ST3189: Assessed Coursework Project

Student number: 240626323

Table of Contents

1. Introduction

For the Assessed Coursework Project, I have chosen the "Steam Games Dataset" released on Kaggle by Martin Bustos Roman (2022), an indie game developer. This dataset is a cross-sectional data collected in 2024 and contains information of games published on the Steam platform, the largest gaming platform for personal computers. The author states that the data was mainly collected directly from Steam's web API and some extra data from the "Steam Spy" web application developed by Galyonkin, S., which provides estimates including the number of owners in a game and the average total playtime using the online profiles of game owners with an estimation of around 0.1% margin of error. This project focuses on majority of the paid games published and sold on the Steam platform and has defined three research questions from this dataset:

i. How can we effectively group user-generated tags into smaller, less redundant groups?
ii. Given some data which could be gathered before publishing a game, how could we estimate the number of recommendations a game would receive in the Steam platform?
iii. Given some data of a game, what price level should the game most likely be priced at?

Several Machine Learning models were provided throughout this project to help address these questions.

2. Dataset and Explorative Data Analysis

The whole dataset was split into two parts: first part is the Games Dataset, which contains core game metadata (e.g. price, playtime, release date), and the second part is the Tags dataset, which stores user-generated tags for a game and their voting frequencies. Both datasets were cleaned separately for efficiency. The Tags Dataset was then filtered to include only games present in the Games Dataset and used for unsupervised clustering. The final dataset includes **5,589 games** with features like: Playtime Metrics, these includes the Median/average playtime, and some Derived Features, these includes estimated skewness of playtime, proportion of positive reviews, industry leader flag, and categorical price tiers derived from the game's price. Table 1 from the appendix summarizes the games dataset features and datatype.

The median playtime instead of mean playtime was used as the playtime metric because it is less skewed by outliers, and correlates strongly with the count of recommendations. The industry leader indicator (*'industryLeader'*) was derived to indicate games that are published or developed from major studios (e.g., Ubisoft, EA, Bandai) and have higher prices and recommendations on average (Figure 1.1)**.** Games that are free or with recommendations count above 100,000 (~exp(11.51)) were out of scope and was removed. The price was categorized into Low (<$10), Medium ($11-20), and High (>$20). There are 2,704 games in the 'Low' class, 1,829 games in the 'Medium' class, and 1,094 games in the 'High' class. Boxplots in Figure 1.2 shows that more recommendations are associated to higher level of price.

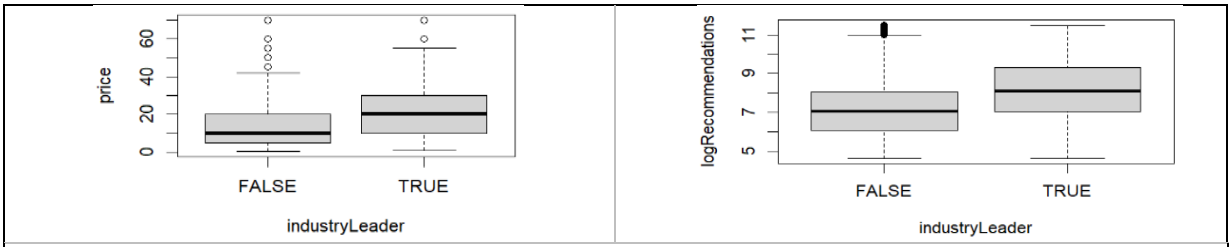


Figure 1.1, Left: *Price of games based on industry leader indicator,* **Right:** *Log recommendations of games based on industry leader indicator.*

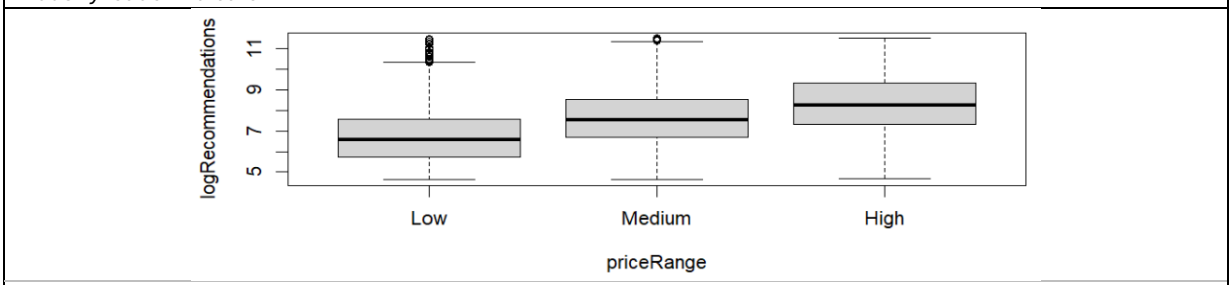


Figure 1.2: *Boxplot of log-transformed recommendations based on different levels of price range.*

The extra downloadable content count (*'dlcCount'*) of games was capped at 30 to ignore niche cases, most games having either zero or under 10 DLC. The boxplot of log recommendations based on different *'dlcCount'* and the count of supported languages for a game *'languagesCount'* is shown in **Figure 1.3**, the left side of this figure indicates that DLC count have some positive association with log recommendations, but the association is not so clear after the 13th DLC as there is not much data for this type of games. On the right side of this figure, the *'languagesCount'* were shown to have some positive association with log recommendations as well.

The estimated playtime skewness (*'estimatedSkew'*) feature was derived using the relative difference between the mean and the median playtime as a rough approximation of the magnitude and direction of the skewness in playtime distribution, it is defined as: *(Mean – Median) / Mean*. Games with an estimated skewness of zero was removed, since a game with the exact same mean and median total playtime minutes would most likely happen only in cases of very small sample sizes for a given game, and the information of that game would likely be inaccurate. The estimated skew versus log recommendations and proportion of positive reviews scatterplot is shown in **Figure 1.4,** The scatterplot on the left side shows that games with a positive skew have higher recommendations count. Similarly, the right side of the figure shows that games with higher proportion of positive reviews have higher recommendations.

To further understand the relationships between features, I have performed Principal Component Analysis on some chosen features that was first scaled to have unit standard deviations. The left side of **Figure 1.5** shows a correlation plot indicating pairwise correlations between features, while the right side displays a principal component analysis biplot of several features projected on the first two principal components, the first two Principal Components explains around 48% of the variance in the data. In the biplot, vectors pointing in similar directions indicate positive correlations between the features, whereas vectors pointing in opposite directions indicate negative correlations.

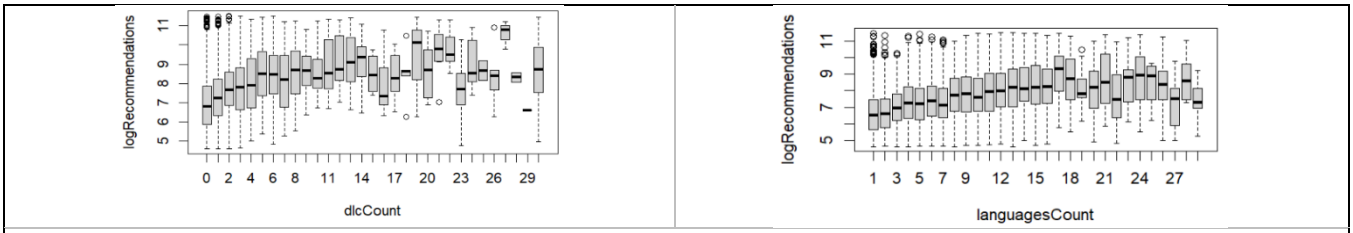


Figure 1.3, Left: *Box plot of log Recommendations based on different DLC counts.* **Right:** *Box plot of log Recommendations based on different supported languages counts.*

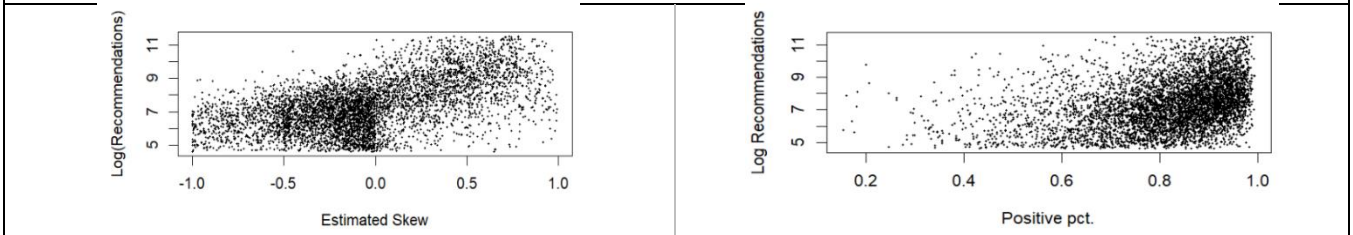


Figure 1.4, Left: *Scatterplot with estimated skew on the X-axis and log recommendations on the Y-axis.* **Right:** *Scatterplot with positive ratio on the X-axis and log recommendations on the Y-axis.*

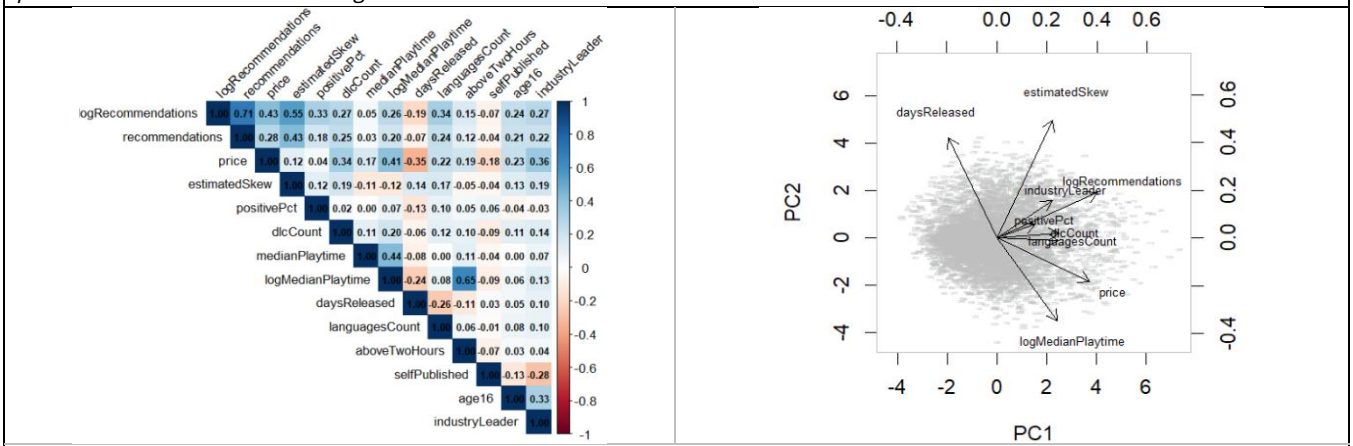


Figure 1.5, Left: *Correlation plot of some chosen features.* **Right:** *Biplot of some features and its loadings on the first two PCs.*

2.1. Correlation between features and Initial Features selection

For the regression task: Based on using *'recommendations'* as the target variable, the correlation plot and PCA biplot in Figure 1.5 shows that *'estimatedSkew'*, *'price', 'positivePct', 'languagesCount', 'age16', 'dlcCount'* and *'industryLeader'* has the strongest correlation with the log target variable, also, log transformed *'medianPlaytime'* have a higher correlation with the log target variable compared to its original values. These features were selected as predictors for the regression task.

For the classification task: Based on using *'priceRange'* as the target variable, *'dlcCount', 'daysReleased', 'languagesCount', 'age16', 'industryLeader', 'selfPublished'* and *'aboveTwoHours'* have relatively strong correlation with the *'price'* variable, also the log transformed *'medianPlaytime'* and *'recommendations'* have a much higher correlation than its original value. These features were selected for the classification task.

2.2. Tags Data

The tags dataset contains the last feature of the whole dataset, which is the "tags" feature. This feature contains a list of tags which are user generated or voted to a game, as well as the frequency of votes associated with each tag. Each tag helps define a game's core mechanics and style of play (e.g. 'Action', 'Strategy', 'Sports', etc.), and the frequency reflects how often that tag is voted by users. It is first extracted in a long format, where each row contains the game ID, a single tag, and the frequency of that tag, it was then pivoted into a wide format so that each tag is represented as a variable for subsequent data processes and machine learning tasks. An example of what the data looks like is given in Figure 2.

appID <chr>	tag <chr>	frequency <dbl>
1026420	Tactical.RPG	255
1026420	Turn.Based.Strategy	248
1026420	Wargame	248
1026420	Historical	239
1026420	Strategy.RPG	236
1026420	Perma.Death	231
1026420	RPG	225
1026420	Difficult	221

appID <chr>	Action.RPG <dbl>	Strategy <dbl>	Point...Click <dbl>	Puzzle <dbl>
219990	34590	0	0	0
392110	0	30531	0	0
370910	0	0	13760	54
231430	0	8346	0	0
252130	0	58	0	7706
285800	0	7582	0	0

Figure 2, Left: *An example of what the original long format looks like,* **Right:** *Tags dataset pivoted to wide format.*

3. Research Questions and Machine Learning tasks

Three research questions emerge from this dataset. Firstly, user-generated tags exhibit significant redundancy, with many closely related tags that could be grouped into broader categories (e.g., ['Multiple Endings', 'Interactive Fiction', 'Choices Matter'] under 'Choice-driven', or ['Horror', 'Psychological Horror', 'Dark', 'Gore'] under 'Horror'). While humans can create these groupings subjectively, the challenge rises as the number of unique user-generated tags grows. The question becomes: how can we effectively group these user-generated tags into less redundant clusters? These clusters could then be used to provide game recommendations based on user-searched tags and their associated groups. This question is addressed in the unsupervised learning section.

Secondly, games with different levels of skewness in their users' playtime distributions receive varying numbers of recommendations. Additionally, game positioning, which is shaped by factors like tags and pricing affects recommendation counts, which also directly influence copies sold and revenue generated. The question becomes: given some data features which could be gathered and estimated before publishing a game, how could we estimate the count of recommendations a game would receive? This question is addressed in the regression learning section.

Lastly, different price levels (low, medium, high) correlate with various contributing factors such as playtime, publisher popularity, age requirements, and game categories. The question becomes: Given some data of a game, what price level should the game most likely be priced at? The answer would provide a game positioning reference and indicate a game's approximate worth. This question is addressed in the classification task section.

4. Unsupervised Learning

For the unsupervised learning component, I used user-generated tag frequencies from the tags dataset to identify clusters of similar tags. Hierarchical clustering groups tags based on pairwise distances. Initially, I used Euclidean distance as the metric, assuming that tags frequently appearing in the same games would have smaller distances and cluster together. However, this approach was problematic due to zero-inflation and popularity bias in tag frequencies. With 433 unique tags and each game limited to 20 tags, many tags had low or zero frequencies across games, leading to biased clusters. Additionally, tags from popular games had large Euclidean distances, causing them to cluster in one group despite being different tags contextually. For example, using Euclidean distance with complete linkage grouped most tags into one cluster, while popular tags like "Multiplayer" and “Casual” formed their own cluster, the clustering process could be visualized through a dendrogram in Figure 3 **(left).**

4.1. Correlation based Hierarchical Clustering

To address these issues, I used a correlation-based distance measure, focusing on tag relationships rather than raw frequencies. The resulting dendrogram **(**Figure 3**, right)** revealed seven coherent clusters (highlighted in red boxes), such as comedy-related tags are grouped in one cluster and single-player, roleplaying tags are grouped together in another cluster. This method mitigated zero-inflation and popularity bias by emphasizing correlation rather than Euclidean distances. The results showed that correlation-based clustering produced more meaningful groupings compared to Euclidean distance.

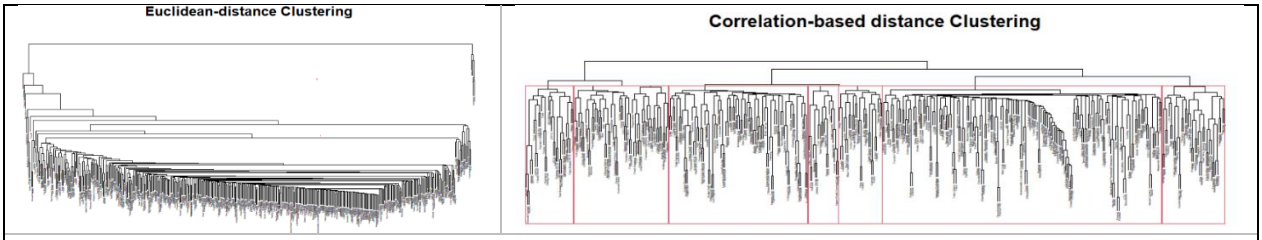


Figure 3, Left: *Hierarchical clustering using Euclidean distance.* **Right:** *Clustering using correlation-based distance (boxed clusters highlight groupings).*

The following table summarizes the types or themes of game tags within each cluster represents:

Cluster	Cluster Theme	Included tags example
1	Dark Fantasy Action RPGs	*Action.RPG, Hack.and.Slash, Dark.Fantasy, PvE, Loot, Medieval, Dungeon.Crawler, …*
2	Realistic Strategy & Simulation	*Strategy, Real.Time.Tactics, RTS, Grand.Strategy, Wargame, Racing, City.Builder, Simulation, …*
3	Narrative Adventure & Metroidvania	*Adventure, Metroidvania, Horror, Great.Soundtrack, Story.Rich, Walking.Simulator, Pixel.Graphics, Choices.Matter, …*
4	Niche Indie Games	*Puzzle, MMORPG, Indie, Artificial.Intelligence, Sailing, Sports, Psychedelic, Jet, Steampunk, Parkour, Logic, …*
5	Open World Survival Multiplayer	*FPS, Space.Sim, PvP, Open.World, Survival, Open.World.Survival.Craft, Multiplayer, First.Person, Team.Based, Co.op, Shooter, Sci.fi Science ,…*
6	Adult Fantasy, Turn Based RPGs	*RPG, Nudity, Mature, Turn.Based.Tactics, LGBTQ, Romance, Female.Protagonist, Fantasy, JRPG, Character.Customization, Turn.Based.Strategy, Turn.Based, …*
7	Dark Humour Action Combat	*Comedy, Action, Funny, Zombies, Gore Combat, Violent, Dark.Humor, Action.Adventure, Third.Person, Blood, Fast.Paced, Satire , …*

4.2. Principal Component Analysis

In an attempt to refine tag clustering, I applied Principal Component Analysis (PCA) to reduce dimensionality and minimize noise in the tag frequency data before grouping. PCA identifies linear combinations of tags that maximize data variance, with tags that frequently appear together loading strongly on the same components. The scree plot in Figure 4.1 shows the cumulative proportion of variance explained by each additional principal component. I selected the first 252 principal components, which together explain 90% of the variance in the data. Figure 4.2 shows how different tags are projected onto the space of the first two principal components, along with a dendrogram of the hierarchical clustering results using correlation-based distances of the projected scores. The same colour in these plots indicates which cluster each tag was assigned to when grouped into 7 clusters.

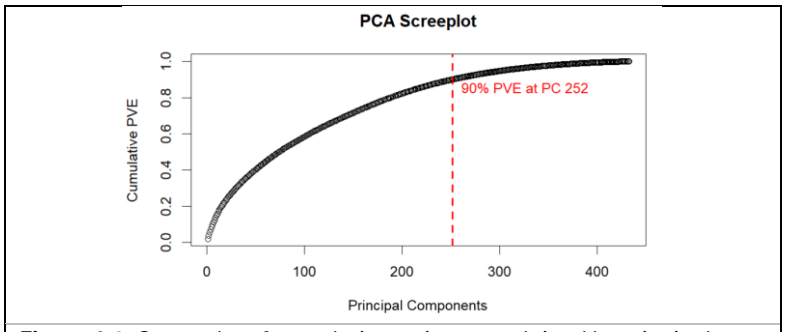


Figure 4.1, *Scree plot of cumulative variance explained by principal components.*

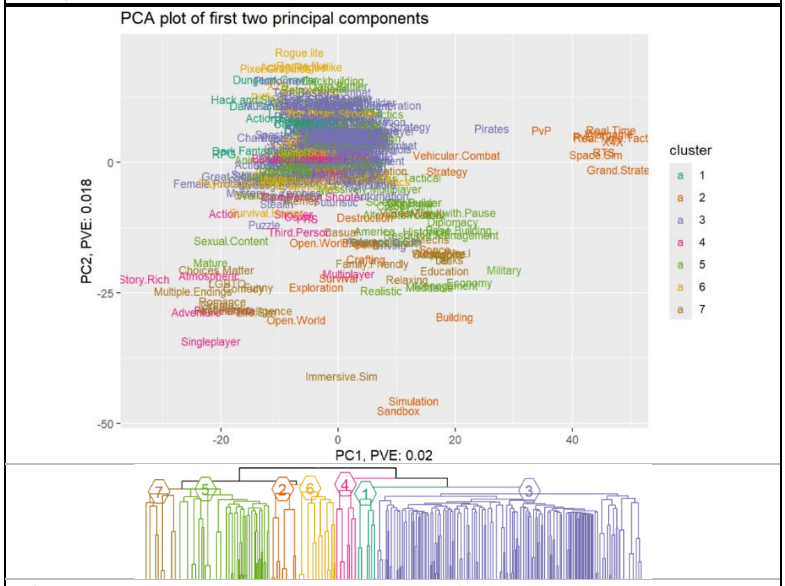


Figure 4.2, Top: *Scatterplot of tags on the first two components;* **bottom:** *Dendrogram of clusters from correlation of tags based on PC scores.*

When hierarchical clustering was applied to the correlation-based distance matrix derived from the principal component (PC) scores of tag frequencies, the resulting cluster assignments slightly matched those from clustering the correlation matrix of raw tag frequencies directly, but due to concerns with zero-inflation, which might distort principal components, and the fact that there is not much reduction in data dimensionality (from 433 to 252 features), I have chose to use the clustering results from the original correlation matrix of raw tag frequencies.

After grouping all 433 tags into 7 clusters, I assigned each game's tags to their respective clusters and calculated cluster weights and counts. The weights were computed by normalizing each cluster's frequency relative to the total frequency across all clusters of the game. When these features were merged with the game dataset, a correlation plot (**Figure 5**) revealed that the number of clusters per game had a moderate association with recommendation counts, suggesting that games linked to more clusters tend to receive more recommendations. Additionally, Cluster 4 and 5 showed moderate correlations with the target variables (*'recommendations'* and *'price'*).

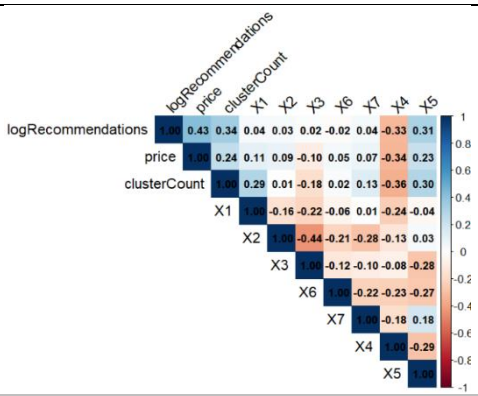


Figure 5, *Correlation plot of target variables and different clusters weights and count.*

5. Regression Task

In the regression task, the goal is to estimate the number of recommendations a game might receive based on available data. Exploratory data analysis revealed that several features (e.g., price, playtime metrics) were associated with recommendation counts. After clustering tag frequencies, cluster weights and counts also showed moderate correlations with recommendations, which is why I have included cluster weights and cluster counts as additional variables for this part. This section discusses results from different regression models.

5.1. Negative Binomial Regression model

The Negative Binomial model is suitable here because the target variable (recommendations) is a count with high variance relative to its mean, indicating overdispersion. It assumes linear relationships between the log-transformed target and predictors. Since selected features had moderate linear associations with log recommendations, this model fits the problem well. Given the inclusion of up to 17 features, I performed subset selection to identify a smaller set of strong predictors, simplifying the model and reducing overfitting risk.

5.1.1. Subset selection and Cross Validation

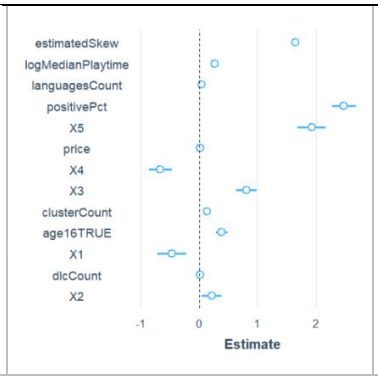
There are various subset selection methods available, due to the large number of features selected, a best subset selection algorithm would not be feasible here as there are 2^17 possible combination of features to be evaluated. A Forward-Stepwise Selection algorithm was used instead as a better alternative for feasibility. The forward-stepwise selection method starts with no features and sequentially adds the one that most improves performance. For the Negative Binomial Regression Model, I have chosen the Akaike Information Criterion as the metric to evaluate models for each step, and the results are shown in **Figure 6**.

After using a training partition of the whole data to perform forward selection, 13 features were chosen out of the 17 features in the initial full model, and all features were statistically significant (P-values < 0.05). The summary of the model could be seen on the left side of **Figure 6**, a dot whisker plot of the coefficients is shown on the middle of the figure. Note that variables starting with '*X*' and a subsequent number represents the cluster weight variables. The final trained model uses the log link function where y = exp (intercept + βX), the dispersion parameter is estimated to be 1.29, which indicates that the assumption of overdispersion is fulfilled. Variables such as *estimatedSkew, logMedianPlaytime, languagesCount, positivePct, X5, X3, X2, clusterCount* and *age16* are associated with an increase in the expected count of recommendations. For example: a 1-unit increase in *'languagesCount'* is associated with an increase of exp (0.04), or around 4% increase in the expected count of recommendations. The partial effect of *'estimatedSkew'* on the target variable could be visualized on the right side of the figure, this shows how changes in *'estimatedSkew'* influence the model's predictions, holding the effects of all other variables constant. It could be seen that the predicted recommendations count grows exponentially as the estimated skew increases. Similarly, cluster weight variables such as *X4* and *X1* have negative coefficients, indicating a decrease in the expected count.

```
Coefficients:
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)       2.788520   0.133944  20.818  < 2e-16 ***
estimatedSkew     1.651871   0.033759  48.931  < 2e-16 ***
logMedianPlaytime 0.270035   0.014182  19.041  < 2e-16 ***
languagesCount    0.042923   0.002539  16.904  < 2e-16 ***
positivePct       2.482547   0.104810  23.686  < 2e-16 ***
X5                1.928823   0.124934  15.439  < 2e-16 ***
price             0.014065   0.001274  11.043  < 2e-16 ***
X4               -0.663502   0.098910  -6.708 1.97e-11 ***
X3                0.809634   0.092694   8.734  < 2e-16 ***
clusterCount      0.134362   0.012206  11.008  < 2e-16 ***
age16TRUE         0.388600   0.052082   7.461 8.57e-14 ***
X1               -0.468189   0.124969  -3.746 0.000179 ***
dlcCount          0.015627   0.003354   4.660 3.17e-06 ***
X2                0.215163   0.086243   2.495 0.012601 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for Negative Binomial(1.2904) family taken to be 1)

    Null deviance: 14393.0  on 4475  degrees of freedom
Residual deviance:  5023.8  on 4462  degrees of freedom
AIC: 78491
```



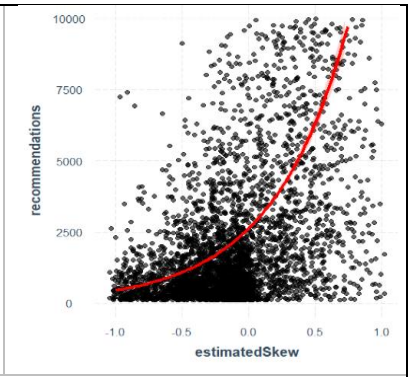


Figure 6, Left: *The summary output of the final model after performing Forward Selection with AIC as criterion.* **Middle:** *Dot whisker plot of the coefficients.* **Right:** *Partial plot showing the effect of 'estimatedSkew' on the prediction, holding all other features constant.*

5.1.2. Model Results on unseen data

After training the final Negative Binomial Regression model on the full training dataset with selected features and testing it on independent data, the model achieved an RMSE of 8,438 and MAE of 3,621, which is lower than the baseline regression model with RMSE of 10,675 and MAE of 5,964. I analysed residuals by computing Pearson residuals (residuals standardized by the model's variance assumptions) and plotted predicted values vs. actual values and predicted values vs. Residuals (Figure 7). The plots showed that Pearson residuals were larger for lower predicted values and smaller for higher predicted values, which aligns with the expected behaviour in count data models since the variance increases with the mean. And a histogram shows that residuals is centred around zero. These findings confirm the model's suitability for predicting recommendation counts, as it properly accounts for overdispersion in the data.

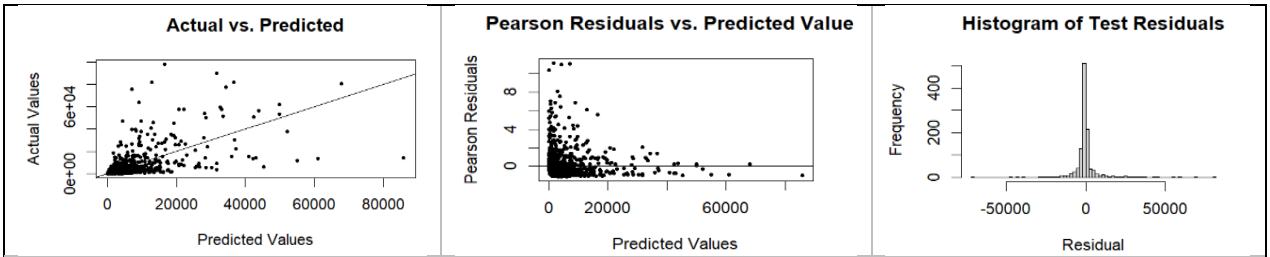


Figure 7, Left & Middle: *Scatterplots of predicted values versus actual values, as well as versus Pearson residuals. The straight line represents the perfect prediction line* **Right:** *Distribution of the residuals.*

5.2. XGBoost Regression Model

The next model I tested was the XGBoost regression model. XGBoost (Extreme Gradient Boosting) is an advanced tree-based ensemble learning algorithm that builds decision trees sequentially while minimizing a loss function. Its gradient-boosting approach enables it to capture complex non-linear relationships between variables, making it highly flexible and robust against overfitting through regularization techniques. I began by performing hyperparameter tuning with cross-validation on the training partition of the data. The tuned parameters included the number of boosting rounds, maximum tree depth, and the fraction of sampling sizes. After identifying the optimal parameters, I trained the final model using train data and made predictions using the test data. Results showed an RMSE of 8,181 and MAE of 3,773, which is slightly better than the Negative Binomial Regression model. Figure 8 shows the plots of residuals and the distribution of test residuals.

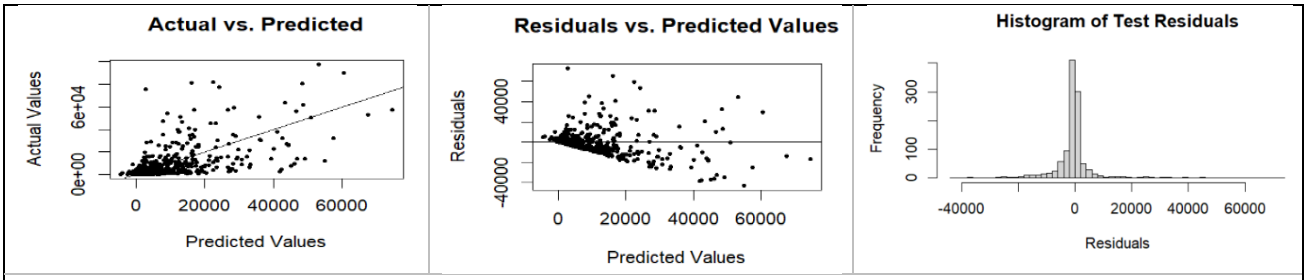


Figure 8, Left & Middle: *Scatterplots of predicted values versus actual values, as well as versus raw residuals. The straight line represents the perfect prediction line* **Right:** *Distribution of the residuals.*

5.2.1. Feature Importance

Based on the feature's contribution in decreasing the node impurity when it is used to split a node, an importance ranking of the features could be computed from this final trained model. The top ten features from most important to least important are shown on the left side of Figure 9**.** The scatterplots in the middle and right side of the panel shows how two of the most important features (*'estimatedSkew'* and *'logMedianPlaytime'*) affects the model's prediction on recommendation. It could be seen that in general, a higher estimated skew or playtime leads to exponentially more recommendations.

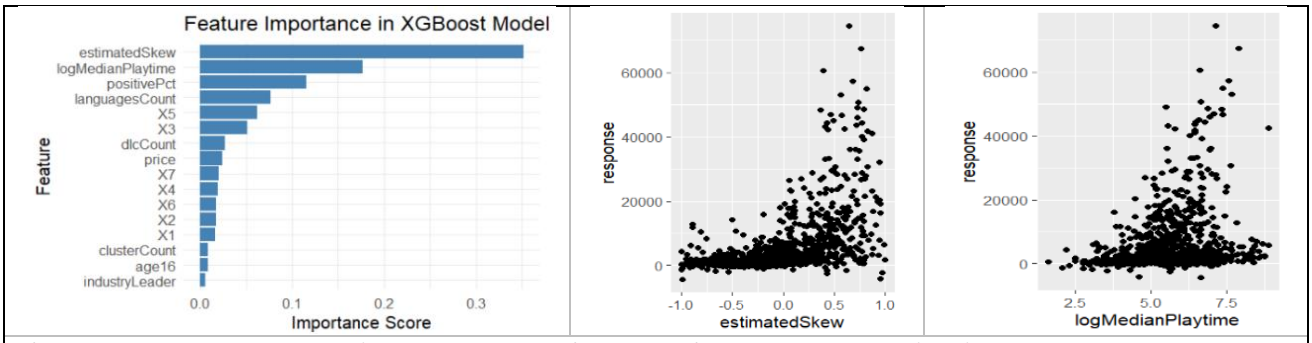


Figure 9, Left: *Bar plot of feature importance score.* **Middle and Right:** *Scatterplots to visualize effects of the two most important features on the prediction of the response.*

6. Classification Task

For the classification task, the main research question is: Given some features of a game, what price level should the game most likely be priced at? Prices are categorized as Low (under $10), Medium ($10–$20), or High (over $20). The target variable has class imbalances: most games fall into Low or Medium, with High accounting for roughly 20%. To address imbalances, stratified sampling was used to preserve the original class distribution in training and testing partitions, and class weights inversely proportional to their frequencies were applied to each model when training so that higher penalties were given to misclassifications in minority classes. Features selected for prediction include *'dlcCount', 'daysReleased', 'languagesCount', 'age16', 'industryLeader', 'selfPublished', 'aboveTwoHours', 'logMedianPlaytime', 'logRecommendations', 'clusterCount',* and *Cluster weight variables*.

6.1. Random Forest Classification

One of the models I tested for the classification task was the Random Forest classifier. Random Forest is a ensemble learning method that constructs multiple decision trees and aggregates their results through majority voting. Its ensemble approach reduces overfitting risk compared to single decision trees while capturing complex non-linear relationships in the data. I began by performing hyperparameter optimization using grid search with cross-validation on the training partition. Hyperparameter tuning focused on optimizing the number of trees, maximum tree depth, and minimum node size. These parameters were adjusted to minimize classification log loss, a metric that evaluates prediction confidence and accuracy across all classes. The final tuned model demonstrates stronger performance predicting Low and High classes compared to the Medium class. The precision, recall scores, and a confusion matrix of the results are shown in Figure 10.

truth			
response	Low	Medium	High
Low	441	133	16
Medium	82	170	68
High	17	61	131

```
Precision scores:
 Low: 0.75
 Medium: 0.53
 High: 0.63
 Average Precision: 0.63
```

```
Recall scores:
 Low: 0.81
 Medium:  0.47
 High: 0.61
 Average Recall: 0.63
```

Figure 10, Left: *Confusion Matrix of Random Forest Classifier,* **Middle and Right:** *Precision & Recall scores for each class as well as aggregated scores.*

6.1.1. Feature importance

Feature importance was calculated based on each feature's contribution to reducing node impurity during splits in the final trained random forest model. The top ten features ranked from most to least important are shown in the left side of Figure 11. The plot on the middle and right panel shows the Predicted vs Actual price range on test data using the two most important features as axes for visualization: *logMedianPlaytime and daysReleased*. Generally, lower values for playtime combined with higher days released associates with lower predicted price ranges. However, it is shown that the model underestimated the price range of a lot of games that are actually in the medium class.

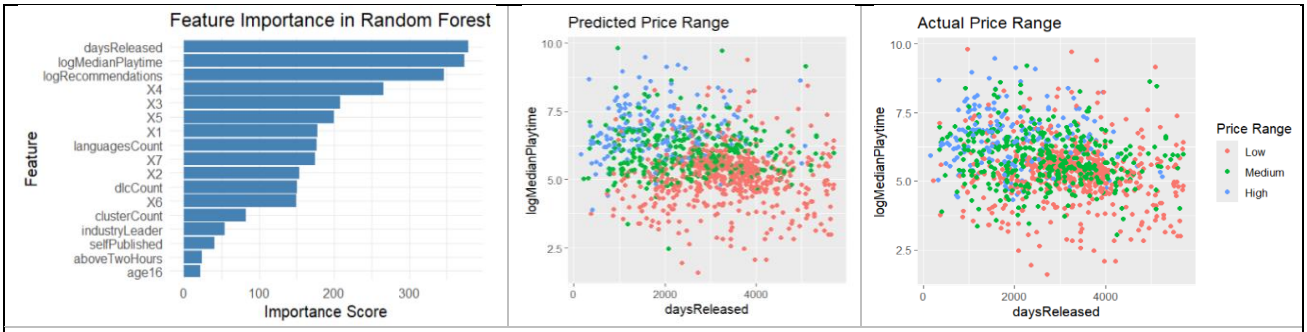


Figure 11, Left: *Bar plot of feature importance.* **Middle and Right:** *Comparison of Predicted (Middle) vs Actual (Right) price ranges using features 'logMedianPlaytime' and 'daysReleased' for visualization.*

6.2. Support Vector Machine

The Support Vector Machine (SVM) classifier identifies the optimal hyperplane (or decision boundary) that separates data points from different classes in high-dimensional space. This hyperplane is selected to maximize its distance from the nearest data points (support vectors), creating the widest possible margin. For predicting price ranges using game features and tag clusters, classes may not be linearly separable. The radial kernel trick addresses this by enlarging the feature space in a specific way, implicitly projecting data into a higher-dimensional space where separation is possible. The Support Vector Machine algorithm can then find the decision boundaries that maximize the margin between classes. Since I have three classes (Low, Medium, High), the algorithm would use a One vs One strategy by default, which would create 3(3-1)/2 = 3 binary classifiers: Low vs Medium, Low vs High, Medium vs High, and the final classification would be the class to which it was most frequently voted in these pairwise binary classifications. There were two important parameters to tune over in a SVM model, the 'cost' and 'gamma'. The cost parameter represents the penalty of misclassification and controls the trade-off between bias and variance, and the gamma parameter controls how much influence does the support vector have on the decision boundary smoothness. Higher cost allows narrower margins to reduce errors, while lower gamma creates smoother boundaries.

Like the Random Forest model, performance was slightly weaker on the medium class compared to Low and High classes, but the recall scores were lower than the Random Forest model for the High class. The precision and recall scores are presented in **Figure 12** alongside the confusion matrix. **Figure 13** illustrates the comparison of Predicted vs Actual price ranges of test observations using the features *'logMedianPlaytime'* and *'daysReleased'* for visualization. It could be seen that the predicted price ranges for this model were quite similar to the predicted price ranges in the random forest model under these variables.

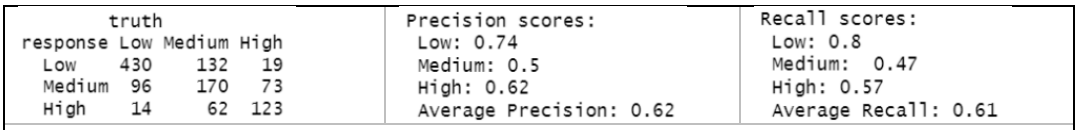


Figure 12, Left: *Confusion Matrix,* **Middle and Right:** *Precision & Recall scores for each class as well as aggregated scores.*

Figure 13, *Comparison of Predicted (Left) vs Actual (Right) price ranges using features 'logMedianPlaytime' and 'daysReleased' for visualization.*

7. Summary

This study analysed the Steam Games Dataset to address three research questions through machine learning: grouping user-generated tags, predicting recommendation counts, and classifying price ranges. Unsupervised clustering of tags using correlation-based distances revealed seven meaningful clusters, mitigating issues from zero-inflation and popularity bias. Additionally, Principal Component Analysis was attempted but were not useful in the end due to not much meaningful reduction in dimensions and concerns for zero inflation related issues. For the regression task, a Negative Binomial Regression model achieved RMSE 8,438 and MAE 3,621, while XGBoost regression showed slightly better performance (RMSE 8,181). In classification, SVM and Random Forest models achieved similar precision and recall scores, with stronger performance on "Low" and "High" price tiers and less reliable predictions for “Medium” price tier.

The regression models identified several significant predictors of recommendation counts. Playtime skewness, measured as the difference between mean and median playtime, demonstrated a strong positive relationship with recommendations. Games with higher skewness, indicating a small proportion of players engaging deeply with the content, tended to receive exponentially more recommendations, suggesting that games that can form deeper engagement drives up recommendations. Median playtime also correlated positively with recommendations, reflecting that games requiring substantial time investment are perceived as higher value and more likely to be shared. Cluster weights, particularly for genres like open-world survival (Cluster 5) had a strong positive effect on recommendations, while genres like niche indie (Cluster 4) exhibited negative associations, potentially reflecting their need for more specialized audiences. These findings suggest that developers aiming to maximize recommendations should prioritize gameplay depth and alignment with popular genres.

In the classification task, the strongest predictors of price ranges from the random forest model were logMedianPlaytime, releasedDays, logRecommendations, and Cluster 4. LogMedianPlaytime is positively correlated with higher prices, suggesting that games offering longer playtimes are perceived as premium products. In contrast, the number of days since release shows a slightly negative relationship with price tier, with newer games tending to be priced higher. Additionally, a high proportion of tags grouped in Cluster 4 (representing niche indie games) is negatively correlated with price, indicating that niche or indie games are generally less expensive. These findings imply that pricing strategies are influenced by gameplay duration, time since release, and genre characteristics. Both the SVM and Random Forest classifiers demonstrated robust performance in predicting the ‘Low’ price tier, achieving approximately 74% precision and successfully identifying 80% of all actual ‘Low’ cases. For the ‘High’ price tier, the models showed moderate effectiveness, with around 62% precision and recall, despite this being a minority class. However, the models performed less effectively for the “Medium” price tier, exhibiting notably lower precision and recall. This suggests that the current feature set may lack sufficient discriminative power for this category, and future improvements could benefit from incorporating additional relevant features. Overall, the classification models performed well and could be practical in determining the minimum price at which a game should be offered.

8. Reference:

1. Bustos Roman, M. (2022). Steam Games Dataset [Data set]. Kaggle. https://doi.org/10.34740/KAGGLE/DS/2109585
2. Galyonkin, S. (n.d.). Steam Spy. https://steamspy.com
3. Bischl, B., Sonabend, R., Kotthoff, L., & Lang, M. (Eds.). (2024). *Applied Machine Learning Using mlr3 in R*. CRC Press. https://mlr3book.mlr-org.com
4. K.Kalogeropoulos (2022). ST3189 Machine Learning Subject Guide. University of London.
5. James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021). *An Introduction to Statistical Learning with Applications in R* (2nd ed.). https://www.statlearning.com/

9. Appendix:

Feature Name	Data Type	Description	Remarks
appID	Character	Index of the game	Used as key index to join with the tags dataset
Name	Character	Name of the game	
releaseDate	Date	Game's released date in Steam	Until '2024-07-01', data was collected at Oct 2024
Price	Numeric	Current price of the game	Only price above zero are included in this project
Positive	Numeric	Positive votes	
Negative	Numeric	Negative votes	
Recommendations	Numeric	Recommendation counts from owners	
medianPlaytime	Numeric	Median of total minutes spent by players on the Game	
averagePlaytime	Numeric	Average of total minutes spent by players on the Game	
Genres	Character	A list of genres describing the game, stored in a string	
Languages	Character	A list of languages supported by the game, stored in a string	
Required_age	Numeric	Minimum age requirement for the game	
estimatedOwners	Factor	Estimated range of owner count	
dlcCount	Numeric	Count of extra downloadable contents	
Achievements	Numeric	Count of achievements available in a game	
Publishers	Character	A list of publishers of the game, stored in a string	
Developers	Character	A list of developers of the game, stored in a string	
developerCount	Numeric	Count of developers	Derived from the `developers` feature
publisherCount	Numeric	Count of publishers	Derived from the `publishers` feature
genresCount	Numeric	Count of genres	Derived from the `genres` feature
languagesCount	Numeric	Count of supported languages	Derived from the `languages` feature
logRecommendations	Numeric	Log-transformed count of recommendations from owners	Derived from the `recommendations` feature
logDlcCount	Numeric	Log-transformed count of extra downloadable contents	Derived from the `dlcCount` feature
logAchievements	Numeric	Log-transformed count of achievements in a game	Derived from the `achievements` feature
positivePct	Numeric	Proportion of positive votes relative to all votes	Computed as: Positive / (Positive + Negative)
estimatedSkew	Numeric	An estimated measure of playtime skewness	Computed as: (averagePlaytime – medianPlaytime) / averagePlaytime
PriceRange	Factor	Set range of price which the game belongs in	Derived from the `price` feature
Age16	Logical	Indicates whether the age requirement is above 16	Derived from the `required_age` feature
industryLeader	Logical	Indicates whether the game's developers/publishers contains a industry leader.	Derived from the `developers` and `publishers` feature. Industry leader is determined by domain knowledge and online research.
AboveTwoHours	Logical	Indicates whether the game's median playtime exceeds two hours.	Derived from the `medianPlaytime` feature

Table 1, *this table includes extra features that were engineered from other original features in the dataset, some of the features that were included in the original dataset were not included here since it is not analysed or used at all throughout the project.*