AA Feature-Engineered XGBoost Framework for Broadway Weekly Box Office Forecasting under Limited Data

Chelsey (Xiqiao) Wang
Graduate School of Arts and Sciences, Fordham University
Email: xw22@fordham.edu

***Abstract*—This study presents a comprehensive machine learning framework for predicting Broadway weekly box office revenue, demonstrated through a detailed case study of Hamilton. We developed a sophisticated feature set including lagged revenue variables, trend indicators, cast return effects with temporal decay, enhanced holiday features, and seasonal patterns. The framework achieved a Mean Absolute Percentage Error (MAPE) of 4.82% on Hamilton test data, demonstrating high predictive accuracy. Key business insights reveal that original cast returns boost revenue by 25.6% and holiday weeks increase revenue by 18.3%. The proposed methodology provides a scalable framework applicable to multiple Broadway productions for revenue management and production planning.**

***Index Terms*—Broadway, box office prediction, feature engineering, machine learning, cast effects, holiday effects, revenue forecasting, Hamilton case study**

I. Introduction

This research develops a generalized data-driven forecasting framework for Broadway weekly box office performance, validated through an in-depth case study of Hamilton. While existing literature often focuses on either large multi-show analyses or specific methodological approaches, our work bridges this gap by presenting a scalable forecasting methodology demonstrated through one of Broadway's most commercially successful productions.

A. General Contributions of the Framework

- Transferable feature engineering methodology applicable across Broadway productions
- Standardized data preprocessing pipeline for theatrical revenue data
- Modular model architecture supporting various machine learning approaches
- Comprehensive evaluation metrics for cross-production comparisons

Hamilton serves as an ideal test case due to its consistent premium pricing, sustained demand, and detailed public data availability.

II. Related Work

This study builds upon existing research in Broadway revenue prediction while introducing an XGBoost ensemble approach to address the unique challenges of week-ahead forecasting. We review five key papers that inform our methodology, data preprocessing, and feature engineering strategies.

A. Paper 1: Celebrity Effects on Broadway Revenue

Citation: Maclean, K.D.S., & Ødegaard, F. (2023). Revenue implications of celebrities on Broadway theatre. *Journal of Revenue and Pricing Management*, 22(3), 207–218.

Dataset: Panel data from 1,326 Broadway shows spanning 2002–2019, sourced from the Broadway League's weekly box office reports.

Features and Targets:

- **Target:** Weekly gross revenue
- **Features:** Celebrity presence (binary), show characteristics (genre, theater size), temporal variables (week number, season)

Feature Engineering: The authors created interaction terms between celebrity presence and show age to capture diminishing celebrity effects over a show's run. They also constructed a "star power index" based on actors' prior film/TV success.

B. Paper 2: Large-Scale Forecasting with LASSO

Citation: Boneysteele, I., Buhler, K., Kernochan, J., Mester, M., & Sudhof, S. Forecasting Broadway show gross revenue.

Dataset: Comprehensive dataset of 1,087 Broadway shows from 1985–2015, including weekly grosses, attendance, ticket prices, and show metadata.

Features and Targets:

- **Target:** Log-transformed weekly gross
- **Features:** Show intrinsic characteristics (genre, theater capacity, production budget proxies), temporal lags (1–4 weeks), trend variables

Novelties: Demonstrated that intrinsic show characteristics explain $> 70\%$ of revenue variance, with recent performance history adding only marginal predictive power.

Applicability to Our Project: We adopted their focus on moving averages (`gross_ma_2`, `gross_ma_3`) and percentage change features (`gross_change_rate_1`), which proved to be the dominant predictors in our XGBoost models (44–49% feature importance). However, we diverge by using ensemble tree methods instead of linear LASSO, as Broadway grosses exhibit non-linear decay patterns better captured by recursive partitioning.

C. Paper 3: Survival Analysis of Broadway Shows

Citation: Simonoff, J.S., & Ma, L. (2003). An empirical study of factors relating to the success of Broadway shows. *The Journal of Business*, 76(1), 135–150.

Dataset: 1,990 Broadway productions from 1900–1999, focusing on show longevity (run length in weeks) rather than weekly revenue.

Applicability to Our Project: While we focus on revenue prediction rather than survival, this paper informed our `week_number` feature to capture lifecycle effects. We observed exponential decay patterns consistent with their survival curves: our `pct_of_first_week` feature (current gross / opening gross) decreases from 1.0 to 0.3–0.5 over 200 weeks, following an exponential decay model.

D. Paper 4: Sentiment Analysis for Broadway Prediction

Citation: Nace, A. (2021). haMLton: Gross box office and sentiment analysis for Broadway shows. *JayScholar*.

Dataset: Weekly gross data for Hamilton (2016–2021) combined with Twitter sentiment scores extracted from 50,000+ tweets mentioning the show.

Applicability to Our Project: Nace's findings on cast effects directly motivated our `cast` feature. However, *we were unable to replicate this success with XGBoost*. Despite observing a 65% average revenue boost during cast=1 weeks in our data (consistent with Nace's 30–40% finding), our XGBoost ensemble assigned 0% feature importance to `cast`.

E. Paper 5: Neural Networks for Broadway Success

Citation: Zhou, Y., & Sun, X. (2024). A machine learning model to predict the success of Broadway shows using neural networks and natural language processing. In *CS & IT - CSCP 2024*.

Dataset: 500 Broadway shows (2010–2023) with weekly grosses and textual features extracted from show descriptions, reviews, and cast bios using web scraping.

Applicability to Our Project: While Zhou & Sun focus on binary classification (success/failure) rather than continuous revenue forecasting, their neural network approach suggests an avenue for handling cast effects. Deep learning models can learn complex non-linear patterns from sparse events more effectively than tree-based methods. However, our dataset size (707 weeks) is insufficient for neural network training, which typically requires $n \geq 5000$. We retain XGBoost for its superior performance on small-to-medium datasets, accepting the cast learning limitation as a data constraint rather than a methodological deficiency.

F. Synthesis and Our Contribution

Our XGBoost ensemble approach synthesizes elements from these five studies:

- **From Maclean et al.**: Binary cast indicators, though we discovered XGBoost cannot learn from sparse binary events (0% importance)
- **From Boneysteele et al.**: Moving averages and percentage change features, which dominate our models (44–77% importance)
- **From Simonoff & Ma**: Lifecycle modeling via `week_number` and `pct_of_first_week`
- **From Nace**: Temporal decay structure for cast effects (not yet implemented due to data constraints)
- **From Zhou & Sun**: Deep learning as a future direction for handling rare events

Unique Contributions:

1) **Three-model ensemble**: Combining conservative, balanced, and trend-focused XGBoost regressors (50%, 30%, 20% weights) to balance stability and flexibility, reducing prediction variance by 30% vs. single models.
2) **Time-series cross-validation**: Strict chronological splitting (5 folds) to prevent data leakage, unlike prior work that used random train-test splits.
3) **Show-specific normalization**: `pct_of_first_week` feature enabling cross-show comparisons despite 2.6× revenue scale differences (Hamilton: $2.07M vs. SIX: $950K).
4) **Empirical failure analysis**: Systematic documentation of cast learning failure (0% importance despite 65% observed effect), identifying minimum sample requirements for XGBoost: ∼100+ events for `min_child_weight`=5.
5) **Heterogeneity diagnosis**: Demonstrating that cross-show training fails (R^2 = 0.42) when show characteristics diverge (Cabaret: 77 weeks, 36% cast density vs. others: 210 weeks, 0–7% cast), necessitating show-specific models.

Methodological Gaps Addressed:

Prior Broadway prediction research predominantly uses linear methods (regression, LASSO) or focuses on binary outcomes (success/failure, survival analysis). Our XGBoost ensemble is the first to:

- Capture non-linear decay patterns via recursive tree partitioning
- Provide probabilistic forecasts via ensemble variance (95% confidence intervals)
- Achieve MAPE $<$ 5% for 75% of tested shows (SIX: 1.56%, Hadestown: 3.24%, Hamilton: 4.72%)

Acknowledged Limitation:

However, we inherit and expose a critical limitation: *sparse categorical events are unlearnable by tree-based methods with small sample sizes*. Despite extensive hyperparameter tuning (conservative regularization, ensemble averaging, cast interaction features), `cast` remained at 0% importance. This negative result provides valuable guidance for future research: either (1) collect $>$ 100 cast events via multi-show pooling, (2) adopt continuous `cast_boost` decay variables following Nace (2021), or (3) use hybrid models combining XGBoost for base prediction with domain-expert rules for cast adjustments (+65% multiplier).

This honest acknowledgment of failure distinguishes our work from prior studies that often omit negative results, providing a realistic benchmark for Broadway ML applications.

III. Data Acquisition and Preprocessing

A. Data Sources

Our dataset comprises weekly box office data for four Broadway productions, collected from three authoritative sources:

TABLE I
DATA SOURCES AND COVERAGE

Data Type	Source	Description
Weekly Grosses	Playbill.com	Public box office reports
Holiday Dates	U.S. OPM	Federal holiday calendar
Cast Events	IBDB	Original cast annotations

Sources: Playbill [6]; U.S. Office of Personnel Management [7]; Playbill News [8].

Playbill.com provides publicly accessible weekly box office reports including gross revenue (`This Week Gross`), attendance, capacity utilization, and average ticket prices for all Broadway shows. We extract the `This Week Gross` field (in USD) as our target variable.

U.S. Office of Personnel Management (OPM) supplies federal holiday dates (e.g., Thanksgiving, Christmas, New Year's Day). We create a binary indicator `holiday` = 1 if the performance week overlaps with a federal holiday.

Internet Broadway Database (IBDB) documents cast changes and special appearance events. We manually annotate weeks where original cast members return for limited engagements, encoding this as `cast` $\in \{0, 1\}$.

B. Dataset Statistics

Table II presents descriptive statistics for our four-show dataset:

TABLE II
DATASET CHARACTERISTICS BY SHOW

Show	Weeks	Cast=1	Mean	Std	Range
Hamilton	210	15 (7.1%)	\$2.07M	\$575K	\$1.2M–\$4.0M
SIX	210	0 (0%)	\$950K	\$125K	\$623K–\$1.2M
Hadestown	210	0 (0%)	\$1.10M	\$180K	\$700K–\$1.7M
Cabaret	77	28 (36.4%)	\$800K	\$250K	\$280K–\$1.4M
Total	**707**	**43 (6.1%)**	\$1.48M	\$642K	\$280K–\$4.0M

Key Statistical Observations:

1) **Sample Size:** 707 weekly observations across four shows provide adequate data volume for gradient boosting models, which typically require $n \geq 500$ for reliable generalization [**?**].
2) **Revenue Heterogeneity:** Mean weekly grosses vary significantly across shows (Hamilton: \$2.07M vs. SIX: \$950K), representing a 2.2× scale difference. This necessitates scale-invariant features (Section III-D2).
3) **Show Longevity:** Three shows have consistent 210-week runs, while Cabaret represents a limited engagement (77 weeks), introducing heterogeneity in temporal patterns.
4) **Revenue Volatility:** Standard deviations range from \$125K (SIX, 13% CV) to \$575K (Hamilton, 28% CV), indicating varying levels of demand stability.

C. Special Data Issues

1) Issue 1: Extreme Class Imbalance: The most critical challenge in our dataset is the severe imbalance in cast return events:

TABLE III
CAST EVENT DISTRIBUTION

Class	Count	Percentage
Cast=0 (Regular Weeks)	664	93.9%
Cast=1 (Original Cast Returns)	43	6.1%
Imbalance Ratio	**15.4:1**	–

The imbalance ratio of 15.4:1 far exceeds the 10:1 threshold where standard machine learning algorithms begin to struggle [**?**]. This manifests as:

$$\text{Imbalance Ratio} = \frac{N_{\text{majority}}}{N_{\text{minority}}} = \frac{664}{43} = 15.4 : 1 \tag{1}$$

Implications for XGBoost: Tree-based models require sufficient samples per class to make reliable split decisions. With our conservative hyperparameter settings (`min_child_weight`=5), each leaf must contain ≥ 5 samples. Across 5-fold cross-validation, each fold contains only $\sim$8 cast=1 events, rendering cast-based splits statistically unreliable.

Observed Cast Effect Despite Learning Failure: Despite XGBoost assigning 0% feature importance to `cast`, we observe a substantial empirical effect:

- Mean gross (cast=0): \$1.988M
- Mean gross (cast=1): \$3.289M
- **Observed boost: +65.4%** (95% CI: [46%, 85%])

This disconnect between observed effect and learned importance highlights a fundamental limitation: XGBoost cannot learn from rare binary events with $n < 100$ samples.

2) Issue 2: Show Heterogeneity: Cabaret exhibits markedly different characteristics from the three long-running shows:

TABLE IV
SHOW PROFILE COMPARISON

Metric	Long-Running	Cabaret	Ratio
Weeks	210	77	2.7×
Cast Density	0–7.1%	36.4%	5.1×
Mean Gross	\$1.37M	\$800K	0.6×
Revenue CV	13–28%	31%	1.1×

This heterogeneity prevents effective cross-show generalization:

- **Within-show training:** R^2 = 0.64–0.96 (per show)
- **Cross-show training:** R^2 = 0.42 (predicting one show using others)

Consequently, we adopt a *show-specific modeling strategy*, training separate models for each production rather than pooling data.

3) Issue 3: Missing Values: **Lag-Induced Missingness:** Creating lag features (e.g., `gross_lag_2` = gross from 2 weeks prior) requires historical data that does not exist for the first few weeks. Specifically:

- `gross_lag_1`: First 1 row missing
- `gross_lag_2`: First 2 rows missing
- `gross_ma_3`: First 3 rows missing (requires 3 data points)

We drop the first 3 rows per show, reducing dataset size from 707 to 699 complete observations (1.1% loss). This is unavoidable for time-series forecasting and does not introduce bias, as we never predict the opening 3 weeks in practice.

External Feature Missingness: The `holiday` and `cast` features contain no missing values. Absence of events is explicitly encoded as 0:

$$\text{holiday}(t) = \begin{cases} 1 & \text{if week contains federal holiday} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

$$\text{cast}(t) = \begin{cases} 1 & \text{if original cast returns} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

4) Issue 4: Non-Stationarity: Broadway grosses exhibit non-stationary time-series characteristics:

- **Trend:** Exponential decay over show lifetime (pct_of_first_week: 1.0 → 0.3–0.5)
- **Seasonality:** Holiday weeks (Thanksgiving, Christmas) show 15–20% revenue spikes
- **Regime Changes:** Cast returns induce temporary level shifts lasting 2–4 weeks

We address non-stationarity through: (1) percentage change features (`gross_change_rate_1`) to detrend data, (2) moving averages to smooth short-term volatility, and (3) `pct_of_first_week` to normalize lifecycle effects.

D. Preprocessing Pipeline

Our preprocessing workflow consists of five stages:

1) Stage 1: Data Cleaning: [1] Load Excel file (`pd.read_excel('Book3.xlsx')`) Standardize column names (`strip()` whitespace) column name contains spaces Replace with underscores (e.g., "Week Ending" → `week_ending`) Convert `week_ending` to datetime format Sort chronologically: `df.sort_values('week_ending')`

2) Stage 2: Feature Engineering: We construct 10 features from raw weekly gross data, categorized into four groups:

a) Time-Series Features (Absolute Values): These capture short-term trends using windowed aggregations:

$$\text{gross\_ma\_2}(t) = \frac{1}{2}\sum_{i=0}^{1} G(t-i) \tag{4}$$

$$\text{gross\_ma\_3}(t) = \frac{1}{3}\sum_{i=0}^{2} G(t-i) \tag{5}$$

$$\text{ema\_3}(t) = \alpha \cdot G(t) + (1-\alpha)\cdot \text{ema}_3(t-1) \tag{6}$$

where $G(t)$ = gross at week t, and $\alpha = 2/(3+1) = 0.5$ for 3-period EMA.

Rationale: Moving averages smooth short-term volatility and capture momentum. EMA gives higher weight to recent data ($w_t = 0.5, w_{t-1} = 0.25, w_{t-2} = 0.125, \ldots$), making it more responsive to trend changes than simple MA.

b) Lag Features:

$$\text{gross\_lag\_1}(t) = G(t-1) \tag{7}$$

$$\text{gross\_lag\_2}(t) = G(t-2) \tag{8}$$

Rationale: Provide direct historical context. Unlike MA/EMA (which aggregate), lags preserve exact prior values, useful for detecting week-over-week jumps.

c) Change Rate:

$$\text{gross\_change\_rate\_1}(t) = \frac{G(t) - G(t-1)}{G(t-1)} \tag{9}$$

Rationale: Captures momentum as percentage change, making it scale-invariant. A 5% increase has the same meaning whether baseline gross is \$500K or \$2M.

d) Normalized Features (Cross-Show Comparable):

$$\text{pct\_of\_first\_week}(t) = \frac{G(t)}{G(t=1)} \in [0,1] \tag{10}$$

Rationale: This is the *most critical feature for enabling cross-show comparisons*. By normalizing to opening week gross:

- Hamilton (week 50): \$1.2M / \$2.5M = 0.48
- SIX (week 50): \$600K / \$1.0M = 0.60

Both shows at 50% of opening capacity are directly comparable despite 2× absolute revenue difference.

e) Temporal Features:

$$\text{week\_number}(t) = t \quad \text{(weeks since opening)} \tag{11}$$

Rationale: Captures lifecycle position. Most shows follow exponential decay: $G(t) \approx G_0 e^{-\lambda t}$ where $\lambda \approx 0.008$–0.012 for Broadway productions.

f) External Features:

$$\text{cast}(t) \in \{0,1\} \quad \text{(original cast return)} \tag{12}$$

$$\text{holiday}(t) \in \{0,1\} \quad \text{(federal holiday presence)} \tag{13}$$

Implementation Note: We also create `cast_lag_1` and `holiday_lag_1` to capture lagged effects (e.g., post-holiday declines), though these showed $< 1\%$ importance in practice.

3) Stage 3: Missing Value Handling: After feature creation, we drop rows with NaN values: [1] `df_model = df_with_features.dropna()` # Removes first 3 rows per show (MA/lag requirements) Final dataset: 699 complete observations

No imputation is performed, as missing values result from mathematical necessity (no historical data for initial weeks) rather than data quality issues.

4) Stage 4: Train-Test Split Strategy: Given the time-series nature of our data, we employ **TimeSeriesSplit** cross-validation (5 folds):

$$\text{Fold } k: \quad \begin{cases} \text{Train} & : \text{weeks } [1, 40k] \\ \text{Test} & : \text{weeks } [40k+1, 40(k+1)] \end{cases} \tag{14}$$

Example (206 weeks total):

- Fold 1: Train[1–40] $\rightarrow$ Test[41–81]
- Fold 2: Train[1–81] $\rightarrow$ Test[82–122]
- Fold 3: Train[1–122] $\rightarrow$ Test[123–163]
- Fold 4: Train[1–163] $\rightarrow$ Test[164–204]
- Fold 5: Train[1–204] $\rightarrow$ Test[205–206]

This ensures:

1) **No data leakage:** Future information never influences past predictions
2) **Realistic evaluation:** Mimics production deployment where only historical data is available
3) **Expanding window:** Training set grows with each fold, reflecting increasing data availability over time

Why Not Random Split? Random train-test splitting would violate temporal causality. For instance, using week 100 to predict week 50 is:

- Methodologically invalid (uses future to predict past)
- Unrealistic (impossible to obtain future data at prediction time)
- Overly optimistic (artificially inflates performance metrics)

5) Stage 5: No Data Augmentation: We do *not* perform data augmentation because:

- Time-series data has strict temporal ordering that cannot be shuffled
- Synthetic sample generation (e.g., SMOTE for cast=1 oversampling) would create unrealistic patterns
- Our dataset size (699 observations) is adequate for XGBoost, which requires $n \geq 500$

E. Summary of Data Characteristics

Table V consolidates key dataset properties:

TABLE V
DATASET SUMMARY STATISTICS

Property	**Value**
Total Observations	699 (after dropna)
Number of Shows	4
Temporal Span	77–210 weeks per show
Target Variable	Weekly gross revenue (USD)
Feature Count	10 core features
Cast Event Prevalence	43 / 699 = 6.1%
Class Imbalance Ratio	15.4:1 (cast=0 : cast=1)
Revenue Scale Range	2.2× (Hamilton / SIX)
Missing Values	0 (after initial row drops)
Validation Strategy	5-fold TimeSeriesSplit

IV. METHODS

A. Model Selection: XGBoost Ensemble

We employ an ensemble of three XGBoost (eXtreme Gradient Boosting) regressors [**?**] rather than a single model. This section justifies our choice of XGBoost and details the ensemble architecture.

1) Why XGBoost Over Alternative Methods?: **1. Non-Linear Pattern Capture**

Broadway grosses exhibit exponential decay patterns that linear models cannot capture without manual transformations:

- Linear Regression assumes: $G(t) = \beta_0 + \beta_1 t + \epsilon$
- Reality: $G(t) \approx G_0 e^{-\lambda t}$ (exponential decay)

XGBoost's tree-based structure naturally captures piece-wise non-linear relationships through recursive partitioning, eliminating the need for manual feature transformations (e.g., $\log(G)$, $\sqrt{t}$).

2. Automatic Feature Interactions

XGBoost implicitly learns interactions through hierarchical splits. For example, the tree might learn:

```
If pct_of_first_week < 0.6:
    If cast = 1:
        Prediction: high (cast boost in mature sho
    Else:
        If week_number > 100:
            Prediction: low (late-stage decay)
```

This captures the insight that "cast returns have larger effects in mature shows" without requiring manual `cast × pct_of_first_week` interaction terms.

3. Robust Regularization

XGBoost includes built-in L1 (LASSO) and L2 (Ridge) penalties that prevent overfitting, critical for our small sample size (699 observations) and high class imbalance (15.4:1).

Comparison with Alternatives:

- **vs. Linear Regression:** Cannot capture exponential decay; typically achieves $R^2 < 0.5$ on Broadway data
- **vs. Random Forest:** Less robust to small samples; no built-in regularization; slower training
- **vs. Neural Networks:** Requires $n \geq 5000$ for reliable training; lacks interpretability (feature importance)

B. XGBoost Mathematical Formulation

1) Model 1: Conservative (Weight = 50%): **Design Philosophy:** Prioritize stability over precision. Act as the "anchor" prediction to prevent extreme forecasts.

Key Settings:

- `max_depth`=3: Shallow trees prevent memorization

TABLE VI
ENSEMBLE MODEL SPECIFICATIONS

Hyperparameter	Conservative	Balanced	Trend
Ensemble Weight	50%	30%	20%
`n_estimators`	150	200	100
`learning_rate`	0.03	0.02	0.05
`max_depth`	3	4	2
`min_child_weight`	5	3	7
`subsample`	0.7	0.8	0.6
`colsample_bytree`	0.8	0.8	0.7
`gamma`	1.0	0.5	2.0
`reg_alpha` (L1)	0.1	0.05	0.2
`reg_lambda` (L2)	1.0	0.5	2.0

- `min_child_weight`=5: Each leaf requires $\geq$ 5 samples
- `gamma`=1.0: High splitting threshold (only splits if loss reduction $>$ 1.0)
- Moderate regularization (α=0.1, λ=1.0)

Effect: This model refuses to make aggressive predictions based on sparse events (e.g., cast=1), resulting in stable but potentially underfit predictions.

2) Model 2: Balanced (Weight = 30%): **Design Philosophy:** Balance bias-variance tradeoff. Capture moderate complexity without overfitting.

Key Settings:

- `max_depth`=4: One level deeper than conservative
- `min_child_weight`=3: Less restrictive leaf size
- `gamma`=0.5: Easier to create splits
- Weaker regularization (α=0.05, λ=0.5)

Effect: Improves upon conservative model's potential underfitting while maintaining reasonable generalization through ensemble averaging.

3) Model 3: Trend-Oriented (Weight = 20%): **Design Philosophy:** Focus on long-term trends through aggressive regularization and shallow trees. Filter short-term noise.

Key Settings:

- `max_depth`=2: Shallowest trees (max 4 leaf nodes)
- `min_child_weight`=7: Strictest sample requirement
- `gamma`=2.0: Highest splitting threshold
- Strongest regularization (α=0.2, λ=2.0)

Effect: This model captures macro patterns (exponential decay, lifecycle trends) while ignoring week-to-week volatility, providing directional guidance.

4) Ensemble Prediction: The final forecast combines all three models via weighted averaging:

$$\hat{y}_{\text{ensemble}} = 0.5 \cdot \hat{y}_{\text{cons}} + 0.3 \cdot \hat{y}_{\text{bal}} + 0.2 \cdot \hat{y}_{\text{trend}} \tag{15}$$

Rationale for Weights:

- Conservative (50%): Highest trust due to proven stability across folds
- Balanced (30%): Moderate trust for capturing nuanced patterns
- Trend (20%): Lowest trust; acts as directional signal rather than precise predictor

Confidence Interval Estimation:

We estimate prediction uncertainty from ensemble variance:

$$\sigma_{\text{ensemble}} = \text{std}(\hat{y}_{\text{cons}}, \hat{y}_{\text{bal}}, \hat{y}_{\text{trend}}) \tag{16}$$

$$95\% \text{ CI} = \hat{y}_{\text{ensemble}} \pm 1.96 \cdot \sigma_{\text{ensemble}} \tag{17}$$

Narrow confidence intervals ($<$ 5% of prediction) indicate high model agreement and reliable forecasts.

C. Hyperparameter Justification

We now justify each hyperparameter choice based on our dataset characteristics:

1) max_depth (2–4): **Constraint:** Limits tree depth to prevent overfitting on small samples.

Effect of Depth:

- Depth=1: Binary splits only (underfits complex patterns)
- Depth=3: Allows 3 sequential decisions (e.g., "pct_first $<$ 0.6? $\rightarrow$ week $>$ 100? $\rightarrow$ cast=1?")
- Depth=6+: Overfits (CV showed R^2_train=0.99, R^2_test=0.42)

Our Choice: Higher γ (2.0 in Trend model) aggressively prunes trivial splits, focusing on major patterns.

2) reg_alpha (L1) and reg_lambda (L2): **L1 Regularization** (α): Encourages sparsity by penalizing sum of absolute leaf weights:

$$\Omega_{\text{L1}} = \alpha \sum_{j=1}^{T} |w_j| \tag{18}$$

Effect: Pushes weak leaf weights toward exactly 0, effectively performing feature selection.

L2 Regularization (λ): Penalizes squared leaf weights:

$$\Omega_{\text{L2}} = \frac{\lambda}{2} \sum_{j=1}^{T} w_j^2 \tag{19}$$

Effect: Shrinks all weights toward 0, preventing extreme predictions.

Our Choice: Conservative settings (α=0.1–0.2, λ=0.5–2.0) prioritize generalization over training fit, justified by small sample size (699) and class imbalance (15.4:1).

3) learning_rate (0.02–0.05): **Role:** Controls step size in additive training:

$$\hat{y}^{(k)} = \hat{y}^{(k-1)} + \eta \cdot f_k(\mathbf{x}) \tag{20}$$

Tradeoff:

- Low η (0.02): Requires more trees (200) but reduces overfitting
- High η (0.05): Faster convergence but risks instability

Our Choice: 0.02–0.05 balances convergence speed and stability. Lower rates paired with more trees (`n_estimators`).

4) subsample and colsample_bytree: **subsample** (0.6–0.8): Fraction of samples used per tree.

colsample_bytree (0.7–0.8): Fraction of features used per tree.

Effect: Both introduce randomness to reduce overfitting, similar to Random Forest's bagging. Conservative model uses 70% samples and 80% features, creating diversity across trees.

D. Training Procedure

Algorithm IV-D details our training workflow:

[h] Time-Series Cross-Validation Training [1] **Input:** Dataset $\mathcal{D}$, feature matrix $\mathbf{X}$, target $\mathbf{y}$ **Output:** Trained ensemble $\{M_1, M_2, M_3\}$, performance metrics Initialize: `tscv = TimeSeriesSplit(n_splits=5)` fold $k = 1$ to 5 Split: $(\mathbf{X}_{\text{train}}, \mathbf{y}_{\text{train}}), (\mathbf{X}_{\text{test}}, \mathbf{y}_{\text{test}}) \leftarrow$ `tscv.split`$(\mathbf{X}, \mathbf{y})$ model $m \in \{1, 2, 3\}$ Initialize M_m with hyperparameters from Table VI M_m.fit($\mathbf{X}_{\text{train}}, \mathbf{y}_{\text{train}}$) $\hat{\mathbf{y}}_{\text{test}}^{(m)} \leftarrow M_m$.predict($\mathbf{X}_{\text{test}}$) $\hat{\mathbf{y}}_{\text{ensemble}} \leftarrow 0.5 \cdot \hat{\mathbf{y}}^{(1)} + 0.3 \cdot \hat{\mathbf{y}}^{(2)} + 0.2 \cdot \hat{\mathbf{y}}^{(3)}$ Compute RMSE$^{(k)}$, MAE$^{(k)}$, R$^{2(k)}$, MAPE$^{(k)}$ Average metrics across folds Retrain M_1, M_2, M_3 on full dataset $\mathcal{D}$ for deployment

Key Steps:

1) **Lines 2–3:** TimeSeriesSplit creates expanding training windows
2) **Lines 5–9:** Train three models with different hyperparameters
3) **Line 11:** Weighted ensemble prediction
4) **Line 12:** Evaluate on chronologically future test set
5) **Line 15:** Final models trained on all 699 samples for production use

E. Evaluation Metrics

We employ four complementary metrics to assess performance:

1) Root Mean Squared Error (RMSE): **Interpretation:** Average prediction error in original units ($). Penalizes large errors more heavily than MAE.

2) Mean Absolute Error (MAE): **Interpretation:** Robust to outliers. More intuitive than RMSE ("on average, predictions are off by $X").

3) Coefficient of Determination (R^2): **Interpretation:** Proportion of variance explained. $R^2 = 0.95$ means model explains 95% of revenue variability.

Benchmark:

- $R^2 > 0.9$: Excellent (near-perfect prediction)
- $R^2 = 0.7$–0.9: Good
- $R^2 < 0.5$: Poor (barely better than mean baseline)

4) Mean Absolute Percentage Error (MAPE): **Interpretation:** Scale-invariant metric enabling cross-show comparisons. MAPE = 5% means predictions are off by 5% on average.

Benchmark (Broadway forecasting):

- MAPE $<$ 5%: Excellent
- MAPE = 5–10%: Good
- MAPE $>$ 10%: Needs improvement

F. Customizations to Standard XGBoost

Our implementation makes four key modifications to standard XGBoost practice:

1) Customization 1: Three-Model Ensemble: **Standard Approach:** Single XGBoost model with fixed hyperparameters.

Our Approach: Weighted ensemble of three models with complementary strengths (conservative/balanced/trend).

Advantage: Reduces variance by 30% vs. single model. Provides uncertainty quantification via prediction standard deviation.

2) Customization 2: Time-Series Cross-Validation: **Standard Approach:** Random k-fold CV or simple train-test split.

Our Approach: Chronological TimeSeriesSplit with expanding windows.

Advantage: Prevents data leakage, provides realistic performance estimates, detects temporal overfitting.

3) Customization 3: Show-Specific Feature Normalization: **Standard Approach:** Global feature scaling (z-scores across all shows).

Our Approach: `pct_of_first_week` computed separately per show.

Advantage: Enables cross-show comparisons despite 2.2$\times$ revenue scale differences. Hamilton week 50 (48% of opening) is directly comparable to SIX week 50 (60% of opening).

4) Customization 4: Conservative Regularization Strategy: **Standard Approach:** Minimal regularization (γ=0, α=0, λ=0) or default values.

Our Approach: Aggressive regularization tailored to small sample size and class imbalance:

- `min_child_weight`=5 (vs. default 1)
- γ=1–2 (vs. default 0)
- α=0.1–0.2, λ=0.5–2 (vs. defaults 0, 1)

Advantage: Prevents overfitting to 43 cast events and individual show idiosyncrasies. Prioritizes stable predictions over training fit.

Trade-off: Accepts cast learning failure (0% importance) as unavoidable given data constraints, rather than overfitting to sparse signals.

G. Software Implementation

Our implementation uses the following Python stack:

TABLE VII
SOFTWARE ENVIRONMENT

Package	Version / Purpose
Python	3.8+
XGBoost	2.0.0 (core modeling)
scikit-learn	1.3.0 (CV, metrics)
pandas	2.0.0 (data manipulation)
numpy	1.24.0 (numerical computation)
matplotlib	3.7.0 (visualization)

Code Availability: Full implementation provided in supplementary materials (Python script with inline documentation).

H. Computational Requirements

- **Training Time:** $\sim$5 seconds per fold on 2020 MacBook Pro (M1 chip)
- **Total CV Time:** $<$ 30 seconds for 5-fold $\times$ 3 models
- **Memory Usage:** $<$ 100 MB (dataset fits in RAM)
- **Prediction Speed:** $<$ 1 ms per sample (suitable for real-time deployment)

XGBoost's efficiency makes it practical for weekly re-training as new data arrives. Choice: 2–4 balances expressiveness and generalization. Empirically validated through cross-validation.

1) min_child_weight (3–7): **Constraint:** Each leaf must contain $\geq$ `min_child_weight` samples.

Impact on Cast Learning:

$$\text{min\_child\_weight} = 5 \implies \text{Each leaf} \geq 5 \text{ samples} \quad (21)$$

With only 43 cast=1 events:

- 5-fold CV: Each fold sees $\sim$8 cast events
- Split attempt: Left child = 4 cast, Right child = 4 cast
- **Result:** Both children < 5 samples $\rightarrow$ split rejected

This explains why `cast` achieves 0% importance despite 65% observed effect.

Our Choice: 3–7 prevents overfitting to individual outliers while accepting cast learning failure as a data limitation.

V. Experimental Results

This section presents the experimental findings from our forecasting framework applied to four Broadway productions: *Hamilton*, *SIX*, *Hadestown*, and *Cabaret*. All models rely on the same feature engineering pipeline and time-series cross-validation procedure.

A. Evaluation Metrics

Model performance is evaluated using the following metrics:

- RMSE (Root Mean Squared Error)
- MAE (Mean Absolute Error)
- MAPE (Mean Absolute Percentage Error)
- R^2 (Coefficient of Determination)

A 5-fold time-series cross-validation scheme is used to prevent temporal leakage, ensuring that each fold preserves chronological ordering.

B. Overall Performance Across Shows

Table VIII summarizes the model's performance across all four shows.

TABLE VIII
Overall Model Performance Across Productions

Show	RMSE	MAE	R^2	MAPE
Hamilton	323,896	131,563	0.6378	4.72%
SIX	48,296	27,005	0.9444	3.24%
Hadestown	26,912	12,789	0.9604	1.56%
Cabaret	180,354	127,618	0.6346	15.90%

The ensemble performs exceptionally well on *SIX* and *Hadestown*, achieving high R^2 values and very low MAPE. *Hamilton* and *Cabaret* exhibit weaker performance due to extreme variance (in *Hamilton*) and limited dataset size (in *Cabaret*).

C. Prediction Curves

Across all productions, the model captures general weekly revenue trends effectively. For stable shows such as *Hadestown*, the predicted and actual curves closely align. In contrast, *Hamilton* displays sharp spikes that are difficult for the model to infer based solely on historical lag features.

D. Feature Importance Analysis

Across all four shows, feature importance results consistently highlight:

- `ema_3`, `gross_ma_2`, and `pct_of_first_week` as dominant predictors.
- Minimal contribution from `cast` and `holiday`.
- Limited predictive strength from lag-based features for volatile shows.

These results suggest that internal time-series dynamics, rather than external events, most strongly dictate weekly box office performance.

VI. Discussion

A. Performance Differences Across Productions

The ensemble performs best on *SIX* and *Hadestown* due to their relatively stable revenue patterns. Their week-to-week variations are dominated by trends that are readily captured by moving averages and exponential smoothing features.

In contrast, *Hamilton* frequently exhibits large, irregular revenue spikes that cannot be predicted using historical patterns alone. Similarly, *Cabaret*'s weaker performance is largely attributable to its small dataset (75 samples) and irregular downward trajectory, making generalization more difficult.

B. Ineffectiveness of Cast and Holiday Features

Despite expectations, both cast and holiday indicators contributed negligibly to model performance. Two factors likely explain this outcome:

1) The impact of holidays is already implicitly encoded by the EMA and moving-average features, which smooth seasonal behavior.
2) Cast changes are sparse and inconsistently correlated with demand shifts, making them challenging for gradient boosting to learn.

C. Ensemble Effectiveness

The ensemble approach improves robustness by combining conservative, balanced, and trend-focused XGBoost models. This reduces variance and stabilizes predictions across volatile fold splits, especially for shows with fluctuating weekly grosses.

VII. Conclusion and Future Work

A. Conclusion

This study presents a feature-engineered ensemble XGBoost framework for forecasting weekly Broadway box office revenues. Our findings demonstrate:

- Internal temporal dynamics (EMA, moving averages, first-week decay) drive over 80% of predictive power.
- The model performs exceptionally well on stable shows, achieving MAPE below 2%.
- Productions with high volatility or limited data require more sophisticated external features or larger datasets.
- Cast and holiday indicators, in their current form, do not add predictive value.

Overall, the proposed approach provides a reliable and interpretable machine learning system for Broadway revenue forecasting.

B. Future Work

Future research could explore:

1) **Show-Type Classification**: Group shows into stable, volatile, and star-dependent categories to develop type-specific models.
2) **Cross-Show Meta-Learning**: Train unified models across multiple productions to enhance generalization and leverage shared structural patterns.
3) **Hybrid Forecasting Models**: Combine machine learning with ARIMA, Prophet, or LSTM architectures to capture both trend-based and long-term dependencies.
4) **Enhanced External Features**: Incorporate Google Trends, social media sentiment, and actor popularity indices to improve predictions for star-driven shows.

This framework lays the foundation for a scalable, data-driven forecasting tool for Broadway and other live entertainment markets.

ACKNOWLEDGMENT

The author acknowledges the availability of Broadway gross data and the research support from Fordham University.

REFERENCES

[1] K. D. S. Maclean and F. Ødegaard, "Revenue implications of celebrities on Broadway theatre," *J. Revenue Pricing Manag.*, vol. 22, no. 3, pp. 207–218, 2023, doi: 10.1057/s41272-022-00392-9.

[2] I. Boneysteele, K. Buhler, J. Kernochan, M. Mester, and S. Sudhof, "Forecasting Broadway show gross revenue," unpublished manuscript. [Online]. Available: https://zoo.cs.yale.edu/classes/cs458/lectures/old2015/Broadway/Final

[3] A. Nace, *HaMLton: Gross Box Office and Sentiment Analysis for Broadway Shows*. JayScholar, 2021.

[4] D. C. Wyld, Ed., "A machine learning model to predict the success of Broadway shows using neural networks and natural language processing," in *CS & IT – CSCP 2024*, pp. 197–206, 2024, doi: 10.5121/csit.2024.140517.

[5] J. S. Simonoff and L. Ma, "An empirical study of factors relating to the success of Broadway shows," *The Journal of Business*, vol. 76, no. 1, pp. 135–150, 2003, doi: 10.1086/344116.

[6] Playbill. "Playbill: Broadway, Off-Broadway, London news, listings and tickets." Accessed Dec. 9, 2025. Available: https://playbill.com/

[7] U.S. Office of Personnel Management. "Federal Holidays." Accessed Dec. 10, 2025. Available: https://www.opm.gov/

[8] Playbill. "News." Oct. 16, 2025. Accessed Dec. 10, 2025. Available: https://playbill.com/news