Anime Analysis

Abstract:
Our goal as a team was to perform some analysis on the Anime. How many releases, ratings, and votes are there? Which year was the most releases?Which MediaType is preffered?How many people watch and drop it?and so on.

So, we decided to look at the Anime data over the past 100 years and is well recorded. This allowed us to shed light on major progress in the movie releases and their ratings and votes.

In this year there hasn't been a major anime, like every year we have one anime that is like the anime of the year, something that everybody is talking about but this year till now there hasn't been one yet

Introduction:
Anime terminology is as niche as its fans. The word cour is used to measure the length of an anime series. In general, a single cour has 10 to 14 episodes that run during a three-month period that coincides with the seasons. That's why one cour belongs to either the Winter, Spring, Summer, or Fall season.
Winter Season: January, February, March
Spring Season: April, May, June
Summer Season: July, August, September
Fall Season: October, November, December

Anime are broadcasted in cours for various reasons:

1. Convenience: Creating an anime in single cours instead of full-blown 24-episode runs (or more) leaves the production company with more options. If the first cour is popular and the ratings are good, then they can follow up with a second cour back-to-back.
2. Inspection: If the fans like the show but there are certain complaints that need to be addressed, the series can go to the split-cour format and

skip a season between the first and second cour. If the show is a total flop, or if the ratings are not promising, the company can conclude the show and start working on something new.

So here we are to clean the raw data and visualise them. The normalisation of data is also an important aspect. Normal Probability plot is plotted for each numeric data column. We also determine the correlation of different columns.

Dataset:

We considered 650 anime.

Source: www.kaggle.com

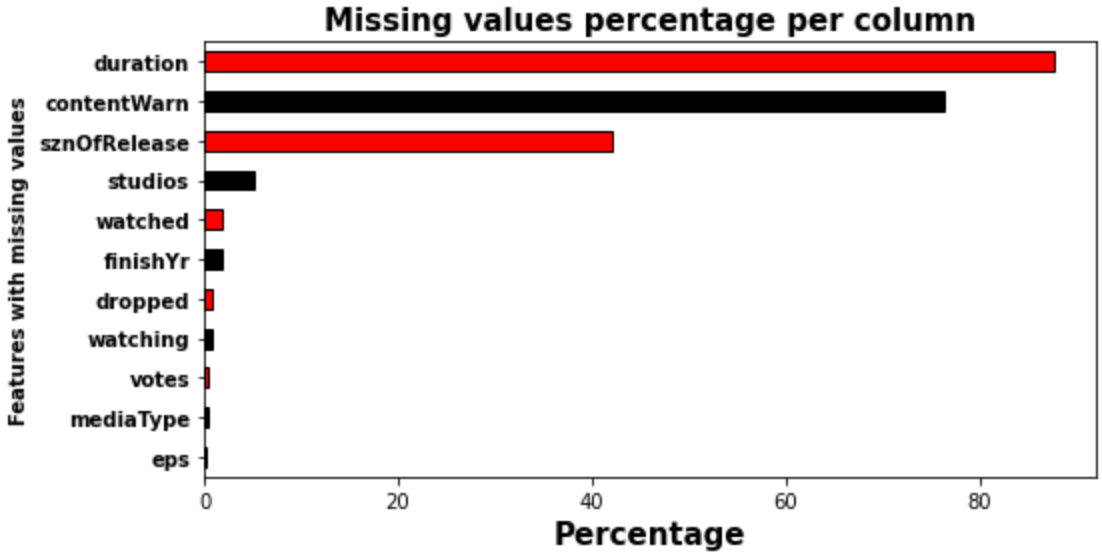
Features:

1. **Media Type** - mediaType (categorical, nominal)
2. **Number of Episodes** - eps (Discrete)
3. **Status (Ongoing or not)** - ongoing (categorical, nominal)
4. **Start Year** - startYr (discrete)
5. **Finish Year -** finishYr (discrete)
6. **Cour of Release** - sznOfRelease (categorical, nominal)
7. **Studio** - studios (categorical)
8. **Genre** - tags (categorical)
9. **Content Warning** - contentWarning (categorical)
10. **Number of people watching** - watching (discrete)
11. **Number of people who watched** - watched (discrete)
12. **Number of people who dropped** - dropped (discrete)
13. **Ratings** - rating (continuous)
14. **Number of Votes** - votes (discrete)
15. **Duration** - duration (discrete)

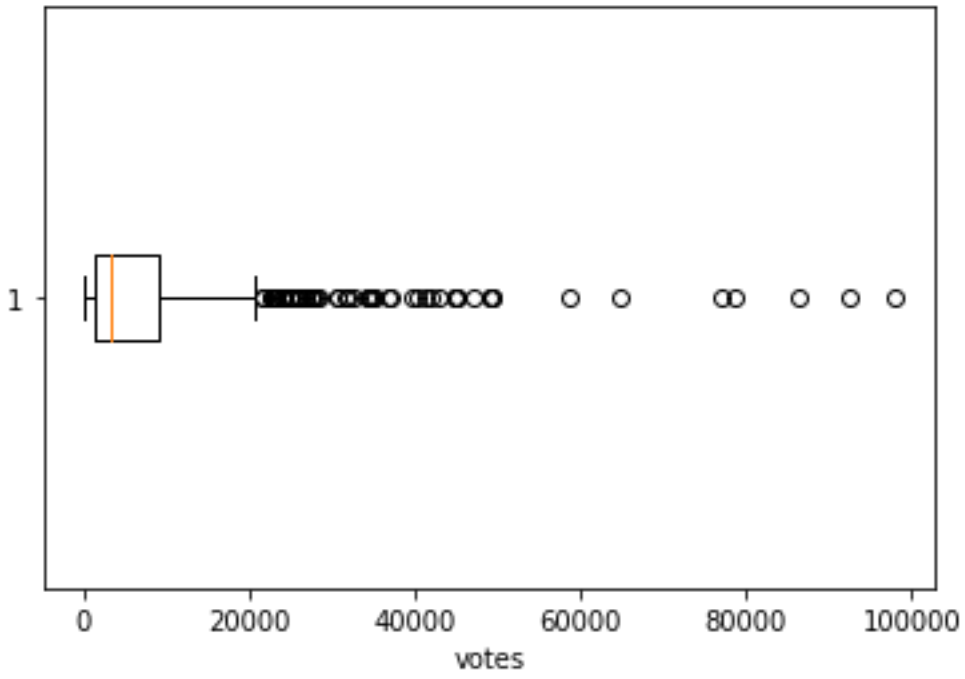
Preprocessing or Data Cleaning:

Data cleansing or data cleaning is the process of detecting and correcting (or removing) corrupt or inaccurate records from a record set, table, or database and refers to identifying incomplete, incorrect, inaccurate or irrelevant parts of the data and then replacing, modifying, or deleting the dirty or coarse data.

Finding the missing values in each column:



1. ‘Finish year’ - Imputing the null values of the finishYr column by finding the mode.
2. ‘Season of Release’ - Imputing the null values of the sznOfRelease column by finding the mode since it is categorical data
3. ‘Watched’ - Imputing the null values of the watched column by finding the mean.
4. ‘Watching’ - Imputing the null values of the watching column by finding the average of the above and before values i.e interpolation.
5. ‘Votes’ - Imputing the null values of the votes column by finding the median since this column has some outliers

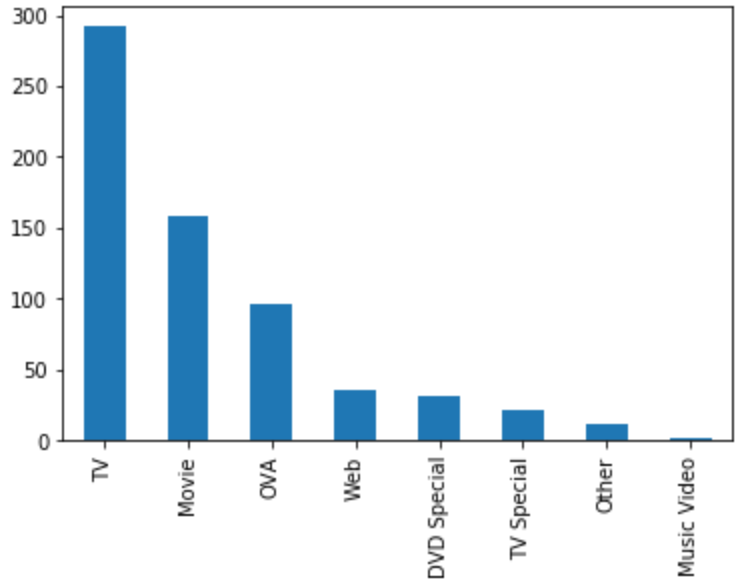


6.
7. ‘Dropped’ - Imputing the null values of the dropped column by finding the average of the above and before values i.e interpolation.
8. ‘Studios’ - Imputing the null values of the studios column by finding the mode since it is categorical data
9. ‘Media Type’ - Imputing the null values of the mediaType column by finding the mode since it is categorical data
10. ‘Number of Episodes' - Imputing the null value present in the "eps" column by interpolation method since we have single null value

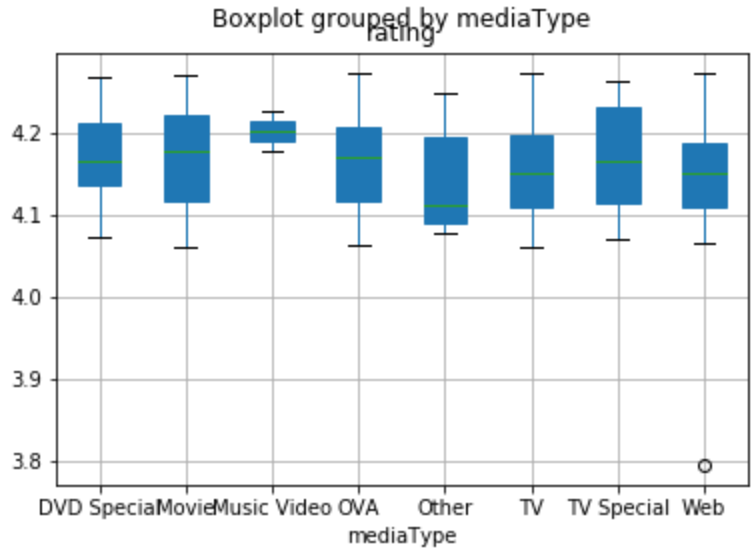
- The column "duration" is dropped because it has large number of missing values
- The column "contentWarn" is dropped because it has large number of missing values

Exploration of Data Analysis:

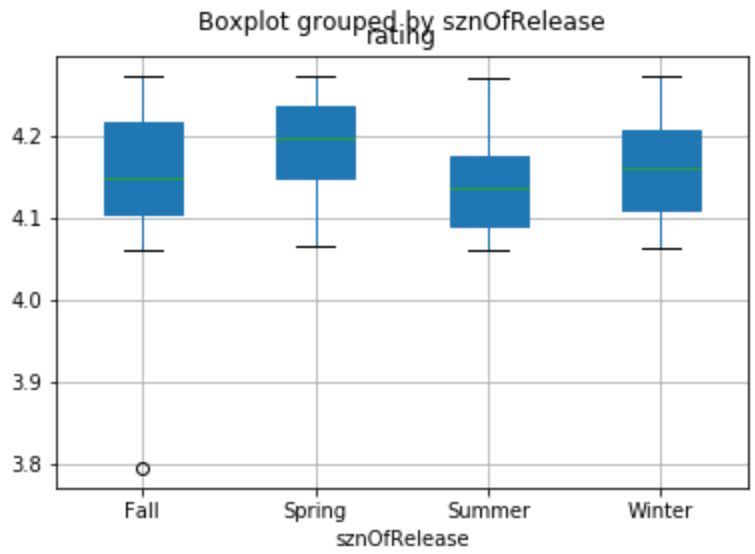
Exploratory Data Analysis refers to the critical process of performing initial investigations on data so as to discover patterns,to spot anomalies,to test hypotheses and to check assumptions with the help of summary statistics and graphical representations.



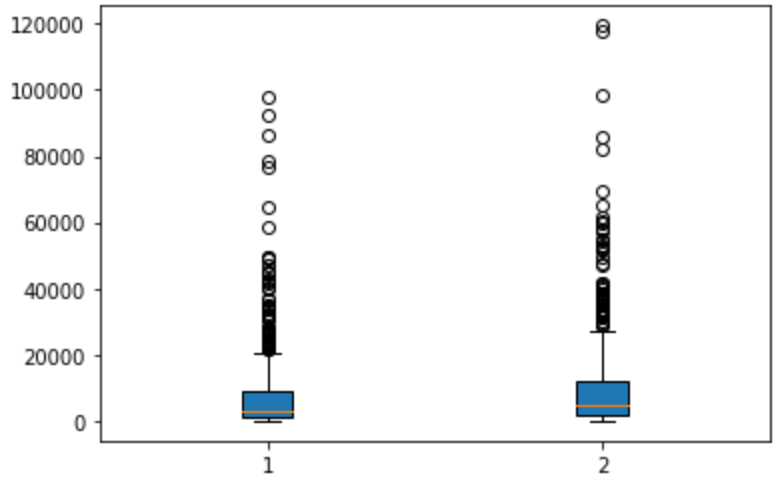
Since most viewers watch through "TV" we observe that the most of the anime are released on the "TV" mediaType and least are released in "Music Video" type



We observe that only "Web" mediaType has an outlier based on the ratings



We observe that only "Fall" sznOfRelease has an outlier with respect to the rating.



Above boxplot comparision shows that median of the "watched" column is greater than that of the "votes" column and also the overall range of the dataset is greater for the "watched" column than the "votes" column.

- Standardization:

Standardization is another scaling technique where the values are centered around the mean with a unit standard deviation. This means that the mean of the

attribute becomes zero and the resultant distribution has a unit standard deviation.

```
from sklearn import preprocessing
x=preprocessing.scale(data.eps)
print(x.mean().round())
print(x.var())
```

```
-0.0
1.0000000000000004
```

```
x=preprocessing.scale(data.startYr)
#print(x)
print(x.mean().round())
print(x.var())
```

```
0.0
1.0
```

```
x=preprocessing.scale(data.finishYr)
#print(x)
print(x.mean().round())
print(x.var())
```

```
0.0
1.0000000000000002
```

```
x=preprocessing.scale(data.watched)
print(x.mean().round())
print(x.var())
```

```
0.0
1.0000000000000002
```

```
x=preprocessing.scale(data.watching)
print(x.mean().round())
print(x.var())
```

```
0.0
1.0
```

```
x=preprocessing.scale(data.rating)
print(x.mean().round())
print(x.var())
```

```
-0.0
1.0
```

```
x=preprocessing.scale(data.votes)
print(x.mean().round())
print(x.var())
```

```
-0.0
1.0
```

- Normalization:

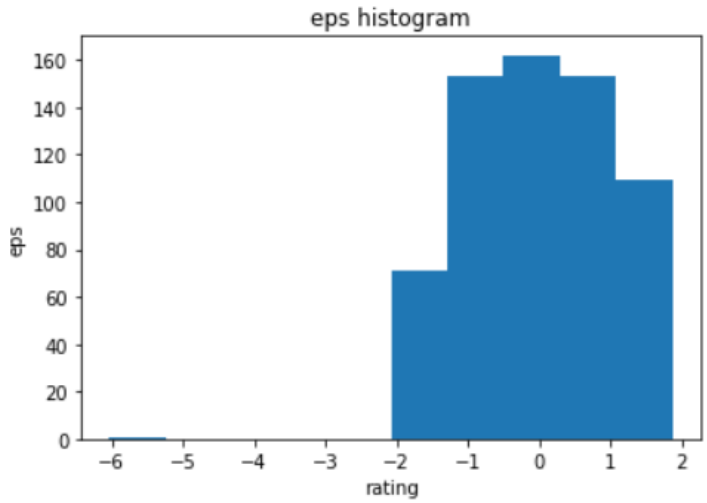
Normalization is a scaling technique in which values are shifted and rescaled so that they end up ranging between 0 and 1. It is also known as Min-Max scaling. Using the following to code to print normalized data of ‘eps’, ‘watched’, ‘rating’, ‘watching’ and ‘dropped’

```
X_Data = data[['eps','watched','rating','watching','dropped']]
normalized_Data = (X_Data - X_Data.mean())/X_Data.std()
print(normalized_Data)
```

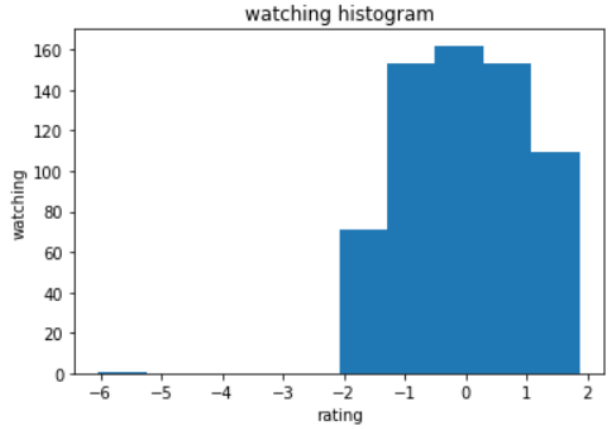
```
          eps   watched    rating  watching   dropped
0    0.135529  1.012132  1.850333  1.838859  1.759091
1   -0.070034  0.367815  1.866797  0.198720  0.030220
2   -0.275596 -0.683033  1.718616 -0.524988 -0.507826
3   -0.296153 -0.347935  1.833868 -0.488437 -0.494639
4   -0.275596 -0.053455  1.833868 -0.367487 -0.436614
..        ...       ...       ...       ...       ...
620 -0.049477 -0.482552 -1.524893 -0.004636 -0.403645
621 -0.172815 -0.628453 -1.491963 -0.502393 -0.439252
622  0.217754 -0.703272 -1.491963 -0.482456 -0.515739
623  0.176642 -0.451313 -1.491963 -0.300366 -0.023848
624 -0.070034 -0.389399 -1.491963 -0.336253 -0.325840
625 -0.049477  1.848749 -1.508428  0.879896  2.380218
626 -0.296153 -0.534171 -1.491963 -0.513691 -0.490682
627 -0.070034 -0.160500 -1.475499  0.329639 -0.313971
628 -0.296153 -0.661314 -1.508428 -0.527646 -0.502551
629 -0.070034  0.240814 -1.524893 -0.169447  0.580136
630 -0.070034 -0.026165 -1.524893  0.317677  0.494418
631 -0.090590 -0.233838 -1.541357  0.006661 -0.170228
632 -0.070034  1.641075 -1.541357  0.035238  0.052639
633  0.197198 -0.652781 -1.541357 -0.354860 -0.452439
634  0.217754 -0.451031 -1.541357 -0.316980  0.177920
635 -0.070034 -0.296880 -1.557822  0.091061  0.164732
636 -0.213927  0.344545 -1.557822  1.969113 -0.184734
637  0.217754 -0.498560 -1.557822 -0.387424 -0.233528
638 -0.111146  0.332204 -1.574286  0.175460  0.627610
639 -0.193371 -0.202810 -1.574286 -0.471823 -0.441889
640  0.053304 -0.506458 -1.574286 -0.377455 -0.451120
641 -0.296153  0.383823 -1.590751 -0.491096 -0.501232
642 -0.070034  0.472957 -1.590751  0.196726  0.071101
643 -0.070034 -0.386578 -1.590751 -0.461190 -0.444526
644 -0.070034  0.124460 -1.590751 -0.306347 -0.259903
645  0.217754 -0.294906 -1.607215 -0.233910  0.135720
646 -0.296153 -0.447294 -1.607215 -0.519007 -0.495957
647 -0.296153 -0.370359 -1.640144 -0.463848 -0.460351
648 -0.049477  0.675694 -1.623680  0.621381  1.176207
649 -0.296153  0.448910 -1.623680 -0.430620 -0.378589

[650 rows x 5 columns]
```

Using graphs to verify that the column are normally distributed:

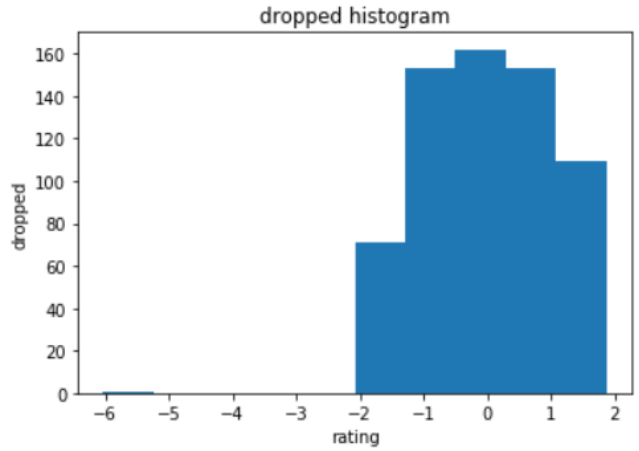
```
freq,bins,patches = plt.hist(normalized_Data.rating)
plt.xlabel('rating')
plt.ylabel('eps')
plt.title('eps histogram')
plt.show()
```




```
freq,bins,patches = plt.hist(normalized_Data.rating)
plt.xlabel('rating')
plt.ylabel('watching')
plt.title('watching histogram')
plt.show()
```



```
freq,bins,patches = plt.hist(normalized_Data.rating)
plt.xlabel('rating')
plt.ylabel('dropped')
plt.title('dropped histogram')
plt.show()
```



Since it forms a u bell shaped curve we can conclude that our data is normally distributed.

Hypothesis Testing:

Hypothesis testing in statistics is a way for you to test the results of a survey or experiment to see if you have meaningful results. You're basically testing whether your results are valid by figuring out the odds that your results have happened by chance. If your results may have happened by chance, the experiment won't be repeatable and so has little use.

Test if sample of votes is from the population of votes:

```
votes=data['votes']
mean_votes=votes.mean()
sigma_votes=votes.std(ddof=0)
print("mean: ", mean_votes, ", sigma:", sigma_votes)
```

```
mean:  7468.912307692308 , sigma: 11615.57786952846
```

```
sample=votes.sample(n=300)
x_bar=sample.mean()
print("sample_mean: ",x_bar)
```

```
sample_mean:  6357.343333333333
```

H0: The sample is from the anime population, x_bar = μ(mean)

HA: The sample is not from the anime population, x_bar != (not equal) μ(mean)

If the p-value is greater than alpha then we say H0 is plausible , otherwise we reject the null and accept the alternative hypothesis.

alpha level of α = 0.05¶

```
# alpha = 0.05
z_critical = 1.96 # alpha level of 0.05 and two-tailed test
x_bar = 6234.243333333333
N = 300
SE = sigma_votes/np.sqrt(N)
z_stat = (x_bar - mean_votes)/SE
print(z_stat)
```

```
-1.8410701715742654
```

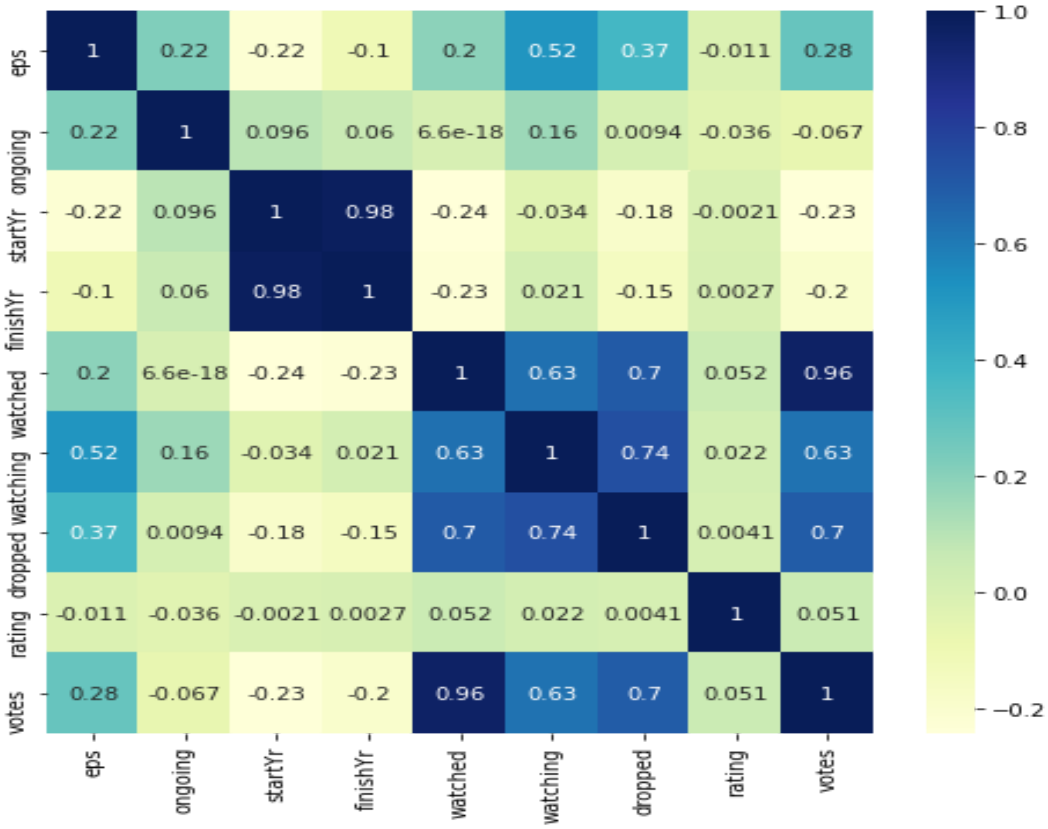
p-value = 0.03*2 = 0.06(since two tailed test) (from z-table)

Since p-value > alpha it is plausible for H0 to be true and therefore we reject the althernative

Statistically, we say the sample mean is no different than the population mean and thus the sample is drawn from the population.

- **Correlation:**

Correlation is used to test relationships between quantitative variables or categorical variables. In other words, it's a measure of how things are related.



Positively Correlated:

- startYr and finishYr
- watched and votes
- votes and dropped

Results and Discussion:

With Anime craze growing constantly and the support from the streaming giant Netflix, anime is becoming a common global currency and also very accessible.

The wait for a new cour or season is no longer significant. The show can be viewed at any time and anywhere the viewer wants.

Even though the dataset points towards “Fall” being the season with most releases, the same cannot be said for the entire population.

The success of anime is not really based on the cour or season of its release; instead it depends on the other aspects. Domestic market is affected due to the production of merch and publicity but when it comes to the international fan community, such details are of minimal use.

The medium in which it is available plays a key role in the ratings and range of the audience. The series viewed through TV tend to have higher ratings.