ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

MSc Data Science AUEB
Data Science Challenge
PRODUCT CLASSIFICATION CHALLENGE

Professor: Ioannis Nikolentzos

Team: Model Minds 🧠
Giagkos Stylianos | f3352410
Kazantzis Gerasimos| f3352406
Vougioukos Dimitris | f3352411

Introduction

- E-commerce growth has made automated product classification essential for search, recommendations, and catalogue management.
- This project tackles a real-world machine learning challenge based on Amazon sports product data, hosted on the Kaggle platform.
- The task: Predict one of 16 product categories using multimodal data (text, price, and graph structure).
- Evaluation metric: Multi-class log loss, with the lowest score ranking highest on the leaderboard.

Problem Description and Dataset Analysis

2.1 Problem Statement

- Goal: Classify Amazon sports products into 16 categories
- Challenge: Combine graph and text features for model building

2.2 Dataset Overview

- **edgelist.txt**: Co-view graph (276K nodes, 1.8M edges)
- **descriptions.txt**: Titles & descriptions of products
- **price.txt**: Price info for ~199K products
- **y_train.txt**: Labels for 182K training samples
- **test.txt**: 45K products to classify (no labels provided)

Problem Description and Dataset Analysis

2.3 Data Preparation

- **Parsing & Conversion:** Raw text files → DataFrames
- **Price & Descriptions:** Saved to .xlsx files
- **ID Formatting:** Ensures consistent integer type across all datasets

Product Description Cleaning

- HTML decoding: Converts HTML entities
- HTML tag removal: Cleans formatting noise
- Non-printable character filtering
- Whitespace normalization & trimming

Final Dataset Assembly

- Merged descriptions with labels/test IDs
- Generated training/test datasets
- Created undirected graph from edgelist

Data Characteristics

- The dataset includes 16 sports product categories, with a **high class imbalance** (1,129 to 43,260 samples per class).
- This imbalance can cause models to favour majority classes, requiring techniques like class weighting or undersampling.
- Data sources are multimodal, including:
  - **Text**: Product titles and descriptions
  - **Price**: Useful but incomplete feature
  - **Graph**: Co-viewing relationships between products
- Price distribution is heavily right-skewed, as shown in Figure 1:
  - Most products are priced between **$0–$100**
  - Very few exceed **$400**
- **Mean price**: $55.30 **Median price**: $24.99
  - The large gap indicates the presence of high-priced outliers
- Suggests a catalogue dominated by affordable items, with a smaller number of premium-priced products

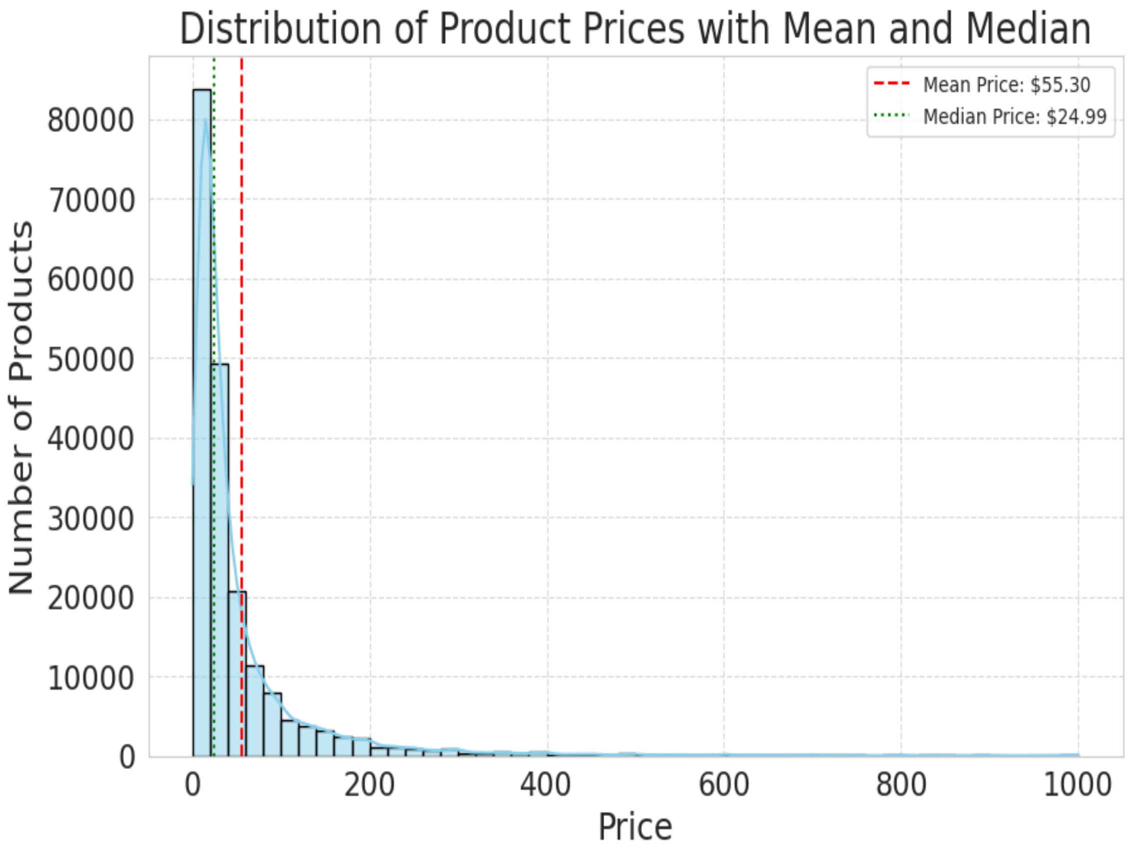

Figure 1 Distribution of Product Prices

Data Characteristics

- **Outliers Abundant**:
  - All categories show many high-priced outliers, some approaching $1000, indicating a heavy-tailed distribution.
- **Median Price Varies**:
  - Most medians fall between $20–$40, though:
  - Label 7 has a much lower median (~$10)
  - Labels 9 & 13 are slightly higher than average
- **IQR Differences (Price Spread)**:
  Some labels (e.g., 1, 3, 14) have a wider IQR, meaning greater price variability.
  - Label 7 again stands out with a narrower IQR → less diversity in pricing.
- **Minimum Prices**:
  - 11 of 16 labels include products priced at **$0.01** (visible near 10^{-2})

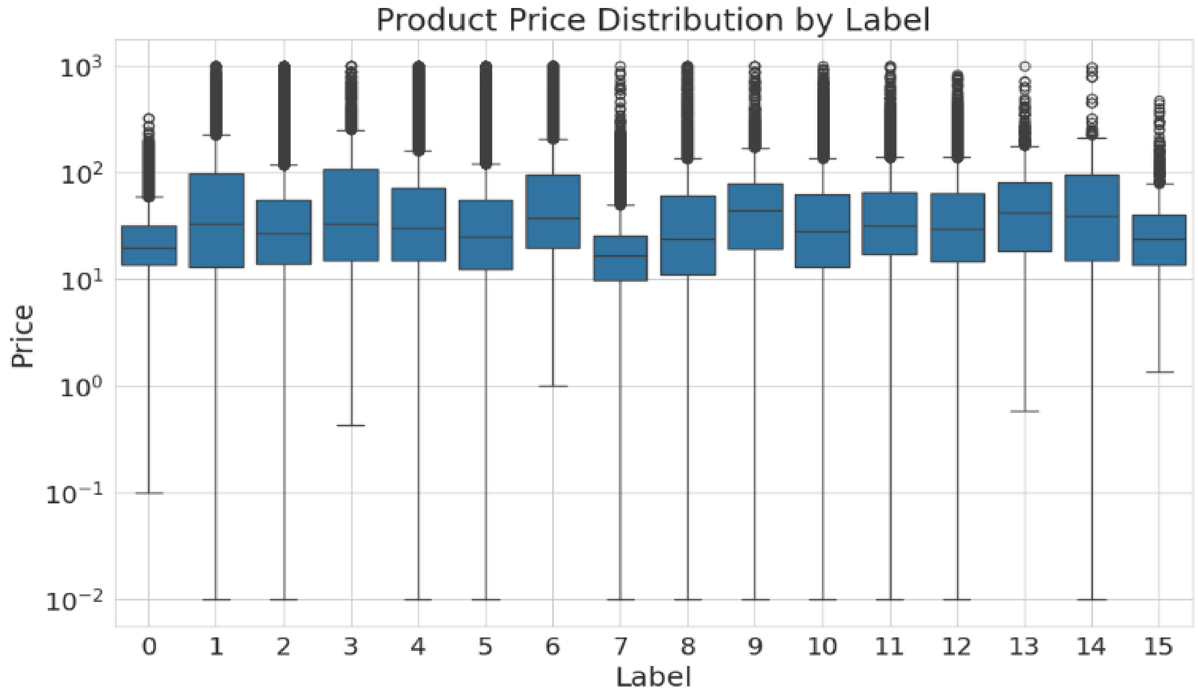

Figure 2 Product Price Distribution by Label

Data Characteristics

- The plot shows a clear **decreasing trend**: as node degree increases, node frequency rapidly drops.
- The distribution follows a **heavy-tailed pattern**, suggesting it may resemble a truncated power-law or log-normal rather than a pure power-law.
- **Low-degree nodes (1–6)** are very common, typical of real-world networks.
- **High-degree nodes (hubs)** are rare but present — visible near degree 1000.
- The network's structure supports **information diffusion**, but centrality is concentrated in a few key nodes.

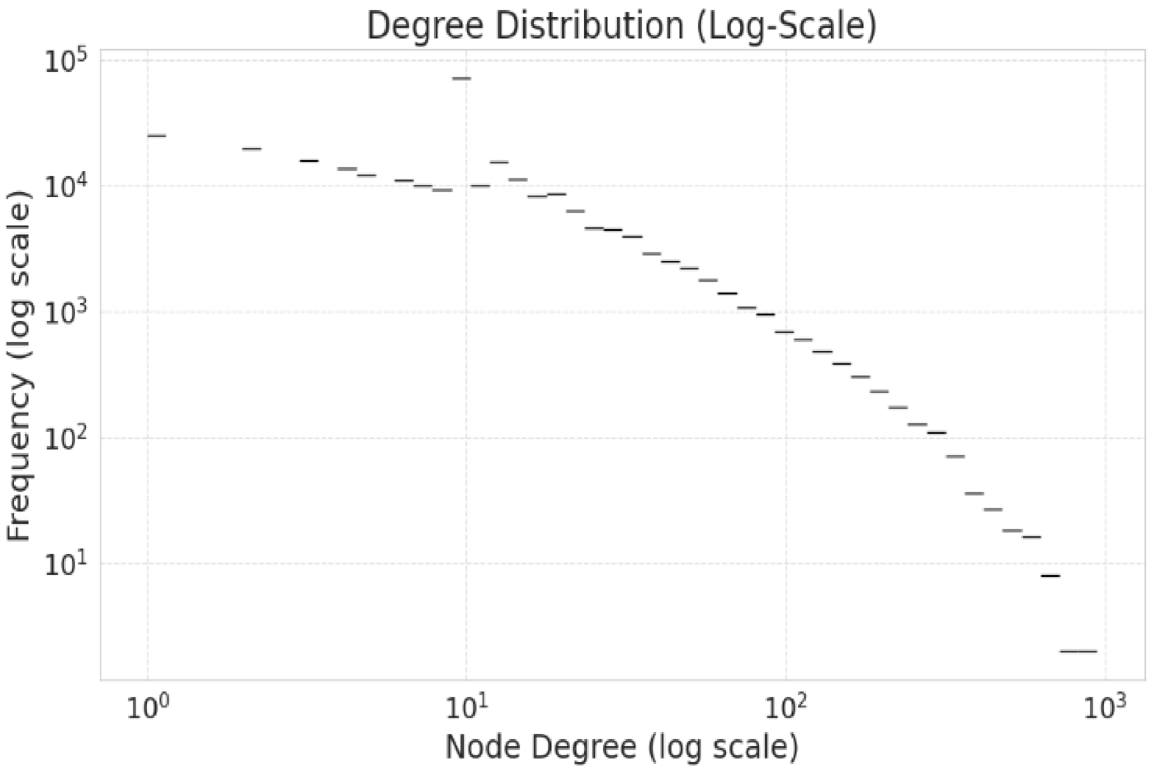

Figure 3 Degree Distribution (Log-Scale)

Evaluation Metric: LogLoss

- Measures confidence-weighted correctness of predictions
- **Sensitive to overconfident wrong predictions**
- Lower LogLoss = better calibrated model

$$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij}\log(p_{ij})$$

N: number of products
C=16: number of classes
yij : true label indicator
pij : predicted probability

Models Methodology

3.1 BERT + LoRA Architecture

- **Goal**: Extract text embeddings from product descriptions
- **Model**: DistilBERT
- **Input**: Tokenized text, max length 128 tokens

Training Strategy

- **Custom WeightedTrainer** for handling class imbalance
- **Loss function**: Class-weighted cross-entropy
- **Trainer API**: Early stopping, epoch-based evaluation
- **Metrics**: Accuracy, Precision, Recall, F1-score

LoRA for Efficient Fine-Tuning

- **LoRA**: Injects low-rank trainable matrices into attention layers
- **Adapted layers**: Query & Value projections (q_lin, v_lin)
- **Config**: Rank = 8, α = 32, dropout = 0.05
- **Advantage**: Fewer trainable parameters, faster training

Embedding Extraction

- Discarded classifier head post-training
- Used [CLS] token from final hidden layer
- Generated **768-dimensional embeddings** per product
- Saved for integration with graph-based models

Models Methodology

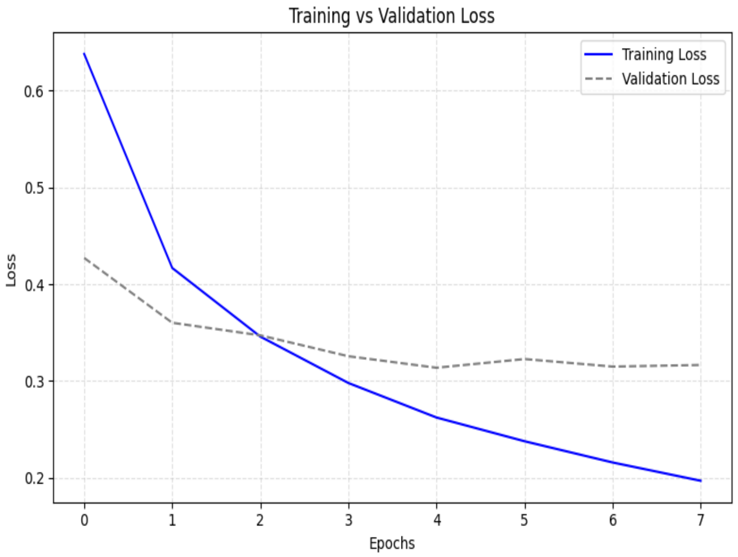

Figure 4 Training and Validation Losses for Finetuned BERT

Class	Precision	Recall	F1-score	Support
0	0.9636	0.969	0.9663	3033
1	0.8564	0.8925	0.8741	2372
2	0.9414	0.8484	0.8925	8652
3	0.925	0.9655	0.9448	1073
4	0.8868	0.9347	0.9101	3016
5	0.9323	0.9313	0.9318	3565
6	0.9007	0.9375	0.9187	1519
7	0.9562	0.9198	0.9376	3752
8	0.9263	0.946	0.9361	1316
9	0.9181	0.9181	0.9181	903
10	0.8081	0.8426	0.825	3589
11	0.8929	0.9011	0.897	1425
12	0.6957	0.8794	0.7768	1318
13	0.7955	0.8793	0.8353	323
14	0.9008	0.9646	0.9316	226
15	0.8739	0.9313	0.9017	320
Accuracy			0.9007	36402
Macro Avg	0.8859	0.9163	0.8998	36402
Weighted Avg	0.9044	0.9007	0.9014	36402

Table 1 Finetuned BERT Classification Report

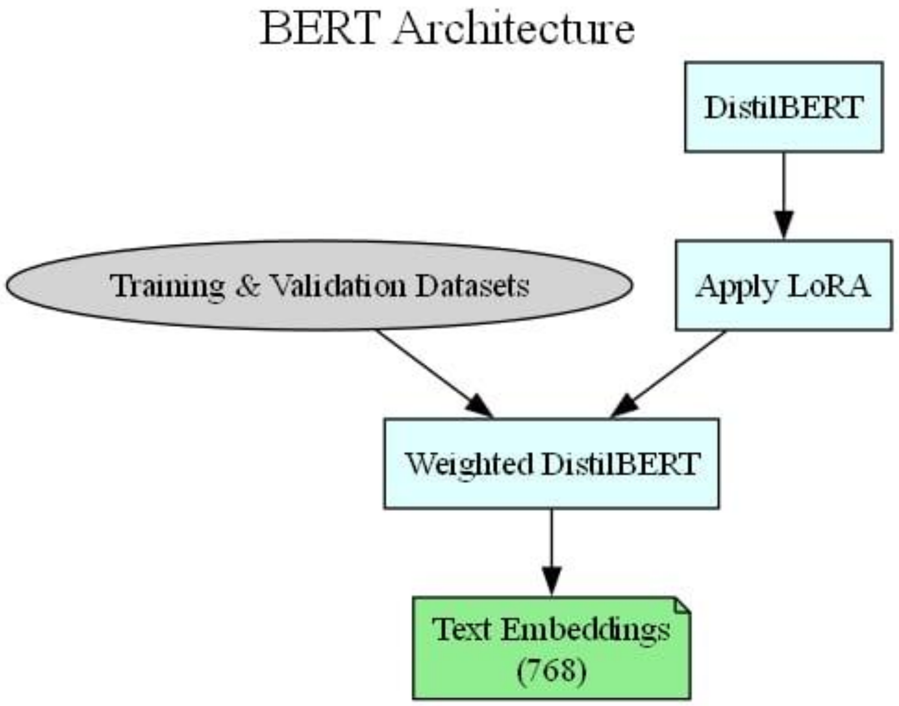

Figure 5 BERT Architecture

Key Outcomes

- **Strong text-only baseline**
- Captured deep semantic features from descriptions
- LoRA enabled **efficient domain-specific fine-tuning**
- No graph or price info used — yet achieved solid performance

Models Methodology

3.2 GraphSAGE

Input Features

- **Text Embeddings (768-dim)**
- Extracted from fine-tuned DistilBERT using the [CLS] token.
- **Normalized Price (1-dim)**
- Rescaled numerical price feature per product.
- **Combined into Product Feature Vector (769-dim)**

Graph Structure

- **Edges**: Represent co-view relationships between products.
- Enables message passing between connected nodes.

Stratified 5-Fold Training

- **Split:** 80% Training, 20% Validation
- **Test Dataset:** Average Class Probabilities

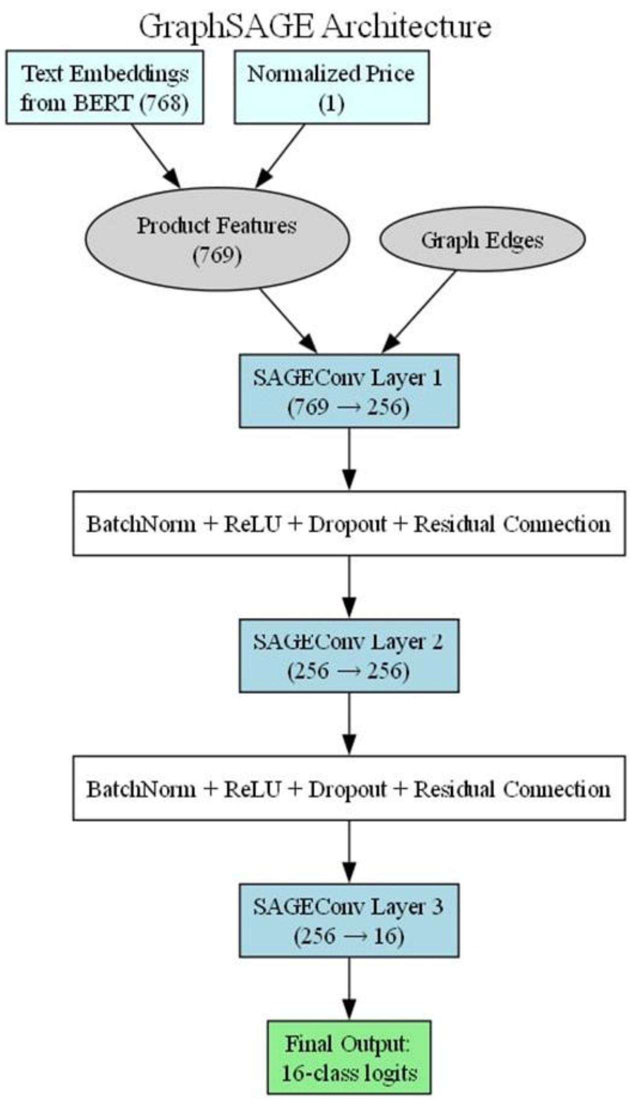

Figure 8 SAGE Architecture

Models Methodology

3.2 GraphSAGE Architecture

Model Layers

- SAGEConv Layer 1 (769 → 256)
  Neighborhood aggregation using learned weighted mean.
  Output passed through:
  - BatchNorm
  - ReLU Activation
  - Dropout
  - Residual Connection
- SAGEConv Layer 2 (256 → 256)
  Same post-processing stack as Layer 1.
- SAGEConv Layer 3 (256 → 16)
  Final transformation to raw **16-class logits** (no activation).

Output

- **Final Output**: Logits over 16 categories.
  Used for multi-class classification with CrossEntropy Loss.

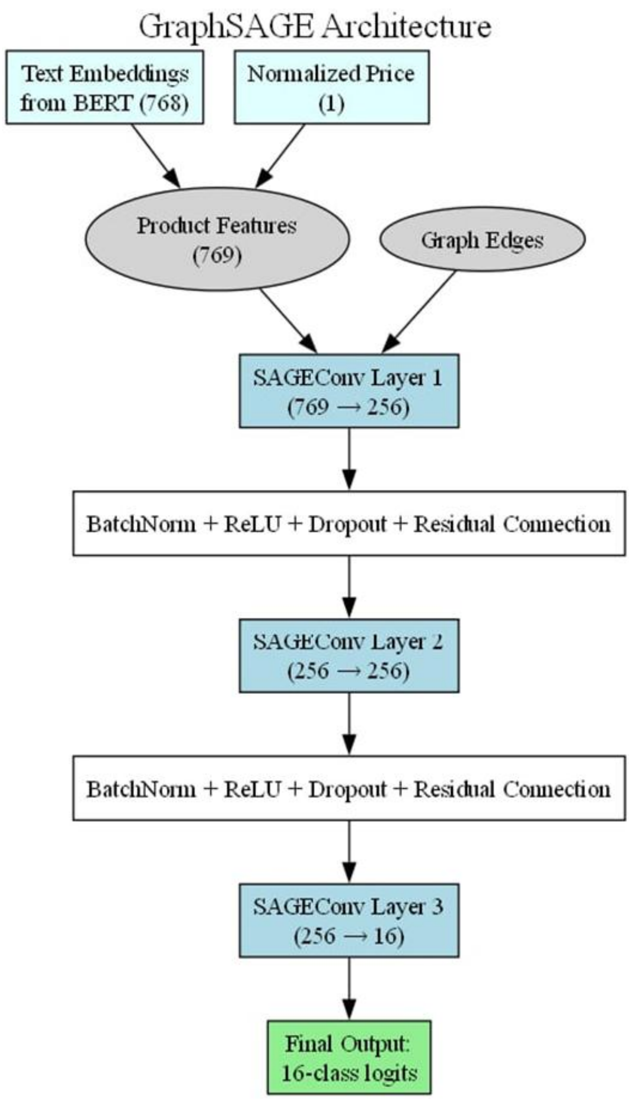

Figure 8 SAGE Architecture

Models Methodology

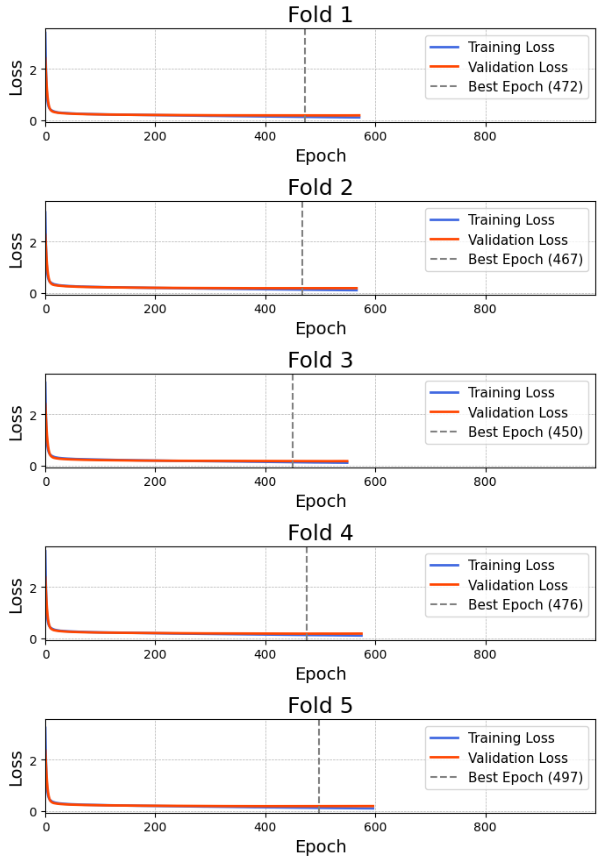

Figure 9 Train and Validation Losses for 5 Folds

Class	**Precision**	**Recall**	**F1-score**	**Support**
0	0.9813	0.9862	0.9837	3033
1	0.9201	0.9216	0.9208	2373
2	0.9274	0.9542	0.9406	8652
3	0.9713	0.9786	0.9749	1073
4	0.9605	0.9672	0.9638	3015
5	0.975	0.9739	0.9745	3564
6	0.9614	0.95	0.9556	1519
7	0.9671	0.94	0.9534	3752
8	0.9674	0.9704	0.9689	1317
9	0.9627	0.9424	0.9524	903
10	0.9031	0.8698	0.8861	3588
11	0.9365	0.9522	0.9443	1424
12	0.8776	0.8543	0.8658	1318
13	0.9236	0.8951	0.9091	324
14	0.964	0.9469	0.9554	226
15	0.9564	0.9594	0.9579	320
Accuracy			0.9446	36401
Macro Avg	0.9472	0.9414	0.9442	36401
Weighted Avg	0.9445	0.9446	0.9445	36401

Table 3 SAGE Fold 5 Classification Report

Models Methodology

3.3 GAT Architecture

- **Input features:**
  - Pretrained text embeddings
  - Normalized price values
  - Graph Edges

Model Design

- **Layer 1**: Multi-head attention → concatenated features
- **Hidden layers**: Single-head attention for refinement
- **Model Layers**:
  - BatchNorm
  - ELU activation
  - Dropout

Skip Input Connection

- Original input projected to output space
- Added to GAT final output
- Benefits: Original feature preservation, improved gradient flow

Output and Classification

- Final layer outputs **logits for 16 categories**
- Softmax applied for prediction

Models Methodology

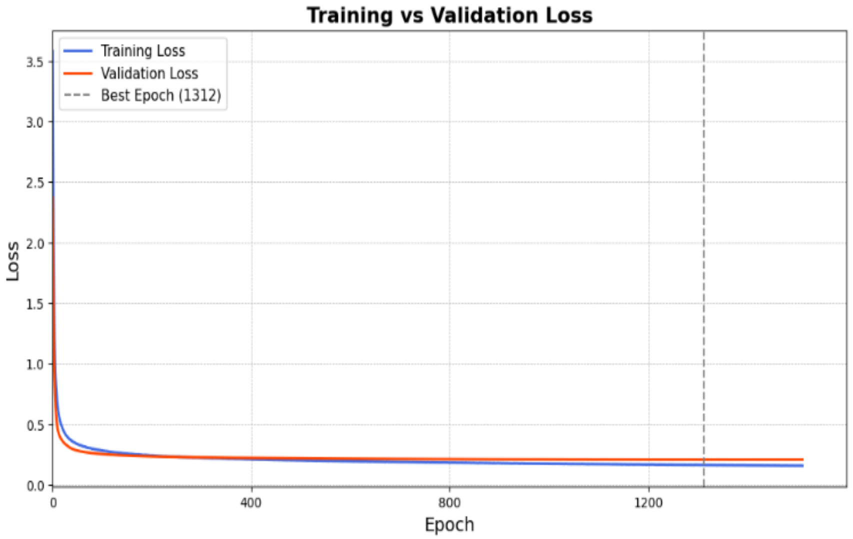

Figure 6 Training and Validation Losses for GAT

Class	Precision	Recall	F1-score	Support
0	0.977	0.9796	0.9783	1517
1	0.9067	0.9182	0.9124	1186
2	0.9272	0.9505	0.9387	4326
3	0.956	0.9721	0.964	537
4	0.9516	0.9655	0.9585	1508
5	0.9723	0.9658	0.969	1782
6	0.9635	0.9368	0.95	760
7	0.9531	0.9424	0.9477	1876
8	0.9697	0.9726	0.9712	658
9	0.9529	0.9424	0.9476	451
10	0.8792	0.8724	0.8758	1794
11	0.9408	0.9157	0.9281	712
12	0.8854	0.8209	0.852	659
13	0.9338	0.8704	0.901	162
14	1	0.9558	0.9774	113
15	0.9742	0.9437	0.9587	160
Accuracy			0.9389	18201
Macro Avg	0.9465	0.9328	0.9394	18201
Weighted Avg	0.9389	0.9389	0.9388	18201

Table 2 GAT Classification Report

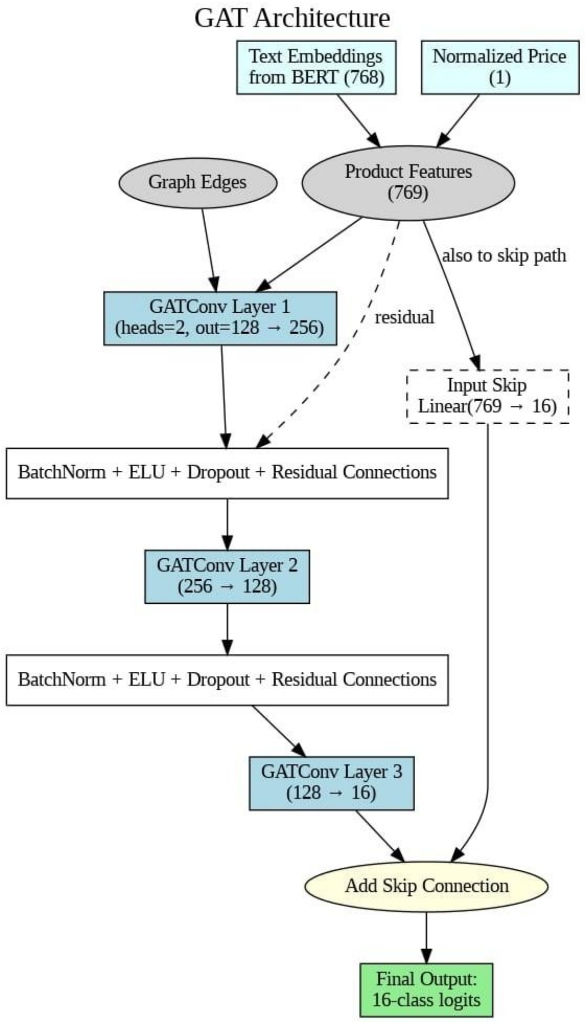

Figure 7 GAT Architecture

Submission Results and Analysis

Top-Performing Submissions

- **GraphSAGE (5-fold averaging)**
  - **Private Score**: 0.1890
  - **Public Score**: 0.1943
  - **File**: submission_graphsage_5fold_avg_new.csv
  - **Insight:** Outperformed all others despite its simplicity. Highlights the strength of a well-tuned, singular GNN architecture.

- **Ensemble (GAT + GraphSAGE)**
  - **Private Score**: 0.1922
  - **Public Score**: 0.1985
  - **File**: ensemble_2.csv
  - **Insight**: Increased model complexity did not improve results. Suggests potential overfitting or lack of complementary learning between models.

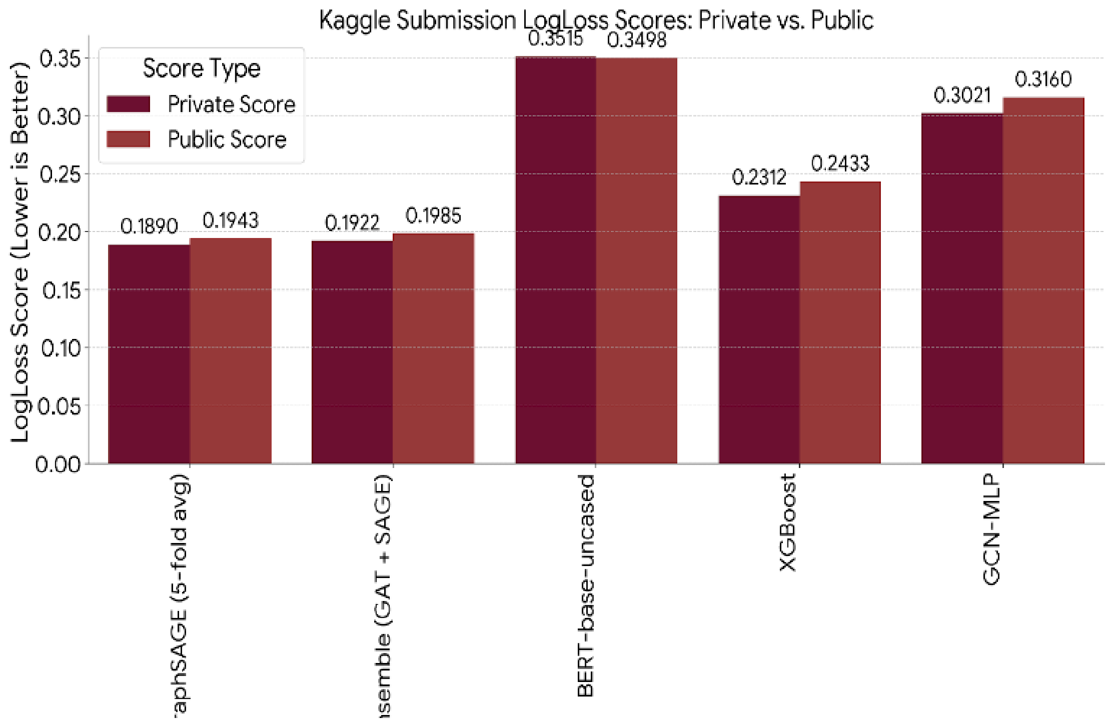

Figure 8 5 Best Kaggle Submissions

Q&A

Thank You!