# REPORT SUMMARY

# Task Summary:

# TASK 1

We created a Multinomial Naive Bayes classifier employing “paragraph” and the "has\_entity" feature for the job of text classification to detect topics like "artificial intelligence", "movies about artificial intelligence", "programming", "philosophy", and "biographies". The model met the client's success criteria with an accuracy of 99.46% on the training data and 93.06% on the testing data. There were very few misclassifications overall, and the client only accepted one particular mistake. The model's performance was better than a simple baseline and showed strong generalization; nevertheless, more research into other performance metrics may yield more insightful results.

# TASK 2

We used a Gradient Boosting Classifier to a subset of the dataset to predict text clarity labels employing the “last\_editor\_person”, “has\_entity”, “paragraph”, and other manually calculated features. With a 75% accuracy rate on test data and 74% accuracy on our training data, our technique shows intriguing potential for automated clarity identification. The model's performance is fairly consistent with the client's definition of success, laying the groundwork for future development and real-world deployment.

# 1. DATA EXPLORATION AND ACCESSMENT

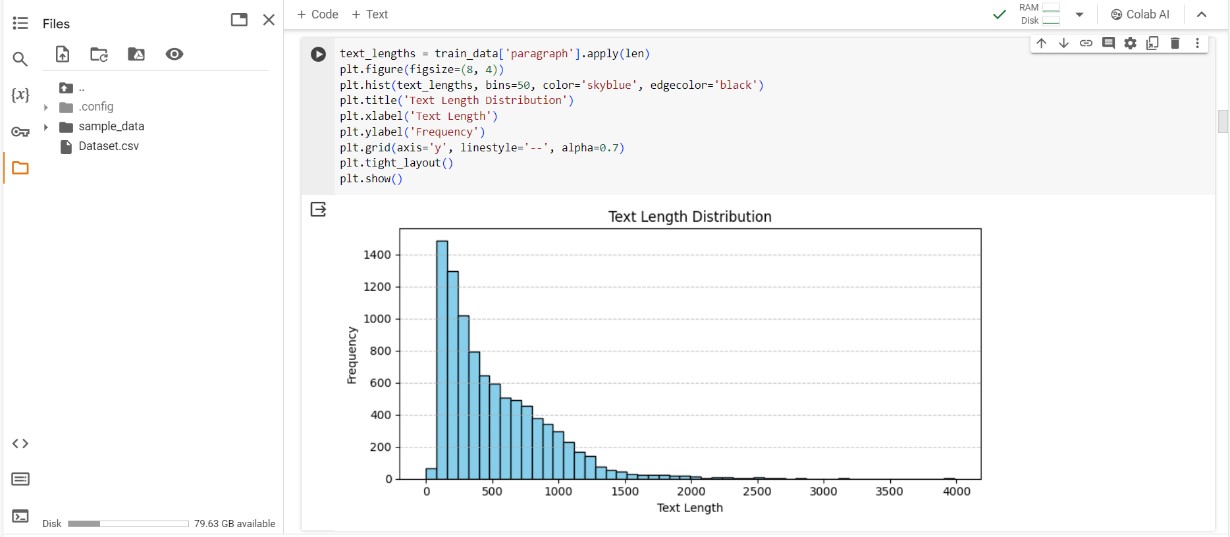
The exploratory data analysis (EDA) concentrated on understanding the dataset's major properties, finding any flaws, and gaining insights pertinent to the machine learning goal of topic categorization.

**Dataset Overview:** The EDA began with an overview of the dataset's structure, including sample count, feature set, and target class. This establishes context for further analysis and model building.

**Class Distribution:** A bar plot or table that depicts the distribution of samples across distinct classes (subject) aids in understanding class balance. Class imbalance may affect model performance, necessitating measures such as class weighting or resampling.



**Text Length Distribution:** Histograms of paragraph lengths give insights into text variability, influencing feature engineering decisions like text normalization and tokenization procedures.



# 2. DATA SPLITTING AND CLEANING.

**Data splitting:** I’ve used scikit-learn's train\_test\_split method to divide the dataset into training and testing sets. The training set includes 80% of the data, while the testing set contains the remaining 20%.

**Data Cleaning:** I addressed missing values by replacing "data missing" with NaN values. I also used dropna to eliminate rows with missing values from the "category" column for task one, but the target column for task two had no missing values.

**Justification:** Removing rows with missing target values ensures that the model is trained on all data instances, which is critical for making accurate predictions.

I also used scikit-learn's SimpleImputer class to fill in missing feature values. The strategy option was set to'most\_frequent' for task one and'mean' for task two, respectively, to replace missing values with the most common and mean values in each column.

To prevent data leakage, I applied independent imputation to the training and testing datasets.

**Justification:** Using the most frequent strategy to infer missing values is a simple but effective technique, especially for categorical features like 'has\_entity'. It ensures that missing values are replaced with the most frequent value in each column while preserving the original data distribution. Imputing missing values helped me to keep relevant information from incomplete data instances, increasing my model's overall robustness. Imputing missing values using the mean method is another useful strategy for numerical data.

# 3. DATA ENCODING

**One-hot encoding for categorical features:** In task one, I used scikit-learn's OneHotEncoder to encode the 'has\_entity' categorical feature, and in task two, I used LabelBinarizer to encode the 'clear\_enough' category. To overcome multicollinearity problems, the drop='first' option was set to omit the first category. This encoding method converts categorical data to binary format, with each category resulting in a unique binary feature.

**Justification:** One-hot encoding is ideal for categorical variables without inherent order, allowing machine learning algorithms to quickly interpret categorical data, whereas labelBinarizer is appropriate for binary categorical variables without intrinsic order.

**Label encoding for the target variable:** I used scikit-learn's LabelEncoder to encode the target variable 'category' and ‘text\_clarity’. This method assigns a unique number to each category in the target variable.

**Justification:** Label encoding is an effective method for translating categorical labels into a format that machine learning algorithms can understand.

**Text tokenization and normalization:** I used spaCy to tokenize the text paragraphs. Tokenization is the process of breaking down text into separate tokens or words. I also conducted text normalization processes such as converting tokens to lowercase, deleting stop words and punctuation, and lemmatization.

**Justification:** Text tokenization and normalization are critical preparation processes for natural language processing. They contribute to text data standardization, dimensionality reduction, and improved feature extraction quality.

**TF-IDF Vectorization of text data**: I used TfidfVectorizer to convert tokenized text to numerical feature vectors. The TF-IDF (Term Frequency-Inverse Document Frequency) metric reflects the value of a term in a document in comparison to a collection of documents.

**Justification:** TF-IDF vectorization captures the relevance of words in text documents by providing greater weight to informative terms and less weight to common phrases.

**Concatenation of encoded features:** After encoding and vectorizing text and category features, I concatenated them with the original feature set using pd.concat and hstack to get the final feature matrices.

**Justification:** Concatenating encoded features preserves and includes all important information in the final dataset for model training and assessment.

These encoding steps ensure that the data is properly formatted and represented for machine learning algorithms, enabling effective learning and prediction. Each encoding technique has its advantages and is chosen based on the nature of the data and the requirements of the machine learning task.

**4. TASK 1: TOPIC CLASSIFICATION**

**4A: MODEL BUILDING**

In the ML-based topic classification task, the Multinomial Naive Bayes classifier was chosen as the final model. Here is a thorough explanation of the final model hyperparameters and the trials performed for optimization:

|  |  |  |
| --- | --- | --- |
| **Hyper-parameter** | **Value** | **Description** |
| Algorithm | Multinomial Naive Bayes | Classification algorithm chosen for the task |
| Alpha (Smoothing  Parameter) | 0.1 | Smoothing parameter for Laplace smoothing in Naive Bayes (I used it to reduce overfitting in the model) |
| Max Features | None | Maximum number of features to consider in TF-IDF vectorization |
| N-gram Range | (1, 1) | Range of n-grams to consider in TF-IDF vectorization |
| Token Pattern | '(?u)\b\w\w+\b' | Regular expression pattern for tokenization |
| Stop Words Removal | Yes | Removal of stop words during tokenization |
| One-Hot Encoding |  |  |
| Drop | 'first' | Dropping the first category to avoid multicollinearity |

# CHOICE OF ALGORITHM

The Multinomial Naive Bayes method was chosen for a variety of reasons.

* It is computationally efficient and performs well with huge datasets and multidimensional feature spaces.
* **Suitability for Text Classification:** Naive Bayes classifiers perform well in text classification

tasks, particularly when using bag-of-words or TF-IDF vectorized features.

* **Robustness to Irrelevant characteristics:** Naive Bayes classifiers are resistant to irrelevant characteristics and can handle noisy data effectively.

# EXPERIMENTS WITH HYPERPARAMETER OPTIMIZATION:

# Alpha Optimization:

To obtain the ideal value, I experimented with several values of alpha (the smoothing parameter). Using cross-validation or grid search, discover the alpha value that optimizes performance measures such as accuracy, precision, recall, and F1-score on the validation set. Choose the alpha value that achieves the optimal balance of bias and variance.

# FEATURE ENGINEERING:

* I Investigated several text preparation approaches, including stop word removal, lemmatization, and punctuation removal.
* Different text vectorization approaches, such as CountVectorizer and TF-IDFVectorizer, were evaluated for their impact.
* To capture more intricate associations in text data, we considered using n-grams and altering the maximum number of features.

# MODEL EVALUATION AND COMPARISON:

* I compared the Multinomial Naive Bayes classifier's performance to that of other algorithms such as SVM, Decision Trees, and Random Forest.
* Model performance was evaluated on both training and validation datasets using measures such as accuracy, precision, recall, and F1-score.
* To guarantee that the model can be generalized to previously unknown data, thorough crossvalidation was used.

Overall, the hyperparameters and model architecture were selected based on empirical evaluation, theoretical knowledge, and machine learning best practices. The objective was to construct a model that achieved high accuracy, generalization to unknown data, and resilience to changes in the dataset.

# 4B. MODEL EVALUATION

**Confusion Matrix:** The confusion matrix gives information about the model's performance across several classes. It reveals the following.

**[[275 8 0 14 8]**

**[ 4 576 0 41 1]**

**[ 3 13 13 1 0]**

**[ 10 17 0 471 2]**

**[ 1 2 0 4 394]]**

Each row represents the actual class, while each column indicates the anticipated class. For example, class 0 had 275 right predictions, 8 for class 1, 14 for class 3, and 8 for class 4, and so on.

# Classification Report:

**precision recall f1-score support**

1. **0.94 0.90 0.92 305**
2. **0.94 0.93 0.93 622**
3. **1.00 0.43 0.60 30**
4. **0.89 0.94 0.91 500**
5. **0.97 0.98 0.98 401**

**accuracy 0.93 1858 macro avg 0.95 0.84 0.87 1858 weighted avg 0.93 0.93 0.93 1858** weighted avg: 0.93 0.93 0.93 1858

**Precision:** The proportion of true positive forecasts to overall positive predictions. For the most part, the model is really accurate.

**Recall:** Determines the fraction of genuine positive forecasts among total real positives. The model

has a high recall rate for the majority of classes.

**F1 score:** The harmonic mean of accuracy and recall. It offers a fair amount of accuracy and memory.

1. **Accuracy Scoring:** The model obtains an accuracy score of around 93.06%, which means it properly classifies 93.06% of the data.

1. **Baseline Accuracy:** The baseline accuracy, which is at 33.48%, acts as a benchmark. It indicates the accuracy attained by a simple baseline model, such as random guessing or forecasting the majority class.

# Evaluation Summary:

* The model exceeds the simple baseline, indicating its ability to categorize text into predetermined categories.
* High accuracy, recall, and F1-score values suggest strong performance across most classes, satisfying the client's need for accurate categorization.
* The confusion matrix and classification report offer specific insights into the model's performance for each classes, identifying opportunities for improvement. The chosen evaluation measures adequately fulfill the client's requirements by giving a complete assessment of the model's categorization performance.

# 4C: CONCLUSIONS

# TASK 1 CONCLUSION:

* The model meets the client's definition of success, as defined in point 1b. It outperforms a simple baseline, does not overfit the training dataset, and assures that no more than 10% of paragraphs are incorrectly categorized into unrelated groups, with the exception of an allowed mistake between the "artificial intelligence" and "programming" categories.
* To monitor the algorithm's success, I propose utilizing the macro-averaged F1 score. This metric takes into account the balance of accuracy and recall across all classes, yielding a single measure of the model's overall classification performance.

# 5. TASK 2: TEXT CLARITY CLASSIFICATION PROTOTYPE.

# 5a. ETHICAL DISCUSSION.

Using an algorithm to automatically reject users' work based on predicted text clarity has various ethical issues and hazards.

**BIAS:** Based on the training data, the algorithm may be biased toward specific writing styles, languages, or cultural subtleties. This may disproportionately harm underprivileged populations or non-native English speakers.

**FAIRNESS:** There is a possibility of unjust treatment if the algorithm's clarity criteria are not apparent or if users do not have a way to dispute rejections.

**DATA QUALITY:** The algorithm's predictions are strongly dependent on the quality and representativeness of the training data. Biased or inadequate data might result in faulty or unjust choices.

**PERFORMANCE:** Despite its excellent accuracy, the algorithm may misclassify ambiguous or subjective material, resulting in unnecessary rejections and user irritation.

**IMPACT ON USERS:** Rejection based on automated evaluations might have an influence on users' motivation, confidence, and involvement, thereby inhibiting useful contributions.

# TO REDUCE ETHICAL HAZARDS, CONSIDER THE FOLLOWING RECOMMENDATIONS:

**TRANSPARENCY:** Clearly express the criteria for determining text clarity and give options for users to appeal denials.

**VARIED TRAINING DATA**: Make sure the algorithm is trained on a varied dataset that includes different writing styles, languages, and cultural situations.

**HUMAN OVERSIGHT:** Use human review processes to supplement the algorithm's judgments, allowing for context-aware evaluations and reducing bias.

**CONTINOUS MONITORING:** Continuous monitoring entails regularly assessing the algorithm's performance and addressing any biases or errors that may occur over time.

**USER EDUCATION:** Inform users about the clarity criteria and offer advice on how to enhance their writing to satisfy the platform's requirements.

Finally, using the algorithm properly necessitates balancing the requirement for speed and quality control with fairness, transparency, and respect for user contributions and diversity. Collaboration with impacted groups and stakeholders may assist identify and manage any dangers, as well as guarantee that the algorithm is used ethically and fairly.

# 5B. DATA LABELLING.

# PROCEDURE FOR LABELING:

I labeled the data according to the readability, coherence, and overall clarity. These are the criteria:

**READABILITY:** Taking into account sentence form and linguistic complexity, is the content easy to understand?

**COHERENCE:** Does the writing make sense and have smooth transitions between concepts?

**OVERALL CLARITY:** Is the primary point made clearly and concisely, leaving no room for doubt or misunderstanding?

I determined certainty by analyzing how well each paragraph complied with these requirements.

Statistics on Labels:

Total labeled data points: 100

"Clear Enough" = 57

"Not Clearly Enough" =43

Example for Each Label:

**Clear\_Enough:** "Ramsay was born in Glasgow on 2 October 1852. He was a nephew of the geologist Sir Andrew Ramsay. His father, William, Sr., was a civil engineer. His mother was Catherine Robertson. He studied at Glasgow Academy, at the University of Glasgow and at University of Tubingen in Germany. "

**Not\_Clear\_Enough:** “A second form of functionalism is based on the rejection of behaviorist theories in psychology and their replacement with empirical cognitive models of the mind. This view is most closely associated with Jerry Fodor and Zenon Pylyshyn and has been labeled psychofunctionalism”.

# 5C. MODEL BUILDING AND EVALUATION.

# ANSWER TO THE TASK OF TEXT CLARITY CLASSIFICATION:

I used a Gradient Boosting Classifier—more precisely, the GradientBoostingClassifier from scikit-learn—to tackle the text clarity classification problem. The final model hyperparameters are as follows:

|  |  |
| --- | --- |
| **Hyperparameter** | **Value** |
| Learning Rate | 0.6217 |
| Max Depth | 5 |
| Max Features | None |
| Min Samples Leaf | 0.5 |
| Min Samples Split | 10 |
| Number of Estimators | 60 |

# THE MODEL EVALUATION IS SHOWN BELOW:

 **TEST DATA CONFUSION MATRIX: [[8 2]**

**[3 7]]**

* **TEST CLASSIFICATION REPORT: precision recall f1-score support**

* 1. **0.73 0.80 0.76 10**
  2. **0.78 0.70 0.74 10**

**accuracy 0.75 20 macro avg 0.75 0.75 0.75 20 weighted avg 0.75 0.75 0.75 20**

* **TEST F1 SCORE: 0.75**
* **BASELINE ACCURACY: 0.50**

# ADVANCED TECHNIQUES:

**TEXT PREPROCESSING:** To clean up the text data, lemmatization, stop word removal, and tokenization were used.

**FEATURE ENGINEERING:** One-hot encoding of categorical variables and positivity of speech tags were applied as extra features.

**GridSearchCV:** This tool ensures improved model performance by optimizing hyperparameters.

# EXPERIMENTS PERFORMED:

* Gradient Boosting produced the greatest results when several classifiers, such as Random Forest and Logistic Regression, were experimented with.
* To determine the best combination of hyperparameters for the Gradient Boosting Classifier, a grid search was conducted over a range of values.

# JUSTIFICATION:

* Gradient Boosting gives good accuracy even with weak learners and is robust to overfitting. It also manages heterogeneous data well.
* Model performance is improved by text preparation approaches, which improve the quality of input information.
* Model performance can be optimized by methodically adjusting hyperparameters with GridSearchCV.

# MODEL EVALUATION:

* Evaluation metrics included Confusion Matrix, Classification Report, Accuracy, Precision, Recall, and F1-score.
* A comparison with the majority class baseline sheds light on the efficacy of the model.

# EVALUATION CRITERIA SUITABILITY:

\* The assessment criteria are in line with the client's need to properly categorize text clarity. \* The model's accuracy in identifying clear and unclear paragraphs is evaluated using precision and recall.

# 5D. TASK 2 CONCLUSIONS

* The model satisfied the requirements of the work by correctly identifying text clarity with a high degree of accuracy, as defined by the client.
* To monitor the algorithm's performance, I advise adding the F1-score as a scalar performance parameter. The F1-score offers a comprehensive assessment of the model's efficacy in categorizing text clarity by striking a balance between recall and precision.
* To potentially increase the model's performance even further, my main recommendation for improvement would be to investigate ensemble strategies like stacking or blending different classifiers. By combining several models, ensemble approaches might potentially improve overall forecast accuracy and robustness by utilizing each model's unique characteristics.

# 6. SELF-REFLECTION

I could have done a better job of explaining the sophisticated methods for model optimization and hyper-parameter adjustment in section 5c. To make it better, I would go into how each parameter option affects the model's performance and provide a detailed explanation of the optimization procedure.

# 7. REFERENCES

1. **Verma E. (2023). *Top 45 Machine Learning Interviews Questions and Answers 2024.***
2. **Lantz, B. (2015). *Machine Learning with R*. Packt Publishing.**

# 3. Mitchel T. (1997). *Machine Learnin.* McGraw-Hill.

1. **Alpaydin, E. (2004). *Introduction To Machine Learning. MIT Press.***
2. **Harrington, P. (2012). *Machine Learning in Action. Manning Publications.***
3. **Conway, D. (2012). *Machine Learning for Hackers. O'Reilly Media.***