Personalize Movie Recommendation System

Data Science Workflow Project Final Writeup

Sakib Hosen Himel [ s91038@bht-berlin.de]

1st June 2023

1. **Abstract**

Recommender systems are algorithms aimed at suggesting relevant items to users (items being movies to watch, text to read, products to buy or anything else depending on industries). I used machine learning to build a personalized content-based recommendation system based on movies’ different content information and did some advance popularity-based filtering using weighted score. Different people have different tastes in movies, and this is not reflected in a single score that we see when we Google a movie. My movie recommendation system helps users instantly discover top five similar taste of movies based on to their liking, regardless of how distinct their tastes may be.

Current recommender systems generally fall into two categories: content-based filtering and collaborative filtering. I experiment with content-based approaches in my project. For content-based filtering, I take movie features such as genres, actors, directors, and movie keywords as inputs and use TF-IDF, doc2vec, and Count Vectorizer to convert features into vectors with cosine similarity methods to calculate the pairwise similarity between movies. From the cast, crew, and keywords features, I need to extract the three most important actors, the director and the keywords associated with that movie. I found that in the majority doc2vec performs better than TF-IDF and Count Vectorizer in terms of top 5 recommend movies average similarity score.

In terms of in-depth key findings on the financial aspects of movie production, revenue generation, and audience reception. It highlights that producing a movie cost an average of $38 million, with budgets mostly exceeding $20 million. Despite most movies having budgets below $5 million, $20 million is the most frequent budget. The revenue generated by movies averages around $122 million, but the majority only earn approximately $8 million. This indicates a high risk in movie production, particularly when considering the distribution of budgets. Additionally, more than half of released movies have revenues below $60 million, with outliers like Avatar, Titanic, and Jurassic Park influencing the revenue figures. Profit, being crucial for producers, can exceed $80 million on average, but the mode value suggests most movies only generate around $2 million in profit. Half of the movies fall within the profit range of $27 million and -$166 million, emphasizing the risky nature of movie production. The introduction also touches on movie runtime, vote counts, popularity scores, and vote averages as factors to consider when evaluating a movie's success.

1. **Dataset and Features**

I use the TMDP dataset available on Kaggle **1**, covering over 4803 movies, and over 3.3 million ratings and no duplicates. The data is separated into two sets: the first set consists of a list of movies with their average ratings and features such as budget, revenue, cast, etc. Table 1 is the top 10 most rated and popular movies by their popularity and weighted score, calculated using the IMDB weighting **2**. I give 50% importance to each feature. We randomly split this dataset into an 80% training set and 20% test set for content-based filtering.

|  |  |  |  |
| --- | --- | --- | --- |
| **Movie Title** | **Normalized Weighted Score** | **Normalized Popularity** | **Score** |
| Interstellar  Minions  Guardians of the Galaxy  Deadpool  Mad Max: Fury Road  The Shawshank Redemption  The Dark Knight  Whiplash  The Godfather  Fight Club | 0.906433  0.460609  0.851866  0.725203  0.670957  1.000000  0.934986  0.916432  0.959622  0.953820 | 0.827162  1.000000  0.549462  0.587690  0.495989  0.156179  0.213941  0.219887  0.164074  0.167611 | 0.866798  0.730305  0.700664  0.656446  0.583473  0.578090  0.574464  0.568159  0.561848  0.560716 |

Table 1: Top 10 most popular movies by popularity and weighted score

The other dataset file contains the full credits for both the cast and crew, which also contains movie ID, and title. I merge these two-dataset based on movie title as they are containing unique values. We randomly split the set of user-movie rating pairs into an 80% training set and 20% test set. Techniques such as TF-IDF, doc2vec, and Count Vectorizer are used to analyze this dataset. At the end, I represent this data as an item features matrix where one dimension represents movies title, and the other dimension represents corresponding features. With this matrix, I compute a similarity score. I will be using the cosine similarity to calculate a numeric quantity that denotes the similarity between two movies.

1. **Data Preprocessing**

Data processing is a crucial step before model implementation. I performed in-depth feature engineering, including data streaming, lowercase conversion, function implementations, data cleaning, and data frame merging. These stages helped refine the data, optimize memory usage, standardize text, extract relevant features, ensure data cleanliness, and integrate multiple datasets. This comprehensive processing enhances the quality of the dataset for more accurate model implementation.

|  |  |
| --- | --- |
| **Data Preprocessing Pipeline** | |
| Merge Data Frame | Here, I merge two data-frame movies and credits based on movie title. |
| Basic Data Cleaning | Here, I did some basic cleaning like remove null duplicates if they are in datasets. |
| **Function [ def convert () ]** | Overall, this function extracting the 'name' values from dictionaries stored in the 'genres' and 'keywords' columns of a Data Frame named movies and storing them in separate lists. |
| **Function [ def convert\_cast ()]** | Overall, this extracting the 'name' values from the dictionaries stored in the 'cast' column of a Data Frame named movies. It first extracts the 'name' values from the first three dictionaries using the convert cast function and then further modifies the 'cast' column to keep only the first three names for each entry. |
| **Function [def convert\_crew ()]** | In summary, this function aims to extract the names of directors from a column named 'crew' and store them in the same column. |
| **Function [ def collapse ()]** | Overall, function designed to remove spaces from the elements in the 'cast', 'genres', and 'keywords' columns of the 'movies' Data Frame, effectively collapsing the spaces in each element. |
| **Make New Tags Columns** | Here I create a new columns name tags which contain documents from cast, crew, keyword, and genres columns. |
| **Convert Tags into String** | Here, I convert all tags columns information into strings. |
| **Convert Tags into Lower Case** | Here, I convert all tags columns information into lower case. |
| **Function [ def stem () ]** | Stemming: Stemming is a process that stems or removes the last few characters from a word, often leading to incorrect meanings and spelling. For instance, stemming the word ‘Caring‘ will return ‘Car‘ |

. Figure 2: Data Preprocessed Pipeline

1. **Method**

My dataset does not contain explicit user identifiers, it can pose a challenge when implementing traditional collaborative filtering techniques. Collaborative filtering relies on user-item interactions or preferences to establish relationships and make recommendations. Without user IDs information, the standard collaborative filtering approaches cannot be directly applied. Content-based recommendation systems focus on the characteristics or attributes of items to make recommendations to users. Instead of relying on user preferences or behavior, these systems analyze the content or features of items to identify similarities and make recommendations. Content-based recommendation systems have some advantages, such as being able to provide personalized recommendations even for users with unique preferences. So, I experiment with content-based approaches in my project.

***4.1 CountVectorizer***

CountVectorizer is a text feature extraction technique used to convert a collection of text documents into a numerical representation suitable for machine learning algorithms. In the project, CountVectorizer is utilized to process textual tags or attributes associated with movies. The vectorization process involves transforming the training data, train['tags'], into a matrix of token counts using the fit\_transform method. The resulting vectorized representation, stored in the vector variable, captures the frequency of terms or n-grams present in the movie tags.

Cosine similarity scores are then computed using vectorized representation. I calculate the cosine similarities between movies based on movie features vectors. I created one similarity matrix –using movie tagline, including cast, crew, and keywords. Cosine similarity calculates a numeric quantity that denotes the similarity between two movies. I use the cosine similarity score since it is independent of magnitude and is relatively easy and fast to calculate. Mathematically, it is defined as follows: Mathematically, it is defined as follows:

***4.2 TF-IDF***

TF-IDF (Term Frequency-Inverse Document Frequency) is a widely used text feature extraction technique that assigns weights to terms or words based on their importance in a document corpus. In the project, TF-IDF is applied to process textual tags associated with movies.

The TF-IDFVectorizer is configured with specific parameters, such as the maximum number of features, stop words, sublinear term frequency scaling, IDF usage, IDF smoothing, and L2 normalization. These parameters control the generation of a feature matrix where each row represents a document, and each column corresponds to a unique term in the corpus. The TF-IDF vectorization process involves transforming the training data, train['tags'], into a matrix of TF-IDF values using the fit\_transform method. The resulting vectorized representation, stored in the train\_tfidf variable, captures the importance of terms in the movie tags within the corpus.

Cosine similarity scores are then computed based on the TF-IDF vectorized representation. Cosine similarity measures the similarity between two vectors by calculating the cosine of the angle between them. It provides a numerical value representing the similarity between movies based on their TF-IDF weighted tags.

The importance of a term t is positively correlated with the number of times it appears in document d, but the frequency of term t among all documents is inversely related to its ability to distinguish between documents. Thus, we calculate the frequency of word t in document d, weighted by the inverse of frequency of t in all documents:

where |D| is the length of the document, and |d : t ∈ d| is the number of documents where t appears.

***4.3 Doc2Vec***

Doc2Vec is an advanced technique for generating document embeddings, which capture the semantic meaning and context of documents. In the project, Doc2Vec is utilized to build a movie recommendation system based on textual tags associated with movies.

The process involves training a Doc2Vec model using the gensim.models.doc2vec module. The training data, represented as tagged documents, is fed into the model. Each document is composed of a list of words and assigned a unique tag. The model learns to generate fixed-length vectors that represent the semantic characteristics of the documents. Once the model is trained, document vectors are inferred for the training data using the infer\_vector method. These vectors capture the semantic meaning of the movie tags, enabling the system to understand the context and relatedness of different tags.

TF-IDF and CountVectorizer look for the frequency of the exact word in a document and could not pick up on synonyms or similar descriptions, so it produces very low similarity scores across all movies. Word2vec is based on a distributional hypothesis where words appear in the same context tend to have similar meanings. To take the context of a word into account, we use doc2vec[16], which is an extension of the word2vec model that added a document-unique feature vector representing the entire document.

Cosine similarity scores are then computed between the document vectors using the cosine\_similarity function from sklearn.metrics.pairwise. This similarity calculation quantifies the resemblance between movies based on their document embeddings.

This approach enhances the accuracy and effectiveness of the recommendation system by considering the deeper semantic relationships between movies.

1. **Results and Discussion**

For content-based filtering, I use the movies and features cosine similarity score with used models, CountVectorizer, TF-IDF and doc2vec to recommend top 5 similar taste movies (Formula (1)). We randomly split the set of movie features into an 80% training set and 20% test set.

Among the three models implemented for movie recommendation based on cosine similarity, the Doc2Vec algorithm showed slightly better performance overall in terms of similarity score.

The CountVectorizer model provided accurate movie recommendations by converting the textual data into a numerical representation. It effectively captured term frequencies but did not consider the importance of rare terms. The TF-IDF-based model improved upon this by incorporating inverse document frequency weighting, capturing the rarity of terms across the dataset. This led to more informative vector representations and more precise movie recommendations.

However, the Doc2Vec algorithm an extension of Word2Vec outperformed the other models slightly in terms of overall performance. It generated document embeddings that captured semantic information, allowing for a more nuanced understanding of movie tags. The embeddings enabled the calculation of cosine similarity scores that considered the semantic relationships between movies. As a result, the Doc2Vec-based model recommended movies that shared similar thematic or conceptual characteristics, leading to more personalized and contextually relevant recommendations.

While all three models were effective in movie recommendation based on cosine similarity, the Doc2Vec algorithm demonstrated a slight edge in performance. Its ability to capture semantic information and leverage the semantic relationships between movie tags contributed to its improved recommendation accuracy. Users relying on the Doc2Vec-based model could expect to receive movie recommendations that were more tailored to their preferences and interests.

Using these models, we could recommend movie names for the most similar movies based on the similarity score calculated from the movie tags column. Table 2 shows the top 5 most similar movies to Skyfall.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Similar Movies | Sim Score | Avg Sim Score | Similar Movies | Sim Score | Avg Sim Score |
| Ant Man | 0.653 | 0.609 | Ant Man | 0.653 | 0.615 |
| The Avengers | 0.645 | The Avengers | 0.645 |
| Captain America: The Winter Soldier | 0.626 | Captain America: The Winter Soldier | 0.626 |
| Thor | 0.527 | Thor | 0.602 |
| Fantastic Four | 0.548 | Fantastic Four | 0.548 |

Table 2: CountVectorizer and TF-IDF Model

|  |  |  |
| --- | --- | --- |
| Similar Movies | Sim Score | Avg Sim Score |
| X-Men: First Class | 0.919 | 0.882 |
| The Incredible Hulk | 0.881 |
| Superman | 0.872 |
| The Wolverine | 0.866 |
| Green Lantern | 0.866 |

Table 2: Doc2Vec Model

A picture containing line, screenshot, plot, rectangle

Description automatically generated

Figure 2: Performance Graph based on Average Similarity Scores of Recommender Models

1. **Key Findings with Visualization**
   1. ***Popularity Based Filtering using Weighted Average***

A picture containing text, screenshot, colorfulness, line

Description automatically generatedAs the name suggests Popularity based recommendation system works with the trend. It basically uses the items which are in trend right now. Weighted average is a calculation that considers the varying degrees of importance of the numbers in a data set. In calculating a weighted average, each number in the data set is multiplied by a predetermined weight before the final calculation is made. Here I used 70th percentile as our cutoff. In other words, for a movie to feature in the charts, it must have more votes than at least 70% of the movies in the list.

A picture containing screenshot, text, colorfulness, line

Description automatically generatedFigure 1: Top 10 movies based on user’s average votes.

Figure 2: Top 10 movies based on user’s popularity score.

A picture containing text, screenshot, colorfulness, line

Description automatically generatedFigure 3: Top 10 best rated & most popular movies based on score of popularity & weight average.

* 1. ***Correlation Matrix:***

A correlation matrix plot is a visual representation of the pairwise correlations between variables in a dataset. It uses a color-coded heatmap to display correlation coefficients, ranging from -1 to 1. This plot helps identify the strength and direction of relationships between variables. It aids in feature selection, identifying multicollinearity, and exploring patterns in the data.

A picture containing text, screenshot, square, rectangle

Description automatically generated

Figure 4: Correlation between all numeric features.

* 1. ***Pair Plot Distribution***

A picture containing text, line

Description automatically generatedHere, I implement seaborn pair plot on budget, revenue, profit, runtime, vote count, popularity, and vote average to see distribution among all numeric features.

Figure 5: Seaborn relationship graph between all numeric features of dataset.

From this graph we get various key findings of the data over the time, and I am going to discuss major key findings followings-

From budget vs revenue vs popularity vs runtime vs vote count vs vote average graph we get info bellows-

* Movies with higher budget seem to have a little bit higher popularity score, although a higher budget for a movie doesn't seem to have any effect on its vote average, I'd say that a movie with very high budget isn't always a good movie.
* It seems that a movie with a higher budget also has a higher revenue, so if you're willing to score the highest revenue for your movie, don't be afraid to take the risk.
* In addition to my friends said, a movie's budget doesn't affect its runtime, but it increases the vote count score, that means that the higher the budget is the more people have watched that movie.

From revenue vs popularity vs runtime vs vote count vs vote average graph we get info bellows-

* A movie's revenue can be a great scale for a good movie, as it correlates positively with both popularity and vote average, so make sure to watch all those movies with extremely high revenue.
* If you looked at the relation between movies' revenues and runtimes, you'll find that all the movies that scored more than 1B dollars have a runtime that are between 1.5 and 3.3 hours, so I guess that a movie with a little bit long runtime can help on increasing its revenue.

From profit vs popularity vs runtime vs vote count vs vote average graph we get info bellows-

* Like the revenue, a high movie profit can refer to a movie with high popularity and vote average scores and so a good movie to watch.
* Also, movies with runtime between 1.3 and 3.3 had profits that are more than 0.5B dollars.

From runtime vs popularity vs vote count vs vote average graph we get info bellows-

* Although the longer the movie is the higher its vote average is, but that doesn't mean always that it's a popular movie, so if you only care about the vote average not popularity, I recommend long movies for you, also I'd like to say that not all the short movies are having a low vote average, so if you're really hope to watch a great movie, don't take the runtime as an estimate for a great movie.

From vote count vs popularity vs vote average graph we get info bellows-

* Movies with vote count that's greater than 5.5K seems to have a high score of popularity and vote average, so watching a movie with a vote count greater than 5K is a very safe deal for you to take.
  1. ***Exploratory Data Analysis with Custom Function***

I created two separate functions named: def dist () and def tops\_in\_decades() to deep visualization of our data over the time period.

The dist function plots a histogram to visualize the distribution of a specified column in a DataFrame. It also adds vertical lines to indicate the mean, median, and mode of the column.

The function tops\_in\_decades generate a bar chart showing the highest value of a specified measure for each decade, along with the corresponding target value. It uses the measure and target parameters to select the columns from the DataFrame and creates a DataFrame DF to store the results.

***6.4.1 Budget Features***

From Figure 6, we get information- Producing a movie will cost you on average about 38 million dollars and mostly it will cost you more than 20 million dollars. Although most movies had budgets that are less than 5 million, but the most frequent movie budget was 20M, also there're some movies that had budgets that are more than 250M.

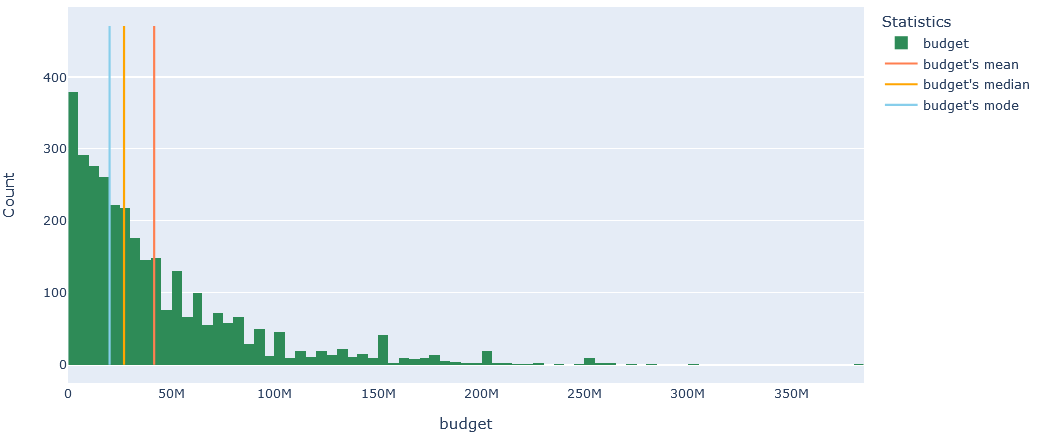


Figure 6:

From Figure 7, we get information- Pirates of the Caribbean has the highest budget till 2016.

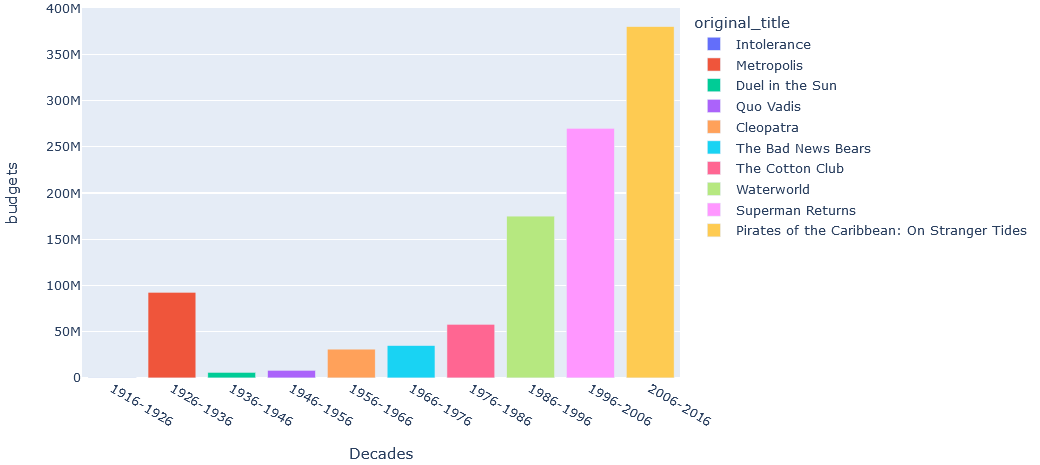


Figure 7:

***6.4.2 Revenue Features***

From Figure 8, we get information- Producing a movie will get you about 122M revenue on average, but mostly it'll get you only 8M dollars revenue, this kind of disturbing as if you remember the distribution of the movies' budgets, you'll figure out that it's a very risky thing to produce a movie, but don't worry as I'm still here to show you how to produce the most successful movie ever later in this report. Although the maximum movie revenue is about 2.75B dollars, but more than half of released movies' revenues is less than 60M dollars.

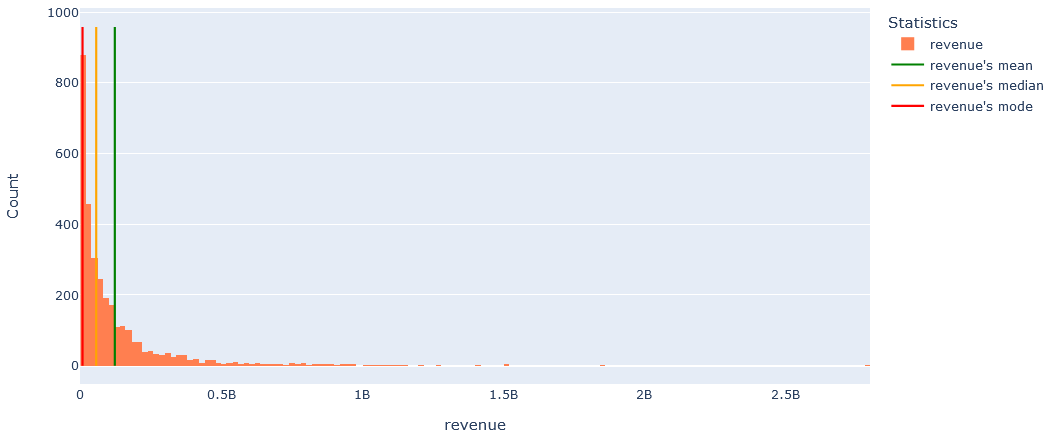


Figure 8:

From Figure 9, we get information- Avatar, Titanic, Jurassic Park. etc. are the movies with outlier revenues.

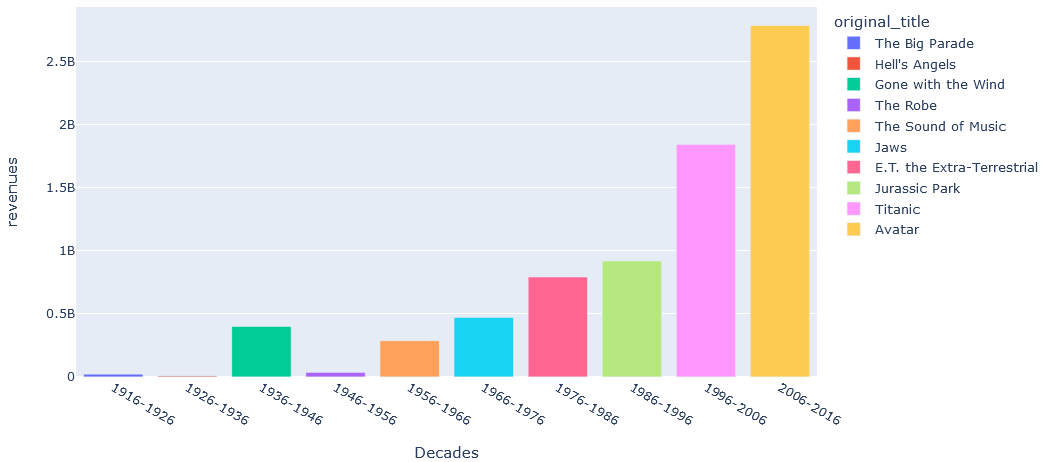


Figure 9:

***6.4.3 Profit Features***

From figure 10, we get information- I believe that Profit is the most important thing to care about as a producer, and if you took a look here on this distribution, you'll find that producing a movie can get a profit of more than 80M dollars, but I believe that you can easily notice the outliers in this data that's definitely affected that value, but if you looked at the mode value, you'll find that producing a movie will mostly get you only 2M dollars profit, again it's a very risky thing to produce a movie, especially when you notice that half of movie profits are between 27M and -166M dollars, and for more accurate statistics.

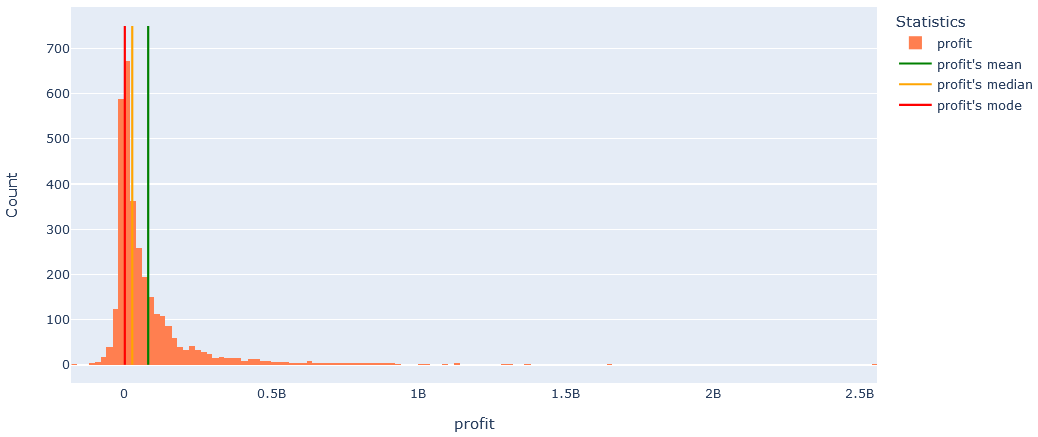


Figure 10:

From Figure 11, The most profitable movies are the same as the most revenue movies, this can mean that profit is highly affected by the revenue more than the budget.

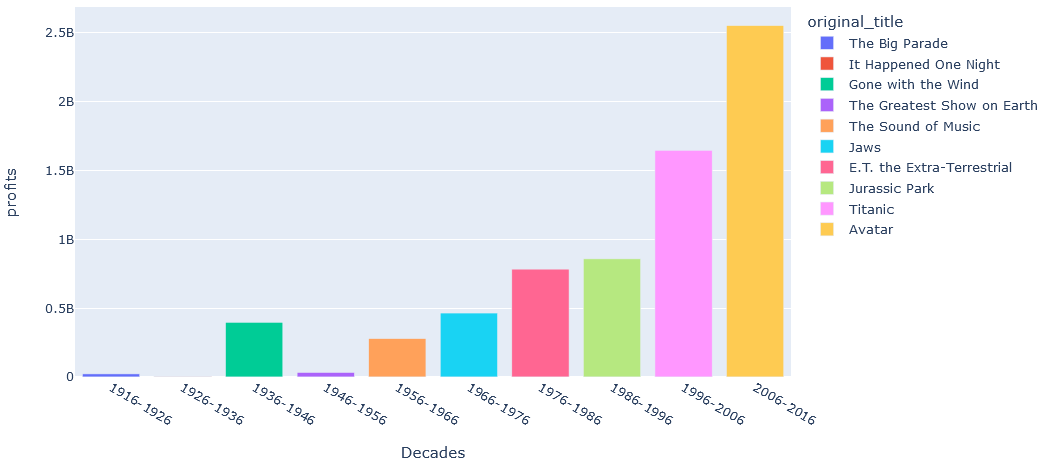


Figure 11:

***6.4.4 Vote Count Features***

From figure 12, we get information- Since the average vote count for a movie is 1000, I believe that a movie with a vote count that's greater than 1000 can be a very good movie to watch, as the more vote count a movie has, the more people watched that movie, and people don't gather around bad movies, right? Most movies have vote counts that are less than 250, and more than half of them have vote counts that are less than 500, so yeah, any vote count of a movie that is greater than 500 is a very satisfactory movie.

A picture containing text, screenshot, plot, line

Description automatically generated

Figure 12:

From Figure 13, You can watch the last 8 movies here as they're having vote counts that are much greater than 1000. Inception, Fight club, Pulp Fiction, Star Wars, The godfather. etc. were the top vote count movies in the last century, again I'm not surprised.

A picture containing text, screenshot, diagram, colorfulness

Description automatically generated

Figure 13:

***6.4.5 Vote Average Features***

From figure 14, we get information- Although the most frequent vote average is 6.5, but the average of vote averages is 6.3, also half of all movies have a vote average that are less than 6.3,so If you asked me, I'd say that a movie with a vote average that is greater than 6.5 is a very nice movie to watch.

A picture containing screenshot, diagram, plot, text

Description automatically generated

Figure 14:

From Figure 15, following these movies are absolutely a very good movies that I'd recommend for you to watch.

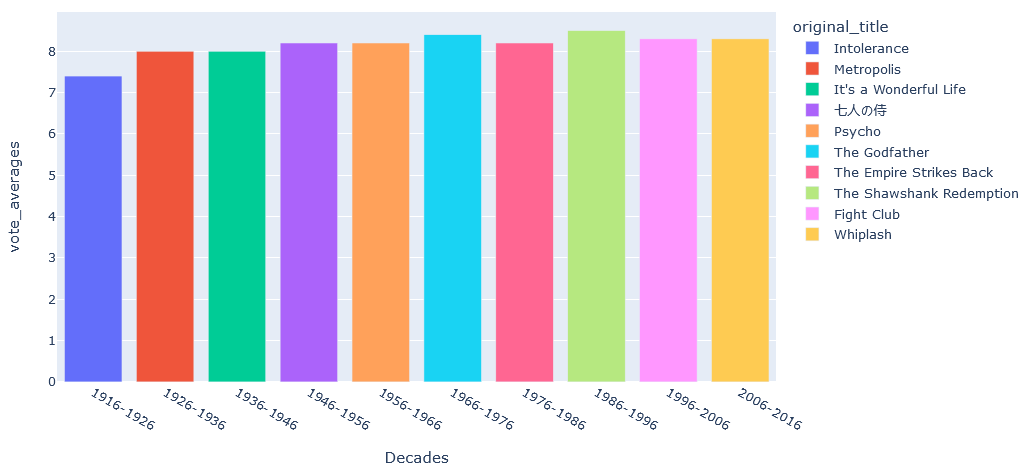


Figure 15: