Final Project

For the following project, you will be working with a movie dataset. The dataset is here. The dataset columns are as follows:

- Title: The movie's title
- Genre: The movie's genre
- Stars: The number of famous actors in the movie
- Runtime: The length of the movie's runtime
- Budget: How much was spent on filming the movie (in millions)
- Promo: How much money was spent promoting the movie (in millions)
- Season: The season in which the movie was released
- Rating: The movie's rating
- R1: Reviewer 1's review
- R1: Reviewer 2's review
- R1: Reviewer 3's review

And the target variable:

- Success: Whether the film was a success or a flop

Fill in the answers to questions in the text field, and show your code below.

Data loading

Load the data

```
import pandas as pd

df = pd.read_csv('CMSC320FinalProjectData.csv')

movie_counts = df['Title'].value_counts()
print(movie_counts)
print(df.dtypes)
```

```
"Whispers of Redemption"           4
"Shadow Strike"                    4
"Fading Memories"                  4
"Aurora's Legacy"                  4
"Echoes of Tomorrow"               4
                                  ..
"Final Strike"                     1
"Thunderbolt Fury"                 1
"Code of Honor: Dark Redemption"   1
"Warrior's Vow"                    1
```

```
"Thunderstorm Showdown"             1
Name: Title, Length: 460, dtype: int64
Unnamed: 0      int64
Title          object
Runtime         int64
Stars           int64
Year            int64
Budget        float64
Promo         float64
Season         object
Rating         object
Genre          object
R1             object
R2             object
R3             object
Success          bool
dtype: object
```

Data Cleaning

List the three biggest data errors below, with a summary of how you fixed them and why you choose that method:

- The first biggest data error I saw was that all 3 reviews for every movie is a string and it is hard to do data science stuff on these strings. My first instinct was to read each string and give it a numerical rating myself, but I quickly realized that it would take way too long. Then I decided (by reading through Piazza) it is a lot easier and a lot cooler to import a sentiment analyzer and have it search for some "red flag" words that determine if the reviewer felt negatively towards or positively towards the film.
- The second biggest data error I saw was that there were some clear outliers in the Runtime and Stars column. Specifically, there were some movies with a 0 minute runtime. To alleviate this, I replaced all the 0 minute runtimes with the average runtime of all the movies. Additionally, the Stars column had some movies that had 100 stars which isn't possible. To alleviate this, I replaced the 100 stars with the number of stars for a movie with that same production budget.
- The last biggest data error I saw was that the data needs to account for inflation. This is a necessary step to truly understand the data when considering the production/promotional budgets of the films and seeing if that has any connection with the success of the films.

First take care of data error 1 (the reviews not being usable since right now they are just long strings)

```
!pip install nltk
import nltk
nltk.download('vader_lexicon')
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages (3.8.1)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from nltk) (8.1.7)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from nltk) (1.3.2)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.10/dist-packages (from nltk) (2023.6.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from nltk) (4.66.1)
```

```
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
```

```
True
```

```python
from nltk.sentiment.vader import SentimentIntensityAnalyzer
from sklearn.preprocessing import LabelEncoder

def analyze_review(text, pos_threshold=0.1, neg_threshold=-0.1):
    analyzer = SentimentIntensityAnalyzer()
    sentiment_scores = analyzer.polarity_scores(text)

    if sentiment_scores['compound'] >= pos_threshold:
        return "Positive"
    else:
        return "Negative"

# example usage to make sure this sentiment analyzer works
review = "I thought this movie wasn't good "
sentiment = analyze_review(review)
print(f"Sentiment: {sentiment}")

# Make a for loop to loop through the 3 columns of reviews and run the seniment analyzer.
# First I am going to make new columns and verify the first few entries to make sure
# the sentiment analyzer is working properly the way I want it to.
columns_to_analyze = ['R1', 'R2', 'R3']
for col in columns_to_analyze:
    df[f'Sentiment_{col}'] = df[col].apply(analyze_review)

print(df)

# display the counts of positive and negative for each column just for me to see, this doesn't impact my project
for col in columns_to_analyze:
    positive_count = (df[f'Sentiment_{col}'] == 'Positive').sum()
    negative_count = (df[f'Sentiment_{col}'] == 'Negative').sum()
    positive_counts = positive_count
    negative_counts = negative_count
```

```python
for col in columns_to_analyze:
    print(f"Column '{col}':")
    print(f"Positive Reviews: {positive_counts}")
    print(f"Negative Reviews: {negative_counts}")
    print()

# then I encode the data so it will be usable for me. 1 for positive
reviews and 0 for negative reviews.

# Create a LabelEncoder instance
label_encoder = LabelEncoder()

# Assuming you have already created your DataFrame 'df' with the
'Sentiment' columns

# Define the columns you want to encode
columns_to_encode = ['Sentiment_R1', 'Sentiment_R2', 'Sentiment_R3']

# Apply label encoding to the specified columns
for col in columns_to_encode:
    df[col] = label_encoder.fit_transform(df[col])

# Display the updated DataFrame
print(df)
```

```
Sentiment: Negative
     Unnamed: 0                          Title  Runtime  Stars  Year  \
0             0          "Love in the Inbox"      126      1  2020
1             1       "Coffee Shop Serendipity"      131      0  2020
2             2      "The Wedding Date Dilemma"      132      4  2000
3             3    "Heartstrings and Highways"      132      1  2015
4             4          "Falling for Cupid"      119      1  2015
..          ...                            ...      ...    ...   ...
535         535              "Shadow Strike"      128      3  2021
536         536              "Riot Protocol"      123      1  2018
537         537          "Deadlock Vendetta"      121      1  2003
538         538      "Blade Runner Protocol"      124      1  2007
539         539            "Eagle Eye Blitz"      126      0  2022

           Budget      Promo  Season Rating            Genre  \
0    6.679387e+07  73.543754  Winter     PG  Romantic Comedy
1    4.667863e+01  33.572003    Fall     PG  Romantic Comedy
2    3.639134e+01  54.561523  Summer     PG  Romantic Comedy
3    9.324732e+01  59.714535  Winter   PG13  Romantic Comedy
4    9.213021e+01  67.643810    Fall   PG13  Romantic Comedy
..            ...        ...     ...    ...              ...
535  6.489702e+01  91.445593    Fall     PG           Action
536  3.098935e+01  46.045408  Summer      R           Action
537  4.857255e+01  63.660912  Summer     PG           Action
```

```
538  1.364682e+02  188.513344  Summer      R           Action
539  1.276156e+02  158.496055  Summer     PG           Action

                                                    R1  \
0    "An unconvincing portrayal of suspense that fa...
1    "A movie that feels disjointed and fails to co...
2    "An underwhelming cinematic effort with unconv...
3    "A film that fails to resonate due to its lack...
4    "A movie that struggles to evoke any genuine e...
..                                                 ...
535  "Weak and contrived dialogue that lacks authen...
536                "A film that lingers in the memory."
537  "A lack of cohesion in the storytelling that m...
538   "An evocative journey that captivates the soul."
539        "A movie that speaks a universal language."

                                                    R2  \
0    "An uninspired plotline that lacks coherence a...
1    "An attempt at humor that lacks cleverness and...
2    "An emotionally resonant movie that connects u...
3    "A beautifully crafted narrative that unfolds ...
4    "A testament to the power of storytelling, lea...
..                                                 ...
535  "An overemphasis on spectacle over substance t...
536       "A celebration of life and its intricacies."
537  "An overly convoluted plot that confuses rathe...
538  "An absence of emotional depth, resulting in a...
539  "Uninspired storytelling that fails to ignite ...

                                                    R3  Success
Sentiment_R1  \
0    "A visually captivating masterpiece that mesme...    False
Negative
1    "A timeless classic that continues to enchant ...    False
Negative
2    "A cinematic triumph that surpasses boundaries...    False
Negative
3    "An uninspired portrayal of drama that feels s...    False
Negative
4    "An uplifting film that leaves a profound impa...    False
Negative
..                                                 ...      ...
...
535  "Unremarkable cinematography that fails to cre...    False
Negative
536         "A cinematic tour de force that enchants."     True
Negative
537  "Flat and unconvincing performances that fail ...    False
Negative
538      "A triumph in storytelling and authenticity."     True
```

```
Negative
539   "A visually striking and emotionally rich film."     True
Negative

    Sentiment_R2 Sentiment_R3
0       Negative     Positive
1       Positive     Negative
2       Negative     Positive
3       Positive     Negative
4       Positive     Negative
..           ...          ...
535     Negative     Negative
536     Negative     Negative
537     Negative     Negative
538     Positive     Positive
539     Negative     Positive

[540 rows x 17 columns]
Column 'R1':
Positive Reviews: 204
Negative Reviews: 336

Column 'R2':
Positive Reviews: 204
Negative Reviews: 336

Column 'R3':
Positive Reviews: 204
Negative Reviews: 336

     Unnamed: 0                       Title  Runtime  Stars  Year  \
0             0           "Love in the Inbox"      126      1  2020
1             1    "Coffee Shop Serendipity"      131      0  2020
2             2   "The Wedding Date Dilemma"      132      4  2000
3             3  "Heartstrings and Highways"      132      1  2015
4             4          "Falling for Cupid"      119      1  2015
..          ...                         ...      ...    ...   ...
535         535              "Shadow Strike"      128      3  2021
536         536              "Riot Protocol"      123      1  2018
537         537          "Deadlock Vendetta"      121      1  2003
538         538     "Blade Runner Protocol"      124      1  2007
539         539            "Eagle Eye Blitz"      126      0  2022

           Budget       Promo  Season Rating            Genre  \
0    6.679387e+07   73.543754  Winter     PG  Romantic Comedy
1    4.667863e+01   33.572003    Fall     PG  Romantic Comedy
2    3.639134e+01   54.561523  Summer     PG  Romantic Comedy
3    9.324732e+01   59.714535  Winter   PG13  Romantic Comedy
4    9.213021e+01   67.643810    Fall   PG13  Romantic Comedy
..            ...         ...     ...    ...              ...
```

```
535  6.489702e+01   91.445593    Fall     PG           Action
536  3.098935e+01   46.045408  Summer      R           Action
537  4.857255e+01   63.660912  Summer     PG           Action
538  1.364682e+02  188.513344  Summer      R           Action
539  1.276156e+02  158.496055  Summer     PG           Action

                                                    R1  \
0    "An unconvincing portrayal of suspense that fa...
1    "A movie that feels disjointed and fails to co...
2    "An underwhelming cinematic effort with unconv...
3    "A film that fails to resonate due to its lack...
4    "A movie that struggles to evoke any genuine e...
..                                                 ...
535  "Weak and contrived dialogue that lacks authen...
536               "A film that lingers in the memory."
537  "A lack of cohesion in the storytelling that m...
538   "An evocative journey that captivates the soul."
539        "A movie that speaks a universal language."

                                                    R2  \
0    "An uninspired plotline that lacks coherence a...
1    "An attempt at humor that lacks cleverness and...
2    "An emotionally resonant movie that connects u...
3    "A beautifully crafted narrative that unfolds ...
4    "A testament to the power of storytelling, lea...
..                                                 ...
535  "An overemphasis on spectacle over substance t...
536       "A celebration of life and its intricacies."
537  "An overly convoluted plot that confuses rathe...
538  "An absence of emotional depth, resulting in a...
539  "Uninspired storytelling that fails to ignite ...

                                                    R3  Success
Sentiment_R1  \
0    "A visually captivating masterpiece that mesme...    False
0
1    "A timeless classic that continues to enchant ...    False
0
2    "A cinematic triumph that surpasses boundaries...    False
0
3    "An uninspired portrayal of drama that feels s...    False
0
4    "An uplifting film that leaves a profound impa...    False
0
..                                                 ...      ...
...
535  "Unremarkable cinematography that fails to cre...    False
0
536         "A cinematic tour de force that enchants."     True
0
```

```
537  "Flat and unconvincing performances that fail ...    False
0
538      "A triumph in storytelling and authenticity."     True
0
539   "A visually striking and emotionally rich film."     True
0

     Sentiment_R2  Sentiment_R3
0               0             1
1               1             0
2               0             1
3               1             0
4               1             0
..            ...           ...
535             0             0
536             0             0
537             0             0
538             1             1
539             0             1

[540 rows x 17 columns]
```

Next, I am going to take care of data error 2 (the clear outliers in the Stars and Runtime column)

```
# first calculate the average run time of movies to replace the 0
minute movie runtimes.
# I don't do anything special for different genres or anything.
average_runtime = df['Runtime'].mean()
print("Average runtime: ", average_runtime)

# then use the lambda function to replace movies with 0 minute runtime
with the average we calculated
df['Runtime'] = df['Runtime'].apply(lambda x: average_runtime if x ==
0 else x)

# next I am going to replace movies that have 100 stars with the
number of stars used in a movie with a similar budget.
def replace_100_stars(row, budget_tolerance=10):  # set 5 as budget
tolerance because (I think) the budgets are in millions of dollars. I
feel like 5 million is close enough
    if row['Stars'] == 100:
        # Find movies with budgets within the specified range
(give/take the budget_tolerance)
        similar_budget_movies = df[
            (df['Budget'] >= (row['Budget'] - budget_tolerance)) &
            (df['Budget'] <= (row['Budget'] + budget_tolerance)) &
            (df['Stars'] != 100)
        ]
        if not similar_budget_movies.empty:
```

```
            return similar_budget_movies['Stars'].mean()
        else:
            # If no similar-budget movie is found, replace with the
average number of stars for all movies
            return df['Stars'].mean()

    return row['Stars']

df['Stars'] = df.apply(replace_100_stars, axis=1)

# Display the updated DataFrame
print(df)
```

```
Average runtime:  130.05835390946504
     Unnamed: 0                        Title  Runtime  Stars  Year  \
0             0           "Love in the Inbox"    126.0    1.0  2020
1             1     "Coffee Shop Serendipity"    131.0    0.0  2020
2             2    "The Wedding Date Dilemma"    132.0    4.0  2000
3             3   "Heartstrings and Highways"    132.0    1.0  2015
4             4           "Falling for Cupid"    119.0    1.0  2015
..          ...                           ...      ...    ...   ...
535         535               "Shadow Strike"    128.0    3.0  2021
536         536               "Riot Protocol"    123.0    1.0  2018
537         537           "Deadlock Vendetta"    121.0    1.0  2003
538         538       "Blade Runner Protocol"    124.0    1.0  2007
539         539            "Eagle Eye Blitz"    126.0    0.0  2022

           Budget       Promo  Season Rating            Genre  \
0    6.679387e+07   73.543754  Winter     PG  Romantic Comedy
1    4.667863e+01   33.572003    Fall     PG  Romantic Comedy
2    3.639134e+01   54.561523  Summer     PG  Romantic Comedy
3    9.324732e+01   59.714535  Winter   PG13  Romantic Comedy
4    9.213021e+01   67.643810    Fall   PG13  Romantic Comedy
..            ...         ...     ...    ...              ...
535  6.489702e+01   91.445593    Fall     PG           Action
536  3.098935e+01   46.045408  Summer      R           Action
537  4.857255e+01   63.660912  Summer     PG           Action
538  1.364682e+02  188.513344  Summer      R           Action
539  1.276156e+02  158.496055  Summer     PG           Action

                                                    R1  \
0    "An unconvincing portrayal of suspense that fa...
1    "A movie that feels disjointed and fails to co...
2    "An underwhelming cinematic effort with unconv...
3    "A film that fails to resonate due to its lack...
4    "A movie that struggles to evoke any genuine e...
..                                                 ...
535  "Weak and contrived dialogue that lacks authen...
536                 "A film that lingers in the memory."
537  "A lack of cohesion in the storytelling that m...
```

```
538   "An evocative journey that captivates the soul."
539        "A movie that speaks a universal language."

                                                    R2  \
0    "An uninspired plotline that lacks coherence a...
1    "An attempt at humor that lacks cleverness and...
2    "An emotionally resonant movie that connects u...
3    "A beautifully crafted narrative that unfolds ...
4    "A testament to the power of storytelling, lea...
..                                                 ...
535  "An overemphasis on spectacle over substance t...
536       "A celebration of life and its intricacies."
537  "An overly convoluted plot that confuses rathe...
538  "An absence of emotional depth, resulting in a...
539  "Uninspired storytelling that fails to ignite ...

                                                    R3  Success  
Sentiment_R1  \
0    "A visually captivating masterpiece that mesme...    False  
0
1    "A timeless classic that continues to enchant ...    False  
0
2    "A cinematic triumph that surpasses boundaries...    False  
0
3    "An uninspired portrayal of drama that feels s...    False  
0
4    "An uplifting film that leaves a profound impa...    False  
0
..                                                 ...      ...  
...
535  "Unremarkable cinematography that fails to cre...    False  
0
536         "A cinematic tour de force that enchants."     True  
0
537  "Flat and unconvincing performances that fail ...    False  
0
538      "A triumph in storytelling and authenticity."     True  
0
539   "A visually striking and emotionally rich film."     True  
0

     Sentiment_R2  Sentiment_R3
0               0             1
1               1             0
2               0             1
3               1             0
4               1             0
..            ...           ...
535             0             0
536             0             0
```

```
537              0             0
538              1             1
539              0             1

[540 rows x 17 columns]
```

Next, I will tackle data error 3 (accounting for inflation)

```
import numpy as np

# first print the value counts of each year in the dataframe so I know
which years i have to account for inflation.
year_counts = df['Year'].value_counts().reset_index()
year_counts.columns = ['Year', 'Count']
sorted_movies_df = year_counts.sort_values(by='Year')

print(sorted_movies_df)
# now I can see that I have to adjust for inflation for years 2000-
2023

# I get my factors of inflation from this website:
https://www.in2013dollars.com/us/inflation/2003?amount=1
inflation_factors = {
    # I'm sure there is a much more elegant way to do this but I don't
know much about inflation curves so to be save I will do it like this
    2000: 1.79,
    2001: 1.74,
    2002: 1.71,
    2003: 1.67,
    2004: 1.63,
    2005: 1.58,
    2006: 1.53,
    2007: 1.48,
    2008: 1.43,
    2009: 1.43,
    2010: 1.41,
    2011: 1.37,
    2012: 1.34,
    2013: 1.32,
    2014: 1.30,
    2015: 1.30,
    2016: 1.28,
    2017: 1.26,
    2018: 1.22,
    2019: 1.20,
    2020: 1.19,
    2021: 1.14,
    2022: 1.05,
    2023: 1.0,  # No inflation adjustment needed for 2023
```

```
}

# Define the range of years you want to adjust for
years_to_adjust = range(2000, 2024)

# tterate through each year and adjust 'Budget' and 'Promo' columns
depending on the year
for year in years_to_adjust:
    if year in inflation_factors:
        inflation_factor = inflation_factors[year]
        # update budget column
        df.loc[df['Year'] == year, 'Budget'] *= inflation_factor
        # update promo column
        df.loc[df['Year'] == year, 'Promo'] *= inflation_factor

# I noticed the budget column has some crazy outliers to I am going to
transform it to reduce the impact of the super high values
df['Budget'] = np.log1p(df['Budget'])

# check data frame
print(df)
```

```
    Year  Count
19  2000     17
18  2001     18
0   2002     37
13  2003     20
9   2004     22
22  2005     15
11  2006     22
3   2007     30
4   2008     30
17  2009     18
12  2010     20
15  2011     19
6   2012     26
23  2013     13
7   2014     26
5   2015     27
14  2016     20
21  2017     16
8   2018     22
2   2019     30
10  2020     22
16  2021     19
1   2022     34
20  2023     17
     Unnamed: 0                        Title  Runtime  Stars  Year
Budget  \
```

```
0              0          "Love in the Inbox"    126.0    1.0  2020
18.365028
1              1    "Coffee Shop Serendipity"    131.0    0.0  2020
4.206208
2              2   "The Wedding Date Dilemma"    132.0    4.0  2000
4.767302
3              3  "Heartstrings and Highways"    132.0    1.0  2015
5.066309
4              4          "Falling for Cupid"    119.0    1.0  2015
5.054333
..           ...                          ...      ...    ...   ...
...
535          535              "Shadow Strike"    128.0    3.0  2021
4.446645
536          536              "Riot Protocol"    123.0    1.0  2018
3.852794
537          537          "Deadlock Vendetta"    121.0    1.0  2003
4.916061
538          538     "Blade Runner Protocol"    124.0    1.0  2007
5.703515
539          539            "Eagle Eye Blitz"    126.0    0.0  2022
4.953686

          Promo  Season Rating            Genre  \
0    104.145310  Winter     PG  Romantic Comedy
1     47.541313    Fall     PG  Romantic Comedy
2    174.820577  Summer     PG  Romantic Comedy
3    100.917564  Winter   PG13  Romantic Comedy
4    114.318039    Fall   PG13  Romantic Comedy
..          ...     ...    ...              ...
535  118.842692    Fall     PG           Action
536   68.533985  Summer      R           Action
537  177.543919  Summer     PG           Action
538  412.919629  Summer      R           Action
539  174.741900  Summer     PG           Action

                                                    R1  \
0    "An unconvincing portrayal of suspense that fa...
1    "A movie that feels disjointed and fails to co...
2    "An underwhelming cinematic effort with unconv...
3    "A film that fails to resonate due to its lack...
4    "A movie that struggles to evoke any genuine e...
..                                                 ...
535  "Weak and contrived dialogue that lacks authen...
536               "A film that lingers in the memory."
537  "A lack of cohesion in the storytelling that m...
538   "An evocative journey that captivates the soul."
539        "A movie that speaks a universal language."

                                                    R2  \
```

```
0    "An uninspired plotline that lacks coherence a...
1    "An attempt at humor that lacks cleverness and...
2    "An emotionally resonant movie that connects u...
3    "A beautifully crafted narrative that unfolds ...
4    "A testament to the power of storytelling, lea...
..                                                 ...
535  "An overemphasis on spectacle over substance t...
536       "A celebration of life and its intricacies."
537  "An overly convoluted plot that confuses rathe...
538  "An absence of emotional depth, resulting in a...
539  "Uninspired storytelling that fails to ignite ...

                                                    R3  Success
Sentiment_R1  \
0    "A visually captivating masterpiece that mesme...    False
0
1    "A timeless classic that continues to enchant ...    False
0
2    "A cinematic triumph that surpasses boundaries...    False
0
3    "An uninspired portrayal of drama that feels s...    False
0
4    "An uplifting film that leaves a profound impa...    False
0
..                                                 ...      ...
...
535  "Unremarkable cinematography that fails to cre...    False
0
536          "A cinematic tour de force that enchants."     True
0
537  "Flat and unconvincing performances that fail ...    False
0
538       "A triumph in storytelling and authenticity."     True
0
539   "A visually striking and emotionally rich film."     True
0

     Sentiment_R2  Sentiment_R3
0               0             1
1               1             0
2               0             1
3               1             0
4               1             0
..            ...           ...
535             0             0
536             0             0
537             0             0
538             1             1
539             0             1
```

```
[540 rows x 17 columns]
```

Data Exploration

Does Season have a stastically significant impact on a movie's success?

p-value: 0.005716268505111858

```
# we will do a chi squared test since the seasons are different
categories
from scipy.stats import chi2_contingency

contingency_table = pd.crosstab(df['Season'], df['Success'])
chi2, p, dof, expected = chi2_contingency(contingency_table)

# i made my significance level .05 because that is commonly chosen
significance level
alpha = 0.05

# Interpret the results
if p < alpha:
    print("There is a statistically significant association between
season and success.")
    print(p)
else:
    print("There is no statistically significant association between
season and success.")
    print(p)
```

```
There is a statistically significant association between season and
success.
0.005716268505111858
```

Do seasons have a statistically significant difference in their distribution of content ratings?

p-value: 0.21507814500508263

```
# we will do naother chi squared test since we are still doing seasons
contingency_table = pd.crosstab(df['Season'], df['Rating'])

chi2, p, _, _ = chi2_contingency(contingency_table)
alpha = 0.05
if p < alpha:
    print("There is a statistically significant difference between
Season and Rating.")
    print(p)
else:
```

```
    print("There is no statistically significant difference between
Season and Rating.")
    print(p)
```

```
There is no statistically significant difference between Season and
Rating.
0.21507814500508263
```

Who is the harshest critic (highest precent of negative reviews)?

Critic: Reviewer one is the harshest critic. They have the highest percent of negative reviews.

```
# keep count of number of negative reviews (according to sentiment
analyzer)
R1_count = 0
R2_count = 0
R3_count = 0

max_negative_count = 0
total_reviews = len(df)

# iterate through each row of the dataframe and increment counts when
appropriate
for index, row in df.iterrows():
    # Count negative reviews (0 values) for each reviewer
    R1_count += row['Sentiment_R1'] == 0
    R2_count += row['Sentiment_R2'] == 0
    R3_count += row['Sentiment_R3'] == 0

    # Calculate the total count of negative reviews for the current
critic
    total_negative_count = R1_count + R2_count + R3_count

    # Check if the current critic has more negative reviews than the
previous maximum
    if total_negative_count > max_negative_count:
        max_negative_count = total_negative_count
        harshest_critic = index  # Use the index (row number) as a
pseudo-critic name

# Print the harshest critic and their counts for each reviewer
print("Percentage of negative reviews for each reviewer:")
print("R1:", (R1_count/total_reviews) * 100, "%")
print("R1:", (R2_count/total_reviews) * 100, "%")
print("R1:", (R3_count/total_reviews) * 100, "%")
```

```
Percentage of negative reviews for each reviewer:
R1: 79.81481481481481 %
R1: 69.62962962962963 %
R1: 62.22222222222222 %
```

What is the covariance between promotional budget and the filming budget?

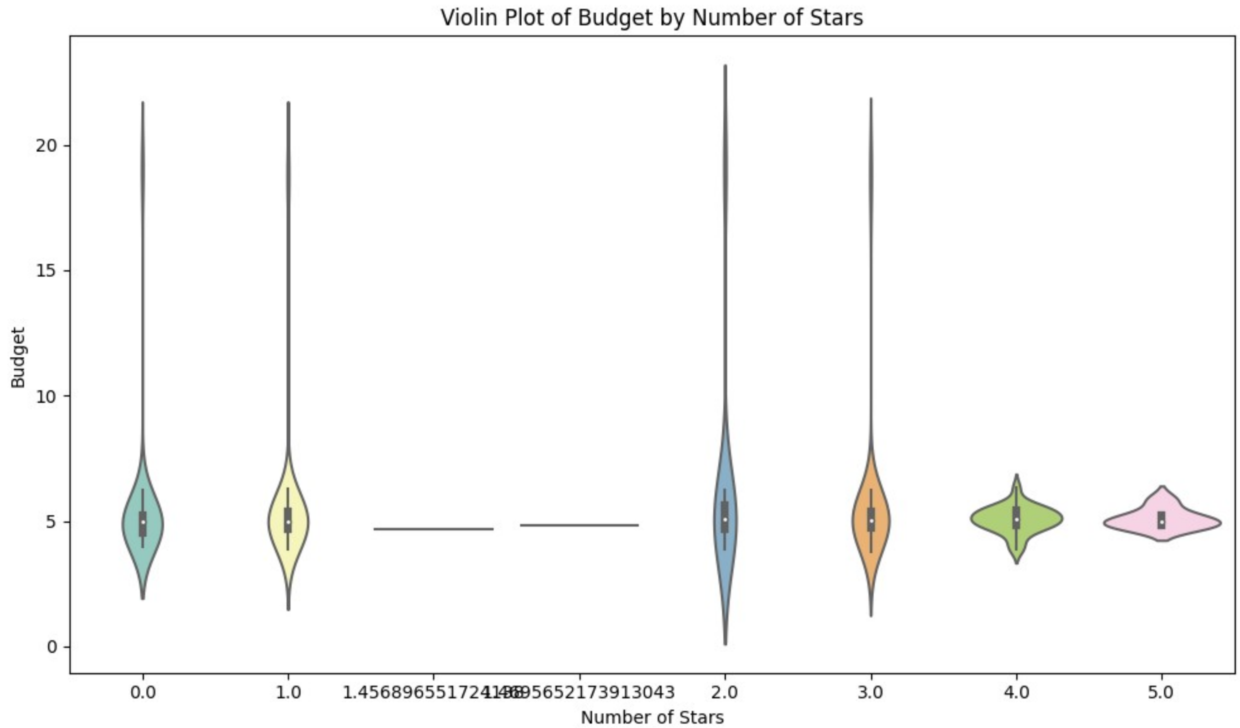
Cov: 56.423789973820185

```
covariance = df['Promo'].cov(df['Budget'])
print("Covariance of promo and budget columns: ", covariance)

Covariance of promo and budget columns:  56.423789973820185
```

Data Visualization

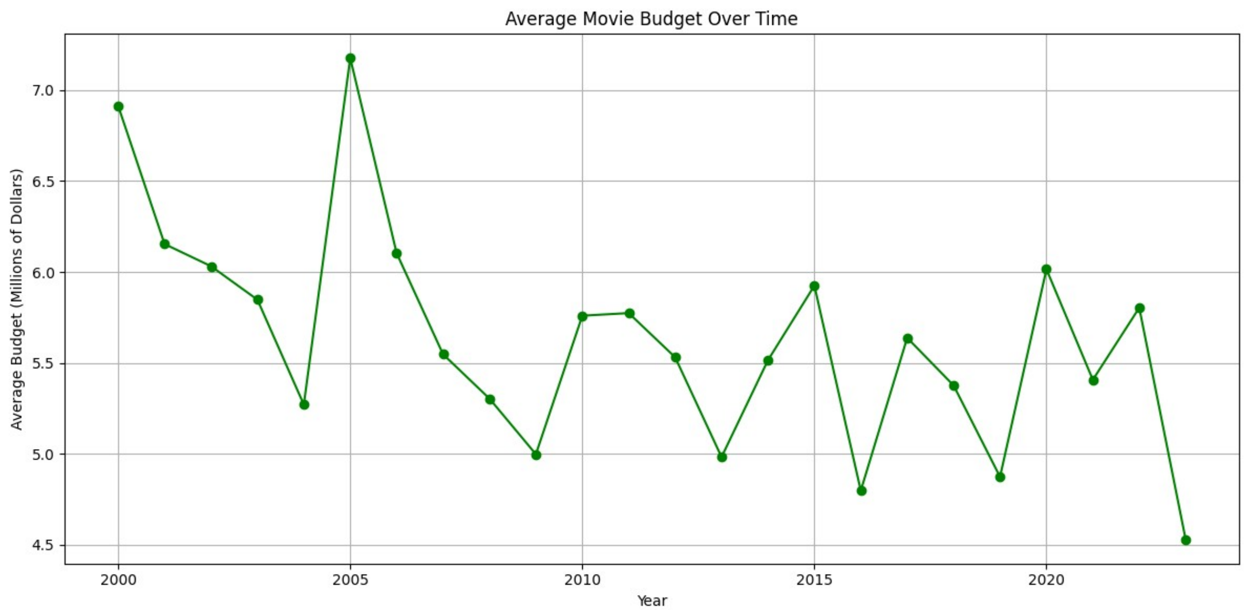
Create a chart that compares the distribution of the budget for each different number of stars. (It does not need to be particularly appealing.

```
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))
sns.violinplot(data=df, x='Stars', y='Budget', palette='Set3')
plt.title('Violin Plot of Budget by Number of Stars')
plt.xlabel('Number of Stars')
plt.ylabel('Budget')
plt.tight_layout()
plt.show()

# this isn't a very pretty graph I know. Also please remember that I
transformed the budget column!
# so this graph more so gives us a generatl relationship idea of
budget/stars
```



Create a graph showing the average movie budget over time.

```
average_budget_by_year = df.groupby('Year')['Budget'].mean()

# Create a line plot to show the average movie budget over time
plt.figure(figsize=(12, 6))  # Adjust the figure size as needed
plt.plot(average_budget_by_year.index, average_budget_by_year.values,
marker='o', linestyle='-', color='g')
plt.title('Average Movie Budget Over Time')
plt.xlabel('Year')
plt.ylabel('Average Budget (Millions of Dollars)')
plt.grid(True)
plt.tight_layout()
plt.show()
```



Feature Engineering

List any features you choose to create (if you are creating many features based on one column, you do not need to list them separately.) You are not required to create any features if you do not wish to. You may create any number of additional features.

I made lots of new columns and features because I was very afraid to modify the original data.

- I made new columns (EX: Sentiment_R1) for each reviewer where I encoded the sentiment I got, so reviews that were analyzed to be Negative were represented by a 0 and those that were analyzed to be Positive were represented by a 1.
- Another new column I made was AverageSentiment that took the average of the 3 new columns I made in the previous bullet point which made it easier to see what the overal sentiment towards the movie.
- There were also many columns I edited (runtime, stars, etc)
- I also made the year counts columns to see how many movies were released in each year and see what years were included in the data frame so I knew which years I had to get the inflation factor for.

For Modeling/Testing, I won't use the R1, R2, and R3 columns since i have the sentiment versions with integers for that. So I don't have to do any encoding.

Since we know seasons don't have a big impact on success, I won't include that either.

Since there are many categories for rating and genre, I won't include them because one hot encoding isn't good for a column with many categories.

Modeling

Create a model of your choice.

Model type choosen: Random forest

```
print(df.dtypes)

# going to make a copy of the dataframe with the columns I want to
include for the testing and stuff
selected_columns = ['Title',
                    'Runtime',
                    'Stars',
                    'Year',
                    'Budget',
                    'Promo',
                    'Success',
                    'Sentiment_R1',
                    'Sentiment_R2',
                    'Sentiment_R3',
                    'AverageSentiment']

selected_df = df[selected_columns].copy()
print(selected_df.head())
```

```
Unnamed: 0            int64
Title                object
Runtime             float64
Stars               float64
Year                  int64
Budget              float64
Promo               float64
Season               object
Rating               object
Genre                object
R1                   object
R2                   object
R3                   object
Success                bool
Sentiment_R1          int64
Sentiment_R2          int64
Sentiment_R3          int64
AverageSentiment    float64
dtype: object
                         Title  Runtime  Stars  Year     Budget
Promo  \
0          "Love in the Inbox"    126.0    1.0  2020  18.365028
104.145310
1     "Coffee Shop Serendipity"    131.0    0.0  2020   4.206208
```

```
47.541313
2   "The Wedding Date Dilemma"    132.0    4.0  2000   4.767302
174.820577
3  "Heartstrings and Highways"    132.0    1.0  2015   5.066309
100.917564
4          "Falling for Cupid"    119.0    1.0  2015   5.054333
114.318039
```

	Success	Sentiment_R1	Sentiment_R2	Sentiment_R3	AverageSentiment
0	False	0	0	1	0.333333
1	False	0	1	0	0.333333
2	False	0	0	1	0.333333
3	False	0	1	0	0.333333
4	False	0	1	0	0.333333

Testing

Shuffle your data and break it into a 10% test set and 90% training set. Show your model's accuracy on the test set. In order to get full credit, the model's accuracy must be higher than 50%.

Model accuracy: Before adjusting threshold in next step: 0.8148148148148148

After adjusting the threshold in the next step (accuracy is still > 50%): 0.7962962962962963

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

X = selected_df.drop(columns=['Success', 'Title'])  # all the columns
to be features (everything except for success)
y = selected_df['Success']  # target !!

# splitting. I think this shuffles for me according to piazza.
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.10, random_state=40)

# make my random forest classifier
clf = RandomForestClassifier(random_state=40)
clf.fit(X_train, y_train)

# makin my predictions
```

```
y_pred = clf.predict(X_test)

# calculate the accuracy of the model on the test set
accuracy = accuracy_score(y_test, y_pred)

# check if the accuracy is higher than 50%
if accuracy > 0.50:
    print(f"Model Accuracy: ", accuracy)
else:
    print(f"Model Accuracy: ", accuracy)
```

```
Model Accuracy:  0.8148148148148148
```

Show the confusion matrix for your model. To get full credit, your false positive rate and false negative rate must be under 30%.

(since one of my rates wasn't under 30, i had to change threshold)

False negative rate: 0.25

False positive rate: 0.19047619047619047

```
from sklearn.metrics import confusion_matrix

# this was the first threshold so like in the last part.
y_pred_default = clf.predict(X_test)

accuracy_default = accuracy_score(y_test, y_pred_default)

confusion_default = confusion_matrix(y_test, y_pred_default)

true_negatives_default, false_positives_default,
false_negatives_default, true_positives_default =
confusion_default.ravel()
false_positive_rate_default = false_positives_default /
(false_positives_default + true_negatives_default)
false_negative_rate_default = false_negatives_default /
(false_negatives_default + true_positives_default)

print("Results BEFORE changing threshold:")
print(f"Accuracy: ", accuracy_default)
print(f"False Positive Rate: ", false_positive_rate_default)
print(f"False Negative Rate: ", false_negative_rate_default)

# lowering the threshold because false negative rate is too high
threshold = 0.3
y_pred_adjusted = (clf.predict_proba(X_test)[:, 1] >
threshold).astype(int)

# do accuracy again
accuracy_adjusted = accuracy_score(y_test, y_pred_adjusted)
```

```
confusion_adjusted = confusion_matrix(y_test, y_pred_adjusted)
true_negatives_adjusted, false_positives_adjusted,
false_negatives_adjusted, true_positives_adjusted =
confusion_adjusted.ravel()
false_positive_rate_adjusted = false_positives_adjusted /
(false_positives_adjusted + true_negatives_adjusted)
false_negative_rate_adjusted = false_negatives_adjusted /
(false_negatives_adjusted + true_positives_adjusted)

print("New results after changing threshold for classification:")
print(f"Accuracy: ", accuracy_adjusted)
print(f"False Positive Rate: ", false_positive_rate_adjusted)
print(f"False Negative Rate: ", false_negative_rate_adjusted)

Results BEFORE changing threshold:
Accuracy:  0.8148148148148148
False Positive Rate:  0.07142857142857142
False Negative Rate:  0.5833333333333334
New results after changing threshold for classification:
Accuracy:  0.7962962962962963
False Positive Rate:  0.19047619047619047
False Negative Rate:  0.25
```

What was the most important feature for your model? Don't guess, either look up how to check or do your own tests.

Source: https://scikit-learn.org/stable/auto_examples/ensemble/plot_forest_importances.html

Most important feature: Budget

```
feature_importances = clf.feature_importances_

feature_names = X_train.columns
most_important_idx = feature_importances.argmax()
most_important_feature = feature_names[most_important_idx]
most_important_importance = feature_importances[most_important_idx]

# printing the most important feature and its importance score
print("Most Important Feature: ", most_important_feature)
print("Importance Score: ", most_important_importance)

Most Important Feature:  Budget
Importance Score:  0.211644383883012
```