Predicting Text Difficulty[1]

Buser, Andre (busera; AB)
Adafinoaiei, Victor (vadafino; VA)

Abstract. This project aimed to classify sentences from articles in the "Simple English Wikipedia" to determine whether they require simplification to enhance their accessibility and comprehensibility for audiences with low reading proficiency, including students, children, and non-native English speakers. The project employed supervised and unsupervised learning techniques to extract and create features for sentence classification. The motivation was to leverage the insights and models generated by the project to improve the readability and comprehensibility of our internal investigation and audit reports. By identifying the factors that make a sentence challenging to read, we can develop targeted strategies for enhancing the clarity and accessibility of our written materials, thereby improving the effectiveness of our communication.

1. Introduction

Our objective was to create a model that categorizes sentences from "Simple English Wikipedia" articles as either "simple" (0) or "complex" (1) to identify texts that would benefit from simplification to improve readability and accessibility for non-native English speakers, children, and students. We intended to use the insights and model produced by the project to enhance the comprehensibility of our internal investigation and audit reports. We applied various techniques to achieve our goals, including constructing features, analyzing feature importance, and performing failure, sensitivity, and learning curve analysis. However, creating a strategy for simplifying sentences was outside this project's scope.

This project utilized a combination of supervised and unsupervised machine learning methods. Specifically, we used supervised learning to train our model to distinguish between "complex" and "simple" sentences and employed unsupervised machine learning techniques to understand the constructed features better. For instance, we utilized principal component analysis (PCA) to determine the most significant features and their correlation to each other and used KMeans clustering to facilitate the failure analysis. By harnessing the strengths of both supervised and unsupervised learning, we gained more insights regarding the data and interpretability of our outcomes.

The main findings of our project were as follows: **(1)** The Random Forest algorithm was the best-performing model for sentence classification with an optimized accuracy score of 0.7544 using 125 of the 151 constructed features. **(2)** However, our failure analysis revealed challenges with the quality of the primary source dataset because, in addition to the usual text pre-processing challenges, we found that the dataset contained samples in over 89 languages, contrary to our initial expectation of only English language samples. Furthermore, the dataset contained many non-sentence samples, including image captions, formulas, file names, and single words. **(3)** Despite our efforts to address these issues, such as removing non-sentence samples and filtering out non-English samples, we could not improve the model's accuracy. This suggests that the dataset also had labeling issues, which was confirmed by our failure analysis. It could also explain why none of the engineered features could clearly separate the sentences into simple or complex.

Overall, while the Random Forest algorithm was the best performing, the quality of the dataset and labeling issues presented significant challenges that need to be addressed in future work.

2. Related Work

Our project team reviewed several relevant papers, including "**Supervised and Unsupervised Neural Approaches to Text Readability**" by Matej M., Senja P., and Marko R.S. The research paper presented novel neural supervised and unsupervised approaches for determining the readability of documents, which are robust and transferable across languages, and allow adaptation to specific readability tasks and datasets. In addition, the study offered a comprehensive analysis of different neural approaches to

[1] SIADS 696 Milestone II project report

readability classification, comparing their performance to current state-of-the-art classification approaches and proposing possibilities for improvements.

We also looked at "**Text Difficulty Classification by Combining Machine Learning and Language Features**" by Han Ding, Qiyu Zhong, Shaohong Zhang, and Liu Yang, which proposed an improved method for constructing a dataset for English language education field readability studies. Their study showed that combining linguistic difficulty features and in-depth features significantly improved text difficulty prediction compared to using individual features.

We also reviewed "**Applying Natural Language Processing, and Hierarchical Machine Learning Approaches to Text Difficulty Classification**" by Renu Balyan, Kathryn S. McCarthy, and Danielle S. McNamara. This study explored advanced artificial intelligence methods, including natural language processing (NLP) and hierarchical machine learning, to predict the difficulty level of practice texts used in a reading comprehension intelligent tutoring system. The study found that incorporating NLP indices and machine learning algorithms improved accuracy by more than 10% compared to traditional readability metrics and that hierarchical machine learning outperformed non-hierarchical approaches for certain text sets.

Our project combined traditional features, like readability scores and text statistics, with more sophisticated approaches, such as NLP POS tagging and word-to-vector features. Notably, our problem formulation focuses on single sentences instead of entire documents, which is the most significant difference from the other papers we reviewed.

3. Methodology

3.1 Data Sources

The following datasets were download from the Kaggle competition page[2].

WikiLarge_Train.csv		**WikiLarge_Test.csv**
The WikiLarge_Train.csv file contains labeled examples for training a machine learning model, labeled with either 0 or 1.		The WikiLarge_Test.csv file contains unlabeled examples that will be used to evaluate the performance of the trained model.
Access method	Last **download** on **2023-01-20**	Last **download** on **2023-01-20**
No. of records	416,768	119,092
No. of attributes	2	3

Concreteness_ratings_Brysbaert_et_al_BRM.txt		**AoA_51715_words.csv**
The Concreteness_ratings_Brysbaert_et_al_BRM.txt file contains concreteness ratings for 40 thousand English lemma words that were gathered via Amazon Mechanical Turk.		The AoA_51715_words.csv file contains "Age of Acquisition" (AoA) estimates for about 51k English words. AoA refers to the approximate age (in years) when a word was learned.
Access method	Last **download** on **2023-01-20**	Last **download** on **2023-01-20**
No. of records	39,954	51,715
No. of attributes	9	16

[2] https://www.kaggle.com/competitions/umich-siads-696-f22-predicting-text-difficulty/data

3.2 Data Cleaning and Data Understanding

Initially, we emphasized applying the standard text pre-processing steps to remove punctuation and special characters. However, upon sampling and spot-checking, we found that the datasets contained not only English sentences but also instances of other languages. To understand the extent of the language problem, we used the fastText library to classify the language of all sentences. But it also needs to be noted that detecting the language based on a single sentence is less accurate than on a complete paragraph. After some minor manual adjustments, we saw 89 different languages in the WikiLarge_Train dataset, with the top five languages as listed here (JN01.01[3]):

(1) English: 409,244; **(2)** French: 2,491; **(3)** German: 1,274; **(4)** Spanish: 736; **(5)** Italian: 424.

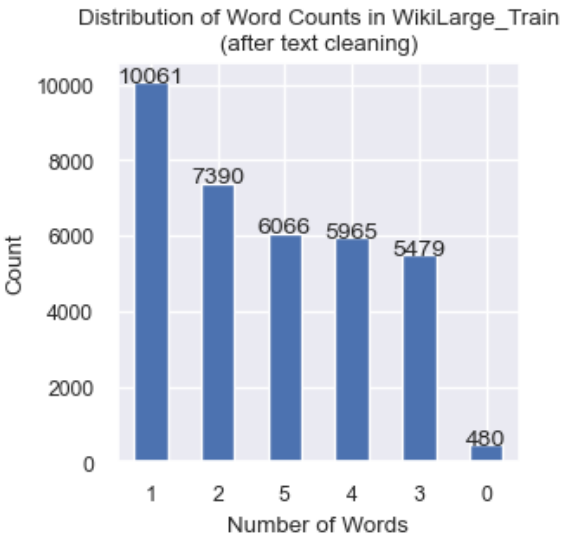


Fig. 1

Further investigation into short/incomplete sentences revealed the following results as shown in figure . To address the issue of different languages and incomplete sentences, we attempted a test run with only English sentences longer than five words. However, these clean-up activities did not result in improved accuracy scores for our base models pointing to a deeper issue: the potential **mislabeling** of instances.

3.3 Feature Engineering

We needed to supplement the text data from the WikiLarge dataset with additional features to create a robust binary classifier. Throughout the project, we developed 151 unique features through a combination of external resources and intuition and established best practices for text processing and classification (JN01.01 to JN01.06). The table below offers an overview of the six feature groups we generated and implemented:

Group	No. of Features	Description	Examples	Notebook Ref.
Text statistics (stats_)	33 (34-1)[4]	Features based on text before and after cleaning activities.	Count characters and words	01.01
Readability scores (rs_)	31	Features created using various readability methods + interactions.	Rix andLiix	01.02
NLTK features (nltk_)	36	Features created based on methods from the NLTK package.	POS Tags, stopwords	01.03
AOA features (aoa_)	40	Features created based on the AOA dataset.	freq_pm_sum, freq_pm_min	01.04
CRB features (crb_)	8	Features created based on the CRB dataset.	concm_mean and perc_known_sum.	01.05
Vector features (w2v_)	3	Used fastText and Gensim packages to create vector-based features.	avg_word2vec_mean, sen2vec_mean	01.06

The complete feature list is available under Appendix 1 - Full Feature List.

[3] Jupyter Notebook 01.01
[4] Language is presented twice, numerically and as a string (ISO code).

3.4 Model Development

Our final model development workflow, which was utilized for the sentence difficulty classification, is illustrated in figure 2:

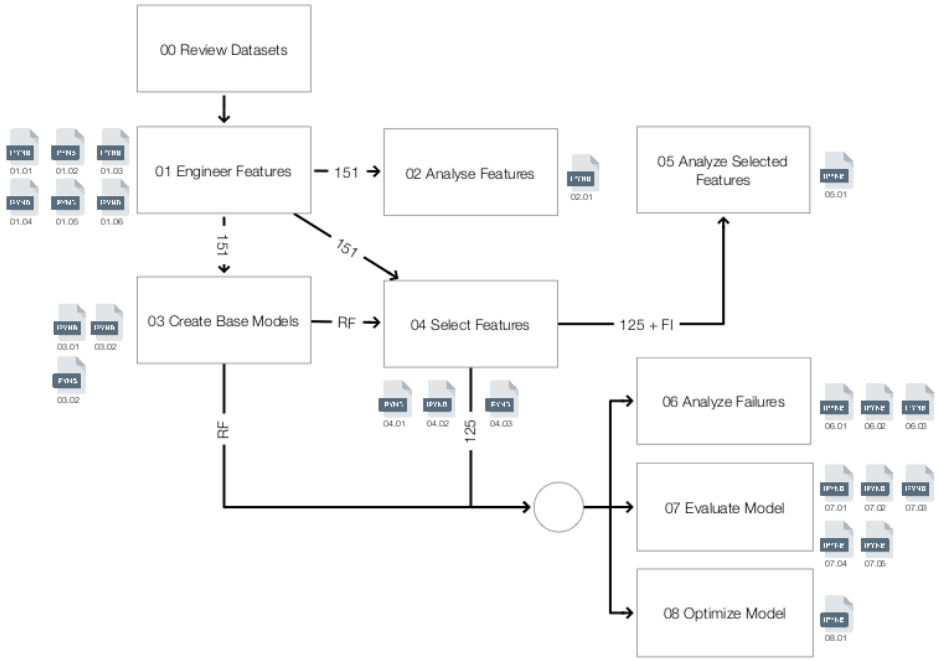


Fig. 2: Final Model Development Workflow

3.4.1 Supervised Machine Learning

After creating the features and exploring the data (JN02.01), our next task was to select suitable machine learning algorithms. We built baseline models based on commonly used algorithms, including RandomForest, Logistic Regression, Gaussian Naive Bayes, Multinomial Naive Bayes, DecisionTree, and KNeighbors. However, since one of the objectives of our project is to achieve interpretability, we assumed that the following machine learning algorithms would be the best candidates for the project (JN03.01 to JN03.03):

ML Algorithm	Family	Assumption / Why included in base models
Random Forest Classifier	Ensemble	The Random Forest algorithm was suitable as it works well with categorical and continuous variables and efficiently handles nonlinear parameters. It is easy to interpret and implement.
Naive Bayes Gaussian Classifier	Naive Bayes	Naive Bayes was another preferred choice when classifying or categorizing data based on the presence of specific features or attributes. It has proven to be very stable and reliable in text classification problems such as spam detection, and it would be a good fit for classifying text difficulty.
Logistic Regression	Linear Model	Logistic regression was a logical choice because the dependent variable is binary.

We used the accuracy score as the measure to evaluate the performance of the models. We also set a benchmark of achieving a score of 0.7 or higher to consider the model successful. We used these algorithms as a starting point to understand which model would be a good fit for our project.

3.4.1.1 Supervised Evaluation

We evaluated the performance of our base models using a combination of classification metrics, including accuracy, recall, precision, F1 score, ROC-AUC, and confusion matrix (see Fig. 3 and Fig. 4). The robustness of the evaluation was ensured by applying a 10-fold stratified cross-validation approach, and the features were scaled using MinMax. By leveraging this methodology, we could compare the performance of our models and select the best one for our classification task.

Model	Accuracy	Acc STD	Acc CI	Recall	Precision	F1	ROC AUC
Random Forerst	0.7489	0.0011	0.0007	0.7636	0.7418	0.7525	0.8437
Logistic Regression	0.6660	0.0015	0.0010	0.6734	0.6635	0.6684	0.7207
Gaussian NB	0.6199	0.0017	0.0011	0.7476	0.5955	0.6630	0.6655
Multinomial NB	0.6133	0.0017	0.0010	0.5418	0.6322	0.5835	0.6690
Decision Tree	0.6940	0.0014	0.0009	0.6908	0.6953	0.6930	0.6954
KNeighbors	0.6673	0.0010	0.0006	0.6919	0.6595	0.6753	0.7324

Fig. 3: Cross-Validation Scores incl. Confidence Intervals (CI)

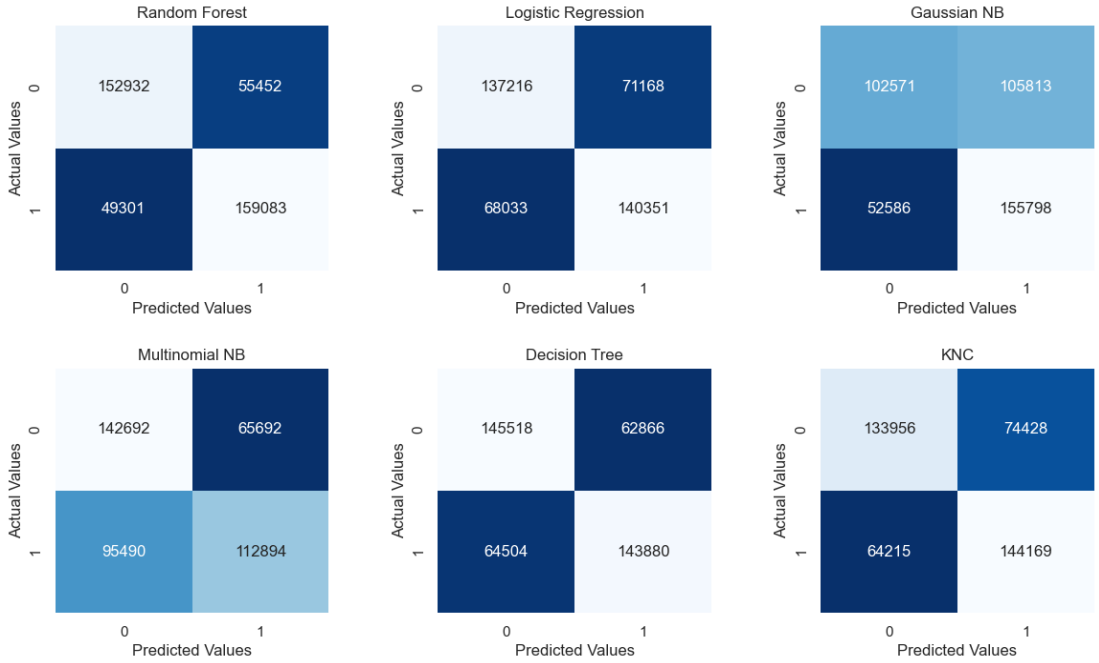


Fig. 4: Cross-Validation Confusion Matrices

The **Random Forest** model demonstrated superior performance across all classification metrics, including an accuracy score of **0.749**, while Naive Bayes showed lower performance than expected. However, further analysis revealed that using TfidfVectorizer as the only feature significantly enhanced the accuracy score of the **Multinomial NB** model to **0.749**. Despite this improvement, we decided to select the Random Forest model moving forward, as it would be easier to fine-tune and interpret.

Feature Selection and Importance

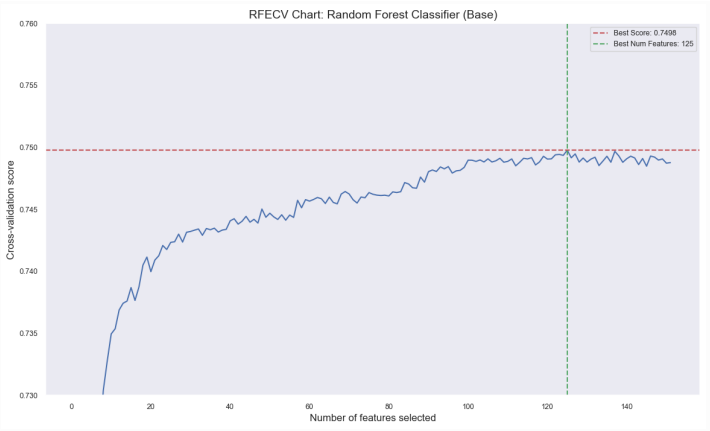


Fig. 5: RFECV Feature Selection

We employed the recursive feature elimination (RFE) method for the feature importance and selection process. RFE works by fitting a model and iteratively removing the weakest features until the desired number of features is reached. The features are ranked based on the model's coef_ or feature_importances_ attributes and by recursively eliminating a small number of features per loop. RFE also aims to eliminate dependencies and collinearity that may exist in the model. This method allowed us to identify the most important features of our model. RFE selected out of 151, the top 125 features most important for the model's performance, as illustrated in figure 5. In addition, RFE also provided the feature importance values, as displayed in figure 6 (JN04.01).

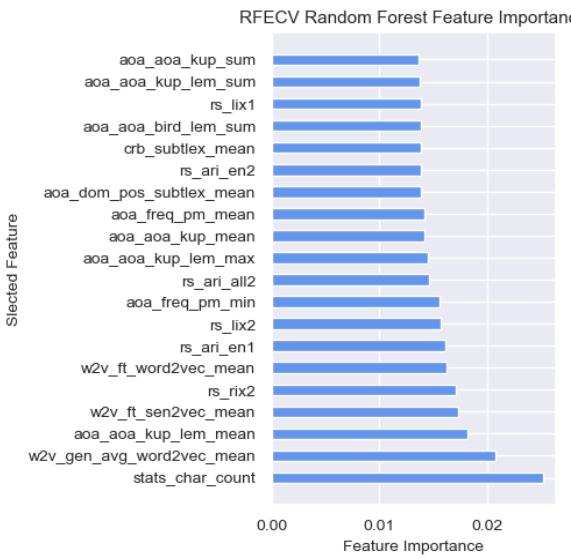


Fig. 6: RFECV Feature Importance (Top 20)

Model Evaluation: Learning Curve

Evaluating the model's performance involved analyzing the relationship between the number of training samples and the model's accuracy. To achieve this, we plotted the learning curve (Fig. 7), which visually represents the model's performance as the number of training samples increased. The learning curve showed that our model's accuracy improved consistently with the size of the training set and did not reach a plateau, suggesting that the model could benefit from additional training samples to further improvements in the performance (JN07.01).

Model Evaluation: Sensitivity Analysis

Once we better understood how the training size impacted the model's performance, we conducted a sensitivity analysis to explore the effects of varying key parameters on performance. The aim was to gain insights into the model's sensitivity to changes in these parameters. To achieve this, we examined the model's sensitivity concerning the following key parameters: n_estimators, max_features, and max_depth. To ensure the robustness of the evaluation, we employed a 10-fold stratified cross-validation approach with the features scaled using MinMax. We used a randomized cv search with 25 iterations to evaluate the performance of the model under different parameter settings. Figures 8 to 10 show the model's performance for max_features, max_depth, and n_estimators with a common trend: As the values of these parameters increase, the model's performance initially improves and reaches a peak value. Beyond this point, further increments in these parameter values do not result in significant changes in performance. This behavior suggests the model is robust and can operate effectively within a range of parameter values without being overly sensitive to small changes (JN07.02 to JN07.05). Two additional 3-dimensional plots are available in Appendix 2.

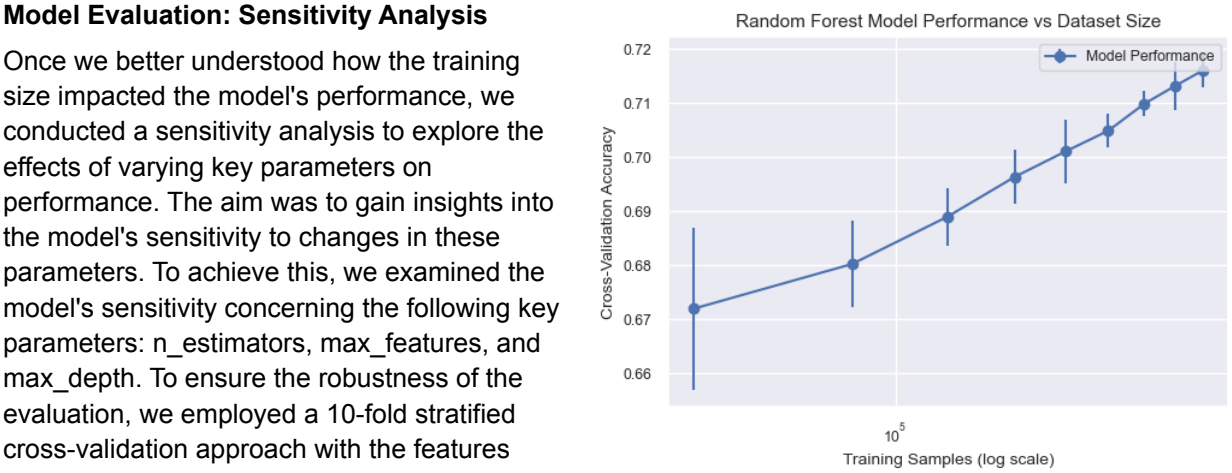


Fig. 7: Learning Curve

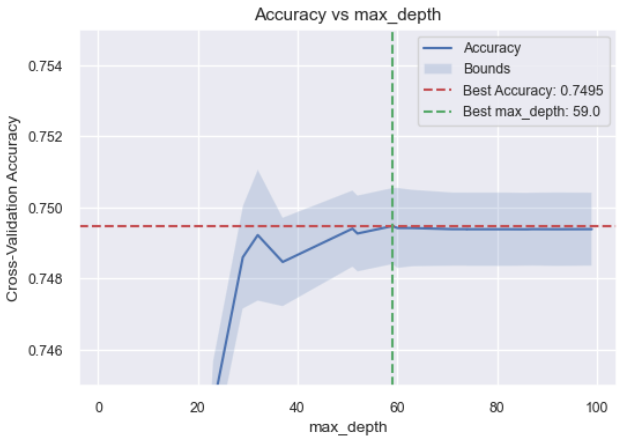


Fig. 8: Sensitivity Max_Depth

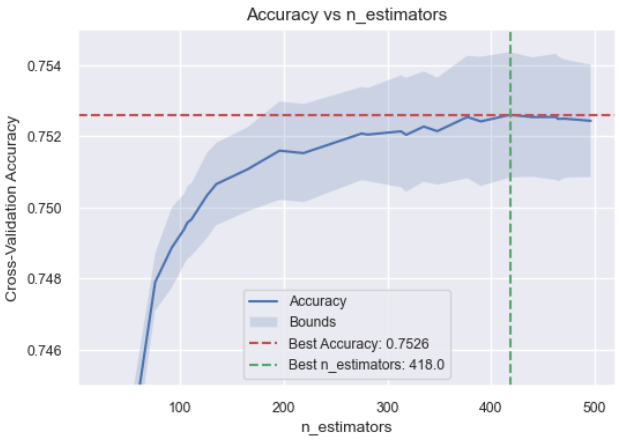


Fig. 9: Sensitivity N_Estimators

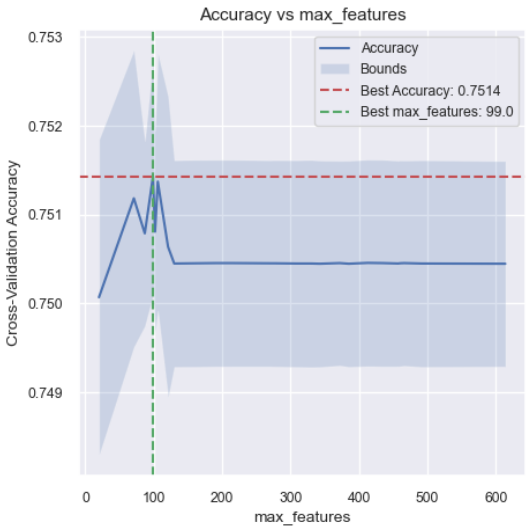


Fig. 10: Sensitivity Max_Features

Hyperparameter Optimization

We also used the randomized cv search with 25 iterations for the hyper-parameter optimization. The search ranges were defined as 'n_estimators' between 5 and 500, 'max_features' between 0 and 100, 'max_depth' between 0 and 100, 'min_samples_split' between 2 and 10, and 'min_samples_leaf' between 1 and 10 (JN08.01).

The best combination of hyper-parameters was found to be 'max_depth': 59, 'max_features': 40, 'min_samples_leaf': 1, 'min_samples_split': 2, and 'n_estimators': 331, which resulted in an improved accuracy score of **0.7544** with a low standard deviation of 0.0016.

3.4.1.2 Failure Analysis

In our next step, we proceeded with a failure analysis to identify any systemic causes of failures in our model. We started with an overview of potential **failure categories** outlined in the following table. By conducting this analysis, we aimed to identify the root causes of any issues to develop corrective actions to improve the model's performance (JN06.01 to JN06.03).

Failure Category	Description	Outcomes	Key Observations and Recommendations
Imbalanced Data	Both classes are well balanced.		
Overfitting	The model was not overfitting: Ensured via 10-fold cross-validation and test submissions to Kaggle.		
Dealing with Ambiguous Examples	Some sentences may be difficult to classify based on the available information, especially if they are simple or complex.	Labeling issues observed	To improve the accuracy of our text classification model, it may be necessary to exclude short and structurally incomplete sentences from the labeling process and the test and training data. These types of sentences are difficult to label accurately and can negatively impact the model's performance. **Please find the complete analysis below.**
Incorrect Features Selection	If the model is not trained on the right features, it might miss the important aspects of the examples.	The features that were chosen may not be the most significant	Selecting the right features is crucial for accurately classifying text data. Even with over 150 features, important information can only be noticed if chosen carefully. It is crucial to monitor the model's performance and ensure it captures the essential aspects of the data.
Data Quality Issues	When working on text classification, messy or error-prone data can make it difficult for the model to classify sentences.	Data quality issues observed	As with the labeling process, it is also necessary to exclude short sentences lacking structure from the training and test processes to improve the accuracy of the text classification model. **Please find the complete analysis below.**
Hyper-parameter Tuning	If the model's parameters are wrong, it might not perform well when tested.	No issues observed	We did not observe any issues related to the model's parameters, which can impact its performance during testing.

As part of the failure analysis, we evaluated the two most important key features for the model's performance: stats_char_count, which measures the number of characters in a text, and w2v_gen_avg_word2vec_mean, which is a statistical measure of the average word embedding for a text. We began our analysis by comparing the true and predicted labels for matched and mismatched classifications. Using exploratory data analysis, we identified areas that required further investigation. Our analysis indicated that most mismatched results were concentrated in a specific region, defined by a red rectangle on the chart (Fig. 11). This region was characterized by values between 0 and 250 for the stats_char_count feature and between -0.3 and 0.3 for the w2v_gen_avg_word2vec_mean feature.

Fig. 11

We focused our investigation on the area previously identified for both labels. Our observations indicated that most misclassifications were found in specific regions (red rectangular), as evidenced by the heatmaps presented below (Fig. 12). However, we also noted some slight variations in these regions between the labels, particularly concerning the number of characters present in the text data. The charts presented depict the percentage of misclassified labels concerning the total number of cases within the selected range. Our analysis indicates that within the selected range, 26.73% of cases for label 0 and 24.18% of cases for label 1 were misclassified. Additionally, we observed that 95% of the misclassified cases were concentrated within the selected range for each label.

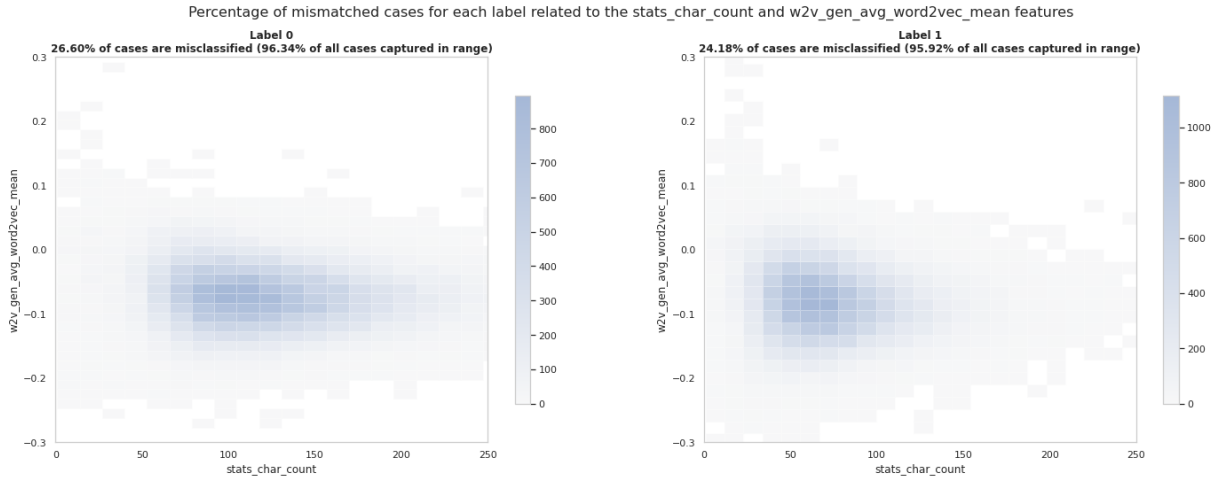


Fig.12

We have made some significant observations during our analysis: **(A)** We noticed that text cases with less than 25 characters are of low quality and pose a challenge in accurately labeling them. Therefore, we recommend reassessing and removing such cases as necessary. **(B)** For text cases between 50 to 150 characters, it is often unclear why some sentences are labeled as 0 or 1. To tackle this issue, we suggest

involving domain experts and considering using paragraphs instead of isolated sentences or text fragments in future analyses. We also utilized KMeans clustering for our failure analysis. For more details, please refer to chapter 3.4.2.1 Unsupervised Evaluation.

3.4.2 Unsupervised Machine Learning

Our initial plan was to use KMeans clustering to evaluate the performance of our constructed features by assessing their ability to separate the samples and potentially use clustering as an additional feature for the model. However, the first tests revealed that clustering was not practical for either of these use cases. Nevertheless, we discovered that clustering could help in the failure analysis by providing insights into how the failed samples are grouped to identify commonalities between them. The Principal Component Analysis (PCA) will be used to understand the significance of the most important features. Also, the principal components could be considered a feature. PCA biplot and scree plot will be used for the visualization.

3.4.2.1 Unsupervised Evaluation

Principal Component Analysis (PCA)

In our work, we utilized PCA for unsupervised learning to decrease the dimensionality of a dataset containing 151 features (JN05.01). To ensure accurate results, we considered the sensitivity of PCA to the scale of the data and employed MinMaxScaler for data normalization. We aimed to scale the data to have unit variance and avoid feature domination. Throughout our analysis, we calculated the first two principal components to detect any existing patterns or trends in the data.

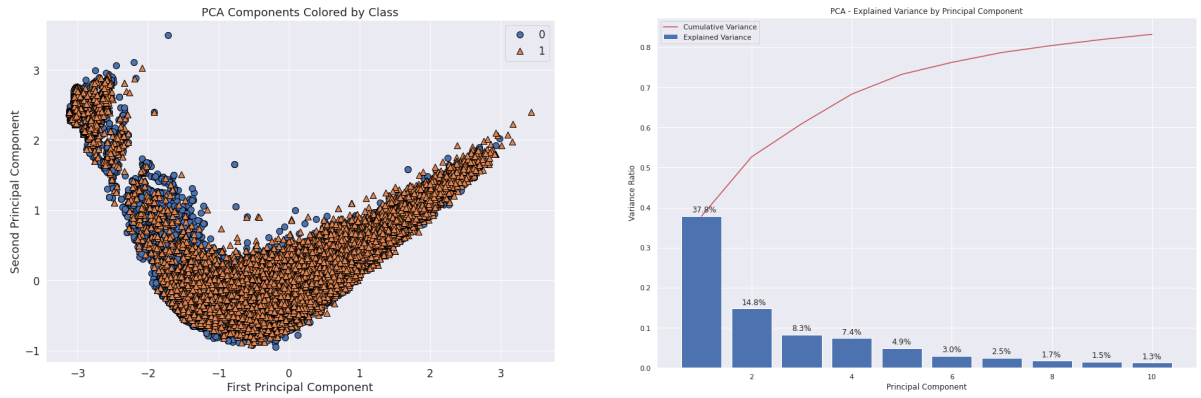


Fig. 13

We also created a heatmap plot of the components to visualize the relationship between the features and the first two principal components. However, we were unable to observe any meaningful results (Fig. 14).

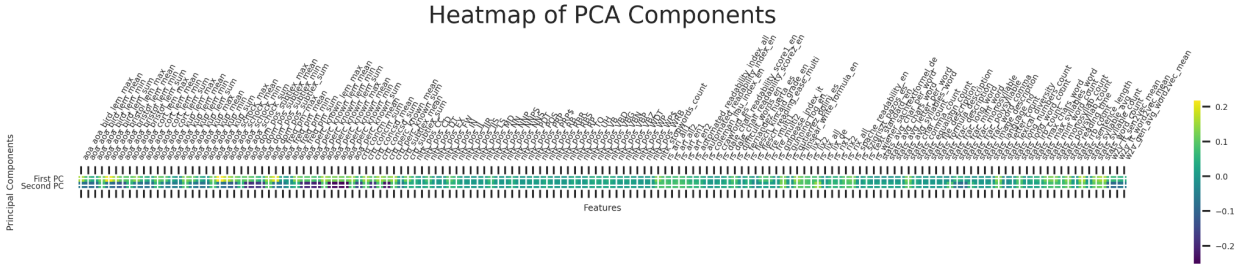


Fig. 14

Therefore, we narrowed the analysis to the top five features significantly contributing to the model's accuracy using PCA. The five most important features in our analysis are:

- "w2v_gen_avg_word2vec_mean" - this feature represents the average word vector for each text using the Word2Vec algorithm, which could capture semantic relationships between words.
- "aoa_aoa_kup_lem_mean" - this feature represents the average age of acquisition of words in the text, which could capture the text's complexity or level of difficulty.

- "w2v_ft_sen2vec_mean" represents the average word embedding for a sentence using the fastText algorithm, which, like the Word2Vec-based feature described above, was included to capture the semantic content of each sentence.
- "rs_rix2" - this feature represents the Rix readability measure, which is a method for determining the complexity of a sentence based on the number of syllables and the number of long words.

After data normalization using the MinMaxScaler, we calculated the first two principal components to uncover any underlying patterns or trends in the data. The first two principal components captured 86.7% of the total variation in the dataset.

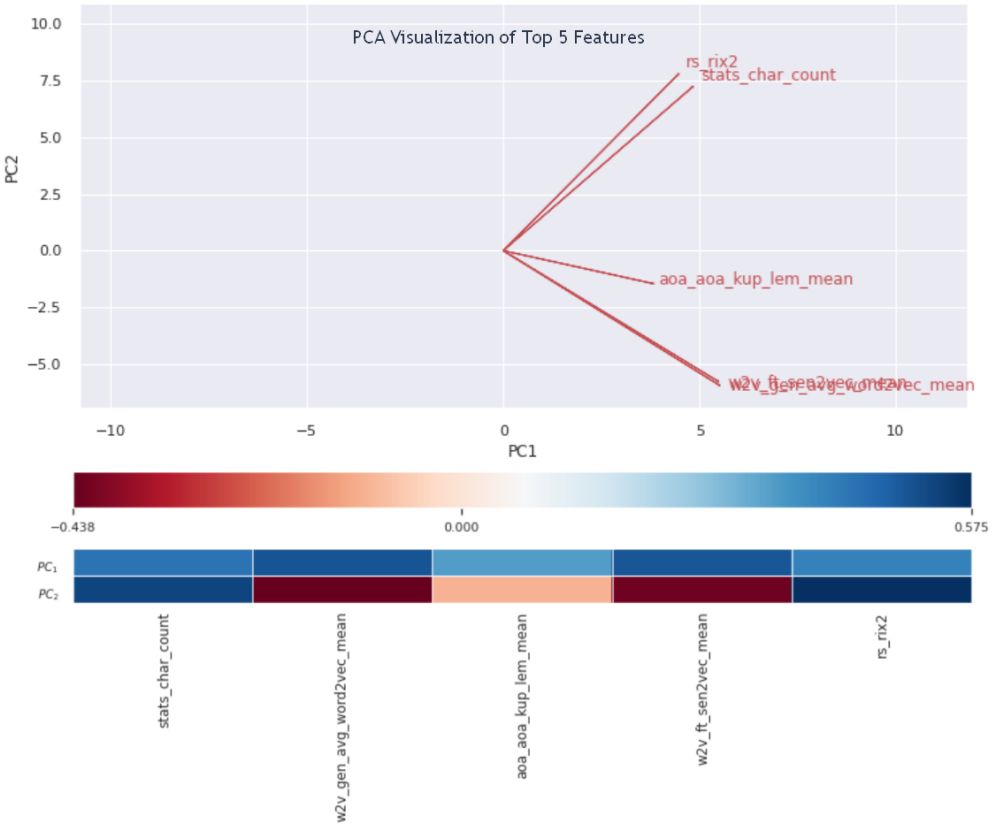


Fig. 15: PCA Biplot and Heatmap

The biplot analysis (Fig. 15) suggests that:

- "w2v_gen_avg_word2vec_mean" and "w2v_ft_sen2vec_mean" are closely associated and clustered together, possibly because they both use word embedding techniques to capture the semantic content of words. This means that these features may be capturing similar information about the overall meaning and topic of the sentence.
- "stats_char_count" and "rs_rix2" are highly associated and clustered together, likely because they both relate to the length and complexity of the sentence. "stats_char_count" measures the number of characters in the sentence. At the same time, "rs_rix2" determines the sentence complexity based on syllables and long words used so that both features may provide information on the overall difficulty or complexity of the sentence.
- "aoa_aoa_kup_lem_mean" is closer in angle to "w2v_gen_avg_word2vec_mean" and "w2v_ft_sen2vec_mean", which suggests some correlation between them. However, this association may be weaker than the other two feature pairs. This could be because "aoa_aoa_kup_lem_mean" measures the age of acquisition of words in the sentence, which relates to familiarity and understanding, but not necessarily the semantic content. As such, "aoa_aoa_kup_lem_mean" may be capturing specific information about the sentence that is not reflected in the other two features but still has some correlation with them.

KMeans Clustering

Once the failed samples were isolated, we divided them into two groups based on the potential misclassification (JN06.03): (A) predicted 1 instead of 0, and (B) predicted 0 instead of 1. We also narrowed our focus to the **four most important features** to simplify readability and interpretation: 'stats_char_count', 'w2v_gen_avg_word2vec_mean', 'aoa_aoa_kup_lem_mean', and 'w2v_ft_sen2vec_mean'. As a first step, we ran an elbow analysis to determine the optimal number of clusters for each group:

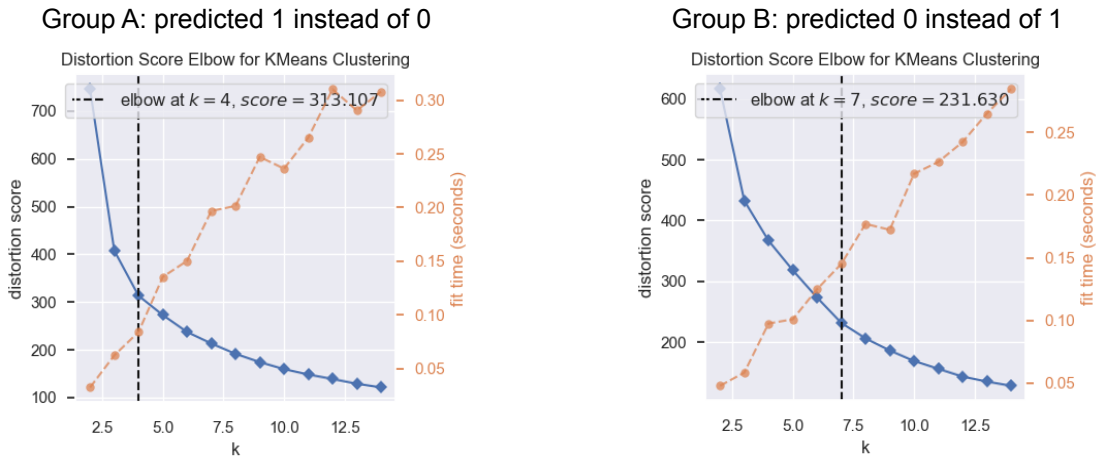


Fig. 16: Group A and B Elbow Method Results

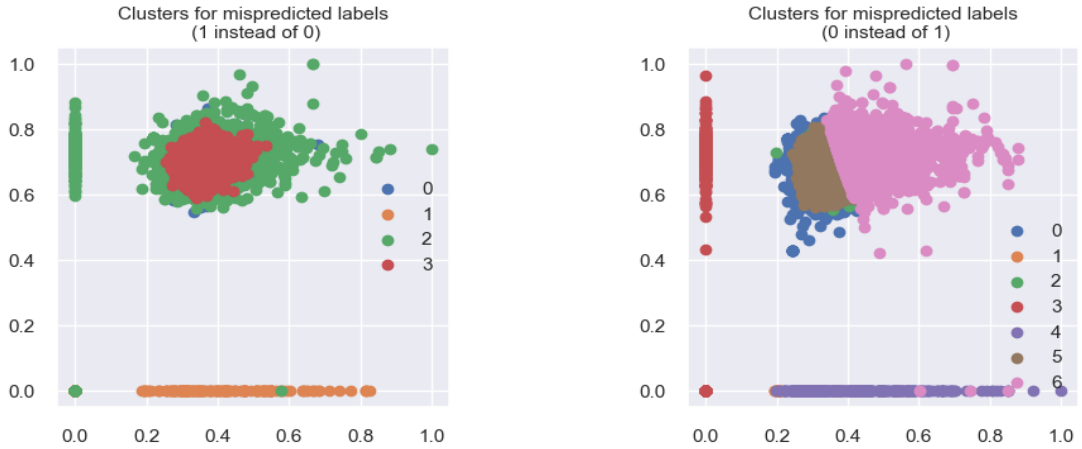
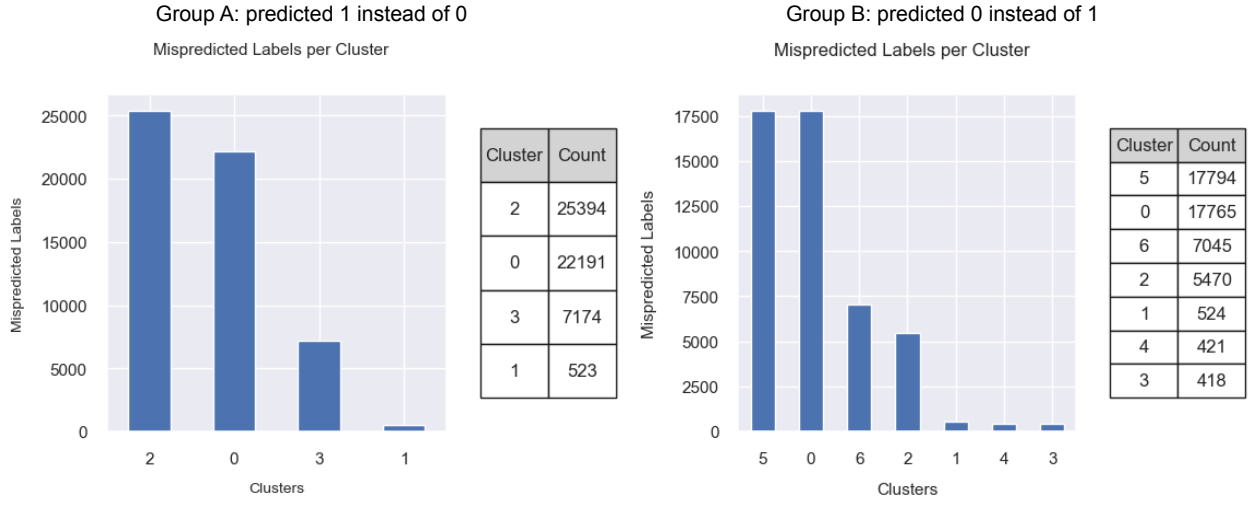


Fig. 17: Group A and B Clusters

Afterwards, we identified and focused on the clusters with the most mispredicted labels:



Cluster	Count
2	25394
0	22191
3	7174
1	523

Cluster	Count
5	17794
0	17765
6	7045
2	5470
1	524
4	421
3	418

Fig. 18

For the two top clusters, we compared the histograms of the features to spot any significant patterns. The graphs below show the most interesting differences.

Group A: Showing the most significant differences between the two clusters:

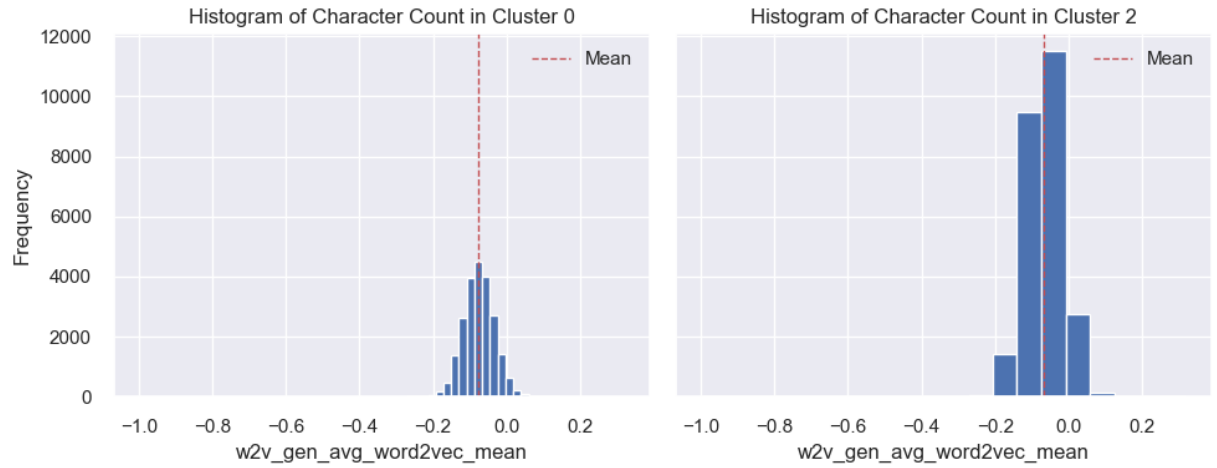


Fig. 19

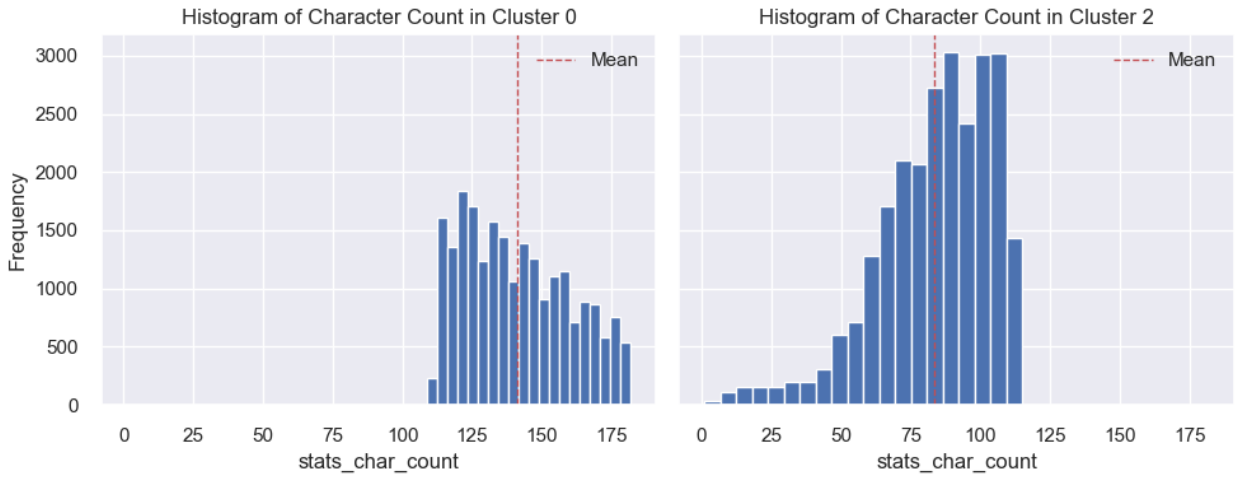


Fig. 20

Conclusion: Group A - Cluster 0

	390849	236573	407231	217147	222358
stats_char_count	163	143	145	114	120
w2v_gen_avg_word2vec_mean	-0.11252	-0.09259	-0.03051	-0.04039	-0.10951
aoa_aoa_kup_lem_mean	5.77	6.01	6.56	6.58	5.56
w2v_ft_sen2vec_mean	-0.00052	0.00005	0.00037	0.00045	0.00187
label	0	0	0	0	0
y_pred	1	1	1	1	1
original_text	With Drake looking like they were going to continue to win , Burton started a plan to purposely lose the fourth Immunity challenge to get rid of the tribe 's weaker players such as Trish and Christa .	More praise was given to its size and future potential based on user-created content . Small criticism was made for certain parts of the gameplay and level creation tools .	The palace is a complex of buildings , with the Papal Apartment , the Catholic Church 's government offices , a handful of chapels , the Vatican Museum and the Vatican library .	A thin film transistor liquid crystal display -LRB- TFT-LCD -RRB- is a technology which is used in LCD monitor and television displays .	After the Yankees took a 4 - 2 lead on a Don Mattingly 2-RBI double in the 6th inning , the Mariners came back to tie the score at 4 - 4 in the 8th inning .
cleaned_text	with drake looking like they were going to continue to win burton started a plan to purposely lose the fourth immunity challenge to get rid of the tribe weaker players such as trish and christa	more praise was given to its size and future potential based on user created content small criticism was made for certain parts of the gameplay and level creation tools	the palace is a complex of buildings with the papal apartment the catholic church government offices a handful of chapels the vatican museum and the vatican library	a thin film transistor liquid crystal display tft lcd is a technology which is used in lcd monitor and television displays	after the yankees took a 4 2 lead on a don mattingly 2 rbi double in the 6th inning the mariners came back to tie the score at 4 4 in the 8th inning
cluster	0	0	0	0	0

Fig. 21

For all mispredicted labels, the stats_char_count is over 109 chars. When looking at the samples, the true labels were most likely incorrect and should be a 1 instead of a 0, pointing to a labeling issue. We would need to consult a domain expert to decide if the original dataset could be re-labeled.

Conclusion: Group A - Cluster 2

	296516	284396	406781	414268	253377
stats_char_count	23	7	2	38	49
w2v_gen_avg_word2vec_mean	-0.03813	-0.11045	-0.10799	-0.15612	-0.04433
aoa_aoa_kup_lem_mean	-1.0	3.75	-1.0	6.85	5.4
w2v_ft_sen2vec_mean	-0.00127	-0.0017	0.00081	-0.00245	0.00209
label	0	0	0	0	0
y_pred	1	1	1	1	1
original_text	Río Grande -LRB- M.D. -RRB-	In music	N !	Szombathely , Hungary -LRB- since 1991 -RRB-	At the 2006 census , Airlie Beach had a population of 2,751 .
cleaned_text	r o grande m d	in music	n	szombathely hungary since 1991	at the 2006 census airlie beach had a population of 2 751
cluster	2	2	2	2	2

Fig. 22

For samples with a stats_char_count less than 50, we are most likely looking at incomplete sentences, which should be removed, and if not, we would need to explore why the aoa_aoa_kup_lem_mean is high or why the value of -1 does not help with the prediction.

Group B: Showing the most significant differences between the two clusters:

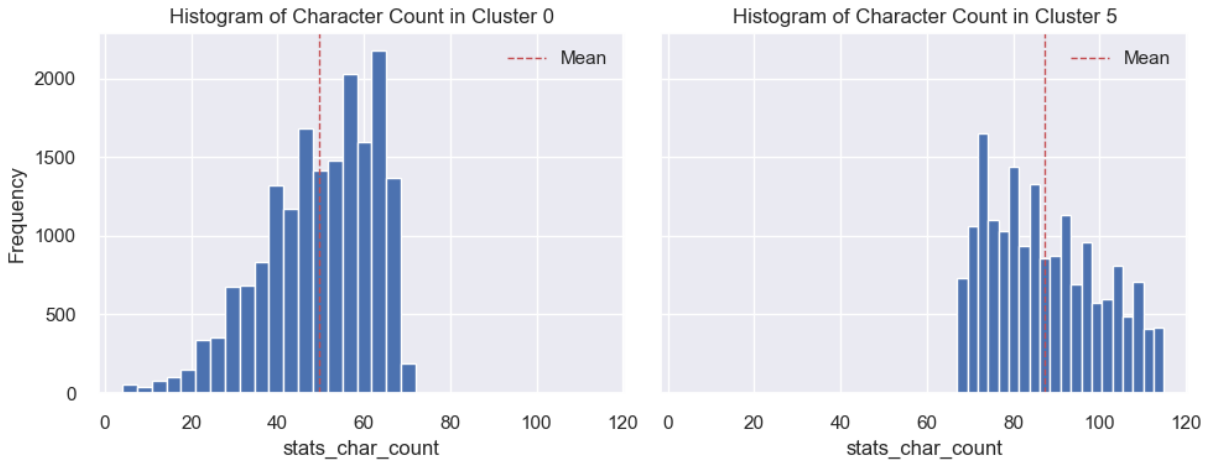


Fig. 23

Conclusion: Group B - Cluster 0

	134516	188872	186269	124530	96336
stats_char_count	45	40	57	58	50
w2v_gen_avg_word2vec_mean	-0.11734	-0.08383	-0.13134	-0.11155	-0.09181
aoa_aoa_kup_lem_mean	4.58	5.41	6.1	5.41	5.13
w2v_ft_sen2vec_mean	-0.00185	0.00292	0.00006	-0.00035	-0.0008
label	1	1	1	1	1
y_pred	0	0	0	0	0
original_text	It was the worst accident in Romania in fifteen years .	The first Post Office opened on 1 November 1851 .	Jennifer Todd -LRB- born c. 1969 - RRB- is an American film producer .	'' 03 Bonnie & Clyde '' is a song by Jay-Z featuring R&B singer Beyoncé .	Grover is shown to be a bit cowardly , getting afraid easily .
cleaned_text	it was the worst accident in romania in fifteen years	the first post office opened on 1 november 1851	jennifer todd born c 1969 is an american film producer	03 bonnie clyde is a song by jay z featuring r b singer beyonce	grover is shown to be a bit cowardly getting afraid easily
cluster	0	0	0	0	0

Fig. 24: Samples Group B, Cluster 0

For all mispredicted labels, the stats_char_count is below 72 chars. When looking at the samples, the actual labels are most likely incorrect and should be a 0 instead of a 1, also pointing to a labeling issue.

Conclusion: Group B - Cluster 5

	102360	163600	131087	1164	199462
stats_char_count	69	71	71	100	85
w2v_gen_avg_word2vec_mean	-0.09809	-0.07186	-0.05501	-0.04899	-0.0675
aoa_aoa_kup_lem_mean	5.35	5.2	5.06	5.77	5.22
w2v_ft_sen2vec_mean	-0.00032	-0.0014	-0.00161	-0.00003	-0.00058
label	1	1	1	1	1
y_pred	0	0	0	0	0
original_text	An avid reader , she also spent a lot of time exploring around her family 's 65 a farm .	Since 1935 , both sides of the Great Seal appear on the reverse of the one-dollar bill .	Saturn was the father of Ceres , Jupiter , Veritas , Pluto , and Neptune , among others .	The town grew after the nation 's administrative centre moved from Bremersdorp -LRB- now called Manzini - RRB- in 1902 .	Originally built for up to one million inhabitants , the city has recently grown way past this number .
cleaned_text	an avid reader she also spent a lot of time exploring around her family 65 a farm	since 1935 both sides of the great seal appear on the reverse of the one dollar bill	saturn was the father of ceres jupiter veritas pluto and neptune among others	the town grew after the nation administrative centre moved from bremersdorp now called manzini in 1902	originally built for up to one million inhabitants the city has recently grown way past this number
cluster	5	5	5	5	5

Fig. 25: Samples Group B, Cluster 5

For cluster 5 in group B, we could not identify a distinct differentiator for this cluster in the current iteration. In future project iterations, we can utilize cluster statistics which can provide valuable information about the characteristics of the cluster. By examining the statistics of each feature jointly, we can uncover patterns and relationships that were not initially apparent.

```
Cluster: 5
```

	min	max	mean	std
stats_char_count	67.00000	115.00000	87.297741	12.740925
w2v_gen_avg_word2vec_mean	-0.25739	0.08325	-0.085102	0.046310
aoa_aoa_kup_lem_mean	3.64000	8.21000	5.497575	0.532644
w2v_ft_sen2vec_mean	-0.00482	0.00411	-0.000437	0.000948
label	1.00000	1.00000	1.000000	0.000000
y_pred	0.00000	0.00000	0.000000	0.000000
cluster	5.00000	5.00000	5.000000	0.000000

Fig. 26: Cluster Statistics Group B, Cluster 5

4.0 Conclusion & Discussion

In summary, our analysis found that the fine-tuned Random Forest model demonstrated the highest accuracy score of 0.7544. Nevertheless, further data cleansing measures failed to yield the anticipated enhancements despite our diligent attempts. Furthermore, we were surprised to discover that the RFECV method identified 125 features, out of the original 151, as the optimal combination for the model. We initially expected that fewer features would be relevant.

In order to evaluate the model's performance, we conducted a learning curve and sensitivity analysis. Our results demonstrated that the Random Forest model exhibits robustness and can operate effectively within a range of parameter values without being overly sensitive to minor changes.

Our failure analysis revealed potential issues with the accuracy of the training set's labels, as some simple sentences may need to be correctly labeled as complex and vice versa. Consequently, this could explain why our extensive data cleaning efforts failed to improve the model's accuracy and why none of our constructed features could adequately separate the sentences into simple or complex categories. Therefore, we recommend discussing our findings with a domain expert to confirm the presence of any underlying issues with the labeling process. We also suggest examining and adjusting the process of scraping and extracting the sentences from the "Simple English Wikipedia" and expanding the sampling to complete paragraphs instead of just single sentences. Such an approach would provide more comprehensive feature creation and model fitting information.

Throughout our analysis, we maintained privacy and data security by using third-party platforms like Google Collab and Trello for source code development and project management while adhering to appropriate access and privacy controls. We also ensured the model's outcomes were interpretable and understandable to promote transparency and accountability. Additionally, we received feedback from our project coach to refine our approach and improve the final product.

Lastly, our initial experiments with the MultinomialNB and TfidfVectorizer algorithms for text classification have shown promising results. These algorithms are simple to implement and computationally less demanding, making them a good starting point for following text classification tasks.

As a final conclusion, while the Random Forest algorithm produced the best results, issues with the dataset and labeling present significant challenges that should be addressed in future project iterations.

5.0 Ethical Considerations

Our analysis did not identify any significant ethical considerations that needed to be addressed in relation to this project. However, it is necessary to continuously monitor and evaluate potential ethical implications as the project progresses in the future.

6.0 Statement of Work

Activities	**Lead**	**Support**
Related Work	Victor	Andre
Data Sources	Victor	Andre
Data Cleaning and Understanding	Jointly	
Feature Engineering	Andre	Victor
Model Development (Flow)	Andre	Victor
Supervised ML: RF + Evaluation	Andre	Victor
Supervised ML: MNB	Victor	Andre
Unsupervised ML	Victor	Andre
Conclusion & Decision	Jointly	
Ethical Consideration	Jointly	

7.0 References

Related Work

- Supervised and Unsupervised Neural Approaches to Text Readability (2021), Matej M., Senja P., and Marko R.S, link
- Text Difficulty Classification by Combining Machine Learning and Language Features (2022), Han Ding, Qiyu Zhong, Shaohong Zhang and Liu Yang, 2022, link
- Applying Natural Language Processing and Hierarchical Machine Learning Approaches to Text Difficulty Classification (2020), Renu Balyan, Kathryn S. McCarthy, and Danielle S. McNamara, link

Jupyter Notebooks

- JN01.01: 01.01_data_cleaning_and_text_stats_features
- JN01.02: 01.02_calculate_readability_score_features
- JN01.03: 01.03_calculate_nltk_features
- JN01.04: 01.04_calculate_aoa_features
- JN01.05: 01.05_calculate_crb_features
- JN01.06: 01.06_calculate_w2v_features
- JN02.01: 02.01_analyze_all_features
- JN03.01: 03.01_modeling_base_models_with_CV
- JN03.02: 03.02_modeling_base_models_with_CV-CM
- JN03.03: 03.03_modeling_base_models_MultinomialNB
- JN04.01: 04.01_feature_selection_RFECV-RF
- JN04.02: 04.02_feature_selection_variance
- JN04.03: 04.03_feature_selection_feature_ablation
- JN05.01: 05.01_analyse_selected_features_PCA
- JN06.01: 06.01_failure_analysis_predicting_y_with_RFECV_features
- JN06.02: 06.02_failure_analysis_misclassification_heatmap
- JN06.03: 06.03_failure_analysis_KMeans
- JN07.01: 07.01_model_evaluation_learning_curve
- JN07.02: 07.02_model_evaluation_sensitivity_analysis_RF-max_features
- JN07.03: 07.03_model_evaluation_sensitivity_analysis_RF-max_depth
- JN07.04: 07.04_model_evaluation_sensitivity_analysis_RF-n_est_and_max_depth
- JN07.05: 07.05_model_evaluation_sensitivity_analysis_RF-n_estimators
- JN08.01: 08.01_model_optimization_hyperparameter_optimization

Document History

Version	Date	Authors	Comments
1.0	2023-02-28	Andre Buser Victor Adafinoaiei	Initial version for submission.

APPENDICES

APPENDIX 1 - Full Feature List

Feature	Feature Group	Description
aoa_aoa_bird_lem_max	aoa	Max value for matched words
aoa_aoa_bird_lem_mean	aoa	Mean value for matched words
aoa_aoa_bird_lem_min	aoa	Min value for matched words
aoa_aoa_bird_lem_sum	aoa	Sum for matched words
aoa_aoa_bristol_lem_max	aoa	Max value for matched words
aoa_aoa_bristol_lem_mean	aoa	Mean value for matched words
aoa_aoa_bristol_lem_min	aoa	Min value for matched words
aoa_aoa_bristol_lem_sum	aoa	Sum for matched words
aoa_aoa_cort_lem_max	aoa	Max value for matched words
aoa_aoa_cort_lem_mean	aoa	Mean value for matched words
aoa_aoa_cort_lem_min	aoa	Min value for matched words
aoa_aoa_cort_lem_sum	aoa	Sum for matched words
aoa_aoa_kup_lem_max	aoa	Max value for matched words
aoa_aoa_kup_lem_mean	aoa	Mean value for matched words
aoa_aoa_kup_lem_min	aoa	Min value for matched words
aoa_aoa_kup_lem_sum	aoa	Sum for matched words
aoa_aoa_kup_max	aoa	Max value for matched words
aoa_aoa_kup_mean	aoa	Mean value for matched words
aoa_aoa_kup_min	aoa	Min value for matched words
aoa_aoa_kup_sum	aoa	Sum for matched words
aoa_aoa_schock_max	aoa	Max value for matched words
aoa_aoa_schock_mean	aoa	Mean value for matched words
aoa_aoa_schock_min	aoa	Min value for matched words
aoa_aoa_schock_sum	aoa	Sum for matched words
aoa_dom_pos_subtlex_max	aoa	Max value for matched words
aoa_dom_pos_subtlex_mean	aoa	Mean value for matched words
aoa_dom_pos_subtlex_min	aoa	Min value for matched words
aoa_dom_pos_subtlex_sum	aoa	Sum for matched words
aoa_freq_pm_max	aoa	Max value for matched words
aoa_freq_pm_mean	aoa	Mean value for matched words
aoa_freq_pm_min	aoa	Min value for matched words
aoa_freq_pm_sum	aoa	Sum for matched words
aoa_perc_known_lem_max	aoa	Max value for matched words
aoa_perc_known_lem_mean	aoa	Mean value for matched words
aoa_perc_known_lem_min	aoa	Min value for matched words

Feature	Feature Group	Description
aoa_perc_known_lem_sum	aoa	Sum for matched words
aoa_perc_known_max	aoa	Max value for matched words
aoa_perc_known_mean	aoa	Mean value for matched words
aoa_perc_known_min	aoa	Min value for matched words
aoa_perc_known_sum	aoa	Sum for matched words
crb_concm_mean	crb	Mean for matched words
crb_concm_sum	crb	Sum for matched words
crb_concsd_mean	crb	Mean for matched words
crb_concsd_sum	crb	Sum for matched words
crb_perc_known_mean	crb	Mean for matched words
crb_perc_known_sum	crb	Sum for matched words
crb_subtlex_mean	crb	Mean for matched words
crb_subtlex_sum	crb	Sum for matched words
nltk_pos_CC	nltk	Count of pos tag in sentence
nltk_pos_CD	nltk	Count of pos tag in sentence
nltk_pos_DT	nltk	Count of pos tag in sentence
nltk_pos_EX	nltk	Count of pos tag in sentence
nltk_pos_FW	nltk	Count of pos tag in sentence
nltk_pos_IN	nltk	Count of pos tag in sentence
nltk_pos_JJ	nltk	Count of pos tag in sentence
nltk_pos_JJR	nltk	Count of pos tag in sentence
nltk_pos_JJS	nltk	Count of pos tag in sentence
nltk_pos_LS	nltk	Count of pos tag in sentence
nltk_pos_MD	nltk	Count of pos tag in sentence
nltk_pos_NN	nltk	Count of pos tag in sentence
nltk_pos_NNP	nltk	Count of pos tag in sentence
nltk_pos_NNPS	nltk	Count of pos tag in sentence
nltk_pos_NNS	nltk	Count of pos tag in sentence
nltk_pos_PDT	nltk	Count of pos tag in sentence
nltk_pos_POS	nltk	Count of pos tag in sentence
nltk_pos_PRP	nltk	Count of pos tag in sentence
nltk_pos_PRP$	nltk	Count of pos tag in sentence
nltk_pos_RB	nltk	Count of pos tag in sentence
nltk_pos_RBR	nltk	Count of pos tag in sentence
nltk_pos_RBS	nltk	Count of pos tag in sentence
nltk_pos_RP	nltk	Count of pos tag in sentence
nltk_pos_TO	nltk	Count of pos tag in sentence
nltk_pos_UH	nltk	Count of pos tag in sentence

Feature	Feature Group	Description
nltk_pos_VB	nltk	Count of pos tag in sentence
nltk_pos_VBD	nltk	Count of pos tag in sentence
nltk_pos_VBG	nltk	Count of pos tag in sentence
nltk_pos_VBN	nltk	Count of pos tag in sentence
nltk_pos_VBP	nltk	Count of pos tag in sentence
nltk_pos_VBZ	nltk	Count of pos tag in sentence
nltk_pos_WDT	nltk	Count of pos tag in sentence
nltk_pos_WP	nltk	Count of pos tag in sentence
nltk_pos_WP$	nltk	Count of pos tag in sentence
nltk_pos_WRB	nltk	Count of pos tag in sentence
nltk_stopwords_count	nltk	Count of stopwords in sentence
rs_ari_all1	rs	stats_word_count * rs_automated_readability_index_all
rs_ari_all2	rs	stats_syllable_count * rs_automated_readability_index_all
rs_ari_en1	rs	stats_word_count * rs_automated_readability_index_en
rs_ari_en2	rs	stats_syllable_count * rs_automated_readability_index_en
rs_automated_readability_index_all	rs	Textstat function: Returns the ARI (Automated Readability Index) which outputs a number that approximates the grade level needed to comprehend the text for all sentences.
rs_automated_readability_index_en	rs	Textstat function: Returns the ARI (Automated Readability Index) which outputs a number that approximates the grade level needed to comprehend the text for English sentences.
rs_coleman_liau_index_en	rs	Textstat function: Returns the grade level of the text using the Coleman-Liau Formula for English sentences. This is a grade formula in that a score of 9.3 means that a ninth grader would be able to read the document. Further reading on Wikipedia.
rs_crawford_es	rs	Textstat function: Returns the Crawford score for the Spanish text. Returns an estimate of the years of schooling required to understand the text. The score is only valid for elementary school level texts. Further reading on this blog post.
rs_dale_chall_readability_score1_en	rs	Textstat function: Different from other tests, since it uses a lookup table of the most commonly used 3000 English words. Thus it returns the grade level using the New Dale-Chall Formula for English sentences. Further reading on Wikipedia. The estimate of the Dale-Chall readability score is calculated as: .. math:: (0.1579*%\ difficult\ words)+(0.0496*avg\ words\ per\ sentence) If the percentage of difficult words is > 5, 3.6365 is added to the score.

Feature	Feature Group	Description
rs_dale_chall_readability_score2_en	rs	Textstat function: Different from other tests, since it uses a lookup table of the most commonly used 3000 English words. Thus it returns the grade level using the New Dale-Chall Formula for English sentences. Further reading on Wikipedia. asl = total_no_of_words / count_of_sentences pdw = (self.difficult_words(text) / total_no_of_words) * 100 raw_score = 0.1579 * (pdw) + 0.0496 * asl adjusted_score = raw_score if raw_score > 0.05: adjusted_score = raw_score + 3.6365
rs_difficult_words_en	rs	Textstat function: easy words are defined as words with 2 syllables or less. difficult words are defined as words with 3 syllables or more.
rs_fernandez_huerta_es	rs	Textstat function: Reformulation of the Flesch Reading Ease Formula specifically for Spanish. The results can be interpreted similarly. Further reading on this blog post.
rs_flesch_kincaid_grade_en	rs	Textstat function: Returns the Flesch-Kincaid Grade of the given English text. This is a grade formula in that a score of 9.3 means that a ninth grader would be able to read the document. Further reading on Wikipedia.
rs_flesch_reading_ease_multi	rs	Textstat function: Returns the Flesch Reading Ease Score, Used for ['en', 'de', 'es', 'fr', 'it', 'nl', 'ru']. Further reading on Wikipedia.
rs_fre_multi1	rs	stats_word_count * 'rs_flesch_reading_ease_multi' Used for ['en', 'de', 'es', 'fr', 'it', 'nl', 'ru'].
rs_fre_multi2	rs	stats_syllable_count * rs_flesch_reading_ease_multi Used for ['en', 'de', 'es', 'fr', 'it', 'nl', 'ru'].
rs_gulpease_index_it	rs	Textstat function: Returns the Gulpease index of Italian text, which translates to level of education completed. Lower scores require a higher level of education to read with ease. Further reading on Wikipedia.
rs_gunning_fog_en	rs	Textstat function: Returns the FOG index of the given text. This is a grade formula in that a score of 9.3 means that a ninth grader would be able to read the document. Further reading on Wikipedia.
rs_gutierrez_polini_es	rs	Textstat function: Returns the Guttiérrez de Polini understandability index for Spanish text. Specifically designed for the texts in spanish, not an adaptation. Conceived for grade-school level texts. Scores for more complex text are not reliable. Further reading on this blog post.
rs_linsear_write_formula_en	rs	Textstat function: Returns the grade level using the Linsear Write Formula for English text. This is a grade formula in that a score of 9.3 means that a ninth grader would be able to read the document. Further reading on Wikipedia.
rs_lix1	rs	stats_word_count * rs_lix_all
rs_lix2	rs	stats_syllable_count * rs_lix_all

Feature	Feature Group	Description
rs_lix_all	rs	Textstat function: The estimate of the LIX score is calculated as: .. math:: LIX = A/B + A*100/C A= Number of words B= Number of sentences C= Number of long words (More than 6 letters) `A` is obtained with `len(text.split())`, which counts contractions as one word. `A/B` is calculated using the method `textstat.avg_sentence_length()`, which counts contractions as two words, unless `__rm_apostrophe` is set to False. Therefore, the definition of a word is only consistent if you call `textstat.set_rm_apostrophe(False)` before calculating the LIX. Used for all languages.
rs_lix_de	rs	Used for German only.
rs_rix1	rs	stats_word_count * rs_rix_all
rs_rix2	rs	stats_syllable_count * rs_rix_all
rs_rix_all	rs	Textstat function: The estimate of the RIX score is calculated as: .. math:: rix = LW/S LW= Number of long words (i.e. words of 7 or more characters) S= Number of sentences Anderson (1983) specifies that punctuation should be removed and that hyphenated sequences and abbreviations count as single words. Therefore, make sure to call `textstat.set_rm_apostrophe(False)` before calculating the RIX. Used for all languages.
rs_spache_readability_en	rs	Textstat function: Returns grade level of english text. Intended for text written for children up to grade four. Further reading on Wikipedia.
rs_szigriszt_pazos_es	rs	Textstat function: Adaptation of Flesch Reading Ease formula for Spanish-based texts. Attempts to quantify how understandable a text is. Further reading on this blog post.
rs_text_standard_en	rs	Textstat function: Based upon all the above tests, returns the estimated school grade level required to understand the text. Optional float_output allows the score to be returned as a float. Set to true.
rs_wiener_sachtextformel_de	rs	Textstat function: Returns a grade level score for the given text. A value of 4 means very easy text, whereas 15 means very difficult text. Further reading on Wikipedia.
stats_avg_char_per_word	stats	Textstat function: Average characters per word from cleaned text.
stats_avg_chars_word	stats	Average characters per word from cleaned text (redundant with stats_avg_char_per_word).

Feature	Feature Group	Description
stats_avg_letter_per_word	stats	Textstat function: Average letter per word (not special characters) from cleaned text.
stats_avg_syllables_word	stats	Average syllables in sentence from cleaned text (redundant with stats_syllable_avg).
stats_char_count	stats	Textstat function: Count of all characters from the original text.
stats_comma_count	stats	Count of all commas in a sentence from the original text.
stats_equalsign_count	stats	Count of all equal signs in a sentence from the original text.
stats_file_description	stats	Detect if text is a file description or name: 0 or 1.
stats_formula_description	stats	Detect if text is a formula description: 0 or 1.
stats_frac_description	stats	Detect if text is a frac description: 0 or 1.
stats_frac_long_word	stats	stats_long_word_count / stats_word_count
stats_frac_mini_word	stats	stats_mini_word_count / stats_word_count
stats_frac_monosyllable	stats	stats_monosyllab_count / stats_word_count
stats_frac_polysyllable	stats	stats_polysyllab_count / stats_word_count
stats_frac_word_comma	stats	stats_comma_count / stats_word_count
stats_image_description	stats	Detect if text is an image description or name: 0 or 1.
stats_language_code	stats	Ftlangdetect function: to detect language ISO code.
stats_language_no	stats	Language ISO code transformed via label encoder.
stats_letter_count	stats	Textstat function: Count letters from the cleaned text.
stats_lexical_diversity	stats	Number of unique words / all words
stats_long_numbers_count	stats	Count number of numbers > 4 digits
stats_long_word_count	stats	Count number of words >= 15 characters
stats_lrb_count	stats	Count appearance of -LRB-
stats_max_chars_word	stats	Longest word (max) in sentence
stats_max_syllables_word	stats	Max syllables (of a word)
stats_mini_word_count	stats	Count number of words <= 3 characters
stats_monosyllab_count	stats	Textstat function: count appearance of monosyllables
stats_polysyllab_count	stats	Textstat function: count appearance of polysyllables
stats_reading_time	stats	Textstat function: Milliseconds per char set to 14.69
stats_rrb_count	stats	Count appearance of -RRB-
stats_sentence_length	stats	Textstat function: Average sentence length from cleaned text.
stats_syllable_avg	stats	Textstat function: Average syllables in sentence from cleaned text.
stats_syllable_count	stats	Textstat function: count of syllables in sentence
stats_word_count	stats	Textstat function: lexicon count of words
w2v_ft_sen2vec_mean	w2v	np.mean(model.get_sentence_vector(sentence_list)) for English sentences

Feature	Feature Group	Description
w2v_ft_word2vec_mean	w2v	np.mean(np.mean([model.get_word_vector(word) for word in sentence_list])) for English sentences
w2v_gen_avg_word2vec_mean	w2v	np.mean(np.mean([model.wv[word] for word in sentence_list if word in model.wv.index_to_key] for English sentences

APPENDIX 2 - 3D Plots for Sensitivity Analysis