# Capstone: Fixing Broken Links in Wikipedia

*Entity Resolution of Internet Archive Books and Wikipedia Citations*

Alex Bass, Maxwell Jones

Spring 2023

DS 6015

Abbas Kazemipour

## Introduction

Wikipedia is one of the most popular online encyclopedias, with millions of articles covering a wide range of topics. As a collaborative platform, Wikipedia relies on the contributions of volunteer editors to maintain and improve its content. One of the most important aspects of editing Wikipedia articles is ensuring that all citations and references are accurate and up-to-date. However, broken links and missing citations are a common problem on Wikipedia, which can lead to inaccurate or incomplete information. Using the Internet Archive (IA), a digital library that contains millions of books, documents, and other media useful for finding and accessing digital copies of books that are no longer in print or are otherwise unavailable (*Books : Free Texts : Free Download, Borrow and Streaming : Internet Archive*, n.d.), we sought to match each book citation with its corresponding IA link, ensuring the link does not break and provide more value to Wikipedia users having access to books freely available on the Internet Archive. Thus, this paper discusses the problem of broken book citations on Wikipedia and proposes a machine learning methodology for fixing them by linking them to their appropriate Internet Archive match.

## Problem Description

Broken book citations on Wikipedia occur when a reference to a book leads to a dead link or an incorrect URL. This problem is particularly challenging to fix because it requires finding the correct version of the book that was originally cited. Many book titles have multiple editions or are reprinted by different publishers, which can make it difficult to determine the correct version of the book. To solve this, we aimed to build a machine learning model with various natural language processing methodologies to correctly identify each Wikipedia book's corresponding IA link. In order to create a machine learning approach to identifying correct book

citation links, we first needed to obtain an appropriate dataset containing every book citation from Wikipedia along with corresponding IA links.

## Data Collection

For our Wikipedia book citations, we first downloaded all of Wikipedia, approximately 23 GB of text data. We then parsed through to only obtain citations of the book template, filtering down our dataset to about 2.5 GB. Each book citation contained various categories, though not every citation shared the same categories–some citations only contained the title, while some citations contained many categories such as language and tertiary author. In total, there were over 750 distinct categories (Wikipedia contributors, n.d.), with many categories containing inconsistencies, such as the date and year categories both containing just the year the book was published. In order to generalize our model, we decided to only use title, author, publisher, and date of publication.

For our corresponding IA links, we searched every title, author, publisher, and date combination gained from each Wikipedia book citation into the IA API. As the information returned by the IA often included multiple different books, we joined every link returned for each book citation search query–our model would then need to predict whether the link returned by the search query was a match or not. Of course, the Wikipedia book citation sometimes would not match with any of the IA resources returned, but as we found the IA to be our best source for reliable links, we decided to move forward despite these occasional gaps, and having collected our dataset, we were ready to start building our model.

## Problems And Path To Solution

### *What is the ideal solution to this problem?*

Meeting with the team at Internet Archive, we discussed that a solution would be a matching algorithm between Wikipedia book citations and Internet Archive books that had a

99% percent likelihood of being a match if a match existed. In the end, they would like to apply this algorithm to every book citation on Wikipedia that did not have a book link in hopes of matching every possible freely available book in the Internet Archive to these citations. This would help enrich the content of Wikipedia if more book citations had a link to a free copy of the book cited.

In this model, the primary concern is reducing Type I error as much as possible, so as not to litter Wikipedia book citations with links to incorrect books. This would also engender trust between Wikipedia and Internet Archive allowing for possible working opportunities in the future. Of secondary importance, we want low Type II error so that we miss as few matches as possible.

***What happens when we try to match text exactly?***

When we first looked at this problem, the most reasonable and simplest solution was: Why couldn't we just match the characters exactly? The book titles that matched exactly would be matches and those that didn't would not be matches. However, the problem with this approach is there are books with the same title. For example, there are many references in Wikipedia to textbooks like "Principles of Biology." As it turns out, there are several textbooks with this exact title, so an exact match on title would not be useful. After we cleaned both titles (removing special characters and setting all the cases to lower), these were our title exact matching results.

| Method | Train Precision | Train Recall |
|---|---|---|
| 100% Title Match | 17.1% | 87% |

These results are hardly satisfactory when we are aiming for a 99% precision, so we would need a better method. As we looked through the results, this led us to our first problem.

***Problem 1: Duplicate Results Returned from the Internet Archive API***

As we combed through the false positives, we noticed that many books were not coded as matches when they actually were matches! They just had different Internet Archive links than the ones in the wikipedia citation! (Remember: To build our test set, we gathered all the Wikipedia book citations that already had Internet Archive links. We then queried these titles in the Internet Archive API Online Book Database which provides a unique web link for each book. We then deemed matches those records where the internet archive web link matched the wikipedia web link in the book citation.) This is problematic because all of the duplicates would be flagged as false in our training data when they were in fact true matches thereby creating bias. An example of a duplicate in our data is shown below:

| IA Book Title | IA Author | IA Publisher | IA Year | IA Link | Wikipedia Book Citation Link | Match |
|---|---|---|---|---|---|---|
| the growth of biological thought diversity evolution and inheritance | ernst mayr | Cambridge , Mass. : Belknap Press | 1982 | archive.org/details/growthofbiologic0000mayr | archive.org/details/growthofbiologic00mayr | No |
| the growth of biological thought diversity evolution and inheritance | ernst mayr | Cambridge , Mass. : Belknap Press | 1982 | archive.org/details/growthofbiologic00mayr | archive.org/details/growthofbiologic00mayr | Yes |

***Solution 1: "Golden Test Set"***

To solve this problem, we pulled the first 100 Wikipedia citations with Internet Archive Links and manually went through each result, marking all duplicates as matches in addition to

the link matches. We called this our “golden test set” of ~800 observations which is free from the duplicate bias. Because of this bias in the training data, we expected to see lower performance on model metrics in the training set and higher metrics in our golden test set. Applied to our example above, in the golden set, both of the values in the match column are set to “Yes.”

| Method | Train Precision | Train Recall | Test Precision | Test Recall |
|---|---|---|---|---|
| 100% Title Match | 17.1% | 87% | 39.5% | 87% |

After compiling our “Golden Test Set,” we see that the test precision is twice as high as the training precision after recoding duplicate records. However, 40% precision is still quite low compared to our goal.

***What happens if we incorporate other columns to help with matching such as author and publishing year?***

There are 750 possible citation information fields to include from a Wikipedia citation, but only around 10 exist in the Internet Archive. Of those existing matching fields, most of the data are missing most of the time. We decided to use 4 fields that exist in both and have a less missing values: Title, Author, Publisher, and Date. We hypothesized that these fields should be sufficient information to tell whether a book was a match or not. Though even among these fields, there was often missing data.

| Variable | Percent Missing |
|---|---|
| Title | 0% |
| Author | 34.7% |
| Publisher | 11.4% |
| Date | 32% |

While we could just drop this missing data and move on to modeling, book citations from wikipedia are often missing this information. So, our solution must be flexible enough to handle missing data in every feature except the title.

Continuing with our cutoff approach, if we exactly matched combinations of title, year, author, and publisher, our results are shown below:

| Method | Train Precision | Train Recall | Test Precision | Test Recall |
|---|---|---|---|---|
| 100% Title | 17.1% | 87% | 39.5% | 87% |
| 100% Title & Author | 18% | 52% | 40.8% | 56% |
| 100% Title & Publisher | 15% | 5% | 75% | 14.2% |
| 100% Title & Date | 48% | 55% | 100% | 27% |
| 100% All | 28.7% | 1.2% | 100% | 7% |

Breaking down the table above, we are starting to see definitive gains in our test precision as we are incorporating these other features using the cutoff approach, but we also are seeing significant losses in our test recall. Notably, the best cutoff approach tested is the “100% match of Title and Date” which sees 97.4% Test precision, but only 27% possible matches are included.

***Problem 2: Matching Differences***

The next question we had is in the best model “100% Title & Date” what do the 55% of matches look like that are missing? Digging through the data, the table below shows examples of what the differences are.

| Case # | IA Title | Wikipedia Title |
|---|---|---|
| 1 | the handbook of augmentative and alternative communication | handbook of augmentative and alternative communication |
| 2 | the nature of the chemical bond and the structure of molecules and crystals | the nature of the chemical bond |
| 3 | 100 voices  an oral history of ayn rand | 100 voices an oral history of ayn rand |

In the first case, there is a difference of the word “The” which exists in the IA title, but not in the wikipedia title. In the second case, the IA displays a descriptive subtitle while the Wikipedia Citation does not. In the third case, there is an extra space from punctuation that was removed. The same cleaning algorithm is applied to both titles, so a space existed before *and* after the colon in one and only before *or* after the colon in the other. In order to boost our recall rate we must find a solution that maintains the high precision rating while boosting our recall

***Solution 2: The Levenshtein Distance***

To solve our problem, we needed a more flexible matching method than only doing cleaning and matching exactly - this method could help solve all three cases above. After research, we found one way to do this was using a text distance metric called the “Levenshtein Distance” (Levenshtein, 1965) between each feature pair in both Wikipedia and Internet Archive which would return a number from 0 to 100 where 100 is a perfect match (Seatgeek, n.d.-b). The Levenshtein Distance is comparable to an edit distance when converting one string to another subtracted from 100 and capped at 0.

This method was appropriate for our use case because the algorithm is efficient and applicable to our problem where we often only expect small typo related differences in book title

matching. For visualization purposes, here are some examples from the dataset with various Levenshtein distances:

| String 1 | String 2 | Levenshtein Distance |
|---|---|---|
| anarchism: arguments for and against | anarchism arguments for and against | 99 |
| the principles of physical geology | principles of geology | 75 |
| encyclopedia of environmental analysis and remediation | environmental encyclopedia | 50 |
| the girl from the fiction department  a portrait of sonia orwell | orwell the life | 25 |

In addition to directly applying the Levenstien distance algorithm, there are other ways to apply it including using partial matching (where some extra characters are ignored in one string and full matched in the other) and sort matching (where the ordering of the words is ignored). Partial matching seems like it would be particularly useful to match case #2 in the title distance table. In the end, both methods were used in our feature engineering process which will be explained in further detail in that section.

Using a Cutoff of 95 Levenshtein distance, here are our new results:

| Method | Train Precision | Train Recall | Test Precision | Test Recall |
|---|---|---|---|---|
| 100% Title | 17.1% | 87% | 39.5% | 87% |
| 100% Title & Date | 48% | 55% | 100% | 27% |
| 95% Title & Date | 49.4% | 54.5% | 100% | 45% |
| 95% Title & 90% Author & 90% Publisher & Date | 82% | 3.3% | 100% | 7.1% |

***Problem 3: Optimizing Recall***

Using Levenshtein distance, we see some strong gains in our Test Recall metric from 27% to 45%, but hopefully we could do even better. As it stands we are matching less than half of possible matches, even if we have a high precision.

Additionally, in our current cut off model, year is missing 33% of the time - meaning 33% of the time, books with the same titles will be counted as matches even though they are different books. It also would be beneficial if we could further utilize publisher and author fields in our model as they are not yielding very helpful results at their current cutoffs.

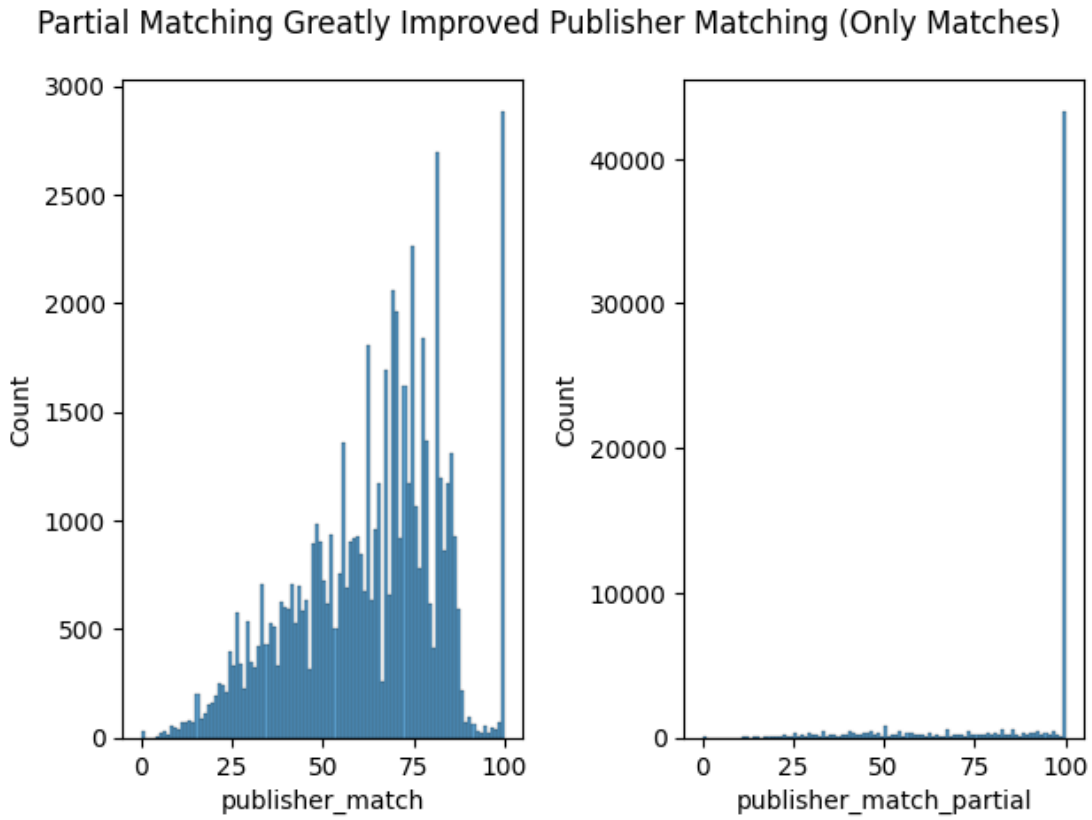
***Solution 3: Feature Engineering and Machine Learning***

To solve this problem, we hypothesized that it would be helpful to try to include the author and publisher features as we believed they contain key information in a book match. We found different matching solutions for both features that will be discussed below.

When looking at the publisher field, we noticed a strong pattern in the data:

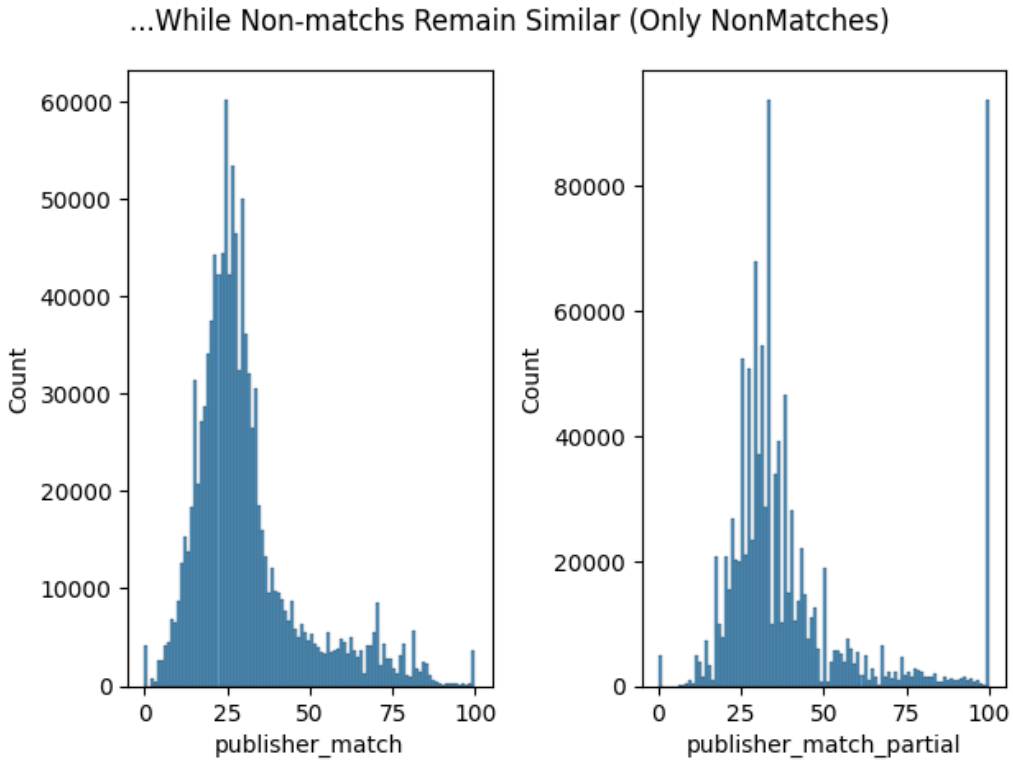
| IA Publisher | Wikipedia Publisher | Regular Match | Partial Match |
|---|---|---|---|
| Garden City, N.Y., Anchor Books | Anchor Books | 56 | 100 |
| Edinburgh, Scotland ; San Francisco : AK Press | AK Press | 30 | 100 |
| Oxford ; New York : Oxford University Press | Oxford University Press | 71 | 100 |
| Detroit : Gale/Thomson-Gale | Thompson Gale | 55 | 88 |

The Internet Archive data often includes more descriptive text than the Wikipedia publisher field. Because of this we decided to apply levenshtein distance partial matching to both publisher and book title. Where we fully match the less descriptive Wikipedia field and partially



match the more descriptive IA field. The graphs below provide a more full view of the effect of partial matching specifically on the publisher field.

We can see huge gains for the publisher feature when using partial matching. While for unmatched records, the distribution (shown below) remains largely the same. We see a spike in Non-match 100s, but this is likely due to duplicates and sometimes different books have the same publisher.




We created another feature based on author. Cleaning the author strings originally proved challenging because of non-consistent formats used in both sources. The author fields for Wikipedia are split in several fields whereas the Internet Archive has just one long string of author names in different formats. Sometimes just an initial would be placed for names and

sometimes not. Sometimes the last name would be first, and sometimes not. In order to combat this issue, we first concatenated wikipedia author fields and used sort matching for author where the order of the words is disregarded.

| IA Author | Wikipedia Author | Regular Match | Sort Match |
|---|---|---|---|
| r. malcolm errington | robert malcolm errington | 86 | 88 |
| rogers van fine txsatam collection. john spotswood antwerp | arnold joseph toynbee | 32 | 54 |
| gilbert m. smith | g m smith | 72 | 75 |
| alistair mcmillan iain mclean | iain mclean alistair mcmillan | 66 | 100 |

Using author sort matching did not see as substantial gains as publisher partial matching, but we did see a 3 point mean increase in Levenshtein distance among matches, and only a 0.5 increase among non-matches.

Note that the partial matching features added were added in addition to the full matched features. We found that including fully and partially matched features in combination created the most valuable feature set. Now with a richer feature set, we hypothesized that our problem could benefit from machine learning algorithms to help us match records as a hard-coded cutoff algorithm becomes more complicated and less feasible with the more features that are added. Additionally, a machine learning algorithm could perhaps learn trends that we have not currently seen from our exploratory data analysis. Before proceeding with modeling it was clear our training dataset was imbalanced with 4.8% matches and 95.2% non-matches (“Why Does Rebalancing Class-Unbalanced Data Improve AUC for Linear Discriminant Analysis?,” 2015).

To account for this difference, we implemented a custom downsampling algorithm which grouped queries and sampled 1 non-matched per 1 match. This way there would be representation of all queries possible. From this point, we followed typical machine learning workflows using a training and test set, using scaling procedures, cross validation, and hyperparameter tuning.

**Results**

We chose 4 different models to test suited for the needs of our classification problem. As an efficient, robust baseline, we chose the logistic regression model which is typically seen in binary classification problems. We chose the k - nearest neighbor algorithm which is a popular clustering based approach. We chose two CART approaches (random forest, gradient boosted tree model) which can handle missing data without imputation. After choosing these models, we selected several hyper-parameters which we tuned through cross-validation. A comprehensive overview of all models and hyper parameters used is included in the appendix.

The table below shows the results after 5 fold cross-validation of the training set.

| Model Name | Training Accuracy | Training Precision | Training Recall | Test Accuracy | Test Precision | Test Recall |
|---|---|---|---|---|---|---|
| GBT | 85.5% | 84.1% | 94.7% | 98.9% | 95.2% | 95.2% |
| Log Reg | 83.1% | 82.8% | 92.1% | 98.2% | 94.9% | 89.2% |
| RF | 92.3% | 91.0% | 97.2% | 97.4% | 94.5% | 82.1% |
| KNN | 87.9% | 88.0% | 93.4% | 96.9% | 85.8% | 86.9% |
| Baseline 95% Title + Year | 66.1% | 85.9% | 54.8% | 93.9% | 100% | 45.2% |

This table is sorted based on the best "Test Precision" with GBT as the top model 95% precision on the golden test set. These results are as expected with many of the models having higher Test metrics than Training metrics.

After selecting GBT as our favored model, we trained it on all of the data together (instead of ⅛ of it) and now have the table below as our final model metrics:

| Model Name | Training Accuracy | Training Precision | Training Recall | Test Accuracy | Test Precision | Test Recall |
|---|---|---|---|---|---|---|
| GBT | 85.5% | 84.3% | 94.3% | 99.0% | 97.5% | 94.0% |

Our final model shows great utility in scoring highly on Test Precision and only missing 6% of possible matches. With also the consideration that in the golden test set, since it was created manually, there are only 84 possible matches which means the final model only misclassified 1 observation as True which as actually False. This model could also be manipulated for higher precision as we can output a probability from GBT and change the threshold accordingly. Currently, the threshold is at 50% meaning those with probability over than 50% of being a match would be considered a match and vice versa. With adjustments to the threshold, the sponsor can decide exactly how they want to balance recall versus precision.

**Limitations and Future Work**

We have achieved strong results with 97.5% precision and 94% recall in our final model which is a substantial start to this problem (We also have a repository with an implementation of this algorithm here: https://github.com/acbass49/wiki2ia). One limitation of this approach is in interaction with the API. When we were matching the Wikipedia citations with the Internet Archive web links, we needed to query the Internet Archive API. We queried with a lowercase

and special character removed version of the book title. One fifth of the time, there were no results found even though we had a link from the Internet Archive. And, about one third of the time, results were found, but there was no matching link. This suggests one of two things: first, it could suggest that these links did exist, but the Internet Archive API needs a more flexible algorithm to return the result; second, it could suggest that these links that were not returned are broken links that no longer exist in the internet archive. Either way, it is something that should be further examined for the improvement of the usefulness of this algorithm and the general health of Internet Archive API.

Another option for the improvement of the algorithm is improving on the author matching. Of our primary text features (title, author, publisher), we have the weakest matching on Author. The table below shows summary statistics on the Levenshtien distance among only matches in the training data.

| Variable | Mean | Median | 25th perc. | 75th perc. |
|---|---|---|---|---|
| Title Levenshtein Distance | 94 | 99 | 99 | 100 |
| Publisher Partial Matching | 88 | 100 | 88 | 100 |
| Author Sort Matching | 86 | 100 | 75 | 100 |

As you can see in the table above, author matching has the lowest mean and lowest 25th percentile of Levenshtein distances among matches. There is opportunity for further work to be done in text standardization which was most complicated for Authors because of abbreviation, ordering, and spelling error differences.

Lastly, one option for future work is trying to incorporate other features. Perhaps there are other features that exist in both Wikipedia citations and the Internet Archive API that would improve key metrics. For example, we did not account for book versioning, but perhaps there is enough accurate data in both sources to accurately predict and find the correct book version as well as a match on the other fields.

**Conclusion**

Broken book citations on Wikipedia can be a significant problem, but our methodology offers a solution for fixing them. By using the Internet Archive and various text correlation and classification models, we have achieved strong results with 97.5% precision and 94% recall in our final model. This methodology can be used by Wikipedia editors to improve the accuracy and reliability of the platform's content. Additionally, it will also give many Wikipedia readers access to books cited in Wikipedia freely available on the Internet Archive website and database. By ensuring that all citations and references are accurate and up-to-date, we can help maintain Wikipedia's reputation as a reliable source of information for millions of people around the world.

**Appendix: Model and Grid Search Information**

| Model Name | Grid Search Information |
|---|---|
| Logistic Regression | Penalty: None, L1, and L2 |
| Random Forest | Trees: 10, 50, 100, 500<br>Max depth: None, 3, 5<br>Max features: None, 2, 3, 4, 5 |
| K Nearest Neighbors | N Neighbors: 3, 5, 10, 25 |
| Gradient Boosted Trees | Learning rate: 0.1, 0.2, 0.5<br>Max depth: None, 3, 5<br>L2 Regularization: 0, 0.01, 0.1 |

**Works Cited**

*Books : Free Texts : Free Download, Borrow and Streaming : Internet Archive*. (n.d.). https://archive.org/details/books

Levenshtein, V. (1965). Binary codes capable of correcting deletions, insertions and reversals. *Proceedings of the USSR Academy of Sciences*, *163*(4), 845–848. http://ci.nii.ac.jp/naid/10012867632

Seatgeek. (n.d.-b). *GitHub - seatgeek/thefuzz: Fuzzy String Matching in Python*. GitHub. https://github.com/seatgeek/thefuzz

Why Does Rebalancing Class-Unbalanced Data Improve AUC for Linear Discriminant Analysis? (2015). *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, *37*(5), 1109–1112.

Wikipedia contributors. (n.d.). *Template:Cite book - Wikipedia*. https://en.wikipedia.org/wiki/Template:Cite_book