Royal Road Initiative

Tim Kueny
kuenytc5@vt.edu

Abstract

The aim of the project was to apply machine learning algorithms to determine whether authors on the site RoyalRoad would be likely to drop or discontinue writing their stories. The project requires a combination of web scraping and the application of machine learning algorithms. The Node.js libraries Axios and Cheerio were used to scrape and parse, respectively. We then applied the KNN algorithm to the 20,000 records that were collected.

1 Introduction

RoyalRoad.com is a website that allows users to share partial or complete works of literature with anyone who uses the site. There are hundreds of thousands of authors on the site. Readers can rate and comment their opinions on a novel and likewise our team is able to read this data. Our machine learning algorithm looks at categories like user ratings, genre, pages completed, and activity, in order to make an educated decision. Our goal is for our machine learning algorithm to recognize which books are going to be successful and which will fail. We will hope this application can be used for authors to get another form of feedback on their work i.e. how likely their book is to succeed.

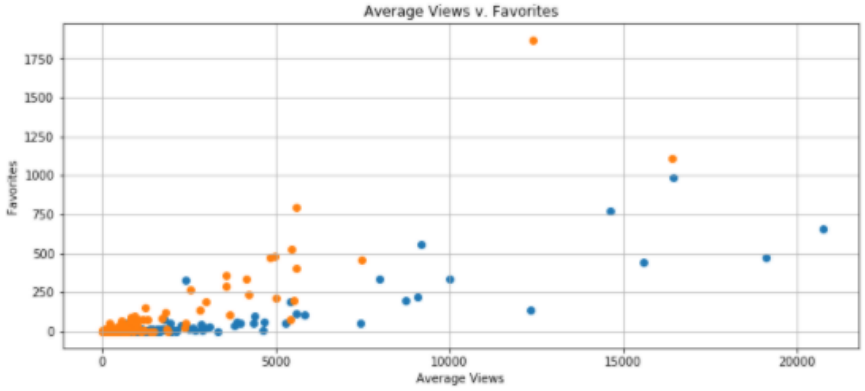
2 Experiment

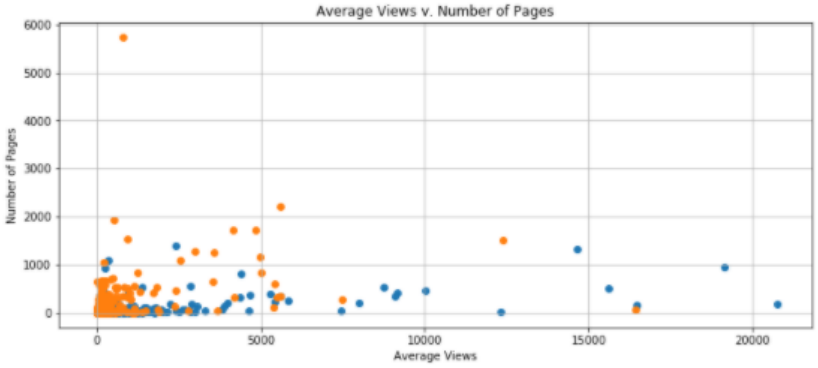
2.1 Data Collection

We have obtained our data from the RoyalRoad website through web scraping utilizing Axios and Cheerio. Those points that we have collected include: words published per day, chapter(s) published per day, views per chapter, Amount of comments received per chapter, the genre, and other related tags such as content warnings. Currently over 17,000 points of data have been collected with our technique. We have split the data so that we have a training set of size 15000 and a testing set of size 2000. The data is not repeated and does not cross over between the testing and training set.

Below we can see of how the collection of data can be used to find correlations. We an see that by graphing some of the targeted features, we can see that many of the data points are partially linearly separable. We can use this information to guide our thinking in how we went about completing this experiment. Basically we can either find a system of equations to find a line of best fit that separates the data, or we can group the data together and those groups will be divided by an arbitrary line

Submitted to 34th Conference on Neural Information Processing Systems (NeurIPS 2020). Do not distribute.





We have put all of the factors into one vector, and each vector into an array. From this data we can add weights to each item in the vector with SVD, or simply compare them as is as we will do with KNN.

3 Methods

We will use cross-validation on the data scraped from the site for training and testing data. We have built test cases saying if a project has been abandoned, completed. Because we only wish to determine the percent accuracy of the algorithm in determining whether a book has been abandoned or not, we know that this problem will be a classification problem. A book can be classified as will be completed or abandoned: true or false respectively. Because we have a binary classification problem, we will be evaluating results with an absolute loss function. We have come up with two different ways of classifying the problem. We Will use a KNN classifier and a SVD classifier.

3.1 KNN Classifier

With all of our different variables in an array we can simply find the distance between each vector in the array. There are a myriad of distance functions we can test, so we must test each against each other. After we test all distance formulas, we must find the K value for grouping. We will do this with our most accurate distance formula. Because there is no use of weights in the KNN algorithm, some of our data values may matter more than others even if they should not. For example, the genre would not matter as much as pages typed per day. Knowing this we can also add and remove some data in order to hone into which data points are more significant than others.

3.2 SVD Classifier

With the SVD Classifier, we are hoping to create weights for each index that would better balance some of the disparages between some of the items of interest. We would hope that we would better be able to pick up patterns that the KNN classifier overlooks by its design.

First we found the pseudo-inverse of our our data set with respect to our 71 features (many of these features are genres and have a small impact on our matrix). after we quire the matrix, we will use that matrix and run it through a linear classifier. The linear classifier will adjust the weights by adjusting the sum of column the values based on correct or incorrect positions.

4 Conclusions

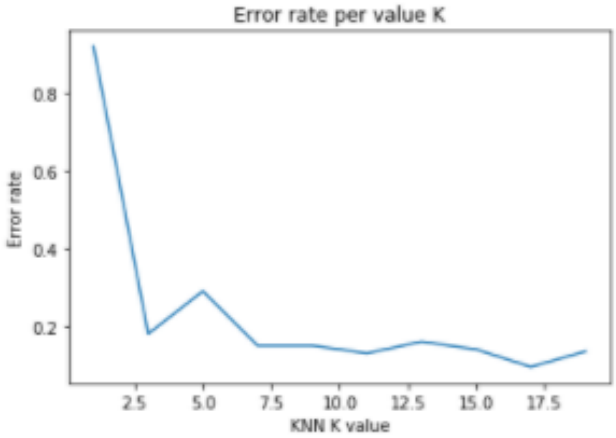
Here we will discuss results that we have obtained and work that we would like to improve upon.

4.1 KNN

Currently , we have tested cosine distance, Euclidean distance, Manhattan Distance and squared chord distance. The squared chord distance has shown the lowest error with 0.12 error with a KNN classifier with K being equal to 3. The Euclidean and Manhattan Distances were very close to the error rate of the Squared Chord Distance, but the squared chord distance does a good job of minimizing error between two points. The cosine distance was slightly better than the Euclidean and Manhattan distances but not the best.

We have tested multiple different K values and found that having a KNN classifier with K equal to 9 or 11 could lower the error rate. we have found that extending our KNN classifier has improved its error rate by increasing the K value to 17. In our testing with cross validation, our error went down to 0.1. In the graph below you can see a graph of the error rate with respect to the current K value with our classifier. Having such a large K for our K nearest neighbors number would leave me to believe that there are a lot of items in our data that are outliers.

The Genre seems to matter more fore the SVD model than the KNN model. When genre was removed from the KNN model, there was no noticeable change in the error rate. I would surmise that this is because genre is represented by a 1 if it is that genre and a 0 otherwise. with the KNN value, this would not effect any of our distances that we calculate when we consider the data.



4.2 SVD

When testing our SVD method we calculated the Pseudo-Inverse of the Data which would leave us a base matrix of size 71x71, the size of of the what data we have collected for each title. after finding this matrix, we can use it as our slope for a given title. Each title array, multiplied by our Pseudo-Inverse(PSD) matrix, will give us a prediction true or false. Initially, using our initial PSD we obtained and error rate of about 0.35. we were able to use linear regression to improve our results

The genre tags for the SVD does add another aspect to the data that is makes considering the genre very significant. The values of the weights in the matrix that correspond to a genre weight vary in magnitude and can be positive or negative. this shows us something the KNN model did not. The SVD model shows that the genres do have an varying importance in respect to the other weights. It also shows some genres have a negative impact on the probability of the book succeeding.

Linear regression took only one cycle through the data to make a decision. it seems as though just taking the derivative of the PSD and adjusting the weights after one iteration was enough to adjust the weights to the optimal level. Note that the weights were effected at each iteration of our linear regression algorithm, so really after 15000 time the weights were changed, we have a regression rate that remains practically stationary. essentially after running it through our linear regression algorithm, we were able to have an error rate of 0.101, only slightly better than the best of our KNN classifier.

4.3 Observations worth further investigation

The next logical step I could see that would improve results would be to use a soft-model SVM model to calculate a better way to separate the data. we have already seen that the data we have is partially linearly separable and a soft margin SVM has been used in the past to solve similar problems.

I would also like to layer the genre in some way so that maybe we would first make a prediction based solely on the genre, than analyse other aspects of the data to influence the algorithms decision.

I can also see that using Clustering and K-Means would also be a beneficial addition into how to better classify the data. A K means clustering algorithm could possibly give perspective to the data that we are giving the model. We could add subsection and give greater emphasis on items like the genre or the average views per day.

References

[1] Abdullatif, H. (2019, January 2). You Don't Know SVD (Singular Value Decomposition) - Towards Data Science. Medium. https://towardsdatascience.com/svd-8c2f72e264f

[2] Seo, J. D. (2018, November 21). Step by Step Backpropagation Through Singular Value Decomposition with Code in Tensorflow. Medium. https://towardsdatascience.com/step-by-step-backpropagation-through-singular-value-decomposition-with-code-in-tensorflow-8056f7fbcbf3

[3]Catalin Ionescu, Orestis Vantzos, and Cristian Sminchisescu. (April 15, 2016). Training Deep Networks with Structured Layers by Matrix Backpropagation. https://arxiv.org/pdf/1509.07838.pdf

[4] James Townsend. (August 10, 2016). Differentiating the Singular Value Decomposition. https://j-towns.github.io/papers/svd-derivative.pdf