Book Price

Farzam Manafzadeh | Data Science

THE DATASET HAS THE FOLLOWING ATTRIBUTES:

Dataset Description

Books of different genres, from thousands of authors. In this challenge, participants are required to use the dataset to build a Machine Learning model to predict the price of books based on a given set of features.

FEATURES:

Title: The title of the book

Author: The author(s) of the book.

Edition: The edition of the book eg (Paperback,– Import, 26 Apr 2018)

Reviews: The customer reviews about the book

Ratings: The customer ratings of the book

Synopsis: The synopsis of the book

Genre: The genre the book belongs to

BookCategory: The department the book is usually available at.

Price: The price of the book (Target variable)

For more information aboat the Submission File see this link:

www.kaggle.com/competitions/book-price-prediction-cs-sbu/overview/evaluation

Data Overview:

The dataset consists of 5699 entries, each representing a book, across 9 columns. Below is a brief description of each column:

Title: The title of the book.

Author: The author of the book.

Edition: Details about the edition of the book, such as the format and publication date.

Reviews: The reviews of the book, given as a string.

Ratings: The ratings of the book, presented as a string.

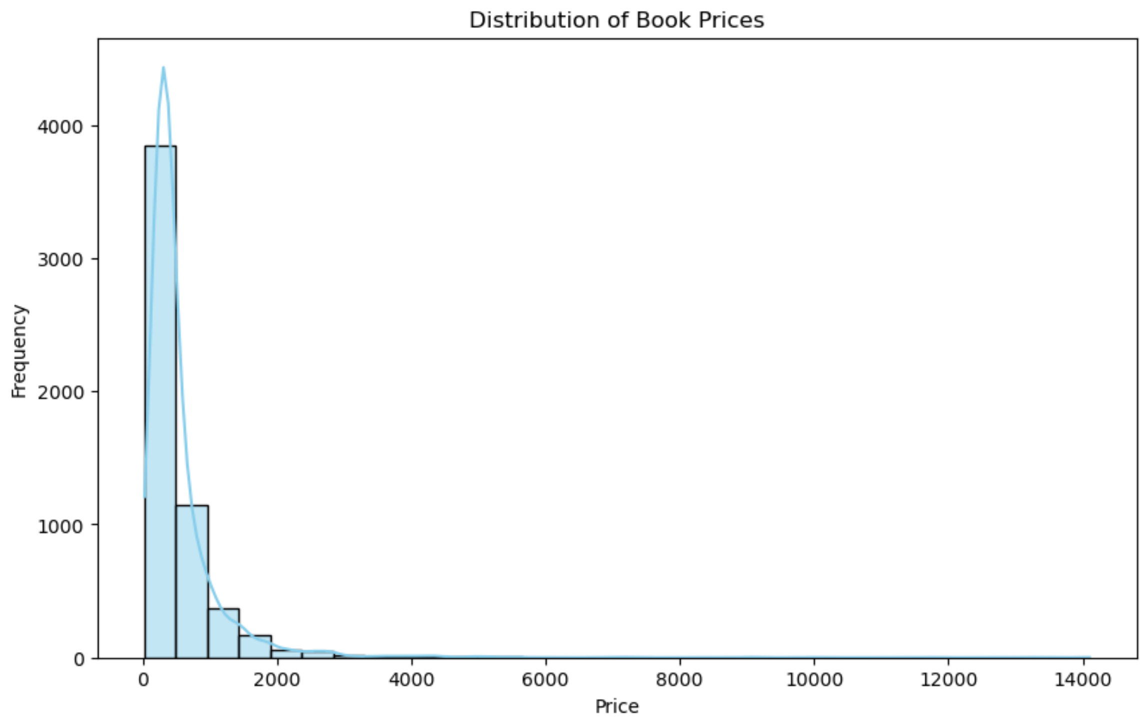
Synopsis: A brief description or summary of the book.

Genre: The genre or category to which the book belongs.

BookCategory: The broader category to which the book is classified.

Price: The price of the book (target variable), represented as a float.

The dataset appears to be clean, with no missing values in any of the columns. The 'Price' column, our target variable, is of data type float.



- Count: There are 5699 data points in the 'Price' column.

- Mean: The average price of the books is approximately 554.86.
- Standard Deviation (std): The standard deviation is relatively high at 674.36, indicating a significant amount of variability in book prices.
- Minimum (min): The minimum price is 25, suggesting that there are books with a relatively low price.
- 25th Percentile (25%): 25% of the books have a price less than or equal to 249.
- 50th Percentile (Median - 50%): The median price is 373, meaning that half of the books have a price below 373 and half above.
- 75th Percentile (75%): 75% of the books have a price less than or equal to 599.
- Maximum (max): The maximum price is 14100, indicating that there are books with a considerably high price.

In summary, dataset has a wide range of book

Data Preprocessing:

Missing Values: Good news! The dataset is clean, with no missing values across all columns. This ensures a solid foundation for our analysis and modeling.

Duplicates: Similarly, there are no duplicate rows in the dataset. Each entry is unique, enhancing the reliability of our data.

However, there are duplicate titles in the dataset. After a thorough examination, we find that there are 569 duplicated titles, and some titles have varying prices. Specifically, 256 titles have different prices among the duplicates.

Action Steps:

> **Handling Duplicates:** Consider whether to keep all duplicates or retain only one instance of each title. If keeping all, ensure that the variations in prices are acceptable or take an average for uniformity.

Outliers Analysis:

Random Forest indeed tends to be less sensitive to outliers, making it a suitable choice for this dataset. To identify potential outliers in the 'Price' column, calculated the Interquartile Range (IQR)

<u>But they haven’t been removed.</u>

Data Transformation:

1. **Conversion to Numerical:**

   - **'Reviews' and 'Ratings' columns were successfully converted to numerical features.**

   - **The updated DataFrame shows the new numerical representations of these features.**

2. **Data Information:**

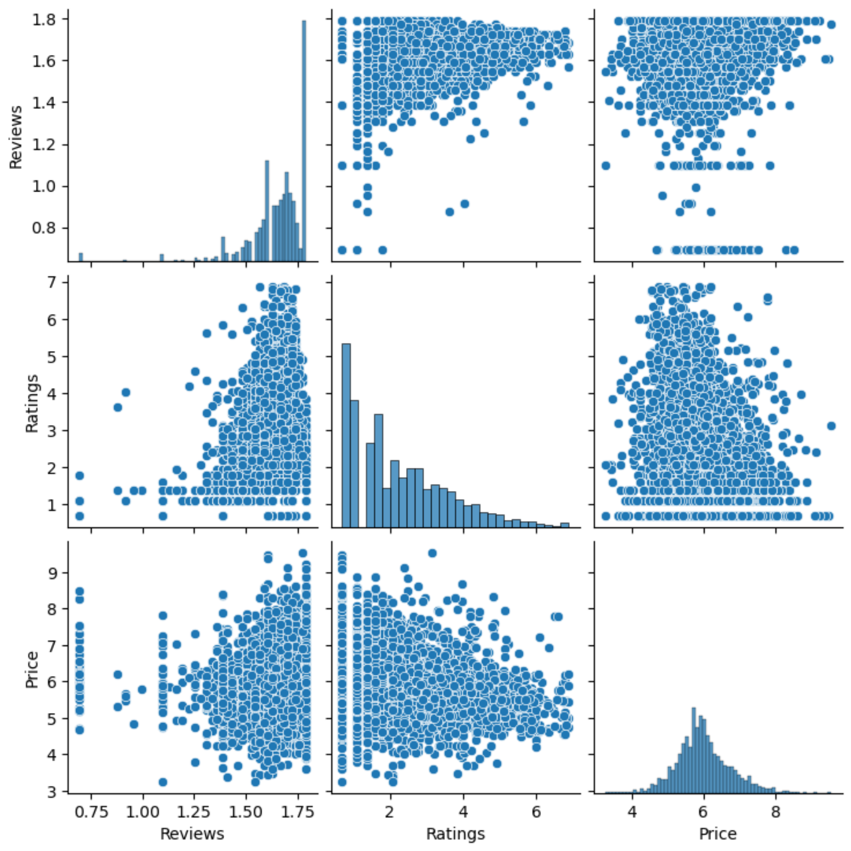
   - **The updated data.info() shows the transformation's success, with 'Reviews' as float64, 'Ratings' as int32, and 'Price' and 'Ratings' now having non-null entries.**

3. **Log Transformations:**

   - **Logarithmic transformations have been applied to numeric columns using np.log1p() to handle skewed distributions and achieve more symmetrical data.**

4. **Visualization:**

   - **A pairplot has been created for visualizing the relationships among numeric features post-logarithmic transformation.**

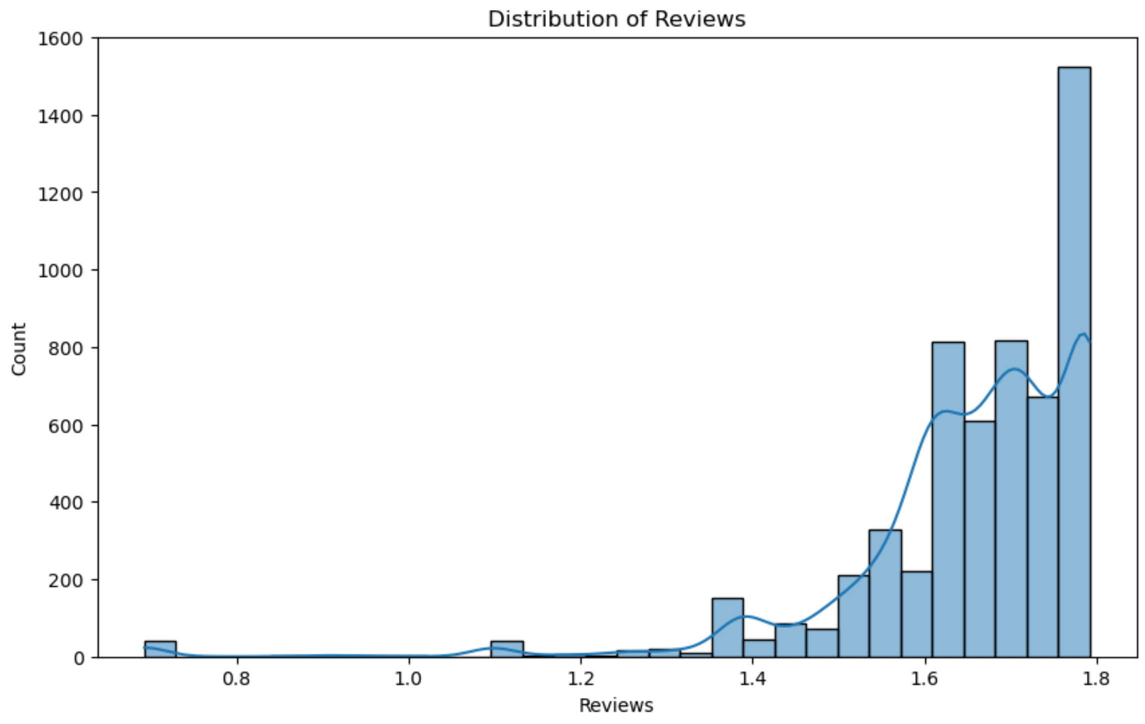


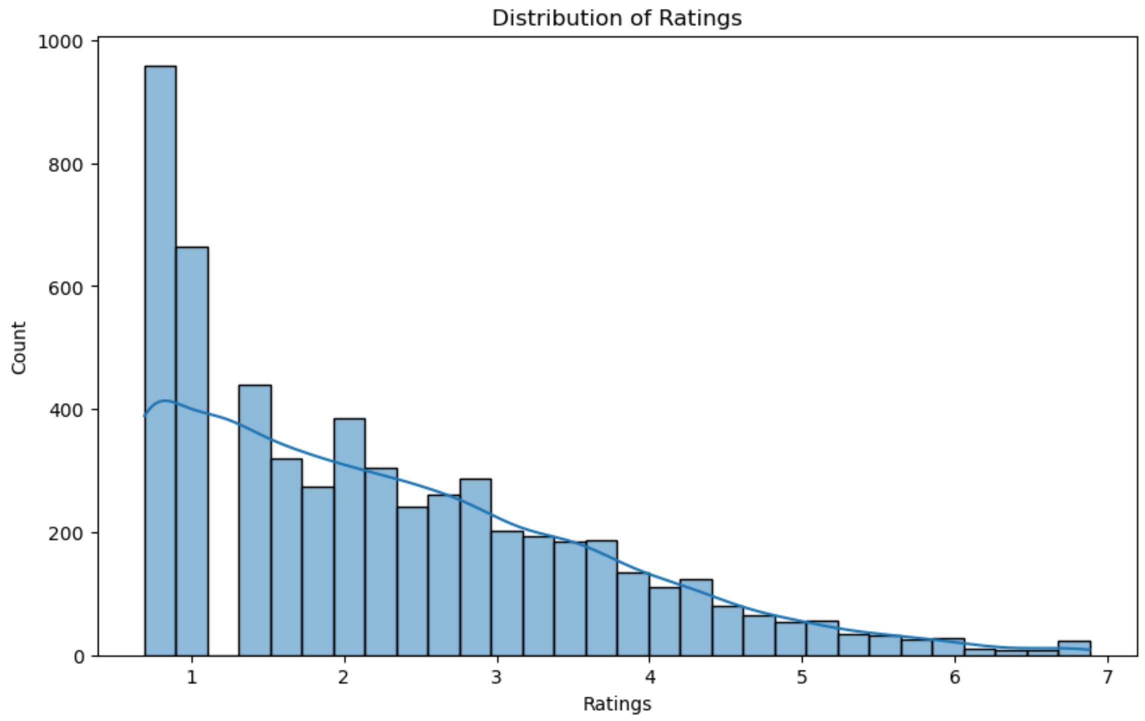
Categorical Variables Handling:

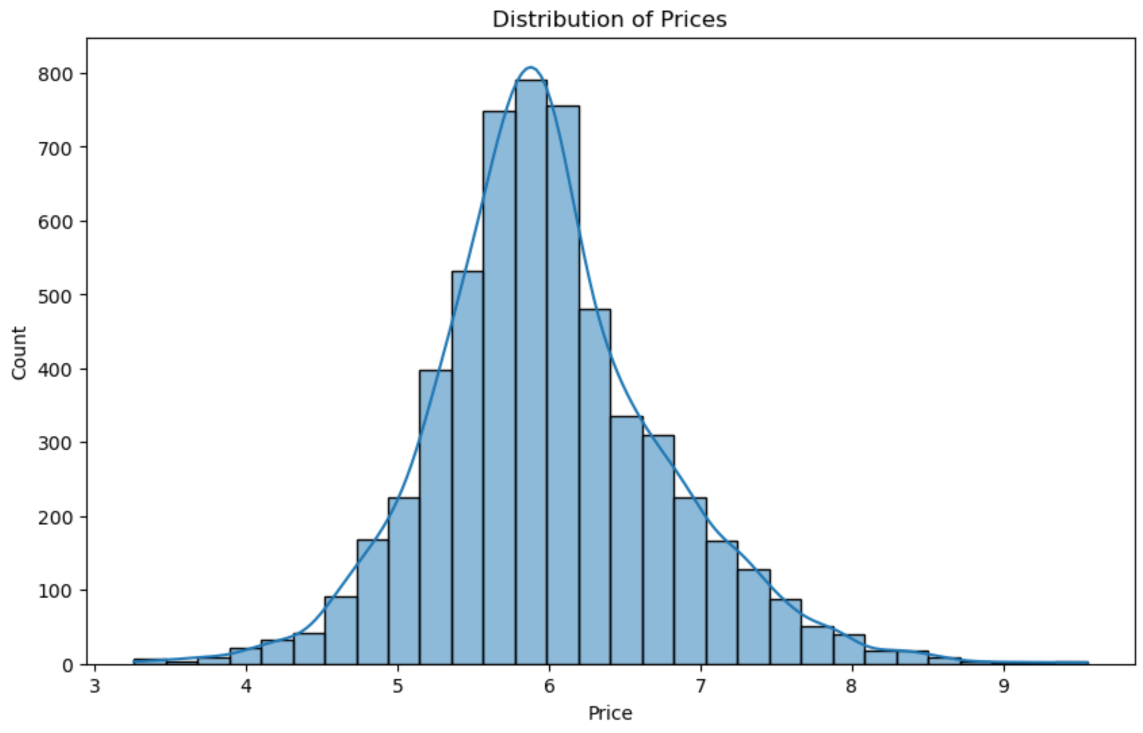
1. **'Edition' Feature Extraction:**
    - Extracted features ('Edition_Type', 'Edition_Details', 'Publication_Date') from the 'Edition' column.
    - Converted 'None' values to NaN for better representation.
2. **'Publication_Date' Feature Extraction:**
    - Extracted 'Day', 'Month', and 'Year' features from the 'Publication_Date' column.
3. **Conversion to Numeric:**
    - Converted 'Day', 'Month', and 'Year' columns to numeric format.
    - Mapped month names to numerical values for 'Month'.
    - Ensured 'Year' is numeric.
4. **Drop Columns:**
    - Dropped 'Edition' and 'Publication_Date' columns as they are no longer needed.
5. **Sample Data Creation:**
    - A copy of the original DataFrame ('sample_data') has been created for further actions without affecting the original data.

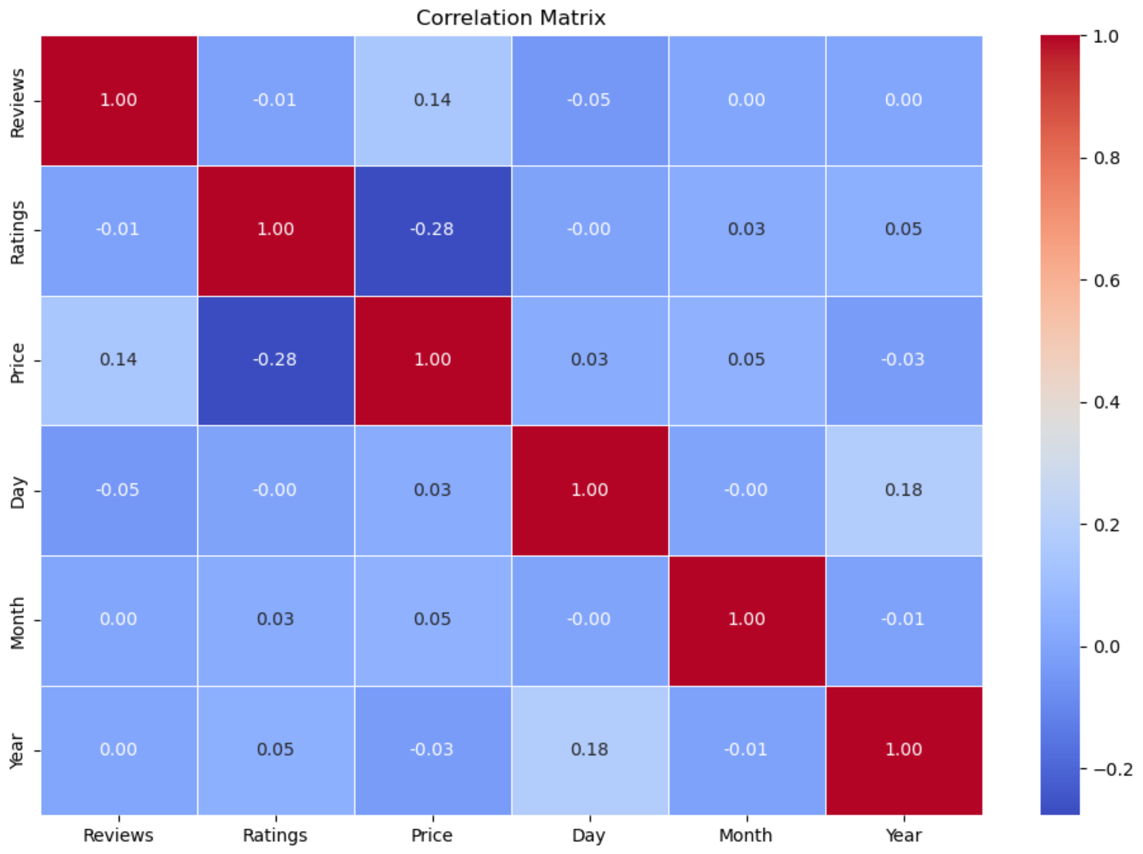
Basic Data Analysis Summary:

1. **Summary Statistics:**
    - Descriptive statistics for numeric features ('Reviews', 'Ratings', 'Price', 'Day', 'Month', 'Year').
    - Provides insights into the central tendency, spread, and distribution of the data.
2. **Distribution of Numeric Features:**
    - Histograms for 'Reviews', 'Ratings', and 'Price'.
    - Visualizes the distribution and spread of each numeric feature.
3. **Correlation Analysis:**
    - Correlation matrix and heatmap.
    - Reveals relationships between numeric features ('Reviews', 'Ratings', 'Price', 'Day', 'Month', 'Year').
    - Useful for identifying potential multicollinearity.
4. **Exploratory Data Analysis (EDA) for Categorical Features:**
    - Count plots for 'Genre' and 'BookCategory'.
    - Visualizes the distribution of books in each genre and category.
5. **Missing Values:**
    - Identifies missing values in columns ('Edition_Details', 'Day', 'Month', 'Year').
6. **Feature Relationships:**
    - Pairplot for numeric features.
    - Visualizes scatterplots and histograms for pairs of numeric features.

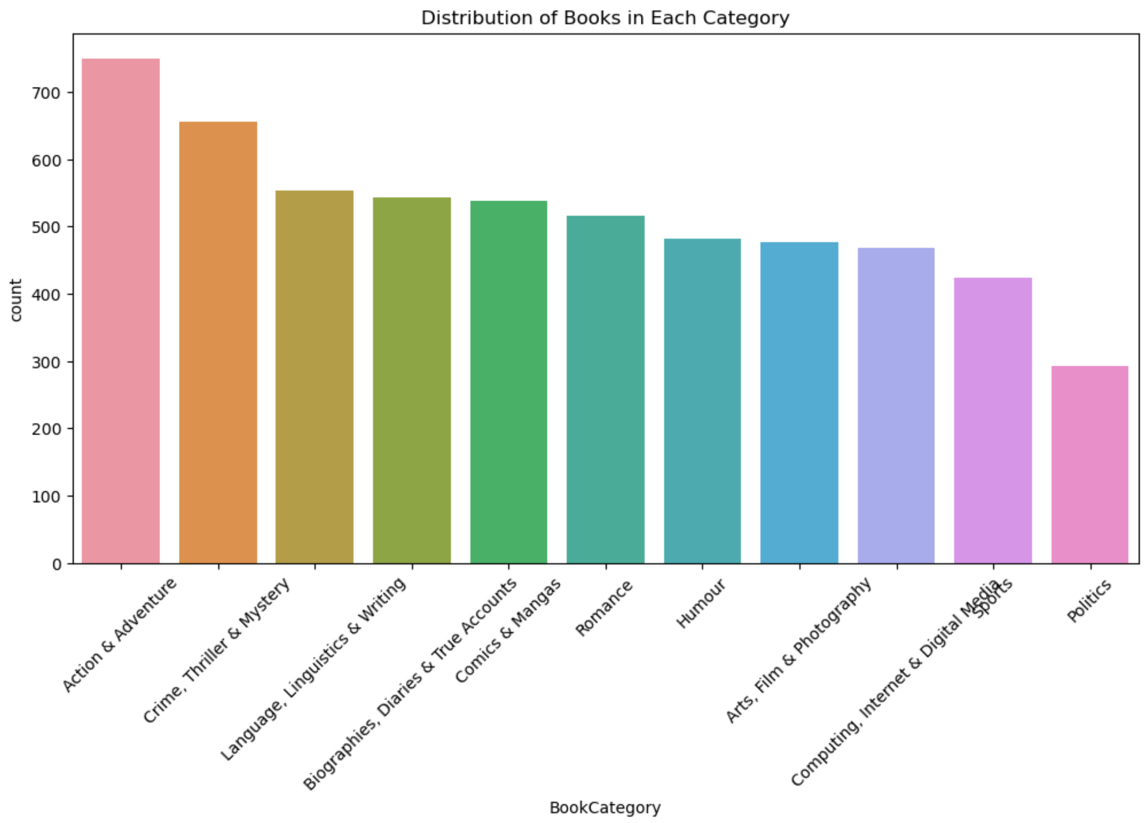

Distribution of Reviews
Count
Reviews
0
200
400
600
800
1000
1200
1400
1600
0.8
1.0
1.2
1.4
1.6
1.8

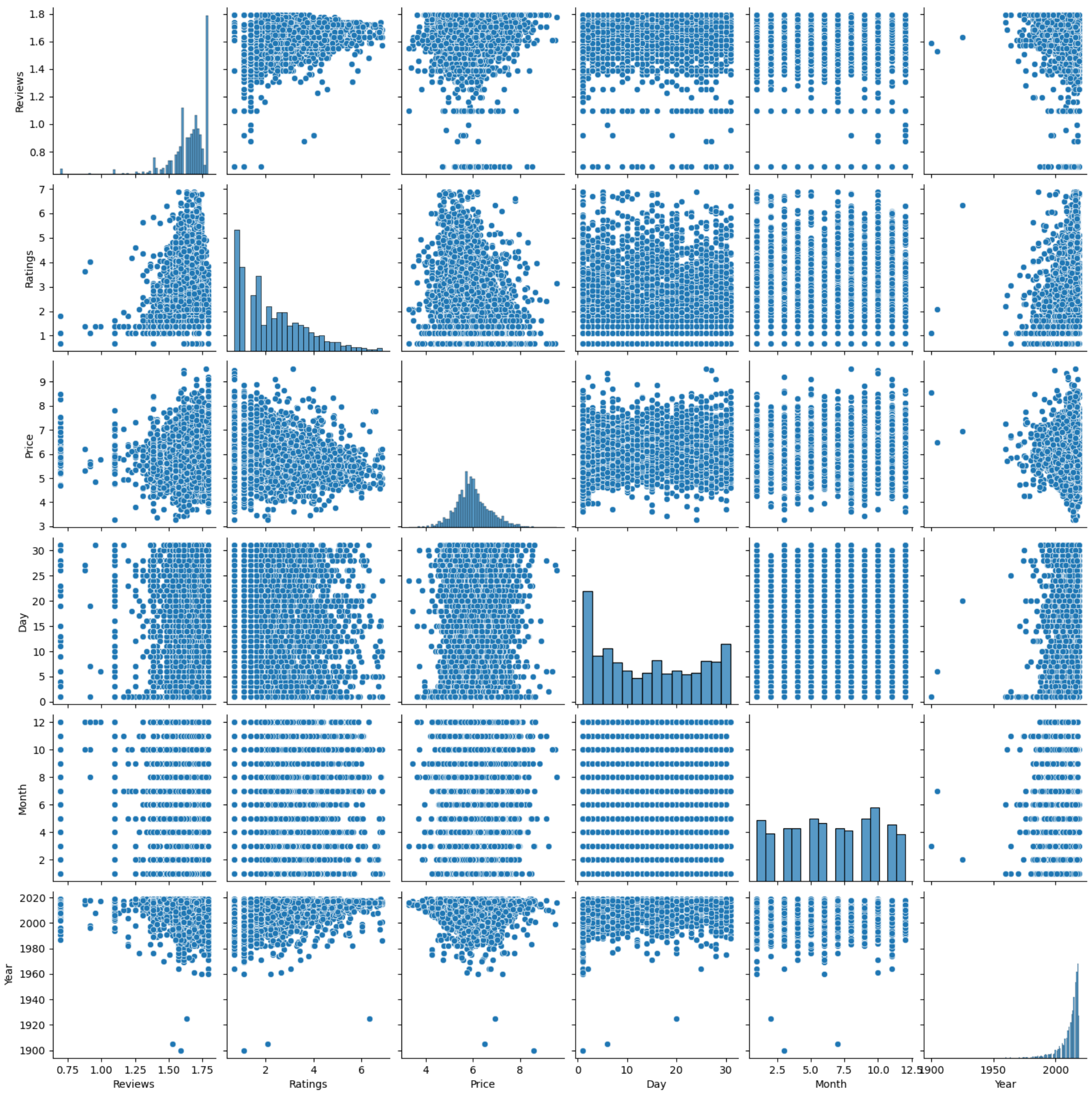

Distribution of Ratings
Count
Ratings
0
200
400
600
800
1000
1
2
3
4
5
6
7





	Reviews	Ratings	Price	Day	Month	Year
Reviews	1.00	-0.01	0.14	-0.05	0.00	0.00
Ratings	-0.01	1.00	-0.28	-0.00	0.03	0.05
Price	0.14	-0.28	1.00	0.03	0.05	-0.03
Day	-0.05	-0.00	0.03	1.00	-0.00	0.18
Month	0.00	0.03	0.05	-0.00	1.00	-0.01
Year	0.00	0.05	-0.03	0.18	-0.01	1.00


Distribution of Books in Each Category
count
700
600
500
400
300
200
100
0
Action & Adventure
Crime, Thriller & Mystery
Language, Linguistics & Writing
Biographies, Diaries & True Accounts
Comics & Mangas
Romance
Humour
Arts, Film & Photography
Computing, Internet & Digital Media
Sports
Politics
BookCategory


Reviews
Ratings
Price
Day
Month
Year

Exploring Edition_Type:

Unique values in 'Edition_Type' in the sample_data were printed.

Grouped the data by 'Edition_Type' and calculated the mean price for each group.

Sorted the 'Edition_Type' based on mean prices in descending order.

Weighted Average Prices for Edition_Type:

Removed rows with missing prices.

Calculated the weighted mean price for each 'Edition_Type' based on the existing prices.

Sorted the 'Edition_Type' based on weighted average prices in descending order.

Edition_Type Encoding:

Created a dictionary mapping 'Edition_Type' to its average price.

Created a new column 'Edition_Type_Encoded' using the average prices as labels.

Exploring Edition_Details:

Displayed unique values in 'Edition_Details' in sample_data.

Checked the info of the DataFrame to understand the data types and missing values.

Creating BookCategory Encoded Feature:

Calculated the weighted mean price for each 'BookCategory.'

Sorted the 'BookCategory' based on weighted average prices in descending order.

Created a dictionary to map categories to their corresponding weighted average prices.

Created a new column 'BookCategory_Encoded' using the mapping.

Identified the most expensive category.

Creating Genre Encoded Feature:

Calculated the weighted mean price for each 'Genre.'

Created a dictionary to map genres to their corresponding weighted average prices.

Created a new column 'Genre_Encoded' using the mapping.

Created a binary column 'Has_Edition_Details' indicating the presence of edition details.

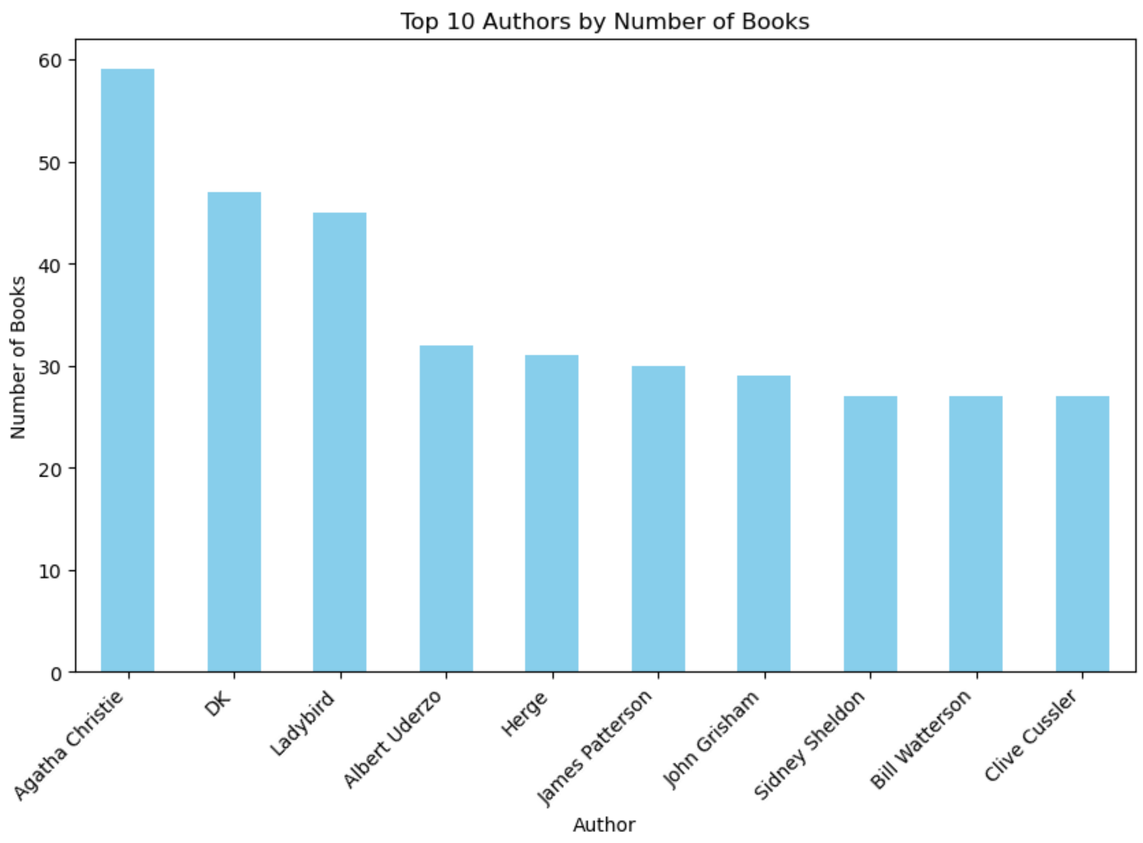
Removing Unnecessary Columns:

Dropped unnecessary columns like 'Genre,' 'BookCategory,' 'Edition_Type,' and 'Edition_Details.'

Feature Engineering - Synopsis Length:

Created a new feature 'Synopsis_Length' based on the length of the 'Synopsis' text.

Created new features 'Title-length' and 'Author-length' based on the lengths of 'Title' and 'Author' strings.



Age Calculation:

Calculated the age of the book by subtracting the publication year from the current year.

Day of the Week Encoding:

Extracted the day of the week from the date and performed one-hot encoding on it.

Text Features:

Created new features for the length of the synopsis, title, and author.

Calculated the number of words in the synopsis.

Polynomial Features:

Selected numerical features and applied polynomial transformation of degree 2.

Created a DataFrame with polynomial features and concatenated it with the original DataFrame.

Interaction Features:

Created interaction features by multiplying pairs of numerical features.

Concatenated the new interaction features with the original DataFrame.

Synopsis Word Count:

Calculated the number of words in the synopsis.

Resulting DataFrame:

The final DataFrame has 90 columns, including original features, one-hot encoded day of the week, text features, polynomial features, and interaction features.

TF-IDF Features Extraction:

Combining Text Columns:

Combined 'Title' and 'Synopsis' into a new column 'Text_combined'.

TF-IDF Vectorization:

Used **TfidfVectorizer** from **sklearn.feature_extraction.text** to convert 'Text_combined' into TF-IDF features.

Limited the number of features to 100 and removed English stop words.

Updating DataFrame:

Created a DataFrame (**tfidf_features_df**) with TF-IDF features.

Concatenated the new TF-IDF features to the original DataFrame.

Resulting DataFrame:

The DataFrame (**sample_data**) now has 191 columns, including the original features, one-hot encoded day of the week, text features, polynomial features, interaction features, and TF-IDF features.

Word Embeddings Features Extraction:

GloVe Word Embeddings Loading:

Loaded pre-trained GloVe word embeddings using **gensim.models.KeyedVectors**.

Sentence Embeddings using GloVe:

Defined a function (**get_sentence_embedding**) to create sentence embeddings using GloVe.

Applied the function to create embeddings for 'Title' and 'Synopsis'.

Converted the embedding columns to NumPy arrays.

DataFrame with Embeddings:

Created DataFrames (**title_embeddings_df** and **synopsis_embeddings_df**) for the title and synopsis embeddings.

Concatenated the new embedding features to the original DataFrame.

Word2Vec Word Embeddings:

Word2Vec Training:

Used **Word2Vec** from **gensim.models** to train a Word2Vec model on the tokenized text from 'Title' and 'Synopsis'.

Document Embedding Function:

Defined a function (**get_doc_embedding**) to get the word embedding for a document.

Word Embedding Features:

Applied the function to create word embedding features for each document.

Concatenated the word embedding features to the original DataFrame.

Modeling Report:

Modeling Report (Before Scaling):

1. RandomForestRegressor Model:

- **Features Used:** All columns except 'Price'.
- **Data Split:** Train-test split with a test size of 10% and a random state of 42.
- **Model Training:**
  - Utilized RandomForestRegressor with squared error criterion and a random state of 42.
  - Handled missing values by filling them with zeros.
  - Trained the model on the training data.
- **Model Evaluation:**
  - Evaluated the model on both training and test datasets using Mean Squared Error (MSE).
  - Training MSE: 47793.34
  - Test MSE: 190838.32
- **Observations:**
  - The model shows a pattern of potential overfitting, with the training MSE significantly lower than the test MSE.

2. Gradient Boosting Model:

- Trained a GradientBoostingRegressor as an alternative model.
- Evaluated the model on both training and test datasets using MSE.
- Training MSE: 135196.50
- Test MSE: 151475.76
- Gradient Boosting did not outperform the RandomForestRegressor.

3. Stacking Model:

- Created a StackingRegressor with base models (RandomForest, GradientBoosting, DecisionTree) and a final LinearRegression estimator.
- Trained the stacking model on the training data.

- Evaluated the model on both training and test datasets using MSE.
- Training MSE: 72390.43
- Test MSE: 147349.05
- Stacking model performance is between RandomForest and Gradient Boosting.

4. Feature Importance Analysis (Before Scaling):

- Accessed feature importances after training the RandomForestRegressor.
- Identified the most important features contributing to the model's predictions.
- Sorted features based on importance.
- **Top Features:**
    1. Edition_Type_Encoded
    2. BookCategory_Encoded_x_Genre_Encoded
    3. poly_30
    4. set
    5. Book_Age_x_Genre_Encoded

Modeling Report (After Scaling):

1. RandomForestRegressor Model:

- **Features Used:** All columns except 'Price'.
- **Data Split:** Train-test split with a test size of 10% and a random state of 42.
- **Model Training:**
    - Utilized RandomForestRegressor with squared error criterion and a random state of 42.
    - Handled missing values by filling them with zeros.
    - Trained the model on the training data.
- **Model Evaluation:**
    - Evaluated the model on both training and test datasets using Mean Squared Error (MSE).
    - Training MSE: 0.0396
    - Test MSE: 0.2407

- **Observations:**
    - The model continues to exhibit good performance on the training data.
    - The test MSE remains significantly higher, indicating potential overfitting.

2. Feature Importance Analysis (After Scaling):

- Accessed feature importances after training the RandomForestRegressor.
- Identified the most important features contributing to the model's predictions.
- **Top Features:**
    - Feature importance analysis remains consistent with the unscaled model.

3. Conclusion:

- The RandomForestRegressor model, even after scaling, shows a pattern of potential overfitting.
- Additional steps such as hyperparameter tuning, cross-validation, or regularization might be considered.
- Feature importance analysis provides insights into crucial predictors for book prices.

- **<u>Further investigation into error patterns on the test data and model improvement strategies is needed.</u>**