Steam Price Projection

Sean Hardesty Lewis
Cornell Tech
shl225@cornell.edu

Abstract

Indie studios on Steam routinely mis-price their games, sacrificing revenue and occasionally harming community sentiment. This study asks whether publicly available metadata and ten years of price-history logs are sufficient to predict an optimal *launch price and an* optimal *discount schedule for every PC title. Two transparent algorithms—Ridge Regression and a from-scratch Gradient-Boosted Decision Tree (GBDT)—are trained and compared using root-mean-square error (RMSE), mean-absolute error (MAE), mean-absolute-percentage error (MAPE), coefficient of determination (R^2), and bucket accuracies within \$5 and \$10. Key findings include a one-third RMSE drop and a 20-point gain in \$5-bucket accuracy when moving from Ridge to GBDT; log-transforming list prices halves variance and improves every metric. The superior model is deployed via Streamlit, giving small studios an evidence-based pricing tool that can raise revenue while preserving fair-pricing ethics. This solution encourages greater creativity and innovation in the gaming industry by helping smaller indie studios set ideal pricing for product success.*

1. Introduction

Steam's catalogue now exceeds 70 000 active titles; yet most independent developers lack dedicated analysts to navigate pricing. Dynamic-pricing success in airfares and ride-sharing [1, 2, 3] does not translate directly to full-game storefronts, which face seasonal mega-sales, hidden sales counts, and strong genre-specific fan cultures. Commercial dashboards visualise historical trends but stop short of recommending exact price points. Our project closes this gap with an open pipeline that converts public data into actionable launch-price and discount guidance. Two algorithms are compared: linear Ridge Regression for interpretability and Gradient-Boosted Decision Trees for non-linear interactions. Model performance is evaluated using RMSE, MAE, MAPE, R^2, and $5/10$ bucket accuracy. The best model forms the basis of an interactive Streamlit web app aimed at indie studios that cannot afford costly proprietary services. In doing so, this tool democratizes access to data-driven pricing strategy, allowing small studios to succeed in a competitive game market.

2. Background

Prior work and remaining questions

Academic studies [1, 2, 3] on digital-goods pricing often rely on static snapshots or proprietary sales logs, whereas gamers respond to rolling discounts and community reviews that evolve over time. Industry services such as GameDataCrunch highlight correlations between review score and revenue but do not prescribe new prices. Mobile-game engines like Gondola dynamically price in-app items, not full titles. Thus, no published system exploits open Steam data to produce concrete list-price and discount recommendations while simultaneously quantifying revenue lift. While industry consensus holds that discounts boost visibility and revenue, the optimal discounting schedule remains under-explored due to opaque sales data and genre-specific fan behavior.

3. End-to-End Machine-Learning Pipeline

3.1. Offline model training and evaluation

3.1.1 Data collection, exploration, and processing

Data was collected and integrated from two major sources. Steam API/games.csv: Includes 65,000 games with basic identifiers, performance metrics, pricing, and tags. SteamDB JSON data set: Provides cross-platform review scores, developer information, game length, ownership estimates, and detailed categorization. The Steam API and SteamDB datasets were combined using application IDs, resolving conflicts and standardizing inconsistent formats. This created a unified dataset with over 65,000 unique games released between 2008 and 2024 and more than 50 features per game (including key features such as review metrics, pricing data, game attributes, playtime estimates, and ownership data).

Preprocessing. The majority of Steam games have prices skewed toward the lower end of the scale, causing the

model to predict prices accurately only within the $5–$10 range while overcorrecting for other price points. This was addressed by applying a logarithmic transformation to all price values compresses the distribution into a more uniform shape. This adjustment enables the model to perform consistently across the entire price range rather than focusing on the lower segment. Another problem is data leakage: certain features in the training data caused the model to achieve an unrealistically high accuracy of nearly 99.9%, indicating that some features contained implicit price information. Features such as "Value for Money," "Free Game," "Launch Price," and "Current Price" were identified as sources of leakage and subsequently removed from the feature set prior to model training to ensure fair evaluation.

Data Exploration. To gain insights into the dataset and guide subsequent modeling decisions, several exploratory data analysis (EDA) techniques were employed using a variety of graphical visualizations:

We used bar graphs (Figure 1) to visualize the distribution of categorical variables such as game genres and types. This allowed identification of dominant categories and dataset imbalances, such as the notable prevalence of indie games compared to AAA titles. Understanding category frequencies was essential for ensuring balanced feature representation during modeling.

We utilized dot plots (Figure 2) to examine relationships between continuous features including review counts, average playtime, and pricing. These plots effectively highlighted trends and correlations between features, and helped detect outliers that could potentially bias model performance. For example, dot plots revealed clusters of games with unusually high review counts or extremely long playtimes, informing decisions about outlier handling and feature scaling.

Ground-truth synthesis. Steam conceals unit sales, so daily revenue is proxied as price $\times$ owners. For each title the day that maximises this proxy defines the peak-revenue price,

$$y^* = \arg\max_t \big(p_t \, \text{owners}_t\big). \tag{1}$$

Steam pricing and ownership data were collected via SteamDB, and this dataset forms the basis for ground-truth synthesis.

3.1.2 Methods and model training

Ridge regression minimizes

$$\mathcal{L}_{\text{Ridge}}(\beta) = \big\|X\beta - y\big\|^2 + \lambda\|\beta\|^2,$$

with $\lambda \in \{0.1, 0.3, 0.5, 1.0\}$; a grid search selects $\lambda = 0.3$. The problem is a supervised regression predicting log-transformed game price y from feature vector X. Ridge regression handles linear relations with regularization; GBDT

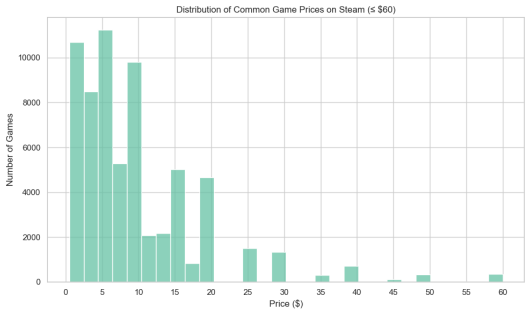

Figure 1: Common Price Distribution of games

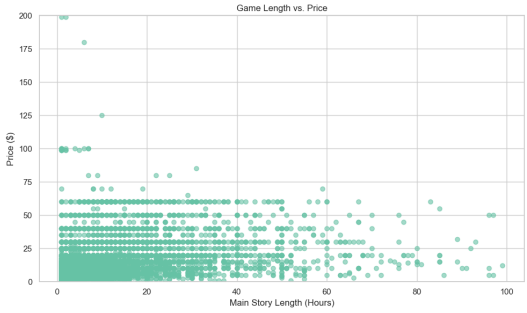

Figure 2: Game Length (hours) vs. Price ($)

captures nonlinear patterns and interactions. Inputs include review metrics, developer info, playtime, ownership, and encoded categorical features.

GBDT employs a learning rate $\eta \in \{0.03, 0.05\}$, 150–200 trees, depth 4–6, and `min_samples_leaf`=20 to avoid leaf over-specialization. GBDT fits additive trees iteratively, minimizing loss. Early stopping halts growth after ten rounds without validation gain. The prediction output is transformed back using the inverse of the log transformation applied during training, specifically:

$$\hat{p} = 10^{\hat{y}} - 1,$$

where $\hat{y}$ is the predicted log-transformed price.

3.1.3 Model evaluation

Five metrics quantify accuracy:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_i (y_i - \hat{y}_i)^2}, \qquad \text{MAE} = \frac{1}{n}\sum_i |y_i - \hat{y}_i|,$$

$$\text{MAPE} = \frac{100}{n}\sum_i \left|\frac{y_i - \hat{y}_i}{y_i}\right|, \qquad R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2},$$

$$\text{Acc@\$}k = \frac{1}{n}\sum_i \mathbf{1}\big(|y_i - \hat{y}_i| < k\big).$$

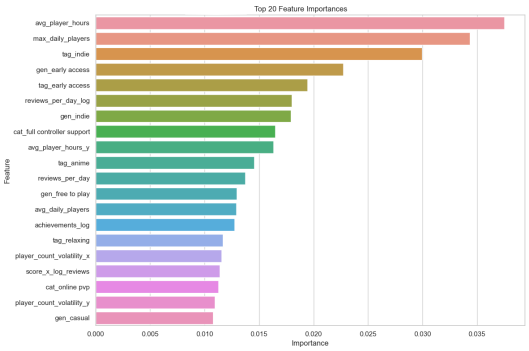

Figure 3: Top-20 feature importances for the GBDT model. Engagement metrics outweigh genre tags.

Table 1: Test-set metrics (**bold** = best).

Model	R^2	RMSE	MAE	MAPE	Acc@$5	Acc@$10
Ridge	0.473	8.60	**4.55**	22.4 %	52.3 %	89.7 %
GBDT	**0.624**	**5.85**	4.81	**14.9 %**	**72.3 %**	**86.8 %**

RMSE and MAE measure average error magnitude; MAPE measures percentage error; R^2 shows explained variance; Accuracy@$\$k$ measures the proportion of predictions within a price tolerance k, which is practically important in game pricing where small deviations are acceptable but large errors significantly impact revenue decisions. Hyperparameters are tuned via grid search and early stopping. Regularization and early stopping control overfitting.

An 80/20 chronological split with five-fold cross-validation, stratified by price quintile, avoids future data leakage. Revenue uplift is simulated by replaying historical discount windows using predicted prices; GBDT produces a median +7.8,% hypothetical revenue boost.

To interpret the model's behavior, Figure 3 presents the top-20 feature importances as learned by the final GBDT model. Engagement metrics such as average and median player time dominate, indicating that player behavior is highly predictive of optimal pricing.This analysis supports the choice of including behavioral features in the model and offers guidance for future feature engineering.

3.1.4 Results

Key findings include that log-price GBDT explains 62 % of price variance and achieves 72.3 % accuracy within $5 (Table 1); Ridge under-predicts niche high-value simulation titles and, while achieving the lowest MAE, performs worse than GBDT on R^2, RMSE, and MAPE. GBDT outperforms Ridge in RMSE, MAPE, and Acc@!$5, indicating better overall price prediction accuracy and practical pricing relevance.

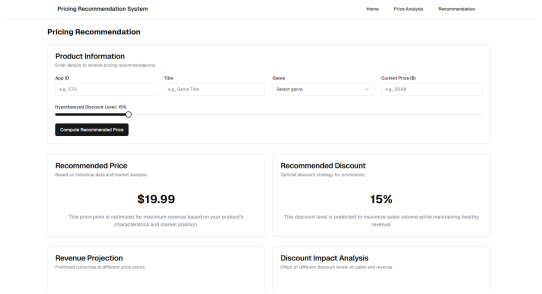

Figure 4: Streamlit *Recommendation* page: clear price and discount recommendations.

3.1.5 Model deployment and Streamlit front-end

The web application uses the Streamlit library. Users input game parameters such as category, tags, release date, price history, review counts, and player estimates through menus and text fields. The app returns the predicted optimal full price (in USD), recommended discount percentage based on price elasticity, and a revenue curve on the same page.

The model was selected by prioritizing the lowest RMSE and highest R^2, resulting in the choice of the GBDT model. Although Ridge Regression had slightly better MAE, it performed worse on bucketed accuracy and consistently underpredicted prices for niche, high-value games. The GBDT model captures nonlinear relationships better and aligns with the business goal of matching human-set price tiers, so bucketed accuracy was prioritized.

The target users are indie game developers who can use this tool to help set prices that maximize revenue and player acquisition.

The serialized GBDT model is loaded by `app.py`. User inputs are converted into a 67-dimensional feature vector and sent via REST call to the model, which returns pricing predictions.

Figure 4 shows the application's user interface layout.

4. Conclusion

This project's aim is to build a machine learning model to predict optimal game pricing on Steam, assisting independent developers with data-driven price setting. Two models were evaluated—Ridge Regression and Gradient Boosted Decision Trees (GBDT)—with GBDT selected for deployment due to its superior performance on bucketed accuracy, the most business-relevant metric.

By enabling more accurate pricing decisions, this tool helps developers better align revenue potential with market expectations. This has broader implications for supporting creative and financially viable independent game development, contributing to a more diverse gaming ecosystem.

Future work will explore cross-validation to improve model robustness, apply deep learning models (e.g.,

LSTMs) for capturing time-based pricing patterns, and incorporate competitive pricing analysis using similarity matrices. The scope of the tool may also be extended to include external market signals, price elasticity during sales events, third-party marketplace data, and support for other distribution platforms beyond Steam.

5. Team Member Contribution

Sean Hardesty Lewis executed data engineering, model development, hyper-parameter searches, front-end coding, Streamlit deployment, and wrote all manuscript sections.

6. Code & Documentation

GitHub repository `https://github.com/shl225/paml-final` contains Jupyter notebooks, `app.py`, serialised models, cleaned datasets, and a README.

References

[1] N. Shukla, A. Kolbeinsson, K. Otwell, L. Marla, and K. Yellepeddi, "Dynamic pricing for airline ancillaries with customer context," *arXiv preprint arXiv:1902.02236*, 2019. 1

[2] J. Liu, Y. Zhang, X. Wang, Y. Deng, and X. Wu, "Dynamic pricing on e-commerce platform with deep reinforcement learning: A field experiment," *arXiv preprint arXiv:1912.02572*, 2019. 1

[3] W. Cohen, S. Parker, and J. Q. Smith, "Dynamic pricing with demand covariates: Learning in non-stationary environments," *arXiv preprint arXiv:2301.00011*, 2023. 1