Supply Chain Risk Modeling Competition

🎯 Project Overview

A comprehensive machine learning pipeline for modeling supply chain risks using **Heterogeneous Graph Neural Networks (GNN) + Gradient Boosting** hybrid architecture.

Key Features

- **GNN Embeddings**: Captures network dependencies between suppliers, buyers, and products
- **Hybrid Fusion**: Combines graph-level and tabular-level features
- **Multi-Task Learning**: Classification + Regression for risk and resilience prediction
- **Model Comparison**: LightGBM vs CatBoost benchmark
- **Production-Ready**: Modular, class-based design with comprehensive auditing

---

📁 Project Structure

```
supply-chain-risk/
├── data/
│   ├── supply_chain_risk.csv        # Raw risk dataset
│   ├── supply_chain_resilience.csv  # Raw resilience dataset
│   └── cleaned/                     # Preprocessed datasets (auto-generated)
├── output/                          # Visualizations & plots
├── reports/                         # Model metrics & audits
├── main.py                          # Complete pipeline
├── requirements.txt                 # Dependencies
└── README.md                        # This file
```

🚀 Quick Start

1. Installation

bash

```bash
# Create virtual environment
python -m venv venv
source venv/bin/activate  # On Windows: venv\Scripts\activate

# Install dependencies
pip install -r requirements.txt
```

2. Prepare Data

Place your datasets in the `data/` folder:

- `supply_chain_risk.csv`
- `supply_chain_resilience.csv`

3. Run Pipeline

bash

```bash
python main.py
```

4. Customize Configuration

Edit paths and hyperparameters in `main.py`:

python

```python
# Modify paths
paths = PathHandler(
    data="your_data_folder",
    cleaned="your_cleaned_folder",
    output="your_output_folder",
    reports="your_reports_folder"
)

# Modify hyperparameters
params = HyperParameters(
    test_size=0.2,
    gnn_hidden_channels=128,  # Increase for larger graphs
    gnn_epochs=200,
    # ... more parameters
)
```

📊 Pipeline Stages

Phase 1: Data Analysis & Preprocessing

1. **Load datasets** from CSV files
2. **Handle missing values** using median/mode imputation
3. **Feature engineering**:
   - Delivery delay calculations
   - Temperature/vibration risk zones
   - Resilience & risk scores
4. **Outlier handling** via IQR clipping
5. **Categorical encoding** using LabelEncoder
6. **EDA visualizations**:
   - Distribution plots
   - Correlation heatmaps
   - VIF audit (multicollinearity)
   - Data leakage audit

Phase 2: Graph Construction

1. **Heterogeneous graph** with 3 node types:
   - Suppliers
   - Buyers
   - Products
2. **Edge types**:
   - Supplier → Buyer (supplies)
   - Buyer → Product (orders)
3. **Node features** aggregated from transactions

Phase 3: GNN Training

1. **GraphSAGE** layers for message passing
2. **Self-supervised** reconstruction loss
3. **Extract embeddings** (64-dim vectors per node)
4. **Visualizations**: t-SNE & PCA plots

Phase 4: Hybrid Fusion

1. **Concatenate** GNN embeddings with tabular features
2. **Creates enriched feature space** combining:
   - Structural patterns (from graph)
   - Attribute patterns (from features)

Phase 5: Model Training & Evaluation

Task 1: Risk Label Classification

- **Target**: `manual_risk_label` or `resilience_label` (Low/Medium/High)
- **Models**: LightGBM Classifier, CatBoost Classifier
- **Metrics**: Accuracy, F1-score (macro/weighted), AUC
- **Outputs**: Confusion matrix, classification report

Task 2: Risk Score Regression

- **Target**: `risk_score` (continuous)
- **Models**: LightGBM Regressor, CatBoost Regressor

- **Metrics**: MAE, RMSE, R^2
- **Outputs**: Prediction vs Actual plot, Residual plot

Task 3: Resilience Score Regression

- **Target**: `resilience_score` (continuous)
- **Models**: LightGBM Regressor, CatBoost Regressor
- **Metrics**: MAE, RMSE, R^2
- **Outputs**: Prediction vs Actual plot, Residual plot

📈 Expected Outputs

`/reports/` Folder

- `model_comparison_summary.csv` - All models' performance metrics
- `*_confusion_matrix.png` - Confusion matrices for classification
- `*_classification_report.csv` - Detailed per-class metrics
- `*_predictions.png` - Regression scatter plots
- `*_residuals.png` - Residual analysis
- `*_feature_importance.csv` - Top features by importance
- `*_vif_audit.csv` - Variance Inflation Factors
- `*_leakage_audit.csv` - Potential data leakage flags
- `*_summary_stats.csv` - Descriptive statistics

`/output/` Folder

- `*_distributions.png` - Feature distributions
- `*_correlation.png` - Correlation heatmaps
- `gnn_embeddings_tsne.png` - t-SNE visualization
- `gnn_embeddings_pca.png` - PCA visualization
- `*_feature_importance.png` - Feature importance bar charts

🏆 Why This Approach Works

1. Novel Architecture

- First-of-its-kind hybrid GNN + Gradient Boosting for supply chain
- Captures both **network effects** (who affects whom) and **attribute patterns** (what matters)

2. Graph Advantage

- Traditional ML treats suppliers independently
- GNNs learn: "If Supplier A fails, Buyers X, Y, Z are at risk"
- Propagates disruption signals through the network

3. Gradient Boosting Strength

- LightGBM: Fast, handles categorical features natively
- CatBoost: Best for mixed data types, automatic handling of missing values
- Both excel at learning complex, non-linear patterns

4. Production-Ready Code

- Modular class-based design
- Centralized configuration (PathHandler, HyperParameters)
- Comprehensive logging and error handling
- Easy to debug and extend

5. Comprehensive Auditing

- VIF check for multicollinearity
- Data leakage detection
- Model comparison across multiple metrics
- Feature importance analysis

🔧 Troubleshooting

CUDA Out of Memory

python

```python
# In HyperParameters
gnn_hidden_channels = 32  # Reduce from 64
gnn_num_layers = 1        # Reduce from 2
```

Graph Too Large

python

```python
# Sample your data before graph construction
resilience_sample = resilience_clean.sample(frac=0.5, random_state=42)
```

Missing Columns

- Ensure your CSV files match the expected column names
- Check `DataPreprocessor.clean_*_data()` methods for required columns

📚 Key Concepts

Heterogeneous Graphs

Unlike homogeneous graphs (single node type), heterogeneous graphs have:

- Multiple node types (Supplier, Buyer, Product)
- Multiple edge types (supplies, orders)

- Different feature dimensions per node type

GNN Message Passing

1. Each node aggregates information from its neighbors
2. Updates its embedding based on neighborhood structure
3. After multiple layers, captures multi-hop dependencies

Feature Fusion

```
[Tabular Features] + [GNN Embeddings] → [Rich Feature Vector]
       ↓                  ↓                  ↓
   Attributes         Topology         Combined Power
```

🎓 Competition Tips

Phase 1 Deliverables

- Provide the EDA visualizations from `/output/`
- Include VIF and leakage audit reports from `/reports/`
- Justify feature engineering in your presentation

Phase 2 Deliverables

- Compare LightGBM vs CatBoost using `model_comparison_summary.csv`
- Show feature importance plots to explain model decisions
- Justify GNN approach: "Captures network dependencies traditional ML misses"

Presentation Strategy

1. **Problem**: Supply chains are interconnected networks, not isolated data points
2. **Solution**: GNN captures topology + Gradient Boosting captures attributes
3. **Results**: Show performance gains from hybrid approach
4. **Novelty**: "First competition solution using heterogeneous GNNs"

📞 Support

For questions or issues:

1. Check this README thoroughly
2. Review inline code documentation
3. Examine error logs in console output

🙏 Acknowledgments

- **PyTorch Geometric** for GNN infrastructure

- **LightGBM/CatBoost** teams for excellent gradient boosting libraries
- **Competition organizers** for the challenging problem statement

---

Good luck with your competition! 🚀