Xbox Game Project - Portfolio

August 27, 2025

This project uses 'Enhanced_Gamepass_Games_v2' *dataset published by Rohan in Kaggle platform.*

1 Excutive summary

- This data analysis project examines the Xbox Game Pass dataset ('Enhanced_Gamepass_Games_v2' from Kaggle), comprising 455 games, to uncover factors influencing game popularity, completion rates, and ratings. The objective is to deliver actionable insights for game developers, enabling them to enhance user experiences and optimize gameplay strategies on the platform.
- Utilizing Python libraries such as Pandas, NumPy, Seaborn, Matplotlib, and Scikit-learn, the analysis began with comprehensive data preparation: inspecting structures, handling missing values (using mode for categorical and mean for numerical data), converting formats (e.g., time ranges to numerical averages), and dropping non-essential columns. Exploratory data analysis focused on genre preferences, revealing Sandbox, Compilation, and MMO as the most popular genres based on average gamers (247,504; 181,648; and 133,438 respectively). Visualization techniques, including bar charts and bubble plots, highlighted two success models: High-Engagement (long playtimes in MMO and Compilation, fostering deep immersion and community) and Mass-Appeal (short, creative experiences in Sandbox, attracting broad audiences).
- Further, K-means clustering on difficulty ratio and playtime classified games into four distinct types: Casual (easy-short), Epic (hard-very long), Challenge (very hard-short), and Mainstream (hard-long). Cross-tabulation and stacked bar charts showed Casual dominating with diverse genres like Action and Adventure, while Epic offers untapped potential with high gamer averages but limited variety. Key insights indicate that popularity stems from engagement depth or accessibility rather than completion rates. Recommendations include prioritizing multiplayer features and narrative depth for loyal communities (e.g., MMO/RPG), while emphasizing creativity and freedom for mass appeal (e.g., Sandbox). This approach can guide Xbox developers in resource allocation, genre diversification, and user-centric design, ultimately boosting retention and acquisition.
- Skills demonstrated: Data cleaning, statistical analysis, visualization, unsupervised machine learning, and insight derivation for business impact.

2 Problem statement

The goal of this project is to analyze the dataset for uncovering factors that affect the popularity, completion rate, and ratings of games. As a result, provide useful insights for game developers to optimize gaming experience.

2.1 Main section

1. Data preparation
2. Genre analysis
3. Difficulty and play time analysis
4. Conclusion

3 Data preparation

3.1 Data inspection

[1]:
```
# import essential library
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

[2]:
```
# call the dataset
game = pd.read_csv('Enhanced_Gamepass_Games_v2.csv')
```

[3]:
```
# briefly inspect structure
game.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 455 entries, 0 to 454
Data columns (total 15 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   GAME              455 non-null    object
 1   RATIO             455 non-null    object
 2   GAMERS            455 non-null    object
 3   COMP %            455 non-null    float64
 4   TIME              421 non-null    object
 5   RATING            452 non-null    float64
 6   ADDED             454 non-null    object
 7   True_Achievement  455 non-null    int64
 8   Game_Score        455 non-null    int64
 9   GENRES            453 non-null    object
 10  Main_Genre        453 non-null    object
 11  second_genre      452 non-null    object
 12  third_genre       350 non-null    object
 13  fourth_genre      5 non-null      object
```

```
 14  name_genre_match  453 non-null    object
dtypes: float64(2), int64(2), object(11)
memory usage: 53.4+ KB
```

[4]:
```
# inspect data in first rows of data set
game.head()
```

[4]:

	GAME	RATIO	GAMERS	COMP %	TIME	RATING	ADDED	True_Achievement	Game_Score	GENRES	Main_Genre	second_genre	third_genre	fourth_genre	name_genre_match
0	Mass Effect Legendary Edition	1.87	84,143	4.1	100-120 hours	4.8	06 Jan 22	5442	2915	Action, RPG, Third-Person Shooter	Action	RPG	Third-Person Shooter	NaN	Mass Effect Legendary Edition
1	The Elder Scrolls V: Skyrim Special Edition	1.97	213,257	8.0	80-100 hours	4.7	15 Dec 20	3055	1550	RPG, Action, Adventure, Open World	RPG	Action	Adventure	Open World	The Elder Scrolls V: Skyrim Special Edition
2	Mass Effect 2	1.34	221,178	9.6	50-60 hours	4.7	09 Nov 20	1819	1355	Action, RPG, Third-Person Shooter	Action	RPG	Third-Person Shooter	NaN	Mass Effect 2
3	Stardew Valley	3.04	51,530	1.0	150-200 hours	4.7	02 Dec 21	3036	1000	Simulation, RPG, Indie, Farming Sim	Simulation	RPG	Indie	Farming Sim	Stardew Valley
4	It Takes Two	1.68	71,981	15.6	12-15 hours	4.7	03 Nov 21	1678	1000	Action, Adventure, Platformer, Co-op	Action	Adventure	Platformer	Co-op	It Takes Two

3.2 Data manipulation

Through data structure and its corresponding values, there are variables that need to be adjusted to correct formats.

Deal with categorical data

[5]:
```
# change data type to categorical data
game_new = game.copy() # create a copy of original data

game_new[['GAME','Main_Genre','second_genre','third_genre','fourth_genre','name_genre_match']]
 ↪= game_new[
   ␣
 ↪['GAME','Main_Genre','second_genre','third_genre','fourth_genre','name_genre_match']].
 ↪astype('category')

game_new[['GAME','Main_Genre','second_genre','third_genre','fourth_genre','name_genre_match']]
 ↪dtypes # check again
```

[5]:
```
GAME                category
Main_Genre          category
second_genre        category
third_genre         category
fourth_genre        category
name_genre_match    category
dtype: object
```

[6]:
```
# since the ratio column realize '-' in values, I inspect and remove it before␣
 ↪changing data type
game_new.groupby('RATIO')['RATIO'].agg('count').head()
```

[6]:
```
RATIO
-       3
1.04    1
1.06    1
1.1     1
1.13    1
Name: RATIO, dtype: int64
```

[7]:
```
# RATIO colum
game_new['RATIO'] = game_new['RATIO'].replace('-',np.nan).astype('float64')

# GAMERS column
game_new['GAMERS'] = game_new['GAMERS'].str.replace(',','').astype('int64')

# check result
game_new[['GAMERS','RATIO']].dtypes
```

[7]:
```
GAMERS      int64
RATIO     float64
dtype: object
```

Deal with Time data (from categorical to number)

[8]:
```
# inspect the values in time column
game_new.groupby('TIME')['TIME'].value_counts(dropna= False) # Na included
```

[8]:
```
TIME
0-0.5 hours       1
0.5-1 hour        1
1-2 hours        17
10-12 hours      12
100-120 hours     9
1000+ hours       5
12-15 hours      15
120-150 hours     3
15-20 hours      31
150-200 hours    19
2-3 hours        11
20-25 hours      36
200-300 hours     7
25-30 hours      30
3-4 hours        11
30-35 hours      10
300-500 hours     3
35-40 hours       4
4-5 hours         7
40-50 hours      39
5-6 hours         5
50-60 hours      24
500-750 hours     1
6-8 hours        20
60-80 hours      42
8-10 hours       27
80-100 hours     31
Name: count, dtype: int64
```

For easier for analysis later, I added a column that take the mean of time frames in time column, except the "1000+ hours" data, which will be maintained as 1000.

[9]:
```
# replace the words "hours" in new column
game_new['time_num'] = game_new['TIME'].str.replace(r' hours|hour','',␣
 ↪regex=True)

# define a function
def time_convert(value):
    value = str(value)
    if "+" in value:
       x = value.replace("+","") # remove "+" for "1000+" cases
       return np.int64(x) # change data type
    else:
        x = value.split("-") # from "40-60" to ["40","60"]
```

```
        num = list(map(float,x)) # map func float to each value and change to␣
 ↪list
        return np.mean(num) # calculate the mean

# apply function for each row in new column
game_new['time_num'] = game_new['time_num'].apply(time_convert)

# compare 2 columns (categorical vs numeric data)
print(game_new[['TIME','time_num']])
```

```
              TIME  time_num
0    100-120 hours     110.0
1     80-100 hours      90.0
2      50-60 hours      55.0
3    150-200 hours     175.0
4      12-15 hours      13.5
..              …       …
450    15-20 hours      17.5
451            NaN       NaN
452            NaN       NaN
453            NaN       NaN
454            NaN       NaN

[455 rows x 2 columns]
```

Deal with Na values

[10]:
```
# inspect Na in each column
x = game_new.isnull().sum().sort_values(ascending=False)

print(x)
```

```
fourth_genre        450
third_genre         105
TIME                 34
time_num             34
RATIO                 3
RATING                3
second_genre          3
GENRES                2
Main_Genre            2
name_genre_match      2
ADDED                 1
GAME                  0
GAMERS                0
COMP %                0
True_Achievement      0
Game_Score            0
dtype: int64
```

Even though fourth and third genre column have the most na values among other variables, I did not process those variables, instead, I chose to eliminate these colomns since it's not highly essential for the analysis target.

[11]:
```
# remove fourth and third genre column
game_new = game_new.drop(['fourth_genre','third_genre'], axis=1) # eliminate␣
 ↪assigned coloumns
```

Handle Na values

- With categorical data such as `TIME`, `second_genre`, `GENRES`, `Main_Genre`, `name_genre_match`, and `ADDED` (Time data), I used *mode* method to fill those Na values
- With continuos data such as `RATIO`, `RATING`, and `time_num`, I used *mean* method to fill those values.

[12]:
```
# fill Na values of categorical and text data
cate_cols = ['TIME','second_genre','GENRES','Main_Genre',␣
 ↪'name_genre_match','ADDED']

# create for loop for automatically repeat process
for col in cate_cols:
    mode_val = game_new[col].mode()[0]
    game_new[col] = game_new[col].fillna(mode_val)
```

[13]:
```
# fill Na values of numeric data
num_cols = ['RATIO','RATING','time_num']

# create for loop for automatically repeat process
for col in num_cols:
    mean_val = np.mean(game_new[col])
    game_new[col] = game_new[col].fillna(mean_val)
```

[14]:
```
# check Na again
x = game_new.isnull().sum().sort_values(ascending=False)

print(x)
```

```
GAME                0
RATIO               0
GAMERS              0
COMP %              0
TIME                0
RATING              0
ADDED               0
True_Achievement    0
Game_Score          0
GENRES              0
Main_Genre          0
second_genre        0
```

```
name_genre_match    0
time_num            0
dtype: int64
```

4 Genre analysis

In this section, I want to determine what are genres that gamers prefer most compared to the others. From that, finding features of those popular games to have a transparent model about a successful game, which enhances developing strategies for game developers in Xbox pass product.

[15]:
```
game_new.info() # recall the structure and columns
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 455 entries, 0 to 454
Data columns (total 14 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   GAME              455 non-null    category
 1   RATIO             455 non-null    float64
 2   GAMERS            455 non-null    int64
 3   COMP %            455 non-null    float64
 4   TIME              455 non-null    object
 5   RATING            455 non-null    float64
 6   ADDED             455 non-null    object
 7   True_Achievement  455 non-null    int64
 8   Game_Score        455 non-null    int64
 9   GENRES            455 non-null    object
 10  Main_Genre        455 non-null    category
 11  second_genre      455 non-null    category
 12  name_genre_match  455 non-null    category
 13  time_num          455 non-null    float64
dtypes: category(4), float64(4), int64(3), object(3)
memory usage: 80.9+ KB
```

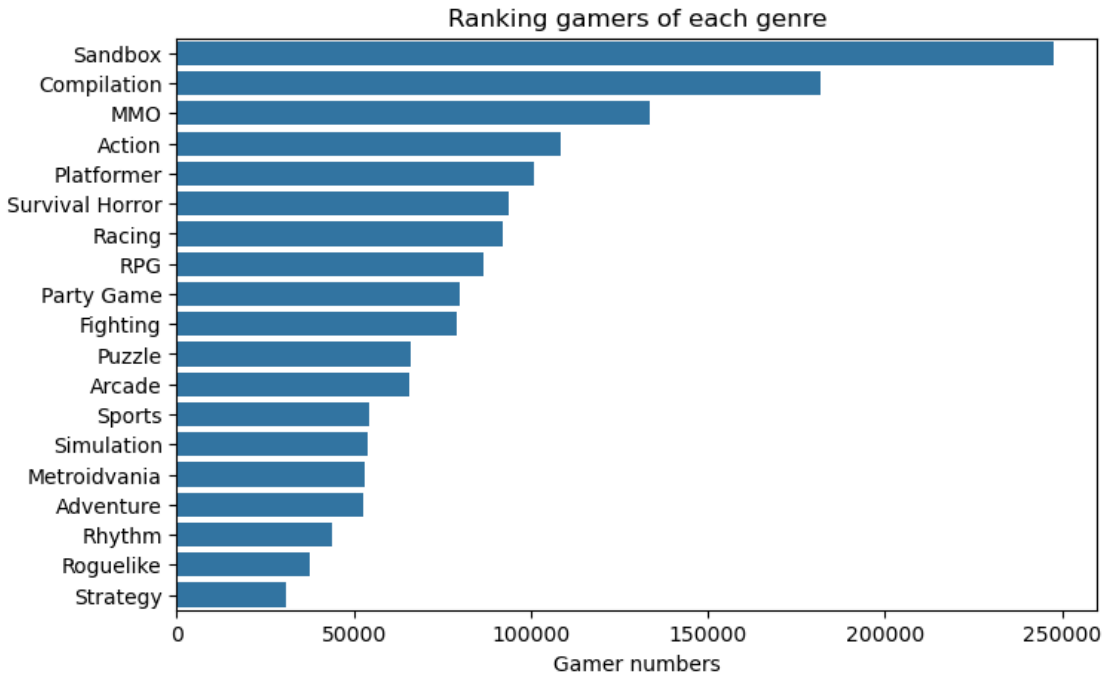
[16]:
```
genre_ana = game_new.groupby('Main_Genre').agg(
    gamer = pd.NamedAgg(column = "GAMERS", aggfunc= "mean"),
    rating = pd.NamedAgg(column = "RATING", aggfunc= "mean"),
    comp_ratio = pd.NamedAgg(column = "COMP %", aggfunc= "mean"),
    play_time = pd.NamedAgg(column= "time_num", aggfunc = "mean")
).round(2).sort_values(by='gamer', ascending=False)
genre_ana
```

```
/var/folders/ds/hq8bdwwx1b115psfdvh2l_yh0000gn/T/ipykernel_10523/2948055656.py:1
: FutureWarning: The default of observed=False is deprecated and will be changed
to True in a future version of pandas. Pass observed=False to retain current
behavior or observed=True to adopt the future default and silence this warning.
  genre_ana = game_new.groupby('Main_Genre').agg(
```

[16]:

Main_Genre	gamer	rating	comp_ratio	play_time
Sandbox	247504.50	4.10	1.10	61.25
Compilation	181648.00	4.20	0.20	250.00
MMO	133438.00	3.65	0.45	812.50
Action	108350.51	3.74	4.38	60.93
Platformer	100755.00	4.17	7.98	16.00
Survival Horror	93950.64	4.16	4.75	35.62
Racing	91995.09	3.69	2.67	54.46
RPG	86764.84	3.89	4.23	66.16
Party Game	79755.00	3.70	2.10	4.50
Fighting	79038.62	3.76	2.84	58.96
Puzzle	66289.00	3.43	3.73	67.70
Arcade	65801.50	3.05	10.95	10.00
Sports	54538.18	3.37	1.59	55.53
Simulation	54124.59	3.46	6.41	104.37
Metroidvania	52927.00	4.50	1.00	70.00
Adventure	52584.51	3.62	18.36	30.11
Rhythm	43721.00	3.90	1.10	55.00
Roguelike	37695.12	3.92	0.85	73.75
Strategy	30667.16	3.63	1.06	63.96

[17]:
```
# visualize the number of gamers of each genre
plt.figure(figsize=(8,5))
f1 = sns.barplot(genre_ana, x="gamer", y="Main_Genre", orient="y", order =␣
 ↪genre_ana.index)
f1.set_title("Ranking gamers of each genre")
f1.set_ylabel("")
f1.set_xlabel("Gamer numbers")
plt.show()
```



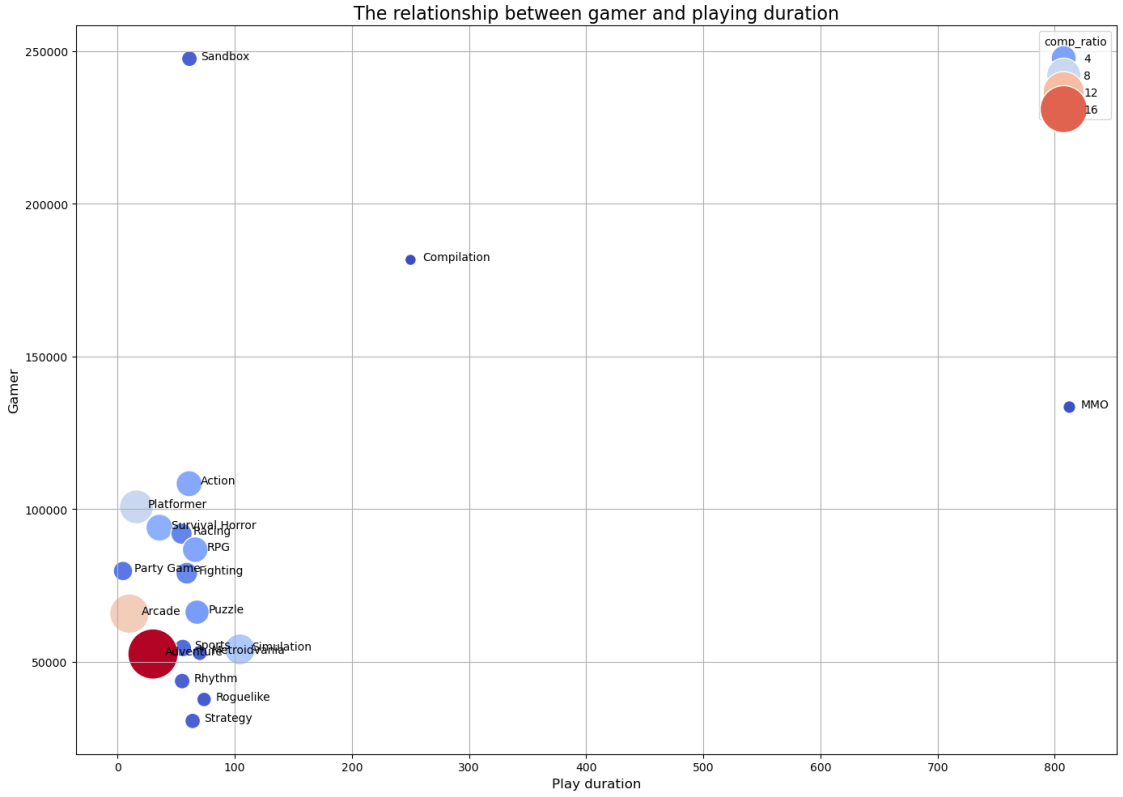
4.0.1 Initial analysis

Overall, it can be seen that Sandbox, Compilation and MMO are 3 genres played most by gamers, while Rhythm, Roguelike, and Strategy are least chosen. Specially, when comparing rating of the top 3 games and the least game, there are insignificant difference among those games. However, the main difference lies in the completion ratio and play time. Specifically, the most genre played (sandbox) have the high completion ratio with short play time, which is similar to least games played. In contrast, the following genres (Compilation and MMO) have the low completion percentage with long play time.

Thus, I dived into these differences among top ranked genres by using bubble chart, which shows the relationship between gamers, completion ratio, and playing duration.

[18]:
```
# create a bubble chart
plt.figure(figsize=(14,10))
f2 = sns.scatterplot(data = genre_ana,
                     y = "gamer",
                     x = "play_time",
                     size = "comp_ratio",
                     hue="comp_ratio",
                     sizes=(100,2000),
                     palette="coolwarm")

# set the legend for each genre in chart
for genre in genre_ana.index:
```

```
    plt.text(x = genre_ana.loc[genre,'play_time'] + 10,
             y = genre_ana.loc[genre, 'gamer'] - 100,
             s = genre,
             fontdict= dict(color="black", size=10))

# set title and legend
plt.title("The relationship between gamer and playing duration", fontsize=16)
plt.xlabel("Play duration", fontsize = 12)
plt.ylabel("Gamer", fontsize = 12)

plt.grid(True)
plt.tight_layout()
plt.show()
```



4.0.2 Insight

Our analysis reveals that popular game genres on Xbox Game Pass achieve success through two distinct models: **building a high-engagement community** and **attracting a mass-appeal audience.**

- The High-Engagement Model (MMO & Compilation): These genres have large number of players with long gaming duration. In other words, with these genres, gamers tend to invest more time in experiencing and exploring games. Moreover, with MMO, a genre whose game

focuses on online multiple player, it also strengthens to the fact that people love player connection experience, which makes them spend more time in gaming. Meanwhile, Compilation genre concentrates on experiencing a game with extended versions or stories, which arouses the curiorsity of gamers to make them spend more time in games.

- The Mass-Appeal Model (Sandbox): This genre appriciates the creativity of players instead of building complex stories inside. In contrast to long average playtime as MMO and Compilation, Sandbox though has a short playing duration, it still obtains advocates from gamers thanks for its emphasize on freedom, creativity, and accessible gameplay. This suggests that games prioritizing freedom and creativity are highly effective in appealing to a wider, more general audience compared to other gameplay styles.

Conclusion & Recommendation: Notably, **completion ratio** is not a primary driver of popularity for these top-tier genres. Success is determined by the mode of engagement—either deep retention or broad appeal. Consequently, Xbox game developers should focus on crafting innovative gameplay experiences and investing in multiplayer enhancements. This dual approach can help sustain a loyal gaming community while also attracting new players with fresh game offerings.

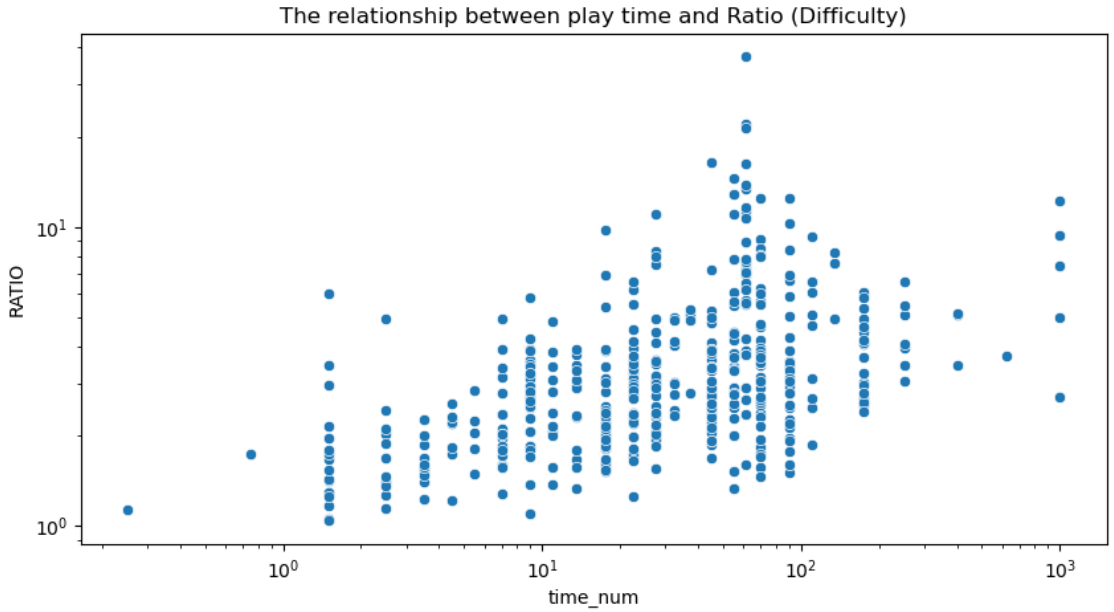
5 Difficulty and Playtime Analysis

In this part, I want to focus on the gaming experience of gamers, particularly in the achievement difficulty of genres compared to true score from players. Therefore, I used `RATIO` and `time_num` columns to inspect this problem.

[19]:
```
# recall the data information
game_new.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 455 entries, 0 to 454
Data columns (total 14 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   GAME              455 non-null    category
 1   RATIO             455 non-null    float64
 2   GAMERS            455 non-null    int64
 3   COMP %            455 non-null    float64
 4   TIME              455 non-null    object
 5   RATING            455 non-null    float64
 6   ADDED             455 non-null    object
 7   True_Achievement  455 non-null    int64
 8   Game_Score        455 non-null    int64
 9   GENRES            455 non-null    object
 10  Main_Genre        455 non-null    category
 11  second_genre      455 non-null    category
 12  name_genre_match  455 non-null    category
 13  time_num          455 non-null    float64
dtypes: category(4), float64(4), int64(3), object(3)
memory usage: 80.9+ KB
```

[20]:
```
plt.figure(figsize=(10,5))
sns.scatterplot(data = game_new, y="RATIO", x="time_num")
plt.title("The relationship between play time and Ratio (Difficulty)")
plt.xscale('log')
plt.yscale('log')
plt.show()
```



Brief decription

In the code above, since values are distributed mostly around low play time and ratio, with a few of long play time and high ratio data. Thus, I transfered two variables to logarit to expand data in low range and narrow the data in high range.

5.0.1 Apply K-means clusterring for better classification

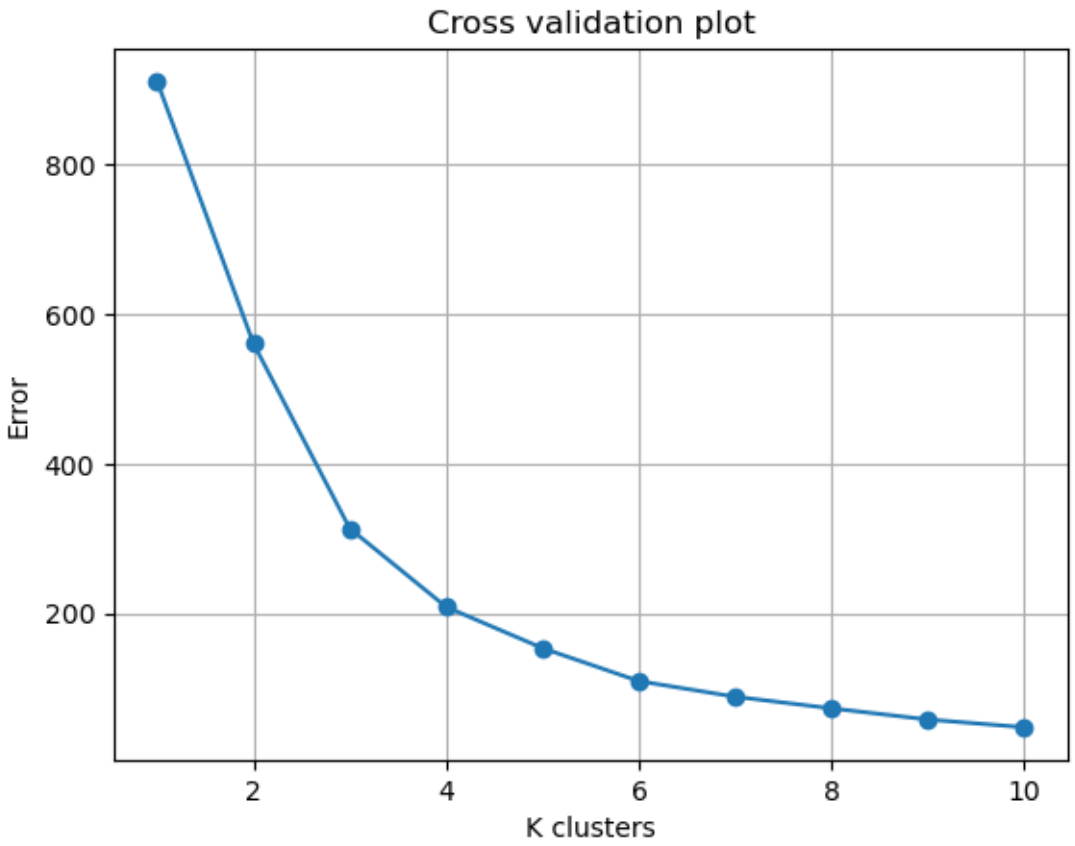
From the scatter plot, it can be seen that games may be divided into clusters with different features, instead of having a linear relationship. However, common observation is objective, which is consequently essential to have an unsupervised machine learning, k-means clustering. The final target is to determine whether the model can classify these clusters as innitally predicted or not.

[21]:
```
# import model
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
```

[22]:
```
# extract and standardize variables for model
km_va = game_new[['RATIO','time_num']]

scaler = StandardScaler()
```

```
km_scaled = pd.DataFrame(
    scaler.fit_transform(km_va), # data scaled (numpy array)
    columns = km_va.columns
)
```

[23]:
```
# Run k-means model with cross validation
inertia = []
K = range(1,11)

for k in K:
    k_mdl = KMeans(n_clusters=k, n_init=20, max_iter= 100, random_state= 42)
    k_mdl.fit(km_scaled)
    inertia.append(k_mdl.inertia_)

# Cross validation plot
plt.plot(K,inertia, marker="o")
plt.xlabel("K clusters")
plt.ylabel("Error")
plt.title("Cross validation plot")
plt.grid(True)
plt.show()
```



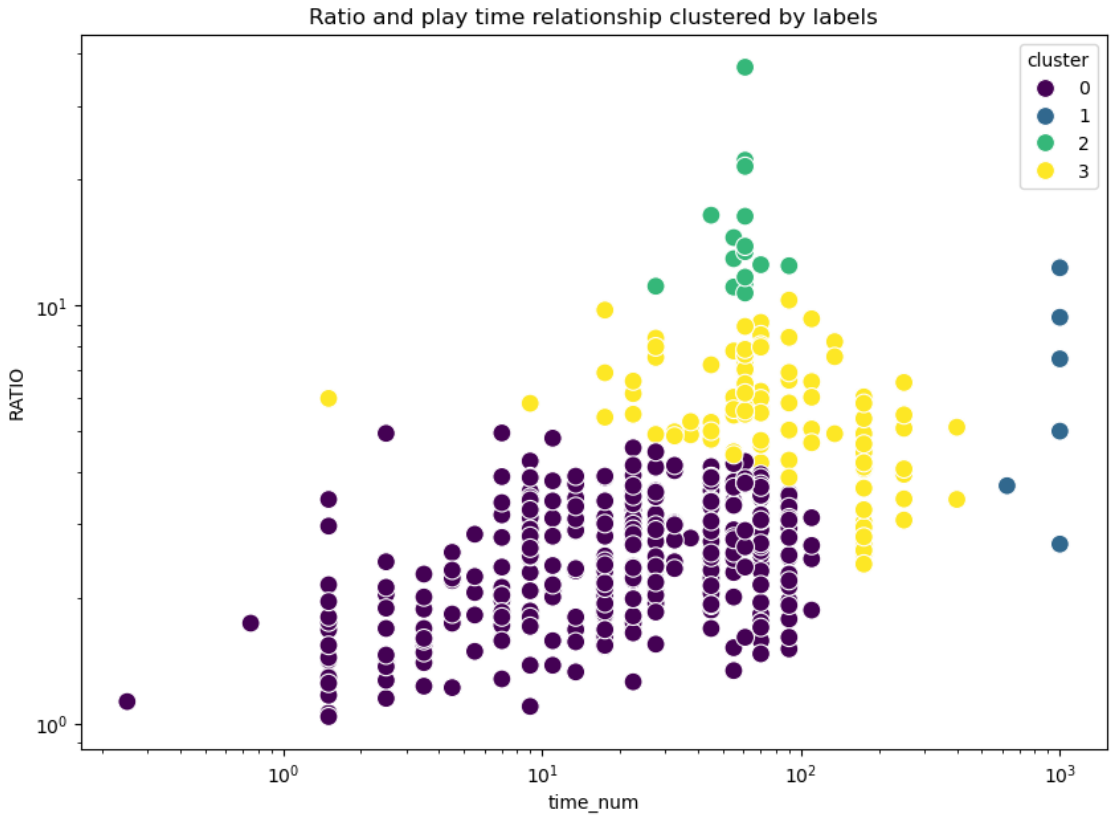
Optimal km mean cluster

From the plot above, k cluster at point 4 shows the kink of error. Following the "elbow method", I choose 4 as an optimal k cluster for the model.

[24]:
```
# run again the model
km_mdl = KMeans(n_clusters = 4, n_init=20, max_iter=100, random_state=42)
mdl_fit = km_mdl.fit(km_scaled)
```

[25]:
```
# add a column containing labels of cluster
km_va['cluster'] = mdl_fit.labels_

# visualize scatter plot using logarit, and coloring data point by cluster label
plt.figure(figsize= (10,7))
sns.scatterplot(data = km_va,
                y= 'RATIO',
                x= 'time_num',
                hue='cluster',
                palette='viridis', s=100)
plt.xscale("log")
plt.yscale("log")
plt.title("Ratio and play time relationship clustered by labels")
plt.show()
```

```
/var/folders/ds/hq8bdwwx1b115psfdvh2l_yh0000gn/T/ipykernel_10523/3879517042.py:2
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  km_va['cluster'] = mdl_fit.labels_
```



[26]:
```
# create the mean of RATIO and time_num by clusters
summary_km = km_va.groupby('cluster').agg(RATIO = pd.NamedAgg(column="RATIO",␣
↪aggfunc= "mean"),
                                time_num = pd.
↪NamedAgg(column="time_num",aggfunc="mean")).round(2)
summary_km
```

[26]:

cluster	RATIO	time_num
0	2.56	31.29
1	6.76	937.50
2	15.60	59.08
3	5.69	107.01

5.0.2 Analysis

Based on the scatter plot colored by clusters and summary of RATIO and play time mean by clusters, the data points are divided into 4 clusters with clear features:

- Cluster 0 (Easy - short): Casual games, conveniently entertaining for common players.
- Cluster 1 (Hard - very long): Triple A games, require players to invest in playing time for exploring sufficient games.

- Cluster 2 (Very Hard - short): Intelligence concentration, which is suitable for players who does not have much time.
- Cluster 3 (Hard - Long): Long-term games, which make gamers spend long time to play for understanding gameplays.

5.1 Extended problem

From the genre analysis and this result, I have a question that even though players in Xbox game pass prefer high-engagement and mass appeal gameplays, what genres do gamers prefer to play within these 4 features?

[27]:
```
# join the cluster into the game_new df
joined_df = game_new.join(km_va['cluster'])

# add name for clusters
cluster_name = {
    0: 'Casual (easy-short)',
    1: 'Epic (hard-very long)',
    2: 'Challenge (very hard-short)',
    3: 'Mainstream (hard-long)'
}

# map the name with corresponding cluster in df
joined_df['Cluster name'] = joined_df['cluster'].map(cluster_name)
```

[28]:
```
# group gamers count by each game type
gamer_vs_type = (
joined_df
    .groupby(['Cluster name','Main_Genre'])
    .agg(total_gamer = pd.NamedAgg(column = 'GAMERS', aggfunc='mean'))
    .unstack(level='Main_Genre')
    .fillna(0)
    .round(0)
# unstack second index into a new column
# instead of NaN, 0 will be filled for cases of Main Genre have 0 players
)
gamer_vs_type
```

```
/var/folders/ds/hq8bdwwx1b115psfdvh2l_yh0000gn/T/ipykernel_10523/1808913657.py:4
: FutureWarning: The default of observed=False is deprecated and will be changed
to True in a future version of pandas. Pass observed=False to retain current
behavior or observed=True to adopt the future default and silence this warning.
  .groupby(['Cluster name','Main_Genre'])
```

[28]:

	total_gamer				\
Main_Genre	Action	Adventure	Arcade	Compilation	
Cluster name					
Casual (easy-short)	106602.0	49776.0	65802.0	0.0	

```
Challenge (very hard-short)      7575.0   50156.0      0.0         0.0
Epic (hard-very long)          345603.0  344597.0      0.0         0.0
Mainstream (hard-long)         110627.0   29242.0      0.0    181648.0

                                                                      \
Main_Genre                   Fighting       MMO Metroidvania Party Game
Cluster name
Casual (easy-short)           75022.0       0.0          0.0    79755.0
Challenge (very hard-short)       0.0       0.0          0.0        0.0
Epic (hard-very long)             0.0  133438.0          0.0        0.0
Mainstream (hard-long)        85733.0       0.0      52927.0        0.0

                                                                          \
Main_Genre                   Platformer   Puzzle       RPG   Racing   Rhythm
Cluster name
Casual (easy-short)            100755.0  77024.0  108377.0  98806.0      0.0
Challenge (very hard-short)         0.0      0.0       0.0  89907.0      0.0
Epic (hard-very long)               0.0      0.0       0.0      0.0      0.0
Mainstream (hard-long)              0.0  50186.0   19965.0  61867.0  43721.0

                                                                          \
Main_Genre                   Roguelike   Sandbox Simulation   Sports Strategy
Cluster name
Casual (easy-short)            16668.0  247504.0    49042.0  49182.0  42522.0
Challenge (very hard-short)    27710.0       0.0    29942.0  84000.0  11810.0
Epic (hard-very long)              0.0       0.0    91677.0      0.0      0.0
Mainstream (hard-long)         48103.0       0.0    63292.0  60697.0  29723.0


Main_Genre                   Survival Horror
Cluster name
Casual (easy-short)                 99812.0
Challenge (very hard-short)             0.0
Epic (hard-very long)                   0.0
Mainstream (hard-long)              35332.0
```

[29]:
```
# create a summary table of genre options across 4 game types
crosstab = pd.crosstab(joined_df['Cluster name'],joined_df['Main_Genre'])
crosstab
```

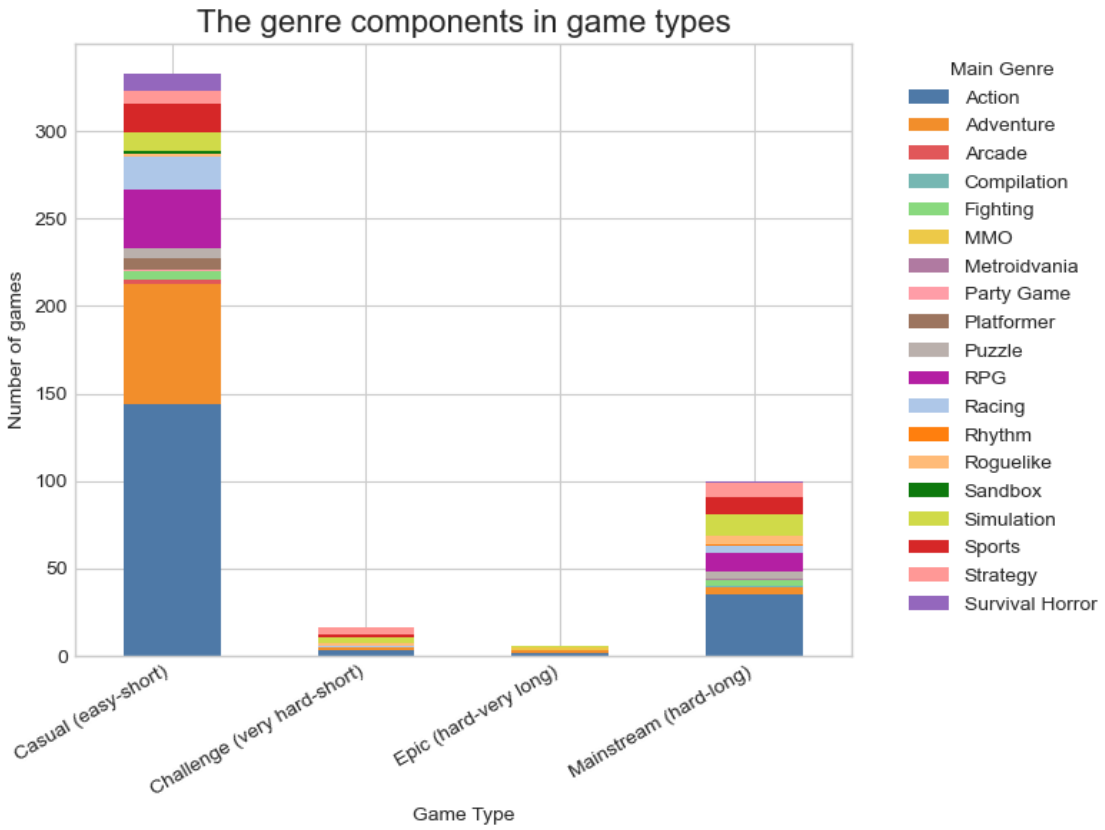
[29]:
```
Main_Genre                    Action  Adventure  Arcade  Compilation  Fighting  \
Cluster name
Casual (easy-short)              144         69       2            0         5
Challenge (very hard-short)        3          2       0            0         0
Epic (hard-very long)              2          1       0            0         0
Mainstream (hard-long)            35          4       0            1         3
```

```
Main_Genre                  MMO  Metroidvania  Party Game  Platformer  \
Cluster name
Casual (easy-short)           0             0           1           6
Challenge (very hard-short)   0             0           0           0
Epic (hard-very long)         2             0           0           0
Mainstream (hard-long)        0             1           0           0

Main_Genre                  Puzzle  RPG  Racing  Rhythm  Roguelike  Sandbox  \
Cluster name
Casual (easy-short)              6   34      18       0          2        2
Challenge (very hard-short)      0    0       1       0          1        0
Epic (hard-very long)            0    0       0       0          0        0
Mainstream (hard-long)           4   11       4       1          5        0

Main_Genre                  Simulation  Sports  Strategy  Survival Horror
Cluster name
Casual (easy-short)                 10      17         7                10
Challenge (very hard-short)          4       1         4                 0
Epic (hard-very long)                1       0         0                 0
Mainstream (hard-long)              12      10         8                 1
```

[30]:
```
# create 2 sets of color palette
palette1 = [
    "#4E79A7", "#F28E2B", "#E15759", "#76B7B2", "#89D97E",
    "#EDC948", "#B07AA1", "#FF9DA7", "#9C755F", "#BAB0AC",
    "#B41FA3", "#AEC7E8", "#FF7F0E", "#FFBB78", "#0D790D",
    "#D0DA49", "#D62728", "#FF9896", "#9467BD"
]

tol21 = [
    "#4477AA", "#2A4CAA", "#228833", "#D2BD1C", "#AA66EE",
    "#D0137E", "#BBBBBB", "#000000", "#0298B2", "#198B3D",
    "#99DDFF", "#44BB99", "#BBCC33", "#15F062", "#EE8866",
    "#FFAABB", "#EE3336", "#C191AE", "#AA4488"
]
```

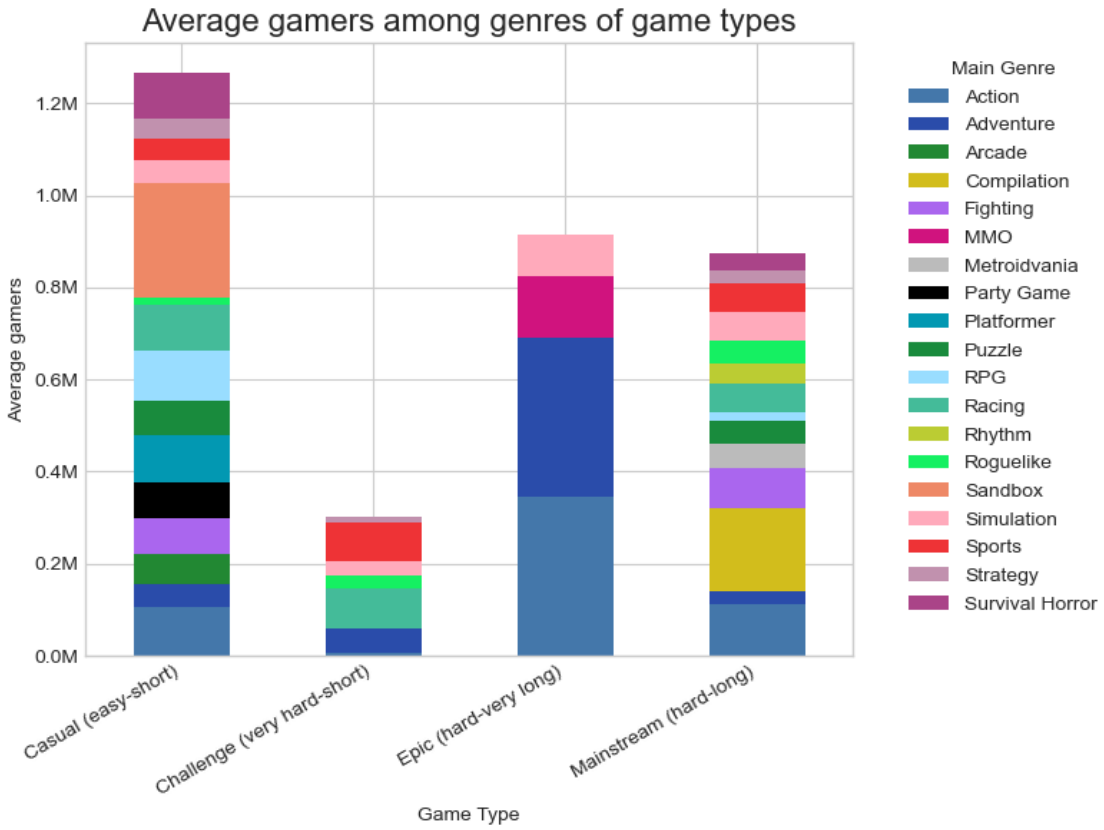
[31]:
```
# from cross tab, create a stacked bar
plt.style.use('seaborn-v0_8-whitegrid')
ax1 = crosstab.plot(kind = "bar", stacked = True,
              figsize=(8,6),
              color= palette1)

# setting labels and layout
plt.title("The genre components in game types", fontsize= 16)
plt.xticks(rotation =30, ha = 'right')
plt.xlabel("Game Type")
plt.ylabel("Number of games")
```

```
plt.legend(title = "Main Genre", bbox_to_anchor=(1.05,1), loc = 'upper left')
plt.tight_layout()
plt.show()
```



[32]:
```
from matplotlib.ticker import FuncFormatter

plt.style.use('seaborn-v0_8-whitegrid')
ax2 = gamer_vs_type.plot(kind = "bar", stacked = True,
              figsize=(8,6),
              color= tol21)

# change unit of gamers number in million unit
def mil_format(x, pos):
    return f'{x / 1_000_000:.1f}M'
ax2.yaxis.set_major_formatter(FuncFormatter(mil_format))

# setting labels and layout
plt.title("Average gamers among genres of game types", fontsize= 16)
plt.xticks(rotation =30, ha = 'right')
plt.xlabel("Game Type")
```

```
plt.ylabel("Average gamers")
plt.legend(title = "Main Genre",
           labels = gamer_vs_type.columns.get_level_values(1), # only take the␣
 ↪second index in legend
           bbox_to_anchor=(1.05,1), # move legend to the right side of chart
           loc = 'upper left')
plt.tight_layout()
plt.show()
```



5.2 Insight

Overall, the two charts show that Casual, Epic, and Mainstream are the three segments that players tend to prefer the most. Among them, the Casual segment stands out with the highest average number of gamers across genres. Notably, despite having fewer game genres, Epic records a remarkably large average user base, which even surpasses that of Mainstream. As a result, Epic represents a niche segment that Xbox Game Pass has not sufficiently exploited, as it attracts a large number of average gamers but still offers limited genre diversity.

Furthermore, although the majority of Casual games fall into the Adventure and Arcade genres, these are not the preferred genres among average gamers, as they feature lower difficulty and shorter playtime. Instead, Sandbox emerges as a potential genre within this segment, attracting a high

proportion of average gamers. In other words, the games favored by players have not been properly leveraged, which calls for further adjustments to avoid resource misallocation and to optimize core game genres more effectively for users.

Additionally, in the Mainstream segment, Compilation and Action attract larger average player bases than the other genres, while the segment overall is largely dominated by Action. Consequently, similar to Epic, this suggests that game development has been disproportionately focused on the Action genre, while neglecting other promising categories such as Compilation, which demonstrates a considerably higher average number of players compared to the rest.

6 Conclusion

In conclusion, the main factors leveraging the success of gamer attraction on Xbox Game Pass are mass appeal (Sandbox) and high engagement (MMO and Compilation). Furthermore, deeper analysis using machine learning has also contributed significantly by identifying four main game types, ranging from Casual (quick entertainment) to Epic (hardcore, immersive experiences). Therefore, the key elements determining the success of a game do not solely depend on its genre, but also on the combination between genre and the experience it delivers to players (difficulty and playtime). Notably, completion ratio, as revealed through analysis, has shown that it is not a vital factor for popularity.

6.0.1 Recommendation

Based on these insights, developers should adopt specific strategies aligned with their ultimate goals. For the mass market, greater attention should be directed toward freedom, creativity, and accessibility, as exemplified by Sandbox games. Meanwhile, building a loyal gamer community should focus on profound narratives, long-term content, and hardcore gameplay such as RPGs and MMOs. In terms of gaming experience, developers should also consider appropriate genre allocations within each game type to enhance diversity and better align with user preferences regarding expected difficulty and playtime.