Top Anime 2024

(myanimelist.net)

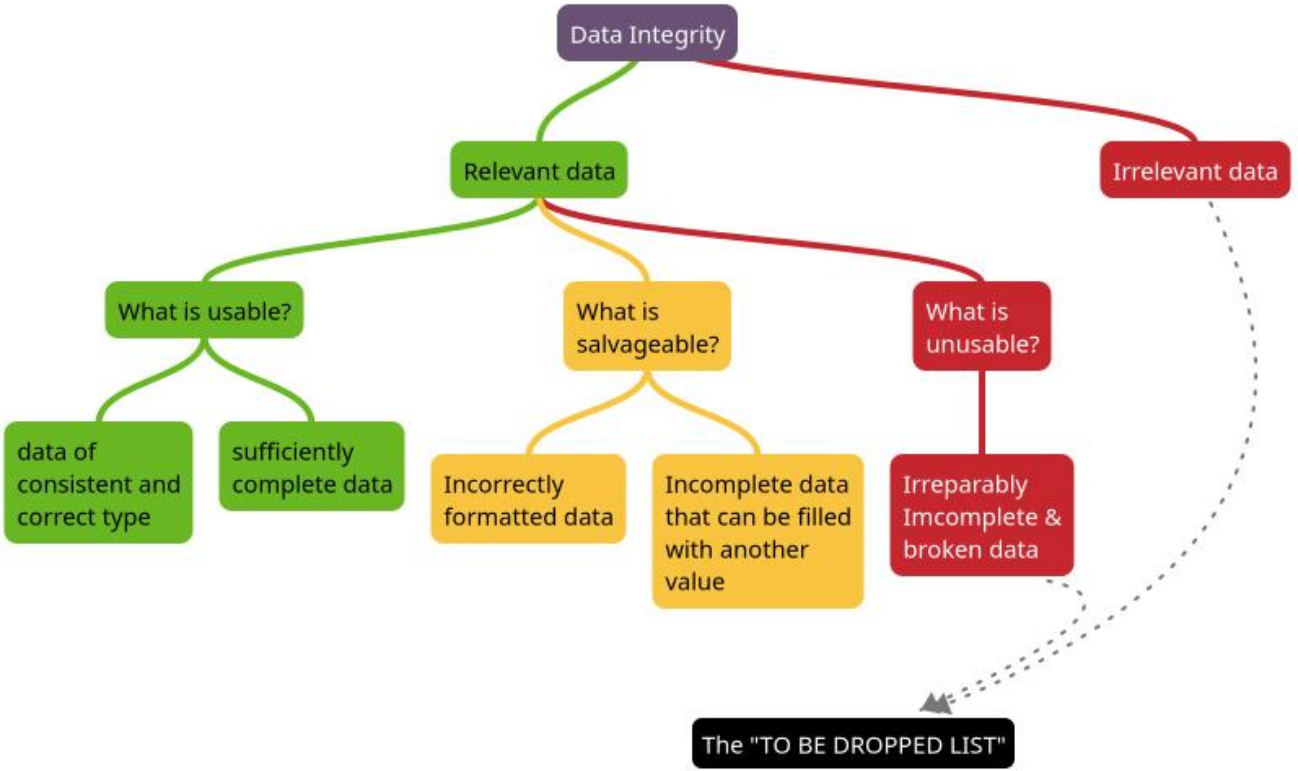
by
Eric Salsac & Aitor Vergara

https://www.kaggle.com/datasets/bhavyadhingra00020/top-anime-dataset-2024/data

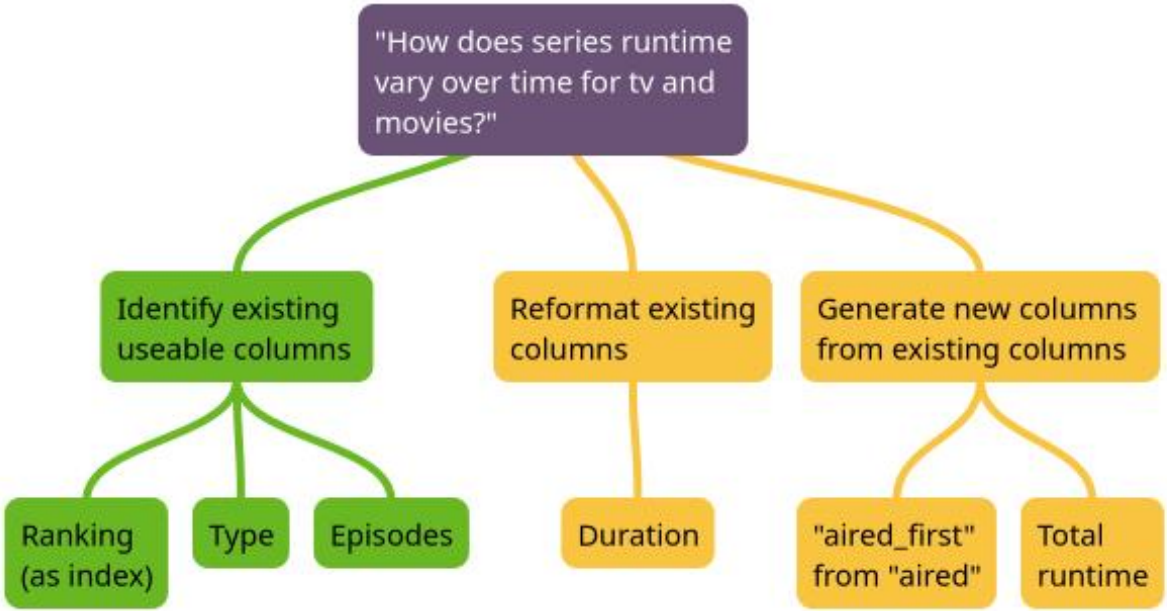
Presentation Overview

1. Overall Cleaning Process Overview
2. Q1 Cleaning Process Specifics
3. Q1 Implementation & Findings
4. Q2 Cleaning Process Specifics
5. Q2 Implementation & Findings
6. “The 19 Columns of Genres” Problem
7. Teamwork Implementation
8. Final Thoughts

The Cleaning Process in General



Q1 Cleaning Overview



Existing Usable Columns •

- No duplicate rows were detected in the dataframe.
- “Rank”, “Episodes” and “Type” were 100% complete and the correct type
- Rank was sequential and unique so made an ideal index.

Zero issues

Reformatting existing columns • Duration

The good

- 100% complete, no null values or blank whitespace

The bad

- Presented as a string when a numeric value was required
  - For TV: formatted as “1 hr. 1 min. per episode”
  - For Movies: formatted as “1 hr. 1 min.”

The solution

- using .replace()
  - to remove per episode
  - to replace min. & hour. with 1 and 60 respectively
- np.prod for sections of list.

Generating New Columns • Aired_first

The good

- “Aired” is 100% complete, no null values or blank whitespace

The bad

- String formatted with both first and last air date separated by “ to ”

The solution

- .split(“ to ”) and keep first index
- pd.to_datetime()

Yes, “aired_last” did exist but ended up being irrelevant…

Generating New Columns • total_runtime

The good

- Simply a product of two existing columns episode_runtime and episodes

The bad

- ¯\(ツ)/¯

The solution

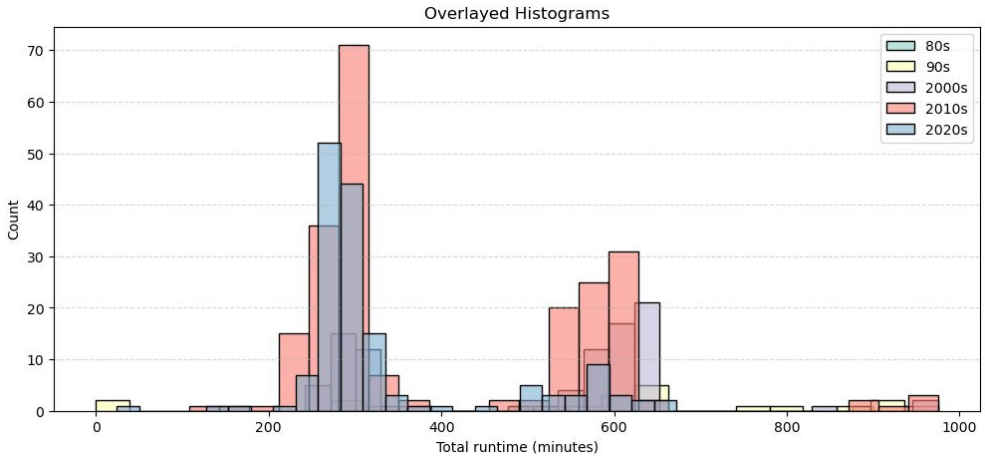
- ¯\(ツ)/¯

Implementation Q1

- Analyse episode duration and episode distribution
  - *With these insights, the current "typical" anime tends to have around 12 episodes with a 23-24 min. duration*
- Check the total runtime => a pattern was presented
- Create decades based on “aired_first”
- Create histograms on each decade => confirms the pattern and its development during decades
- Overlay histograms by pair in chronological order, to analyse the transition.

<u>Conclusions - by 2024 standards</u>

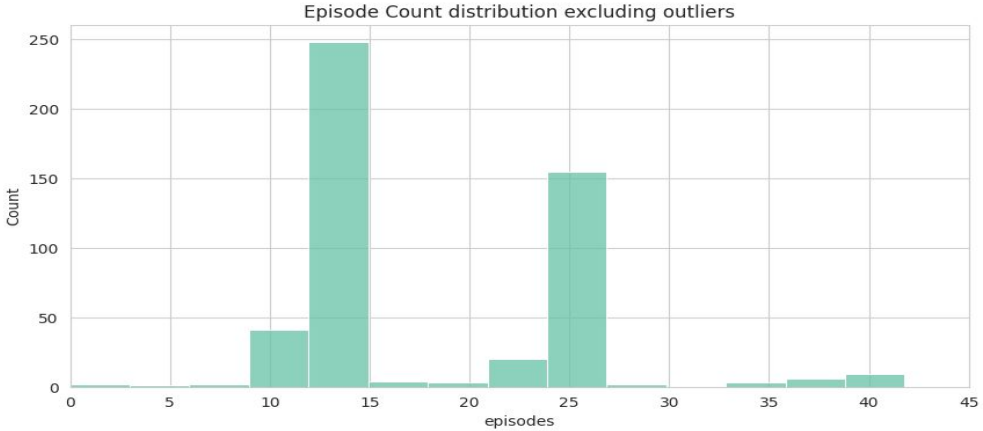
- Popular **modern anime tends to be shorter** in total run time than older anime.
- The trend is to have **a lower episode count** as time moves forward.

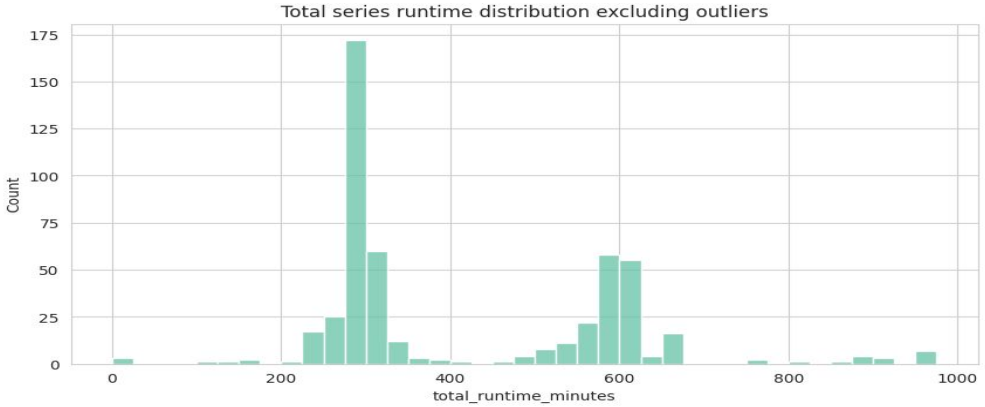


Episode Count & Total Runtime

- The first plot is the distribution of episode count.

- The second plot is the distribution of episode length.





Total Runtime by Decade

- First column of plots is with a variable y-axis to highlight internal trends.
- Second column of plots is with a fixed y-axis to highlight the bias in the volume of modern anime.

1980s

1990s

2000s

2010s

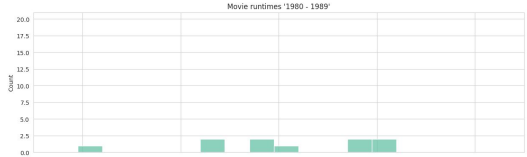
2020 - 2024

Movies

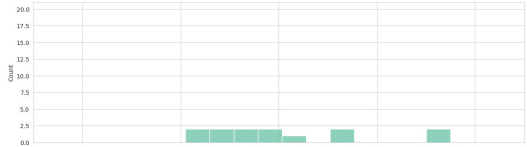
- The distribution maintains a steady mean and median.
- This dataset, in terms of films, is heavily biased to the 2010s.

However, not much else can be said. Any variation can be attributed to low-volume sampling. The majority of the dataset consists of TV shows.

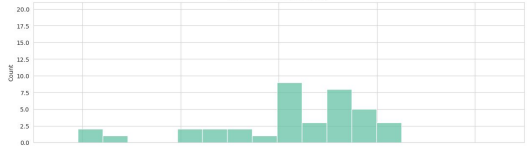
1980s



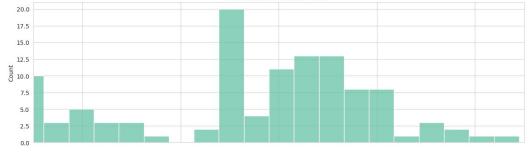
1990s



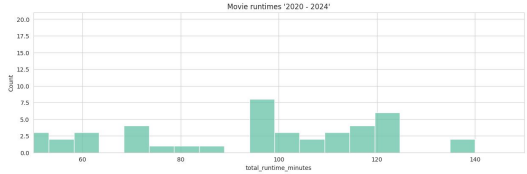
2000s



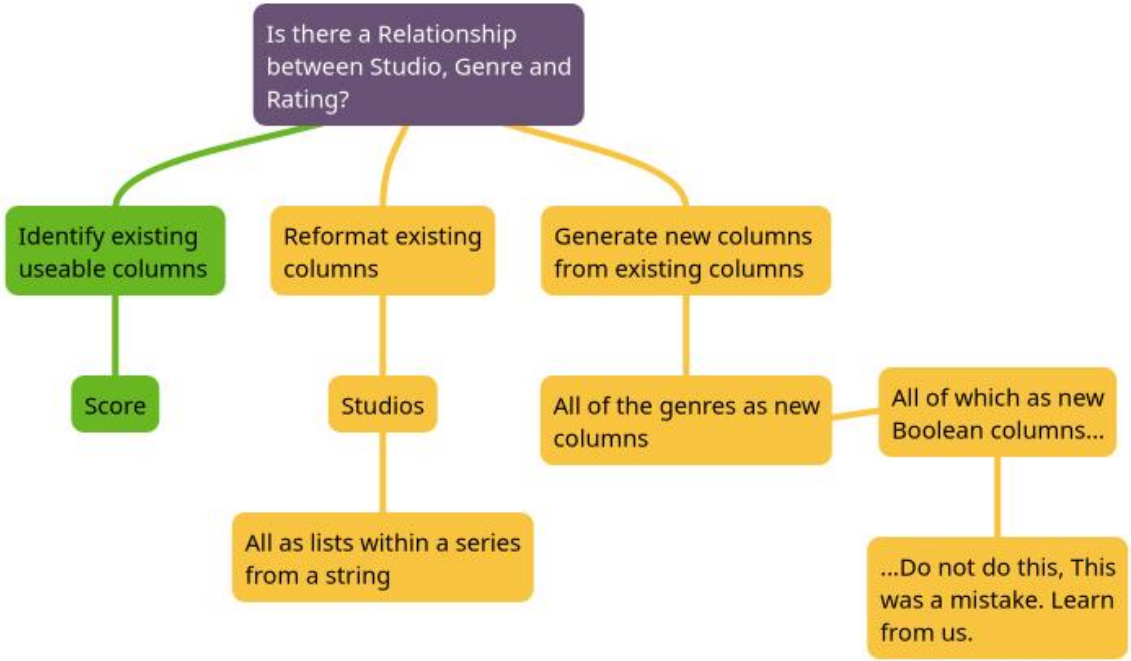
2010s



2020 - 2024



Q2 Cleaning Overview



Existing Usable Columns • Score

- No duplicate rows were detected in the dataframe.
- “Score” was 100% complete and the correct type (float64).
- All values in expected range, i.e. no 11.5/10 scores

Zero issues

Reformatting existing columns • Studios

The good

- 100% complete, no blank whitespace or null values*

The bad

- Defined as strings
- * “None found” and “add some” were present as studios

The solution

- Remove the non-valid values
- Convert the strings in lists. Use an empty set and union with unpack
  - list_unique_studios = set().union(*df["studios"])

Reformatting existing columns • Genres

The good

- The genres were consistently formatted

The bad

- Only 77.1% of the frame had genre data.
- Stored as a list in a string
- The string has every element repeated, (actionaction, romanceromance…)
- String nan values exist and break the pattern of repeating.

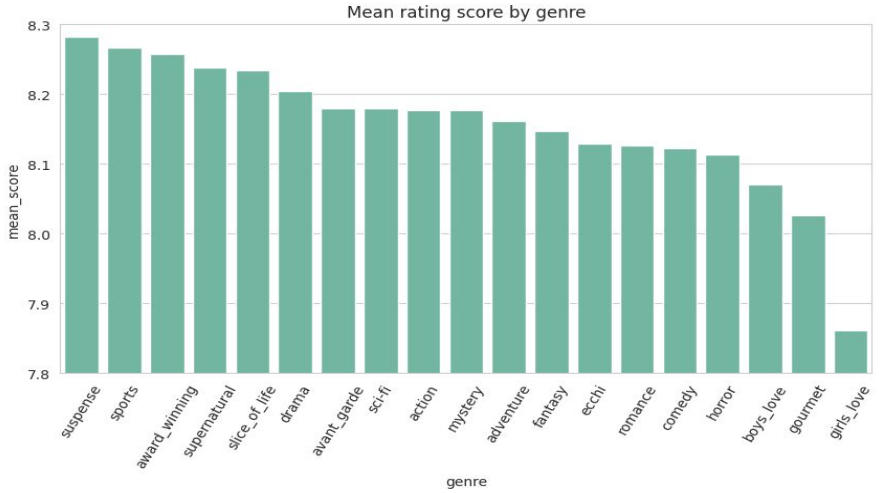
Reformatting existing columns • Genres

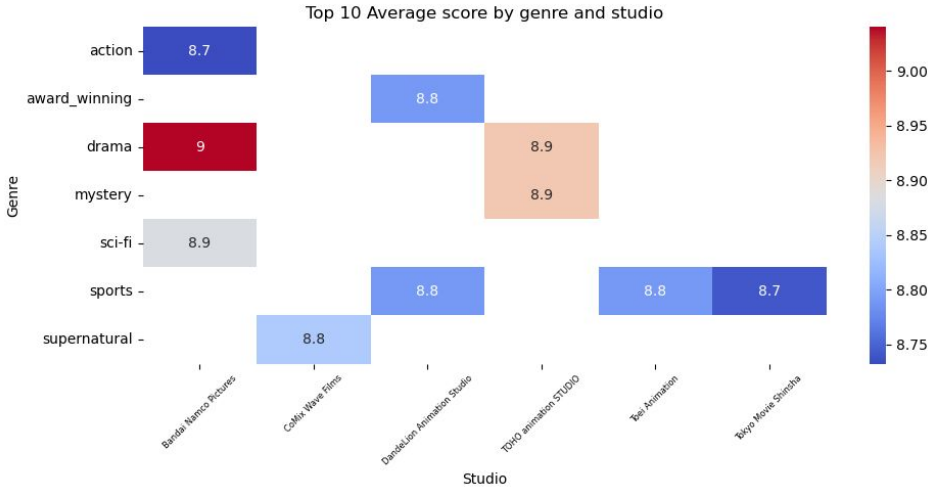
The solution

- In a temporary frame:
  - Split the springs on “,”
  - .explode() so each value has its own row
  - .strip() each row
  - .replace “nan” with NaN via pd.Na
- For every unique value, slice string in half and send to a blank genre list.
- **DO NOT** make new boolean series for each genre.
- Instead return them to lists like we did for studios…

Implementation Q2

- Calculate the average score of each genre: ***Suspense*** *is the most popular*
- Group genres & studios
  - Melting and exploding
- Calculate the average score per genre & studio combination

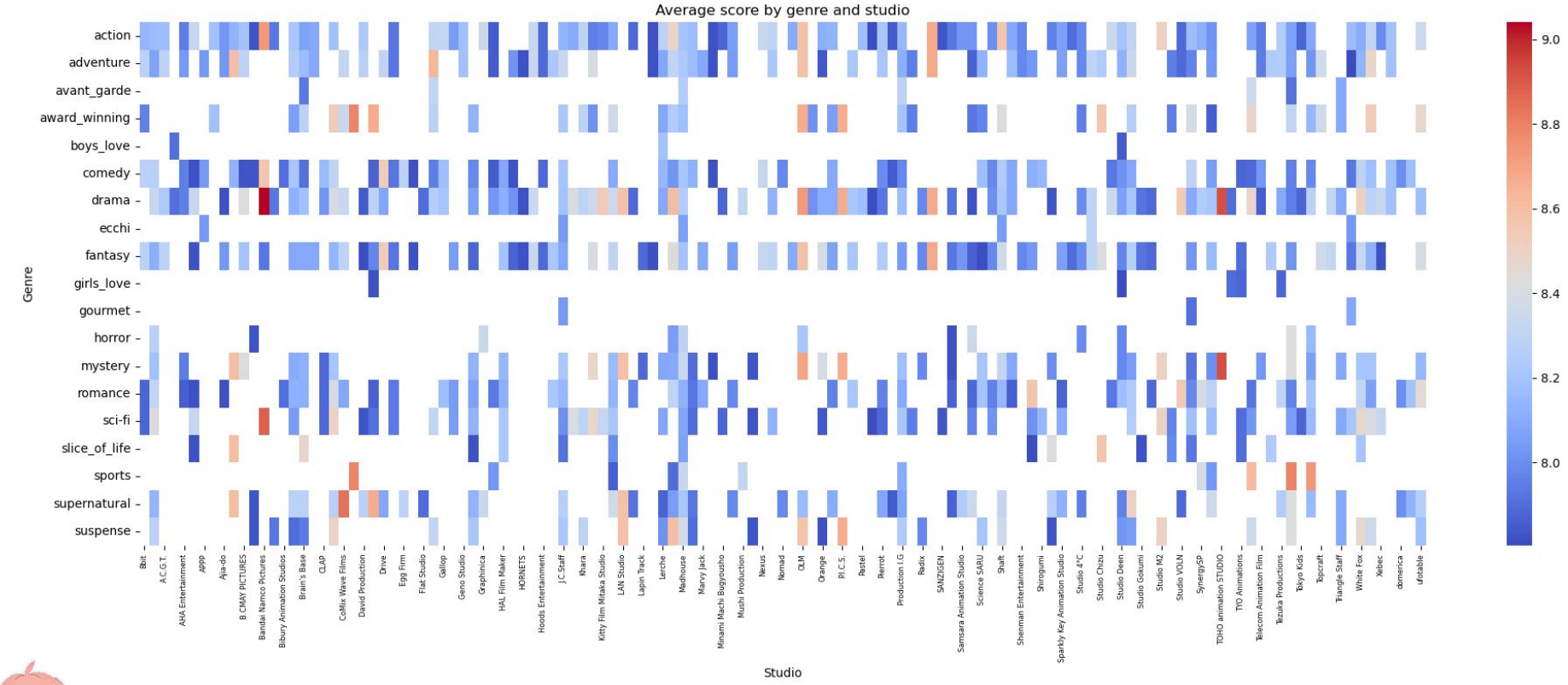




Conclusions

- The most successful combination was: **Drama** animes made by **Bandai Namco Pictures**

Implementation Q2



Major Obstacle - 19 Columns of Genres: The .melt()down

Why it happened:

- Uncertainty over whether storing genres in list form within a series or in boolean series of their own was easier to work with.

How we fixed it:

- .melt() to get a nice long tall frame with genre name and boolean columns
  - (771 rows -> 14649 rows)
- Filter by true on the boolean column
  - (14649 -> 2287 rows)
- Use .groupby() for aggregation needs

19 Columns of Genres: Just .explode() your series instead

Why it didn’t happen:

- Ignorance,

  …yes, yes we used .explode() before but we didn’t realise it would be useful here when formatting and cleaning genres.

Benefits:

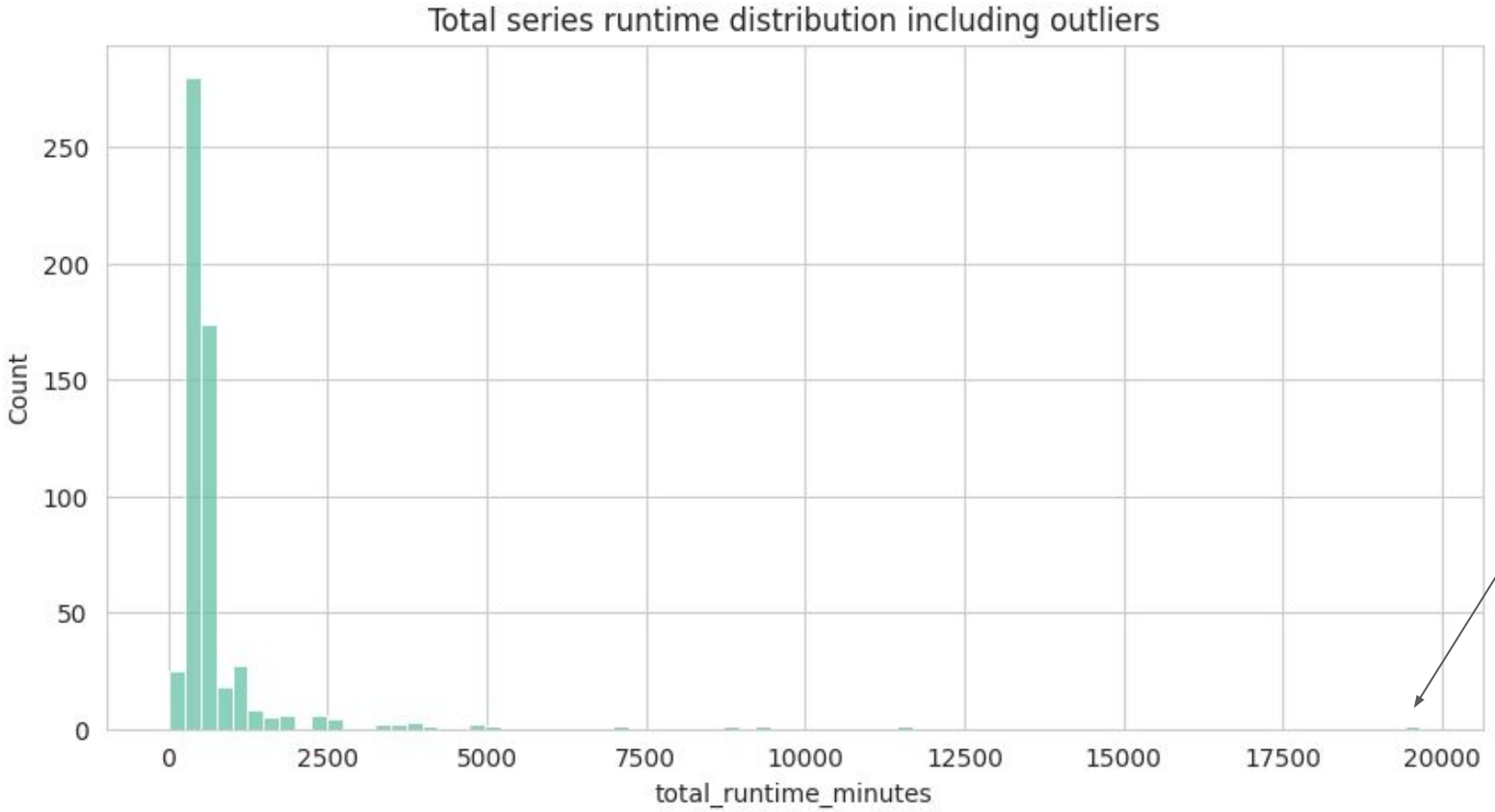
- Your data frame won’t have more than 20 columns of data entries mostly composed of “False”
- The data is more accessible and organised. For example you can count how many times a list element appears in a series.
- Feels more pythonic


TEAM WORK
MAKES THE DREAM WORK
imgflip.com

Final Thoughts

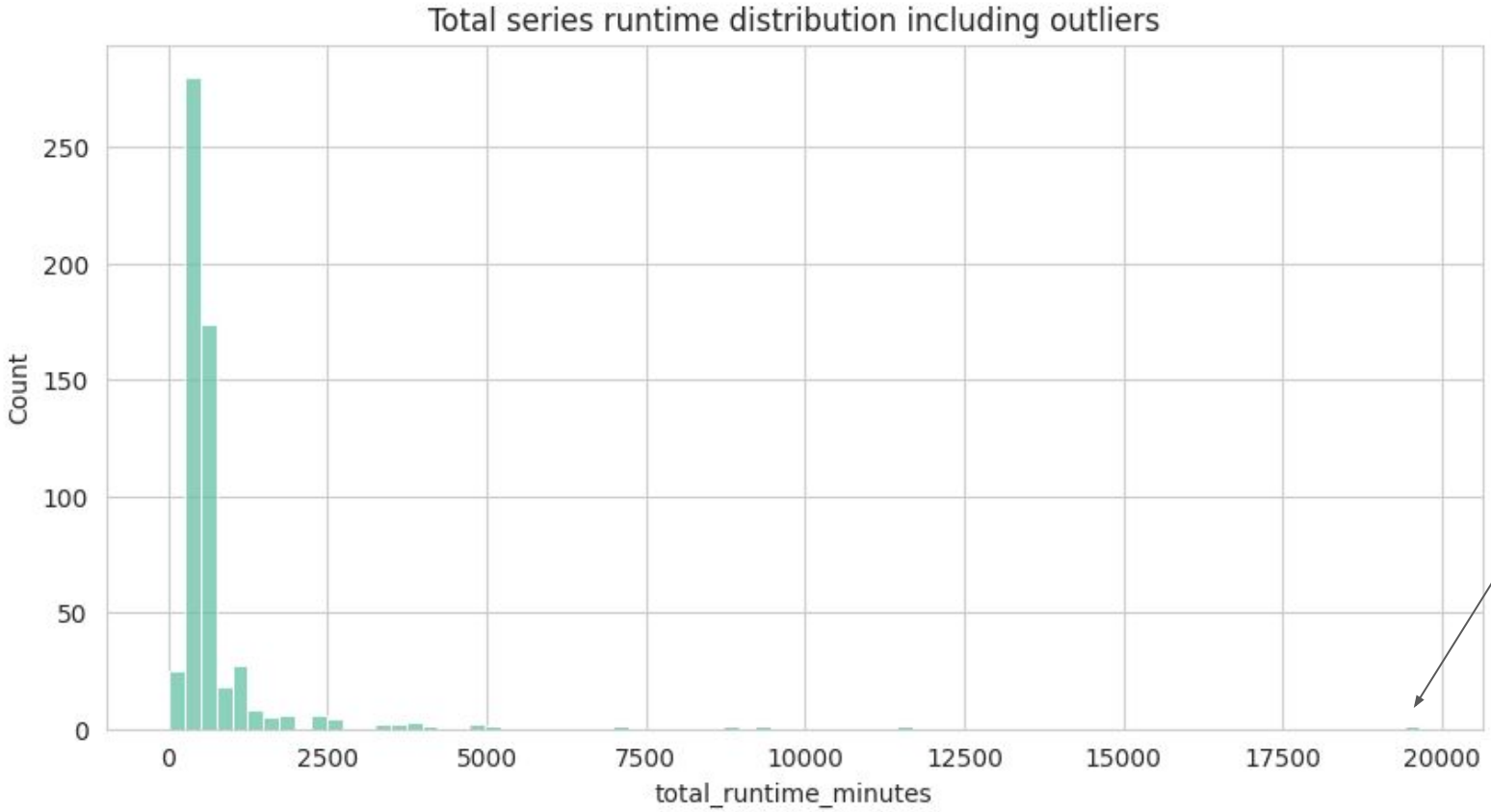
- What can we do with this set in future
  - See which publishers are most prolific in terms of media types (tv ,movies, etc…)
  - Compare number of years on air (yes we did make “last_aired” and standardised it for movies)
  - Compare episode length over the decades

- What would we like from a future set
  - A larger range of data to avoid sample population limits with obscure types (ova, ona, etc…)
  - A dataset that is not so heavily biased to recent releases.
  - *An episode count for “One Piece”*



Filtering outliers Is good…

!?



…Even if
it is Doraemon