Final Project

Aaron J. Brown

DSC 630: Predictive Modeling
Andrew Hua
Fall 2023

Academic Report

CONTENTS

INTRODUCTION

The purpose of this project is to determine factors that contribute to an Japanese-produced animated productions' (shortened to "anime" from this point on) success based on user ratings or "score", which is a score between 1 to 10 provided by users. Having an understanding of fan-based estimation of an anime's rating and factors that amalgamate to ratings is proves useful from both perspectives: a consumer's, and an investor's — providing insight into potential recommendations for future viewing, as well as determining which productions to invest in, especially with the current success of the Anime Global Market, which has amassed an estimated global market value of over $27 billion, with expectations of close to an 50% increase by the year 2028. The recent surge in the use of streaming services like Netflix, Hulu, and the amazing Crunchyroll, is expected to be a major factor in this exponential growth, adding to the usefulness of analyses like this which utilizes such data.

As of 2021, anime series are recognized as the most demanded foreign-language TV shows in the United States accounting for 30.5% of the market share. In 2021 more than half of Netflix's global members watched anime. In addition, anime and manga alongside many other parts of Japanese pop culture have helped Japan to gain a positive worldwide image and improve its relations with other countries.

This project utilizes only data obtained from website-data, such as members, anime-type, and favorites to provide predictions of success based on the target variable — score; this project does not make use of financial and market data, like number of sales or market value. Though lacking in market/financial data, this project will aim to provide a user-based

perspective and prediction of “success”, which will provide insight on anime consumers’ likes/ dislikes, as well.

DATA SOURCING

The dataset used in this project was obtained from Kaggle - it was scraped from the MyAnimeList website, which consists of data collected from millions of users over tens of thousands productions. Features retained from the data for the purpose of this project include:

- title: the name of the animated productions in the data.
- episodes: the number of episodes for the productions.
- type: the classification of the productions (movie, series, etc.).
- favorites: the total number of favorites the titles have received.
- score: (1-10) average rank given by the members or subscribers.
- members: total number of subscribers to the titles.
- studios: the production studio responsible for the titles’ adaptation.
- demographic: targeted age/gender group.
- genres: listed genres for each title, can have multiple.
- synopsis: a short overview of the plot for each title.

DATA PREPARATION

The data contained over 8000 missing values in the "score" column to be removed, as well as some elements containing an “Unknown” string, which was treated as a missing value in this project. Though the amount of missing values is large, the data still retained over 15,200 rows, allowing its use. Dummy variables were created for categorical variable "type". was handled by creating dummy variables. Subsequent to controlling for NaN values in the trimmed

data, steps to further clean the data, as well as creating an additional dataset for the Anime Recommendation program which piggybacks on this project, included:

A. Removing punctuation from the numerical features ("members" and "favorites") and converting them to integers/floats, as they were originally strings.
B. Replacing the “Unknown” number of episodes with the mean.
C. Removing commas, excess spacing, and other punctuation from the "genres" and "synopsis" features.
D. Tokenizing the text of the features containing strings.
E. Creating a new feature "tags" that contains all of the text data from the "type", "studios", “demographic", "genres" and “synopsis” columns for the recommendation system.
F. Joining and vectorizing the text of the new “tags” feature to measure the cosine similarity between the vectors.

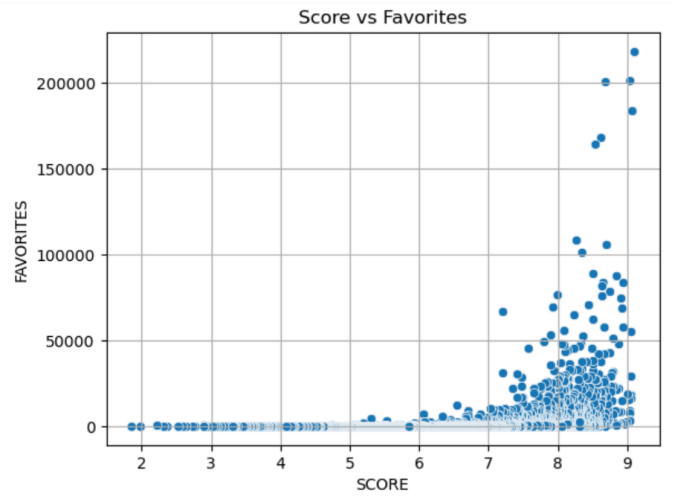
METHODOLOGY

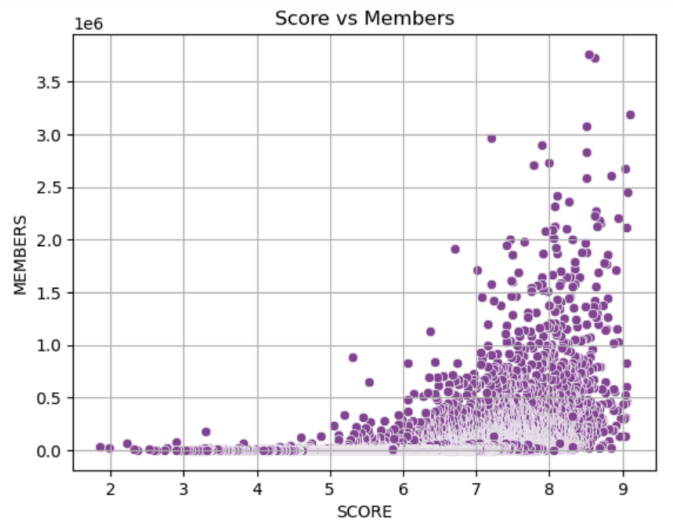
The model selected for this project is the Linear Regression Model. This model was considered over other options primarily due to its ease in understanding and deployment. The model's flexibility and adaptability are also considered, especially with the inclusion of our categorical variables. This model, however, has its limits such as overfitting or under-fitting due to its sensitivity to outliers. Outliers proved not to be a significant issue in this project as the target is a range between 1-10.

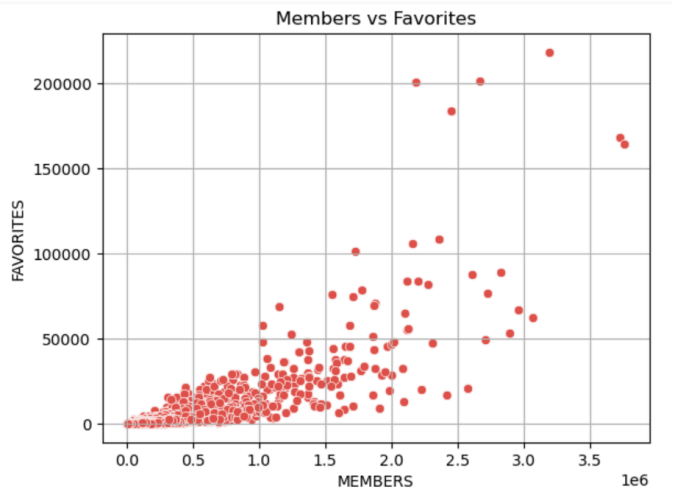
A correlation matrix was used to visually digest the relationships between the variables. We can see that 'members' and 'favorites' have a pretty clear relationship with the target

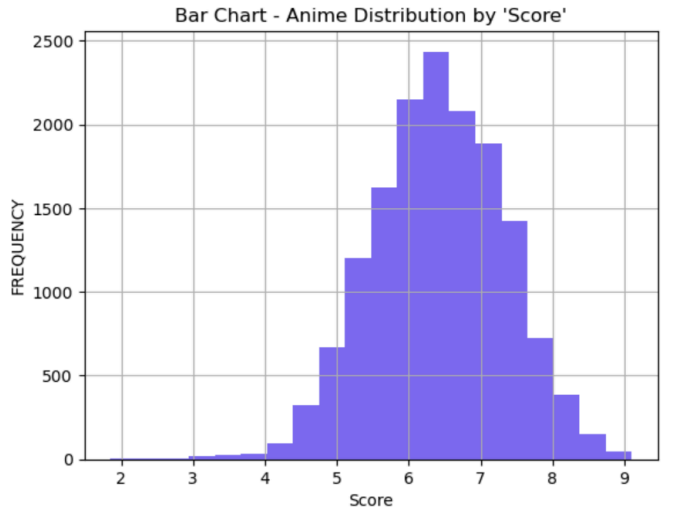
variable 'score'. This is expected as the more members an anime has, the more favorites it can get. Higher scoring anime may also have more members as well.

Additional visualizations to understand the relationships between the target and “members” and “favorites” can be seen below, in the form of scatterplots. As expected, higher scores seem to relate with increasing members. To complete the trio of scatterplots, a comparison of 'members' and 'favorites' is made. These three scatterplots visually outline the relationships seen above in the correlation matrix. We see that increasing members increase favorites and scores, so determining ways to get users to become more inclined to become members of titles would directly increase favorites and scores.









The distribution of the target “score” can also be seen above, with the vast majority of the data concentrated in the upper half of the score scale — a slight left-skew in the distribution; we also see that the vast majority of scores are above 6.

Using the numerical variables subsetted into a new data-frame, the data was split into training and test sets, with a test size of 25%, and where “score” is the target variable. Using the Linear Regression model to generate predictions of “score” followed — leading to a calculation of the RMSE to determine the performance of the model: 0.8209. The model appears to perform well, suggesting that the linear regression model is decent at predicting score; it seems that ‘score' can be predicted using features of this data.

CONCLUSION

From the Linear Regression model, we received a pretty low RMSE! This could indicate that the model fits the data well and has more precise predictions. Seeking methods to increase the number of members an anime has seems to be the most influential way to increase score. Fan service is surely the key to popular anime, so increasing fan interaction outside of normal viewing could also positively affect the titles' scores. Considering the implications and assumptions made by the Linear Regression Model, it is possible that the low RMSE could be the result of overfitting; further analysis with other models could be useful, but this model does

Splitting the data into a training and test set, where 'score' is the target.

```
import sklearn
from sklearn.model_selection import train_test_split
```

```
target = mal_model_data.score
features = mal_model_data.drop('score',axis=1)

x_train, x_test, y_train, y_test = train_test_split(features, target
                                                    , test_size=0.25, random_state=42)

print("shape of - features_train", x_train.shape)
print("shape of - target_train", y_train.shape)
print("shape of - features_test", x_test.shape)
print("shape of - target_test", y_test.shape)
```

```
shape of - features_train (11469, 4)
shape of - target_train (11469,)
shape of - features_test (3824, 4)
shape of - target_test (3824,)
```

Using Linear Regression model to generate 'score' predictions.

```
from sklearn.linear_model import LinearRegression
```

```
linreg_model = LinearRegression()

linreg_model.fit(x_train, y_train)
```

```
LinearRegression()
```

```
score_predictions = linreg_model.predict(x_test)
score_predictions
```

```
array([6.39154534, 6.60276127, 6.35411345, ..., 5.91440097, 6.42710106,
       7.08204085])
```

Calculation of RMSE to determine the performance for the regression model.

```
from sklearn import metrics
```

```
RMSE = (metrics.mean_squared_error(y_test, score_predictions, squared=False))
print('Root Mean Squared Error (RMSE): %.4f' % RMSE)
```

```
Root Mean Squared Error (RMSE): 0.8209
```

satisfy the objective of this project, and can be used for decision making, and building recommendations, as mentioned in the rise of this report — a small anime recommendation program is included in this project.

In [ ]:

AARON J. BROWN

DSC 630 - Fall 2023

Week 9 - Milestone 4: Finalizing Your Results

In Milestone 4, most of the technical work for the project should be done. You should include the information from Milestone 3 and address the following additional items:

- Explain your process for prepping the data
- Build and evaluate at least one model
- Interpret your results
- Begin to formulate a conclusion/recommendations

Changes and Alterations from Milestone #3.

After Milestone #3, it was determined that my features would not be great predictors of popularity - this led to the search for new data. For Milestone #4, data scraped from MyAnimeList.com is used as a replacement for the CrunchyRoll dataset. An additional adjustment to this report includes: changing the target variable from 'popularity' to 'score' - this new target variable is also a great reference to an anime's success as it relates to the amount of 'members' and 'favorites' that the production receives from the site. This new data is much larger and has more missing values, though, the size of the data compensates. Model selection for this milestone is also changed to Linear Regression, which improved results significantly compared to the previous milestone.

Importing Libraries

In [1]:
```
import os
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

Data from MyAnimeList is replacing CrunchyRoll data.

In [2]:
```
mal_df = pd.read_csv('/Users/aaronbrown/Documents/Classwork/DSC 630 - Predic
```

In [3]:
```
mal_df.head(3)
```

Out[3]:

	Unnamed: 0	title	episodes	status	theme	demographic	
0	0	Fullmetal Alchemist: Brotherhood	64	Finished Airing	Military	Shounen	Action,Adventure,Drama
1	1	Steins;Gate	24	Finished Airing	Unknown	Unknown	Drama,Sci-Fi,S
2	2	Bleach: Sennen Kessen-hen	13	Finished Airing	Unknown	Shounen	Action,Adventure

In [4]:
```
mal_df.columns.values
```

Out[4]:
```
array(['Unnamed: 0', 'title', 'episodes', 'status', 'theme',
       'demographic', 'genres', 'type', 'favorites', 'popularity', 'rank',
       'score', 'members', 'synopsis', 'aired', 'duration', 'premiered',
       'studios'], dtype=object)
```

Unneeded features like "status", "popularity" (which is only a rank of the number of 'members' in this data), and "synopsis" are removed. Features like "title", "type", and "studio" are kept in this dataframe for aiding visualizations. Dummy Variables will be generated for "type" variable to include in modeling.

Removing unneeded columns from the data, keeping columns:

- title
- episodes
- type
- favorites
- score
- members
- studios

In [5]:
```
mal_data = mal_df[['title', 'episodes', 'type', 'favorites', 'score', 'membe
mal_data.tail(3)
```

Out[5]:

	title	episodes	type	favorites	score	members	studios
24259	Seikoujo: Haitoku no Biden Dorei	2	OVA	10	5.86	4,130	Studio Jam
24260	Chikan Juunin-tai The Animation	5	OVA	7	5.86	2,855	Sugar Boy
24261	Gibo	2	OVA	16	5.86	5,535	Y.O.U.C

Removing Missing Values

Our target variable "score" has quite a few missing values, but since the dataset is so large, this should not be an issue.

In [6]:
```
mal_data.isnull().sum()
```

Out[6]:
```
title            0
episodes         0
type             0
favorites        0
score         8968
members          0
studios          0
dtype: int64
```

In [7]:
```
mal_data.dropna(inplace=True)
mal_data.isnull().sum()
```

Out[7]:
```
title        0
episodes     0
type         0
favorites    0
score        0
members      0
studios      0
dtype: int64
```

Most of the numerical variables are not float or int which will present an issue in the future - the next few steps will handle this by removing punctuation (if necessary) and converting to float/int.

In [8]:
```
mal_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 15294 entries, 0 to 24261
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   title      15294 non-null  object 
 1   episodes   15294 non-null  object 
 2   type       15294 non-null  object 
 3   favorites  15294 non-null  object 
 4   score      15294 non-null  float64
 5   members    15294 non-null  object 
 6   studios    15294 non-null  object 
dtypes: float64(1), object(6)
memory usage: 955.9+ KB
```

Removing Punctuation.

In [9]:
```
'Libraries needed for text cleaning functions.'
import string
import nltk
```

In [10]:
```
## "punctuation_be_gone" function removes punctuation and special characters

def punctuation_be_gone(text_only):
    for punctuation in string.punctuation:
            text_only = text_only.replace(punctuation, '') # <=replace with
    return text_only
```

Removing punctuation from the 'favorites' and 'members' values and converting to integers.

In [11]:
```
mal_data['favorites'] = mal_data['favorites'].apply(punctuation_be_gone)
mal_data['favorites'] = mal_data['favorites'].astype(int)

mal_data['members'] = mal_data['members'].apply(punctuation_be_gone)
mal_data['members'] = mal_data['members'].astype(int)
```

Handling "Unknown" Values.

There are some rows in the 'episodes' and 'type' columns with the word "Unknown" in lieu of missing the value.

Mean number of episodes (excluding "unknown" values): 11.599.

Replacing "Unknown" with the mean and converting to float.

In [12]:
```
mal_data['episodes'].replace('Unknown', '11.599', inplace=True)
mal_data['episodes'] = mal_data['episodes'].astype(float)
```

Removing "Unknown" values from 'type' feature.

In [13]:
```
mal_data = mal_data[mal_data.type != "Unknown"]
```

In [14]:
```
mal_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 15293 entries, 0 to 24261
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   title      15293 non-null  object 
 1   episodes   15293 non-null  float64
 2   type       15293 non-null  object 
 3   favorites  15293 non-null  int64  
 4   score      15293 non-null  float64
 5   members    15293 non-null  int64  
 6   studios    15293 non-null  object 
dtypes: float64(2), int64(2), object(3)
memory usage: 955.8+ KB
```

Now, all of the values in the 'episodes', 'favorites', and 'members' columns have been converted to usable float or integer values. The 'score' feature was a float value from the beginning. Next, dummy variables will be created from the quantitative feature 'type'.

Creating Dummy Variables for "type" - "type_DV".

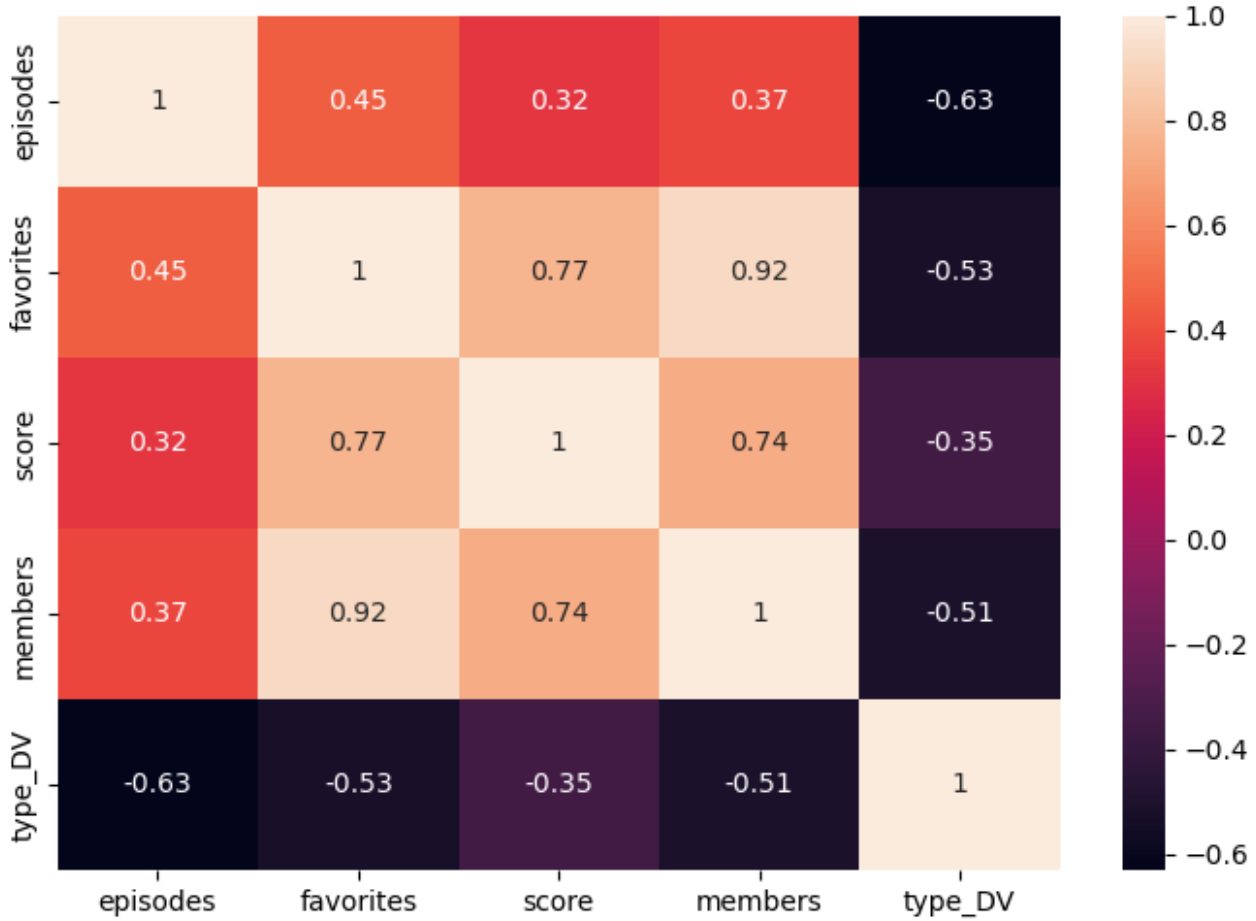
In [15]:
```
type_dict = dict(TV=1, OVA=2, Movie=3, Special=4, ONA=5, Music=6)

mal_data['type_DV'] = mal_data['type'].map(type_dict)
mal_data.tail(3)
```

Out[15]:

	title	episodes	type	favorites	score	members	studios	type_DV
24259	Seikoujo: Haitoku no Biden Dorei	2.0	OVA	10	5.86	4130	Studio Jam	2
24260	Chikan Juunin-tai The Animation	5.0	OVA	7	5.86	2855	Sugar Boy	2
24261	Gibo	2.0	OVA	16	5.86	5535	Y.O.U.C	2

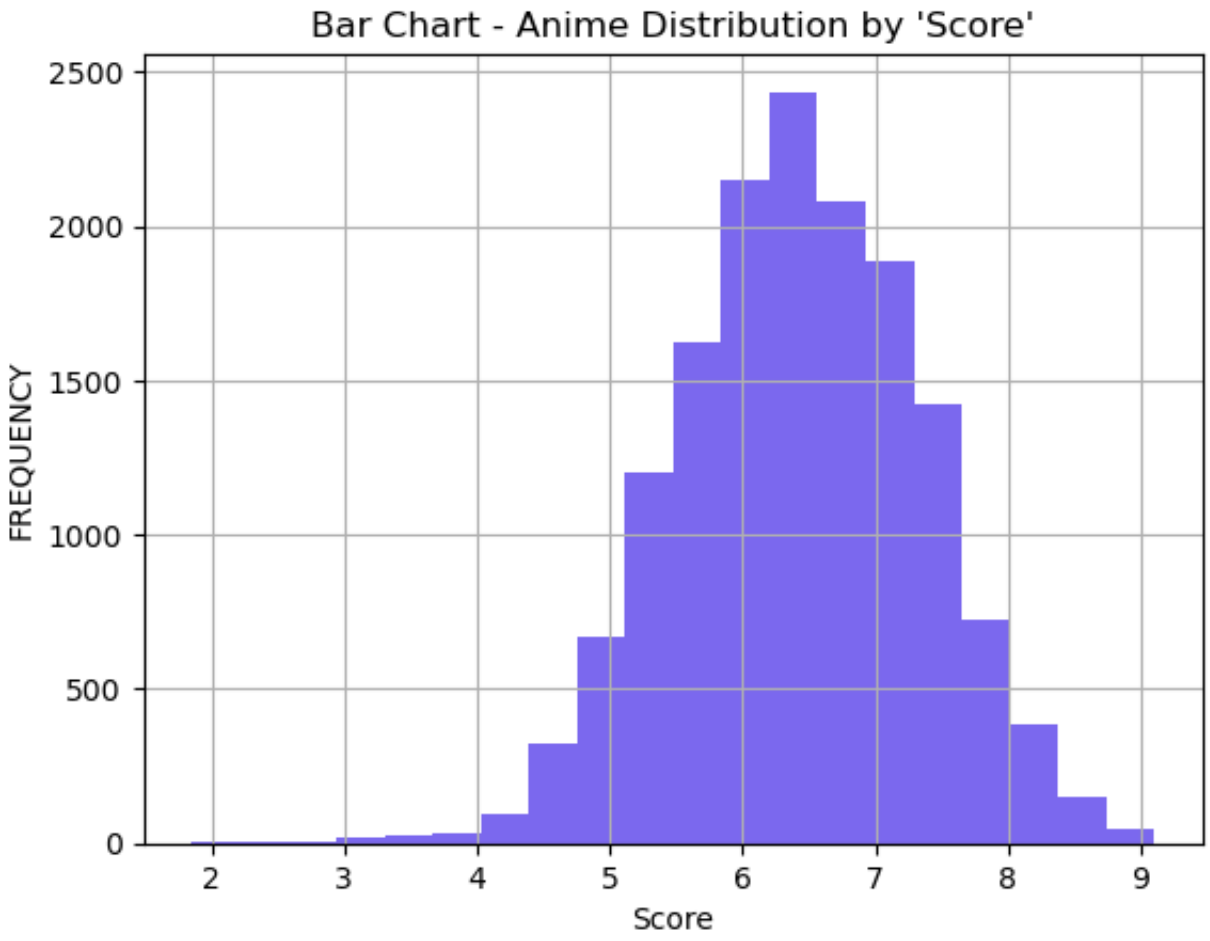
Generating visualizations to examine features and relationships.

In [16]:
```
corr_matrix = mal_data.corr(method = 'spearman')
fig, ax = plt.subplots(1, 1, figsize=(7,5), tight_layout = True)
sns.heatmap(corr_matrix, annot = True)
plt.show()
```



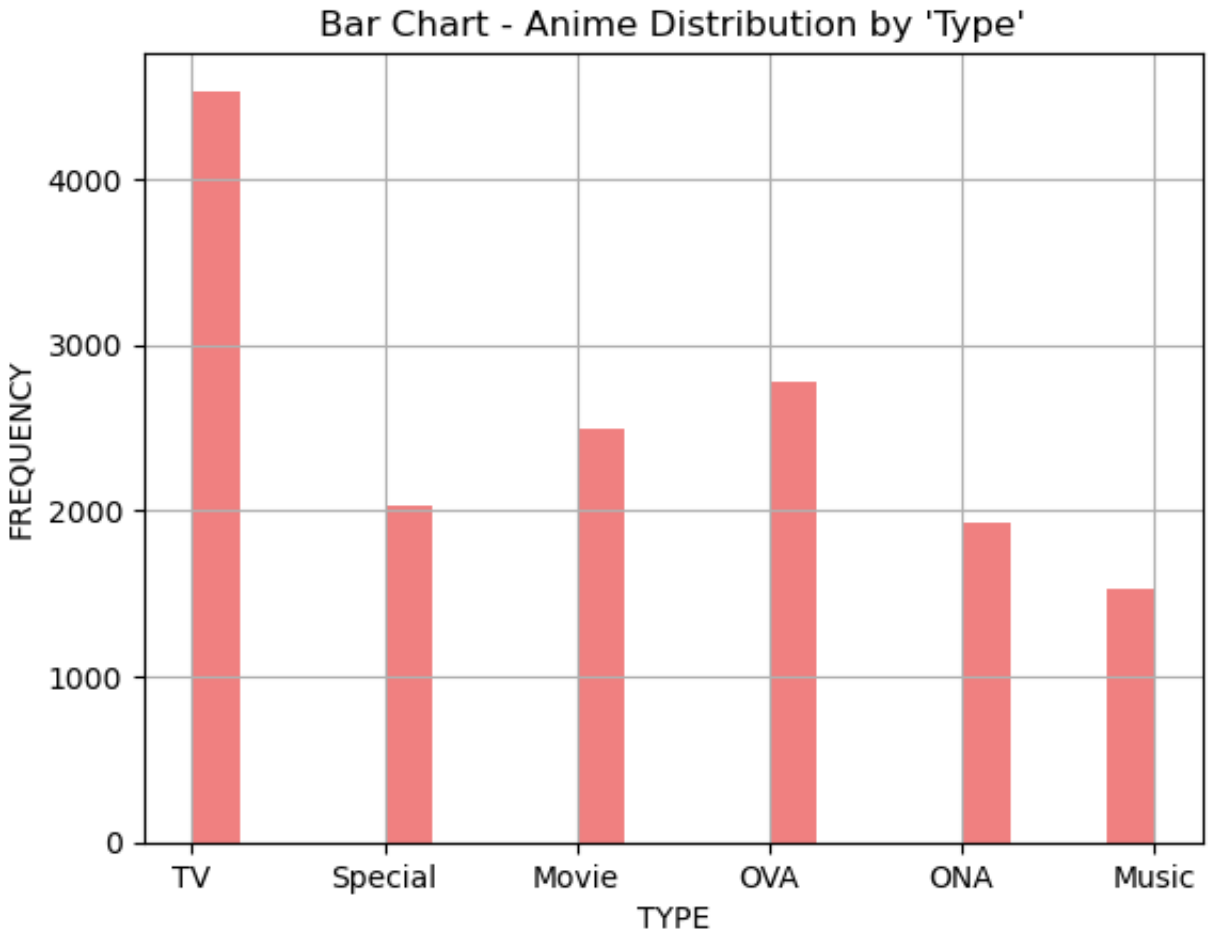
From this correlation matrix, we can see that 'members' and 'favorites' have a pretty clear relationship with the target variable 'score'. This is expected as the more members an anime has, the more favorites it can get. Higher scoring anime may also have more members as well.

In [17]:
```
plt.hist(mal_data['score'], bins=20, color='mediumslateblue')
plt.title("Bar Chart - Anime Distribution by 'Score'")
plt.xlabel("Score")
plt.ylabel("FREQUENCY")
plt.grid(True)
plt.show()
```



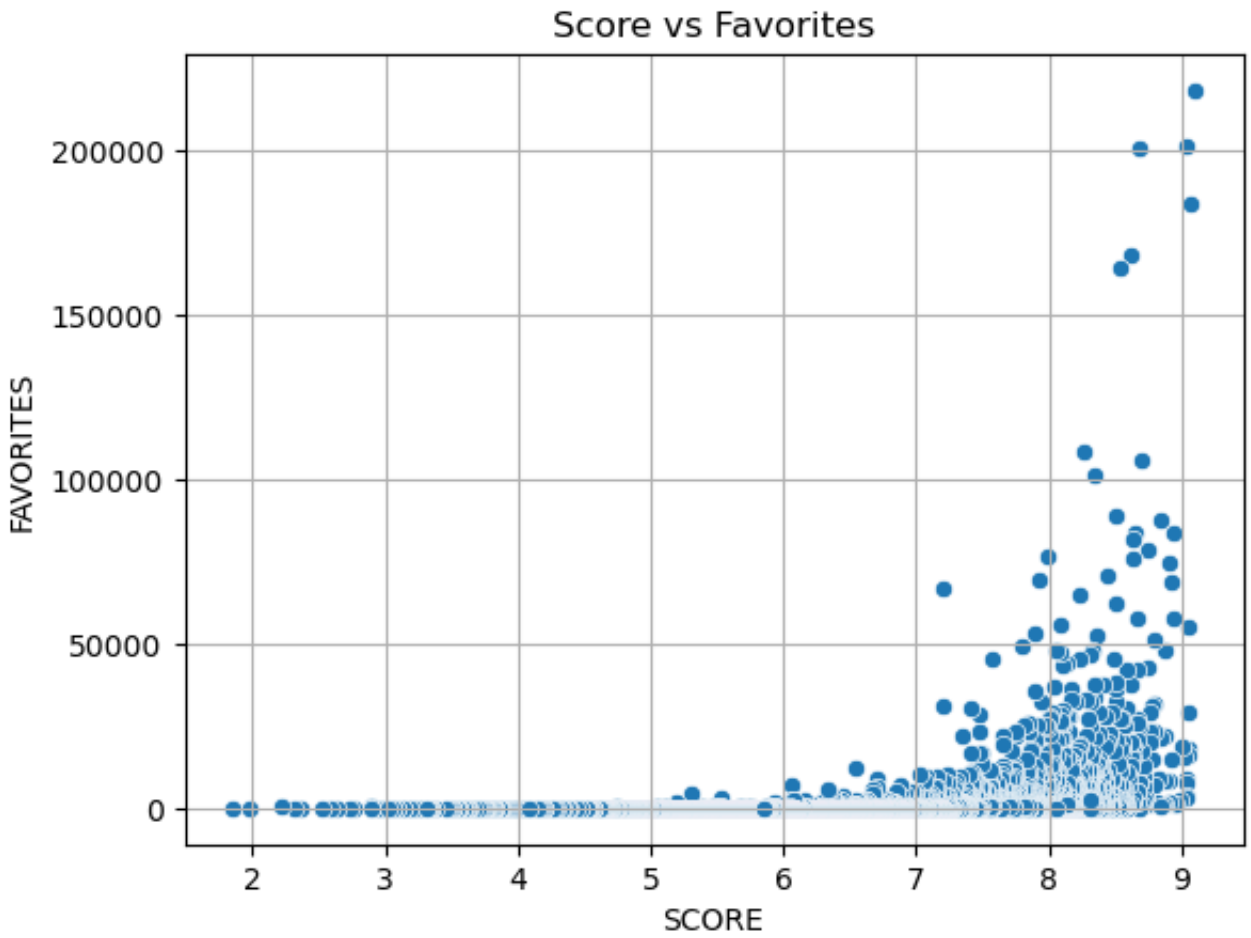
A slight skew to the right can be seen in this distribution with the vast majority of the data concentrated in the upper half of the score scale. So at least most of all anime are at least "decent"...!

In [18]:
```
plt.hist(mal_data['type'], bins=20, color='lightcoral')
plt.title("Bar Chart - Anime Distribution by 'Type'")
plt.xlabel("TYPE")
plt.ylabel("FREQUENCY")
plt.grid(True)
plt.show()
```



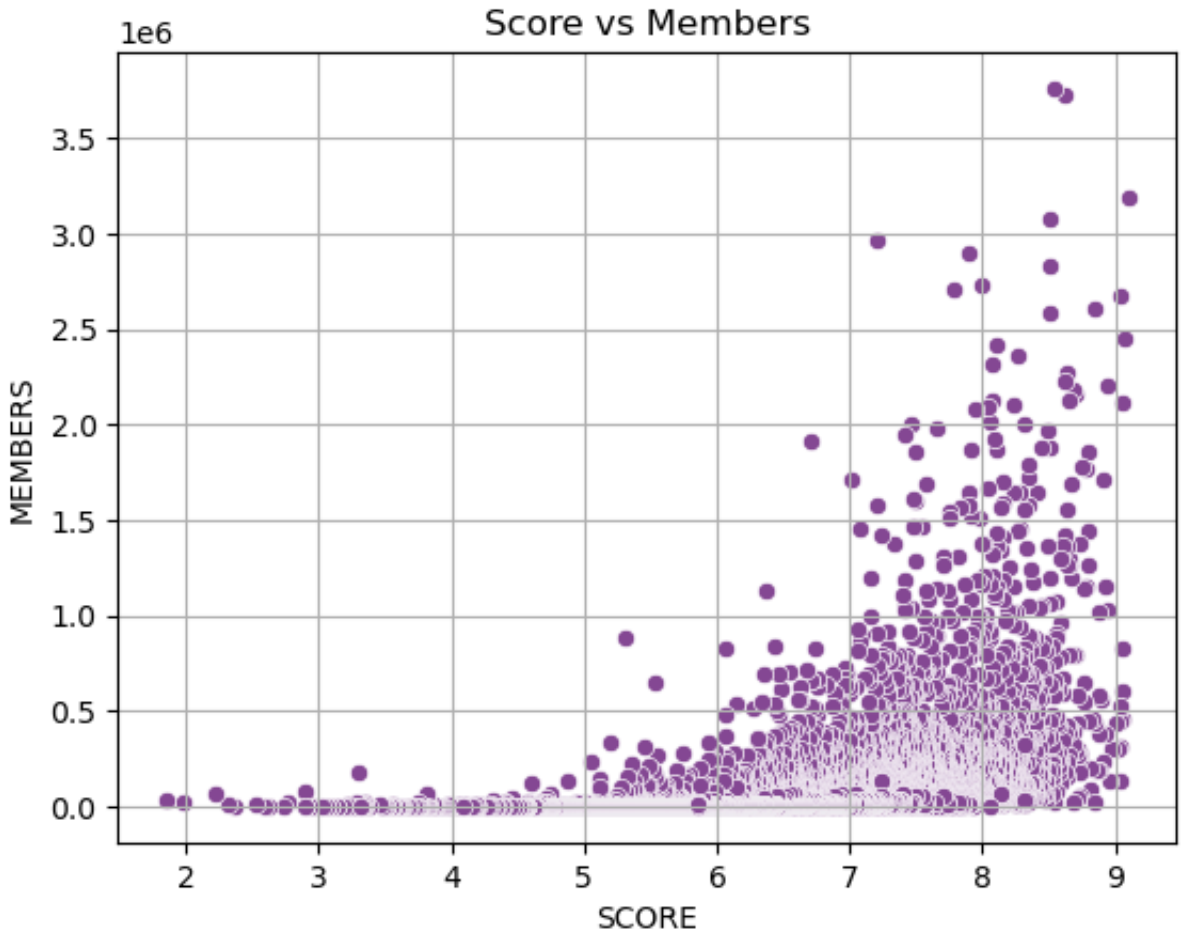
Most anime titles are TV-type, which is expected as most are series

In [19]:
```
plt.figure(7)
plt.title("Score vs Favorites")
plt.xlabel("SCORE")
plt.ylabel("FAVORITES")
sns.scatterplot(mal_data['score'], mal_data['favorites'])
sns.set_palette("PRGn")
plt.grid(True)
```



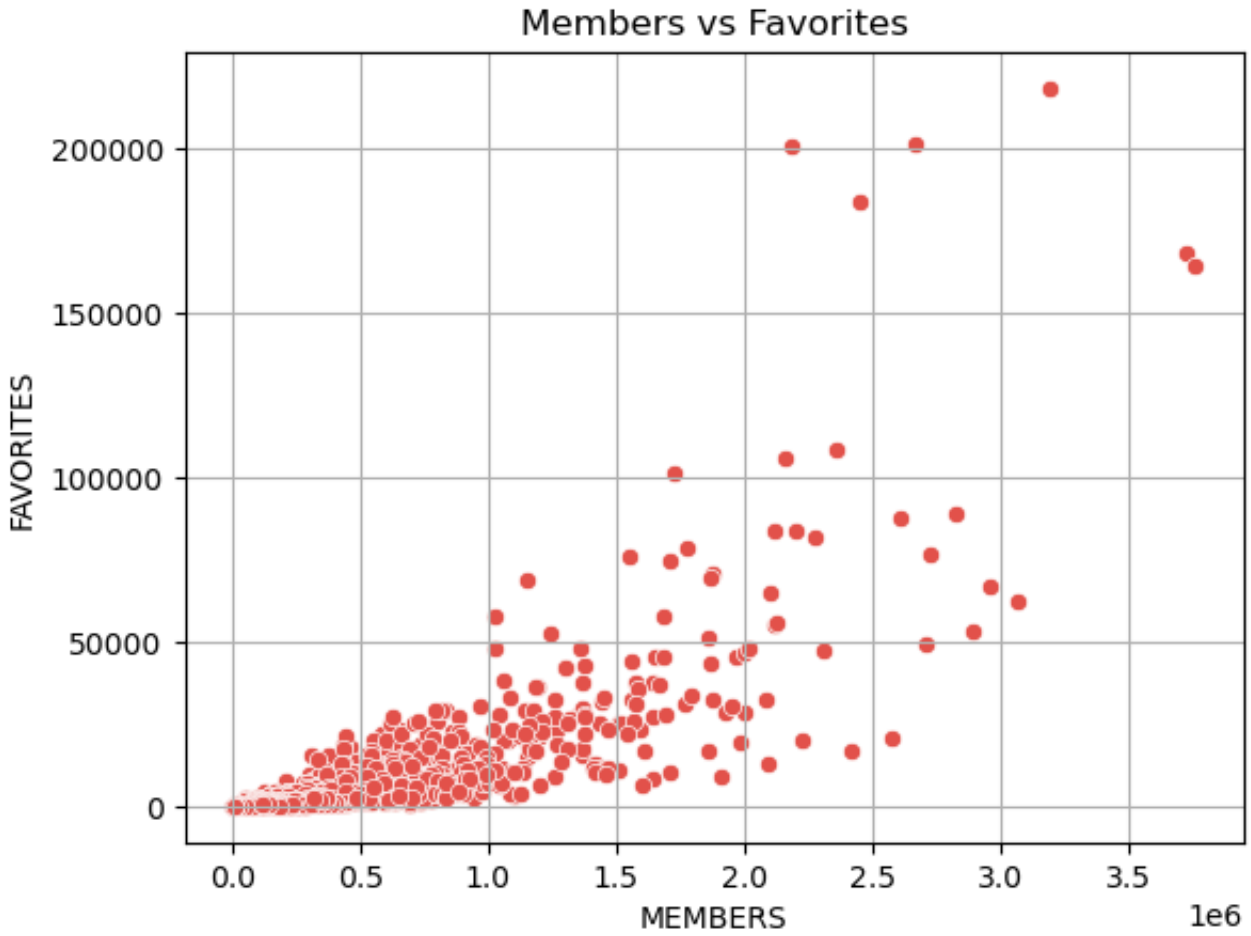
This scatterplot shows that most of the variation and number of favorites lie in the right-half, past the score of around six. It can also be seen that titles with the higher scores do tend to have more favorites. A similar assumption is made regarding 'score' and 'members'.

In [20]:
```
plt.figure(7)
plt.title("Score vs Members")
plt.xlabel("SCORE")
plt.ylabel("MEMBERS")
sns.set_palette("Spectral")
sns.scatterplot(mal_data['score'], mal_data['members'])
plt.grid(True)
```



As expected, higher scores seem to relate with increasing members. To complete the trio of scatterplots, a comparison of 'members' and 'favorites' will be made.

In [21]:
```
plt.figure(7)
plt.title("Members vs Favorites")
plt.xlabel("MEMBERS")
plt.ylabel("FAVORITES")
sns.set_palette("RdGy")
sns.scatterplot(mal_data['members'], mal_data['favorites'])
plt.grid(True)
```



These three scatterplots visually outline the relationships seen above in the correlation matrix. We can see that increasing members increase favorites and scores, so determining ways to get users to become more inclined to become members of titles would directly increase favorites and scores.

Model Building - Linear Regression

Here, linear regression is considered primarily due to its ease in understanding and deployment. The model's flexibility and adaptability are also considered, especially with the inclusion of our categorical variables. This model, however, has its limits such as overfitting or under-fitting due to its sensitivity to outliers

In [22]:
```
mal_model_data = mal_data[['episodes', 'type_DV', 'favorites', 'score', 'mem
mal_model_data.tail(3)
```

Out[22]:

	episodes	type_DV	favorites	score	members
24259	2.0	2	10	5.86	4130
24260	5.0	2	7	5.86	2855
24261	2.0	2	16	5.86	5535

Creating dataframe (mal_model_data) that houses the cleaned quantitative data from "mal_data" to be used for model building.

In [23]:
```
mal_model_data.describe()
```

Out[23]:

	episodes	type_DV	favorites	score	members
count	15293.000000	15293.000000	15293.000000	15293.000000	1.529300e+04
mean	11.598634	2.909697	706.077225	6.417142	6.029173e+04
std	48.792904	1.701085	5563.185671	0.919550	1.978039e+05
min	1.000000	1.000000	0.000000	1.850000	1.600000e+02
25%	1.000000	1.000000	1.000000	5.800000	1.138000e+03
50%	2.000000	3.000000	8.000000	6.420000	5.273000e+03
75%	12.000000	4.000000	66.000000	7.090000	2.898100e+04
max	3057.000000	6.000000	218277.000000	9.100000	3.762148e+06

Splitting the data into a training and test set, where 'score' is the target.

In [24]:
```
import sklearn
from sklearn.model_selection import train_test_split
```

In [25]:
```
target = mal_model_data.score
features = mal_model_data.drop('score',axis=1)

x_train, x_test, y_train, y_test = train_test_split(features, target
                                                    , test_size=0.25


print("shape of - features_train", x_train.shape)
print("shape of - target_train", y_train.shape)
print("shape of - features_test", x_test.shape)
print("shape of - target_test", y_test.shape)
```

```
shape of - features_train (11469, 4)
shape of - target_train (11469,)
shape of - features_test (3824, 4)
shape of - target_test (3824,)
```

Using Linear Regression model to generate 'score' predictions.

In [26]:
```
from sklearn.linear_model import LinearRegression
```

In [27]:
```
linreg_model = LinearRegression()

linreg_model.fit(x_train, y_train)
```

Out[27]:
```
LinearRegression()
```

In [28]:
```
score_predictions = linreg_model.predict(x_test)
score_predictions
```

Out[28]:
```
array([6.39154534, 6.60276127, 6.35411345, ..., 5.91440097, 6.42710106,
       7.08204085])
```

Calculation of RMSE to determine the performance for the regression model.

In [29]:
```
from sklearn import metrics
```

In [30]:
```
RMSE = (metrics.mean_squared_error(y_test, score_predictions, squared=False)
print('Root Mean Squared Error (RMSE): %.4f' % RMSE)
```

```
Root Mean Squared Error (RMSE): 0.8209
```

Conclusion/Recommendations

From the Linear Regression model, we received a pretty low RMSE! This could indicate that the model fits the data well and has more precise predictions. It seems as though we can accurately predict 'score' by using the features of this data. Seeking methods to increase the number of members an anime has seems to be the most influential way to increase score. Fan service is surely the key to popular anime, so increasing fan interaction outside of normal viewing could also positively affect the titles' scores. Considering the implications and assumptions made by the Linear Regression Model, it is likely that the low RMSE could be the result of overfitting; further analysis and experimentation with other models may be needed.

Week 6 - Milestone 3: Preliminary Analysis

Data vs Expectations

Upon my initial exploration of the data, I noticed that there were many more missing values than I anticipated, especially from those features taken from the IMDB website. The data also lacks certain features that I found interesting on a personal, fan-based, level like: production studio and the season released. I was forced to remove the "genre" feature as I was unable to figure out a way to utilize the info how it is currently set up. I am researching ways of maybe tokenizeing the text and separating each genre listed, but I have not had much experience doing so.

Displaying the list of columns remaining in the data.

In [ ]:
```
crunchy_data2.columns.values
```

Displaying a view of the total missing values from each feature.

In [ ]:
```
crunchy_data2.isnull().sum()
```

I see that there are a significant amount of missing values from the 'imdb_score' and 'imdb_votes' columns, this may be an issue if this will call for removing over 150 titles from our data.

Removing missing values from data.

In [ ]:
```
crunchy_data2.dropna(inplace=True)
```

In [ ]:
```
crunchy_data2.isnull().sum()
```

Will I be able to answer the questions I want to answer with the data I have?

I wanted to gain additional insight into the "tmdb_popularity" feature, but as seen below, there are no obvious significant relationship between tmdb_popularity and other variables on this matrix, sans "imdb_votes". "imdb_score" and "imdb_votes" also appear to have a relationship explained by the data.

Below, histograms and a correlation matrix will be used to provide visualizations at a glance about the data. There are no strong relationships between the tmdb_popularity feature and the others that can be seen immediately, other than imdb_votes. Still, I think that popularity can be estimated using the data pending model evaluation.

Checking distributions of features using histograms.

In [ ]:
```
fig = plt.figure(figsize = (15,20))
ax = fig.gca()
crunchy_data2.hist(ax = ax, color='palevioletred')
```

Some thing noticed off first glance of these visualizations:

- Most anime productions are within the last decade.
- The runtime of most productions are under 50 minutes.
- Most productions have under 10 seasons.
- The IMBD and TMDB scores are skewed to the right.

Will use the describe method here for further insight into the data.

In [ ]:
```
crunchy_data2.describe()
```

Some interesing points here are:

- The average anime runtime is 21 minutes.
- The average number of seasons is 1.8, which is surprisingly low to me.
- The average score on IMBD is 6.84 and 7.19 on TMDB; I would have guessed around 7, so that tracks.
- The average 'popularity' is 23.22 on TMDB; not really sure how popularity is calculated, nor its scale, so some exploration into this feature would be useful.

In [ ]:
```
corr_matrix = crunchy_data2.corr(method = 'spearman')
fig, ax = plt.subplots(1, 1, figsize=(7,5), tight_layout = True)
sns.heatmap(corr_matrix, annot = True)
plt.show()
```

What visualizations are especially useful for explaining my data?

In addition to the histograms and heatmap used earlier, this analysis would benefit from additional bar graphs outlining the most successfull productions based on fan-service, like shown below with the top-10 anime productions based on the "tmdb_popularity".

In [ ]:
```
top_10_popular = crunchy_data2.sort_values("tmdb_popularity", ascending=Fals
top_10_popular
```

In [ ]:
```
fig, ax = plt.subplots()
plt.title("Top 10 Anime by Popularity")
sns.barplot('tmdb_popularity', 'title', data = top_10_popular, palette='PuRd
plt.xlabel("Anime's Popularity (tmdb_popularity)")
plt.ylabel("Anime Title (title)")
plt.grid(False)
plt.show()
```

This visualization in particular does not explain our estimation of popularity or model, but does provide useful information for the perspective client wishing to enter into the Anime World Market.

Since data from two sites that share a similar feature for comparison, i.e. "score", is assumed, scatterplots comparing their similarities and differences would be a useful visualizaion. Some clustering analyses may add to the benefits of the visualization.

In [ ]:
```
plt.figure(7)
plt.title("IMBD Score Vs TMBD Score")
sns.scatterplot(crunchy_data2['imdb_score'], crunchy_data2['tmdb_score'])
plt.grid(True)
```

Boxplots, being great for visually outlining features and their outliers, makes it useful to include in this analysis. Below, we can see this one-great outlier that appears to be the "SPY x FAMILY" production, exceeding in popularity by about 4x as the next highest point, with potential to be excluded. I think this may be cause of an error, but "SPY x FAMILY" was exceedingly popular and viral on social media during it's debut, so it may be accurate.

In [ ]:
```
sns.boxplot (y = 'tmdb_popularity', x = 'age_certification', data = crunchy_
```

Do I need to adjust the data and/or driving questions?

Before beginning this assignment, I did not anticipate that the "tmdb_popularity" variable would have such small and few relationships with the other features. I am concerned that I may encounter difficulty getting the best prediction of anime popularity with the data as-is. I will attempt to find data to merge and supplement this data, and add to its predictive capability.

Do I need to adjust my model/evaluation choices?

The Gradient Boosting Algorithm did not yeild promising predictive power with the current data. My partner suggested that I experiment with other models (Logistic regression, Lasso regression, and Random Forest to name a few). Since I have far less modeling experience than her, I am obliged to try others. If my data is to be supplemented or even replaced, this will be even more necessary. I think that my issue currently lies not with my model, but with the data - if it is complete enough to conduct my analysis.

Are my original expectations still reasonable?

I remain confident that popularity can be predicted using data such as this. I think that this data still has a great opportunity to showcase the different characteristics and building blocks of animated productions' success based on fan-service. This data is missing some details that I desired personally for my project, such as producion studios and the season (of year), which I think are pretty important when it comes to describing popular anime. Since the data and model combo did not really live up to expectations, I applied the KNN and Logistic Regression models, with lackluster results. I will try to merge this data with another scraped from a different site, but I anticipate issues with that as well due to the title of the productions having potential discrepancies - maybe additional punctuation or different spellings or translations.

References

Predictive Analytics:

https://www.linkedin.com/advice/1/what-advantages-disadvantages-using-linear-1e#:~:text=Advantage%3A%20Easy%20to%20understand%20and%20interpret&text=You%

Choosing Colormaps in Matplotlib:

https://matplotlib.org/stable/tutorials/colors/colormaps.html

List of named colors in matplotlib:

https://matplotlib.org/stable/gallery/color/named_colors.html

Seaborn Styling, Color:

https://www.codecademy.com/article/seaborn-design-ii