CHICAGO BOOTH

The University of Chicago Booth School of Business

BUSN 41202 - Big Data
Spring 2024
Professor Veronika Ročková

Leveling up: A Deep Dive into the Optimal Strategies Behind Microsoft's $69B Gaming Move

What games should Microsoft go for next? Is Microsoft best positioned to meet these trends?

26 May 2024

G12: Yufei Liu, Kathy Zhang, Liujun Hua

Overview and Section Breakdown

1. Executive Summary

Microsoft's acquisition of Activision Blizzard (Nasdaq: ATVI), a prominent publisher known for multi-billion-dollar video game franchises like Call of Duty, represents a historic moment in the industry with a transaction value of $69 billion. This move, marking the largest acquisition in video game history, positions Microsoft to become the third-largest gaming company globally by revenue. This paper delves into the industry trend and advises Microsoft on its next moves in the gaming sector - which genres of games to pursue, to publish on which platforms, etc. We then examine Microsoft's alignment with industry potentials and advise on its strategic moves, with a R-based research method detailed in this write-up. We limit our analysis to games published from 2005 to 2023.

In our exploratory analysis, we will examine the dataset by analyzing distributions, correlations, trends, and networks. This will allow us to showcase game attributes and industry trends over the years, providing valuable insights that will inform our predictions.

In the second part of our analysis, we will develop and evaluate prediction models to forecast game popularity and understand the complex effects of various factors. Our methodology will include linear regression and False Discovery Rate, LASSO regression, two-stage LASSO, K-Nearest Neighbors, multinomial logistic regression, topic modeling, decision trees, and random forests. We mainly focus on answering the below questions:

- ➔ What attributes could potentially lead to higher to higher popularity?
- ➔ Major Predictive Models - How do we predict if a game can gain popularity?
- ➔ Classification Models - How to classify if a game could gain above-average popularity?
- ➔ Causal Inference - Does higher rating cause higher popularity?
- ➔ Does Review Impact Rating?
- ➔ What games could lead a game into a potential big IP?

In our analysis, we will consider the below in the Implication for Microsoft Section:

1. **Industry Trends and Opportunities for Microsoft:** Identify key industry trends that Microsoft should be aware of. Analyze whether developing games in specific genres with certain quality standards can lead to higher player engagement and increased sales.
2. **Market Position of Activision Blizzard and Microsoft**: Assess whether Activision Blizzard and Microsoft currently hold a strong position in the market. For example, we will examine if the games created by Activision Blizzard and Microsoft are aligned with prevailing industry trends, and whether the Xbox console (owned by Microsoft) is preferred in the market compared to its competitors.

By exploring these areas, we aim to understand the strategic positioning of Microsoft within the gaming industry and identify potential opportunities for Microsoft to capitalize on.

2. Introduction to Our Dataset

We use the Video Game Sales 2024[1] and Popular Video Games 1980-2023[2] datasets to perform this analysis. We merged the two dataset by game Title, using only games published after 2005.

A. Dataset Variables Descriptions Used

Our merged dataset contains 648 video games dating from 2005 to 2023, with the below variables listed for each game.

Variable Name	**Variable Description**
Title	Title of the Game
Release.Date	Date of release of the game's first version
Rating	Average rating, out of 5
Times.Listed	Number of users who listed this game
Number.of.Reviews	Number of reviews received from the users
genre	Genre of the game; another variable Genres contains all genres of a game, but potential too many
Summary	Summary provided by the team
Reviews	User reviews
Plays	Number of users that have played the game before
Playing	Number of current users who are playing the game
Backlogs	Number of users who have access but haven't started with the game yet
Wishlist	Number of users who wish to play the game
console	Console the game was released for, could be released on multiple consoles
publisher	Publisher of the game
developer	Developer of the game
total_sales	Global sales of copies in millions

[1] Video Game Sales 2024, https://www.kaggle.com/datasets/asaniczka/video-game-sales-2024?resource=download
[2] Popular Video Games 1980 - 2023, https://www.kaggle.com/datasets/arnabchaki/popular-video-games-1980-2023

B. Data Cleaning and Additional Variables

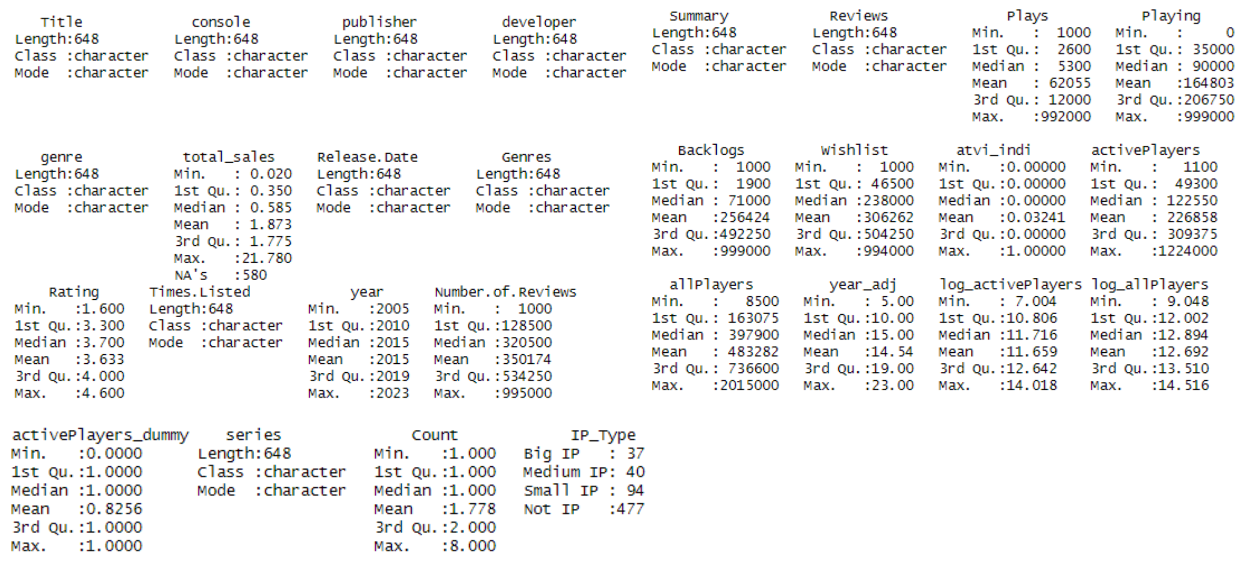
Given that the sales dataset includes multiple entries for each game based on the console it is released on, we consolidated the consoles into a single cell for each game and summed the total sales across all consoles. Additionally, we merged reviews into one cell per game. We filtered out entries with N/A values in the Rating, Summary, Reviews, and Year columns.

New variables created:

- activePlayers: Made up with plays and playing – the number of players who have played this game.
- allPlayers: Made up with all players who have bought the game (activePlayers + Backlogs) but some may not have started the game.
- activePlayers_dummy: Higher than median, 1, else 0.
- log_activePlayers: log(activePlayers+1)
- log_allPlayers: log(allPlayers+1)
- year: Release year
- year_adj: year-1 to first normalize the year effect
- atvi_indicator: 1 if the game was produced by Activision or Blizzard Entertainment
- IP_Type: Big IP, Medium IP, Small IP, and Not IP indicator

Additional sparse matrices created:

- Tdm_sparse: Significant game summary words appearance for each game
- console_sparse/console_df: Console dummy matrix for each game



```
   Title              console            publisher          developer
 Length:648         Length:648         Length:648         Length:648
 Class :character   Class :character   Class :character   Class :character
 Mode  :character   Mode  :character   Mode  :character   Mode  :character



   genre             total_sales      Release.Date          Genres
 Length:648         Min.   : 0.020   Length:648         Length:648
 Class :character   1st Qu.: 0.350   Class :character   Class :character
 Mode  :character   Median : 0.585   Mode  :character   Mode  :character
                    Mean   : 1.873
                    3rd Qu.: 1.775
                    Max.   :21.780
                    NA's   :580
     Rating       Times.Listed            year      Number.of.Reviews
 Min.   :1.600   Length:648         Min.   :2005   Min.   :  1000
 1st Qu.:3.300   Class :character   1st Qu.:2010   1st Qu.:128500
 Median :3.700   Mode  :character   Median :2015   Median :320500
 Mean   :3.633                      Mean   :2015   Mean   :350174
 3rd Qu.:4.000                      3rd Qu.:2019   3rd Qu.:534250
 Max.   :4.600                      Max.   :2023   Max.   :995000
```

```
   Summary            Reviews               Plays            Playing
 Length:648         Length:648         Min.   :  1000   Min.   :     0
 Class :character   Class :character   1st Qu.:  2600   1st Qu.: 35000
 Mode  :character   Mode  :character   Median :  5300   Median : 90000
                                       Mean   : 62055   Mean   :164803
                                       3rd Qu.: 12000   3rd Qu.:206750
                                       Max.   :992000   Max.   :999000

    Backlogs          Wishlist          atvi_indi        activePlayers
 Min.   :  1000   Min.   :  1000   Min.   :0.00000   Min.   :   1100
 1st Qu.:  1900   1st Qu.: 46500   1st Qu.:0.00000   1st Qu.:  49300
 Median : 71000   Median :238000   Median :0.00000   Median : 122550
 Mean   :256424   Mean   :306262   Mean   :0.03241   Mean   : 226858
 3rd Qu.:492250   3rd Qu.:504250   3rd Qu.:0.00000   3rd Qu.: 309375
 Max.   :999000   Max.   :994000   Max.   :1.00000   Max.   :1224000

   allPlayers         year_adj      log_activePlayers log_allPlayers
 Min.   :   8500   Min.   : 5.00   Min.   : 7.004    Min.   : 9.048
 1st Qu.: 163075   1st Qu.:10.00   1st Qu.:10.806    1st Qu.:12.002
 Median : 397900   Median :15.00   Median :11.716    Median :12.894
 Mean   : 483282   Mean   :14.54   Mean   :11.659    Mean   :12.692
 3rd Qu.: 736600   3rd Qu.:19.00   3rd Qu.:12.642    3rd Qu.:13.510
 Max.   :2015000   Max.   :23.00   Max.   :14.018    Max.   :14.516
```

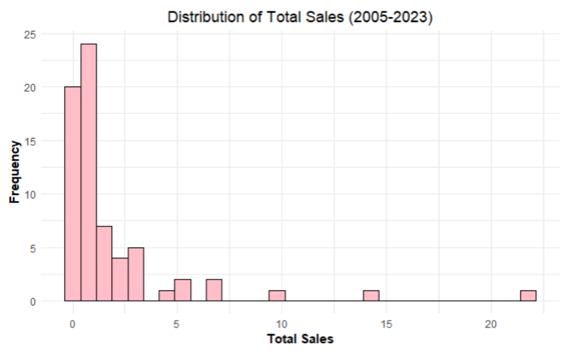
```
 activePlayers_dummy    series                Count               IP_Type
 Min.   :0.0000      Length:648         Min.   :1.000   Big IP   : 37
 1st Qu.:1.0000      Class :character   1st Qu.:1.000   Medium IP: 40
 Median :1.0000      Mode  :character   Median :1.000   Small IP : 94
 Mean   :0.8256                         Mean   :1.778   Not IP   :477
 3rd Qu.:1.0000                         3rd Qu.:2.000
 Max.   :1.0000                         Max.   :8.000
```

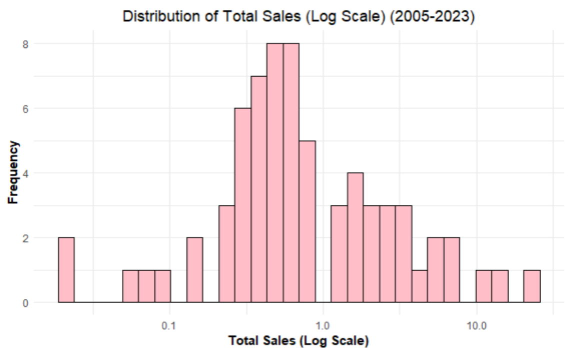
3. Exploratory Data Analysis

A. Exploring Potential Y Variables

a) Total Sales

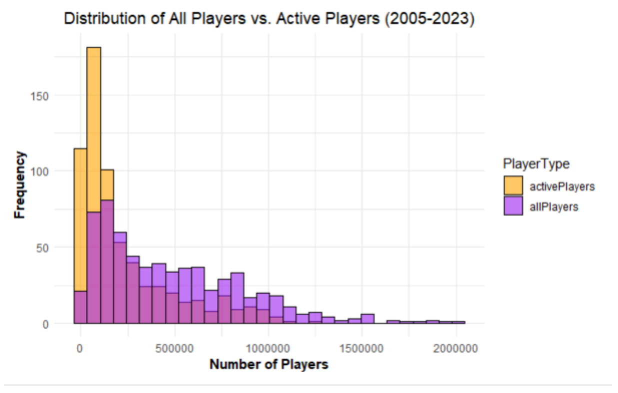
Linear Scale: The histogram of total sales on a linear scale shows a heavily right-skewed distribution. Most games have low sales, with few games achieving very high sales.
Log Scale: The log scale reveals a more normalized distribution, suggesting that while many games still have lower sales, the disparity in sales volumes isn't as extreme on a logarithmic scale.

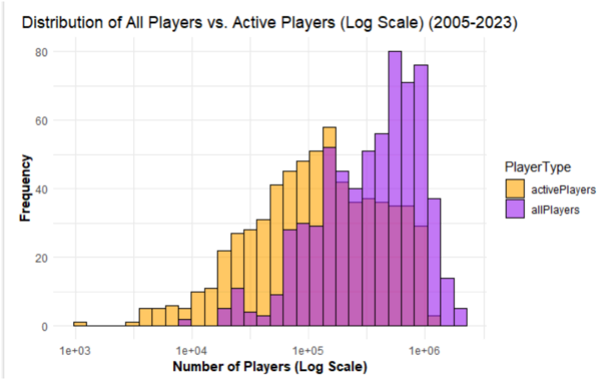




b) All Players vs. Active Player

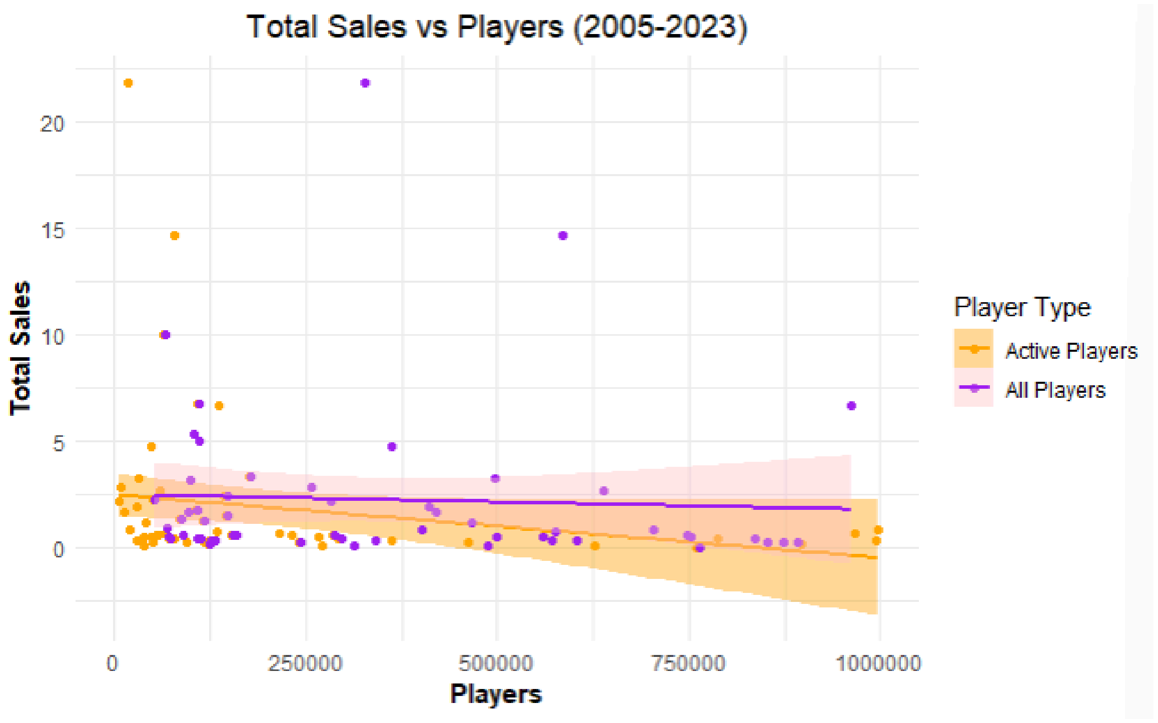
Linear Scale: Similar to total sales, the distribution of both all players and active players is right-skewed. Most games have fewer players, but a few games have very high player counts.
Log Scale: On the log scale, the distribution becomes more bell-shaped, especially for active players, indicating a more typical statistical distribution. This suggests variability in player engagement across games. Compared to total number of players, number of active players may be more suitable





The original skewness in both total sales and player counts indicates the presence of outliers. Such outliers can significantly influence the performance of predictive models, especially linear models, which assume normally distributed errors and equal variance across the range of predictors. The use of log scales and the resulting normalization suggest that transformations (such as logarithmic transformations) may be necessary when using these variables as predictors in regression models to stabilize variance and reduce skewness.

c) Relationship Between Total Sales and Number of (Active) Players

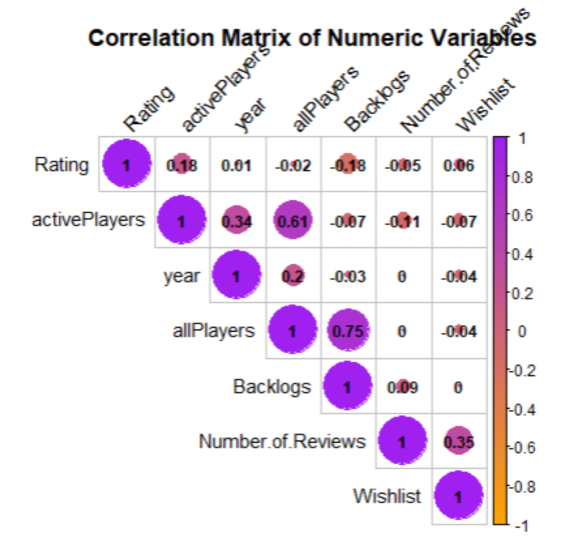


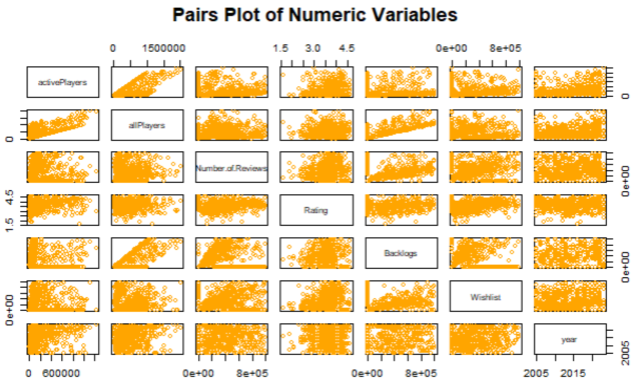
Higher player counts generally suggest greater game popularity and might intuitively be expected to correlate with increased sales. However, it is interesting to observe counterintuitive, a slightly negative relationship between total sales and player numbers in the data. Upon further examination of the data, it becomes evident that numerous NAs for total sales, particularly in online games like 'League of Legends', which enjoy high popularity and significant revenue without entry costs, may distort this expected relationship. Consequently, *total sales may not serve as a reliable predictor for understanding both popularity and game performance* due to these inconsistencies in the data set.

B. Exploring Potential X variables

a) Exploring Numerical Variables

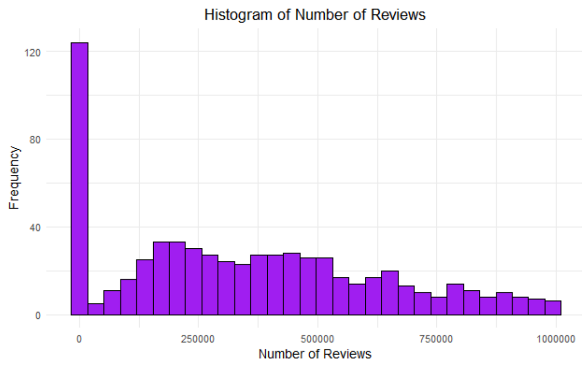
We plot the correlation matrix of numeric variables and pair plots for interpretation. For some numerical variables, we plot the purple distribution histogram in purple on the left and average per publishing year in orange on the right.

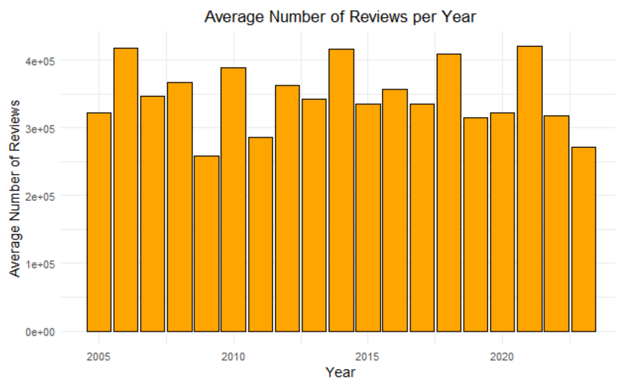




(1) Number of Reviews:

The correlations between activePlayers and Number.of.Reviews (-0.11) and between allPlayers and Number.of.Reviews (0) are slightly negative or negligible. This shows that as the number of active players increases, the number of reviews decreases slightly, but the relationship is weak. This suggests that games with higher active players do not necessarily receive more reviews, indicating that players might be more engaged in playing rather than leaving reviews. For example, "Minecraft" is played extensively but has relatively fewer reviews because players are more engaged in playing and less likely to leave reviews due to its popularity on streaming platforms and social media.

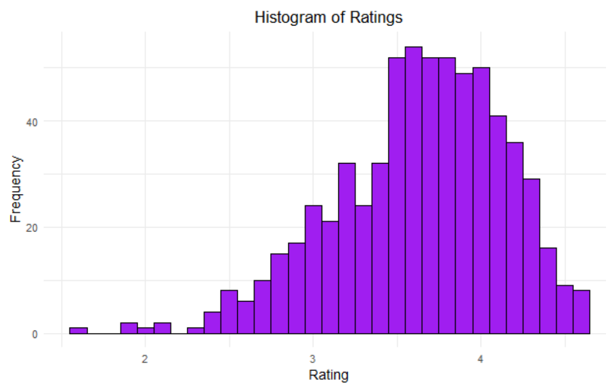


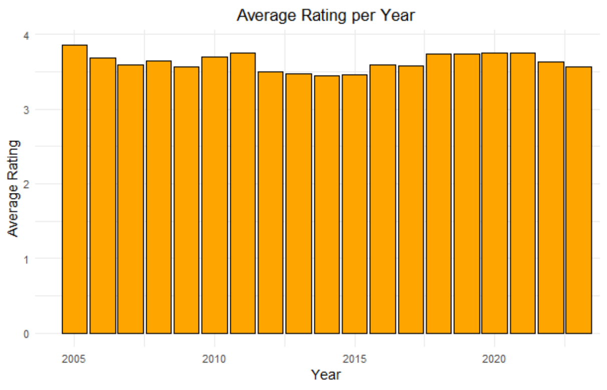


(2) Rating:

Rating has a weak negative correlation with Number.of.Reviews (-0.05) and a weak positive correlation with activePlayers (0.18) and Wishlist (0.06). There is also a very weak negative correlation with allPlayers (-0.02).

Games with more reviews tend to have slightly lower ratings, possibly because players might be more motivated to leave a review when they have a negative experience. "No Man's Sky" initially received many reviews with lower ratings due to unmet expectations, despite its popularity. Popular games like "Fortnite" and "League of Legends" have diverse ratings due to varying player experiences, but their popularity in terms of plays and reviews does not necessarily correlate with higher ratings. These games are too popular on streaming platforms and social media to make its gaming platform review matter. The positive effect of rating on activePlayers may have a marginal diminishing effect, which is worth discovering whether ratings have a causal effect on the number of players.



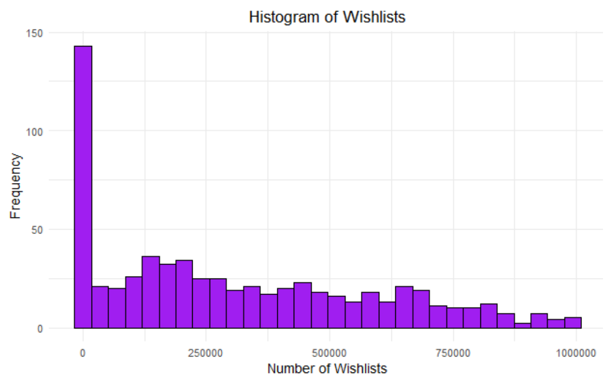


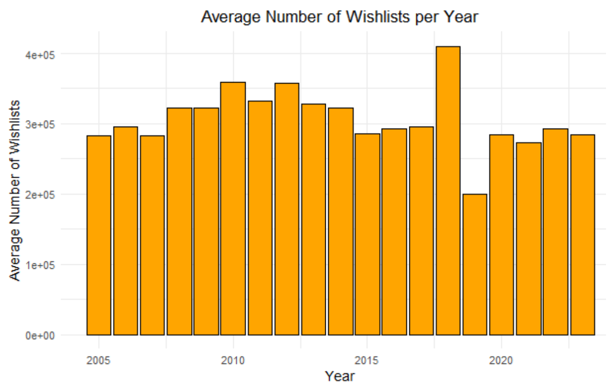
(3) Wishlist

The correlations between activePlayers and Wishlist (-0.07) and between allPlayers and Wishlist (-0.04) are very weak and negative.

This suggests that the number of plays does not strongly influence whether a game is added to wishlists. Players may add games to their wish lists for various reasons unrelated to how often they are played. For example, high-priced games may be wishlisted until they go on sale, while free games may be tried immediately and thus not appear on wishlists. Popular games such as "League of Legends" are not typically placed on wish lists since they are free and can be accessed anytime.

The high correlation between Wishlist and Number.of.Reviews (0.35) highlights that reviews promote people's interest. Good games that are not published by larger publishers can still gain interest from players through positive reviews.





(4) Year

The year variable has a moderate positive correlation with activePlayers (0.34) and allPlayers (0.61). This suggests that newer games tend to have more active players and total plays, indicating a trend of increasing player engagement over time. This trend can be useful for predicting future player engagement based on the release year.

Implications for Forecast and Analysis

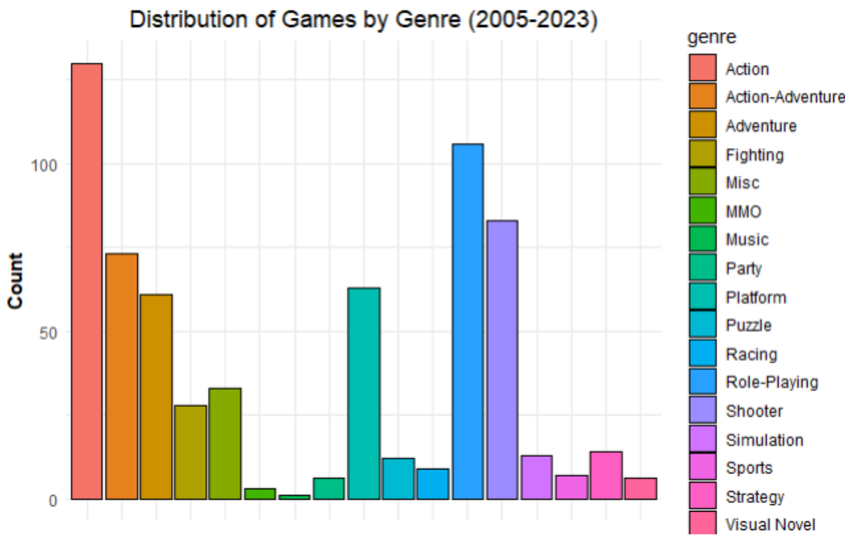
Focus on Popularity: To achieve higher game popularity, Microsoft should prioritize factors that drive the number of players and active engagement. These include effective marketing, in-game events, and social features. However, popularity metrics (like the number of players) often have weak correlations with ratings.
Leveraging Existing IPs: Existing game IPs with strong player bases can guarantee good sales for sequels or related titles. Microsoft should consider developing sequels for these popular franchises to capitalize on their established success.
Focus on Ratings: For sustained revenue and long-term success, game quality and ratings are crucial. Higher-rated games can lead to continuous cash flows and a loyal player base. Addressing player feedback and ensuring high-quality game development are key strategies.
Reviews: For newer or lesser-known titles, encouraging positive reviews can boost interest and wishlist additions.

We will focus on predicting popularity and conduct a causal effect analysis below to see how rating affects popularity.

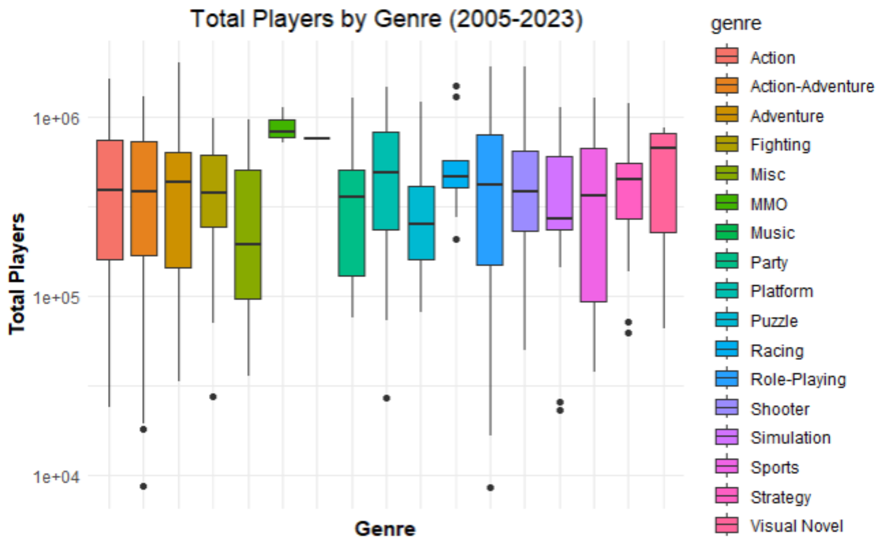
b) Exploring Categorical Variables

(1) Genre

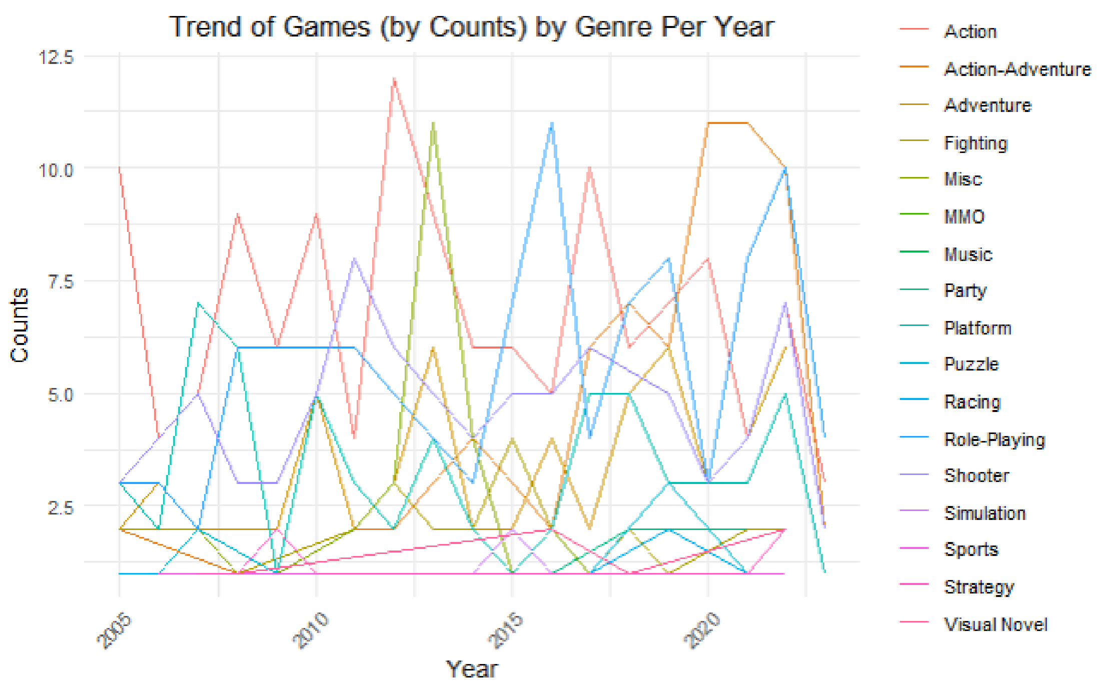
There are in total 17 genres within our dataset, with Action being the most common (130), followed by Role-Playing (106) and Shooter (83). The least common genres are Music (1) and Massively multiplayer online games (MMO) (3).

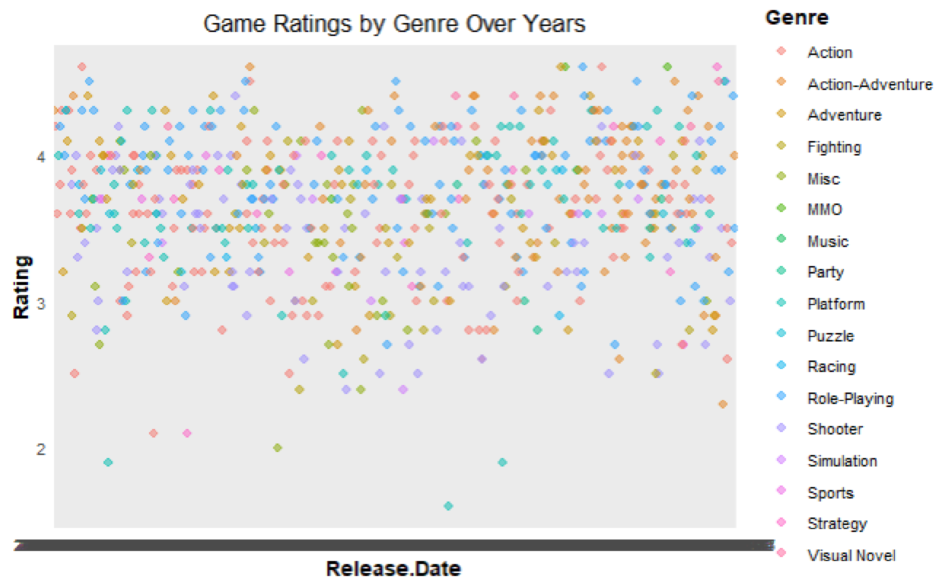


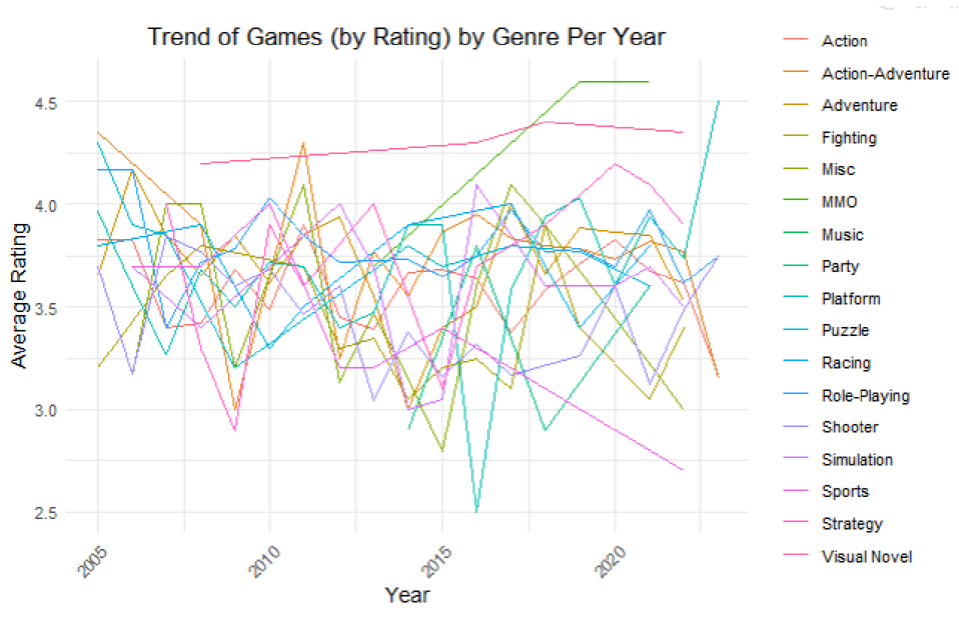
We also explored the distribution of different attributes by genre. For the number of total players, MMO games have the highest mean of total players and comparatively less variation, likely because the three games in this genre are strong enough to capture most of the players. In contrast, genres like Visual Novel, despite having a small count (6), exhibit greater variation. Furthermore, games in major genres like Action, Adventure, and Role-Playing all have similar means and distributions for the total number of players.



In a time span, we plotted the trend of the number of games by genre. Action-Adventure and Role-Playing genres show the most significant increase in recent years, which might be promising directions for developers to pursue. However, game ratings by genre are more scattered over the years, indicating a diverse range of player preferences. Additionally, we reviewed the trend of average ratings, and genres such as MMO, Strategy, and Platform tend to yield higher ratings.

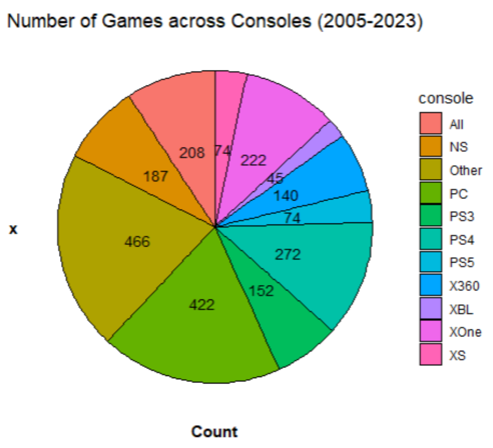


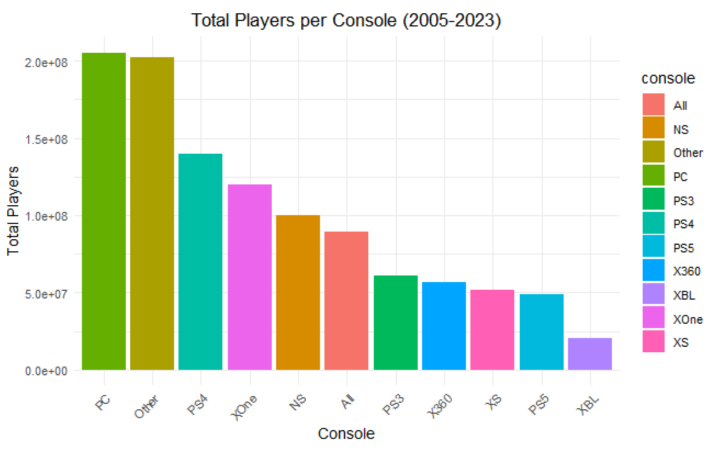


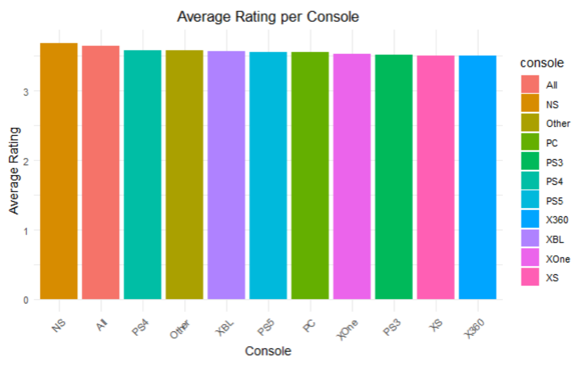


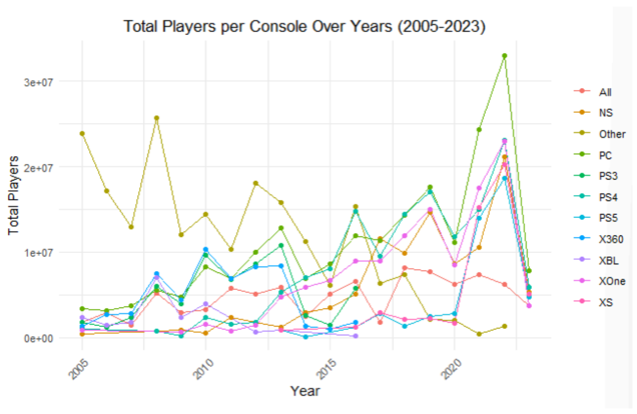
(2) Consoles

The console variable includes 11 categories, encompassing "All," "Other," and nine major gaming consoles. Excluding "Other," the largest category is PC with 466 entries, while the smallest is XBL with 45 entries. The PS series also constitutes a significant portion. The average rating per console does not vary much, suggesting that consoles do not significantly influence the perceived quality of the game. However, there exists a variance in the total number of players. Platforms like PC and XOne have seen an increase in recent years, while Nintendo Switch (NS) remains stable.





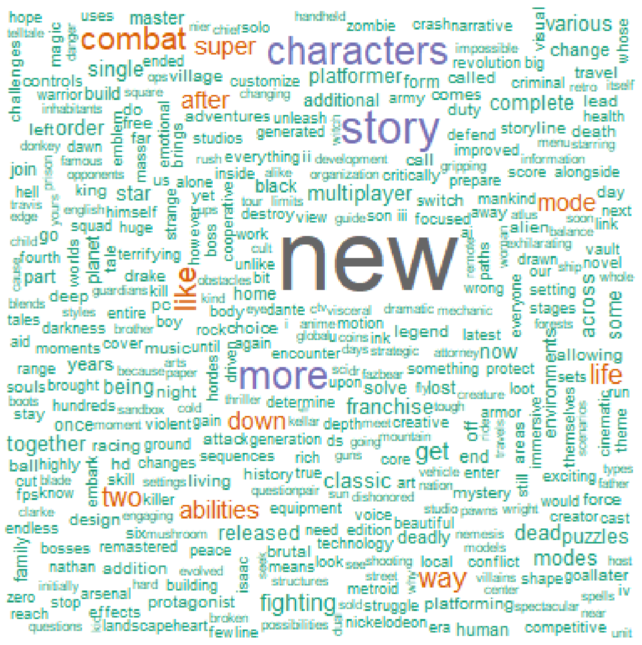




(3) Summary

We have performed text processing on summary for all games, filtering out filler and unnecessary summary words, leaving with words that are relatively meaningful to the specific game and could imply games' content to affect the number of active players. Below shows a word cloud of the summary words. Words like "new", "story", "combat", "characters", "multiplayer", "fighting", "franchise" appear the most.

Figure: Word Cloud of Game Summary Words



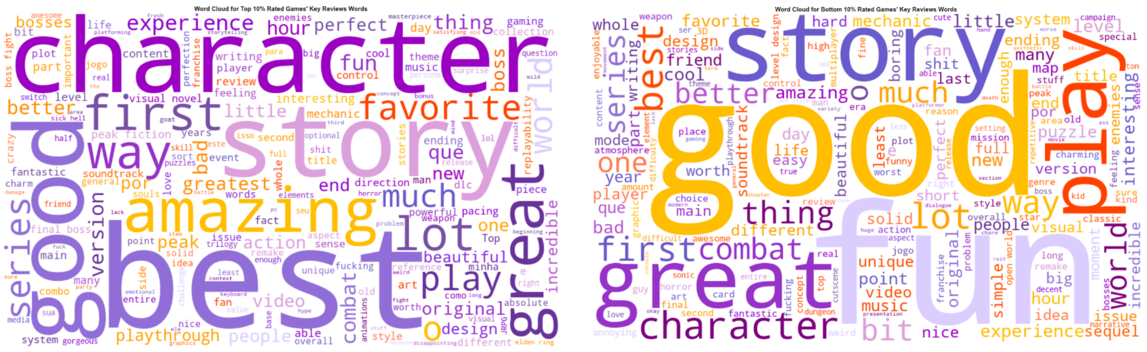
We have also used the *Topics* package to group the summary words that appear together most frequently into five topics, each representing a distinctive theme, providing an insight into the prevalent themes and popular phrases associated with these games. We observed the top 10 bigrams (phrases) of each topic to interpret the themes as: 1. Heroic; 2. Action-packed; 3. Big name; 4. Revolutionary; 5. Acclaimed.

The topics decreased significantly in frequency usage; however, they can serve as a reference for developing new games, providing a sense of what elements resonate with players and allowing the incorporation of similar features into future marketing strategies.

```
Top 10 phrases by topic-over-null term lift (and usage %):

[1] 'hero_initially', 'initially_referred', 'referred_guitar', 'hero_music', 'music_rhythm', 'fifth_main',
'entry_guitar', 'hero_series', 'developed_neversoft', 'neversoft_published' (60.5)
[2] 'addictive_action', 'action_frantic', 'combat_massive', 'massive_arsenal', 'weaponry_rpg',
'elements_four', 'op_borderlands', 'borderlands_breakthrough', 'breakthrough_experience',
'experience_challenges' (13.4)
[3] 'started_sonic', 'sonic_origins', 'origins_brand', 'new_collection', 'collection_including',
'including_sonic', 'cd_sonic', 'knuckles_play', 'sonic_tails', 'tails_knuckles' (11)
[4] 'london_industrial', 'industrial_revolution', 'revolution_unleashes', 'unleashes_incredible',
'incredible_age', 'age_invention', 'invention_transforming', 'transforming_lives', 'lives_millions',
'millions_technologies' (7.5)
[5] 'acclaimed_director', 'kojima_metal', 'solid_hd', 'offers_handful', 'popular_metal', 'titles_past',
'past_true', 'true_hd', 'hd_first', 'ever_featuring' (7.5)
```

(4) Reviews

Figure: Highly Rated vs. Lowly Rated Games



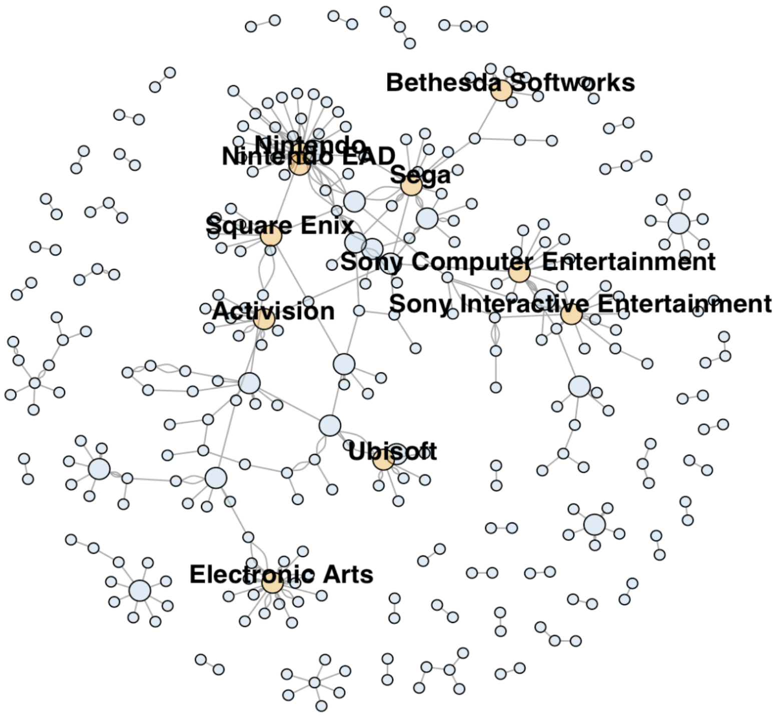
Following the same process for text processing, we created word clouds for the reviews of highly rated versus lowly rated games. As shown in the figure above, highly rated games are characterized by words like "best," "story," and "character," while lowly rated games feature words like "good" and "fun." We did not identify a major distinction between the two categories, likely because the game reviews in our dataset tend to be overall general in nature. This suggests that while certain terms may be more prevalent in reviews of highly rated games, the overall sentiment and vocabulary used in reviews do not differ significantly between highly and lowly rated games.

(5) Publishers and Developers Network

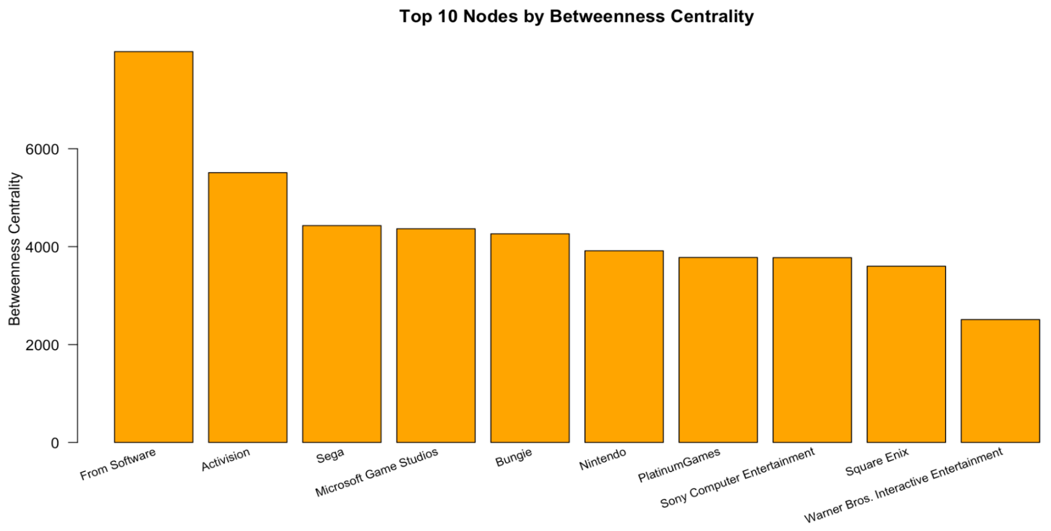
To understand the structure and dynamics of the game industry, We have used the *igraph package* to plot the relationships between publishers and developers for games to explore the

connections between companies, their collaborations, and the importance of each company within the network. We also identified the top 10 companies with the highest degrees and labeled them.
As shown in the graph, the top companies include Nintendo, Electronic Arts, Sega, Activision, Ubisoft, Sony, Square Enix, and Bethesda Softworks, among others. These companies are all closely connected with several other game studios and are recognized as major game developers.

However, these major studios do not exhibit direct close connections with each other. For example, Activision is only related to Nintendo through Square Enix, and connected to Electronic Arts through other two big studios.



We also summarized the betweenness of these edge lists and found that Activision ranks second in betweenness within our network analysis, just after From Software, suggesting its highly important status in the industry. Interestingly, Microsoft Game Studios ranks fourth in betweenness, underscoring its significant role in connecting different parts of the network as well.



(6) IP

We define the concept "IP" by finding a series consisting of at least 2 games with the same name of theories and builders. Besides, there are different sizes of series filtered by numbers of games in the series, including Big IP, Medium IP, Small IP.

IP Types	Numbers of Series (Threshold)	Amount	Avg of ActivePlayers
Big IP	>= 6	37	200616.2
Medium IP	>=4	40	169802.5
Small IP	>= 2	94	169210.6
Not IP	rest	477	245038.6

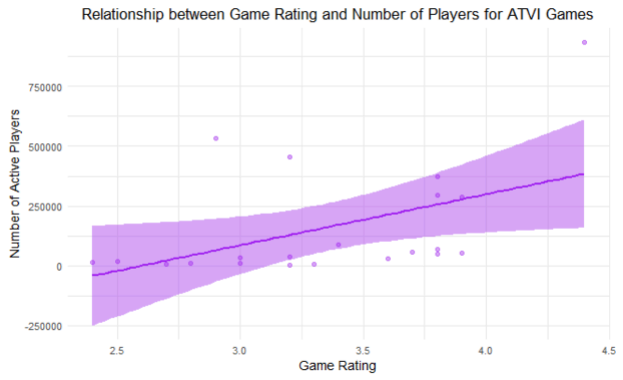
Implications for Forecast and Analysis

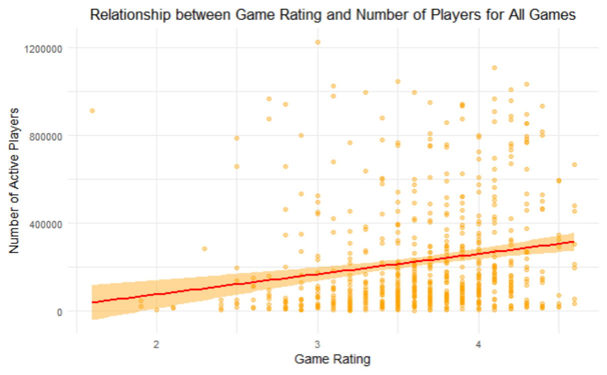
The high dimensionality in our data is primarily due to categorical variables. Therefore, it is crucial to identify the specific genres, consoles, publishers, summaries, and review patterns that lead to higher game popularity.

That is to say, solely based on categorical data, such as genre and publisher, without relying on numerical variables like the number of wishlists and backlogs, or whether we can predict game popularity. In addition, understanding these trends will also help us assess if Microsoft is well-positioned to produce popular games.

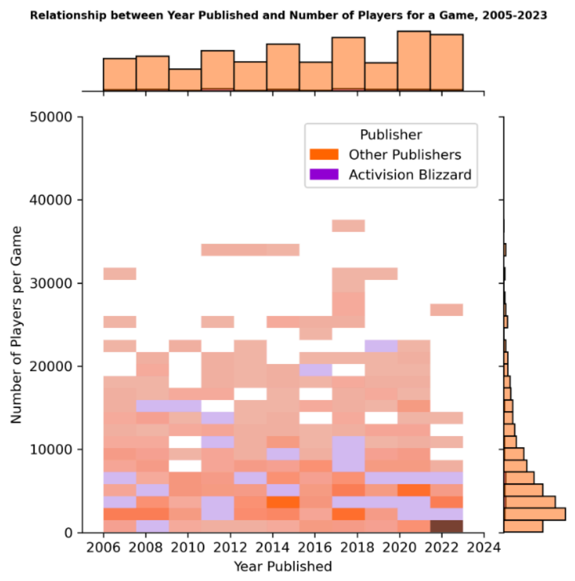
By focusing on these aspects, we can gain valuable insights into the key factors that drive game popularity and make informed recommendations for Microsoft's game development strategy.

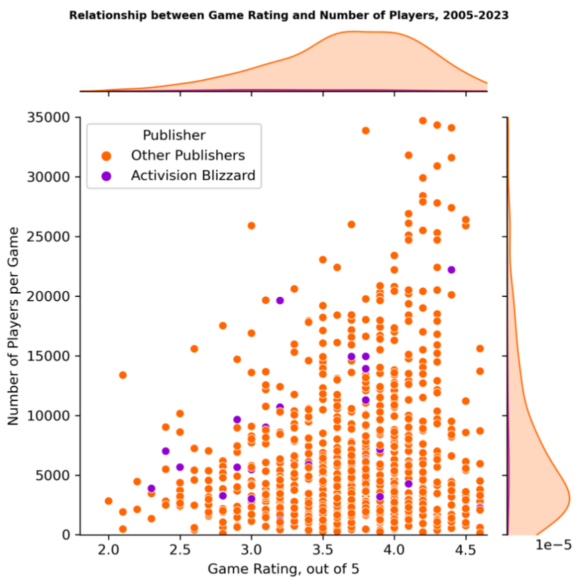
C. Exploring Activision Blizzard Games Compared to All Games





Connecting to Activision Blizzard: Before data cleaning to exclude games lacking summary, rating, or reviews, the analysis reveals distinct trends between all games and those published by Activision Blizzard (game rating as x-axis, number of total players as y-axis). The first plot, representing all games, displays a modest upward trend, indicating a slight increase in the number of players with higher game ratings. In contrast, the second plot focused on Activision Blizzard games shows a more pronounced positive correlation. Notably, Activision Blizzard titles maintain a higher average number of players across all ratings and demonstrate a steeper increase in player numbers with rising ratings. This suggests that Activision Blizzard's games attract more players regardless of game rating, although they also include games with generally lower ratings compared to all games.





4. Prediction

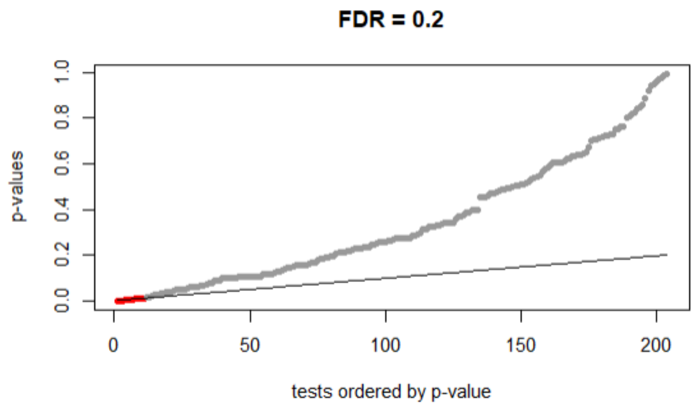
A. Exploratory Variable Selection- What attributes could potentially lead to higher to higher popularity?

(1) Linear Regression and FDR Using Existing Variables

Running an OLS regression of log_activePlayers on year, number of reviews, publisher, genre, rating, and console, R2 is quite high, 51.13%, with adjusted R2 being only 0.2879 due to the high dimensions we have.

Then, using this OLS, 10 variables are significant at 20% FDR out of 194, while only 2 are significant at 10% FDR–year and rating are significant. With 20% FDR, several publishers, Misc and Platform genres, and the console PS4 being significant with p values lower than 0.0093.

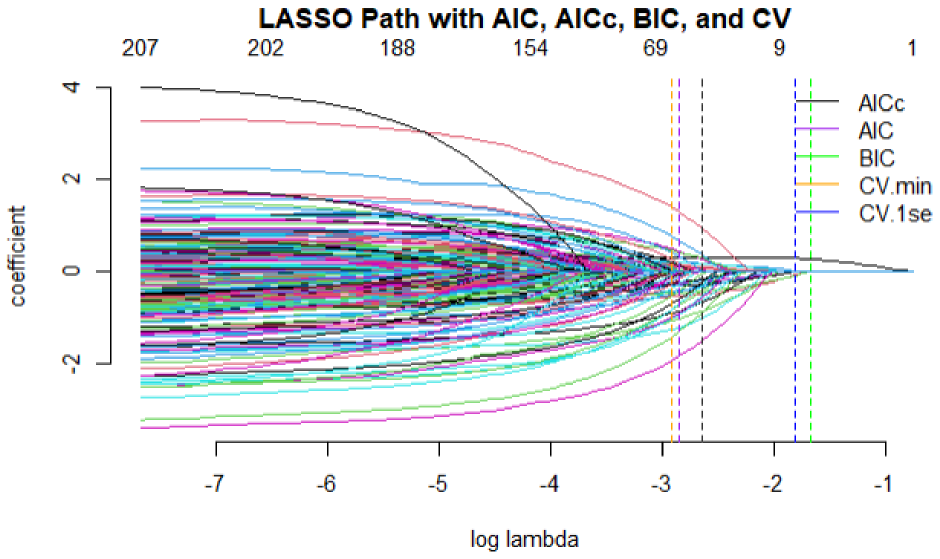
FRD helps control false discoveries, but the fact that it selects so few variables may suggest high noise in the data to mask true effects as well as overfitting with high dimensions of data. It also tells us the significance of year and rating only by looking at p values, without us understanding the direction of the coefficients.We would like to conduct additional predictive analysis instead.

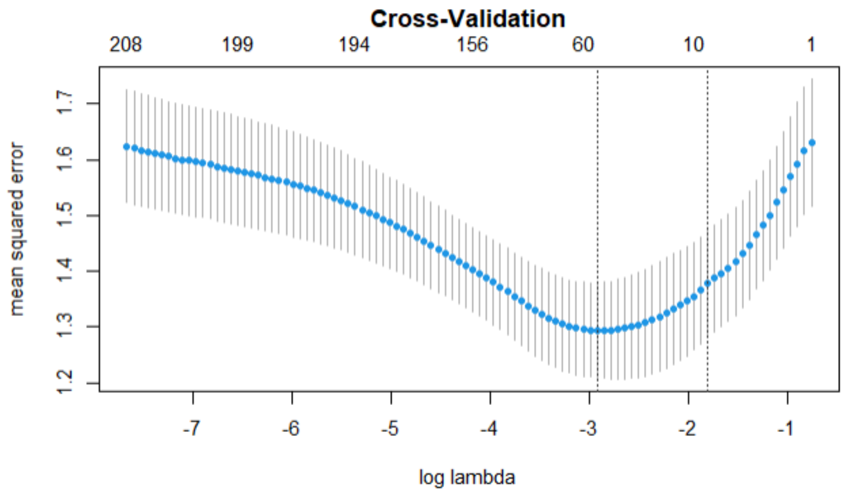


```
 [1] "(Intercept)"
 [2] "year_adj"
 [3] "publisherCD Projekt Red Studio"
 [4] "publisherEA Sports"
 [5] "publisherMatt Makes Games Inc."
 [6] "publisherMedia.Vision"
 [7] "publisherTeam Cherry"
 [8] "genreMisc"
 [9] "genrePlatform"
[10] "Rating"
[11] "PS4"
```

(2) Using LASSO to predict log_activePlayers and Compare Testing Metrics Using Existing Variables

We fit a LASSO model to predict log_activePlayers, using the variables: publishing year, number of reviews, publishers, genres, rating, and consoles. We observed model selection criteria including AIC, AICc, BIC, CV.min, and CV.1se. Notably, CV.min, which minimizes cross-validation error, selects slightly more variables (48) than AIC and AICc (35), which tend to overfit. BIC and CV.1se (9) select the fewest variables. In terms of the number of variables selected, the order is: CV.min > AIC = AICc > BIC > CV.1se. The CV graph shows that MSE decreases as variables are shrunk, then starts increasing after log(lambda) = -2.92. The model using minimum CV explains 29.64% variability in log active players.



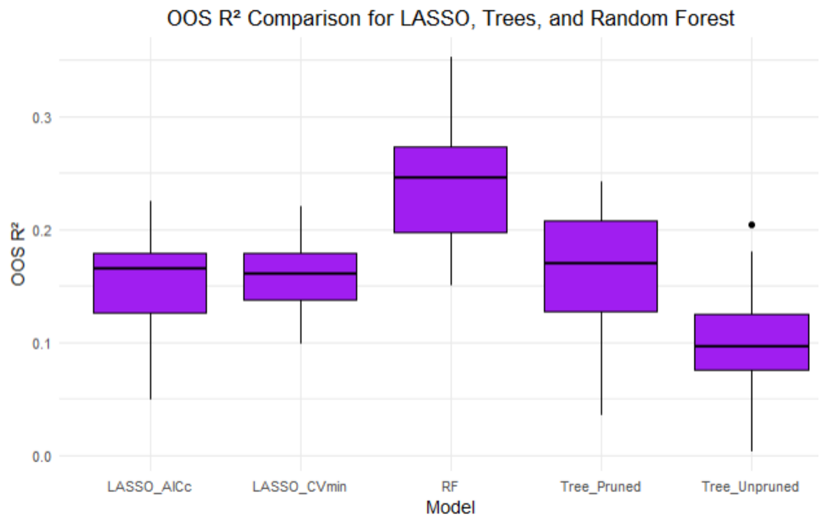


B. Major Predictive Models - How do we predict if a game can gain popularity?

After the initial exploratory analysis, we are determined to use attributes that Microsoft can decide on a game now to predict its popularity, including summary words (game content), genre, year, publisher, rating (imply the quality of a game, whether Microsoft has to make a quality game), IP, and console. We call these attributes the **pre-game-release variables**. Variables such as number of reviews and reviews are removed because they are not available to assess before publishing.

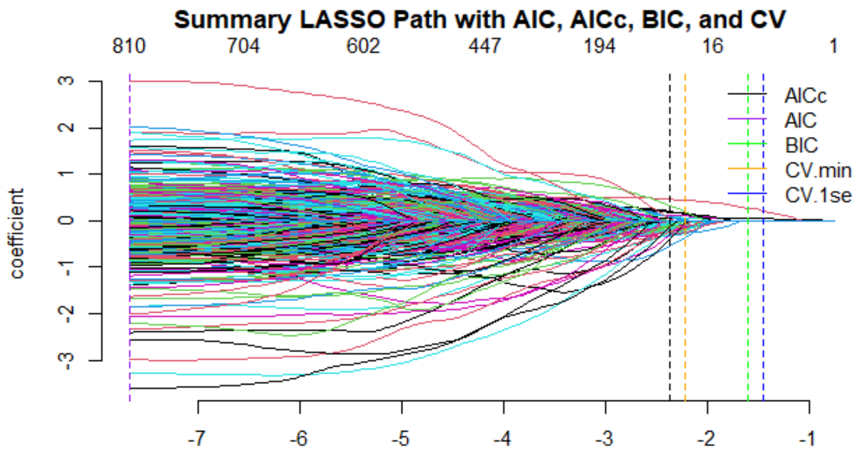
We have performed LASSO (selected by AICc and minimum CV), decision tree (unpruned and pruned trees) and random forest using pre-game-release variables. By splitting the sample data to 70%/30% training and testing set 20 times, we calculate the out-of-samples R2 for all models. Comparing out-of-sample R2, it looks like Random Forest > Pruned Tree > LASSO_ AICc > LASSO_CVmin > Unpruned Tree.

Figure: OOS R2 Comparison for LASSO, Trees, and Random Forest



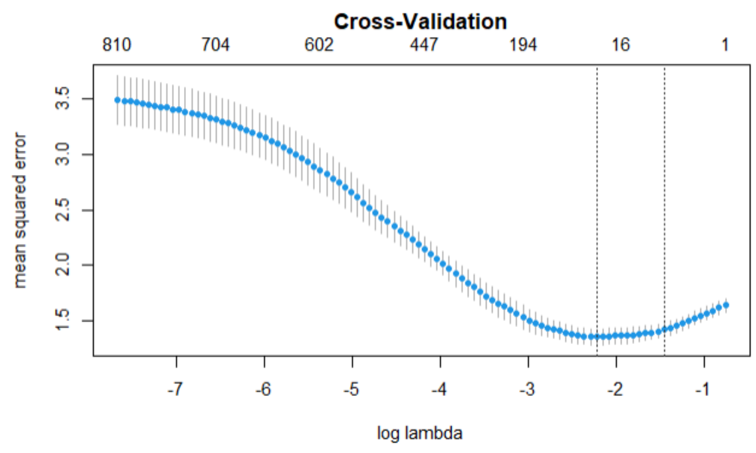
(1) Dealing with Summary words – to Predict Popularity with LASSO

Recall the processed and filtered summary words we have obtained, we now perform a LASSO regression on the filtered 1165 summary words, plus year, genre, publisher, rating, IP, and console. As the below path plot and summary shows, AICc selected 34 variables, including 20 summary words. CV.min selects 28 while CV.1se only selects 3 variables.



Positive effects were observed from words like "shoot," "truth," and "score," while negative effects were associated with "guitar," "ball," "improvements," "Tokyo," "America," and "latest."

	Coefficient	Feature
	<dbl>	<chr>
2	0.456292019	Rating
3	0.241308056	shoot
4	0.175455086	XS
5	0.172614976	environments
6	0.101405804	PS4
7	0.083371100	genreRole.Playing
8	0.067313837	truth
9	0.057737839	year_adj
10	0.025906210	score
11	-0.002395502	published
12	-0.009020608	guitar
13	-0.010229955	ball
14	-0.021969282	publisherSony.Online.Entertainment
15	-0.038009896	Wii
16	-0.046638755	maps
17	-0.055375429	improvements
18	-0.075472586	coins
19	-0.084833600	tokyo
20	-0.126072165	WiiU
21	-0.134310511	controllers
22	-0.138817379	latest
23	-0.146721625	outbreak
24	-0.173482256	help
25	-0.178364057	genreMisc
26	-0.178555974	competitive
27	-0.210962944	publisherMedia.Vision
28	-0.216729796	genreFighting
29	-0.234404047	america
30	-0.239404729	points
31	-0.246418670	publisherJackbox.Games
32	-0.289053379	selling
33	-0.326600941	sold
34	-0.469233819	publisherEA.Sports



Implication for Microsoft

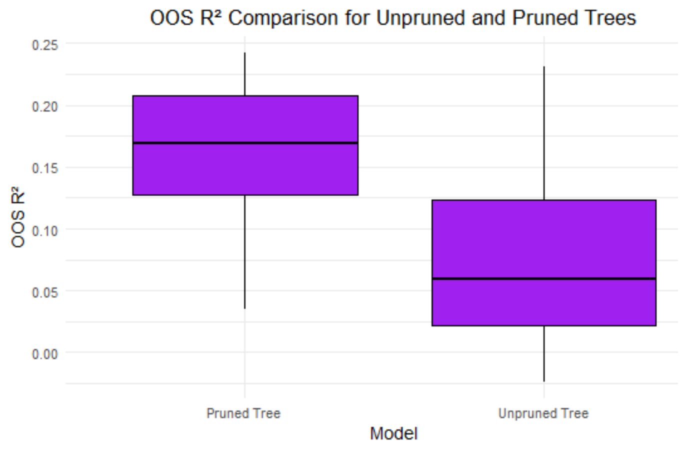
These findings suggest that the summary content is important for players in determining a game's worth or in providing helpful information about game preferences. Notably, words like "America" consistently had negative associations. Upon reviewing the dataset, these words often appeared in games involving "the American colonies" or "world wars," implying that players may not favor such historical and combat-focused games. Words like “improvements” and “latest” may suggest the game just went through improvements, which could imply they did not gain popularity in the past.

Furthermore, the LASSO model highlighted the significant effects of rating, year, and consoles like Xbox (XS) and PS4 on the number of active players. Genres like role-playing also showed a significant impact. This is positive news for Microsoft, as Xbox (owned by Microsoft) is performing well. Interestingly, Activision Blizzard appears neutral in this analysis, suggesting that it is not negatively impacted by the less favored genres. Overall, the model provides valuable insights and recommendations for the gaming industry.

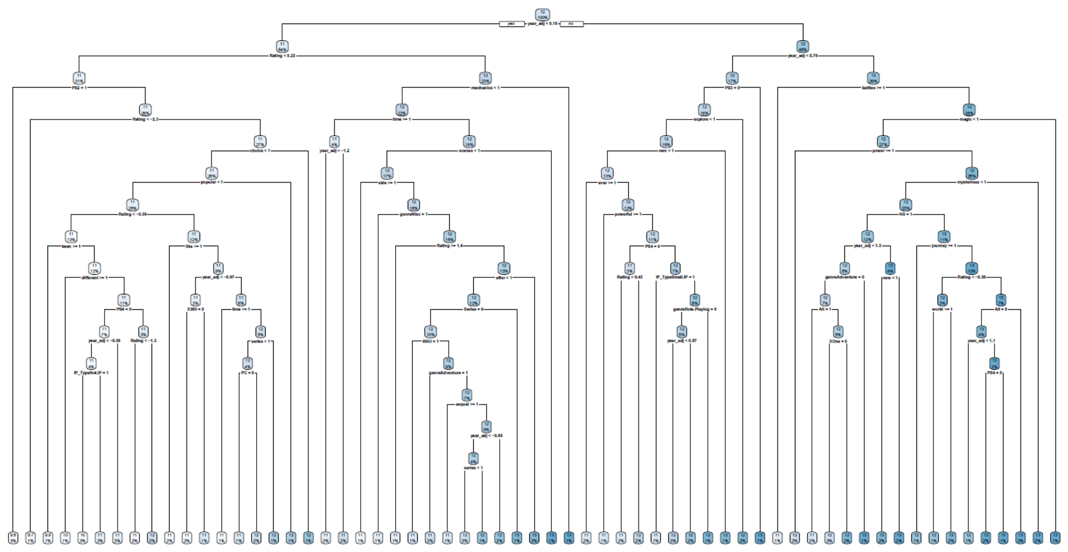
(2) Decision Trees - Unpruned Tree and Pruned Tree

Using the same variables, we fit both an unpruned and a pruned decision tree. Summary words tend to cause overfitting, so we excluded them. The OOS R² from the unpruned tree underperforms compared to LASSO, while the pruned tree outperforms LASSO. Unpruned trees are more complex with many splits and tend to overfit. In contrast, the pruned tree balances complexity and generalizability more effectively, resulting in better performance on new data.

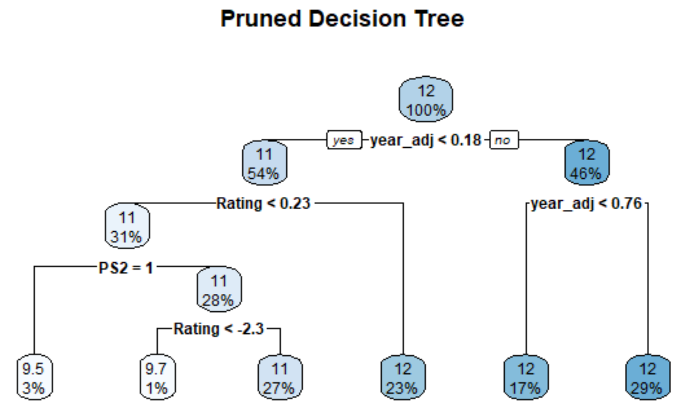
Comparing pruned (left) and unpruned (right) trees, the prior clearing outperforms on the testing set with OOS R2



The Unpruned Tree is very complicated



The Pruned Tree selects fewer variables



```
Variables actually used in tree construction:
 [1] All              genreAction        genreAdventure
 [4] genrePlatform    genreRole.Playing  genreShooter
 [7] IP_TypeNot.IP    IP_TypeSmall.IP    NS
[10] OSX              PC                 PS2
[13] PS3              PS4                PS5
[16] publisherNintendo Rating            XOne
[19] XS               year_adj

Root node error: 1062/648 = 1.6389

n= 648
```

(3) Random Forest

We used the same variables from the decision trees to fit the random forest model, which outperformed all other models.

Interestingly, when including summary words as variables, the percentage of variance explained decreased, the mean squared residuals increased, and the number of splits reduced significantly from 460 to 7. The variable importance plot shows that 'year' and 'Rating' remain the most influential variables, followed by Xbox consoles (XS, XOne). Certain genres, like Role-Playing, require more attention. Additionally, it is noteworthy that Activision as a publisher has a significant impact.

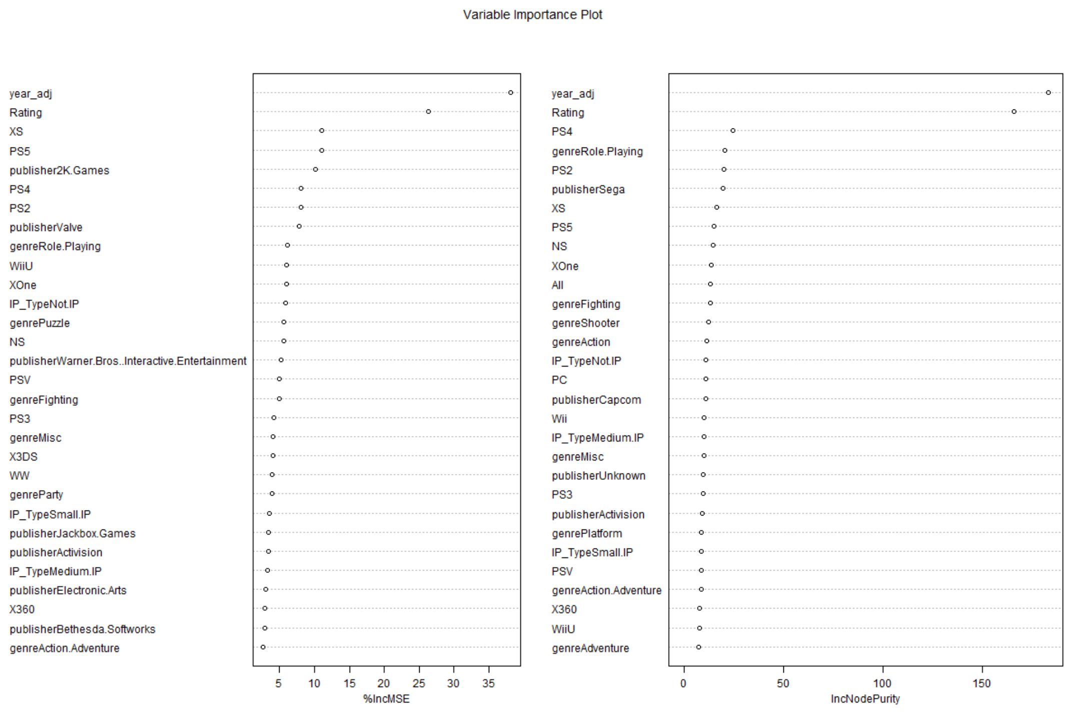
With summary words:

```
Call:
 randomForest(formula = game$log_activePlayers ~ ., data = data.frame(x_tree),      ntree =
500, importance = TRUE)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 460

          Mean of squared residuals: 1.263229
                    % Var explained: 22.92
```

Without summary words:

```
Call:
 randomForest(formula = game$log_activePlayers ~ ., data =
data.frame(x_tree),      ntree = 500, importance = TRUE)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 71

          Mean of squared residuals: 1.257562
                    % Var explained: 23.27
```

Figure: Variable Importance Plot

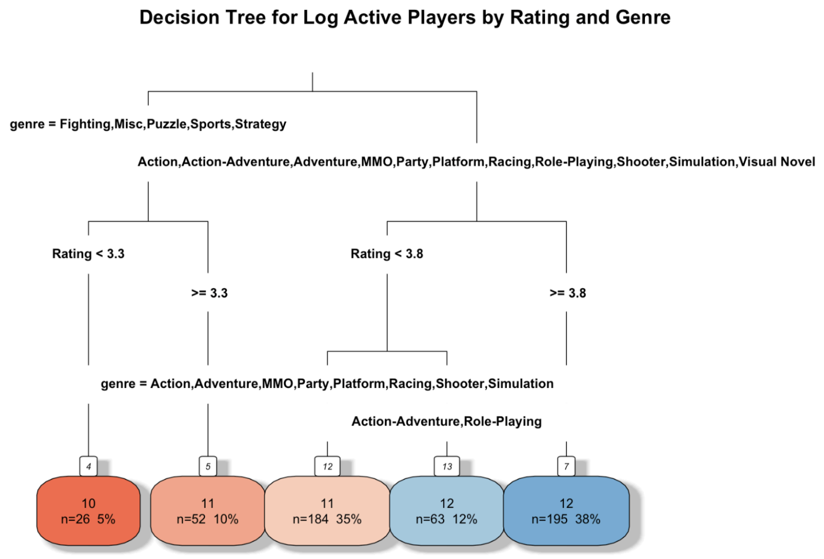


(4) Focused Analysis and Modeling Using Trees and Random Forest

After comparing the models above using all of the pre-game-release variables, we attempt to conduct focused modeling and detailed analysis that focus on genres and rating.

a. Decision Tree

This decision tree analysis focuses on predicting the log-transformed number of active players based on the game rating and genre.


Decision Tree for Log Active Players by Rating and Genre
genre = Fighting,Misc,Puzzle,Sports,Strategy
Action,Action-Adventure,Adventure,MMO,Party,Platform,Racing,Role-Playing,Shooter,Simulation,Visual Novel
Rating < 3.3
>= 3.3
Rating < 3.8
>= 3.8
genre = Action,Adventure,MMO,Party,Platform,Racing,Shooter,Simulation
Action-Adventure,Role-Playing
4
10
n=26 5%
5
11
n=52 10%
12
11
n=184 35%
13
12
n=63 12%
7
12
n=195 38%

The root of the tree firstly splits the data based on the genre. Games in genres like Fighting, Misc, Puzzle, Sports, and Strategy are grouped together on the left, while genres such as Action, Action-Adventure, Adventure, MMO, Party, Platform, Racing, Role-Playing, Shooter, Simulation, and Visual Novel are grouped on the right. This initial split indicates the significant influence of genre on the log-transformed number of active players.

For games in the left branch (Fighting, Misc, Puzzle, Sports, Strategy), the next split occurs at a rating of 3.3. If the rating is less than 3.3, the log-transformed active players tend to be lower (node with n=26). If the rating is greater than or equal to 3.3, the tree further splits based on genre, grouping Action, Adventure, MMO, Party, Platform, Racing, Shooter, and Simulation together. This demonstrates that rating is a critical factor in distinguishing game popularity within these genres.

For games in the right branch (Action, Action-Adventure, Adventure, MMO, Party, Platform, Racing, Role-Playing, Shooter, Simulation, Visual Novel), the initial split occurs at a rating of 3.8. If the rating is less than 3.8, the tree does not further split, indicating this subset has a relatively stable number of log-transformed active players (node with n=63). If the rating is greater than or equal to 3.8, the tree further splits based on genre, differentiating Action-Adventure and Role-Playing from the other genres. This highlights the importance of rating thresholds in predicting game popularity within these genres.

The terminal nodes (leaves) show the log-transformed number of active players for different combinations of genre and rating. Nodes with mixed genres and ratings illustrate how multiple factors interact to influence player engagement.

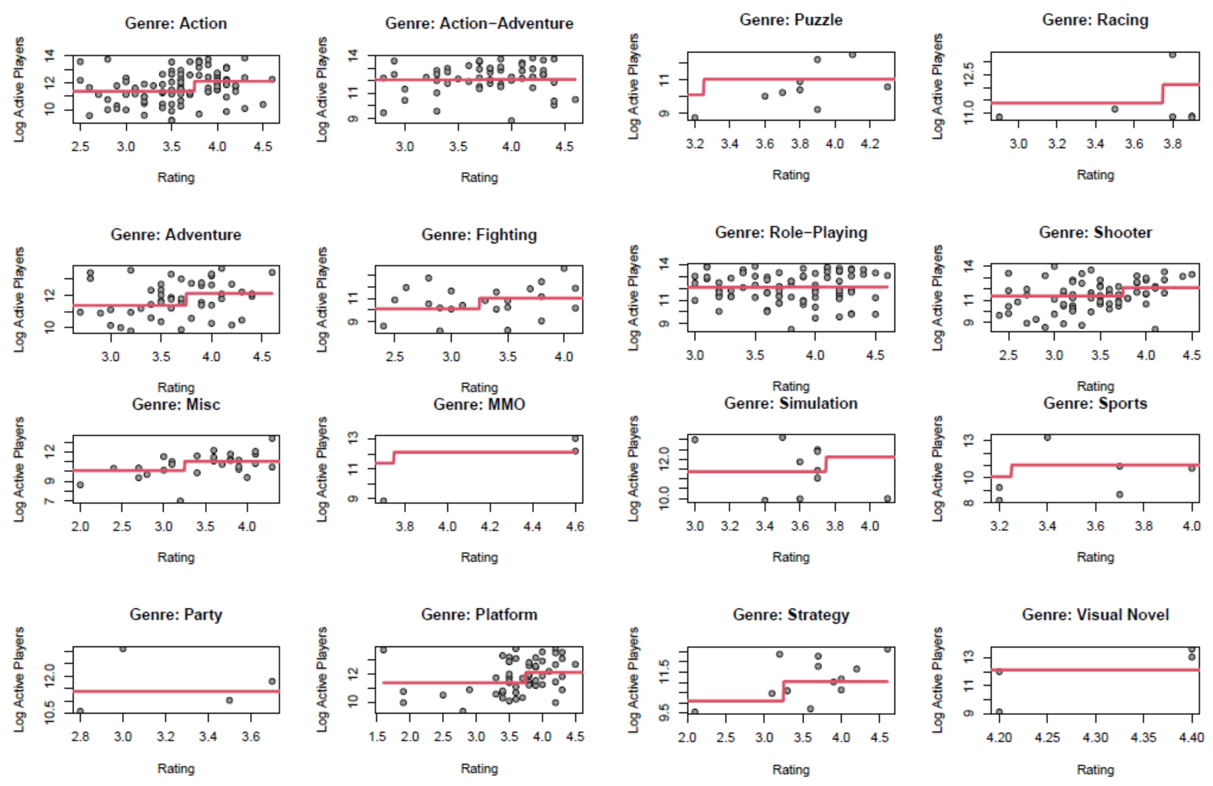
Implication for Microsoft

Microsoft should target high-rating especially in genres like Action, Adventure, MMO, Party, Platform, Racing, Shooter, and Simulation. For even higher engagement, strive for ratings above 3.8, particularly in genres like Action-Adventure and Role-Playing. Even for genres like Fighting, Misc, Puzzle, Sports, and Strategy, ensuring the game rating is above 3.3 can significantly increase active players. By concentrating on these areas, game companies can effectively increase active players, driving higher engagement and revenue.

b. Random Forest

The random forest analysis illustrates how game ratings influence the log-transformed number of active players across various genres.

Figure: Model Averaging with Random Forest for each Game Genre



1. Action: The model shows a relatively stable increase in log active players with higher ratings. The predictions are close to the actual data points, suggesting a balanced fit without overfitting or underfitting.

2. Action-Adventure: There is a consistent positive relationship between ratings and log active players. The model captures the trend accurately, indicating a well-balanced fit with no significant overfitting.

3. Adventure: The increase in log active players with higher ratings is evident. The model's predictions closely follow the actual data, showing a balanced fit without signs of overfitting.

4. Fighting: The log active players remain relatively stable across the rating range. The model's fit appears less tight, indicating a potential slight underfitting, as it doesn't capture all the variations in the data.

5. Misc: A gradual increase in log active players with higher ratings is observed. The model appears to generalize well, showing no signs of overfitting or underfitting, suggesting a balanced fit.

6. MMO: The model shows a strong positive relationship between ratings and log active players for higher ratings. The predictions are consistent with the actual data, indicating no overfitting

and a balanced fit.

7. Party: The stepwise increase in log active players with higher ratings is well-captured by the model. The fit appears balanced, with the model accurately reflecting the data without overfitting.

8. Platform: The model demonstrates a steady increase in log active players with higher ratings. The fit seems balanced, with no significant signs of overfitting or underfitting.

9. Puzzle: The model captures the consistent upward trend in log active players with higher ratings well. The predictions align with the actual data, indicating a balanced fit without overfitting.

10. Racing: The gradual increase in log active players with higher ratings is well-represented by the model. The fit appears balanced, with predictions closely following the data trend.

11. Role-Playing: The stable trend in log active players across various ratings is accurately captured by the model. The fit is balanced, showing no signs of overfitting or underfitting.

12. Shooter: The steady increase in log active players with higher ratings is well-predicted by the model. The fit appears balanced, with no overfitting.

13. Simulation: The decline in log active players across various ratings is reflected in the model. The fit seems balanced, with the model accurately capturing the downward trend without overfitting.

14. Sports: The stable trend in log active players with higher ratings is well-captured. The fit appears balanced, with no overfitting or underfitting.

15. Strategy: The positive trend in log active players with higher ratings is accurately represented. The fit is balanced, showing no signs of overfitting.

16. Visual Novel: The model captures the strong positive correlation between high ratings and log active players. The fit appears balanced, with no significant signs of overfitting.

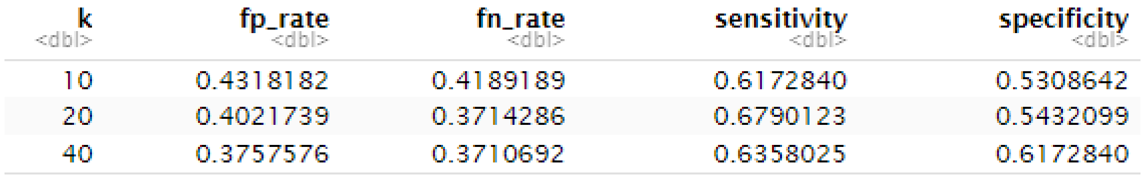
C. Classification Models - How to classify if a game could gain above-average popularity?

(1) KNN

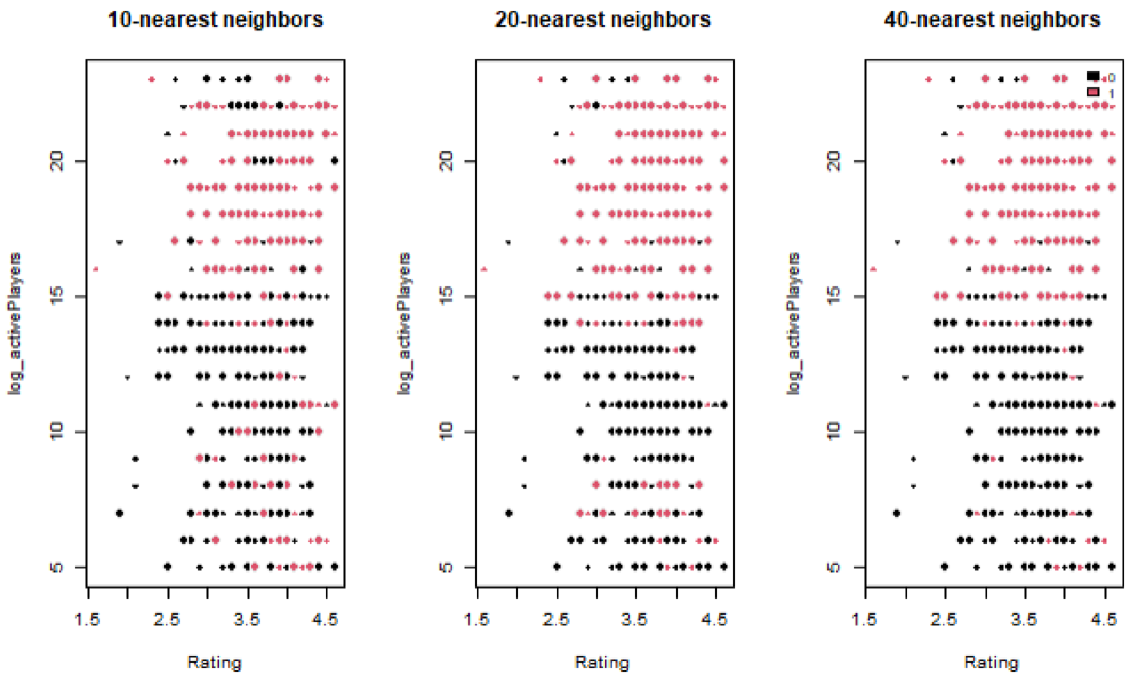
To advise Microsoft on their strategy to produce a game, we built a KNN with K = 40 (square root of dimensions) to classify games based on their attributes that could be determined prior to publishing: publisher, developer, summary words, genre, IP, publishing year, rating (quality of the game), console with classification being a dummy of whether the number of active players are higher than industry average.

Comparing K = 10, 20, and 40, the last model clearly predicts much better than the prior, and it is close to the square root of dimensions convention. The false positive, false negative, sensitivity, and specificity rate are shown below. It looks like the sensitivity is better–our KNN is better at predicting lowly played games, and specificity grows fast from K=20 to K=40.

Table and Plot: KNN comparing Outcomes with K = 10, 20, 40



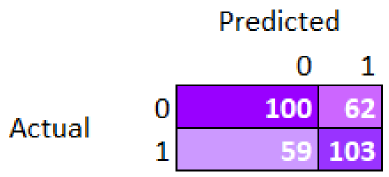
k <dbl>	fp_rate <dbl>	fn_rate <dbl>	sensitivity <dbl>	specificity <dbl>
10	0.4318182	0.4189189	0.6172840	0.5308642
20	0.4021739	0.3714286	0.6790123	0.5432099
40	0.3757576	0.3710692	0.6358025	0.6172840



Choosing K=40, the ROC curve is shown below. The ROC curve for the KNN classifier with k=40 demonstrates its performance in distinguishing between classes. The ROC curve lies above the diagonal dashed line, indicating that the KNN classifier performs better than random guessing. It rises quickly at the beginning, suggesting that the classifier is effective at identifying true positives at lower thresholds. As the false positive rate increases, the curve flattens out. This indicates that the sensitivity (true positive rate) gains diminish with higher false positive rates. In practical terms, this means the classifier starts to overpredict the positive class, resulting in many false positives without a corresponding increase in true positives, similar to what we saw in the

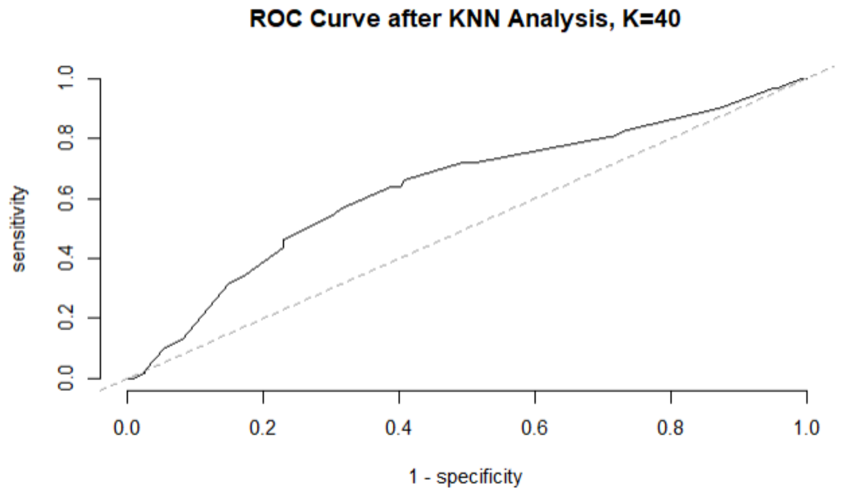
10/20/40 KNN plots–we saw more pink among the lower end but fewer .

Table: Confusion Matrix



		Predicted	
		0	1
Actual	0	100	62
	1	59	103

Figure: ROC Curve after KNN Analysis, K=40

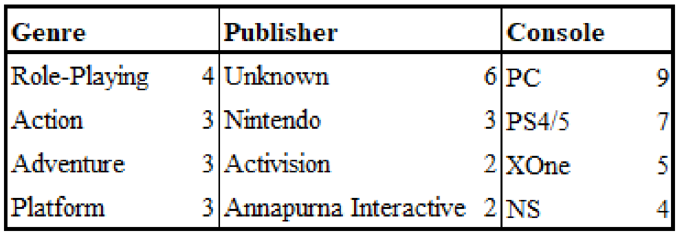


Implication for Microsoft

The analysis highlights that even with control over game quality, genre, and developer, these factors alone do not guarantee higher player counts. This aligns with the intuition that making a popular and engaging game is inherently challenging.

We analyzed the top five clusters where the probability of having above-average numbers of active players exceeded 75%. This analysis highlighted that genres such as role-playing, action, adventure, and platform are more popular. Regarding consoles, PC, PlayStation, Xbox One, and Nintendo Switch showed higher popularity, which is advantageous given Microsoft's ownership of Xbox. Regarding publishers, it is reassuring that having Activision (a subsidiary of Activision Blizzard company) as the publisher is associated with higher active players potentials.

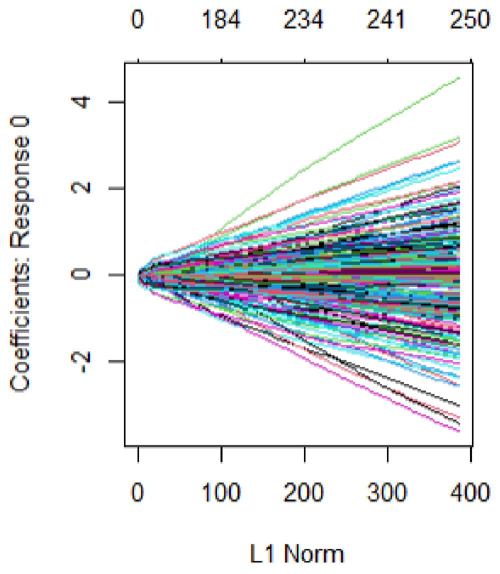
Table: Top 5 K Neighbors and the These Games' Top Attributes with Frequencies

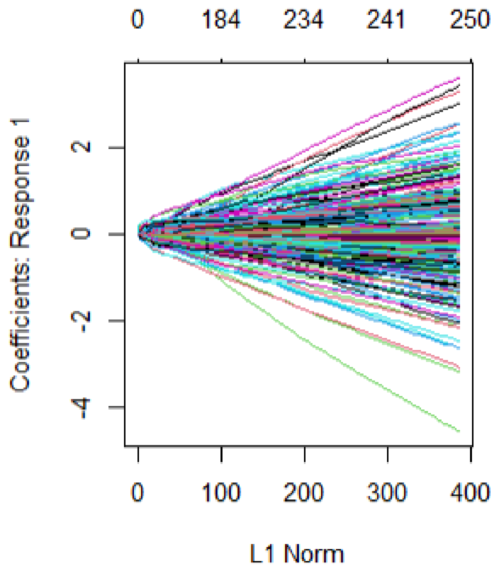


Genre		Publisher		Console	
Role-Playing	4	Unknown	6	PC	9
Action	3	Nintendo	3	PS4/5	7
Adventure	3	Activision	2	XOne	5
Platform	3	Annapurna Interactive	2	NS	4

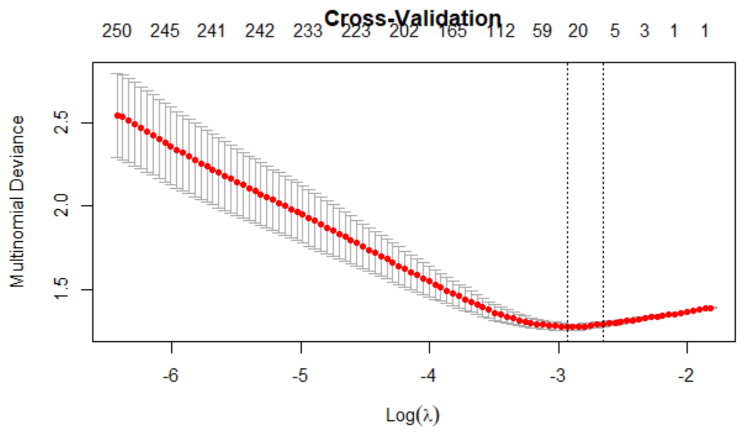
(2) Multinomial Logistic Regression

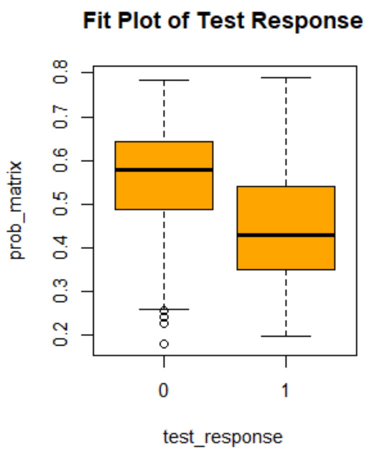
Following the above KNN, we conduct a multinomial logistic regression using 50/50 training testing split, using the same predictors to predict activePlayers_dummy = 1 and 0. The below spaghetti plot shows separate paths for activePlayers_dummy = 1 (high) and 0 (low).





Accuracy is only 0.17 for this model, and multinomial deviance decreases sharply until it reaches the ~30 variables. Similarly, the model does not do well for predicting higher numbers of active players, with median slightly below 50%, worse than random guessing, while at the lower end, the median is ~60%, as shown in the below box fit plot. Again, it proves the point that it is hard to predict if a game could gain huge popularity.

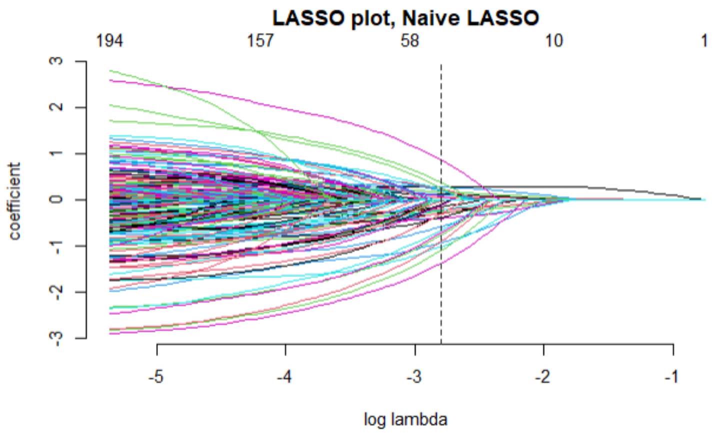




D. Causal Inference - Does higher rating cause higher popularity?

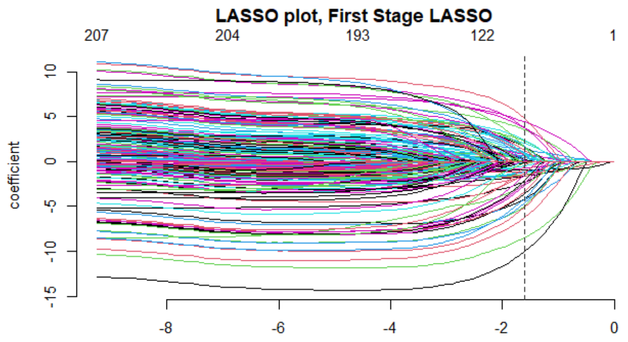
(1) Two-Stage LASSO

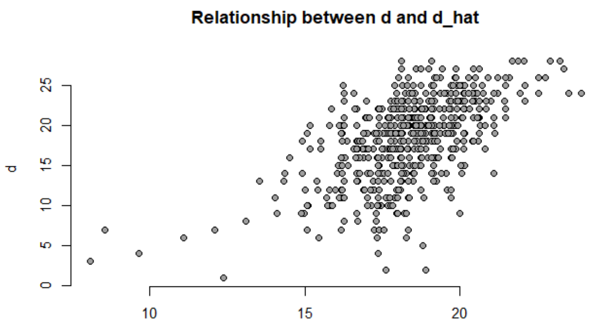
Using a naive Lasso regression, we analyzed the effect of the game rating d on the logarithm of the number of active players y, controlling for genre, publisher, adjusted year, and number of reviews (x). The analysis indicated a positive effect of ~0.05.



To further explore this causal relationship, we implemented a Two-Stage Lasso approach with Bootstrapping:

- **First Stage Lasso**: Predicted the treatment (d_hat) from the controls (x) by conducting a lasso of d on x, yielding an in-sample R2 of 0.351, as seen from the relationship between d and d_hat. This suggests that the controls explain approximately 35.1% of the variability in d, allowing for an independent effect of the rating on the number of players.

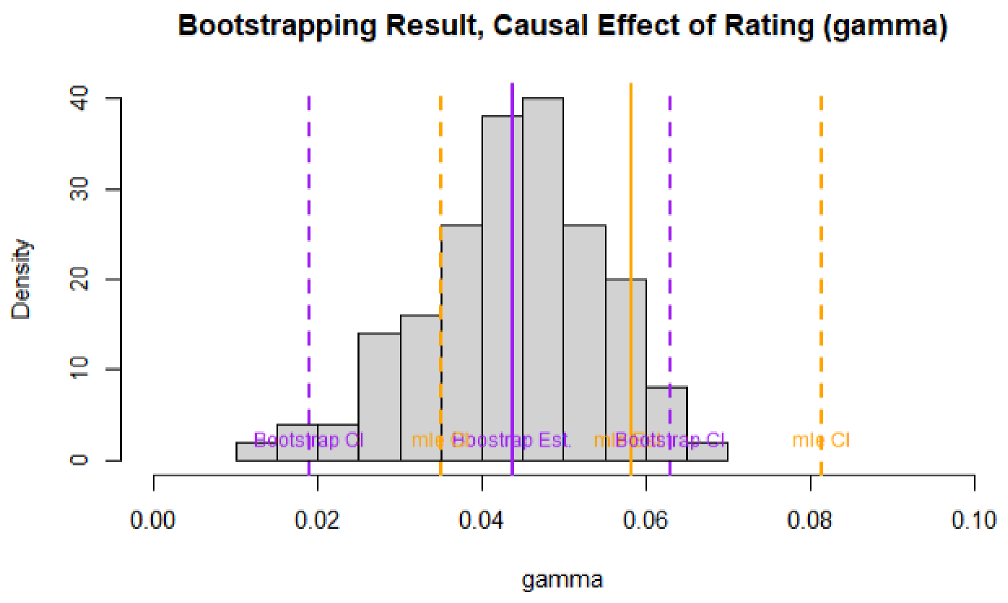




- **Second Stage Lasso**: Combined d_hat, d, and x to model y, confirming a positive causal effect of 0.0414, but much lower than the naive lasso coefficient.

- **Bootstrapping:** We performed 100 bootstrapping iterations on the Two-Stage Lasso to assess the robustness of our findings. The mean estimated treatment effect (gamma or d) was 0.0437, with a median of 0.0445. The histogram below, with 95% confidence intervals from bootstrap (dashed purple lines) and maximum likelihood estimation (MLE, dashed orange lines), supports a positive causal effect. Both intervals exceed zero, confirming the effect's direction, though bootstrapping suggests a slightly smaller effect compared to the naive estimate. Standard errors from MLE and bootstrapping are very similar, ~ 0.011. MLE SEs measure variation in predicted gamma for random draw from the conditional distribution [y|x], whereas bootstrap sees how predicted gamma varies for random draw from the joint distribution for [x, y].

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
0.01370 0.03742 0.04452 0.04369 0.05153 0.06608 
```

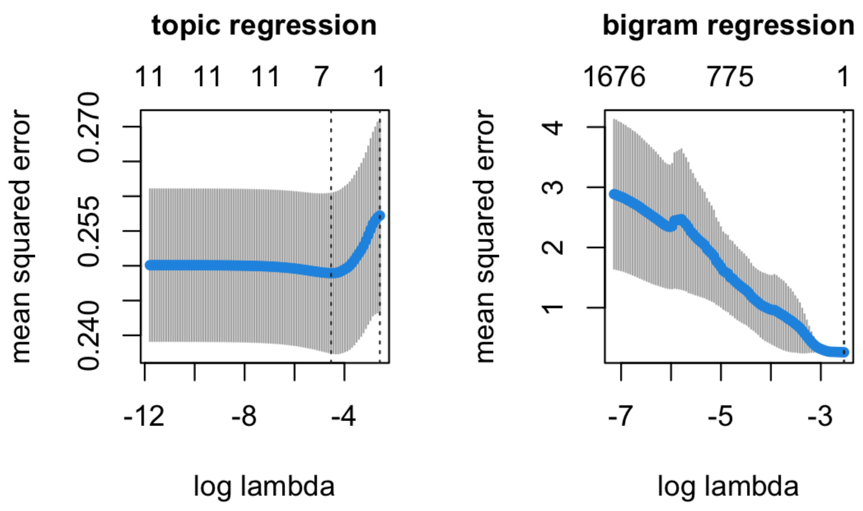


E. Does Review Impact Rating?

(1) Topic Model

We use the topic model to explore the relationship between Reviews and Rating of the games. The topic model demonstrates superior explanatory power, achieving a minimum OOS MSE of 0.258, significantly lower than the standard regression model.

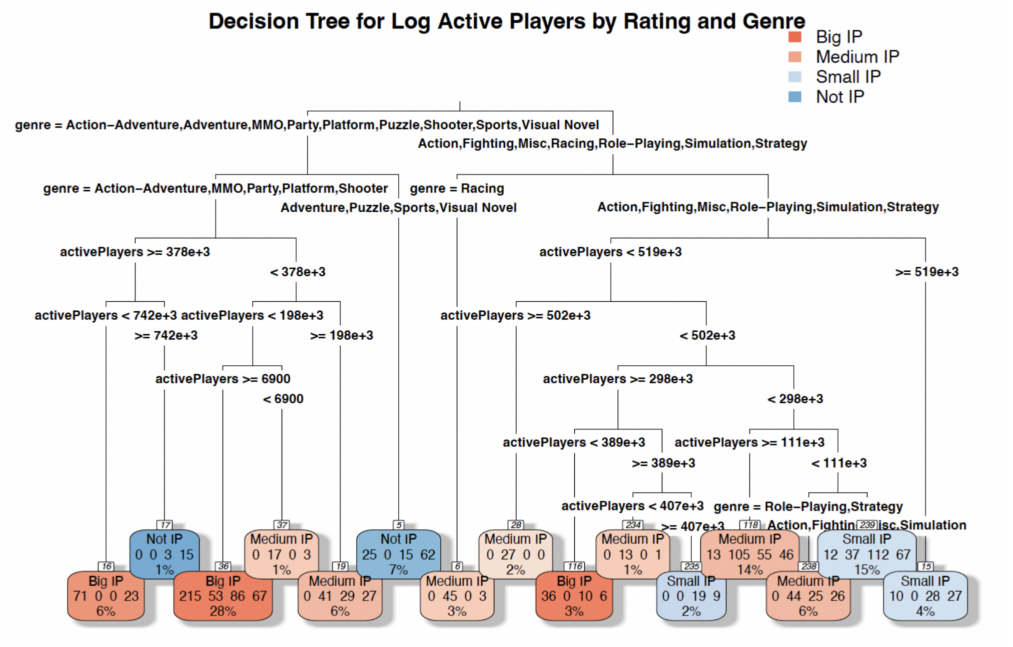
A topic model identifies the hidden themes or topics within a game review. By extracting these underlying topics, the model can better capture the nuances and patterns in the reviews that might be associated with game ratings. This approach allows for a more refined analysis compared to traditional regression methods that might overlook such complexities.



F. What games could lead a game into a potential big IP?

(1) Decision Trees

We find some good games might be developed into IPs which have a series of games. And we want to explore what factors would lead a game into a potential IP, which would benefit to the business decision of Microsoft game move. We would use the tree method to explore it using mainly the genre and active players.


Decision Tree for Log Active Players by Rating and Genre
Big IP
Medium IP
Small IP
Not IP
genre = Action–Adventure,Adventure,MMO,Party,Platform,Puzzle,Shooter,Sports,Visual Novel
Action,Fighting,Misc,Racing,Role–Playing,Simulation,Strategy
genre = Action–Adventure,MMO,Party,Platform,Shooter
Adventure,Puzzle,Sports,Visual Novel
genre = Racing
Action,Fighting,Misc,Role–Playing,Simulation,Strategy
activePlayers >= 378e+3
< 378e+3
activePlayers < 519e+3
>= 519e+3
activePlayers < 742e+3
>= 742e+3
activePlayers < 198e+3
>= 198e+3
activePlayers >= 502e+3
< 502e+3
activePlayers >= 6900
< 6900
activePlayers >= 298e+3
< 298e+3
activePlayers < 389e+3
>= 389e+3
activePlayers >= 111e+3
< 111e+3
activePlayers < 407e+3
>= 407e+3
genre = Role–Playing,Strategy
Action,Fighting,Misc,Simulation
16 Big IP 71 0 0 23 6%
17 Not IP 0 0 3 15 1%
36 Big IP 215 53 86 67 28%
37 Medium IP 0 17 0 3 1%
19 Medium IP 0 41 29 27 6%
5 Not IP 25 0 15 62 7%
6 Medium IP 0 45 0 3 3%
28 Medium IP 0 27 0 0 2%
116 Big IP 36 0 10 6 3%
234 Medium IP 0 13 0 1 1%
235 Small IP 0 0 19 9 2%
118 Medium IP 13 105 55 46 14%
238 Medium IP 0 44 25 26 6%
239 Small IP 12 37 112 67 15%
15 Small IP 10 0 28 27 4%

a. Description of the Decision Tree

The decision tree shown in the figure is used to predict the IP Type (Big IP, Medium IP, Small IP, Not IP) based on the number of active players and the genre of the games. Here's a detailed explanation of the tree:

<u>Root Node:</u>

Group 1: Action-Adventure, Adventure, MMO, Party, Platform, Puzzle, Shooter, Sports, Visual Novel.
Group 2: Racing.
Group 3: Action, Fighting, Misc, Racing, Role-Playing, Simulation, Strategy.

<u>Intermediate Nodes:</u>

For games in Group 1, the next split is based on the number of active players (with a threshold of 378,000 active players).
For games in Group 2, the split is further based on the number of active players (with a threshold of 502,000 active players).
For games in Group 3, the split again is based on the number of active players (with a threshold of 519,000 active players).

Leaf Nodes: The leaf nodes at the bottom show the final classification into IP types. Each node provides the following information:

Number of games in that node.
The proportion of each IP type within that node.
The overall percentage of the dataset represented by that node.

From the tree, we observe that the number of active players and the genre significantly influence the classification into IP types. We can find that some genres of games have great potential to become big IPs, like Action-Adventure, MMO, Party, Platform and Shooter, etc., while others might need to have attributes to attract more players to become bigger IPs. So we strongly suggest prioritizing these types of games which might be of more exciting and story-like genre.

Then we use the tree model to do the prediction by resampling the datasets. Here are the performance metrics of the decision tree model:

Training set prediction(520 datasets)			
Big IPs	Medium IPs	Small IPs	Not IPs
143	154	139	84

Test set prediction(120 datasets)			
Big IPs	Medium IPs	Small IPs	Not IPs
31	38	39	20

- Training Set Accuracy: 30%
  This indicates that the model correctly predicts the IP type for 30% of the training data.
- Test Set Accuracy: 20.31%
  This shows that the model correctly predicts the IP type for approximately 20.31% of the test data.
- Training Set R^2: 0.0549
  The R^2 value for the training set is very low (5.49%), indicating that the model explains only a small portion of the variance in the training data.
- Test Set R^2: 0.0049
  Similarly, the R^2 value for the test set is extremely low (0.49%), suggesting that the model does not generalize well to unseen data and explains very little variance.

From the results of prediction, we find the genre and the amount of players can explain near a half of the IP foundation. For more accurate prediction, we might need to look for other factors.

b. Tree Analysis

This decision tree analysis focuses on predicting the log-transformed number of active players based on the game rating and genre.

The root of the tree firstly splits the data based on the genre. Games in genres like Action-Adventure, Adventure, MMO, Party, Platform, Puzzle, Shooter, Sports, and Visual Novel are grouped together on the left, while genres such as Action, Fighting, Misc, Racing, Role-Playing, Simulation, and Strategy are grouped on the right. This initial split indicates the significant influence of genre on the potential of being IPs.

For games in the left branch (Action-Adventure, Adventure, MMO, etc.), the next split occurs at

378,000 active players. If the number of active players is greater than or equal to 378,000, the tree further splits at 742,000, and if less, another split happens at 198,000 active players. This demonstrates that active player count is a critical factor in distinguishing game popularity within these genres.

For the right branch (Action, Fighting, Misc, etc.), the initial split is at 519,000 active players.For counts greater than or equal to 519,000, the next split is at 298,000 active players, highlighting the importance of active player thresholds in predicting potentials of being IPs.

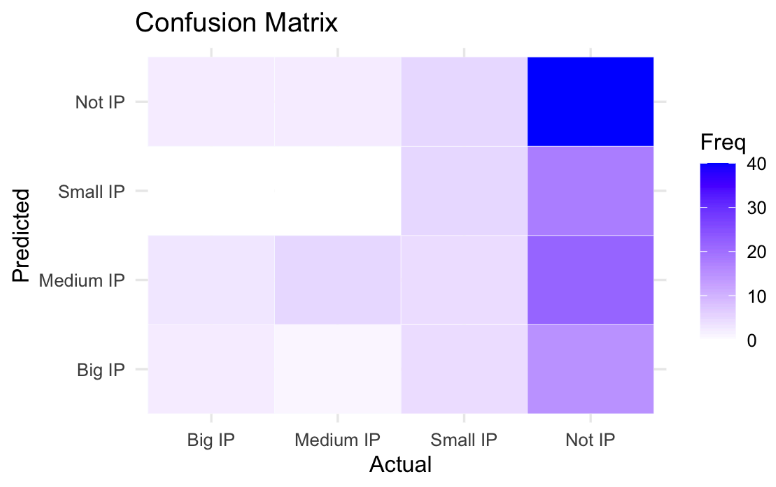
The terminal nodes (leaves) show the distribution of IP_Type across different branches. Nodes with mixed genres and player counts (e.g., 298,000 ≤ active players < 519,000) illustrate how multiple factors (genre and active player counts) interact to influence player engagement.

Implications for Microsoft

To become a Big IP, Microsoft should focus on genres like Action-Adventure, Adventure, MMO, Party, Platform, Puzzle, Shooter, Sports, and Visual Novel, which are more likely to result in higher active player counts. Then they should reach or exceed 378,000 active players is a critical threshold. Within this segment, further strive for thresholds such as 742,000 active players to significantly increase the chances of becoming a Big IP.

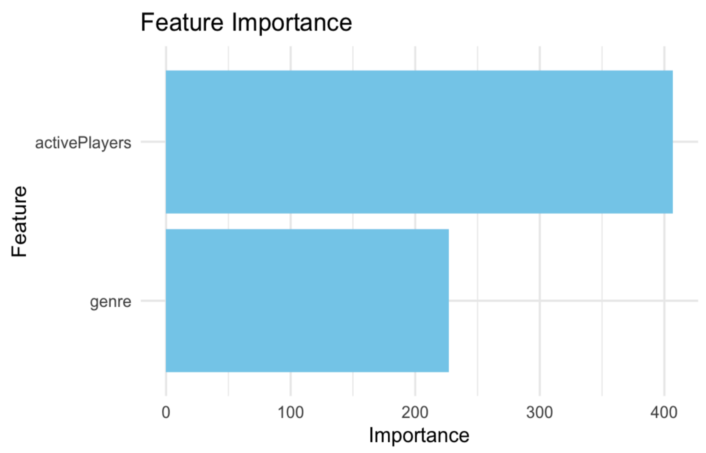
Even for lower thresholds, ensure the game achieves at least 198,000 active players. Improving from lower active player counts to mid-range (e.g., reaching 6900 active players) can already position the game in a favorable trajectory.

For genres like Action, Fighting, Misc, Racing, Role-Playing, Simulation, and Strategy, achieving higher active player counts (e.g., above 519,000) is crucial. Exceeding 502,000 active players within these genres can also position the game towards becoming a Big IP.



Specificity measures the proportion of actual negatives correctly identified. Positive Predictive Value (PPV) measures the proportion of positive predictions that are actually correct.

The model has moderate accuracy for predicting "Medium IP" (0.71667 balanced accuracy) but struggles with "Big IP"(0.55195) and "Small IP"(0.55253). The model has moderate accuracy for predicting "Medium IP" (0.71667 balanced accuracy) but struggles with "Big IP"(0.55195) and "Small IP"(0.55253).



The feature importance plot shows the importance of each feature used in the model. The feature importance plot indicates that activePlayers is a critical factor in predicting IP type, followed by genre.

These results suggest that while the model can capture some patterns in the data, further tuning or additional features may be necessary to improve the accuracy for certain classes, especially "Big IP" and "Small IP".

5. Summary

We have explored from macro to micro perspectives how Microsoft could produce games that will gain high popularity. We have summarized our findings and made recommendations:

Predictive Models of Game Popularity
Our predictive models (LASSO, pruned/unpruned trees, random forest) indicate that the random forest model outperforms others based on out-of-sample R^2. It should be used for predictions based on attributes like genres and consoles before developing a game. The variables 'year' and 'rating' are consistently influential, suggesting that players prefer newer, high-quality games.

Causal Effect of Game Quality
Using two-stage LASSO and bootstrapping, we found that game quality, as indicated by ratings, significantly impacts popularity.

Importance of Game Reviews
Our topic model reveals that themes in player reviews correlate with game ratings, highlighting the importance of feedback. Understanding these themes can help Microsoft tailor game features to meet player expectations and improve ratings.

Challenges in Classifying Popular Games
Our classification models (KNN, Multinomial Logistic Regression) show that predicting game success is more challenging than predicting game failures, even when considering key attributes. The good news is, XBox and Activision Blizzard produced games indicating a positive impact on popularity.

Impact of Game Content
Our LASSO analysis suggests that certain content, such as "colonies" and "fighting," is disliked. Avoiding these themes or shifting to other content could improve game reception.

Building Big IPs
Our IP analysis suggests that Microsoft should use the active player threshold we defined to position their games towards becoming major intellectual properties.

Strategic Recommendations for Microsoft

We recommend that Microsoft prioritize developing high-quality games, particularly in genres with high potential for active player engagement, such as Action-Adventure and Role-Playing, on both Xbox and PC. Leveraging player reviews and making informed decisions on game content, based on our analysis, can further enhance game success. Additionally, developing sequels for popular franchises can capitalize on established player bases and ensure strong sales. Microsoft, with the acquisition of Activision Blizzard, is well suited in the gaming industry to go for big sales given its existing franchises, consoles, and reputation. Implementing these strategies will help Microsoft navigate the competitive gaming industry and achieve sustained growth.

6. Appendices and Code

We have included in this section the R code we used to conduct the data processing, produce the tables, models, plots, and additional output not shown in the main sections.

Section 2.A/B, Data Cleaning and Variable Creation, P4

```
## Data Cleaning and Variable Creation
# Load the datasets
popular_vg <- read.csv("popular_vg_1980-2023.csv", stringsAsFactors = FALSE)
vgchartz_2024 <- read.csv("vgchartz-2024.csv", stringsAsFactors = FALSE)
names(vgchartz_2024)[names(vgchartz_2024) == "title"] <- "Title"
popular_vg$Release.Date <- as.Date(popular_vg$Release.Date,
format="%Y-%m-%d")
popular_vg$year <- year(popular_vg$Release.Date)
popular_vg <- popular_vg[order(popular_vg$year), ]
# Clean two datasets
popular_vg <- popular_vg %>% group_by(Title) %>%
  summarise(
    Release.Date = first(Release.Date),
    year = first(year),
    Team = first(Team),
    Rating = first(Rating),
    Times.Listed = first(Times.Listed),
    Number.of.Reviews = first(Number.of.Reviews),
    Genres = first(Genres),
    Summary = first(Summary),
    Reviews = paste(unique(Reviews), collapse = " "),
    Plays = first(Plays),
    Playing = first(Playing),
    Backlogs = first(Backlogs),
    Wishlist = first(Wishlist),
    .groups = 'drop')
vgchartz_2024 <- vgchartz_2024 %>%
  group_by(Title) %>%
  summarise(
    console = paste(unique(console), collapse = " "),
    publisher = first(publisher),
    developer = first(developer),
    genre = first(genre),
    total_sales = sum(total_sales),  # Corrected typo here
    .groups = 'drop')
```

```r
# Perform the inner join
game <- inner_join(vgchartz_2024[, c("Title", "console", "publisher",
"developer", "genre", "total_sales")], popular_vg[, c("Title","Release.Date",
"Genres", "Rating", "Times.Listed", "year", "Number.of.Reviews", "Summary",
"Reviews", "Plays", "Playing", "Backlogs", "Wishlist")], by = "Title")
game <- game %>% distinct()
# Convert 'Plays' and 'Number.of.Reviews' from K format to numeric
game$Plays <- as.numeric(str_replace(game$Plays, "K", "")) * 1000
game$Playing <- as.numeric(str_replace(game$Playing, "K", "")) * 1000
game$Backlogs <- as.numeric(str_replace(game$Backlogs, "K", "")) * 1000
game$Wishlist <- as.numeric(str_replace(game$Wishlist, "K", "")) * 1000
game$Number.of.Reviews <- as.numeric(str_replace(game$Number.of.Reviews, "K",
"")) * 1000
# Keep only post 2005 data
game <- game %>% filter(year >= 2005)
# Remove rows with NAs only in the specified columns
columns_to_clean <- c("Rating", "Summary", "Reviews", "year")
game <- game %>%
  filter(!if_any(all_of(columns_to_clean), is.na))
# Create additional vars
game <- game %>%
  mutate(atvi_indi = ifelse(publisher %in% c("Activision", "Blizzard
Entertainment") | developer == "Blizzard Entertainment", 1, 0))
game$activePlayers <- game$Plays + game$Playing
game$allPlayers <- game$Plays + game$Playing + game$Backlogs
game$year_adj <- game$year - 2000
game$log_activePlayers <- log(game$activePlayers+1)
game$log_allPlayers <- log(game$allPlayers+1)
game <- game %>% mutate(activePlayers_dummy = ifelse(allPlayers >
median(game$activePlayers), 1, 0))
summary(game)
```

```r
## Clean Summary words
# Define filler words to be removed
filler_words <- c("the", "is", "a", "has", "have", "and", "of", "in", "to",
"for", "with", "on", "that", "lets","as", "out", "by","from", "this", "be",
"an", "v", "or", "so", "you", "are", "can", "will", "which", "t", "who",
"where", "also", "his", "her", "their", "they", "up", "he", "she", "its",
"it", "includes", "include","your", "you", "all","���������", "s",
"any", "ll", "was", "but", "if", "there", "these")

# Function to clean and tokenize text, removing filler words
clean_and_tokenize <- function(text, filler_words) {
  text_clean <- tolower(enc2utf8(text))
  # text_clean <- tolower(gsub("[^\\x01-\\x7F]", "", text_clean))
  text_clean <- str_replace_all(text_clean, "[[:punct:]]", " ")
  words <- unlist(strsplit(text_clean, "\\s+"))
  words <- words[!words %in% filler_words]
  return(words)
}

game$Summary_clean <- sapply(game$Summary, function(x)
paste(clean_and_tokenize(x, filler_words), collapse = " "))

# Create a corpus from the cleaned summaries
corpus <- Corpus(VectorSource(game$Summary_clean))
tdm <- TermDocumentMatrix(corpus, control = list(wordLengths = c(1, Inf)))
tdm_matrix <- as.matrix(tdm)

# Get word frequencies
word_freq <- sort(rowSums(tdm_matrix), decreasing = TRUE)
word_freq_df <- data.frame(word = names(word_freq), frequency = word_freq)

# Filter for significant words
sig_word_sum <- word_freq_df %>% filter(frequency > 5)
sig_word_sum <- sig_word_sum %>% arrange(word) %>% slice(-1:-19)
sig_word_sum <- as.character(sig_word_sum$word)
tdm_filtered <- tdm_matrix[sig_word_sum, ]
tdm_sparse <- as(t(tdm_filtered), "sparseMatrix")

# Filter the term-document matrix to keep only significant words
tdm_filtered <- tdm_matrix[sig_word_sum, ]
tdm_sparse <- as(t(tdm_filtered), "sparseMatrix")
```

```r
## Processing Console

consoles_list <- strsplit(game$console, split = " ")
unique_consoles <- unique(unlist(consoles_list))
console_matrix <- matrix(0, nrow = nrow(game), ncol =
length(unique_consoles), dimnames = list(NULL, unique_consoles))
for (i in seq_along(consoles_list)) {
  console_matrix[i, consoles_list[[i]]] <- 1
}
console_sparse <- Matrix(console_matrix, sparse = TRUE)
console_df <- as.data.frame(as.matrix(console_sparse))
colnames(console_df) <- unique_consoles

## Combine x

set.seed(1234)

genre_dummies <- model.matrix(~ genre - 1, data = game)
year_cont <- model.matrix(~ year_adj, data = game)
```

```
publisher_dummies <- model.matrix(~ publisher - 1, data = game)
game$Rating <- as.numeric(game$Rating)
rating_cont <- model.matrix(~ Rating, data = game)
ip_dummies <- model.matrix(~ IP_Type - 1, data = game)
x <- cbind(genre_dummies, year_cont, publisher_dummies, rating_cont,
ip_dummies, tdm_sparse, console_sparse)
x <- as(x, "sparseMatrix")

# Make into a df for 3.B(2)(3) trees and forest
x_tree <- cbind(game[, c("year_adj", "publisher", "genre", "IP_Type",
"Rating")], console_df)

y <- game$log_activePlayers
```

Section 3.A, Exploring Potential Y Variables - Plots, P6-7

```
# Plot the distribution of total_sales and plays
ggplot(game, aes(x = total_sales)) +
  geom_histogram(bins = 30, fill = "pink", color = "black") +
  theme_minimal() +
  labs(title = "Distribution of Total Sales (2005-2023)", x = "Total Sales",
y = "Frequency") +
  theme(plot.title = element_text(hjust = 0.5),
        text = element_text(size = 12),
        axis.title = element_text(size = 12, face = "bold"))

ggplot(game, aes(x = total_sales)) +
  geom_histogram(bins = 30, fill = "pink", color = "black") +
  theme_minimal() +
  scale_x_log10() +
  labs(title = "Distribution of Total Sales (Log Scale) (2005-2023)", x =
"Total Sales (Log Scale)", y = "Frequency") +
  theme(plot.title = element_text(hjust = 0.5),
        text = element_text(size = 12),
        axis.title = element_text(size = 12, face = "bold"))
```

```
# Reshaping the data to long format
game_long <- game %>%
  # select(allPlayers, activePlayers) %>%  # Select the necessary columns
  pivot_longer(
    cols = c(allPlayers, activePlayers),
    names_to = "PlayerType",
    values_to = "Players"
  )

ggplot(game_long, aes(x = Players, fill = PlayerType)) +
  geom_histogram(bins = 30, alpha = 0.6, position = "identity", color =
"black") +
  scale_fill_manual(values = c("orange", "purple")) +
  labs(title = "Distribution of All Players vs. Active Players (2005-2023)",
       x = "Number of Players",
       y = "Frequency") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5),
        text = element_text(size = 12),
        axis.title = element_text(size = 12, face = "bold"))

ggplot(game_long, aes(x = Players, fill = PlayerType)) +
  geom_histogram(bins = 30, alpha = 0.6, position = "identity", color =
"black") +
  scale_x_log10() +
  scale_fill_manual(values = c("orange", "purple")) +
  labs(title = "Distribution of All Players vs. Active Players (Log Scale)
(2005-2023)",
       x = "Number of Players (Log Scale)",
       y = "Frequency") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5),
        text = element_text(size = 12),
        axis.title = element_text(size = 12, face = "bold"))

# Plot relationship between total sales and players
p <- ggplot() +
  geom_point(data = game, aes(x = activePlayers, y = total_sales, color =
"Active Players")) +
  geom_smooth(data = game, aes(x = activePlayers, y = total_sales, color =
"Active Players"), method = "lm", fill = "orange") +
  geom_point(data = game, aes(x = allPlayers, y = total_sales, color = "All
Players")) +
  geom_smooth(data = game, aes(x = allPlayers, y = total_sales, color = "All
Players"), method = "lm", fill = "pink") +
  labs(title = "Total Sales vs Players (2005-2023)",
       x = "Players",
       y = "Total Sales",
       color = "Player Type") +  # Label for the legend
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5),
        text = element_text(size = 12),
        axis.title = element_text(size = 12, face = "bold")) +
  scale_x_continuous(limits = c(0, 1000000)) +
  scale_color_manual(values = c("Active Players" = "orange", "All Players" =
"purple"))
```

Section 3.B, Exploring Potential Numerical X variables - Plots, P8-10

```
col_purple_orange <- colorRampPalette(c("orange","purple"))
# Plot the correlation matrix with correlation coefficients
```

```
numeric_vars <- game[, c("activePlayers", "allPlayers", "Number.of.Reviews",
"Rating", "Backlogs", "Wishlist", "year")]
cor_matrix <- cor(numeric_vars)
corrplot(cor_matrix, method = "circle", col = col_purple_orange(200),
         type = "upper", order = "hclust", tl.col = "black", tl.srt = 45,
         addCoef.col = "black", cl.cex = 0.8, number.cex = 0.8)
title("Correlation Matrix of Numeric Variables", line = 2.5, cex.main = 1.2)

# Plot the pair plots

pairs(numeric_vars, col = "orange", main = "Pairs Plot of Numeric Variables")
cor_matrix

##                   activePlayers   allPlayers Number.of.Reviews      Rating
## activePlayers        1.00000000  0.608729668      -0.113887930  0.18445853
## allPlayers           0.60872967  1.000000000      -0.004954418 -0.01984942
## Number.of.Reviews   -0.11388793 -0.004954418       1.000000000 -0.05016626
## Rating               0.18445853 -0.019849424      -0.050166264  1.00000000
## Backlogs            -0.06767694  0.750361763       0.088658885 -0.17864890
## Wishlist            -0.06560575 -0.041325219       0.345962458  0.05847168
## year                 0.33535884  0.201225221      -0.004566309  0.01196529
##                       Backlogs     Wishlist         year
## activePlayers     -0.067676940 -0.065605754  0.335358841
## allPlayers         0.750361763 -0.041325219  0.201225221
## Number.of.Reviews  0.088658885  0.345962458 -0.004566309
## Rating            -0.178648902  0.058471680  0.011965287
## Backlogs           1.000000000  0.002693206 -0.026365140
## Wishlist           0.002693206  1.000000000 -0.044853688
## year              -0.026365140 -0.044853688  1.000000000

# Plot histograms
p1 <- ggplot(game, aes(x = Number.of.Reviews)) +
  geom_histogram(fill = "purple", color = "black") +
  theme_minimal() +
  labs(title = "Histogram of Number of Reviews", x = "Number of Reviews", y =
"Frequency") +
  theme(plot.title = element_text(hjust = 0.5))
p2 <- ggplot(game, aes(x = Rating)) +
  geom_histogram(binwidth = 0.1, fill = "purple", color = "black") +
  theme_minimal() +
  labs(title = "Histogram of Ratings", x = "Rating", y = "Frequency") +
  theme(plot.title = element_text(hjust = 0.5))

p3 <- ggplot(game, aes(x = Wishlist)) +
  geom_histogram(fill = "purple", color = "black") +
  theme_minimal() +
  labs(title = "Histogram of Wishlists", x = "Number of Wishlists", y =
"Frequency") +
  theme(plot.title = element_text(hjust = 0.5))
```

```
# Plot average per year
avg_reviews_per_year <- ggplot(game, aes(x = year, y = Number.of.Reviews)) +
  stat_summary(fun = mean, geom = "bar", fill = "orange", color = "black") +
  theme_minimal() +
  labs(title = "Average Number of Reviews per Year", x = "Year", y = "Average
Number of Reviews") +
  theme(plot.title = element_text(hjust = 0.5))

avg_rating_per_year <- ggplot(game, aes(x = year, y = Rating)) +
  stat_summary(fun = mean, geom = "bar", fill = "orange", color = "black") +
  theme_minimal() +
  labs(title = "Average Rating per Year", x = "Year", y = "Average Rating") +
  theme(plot.title = element_text(hjust = 0.5))

avg_wishlists_per_year <- ggplot(game, aes(x = year, y = Wishlist)) +
  stat_summary(fun = mean, geom = "bar", fill = "orange", color = "black") +
  theme_minimal() +
  labs(title = "Average Number of Wishlists per Year", x = "Year", y =
"Average Number of Wishlists") +
  theme(plot.title = element_text(hjust = 0.5))

ggplot(game, aes(x = Rating)) +
  geom_histogram(bins = 30, fill = "pink", color = "black") +
  theme_minimal() +
  scale_x_log10() +
  labs(title = "Distribution of Rating (2005-2023)", x = "Rating", y =
"Frequency") +
  theme(plot.title = element_text(hjust = 0.5),
        text = element_text(size = 12),
        axis.title = element_text(size = 12, face = "bold"))
```

Section 3.B, Exploring Potential Categorical X variables - Plots, P8-10

```
## Games by Genre
# Number of Games by Genre
ggplot(game, aes(x = genre, fill = genre)) +
  geom_bar(color = "black") +
  labs(title = "Number of Games by Genre (2005-2023)",
       x = "",
       y = "Count") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5),
        text = element_text(size = 12),
        axis.title = element_text(size = 12, face = "bold"),
        legend.title = element_text(),
        axis.text.x = element_blank(),
        axis.ticks.x = element_blank(),
        axis.title.x = element_blank())

# Total Players by Genre
```

```r
ggplot(game, aes(x = genre, y = allPlayers, fill = genre)) +
  geom_boxplot() +
  scale_y_log10() +
  labs(title = "Total Players by Genre (2005-2023)",
       x = "Genre",
       y = "Total Players") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5),
        text = element_text(size = 12),
        axis.title = element_text(size = 12, face = "bold"),
        legend.title = element_text(),
        axis.text.x = element_blank())

# Prepare the data: count games per year per genre
game_year_genre <- game %>%
  group_by(year, genre) %>%
  summarise(count = n(), .groups = 'drop')


# Trend of Games (by Counts) by Genre Per Year
ggplot(game_year_genre, aes(x = year, y = count, color = genre, group =
genre)) +
  geom_line() +
  labs(title = "Trend of Games (by Counts) by Genre Per Year", x = "Year", y
= "Counts") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5),
        axis.text.x = element_text(angle = 45, hjust = 1),
legend.title = element_blank())

game_year_genre <- game %>%
  group_by(year, genre) %>%
  summarise(Average_Rating = mean(Rating, na.rm = TRUE), .groups = 'drop')

# Trend of Games (by Rating) by Genre Per Year
ggplot(game_year_genre, aes(x = year, y = Average_Rating, color = genre,
group = genre)) +
  geom_line() +
  labs(title = "Trend of Games (by Rating) by Genre Per Year",
       x = "Year",
       y = "Average Rating") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5),
        axis.text.x = element_text(angle = 45, hjust = 1),
        legend.title = element_blank())

# Game Ratings by Genre Over Years
ggplot(game, aes(x = Release.Date, y = Rating, color = genre)) +
  geom_point(alpha = 0.5, size = 2) +
  labs(title = "Game Ratings by Genre Over Years",
       x = "Release.Date",
       y = "Rating") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5),
        axis.title = element_text(face = "bold"),
        legend.title = element_text(face = "bold")) +
  guides(color = guide_legend(title = "Genre"))

## Games by Console

game_long <- game %>%
  tidyr::separate_rows(console, sep = " ") %>%
  filter(!is.na(console)) # Ensure console is not NA

console_counts <- game_long %>%
  group_by(console) %>%
  summarise(Count = n(),
            Total_activePlayers = sum(as.numeric(activePlayers), na.rm =
TRUE),
            Total_Players = sum(as.numeric(allPlayers), na.rm = TRUE),
            Average_Rating = mean(as.numeric(Rating), na.rm = TRUE)) %>%
  ungroup()

top_consoles <- console_counts %>%
  top_n(10, Count) %>%
  arrange(desc(Count))
other <- console_counts %>%
  filter(!console %in% top_consoles$console) %>%
  summarise(console = "Other",
            Count = sum(Count),
            Total_activePlayers = sum(Total_activePlayers),
            Total_Players = sum(Total_Players),
            Average_Rating = mean(Average_Rating))
final_console_data <- bind_rows(top_consoles, other)

# Pie chart: distribution of games by console
pie_data <- final_console_data %>%
  mutate(label = scales::percent(Count / sum(Count)))

ggplot(pie_data, aes(x = "", y = Count, fill = console)) +
  geom_col(color = "black") +
  geom_text(aes(label = Count),
            position = position_stack(vjust = 0.5)) +
  labs(title = "Number of Games across Consoles (2005-2023)") +
  theme_void() +
  coord_polar(theta = "y") +
  theme(plot.title = element_text(hjust = 0.5),
        text = element_text(size = 12),
        axis.title = element_text(size = 12, face = "bold"))
```

```
# Total Players per Console (2005-2023)
ggplot(final_console_data, aes(x = reorder(console, Total_Players, decreasing
= TRUE), y = Total_Players, fill = console)) +
  geom_bar(stat = "identity") +
  labs(title = "Total Players per Console (2005-2023)", x = "Console", y =
"Total Players") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5),
        axis.text.x = element_text(angle = 45, hjust = 1))
game_long <- game %>%
  separate_rows(console, sep = " ") %>%
  mutate(allPlayers = as.numeric(allPlayers),
         year = as.numeric(year)) %>%
  filter(!is.na(console) & !is.na(allPlayers))

# Summarize data by console and year, considering only top consoles and
others
yearly_console_data <- game_long %>%
  group_by(year, console) %>%
  summarise(Total_Players = sum(allPlayers, na.rm = TRUE), .groups = 'drop')

# Incorporating top 10 consoles logic and "Other"
top_consoles_list <- top_consoles$console  # Extract just the console names
from the previous aggregation

yearly_console_data <- yearly_console_data %>%
  mutate(Grouped_Console = if_else(console %in% top_consoles_list,
as.character(console), "Other")) %>%
  group_by(year, Grouped_Console) %>%
  summarise(Total_Players = sum(Total_Players), .groups = 'drop') %>%
  ungroup()


# Total Players per Console Over Years (2005-2023)
ggplot(yearly_console_data, aes(x = year, y = Total_Players, group =
Grouped_Console, color = Grouped_Console)) +
  geom_line() +
  geom_point() +
  labs(title = "Total Players per Console Over Years (2005-2023)",
       x = "Year",
       y = "Total Players",
       color = "Console") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5),
        axis.text.x = element_text(angle = 45, hjust = 1),
        legend.title = element_blank())

# Average Rating per Console
ggplot(final_console_data, aes(x = reorder(console, Average_Rating,
decreasing = TRUE), y = Average_Rating, fill = console)) +
  geom_bar(stat = "identity") +
  labs(title = "Average Rating per Console", x = "Console", y = "Average
Rating") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5),
        axis.text.x = element_text(angle = 45, hjust = 1))


## Plot word cloud
# Create the word cloud
word_freq_filtered <- sort(rowSums(tdm_filtered), decreasing = TRUE)
wordcloud(names(word_freq_filtered), freq = word_freq_filtered, min.freq = 1,
scale = c(4, 0.5), colors = brewer.pal(8, "Dark2"))
```

Section 3.B.c), Exploring Activision Blizzard Games Compared to All Games - Plots, P18

```
# Plot for Activision Blizzard games
p1 <- ggplot(game %>% filter(atvi_indi == 1), aes(x = Rating, y =
activePlayers)) +
  geom_point(alpha = 0.4, color = "purple") +
  geom_smooth(method = "lm", color = "purple", fill = "purple", se = TRUE) +
  labs(title = "Relationship between Game Rating and Number of Players for
ATVI Games",
       x = "Game Rating", y = "Number of Active Players") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))

# Plot for all games
p2 <- ggplot(game, aes(x = Rating, y = activePlayers)) +
  geom_point(alpha = 0.4, color = "orange") +
  geom_smooth(method = "lm", color = "red", fill = "orange", se = TRUE) +
  labs(title = "Relationship between Game Rating and Number of Players for
All Games",
       x = "Game Rating", y = "Number of Active Players") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```

Section 4.A.(1), Linear Regression and FDR Using Existing Variables, P19

```
predictors_fdr <- cbind(year_cont, genre_dummies, publisher_dummies,
wishlist_dummies, developer_dummies, nReviews, nRating)
predictors_fdr <- as(predictors_fdr, "sparseMatrix")
predictors_fdr <- as.data.frame(as.matrix(predictors_fdr))

fit <- lm(log_activePlayers ~ ., data=game_console)
summary_fit <- summary(fit)
coefficients <- summary_fit$coefficients
```

```
mrgpvals <- coefficients[, 4]
source("fdr.R")
cutoff <- fdr_cut(mrgpvals,0.2,TRUE)

significant_indices <- mrgpvals <= cutoff
significant_coefficients <- coefficients[significant_indices, ]
significant_predictors <- rownames(coefficients)[significant_indices]

significant_predictors

##  [1] "(Intercept)"                     "year_adj"
##  [3] "publisherCD Projekt Red Studio" "publisherEA Sports"
##  [5] "publisherMatt Makes Games Inc." "publisherMedia.Vision"
##  [7] "publisherTeam Cherry"            "genreMisc"
##  [9] "genrePlatform"                   "Rating"
## [11] "PS4"

kable(table(mrgpvals<cutoff))
```

Var1	Freq
FALSE	194
TRUE	10

```
cutoff

## [1] 0.00931286
```

Section 4.A.(2), LASSO Path Plot and CV Plot Using Existing Variables, P20

```
lasso3.cv <- cv.gamlr(x, game$log_activePlayers, lambda.min.ratio = 1e-3,
                      family = "gaussian", verb = TRUE)
sum(coef(lasso3.cv)!=0) # 1se

## [1] 7

sum(coef(lasso3.cv, s="min")!=0) # min

## [1] 38

sum(coef(lasso3.cv$gamlr)!=0) # AICc

## [1] 42

## log lambdas selected under various criteria
log_lambdas <- function(cv_obj) {
  gamlr_obj <- cv_obj$gamlr
  n_lambdas <- length(gamlr_obj$lambda)
  n <- nrow(cv_obj$gamlr$x)

  # Calculate AIC, AICc, and BIC
  aic_values <- AIC(gamlr_obj)
  aicc_values <- AICc(gamlr_obj)
  bic_values <- BIC(gamlr_obj)

  # Extracting lambda values
  lambda_aicc <- gamlr_obj$lambda[which.min(aicc_values)]
  lambda_aic <- gamlr_obj$lambda[which.min(aic_values)]
  lambda_bic <- gamlr_obj$lambda[which.min(bic_values)]
  lambda_min <- cv_obj$lambda.min
  lambda_1se <- cv_obj$lambda.1se

  return(list(lambda_aicc = lambda_aicc,
              lambda_aic = lambda_aic,
              lambda_bic = lambda_bic,
              lambda_min = lambda_min,
              lambda_1se = lambda_1se))
}

lambdas <- log_lambdas(lasso3.cv)

# Log lambdas
log(lambdas$lambda_aicc)

##     seg31
## -2.849217

log(lambdas$lambda_aic)

##     seg31
## -2.849217

log(lambdas$lambda_bic)

##     seg14
## -1.663037

log(lambdas$lambda_min)

## [1] -2.779441

log(lambdas$lambda_1se)

## [1] -1.872362

# Plot the LASSO path from gamlr
plot(lasso3.cv$gamlr, main = "LASSO Path with AIC, AICc, BIC, and CV")

# Adding vertical lines for the different criteria
abline(v = log(lambdas$lambda_aicc), col = "black", lty = 2)
abline(v = log(lambdas$lambda_aic), col = "purple", lty = 2)
abline(v = log(lambdas$lambda_bic), col = "green", lty = 2)
abline(v = log(lambdas$lambda_min), col = "orange", lty = 2)
abline(v = log(lambdas$lambda_1se), col = "blue", lty = 2)
```

```
legend("topright", bty = "n", lwd = 1,
       col = c("black", "purple", "green", "orange", "blue"),
       legend = c("AICc", "AIC", "BIC", "CV.min", "CV.1se"))

# Plot the cross-validation plot
plot(lasso3.cv, main = "Cross-Validation")

y_pred <- predict(lasso3.cv, x, select = "min")
rss <- sum((game$log_activePlayers - y_pred)^2)
tss <- sum((game$log_activePlayers - mean(game$log_activePlayers))^2)
R2<- 1 - rss/tss
```

Section 4.B, OOS R2 Comparison, P20

(1)LASSO

Similar to above, with different predictors X. We will proceed with OOS testing code.

```
# Function to split data and calculate OOS R² for LASSO
calculate_oos_r2 <- function(x, y, train_fraction = 0.7, n_splits = 20) {
  oos_r2_values <- numeric(n_splits)

  for (i in 1:n_splits) {
    # Split data into training and testing sets
    train_indices <- sample(1:nrow(x), size = floor(train_fraction *
nrow(x)))
    test_indices <- setdiff(1:nrow(x), train_indices)

    x_train <- x[train_indices, ]
    y_train <- y[train_indices]
    x_test <- x[test_indices, ]
    y_test <- y[test_indices]

    # Fit LASSO model
    lasso_model <- cv.gamlr(x_train, y_train, lambda.min.ratio = 1e-3, family
= "gaussian")

    # Predict on test set using the lambda with minimum cross-validated error
    y_pred <- predict(lasso_model, x_test, select = "min")

    # Calculate OOS R²
    rss <- sum((y_test - y_pred)^2)
    tss <- sum((y_test - mean(y_test))^2)
    oos_r2 <- 1 - rss/tss

    oos_r2_values[i] <- oos_r2
  }

  return(oos_r2_values)
}
```

```r
# Apply function to your data
x <- as.matrix(x)  # Ensure x is a matrix
y <- game$log_activePlayers
oos_r2_lasso <- calculate_oos_r2(x, y)

# Combine OOS R² values into a data frame
model_names <- rep(c("LASSO"), each = 20)  # Extend with other model names
oos_r2_values <- c(oos_r2_lasso)  # Combine with other OOS R² values

results_df <- data.frame(model = model_names, OOS_R2 = oos_r2_values)
```

(2)Unpruned Tree & Pruned Tree

```r
genre_df <- as.data.frame(model.matrix(~ genre - 1, data = game))
publisher_df <- as.data.frame(model.matrix(~ publisher - 1, data = game))
ip_df <- model.matrix(~ IP_Type - 1, data = game)
tdm_df <- as.data.frame(as.matrix(tdm_sparse))
x_tree <- cbind(game[, c("year_adj", "Rating")], genre_df, publisher_df,
ip_df, console_df) # , tdm_df
x_tree$year_adj <- scale(x_tree$year_adj)
x_tree$Rating <- scale(x_tree$Rating)

# Fit the unpruned decision tree model
tree_model <- rpart(game$log_activePlayers ~ ., data = data.frame(x_tree),
control = rpart.control(cp = 0))

# Plot the unpruned tree
rpart.plot(tree_model, main = "Unpruned Decision Tree")

# Function to split data and calculate OOS R² for the unpruned tree
calculate_oos_r2_tree <- function(x, y, train_fraction = 0.7, n_splits = 20)
{
  oos_r2_values <- numeric(n_splits)

  for (i in 1:n_splits) {
    # Split data into training and testing sets
    train_indices <- sample(1:nrow(x), size = floor(train_fraction *
nrow(x)))
    test_indices <- setdiff(1:nrow(x), train_indices)

    x_train <- x[train_indices, ]
    y_train <- y[train_indices]
    x_test <- x[test_indices, ]
    y_test <- y[test_indices]

    # Fit the unpruned decision tree model
    tree_model <- rpart(y_train ~ ., data = data.frame(x_train), control =
rpart.control(cp = 0))
```

```
    # Predict on the test set
    y_pred <- predict(tree_model, newdata = data.frame(x_test))

    # Calculate OOS R²
    rss <- sum((y_test - y_pred)^2)
    tss <- sum((y_test - mean(y_test))^2)
    oos_r2 <- 1 - rss/tss

    oos_r2_values[i] <- oos_r2
  }

  return(oos_r2_values)
}

oos_r2_tree <- calculate_oos_r2_tree(x_tree, game$log_activePlayers)


## Pruned tree
# Prune the tree based on the cp that minimizes cross-validated error
cp_table <- printcp(tree_model)
##
## Regression tree:
## rpart(formula = game$log_activePlayers ~ ., data = data.frame(x_tree),
##     control = rpart.control(cp = 0))
##
## Variables actually used in tree construction:
##  [1] All               genreAction       genreAdventure    genrePlatform
##  [5] genreRole.Playing genreShooter      IP_TypeNot.IP     IP_TypeSmall.IP
##  [9] NS                OSX               PC                PS2
## [13] PS3               PS4               PS5
publisherNintendo
## [17] Rating            XOne              XS                year_adj
##
## Root node error: 1062/648 = 1.6389
##
## n= 648
##
##            CP nsplit rel error  xerror     xstd
## 1  0.14631328      0   1.00000 1.00493 0.052305
## 2  0.07237197      1   0.85369 0.87250 0.049656
## 3  0.03281006      2   0.78131 0.81401 0.047317
## 4  0.01564264      3   0.74850 0.79276 0.045053
## 5  0.01236856      4   0.73286 0.77532 0.045875
## 6  0.00875069      5   0.72049 0.78510 0.047129
## 7  0.00729957      6   0.71174 0.78832 0.046981
## 8  0.00703008      9   0.68984 0.78945 0.047292
## 9  0.00660590     10   0.68281 0.79062 0.048658
## 10 0.00607271     13   0.66173 0.81210 0.049977
## 11 0.00535220     16   0.64351 0.82295 0.050523
## 12 0.00519813     18   0.63281 0.82879 0.050628
## 13 0.00452917     19   0.62761 0.82188 0.050173
## 14 0.00414101     22   0.61402 0.83337 0.050443
## 15 0.00367394     23   0.60988 0.83630 0.050693
## 16 0.00349528     24   0.60621 0.84127 0.050765
## 17 0.00338884     25   0.60271 0.84686 0.050518
## 18 0.00327202     26   0.59932 0.85190 0.051112
## 19 0.00313580     28   0.59278 0.85387 0.050879
## 20 0.00285690     29   0.58964 0.85642 0.051014
## 21 0.00282322     31   0.58393 0.86065 0.051137
## 22 0.00273229     32   0.58111 0.86291 0.050266
## 23 0.00267591     34   0.57564 0.85888 0.050055
## 24 0.00263935     35   0.57297 0.85670 0.050053
## 25 0.00242814     37   0.56769 0.85857 0.050201
## 26 0.00229782     38   0.56526 0.86522 0.050378
## 27 0.00225543     39   0.56296 0.86399 0.050398
## 28 0.00201438     40   0.56071 0.86726 0.050374
## 29 0.00201363     41   0.55869 0.86810 0.050496
## 30 0.00199290     42   0.55668 0.86810 0.050496
## 31 0.00177954     43   0.55468 0.86676 0.050673
## 32 0.00158171     44   0.55290 0.87309 0.050305
## 33 0.00134837     45   0.55132 0.88265 0.050646
## 34 0.00119135     46   0.54997 0.88647 0.050925
## 35 0.00112065     48   0.54759 0.88557 0.050816
## 36 0.00095332     49   0.54647 0.88390 0.050521
## 37 0.00094098     50   0.54552 0.88459 0.050521
## 38 0.00086276     51   0.54458 0.88409 0.050514
## 39 0.00072193     53   0.54285 0.88435 0.050446
## 40 0.00000000     54   0.54213 0.88371 0.050436

best_cp <- cp_table[which.min(cp_table[, "xerror"]), "CP"]
pruned_tree <- prune(tree_model, cp = best_cp)

# Plot the pruned tree
rpart.plot(pruned_tree, main = "Pruned Decision Tree")

pdf("pruned_tree_plot.pdf", width = 8, height = 6)
rpart.plot(pruned_tree, main = "Pruned Decision Tree")
dev.off()

# Function to calculate OOS R² for a given tree model
calculate_oos_r2_tree <- function(x, y, tree_control, train_fraction = 0.7,
n_splits = 20) {
  oos_r2_values <- numeric(n_splits)

  for (i in 1:n_splits) {
    # Split data into training and testing sets
    train_indices <- sample(1:nrow(x), size = floor(train_fraction *
nrow(x)))
    test_indices <- setdiff(1:nrow(x), train_indices)
    x_train <- x[train_indices, ]
    y_train <- y[train_indices]
    x_test <- x[test_indices, ]
    y_test <- y[test_indices]

    # Fit the tree model
    tree_model <- rpart(y_train ~ ., data = data.frame(x_train), control = 
tree_control)

    # Predict on the test set
    y_pred <- predict(tree_model, newdata = data.frame(x_test))

    # Calculate OOS R²
    rss <- sum((y_test - y_pred)^2)
    tss <- sum((y_test - mean(y_test))^2)
    oos_r2 <- 1 - rss/tss

    oos_r2_values[i] <- oos_r2
  }

  return(oos_r2_values)
}

unpruned_control <- rpart.control(cp = 0)
oos_r2_unpruned <- calculate_oos_r2_tree(x_tree, game$log_activePlayers, 
unpruned_control)

pruned_control <- rpart.control(cp = best_cp)
oos_r2_pruned <- calculate_oos_r2_tree(x_tree, game$log_activePlayers, 
pruned_control)

# Combine OOS R² values into a df
model_names <- rep(c("Unpruned Tree", "Pruned Tree"), each = 20)
oos_r2_values <- c(oos_r2_unpruned, oos_r2_pruned)

results_df <- data.frame(model = model_names, OOS_R2 = oos_r2_values)

# Plot Tree OOS R²
ggplot(results_df, aes(x = model, y = OOS_R2)) +
  geom_boxplot(fill = "purple", color = "black") +
  theme_minimal() +
  labs(title = "OOS R² Comparison for Unpruned and Pruned Trees", x = 
"Model", y = "OOS R²") +
  theme(plot.title = element_text(hjust = 0.5))
```

(3)Random Forest

```
# Convert factors to numeric
x_tree <- cbind(game[, c("year_adj", "Rating")], genre_df, publisher_df, 
ip_df, console_df)
```

```
x_tree$year_adj <- scale(x_tree$year_adj)
x_tree$Rating <- scale(x_tree$Rating)
x_tree <- data.frame(lapply(x_tree, function(x) if(is.factor(x))
as.numeric(x) else x))

# Fit the Random Forest model
rf_model <- randomForest(game$log_activePlayers ~ ., data =
data.frame(x_tree), ntree = 500, importance = TRUE)
print(rf_model)

##
## Call:
##  randomForest(formula = game$log_activePlayers ~ ., data =
data.frame(x_tree),      ntree = 500, importance = TRUE)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 71
##
##           Mean of squared residuals: 1.279812
##                     % Var explained: 21.91

# Plot the importance of variables
png("variable_importance_plot.png", width = 1200, height = 800)
par(mar = c(5, 15, 4, 2) + 0.1)
varImpPlot(rf_model, main = "Variable Importance Plot", n.var = min(30,
nrow(rf_model$importance)))
dev.off()

# Function to split data and calculate OOS R² for Random Forest
calculate_oos_r2_rf <- function(x, y, train_fraction = 0.7, n_splits = 20) {
  oos_r2_values <- numeric(n_splits)

  for (i in 1:n_splits) {
    # Split data into training and testing sets
    train_indices <- sample(1:nrow(x), size = floor(train_fraction *
nrow(x)))
    test_indices <- setdiff(1:nrow(x), train_indices)

    x_train <- x[train_indices, ]
    y_train <- y[train_indices]
    x_test <- x[test_indices, ]
    y_test <- y[test_indices]

    # Fit the Random Forest model
    rf_model <- randomForest(y_train ~ ., data = data.frame(x_train), ntree =
500)

    # Predict on the test set
    y_pred <- predict(rf_model, newdata = data.frame(x_test))
    # Calculate OOS R2
    rss <- sum((y_test - y_pred)^2)
    tss <- sum((y_test - mean(y_test))^2)
    oos_r2 <- 1 - rss/tss

    oos_r2_values[i] <- oos_r2
  }

  return(oos_r2_values)
}

x <- as.matrix(x_tree)
y <- game$log_activePlayers
oos_r2_rf <- calculate_oos_r2_rf(x, y)
```

```
# Combine OOS R² values into a data frame
model_names <- rep(c("LASSO_AICc", "LASSO_CVmin", "Tree_Unpruned",
"Tree_Pruned", "RF"), each = 20)

oos_r2_values <- c(oos_r2_results$oos_r2_aicc, oos_r2_results$oos_r2_cv_min,
oos_r2_tree, oos_r2_pruned, oos_r2_rf)

results_df <- data.frame(model = model_names, OOS_R2 = oos_r2_values)

# Plot OOS R² from all models

ggplot(results_df, aes(x = model, y = OOS_R2)) +
  geom_boxplot(fill = "purple", color = "black") +
  theme_minimal() +
  labs(title = "OOS R² Comparison for LASSO, Trees, and Random Forest", x =
"Model", y = "OOS R²") +
  theme(plot.title = element_text(hjust = 0.5))
```

Section 4.C, Classification Models, P28

(1)KNN, P29-30

```
set.seed(1234)
# Selecting columns and creating model matrix
publisher_dummies <- model.matrix(~ publisher - 1, data = game)
developer_dummies <- model.matrix(~ developer - 1, data = game)
genre_dummies <- model.matrix(~ genre - 1, data = game)
developer_dummies <- model.matrix(~ developer - 1, data = game)
ip_dummies <- model.matrix(~ IP_Type - 1, data = game)
year_cont <- model.matrix(~ year_adj, data = game)
rating_cont <- model.matrix(~ Rating, data = game)

predictors_knn <- cbind(year_cont, genre_dummies, rating_cont,
publisher_dummies, developer_dummies, ip_dummies, tdm_sparse, console_sparse)
predictors_knn <- as(predictors_knn, "sparseMatrix")

set.seed(1234)
quantiles <- quantile(game$activePlayers, probs=c(0, 0.5, 1), na.rm=TRUE)
y <- cut(game$activePlayers, breaks=quantiles, include.lowest=TRUE,
labels=c(0, 1))
train <- createDataPartition(y, p = 0.5, list = FALSE)

## Compare K = 10, 20, 40
set.seed(1234)
nearest10 <- class::knn(train=predictors_knn[train,],
test=predictors_knn[-train,], cl=y[train], prob=TRUE, k=10)
nearest20 <- class::knn(train=predictors_knn[train,],
test=predictors_knn[-train,], cl=y[train], prob=TRUE, k=20)
nearest40 <- class::knn(train=predictors_knn[train,],
test=predictors_knn[-train,], cl=y[train], prob=TRUE, k=40)
data.frame(y[-train],nearest10,nearest20, nearest40)

confusion_matrix_10 <- table(y[-train], nearest10)
confusion_matrix_20 <- table(y[-train], nearest20)
confusion_matrix_40 <- table(y[-train], nearest40)
fp_rate_10 <- confusion_matrix_10[1,2] / sum(confusion_matrix_10[,2]) # False
Positive Rate
fn_rate_10 <- confusion_matrix_10[2,1] / sum(confusion_matrix_10[,1]) # False
Negative Rate
sensitivity_10 <- confusion_matrix_10[2,2] / sum(confusion_matrix_10[2,]) #
Sensitivity
specificity_10 <- confusion_matrix_10[1,1] / sum(confusion_matrix_10[1,]) #
Specificity

# Calculate performance metrics for k = 20
fp_rate_20 <- confusion_matrix_20[1,2] / sum(confusion_matrix_20[,2]) # False
Positive Rate
fn_rate_20 <- confusion_matrix_20[2,1] / sum(confusion_matrix_20[,1]) # False
Negative Rate
sensitivity_20 <- confusion_matrix_20[2,2] / sum(confusion_matrix_20[2,]) #
Sensitivity
specificity_20 <- confusion_matrix_20[1,1] / sum(confusion_matrix_20[1,]) #
Specificity

# Calculate performance metrics for k = 40
fp_rate_40 <- confusion_matrix_40[1,2] / sum(confusion_matrix_40[,2]) # False
Positive Rate
fn_rate_40 <- confusion_matrix_40[2,1] / sum(confusion_matrix_40[,1]) # False
Negative Rate
sensitivity_40 <- confusion_matrix_40[2,2] / sum(confusion_matrix_40[2,]) #
Sensitivity
specificity_40 <- confusion_matrix_40[1,1] / sum(confusion_matrix_40[1,]) #
Specificity
```

```
# Output the performance metrics
results <- data.frame(
  k = c(10, 20, 40),
  fp_rate = c(fp_rate_10, fp_rate_20, fp_rate_40),
  fn_rate = c(fn_rate_10, fn_rate_20, fn_rate_40),
  sensitivity = c(sensitivity_10, sensitivity_20, sensitivity_40),
  specificity = c(specificity_10, specificity_20, specificity_40)
)

print(results)

##    k   fp_rate   fn_rate sensitivity specificity
## 1 10 0.4318182 0.4189189   0.6172840   0.5308642
## 2 20 0.4021739 0.3714286   0.6790123   0.5432099
## 3 40 0.3757576 0.3710692   0.6358025   0.6172840

predictors_knn <- as.data.frame(as.matrix(developer_dummies))

## Plot the three KNN models
par(mfrow = c(1, 3))
# Plot for 10/20/40-nearest neighbors
plot(game[train, 'Rating'], predictors_knn[train, 2], col = y[train], cex =
0.8, pch = 18, xlab = "Rating", ylab = "log_activePlayers", main =
"10-nearest neighbors")
points(game[-train, 'Rating'], predictors_knn[-train, 2], pch = 21, col = 1,
cex = 1.25)
points(game[-train, 'Rating'], predictors_knn[-train, 2], bg = nearest10, pch
= 21, col = grey(0.9), cex = 1.25)

plot(game[train, 'Rating'], predictors_knn[train, 2], col = y[train], cex =
0.8, pch = 18, xlab = "Rating", ylab = "log_activePlayers", main =
"20-nearest neighbors")
points(game[-train, 'Rating'], predictors_knn[-train, 2], pch = 21, col = 1,
cex = 1.25)
points(game[-train, 'Rating'], predictors_knn[-train, 2], bg = nearest20, pch
= 21, col = grey(0.9), cex = 1.25)

plot(game[train, 'Rating'], predictors_knn[train, 2], col = y[train], cex =
0.8, pch = 18, xlab = "Rating", ylab = "log_activePlayers", main =
"40-nearest neighbors")
points(game[-train, 'Rating'], predictors_knn[-train, 2], pch = 21, col = 1,
cex = 1.25)
points(game[-train, 'Rating'], predictors_knn[-train, 2], bg = nearest40, pch
= 21, col = grey(0.9), cex = 1.25)

# Add Legend
legend("topright", legend = levels(y), fill = 1:2, bty = "n", cex = 0.75)
```

```
set.seed(1234)
nearest <- class::knn(train=predictors_knn[train,],
test=predictors_knn[-train,], cl=y[train], prob=TRUE, k=floor(40))  #
sqrt(1735)
attr<-attributes(nearest)
t1 <- table(y[-train], nearest)
t1

##    nearest
##      0  1
##   0 85 77
##   1 74 88

# Calculate relevant rates
t1[1,2]/sum(t1[,2]) # FALSE POSITIVE RATE:

## [1] 0.4666667

t1[2,1]/sum(t1[,1]) # FALSE NEGATIVE RATE:

## [1] 0.4654088

t1[2,2]/sum(t1[2,]) # SENSITIVITY:

## [1] 0.5432099

t1[1,1]/sum(t1[1,]) # SPECIFICITY:

## [1] 0.5246914

source("roc.R")
roc(p= attr$prob, y=y[-train], bty="n")
title("ROC Curve after KNN Analysis, K=40")


# Combine the test set indices with their probabilities
test_indices <- (1:nrow(game))[-train]
prob_data <- data.frame(index = test_indices, probability = probabilities,
nearest = nearest)

# Sort by probability to get top 5 games
top_5k <- prob_data %>%
  arrange(desc(probability)) %>%
  head(26)

top_5k

# Get the details of the top 5 games
top_5k_games <- game[top_5k$index, ]
top_5k_games <- top_5k_games[top_5k_games$activePlayers_dummy == 1, ]
# Function to get top three most common values
get_top_three <- function(x) {
  as.data.frame(sort(table(x), decreasing = TRUE)[1:10])}

# Analyze the characteristics of the top 5 games
characteristics <- top_5k_games %>%
  summarise(
    average_rating = mean(Rating),
    average_year = mean(year_adj)
  )

# Get top three genres, publishers, developers, and consoles
top_three_genres <- get_top_three(top_5k_games$genre)
top_three_publishers <- get_top_three(top_5k_games$publisher)
top_three_developers <- get_top_three(top_5k_games$developer)
top_three_consoles <- get_top_three(top_5k_games$console)

# Print results
print(characteristics)

##   average_rating average_year
## 1       3.628571     14.47619

print("Top Three Genres:")

## [1] "Top Three Genres:"

print(top_three_genres)

##                    x Freq
## 1             Action    6
## 2            Shooter    5
## 3          Adventure    3
## 4   Action-Adventure    2
## 5         Simulation    2
## 6              Party    1
## 7           Platform    1
## 8       Role-Playing    1
## 9               <NA>   NA
## 10              <NA>   NA

print("Top Three Publishers:")

## [1] "Top Three Publishers:"

print(top_three_publishers)

##                                       x Freq
## 1                       Electronic Arts    5
## 2                                  Sega    4
## 3                               Unknown    3
## 4                              Nintendo    2
## 5                 Warner Bros. Interactive    2
## 6                   Focus Home Interactive    1
## 7                      Gears for Breakfast    1
## 8                                  Level 5    1
## 9                           Telltale Games    1
## 10 Warner Bros. Interactive Entertainment    1
```

```
print("Top Three Developers:")
```

```
## [1] "Top Three Developers:"
```

```
print(top_three_developers)
```

```
##                           x Freq
## 1                  EA DICE    2
## 2            PlatinumGames    2
## 3        Rocksteady Studios    2
## 4  EA Digital Illusions CE    1
## 5            From Software    1
## 6            Gabe Cuzzillo    1
## 7       Gears for Breakfast    1
## 8                Hazelight    1
## 9               Innersloth    1
## 10       Millenium Kitchen    1
```

```
print("Top Three Consoles:")
```

```
## [1] "Top Three Consoles:"
```

```
print(top_three_consoles)
```

```
##                       x Freq
## 1                  3DS    2
## 2       PS4 XOne PC All    2
## 3          All NS WiiU    1
## 4  All PC OSX XOne PS4    1
## 5         All PS4 NS PC    1
## 6                   NS    1
## 7        NS PS4 PC XOne    1
## 8                PC NS    1
## 9           PS3 X360 PC    1
## 10     PS3 X360 PC All    1
```

(2)Multinomial Logistic Regression, P31

```
# Split data for cross-validation
set.seed(123)
predictors_knn <- cbind(year_cont, genre_dummies, rating_cont,
publisher_dummies, developer_dummies, ip_dummies, tdm_sparse, console_sparse)
predictors_knn <- as(predictors_knn, "sparseMatrix")
quantiles <- quantile(game$activePlayers, probs=c(0, 0.5, 1), na.rm=TRUE)
y <- cut(game$activePlayers, breaks=quantiles, include.lowest=TRUE,
labels=c(0, 1))
```

```
train_indices <- createDataPartition(y, p = 0.5, list = TRUE)
train_data <- predictors_knn[train_indices[[1]], ]
train_response <- y[train_indices[[1]]]
test_data <- predictors_knn[-train_indices[[1]], ]
test_response <- y[-train_indices[[1]]]

set.seed(1234)

# Fit the glmnet model
cv_fit <- cv.glmnet(train_data, train_response, family = "multinomial")
plot(cv_fit, main="Cross-Validation")

par(mfrow=c(1,2))
plot(cv_fit$glmnet)

best_coefs <- coef(cv_fit, s = "lambda.min")
coefs_matrix <- as.matrix(best_coefs)
print(coefs_matrix)

##   [,1]
## 0 <S4 class 'dgCMatrix' [package "Matrix"] with 6 slots>
## 1 <S4 class 'dgCMatrix' [package "Matrix"] with 6 slots>

# Predict and evaluate model
predictions <- predict(cv_fit, newx = test_data, s = "lambda.min", type =
"response")
predicted_classes <- apply(predictions, 1, which.max)
accuracy <- mean(predicted_classes == test_response)
print(paste("Accuracy: ", accuracy))

## [1] "Accuracy:  0.169753086419753"

# Plotting class probabilities
prob_matrix <- predict(cv_fit, newx = test_data, type = "response", s =
"lambda.min")
boxplot(prob_matrix ~ test_response, col = "orange", varwidth = TRUE,
main="Fit Plot of Test Response")
```

Section 4.D, Causal Inference - Two-Stage LASSO & Bootstrapping, P32-33

```
# Selecting columns and creating model matrix
game_console <- cbind(game[, c("year_adj", "Number.of.Reviews", "publisher",
"genre")], console_df)
game_console$year_adj <- scale(game_console$year_adj)
game_console$Number_of_Reviews <- scale(game_console$Number.of.Reviews)
game$genre <- as.factor(game$genre)
game$publisher <- as.factor(game$publisher)
game$Rating <- as.numeric(game$Rating)

# Convert to a model matrix for Lasso
```

```
x <- model.matrix(~ ., data = game_console)
d <- game$Rating # Treatment
y <- game$log_activePlayers  # Outcome

## NAIVE LASSO regression
# Naive LASSO adds "treatment" as an extra covariate without giving it any
special attention
naive <- gamlr(cbind(d,x),y)
coef(naive)["d",] # effect is AICc selected <0

## [1] 0.5007839

plot(naive, main = "LASSO plot, Naive LASSO")

coef(naive, select=which.min(AICc(naive)))

## Two stage LASSO
# First stage Lasso to predict treatment
treat <- gamlr(x, d, lambda.min.ratio=1e-4)
plot(treat, main = "LASSO plot, First Stage LASSO")  # Visualize the variable
selection

# Predict the treatment (d_hat)
dhat <- predict(treat, x, type="response")
cor(drop(dhat),d)^2

## [1] 0.3772716

plot(dhat,d,bty="n",pch=21,bg=8, main = "Relationship between d and d_hat")

# Second stage Lasso to predict the outcome
causal <- gamlr(cbind(d, dhat, x), y, free=2, lmr=1e-4)

## 'as(<dgeMatrix>, "dgCMatrix")' is deprecated.
## Use 'as(., "CsparseMatrix")' instead.
## See help("Deprecated") and help("Matrix-deprecated").

coef(causal)["d",]  # Extract the coefficient for the treatment

## [1] 0.3954586

n <- nrow(x)
gamma <- c()  # Initialize storage for bootstrap results

for(b in 1:100){
    ib <- sample(1:n, n, replace=TRUE)
    xb <- x[ib, ]
    db <- d[ib]
    yb <- y[ib]
    treatb <- gamlr(xb, db, lambda.min.ratio=1e-3)
    dhatb <- predict(treatb, xb, type="response")
    fitb <- gamlr(cbind(db, dhatb, xb), yb, free=2)
    gamma <- c(gamma, coef(fitb)["db", ])
}

summary(gamma)  # Summarize the bootstrap results

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##  0.1111  0.3687  0.4348  0.4291  0.5127  0.6520

mle <- glm(y ~ cbind(d, x))

# # get a standard error from Bootstrap
#
mean(gamma)+2*sd(gamma)

## [1] 0.6520635

mean(gamma)-2*sd(gamma)

## [1] 0.2061821

se <- summary(mle)$coef[2, 2]
se

## [1] 0.1151139

sd(gamma)

## [1] 0.1114704

# Plot Boostrap
hist(gamma, freq = FALSE, main = "Bootstrapping Result, Causal Effect of
Rating (gamma)", xlim = c(0, 0.1))

# Calculate the standard error and coefficient from your model mle

coef_estimate <- coef(mle)["cbind(d, x)d"]

# Add vertical lines
abline(v = coef_estimate, col = "orange", lwd = 2)  # Original estimate
text(coef_estimate, 0, labels = "mle Est.", pos = 3, cex = 0.8, col =
"orange")
abline(v = mean(gamma), col = "purple", lwd = 2)  # gamma mean
text(mean(gamma), 0, labels = "Boostrap Est.", pos = 3, cex = 0.8, col =
"purple")

# Confidence interval from mle
abline(v = coef_estimate + 2 * se, col = "orange", lwd = 2, lty = "dashed")
# Upper mle CI
text(coef_estimate + 2 * se, 0, labels = "mle CI", pos = 3, cex = 0.8, col =
"orange")
abline(v = coef_estimate - 2 * se, col = "orange", lwd = 2, lty = "dashed")
```

```
# Lower mle CI
text(coef_estimate - 2 * se, 0, labels = "mle CI", pos = 3, cex = 0.8, col =
"orange")

# Confidence interval from bootstrap
abline(v = quantile(gamma, 0.025), col = "purple", lwd = 2, lty = "dashed")
text(quantile(gamma, 0.025), 0, labels = "Bootstrap CI", pos = 3, cex = 0.8,
col = "purple")
abline(v = quantile(gamma, 0.975), col = "purple", lwd = 2, lty = "dashed")
text(quantile(gamma, 0.975), 0, labels = "Bootstrap CI", pos = 3, cex = 0.8,
col = "purple")
```

Section 4.B(4),Topic Model, P25-28

```
game$IP_Type <- factor(game$IP_Type, levels = c("Big IP", "Medium IP", "Small
IP", "Not IP"))

game <- game %>%
  drop_na(activePlayers, genre, IP_Type)

game$genre <- factor(game$genre)
levels_genre <- levels(game$genre)

set.seed(123)
trainIndex <- createDataPartition(game$IP_Type, p = .8,
                                  list = FALSE,
                                  times = 1)

gameTrain <- game[ trainIndex,]
gameTest  <- game[-trainIndex,]

gameTrain$genre <- factor(gameTrain$genre, levels = levels_genre)
gameTest$genre <- factor(gameTest$genre, levels = levels_genre)

gameTrain_balanced <- upSample(x = gameTrain[, c("activePlayers", "genre")],
y = gameTrain$IP_Type)

tree_model <- rpart(Class ~ activePlayers + genre, data = gameTrain_balanced,
method = "class")

summary(tree_model)

## Call:
```

```
## rpart(formula = Class ~ activePlayers + genre, data = gameTrain_balanced,
##     method = "class")
##   n= 1528
##
##            CP nsplit rel error    xerror       xstd
## 1  0.17452007      0 1.0000000 1.0488656 0.01397381
## 2  0.03228621      1 0.8254799 0.8254799 0.01656383
## 3  0.02006981      2 0.7931937 0.8324607 0.01651900
## 4  0.01832461      3 0.7731239 0.8158813 0.01662214
## 5  0.01788831      7 0.6998255 0.7914485 0.01675341
## 6  0.01657941      9 0.6640489 0.7617801 0.01688035
## 7  0.01483421     11 0.6308901 0.7347295 0.01696570
## 8  0.01308901     12 0.6160558 0.7059337 0.01702521
## 9  0.01134380     13 0.6029668 0.6701571 0.01705458
## 10 0.01000000     14 0.5916230 0.6439791 0.01704494
```

```
rpart.plot(tree_model, type = 3, extra = 101, fallen.leaves = TRUE,
           main = "Decision Tree for IP Type by Active Players and Genre",
           cex = 0.4,
           tweak = 1.2,
           box.palette = "RdBu", shadow.col = "gray", nn = TRUE)

## Warning: cex and tweak both specified, applying both
```

Section 4.B(4)a,Decision Tree for active players, P25-26

```
# Training set prediction
train_pred <- predict(tree_model, newdata = gameTrain, type = "class")

# Test set prediction
test_pred <- predict(tree_model, newdata = gameTest, type = "class")

# Calculating training set accuracy
train_accuracy <- mean(train_pred == gameTrain$IP_Type)
test_accuracy <- mean(test_pred == gameTest$IP_Type)

# Output results
print(paste("Training set accuracy:", train_accuracy))

## [1] "Training set accuracy: 0.3"

print(paste("Test set accuracy:", test_accuracy))
```

```
## [1] "Test set accuracy: 0.203125"

# Calculating training set and test set R²
train_R2 <- caret::postResample(as.numeric(train_pred),
as.numeric(gameTrain$IP_Type))["Rsquared"]
test_R2 <- caret::postResample(as.numeric(test_pred),
as.numeric(gameTest$IP_Type))["Rsquared"]

# Output results
print(paste("Training set R²:", train_R2))

## [1] "Training set R²: 0.0548582100060946"

print(paste("Test set R²:", test_R2))

## [1] "Test set R²: 0.0049286465494176"

gameTrain <- gameTrain %>% drop_na(activePlayers, genre, log_activePlayers)
gameTest <- gameTest %>% drop_na(activePlayers, genre, log_activePlayers)

# Random Forest
rf_model <- randomForest(log_activePlayers ~ activePlayers + genre, data =
gameTrain, ntree = 500)

# Predict
rf_predictions <- predict(rf_model, gameTest)

# MSE
mse <- mean((rf_predictions - gameTest$log_activePlayers)^2)
rmse <- sqrt(mse)


print(rf_model)

##
## Call:
##  randomForest(formula = log_activePlayers ~ activePlayers + genre,
data = gameTrain, ntree = 500)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 1
##
##           Mean of squared residuals: 0.04055452
##                     % Var explained: 97.53

print(paste("RMSE:", rmse))
```

```
## [1] "RMSE: 0.22932862798905"

library(tree)
game_tree <- rpart(log_activePlayers ~ Rating + genre, data=gameTrain)

# Visualization
rpart.plot(game_tree, type = 3, extra = 101, fallen.leaves = TRUE,
           main = "Decision Tree for Log Active Players by Rating and Genre",
           cex = 0.6, tweak = 1.2, box.palette = "RdBu", shadow.col = "gray",
nn = TRUE)

## Warning: cex and tweak both specified, applying both
```

Section 4.B(4)b,Random Forests for each genre, P26-28

```
par(mfrow=c(1,2))
rating_grid <- expand.grid(Rating = seq(min(game$Rating), max(game$Rating),
length = 1000),
                           genre = levels_genre)

for (g in levels_genre) {
  genre_grid <- rating_grid %>% filter(genre == g)
  train_data <- gameTrain %>% filter(genre == g)

  if (nrow(train_data) > 0) {
    plot(train_data$Rating, train_data$log_activePlayers,
         pch=21, bg=8, xlab="Rating", ylab="Log Active Players",
main=paste("Genre:", g))
    lines(genre_grid$Rating, predict(game_tree, newdata=genre_grid), col=2,
lwd=3)
  }
}
```

Section 4.E,Topic Model, P33-34

```
set.seed(123)
kmeans_result <- kmeans(dtm_matrix, centers = 4)

# top 20 words
```

```
apply(kmeans_result$centers, 1, function(cluster_center) {
  colnames(dtm_matrix)[order(-cluster_center)[1:20]]
})
```

```
##       1                          2                    3
##  [1,] "\"abandons\","            "\"acompanha\","     "\"abilitiesit\","
##  [2,] "\"actionsand\","          "\"aconteceria\","   "\"afterpirates\","
##  [3,] "\"additionbut\","         "\"adio\","          "\"aiding\","
##  [4,] "\"afterwardsbut\","       "\"algumtalvez\","   "\"announcedbut\","
##  [5,] "\"alongwhere\","          "\"alissa\","        "\"annoyingas\","
##  [6,] "\"amodest\","             "\"alto\","          "\"apartand\","
##  [7,] "\"annoyingeach\","        "\"apertar\","       "\"arthgh\","
##  [8,] "\"apologizesbut\","       "\"arch\","          "\"asyetunreleased\","
##  [9,] "\"appearancesthanks\","   "\"artistas\","      "\"badmost\","
## [10,] "\"artifactthis\","        "\"asla\","          "\"bangersmy\","
## [11,] "\"artificialand\","       "\"atiramais\","     "\"beforeit\","
## [12,] "\"assaulted\","           "\"aumentar\","      "\"beginningyou\","
## [13,] "\"awe\","                 "\"autorais\","      "\"betweenshantae\","
## [14,] "\"beamy\","               "\"baseadas\","      "\"capped\","
## [15,] "\"beatsthe\","            "\"basiimagine\","
"\"chibistyledsmooth\","
## [16,] "\"beforeonly\","          "\"bossu\","         "\"choreas\","
## [17,] "\"belongs\","             "\"botes\","         "\"cleanest\","
## [18,] "\"betrayedassaults\","    "\"busca\","         "\"consisting\","
## [19,] "\"boma\","                "\"captulo\","       "\"cursewhile\","
## [20,] "\"bordersby\","           "\"captuloscada\","  "\"cuteand\","
##       4
##  [1,] "\"really\","
##  [2,] "\"game\","
##  [3,] "\"playing\","
##  [4,] "\"one\","
##  [5,] "\"love\","
##  [6,] "c(\"\","
##  [7,] "\"ever\","
##  [8,] "\"play\","
##  [9,] "\"ive\","
## [10,] "\"time\","
## [11,] "\"combat\","
## [12,] "\"im\","
## [13,] "\"much\","
## [14,] "\"lot\","
## [15,] "\"design\","
## [16,] "\"level\","
## [17,] "\"know\","
## [18,] "\"\")"
## [19,] "\"just\","
## [20,] "\"everything\","
```

```
#  topic model analysis
dtm_simple_triplet <- as.simple_triplet_matrix(dtm)
dtm_simple_triplet <- dtm_simple_triplet[complete_rows, ]
topics_result <- topics(dtm_simple_triplet, K = 10)

summary(topics_result, n = 10)
```

```
##
## Top 10 phrases by topic-over-null term lift (and usage %):
##
## [1] '"abandons",', '"actionsand",', '"additionbut",', '"afterwardsbut",',
'"alongwhere",', '"amodest",', '"annoyingeach",', '"apologizesbut",',
'"appearancesthanks",', '"artifactthis",' (16)
## [2] '"advertisingi",', '"aggressivelynot",', '"allalso",',
'"animesometimes",', '"aroundand",', '"bandwanted",', '"boxed",',
'"canceling",', '"charmlikethe",', '"closely",' (14)
## [3] '"act")', '"adept",', '"affecting",', '"alienexcept",',
'"amazingthen",', '"anticlimacticespecially",', '"awesomelike",',
'"btwnecromorphs",', '"characterwise",', '"clearevery",' (12.8)
## [4] '"acciones",', '"acertada",', '"acomete",', '"actiony",',
'"aliengena",', '"aqu",', '"asociadas",', '"bsicamente",', '"chulosbuena",',
'"creadores",' (11)
## [5] '"accomplishment",', '"afternoon",', '"analysis",',
'"bettereverything",', '"brush",', '"centralized",', '"choseni",',
'"claustrophobic",', '"contain",', '"crampacked",' (11)
## [6] '"alongthe",', '"battleto",', '"bettergreat",', '"castthe",',
'"chargethe",', '"cocreator",', '"combatcreature",', '"designsa",',
'"dooverall",', '"expectedthe",' (8.5)
## [7] '"chore")', '"choremy",', '"acompanha",', '"aconteceria",', '"adio",',
'"algumtalvez",', '"alissa",', '"alto",', '"apertar",', '"arch",' (8.2)
## [8] '"abilitiesit",', '"afterpirates",', '"aiding",', '"announcedbut",',
'"annoyingas",', '"apartand",', '"arthgh",', '"asyetunreleased",',
'"badmost",', '"bangersmy",' (6.8)
## [9] '"absurdity",', '"areasand",', '"atmospheremight",', '"authors",',
'"charmfun",', '"come")', '"consolecouldnt",', '"cosmic",', '"dissects",',
'"gamingmechanically",' (6.5)
## [10] '"aperta",', '"basicamente",', '"cercomas",', '"disfara",',
'"drogadoin",', '"empurravameu",', '"eram",', '"esbarrava",',
'"esfaqueiadepois",', '"guardsne",' (5.3)
```

```
##
## Dispersion = 2.39
```

```
stars <- game$Rating
tpcreg <- gamlr(topics_result$omega, stars)

coef(tpcreg) * 0.1
```

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                   seg37
## intercept  0.362318805
## 1          0.008871286
## 2          .
## 3         -0.007945117
## 4          .
## 5          0.008140329
## 6         -0.007640949
## 7          .
## 8          0.009159191
## 9          0.005369313
## 10        -0.012222361
```

```
regtopics_cv <- cv.gamlr(topics_result$omega, stars, lambda.min.ratio =
10^-4)
x <- 100 * dtm_matrix / rowSums(dtm_matrix)
regwords_cv <- cv.gamlr(x, stars)
# Lasso
par(mfrow = c(1, 2))
plot(regtopics_cv)
mtext("topic regression", font = 2, line = 2)
plot(regwords_cv)
mtext("bigram regression", font = 2, line = 2)
```

```
##
## Estimating on a 648 document collection.
## Fitting the 10 topic model.
## log posterior increase: 7253.6, 648.3, 228.8, 104.5, 128.6, 60.6, 53.4,
100.8, 39.9, 31.8,
```

Section 4.Fa,Decision Tree for IPs, P36-37

```
set.seed(123)
trainIndex <- createDataPartition(game$IP_Type, p = .8,
                                  list = FALSE,
                                  times = 1)
gameTrain <- game[ trainIndex,]
gameTest  <- game[-trainIndex,]

gameTrain$genre <- factor(gameTrain$genre, levels = levels_genre)
gameTest$genre <- factor(gameTest$genre, levels = levels_genre)

gameTrain_balanced <- upSample(x = gameTrain[, c("activePlayers", "genre")],
y = gameTrain$IP_Type)

tree_model <- rpart(Class ~ activePlayers + genre, data = gameTrain_balanced,
method = "class")

summary(tree_model)
```

```
## Call:
## rpart(formula = Class ~ activePlayers + genre, data = gameTrain_balanced,
##     method = "class")
##   n= 1528
##
##            CP nsplit rel error    xerror       xstd
## 1  0.17452007      0 1.0000000 1.0488656 0.01397381
## 2  0.03228621      1 0.8254799 0.8254799 0.01656383
## 3  0.02006981      2 0.7931937 0.8324607 0.01651900
## 4  0.01832461      3 0.7731239 0.8158813 0.01662214
## 5  0.01788831      7 0.6998255 0.7914485 0.01675341
## 6  0.01657941      9 0.6640489 0.7617801 0.01688035
## 7  0.01483421     11 0.6308901 0.7347295 0.01696570
## 8  0.01308901     12 0.6160558 0.7059337 0.01702521
## 9  0.01134380     13 0.6029668 0.6701571 0.01705458
## 10 0.01000000     14 0.5916230 0.6439791 0.01704494
##
## Variable importance
## activePlayers         genre
##            56            44
```

```
rpart.plot(tree_model, type = 3, extra = 101, fallen.leaves = TRUE,
           main = "Decision Tree for IP Type by Active Players and Genre",
           cex = 0.4,
           tweak = 1.2,
           box.palette = "RdBu", shadow.col = "gray", nn = TRUE)
```

```
## Warning: cex and tweak both specified, applying both
```

Section 4.Fb,Visualization for Prediction Evaluation, P37-38

```r
library(ggplot2)
gameTrain_balanced <- upSample(x = gameTrain[, c("activePlayers", "genre")],
y = gameTrain$IP_Type)
names(gameTrain_balanced)[names(gameTrain_balanced) == "Class"] <- "IP_Type"

rf_model <- randomForest(IP_Type ~ activePlayers + genre,
data=gameTrain_balanced, ntree=500)

#  Prediction on test dataset
predictions <- predict(rf_model, gameTest)

#  confusion matrix
conf_matrix <- confusionMatrix(predictions, gameTest$IP_Type)
print(conf_matrix)
```

```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction   Big IP Medium IP Small IP Not IP
##   Big IP          2         1        5     16
##   Medium IP       2         4        3     16
##   Small IP        0         1        5     19
##   Not IP          3         2        5     44
##
## Overall Statistics
##
##                Accuracy : 0.4297
##                  95% CI : (0.3426, 0.5201)
##     No Information Rate : 0.7422
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1046
##
##  Mcnemar's Test P-Value : 5.93e-06
##
## Statistics by Class:
##
##                      Class: Big IP Class: Medium IP Class: Small IP
## Sensitivity                0.28571          0.50000         0.27778
## Specificity                0.81818          0.82500         0.81818
## Pos Pred Value             0.08333          0.16000         0.20000
## Neg Pred Value             0.95192          0.96117         0.87379
## Prevalence                0.05469          0.06250          0.14062
## Detection Rate            0.01562          0.03125          0.03906
## Detection Prevalence      0.18750          0.19531          0.19531
## Balanced Accuracy         0.55195          0.66250          0.54798
##                       Class: Not IP
## Sensitivity                0.4632
## Specificity                0.6970
## Pos Pred Value             0.8148
## Neg Pred Value             0.3108
## Prevalence                 0.7422
## Detection Rate             0.3438
## Detection Prevalence       0.4219
## Balanced Accuracy          0.5801
```

```
library(caret)
confusion <- confusionMatrix(predictions, gameTest$IP_Type)
confusion_df <- as.data.frame(confusion$table)

ggplot(data = confusion_df, aes(x = Reference, y = Prediction)) +
  geom_tile(aes(fill = Freq), color = "white") +
  scale_fill_gradient(low = "white", high = "blue") +
  labs(title = "Confusion Matrix", x = "Actual", y = "Predicted") +
  theme_minimal()

# feature importance
importance <- importance(rf_model)
importance_df <- data.frame(Feature=rownames(importance),
Importance=importance[,1])

ggplot(data=importance_df, aes(x=reorder(Feature, Importance), y=Importance))
+
  geom_bar(stat="identity", fill="skyblue") +
  coord_flip() +
  labs(title="Feature Importance", x="Feature", y="Importance") +
  theme_minimal()
```