Streaming Website Comparison

Jessie Lu (jl376@uw.edu)

CSE 163 - Intermediate Data Programming | Data Science Fair Project Report & Code | University of Washington

Summary

Research Question 1: Can IMDb scores of movies and TV shows be predicted based on their duration, streaming_website, production country count, number of voters and release year? What are the most significant factors influencing the IMDb scores of movies and TV shows across different streaming platforms?

IMDb scores of movies and TV shows can be predicted to some extent using their duration, streaming website, production country count, number of voters, and release year. Among these, the streaming website are the most significant factors influencing IMDb scores across different streaming platforms for movies. In TV shows, all factors has a slightly impact on TV shows.

Research Question 2: What is the distribution of genre for movies and TV shows between three different streaming website?

From the visualized data, it is evident that Netflix has the most extensive and diverse genre offerings for both movies and TV shows, followed by Hulu and Disney+. Netflix leads in genres like Comedies, Dramas, and International TV Shows, while Disney+ focuses more on Family-oriented content and Animation. Hulu offers a balanced distribution across Comedy, Documentaries, and Reality TV.

Research Question 3: What is the distribution of date added for movies and TV shows between three different streaming website?

The content addition trend for Netflix movies shows a steady increase over the years, with a noticeable spike around 2016. This indicates Netflix's growing library and investment in movie content over time. Similar to movies, the addition of TV shows on Netflix has steadily increased, with significant growth post-2014, which indicates Netflix's strategy to expand its TV show offerings continuously. The addition of content on Hulu shows a gradual increase, with a significant spike around 2020. Disney+ shows a massive spike in content addition around late 2019, aligning with the launch of the streaming service. Post-launch, the content addition stabilizes, reflecting a steady stream of new content being added.

Challenge Goals

Multiple Datasets:

I used multiple datasets from three different streaming website: Netflix, Hulu, and Disney. Each streaming platform dataset includes informations for movies and TV show. Then, I collected two extra datasets: IMDb score for movies and TV shows. Selecting out columns "scores" and "number of voters" and merging them with my three original streaming website datasets, I am able to answer my first research question. Also, by using multiple datasets, I am able to achieve higher relationality and diversity in the kinds of data that were analyzed.

New Libraries:

I used Plotly Express and Plotly Figure Factory from the Plotly library to make interactive visualizations for our analyses. We think adding interactive visualizations helps viewers pay more attention and understand the data better. We especially liked using the hover label feature in Plotly, which lets viewers see the x and y values of a data point by hovering over it, without needing to look at the axes. This was really useful for showing the relationships between different features. By stacking the plots on top of each other, we created a sense of comparison, and the hover labels highlighted detailed information, making it easier to understand the data.

Collaboration and Conduct

Students are expected to follow Washington state law on the Student Conduct Code for the University of Washington. In this course, students must:

- Indicate on your submission any assistance received, including materials distributed in this course.
- Not receive, generate, or otherwise acquire any substantial portion or walkthrough to an assessment.
- Not aid, assist, attempt, or tolerate prohibited academic conduct in others.

Update the following code cell to include your name and list your sources. If you used any kind of computer technology to help prepare your assessment submission, include the queries and/or prompts. Submitted work that is not consistent with sources may be subject to the student conduct process.

In [20]:
```
your_name = "Jessie Lu"
sources = [
    "Lecture",
    "Office Hour",
    {"Google":["How to make a random forest regression",
               "How to change the date into datetime",
               "How to converts the index to a DatetimeIndex",
               "How to pivot the data to have genres as rows and streaming sources as columns"]
              },
    {"Stack OverFlow": "https://stackoverflow.com/"},
    {"Plotly": "https://plotly.com/"},
    {"Copilot":["How to add the table to the figure in RQ3",
                "How converts the dates to strings in the 'YYYY-MM' format",
                "How to draw a World Cloud"
               ]},
    {"Sample project":["How to get OLS regression result",
                       "How to draw linear relationship plot"]},
    "ed",
    {"Kaggle":["https://www.kaggle.com/"]}
]

assert your_name != "", "your_name cannot be empty"
assert ... not in sources, "sources should not include the placeholder ellipsis"
assert len(sources) >= 6, "must include at least 6 sources, inclusive of lectures and sections"
```

Data Setting

The first dataset is Hulu Movies and TV Shows.

Link: https://www.kaggle.com/datasets/shivamb/hulu-movies-and-tv-shows

Hulu is an online movie and tv shows streaming platform owned by The Walt Disney Company. Hulu is exclusive to the United States and is not available in other countries. This tabular dataset consists of listings of all the movies and tv shows available on Hulu, along with details such as - cast, directors, ratings, release year, duration, etc. In this dataset, columns like “day-added”-The date the title was added to Hulu., “release-year”-the year the movie or TV show was originally released, “listed_in”- the genres the title falls under, and “country”-the country or countries where the movie or TV show will help me to draw some plots and build regression models.

The second dataset is Netflix Movies and TV Shows.

Link: https://www.kaggle.com/datasets/rahulvyasm/netflix-movies-and-tv-shows

Netflix stands as a leading force in the realm of media and video streaming. This dataset comprehensively includes all offerings on Netflix, including vital details such as cast, directors, ratings, release year, duration, and more. In this dataset, columns like “day-added”-The date the title was added to Netflix, “release-year”- the year the movie or TV show was originally released, “listed_in”- the genres the title falls under, and “country”-the country or countries where the movie or TV show was produced will help me to draw some plots and build regression models.

The third dataset is Disney Movies and TV Shows.

Link: https://www.kaggle.com/datasets/shivamb/disney-movies-and-tv-shows

Disney+ is another one of the most popular media and video streaming platforms. This dataset consists of listings of all the movies and tv shows available on Amazon Prime, along with details such as - cast, directors, ratings, release year, duration, etc. In the same way, columns like “day-added”, “release-year”, “listed_in”, “director” and “country” in this dataset will help me to draw some plots and build regression models.

The fourth dataset is IMDb_movies_rating.

Link: https://www.kaggle.com/datasets/ashnaarora251/imdb-movies-dataset

The IMDB dataset contains about 80k movies listed on the official website of IMDb with information about titles, date published, user ratings, genres, overviews, cast and crew members, original titles, original languages, writer, director and countries of origin. From this dataset, I will merge columns “avg_vote”-average IMDb score for each movie, and “votes”-number of people who vote for the IMDb score into each of the previous three datasets: Hulu, Netflix and Disney.

The fifth dataset is IMDb_TV_Shows_rating.

Link: https://www.kaggle.com/datasets/ikjotsingh221/top-2k-tv-shows

This dataset is scraped from IMDB's official website, which includes details of the name of the TV show, the running time of the show (in minutes), its IMDB Rating, the number of votes it got, etc. From this dataset, I will merge columns “show_rating”-the IMDb score for each TV show, and “show_votes”-the number of people who vote for the IMDb score into each of the previous three datasets: Hulu, Netflix and Disney.

Methods

1. Write function if modify original dataset

2. Write test function

RQ1.

1. Load the total 5 datasets
2. Selct columns we need and merge these columns on each of other three datasets
3. Data cleaning and rename the columns
4. Combine three datasets and run RandomForest regression on movie and TV (including test) respectively with OlS regression result
5. Draw plots for actual & predicted value and residual&predicted value
6. Rename the columns into “movies_ratings number” and “TV_rating number” and combine them into one column, named “IMDb rating number”

RQ2.

7. Filter out movies or TV shows for each of other three datasets respectively
8. Combine three datasets and clean selected columns
9. Groupby based on streaming websites and use plotly to have plots of distribution of genre

RQ3.

10. Clean selected columns, change to the form of datetime(month), and group by month_added and type for each three datasets
11. Use function and plotly to better reveal three datasets together and identify trends in accumulated dataset.

Results and Code

Importing, Cleaning and Processing

It is important to highlight the dataset setup and cleaning process before dealing with the code that generated results. This crucial step was essential for understanding and utilizing the data and accounted for a significant portion of the project's coding effort.

The following packages were imported:

In [ ]:
```
pip install wordcloud
```

In [ ]:
```
pip install plotly
```

In [5]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import doctest

from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor
from sklearn.metrics import accuracy_score, mean_squared_error
from scipy.stats import randint
from sklearn.tree import export_graphviz
from IPython.display import Image

import plotly.express as px
import plotly.graph_objects as go
import plotly.figure_factory as ff
import statsmodels.api as sm
from plotly.subplots import make_subplots
from wordcloud import WordCloud

import plotly.io as pio
pio.renderers.default = 'notebook'
import plotly.io as pio
pio.renderers.default = 'colab'
```

The following cleaning and reformatting functions were used:

In [9]:
```
def read_csv(file_path):
    """
    Reads a CSV file and returns a pandas DataFrame.

    >>> TV = read_csv('imdb_TV_Shows.csv')
    >>> TV.iloc[0, 0]
```

```
    'Game of Thrones'
    """
    return pd.read_csv(file_path)


# example
ntitle = read_csv('netflix_titles.csv')
htitle = read_csv('hulu_titles.csv')
disney = read_csv('disney.csv')
TV = read_csv("imdb_TV_Shows.csv")
imdb5 = read_csv("IMDb movies 5.csv")

doctest.run_docstring_examples(read_csv, globals())
```

In [10]:
```
def clean_missing_values(df, columns):
    """
    Removes rows with missing values in the specified columns.

    Args:
        df (pd.DataFrame): The DataFrame to clean.
        columns (list): A list of columns to check for missing values.
    """
    return df.dropna(subset=columns)

test_1 = read_csv("test_1.csv")
test_2 = read_csv("test_2.csv")
test_3 = read_csv("test_3.csv")

cleaned_test_1 = clean_missing_values(test_1 , "title")
cleaned_test_2 = clean_missing_values(test_2 , "type")
cleaned_test_3 = clean_missing_values(test_3 , "score")

assert cleaned_test_1.isnull().values.any() == True
assert cleaned_test_2.isnull().values.any() == False
assert cleaned_test_3.isnull().values.any() == False
```

In [11]:
```
def add_streaming_website_label(df, streaming_website):
    """
    Adds a 'Streaming_website' column with the specified label to the DataFrame.

    Args:
        df (pd.DataFrame): The DataFrame to which the label should be added.
        streaming_website (str): The name of the streaming website to add to the 'Streaming_website' colum

    Returns:
        pd.DataFrame: The DataFrame with the new 'Streaming_website' column.
    """
    df.loc[:, "Streaming_website"] = streaming_website
    return df

test_1 = read_csv("test_1.csv")
test_2 = read_csv("test_2.csv")
test_3 = read_csv("test_3.csv")

added_test_1 = add_streaming_website_label(test_1, "Netflix")
added_test_2 = add_streaming_website_label(test_2, "Hulu")
added_test_3 = add_streaming_website_label(test_3, "Disney+")

assert added_test_2.columns.tolist() == ['type', 'title', 'Streaming_website']
assert added_test_1.columns.tolist() == ['title','score','votes','Unnamed: 3', 'Streaming_website']
assert added_test_3.columns.tolist() == ['title','score', 'Streaming_website']
```

In [12]:
```
def merge_datasets(df1, df2, on, how='left'):
    """
    Merges two DataFrames on a specified column using a specified join method.

    Args:
        df1 (pd.DataFrame): The first DataFrame.
        df2 (pd.DataFrame): The second DataFrame.
        on_column (str): The column name to merge on.
        how (str): The type of merge to perform (default is 'left').

    Returns:
        pd.DataFrame: The merged DataFrame.
    """
    merged_df = pd.merge(df1, df2, on=on, how=how)
    return merged_df

test_2 = read_csv("test_2.csv")
test_3 = read_csv("test_3.csv")
test_4 = read_csv("test_4.csv")
test_5 = read_csv("test_5.csv")

merged_test_2 = merge_datasets(test_2, test_3, on='title', how='left')
merged_test_4 = merge_datasets(test_4, test_5, on='title', how='left')

assert merged_test_2.columns.tolist() == ["type", "title", "score"]
assert merged_test_4.columns.tolist() == ["title", "score", "Country_count"]
```

The following code was used to import and set up the dataframes:

`RQ.1–Movie & TV`

In [13]:
```
## Movie
# reading in csv and select columns we need
imdb5 = read_csv("IMDb movies 5.csv")
imdb5.rename(columns={'avg_vote': 'movie_score', 'votes':'movie_votes'}, inplace=True)
imdb5_subset = imdb5[['title', 'movie_score', 'movie_votes']]

# merging datasets and cleaning missing value for each streaming website dataset
merged_net = merge_datasets(ntitle, imdb5_subset, on="title", how='left')
merged_net_clean = clean_missing_values(merged_net, ['movie_score', 'movie_votes'])
merged_hulu = merge_datasets(htitle, imdb5_subset, on="title", how='left')
merged_hulu_clean = clean_missing_values(merged_hulu, ['movie_score', 'movie_votes'])
merged_dis = merge_datasets(disney, imdb5_subset, on="title", how='left')
merged_dis_clean = clean_missing_values(merged_dis, ['movie_score', 'movie_votes'])

# filtering by type and adding extra column on merged dataset
filter = merged_net_clean["type"].str.contains("TV Show")
filtered_merged_net_clean = merged_net_clean[~filter]
filtered_merged_net_clean = add_streaming_website_label(filtered_merged_net_clean, "Netflix")
filter = merged_hulu_clean["type"].str.contains("TV Show")
filtered_merged_hulu_clean = merged_hulu_clean[~filter]
filtered_merged_hulu_clean = add_streaming_website_label(filtered_merged_net_clean, "Hulu")
filter = merged_dis_clean["type"].str.contains("TV Show")
filtered_merged_dis_clean = merged_dis_clean[~filter]
filtered_merged_dis_clean = add_streaming_website_label(filtered_merged_net_clean, "Disney+")

# combining three filtered merged dataset into one big dataset
combined_movie = pd.concat([filtered_merged_net_clean, filtered_merged_hulu_clean, filtered_merged_dis_clea
combined_movie = clean_missing_values(combined_movie, ['country', 'duration'])
combined_movie.loc[:, 'country_count'] = combined_movie['country'].str.split(',').str.len().astype(int)

# selecting columns what we need
columns = ['type','title','country','date_added', 'release_year', 'duration', 'movie_score','movie_votes',
           'Streaming_website','country_count']
combined_movie = combined_movie[columns]

# setting up dataset
combined_movie['Streaming_website'] = combined_movie['Streaming_website'].map({'Netflix':1,'Hulu':2,'Disney
combined_movie.loc[:, 'duration'] = combined_movie['duration'].astype(str)
combined_movie.loc[:, 'duration'] = combined_movie['duration'].str.replace('min', '')

##TV
# reading in csv and select columns we need
TV = read_csv("imdb_TV_Shows.csv")
TV.rename(columns={'show_name':'title','show_rating':'TV_score', 'show_votes':'TV_votes','show_genre':'TV_g
TV_subset = TV[['title', 'TV_score','TV_votes']]

# merging datasets and cleaning missing value
merged_net_TV = merge_datasets(ntitle, TV_subset, on='title', how='left')
merged_net_TV_clean = clean_missing_values(merged_net_TV, ['TV_score', 'TV_votes'])
merged_hulu_TV = merge_datasets(htitle, TV_subset, on='title', how='left')
merged_hulu_TV_clean =  clean_missing_values(merged_hulu_TV, ['TV_score', 'TV_votes'])
merged_dis_TV = merge_datasets(disney, TV_subset, on='title', how='left')
merged_dis_TV_clean = clean_missing_values(merged_dis_TV, ['TV_score', 'TV_votes'])


# filtering by type and adding an extra column to merged dataset
filter = merged_net_TV_clean["type"].str.contains("Movie")
filtered_merged_net_TV_clean = merged_net_TV_clean[~filter]
filtered_merged_net_TV_clean = add_streaming_website_label(filtered_merged_net_TV_clean, "Netflix")
filter = merged_hulu_TV_clean["type"].str.contains("Movie")
filtered_merged_hulu_TV_clean = merged_hulu_TV_clean[~filter]
filtered_merged_hulu_TV_clean = add_streaming_website_label(filtered_merged_hulu_TV_clean, "HULU")
filter = merged_dis_TV_clean["type"].str.contains("Movie")
filtered_merged_dis_TV_clean = merged_dis_TV_clean[~filter]
filtered_merged_dis_TV_clean = add_streaming_website_label(filtered_merged_dis_TV_clean, "Disney")

# combining three filtered merged dataset into one big dataset
combined_TV = pd.concat([filtered_merged_net_TV_clean, filtered_merged_hulu_TV_clean, filtered_merged_dis_T
combined_TV = clean_missing_values(combined_TV,['country', 'duration'])
combined_TV.loc[:, 'country_count'] = combined_TV['country'].str.split(',').str.len().astype(int)

# selecting columns what we need
columns = ['type','title','country','date_added', 'release_year', 'duration', 'TV_score','TV_votes',
           'Streaming_website','country_count']
combined_TV = combined_TV[columns]

# setting up dataset
combined_TV['Streaming_website'] = combined_TV['Streaming_website'].map({'Netflix':1,'HULU':2,'Disney':3})
combined_TV.loc[:, 'duration'] = combined_TV['duration'].astype(str)
combined_TV.loc[:, 'duration'] = combined_TV['duration'].str.replace('Season|Seasons', '').str.strip().str
combined_TV['TV_votes'] = combined_TV['TV_votes'].str.replace(',', '')
```

`RQ.2–Movie&TV`

In [14]:
```
##Movie
# cleaning missing value for specific columns
ntitle_2 = clean_missing_values(ntitle, ['listed_in', "country"])
htitle_2 = clean_missing_values(htitle, ['listed_in', "country"])
disney_2= clean_missing_values(disney, ['listed_in', "country"])

# adding column to original dataset
filter_movie = ntitle_2["type"].str.contains("TV Show")
filtered_ntitle_2_movie = ntitle_2[~filter_movie]
filtered_ntitle_2_movie = add_streaming_website_label(filtered_ntitle_2_movie, "Netflix")
filter_movie = htitle_2["type"].str.contains("TV Show")
filtered_htitle_2_movie = htitle_2[~filter_movie]
filtered_htitle_2_movie = add_streaming_website_label(filtered_htitle_2_movie, "Hulu")
filter_movie = disney_2["type"].str.contains("TV Show")
filtered_disney_2_movie = disney_2[~filter_movie]
filtered_disney_2_movie.loc[:,["Streaming_website"]] = "Disney+"

# combined three filtered dataset into a big dataset
df_combined_2_movie = pd.concat([filtered_ntitle_2_movie, filtered_htitle_2_movie, filtered_disney_2_movie

# setting up dataset
df_genres_movie = df_combined_2_movie.assign(genre=df_combined_2_movie.loc[:,'listed_in'].str.split(',')).
df_genres_movie.loc[:,'genre'] = df_genres_movie['genre'].str.strip()
df_genre_count_movie = df_genres_movie.groupby(['Streaming_website', 'genre']).size().reset_index(name='cou
df_pivot_movie = df_genre_count_movie.pivot(index='genre', columns='Streaming_website', values='count').fi
df_pivot_movie = df_pivot_movie[(df_pivot_movie >= 150).any(axis=1)]

##TV
# cleaning missing value for specific columns
ntitle_2 = clean_missing_values(ntitle,['listed_in', "country"])
htitle_2 = clean_missing_values(htitle,['listed_in', "country"])
disney_2 = clean_missing_values(disney,['listed_in', "country"])

# adding column to original dataset
filter_TV = ntitle_2["type"].str.contains("Movie")
filtered_ntitle_2_TV = ntitle_2[~filter_TV]
filtered_ntitle_2_TV = add_streaming_website_label(filtered_ntitle_2_TV, "Netflix")
filter_TV = htitle_2["type"].str.contains("Movie")
filtered_htitle_2_TV = htitle_2[~filter_TV]
filtered_htitle_2_TV = add_streaming_website_label(filtered_htitle_2_TV, "Hulu")
filter_TV = disney_2["type"].str.contains("Movie")
filtered_disney_2_TV = disney_2[~filter_TV]
filtered_disney_2_TV = add_streaming_website_label(filtered_disney_2_TV, "Disney+")

# combined three filtered dataset into a big dataset
df_combined_2_TV = pd.concat([filtered_ntitle_2_TV, filtered_htitle_2_TV, filtered_disney_2_TV], ignore_in

# setting up dataset
df_genres_TV = df_combined_2_TV.assign(genre=df_combined_2_TV.loc[:,'listed_in'].str.split(',')).explode('
df_genres_TV.loc[:,'genre'] = df_genres_TV['genre'].str.strip()
df_genre_count_TV = df_genres_TV.groupby(['Streaming_website', 'genre']).size().reset_index(name='count')
df_pivot_TV = df_genre_count_TV.pivot(index='genre', columns='Streaming_website', values='count').fillna(0
df_pivot_TV = df_pivot_TV[(df_pivot_TV >= 150).any(axis=1)]
```

RQ.3

In [15]:
```
# Netflix
ntitle_3 = clean_missing_values(ntitle, ['date_added', 'type'])
ntitle_3['date_added'] = pd.to_datetime(ntitle_3['date_added'], format='%B %d, %Y', errors='coerce')
ntitle_3 = clean_missing_values(ntitle_3,['date_added'])
ntitle_3['month_added'] = ntitle_3['date_added'].dt.to_period('M')
agg_ntitle_3 = ntitle_3.groupby(['month_added', 'type']).size().unstack(fill_value=0)
agg_ntitle_3.index = agg_ntitle_3.index.to_timestamp()

#Hulu
htitle_3 = clean_missing_values(htitle, ['date_added', 'type'])
htitle_3['date_added'] = pd.to_datetime(htitle_3['date_added'], format='%B %d, %Y', errors='coerce')
htitle_3 = clean_missing_values(htitle_3,['date_added'])
htitle_3['month_added'] = htitle_3['date_added'].dt.to_period('M')
agg_htitle_3 = htitle_3.groupby(['month_added', 'type']).size().unstack(fill_value=0)
agg_htitle_3.index = agg_htitle_3.index.to_timestamp()

#Disney
disney_3 = clean_missing_values(disney, ['date_added', 'type'])
disney_3['date_added'] = pd.to_datetime(disney_3['date_added'], format='%B %d, %Y', errors='coerce')
disney_3 = clean_missing_values(disney_3,['date_added'])
disney_3['month_added'] = disney_3['date_added'].dt.to_period('M')
agg_disney_3 = disney_3.groupby(['month_added', 'type']).size().unstack(fill_value=0)
agg_disney_3.index = agg_disney_3.index.to_timestamp()
```

RQ1: Can IMDb scores of movies and TV shows be predicted based on their duration, streaming_website, production country count, number of voters and release year? What are the most significant factors influencing the IMDb scores of movies across different streaming platforms?

We include five regressors : "release year", "duration", "movie_votes", "Streaming websites", and "country_count" in RandomForest Regression, trying to predict actual movie IMDb score. Based on OLS regression model, each regressor has p_value of 0.0, which indicates that all of them are statistically significant. Only the "release year" has negative effect on IMdb score, which suggests that newer movies tend to have slightly lower IMDb scores. The rest of regressors has the positive effect, especially for the "Streaming_website". Based on the linear relationship between streaming website and movies score, we can observe that Disney has a relatively higher IMDb score, which is a surprising outcome. I expected that Netflix will have a better performance on IMDb score, since Netflix stands as a leading force in the realm of media and video streaming. The outcome is possibly biased by small dataset of Disney.

Also, R-squared and Adjusted R-squared, both values are 0.183, which indicates that approximately 18.3% of the variance in the dependent variable (movie_score) can be explained by the independent variables (release_year, duration, movie_votes, Streaming_website, country_count). While not very high, it does suggest that these factors have some predictive power, suggesting that other factors not included in the model may have a significant impact on movie scores.

The scatter plot shows a strong linear relationship between the actual and predicted values, indicating that the model predicts the movie scores reasonably well, which also proved by low MSE value. Most of the points lie along the diagonal, suggesting good predictive performance. However, there are some deviations and outliers, which could be due to factors not captured by the model. The residuals plot shows a relatively random scatter around the zero line, indicating that the model does not suffer from heteroscedasticity or non-linearity. There is no clear pattern in the residuals, which is a good sign for the model's validity.

In [16]:
```
columns=['type','title','country','date_added','movie_score']
X = combined_movie.drop(columns, axis=1)
y = combined_movie['movie_score']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

clf = RandomForestRegressor(n_estimators=100, random_state=42)
clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse}')
```

```
Mean Squared Error: 0.13846204393705486
```

In [17]:
```
X = combined_movie.drop(columns,axis=1)
Y = combined_movie['movie_score']
X = sm.add_constant(X)
m = sm.OLS(Y.astype(float), X.astype(float))

r = m.fit()
r.summary()
```

Out[17]:

OLS Regression Results

Dep. Variable:	movie_score	**R-squared:**	0.183
Model:	OLS	**Adj. R-squared:**	0.183
Method:	Least Squares	**F-statistic:**	560.9
Date:	Wed, 05 Jun 2024	**Prob (F-statistic):**	0.00
Time:	18:01:45	**Log-Likelihood:**	-15187.
No. Observations:	10017	**AIC:**	3.038e+04
Df Residuals:	10012	**BIC:**	3.042e+04
Df Model:	4		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
release_year	-0.0134	0.001	-11.912	0.000	-0.016	-0.011
duration	0.0115	0.000	23.283	0.000	0.011	0.012
movie_votes	2.747e-06	9.48e-08	28.966	0.000	2.56e-06	2.93e-06
Streaming_website	10.5188	0.759	13.862	0.000	9.031	12.006
country_count	0.0649	0.013	5.011	0.000	0.040	0.090

Omnibus:	809.014	**Durbin-Watson:**	1.863
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	1089.672
Skew:	-0.690	**Prob(JB):**	2.40e-237
Kurtosis:	3.841	**Cond. No.**	8.79e+06

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 8.79e+06. This might indicate that there are
strong multicollinearity or other numerical problems.

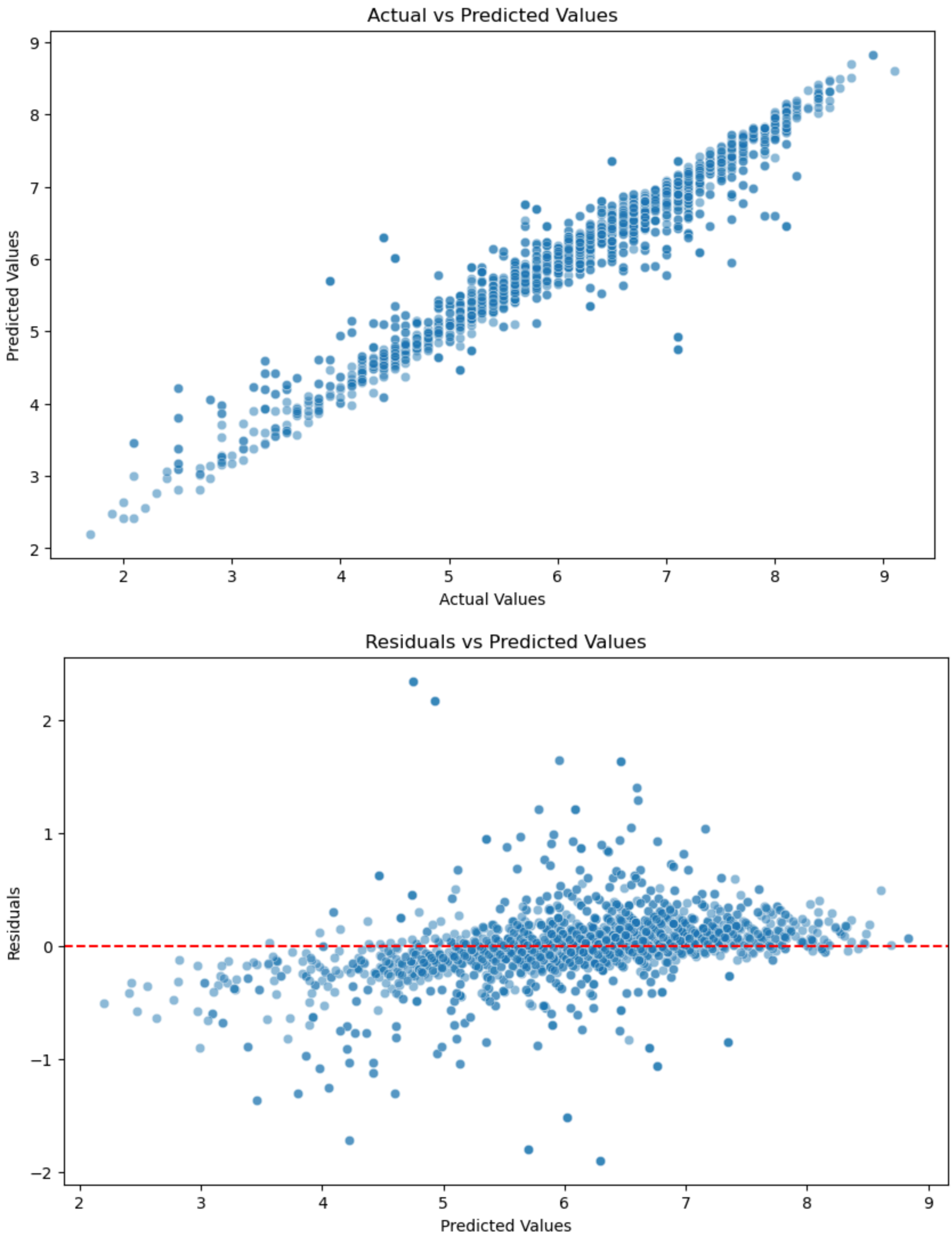
In [18]:
```
fig = px.scatter(combined_movie, x='Streaming_website', y='movie_score', trendline='ols',
                 title='Linear Relationship between streaming_website and Movie Score')
fig.show()
```

In [184…
```
#Create a scatter plot to compare the actual and predicted values
plt.figure(figsize=(10, 6))
sns.scatterplot(x=y_test, y=y_pred, alpha=0.5)
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.title('Actual vs Predicted Values')
plt.show()

#Create a scatter plot to compare the residual and predicted values
```

```
residuals = y_test - y_pred

plt.figure(figsize=(10, 6))
sns.scatterplot(x=y_pred, y=residuals, alpha=0.5)
plt.axhline(0, color='red', linestyle='--')
plt.xlabel('Predicted Values')
plt.ylabel('Residuals')
plt.title('Residuals vs Predicted Values')
plt.show()
```



In TV shows, there is a much smaller dataset that movie part. I also include five regressors : "release year", "duration", "TV_votes", "Streaming websites", and "country_count" in RandomForest Regression, trying to predict TV IMDb score. Based on OLS regression model, "release year", "duration" and "country_count" has p-value of 0.194, 0.509, and 0.123, which indicates that they are not statistically significant. Only the "TV_votes" and "Streaming websites" are statistically significant. Same as the result from movie regression model, "release year" has negative effect on IMdb score, which suggests that newer movies tend to have slightly lower IMDb scores. The rest of regressors has the positive effect. However, based on the coefficient of each regressor, we are able to observe that they all have slightly impact on predicting IMDb TV score, which is consistent to the value of adjusted R-squared : 0.144. This number indicates that only about 14.4% of the variance in the dependent variable (TV_score) can be explained by the independent variables (release_year, duration, TV_votes, Streaming_website, country_count). This is relatively low, suggesting that many factors influencing TV scores are not captured by the model.

The plot for linear relationship between country count and TV score exposes a moderately positive connection. We can speculate that with more countries in producing TV shows, it will possibly get a higher score.

The scatter plot shows a weak linear relationship. The predicted values do not align closely with the actual values, indicating that the model's predictive performance for TV scores is limited. The spread of points suggests that the model does not capture the variability in TV scores well. The residuals plot shows a relatively random scatter around the zero line, indicating that the model does not suffer from heteroscedasticity or non-linearity. There is no clear pattern in the residuals, which is a good sign for the model's validity.

In [21]:
```
columns=['type','title','country','date_added','TV_score']
X = combined_TV.drop(columns, axis=1)
y = combined_TV['TV_score']

#test_size=0.2indicates 80% training dataset and 20% testing dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

clf = RandomForestRegressor(n_estimators=100, random_state=42)
clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse}')
```

```
Mean Squared Error: 0.4704164036697254
```

In [22]:
```
X = combined_TV.drop(columns,axis=1)
Y = combined_TV['TV_score']
X = sm.add_constant(X)
m = sm.OLS(Y.astype(float), X.astype(float))

r = m.fit()
r.summary()
```

Out[22]:

OLS Regression Results

Dep. Variable:	TV_score	**R-squared:**	0.144
Model:	OLS	**Adj. R-squared:**	0.136
Method:	Least Squares	**F-statistic:**	18.04
Date:	Wed, 05 Jun 2024	**Prob (F-statistic):**	1.48e-16
Time:	18:06:41	**Log-Likelihood:**	-695.39
No. Observations:	541	**AIC:**	1403.
Df Residuals:	535	**BIC:**	1429.
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	17.2126	7.399	2.326	0.020	2.678	31.747
release_year	-0.0048	0.004	-1.301	0.194	-0.012	0.002
duration	0.0065	0.010	0.661	0.509	-0.013	0.026
TV_votes	1.803e-06	2.3e-07	7.857	0.000	1.35e-06	2.25e-06
Streaming_website	-0.2210	0.067	-3.283	0.001	-0.353	-0.089
country_count	0.1057	0.069	1.543	0.123	-0.029	0.240

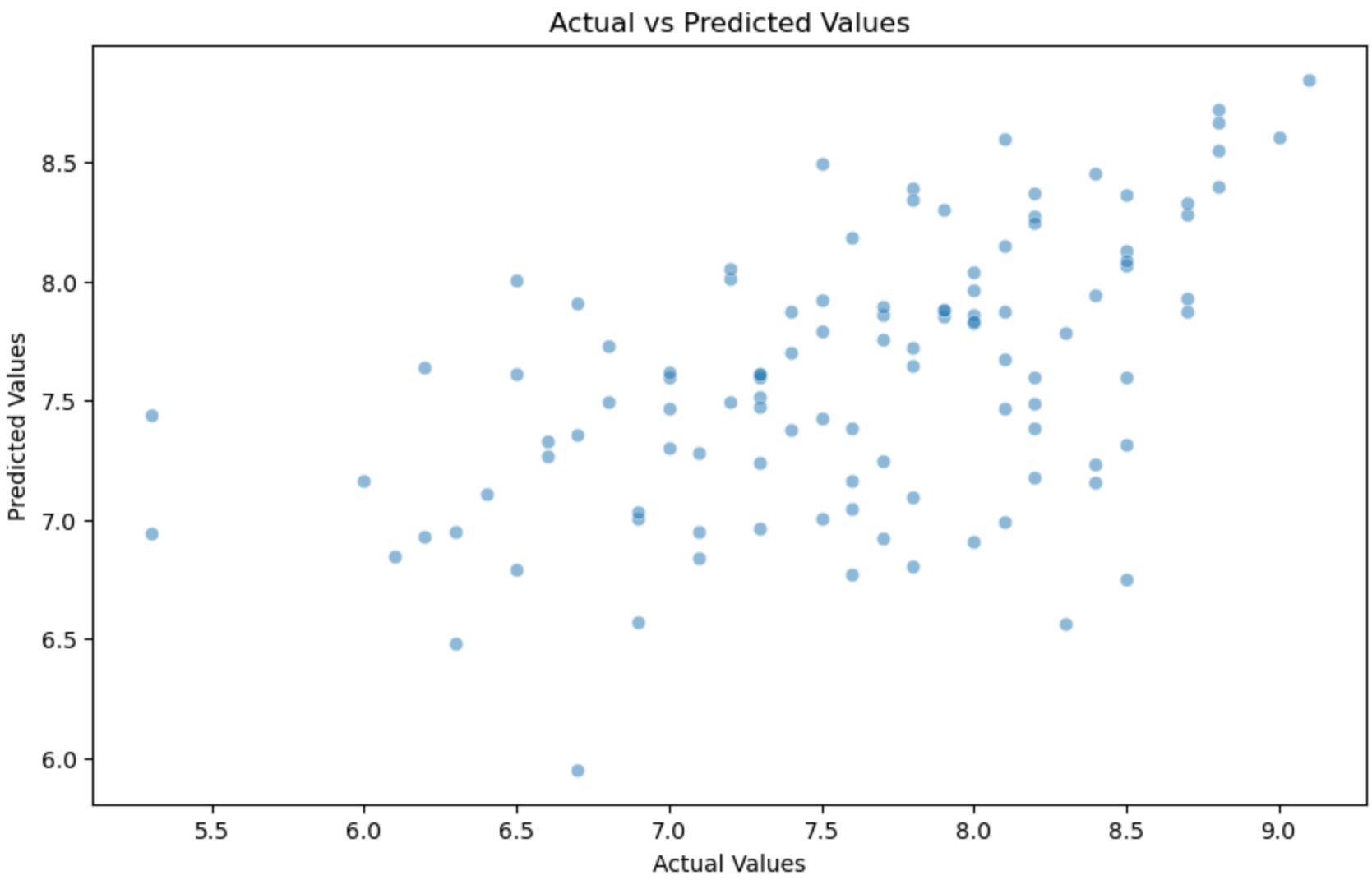
Omnibus:	202.130	**Durbin-Watson:**	1.794
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	745.563
Skew:	-1.717	**Prob(JB):**	1.27e-162
Kurtosis:	7.613	**Cond. No.**	3.71e+07

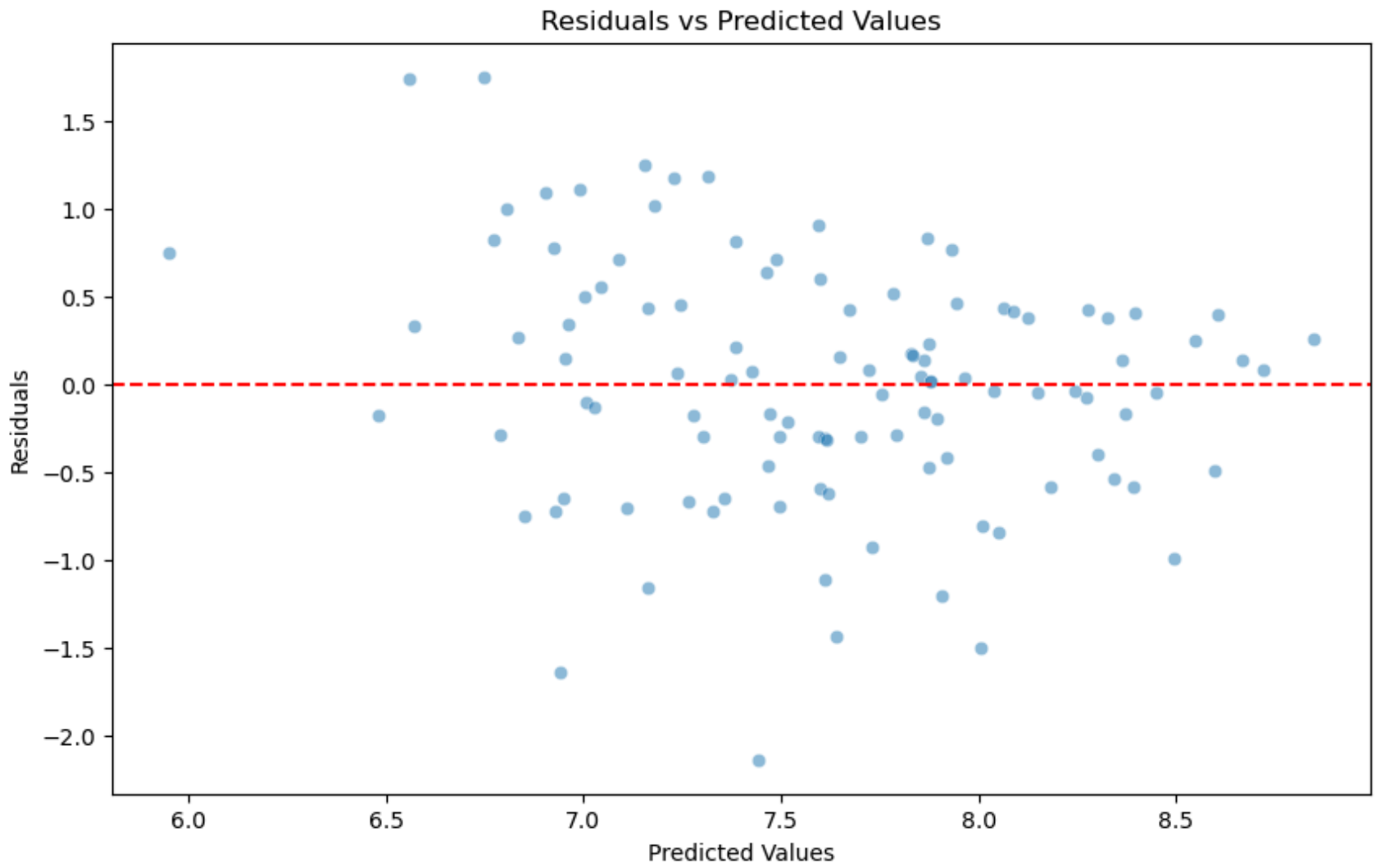
Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.71e+07. This might indicate that there are
strong multicollinearity or other numerical problems.

In [23]:
```
fig = px.scatter(combined_TV, x='country_count', y='TV_score', trendline='ols',
                 title='Linear Relationship between Country count and TV Score')
fig.show()
```

In [24]:
```
plt.figure(figsize=(10, 6))
sns.scatterplot(x=y_test, y=y_pred, alpha=0.5)
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.title('Actual vs Predicted Values')
plt.show()

residuals = y_test - y_pred
plt.figure(figsize=(10, 6))
sns.scatterplot(x=y_pred, y=residuals, alpha=0.5)
plt.axhline(0, color='red', linestyle='--')
plt.xlabel('Predicted Values')
plt.ylabel('Residuals')
plt.title('Residuals vs Predicted Values')
plt.show()
```





RQ2: What is the distribution of genre for movies and TV shows between three different streaming website?

In the first plot, we can notice that Netflix has a significantly larger count of movies across most genres compared to Hulu and Disney+. Genres like Comedies, Dramas, International Movies, and Documentaries have the highest count on Netflix. Hulu shows a relatively lower count in all movie genres compared to Netflix, with noticeable representation in Documentaries, Animation, and Horror Movies. Disney+, while also trailing Netflix, shows a strong presence in Family, Animation, and Children & Family Movies genres, aligning with its brand focus on family-friendly content.

The second plot exhibits that similar to movies, Netflix leads in the number of TV shows across various genres. Particularly high counts are observed in International TV Shows, TV Dramas, and TV Comedies. Hulu shows substantial counts in genres such as Anime, Adventure, and Comedy TV Shows. It indicates Hulu's focus on these specific genres to attract niche audiences. Disney+ has a smaller but focused range of TV shows, with significant counts in genres like Kids' TV and Comedy. This aligns with Disney's family-oriented content strategy, extending to TV shows as well.

These visualizations and insights are crucial for understanding how each streaming platforms show its content to attract and retain subscribers. The data-driven approach used in this analysis highlights the competitive landscape and strategic differences among the major streaming platforms.

In [25]:
```
fig1 = go.Figure()

fig1.add_trace(go.Bar(
    x=df_pivot_movie.index,
    y=df_pivot_movie['Netflix'],
    name='Netflix',
    marker_color='green',
    width=0.4
))

fig1.add_trace(go.Bar(
    x=df_pivot_movie.index,
    y=df_pivot_movie['Hulu'],
    name='Hulu',
    marker_color='pink',
    width=0.4
))

fig1.add_trace(go.Bar(
    x=df_pivot_movie.index,
    y=df_pivot_movie['Disney+'],
    name='Disney+',
    marker_color='red',
    width=0.4
))

fig1.update_layout(
    width=1300,
    height=800,
    barmode='group',
    xaxis_title={'text': 'Genre', 'font': {'size': 20}
```

```
                },
    yaxis_title={'text': 'Count', 'font': {'size': 20}
                },
    title={
        'text': "Genre Distribution Across Streaming Services-Movies",
        'font': {'size': 24}  # Set the font size for the title
    }
)
fig1.update_xaxes(tickangle=-45, tickfont=dict(size=15))

fig2 = go.Figure()

fig2.add_trace(go.Bar(
    x=df_pivot_TV.index,
    y=df_pivot_TV['Netflix'],
    name='Netflix',
    marker_color='green',
    width=0.4
))

fig2.add_trace(go.Bar(
    x=df_pivot_TV.index,
    y=df_pivot_TV['Hulu'],
    name='Hulu',
    marker_color='pink',
    width=0.4
))

fig2.add_trace(go.Bar(
    x=df_pivot_TV.index,
    y=df_pivot_TV['Disney+'],
    name='Disney+',
    marker_color='red',
    width=0.4
))

fig2.update_layout(
    width=1300,
    height=800,
    barmode='group',
     xaxis_title={'text': 'Genre', 'font': {'size': 20}
                },
    yaxis_title={'text': 'Count', 'font': {'size': 20}
                },
    title={
        'text': "Genre Distribution Across Streaming Services-TV show",
        'font': {'size': 24}  # Set the font size for the title
    }
)

fig2.update_xaxes(tickangle=-45, tickfont=dict(size=15))

fig1.show()
fig2.show()
```

In [28]:
```
fig, (ax1, ax2, ax3) = plt.subplots(3, 1, figsize=(30, 15))

# Generate word clouds
wordcloud1 = WordCloud(
    background_color='white',
    width=1024,
    height=768,
    colormap='rainbow'
).generate(" ".join(ntitle_2['listed_in']))

wordcloud2 = WordCloud(
    background_color='white',
    width=1024,
    height=768,
    colormap='rainbow'
).generate(" ".join(htitle_2['listed_in']))

wordcloud3 = WordCloud(
    background_color='white',
    width=1024,
    height=768,
    colormap='rainbow'
).generate(" ".join(disney_2['listed_in']))


ax1.imshow(wordcloud1)
ax1.axis('off')
ax1.set_title('Netflix',fontsize=23)

ax2.imshow(wordcloud2)
ax2.axis('off')
ax2.set_title('Hulu',fontsize=27)

ax3.imshow(wordcloud3)
ax3.axis('off')
ax3.set_title('Disney+',fontsize=20)


plt.tight_layout(pad=2.0)
plt.show()
```

Netflix

Hulu

Disney+

RQ3:What is the distribution of date added for movies and TV shows between three different streaming website?

Netflix has the addition of movies started around 2008 and shows a significant increase starting around 2016, with a consistent upward trend and the addition of TV shows also began around 2008, with a notable increase around 2016, similar to movies. Its consistent and robust addition of both movies and TV shows over time reflect its strategy to provide a vast and diverse library of content. The sharp increase post-2016 may be attributed to Netflix's aggressive expansion and original content production strategy.

Hulu has the addition of movies started later compared to Netflix, around 2010, and has shown a steady increase over the years, and the addition of TV shows also began around 2010, with a steady but slower growth rate compared to movies. Although Hulu exhibits steady growth in content addition but at a slower pace compared to Netflix. Their focus seems more balanced between movies and TV shows without significant spikes, indicating a more measured approach to content addition. Furthermore, we can easily observe the higher value from November to January, especially in Netflix. We expect the reason of long holiday, such as Thanksgiving and Christmas. People are tend to have leisure time to take a look on movies or TV shows.

Disney+ shows a unique trend with a significant spike around 2020, likely coinciding with its launch. After this spike, the addition rate stabilized at a lower but consistent level. I infer that the sharp spike in content addition around 2020 aligns with its launch, reflecting an initial large upload of content to establish its library. Post-launch, the content addition rate stabilized, indicating a focus on maintaining rather than rapidly expanding the library.

In [27]:
```
def create_plot(dataframes):
    """
    Creates a plot for multiple DataFrames.

    Args:
        dataframes (list): A list of tuples, each containing a DataFrame and its corresponding label.
    """
    # Calculate the number of rows needed for the subplots
    num_services = len(dataframes)
    rows = num_services + 1

    # Create the Plotly figure with subplots
    fig = make_subplots(
        rows=rows, cols=1,
        shared_xaxes=True,
        vertical_spacing=0.03,
        specs=[[{"type": "table"}]] + [[{"type": "scatter"}]] * num_services
    )

    # Add the table to the figure
    table_values = [dataframes[0][0].index.strftime('%Y-%m')]
    for df, label in dataframes:
        table_values.append(df['Movie'])
        table_values.append(df['TV Show'])

    header_values = ["Date"]
    for _, label in dataframes:
        header_values.append(f"{label} Movies")
        header_values.append(f"{label} TV Shows")

    fig.add_trace(
        go.Table(
            header=dict(
                values=header_values,
                font=dict(size=10),
                align="left"
            ),
            cells=dict(
                values=table_values,
                align="left")
        ),
        row=1, col=1
    )

    # Define colors and symbols for each streaming website
    movie_colors = ['#1f77b4', '#ff7f0e', '#2ca02c']
    tv_colors = ['#d62728', '#9467bd', '#8c564b']
    symbols_movies = ['circle', 'triangle-up', 'cross']
    symbols_tv = ['square', 'triangle-down', 'diamond']

    # Add scatter plots for each streaming website
    for i, (df, label) in enumerate(dataframes):
        fig.add_trace(
            go.Scatter(
                x=df.index,
                y=df['Movie'],
                mode="lines+markers+text",
                name=f"{label} Movies Added",
                marker=dict(color=movie_colors[i], size=5, symbol=symbols_movies[i]),
                line=dict(color=movie_colors[i]),
            ),
```

```
            row=i+2, col=1
        )

        fig.add_trace(
            go.Scatter(
                x=df.index,
                y=df['TV Show'],
                mode="lines+markers+text",
                name=f"{label} TV Shows Added",
                marker=dict(color=tv_colors[i], size=5, symbol=symbols_tv[i]),
                line=dict(color=tv_colors[i]),
            ),
            row=i+2, col=1
        )

    fig.update_layout(
        height=1200,
        showlegend=True,
        title_text="Content Addition Over Time",
        xaxis=dict(
            tickformat="%Y-%m",
            title="Date"
        )
    )
    fig.show()


create_plot([
    (agg_ntitle_3, "Netflix"),
    (agg_htitle_3, "Hulu"),
    (agg_disney_3, "Disney+")
])
```

Implications and Limitations

In the research question 1, the regression analyses for both movie and TV scores provide valuable insights but come with significant limitations. The positive impact of streaming platforms on movie scores and the negative impact on TV scores highlight the influential role of content distribution channels. Additionally, user engagement, measured by the number of votes, positively correlates with higher ratings for both movies and TV shows. However, the low R-squared values (0.183 for movies and 0.144 for TV shows) indicate that the models explain only a small portion of the variance in scores, suggesting that many important predictors are missing. The non-normal distribution of residuals further questions the models' validity, and high condition numbers point to potential multicollinearity issues, complicating the interpretation of individual predictors' effects, which explains Note[2], in the OLS regression results. The insignificant coefficients for variables like release year, duration, and country count in the TV model were unexpected and indicate that other unaccounted factors might play a more crucial role. These findings underscore the need for more comprehensive models that include additional variables, address multicollinearity, and explore non-linear relationships to enhance predictive power and validity. Moreover, the smaller sample

size for TV shows (541) compared to movies (10,017) may limit the generalizability of the TV score model, necessitating caution in interpreting these results. Future research should focus on these areas to provide more robust and insightful analyses.

In the research question 2, these plots provide a clear overview of the genre distribution across Netflix, Hulu, and Disney+, highlighting how each streaming service tailors its content to specific audience segments. Netflix appears to aim for broad appeal with extensive genre coverage, while Disney+ and Hulu focus on niche markets. This can help in understanding each platform's market positioning and strategic content decisions. For example, companies like Netflix, Hulu, and Disney+ can use this analysis to understand their strengths and weaknesses in content distribution. This can guide their content acquisition and production strategies to fill gaps and enhance their market positioning. However, the broad genre categories, lack of data on shared content, and absence of total counts limit the depth of insights. Future analysis could benefit from more granular genre breakdowns, current data updates, and consideration of content popularity and quality to offer a more comprehensive view of streaming service offerings.

In the research question 3, the plots clearly show the trends in content addition over time for Netflix, Hulu, and Disney+. This helps in understanding how each platform has evolved its content library over the years. By displaying multiple platforms on the same graph, it is easy to compare the growth rates and content addition strategies of each platform. Streaming providers, analysts, and content creators all can benefit from these plots. For example, investors can identify platforms with aggressive growth strategies and make informed investment decisions based on content library expansion trends. However, it might harm smaller services and consumers by not adequately addressing content quality and diversity. Additionally, the plots do not provide contextual information about why certain spikes occur, such as major platform launches, acquisitions, or market shifts. Annotations could help in understanding these events.