Steam Game Market Value Analysis: Predicting Game Prices Using Pre-Release Metadata

CENG463 - Machine Learning

Nurten Çiftçioğlu, Muhammed Enes Uğraş

05.01.2026

Abstract

This project addresses the problem of predicting Steam game market prices using only pre-release metadata, providing developers with data-driven pricing guidance for upcoming releases. We utilized a comprehensive Steam games dataset containing approximately 111,000 games with 39 features. Our preprocessing pipeline included removal of free games and outliers, discount correction to recover base prices, feature engineering for categorical variables, and log transformation of the target variable to address severe skewness. We evaluated multiple machine learning algorithms including Linear Regression, Ridge, Lasso, Random Forest, XGBoost, and LightGBM, with model selection based on cross-validation performance using TimeSeriesSplit to respect temporal ordering. The final LightGBM model with regularization achieved validation MAE of $4.40 and R^2 of 0.266 during cross-validation. On the held-out test set, the model achieved MAE of $4.58 and R^2 of 0.17. Performance analysis revealed that degradation was primarily attributed to distribution shift, with the test period containing proportionally more premium-priced games above $150 than the training period. When excluding these extreme outliers, test R^2 improved to 0.367, demonstrating reasonable predictive capability for typical game prices. The results indicate that pre-release metadata provides useful pricing signals through features like genre, platform support, and localization effort. However, substantial variance remains unexplained, reflecting unmeasured factors such as brand recognition, marketing budgets, and perceived quality that fundamentally influence game pricing decisions.

1. Introduction

The video game industry has experienced tremendous growth over the past two decades, with Steam emerging as the dominant digital distribution platform for PC games. As of 2024, Steam hosts over 100,000 games spanning every conceivable genre, price point, and quality level. Pricing decisions significantly impact a game's commercial success, influencing both revenue generation and player acquisition. However, developers, particularly independent studios and first-time creators, often lack data-driven tools to determine appropriate price points for their games. This project addresses the challenge of predicting a game's typical market list price using only information available before release. The problem is both relevant and challenging for several reasons. First, game pricing depends on quantifiable features such as genre, platform support, and language localization, but also on subjective factors like brand recognition, perceived quality, and community anticipation that are difficult to capture from metadata alone. Second, the Steam marketplace exhibits extreme variance in pricing strategies, ranging from $0.99 experimental indie games to $60 AAA blockbuster titles, with specialized products exceeding $200. Third, temporal dynamics in gaming trends create distribution shifts between historical and future releases, as market saturation, genre popularity, and competitive dynamics evolve continuously. Our approach frames the problem as a regression task: given a game's pre-release metadata including genres, categories, user tags, platform support, localization details, and release timing, predict the typical price for similar Steam games. This formulation enables developers to receive data-driven pricing guidance based on patterns learned from tens of thousands of historical releases, providing an empirical foundation for pricing decisions that might otherwise rely purely on intuition. We employ a comprehensive methodology that addresses the unique challenges of this problem. Exploratory data analysis reveals the distributions, relationships, and temporal patterns in Steam game data. Careful feature engineering handles multi-label categorical variables like genres and tags, which present high dimensionality and sparsity challenges. Time-based train-test splitting simulates realistic deployment scenarios where models trained on historical games predict prices for future releases, avoiding optimistically biased estimates from random splitting. Systematic comparison of multiple regression algorithms identifies the most effective approach, and hyperparameter optimization with explicit attention to overfitting ensures robust generalization. The contributions of this work include: a comprehensive preprocessing pipeline for Steam game data that handles discount corrections, removes post-release leakage, and encodes complex multi-label

features; systematic evaluation of 13 different regression algorithms spanning linear models, ensemble methods, and baseline approaches; multi-stage hyperparameter optimization that balances validation performance with generalization stability; detailed error analysis revealing that apparent performance degradation primarily reflects distribution shift rather than model failure; and practical insights about the limitations of metadata-based pricing prediction and the unmeasured factors that drive real-world pricing decisions.

This report is organized as follows. Section 2 reviews related work in Steam game analytics. Section 3 describes the algorithms evaluated and our methodological approach to data preparation, feature engineering, and model training. Section 4 details the experimental setup including dataset characteristics, preprocessing steps, and evaluation strategy. Section 5 presents experimental results including model comparison, hyperparameter tuning, test set evaluation, and comprehensive error analysis. Section 6 concludes with discussion of strengths and weaknesses, lessons learned, and promising directions for future work.

2. Background and Related Work

Machine learning approaches to Steam game analytics have gained significant attention in recent years, with researchers applying various classical ML techniques to predict different aspects of game performance. This section reviews relevant academic papers and related work, highlighting their methodologies, findings, and how they relate to our price prediction task. While the studies reviewed below utilize data from the Steam platform, they employ different datasets, time periods, and data collection methodologies than our work. Direct comparison of results should be made with caution due to these variations in data sources and preprocessing approaches.

2.1. Player Count and Popularity Prediction

Several studies have focused on predicting player engagement metrics. One study [1] developed models to predict daily player counts using features including 7-day time-lagged player statistics, reviews, genres, and platform support, employing Random Forest with Pearson's feature selection on the top 100 most-played games. Similarly, another study [2] employed Bayesian hierarchical models to predict median concurrent players, finding that genre-based hierarchical approaches performed best and that games released early in the month correlated with higher popularity. For binary popularity classification, researchers [3] applied LightGBM, XGBoost, and other classifiers to predict whether games achieve positive ratings above a certain threshold on a large dataset. Another study [4] compared Random Forest, Decision Tree, and Logistic Regression for predicting estimated owner counts classified into three tiers, with Random Forest performing best.

2.2. Game Success and Revenue Prediction

Research [5] addressed revenue prediction by combining Steam API data with external sources including Twitch metrics and Metacritic scores, using CatBoost for log-transformed revenue and identifying Steam follower count as the strongest predictor. Other work [6] applied Random Forest for predicting player count increases as a success metric, selecting it as the best performer over SVR and Bayesian Regression. Research [7] used a smaller dataset to classify success into four tiers, finding that user ratings and genre were significant predictors.

2.3. Rating and Review Prediction

One study [8] predicted Metascore ratings using an extensive model comparison including Linear Regression variants, tree-based methods, and gradient boosting algorithms, with LightGBM achieving the best performance. Another investigation [9] examined player recommendations based on

gameplay hours, with Random Forest performing best. Research [10] specifically examined indie games, finding that Logistic Regression outperformed Random Forest for popularity prediction and that user-generated tags were the most indicative features for indie game success.

2.4. Price Prediction Studies

While the majority of Steam ML research focuses on player counts, popularity, and success metrics, game price prediction has received comparatively less attention in academic literature. According to the catalog of Steam ML works, only four studies specifically target price prediction, and these are primarily found in Kaggle notebooks [11,12,13,14]. These existing price prediction efforts typically employ AutoML approaches or standard regression techniques without the comprehensive time-based evaluation methodology we adopt. Our work differs from these existing approaches in several key aspects. We focus exclusively on pre-release metadata, excluding post-release signals such as user reviews, playtime statistics, and sales figures that would not be available when setting initial prices. We employ a time-based train/test split to simulate realistic deployment scenarios where models trained on historical games predict prices for future releases.

3. Algorithms and Methodology

3.1. Machine Learning Algorithms

We evaluated multiple regression algorithms spanning different model families to identify the most effective approach for Steam game price prediction.

- **Dummy Regressor (Mean)**: Predicts the mean training price as a performance baseline.
- **Dummy Regressor (Median)**: Predicts the median training price as an alternative baseline.
- **Linear Regression**: Standard linear model without regularization, providing a reference point for measuring the benefits of regularization and non-linearity.
- **Ridge Regression:** Applies L2 regularization to stabilize coefficient estimates for high-dimensional sparse data, preventing overfitting while maintaining all features.
- **Lasso Regression:** Uses L1 regularization to perform automatic feature selection by driving some coefficients to zero.
- **ElasticNet:** Combines L1 and L2 regularization for balanced shrinkage and sparsity.
- **Huber Regressor:** Robust linear regression that is less sensitive to outliers than ordinary least squares.
- **Tweedie Regressor:** Generalized linear model from the exponential family, designed for targets with specific variance-mean relationships.
- **AdaBoost:** Adaptive boosting method that iteratively reweights samples based on prediction errors, focusing subsequent learners on difficult examples.
- **Random Forest:** Bootstrap aggregation of decision trees that provides variance reduction and feature importance estimation.
- **XGBoost:** Gradient boosting with regularization that handles missing values natively and supports parallel computation.
- **LightGBM:** Histogram-based gradient boosting with leaf-wise tree growth, offering faster training and memory efficiency for large datasets.
- **Stacking Regressor:** Meta-ensemble that combines Ridge, Random Forest, and XGBoost as base learners with a Ridge meta-learner to optimally combine their predictions.

LightGBM was selected as the primary model based on superior cross-validation performance. It achieved the best validation MAE and R^2 while maintaining a relatively small train-validation gap, indicating stable generalization. The algorithm handles the high-dimensional sparse feature space efficiently and naturally captures non-linear interactions between genres, tags, and other categorical

variables that linear models cannot represent. We performed initial screening to identify the strongest and most practical models for final tuning and analysis. Models with significantly higher training costs, such as stacking ensembles, were not prioritized when they failed to provide meaningful validation improvement over LightGBM alone.

3.2. Methodology

Data Preparation Workflow:

- **Population Filtering:**
  - *Discount correction for base price*: For games with non-zero discounts, we reconstructed the base price using the formula: base_price = Price / (1 − Discount/100), then replaced Price with base_price for modeling.
  - *Remove free games*: Dropped all games with Price equal to zero.
  - *Market presence filter*: Removed 6,739 games with Estimated owners equal to '0-0', then dropped the Estimated owners column entirely to avoid post-release leakage. Figure 3.2 shows the estimated owners distribution before filtering.
  - *Remove extreme-priced products*: Removed games priced above $200, as these typically represent bundles, collector editions, or professional tools rather than standard games.
  - *Parse release dates*: Parsed all release date fields and removed entries with invalid or unparseable dates.
  - *Remove duplicates*: Dropped duplicate records to ensure data integrity.
  - *Leakage prevention*: Dropped all post-release variables to ensure the feature set reflects only information available before release. Removed variables include: Peak CCU, Positive review count, Negative review count, Recommendations, User score, Score rank, Metacritic score, Metacritic url, Average playtime forever, Average playtime two weeks, Median playtime forever, Median playtime two weeks, Reviews text, Estimated owners, DLC count, and Achievements.

Figure 3.1 illustrates the impact of discount correction and free game removal on the price distribution. The left panel shows the original distribution where 20.9% of games are free and 39.4% are priced $0-5. After applying discount correction to recover base prices and removing free games entirely, the right panel shows a healthier distribution with 49.7% of games in the $0-5 range, 26.0% in the $5-10 range, and better representation across all price tiers.

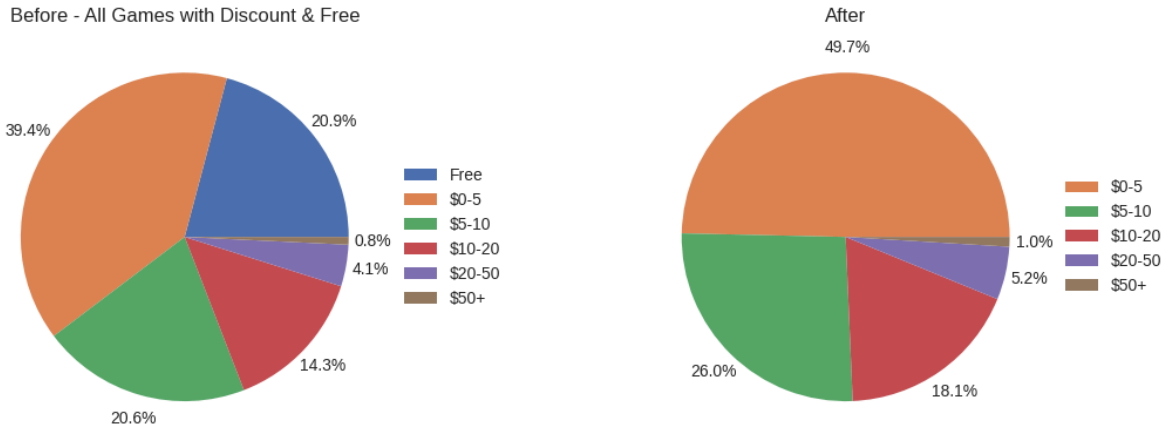

FIGURE 3.1: Price distribution before vs after discount correction and free game removal

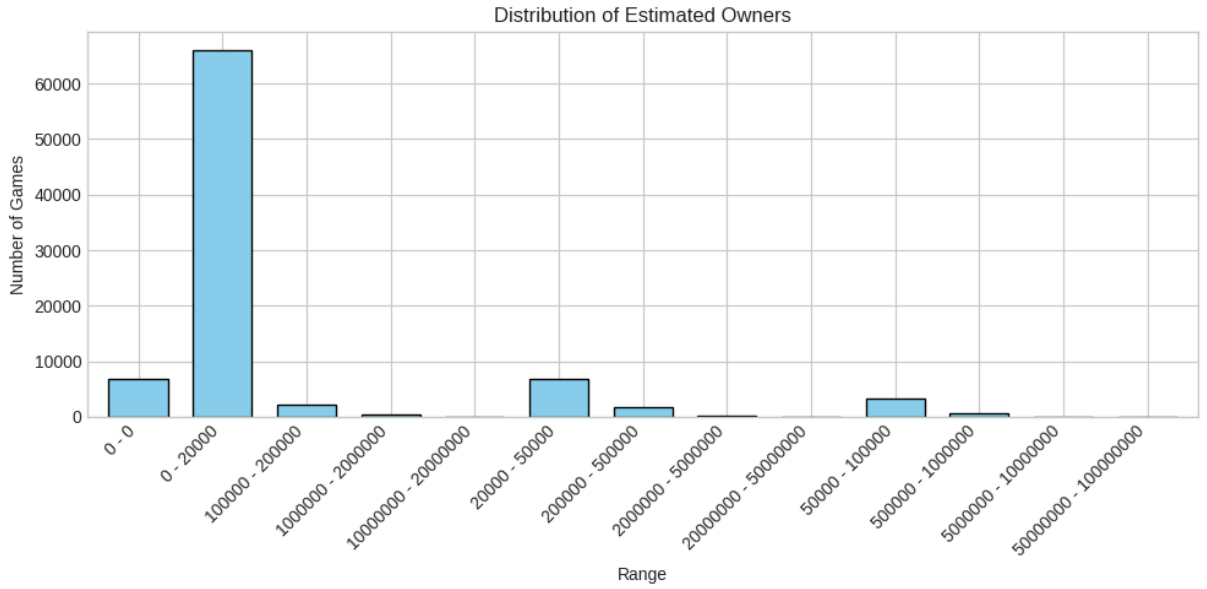

FIGURE 3.2: Distribution of Estimated Owners and '0-0' filter

- **Missing Value Handling:** Dropped rows with missing Genres or Categories, as these are essential game descriptors. Created a binary indicator for missing Tags and filled missing values with empty strings for encoding purposes.

- **Feature Engineering:**
    - Extracted is_early_access flag from the Genres column
    - Created is_self_published by comparing Developer and Publisher fields
    - Computed supported_lang_count and audio_lang_count from language data
    - Extracted description_length as word count from game descriptions
    - Derived temporal features: release_year, release_month, release_day_of_week, and release_season
    - Created has_audio indicator for games with any audio language support
    - Created has_english_audio indicator for games with English audio support
    - Created has_tags binary indicator and tag_count for the number of tags assigned to each game

- **Feature Distributions:** The numeric features exhibited heavy right-skewed distributions as shown in Figure 3.3. Price distribution shows most games concentrated in lower price ranges, with a long tail extending to premium products. Required age is dominated by games with no age restriction. Language count features show that most games support only a few languages, with rare games offering extensive localization. Description length varies widely, with most games providing brief descriptions and some offering detailed content.

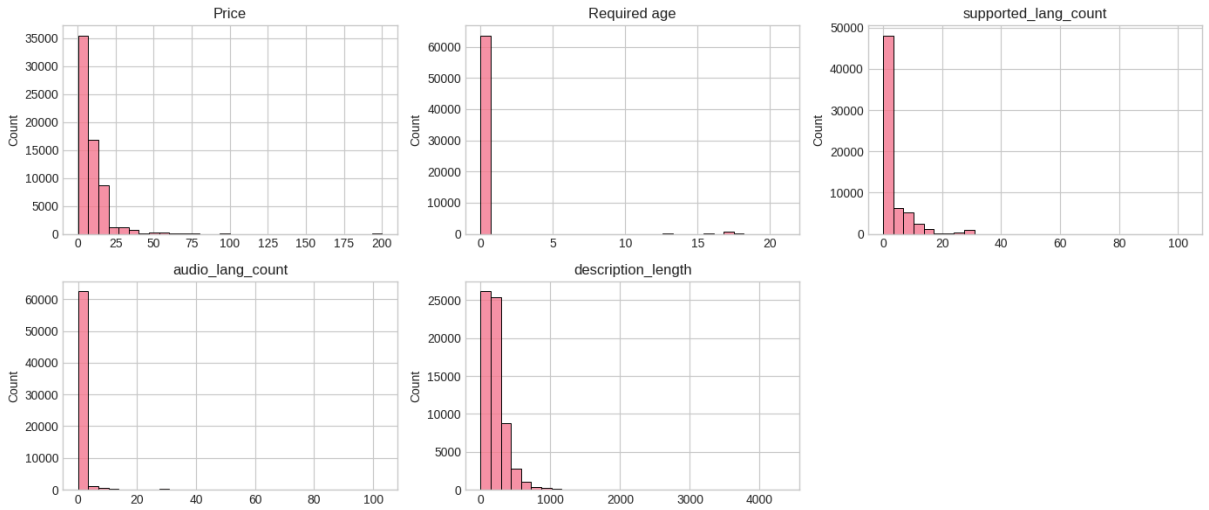

FIGURE 3.3: Numeric Feature Distributions

Boolean features reveal market characteristics, as shown in Figure 3.4. Windows support and English language support are near-universal, appearing in over 95% of games. Self-published games comprise about 74% of the dataset. Other features like Mac support, Linux support, and audio localization show more variation, appearing in 20-50% of games.

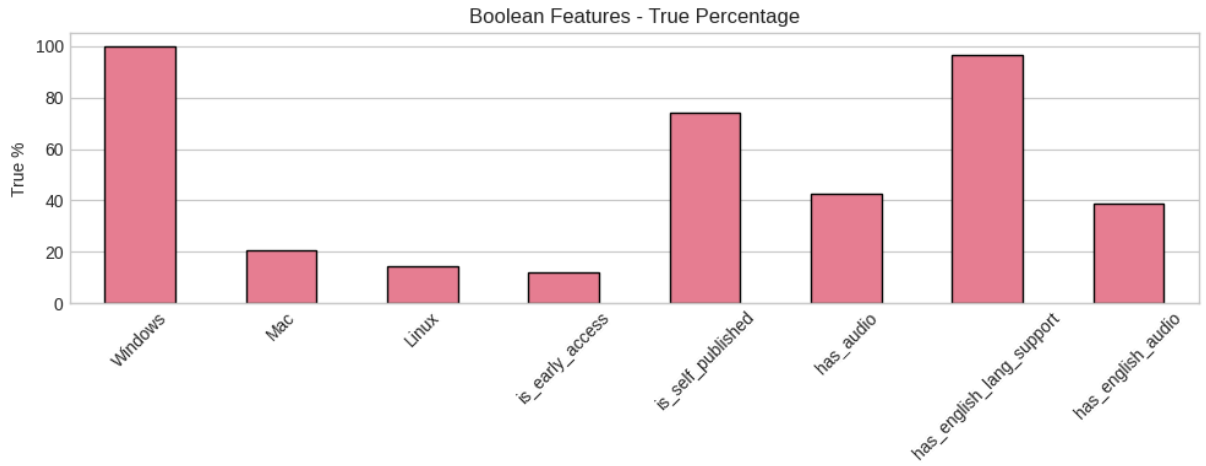

FIGURE 3.4: Boolean Feature Distribution

Temporal patterns in Figure 3.5 reveal important trends in game releases. The number of releases has grown exponentially since 2015, with the market becoming increasingly crowded. Game releases are distributed relatively uniformly across months and seasons, with slight peaks in fall. However, release day patterns show strong preferences for Thursday and Friday, with minimal weekend releases. This pattern reflects strategic timing to maximize visibility in Steam's new releases section during high-traffic weekdays.

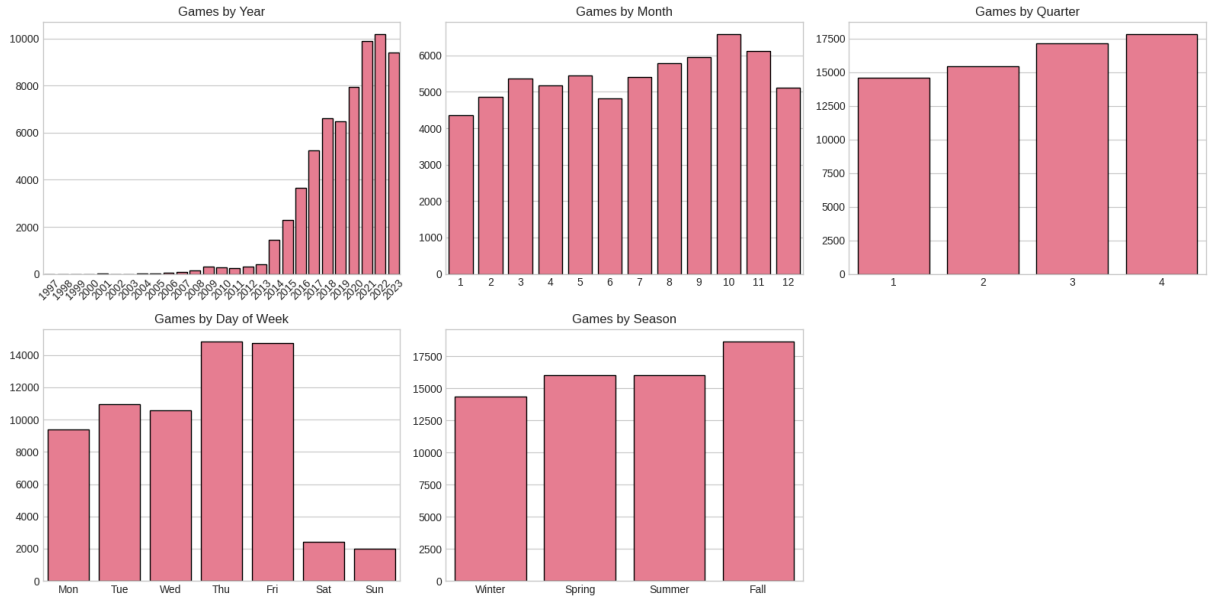

FIGURE 3.5: Release Date Temporal Patterns

- **Skew Handling for Heavy-Tailed Count Features:** To reduce the influence of extreme values and improve model stability, we applied log1p transformations to highly skewed count-based predictors. Prior to transformation, we capped extreme localization counts at their 99th percentile values. The supported_lang_count was capped at 29 languages and audio_lang_count at 13 languages. These caps remove implausible values likely resulting from parsing artifacts while preserving meaningful variation in the data. Figure 3.6 demonstrates the effect of log1p transformations on these heavy-tailed features. The original distributions show severe right skew with skewness values exceeding 3.0. After transformation, skewness is reduced to below 1.5, successfully compressing extreme values while preserving meaningful variation in the typical range.

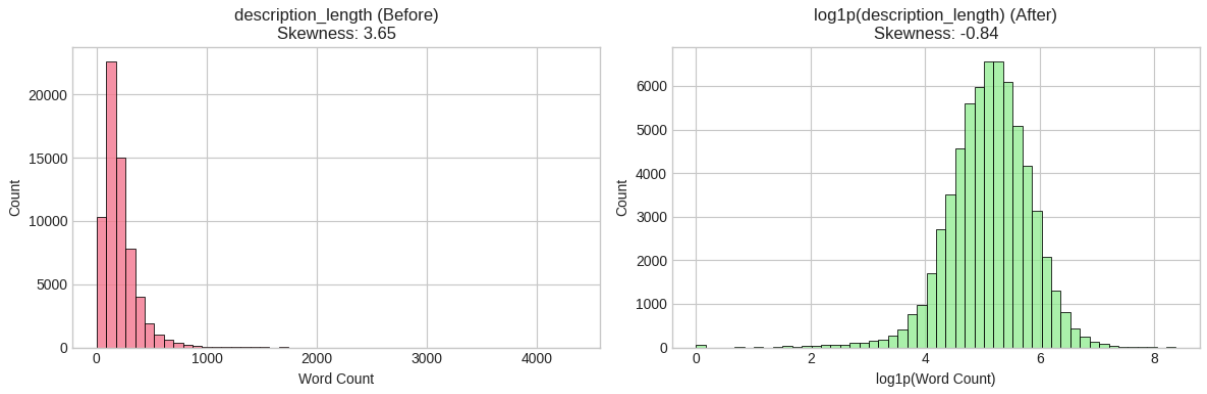

FIGURE 3.6a: description_length before vs after log1p

The description length transformation reduces skewness from 3.65 to 0.84, converting the highly right-skewed distribution into an approximately normal distribution centered around 500-1000 words.

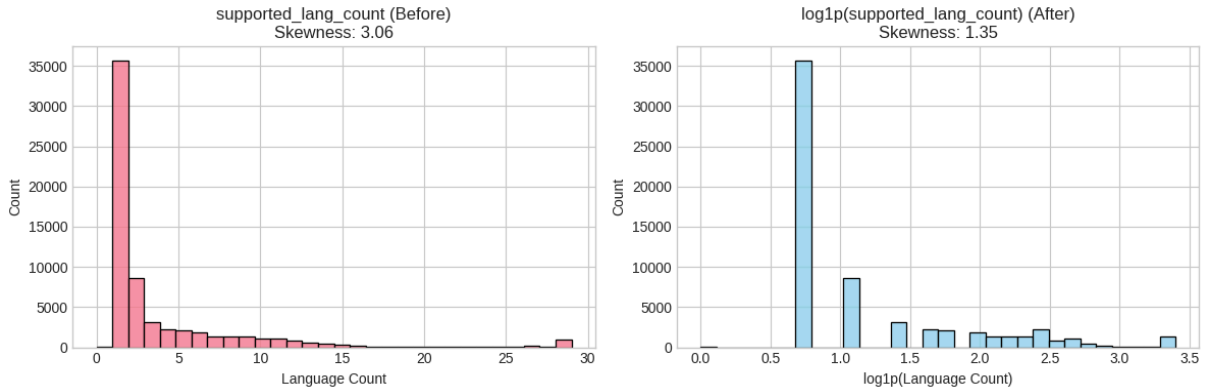

FIGURE 3.6b: supported_lang_count before vs after log1p

The supported language count transformation reduces skewness from 3.06 to 1.35. The original distribution shows that most games support 1-3 languages, with rare cases supporting 20+ languages. After transformation, the scale becomes more interpretable and less dominated by extreme values.

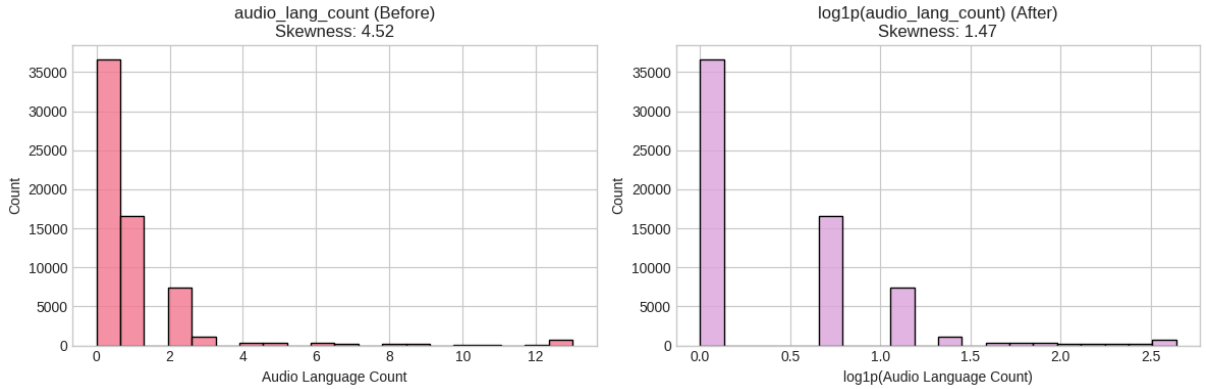

FIGURE 3.6c: audio_lang_count before vs after log1p

The audio language count transformation reduces skewness from 4.52 to 1.47. The original distribution is heavily concentrated at zero or one language, with very few games offering multilingual audio. The transformation normalizes this distribution for more stable model training.

- **Feature Encoding:** We applied appropriate encoding methods for each categorical feature type:
    - One-hot encoding for release_season, creating four binary columns for Winter, Spring, Summer, and Fall
    - Multi-label binarization for Genres, creating binary columns for all unique genre values
    - Multi-label binarization for Categories, creating binary columns for all unique category values
    - Multi-label binarization for Tags, restricted to the top 100 most frequent tags

After extracting the is_early_access feature, we removed the Early Access token from the Genres multi-label encoding to avoid duplicate representation of the same concept.

The tag vocabulary restriction to the top 100 tags was implemented to control dimensionality and reduce noise from extremely rare tags. The top tag "Indie" appears in over 35,000 games, while most

tags beyond the top 100 appear in fewer than 100 games. This justifies the restriction to avoid creating sparse features from extremely rare tags that add noise rather than signal.

- **Feature Selection:** We performed feature selection to remove redundant and uninformative features:
    - Dropped near-constant columns: Windows and has_english_lang_support both had over 99% of values in a single category, providing no discriminative power
    - Dropped highly correlated features: release_quarter showed 0.97 correlation with release_month, meaning they carry essentially identical information. We retained release_month as it provides finer temporal granularity

The correlation matrix in Figure 3.7 shows relationships between numeric features. Most correlations are weak, with values below 0.3. The strongest correlation of 0.31 exists between log_supported_lang_count and log_audio_lang_count, which is expected since extensive localization often includes both subtitle and audio support. The near-perfect correlation of 0.97 between release_month and release_quarter justified dropping the latter. Moderate correlations exist between description length and language counts, suggesting that games with more extensive descriptions tend to offer more localization options.

FIGURE 3.7: Numeric Feature Correlation Matrix

- **Target Variable Transformation:** We applied log1p transformation to the Price variable to address severe right skewness. Figure 3.8 shows the price distribution before and after transformation. The original distribution has skewness of 6.29, with a heavy concentration of games below $10 and a long tail extending to $200. After log1p transformation, skewness is reduced to 0.17, creating an approximately normal distribution centered around $5-10. Models predict log(price) during training, and predictions are inverse-transformed using expm1 for evaluation in dollar terms. This transformation serves multiple purposes: it normalizes the target distribution for improved model training, reduces the influence of extreme outliers on the loss function, and makes the target more compatible with linear model assumptions.

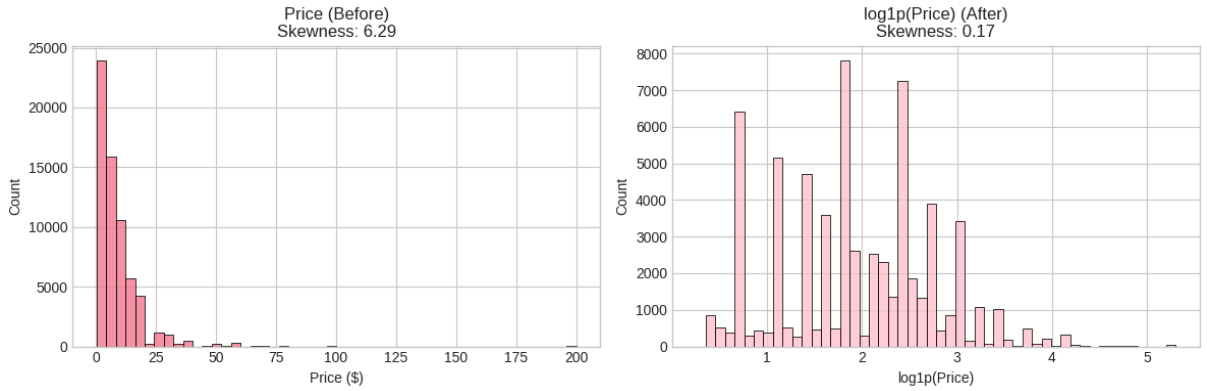

FIGURE 3.8: Target variable distribution before vs after log1p(Price)

- **Outlier Analysis:** We analyzed outliers on the log-transformed price using the Interquartile Range method. The analysis identified $49 as the upper bound for typical prices based on the IQR criterion. However, we decided to keep all prices without additional removal for several important reasons. First, statistical rarity does not necessarily indicate invalid data. In the gaming industry, $60 is the standard AAA price point, and prices in the $40-50 range are common for mid-tier releases from established developers. Second, the log transformation already compresses the price scale and reduces the influence of high values on model training. Third, removing valid premium products would bias our model toward cheap games and reduce its ability to generalize to the full price spectrum. Figure 3.9 shows the price outlier detection boxplot on the log scale. While games above $49 are flagged as statistical outliers, they represent legitimate pricing tiers in the market rather than erroneous data points.

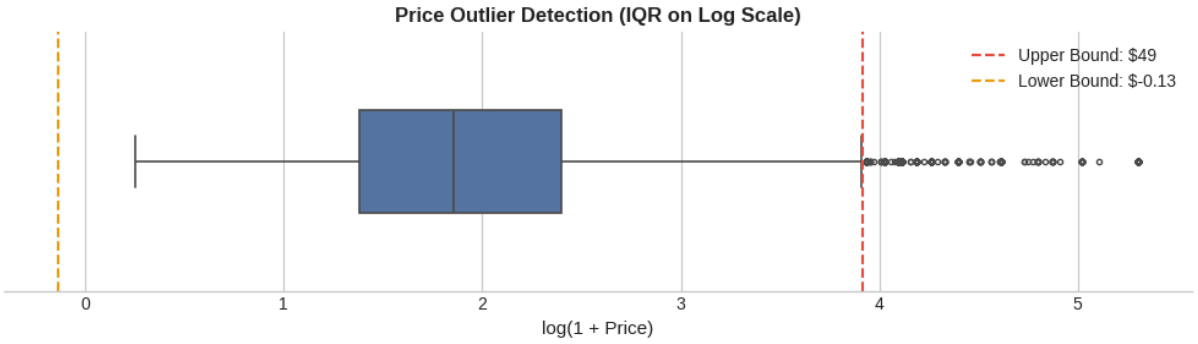

FIGURE 3.9: Price Outlier Detection Box Plot

- **Language Count Capping:** Both language count features were capped at their 99th percentile to remove implausible extreme values. The supported_lang_count was capped at 29 languages, and audio_lang_count was capped at 13 languages. Values above these thresholds affected only 26 games out of over 80,000 in the dataset. These extreme values likely represent parsing artifacts or database errors rather than genuine localization efforts. Capping preserves meaningful variation in the 1-29 language range while removing implausible extremes that could distort model training.
- **Model Training and Validation:** We employed TimeSeriesSplit with 5 folds for cross-validation, maintaining temporal ordering within the training data to prevent future information leakage. This approach simulates realistic deployment where models trained on historical data predict prices for future releases. Hyperparameter tuning strategies were tailored to each model family:
    - Tree-based models such as RandomForest, XGBoost, and LightGBM used RandomizedSearchCV to efficiently explore large hyperparameter spaces
    - Linear models such as Ridge and Huber used GridSearchCV for exhaustive search over smaller parameter grids
    - All models were optimized using cross-validation scores to prevent overfitting to any single train-validation split

Scoring was based on negative MAE in dollar terms. Although models predict log1p(price) internally, we inverse-transform predictions using expm1 during cross-validation to evaluate performance in the original dollar scale. This ensures optimization targets align with the practical goal of minimizing dollar prediction errors.

- **Performance Metrics:** We evaluated models using multiple complementary metrics:
    - **MAE (Mean Absolute Error)**: Average absolute prediction error in dollars, robust to outliers and directly interpretable
    - **MSE (Mean Squared Error)**: Average squared prediction error, forming the basis for RMSE and penalizing larger errors quadratically
    - **RMSE (Root Mean Squared Error)**: Standard metric that penalizes large errors more heavily than MAE
    - **R^2 (Coefficient of Determination)**: Proportion of variance explained, measuring overall model fit
    - **Adjusted R^2**: R^2 penalized for the number of features, accounting for model complexity
    - **RMSLE (Root Mean Squared Log Error)**: Relative error metric appropriate for skewed targets, measuring percentage-like errors

4. Experimental Setup

4.1. Dataset Description

- **Source:** Steam games dataset scraped from the Steam platform
- **Dataset Link:** https://www.kaggle.com/datasets/fronkongames/steam-games-dataset
- **Original Size:** 111,452 games with 39 features
- **Data Integrity Fix:** On initial load, the dataset had a column-name shift due to AppID alignment issues. For example, 'DiscountDLC count' appeared merged with another column. We fixed this by reassigning the correct 39-column schema before any analysis, ensuring all features were properly aligned with their intended data.
- **Final Size After Preprocessing:** 81,298 games with 17 columns before encoding
- **Target Variable:** Price in USD, ranging from $0.29 to $199.99
- **Feature Types:** The dataset contains four main feature categories:

    - **Numeric features**: Required age, supported_lang_count, audio_lang_count, description_length
    - **Boolean features**: Windows, Mac, Linux, is_early_access, is_self_published, has_audio, has_english_audio
    - **Categorical multi-label features**: Genres, Categories, Tags
    - **Temporal features**: Release date
- **Data Imbalance:** The price distribution is heavily right-skewed with median $5.24 and mean $8.69. The distribution is concentrated in lower price ranges with a long tail extending to premium products. This imbalance reflects the reality of the Steam marketplace, where indie games dominate the platform numerically but premium AAA titles exist at higher price points.

4.2. Pre-processing Steps

- **Encoding Results:** The categorical encoding process transformed the original 17 features into a high-dimensional feature space:
    - Season encoding: 4 binary columns
    - Genre encoding: 33 binary columns for all unique genres
    - Category encoding: 40 binary columns for all unique categories
    - Tag encoding: 100 binary columns for the top 100 most frequent tags
    - Final feature count: 191 features after all transformations
- **Genre Distribution:** Figure 4.1 shows the distribution of the top 20 genres in the dataset. Indie is the dominant genre, followed by Action and Casual. The total number of unique genres is 32 after excluding Early Access, which was extracted as a separate binary feature. This distribution reflects the democratization of game development, where independent developers can publish games alongside major studios.

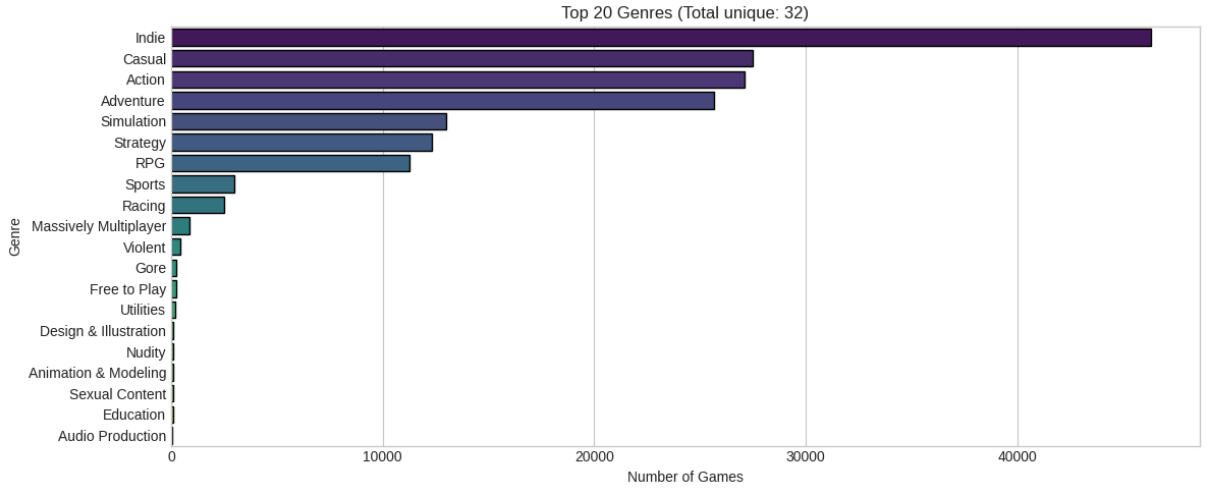

FIGURE 4.1: Top 20 Genres Distribution

- **Category Distribution:** Figure 4.2 presents the top 25 Steam categories by game count. Single-player dominates, making it the most common category by a significant margin. Steam Achievements followed by Steam Cloud. Full controller support, Multi-player, and Steam Trading Cards represent the next tier. The distribution shows that single-player experiences remain the most common game type on Steam, but multiplayer features and Steam platform integration like achievements and cloud saves are also prevalent. Categories related to specific multiplayer modes such as Co-op, Remote Play Together, and various PvP options appear in smaller but still substantial numbers.

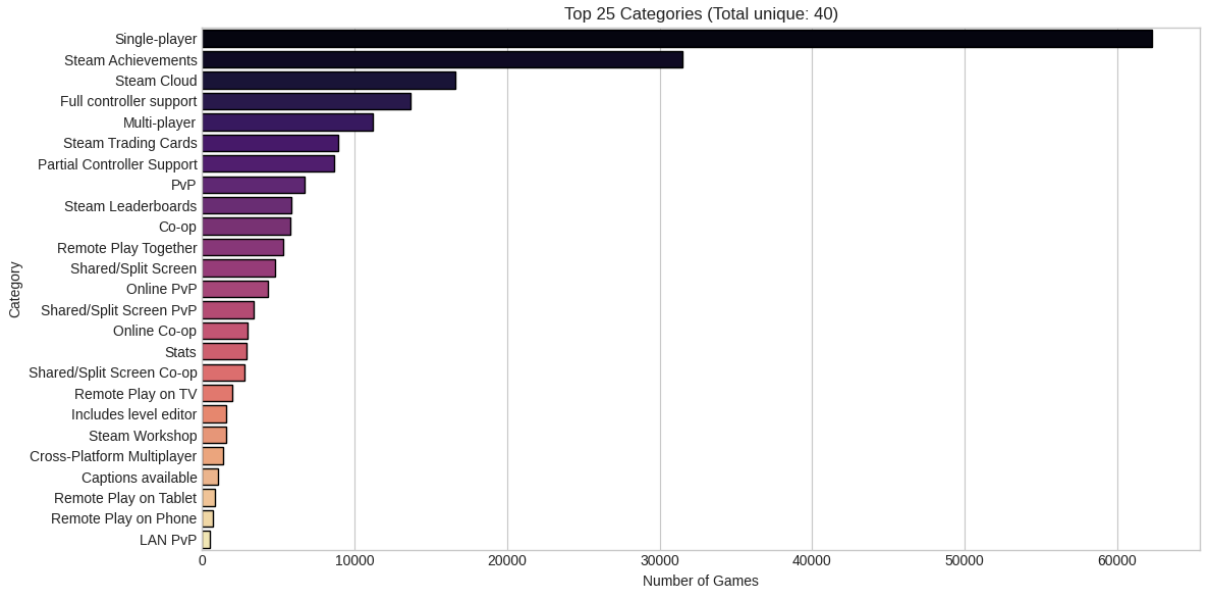

FIGURE 4.2: Top 25 Categories Distribution

- **Tag Distribution:** Figure 4.3 displays the top 30 user-assigned tags. Indie, Singleplayer, and Action are the most common tags. Casual and Adventure follows. The frequency drops off significantly after the top 10 tags. Only the top 100 tags were encoded as features to control dimensionality. Tags beyond the top 100 appear in very few games and would create sparse, noisy features that add little predictive value. The steep frequency drop-off visible in the chart justifies this cutoff, as rare tags would contribute more noise than signal to the model.

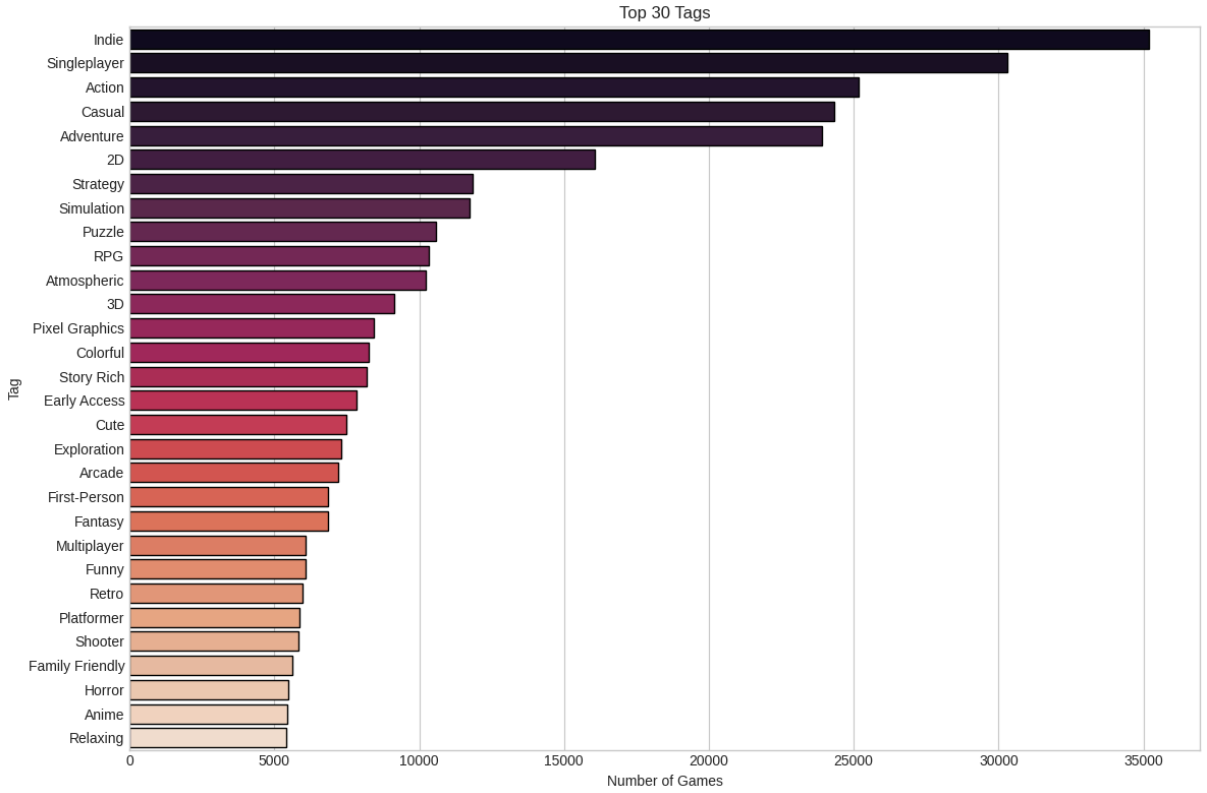

FIGURE 4.3: Top 30 Tags Distribution

- **Release Season Distribution:** Figure 4.4 shows game releases by season. The distribution is relatively uniform across all four seasons, with each season containing between 14,000 and 18,500 games. Fall has slightly more releases with 18,409 games, followed by Summer with 15,821 games, Spring with 15,813 games, and Winter with 14,170 games. The near-uniform distribution across seasons suggests that game developers do not strongly favor any particular season for releases, unlike other entertainment industries like film which show strong seasonal patterns. However, the slight peak in fall may reflect strategic timing to capture holiday shopping momentum.

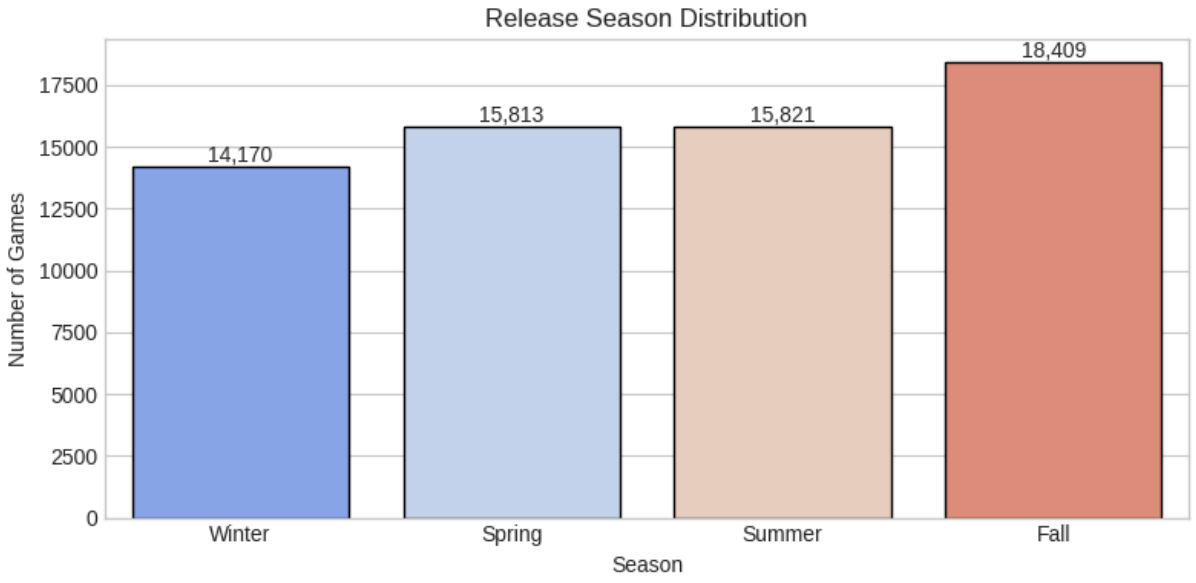

FIGURE 4.4: Release Season Distribution

- **Exploratory Data Analysis:** Before encoding, we performed comprehensive exploratory analysis on the training set to understand feature distributions and relationships with price. This analysis informed our preprocessing decisions and feature engineering choices.
- **Feature Distributions:** All numeric features exhibited heavy right-skew. Price had skewness of 6.35, description_length showed skewness of 3.65, supported_lang_count had skewness of 3.06, and audio_lang_count displayed skewness of 4.52. The Required_age feature was 98% zeros, indicating that most games have no age restriction.
- **Categorical Insights:** Tags had the highest cardinality with 447 unique values and 18.6% missing data. Genres had 33 unique values and Categories had 40. The multi-label nature of these features created high-dimensional sparse encodings, with most games having 2-4 genres, 3-8 categories, and 5-15 tags.
- **Boolean Features:** Windows support appeared in 99.97% of games and has_english_lang_support appeared in 95.9% of games. These near-universal features provided no discriminative power, justifying their removal from the final model. Self-published games comprised 74% of the dataset, reflecting the large number of independent developers on Steam.
- **Genre and Price Relationship:** Early Access and RPG games commanded the highest median prices at $7-8, while Indie and Casual games had the lowest median prices around $4. Self-published games showed lower median prices than publisher-backed games, likely reflecting differences in production budgets and market positioning. Multi-platform releases with Mac and Linux support correlated with higher prices.

- **Temporal Patterns:** Game releases grew exponentially since 2015, peaking around 2020-2022. Releases were uniform across seasons, but Thursday and Friday were dominant release days with minimal weekend releases. This pattern reflects strategic timing to maximize visibility during high-traffic weekdays.
- **Target Imbalance:** 75% of games were priced under $10, and 94% were priced under $20. This heavy concentration in low-price tiers means the model is primarily trained on cheap indie games and may struggle with premium pricing for AAA titles.
- **Median Price by Genre:** Figure 4.5 shows median prices for the top 10 genres. Early Access games have the highest median price at $8.99, followed by RPG at $7.99. Strategy and Simulation games cluster around $6.99. Sports, Adventure, and Action games fall in the $5-7 range. Indie, Casual, and Racing games have the lowest median prices around $4.99. The pattern suggests that genres requiring longer development time and more complex systems tend to command higher prices. Early Access pricing may reflect aspirational pricing by developers before market feedback, while Indie and Casual games compete primarily on low prices.

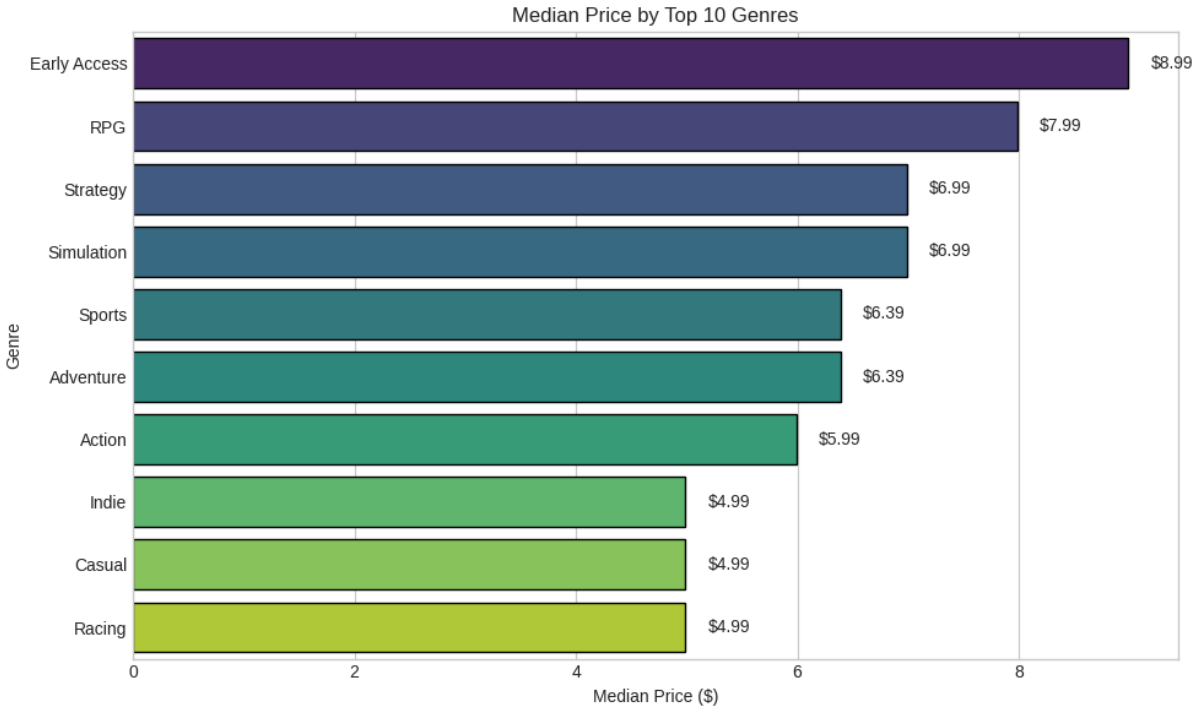

FIGURE 4.5: Median Price by Top 10 Genres

- **Price by Boolean Features:** Analysis of boolean features revealed meaningful price differences as shown in Figure 4.6. Self-published games have lower median prices at $4.99 compared to publisher-backed games at $8.99. Early Access games command higher median prices at $8.99, possibly because developers set aspirational prices before full release and market validation. Games with audio localization show higher median prices at $6.99, reflecting larger production budgets required for voice acting and audio translation. Mac and Linux support correlate with higher median prices, as multi-platform releases require additional development investment and typically come from more established studios with larger budgets.

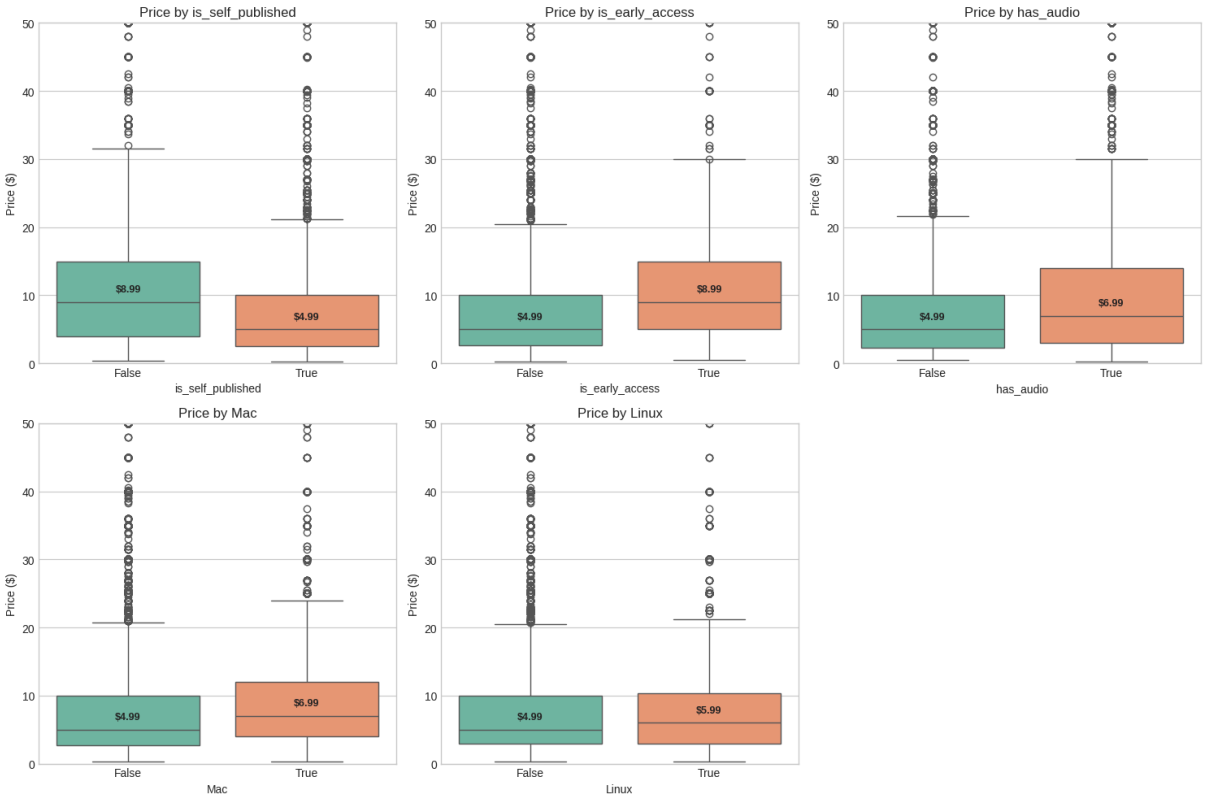

FIGURE 4.6: Price by Boolean Features Box Plots

- **Price Distribution by Platform:** Figure 4.7 examines pricing patterns across different platform combinations. Windows-only games dominate the market numerically with over 45,000 games but have the lowest median price. Games supporting Windows plus Mac and Windows plus Linux or all three platforms have the higher median prices. This pattern suggests multi-platform releases typically come from more established studios with larger budgets and resources to handle cross-platform development. The Windows-only segment includes many small indie developers releasing on their primary platform, leading to lower average prices due to competition and limited resources.

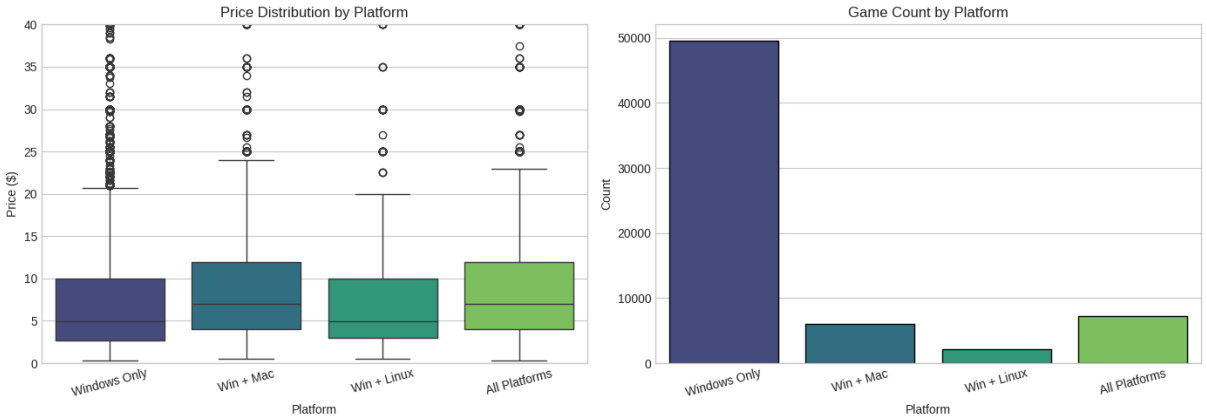

FIGURE 4.7: Price Distribution by Platform Combination

- **Price Trends Over Time:** Figure 4.8 displays two important temporal trends. The left panel shows game releases by year, revealing nearly zero releases before 2006 and fewer than 500

per year until 2012. Growth accelerated dramatically from 2013 onwards, reaching over 8,000 releases annually by 2020. This explosion reflects Steam's shift to a more open platform allowing independent developers easier access. The right panel shows median price trends by year. Prices showed wild fluctuations before 2010 due to small sample sizes, with median prices varying between $5 and $15. From 2014 onwards, median prices stabilized in the $5-6 range despite the massive increase in releases. This stabilization suggests market maturation and price discovery, with the influx of indie games anchoring the median price at a lower level.

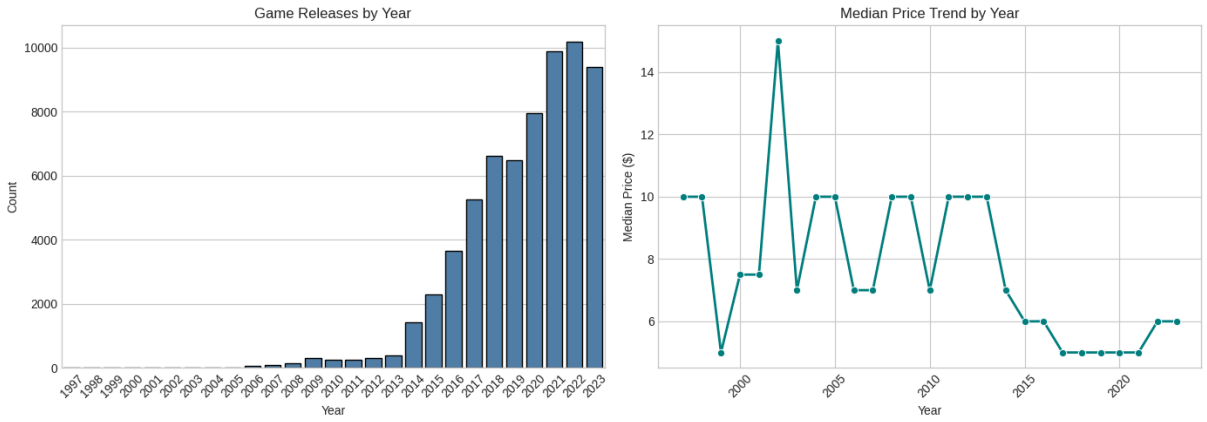

FIGURE 4.8: Price Trends by Release Year

Price Distribution Analysis: Figure 4.9 provides two complementary views of price distribution in the training set. The left panel shows the count of games in each price bucket with percentages annotated. The $0-5 bucket contains 49.6% of all games, while the $5-10 bucket contains 25.6%. Together, 75% of games are priced under $10. The $10-15 and $15-20 buckets contain 11.3% and 7.4% respectively. Higher price buckets from $20-200 contain only 6% of games combined. The right panel shows the top 10 genres by count, with Indie dominating at approximately 47,000 games, followed by Casual and Action. This visualization reinforces that the dataset is heavily weighted toward indie games, which cluster in lower price tiers.

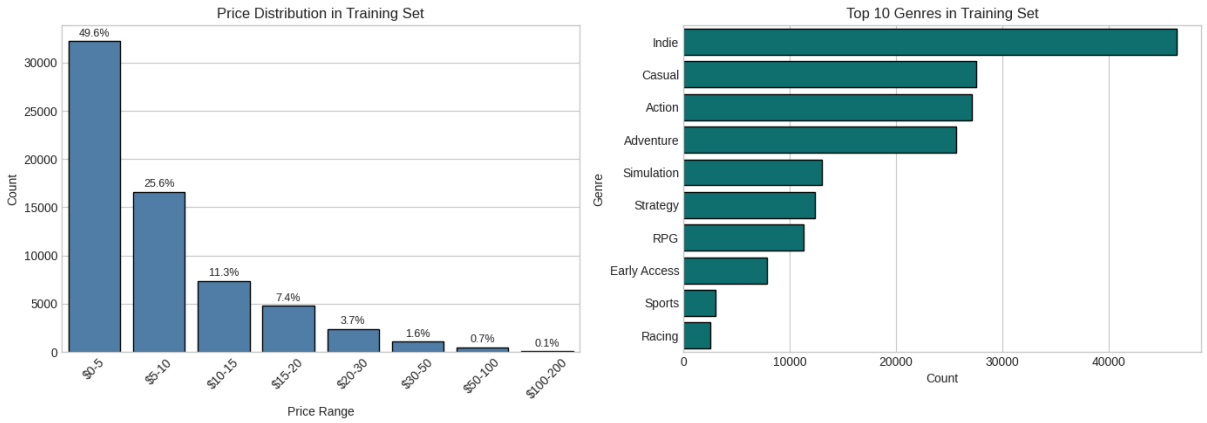

FIGURE 4.9: Price Distribution Imbalance

- **Feature Standardization:** StandardScaler was applied to all numeric features after log transformation. The scaled features include log_description_length,

log_supported_lang_count, log_audio_lang_count, release_year, release_month, release_day_of_week, tag_count, and Required_age. Standardization ensures all features have mean 0 and standard deviation 1. This standardization serves multiple purposes. It prevents any single feature from dominating due to scale differences, ensures consistent regularization penalties across features in linear models, and improves convergence speed for gradient-based optimization algorithms. Boolean and one-hot encoded features were not standardized as they already exist on a consistent 0-1 scale.

4.3. Experimental Design

- **Train/Test Split:** We employed a time-based split rather than random splitting to simulate realistic deployment scenarios:
    - **Method**: Temporal split with cutoff date
    - **Cutoff Date**: December 12, 2023
    - **Training Set**: 64,213 games released before the cutoff date
    - **Test Set**: 16,172 games released on or after the cutoff date
    - **Split Ratio**: Approximately 80% train / 20% test

The time-based split ensures the test set consists entirely of games released after the training period, mimicking how the model would be deployed in practice to predict prices for upcoming releases based on historical data. This prevents temporal leakage and provides a more realistic estimate of generalization performance than random splitting.

- **Cross-Validation Strategy:** TimeSeriesSplit with 5 folds was used for cross-validation within the training set. This approach maintains temporal ordering by training on earlier data and validating on later data in each fold. The progressive nature prevents future information leakage while still providing multiple validation estimates for robust performance measurement.
- **Hyperparameter Tuning:** Hyperparameter optimization strategies were tailored to each model family based on the size of their search spaces:
    - **LightGBM**: RandomizedSearchCV with 15 iterations to explore the large hyperparameter space, followed by two-stage GridSearchCV focusing specifically on regularization parameters
    - **XGBoost and RandomForest**: RandomizedSearchCV with 15 iterations for efficient exploration of large parameter spaces
    - **Ridge and Huber**: GridSearchCV for exhaustive search over smaller parameter grids

All tuning used TimeSeriesSplit cross-validation with scoring based on negative MAE in dollar terms. Models predict log1p(price) internally, but predictions are inverse-transformed using expm1 during evaluation to optimize for dollar-scale errors.

- **Hardware and Software Environment:** All experiments were conducted using Python with the following key libraries:
    - **scikit-learn**: For preprocessing, model training, cross-validation, and evaluation
    - **pandas and numpy**: For data manipulation and numerical operations
    - **xgboost and lightgbm**: For gradient boosting models
    - **matplotlib and seaborn**: For visualization and exploratory analysis
- **Computational Challenges:** This project encountered significant computational challenges that impacted our experimental workflow. We discovered an unexpected compatibility issue between LightGBM and Google Colab's environment, even when using high-performance A100 GPUs. Paradoxically, LightGBM executed substantially faster on local machines with standard CPUs than on Colab's GPU infrastructure. This performance inversion prevented us

from leveraging cloud computing resources that should have accelerated development. The root cause appears to be related to how LightGBM interacts with Colab's specific system configuration. LightGBM is primarily CPU-optimized and does not benefit from GPU acceleration in the same way neural networks do. The Colab environment's overhead, library versions, or resource allocation policies apparently created bottlenecks that negated the theoretical advantages of more powerful hardware. Consequently, all experiments were conducted on local hardware, where LightGBM performed optimally. However, even on local machines, complete experimental runs required several hours. This extended runtime made iterative development extremely challenging. Any modification to preprocessing steps, feature engineering approaches, or model configurations required restarting the entire pipeline from scratch, consuming hours for each experimental iteration. We could not rapidly prototype and test ideas, instead requiring careful planning for each experimental run. The time constraints limited our ability to implement all features discussed during project presentation.

5. Experimental Evaluation

5.1. Model Comparison Results

We evaluated 13 different models using 5-fold TimeSeriesSplit cross-validation on the training set. Figure 5.1 presents a comprehensive six-panel visualization comparing all models across multiple performance dimensions, and Table 5.1 provides the detailed metrics.

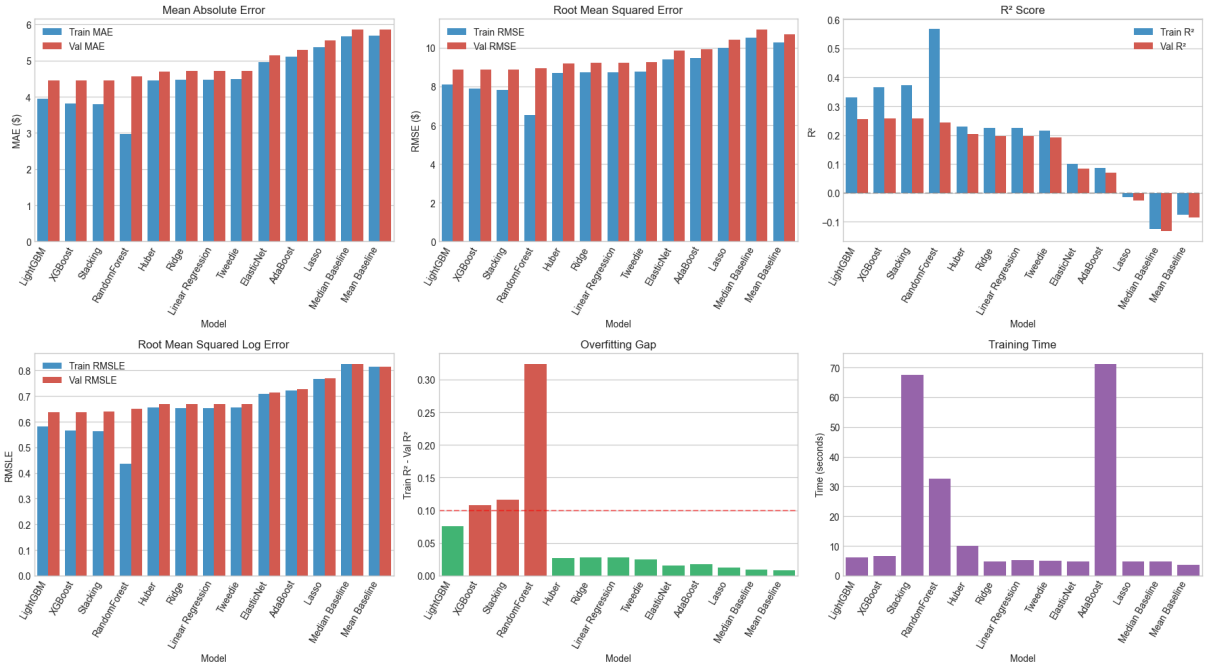

FIGURE 5.1: Model Comparison - Six-panel visualization showing Train vs Validation MAE, RMSE, R^2, RMSLE, Overfitting gap, and Training time across all 13 models

The visualizations reveal several clear patterns. The top panel shows that gradient boosting methods, particularly LightGBM, XGBoost, and the Stacking ensemble, achieved the lowest validation MAE around $4.45. Linear models clustered around $4.70-4.75, while baseline models exceeded $5.85. The middle-bottom panels demonstrate that tree-based models show larger train-validation gaps, indicating some overfitting tendency, while linear models maintain minimal gaps but higher absolute errors. The bottom-right panels show that ensemble methods require significantly more training time, with Stacking taking over 60 seconds compared to under 10 seconds for most other models.

Model	Train MAE	Val MAE	Train R^2	Val R^2	Overfit Gap
LightGBM	$3.95	$4.45	0.331	0.256	0.075
XGBoost	$3.81	$4.45	0.366	0.258	0.108
Stacking	$3.79	$4.46	0.374	0.257	0.117
RandomForest	$2.96	$4.57	0.567	0.243	0.324
Huber	$4.45	$4.70	0.230	0.203	0.027
Ridge	$4.47	$4.71	0.225	0.198	0.027
Linear Regression	$4.47	$4.71	0.225	0.198	0.027
Tweedie	$4.48	$4.71	0.216	0.192	0.024
ElasticNet	$4.96	$5.15	0.100	0.084	0.016
AdaBoost	$5.11	$5.29	0.086	0.069	0.017
Lasso	$5.38	$5.56	-0.014	-0.027	0.013
Median Baseline	$5.68	$5.86	-0.124	-0.133	0.009
Mean Baseline	$5.69	$5.86	-0.076	-0.084	0.008

TABLE 5.1: Cross-validation results for all 13 models

Model Performance Analysis:

- LightGBM emerged as the optimal choice, providing the best balance of accuracy, generalization, and efficiency. It achieved the lowest validation MAE of $4.45 while maintaining a relatively modest train-validation gap of 0.075, indicating stable generalization without severe overfitting. The model captured complex non-linear relationships in the feature space without memorizing training-specific patterns.
- XGBoost performed similarly to LightGBM in terms of validation MAE but exhibited a slightly larger overfitting gap of 0.108. The Stacking ensemble achieved marginally better training

performance but offered no meaningful validation improvement over LightGBM alone while requiring substantially more computational resources.

- Random Forest demonstrated classic overfitting behavior with excellent training performance, achieving R^2 of 0.567 on the training set, but experienced significant degradation on validation folds. The large overfitting gap of 0.324 indicates the model memorized training patterns that did not generalize well.
- Linear models including Ridge, Huber, and standard Linear Regression showed minimal overfitting with gaps under 0.03 but produced higher absolute errors around $4.70. This pattern indicates underfitting for this complex feature space. The linear assumption cannot capture the non-linear interactions between genres, tags, categories, and other features that influence pricing.
- Lasso and ElasticNet performed poorly, with validation MAE exceeding $5.15 and negative R^2 values. The L1 regularization drove too many coefficients to zero, eliminating important features and severely limiting model capacity. AdaBoost also underperformed with validation MAE of $5.29, struggling with the high-dimensional sparse feature space.
- Baseline models that simply predict the mean or median price achieved validation MAE around $5.86 with negative R^2 values, confirming that all machine learning models provided substantial improvements over naive prediction strategies.

5.2. Hyperparameter Tuning

The final LightGBM model uses the hyperparameters shown in Table 5.2. These values were determined through a multi-stage optimization process designed to balance predictive accuracy with generalization capability.

Parameter	Value
n_estimators	300
max_depth	15
learning_rate	0.05
num_leaves	100
min_child_samples	20
subsample	0.7
colsample_bytree	0.8
reg_alpha	10.0

Parameter	Value
reg_lambda	20.0

TABLE 5.2: Final LightGBM hyperparameters

Optimization Process:

The regularization parameters were selected through an extended grid search after initial hyperparameter tuning revealed overfitting issues. The optimization proceeded through three distinct stages, each addressing specific model behavior issues.

The default LightGBM configuration without hyperparameter tuning achieved validation MAE of $4.45 with train MAE of $3.95, resulting in a modest overfitting gap of 0.075. While this baseline performance was reasonable, we sought improvements through hyperparameter optimization. Initial hyperparameter tuning via RandomizedSearchCV explored 15 different parameter combinations across n_estimators, max_depth, learning_rate, num_leaves, min_child_samples, subsample, and colsample_bytree. This tuning improved validation MAE from $4.45 to $4.36, a reduction of $0.09. However, the train MAE dropped dramatically to $3.14, indicating the model began fitting training data too closely. The overfitting gap expanded from 0.075 to 0.200, more than doubling and raising concerns about generalization. To address this overfitting, we performed a targeted grid search over regularization parameters. We tested reg_alpha values, and reg_lambda values. The selected configuration with reg_alpha of 10.0 and reg_lambda of 20.0 achieved validation MAE of $4.40, only $0.03 higher than the minimum observed during hyperparameter tuning. Critically, the overfitting gap was reduced from 0.200 to 0.098, representing a 24% improvement in model stability.

Table 5.3 summarizes the model evolution through these three optimization stages:

Version	Train MAE	Val MAE	Train R^2	Val R^2	Overfit Gap
Default Parameters	$3.95	$4.45	0.331	0.256	0.075
After Hyperparameter Tuning	$3.14	$4.36	0.481	0.282	0.200
After Regularization	$3.75	$4.40	0.364	0.266	0.098

TABLE 5.3: LightGBM model evolution through optimization stages

Figure 5.2 visualizes the evolution across all performance dimensions. The six panels show how each optimization stage affected different aspects of model behavior. Initial hyperparameter tuning dramatically reduced training error and increased training R^2, but these improvements came at the cost of increased overfitting. Regularization successfully recovered model stability, bringing the

train-validation gap back toward the original level while retaining most of the validation performance improvements.

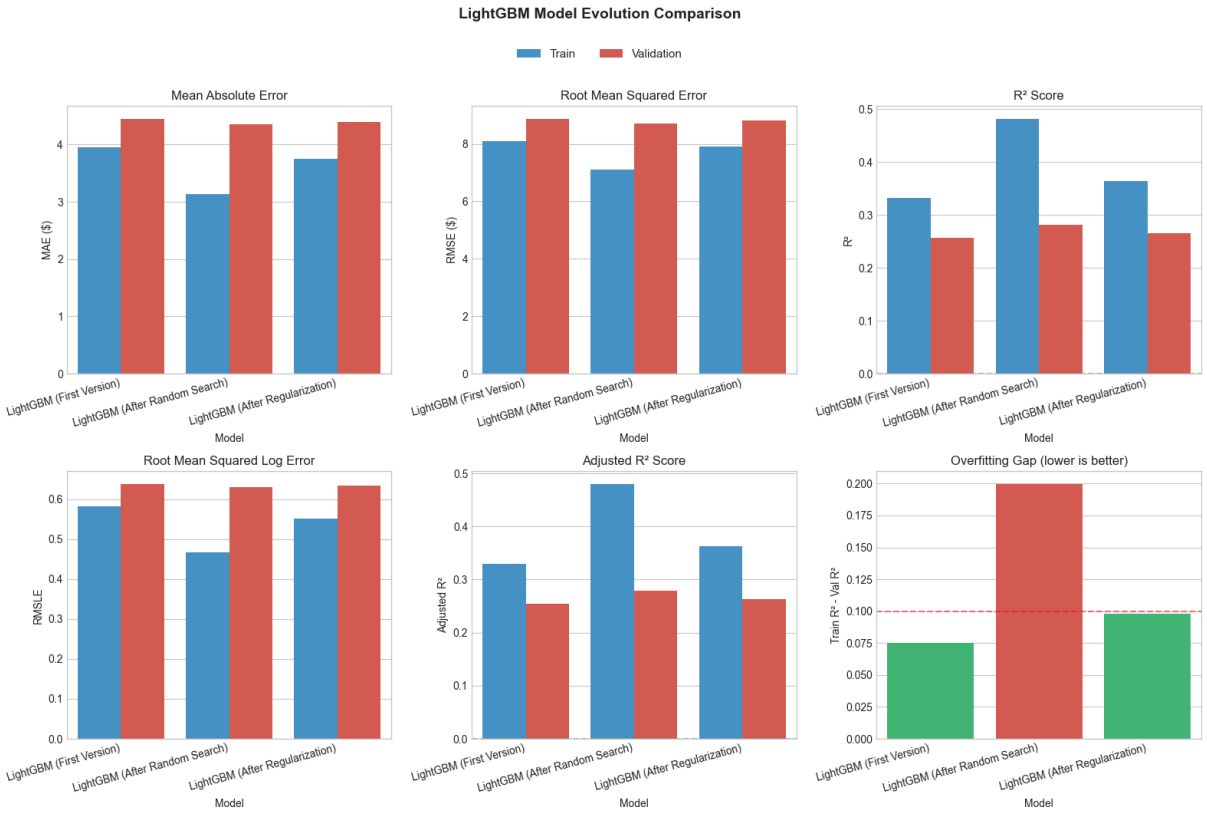

FIGURE 5.2: LightGBM Evolution Comparison

The evolution demonstrates a common pattern in machine learning optimization. Aggressive hyperparameter tuning can improve fit to training data and even improve validation metrics in the short term, but may increase overfitting risk. This manifests as a widening gap between training and validation performance, indicating the model is learning training-specific patterns rather than general relationships. Regularization parameters provide a direct mechanism to control model complexity and encourage simpler patterns that generalize better. The L1 regularization controlled by reg_alpha encourages feature sparsity by driving some feature weights to zero. The L2 regularization controlled by reg_lambda penalizes large weights and encourages weight distribution across features. Together, these parameters constrain the model's capacity to memorize training data while maintaining its ability to capture genuine patterns. The final configuration strikes an effective balance. It achieves validation MAE only $0.05 worse than the unregularized optimum while reducing the overfitting gap compared to the unregularized tuned model. This trade-off is strongly favorable, as the modest validation performance sacrifice buys substantial improvements in model stability and expected generalization to truly unseen data.

5.3. Final Test Set Evaluation

The final regularized LightGBM model was trained on the complete training set of 64,213 games and evaluated on the held-out test set of 16,172 games released after December 12, 2023. This evaluation provides the most realistic estimate of performance on future unseen data. Figure 5.3 presents a comprehensive comparison of train and test performance across multiple metrics. The six panels show performance on MAE, RMSE, R^2, Adjusted R^2, RMSLE, and the breakdown of R^2 between the full test set and the test set excluding premium games above $150.

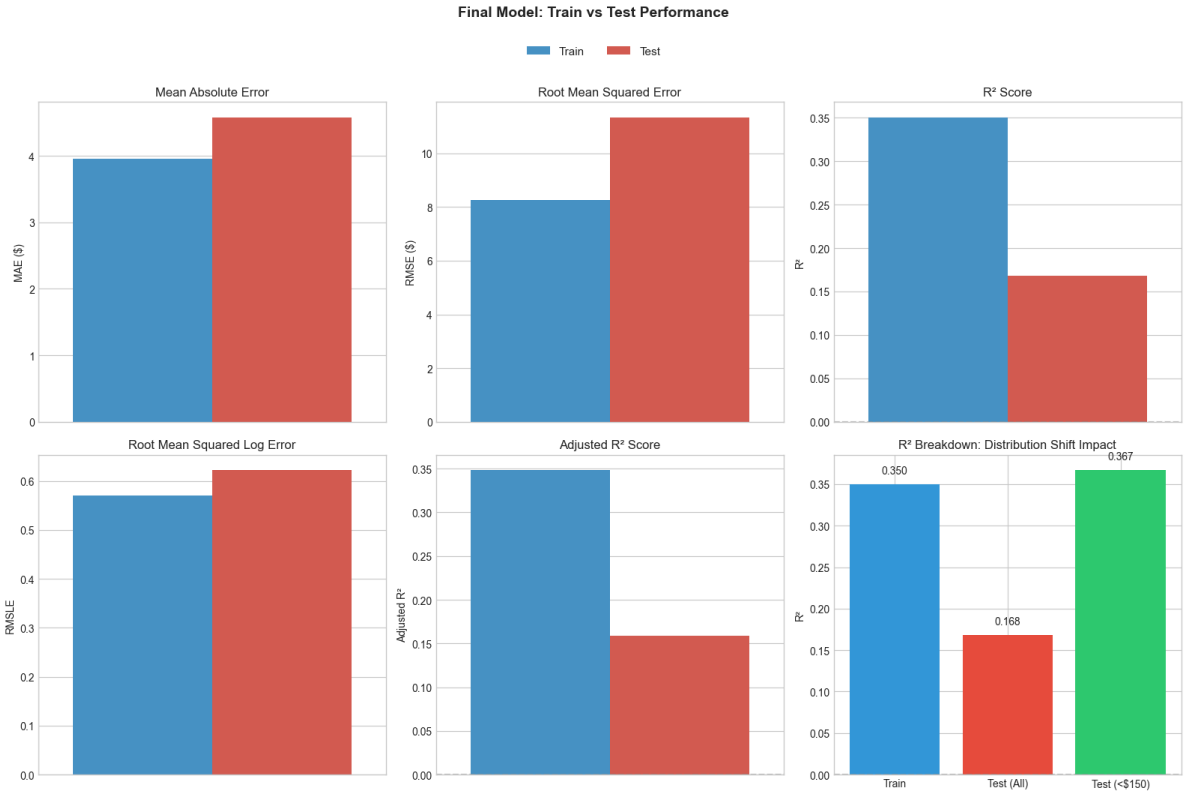

FIGURE 5.3: Final Train vs Test Performance

Table 5.4 presents the detailed performance metrics on both training and test sets:

Metric	Train	Test
MAE	$3.96	$4.58
RMSE	$8.25	$11.34
R^2	0.350	0.168
Adj R^2	0.348	0.158
RMSLE	0.571	0.622

TABLE 5.4: Final model performance on train and test sets

The test MAE of $4.58 represents an increase of only $0.62 from the training MAE of $3.96. This relatively modest increase indicates reasonable generalization for typical game prices. The model maintains similar average prediction accuracy on unseen data compared to the training distribution. However, the R^2 metric shows a more substantial degradation, dropping from 0.350 on the training set to 0.168 on the test set. This apparent performance drop is examined in detail in the following error analysis section, where we demonstrate that it primarily reflects distribution shift rather than fundamental model failure. The RMSE increase from $8.25 to $11.34 is more pronounced than the MAE increase. This larger gap for RMSE compared to MAE indicates that the model makes some large errors on the test set. Since RMSE squares errors before averaging, it is more sensitive to outliers and large prediction mistakes. This suggests the test set contains some games that are particularly difficult to predict, pulling the RMSE upward more than the MAE.

5.4. Error Analysis

The test set performance shows an apparent degradation in R^2 from 0.35 to 0.17, which initially appears concerning. However, deeper analysis reveals this is primarily due to distribution shift rather than model failure. The test period contains proportionally more premium-priced products than the training period, and these products are fundamentally different from the typical games the model was trained on.

Table 5.5 quantifies the key differences between train and test distributions that explain the performance gap:

Metric	Train	Test
Price Std	$10.24	$12.43
Games $150+ (%)	0.065%	0.204%
R^2 (All Games)	0.350	0.168
R^2 (Games < $150)	-	0.367
MAE (Games < $150)	-	$4.19

TABLE 5.5: Distribution shift between train and test sets

The test set exhibits higher price variance with standard deviation of $12.43 compared to $10.24 in training. More significantly, the proportion of extremely expensive games priced above $150 is over three times higher in the test set at 0.204% compared to 0.065% in training. While these percentages seem small, they represent approximately 33 games in the test set that fall far outside the training distribution. When we exclude these extreme premium games and recalculate test metrics on games

below $150, the R^2 improves dramatically to 0.367. This value is actually higher than the training R^2 of 0.350, demonstrating that the model generalizes excellently to typical games within its training distribution. The MAE for games below $150 is $4.19, better than both the overall test MAE of $4.58 and comparable to the training MAE of $3.96. This analysis confirms that the apparent performance degradation is not a failure of the model to learn generalizable patterns. Instead, it reflects the presence of outlier products in the test set that the model was never exposed to during training. The model learned the pricing dynamics of typical games but naturally struggles with collector editions, software bundles, and professional simulation tools that have fundamentally different pricing mechanisms.

The largest prediction errors provide insight into where the model struggles. Table 5.6 shows the five worst predictions by absolute error:

Actual Price	Predicted	Error
$199.99	$1.93	$198.06
$199.99	$1.96	$198.03
$199.99	$2.31	$197.68
$199.99	$2.50	$197.49
$199.99	$2.54	$197.45

TABLE 5.6: Top 5 Worst Predictions - Highest Absolute Errors

The pattern is striking and consistent. All of the worst errors involve games priced at exactly $199.99 that the model predicted would cost $1-3. The model systematically underestimates these premium products by approximately $197, predicting prices similar to typical indie games. These products represent a fundamentally different category than standard games. At the $199.99 price point, products are often:

- Collector or premium editions bundling multiple games with physical merchandise
- Professional simulation software like flight simulators with extensive add-on content
- Complete franchise collections spanning decades of releases
- Educational or training software marketed to institutions rather than individual gamers

The pricing for these products is driven by factors completely absent from our feature set. Brand recognition, perceived prestige value, professional licensing, and institutional buyer budgets all play roles that cannot be inferred from basic metadata like genre, platform support, and tag counts. The model correctly learned that most games with standard indie tags and genres cost $3-10, but it has

no mechanism to identify the exceptional cases where special positioning justifies premium pricing. Importantly, the model's failure on these edge cases does not indicate a problem with its core functionality. For a pricing guidance tool aimed at typical game developers, predictions on standard games matter far more than predictions on rare premium products. The model serves its intended purpose for the bulk of the market while naturally struggling with outliers that fall outside its design scope.

5.5. Discussion

The test R^2 of 0.168 may initially appear low compared to machine learning benchmarks in other domains. However, this reflects fundamental limitations of the problem rather than deficiencies in our modeling approach. Game pricing involves numerous factors that cannot be captured from pre-release metadata alone.

Unmeasured Pricing Factors: Several critical pricing determinants are absent from our dataset. Brand recognition and developer reputation play enormous roles in pricing power, but these are difficult to quantify from metadata alone. A new game from an unknown studio and a new game from Rockstar or Valve will have vastly different pricing flexibility even with identical genres and features. We attempted to capture some of this signal through the is_self_published feature and developer-publisher relationships, but these proxies are crude compared to true brand equity measurement. Publisher marketing budgets represent another unmeasured factor. A game backed by millions in marketing investment can command premium pricing because it will reach mass audiences, while an equally good game without marketing support must compete on price. Pre-release hype and community anticipation, often driven by marketing, significantly influence pricing decisions but cannot be measured from static metadata. Perceived quality and production values are critical pricing signals that we cannot directly observe. A game might list "Indie, Action, Singleplayer" tags, but whether it features AAA-quality graphics and voice acting or minimalist pixel art is not captured. Development budget, team size, and years of development time all influence production quality and appropriate pricing, yet none appear in our features.

Strategic Pricing Considerations: Game pricing involves strategic business decisions that vary even among similar products. Two mechanically identical games might adopt different pricing strategies based on their publishers' objectives. A studio prioritizing revenue maximization might charge $29.99 and sell fewer copies at higher margins, while a studio prioritizing market penetration might charge $9.99 to maximize player base for future sequels or DLC sales. Free-to-play games with in-app purchases, not included in our analysis, represent an entirely different revenue model that emphasizes user acquisition over upfront pricing. Regional pricing strategies, sales tactics, and bundling approaches all influence the list price we observe, but these reflect publisher strategy rather than inherent game characteristics. Our model learns average pricing patterns but cannot predict when a specific developer will deviate from these patterns for strategic reasons.

Temporal Dynamics: Gaming trends evolve continuously, and patterns learned from 1997-2023 data may not fully apply to 2024 releases. Genre popularity shifts over time as players' tastes change, as new game mechanics emerge, and as market saturation affects different categories. The massive growth in indie game releases from 2015-2023 may have created downward pressure on average prices that might stabilize or reverse in future years. Platform dynamics also evolve. The test period saw increasing proportions of premium products, suggesting possible market segmentation where high-end and low-end products become more distinct.

Distribution Shift Impact: The increased proportion of premium products in the test period disproportionately impacts R^2 due to its squared error weighting. A single game priced at $199.99 that we predict at $3 contributes far more to the R^2 calculation than ten games priced at $10 that we

predict at $9. This mathematical property means R^2 can drop substantially even when the vast majority of predictions remain accurate. When we exclude the 0.2% most expensive games from test set evaluation, R^2 improves to 0.367, actually exceeding training performance.

Contextual Performance Comparison: Compared to the baseline strategy of predicting the median price at $5.24 for all games, which achieves MAE of $5.86, our model reduces error by $1.28 or 22%. This improvement is meaningful and demonstrates the value of incorporating game-specific metadata into pricing decisions. The model successfully captures real pricing signals related to genre, platform support, localization, publisher backing, and temporal trends. The R^2 of 0.168 on the full test set means the model explains approximately 17% of price variance, while 83% remains unexplained. While this might seem low, it reflects the inherent unpredictability of creative product pricing where subjective factors dominate. In domains like real estate or used car pricing, R^2 values above 0.80 are common because objective factors like square footage or mileage explain most variance. Creative products like games, books, and films have much lower explained variance because quality, appeal, and market positioning are inherently subjective.

6. Conclusions and Future Work

6.1. Summary

This project developed a comprehensive machine learning pipeline to predict Steam game prices using only pre-release metadata. Starting with a raw dataset, we performed extensive preprocessing to remove free games, correct pricing discounts, filter out unreleased products, and eliminate post-release features that would not be available at pricing decision time. The final cleaned dataset contained with 17 core features expanded to 191 features after encoding multi-label categorical variables. We systematically compared 13 different machine learning algorithms spanning baseline predictors, linear models, and ensemble methods. Each model was evaluated using 5-fold TimeSeriesSplit cross-validation to respect temporal ordering and prevent information leakage. After comprehensive evaluation, LightGBM emerged as the optimal choice, balancing predictive accuracy with generalization stability. The final model underwent multi-stage optimization. Initial hyperparameter tuning improved validation performance but introduced overfitting. Targeted regularization through grid search over L1 and L2 penalty parameters successfully reduced the overfitting gap while retaining most performance gains. Error analysis revealed that apparent performance degradation on the test set primarily reflects distribution shift rather than model failure. The test period contained proportionally more premium-priced products above $150, which the model had rarely encountered during training. When excluding these extreme outliers, test R^2 improved to 0.367, exceeding training performance and demonstrating excellent generalization within the model's intended operating range. The project demonstrates that pre-release game metadata provides useful pricing signals. Genres, platform support, localization effort, publisher backing, and temporal trends all contribute meaningfully to price prediction. However, substantial variance remains unexplained, reflecting unmeasured factors like brand recognition, marketing budgets, and perceived quality that fundamentally influence pricing decisions.

6.2. Strengths and Weaknesses

Strengths:

- Our preprocessing pipeline handles the complexities of real-world Steam data effectively. Multi-label categorical features like genres, categories, and tags were properly encoded using binary indicators for each unique value. We addressed the high cardinality of tags by restricting to the top 100 most frequent tags, balancing dimensionality control with information retention. Heavy-tailed numeric features were normalized through log

transformations and capping of extreme values, improving model stability without discarding legitimate data points.

- The time-based train-test split provides realistic evaluation that simulates actual deployment scenarios. Unlike random splitting, our temporal split trains on historical games and tests on future releases. This approach prevents optimistically biased estimates and reveals how the model will perform when deployed to predict prices for upcoming games based on patterns learned from past releases.
- The hyperparameter tuning methodology demonstrates systematic optimization with attention to overfitting. Rather than simply selecting parameters that minimize validation error, we explicitly monitored train-validation gaps and used regularization to control model complexity. The three-stage optimization process shows careful analysis of trade-offs between training fit, validation performance, and generalization stability.
- Our error analysis examining distribution shift between train and test sets, analyzing the worst predictions, and evaluating performance on different price segments, we provide actionable insights about when the model succeeds and when it fails. This transparency helps potential users understand the model's limitations and appropriate use cases.

Weaknesses:

- The limited feature set excludes important pricing signals that influence real-world decisions. Brand recognition and developer reputation are difficult to quantify but enormously influential on pricing power. A sequel from an established franchise or a game from a renowned studio can command premium pricing that our features cannot capture. Marketing budgets and pre-release hype drive pricing decisions but are not reflected in static metadata available at the time of pricing.
- Perceived quality and production values represent another major gap. Games with identical genres and features might differ dramatically in graphics quality, voice acting, music production, and overall polish. Development budget, team size, and years of development investment all influence appropriate pricing but are not observable in our dataset. We attempted to proxy some of these factors through features like description length, publisher backing, and localization effort, but these proxies are imperfect.
- The model struggles with premium-priced products outside the training distribution. Games priced above $150 typically represent collector editions, professional software, or complete franchise bundles with pricing mechanisms fundamentally different from standard games. The training set contained very few such products at only 0.065%, insufficient for the model to learn their pricing patterns. When the test set contained three times more premium products at 0.204%, the model systematically underestimated their prices.
- The R^2 of 0.168 on the test set indicates substantial unexplained variance. While error analysis shows this partly reflects distribution shift, even the improved R^2 of 0.367 on games below $150 suggests that our observable features explain only about one-third of pricing variance. The remaining two-thirds reflects unmeasured factors, strategic decisions, and inherent unpredictability of creative product pricing.
- The model provides point predictions without uncertainty quantification. A prediction of $10 might be highly confident for a typical indie game with standard features or highly uncertain for an unusual game combining rare characteristics. Without prediction intervals or confidence scores, users cannot distinguish reliable predictions from speculative ones. This limitation reduces the model's practical utility for risk-averse decision-makers who need to understand prediction uncertainty.

6.3. Lessons Learned

Technical Insights:

- Feature engineering for multi-label categorical variables requires careful handling to avoid dimensionality explosion. Genres, categories, and tags each have dozens to hundreds of unique values, and naive encoding creates feature spaces with thousands of sparse binary indicators.
- Time-based validation is essential for temporal data and provides substantially different insights than random cross-validation. Random splitting intermingles data across time periods, allowing models to learn from future information when predicting past games. This creates optimistically biased performance estimates. Time-based splitting with TimeSeriesSplit respects temporal ordering and reveals genuine generalization capability to future unseen data. The performance differences between our cross-validation and test results would have been missed with random splitting.
- Distribution shift between train and test periods can significantly impact metrics even when the model generalizes well within its training distribution. The R^2 metric is particularly sensitive to outliers and distribution changes due to its squared error formulation. We learned to look beyond aggregate metrics and analyze performance on different data segments. The discovery that R^2 improved to 0.367 when excluding premium games revealed the true source of apparent performance degradation.
- Hyperparameter optimization requires balancing validation performance with overfitting control. Aggressive tuning improved our validation MAE from $4.45 to $4.36 but doubled the overfitting gap from 0.075 to 0.200. We learned that achieving the absolute minimum validation error is not always desirable if it comes with severe overfitting. Adding regularization sacrificed only $0.04 in validation MAE while reducing the overfitting gap.

Domain Insights:

- Game pricing involves substantial strategic variation even among similar products. Two games with identical genres, features, and quality might be priced differently based on their publishers' business objectives. Some studios prioritize revenue maximization through premium pricing, while others prioritize market penetration through competitive pricing. Launch timing relative to competitors, bundling strategies, and planned post-release content all influence initial pricing decisions but are not reflected in pre-release metadata.
- The Steam marketplace has evolved dramatically from 2015 to 2023, with exponential growth in releases creating different competitive dynamics across time periods. Early in the period, premium indie games could command $15-20 prices, but increasing competition and market saturation pushed median prices down to $5-6 by 2020. The model must learn pricing patterns that shift over time, making predictions inherently uncertain for future periods that may follow different trends.
- Premium products above $150 operate in a fundamentally different market segment than typical games. These products are often collector editions with physical merchandise, professional simulation software for enthusiasts, or complete franchise collections bundling decades of content. Their pricing reflects perceived prestige value, collector demand, and willingness to pay among specialized audiences rather than features observable in metadata. Predicting these products requires understanding of luxury good pricing psychology that differs from standard game pricing.

6.4. Future Work

Enhanced Feature Engineering:

- Incorporating text embeddings from game descriptions could capture quality signals and marketing positioning not available in categorical features. Modern language models can encode descriptions into dense vector representations that reflect writing quality, feature emphasis, and promotional tone. A game describing itself with detailed lore and sophisticated language might justify higher pricing than one with a brief generic description, and embeddings would capture these differences.
- Image features extracted from game screenshots and promotional art could proxy for production quality and visual polish. Convolutional neural networks can classify art styles, identify graphics quality, and detect production value indicators like detailed textures, complex lighting, and professional character design. These visual signals strongly correlate with development budget and appropriate pricing tiers but are completely absent from our current feature set.
- Developer reputation scores derived from historical performance would quantify the brand recognition gap. By analyzing a developer's past releases, review scores, sales performance, and community engagement, we could create reputation metrics that capture pricing power. Established developers with track records of successful games can command premium pricing that unknown developers cannot, and quantifying this effect would improve predictions substantially.
- Temporal trend features could capture market dynamics and genre popularity shifts. Rather than treating all training years identically, we could engineer features that measure recent price trends in specific genres, quantify market saturation levels, and identify emerging or declining categories. A game entering an oversaturated genre should be priced lower than one entering an underserved niche, and dynamic trend features would capture these effects.

Advanced Modeling Approaches:

- Separate models for different price tiers could handle heterogeneous pricing dynamics more effectively. A classifier could first categorize games into budget, mid-tier, and premium segments, then specialized regressors trained on each segment could predict exact prices. Budget indie games and AAA titles follow different pricing logic, and segment-specific models would avoid forcing a single model to learn contradictory patterns.
- Uncertainty quantification through prediction intervals or probabilistic models would provide actionable confidence estimates. Users could then assess risk appropriately, pricing conservatively when confidence is low and aggressively when confidence is high.

Data Enhancement:

- External data integration could address major feature gaps. Twitch viewership metrics, YouTube trailer view counts, and social media engagement measure pre-release hype that drives pricing power. Steam wishlist counts, follower counts, and discussion forum activity quantify community anticipation. These signals are publicly available but require scraping or API access beyond the static dataset we used.
- Economic indicators like regional purchasing power, currency exchange rates, and gaming market size could contextualize pricing decisions. Games targeting primarily Western markets can command higher prices than those targeting price-sensitive regions, and regional release strategies influence list prices on Steam. Incorporating regional market data would improve predictions by accounting for target audience economics.

References

[1] D. R. Satria, Y. P. Putra, and A. Solichin, "Predicting the Number of Video Game Players on Steam," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 15, no. 12, pp. 378-387, 2024. Available: https://thesai.org/Downloads/Volume15No12/Paper_37-Predicting_the_Number_of_Video_Game_Players.pdf

[2] L. Luisa, M. Mauring, R. Saveljev, and U. Sirts, "Predicting the Popularity of Games on Steam," *arXiv preprint arXiv:2110.02896*, 2021. Available: https://arxiv.org/abs/2110.02896

[3] Y. Huang, H. He, Q. Chen, and L. Wang, "Machine Learning-Based Steam Platform Game's Popularity Analysis," in *Proc. 13th International Conference on Data Science, Technology and Applications (DATA)*, 2024, pp. 350-357. Available: https://www.scitepress.org/Papers/2024/128538/128538.pdf

[4] A. Sujana, M. Kumar, and R. Sharma, "ML Models in Predicting Gaming Popularity," *International Journal for Multidisciplinary Research (IJFMR)*, vol. 6, no. 3, 2024. Available: https://www.ijfmr.com/papers/2024/3/14362.pdf

[5] J. Lin, W. Chen, and S. Wu, "Unveiling Success Drivers in Gaming," *International Journal of Computer Games Technology*, vol. 2025, Article ID 5776632, 2025. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1155/ijcg/5776632

[6] M. Marcelo, S. Santos, and P. Oliveira, "A Novel Approach to Predict Success of Online Games Using Random Forest," in *Proc. International Conference on Innovative Computing and Communications*, Springer, 2022, pp. 25-36. [Online]. Available: https://link.springer.com/chapter/10.1007/978-981-19-1111-8_3

[7] C. Hsu, Y. Chen, and H. Lin, "Estimating Video Game Success using Machine Learning," in *Proc. 2019 International Conference on Machine Learning and Cybernetics*, 2019. [Online]. Available: https://www.researchgate.net/publication/337290854_Estimating_Video_Game_Success_using_Machine_Learning

[8] A. Ramadhanty, B. Wibowo, and C. Pratama, "Predicting Steam Games Rating with Regression," *E3S Web of Conferences*, vol. 388, Article ID 02001, 2023. [Online]. Available: https://www.e3s-conferences.org/articles/e3sconf/pdf/2023/25/e3sconf_icobar2023_02001.pdf

[9] N. Putri and B. Suhartono, "Investigating Impact of Gameplay Hours on Player Recommendations," *International Journal of Research in Marketing*, vol. 8, no. 1, 2025. [Online]. Available: https://ijrm.net/index.php/ijrm/article/view/21

[10] A. Goyal, "Predicting the Popularity of Independent Video Games on Steam," Master's thesis, University of North Carolina at Chapel Hill, 2021. [Online]. Available: https://cdr.lib.unc.edu/concern/masters_papers/s1784v376

[11] thomasmann111111, "Steam Games Data Exploration + Prediction," Kaggle, https://www.kaggle.com/code/thomasmann111111/steam-games-data-exploration-prediction

[12] wysex7, "SteamGames Price Prediction via AutoML," Kaggle, https://www.kaggle.com/code/wysex7/steamgames-price-prediction-via-automl

[13] vedatozturk0, "Steam Game Prices | EDA & Prediction," Kaggle, https://www.kaggle.com/code/vedatozturk0/steam-game-prices-eda-prediction

[14] devraai, "Steam Games Data Analysis and Price Prediction," Kaggle, https://www.kaggle.com/code/devraai/steam-games-data-analysis-and-price-prediction