Predicting Vanishing Coefficients of Denominator Graphs: From l-loops to $(l+1)$-loops OR Zero and few shot learning

Contents

1 Introduction

We consider the denominator graphs. It is understood that if the denominator has zero coefficient then any full graph that uses it will also have a coefficient of zero. The denominator graph is likely to represent the easiest task we can solve.

In this short document we will investigate predicting higher-loop coefficients from lower-loop coefficients. This goes towards the overall ambition of predicting coefficients of currently unseen or unknown loops.

2 Tabular Features

The notebook produces a diverse set of graph-based features, each reflecting a specific structural property of the graph. Below we describe each feature group and its potential usefulness.

Core-related Features

- `Core_core_index_mean` — The average *core index* across all nodes. A node's core index measures its position in the k-core decomposition, indicating how deeply embedded it is in the network. Useful for quantifying overall network cohesion.

- `Core_max_core_index` — The maximum core index found in the graph. High values suggest a highly interconnected core, indicating robustness or a central structure.

Cycle-based Features

- `Cycle_num_cycles_len_5`, `Cycle_num_cycles_len_6` — Counts of cycles of length 5 and 6. Cycle counts can reveal repeating motifs in networks, important in chemistry (ring structures in molecules) or in network motif analysis.

Basic Connectivity Metrics

- `EDGES`, `NUM_EDGES` — Number of edges in the graph. A fundamental density measure.
- `DEN_EDGES` — Likely edge density (ratio of actual edges to possible edges). High density means more connections between nodes.

Spectral Graph Theory Features

- `Spectral_algebraic_connectivity` — The second-smallest eigenvalue of the Laplacian; higher values mean the graph is harder to disconnect, useful for robustness analysis.
- `Spectral_laplacian_mean`, `Spectral_laplacian_std`, `Spectral_laplacian_skew` — Statistical moments of the Laplacian spectrum, capturing the shape of the spectrum which reflects graph connectivity patterns.
- `Spectral_spectral_gap` — The difference between the first two eigenvalues of the adjacency or Laplacian matrix; often relates to community structure detectability.

Planarity-based Features

- `Planarity_num_faces` — Number of faces in a planar embedding (if planar). More faces can suggest more complex local structures.
- `Planarity_face_size_max`, `Planarity_face_size_mean` — Maximum and average number of edges per face; reflects how "spread out" cycles are in a planar representation.

Symmetry Features

- `Symmetry_automorphism_group_order` — Size of the automorphism group (number of graph symmetries). Higher values indicate more structural regularity.

- `Symmetry_num_orbits` — Number of distinct node equivalence classes under symmetries. Fewer orbits mean more symmetry.
- `Symmetry_orbit_size_max` — Largest equivalence class of nodes under symmetries; can highlight repeated patterns.

Robustness Features

- `Robust_articulation_points` — Number of articulation points (nodes whose removal increases the number of connected components). Fewer points means more robust connectivity.
- `Robust_bridge_count` — Number of bridges (edges whose removal disconnects the graph). Fewer bridges imply greater robustness.

Global Network Descriptors

- `Kirchhoff_index` — A resistance-based distance measure, summing effective resistances over all pairs of nodes. Useful for characterising transport efficiency or redundancy.

3 Methodology

In this work, our objective is to predict the properties of *unseen loops* by leveraging the currently known lower-loop coefficients. We frame this task as a supervised learning problem using the `XGBoost` algorithm, which is widely recognized as one of the most competitive approaches for tabular data, often outperforming or matching the performance of deep neural networks in such settings. Importantly, this setup naturally corresponds to a *zero-shot learning* scenario: at inference time, the model is required to make predictions for $(l+1)$-loop quantities without having observed any corresponding training data for that loop order. While the primary focus is on the zero-shot setting, we will also explore the impact of incorporating a small amount of $(l+1)$-loop data into training to assess potential performance gains.

This setup corresponds to a zero-shot learning scenario in the strict case where no $(l+1)$-loop data are included during training (with the model generalizing solely from patterns learned at lower loops), whereas the inclusion of even a small fraction of $(l+1)$-loop examples would instead place it in a few-shot learning regime.

We will use cross validation in all our experiments with target stratification - with the average cross validated score over folds being our main result. We will use ROC-AUC to measure performance. In general - the interpretation of the ROC-AUC score is if I randomly choose a graph with coefficient non-zero and a graph with coefficient zero, what is the probability that the model output score of first graph is larger than the second.

Finally, our analysis covers loops 7,8,9 and 10 - mostly because other edge data was not available.

4 Results and conclusions

4.1 Intra-loop and mixed loops

Pair	Loop	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Avg ROC AUC	Std Dev
Single Loops								
–	7	0.789	0.847	0.817	0.845	0.662	0.792	0.068
–	8	0.859	0.865	0.882	0.879	0.869	0.871	0.009
–	9	0.904	0.893	0.900	0.904	0.904	0.901	0.004
–	10	0.914	0.916	0.914	0.916	0.918	0.916	0.001
–	11	0.919	0.920	0.920	0.918	0.919	0.919	0.000
Mixed Loops								
7 & 8	7	0.884	0.888	0.810	0.873	0.814	0.854	0.035
	8	0.847	0.884	0.864	0.874	0.839	0.862	0.017
8 & 9	8	0.868	0.887	0.883	0.882	0.876	0.879	0.007
	9	0.899	0.894	0.906	0.898	0.909	0.901	0.006
9 & 10	9	0.902	0.895	0.896	0.912	0.905	0.902	0.006
	10	0.916	0.914	0.913	0.914	0.916	0.914	0.001
10 & 11	10	0.906	0.905	0.905	0.907	0.909	0.906	0.002
	11	0.919	0.919	0.918	0.918	0.918	0.919	0.000
8 & 10	8	0.850	0.856	0.864	0.872	0.845	0.858	0.010
	10	0.917	0.915	0.914	0.917	0.914	0.915	0.001
7 & 9	7	0.744	0.847	0.782	0.877	0.714	0.793	0.061
	9	0.910	0.888	0.905	0.896	0.902	0.900	0.008
7 & 10	7	0.756	0.736	0.774	0.810	0.662	0.747	0.049
	10	0.918	0.917	0.915	0.916	0.915	0.916	0.001

Table 1: ROC AUC scores for single loops and mixed loop combinations across 5 folds, with mean and standard deviation.

A few notables:

- Mixing $l + 1$ with l always improves performance on l

- Mixing $l+1$ with l seems to degrade performance on $l+1$

$$\forall f_{j,\alpha}^{\nu^{(n-1)}} : \sum_{f_{i,\mu}^{\rho^{(n)}}} \frac{\left|f_{j,\alpha}^{\nu^{(n-1)}}\right|}{\left|f_{i,\mu}^{\rho^{(n)}}\right|} c_i^{(n)} = c_j^{(n-1)}; \quad \text{in particular,} \quad c_0^{(n-1)} \equiv 0$$

Cusp rule tells us that pinching on an l loop coefficient is equivalent to the weighted sum of $l+1$ loop coefficients. This would mean that there is information in $l+1$ loops that l loops can use and we indeed observe this in the results. (Note - we should do experiments or study ways to deconstruct this from simply having more data - not sure how much an issue this is).

4.2 Directionality: Zero-shot learning

In this section - we are modelling upwards - what we can call directionality. We train a model on lower loops in order to predict on higher loops. If we want to make use of a model to predict loops currently unseen we would want to do this.

This is an example of zero-shot learning. We will not show the model any information about higher loops - instead we want it to learn enough structure in lower loops and make good inferences of high loops based on information available.

As an exercise we will show the precision recall curves - recall that high precision means fewer false positives whilst higher recall means fewer false negatives.

4.2.1 $[7] \rightarrow [8, 9, 10]$

Dataset	ROC AUC
Train [7]	1.000
Test [8]	0.760
Test [9]	0.754
Test [10]	0.699

Table 2: Directionality summary: trained on loop 7, evaluated on loops 8–10.

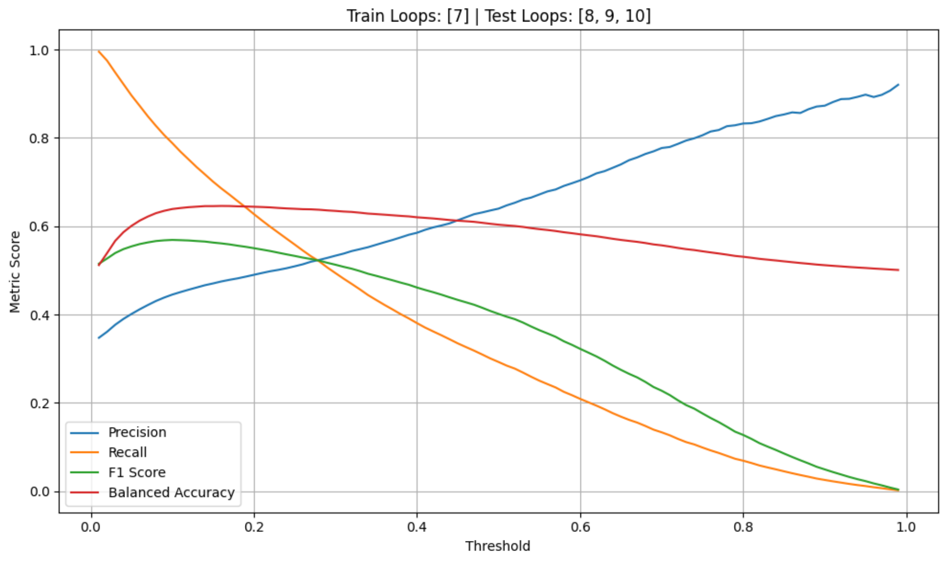

Figure 1: Threshold plots

Clearly - these are expectantly poor results.

- Recall drops off quickly indicating large amount false negatives comes in when we increase the threshold - i.e. bringing up the threshold makes the model think there are zero coefficients when there aren't (we expect this - but its happening quickly).
- Precision never reaches 1. If we push the threshold to near 1, a well calibrated model will reveal the high confidence coefficients - we do not get that here.

4.2.2 $[7, 8] \rightarrow [9, 10]$

Dataset	ROC AUC
Train [7, 8]	1.000
Test [9]	0.790
Test [10]	0.799

Table 3: Directionality summary: trained on loops 7 and 8, evaluated on loops 9 and 10.

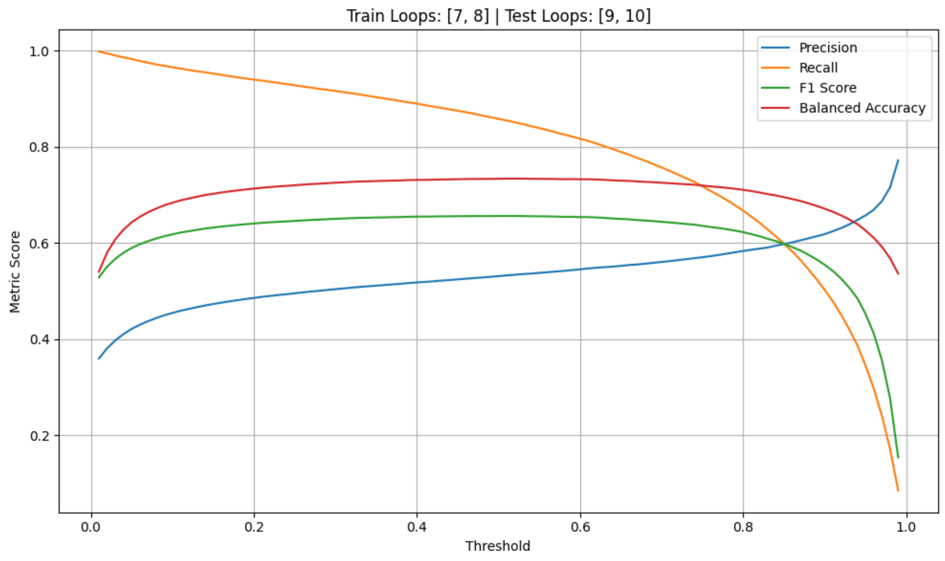

Figure 2: Threshold plots

- Recall drops off much slower than before - bringing the threshold up does not quickly drop coefficients - it is likely that the model is not well calibrated.
- Precision never reaches 1. If we push the threshold to near 1, a well calibrated model will reveal the high confidence coefficients - we do not get that here.

4.2.3 $[7, 8, 9] \rightarrow [10]$

Dataset	ROC AUC
Train [7, 8, 9]	0.997
Test [10]	0.817

Table 4: Directionality summary: trained on loops 7, 8, and 9, evaluated on loop 10.

This is the case that is of interest to us in this current study.

Really importantly - the training AUC-ROC does not reach 1. This indicates conflicting datapoints in loop 9 and data in loops 7 and 8. We

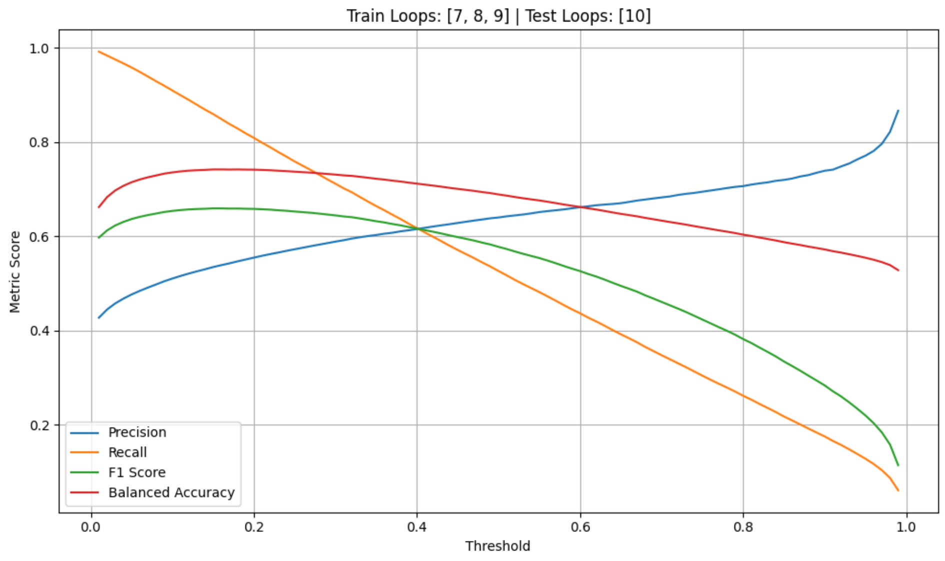

Figure 3: Threshold plots

need to enhance the feature space further to remove this conflict.

- Recall drops off quite linearly indicating the best calibrated we have had so far.
- Again precision never reaches 1. If we push the threshold to near 1, a well calibrated model will reveal the high confidence coefficients - we do not get that here.

4.2.4 $[7, 8, 9, 10] \to [11]$

Dataset	ROC AUC
Train [7, 8, 9,10]	0.943
Test [11]	0.857

Table 5: Directionality summary: trained on loops 7, 8, 9, adn 10 evaluated on loop 11.

4.3 Directionality + pieces of loop 10 needed to get intra-loop 10 level performance: Few-shot learning

In this section we look a the $[7, 8, 9] \rightarrow [10]$ problem again but we now cluster the 10 loop graphs and work out which clusters have the mosts value.

This is something akin for Few-shot learning (although we are taking quite a few more 10 loop examples into a training set than we might normally).

Cluster	Rows $\rightarrow$ Train	Remaining Test Rows	AUC
0	21310	147510	0.889
1	4927	163893	0.831
2	12960	155860	0.864
3	2294	166526	0.850
4	1424	167396	0.852
5	15156	153664	**0.890**
6	2467	166353	0.849
7	16574	152246	0.878
8	21277	147543	**0.890**
9	143	168677	0.834
10	6077	162743	0.838
11	654	168166	0.828
12	2600	166220	0.871
13	6539	162281	0.830
14	2476	166344	0.867
15	8740	160080	0.851
16	16018	152802	0.872
17	12670	156150	0.862
18	164	168656	0.835
19	14350	154470	0.867

Table 6: Per-cluster AUC results when promoting one cluster from loop 10 into training. Highest values in bold.

Promotion Step	Clusters Promoted	Overall AUC
Best 1st Cluster	[5]	0.890
After 2 Clusters	[5, 8]	0.908
After 3 Clusters	[5, 8, 19]	0.917

Table 7: Progressive improvement in AUC as additional clusters are promoted into the training set.

The additional clusters bring in 50783 more datapoints which are 10-loop specific. Training with the same amount of 10-loop data on a 10-loop problem alone gives a ROC AUC of about 0.907. Ultimately - its not so clear if this is a useful thing to do until we do further comparisons.

5 Uncertainty Quantification

TODO

6 Next steps

Questions to ask ourselves in this work

1. Try this on higher loops again
2. Move to full graph
3. Data Augmentation
   a) In vanilla modelling - we expect there to be a rich generating function - not particularly obvious how invariance we have that is not already encoded in features
   b) Fix the 9-loop problem.
4. Confidence scores and how in particular to use for 13-loops
5. Are we finding coefficients for graphs that cannot be found through rules like cusp rules?

On the GNN approach - we should get graph representations that are far richer.

7 Tree Misclassification by inclusion of loop order

Table 8: Overall by combo

train_loops	n_train	train_auc_overall	test_loop	test_auc
(9, 10)	167,224	0.942,495	11	0.869,956
(7, 9, 10)	167,388	0.943,104	11	0.864,540
(8, 9, 10)	168,656	0.942,171	11	0.862,627
(7, 8, 9, 10)	168,820	0.942,916	11	0.856,828
(8, 10)	154,684	0.945,206	11	0.840,198
(7, 8, 10)	154,848	0.945,260	11	0.838,728
(7, 8)	1596	1.000,000	11	0.811,331
(10,)	153,252	0.945,841	11	0.809,871
(8,)	1432	1.000,000	11	0.807,204
(9,)	13,972	0.997,817	11	0.798,707
(7, 9)	14,136	0.998,211	11	0.793,281
(8, 9)	15,404	0.996,969	11	0.789,976
(7, 10)	153,416	0.946,195	11	0.786,496
(7, 8, 9)	15,568	0.996,618	11	0.779,690
(7,)	164	1.000,000	11	0.636,116

8 Addition of new features

8.1 Additional features

9 Feature Engineering Update: From Classical Descriptors to Rich Graph Signatures

Overview. We extend the original feature set (degree statistics, connectivity, centrality, k-core, robustness, limited cycle counts, Laplacian spectral summaries, planarity, symmetry, and Kirchhoff index) with several new, orthogonal families that capture higher-order structure, multiscale diffusion, community organization, and topological signals. The goal is to (i) improve expressivity for small planar graphs, (ii) increase robustness across loop orders by controlling for graph size, and (iii) enable interpretable ablations by grouping features into coherent categories.

Groups and motivations.

(A) Classical structure (baseline). *Degree & density, connectivity (components/diameter/radius/ASPL/Wiener), centralities (betweenness/ close-*

Table 9: Training per-loop AUCs (long-form)

train_loops	loop_id	n_train_loop	train_auc_loop
(7,)	7	164	1.000,000
(7, 8)	7	164	1.000,000
(7, 8)	8	1432	1.000,000
(7, 8, 9)	7	164	1.000,000
(7, 8, 9)	8	1432	0.999,772
(7, 8, 9)	9	13,972	0.996,046
(7, 8, 9, 10)	7	164	0.996,557
(7, 8, 9, 10)	8	1432	0.970,624
(7, 8, 9, 10)	9	13,972	0.946,507
(7, 8, 9, 10)	10	153,252	0.942,077
(7, 8, 10)	7	164	0.986,719
(7, 8, 10)	8	1432	0.970,495
(7, 8, 10)	10	153,252	0.944,890
(7, 9)	7	164	1.000,000
(7, 9)	9	13,972	0.998,165
(7, 9, 10)	7	164	0.971,635
(7, 9, 10)	9	13,972	0.952,468
(7, 9, 10)	10	153,252	0.942,129
(7, 10)	7	164	0.999,016
(7, 10)	10	153,252	0.946,104
(8,)	8	1432	1.000,000
(8, 9)	8	1432	0.999,653
(8, 9)	9	13,972	0.996,556
(8, 9, 10)	8	1432	0.962,762
(8, 9, 10)	9	13,972	0.944,510
(8, 9, 10)	10	153,252	0.941,615
(8, 10)	8	1432	0.977,820
(8, 10)	10	153,252	0.944,844
(9,)	9	13,972	0.997,817
(9, 10)	9	13,972	0.953,727
(9, 10)	10	153,252	0.941,386
(10,)	10	153,252	0.945,841

ness/ eigenvector), k-core indices, articulation points & bridges, limited cycle counts (5/6-cycles), Laplacian spectrum summaries (algebraic connectivity, spectral gap, moments, first 10 eigenvalues), planarity (faces, face sizes), symmetry (automorphism order, orbits), Kirchhoff index. These provide coarse geometry (how connected), flow/importance (centralities), and rigidity (algebraic connectivity, bridges).

(B) Motifs & graphlets (new). *Triangles, wedges, 4-cycles, 4-cliques; induced 4-node graphlets ($K_{1,3}$, P_4, C_4, TailedTriangle, Diamond, K_4); triangle–edge embeddedness statistics (mean/std/median/q90, zero/ $\geq$ 2 fractions).* Motifs capture *localized* wiring rules and are sensitive to small structural differences that the baseline may smooth over; induced graphlets disambiguate shapes with identical degree sequences.

(C) Adjacency-spectrum & energy (new). *Graph energy, Estrada index, adjacency spectral moments (2/3/4).* Complementary to Laplacian features: adjacency spectra reflect walk counts and subgraph densities; Estrada and energy summarize global communicability and overall "activity" of the network.

(D) Heat traces & NetLSD (new). *Laplacian heat trace at multiple scales; NetLSD summary (mean/std/quantiles over* log*-spaced times).* These are *multiscale* diffusion fingerprints invariant to node relabeling, providing stable comparisons across graphs and enhancing cross-loop generalization.

(E) Topological Data Analysis (new). *Vietoris–Rips persistent homology on shortest-path distances: H_0/H_1 counts, total/mean/max persistence, persistence entropy, mean birth/death; Betti numbers at distance quantiles.* Persistence measures quantify the *lifetimes* of connectivity and loop-like features as the metric thickens, offering robustness to small perturbations and complementary global shape information.

(F) Community structure (new). *Louvain modularity, community count/max size, community-size Gini, internal-edge fraction.* Detects meso-scale organization; modular graphs can differ in coefficients even with similar local statistics.

(G) Distance/coverage & local distributions (new). *Effective diameter (p90 of distances), eccentricity mean/q90, degree assortativity, clustering coefficient mean/quantiles, fractions at* 0 *and* 1*, degree Gini.* Captures tail behavior and heterogeneity beyond means, which often governs extremal behaviors relevant to coefficients.

(H) Normalized/size-invariant variants (new). *Per-node/edge rates, divisions by combinatorial maxima ($\binom{n}{k}$), ratios over average degree or diameter, log-scaled automorphism size over* $\log(n!)$*, etc.* These reduce confounding by graph size and facilitate *cross-loop* transfer by aligning distributions across n.

Why these additions help. (i) *Higher-order locality:* Motifs and induced graphlets distinguish micro-patterns invisible to degree-based summaries. (ii) *Multiscale sensitivity:* Heat traces/NetLSD encode diffusion at multiple time scales, bridging local and global structure. (iii) *Global shape:* TDA summarizes how components and cycles persist across thresholds, complementing spectral views. (iv) *Meso-scale organization:* Community features identify clustered vs. homogeneous wiring, often predictive in planar settings. (v) *Stability and transfer:* Normalizations mitigate size effects and improve generalization when predicting at unseen loop orders.

Practical notes. All new groups are compatible with tree-based learners (nonlinear, interaction-aware) and support interpretable ablations: train with one group at a time, or remove a group from the full set and measure Δ performance. Normalized counterparts are recommended when mixing loop orders, while raw counts can be informative within a fixed n.

9.1 results

Pair	Loop	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Avg ROC AUC	Std Dev
				Single Loops				
–	7	0.905	0.876	0.798	0.813	0.840	0.846	0.040
–	8	0.896	0.915	0.905	0.947	0.886	0.910	0.021
–	9	0.940	0.936	0.935	0.941	0.937	0.938	0.002
–	10	0.944	0.945	0.943	0.945	0.946	0.945	0.001
				Mixed Loops				
7 & 8	7	0.967	0.971	0.841	0.964	0.866	0.922	0.056
	8	0.902	0.918	0.918	0.927	0.885	0.910	0.015
8 & 9	8	0.908	0.927	0.927	0.943	0.906	0.922	0.014
	9	0.939	0.940	0.934	0.932	0.943	0.938	0.004
7 & 9	7	0.926	0.893	0.806	0.790	0.801	0.843	0.055
	9	0.942	0.929	0.944	0.941	0.944	0.940	0.006
9 & 10	9	0.940	0.935	0.936	0.940	0.937	0.938	0.002
	10	0.944	0.943	0.943	0.944	0.945	0.944	0.001

Table 10: ROC AUC scores for single loops and mixed loop combinations across 5 folds, with mean and standard deviation.

10 Tree Misclassification by Inclusion of Loop Order (Updated Results)

Table 11: Overall by combo (updated results)

train_loops	n_train	train_auc_overall	test_loop	test_auc	best_ntrees
(7, 8, 9, 10)	168,820	0.968,639	11	0.902,108	600
(9, 10)	167,224	0.968,027	11	0.900,879	600
(7, 10)	153,416	0.970,139	11	0.900,487	600
(8, 9, 10)	168,656	0.968,276	11	0.899,943	600
(7, 8, 10)	154,848	0.970,081	11	0.898,898	600
(10,)	153,252	0.970,283	11	0.898,484	600
(8, 10)	154,684	0.970,250	11	0.896,242	600
(7, 9, 10)	167,388	0.968,272	11	0.891,610	600
(7, 9)	14,136	0.997,864	11	0.869,870	600
(9,)	13,972	0.998,239	11	0.868,378	595
(7, 8, 9)	15,568	0.998,339	11	0.867,978	600
(8, 9)	15,404	0.998,303	11	0.866,817	600
(7, 8)	1596	0.990,986	11	0.836,198	103
(8,)	1432	0.993,116	11	0.813,402	127
(7,)	164	0.989,179	11	0.699,997	68

Table 12: Training per-loop AUCs (updated results, long-form)

train_loops	loop_id	n_train_loop	train_auc_loop
(7,)	7	164	0.989,179
(7, 8)	7	164	0.987,211
(7, 8)	8	1432	0.991,405
(7, 8, 9)	7	164	1.000,000
(7, 8, 9)	8	1432	0.999,686
(7, 8, 9)	9	13,972	0.998,152
(7, 8, 9, 10)	7	164	0.998,196
(7, 8, 9, 10)	8	1432	0.986,078
(7, 8, 9, 10)	9	13,972	0.971,860
(7, 8, 9, 10)	10	153,252	0.968,128
(7, 8, 10)	7	164	0.998,360
(7, 8, 10)	8	1432	0.986,166
(7, 8, 10)	10	153,252	0.969,894
(7, 9)	7	164	0.988,031
(7, 9)	9	13,972	0.997,944
(7, 9, 10)	7	164	0.999,508
(7, 9, 10)	9	13,972	0.973,026
(7, 9, 10)	10	153,252	0.967,793
(7, 10)	7	164	0.998,360
(7, 10)	10	153,252	0.970,093
(8,)	8	1432	0.993,116
(8, 9)	8	1432	0.998,997
(8, 9)	9	13,972	0.998,239
(8, 9, 10)	8	1432	0.989,159
(8, 9, 10)	9	13,972	0.971,840
(8, 9, 10)	10	153,252	0.967,733
(8, 10)	8	1432	0.988,593
(8, 10)	10	153,252	0.970,073
(9,)	9	13,972	0.998,239
(9, 10)	9	13,972	0.974,006
(9, 10)	10	153,252	0.967,488
(10,)	10	153,252	0.970,283

10.1 Per category Ablations

Table 13: Ablation-style evaluation of feature categories. Top block: models trained on a single category. Middle: all-features baseline. Bottom: minus-one ablations (more negative Δ vs. ALL = more important).

Category / Setting	**#Features**	**Test AUC**	**Best Trees**	Δ vs. ALL
Per-category models (higher AUC is better)				
Centrality	14	0.877	600	–
Spectral (Laplacian)	27	0.860	600	–
Spectral (Adjacency/Energy)	12	0.799	599	–
Connectivity	13	0.796	291	–
Assortativity/Clustering	8	0.765	600	–
Community	5	0.752	600	–
Motifs (counts)	15	0.744	600	–
Symmetry	7	0.740	274	–
Motifs (induced-4)	13	0.719	600	–
Planarity	5	0.638	92	–
Basic	12	0.630	94	–
Cycle	2	0.624	212	–
TDA	38	0.547	46	–
Robustness	4	0.500	229	–
Core	2	0.500	14	–
(Other/Unassigned)	1	0.499	394	–
All-features baseline				
ALL features	178	0.902	600	reference
Minus-one ablations (more negative Δ = more important)				
Assortativity/Clustering	170	0.882	600	-0.020
Centrality	164	0.887	600	-0.015
Spectral (Laplacian)	151	0.887	600	-0.015
Community	173	0.893	600	-0.009
Connectivity	165	0.894	600	-0.008
Motifs (induced-4)	165	0.896	600	-0.007
Symmetry	171	0.896	600	-0.006
Basic	166	0.898	600	-0.004
(Other/Unassigned)	177	0.899	600	-0.003
Robustness	174	0.899	600	-0.003
Motifs (counts)	163	0.899	600	-0.003
Cycle	176	0.899	600	-0.003
TDA	140	0.900	600	-0.002
Planarity	173	0.901	600	-0.001
Core	176	0.901	600	-0.001
Spectral (Adjacency/Energy)	166	0.902	600	-0.000

11 adding in 5 motifs

In this next section we add a selection of new features surroounding the 5 motifs.

5-node / 5-cycle Motif Features

5-cycle (`Motif_5_cycles`) A *5-cycle* is a simple ring of five distinct nodes:

$$A - B - C - D - E - A,$$

with each node connected to exactly its two neighbors and *no chords* (no shortcuts across the ring). The code counts how many such rings appear in the graph.

Normalized 5-cycle counts To compare graphs of different sizes, the raw count is normalized by the number of 5-node choices:

$$\texttt{Motif\_5\_cycles\_per\_Cn5} = \frac{\texttt{Motif\_5\_cycles}}{\binom{n}{5}}.$$

There is also a density-style normalization against (theoretical) maximum in a complete graph:

$$\texttt{Motif\_5\_cycles\_per\_Kn}.$$

5-clique (`Motif_5_cliques`) A *5-clique* is a fully connected set of five nodes (every pair has an edge). The code counts how many such fully interlinked 5-node groups exist. A size-normalized version is:

$$\texttt{Motif\_5\_cliques\_per\_Cn5} = \frac{\texttt{Motif\_5\_cliques}}{\binom{n}{5}}.$$

Induced 5-node motifs (`Motif_induced5_*`) Beyond cycles and cliques, there are many possible *connected* shapes on five nodes (e.g., a square with a tail, a "house" shape, etc.). For each 5-node subset, the code forms the *induced* subgraph (include exactly those edges that exist among the five nodes in the original graph). If that induced subgraph is connected, it is classified (via Weisfeiler–Lehman hashing) into a canonical type labeled

$$\texttt{G5\_00}, \texttt{G5\_01}, \ldots$$

and counted in columns like `Motif_induced5_G5_xx`. Each also has a normalized version:

$$\texttt{Motif\_induced5\_G5\_xx\_per\_Cn5} = \frac{\texttt{Motif\_induced5\_G5\_xx}}{\binom{n}{5}}.$$

Connected 5-sets fraction The script also records how often a random 5-node choice yields a connected induced subgraph:

$$\texttt{Motif\_induced\_connected\_per\_5set} = \frac{\#\{\text{connected induced 5-node subgraphs}\}}{\binom{n}{5}}.$$

Why "induced" matters "Induced" means the pattern must match *exactly* the edges among the five nodes—no missing and no extra edges. For example, a clean 5-cycle differs from a 5-cycle with a chord; these are counted as different motifs.

Sampling and performance Enumerating all 5-node subsets is combinatorial. The flag `--ind5-sample-frac` (default 1.0) allows sampling a fraction of 5-sets. Use the normalized columns ("`_per_Cn5`") to compare across graphs or runs when sampling is used.

How to read the numbers

- High `Motif_5_cycles`: many chordless 5-node loops ("loopy" structure).
- High `Motif_5_cliques`: many very dense 5-node pockets (tight communities).
- Specific `Motif_induced5_G5_xx` high: that exact 5-node shape is common; compare via the normalized `_per_Cn5` versions.

Pair	Loop	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Avg ROC AUC	Std Dev
Single Loops								
–	6	0.850	0.100	1.000	1.000	1.000	0.790	0.350
–	7	0.897	0.868	0.794	0.817	0.823	0.840	0.037
–	8	0.931	0.937	0.922	0.942	0.904	0.927	0.014
–	9	0.955	0.954	0.952	0.953	0.955	0.954	0.001
–	10	0.956	0.957	0.955	0.956	0.958	0.957	0.001
Mixed Loops								
6 & 7	6	1.000	0.600	1.000	1.000	1.000	0.920	0.160
	7	0.909	0.884	0.764	0.881	0.734	0.835	0.071
7 & 8	7	0.983	0.967	0.849	0.952	0.922	0.935	0.047
	8	0.933	0.925	0.927	0.937	0.923	0.929	0.005
8 & 9	8	0.956	0.963	0.951	0.969	0.926	0.953	0.015
	9	0.957	0.956	0.952	0.952	0.959	0.955	0.003
9 & 10	9	0.954	0.949	0.954	0.956	0.955	0.953	0.002
	10	0.957	0.955	0.956	0.956	0.957	0.956	0.001
Progressive Training / Transfer Tests								
5,6 $\rightarrow$ 7	Train			1.000			1.000	–
	Test (7)			–			0.831	–
5,6,7 $\rightarrow$ 8	Train			1.000			1.000	–
	Test (8)			–			0.800	–
5,6,7,8 $\rightarrow$ 9	Train			1.000			1.000	–
	Test (9)			–			0.828	–
5,6,7,8,9 $\rightarrow$ 10	Train			1.000			1.000	–
	Test (10)			–			0.852	–

Table 14: ROC AUC scores for single loops, mixed loop combinations, and progressive training-transfer tests across folds, with mean and standard deviation.

12 f-graphs

12.1 TODO: feature explanation

12.2 Results

TODO: parameter search

Pair	Loop	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Avg ROC AUC	Std Dev
Single Loops (F-Graphs)								
–	6	1.000	1.000	0.700	0.900	1.000	0.920	0.117
–	7	0.853	0.882	0.796	0.813	0.920	0.853	0.045
–	8	0.956	0.952	0.953	0.948	0.953	0.953	0.002
–	9	0.968	0.970	0.967	0.970	0.967	0.968	0.001
Mixed Loops (F-Graphs)								
6 & 7	6	1.000	1.000	0.900	0.900	1.000	0.960	0.049
	7	0.901	0.835	0.991	0.848	0.829	0.881	0.061
7 & 8	7	0.936	0.981	0.966	0.914	0.996	0.958	0.030
	8	0.949	0.953	0.953	0.948	0.960	0.953	0.004
8 & 9	8	0.973	0.979	0.974	0.968	0.974	0.973	0.003
	9	0.969	0.970	0.969	0.967	0.968	0.969	0.001
Progressive Training / Transfer Tests (F-Graphs)								
5,6 $\to$ 7	Train			1.000			1.000	–
	Test (7)			–			0.820	–
5,6,7 $\to$ 8	Train			1.000			1.000	–
	Test (8)			–			0.860	–
5,6,7,8 $\to$ 9	Train			1.000			1.000	–
	Test (9)			–			0.910	–

Table 15: ROC AUC scores for f-graph single loops, mixed loop combinations, and progressive training-transfer tests across folds, with mean and standard deviation.