ANALYSING

# ANIME PREFERENCES

SC1015 MINI PROJECT

AN AVERAGE 13-EPISODE ANIME CAN COST

# US$2 MILLION

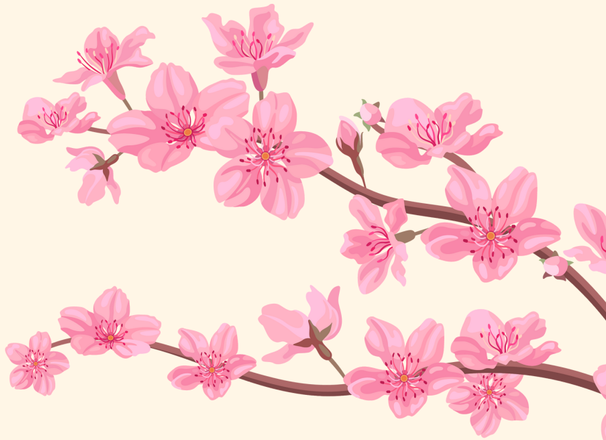
Anime production incurs high cost. How can we maximise the costs efficiently?

# THROUGH...

- Enhanced content creation
  - Analyzing viewers' liking of a new anime can target marketing efforts more effectively towards a specific audience
- Better decision-making
  - Better allocate resources towards anime with a higher chance of success
- Increased revenue
  - With accurate predictions, production companies can create more successful anime, increasing revenue and profitability, while reducing the risk of financial losses.

# PROBLEM STATEMENT

We want to predict viewers' liking of a new anime based on various factors such as producing company, genres, age ratings and number of episodes, etc.



ONE

# DATA CLEANING

# 2022 Anime Database

**Anime Database 2022**

21460 Anime Data at MyAnimeList.net

k kaggle.com

Retrieved from Kaggle, from MyAnimeList.net.

## Remove shows without score

## Remove duplicate data

## Replace missing data

## Combine similar categories into one

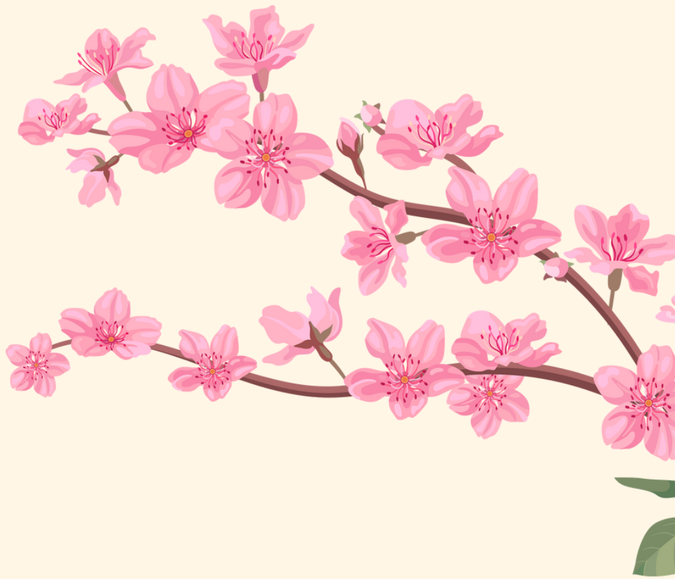
- Themes_Genres

## Remove irrelevant columns

- ID, Japanese (title) etc.

## Remove future anime

- Remove anime with status 'Not Yet Aired'
- Year greater than 2022



TWO

# EXPLORATORY DATA ANALYSIS

# Why score?

- Numerical measure of whether a user likes an anime
  - 7 to 10: Favourable
  - 5 to 6: Moderate
  - 1 to 4: Unfavourable

- Most abundant value in dataset (as compared to other measures like favourites)

Scoring metric on MyAnimeList.net

# Variables Explored

01 TYPES

02 SOURCE

03 THEMES & GENRES (THEMES_GENRES)

04 STUDIOS

05 DEMOGRAPHIC

06 PRODUCERS

07 LICENSORS

08 RATINGS

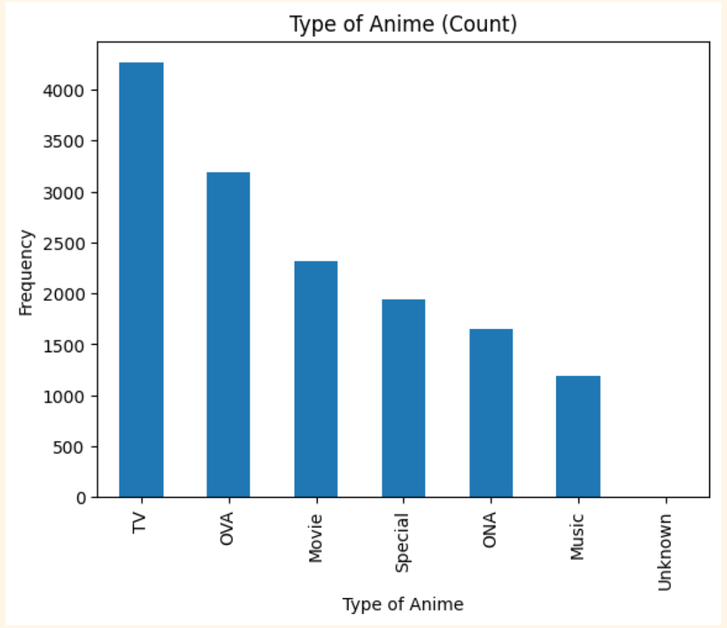
# Variables Explored

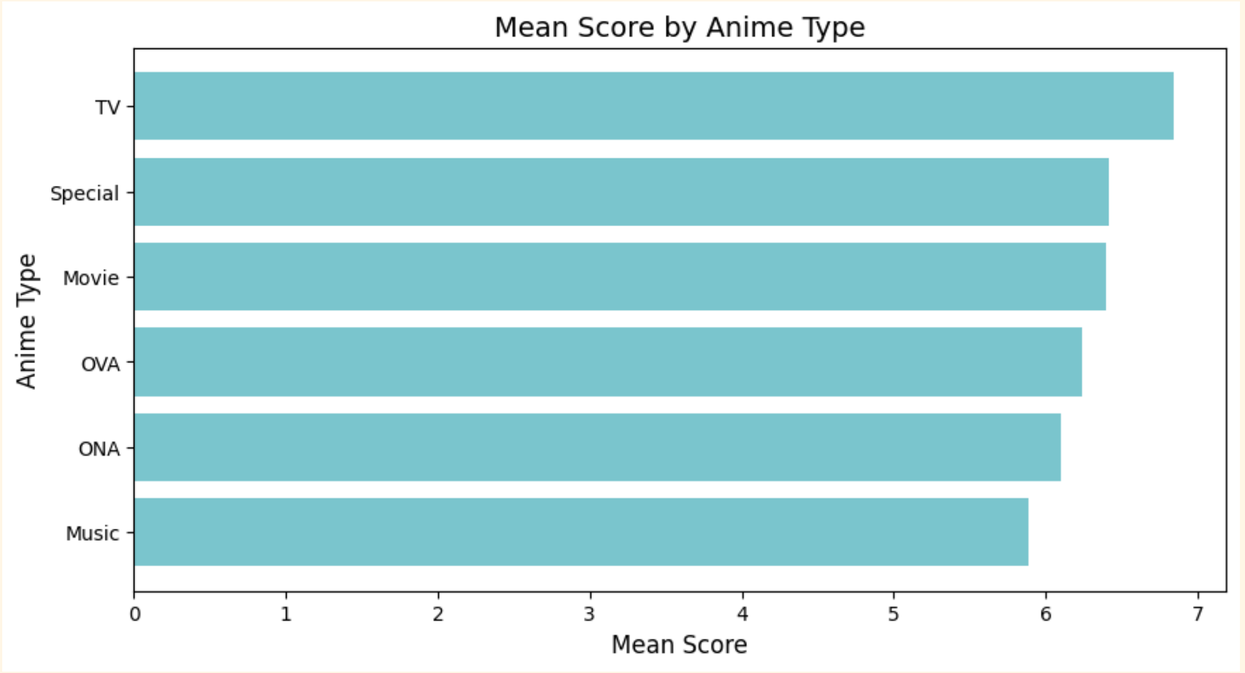
01 TYPES

04 STUDIOS

# TYPES VS SCORE

- Observe how the mean 'Score' of an anime differs according to its 'Type'
- Audiences view 'TV' as most favourable
- 'Special', 'Movie', 'OVA' and 'ONA' and 'Music' as moderate

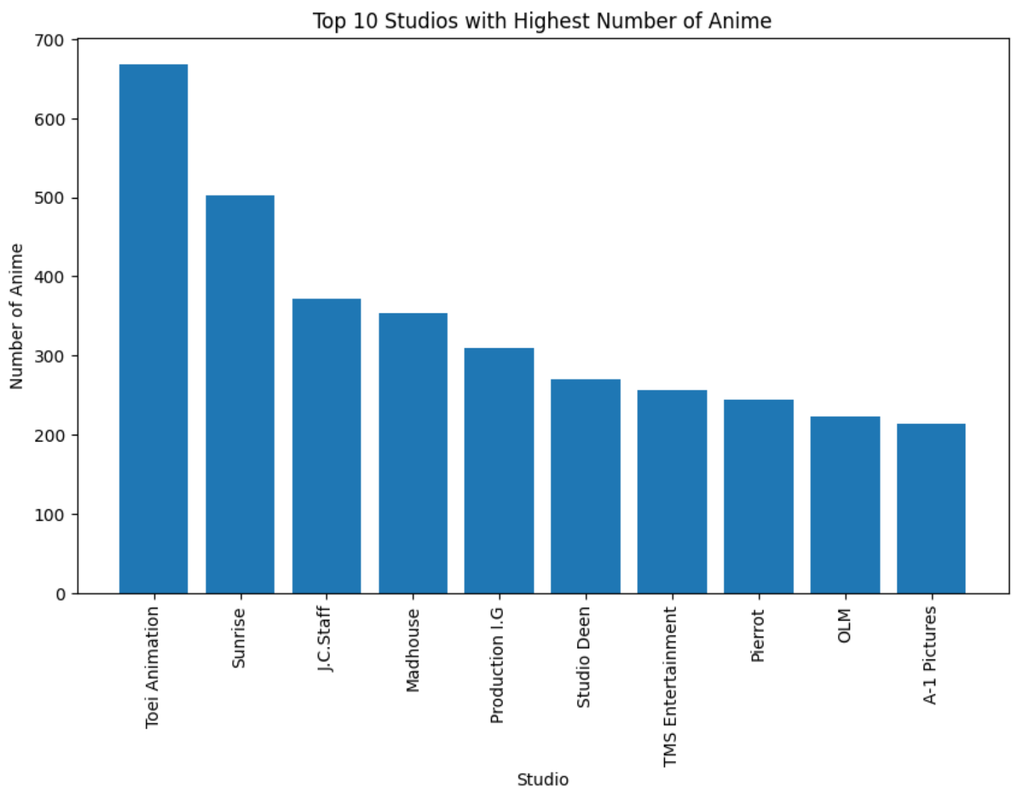


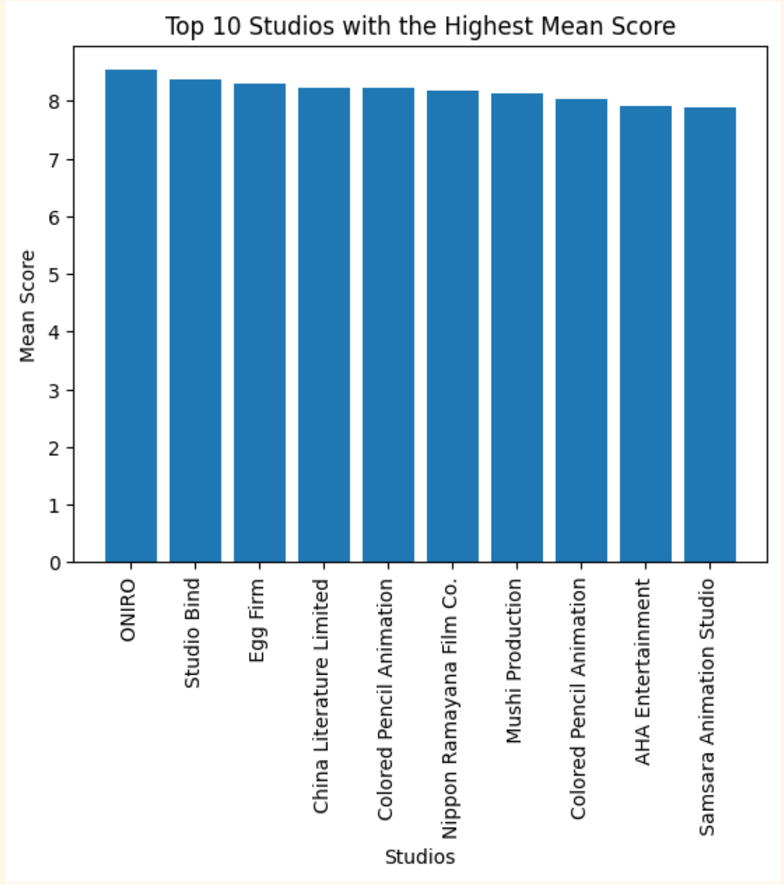

| | Type | Score |
|---|---|---|
| 0 | TV | 6.847 |
| 1 | Special | 6.417 |
| 2 | Movie | 6.400 |
| 3 | OVA | 6.244 |
| 4 | ONA | 6.096 |
| 5 | Music | 5.886 |

# STUDIOS VS SCORE

- Evaluate best studio through:
  - Quantity and Quality




Most productive studio: Toei, Sunrise, and J.C. staff




Most acclaimed studio: ONIRO, Studio Bind, Egg Firm

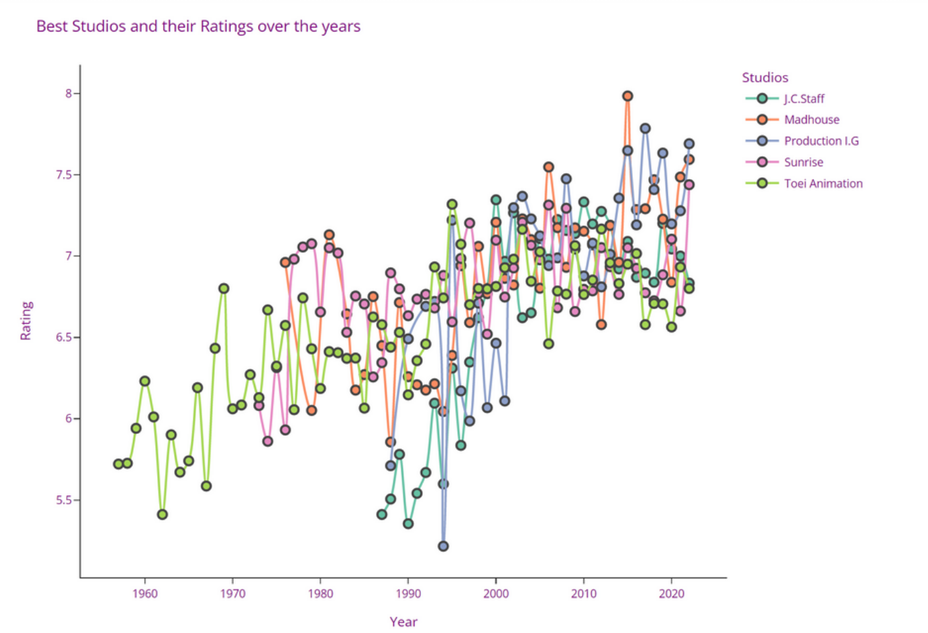
# COMBINE BOTH TO HAVE BEST STUDIO

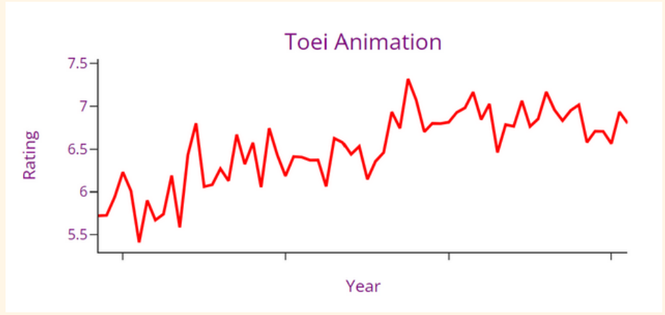
- Tabulate the composite score
- Toei Anime, Sunrise and J.C. Staff among the top 3 best studios
- Same as most productive studio
- Could indicate that audiences prefer quantity over quality

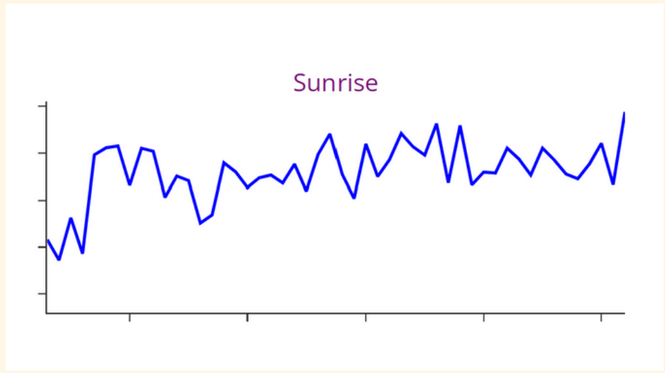
| | Studio | Total Shows | Mean Score | Normalized Mean Score | Composite Score |
|---|---|---|---|---|---|
| **2** | Toei Animation | 668.0 | 6.647937 | 0.405549 | 67.164994 |
| **3** | Sunrise | 503.0 | 6.857544 | 0.679291 | 50.911362 |
| **4** | J.C.Staff | 372.0 | 6.781567 | 0.580066 | 37.722060 |
| **5** | Madhouse | 353.0 | 6.928680 | 0.772194 | 35.994974 |
| **6** | Production I.G | 310.0 | 7.071324 | 0.958483 | 31.862635 |
| **...** | ... | ... | ... | ... | ... |
| **1626** | Public & Basic, Ripple Film | NaN | 4.441000 | -2.476665 | NaN |
| **1627** | Sugar Boy, Blue Cat | NaN | 4.307667 | -2.650795 | NaN |
| **1628** | Studio elle, I-a-unch・BOX | NaN | 3.821000 | -3.286372 | NaN |
| **1629** | ILCA, DRAWIZ, Toho Interactive Animation | NaN | 3.741000 | -3.390850 | NaN |
| **1630** | Production I.G, General Entertainment | NaN | 3.231000 | -4.056899 | NaN |

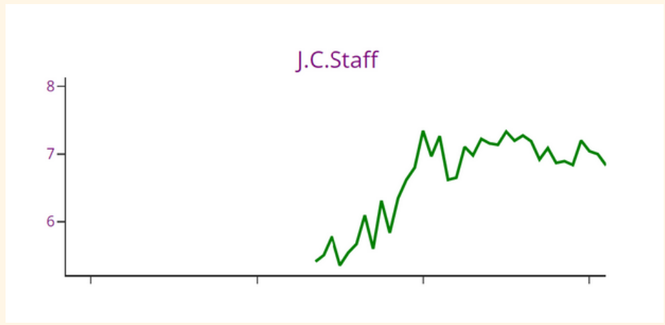
1629 rows × 5 columns

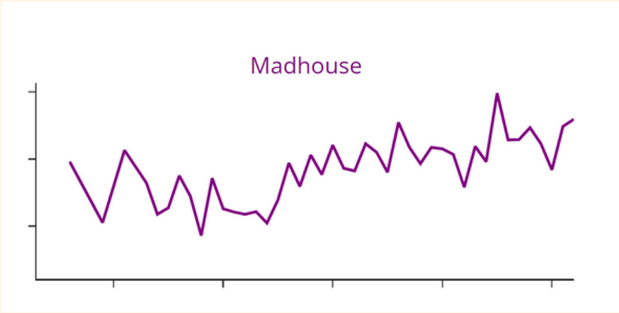
# Best Anime Studio OVER THE YEARS

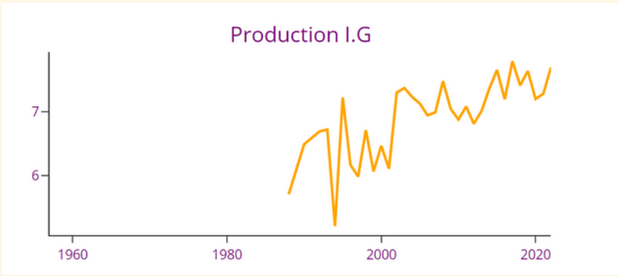


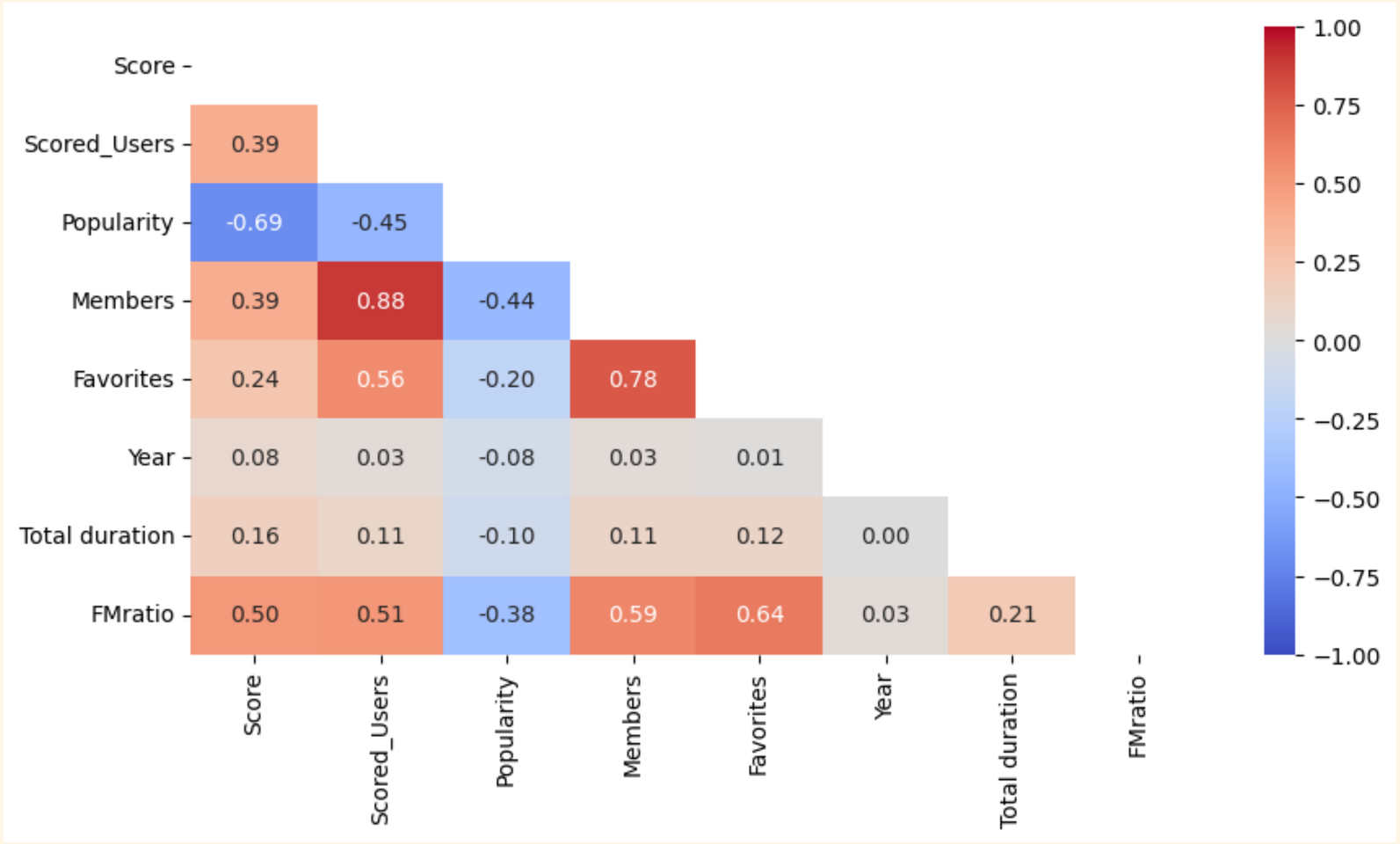

- Stable ratings: Toei Animation
- Increasing ratings: Production I.G. and Madhouse
- Decreasing ratings: J.C. Staff

THREE

# MACHINE LEARNING

# Numerical Data

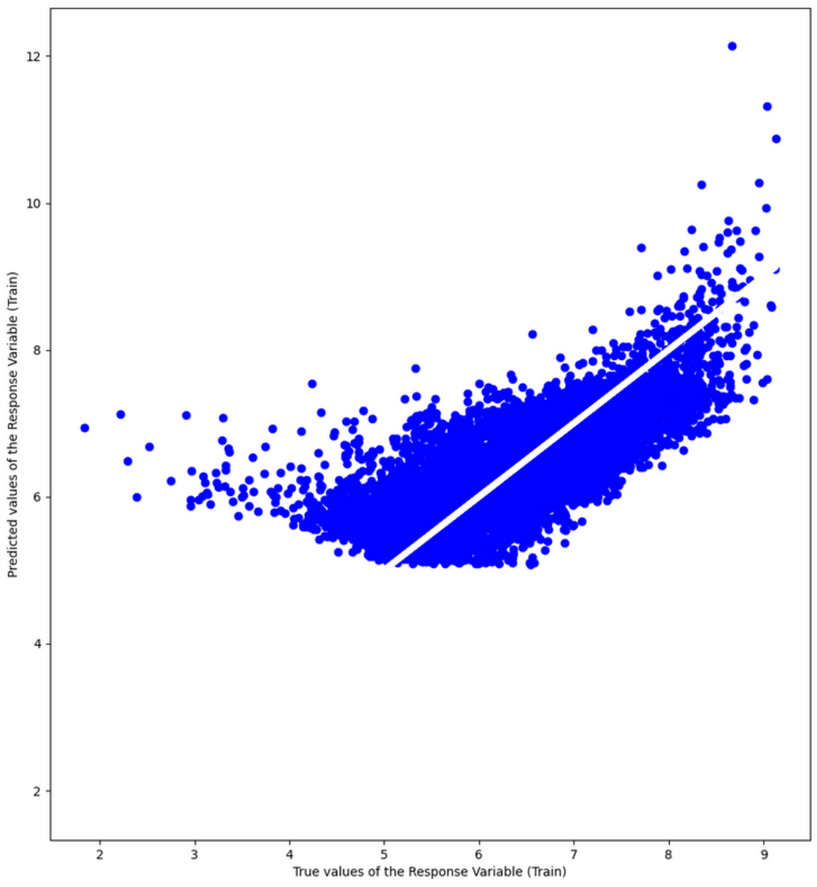
- Scored_Users: Number of Users who gave a score out of 10
- Popularity: Number of Times Searched on the website
- FMratio: Ratio of Favourites to members
  - Favourites: Number of Users who added the title to their "Favourite of all time" album
  - Members: Number of Users who subscribe for updates of the title
- Year: The year the title was published
- Total Duration = No. of Episodes * Duration per Episode

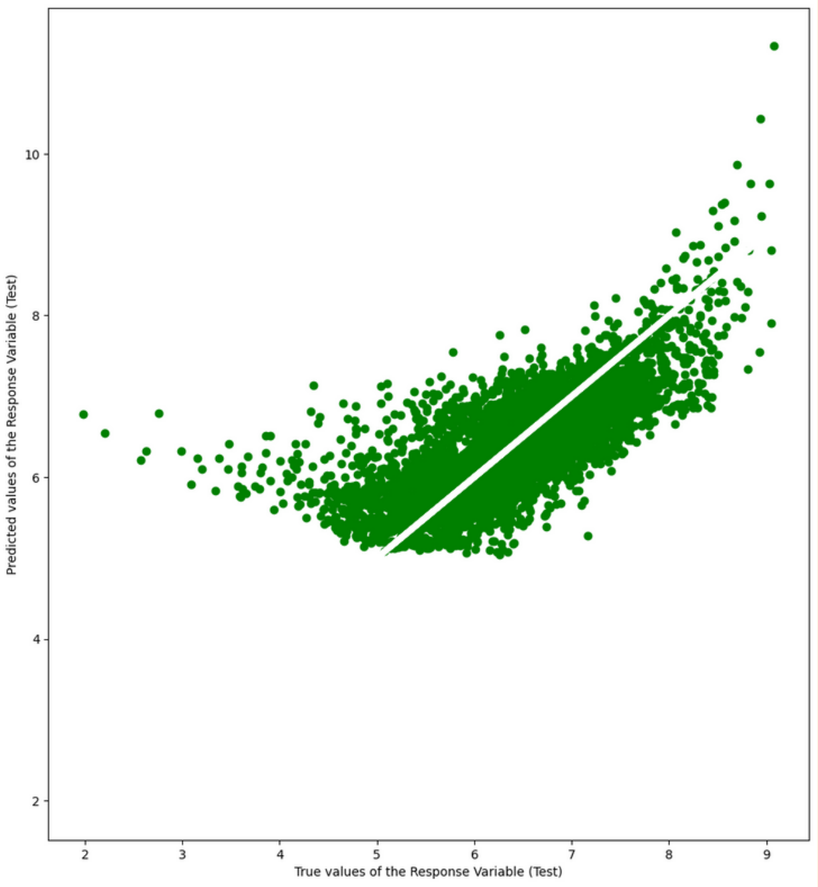



- From this heatmap, we observe low correlation between Predictors, Year and Total duration, to Result Score.
- We continue to drop these variables as predictors.

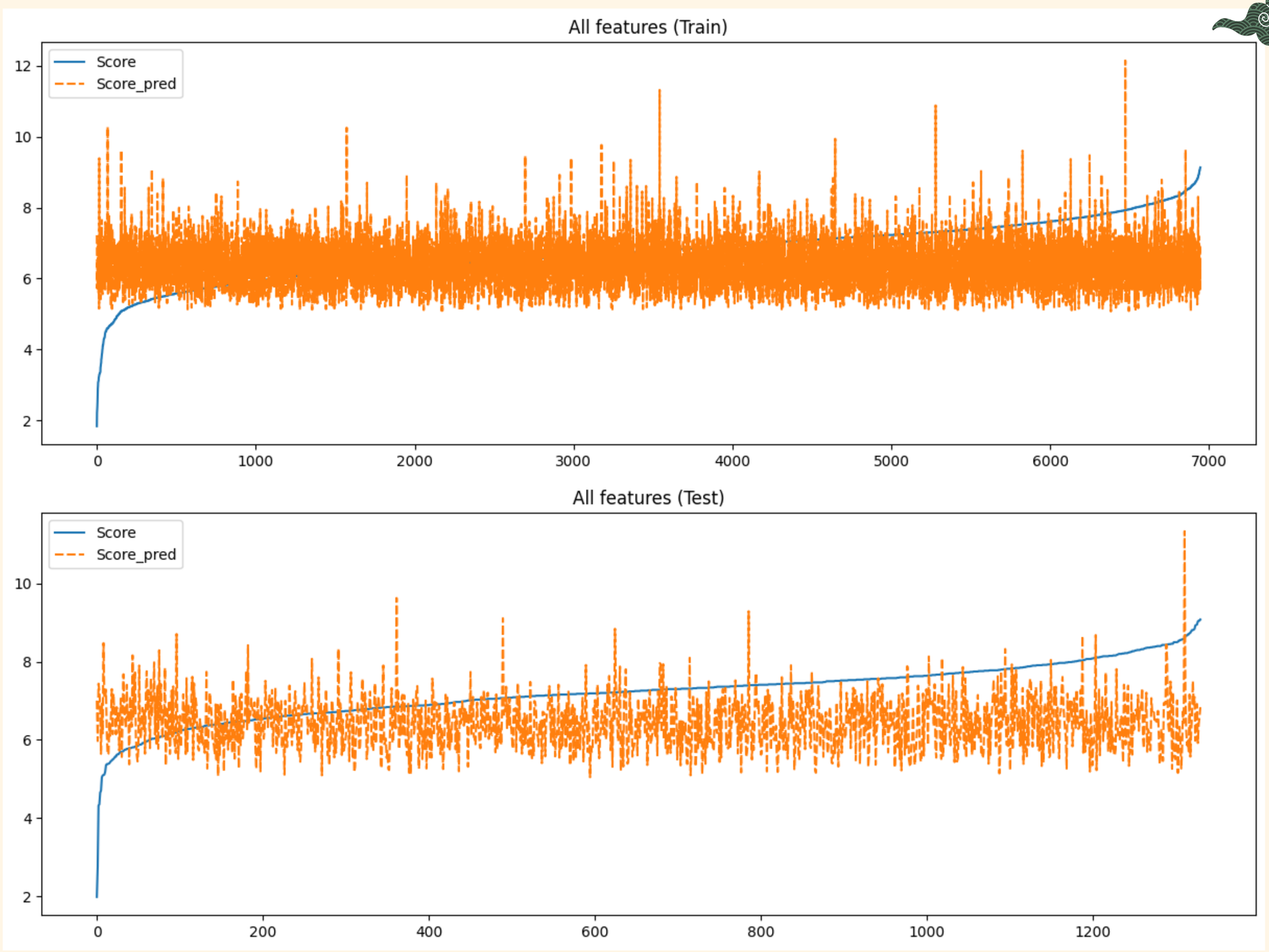
# Multivariate Linear Regression

| | |
|---|---|
| R^2 on Train Set | 0.537 |
| MSE on Train Set | 0.380 |
| MSE on Test Set | 0.379 |



- Tendency to over-predict for lower quartiles of the dataset.
- Good fit for train data but a bad fit for test data
- Overfitting



# Random Forest Classifier

1. Clean the Data
2. Create Dummies for Producers, Licensors and Studios

```
# Split the 'Themes_Genres' column by ',' and create dummy columns for each theme/genre
newdf = newdf.join(newdf['Themes_Genres'].str.get_dummies(', '))

# Drop the original 'Themes_Genres' column
newdf.drop(columns=['Themes_Genres'], inplace=True)

# Split the 'Producers' column by ',' and create dummy columns for each producer

# Create 'Producers' dummy variables
newdf_producers = newdf['Producers'].str.get_dummies(', ')
newdf_producers.columns = [f'Producer_{col}' for col in newdf_producers.columns]  # Add a prefix to column names
if 'Unknown' in newdf_producers.columns:
    newdf_producers['Producer_Unknown'] = newdf_producers['Unknown']
    newdf_producers.drop(columns=['Unknown'], inplace=True)
newdf = newdf.join(newdf_producers)

# Create 'Licensors' dummy variables
newdf_licensors = newdf['Licensors'].str.get_dummies(', ')
newdf_licensors.columns = [f'Licensor_{col}' for col in newdf_licensors.columns]  # Add a prefix to column names
if 'Unknown' in newdf_licensors.columns:
    newdf_licensors['Licensor_Unknown'] = newdf_licensors['Unknown']
    newdf_licensors.drop(columns=['Unknown'], inplace=True)
newdf = newdf.join(newdf_licensors)

# Create 'Studios' dummy variables
newdf_studios = newdf['Studios'].str.get_dummies(', ')
newdf_studios.columns = [f'Studio_{col}' for col in newdf_studios.columns]  # Add a prefix to column names
if 'Unknown' in newdf_studios.columns:
    newdf_studios['Studio_Unknown'] = newdf_studios['Unknown']
    newdf_studios.drop(columns=['Unknown'], inplace=True)
newdf = newdf.join(newdf_studios)

# Replace all occurrences of 1 in the 'Unknown' column with 0
newdf['Unknown'] = newdf['Unknown'].replace(1, 0)
```

3. Rescale Score and FMratio
4. Convert Data Types
5. Use Ordinal Encoder, Label Encoder to find accuracy (0.772)

In [8]:
```
# Instantiate the OrdinalEncoder
ordinal_encoder = OrdinalEncoder()

# Encode categorical columns using ordinal encoding
X_encoded = ordinal_encoder.fit_transform(newdf.drop(['Score', 'English','Producers','Studios', 'Licensors'], axis=1))

# Encode the 'Score' column using label encoding
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(newdf['Score'])

# Split the dataset into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y_encoded, test_size=0.2, random_state=1)

# Instantiate the RandomForestClassifier
rf_classifier = RandomForestClassifier(random_state=42)

# Fit the model to the training data
rf_classifier.fit(X_train, y_train)

# Predict on the test data
y_pred = rf_classifier.predict(X_test)

# Decode the predicted labels back to original score values
y_pred_decoded = label_encoder.inverse_transform(y_pred)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.771978021978022
```

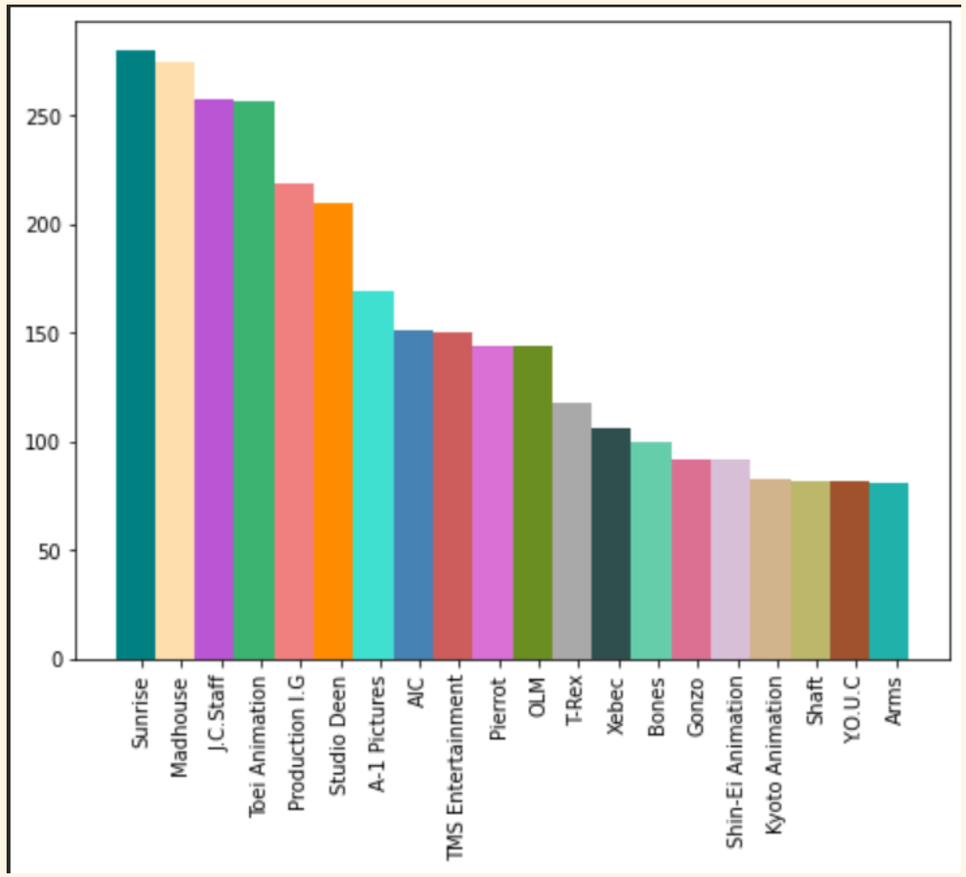
# 6. Using CatBoost from CatBoostRegressor

| RMSE using CatBoost | |
|---|---|
| RMSE on Train Set | 0.472 |
| RMSE on Test Set | 0.601 |

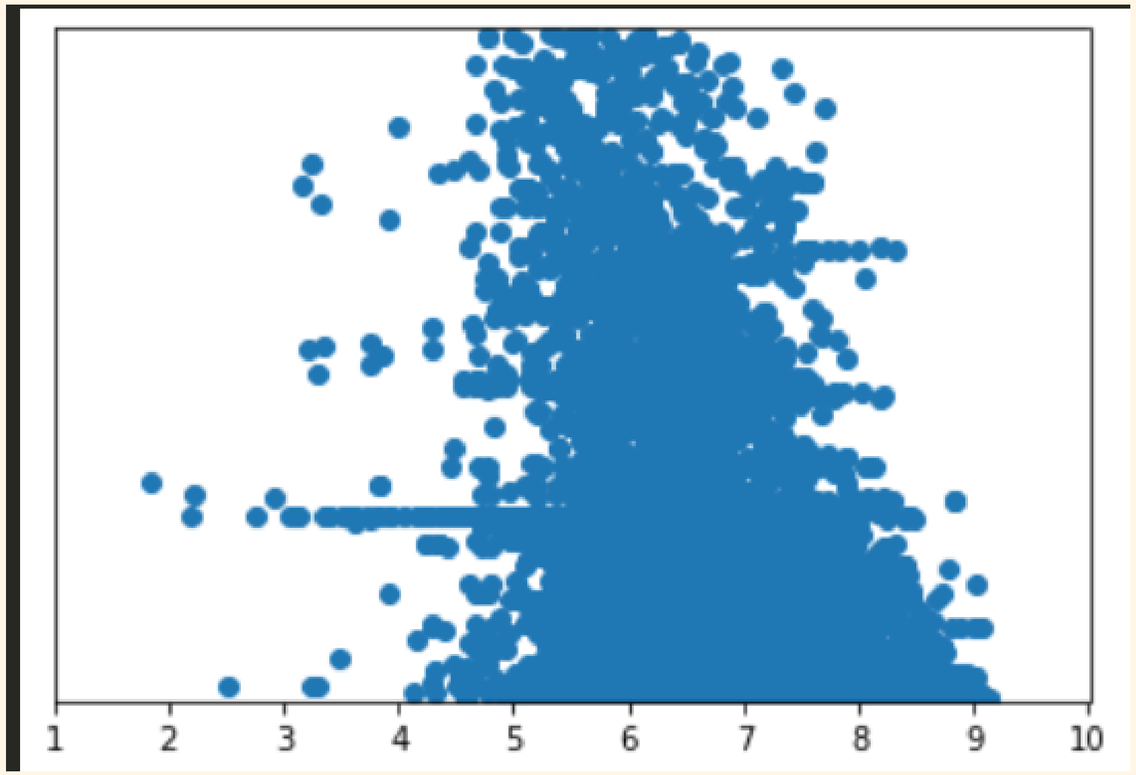
| Baseline RMSE | |
|---|---|
| RMSE on Train Set | 1.85 |
| RMSE on Test Set | 1.839 |

Compared to Baseline RMSE where data is randomly split, CatBoostRegressor explicitly takes into account categorical features, returning a much lower RMSE on our Train and Test Set

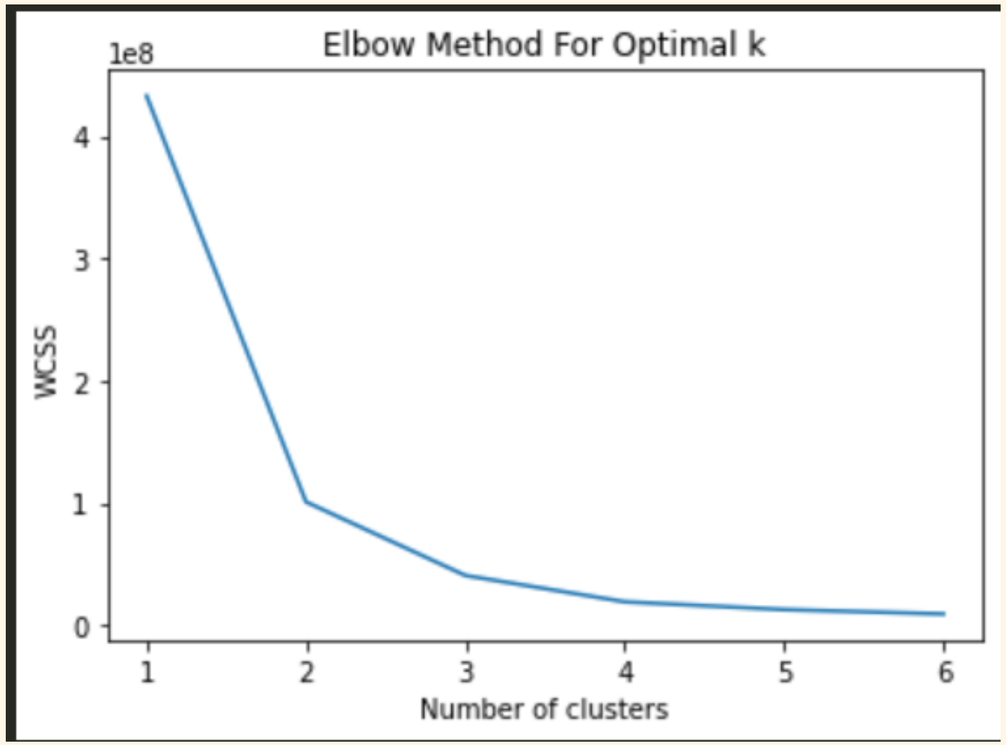
# K-Means Clustering



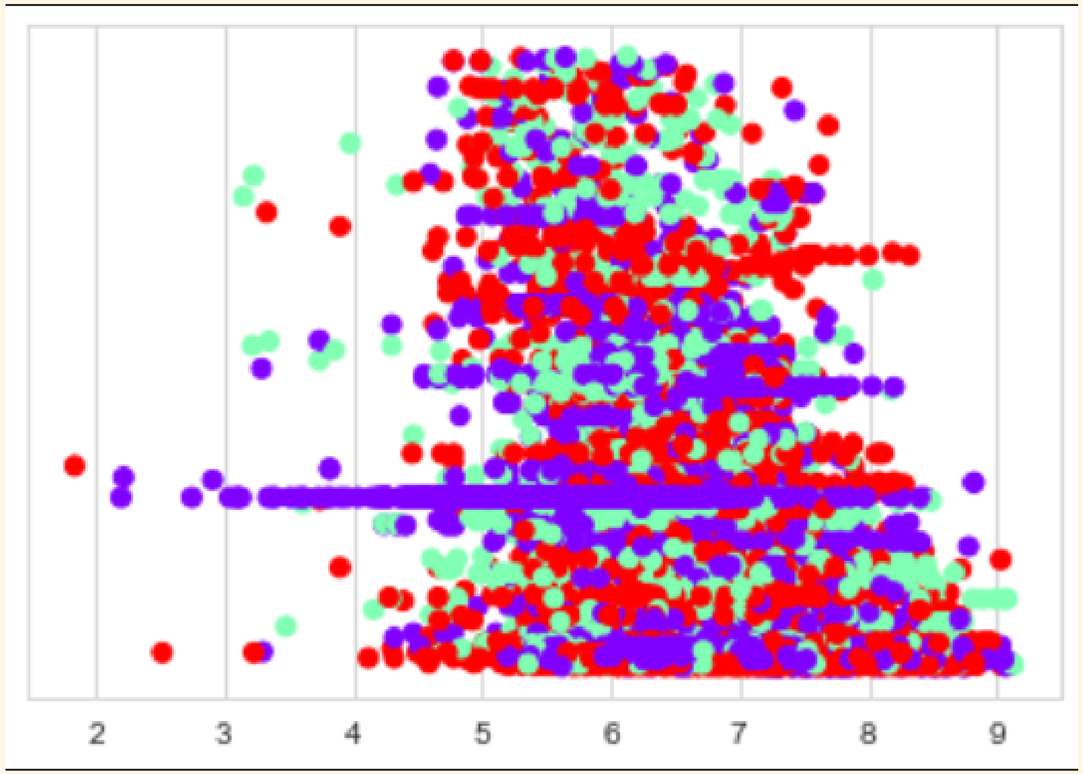

HISTOGRAM PLOT FOR TOP 20 STUDIOS




SCATTER PLOT OF Score vs. Studios




No. of Clusters = 4

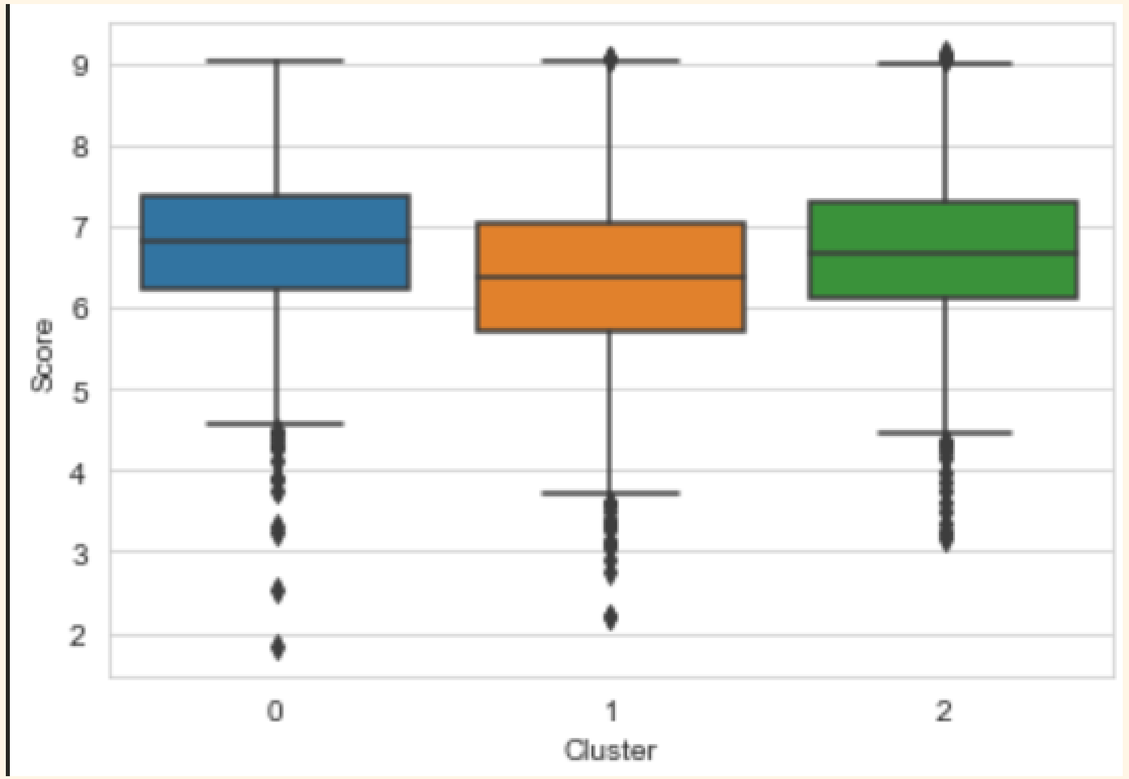


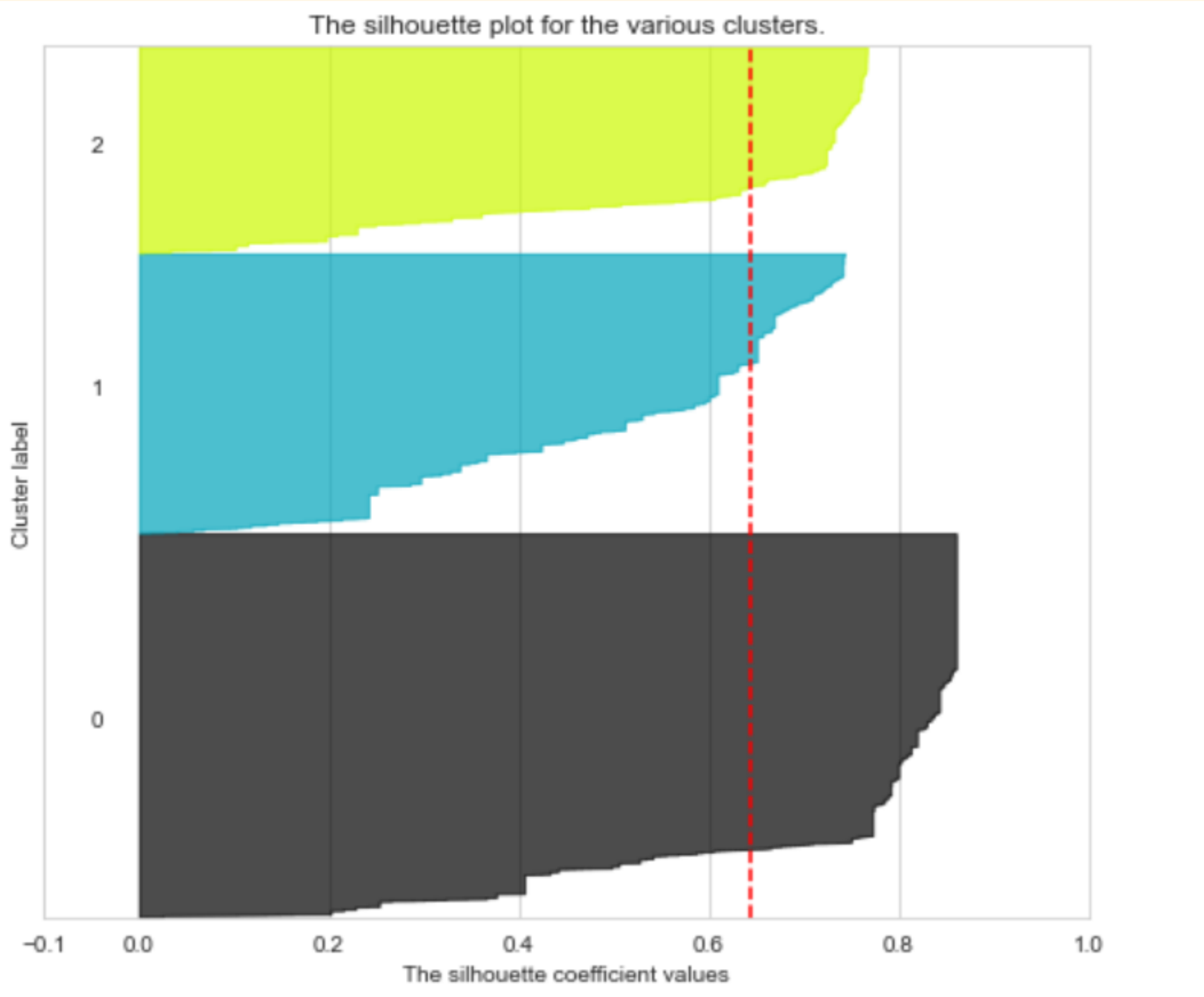

Cluster #0, #1, #2

| | Cluster | Cluster | Cluster | Cluster | Score |
|---|---|---|---|---|---|
| count | 4257.0 | 4257.000000 | 4257.000000 | 4257.0 | 4257.000000 |
| mean | 0.0 | 1.329340 | 1.329340 | 0.0 | 6.379537 |
| std | 0.0 | 0.741852 | 0.741852 | 0.0 | 0.920071 |
| min | 0.0 | 1.000000 | 1.000000 | 0.0 | 2.201000 |
| 25% | 0.0 | 1.000000 | 1.000000 | 0.0 | 5.721000 |
| 50% | 0.0 | 1.000000 | 1.000000 | 0.0 | 6.361000 |
| 75% | 0.0 | 1.000000 | 1.000000 | 0.0 | 7.051000 |
| max | 0.0 | 3.000000 | 3.000000 | 0.0 | 9.081000 |

| | Cluster | Cluster | Cluster | Cluster | Score |
|---|---|---|---|---|---|
| count | 3100.0 | 3100.000000 | 3100.000000 | 3100.0 | 3100.000000 |
| mean | 2.0 | 1.463226 | 1.463226 | 1.0 | 6.796961 |
| std | 0.0 | 1.499791 | 1.499791 | 0.0 | 0.842899 |
| min | 2.0 | 0.000000 | 0.000000 | 1.0 | 1.841000 |
| 25% | 2.0 | 0.000000 | 0.000000 | 1.0 | 6.241000 |
| 50% | 2.0 | 0.000000 | 0.000000 | 1.0 | 6.811000 |
| 75% | 2.0 | 3.000000 | 3.000000 | 1.0 | 7.361000 |
| max | 2.0 | 3.000000 | 3.000000 | 1.0 | 9.041000 |

| | Cluster | Cluster | Cluster | Cluster | Score |
|---|---|---|---|---|---|
| count | 2283.0 | 2283.000000 | 2283.000000 | 2283.0 | 2283.000000 |
| mean | 1.0 | 1.515550 | 1.515550 | 2.0 | 6.693282 |
| std | 0.0 | 0.857047 | 0.857047 | 0.0 | 0.842663 |
| min | 1.0 | 0.000000 | 0.000000 | 2.0 | 3.151000 |
| 25% | 1.0 | 2.000000 | 2.000000 | 2.0 | 6.131000 |
| 50% | 1.0 | 2.000000 | 2.000000 | 2.0 | 6.671000 |
| 75% | 1.0 | 2.000000 | 2.000000 | 2.0 | 7.291000 |
| max | 1.0 | 2.000000 | 2.000000 | 2.0 | 9.131000 |






FOR N_CLUSTERS = 3 THE AVERAGE SILHOUETTE_SCORE IS : 0.644988067399859

FOUR

# CONCLUSION

# WHAT WE LEARNED

- **Different Machine Learning (ML) techniques**
  - Random Forest Classifier
  - Multivariate Linear Regression
  - K-Means Clustering

- **Encoders**
  - One Hot Encoder
  - Label
  - Ordinal and Nominal Encoding

- **Data Imputation**
  - most-frequent (for categorical)
  - mean (for numerical)

# RECOMMENDATIONS

- Studios should produce more 'TV' anime and their corresponding specials to suit viewer's preferences
- Studios should also focus more on the quantity of anime, instead of quality
- Studios should also remain consistent in producing anime, in order to have consistent scores to suit viewer's preference for quantity
- Overall, studios should produce 'TV' anime consistently, at high quantity, to suit viewer's preferences, and thus increase profits.

# THANK YOU