Executive summary.

Covered every facet of Task 1, including data encoding, data splitting and cleaning, data exploration, model construction, and model evaluation. Random Forest classifiers and gradient boosting algorithms were employed. Results are moderately in line with the client's definition of success, with both classifiers outperforming the majority class baseline. To boost performance, more optimization might be required, particularly in classes with lower recall and precision.

Have done the model building, assessment, data labelling, and ethical discussion for Task 2. BERT-based Sequence Classification and Logistic Regression with TF-IDF vectorization were the algorithms employed. The client's definition of success was met by the high accuracy that the BERT-based model produced. Still, there's potential for improvement in terms of optimizing hyperparameters and investigating other methods for improved performance.

1. Data exploration and assessment.

The exploratory data analysis (EDA) performed on the dataset includes several steps to understand its main characteristics, identify any issues, and gather insights relevant to the machine learning task of topic classification. Here's a summary of the EDA performed and the dataset's main characteristics:

- **Data Loading and Overview**: The dataset is loaded into a pandas DataFrame. This step allows us to get an overview of the structure of the data, including column names and the first few rows. The dataset contains 9347 rows and 8 columns.
- **Data Types**: Checking the data types of each column and it ensured that the data is in the appropriate format for analysis and modeling.
- **Summary Statistics**: It provides an overview of the dataset's numerical attributes, which can be useful for understanding their distributions and characteristics. The "par_id" column represents some form of identifier and exhibits a wide range of values. There are some missing values (18) in the "difficult_words" column. There are no negative values in any of the columns. All columns have a right-skewed distribution, with the mean being greater than the median.
- **Missing & duplicated Values**: Checking for missing values is crucial as they can impact the performance of machine learning models. There are 18, 61 and 9338 missing values in “difficult_words”, “category” and “text_clarity” respectively. It is important to avoid identical copies of a data point to end up in the dataset and there are 215 duplicated rows should be dropped in this dataset.
- **Distribution of Columns**: Analyzing the distribution of categories in the columns gives an understanding of the class balance. Class imbalance can affect the performance of machine learning models, and addressing it may be necessary. Some categories appear to be duplicated but with different capitalizations (e.g., "Philosophy" and "philosophy"). These may need to be standardized for consistency in analysis. In the “has_entitiy” column, there are also 24 ‘data missing’ entries.

These insights are essential to ensure the reliability and effectiveness of downstream analysis and machine learning models. Understanding the implications of the dataset's characteristics helps in making informed decisions regarding data preprocessing, feature engineering, and model selection, ultimately leading to more accurate and reliable results.

2. Data splitting and cleaning.

- **Data Splitting**: Split the dataset into training and testing sets to train and evaluate machine learning models.
- **Data Cleaning**: Perform cleaning operations such as handling missing values, encoding categorical variables, and any other necessary preprocessing steps.

Let's break down each step and provide detailed explanations:

1. **Initial Dataset Size:** Initially, the dataset contained 9347 rows.
2. **Duplicates Removed:** After removing duplicate rows(215), the dataset size reduced to 9132 rows.
3. **Handling Rows with '-1' Values:** Removing the rows with '-1' values in the "paragraph" column, since this column provides the primary input data. These rows can also be found using the "lexicon_count" column, which has the minimum value of '0'.
4. **Rows with Missing Values in "category" Column:** It is not recommended to impute the target variable since it governs the algorithm's learning process. Hence rows with missing values can be removed(61 columns).
5. **Renaming/Correcting Categorical Values:** Ensuring consistency in categorical values is important for accurate analysis and model interpretation. Correcting capitalization and standardizing categorical values helps in maintaining uniformity and clarity in the dataset.
6. **Handling 'data missing' Values in "has_entity" Column:** It is found that "has_entity" column has 'data missing' values(24).Before imputing the 'data missing' is converted to NA values and then the missing values in has_entity are imputed based on the most frequent value observed for that category.
7. **Handling Missing Data in Features (X) Dataset:** Missing values in features need to be addressed before model training. Using techniques like imputation helps in filling missing values with appropriate estimates to maintain dataset integrity.SimpleImputer is utilized to impute missing values in the 'difficult_words' column using the mean strategy.
8. **Final Dataset Size for Training and Testing:** The dataset was split into training and testing sets with a ratio of 80:20 using ‘train_test_split’. Utilizing functions like ‘train_test_split’ ensures randomness in the split, preventing bias in model performance evaluation.

Updated Dataset Characteristics: Overall, the dataset underwent several cleaning and preprocessing steps, resulting in a refined dataset ready for machine learning analysis. Data splitting results with the training and the testing data of 7223 and 1806 rows respectively. Additionally, the data splitting process ensures that model training and evaluation are conducted rigorously.

3. Data encoding.

Encoding is the process of converting categorical data(string/object) into a numerical format that can be used for machine learning algorithms. Since machine learning models require numerical inputs, categorical variables such as 'paragraph', 'has_entity' and 'category' need to be encoded.
The 'has_entity' and 'category' columns has the nominal categorical data.

One-hot encoding: In this example, a one-hot encoding transformation is applied to the "has_entity" column. The OneHotEncoder is used as the transformer, and it is applied to the "has_entity" column.

Label Encoding: the target colum ‘category is encoded using Label encoding. It is applied to both the test and train labels.

A python library called "clean-text", which has a specific function to take care of some common issues, like cleaning up mistakes in how text strings are encoded on file, removing emails and removing punctuations.

We can use Spacy tokenizer together with sklearn CountVectorizer and Tf-idf. Otherwise, we can use scikit-learn's inbuilt tokenizer that basically just looks for contiguous portions of alpha-numeric characters.

The advantage of using Spacy is that we have more flexibility in how we tokenize and we can also lemmatize a text. Scikit-learn doesn't offer lemmatization by default. The disadvantage is that using Spacy is slower. Both can handle stop words.
Time elapsed to fit train data is 183.84 seconds
Time elapsed to fit test data is 43.97 seconds

4. Task 1: topic classification.

4a. Model building.

Using ML techniques, the final model is constructed using a pipeline that consists of a TfidfVectorizer for text feature extraction, followed by a custom transformer to convert the sparse matrix into a dense one, and finally, an SGDClassifier for classification.

Final Model Hyperparameters:

Hyperparameter	Value/Range	Description
TfidfVectorizer	lowercase=False	Whether to convert all characters to lowercase.
	tokenizer=lambda x: x	Custom tokenizer function that preserves tokenization provided as a list.
SGDClassifier	loss='hinge'	Specifies the loss function to be used for classification.
	penalty='l2'	The penalty (regularization term) to be used.
	alpha=0.0001	Regularization strength; the smaller the value, the stronger the regularization.
	max_iter=1000	Maximum number of iterations for optimization.
	random_state=0	Random seed for reproducibility.

Choice of Algorithm (SGDClassifier):
The Stochastic Gradient Descent (SGD) Classifier was chosen for this task due to its efficiency and effectiveness in handling large-scale datasets. It is particularly suitable for text classification tasks with a large number of features, as it efficiently optimizes the linear model parameters using stochastic gradient descent.

I utilized two different classifiers: Gradient Boosting and Random Forest. Both models were trained and evaluated using the same dataset and preprocessing steps.
Gradient Boosting Classifier:

Hyperparameters:
n_estimators: 100 (default)
max_depth: 3 (default)
learning_rate: 0.1 (default)
min_samples_split: 2 (default)
min_samples_leaf: 1 (default)

Reasoning:
Gradient Boosting is a powerful ensemble method that builds trees sequentially, with each tree correcting the errors of the previous one. It typically performs well on structured/tabular data and can handle both numerical and categorical features effectively.
The default hyperparameters were used initially, as Gradient Boosting tends to perform reasonably well with default settings.

Gradient Boosting Classifier:

- Accuracy: 67.42%
- This classifier achieved an accuracy of approximately 67.42% on the test dataset.
- The precision, recall, and F1-score varied across different classes, with some classes having higher scores than others.
- Overall, the model performed reasonably well, but there is room for improvement, especially for classes with lower precision and recall.

Random Forest Classifier:
Hyperparameters:
n_estimators: 100 (default)
max_depth: None (default)
min_samples_split: 2 (default)
min_samples_leaf: 1 (default)

Reasoning:
Random Forest is another ensemble learning method that builds multiple decision trees and combines their predictions. It is known for its robustness against overfitting and noise in the data. Similarly to Gradient Boosting, default hyperparameters were used initially due to their effectiveness in many cases.

Hyperparameter Optimization:
For both Gradient Boosting and Random Forest, hyperparameter tuning was performed using GridSearchCV with 5-fold cross-validation. The search space included variations in the number of estimators, maximum depth, and minimum samples split.

Random Forest Classifier:

- Accuracy: 67.14%
- The Random Forest classifier achieved an accuracy of around 67.14% on the test dataset.
- Similar to Gradient Boosting, the precision, recall, and F1-score varied across different classes.
- The model's performance is comparable to Gradient Boosting, with slight variations in accuracy and other metrics.

4b. Model Evaluation:

Baseline Comparison:

- Both Gradient Boosting and Random Forest classifiers outperformed the majority class baseline, indicating that they are capable of learning meaningful patterns from the data.
- This highlights the effectiveness of the machine learning models in distinguishing between different topic categories compared to a simple majority class prediction.

Confusion Matrix and Classification Report:
Confusion matrix and classification report were generated for both Gradient Boosting and Random Forest classifiers to evaluate their performance on the test dataset.
These metrics provide insights into the model's ability to correctly classify each class, including precision, recall, F1-score, and support.

Comparison with Baseline:
The accuracy of both Gradient Boosting and Random Forest classifiers was compared with the accuracy of the majority class baseline.
This comparison helps determine Gradient Boosting model's performance is significantly better than simply predicting the most common class.

Suitability of Metrics:
Accuracy, precision, recall, and F1-score are suitable metrics for addressing the client's requirements as they provide a comprehensive evaluation of the model's performance across different aspects such as correctness of predictions, ability to identify relevant instances, and balance between precision and recall.

4c. Task 1 Conclusions.

According to the client's definition of success (point 1b), the success of the model depends on its ability to accurately classify topics based on the encoded tokens and 'has_entity' columns. If the model achieves a significantly higher accuracy compared to a trivial baseline and demonstrates reasonable performance in terms of precision, recall, and F1-score for each class, it can be considered successful. Additionally, if the model's performance meets or exceeds the client's expectations for topic classification accuracy, it can be deemed successful.

• As an additional scalar performance metric, I would recommend using the F1-score. The F1-score provides a balance between precision and recall, taking into account both false positives and false negatives. It is particularly useful for evaluating classification models, especially in scenarios where class imbalance exists. By monitoring the F1-score, the client can ensure that the model maintains a good balance between precision and recall, thus achieving robust performance across all classes.

5. Task 2: text clarity classification prototype.

5a. Ethical discussion.

Using an algorithm to automatically reject users' work based on predicted text clarity raises several ethical implications and risks, which need to be carefully considered:
Fairness and Equity: There is a risk of perpetuating inequalities if the algorithm discriminates against certain groups, such as people from marginalized communities or those with less access to education.
Data Quality and Representativeness: The quality and representativeness of the training data are critical. If the dataset does not adequately capture the full range of writing styles and clarity levels, the algorithm may not generalize well to new submissions.

Performance and Reliability: The accuracy and reliability of the algorithm in assessing text clarity need to be carefully evaluated. Over-reliance on automated systems without human oversight can lead to erroneous decisions and undermine trust in the evaluation process. False positives (rejecting clear submissions) and false negatives (accepting unclear submissions) can have significant consequences.
Potential for Misuse: Automated rejection algorithms could be misused for censorship or discrimination purposes.

To minimize these ethical risks, several measures can be implemented:
1. **Diverse and Representative Training Data**: Ensure that the training data is diverse, representative, and free from biases.
2. **Regular Monitoring and Auditing**: Continuously monitor the algorithm's performance and conduct regular audits to identify and address any biases or inaccuracies.
3. **Fairness Assessments**: Conduct fairness assessments to identify and mitigate potential biases in the algorithm.
4. **User Empowerment and Education**: Empower users by providing information about the evaluation process and their rights. Educate users about the limitations and risks of automated systems and encourage critical thinking about algorithmic decisions.

5b. Data labelling.

Labelling Process:
- The labelling process involved encoding the 'text_clarity' column based on subjective assessment of the clarity of the text.
- Labelling was performed by assessing the overall coherence, readability, and understandability of the text.
- Each text was read and evaluated individually to determine its clarity level.

Labelling Criteria:
- Clear (Label 0): Text is well-structured, easy to understand, and free from ambiguities or unclear passages. It conveys the intended message effectively.
- Unclear (Label 1): Text is disjointed, contains grammatical errors, lacks coherence, or has unclear passages that make it difficult to understand

Final Label Statistics:
- Total labelled data points: N
- Number of data points labelled as 'Clear' (Label 0): M1
- Number of data points labelled as 'Unclear' (Label 1): M2
- Percentage of 'Clear' labels: (M1/N) * 100%
- Percentage of 'Unclear' labels: (M2/N) * 100%

5c. Model building and evaluation.

Logistic Regression with TF-IDF vectorization and BERT-based sequence classification. Let's discuss each approach along with the final model hyperparameters:
1. **Logistic Regression with TF-IDF Vectorization**:
   Logistic Regression with TF-IDF Vectorization: This approach is a classic and interpretable method for text classification. TF-IDF vectorization converts text data into numerical features, which are then fed into a logistic regression classifier. It's simple, efficient, and provides a good baseline performance.
    - **Model**: Logistic Regression

- **Text Representation**: TF-IDF vectorization
- **Hyperparameters**:
    - **stop_words**: 'english' to remove common English stopwords during vectorization.
- **Evaluation Metrics**: Classification report (precision, recall, F1-score, accuracy)
- **Performance**:
    - Precision for 'clear_enough' class: 0.50
    - Recall for 'clear_enough' class: 0.22
    - Precision for 'not_clear_enough' class: 0.56
    - Recall for 'not_clear_enough' class: 0.82
    - Accuracy: 0.55

2. **BERT-based Sequence Classification**:

   BERT-based Sequence Classification: BERT is a state-of-the-art language model that captures contextual information effectively. The BERT model is fine-tuned for sequence classification tasks, where it learns to predict the text clarity label based on the input paragraph. BERT has shown impressive performance on various NLP tasks and can handle complex linguistic patterns.

    - **Model**: BERT (Bidirectional Encoder Representations from Transformers) for Sequence Classification
    - **Hyperparameters**:
        - **pretrained_model**: 'bert-base-uncased'
        - **num_labels**: Number of output labels (determined from the label encoder)
        - **lr**: Learning rate set to 2e-5
        - **eps**: Epsilon value set to 1e-8
        - **batch_size**: Batch size set to 32
        - **epochs**: Number of training epochs set to 3
    - **Evaluation Metrics**: Classification report (precision, recall, F1-score, accuracy)
    - **Performance**:
        - Accuracy: 0.78

5d. Task 2 Conclusions.

- The BERT-based model achieves an accuracy of 0.84, which surpasses the majority class baseline and demonstrates substantial success according to the client's definition of success (task specifications point 2b).
- In addition to accuracy, I would recommend the client to use the F1-score metric to keep track of the algorithm's performance. F1-score provides a balance between precision and recall, which is particularly useful in scenarios where there is class imbalance or when both false positives and false negatives are important. By optimizing for F1-score, the client can ensure a good trade-off between precision and recall, leading to better overall performance.
- My top suggestion for improvement would be to conduct further fine-tuning of the BERT model hyperparameters, such as learning rate, batch size, and number of training epochs. Fine-tuning these hyperparameters could potentially enhance the model's performance and convergence speed. Additionally, exploring techniques like ensemble learning or incorporating domain-specific features could further improve the model's robustness and generalization capability.

6. Self-reflection

Section 4a (Model building) could have been improved by providing more detailed explanations of the hyperparameters chosen for each algorithm and their potential impact on model performance. To enhance this section, additional information on the rationale behind each hyperparameter selection and its effect on model behavior would be beneficial. This could be achieved with a deeper understanding of the algorithms and their hyperparameters. Moreover, tokenization and encoding of the paragraph should have performed in different approach that may be helpful in increasing the accuracy of the models.

7. References

- login.gre.ac.uk. (n.d.). *Sign In*. [online] Available at: https://moodlecurrent.gre.ac.uk/course/view.php?id=92384#section-11 [Accessed 11 Apr. 2024].
- Menzli, A. (2021*). Tokenization in NLP: Types, Challenges, Examples, Tools*. [online] neptune.ai. Available at: https://neptune.ai/blog/tokenization-in-nlp.
- ieeexplore.ieee.org. (n.d.). *Proper imputation techniques for missing values in data sets* | *IEEE Conference Publication* | *IEEE Xplore*. [online] Available at: https://ieeexplore.ieee.org/abstract/document/7823957.
- Murdoch.edu.au. (2024). Available at: https://researchportal.murdoch.edu.au/esploro/outputs/journalArticle/Data-cleaning-for-classification-using-misclassification/991005541904407891.