**TMDB\_Movies\_DATA\_ ANALYSIS**

**Group Members:**

| **First name** | **Last Name** | **Student number** |
| --- | --- | --- |
| Mary | Gomez | C0891136 |
| Rehan | Khokhar | C0896278 |
| Andres | Correal | C0872634 |
| Viki | Patel | C0906295 |
| Bhavya | Vadher | C0894977 |
| Prashant | Bhattarai | C0898956 |

**Submission date:** 10 November 2023

Contents

[Abstract 3](#_gjdgxs)

[Introduction](#_30j0zll) 3

[Methods](#_1fob9te) 4

Step:1……………………………………………………………………………………………………….4

Step:2……………………………………………………………………………………………………….5Step:3……………………………………………………………………………………………………….7Step:4……………………………………………………………………………………………………...16Step:5……………………………………………………………………………………………………...17Step:6……………………………………………………………………………………………………...18Step:7……………………………………………………………………………………………………...19

[Results](#_3znysh7) 24

[Conclusions and Future Work](#_2et92p0) 32

[References](#_tyjcwt) 32

Github Repository link : <https://github.com/mlgomez0/advanced_python_project/tree/main/notebooks>

# 

# Abstract

In this project we try to scrape data from a website which keeps data about movies. The data used in this report was pulled from TMDB ([https://www.themoviedb.org](https://www.themoviedb.org/)) using their public API.Here we perform all the steps of all the data life cycle like organize, transform , analyze and share our insights.All of our scraped data was text data and numerical so we used libraries for NLP to manipulate our data and clean the data so we can use the data in our Machine learning model. We used 2 machine learning models those are K-means clustering and Topic Modeling LDA (Latent Dirichlet Allocation)

# Introduction

In the 21st century movies are one of the most consumed digital content by masses and by analyzing this textual data will give us insights into preferences of the masses and try to find patterns for the same . Moreover this study will help us to understand their likes and dislikes and the relation of movies with other factors of movie making , its demographics and trends in current movies.Analysing all this will give us insight about the factors which contribute to success or failures of the movies.

This project revolves around the exploration of the textual movie data which we scraped TMDB using their public API.In this project we extracted data in different files then merged data files, did pre-processing , analyzed the data using it in our machine learning models and we shared our insights and result at the end of the report.

# 

# 

# 

# Methods

**Step:1**

The first set towards this problem was to collect data soThe data used in this report was pulled from TMDB (https://www.themoviedb.org) using their public API. To be able to use the API, we created a user and sent a GET request to the authentication endpoint (https://api.themoviedb.org/3/authentication). By doing this, we were able to confirm proper client setup. Furthermore, we conducted GET requests to the endpoint (https://api.themoviedb.org/3/discover/movie) to get as many records as possible. We stored those records in CSV format.

The data collected has the following features:

GenreIds: This is a list of different genres applicable to the movie. The map between genre id and genre is below:

* Action 28
* Adventure 12
* Animation 16
* Comedy 35
* Crime 80
* Documentary 99
* Drama 18
* Family 10751
* Fantasy 14
* History 36
* Horror 27
* Music 10402
* Mystery 9648
* Romance 10749
* Science Fiction 878
* TV Movie 10770
* Thriller 53
* War 10752
* Western 37

Id: this is the unique identifier for each movie OriginalLanguage: The movie's original language OriginalTitle: The movie's title Overview: short description of the movie content Popularity: it is a metric used to measure how popular the movie is, it takes into account the following aspects

* Number of votes for the day
* Number of views for the day
* Number of users who marked it as a "favorite" for the day
* Number of users who added it to their "watchlist" for the day
* Release date
* Number of total votes
* Previous days score

ReleaseDate: movie data of release Title: title of the movie VoteAverage: It's the average of all of the TMDb user ratings VoteCount: number of people who voted for the movie

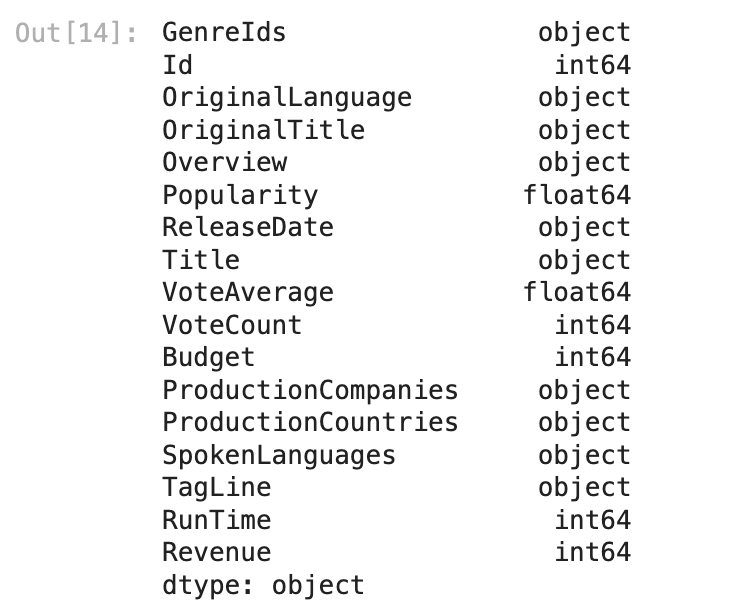
**Step 2**

**Data Wrangling**

In this section, we conducted several steps to clean up and prepare the data for analysis. The steps we followed are listed below:

* Merge data sets: So far, we have two CSV files, one with overall movie information and one with detailed movie data. We wanted to merge these two datasets using the movie "Id" as the primary key.
* Features and datatypes: In this steps we will process features into the proper data type, inspecting and correcting invalid values
* Inspect and impute nulls
* Inspect and correct duplicated rows
* Text Wrangling
* create a new cleaned dataset (movies\_cleaned.csv)

**Features and description:**



* Eliminate SpokenLanguages as there is already a column for original language and we believe the translations may not be predictive to the model.
* Preprocessing of column 'GenreIds' and making a dictionary to map genre ids to genre names.
* Checking the cardinality for the column ProductionCompanies which is 9980
* The cardinality for the column ProductionCompanies is high, with 9980 different production companies. Therefore, we won't consider this column as it might not be discriminative enough.
* The column ProductionCountries had high cardinality (107). On the other hand, we considered this column to be meaningful to the model. Therefore, we decided to reduce the number of columns by creating groups of production countries based on continents.
* Using this strategy, we were able to reduce the cardinality for ProductionCountries from 107 to 6
* The column OriginalLanguage presents a high cardinality of 50; we looked to reduce this cardinality.
* As can be seen, English 'en' has a considerable share compared to other languages. Therefore, we are going to limit this to two categories only, English and not\_english

### Inspect and impute nulls

* checking number of nulls by column
* TagLine nulls will be replaced with empty strings throughout the code.
* Remove other nulls in ReleaseDate and Overview, there are just 72 records and there is not a reasonable way to impute those

### Inspect and correct duplicated rows

* Note: There were not duplicated rows

### Text Wrangling

* checking textual data for non english columns
* Filtering out OriginalTitle as it contains other languages and Title is the same column but in english

#### Remove punctuations, numbers and stop words

* We used NLTK to remove stopwords from data
* Processing the columns: 'Overview', 'Title', 'Genres
* After cleaning, title ended up with some empty strings
* We need to remove this rows as those will be parsed as nulls when imported
* Checking empty string in Genres and we found 49
* Dropping nulls in 'Genres'

### Create new cleaned dataset (movies\_cleaned.csv)

* We store final cleaned dataset in CSV

# **Step 3**

## Text Data Visualizations

### Words Cloud of Movie Titles

Figure :1



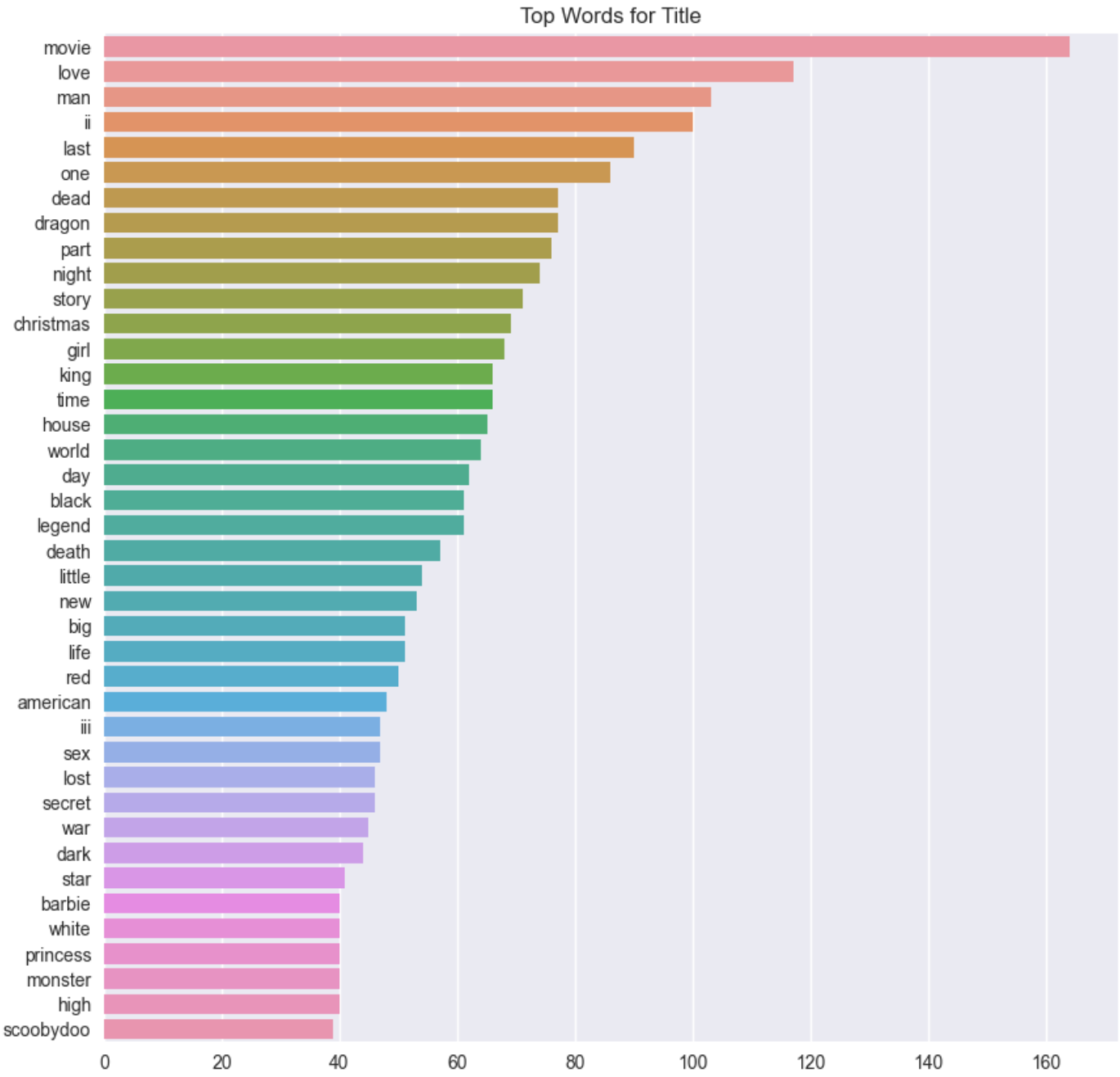


Figure:2

Here you can see that in movie titles most of the words like day,love,movie, last , girl are used the most.

### 

### Words Cloud of Movie Overviews



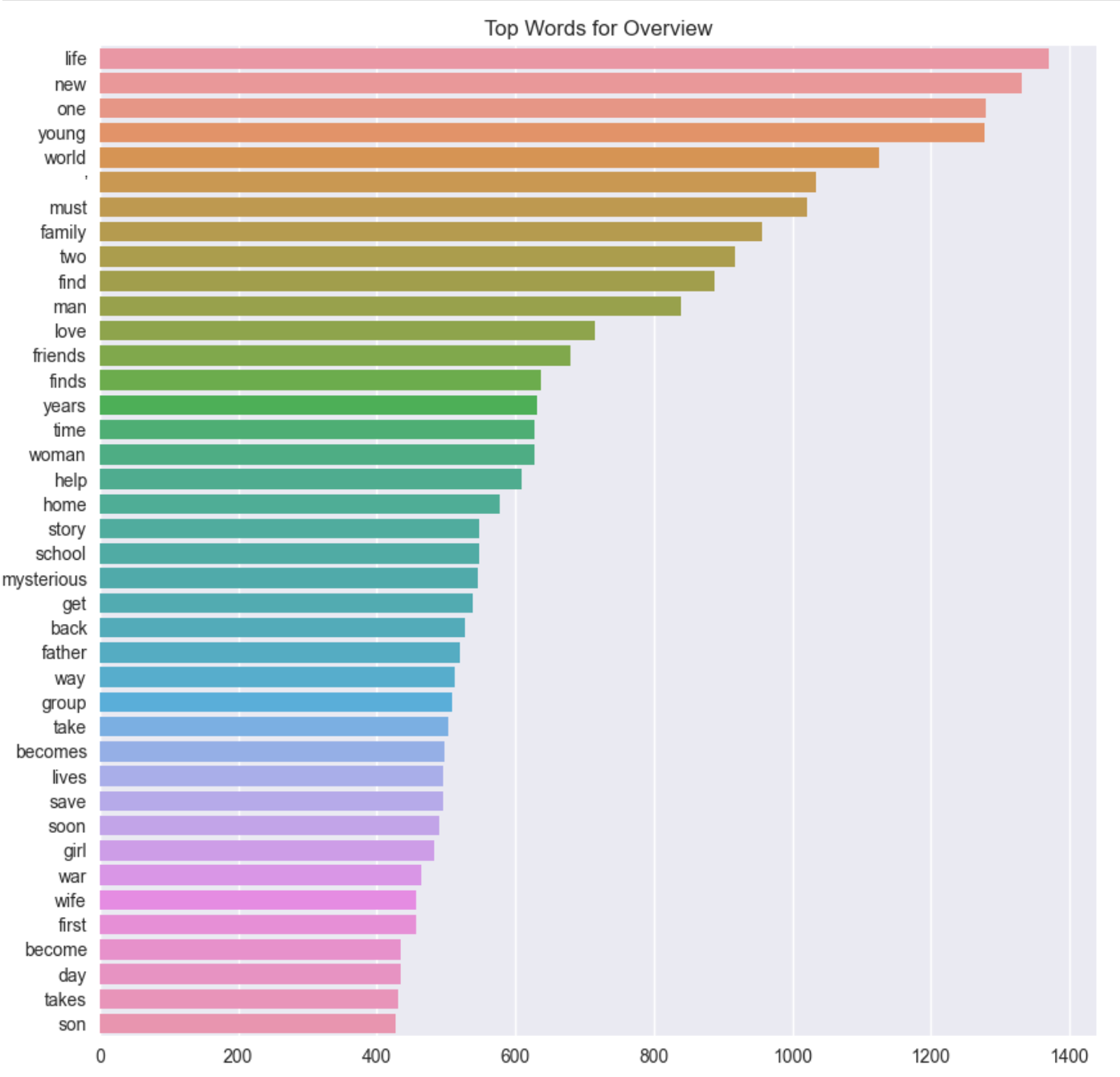
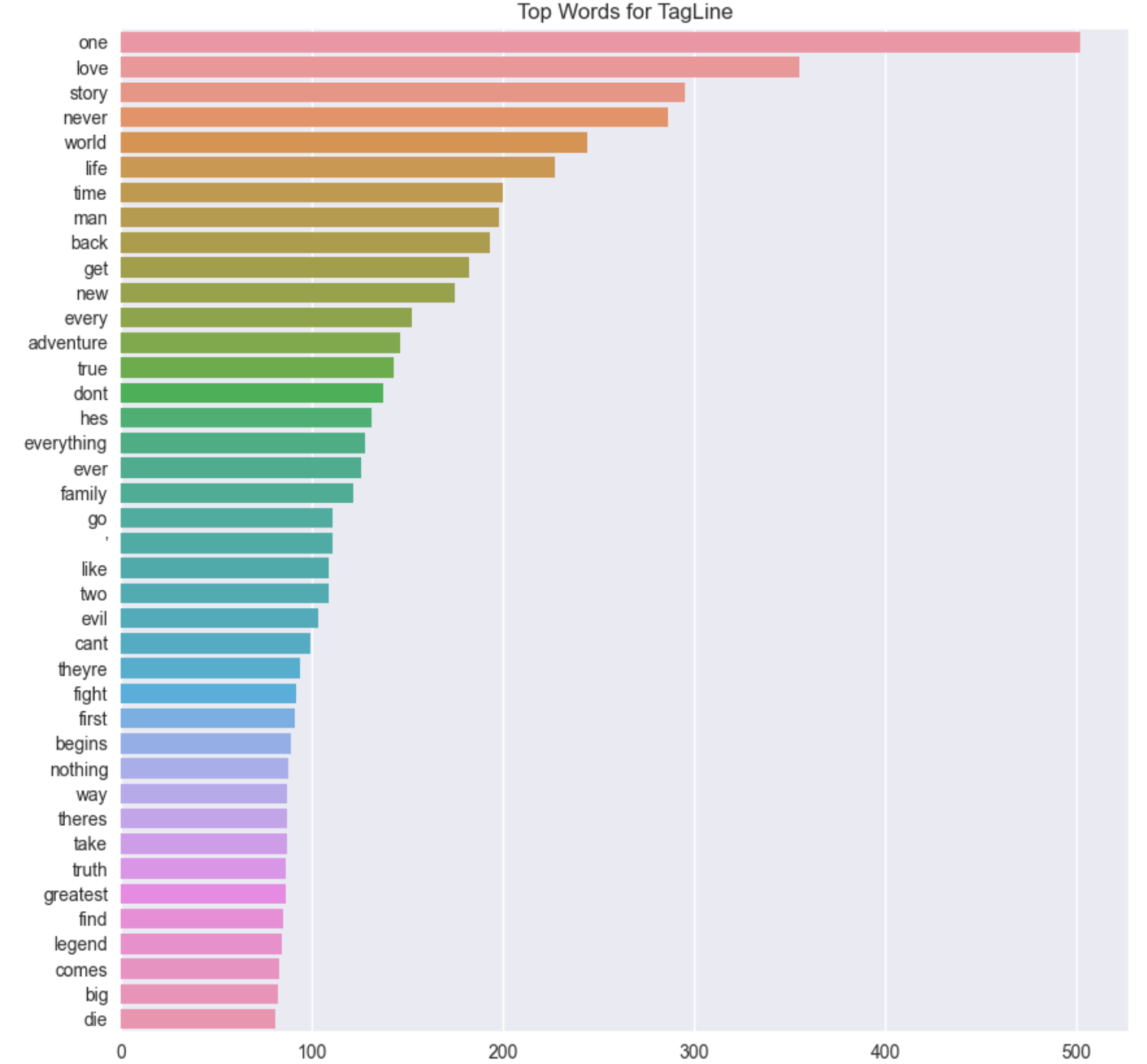


Figure :3

This is the word cloud of movie overview and you can see which words are used the most in overviews.

### Words Cloud of Taglines



Figure:4

This word cloud represents the words which are used most in the movie taglines.

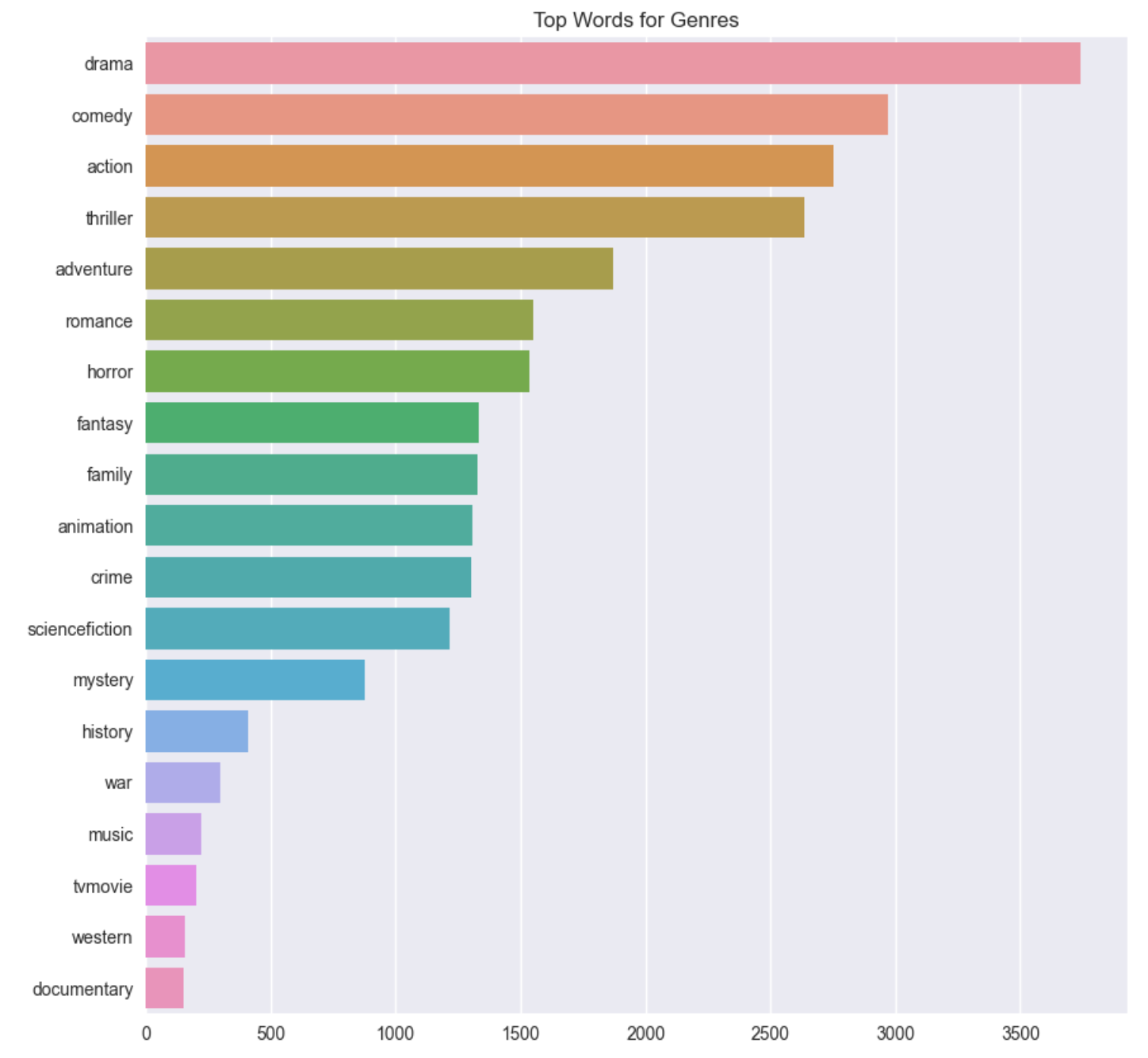
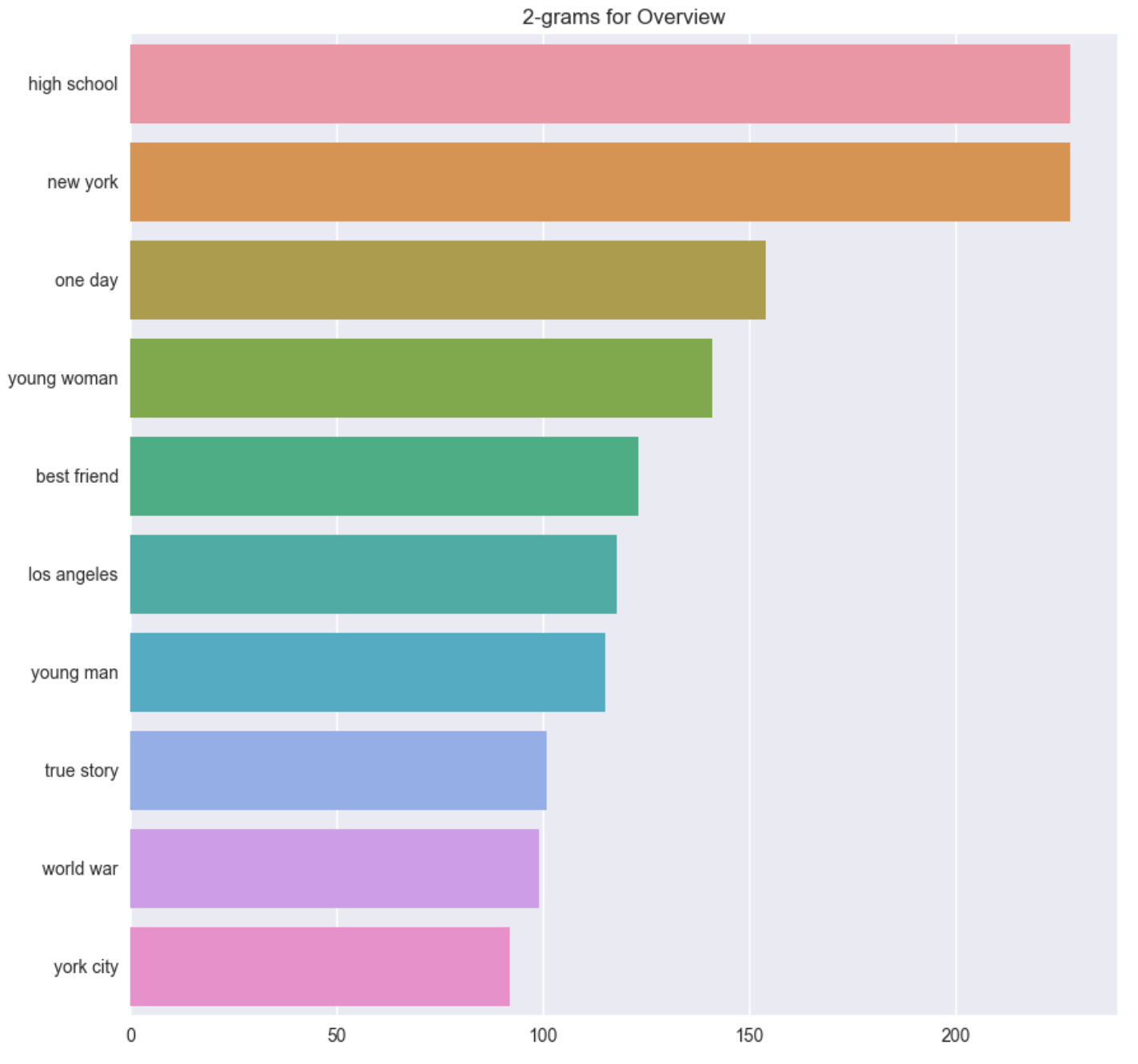


Figure:5

Note: We identified thanks to the previous graph, the top most frequent words for overview and title are generally positive and neutral. In terms of genres, we see drama in the top, followed by comedy and thriller.



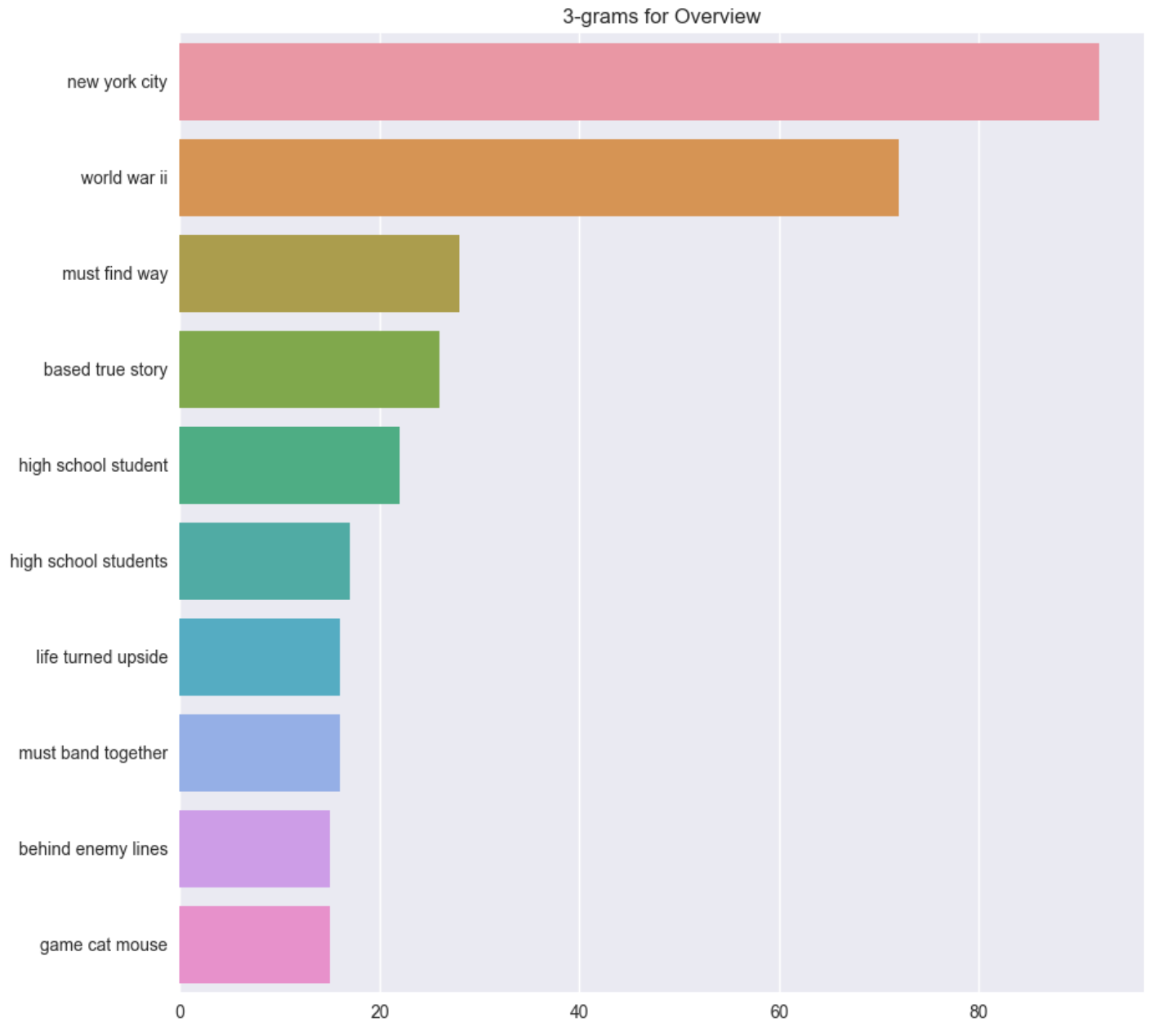


Figure:6

## Numerical Data Visualizations

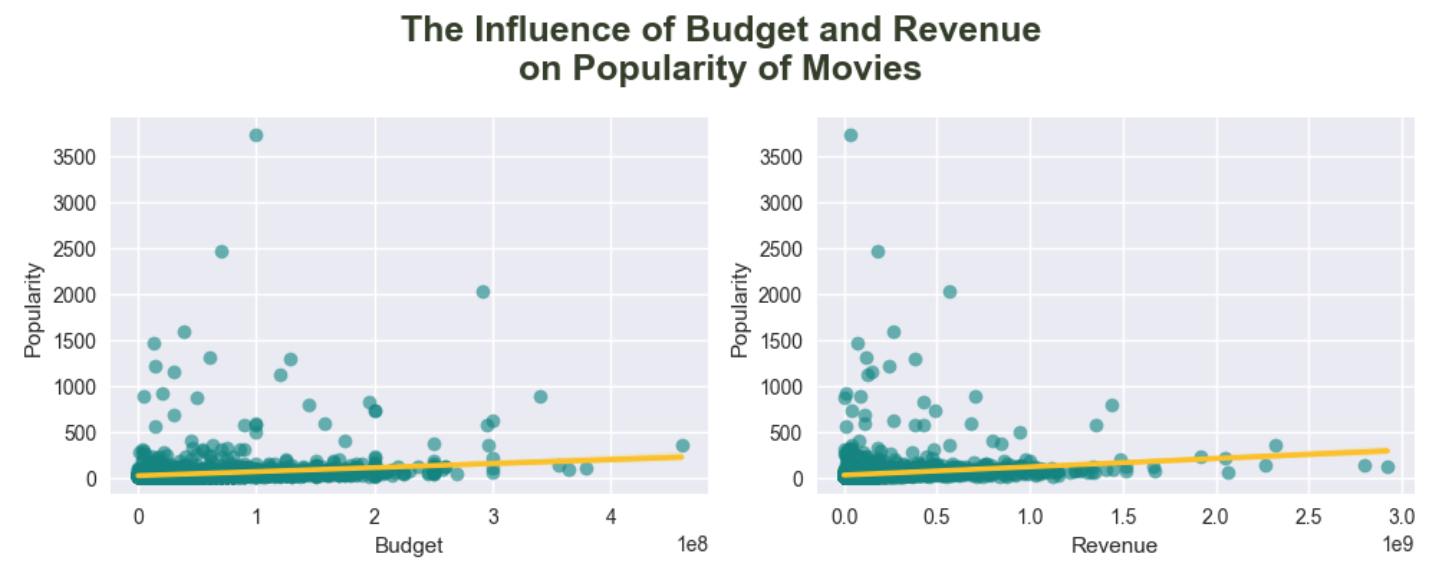


Figure:7

* It seems there may be a positive correlation between Budget and Popularity.

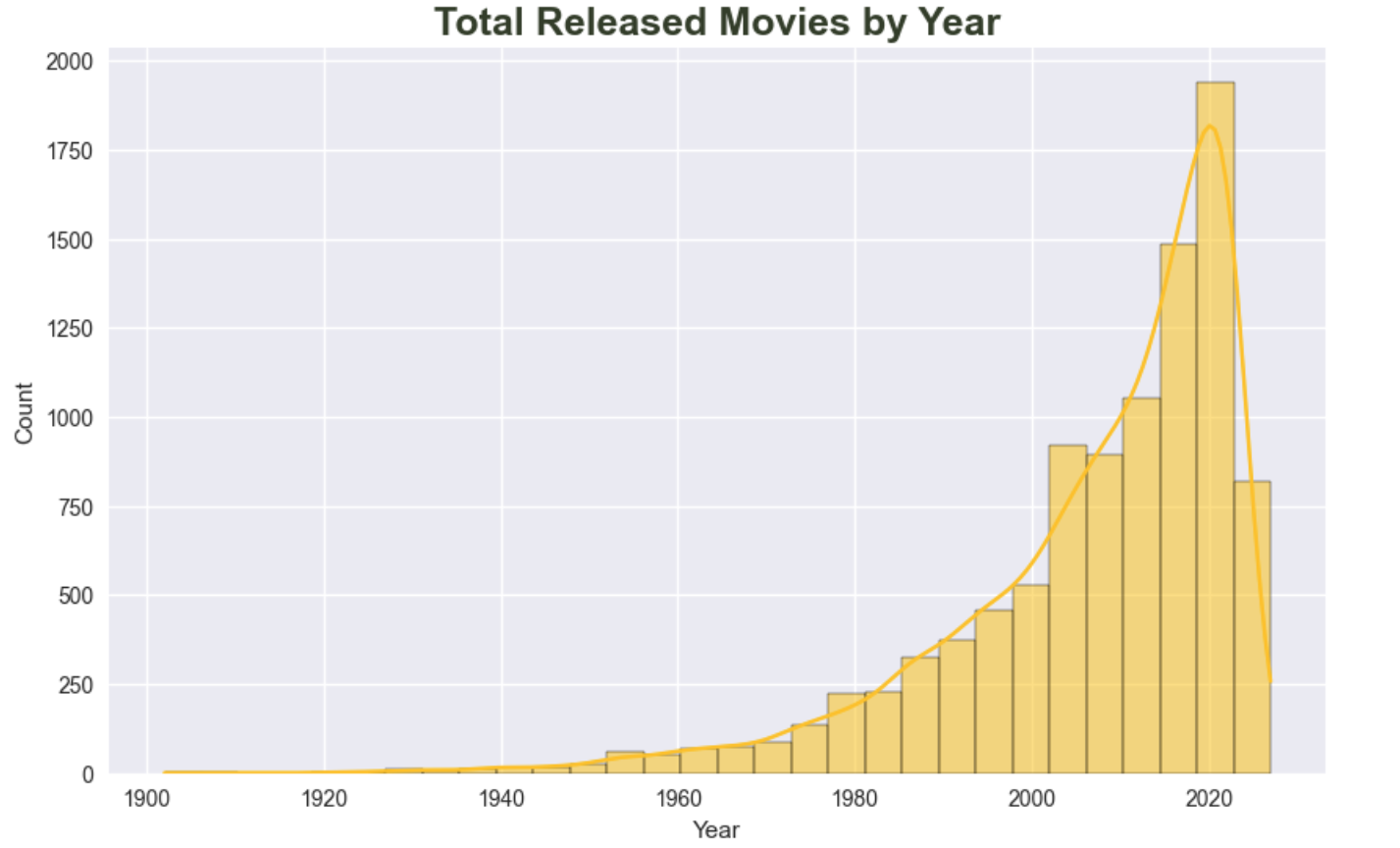


Figure:8

* The total releases are higher for more recent years.

**Correlations between numerical columns**

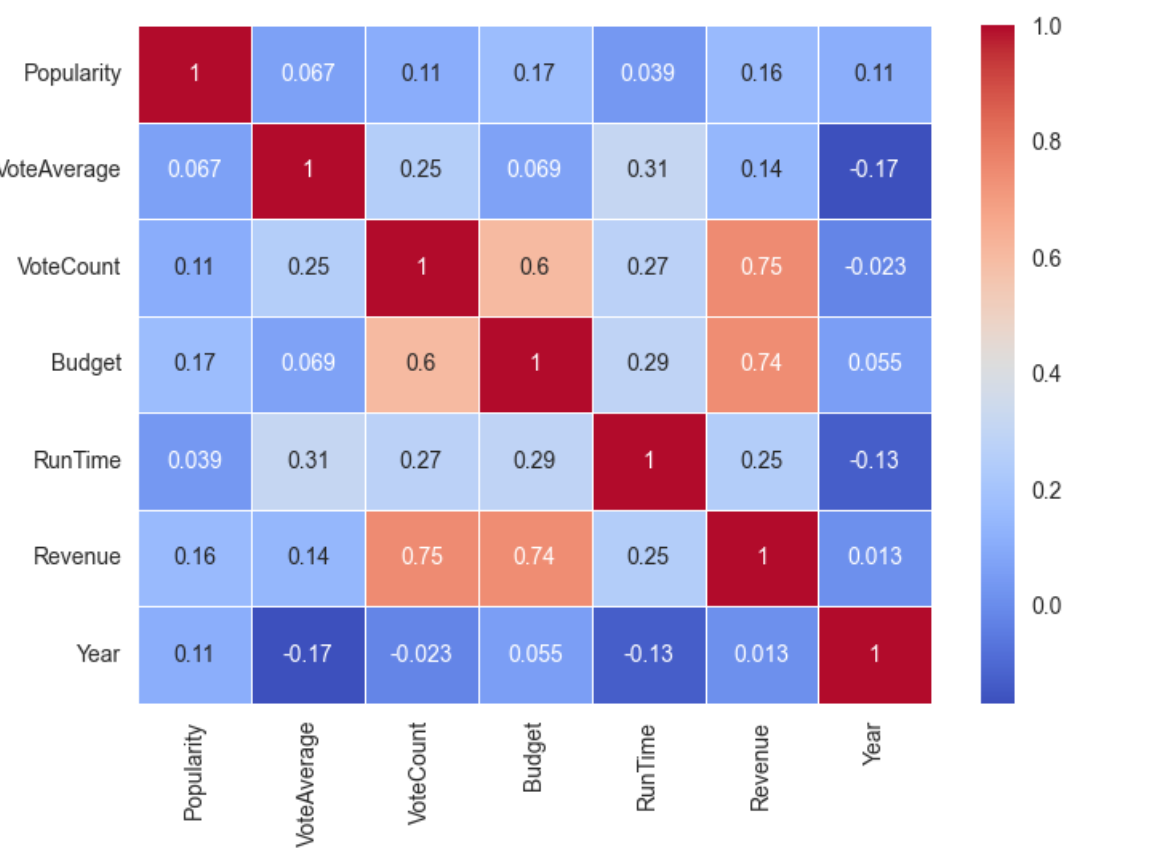
****

Figure:9

* The above Correlation matrix shows a potential correlation between Budget, VoteCount with Revenue.

# 

Figure:10

* Popularity shows a skewed right distribution. Additionally, it presents a long tail, which may indicate a high presence of outliers.
* VoteAverage and VoteCount presents symmetrical distributions
* Revenue and Budget present a non-symmetrical bimodal distribution
* RunTime exhibits a slightly skewed right distribution

# 

# **Step 4**

**Pandas profiling**

We attached pandas profiling html file with our submission.

The profiling report helps us to conclude the following:

* VoteCount is highly correlated with Budget and Revenue. Additionally, Budget and Revenue have a vast presence of zeros, 42% and 37%, respectively. Therefore, we decided to move forward with VoteCount and remove the other mentioned columns.
* We could see a high imbalance for the production continents. Additionally, high correlation with OriginalLanguage for some of the continents as Asia and North America. Therefore, we decided to remove modeling the production continents and keep the OriginalLanguage column.

# 

# 

# 

# 

# 

# **Step 5**

**Create a ordinal column with the next categories:**

**1: Low, 2: Medium, 3: High**

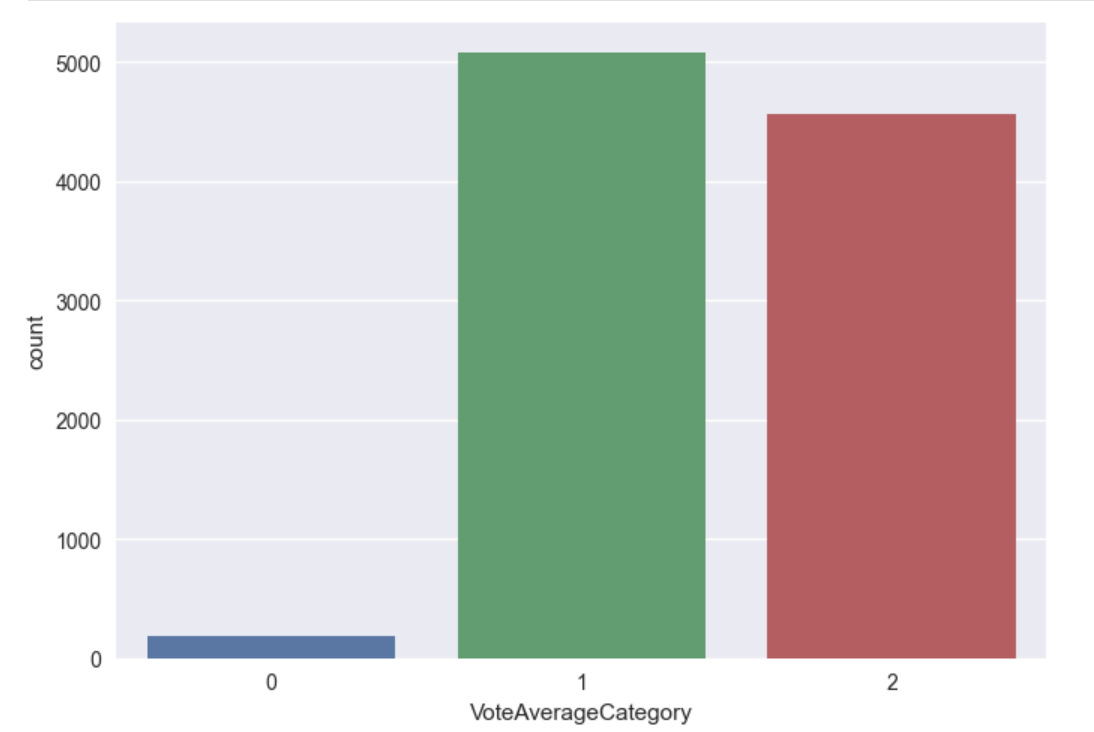


Figure:11

* Using np.digitize, we could transform a continuous variable 'VoteAverage' into categories. We created three categories which are initially imbalanced. Consequently, we can conclude that to create a proper discretization for this column, we would need further analysis of the variables.

# 

# 

# 

# **Step 6**

## Outliers Identification

* Function to create boxplot to see percentiles and outliers presence
* Popularity: Shows a distribution positively skewed (right-skewed) and the presence of outliers in the upper level. 25% of the data fall below 15.59, 50% of the data fall below 20.1, 75% of the data fall below 30.33. The upper extreme is at 52, and the lower extreme is at 13.05.
* VoteAverage: This shows a symmetric distribution and the presence of outliers in the upper and lower levels. 25% of the data fall below 6.00, 50% of the data fall below 6.60, 75% of the data fall below 7.10. The upper extreme is at 8.75, and the lower extreme is at 4.35
* VoteCount: Shows a distribution positively skewed (right-skewed) and the presence of outliers in the upper level. 25% of the data fall below 170, 50% of the data fall below 564, and 75% of the data fall below 1668. The upper extreme is at 3915, and the lower extreme is at 0.
* Budget: The budget was scaled down to facilitate plotting. It shows a distribution positively skewed (right-skewed) and the presence of outliers in the upper level. 50% of the data fall below 2.2M, and 75% of the data fall below 25M. The upper extreme is at 62.5M, and the lower extreme is at 0.
* RunTime: Shows a symmetric distribution and the presence of outliers in the upper and lower levels. 25% of the data fall below 91 min, 50% of the data fall below 101 min, and 75% of the data fall below 115 min. The upper extreme is at 151 min, and the lower extreme is at 55 min.
* Revenue: Revenue was scaled down to facilitate plotting. It shows a distribution positively skewed (right-skewed) and the presence of outliers in the upper level. 50% of the data fall below 8.8 M, and 75% of the data fall below 54.6 M. The upper extreme is at 136.5 M, and the lower extreme is at 0.

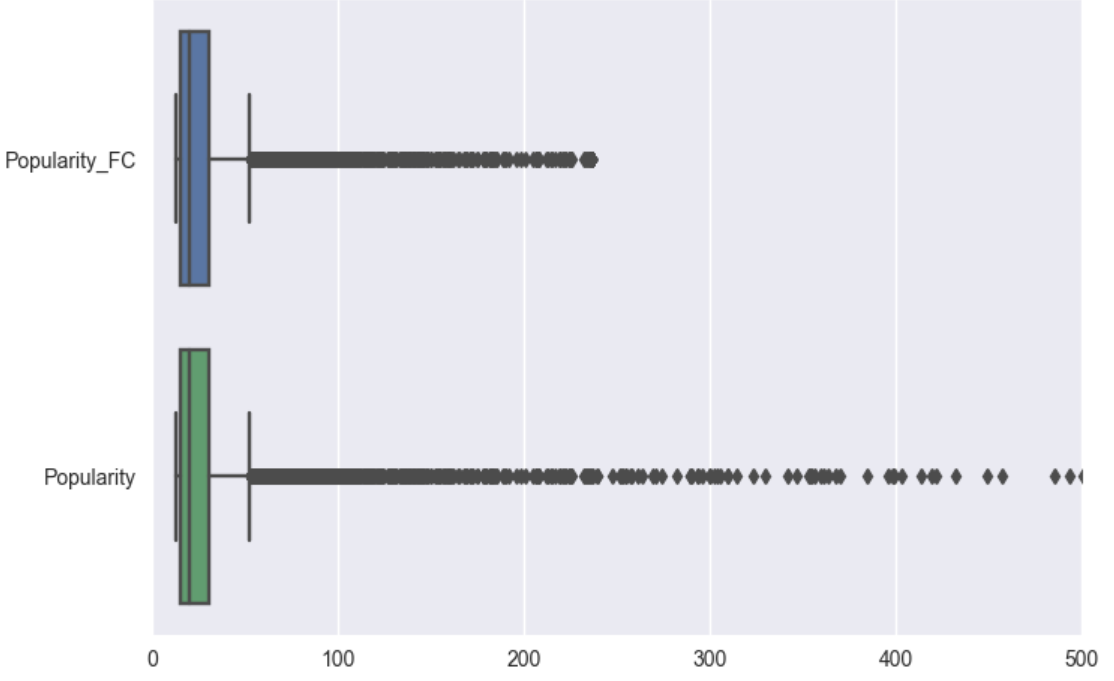
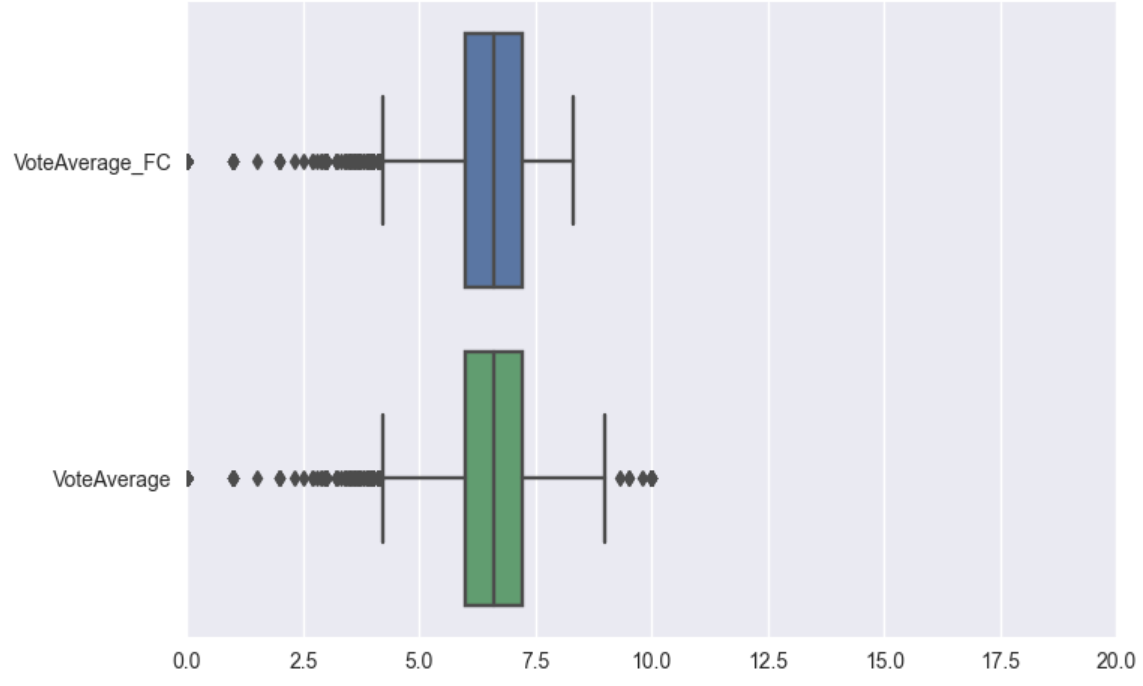
# 

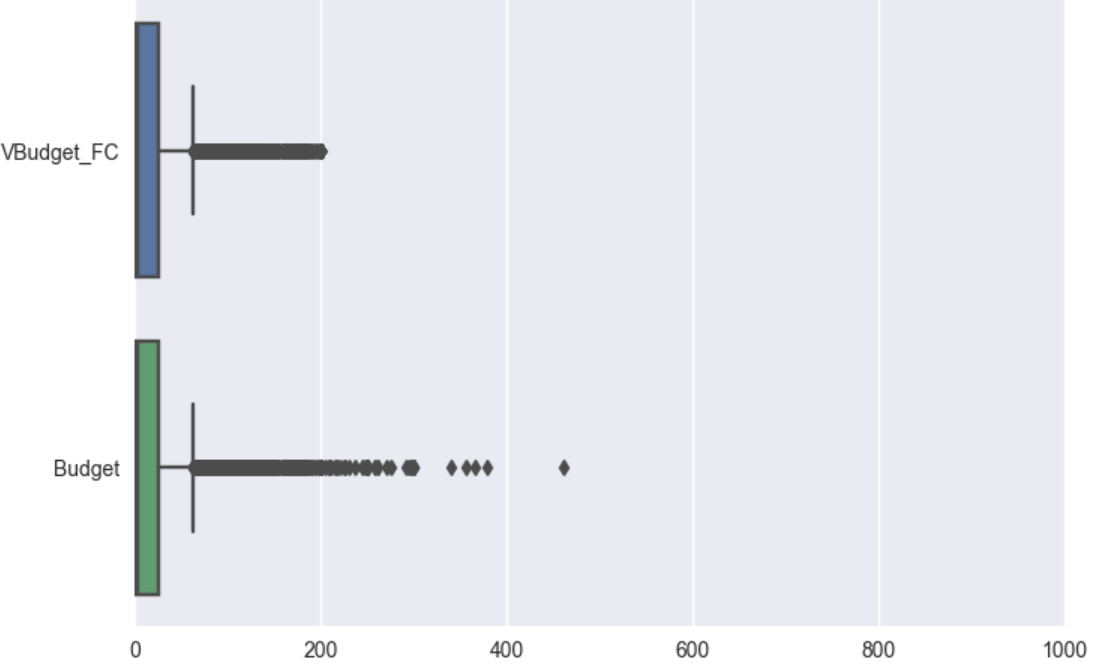
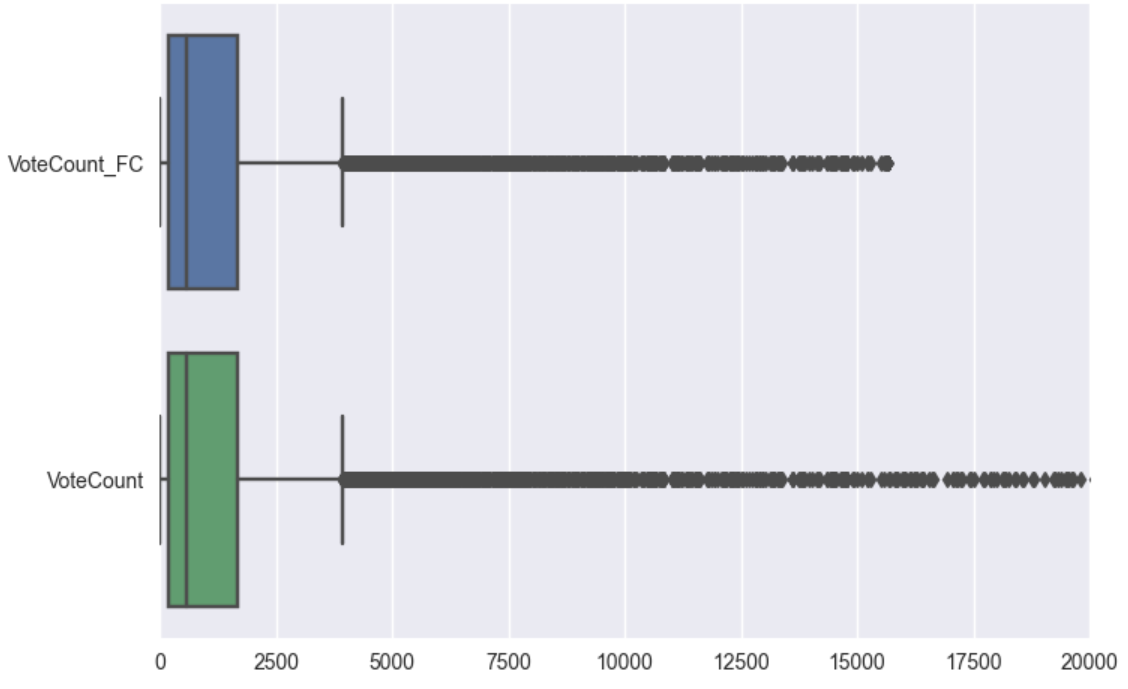
# **Step 7**

## Outliers Handling

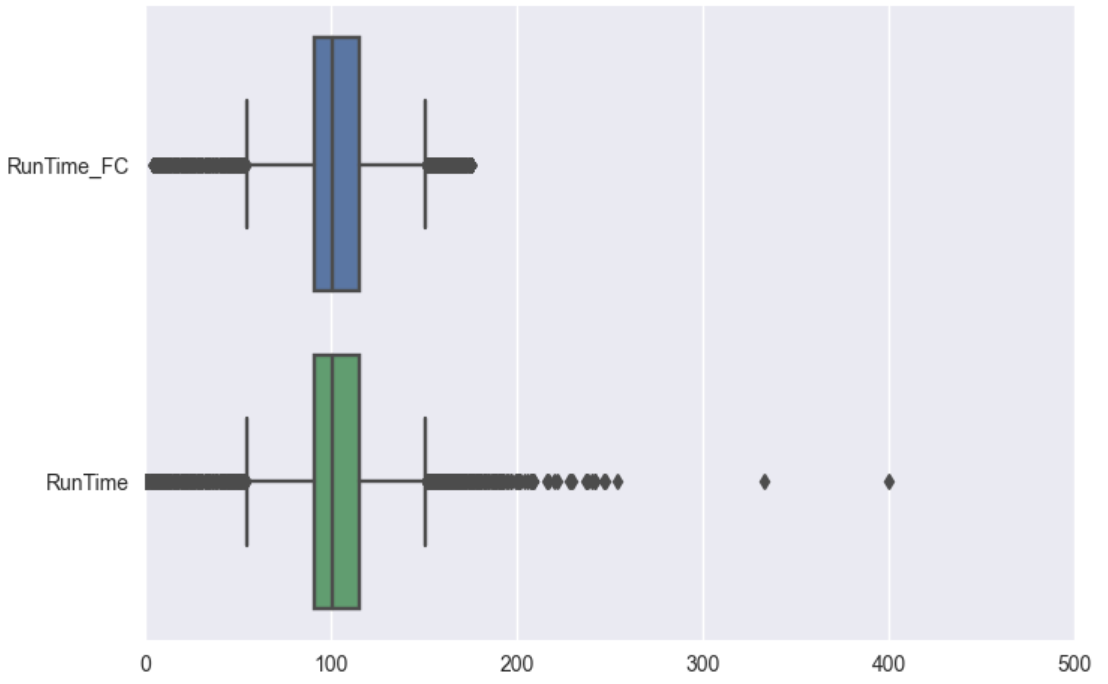
### Method 1: Quantile-based Flooring and Capping

* We will apply quantile based flooring and capping to popularity,vote average,votecount,budget,runtime,revenue

****

****

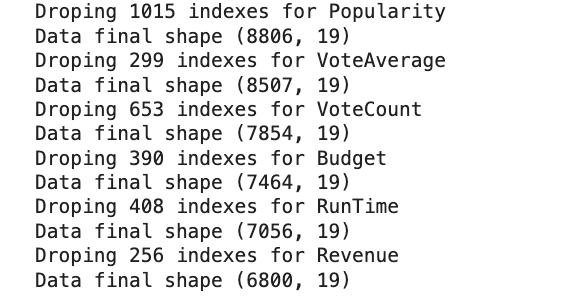
**Figure:12**

****

**Figure:13**

### Method 2: Trimming

We trimmed popularity,vote average,votecount,budget,runtime,revenue columns.



### 

### 

### 

### 

### 

### 

### 

### 

### 

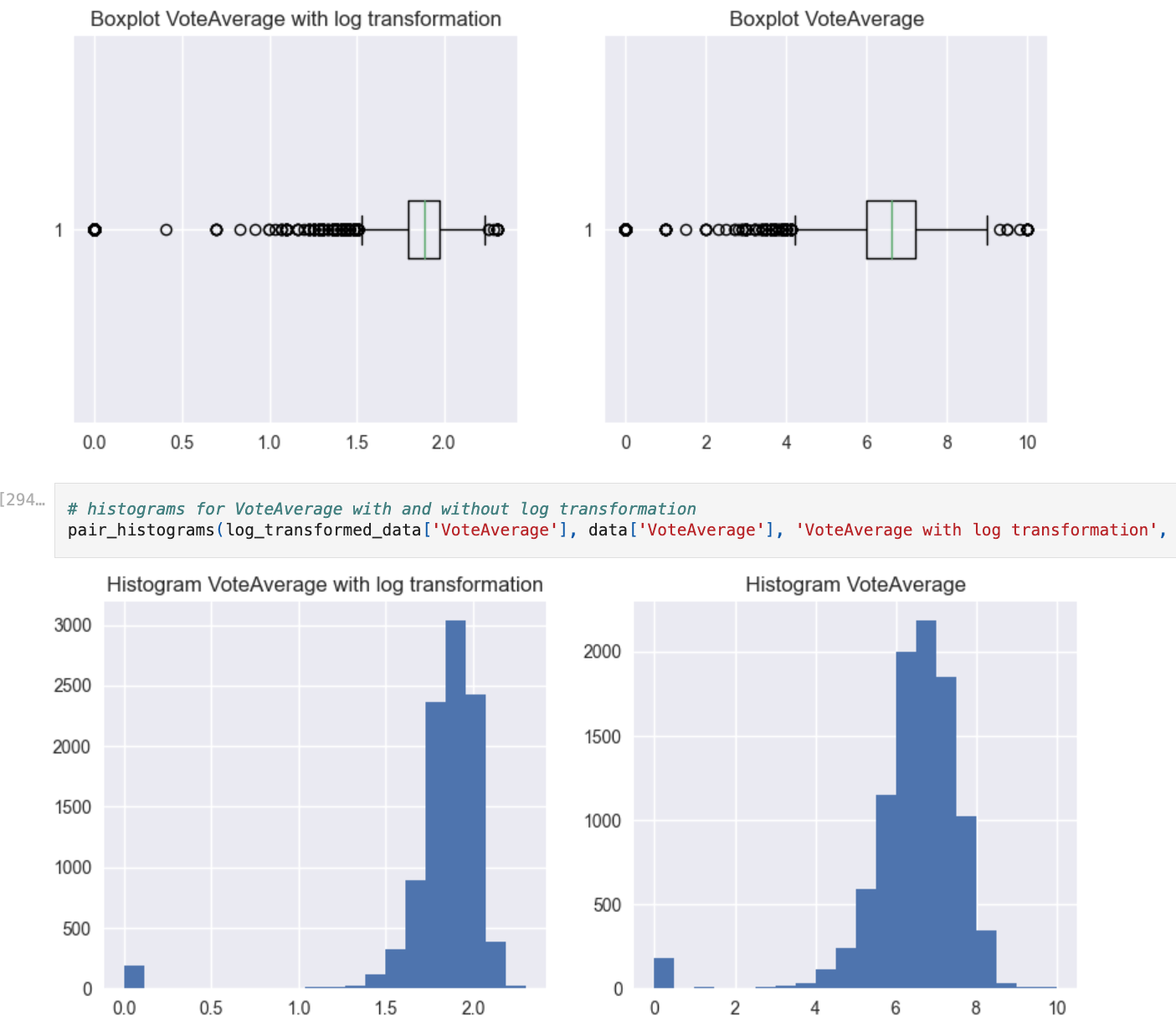
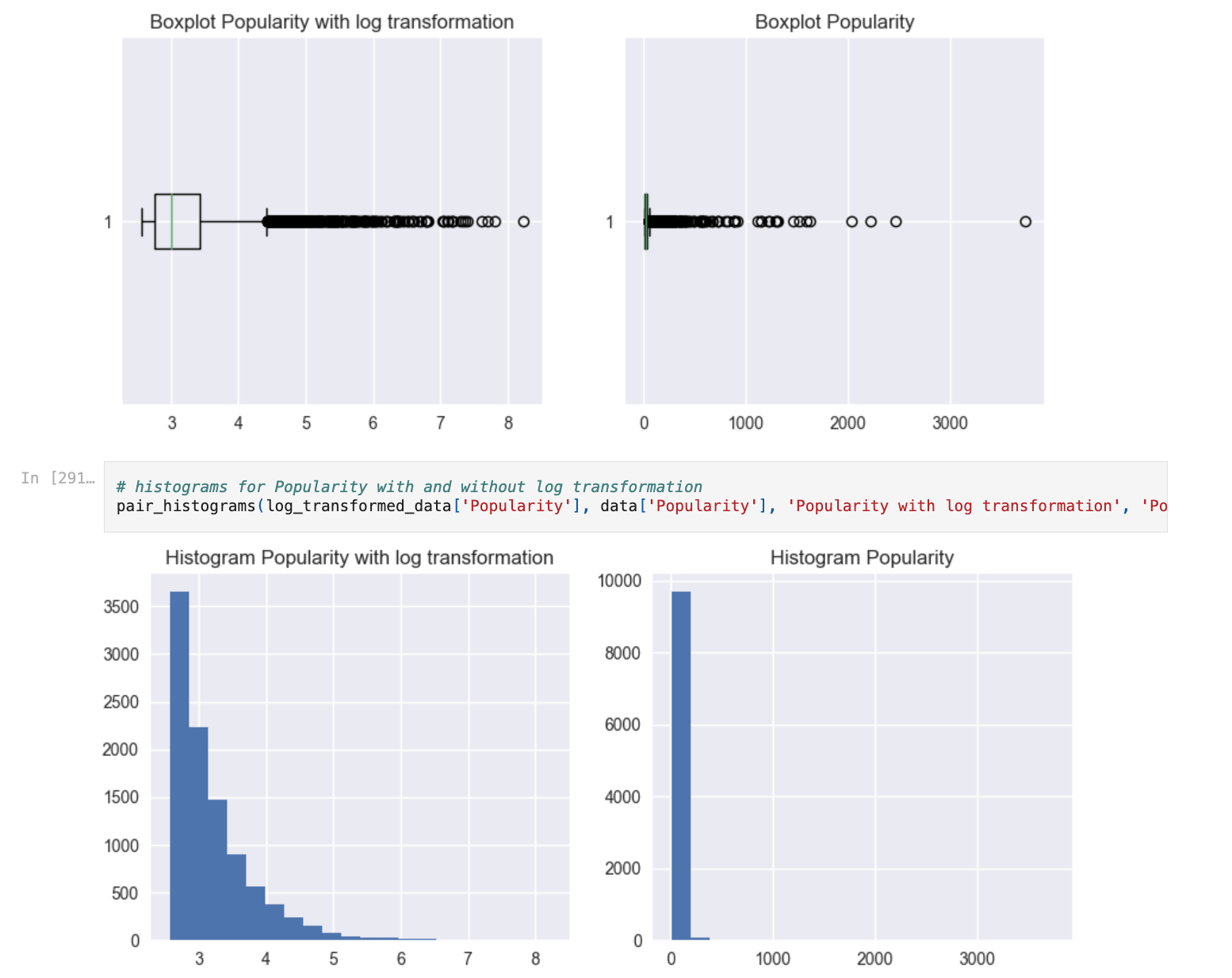
### 

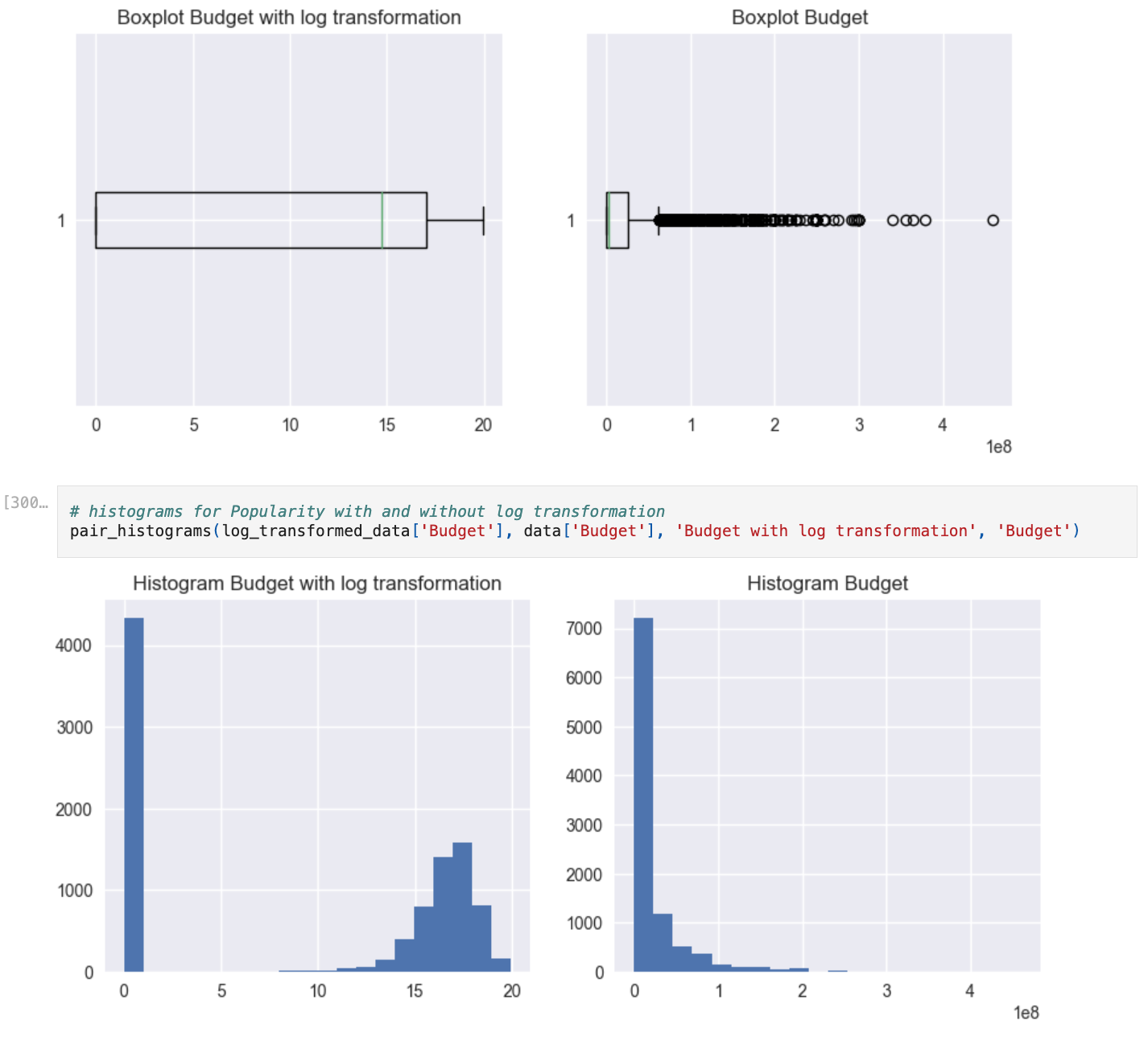
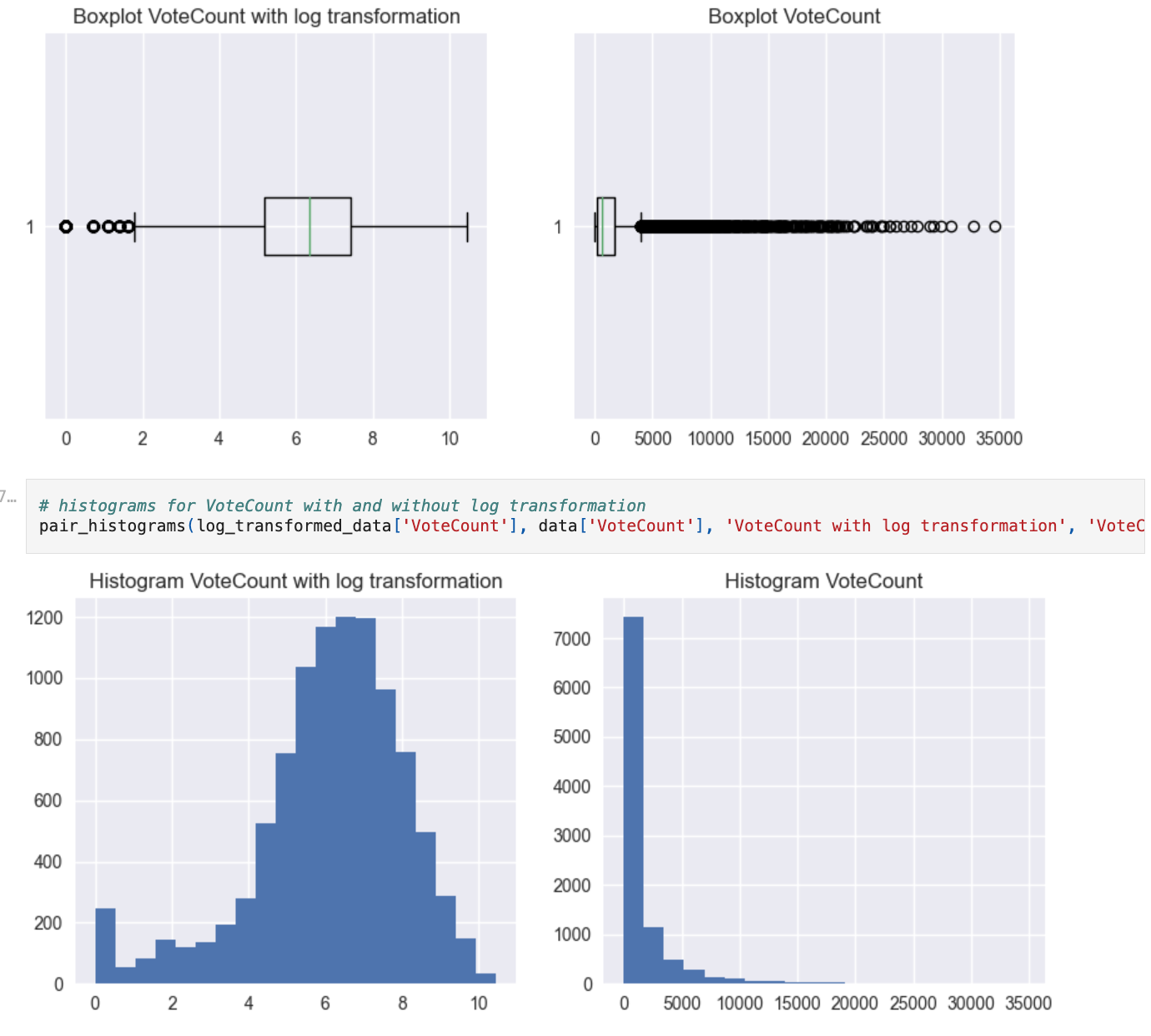
### 

### 

### Method 3: Log Transformation

We also applied Log Transformation on all of the numerical columns.





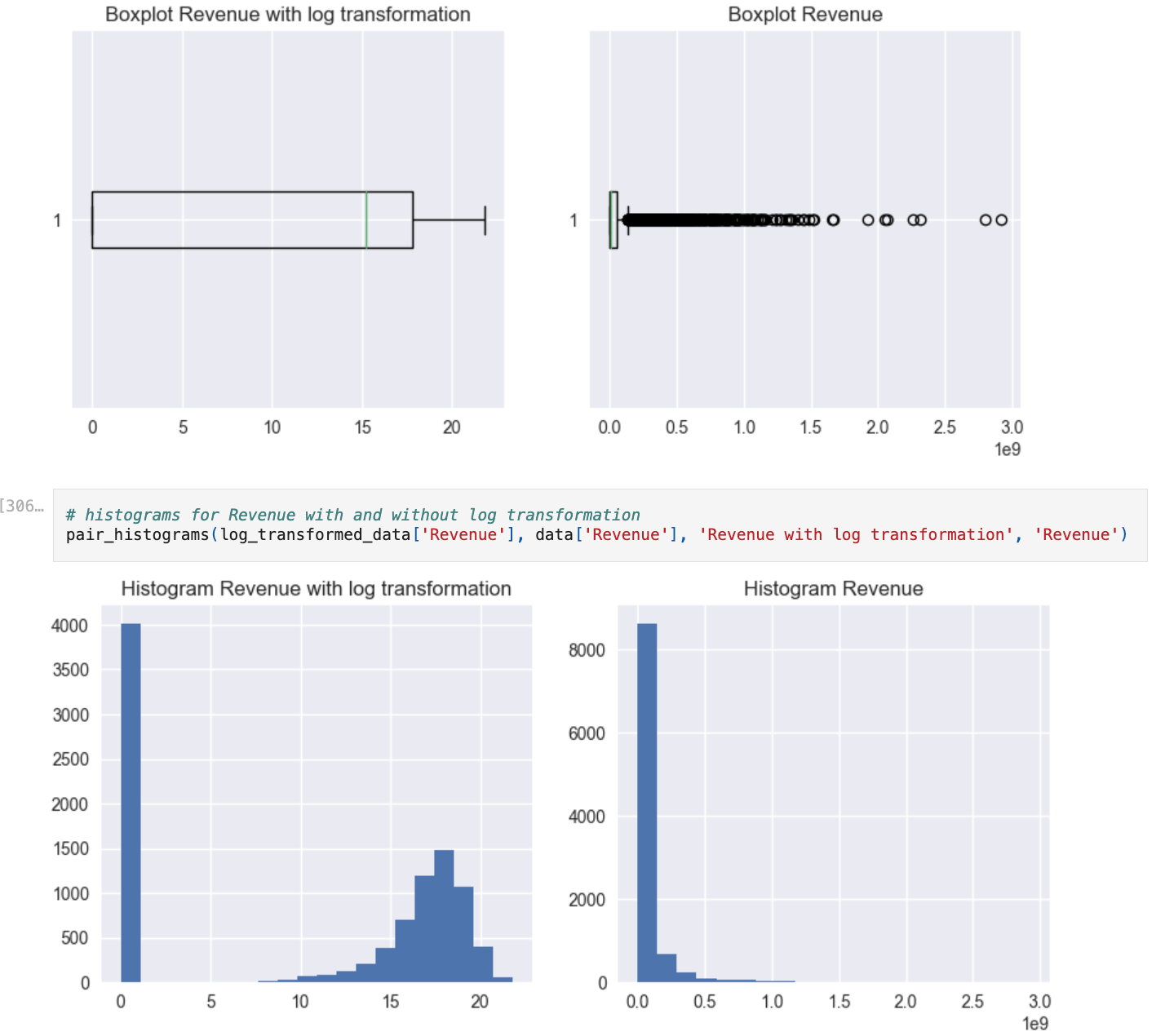
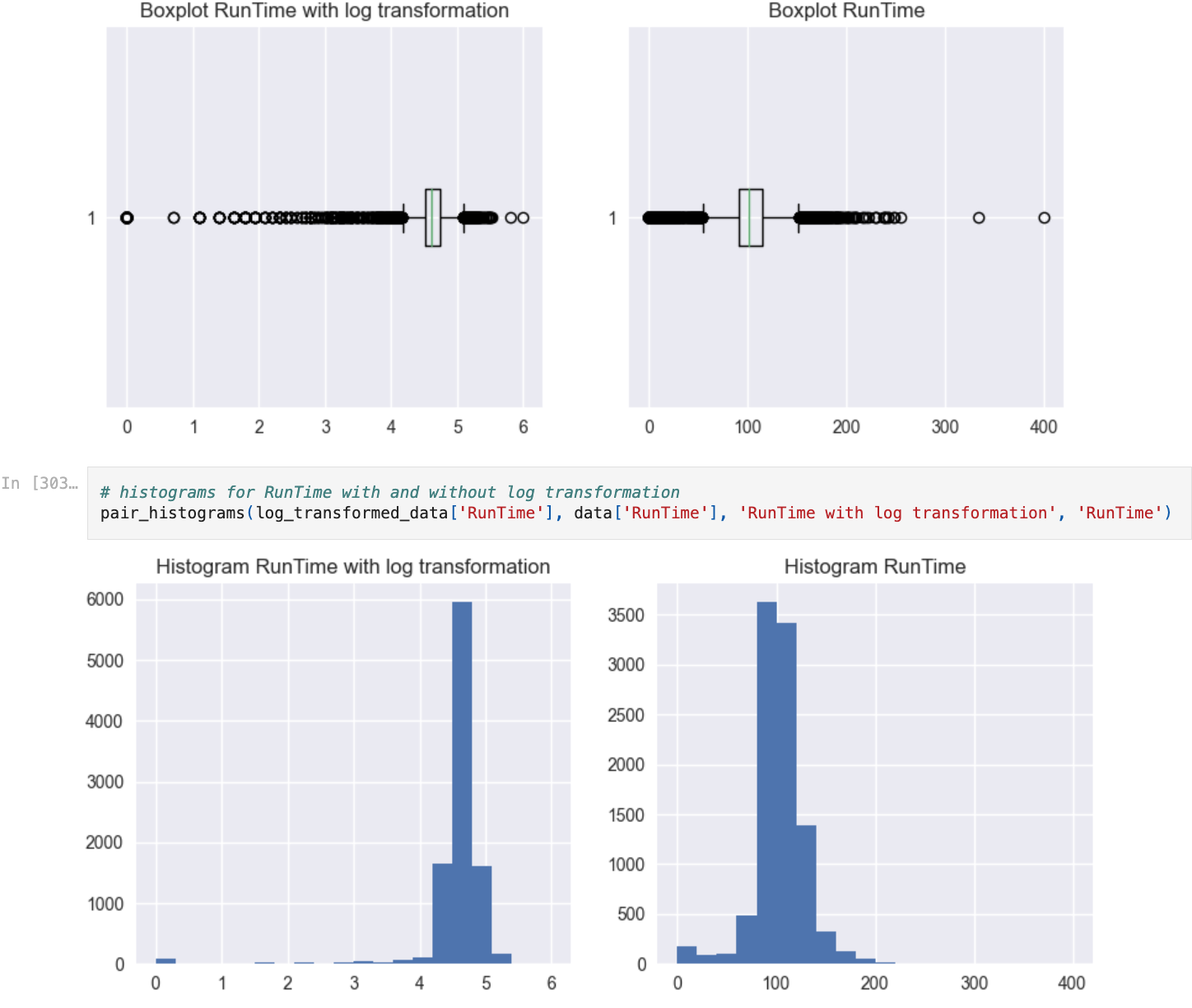


Figure:14

After trimming and doing log-transformation our data is ready for applying machine learning

**Using boxplot to final confirm all distributions**

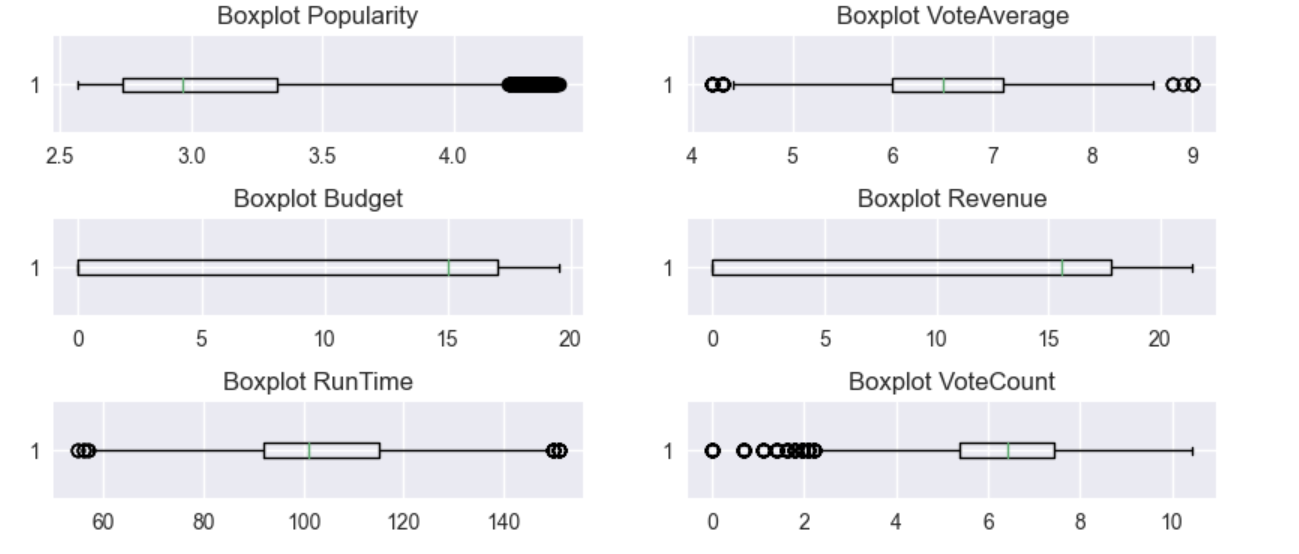


Figure:15

Quantile-based Flooring and Capping:

* Overall, it showed an improvement in skewness. However, for all the columns, we still got a significant presence of outliers.

Trimming:

* This method is not an option as a unique method to fix outliers due to a significant data loss. After trimming, the number of rows was reduced from 9979 to just 6796 (32%). Additionally, in terms of improving outliers, although it reduced the skewness for some columns, it was more effective for 'VoteAverage' and 'RunTime' (both have the most symmetric distributions). The other columns still have a significant presence of outliers.

Log Transformation

* This method presents a small change for those columns that already have a symmetrical distribution, such as RunTime and VoteAverage. On the other hand, for those skewed distributions, this method allowed a more symmetrical one, as seen in Popularity, VoteCount, Budget, and Revenue.
* This transformation successfully removed outliers for 'Budget', 'Revenue' and 'VoteCount'. Additionally, it improved the distributions for Popularity and VoteCount.
* Considering the results applying each of the methods for all the columns, we decided to remove the outliers as follows:
* 'VoteAverage' and 'RunTime' -> Trimming
* 'Budget', 'Revenue' and VoteCount-> Log Transformation
* 'Popularity' -> Log Transformation + Trimming
* Using the above strategy, we were able to remove most of the outliers and improve skewness for the columns.
* Finally, we stored the dataset as movies\_data.csv

# **Results**

# 

## Unsupervised Learning

We decided to used two methods:

* Latent Dirichlet Allocation, Topic modeling for textual data
* K-means clustering

**1.Latent Dirichlet Allocation, Topic modeling for textual data**

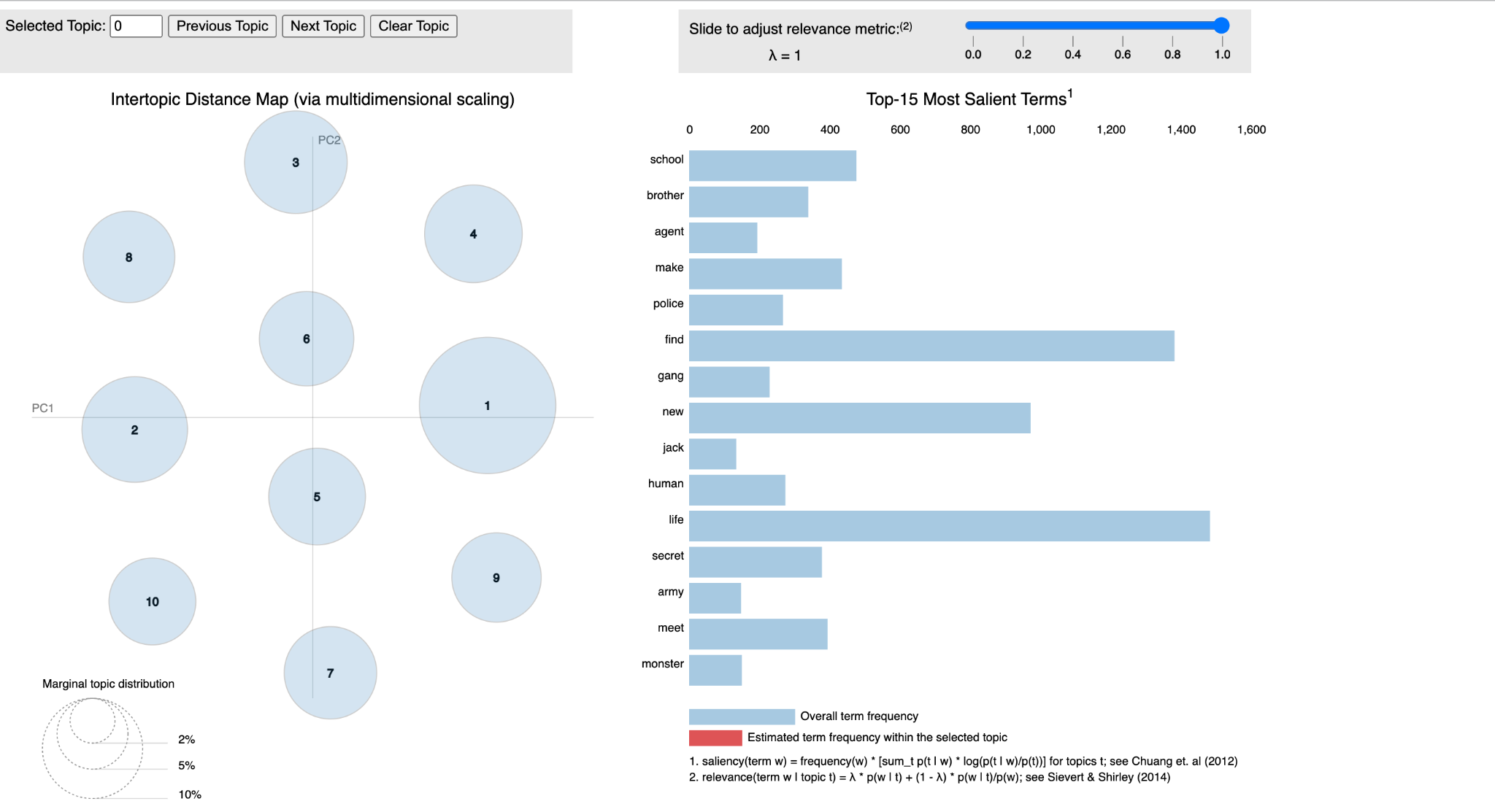


Figure:16

We used Latent Dirichlet Allocation, a model used for topic modeling in NLP. This particular model helps us discover the abstract "topics" in the documents. To train the LDA model, we defined the number of topics as 10. Additionally, we used the TfidfVectorizer as a vectorizer; we specified as 10 the number of passes through the corpus. Finally, we set the number of threads for paralyzation (workers) as 2.

Additionally, to read the visualizations, we had to select the relevancy metric to use (lambda at the top right slider)

* lambda = 1, Sorts the words based on the frequency in the topic.
* lambda = 0, Ranks the words based on their uniqueness.
* Selection: As we want to understand the differences between the topics, we want to balance those unique words with more general words. Therefore, we selected lambda = 0.5 and reduced it to lambda=0 for some cases.
* We created the topics for the Overview to represent the movie's content.
* Based on the dynamic visualizations built for the topics and the different lambda configurations, we can conclude:

The topics are not close or interconnected. Furthermore, Topic 1 has the biggest circle, which means there are more words associated with this topic compared to other topics. However, the circle size between the other topics is similar.

**Topics:**

* Topic 1: With lambda=0, we see words that can be used mostly in scary movies, such as ash, evil, dimension (other dimensions?), and treacherous.
* Topic 2: We can see terms that lead us to consider this topic related to drama, words like planet, brother, and angeles.
* Topic 3: We could see words like monster, mysterious, ruby, russian, human, father, quentin. This topic seems related to action-drama type of movies.
* Topic 4: This topic resembles war, with words such as war, jung, royal, american, border, halt, cartel, and stitch.
* Topic 5: Seems related to drama in the context of sports, with words such as football, roman, prison, mark, campus, killer.
* Topic 6: This topic is about teenage dramas, with words such as teacher, school, popular, one, gym, and meet.
* Topic 7: This category covers drama in a family context, with words such as family, grace, finding, parent, detective, neighbor, and country.
* Topic 8: This topic seems to cover teenage drama in a science fiction context with words such as school, high, tunnel, station, robot, drug, and ship.
* Topic 9: with words such as vill, take, and christmas, this topic may be related to Christmas movies.
* Topic 10: This topic is related to policy and crime, with words such as agent, gang, driver, thief, robbery, and police.

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### Clustering K-means

In K-means clustering we used Elbow method to decide numbers of clusters

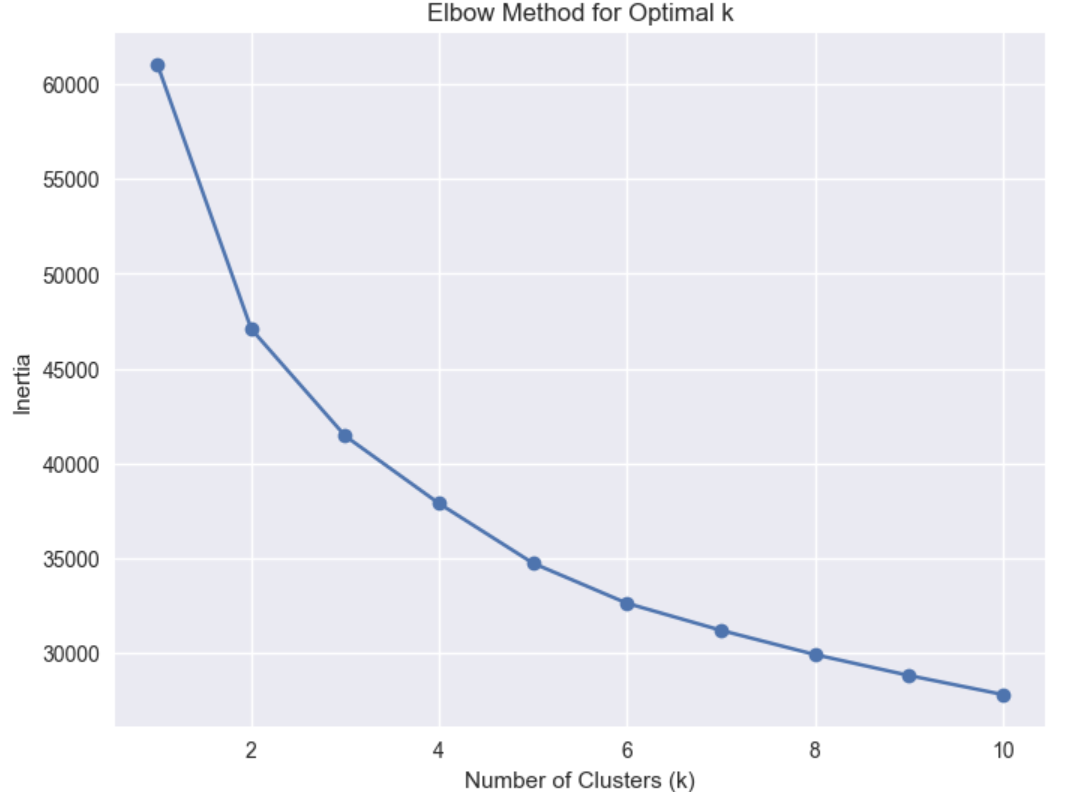


Figure :17

Using the Elbow method, we found a proper k for the clustering activity; in this case, a k=6 would be adequate.

**Clusters Analysis**

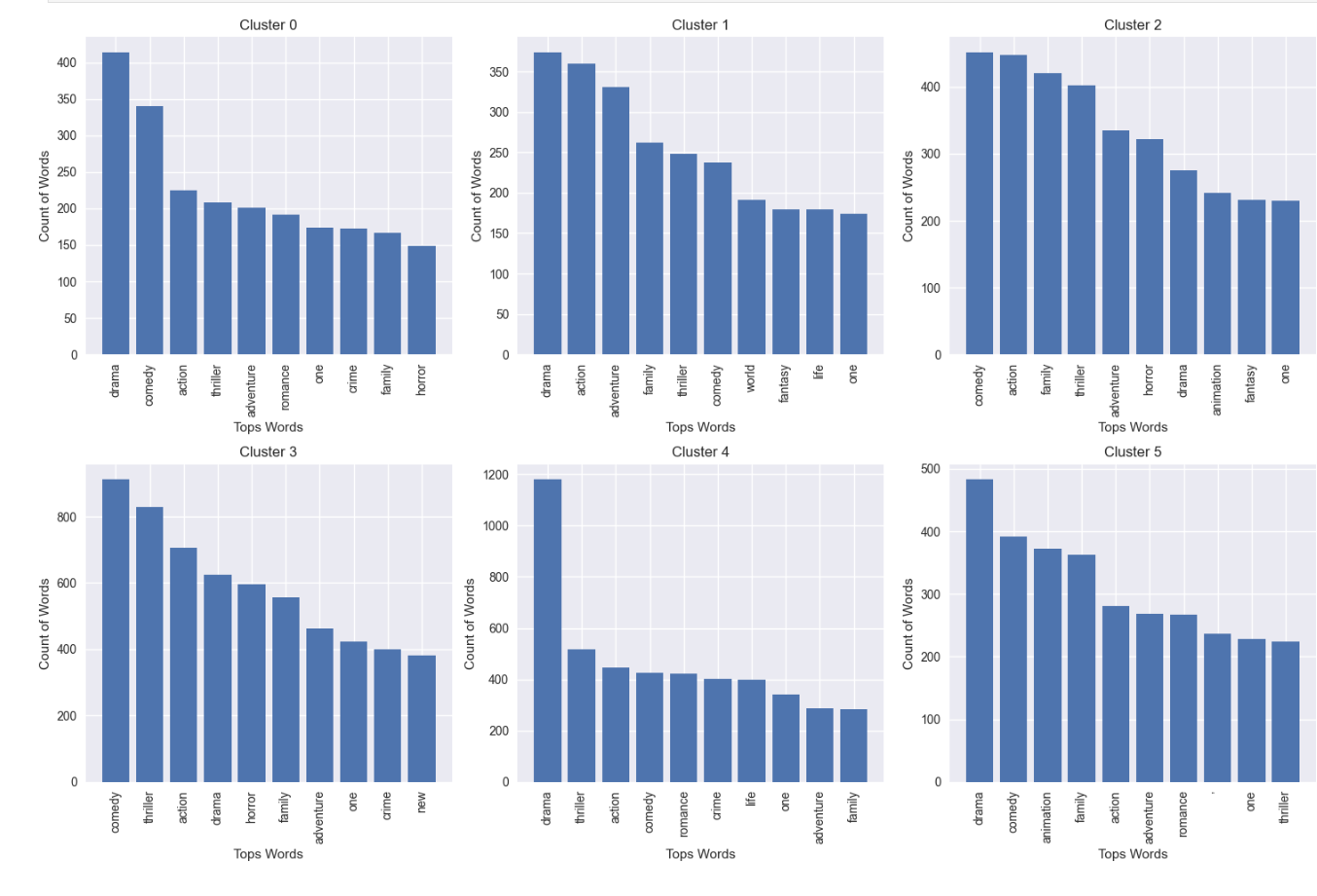
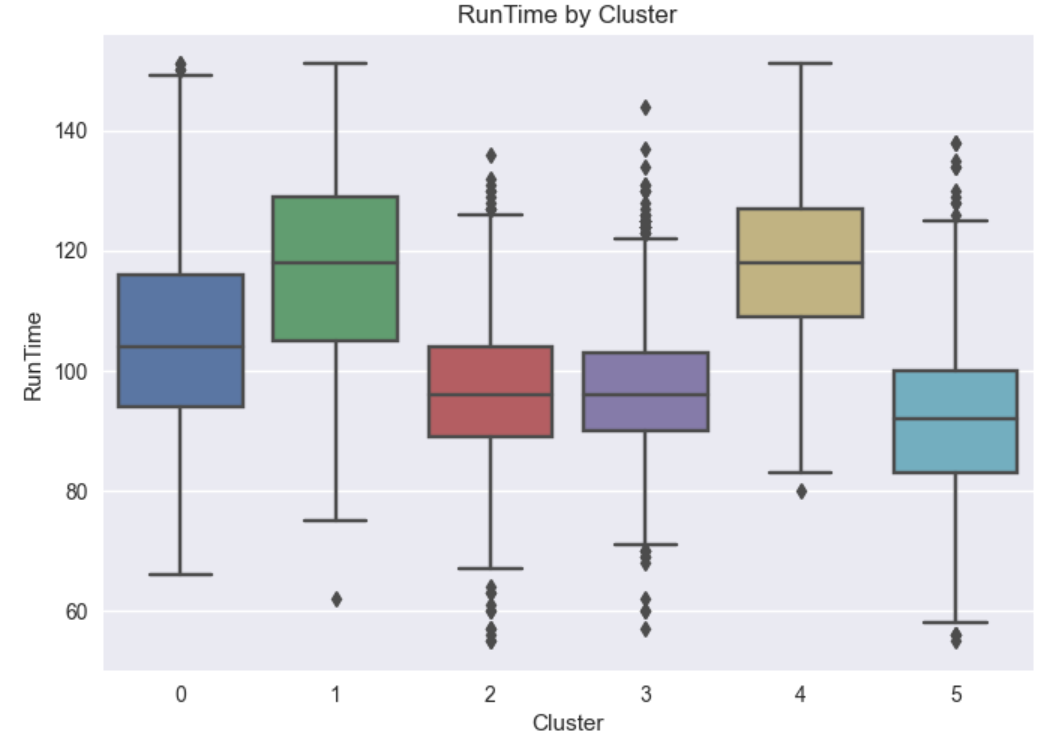
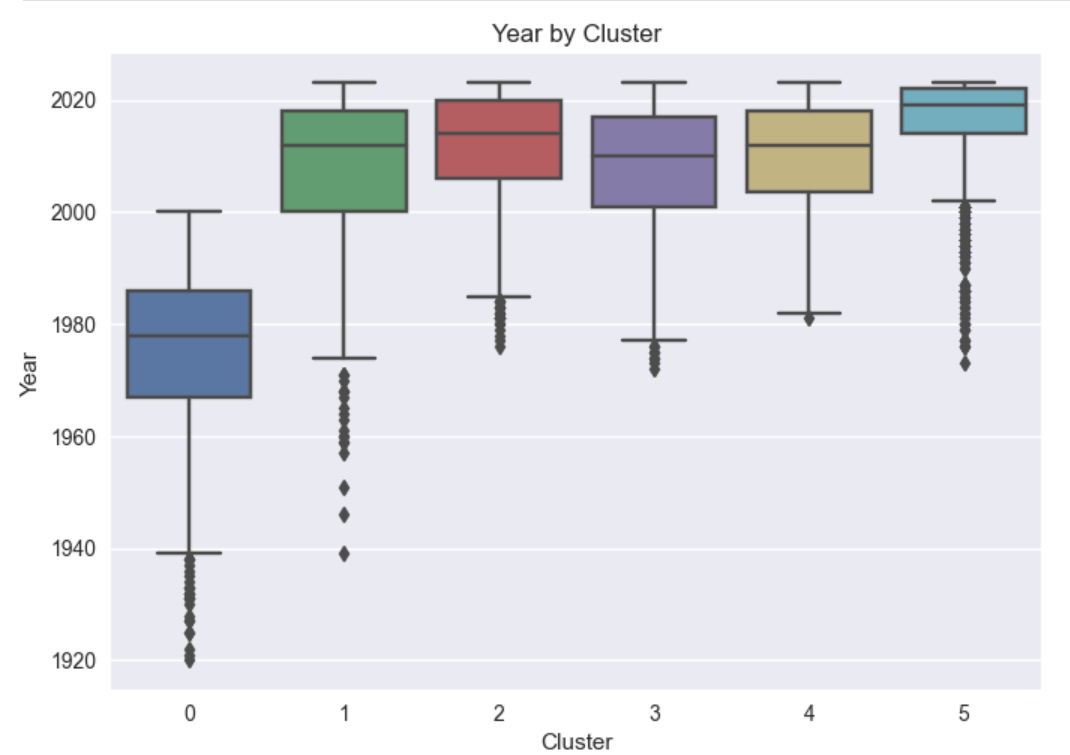
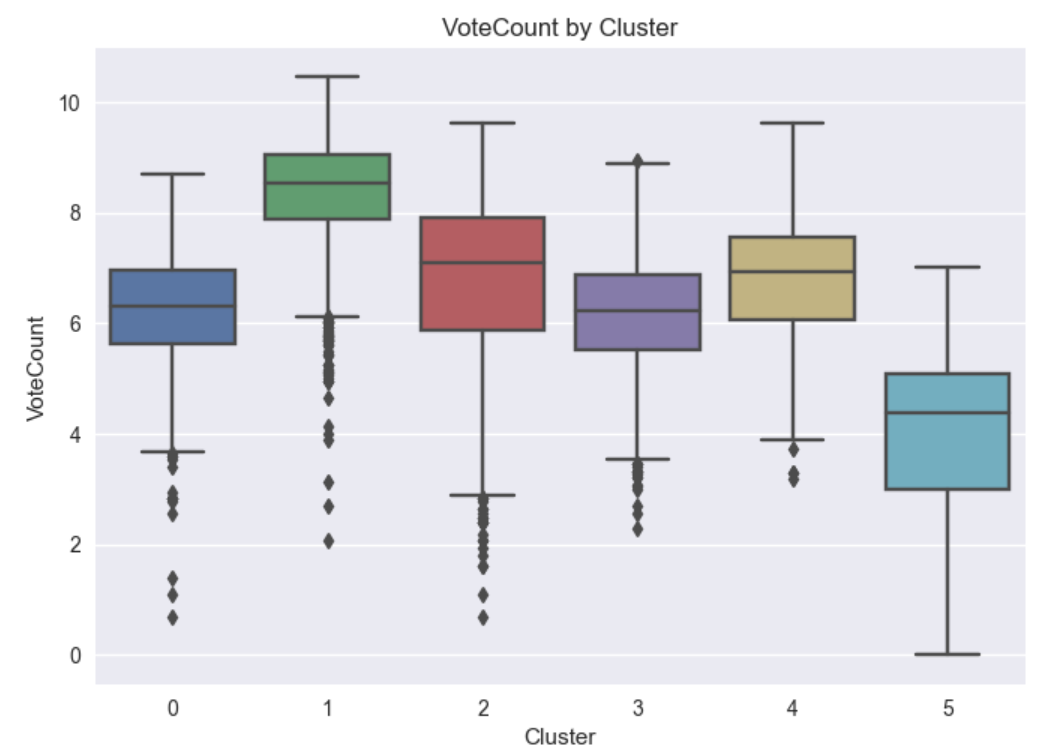
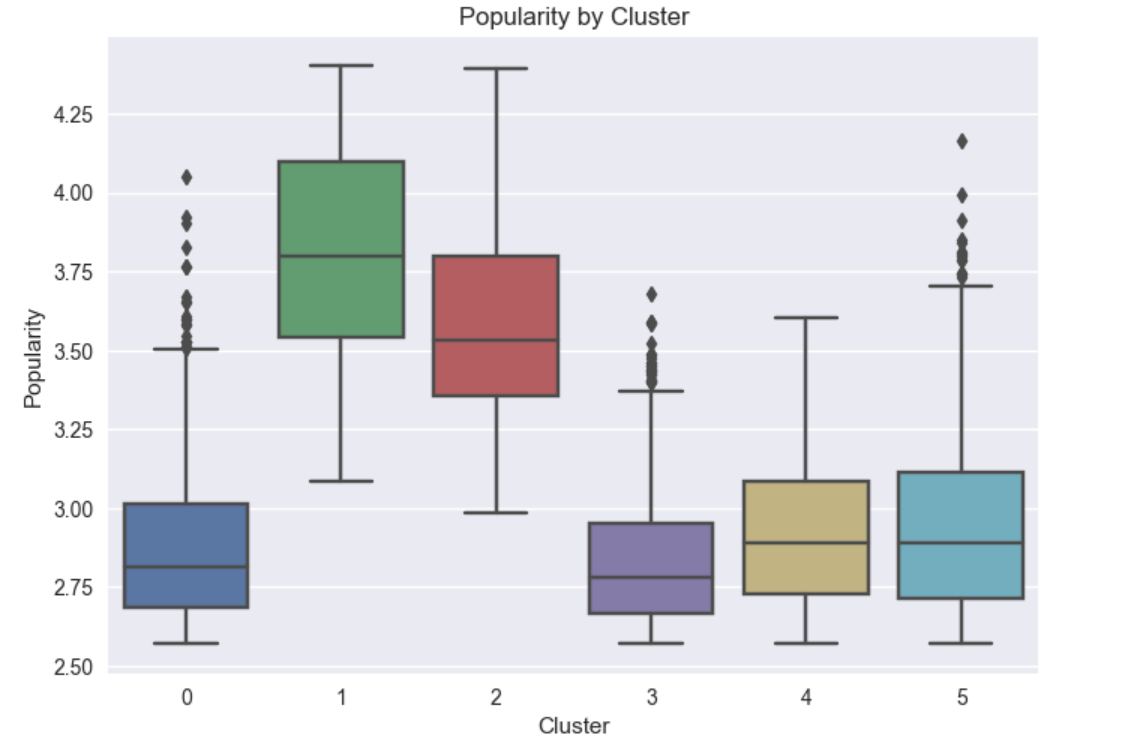
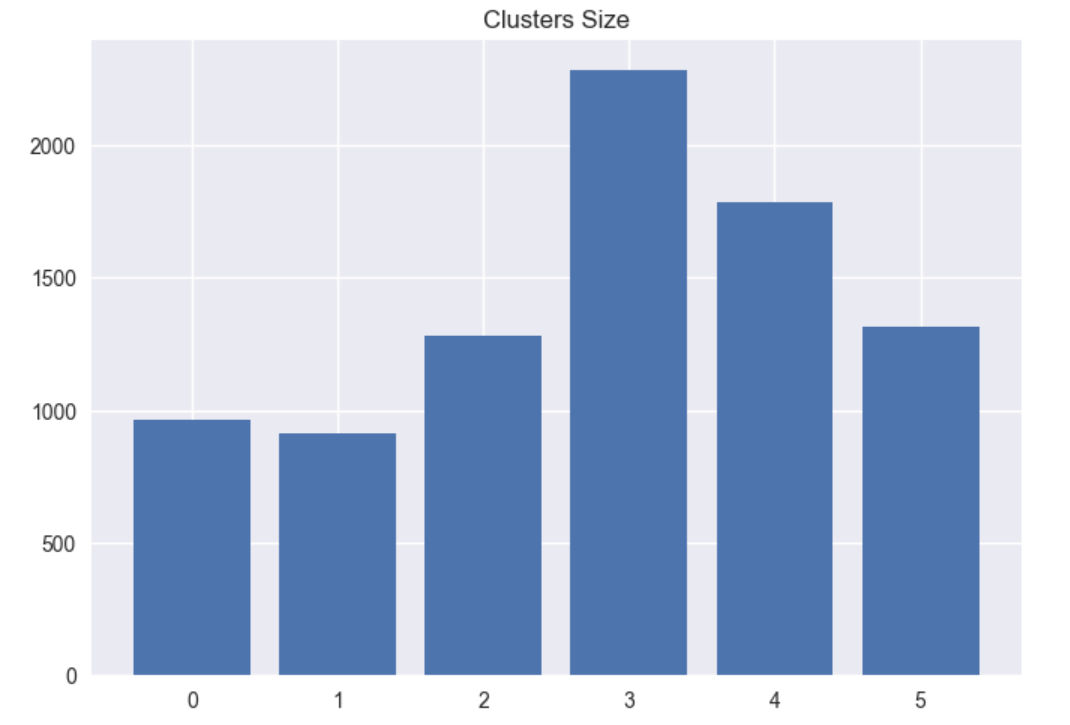
****

 Figure :18

# **Conclusions**

We aimed to use the TMDB dataset to understand successful movies' top characteristics.

* Based on our word cloud analysis (Figures 1, 2, and 3), movies tend to include primarily positive and neutral wording in their overview, tagline, and title. For instance, words like love, friend, family, and life can be seen at the top of frequency. However, in the same graphs, we still see some negative words, although in less frequency, such as die, fear, and dead. Additionally, we did not see a huge difference when comparing these results with those for movies with high popularity (Figures 1A, 2A, and 3A).
* Additionally, based on word frequency for textual columns, the most popular genres are drama, comedy, thriller, action, adventure, and romance (Figure 6).
* Furthermore, the most frequent combinations (n-grams) of words we found for the textual data (Figure 8, Figure 9) are New York, High School, One Day, Young Woman, New York City, World War II, Must Find Way, and Base True Story.
* Even though plotting Budget and Revenue vs Popularity shows a slight positive correlation between them (Figure 10), this correlation is not evident in the correlation matrix (Figure 12). Considering that the numerical columns had many outliers when the plots were created, it would be required to repeat the plots after removing outliers (step 7).
* LDA topic modeling successfully created different groups of movies based on textual data (Ida\_visualization\_overview.html). It gave us some words that are distinctive for various groups, which shows us the potential to create a new movie's taxonomy different from the traditional genres. On the other hand, k-means gave us clusters that were highly influenced by the genres.

# **Future Work**

* It would be essential to repeat the visualizations after removing outliers to have a solid understanding of the relation between the features.
* We could train a supervised machine learning model using popularity as the target variable. After having a model that performs well with the data, we could proceed with the feature importance analysis to conclude on those variables that influence the most the popularity results.
* As we discovered in this work the potential to group the movies based on textual features, we could add the groups to a new categorical column for training a supervised ML model.

# 

# References

## Datasets and Reports Download

* Please, download the datasets, pandas profiling report and lda dynamic visualization from this link: <https://mylambton-my.sharepoint.com/:f:/g/personal/c0891136_mylambton_ca/EsoIDptwI6VGgu5woWRD_xsBX2Z_tZhQbf7xBBFL0ZhLsw?email=Vahid.Hadavi%40cestarcollege.com&e=9QJx0n>
* The data used in this report was pulled from TMDB ([https://www.themoviedb.org](https://www.themoviedb.org/)).
* To be able to use the API, we created a user and sent a GET request to the authentications end
* point (<https://api.themoviedb.org/3/authentication>)
* we conducted GET requests to the endpoint (<https://api.themoviedb.org/3/discover/movie>) to get as many records as possible