SAN JOSE STATE UNIVERSITY

SAN JOSÉ STATE
UNIVERSITY

PROJECT ON

GOODREADS SPOILER DETECTION

Submitted in fulfillment of the requirements for the completion of

the course **CMPE-256** in Fall 2019

LARGE SCALE ANALYTICS

SUBMITTED BY TEAM,

DATAJAM

ZEESHAN AHMED PACHODIWALE (012659920)

NEHA PARAKH (012449203)

JASMINE AKKAL (013773825)

Under the guidance of:

Dr. Magdalini Erinaki

DEPARTMENT OF ENGINEERING

SAN JOSÉ STATE UNIVERSITY

1 WASHINGTON SQ, SAN JOSE, CA 95192

FALL 2019

Contents

Chapter 1 – Introduction

1.1 Motivation

Goodreads is a social platform for readers and bloggers where there is a huge catalogue for books, their reviews and a social platform to find out what your friends/fellow bloggers are reading. It is a great website to find out what a person wants to read next and what are the reviews for that book. Goodreads has around 2.2 billion books in its database and 75 million users making it a reliable website for book reviews and recommendations for next read.

The review platform is also used as a discussion platform by the users making it prone to "spoilers". These spoilers can curb the curiosity of the reader and may decrease the excitement to read a book which might contain plot twists etc. Moreover, it is crucial for the authors that the plot line is not revealed in the reviews. The concept perceived by one of the users might not be the one which the author wants to convey. Potential buyers who read this might be misled by the lack of understanding of other users. If the story line is revealed in the review and the customer happens to read it, the book might not seem as appealing to him/her. Thus, this results in lack of business for the authors.

One of the solutions to this problem is crowdsourcing where the users can report a certain review having spoiler. However, this solution is not scalable due to the large amount of data available and timely response. Machine learning techniques can be helpful in this domain. Automatic detection of spoilers from the reviews given by users that can result in timely reporting of reviews with spoilers. In this project, we are trying to develop machine learning techniques that can automatically classify the reviews as spoilers and non-spoilers to avoid impractical solutions like crowdsourcing and self-reporting.

1.2 Objective

In this project, we are trying to develop machine learning model to automatically detect spoilers in reviews and classify the reviews as "spoilers" and "non-spoilers". The Goodreads dataset is used for this purpose and data cleaning, visualization and feature engineering are applied on the same followed by implementation of several machine learning models to classify the reviews as "spoilers". This can help the users who want to get genuine feedback of other readers without the revelation of the story plot and helps those who want to discuss their feedback and views about the plot freely on the website without the hustle of self-reporting spoilers.

The main objective of this project is:

- Detect spoilers a review.
- Create a safe space for the readers to review a book and make purchases based on them.
- Promote healthy business by hiding the plot line.

From learning perspective, the objective of this project is:

- Get experience in the domain of data visualization, data cleaning and feature engineering.
- Application of machine learning models and the selection of the right models and the parameters.
- Knowledge about the dataset and the features that are critical in spoiler detection

1.3 Project Goals

1. **Classifying a review as spoiler or non-spoiler**:

We are applying several text mining and feature engineering techniques on the dataset followed by training the model to classify if the review has a spoiler or not. This prediction is based on the reviews provided by the user, the user spoiler ratio, the boo spoiler ratio and the rating provided by the user.

2. **Classifying each sentence of the review as spoiler or non-spoiler:**

The entire review consists of several sentences where each sentence can be further segregated as spoiler or non-spoiler. This can be beneficial to hide just the spoiler sentences and reveal the ones which does not have spoilers so that the users can obtain appropriate feedback from a review without the revelation of crucial plot or twists.

3. Using PySpark to handle Big Data Analysis:

The Goodreads dataset is a huge dataset and thus requires a lot of computational power which other libraries fail to support. Pyspark supports big data analysis by usage od RDD and using cache and disk persistence. PySpark is comparatively faster on larger datasets when compared to pandas. The implementation of models using this framework is one of the main learning curves for this project.

Chapter 2 – System design and Implementation details

2.1 Algorithm selected:

The algorithms selected were:

- Vanilla Logistic Regression
- LDA
- Naïve Bayes
- Random Forest

2.2 Tool and Technologies used:

Libraries used: Pyspark, numpy, sklearn, pandas, nltk

Computing Platform: Python 3, VS Code, Jupyter Notebook, HPC

2.3 System Architecture design:

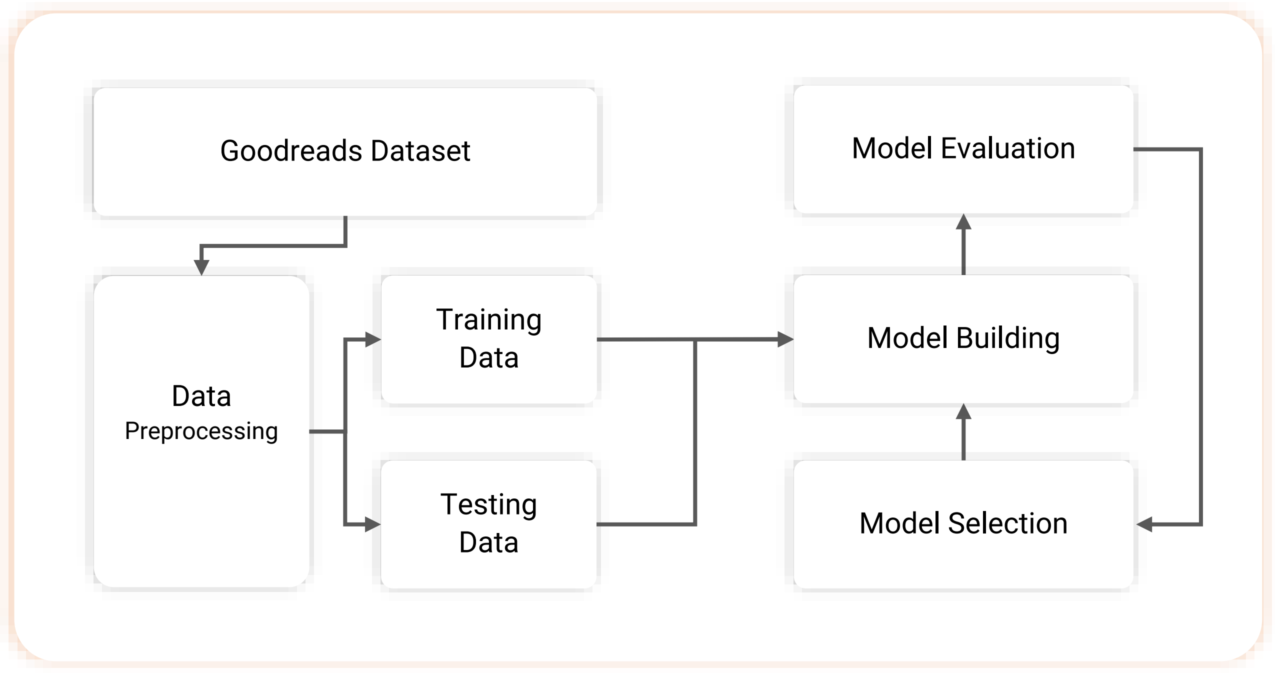


Fig.1 System Architecture Design

Chapter 3 – Experiments

3.1 Dataset:

The dataset used for the model is provided by the University of California, San Diego[1]. The whole dataset (goodread_reviews_dedup.json.gz) consists of the complete multilingual review text without spoilers. This dataset is relatively large and contains more 15M reviews for about 2M books and 465K users.

The scope of the project considers only reviews written in English. So, we selected a subset of this dataset that contains two files, one which provides spoiler tags for each sentence of the review and another which surrounds spoiler text with tags. This dataset contains more than 1.3M book reviews about 25K books and approx. 19K users.

File	Size	Format
goodreads_reviews_spoiler.json.gz	1.3M	reviews and spoiler tags are parsed into sentence-level
goodreads_reviews_spoiler_raw.json.gz	1.3M	the original review text is included where spoiler contents are labeled using '(view spoiler)[' and '(hide spoiler)]'

Brief description about the dataset:

- **user_id:** Each reviewer's unique id.
- **book_id:** Book id of Goodreads dataset
- **review_id**: Unique id each review
- **rating:** rating of the book provided by the user
- **review_sentences:** Reviews given by the user. Each spoiler sentence is tagged with '1' indicating that it has a spoiler or '0' indicating "non-spoiler" sentence
- **timestamp:** time at which the user reviewed the book
- **has_spoiler:** Boolean parameter with values 'True' or 'False' implying "spoiler" or "non spoiler" respectively

3.2 Data preprocessing:

The following data preprocessing techniques were implemented:

Feature selection: After data analysis, we found out that the columns like timestamp and rating did not have any impact in the prediction and hence were dropped.

Feature Engineering: New features were added to the dataset:

User-spoiler ratio: The user spoiler was determined based on the user-id and how likely is it for a user to give a spoiler. It can be calculated as follows:

$$\text{User spoiler ratio} = \frac{\text{Total number of spoilers given by the user}}{\text{Total reviews given by the user}}$$

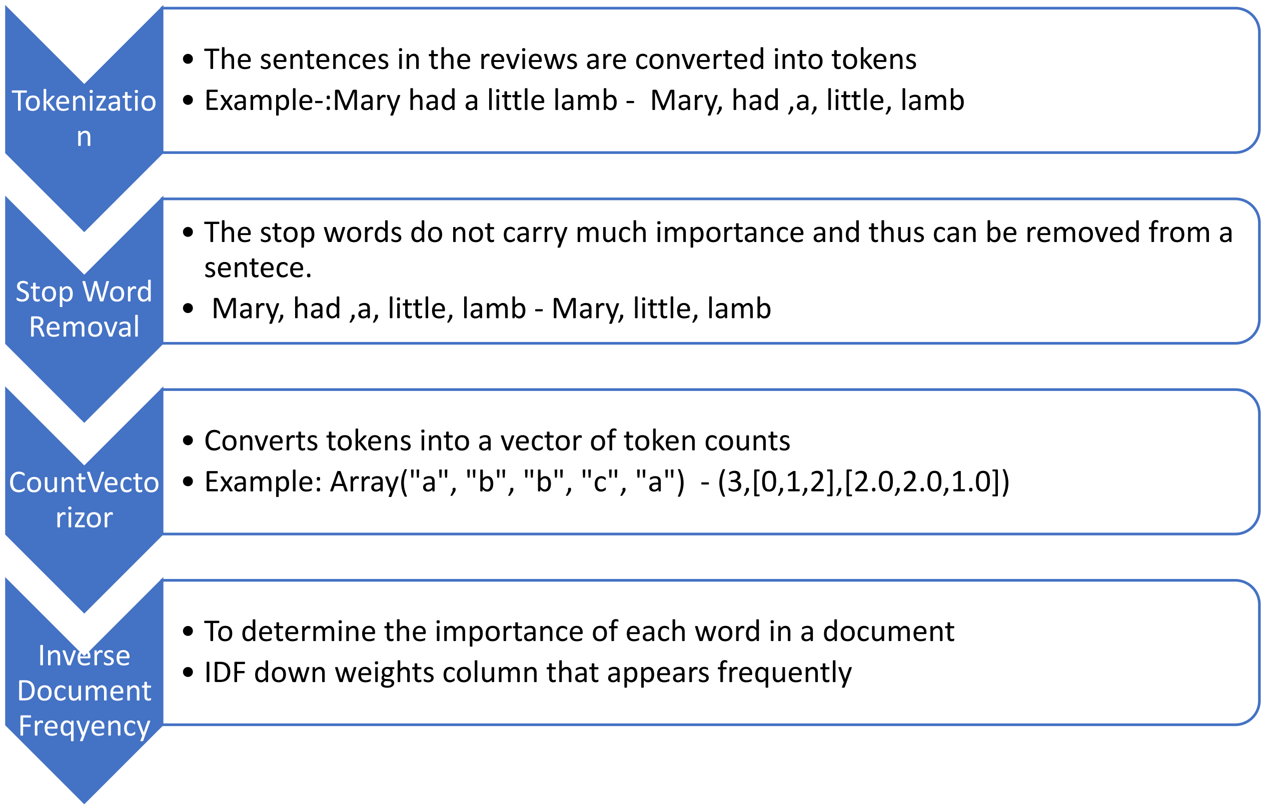
Book-spoiler ratio: The book spoiler ratio was determined based on the book-id and is determined to indicate how likely a book can get a spoiler. It is given as follows:

$$\text{Book spoiler ratio} = \frac{\text{Total number of spoilers given for the book}}{\text{Total reviews given for the book}}$$

Label Encoding: The column "has_spoiler" was converted into a Boolean column using LabelEncoder.

Text Mining:

The following techniques were implemented on the review column to convert it into data for analysis and feeding the machine learning model for SVM, Logistic Regression, LDA and Naïve Bayes:



Tokenizatio n

- The sentences in the reviews are converted into tokens
- Example-:Mary had a little lamb - Mary, had ,a, little, lamb

Stop Word Removal

- The stop words do not carry much importance and thus can be removed from a sentece.
- Mary, had ,a, little, lamb - Mary, little, lamb

CountVecto rizor

- Converts tokens into a vector of token counts
- Example: Array("a", "b", "b", "c", "a") - (3,[0,1,2],[2.0,2.0,1.0])

Inverse Document Freqyency

- To determine the importance of each word in a document
- IDF down weights column that appears frequently

Word2Vector: We implemented another approach which used text mining techniques of tokenization and word2vector followed by feeding this to a Random Forest Model. In this approach, each word is converted into a vector and similar words lie close to each other. This is directly fed to the Random Forest Model for classification.

3.3 Methodology followed:

1. Data Visualization: Several data visualization techniques were implemented to find the trends in data, detect outliers and patterns in the dataset.

Average Review Length: We found out that the average length of the spoiler review was longer than the non-spoiler reviews as shown in Figure 2.

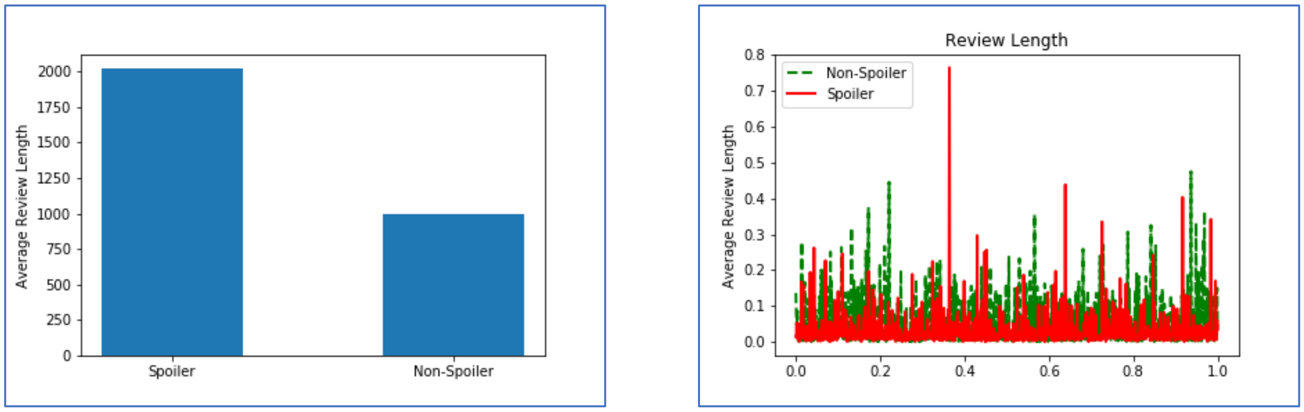


Fig 2: Average Review Length

Average Length of Sentences: The average length of non-spoiler sentences in a review was longer than spoiler sentences as shown in Figure 3.

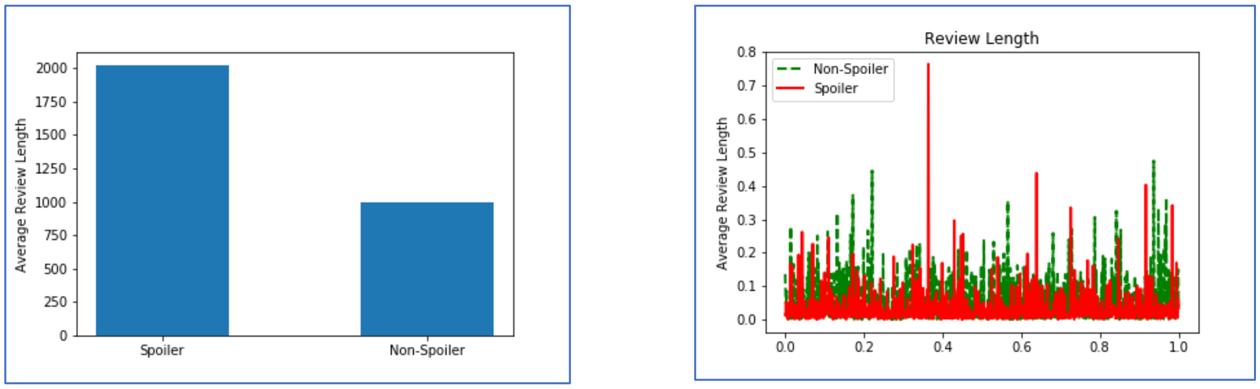


Fig 3: Average Length of Sequences

Ratio of spoiler and non-spoiler: We found out that this was an imbalances dataset as the number of spoiler reviews were way less than non-spoiler reviews as shown in Figure 4.

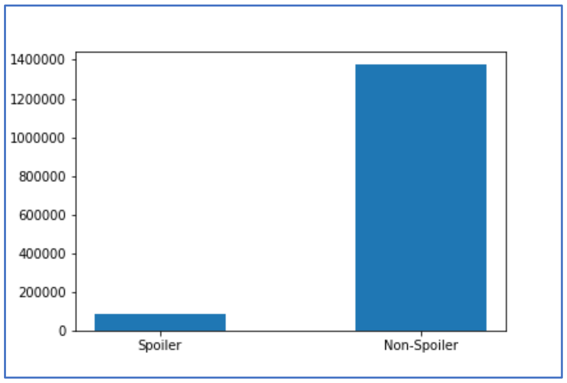
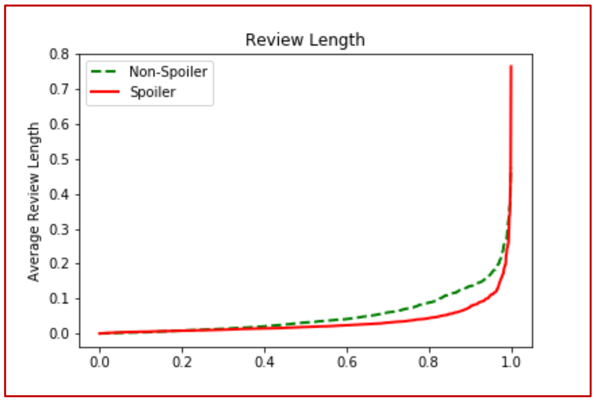


Fig 4: Number of spoiler and non-spoiler reviews

Review Length after stop word removal: We also visualized if the review length changes after and before preprocessing as shown in Figure 5.



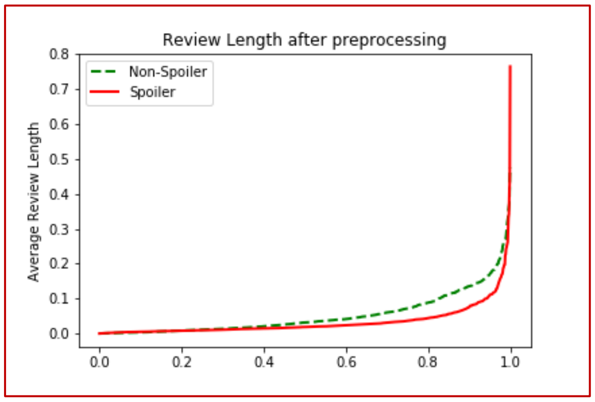


Fig 5. Review length before and after removing English stop words

2. Feature Engineering: New features like the user-spoiler ratio and book-spoiler ratio were created to enrich the dataset. Label Encoding was performed on “has_spoiler” column.

3. Feature selection: Columns like timestamp and rating were dropped as they did not contribute to the prediction of the model.

4. Preprocessing: Several text mining techniques like tokenization, count vectorizer, removal of stop words and IDF were implemented on the review text so that they can be converted into numbers and fed to the machine learning algorithm.

5. Train test split: The data was split into training and testing in the ratio 0.8 and 0.2.

6. Model implementation: The following machine learning models were implemented:

- Vanilla Logistic Regression
- Latent Dirichlet Allocation (LDA)
- Naïve Bayes
- Random Forest Classifier

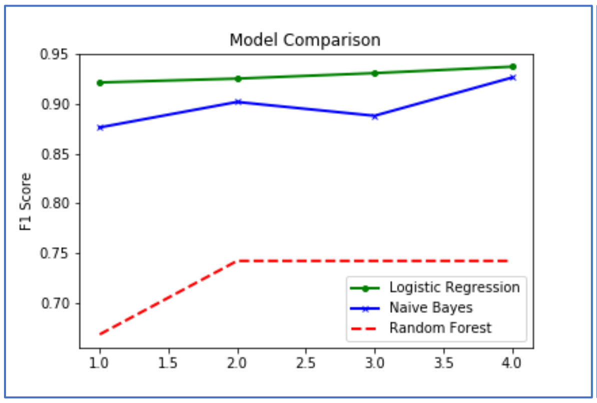
7. Evaluation: The classification model was evaluated using following evaluators on the metric F1 score:

- BinaryClassificationEvaluator
- MulticlassClassificationEvaluator

The model was also tested using several reviews generated by us and also several reviews from other web sources.

3.5 Comparisons:

We trained all the mentioned models using 80% of the dataset and tested them with the remaining 20% of the data. The F1 scores and comparisons between different models is as below,



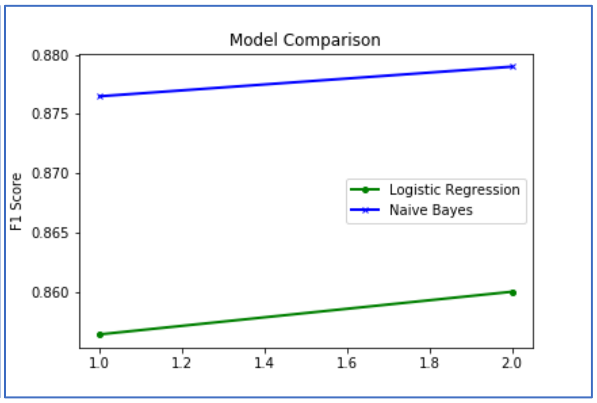


Fig 6: Model Comparison (bag of words data)   Fig 7: Model Comparison (word2vec data)

3.6 Result analysis:

We found out that Logistic Regression performed better than other models in predicting spoilers. Logistic regression works best when we remove attributes that are unrelated to the output data. Here, we dropped the uncorrelated features like rating, timestamp, user-spoiler ratio, book-spoiler ratio and the respective ids since they were not related to predicting whether the review was a spoiler or not. It works very well with binary classification problems, in this case the classification being 0 or 1. Moreover, it is robust to correlated features that if f1 and f2 are perfectly correlated, regression will simply assign part of the weight to w1 and another to w2[2]. Thus, it works better with larger dataset when compared to Naïve Bayes.

The SVM hyper parameters are Cost -C and gamma. It is not that easy to fine-tune these hyper-parameters and visualize their results[3]. Naïve Bayes works well with smaller datasets and thus fails to perform as well as Logistic Regression. Random Forests model is large and results in over fitting. When the data is dynamic, RF fails to perform too well when compared to other models. LDA needs relatively larger dataset so that they can scrape the document for topics. If there aren't enough words in the document, there isn't enough data for it. Secondly, the reviews were very short and very-context dependent, thus making it unreliable[4].

Chapter 4 – Discussion and Conclusions

4.1 Decisions made:

We started with working on small chunks of data by converting it into pandas dataframe and applying machine learning techniques after preprocessing the data. However, we realized that it was computationally intensive to perform these on the entire dataset and thus we switched to Pyspark implementation so that the computations are faster and efficient.

4.2 Difficulties faced:

- The dataset selected is huge thus the preprocessing and data cleaning required a lot of time.
- The number of columns available were less and thus we had to create new columns to enrich the dataset
- However, after analysis we found out that these columns were only slightly correlated to predicting whether the review has spoiler or not
- Text mining techniques were computationally intensive and time consuming

4.3 Things that worked:

- The transition from vanilla pandas dataframe was simple and required less efforts
- With text mining and data cleaning, we got good results after training the model.
- Data visualization and preprocessing gave us a lot of insights on the dataset and thus we could drop the unwanted features and select the relevant ones

4.4 Things that did not work:

- We tried to introduce new variables like user-spoiler ratio and book-spoiler ratio to the dataset to enrich it, however we found out that they were not correlated to the "has_spoiler" feature and thus had to drop them
- For simplicity, we tried using pandas dataframe but it was consuming a lot of time and thus we had to drop that approach and switch to PySpark

4.5 Conclusion:

- We learnt how to handle large imbalanced datasets and apply classification algorithms on them.
- We also learnt that a lot of feature engineering and text mining is required when it comes to text datasets for the algorithms to perform well.
- Simple models may outperform complex machine learning models. In our case, we expected Random Forest to perform the best but the results showed that Logistic Regression outperformed both RF and LDA.
- We had to drop a lot of features based on the correlation matrix for the algorithms to perform well
- PySpark is faster for big data analysis when compared to other frameworks.

Chapter 5 – Project Plan and Task Distribution

5.1 Task Distribution:

Task	Responsibility
Dataset Selection	All
Data Exploration	Jasmine
Data Visualization	Zeeshan
Feature Engineering and selection	Neha
Research on classification and Regression Models	All
Text Mining	Jasmine
Logistic Regression, LDA	Zeeshan
Naïve Bayes, Random Forest Classifier	Neha
Documentation	All

References

1. Mengting Wan, Rishabh Misra, Ndapa Nakashole, Julian McAuley, "Fine-Grained Spoiler Detection from Large-Scale Review Corpora", in ACL'19.
2. Speech and Language Processing. Daniel Jurafsky & James H. Martin.
3. https://statinfer.com/204-6-8-svm-advantages-disadvantages-applications/
4. https://stackoverflow.com/questions/29786985/whats-the-disadvantage-of-lda-for-short-texts

Project Repository

Link: https://github.com/zeetherocker/Goodreads-Spoiler-Detection