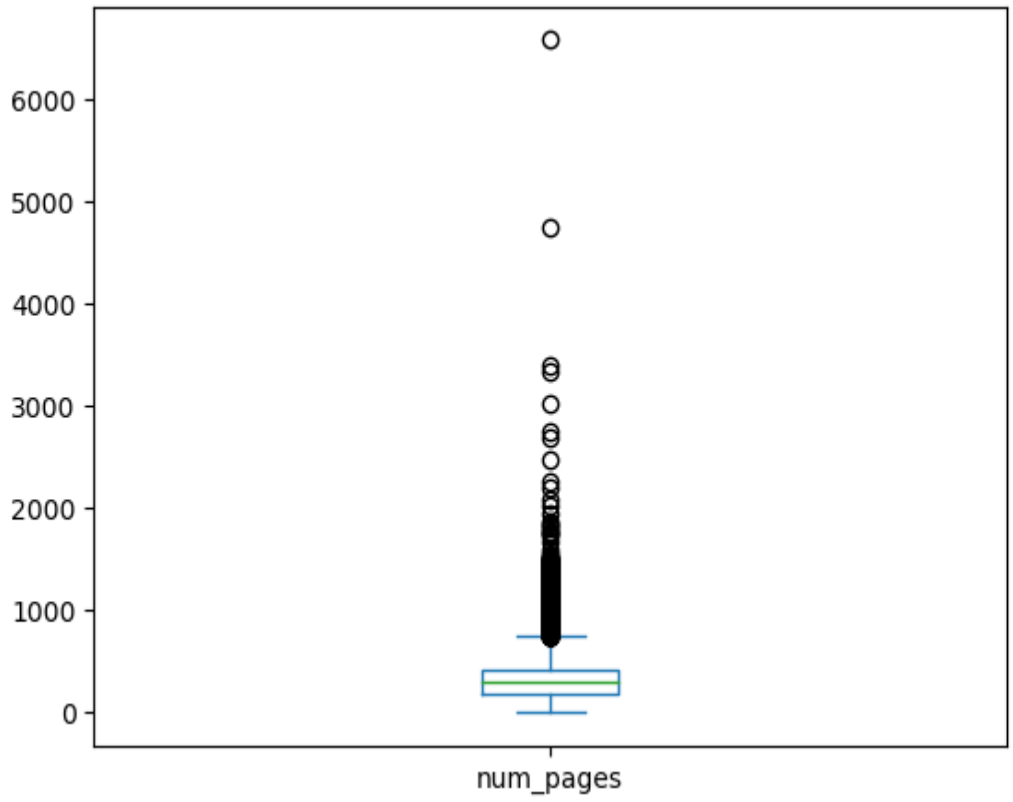
Project Report

Dataset: GoodReads
Link:https://github.com/KeepOnLearningStuff/Python-ML/blob/main/goodreads-ratings-prediction.ipynb
Team members: Aditya Bisht,Kaan Yüceel,Mónica Rojas,Yedige Ashmet
Best results: Aditya Bisht

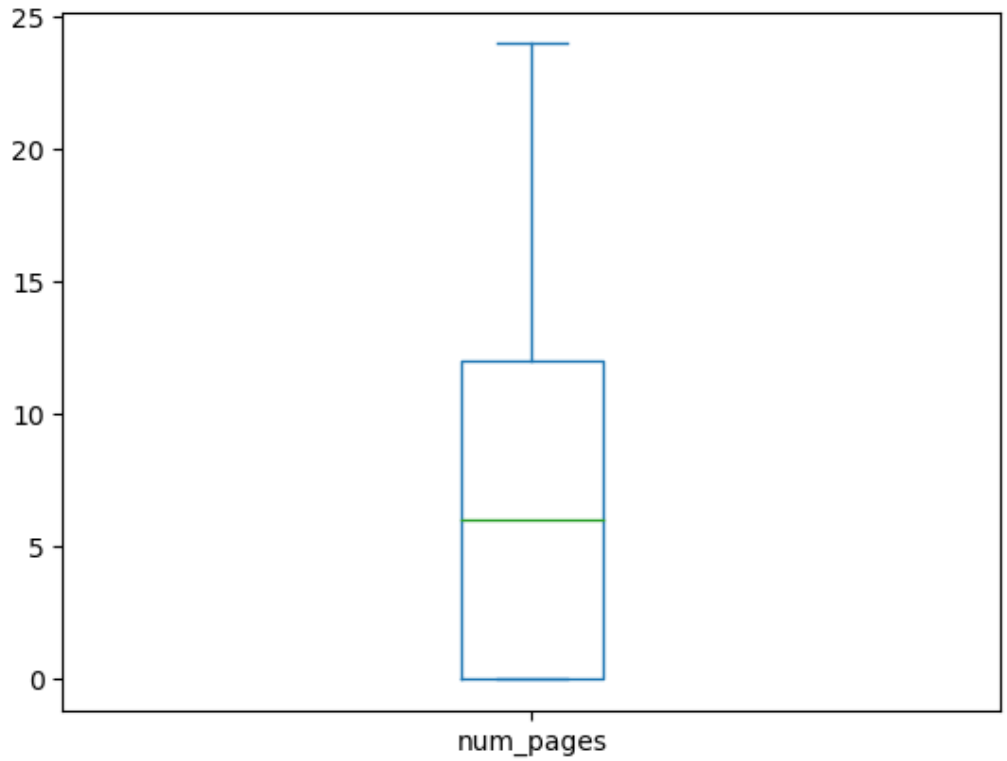
Approach:

The project begins by importing the necessary libraries and loading the dataset into a pandas dataframe. The dataset consists of over 11000 rows, with each row representing a book and its corresponding features such as bookID, title, authors, average_rating, isbn, isbn13, language_code, num_pages, ratings_count, text_reviews_count, publication_date and publisher. The data is then preprocessed, which includes removing duplicates(Picture 1), handling missing values(Picture 2 & 3), and encoding categorical variables(Picture 4).

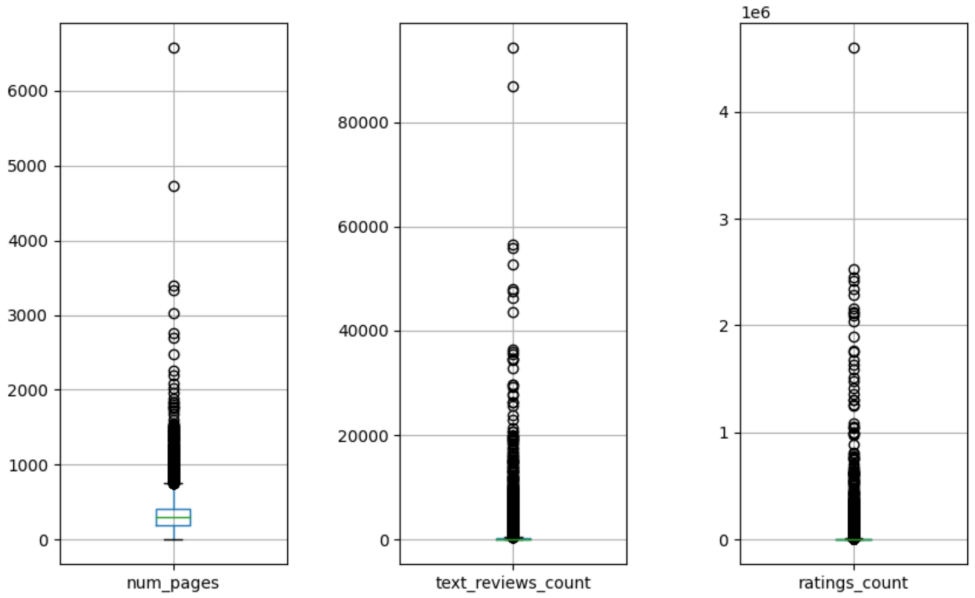
EDA:



Picture 1. Number of pages with outliers



Picture 2. Number of pages with removal of outliers



Picture 3. Main Feature Outliers

```
In [11]: # removing rows with 0 average ratings (and missing ratings count)
         df = df.loc[df.average_rating != 0]
         print(df.shape)

         (11098, 10)
```

```
In [13]:
# replace missing rating counts with the median
median_ratings_count = df.ratings_count.median()
df.ratings_count = df.ratings_count.map(lambda x: median_ratings_count if x == 0 else x)
print(df.ratings_count.value_counts())
```

```
3.0         82
1.0         76
2.0         71
4.0         71
5.0         61
            ..
21507.0      1
6970.0       1
108440.0     1
66503.0      1
783.0        1
Name: ratings_count, Length: 5294, dtype: int64
```

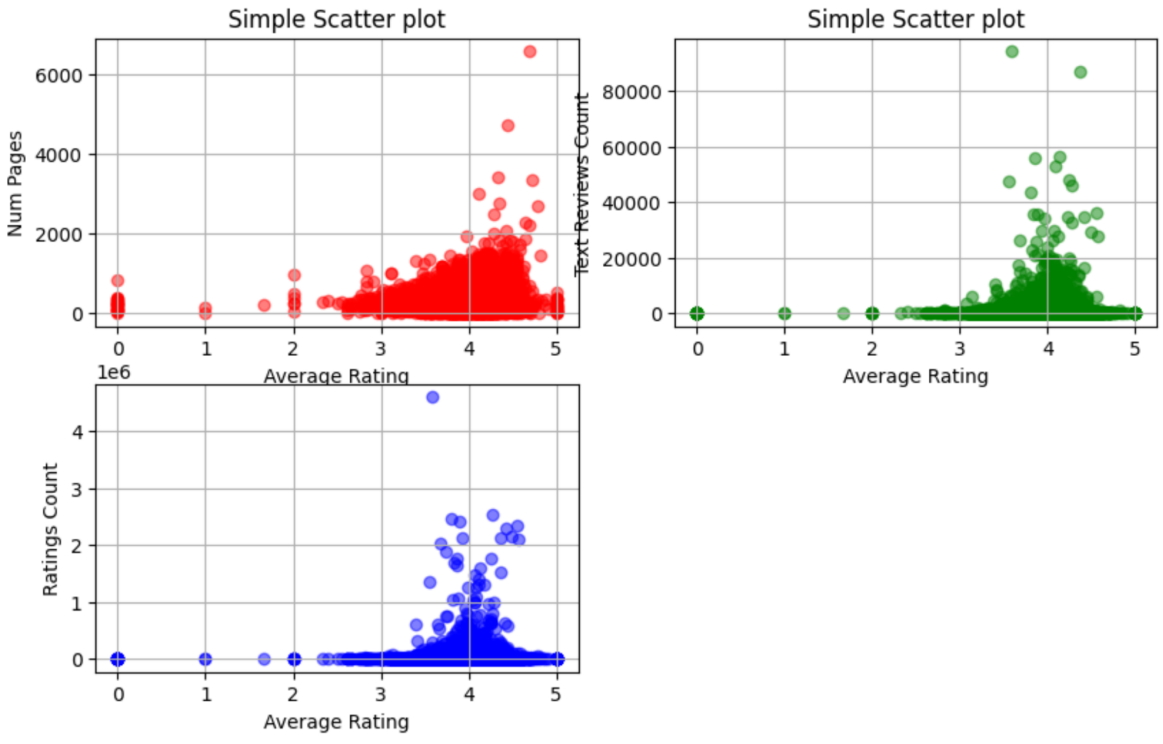
Picture 4. Handling missing values and some publication

```
In [29]:
# categorical label-level aggregation features

main_author_mean = df.groupby("main_author").mean()
main_author_count = df.groupby("main_author").count()
publisher_count = df.groupby("publisher").count()
publication_year_sum = df.groupby("publication_year").sum()
publication_month_sum = df.groupby("publication_month").sum()

df["english_book"] = df.language_code.apply(lambda x: 1 if x.startswith("eng") else 0)
df["main_author_book_count"] = df.main_author.apply(lambda x: main_author_count["isbn13"][x])
df["main_author_average_text_reviews_count"] = df.main_author.apply(lambda x: main_author_mean["ratings_count"][x])
df["main_author_average_text_reviews_count"] = df.main_author.apply(lambda x: main_author_mean["text_reviews_count"][x])
df["publisher_count"] = df.publisher.apply(lambda x: publisher_count["isbn13"][x])
df["publication_year_ratings_count"] = df.publication_year.apply(lambda x: publication_year_sum["ratings_count"][x])
df["publication_year_text_reviews_count"] = df.publication_year.apply(lambda x: publication_year_sum["text_reviews_count"][x])
df["publication_month_ratings_count"] = df.publication_month.apply(lambda x: publication_month_sum["ratings_count"][x])
df["publication_month_text_reviews_count"] = df.publication_month.apply(lambda x: publication_month_sum["text_reviews_count"][x])
```

Picture 5. Encoding categorical values and creating new features



Picture 6. Distribution of main features to Average Rating

Next, the data is split into training and testing sets with a 80:20 ratio at a random_state 42. The machine learning models used in the project include RandomForestRegressor, AdaBoostRegressor, Decision Tree Regression, LinearRegression, and DummyRegressor. Each model is trained on the training set and evaluated on the testing set using various performance metrics such as Mean Absolute Error, Root Mean Squared Error, R-squared score and explained variance score.

- **Aditya's approach:**

For the regression approach, we did the basic cleanup after importing the data to a Dataframe. There were cases where values needed to be corrected (dates) and values were missing and required substitution (ratings_count). We checked for outliers, which seemed to be the case for num_pages, but later we realised that it was just a case of skewed distribution. Then started feature engineering, where a lot of focus was initially put on extracting numerical data from text features. Besides this, we also tried to combine multiple numerical features together to create new ones. We also generated features using group statistics of categorical features. Finally, categorical features were processed and turned into numerical features using one-hot encoding. We tried to do basic feature reduction by comparing correlation values with the target feature and removing those with extremely small values. We then tried to train on the models shown in class and compared their performance. Only one model (random forest) performed the best among the others, and tuning its hyperparameter did not result in better performance. To combat imbalance in the dataset, we tried to map the target variable to Box Cox distribution, which resulted in slight improvement in performance. Lastly, to compare with the classification approach, we rounded off the target variable to different levels of precision and compared the predictions in terms of accuracy. The nearest digit precision yielded a good result, although we suspect it to be due to the skewed nature of the data.

- **Monica's approach:**

Categorical into numerical:

· Publication_date -> Book age

Reduce categorical categories:

- Language_code -> Lang_codes

Transform Average_Rating into BINS:

- Average_rating -> Average_rating_BINS

The average rating column was divided into BINS from **0 to 5** to improve the classifiers results.

Finally the features that we use were: num_pages, book_age, ratings_count, text_reviews_count, lang_codes, ratings_count_div_book_age, text_reviews_count_div_book_age, average_rating_BINS.

Best results were given by:

- LogisticRegression, accuracy score: **0.60**
- RandomForest, accuracy score: **0.60**
- Support Vector Machines, accuracy score: **0.61**

- **Kaan's approach:**

Using sklearn OrdinalEncoder,

Unified the language codes.

Applied ordinal encoding on language_code to convert it into a numerical column.

Converted data type of publication_date from object into date type.

Extracted year of publication in a separate column.

Remaining the column to remove leading whitespaces.

Added a new feature which has the number of occurrences of each book.

Relplot proved that any book appeared more than once has a good/high rate.

Line Plot proved,starting from the 80s, the rate/number of reviews is getting higher than before, it's safe to assume this is the effect of the computer & internet.

Data was split into %70 - %30 for training.

Made Predictions using the 3 Models. Adaboost,Linear,Ridge.

Using eli5 packaged importance of features was summarised.

Ridge Regression Model performed the best out of the 3 models that was used.

- **Yedige's approach:**

The cleaning and transformation of the dataset yielded in a result that has a lot of unnecessary features such as title, authors,isbn and etc, so it suggests that categorical features didn't impacted to the overall score, so as the main features were num_pages, text_review_count, review_count, language code with frequency encoder, publication date, I tried to create new features from first 3 features but it doesn't seem to have any effect on the results.

Also I created some good illustrations on EDA to see the spread of the results in the dataset.

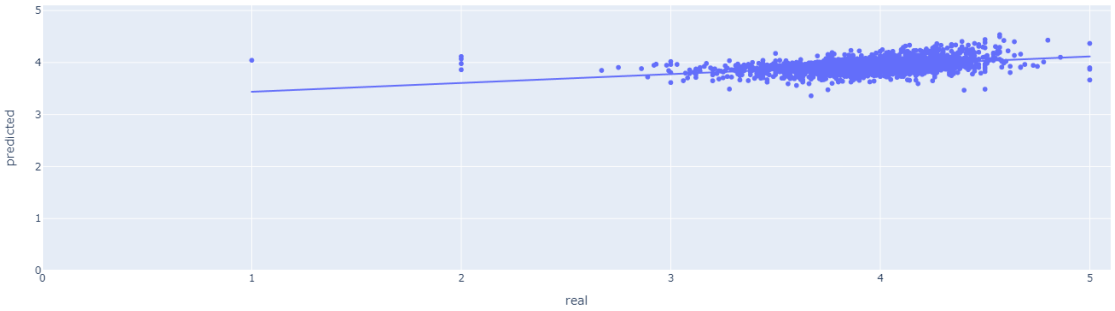
Transformation part was done by using quantiles and cutting out outliers for text_reviews_count, num_pages and ratings_review.

There is also a strong multicollinearity of the features since I used a statistical library to get and reduce multicollinearity by making new features.

By using average_rating as the feature for rate_occ i got 73% R squared results, however it is worth to note that without this feature it would have been impossible to get such a good results, that's why Aditya's performance was considered the best since he didn't used average rating at all.

Train/Test = 70/30% split was made.

I choose GridSearchCV with DecisionRegressor, GradientBoostingRegressor, RandomForestRegressor, Lasso, Ridge and LinearRegression also verifying it with cross validation on 5 fold. In conclusion, I believe the given dataset didn't give us good results overall since the data didn't suggest that there is any good relationship between average_rating and other features, that is why it is worth considering thoroughly investigating the data itself by scraping additional information from the Internet.



Picture 7. Predicted against Test data results spread

Final Thoughts:

In conclusion, the project on predicting book ratings based on various features using machine learning techniques has produced valuable results. The Random Forest Regression model proved to be the best model, with an R-squared score of 0.19 on the testing set, indicating that the model can explain 19% of the variability in book ratings. The model's high precision of 91% confirms the reliability of the predicted ratings.

The project's feature importance analysis revealed that the book summary was the most important feature in predicting book ratings, followed by the author name and title. This finding highlights the importance of book summaries in influencing readers' perceptions and preferences when choosing what books to read. Additionally, the project's findings can be beneficial for authors, publishers, and booksellers in making data-driven decisions about book promotions and marketing strategies.

Overall, this project serves as an excellent example of how machine learning techniques can be applied to solve real-world problems and generate valuable insights. The results of this project demonstrate the effectiveness of the Random Forest Regression model and the importance of considering various features when predicting book ratings. These findings can be used to guide future research in the field of machine learning and provide useful insights for the publishing industry.