Building a book's rating classifier for COMP30027 Report

Anonymous
word count: 2071

1. Introduction

A book's rating can be affected by many elements, from authors to description. In this task, we aim to build a model which can estimate the books' rating. We explore whether there is a relationship between the features and the rating of books, and to what extent rating is affected by other factors. Our dataset is from Goodreads, it's a social cataloging website, which provides a platform for individuals. Users can search the database of books and rate the books and write the reviews of books.

1.1 Task Statement

Our task is to construct and analyse a model which predicts a book's rating given several of its features including books Title, Author, Published date, Publisher, Description, and PageNumber. Our assumption is that Name, Author and Description would be a deciding factor for a book's rating.

1.2 Data overview

Dataset contains 2 csv: book_rating_train.csv with 10 features and 23063 instances, book_rating_test.csv with 9 columns and 5766 instances.

Features and explanation (Table 1):

Attribute	Explanation
Name	name of the book
Authors	Names of the authors of the book
PublishYear	the year of the book's release
PublishMonth	the month of the book's release
PublishDay	the day of the book's release
Publisher	a company or person issues the book
Language	the author use which language to write book, in abbreviated form
pagesNumber	number of pages the book contain
Description	a brief description of the book
rating_label	three levels of the books:3,4,5

Table 1- feature explanation

The boxplot of the feature "PublishDay" (Figure 1) has shown a slight difference in the mean values among the different labels of rating. To further explore the relationship between publication date and ratings, we also plotted the boxplots for "PublishYear" and "PublishMonth" (Figure 2). Interestingly, these boxplots displayed similar mean values across the various rating labels, we therefore assume these features may not be useful in distinguishing between class labels. Though these are only preliminary investigations on the data thus cannot be solid evidence for deciding the information contained in a feature.

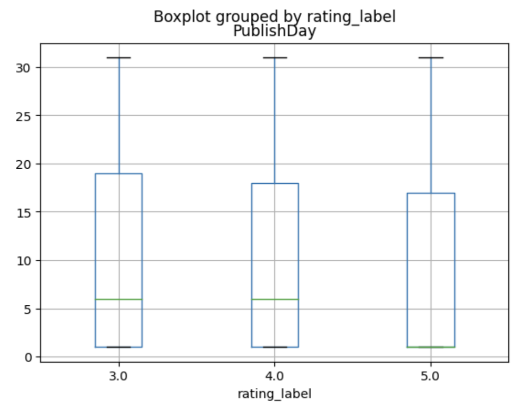
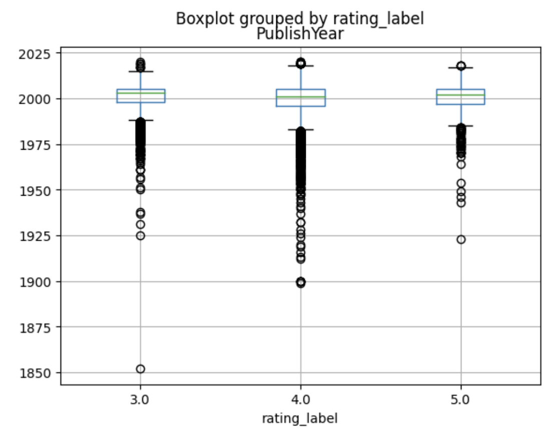


Figure 1- rating of book VS publish day.



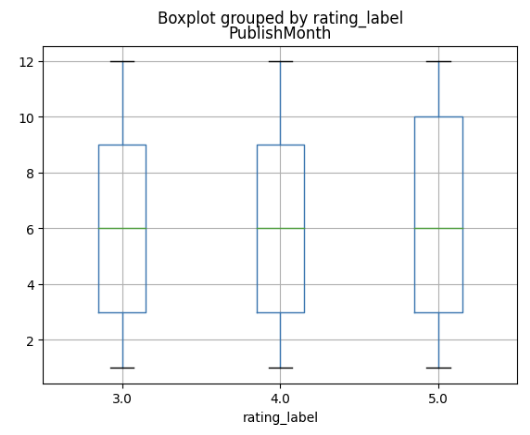


Figure 2- rating of book VS publish year (top) and rating of book VS publish month (bottom).

In addition, the data has 16301 unique authors and the minimum value of "PagesNumber" is 0 and maximum is 37000 which seems impossible further steps will be taken in preprocessing.

2. Methodology

2.1 Preprocessing

From the initial inspection, the "Language" column contains many empty values then computes the proportion of empty values in this column is almost 0.75. Although "Language" may contribute to estimate the rating, the amount of empty values are large and complex to correspond to the correct languages, so choosing to remove this column instead of filling. After processing the data has 8 features and 20753 instances.

At the same time, "Authors" and "Publisher " may have a relatively high correlation by intuition(since authors may be tied to publishers). Calculating the mutual information with categorical encoding on the two features resulted in 6.84 out of the lowest entropy of the two feature fields 10.42. Therefore we recognise having both of them wouldn't provide the worth of extra information as they increase feature space dimensionality. Therefore we decided to drop "Publisher".

Secondly we decided to implement doc2vec (Le and Mikolov, 2014) on all the text fields, especially on "Description" and "Name". The use of doc2vec generates a vector representation of a text feature, which we would expect a high amount of semantic information embedded inside the "Description". Also the "Name" field should also be a deciding factor for the book rating. We are expecting the "Name" field to be able to contain potential psychological implicative signs similar to "ClickBait" in social media terms.

Therefore based on their relative length we decided to set the vector size of "Description" to 100 while "Name" and "Author" to 20, as we have lower expected importance on the latter two shorter features.

2.2 Model Selection

For the given task we have considered various supervised models from Scikit-learn (Pedregosa et al., 2011) including Gaussian Naive Bayes for the reason that Naive Bayes's algorithm is extremely lightweight and also extremely simple, making it less likely to experience overfit from high dimensional data.

We also experimented with a support vector machine, in the consideration of its versatile kernel which can provide a more flexible decision boundary against data that are not linearly separable in its original feature space. Though the downside is that it's not very compatible with multiclass classification as it will be requiring more models corresponding to each class label, which is at least 3 for the task of this assignment.

However, we find that the Support Vector machine is capable of achieving a reasonably well accuracy score as a standalone model, with an accuracy of about 71%. Although even after a long lasting grid search for a better parameter, it exhibits a tendency to overfit the data.

Finally we also have logistic regression, for similar reasons as the choice on Naive Bayes, the logistic model is less computationally intensive, on top of that, it does not require the assumption of independence between features and therefore we expect it would be capable of handling more complex data.

Throughout the training process, we also discovered some limitation of the previous 3 model, and implemented:

- XGBoost(Chen and Guestrin, 2016)
- Stacking Ensemble learning with XGboost + Naive Bayes + Logistic regression

3. Critical Analysis

Upon initial training, all models seem to be affected by the unbalanced training and exhibit a different degree of unbalanced prediction class.

For instance, SVM showed an extreme bias toward predicting class of rating=4, where such class takes up 97% of all predictions and about 29% false negative rate (Figure 3).

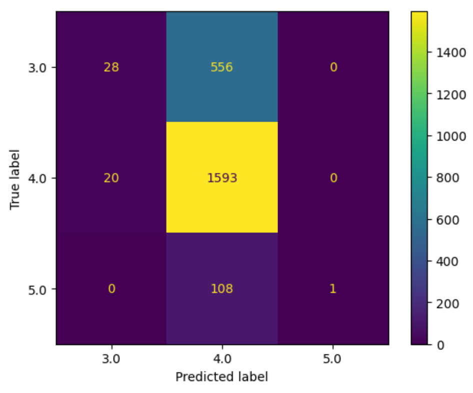


Figure 3- Confusion matrix of SVM

Our initial hypothesis to the unbalanced prediction class result is purely because of the extreme unbalanced class in training data, since class 4 takes up about 75% of all training cases.

3.1 Balancing class

In response to this issue, we compared the models with different strategies with SMOTE oversampling, referenced from another study(Chawla et al., 2002) where it would improve classifier performance compared with traditional oversampling methods. However, by comparing the confusion matrix result before and after applying SMOTE, we also observed that the models trained with SMOTE are highly biased, and are still having unbalanced predictions. And on top of that the accuracy score drops to 23% (Figure 4).

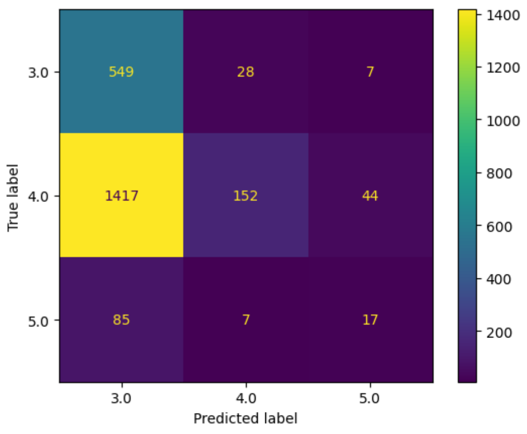


Figure 4- Confusion matrix of SMOTE

Upon further analysis, we concluded that the limited separability of our data within its feature space contributed to the non-optimal performance of SMOTE. Since applying SMOTE, the algorithm generates synthetic minority class instances by interpolating its features(Chawla et al., 2002), which given our data and the preprocessing we applied, the feature of minority class is densely distributed among the majority class, for which the limited range introduced noise and further deteriorated our model's performance. To further illustrate this we constructed a scatter plot applying PCA to original feature space down to a 3D representational space to demonstrate the low separability (Figure 5):

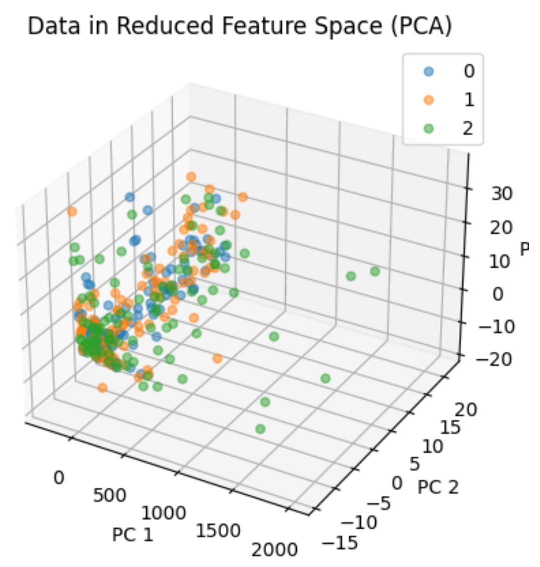


Figure 5- Sampled instance in their down graded feature space

We also attempted undersampling data, as an alternative to counter the class imbalance. However, while this approach successfully balanced the prediction, the model trained with under-sampled data shows an increase in model variance, indicating the model was unable to capture the important feature therefore underfitting the data. This can be indicated by the low average F1 score of 0.3788 (Figure 6) .

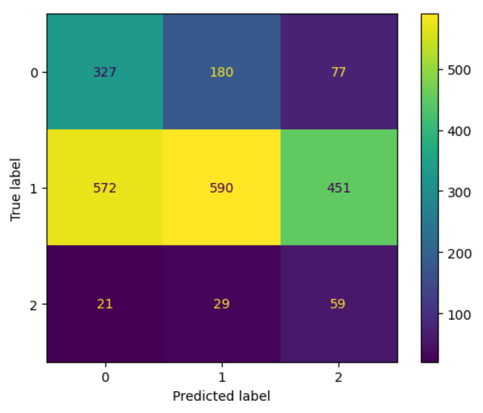


Figure 6- confusion matrix of undersampling

3.2 Overfitting

Another aspect we may be able to observe was the overfitting of the model. We observed that the Gaussian Naive Bayes struggled to handle the complexity of the problem, evidenced by the learning curve with 5 fold cross validation, the performance stabilizes as training instances increase, it tends to plateau around accuracy of 60% (Figure 7).

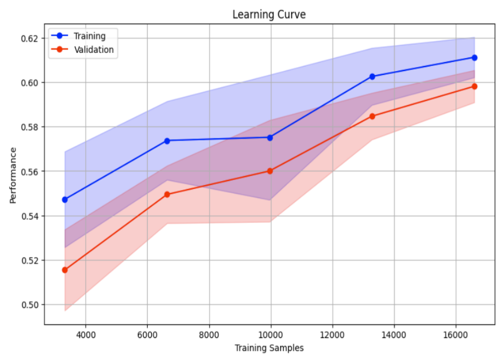


Figure 7- learning curve of Gaussian Naive Bayes

We reckon this observation suggests that the Gaussian Naive Bayes model is overly simplistic given high dimensional features, and thus unable to grasp important features. Resulting in a bottleneck in performance, Thus we render Naive Bayes to be an overly weak learner to be a stand alone model (Figure 8).

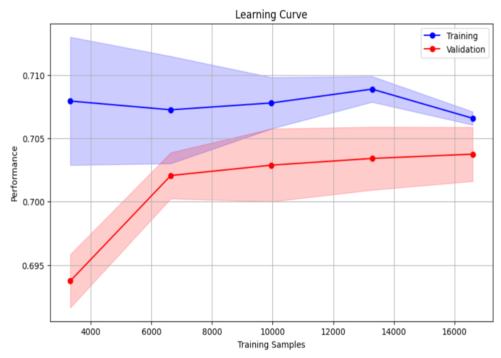


Figure 8- learning curve of Logistic regression model

Meanwhile, both Logistic regression and Support Vector Machine showed a better performance in comparison. However the validation curves plateaued as the number of training instances increased, we identify this behavior to be an indication of overfitting.

For this reason we attempted a boosting approach using XGBoost, XGBoost implements a regularised regression tree based boosting method along with shrinkage and column subsampling(Chen and Guestrin, 2016) to counter the problem of overfitting.

4. Model Training

To optimize the performance of the XGBoost model, applied grid search over its hyperparameters, mainly focusing on max tree depth, leaf weight and L1&L2 regularisation. The regularisation plays a crucial role for the data we are handling since our data can be noisy, regularisation techniques may help mitigate the impact brought by our attempt to preserve more information.

To evaluate parameter combination during the grid search, we selected AUC-ROC score for the cross fold evaluation, reason being AUC-ROC considers the trade-off between sensitivity and specificity, allowing it to address better for the imbalanced classification problems, specifically AUC-ROC assess a model not only prioritising true positive rate but also controls for false positives. In such a way the model will penalise more false positive cases, effectively avoiding imbalanced classification in some way.

Also given the dimensionality of the preprocessed data, we focus our tuning on L1 and L2 hyperparameters. The two regularisation parameters calibrate the effective "pruning" of the feature: for a less important feature, L1 and L2 will drive its leaf weight closer to 0, consequently excluding features that do not contribute to the prediction of class label.

After fine-tuning the parameters of the XGBoost model, we achieved a slightly more promising result of 71.2% on the validation dataset, showing an improvement over the default configuration of the XGBoost model.

To further ensure the reliability of the results as well as to measure the model's capability on generalising data. Applying 10 fold cross validation on the tuned model yielded an average accuracy of 70.4% accuracy on the given dataset.

In our final attempt to improve the model's performance we explored ensemble learning by combining three base models: Logistic regression, Gaussian Naive Bayes and XGBoost. Logistic regression as a level 1 model.

We adopted the stacking method, where the prediction of the base models are fed into the meta model as input. Which allows us to take advantage of each base model and improve the model performance.

We considered adding Support Vector Machine into the stacking-ensemble model as well. However, because of the high feature dimension and the nature of Support Vector Machine, as well as the large amount of data and multiclass classification. We decided Support Vector Machine is not particularly suited for the stacking ensemble model. As the intense computational complexity is considered unfeasible.

5. Results

Examining the learning curve of the final ensemble model, the validation score is stable and is slowly increasing, which could be a sign for which the model has learned the underlying feature pattern reasonably well, producing a consistent validation set (Figure 9).

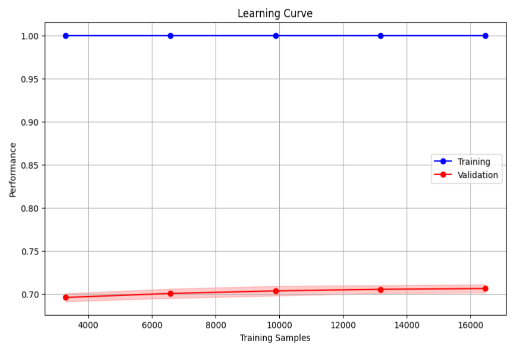


Figure 9- Learning curve of stacking ensemble model

The table above shows a summary of the top 5 features of their relative importance (Table 2). Description vector components are, as expected, a crucial deciding factor for the rating of the book. Though the Page Numbers was unexpectedly a strong dependency of book rating, based on our intuition this perhaps decides the extent to which a book can convey information in a concise manner.

Feature	Importance
PageNumber	0.02474
d2v_desc_4	0.02068
d2v_desc_7	0.02065
d2v_desc_18	0.01964
PublishYear	0.01917

Table 2- Feature and their relative importance

6. Conclusions

In conclusion, our analysis utilised multiple supervised models including Gaussian Naive Bayes, Logistic Regression, while also explored into boosting and stacking ensemble methods to further improve performance. Through evaluation and experimentation, We found that Gaussian Naive Bayes alone was not suitable for the complexity of the problem. While XGBoost exhibited better performance, and through further tuning, achieved accuracy of 71.2% with XGBoost. Moreover, we assembled these learners into a stacking model to harness their respective strength to achieve a more stable model. Finally, from the importance of features analysed by XGBoost, the rating of the book is surprisingly highly dependent on the page number.

7. References

Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*, 785–794. https://doi.org/10.1145/2939672.2939785

Ranstam, J., & Cook, J. A. (2018). LASSO regression. *British Journal of Surgery*, *105*(10), 1348–1348. https://doi.org/10.1002/bjs.10895

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. and Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, [online] 12(85), pp.2825–2830. Available at: https://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html.

Chen, T. and Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. [online] Available at: https://arxiv.org/pdf/1603.02754.pdf.

Le, Q. and Mikolov, T. (2014). Distributed Representations of Sentences and Documents. [online] Available at: https://arxiv.org/pdf/1405.4053.pdf.