και "House-Bottom" που υποδεικνύουν τη θέση του κόμβου μέσα στο μοτίβο. Το συγκεκριμένο σύνολο δεδομένων χρησιμοποιείται ευρέως σε εργασίες ερμηνευσιμότητας, εξαιτίας της ύπαρξης ετικετών ground-truth σχετικά με τη θέση κάθε κόμβου μέσα στο μοτίβο.

	BA Shapes
# classes	4
# nodes in motif	5
# edges in motif	6
# nodes in total	3000
# edges in total	7691
Avg node degree	5.13

Table 2: Statistics for the BA Shapes dataset.

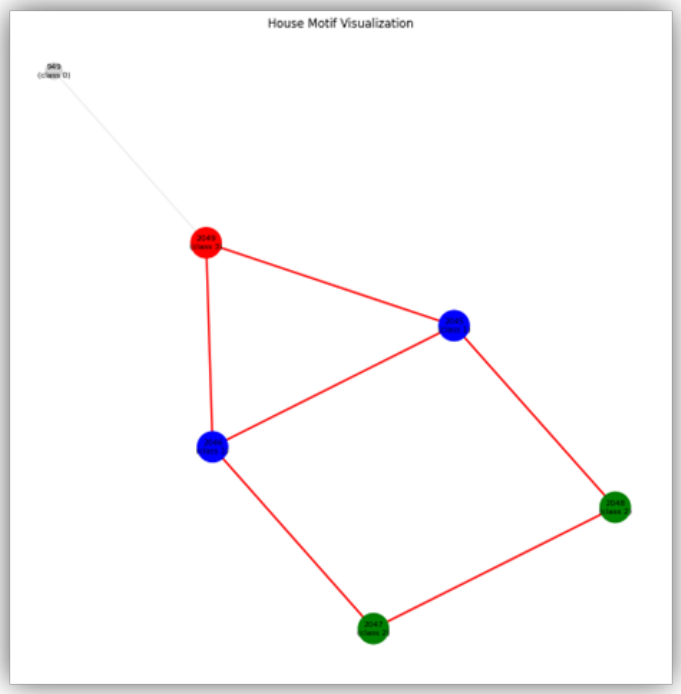


Figure 0.3.1: Το μοτίβο σε σχήμα σπιτιού που συνδέουμε σε τυχαίους κόμβους σε έναν Barabasi Graph.

Για την αξιολόγηση της απόδοσης του explainer μας, χρησιμοποιούμε δύο διαδεδομένους δείκτες ερμηνευσιμότητας σε γραφήματα, το **Ποσοστό Αλλαγής Ετικέτας (Prediction Flip Rate) (PFR)** και την **Απόσταση Επεξεργασίας Γραφήματος (Graph Edit Distance) (GED)**.

Η μετρική PFR καταγράφει το ποσοστό των δειγμάτων (γραφημάτων ή κόμβων) των οποίων η πρόβλεψη άλλαξε με επιτυχία σε σχέση με το σύνολο των δειγμάτων για τα οποία επιχειρήθηκε αλλαγή. Συμβολικά ορίζεται ως:

$$\text{PFR} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}\Big[f(x_i) \neq f(x_i')\Big],$$

όπου:

- N είναι το πλήθος των δειγμάτων στο σύνολο δεδομένων μας.
- $f(x_i)$ είναι η αρχική πρόβλεψη του μοντέλου.
- $f(x_i')$ είναι η πρόβλεψη μετά την τροποποίηση.
- $\mathbb{1}[\cdot]$ είναι η συνάρτηση δείκτης (indicator function).

Στο πλαίσιο αυτής της εργασίας, εκφράζουμε το PFR ως ποσοστό (%) του συνόλου των δειγμάτων που άλλαξαν πρόβλεψη σε σχέση με όλο το σύνολο των δεδομένων που επιχειρήθηκε τροποποίηση.

Το GED ποσοτικοποιεί την ανομοιότητα ανάμεσα σε δύο γραφήματα, μετρώντας τον αριθμό ενεργειών επεξεργασίας που απαιτείται για να μετασχηματιστεί το ένα στο άλλο. Δεδομένου ότι ο explainer μας διαγράφει αποκλειστικά ακμές, ορίζουμε το GED ως τον λόγο των διαγεγραμμένων ακμών προς το σύνολο των ακμών του αρχικού γραφήματος:

$$\text{GED}(G,\ G') = \frac{\text{Number of edges deleted to transform } G \text{ into } G'}{|E|},$$

όπου:

- $G = (V, E)$ το αρχικό γράφημα.
- $G' = (V, E')$ το τροποποιημένο γράφημα.
- $|E|$ ο συνολικός αριθμός ακμών του αρχικού γραφήματος.

Καθώς η συνάρτηση κόστους περιλαμβάνει όρους που αντιστοιχούν τόσο στην ανατροπή της πρόβλεψης (PFR) όσο και στην ελαχιστοποίηση των αλλαγών (GED), προκύπτει αναπόφευκτος συμβιβασμός (trade-off). Πιο συγκεκριμένα, η παράμετρος λ_{dist} καθορίζει πόσο πολύ τιμωρείται η διαγραφή ακμών:

- Αν λ_{dist} είναι μικρό, ο explainer αφαιρεί πολλές ακμές, επιτυγχάνοντας υψηλό PFR αλλά με αυξημένο GED.
- Αν λ_{dist} είναι μεγάλο, οι διαγραφές περιορίζονται, με αποτέλεσμα χαμηλότερο GED αλλά και ταυτόχρονα μικρότερο ποσοστό αλλαγμένων προβλέψεων.

Συνεπώς, για να προσδιορίσουμε τη βέλτιστη ισορροπία ανάμεσα σε PFR και GED, μπορούμε να ορίσουμε μία *συνδυαστική* συνάρτηση, π.χ.:

$$Trade\text{-}off = k \cdot \text{PFR} \; + \; t \cdot \text{GED}, \quad \text{με} \;\; k + t = 1,$$

Ο χρήστης μπορεί να ρυθμίσει τις τιμές k και t αναλόγως της σημασίας που αποδίδει σε καθεμία από τις δύο μετρικές. Για παράδειγμα, $k = t = 0.5$ δίνει ίση βαρύτητα στην ανατροπή της πρόβλεψης και στην ελαχιστοποίηση των διαγραφών.

0.3.2 Αποτελέσματα

Σε αυτή την ενότητα, παρουσιάζουμε ποσοτικά αποτελέσματα για την απόδοση των διαφόρων explainer μοντέλων στα τρία σύνολα δεδομένων. Επιπλέον, απεικονίζουμε αναλυτικά τον συμβιβασμό (trade-off) ανάμεσα στους δύο βασικούς δείκτες Prediction Flip Rate και Graph Edit Distance.

Για τις ενότητες που ακολουθούν, όταν αναφερόμαστε στο Base Model, εννοούμε τον explainer που εφαρμόζει αποκλειστικά τον όρο απόστασης $L1$ (χωρίς πρόσθετους περιορισμούς). Στη συνέχεια, παρουσιάζουμε τα αποτελέσματα για το Node Degree Model και το Edge Betweenness Model, τα οποία ενσωματώνουν τον αντίστοιχο περιορισμό στη συνάρτηση κόστους. Το Base Model λειτουργεί ως σημείο αναφοράς, ενώ τα άλλα δύο μοντέλα βελτιώνουν τα αποτελέσματα των παραπάνω μετρικών.

	REDDIT-BINARY		**IMDB-BINARY**		**BA-SHAPES**	
Method	**Flip_rate↑**	**GED↓**	**Flip_rate↑**	**GED↓**	**Flip_rate↑**	**L-GED↓**
Base Model	53.2	0.2	66	0.12	50.4	0.21
Node_Degree	64.5	0.25	72	0.2	–	–
Edge_Betweenness	50.7	0.2	–	–	51.7	0.2

Table 3: Σύγκριση των μεθόδων στα 3 datasets. Τα αποτελέσματα αυτά βρέθηκαν για $\lambda_{\text{dist}} = 0.01$. Στο Reddit-Binary χρησιμοποιήθηκε $\lambda_{\text{deg}} = 1$ και για το IMDB-Binary $\lambda_{\text{deg}} = 0.1$. Για το μοντέλο Edge Betweenness χρησιμοποιήθηκε $\lambda_{\text{bet}} = 0.01$ και $\lambda_{\text{bet}} = 0.1$ για το Reddit-Binary και το Ba-shapes αντίστοιχα.

Από αυτά τα αποτελέσματα παρατηρούμε ότι το Node Degree μοντέλο παρουσιάζει υψηλότερο Flip Rate σε σχέση με το Base Model, αλλά ταυτόχρονα εμφανίζει αυξημένο GED για την ίδια τιμή της υπερπαραμέτρου απόστασης. Αυτό υποδηλώνει ότι ο συγκεκριμένος περιορισμός οδηγεί τον explainer στην αφαίρεση περισσότερων ακμών, παρέχοντας όμως δυνατότητα εξήγησης σε περισσότερα δείγματα. Πρόκειται για αναμενόμενη συμπεριφορά, καθώς και τα δύο σύνολα δεδομένων ταξινόμησης γραφημάτων περιέχουν κόμβους με υψηλό βαθμό. Για να αλλάξει η προβλεπόμενη κλάση, το Node Degree μοντέλο στοχεύει σε ακμές που συνδέονται με αυτούς τους κόμβους, με αποτέλεσμα να αυξάνεται το GED. Σε πολλές περιπτώσεις, απαιτείται να διαγραφούν περισσότερες ακμές, γεγονός που εξηγεί την αύξηση της απόστασης επεξεργασίας γραφήματος. Επιπλέον, το Node Degree μοντέλο έχει σχεδιαστεί για να βελτιώνει συστηματικά το Flip Rate, θυσιάζοντας όμως μια πιο εκτεταμένη δομική αλλοίωση του γραφήματος. Όπως θα δούμε στη συνέχεια, υπερέχει σταθερά ως προς τον δείκτη Flip Rate συγκριτικά με το Base Model, ωστόσο συνοδεύεται από υψηλότερο GED. Αντίθετα, στο σύνολο δεδομένων

BA-shapes το Node Degree μοντέλο δεν χρησιμοποιήθηκε, επειδή ο βαθμός κόμβου στο μοτίβο (motif) είναι προκαθορισμένος και δεν προσέθετε ουσιαστική πληροφορία σε σχέση με το Base Model.

	REDDIT-BINARY		IMDB-BINARY		BA-SHAPES	
Method	**Flip_rate↑**	**GED↓**	**Flip_rate↑**	**GED↓**	**Flip_rate↑**	**L-GED↓**
Base Model	80	0.32	79.5	0.17	55.6	0.3
Node_Degree	87.5	0.38	84	0.28	–	–
Edge_Betweenness	77	0.29	–	–	54.7	0.28

Table 4: Σε αυτά τα πειράματα χρησιμοποιούμε $\lambda_{\text{dist}} = 0.001$. Οι υπόλοιπες υπερπαράμετροι παραμένουν ίδιες με πριν.

Όπως φαίνεται από τα αποτελέσματα, η τάση που παρατηρήσαμε για το Node Degree Model επιβεβαιώνεται. Αντίθετα, το Edge Betweenness Model καταφέρνει να διατηρήσει το ίδιο Prediction Flip Rate με το Base Model, αφαιρώντας παράλληλα λιγότερες ακμές. Και στα δύο tasks, ο περιορισμός αυτός οδηγεί τον explainer στην αφαίρεση bridge ακμών (ακμές-γέφυρες), οι οποίες διαδραματίζουν κρίσιμο ρόλο στη μετάδοση πληροφορίας στο μοντέλο. Εστιάζοντας σε αυτές τις ακμές, το Edge Betweenness Model υπερέχει του Base Model ως προς το Graph Edit Distance (GED), χωρίς να υστερεί σε Prediction Flip Rate. Αυτό μπορεί να εξηγηθεί καθώς τα σύνολα δεδομένων Reddit-Binary και BA-shapes περιέχουν μεγάλα, πυκνά γραφήματα με πολλαπλές ακμές-γέφυρες. Αντιθέτως, στο IMDB-Binary, το Edge Betweenness Model δεν εμφανίζει βελτίωση έναντι του Base Model, πιθανότατα επειδή δεν υπάρχουν αρκετές διακριτές συντομότερες διαδρομές σε αυτό το σύνολο δεδομένων, με αποτέλεσμα ο συγκεκριμένος περιορισμός να παραπλανά τον explainer.

0.3.3 Συμβιβασμός (Trade-off)

Σε αυτό το σημείο θα προχωρήσουμε στην παρουσίαση της ανάλυσης σχετικά με το trade-off μεταξύ PFR και GED.Έχει καταστεί σαφές ότι οι explainer μας διαθέτουν δύο υπερπαραμέτρους, γεγονός που δυσχεραίνει την παράλληλη απεικόνιση της επίδρασής τους στις μετρικές μας σε έναν μόνο πίνακα αποτελεσμάτων. Επομένως, σε αυτή την υποενότητα, θα παρουσιάσουμε ένα γράφημα που αποτυπώνει τον τρόπο με τον οποίο κάθε υπερπαράμετρος επηρεάζει τον explainer, αναδεικνύοντας το εύρος τιμών που είναι αποτελεσματικές, καθώς και τον συμβιβασμό μεταξύ Prediction Flip Rate (PFR) και Graph Edit Distance (GED).

Αρχικά, παραθέτουμε ξεχωριστά κάθε σύνολο δεδομένων. Για το καθένα, θα υπάρξει ένα διάγραμμα που εμφανίζει τόσο την τιμή του PFR (στον αριστερά άξονα σε κλίμακα (0%–100%), με χρώμα μπλε) όσο και την τιμή του 1-GED (στον δεξιά άξονα σε κλίμακα 0-1, με χρώμα πράσινο), συναρτήσει της κάθε υπερπαραμέτρου (άξονας x). Η αναστροφή του GED σε 1-GED επιλέγεται ώστε να τοποθετούνται οι επιθυμητές υψηλές τιμές κοντά στην κορυφή του γραφήματος, αναδεικνύοντας τη αντισυμμετρική σχέση με το PFR (καθώς το ένα αυξάνεται, το άλλο τείνει να μειώνεται). Για κάθε σύνολο δεδομένων, παρουσιάζουμε αρχικά το Base Model και ακολούθως τις αντίστοιχες γραφικές παραστάσεις για το Node Degree Model και το Edge Betweenness Model. Στις γραμμές του Node Degree Model και του Edge Betweenness Model συμπεριλαμβάνονται επιπλέον τα αποτελέσματα του Base Model (με χρώματα κόκκινο και πορτοκαλί), ώστε να γίνεται ορατή κάθε πιθανή βελτίωση.

Reddit-Binary

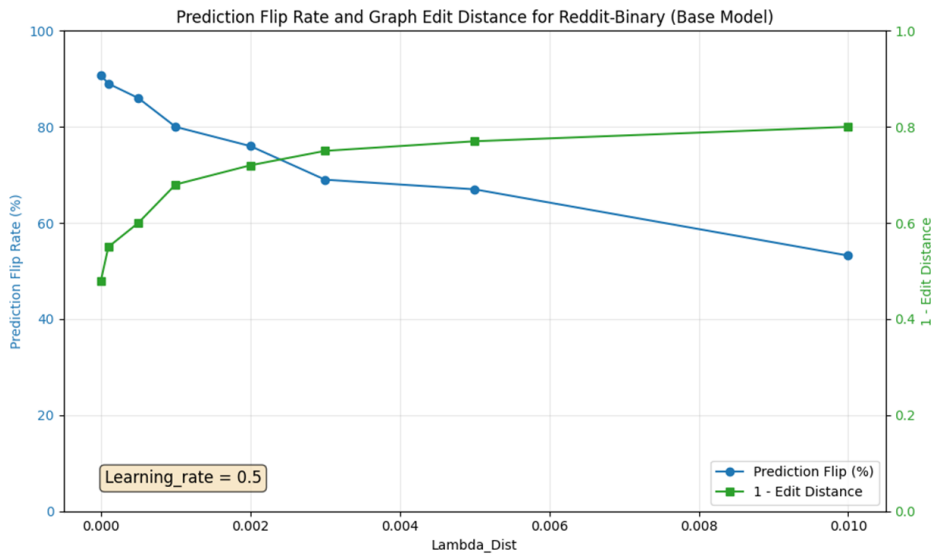


Figure 0.3.2: Σύγκριση μεταξύ του PFR και του 1-GED για το Base model στο Reddit-Binary dataset.

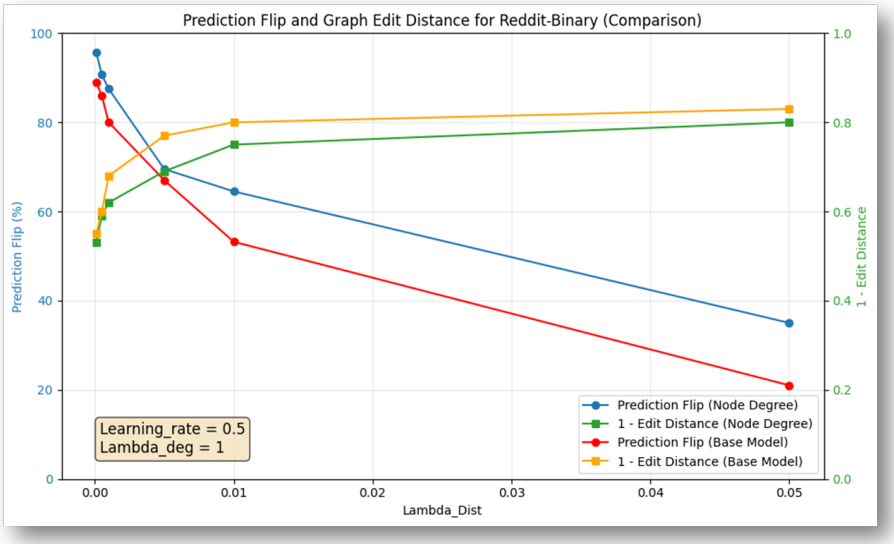


Figure 0.3.3: Αποτελέσματα από την χρήση του Node Degree explainer στο Reddit-Binary dataset.

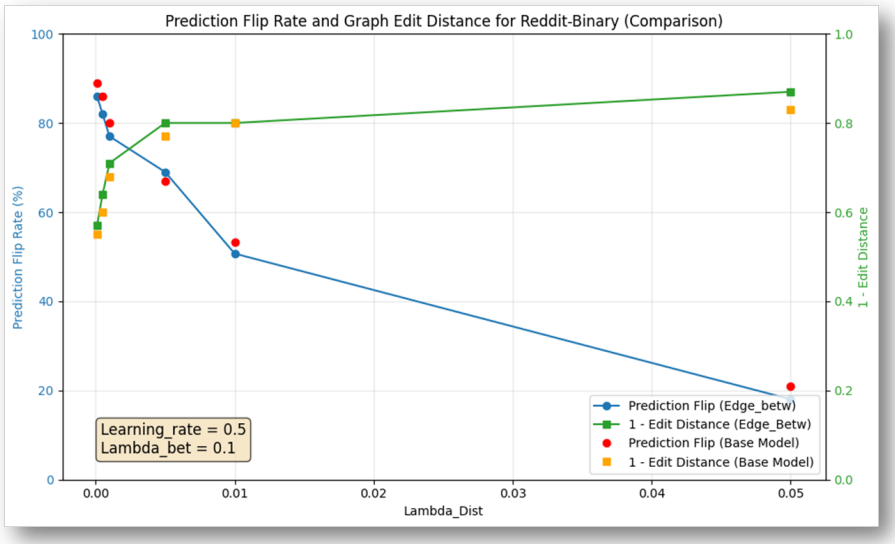


Figure 0.3.4: Αποτελέσματα από την χρήση του Edge Betweenness μοντέλου στο Reddit-Binary dataset.

Συνοψίζοντας τα αποτελέσματα για το Reddit-Binary, διακρίνουμε δύο βασικές βελτιώσεις έναντι του Base μοντέλου. Αρχικά, το Node Degree μοντέλο αυξάνει σημαντικά το PFR, προσφέροντας εξηγήσεις με αντιπαράδειγμα για περισσότερα γραφήματα. Αυτό επιτυγχάνεται με τη διαγραφή πρόσθετων ακμών που το Base μοντέλο δεν εντόπιζε ως κρίσιμες, παρότι χρησιμοποιεί την ίδια υπερπαράμετρο λ_{dist}. Το γεγονός αυτό υποδεικνύει ότι ένας κατάλληλος περιορισμός στη συνάρτηση κόστους μπορεί να εστιάσει σε συγκεκριμένα χαρακτηριστικά του γραφήματος και να ενισχύσει την αποτελεσματικότητα του μηχανισμού των διαγραφών. Στο Reddit-Binary, οι κλάσεις στηρίζονται σε μεγάλο βαθμό στους hub κόμβους, και ειδικότερα στην "Question and Answer" κλάση, ένας τέτοιος κόμβος (χρήστης) απαντά σε ερωτήσεις άλλων χρηστών και έτσι συνδέεται με την πλειονότητα των κόμβων. Στοχεύοντας, λοιπόν, τις ακμές που συνδέονται με αυτόν τον κόμβο, παρατηρούμε πως μπορούμε να αλλάξουμε την προβλεπόμενη κλάση σε περισσότερους γράφους. Αντίθετα, το Edge Betweenness μοντέλο δεν ενισχύει το PFR, αλλά καταφέρνει σε αρκετές περιπτώσεις να επιτύχει χαμηλότερο GED από το Base μοντέλο. Αυτό υποδηλώνει ότι η στόχευση ακμών-γεφυρών (bridges), οι οποίες αποτελούν συνήθως συντομότερες διαδρομές στον γράφο και μεταφέρουν κρίσιμη πληροφορία, μπορεί να περιορίσει τις διαγραφές και να προσφέρει πιο συνοπτικές επεξηγήσεις. Τα γραφήματα του Reddit-Binary είναι μεγάλα και πυκνά, περιλαμβάνοντας πολλές τέτοιες ακμές-γέφυρες, οπότε ένα τέτοιο αποτέλεσμα είναι αναμενόμενο.

IMDB-BINARY

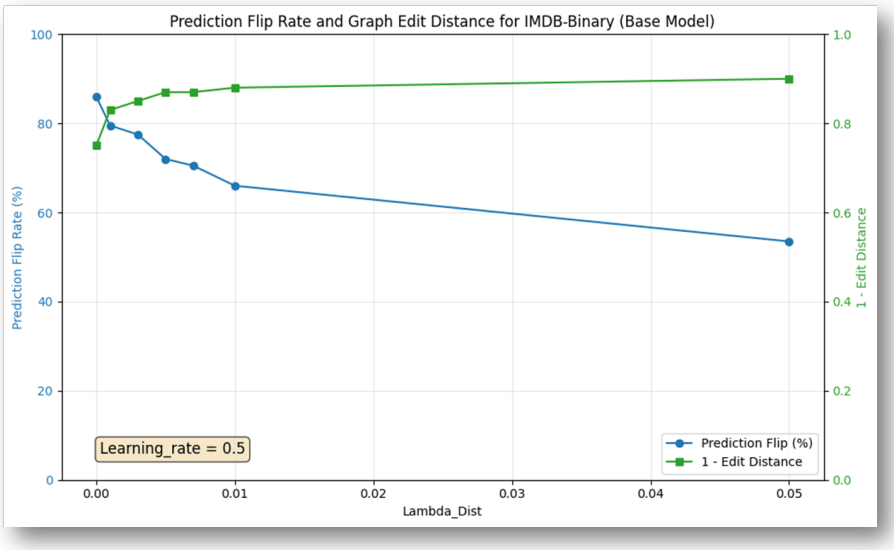


Figure 0.3.5: Αποτελέσματα από την χρήση του Base model στο IMDB-Binary dataset.

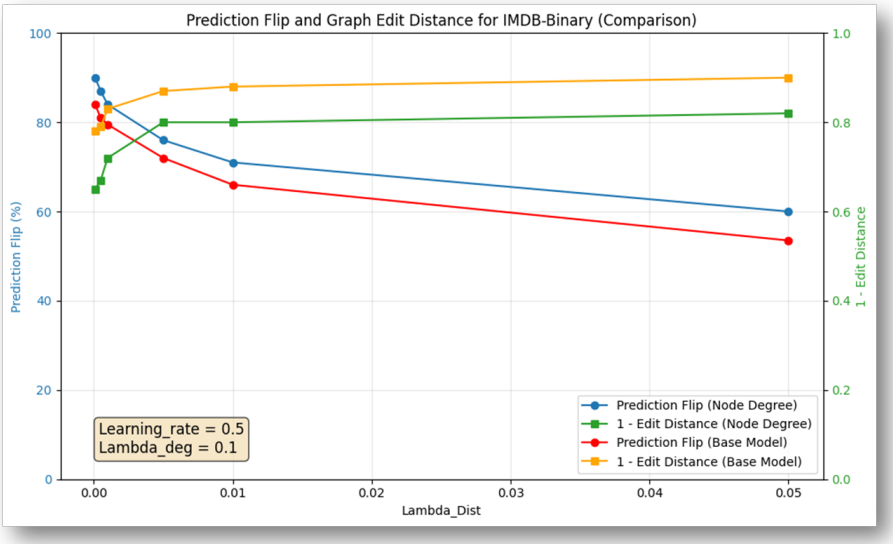


Figure 0.3.6: Αποτελέσματα από την χρήση του Node Degree μοντέλου στο IMDB-Binary dataset.

Το Node Degree μοντέλο παρουσιάζει και πάλι την ίδια συμπεριφορά με προηγουμένως, επιτυγχάνει υψηλότερο PFR από το Base μοντέλο, θυσιάζοντας όμως περισσότερες ακμές. Στο IMDB-Binary dataset, κάθε γράφος αντιστοιχεί στο ego-network ενός ηθοποιού που έχει συμμετάσχει σε ταινίες στο IMDB, με αποτέλεσμα τη συχνή ύπαρξη ενός κόμβου με μεγάλο βαθμό (hub node), που συνδέεται με κάθε άλλον κόμβο. Καθοδηγώντας, λοιπόν, τον explainer να διαγράφει ακμές που συνδέονται με αυτόν τον κόμβο, καταφέρνουμε να παράξουμε εξηγήσεις για περισσότερα γραφήματα. Αυτός είναι και ο λόγος που το Edge Betweenness μοντέλο δεν εντάχθηκε σε αυτό το σύνολο δεδομένων, καθώς η έλλειψη διακριτών συντομότερων διαδρομών καθιστά τον συγκεκριμένο περιορισμό μη αποδοτικό.

BA-SHAPES

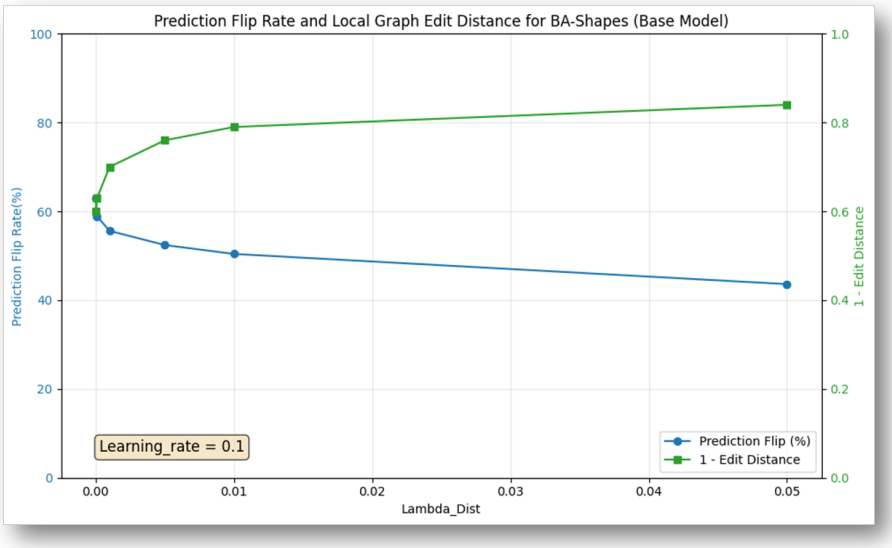


Figure 0.3.7: Αποτελέσματα από την χρήση του Base μοντέλου στο Ba-shapes dataset.

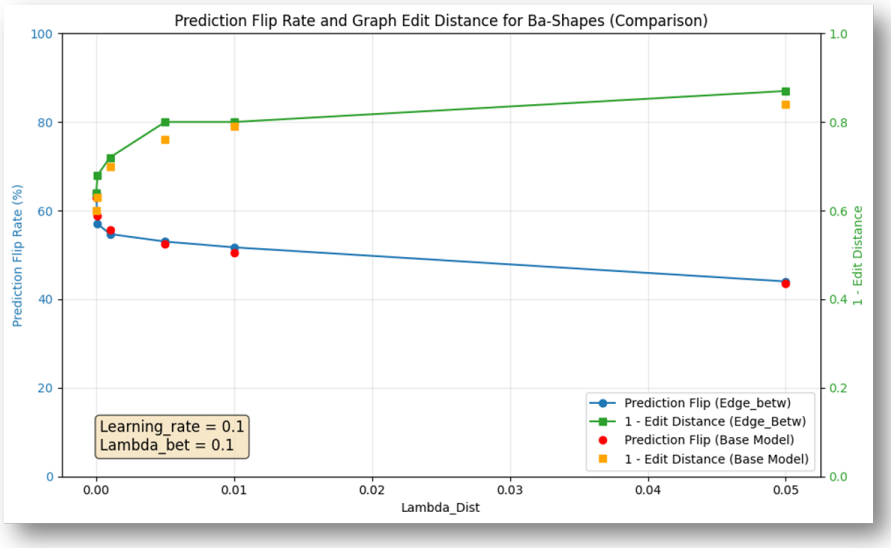


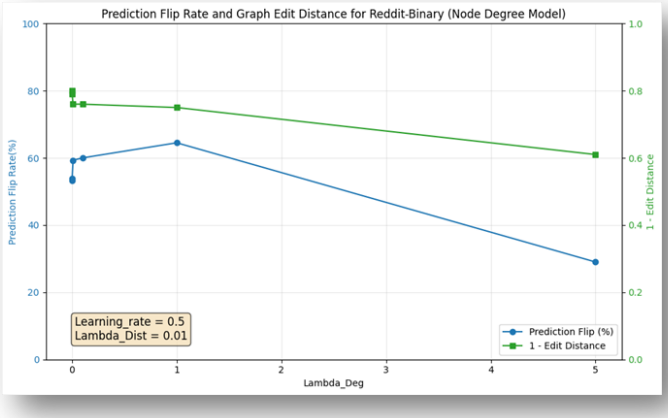
Figure 0.3.8: Αποτελέσματα από την χρήση του Edge Betweenness μοντέλου στο Ba-shapes dataset.

Όσον αφορά το σύνολο δεδομένων Ba-Shapes, παρατηρούμε ότι το Base μοντέλο παρουσιάζει χαμηλότερο PFR συγκριτικά με τα άλλα δύο σύνολα. Αυτό οφείλεται στο γεγονός ότι η συγκεκριμένη εργασία ταξινόμησης κόμβων είναι multi-label. Πιο συγκεκριμένα, στο πλαίσιο της υλοποίησής μας εξετάζουμε την αλλαγή της ετικέτας ενός κόμβου από μία κλάση προς όλες τις υπόλοιπες ξεχωριστά. Έτσι, ο αριθμός πιθανών μεταβάσεων ετικέτας είναι τριπλάσιος (αφού έχουμε 4 κλάσεις) από αυτόν που θα προέκυπτε αν δεχόμασταν μόνο τη "ευκολότερη" μετάβαση κλάσης. Κατά συνέπεια, ο PFR μειώνεται συνολικά, διότι ορισμένες μεταβάσεις ετικέτας είναι δυσκολότερο να επιτευχθούν και μπορεί να απαιτούνται περισσότερες διαγραφές ακμών.

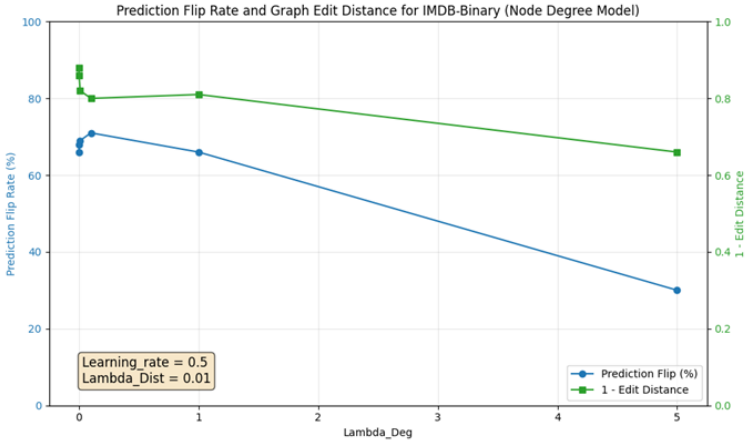
Όπως και στις προηγούμενες περιπτώσεις, διαφαίνεται ο συμβιβασμός (trade-off) μεταξύ PFR και GED: όταν αυξάνουμε την υπερπαράμετρο απόστασης, πετυχαίνουμε λιγότερες εξηγήσεις (χαμηλότερο PFR) αλλά αυτές

είναι συνήθως πιο περιορισμένες σε διαγραφές ακμών (χαμηλότερο GED κατά μέσο όρο). Στο BA-Shapes, το Edge Betweenness μοντέλο ακολουθεί την ίδια λογική: επιτυγχάνει αντίστοιχο PFR με το Base μοντέλο, αλλά με λιγότερες διαγραφές ακμών, καθώς ο συγκεκριμένος περιορισμός τιμωρεί τη διατήρηση ακμών που ανήκουν στις συντομότερες διαδρομές στο τυχαίο Barabasi γράφημα και στα προσαρτημένα μοτίβα.

Τέλος, θα παρουσιάζουμε μια ανάλυση της επίδρασης που ασκούν οι υπερπαράμετροι των περιορισμών στην απόδοση του explainer. Για αυτόν τον σκοπό, κρατάμε σταθερή την υπερπαράμετρο απόστασης και μεταβάλλουμε την υπερπαράμετρο του περιορισμού, παρατηρώντας πώς επηρεάζεται τόσο το Prediction Flip Rate (PFR) όσο και το 1-GED. Η προσδοκία είναι ότι για τιμές πολύ κοντά στο μηδέν, το μοντέλο συμπεριφέρεται σχεδόν σαν το Base μοντέλο, ενώ για πολύ υψηλές τιμές, τα αποτελέσματα υποβαθμίζονται, καθώς η τιμή της συνάρτησης κόστους εκτοξεύεται. Οι βέλτιστες τιμές (που είδαμε και στα προηγούμενα πειράματα) εντοπίζονται συνήθως στο ενδιάμεσο εύρος, όπου το πρόσθετο penalty δεν είναι τόσο υψηλό ώστε να εμποδίζει τις εξηγήσεις, ούτε τόσο χαμηλό ώστε να μη συνεισφέρει στην βελτίωση.

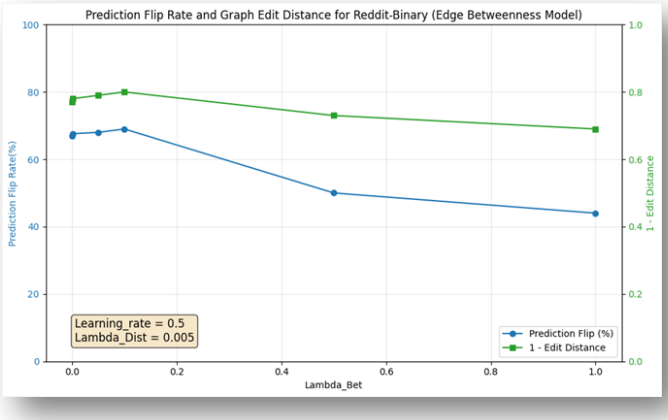


(a) Reddit-Binary dataset.

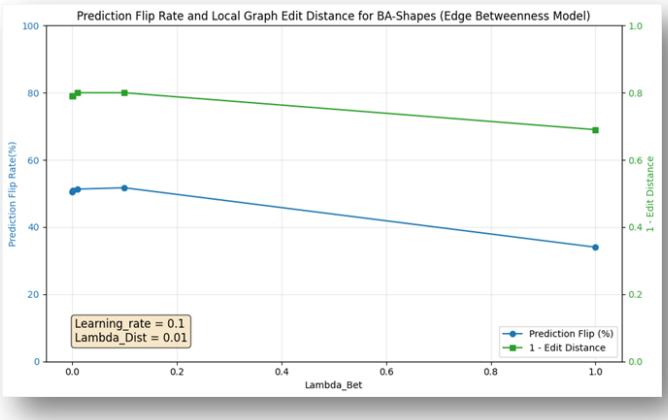


(b) IMDB-Binary dataset

Figure 0.3.9: Αναπαράσταση του λ_{deg} σε σχέση με την απόδοση του Node Degree μοντέλου για το Reddit-Binary και το IMDB-Binary datasets.



(a) Reddit-Binary dataset.



(b) BA-Shapes dataset

Figure 0.3.10: Αναπαράσταση του λ_{bet} σε σχέση με την απόδοση του Edge Betweenness μοντέλου για το Reddit-Binary και το BA-Shapes datasets.

Από την εξέταση των διαγραμμάτων αυτών, αντιλαμβανόμαστε ότι, στις περισσότερες περιπτώσεις, η συμπεριφορά των υπερπαραμέτρων επιβεβαιώνει τις θεωρητικές μας προσδοκίες. Όταν οι υπερπαράμετροι του περιορισμού είναι μηδενικές, κάθε μοντέλο λειτουργεί ουσιαστικά όπως το Base μοντέλο. Με την αύξηση αυτών των τιμών, παρατηρούμε τη συνεισφορά του εκάστοτε περιορισμού, σε ένα συγκεκριμένο εύρος τιμών, το μοντέλο βελτιώνει είτε το Prediction Flip Rate είτε το Graph Edit Distance σε σχέση με το Base μοντέλο. Ωστόσο, μετά από κάποια τιμή, οι υπερπαράμετροι οδηγούν λανθασμένα τον explainer στην αφαίρεση περιττών ακμών ή αποτυγχάνουν να δημιουργήσουν ένα έγκυρο αντιπαράδειγμα.

0.4 Συμπεράσματα

0.4.1 Συζήτηση

Η εξηγήσιμη μηχανική μάθηση (Explainable AI), και ιδιαίτερα ο χώρος των Νευρωνικών Δικτύων με Γράφους (GNNs), εμφανίζει σημαντική προοπτική για περαιτέρω ενίσχυση της εμπιστοσύνης, της διαφάνειας και της λογοδοσίας στα αυτοματοποιημένα συστήματα λήψης αποφάσεων. Καθώς τα μοντέλα τύπου "black box" γίνονται ολοένα πιο περίπλοκα, η κατανόηση του τρόπου λειτουργίας τους καθίσταται κρίσιμη τόσο για τους χρήστες όσο και για εκείνους που επηρεάζονται από τις προβλέψεις τους. Στο ευρύτερο πεδίο της ερμη-

νεύσιμης μηχανικής μάθησης, οι εξηγήσεις με αντιπαραδείγματα (counterfactual explanations) έχουν αναδειχθεί σε πολύτιμο μέσο καθώς δεν περιορίζονται στο να υποδεικνύουν ποια στοιχεία επηρεάζουν την πρόβλεψη, αλλά ταυτόχρονα παρέχουν συγκεκριμένες τροποποιήσεις που μπορούν να μεταβάλουν αυτήν την πρόβλεψη. Με αυτόν τον τρόπο, καθίσταται σαφές πώς μικρές αλλαγές στην είσοδο του μοντέλου μπορούν να αλλάξουν την τελική απόφαση, ενισχύοντας τόσο την ερμηνευσιμότητα των μοντέλων αυτών αλλά και τις δυνατότητες παρέμβασης στην λήψη της απόφασης.

Σε αυτό το έργο, εξετάσαμε την ιδέα των Εξηγήσεων με Αντιπαραδείγματα σε Νευρωνικά Δίκτυα με Γράφους, προτείνοντας μια μεθοδολογία που αφαιρεί επιλεκτικά ακμές από το γράφο ώστε να παραγάγει κατάλληλες εξηγήσεις. Η διαδικασία ξεκινά με μια συνεχή μάσκα (continuous edge masking) και στη συνέχεια ακολουθεί μια διαδικασία δυαδικής αναζήτησης κατωφλίου (binary threshold search), εξασφαλίζοντας ότι διατηρείται η ελαχιστοποίηση των διαγραφών. Επιπρόσθετα, προτείνουμε έναν πρόσθετο όρο στη συνάρτηση κόστους, ώστε να ληφθεί υπόψη η πρότερη γνώση του εκάστοτε πεδίου (domain knowledge). Η συνολική συνάρτηση κόστους συνδυάζει τρία κριτήρια: την επίτευξη αναστροφής της πρόβλεψης, τη διατήρηση μικρού αριθμού τροποποιήσεων (ώστε να υπάρχει σύνδεση μεταξύ του αρχικού γράφου και της εξήγησης) και την ενσωμάτωση περιορισμών που προκύπτουν από χαρακτηριστικά και απαιτήσεις του κάθε προβλήματος. Σε αυτή τη μελέτη, εφαρμόσαμε δύο τέτοιους περιορισμούς: ο πρώτος αξιοποιεί μετρικές βαθμού κόμβων (Node Degree) και ο δεύτερος ενθαρρύνει την αφαίρεση ακμών με υψηλή Edge Betweenness τιμή.

Για την αξιολόγηση της προσέγγισής μας, χρησιμοποιήσαμε τρία σύνολα δεδομένων: ένα για ταξινόμηση κόμβων και δύο για ταξινόμηση γραφημάτων. Χρησιμοποιήσαμε δύο μετρικές: το Ποσοστό Αλλαγής Ετικέτας (Prediction Flip Rate) και την Απόσταση Επεξεργασίας Γραφήματος (Graph Edit Distance). Ο πρώτος καταγράφει το πόσο συχνά μια τροποποίηση αλλάζει επιτυχώς την ετικέτα του μοντέλου, ενώ η δεύτερη μετράει το κόστος των τροποποιήσεων συγκρίνοντας την αρχική με την τροποποιημένη δομή. Ανάμεσα σε αυτές τις δύο μετρικές υπάρχει μια trade-off σχέση: η αύξηση του ρυθμού ανατροπής συχνά προϋποθέτει πιο εκτεταμένες αλλαγές στο γράφημα, ενώ η διατήρηση μικρών διαγραφών μπορεί να μειώσει τον αριθμό των επιτυχών ανατροπών. Κατά συνέπεια, η βελτιστοποίηση των εξηγητικών δυνατοτήτων προϋποθέτει την κατάλληλη διερεύνηση των σχετικών υπερπαραμέτρων και την ανάλυση αυτής της αντίθετης σχέσης ανάμεσα στις δύο αυτές μετρικές.

Εν κατακλείδι, η συμβολή της παρούσας εργασίας είναι πολυπαραγοντική. Πρώτον, εισάγουμε δύο δομικούς περιορισμούς, οι οποίοι, όταν εντάσσονται στη συνάρτηση κόστους του explainer, βελτιώνουν την αναλογία των παραδειγμάτων που επιδέχονται εξήγηση, καθώς και το μέγεθος αυτών των εξηγήσεων. Δεύτερον, πραγματοποιούμε μια αναλυτική μελέτη trade-off, προσδιορίζοντας τα αποτελεσματικά όρια των μοντέλων μας και επιτρέποντας στον χρήστη να επιλέξει την καταλληλότερη ρύθμιση των παραμέτρων για την εκάστοτε περίπτωση. Τρίτον, προτείνουμε ένα ευέλικτο πλαίσιο που διευκολύνει την ένταξη πρόσθετων περιορισμών στη συνάρτηση κόστους, επιτρέποντας την προσαρμογή των εξηγήσεων στις εκάστοτε απαιτήσεις. Με αυτό τον τρόπο, ενισχύεται η ερμηνευσιμότητα των Νευρωνικών δικτύων με Γράφους σε ένα ευρύ φάσμα εφαρμογών.

0.4.2 Επίδραση

Οι εξηγήσεις με αντιπαραδείγματα σε δεδομένα με γράφους έχουν βαθιά επίδραση σε ένα ευρύ φάσμα πρακτικών εφαρμογών, από την ανάλυση κοινωνικών δικτύων και την ανακάλυψη φαρμάκων έως τον εντοπισμό οικονομικών απατών. Σε κάθε μια από αυτές τις περιπτώσεις, η ικανότητα να αναγνωρίζουμε πώς στοχευμένες μεταβολές σε κόμβους ή ακμές μπορούν να αλλάξουν δραστικά το αποτέλεσμα ενός μοντέλου, αποτελεί κρίσιμο βήμα τόσο για την οικοδόμηση εμπιστοσύνης στα μοντέλα όσο και για τη χρήση της Τεχνητής Νοημοσύνης στο ευρύτερο κοινό. Η μέθοδός μας καλύπτει ακριβώς αυτές τις απαιτήσεις, προσφέροντας ερμηνείες που προσαρμόζονται στις ανάγκες του εκάστοτε προβλήματος. Ενσωματώνοντας πρότερη γνώση στον σχεδιασμό της, καθιστούμε εφικτή τη δημιουργία εξηγήσεων που είναι αφενός κατανοητές και αξιόπιστες, και αφετέρου εναρμονίζονται με τα ιδιαίτερα χαρακτηριστικά πολύπλοκων, διασυνδεδεμένων δομών.

0.4.3 Μελλοντικές Κατευθύνσεις

Τέλος, θα θέλαμε να προτείνουμε κάποιες κατευθύνσεις που μπορούν να επεκτείνουν και να βελτιώσουν τη συγκεκριμένη έρευνα. Μια πρώτη προσέγγιση θα ήταν η διεύρυνση του πεδίου εφαρμογής του explainer μας πέρα από τη διαγραφή ακμών, ώστε να περιλαμβάνει και την αλλαγή χαρακτηριστικών τόσο στους κόμβους όσο και στις ίδιες τις ακμές. Στο πλαίσιο αυτής της εργασίας, επιλέξαμε σύνολα δεδομένων με έμφαση στη δομή του γράφου, χωρίς χαρακτηριστικά (features). Εντούτοις, σε πολλά διαφορετικά σενάρια τα δεδομένα εμπεριέχουν

σημαντικά χαρακτηριστικά που θα μπορούσαν να τροποποιηθούν δημιουργικά (feature masking), προκειμένου να παραχθούν πιο στοχευμένες εξηγήσεις.

Μια άλλη σημαντική προοπτική είναι η εισαγωγή περισσότερων σημασιολογικών (semantic) περιορισμών στη συνάρτηση κόστους. Με τον τρόπο αυτό, οι εξηγήσεις με αντιπαραδείγματα θα λαμβάνουν υπόψη όχι μόνο τη δομή του γράφου αλλά και επιπρόσθετες παραμέτρους, όπως πρότερη γνώση, λογικούς κανόνες ή άλλα νοηματικά στοιχεία. Αυτές οι προσαρμογές θα μπορούσαν να προέλθουν από ταξινομητές εκπαιδευμένους ειδικά για τις απαιτήσεις του εκάστοτε προβλήματος.

Chapter 1

Introduction

Advances in machine learning (ML) have led to breakthroughs in several areas of science and engineering, ranging from computer vision and robotics to natural language processing and Large Language Models (LLMs), with important concerns arising along the development of these models [32, 42, 28]. Another research area that has received a lot of attention recently is the one focusing on graph data with complex relationships and structure. Graph Neural Networks (GNNs) have emerged as a promising, novel way of capturing and learning all these high-dimensional relationships whilst being able to make solid predictions on various tasks. However, GNNs have several drawbacks, such as lacking interpretability, easily inheriting the bias of data, and a general black-box behavior that fails to publicize the internal mechanisms of the network that provide predictions.

Given the severity of the tasks handled by AI models, there is a pressing need for methods that can provide clear, actionable insights into the behavior of these models. The field of Explainable AI (XAI) [33] has emerged as a promising area enhancing model interpretability through various forms of explanations, from rule-based methods [21, 29, 22], prototypes [31], and feature importance approaches, to methods tackling multimodal explainability [47]. A widely used method is called counterfactual explanations, which acts opposite to the well-known concept of factual explanations. Factual explanations answer the question: "Given A already happened, will B happen?" , turning the focus on understanding the conditions that made the prediction possible. On the other hand, counterfactual reasoning focuses on: "If A did not happen, will B still happen?"[49], paving the way for understanding the necessary changes we have to make to alter a prediction and therefore understand it. There is a wide range of applications of counterfactual explanations, that affect the desired characteristics of the counterfactuals [27]. Recent works, have studied the evaluation of such explanations in the area of NLP [12], as well as the use of explainability methods in evaluating semantic similarity of visual concepts [25].

In this work, we propose a counterfactual explainer designed to explain any prediction made by a GNN model, by providing thoughtful and actionable examples. These examples act as an answer to the question 'Which aspects of the input should we alter in order to receive a different prediction?', set by counterfactual reasoning. The primary objective of our explainer is the ability to alter any GNNs prediction while making the least amount of modifications possible. The explainer learns a soft edge mask, utilizing domain knowledge which, when applied to the input data, perturbs it by deleting minimal edges, making the model alter its original prediction. This mask is learned by minimizing a multi-objective loss function, consisting of terms the tend for the flipping of the original prediction, the minimality of modifications and any other domain knowledge constrains the user wants to adopt. Afterwards, we apply a binary threshold to that mask, deciding which edges should be removed and which should be kept, and finally we perform an exhaustive search to obtain the optimal threshold. Throughout this process, we balance a delicate trade-off regarding the portion of the graph edges that will be deleted in order to arrive at a reasonable counterfactual example. This approach is model-agnostic as it does not require any of the model's internal representation or weights and can work on the two primary graph prediction tasks, node classification and graph classification.

We evaluate the impact of our explainer on three benchmark datasets, two for the graph classification task

and one for the node classification task. We base our evaluation on the widely used metrics, prediction flip and graph edit distance that encapsulate both the portion of the dataset that our explainer is able to provide explanations for and the minimality of the perturbations, respectively. Our experiments demonstrate that our framework generates counterfactual explanations for the vast majority of the dataset with the least possible modifications, while ensuring that these perturbations are actionable and reasonable.

The outline of this thesis is as follows:

- First and foremost, we provide all the necessary theoretical background regarding Machine Learning algorithms, graph theory basics and Graph Neural Networks.
- Then, we will analyze in depth the characteristics of counterfactual theory as well as highlight the reasons behind its importance for the entirety of Machine Learning.
- Lastly, we will propose our method and analyze all the characteristics that our explainer entails. We will summarize the results on all four datasets and evaluate these results with benchmark metrics. To complete our analysis, we will discuss all the trade-off possibilities and the ways the user can incorporate our explainer into their work.

Chapter 2

Preliminaries - Theory

Machine learning (ML) is a subfield of artificial intelligence (AI) concerned with the design of algorithms that learn patterns from data and make decisions on specific tasks [61]. By analyzing large amount of data, machine learning models can identify trends, detect correlations, and generate insights that surpass traditional statistical methods and compare to human-like thinking and problem solving [13]. Modern day machine learning algorithms have infiltrated social life in the majority of tasks, ranging from everyday socio-economic activities to vast research domains. The basic concepts of AI have become essential knowledge for everyday endeavors, equipping users with a valuable problem-solving tool.

Guided by the advances made in machine learning, researchers have increasingly turned their attention to methods that can handle more complex relational structures and interconnected data. Graph Neural Networks (GNNs) are an example of this effort, as they extend conventional machine learning techniques to graph-structured data by leveraging their rich structural and semantic properties. This capability allows GNNs to exceed the limitations of previous attempts at such data and perform a variety of tasks, such as node classification, link prediction, and graph classification, that were previously unexplored. Recent work and experimentation provided exciting results across fields such as social network analysis, drug discovery, and knowledge graph inference.

As GNNs become more powerful and find applications in crucial decision-making scenarios, there is a pressing need to ensure that their decisions are transparent, interpretable, and fair. With an increase in the dimensionality of the input data comes a greater risk of hidden biases and spurious correlations between different parts of the graph. This concern has led the way for the emergence of Explainable AI (XAI), a field devoted to developing methods that illustrate how predictive models arrive at their conclusions. Among the most promising techniques in XAI for GNNs are counterfactual explanations, which aim to show how slight modifications to a graph's structure or features could alter the model's outcome. By highlighting these pivotal changes, counterfactual explanations offer valuable insight into the inner workings of GNNs, increasing trust and confidence in the models' predictions.

Contents

2.1 Machine Learning

2.1.1 Basic Concepts

Machine learning algorithms can be classified according to the task they are assigned to predict. There exist three basic categories, supervised, unsupervised, and reinforcement learning, while many other subcategories emerge when the training data or the task require special handling. These main categories will be summarized below [61].

Supervised Learning

In supervised learning, models are trained on labeled data, where both the input (features) and the desired output (labels) are known. The algorithm iteratively adjusts its parameters to minimize the discrepancy between predicted and actual labels. Suppose we have a feature vector x and a target label y , in supervised learning the model will try to predict y by learning the properties of the feature matrix x. The most frequent supervised learning tasks are classification into distinct classes and regression. In this work, we will focus on explaining models assigned with supervised learning tasks [63].

Unsupervised Learning

In unsupervised learning, models work with unlabeled data, discovering hidden patterns or groupings. Each time, the model tries to learn the probability distribution of the entire dataset without having access to labels attached to the feature vectors, and thus group input instances according to their properties. The most typical tasks in this category are clustering and dimensionality reduction.

Semi-Supervised Learning

This approach combines both labeled and unlabeled data. A small portion of the labeled data can guide the learning process, while large amounts of unlabeled data reinforce the model's understanding of the underlying structures. Semi-supervised learning is particularly useful when obtaining labels is costly, like in the events of link prediction or fraud detection [55].

Reinforcement Learning

In reinforcement learning, an agent learns by interacting with its environment, receiving rewards or penalties based on its actions. Over time, the agent refines its policy by learning to maximize the reward while minimizing penalties. This method is mostly popular in the field of robotics, where mechanical agents learn to navigate their surroundings or complete a puzzle.

The most frequent architecture that employs such algorithms are Neural Networks whose structure mimics that of the human brain and nervous system.

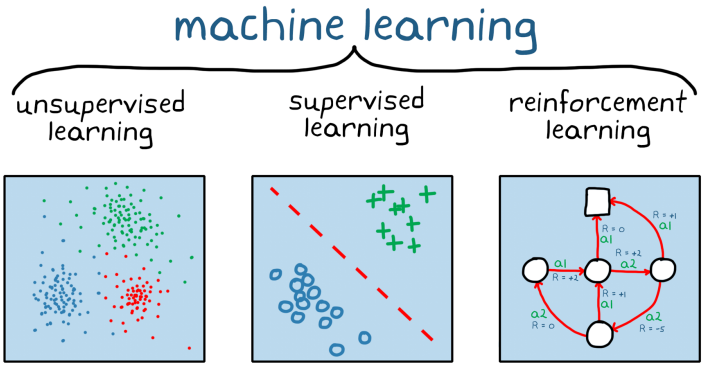


Figure 2.1.1: ML Algorithms [30]

2.1.2 Neural Networks

Neural networks are computational machine learning models inspired by the structure of the human brain. They consist of interconnected nodes (or neurons) organized in layers. Firstly, we will explain the basic concepts of a neural network by presenting the architecture of a single neuron, which is essential in order to perceive more complex architectures [13].

Given a set of data points $D = \{(x_1, y_1), ..., (x_n, y_n)\}$ with N different samples, a neural network model uses these samples and their labels to compute a function $f : X- > Y$ that maps the inputs X_i to an output Y_i using trainable parameters called weights. A simple feed-forward neural network includes:

- An **input layer**, which receives the raw data.
- One or more **hidden layers**, in which neurons progressively learn representations of the data.
- An **output layer**, whose neurons produce the final prediction of the network.

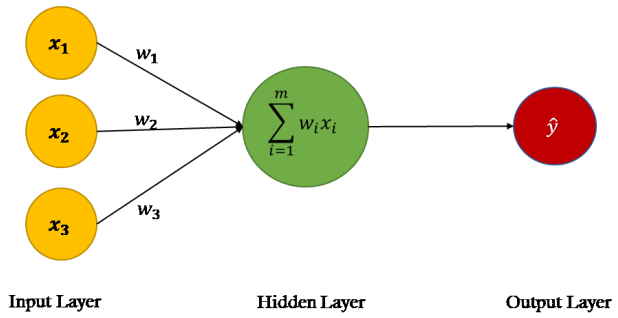


Figure 2.1.2: A single neuron [39]

The functionality of each neuron in any layer l can be described as follows:

$$z_i^{(l)} = \sum_j w_{ij}^{(l)} a_j^{(l-1)} + b_i^{(l)},$$

where:

- $z_i^{(l)}$ is the weighted sum for the i-th neuron in the l-th layer.
- $w_{ij}^{(l)}$ represents the weight from the j-th neuron in the $(l-1)$-th layer to the i-th neuron in the l-th layer.
- $a_j^{(l-1)}$ is the output of the j-th neuron in the $(l-1)$-th layer.
- $b_i^{(l)}$ is the bias term for the i-th neuron in the l-th layer.
- Lastly $a_j^{(l)}$,the output of this neuron, is obtained by applying an activation function to $z_i^{(l)}$.

As described above, once the output of a neuron is computed, it is transferred to the next layer with an **activation function** [56]. This function serves as a mapping mechanism so that the weighted sum is transformed into an appropriate input for the next layer. This activation function is selected by the model's designer and can differ from one layer to the next according to the models needs and parameters. There are no specific rules about the formula of the function but most problems require an activation function with non-constant derivatives to allow for the backpropagation algorithm to work. As shown below, some of the most widely used activation functions are Sigmoid, Tanh, ReLU, and its variants. Each activation function provides unique advantages while also being vulnerable to potential drawbacks, and so they must be chosen carefully.

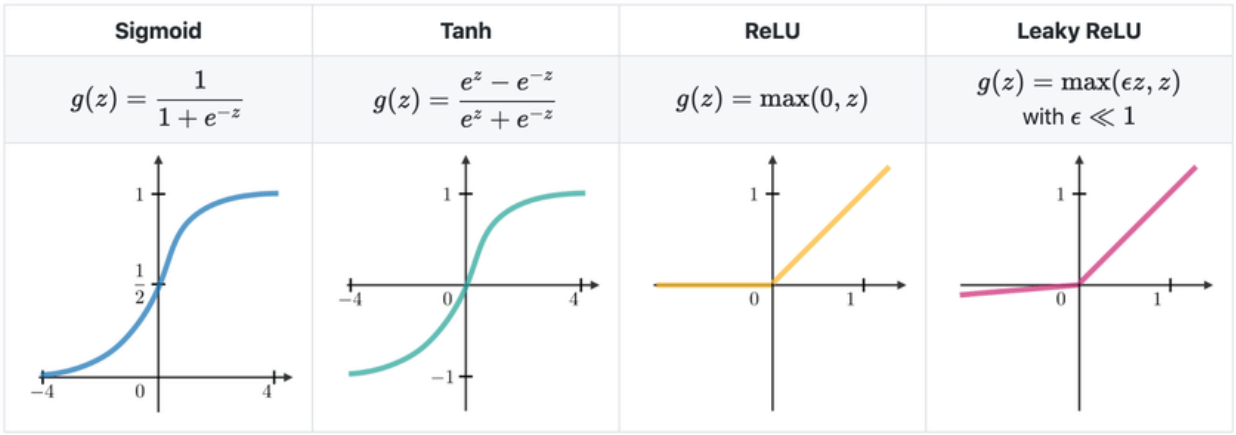


Figure 2.1.3: Frequent Activation Functions

Another critical component of a neural network is the **loss function** [60]. It represents a mathematical tool designed to measure the disparity between the predicted values of a model and the actual (target) values. During the training process, the model's parameters (such as weights and biases) are iteratively refined to minimize this loss, effectively driving the predictions closer to the desired outputs. This process is repeated for a given number of iterations. If not specified otherwise, the total loss of the model in each iteration can be defined as a normalized average of the output of the loss function for each data point in the training set. By quantifying prediction errors, the loss function provides critical feedback to the learning algorithm, guiding corrective adjustments to the model's parameters until optimally a local minimum is reached and the model has been successfully trained. The choice of the loss function requires deep understanding of the task and the other characteristics of the model at hand. Most large-scale problems, like the one in this work, construct custom loss functions that adhere to the specifics of the problem. The most frequent loss functions and building blocks for many custom ones are presented below.

- First, we have the Mean Squared Error (MSE), where $\hat{y}_i$ is the prediction for sample i, y_i is the true label and N is the total number of samples. This loss function characteristically punishes large errors more severely and is often used in regression tasks.

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2,$$

- Then we have the $L1$ Regularization loss [62], which ignores the direction of the error but punishes extreme errors less severely.

$$L1 = \frac{\sum_{i=1}^{n} |y_i - f(x_i)|}{N}$$

- Lastly, the loss function utilized in this work is the Cross-Entropy loss [53]. It originates from information theory as it quantifies the difference between two probability distributions, in our case the predicted probabilities $\hat{y}_i$ and the true probabilities y_i. The key to its use in classification tasks is that it heavily penalizes confident wrong predictions. In other words, if the model assigns a high probability (close to 1) to the incorrect class, the loss grows significantly, driving the parameter update more forcefully towards the desired class in the next iterations.

$$CE = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log(\hat{y}_i + (1-y_i)\log(1-\hat{y}_i\right],$$

The most frequent and effective way of minimizing the loss function in every iteration is through gradient-based algorithms. By iteratively calculating the gradients of the loss with respect to the model's parameters, we can determine how to adjust these parameters to reduce errors. The most widely used algorithm is **Gradient Decent**, which forces the model's parameters θ to be updated opposite to the gradient of the loss:

$$\theta' = \theta - \epsilon \nabla_\theta L(\theta)$$

where ε is a controllable parameter that decides the rate at which the network learns. The gradient-based optimizer used in this work is the Adam optimizer. This optimizer has the ability to adaptively adjust the learning rate for each parameter, allowing it to make larger updates for features that occur infrequently and smaller updates for features that occur frequently. It is selected because of its fast convergence and robustness across a wide range of neural network architectures.

The aforementioned gradients have to be calculated in a cost-effective manner in order to account for systems with millions of parameters and multiple layers. The backpropagation algorithm provides a robust way for the model to calculate all the required gradients needed for parameter updates. In the training stage, the model first performs a forward pass, performing all the calculations for each layer up to the output layer, as we described before. Then, once the **loss function** is calculated at the output layer, the goal is to discover how each parameter (weights and biases) influenced this loss. This is done by moving backward, layer by layer, and computing:

$$\frac{\partial L}{\partial w_{ij}^{(l)}} \quad \text{and} \quad \frac{\partial L}{\partial b_i^{(l)}}$$

for each weight $w_{ij}^{(l)}$ and bias $b_i^{(l)}$ by using the chain rule in reverse, from the output layer to the input layer. calculating how each parameter affects the loss. After these computations, the model parameters are updated in a direction that reduces the loss.

To conclude, we will perform a summary of the entire training process for any Neural Network and highlight its key features.

The process begins with **parameter initialization**, where weights and biases are set to random values. This choice of initialization is crucial to ensure stable and efficient training. Once the parameters are initialized, the **forward propagation** phase takes place: inputs are passed through each layer of the network, where each layer applies its parameters (weights and biases) and an activation function. The output of one layer serves as the input of the next, leading to the final predictions of the model.

Next, these predictions are compared with the true labels by computing the **loss function**. The loss quantifies how far off the predictions are from the true labels. With a measure of this error established, **backpropagation** is then used to systematically compute the gradients of the loss with respect to each parameter, working backward from the output layer to the earlier layers [45]. The chain rule of calculus is applied layer by layer, and these gradients indicate how each weight and bias should be modified to reduce the overall error.

Once the gradients are calculated, the parameters are updated by an **optimization algorithm**, commonly Gradient Descent or one of its variants. The parameters are adjusted in the direction opposite to the gradients in order to decrease the loss. This process repeats over multiple iterations, called **epochs**, each epoch representing a complete pass through the training set.

Finally, once training is complete, the neural network is evaluated on a separate **test set** consisting of unseen data. This step guarantees the model's ability to perform in real-world conditions and confirms that the training process has resulted in an effective output.

2.1.3 Deep Learning

Since we explained the concepts of a simple Neural Network and the process of training, it is time to discuss more complex models and algorithms, designed to tackle more challenging tasks. The most prominent ones in today's research world are Deep Learning algorithms.

Deep Learning relies heavily on the concept of a Multi-Layer Perceptron (MLP) [70], a model introduced to surpass the limitations of linear models, that can only map the output to the same direction as the change in the input. The basic structure of an MLP network consists of multiple layers of nodes in a directed graph, each layer fully connected to the next. The training process remains the same as before, with each node computing the weighted sum of the inputs and producing an output using a non-linear activation function. At its core, an MLP network employs a series of linear transformations followed by non-linear activation functions, enabling the network to approximate complex mappings from inputs to outputs. This architecture equips the user with a variety of options to explore, such as the number of hidden layers, the activation function after each layer, and the input and output size of each layer [54]. This is the driving factor behind Deep Learning algorithms experiencing a huge rise in usage and experimentation, as they provide insightful solutions to many different tasks.

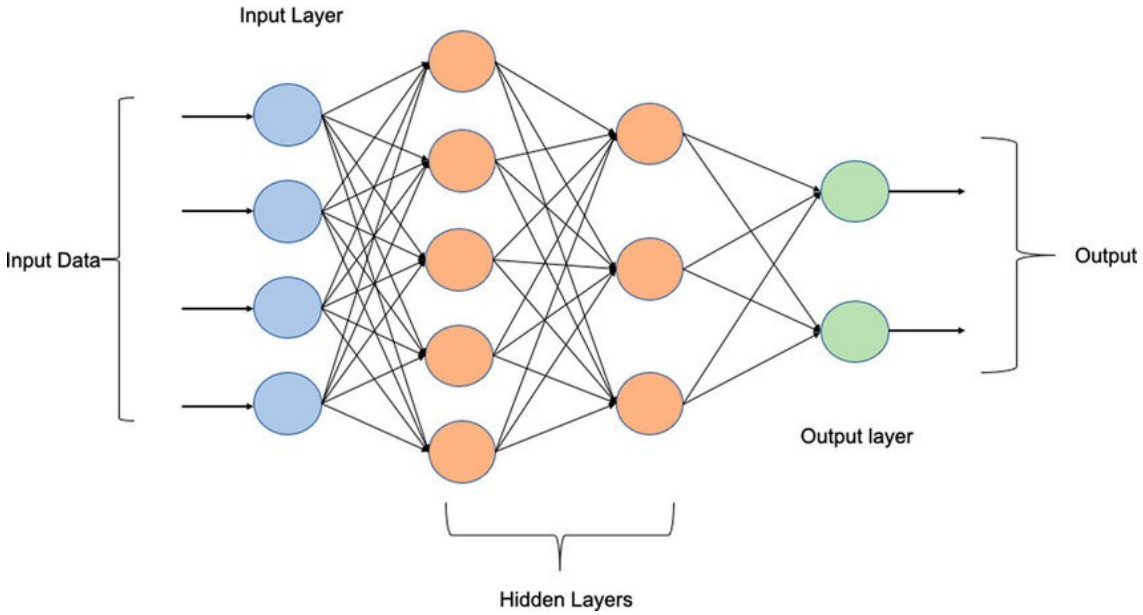


Figure 2.1.4: Multi Layer Perceptron [37]

Another very interesting architecture that relates to the work of this thesis is the Convolutional Neural Network (CNN). This architecture was created to satisfy the need to process high-dimensional data such as images and graphs. An MLP neural network would have to be enormous in size with millions of parameters in order to cope with such a large input. The core concept that allows CNNs to capture the essence of grid-like data is the convolution operation, which involves learnable filters (kernels), sliding across the input to produce feature maps. These convolutional layers consist of many trainable filters that are able to recognize diverse patterns within the input. In practice, a filter is a small matrix of learnable weights (e.g., 3×3 or 5×5 in spatial dimensions) that slides across an input. Afterwards, the dot product between filters and the input across both dimensions is calculated and therefore a 2-dimensional activation map of that filter is produced. Mathematically, if x denotes an input volume (e.g. an image with multiple channels) and w a learnable filter, the output feature map y at position (i, j) can be expressed in the discrete form as:

$$y_{i,j} = \sum_{u=1}^{U} \sum_{v=1}^{V} \sum_{c=1}^{C} w_{u,v,c} \cdot x_{i+u-1,j+v-1,c} + b,$$

where U and V are the height and width of the filter, C is the number of channels and B is a bias term. After a convolution layer, typically follows a pooling layer that reduces the spatial dimensions of the feature maps, thereby providing translation invariance and lowering computational complexity. Everything else resembles a normal MLP network with nonlinear activation functions applied at each layer to model complex patterns and the use of the backpropagation algorithm to adapt filters and biases. This concept will be very useful later when we present the expansion of CNNs to graph data with the introduction of Graph Convolutional Networks.

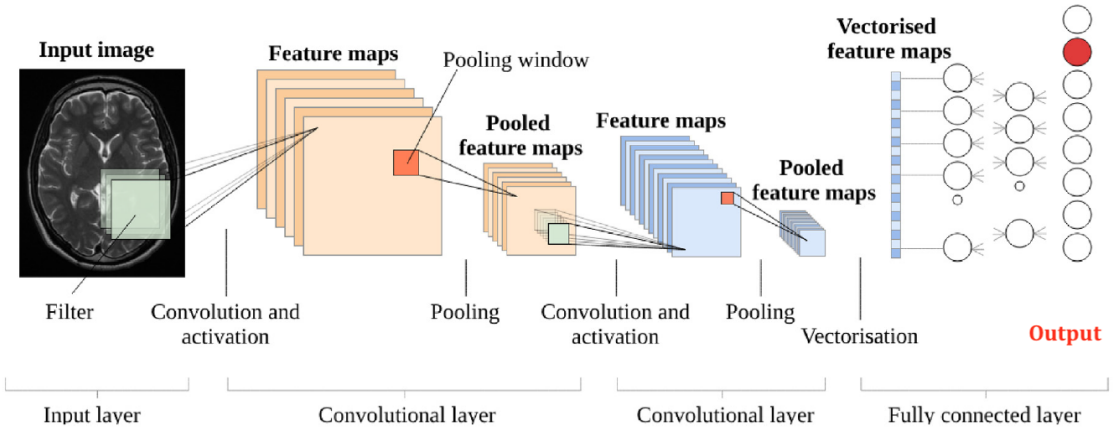


Figure 2.1.5: CNN Architecture [44]

2.2 Graph Neural Networks

2.2.1 Graph Theory

In this section, we will examine the basic concepts of graph theory and the necessary background tools for Graph Neural Networks.

Let's start by denoting a graph object as G. This object is a nonlinear data structure consisting of a set of nodes or vertices V and a set of edges E where:

$$G(V, E) = \{(u, v) : u, v \in V, (u, v) \in E\}$$

Nodes represent any entity or instance, such as a person in a social network, a city in a transportation system, or the atom of a molecule, while edges represent the relationships, connections, or interactions between them. These relationships can be categorized based on their orientation. In undirected graphs, edges have no inherent direction, and any pair of connected vertices can be traversed freely in both directions, making the relationship symmetric, while in directed graphs, edges have a fixed orientation from one vertex to another, typically representing a one-way relationship. Edges can also hold a numerical value that quantifies the cost, capacity, or strength of connection between the vertices it links, further categorizing graphs as weighted or unweighted depending on the existence of these weights.

Another critical aspect of graph data, beyond topological meaning, is that both nodes and edges often carry additional information, called features or attributes. These features could be of any form, numerical or categorical, and are represented through a feature matrix X with dimensions $N * d$ where each of the N rows corresponds to a node and the d columns encode the various attributes or features for that node. A similar principle applies if you assign features to edges where these features are arranged in an edge feature matrix of size $E * d'$ where E is the number of edges and d' is the dimension of each edge's feature vector.

One common representation of a graph G is the adjacency matrix [57] $A \in \mathbb{R}^{n \times n}$ whose entries a_{ij} are defined as:

$$a_{ij} = \begin{cases} 1, & \text{if } \{v_i, v_j\} \in E \\ 0, & \text{otherwise} \end{cases}$$

In an unweighted, undirected graph, a value of 1 indicates the presence of an edge and 0 indicates that there is no edge, leading to a symmetric adjacency matrix, whereas for weighted graphs, a_{ij} may hold any nonnegative value representing the weight of the edge. The concept of the adjacency matrix is very important in this work, as it is the structure that allows for edge modifications.

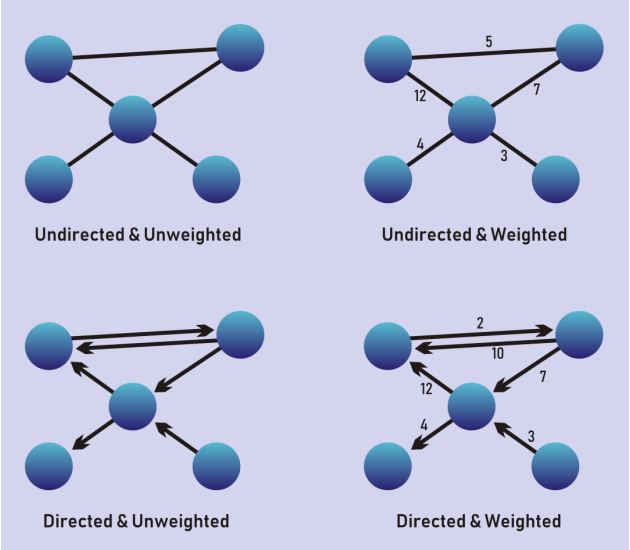


Figure 2.2.1: Different Graph Types [5]

There are many more graph properties that require noting.

- A path is a sequence of vertices such that consecutive vertices in the sequence are connected by edges.
- A cycle is a path whose first and last vertices coincide (and all edges are distinct). A graph is considered connected if there is a path between every pair of vertices.
- The neighborhood $N(u)$ of a node u in a graph G is defined as the subgraph of G induced by all vertices adjacent to u, i.e, the graph composed of the vertices adjacent to u and all edges connecting vertices adjacent to u.
- A shortest path between two vertices u and v in a graph is a path that minimizes the number of edges traversed from node u to node v and vice versa (for undirected scenarios).

These properties lead to very important graph traversal metrics that help quantify the importance of nodes, edges, and their relationship both for the local neighborhood of a node and the entirety of the graph [46, 71]. Some of these metrics will be used by our explainer later in search for more meaningful and actionable explanations.

Node Degree Centrality

The most straightforward type of centrality for a node u is the degree centrality. In an undirected graph, the degree of a node u is the number of edges incident to u. This degree can be calculated as:

$$\deg(v) = \sum_{u=1}^{N} a_{vu}$$

where $a_{vu} = 1$ if there is an edge between u and v and zero otherwise. Nodes with high-degree centrality often assume the role of the distributor of critical information along the graph as they connect multiple nodes together.

Edge Betweenness Centrality

Betweenness centrality for any edge e measures how frequently that edge lies on the shortest paths between pairs of nodes. It is calculated through:

$$\text{BetweennessCentrality}(e) = \sum_{s \neq t \in V} \frac{\sigma_{s,t}(e)}{\sigma_{s,t}}$$

where s and t are two distinct nodes, $\sigma_{s,t}$ is the total number of shortest paths from s to t and $\sigma_{s,t}(e)$ is the number of those paths that pass through e. Edges with high betweenness values often act as a bridge for the flow of information among the network.

Clustering Coefficient

The local clustering coefficient of a node u can be computed by:

$$CC(v) = \frac{\text{number of edges among neighbors of } v}{\binom{\deg(v)}{2}}$$

where $\binom{\deg(v)}{2}$ is the number of total possible edges among the neighbors if they formed a complete subgraph. This metric signifies how well the neighbors of node u are interconnected with a higher clustering coefficient indicating a tightly knit local neighborhood.

Closeness Centrality

Closeness centrality measures how close a node is to all other nodes, on average, by taking the reciprocal of the sum of the shortest path distances from v to every other node u:

$$\text{Closeness}(v) = \frac{1}{\sum_{u \in V} d(v,u)}$$

where $d(v,u)$ is the length of the shortest path from v to u. Nodes with high closeness centrality are typically positioned to spread or receive information rapidly within the network.

These metrics and many others provide us with valuable tools for analyzing the complex relationships of graph objects. Graph data differ from traditional data structures because they explicitly encode relationships between entities rather than simply listing features for each data point. While tabular data is structured in rows and columns, and images arrange data on a grid of pixels, graphs leverage a non-Euclidean topology that focuses on connectivity and the pairwise interactions between nodes. This means that the analysis must account for not only the node-level characteristics but also the structural properties of the graph, leading to insights that might remain hidden if the data were represented only in a 'flat' format. It is these challenges that have led to the creation of Graph Neural Networks, an architecture solely designed to tackle these demanding problems.

2.2.2 GNN Taxonomy

As mentioned above, Graph Neural Networks (GNNs) were developed to address challenges in applying conventional neural network architectures to graph-structured data. Although most machine learning models are designed for structured data, graphs present unique complexities such as unordered structure, variable-sized neighborhoods, and inter-dependencies among nodes [72, 7].

A key challenge in applying machine learning algorithms to graphs is the assumption of instance independence, which does not hold in node-level tasks where nodes are interconnected. Unlike traditional models, GNNs account for these dependencies.

Let's start our analysis by exploring the taxonomic groups of tasks and learning algorithms that GNNs employ.

Learning Algorithms

- Supervised learning tasks focus on labeled data. Provided that the true labels of the instances we want to predict are available, we use supervised learning algorithms to predict the class of previously unseen data, from a given set of potential classes.
- Semi-supervised learning works like supervised learning but with the exception that we do not possess the labels for each instance, making the task of extracting an accurate prediction for unseen data more challenging.

- Lastly, just like in any ML domain, there are many Unsupervised learning tasks, where there is a full lack of labels and the goal of the model is either to capture patterns among the graph data or separate the data in semantic groups based on their properties.

Task Types

GNN models handle a variety of graph tasks, each of which may focus on different attributes of the graph structure.

- Node-level tasks require models that predict the properties of each node in a larger graph. In this case, the instances are the nodes and the model has to either assign each node with a class or calculate a different value for each node. The most common node-level task is the supervised node classification task, which is also being treated in this work.
- Edge-level tasks require models to predict some attribute of each edge in the graph. It is directly analogous to node-level tasks, and the most frequent ones are edge classification and link prediction.
- Finally, we have Graph-level tasks, where the dataset now consists of many graphs instead of a large one, and the model has to assign either a class or a property to each of these graphs. This is done by capturing patterns on a graph level and downsizing them into a single instance at a time. This work also focuses on the most common task, graph classification.

2.2.3 Training a GNN

In this section, we will provide the background for the training process of Graph Neural Networks along with the key concepts of representation learning and message passing [15].

The way most GNNs are usually trained is through a message-passing mechanism that learns a node representation by iteratively aggregating the node's neighborhood information and embeddings. At each layer, a node collects information from its immediate neighbors, transforms or pools these neighbor embeddings, and combines them with its own embedding to produce an updated representation. These learned representations capture both node attributes and local neighborhood information. Let's start by denoting the representation of node u_i in the l-th layer of the GNN model as $h_i^{(l)}$ and the overall representation of all the nodes in the l-th layer as $H^{(l)}$. The set of neighboring nodes for node u_i is $\mathcal{N}_i = \{v_j \in \mathcal{V} \mid (v_i, v_j) \in \mathcal{E}\}$. Each node $v_i \in \mathcal{V}$ in the l-th layer aggregates information from its neighbors as:

$$h_i^{(l)} = \mathrm{UP}^{(l-1)}\left(h_i^{(l-1)}, \mathrm{AGG}^{(l-1)}\left(\left\{h_j^{(l-1)}, \forall v_j \in \mathcal{N}_i\right\}\right)\right)$$

$$= \mathrm{UP}^{(l-1)}\left(h_i^{(l-1)}, \mathbf{m}_{\mathcal{N}_i}^{(l-1)}\right)$$

where AGG is an aggregation function and UP is an update function. The choice of these two functions defines the GNN variant. The most common choices for the aggregation function are the sum, mean, max, or an attention-based method, and for the update function, often a neural network is employed. This message-passing framework is used to get node representations, and most of the time GNNs are used as an encoder to learn these node representations. Training typically proceeds via backpropagation through these layers of neighbor aggregation, using gradient-based optimization on a chosen loss, as is the case for most ML algorithms. The ground-breaking ability of GNNs to exploit both node features and graph topology during learning provides rich, context-aware embeddings that capture local and global structures.

2.2.4 Architectural Variations

Another way to distinguish GNNs is through their architectural differences and variations. In this section, we will present the most common GNN framework taxonomies and also analyze the most influential models [65, 73, 66].

- **Convolutional Graph Neural Networks (ConvGNNs)**. These networks were inspired by the classical CNNs models, as presented before, and they naturally extend the notion of convolution to non-Euclidean graph domains. In essence, they identify the neighborhood of each node, aggregate

messages from neighboring nodes, and update each node's representation. Similar to stacking multiple convolutional layers in CNNs, ConvGNNs layer multiple rounds of neighborhood aggregation, progressively capturing higher-level features at deeper layers. There exist two subcategories of ConvGNNs, **spatial** approaches, which directly implement local aggregations in the node domain, and **spectral** approaches, which treat graph data as signals in the frequency domain. Some of the most impactful GNNs in today's bibliography fall under these two categories and are the subject of this work.

- **Graph Autoencoders (GAEs)**. GAEs adapt the classic autoencoder paradigm to graph-structured inputs. They encode node (or graph) features into a lower-dimensional latent space and aim to reconstruct either the original adjacency relationships, node attributes, or both. This architecture is most often used in tasks such as network embedding and graph generation, where the model learns how to reconstruct nodes and edges.
- **Recurrent Graph Neural Networks (RecGNNs)**. These types of networks, assume that each node repeatedly exchanges messages with its neighbors until the representations converge to a stable equilibrium. This iterative message-passing framework was among the earliest in GNN research and has become a stepping stone for many modern variants.
- **Spatiotemporal Graph Neural Networks (STGNNs)**. This architecture is designed to tackle a very specific problem that differs from others in many ways. STGNNs handle time-evolving graphs whose topology or node/edge features change over time. They capture both critical dependencies, the *spatial* aspect (which parts of the graph are connected or close) and the *temporal* aspect (how these connections and features evolve over time). These models are tasked with scenarios such as traffic forecasting, dynamic social networks, and sensor-based systems, where patterns must be learned both across nodes and across time steps.

Now let's discover the inner workings of the most influential GNN models from the aforementioned taxonomies. We will primarily examine ConvGNNs networks because they are the subject of this work and are also used in a plethora of graph tasks.

Spectral-based ConvGNNs

The networks that fall under this category derive from spectral graph theory by formulating convolution in the graph Fourier domain, utilizing the eigen decomposition of the graph Laplacian. The Laplacian matrix is responsible for providing crucial information about the graph, while the Fourier transform is used to project the graph to an orthonormal space. Given a symmetric normalized Laplacian $L = U\Lambda U^T$, where U is the matrix of eigenvectors ordered by eigenvalues resulting from the factorization of the normalized Laplacian matrix of the input graph and Λ is a diagonal matrix of eigenvalues, the input is a graph signal x that is then transformed into the spectral domain by $\hat{x} = U^T x$. Then the convolution operation is performed using a filter g_θ defined as:

$$g_\theta * x = U g_\theta(\Lambda) U^T x$$

This approach is inspired by classical signal processing techniques, where convolution in the spatial domain corresponds to element-wise multiplication in the frequency domain. However, due to the high computational cost of eigen-decomposition, new methods were proposed that utilize a more efficient choice of filter and will be examined below.

The **Chebyshev Spectral CNN (ChebNet)** [10] is a form of spectral-based Graph Neural Networks that efficiently approximate spectral filters without the need for costly eigen-decomposition of the graph Laplacian. This is being done by expanding the spectral filter g_θ in terms of Chebyshev polynomials, thereby localizing the convolution operation to a finite neighborhood. This convolution can be expressed as:

$$g_\theta * x \approx \sum_{k=0}^{K} \theta_k T_k(\tilde{L}) x$$

where T_k is the Chebyshev polynomial of degree k, θ_k are the learnable coefficients and $\tilde{L} = \frac{2}{\lambda_{\max}} L - I$ is the rescaled Laplacian with $\lambda_{\max}$ being the largest eigenvalue of L. The Chebyshev polynomial for the k-th order can be computed from the above equation as:

$$T_k(\tilde{L}) = U T_k(\tilde{\Lambda}) U^T$$

This formulation reduces computational overhead by avoiding explicit Fourier transforms and also maintains a localized receptive field, which is crucial for capturing local graph structures. This network has found success in domains such as sensor networks, social graphs, and molecular chemistry.

Next we have the **Graph Convolutional Network (GCN)**, that simplifies the spectral convolution [19] operation presented in the Chebyshev network. It essentially embodies a first-order approximation of the Chebyshev polynomial by enforcing $K = 1$ and $\lambda_{\max} = 2$. The convolution operation is then defined as:

$$x *_G g_\theta = \theta \left(I_n + D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \right) x$$

where $D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$ is another representation of the Laplacian matrix using the degree matrix D and the adjacency matrix A. By normalizing both terms, we can write the GCN layer as:

$$H^{(l+1)} = \sigma \left(\tilde{D}^{-1/2} \tilde{A} \tilde{D}^{-1/2} H^{(l)} W^{(l)} \right)$$

where σ is an activation function, $H^{(l)}$ is the feature matrix at layer l and $W^{(l)}$ the trainable weight matrix. This reassignment of the Chebyshev polynomial ensures numerical stability and improved convergence during training. The GCN has been widely adopted across various applications due to its simplicity, computational efficiency, and effectiveness in capturing local graph structure, and it is the main model for which we will try to provide explanations in this work.

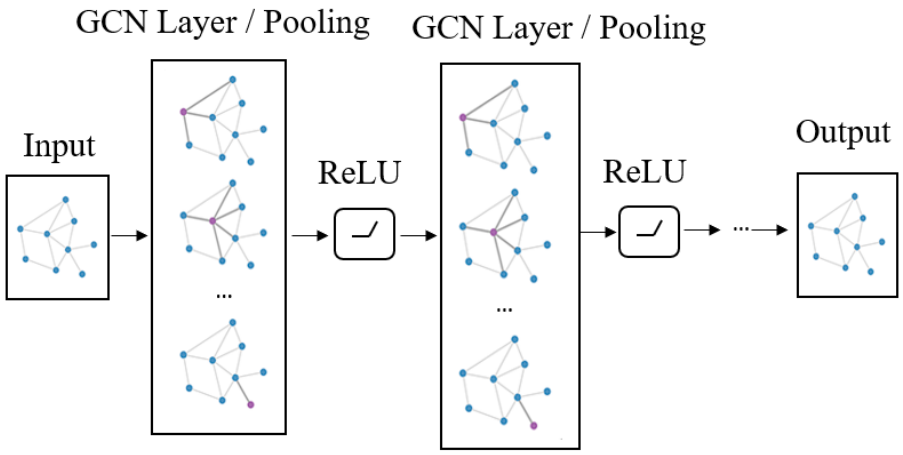


Figure 2.2.2: Convolutional GNN [43]

Spatial-based ConvGNNs

These networks typically operate in a manner similar to that discussed in Section 2.2.3. They define the graph convolution operation taking into account the spatial locality of nodes mimicking the process of message passing on graphs. In these models, each node v updates its feature representation by combining the features of its neighbors $N(v)$ as well as its own features following the formula in Section 2.2.3. Because these convolution-like operations are executed over a node's immediate locality, spatial-based GNNs have been successfully applied in domains such as social network analysis, molecular chemistry, and urban transportation networks, where local interactions are critical to the underlying phenomena. Lets dive into the most used spatial GNN models.

GraphSAGE (Graph Sample and Aggregate) [16] is a spatial-based framework designed to efficiently generate node embeddings for large graphs through neighborhood sampling and a learnable aggregation function. Instead of processing the entire graph, GraphSAGE samples a fixed-size set of neighbors for each node and aggregates their features using various strategies such as mean or max pooling aggregation. The update function for node v can be written as:

$$h_v^{(l+1)} = \sigma \left(W^{(l)} \cdot \left[h_v^{(l)} \parallel \text{AGGREGATE} \left(\{ h_u^{(l)} : u \in \mathcal{N}(v) \} \right) \right] \right)$$

where $h_v^{(l)}$ denotes the feature vector of node v at layer l, $\|$ denotes concatenation, $W^{(l)}$ is a learnable weight matrix, and σ is a non-linear activation function similar to what we have discussed. This model acts as an extension of the GCN to unsupervised learning.

Graph Isomorphism Network (GIN) [67] is a spatial-based model designed to have the ability to distinguish between different graph structures based on the embeddings produced. GIN employs an aggregation scheme that sums up the features of a node and its neighbors before applying a multilayer perceptron (MLP) to update the representation. This process is expressed as:

$$h_v^{(l+1)} = MLP^{(l)}\left(\left(1+\epsilon^{(l)}\right)\cdot h_v^{(l)} + \sum_{u\in\mathcal{N}(v)} h_u^{(l)}\right)$$

where ε is a learnable weight parameter for the central node of the convolution and the summation aggregates the features of the neighbors of v. For each layer, node embeddings are summed, and the result is concatenated. Because this architecture is inspired by the need to distinguish different graph structures and preserve the expressive power that traditional GNNs sometimes lack, it has achieved great results in areas where capturing subtle differences in graph topology is crucial.

Lastly, we have to mention the **Graph Attention Network (GAT)** [50]. This model differs from any other we have mentioned because it employs an attention mechanism into the graph convolution paradigm, allowing nodes to weigh the importance of their neighbors' features dynamically. In GAT, the updated feature of node v is computed based on attention coefficients that measure the relevance between node v and each of its neighbors u, thus making the concept of neighborhood dynamic and not predetermined. Node features are transformed by a shared weight matrix W, followed by the calculation of the attention coefficients:

$$e_{vu} = \text{LeakyReLU}\left(\mathbf{a}^T\left[Wh_v \,\|\, Wh_u\right]\right)$$

where a is a learnable parameter vector and $\|$ denotes concatenation as before. These coefficients are then normalized using a softmax function:

$$\alpha_{vu} = \frac{\exp(e_{vu})}{\sum_{k\in\mathcal{N}(v)}\exp(e_{vk})}$$

The final node representation is obtained by aggregating the transformed features of its neighbors weighted by the attention coefficients:

$$h_v^{(k)} = \sigma\left(\sum_{u\in\mathcal{N}(u)\cup v}\alpha_{vu}^{(k)}W^{(k)}h_u^{(k-1)}\right)$$

By using this adaptive attention weighting, GAT enables the network to focus on the most informative parts of a node's neighborhood.

Chapter 3

Counterfactual Explanations

A research area that has emerged in recent years is that of **Explainable Artificial Intelligence (XAI)** [36, 4]. Its creation was driven by the increasing integration of complex machine learning models into real-world applications. Traditional "black-box" models, while often achieving impressive predictive performance, lack transparency in their decision-making processes, which poses significant concerns in domains such as healthcare, finance, and criminal justice where stakes are high. Consequently, in order for these models to be frequently used in such tasks, everyone should be able to answer the questions 'How did this model arrive at the decision', 'What were the driving factors that led to this decision' and 'If any initial aspect of the problem was different would we have arrived at the same outcome?'[49]. XAI aims to bridge this gap and answer these questions by providing methods and tools that elucidate the underlying rationale of machine learning systems. This pursuit not only fosters trust and accountability, but also facilitates debugging and model improvement by understanding which features or data patterns significantly influence outcomes.

One method that has gained considerable attention in the field of explainable artificial intelligence (XAI) is Counterfactual explanations. A counterfactual explanation for a given instance is a set of minimal modifications to its features that would alter the model's prediction in a prescribed way. By identifying how slightly altering certain inputs leads to a different outcome, counterfactual explanations offer insight into model decision boundaries and suggest potential real-world actions end-users might take to achieve a desired prediction. In this thesis, we focus on providing Counterfactual Explanations for Graph Neural Networks. Specifically, in this chapter we will place Counterfactual Explanations inside the vast XAI framework, we will provide all important taxonomies and subcategories, and lastly we will enhance the readers' knowledge with the mathematical background needed to conceptualize Counterfactual Explanations.

Contents

3.1 Motivation

Deep Learning models have achieved remarkable success in various domains, such as molecular biology, social networks, and financial systems. However, their success is hindered by their 'black-box' nature [15]. Due to their reliance on nonlinear activations for learning feature representations, and usage of multiple hidden layers, Deep Neural Networks remain opaque to users, resulting in limited adoption. Graph Neural Networks (GNNs) specifically have demonstrated remarkable capabilities in processing and deriving insights from complex graph-structured data, yet they are not immune to inheriting and even amplifying biases present in the data. Since GNNs learn representations through the aggregation of neighboring node features, the biases embedded within the graph can directly influence the predictions of the model. This combination of lack of transparency and the adoption of data biases makes it challenging to discern how specific substructures or node attributes contribute to the final outcome and whether these contributions are biased.

Counterfactual explanations can provide meaningful answers to these types of problem by providing an actionable and intuitive mechanism for model **interpretability**, bridging the gap between 'why' a decision was made and 'how' an individual can change that decision that no other method can do simultaneously. The main areas where Counterfactual Explanations can become most necessary are **actionability**, transparency, and regulatory compliance. Counterfactuals, by design, recommend feasible modifications to input features, thus acting as a guide towards a desired result and not only as an indicator of where mistakes were made, and can thus provide tangible justifications that satisfy regulatory needs and user demands for interpretability [35, 51].

Besides graph data that this work focuses on, there are multiple other aspects of human endeavors that counterfactual explanations find good use. The most prominent is in the healthcare sector, where counterfactual reasoning provides better-informed clinical decisions by identifying the minimal modifications in patient characteristics that could alter a diagnosis or treatment recommendation, thus providing doctors with either an alternate path to explore or another reinforcing opinion. In finance, counterfactuals are instrumental in explaining credit scoring and risk assessment systems, clarifying what minimal changes to an applicant's profile could improve their loan eligibility, and fostering a more transparent lending environment [18]. Legal and criminal justice domains can also benefit from these insights, as they allow for scrutiny of predictive algorithms, thereby aiding in the pursuit of fairness and accountability. In all of these sections, data types may vary from tabular to image and textual to graph, leading to multiple counterfactual explanation methods and algorithms.

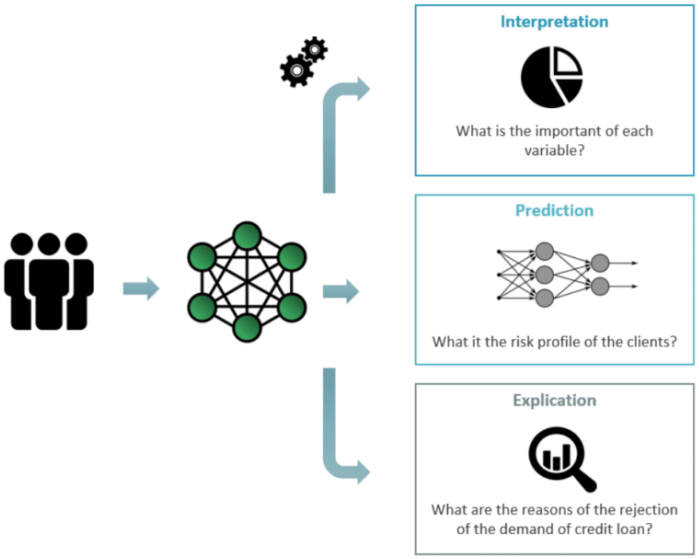


Figure 3.1.1: Importance of Explainable and Trustworthy AI predictions

3.2 Factual Reasoning

In this section, we will analyze a critical domain of Explainable AI called factual reasoning [49, 17]. This method is important in our work because while it works opposite to counterfactual reasoning, together they provide a comprehensive framework for ML Explainability. It was also the improvements in the field of factual reasoning that paved the way for researchers to develop the field of counterfactual theory.

Factual reasoning refers to the process of identifying the direct determinants that lead a model to a particular decision. In other words, it outlines the features of the input instance that most strongly supported the outcome in its current observed state. By focusing on the factors that drove the model's result, factual reasoning provides insights into how the decision was formed and which real-world aspects of the input were key contributors [6].

In graph-based domains, factual theory highlights the specific substructures that influenced the model's output. This may include nodes, edges, and in its most general form entire subgraphs. These subgraph explanations generally define an importance score for edges or node subsets that quantifies how each feature (or group of features) in the original instance x contributed to the model's prediction $f(x)$.

Now that we have established the basic concepts of factual reasoning, it is time to compare it with our main focus, counterfactual explanations. Whereas factual reasoning focuses on how the existing inputs led to a given outcome, counterfactual reasoning addresses how the outcome would change under hypothetical alterations of those inputs. Thus, factual explanations have a more descriptive character, showcasing the immediate evidence behind a model's decision. In contrast, counterfactual explanations adopt a prescriptive stance, suggesting minimal modifications that are both actionable and transparent and would ultimately change the prediction. Factual explanations answer the question "Given A already happened, will B happen?" providing a sufficient subgraph to recreate the prediction, while counterfactual explanations answer "If A did not happen, will B still happen?" thus generating a necessary example that without it we would arrive at a different outcome.

In this work, we are more interested in counterfactual explanations, as we deem providing reasonable and actionable alternatives more important than simply highlighting the driving factors of a decision. Counterfactual explanations provide the user with several alternatives that can help familiarize the public with AI.

3.3 Counterfactual Reasoning

3.3.1 Taxonomies

Its is now time to dissect Counterfactual Explanations into multiple taxonomies and analyze the characteristics that each method possesses. The first level of distinction between different Counterfactual Explanation methods can be made regarding the target to be explained [15, 41, 2].

Model-Level Counterfactual Explanations

These types of explanation aim to provide a counterfactual example that represents the entire model regardless of the inputs of the target model. Such works produce an explanation of the overall logic of the black-box model assuming that the provided explanation is complete and valid for any instance. A model-level approach might produce sets of counterfactual rules or prototypical examples that illustrate how various inputs could be minimally changed to achieve various target outcomes, thus helping users understand the global structure of the model's reasoning [20].

Instance-Level Counterfactual Explanations

Instance-level explainers aim to find counterfactual examples that would change the models prediction for a single instance. They are particularly useful in practice because they provide personal guidance on how to achieve a desired outcome. In a graph classification scenario this would imply changing the models' prediction of a graph instance into a different label, while in node classification tasks the explainer would try to alter the prediction of a single node [9].

Another distinction between different methods for graph data can be made by looking at the high-level approach that each explainer uses and the general architecture they employ.

- **Search-based methods** typically begin with an initial candidate counterfactual example x' and iteratively refine it until its prediction has changed. They rely on a specific criterion to search for a counterfactual within the dataset for a given input instance. This process is carried out by gradient-based optimization if the model is differentiable or by methods such as evolutionary algorithms and sampling-based procedures if it is not. Their advantage is that the user has control over the search method and properties [11, 23].
- **Heuristic-based methods** rely on domain-specific policies that modify the input graph until they reach a valid counterfactual. Instead of performing exhaustive exploration of the input space, these approaches use heuristics such as greedy methods to iteratively modify a given instance until a target outcome is reached [3, 1, 52].
- **Learning methods** typically train a second model that produces counterfactual explanations. These methods learn the variations of the output with regard to input changes and minimize a counterfactual loss term so that the learned generator outputs candidate modifications.

Learning-based methods can be further sub-categorized to the following three methods:

- **Perturbation** methods, where the model uses a soft masking technique in order to iteratively remove elements from either the adjacency matrix or the feature matrix until a counterfactual example is reached. These techniques typically treat the original model as a "black box", querying it multiple times to assess how close a candidate example x' is to achieving the desired outcome. This method is used in this work's explainer [64, 8].
- **Reinforcement learning methods** use agents that take actions in a predefined action space with the task of maximizing a reward function. Such functions are created by the user and are used to guide the agent to reach a counterfactual example [38].
- **Generative methods** generally use a generative network or a variational autoencoder (VAE) to synthesize counterfactual examples in a learned latent space [48, 26].

Finally, another important aspect of Counterfactual explainers is model access. An explainer that does not require access to the internal representations of the model, such as weights or gradients, is considered **model-agnostic**. These explainers can act on any model regardless of its training process, since they only rely on input-output pairs. **Model-specific** explainers on the other hand, rely on the usage of the model's training parameters and utilize them in order to understand why the model arrives at certain decisions. Since access to a black-box model is generally limited, model-agnostic explainers provide good generalization to various tasks. It is important to note that these taxonomic separations are not strict, and many attempts may include elements from multiple taxonomic groups.

3.3.2 Background

In this section, we will provide all the necessary background and formulation for Counterfactual Explanations. We will specifically address graph domains along with the characteristics that a counterfactual example possesses.

A Counterfactual Example is an instance x' that resembles the original instance x but provides a different prediction, when used as an input to a given model [14]. Lets start by denoting a model Φ mapping input data $\mathbf{x} \in \mathbb{R}^d$ to a prediction $\hat{y} = \Phi(\mathbf{x})$. A counterfactual example x' must satisfy the following equation:

$$\Phi(x) \neq \Phi(x')$$

This translates to counterfactual explanations always being predicted in a different class than the original input instance. For example, in the case of graph classification where $x = G$ and its counterfactual is also a graph $x' = G'$. Counterfactual examples must also remain minimal in regard to the original instance. Given that x' is a modified version of x and D is a distance function relevant to the task at hand, the distance of x' to x must remain minimal:

$$\mathcal{D}(x, x') \leq t$$

where t is an appropriate threshold ensuring minimal changes to the original instance. In graph domains the process of generating counterfactual examples can be formulated by minimizing a loss function that balances the distance between x and x', and the loss between $\Phi(x)$ and $\Phi(x)'$:

$$\arg\min_{x'} \mathcal{L}\left(\Phi(\mathbf{x}'), \Phi(x)\right) + \lambda \cdot D(\mathbf{x}', \mathbf{x})$$

where $\mathcal{L}$ is a differentiable loss function (in classification tasks, most often is the cross entropy loss Section 2.1.2) and λ is a hyperparameter controlling the trade-off between correctness of the prediction and proximity to the original instance x [15, 41]. In this minimization function, more constrain terms can be included to ensure actionability or guidance towards a certain target. This practice will be used in this work to help us create actionable and meaningful counterfactual explanations.

In general, the process of evaluating a counterfactual begins with the user specifying a desired outcome for a specific task. Hence, the first and most critical aspect of counterfactual theory is to produce a usable example that indeed generates a different model prediction. This proposal should also be created as minimally and efficiently as possible in order to be meaningful. One could argue that any model will produce a different prediction if the original input has been altered multiple times. However, this is not useful for the user that requires a minimal set of actions to incorporate towards the desired outcome. All of the above constitute a challenging task for AI engineers that want to enhance the public's trust towards black-box models and provide actionable alternatives to everyday tasks.

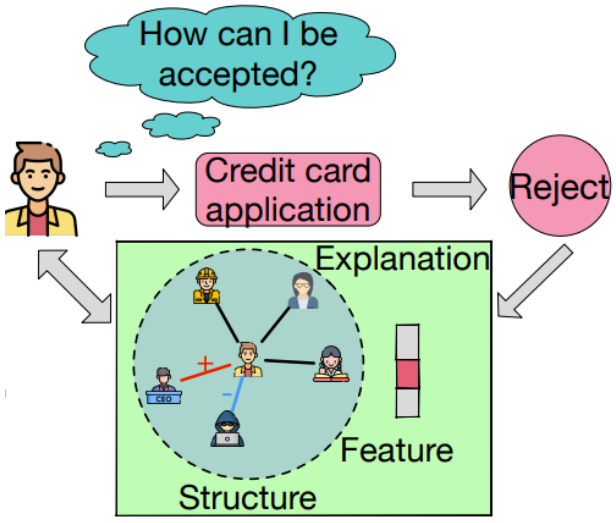


Figure 3.3.1: Example of user assistance through counterfactual explanations [15]

3.4 Related Work

After examining the basic concepts of Counterfactual Explanations, we are going to present relevant work done on graph domains in order to later place our method inside the spectrum of Counterfactual Explanations for Graph Neural Networks.

The first attempt to explain the predictions of GNNs was made by GNNExplainer [69]. This can be treated as the foundation for graph explainability methods. It includes identifying important subgraph substructure and node attributes that can preserve the prediction that a GNN made on the raw input graph. This work rests on factual theory, but its methodology can be easily implemented in counterfactual theory as well. The key contribution of GNNExplainer is that it treats the explanation process as an optimization task that maximizes the mutual information between the GNN's prediction Y , for a given instance, and an explanation (G_s, X_s) consisting of a subgraph G_s and the associated features X_s. This is done by introducing continuous mask variables that relax the selection of edges and features and a loss function function for these masks.

In practice, GNNExplainer maximizes the predicted probability of the true label $\hat{y}$ under the masked graph, while adding regularization terms to enforce sparsity.

Moving on to methods that incorporate Counterfactual reasoning we have CF-GNNExplainer [24]. This method targets at instance-level post-hoc explanation for node classification tasks. It produces counterfactual explanations by finding a binary perturbation matrix that sparsifies the adjacency matrix of the graph. The goal is to remove edges by zeroing out existing edges in the adjacency matrix. This is done by creating a mask M that perturbs the adjacency matrix A as $\hat{\mathbf{A}} = \mathbf{M} \odot \mathbf{A}$. Then in uses the general loss function we described in Section 3.3.2 in order to optimize the mask. The second term is the element-wise distance between the adjacency matrix and its perturbed counterfactual adjacency matrix, serving as a constraint for minimal perturbations.

Improving on the work made by the two aforementioned methods, CF^2 [49], produces factual explanations by balancing factual and counterfactual reasoning. By combining these two approaches, this work aims to obtain necessary and sufficient explanations. In this domain sufficiency accounts for factual reasoning, where the information induced by an explanation should be enough to produce the same prediction as the original graph. Necessity means that removing the minimal part will result in different prediction results that form counterfactual reasoning. The task is again a multi-objective optimization problem using a mask M. The loss function balances both factual and counterfactual reasoning in the sense that the counterfactual graph is the input graph without the factual subgraph. The loss function can be formulated as:

$$\mathcal{L}_{\text{explain}} = \mathcal{L}_{\text{input}} + \lambda \left(\alpha \mathcal{L}_f + (1 - \alpha)\mathcal{L}_c\right),$$

where the terms $\mathcal{L}_f$ and $\mathcal{L}_c$ serve to ensure that the sub-graph explanation is both necessary and sufficient for the model's prediction and α controls the weight that each of the two has in the final explanation. These last two methods provided the inspiration for our work where we use the masking approach along with a multi-factor loss function in order to create counterfactual explanations.

Moving on from masking techniques, there has been a lot of research in the reinforcement learning domain, especially for molecular and drug prediction tasks. MEG [40] is a reinforcement learning approach that produces counterfactuals for a given input molecule. In the reinforcement learning framework of MEG, it defines the action space as:

$$A_t = A_a \cup A_b^+ \cup A_b^- \cup \{\perp\},$$

where A_a stands for adding nodes (atoms), A_b^{-+} or A_b^- stands for adding or removing edges (bonds) and the null action. The reward function of the procedure incorporates a task-dependent regularization term that affects the policy of choosing the next action to perturb the input. Its policy is designed to choose only those actions that lead toward the generation of 'new' valid molecules using appropriate domain knowledge.

A method that lies in the domain of generative methods is CLEAR [26]. Unlike other methods that use masks to perturb the graph structure, CLEAR uses a variational auto-encoder (VAE) to generate the counterfactual graph. The encoder maps each input graph G to a latent representation Z, and the decoder generates the counterfactual based on Z. The counterfactuals are complete graphs with stochastic weights on the edges where the node features and graph structure are similar to the original graph G. The advantage of using VAE is that, by properly injecting auxiliary information, VAE can preserve the latent causal structure of the graph.

Lastly, in order to fully understand the spectrum of Counterfactual explanations, we will mention a model-level explanation method where the aforementioned works focus on instance-level explanations. GCFExplainer [20], aims to provide model-level counterfactual explanations for graph classification tasks. Given a GNN prediction $\Phi(x)$the global counterfactual explanation aims to find a global rule r_c ,for each class c, such that for each graph G_i we get $\Phi(r(G_i)) = c$. This work finds a set of representative counterfactual graphs for each class, where counterfactual graph means a modified from the original graph but with a different label with the original graph. This is being done by organizing the search space as a meta-graph where the vertices are all the graphs that can be obtained by performing no more than k edits on any input G, and edges connect graphs that are at one edit distance. After building this search space, GCFExplainer relies on vertex-reinforced random walks to obtain a set of counterfactuals, prioritizing those that most instances can reach. A greedy algorithm is then implemented to select a set of graphs $\mathcal{G}'$ that maximizes the coverage of

the original set's G_i 's coverage. The loss function incorporates regularization at both the input level and the output level, where regularization encourages the prediction of the model in the desired class at the output level as well as ensuring that the manipulation is sparse at the input level.

We have now provided some of the necessary relevant work on Counterfactual Explanations on graph data and discussed their strengths. Our method is placed among the masking methods for producing counterfactuals. We utilize a continuous masking approach with additional constraints in the loss function. By adding these constraints, we aim at creating dataset-specific rules that help guide the explainer to more meaningful and minimal counterfactual explanations.

Chapter 4

Methodology

In this section, we propose a model agnostic Counterfactual explainer class that handles both graph and node classification tasks. This explainer iteratively removes edges from the original graph structure until the prediction of the targeted instance is changed, making it an instance-level post hoc technique. Our method is based on the concept of soft masking to remove edges from the original graph, along with the incorporation of domain knowledge to guide the explainer to more insightful and minimal explanations. This explainer can be applied to any black-box GNN model while also being adaptable to any user or task specific constrains that constitute domain knowledge.

The contributions of this work can be summarized as follows:

- We generate counterfactual examples for graph and node classification tasks, using a model-agnostic explainer. This explainer uses the method of soft masking along with a binary thresholding scheme to iteratively remove edges and perturb the structure of the graph in order to provide counterfactual examples.
- We expand on the current bibliography by incorporating domain knowledge constrains within the loss function established in Section 3.3.2 in order to guide the explainer to produce more meaningful and minimal counterfactuals.
- We perform an iterative threshold search that helps the explainer locate the minimum number of edge deletions possible to achieve the transition from one label to another.
- We analyze the concept of trade-off relationships between minimality and fidelity as well as discuss how different hyper-parameters in the loss function can affect our evaluation metrics, prediction flip and graph edit distance.
- Lastly, we evaluate our explainer on two graph classification and one node classification datasets as well as visualize trade-off parameters and metrics.

Contents

4.1 Counterfactual Explainer

In this section, we will analyze all the important background, mathematical formulation and implementations that constitute our explainer class.

4.1.1 Architecture

First and foremost, it is important to define the high-level architecture capable of creating Counterfactual explanations. We begin by denoting our graph dataset, a black-box GNN classifier Φ and our explainer class. The process begins by training the black-box GNN model on the input dataset on the appropriate classification task (graph or node classification). The model provides predictions $\Phi(x)$ on any input instance. The internal parameters (weights and biases) of the GNN model are then frozen, in order to remain intact for every iteration of the explainer and maintain consistency between experiments. Then the explainer class receives the following inputs: the original input instance x, the GNN's prediction for this instance $\Phi(x)$ and any important domain knowledge appropriate for the task. The explainer then creates a candidate counterfactual example by iteratively querying the GNN during a process that we will analyze in depth in the following section. Lastly, this example is being given to the GNN model as a new input instance such that potentially, the model provides a different predicted label than before:

$$\Phi(x) \neq \Phi(x')$$

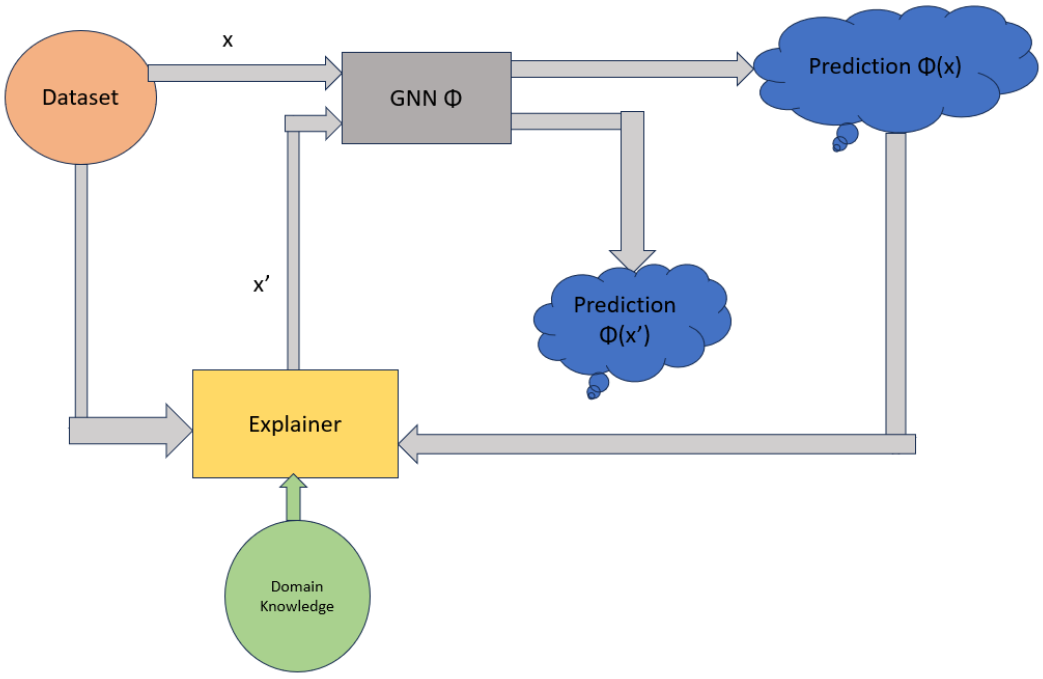


Figure 4.1.1: The architecture of the counterfactual example generation process. The pre trained GNN model provides predictions for any input instance. The explainer then utilizes this prediction along with domain knowledge to alter the model's prediction of that instance.

4.1.2 Continuous Masking

The overarching goal of this explainer is to minimally modify a given graph input instance, with the aim of altering a GNN model's prediction. This modification takes place in the graph structure and, more specifically, in the edges. In order to achieve edge perturbations, our explainer implements a soft masking procedure, where the term soft derives from the fact that this mask contains continuous values. The masking mechanism is designed to selectively modify graph connectivity, enabling the identification of critical edges whose removal alters the model's prediction.

This approach is being used because straight deletion of edges is a discrete and thus not differentiable procedure. We want to treat this problem as an optimization problem where we allow the explainer class to 'learn' which edges to delete and not as an exhaustive search problem, where the computational load is unbearable for large graphs. Hence, we employ this masking technique that allows for optimization. The key idea can be summarized as follows. By creating a mask array object M with the same size as the adjacency matrix and updating it appropriately, we can then use the dot product of the two, $M \cdot A$ in order to zero out some entries in the adjacency matrix and thus remove edges, resulting in the creation of counterfactual examples. In this subsection, we will provide the necessary background information needed for this technique.

Lets start by denoting as σ the sigmoid activation function:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

and two mask objects Z and M that have the same shape as the adjacency matrix (or in our case an edge index PyTorch Geometric object). The adjacency matrix contains the value 1 in places where an edge exists and the value 0 in places where an edge does not exist. The PyTorch Geometric object called edge index contains only the value 1 in places where an edge exists, avoiding to enlist all the zeros for memory efficiency in a process that resembles that of the adjacency matrix. So, our mask Z matches the shape of the edge index tensor:

$$Z = [z_1, z_2, \ldots, z_E]$$

where E is the number of edges. We start by initializing the mask as $z_i = 0$. Then we pass the sigmoid function through every mask value z_i, thus providing us with the second mask M where $m_i = 0.5$.

Since the explainer deals with undirected, unweighted graphs, the GNN models that we train in PyTorch Geometric use a weight of 1 for each existing edge and can receive an argument of an edge weight tensor W which in the training process consists of all the values being 1. It is in the edge weight matrix of the GNN model we perform our multiplication with our mask M. By initializing each mask value at 0.5 and then multiplying it with the edge weight matrix, we essentially assign each existing edge a weight of 0.5 at the first iteration of the explainer. After the entire process is complete, edges that have weights close to 1 will be kept and edges that have weights close to 0 will be deleted. This is the reason why we chose to initialize our mask at zero, so that we assign each edge with an equal chance of being deleted or retained.

In the first forward pass that our explainer makes through the GNN model, it inserts the dot product $W' = M \cdot W$ as an argument to the model. The model then makes a new prediction (using its frozen parameters that have not been altered) on the new instance. Then the explainer uses this new prediction along with the desired counterfactual class to calculate a multi-factor loss function:

$$\mathcal{L}(Z) = \mathcal{L}_{\text{pred}}(\Phi(G; M(Z)), \hat{y}) + \lambda \cdot \mathcal{L}_{\text{dist}}(M(Z))$$

where $\Phi(G; M(Z))$ represents the GNN's output when using mask M, $\hat{y}$ is the target label, λ is a hyperparameter controlling how much the second term contributes in the total loss.

The prediction loss $\mathcal{L}_{\text{pred}}$ is the cross-entropy loss as described in Section 2.1.2 that encourages the model to predict the target class $\hat{y}$ when the mask is applied:

$$\mathcal{L}_{\text{pred}}(\Phi(G; M(Z)), \hat{y}) = -\log P_{\Phi}(\hat{y} \mid G; M(Z))$$

The second term $\mathcal{L}_{\text{dist}}$ is a penalty term that encourages the mask values to be close to 1 and thus keeping them.

$$\mathcal{L}_{\text{distance}}(M) = \sum_{e \in E} (1 - m_e)$$

This formulation encourages optimization to find a minimal perturbation, only removing edges that are critical for changing the prediction. The higher this loss term, the more edges are being removed, so minimizing this term encourages edge retention. In this problem, we select this type of distance function, but, as we described in Section 3.4, any appropriate distance function can be implemented. These two terms constitute our base explainer model. In the following section, we will incorporate more constraints and penalty terms in the loss function in order to guide the explainer to more insightful deletions using domain knowledge.

After the calculation of the loss function that forces the prediction toward a target class while keeping deletions minimal, the backpropagation algorithm calculates the gradients and updates the mask Z :

$$Z^{(t+1)} = Z^{(t)} - \alpha \nabla_Z \mathcal{L}^{(t)}$$

where a is the optimer's learning rate. After the first iteration, the mask Z is again passed through the sigmoid transformation, creating mask M that is scaled in $[0, 1]$. This process continues for many iterations, each time using the newly updated mask to call the GNN again with the new edge weights. After this process is completed, our mask M is full with continuous values $m_i \in [0, 1]$ each representing the final weight possessed by each edge. At this stage, no edge has been deleted, only their weight in the message passing algorithm has been modified, and it is time for the explainer to decide which edges should be deleted. This is done by enforcing a binary threshold τ where every edge with a value above the threshold is kept and every edge with a weight below the threshold is deleted:

$$M_{\text{binary}} = \{m_e^{\text{binary}} \mid m_e^{\text{binary}} = \mathbb{1}[m_e > \tau],\ e \in E\}$$

where $\mathbb{1}[\cdot]$ is the indicator function. The resulting binary mask identifies edges that should be retained in the counterfactual explanation. The counterfactual graph G' is then constructed by keeping only these edges of the original graph:

$$G' = \left(V, \{e \in E \mid m_e^{\text{binary}} = 1\}\right)$$

Intuitively, the first threshold that comes to mind is 0.5 but this does not provide the most minimal solution, as there can be many edges with a value below 0.5 that should not be deleted and the prediction would change whatsoever. So, at this stage, the explainer performs an iterative threshold search for threshold values that removed the least amount of edges possible while managing to alter the predicted class. The explainer creates a new graph G' for every $\tau \in \{0.05,\ 0.1,\ 0.15,\ 0.2,\ 0.25,\ 0.3,\ 0.35,\ 0.4,\ 0.45,\ 0.5\}$ stopping at the first threshold value that flips the GNNs prediction. This threshold value is guaranteed to delete the fewest edges possible among the other thresholds. The user can decide whether to include more threshold values above 0.5 but this would emerge the issue of trade-off between flipping the prediction of a particular instance and the size of the modifications that we will discuss in the following sections.

This masking approach offers several advantages, such as maintaining a fixed computational graph structure throughout optimization, which is more efficient than rebuilding the graph at each step. Most importantly, it allows for gradients to flow properly through the network, as the masking operation is differentiable, thus allowing for optimization at each step.

4.1.3 Loss Function Constraints

It is now time to introduce more constraints in the loss function with the aim of guiding the explainer towards more minimal and meaningful explanations. These constraints derive from domain knowledge on the task at hand. The introduction of an extra constraint or penalty term is not guaranteed to push the explainer towards more minimal or insightful counterfactual. It requires a deep understanding of the task and the available data, as well as exhaustive hyperparameter tuning to find the optimal hyperparameter windows. With the addition of an extra term and a hyperparameter controlling its impact, the model complexity increases, leading to an increase of the trade-off scenarios. In this section, we will provide the theoretical background for the structural constraints used in this work. It is important to note that these constraints can have any form or meaning the user wants. If the task demands a more semantic constraint that derives from the data, or the user seeks counterfactual examples toward any direction, they can be incorporated into the loss function. For the scope of this work, we focus on structural constraints, deriving from the preprocessing of graph data, that help guide the explainer to more meaningful explanations.

Node Degree Centrality Constraint

The intuition behind this constraint is the existence of nodes with very high degree centrality in comparison to other nodes. These nodes are often called hub nodes because they represent a bridge, where crucial information has to cross from these nodes to reach the entire graph. This information is also crucial for a GNN model making a prediction, and hence we want to utilize this information to make minimal deletions and alter the model's predicted class.

We begin the process by computing the degree for each node v in our graph G.

$$\text{degree}(v_i) = \sum_j \mathbb{1}[(v_i, v_j) \in E]$$

And then identify the node with the maximum degree:

$$v_{\text{max\_deg}} = \arg\max_{v_i \in V} \text{degree}(v_i)$$

In different tasks and datasets, there can be many hub nodes or just one hub node per graph. For simplicity, we assume the existence of one hub node, with the biggest node degree, but this is easily applicable to more than one hub nodes. Then we create the list of edges connected to the hub node:

$$E_{\text{hub}} = \{(v_i, v_j) \in E : v_i = v_{\text{max\_deg}} \vee v_j = v_{\text{max\_deg}}\}$$

Finally, for edges $e \in E_{\text{hub}}$ we construct a penalty term that penalizes the mask values m_e

$$\mathcal{L}_{\text{degree}} = \sum_{e \in E_{\text{hub}}} m_e$$

The total loss function of the explainer can now be formalized as:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{pred}} + \lambda_{\text{dist}} \cdot \mathcal{L}_{\text{dist}} + \lambda_{\text{degree}} \cdot \mathcal{L}_{\text{degree}}$$

When mask values m_e of these edges are close to 1 (thus close to be kept), the loss of the total model increases. The optimizer, in its attempt to lower the total loss, will try to remove these edges, and thus alter the predicted class. We also add a hyperparameter λ_{degree} to control the influence of this penalty term. So, to sum up what each term in the loss function contributes:

- $\mathcal{L}_{\text{pred}}$ encourages changing the classification label to a desired label.
- $\mathcal{L}_{\text{dist}}$ encourages keeping as many edges as possible by penalizing removal.
- $\mathcal{L}_{\text{degree}}$ encourages removing hub-connected edges.

By incorporating this extra penalty term into the loss function, we punish keeping edges that are connected to the node with the highest degree. This is crucial as these nodes play an important role in the message passing procedure and can highly influence the prediction of a GNN model.

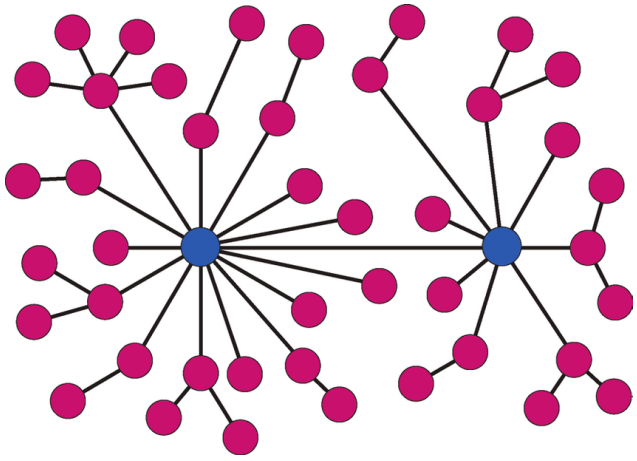


Figure 4.1.2: Example of nodes with high degree centrality. The two nodes in blue represent hub nodes.

Edge Betweenness Centrality Constraint

This constraint was designed to leverage the fact that, in real-world scenarios, some edges are more pivotal in the information flow than others within a graph. These edges are often called bridges, similar to the hub nodes, and connect different parts of the graph that would otherwise be separate. The property that reveals whether an edge acts as mentioned is the edge betweenness. Edge betweenness centrality represents the number of shortest paths that run through an edge to the total number of shortest paths. To begin, we calculate the edge betweenness centrality for each edge e in the graph as:

$$C_B(e) = \sum_{v,u \in V} \frac{\sigma(v, u \mid e)}{\sigma(v, u)}$$

where (v, u) is a pair of nodes, $\sigma(v, u \mid e)$ is the number of shortest paths between v and u that pass through edge e and $\sigma(v, u)$ is the total number of shortest paths between nodes v and u. Then we gather the edges that are above the 90th quartile (depending on the task) in terms of edge betweenness score. This is crucial as we want to only perturb the edges that have the highest edge betweenness score and not all the edges in the graph. This group of edges can be denoted as:

$$E_{\text{bet}} = \{e \in E \mid C_B(e) > Q_{0.9}(C_B)\}$$

Finally, for edges $e \in E_{\text{bet}}$ we construct a penalty term that penalizes the mask values m_e as before:

$$\mathcal{L}_{\text{bet}} = \sum_{e \in E_{\text{bet}}} m_e$$

The total loss function of the explainer can now be formalized as:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{pred}} + \lambda_{\text{dist}} \cdot \mathcal{L}_{\text{dist}} + \lambda_{\text{bet}} \cdot \mathcal{L}_{\text{bet}}$$

Similarly to the node degree penalty, when the mask values m_e are close to 1, the total loss increases. The optimizer will attempt to bring the mask values of these edges closer to 0 and thus remove these edges. The hyperparameter λ_{bet} controls the weight of the term in the total loss function. The contribution of each term in the loss function is as follows:

- $\mathcal{L}_{\text{pred}}$ encourages changing the classification label to a desired label.
- $\mathcal{L}_{\text{dist}}$ encourages keeping as many edges as possible by penalizing removal.
- $\mathcal{L}_{\text{bet}}$ encourages removing edges with high betweenness centrality.

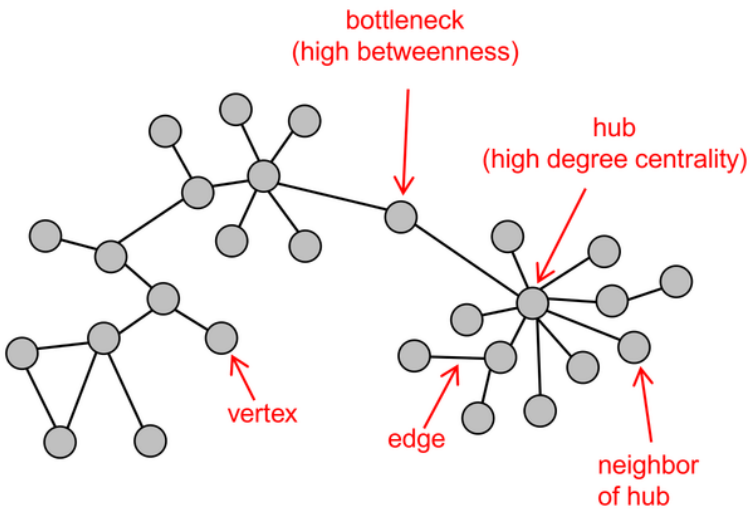


Figure 4.1.3: Example describing nodes with high degree centrality and high betweenness centrality. Edges connected to these nodes are crucial for information flow across the graph. [34]

To conclude, it is important to note that the two described constraints derive only from the graph structure. Our method does not utilize node or edge features and focuses only on explaining models that arrive at their predictions by analyzing the graph structure. It does so by perturbing edges and creating graphs similar to the original for the model to predict. However, the framework does not prohibit node or edge additions, as well as utilization of features. Such methods exceed the scope of this thesis and require further exploration.

Chapter 5

Experiments and Results

In order to evaluate the proposed base method along with the models with the additional constraints, we carried a variety of experiments across two GNN tasks. In this chapter, we will provide all the necessary background and details of our experimental setup, the datasets we used, the GNN models we explained, and the evaluation metrics that were used.

In addition, we will present quantitative results across these datasets for each evaluation metric. We will discuss the potential of a trade-off parameter between our two metrics that should be taken into account when providing an explanation. Lastly, we will provide visual representations of both the data used and the counterfactual examples that were created, to fully conceptualize the impact of this work.

Contents

5.1 Experimental Setup

5.1.1 Technologies used

In this segment, we discuss the technical aspects needed to set up our counterfactual explainer, from the framework and libraries to the systems used.

The programming language used for all the code implemented within our explainer classes, data processing workflows, and experimental evaluation was **Python**. It is a robust and widely adopted language that is particularly indispensable for machine learning and graph tasks.

First, Python offers an extensive suite of machine learning libraries, notably PyTorch and scikit-learn, which were used in this work, providing powerful tools and frameworks for developing and deploying machine learning models. These libraries encompass a variety of techniques, from traditional algorithms to deep learning architectures, thus facilitating the construction of sophisticated machine learning pipelines for any graph task endeavor. In addition, Python demonstrates an exceptional capability in data manipulation and preprocessing tasks, which are pivotal for both machine learning and deep learning. The NumPy library, for instance, provides efficient data structures and functionalities for data cleaning, handling, and analysis, thus enabling efficient preparation of graph data for modeling.

The deep learning framework that was used is **PyTorch**. It plays a pivotal role in facilitating graph-based tasks, offering a user-friendly and flexible platform for model development. Notably, PyTorch supports a variety of graph neural network (GNN) architectures and includes pre-trained model weights, node/edge feature utilities, and evaluation tools, thereby streamlining their incorporation into graph data processing pipelines. Furthermore, its capacity for GPU acceleration enables faster training and evaluation, which is essential for complex graph-related tasks. It also provides a framework called **PyTorch Geometric** which is a widely adopted extension library, dedicated to the design and implementation of graph neural network (GNN) models. By offering specialized data structures and utility functions, it streamlines the handling of node, edge, and graph-level information, and thus simplifies the development of advanced graph-based architectures. Notably, PyTorch Geometric provides a comprehensive suite of GNN layers and sampling techniques, enabling efficient mini-batching, neighborhood aggregation, and message passing across large-scale graphs. As we discussed in Section 4.1.2 ,PyTorch Geometric provides the basic adjacency data structure and the appropriate edge weight handling that our explainer utilizes to create predictions.

For calculating graph specific metrics, visualizations and graph representation models we used **Networkx**. It is a widely recognized Python library dedicated to the creation, manipulation, and analysis of graph-based data. By providing specialized data structures and utility functions, it streamlines the management of nodes, edges, and associated attributes, simplifying the development of sophisticated network algorithms. All centrality measures and graph visualizations were computed with its use.

Lastly, all our code implementation was done in the **Google Colaboratory** environment. Commonly known as Colab, it is a cloud-based Jupyter notebook environment that also provides free access to GPU resources. Users in this way can easily have access to GPU-enabled environments, eliminating the need for local hardware usage and intense runtimes.

5.1.2 Datasets

For our experiments, we used two benchmark datasets for graph classification and one for node classification [68].

REDDIT-BINARY

This dataset is primarily used in graph classification tasks. Each graph represents a Reddit discussion thread, where the nodes correspond to users, two of which are connected by an edge if one responded to a comment of the other. The task is to classify the entire graph as either discussion-based thread or question/answer-based thread.

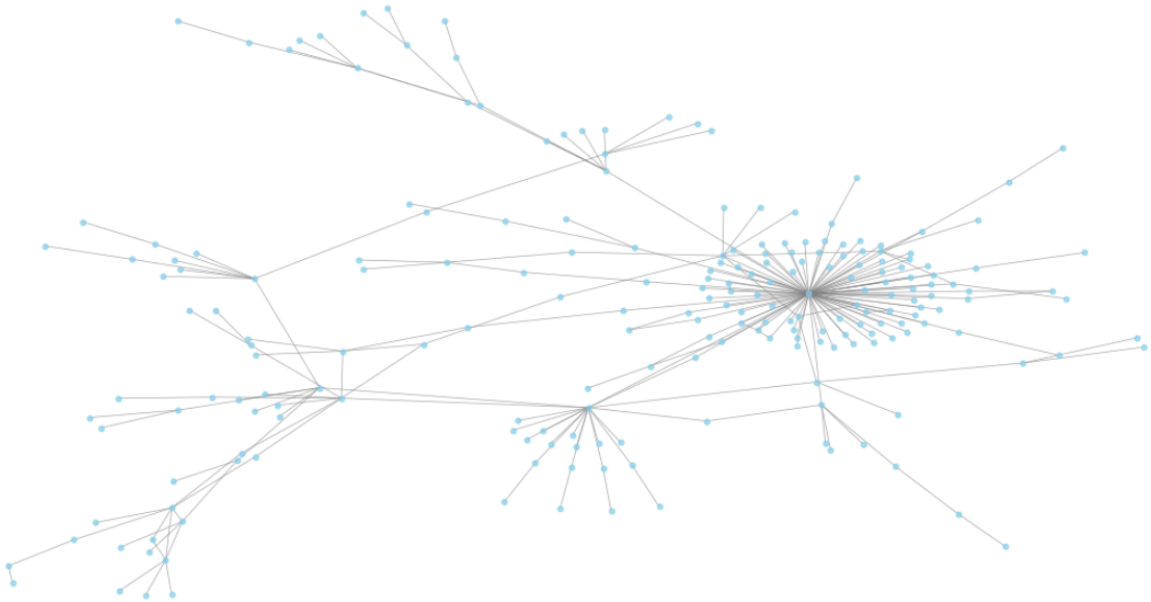


Figure 5.1.1: Example graph from the Reddit-Binary dataset labeled as discussion thread.

IMDB-BINARY

IMDB-BINARY is a movie collaboration dataset that consists of the ego-networks of 1,000 actors/actresses who played roles in movies in IMDB. In each graph, nodes represent actors/actress, and there is an edge between them if they appear in the same movie. If the ego network was derived from an Action movie, the label for that ego network is "Action." If it came from a Romance movie, the label is "Romance." Thus, the classification task is to determine if an unseen ego-network derived from an Action or Romance genre.

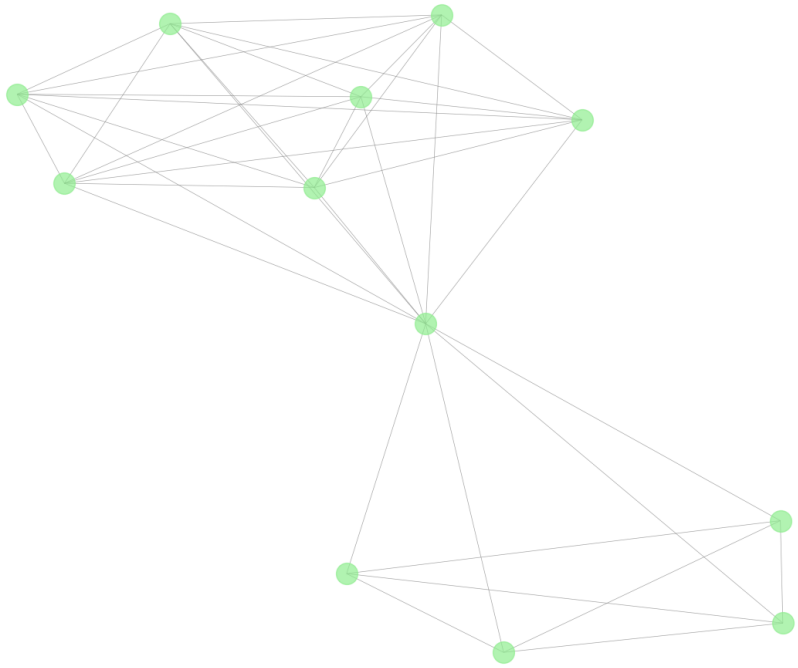


Figure 5.1.2: Example graph from the IMDB-Binary dataset labeled as 'Romance'.

	Reddit-Binary	**IMDB-Binary**
# classes	2	2
# graphs	2000	1,000
# node features	0	0
# edge features	0	0
Avg # nodes	429.6	19.8
Avg # edges	497.75	96.5

Table 5.1: Graph Classification Datasets characteristics. We can see that Reddit-Binary contains larger and more dense graphs, while IMDB-Binary has smaller graphs, as is expected for an actor ego-network. Both datasets rely on their structure as they contain no features.

BA-SHAPES

For the node classification task, we used the synthetic benchmark dataset, BA-shapes. This dataset is designed by adding a motif to random nodes from a base Barabasi-Albert (BA) graph [58]. We chose a random Barabasi graph with 1000 nodes and added 400 motifs to random base nodes. Each node is classified into 4 different labels. 'Non-House' refers to nodes that are outside the motif, and then we have 'House-Top', 'House-middle' and 'House-Bottom' referring to the position of the node inside the motif. This dataset is extensively used in explainability tasks because of the existence of ground-truth labels, regarding the position of each node in the motif.

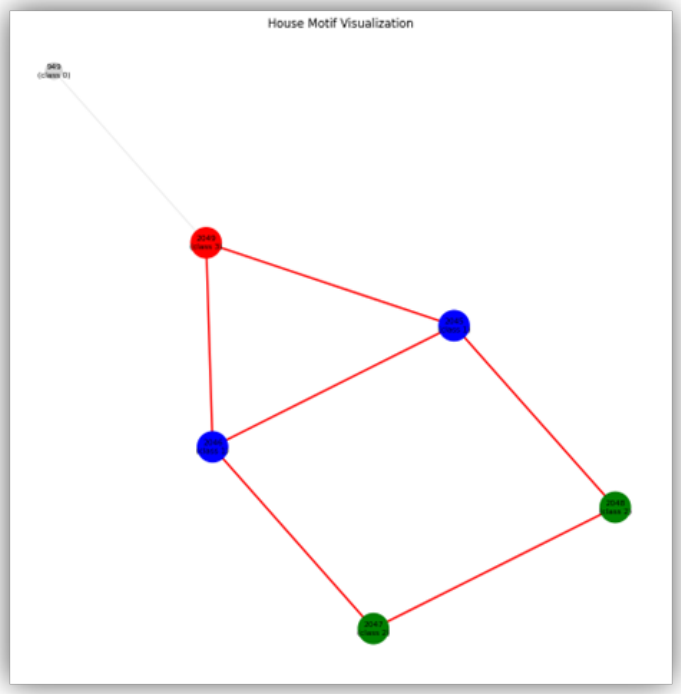


Figure 5.1.3: House motif connected with a node from a Barabasi Graph. Green nodes are labeled 'House-Bottom', blue nodes are labeled 'House-middle', the red one 'House-top' and the grey one is 'Non-House'.

	BA Shapes
# classes	4
# nodes in motif	5
# edges in motif	6
# nodes in total	3000
# edges in total	7691
Avg node degree	5.13

Table 5.2: Statistics for the BA Shapes dataset.

It is again important to highlight the absence of node and edge features. The datasets are chosen specifically to lack these attributes, so predictions and graph modifications are made solely on the graph structure.

5.1.3 Classifier Models

In this subsection, we will analyze the black-box GNN models that were used in our experiments. Our goal was to explain simple GNN models in order to designate the capabilities of the explainer rather than the complexity of the models themselves. Hence, we chose simple GNNs that delivered acceptable results on both tasks, rather than the state-of-the-art models for each dataset.

For the Reddit-Binary dataset, we implemented a Graph Convolutional Network (GCN) [19] architecture specifically designed for graph-level classification tasks. The model is structured as a three-layer neural network. The architecture begins with a GCNConv layer that transforms the single-dimensional node features into 64-dimensional latent representations by aggregating information from neighboring nodes according to

the graph topology. Following a ReLU nonlinearity, a second GCNConv layer maintains the 64-dimensional representation while enabling higher-order neighborhood interactions. Both convolutional layers support edge weighting through the edge-weight parameter, allowing the explainer later to make edge deletions. After having set all node features to 1 (since they are unavailable), we employ global mean-pooling, which aggregates node features across each graph by computing the mean of node representations within each graph in the batch. This pooled representation is then transformed by a linear layer into a 2-dimensional output space corresponding to the binary classification task.

For the IMDB-Binary, a more sophisticated model was required. The model initializes with an embedding layer to map the single input feature to a hidden-channels representation space with 128 dimensions. The core of the architecture consists of four sequential Graph Convolutional Network layers (GCNConv), each maintaining the dimensionality of the hidden representations while enabling hierarchical message passing across the graph structure. Each convolutional layer is followed by batch normalization to stabilize training, and a ReLU activation function to introduce non-linearity. A distinguishing feature of this architecture is the incorporation of multiple pooling strategies to generate comprehensive graph-level representations. Specifically, we employ global mean-pool, global max-pool, and global add-pool in parallel to capture complementary aspects of the node representations across each graph. The concatenation of these three pooling outputs results in a $3 * 128$ dimensional vector that encapsulates diverse statistical properties of the graph structure. This graph representation is then processed through a three-layer MLP with batch normalization and ReLU activations, progressively reducing dimensionality to the final two-dimensional output space corresponding to the binary classification task.

For the Ba-Shapes dataset and the node classification task, we used a Graph Convolutional Network (GCN) architecture to learn node representations from the graph structure. The model is constructed as a three-layer neural network. The architecture begins with an embedding layer that maps each node to a hidden channels-dimensional vector space since node features are unavailable. The core computational components consist of two sequential GCNConv layers, each with a hidden layer dimension of 64. The GCNConv operations aggregate information from neighboring nodes according to the normalized adjacency matrix, effectively implementing the spectral graph convolution approximation. Between convolutional layers, we apply ReLU activation functions followed by dropout regularization with a rate of 0.5 to prevent overfitting during training. The final linear transformation layer maps the node representations to a 4-dimensional output space, corresponding to the target labels. As described above, the model utilizes edge weights that are paramount for our explainer. This architecture enables the model to capture both local and higher-order structural patterns within the graph through message passing, while the dropout mechanisms enhance generalization capabilities.

Datasets	**Accuracy(%)↑**
Reddit-Binary	78
IMDB-Binary	75
BA-Shapes	90

Table 5.3: Predicted Accuracy for each dataset by the corresponding GCN models.

5.1.4 Evaluation Metrics

To assess the performance of our explainer, we incorporate two widely-used metrics for graph explainability, Prediction Flip Rate and Graph Edit Distance (GED).

The Prediction Flip Rate measures how many instance's predictions were successfully altered to the total number of instances that were attempted to alter. It is defined as:

$$\text{PFR} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}[f(x_i) \neq f(x_i')]$$

Where:

- N is the total number of instances (graphs or nodes).

- $f(x_i)$ is the predicted label of the original instance.
- $f(x_i')$ is the predicted label of the perturbed instance.
- $\mathbb{1}$ is the indicator function.

In this work, we will present PFR as a percentage (%) of the test set instances, whose prediction the explainer was able to alter to the total instances in the test set.

The Graph Edit Distance (GED) [59] quantifies the dissimilarity between two graphs by counting the minimum number of edit operations needed to transform one graph into another. In this work, since our explainer only deletes edges, we denote GED as the number of edges deleted to the total number of edges in the graph in instances where the explainer manages to alter their prediction (true GED): Then we present the average GED across the entire test set.

$$\text{GED}(G, G') = \frac{\text{Number of edges deleted to transform } G \text{ into } G'}{|E|}$$

Where:

- $G = (V, E)$ is the original graph.
- $G' = (V, E')$ is the modified graph.
- E is the total number of edges in the original graph.

For the node classification task specifically, we introduce Local GED and Global GED. Since our explainer deletes edges only in the k-hop neighborhood of the target node, we keep track of the deletions in the local and in the global scope of the graph.

$$\text{Local GED}(G, G') = \frac{\text{Number of edges deleted to transform } G \text{ into } G'}{\text{Edges in the node neighborhood}}$$

$$\text{Global GED}(G, G') = \frac{\text{Number of edges deleted to transform } G \text{ into } G'}{\text{Total edges in the graph}}$$

5.1.5 Trade-off

After having formulated our loss function and the evaluation metrics, it becomes apparent that a trade-off occurs between Prediction Flip Rate and Graph Edit Distance. When searching for the optimal λ_{dist} we observe that a higher value leads to fewer deletions, while a value close to 0 leads to more edge deletions. On the other hand, this also affects Prediction Flip Rare because an instance may require a specific amount of modifications for its prediction to change, and thus not deleting enough edges may result in the label of the said instance not changing. So we arrive at a complex optimization problem where Prediction Flip and Graph Edit Distance are heavily impacted by the choice of λ_{dist} and are inversely proportional. As we strive for more instances in the test set to be explained, we have to accept an increase in the number of edges that we remove. It is obvious that when $\lambda_{\text{dist}} = 0$ the explainer achieves its highest Prediction Flip Rare but with the trade-off of a highly altered counterfactual graph.

Consequently, it becomes crucial to formulate this trade-off and provide an effective range of impact that the explainer has. In cases where a counterfactual explanation is absolutely necessary and the user must be provided with one, it is important to relax our distance constraints and try to provide an explanation for every instance. On the other hand, if a counterfactual explanation is easily achieved and we strive for it to be as minimal as possible, we can increase the weight of the constraints and try to provide minimal explanations for only the instances that can be explained with few deletions. So we can formulate the trade-off as a weighted sum of the Prediction Flip Rate and the Graph Edit Distance for a run on the entire test set as:

$$Trade - off = k * PFR + t * GAE$$

where $k + t = 1$. The user can decide the value of the weight that each metric should have and adjust the hyperparameters of the explainer accordingly. A balanced approach that accounts for both metrics would

be to use $k = t = 0.5$ and assign an equal weight to both metrics. In the following section, we will provide analytical visualization of this trade-off that enlists all possible combinations and how the value of each hyperparameter affects our explanations.

5.2 Results

In this section, we will provide quantitative results regarding the performance of our explainer models for all three datasets for both classification tasks. We will also provide a complete visualization of the trade-off relationship between our two metrics, as well as a few qualitative examples of the explanations that our models provide.

For the following subsections, whenever we refer to the Base Model we refer to the explainer without any additional constraint other than the L1 regularization distance term. Then we will provide results for the Node Degree Model as well as the Edge Betweenness Model that incorporate the respective constraints in the loss function as we discussed in Section 4.1.3. The Base model acts as a benchmark and the other two models as an improvement upon it. However, the choice of the appropriate constraint is very important and these particular constraints do not guarantee an improvement for any task. The proposed framework works with any carefully chosen constraint that derives from domain knowledge, but it is not certain that any constraint would improve upon the Base model.

5.2.1 Quantitative Results

In this subsection, we will provide the quantitative results for all of our explainer models for all of the datasets and tasks we discussed. It is important to note that we performed a hyperparameter search for the explainer learning rate as well as the number of epochs with which the explainer should be trained. For the graph classification tasks, we used an Adam optimizer with a learning rate of 0.5 and for the node classification task we used a learning rate of 0.1. The best number of epochs were found to be 600. For the node classification task, we denote only the local GED as it is the most impactful one, and all the GED values represent the portion of edges deleted only in successfully explained instances. The best k-hop value for the node neighborhood was found to be $k = 3$.

The tables that will be listed below contain some representative results for the best runs of the explainer for the corresponding model and a given value of λ_{dist} and $\lambda_{\text{constraint}}$. The values that are considered best depend on the trade-off. In this subsection, we will provide some representative values, along with the hyperparameters that cause them, to show the improvement that an additional well-designed constraint in the loss function could have on each metric.

	REDDIT-BINARY		**IMDB-BINARY**		**BA-SHAPES**	
Method	**Flip_rate↑**	**GED↓**	**Flip_rate↑**	**GED↓**	**Flip_rate↑**	**L-GED↓**
Base Model	53.2	0.2	66	0.12	50.4	0.21
Node_Degree	64.5	0.25	72	0.2	–	–
Edge_Betweenness	50.7	0.2	–	–	51.7	0.2

Table 5.4: Comparison of methods on REDDIT-BINARY, IMDB-BINARY, and BA-SHAPES datasets. The shown results were calculated using $\lambda_{\text{dist}} = 0.01$ across all datasets. For the Reddit-Binary we used $\lambda_{\text{deg}} = 1$ and for the IMDB-Binary $\lambda_{\text{deg}} = 0.1$. For the Betweenness model we found $\lambda_{\text{bet}} = 0.01$ and $\lambda_{\text{bet}} = 0.1$ for the Reddit-Binary and Ba-Shapes respectively.

From these results we can observe that the Node Degree model performs better at Flip Rate than the Base Model while under-performing in GED for the same distance hyperparameter. This shows that it guides the explainer into deleting more edges, while providing the ability to explain more instances. This is an expected result, as both graph classification datasets contain graphs with high degree nodes. Hence, in order to alter the predicted class, the Node Degree model guides deletions towards edges connected to these nodes. However, this comes with a trade-off because in multiple occasions the explainer may have to delete more edges to reach the desired flipping of the label, and thus increase GED. This is a pattern we will observe for

the rest of this work, as the Node Degree model was designed to increase Prediction Flip Rate, by guiding deletions towards the edges connected to hub nodes. As will be shown in the next subsection, this model systematically outperforms the Base Model in Prediction Flip Rate while deleting more edges. The Node Degree model was not used in the Ba-shapes dataset, as the node degree is fixed inside the motif, and results did not show any improvement over the Base Model.

	REDDIT-BINARY		**IMDB-BINARY**		**BA-SHAPES**	
Method	**Flip_rate↑**	**GED↓**	**Flip_rate↑**	**GED↓**	**Flip_rate↑**	**L-GED↓**
Base Model	80	0.32	79.5	0.17	55.6	0.3
Node_Degree	87.5	0.38	84	0.28	–	–
Edge_Betweenness	77	0.29	–	–	54.7	0.28

Table 5.5: Results using $\lambda_{\text{dist}} = 0.001$. The rest of the hyperparameters remain the same as we examine the impact of λ_{dist} .

As we can see, the trend for the Node Degree Model continues. The Edge Betweenness Model, on the other hand, manages to perform at the same level as the Base model in terms of Prediction Flip Rate while removing fewer edges. In both tasks, it manages to guide the explainer to deleting bridge edges that carry crucial information for the message passing algorithm. By targeting these edges, we can observe that it outperforms the base model in terms of GED while maintaining the same levels of PFR. This trend will also continue throughout all the experiments as both the Reddit-Binary and the Ba-shapes contain large and dense graphs that have multiple bridge edges. We do not show results for the Edge Betweenness Model for the IMDB-Binary because it does not improve upon the Base model. This can be explained by the fact that there are not many shortest paths in the IMDB-Binary dataset, and so this constraint misguides the explainer.

	REDDIT-BINARY		**IMDB-BINARY**		**BA-SHAPES**	
Method	**Flip_rate↑**	**GED↓**	**Flip_rate↑**	**GED↓**	**Flip_rate↑**	**L-GED↓**
Base Model	89	0.45	84	0.22	58.9	0.37
Node_Degree	95.75	0.47	90	0.45	–	–
Edge_Betweenness	86	0.43	–	–	57	0.32

Table 5.6: Results using $\lambda_{\text{dist}} = 0.0001$. The rest of the hyperparameters remain the same as we examine the impact of λ_{dist} .

To conclude this section, we can observe that both aforementioned patterns continue to hold. The Node Degree model consistently outperforms the Base model in terms of Flip Rate, providing explanations for more instances, but with a trade-off tendency to delete more edges to achieve so. The Edge Betweenness model achieves the opposite as it maintains the same level of Flip Rate to the Base model, but does so while deleting fewer edges and providing more minimal explanations. As λ_{dist} decreases, the explainer prioritizes flipping the predicted class heavily over the minimality of deletions. The best Flip Rate can be achieved by setting $\lambda_{\text{dist}} = 0$, thus completely disregarding the distance term in the loss function, but acquiring examples that differ significantly from the original instance.

The proposed framework allows for the creation of custom constraints that, when implemented carefully in the loss function, guide the explainer towards a desired set of counterfactual explanations. In this work, we decided to implement structural constraints that derive from the graph properties of each dataset and task. In order for these constraints to have an impact, a preprocessing study has to be done on the graph data, so that the property chosen reflects on the graphs characteristics. Dense graphs may require a different approach from small graphs, and star-like graphs may need a different constraint than community formations. However, these constraints do not necessarily have to be structural. They can represent any semantic pattern possible as long as the designer of the constraint carefully incorporates it into the loss function. Consequently, the choice of the constraint is very important as it can either provide more meaningful explanations or completely misguide the explainer if not executed appropriately.

5.2.2 Trade-off Study

It becomes apparent that, since our explainer models consist of two hyperparameters, we cannot simultaneously monitor the impact of each one of them on the evaluation metics in a single results table. Hence, in this subsection, we will provide a visualization study of how each hyperparameter impacts the explainer. We will provide the effective range of each hyperparameter as well as a way to conceptualize the trade-off between Prediction Flip Rate and Graph Edit Distance.

We will begin our study by denoting all the important characteristics of each figure. First, we will display each dataset separately. For each dataset, we will provide a figure that displays both Prediction Flip Rate and Graph Edit Distance in the same figure for all three models. On the left axis we present Prediction Flip Rate on a scale of $(0\% - 100\%)$ with the color blue. On the right axis we display $1 - GED$ on a scale of $(0 - 1)$ with the color green, and on the x-axis the value of each hyperparameter. We invert Graph Edit Distance because we want the desired values to be at the top as is in PFR to point out the symmetrical nature of the two (as one increases, the other decreases). For each dataset, we will first present the Base Model and then the other two. For the Node Degree Model and the Edge Betweenness Model we will also include the metrics for the Base Model with the colors red and orange in order for improvements to be visible.

Reddit-Binary

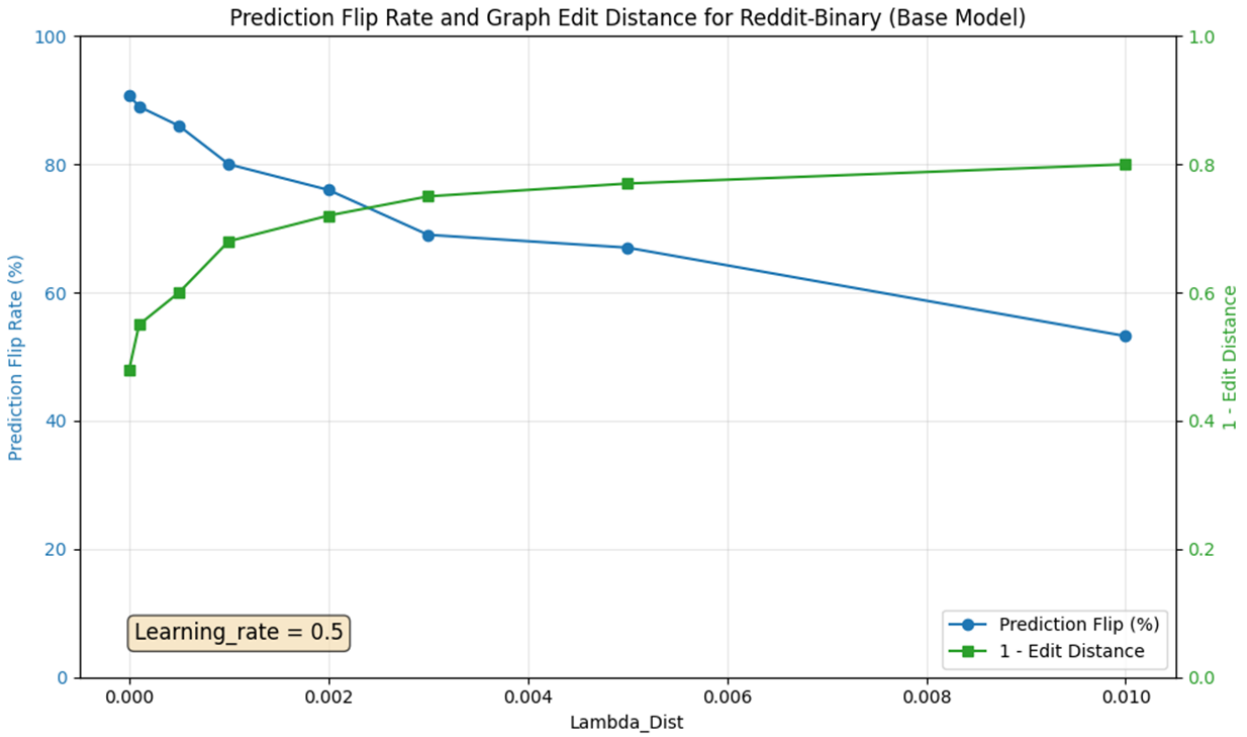


Figure 5.2.1: Base model comparison of Flip Rate and Graph Edit Distance for the Reddit-Binary dataset. The blue line indicates PFR and the green one $1 - GED$ to indicate the trade-off relationship and that optimal behavior is found at the top.

From this figure, we can clearly see the trade-off relationship between Prediction Flip Rate and Graph Edit Distance. The best PFR is acquired at $\lambda_{\text{dist}} = 0$, along with the worst GED. As the hyperparameter increases, the penalty becomes stronger, and the explainer manages to provide fewer counterfactual explanations with fewer edges deleted on average. When the value of λ_{dist} becomes too large, the explainer cannot produce any counterfactual explanations.

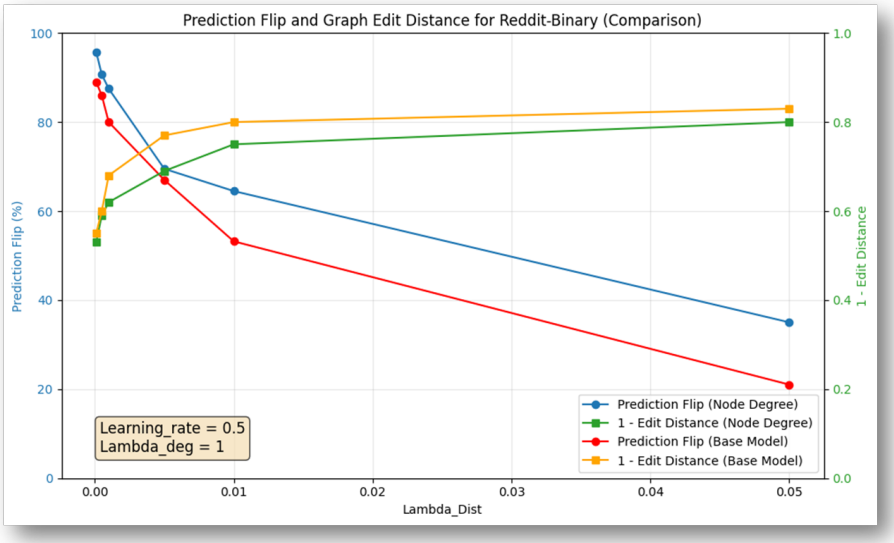


Figure 5.2.2: Node Degree model comparison with the Base model for the Reddit-Binary dataset. The blue line indicates PFR, the green one $1 - GED$ for the Node Degree model and the red and orange ones the same metrics for the Base model respectively.

Here, we can observe that the Node Degree model outperforms the Base model for any value of the distance hyperparameter. In order to display these values we have frozen the constraint hyperparameter value $\lambda_{\text{deg}} = 1$ where the best results occurred. The trade-off relationship is once again visible as the Node Degree model deletes more edges on average in order to increase PFR.

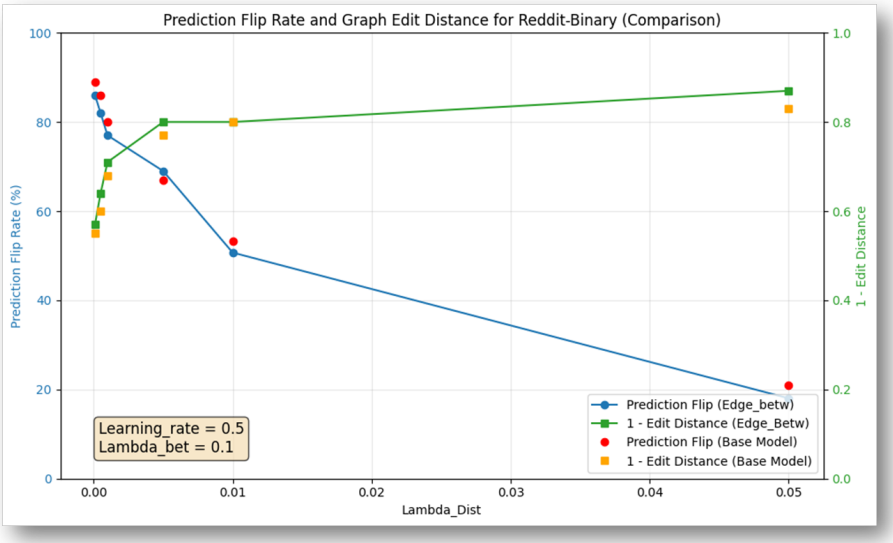


Figure 5.2.3: Edge Betweenness Model comparison with the Base model for the Reddit-Binary dataset. The blue line indicates PFR, the green one $1 - GED$ for the Edge Betweenness model and the red and orange dots the same metrics for the Base model respectively.

The Edge Betweenness model provides a different result than previously. We can see that it remains on the same levels of PFR as in the Base model but achieves so by deleting fewer edges (indicated by the green line being above the orange line in the $1 - GED$ metric).

To sum up the results for the Reddit-Binary dataset, we can clearly see two major improvements to the Base model. First, the Node Degree model manages to increase PFR significantly, providing explanations for more graph instances in the test set. It achieves this by removing more edges that the Base model could not identify as important while using the same hyperparameter λ_{dist}. This indicates that a well-crafted constraint added in the loss function could target graph characteristics and improve the masking approach. Our dataset contains classes that heavily rely on hub nodes. Specifically, the 'Question and Answer' class consists of a hub user that answers questions from other users and is hence connected to the majority of nodes in the graph. So, by targeting the edges connected to this node, we can alter the predicted class more easily. The Edge Betweenness Model on the other hand, doe not improve PFR but in many cases achieves a lower GED than the Base model. This shows that targeting edges that act as a shortest path and transfer critical information in the message passing algorithm can limit deletions and provide minimal explanations. The Reddit-Binary dataset consists of large dense graphs that include many bridge edges, and so a behavior like this is expected.

IMDB-BINARY

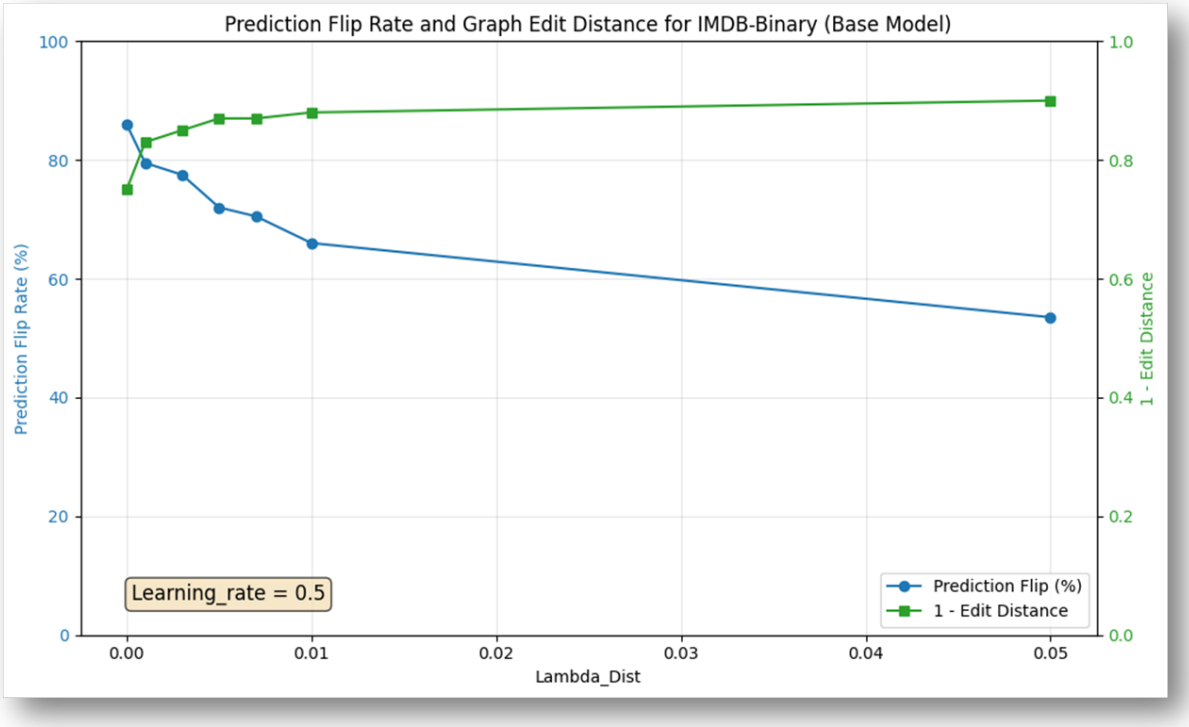


Figure 5.2.4: Base model comparison of Flip Rate and Graph Edit Distance for the IMDB-Binary dataset. The blue line indicates PFR and the green one $1 - GED$ to indicate the trade-off relationship.

Once again, the trade-off relationship between PFR and GED is obvious. As we decrease the hyperparameter λ_{dist}, we achieve better PFR while reducing performance in GED. When the weight of the penalty constraint in the loss function is increased, by increasing λ_{dist}, the explainer is more strict in deleting edges in order to produce counterfactual explanations. Generally, all models perform better on the IMDB-Binary dataset than previously. This can be attributed to the smaller and simpler graphs in the dataset and also to the increased complexity of the GNN used for this dataset.

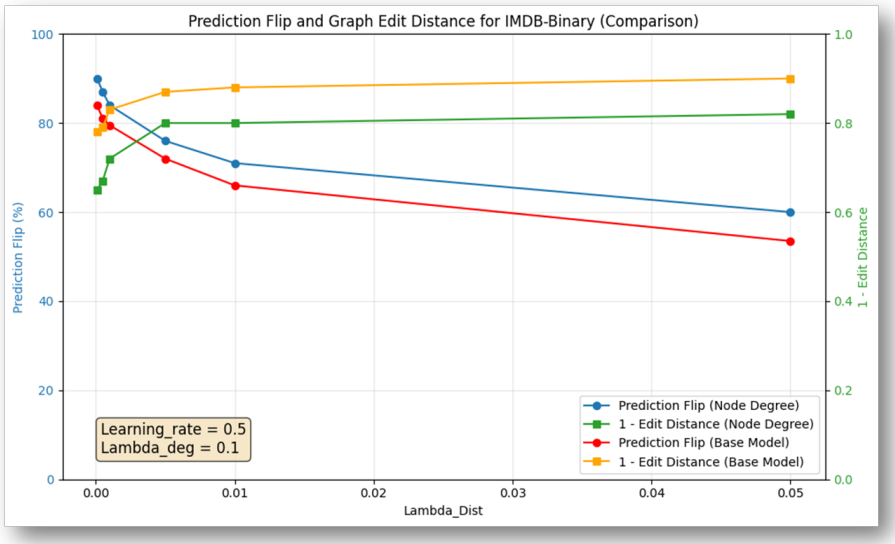


Figure 5.2.5: Node Degree model comparison with the Base model for the IMDB-Binary dataset. The blue line indicates PFR, the green one $1 - GED$ for the Node Degree model and the red and orange ones the same metrics for the Base model respectively.

The Node Degree model again provides the same impact as previously. We can observe an increase in the PFR that the Base model could not achieve while deleting more edges. The IMDB-Binary dateset consists of graphs representing the ego-network of actors that played in movies listed in the IMDB. This results in the existence of a hub node in each graph that connects with every other node. Consequently, guiding the explainer to delete edges connected to this node provides explanations for more graph instances. This is also the reason why the Edge Betweenness model was not included for this dataset, as the lack of distinct shortest paths led to this constraint being counterproductive.

BA-SHAPES

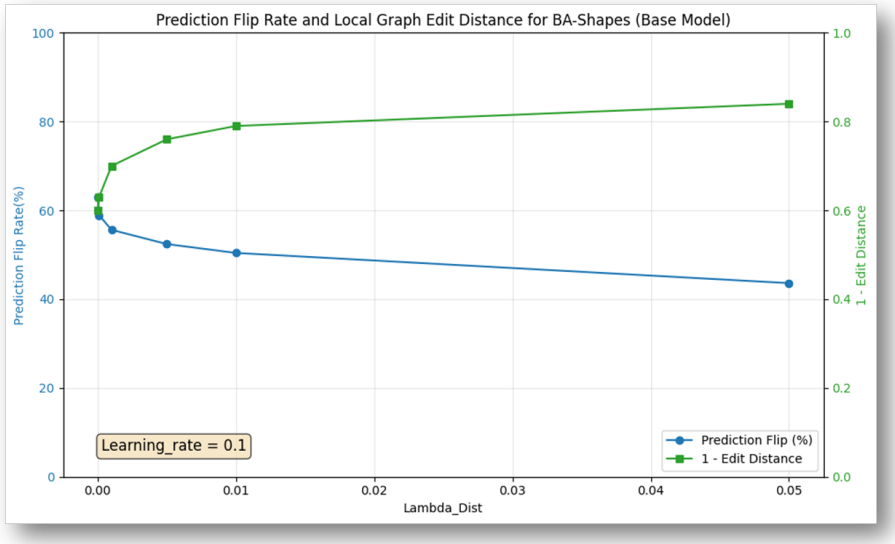


Figure 5.2.6: Base model comparison of Flip Rate and Graph Edit Distance for the BA-Shapes dataset. The blue line indicates PFR and the green one $1 - GED$ to indicate the trade-off relationship.

As far as the Ba-Shapes dataset is concerned, we can see a drop in PFR in the Base model in relation to the other two datasets. This can be attributed to the fact that this node classification task is a multi-label one. In our implementation, we examine the task of flipping the label for a given class towards all other classes separately. Thus, the number of possible labels flips is three times (since we have 4 classes) the number as it would have been if we had accepted the easiest class transition as the only counterfactual. This leads to a general decrease in PFR, since some class transitions are much more difficult than others or may cost more edge deletions. Again, the trade-off between PFR and GED is apparent, as we increase the value of the hyperparameter, we arrive at less explanations but more minimal ones on average.

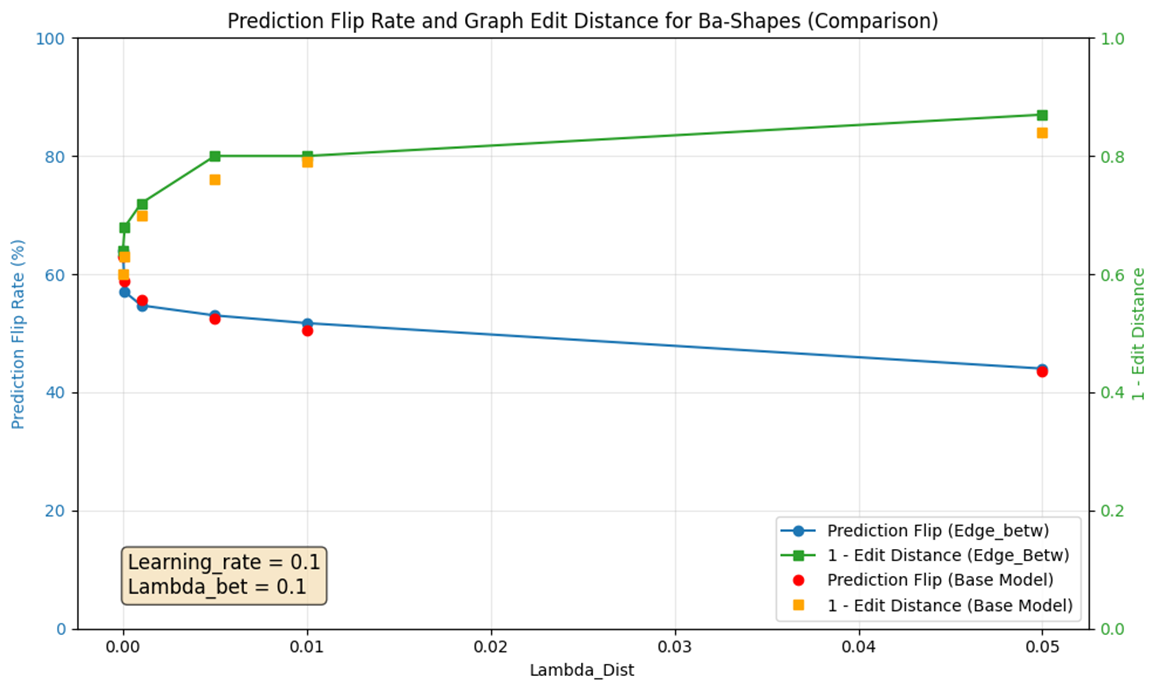
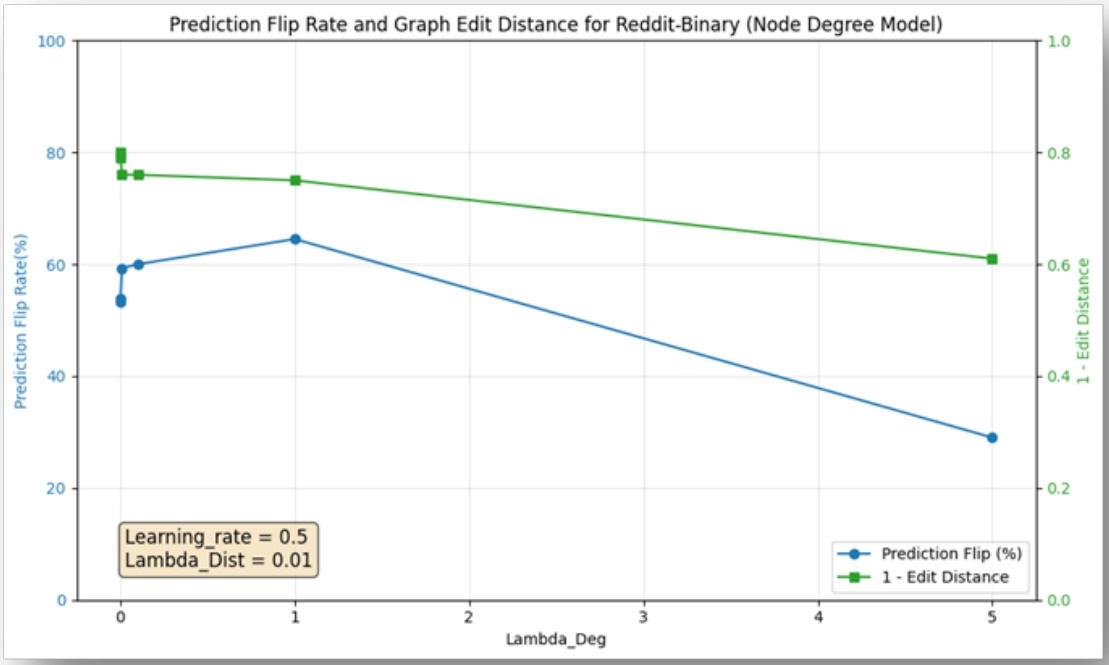


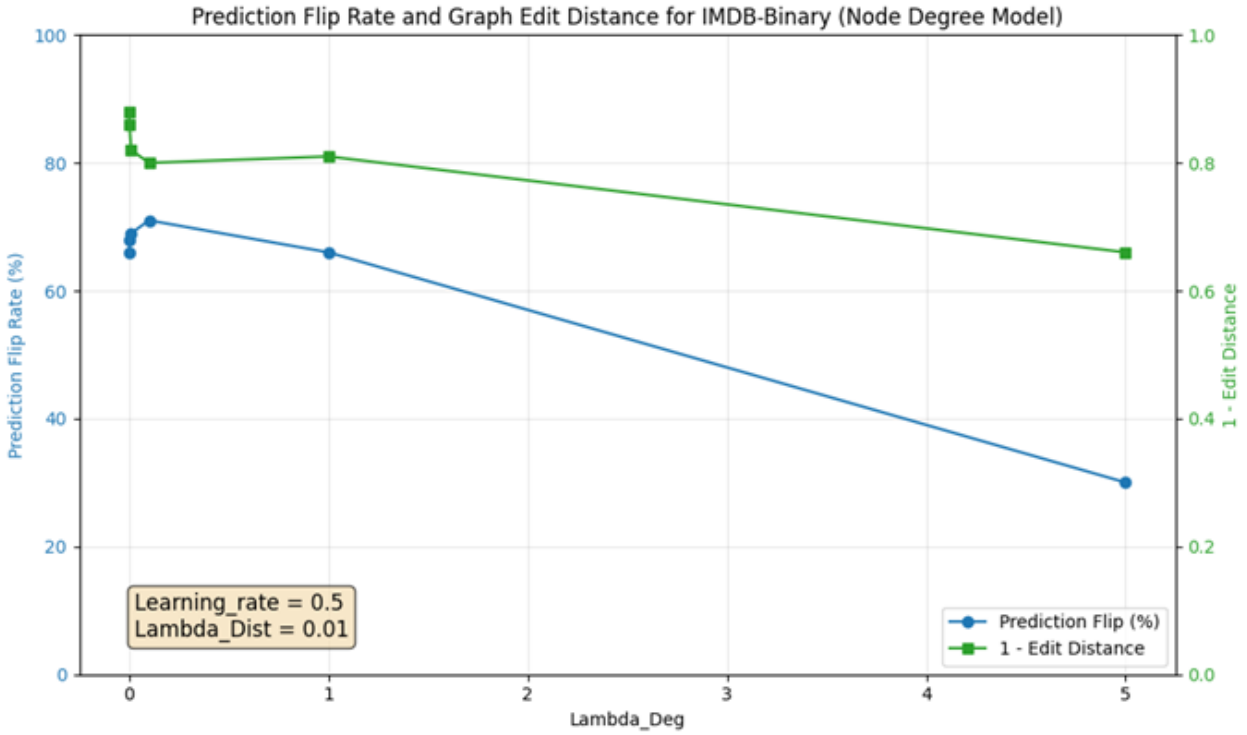
Figure 5.2.7: Edge Betweenness Model comparison with the Base model for the BA-Shapes dataset. The blue line indicates PFR, the green one $1 - GED$ for the Edge Betweenness model and the red and orange dots the same metrics for the Base model respectively.

For the BA-Shapes dataset, the Edge Betweenness model produces the same pattern as before. We can observe the same PFR as the Base model but with the deletion of fewer edges, since the constraint penalizes the keeping of edges in shortest paths in the random Barabasi graph with the attached motifs.

Lastly, we will provide an analysis for how the constraint hyperparameters impact the performance of the explainer. We will do so by setting the value for the distance hyperparameter steady and plotting how changes in the constraint hyperparameter affect both our metrics. What we expect to see is the effective range of the hyperparameters. With values near 0 we expect the model to behave close to the Base model, and with very high values we expect a hindrance of the results, as the loss increases dramatically. The best values that we showcased before, are found in the middle of the range, where the penalty is not too high or not too low and practically guides the explainer to deleting the absolutely necessary amount of edges in order to deliver a counterfactual example. In the following figures, we will present PFR and $1 - GED$ as before in the y-axis and the x-axis we will use λ_{deg} and λ_{bet} for the Node Degree and the Edge Betweenness models respectively.

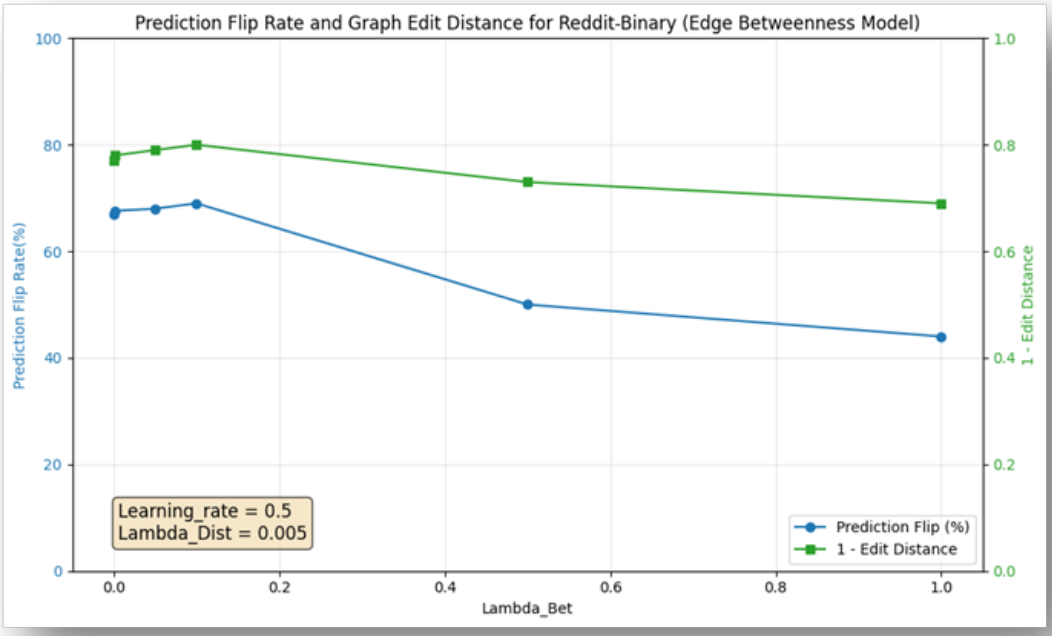


(a) Reddit-Binary dataset.

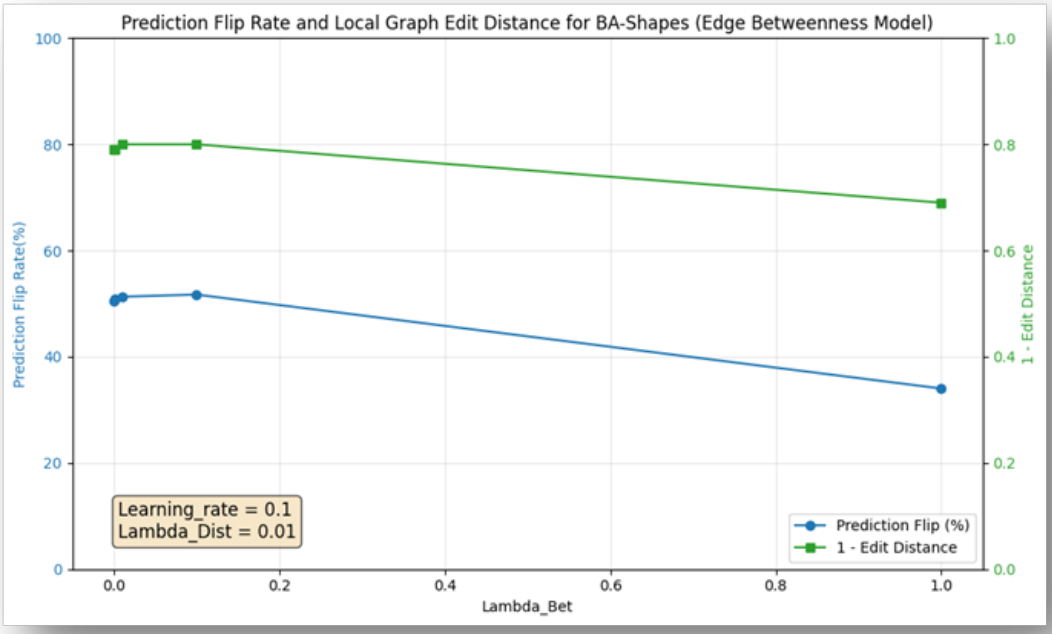


(b) IMDB-Binary dataset

Figure 5.2.8: Representation of how λ_{deg} affects the performance of the Node Degree model for the Reddit-Binary and the IMDB-Binary datasets.



(a) Reddit-Binary dataset.



(b) BA-Shapes dataset

Figure 5.2.9: Representation of how λ_{bet} affects the performance of the Edge Betweenness model for the Reddit-Binary and the BA-Shapes datasets.

By examining these figures, we can arrive at the conclusion that, in most cases, the hyperparameters follow our theoretical expectations. When the constraint hyperparameters are equal to zero, each model is reduced to being the equivalent of the Base model, as excepted. When we increase their values, we can see the real contribution that each constraint offers. At some point, the hyperparameters reach an effective range where they improve on the Base model either the Prediction Flip Rate or the Graph Edit Distance, and after reaching a specific value, they start misguiding the explainer into deleting unnecessary edges or failing to provide a counterfactual example.

5.2.3 Qualitative Examples

Lastly, in this subsection we will provide some qualitative visualization examples of instances from our dataset where a counterfactual explanation was created.

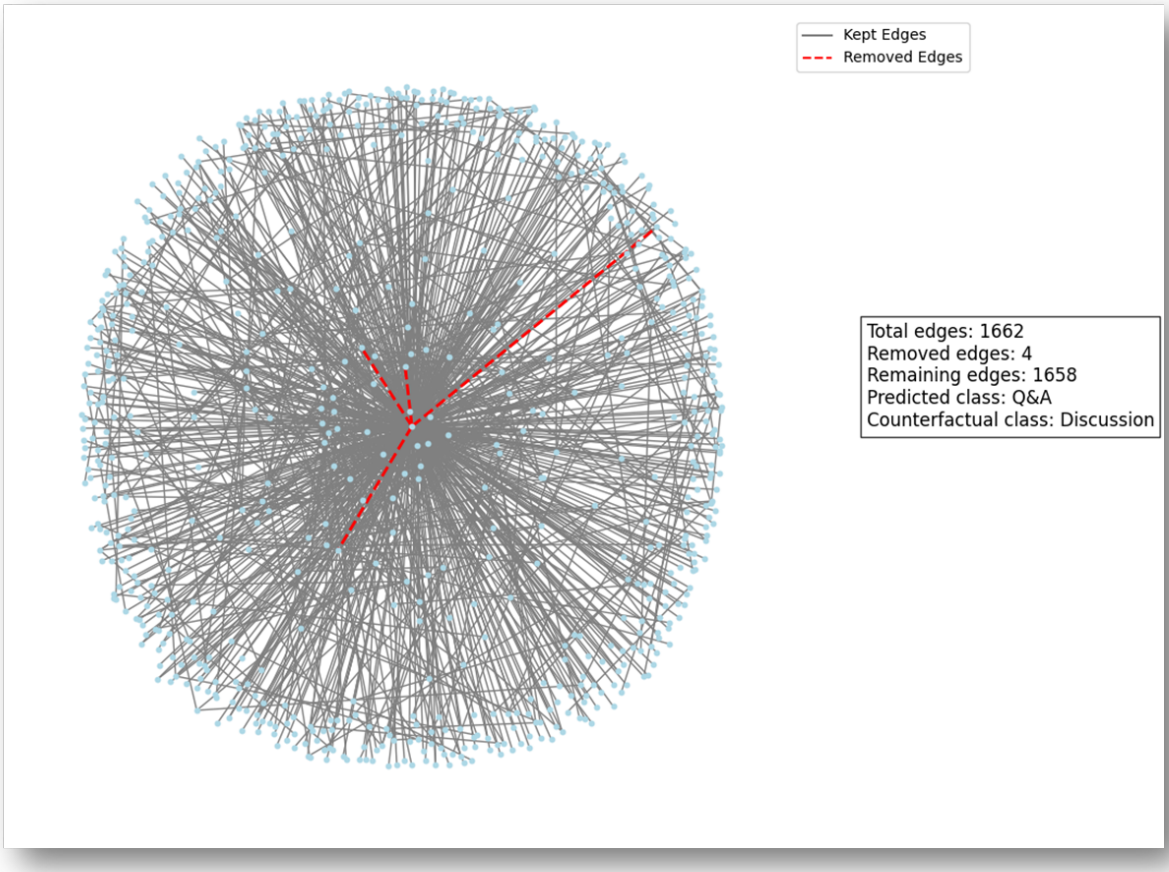


Figure 5.2.10: Counterfactual example from the Reddit-Binary dataset. Nodes are shown in light blue and represent Reddit users. The original graph was labeled 'Q/A' and the counterfactual graph was labeled 'Discussion'. The edges marked with red are the ones deleted. We achieve a counterfactual explanation with 4 edge deletions.

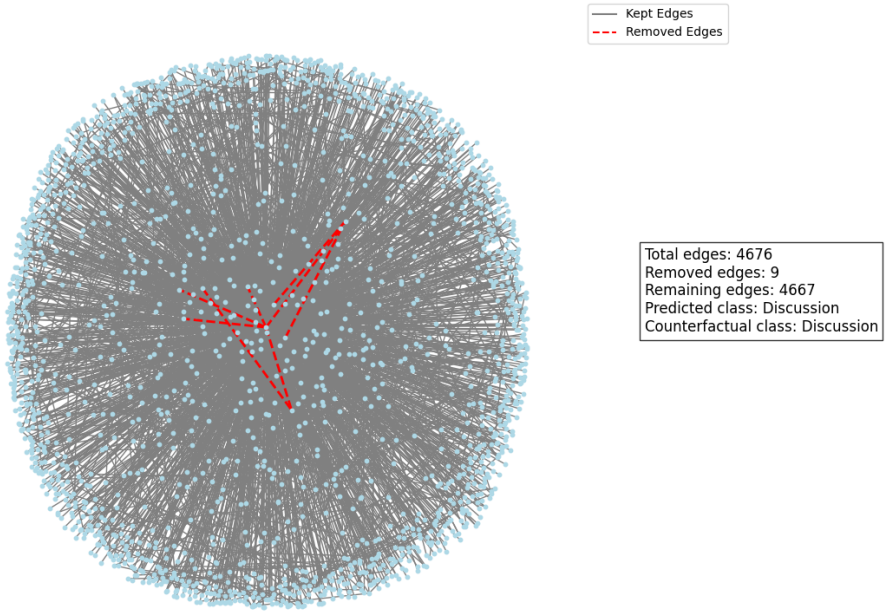


Figure 5.2.11: Counterfactual example from the Reddit-Binary dataset.The original graph was labeled 'Discussion' and the counterfactual graph was also labeled 'Discussion'. Here we can see a failed attempt to create a counterfactual explanation.

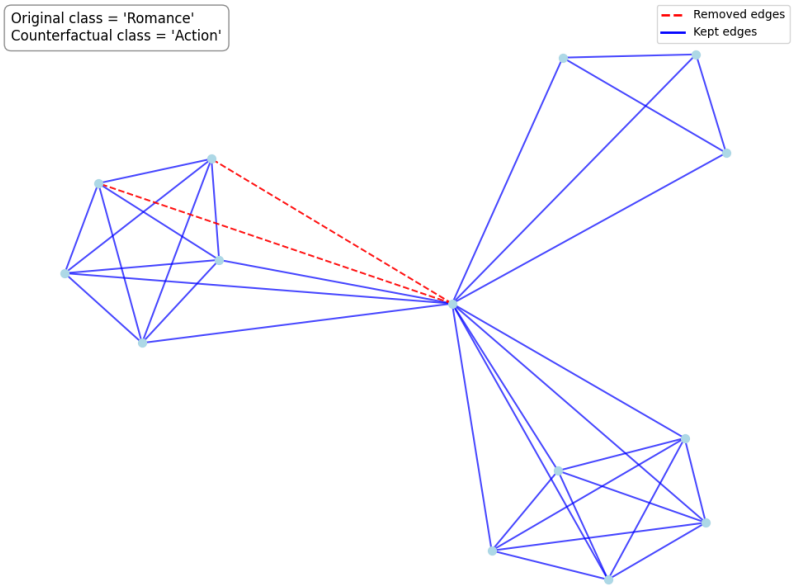


Figure 5.2.12: Counterfactual example from the IMDB-Binary dataset. Nodes represent actors and are shown in light blue. The original graph was labeled 'Romance' and the counterfactual graph was labeled 'Action'. The edges marked with red are the ones deleted. We achieve a counterfactual explanation with 2 edge deletions.

From these two examples, we can clearly see the impact that our explainer has. In the first and third figures we can see that with very few modifications we can achieve an alteration of the predicted class. However, in the second figure, the explainer could not produce a counterfactual example. This example indicates that not all instances can be explained with the same ease. Another important aspect is the confidence of the model. If a model's confidence is too high, then it requires a lot more effort to alter its predictions, regardless of whether they are accurate or not. All of the above indicate that the added constraint in the loss function must be very carefully designed and must represent the characteristics of each dataset.

Chapter 6

Conclusion

6.1 Discussion

Explainable machine learning, particularly in the context of graph neural networks (GNNs), holds substantial promise for improving trust, transparency, and accountability in automated decision-making systems. As black-box models become more complex, the ability to understand their internal mechanisms is vital not only for their users, but also for anyone impacted by their predictions, and requires clarity about how predictions are formed. Within the broader domain of interpretable machine learning, counterfactual explanations have emerged as a powerful tool. These explanations not only identify the components that are relevant to a model's prediction, but they also provide actionable changes that would alter that prediction. By explicitly showing how minimal modifications to the input can flip the model's outcome, counterfactuals support both understanding and actionable interventions.

In this work, we explore the concept of Counterfactual Explanations for Graph Neural Networks. We propose an explainer method for generating counterfactual explanations that selectively perturbs the graph structure by removing edges. These deletions are performed using a continuous masking approach that is then followed by a binary threshold search in order to ensure that the minimality of modifications. We expand on the current bibliography by incorporating an extra term in the loss function to account for domain knowledge. This objective function balances three critical components: encouraging class-flipping behavior in the model, enforcing minimality of the changes introduced (in order to remain faithful to the original graph), and applying domain-specific constraints to guide edge deletions toward plausible or desirable outcomes. These constraints derive from graph characteristics and are designed to reflect on each task's properties and explicit behavior. For this work, we implemented two different constraints for our explainer, one that utilizes Node Degree centrality metrics for nodes inside the graph and one punishing edges with high Edge Betweenness.

For the evaluation of our explainer, we used three benchmark datasets, one for a node classification task, and two for graph classification tasks. To assess performance, we relied on two primary metrics: Prediction Flip Rate and Graph Edit Distance. Prediction flip Rate measures how effectively counterfactual modifications lead to a different predicted label by a black-box model. Meanwhile, GED quantifies the structural cost of these modifications, indicating the difference between the original instance and the counterfactual one. These two metrics have an inherited trade-off relationship. While increasing the prediction flip rate can often require larger structural changes, minimizing structural changes may constrain how frequently flips can be achieved. Therefore, in order to optimize the explainer's ability to make meaningful and minimal explanations, a trade-off study must be performed. This study included the definition of a range for each hyperparameter that the explainer used, and a visualization of the asymmetric relationship among our two metrics.

In conclusion, the contributions of this work are multifarious. First, we introduce two structural constraints that, when added in the loss function of the general counterfactual framework, improve either the number of instances that we can explain or the size of these explanations. To continue, we perform an exhaustive trade-off study that extracts the effective range of each explainer model and provides the user with options

to explain each instance. Lastly, we created the general framework for the addition of any well-designed relevant constraint in the loss function that guides explanations towards any desired outcome.

6.2 Impact

Counterfactual explanations on graph-structured data have profound implications for a multitude of real-world applications, from social network analysis and molecular drug discovery to financial fraud detection. In each of these domains, the ability to identify how small, targeted interventions at the node or edge level can radically alter the outcome of a predictive model is valuable for enhancing trust in those models and generally for the democratization of Artificial Intelligence. Our proposed method directly addresses these needs by offering tailored, actionable insights. By encapsulating domain knowledge within its design, this approach lays a foundation for generating interpretable, reliable, and context-aware explanations, ultimately bolstering user confidence in machine learning systems that operate over complex and interconnected structures.

6.3 Future Work

On a final note, we would like to suggest further avenues and directions that could improve and expand this work's research. As a first step one could try expanding the scope of our counterfactual explainer beyond structural edge deletion to include masking of node and edge features. Our datasets were chosen based on their rich structure and their lack of node and edge characteristics. However, this is not the case for many other datasets and tasks that could be explained using a masking approach of perturbing both edges and features. By moving beyond edge-centric modifications, it becomes possible to capture more intricate and context-specific manipulations that can better align with domain constraints.

Another key direction is the incorporation of semantic constraints within the loss function, ensuring that counterfactuals respect not only the underlying graph structure but also any additional domain knowledge, logical rules, or other semantic patterns. These constraints could be introduced through a wide range of classifiers, trained to reflect particular domain requirements. Integrating multiple classifiers for different semantic properties or real-world plausibility checks would enrich the counterfactual generation process, leading to explanations that are both more nuanced and more robust in practice. Over time, such methods have the potential to further improve the interpretability, reliability, and trustworthiness of machine learning systems that operate on graph-structured data.

Chapter 7

Bibliography

[1] Abrate, C. and Bonchi, F. "Counterfactual Graphs for Explainable Classification of Brain Networks". In: *CoRR* abs/2106.08640 (2021). arXiv: 2106.08640. URL:

[2] Agarwal, C. et al. *Evaluating Explainability for Graph Neural Networks*. 2023. arXiv: 2208.09339 [cs.LG]. URL:

[3] Bajaj, M. et al. "Robust Counterfactual Explanations on Graph Neural Networks". In: *CoRR* abs/2107.04086 (2021). arXiv: 2107.04086. URL:

[4] Barredo Arrieta, A. et al. "Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI". In: *Information Fusion* 58 (2020), pp. 82–115. DOI: 10.1016/j.inffus.2019.12.012.

[5] Bettilyon, T. E. [Online; accessed 10-March-2025]. 2019. URL:

[6] Billington, D. *Factual and Plausible Reasoning*. College Publications, Apr. 2019. ISBN: 978-1-84890-303-6.

[7] Bronstein, M. M. et al. "Geometric deep learning: going beyond Euclidean data". In: *CoRR* abs/1611.08097 (2016). arXiv: 1611.08097. URL:

[8] Cai, R. et al. *On the Probability of Necessity and Sufficiency of Explaining Graph Neural Networks: A Lower Bound Optimization Approach*. 2024. arXiv: 2212.07056 [cs.LG]. URL:

[9] Chhablani, C. et al. *Game-theoretic Counterfactual Explanation for Graph Neural Networks*. 2024. arXiv: 2402.06030 [cs.LG]. URL:

[10] Defferrard, M., Bresson, X., and Vandergheynst, P. "Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering". In: *CoRR* abs/1606.09375 (2016). arXiv: 1606.09375. URL:

[11] Faber, L., Moghaddam, A. K., and Wattenhofer, R. "Contrastive Graph Neural Network Explanation". In: *CoRR* abs/2010.13663 (2020). arXiv: 2010.13663. URL:

[12] Filandrianos, G. et al. "Counterfactuals of Counterfactuals: a back-translation-inspired approach to analyse counterfactual editors". In: *arXiv preprint arXiv:2305.17055* (2023).

[13] Goodfellow, I., Bengio, Y., and Courville, A. *Deep Learning*. MIT Press, 2016.

[14] Guidotti, R. "Counterfactual explanations and how to find them: literature review and benchmarking". In: *Data Mining and Knowledge Discovery* 38 (2024), pp. 2770–2824. DOI: 10.1007/s10618-022-00831-6.

[15] Guo, Z. et al. "Counterfactual Learning on Graphs: A Survey". In: *Machine Intelligence Research* 22.1 (2025), pp. 17–59. DOI: 10.1007/s11633-024-1519-z. URL:

[16] Hamilton, W. L., Ying, R., and Leskovec, J. "Inductive Representation Learning on Large Graphs". In: *CoRR* abs/1706.02216 (2017). arXiv: 1706.02216. URL:

[17] Huang, S. et al. *Reasoning Factual Knowledge in Structured Data with Large Language Models*. 2024. arXiv: 2408.12188 [cs.CL]. URL:

[18] Jiang, J. et al. *Robust Counterfactual Explanations in Machine Learning: A Survey*. 2024. arXiv: 2402.01928 [cs.LG]. URL:

[19] Kipf, T. N. and Welling, M. "Semi-Supervised Classification with Graph Convolutional Networks". In: *CoRR* abs/1609.02907 (2016). arXiv: 1609.02907. URL:

[20] Kosan, M. et al. *Global Counterfactual Explainer for Graph Neural Networks*. 2022. arXiv: 2210.11695 [cs.LG]. URL:
[21] Liartis, J. et al. "Semantic Queries Explaining Opaque Machine Learning Classifiers." In: *DAO-XAI*. 2021.
[22] Liartis, J. et al. "Searching for explanations of black-box classifiers in the space of semantic queries". In: *Semantic Web* 15.4 (2024), pp. 1085–1126.
[23] Liu, Y. et al. "Multi-objective Explanations of GNN Predictions". In: *CoRR* abs/2111.14651 (2021). arXiv: 2111.14651. URL:
[24] Lucic, A. et al. "CF-GNNExplainer: Counterfactual Explanations for Graph Neural Networks". In: *CoRR* abs/2102.03322 (2021). arXiv: 2102.03322. URL:
[25] Lymperaiou, M. et al. "Towards explainable evaluation of language models on the semantic similarity of visual concepts". In: *arXiv preprint arXiv:2209.03723* (2022).
[26] Ma, J. et al. "CLEAR: Generative Counterfactual Explanations on Graphs". In: *Advances in Neural Information Processing Systems (NeurIPS)*. Ed. by S. Koyejo et al. Vol. 35. Curran Associates, Inc., 2022, pp. 25895–25907. URL:
[27] Mastromichalakis, O. M., Liartis, J., and Stamou, G. "Beyond One-Size-Fits-All: Adapting Counterfactual Explanations to User Objectives". In: *arXiv preprint arXiv:2404.08721* (2024).
[28] Mastromichalakis, O. M. et al. "GOSt-MT: A Knowledge Graph for Occupation-related Gender Biases in Machine Translation". In: *arXiv preprint arXiv:2409.10989* (2024).
[29] Mastromichalakis, O. M. et al. "Rule-Based Explanations of Machine Learning Classifiers Using Knowledge Graphs". In: *Proceedings of the AAAI Symposium Series*. Vol. 3. 1. 2024, pp. 193–202.
[30] Mathworks. [Online; accessed 9-March-2025]. 2025. URL:
[31] Menis Mastromichalakis, O. et al. "Semantic Prototypes: Enhancing Transparency Without Black Boxes". In: *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*. 2024, pp. 1680–1688.
[32] Menis Mastromichalakis, O. et al. "Assumed Identities: Quantifying Gender Bias in Machine Translation of Ambiguous Occupational Terms". In: *arXiv e-prints* (2025), arXiv–2503.
[33] MENIS-MASTROMICHALAKIS, O. "Explainable Artificial Intelligence: An STS perspective". In: (2024).
[34] Mitchell, H. et al. "A Network Integration Approach to Predict Conserved Regulators Related to Pathogenicity of Influenza and SARS-CoV Respiratory Viruses". In: *PLOS ONE* 8.7 (2013), e69374. DOI: 10.1371/journal.pone.0069374. URL:
[35] Molnar, C. *Interpretable Machine Learning-A Guide for Making Black Box Models Explain-443 able*. 2019.
[36] Molnar, C. *Interpretable Machine Learning. A Guide for Making Black Box Models Explainable*. 3rd ed. 2025. ISBN: 978-3-911578-03-5. URL:
[37] NG, K. [Online; accessed 10-March-2025]. 2023. URL:
[38] Nguyen, T. et al. "Explaining Black Box Drug Target Prediction Through Model Agnostic Counterfactual Samples". In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 20.2 (Mar. 2023). Epub 2023 Apr 3. PMID: 35820003, pp. 1020–1029. DOI: 10.1109/TCBB.2022.3190266.
[39] Nkwawir, B. [Online; accessed 9-March-2025]. 2020. URL:
[40] Numeroso, D. and Bacciu, D. "MEG: Generating Molecular Counterfactual Explanations for Deep Graph Networks". In: *CoRR* abs/2104.08060 (2021). arXiv: 2104.08060. URL:
[41] Prado-Romero, M. A. et al. "A Survey on Graph Counterfactual Explanations: Definitions, Methods, Evaluation, and Research Challenges". In: *ACM Computing Surveys* 56.7 (Apr. 2024), pp. 1–37. ISSN: 1557-7341. DOI: 10.1145/3618105. URL:
[42] Premptis, I. et al. "AILS-NTUA at SemEval-2025 Task 4: Parameter-Efficient Unlearning for Large Language Models using Data Chunking". In: *arXiv preprint arXiv:2503.02443* (2025).
[43] ResearchGate, S. F. on. *Using Hybrid Models for Action Correction in Instrument Learning Based on AI*. Accessed on 13 March 2025. 2024. URL: (visited on 03/13/2025).
[44] ResearchGate, S. F. on. *COVID-19 detection using machine learning and fusion-based deep learning models*. Accessed on 13 March 2025. 2025. URL: (visited on 03/13/2025).
[45] Rumelhart, D., Hinton, G., and Williams, R. "Learning representations by back-propagating errors". In: *Nature* 323 (1986), pp. 533–536. DOI: 10.1038/323533a0. URL:

[46] Saxena, A. and Iyengar, S. "Centrality Measures in Complex Networks: A Survey". In: *CoRR* abs/2011.07190 (2020). arXiv: 2011.07190. URL:
[47] Sotirou, T. et al. "Musiclime: Explainable multimodal music understanding". In: *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2025, pp. 1–5.
[48] Sun, Y. et al. "Preserve, Promote, or Attack? GNN Explanation via Topology Perturbation". In: *CoRR* abs/2103.13944 (2021). arXiv: 2103.13944. URL:
[49] Tan, J. et al. "Learning and Evaluating Graph Neural Network Explanations based on Counterfactual and Factual Reasoning". In: *Proceedings of the ACM Web Conference 2022*. WWW '22. ACM, Apr. 2022, pp. 1018–1027. DOI: 10.1145/3485447.3511948. URL:
[50] Veličković, P. et al. *Graph Attention Networks*. 2018. arXiv: 1710.10903 [stat.ML]. URL:
[51] Wachter, S., Mittelstadt, B., and Russell, C. "Counterfactual Explanations Without Opening the Black Box: Automated Decisions and the GDPR". In: *Harvard Journal of Law & Technology* 31.2 (2018). Spring issue.
[52] Wellawatte, G. P., Seshadri, A., and White, A. D. "Model agnostic generation of counterfactual explanations for molecules". In: *Chemical Science* 13 (2022), pp. 3697–3705. DOI: 10.1039/D1SC05259D.
[53] Wikipedia contributors. *Cross-entropy — Wikipedia, The Free Encyclopedia*. [Online; accessed 14-March-2025]. 2024.
[54] Wikipedia contributors. *Multilayer perceptron — Wikipedia, The Free Encyclopedia*. [Online; accessed 12-March-2025]. 2024.
[55] Wikipedia contributors. *Weak supervision — Wikipedia, The Free Encyclopedia*. [Online; accessed 14-March-2025]. 2024. URL:
[56] Wikipedia contributors. *Activation function — Wikipedia, The Free Encyclopedia*. [Online; accessed 12-March-2025]. 2025. URL:
[57] Wikipedia contributors. *Adjacency matrix — Wikipedia, The Free Encyclopedia*. [Online; accessed 16-March-2025]. 2025.
[58] Wikipedia contributors. *Barabási–Albert model — Wikipedia, The Free Encyclopedia*. [Online; accessed 10-April-2025]. 2025. URL:
[59] Wikipedia contributors. *Graph edit distance — Wikipedia, The Free Encyclopedia*. [Online; accessed 20-March-2025]. 2025.
[60] Wikipedia contributors. *Loss function — Wikipedia, The Free Encyclopedia*. [Online; accessed 14-March-2025]. 2025.
[61] Wikipedia contributors. *Machine learning — Wikipedia, The Free Encyclopedia*. [Online; accessed 10-March-2025]. 2025. URL:
[62] Wikipedia contributors. *Regularization (mathematics) — Wikipedia, The Free Encyclopedia*. [Online; accessed 13-March-2025]. 2025.
[63] Wikipedia contributors. *Supervised learning — Wikipedia, The Free Encyclopedia*. [Online; accessed 10-March-2025]. 2025. URL:
[64] Wu, H. et al. "Counterfactual Supporting Facts Extraction for Explainable Medical Record Based Diagnosis with Graph Network". In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Ed. by K. Toutanova et al. Online: Association for Computational Linguistics, June 2021, pp. 1942–1955. DOI: 10.18653/v1/2021.naacl-main.156. URL:
[65] L. Wu et al., eds. *Graph Neural Networks: Foundations, Frontiers, and Applications*. 1st ed. Published 04 January 2022. Singapore: Springer Singapore, 2022. ISBN: 978-981-16-6053-5. DOI: 10.1007/978-981-16-6054-2.
[66] Wu, Z. et al. "A Comprehensive Survey on Graph Neural Networks". In: *CoRR* abs/1901.00596 (2019). arXiv: 1901.00596. URL:
[67] Xu, K. et al. "How Powerful are Graph Neural Networks?" In: *CoRR* abs/1810.00826 (2018). arXiv: 1810.00826. URL:
[68] Yanardag, P. and Vishwanathan, S. "Deep Graph Kernels". In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '15. Sydney, NSW, Australia: Association for Computing Machinery, 2015, pp. 1365–1374. ISBN: 9781450336642. DOI: 10.1145/2783258.2783417. URL:

[69] Ying, R. et al. "GNN Explainer: A Tool for Post-hoc Explanation of Graph Neural Networks". In: *CoRR* abs/1903.03894 (2019). arXiv: 1903.03894. URL:

[70] Zhang, A. et al. "Dive into Deep Learning". In: *CoRR* abs/2106.11342 (2021). arXiv: 2106.11342. URL:

[71] Zhang, J. and Luo, Y. "Degree Centrality, Betweenness Centrality, and Closeness Centrality in Social Network". In: *Proceedings of the International Conference on Modeling, Simulation and Applied Mathematics (MSAM)*. Jan. 2017. DOI: 10.2991/msam-17.2017.68.

[72] Zhou, J. et al. "Graph Neural Networks: A Review of Methods and Applications". In: *CoRR* abs/1812.08434 (2018). arXiv: 1812.08434. URL:

[73] Zhou, Y., Zheng, H., and Huang, X. "Graph Neural Networks: Taxonomy, Advances and Trends". In: *CoRR* abs/2012.08752 (2020). arXiv: 2012.08752. URL: