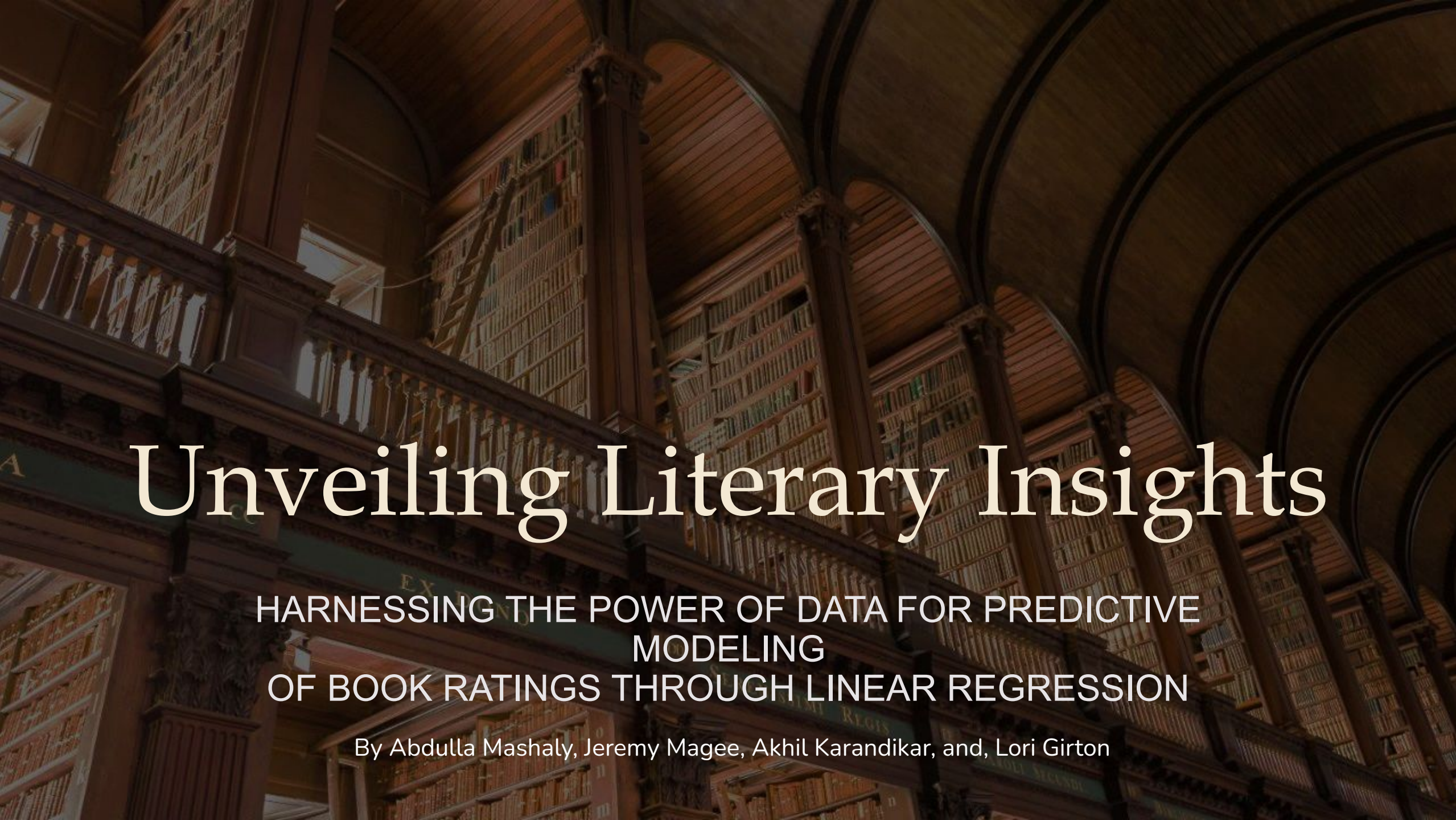


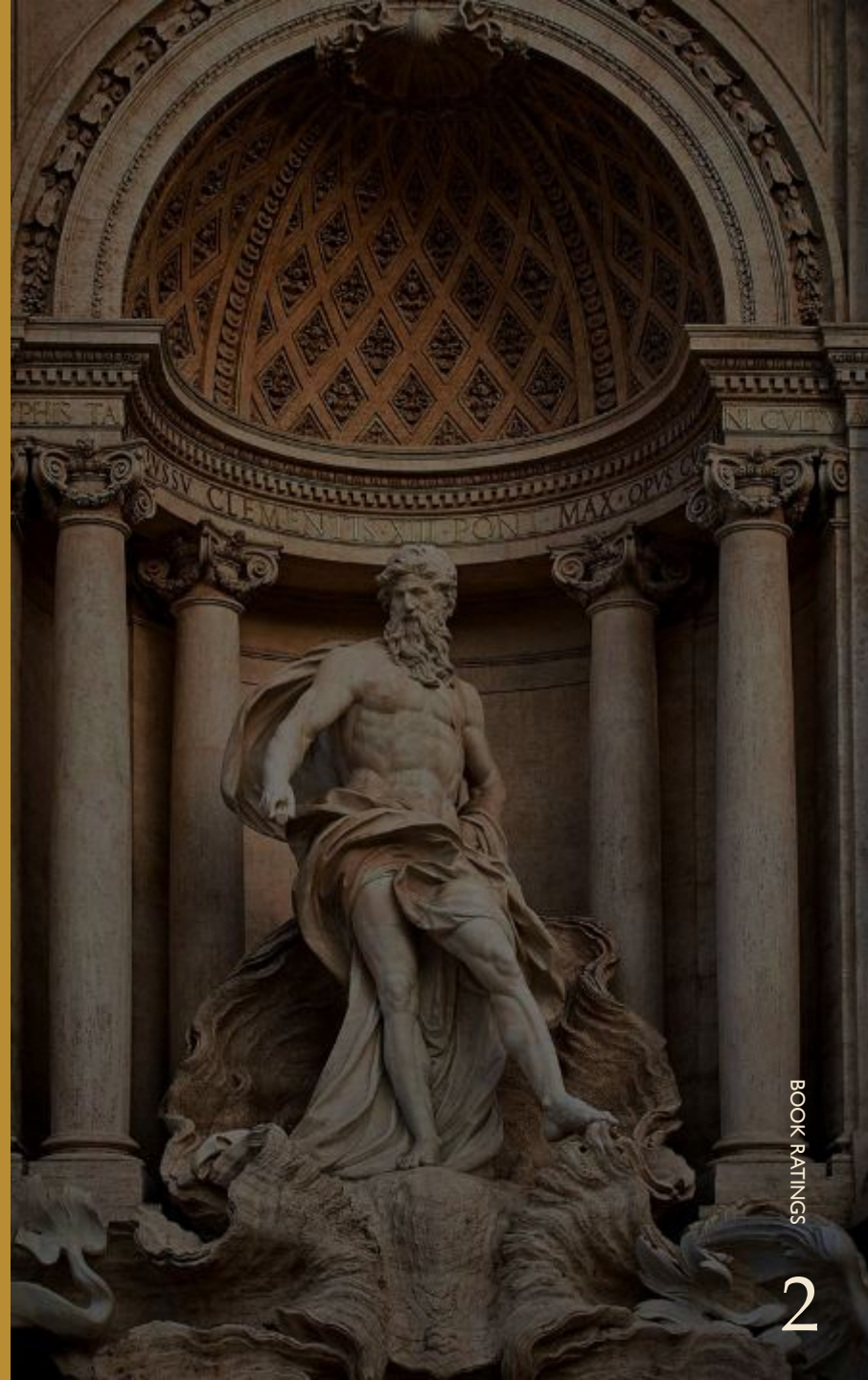
Unveiling Literary Insights

HARNESSING THE POWER OF DATA FOR PREDICTIVE MODELING OF BOOK RATINGS THROUGH LINEAR REGRESSION

By Abdulla Mashaly, Jeremy Magee, Akhil Karandikar, and, Lori Girton

Our Epic Odyssey

- Initial idea was to make a prediction engine for book recommendations based on user input
- Identified challenges in measuring accuracy
- Shifted focus to predict book ratings using extensive API data
- Experimented with various models, ultimately adopting a linear regression learning model
- Explored random forest classification, achieving successful outcomes
- Commenced optimization efforts



Sample Data

- Our API was from Goodreads (via Kaggle), an open data source with multitudes of information about a given book.
- This dataset includes everything about each book from ISBN, average author ratings, number of times a book appears on a to-read list, number of pages, etc.

```
root
 |-- authors: array (nullable = true)
 |    |-- element: struct (containsNull = true)
 |    |    |-- author_id: string (nullable = true)
 |    |    |-- role: string (nullable = true)
 |-- average_rating: float (nullable = true)
 |-- book_id: string (nullable = true)
 |-- format: string (nullable = true)
 |-- isbn13: string (nullable = true)
 |-- num_pages: integer (nullable = true)
 |-- popular_shelves: array (nullable = true)
 |    |-- element: struct (containsNull = true)
 |    |    |-- count: string (nullable = true)
 |    |    |-- name: string (nullable = true)
 |-- publication_year: string (nullable = true)
 |-- ratings_count: integer (nullable = true)
 |-- series: array (nullable = true)
 |    |-- element: string (containsNull = true)
 |-- text_reviews_count: integer (nullable = true)
```

Data Collection and Cleaning

PYSPARK AND SQL!

- Our initial dataset had a size of 2 GB, which was unwieldy for the computing power and memory we have on hand.
- PySpark was irreplaceable for caching our info into a manageable dataframe so that we could decide on necessary features for our model.

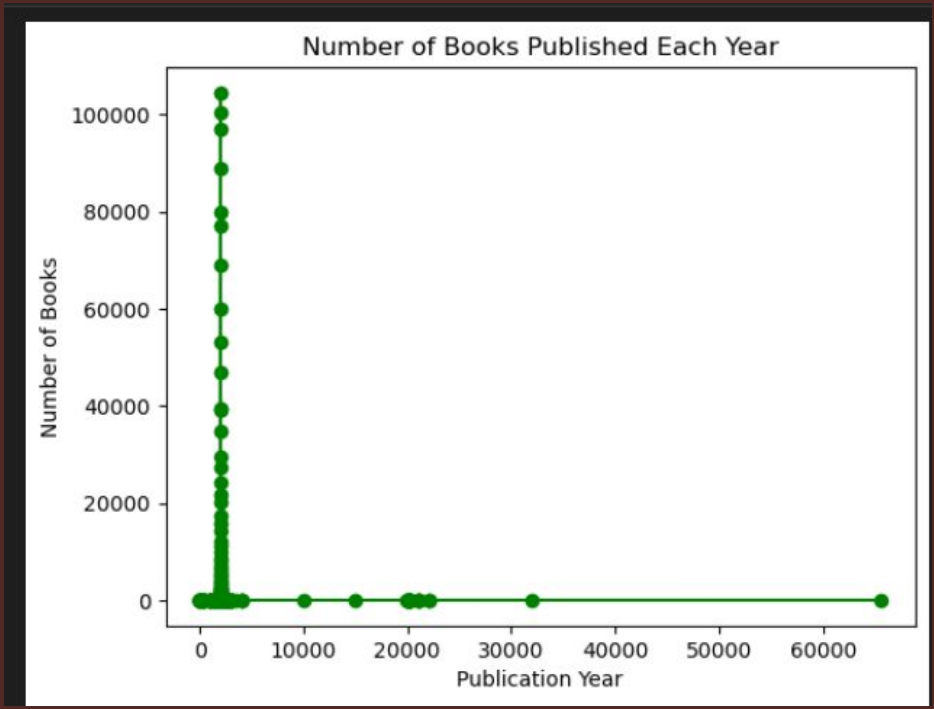
DATA CLEANING:

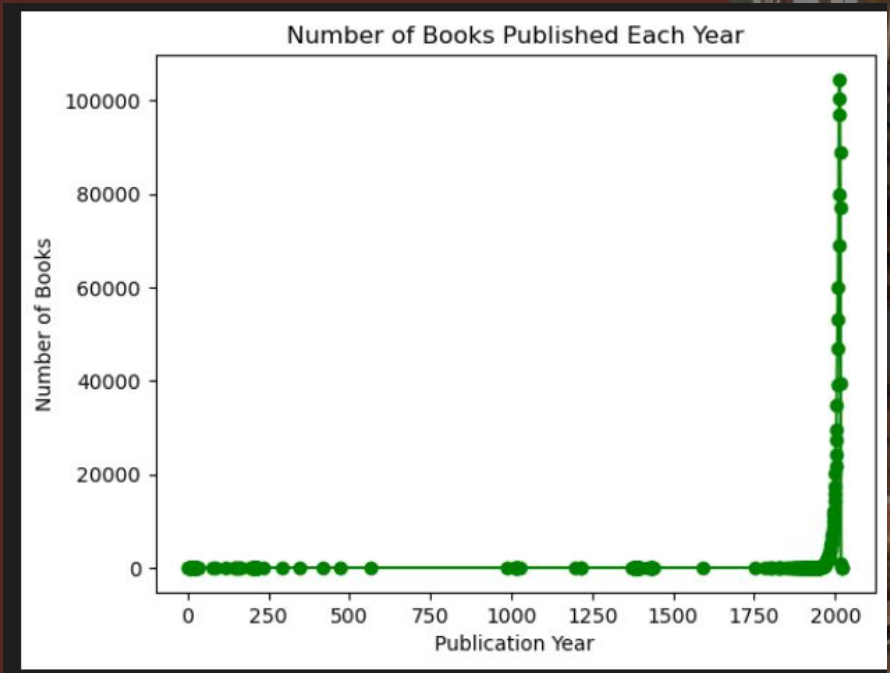
- Data cleaning is necessary for the proper EDA needed to make a model
- We cleaned the data and found many curious and odd entries:
- Some examples:

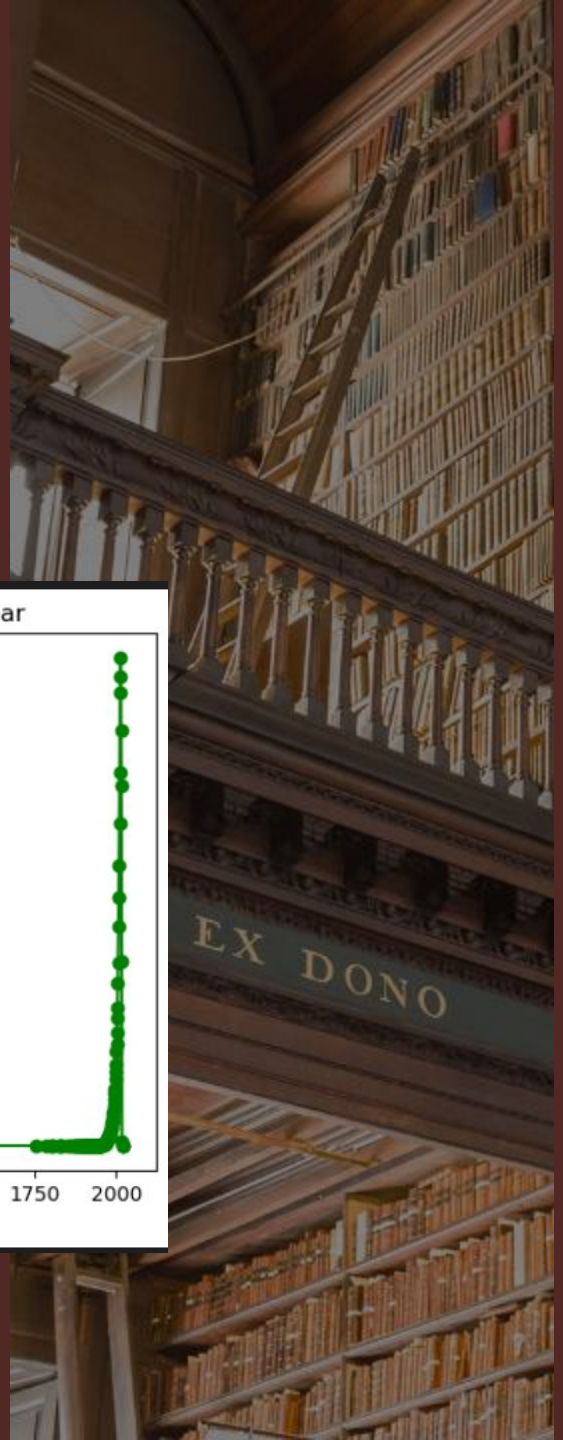
Page Count	Published Year	Ratings
• 0 • 8,345	• 5 (yes, 5) • 56204	. 0

Data Exploration & Analysis

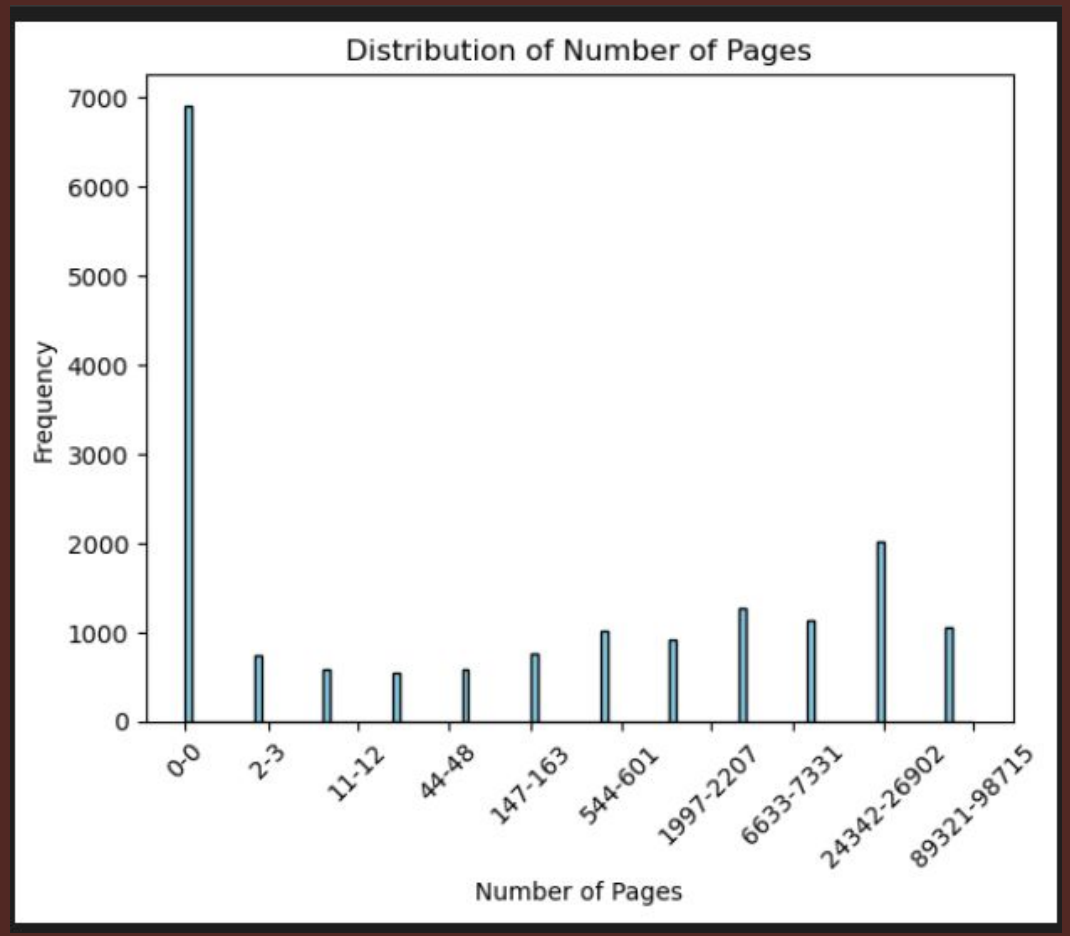
EDA
Step 1


Number of Books Published Each Year
Number of Books
100000
80000
60000
40000
20000
0
0
10000
20000
30000
40000
50000
60000
Publication Year


Number of Books Published Each Year
Number of Books
100000
80000
60000
40000
20000
0
0
250
500
750
1000
1250
1500
1750
2000
Publication Year

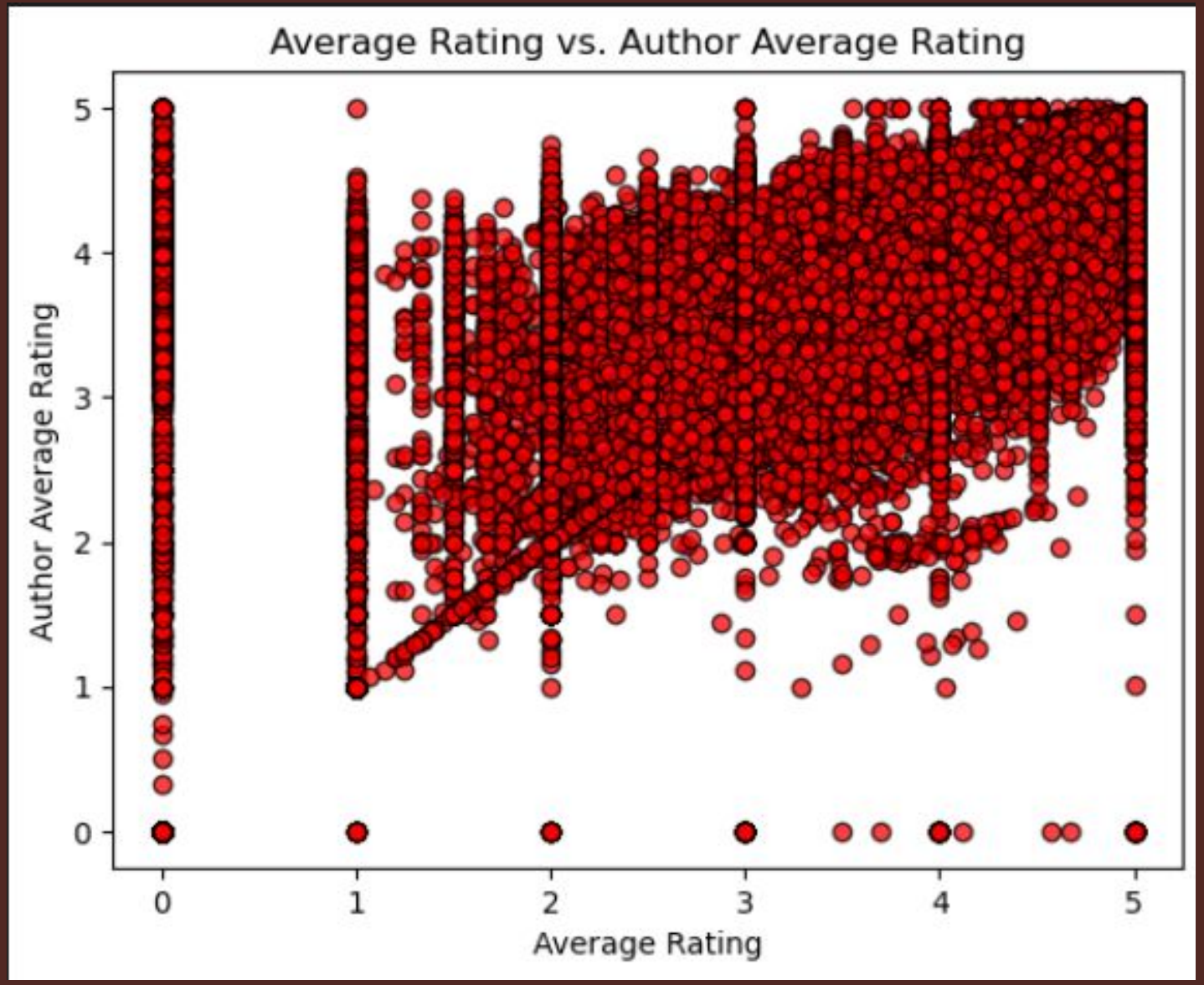

EX DONO

EDA Step 1


Distribution of Number of Pages
Frequency
7000
6000
5000
4000
3000
2000
1000
0
0-0
2-3
11-12
44-48
147-163
544-601
1997-2207
6633-7331
24342-26902
89321-98715
Number of Pages

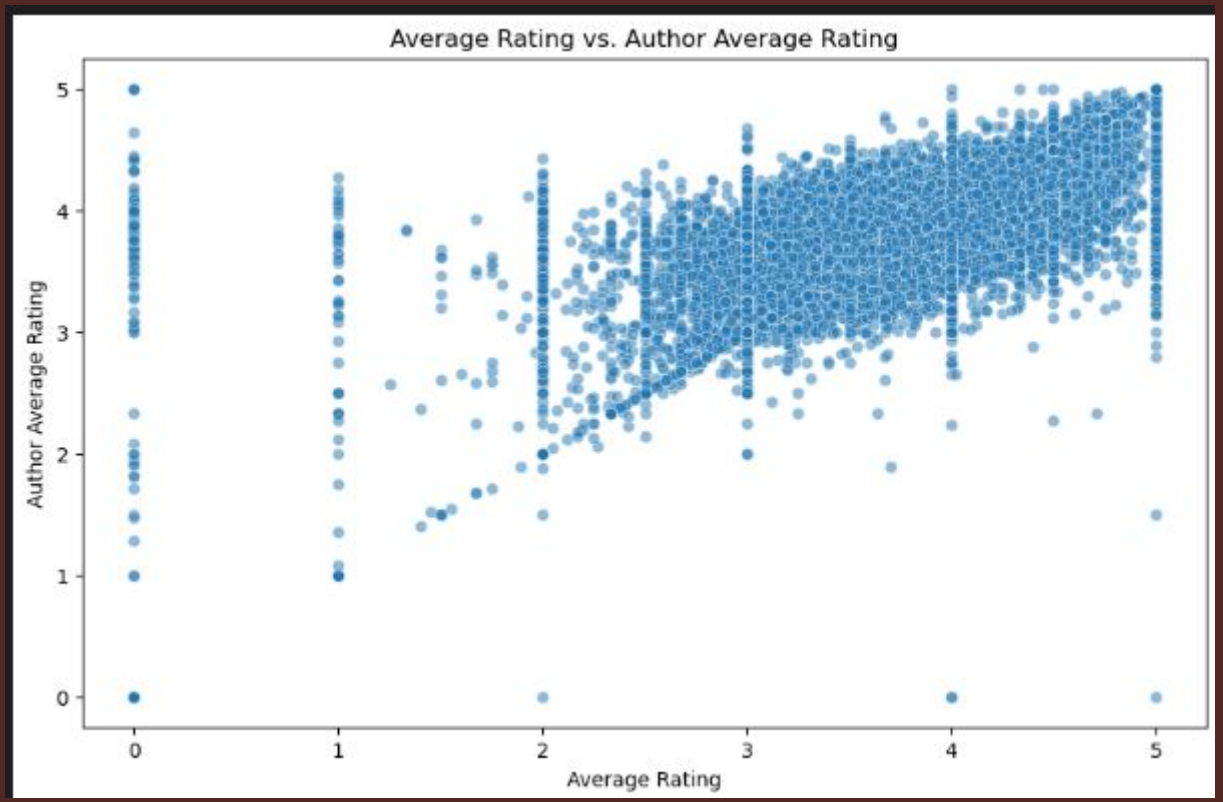

EX DONO

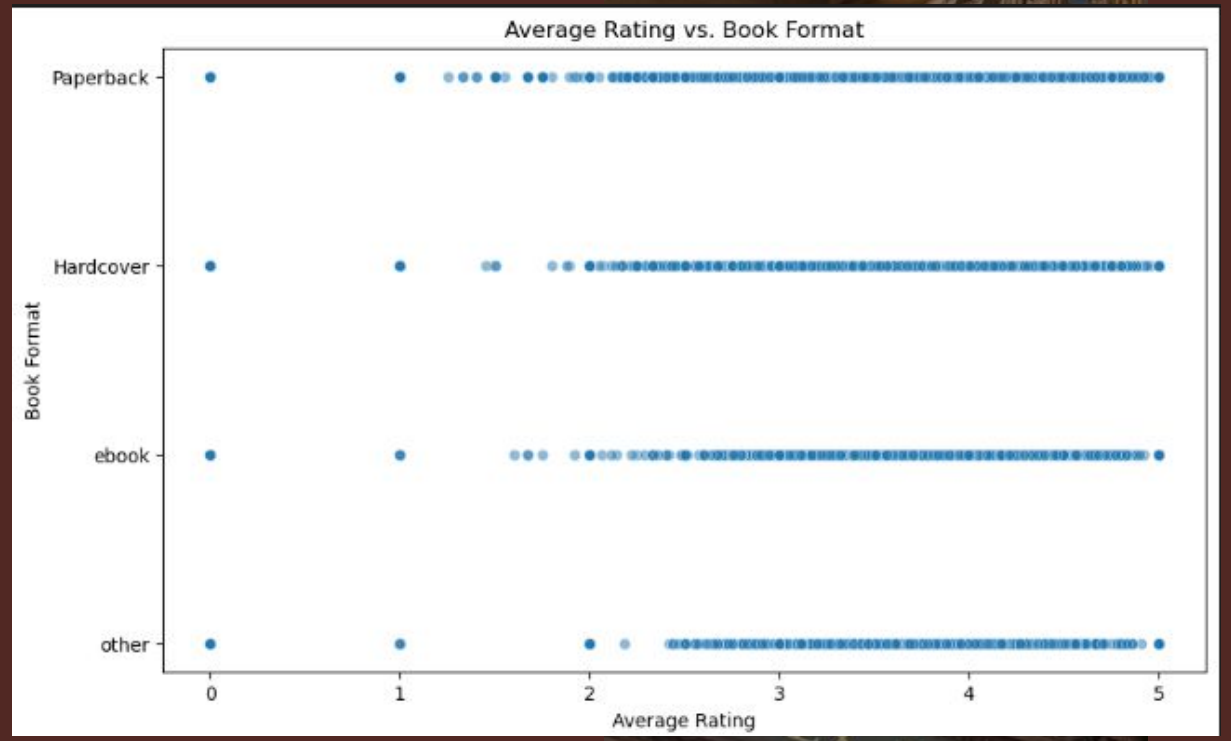
EDA Step 2


Average Rating vs. Author Average Rating
Author Average Rating
Average Rating
0
1
2
3
4
5

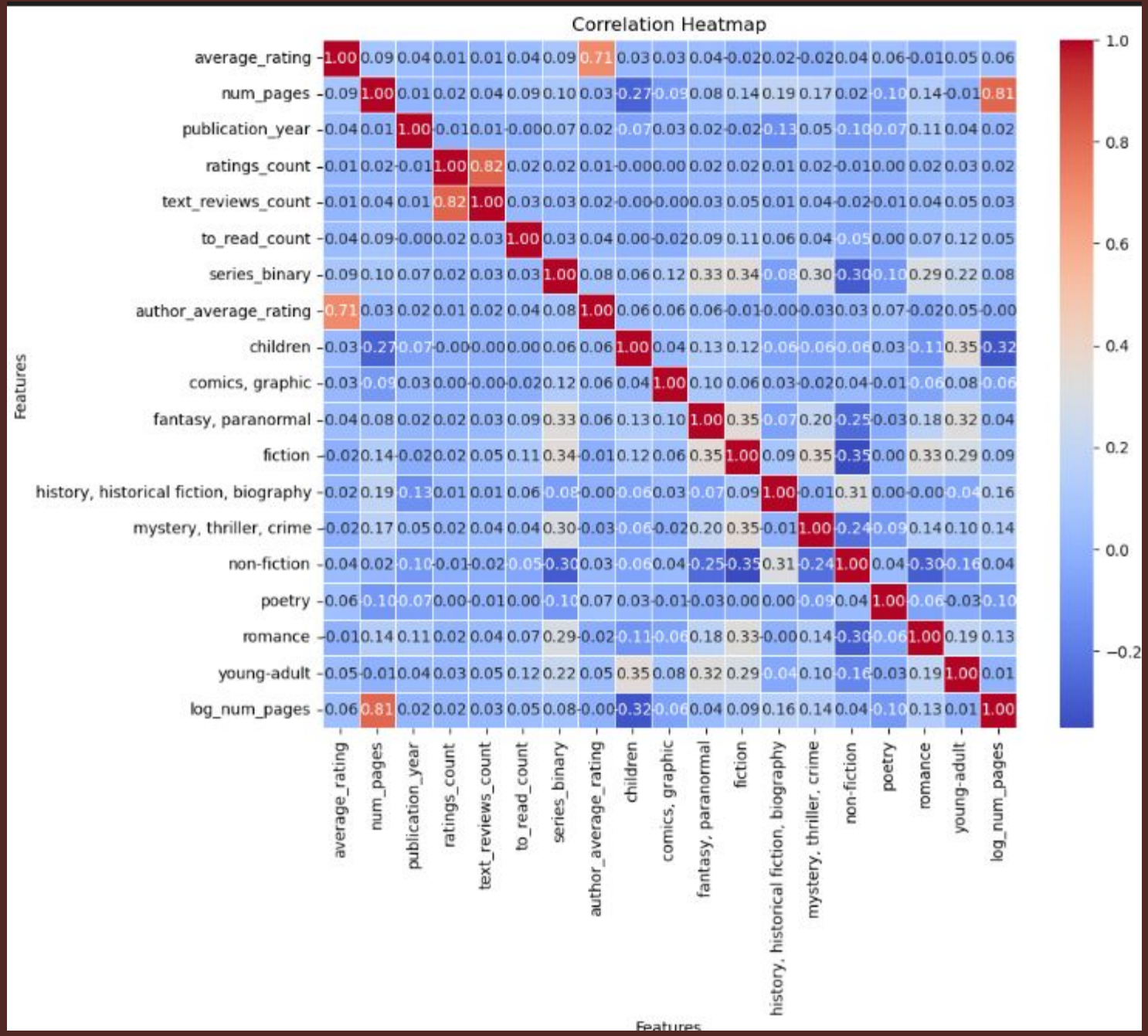

EX DONO

EDA: Sample 3 Data



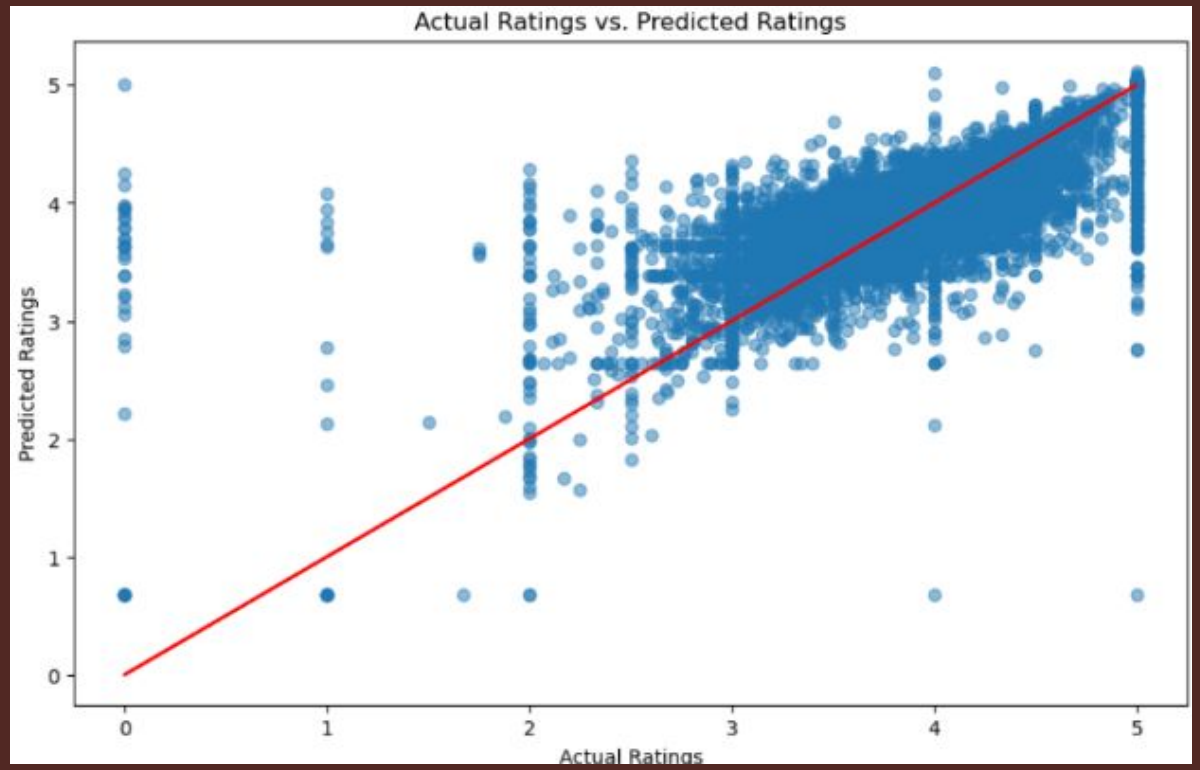


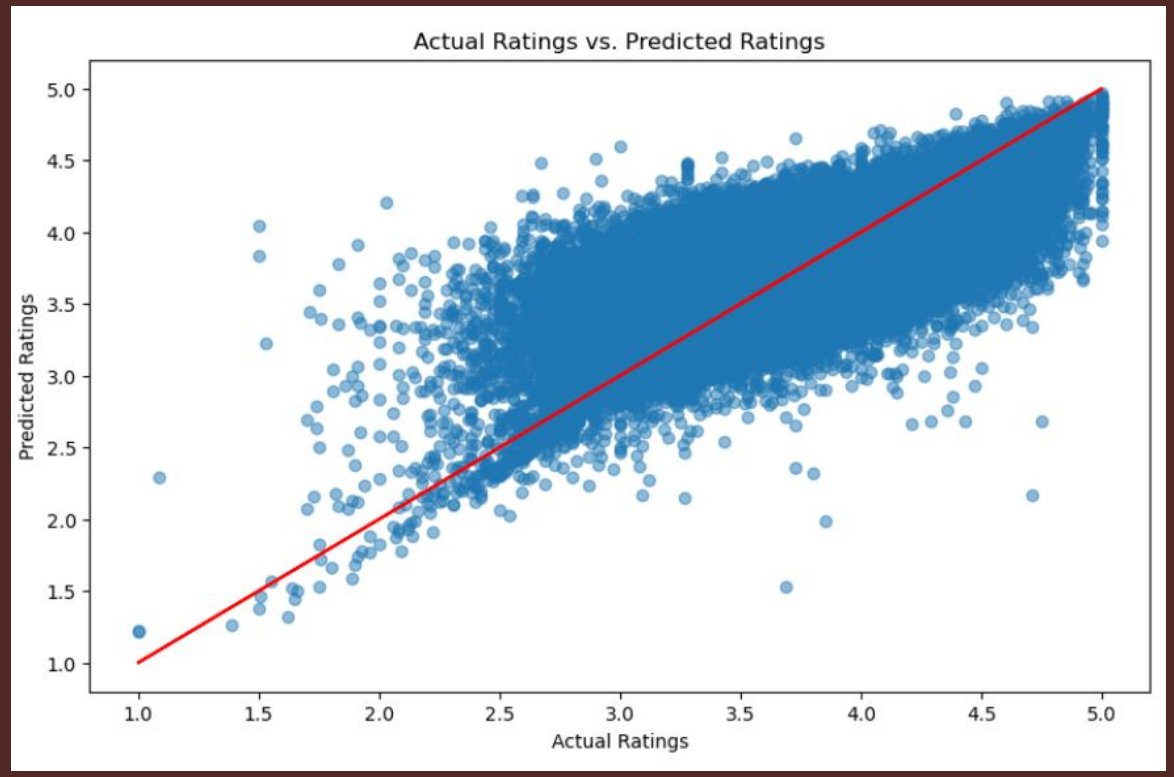
EDA: Sample 3 Data



Model Optimization & Selection

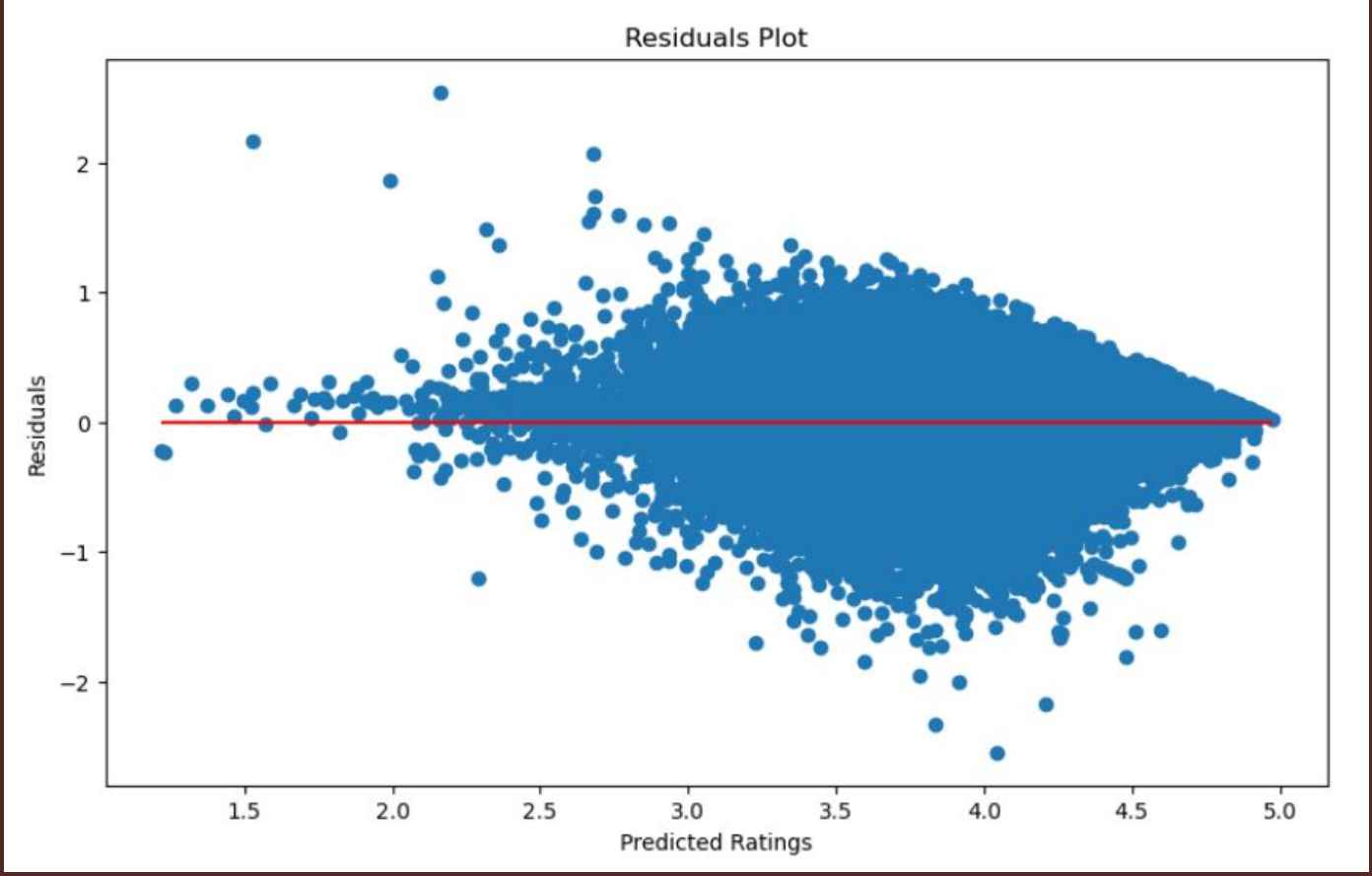
Model 1: Neural Network



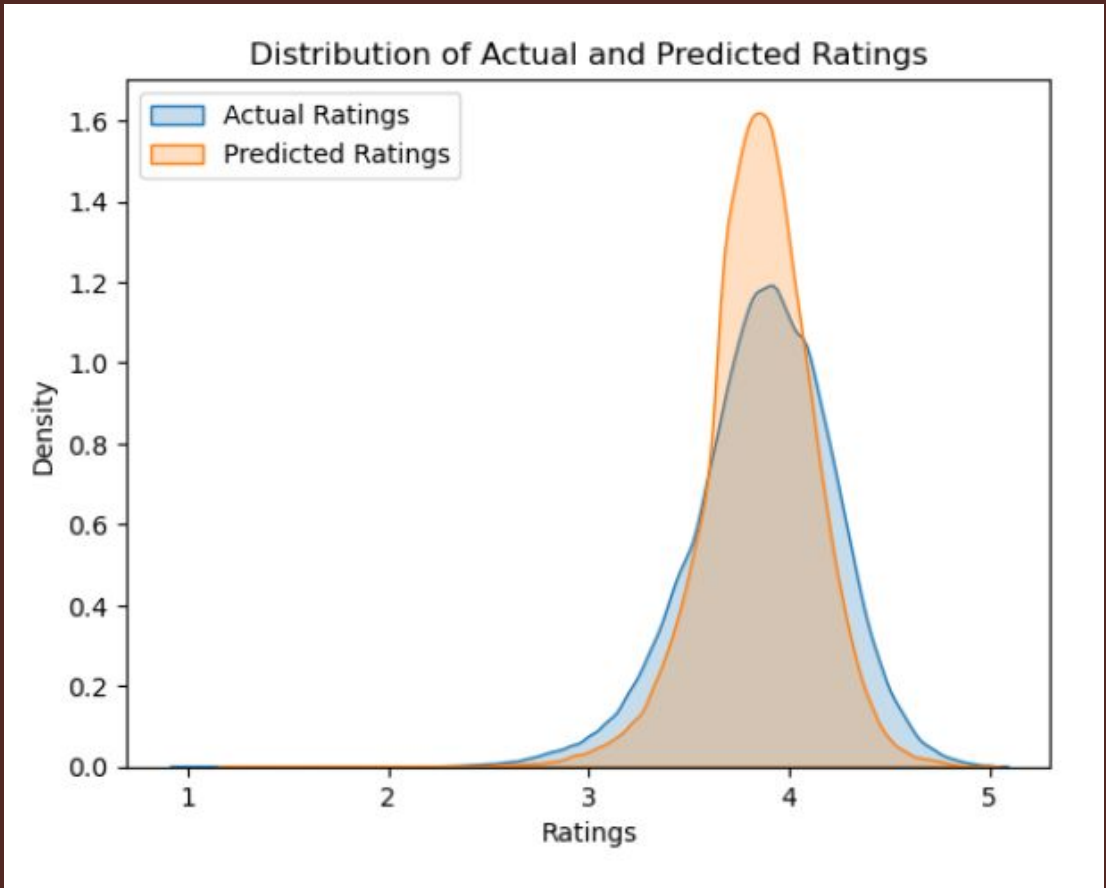


BOOKS RATINGS

Model 1: Neural Network



Model 1: Neural Network



Model 2: Linear Regression

```
#Apply the imputer to the testing set
X_test_imputed = imputer.transform(X_test)

#Model Evaluation
y_pred = model.predict(X_test_imputed)

mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'Mean Squared Error: {mse}')
print(f'R-squared: {r2}')
```

```
Mean Squared Error: 0.1177539659888109
R-squared: 0.5152547969739771
```

Model 2: Linear Regression

```
#Apply the imputer to the testing set
X_test_imputed = imputer.transform(X_test)

#Model Evaluation
y_pred = model.predict(X_test_imputed)

mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'Mean Squared Error: {mse}')
print(f'R-squared: {r2}')
```

```
Mean Squared Error: 0.1181195196657872 9
R-squared: 0.4844898849903885
```

Model 3: Random Forest

Confusion Matrix

	Predicted 0	Predicted 1	Predicted 2	Predicted 3	Predicted 4
Actual 0	37	1	6	5	2
Actual 1	2	17	15	11	5
Actual 2	3	2	195	332	17
Actual 3	3	0	59	7848	1030
Actual 4	2	0	14	1656	3523

Accuracy Score : 0.7859316875211363

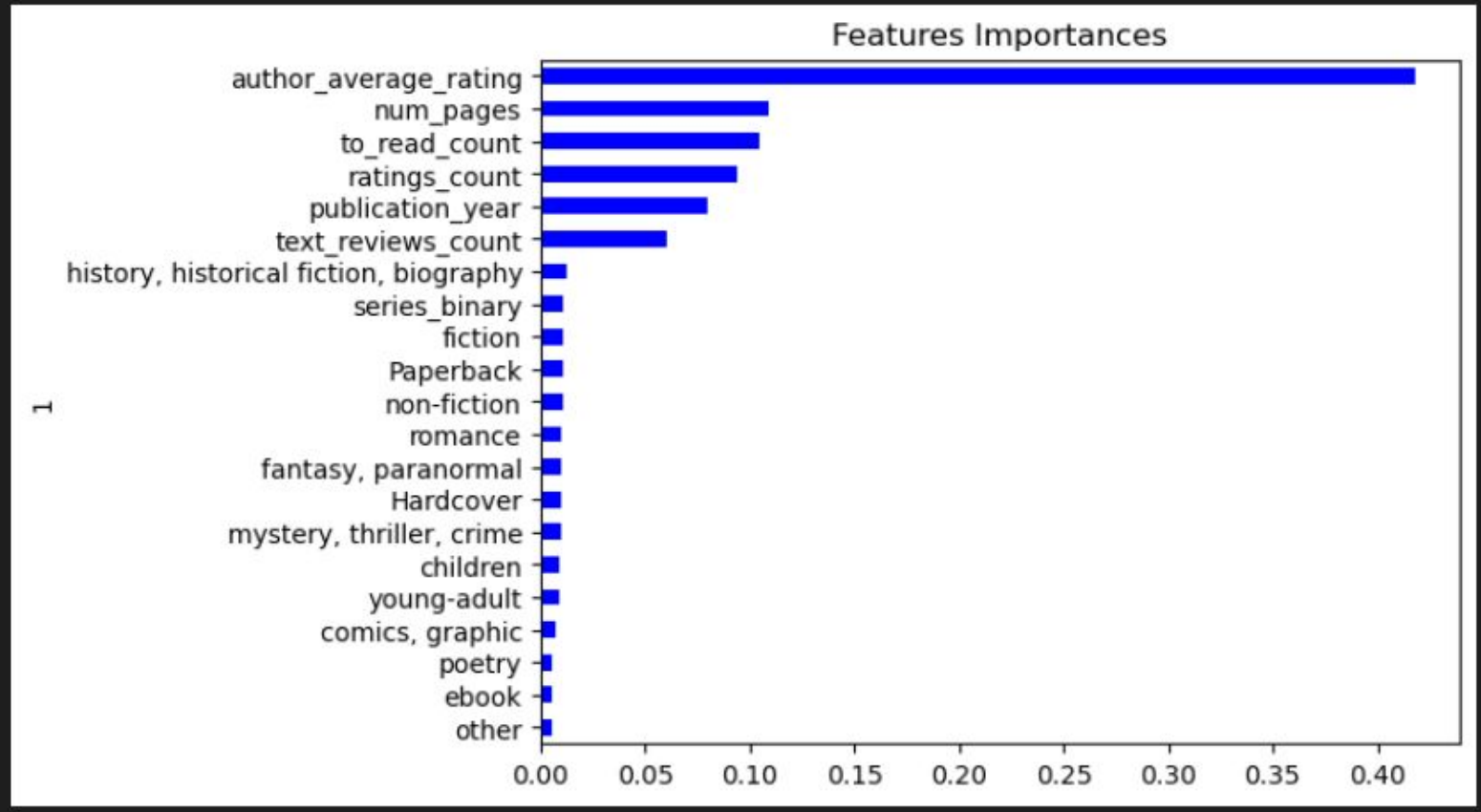
Classification Report

	precision	recall	f1-score	support
0	0.79	0.73	0.76	51
1	0.85	0.34	0.49	50
2	0.67	0.36	0.47	549
3	0.80	0.88	0.84	8940
4	0.77	0.68	0.72	5195
accuracy			0.79	14785
macro avg	0.78	0.60	0.65	14785
weighted avg	0.78	0.79	0.78	14785

Random Forest Model:

Feature Rankings

```
# Visualize the features by importance
importances_df = pd.DataFrame(sorted(zip(rf_model.feature_importances_, X.columns)))
importances_df.plot(x=1, y=0, kind='barh', color='blue', legend=None )
plt.title('Features Importances')
plt.show()
```



“BUT YOU, BRAVE AND ADEPT FROM THIS DAY ON . . . THERE’S HOPE THAT YOU WILL REACH YOUR GOAL . . . THE JOURNEY THAT STIRS YOU NOW IS NOT FAR OFF.”

- HOMER, THE ODYSSEY

CITATIONS

ARTICLES

- Mengting Wan, Julian McAuley, "Item Recommendation on Monotonic Behavior Chains", in RecSys'18 [bibtex}
- Mengting Wan, Rishabh Misra, Ndapa Nakashole, Julian McAuley, "Fine-Grained Spoiler Detection from Large-Scale Review Corpora", in ACL'19 [bibtex]

DATA

- Wan, Mengting. (2023).goodreads.GitHub.https://github.com/MengtingWan/goodreads
- Ahmad. (2023, October). Goodreads Book Reviews, Version 1. Retrieved November 22, 2023 from https://www.kaggle.com/datasets/pypiahmad/goodreads-book-reviews1