Assessment 3: Webscrawler and NLP System

Overview

Autism Spectrum Disorder (ASD) is a neurodevelopmental condition characterised by deficiencies in social communication and rigid behaviour patterns. The Diagnostic and Statistical Manual of Mental Health Disorders, 5th Edition (DSM-5), categorises ASD into three levels of severity (*Autism speaks, 2024*). While the DSM-5 provides a standard framework for diagnosis and treatment, it also has limitations, such as oversimplifying human behaviour, stigmatising labels a potential misdiagnosis (*Fritscher. L, 2023*).

The DSM has undergone five revisions since its first publication in 1952, which infamously classified homosexuality as a "*sociopathic personality disturbance*" (*Miller. A, 2023*). Given this evolving history, it is crucial to use the DSM-5 as a flexible guide rather that a rigid classification tool. For parents, finding community and shared experiences is vital and literature serves as a valuable resource in this journey.

A wealth of information is available online, with websites like 'Goodreads' offering reviews of books that provide insights and support. Machine learning, particularly sentiment analysis using VADER, can help identify which books resonate most with individuals needs. The VADER score can then be used to classify the reviews and be used to build machine learning models. By analysing the review sentiment, we can guide readers towards the most beneficial literature for understanding autism.

Related works have occurred for this issue. A report was written to assess if ASD can be detected using machine learning models such as Support vector Machine (SVM) learning (*Farooq. M. S. et al, 2023*). The accuracy metrics from this report were quite high measuring at 81%. Another related report was written about the topic but this time used multiple algorithms to help diagnose diagnosis of ASD. (*Rasul. A. R. et al, 2024*). The machine learning algorithms performing the best were the SVM and logistic regression models with 100% accuracy scores.

WebCrawler

Domains

One of the considerations when performing webscraping, is to assess if a website contains Cloudflare. Cloudflare is used to increase the security and performance of websites (*Cloudflare, 2024*) by blocking the ability of web scraping to occur. When checking the website www.doesitusecloudflare.com, it is confirmed that goodreads does not, making it a good resource for this assessment. When performing webscraping, it is important to take into consideration the restrictions stated via the robots.txt file. Upon reviewing the robots.txt disallow directives, URLs that will be scraped in the '/book/show/' area. This is not stated as 'disallow' and so the urls used for scraping will be fine to use for this assessment. When scraping the website, we will be careful not to scrape any information that can be linked to anyone by ensuring this process is performed anonymised. Only the review, date and rating will be extracted.

The Goodreads website is owned by amazon, and they hold copyright over the content on the reviews (*Goodreads, 2023*). The legal framework outlined states that claims of copyright infringement could be subject to a takedown or legal action through the Digital Millennium Copyright Act (DMCA). As the information that will be extracted from the website will not be published and is for research purposes only, this assessment will not be in breach of this stipulation.

One of the issues with the data is that most of the reviews are positive. This will likely cause unbalanced classes for my dataset and will need to be handled when creating the models. One good thing about the 'goodreads' website is that the reviews are in order by how many likes a review has. This in effect has included a higher proportion of negative reviews in the top 30 reviews. As when a person agrees with a negative review, then they can 'like' the review increasing the weight of the importance, meaning that the review is unlikely to be concealed by the majority of positive reviews.

WebCrawler workflow

For this assessment, the technological component to be used to scrape websites will be '*Beautiful Soup'* which is a python library used to extract information from HTML and XML files (*Beautiful Soup, N.D.)*. Another technological component used is the '*requests'* library, specifically the .get method which is able to retrieve HTML content from the web. Other technological components which could have been used are '*Scrapy'*, which crawls webpages and has an '*Autothrottling'* function to adjust the crawling speeds as to not exhaust the target server's resources. However, this form of scrapping websites has a steep learning curve (*Pavlovskytė, 2023*). Selenium is a '*browser automated tool'* that is useful for webscraping. However, compared to '*Beautiful Soup'*, it is quite complex to setup and use (*Zenrows, 2023*). As the data that will be extracted is quite a simple process, Beautiful Soup library will be used for this assessment.

The URLs chosen to be scrapped are the top 20 books listed when searching the keyword '*autism'* on the '*Goodreads'* website. For each URL, there are 30 reviews present. The target data which we are looking to extract, can be located by right clicking and selecting the '*inspect'* option. From this, we can click the '*inspect element'* button and then hover over the section we are looking to extract. From doing this, we can see that the information can be found in the <section class="ReviewCard__content"></section> (**Figure 1**). From this, we are able to extract the date of the review, the rating given for the book (out of 5 stars) and the review.

Figure 1: The element which contains the targeted data

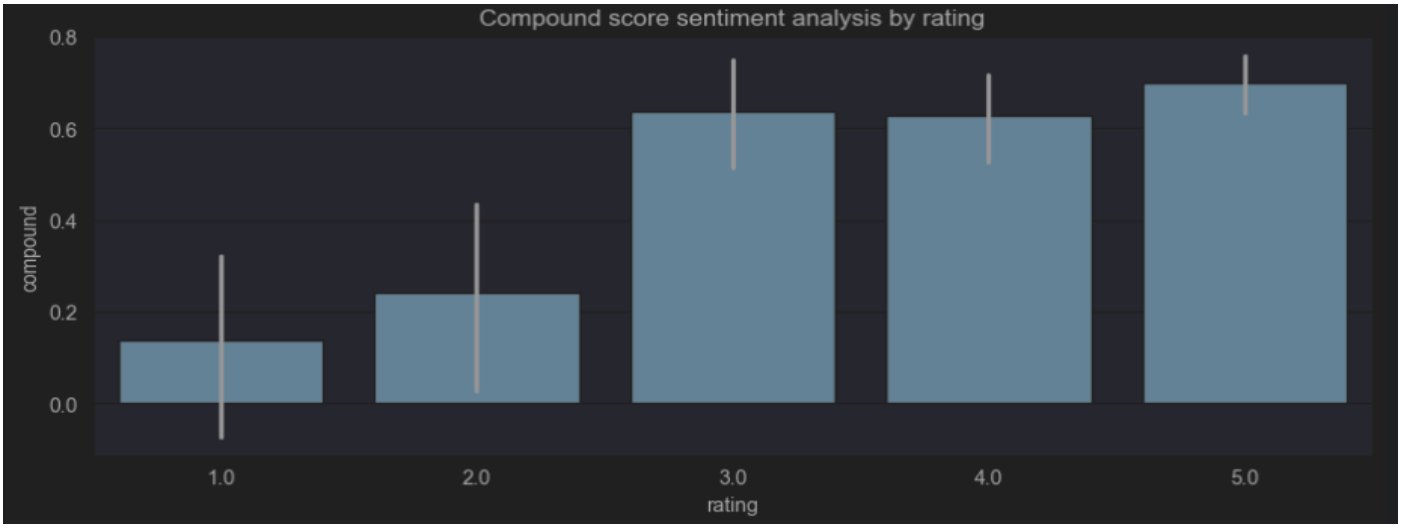
The domain is quite complex and there are many layers to this HTML page. It is quite difficult to extract the star rating. The information that was required to be extracted was in the '*aria-label'* location. When using the soup.find('div','aria-label') call, no results would appear. To extract the text '*Rating 5 out of 5'* a for loop needed to be written and the '*aria-class'* needed to be subsetted from the parent tag.

Data Wrangling

Using VADER for sentiment analysis the SentimentIntensityAnalyzer assigns negative, neutral, positive and compound scores to the reviews. It is important to apply this analysis on the unprocessed text data, as the use of capitalisation and punctuation affects the scores. With these scores, we can then compare the results with the star rating provided by the users. The corpus for this NLP task consists of all the book reviews. Some reviews measured a neutral score of 1.000 and 0.000 for positive and negative. When reviewing the text, the reviews were in another language and so were removed. Reviews without a star rating were removed since they are necessary for later comparison in machine learning. No duplicates were found in the data.

After obtaining and augmenting the sentiment scores to the data, the review text is cleaning by retaining only letters (removing punctuation, numbers etc). All text in converted to lower case and lemmatisation will be performed to group word variants together. Any stopwords will be removed as they add little value in text similarity analysis. The sentiment scores are then used to classify reviews as either '*negative'* or '*positive'*. Reviews with a compound score above 0.5 are labelled as positive ('1'). While tode blelow 0.5 are labelled as negative ('0'). This binary classification serves as the label for the machine learning algorithms, based on the clear separation around the 0.5 score between ratings 2 and 3 (**Graph 1)**.

Graph 1: Compound Sentiment score distribution for each star rating for all books



TF-IDF

In order to perform machine learning, first the text data needs to be converted into numerical data. Term frequency (TF) is a measure of how many times a word appears within the text divided by the total number of words. Inverse Document Frequency (IDF) is a measure of how rare a word is in a collection of documents (corpus). By multiplying these together, we can quantify the importance of a word in a document related to the number of times the word appears in a corpus (*Jain, 2024*). The higher a TF-IDF score for a term, then the more relevant that term is related to the rest of the terms. As a term approaches 0 for score, it becomes less relevant.

The classes in this dataset are not weighted evenly (**Graph 2)**. There are a significantly larger number of positive reviews than negative reviews. The first step to overcome this issue, is to use stratified sampling when splitting the dataset into training and test data (**Figure 2**). The benefits of stratified sampling ensure that the '*negative review'* class will be represented proportionately *(Murphy. C. B, 2021)*.

Figure 2: splitting the data into training and test sets using stratified sampling.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, stratify=y, random_state = 110)
Executed at 2024.08.15 14:01:25 in 32ms
```

Graph 2: Stratified sampling of the classes in the training data

We can see that there is a clear imbalance in the classes after performing stratified sampling (**Graph 2)**. This imbalance will cause issues when creating the classification models due to bias towards the majority class. This is because the model will attempt to reduce the error rate and so will avoid the minority class (*Yenigün. O, 2023*). To address the imbalance, a technique called Synthetic Minority Oversampling Technique (SMOTE) will be used. SMOTE is an oversampling method that creates new synthetic observations from samples in the minority class. These new observations are made randomly by the use of k- nearest neighbours (*Turing, N.D.)*.

Graph 3: Distribution of the training data after using SMOTE

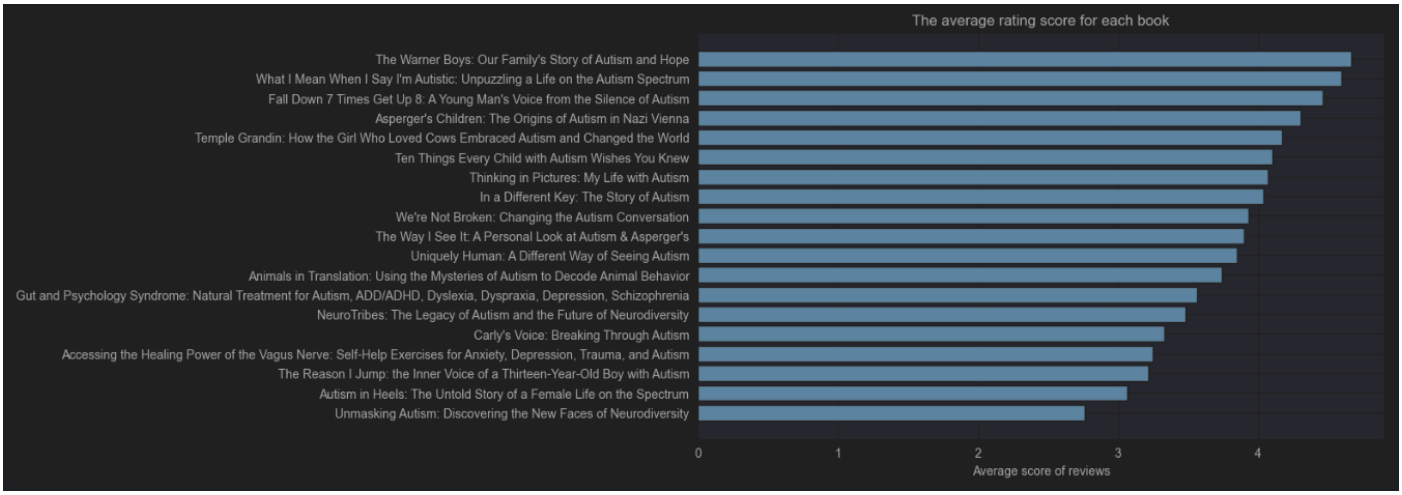
From using the SMOTE technique, the classes are now balanced and are appropriate for fitting a machine learning model (**Graph 3)**. An important consideration when using synthetic data, is to only use it for the training data. For the test data, it would be incorrect to generate synthetic data as this would obscure the true nature of our data.

Data Summarisation

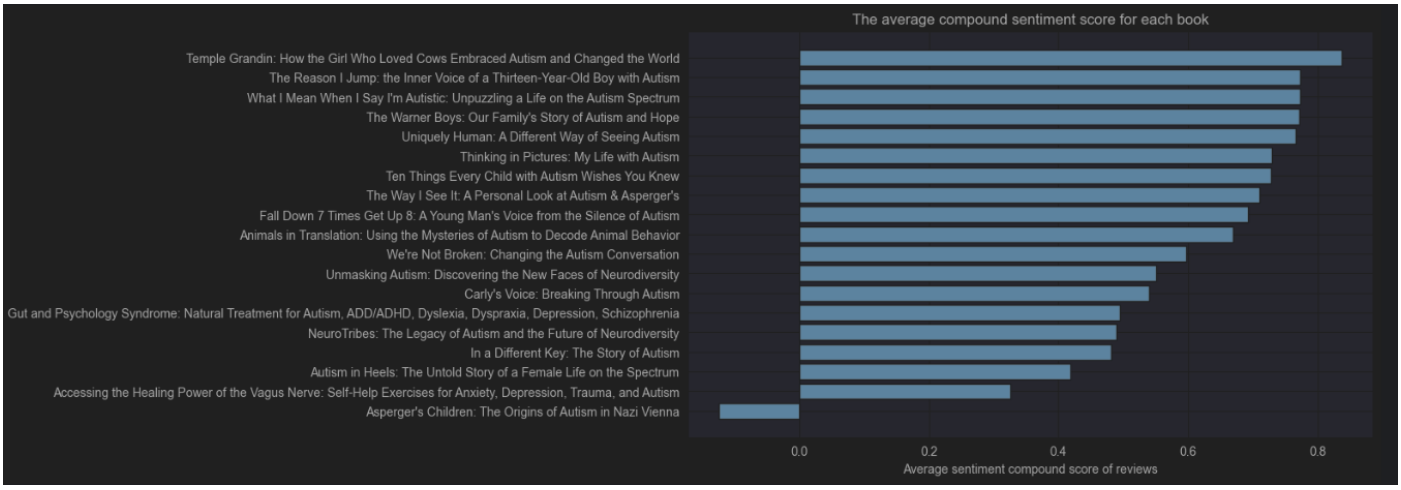
Graph 4: The number of reviews for each book.

From visualising the number of reviews for each book, there are reviews that were removed due to either the language not being in English or a review score was not used. The removal of these reviews has unbalanced the classes slightly (**Graph 4**). These reviews could have been valuable and would have provided context to opinions from a varied perspective adding more value to the dataset.

Graph 5: The average rating score for each book

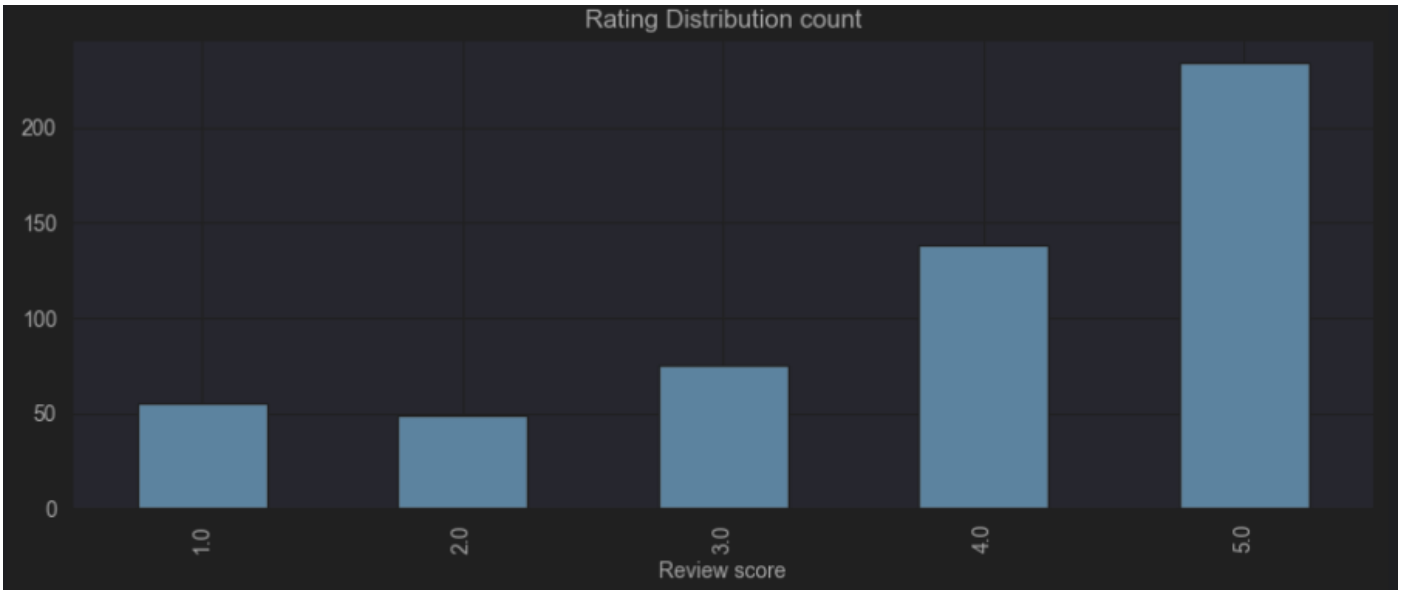


Graph 6: The average sentiment compound score for each book.

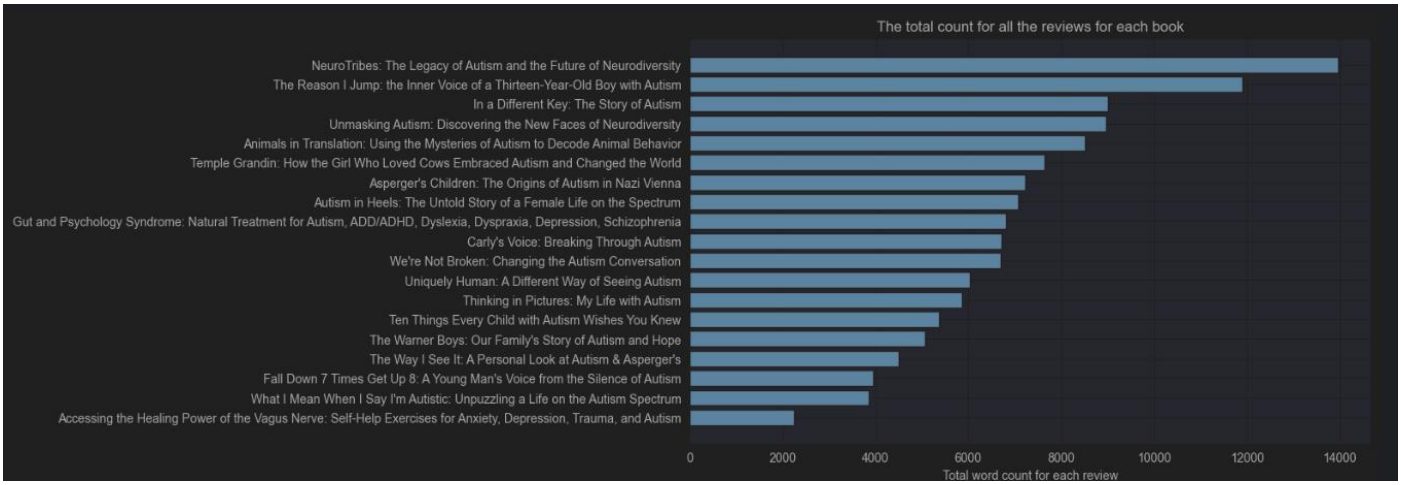


The average compound sentiment score and the rating score has the books listed in a similar order. This indicates that the sentiment score is working well and accurately reflecting the score of the review (**Graph 5 and 6**). There is one anomaly however, the book '*Asperger's children: The origins of Autism in Nazi Vienna'* scored quite high for rating. But for the Sentiment score, the reading is the only negative score for all the books. This is an area in which the sentiment analyser could underperform in regard to subject matters that have a strong positive message but is layered in a dark context.

Graph 7: The number of reviews for each rating score

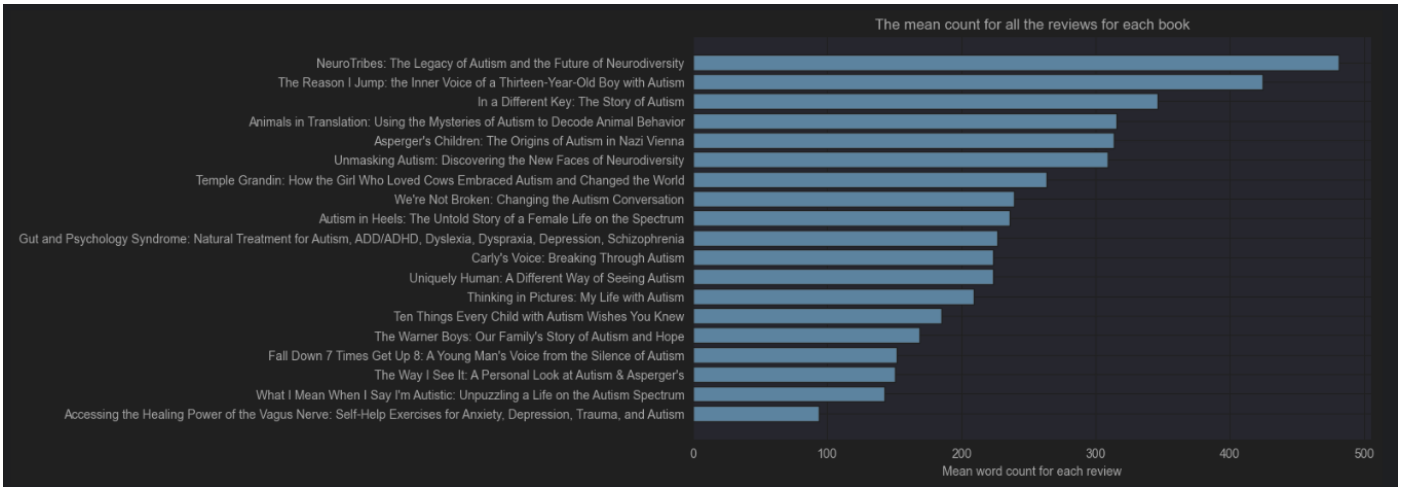


Graph 8: The total number of words used for all the reviews for each book.

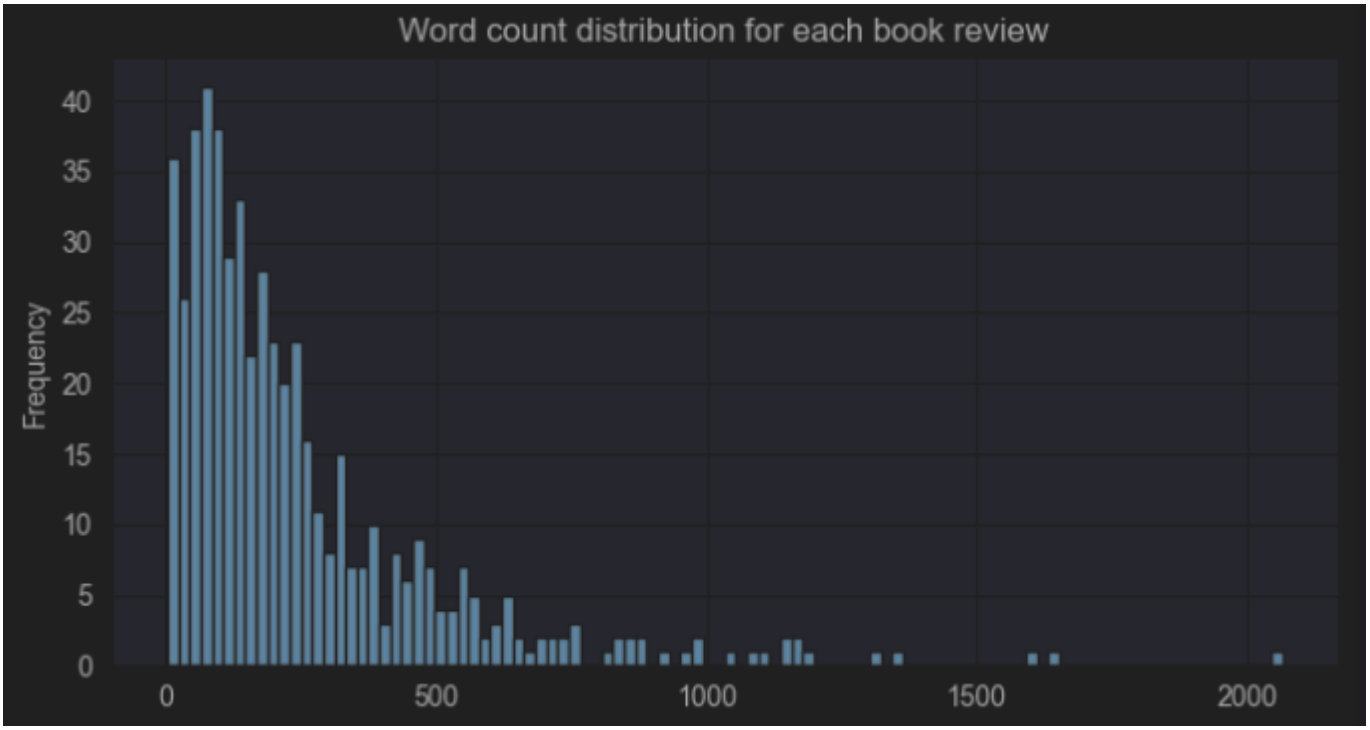


There is a spread of review scores throughout the corpus (**Graph 7**). The majority or the scores are positive. This imbalance of classes will likely affect the models which will be created later in the assessment. The total number and the mean of words used for each review will also likely affect the model performances as a review is most likely to have the correct context extracted when the lexical density is increased (**Graph 8 and 9**).

Graph 9: The mean number of words used for all the reviews for each book.



Graph 10: The distribution of the number of words used for all of the reviews in the corpus



The frequency of the corpus is positively skewed meaning that most of the word counts of the reviews are towards the lower end of the data. Most of the review lengths tend to be around the 100 word length which is quite short and may be difficult to build an effective model once we have removed all the stop words and performed lemmatisation (**Graph 10**).

Zipf's law is an empirical observation that states that the frequency of a word in inversely proportional to its rank in a frequency table (*Yashyad, 2023)*. By taking a count of the words in the whole corpus by using the FreqDist function (**Figure 3**), we can view the most important words in the corpus after the data has been cleaned and the stopwords have been removed (**Graph 11**).

Graph 11: The frequency of words in the cleaned corpus (reviews)

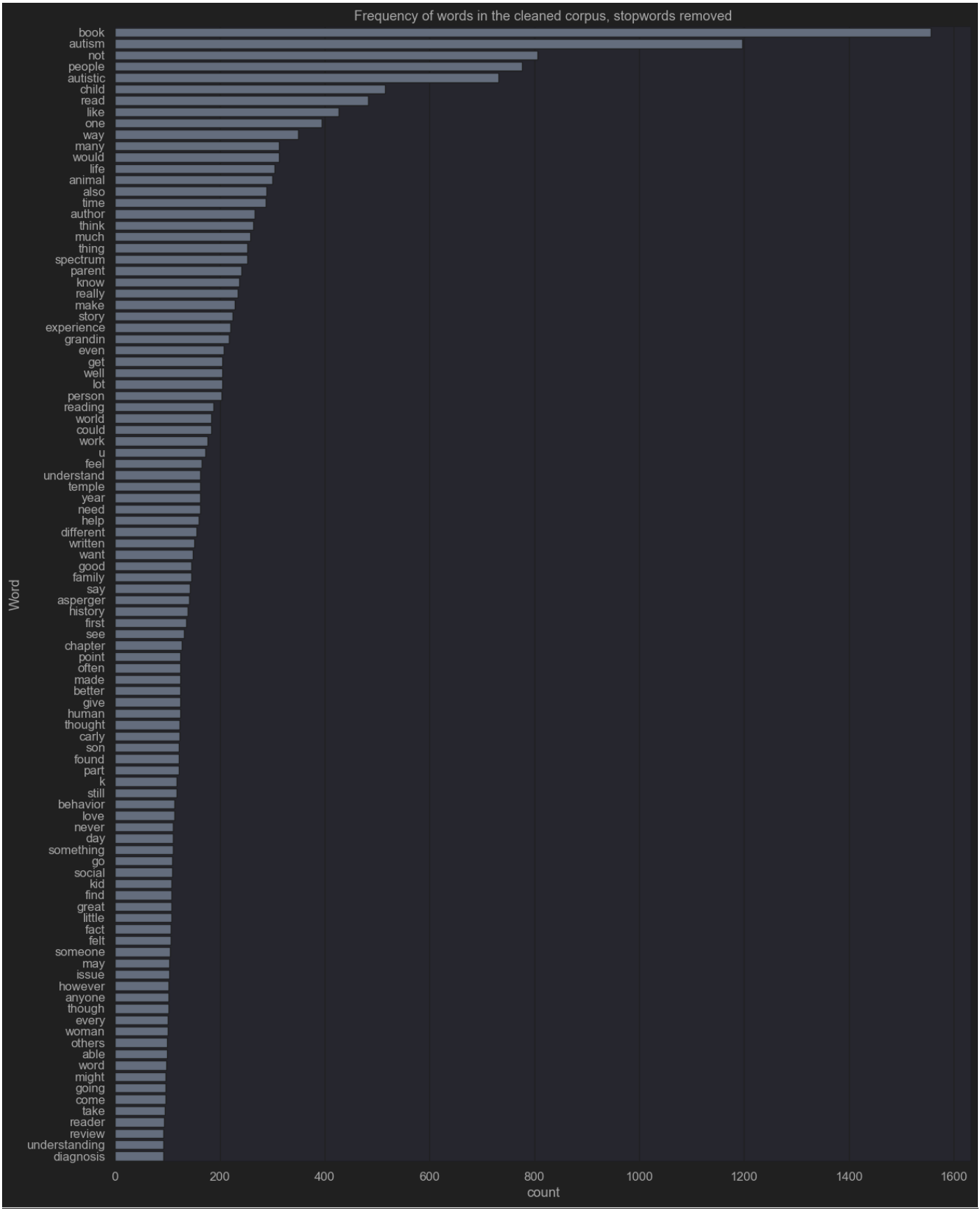


Figure 3: The function used to visualise the 100 highest occurring terms in the corpus

```
def freq_words(x, terms = 30): # show the most prevalent words in the corpus
  all_words = ' '.join([text for text in x])
  all_words = all_words.split()
  fdist = nltk.FreqDist(all_words)
  words_df = pd.DataFrame({'word':list(fdist.keys()), 'count':list(fdist.values())})

  # selecting top 20 most frequent words
  d = words_df.nlargest(columns="count", n = terms)

  # visualize words and frequencies
  plt.figure(figsize=(15,20))
  ax = sns.barplot(data=d, x= "count", y = "word")
  ax.set(ylabel = 'Word')
  plt.title('Frequency of words in the cleaned corpus, stopwords removed')
  # plt.tight_layout()
  plt.show()
```

Table 1: Descriptive Statistics of the Corpus and sentiment scores

	Mean	Standard deviation	Minimum	Maximum	Median
Word Count	247.86	260.60	2.00	2065.00	170.00
Compound Sentiment score	0.60	0.60	-1.00	1.00	0.90
Negative Sentiment score	0.06	0.06	0.18	0.77	0.05
Positive Sentiment score	0.14	0.09	0.00	0.83	0.18

The word count is quite varied in the corpus as can be seen from the standard deviation score which is higher that the mean. The maximum word count is nearly a factor of 10 higher than the mean which is skewing the data (**Table 1**). It was considered to remove this data as it could be causing bias in the dataset. However, the longest reviews tended to be the most meaningful and contained a lot of insight towards the book that I wanted to capture in the models that will be built. The Mean is higher than the median which is another indication that the data is positively skewed for the word length. The sentiment score is conversely negatively skewed as most of the reviews are positive in this dataset.

Machine Learning

Machine Learning Structure

Table 2: Results from the various metrics from the models used

Model	Review_class	F1_score	Precision	Recall	Accuracy	AUC
XGBClassifier	Positive	0.84	0.84	0.84	0.75	0.75
	Negative	0.50	0.50	0.50		
GussianNB	Positive	0.83	0.77	0.91	0.73	0.53
	Negative	0.22	0.36	0.15		
RandomForrest	Positive	0.83	0.82	0.85	0.75	0.74
	Negative	0.45	0.48	0.42		
SVM	Positive	0.87	0.78	1.00	0.78	0.49
	Negative	0.12	1.00	0.21		
Logistic Regression	Positive	0.84	0.83	0.85	0.75	0.75
	Negative	0.48	0.50	0.46		

To be able to determine which would be the most appropriate classifier to focus on, various algorithms were tested to ascertain the best performing model. Once this is has been assessed, we can then tune the hyperparameters to maximise the performance. From viewing the metrics (**Table 2**), we can see that the performance does tend to drop when determining the Negative class particularly for the GaussianNB and SVM models for the F1 score and Recall. The best performing models were the XGBClassifier and Logistic regression models. All of the metrics (F1 score, Precision, Recall, Accuracy and AUC) score quite well for the positive class. The negative class is performing about as well as a random prediction. Hopefully by tuning the hyperparameters, the results will improve for this classification. The XGBoost classifier was chosen as the algorithm to tune the hyperparamters as it is able to handle imbalanced datasets (*Filho, 2023)*.

For tuning the hyperparameters, the GridSearchCV library will be used. This package allows us to return the parameter setting for returning the highest scoring metric of our choice (*Shah. R, 2024)*. The hyperparameters that we will look to tune for the XGBoost model are the 'n_*estimators'*, 'max_*depth*', '*learning_rate'* and '*gamma'* values. The '*n_estimators'* is the number of trees used in the model. The '*max_depth*' is the maximum tree depth for base learners. The 'learning rate' is the contribution of each new tree to the final prediction. The 'gamma' is the is the minimum loss reduction required to make a further partition on a leaf node of the tree (*DMLC XGBoost, 2022)*.

Figure 2: Code used to hypertune the parameters for the XGBoost classifier

```
gs = GridSearchCV(estimator = xgb_model,
                  param_grid = search_space,
                  scoring = 'accuracy',
                  refit = 'accuracy',
                  cv = 5,
                  verbose = 4)
```

From tuning the hyperparameters, we were able to assess that the most optimal settings were as follows:

Figure 3: The best parameters to use for the XGBoost model

```
print(gs.best_params_)
Executed at 2024.08.16 12:31:32 in 30ms

{'gamma': 0.01, 'learning_rate': 0.01, 'max_depth': 9, 'n_estimators': 500}
```

When applying these hyperparameters, the F1 score increases to 0.88. The F1 metric was used as this considers both the '*precision*' and '*recall*' metrics (*Kundu, 2022)* which should help improve the model. (**Figure 2**). To ensure that the hyperparameter tunning was as accurate as it could be, the tuning process was conducted using cross validation which uses different blocks of sample, ensuring that the model is based on the full dataset and not just a part of it, making the model more representative.

Evaluation

After tuning the hyperparameters using the XGBoost Classifier, the metric scores did not improve noticeably than the default model. Many various permutations were attempted by varying the settings as outlined using the XGBoost documentation (*DMLC XGBoost, 2022*) but no distinct improvements could be made. The AUC and recall score did improve for the positive class, but the negative class performed worse **(Graph 12)**. The XGB Classifier does diverge from the dotted line demonstrating that the model is able to score higher true positives than compared to false positives.

Graph 12: The ROC-AUC fit using the XGBoost Classifier

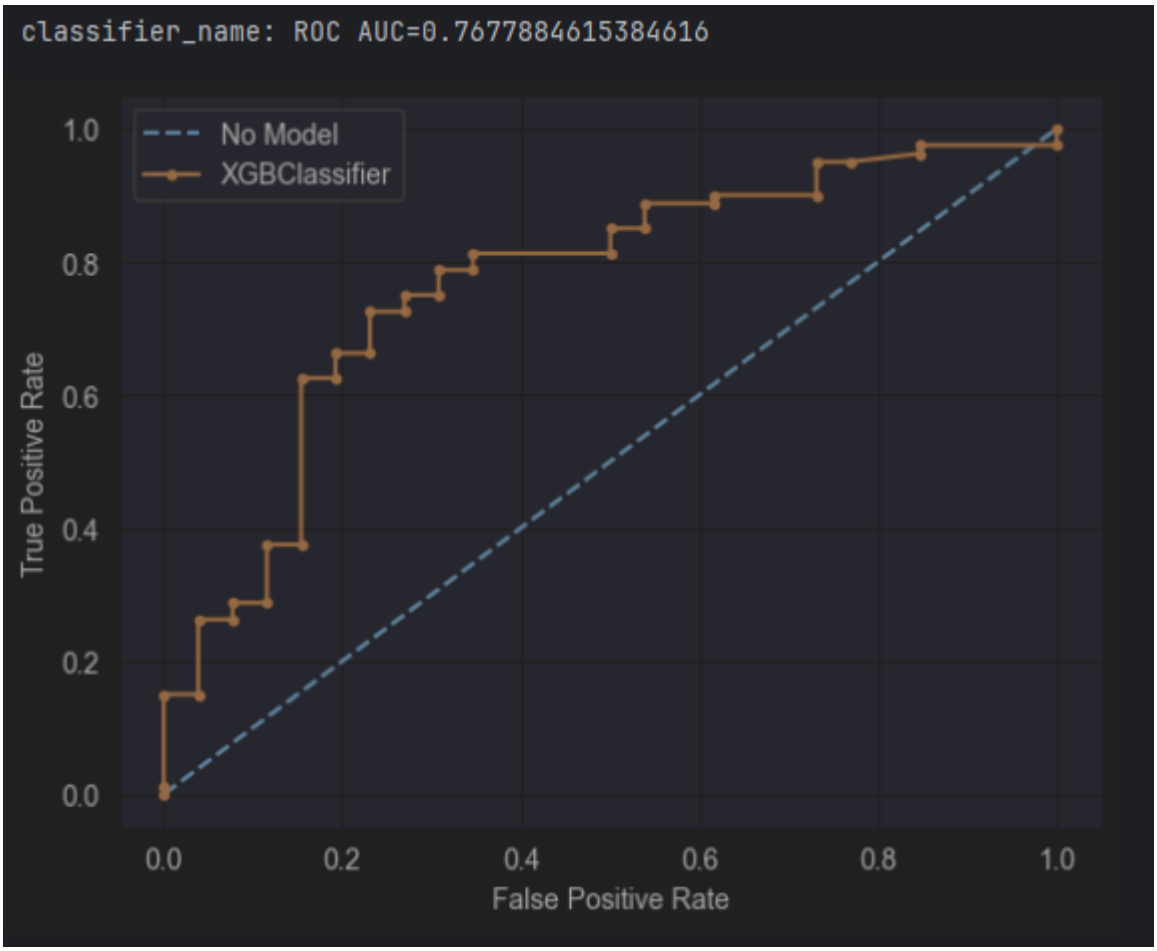


Table 3: The metric results for the tuned XGBoost model:

Model	Review_class	F1_score	Precision	Recall	Accuracy	AUC
XGBClassifier	Positive	0.84	0.83	0.85	0.75	0.77
	Negative	0.48	0.50	0.46		

When determining the positive class reviews, the model is effective in classifying the review but unfortunately the negative class performed poorly being able to predict only as well as random when viewing all the metrics (**Table 3)**. This could be due to bias that is inherently present in the data due to the imbalanced classes. One way to use XGBoost to handle imbalanced classes is to use the '*scale_pos_weight'* argument. This operates by applying a higher penalty to misclassification of the minority class. Setting this will focus more on the negative class (*Filho, 2023)*.

Table 3: The metric results for the XGBoost model when using 'scale_pos_weight'

Model	Review_class	F1_score	Precision	Recall	Accuracy	AUC
XGBClassifier	Positive	0.82	0.85	0.79	0.74	0.76
	Negative	0.52	0.47	0.58		

After applying the '*scale_pos_weight'* argument, the Recall score for the negative class did increase meaning that there were fewer false negatives predicted by the model (**Table 4)**. However, the rest of the metrics did not perform much better than without this argument. Most of the metrics performed slightly worse including the Accuracy and AUC scores.

Conclusion and Further work

It has been difficult to manage the class imbalances within the dataset. Even when performing stratified sampling and SMOTE to balance the training data, the models that were built were not the most effective especially for the negative class. To improve this issue in future work, it would be helpful to enhance the data scraping technique by targeting specific reviews so that an equal amount of positive and negative reviews are taken from the website rather than relying on synthetic data. Another reason as to why the classifier is under performing could be due to the sample sizes being too small. In future work, the amount of data extracted could be extended to a larger sample size to aid the performance of the final model.

References

Autism speaks. (2024). *ASD levels of severity.* https://www.autismspeaks.org/levels-of-autism#:~:text=The%20DSM%2D5%20introduced%20three,with%20permission%20from%20the%20APA.

Beautiful Soup (N.D.). *Beautiful Soup Documentation. https://beautiful-soup-4.readthedocs.io/en/latest/*

Cloudflare. (2024). *So what is cloudflare? https://www.cloudflare.com/en-gb/learning/what-is-cloudflare/*

DMLC XGBoost. (2022). *Python API Reference. https://xgboost.readthedocs.io/en/stable/python/python_api.html*

Farooq. M. S, Tehseen. R, Sabir. M, Atal. Z. (2023). Detection of autism spectrum disorder (ASD) in children and adults using machine learning. *Scientific reports*. 13, 9605 (2023). https://doi.org/10.1038/s41598-023-35910-1.

Filho. M. (2023). *How to handle Imbalanced Data In XGBoost Using scale_pos_weight In Python*. Forecastegy. https://forecastegy.com/posts/xgboost-imbalanced-data-scale_pos_weight-python/

Fritscher. L (2023). *Advantage and Disadvantages of the Diagnostic Statistical Manual*. https://www.verywellmind.com/dsm-friend-or-foe-2671930

goodreads. (2023). *Terms of Use*. https://www.goodreads.com/about/terms

Jain. A. (2024). *TF-IDF in NLP (Term Frequency Inverse Document Frequency)*. Medium. https://medium.com/@abhishekjainindore24/tf-idf-in-nlp-term-frequency-inverse-document-frequency-e05b65932f1d

Kunku. R. (2022). *F1 Score in Machine Learning: Intro & Calculation. V7. https://www.v7labs.com/blog/f1-score-guide*

Miller. A. (2023). *2023 marks 50-year anniversary of removal of homosexuality from disorders handbook'.* The Hawks Newspaper. https://sjuhawknews.com/30785/features/removal-of-homosexuality-from-disorders-handbook-50-years/

Murphy. C. B. (2021). *Stratified Random Sampling: Advantages and Disadvantages.* Investopedia. https://www.investopedia.com/ask/answers/041615/what-are-advantages-and-disadvantages-stratified-random-sampling.asp

Pavlovskytė. E. (2023). *Scrapy vs. Beautiful Soup: A Comparison of Web Scraping Tools. https://oxylabs.io/blog/scrapy-vs-beautifulsoup*

PsychDB. (2024). *History of the DSM*. https://www.psychdb.com/teaching/1-history-of-dsm

Rasul. R. A, Saha. P, Bala. D, Rakib Ul Karim. S.M, Abdullah. I, Saha. B. (2024). An evaluation of

machine learning approaches for early diagnosis of autism spectrum disorder. *Science Direct*. 5, 100293. https://doi.org/10.1016/j.health.2023.100293.

Shah. R. (2024). *Tune Hyperparamters with GridSearchCV.* AnalyticsVidhya. https://www.analyticsvidhya.com/blog/2021/06/tune-hyperparameters-with-gridsearchcv/

Simha. A. (2021). *Understanding TF-IDF for Machine Learning*. CaptialOne. https://www.capitalone.com/tech/machine-learning/understanding-tf-idf/

Turing. (N.D.) *How to use SMOTE for an imbalanced Dataset*. https://www.turing.com/kb/smote-for-an-imbalanced-dataset

Yashyad. (2023). *Zipf's law.* Medium. https://medium.com/@yashyad2812/zipfs-law-831869356ca7

Yenigün. O. (2023). *Handling Class imbalance in Machine Learning*. Medium. https://python.plainenglish.io/handling-class-imbalance-in-machine-learning-cb1473e825ce

Zenrows. (2023). Zenrows. *Selenium vs. BeautifulSoup in 2024: Which is Better? https://www.zenrows.com/blog/selenium-vs-beautifulsoup#ease-of-use*