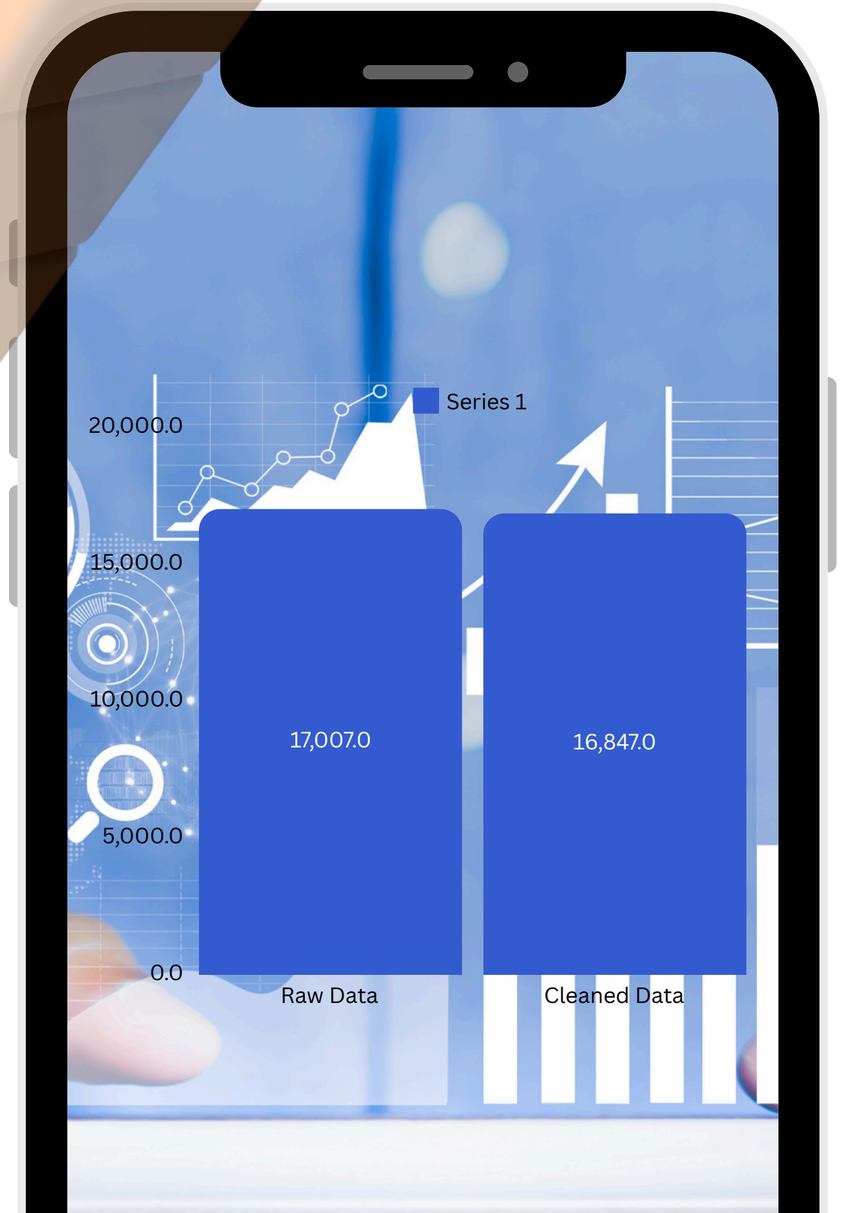
Mobile Games

Data Analysis

The dataset focuses on the mobile gaming industry's strategy game sub-market, featuring insights into popular titles like Clash of Clans, Plants vs. Zombies, and Pokémon GO. It includes data on user engagement, market trends, and game performance, offering a comprehensive view of this billion-dollar segment.



AGENDA

- Introduction
- Objective
- Data Overview
- Data Cleaning and Preparation
- Exploratory Data Analysis (EDA)
- Feature Engineering
- Outlier Detection and Handling
- Model Development
- Results and Insights
- Conclusion and Recommendations

Introduction

The mobile gaming industry has rapidly emerged as the dominant force in the global games market, surpassing $100 billion by 2021 with a decade of double-digit growth. Key factors driving this success include smartphone advancements, diverse payment models, and the portability of devices, enabling users to play anywhere. Mobile games attract millions of players daily, offering both casual and immersive experiences. Strategy games like Clash of Clans and Pokémon GO highlight a lucrative sub-market supported by innovative technologies and substantial investments. With broad demographic appeal and evolving features like AR and multiplayer options, mobile gaming continues to reshape the global entertainment landscape.

Objective

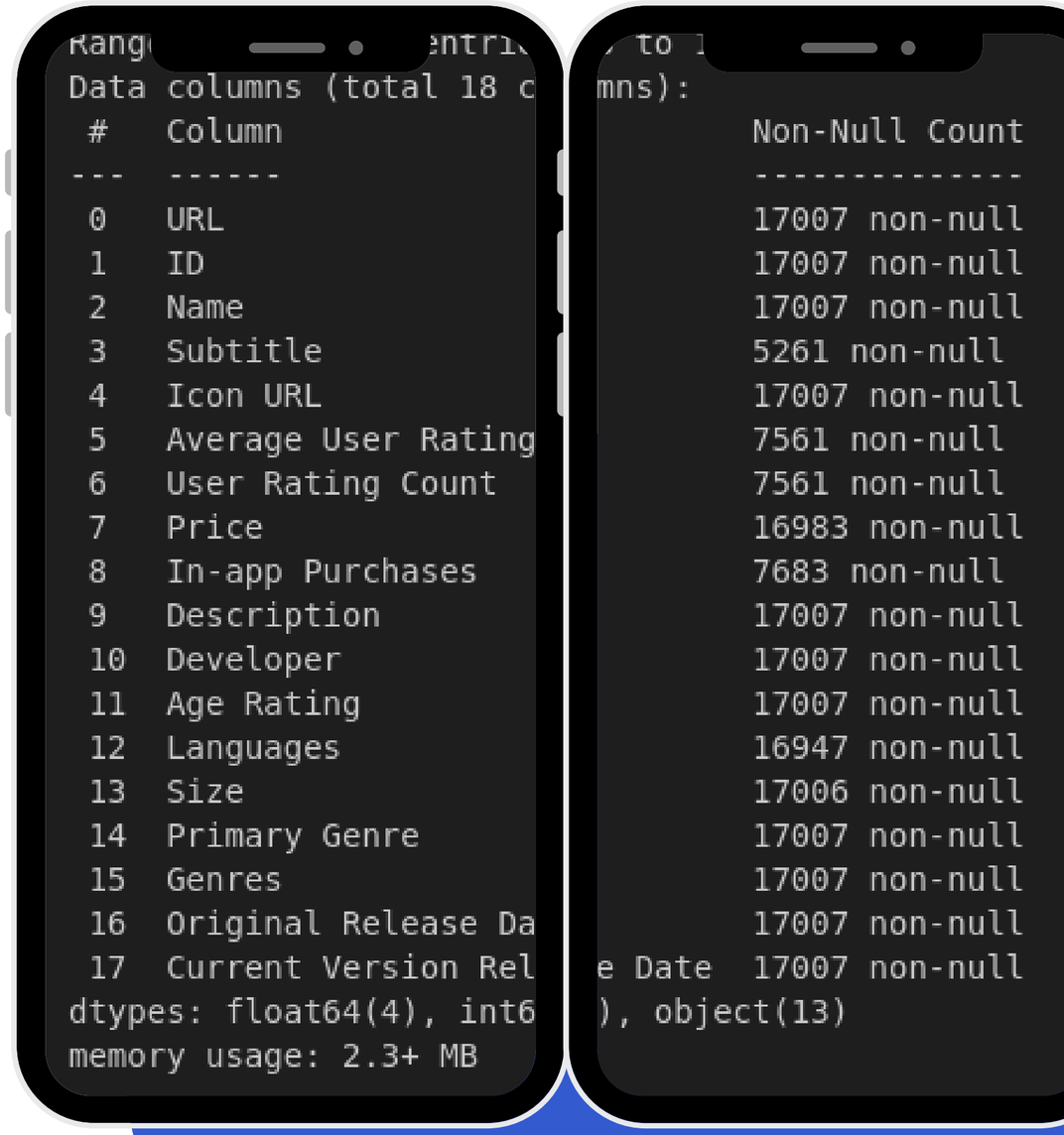
The objective of this project is to analyze a dataset containing various attributes of mobile games, including user ratings, price, genre, developer information, and release dates. The analysis involves using SQL to explore key metrics such as average user ratings, user rating counts, and pricing trends. Python-based Exploratory Data Analysis (EDA) will be employed to visualize the relationships between features like age ratings, languages, and game genres. Additionally, machine learning models will be developed to predict factors influencing user ratings and game success based on features like price, size, and genre. Missing values will be addressed, and games will be grouped and segmented by genre and age rating to uncover user preferences. The findings aim to provide actionable insights that can help mobile game developers and marketers optimize their strategies and improve user engagement.

Data Overview

The dataset contains 17,007 entries with 18 columns, providing detailed information about various mobile games. The columns include key attributes such as game ID, name, average user rating, price, developer, genres, and release dates. Out of the total entries, 16,847 are unique, with some columns containing missing values. Specifically:

- Average User Rating and User Rating Count have missing values for several entries, with only 7,561 non-null entries in each.
- Subtitle, In-app Purchases, and Languages also contain missing values, with 5,261, 7,683, and 16,947 non-null entries, respectively.
- Other columns like URL, ID, Name, Icon URL, Description, and Developer are fully populated, offering complete data for analysis.

The data types include integers, floats, and objects, with the majority of columns being categorical (object type). This dataset provides a broad view of mobile game characteristics, including ratings, pricing, genre classification, and release information, which can be leveraged for deeper analysis and modeling.



Data Cleaning and Preparation

```
app_c1 = app.copy()
app_c1 = app_c1.drop(columns=['URL','Icon URL','Description','Subtitle','In-app Purchases'])
```

```
print(f'Duplicates: {app_c1.duplicated().sum()}')
app_c1.drop_duplicates(inplace=True)
print(f'Shape after deduplication: {app_c1.shape}')
```

```
Duplicates: 160
Shape after deduplication: (16847, 13)
```

```
# Handle missing values
missing_pct = np.round((app_c1.isnull().sum()/app.shape[0])*100,2)
print(missing_pct)
```

```
ID                             0.00
Name                           0.00
Average User Rating           55.03
User Rating Count             55.03
Price                          0.14
Developer                      0.00
Age Rating                     0.00
Languages                      0.35
Size                           0.01
Primary Genre                  0.00
Genres                         0.00
Original Release Date          0.00
Current Version Release Date   0.00
dtype: float64
```

Handling Missing Values:

- Missing Average User Rating values were imputed with the median rating for each Primary Genre, ensuring that the imputation reflects the average rating within the same genre.
- Missing User Rating Count values were similarly filled with the median rating count for each Primary Genre.

```
[8]
# Fill missing 'Average_User_Rating' with median
median_ratings = df.groupby('Primary_Genre')['Average_User_Rating'].median()
df['Average_User_Rating'] = df.apply(
    lambda row: median_ratings[row['Primary_Genre']] if pd.isnull(row['Average_User_Rating']) else row['Average_User_Rating'],
    axis=1
)
```

```
[9]
# Fill missing 'User_Rating_Count' with median
median_ratings_user = df.groupby('Primary_Genre')['User_Rating_Count'].median()
df['User_Rating_Count'] = df.apply(
    lambda row: median_ratings_user[row['Primary_Genre']] if pd.isnull(row['User_Rating_Count']) else row['User_Rating_Count'],
    axis=1
)
```

```
10]
# Fill missing 'Price' with median
median_ratings_user = df.groupby('Primary_Genre')['Price'].median()
df['Price'] = df.apply(
    lambda row: median_ratings_user[row['Primary_Genre']] if pd.isnull(row['Price']) else row['Price'],
    axis=1
)
```

```
11]
# Fill missing 'Languages' with 'Unknown'
df['Languages'].fillna('Unknown', inplace=True)
```

```
12]
df['Language_Count'] = df['Languages'].str.split(', ').apply(len)
df.drop(columns=['Languages'], inplace=True)
```

```
13]
df['Size'] = df['Size'].fillna(df['Size'].median())
```

- Missing Price values were imputed using the median price within each Primary Genre, maintaining consistency within genre categories.
- Languages with missing values were replaced with 'Unknown', indicating that the language information was not available for some apps.
- Size values, with a few missing entries, were filled with the median size value, ensuring that the size data remains complete without introducing bias.

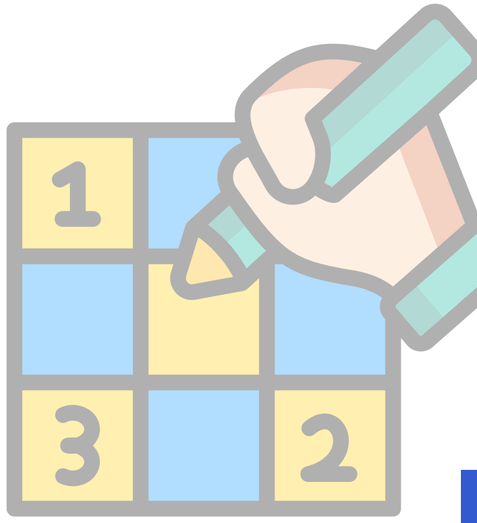
Data Cleaning and Preparation

Text Cleaning:

- Game Names: The Name column was cleaned by removing any special characters (non-alphanumeric characters), ensuring that only alphanumeric characters and spaces remain. The length of the game name was then limited to the first 60 characters to standardize the naming convention.
- Developer Names: The Developer column was similarly cleaned to remove any special characters. The length of the developer name was restricted to the first 30 characters to maintain consistency across the dataset.

```
df['Name'] = df['Name'].str.replace(r'[^A-Za-z0-9 ]+', '', regex=True)
df['Name'] = df['Name'].str[:60]
df['Developer'] =df['Developer'].str.replace(r'[^A-Za-z0-9 ]+', '', regex=True)
df['Developer'] = df['Developer'].str[:30]
```

```
# Remove "Games" and limit to the first 3 genres
df['Genres'] = df['Genres'].str.replace("Games, ", "")
df['Genres'] = df['Genres'].str.split(', ').apply(lambda x: ', '.join(x[:3]))
```



Genres Cleaning:

- The word "Games" was removed from the Genres column to ensure that only relevant genre labels remain.
- The genres were then limited to the first three entries in cases where the genre list had more than three, ensuring that only the top three genres are considered for each game.

Exploratory Data Analysis (EDA)

The EDA performed on the dataset provided valuable insights into various aspects of mobile gaming applications:

Distribution of Average User Ratings:

- The count plot visualized the distribution of Average User Ratings, highlighting the number of games falling into each rating category. This provided a sense of user satisfaction levels across the dataset.

Game Size Analysis:

- Game sizes were transformed into MB for better readability, and a Kernel Density Estimate (KDE) plot was used to examine the distribution of sizes.
- The dataset was categorized into three size groups:
- Games smaller than 250MB.
- Games between 250MB and 1GB.
- Games larger than 1GB.
- Histograms were plotted for each size group, revealing the density and distribution within these ranges.

Top Developers:

- The top 20 most common developers were identified and visualized using a horizontal bar chart. This shed light on which developers contribute the most to the dataset, providing insights into prominent players in the industry.

Age Rating Distribution:

- A pie chart was created to illustrate the distribution of games across different Age Rating categories. The visualization highlighted the percentage and count of games suitable for each age group, offering insights into target audiences.

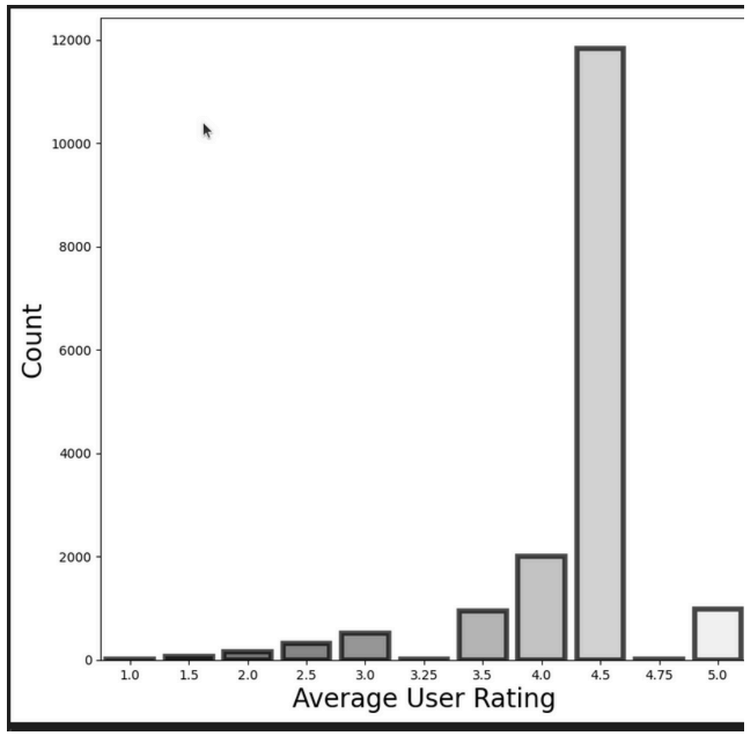
Game Genres:

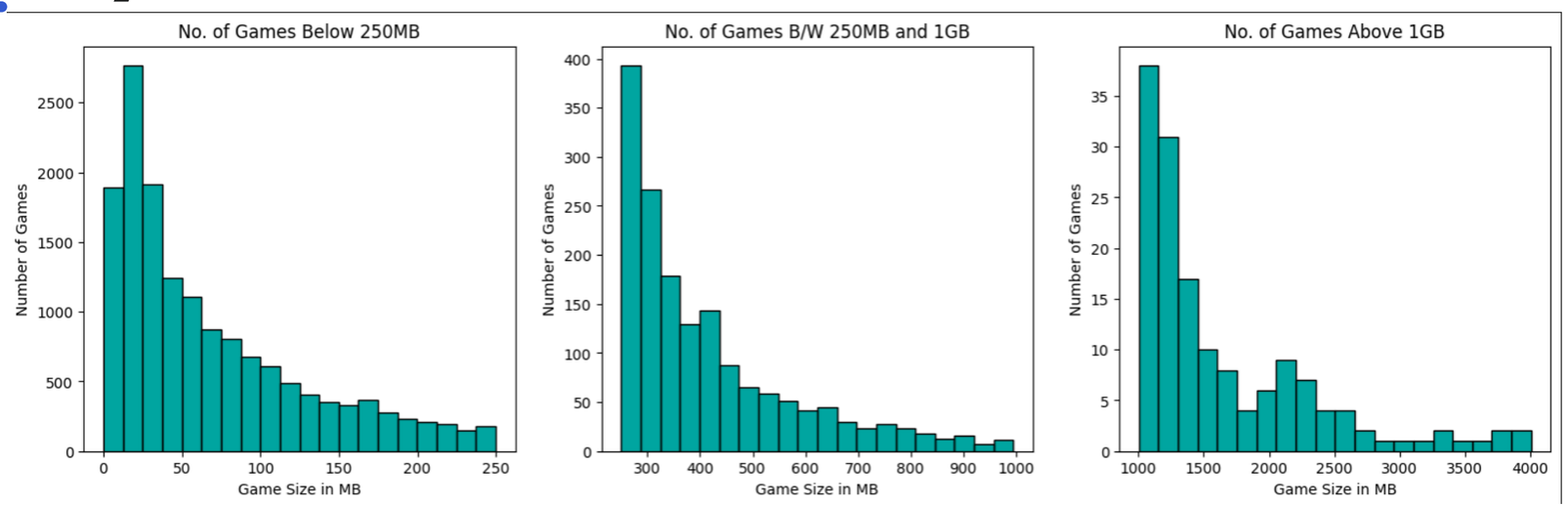
- Genres were extracted and analyzed to determine the popularity of different game types. A horizontal bar chart visualized the distribution, showing the most common genres available in the App Store.

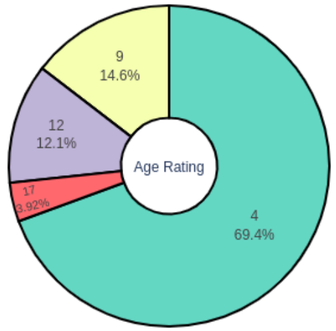
Insights into Game Genres:

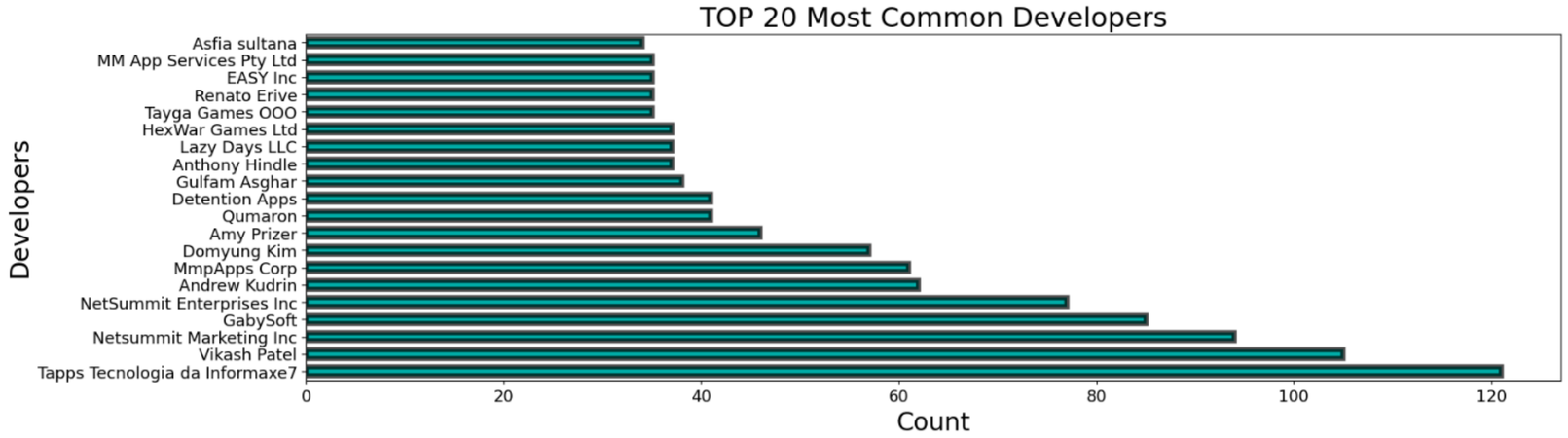
- Genres were extracted and analyzed further, limiting them to primary genres. This was visualized using a bar chart, highlighting the number of games associated with each genre and their popularity trends.

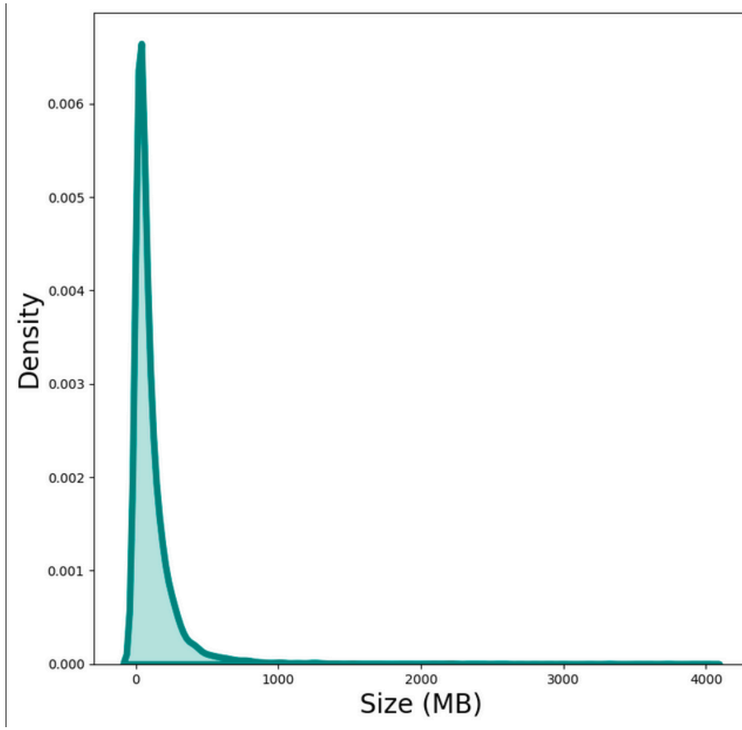
Exploratory Data Analysis (EDA)











Exploratory Data Analysis (EDA)

The EDA performed on the dataset provided valuable insights into various aspects of mobile gaming applications:

Distribution of Average User Ratings:

- The count plot visualized the distribution of Average User Ratings, highlighting the number of games falling into each rating category. This provided a sense of user satisfaction levels across the dataset.

Game Size Analysis:

- Game sizes were transformed into MB for better readability, and a Kernel Density Estimate (KDE) plot was used to examine the distribution of sizes.
- The dataset was categorized into three size groups:
- Games smaller than 250MB.
- Games between 250MB and 1GB.
- Games larger than 1GB.
- Histograms were plotted for each size group, revealing the density and distribution within these ranges.

Top Developers:

- The top 20 most common developers were identified and visualized using a horizontal bar chart. This shed light on which developers contribute the most to the dataset, providing insights into prominent players in the industry.

Age Rating Distribution:

- A pie chart was created to illustrate the distribution of games across different Age Rating categories. The visualization highlighted the percentage and count of games suitable for each age group, offering insights into target audiences.

Game Genres:

- Genres were extracted and analyzed to determine the popularity of different game types. A horizontal bar chart visualized the distribution, showing the most common genres available in the App Store.

Insights into Game Genres:

- Genres were extracted and analyzed further, limiting them to primary genres. This was visualized using a bar chart, highlighting the number of games associated with each genre and their popularity trends.

Feature Engineering

One-hot Encoding for categorical data

```
df_new.select_dtypes(include = 'object')
```

Python

	Name	Developer	Primary_Genre	Genres	Genreslist
0	Sudoku	Mighty Mighty Good Games	Games	Strategy, Puzzle	Strategy
1	Reversi	Kiss The Machine	Games	Strategy, Board	Strategy
2	Morocco	Bayou Games	Games	Board, Strategy	Strategy
3	Sudoku Free	Mighty Mighty Good Games	Games	Strategy, Puzzle	Strategy
4	Senet Deluxe	RoGame Software	Games	Strategy, Board, Education	Strategy
...	...	...	...	...	...
17002	Stack Puzzle Rise Tower	Zhigang Pei	Games	Entertainment, Casual, Strategy	Entertainment
17003	EachOther	Sultan Shindi	Games	Family, Strategy	Family
17004	Rabbit Vs Tortoise	Vishal Baldha	Games	Strategy	Strategy
17005	FaTaLL	Tayrem Games	Games	Strategy, Action	Strategy
17006	The Three Kingdoms Bomb	ming bo tang	Games	Strategy, Puzzle	Strategy

16847 rows × 5 columns

```
df_new.Genres.nunique()
```

Python

```
787
```

```
df_new = df_new.drop(columns = ['Name','Developer','Genres','Genreslist'])
```

Python

```
df_new = pd.get_dummies(df_new, columns = ['Primary_Genre'], dtype = int)
```

Language Count Creation:

A new column, Language_Count, was derived by counting the number of languages listed in the Languages column for each app.

This simplified the representation of language diversity while retaining the essential information.

Subsequently, the Languages column was dropped as it became redundant.

One-Hot Encoding:

Categorical data in the Primary_Genre column was transformed using one-hot encoding.

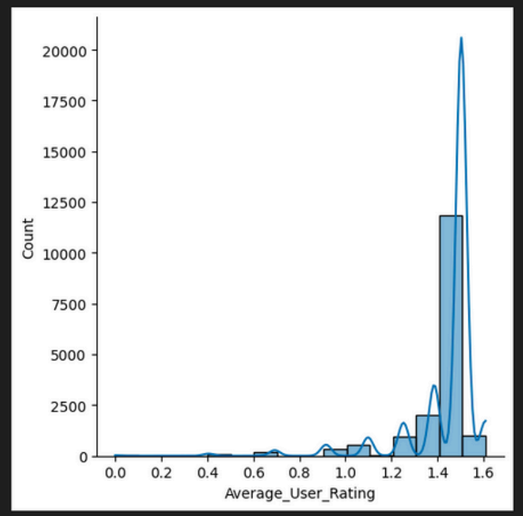
This process created separate binary columns for each unique genre, representing the presence (1) or absence (0) of a particular genre for each app.

The transformation ensures that the categorical data is compatible with machine learning models.

Outlier Detection and Handling

```
sns.displot(np.log(df.Average_User_Rating), kde = True)
#We have done variaous transformations on the columns, but we cannot see much big difference in the distribution of the data.
#So, we are keeping the original variable without any transformation.
```

```
<seaborn.axisgrid.FacetGrid at 0x7f7d1df6fc10>
```



Probability Distribution Analysis:

The distribution of the Average_User_Rating column was visualized using a probability density plot (sns.displot).

A logarithmic transformation (np.log) was applied to the Average_User_Rating column to assess whether it could normalize the data.

Despite the transformation, the distribution remained similar, indicating that the original variable provided sufficient representation without requiring transformation.

Size Column Transformation:

To manage the wide range of values in the Size column, a logarithmic transformation (Log_Size) was created, reducing skewness in the data.

The original Size column was subsequently dropped, as the transformed version retained the essential information with reduced variability.

Outlier Handling Decision:

Based on the transformations and visual inspections, no significant outliers were identified in the analyzed columns that warranted removal.

The decision was made to retain the original data variables (or their transformed versions) as they provided meaningful insights without further modification.

Model Development

Objective:

- The goal was to predict the Average_User_Rating of apps based on various features using multiple regression models.

Data Preparation:

- The dataset was split into features (X) and target (y), where X included all columns except Average_User_Rating, and y was the target variable.
- A train-test split was performed with 70% of the data for training and 30% for testing, ensuring a robust evaluation.

Models Used:

Several regression models were evaluated:

- Linear Regression
- Decision Tree Regressor
- Random Forest Regressor
- Support Vector Regressor (SVR)
- k-Nearest Neighbors (KNN) Regressor

Model Development

The RMSE (Root Mean Squared Error) values for each model are as follows:

Linear Regression:

- RMSE: 0.512
- A simple, interpretable model that performed decently, but slightly less effective than some ensemble methods.

Decision Tree Regressor:

- RMSE: 0.645
- This model overfits the training data more easily, leading to higher error on the test data.

Random Forest Regressor:

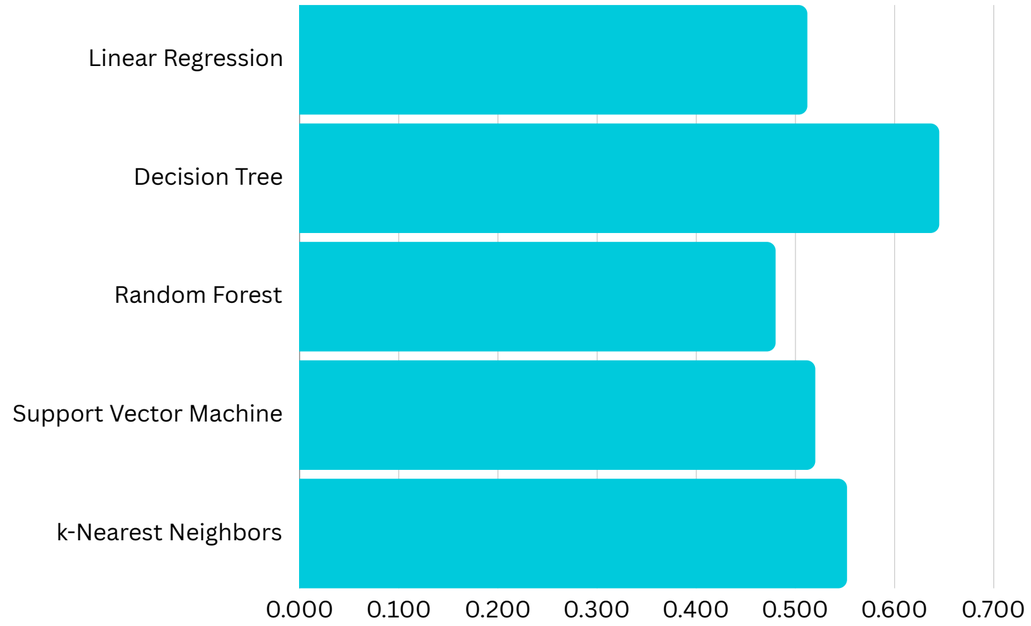
- RMSE: 0.480
- Best-performing model among the tested options, leveraging ensemble learning to reduce variance and improve accuracy.

Support Vector Regressor (SVR):

- RMSE: 0.520
- Competitive performance, demonstrating the ability to capture complex patterns in the data but slightly less effective than Random Forest.

k-Nearest Neighbors (KNN) Regressor:

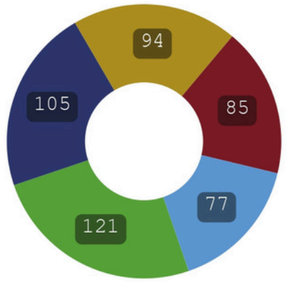
- RMSE: 0.552
- Performed reasonably well, but its accuracy can be sensitive to the choice of hyperparameters like the number of neighbors.



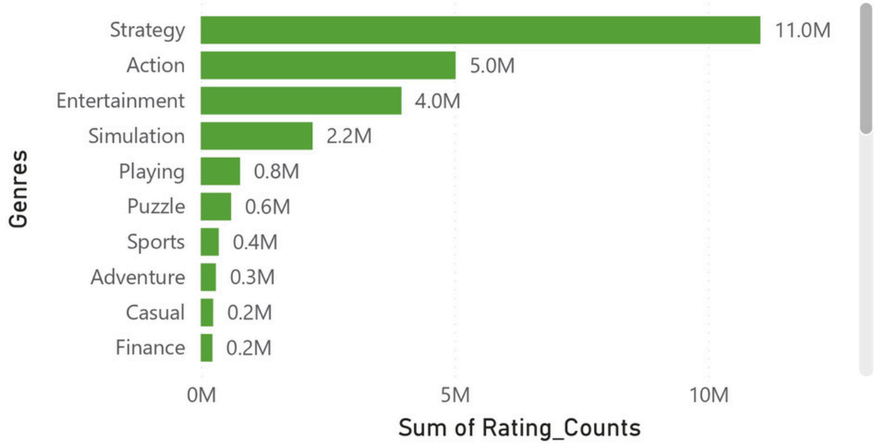
Results and Insights

Comprehensive Game Data Analytics

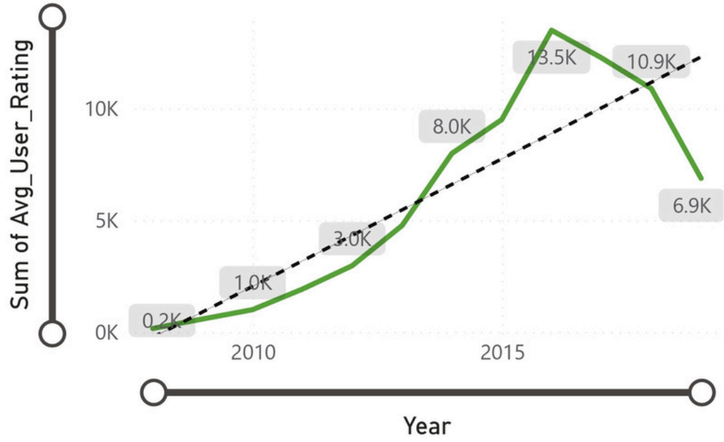
Top 5 Developers by APP Count



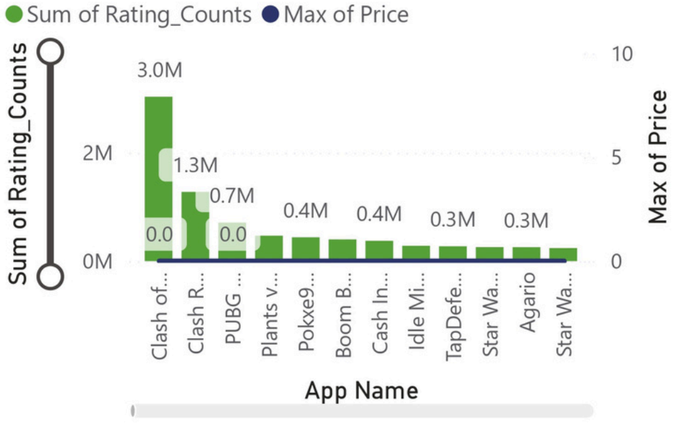
Sum of Rating_Counts by Genres



Sum of Avg_User_Rating by Year



Sum of Rating_Counts and Max of Price by App Name



TOP 5 BY User Rating Count

App Name	Sum of Avg_User_Rating	Rating_Counts
Clash of Clans	4.50	3,032,734.00
Clash Royale	4.50	1,277,095.00
Plants vs Zombiesu2122 2	4.50	469,562.00
Pokxe9mon GO	3.50	439,776.00
PUBG MOBILE	4.50	711,409.00

4
Min of Age_Rating

17
Max of Age_Rating

Most Expensive Games

Price	App Name
179.99	African Cheetah Wild Animal Simulator 3D Full
179.99	Army Truck Offroad Simulator 3D Full Drive mili truck
179.99	Big Blue Whale Survival 3D Full Try whale simul ocea
179.99	Big Elephant Simulator Wild African Animal 3D
179.99	Black Wild Panther Simulator 3D Full Be a wild anima
179.99	Cat Simulator Cute Pet 3D Full Be a kitten tease
179.99	City Goat Animal Survival Simulator 3D Full
179.99	City Tower Crane 3D Simulator Full Real city

Results and Insights

Key recommendations and insights from the game data analytics image:

Top developers by app count:

- Tapps Technology and Vikash Patel are the top two developers with 121 and 94 apps respectively.

Sum of rating counts by genres:

- Strategy games have the highest total rating counts at 11M, followed by Action at 5M and Entertainment at 4M.

Average user rating trend:

- The average user rating has steadily increased from around 0.2 in 2010 to around 10.9 in 2015, indicating an overall improvement in game quality over time.

Top 5 games by user rating count:

- Clash of Clans, Clash Royale, Plants vs Zombies 2, Pokémon GO, and PUBG MOBILE are the top 5 games by user rating count.

Age rating:

- The minimum age rating is 4, while the maximum is 17, suggesting a range of game content from family-friendly to more mature.

Most expensive games:

- The most expensive games listed are premium titles like African Cheetah Wild Animal Simulator 3D Full and Army Truck Offroad Simulator 3D Full Drive mini truck.

Conclusion

The analysis provides valuable insights into the game app ecosystem, highlighting key trends in developers, genres, and user engagement. Tapps Technology and Vikash Patel are the top developers, with Tapps leading by a significant margin in the number of apps. Strategy games, especially, dominate in terms of total rating counts, followed by action and entertainment genres. The steady rise in average user ratings from 2010 to 2015 indicates a noticeable improvement in game quality.

Popular games such as Clash of Clans, Clash Royale, and Pokémon GO stand out as the top performers in terms of user ratings. The broad range of age ratings (from 4 to 17) reflects the diversity of content, catering to both family-friendly and mature audiences.

Pricing insights reveal that premium games, such as African Cheetah Wild Animal Simulator 3D and Army Truck Offroad Simulator 3D, are among the most expensive, indicating the potential for high-value, specialized content.

In terms of Machine Learning Models, the performance of various algorithms was evaluated for predicting the Average User Rating of games. The Random Forest Regressor emerged as the best-performing model with the lowest RMSE (Root Mean Squared Error) of 0.480, followed closely by Linear Regression (RMSE: 0.512). Other models such as Support Vector Machine (SVM) and k-Nearest Neighbors (KNN) showed competitive results but slightly higher RMSE scores. These insights suggest that ensemble models like Random Forest are particularly effective for predicting ratings, while simpler models like Linear Regression still provide a reasonable level of accuracy.

Overall, the analysis recommends focusing on game quality improvements, exploring high-demand genres, and leveraging machine learning models like Random Forest for performance predictions and trend analysis to guide future development strategies.

THANK YOU

gomathisankar400@gmail.com