Assessment Report

1. Introduction

Welcome to my final report, where I walk you through the journey of predicting company categories based on their web content. This report provides a clear and concise overview of the methodologies used, insights gained, and the approaches taken during feature selection, model training, and evaluation. I also address the challenges encountered along the way.

The report is structured into well-organized sections, covering everything from data cleaning and exploratory analysis to model performance and potential improvements. I've included a detailed account of my process, along with notes for code reviewers to help you navigate through the code efficiently.

It's an extensive report, so make sure you have enough time to go through it. I hope you find it insightful and enjoyable!

2. Methodologies Used

2.1 Data Cleaning and Preprocessing

- **Missing Values Investigation:** Missing values were identified and analyzed per column. Missing values analysis provided guidance for further analysis and feature selection.

- **Text Column Standardization:** Text columns were cleaned by removing special characters, trimming whitespace, and normalizing text case. This ensured consistency across the dataset. This step is important for accurate table joining, duplicate flagging and investigation.

- **Duplicate Flagging and Prioritization:** Duplicates were flagged on Company x Website level (common columns), to prevent skewing the results of further analysis. The goal here is to keep the most complete and accurate information per key.

  CompanyDataset: Flag duplicate records, prioritizing those with the least total missing values, higher total employee estimate and most recent year founded.

  CompanyClassification: Flag duplicate records, prioritizing those with the least total missing values and those with filled meta_description and homepage_text. However analysis revealed that prioritization was not necessary here, since duplicates on key level were also duplicates on record level.

- **Create Indexes**: Create indexes for all tables for the most filtered columns for faster querying.

- **Outlier Detection:** Outliers were flagged in Webpage text columns based on text length (words). These outliers were excluded from specific analyses to ensure they did not skew the results.

- **Common Websites:** Websites that correspond to multiple Companies and/or Categories (eg. ‘blogspot.com’, ‘eu.com’, ‘uk.com’ etc.) were removed in later steps to reduce noise and prevent confusing the model.

- **English Detection**: Came up with an efficient algorithm to score text (between 0-1), based on the amount of English words detected for the first 20 words of each text. Low score records (below 0.35) were removed in later steps after investigation.

2.2 Exploratory Data Analysis (EDA)

- **Text Length Distribution:** The distribution of text lengths across various columns was analyzed, revealing that certain columns, like homepage_text, had much longer texts than others.

- **Correlation Analysis:** The correlation between the total length of text columns and the Category was analyzed. This helped understand the relationship between the amount of webpage text and the company's category.

- **Visualization Techniques:** Boxplots and histograms were used to visualize text lengths and identify potential outliers. Bar plots were used to visualize the distribution of companies across different countries and categories.

- **Word Clouds and tf-idf top terms**: Visualized the most frequent words and terms per Webpage column (homepage_text,h1, h2, … , meta_description) to get a better understanding of their context. A deep dive for meta_description per Category revealed significant context differences.

2.3 Model Building

- **BERT-based Model:** A BERT-based model (pre-trained LLM) was employed for category classification. The model was fine-tuned on the meta_description text feature, which had the most structured and concise webpage information.

- **Label Encoding:** The target variable, Category, was label-encoded to convert the text labels into numerical values suitable for model training.

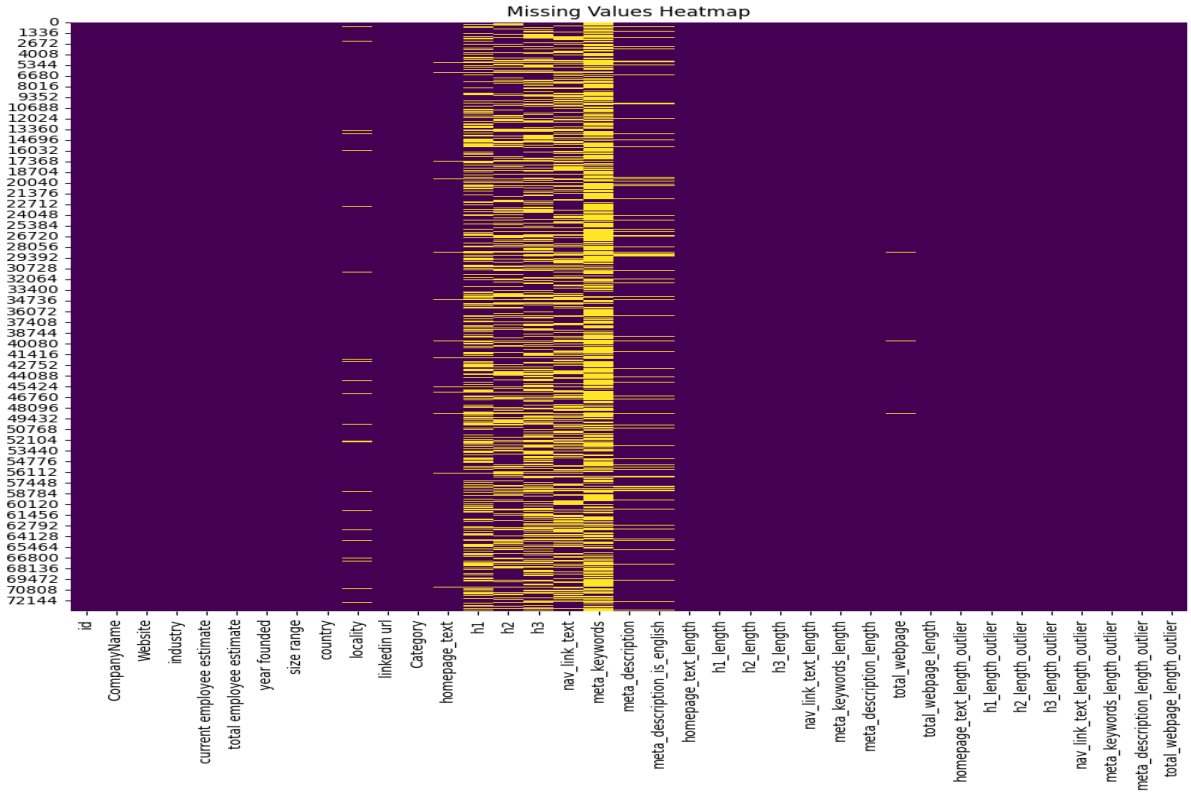
- **Stratified Split**: Due to Class imbalances, a stratified split was chosen to ensure accurate representation of Classes across the training, validation and test datasets.

- **Training and Evaluation:** The model was trained on a training dataset, and its performance was evaluated using a separate test set. Standard metrics such as accuracy, precision, recall, and F1-score were used to assess model performance. These metrics were computed as **weighted averages** to account for class imbalances. **Precision, recall, and F1-score** were also calculated **per class** to provide insight on low performing classes. A **confusion matrix** is also generated to further help identify mis-classification across classes.

3. Insights Gained

3.1 Data Characteristics

- **Missing Data Patterns:** Certain columns had a high proportion of missing values (e.g., meta_keywords), which led to the consideration of these patterns in feature engineering and analysis.

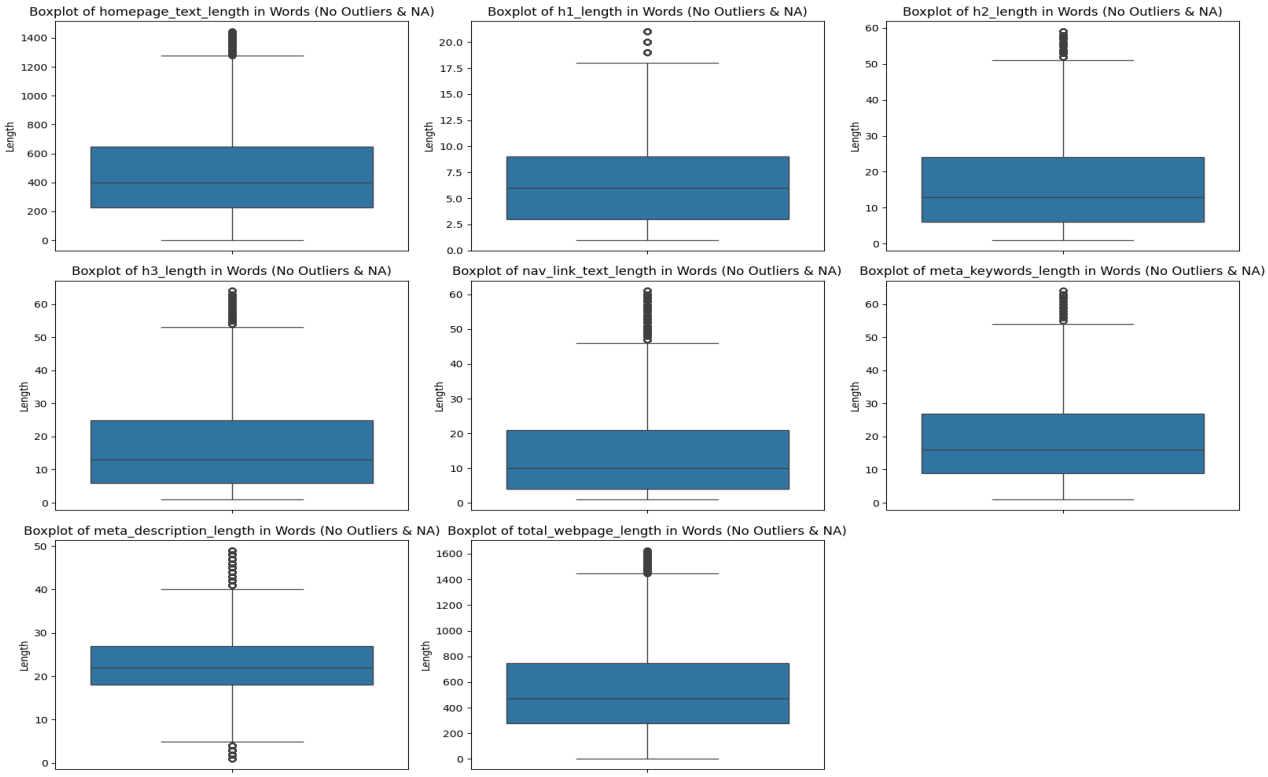


y axis spans across the id column

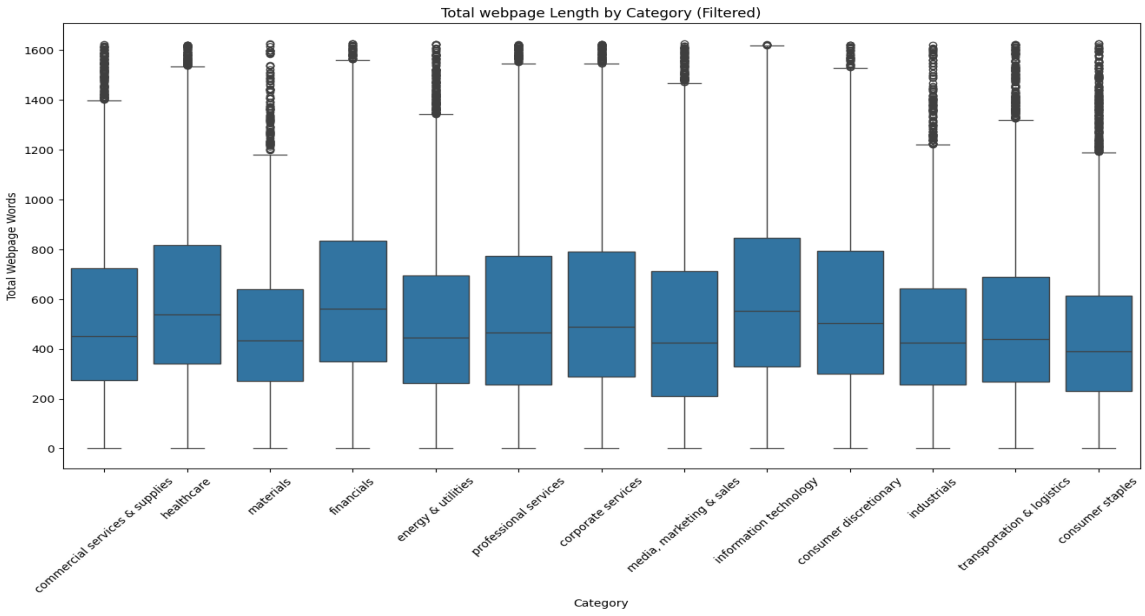
Columns h1, h2, h3, nav_link_text and meta_keywords show a significant amount of missing values, which suggests they should not be included as standalone features.

- **Imbalance in Text Length:** Significant variation in text length was observed across different columns, with homepage_text having by far the longest average length. This required special attention in both EDA and model preparation.

Boxplots of Webpage columns length

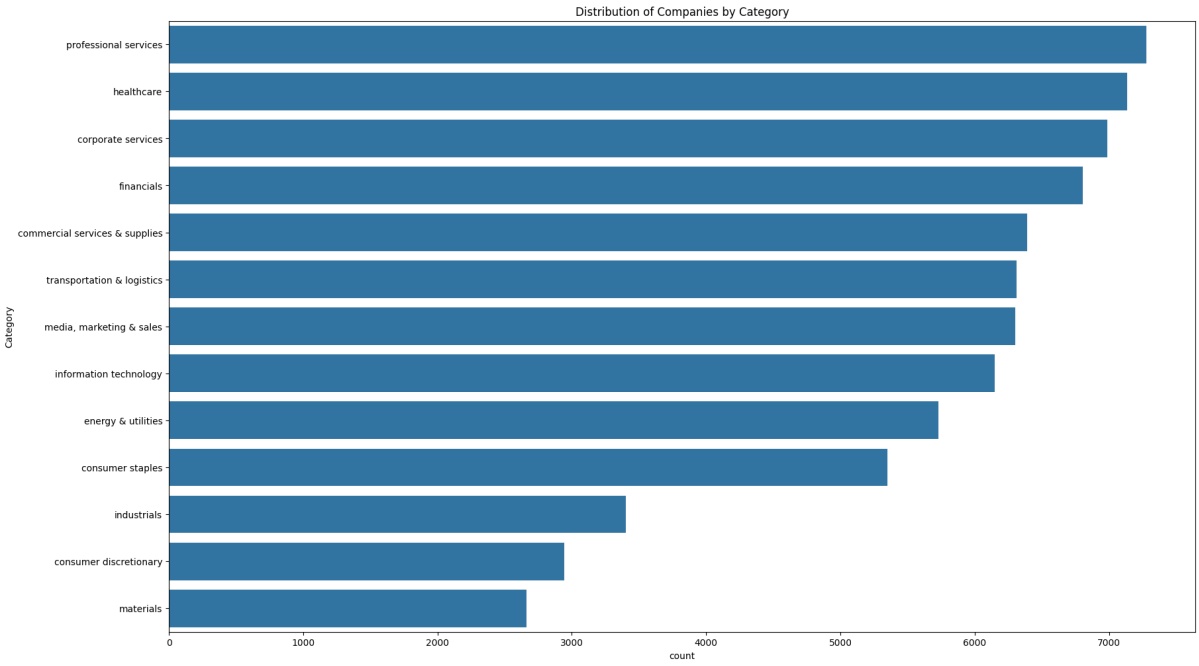


Total Webpage length per category



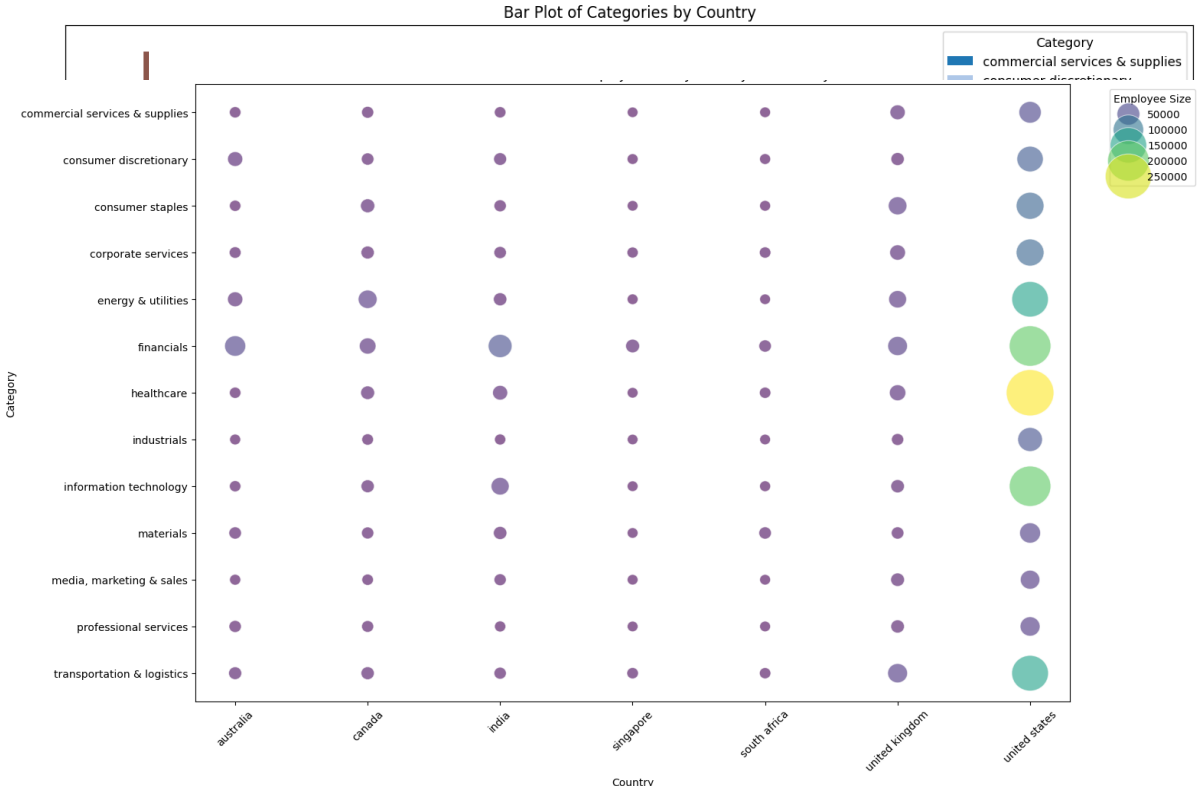
There are very slight variations for Total Webpage length across categories.

- **Category Distribution Imbalance:** Target variable Category is distributed unevenly across Classes with Industrials, Consumer Discretionary and Materials being significantly underrepresented.



- **Category Frequency per Country:** The United States dominates this dataset as they act as a base for almost 50% of companies. Here it's interesting to observe the most dominant Category per country. Healthcare is the most popular category in the US, Corporate Services in the UK and Energy & Utilities in Canada.
  What also stands out here is the amount of companies that specialize in IT in India.

- **Current Employee Count per Country and Category:** As expected, most employees are occupied in the US and more specifically in Healthcare, Financials and IT categories. However, when comparing with the above diagram it's interesting to see that for example while in India IT is the most common Category by far, Financials occupy more people.



Current Employee Size per Category and Country

3.2 Deep Dive - meta_description

As discussed previously meta_description column is a very strong candidate as the primary feature for Category classification due to its:

1. Relatively low amount of missing values
2. Constructed text format, which is ideal for Bert models
3. Concise and clear information, which means lower training times and faster model development

The only thing left to know is if the meta_description column differs enough across Categories, in order for the model to be able to distinguish and identify effectively between different Categories. To investigate that, I created word cloud diagrams and found the top tf-idf terms **per Category**.

Here are the results starting sorted based on Category Popularity:

1. **Professional Services**

Top 5 tf-idf terms: law firm, consulting services, business firm, services, legal firm, tax

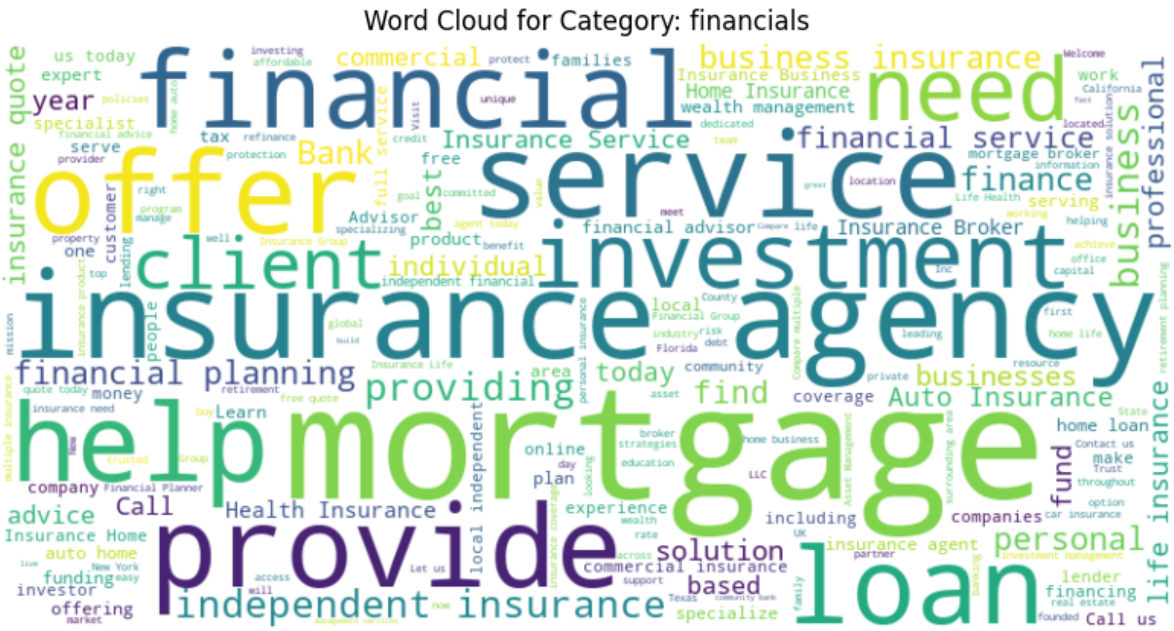
2. **Healthcare**

Top 5 tf-idf terms: medical care, medical, hospital, veterinary, health services, animal care

3. Corporate Services

Top 5 tf-idf terms: property management, hotel management, real estate, recruitment services

4. Financials



Top 5 tf-idf terms: financial insurance, financial, business, mortgage, home insurance

5. **Commercial Services and Supplies**

Top 5 tf-idf terms: security services, electrical services, commercial, cleaning services

These examples suggest that fine-tuning a large LLM model like BERT on this feature could lead to the development of a robust classification model.

4. Model and Evaluation

4.1 Training Dataset

The training dataset consists of around 57.000 examples of unique cleaned meta_descriptions, each of those corresponds to a unique Category.

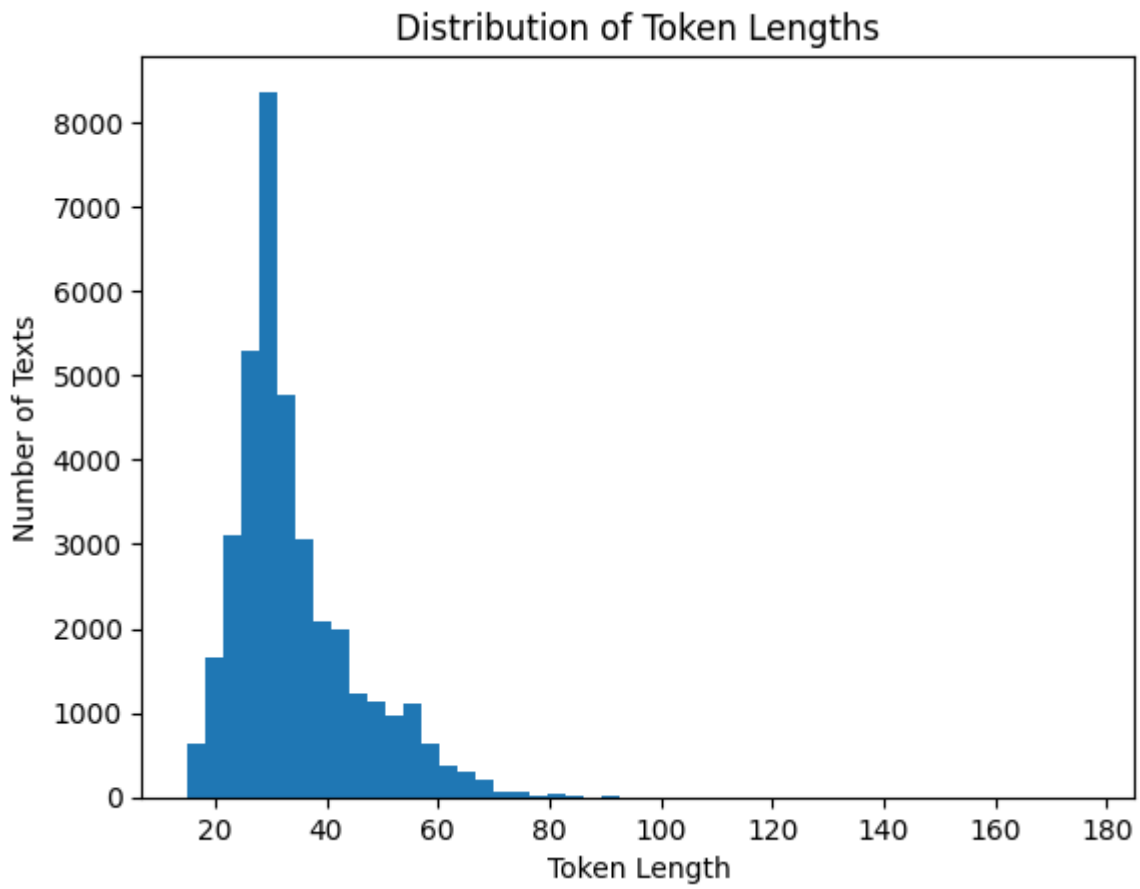
As said before, due to category imbalances we will perform a **stratified split** on this dataset to maintain the Class distributions across the training, evaluation and test datasets.

This results in:

- 37.071 training samples
- 7.944 validation sample
- 7.945 test samples

4.2 Token Length

The token length and tokenization strategy will be determined based on the token distribution of token lengths of the training dataset.



Most samples have 30 to 40 tokens, so will set the max length to 64 tokens to include information from slightly bigger samples. We will use the 'bert-base-uncased' tokenizer, with enabled **padding** and **truncation**.

4.3 Model Configuration

We will use the 'bert-base-uncased' model from Hugging Face's transformers library, and we will fine-tune it on the training samples.

Hyper parameters

- **Learning Rate**: learning_rate=3e-5 A lower learning rate is used to ensure gradual and stable updates to the model weights, reducing the risk of overshooting the optimal solution.

- **Warmup Steps**: warmup_steps=500 Gradually increases the learning rate from 0 to the specified value over the first 500 steps, which helps in stabilizing the training process.

- **Weight Decay**: weight_decay=0.02 Regularization technique to prevent overfitting by penalizing large weights, promoting simpler models.

- **Evaluation and Saving Strategy**:

    - eval_strategy="epoch": Evaluates the model at the end of each epoch.
    - save_strategy="epoch": Saves the model at the end of each epoch.
    - load_best_model_at_end=True: Loads the best-performing model at the end of training based on the specified metric.

- **Early Stopping Metric**: metric_for_best_model="precision", greater_is_better=True

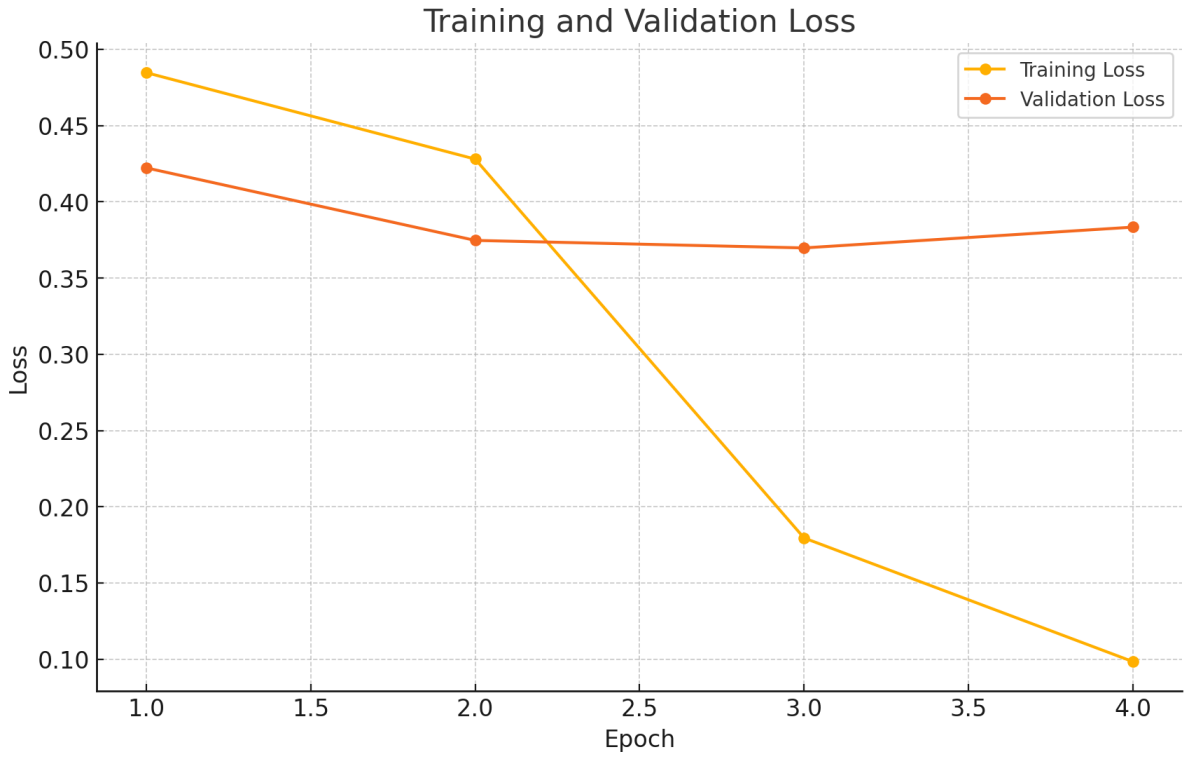
    - The model is evaluated and saved based on precision, optimizing for higher precision, which is crucial in tasks where minimizing false positives is important. This approach helps ensure that the model doesn't simply favor predicting the most frequent classes due to their prevalence.

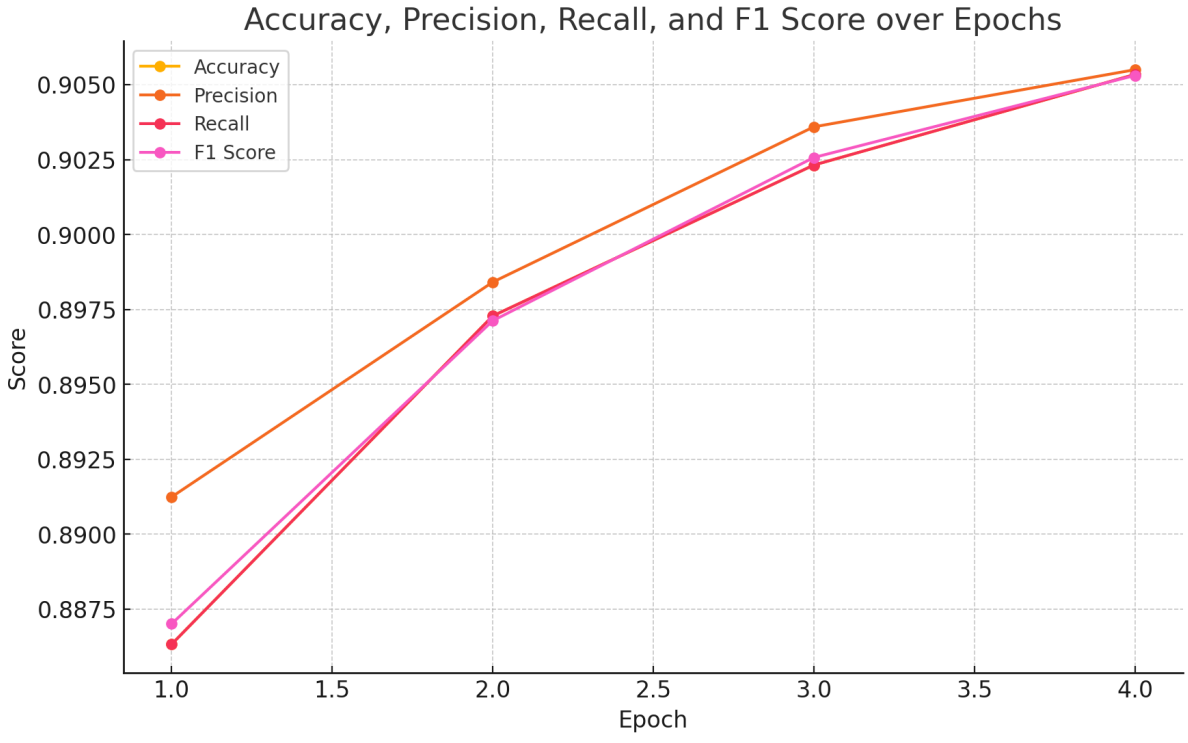
- **Learning Rate Scheduler**: lr_scheduler_type="linear"
    - Uses a linear scheduler that gradually decreases the learning rate after the warmup period.

- **Optimizer**: AdamW A variant of the Adam optimizer with weight decay, well-suited for fine-tuning BERT.

- **Scheduler**: get_linear_schedule_with_warmup Combined with the optimizer, the scheduler manages the learning rate over time, with a warmup phase and linear decay.

4.4 Training Phase





1. **Training and Validation Loss**:
    - The plot shows a decrease in both training and validation loss over the epochs, with training loss decreasing more sharply. The slight increase in validation loss during the final epoch might suggest the onset of overfitting.

2. **Accuracy, Precision, Recall, and F1 Score**:
    - All these metrics show a steady increase across the epochs, indicating consistent improvement in model performance. By the final epoch, the model achieves high scores across all metrics, reflecting balanced and robust performance.

4.5 Model Performance Report

This report summarizes the performance of the model across different classes, providing insights into precision, recall, F1-score, and other key metrics for each class. The report also includes details on true positives, false positives, false negatives, and true negatives, which are derived from the confusion matrix.

Key Metrics

- **Precision**: Measures the accuracy of the positive predictions. High precision means that when the model predicts a class, it is usually correct.
- **Recall**: Measures how well the model identifies all instances of a class. High recall indicates that the model misses fewer instances of the class.
- **F1-Score**: The harmonic mean of precision and recall, providing a single metric that balances both.
- **Support**: The number of actual instances for each class in the dataset.
- **True Positives (TP)**: The number of correct predictions for each class.
- **False Positives (FP)**: The number of incorrect predictions for each class (predicted as the class but actually belongs to another).
- **False Negatives (FN)**: The number of instances where the model failed to predict the correct class.
- **True Negatives (TN)**: The number of correct predictions where the class was not predicted.

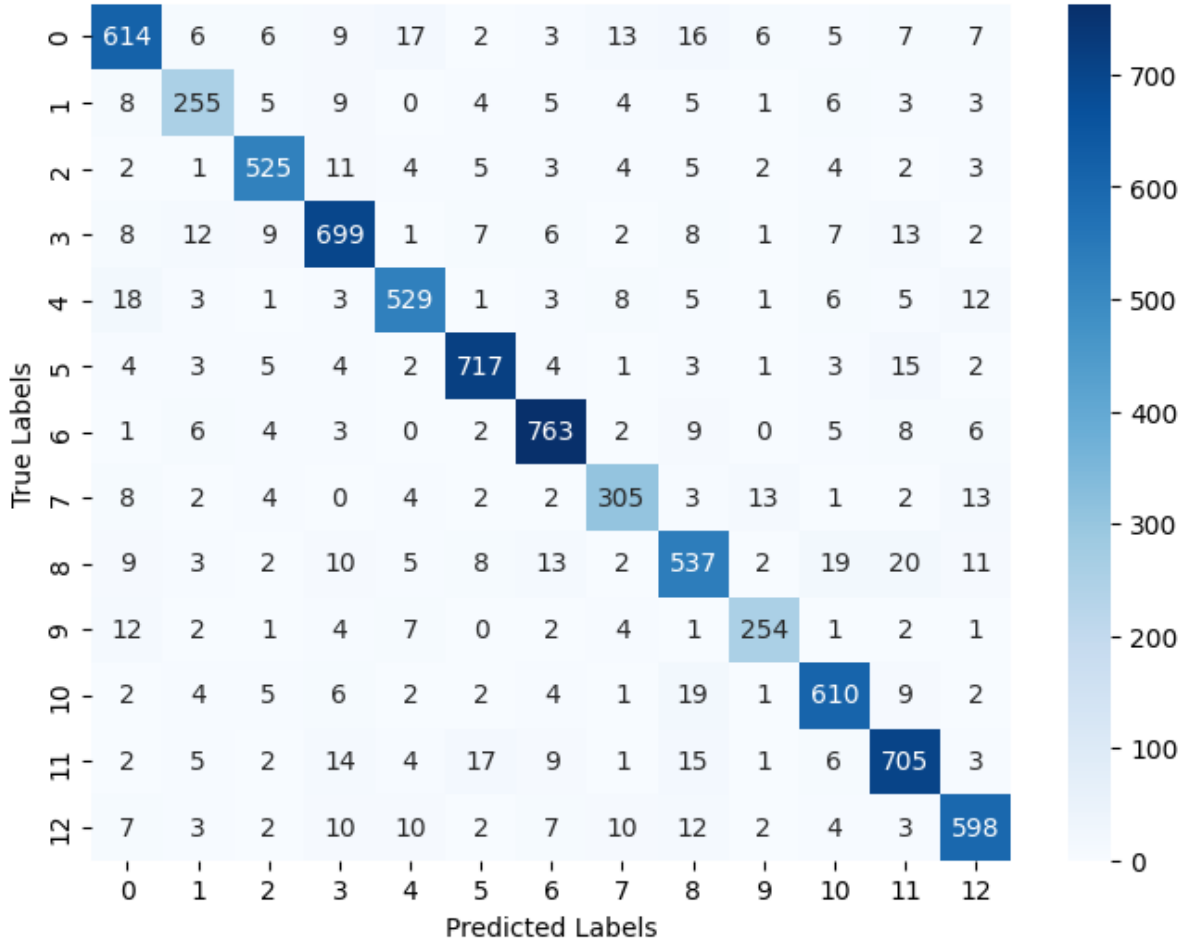
weighted average metrics

accuracy	precision	recall	f1
0.8950283196979232	0.8948877421597973	0.8950283196979232	0.8949175308894523

model performance per Class

Label	Class Name	Precision	Recall	F1-Score	Support	True Positives (TP)	False Positives (FP)	False Negatives (FN)	True Negatives (TN)
0	**commercial services & supplies**	0.8834532374100072	0.8635724331926864	0.8733997155049786	711.0	614	81	97	7153
1	**consumer discretionary**	0.8360655737704918	0.8279220779220078	0.8319738988580752	308.0	255	50	53	7587
2	**consumer staples**	0.9194395796847636	0.9194395796847636	0.9194395796847636	571.0	525	46	46	7328
3	**corporate services**	0.8938618925831202	0.9019354838709678	0.8978805394990367	775.0	699	83	76	7087
4	**energy & utilities**	0.9042735042735044	0.8890756302521008	0.8966101694915254	595.0	529	56	66	7294
5	**financials**	0.9323797139141744	0.938481675392267	0.9354207436399216	764.0	717	52	47	7129
6	**healthcare**	0.9259708737864076	0.9431396786155748	0.9344764237599512	809.0	763	61	46	7075
7	**industrials**	0.8543417366946778	0.8495821727019499	0.8519553072625697	359.0	305	52	54	7534
8	**information technology**	0.8416927899686652	0.8377535101404057	0.8397185301016421	641.0	537	101	104	7203
9	**materials**	0.8912280701754386	0.8728522336769976	0.8819444444444444	291.0	254	31	37	7623
10	**media, marketing & sales**	0.9010339734121122	0.9145427286356822	0.9077380952380952	667.0	610	67	57	7211
11	**professional services**	0.8879093198992444	0.8992346938775511	0.8935361216730039	784.0	705	89	79	7072
12	**transportation & logistics**	0.9019607843137256	0.8925373134328358	0.8972243060765192	670.0	598	65	72	7210

Confusion Matrix



Insights

1. **Class 2: Consumer Staples**
    - **Highest Performance**: This class has the highest F1-score (0.919) with balanced precision and recall, indicating that the model is very effective in predicting this class. The confusion matrix shows minimal misclassification for this class.

2. **Class 0: Commercial Services & Supplies**
    - **Balanced Performance**: This class has a good F1-score (0.873) but shows some misclassification, with 81 false positives and 97 false negatives. There is room for improvement in reducing these errors.

3. **Class 1: Consumer Discretionary**
    - **Lower Performance**: This class has a slightly lower F1-score (0.832) compared to others. With 50 false positives and 53 false negatives, the model struggles a bit more with this class, suggesting that it might benefit from further tuning.

4. **Class 4: Energy & Utilities**
    - **Moderate Misclassification**: While the model performs well (F1-score of 0.897), there are 66 false negatives, indicating that the model sometimes fails to identify instances of this class.

Conclusion

The model appears to be performing well overall, with good precision, recall, and F1-scores for most classes. Class 2 is performing particularly well, while Class 1 has slightly lower metrics, indicating potential areas for improvement. The balanced F1-scores across classes suggest that the model generalizes well, but a further evaluation on a separate test set is recommended to confirm these findings.

4.6 Potential Improvements

1. **Class-Specific Tuning**: Focus on improving the model's performance for classes with lower F1-scores, particularly Class 1 (Consumer Discretionary) and Class 4 (Energy & Utilities). This could involve collecting more data for these classes, adjusting class weights, or employing techniques like SMOTE to handle class imbalance.

2. **Error Analysis**: Conduct a deeper analysis of the false positives and false negatives to understand common patterns in misclassification. This could guide feature engineering or help in refining the model's architecture.

3. **Confusion Matrix Analysis**: Further inspection of off-diagonal elements in the confusion matrix could reveal specific classes that are often confused with each other, providing insights into potential model enhancements.

5. Challenges Faced

5.1 Data cleaning

Duplicate Handling

To generate accurate reports and train the model effectively, investigating and excluding duplicates is a challenging yet crucial step.

In the CompanyDataset table, many duplicates were found on the Company x Website key. These duplicates had to be carefully sorted and removed, with the most complete record being retained.

In contrast, the CompanyClassification table contained exact duplicates, which were easier to identify and remove. This thorough process ensured that only unique records were used for analysis and model training.

Common Websites across companies

These websites had common webpage information, while corresponding to different companies and categories, thus had to be identified and removed from the dataset.

website	company count (unique)	category count (unique)	homepage_text count (unique)
blogspot.com	13	7	1
blogspot.in	5	5	1
eu.com	7	6	1
uk.com	84	13	1
uk.net	6	6	1
us.com	24	10	1

These websites are:

- Blogging Platforms (blogspot.com, blogspot.in): Multiple companies create blogs under these shared domains, leading to multiple company associations.
- Regional Domains (eu.com, uk.com, uk.net, us.com): These are shared domains managed by registrars that allow multiple companies to register subdomains or web

addresses under a common regional domain, resulting in several companies using the same main domain.

Different Websites with Identical Website Information

This situation posed a similar challenge, as it could confuse the model by presenting duplicated features associated with different categories. To ensure clarity and uniqueness in the dataset, I needed to carefully investigate and resolve these duplicates.

I started by removing duplicate meta_descriptions that were linked to different categories. Then, I re-examined the remaining meta_description duplicates to ensure that only one was retained, thereby maintaining the uniqueness of meta_descriptions in the machine learning training dataset.

Text Cleaning

Webpage-related text columns required extensive cleaning to eliminate noise and ensure the data was clean and readable. This process was essential while preserving punctuation, which is important for BERT-based models.

To achieve this, I developed a specialized module dedicated to handling these tasks, which effectively removed noisy data and standardized the text columns.

This refinement ensured that the text was optimized for model training, maintaining the necessary structure for accurate processing by the BERT model.

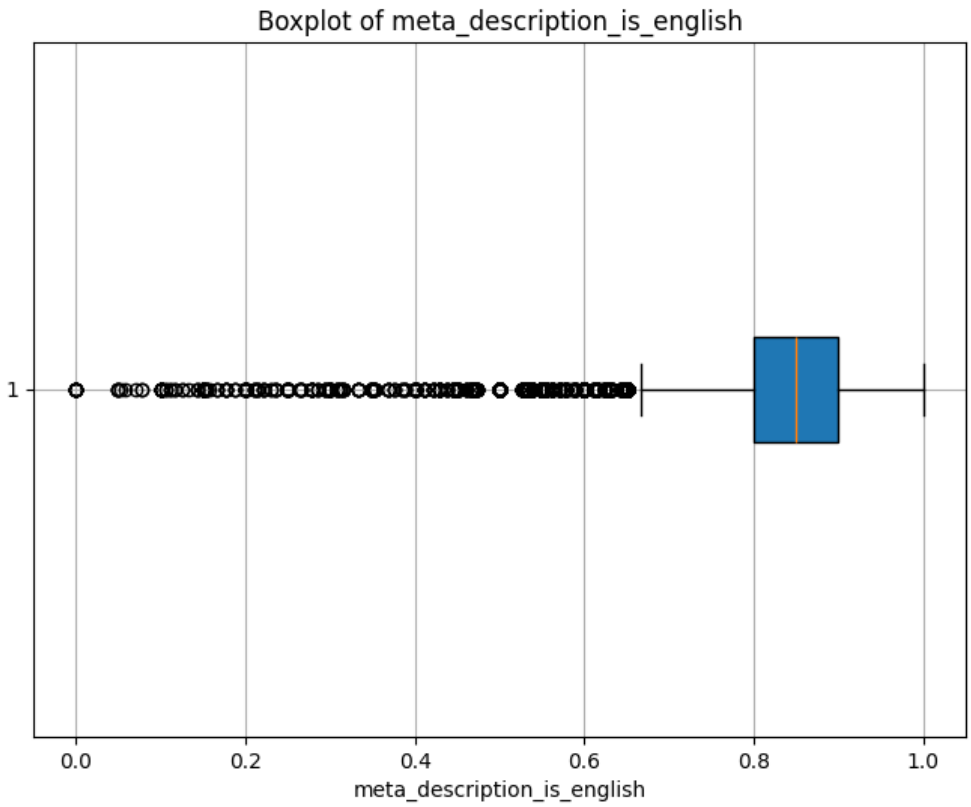
Identification of non-english text

Identifying non-English text was a critical step, as these records needed to be excluded from the BERT model training dataset to ensure accuracy. Initially, I explored using modules like langdetect, langid, and polyglot for this task. However, these solutions proved inefficient due to the large size of the texts and slow processing times.

To address this, I developed a custom and efficient solution that assigns each text a float rating from 0 to 1, based on the proportion of English words detected within the first 20 words of each record in the meta_description column. After thorough investigation, I determined that scores below 0.35 typically indicated that the text was in a non-English language, such as Mandarin, Greek, or French. Consequently, these records were removed from the dataset.

This approach revealed that the vast majority of meta_description records were indeed in English. However, careful outlier analysis was crucial in establishing the final acceptable threshold, ensuring that no important English records were mistakenly discarded.

The distribution of this new feature is shown below:



5.2 Feature selection

Initially, the homege_text appeared to be the most promising feature for training the model. However, its large size and unstructured format made it unsuitable for effective training. As a result, I conducted a thorough investigation into other website features, ultimately identifying meta_description as the primary feature for training.
This decision involved trading a larger training dataset for more concise and structured information, which proved to be better suited for the model's learning process.

5.3 Model Training Resources / Hardware

Training BERT and experimenting with large language models (LLMs) is a resource-intensive task that demands significant computational power and appropriate software. To efficiently handle this, I utilized cloud resources, specifically Google Colab, to conduct the training. This approach enabled me to train six different versions of the model, each with varying hyperparameters and configurations, optimizing the experimentation process.

Final Model Implementation

To streamline the model deployment, I developed a Python module named ml_model_training.py, which encapsulates the code for the final version of the model. This module ensures that the final model is easily reproducible and deployable, consolidating all the necessary training logic and configurations in one place.

6. Notes for Code Reviewer

First of all, thank you for taking the time to delve into this report!

Since the codebase is highly modular and structured as an application, I recommend consulting the directory tree to navigate through the code effectively. You can find this directory tree in the top-level directory (root) within a file named dir_tree.txt. This should help you quickly locate the relevant modules and understand the overall structure.