アニメ

SC1015
Mini Project

B128: Group 1

Yon Nan

Aaron

Chen Wei

Table Of Content

What makes an Anime Good ?

Our Motivations

1. Anime is a unique art form that we wish to gain a deeper appreciation for
2. Over the years we have seen many good and mediocre Anime's
3. We love Anime and see it as an interesting topic to research

Datasets

1. Anime2023February by Jerry Kim
    - Contains scraped data on Anime's from MyAnimeList
    - https://www.kaggle.com/datasets/gaeranbab/anime2023february24000

2. Anime Character Traits Dataset by Michael Roberts
    - Contains scraped data on Anime Characters
    - https://www.kaggle.com/datasets/mjrone/anime-character-traits-dataset

Data Cleaning

Columns we dropped

```
[86]: ani_s[['Unnamed: 0', 'Link', "Openings", 'Opening Artists', 'Endings', 'Ending Artists']]
```

[86...

	Unnamed: 0	Link	Openings	Opening Artists	Endings	Ending Artists
0	0	https://myanimelist.net/anime/5114/Fullmetal_A...	['again', 'Hologram (ホログラム)', 'Golden Time Lov...	['YUI', 'NICO Touches the Walls', 'Sukima Swit...	['Uso (嘘)', 'LET IT OUT', 'Tsunaida Te (つないだ手)...	['SID', 'Miho Fukuhara', "Lil'B", 'SCANDAL', '...
1	1	https://myanimelist.net/anime/41467/Bleach__Se...	['スカー']	['キタニタツヤ']	['Rapport', '最果て', 'Number One']	['キタニタツヤ', 'SennaRin', 'Shiro Sagisu, Hazel Fe...
2	2	https://myanimelist.net/anime/9253/Steins_Gate	['Hacking to the Gate']	['Kanako Itou']	['Toki Tsukasadoru Juuni no Meiyaku (刻司ル十二ノ盟約)...	['Yui Sakakibara', 'Takeshi Abo', 'Kanako Itou...
3	3	https://myanimelist.net/anime/28977/Gintama°	['DAY×DAY', 'Pride Kakumei (プライド革命)', 'Saigo m...	['BLUE ENCOUNT', 'CHiCO with HoneyWorks', 'Aqu...	['DESTINY', 'Saigo made II (最後まで II)', 'Pride Ka...	['Negoto', 'Aqua Timez', 'CHiCO with HoneyWork...
4	4	https://myanimelist.net/anime/43608/Kaguya-sam...	['GIRI GIRI feat Suu (GIRI GIRI feat. すぅ)']	['Masayuki Suzuki']	['GIRI GIRI feat Suu (GIRI GIRI feat. すぅ)', 'H...	['Masayuki Suzuki', 'Airi Suzuki', 'Miyuki Shi...
...	...	...	...	...	...	...
23995	23995	https://myanimelist.net/anime/8574/Aoi_Taiken	[]	[]	[]	[]
23996	23996	https://myanimelist.net/anime/17823/Ikenai_Boy	[]	[]	[]	[]
23997	23997	https://myanimelist.net/anime/2240/Sei_Michael...	[]	[]	[]	[]
23998	23998	https://myanimelist.net/anime/4178/Dream_Hazar...	[]	[]	[]	[]
23999	23999	https://myanimelist.net/anime/5160/Wake_Up_Ari...	[]	[]	[]	[]

- The opening and closing songs for an Anime are all different, even possibly changing between episodes.

```
[6]: ani_s.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 24000 entries, 0 to 23999
Data columns (total 31 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Unnamed: 0        24000 non-null  int64
 1   Rank              22122 non-null  float64
 2   Title             24000 non-null  object
 3   Link              24000 non-null  object
 4   Score             15215 non-null  float64
 5   Type              23976 non-null  object
 6   # Episodes        23976 non-null  object
 7   Source            23976 non-null  object
 8   Status            23976 non-null  object
 9   Premiered         7477 non-null   object
 10  Aired Date        23976 non-null  object
 11  Studios           23976 non-null  object
 12  Genres            19326 non-null  object
 13  Themes            13168 non-null  object
 14  Demographic       9287 non-null   object
 15  Duration          23976 non-null  object
 16  Age Rating        23976 non-null  object
 17  Number of Raters  15195 non-null  float64
 18  Popularity        23976 non-null  object
 19  Members           23976 non-null  object
 20  Favorites         23976 non-null  object
 21  Adaptation        5846 non-null   object
 22  Sequel            4106 non-null   object
 23  Prequel           4090 non-null   object
 24  Characters        24000 non-null  object
 25  Role              24000 non-null  object
 26  Voice Actors      24000 non-null  object
 27  Openings          23976 non-null  object
 28  Opening Artists   23976 non-null  object
 29  Endings           23976 non-null  object
 30  Ending Artists    23976 non-null  object
dtypes: float64(3), int64(1), object(27)
memory usage: 5.7+ MB
```

Anime2023February Dataset:

1. Most have the data types incorrectly loaded into Dataframe

```
print(ani_s["Characters"][0])
print(type(ani_s["Characters"][0]))
```

```
['Elric, Edward', 'Elric, Alphonse', 'Mustang, Roy',
<class 'str'>
```

Data Cleaning

```
# set 1 of columns to make into a list
# NOTE: Can only run once due to the change it types
col_to_clean = ["Characters", "Role", 'Voice Actors']

for col in col_to_clean:
    # a temp list to hold list of each row
    tmp = []
    for i, l in enumerate(clean_ani_s[col]):
        b = l.replace("',", '@')
        b = b.replace(",", '')
        b = b.replace("@", "',")
        b = b.split(",")
        for j, name in enumerate(b):
            c = regexAN.sub('', name).lower()
            if (c == ""):
                continue
            else:
                # to remove the space infront of the name
                while(c[0] == " "):
                    c = c[1:]
                b[j] = c

        tmp.append(b)

    # overide the Tags column to be a list
    clean_ani_s[col] = tmp

# removing rows that do not have a character list
for i, l in enumerate(clean_ani_s["Characters"]):
    if (l == ['']):
        clean_ani_s = clean_ani_s.drop([i])

# reset the index
clean_ani_s.index = range(len(clean_ani_s))
```

- Converting the String of a List into a List

```
# cleaning the studios column
tmp = []

for i, string in enumerate(clean_ani_s["Studios"]):
    if (string == "None found, add some" or string == ""):
        tmp.append([])
    else:
        lis = string.split(",")
        for j, std in enumerate(lis):
            # to remove the spaces at the start
            while(std[0] == " "):
                std = std[1:]
            lis[j] = std
        tmp.append(lis)
clean_ani_s["Studios"] = tmp
```

Data Cleaning

```
# turning selected cols into a list
# NOTE: Can only run once due to the change it types
col_to_clean = ["Tags", "Anime"]

for col in col_to_clean:
    # a temp list to hold list of list values
    tmp = []

    for i, l in enumerate(clean_ani_t[col]):
        if (type(l) != str and l.isnan()):
            tmp.append(np.nan)
            continue
        b = l.split(",") # b is a list of the split items of the str of col item
        for i in range(len(b)):
            b[i] = regexAN.sub('', b[i])
        tmp.append(b)

    # overide the Tags column to be a list
    clean_ani_t[col] = tmp
```

```
# remove the sapce infront of the tags items
for i,l in enumerate(clean_ani_t["Tags"]):
    for j,m in enumerate(l):
        if (m[0] == " "):
            clean_ani_t["Tags"][i][j] = clean_ani_t["Tags"][i][j][1:]
```

Anime Character Traits Dataset:

- Converting the String of Lists back into a List

```
ani_t.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 119825 entries, 0 to 119824
Data columns (total 7 columns):
 #   Column      Non-Null Count   Dtype
---  ------      --------------   -----
 0   Id          119825 non-null  int64
 1   Names       119824 non-null  object
 2   Hair_Color  96059 non-null   object
 3   Gender      113212 non-null  object
 4   Tags        119391 non-null  object
 5   Anime       104426 non-null  object
 6   Manga       35540 non-null   object
dtypes: int64(1), object(6)
memory usage: 6.4+ MB
```

Data Cleaning

```
tags = dict()

for i,l in enumerate(clean_ani_t["Tags"]):
    for var in set(l):
        tags[var] = tags.get(var, 0) + 1

tags = pd.DataFrame(data = tags, index = ["Count"])
tags = tags.transpose()
tags = tags.reset_index()
tags = tags.rename({"index": "Tag"}, axis=1)
tags = tags.sort_values(by=['Count'], ascending=False)
tags = tags.loc[tags["Count"] > 10]
# reset the index
tags.index = range(len(tags))
tags
```

- Manually remove “Tags” that did not make sense or referred to the Anime Title

```
# manual cleaning of tags
# only run once
drop = [505, 518, 525, 527, 528, 533, 536, 541, 543, 544, 545, 547,
tags = tags.drop(drop)
print(tags.to_string())
```

```
90                Knight    835
91                  Bald    831
92                Artist    827
93        Monster Tamer     824
94        Rapunzel Hair     780
95          Mischievous     779
96        Face Markings     767
97               Doctor     760
```

Data Cleaning

```
# make all names lower case and first name first
tmp = []
for i, name in enumerate(clean_ani_t["Names"]):
    name = name.split(" ")
    if (len(name) <= 2):
        name.append(name[0])
        name.remove(name[0])
    else:
        up = name[0].isupper()
        for j in range(len(name)):
            if (name[0].isupper() == up):
                name.append(name[0])
                name.remove(name[0])
            else:
                break
    name = " ".join(name)
    tmp.append(name.lower())
clean_ani_t["Names"] = tmp
```

- Change the “Name” to be in the same format as the Anime2023February Dataset Character names

Exploratory Analysis

Possible Response Variables

	Rank	Score	Popularity
0	1.0	9.11	#3
1	2.0	9.10	#525
2	3.0	9.08	#13
3	4.0	9.07	#336
4	5.0	9.07	#226
...	...	...	...
23995	NaN	4.61	#12360
23996	NaN	4.61	#12682
23997	NaN	4.59	#11746
23998	NaN	4.59	#11343
23999	NaN	4.54	#11933

- **Rank**: The Anime's ranking based on the score
- **Score**: The rating from 0 to 10 aggregated by all scores given from MyAnimeList users
- **Popularity**: Ranked based on the number of users that have indicated they watched the Anime (lower the better)

Themes

- **Theme**: The main focus of the Anime

```
# spliting the dataset by themes
df_dict_themes = dict()

# for each specific theme get all anime with that genre
for i, thm in enumerate(themes["Themes"]):
    tmp = clean_ani_s.copy()
    drop_ = [x for x in range(len(tmp))]
    for j, thm_list in enumerate(clean_ani_s["Themes"]):
        if thm in thm_list:
            drop_.remove(j)
    # drop all rows without the genre
    tmp = tmp.drop(index = drop_)
    # reset the index
    tmp.index = range(len(tmp))
    # add the dataframe into a dictionary
    df_dict_themes[thm] = tmp
```

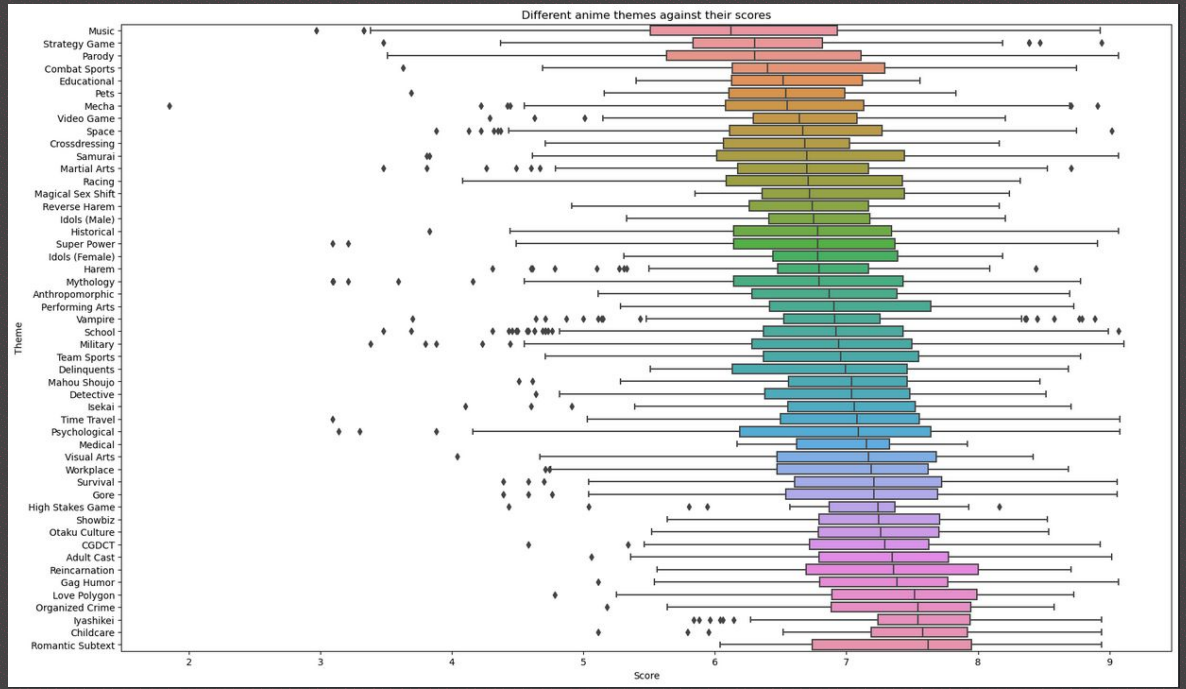
```
# Assign each of the respective dataframe with a tag of theme category
for i, thm in enumerate(themes["Themes"]):
    df_dict_themes[thm] = df_dict_themes[thm].assign(Theme=thm)
```

	Themes	Count
0	School	1714
1	Music	1178
2	Mecha	1153
3	Historical	1105
4	Parody	653
5	Anthropomorphic	637
6	Military	596
7	Super Power	596
8	Mythology	570
9	Space	542
10	Martial Arts	477

```
# concate all the df
for i, thm in enumerate(themes["Themes"]):
    if (i == 0):
        thm_concat = df_dict_themes[thm]
    else:
        thm_concat = pd.concat([thm_concat,df_dict_themes[thm]])
```

Themes

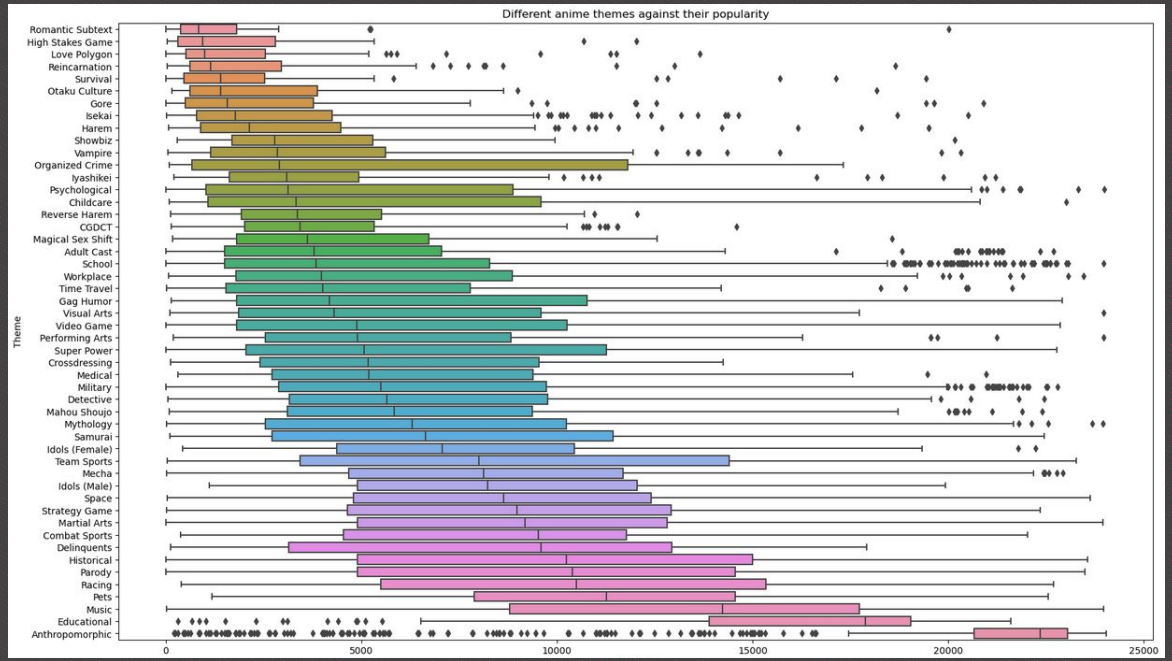
- **Theme**: The main focus of the Anime



Theme vs Score

Themes

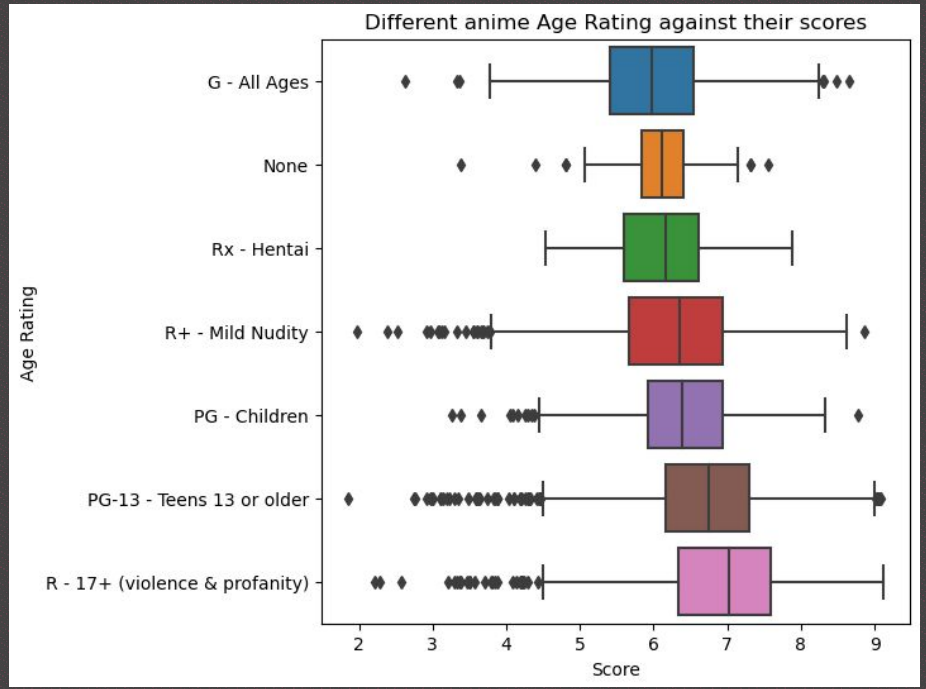
Theme vs Popularity



- Greater variation between categories
- Possible correlation which is stronger than with “Score”

Age Rating

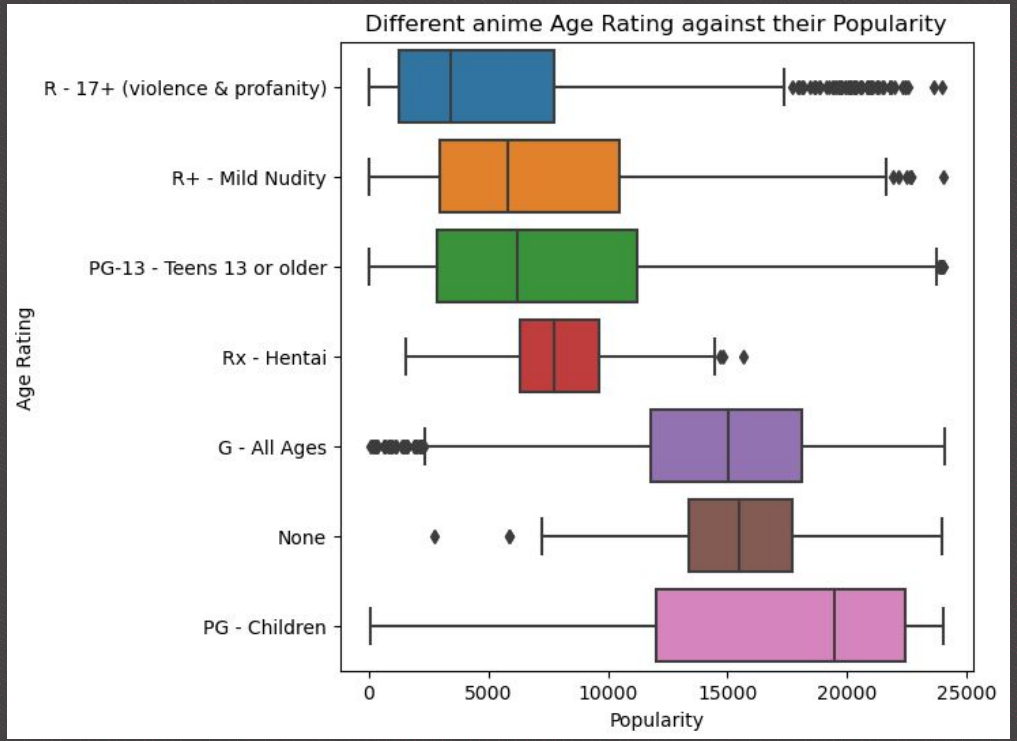
- **Age Rating**: The Anime's suitability for audience based on content



Age Rating vs Score

Age Rating

Age Rating vs Popularity



- Greater variation between categories
- Q3 of first 4 categories lower than Q1 of the rest
- Possible correlation which is stronger than with “Score”

Studios

- **Studio**: The company that made the Anime

```
#Only include the studio that have produced at least the average number of anime across all studios
studio_Total_Pop = dict()

for x, lis in enumerate(tmp_studio_eda["Studios"]):
    if (lis == []):
        continue
    for std in lis:
        if std not in studio_Total_Pop:
            studio_Total_Pop[std] = [x]
        else:
            studio_Total_Pop[std].append(x)


tmp = []
for std, pop_indices in studio_Total_Pop.items():
    pop_indices = sorted(pop_indices, key=lambda i: clean_ani_s["Popularity"][i])
    tmp_ = [clean_ani_s["Popularity"][i] for i in pop_indices if ~np.isnan(clean_ani_s["Popularity"][i])]
    avg_pop = sum(tmp_) / len(tmp_)
    top_pop = tmp_[0]

    score_indices = sorted(pop_indices, key=lambda i: clean_ani_s["Score"][i], reverse=True)
    tmp_ = [clean_ani_s["Score"][i] for i in score_indices if ~np.isnan(clean_ani_s["Score"][i])]
    if (len(tmp_) > 0):
        avg_score = sum(tmp_) / len(tmp_)
        top_score = tmp_[0]
    else:
        avg_score = np.nan
        top_score = np.nan

    anime_titles = [clean_ani_s["Title"][i] for i in pop_indices]
    if ((len(anime_titles) > 0) and (avg_pop is not None) and (avg_score is not None)):
        if std not in [tmp_["Studios"] for tmp_ in tmp]:
            tmp.append({"Studios": std, "Animes' Average Popularity": avg_pop,
                        "Animes' Top Popularity": top_pop, "Animes' Average Score": avg_score,
                        "Animes' Top Score": top_score, "Anime Titles": anime_titles,
                        "# of Animes'": len(pop_indices)})
df11 = pd.DataFrame(tmp)
```

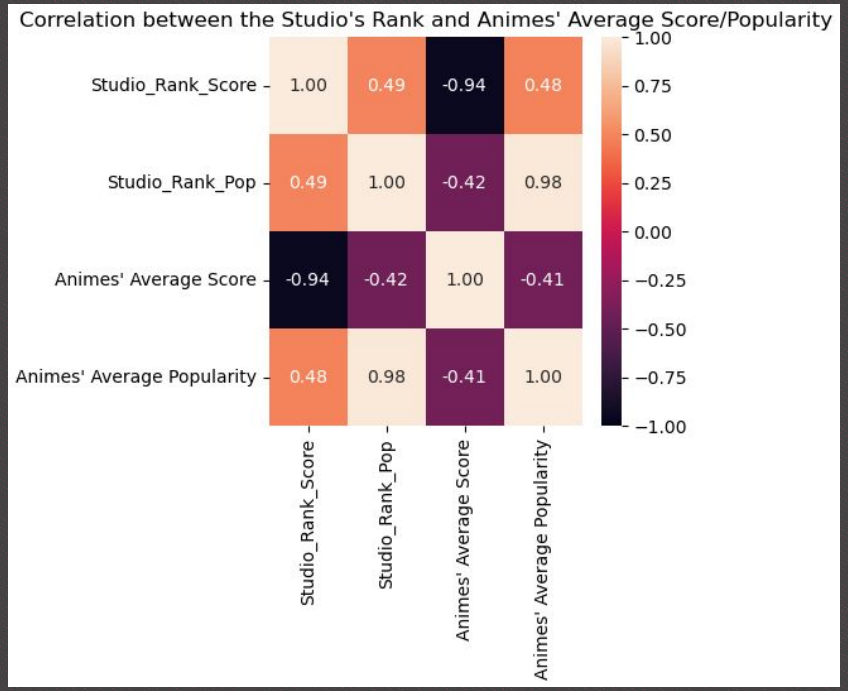
	Studios	Count	Anime Titles
0	Toei Animation	672	[Slam Dunk, Mononoke, World Trigger 3rd Season...
1	Sunrise	473	[Gintama', Gintama': Enchousen, Gintama, Ginta...
2	J.C.Staff	368	[Bakuman. 3rd Season, Kono Subarashii Sekai ni...
3	Madhouse	349	[Hunter x Hunter (2011), Monster, Hajime no Ip...
4	Production I.G	311	[Haikyuu!! Karasuno Koukou vs. Shiratorizawa G...
...	...	...	...
841	Digital Network Animation	1	[Kimi wa Kanata]
842	Ripromo	1	[Tokyo Futago Athletic]
843	Studio UGOKI	1	[Robber's Company]
844	I-move	1	[Pia Carrot e Youkoso!!: Sayaka no Koi Monogat...
845	Studio Jin	1	[Ryoku Tama Shinshi: 10 Short Stories]

Studios

Studios	Animes' Average Popularity	Animes' Average Score	Studio_Rank_Score	Studio_Rank_Pop
K-Factory	3171.666667	8.396667	1.0	61.0
Studio Bind	660.000000	8.340000	2.0	2.0
Egg Firm	650.250000	8.292500	3.0	1.0
AHA Entertainment	2094.000000	7.910000	4.0	11.0
Samsara Animation Studio	2900.000000	7.850000	5.0	49.0
Studio Massket	6958.000000	7.800000	6.0	282.0
Djinn Power	4832.000000	7.780000	7.0	157.0
Code	5021.000000	7.660000	8.0	168.0
Shenman Entertainment	8276.000000	7.650000	9.0	369.0
Animation Do	2311.500000	7.635000	10.0	14.0

- Ranked the studios by the Average Score/ Popularity

Studios



- Strong correlation of 0.94 between Studios' Rank and Animes' Average Score
- Strong correlation of 0.98 between Studios' Rank and Animes' Average Popularity
- Studio rank should be used as a predictor

Voice Actors

- **Voice Actor**: The voice actor that voiced in the anime

```
voice_df = clean_ani_s["Voice Actors"]
Actor_Total_Pop = {}

# associated the anime index in the dataframe to the actor
for x, title in enumerate(clean_ani_s["Title"]):
    for actor in clean_ani_s["Voice Actors"][x]:
        if (actor == "[]"):
            continue
        elif actor not in Actor_Total_Pop:
            Actor_Total_Pop[actor] = [x]
        else:
            Actor_Total_Pop[actor].append(x)


tmp = []
index = 0
for actor, pop_indices in Actor_Total_Pop.items():
    pop_indices = sorted(pop_indices, key=lambda i: clean_ani_s["Popularity"][i])
    tmp_ = [clean_ani_s["Popularity"][i] for i in pop_indices if ~np.isnan(clean_ani_s["Popularity"][i])]
    avg_pop = sum(tmp_) / len(tmp_)
    top_pop = tmp_[0]

    score_indices = sorted(pop_indices, key=lambda i: clean_ani_s["Score"][i], reverse=True)
    tmp_ = [clean_ani_s["Score"][i] for i in score_indices if ~np.isnan(clean_ani_s["Score"][i])]
    if (len(tmp_) > 0):
        avg_score = sum(tmp_) / len(tmp_)
        top_score = tmp_[0]
    else:
        avg_score = np.nan
        top_score = np.nan

    anime_titles = [clean_ani_s["Title"][i] for i in pop_indices]
    if len(anime_titles) > 0 and avg_pop is not None and avg_score is not None:
        if actor not in [d["Voice Actor"] for d in tmp]:
            tmp.append({"Index":index,"Voice Actor": actor, "Animes' Average Popularity": avg_pop,
                        "Animes' Top Popularity": top_pop, "Animes' Average Score": avg_score,
                        "Animes' Top Score": top_score, "Anime Titles": anime_titles,
                        "# of Animes'": len(pop_indices)})
    index += 1
df11 = pd.DataFrame(tmp)
```

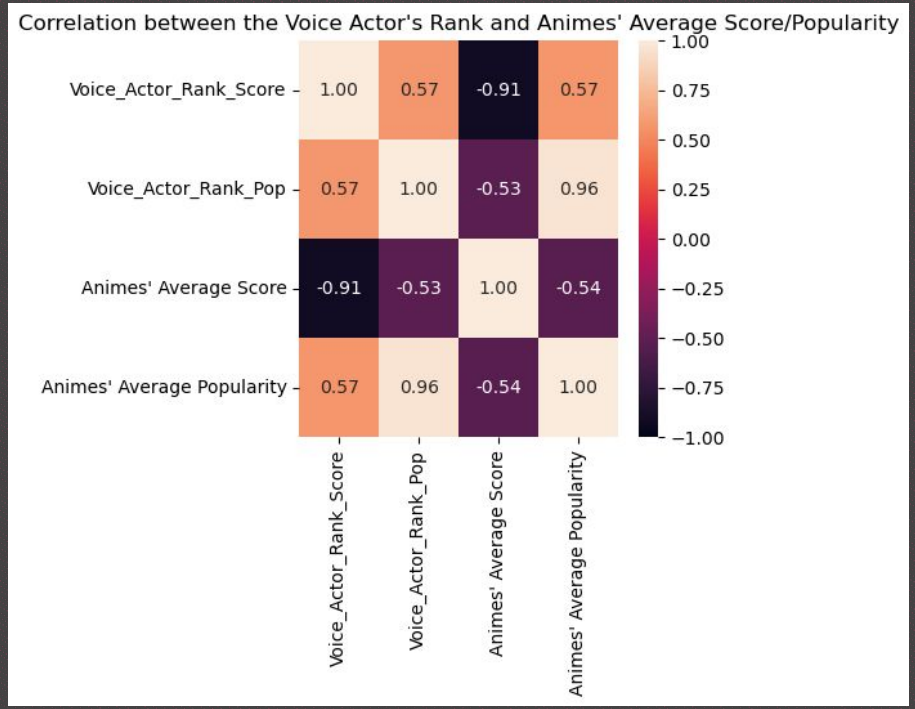
	Voice Actor	Count	Anime Titles
1	sakurai takahiro	412	[Fruits Basket: The Final, 3-gatsu no Lion 2nd...
2	hanazawa kana	385	[Steins;Gate, 3-gatsu no Lion 2nd Season, Owar...
3	sawashiro miyuki	345	[Hunter x Hunter (2011), Owarimonogatari 2nd S...
4	koyasu takehito	342	[Gintama°, Kaguya-sama wa Kokurasetai: Ultra R...
5	kugimiya rie	338	[Fullmetal Alchemist: Brotherhood, Gintama°, G...
...	...	...	...
4717	maeda masaaki	1	[Kidou Senshi Gundam F91]
4718	sanada hiroyuki	1	[Kamui no Ken]
4719	tamomura shin	1	[Gilgamesh]
4720	wada kei	1	[Esper Mami]
4721	kurukuru paama	1	[Souryo to Majiwaru Shikiyoku no Yoru ni...]

Voice Actors

Voice Actor	Animes' Average Popularity	Animes' Average Score	Voice_Actor_Rank_Score	Voice_Actor_Rank_Pop
kamata erena	19.0	8.94	1.0	61.0
kobashi megumi	864.0	8.73	2.0	2.0
miyamura yoshito	2957.0	8.71	3.0	1.0
tanaka emi	2957.0	8.71	4.0	11.0
daruma jiro	2957.0	8.71	5.0	49.0
tsuda ayaka	761.0	8.70	6.0	282.0
meteor	561.0	8.70	7.0	157.0
asei	561.0	8.70	8.0	168.0
oka akiko	208.0	8.66	9.0	369.0
harada daijiro	8146.5	8.66	10.0	14.0

- Add Average Score/Popularity of animes they voice acted in
- Sort and Ranked based on the Animes' Average Score and Popularity

Voice Actors



- Strong correlation of 0.91 between voice actors' rank and animes' average score
- Strong correlation of 0.97 between voice actors' rank and animes' average popularity
- Voice actor rank should be used as a predictor

Character Ranks

- **Character Rank**: The popularity ranking of the Character

```
tmp_list = []

for i, list_ in enumerate(clean_ani_s["Characters"]):
    # this holds the ranks of chars per anime
    tmp = []
    for j, name in enumerate(list_):
        tmp_ = clean_ani_t.loc[clean_ani_t["Names"] == name]
        tmp_.index = range(len(tmp_))
        found = False
        if (tmp_.shape[0] == 1):
            tmp.append(int(tmp_["Id"]))
            found = True
        elif (tmp_.shape[0] > 1):
            for k, ani in enumerate(tmp_["Anime"]):
                if (clean_ani_s["Adaptation"][i] in ani):
                    tmp.append(int(tmp_["Id"][k]))
                    found = True
                    break
        else:
            tmp.append(-1)

    if (~found):
        tmp.append(-1)

    tmp_list.append(tmp)
clean_ani_s["Char Ranks"] = tmp_list
```

```
ani_t.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 119825 entries, 0 to 119824
Data columns (total 7 columns):
 #   Column      Non-Null Count   Dtype
---  ------      --------------   -----
 0   Id          119825 non-null  int64
 1   Names       119824 non-null  object
 2   Hair_Color  96059 non-null   object
 3   Gender      113212 non-null  object
 4   Tags        119391 non-null  object
 5   Anime       104426 non-null  object
 6   Manga       35540 non-null   object
dtypes: int64(1), object(6)
memory usage: 6.4+ MB
```

Anime Character Traits Dataset:

1. The Id gave the rank for each Character
2. Could be matched to characters in the Anime2023February Dataset we have been using thus far.

Character Ranks

```
INT_MAX = 99999
tmp_avg = []
tmp_main_avg = []
tmp_main_top = []
tmp_sup_avg = []
tmp_sup_top = []

for i, list_ in enumerate(clean_ani_s["Char Ranks"]):
    role = clean_ani_s["Role"][i]
    rk_m = []
    rk_s = []
    for j, rank in enumerate(list_):
        if (rank == -1):
            continue
        else:
            try:
                if (role[j] == "main"):
                    rk_m.append(rank)
                elif (role[j] == "supporting"):
                    rk_s.append(rank)
            except:
                continue

    # get the avg and top of the main
    if (len(rk_m) > 0):
        tmp_main_avg.append((sum(rk_m) / len(rk_m)))
        tmp_main_top.append(max(rk_m))
    else:
        tmp_main_avg.append(np.nan)
        tmp_main_top.append(np.nan)

    # get the avg and top of the supporting
    if (len(rk_s) > 0):
        tmp_sup_avg.append((sum(rk_s) / len(rk_s)))
        tmp_sup_top.append(max(rk_s))
    else:
        tmp_sup_avg.append(np.nan)
        tmp_sup_top.append(np.nan)
```

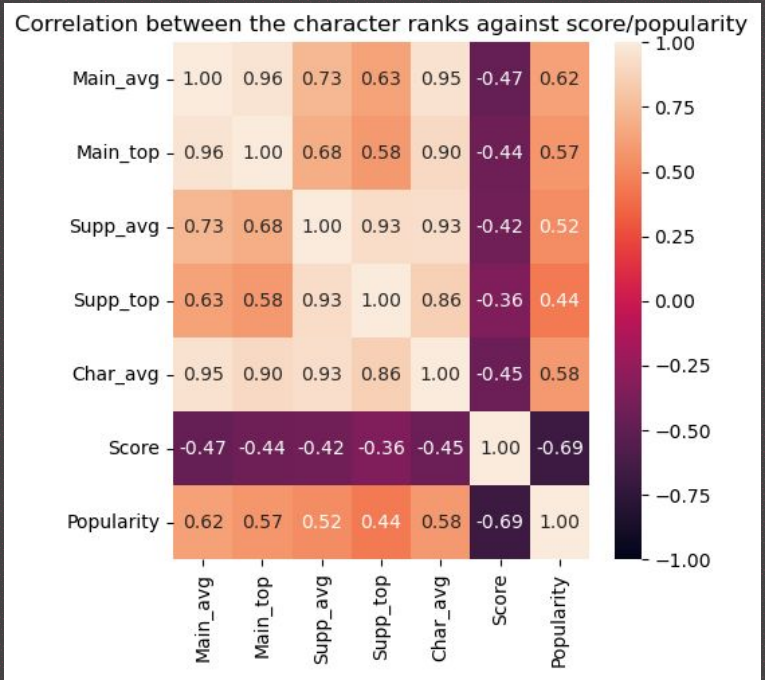
```
# get the avg and top of the main
if (len(rk_m) > 0):
    tmp_main_avg.append((sum(rk_m) / len(rk_m)))
    tmp_main_top.append(max(rk_m))
else:
    tmp_main_avg.append(np.nan)
    tmp_main_top.append(np.nan)

# get the avg and top of the supporting
if (len(rk_s) > 0):
    tmp_sup_avg.append((sum(rk_s) / len(rk_s)))
    tmp_sup_top.append(max(rk_s))
else:
    tmp_sup_avg.append(np.nan)
    tmp_sup_top.append(np.nan)

# get the avg of the main and supporting
tmp = rk_m + rk_s
if (len(tmp) > 0):
    tmp_avg.append(sum(tmp) / len(tmp))
else:
    tmp_avg.append(np.nan)
```

1. Main_avg: The average rank of all the main characters in the Anime
2. Main_top: The highest rank for a main character in the Anime
3. Supp_avg: The average rank of all the supporting characters in the Anime
4. Supp_top: The highest rank for a supporting character in the Anime
5. Char_avg: The average rank for all characters in the Anime

Character Ranks



- The Top 2 categories with highest correlation with Score/Popularity were “Main_avg” and “Char_avg”
- Correlation was about 0.6 (Moderate Strength)
- Both being with respect to Popularity, similar to all other predictor variables thus far.

Machine Learning

ML for What makes an anime good?

Response : Anime's Popularity

Predictors :

1. Themes
2. Age Rating
3. Anime Studio Rank
4. Anime Voice Actors' Rank
5. Main Character Average Rank
6. The Average Character Rank

Models considered

Models considered:

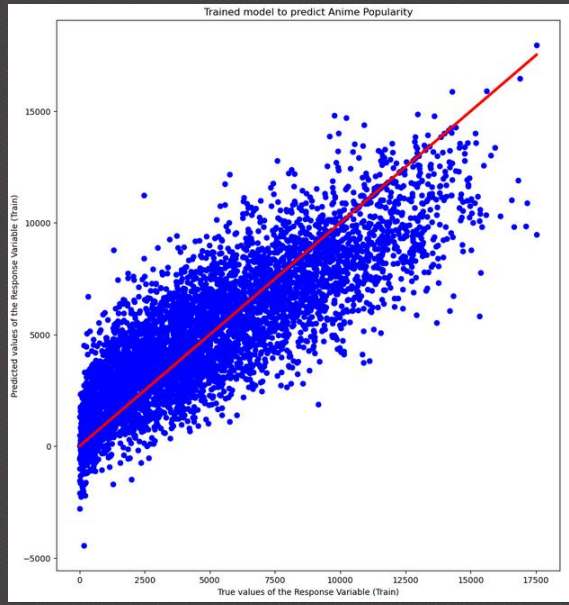
1. **Linear Regression**
    - As a starting point as response was numeric
2. **Random Forest Regressor**
    - Account for non-linear relationships and interactions between variables
3. **Extreme Gradient Boosting**
    - Handle mix of categorical and continuous variables

What our Data was like:

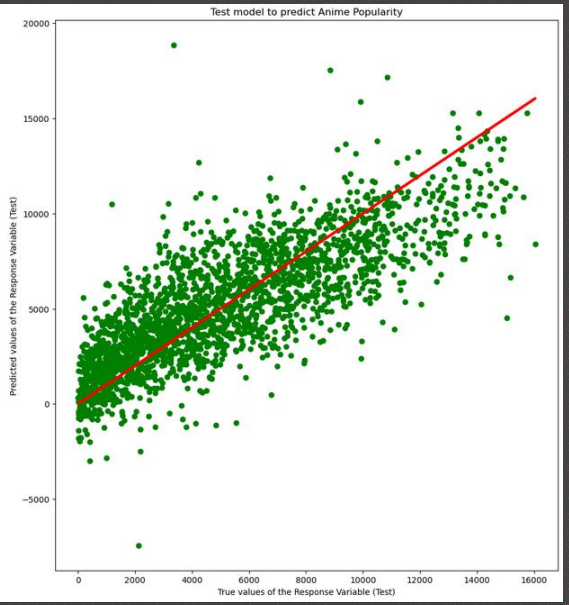
1. Response variable was Numeric
2. There were many predictors (Both categorical and numeric)
3. All the Data was labeled

Linear Regression

Train Model



Test Model



Moderate Explained Variance

Moderate MSE: sqr(49x10^5) = 2200

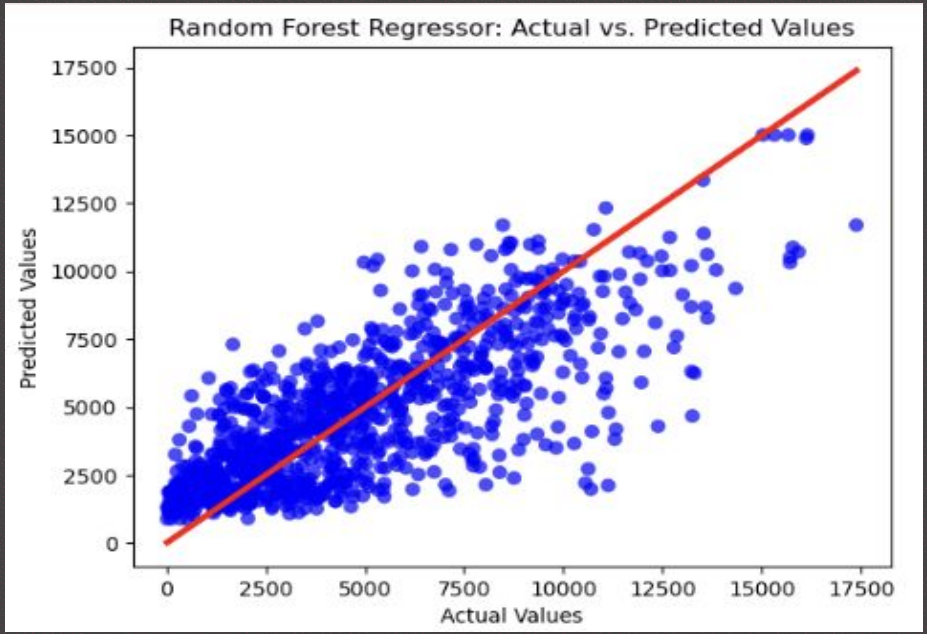
- 2200/24000 = 9%

Moderate Precision and Accuracy

```
Goodness of Fit of Model        Train Dataset
Explained Variance (R^2)        : 0.6012963606521916
Mean Squared Error (MSE)        : 4915120.184791429

Goodness of Fit of Model        Test Dataset
Explained Variance (R^2)        : 0.5708211632142839
Mean Squared Error (MSE)        : 4954797.134725979
```

Random Forest Regression



Moderate Explained Variance

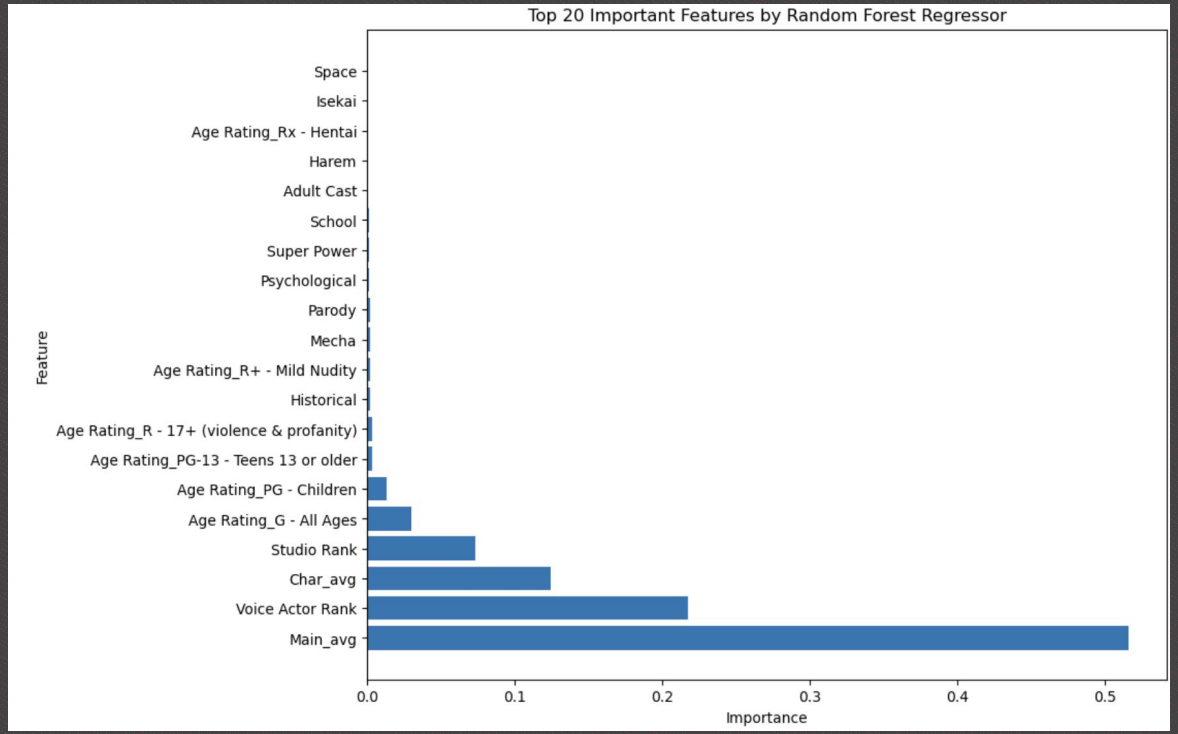
Moderate MSE: sqr(47x10^5) = 2100

- 2100/24000 = 8.8%

Moderate Precision and Accuracy

```
Best explained variance: 0.5924023873086445
Mean squared error: 4705645.550021893
```

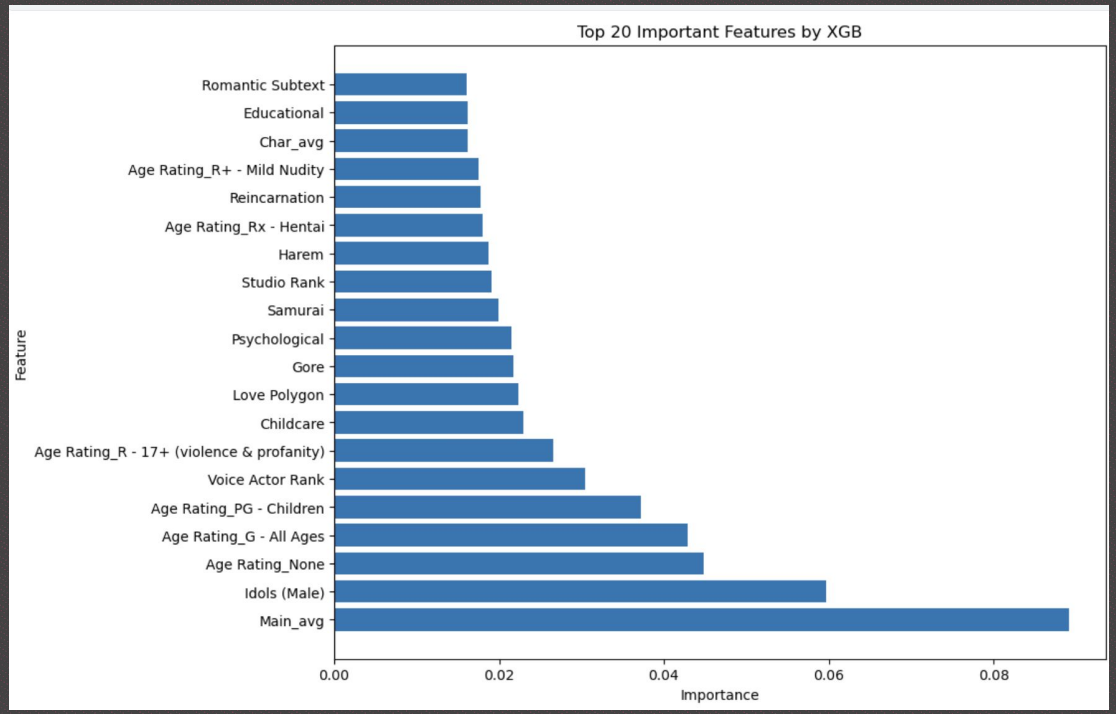
Random Forest Regression



Importance of Features:

	Feature	Importance
59	Main_avg	0.501336
58	Voice Actor Rank	0.218634
60	Char_avg	0.138603
57	Studio Rank	0.071943
50	Age Rating_G - All Ages	0.031007

Extreme Gradient Boosting



Moderate Explained Variance

Moderate MSE: sqr(42x10^5) = 2050

- 2050/24000 = 8.5%

Moderate Precision and Accuracy

```
Best explained variance: 0.6229724273541373
Mean squared error: 4267435.798278369
```

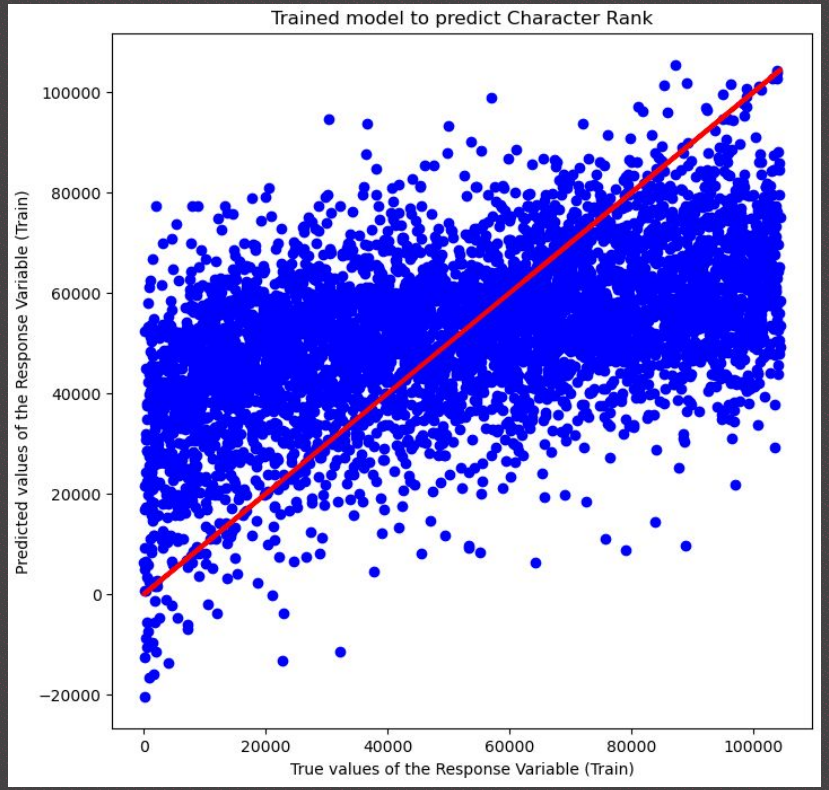
ML for predicting Character's Rank

Response : Rank

Predictors :

1. Character's Tag
2. Hair Colour
3. Gender

Linear Regression



Low Explained Variance

Moderate MSE: sqr(65x10^6) = 8000

- 8000/100,000 = 8%

Low Precision, moderate Accuracy

```
Goodness of Fit of Model        Train Dataset
Explained Variance (R^2)        : 0.2794512505618745
Mean Squared Error (MSE)        : 654368141.7571347
```

Analysis of all ML models

1. All the models had an explain variance of around 0.6
2. The Mean Square Errors were moderately high, with it being lowest for the XGB Model
3. XGB was the best model
4. However, even though the model can be explained by the predictor variables to about 60% the predicted value for Popularity is still about 10% off.
5. The models are good for giving a general indication of how good an Anime may be.

What we learned

- Regex to clear string
- One Hot encoding
- Using a different Machine Learning Function
  - Random Forest
  - Extreme Gradient Boosting
  - RandomSearchCV

Conclusion

Project Outcome:

- What makes an Anime good is more than the themes or just the characters
- Characters play a large role in how popular the Anime will be
- Ultimately, there are also many unknown or intangible elements that contribute to an anime's success or failure.