COMP1804 – Applied Machine Learning Coursework

26-03-2024

Word Count: 2730

Task 1

0. Executive Summary

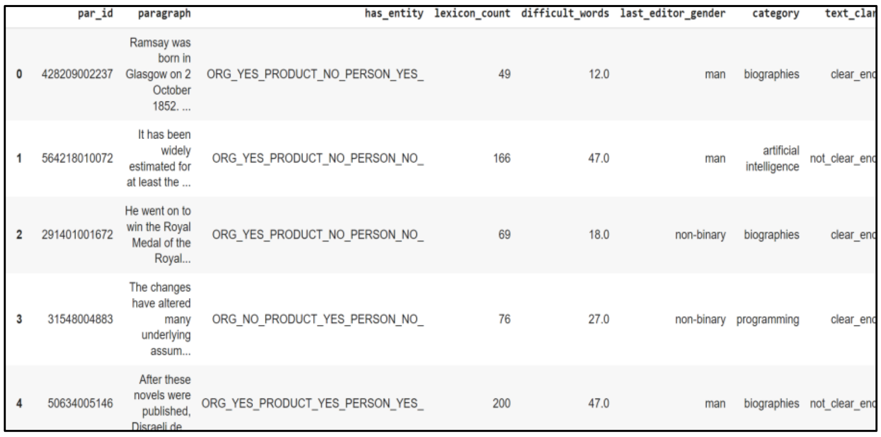
This report outlines the potential application of machine-learning techniques in discerning paragraph topics effectively for the non-profit organization NotTheRealWikipedia. This analysis entails a comprehensive investigation into the feasibility of using machine learning to identify paragraph topics and clarity.

Various data exploratory techniques were utilized to gain insights into the dataset and prepare for model development. Among the machine learning algorithms explored Multinomial Logistic Regression was selected for its ability to handle multiclass classifications efficiently in Task 1. Performance metrics were rigorously employed and explained to evaluate the model's effectiveness.

For Task 2, manual labeling of 100 data points was conducted using conditions on input features. Data preprocessing involved steps such as data cleaning, Label Encoding, and TF-IDF vectorization. Multiple machine learning algorithms were utilized to achieve optimal accuracy. Comprehensive performance evaluations were conducted, leading to insightful conclusions and considerations based on client specifications.

1. Data Exploration and Assessment

This section includes the steps involved in data preprocessing and utilizing exploratory analysis techniques to gain insights into data characteristics. Various data preprocessing techniques were utilized to scrutinize the dataset through different analytical lenses and lay the groundwork for informed model training and development.



	par_id	paragraph	has_entity	lexicon_count	difficult_words	last_editor_gender	category	text_clar
0	428209002237	Ramsay was born in Glasgow on 2 October 1852. ...	ORG_YES_PRODUCT_NO_PERSON_YES_	49	12.0	man	biographies	clear_enc
1	564218010072	It has been widely estimated for at least the ...	ORG_YES_PRODUCT_NO_PERSON_NO_	166	47.0	man	artificial intelligence	not_clear_enc
2	291401001672	He went on to win the Royal Medal of the Royal...	ORG_YES_PRODUCT_NO_PERSON_NO_	69	18.0	non-binary	biographies	clear_enc
3	31548004883	The changes have altered many underlying assum...	ORG_NO_PRODUCT_YES_PERSON_NO_	76	27.0	non-binary	programming	clear_enc
4	50634005146	After these novels were published, Disraeli de	ORG_YES_PRODUCT_YES_PERSON_YES_	200	47.0	man	biographies	not_clear_enc

Fig 1: Imported dataset.

Scatter Plots of Dataset Features

We created scatter plot diagrams for both Tasks 1 and 2, to depict various input features on a two-dimensional axis. The plots help in visually exploring the relationship and correlation between various input parameters. An analysis of various dataset parameters can be achieved through scatter plots diagrams.

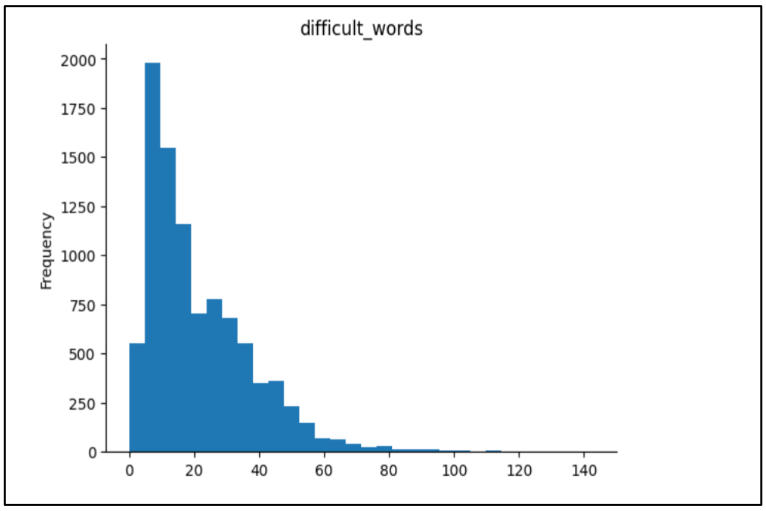


Fig 2: Scatter plot for 'difficult_words'

The scatter diagram for the parameters 'difficult_words' and 'lexicon_counts' visualizes the distribution and potential patterns of numerical data within the dataset.

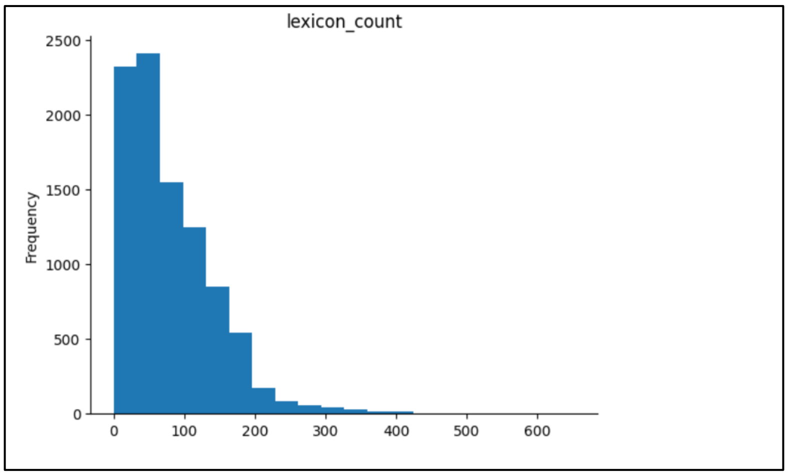


Fig 3: Scatter plot for 'lexicon_count'

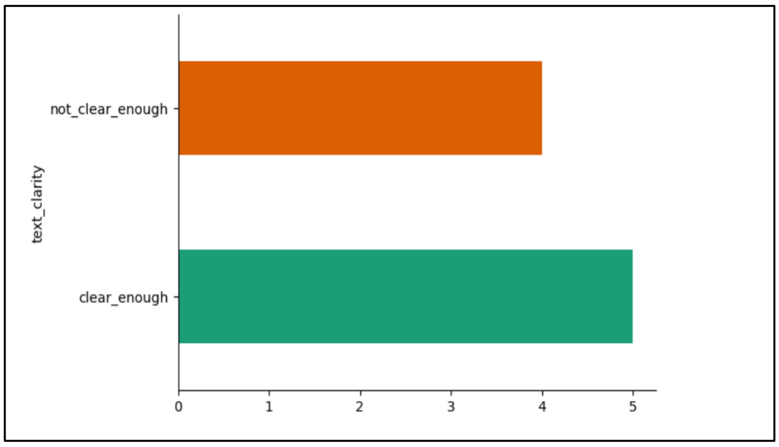
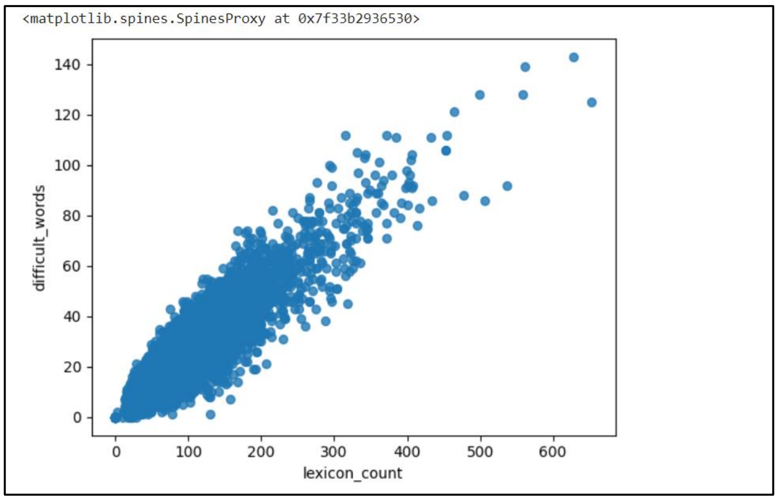


Fig 4: Scatter plot for output 'text_clarity'

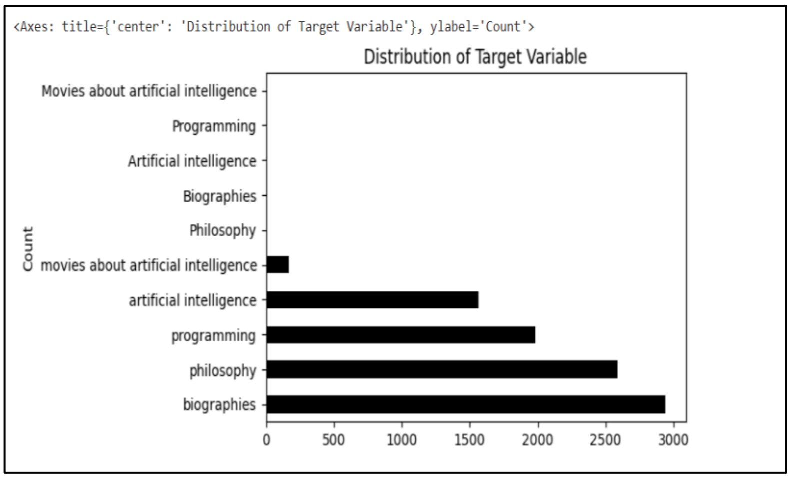
In the 'text_clarity' feature the binary values 'clear_enough' and 'not_clear_enough' are depicted in a horizontal bar chart.



```
<matplotlib.spines.SpinesProxy at 0x7f33b2936530>
```

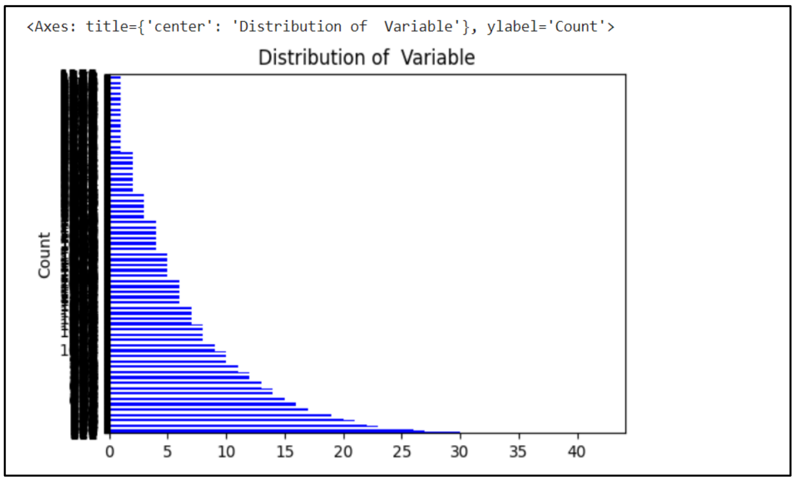
Fig 5: Scatter plot depicting relationship between features.

We create a scatter plot to visualize the relationship between the two parameters: ‘lexicon_count’ and ‘difficult_words’ in the dataset. As illustrated, the instances of these parameters show a clear uptrend from left to right indicating a linear relationship.



```
<Axes: title={'center': 'Distribution of Target Variable'}, ylabel='Count'>
```

Fig 6: Distribution of target variable ‘category’



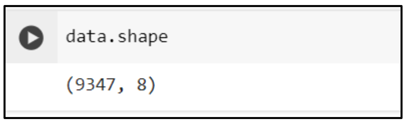
```
<Axes: title={'center': 'Distribution of  Variable'}, ylabel='Count'>
```

Fig 7: Distribution of input feature ‘paragraph’

We utilized the **.apply**(len) function to compute the length of each element in the ‘paragraph’ column and then visualized the distribution of these lengths. This helps identify missing or empty values for the feature offering insights into data integrity.

Statistical Summary of the Dataset

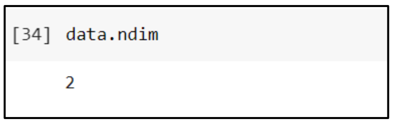
The subsequent phase of our exploratory assessment includes generating a comprehensive statistical summary of the dataset. The summary will assist us in gaining a deeper understanding of the dataset’s characteristics and properties.



```
data.shape

(9347, 8)
```

Fig 8: Shape of the dataset

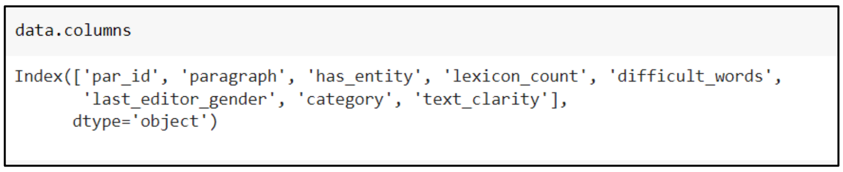


```
[34] data.ndim

     2
```

Fig 9: Dimension of the dataset

We utilize the shape function in pandas to retrieve a tuple representing the dimensions of the dataset: 9347 rows and 8 columns while the **ndim** function displays the two-dimensionality of the dataset.

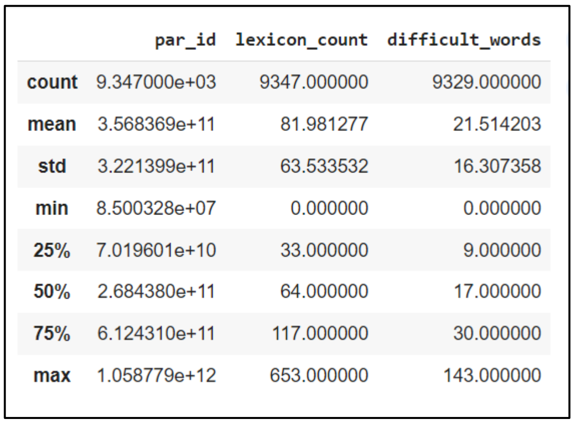


```
data.columns

Index(['par_id', 'paragraph', 'has_entity', 'lexicon_count', 'difficult_words',
       'last_editor_gender', 'category', 'text_clarity'],
      dtype='object')
```

Fig 10: Statistical summary of dataset columns

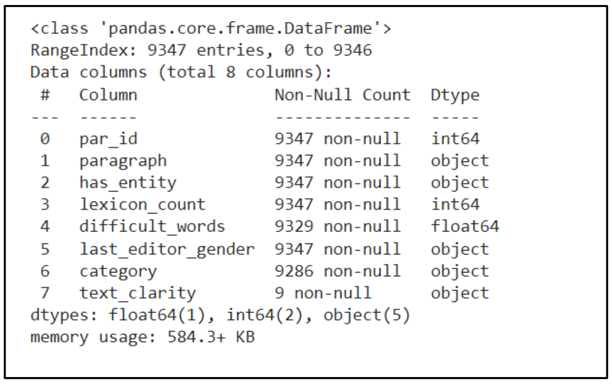
We also use the columns function to retrieve a list of all the columns present in the dataset. In this instance, this attribute returns all the 8 column labels of the dataframe as depicted above.
The pandas library **describe()** provides a statistical summary of the dataset. This summary includes various numerical and descriptive statistics that help us understand the distribution and properties of the dataset. The library provides the following statistics regarding the given dataset:



	par_id	lexicon_count	difficult_words
count	9.347000e+03	9347.000000	9329.000000
mean	3.568369e+11	81.981277	21.514203
std	3.221399e+11	63.533532	16.307358
min	8.500328e+07	0.000000	0.000000
25%	7.019601e+10	33.000000	9.000000
50%	2.684380e+11	64.000000	17.000000
75%	6.124310e+11	117.000000	30.000000
max	1.058779e+12	653.000000	143.000000

Fig 11: Statistical summary of the dataset

The **data.info()** function provides a comprehensive overview of the dataset's structure and properties, In this instance, the function yields a concise summary of our dataset. Additionally, the memory usage for our dataset is reported to 584.3+KB.



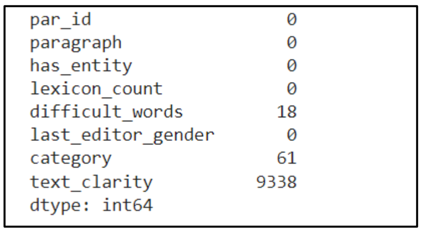
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9347 entries, 0 to 9346
Data columns (total 8 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   par_id              9347 non-null   int64
 1   paragraph           9347 non-null   object
 2   has_entity          9347 non-null   object
 3   lexicon_count       9347 non-null   int64
 4   difficult_words     9329 non-null   float64
 5   last_editor_gender  9347 non-null   object
 6   category            9286 non-null   object
 7   text_clarity        9 non-null      object
dtypes: float64(1), int64(2), object(5)
memory usage: 584.3+ KB
```

Fig 13: Statistical overview of the dataset

2. Data splitting and cleaning

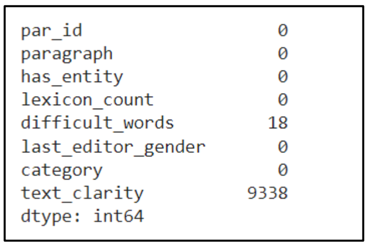
Imputing Missing Values

The subsequential step in our data analysis involves addressing the missing values within the dataset. We utilize the 'pandas' **'isnull()'** function to determine the count of null values in our dataset in both Tasks 1 and 2:



```
par_id                 0
paragraph              0
has_entity             0
lexicon_count          0
difficult_words       18
last_editor_gender     0
category              61
text_clarity        9338
dtype: int64
```

Fig 14: Before imputing null values



```
par_id                 0
paragraph              0
has_entity             0
lexicon_count          0
difficult_words       18
last_editor_gender     0
category               0
text_clarity        9338
dtype: int64
```

Fig 15: After imputing null values

Specifically, for Tasks 1 and 2, the input features consist of 'paragraph' and 'has_entity' while the topic to be predicted is 'category'. While the input features exhibit no null values, the 'category' column has 61 null values that are imputed using the **mode()** function to replace null values with the most commonly occurring values.

Checking and dropping duplicates

The next step of EDA involves examining the dataset for duplicates and dropping them if any. We use the pandas **drop_duplicates()** function for identifying and dropping any duplicates, specifying the subset of columns in the dataset. This step is performed for both Tasks 1 and 2. Before identifying and dropping duplicates, we convert our concerned data series into lowercase using the **.lower()** function. This ensures that the text is uniformly formatted in lowercase, facilitating consistent analysis.

Reading and Splitting Data

Next, we move on to the model-building and training phase. We define variables X and Y where: X contains the input features (‘paragraph’ and ‘has_entity’) used for training the model and ‘Y’ contains the target variable (‘category’) that the model aims to predict. We use the **‘train_test_split()’** function from the **scikit** library and specify three parameters to split the data:

Train_test_split() parameters	Rationale
Test_size=0.2	splits the dataset into training and test data in the ratio 80:20 respectively.
Random_state=42	sets the random seed for reproducibility.
Shuffle=True	shuffles data before splitting, to ensure that data points are randomly distributed between training and testing sets.

Fig 16: Dataset splitting parameters.

3. Data Encoding

The input features within our dataset contain categorical values. However, since machine learning models exclusively accept numerical values, encoding becomes essential. We utilized **LabelEncoder()** class from Scikit Learn to convert categorical values into numerical presentations, facilitating the training and processing of data within machine learning algorithms.

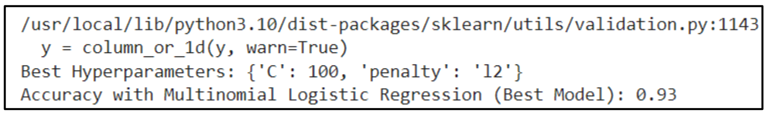
Furthermore, we perform text vectorization to improve the efficiency of categorical machine learning models. The **TF-IDF** text vectorization technique further enhances the effectiveness of text encoding by measuring the relevance of words not just within individual components, but also against the entire dataset (Goyal, 2021).

The data encoding and text vectorization steps are executed consistently across both Tasks 1 and 2. This ensures uniformity in the preprocessing pipeline and accurate modeling.

4. Task 1: topic classification

4a. Model Building

The final step in our model-building process involves training the model using a suitable machine-learning algorithm and evaluating its accuracy. In this task, we aim to classify input features to discern the topics of paragraphs containing text. Fundamentally, this task constitutes a classification problem, and to address it, we opted for the Multinomial Logistic Regression algorithm.



```
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:1143
  y = column_or_1d(y, warn=True)
Best Hyperparameters: {'C': 100, 'penalty': 'l2'}
Accuracy with Multinomial Logistic Regression (Best Model): 0.93
```

Fig 17: Implementing Hyperparameter optimization.

To further enhance our model's performance, we implemented hyperparameter optimization using **GridSearchCV**. This involves fine-tuning the key parameters to perform an exhaustive search for optimal parameters (Pandian, 2022). The table below elucidates the list of hyperparameters integrated into the model and provides a rationale for their significance:

Hyperparameter	Value	Explanation
C	100	This parameter helps prevent overfitting promoting the model to fit training data more closely.
Penalty	l2	'l2' penalty and potentially reduces overfitting by adding squared coefficients magnitude to the loss function.
Multi_class	Multinomial	Suitable for multiclass classification which indicates that the model uses a softmax function to predict probabilities.
Solver	lbgfs	The 'lbgfs' solver provides efficiency in optimization and is well-suited for multiclass logistic regression.
Random_state	42	Setting this parameter ensures that the results are reproduced across different runs of the model.

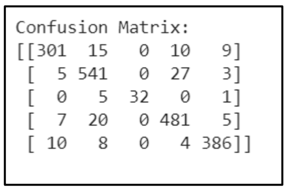
Fig 17: List of Hyperparameters employed.

4b. Model Evaluation

We proceed to the model evaluation stage, where we employ several performance metrics to comprehensively assess the model's performance. In addition to accuracy, we delve into deeper insights using evaluation metrics such as the confusion matrix and classification report and comparing our model's performance against a trivial baseline.

Accuracy: Stands as our primary metric, providing an effective measure of the model's correctness. After retrieving the best hyperparameters, the accuracy achieved by our best model is **0.93**, underscoring the overall proficiency.

Confusion Matrix: The confusion matrix serves as a comprehensive summary of the model's performance in various classes, where each row in the matrix represents the actual class labels while each column corresponds to the predicted classes.

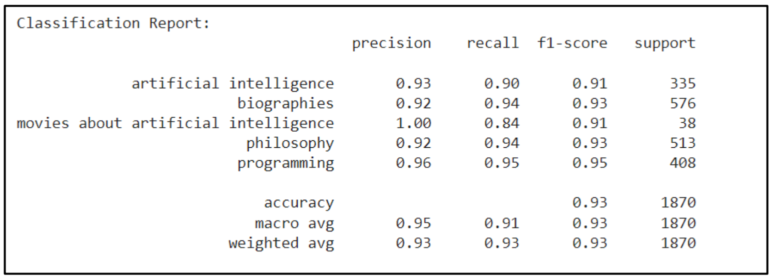


```
Confusion Matrix:
[[301  15   0  10   9]
 [  5 541   0  27   3]
 [  0   5  32   0   1]
 [  7  20   0 481   5]
 [ 10   8   0   4 386]]
```

Fig 17: Confusion Matrix

The diagonal elements from top-left to bottom-right represent the number of correct predictions for each class and the confusion matrix shows that our model excels well in making correct predictions across multiple classes.

Classification Report: Provides a detailed summary (Zach, 2022) of the model's performance for each class in the dataset, along with other metrics:



```
Classification Report:
                                   precision    recall  f1-score   support

           artificial intelligence       0.93      0.90      0.91       335
                       biographies       0.92      0.94      0.93       576
movies about artificial intelligence       1.00      0.84      0.91        38
                        philosophy       0.92      0.94      0.93       513
                       programming       0.96      0.95      0.95       408

                          accuracy                           0.93      1870
                         macro avg       0.95      0.91      0.93      1870
                      weighted avg       0.93      0.93      0.93      1870
```

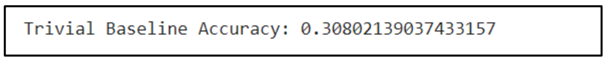
Fig 18: Classification Report

The table below summarizes the metrics displayed in the classification report along with their rationale:

Classification Report Metrics	Model performance and explanation [Value*100%]
Precision	Measures the proportion of true positive predictions out of all positive predictions. The model demonstrates high accuracy in correctly predicting positive instances across various classes.
Recall	Indicates our model's ability to correctly identify positive instances. The model shows significant effectiveness in predicting true positive instances across all classes.
F1-Score	Indicates the harmonic mean of precision and recall. The F1-Score shows a good balance between Precision and Recall.
Support	Refers to the number of actual occurrences of each class in the dataset.
Accuracy	Indicates the overall correctness of the model. Our model shows 93% accuracy in predicting classes correctly.
Macro Average	The model performs well across all classes, giving equal weight to each class.
Micro Average	The model performs consistently well across all classes, regardless of class imbalance.

Fig 19: Performance metrics from the Classification report

Comparison with one trivial baseline: We implemented a trivial baseline model using DummyClassifier with strategy set to 'most_frequent' as shown below:



```
Trivial Baseline Accuracy: 0.30802139037433157
```

Fig 20: Trivial Baseline Accuracy

The trivial baseline accuracy of 0.308 indicates that if we predict the majority class in the test set using the 'most_frequent' strategy, we will achieve an accuracy of 30.8%.

Comparing this baseline with the accuracy of our current model, which is 0.93, reveals that our model outperforms the trivial baseline, and it correctly predicts the class for approximately 93% of the samples in the test set.

4c. Task 1 Conclusions

- Based on client requirements, the model excels in making correct predictions to identify the paragraph topic. Our model shows exceptional consistency in correctly predicting positive instances of all classes and performs well across all performance metrics and evaluation criteria.
- In addition to the standard performance metrics, the client should consider using **Coehn's Kappa coefficient** (Datatab, 2024) as an additional scalar performance metric to keep track of the algorithm's performance. Coehn's Kappa coefficient evaluates the level of agreement between two raters utilizing the same approach, providing a measure of interrater reliability. By incorporating Coehn's Kappa coefficient, the client can obtain a more comprehensive understanding of the model's effectiveness.

5. Task 2: text clarity classification prototype

5a. Ethical discussion

In Task 2, the client's aim to build an algorithm that automatically detects the clarity of a paragraph and rejects it if they are unclear entails certain ethical concerns:

Data Bias: The training data might reflect the writing styles of creators and lead to biases against writers with different styles.
Transparency and Accountability: Automatic rejection may lead to a lack of clarity for users who do not understand the reason for rejection.
Performance and Reliability: The accuracy of the algorithm depends on the quality of training data and poor accuracy may lead to unfair rejections.
Impact on communities and individuals: Unless trained properly, the algorithm may reject certain dialects and language styles leading to impacts on communities.

Certain measures could be undertaken to mitigate these risks (Ghosh, 2023):

Data training and diversity: Training the algorithm on a diverse dataset that reflects different writing styles and backgrounds and is free from biases.
Transparency: Clearly defining the criteria for text clarity for users and algorithm.
Human Oversight: Combining algorithm decisions with human reviews to ensure fairness.
Regular Monitoring: Conducting regular audits to evaluate the performance and impact of the algorithm.

5b. Data Labelling

We manually labeled 100 datapoints based on the presence of entities in the ‘has_entity’ feature. This feature contains information about the presence of entities related to three topics: ‘ORG’, ‘PRODUCT’, and ‘PERSON’. Each paragraph is associated with a label indicating the presence or absence of these entities, such as ‘ORG_YES_PRODUCT_NO_PERSON_YES’. Our labeling strategy involved considering paragraphs ‘not_clear_enough’ if any two entities had a ‘YES’ label. Additionally, if only one entity had a ‘YES’ label, the paragraph was labeled as ‘clear_enough’. Conversely, if all three entities had ‘YES’ labels or if all of them had ‘NO’ labels, the paragraph was labeled as ‘not_clear_enough’.

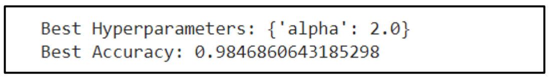
5c. Model building and evaluation

In the final stage, we implemented a machine-learning algorithm to build our model. After testing both Logistic regression and Naïve-Bayes algorithms, we opted for Naïve-Bayes due to its superior performance (Ottesen, 2017).

Algorithm	Rationale
Naïve Bayes	We find the Naïve-Bayes algorithm performs well with categorical data and proves to be versatile and powerful for our classification task. Naïve-Bayes proves to be more computationally efficient than Logistic Regression and is simple to use.

Fig 22: Naïve Bayes algorithm rationale

We further implemented hyperparameter tuning for the algorithm, defining a grid of hyperparameters including the smoothing parameter ‘alpha’ and ‘GridSearchCV’.



```
Best Hyperparameters: {'alpha': 2.0}
Best Accuracy: 0.9846860643185298
```

Fig 22: Hyperparameter tuning.

The best combination is determined by the highest cross-validation score. The 'alpha' values ranging from 0.1 to 2.0 are tested and the best-performing value and score are identified as depicted below:

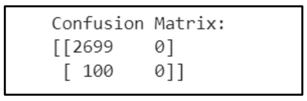
Parameter Alpha	Mean_test_score
2.0	0.984686
1.0	0.984533
0.5	0.984380
0.1	0.981930

Fig 22: Hyperparameter tuning values

We proceed to the model evaluation, which is a crucial step to assess the effectiveness of the trained model. We consider several performance metrics to assess our model's performance.

Accuracy: The primary metric for evaluating performance. After hyperparameter tuning, the accuracy achieved by the best model stands at **0.98**.

Confusion Matrix: Provides a comprehensive summary of the model's performance:

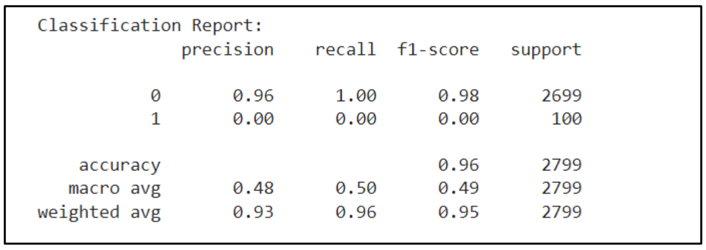


```
Confusion Matrix:
[[2699    0]
 [ 100    0]]
```

Fig 23: Confusion Matrix

The model correctly predicted the negative class for 2699 instances and 100 false negatives but failed to predict any instances belonging to the positive class.

Classification Report: Provides a detailed summary of the model's performance for each class:



```
Classification Report:
              precision    recall  f1-score   support

           0       0.96      1.00      0.98      2699
           1       0.00      0.00      0.00       100

    accuracy                           0.96      2799
   macro avg       0.48      0.50      0.49      2799
weighted avg       0.93      0.96      0.95      2799
```

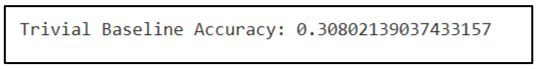
Fig 24: Classification Report.

The report is further explained in the following table:

Metric	Performance
Precision	Measure of true positive predictions. For class 1, the precision is 0.96 while for class 1 it is 0.
Recall	The proportion of actual positive instances correctly predicted: 1 for class 0 and 0 for class 1.
F1-Score	The harmonic mean of Precision and Recall: for class 0 it is 0.98 while for class 1 it is 0.
Support	Indicates the total number of occurrences of class in the dataset.

Fig 25: Classification report performance metrics

Comparison with one trivial baseline: Implemented comparison with a majority class baseline:

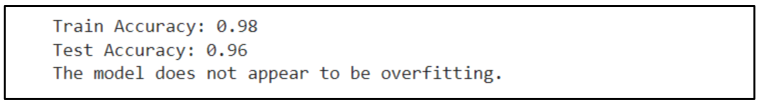


```
Trivial Baseline Accuracy: 0.30802139037433157
```

Fig 26: Comparison with Majority Baseline

The majority baseline accuracy of 0.308 indicates that correct prediction of the majority class occurs 30.8% of the time. This serves as a reference point for evaluation of our model which only predicts the true negatives 98.4% of the time.

Check for Ovefitting: Comparing the accuracy of the model on training and test datasets to check for overfitting.



```
Train Accuracy: 0.98
Test Accuracy: 0.96
The model does not appear to be overfitting.
```

Fig 27: Comparing training and test data to check for overfitting.

The difference between training and test data accuracy is less than the threshold (0.05), which indicates that the model does not seem to be overfitting.

5d. Task 2 Conclusions:

- The model seems successful according to the client's definition of success, as it achieves high accuracy on both the training (98%) and test (96%) datasets. We compared the model performance based on training data versus test data which performs equally well on both training and test data, aligning with client expectations.
- Given the class imbalance in the dataset, where the positive class has significantly fewer samples than the negative class, I would recommend the client use the F1-score as an additional scalar metric to keep track of the algorithm's performance.
- For improvements, the top suggestion would be to address the imbalance in the dataset by exploring techniques such as oversampling the minority class or undersampling the majority class.

6. Self-reflection

The model-building section for Task 2, can be enhanced by optimizing hyperparameters and addressing data imbalance through undersampling and balancing distribution. Additionally, attempts to use BERT and DISTILBERT natural language processing (NLP) models as advanced techniques were hindered by RAM limitations in the evaluation stage which served as a drawback.

7. References

Datatab. (2024) *Cohen's Kappa.* Available at: https://datatab.net/tutorial/cohens-kappa (Accessed: 22 March 2024).

Ghosh, Paramita (2023) *Top Ethical Issues with AI and Machine Learning.* Available at: https://www.dataversity.net/top-ethical-issues-with-ai-and-machine-learning/ (Accessed: 25 March 2024).

Goyal, Chirag (2021) *Part 5: Step by Step Guide to Master NLP – Word Embedding and Text Vectorization.* Available at: https://www.analyticsvidhya.com/blog/2021/06/part-5-step-by-step-guide-to-master-nlp-text-vectorization-approaches/ (Accessed: 22 March 2024).

Ottesen, Christopher (2017) *Comparison between Naïve Bayes and Logistic Regression.* Available at: https://dataespresso.com/en/2017/10/24/comparison-between-naive-bayes-and-logistic-regression/ (Accessed: 24 March 2024).

Pandian, Shanthababu (2022) *A Comprehensive Guide on Hyperparameter Tuning and its Techniques.* Available at: https://www.analyticsvidhya.com/blog/2022/02/a-comprehensive-guide-on-hyperparameter-tuning-and-its-techniques/ (Accessed: 24 March 2024).

Zach. (2022) *How to Interpret the Classification Report in sklearn (With Example).* Available at: https://www.statology.org/sklearn-classification-report/ (Accessed: 22 March 2024).