Final Project Report for Stats 170B, Spring 2020

Project Title: Predicting the Difference between Goodreads Ratings and Amazon Ratings

Student Names
Tu Lam, 43139271, lamtk1@uci.edu
Tien Nguyen, 68035743 , tientn6@uci.edu

Github: https://github.com/4tiennguyen/Stats_170AB

1. Introduction and Problem Statement

Goodreads and Amazon are both well-established sources for booklovers to look for reviews and recommendations to determine if they should start a book. Although both Goodreads and Amazon books are rated on a scale of 1 to 5 stars, how the stars' meaning are presented and interpreted greatly differ between the two sites. Star rating definitions on Goodreads in ascending order are "Did not like it," "It was ok," "Liked it," "Really like it," and "It was amazing." Star rating definitions on Amazon in ascending order are "I hate it," "I don't like it," "It's okay," "I like it," and "I love it."

It appears that Amazon's ratings have lower standards (e.g., an okay book would get a rating of 3 on Amazon but only a 2 on Goodreads). One possible explanation for the lower rating bar is that Amazon's primary goal is to encourage sales, not to give unbiased review statistics. We believe that Amazon defines its ratings could potentially lead to higher books' ratings than Goodreads. Additionally, the user base of these websites could have different rating criteria, which could also contribute to the differences in books' ratings between Goodreads and Amazon. We developed an algorithm to predict a book's rating difference between Goodreads and Amazon based on features and text reviews from both platforms if the book is present on both, as well as predict the rating of a book on one platform based on features and text reviews from the other site's platform if the book is only available on the latter's site.

2. Related Work

For multiple regression, we read "Four assumptions of multiple regression that researchers should always test" by Jason W. Osborne and Elaine Waters. In order to extract features from text reviews, we performed Latent Dirichlet Allocation (LDA). We referred to the article "Topic Modeling with Gensim (Python)" by Selva Prabhakaran for this process. We also looked at University of California, Berkeley's paper, "Random Forests" to build our prediction model. There are no works that we know of that use the above methods for this prediction problem. We also referred to the article "Implement Baseline Machine Learning Algorithms" by Jason Brownlee to establish baseline error on a predictive modeling problem.

3. Data Sets

3.1. Dataset description

There are four datasets in our project: Goodreads reviews, Goodreads book metadata, Amazon reviews, and Amazon book metadata. Both review datasets come with a set of user ratings and user comments for a set of books. The Goodreads reviews and Goodreads book metadata are crawled directly from Goodreads' API. The Amazon reviews and Amazon books metadata are provided by Dr. Julian McAuley's group at UC San Diego (https://nijianmo.github.io/amazon/index.html).

The Goodreads reviews dataset has 906,876 reviews. For each review, the data consist of user rating, user review time, book's ASIN/ISBN, word count before cleaning, word count after cleaning, raw review text (i.e. comment), and processed review text. Note that the user ratings, user review times, book's ASIN/ISBN, and user comments are the data in the website, while the other metadata is processed by us.

	overall	reviewTime	asin	count_before	count_after	reviewText	cleaned_text
0	4	Dec 14 2016	0307408868	462	206	Another hard to put down nonfiction book from ...	another hard put nonfiction book eric arson en...
1	5	Dec 21 2016	0062273205	1328	619	I haven't read many (any?) books that are writ...	read many book write leos leo ceo aspire ceo m...
2	0	Mar 20 2014	006073731X	4	3	Sacca and Nate recommend	sicca name recommend

Fig 1: An excerpt from Goodreads reviews dataset.

The Goodreads book metadata dataset has 37,233 rows and 21 columns, where each row consists of the book's ASIN/ISBN, number of ratings, number of pages, and many other data as listed by Goodreads. This dataset has both categorical (ASIN, publisher, format, description, genres) and numerical variables (the rest of the columns).

asin	average	ratings	reviews	text_reviews	total_ratings	total_reviews	total_text	publication_ye	publication_mc	publication_da	publisher	num_pages	formi	descr	cleaned_des	gr_countD	gr_countDes	cleaned_genres	gr_countText_	gr_countText
00010003	4.23	2E+05	163625	5535	220088	196528	8847	2010	1	1	Rupa & Co	127	Paper	Kahli	tahsil vibrant	106	66	poetry, fiction, n	42320	17834
1053655	4.08	16	33	6	676	1552	85	1997			HarperColl	268	Hardcover					history, historica	158	75
1061240	4.62	10	22	2	221	603	36	1959	12	1	Western P	324	Hardcover					poetry, children	49	18

Fig 2: An excerpt from Goodreads book metadata dataset.

The Amazon reviews dataset has 5,683,680 reviews. Like the Goodreads reviews dataset, Amazon data being saved includes the rating, review time, ASIN/ISBN, word count before cleaning, word count after cleaning, raw review text, and processed review text.

	overall	reviewTime	asin	count_before	count_after	reviewText	cleaned_text
0	5.0	08 12 2005	1713353	23	7	This book is a winner with both of my boys. T...	book winner boys enjoy picture story classic
1	5.0	03 30 2005	1713353	170	81	The King, the Mice and the Cheese by Nancy Gur...	king mouse cheese nancy gurney excellent child...
2	5.0	04 4 2004	1713353	55	27	My daughter got her first copy from her great-...	daughter get first copy greatgrandmother fathe...

Fig 3: An excerpt from the Amazon reviews dataset.

The dimension of the Amazon metadata dataset is 37233 rows x 10 columns. Each row contains a book found on both Goodreads and Amazon. Numeric features include ratings, ratings count, reviews count, rank, reviews word counts before cleaning, and reviews word counts after cleaning. Categorical features include genres and book format.

	asin	average	rating_count	text_reviews_count	genres	rank	verifiedTrue_count	Format	am_countText_before	am_countText_after
0	1713353	4.83	54	54	Childrens Books, Literature & Fiction	1461315	36	Paperback, Hardcover	2362	1037
1	1061240	4.87	45	45	Childrens Books, Literature & Fiction	321557	30	Hardcover	3085	1326
2	1711296	4.44	107	107	Literature & Fiction	2884610	69	Library Binding, VHS Tape, Paperback, Hard...	5667	2574

Fig 4: An excerpt from Amazon book metadata dataset.

3.1. Exploratory data analysis

To find the difference in ratings, we subtracted Goodreads's average rating from Amazon's average rating for each book. The average rating difference is around 0.4, meaning on average Amazon's ratings is approximately half a star more than that of Goodreads.

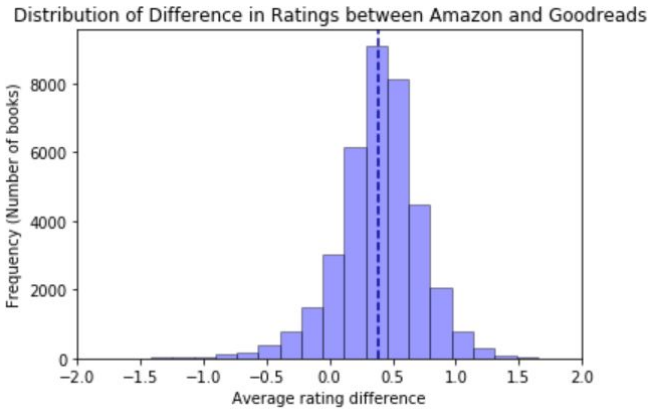


Fig 5: The histogram of Amazon and Goodreads rating difference. The graph closely resembles Normal distribution.

4. Overall Technical Approach

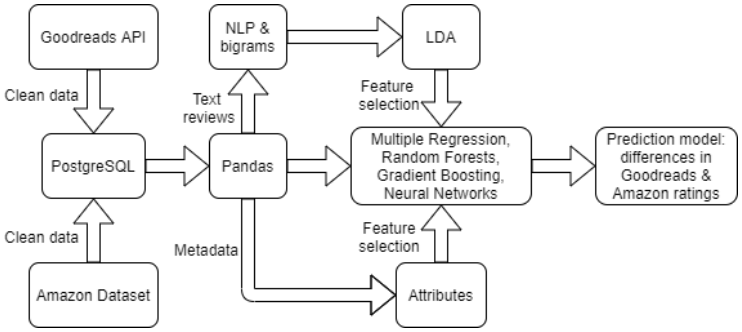


Fig 6: Pipeline of the project

4.1 Preprocessing Steps

We first collected data from Amazon and Goodreads and separated them into 4 datasets as mentioned in part 3 above. We decided to perform feature selection on metadata as raw attributes and on text reviews using Natural Language Processing (NLP) and Latent Dirichlet Allocation (LDA) in order to generate attributes as features for our prediction model.

a. Natural Language Processing (NLP)

Before using text reviews, we performed text cleaning on the users' comments. This process involves removing html tags, removing emojis, removing extra white space, converting texts to lowercase, removing numbers, expanding contraction, removing all punctuation, correcting spelling, removing all stopwords, and lemmatizing. This process removes noise from user comments since they do not contribute to the meaning of a comment. After performing text cleaning, we counted the number of words before and after cleaning text reviews for each book to use them as features in addition to metadata attributes.

b. Latent Dirichlet Allocation (LDA)

LDA is a generative statistical model that is widely used in big data applications. For NLP applications in particular, LDA finds k-clusters and gives the score of each membership of each topic to each set of texts. Intuitively, each topic will have a specific distribution of words learned by the LDA model, and the membership of a collection of words to a topic is determined based on the similarity of the text's words distribution to the topic's distribution

For NLP applications, the input to the LDA model is a collection of text. LDA also has a parameter of the number of topics, k. The output of the model has two parts. One part contains the

definitions of the clustered topics: it is a set of k vectors (one-to-one maps to k topics) of words and their weights. The dot product of one of those vectors with the distribution of words of a fragment of text gives the probability of that text being associated with that topic. The other part of the output contains a set of vectors, where each vector contains the probabilities of each fragment of text to each of k topics.

In our project, we used user's text reviews of each book as the input to the LDA model and the probability part of the output as part of the features to the input of our training and testing datasets. In the context of our work, the LDA score determines how relevant each topic is to each book based on its distribution of the words of the concatenated reviews. In order to do this, we used Gensim's Mallet wrapper to build and improve the quality of topics.

There are many methods to calculate the coherence score. The most reliable method is to use supervised labels that are labelled by humans; however, those labels are rare and hard and expensive to obtain. Luckily , there are many other methods to calculate coherence score, including UCI coherence score, UMass coherence score, and CV coherence score. CV coherence score is shown to be the most reliable, meaning it achieves the highest correlation to between its ranking and human ranked topics. Therefore, we chose to use the CV coherence score in our evaluation.

LDA requires the number of topics as an input. In order to find the optimal number of topics, we performed a parameter search on a range of numbers, and evaluated each of LDA models using the coherence score. A topic is said to be coherent when its top words having high scores share a high degree of semantic similarity. Technically, the LDA model will put a weight for each of N words for all topics, and the coherence score measures the sum of the similarity measurement between the semantic interpretability and those weights. The higher the CV coherence score is, the better the quality of topics. We ran the LDA model with various numbers of topics and chose the smallest number of topics that achieves the best coherence score (i.e. the elbow method).

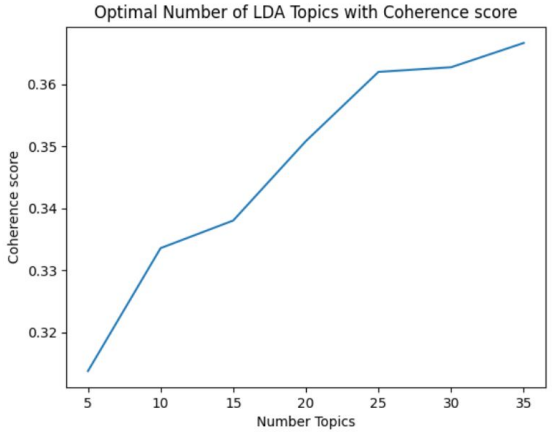


Fig 7: The optimal number of topics is 35, which has the highest score of 0.3667

Like the other tables, the LDA table has the ASIN feature along with 35 proportion topics features (eg: prop_topic_1, prop_topic_2, etc.). We merged this table with the metadata table through ASIN.

4.2 Prediction models (without LDA features)

a. Multiple Linear Regression

Firstly, we performed Multiple Linear Regression without using LDA features. We calculated p values of each feature--which is one of the best ways to determine if a variable is redundant--but they provided no information about how useful a variable is for each feature. After that, we checked the model's assumptions and found out that our data violated the Linearity and Normality of the Error Terms assumptions, so we scrapped this model. See appendix.

b. Random Forests (abbr. RF) and Neural Network (abbr. NN)

Next, we performed Random Forests and NN separately with just the metadata from Goodreads and Amazon and without the LDA features. We evaluated our models with varying metadata attributes using MAE, MSE, RMSE, and adjusted R2 (more information on these evaluation metrics will be discussed in section 6). From the best model, we used feature importances to select the top most important features in our model in order to reduce model complexity, training time, and noise.

i. We found that almost all the numeric features are useful in the model. Some variables such as reviews_count and text_reviews_count from the Goodreads metadata dataset are collinear, which made one of them contribute less to the prediction model. As a result, we decided to combine these variables. Additionally, some of the numerical variables in the Goodreads metadata dataset contain null values, which we dealt with by trying various methods. We tested omitting these values, filling in these values with the mean and median, and imputing them. We discovered that omitting these null values during training gave the best performance since the percentage of null values is small compared to the size of the dataset. Lastly, we decided to remove books with extreme rating differences since it increased our error results.

ii. As for the categorical variables, we one-hot encoded them and incorporated them after selecting the numeric features. Comparing the train and test performance, we found that only Goodreads's genres improved the errors for our models. After a closer look at the variables, we found that the other categorical variables contained too many levels, which created many unnecessary features and noise in the dataset and hindered the performance of the models. We once again looked at the feature importances of the model with Goodreads's genres and realized that that only the "Children" genre seemed to be an important predictor of the model, so we only selected "Children" in the model and found that the performance remained about the same.

c. Extreme Gradient Boosting (abbr. XGB)

The only thing different about this model than the two models in part b was we did not have to deal with the null values as XGB automatically deals with them.

4.3 Prediction models (with LDA features)

After having LDA features, we combined them with basic features and performed the three algorithms listed in section 4.2. We tested both weighted and unweighted averages of RF, NN, and XGB with cross-validation train/test split to try and improve the model performance beyond just the three individual models. We evaluated the ensembles based on MAE, MSE, RMSE, and R2 adjusted. We trained the ensemble based on:

i. Amazon features only to predict rating differences
ii. Goodreads features only to predict rating differences
iii. Both Goodreads and Amazon features to predict rating differences

5. Software

We used PostgreSQL to store our data for the project. PostgreSQL is a SQL relational database management system, which means we have to define schemas and clean data thoroughly before putting them into tables. One nice feature we discovered while working on the project is that we can use SQLAlchemy to export our cleaned dataframes into PostgreSQL without having to define the tables beforehand. An issue we had with PostgreSQL is that it cannot store the Amazon reviews dataset because

it is too large for it, so we had to keep the Amazon dataset in a tsv file and work directly with it from Jupyter Notebook by reading the file.

Software / Library / Package	**Functionality**
PostgreSQL, SQLAlchemy	Data storage
Requests, json, xml, Selenium	Web crawling from Goodreads API
Matplotlib, Seaborn	Data visualization
Natural Language Toolkit (NLTK)	Text cleaning
Scikit-learn (sklearn), XGBoost (XGB)	Prediction modeling
Gensim/Mallet	LDA model

6. Experiments and Evaluation

We used 4 metrics to evaluate our models in combination with train / test split cross-validation for each of our models.

- **Mean Absolute Error** (MAE) is the mean of the absolute value of the errors:

$$\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

- **Mean Squared Error** (MSE) is the mean of the squared errors:

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

- **Root Mean Squared Error** (RMSE) is the square root of the mean of the squared errors:

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

- **Adjusted R-Squared** is a modified version of R-squared that has been adjusted for the number of predictors in the model.

$$1-\left[\frac{(1-R^2)(n-1)}{n-k-1}\right]$$

  Where n is the number of observations in the dataset and k is the number of independent regressors, i.e. the number of variables in the model, excluding the constant.

For our baseline, we decided to calculate the errors by predicting the mean for all books' rating differences (it is known as Zero Rule Algorithm. It works by calculating the mean value for the observed output values.) We used this baseline to compare to our models' errors in order to know if our prediction model is working correctly as intended. The baseline errors are as follows:

	BASELINE
MAE	0.2473
MSE	0.1124
RMSE	0.3353
R2 ADJ	-0.0013

6.1 Random Forests

For Random Forests, we started off using just the metadata features. We used GridSearchCV and RandomForestRegressor from the sklearn library in order to perform an exhaustive search over the model's parameters. GridSearchCV tests every parameter value / range in the model specified by the users and finds the best parameter values for the model optimized by cross-validation. The best model found by GridSearchCV had parameters: n_estimators=1155, min_samples_split=35, min_samples_leaf=3, max_leaf_nodes=7000, max_features='auto', max_depth=30, bootstrap=True. This model gave us the following error rates:

	TRAIN ERROR	TEST ERROR
MAE	0.1760	0.2260
MSE	0.0598	0.0961
RMSE	0.2446	0.3101
R2 ADJ	0.4668	0.1467

The train and test error appear are both lower than the baseline, but not by much so we needed to incorporate more features. The difference between the train and test error--the difference in RMSE in this case is 0.0655, about 16% of the average rating difference--is higher than what we would like, which indicated there might be some overfitting in the model.

6.2 Extreme Gradient Boosting

We tuned hyperparameters in the XGB model using just metadata attributes using the built-in method cv (cross-validation). The method takes a DMatrix (XGB's built-in dataframe-like data structure called data matrix) and a dictionary of parameters; it performs a specified amount of cross-validation rounds of extreme gradient boosting and returns the train and test RMSE and average RMSE over all rounds. Our model had the following parameters: eta=0.1, min_child_weight=10, tree_method='approx'. The errors achieved by the model were:

	TRAIN ERROR	TEST ERROR
MAE	0.2067	0.2254
MSE	0.0793	0.0959
RMSE	0.2816	0.3097
R2 ADJ	0.2932	0.1484

Although the test error oF XGB is worse than that of RF, we thought XGB performed much better since the train errors were much closer to the test errors, which meant the model was not overfitting. Like the RF results, we need to incorporate more features to improve the baseline error rate.

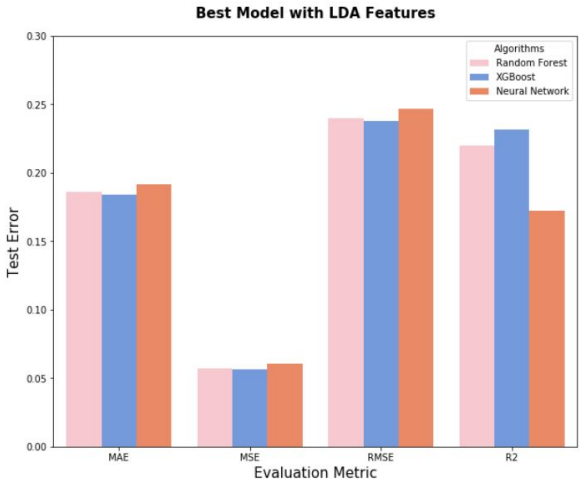
6.3 Neural Networks

In the Neural Networks metadata-only model, we manually tuned the hyperparameters since we couldn't get it to work with GridSearchCV. We found the best hyperparameters were: hidden_layer_sizes=14, activation='relu', solver='lbfgs', alpha=0.002, learning_rate='constant', max_iter=500. The errors were:

	TRAIN ERROR	TEST ERROR
MAE	0.2277	0.2316
MSE	0.0962	0.0999
RMSE	0.3101	0.3163
R2 ADJ	0.1429	0.1135

Although the test error of NN is slightly worse than that of RF, we thought RF performed much better since the train errors were much closer to the test errors, which meant the model was not overfitting. Like the models above, we decided to add more features from LDA in order to get better results.

6.4 Models with LDA features



Next, we incorporated the 35 LDA features into the model using both Goodreads and Amazon metadata. The features significantly improved our model performance.

The graph on the left illustrates the different error metrics evaluated on 3 different algorithms for this model: RF, XGB, and NN. Among the metrics, XGB consistently has a lower test error rate than the other two as well as has the highest adjusted R2 value. On the other hand, NN always has the worst test errors and R2 values regardless of metrics. Therefore, XGB is the best algorithm for the model that uses Goodreads and Amazon features to predict rating differences.

Fig 8: Test errors of multiple algorithms trained using LDA features

Despite being the best algorithm, XGB test error is still high. Note that the MAE test error, for example, of XHB is about 0.1837, while the mean difference is 0.4 (based on EDA), this means that, on average, the error of the prediction could be as high as 45% for each prediction. Hence, it is not reliable enough for non-experimental settings.

After knowing that XGB performed best, then followed by RF and NN, we tested out some weighted average ensembles and we came up with the best model with the formula: .30*RF + .60*XGB + .10*NN which gives us the best error performance:

	TRAIN ERROR	TEST ERROR
MAE	0.1542	0.1834
MSE	0.0399	0.0562
RMSE	0.1998	0.2370
R2 ADJ	0.4779	0.2353

Taking a step further, we also implemented 2 additional models to predict rating differences. The first one only uses Amazon features to predict rating differences in the case that the book only has reviews on Amazon. The second one only uses Goodreads features to predict rating differences for similar reasoning. For the model that only uses Amazon features, we used all features except for format since it has too many levels after encoding, in addition to 15 LDA topics from Amazon reviews. Our best performing model is the ensemble of .25*RF + .38*XGB + .37*NN. We discovered that the model performed much better than the ensemble model that uses both Amazon and Goodreads features. Our explanation for this is that unlike Amazon's metadata, Goodreads's metadata contained much more noise

with many more variables that correlate less to the rating (for example, Amazon's rank feature highly contributes to a book's rating). The errors for the model are:

	TRAIN ERROR	TEST ERROR
MAE	0.1424	0.1589
MSE	0.0350	0.0445
RMSE	0.1871	0.2109
R2 ADJ	0.6893	0.5996

As for the model that uses only Goodreads features to predict rating differences, we used most of the numerical variables as well as a dummy variable representing whether a book is a children's book, 15 LDA topics from Goodreads reviews, and an additional LDA topics from Goodreads books' descriptions. The best model was an ensemble of .60*RF + .20*XGB + .20*NN. The model performed much poorer than the previous two models, but slightly better than the individual models that took attributes from both Goodreads and Amazon. We conjectured that this occurred because the Goodreads metadata features are simply not as useful as the Amazon metadata features. The results of the model are:

	TRAIN ERROR	TEST ERROR
MAE	0.1799	0.2204
MSE	0.0630	0.0900
RMSE	0.2510	0.3000
R2 ADJ	0.4426	0.1597

Overall, all of our models outperformed the baseline, which is a good sign that we created a good prediction model. However, our two best models' RMSE came out to be 0.2370 and 0.2109, which is quite high considering that the average rating difference between Goodreads and Amazon is about 0.4 (it is half of the average rating difference!). We also took into consideration that the possible range of rating differences is [-4,4], so our RMSE is only about 3% of the possible rating differences. Additionally, the actual range of our data is [-2.78,1.99], so our RMSE is about 5% of the actual rating differences. 5% coincidentally is the cut off point we decided was a "good enough" RMSE for our models, and our models made the cut. However, if given more time and resources, we would like to improve the models and bring the RMSE down to about 1% of the range of the rating differences.

7. Notebook Description

On our github repository, we have two folders: complete_project and demo_project (a small sample of the datasets and codes from complete_project that only shows important aspects of the project in order to reduce grading run time.) Complete_project folder contains all the files (ipynb and py files) to complete the project. The numbers presented in this report are the results of running the scripts and the notebooks in this folder. In the demo_project folder, we merged the scripts and the notebooks so that it could run under a minute. Specifically, the file cleaning_text.ipynb demonstrates how we cleaned the text reviews. The eda.ipynb file contains the code for performing exploratory data analysis on metadata. For the LDA model, we built the optimal model and extracted features by running LDA_wmallet_tune.ipynb. models_with_lda.ipynb contains the code running RF, XGB, NN and testing out some weighted average ensembles. am_predict_gr.ipynb contains codes for Amazon features only to predict rating differences.

8. Members Participation

For data collection, both Tien collected the Amazon data while Tu collected the Goodreads data. Tien was responsible for the text cleaning and processing algorithm while Tu cleaned up the datasets by selecting metadata features to keep merging and separating various data tables into four datasets. Both

members fairly contributed to exploratory data analysis and features selection. Tu did the modeling and Tien helped to tune the models and ensembles.

Task	**Subtask**	**Tu Lam**	**Tien Nguyen**
Data collection	N/A	50%	50%
Data cleaning	N/A	30%	70%
Exploratory data analysis	N/A	40%	60%
Feature selection / extraction	N/A	30%	70%
Prediction model (RF + XGB + NN + ensemble)	Multiple linear regression	0%	100%
	GR & AM predict rating difference	70%	30%
	AM predict rating difference	90%	10%
	GR predict rating difference	90%	10%

9. Discussion and Conclusion

One of the strengths of Multiple Linear Regression is that its speed is fast, and it runs fast when the amount of data is large. However, the downside is that it has many assumptions. It is only efficient for linear data. Our data shows the nonlinearity, therefore linear regression can not be useful for such kind of data. We also chose Random Forest, XGBoost (which is a gradient boosting tree algorithm), and Neural Network because they can handle many predictor variables, including categorical variables after encoding. These algorithms have no formal assumptions such as the ones for multiple regression.

Something that ended up being harder than we expected in our project was managing the dataset. We modified the code of preprocessing data and the training model many times, hence we had multiple copies of data. We find it challenging to keep track of the data in terms of which methods were used to process the file. Other than that, since the dataset is huge compared to our storage resource, especially when we decompressed the raw data, we often run out of memory because could not fit all data to RAM at once, and we ran out of hard drive storage because we have multiple copies of large processed data.

We learned that using version control software like github is useful for managing the code base. It solved our problem of keeping track of the processed file as discussed above. We also learned to implement the LDA model and to fine tune its parameter. The most surprising part of our project is that even though it has millions of reviews and a magnitude of ten thousands of books, the optimal number of topics to cluster them is surprisingly low (35), and this low number of topics is a relief for our limited computing resources as the lower number of topics, the smaller input size to our prediction models.

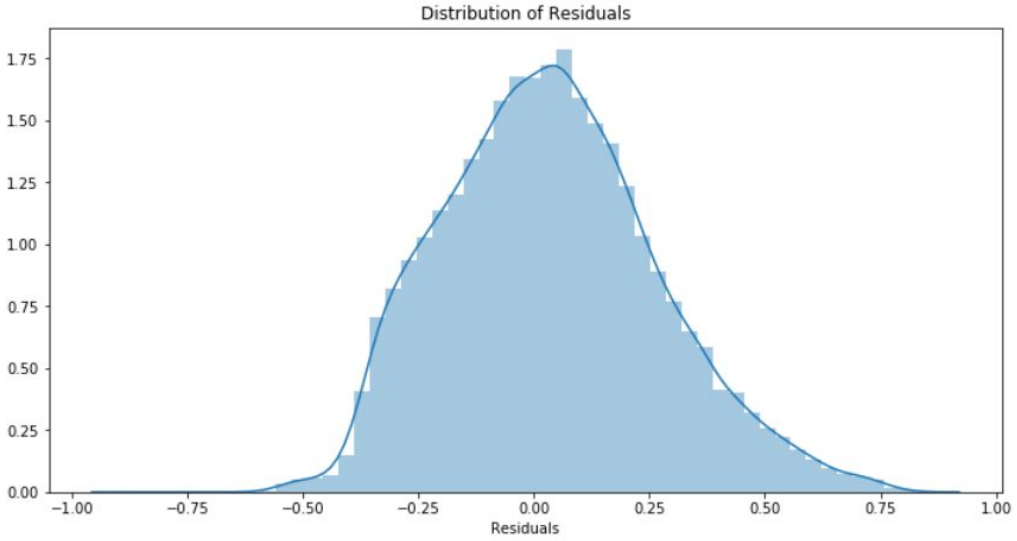
The majority of challenges we faced in this project was conducting experiments with limited computing resources and working with a small amount of data (compared to the input to state-of-the-art deep learning models). If we were in charge of more computing resources and a large amount of dataset from Goodreads, Amazon and other book reviewing websites, we would like to investigate the effectiveness of pretrained BERT, MegatronLM and T-NLG along with a deep neural network for the task of predicting the rating differences between any two websites.

Appendix

Checking some assumptions for Multiple Regression

- Normality: Assumes that the error terms are normally distributed. If they are not, nonlinear transformations of variables may solve this.
  This assumption being violated primarily causes issues with the confidence intervals

```
Assumption 2: The error terms are normally distributed

Using the Anderson-Darling test for normal distribution
p-value from the test - below 0.05 generally means non-normal: 0.0
Residuals are not normally distributed
```

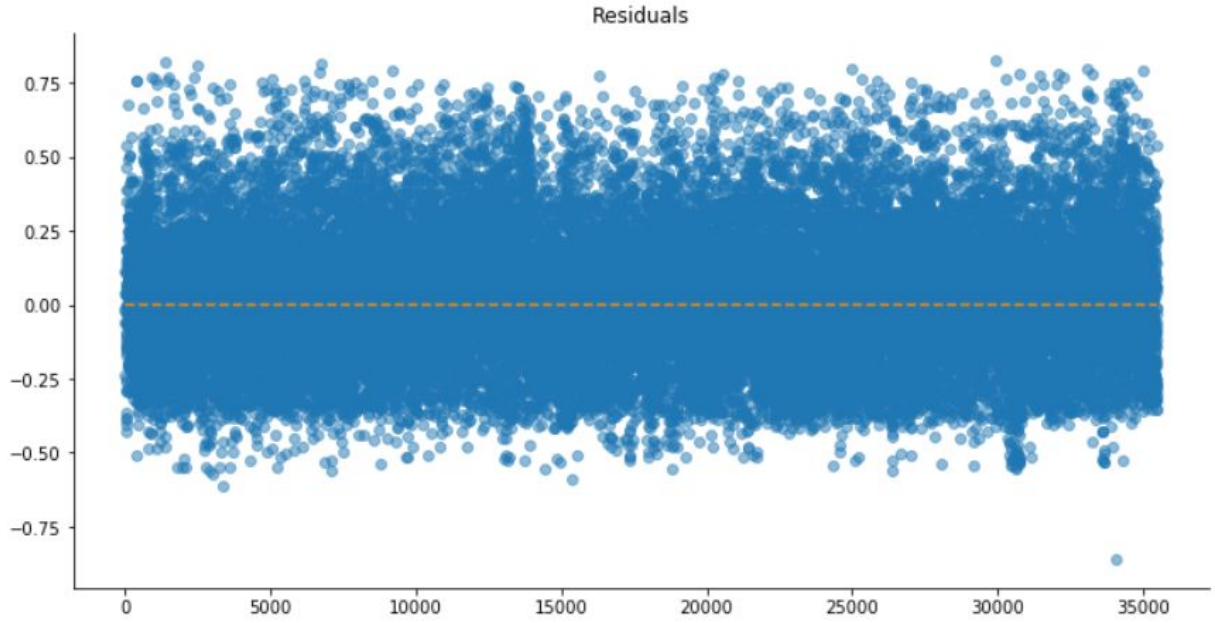



```
Assumption not satisfied

Confidence intervals will likely be affected
Try performing nonlinear transformations on variables
```

=> This isn't ideal, and we can see that our model is biasing towards under-estimating

- Homoscedasticity: Assumes that the errors exhibit constant variance

```
Assumption 5: Homoscedasticity of Error Terms

Residuals should have relative constant variance
```



=> We can see a fully uniform variance across our residuals. There don't appear to be any obvious problems with that.