SC1015 MINI-PROJECT

ANIME COMPLETION ANALYSIS

SC20 GROUP 7
Noel Wee Ee Cher (U2121856B)
Tharun Swaroop (U2122053A)

TABLE OF CONTENTS

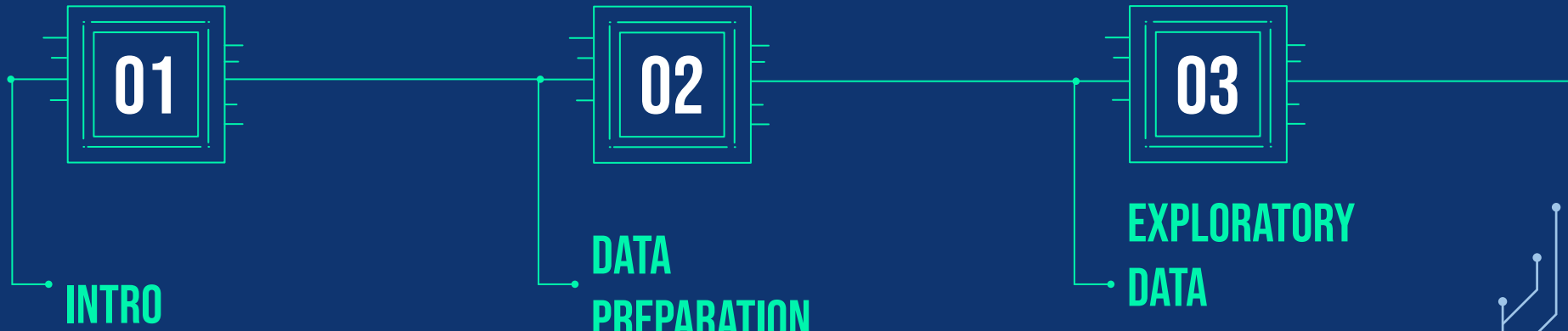
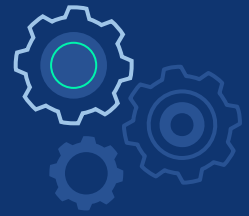


TABLE OF CONTENTS

01

INTRODUCTION



PROBLEM STATEMENT:

HOW LIKELY IS SOMEONE COMPLETING AN ANIME, BASED ON VARIOUS FACTORS?

DATASET:

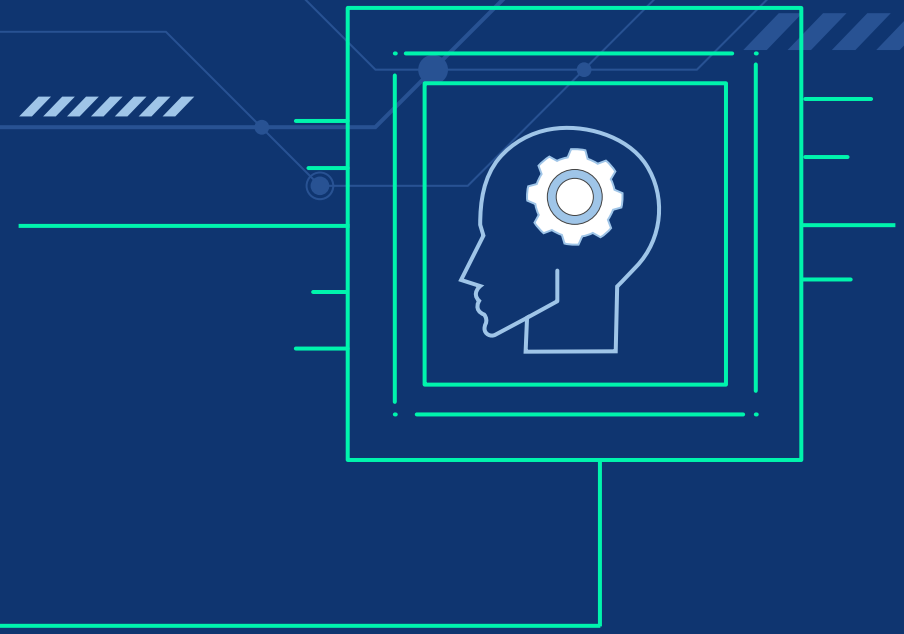
“ANIME RECOMMENDATIONS DATABASE” ON KAGGLE

HERNAN VALDIVIESO · UPDATED 9 MONTHS AGO

320

Anime Recommendation Database 2020

Recommendation data from 320.0000 users and 16.000 animes at myanimelist.net



Sample Collection	Data Preparation	Exploratory Analysis	Analytic Visualisation	Algorithmic Optimisation	Information Presentation

02

DATA PREPARATION

DATA COLLECTED

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17562 entries, 0 to 17561
Data columns (total 81 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   MAL_ID                  17562 non-null  int64
 1   Name                    17562 non-null  object
 2   Score                   17562 non-null  object
 3   Genres                  17562 non-null  object
 4   Genre 1                 17562 non-null  object
 5   Genre 2                 13270 non-null  object
 6   Genre 3                 9029 non-null   object
 7   Genre 4                 5534 non-null   object
 8   Genre 5                 2959 non-null   object
 9   Genre 6                 1237 non-null   object
 10  Genre 7                 439 non-null    object
 11  Genre 8                 143 non-null    object
 12  Genre 9                 58 non-null     object
 13  Genre 10                21 non-null     object
 14  Genre 11                7 non-null      object
 15  Genre 12                1 non-null      object
 16  Genre 13                1 non-null      object
 17  English name            17562 non-null  object
 18  Japanese name           17562 non-null  object
 19  Type                    17562 non-null  object
 20  Episodes                17562 non-null  object
 21  Aired                   17562 non-null  object
 22  Premiered               17562 non-null  object
 23  Producers               17562 non-null  object
 24  Producer 1              17562 non-null  object
 25  Producer 2              4037 non-null   object
 26  Producer 3              2163 non-null   object
 27  Producer 4              1418 non-null   object
 28  Producer 5              981 non-null    object
 29  Producer 6              645 non-null    object
 30  Producer 7              392 non-null    object
 31  Producer 8              237 non-null    object
 32  Producer 9              155 non-null    object
 33  Producer 10             75 non-null     object
 34  Producer 11             43 non-null     object
 35  Producer 12             23 non-null     object
 36  Producer 13             11 non-null     object
 37  Producer 14             7 non-null      object
 38  Producer 15             6 non-null      object
 39  Producer 16             5 non-null      object
 40  Producer 17             5 non-null      object
```

81 Columns, 17562 Rows

Varying number of non-NULL values

Some of the columns are irrelevant to our analysis

```
 41  Producer 18             3 non-null      object
 42  Producer 19             3 non-null      object
 43  Producer 20             1 non-null      object
 44  Licensors               17562 non-null  object
 45  Licensor 1              17562 non-null  object
 46  Licensor 2              756 non-null    object
 47  Licensor 3              51 non-null     object
 48  Licensor 4              5 non-null      object
 49  Studios                 17562 non-null  object
 50  Studio 1                17562 non-null  object
 51  Studio 2                759 non-null    object
 52  Studio 3                46 non-null     object
 53  Studio 4                4 non-null      object
 54  Studio 5                1 non-null      object
 55  Studio 6                1 non-null      object
 56  Studio 7                1 non-null      object
 57  Source                  17562 non-null  object
 58  Duration                17562 non-null  object
 59  Rating                  17562 non-null  object
 60  Ranked                  17562 non-null  object
 61  Popularity              17562 non-null  int64
 62  Members                 17562 non-null  int64
 63  Favorites               17562 non-null  int64
 64  Watching                17562 non-null  int64
 65  Completed               17562 non-null  int64
 66  On-Hold                 17562 non-null  int64
 67  Dropped                 17562 non-null  int64
 68  Plan to Watch           17562 non-null  int64
 69  Completion Percentage   17562 non-null  object
 70  Score-10                17562 non-null  object
 71  Score-9                 17562 non-null  object
 72  Score-8                 17562 non-null  object
 73  Score-7                 17562 non-null  object
 74  Score-6                 17562 non-null  object
 75  Score-5                 17562 non-null  object
 76  Score-4                 17562 non-null  object
 77  Score-3                 17562 non-null  object
 78  Score-2                 17562 non-null  object
 79  Score-1                 17562 non-null  object
 80  High Score Percentage   17562 non-null  object
dtypes: int64(9), object(72)
memory usage: 10.9+ MB
```

Sample Collection	Data Preparation	Exploratory Analysis	Analytic Visualisation	Algorithmic Optimisation	Information Presentation

DATA PREPARATION

- Dropped ‘Japanese name’, ‘Premiered’, ‘Members’, ‘Favourites’, ‘Popularity’, ‘Watching’, ‘Plan to Watch’, ‘Producers’ and ‘Licensors’ as they are irrelevant to our analysis

- We also removed these columns as they are repeated due to formatting differences: 'Genre2', 'Genre3', 'Genre4', 'Genre5', 'Genre6', 'Genre7', 'Genre8', 'Genre9', 'Genre10', 'Genre11', 'Genre12', 'Genre13', 'Studio2', 'Studio3', 'Studio4', 'Studio5', 'Studio6', 'Studio7'

```
#REMOVING UNWANTED COLUMNS
animeDF = animeDF.drop(columns = ['Japanese name', 'Premiered', 'Members', 'Favorites', 'Popularity',
                                  'Watching', 'Plan to Watch', 'Producers' ])
```

```
# Removing repeated columns
animeDF1 = animeDF.drop(columns = ['Genre2', 'Genre3', 'Genre4', 'Genre5', 'Genre6', 'Genre7', 'Genre8', 'Genre9',
                                   'Genre10', 'Genre11', 'Genre12', 'Genre13', 'Studio2', 'Studio3', 'Studio4',
                                   'Studio5', 'Studio6', 'Studio7'])
```

DATA PREPARATION

0	MAL_ID	17562 non-null	int64
1	Name	17562 non-null	object
2	Score	17562 non-null	object
3	Genres	17562 non-null	object
4	Genre 1	17562 non-null	object
5	Genre 2	13270 non-null	object
6	Genre 3	9029 non-null	object
7	Genre 4	5534 non-null	object
8	Genre 5	2959 non-null	object
9	Genre 6	1237 non-null	object
10	Genre 7	439 non-null	object
11	Genre 8	143 non-null	object
12	Genre 9	58 non-null	object
13	Genre 10	21 non-null	object
14	Genre 11	7 non-null	object
15	Genre 12	1 non-null	object
16	Genre 13	1 non-null	object
17	English name	17562 non-null	object
18	Type	17562 non-null	object
19	Episodes	17562 non-null	object
20	Aired	17562 non-null	object
21	Studios	17562 non-null	object
22	Studio 1	17562 non-null	object
23	Studio 2	759 non-null	object
24	Studio 3	46 non-null	object
25	Studio 4	4 non-null	object
26	Studio 5	1 non-null	object
27	Studio 6	1 non-null	object
28	Studio 7	1 non-null	object
29	Source	17562 non-null	object
30	Duration	17562 non-null	object

Reduced to 48 columns of data!

31	Rating	17562 non-null	object
32	Ranked	17562 non-null	object
33	Completed	17562 non-null	int64
34	On-Hold	17562 non-null	int64
35	Dropped	17562 non-null	int64
36	Completion Percentage	17562 non-null	object
37	Score-10	17562 non-null	object
38	Score-9	17562 non-null	object
39	Score-8	17562 non-null	object
40	Score-7	17562 non-null	object
41	Score-6	17562 non-null	object
42	Score-5	17562 non-null	object
43	Score-4	17562 non-null	object
44	Score-3	17562 non-null	object
45	Score-2	17562 non-null	object
46	Score-1	17562 non-null	object
47	High Score Percentage	17562 non-null	object

dtypes: int64(4), object(44)
memory usage: 6.4+ MB

DATA CLEANING

- Removed ‘Unknown’ and invalid values in the ‘Studios’, ‘Score’, ‘Genres’, ‘Source’, ‘Episodes’ and ‘Rating’ columns for EDA and further analysis.
- Removed ‘Rx - Hentai’ due to it’s high ‘Unknown’ Value count across all columns, as well as inappropriateness

```
StudiosDF = animeDF[animeDF['Studio 1'] != 'Unknown'] #REMOVING UNKNOWN VALUES IN STUDIOS
EpisodesDF = animeDF[animeDF['Episodes'] != 'Unknown'] #REMOVING 'UNKNOWN' VALUES IN EPISODES
ScoreDF = animeDF[animeDF['Score'] != 'Unknown'] #REMOVING 'UNKNOWN' VALUES FOR SCORE
StudiosDF1 = StudiosDF[StudiosDF['Rating'] != 'Rx - Hentai'] #REMOVING HENTAI IN RATINGS
animeDFClean = animeDataUpdated[animeDataUpdated['Source'] != 'Unknown']
animeDFClean1 = animeDFClean[animeDFClean['Completion Percentage'] != '#DIV/0!']
animeDFClean = animeDataUpdated[animeDataUpdated['Genres'] != 'Unknown']
animeDFClean2 = animeDFClean1[animeDFClean1['High Score Percentage'] != '#VALUE!']
```

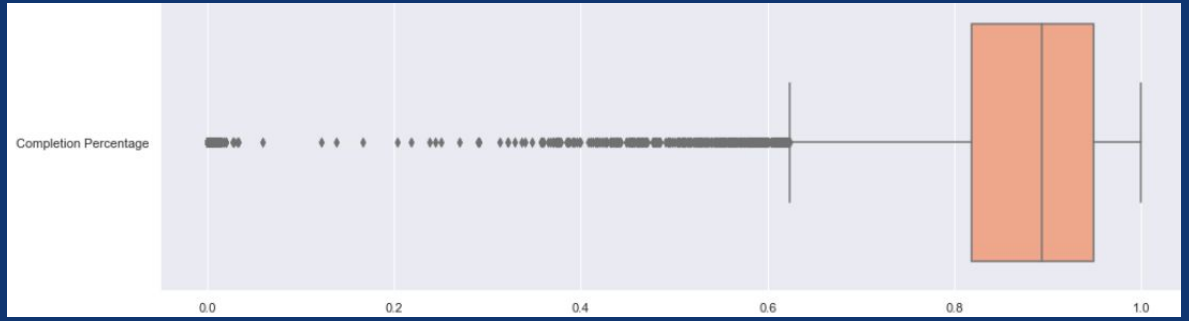
03

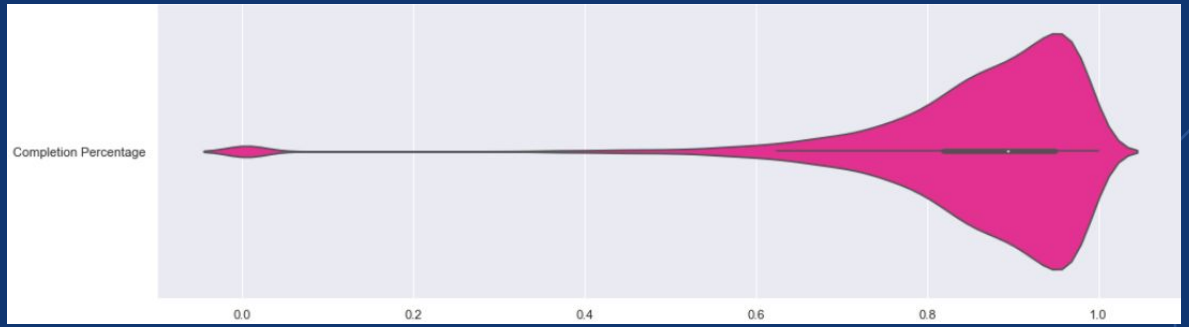
EXPLORATORY DATA ANALYSIS

COMPLETION PERCENTAGE

DISTRIBUTION OF COMPLETION PERCENTAGE ACROSS ALL ANIMES

	Completion Percentage
count	12116.000000
mean	0.856998
std	0.147988
min	0.000000
25%	0.818558
50%	0.893870
75%	0.948551
max	1.000000





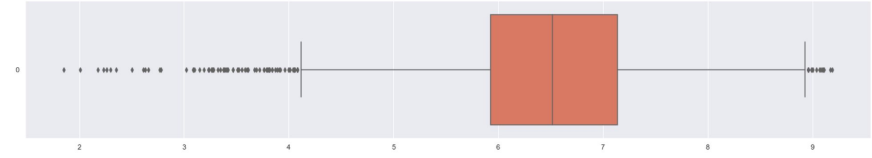
SCORE:

DISTRIBUTION OF SCORES ACROSS ALL ANIMES

```
count    12421.000000
mean         6.509999
std          0.886717
min          1.850000
25%          5.930000
50%          6.520000
75%          7.140000
max          9.190000
Name: Score, dtype: float64
```

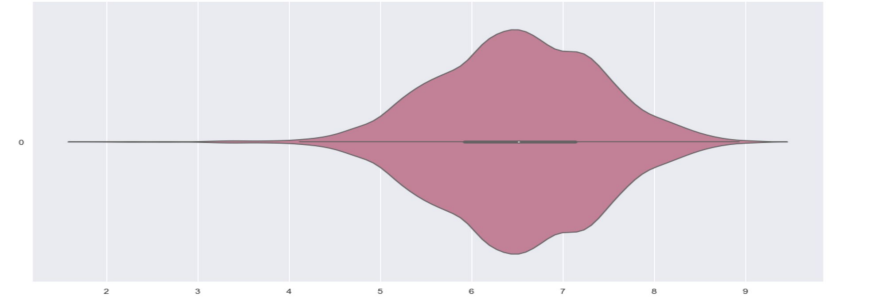
```
#BOX PLOT FOR SCORES

f = plt.figure(figsize=(24, 4))
sb.boxplot(data = Score['Score'], orient = "h", color = 'tomato')
```

```
<AxesSubplot:>
```



```
#VIOLIN PLOT FOR SCORES

f = plt.figure(figsize=(16, 8))
sb.violinplot(data = Score['Score'], orient = "h", color = 'palevioletred')
```

```
<AxesSubplot:>
```



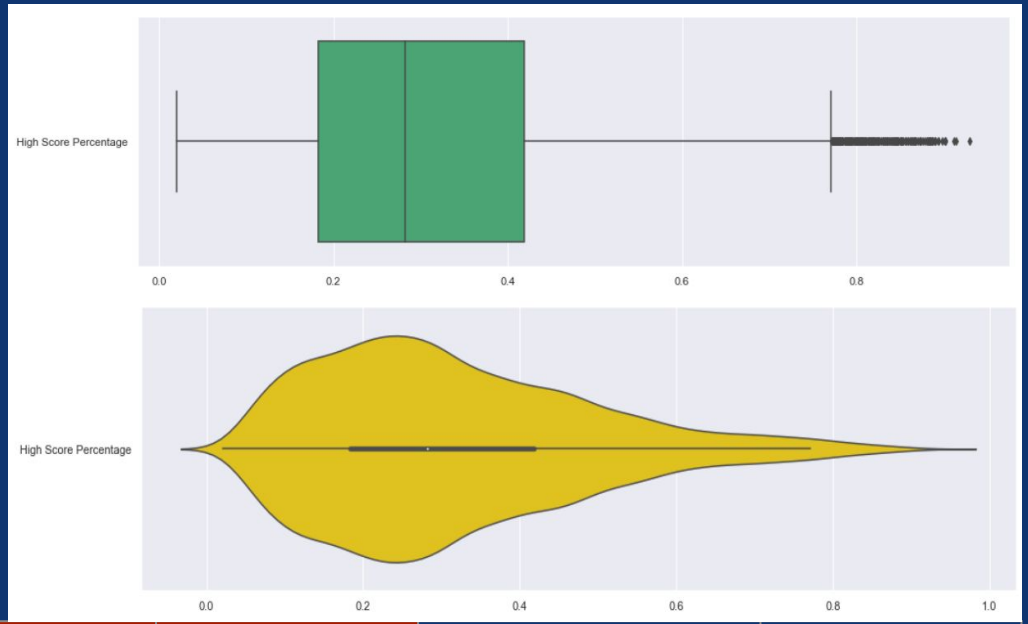
Sample Collection	Data Preparation	Exploratory Analysis	Analytic Visualisation	Algorithmic Optimisation	Information Presentation

HIGH SCORE PERCENTAGE:

DEFINITION: PERCENTAGE OF SCORES THAT ARE 8 AND ABOVE

DISTRIBUTION OF HIGH SCORE PERCENTAGE ACROSS ALL ANIME

	High Score Percentage
count	12116.000000
mean	0.313720
std	0.174705
min	0.020408
25%	0.182823
50%	0.282094
75%	0.418386
max	0.929694



Sample Collection	Data Preparation	Exploratory Analysis	Analytic Visualisation	Algorithmic Optimisation	Information Presentation

STUDIOS:

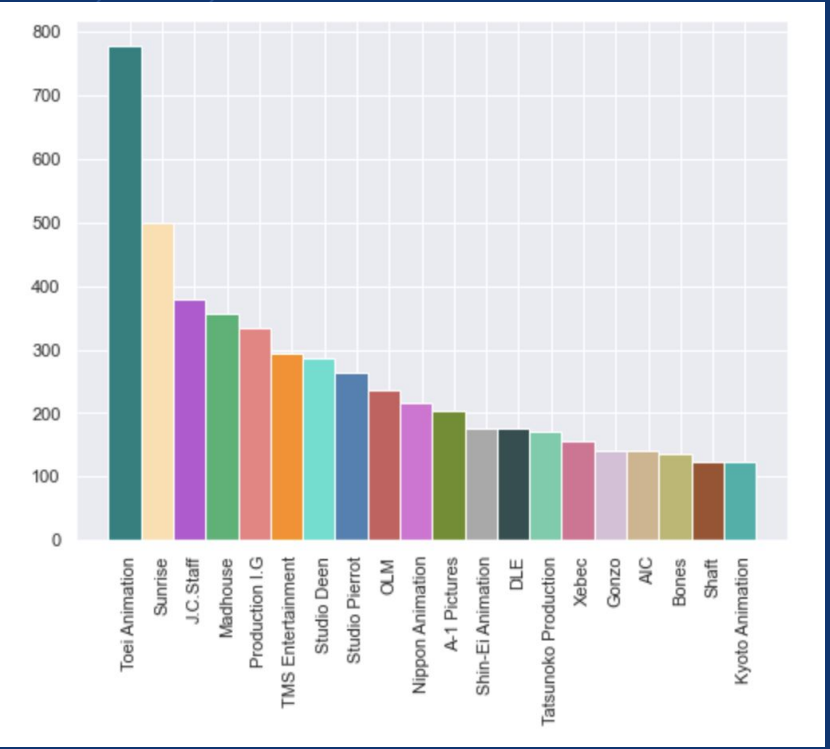
WHICH STUDIOS PRODUCED THE MOST ANIME?

```
#GETTING THE TOP 20 STUDIOS IN TERMS OF NO. OF ANIME PRODUCED

n = 20
from operator import itemgetter
Top20_Studios = dict(sorted(Top20.items(), key = itemgetter(1), reverse = True)[:n])
print(Top20_Studios)
```

```
{'Toei Animation': 778, 'Sunrise': 499, 'J.C.Staff': 380, 'Madhouse': 357, 'Production I.G': 334, 'TMS Entertainmen
t': 294, 'Studio Deen': 287, 'Studio Pierrot': 263, 'OLM': 237, 'Nippon Animation': 216, 'A-1 Pictures': 204, 'Shin
-Ei Animation': 177, 'DLE': 175, 'Tatsunoko Production': 170, 'Xebec': 155, 'Gonzo': 140, 'AIC': 140, 'Bones': 137,
'Shaft': 124, 'Kyoto Animation': 123}
```

STUDIOS:

WHICH STUDIOS PRODUCED THE MOST ANIME?



	Count
Toei Animation	778
Sunrise	499
J.C.Staff	380
Madhouse	357
Production I.G	334
TMS Entertainment	294
Studio Deen	287
Studio Pierrot	263
OLM	237
Nippon Animation	216
A-1 Pictures	204
Shin-Ei Animation	177
DLE	175
Tatsunoko Production	170
Xebec	155
Gonzo	140
AIC	140
Bones	137
Shaft	124
Kyoto Animation	123

GENRES:

FINDING TOP 5 GENRES (IN TERMS OF FREQUENCY) IN EACH GENRE COLUMN

```
Top 5 Genres for Genre 1: {'Action': 3751, 'Comedy': 3067, 'Adventure': 1456, 'Music': 1424, 'Kids': 1078}
Top 5 Genres for Genre 2: {'Comedy': 1907, 'Adventure': 1207, 'Kids': 936, 'Fantasy': 894, 'Drama': 808}
Top 5 Genres for Genre 3: {'Fantasy': 869, 'Comedy': 750, 'Romance': 574, 'Sci-Fi': 529, 'School': 506}
Top 5 Genres for Genre 4: {'Shounen': 542, 'Fantasy': 465, 'Romance': 394, 'Sci-Fi': 343, 'School': 336}
Top 5 Genres for Genre 5: {'Shounen': 470, 'Fantasy': 253, 'Sci-Fi': 217, 'Supernatural': 192, 'School': 181}
Top 5 Genres for Genre 6: {'Shounen': 252, 'Fantasy': 113, 'Supernatural': 98, 'Sci-Fi': 84, 'Romance': 64}
Top 5 Genres for Genre 7: {'Shounen': 109, 'Supernatural': 30, 'School': 27, 'Super Power': 25, 'Fantasy': 24}
Top 5 Genres for Genre 8: {'Shounen': 28, 'Fantasy': 16, 'Romance': 10, 'School': 9, 'Space': 8}
Top 5 Genres for Genre 9: {'Supernatural': 11, 'School': 8, 'Romance': 8, 'Shounen': 6, 'Seinen': 6}
Top 5 Genres for Genre 10: {'Supernatural': 7, 'Shounen': 6, 'Fantasy': 4, 'School': 1, 'Mecha': 1}
Top 5 Genres for Genre 11: {'Shounen': 5, 'School': 1, 'Supernatural': 1}
Top 5 Genres for Genre 12: {'Drama': 1}
Top 5 Genres for Genre 13: {'Sci-Fi': 1}
```

GENRES:

```
n = 5
from operator import itemgetter
top5Genres = dict(sorted(top5.items(), key = itemgetter(1), reverse = True)[:n])

print('The top 5 genres are:', top5Genres)
```

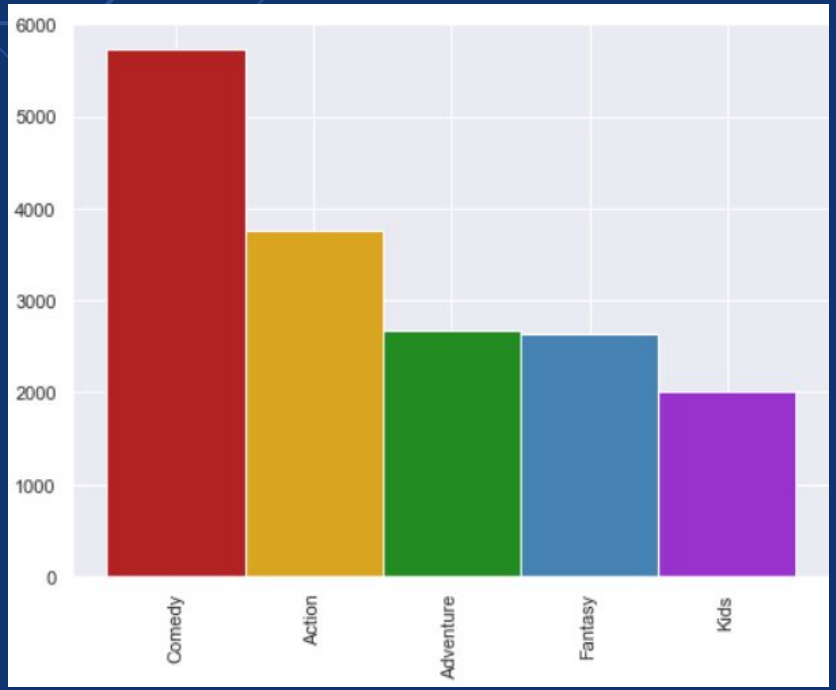
```
The top 5 genres are: {'Comedy': 5724, 'Action': 3751, 'Adventure': 2663, 'Fantasy': 2638, 'Kids': 2014}
```

The top 5 genres (in terms of frequency) in the entire dataset are:

1) Comedy
2) Action
3) Adventure
4) Fantasy
5) Kids

GENRES:

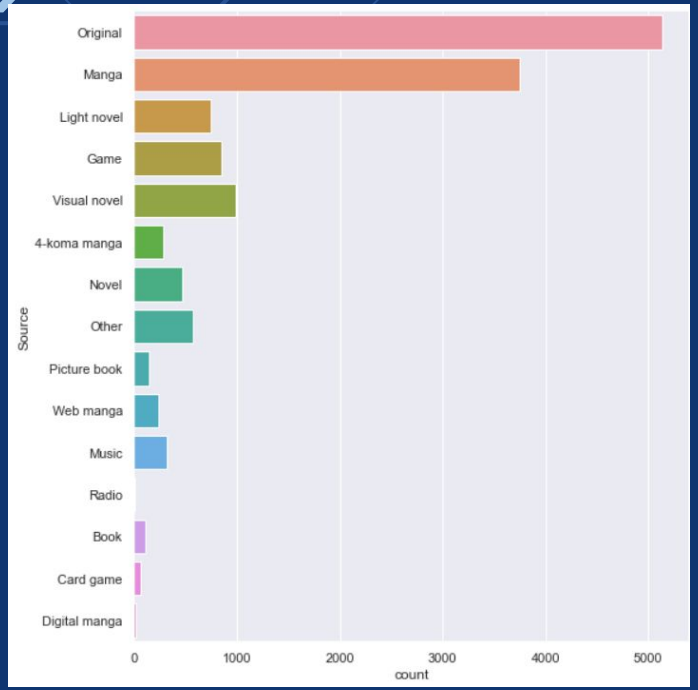
HISTOGRAM FOR THE TOP 5 GENRES



	Count
Comedy	5724
Action	3751
Adventure	2663
Fantasy	2638
Kids	2014

SOURCE:

COUNTPLOT FOR SOURCES

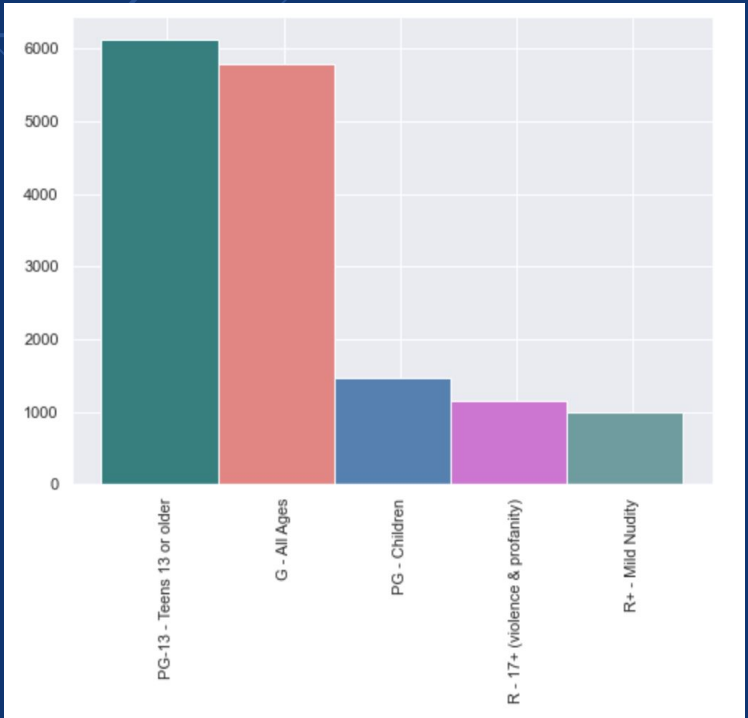


Original	5136
Manga	3752
Visual novel	985
Game	853
Light novel	742
Other	579
Novel	473
Music	314
4-koma manga	284
Web manga	240
Picture book	144
Book	111
Card game	64
Digital manga	15
Radio	12

Name: Source, dtype: int64

AGE-RATINGS:

HISTOGRAM PLOT FOR FREQUENCY OF AGE-RATINGS ACROSS ALL ANIME



	Count
PG-13 - Teens 13 or older	6132
G - All Ages	5782
PG - Children	1461
R - 17+ (violence & profanity)	1157
R+ - Mild Nudity	997

EPISODES:

DISTRIBUTION OF THE NUMBER OF EPISODES ACROSS ALL ANIME

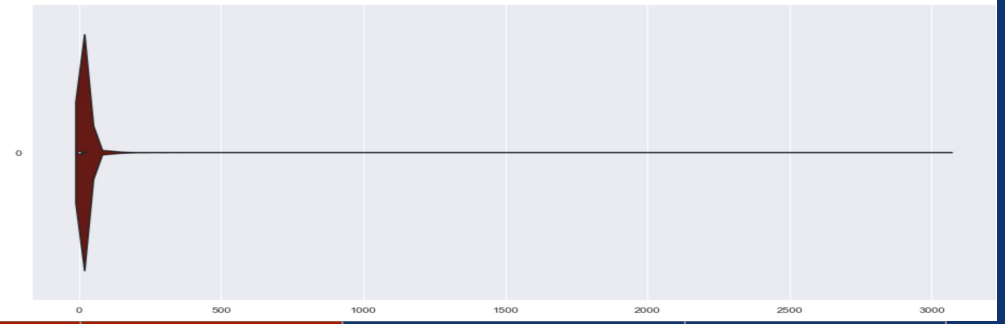
```
count    17046.000000
mean        11.525519
std         47.348640
min          1.000000
25%          1.000000
50%          2.000000
75%         12.000000
max       3057.000000
Name: Episodes, dtype: float64
```

```
#BOX PLOT FOR EPISODES

f = plt.figure(figsize=(24, 4))
sb.boxplot(data = EpisodesDF['Episodes'], orient = "h", color = 'teal')
```

```
<AxesSubplot:>
```


```
#VIOLIN PLOT FOR EPISODES

f = plt.figure(figsize=(16, 8))
sb.violinplot(data = EpisodesDF['Episodes'], orient = "h", color = 'maroon')
```

```
<AxesSubplot:>
```



Sample Collection	Data Preparation	Exploratory Analysis	Analytic Visualisation	Algorithmic Optimisation	Information Presentation

04

MACHINE LEARNING

MACHINE LEARNING:

- Problem Statement: **How likely is someone to complete an Anime, based on various factors?**
- Predictors: Source, Genres, Ratings, Studios and High Score Percentage
  - Did not include Episodes due to its drastic outliers and very low Median
  - Used High Score Percentage instead of Score - curious on their relationship
- Response Variable: Completion Percentage
- Models used:
  - High Score Percentage (Numeric) - Linear Regression, Polynomial Regression
  - Genres, Studios and Source (Categorical) - K-Means
  - Ratings (Categorical) - K-Modes

Sample Collection	Data Preparation	Exploratory Analysis	Analytic Visualisation	Algorithmic Optimisation	Information Presentation

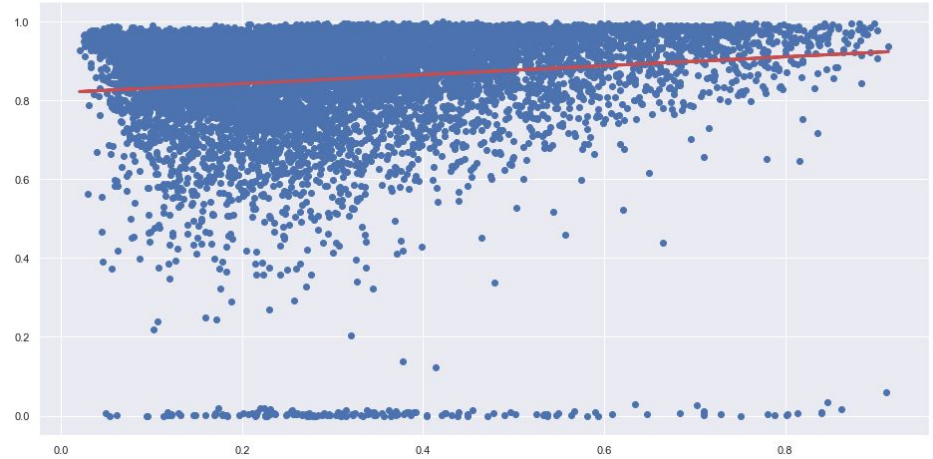
LET'S TAKE A LOOK AT HIGH SCORE PERCENTAGE...

- Standard Linear Regression:
  - X → High Score Percentage (Predictor)
  - Y → Completion Percentage (Response)
- Application of learnt concepts:
  - Using R-squared as a means of regression accuracy analysis primarily
  - Splitting the data into train and test sets
  - Using Scikit's Linear Regression model to fit the best linear regression model

Sample Collection	Data Preparation	Exploratory Analysis	Analytic Visualisation	Algorithmic Optimisation	Information Presentation

LINEAR REGRESSION OF CLEANED DATA

```
# Formula for the Regression Line
regline_highScorePrc = highScorePrc_train
regline_completePrc = linreg.intercept_ + linreg.coef_ * highScorePrc_train

# Plot the Linear Regression Line
f = plt.figure(figsize=(16, 8))
plt.scatter(highScorePrc_train, completePrc_train)
plt.plot(regline_highScorePrc, regline_completePrc, 'r-', linewidth = 3)
plt.show()
```



OBSERVATIONS

- Some outliers at the bottom
- Many points bunched at the top

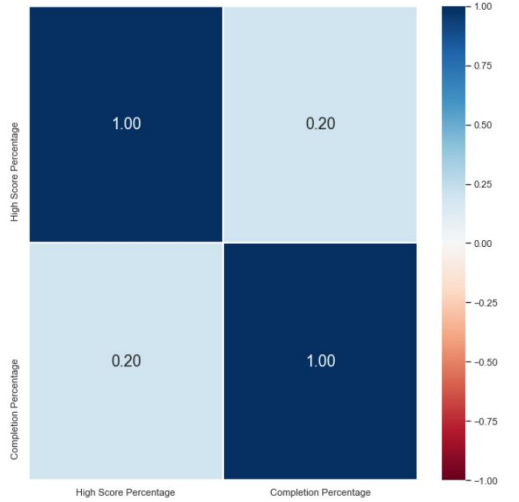
```
# Coefficients for the regression line
print('Intercept \t: b = ', linreg.intercept_)
print('Coefficients \t: a = ', linreg.coef_)
```

```
Intercept       : b =  [0.81989257]
Coefficients    : a =  [[0.11332813]]
```

CORRELATION BETWEEN HIGH SCORE PERCENTAGE AND COMPLETION PERCENTAGE

	High Score Percentage	Completion Percentage
High Score Percentage	1.00000	0.19861
Completion Percentage	0.19861	1.00000

Out[24]: <AxesSubplot:>



Low correlation coefficient of 0.19861

Sample Collection	Data Preparation	Exploratory Analysis	Analytic Visualisation	Algorithmic Optimisation	Information Presentation

LINEAR REGRESSION OF CLEANED DATA

```
# Check the Goodness of Fit (on Train Data)
print("Goodness of Fit of Model \tTrain Dataset")
print("Explained Variance (R^2) \t:", linreg.score(highScorePrc_train, completePrc_train))
print("Mean Squared Error (MSE) \t:", mean_squared_error(completePrc_train, completePrc_train_pred))
print("Root Mean Squared Error (RMSE) \t:", np.sqrt(mean_squared_error(completePrc_train, completePrc_train_pred)))
print()

# Check the Goodness of Fit (on Test Data)
print("Goodness of Fit of Model \tTest Dataset")
print("Explained Variance (R^2) \t:", linreg.score(highScorePrc_test, completePrc_test))
print("Mean Squared Error (MSE) \t:", mean_squared_error(completePrc_test, completePrc_test_pred))
print("Root Mean Squared Error (RMSE) \t:", np.sqrt(mean_squared_error(completePrc_test, completePrc_test_pred)))
print()
```

```
Goodness of Fit of Model        Train Dataset
Explained Variance (R^2)        : 0.016538920426774517
Mean Squared Error (MSE)        : 0.021218490542500246
Root Mean Squared Error (RMSE)  : 0.14566568072988312

Goodness of Fit of Model        Test Dataset
Explained Variance (R^2)        : 0.023639737825689844
Mean Squared Error (MSE)        : 0.022642602863277447
Root Mean Squared Error (RMSE)  : 0.15047459208543298
```

Very low R^2 value obtained for both train and test sets using linear regression

Sample Collection	Data Preparation	Exploratory Analysis	Analytic Visualisation	Algorithmic Optimisation	Information Presentation

AFTER REMOVAL OF OUTLIERS...

STATS OF COMPLETION PERCENTAGE

```
# Stats for Completion Percentage
completePrc_Q3 = completePrc.quantile(0.75)
print("Q3 of completePrc is: ", completePrc_Q3)
completePrc_Q1 = completePrc.quantile(0.25)
print("Q1 of completePrc is: ", completePrc_Q1)
completePrc_IQR = completePrc_Q3 - completePrc_Q1
print("IQR of completePrc is: ", completePrc_IQR)
print()

completePrc_lowerLimit = completePrc_Q1 - (1.5 * completePrc_IQR)
print("Lower limit of completePrc is: ", completePrc_lowerLimit)
completePrc_upperLimit = completePrc_Q3 + (1.5 * completePrc_IQR)
print("Upper limit of completePrc is: ", completePrc_upperLimit)

# Column to find out whether they are the outliers
completePrc_outliers = (completePrc < completePrc_lowerLimit) | (completePrc > completePrc_upperLimit)
completePrc_outliers
```

```
Q3 of completePrc is:  Completion Percentage    0.948551
Name: 0.75, dtype: float64
Q1 of completePrc is:  Completion Percentage    0.818558
Name: 0.25, dtype: float64
IQR of completePrc is:  Completion Percentage    0.129992
dtype: float64

Lower limit of completePrc is:  Completion Percentage    0.62357
dtype: float64
Upper limit of completePrc is:  Completion Percentage    1.14354
dtype: float64
```

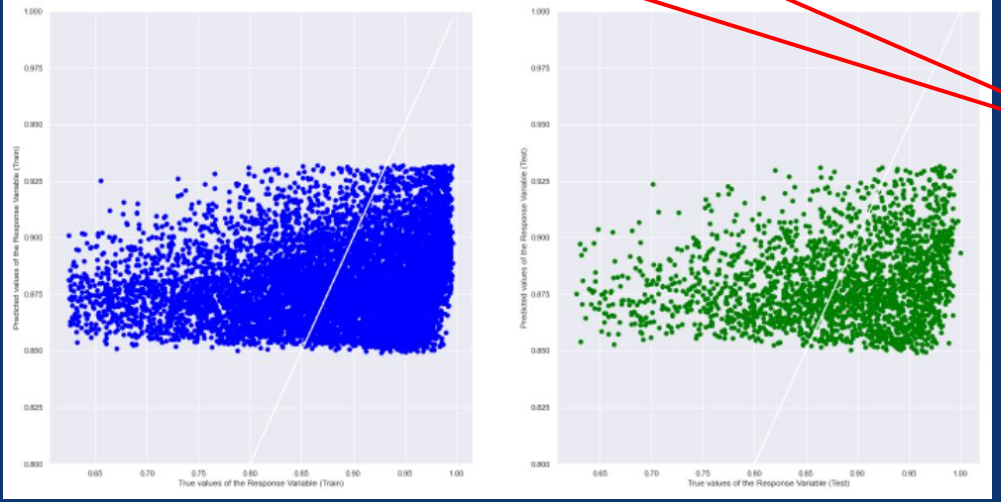
STATS OF HIGH SCORE PERCENTAGE

```
# Stats for High Score Percentage
highScorePrc_Q3 = highScorePrc.quantile(0.75)
print("Q3 of highScorePrc is: ", highScorePrc_Q3)
highScorePrc_Q1 = highScorePrc.quantile(0.25)
print("Q1 of highScorePrc is: ", highScorePrc_Q1)
highScorePrc_IQR = highScorePrc_Q3 - highScorePrc_Q1
print("IQR of highScorePrc is: ", highScorePrc_IQR)
print()

highScorePrc_lowerLimit = highScorePrc_Q1 - (1.5 * highScorePrc_IQR)
print("Lower limit of highScorePrc is: ", highScorePrc_lowerLimit)
highScorePrc_upperLimit = highScorePrc_Q3 + (1.5 * highScorePrc_IQR)
print("Upper limit of highScorePrc is: ", highScorePrc_upperLimit)

# Column to find out whether they are the outliers
highScorePrc_outliers = (highScorePrc < highScorePrc_lowerLimit) | (highScorePrc > highScorePrc_upperLimit)
highScorePrc_outliers
```

```
Q3 of highScorePrc is:  High Score Percentage    0.418386
Name: 0.75, dtype: float64
Q1 of highScorePrc is:  High Score Percentage    0.182823
Name: 0.25, dtype: float64
IQR of highScorePrc is:  High Score Percentage    0.235563
dtype: float64

Lower limit of highScorePrc is:  High Score Percentage   -0.170522
dtype: float64
Upper limit of highScorePrc is:  High Score Percentage    0.77173
dtype: float64
```

LINEAR REGRESSION OF OUTLIER-FREE DATA

```
Intercept of Regression          : d =  [0.84676808]
Coefficients of Regression       : c =  [[0.11058264]]

Goodness of Fit of Model         Train Dataset
Explained Variance (R^2)         : 0.045674718637584655
Mean Squared Error (MSE)         : 0.006860859999962649
Root Mean Squared Error (RMSE)   : 0.08286458797220106

Goodness of Fit of Model         Test Dataset
Explained Variance (R^2)         : 0.029041025270465926
Mean Squared Error (MSE)         : 0.006473529510756738
Root Mean Squared Error (RMSE)   : 0.0804519889111309
```



Very low R^2 value obtained for both train and test sets even after removing outliers, although there is an increase

Sample Collection	Data Preparation	Exploratory Analysis	Analytic Visualisation	Algorithmic Optimisation	Information Presentation

INTRODUCING POLYNOMIAL REGRESSION

- Using n-th degree polynomial to estimate the relationship between high score percentage and completion percentage
- Overcome the assumption that the relationship between high score percentage and completion percentage is linear in nature, hence yielding a more accurate regression model.

Sample Collection	Data Preparation	Exploratory Analysis	Analytic Visualisation	Algorithmic Optimisation	Information Presentation

POLYNOMIAL REGRESSION ON OUTLIER-FREE DATA

DEGREE 2 POLYNOMIAL REGRESSION

```
poly_features = PolynomialFeatures(degree = 2, include_bias = False)
highScorePrc_poly = poly_features.fit_transform(highScorePrcClean)
highScorePrc_poly[2]
```

```
array([0.35167989, 0.12367875])
```

```
lin_reg = LinearRegression()
lin_reg.fit(highScorePrc_poly, completePrcClean)
print('Coefficients of highScorePrc_poly are', lin_reg.coef_)
print('Intercept is', lin_reg.intercept_)
```

```
Coefficients of highScorePrc_poly are [[-0.06594957  0.24062548]]
Intercept is [0.87207617]
```

```
# model evaluation
completePrcDeg = lin_reg.predict(highScorePrc_poly)
mse = mean_squared_error(completePrcClean, completePrcDeg)

rmse = np.sqrt(mean_squared_error(completePrcClean, completePrcDeg))
r2 = r2_score(completePrcClean, completePrcDeg)


print('MSE of Polyregression model: ', mse)
print('RMSE of Polyregression Model: ', rmse)
print('R2 score of Linear model: ', r2)
```

```
MSE of Polyregression model:  0.006729073253279862
RMSE of Polyregression Model:  0.08203092863840969
R2 score of Linear model:  0.050828350361669994
```

DEGREE 3 POLYNOMIAL REGRESSION

```
poly_features = PolynomialFeatures(degree = 3, include_bias = False)
highScorePrc_poly = poly_features.fit_transform(highScorePrcClean)
highScorePrc_poly[2]
```

```
array([0.35167989, 0.12367875, 0.04349533])
```

```
lin_reg = LinearRegression()
lin_reg.fit(highScorePrc_poly, completePrcClean)
print('Coefficients of highScorePrc_poly are', lin_reg.coef_)
print('Intercept is', lin_reg.intercept_)
```

```
Coefficients of highScorePrc_poly are [[-0.58072343  1.8657202  -1.43756512]]
Intercept is [0.91437648]
```

```
# model evaluation
completePrcDeg = lin_reg.predict(highScorePrc_poly)
mse = mean_squared_error(completePrcClean, completePrcDeg)

rmse = np.sqrt(mean_squared_error(completePrcClean, completePrcDeg))
r2 = r2_score(completePrcClean, completePrcDeg)


print('MSE of Polyregression model: ', mse)
print('RMSE of Polyregression Model: ', rmse)
print('R2 score of Linear model: ', r2)
```

```
MSE of Polyregression model:  0.006654164793573544
RMSE of Polyregression Model:  0.08157306414235979
R2 score of Linear model:  0.06139458787983876
```

POLYNOMIAL REGRESSION ON OUTLIER-FREE DATA

DEGREE 5 POLYNOMIAL REGRESSION

```
poly_features = PolynomialFeatures(degree = 5, include_bias = False)
highScorePrc_poly = poly_features.fit_transform(highScorePrcClean)
highScorePrc_poly[2]
```

```
array([0.35167989, 0.12367875, 0.04349533, 0.01529643, 0.00537945])
```

```
lin_reg = LinearRegression()
lin_reg.fit(highScorePrc_poly, completePrcClean)
print('Coefficients of highScorePrc_poly are', lin_reg.coef_)
print('Intercept is', lin_reg.intercept_)
```

```
Coefficients of highScorePrc_poly are [[ -1.74350627   8.73403622 -18.08969995  17.06807838  -5.93302089]]
Intercept is [0.97516037]
```

```
# model evaluation
completePrcDeg = lin_reg.predict(highScorePrc_poly)
mse = mean_squared_error(completePrcClean, completePrcDeg)

rmse = np.sqrt(mean_squared_error(completePrcClean, completePrcDeg))
r2 = r2_score(completePrcClean, completePrcDeg)


print('MSE of Polyregression model: ', mse)
print('RMSE of Polyregression Model: ', rmse)
print('R2 score of Linear model: ', r2)
```

```
MSE of Polyregression model:  0.0066155354696909845
RMSE of Polyregression Model:  0.08133594205325825
R2 score of Linear model:  0.06684345991524887
```

DEGREE 10 POLYNOMIAL REGRESSION

```
poly_features = PolynomialFeatures(degree = 10, include_bias = False)
highScorePrc_poly = poly_features.fit_transform(highScorePrcClean)
highScorePrc_poly[2]
```

```
array([3.51679892e-01, 1.23678746e-01, 4.34953282e-02, 1.52964323e-02,
       5.37944767e-03, 1.89184357e-03, 6.65323344e-04, 2.33980842e-04,
       8.22863571e-05, 2.89384572e-05])
```

```
lin_reg = LinearRegression()
lin_reg.fit(highScorePrc_poly, completePrcClean)
print('Coefficients of highScorePrc_poly are', lin_reg.coef_)
print('Intercept is', lin_reg.intercept_)
```

```
Coefficients of highScorePrc_poly are [[-3.58959846e+00  4.05313419e+01 -3.10166728e+02  1.64351904e+03
  -5.82468644e+03  1.36410121e+04 -2.07726840e+04  1.97204789e+04
  -1.05664608e+04  2.43266636e+03]]
Intercept is [1.01678939]
```

```
# model evaluation
completePrcDeg = lin_reg.predict(highScorePrc_poly)
mse = mean_squared_error(completePrcClean, completePrcDeg)

rmse = np.sqrt(mean_squared_error(completePrcClean, completePrcDeg))
r2 = r2_score(completePrcClean, completePrcDeg)


print('MSE of Polyregression model: ', mse)
print('RMSE of Polyregression Model: ', rmse)
print('R2 score of Linear model: ', r2)
```

```
MSE of Polyregression model:  0.00661499502615085
RMSE of Polyregression Model:  0.08133261969315171
R2 score of Linear model:  0.06691969235725359
```

OBSERVATIONS FROM POLYNOMIAL REGRESSION

- Although the R^2 increased, it still remains very low, even after polynomial regression. The R^2s are:
  - Degree 2: 0.0508...
  - Degree 3: 0.0614...
  - Degree 5: 0.0668...
  - Degree 10: 0.0669...
- High score percentage not a good indicator of completion percentage.
- Significant skewness in both completion percentage and high score percentage
  - Might be the reason to poor R^2 score and hence poor relation between them
- Let's fix it and see whether the results improve.

Sample Collection	Data Preparation	Exploratory Analysis	Analytic Visualisation	Algorithmic Optimisation	Information Presentation

SKEWNESS ANALYSIS

- CHECKING SKEWNESS OF COMPLETION PERCENTAGE AND HIGH SCORE PERCENTAGE

```
num_feats1 = prcDF.dtypes[animeDFClean2.dtypes != 'object'].index
skew_feats1 = prcDF[num_feats1].skew().sort_values(ascending = False)
skewness1 = pd.DataFrame({'Skew': skew_feats1})
skewness1
```

	Skew
High Score Percentage	0.617721
Completion Percentage	-0.859160

High Score Percentage is skewed to the left, whereas Completion Percentage is skewed to the right.

ACCEPTABLE RANGE: - 0.5 to 0.5

METHODS TO DECREASE SKEWNESS

3 Methods to decrease skewness:

1) Using logarithmic versions of the data
2) Using square-rooted versions of the data
3) Using boxcox method
    - Takes in original non-normal data as input and returns fitted data along with the lambda value that was used to fit the non-normal distribution to normal distribution

TRANSFORMATION OF DATA AND SKEWNESS ANALYSIS

LOGARITHMIC TRANSFORMATION

```
# Transform the data to log version
completePrcLog = pd.DataFrame(np.log(prcDF['Completion Percentage']))
print(completePrcLog.skew())
highScorePrcLog = pd.DataFrame(np.log(prcDF['High Score Percentage']))
print(highScorePrcLog.skew())
completePrcLog
```

```
Completion Percentage   -1.076307
dtype: float64
High Score Percentage   -0.651648
dtype: float64
```

Skewness of both features are not within acceptable range.

SQUARE-ROOT TRANSFORMATION

```
# Transform the data to square root version
completePrcSqrt = pd.DataFrame(np.sqrt(prcDF['Completion Percentage']))
print(completePrcSqrt.skew())
highScorePrcSqrt = pd.DataFrame(np.sqrt(prcDF['High Score Percentage']))
print(highScorePrcSqrt.skew())
highScorePrcSqrt
```

```
Completion Percentage   -0.964922
dtype: float64
High Score Percentage    0.044418
dtype: float64
```

Only the skewness of high score percentage is within the acceptable range

BOX COX TRANSFORMATION

```
from scipy import stats

completePrcBoxcox = pd.DataFrame(stats.boxcox(prcDF['Completion Percentage'])[0])
completePrcBoxcox.rename(columns = {0: 'Completion Percentage Boxcox'}, inplace = True)
print(completePrcBoxcox.skew())
highScorePrcBoxcox = pd.DataFrame(stats.boxcox(prcDF['High Score Percentage'])[0])
highScorePrcBoxcox.rename(columns = {0: 'High Score Percentage Boxcox'}, inplace = True)
print(highScorePrcBoxcox.skew())
highScorePrcBoxcox
```

```
Completion Percentage Boxcox   -0.186478
dtype: float64
High Score Percentage Boxcox   -0.05184
dtype: float64
```

Skewness of both features are within acceptable range and in the same direction!

Let's do regression on this data!

VISUALISATION OF LESS-SKEWED DATA

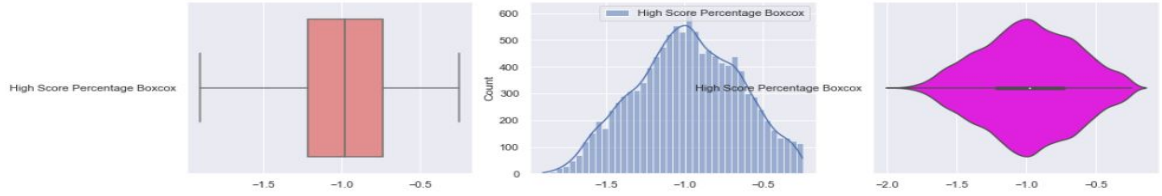
QUICK LOOK AT THE RESPECTIVE PLOTS AFTER SKEWNESS TRANSFORMATION

```
f, axes = plt.subplots(1, 3, figsize=(16, 4))

sb.boxplot(data = completePrcBoxcox, orient = "h", ax = axes[0], color = 'lightsalmon')
sb.histplot(data = completePrcBoxcox, ax = axes[1], kde = True)
sb.violinplot(data = completePrcBoxcox, orient = "h", ax = axes[2], color = 'powderblue')
```

```
<AxesSubplot:>
```


```
f, axes = plt.subplots(1, 3, figsize=(16, 4))

sb.boxplot(data = highScorePrcBoxcox, orient = "h", ax = axes[0], color = 'lightcoral')
sb.histplot(data = highScorePrcBoxcox, ax = axes[1], kde = True)
sb.violinplot(data = highScorePrcBoxcox, orient = "h", ax = axes[2], color = 'magenta')
```

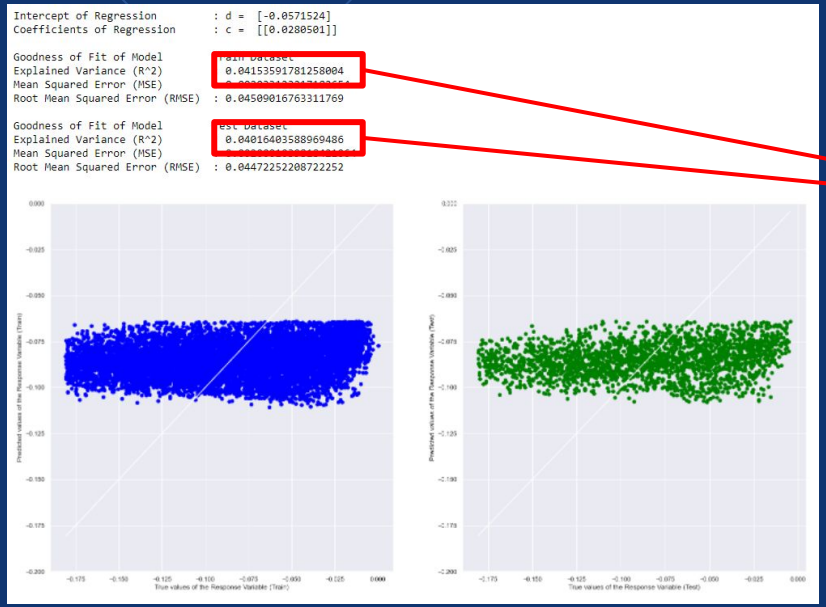
```
<AxesSubplot:>
```



Sample Collection	Data Preparation	Exploratory Analysis	Analytic Visualisation	Algorithmic Optimisation	Information Presentation

LINEAR REGRESSION ON LESS-SKEWED DATA

LINEAR REGRESSION RESULTS



R^2 for train data: 0.0415…
R^2 for test data: 0.0401…

Does not look like the R^2 has improved at all!

Sample Collection	Data Preparation	Exploratory Analysis	Analytic Visualisation	Algorithmic Optimisation	Information Presentation

POLYNOMIAL REGRESSION ON LESS-SKEWED DATA

DEGREE 2 POLYNOMIAL REGRESSION

Polynomial Regression between High Score Percentage and Completion Percentage (after skewness reduction)

```
poly_features = PolynomialFeatures(degree = 2, include_bias = False)
highScorePrcBoxcox_poly = poly_features.fit_transform(highScorePrcBoxcox)
highScorePrcBoxcox_poly[11]
```

```
array([-0.63293672,  0.40060889])
```

```
lin_reg = LinearRegression()
lin_reg.fit(highScorePrcBoxcox_poly, completePrcBoxcox)
print('Coefficients of highScorePrcBoxcox_poly are', lin_reg.coef_)
print('Intercept is', lin_reg.intercept_)
```

```
Coefficients of highScorePrcBoxcox_poly are [[0.1468277  0.05995687]]
Intercept is [-0.00507403]
```

```
# model evaluation
completePrcBCDeg = lin_reg.predict(highScorePrcBoxcox_poly)
mse1 = mean_squared_error(completePrcBoxcox, completePrcBCDeg)

rmse1 = np.sqrt(mean_squared_error(completePrcBoxcox, completePrcBCDeg))
r2_1 = r2_score(completePrcBoxcox, completePrcBCDeg)


print('MSE of Polyregression model: ', mse1)
print('RMSE of Polyregression Model: ', rmse1)
print('R2 score of Polyregression model: ', r2_1)
```

```
MSE of Polyregression model:  0.0019637228858227322
RMSE of Polyregression Model:  0.044313913005090536
R2 score of Polyregression model:  0.07099555398934987
```

DEGREE 3 POLYNOMIAL REGRESSION

Polynomial Regression between High Score Percentage and Completion Percentage (after skewness reduction)

```
poly_features = PolynomialFeatures(degree = 3, include_bias = False)
highScorePrcBoxcox_poly = poly_features.fit_transform(highScorePrcBoxcox)
highScorePrcBoxcox_poly[11]
```

```
array([-0.63293672,  0.40060889, -0.25356008])
```

```
lin_reg = LinearRegression()
lin_reg.fit(highScorePrcBoxcox_poly, completePrcBoxcox)
print('Coefficients of highScorePrcBoxcox_poly are', lin_reg.coef_)
print('Intercept is', lin_reg.intercept_)
```

```
Coefficients of highScorePrcBoxcox_poly are [[-0.04043064 -0.14655176 -0.06906929]]
Intercept is [-0.05502197]
```

```
# model evaluation
completePrcBCDeg = lin_reg.predict(highScorePrcBoxcox_poly)
mse1 = mean_squared_error(completePrcBoxcox, completePrcBCDeg)

rmse1 = np.sqrt(mean_squared_error(completePrcBoxcox, completePrcBCDeg))
r2_1 = r2_score(completePrcBoxcox, completePrcBCDeg)


print('MSE of Polyregression model: ', mse1)
print('RMSE of Polyregression Model: ', rmse1)
print('R2 score of Polyregression model: ', r2_1)
```

```
MSE of Polyregression model:  0.0019502701140470251
RMSE of Polyregression Model:  0.04416186266505326
R2 score of Polyregression model:  0.07735983526397705
```

POLYNOMIAL REGRESSION ON LESS-SKEWED DATA

DEGREE 5 POLYNOMIAL REGRESSION

Polynomial Regression between High Score Percentage and Completion Percentage (after skewness reduction)

```
poly_features = PolynomialFeatures(degree = 5, include_bias = False)
highScorePrcBoxcox_poly = poly_features.fit_transform(highScorePrcBoxcox)
highScorePrcBoxcox_poly[11]
```

```
array([-0.63293672,  0.40060889, -0.25356008,  0.16048748, -0.10157842])
```

```
lin_reg = LinearRegression()
lin_reg.fit(highScorePrcBoxcox_poly, completePrcBoxcox)
print('Coefficients of highScorePrcBoxcox_poly are', lin_reg.coef_)
print('Intercept is', lin_reg.intercept_)
```

```
Coefficients of highScorePrcBoxcox_poly are [[0.36887305 0.83171036 1.00320602 0.54754395 0.10546194]]
Intercept is [0.00650128]
```

```
# model evaluation
completePrcBCDeg = lin_reg.predict(highScorePrcBoxcox_poly)
mse1 = mean_squared_error(completePrcBoxcox, completePrcBCDeg)

rmse1 = np.sqrt(mean_squared_error(completePrcBoxcox, completePrcBCDeg))
r2_1 = r2_score(completePrcBoxcox, completePrcBCDeg)


print('MSE of Polyregression model: ', mse1)
print('RMSE of Polyregression Model: ', rmse1)
print('R2 score of Polyregression model: ', r2_1)
```

```
MSE of Polyregression model:  0.001949300635727454
RMSE of Polyregression Model:  0.044150888488045799
R2 score of Polyregression model:  0.07781847923848839
```

DEGREE 10 POLYNOMIAL REGRESSION

Polynomial Regression between High Score Percentage and Completion Percentage (after skewness reduction)

```
poly_features = PolynomialFeatures(degree = 10, include_bias = False)
highScorePrcBoxcox_poly = poly_features.fit_transform(highScorePrcBoxcox)
highScorePrcBoxcox_poly[11]
```

```
array([-0.63293672,  0.40060889, -0.25356008,  0.16048748, -0.10157842,
        0.06429271, -0.04069322,  0.02575623, -0.01630207,  0.01031818])
```

```
lin_reg = LinearRegression()
lin_reg.fit(highScorePrcBoxcox_poly, completePrcBoxcox)
print('Coefficients of highScorePrcBoxcox_poly are', lin_reg.coef_)
print('Intercept is', lin_reg.intercept_)
```

```
Coefficients of highScorePrcBoxcox_poly are [[ -12.24184165  -76.78882372 -264.19107912 -560.5550533  -773.5556882
  -708.32036942 -427.33758263 -163.25985147  -35.78413505   -3.42654596]]
Intercept is [-0.85179074]
```

```
# model evaluation
completePrcBCDeg = lin_reg.predict(highScorePrcBoxcox_poly)
mse1 = mean_squared_error(completePrcBoxcox, completePrcBCDeg)

rmse1 = np.sqrt(mean_squared_error(completePrcBoxcox, completePrcBCDeg))
r2_1 = r2_score(completePrcBoxcox, completePrcBCDeg)


print('MSE of Polyregression model: ', mse1)
print('RMSE of Polyregression Model: ', rmse1)
print('R2 score of Polyregression model: ', r2_1)
```

```
MSE of Polyregression model:  0.0019486996528665658
RMSE of Polyregression Model:  0.0441440783442872
R2 score of Polyregression model:  0.07810279417608579
```

EVALUATING THE ACCURACY OF REGRESSION MODELS

- R^2 still remained very low
  - Degree 2: 0.0709…
  - Degree 3: 0.0774…
  - Degree 5: 0.0778…
  - Degree 10: 0.0781…

- **CONCLUSION:** No linear or a non-linear relationship between high score percentage and completion percentage, even after skewness of data has been reduced.

Sample Collection	Data Preparation	Exploratory Analysis	Analytic Visualisation	Algorithmic Optimisation	Information Presentation

INTRODUCING K-MEANS

- K-Means clustering is an unsupervised learning algorithm
- No labelled data for this clustering unlike in supervised learning
- K-Means performs the division of objects into clusters that share similarities and are dissimilar to the objects belonging to another cluster
- We will be using K-Means to evaluate some categorical variables

Sample Collection	Data Preparation	Exploratory Analysis	Analytic Visualisation	Algorithmic Optimisation	Information Presentation

LET'S TAKE A LOOK AT STUDIOS...

- Usage of new K-Means model (Unsupervised, for Categorical Variables)
- Challenges in evaluating 'Studios':
  - How can we change the strings from categorical variables into something our model can evaluate? (Label Encoding)
  - Need to find a suitable K Value for K-Means Analysis (Elbow Test)
  - How can we evaluate the K-Means Clusters? (Box-Plots of separate clusters)

Sample Collection	Data Preparation	Exploratory Analysis	Analytic Visualisation	Algorithmic Optimisation	Information Presentation

ENCODING THE STUDIOS

Usage of Label Encoding to change the studio strings into numeric values for K-Means Analysis

```
#LABEL ENCODING STUDIOS COLUMN VALUES TO BECOME MULTICLASS FOR KMEANS ANALYSIS

lab_enc = preprocessing.LabelEncoder()
studios_encoded = lab_enc.fit_transform(studiosDFClean['Studio 1'])
print(studios_encoded)
print(utils.multiclass.type_of_target(studios_encoded))
print(utils.multiclass.type_of_target(studios_encoded.astype('int')))
print(utils.multiclass.type_of_target(studios_encoded))
```

```
[514  79 277 ... 348   4 514]
multiclass
multiclass
multiclass
```

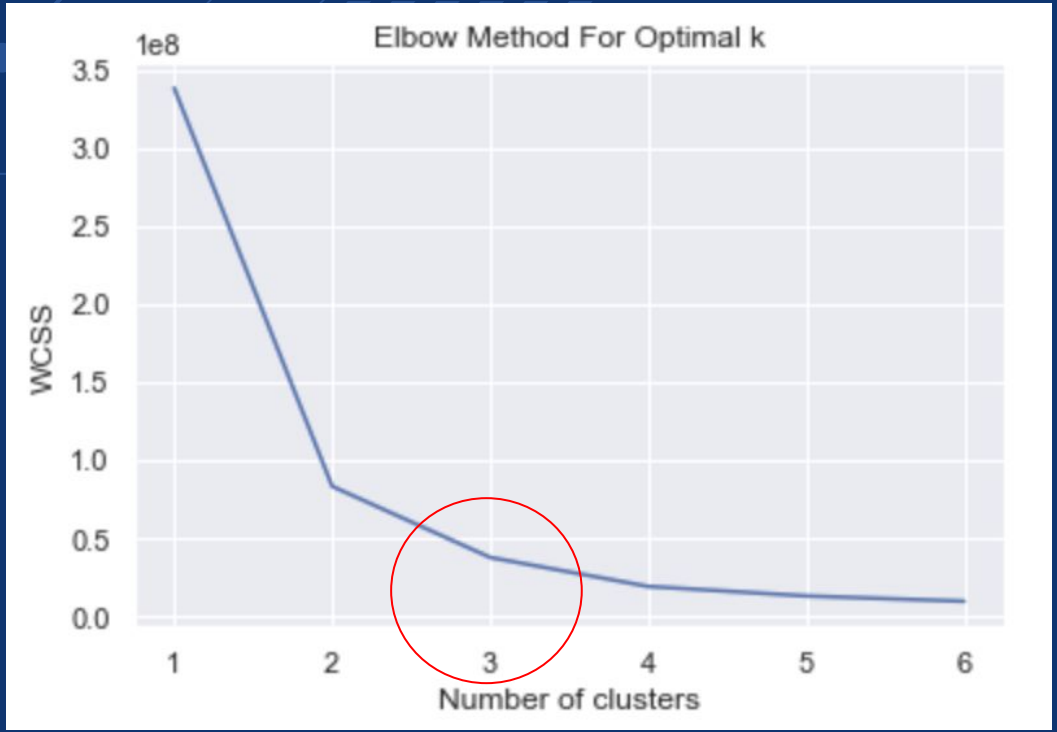
FINDING APPROPRIATE K-VALUE

Usage of Elbow Test to find appropriate K-Value

- Elbow Test is a popular heuristic to use .
- Idea is to run k-means clustering for a range of clusters k and for each value, we are calculating the sum of squared Euclidean distances from each point to its assigned center(distortions).
- Hence the ‘elbow’ number, would be the number of clusters to use.

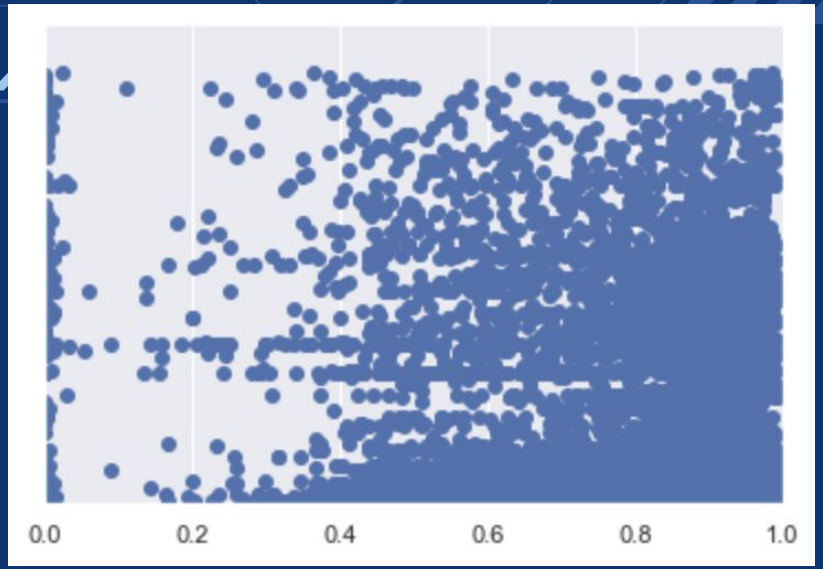
Sample Collection	Data Preparation	Exploratory Analysis	Analytic Visualisation	Algorithmic Optimisation	Information Presentation

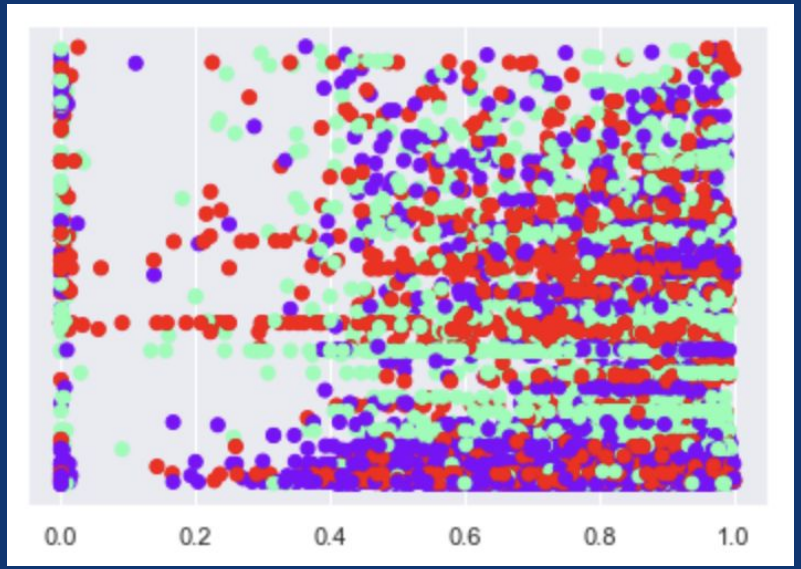
FINDING APPROPRIATE K-VALUE



Sample Collection	Data Preparation	Exploratory Analysis	Analytic Visualisation	Algorithmic Optimisation	Information Presentation

CONDUCTING K-MEANS USING K = 3





BEFORE

AFTER

Sample Collection	Data Preparation	Exploratory Analysis	Analytic Visualisation	Algorithmic Optimisation	Information Presentation

ANALYSING THE CLUSTERS

	Cluster	Studio 1	Completion Percentage
0	0	Sunrise	0.879720
1	1	Bones	0.987182
2	2	Madhouse	0.897122
3	0	Sunrise	0.814715
4	0	Toei Animation	0.796038

ANALYSING THE CLUSTERS

	Cluster	Studio 1	Completion Percentage
0	0	Sunrise	0.879720
3	0	Sunrise	0.814715
4	0	Toei Animation	0.796038
10	0	Studio Pierrot	0.900515
11	0	Toei Animation	0.000102
...	...	...	...
17519	0	ufotable	0.300000
17536	0	Wit Studio	1.000000
17538	0	Studio Pierrot	1.000000
17549	0	Studio Kafka	0.363636
17561	0	Sunrise	0.000000

	Cluster	Studio 1	Completion Percentage
1	1	Bones	0.987182
5	1	Gallop	0.752271
8	1	A.C.G.T.	0.953772
15	1	Gonzo	0.791165
17	1	Gonzo	0.827863
...	...	...	...
17528	1	Fanworks	0.000000
17532	1	CG Year	0.000000
17552	1	Bones	0.986045
17555	1	Arvo Animation	0.000000
17560	1	8bit	0.400000

	Cluster	Studio 1	Completion Percentage
2	2	Madhouse	0.897122
6	2	J.C.Staff	0.779701
7	2	Nippon Animation	0.873463
9	2	Madhouse	0.752634
16	2	Madhouse	0.726134
...	...	...	...
17524	2	Ruo Hong Culture	0.000000
17537	2	MAPPA	0.000000
17543	2	SILVER LINK.	0.294118
17544	2	SILVER LINK.	0.222222
17558	2	Passione	0.000000

ANALYSING THE CLUSTERS

```
#CLUSTER 0 STATISTICS

Cluster0DF.describe()
```

	Cluster	Completion Percentage
count	4178.0	4178.000000
mean	0.0	0.808968
std	0.0	0.189087
min	0.0	0.000000
25%	0.0	0.736486
50%	0.0	0.867809
75%	0.0	0.945288
max	0.0	1.000000

```
#CLUSTER 1 STATISTICS

Cluster1DF.describe()
```

	Cluster	Completion Percentage
count	2646.0	2646.000000
mean	1.0	0.805703
std	0.0	0.202029
min	1.0	0.000000
25%	1.0	0.747907
50%	1.0	0.874916
75%	1.0	0.940630
max	1.0	0.995724

```
#CLUSTER 2 STATISTICS

Cluster2DF.describe()
```

	Cluster	Completion Percentage
count	3504.0	3504.000000
mean	2.0	0.823762
std	0.0	0.198056
min	2.0	0.000000
25%	2.0	0.784793
50%	2.0	0.890310
75%	2.0	0.949414
max	2.0	1.000000

ANALYSING THE CLUSTERS

```
#TOP 5 STUDIOS IN CLUSTER 0

n = 5
Cluster0DF['Studio 1'].value_counts()[:n].index.tolist()
```

```
['Toei Animation',
 'Sunrise',
 'TMS Entertainment',
 'Studio Deen',
 'Studio Pierrot']
```

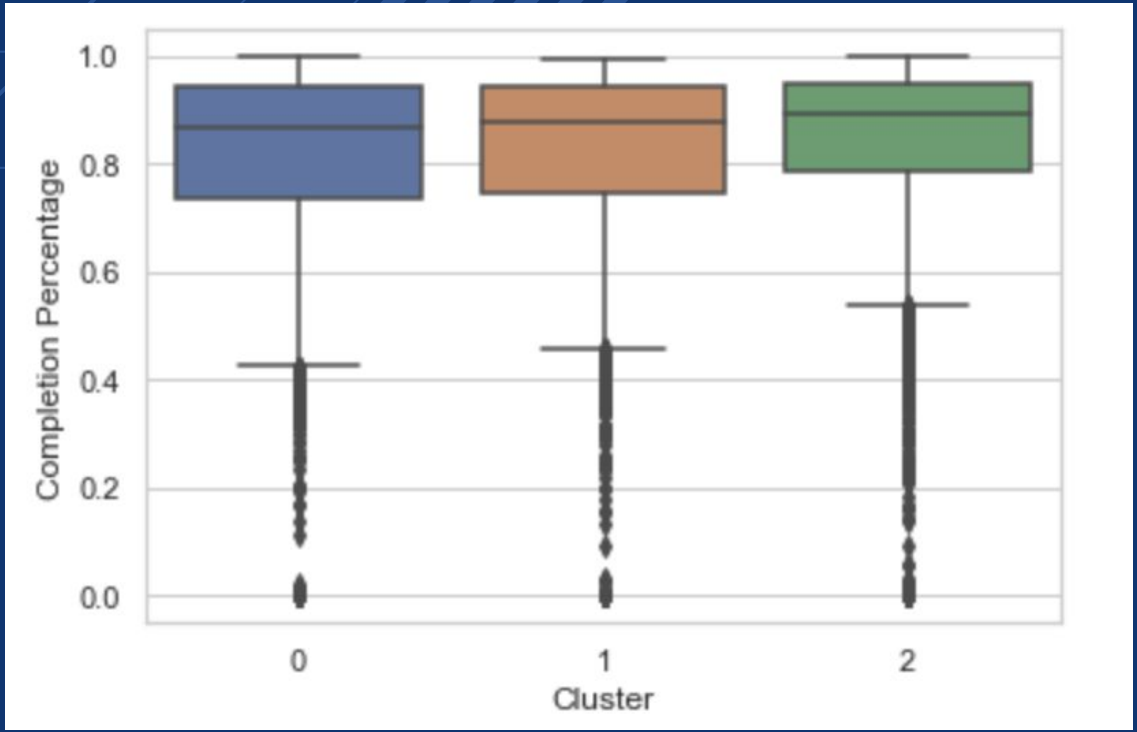
```
#TOP 5 STUDIOS IN CLUSTER 1

n = 5
Cluster1DF['Studio 1'].value_counts()[:n].index.tolist()
```

```
['A-1 Pictures', 'DLE', 'AIC', 'Gonzo', 'Bones']
```

```
#TOP 5 STUDIOS IN CLUSTER 2

n = 5
Cluster2DF['Studio 1'].value_counts()[:n].index.tolist()
```

```
['J.C.Staff', 'Madhouse', 'Production I.G', 'OLM', 'Nippon Animation']
```

VISUALISING THE CLUSTERS



Sample Collection	Data Preparation	Exploratory Analysis	Analytic Visualisation	Algorithmic Optimisation	Information Presentation

EVALUATING ACCURACY

- Usage of Silhouette Analysis
- Silhouette analysis can be used to determine degree of separation between clusters.
- For each sample:
  - Compute the average distance from all data points in the same cluster
  - Compute average distance from all data points to the closest cluster
  - Compute the coefficient [-1,1]

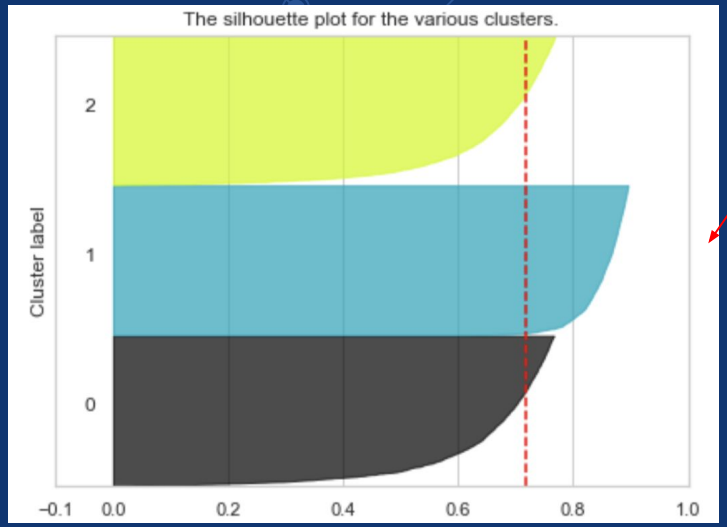
$$\frac{b^i - a^i}{max(a^i, b^i)}$$

- Thus we want the coefficient to be big as possible and be close to 1 to have good clusters

EVALUATING ACCURACY

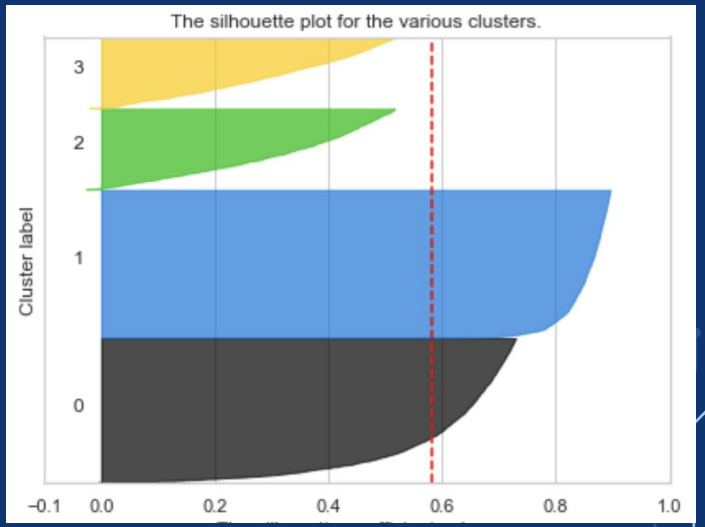
```
For n_clusters = 3 The average silhouette_score is : 0.7188541797952277
For n_clusters = 4 The average silhouette_score is : 0.5825308202638994
For n_clusters = 5 The average silhouette_score is : 0.4029400287881206
For n_clusters = 6 The average silhouette_score is : 0.3088229206176786
```

N_clusters = 3 has the best Silhouette score of 0.72



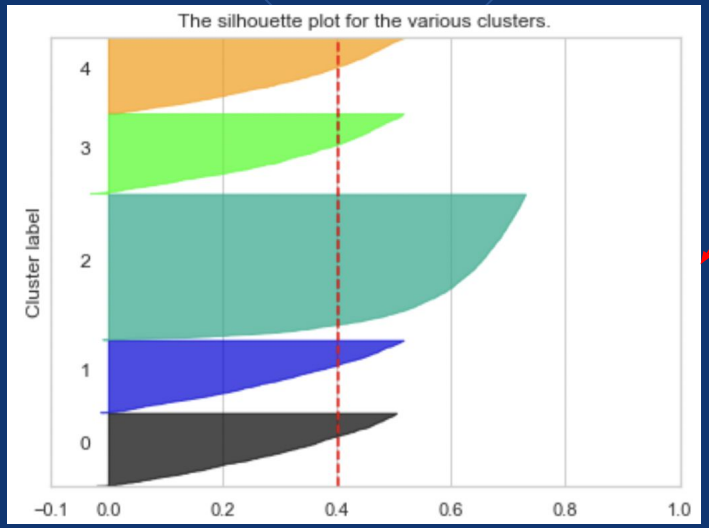
Number of Clusters = 3

Number of Clusters = 4



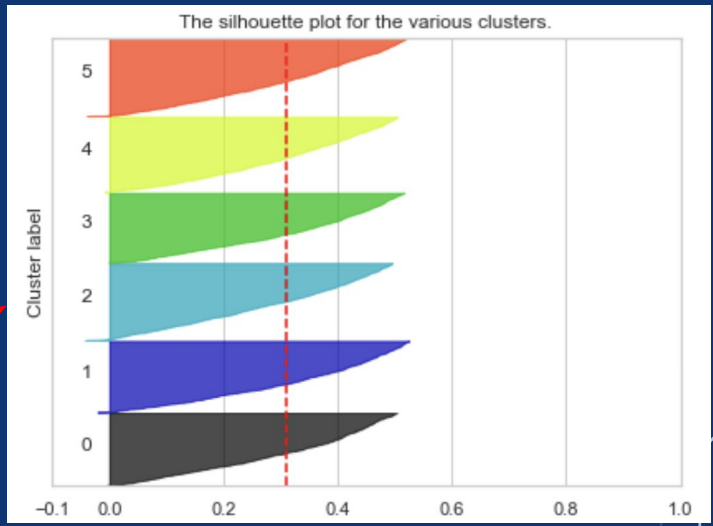
Sample Collection	Data Preparation	Exploratory Analysis	Analytic Visualisation	Algorithmic Optimisation	Information Presentation

EVALUATING ACCURACY



Number of Clusters = 5

Number of Clusters = 6



Sample Collection	Data Preparation	Exploratory Analysis	Analytic Visualisation	Algorithmic Optimisation	Information Presentation

EVALUATING ACCURACY

- From Silhouette analysis, cluster with k=3 had highest score of 0.72
  - Suggests clusters are sufficiently spaced away from each other
  - Clusters are also big compared to other clusters of differing K values
- Overall, this highlights the high accuracy of our K-Means clustering

LET'S TAKE A LOOK AT SOURCE...

- Challenges faced were similar to the ones that we faced when analysing Studios
- Since we used the K-Means analysis to evaluate Sources as well, we used the same techniques to circumvent these challenges once more (Label Encoding, Elbow and Silhouette Test)

Sample Collection	Data Preparation	Exploratory Analysis	Analytic Visualisation	Algorithmic Optimisation	Information Presentation

LET'S TAKE A LOOK AT SOURCE...

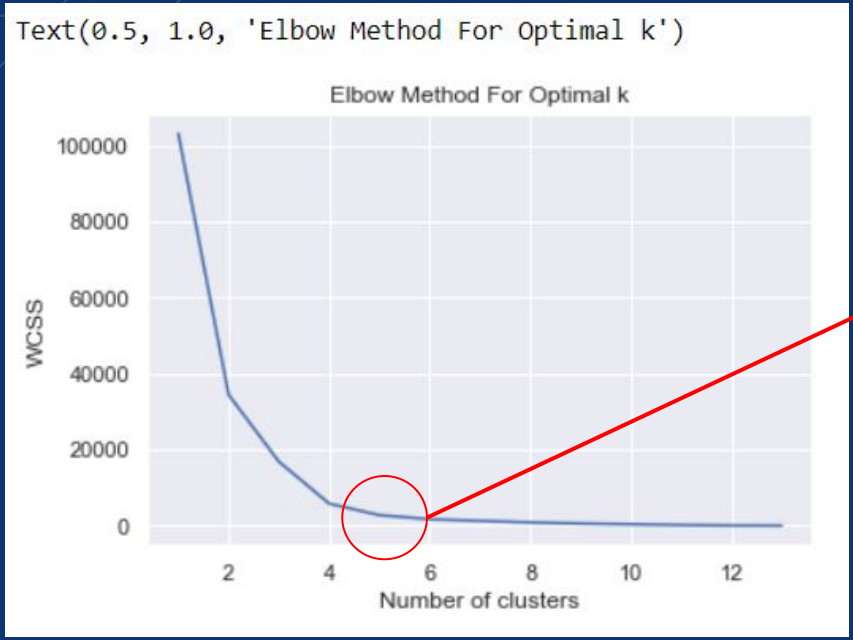
Using Label Encoding (just as mentioned earlier) to encode the different types of sources into numeric variables

Table of sources and its encodings

Source	Source Numeric	
Original	9	5136
Manga	6	3752
Visual novel	13	985
Game	4	853
Light novel	5	742
Other	10	579
Novel	8	473
Music	7	314
4-koma manga	0	284
Web manga	14	240
Picture book	11	144
Book	1	111
Card game	2	64
Digital manga	3	15
Radio	12	12

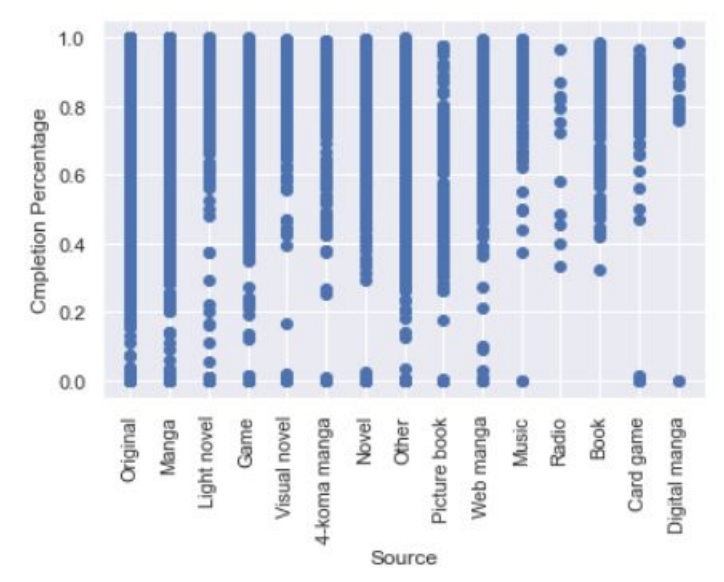
dtype: int64

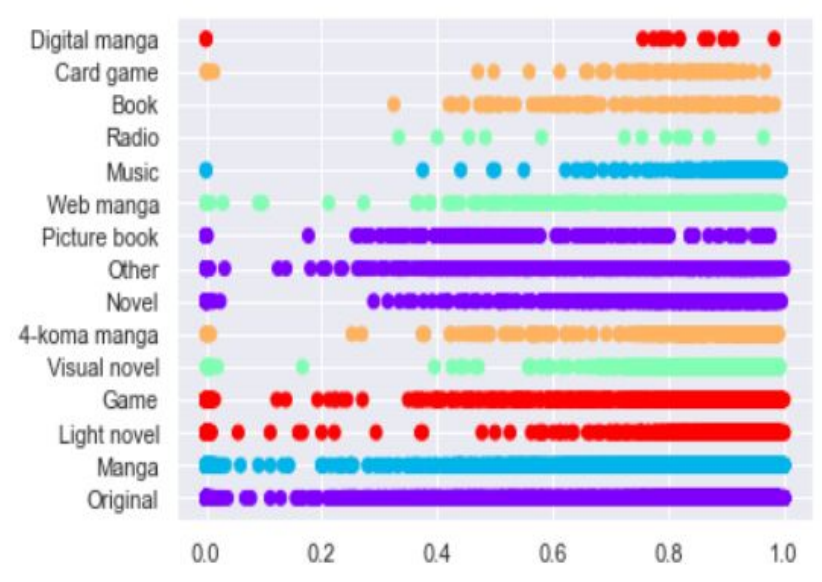
FINDING APPROPRIATE K-VALUE



Sample Collection	Data Preparation	Exploratory Analysis	Analytic Visualisation	Algorithmic Optimisation	Information Presentation

CONDUCTING K-MEANS USING K = 5





BEFORE

AFTER

Sample Collection	Data Preparation	Exploratory Analysis	Analytic Visualisation	Algorithmic Optimisation	Information Presentation

ANALYSING THE CLUSTERS

	Cluster	Source	Completion Percentage
0	0	Original	0.879720
1	0	Original	0.987182
3	0	Original	0.814715
16	0	Original	0.726134
20	0	Original	0.935857
...	...	...	...
17524	0	Original	0.000000
17526	0	Original	0.925000
17537	0	Original	0.000000
17538	0	Other	1.000000
17552	0	Original	0.986045

[6332 rows x 3 columns]

['Original', 'Novel', 'Other', 'Picture book']
Categories (4, object): ['Original', 'Novel', 'Other', 'Picture book']

CLUSTER 0

	Cluster	Source	Completion Percentage
2	1	Manga	0.897122
4	1	Manga	0.796038
5	1	Manga	0.752271
6	1	Manga	0.779701
7	1	Manga	0.873463
...	...	...	...
17549	1	Manga	0.363636
17551	1	Manga	0.000000
17556	1	Manga	0.000000
17558	1	Manga	0.000000
17560	1	Manga	0.400000

[4066 rows x 3 columns]

['Manga', 'Music']
Categories (2, object): ['Manga', 'Music']

CLUSTER 1

ANALYSING THE CLUSTERS

	Cluster	Source	Completion Percentage
43	2	Visual novel	0.843385
58	2	Visual novel	0.869598
80	2	Visual novel	0.903788
98	2	Visual novel	0.922003
112	2	Visual novel	0.912363
...	...	...	...
17215	2	Visual novel	0.729730
17239	2	Visual novel	0.878327
17248	2	Web manga	0.500000
17313	2	Web manga	0.859617
17317	2	Visual novel	0.006593

[1237 rows x 3 columns]

['Visual novel', 'Web manga', 'Radio']
Categories (3, object): ['Visual novel', 'Web manga', 'Radio']

CLUSTER 2

	Cluster	Source	Completion Percentage
47	3	4-koma manga	0.830622
279	3	4-koma manga	0.864372
388	3	4-koma manga	0.967684
455	3	4-koma manga	0.859681
458	3	4-koma manga	0.889889
...	...	...	...
17088	3	4-koma manga	0.000000
17112	3	Card game	0.857724
17145	3	4-koma manga	0.004662
17189	3	4-koma manga	0.003906
17452	3	Book	0.611111

[459 rows x 3 columns]

['4-koma manga', 'Book', 'Card game']
Categories (3, object): ['4-koma manga', 'Book', 'Card game']

CLUSTER 3

	Cluster	Source	Completion Percentage
17	4	Light novel	0.827863
36	4	Game	0.812009
51	4	Light novel	0.898549
52	4	Light novel	0.962417
53	4	Light novel	0.966722
...	...	...	...
17540	4	Light novel	0.160000
17543	4	Light novel	0.294118
17544	4	Light novel	0.222222
17555	4	Light novel	0.000000
17561	4	Game	0.000000

[1610 rows x 3 columns]

['Light novel', 'Game', 'Digital manga']
Categories (3, object): ['Light novel', 'Game', 'Digital manga']

CLUSTER 4

ANALYSING THE CLUSTERS

	Cluster	Completion Percentage
count	6332.0	6332.000000
mean	0.0	0.761859
std	0.0	0.230111
min	0.0	0.000000
25%	0.0	0.652998
50%	0.0	0.846968
75%	0.0	0.932383
max	0.0	1.000000

CLUSTER 0

	Cluster	Completion Percentage
count	4066.0	4066.000000
mean	1.0	0.839599
std	0.0	0.181202
min	1.0	0.000000
25%	1.0	0.802089
50%	1.0	0.895344
75%	1.0	0.952550
max	1.0	1.000000

CLUSTER 1

	Cluster	Completion Percentage
count	1237.0	1237.000000
mean	2.0	0.832807
std	0.0	0.153898
min	2.0	0.000000
25%	2.0	0.821256
50%	2.0	0.865005
75%	2.0	0.906924
max	2.0	0.993765

CLUSTER 2

ANALYSING THE CLUSTERS

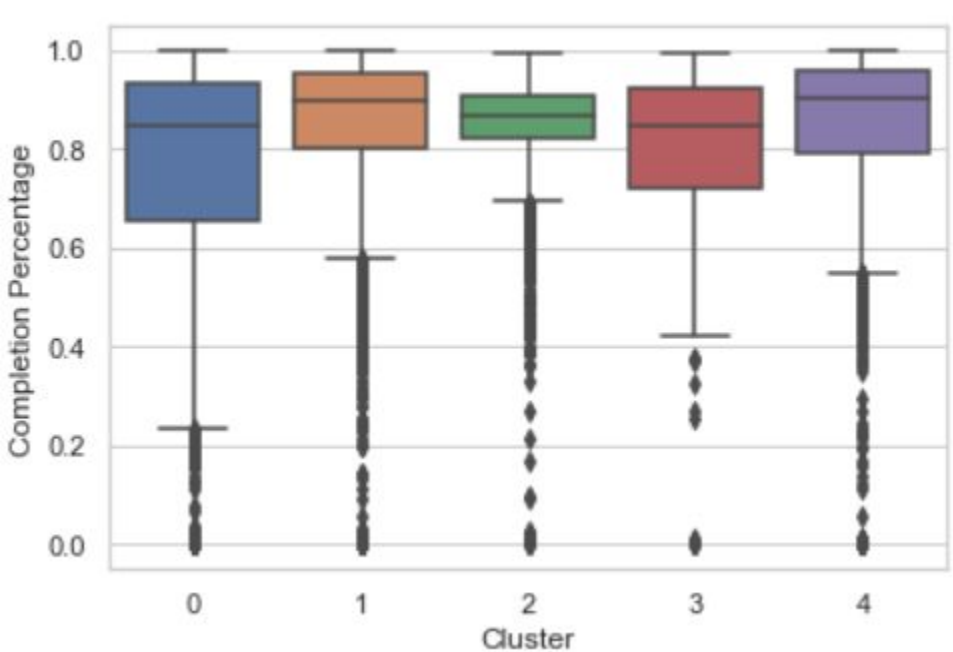
	Cluster	Completion Percentage
count	459.0	459.000000
mean	3.0	0.784325
std	0.0	0.202767
min	3.0	0.000000
25%	3.0	0.721158
50%	3.0	0.846077
75%	3.0	0.923023
max	3.0	0.991179

CLUSTER 3

	Cluster	Completion Percentage
count	1610.0	1610.000000
mean	4.0	0.819395
std	0.0	0.225848
min	4.0	0.000000
25%	4.0	0.793356
50%	4.0	0.902246
75%	4.0	0.956569
max	4.0	1.000000

CLUSTER 4

VISUALISING THE CLUSTERS



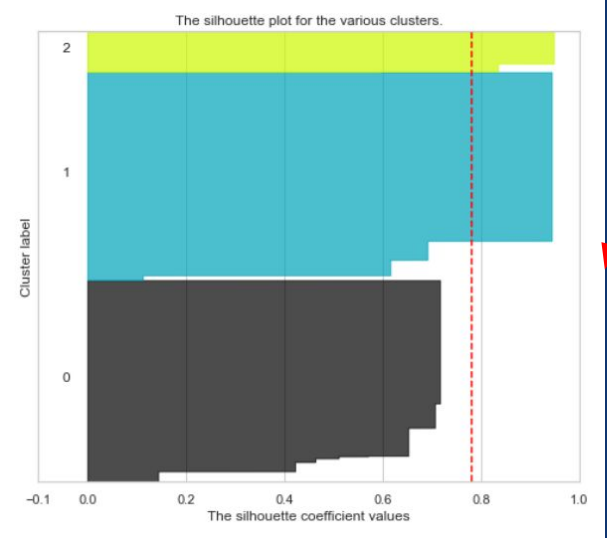
Sample Collection	Data Preparation	Exploratory Analysis	Analytic Visualisation	Algorithmic Optimisation	Information Presentation

EVALUATING ACCURACY

Silhouette Scores for Number of Clusters

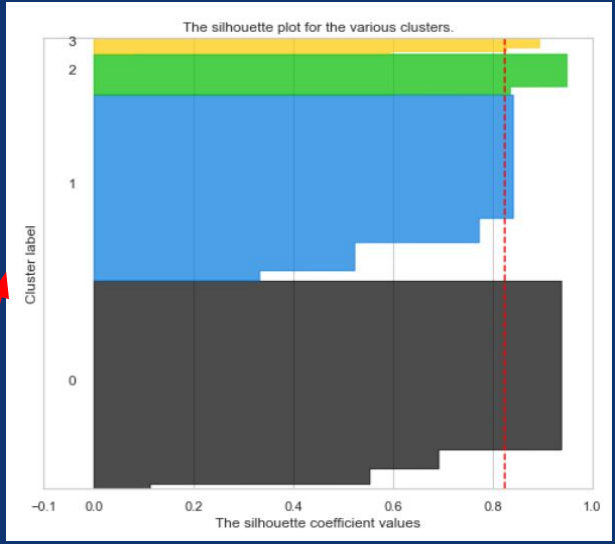
```
For n_clusters = 3 The average silhouette_score is : 0.7825095181287559
For n_clusters = 4 The average silhouette_score is : 0.8242666658935132
For n_clusters = 5 The average silhouette_score is : 0.8511739993359606
```

N_clusters = 5 has the best silhouette score of 0.851



Number of Clusters = 3

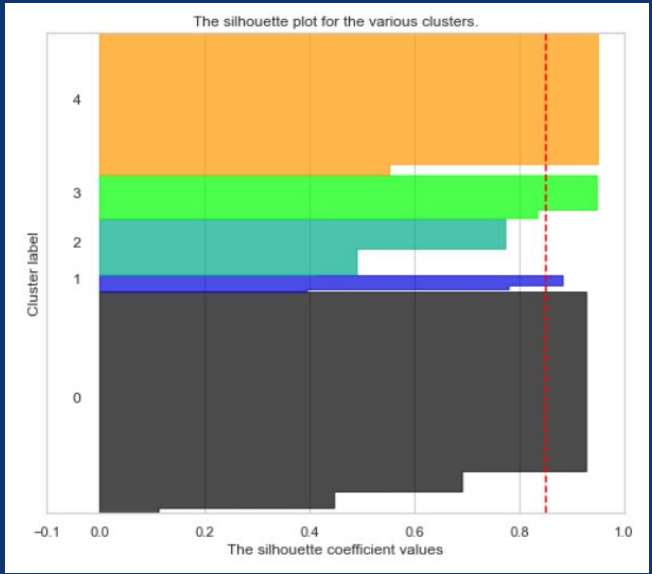
Number of Clusters = 4



Sample Collection	Data Preparation	Exploratory Analysis	Analytic Visualisation	Algorithmic Optimisation	Information Presentation

EVALUATING ACCURACY

Number of Clusters = 5



Sample Collection	Data Preparation	Exploratory Analysis	Analytic Visualisation	Algorithmic Optimisation	Information Presentation

EVALUATING ACCURACY

- From Silhouette Analysis, the value that we chose, K=5, had the highest Silhouette score at 0.851

Sample Collection	Data Preparation	Exploratory Analysis	Analytic Visualisation	Algorithmic Optimisation	Information Presentation

LET'S TAKE A LOOK AT GENRES...

- Challenges in evaluating 'Genres':
  - In addition to the challenges faced when analysing Studios and Sources…
  - Genres were also arranged alphabetically, and not based on their order of significance.
  - Hence in addition to Genre 1, we also chose Genres 2 to 5, as they had significant number of non-NULL entries to do K-means clustering.

Sample Collection	Data Preparation	Exploratory Analysis	Analytic Visualisation	Algorithmic Optimisation	Information Presentation

NUMBER OF ENTRIES IN EACH COLUMN

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 15862 entries, 0 to 17561
Data columns (total 2 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Genre 1               15862 non-null  object
 1   Completion Percentage 15862 non-null  float64
dtypes: float64(1), object(1)
memory usage: 371.8+ KB
None
```

GENRE 1

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 15862 entries, 0 to 17561
Data columns (total 2 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Genre 2               12655 non-null  object
 1   Completion Percentage 15862 non-null  float64
dtypes: float64(1), object(1)
memory usage: 371.8+ KB
None
```

GENRE 2

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 15862 entries, 0 to 17561
Data columns (total 2 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Genre 3               8666 non-null   object
 1   Completion Percentage 15862 non-null  float64
dtypes: float64(1), object(1)
memory usage: 371.8+ KB
None
```

GENRE 3

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 15862 entries, 0 to 17561
Data columns (total 2 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Genre 4               5344 non-null   object
 1   Completion Percentage 15862 non-null  float64
dtypes: float64(1), object(1)
memory usage: 371.8+ KB
None
```

GENRE 4

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 15862 entries, 0 to 17561
Data columns (total 2 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Genre 5               2870 non-null   object
 1   Completion Percentage 15862 non-null  float64
dtypes: float64(1), object(1)
memory usage: 371.8+ KB
None
```

GENRE 5

Sample Collection	Data Preparation	Exploratory Analysis	Analytic Visualisation	Algorithmic Optimisation	Information Presentation

NUMBER OF UNIQUE GENRES IN EACH COLUMN

```
Genre 1                    40
Completion Percentage    13074
dtype: int64
Genre 2                    40
Completion Percentage    13074
dtype: int64
Genre 3                    40
Completion Percentage    13074
dtype: int64
Genre 4                    40
Completion Percentage    13074
dtype: int64
Genre 5                    38
Completion Percentage    13074
dtype: int64
```

We need to know this information to have an idea of the number of clusters we need to make.

GENRE 1 ANALYSIS

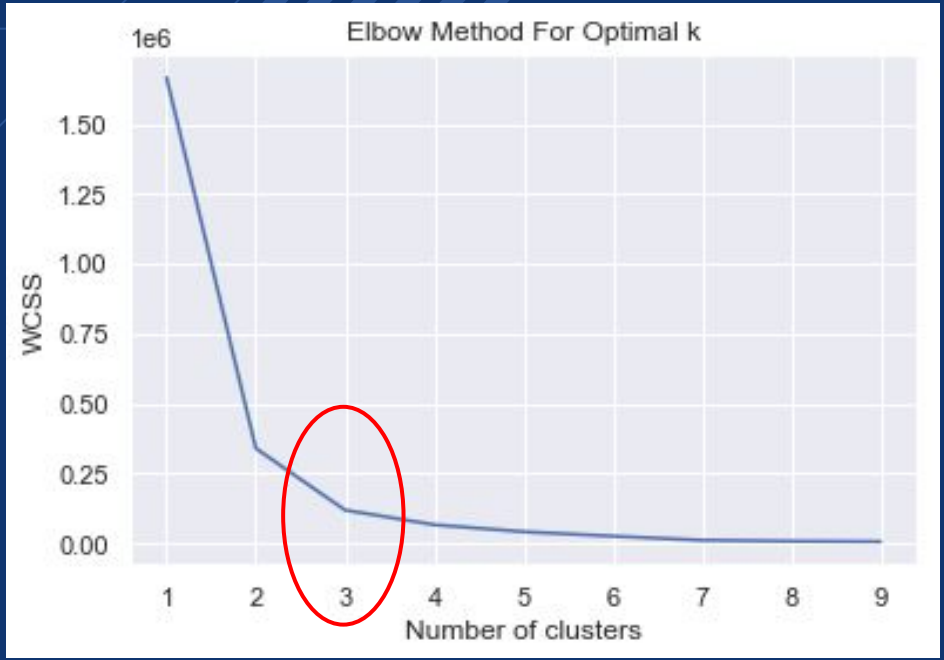
LABEL ENCODING GENRES IN GENRE 1

Genre 1	Genre 1 Encoded	
Action	0	3751
Comedy	3	3067
Adventure	1	1456
Music	19	1424
Kids	14	1078
Slice of Life	32	942
Drama	6	775
Sci-Fi	27	510
Fantasy	8	481
Dementia	4	379
Historical	11	257
Mystery	20	181
Game	9	181
Sports	34	162
Romance	24	143
Ecchi	7	130
Harem	10	114
Magic	15	107
Military	18	99
Mecha	17	82
Horror	12	80
Parody	21	71
Cars	2	62
Supernatural	36	60
Demons	5	54
Psychological	23	47
School	26	45
Space	33	31
Shounen	30	22
Police	22	13
Seinen	28	10
Super Power	35	10
Samurai	25	9
Martial Arts	16	9
Shoujo	29	8
Thriller	37	3
Yaoi	39	3
Josei	13	2
Shounen Ai	31	2
Vampire	38	2

dtype: int64

40 genre types to work with in the first genre column!
(Encoded from 1 to 39)

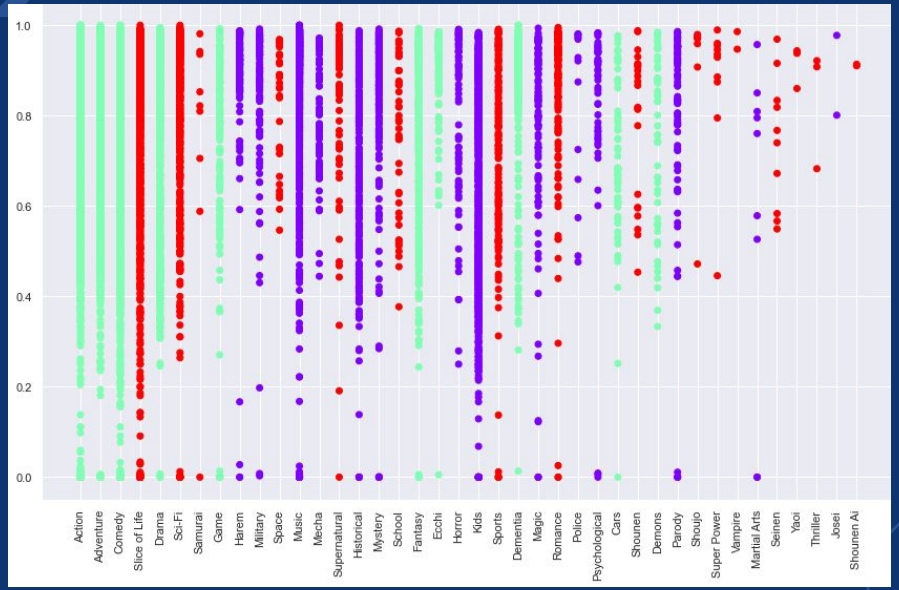
FINDING APPROPRIATE K-VALUE



Sample Collection	Data Preparation	Exploratory Analysis	Analytic Visualisation	Algorithmic Optimisation	Information Presentation

CONDUCTING K-MEANS USING K = 3

BEFORE



AFTER

Sample Collection	Data Preparation	Exploratory Analysis	Analytic Visualisation	Algorithmic Optimisation	Information Presentation

ANALYSING THE CLUSTERS

CLUSTER 0

	Cluster	Genre 1	Completion Percentage	Genre 1 Encoded
34	0	Harem	0.857614	10
43	0	Harem	0.843385	10
58	0	Harem	0.869598	10
61	0	Military	0.968263	18
62	0	Military	0.968998	18
...	...	...	...	...
17510	0	Mystery	0.000000	20
17513	0	Mystery	0.962311	20
17524	0	Martial Arts	0.000000	16
17526	0	Music	0.925000	19
17543	0	Magic	0.294118	15

[3564 rows x 4 columns]

['Harem', 'Military', 'Music', 'Mecha', 'Historical', ..., 'Police', 'Psychological', 'Parody', 'Martial Arts', 'Josei']
Length: 14
Categories (14, object): ['Harem', 'Military', 'Music', 'Mecha', ..., 'Psychological', 'Parody', 'Martial Arts', 'Josei']

CLUSTER 1

	Cluster	Genre 1	Completion Percentage	Genre 1 Encoded
0	1	Action	0.879720	0
1	1	Action	0.987182	0
2	1	Action	0.897122	0
3	1	Action	0.814715	0
4	1	Adventure	0.796038	1
...	...	...	...	...
17552	1	Comedy	0.986045	3
17556	1	Adventure	0.000000	1
17558	1	Comedy	0.000000	3
17560	1	Adventure	0.400000	1
17561	1	Action	0.000000	0

[10336 rows x 4 columns]

]: ['Action', 'Adventure', 'Comedy', 'Drama', 'Game', 'Fantasy', 'Ecchi', 'Dementia', 'Cars', 'Demons']
Categories (10, object): ['Action', 'Adventure', 'Comedy', 'Drama', ..., 'Ecchi', 'Dementia', 'Cars', 'Demons']

ANALYSING THE CLUSTERS

	Cluster	Genre 1	Completion Percentage	Genre 1 Encoded
7	2	Slice of Life	0.873463	32
22	2	Sci-Fi	0.991865	27
27	2	Samurai	0.981051	25
33	2	Sci-Fi	0.857797	27
40	2	Sci-Fi	0.870601	27
...	...	...	...	...
17536	2	Supernatural	1.000000	36
17537	2	Sports	0.000000	34
17549	2	Slice of Life	0.363636	32
17551	2	Slice of Life	0.000000	32
17555	2	Sci-Fi	0.000000	27

[1962 rows x 4 columns]

['Slice of Life', 'Sci-Fi', 'Samurai', 'Space', 'Supernatural', ..., 'Vampire', 'Seinen', 'Yaoi', 'Thriller', 'Shounen Ai']
Length: 16
Categories (16, object): ['Slice of Life', 'Sci-Fi', 'Samurai', 'Space', ..., 'Seinen', 'Yaoi', 'Thriller', 'Shounen Ai']

CLUSTER 2

ANALYSING THE CLUSTERS

	Cluster	Completion Percentage	Genre 1 Encoded
count	3564.0	3564.000000	3564.000000
mean	0.0	0.771221	16.415264
std	0.0	0.219821	3.194834
min	0.0	0.000000	10.000000
25%	0.0	0.656107	14.000000
50%	0.0	0.858155	18.000000
75%	0.0	0.936958	19.000000
max	0.0	1.000000	23.000000

CLUSTER 0

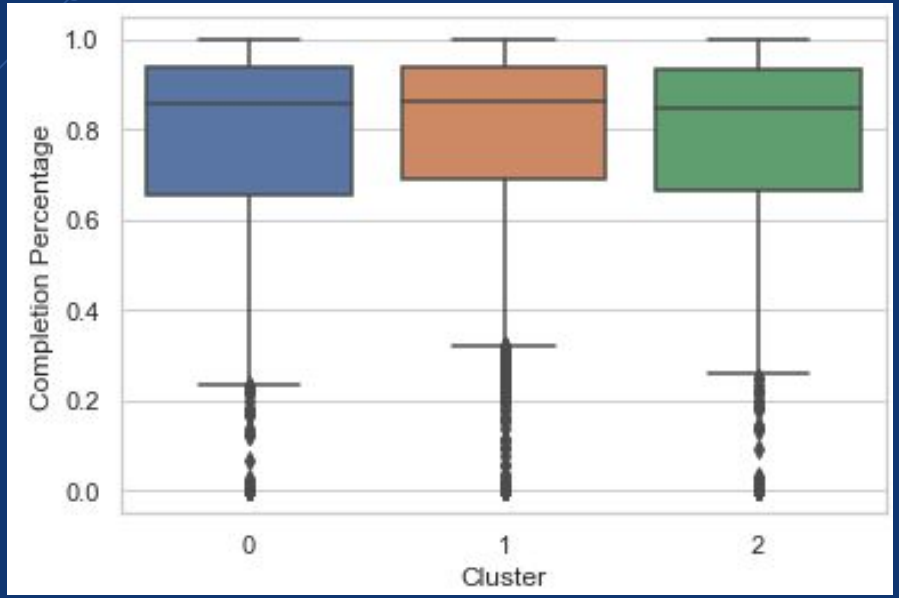
	Cluster	Completion Percentage	Genre 1 Encoded
count	10336.0	10336.000000	10336.000000
mean	1.0	0.790218	2.283669
std	0.0	0.203668	2.459784
min	1.0	0.000000	0.000000
25%	1.0	0.692308	0.000000
50%	1.0	0.862130	1.000000
75%	1.0	0.939735	3.000000
max	1.0	1.000000	9.000000

CLUSTER 1

	Cluster	Completion Percentage	Genre 1 Encoded
count	1962.0	1962.000000	1962.000000
mean	2.0	0.767646	30.234455
std	0.0	0.228820	3.208873
min	2.0	0.000000	24.000000
25%	2.0	0.662891	27.000000
50%	2.0	0.846865	32.000000
75%	2.0	0.935001	32.000000
max	2.0	1.000000	39.000000

CLUSTER 2

VISUALISING THE CLUSTERS



Sample Collection	Data Preparation	Exploratory Analysis	Analytic Visualisation	Algorithmic Optimisation	Information Presentation

ANALYSING THE CLUSTERS (GENRE 1)

Based on the median values, which are:

- Cluster 0: 0.858155
- Cluster 1: 0.862130
- Cluster 2: 0.846865

Cluster 1 has the best completion rate. It consists of these genres:

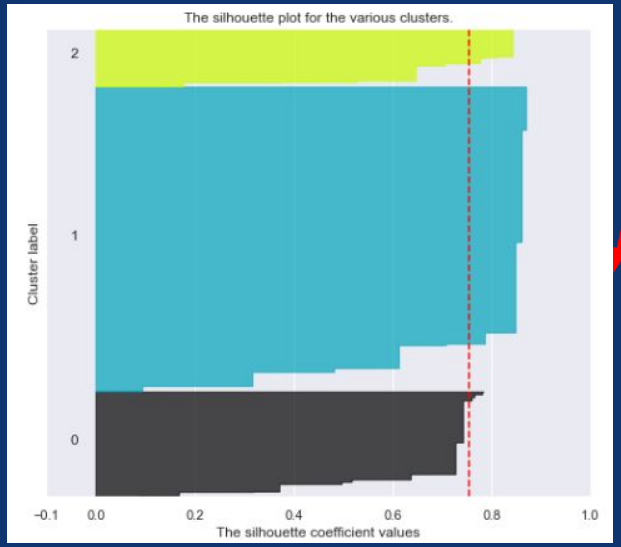
- 'Action', 'Adventure', 'Comedy', 'Drama', 'Game', 'Fantasy', 'Ecchi', 'Dementia', 'Cars', 'Demons'

EVALUATING ACCURACY

Silhouette Scores for Number of Clusters

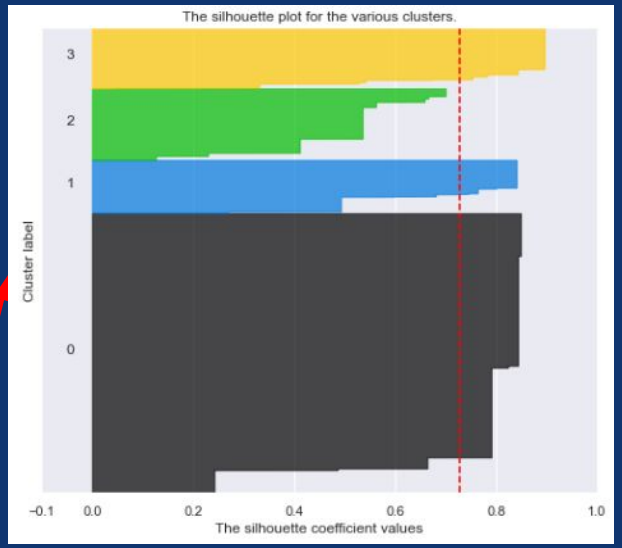
```
For n_clusters = 3 The average silhouette_score is : 0.7560270106514325
For n_clusters = 4 The average silhouette_score is : 0.7277634136593208
For n_clusters = 5 The average silhouette_score is : 0.7145792445303754
```

n_clusters = 3 has the best silhouette score of 0.756



Number of Clusters = 3

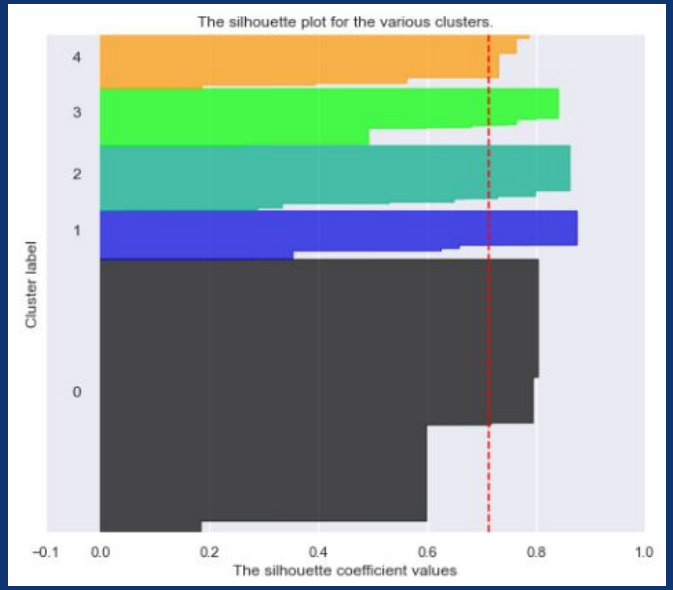
Number of Clusters = 4



Sample Collection	Data Preparation	Exploratory Analysis	Analytic Visualisation	Algorithmic Optimisation	Information Presentation

EVALUATING ACCURACY

Number of Clusters = 5



Sample Collection	Data Preparation	Exploratory Analysis	Analytic Visualisation	Algorithmic Optimisation	Information Presentation

GENRE 2 ANALYSIS

NOTE BEFORE CONTINUING...

Genres 2 to 5 has rows with null / empty entries. Hence, such values from Genres 2 to 5 need to be removed before further analysis.

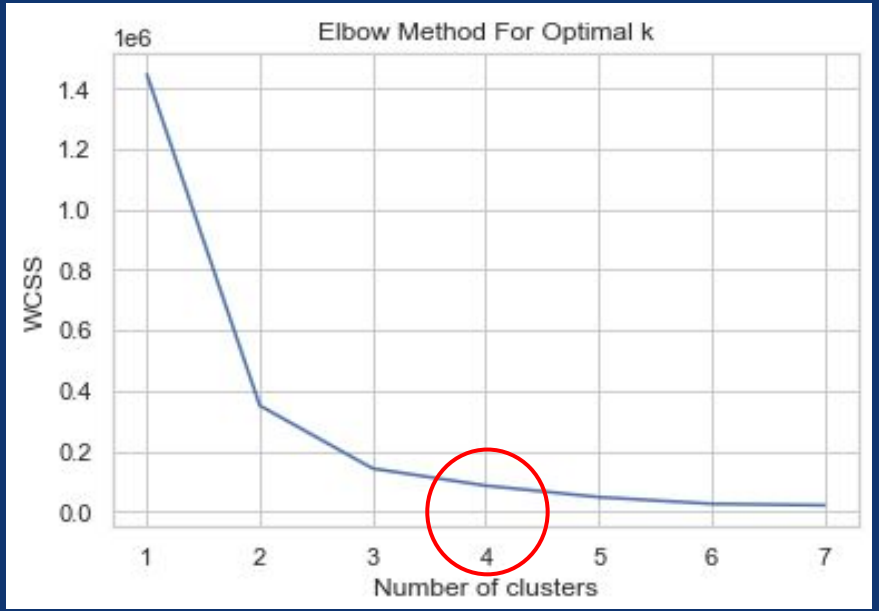
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 15862 entries, 0 to 17561
Data columns (total 2 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Genre 2                12655 non-null  object
 1   Completion Percentage  15862 non-null  float64
dtypes: float64(1), object(1)
memory usage: 887.8+ KB
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 12655 entries, 0 to 17561
Data columns (total 2 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Genre 2                12655 non-null  object
 1   Completion Percentage  12655 non-null  float64
dtypes: float64(1), object(1)
memory usage: 296.6+ KB
```

Example: Genre 2

FINDING APPROPRIATE K-VALUE

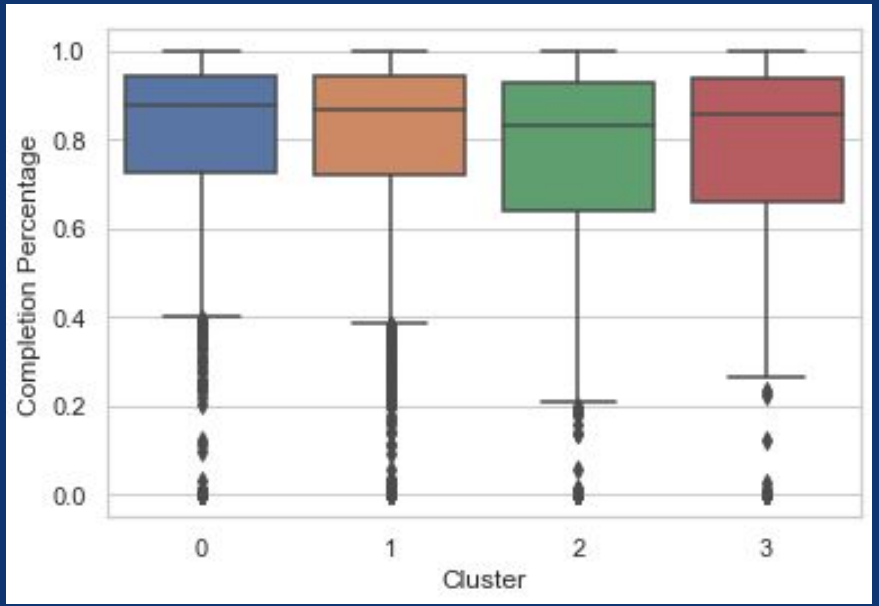
After label-encoding the genres in Genre 2, we conducted the elbow test, and concluded that k = 4 is the best value for the number of clusters



Sample Collection	Data Preparation	Exploratory Analysis	Analytic Visualisation	Algorithmic Optimisation	Information Presentation

VISUALISING THE CLUSTERS

We then broke the data in Genre 2 into 4 clusters, and obtained these box-plots for each cluster.



ANALYSING THE CLUSTERS (GENRE 2)

Based on the median values, which are:

- Cluster 0: 0.876052
- Cluster 1: 0.867407
- Cluster 2: 0.833805
- Cluster 3: 0.854839

Cluster 0 has the best completion rate. It consists of these genres:

- 'Sci-Fi', 'Romance', 'School', 'Psychological', 'Parody', 'Shoujo', 'Seinen', 'Samurai', 'Police'

EVALUATING ACCURACY

- We also conducted the silhouette test, and it concurred to the optimal k value that was obtained by the elbow test, which was k = 4.
- Again, this highlights the high accuracy of our K-Means clustering model that we have used

```
For n_clusters = 3 The average silhouette_score is : 0.6502731174975378
For n_clusters = 4 The average silhouette_score is : 0.651389554022867
For n_clusters = 5 The average silhouette_score is : 0.647018017356748
```

GENRE 3 ANALYSIS

NOTE BEFORE CONTINUING...

Genre 3 has rows with null / empty entries. Hence, these need to be removed before further analysis.

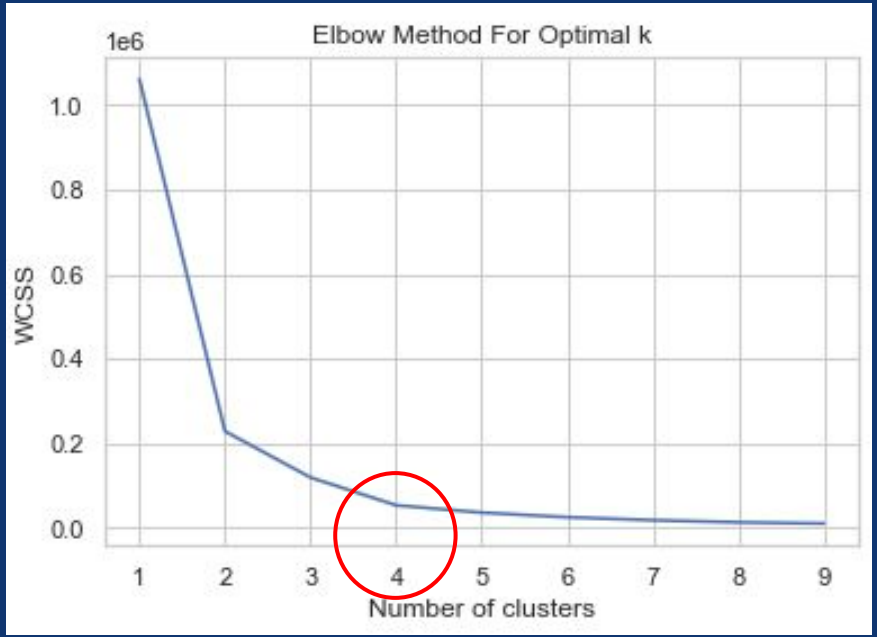
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 15862 entries, 0 to 17561
Data columns (total 2 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Genre 3               8666 non-null   object
 1   Completion Percentage 15862 non-null  float64
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 8666 entries, 0 to 17560
Data columns (total 4 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Cluster               8666 non-null   int32
 1   Genre 3               8666 non-null   category
 2   Completion Percentage 8666 non-null   float64
```

Sample Collection	Data Preparation	Exploratory Analysis	Analytic Visualisation	Algorithmic Optimisation	Information Presentation

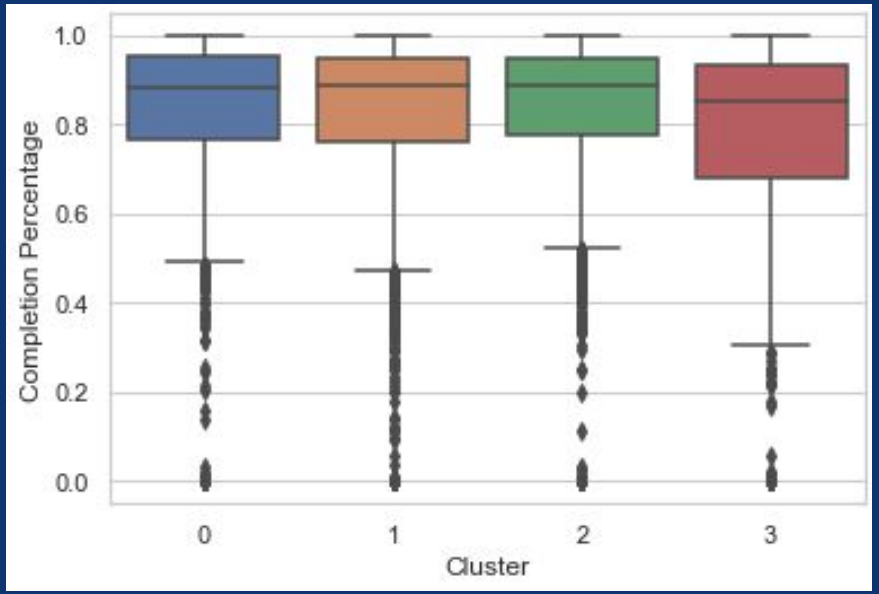
FINDING APPROPRIATE K-VALUE

After label-encoding the genres in Genre 3, we conducted the elbow test, and concluded that k = 4 is the best value for the number of clusters



VISUALISING THE CLUSTERS

We then broke the data in Genre 3 into 4 clusters, and obtained these box-plots for each cluster.



Sample Collection	Data Preparation	Exploratory Analysis	Analytic Visualisation	Algorithmic Optimisation	Information Presentation

ANALYSING THE CLUSTERS (GENRE 3)

Based on the median values, which are:

- Cluster 0: 0.884414
- Cluster 1: 0.889319
- Cluster 2: 0.885350
- Cluster 3: 0.850824

Cluster 1 has the best completion rate. It consists of these genres:

- 'Comedy', 'Adventure', 'Dementia', 'Demons', 'Drama', 'Fantasy', 'Harem', 'Ecchi', 'Cars', 'Game'

EVALUATING ACCURACY

- We also conducted the silhouette test, and it concurred to the optimal k value that was obtained by the elbow test, which was k = 4.
- Again, this highlights the high accuracy of our K-Means clustering model that we have used

```
For n_clusters = 3 The average silhouette_score is : 0.5931984229788559
For n_clusters = 4 The average silhouette_score is : 0.6456642281693862
For n_clusters = 5 The average silhouette_score is : 0.6437619602222499
```

GENRE 4 ANALYSIS

NOTE BEFORE CONTINUING...

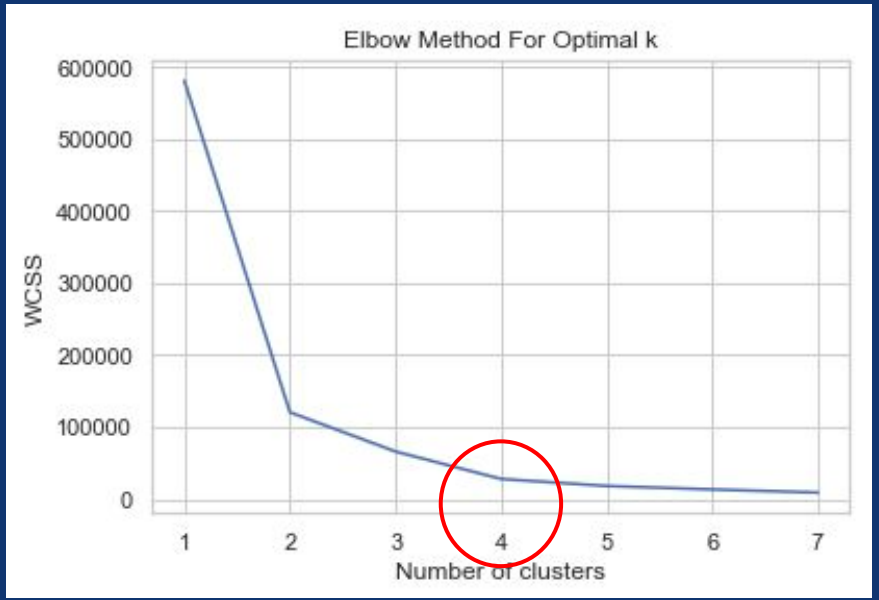
Genre 4 has rows with null / empty entries. Hence, these need to be removed before further analysis.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 15862 entries, 0 to 17561
Data columns (total 2 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Genre 4               5344 non-null   object
 1   Completion Percentage 15862 non-null  float64
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5344 entries, 0 to 17549
Data columns (total 2 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Genre 4               5344 non-null   object
 1   Completion Percentage 5344 non-null   float64
```

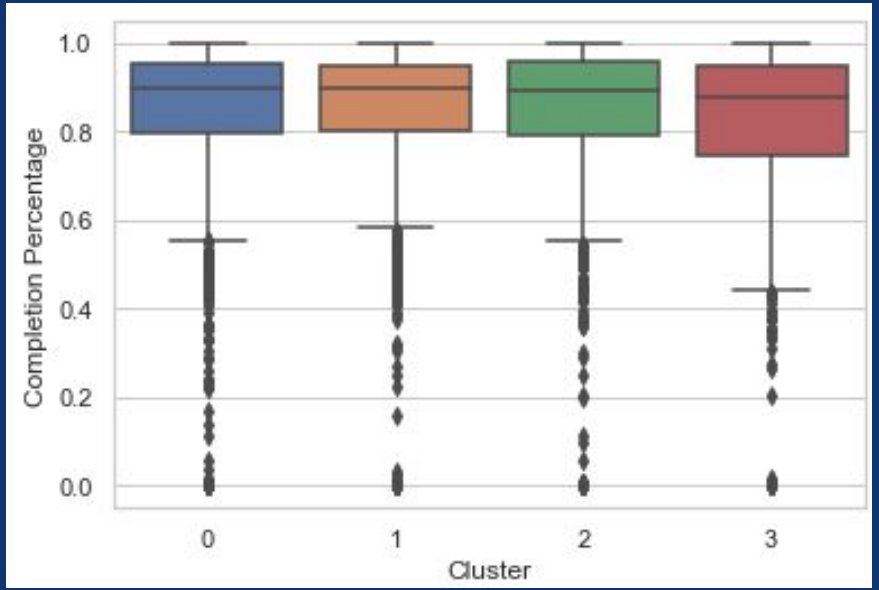
FINDING APPROPRIATE K-VALUE

After label-encoding the genres in Genre 4, we conducted the elbow test, and concluded that k = 4 is the best value for the number of clusters



VISUALISING THE CLUSTERS

We then broke the data in Genre 4 into 4 clusters, and obtained these box-plots for each cluster.



Sample Collection	Data Preparation	Exploratory Analysis	Analytic Visualisation	Algorithmic Optimisation	Information Presentation

ANALYSING THE CLUSTERS (GENRE 4)

Based on the median values, which are:

- Cluster 0: 0.899512
- Cluster 1: 0.896596
- Cluster 2: 0.893880
- Cluster 3: 0.876876

Cluster 0 has the best completion rate. It consists of these genres:

- 'Drama', 'Comedy', 'Ecchi', 'Historical', 'Adventure', 'Fantasy', 'Demons', 'Harem', 'Game', 'Dementia', 'Cars'

EVALUATING ACCURACY

- From the Silhouette Analysis it also concurred to the optimal k value that was obtained by the elbow test, which was K = 4.
- Highlights the high accuracy of our K-Means clustering model that we have used

```
For n_clusters = 3 The average silhouette_score is : 0.6077887023034122
For n_clusters = 4 The average silhouette_score is : 0.6487190690553434
For n_clusters = 5 The average silhouette_score is : 0.6476195298351732
```

GENRE 5 ANALYSIS

NOTE BEFORE CONTINUING...

Genre 5 has rows with null / empty entries. Hence, these need to be removed before further analysis.

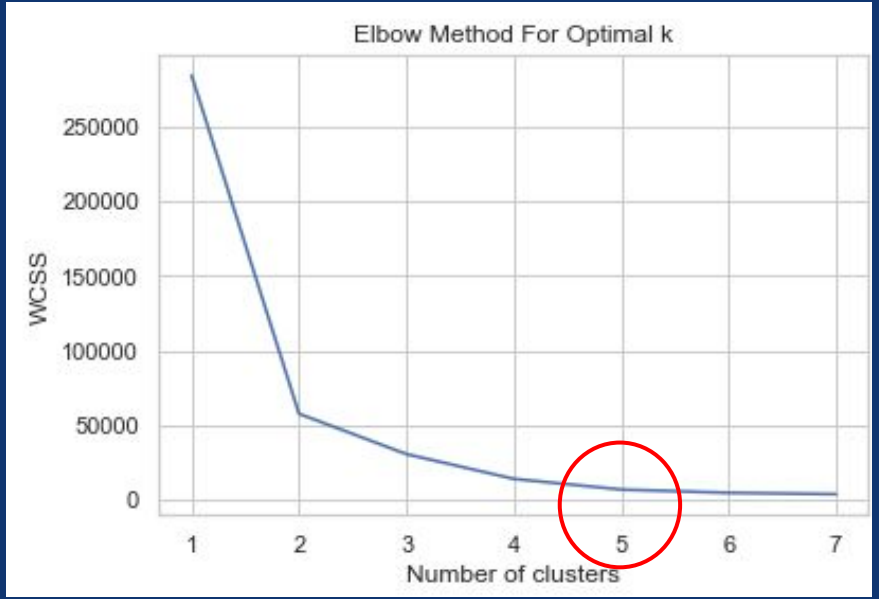
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 15862 entries, 0 to 17561
Data columns (total 2 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Genre 5               2870 non-null   object
 1   Completion Percentage  15862 non-null  float64
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2870 entries, 0 to 17544
Data columns (total 2 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Genre 5               2870 non-null   object
 1   Completion Percentage  2870 non-null   float64
```

Sample Collection	Data Preparation	Exploratory Analysis	Analytic Visualisation	Algorithmic Optimisation	Information Presentation

FINDING APPROPRIATE K-VALUE

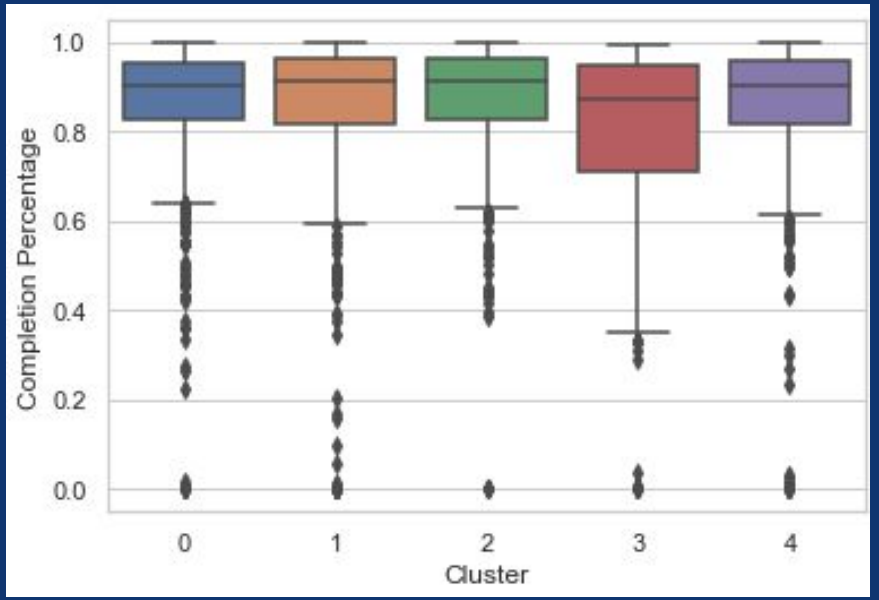
After label-encoding the genres in Genre 5, we conducted the elbow test, and concluded that k = 5 is the best value for the number of clusters



Sample Collection	Data Preparation	Exploratory Analysis	Analytic Visualisation	Algorithmic Optimisation	Information Presentation

VISUALISING THE CLUSTERS

We then broke the data in Genre 5 into 5 clusters, and obtained these box-plots for each cluster.



ANALYSING THE CLUSTERS (GENRE 5)

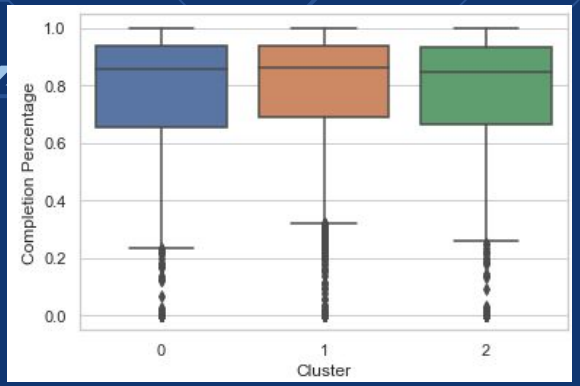
Based on the median values, which are:

- Cluster 0: 0.900137
- Cluster 1: 0.914270
- Cluster 2: 0.911625
- Cluster 3: 0.870368
- Cluster 4: 0.904849

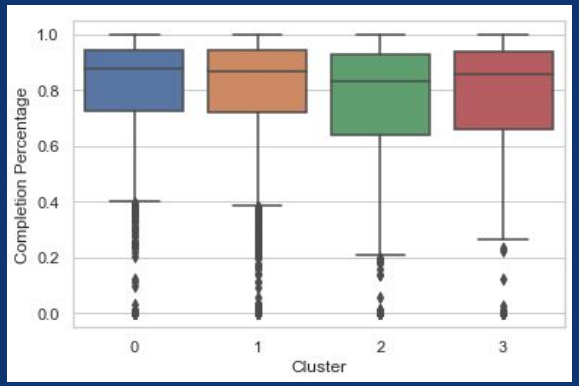
Cluster 1 has the best completion rate. It consists of these genres:

- 'Drama', 'Fantasy', 'Ecchi', 'Comedy', 'Historical', 'Dementia', 'Demons', 'Harem', 'Game'

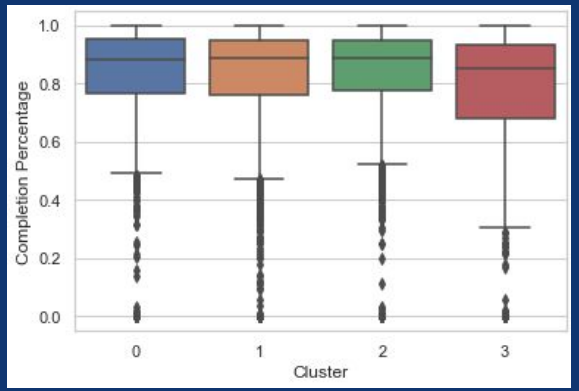
BOXPLOTS FOR GENRES 1 TO 5



Genre 1



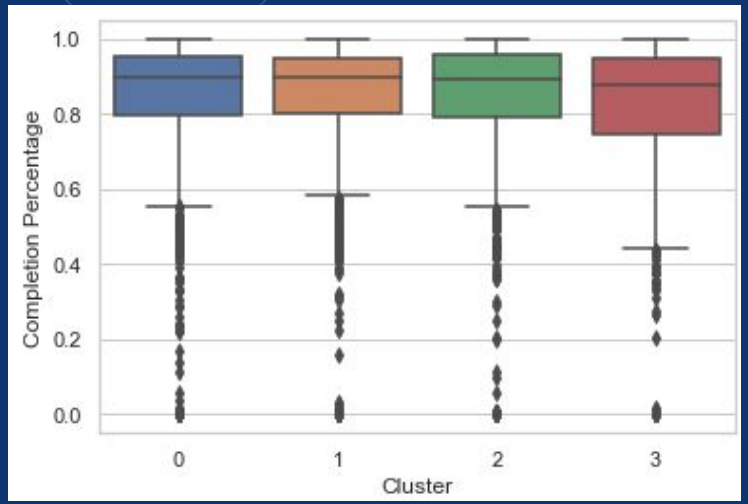
Genre 2



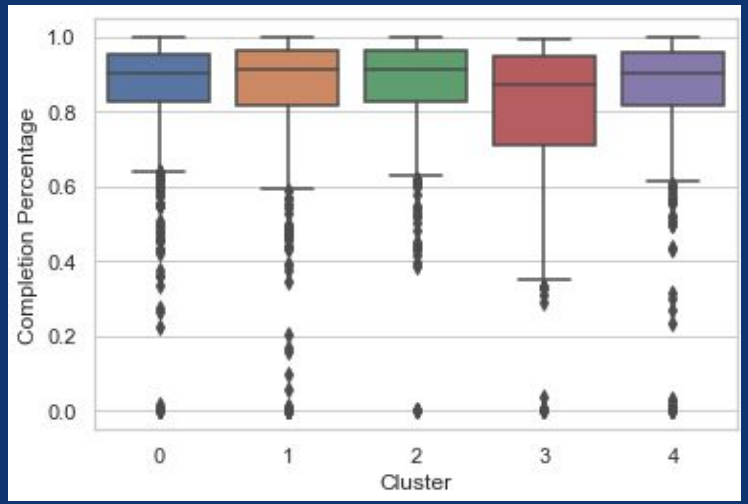
Genre 3

Sample Collection	Data Preparation	Exploratory Analysis	Analytic Visualisation	Algorithmic Optimisation	Information Presentation

BOXPLOTS FOR GENRES 1 TO 5



Genre 4



Genre 5

ANALYSING THE BEST CLUSTERS (GENRES 1 - 5)

- Based on our analysis, these are the median values of completion rates for the top clusters of each of the 5 genre columns, Genres 1 to 5:
    - Genre 1: Cluster 1 (86.21%)
    - Genre 2: Cluster 0 (87.61%)
    - Genre 3: Cluster 1 (88.93%)
    - Genre 4: Cluster 0 (89.95%)
    - Genre 5: Cluster 1 (91.43%)

EVALUATING ACCURACY

- We also conducted the silhouette test, and it concurred to the optimal k value that was obtained by the elbow test, which was k = 5.
- Again, this highlights the high accuracy of our K-Means clustering model that we have used

```
For n_clusters = 3 The average silhouette_score is : 0.6157314563253975
For n_clusters = 4 The average silhouette_score is : 0.6596540035412051
For n_clusters = 5 The average silhouette_score is : 0.6981317361081257
```

INTRODUCING K-MODES

- K-Modes clustering is also an unsupervised learning algorithm
- Unlike K-Means which uses Euclidean distance to cluster, K-Modes uses dissimilarities (total mismatches) between data points. The lesser the dissimilarities, the more similar the data points are.
- Uses Modes instead of Means

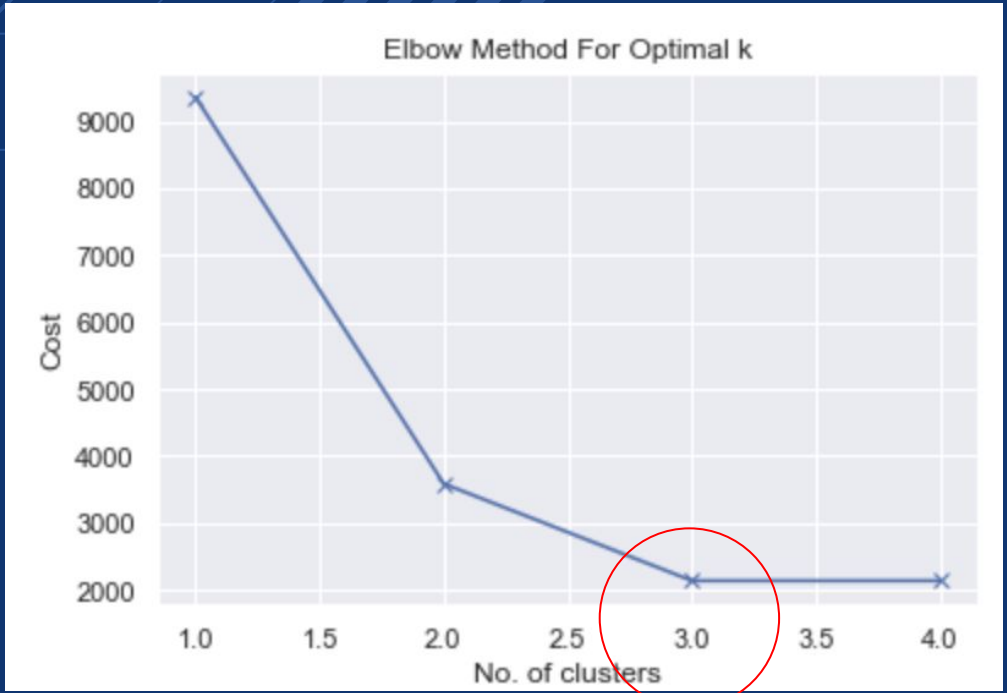
Sample Collection	Data Preparation	Exploratory Analysis	Analytic Visualisation	Algorithmic Optimisation	Information Presentation

LET'S TAKE A LOOK AT RATINGS...

- Usage of K-Modes model (Dissimilarities instead of Distance which K-Means utilises)
- Challenges in evaluating 'Ratings':
  - Similar to all the aforementioned input variables
  - Usage of new model, K-Modes

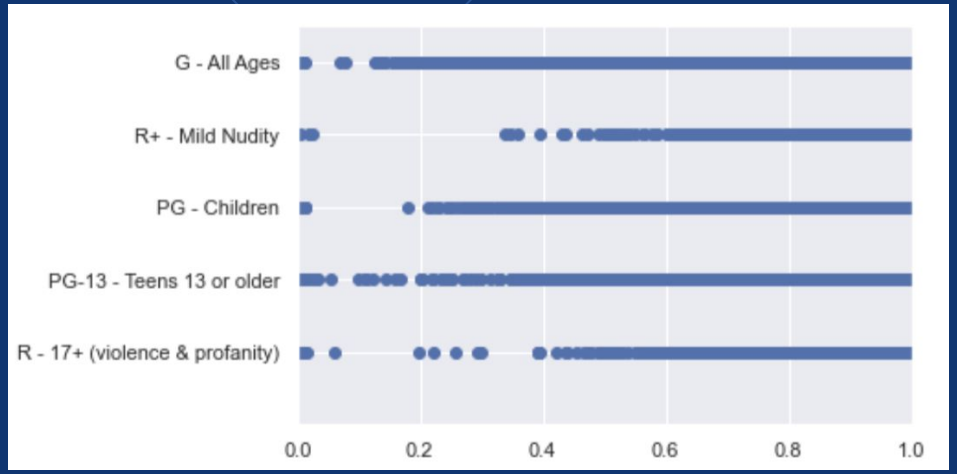
Sample Collection	Data Preparation	Exploratory Analysis	Analytic Visualisation	Algorithmic Optimisation	Information Presentation

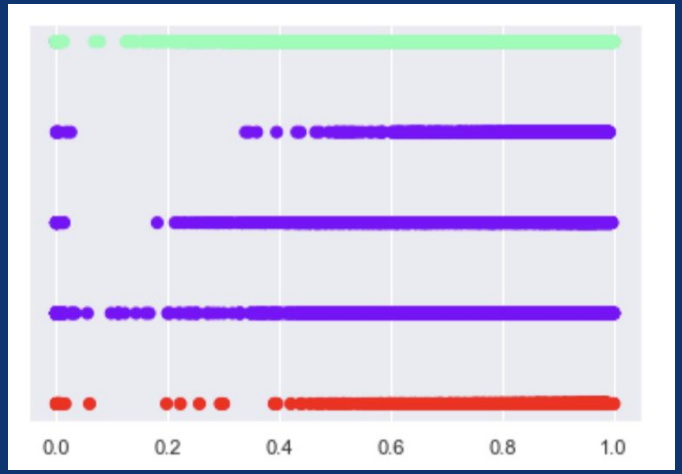
FINDING APPROPRIATE K-VALUE



Sample Collection	Data Preparation	Exploratory Analysis	Analytic Visualisation	Algorithmic Optimisation	Information Presentation

CONDUCTING K-MODES USING K = 3





BEFORE

AFTER

Sample Collection	Data Preparation	Exploratory Analysis	Analytic Visualisation	Algorithmic Optimisation	Information Presentation

ANALYSING THE CLUSTERS

	Cluster	Rating	Completion Percentage
2	0	PG-13 - Teens 13 or older	0.897122
3	0	PG-13 - Teens 13 or older	0.814715
4	0	PG - Children	0.796038
5	0	PG-13 - Teens 13 or older	0.752271
6	0	PG-13 - Teens 13 or older	0.779701
...	...	...	...
17538	0	PG-13 - Teens 13 or older	1.000000
17540	0	PG-13 - Teens 13 or older	0.160000
17549	0	PG-13 - Teens 13 or older	0.363636
17552	0	PG-13 - Teens 13 or older	0.986045
17560	0	PG-13 - Teens 13 or older	0.400000

ANALYSING THE CLUSTERS

	Cluster	Rating	Completion Percentage
2	0	PG-13 - Teens 13 or older	0.897122
3	0	PG-13 - Teens 13 or older	0.814715
4	0	PG - Children	0.796038
5	0	PG-13 - Teens 13 or older	0.752271
6	0	PG-13 - Teens 13 or older	0.779701
...	...	...	...
17538	0	PG-13 - Teens 13 or older	1.000000
17540	0	PG-13 - Teens 13 or older	0.160000
17549	0	PG-13 - Teens 13 or older	0.363636
17552	0	PG-13 - Teens 13 or older	0.986045
17560	0	PG-13 - Teens 13 or older	0.400000

	Cluster	Rating	Completion Percentage
0	1	R - 17+ (violence & profanity)	0.879720
1	1	R - 17+ (violence & profanity)	0.987182
15	1	R - 17+ (violence & profanity)	0.791165
17	1	R - 17+ (violence & profanity)	0.827863
21	1	R - 17+ (violence & profanity)	0.986305
...	...	...	...
17494	1	R - 17+ (violence & profanity)	0.000000
17513	1	R - 17+ (violence & profanity)	0.962311
17519	1	R - 17+ (violence & profanity)	0.300000
17543	1	R - 17+ (violence & profanity)	0.294118
17544	1	R - 17+ (violence & profanity)	0.222222

	Cluster	Rating	Completion Percentage
54	2	G - All Ages	0.908773
82	2	G - All Ages	0.721201
122	2	G - All Ages	0.891114
140	2	G - All Ages	0.818306
155	2	G - All Ages	0.839234
...	...	...	...
17447	2	G - All Ages	1.000000
17448	2	G - All Ages	1.000000
17449	2	G - All Ages	1.000000
17452	2	G - All Ages	0.611111
17505	2	G - All Ages	0.983871

ANALYSING THE CLUSTERS

```
#CLUSTER 0 STATISTICS
Cluster0DF.describe()
```

	Cluster	Completion Percentage
count	8514.0	8514.000000
mean	0.0	0.820757
std	0.0	0.183817
min	0.0	0.000000
25%	0.0	0.757027
50%	0.0	0.885728
75%	0.0	0.947540
max	0.0	1.000000

```
#CLUSTER 1 STATISTICS
Cluster1DF.describe()
```

	Cluster	Completion Percentage
count	5773.0	5773.000000
mean	1.0	0.735516
std	0.0	0.221013
min	1.0	0.000000
25%	1.0	0.582418
50%	1.0	0.802395
75%	1.0	0.919383
max	1.0	1.000000

```
#CLUSTER 2 STATISTICS
Cluster2DF.describe()
```

	Cluster	Completion Percentage
count	1147.0	1147.000000
mean	2.0	0.862097
std	0.0	0.155416
min	2.0	0.000000
25%	2.0	0.824079
50%	2.0	0.912369
75%	2.0	0.959550
max	2.0	1.000000

ANALYSING THE CLUSTERS

```
#UNIQUE AGE RATINGS IN CLUSTER 0
Cluster0DF.Rating.unique()
```

```
array(['PG-13 - Teens 13 or older', 'PG - Children', 'R+ - Mild Nudity'],
      dtype=object)
```

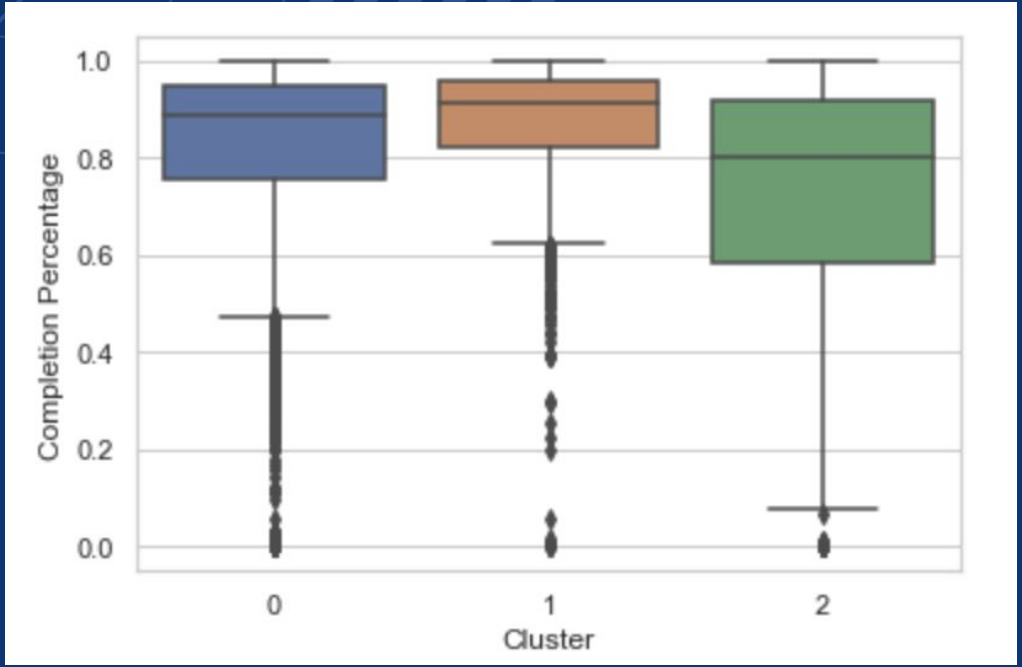
```
#UNIQUE AGE RATINGS IN CLUSTER 1
Cluster1DF.Rating.unique()
```

```
array(['R - 17+ (violence & profanity)'], dtype=object)
```

```
#UNIQUE AGE RATINGS IN CLUSTER 2
Cluster2DF.Rating.unique()
```

```
array(['G - All Ages'], dtype=object)
```

VISUALISING THE CLUSTERS



Sample Collection	Data Preparation	Exploratory Analysis	Analytic Visualisation	Algorithmic Optimisation	Information Presentation

EVALUATING ACCURACY

- Similar to our K-Means analysis, the Elbow Test used for K-Modes also returned 3 to be the most optimal number of clusters

Sample Collection	Data Preparation	Exploratory Analysis	Analytic Visualisation	Algorithmic Optimisation	Information Presentation

05

PRESENTATION OF FINDINGS

FROM HIGH SCORE PERCENTAGE

- High score percentage is **a bad indicator** for Completion Percentage.
  - Further supported by **low correlation value**, even after removing of skewness
- Debunks common stereotype that **High Scores = High Completion Percentage**

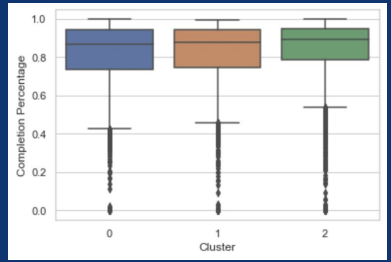
	High Score Percentage	Completion Percentage
High Score Percentage	1.00000	0.19861
Completion Percentage	0.19861	1.00000

	Completion Percentage Boxcox
Completion Percentage Boxcox	1.000000
High Score Percentage Boxcox	0.203206

	High Score Percentage Boxcox
Completion Percentage Boxcox	0.203206
High Score Percentage Boxcox	1.000000

FROM STUDIOS

- Cluster 2 seems to be the best cluster
  - **Highest Median Completion Rate of 88.7%**, compared to 86.9% in Cluster 0 and 87.5% in Cluster 1.
  - **Smallest Inter-Quartile Range at 17.2%**, compared to 20.6% in Cluster 0 and 19.3% in Cluster 1.
- Cluster 2's top 5 studios (in terms of anime produced): **J.C Staff, Madhouse, Production I.G, OLM and Nippon Animation.**
- Suggests that people who watch anime or films made from Cluster 2's studios, are more likely to complete it compared to other studios.



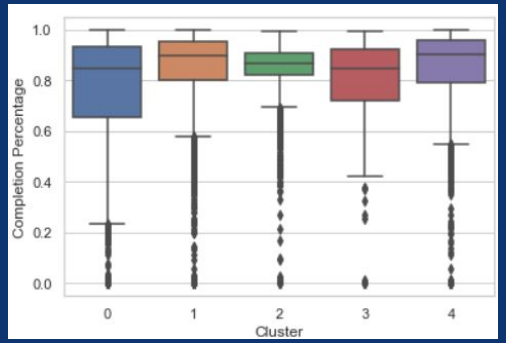
Sample Collection	Data Preparation	Exploratory Analysis	Analytic Visualisation	Algorithmic Optimisation	Information Presentation

FROM SOURCE

- Cluster 4 is the best cluster:
  - **Highest median Completion Rate of 90.22%** approximately.

- The sources in cluster 4 are **Light Novel, Game and Digital Manga.**

- Suggests that people are more likely to complete an anime if it originated from these sources



FROM GENRES

- Compiled the top genres across each Genre column

Genre 1: 'Action', 'Adventure', **'Comedy'**, **'Drama'**, **'Game'**, **'Fantasy'**, 'Ecchi', 'Dementia', 'Cars', 'Demons'
Genre 2: 'Sci-Fi', 'Romance', 'School', 'Psychological', 'Parody', 'Shoujo', 'Seinen', 'Samurai', 'Police'
Genre 3: '**Comedy**', 'Adventure', 'Dementia', 'Demons', **'Drama'**, **'Fantasy'**, 'Harem', 'Ecchi', 'Cars', **'Game'**
Genre 4: **'Drama'**, **'Comedy'**, 'Ecchi', 'Historical', 'Adventure', **'Fantasy'**, 'Demons', 'Harem', **'Game'**, 'Dementia', 'Cars'
Genre 5: **'Drama'**, **'Fantasy'**, 'Ecchi', 'Comedy', 'Historical', 'Dementia', 'Demons', 'Harem', 'Game'

- Due to their frequency of appearance, we believe that Comedy, Drama, Fantasy, and Game are the best genres in terms of completion rates.

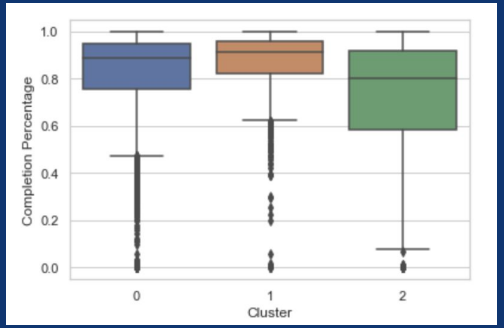
Sample Collection	Data Preparation	Exploratory Analysis	Analytic Visualisation	Algorithmic Optimisation	Information Presentation

FROM RATINGS

- Cluster 1 seems to be the best cluster:
  - Highest Median Completion Rate of 91.2%, compared to 88.6% in Cluster 0 and 80.2% in Cluster 1.
  - Smallest Inter-Quartile Range at 13.5%, compared to 19.1% in Cluster 0 and 33.7% in Cluster 1.

- Cluster 1 contains the **R-17+ (Violence and Profanity)** age rating.

- This suggests that people who watch anime or films with this age rating, are more likely to complete it compared to other media of other age ratings



06

CONCLUSION

CONCLUSION

Benefits to our analysis:

- Informative
- Debunked common stereotypes
- Allows existing studios or new ones with potential to be more aware on popular genres, leading to high completion rates

THANKS!

CONTRIBUTIONS

NOEL WEE

EDA, K-Means,
K-Modes, Analysis

THARUN SWAROOP

EDA, Linear Regression,
Polynomial Regression,
Analysis

RESOURCES

Agarwal, A. (2018). *Polynomial Regression - Towards Data Science*. Towards Data Science. from https://towardsdatascience.com/polynomial-regression-bbe8b9d97491[Accessed on April 3]

Kaggle. *Anime Recommendations Dataset* https://www.kaggle.com/datasets/hernan4444/anime-recommendation-database-2020?select=watching_status.csv [Accessed on 12 March]

Imad Dabbura. *K-means Clustering: Algorithm, Applications, Evaluation Methods, and Drawbacks* https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a [Accessed on 10 April]

Radečić, D. (2022). *Top 3 Methods for Handling Skewed Data - Towards Data Science*. Towards Data Science. From https://towardsdatascience.com/top-3-methods-for-handling-skewed-data-1334e0debf45#:%7E:text=Log%20transformation%20is%20most%20likely,coefficient%20of%205.2%20to%200.4. [Accessed on April 8, 2022]

RESOURCES

Harika, B. (2021) *K-Modes Clustering Algorithm for Categorical Data* https://www.analyticsvidhya.com/blog/2021/06/kmodes-clustering-algorithm-for-categorical-data/ [Accessed on 15 April]