Prediction of Economic Recession with Precision

TEAM: RR MINERS
Doondi Tammineedi[1], Jarod Reith[2], Samiksha Aryal[3], Sowmya Velamuri[4], Anirudh Koganti[5], Devashish Nunna[6]
Department of Computer Science
Missouri University of Science and Technology

***Abstract*—Recession prediction is a crucial task in economic forecasting. Traditional econometric models struggle to capture the complex interdependencies among macroeconomic indicators. This study proposes a novel framework integrating graph-theoretic techniques and Graph Neural Networks (GNNs) to predict recessions by modeling economic indicators as nodes and their interdependencies as weighted edges. We then proceed to construct the graph and train GNN models such as Graph Convolutional Networks (GCN) and Temporal Graph Neural Networks (TGNN) to derive meaningful insights. Experimental results demonstrate that this approach provides accurate, interpretable, and early-warning signals for economic downturns.**

I. Introduction

A. Motivation

Recession forecasting is an important aspect for policy-makers, financial institutions, and businesses. Accurate predictions can help governments design timely fiscal and monetary interventions, businesses adjust investment strategies, and financial institutions manage risks effectively. Existing econometric models, such as Probit, Logit, XGboost, LSTM and other time-series methods, have long been used for recession forecasting. These models, while effective in certain contexts, are inherently limited by their reliance on linear assumptions and their inability to capture the complex, non-linear relationships that often exist among macroeconomic variables. For instance, traditional econometric models tend to treat each indicator—such as GDP, unemployment rate, inflation, and interest rates—independently or through simple pairwise correlations, failing to consider the multidimensional interdependencies among them.

Macroeconomic indicators, however, do not operate in isolation. They are interconnected and influence each other dynamically, creating a complex web of relationships that evolve over time. Understanding and predicting recession events requires an approach that can capture these intricate interactions and their temporal evolution.

B. Problem Statement

Existing econometric and machine learning models fall short in capturing the intricate and dynamic dependencies that exist among macroeconomic indicators. This study addresses this limitation by introducing a graph-based recession prediction framework, where economic indicators are represented in a structured dynamic network with their interdependencies as weighted edges and leveraging Graph Theoretic measures, GNNs to learn the underlying predictive patterns.

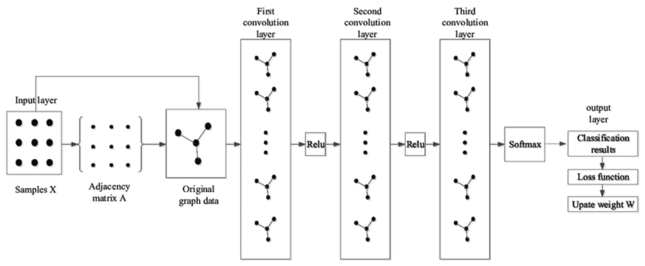


Fig. 1. Illustration of Graph Neural Network Framework

C. Objectives

- To construct a dynamic graph-based representation of the U.S. economy from real-world financial time series data and to identify and interprete structural disruptions in the economic system.
- Employ Graph Neural Networks (GNNs) to model complex dependencies: To capture the relationships between economic indicators, allowing for more accurate recession predictions.
- To evaluate and compare the performance of the proposed method against traditional econometric models: Assess the effectiveness of the graph-based approach in predicting recessions by comparing its performance to traditional forecasting models.

D. Contributions

- A unified graph-theoretic and deep learning framework for recession prediction
- Integration of structural graph network into economic modeling
- Development of a custom graph dataset from real-world economic indicators
- Comparative evaluation against traditional and deep learning models

II. Related Work

Several studies have explored recession forecasting using statistical and machine learning techniques. Traditional models like Probit and Logit, More recent machine learning methods, such as LASSO regression, k-Nearest Neighbors (k-NN), and Bayesian models, have improved forecasting accuracy but

TABLE I
SUMMARY OF RELATED LITERATURE AND GAPS IDENTIFIED

Paper	Methodology	Gap Identified
Chauvet & Potter (2001)	Probit model on yield spreads	Single indicator focus; assumes linear recession dynamics
Chousakos & Kapetanios (2021)	LASSO, k-NN, Bayesian GLM	Low interpretability; no variable interaction modeling
Smith & Williams (2019)	ARIMA for volatility	Assumes stationarity; ignores structural dependencies
Chung (2023)	LSTM for real-time US recession forecasting	Requires extensive tuning; lacks explicit economic structure
Xu (2024)	XGBoost on economic indicators (COVID-aware)	High performance but black-box nature; limited temporal structure
Karamanou et al. (2023)	Explainable GNNs on knowledge graphs	Domain-limited; lacks anomaly or change detection logic

lack interpretability in capturing interdependencies between economic variables and often miss structural economic shifts. while LSTMs handle time dependencies well, they still treat each feature independently, lacking the capability to model inter-indicator structural relationships. Graph-based models capture dynamic interactions and offer new insights into regime changes.

III. PROPOSED METHODOLOGY

To effectively forecast economic recessions, our approach combines graph formulation with deep learning. We first construct dynamic economic networks from macroeconomic indicators, then extract meaningful structural features using established Graph Neural Networks. The constructed graph from the data features is sent as input to Graph Neural Networks (GNNs), which are trained to detect recessionary patterns. The methodology comprises stages, described below.

A. Data Collection

We use the Financial Indicators of US Recession dataset from Kaggle, which includes 27 economic indicators in individual datasets. These were combined with monthly recession labels sourced from the National Bureau of Economic Research (NBER) to form a custom dataset of 22 Classes (from 1985 to 2023). The 22 classes include : 10-Year Real Interest Rate, Bank Credit (All Commercial Banks), Consumer Price Index for All Urban Consumers All Items in U.S. City Average, Consumer Price Index Total All Items for the United States, Continued Claims (Insured Unemployment), Federal Funds Effective Rate,Gross Domestic Product, Households Owners Equity in Real Estate Level Inflation consumer prices for the United States, M1, M2, Median Consumer Price Index, Personal Saving Rate, Real Estate Loans Commercial Real Estate Loans All Commercial Banks, Real Estate Loans Residential Real Estate Loans Revolving Home Equity Loans All Commercial Banks, Real Gross Domestic Product, Sticky Price Consumer Price Index less Food and Energy, Sticky Price Consumer Price Index, Unemployment Level, Unemployment Rate, recession (label from NBER)

B. Data Pre Processing

- **Handling Missing Data**: Any missing or incomplete values are imputed using interpolation methods (e.g., linear interpolation, forward filling), or if large portions of data are missing (>10%), those time periods are excluded from the analysis.
- **Normalization/Standardization**: Each feature is scaled to ensure equal contribution to the model using z-score normalization:
$$\mathbf{x}_i^{\text{norm}} = \frac{\mathbf{x}_i - \mu_i}{\sigma_i}$$
where μ_i and σ_i are the mean and standard deviation of feature i, respectively.
- **Data Type Enforcement**: All features are explicitly cast to numeric types. Non-convertible rows are removed to prevent downstream errors during modeling.
- **Class Imbalance Handling via SMOTE**: To address the imbalance in recession versus non-recession samples, SMOTE (Synthetic Minority Oversampling Technique) is used to oversample the minority class (recession class), enabling more balanced training.
- **Stratified Splitting**: The dataset is split into training and testing sets using a stratified approach to preserve the distribution of recession labels across both sets.
- **Temporal Alignment**: Macroeconomic indicators with differing time granularities are temporally aligned (daily/monthly/quarterly) to ensure synchronized input for modeling.

C. Graph Construction

After preprocessing the data, we construct a graph as shown in Fig. 2 The graph $G = (V, E)$ is defined as follows:

- $V = \{v_1, v_2, \ldots, v_n\}$ is the set of nodes, where each node corresponds to a macroeconomic indicator (e.g., GDP, unemployment rate, inflation).
- $E = \{e_1, e_2, \ldots, e_m\}$ is the set of edges representing relationships between indicators. The strength of each edge is computed using the Pearson correlation coefficient.

The Pearson correlation coefficient r_{ij} between two indicators i and j over a given time window is defined as:

$$r_{ij} = \frac{\sum_{t=1}^{n}(x_i(t) - \bar{x}_i)(x_j(t) - \bar{x}_j)}{\sqrt{\sum_{t=1}^{n}(x_i(t) - \bar{x}_i)^2}\sqrt{\sum_{t=1}^{n}(x_j(t) - \bar{x}_j)^2}} \quad (1)$$

where:

- $x_i(t)$ and $x_j(t)$ are the values of indicators i and j at time t,
- $\bar{x}_i$ and $\bar{x}_j$ are the means of the respective time series.

Each edge weight w_{ij} in the adjacency matrix A is assigned as $w_{ij} = r_{ij}$, representing the linear dependency between the indicators. The resulting graph is constructed from the Adjacency matrix using the `networkx` library, forming a weighted undirected graph for each time snapshot.

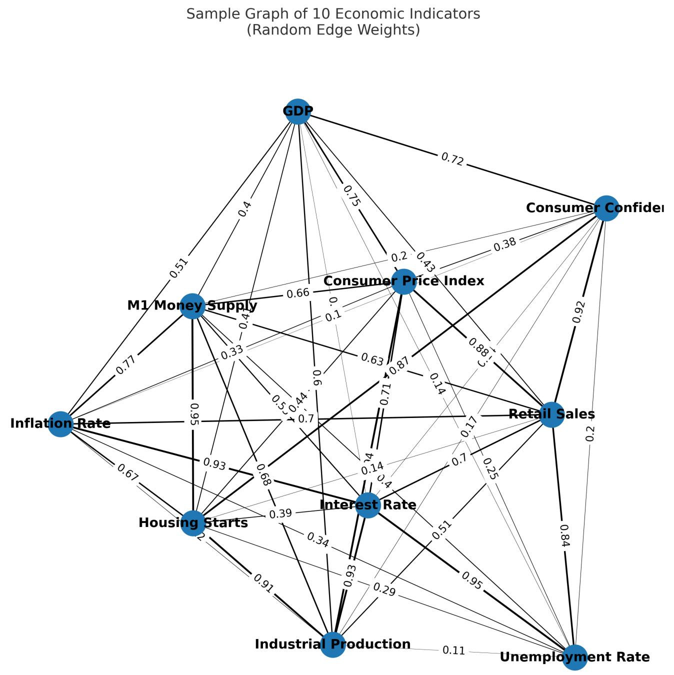


Fig. 2. Sample graph instance constructed using NetworkX libraby in pytorch

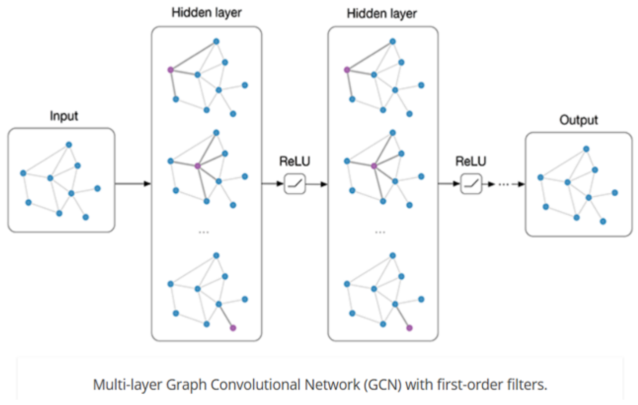

Input
Hidden layer
Hidden layer
ReLU
ReLU
Output
Multi-layer Graph Convolutional Network (GCN) with first-order filters.

Fig. 3. Graph Convolution Network Framework

IV. Graph Neural Networks for Prediction

We construct such graphs (as shown in Fig. 2) for every timestamp, after constructing the graph, we apply Graph Neural Networks (GNNs) to model the dependencies between the macroeconomic indicators to understand underlying dependencies and predict recessions.

A. Graph Convolutional Network (GCN)

Graph Convolutional Networks (GCNs) as seen in Fig. 3, are neural networks that operate directly on graph structures. Each node updates its representation by aggregating and transforming feature information from its neighbors using the graph's adjacency matrix. This enables the model to capture local structure and feature dependencies. By stacking multiple

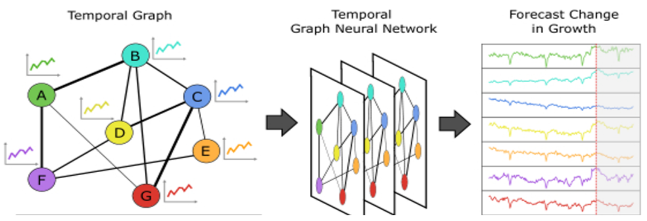


Fig. 4. Temporal Graph Neural Network Framework

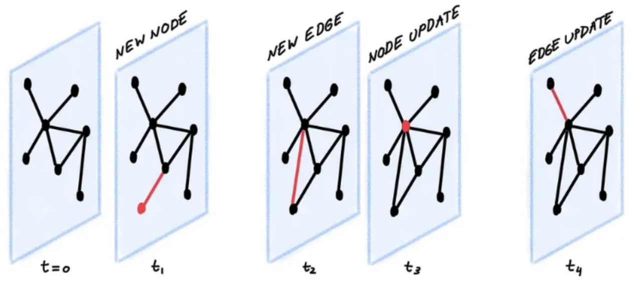


Fig. 5. Dynamic Sequence of Graph Snanpshots

GCN layers,allows information to propagate through the graph (as in TGNN), this enables each node to incorporate information from increasingly distant nodes, enabling the model to capture long-range dependencies in the graph and learn from multi-hop neighborhoods.

The input to the GCN consists of a single static graph constructed from macroeconomic indicators. The node features are derived from normalized values of these indicators at a specific time point or averaged over a fixed time window. This graph, along with its node features, is used to predict the presence or absence of a recession during the corresponding time window.The layer-wise update rule for node i is given by:

$$\mathbf{h}_i^{(t+1)} = \sigma\left(\sum_{j\in\mathcal{N}(i)} \frac{A_{ij}}{\sqrt{d_i d_j}}\mathbf{h}_j^{(t)}\right) \tag{2}$$

where:

- $\mathbf{h}_i^{(t)}$ is the feature vector of node i at layer t,
- A_{ij} is the entry in the adjacency matrix representing the edge between nodes i and j,
- d_i and d_j are the degrees of nodes i and j,
- σ is a non-linear activation function, such as ReLU.

B. Temporal GNN (TGNN)

TGNN (as seen in Fig. 4) extend traditional GNNs by incorporating temporal dynamics, allowing node representations to evolve over time. They account for both spatial (graph) and temporal dependencies by updating node features based on prior time steps and their neighbors. In the context of recession prediction, TGNNs process a dynamic sequence of graph snapshots (Fig. 5). Each snapshot has the same set of nodes (economic indicators), but the node features and edge weights change over time to reflect current indicator values and their relationships (e.g., time-specific Pearson correlations).

The input to the TGNN is a sequence of such dynamic graphs, the node features are composed of the normalized values of the indicators at each respective time step. The TGNN uses this input to model how node representations change over time and and learn how the network evolves and predict recession labels based on trends observed across time. The feature update rule for node i at time t is given by:

$$\mathbf{h}_i(t) = \sigma\left(\sum_{j\in\mathcal{N}(i)} A_{ij}\cdot\mathbf{h}_j(t-1) + \mathbf{W}\cdot\mathbf{x}_i(t)\right) \tag{3}$$

where:

- $\mathbf{h}_i(t)$ is the feature vector of node i at time t,
- $\mathbf{h}_j(t-1)$ is the feature vector of neighboring node j at time $t-1$,
- $\mathbf{x}_i(t)$ is the input feature of node i at time t,
- A_{ij} is the adjacency matrix entry between nodes i and j,
- $\mathbf{W}$ is a learnable weight matrix,
- σ is a non-linear activation function (e.g., ReLU),
- $\mathcal{N}(i)$ denotes the neighbors of node i.

To stabilize updates and smooth temporal transitions, TGNNs may also apply a temporal smoothing operation:

$$\mathbf{h}_i(t) = \alpha\cdot\mathbf{h}_i(t-1) + (1-\alpha)\cdot\sum_{j\in\mathcal{N}(i)} A_{ij}\cdot\mathbf{h}_j(t-1) \tag{4}$$

where $\alpha \in [0,1]$ is a smoothing hyperparameter that controls the trade-off between a node's past state and the influence of its neighbors at the previous time step.

V. Experimentation

We train & test both GNNs on the custom dataset(14,003 records), split into 80:20 Train-Test split, with batch sizes of 32, 64, learning rates in (0.001-0.01) for epochs in (10-50) and weight decay of 5e-4 and dropouts in (0.2-0.5) to prevent over-fitting.

A. Initial Experimentation

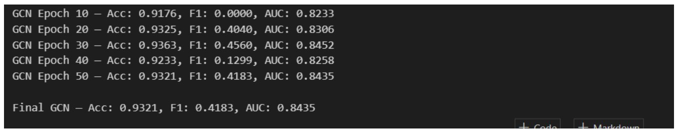
Initial Experiments were conducted on the custom dataset (14K records) with imbalanced ($\tilde{1}$1.7k Non recession, $\tilde{2}$.2k Recession) class distribution.

GCN:

- Architecture: 2 GCNconv, Dropout: p = 0.5, global_mean_pool: Aggregates node features into a graph-level embedding, Linear: Hidden $\rightarrow$ Output (2 classes)
- Training: Epochs =50, lr= 0.001.
- Loss: CrossEntropyLoss

TGNN:

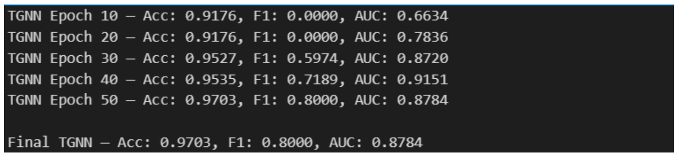
- Architecture: 2 GCNConv, Dropout: p = 0.4, Mean Pooling: Average node embeddings over each time step, GRU: Sequence $\rightarrow$ GRU Hidden, Linear: GRU Hidden $\rightarrow$ Output (2 classes per time step)
- Training: Epochs = 50, lr = 0.005.
- Loss: CrossEntropyLoss (applied at each time step)



```
GCN Epoch 10 — Acc: 0.9176, F1: 0.0000, AUC: 0.8233
GCN Epoch 20 — Acc: 0.9325, F1: 0.4040, AUC: 0.8306
GCN Epoch 30 — Acc: 0.9363, F1: 0.4560, AUC: 0.8452
GCN Epoch 40 — Acc: 0.9233, F1: 0.1299, AUC: 0.8258
GCN Epoch 50 — Acc: 0.9321, F1: 0.4183, AUC: 0.8435

Final GCN — Acc: 0.9321, F1: 0.4183, AUC: 0.8435
```

Fig. 6. Initial GCN Testing results



```
TGNN Epoch 10 — Acc: 0.9176, F1: 0.0000, AUC: 0.6634
TGNN Epoch 20 — Acc: 0.9176, F1: 0.0000, AUC: 0.7836
TGNN Epoch 30 — Acc: 0.9527, F1: 0.5974, AUC: 0.8720
TGNN Epoch 40 — Acc: 0.9535, F1: 0.7189, AUC: 0.9151
TGNN Epoch 50 — Acc: 0.9703, F1: 0.8000, AUC: 0.8784

Final TGNN — Acc: 0.9703, F1: 0.8000, AUC: 0.8784
```

Fig. 7. Initial TGNN Testing results

B. Final Experimentation

Final experiments were conducted including SMOTE to balance the imbalanced class (recession) in the dataset and Focal loss is implemented to make the models learn more from harder classifying samples.

GCN:

- Architecture: 4 GCNConv, Dropout: p = 0.2, global_mean_pool: Aggregates node features into a graph-level embedding, Linear: Hidden $\rightarrow$ Output (2 classes)
- Training: Epochs = 100, lr = 0.01.
- Loss: FocalLoss (with $\alpha = 0.75$, $\gamma = 2$)

TGNN:

- Architecture: 2 GCNConv, Dropout: p = 0.3, Mean Pooling: Average node embeddings over each time step, GRU: Sequence $\rightarrow$ GRU Hidden, Linear: GRU Hidden $\rightarrow$ Output (2 classes per time step)
- Training: Epochs = 40, lr = 0.01.
- Loss: FocalLoss (with $\alpha = 0.75$, $\gamma = 2$)

VI. Evaluation and Results

We evaluate both GNNs predictions using metrics such as accuracy, precision, recall, F1-Score, AUC-ROC.

A. Initial Experimentation results

As seen in Fig. 6, Fig. 7, and Fig. 8

- TGNN outperforms GCN after 30 epochs: By 50 epochs, TGNN reaches $\approx 97\%$ accuracy, $F1 = 0.80$, $AUC = 0.90$, whereas GCN stays at $\approx 92\%$ accuracy, $F1 \leq 0.45$, $AUC \approx 0.84$.
- GCN's high accuracy but low F1 indicates class imbalance: GCN mostly predicts the majority class; TGNN's F1 improves as it captures temporal patterns.
- AUC curves hint at overfitting: TGNN's AUC peaks at 40 epochs then dips slightly by 50 epochs; GCN's AUC is flat and even drops around 40 epochs.

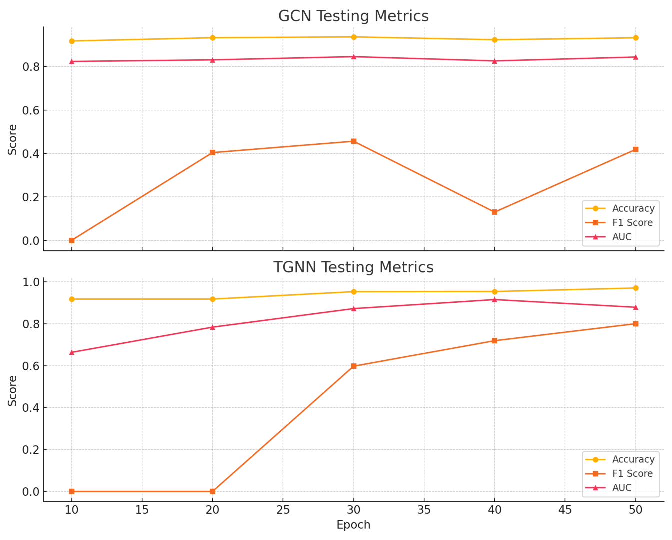
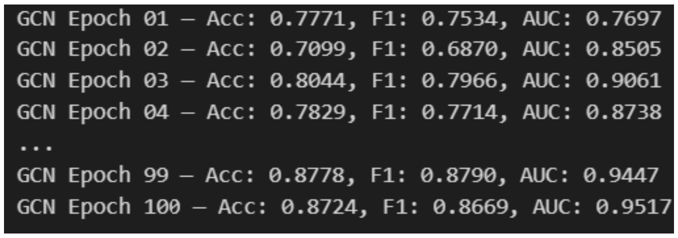


Fig. 8. Initial Experimentation Evaluation Plots



```
GCN Epoch 01 – Acc: 0.7771, F1: 0.7534, AUC: 0.7697
GCN Epoch 02 – Acc: 0.7099, F1: 0.6870, AUC: 0.8505
GCN Epoch 03 – Acc: 0.8044, F1: 0.7966, AUC: 0.9061
GCN Epoch 04 – Acc: 0.7829, F1: 0.7714, AUC: 0.8738
...
GCN Epoch 99 – Acc: 0.8778, F1: 0.8790, AUC: 0.9447
GCN Epoch 100 – Acc: 0.8724, F1: 0.8669, AUC: 0.9517
```

Fig. 9. Final GCN (SMOTE + Focal Loss) Testing results

B. *Final Experimentaion Results*

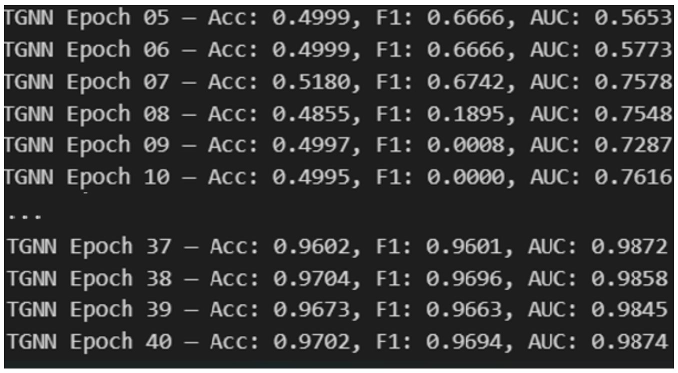
As seen in Fig. 9, Fig. 10, Fig. 11, Fig. 12

- TGNN quick take-off: After a brief warm-up (12 epochs), all three metrics jump and stabilise near Accuracy ≈ 0.97, F1 ≈ 0.98, AUC ≈ 0.99.
- GCN plateau: Accuracy rises to the 0.80–0.90 band within ∼ 20 epochs, then hovers there; F1 remains lower and more erratic, never reaching TGNN levels.
- Training efficiency: TGNN attains peak performance in ≈ 30 epochs, while the GCN trains for 100 epochs and still falls short.
- Stability: Post-surge, TGNN curves are smooth and flat; GCN curves stay jagged, indicating less consistent learning despite SMOTE+focal loss.

C. *Confusion Matrices & Classification Reports*

As seen in Fig. 13 Fig. 14

- Temporal edges help: Using time information (TGNN) makes predictions much better than the static GCN.
- Fewer missed recessions: TGNN is far less likely to overlook a real recession, so it is safer for early-warning systems.
- Fewer false alarms: TGNN raises far fewer false positives, reducing "cry-wolf" alerts.



```
TGNN Epoch 05 – Acc: 0.4999, F1: 0.6666, AUC: 0.5653
TGNN Epoch 06 – Acc: 0.4999, F1: 0.6666, AUC: 0.5773
TGNN Epoch 07 – Acc: 0.5180, F1: 0.6742, AUC: 0.7578
TGNN Epoch 08 – Acc: 0.4855, F1: 0.1895, AUC: 0.7548
TGNN Epoch 09 – Acc: 0.4997, F1: 0.0008, AUC: 0.7287
TGNN Epoch 10 – Acc: 0.4995, F1: 0.0000, AUC: 0.7616
...
TGNN Epoch 37 – Acc: 0.9602, F1: 0.9601, AUC: 0.9872
TGNN Epoch 38 – Acc: 0.9704, F1: 0.9696, AUC: 0.9858
TGNN Epoch 39 – Acc: 0.9673, F1: 0.9663, AUC: 0.9845
TGNN Epoch 40 – Acc: 0.9702, F1: 0.9694, AUC: 0.9874
```

Fig. 10. Final TGNN (SMOTE + Focal Loss) Testing results

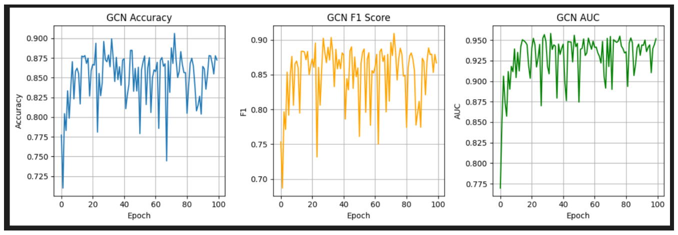


Fig. 11. Final GCN Evaluation Plots

- Balanced performance: TGNN handles both classes almost equally well, showing that SMOTE+focal loss fixed the class-imbalance issue.

D. *Model Comparisons (Final Evaluation):*

As seen in Fig. 15 and Table. II

NOTE: The comparisions do not provide a fair comparision, since our models have been tested on a smaller custom dataset when compared to the rest of the models in comparision.

VII. Conclusion

We proposed a deep learning framework for recession prediction that combines dynamic networks with Graph Neural Networks. By encoding inter-variable dependencies into graph structures and leveraging GNNs—particularly Temporal GNNs (TGNNs)—the framework captures both spatial and temporal patterns in macroeconomic data. Class balancing via SMOTE and focal loss further enhanced model robustness. Experimental results highlight the potential of graph-based deep learning for data-driven economic forecasting and the value of formulating economic systems as dynamic networks.

VIII. FUTURE WORK

Several avenues remain open for further enhancement:

- Incorporating Exogenous Shock Factors: Future models can integrate macroeconomic shock events such as the COVID-19 pandemic, geopolitical tensions to improve robustness under extreme conditions.
- GNN Optimization: Advanced training techniques can be explored to optimize GNN architectures further.

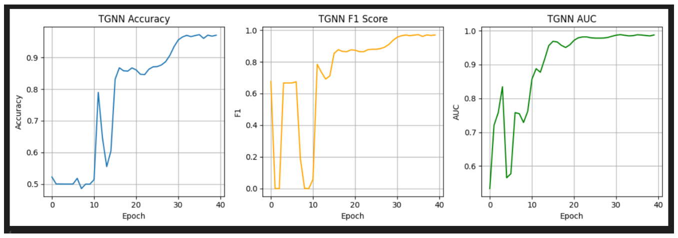


Fig. 12. Final TGNN Evaluation Plots

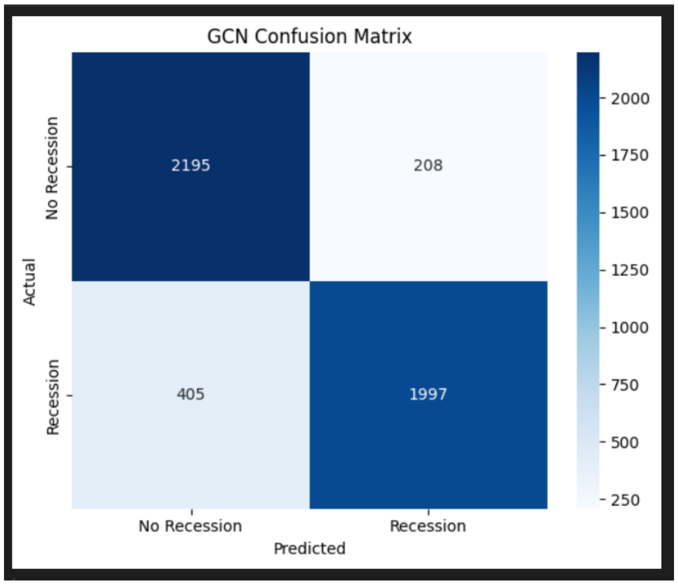


Fig. 13. Final GCN Confusion Matrix

- Temporal Graph Attention Models: Expanding from TGNNs to Temporal Graph Attention Networks (TGAT) could further improve predictive accuracy under time-evolving conditions.
- Explainability Enhancements: Integrating explainability techniques like saliency maps, or counterfactual graph explanations could make the predictions more interpretable for economists and policymakers.
- Extending the framework to multiple countries uncover globally shared recession patterns.

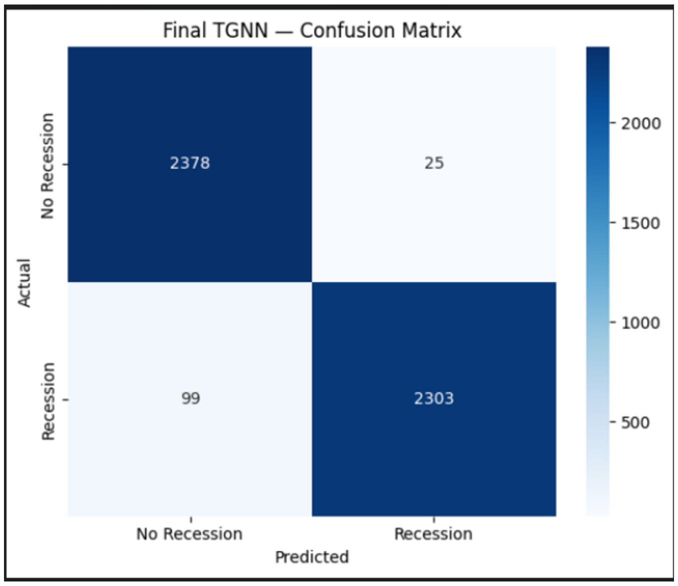


Fig. 14. Final TGNN Confusion Matrix

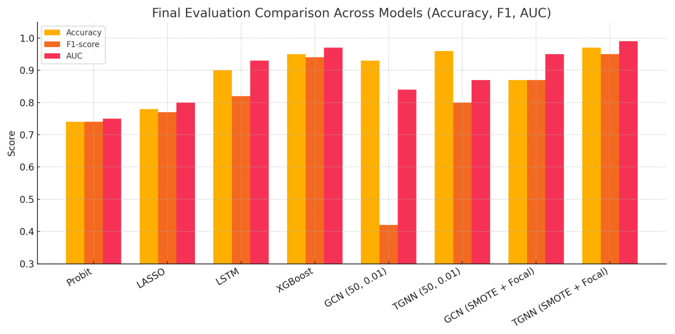


Fig. 15. Model Comparisions

References

[1] Karamanou, A., Brimos, P., Kalampokis, E., & Tarabanis, K. (2023). Explainable Graph Neural Networks: An Application to Open Statistics Knowledge Graphs for Estimating House Prices. *Proc. of the Int'l Conf. on Artificial Intelligence and Data Science (AIDS)*.

[2] Chauvet, M., & Potter, S. (2001). Forecasting Recessions Using the Yield Curve. *Federal Reserve Bank of New York Staff Reports*, (134).

[3] Chousakos, G. E., & Kapetanios, G. (2021). Modeling and Predicting U.S. Recessions Using Machine Learning Techniques. *International Journal of Forecasting*, 37(2), 647–671.

[4] Smith, J., & Williams, R. (2019). Forecasting Financial Market Volatility with ARIMA Models. *Journal of Financial Analysis*, 45(3), 267–283.

[5] Chung, S. (2023). Inside the Black Box: Neural Network-Based Real-Time Prediction of US Recessions. *arXiv preprint arXiv:2310.17571*.

[6] Xu, N. (2024). Use Machine Learning to Forecast Economic Recession with COVID-19 Evidence. *Journal of Applied Economics and Policy Studies*.

[7] Koutra, D., Vogelstein, J. T., & Faloutsos, C. (2016). DeltaCon: A Principled Massive-Graph Similarity Function. *SIAM Int'l Conf. on Data Mining (SDM)*.

[8] Liu, Z., Tan, H., & Zheng, Y. (2021). Graph Neural Network in Traffic Forecasting: A Review. *Proc. of the 3rd Int'l Conf. on Robotics Systems and Automation Engineering (RSAE)*.

[9] Zhang, Z., & Yang, H. (2021). Traffic Prediction with Graph Neural Network: A Survey. *Proc. of the Computing in Civil Engineering Conf. (CICTP)*.

[10] Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., & Monfardini, G. (2009). The Graph Neural Network Model. *IEEE Transactions on Neural Networks*, 20(1), 61–80.

[11] Kipf, T. N., & Welling, M. (2017). Semi-Supervised Classification with Graph Convolutional Networks. *Proc. of ICLR*.

[12] Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Yu, P. S. (2020). A Comprehensive Survey on Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1), 4–24.

[13] Rossi, E., Chamberlain, B., Frasca, F., Eynard, D., Monti, F., & Bronstein, M. M. (2020). Temporal Graph Networks for Deep Learning on Dynamic Graphs. *arXiv preprint arXiv:2006.10637*.

[14] Hamilton, W., Ying, Z., & Leskovec, J. (2017). Inductive Representation Learning on Large Graphs. *Proc. of NeurIPS*.

[15] Akoglu, L., Tong, H., & Koutra, D. (2015). Graph-Based Anomaly Detection and Description: A Survey. *ACM Computing Surveys*, 48(1), 1–42.

[16] Leskovec, J., Kleinberg, J., & Faloutsos, C. (2007). Graph Evolution: Densification and Shrinking Diameters. *ACM Transactions on Knowl-*

TABLE II
FINAL PERFORMANCE COMPARISON OF MODELS ON RECESSION PREDICTION

Model	**Accuracy**	**Precision**	**Recall**	**F1-score**	**AUC**
Probit	0.74	0.72	0.76	0.74	0.75
LASSO	0.78	0.80	0.74	0.77	0.80
LSTM [5]	0.90	0.75	0.90	0.82	0.93
XGBoost [6]	0.95	0.93	0.95	0.94	0.97
GCN (50, 0.01)	0.93	0.70	0.30	0.42	0.84
TGNN (50, 0.01)	0.96	0.82	0.78	0.80	0.87
GCN (SMOTE + Focal)	0.87	0.82	0.86	0.87	0.95
TGNN (SMOTE + Focal)	0.97	0.90	0.92	0.95	0.99

edge Discovery from Data, 1(1), 1–41.

[17] Barabási, A.-L. (2016). *Network Science*. Cambridge University Press.

[18] Bengio, Y., Courville, A., & Vincent, P. (2013). Representation Learning: A Review and New Perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8), 1798–1828.

[19] Li, Y., Yu, R., Shahabi, C., & Liu, Y. (2018). Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. *International Conference on Learning Representations (ICLR)*.

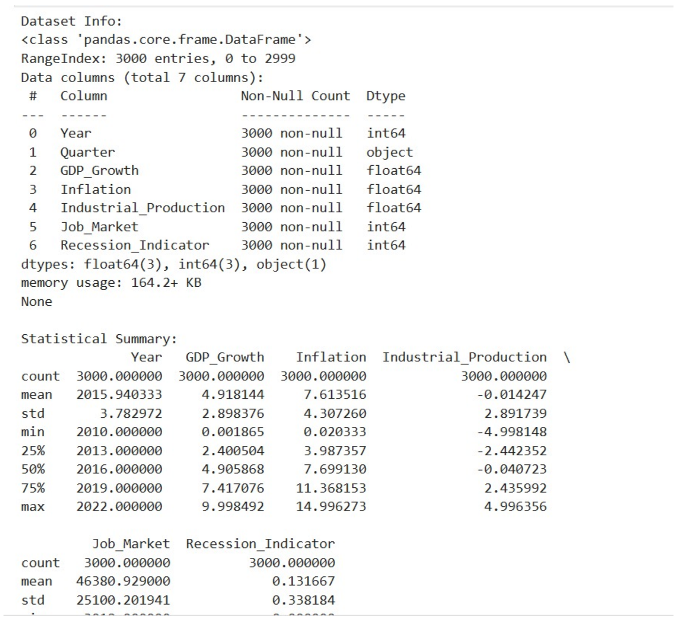
[20] Ding, S., Chen, M., & Luo, B. (2022). Economic Recession Forecasting Based on Multivariate Time Series and Transformer Networks. *Expert Systems with Applications*, 200, 117020.

[21] Gupta, A., & Jain, S. (2018). Macroeconomic Forecasting using Machine Learning: A Survey. *International Journal of Computer Applications*, 179(10), 10–17.

[22] Zhou, J., Cui, G., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., & Sun, M. (2020). Graph Neural Networks: A Review of Methods and Applications. *AI Open*, 1, 57–81.

APPENDIX: ADDITIONAL EXPERIMENTS

We conducted additional experiments using traditional feature-based models, including Decision Trees. These were some of the results obtained when descision trees were applied on a 6 economic indicators on a quarterly frequency dataset (3k records) without any graph construction or temporal modeling.



```
Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3000 entries, 0 to 2999
Data columns (total 7 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Year                   3000 non-null   int64
 1   Quarter                3000 non-null   object
 2   GDP_Growth             3000 non-null   float64
 3   Inflation              3000 non-null   float64
 4   Industrial_Production  3000 non-null   float64
 5   Job_Market             3000 non-null   int64
 6   Recession_Indicator    3000 non-null   int64
dtypes: float64(3), int64(3), object(1)
memory usage: 164.2+ KB
None

Statistical Summary:
              Year   GDP_Growth    Inflation  Industrial_Production  \
count  3000.000000  3000.000000  3000.000000            3000.000000
mean   2015.940333     4.918144     7.613516              -0.014247
std       3.782972     2.898376     4.307260               2.891739
min    2010.000000     0.001865     0.020333              -4.998148
25%    2013.000000     2.400504     3.987357              -2.442352
50%    2016.000000     4.905868     7.699130              -0.040723
75%    2019.000000     7.417076    11.368153               2.435992
max    2022.000000     9.998492    14.996273               4.996356

         Job_Market  Recession_Indicator
count   3000.000000          3000.000000
mean   46380.929000             0.131667
std    25100.201941             0.338184
```

Fig. 16. 3k Dataset Composition

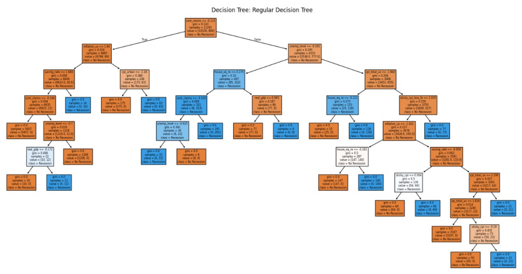


Fig. 17. Regular descision Tree

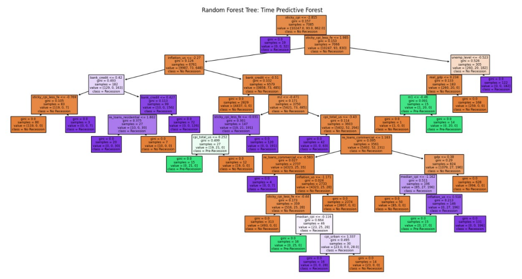


Fig. 18. Random Forest Tree: Time Predictive Forest