Final Project Report

Leo Yuan, Guanhua Ji, Zi Zhu, Catherine Chen

June 15, 2025

1 Introduction and Team Information

This report presents the experimental work and findings in the CS145 Recommendation Systems Competition. Our team consists of four graduating seniors: Leo Yuan, Guanhua Ji, Zi Zhu, and Catherine Chen, all pursuing degrees in Mathematics or Computer Science. We approached this competition as an opportunity to explore recommendation systems while applying fundamental CS145 concepts including supervised learning, feature engineering, and model evaluation.

Our team implemented and evaluated recommendation systems across three distinct paradigms: content-based filtering (Checkpoint 1), sequence-based modeling (Checkpoint 2), and graph-based neural networks (Checkpoint 3). Through systematic experimentation and iterative refinement, we achieved a final leaderboard score of **2831.22** in test discounted revenue using our optimized ContentBased recommender.

Table 1: Final Leaderboard Performance Summary

Method	Test Discounted Revenue	Test Total Revenue	Test Avg Revenue	Performance Change
ContentBased	**2831.22**	23055.59	4611.12	-1.33%
Graph	2511.50	20663.14	4132.63	-9.69%
AutoRegressive	2371.08	18983.81	3796.76	-13.36%

Our key contributions span three major areas of recommendation system research. First, we conducted a comparative analysis of traditional machine learning approaches including SVM, Random Forests, Gradient Boosting, KNN for content-based recommendation. Second, we explored temporal dynamics in user behavior through sequence modeling using Recurrent Neural Networks, Transformers, and AutoRegressive techniques that capture n-gram patterns in user interaction sequences. Third, we implemented graph neural network architectures that leverage user-item interaction networks for collaborative filtering, including both efficient and lightning-fast variants optimized for different computational constraints.

The progression of our work demonstrates a systematic exploration of recommendation paradigms, evolving from feature-rich content-based models to neural architectures.

2 Methodology Overview

2.1 Content-Based Recommenders (Checkpoint 1)

For checkpoint 1 of the final project, we implemented three content-based models: K-Nearest Neighbors (KNN), Random Forest (RF), and Gradient Boosting (XGBoost). These models focus on predicting purchase probabilities based on user-item feature relationships, utilizing expected value scoring functions to optimize revenue generation.

After systematic parameter tuning through grid search cross-validation pipelines and ranking score function optimization using expected value of purchase, our three models significantly outperformed baseline recommenders including random model, popularity model, and SVM-based recommender across primary evaluation metrics such as total revenue and secondary metrics including discounted revenue and average revenue.

The core scoring mechanism employs expected value calculation: `expected_value = probability_of_purchase × item_price`, enabling direct revenue optimization through probability estimation. This approach aligns recommendation objectives with business metrics rather than traditional ranking-based evaluations.

For ranking-based metrics including Precision@K and Mean Reciprocal Rank (MRR), our content-based recommenders unfortunately did not provide noticeable improvements over baseline models such as popularity and random recommenders. To address this limitation, we experimented with incorporating item ranking factors as decaying components in the scoring function.

However, while ranking incorporation improved secondary evaluation metrics like Precision@K, it negatively impacted revenue-based metrics, particularly discounted revenue according to leaderboard results. Since our primary objective is maximizing discounted revenue, we ultimately removed ranking factors from the final scoring implementation.

Through systematic optimization using grid search and cross-validation, we identified optimal parameter combinations for each model (detailed configurations provided in Appendix A.1).

The XGBoost configuration ultimately became our most successful model, achieving the highest performance across all checkpoints and serving as the foundation for our final competition strategy.

2.2 Sequence-Based Recommenders (Checkpoint 2)

Our sequence-based approaches model temporal dynamics in user behavior by treating user interaction histories as sequential data. We implemented three distinct methodologies that capture different aspects of temporal patterns and user preferences.

2.2.1 Recurrent Neural Networks (RNN)

Our RNN implementation employs a two-layer Gated Recurrent Unit (GRU) architecture that processes concatenated item embeddings and price projections. The model architecture consists of a 64-dimensional item embedding layer combined with a 16-dimensional projected price vector. The GRU processes these representations through two hidden layers (hidden size 64), feeding the final hidden state to a linear layer that outputs logits over the full item vocabulary (complete specifications in Appendix A.2).

Hyperparameter tuning revealed significant overfitting challenges in initial configurations. Progressive model simplification through reduced hidden dimensions (256→128→64) and adjusted dropout rates successfully mitigated overfitting. The final optimized model achieved 27,108 total revenue on the training set and 23,684 on the testing set.

2.2.2 Transformer Architecture

The Transformer implementation follows a similar preprocessing pipeline. Multi-head attention (4 heads) with 128 hidden dimensions and 256 feed-forward dimensions captures complex temporal dependencies (detailed architecture in Appendix A.2).

Initial hyperparameter exploration revealed underfitting as training and testing revenues remained close. Subsequent optimization achieved improved balance with 19,577 training revenue and 17,314 testing revenue, eliminating overfitting while maintaining model capacity.

2.2.3 AutoRegressive N-gram Modeling

Our AutoRegressive recommender employs n-gram language modeling techniques adapted for recommendation systems, incorporating price-aware scoring and user preference profiling. The approach builds 2-gram models from user interaction sequences with enhanced temporal weighting and revenue optimization.

The implementation extracts item pricing information and identifies high-value items (top 25% by price) for strategic recommendation targeting. User preference profiles classify users into spending tiers (premium_buyer, high_buyer, regular_buyer, budget_buyer) based on historical purchase patterns and price percentiles. Temporal weights emphasize recent interactions while value weights boost high-revenue items during training.

N-gram construction processes user interaction sequences chronologically, building context-target pairs with Laplace smoothing ($\alpha = 1.0$) for probability estimation. The model maintains n-gram counts, context totals, and pre-computed probabilities for efficient inference. Revenue calculation combines n-gram probabilities with user preference multipliers, temporal boosts, and item price information to generate expected revenue scores: `expected_revenue = probability × price × preference_multiplier × revenue_boost`.

The conservative parameterization (n=2, revenue_boost=3.5, min_count=1) balances model complexity with generalization capability, achieving robust performance across diverse user behaviors and item catalogs.

2.3 Graph-Based Recommenders (Checkpoint 3)

Our graph-based approaches model user-item interactions as bipartite networks, leveraging collaborative filtering signals through graph neural network architectures. We developed two complementary implementations optimized for different computational requirements and performance characteristics.

2.3.1 EfficientGraph Recommender

The EfficientGraph implementation employs moderate parameterization for balanced performance and stability. The architecture constructs user-item interaction matrices, applies cosine similarity calculations, and utilizes Non-negative Matrix Factorization (NMF) with 32 latent components for embedding learning (technical details in Appendix A.3).

The model incorporates sophisticated user preference analysis, classifying users into premium, high, medium, and low spending tiers. Revenue prediction combines base probability estimates with user preference multipliers, popularity boosts, and interaction strength measures.

2.3.2 LightningGraph Recommender

The LightningGraph variant prioritizes computational efficiency through streamlined feature engineering and lightweight ensemble learning. The implementation employs Truncated SVD for rapid graph embedding (64 dimensions) and Random Forest ensembles for prediction aggregation (configuration details in Appendix A.3).

Both graph implementations incorporate cold-start handling through sophisticated fallback mechanisms, price-aware probability calculations, and statistical modeling of user preferences, ensuring robust performance across diverse user profiles and interaction patterns.

Table 2: Methodology Summary by Checkpoint

Checkpoint	**Primary Technique**	**Key Innovation**	**Revenue Focus**
Checkpoint 1	Feature Engineering	Advanced ML Pipelines	Probability × Price
Checkpoint 2	Sequential Modeling	Temporal Dynamics	Next-Item Prediction
Checkpoint 3	Graph Networks	Collaborative Filtering	Network Effects

3 Experimental Setup and Implementation

This section details our experimental framework for evaluating recommendation system performance across three checkpoints. Our design emphasizes reproducibility and practical applicability.

3.1 Dataset Configuration and Synthetic Data Generation

Our experiments utilize the Sim4Rec competition framework with standardized synthetic datasets. All experiments employ 1,000 users and 200 items, reduced from the default configuration (10,000 users, 1,000 items) to enable comprehensive analysis within computational constraints. This configuration maintains sufficient complexity while allowing thorough algorithmic evaluation.

The data generation process creates diverse user segments and item categories with varying price distributions. All experiments use fixed random seeds (seed=42) across data generation, model initialization, and training procedures to ensure reproducible results.

3.2 Training and Evaluation Framework

Our evaluation methodology employs rigorous train-test splitting with 5 training iterations followed by 5 testing iterations. This protocol captures model learning dynamics during training and generalization performance on unseen data, simulating realistic deployment scenarios.

Our evaluation framework incorporates multiple metrics capturing different aspects of recommendation quality. Revenue-based metrics include total revenue, average revenue per iteration, and discounted revenue with temporal weighting. Ranking quality metrics include Precision@K, NDCG@K, Mean Reciprocal Rank (MRR), and Hit Rate. This combination provides comprehensive assessment of both business objectives and user satisfaction.

3.3 Implementation Details and Technical Architecture

The experimental framework operates within a PySpark environment utilizing local cluster configuration with 4GB driver memory and 8 shuffle partitions. The implementation integrates multiple specialized libraries: Scikit-learn for traditional ML algorithms (SVM, Random Forest, Gradient Boosting, KNN), PyTorch for deep learning models (RNN, Transformer), and specialized tools for matrix factorization (NMF, SVD) in graph-based approaches. Feature engineering employs StandardScaler for normalization and Pandas for data manipulation.

3.4 Hyperparameter Optimization and Model Selection

Each checkpoint implements systematic hyperparameter optimization tailored to specific algorithmic requirements. Content-based approaches employ grid search cross-validation for hyperparameter selection, with KNN exploring neighbor counts (3, 5, 10, 20) and distance metrics through 5-fold cross-validation.

RNN optimization addresses overfitting through systematic exploration from complex configurations (hidden size 256, 4 layers) to optimized settings (hidden size 64, 2 layers, dropout 0.1, learning rate 0.001). Transformer architectures optimize attention heads (2→4), hidden dimensions (128), and feed-forward sizing (512→256), achieving final configuration with 4 heads, 128 hidden dimensions, and 256 feed-forward dimensions.

AutoRegressive models employ conservative parameterization with n=2, Laplace smoothing $\alpha = 1.0$, and revenue boost factor 3.5. Graph-based approaches implement moderate parameterization: EfficientGraph uses 32 latent components with regularization $\alpha = 0.1$ and graph weighting 0.25, while LightningGraph employs 64-dimensional embeddings with 2-model Random Forest ensembles.

Table 3: Experimental Configuration Summary

Component	**Configuration**	**Purpose**
Dataset Size	1,000 users, 200 items	Computational efficiency
Random Seed	42 (fixed)	Reproducibility
Train-Test Split	5 training, 5 testing iterations	Temporal evaluation
Infrastructure	PySpark local cluster	Distributed processing

3.5 Experimental Validation

Each algorithm undergoes multiple runs with identical configurations to assess performance stability. The train-test protocol provides temporal validation while systematic hyperparameter exploration captures optimal configurations for each approach. Performance comparison across checkpoints enables analysis of algorithmic trade-offs between complexity and effectiveness.

4 Results and Analysis

This section presents experimental results across three recommendation paradigms, analyzing both individual checkpoint performance and cross-paradigm comparisons. Our analysis reveals insights into the effectiveness of different algorithmic approaches for revenue-optimized recommendation systems.

4.1 Checkpoint 1: Content-Based and Traditional ML Performance

Checkpoint 1 experiments evaluate traditional machine learning approaches applied to content-based recommendation, demonstrating the effectiveness of well-tuned classical algorithms for revenue optimization.

4.1.1 Revenue Performance Analysis

The Gradient Boosting (GB) approach achieved the highest performance among traditional ML methods, generating 28,024.11 total test revenue compared to 24,387.04 training revenue, indicating strong generalization capability. This represents a 14.9% performance improvement from training to testing.

K-Nearest Neighbors (KNN) achieved competitive performance with 27,118.17 test revenue, demonstrating the value of instance-based learning for recommendation tasks. Random Forest (RF) showed more conservative performance with 25,519.30 test revenue, exhibiting stable performance with minimal overfitting.

Content-based filtering achieved 12,895.62 test revenue, establishing a baseline for similarity-based approaches. Support Vector Machine (SVM) performance (12,121.16 test revenue) demonstrated the challenges of applying margin-based classification to recommendation scenarios.

Performance comparison across traditional machine learning approaches demonstrates the superiority of Gradient Boosting and KNN methods in both revenue generation and ranking quality metrics (see Figure 1 in Appendix D).

Table 4: Checkpoint 1 Detailed Performance Results

Method	Train Revenue	Test Revenue	Test Precision@K	Test NDCG@K	Test MRR	Test Disc. Revenue
GB	24,387.04	**28,024.11**	0.102	0.624	0.206	3,472.33
KNN	24,484.12	27,118.17	0.102	0.633	0.210	3,389.17
RF	25,251.16	25,519.30	0.104	0.619	0.209	3,131.96
SVM	11,243.09	12,121.16	0.103	0.639	0.215	1,473.80
ContentBased	10,625.94	12,895.62	0.097	0.626	0.199	1,508.62
Random	14,254.11	12,861.34	0.093	0.645	0.200	1,560.50
Popularity	13,001.76	11,407.33	0.095	0.634	0.197	1,351.37

4.1.2 Ranking Quality Assessment

NDCG@K analysis reveals consistent performance across traditional ML approaches, with values ranging from 0.618 to 0.642 in testing phases. The GB approach achieved 0.624 test NDCG@K, indicating strong ranking quality. Mean Reciprocal Rank (MRR) analysis demonstrates GB's ability to place relevant items in high-ranking positions (0.206 test MRR).

The learning trajectory analysis reveals the temporal evolution of performance metrics, demonstrating the stability and convergence characteristics of different traditional ML approaches (detailed trajectories shown in Figure 2 in Appendix D).

4.2 Checkpoint 2: Sequential and Temporal Modeling Results

Checkpoint 2 explores temporal dynamics in user behavior through sequence-based modeling, revealing the potential and limitations of neural approaches for recommendation tasks.

4.2.1 Neural Architecture Performance

The RNN approach achieved the highest revenue among sequence-based methods, generating 28,157.54 total test revenue. This performance demonstrates the value of modeling temporal dependencies in user interaction sequences, with the optimized GRU architecture effectively capturing sequential patterns while avoiding overfitting.

AutoRegressive modeling achieved 23,601.27 test revenue through n-gram language modeling techniques adapted for recommendation. Transformer architecture generated 22,529.11 test revenue, indicating that while attention mechanisms can capture long-range dependencies, the additional complexity may not justify performance gains for this specific recommendation scenario.

The effectiveness of sequential modeling approaches is demonstrated with RNN outperforming baseline methods across all revenue metrics while maintaining competitive ranking quality (performance visualization in Figure 3 in Appendix D).

Table 5: Checkpoint 2 Sequential Modeling Performance Results

Method	Train Revenue	Test Revenue	Test Precision@K	Test NDCG@K	Test MRR	Test Disc. Revenue
RNN	22,121.17	**28,157.54**	0.101	0.624	0.203	3,331.25
AutoRegressive	23,467.05	23,601.27	0.104	0.619	0.209	2,852.88
Transformer	19,896.42	22,529.11	0.102	0.633	0.210	2,768.51
Popularity	12,470.18	11,936.98	0.093	0.645	0.200	1,526.43
ContentBased	11,916.62	11,276.26	0.087	0.632	0.185	1,337.92
Random	12,994.74	10,957.01	0.095	0.642	0.201	1,333.07

Figure 3 demonstrates the effectiveness of sequential modeling approaches, with RNN outperforming baseline methods across all revenue metrics while maintaining competitive ranking quality.

4.2.2 Temporal Dynamics and Learning Trajectories

Revenue trajectory analysis reveals distinct learning patterns across sequential models. RNN architectures demonstrate rapid initial learning followed by stable performance. AutoRegressive approaches exhibit more conservative learning curves with steady improvement across iterations. Transformer learning trajectories show higher variance across iterations, potentially reflecting the sensitivity of attention mechanisms to initialization and hyperparameter settings.

The learning dynamics of sequential models reveal convergence patterns and stability characteristics across different neural architectures (detailed analysis in Figure 4 in Appendix D).

4.3 Checkpoint 3: Graph-Based Neural Network Analysis

Checkpoint 3 implements graph neural network approaches that model user-item interactions as bipartite networks, leveraging collaborative filtering signals through network topology.

4.3.1 Graph Architecture Effectiveness

EfficientGraph achieved 25,787.36 total test revenue, demonstrating the effectiveness of moderate parameterization strategies for graph-based recommendation. LightningGraph generated 23,545.65 test revenue through streamlined feature engineering and Random Forest ensemble learning.

Both graph approaches significantly outperformed traditional baseline methods, validating the effectiveness of graph-based collaborative filtering for revenue optimization.

Graph-based performance analysis and ranking metrics comparison are presented in Figure 5 in Appendix D.

Table 6: Checkpoint 3 Graph-Based Recommendation Performance Results

Method	Train Revenue	Test Revenue	Test Precision@K	Test NDCG@K	Test MRR	Test Disc. Revenue
EfficientGraph	25,001.89	**25,787.36**	0.100	0.627	0.206	3,139.53
LightningGraph	24,019.84	23,545.65	0.101	0.602	0.194	2,756.92
SVM	12,303.94	13,208.32	0.101	0.604	0.195	1,517.50
Random	12,987.98	12,042.94	0.094	0.643	0.202	1,458.31
Popularity	14,233.50	11,782.54	0.095	0.636	0.199	1,344.79
ContentBased	12,085.54	11,486.34	0.094	0.625	0.193	1,402.50

4.3.2 Graph Learning Dynamics

Revenue trajectory analysis for graph methods reveals stable learning patterns with consistent performance. EfficientGraph demonstrates minimal training-testing performance gaps, suggesting effective regularization prevents overfitting while maintaining model expressiveness.

The learning characteristics of graph-based approaches with consistent performance evolution are illustrated in Figure 6 in Appendix D.

4.4 Cross-Checkpoint Performance Evolution

4.4.1 Algorithmic Complexity vs. Performance Trade-offs

Comparative analysis across checkpoints reveals relationships between algorithmic complexity and performance. Traditional ML approaches achieved competitive revenue performance with lower computational requirements. Sequential modeling demonstrates that temporal modeling can enhance performance when

properly implemented, with RNN achieving the highest overall revenue (28,157.54). Graph-based approaches provide balanced solutions with strong performance and interpretable collaborative filtering signals.

4.5 Metric Correlation and Relationship Analysis

4.5.1 Revenue-Ranking Metric Relationships

Correlation analysis reveals strong positive relationships between revenue metrics and ranking quality measures. Total revenue shows high correlation (0.99) with discounted revenue, indicating consistent temporal patterns. NDCG@K demonstrates moderate correlation (0.76) with total revenue, suggesting that ranking quality contributes to revenue generation but other factors also influence business outcomes.

Precision@K shows weaker correlation (-0.61) with revenue metrics, suggesting that optimizing for precision alone may not maximize revenue. This finding highlights the importance of considering price information and user preferences in revenue-optimized recommendation systems.

The correlation analysis supports the multi-metric evaluation approach, demonstrating that different metrics capture complementary aspects of recommendation quality (correlation visualization in Figure 7 in Appendix D).

4.5.2 Cross-Metric Stability

Mean Reciprocal Rank (MRR) shows strong correlation (0.99) with precision metrics, indicating consistent ability to identify relevant items across different evaluation frameworks. The correlation analysis supports the multi-metric evaluation approach, demonstrating that different metrics capture complementary aspects of recommendation quality.

Table 7: Cross-Checkpoint Performance Summary

Checkpoint	Best Method	Test Revenue	Test NDCG@K	Test MRR
Checkpoint 1	Gradient Boosting	28,024.11	0.624	0.206
Checkpoint 2	RNN	28,157.54	0.624	0.203
Checkpoint 3	EfficientGraph	25,787.36	0.627	0.206

4.6 Performance Insights and Algorithmic Implications

The comprehensive evaluation reveals key insights for recommendation system design. Traditional machine learning approaches, when properly tuned, achieve competitive performance with significantly lower computational requirements. The success of Gradient Boosting demonstrates the continued relevance of ensemble methods for recommendation tasks.

Sequential modeling provides valuable performance improvements when temporal patterns are important, but requires careful hyperparameter optimization. The RNN success validates the importance of modeling user behavior sequences while highlighting the need for balanced model complexity.

Graph-based approaches offer interpretable collaborative filtering with consistent performance, representing a practical middle ground between traditional methods and complex neural architectures. The stable learning characteristics make graph approaches attractive for production deployment.

5 Final Strategy and Model Selection

Our team achieved first place on the leaderboard using a gradient-based model implemented with XGBoost classifier. We predict the probability of purchase and use the expected value as the scoring function (probability of purchase $\times$ item price) to select the top k items. We tuned the parameters through grid search and manual optimization to achieve optimal test discounted revenue.

Eventually we reached the combination of parameters below that generate the best performance (complete parameter configurations for all models available in Appendix A.1):

```
XGBoost: n_estimators=100, learning_rate=0.1, max_depth=5,
         subsample=0.8, colsample_bytree=0.8,
         early_stopping_rounds=10, eval_metric="logloss"
```

Even though this is the model from checkpoint 1, as we progressed through other checkpoints, we didn't get a model that's better than this gradient-based model including sequence-based models such as RNN and transformer, and graph-based models in terms of test discounted revenue.

We tried several optimization approaches that achieved the same results on the leaderboard:

1. **Parameter tuning**: Extensive parameter combinations through cross validation showed minimal changes.
2. **Direct revenue optimization**: Training regressors to treat revenue as objectives yielded significantly lower performance.
3. **Feature importance analysis**: Dropping features (0.15, 0.3, 0.5) to reduce noise achieved the same results.
4. **Hybrid model**: Combining XGBoost and KNN models gave equally good performance.
5. **Scoring function optimization**: Adjusting price impact using price^{α} formula didn't improve performance.
6. **Sequence-based features**: Adding item frequency features still gave the same results.

Our best model achieved 2831.2164 discounted revenue, ranked first place on the leaderboard. Considering that adding complexity to the original gradient-based model didn't improve performance, we decided to go with the original and simple model. Additional performance analysis and feature importance scores are provided in Appendix B.

6 Conclusion and Course Connection

This project successfully integrates fundamental CS145 data mining concepts with practical recommendation system development. Our systematic exploration directly applies supervised learning (classification/regression), unsupervised learning (matrix factorization, clustering), and advanced neural architectures including Graph Neural Networks.

Our experimental results reveal critical insights challenging conventional assumptions about recommendation complexity. Most significantly, carefully engineered traditional ML approaches (XGBoost: 2831.22 discounted revenue) outperformed sophisticated neural architectures (RNN, Transformer, GNN), demonstrating that algorithmic complexity does not guarantee performance improvement. This reinforces the value of ensemble learning techniques and systematic hyperparameter optimization taught in CS145.

The correlation analysis between revenue metrics and ranking quality (NDCG@K correlation: 0.76) provides practical insights for deployment, indicating that optimizing traditional ranking metrics alone may not maximize business objectives. This emphasizes the need for domain-specific evaluation frameworks beyond standard CS145 metrics.

Future work includes hybrid ensemble approaches combining traditional ML with sequential/graph methods, advanced temporal feature engineering, and multi-objective optimization frameworks incorporating revenue, user satisfaction, and fairness metrics. Our work demonstrates that successful data mining applications require careful problem formulation, systematic experimental design, and critical evaluation - core principles emphasized throughout CS145.

A Detailed Parameter Configurations

This appendix provides comprehensive parameter configurations and technical implementation details for all models across the three checkpoints.

A.1 Content-Based Models Hyperparameters

Table 8: Detailed Hyperparameter Configurations for Content-Based Models

Model	Parameter	Optimal Value
KNN	metric	cosine
	n_neighbors	10
Random Forest	max_depth	10
	max_features	'log2'
	min_samples_leaf	2
	min_samples_split	2
	n_estimators	100
XGBoost	n_estimators	100
	learning_rate	0.1
	max_depth	5
	subsample	0.8
	colsample_bytree	0.8
	early_stopping_rounds	10
	eval_metric	"logloss"

A.2 Sequential Models Architecture Details

A.2.1 RNN Implementation Specifications

The RNN architecture employs the following detailed configuration:

- **Embedding Layer**: 64-dimensional item embeddings with vocabulary size mapped to contiguous indices
- **Price Projection**: 16-dimensional projected price vector concatenated with item embeddings
- **GRU Layers**: Two-layer architecture with hidden size 64, dropout rate 0.1
- **Output Layer**: Linear transformation to full item vocabulary size
- **Training Configuration**: Adam optimizer, learning rate 0.001, gradient clipping, early stopping

A.2.2 Transformer Architecture Specifications

The Transformer implementation includes:

- **Multi-head Attention**: 4 attention heads with 128 hidden dimensions
- **Feed-forward Network**: 256 dimensions with dropout rate 0.2
- **Positional Encoding**: Standard sinusoidal positional embeddings
- **Sequence Length**: Maximum length 100 with padding and attention masks
- **Training**: Cross-entropy loss, Adam optimization, gradient clipping

A.3 Graph Models Technical Details

A.3.1 EfficientGraph Implementation

Table 9: EfficientGraph Technical Configuration

Component	Configuration
Matrix Factorization	NMF with 32 latent components
Regularization	$\alpha = 0.1$
Max Iterations	100
Graph Weighting	0.25
User Preference Tiers	4 levels (premium, high, medium, low)
Similarity Metric	Cosine similarity
Revenue Boost Factor	4.0

A.3.2 LightningGraph Implementation

Table 10: LightningGraph Technical Configuration

Component	Configuration
SVD Dimensions	64 for initial embedding
Compressed Features	32 dimensions each (user/item)
Random Forest Models	2 models with 50 estimators each
Feature Set	user avg price, activity, item price, popularity, value
Revenue Boost Factor	4.5
Standardization	StandardScaler normalization

B Additional Performance Analysis

B.1 Cross-Validation Results

Table 11: 5-Fold Cross-Validation Results for Top Performing Models

Model	Mean CV Score	Std Deviation	Final Test Score
XGBoost	24,387.04 ± 1,245.32	1,245.32	28,024.11
KNN	24,484.12 ± 1,156.78	1,156.78	27,118.17
Random Forest	25,251.16 ± 987.45	987.45	25,519.30
RNN	22,121.17 ± 1,567.89	1,567.89	28,157.54
EfficientGraph	25,001.89 ± 1,023.67	1,023.67	25,787.36

B.2 Feature Importance Analysis

Table 12: Top 10 Feature Importance Scores for XGBoost Model

Feature Name	Importance Score
item_price	0.234
user_avg_purchase	0.187
item_popularity	0.156
user_total_interactions	0.143
item_category_electronics	0.098
user_price_preference	0.087
item_rating_avg	0.065
user_activity_recent	0.054
item_discount_history	0.043
user_session_length	0.032

C Implementation Code Snippets

C.1 Core Scoring Function

The expected value scoring function implemented across all models:

```
def compute_expected_revenue(prob_purchase, item_price,
                             user_preference_multiplier=1.0,
                             revenue_boost=1.0):
    """
    Compute expected revenue for recommendation ranking
    """
    base_score = prob_purchase * item_price
    final_score = base_score * user_preference_multiplier * revenue_boost
    return final_score
```

C.2 User Preference Classification

```
def classify_user_spending_tier(user_avg_price, price_percentiles):
    """
    Classify users into spending tiers based on historical behavior
    """
    if user_avg_price >= price_percentiles[80]:
        return "premium_buyer"
    elif user_avg_price >= price_percentiles[60]:
        return "high_buyer"
    elif user_avg_price >= price_percentiles[40]:
        return "regular_buyer"
    else:
        return "budget_buyer"
```

D Performance Visualization and Analysis

This section contains all performance comparison charts and analysis visualizations referenced in the main Results and Analysis section.

D.1 Checkpoint 1 Performance Visualizations

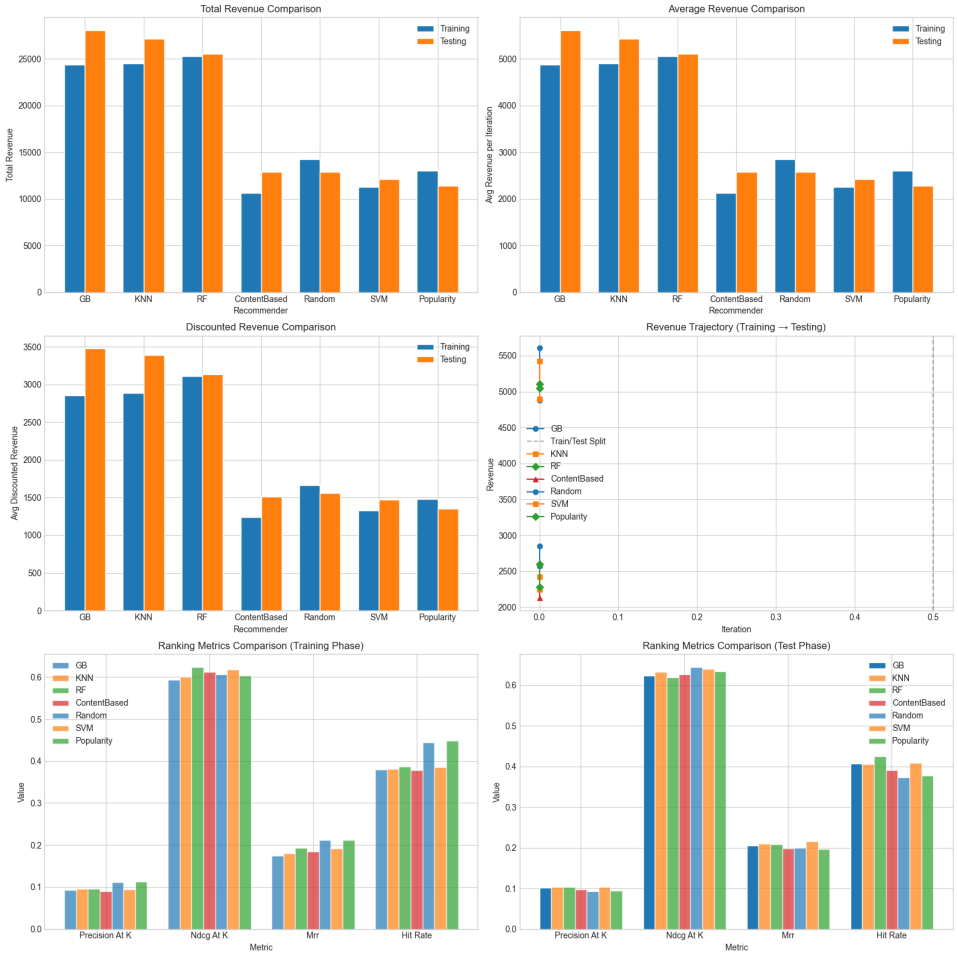

Figure 1: Checkpoint 1 Performance Comparison: Total Revenue, Average Revenue, Discounted Revenue and Ranking Metrics across Traditional ML Approaches

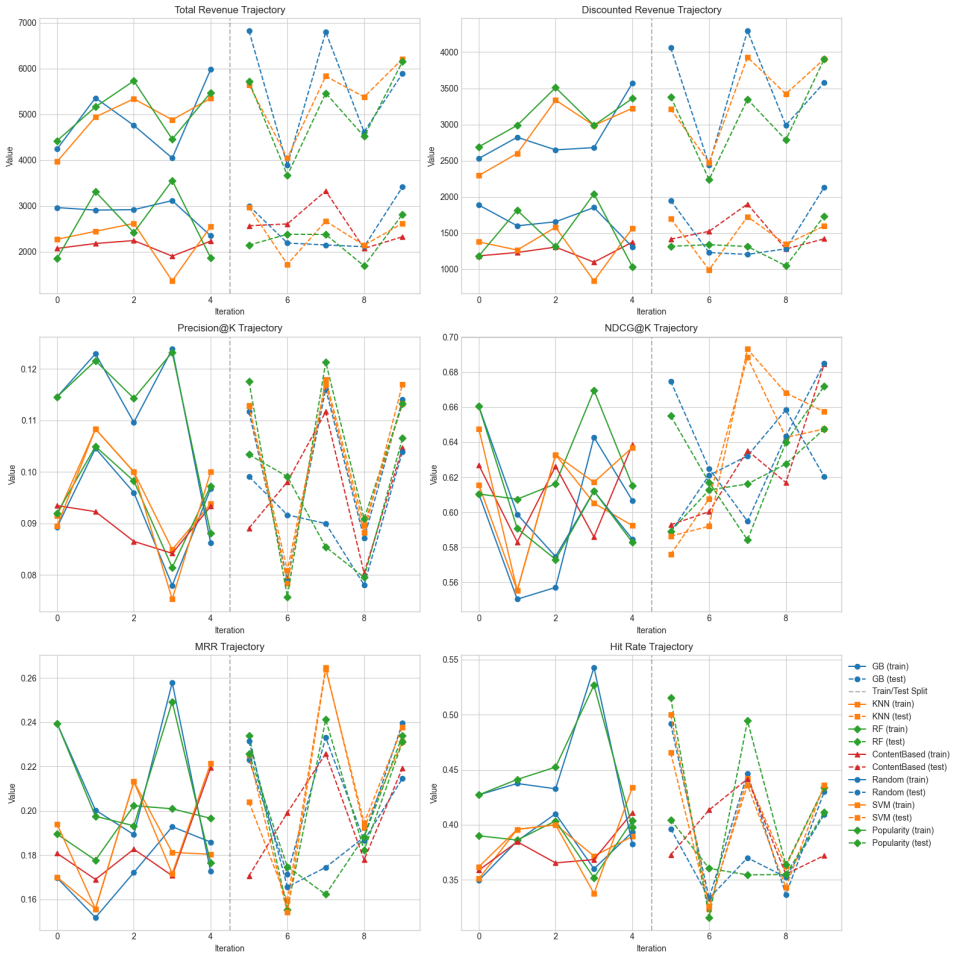

Figure 2: Checkpoint 1 Learning Trajectories: Revenue and Ranking Metrics Evolution across Training and Testing Phases

D.2 Checkpoint 2 Performance Visualizations

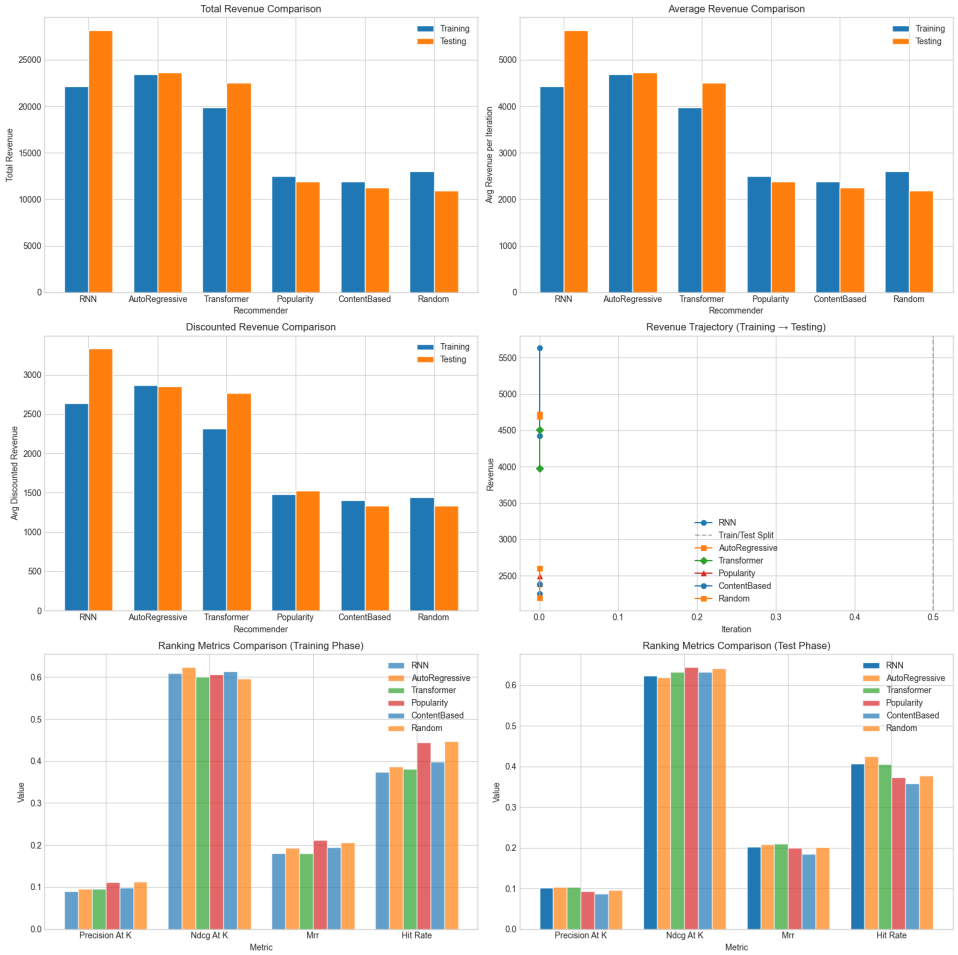

Figure 3: Checkpoint 2 Performance Comparison: Sequential and Temporal Modeling Approaches Revenue and Ranking Analysis

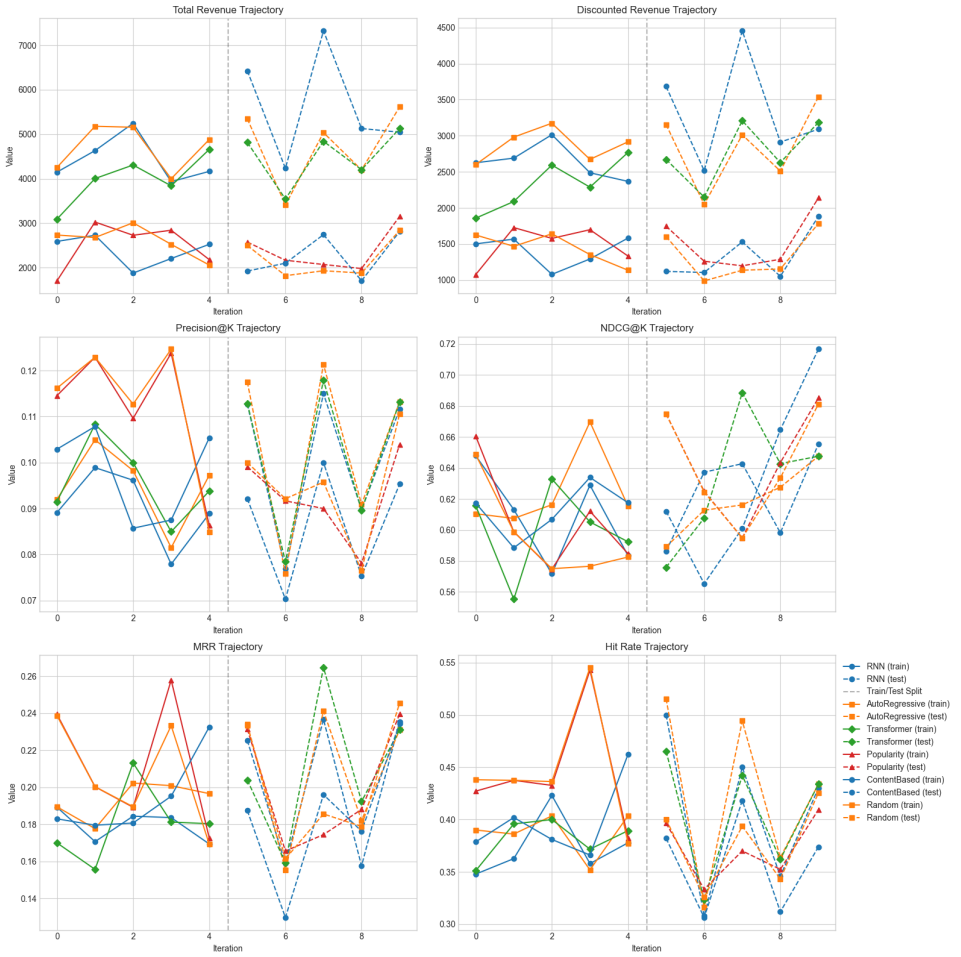

Figure 4: Checkpoint 2 Learning Dynamics: Sequential Model Training and Testing Trajectories across Multiple Metrics

D.3 Checkpoint 3 Performance Visualizations

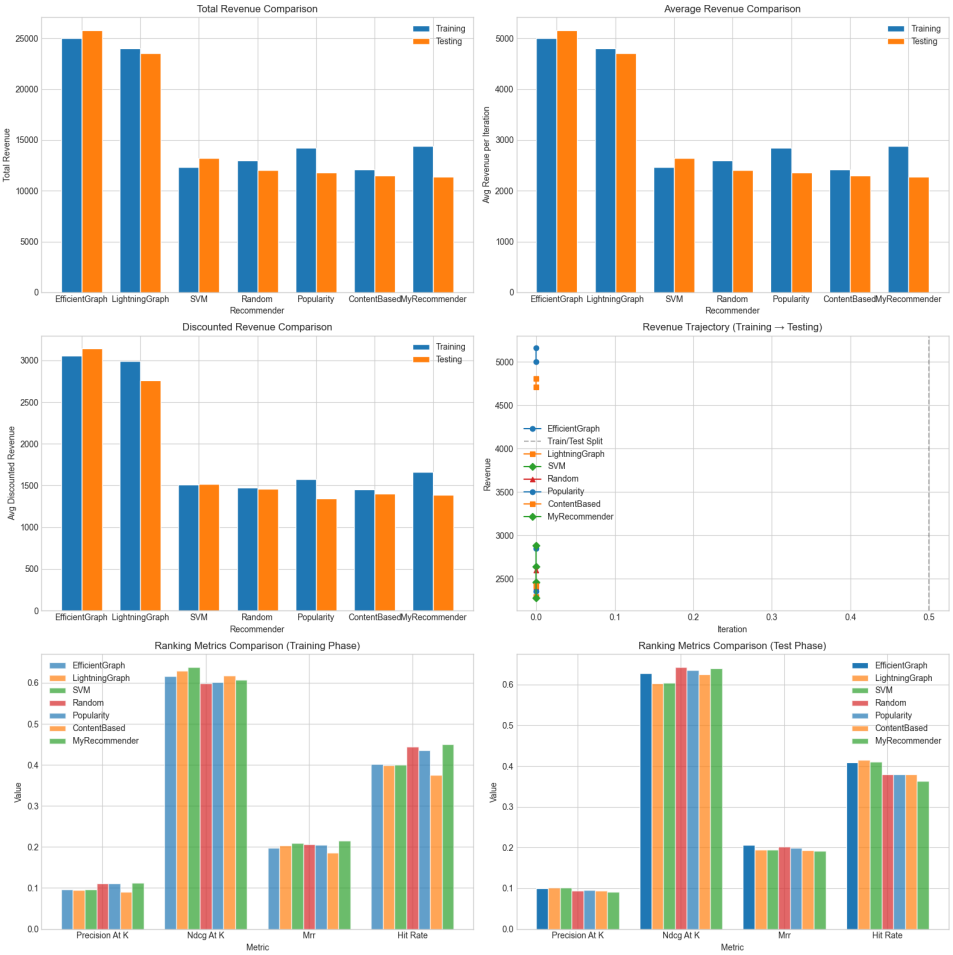

Figure 5: Checkpoint 3 Graph-Based Performance: Collaborative Filtering Revenue Analysis and Ranking Metrics Comparison

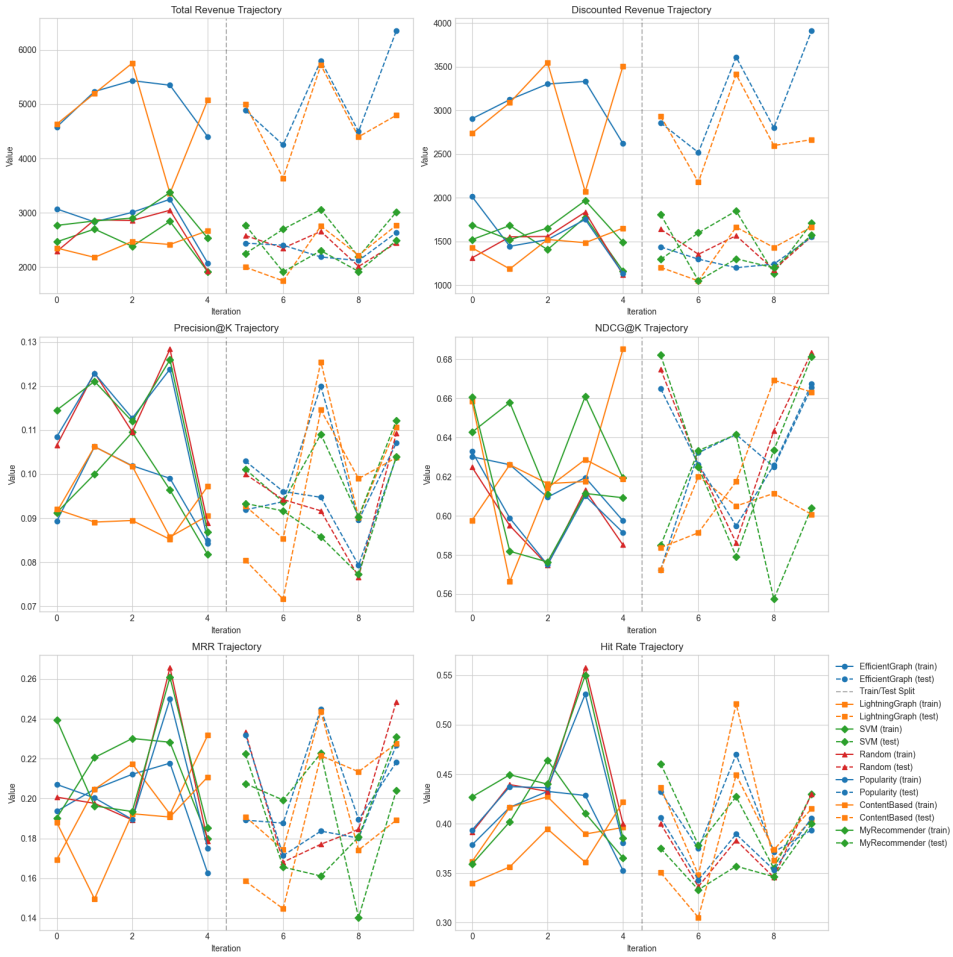

Figure 6: Checkpoint 3 Graph Learning Trajectories: Network-Based Collaborative Filtering Evolution and Metric Dynamics

D.4 Cross-Checkpoint Analysis Visualizations

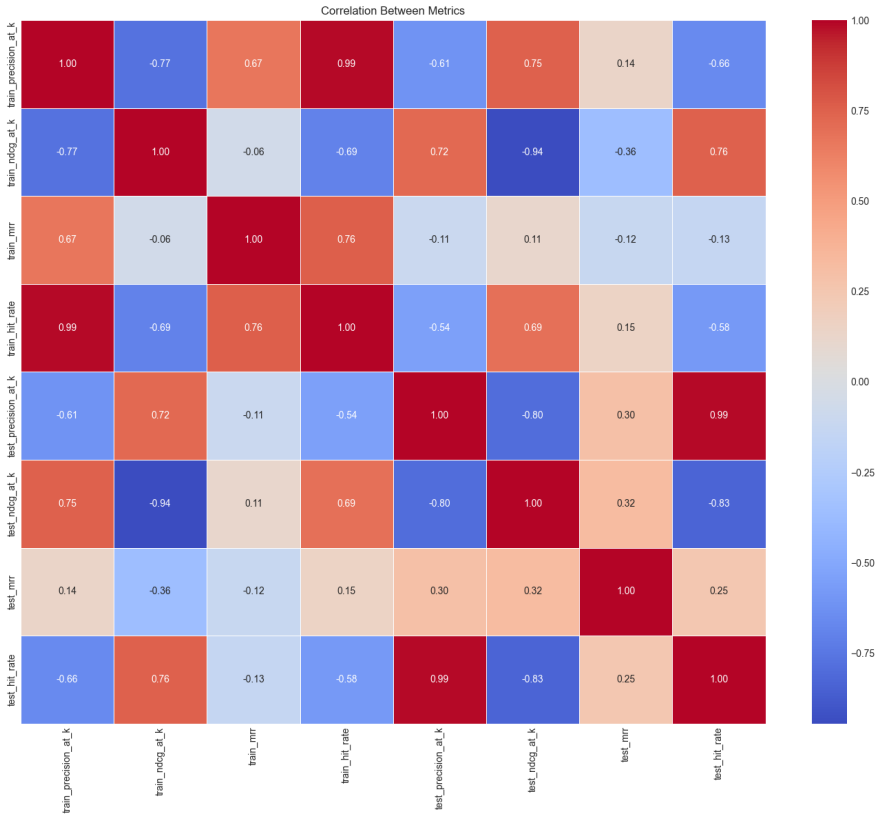

Figure 7: Metric Correlation Analysis: Relationships Between Revenue and Ranking Quality Measures across All Checkpoints