COGS 108 - Predicting Review Scores and Satisfaction of Books from Amazon

Permissions

Place an `X` in the appropriate bracket below to specify if you would like your group's project to be made available to the public. (Note that student names will be included (but PIDs will be scraped from any groups who include their PIDs).

- [x] YES - make available
- [ ] NO - keep private

Video: https://www.youtube.com/watch?v=VUi9a28-zEI

Names

- Bryan Ung
- Gunju Kim
- Asif Mahdin
- ChaoYuan Lin
- Merak Olson

Abstract

This project aims to predict book review scores (out of 5 stars) based on various factors such as price, year of publication, and categories, using U.S. Amazon reviews from 2010 to 2023. We explored multiple modeling approaches, starting with a basic vectorizer paired with linear regression, followed by TF-IDF with Support Vector Machine (SVM), and finally, sentiment analysis using BERT combined with a Random Forest regressor.

Our findings indicate that the initial linear regression model struggled with the complexity of the data, resulting in poor predictive performance. The TF-IDF and SVM model showed improvement, but it was the BERT-based sentiment analysis and Random Forest model that provided the best results, significantly reducing error metrics and increasing the R^2 score.

Despite these advancements, limitations such as dataset inconsistencies and hardware constraints impacted our analysis. Reviewers often provided ratings that did not align with their written feedback, and using full reviews instead of summaries with BERT was computationally challenging.

Overall, this study demonstrates the potential of advanced NLP and machine learning techniques in predicting consumer ratings, offering valuable insights for the publishing industry to enhance marketing

strategies and improve reader satisfaction.

Research Question

- Can we predict the review scores (out of 5 stars) of books based on factors such as prices, year of publication, and categories based on U.S. Amazon reviews from 2010 to 2023?

Variable

1. **Price Ranges**: Continue with the segmented price categories:
   - Low-priced: $0-15
   - Mid-priced: $16-21
   - High-priced: $22+
2. **Time Range**: The period from 2010 to 2023 remains suitable for capturing a wide range of market behaviors and consumer trends.
3. **Geographic Focus**: Specify that the analysis is limited to reviews on the U.S. Amazon site, which helps control for regional variations in purchasing behavior and product availability.
4. **Type of Reviews**:
   - **Verified Purchases**: Limiting the dataset to verified purchases can reduce noise from potentially fake reviews or those not based on actual product use.
   - **Sentiment Ratio**: The primary measure will be the ratio of positive reviews (4 or 5 stars) to negative reviews (1 or 2 stars). This is a direct indicator of customer satisfaction and dissatisfaction.
5. **Additional Variables**:
   - **Number of Comments**: Including the number of comments per review could give insights into the level of engagement or controversy a product might elicit. More comments can indicate higher interest or issues with the product.
   - **Review Helpfulness Votes**: Analyze how many times a review was voted helpful as this might correlate with the persuasiveness or relevance of the review content.

Background and Prior Work

Amazon is one of the most popular online shopping websites and contains a wide variety of products. Books, one of the most popular categories on Amazon, have prices that range from around $1 to $21+. Because of this wide range of prices, we want to see what kind of relationship the prices have with customer satisfaction.

In “Effect of Price on Firm Reputation”, Luca and Reshef collected data from Yelp and found that the average rating for restaurants in the cheapest category is 3.4. For the more expensive restaurants, they found that the average rating is 3.6. This shows that ratings can be used to predict the quality and price of restaurants (Luca & Reshef, 2020). They also found that increasing 1% of the price results in a 2.5%–5% decrease in ratings. This shows that customers use prices as one of the factors to assess their satisfaction with the product.

However, in the category of books, the price of the product might not correlate with satisfaction with the product. In the article, “Of book reviews and selection bias,” Professor Andrew Gelman talks on his experiences writing book reviews or, more specifically, the times he does not. As he puts it, most of his reviews are positive simply because the fact that he did not enjoy a book does not always have to do with the quality of the book; also, his interests and expertise would make him unable to review without bias (Gelman, 2020). So not only could many books have a positive bias because of personal interests, but negative reviews could be just as biased based on a lack of interest in or understanding of the topic of the book.

Another source that may be relevant to our research question is "High price, higher satisfaction, except when selling books: A defense of Hugh Howey" by Derek Murphy. This article brings the case where low priced books from unpopular publishers have more positive review than the higher priced books from Big five publishers (Murphy, 2024). In the article, Derek suggests that customer satisfaction may not necessarily correlate with pricing when it comes to books, providing a useful perspective for our analysis (Murphy, 2024).

1. ^ Luca, M., & Reshef, O. (2020). The Effect of Price on Firm Reputation. https://sist.sathyabama.ac.in/sist_naac/documents/1.3.4/39280062 KEERTHANA T.pdf
2. ^ Gelman, A. (2020, January 6). Of book reviews and selection bias | statistical modeling, causal inference, and social science. https://statmodeling.stat.columbia.edu/2020/01/06/of-book-reviews-and-selection-bias/
3. ^ Murphy, Derek. (n.d.). High price, higher satisfaction, except when selling books: A defense of Hugh Howey. https://www.creativindie.com/high-price-higher-satisfaction-except-when-selling-books-a-defense-of-hugh-howey/

Hypothesis

We hypothesize that the review score of a book can be predicted based on variables such as price, year of publication, review text, and categories based on U.S. Amazon reviews from 2010 to 2023. We aim to identify which of these factors most significantly influence a book's rating.

We predict that there is a strong relationship between the review score of a book, and other factors such as price, year of publication, review text, and categories. For prices, year of publication, review text, and categories people set different expectations for the book they purchase.

Data

Data overview

- Dataset #1
  - Dataset Name: Amazon Book Ratings
  - Link to the dataset: https://www.kaggle.com/datasets/mohamedbakhet/amazon-books-reviews
  - Number of observations: 300000
  - Number of variables: 10
- Dataset #2 (if you have more than one!)
  - Dataset Name: Amazon Book Data
  - Link to the dataset: https://www.kaggle.com/datasets/mohamedbakhet/amazon-books-reviews
  - Number of observations: 212404
  - Number of variables: 10

Key Variable	Description
title	Title of the book
price	The prices of the book the customer bought on Amazon in US dollars.
review/helpfulness	The percentage of people who found this review helpful from 0 to 1. A value of 0 shows that no one found this customer's review helpful. A value of 1 shows that among the people who rate the helpfulness of the customer's review, all of them found it helpful.
review/score	The customer's rating of the book from 0 to 5. 0 indicates the customer's dissatisfaction while 5 shows the customer's satisfaction of the book.
review/summary	The summary of a customer's review when buying a specific book on Amazon.
review/text	The content or message of a customer's review when buying a specific book on Amazon.
authors	A list of names who wrote the book.
publishedYear	The year in which the book was published.
categories	Describes the categories of the book.
ratingsCount	The total number of reviews the book has.

In the Amazon book reviews dataset, there are title(String), price(float) in dollars, review score(float), and review text(String). This dataset allows us to compare the prices with the ratio of positive and negative reviews. To clean this dataset, we would remove rows that does not contain any value in the price and text column. Since we want to enforce privacy onto our dataset, we are also removing the column that include the reviewer's name. We dropped review/time column, User_id, and Id columns as they are are not related to our research question.

In the Amazon Book Dataset, we have many variables linked directly to the book, including things like genre, total review count, and even the authors. This kind of data could be used for certain small comparisons, like seeing if an author with a good/bad reputation or certain genres may show bias in the overall review count and ratio. Having a total review count is also very important, giving us a percentage count for good and bad reviews as well as a ratio. To clean this dataset, we drop the image, previewLink, publisher, authors, and infoLink columns as they are not related to our research question.

Setup

In [39]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
import torch
from transformers import BertTokenizer, BertForSequenceClassification, pipeline

from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score, classific
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler, OneHotEncoder, MultiLabelBinarizer

import re
import patsy
import statsmodels.api as sm
import scipy.stats as stats

import warnings
warnings.filterwarnings('ignore')
```

In [4]:
```
nltk.download('stopwords')
nltk.download('punkt')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/bryanung/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to /Users/bryanung/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

Out[4]:
```
True
```

Cleaning `Books_rating.csv`

In [40]:
```
# Read data into the dataframe
rating = pd.read_csv('Books_rating.csv')
rating.head()
```

Out[40]:

	Id	Title	Price	User_id	profileName	review/helpfulness	review/score	review/tir
0	1882931173	Its Only Art If Its Well Hung!	NaN	AVCGYZL8FQQTD	Jim of Oz "jim-of-oz"	7/7	4.0	9406368
1	0826414346	Dr. Seuss: American Icon	NaN	A30TK6U7DNS82R	Kevin Killian	10/10	5.0	10957248
2	0826414346	Dr. Seuss: American Icon	NaN	A3UH4UZ4RSVO82	John Granger	10/11	5.0	10787904
3	0826414346	Dr. Seuss: American Icon	NaN	A2MVUWT453QH61	Roy E. Perry "amateur philosopher"	7/7	4.0	10907136
4	0826414346	Dr. Seuss: American Icon	NaN	A22X4XUPKF66MR	D. H. Richards "ninthwavestore"	3/3	4.0	11079936

In [41]:
```
print('Rating size: ', len(rating))
```

```
Rating size:  3000000
```

Drop all the rows that does not have a numerical price from `rating`

In [42]:
```
rating = rating.dropna(subset=['Price'])
print('Rating size: ', len(rating))
```

```
Rating size:  481171
```

```
In [43]: rating.head()
```

Out[43]:

	Id	Title	Price	User_id	profileName	review/helpfulness	review/score	review/time
10	0829814000	Wonderful Worship in Smaller Churches	19.40	AZ0IOBU20TBOP	Rev. Pamela Tinnin	8/10	5.0	991440000
11	0829814000	Wonderful Worship in Smaller Churches	19.40	A373VVEU6Z9M0N	Dr. Terry W. Dorsett	1/1	5.0	1291766400
12	0829814000	Wonderful Worship in Smaller Churches	19.40	AGKGOH65VTRR4	Cynthia L. Lajoy "Cindy La Joy"	1/1	5.0	1248307200
13	0829814000	Wonderful Worship in Smaller Churches	19.40	A3OQWLU31BU1Y	Maxwell Grant	1/1	5.0	1222560000
14	0595344550	Whispers of the Wicked Saints	10.95	A3Q12RK71N74LB	Book Reader	7/11	1.0	1117065600

Drop the unnecessary columns (User_id, profileName, Id, review/time)

```
In [44]: rating.drop(['User_id', 'profileName', 'Id', 'review/time'], axis=1, inplace=True)
         rating.head()
```

Out[44]:

	Title	Price	review/helpfulness	review/score	review/summary	review/text
10	Wonderful Worship in Smaller Churches	19.40	8/10	5.0	Outstanding Resource for Small Church Pastors	I just finished the book, &quot;Wonderful Wors...
11	Wonderful Worship in Smaller Churches	19.40	1/1	5.0	Small Churches CAN Have Wonderful Worship	Many small churches feel like they can not hav...
12	Wonderful Worship in Smaller Churches	19.40	1/1	5.0	Not Just for Pastors!	I just finished reading this amazing book and ...
13	Wonderful Worship in Smaller Churches	19.40	1/1	5.0	Small church pastor? This is the book on worship	I hadn't been a small church pastor very long ...
14	Whispers of the Wicked Saints	10.95	7/11	1.0	not good	I bought this book because I read some glowing...

```
In [45]: rating.shape
```

```
Out[45]: (481171, 6)
```

Cleaning `books_data.csv`

Read the data into the dataframe

In [46]:
```
data = pd.read_csv('books_data.csv')
data.head()
```

Out[46]:

	Title	description	authors	image	previewLink	publisher
0	Its Only Art If Its Well Hung!	NaN	['Julie Strain']	http://books.google.com/books/content?id=DykPA...	http://books.google.nl/books?id=DykPAAAACAAJ&d...	NaN
1	Dr. Seuss: American Icon	Philip Nel takes a fascinating look into the k...	['Philip Nel']	http://books.google.com/books/content?id=IjvHQ...	http://books.google.nl/books?id=IjvHQsCn_pgC&p...	A&C Black
2	Wonderful Worship in Smaller Churches	This resource includes twelve principles in un...	['David R. Ray']	http://books.google.com/books/content?id=2tsDA...	http://books.google.nl/books?id=2tsDAAAACAAJ&d...	NaN
3	Whispers of the Wicked Saints	Julia Thomas finds her life spinning out of co...	['Veronica Haddon']	http://books.google.com/books/content?id=aRSIg...	http://books.google.nl/books?id=aRSIgJlq6JwC&d...	iUniverse
4	Nation Dance: Religion, Identity and Cultural ...	NaN	['Edward Long']	NaN	http://books.google.nl/books?id=399SPgAACAAJ&d...	NaN

In [47]:
```
print('Data size: ', len(data))
```

```
Data size:  212404
```

Drop all the rows that does not have a numerical value in `ratingCount` and `categories`

In [48]:
```
data = data.dropna(subset=['ratingsCount', 'categories'])
print('Data size: ', len(data))
```

```
Data size:  47797
```

In [49]:
```
data.head()
```

Out[49]:

	Title	description	authors	image	previewLink	publi
5	The Church of Christ: A Biblical Ecclesiology ...	In The Church of Christ: A Biblical Ecclesiolo...	['Everett Ferguson']	http://books.google.com/books/content?id=kVqRa...	http://books.google.nl/books?id=kVqRaiPlx88C&p...	W Eerdı Publis
31	Voices from the Farm: Adventures in Community ...	Twenty-five years ago, at the height of the co...	['Rupert Fike']	http://books.google.com/books/content?id=IjTAB...	http://books.google.nl/books?id=IjTABgAAQBAJ&p...	Publis Com
33	The Battleship Bismarck	The Bismarck is perhaps the most	['Stefan Draminski']	http://books.google.com/books/content?id=nxttD...	http://books.google.nl/books?id=nxttDwAAQBAJ&p...	Blooms Publis

		famous – and ...				
42	Tess and the Highlander	In 1543, on a windswept isle off of Scotland, ...	['May Mcgoldrick']	http://books.google.com/books/content?id=VmCRS...	http://books.google.nl/books?id=VmCRSPmY3WkC&d...	Ha C
43	Beginner's Yoruba (Hippocrene Beginner's Series)	"Beginner's Yoruba" is now available with two ...	['Kayode J. Fakinlede']	http://books.google.com/books/content?id=xLe4n...	http://books.google.nl/books?id=xLe4nWzeSw0C&p...	Hippoc B

Drop the unnecessary columns (image, previewLink, publisher, authors, infoLink)

In [50]:
```
data.drop(['image', 'previewLink', 'publisher', 'infoLink', 'description'], axis=1, inpl
data.head()
```

Out[50]:

	Title	authors	publishedDate	categories	ratingsCount
5	The Church of Christ: A Biblical Ecclesiology ...	['Everett Ferguson']	1996	['Religion']	5.0
31	Voices from the Farm: Adventures in Community ...	['Rupert Fike']	2012-08-21	['Biography & Autobiography']	1.0
33	The Battleship Bismarck	['Stefan Draminski']	2018-09-20	['History']	1.0
42	Tess and the Highlander	['May Mcgoldrick']	2002-11	['Juvenile Fiction']	2.0
43	Beginner's Yoruba (Hippocrene Beginner's Series)	['Kayode J. Fakinlede']	2005	['Foreign Language Study']	1.0

In [51]:
```
data.shape
```

Out[51]:
```
(47797, 5)
```

Standardized publishedData to year only and change `publishedDate` to `publishYear`

In [52]:
```
def standardized_year(string):
    string = string.strip()
    if len(string) >= 4:
        return string[:4]
    return string

def review_to_float(string):
    string = string.strip()
    lst = string.split('/')
    if float(lst[0]) == 0 or float(lst[1]) == 0:
        return 0.0
    return float(lst[0]) / float(lst[1])
```

In [53]:
```
data['publishedDate'] = data['publishedDate'].astype(str)
data['publishedDate'] = data['publishedDate'].apply(standardized_year)
data.rename(columns={'publishedDate' : 'publishedYear'}, inplace=True)
data.head()
```

Out[53]:

	Title	authors	publishedYear	categories	ratingsCount

5	The Church of Christ: A Biblical Ecclesiology ...	['Everett Ferguson']	1996	['Religion']	5.0
31	Voices from the Farm: Adventures in Community ...	['Rupert Fike']	2012	['Biography & Autobiography']	1.0
33	The Battleship Bismarck	['Stefan Draminski']	2018	['History']	1.0
42	Tess and the Highlander	['May Mcgoldrick']	2002	['Juvenile Fiction']	2.0
43	Beginner's Yoruba (Hippocrene Beginner's Series)	['Kayode J. Fakinlede']	2005	['Foreign Language Study']	1.0

In [54]:
```
data.shape
```
Out[54]:
```
(47797, 5)
```

In [55]:
```
data['publishedYear'].unique()
```
Out[55]:
```
array(['1996', '2012', '2018', '2002', '2005', '2003', '1981', '1984',
       '1992', '1986', '1987', '2016', '1973', '1990', '2004', '1895',
       '1988', '1962', '2010', '2008', '1989', '2000', '2017', '2015',
       '1967', '2013', '1859', '1970', '2014', '2007', '1997', '1969',
       '2009', '1966', '2020', 'nan', '1995', '2019', '1977', '1878',
       '2011', '1979', '1994', '1976', '1993', '1985', '1998', '1991',
       '1975', '2001', '2021', '1861', '2006', '1944', '1983', '1982',
       '1953', '1999', '1978', '1889', '1951', '1980', '1961', '1972',
       '1963', '1863', '1956', '1936', '2022', '1957', '1937', '1946',
       '1958', '1971', '1960', '1952', '1887', '1964', '1890', '1916',
       '1883', '1867', '1974', '1923', '1947', '1949', '1931', '1965',
       '1929', '1853', '1925', '1913', '1928', '1922', '1955', '1939',
       '1869', '1845', '1891', '1776', '1903', '1968', '1885', '1835',
       '1959', '1900', '1893', '1950', '1873', '1941', '1871', '1866',
       '1940', '1942', '1852', '1843', '1927', '1818', '1908', '1865',
       '1872', '1920', '1919', '1825', '1934', '19??', '1894', '1935',
       '1860', '1907', '1844', '1810', '1924', '1933', '1954', '1750',
       '1910', '1896', '1899', '1915', '1921', '1918', '1911', '1851',
       '1884', '1897', '1855', '1870', '1914', '1839', '1841', '1886',
       '1876', '199?', '1945', '1905', '1943', '1909', '1874', '1856',
       '1875', '1892', '1879', '1898', '1819', '1882', '1901', '1932',
       '1906', '1868', '1904', '1912', '1902', '1831', '1880', '1862',
       '1917', '1948', '1877', '1888', '1807', '1864', '1849', '1801',
       '1814', '1854', '1857', '1926', '1930', '1881', '1833', '1766',
       '1938', '1848', '2023', '1829', '1834', '1850', '1828', '1789',
       '1858', '1823', '1840', '195?', '1820', '1806', '1758', '1842',
       '1703', '1809', '1846', '1741', '1847', '1826', '1830', '1798',
       '1816', '1778', '1794'], dtype=object)
```

Merging `data` and `rating` datasets together

In [56]:
```
merge_table = pd.merge(rating, data, on='Title', how='inner')
merge_table['review/helpfulness'] = merge_table['review/helpfulness'].apply(review_to_fl
merge_table['review/summary'] = merge_table['review/summary'].astype(str)
merge_table['review/text'] = merge_table['review/text'].astype(str)
merge_table.head()
```
Out[56]:

	Title	Price	review/helpfulness	review/score	review/summary	review/text	authors	publishedYear
0	The Church of Christ: A Biblical Ecclesiology ...	25.97	0.913580	5.0	Ecclesiological Milestone	With the publication of Everett Ferguson's boo...	['Everett Ferguson']	1996

1	The Church of Christ: A Biblical Ecclesiology ...	25.97	0.666667	5.0	Early Christian development of the Church	Everett Ferguson approaches the subject of ear...	['Everett Ferguson']	1996
2	The Church of Christ: A Biblical Ecclesiology ...	25.97	0.666667	4.0	An Excellent Presentation of the Beliefs of th...	This book is a continual resource. It is so bi...	['Everett Ferguson']	1996
3	The Church of Christ: A Biblical Ecclesiology ...	25.97	0.600000	4.0	Christ is Lord	This is a very useful and thorough text book. ...	['Everett Ferguson']	1996
4	The Battleship Bismarck	34.95	1.000000	3.0	The Battleship Bismarck reviewed	This book is both a history and a photo album ...	['Stefan Draminski']	2018

In [57]:
```
merge_table = merge_table.groupby('Title').agg({
    'Price' : 'mean',
    'review/helpfulness' : 'mean',
    'review/score' : 'mean',
    'review/summary' : ' '.join,
    'review/text' : ' '.join,
    'authors' : 'first',
    'publishedYear' : 'first',
    'ratingsCount' : 'sum',
    'categories' : 'first'
}).reset_index()
```

Convert year from strings to num and drop nulls

In [58]:
```
merge_table['publishedYear'].unique()
```

Out[58]:
```
array(['1994', '1991', '2001', '1998', '2004', '2000', '1877', '2013',
       '2012', '2003', '2014', '2002', '1983', '2021', '1992', '2005',
       '2018', '2015', '2009', '2006', '1996', '1999', '2011', '1997',
       '2008', '1985', '1987', '2010', '1982', '1950', '2007', '2020',
       '2016', '1993', '1974', '1941', '1986', 'nan', '1990', '1988',
       '1995', '2017', '1887', '1981', '1962', '1895', '1989', '1874',
       '1971', '1961', '1955', '2019', '1956', '1963', '1973', '1975',
       '2022', '1911', '1972', '1964', '1969', '1894', '1914', '1902',
       '1945', '1984', '1906', '1976', '1967', '1957', '1978', '1980',
       '1912', '1916', '1889', '1852', '1939', '1953', '1979', '1965',
       '1933', '1968', '1885', '1875', '19??', '1899', '1922', '1947',
       '1816', '1883', '1940', '1958', '1919', '2023', '1918', '1915',
       '1959', '1966', '1913', '1891', '1954', '1977', '1851', '1926',
       '1960', '1809', '1932', '1917', '1970', '1942', '1871', '1921',
       '1863', '1920', '1904', '1948', '1901', '1909', '1905', '1951',
       '1872', '1882', '1892', '1952', '1910', '1868', '1907', '1865',
       '1853'], dtype=object)
```

In [59]:
```
# Step 1: Convert publishedYear to numeric, setting errors='coerce' to handle non-numeri
merge_table['publishedYear'] = pd.to_numeric(merge_table['publishedYear'], errors='coerc

# Step 2: Drop rows with NaN values in the publishedYear column
merge_table = merge_table.dropna(subset=['publishedYear'])
```

```
# Step 3: Convert publishedYear to integer type
merge_table['publishedYear'] = merge_table['publishedYear'].astype(int)
```

In [60]:
```
# Check for NaN values in merge_table
nan_values = merge_table.isna().sum()

#import ace_tools as tools; tools.display_dataframe_to_user(name="NaN Values in Merge Ta

nan_values
```

Out[60]:
```
Title                    0
Price                    0
review/helpfulness       0
review/score             0
review/summary           0
review/text              0
authors                106
publishedYear            0
ratingsCount             0
categories               0
dtype: int64
```

In [61]:
```
merge_table = merge_table.dropna(subset=['authors'])
```

In [62]:
```
# Check for NaN values in merge_table
nan_values = merge_table.isna().sum()

#import ace_tools as tools; tools.display_dataframe_to_user(name="NaN Values in Merge Ta

nan_values
```

Out[62]:
```
Title                  0
Price                  0
review/helpfulness     0
review/score           0
review/summary         0
review/text            0
authors                0
publishedYear          0
ratingsCount           0
categories             0
dtype: int64
```

Now, there are no nan values in our table.

In [63]:
```
merge_table.head()
```

Out[63]:

	Title	Price	review/helpfulness	review/score	review/summary	review/text	authors	publishedYear
0	"Come to Me"	10.19	0.500000	5.000000	Find rest in the arms of God	This book absolutely stunned me. I started rea...	['Amy Bloom']	1994
1	"Forget Not Love": The Passion of Maximilian K...	10.16	0.612229	5.000000	&quot;Forget Not Love&quot; is Special Simply ...	&quot;Forget Not Love&quot; is the story of St...	['André Frossard']	1991
2	"Happiness Is Not My Companion": The Life of G...	24.95	0.898280	4.833333	FILLS A VOID Justice Delayed An apt title for ...	"Happiness Is Not My Companion" The Life of Go...	['David M. Jordan']	2001
3	"Life Was	5.00	0.547214	4.473684	Stuart Wilde	Stuart Wilde	['Stuart	1998

	Never Meant to Be a Struggle"				changed my life...I gift this to ...	changed my life... I found this l...	Wilde']	
4	"Mom, Dad . . . I'm Pregnant": When Your Daugh...	10.22	0.984615	4.800000	Helpful book full of advice, encouragement, an...	The whole day is fuzzy in my memory. The conve...	['Jayne E. Schooler']	2004

Results

Exploratory Data Analysis

The goal of this EDA is to explore the relationships of review scores and other factors individually while also isolating variables such as ratings count and word frequencies in the review text. According to the plots, there seems to be a weak correlation between the review score and other factors such as price, number of ratings, and the year of publication.

When looking at the barplot for the average review score and their corresponding price range (e.g., 0-19, 20-40, 40-60...), the review score had little to no variation and all these average review scores seemed to be around 4 stars. This can be explained when we take a look at the review score and the number of reviews (only books with less than 1000 reviews). These book review scores showed similar behavior in comparison to the first example as it is hovering around around 4 stars, but it does show a bit of variation in the review scores.

For the average review score and the year of publication, we can see that the review score seemed to vary a lot in older publications. As year of publication increase, the variation of review scores decreases and start to converge towards the review score of 4. One possible explanation for this can be seen in the number of ratings of each book and the published year graph. This graph shows a bimodal distribution with the peaks around 2000 and 2010. When we look below the year 1950, there seemed to be little to no reviews for those books. This could explain the reason why we see more variation of review scores in the older published books than the recent published books.

Average Review Score & Price

In [33]:
```python
merge_table.sort_values(by=['Price'], ascending=False, inplace=True)
merge_table.dropna(subset=['authors'], inplace=True)
merge_table.head()
```

Out[33]:

	Title	Price	review/helpfulness	review/score	review/summary	review/text	authors	published
5628	New Interpreter's Bible (12 Volume Set + Index)	665.56	0.543254	4.666667	Alpha and Omega Excellent, But Expensive, Reso...	The New Interpreter's Bible is a twelve-volume...	['Michael J. Gorman']	
4226	International Textbook of Obesity	530.00	0.000000	4.000000	International textbok of obesity	Childhood and adolescent obesity presents one ...	['Per Björntorp']	
9605	The Spirit of Prophecy:	499.00	0.586667	4.000000	Victorian age at its best Very	I read this book, here in	['Ellen Gould	

	The Great Controversy ...				interesting boo...	Brazil.I found it on...	Harmon White']
4787	Light to the Isles	385.00	0.500000	3.500000	The Tale of Religions, a spiritual, not a phys...	Okay, honestly, haven't read this book. It jus...	['Juliet Marillier']
5883	Optical Waveguide Theory (Science Paperbacks, ...	382.28	0.333333	4.000000	Good fundamentals on waveguide theory quality ...	The book outlines the fundamentals of electrom...	['A.W. Snyder', 'J. Love']

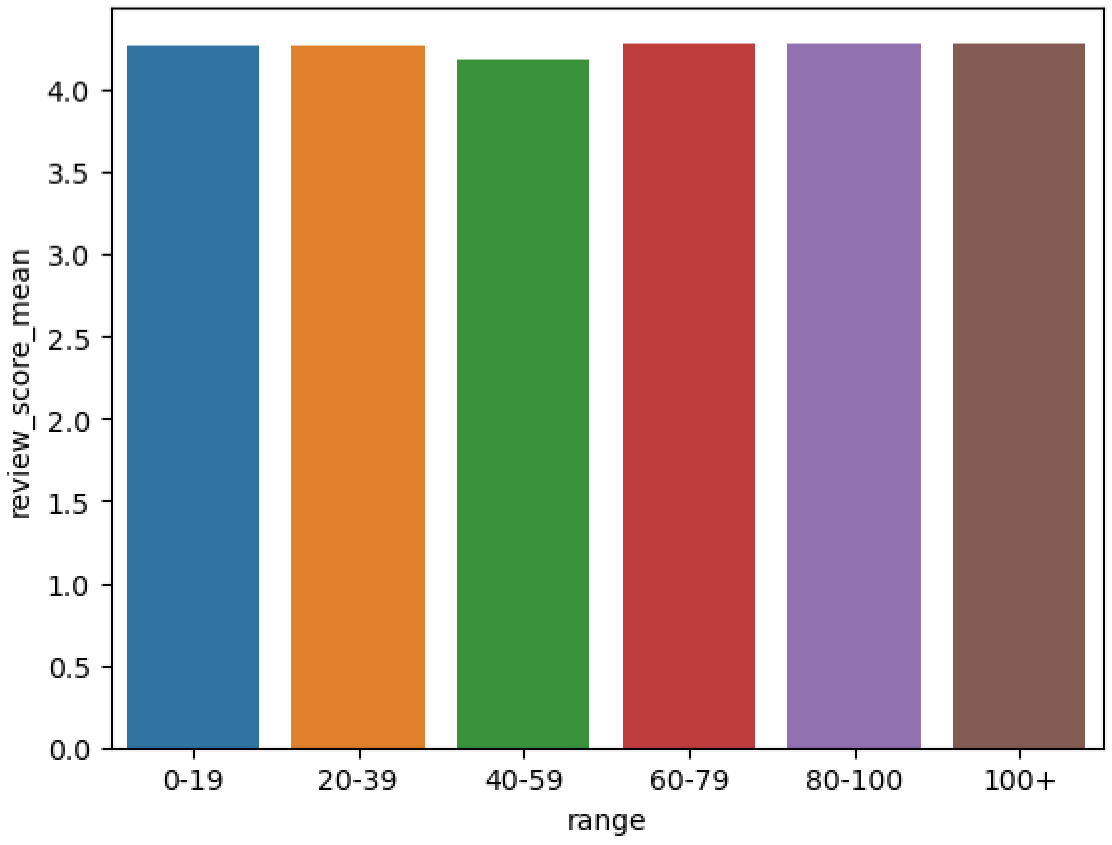
In [27]:
```python
# Group the book review scores by price ranges and finding the mean score for each one
range1 = merge_table[merge_table['Price'] < 20]
mean1 = range1['review/score'].mean()

range2 = merge_table[(merge_table['Price'] >= 20) & (merge_table['Price'] < 40)]
mean2 = range2['review/score'].mean()

range3 = merge_table[(merge_table['Price'] >= 40) & (merge_table['Price'] < 60)]
mean3 = range3['review/score'].mean()

range4 = merge_table[(merge_table['Price'] >= 60) & (merge_table['Price'] < 80)]
mean4 = range4['review/score'].mean()

range5 = merge_table[(merge_table['Price'] >= 80) & (merge_table['Price'] < 100)]
mean5 = range5['review/score'].mean()

range6 = merge_table[merge_table['Price'] >= 100]
mean6 = range6['review/score'].mean()


# Create a new dataframe
d = {'range': ['0-19', '20-39', '40-59', '60-79', '80-100', '100+'],
    'review_score_mean': [mean1, mean2, mean3, mean4, mean5, mean6]}
temp_df = pd.DataFrame(data=d)
temp_df
```

Out[27]:

	range	review_score_mean
0	0-19	4.267522
1	20-39	4.264316
2	40-59	4.188714
3	60-79	4.278582
4	80-100	4.289903
5	100+	4.281532

In [35]:
```python
sns.barplot(temp_df, x=temp_df['range'], y=temp_df['review_score_mean'])
```

Out[35]:
```
<Axes: xlabel='range', ylabel='review_score_mean'>
```



The price range does not influence the review score based on this plot.

Review Score & Review Count

In [28]:
```
# Grab books that contain less than 1000 years to get rid of outliers
count_df = merge_table[merge_table['ratingsCount'] < 1000]
count_df
```

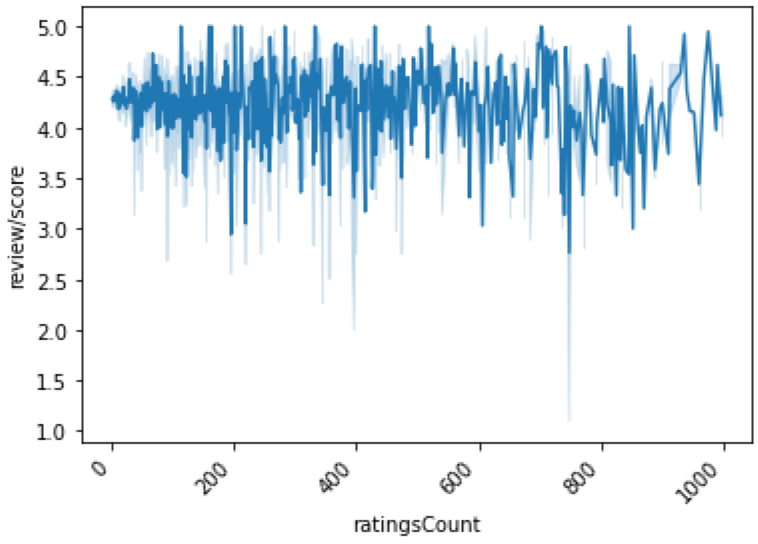
Out[28]:

	Title	Price	review/helpfulness	review/score	review/summary	review/text	authors	publish
0	"Come to Me"	10.19	0.500000	5.000000	Find rest in the arms of God	This book absolutely stunned me. I started rea...	['Amy Bloom']	
1	"Forget Not Love": The Passion of Maximilian K...	10.16	0.612229	5.000000	&quot;Forget Not Love&quot; is Special Simply ...	&quot;Forget Not Love&quot; is the story of St...	['André Frossard']	
2	"Happiness Is Not My Companion": The Life of G...	24.95	0.898280	4.833333	FILLS A VOID Justice Delayed An apt title for ...	"Happiness Is Not My Companion" The Life of Go...	['David M. Jordan']	
3	"Life Was Never Meant to Be a Struggle"	5.00	0.547214	4.473684	Stuart Wilde changed my life...I gift this to ...	Stuart Wilde changed my life... I found this l...	['Stuart Wilde']	
4	"Mom, Dad . . . I'm Pregnant": When Your Daugh...	10.22	0.984615	4.800000	Helpful book full of advice, encouragement, an...	The whole day is fuzzy in my memory. The conve...	['Jayne E. Schooler']	
...	...	...	...	...	...	...	...	

10993	eBay: The Missing Manual	16.70	0.723429	4.888889	INDISPENSABLE eBay guide! Whatever "it" is, yo...	I purchased this book only a week ago and it h...	['Nancy Conner']
10994	eMbedded Visual Basic: Windows CE and Pocket P...	45.63	0.812500	4.000000	Great starter book Great eVB coverage!	This book was really helpful when I was first ...	['Christopher Tacke', 'Timothy Bassett']
10995	mo+th issue 1	10.63	0.000000	4.000000	Prose and Poems For Today's World	I found several of the poems and prose pieces ...	['Angie Cruz']
10996	sendmail Desktop Reference (Pocket Reference)	9.99	0.500000	3.500000	V8.10 is out now Very Outdated Book Reads like...	Major changes in v8.10, including a much easie...	['Greg Neilson']
10997	tick, tick ... BOOM!	14.00	0.458333	4.428571	Death to Melody Doubling Only gave 4 stars bec...	I have yet to understand why the people who ar...	['Jonathan Larson']

10550 rows × 10 columns

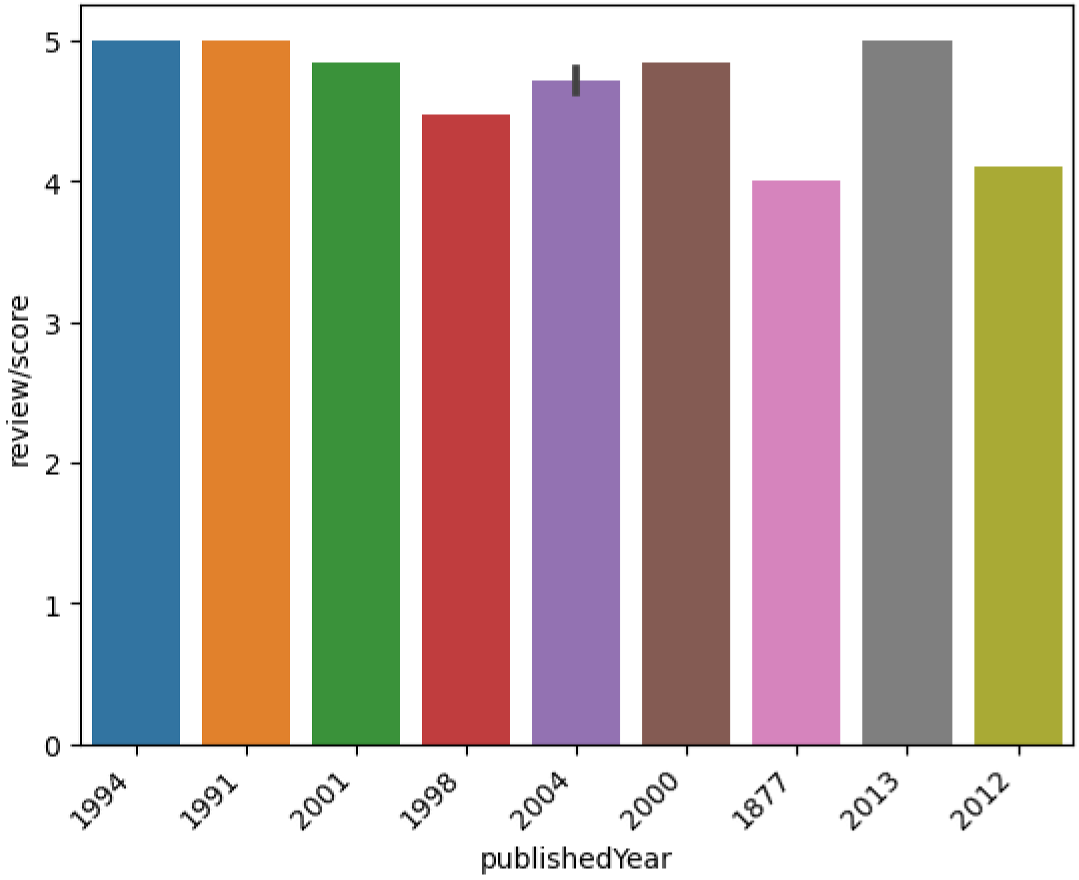
In [75]:
```
sns.lineplot(data=count_df, x=count_df['ratingsCount'], y=count_df['review/score'])

plt.xticks(rotation=45, ha='right')

plt.show()
```



This line plot shows that there are little to no relationship between the number of ratings and the review score (0-5 stars).

Review Scores & Year of Publication

In [30]:
```
# Show the first 10 observations
sns.barplot(data=merge_table, x=merge_table['publishedYear'][:10], y=merge_table['review
plt.xticks(rotation=45, ha='right')

plt.show()
```



The year of publication seems to have little to no relationship with review scores.

In [77]:
```
# For visualization
grouped_data = merge_table.groupby('publishedYear')['review/score'].mean().reset_index()
grouped_data = grouped_data[:-1]
grouped_data
```

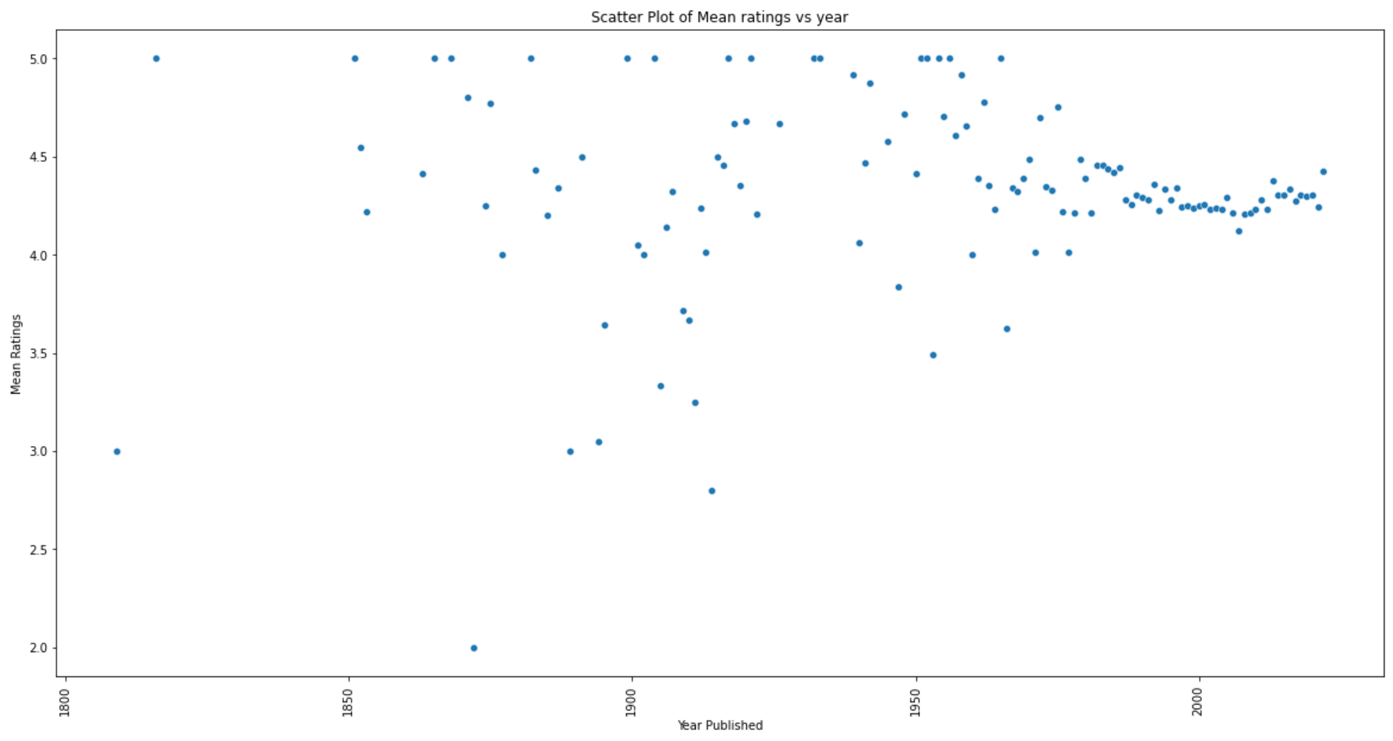
Out[77]:

	publishedYear	review/score
0	1809	3.000000
1	1816	5.000000
2	1851	5.000000
3	1852	4.545455
4	1853	4.218310
...	...	...
120	2018	4.302368
121	2019	4.299262
122	2020	4.305834
123	2021	4.241813
124	2022	4.427005

125 rows × 2 columns

In [78]:
```
# Create a scatter plot with mean ratings
plt.figure(figsize=(20, 10))
```

```
sns.scatterplot(data=grouped_data, x='publishedYear', y='review/score')

# Add labels and title
plt.xlabel('Year Published')
plt.ylabel('Mean Ratings')
plt.title('Scatter Plot of Mean ratings vs year')

# Rotate x-axis labels
plt.xticks(rotation=90)

# Show plot
plt.show()
```



Books published before 1985 seems to have more variablity in mean ratings

In [79]:
```
# For visualization
grouped_data_count = merge_table.groupby('publishedYear')['review/score'].count().reset_
grouped_data_count = grouped_data_count[:-1]
grouped_data_count
```

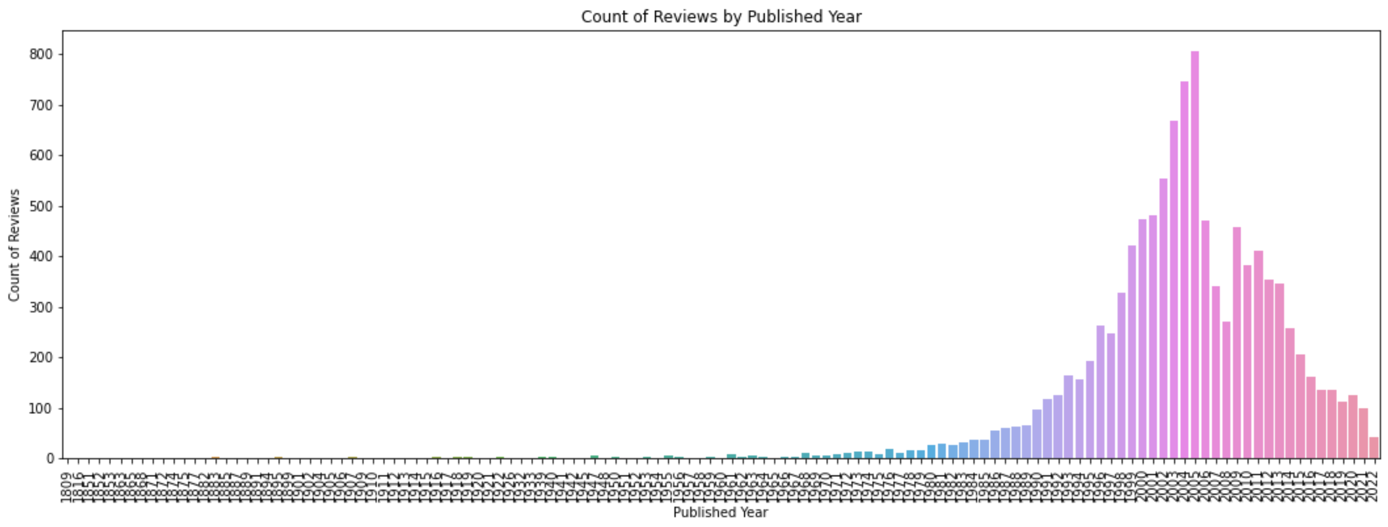
Out[79]:

	publishedYear	review/score
0	1809	1
1	1816	1
2	1851	1
3	1852	2
4	1853	1
...	...	...
120	2018	137
121	2019	113
122	2020	125
123	2021	99
124	2022	42

125 rows × 2 columns

Review Count by published year

In [80]:
```
# Create the bar plot
plt.figure(figsize=(18, 6))
sns.barplot(x='publishedYear', y='review/score', data=grouped_data_count)

# Add labels and title
plt.xlabel('Published Year')
plt.ylabel('Count of Reviews')
plt.title('Count of Reviews by Published Year')
plt.xticks(rotation=90)  # Rotate x labels if needed for better readability

# Display the plot
plt.show()
```



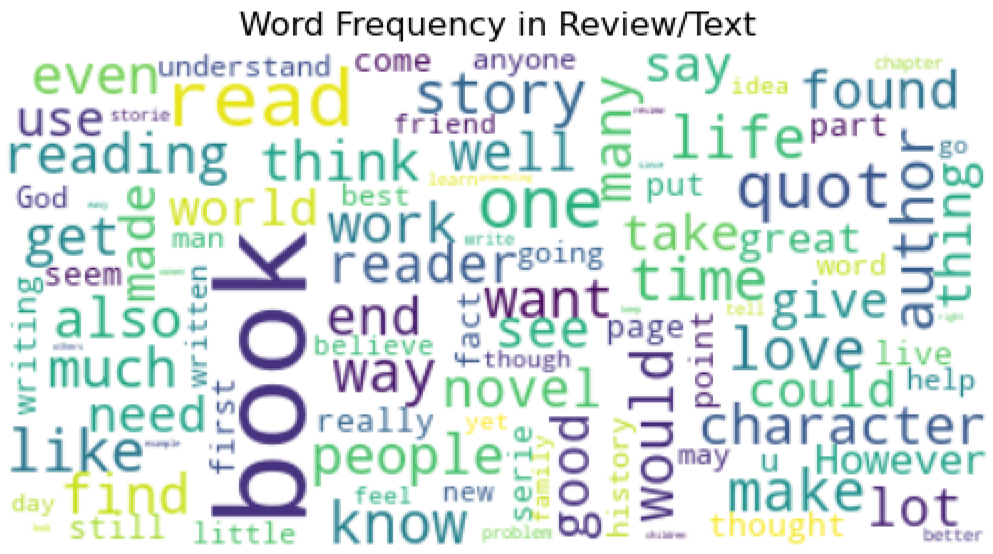
Visualizing the Most Frequent Words in `review/text`

In [81]:
```
%pip install wordcloud
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: wordcloud in /home/amahdin/.local/lib/python3.9/site-pack
ages (1.9.3)
Requirement already satisfied: matplotlib in /opt/conda/lib/python3.9/site-packages (fro
m wordcloud) (3.4.2)
Requirement already satisfied: pillow in /opt/conda/lib/python3.9/site-packages (from wo
rdcloud) (10.2.0)
Requirement already satisfied: numpy>=1.6.1 in /home/amahdin/.local/lib/python3.9/site-p
ackages (from wordcloud) (1.26.4)
Requirement already satisfied: kiwisolver>=1.0.1 in /opt/conda/lib/python3.9/site-packag
es (from matplotlib->wordcloud) (1.3.1)
Requirement already satisfied: python-dateutil>=2.7 in /opt/conda/lib/python3.9/site-pac
kages (from matplotlib->wordcloud) (2.8.2)
Requirement already satisfied: cycler>=0.10 in /opt/conda/lib/python3.9/site-packages (f
rom matplotlib->wordcloud) (0.10.0)
Requirement already satisfied: pyparsing>=2.2.1 in /opt/conda/lib/python3.9/site-package
s (from matplotlib->wordcloud) (2.4.7)
Requirement already satisfied: six in /opt/conda/lib/python3.9/site-packages (from cycle
r>=0.10->matplotlib->wordcloud) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

In [31]:
```
from wordcloud import WordCloud

text = " ".join( review for review in merge_table['review/text'])
```

```
wordcloud = WordCloud(
    max_font_size=50,
    max_words=100,
    background_color="white",
    stopwords=stopwords.words("english")).generate( text )

# Create a wordcloud
plt.figure()
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.title('Word Frequency in Review/Text')
plt.show()
```



The word cloud shows that people use the word "book" most frequently. The tone of the words are mostly positive possibly implying a bias towards positive reviews in general. This implies that positive reviewers post more often and/or are wordier than negative reviewers.

Model 1

Linear Regression Model

In [64]:
```
model = LinearRegression()
vectorizer = CountVectorizer(analyzer='word', max_features=2000, tokenizer=word_tokenize
merge_table.head()
```

Out[64]:

	Title	Price	review/helpfulness	review/score	review/summary	review/text	authors	publishedYear
0	"Come to Me"	10.19	0.500000	5.000000	Find rest in the arms of God	This book absolutely stunned me. I started rea...	['Amy Bloom']	1994
1	"Forget Not Love": The Passion of Maximilian K...	10.16	0.612229	5.000000	&quot;Forget Not Love&quot; is Special Simply ...	&quot;Forget Not Love&quot; is the story of St...	['André Frossard']	1991
2	"Happiness Is Not My Companion":	24.95	0.898280	4.833333	FILLS A VOID Justice Delayed An apt title for ...	"Happiness Is Not My Companion"	['David M. Jordan']	2001

	The Life of G...					The Life of Go...		
3	"Life Was Never Meant to Be a Struggle"	5.00	0.547214	4.473684	Stuart Wilde changed my life...I gift this to ...	Stuart Wilde changed my life... I found this l...	['Stuart Wilde']	1998
4	"Mom, Dad . . . I'm Pregnant": When Your Daugh...	10.22	0.984615	4.800000	Helpful book full of advice, encouragement, an...	The whole day is fuzzy in my memory. The conve...	['Jayne E. Schooler']	2004

In [65]:
```
text_X = vectorizer.fit_transform(merge_table['review/text']).toarray()
```

In [85]:
```
price_X = merge_table['Price'].to_numpy().reshape(len(text_X), 1)
helpfulness_X = merge_table['review/helpfulness'].to_numpy().reshape(len(text_X), 1)
rating_X = merge_table['ratingsCount'].to_numpy().reshape(len(text_X), 1)

X = price_X + text_X + helpfulness_X + rating_X
Y = merge_table['review/score'].to_numpy()
```

In [86]:
```
train_X, test_X, train_Y, test_Y = train_test_split(X, Y, test_size=0.2, random_state=20
```

In [87]:
```
def regression_report(y_true, y_pred):
    mse = mean_squared_error(y_true, y_pred)
    mae = mean_absolute_error(y_true, y_pred)
    r2 = r2_score(y_true, y_pred)
    rmse = np.sqrt(mse)

    report = {
        'Mean Squared Error (MSE)': mse,
        'Mean Absolute Error (MAE)': mae,
        'Root Mean Squared Error (RMSE)': rmse,
        'R^2 Score': r2
    }

    return report

def print_regression_report(report):
    for metric, value in report.items():
        print(f'{metric}: {value:.4f}')

model.fit(train_X, train_Y)
```

Out[87]:
```
□ LinearRegression
LinearRegression()
```

Prediction for training set

In [88]:
```
pred_train_Y = model.predict(train_X)
pred_train_Y = np.clip(pred_train_Y, 0, 5)
report = regression_report(train_Y, pred_train_Y)
print_regression_report(report)
```

```
Mean Squared Error (MSE): 0.3519
Mean Absolute Error (MAE): 0.4181
Root Mean Squared Error (RMSE): 0.5932
R^2 Score: 0.2702
```

According to these results, the errors are very close to 0 which indicates that the model is either overfitting or performing well. The value of R^2 however seems to be pretty low as it is closer to 0 than 1.

Prediction for testing set

```
In [89]: pred_test_Y = model.predict(test_X)
         pred_test_Y = np.clip(pred_test_Y, 0, 5)
         report = regression_report(test_Y, pred_test_Y)
         print_regression_report(report)
```

```
Mean Squared Error (MSE): 0.5964
Mean Absolute Error (MAE): 0.5445
Root Mean Squared Error (RMSE): 0.7723
R^2 Score: -0.2648
```

Similar to the prediction above, the error values in this result are closer to 0. This indicates that the model is either overfitting or performing well. Since the R^2 value is negative, this indicates that the model is not performing well on the testing set. This means that it is not able to accurately predict the review scores based on prices, text, review helpfulness and the number of ratings. In other words, the variables(price, text, helpfulness, number of ratings) have little to no influence on the review score.

Model 2

TF-IDF on sentiment analysis of book reviews

Create a TfidfVectorizer object to transform the text review data into vectors.

```
In [90]: tfidf = TfidfVectorizer(
             sublinear_tf = True,
             analyzer='word',
             max_features=2000,
             stop_words='english',
             token_pattern=r'\b\w+\b')
```

```
In [91]: len(merge_table['review/text'])
```

```
Out[91]: 10872
```

```
In [92]: merge_table['review/text']
```

```
Out[92]: 5628    The New Interpreter's Bible is a twelve-volume...
         4226    Childhood and adolescent obesity presents one ...
         9605    I read this book, here in Brazil.I found it on...
         4787    Okay, honestly, haven't read this book. It jus...
         5883    The book outlines the fundamentals of electrom...
                                       ...
         1791    It was soon nice to be able to get my book at ...
         1792    I bought this for my granddaughter, who is in ...
         1796    Homer's The Odyssey is a drudgery to read not...
         1793    Once, we were impressed by the characters and ...
         1802    Perfect. A great addition to the book. if you ...
         Name: review/text, Length: 10872, dtype: object
```

```
In [93]: Review_tfidf_X = tfidf.fit_transform(merge_table['review/text']).toarray()
```

```
In [94]: assert isinstance(Review_tfidf_X, np.ndarray)
In [95]: Review_Y = merge_table['review/score'].values
         Review_Y = np.round(Review_Y).astype(int)
```

```
In [96]: assert Review_Y.shape == (10872,)
```

```
In [97]: Review_train_tfidf_X,Review_test_tfidf_X,Review_train_tfidf_Y,Review_test_tfidf_Y=train_
             Review_tfidf_X,
             Review_Y,
             test_size=0.2,
             random_state=42)
```

Training

```
In [98]: def train_SVM(X,y,kernel='linear'):
             clf = SVC(kernel=kernel)
             clf.fit(X,y)
             return clf
```

```
In [109… Review_tfidf_clf = train_SVM(Review_train_tfidf_X, Review_train_tfidf_Y)
```

Prediction

```
In [111… # Predict and evaluate
         y_pred_train = Review_tfidf_clf.predict(Review_train_tfidf_X)
         y_pred_test = Review_tfidf_clf.predict(Review_test_tfidf_X)

         # Calculate metrics
         mae_train = mean_absolute_error(Review_train_tfidf_Y, y_pred_train)
         mse_train = mean_squared_error(Review_train_tfidf_Y, y_pred_train)
         rmse_train = np.sqrt(mse_train)
         r2_train = r2_score(Review_train_tfidf_Y, y_pred_train)

         mae_test = mean_absolute_error(Review_test_tfidf_Y, y_pred_test)
         mse_test = mean_squared_error(Review_test_tfidf_Y, y_pred_test)
         rmse_test = np.sqrt(mse_test)
         r2_test = r2_score(Review_test_tfidf_Y, y_pred_test)

         print(f'Training Mean Absolute Error: {mae_train}')
         print(f'Training Mean Squared Error: {mse_train}')
         print(f'Training RMSE: {rmse_train}')
         print(f'Training R^2 Score: {r2_train}')
         print(f'Test Mean Absolute Error: {mae_test}')
         print(f'Test Mean Squared Error: {mse_test}')
         print(f'Test RMSE: {rmse_test}')
         print(f'Test R^2 Score: {r2_test}')
```

```
Training Mean Absolute Error: 0.2767621018742095
Training Mean Squared Error: 0.35770955501897206
Training RMSE: 0.5980882501930397
Training R^2 Score: 0.36326921335444673
Test Mean Absolute Error: 0.3990804597701149
Test Mean Squared Error: 0.5085057471264368
Test RMSE: 0.7130958891526699
Test R^2 Score: 0.13371078821856763
```

Model 3

BERT and Random Forrest

In [66]:
```python
def standardize_text(string):
    string = string.lower()
    string = string.strip()

    string = re.sub("[^a-z0-9 ]", ' ', string)
    string = re.sub(r'\s+',' ',string)

    string = string.strip()
    return string

def standardize_price(float):
    if float < 15:
        return 'low'
    elif float < 30:
        return 'mid'
    else:
        return 'high'
```

In [67]:
```python
# Apply the standardize_text function to the 'review' column
merge_table['review/summary'] = merge_table['review/summary'].apply(standardize_text)
merge_table['review/text'] = merge_table['review/text'].apply(standardize_text)
# Apply the standardize_price function to the 'price' column
merge_table['Price'] = merge_table['Price'].apply(standardize_price)
merge_table.drop(columns = ['review/helpfulness', 'authors', 'ratingsCount', 'Title', 'r
```

In [68]:
```python
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
device
```

Out[68]:
```
device(type='cuda')
```

In [69]:
```python
#Renaming merge_table
df = merge_table
```

In [70]:
```python
#import logging
#logging.basicConfig(level=logging.ERROR)
#warnings.filterwarnings('ignore')
# Load pre-trained model and tokenizer
model_name = 'nlptown/bert-base-multilingual-uncased-sentiment'
tokenizer = BertTokenizer.from_pretrained(model_name)
model = BertForSequenceClassification.from_pretrained(model_name).to(device)

# Create a sentiment analysis pipeline
sentiment_pipeline = pipeline('sentiment-analysis', model=model, tokenizer=tokenizer, de


# Function to truncate the text to the maximum length
def truncate_text(text, tokenizer, max_length=512):
    encoded_input = tokenizer.encode_plus(text, max_length=max_length, truncation=True,
    return tokenizer.decode(encoded_input['input_ids'][0], skip_special_tokens=True)

# Advanced Sentiment Analysis with truncation
def advanced_sentiment_analysis(text):
    truncated_text = truncate_text(text, tokenizer)
    result = sentiment_pipeline(truncated_text)
    return result[0]['label']

df['sentiment_score'] = df['review/summary'].apply(advanced_sentiment_analysis)
```

In [71]:
```python
# Mapping for sentiment labels to integer scores
label_to_int = {
    '1 star': 1,
    '2 stars': 2,
    '3 stars': 3,
    '4 stars': 4,
    '5 stars': 5
}
# Convert sentiment labels to integer scores
df['sentiment_score'] = df['sentiment_score'].apply(lambda x: label_to_int[x])
score = df['sentiment_score']
```

In [72]:
```
# Apply MultiLabelBinarizer to the categories column
mlb = MultiLabelBinarizer()
categories_transformed = mlb.fit_transform(df['categories'])

# Create a DataFrame from the transformed categories and merge it back with the original
categories_df = pd.DataFrame(categories_transformed, columns=mlb.classes_, index=df.inde
df = pd.concat([df, categories_df], axis=1)

# Define the feature columns and target
X = df.drop(columns=['review/score', 'review/summary', 'categories'])
y = df['review/score']

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42

# preprocessor for the remaining columns with handle_unknown='ignore'
preprocessor = ColumnTransformer(
    transformers=[
        ('price', OneHotEncoder(handle_unknown='ignore'), ['Price']),
        ('num', StandardScaler(), ['publishedYear', 'sentiment_score'])
    ],
    remainder='passthrough'
)

# RandomForestRegressor model
rf_model = RandomForestRegressor(random_state=42)

# Model pipeline
model = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('regressor', rf_model)
])

# Hyperparameter tuning
param_grid = {
    'regressor__n_estimators': [100, 200, 300],
    'regressor__max_depth': [3, 6, 10, None],
    'regressor__min_samples_split': [2, 5, 10],
    'regressor__min_samples_leaf': [1, 2, 4]
}

grid_search = GridSearchCV(model, param_grid, cv=5, scoring='neg_mean_absolute_error', n
grid_search.fit(X_train, y_train)

# Best model
best_model = grid_search.best_estimator_
```

In [73]:
```
grid_search.best_params_
```

Out[73]:
```
{'regressor__max_depth': 6,
 'regressor__min_samples_leaf': 4,
 'regressor__min_samples_split': 2,
 'regressor__n_estimators': 300}
```

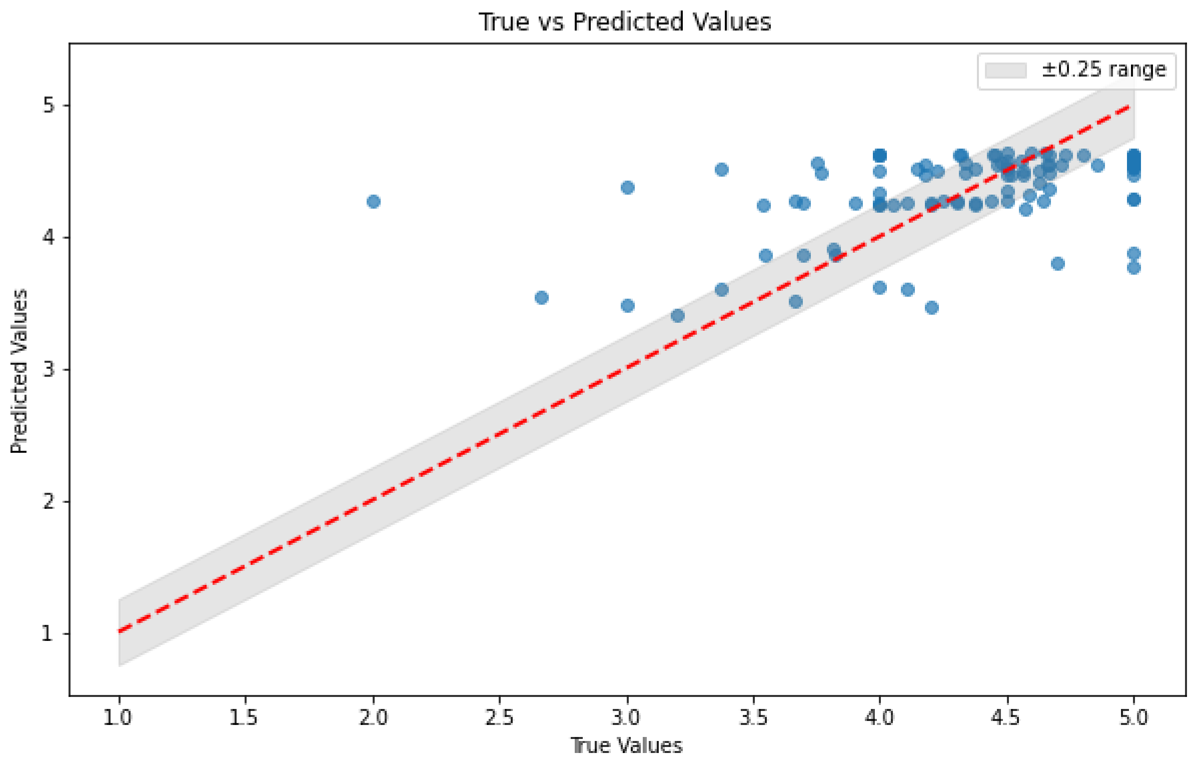
In [74]:
```
# Predict and evaluate
y_pred_train = best_model.predict(X_train)
y_pred_test = best_model.predict(X_test)
```

```python
mae_train = mean_absolute_error(y_train, y_pred_train)
mse_train = mean_squared_error(y_train, y_pred_train)
rmse_train = np.sqrt(mse_train)
r2_train = r2_score(y_train, y_pred_train)

mae_test = mean_absolute_error(y_test, y_pred_test)
mse_test = mean_squared_error(y_test, y_pred_test)
rmse_test = np.sqrt(mse_test)
r2_test = r2_score(y_test, y_pred_test)
```

In [75]:
```python
print(f'Training Mean Absolute Error: {mae_train}')
print(f'Training Mean Squared Error: {mse_train}')
print(f'Training RMSE: {rmse_train}')
print(f'Training R^2 Score: {r2_train}')
print(f'Test Mean Absolute Error: {mae_test}')
print(f'Test Mean Squared Error: {mse_test}')
print(f'Test RMSE: {rmse_test}')
print(f'Test R^2 Score: {r2_test}')
```

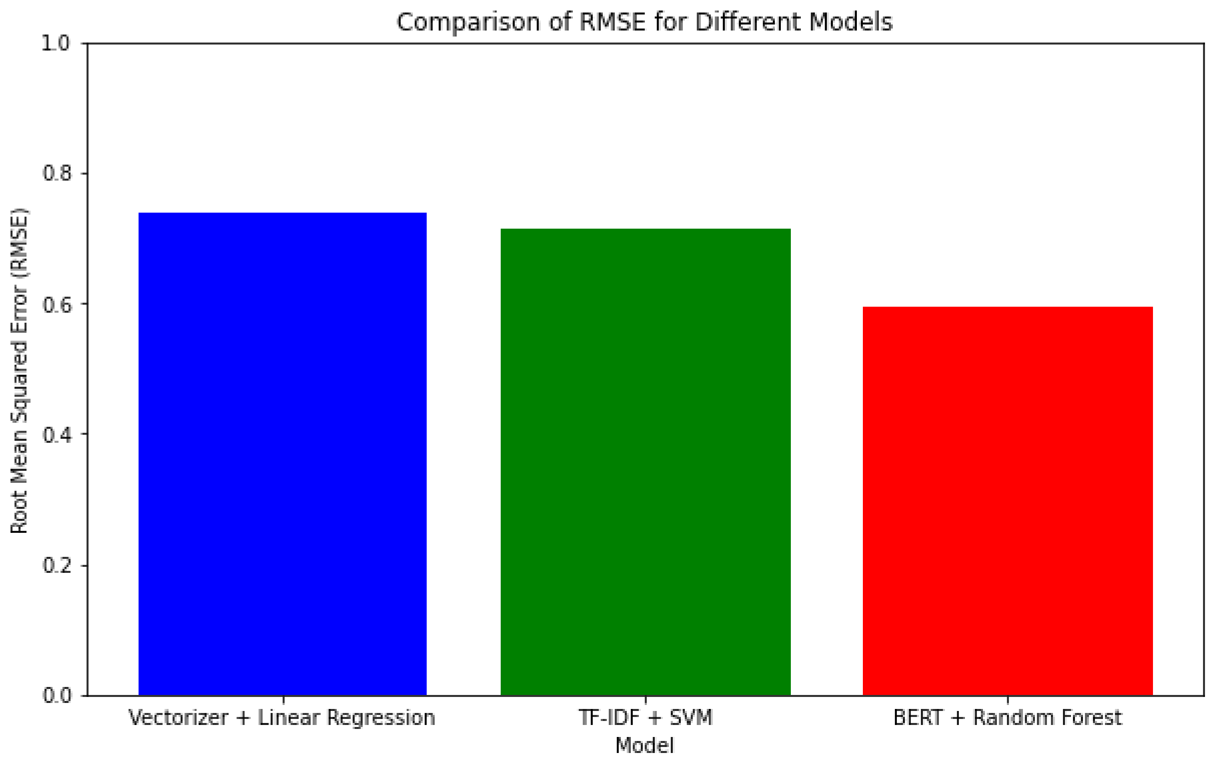
```
Training Mean Absolute Error: 0.41249198176069124
Training Mean Squared Error: 0.32306905873898656
Training RMSE: 0.5683916420382926
Training R^2 Score: 0.3262075329847932
Test Mean Absolute Error: 0.4248941516967762
Test Mean Squared Error: 0.35196864486854296
Test RMSE: 0.5932694538475269
Test R^2 Score: 0.2705927262644512
```

In [78]:
```python
# Sample 100 data points from predictions and true values
sample_indices = np.random.choice(len(y_test), size=100, replace=False)
y_test_sample = y_test.iloc[sample_indices]
y_pred_sample = y_pred_test[sample_indices]

# Plotting
plt.figure(figsize=(10, 6))
plt.scatter(y_test_sample, y_pred_sample, alpha=0.7)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--', lw=2)

# with a 0.25 tolerance on each side
plt.fill_between([y_test.min(), y_test.max()],
                 [y_test.min() - 0.25, y_test.max() - 0.25],
                 [y_test.min() + 0.25, y_test.max() + 0.25],
                 color='gray', alpha=0.2, label='±0.25 range')

plt.xlabel('True Values')
plt.ylabel('Predicted Values')
plt.title('True vs Predicted Values')
plt.legend()
plt.show()
```



Model comparisons

In [79]:
```
import matplotlib.pyplot as plt

# Test Results
results = {
    "Vectorizer + Linear Regression": 0.7389,
    "TF-IDF + SVM": 0.7131,
    "BERT + Random Forest": 0.5933
}

# Create a bar plot
models = list(results.keys())
rmse_values = list(results.values())

plt.figure(figsize=(10, 6))
plt.bar(models, rmse_values, color=['blue', 'green', 'red'])
plt.xlabel('Model')
plt.ylabel('Root Mean Squared Error (RMSE)')
plt.title('Comparison of RMSE for Different Models')
plt.ylim(0, 1)  # Adjust the y-axis limit for better visualization
plt.show()
```



Analysis

Approaches Used

1. **Vectorizer with Linear Regression:**
   - **Why:** Linear regression is a straightforward and interpretable model, and vectorizing text data is a common method to convert textual data into numerical form for regression tasks.
   - **How:** Applied count vectorization to the review text and used linear regression to predict review scores.

1. **TF-IDF with Support Vector Machine (SVM):**
   - **Why:** TF-IDF (Term Frequency-Inverse Document Frequency) helps in understanding the importance of words in the context of the document and the corpus. SVM is a machine learning algorithm good for classification tasks.
   - **How:** We used TF-IDF to transform the review text and then trained an SVM regressor to predict the review scores.
2. **Sentiment Analysis with BERT and Random Forest:**
   - **Why:** BERT (Bidirectional Encoder Representations from Transformers) is a powerful transformer-based model known for its superior performance in various NLP tasks, including sentiment analysis. Random Forest is a versatile ensemble method that combines multiple decision trees to improve performance.
   - **How:** We performed sentiment analysis using BERT to obtain sentiment scores for the reviews and then used these scores along with other features to train a Random Forest regressor.

Results

1. **Vectorizer Paired with Linear Regression:**
   - **Test Results:**

        - Mean Squared Error (MSE): 0.5460
        - Mean Absolute Error (MAE): 0.5405
        - Root Mean Squared Error (RMSE): 0.7389
        - R^2 Score: -0.1301
2. **TF-IDF and Support Vector Machine (SVM):**

    - **Test Results:**
        - Mean Absolute Error (MAE): 0.3991
        - Mean Squared Error (MSE): 0.5085
        - Root Mean Squared Error (RMSE): 0.7131
        - R^2 Score: 0.1337
3. **Sentiment Analysis with BERT and Random Forest:**

    - **Test Results:**
        - Mean Absolute Error (MAE): 0.4249
        - Mean Squared Error (MSE): 0.3520
        - Root Mean Squared Error (RMSE): 0.5933
        - R^2 Score: 0.2706

Interpretation of Findings

1. **Vectorizer Paired with Linear Regression:**
    - **Performance:** This model performed poorly on the test set, as indicated by the negative R^2 score. This suggests that the model did not capture the underlying patterns in the data and performed worse than a simple mean prediction.

1. **TF-IDF and Support Vector Machine (SVM):**
    - **Performance:** The SVM model showed an improvement over the linear regression model, with a positive R^2 score. However, the improvement was modest, suggesting that while the TF-IDF representation and SVM were better suited for the task, they still did not fully capture the complexity of the data.

1. **Sentiment Analysis with BERT and Random Forest:**
    - **Performance:** This model achieved the best results, with the lowest MSE and RMSE and the highest R^2 score. The use of BERT for sentiment analysis provided a deeper understanding of the text data, which, when combined with the Random Forest regressor, led to improved predictions.

Conclusion

- **Improvement Over Baseline:** Each subsequent approach demonstrated an improvement over the previous one, highlighting the importance of advanced feature extraction techniques and modeling methods.

- **Future Work:** Future improvements could involve using BERT on the **review/text** instead which is longer but is going to take more processing power, further fine tuning it, or employing other advanced regression techniques like gradient boosting to further enhance performance.

Conclusion & Discussion

Summary of Data and Question

In this project, we aimed to predict the review scores (out of 5 stars) of books based on factors such as prices, year of publication, and categories, using U.S. Amazon reviews from 2010 to 2023. Our hypothesis was whether these factors, along with advanced text-based features like sentiment analysis, could be used to accurately forecast the review ratings.

Analysis Overview

We began our analysis with a simple approach, using a vectorizer paired with linear regression. This initial model was followed by a more advanced technique that incorporated TF-IDF and Support Vector Machine (SVM) regression. Finally, we used sentiment analysis with BERT (a transformer-based model) and combined the extracted sentiment scores with a Random Forest regressor to make our predictions. Each model was evaluated using key performance metrics, including Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared (R^2) score. We also performed hyperparameter tuning using GridSearchCV to optimize the performance of our models.

Ethics & Privacy

- There are not many concerns with the ethics of our current dataset as it is mostly based on publicly posted reviews. The largest concern is in collecting comments we will also be collecting names, though we intend to exclude usernames from our dataset since they do not serve any purpose.
- People are not intended to participate in our study, as they posted the reviews for other shoppers. However, they are posted publicly, and any person would be able to find this information even without prior knowledge of our research.
- We expect a bias towards negative reviews in general, since negative experiences are strong motivators to leave reviews.
- Missing perspective (i.e. people who don’t leave written reviews/review at all)
  - Compare overall rating to average written review rating
- Bots skewing results (i.e. lower review count products buying reviews)
  - Exclude products that are found to have too many bot reviews.
- Possible seller bias (i.e. setting prices, no correlation between price and quality)
  - Comparing prices with customer review, also directly related to what our hypothesis is asking.

Discusison and Conclusion

Our findings showed a clear progression in model performance. The initial vectorizer and linear regression model performed poorly, with an R^2 score of -0.1301 on the test set, indicating it did not capture the underlying patterns. The TF-IDF and SVM model improved the R^2 score to 0.1337, demonstrating that more sophisticated feature extraction and modeling techniques could better predict review scores. The final approach, which combined BERT for sentiment analysis with a Random Forest regressor, yielded the best results, achieving an R^2 score of 0.2706 and significantly lower MAE and RMSE values. This indicates that incorporating advanced NLP techniques and ensemble learning methods can substantially enhance predictive accuracy.

Limitations:

1. **Dataset Inconsistencies:** One major limitation is the inconsistency between written reviews and the corresponding star ratings. For instance, a reviewer might praise a book as "a very good read" yet only give it 3 or 4 stars due to their higher personal standards. This discrepancy can introduce noise and reduce the model's predictive accuracy.
2. **Hardware Constraints:** While BERT was used effectively on review summaries, applying it to the full reviews could potentially yield better results by capturing more comprehensive sentiment and contextual information. However, this approach would require more powerful GPU resources to handle the increased computational load.

Impact:

Our study shows that using advanced machine learning and NLP techniques can help predict book ratings. This can be useful for publishers and marketers to better understand what readers think and to improve their marketing strategies. Better prediction models can enhance book recommendations and make marketing more personalized, ultimately leading to happier readers.

Team Contributions

- Bryan Ung
  - review score vs. price
  - visualizing most frequent words
  - wrote some background and prior work
  - added references to the background and prior work
  - data overview
  - wrote some explanation for analysis
  - update timeline
- Gunju Kim
  - data analysis using TF-IDF
  - wrote background paragraph
  - created one visualization
  - train the TF-IDF model
  - test the TF-IDF model
- Asif Mahdin
  - wrangled the data
  - handled null values
  - created 4 of the visualizations
  - trained the BERT model
  - trained the Random Forest model
  - did the model evaluation section
  - wrote the abstract of the final report
  - wrote the analysis section of the final report
  - wrote the conclusion part of the final report
- ChaoYuan Lin
  - Data wrangling
  - Linear Regression model
  - merge tables
  - Graphs

    - Handle null values
- Merak Olson
    - wrote timeline
    - coded wordcloud
    - wrote some background and prior work
    - EDA
    - edit and directing video