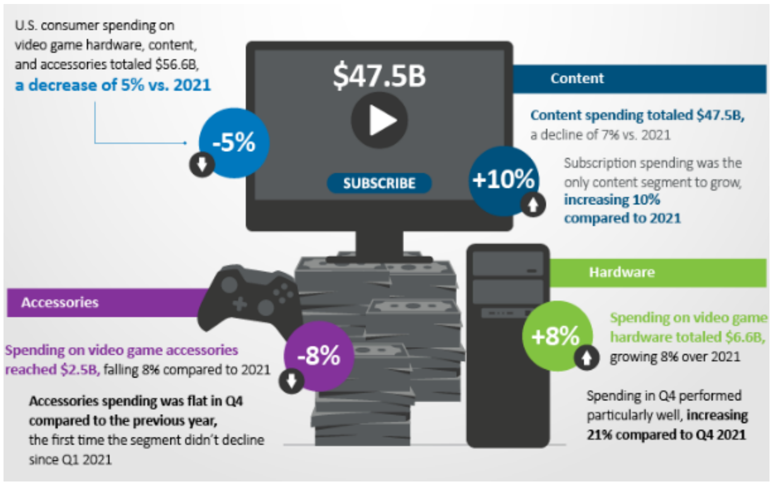
Predicting TOP rated upcoming games

Project by: Andrea Ternera

Introduction to the gaming industry:

In 2021 the US spent $60.4bn USD. According to NDPs report, that was an increase of 8% year-over-year. Hardware sales saw more growth than software with console revenues up 14% from 2020. *However, in 2022 the industry saw a decrease of 5% compared to 2021 across all video game markets. North America was the second highest region spending $49.7bn which saw a decrease of 2.5% YOY.* According to NDPs video game market snapshot of 2022 spending, we can see the following: Hardware spending totaled $6.6 bn which entails a 8% growth YOY. Accessories spending reached $2.5bn falling 8% YOY. Content spending totaled $47.5 bn falling 7% YOY; however, subscription spending increase by 10%*.



Recap of problem:

Overall the gaming industry keeps growing significantly and is forecasted to reach $206.4 bn globally in 2025*. For this analysis we will focus on games released in North America, which take up roughly 27 percent of the total global market (United States accounting for 24.8 percent), and games released worldwide. Lastly, the analysis will separate results by segment (mobile, console, PC, etc.) and analyze game features (content, perspective, mode, etc.) of top ranked games that could potentially make them more popular than other games in similar genres. The goal is to develop a model that could potentially predict a game's rating based on various features.

Obtaining the Data, Cleaning and performing EDA:

In order to complete this project, the following was used as source of data:

IGBD developer API – for detailed information on games data.

- It consists of different endpoints that provide very detailed information about various games. Depending on the goal of the project different endpoints can be used to obtain the different features desired for modeling.
- The data collected include details such as collection, company, event, language, and platform details. It also provides region, release date-status, genre, and franchise endpoints.
- While the API provides access to a great number of game details, this project utilized the following endpoints to gather the necessary data and features for modeling:
    - Game
        - age_ratings, aggregated_rating, aggregated_rating_count, alternative_names,artworks, bundles,category,checksum, collection,collections, cover,created_at,dlcs, expanded_games,expansions, external_games, first_release_date, follows, forks, franchise, franchises, game_engines, game_localizations, game_modes, genres, hypes, involved_companies, keywords, language_supports, multiplayer_modes, name, parent_game, platforms, player_perspectives, ports, rating,rating_count, release_dates,remakes, remasters, screenshots, similar_games, slug,standalone_expansions, status, storyline, summary, tags, themes, total_rating, total_rating_count, updated_at, url, version_parent, version_title, videos, websites.
    - Platforms
        - abbreviation, alternative_name, category,checksum, created_at,generation, name, platform_family, platform_logo, slug,summary, updated_at, url, versions, websites.
    - Genres
        - Id, checksum, created_at, name, slug, updated_at, url

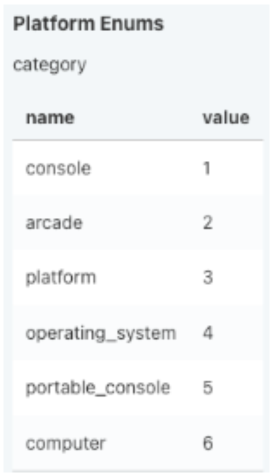
<u>Game endpoint</u>:

- After requesting over 5800 entries, using POST command in Jupyter notebook using game endpoint there were multiple features that had over 60% missing values for the games provided. These columns were dropped which left the dataset with the following features:
    - Age_ratings: The PEGI rating
    - Category: The category of the game
    - checksum: Hash of the object
    - Collection: The collections that the game is in
    - Cover: The cover of the game
    - Created_at: Date this was initially added to the IGDB database.
    - External_games: External IDs the game has on other services.
    - First_release_date: The first release date for the game
    - Follows: Number of people following a game
    - Game_modes: Modes of gameplay
    - Genres: Genres of the game
    - Id: unique identifier of game
    - Involved_companies: companies who developed this game.
    - Name
    - Platforms: Platforms the game was released on
    - Player_perspectives: The main perspective of the player
    - Rating_count: Total number of IGDB user ratings
    - Rating: Average IGDB user rating
    - Release_dates: Release dates of the game
    - Similar_games
    - Slug: A url-safe, unique, lower-case version of the name
    - Summary: a description of the game
    - Tags: Related entities in the IGDB API
    - Themes: Themes of the game
    - Total_rating_count: Total number of user and external critic scores.
    - Total_rating: Average rating based on both IGDB user and external critic scores.
    - Updated_at: The last date this entry was updated in the IGDB database.

  - url: The website address (URL) of the item
  - Websites: Websites associated with the game
- There were 29 games with a duplicate name, every one due to a sequel or remake of the game. Therefore, ID was used as the unique identifier for each games.

<u>Platform endpoint:</u>

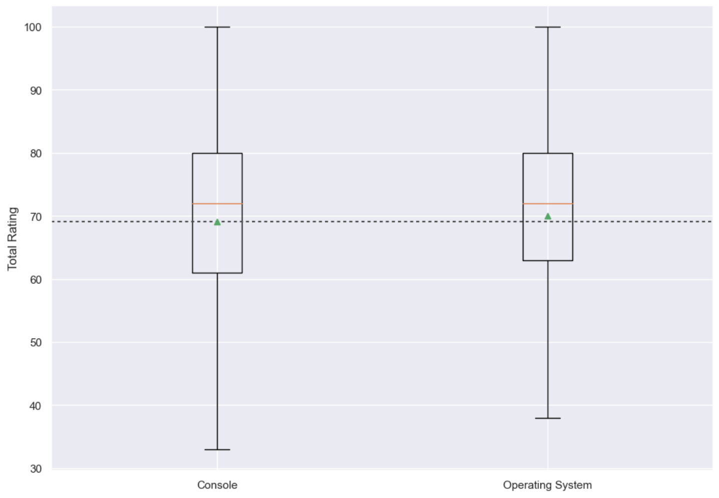
- We also used a POST request to gather information about the platforms since the platforms provided in the game endpoint are a list of platform IDs and does not really support or assist with deeper analysis.
  - This endpoint provided all 200 unique platform IDs with its corresponding name. These 200 platforms fall within 6 categories:
    - Category 1 contains 80 unique console platforms (Sega Pico, PS2, Meta Quest, Odyssey, etc.)
    - Category 2 contain 3 unique Arcade platforms (Hyper Neo Geo 64, Neo Geo MVS, and Arcade)
    - Category 3 contain 8 unique platform (Amazon Fire TV, Web browser, Google Stadia, AirConsole, etc.)
    - Category 4 contain 10 unique operating_system platforms (iOS, PC=Microsoft Windows, Mac, Android, Linux, etc.)
    - Category 5 contain 34 unique portable_console platforms (Gamate, Playdate, Sega Game Gear, Nintendo DSi/3DS/DS, Game Boy, PSP, etc.)
    - Category 6 contain 55 unique computer platforms (Commodore CDTV/Plus4/PET/16, Atari 8-bit, Amiga, Apple II, HP 2100/3000, etc.)


Platform Enums

category

name	value
console	1
arcade	2
platform	3
operating_system	4
portable_console	5
computer	6

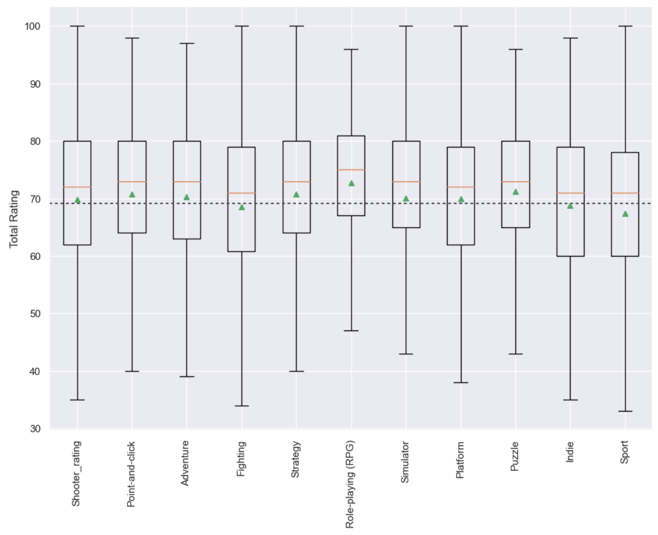
- At this stage, we created a new column to represent each category and provided binary response if platform id in game dataset matched id in platform dataset.
- Having gathered the necessary data from both sources we found our games have the following spread of platform category:
  - Console found in 3369 games.
  - Operating system found 2931 games.
  - Total rating on both platforms is very similar, median of console games is slightly higher than operating systems. The spread of operating systems rating is also smaller than console games.



- The most rated game (by count) in the dataset is Grand Theft Auto V

<u>Genre endpoint:</u>

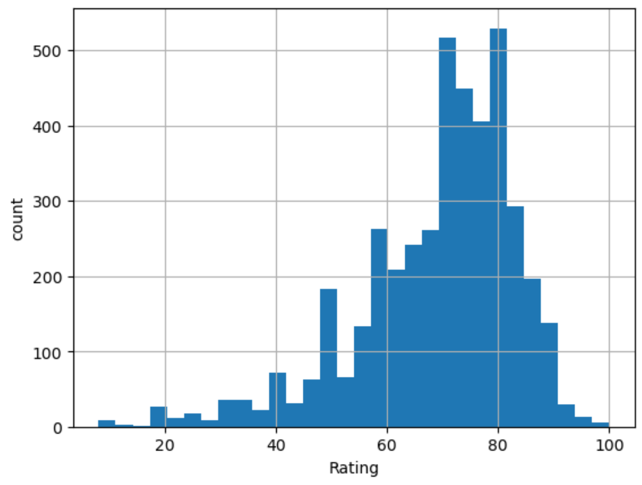
- In order to also identify the different game genres and understand their relation with rating we also created columns for each genre in the data by using another post request for the genre endpoint. This provided the name of the id for each genre. It was also identified that a singular game can have multiple genres. There is a total of 23 different genres with the most common genres being:
    - Shooter
    - Point-and-click
    - Adventure
    - Fighting
    - Strategy



- The shooter genre also has the highest overall count of games with the highest ratings with around 1000 games rated 51-75 and 800 rated 76-100.
- Strategy and Role-playing genre games have a pretty similar distribution in rating with an almost 50-50 balance between games rated 51-75 and 76-100.
- Shooter, Adventure, and Point-and-click have the largest count of low voted games with almost 200 for each genre that falls below the 50% rating.

<u>Selecting target variable:</u>

- After evaluation of both rating and total rating, it was decided to use total_rating as the target variable as it has ratings from IGDB as well as other sites and provides more rating in count as well. As seen in the graph most games are rated between 70 and 80.

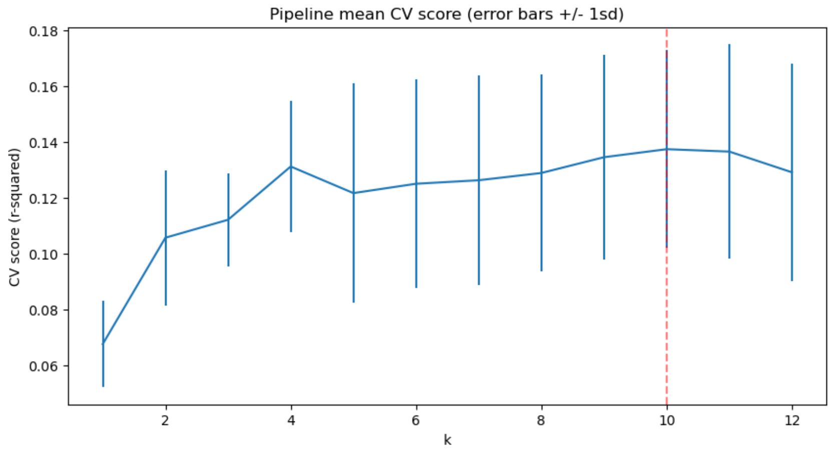


Model Training and Testing:

- We decided to split the data with a 70% training - 30% testing. Also all object dtypes were moved added to "name_list" and dropped from both training and test sets.
    - This provided a test set of 2985 entries and a test set with 1280 entries, both with 12 columns with dtypes float and int.
- Using DummyRegressor we tried the mean as the predictor for our model.

Model - Linear Regression:

- Using .fillna we imputed null values with the median and used StandardScaler to have all values in a consistent scale. Then we trained a LinearRegression model and evaluated its performance.
- Similarly we used .fillna but this time using the mean to identify if it was a better metric to use with null values. Performance wise we evaluated r2 score, MAE and MSE to which imputing with mean performed slightly better in all MAE and MSE.
- By using a pipeline with SimpleImputer, StandardScaler, LinearRegression, and adding SelecKBest we also checked a couple more options for our model.
    - Based on our GridSearchCV, best K for our linear regression model is k=10
    - The CV score is the same as k=11 but the variance at k=10 is less thus a better option for the model.
    - Based on this model the top positive features include total_rating_count, id, category, console, updated_at, and operating system.

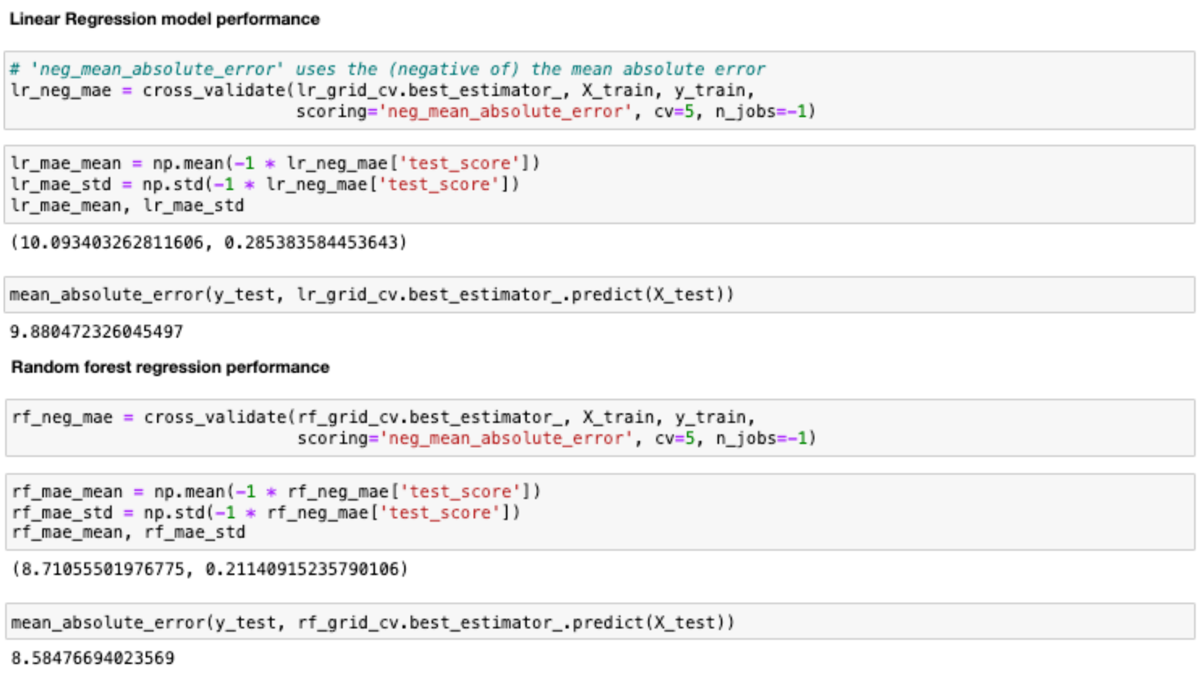


Model – Random Forest:

- Using a pipeline from the start, we developed a model using median as the strategy for the SimpleImputer and RandomForestRegressor as the model.
- When checking for recommended parameters, it was determined that best n_stimators would be 297 and the simple imputer strategy would be mean as well as using standard scaler for the data.
    - Based on this model, the top positive features include total_rating_count (which concurs with LR model), cover, updated_at, year, collection.
    - Interestingly, the least positive features are operating_system, console, and category.

Model – Selection:

- After comparing the performance of both models the *random forest model* showed a lower cross-validation MAE by over 1 point. It also exhibited less variability than the Linear Regression model.



Linear Regression model performance

```
# 'neg_mean_absolute_error' uses the (negative of) the mean absolute error
lr_neg_mae = cross_validate(lr_grid_cv.best_estimator_, X_train, y_train,
                            scoring='neg_mean_absolute_error', cv=5, n_jobs=-1)
```

```
lr_mae_mean = np.mean(-1 * lr_neg_mae['test_score'])
lr_mae_std = np.std(-1 * lr_neg_mae['test_score'])
lr_mae_mean, lr_mae_std
```

```
(10.093403262811606, 0.285383584453643)
```

```
mean_absolute_error(y_test, lr_grid_cv.best_estimator_.predict(X_test))
```

```
9.880472326045497
```

Random forest regression performance

```
rf_neg_mae = cross_validate(rf_grid_cv.best_estimator_, X_train, y_train,
                            scoring='neg_mean_absolute_error', cv=5, n_jobs=-1)
```

```
rf_mae_mean = np.mean(-1 * rf_neg_mae['test_score'])
rf_mae_std = np.std(-1 * rf_neg_mae['test_score'])
rf_mae_mean, rf_mae_std
```

```
(8.71055501976775, 0.21140915235790106)
```

```
mean_absolute_error(y_test, rf_grid_cv.best_estimator_.predict(X_test))
```

```
8.58476694023569
```

- We also evaluated the training set size to make sure we had sufficient data for the model. The errorbar indicated that when the data set reaches a size of 700 entries it reaches its peak and stabilizes after that which shows we have more than enough training data for the model.

Conclusion:

Our model has an error of around 8.6 rating points using our random forest model. While this can be acceptable for some popular games, the findings of this project tell us that our models are not necessarily good enough for less popular games. Therefore, given the short time frame for this project, there are several limitations that will need to be worked on. We may conclude that data of just selected games in the post request might not provide sufficient information that leads to high rating in a game.
First, the relation between a game's rating and the total rating count. It is inherent that popularity of a game increases the number of reviews/ratings provided for such game.

Therefore, the connection between the two is very strong and might take away from other game features that can provide a better or worse gaming experience which leads to a high or low rating.
Another limitation is the lack of platform diversity on the provided range of games with the post request. This limits the visibility on the possible influence of platform on rating.
Given our conclusion and the provided limitations, more data would need to be gathered to evaluate if it is possible to predict a game's rating regardless of popularity.