Embedding-Based Markov Blanket Discovery for Robust Feature Selection Across Environments

Muhammed Hunaid Topiwala
mtopiwal@asu.edu

Ang Li
angli24@asu.edu

November 2025

Abstract

Causal feature selection through Markov Blanket (MB) discovery offers theoretical guarantees for optimal prediction while maintaining interpretability. However, identifying the MB without access to the underlying causal graph remains challenging, particularly under distribution shift. We present a meta-learning approach that combines TabPFN embeddings with neural MB predictors to perform simultaneous feature selection and regression across heterogeneous tasks. Our method trains separate multi-layer perceptron classifiers on TabPFN-extracted representations to predict binary MB masks, which then filter features for downstream regression. Evaluated on 182 meta-training tasks and 46 held-out tasks spanning multiple data-generating processes and environmental conditions, our approach achieves a validation score of 0.1822 (RMSE: 0.5997, Jaccard: 0.7583), demonstrating the feasibility of learning MB structure from embeddings without explicit causal discovery.

1 Introduction

Feature selection is fundamental to machine learning, directly impacting model performance, interpretability, and generalization. Traditional methods often select features based on correlation with the target variable, which can fail under distribution shift when spurious correlations change across environments [1]. Causal feature selection addresses this limitation by identifying features with stable causal relationships to the target.

The Markov Blanket (MB) of a target variable provides a theoretically optimal feature set for prediction. Pearl [1] showed that the MB renders the target conditionally independent of all other variables, implying that knowing the MB is sufficient and necessary for optimal prediction. For a target variable Y, the MB consists of its parents, children, and spouses (parents of children) in the causal graph. Our experimental work in Labs 2 and 3 empirically validated this optimality: oracle MB features achieved the lowest RMSE (0.329) compared to all features (0.332) in IID settings, and maintained robustness across out-of-distribution scenarios including covariate shift (RMSE: 0.337) and label shift (RMSE: 0.406).

Despite this theoretical and empirical support, discovering the MB in practice remains challenging. Standard approaches require causal discovery algorithms to first recover the full causal graph, then extract the MB. However, causal discovery methods are computationally expensive, require strong assumptions, and can be unreliable on finite samples. Our evaluation in Lab 1 showed that even the best causal discovery method (GES) achieved only moderate F1 scores (0.61) with high variance.

We address this challenge by framing MB discovery as a meta-learning problem. Given a distribution over causal data-generating processes, we learn to predict MB masks directly from tabular

data without explicit graph reconstruction. Our approach leverages TabPFN [3], a transformer-based foundation model for tabular data, to extract rich representations that capture both predictive patterns and structural properties of datasets. These embeddings serve as inputs to learned MB predictors that generalize across tasks with varying feature dimensions and causal structures.

Our main contributions are: (1) A practical pipeline for MB-based feature selection that bypasses causal discovery, (2) Demonstration that TabPFN embeddings encode sufficient information for cross-task MB prediction, and (3) Empirical validation across 228 tasks with diverse data-generating processes and environmental conditions.

2 Related Works

Markov Blanket Theory. Pearl [1] formalized the concept of the Markov Blanket and proved its sufficiency for prediction. Tsamardinos and Aliferis [2] developed algorithms for MB discovery from observational data. Recent work by Nastl et al. (2024) challenged conventional wisdom by showing that individual causal feature sets (parents or children alone) can underperform non-causal baselines. Our Lab 3 results reconcile this finding: while parents (RMSE: 0.493) and children (RMSE: 0.513) individually performed poorly, the complete MB (parents + children + spouses) achieved optimal performance (RMSE: 0.329).

TabPFN and Tabular Foundation Models. Hollmann et al. [3] introduced TabPFN, a transformer trained on synthetic datasets that performs in-context learning for tabular prediction. Unlike traditional models that require task-specific training, TabPFN processes train-test pairs in a single forward pass, making predictions based on learned priors over data-generating processes. TabPFN 2.5 extends this capability to regression tasks with up to 50,000 samples and 2,000 features. The model's embedding layer, accessed through the tabpfn_extensions package, provides dense representations that capture both statistical patterns and structural properties of tabular data.

Meta-Learning for Tabular Data. Meta-learning frameworks learn to adapt quickly to new tasks by leveraging experience from related tasks. In the tabular domain, this typically involves learning good initializations or adaptation strategies. Our approach differs by learning a direct mapping from embeddings to structural properties (MB masks), treating MB discovery as a multi-label classification problem across tasks.

Causal Feature Selection. Traditional causal feature selection methods like PC algorithm, GES, and NOTEARS first recover the causal graph, then extract relevant features. Our Lab 1 evaluation showed these methods achieve F1 scores ranging from 0.48 (PC) to 0.61 (GES), with significant computational cost (up to 9.8 seconds per graph). Our embedding-based approach bypasses explicit graph recovery, directly predicting MB masks in a fraction of the time.

Software and Libraries. Our implementation builds on PyTorch [4] for neural network training, scikit-learn [5] for evaluation metrics and data utilities, TabPFN [3] for feature extraction and regression, and Hugging Face datasets [6] for data management. All experiments were conducted in Python 3.12 using the pytorch-gpu-2.3.1-cuda-12.1 environment.

3 Model Description

Our approach consists of a three-stage pipeline that performs MB prediction followed by filtered regression.

3.1 Stage 1: Embedding Extraction

For each task with support set (X_{train}, y_{train}) and query set X_{test}, we extract dense representations using TabPFN's embedding layer. To obtain robust embeddings, we employ n-fold cross-validation on the support set:

1. Split X_{train} into $K = 5$ folds
2. For each fold k, train TabPFN on the remaining $K - 1$ folds
3. Extract embeddings for the held-out fold
4. Concatenate embeddings across folds to obtain representations for all support samples

We use $N_{est} = 8$ ensemble members per fold, where each ensemble member is an independent TabPFN forward pass with different random initializations. The final embedding for each sample is obtained by averaging the outputs across all ensemble members, yielding a 192-dimensional representation per sample (TabPFN's native embedding dimension). This n-fold embedding strategy provides more robust representations than single-pass embedding, at the cost of $K \times N_{est}$ forward passes.

3.2 Stage 2: MB Prediction via Neural Networks

The meta-training dataset contains tasks with two distinct feature dimensionalities: 87 tasks with 9 features and 95 tasks with 19 features. Rather than using a single unified model with padding, we train separate MB predictors for each dimensionality, allowing each model to specialize to its input space.

For tasks with $d \in \{9, 19\}$ features, we define:

$$\text{MLP}_d : \mathbb{R}^{192} \to [0,1]^d \tag{1}$$
$$h_1 = \text{ReLU}(W_1 x + b_1) \tag{2}$$
$$h_1' = \text{Dropout}(h_1, p = 0.3) \tag{3}$$
$$h_2 = \text{ReLU}(W_2 h_1' + b_2) \tag{4}$$
$$h_2' = \text{Dropout}(h_2, p = 0.3) \tag{5}$$
$$\hat{m} = \sigma(W_3 h_2' + b_3) \tag{6}$$

where $W_1 \in \mathbb{R}^{256\times192}$, $W_2 \in \mathbb{R}^{128\times256}$, $W_3 \in \mathbb{R}^{d\times128}$, and σ is the sigmoid function. The output $\hat{m} \in [0,1]^d$ represents per-feature probabilities of MB membership.

Training Procedure. For each task j with n_j support samples and MB mask $m^{(j)} \in \{0,1\}^d$, we extract embeddings $E^{(j)} \in \mathbb{R}^{n_j\times192}$. Since the MB is a property of the task (not individual samples), we replicate the mask across all samples: $M^{(j)} = \mathbf{1}_{n_j} \otimes m^{(j)} \in \{0,1\}^{n_j\times d}$. This data augmentation strategy allows the model to learn the mapping from diverse sample representations to a consistent MB structure. Importantly, this approach assumes that sample embeddings encode sufficient task-level structure to support aggregation to a single MB prediction.

We train using binary cross-entropy loss:

$$\mathcal{L} = -\frac{1}{N}\sum_{j=1}^{T}\sum_{i=1}^{n_j}\sum_{k=1}^{d}\left[m_k^{(j)}\log\hat{m}_{i,k}^{(j)} + (1-m_k^{(j)})\log(1-\hat{m}_{i,k}^{(j)})\right] \tag{7}$$

where T is the number of training tasks (70 for 9-feature, 76 for 19-feature), and $N = \sum_{j=1}^{T} n_j$ is the total number of samples. We optimize using Adam with learning rate $\alpha = 0.001$, weight decay $\lambda = 10^{-5}$, batch size 128, and 100 epochs.

Inference. At test time, we extract embeddings for a new task, pass them through the appropriate MLP_d, and aggregate predictions across samples via mean pooling followed by thresholding:

$$\hat{m}_{\text{task}} = \mathbb{1}\left[\frac{1}{n}\sum_{i=1}^{n} \hat{m}_i > 0.5\right] \tag{8}$$

If no features are selected (all probabilities below threshold), we employ a fallback strategy: select the top $\min(3, d)$ features by mean probability to ensure valid predictions.

3.3 Stage 3: Filtered Regression

Given predicted MB mask $\hat{m} \in \{0, 1\}^d$, we filter both support and query sets:

$$X'_{train} = X_{train}[:, \hat{m} = 1] \tag{9}$$

$$X'_{test} = X_{test}[:, \hat{m} = 1] \tag{10}$$

We then train a TabPFN regressor on the filtered support set (X'_{train}, y_{train}) and generate predictions for the filtered query set X'_{test}. For regression, we use $N_{est} = 24$ ensemble members to obtain more stable predictions. TabPFN's in-context learning capability allows it to adapt to the filtered feature space without requiring iterative optimization.

4 Experiment

4.1 Dataset and Task Distribution

We use the CSE472-blanket-challenge/final-dataset from Hugging Face, which contains tasks generated from a hierarchical causal data-generating process. Each task $\mathcal{T} = (G, \text{SCM}, E)$ is defined by a directed acyclic graph G, a structural causal model specifying functional relationships, and an environment $E \in \{\text{IID}, \text{covariate shift}, \text{label shift}\}$.

The meta-training set (develop) contains 182 tasks with the following distribution:

- Feature dimensions: 87 tasks with 9 features, 95 tasks with 19 features
- MB sizes: Range from 2 to 15 features (varies per task)
- Environments: 61 IID, 61 label shift, 60 covariate shift
- SCM types: 91 linear, 91 nonlinear
- Sample sizes: 400 training samples, 100 test samples per task

The held-out set (submit) contains 46 tasks with ground truth MB masks and test labels withheld for final evaluation. Tasks in submit have the same distribution as develop, ensuring consistency between meta-training and meta-testing conditions.

4.2 Data Preprocessing

We perform minimal preprocessing, leveraging TabPFN's ability to handle raw numerical data. The only transformation is type conversion from Hugging Face dataset format to NumPy arrays. For submit tasks, which lack the n_features metadata field, we infer feature dimensionality from the shape of X_{train}.

4.3 Training Configuration

We split the 182 develop tasks into train/validation sets with an 80/20 ratio per feature dimension:

- 9-feature tasks: 70 train, 17 validation
- 19-feature tasks: 76 train, 19 validation

Hyperparameters:

Component	Parameter	Value
TabPFN (embedding)	n_estimators	8
	n_fold	5
MLP Predictor	hidden_dims	[256, 128]
	dropout	0.3
	learning_rate	0.001
	weight_decay	10^{-5}
	epochs	100
	batch_size	128
TabPFN (regression)	n_estimators	24

Table 1: Hyperparameter configuration for all pipeline components

4.4 Computational Resources

All experiments were conducted on ASU Sol HPC cluster using A100 GPUs (40GB VRAM) with the pytorch-gpu-2.3.1-cuda-12.1 conda environment. Total pipeline runtime was approximately 2 hours:

- Embedding extraction: 75 minutes (146 training tasks)
- MLP training: 15 minutes (both 9-feature and 19-feature models)
- Validation: 10 minutes (10 held-out tasks)
- Submission generation: 30 minutes (46 held-out tasks)

4.5 Validation Results

We evaluate on a 10-task holdout set (5 from each feature dimension) using the official metrics:

The Jaccard score of 0.7583 indicates that our predicted MB masks have substantial overlap with ground truth, capturing approximately 76% of the true structure on average. The RMSE of 0.5997 reflects regression performance using predicted (imperfect) MB masks rather than oracle features.

Metric	Value	Description
RMSE	0.5997	Regression error (lower is better)
Jaccard	0.7583	MB prediction accuracy (higher is better)
Score	0.1822	Final: $\text{avg}_i(\text{RMSE}_i \times (1 - \text{Jaccard}_i))$

Table 2: Validation performance on 10-task holdout set. Score is computed per-task then averaged, not from averaged metrics.

4.6 Comparison to Baselines

We compare our approach against relevant baselines from our lab experiments:

Method	RMSE	Notes
Oracle MB (ground truth)*	0.329	Best possible with true MB (IID only)
All Features*	0.332	Baseline: use all features (IID only)
Parents Only*	0.493	Poor: incomplete causal set (IID only)
Children Only*	0.513	Poor: incomplete causal set (IID only)
Our Method (predicted MB)	0.600	Mixed environments (IID + OOD)

Table 3: Comparison of feature selection strategies. *Baseline values from Lab 2/3 IID experiments; our method evaluated across mixed IID, covariate shift, and label shift environments. Direct comparison is approximate due to environment mismatch.

The gap between oracle MB (0.329) and our predicted MB (0.600) primarily reflects errors in MB mask prediction. With perfect MB prediction (Jaccard = 1.0), we would expect performance approaching the oracle baseline. The current Jaccard of 0.7583 suggests that approximately 24% of feature selection decisions are incorrect, leading to either missing relevant features or including spurious ones.

5 Future Works

5.1 Architectural Improvements

Unified Multi-Task Model. Our current approach trains separate MLP predictors for each feature dimension (9 and 19). A unified architecture using adaptive pooling or padding could leverage shared structure across dimensions while maintaining specialized capacity. Graph neural networks could model feature dependencies explicitly, potentially improving MB prediction for features with complex interdependencies.

Attention-Based Aggregation. We currently aggregate sample-level predictions via mean pooling. An attention mechanism could learn to weight samples based on their informativeness, potentially improving robustness to outliers or low-quality training examples. Multi-head attention across both samples and ensemble members could better capture task-level structure.

5.2 Training Enhancements

TabPFN Fine-Tuning. TabPFN 2.5 supports fine-tuning via batched training mode. Rather than using zero-shot predictions, fine-tuning on develop tasks could adapt the model to the specific

distribution of causal data-generating processes, potentially improving both embedding quality and downstream regression performance.

Uncertainty-Aware Selection. Our current threshold of 0.5 for MB membership is fixed. A learned or adaptive threshold based on prediction confidence could improve precision-recall trade-offs. Additionally, maintaining multiple candidate MB masks with associated uncertainty estimates could enable ensemble-based predictions.

5.3 Evaluation and Analysis

Per-Environment Analysis. Our validation metric averages across all environments (IID, covariate shift, label shift). Stratified evaluation would reveal whether performance degrades differentially under distribution shift, informing environment-specific adaptations.

Sample Efficiency Studies. Investigating performance as a function of support set size would characterize the method's data efficiency. TabPFN's in-context learning may enable reasonable performance with very few support examples, valuable for applications with limited data.

Ablation Studies. Systematic ablations of key design choices (n-fold embedding vs. single-pass, MLP depth, embedding dimension) would identify performance-critical components and potential simplifications.

6 References

References

[1] Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers.

[2] Tsamardinos, I., & Aliferis, C. F. (2003). Towards principled feature selection: Relevancy, filters and wrappers. *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*.

[3] Hollmann, N., Müller, S., Eggensperger, K., & Hutter, F. (2023). TabPFN: A Transformer That Solves Small Tabular Classification Problems in a Second. *arXiv preprint arXiv:2207.01848*.

[4] Paszke, A., et al. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Advances in Neural Information Processing Systems*, 32.

[5] Pedregosa, F., et al. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, 2825-2830.

[6] Wolf, T., et al. (2020). Transformers: State-of-the-Art Natural Language Processing. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*.