Charles Garman

Valentine Schmidt

Joel Thompson

Machine Learning Project Report

The Dataset-

The dataset used for this project consists of the top fifty bestselling books from Amazon recorded annually, beginning in 2009 and having 2019 as the last year recorded. It can be found on Kaggle (Saalu, 2020). The file, “best sellers with categories.csv”, has seven columns-

Name: lists the name of the book

Author: lists the author of the book

User Rating: lists the overall rating on a five point scale with the format #.# where # is a number and the decimal and following number are excluded if the rating is a full number

Reviews: the number of reviews that the book has

Price: the price of the book in dollars as of October 13, 2020 represented as a number with no symbols

Year: the year the book ranked on the top 50 bestsellers list

Genre: whether the book is fiction or non-fiction

The dataset contains 550 entries.

The Objectives-

We would like to predict the user rating value given most of the other columns as input attributes, as the name and author of the book being considered in prediction would simply train the model to remember how well a specific book was received or possibly memorize the scores of specific authors, both scenarios would fail to yield more general patterns. The motivation of this project is that it seems intuitive that there would be some correlation between the input attributes and the user ratings, but how do these human intuitions compare to machine learning results? We hope to answer that question in this specific case, and, if significant patterns are revealed, they would prompt questions pertaining to why certain attributes correlate with user ratings. Regression techniques will be used as the predicted value is numeric. Of note is that the same book can be listed in the dataset more than once if it makes it onto the top 50 bestselling books list for two or more years. In testing we will run our procedures on this dataset without modification, as the same book being listed multiple times could be indicative that it describes the traits of bestselling books that much more and should be considered additional times, and we will also use the dataset with the duplicates removed, as they could be obfuscating the real patterns by being counted more than once, and we will compare results to see which is seemingly more appropriate. The results of our testing could indicate if there is a correlation that can be found between the input attributes, which include the number of reviews, the price, the year it made it onto the top fifty bestsellers list, and the genre, and the user rating. If prediction is largely unsuccessful then that demonstrates that no reasonable estimate can be obtained regarding how well a bestselling book will be rated given the aforementioned attributes, and that there are likely more complex patterns that influence how well a bestselling book is regarded.

The Design-

Programming was done in Spyder (Spyder, n.d.), but first the data had to undergo some preprocessing. In order to remove the name and author columns from the dataset, we opened the file in Microsoft Excel (Microsoft, n.d.) and deleted them. We also repositioned the User Rating column to be the rightmost one, as per convention. We then saved this copy as a new file, “bestsellnormtrim.csv”.

Table

Description automatically generated

Given that this is a regression problem, the genre column is problematic, as it is categorical and not numerical. While we do not automatically know the best way to numerically encode this information, we can write a function which operates on a pandas (pandas, n.d.) DataFrame that contains the dataset to use a numeric parameter as the value that will replace the categorical values. It was decided that Fiction will be mapped to the value supplied and Non Fiction will be mapped to the value supplied with inverted parity. The function will then return the modified dataset. After the Genre values have been normalized the only two outcomes for the complete dataset are that

Fiction = 1.135481

Non Fiction = -0.879082

if the supplied number is positive and that

Fiction = -1.135481

Non Fiction = 0.879082

if the supplied number is negative. A similar observation is made on the no duplicate dataset, with

Fiction = 1.088167

Non Fiction = -0.916351

if the supplied number is positive and that

Fiction = -1.088167

Non Fiction = 0.916351

if the supplied number is negative. Zero is not to be used. Because of these observations, it is unlikely that changing the way Genre is encoded will improve the results.

A screenshot of a computer

Description automatically generated with medium confidence

This function will be used in testing. With regards to normalizing the columns so they all have a mean of 0 and a standard deviation of 1, a simple function can be written to normalize the first three columns and another function normalizes the Genre column. This separation is so the first three columns only need to be normalized once and the Genre column can easily be normalized after a particular numerical conversion is applied to a dataset that already has its first three columns normalized. Both functions, like the replaceGenre function, utilize the pandas DataFrame construct.

A screenshot of a computer

Description automatically generated with medium confidence

In order to generate the alternate, duplicate free dataset, the original dataset, the one with the Name and Author columns intact, is opened in Microsoft Excel. The Data tab is selected and then the Remove Duplicates option that is listed in the Data Tools section is used. Duplicates are removed based on the value of the Name column, as duplicate books should be eliminated.

Graphical user interface, text, application

Description automatically generated

After selecting OK in the above window, 350 entries remain in the dataset. As they were sorted alphabetically, among multiple entries for the same book only the oldest one will survive, so the Year column now indicates the first year in which the book was on the top 50 bestsellers list, as subsequent placements are no longer recorded. From there the Name and Author columns are deleted and the User Rating column is made to be the rightmost one, as before, and this file is saved as “bestsellnoduptrim.csv”.

Table

Description automatically generated

Now both of these datasets can be tested with ridge regression, neural networks, and support vector machines using train-test split and cross validation as model selection measures.

The Training Process-

For ease of use, the first four columns in the pandas DataFrame and the final column were converted to two numpy (NumPy, n.d.) arrays, X and y respectively.

A picture containing text

Description automatically generated

To start off simple, ridge regression was the first algorithm used, as it completes quickly and, being the least sophisticated model we will use, it can provide a base line with which to compare the performance of the other models. The model selection techniques that we used were a random train-test split and 5-fold cross validation. Both can perform scoring, and the random train-test split provides information on the performance of the model in predicting the training and testing data. The random state for the random train-test split was set to be 1. Altering the random state did not produce meaningful differences, as the training and testing scores are quite low, even when the optimal alpha value was found. This could be indicative of underfitting, which is sensible considering that this is a simple linear model. 10% of the data was used for testing in order for the training to be as robust as it can be while also having a decent number of test entries, but the scores were still subpar, and altering the percentage of test data did not remedy this. As expected, 5-fold cross validation also yielded very low scores, and altering the number of folds did not result in any meaningful differences. Five folds were chosen specifically to have consistency with tests performed on the other two algorithms, which are far more intensive and any reduction in computational cost by having relatively few folds could be quite helpful. The aforementioned was true for both the complete and the duplicate-free datasets. Although the highest scores were different and resulted from different alpha values, all scores were low. The differences in scores would be meaningful if a different model we used performed decently. Altering the way Genre was encoded did not affect the scores at all, as anticipated.

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generated

A picture containing chart

Description automatically generatedChart

Description automatically generated with medium confidence

Afterwards, a neural network regressor was used on both datasets. To try to find the best score in a reasonable amount of time, different neural networks were scored with all combinations of five layers where each layer can have one to twenty inclusive neurons. The combination that produced the highest score was noted and then used consistently in another loop that obtained several scores using changing alphas, which varied from 0 to 10 will 100 values, for regularization. While this does not guarantee that the best network-alpha combination is found, it does provide a reasonable indication of performance and is far less strenuous. This procedure was done via a train-test split with the same parameters as before and 5-fold cross validation. All neural networks used had a random state of 1, 5,000,000,000 set as the maximum number of iterations to avoid non-convergence, the logistic sigmoid function used as the activation function, and the solver used was lbfgs, as it is recommended for relatively smaller datasets, those that have fewer than 1,000 entries. (<https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html>)

The alpha values versus the scores were plotted for both datasets for each method of model selection used on them.

Graphical user interface, application

Description automatically generatedGraphical user interface, application

Description automatically generated

Graphical user interface

Description automatically generated with medium confidenceGraphical user interface, chart

Description automatically generated

For the complete dataset, the best network used for the train-test split was (1, 2, 8, 10, 11), which was surprising since it begins with one neuron and ascends, and it performed better than one with the maximum number of neurons. However, this is unlikely to be meaningful, considering that all scores are very low. The best network used for the 5-fold cross validation was also (1, 2, 8, 10, 11), which was surprising as well, but again not likely to be meaningful. For the no duplicate dataset, for the train-test split, the best network was (2, 4, 6, 8, 10), while the best network for the 5-fold cross validation was the familiar (1, 2, 8, 10, 11) network. Considering just how low these scores are, it was beginning to seem unlikely that there was any semblance of a pattern that could be found in either dataset. A more robust neural network could potentially perform better, but it is not likely to lead to the drastic spike in performance that is needed for results to be significant, and the limits of the machines we had access to were encountered with the testing already completed. Once again, altering the train-test split and the way Genre was encoded did nothing to alter the results. The final model that we used to attempt to predict the user rating value was a support vector machine, or SVR in sklearn. To start, the same train-test split that was used in the previous models is used here, and the RBF kernel is used. A two dimensional search is done to find the optimal values of gamma and C. This is done by, for every gamma value, taking note of what value of C produced the highest score, and then writing a list containing the gamma value, the C value that produced the highest score, and the highest score itself into a 2D list, in that order. Both the gamma and C values were varied from 0.001 to 5 with 100 values being used. For the complete, duplicate-having set, the maximum score returned from the 2D list was 0.41444746017981693, which resulted from a gamma value of 3.8891111111111103 and a C value of 1.566343434343434. This was found by interacting with the Python variables after the program ran, which allowed directly looking into the results as well as writing short loops and other commands to find specific entries and relevant details regarding them. For example, the command “max([ls[2] for ls in rbf\_tts\_scores])” can be used to find the maximum score in this case. Expanding the ranges of the search did not provide a significantly greater score. Next the polynomial kernel was tested via a three dimensional search, where the degree was varied from 1 to 5, the gamma value from 0.001 to 1, and the C value from 0.001 to 1, in that order, while only five values for gamma and C were used due to the computational cost. The highest score found was 0.13965349960420514, which was yielded by a degree of 5, a gamma value of 0.25075, and a C value of 0.5005. Additional exploration was not viable due to the extensive amount of time required, and the scores recovered from the first test did not seem competitive enough to justify use of the polynomial kernel either. Due to the intensive nature of these tests, while they may all reside in the same Python file in their completed forms, during testing only one was left as not being commented out. After some adjustments the sigmoid kernel was tested in a way very similar to the two dimensional search done on the RBF kernel, except the gamma values varied from 0.0001 to 0.1. The maximum score was 0.05392363646599552 with a gamma value of 0.004136363636363637 and a C value of 4.596040404040404. For an instance of LinearSVR only the C value was varied, so a graph could be created and training scores were recorded as well, seeing as they would not crowd the results as they were already sparse. The C value varied from 0.001 to 0.25 (as any higher lead to a lack of convergence) and had 100 values.

Graphical user interface, application

Description automatically generated

The highest testing score achieved was 0.11186684776852407 which was obtained from a C value of 0.03118181818181818. The same procedures were used with 5-fold cross validation. For the RBF kernel, the search was adjusted so that gamma values would be 0.001 to 0.1 and C values would be 0.001 to 10.0, the highest score was 0.07674351016014633 which was obtained from a gamma value of 0.057 and a C value of 0.203. When running the same procedure that was done on the train-test split with the polynomial kernel with 5-fold cross validation instead, the maximum score was found with degree 1 and a gamma value of 1. To refine the search, only degree 1 was investigated, 100 gamma values were used from 0.001 to 2.0, and 100 C values were used from 0.001 to 1.0. The maximum score retrieved from this, which was slightly higher than the one obtained before said adjustments were made, was 0.033862876048245893, which resulted from a gamma value of 0.3644545454545455 and a C value of 0.08172727272727273. For the sigmoid kernel both gamma and C had 100 values from 0.001 to 5.0, and the highest score obtained was 0.030510375792903565, with a gamma value of 0.001 and a C value of 3.5356464646464643. None of the scores found by varying the ranges for gamma and C were significantly higher than this. For an instance of LinearSVR the C value was varied from 0.001 to 0.13 with 100 values, as a C value above 0.13 lead to a lack of convergence. The cross validation scores were plotted along the C value X-axis.

Chart

Description automatically generated with medium confidence

The highest score was 0.04994548115393212 which was obtained from a C value of 0.05963636363636364. Similar procedures were conducted on the no duplicate dataset, with all relevant parameters and results in the table below.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Kernel | Model Selection | Gamma (min, max, number of values) | C value (min, max, number of values) | Degree (min, max, number of values) | Highest Score |
| RBF | Train-Test Split | 3.131686868686868 (0.001, 5.0, 100) | 0.7584242424242423  (0.001, 5.0, 100) | N/A | 0.3119525826546068 |
| Polynomial | Train-Test Split | 0.25075 (0.001, 1, 5) | 0.25075 (0.001, 1, 5) | 3 (1, 5, 5) | 0.1316687492454337 |
| Sigmoid | Train-Test Split | 0.028354545454545455 (0.0001, 0.1, 100) | 1.0613939393939391 (0.001, 5.0, 100) | N/A | 0.26461390590144696 |
| Linear | Train-Test Split | N/A | 0.06258585858585858 (0.001, 0.47, 100) | N/A | 0.13114809661809312 |
| RBF | 5-fold Cross Validation | 0.203 (0.001, 10.0, 100) | 0.10200000000000001 (0.001, 10.0, 100) | N/A | 0.03906219174649741 |
| Polynomial | 5-fold Cross Validation | 0.4248181818181818 (0.001, 1.0, 100) | 0.011090909090909092 (0.001, 1.0, 100) | 1 (1, 1, 1) | 0.018921037556236753 |
| Sigmoid | 5-fold Cross Validation | 0.024 (0.001, 0.1, 100) | 0.36894949494949497 (0.001, 0.5, 100) | N/A | 0.02976639513478425 |
| Linear | 5-fold Cross Validation | N/A | 0.08077777777777778 (0.001, 0.36, 100) | N/A | 0.012312853044833805 |

The graphs produced by the linear kernel are displayed below:

Graphical user interface, application

Description automatically generated with medium confidenceA picture containing chart

Description automatically generated

Results and Conclusions-

Below is a table comparing the best results obtained for each type of model, each model selection measure, and the parameters that produced those best scores, for the complete dataset.

|  |  |  |  |
| --- | --- | --- | --- |
| Algorithm | Model Selection | Parameters | Score |
| Ridge Regression | Train-Test Split | alpha=0 | 0.09859999943205422 |
| Neural Network | Train-Test Split | hidden\_layer\_size=(1, 2, 8, 10, 11), random\_state=1, max\_iter=5000000000, activation=’logistic’, alpha0, solver=’lbfgs’ | -0.011 |
| Support Vector Machine | Train-Test Split | kernel=’rbf’, gamma=3.8891111111111103, C=1.566343434343434 | 0.41444746017981693 |
| Ridge Regression | 5-fold Cross Validation | alpha=38.27827827827828 | 0.04794605684573605 |
| Neural Network | 5-fold Cross Validation | hidden\_layer\_size=(1, 2, 8, 10, 11), random\_state=1, max\_iter=5000000000, activation=’logistic’, alpha0, solver=’lbfgs’ | -0.0181 |
| Support Vector Machine | 5-fold Cross Validation | kernel=’rbf’, gamma=0.057, C=0.203 | 0.07674351016014633 |

Below is a similar table for the no duplicate dataset.

|  |  |  |  |
| --- | --- | --- | --- |
| Algorithm | Model Selection | Parameters | Score |
| Ridge Regression | Train-Test Split | alpha=0 | 0.14991272421329827 |
| Neural Network | Train-Test Split | hidden\_layer\_size=(2, 4, 6, 8, 10), random\_state=1, max\_iter=5000000000, activation=’logistic’, alpha0, solver=’lbfgs’ | 0.0007 |
| Support Vector Machine | Train-Test Split | kernel=’rbf’, gamma=3.131686868686868, C=0.7584242424242423 | 0.3119525826546068 |
| Ridge Regression | 5-fold Cross Validation | alpha=79.43943943943944 | 0.02578389774682539 |
| Neural Network | 5-fold Cross Validation | hidden\_layer\_size=(1, 2, 8, 10, 11), random\_state=1, max\_iter=5000000000, activation=’logistic’, alpha3.5, solver=’lbfgs’ | -0.01312 |
| Support Vector Machine | 5-fold Cross Validation | kernel=’rbf’, gamma=0.203, C=0.10200000000000001 | 0.03906219174649741 |

Some observations emerge from these results. Regardless of the dataset they are run on or the model selection used to compare them, the support vector machine performed the best, ridge regression was the next best, and neural networks did the worst. It is difficult to find meaning in this, however, since all scores are quite low, showing that it is rather unlikely that patterns can be found in either dataset. Another observation is that scores are overall higher for the dataset that contains duplicates, showing that it may be beneficial to count the same book more than once if it made it to the bestsellers list multiple times, their many placings could indicate that they possess traits that are more characteristic of bestselling books. Unfortunately, this cannot be substantively believed, considering that all of the scores were so low. Finally, the scores for the train-test split were significantly higher than those from the 5-fold cross validation. At one point the support vector machine even got a comparatively high score of 0.41444746017981693 on the complete dataset, only to get a paltry score of 0.07674351016014633 when using the more representative cross validation. This drastic difference highlights the danger in only relying on a random train-test split, especially on datasets that are relatively small, such as this one. Despite seemingly sensible intuitions about how the number of reviews, price, year of release, and genre could have some correlations with the user ratings for bestselling books, not only were scores low, they were consistently and significantly low. This strongly implies that there are more dimensions that determine how well a bestselling book will be received, and/or that the patterns in the input attributes are not obvious and require the processing of significantly more data points for bestselling books to be uncovered.

Resources

176coding. (2016, February 26). *Convert select columns in pandas Dataframe to numpy array*. Stack Overflow. Retrieved November 15, 2021, from <https://stackoverflow.com/questions/31789160/convert-select-columns-in-pandas-dataframe-to-numpy-array>

H, J. (2019, July 1). *A Python beginner’s look at .loc*. Towards Data Science. Retrieved November 15, 2021, from <https://towardsdatascience.com/a-python-beginners-look-at-loc-part-2-bddef7dfa7f2>

Matplotlib. (2021, November 15). *Matplotlib – Visualization with Python*. Retrieved November 21, 2021, from <https://matplotlib.org/>

Microsoft. (n.d.). *Microsoft Excel*. Retrieved November 15, 2021, from <https://www.microsoft.com/en-us/microsoft-365/excel>

NumPy. (n.d.). *NumPy*. Retrieved November 15, 2021, from <https://numpy.org/>

Pandas. (n.d.). *Pandas – Python data analysis library*. Retrieved November 15, 2021, from <https://pandas.pydata.org/>

Rustagi, D. (2020, December 11). *Normalize a column in pandas*. GeeksforGeeks. Retrieved November 15, 2021, from <https://www.geeksforgeeks.org/normalize-a-column-in-pandas/>

Saalu, S. (2020, October 13). *Amazon Top 50 bestselling books 2009 - 2019*. Kaggle. Retrieved November 15, 2021, from <https://www.kaggle.com/sootersaalu/amazon-top-50-bestselling-books-2009-2019>

Scikit-Learn. (n.d.). *Scikit-Learn: Machine learning in Python – Scikit-Learn 1.0.1 documentation*. Retrieved November 15, 2021, from <https://scikit-learn.org/stable/>

Spyder. (n.d.). *Home — Spyder IDE*. Retrieved November 15, 2021, from <https://www.spyder-ide.org/>