Comp1804 Report

Mar 2025

Word count: 2607

Executive summary.

Task 1: A BiLSTM model with FastText embeddings was employed to classify petition topics. Despite high test accuracy and generalisation, the model failed to meet class-wise misclassification and precision criteria for sensitive classes.

Task 2: A lightweight XGBoost prototype was trained to identify important petitions. The model achieved 1.00 recall for “important” class, aligning with the client’s priority of minimizing false negatives. Trade-offs in precision were acceptable given the decision-support context.

1. Data exploration and assessment.

The dataset was loaded using pandas and inspected to understand its shape (8898,8) and column types. Basic EDA revealed the presence of nulls, 378 identical rows and 104 text duplicates as shown in Table 1.

Column	Null count
relevant_department	1799
deviation_across_regions	1577
petition_topic	31
petition_text	2

Table 1

With approximately 20% of the dataset being null in the last two mentioned columns further investigation was done showing they all belonged to the rejected petitions; likely because they didn’t move forward in the system.

Twenty-four hidden nulls were found in the has_entity column, recorded as "data missing." Similarly, thirteen placeholder petitions labelled as empty_test_petition were identified in the petition_text column. Figure 1 shows the fragmentation of petition topics.

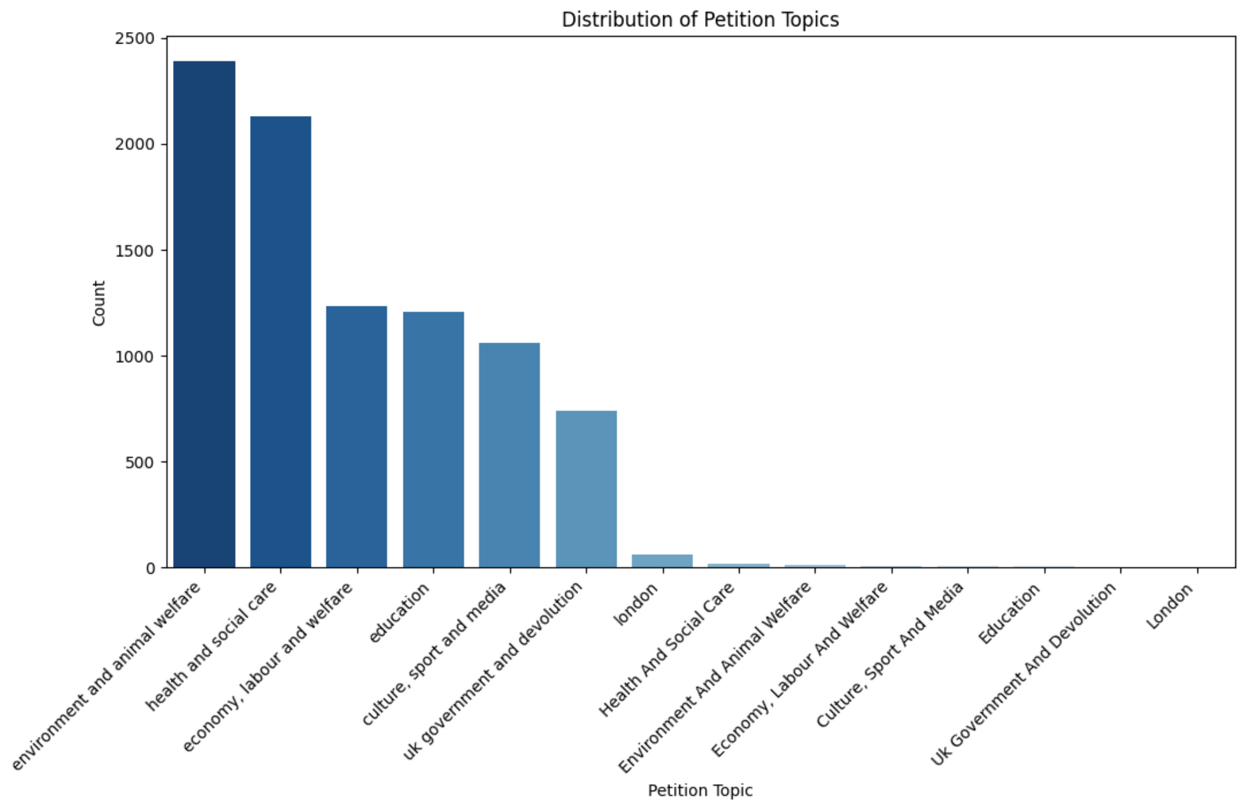


Figure 1

Text analysis included detecting the most common words with and without stopwords, looking for unusually short petitions and counting special characters. An examination of Zipf's Law in Figure 2 ensured natural language pattern. Text findings are summarized in Table 2.

Text findings	count
Multiple spaces	2846
Newline characters	3228
Average special characters per text	12.07
Petitions under 50 words	0

Table 2

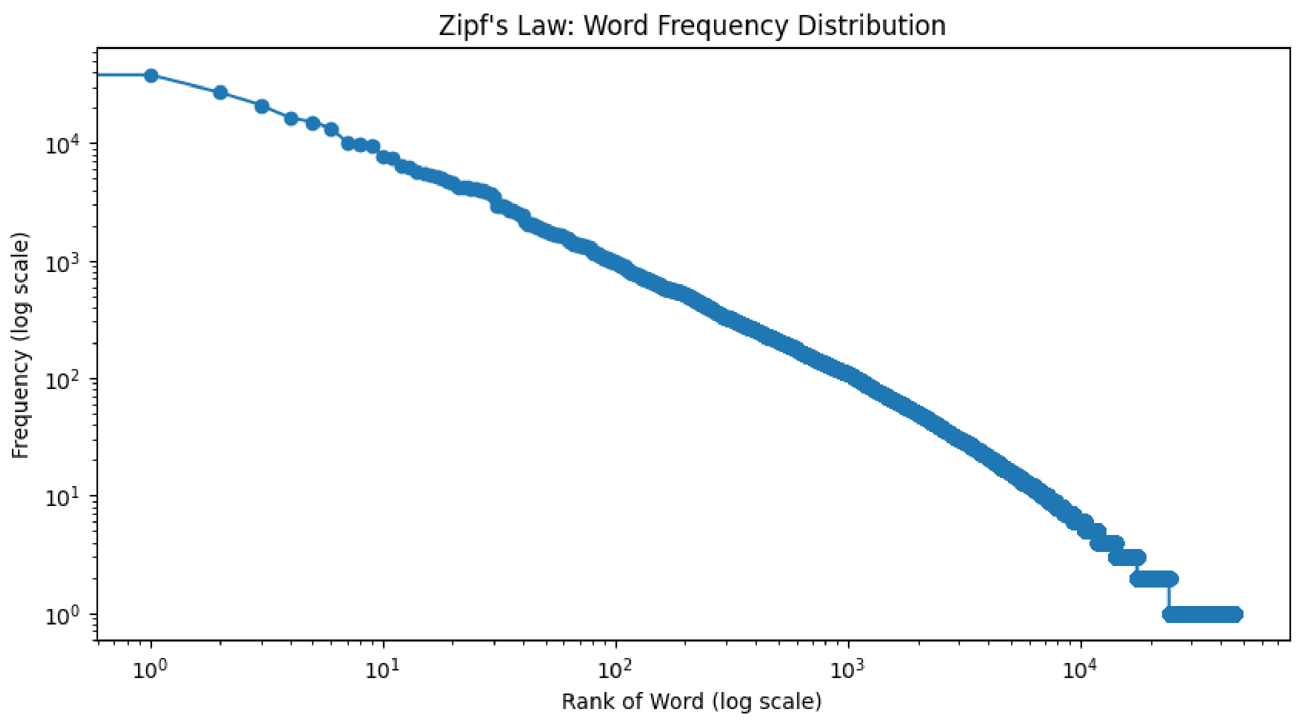


Figure 2

Most petitions are ranged between 86–134 words, with a peak around 130 following a right-skewed distribution (Figure 3). Frequent word analysis (Figures 4) confirmed key terms like "government" and "NHS" were retained out of stopwords.

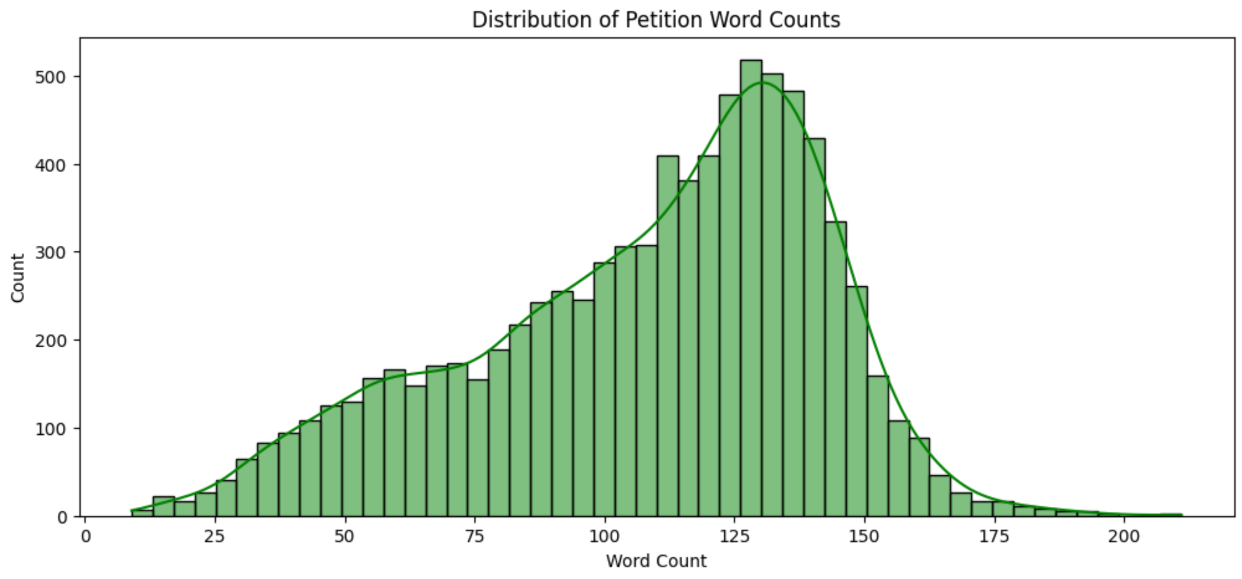


Figure 3

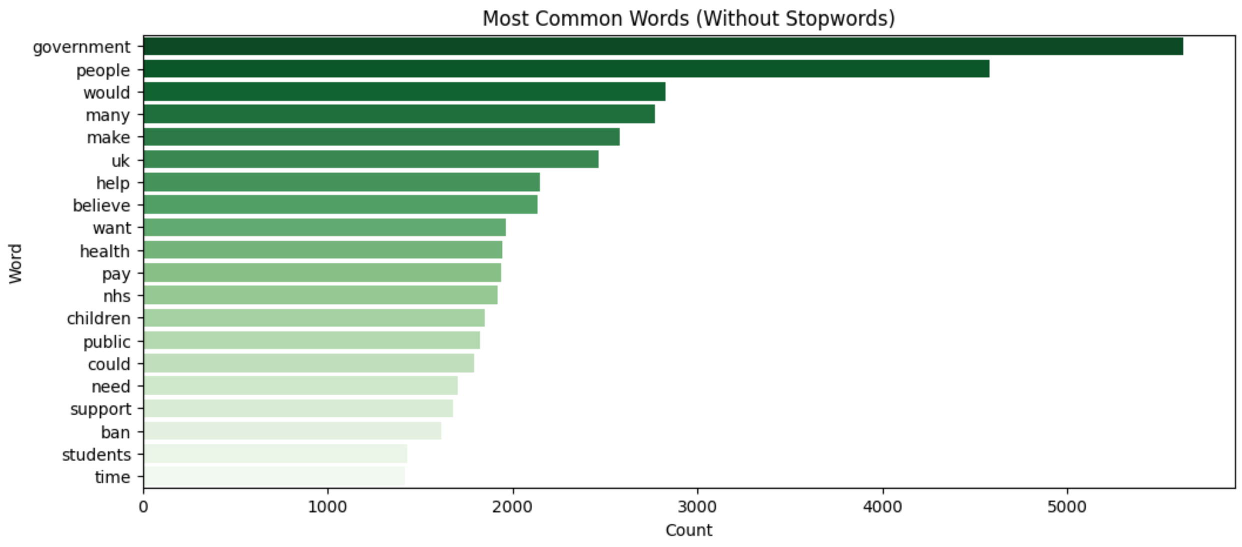


Figure 4

Minor issues like extra spaces, newlines, and excessive punctuation were found in a subset of texts and fixed during cleaning. No actual typos were detected. Key text issues are summarized in Table 3:

Issue	Count
Extra spaces	1,973
Newline characters	2,259
Excessive punctuation	46
Repeated characters	9
Spelling errors (true typos)	0

Table 3

2. Data splitting and cleaning.

All values in the petition_topic column were lowercased to merge duplicates caused by inconsistent casing. Figure 5 highlights the class imbalance across topics. No resampling was used, as synthetic balancing could distort real-world distributions.

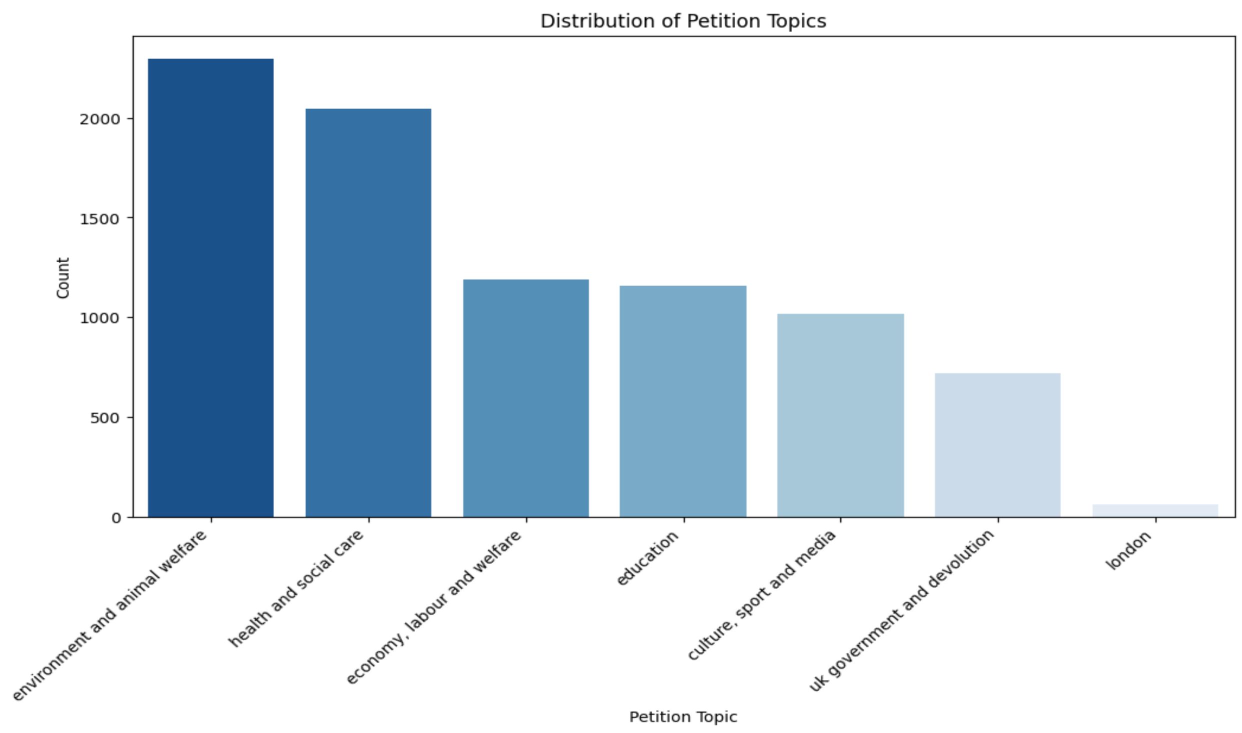
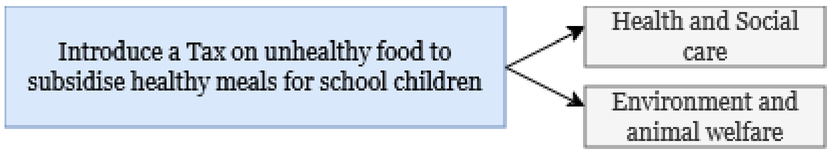


Figure 5

After removing full row duplicates, only two nulls were left from petition_text which were also dropped as text itself is an important input and is better left without imputation. Petition_topic nulls were removed as they were unusable for classification.

Among the 104 petition_text dups, three were identified where the same text appeared under conflicting topics. After manual review, the most contextually appropriate topic was selected for each case as illustrated in Figure 6. The rest were removed to prevent overfitting in Task A, ensuring the model learns generalizable patterns rather than memorizing repeated text.



Introduce a Tax on unhealthy food to subsidise healthy meals for school children
Health and Social care
Environment and animal welfare

Figure 6

While it was theoretically possible to infer the missing values in the has_entity column from petition_text, this was not done for two key reasons. First, modifying the column would introduce inconsistencies, as the original feature engineering method is unknown and may have applied specific filtering rules. Second, given the small number of missing values, filling them in would not have had a meaningful impact on model performance. Similarly, 'empty_test_petition' were removed as they do not represent real user-generated content.

Nulls in deviation_across_regions were retained, as variation cannot exist without collected signatures. Since Task 2 uses a tree-based model, it will inherently handle and interpret these missing values. This process left no nulls or duplicates, enabling a final review of petition_text.

Data Splitting for Task 1

The dataset was stratified by petition_topic and split into 70% training (5905, 3), 15% validation (1265, 3), and 15% test (1266, 3) to maintain a balanced distribution.

Data Splitting for Task 2

The dataset was split into 85% training (85, 4) and 15% test (15, 4), with no validation set used due to the model's smaller size.

Text Cleaning After Splitting

To prevent data leakage, all cleaning steps were applied post-split. Cleaning ensured that noisy elements like URLs, emails, phone numbers, special characters, excessive punctuation, and formatting issues were removed. Unicode normalization (NFKC) and contraction expansion helped standardize the text, resulting in clean, lowercase, whitespace-normalized inputs suitable for modeling. No stemming or stopword removal was performed. This decision was supported by the use of pre-trained FastText embeddings, which capture semantic relationships across full vocabulary.

Following cleaning, the word count distribution (Figure 7) remained stable, with a slightly sharper peak and fewer low-word-count petitions.

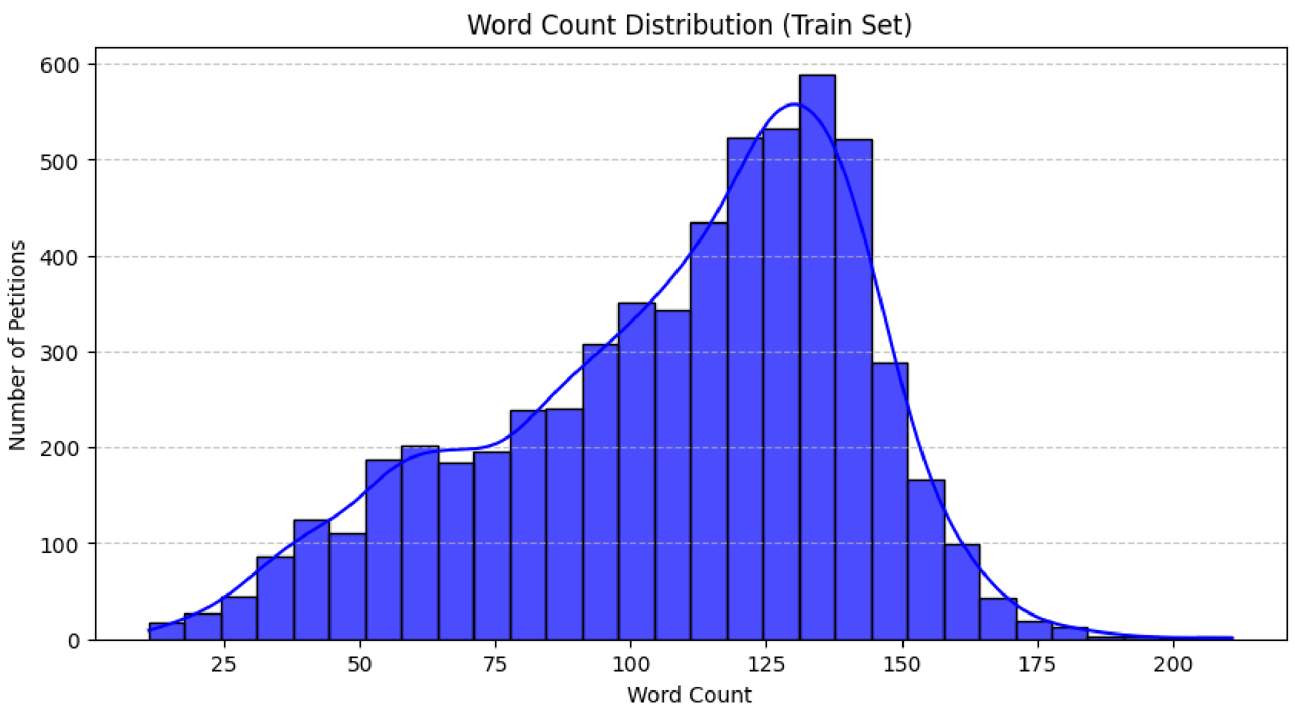

Word Count Distribution (Train Set)
Number of Petitions
Word Count
0
100
200
300
400
500
600
25
50
75
100
125
150
175
200

Figure 7

3. Data encoding.

Figure 8 summarizes the full encoding process. All encoders were fit on the training set and then applied to the rest to prevent data leakage. After encoding, the original categorical columns were dropped in both tasks.

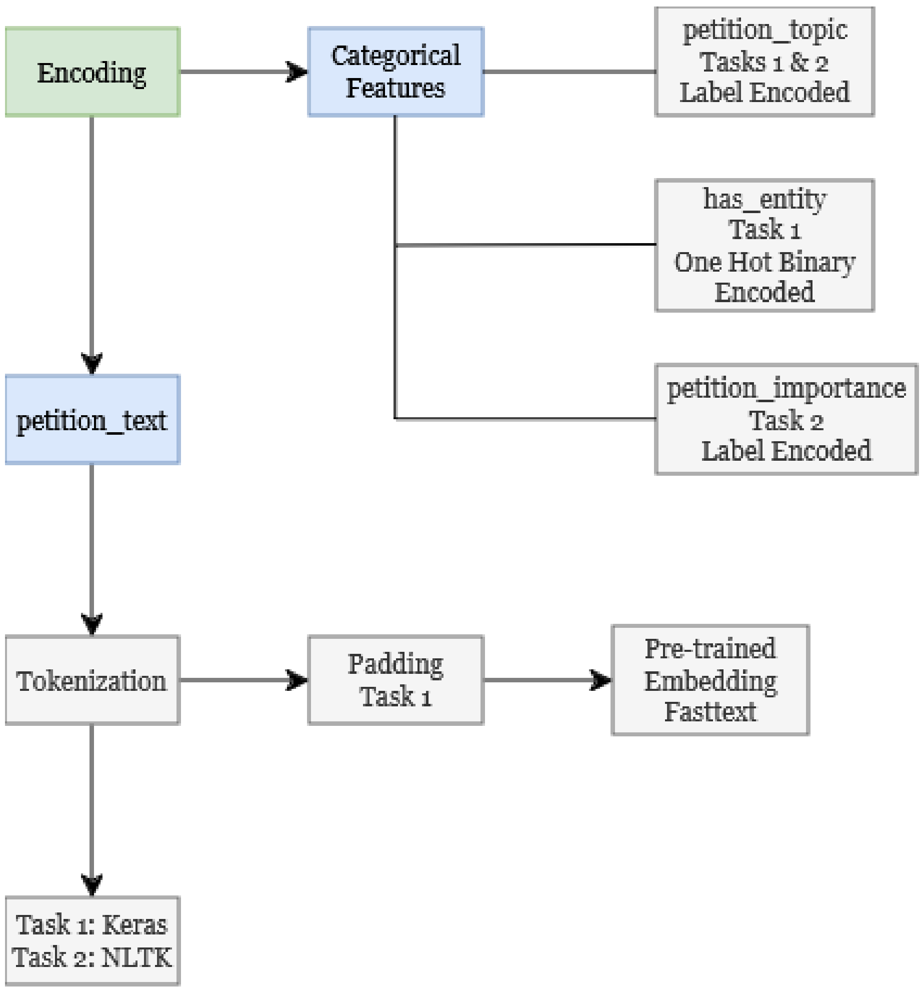

Encoding
Categorical Features
petition_topic Tasks 1 & 2 Label Encoded
has_entity Task 1 One Hot Binary Encoded
petition_importance Task 2 Label Encoded
petition_text
Tokenization
Padding Task 1
Pre-trained Embedding Fasttext
Task 1: Keras Task 2: NLTK

Figure 8

Text Encoding

Task 1

Cleaned text was tokenized using Keras Tokenizer, fitted on the training set to prevent leakage. The 20,000 most frequent words were retained, with <OOV> for unseen tokens, resulting in a vocabulary size of 18,340. Single-occurrence words were removed to reduce noise. Sequences were post-padded and truncated to 219 tokens, preserving full structure. Pre-trained FastText (300D) embeddings were applied to capture semantic meaning, including rare and misspelled words.

Task 2

For Task 2, text encoding followed a simpler pipeline suitable for tree-based models. NFKC was applied and each petition was tokenized by splitting on whitespace. No padding was needed since each petition was represented by a single fixed-size embedding vector. These vectors were further compressed to 50 dimensions using PCA, since high-dimensional embeddings and a small dataset is in risk of overfitting.

4. Task 1: topic classification.

4a. Model building.

Core Model

Bidirectional LSTMs were chosen as the foundational model due to their ability to capture both past and future context, making them well-suited for nuanced petition texts with temporal or conditional clauses. Unlike CNNs or simpler RNNs, BiLSTMs retain longer-term dependencies and better capture contextual and syntactic information. FastText provides meaningful word vectors upfront, allowing the BiLSTM to focus on learning domain-specific sequential patterns rather than basic word semantics.

Model 1 – Optuna Based BiLSTM

To establish a baseline for Task 1, Optuna was used to explore the model's sensitivity. The best result from 10 trials is summarised in Table 4.

Hyperparameter	Search Range	Type	Best Value Found	Justification
lstm_units	64, 96, 128	Categorical	128	Controls the dimensionality of the hidden state exploring a balance between underfitting (64) and potential overfitting (128), with 96 as an intermediate
dropout_rate	0.3 – 0.5	Continuous (float)	0.4645	Applied between layers to reduce overfitting Offering a balance between retaining information and improving generalization
batch_size	32, 64	Categorical	32	Impacts learning dynamics. Smaller batches (32) may generalize better, while 64 enables faster training on GPU

Hyperparameter	Search Range	Type	Best Value Found	Justification
learning_rate	2e-5 – 1e-4 (log-uniform)	Log-uniform float	8.97e-5	Critical for convergence. Log-uniform scale allows fine control over very small learning rates common in fine-tuning embeddings

Table 4

The model starts with a FastText embedding layer (300D), aligned to the tokenizer via a custom matrix and set as trainable to allow fine-tuning. It outputs dense sequences passed through two Bidirectional LSTM layers: the first returns sequences, the second a single vector. Spatial dropout follows the embedding, and dropout is applied after each LSTM to reduce overfitting.

To handle class imbalance, balanced weights (capped at 5.0) were passed to the loss function. The model uses a dual-input setup, combining text and encoded features. Structured inputs pass through Dense(64) → BatchNorm → Dropout(0.3), then are concatenated with the BiLSTM output. This is followed by a final Dense(64) layer with Dropout(0.35) and a softmax output.

This dense-layer (Table 5) and embedding structure was consistent across all models, unless stated otherwise.

Component	Configuration	Purpose
Dense Layer (structured input)	64 units, ReLU activation	Learns non-linear patterns from structured binary features
Batch Normalisation	Applied after Dense(64)	Stabilises learning and improves convergence
Dropout (structured path)	0.3	Regularises structured feature subnetwork
Dense layer (after concatenation)	64 units, ReLU activation	Adds abstraction after combining text and structured features
Dropout (fusion path)	0.35	Reduces overfitting before final classification
Output layer	Dense(7), Softmax activation	Produces class probabilities for 7-class topic classification

Table 5

The model was trained for 10 epochs at a learning rate of 8.97×10^{-5}, followed by 3 fine-tuning epochs at 1×10^{-5}. Early stopping (patience = 3) was applied to prevent overfitting.

Model 2 – BiLSTM with Frozen → Unfrozen Embeddings

This model retains the previous configuration but increases batch_size to 64 and adopts a two-stage training strategy. Embeddings were frozen for 5 epochs (LR = 1.5×10^{-4}), then unfrozen and fine-tuned for 5 more (LR = 5×10^{-5}). Early stopping (patience = 2) was applied throughout. Overall performance was improved, yet minority classes (5 and 6) were underperforming.

Model 3 - Enhanced Class Weighting

To prioritize underrepresented topics during training, additional multipliers were applied: ×2.0 for class 5 and ×1.8 for class 6 with the previous configurations. Model 10 was fully trainable, starting with frozen embeddings and gradually reducing the learning rate across four stages: 1e-4 → 2e-5 → 1e-5 → 5e-6, over a total of 23 epochs. While recall improved for the minority classes, the overall test accuracy decreased.

Model 4 - Stacked BiLSTM with Attention

This model extends the architecture by adding attention on top of two BiLSTM layers (return_sequences=True), allowing it to focus on key tokens after learning full sequence context. It used frozen embeddings for 5 epochs (lr = 1.5×10^{-4}), then unfroze for 5 fine-tuning epochs (lr = 5×10^{-5}), with early stopping (patience = 2). While slightly lower in test accuracy than Model 3, it improved interpretability.

Model Performance

As shown in Table 6, Model 2 outperformed all other variants overall. While Model 4 improved interpretability with attention, it did not exceed in generalization or stability. It balanced precision and recall across all classes, and demonstrated no signs of overfitting.

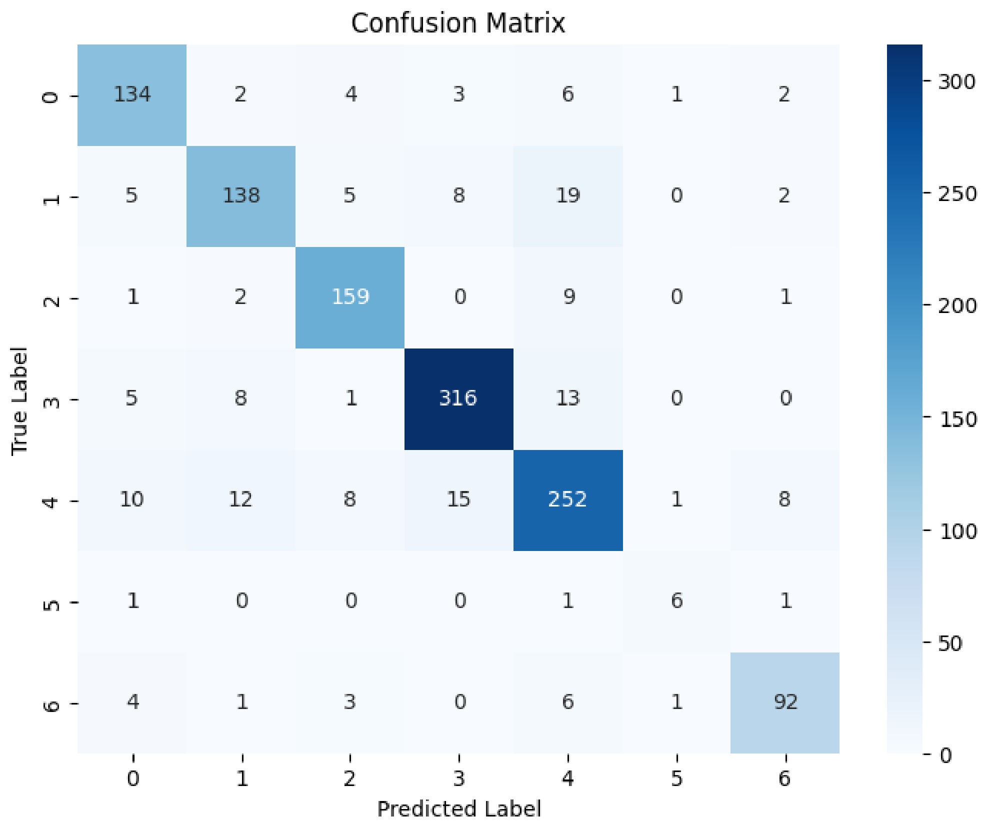
Model	Accuracy	Macro f1	Log Loss	AUC (OvR)
1	0.8025	0.7345	0.6627	0.9588
2	0.8665	0.8358	0.4544	0.99
3	0.7915	0.7300	0.6967	0.9556
4	0.8571	0.8261	0.5517	0.9665

Table 6

4b. Model evaluation.

As shown in Table 6, Model 2 outperformed all other variants, achieving the highest overall accuracy, macro F1 score (suitable for imbalanced multi-class cases), and lowest log loss, with an AUC of 0.99, indicating confident predictions.

Fig 9 shows strong predictions across major classes. However, the model struggled more with underrepresented categories such as class 5 and 6, due to limited training samples and semantic overlap.



True \ Predicted	0	1	2	3	4	5	6
0	134	2	4	3	6	1	2
1	5	138	5	8	19	0	2
2	1	2	159	0	9	0	1
3	5	8	1	316	13	0	0
4	10	12	8	15	252	1	8
5	1	0	0	0	1	6	1
6	4	1	3	0	6	1	92

Figure 9

Analysis of false negatives revealed that errors were not random. They often occurred between semantically adjacent classes, indicating that the model had learned meaningful patterns but was constrained by the limitations of single-label classification. For instance, the true label of the petition in Figure 10 was "Economy," but words like *"Covid"* and *"hospital"* led to a false prediction of "Health." This suggests that a multi-label classification approach may be more
appropriate for this task.

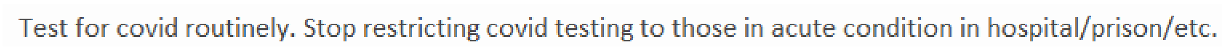

Test for covid routinely. Stop restricting covid testing to those in acute condition in hospital/prison/etc.

Figure 10

According to Table 7, only three classes had ≤13% misclassification rate and class 6 did not meet the required minimum precision of 91%. Therefore, the client's expectations were not met. The training vs. validation accuracy curve (Figure 11) shows stable learning, with no signs of overfitting with early stopping triggered before performance declines.

Class	precision	recall	F1-score	support	misclassification
0: culture	0.84	0.88	0.86	152	11.84
1: economy	0.85	0.78	0.81	177	22.03
2: education	0.88	0.92	0.90	172	7.55
3: environment	0.92	0.92	0.92	343	7.87
4: health	0.82	0.82	0.82	306	17.64
5: London	0.67	0.67	0.67	9	33.33
6: Uk	0.87	0.86	0.86	107	14.01

Table 7

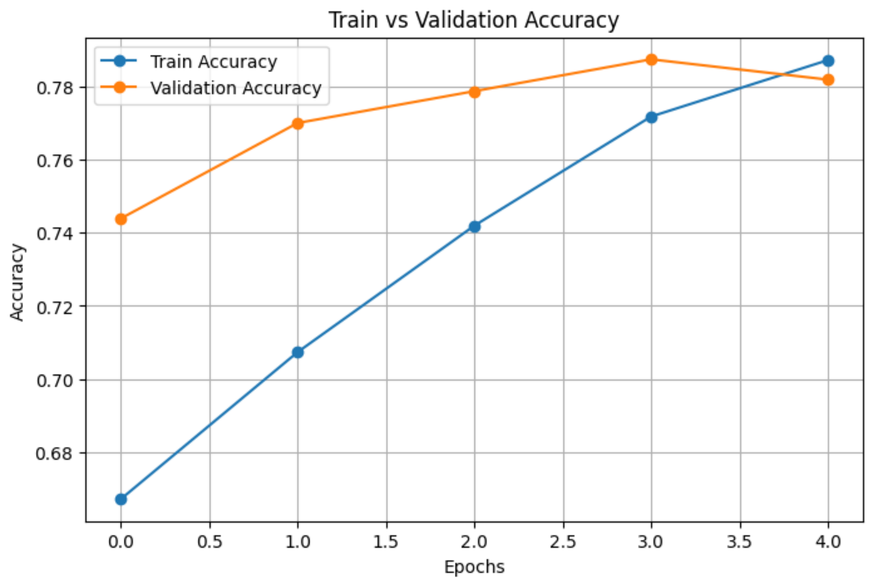


Figure 11

4c. Task 1 Conclusions.

- The model met two of the client's conditions: it achieved high test accuracy and showed no signs of overfitting. However, the high accuracy is inflated by correct predictions on dominant classes and true negatives for underrepresented ones.
- Macro F1 score is recommended as a long-term performance metric. It accounts for class imbalance and ensures fair measurements across all topics.

5. Task 2: petition importance classification prototype.

5a. Ethical discussion.

If petitions from certain groups (e.g. Health) are historically overrepresented in “important” labels, the model may learn patterns that marginalize underrepresented causes. As a result, petitions from smaller communities may be misclassified as “not important”. This echoes the *"reinforces existing bias"* hazard. To mitigate this risk, labels should be created using a consistent rubric only assessing petition text. Sampling for manual labelling should aim for diversity across all outcomes. Further bias testing is recommended after model training.
This task aligns with the *Automates Decision Making* hazard. If used in practice, the model could influence how petitions are prioritized, potentially replacing human judgement. To reduce harm, this model should never replace human judgement. It should instead serve as a decision-support tool, paired with transparency and oversight mechanisms.
By assigning petitions to “important” or “not important,” the model effectively ranks citizen voices. Incorrect predictions — especially false negatives — may lead to meaningful causes being dismissed. Fairness audits by department or region would be essential in a production setting.
If embeddings or complex models (e.g., tree ensembles) are used, decisions may be difficult to explain. This aligns with the *Difficult to Understand* hazard. Use of interpretable models or post-hoc explanation tools (e.g., SHAP) can mitigate this. While the model itself is not inherently unethical, it could become harmful if deployed without proper oversight — particularly if it reinforces systemic bias or deprioritizes underrepresented petitions. Responsible use requires human supervision, transparency, and safeguards to prevent misuse.

5b. Data labelling.

Eighty petitions were selected using stratified sampling across four petition statuses: 15 debated, 15 responded, 30 unsuccessful, and 20 rejected to ensure diversity. Petition status was used solely for sampling and no meta data was considered during the labelling process. Figure 12 shows the rubric used; the final label was derived by aggregating the five scores and applying a threshold (above 3 being important).

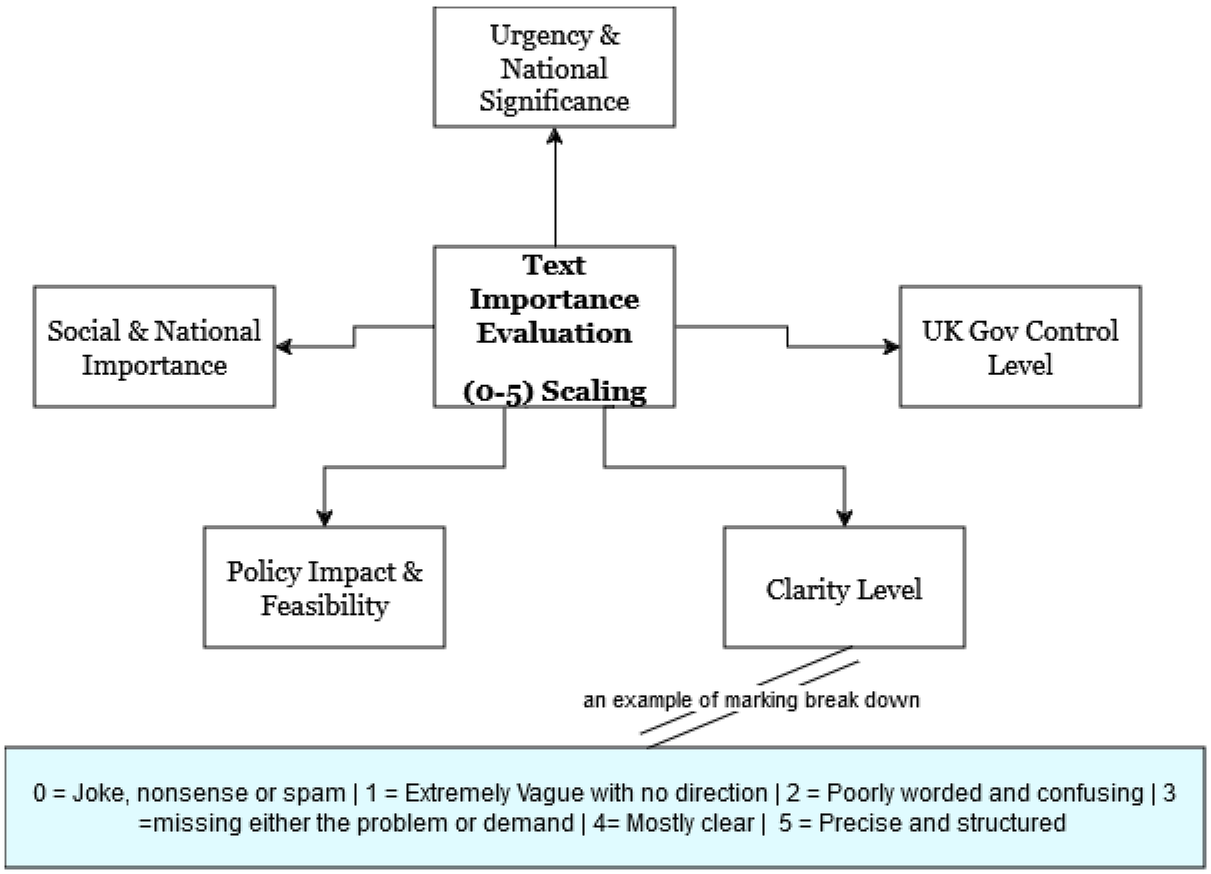

Urgency & National Significance
Text Importance Evaluation (0-5) Scaling
Social & National Importance
UK Gov Control Level
Policy Impact & Feasibility
Clarity Level
an example of marking break down
0 = Joke, nonsense or spam | 1 = Extremely Vague with no direction | 2 = Poorly worded and confusing | 3 =missing either the problem or demand | 4= Mostly clear | 5 = Precise and structured

Figure 12

This method provided a transparent and reproducible approach that reduced subjective bias and ensured consistency across examples. By avoiding outcome-based metadata and applying a fixed decision rule, this labelling process mitigates the risk of reinforcing existing bias. Figure 13 shows two example petitions scored using the five-factor rubric. Labelling confidence was lower for borderline cases near the threshold, where second-person judgement was involved; however, the same rubric was consistently applied to reduce subjective bias. Table 8 describes the data frame after adding the newly 80 labeled to the existing ones.

Split	Important count	Not_important
Train	54	31
Test	10	5

Table 8

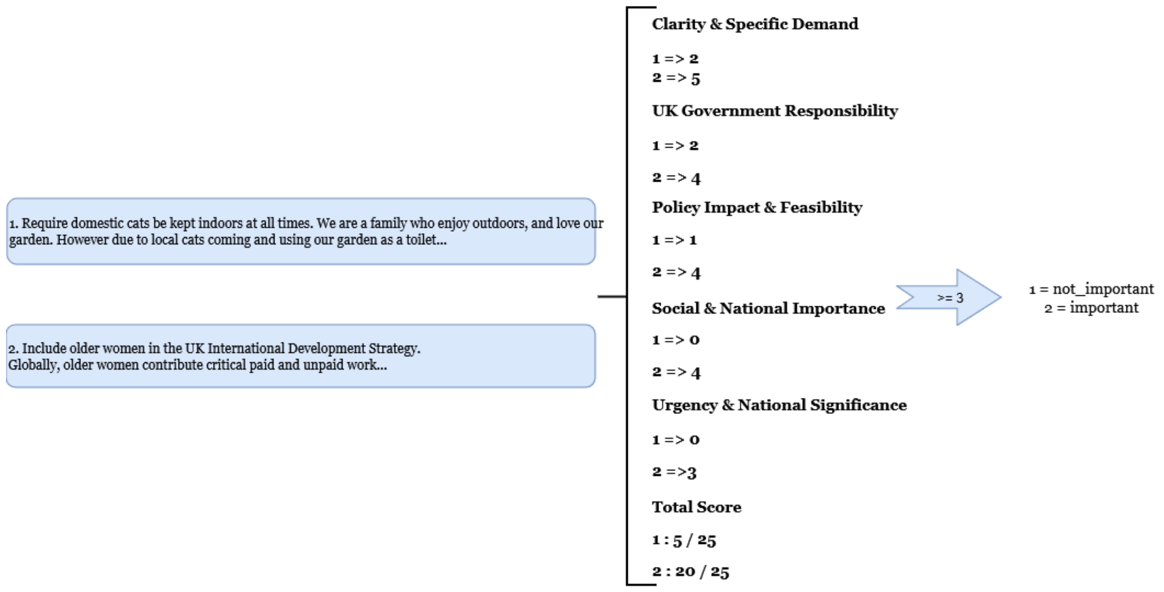

1. Require domestic cats be kept indoors at all times. We are a family who enjoy outdoors, and love our garden. However due to local cats coming and using our garden as a toilet...
2. Include older women in the UK International Development Strategy. Globally, older women contribute critical paid and unpaid work...
Clarity & Specific Demand
1 => 2
2 => 5
UK Government Responsibility
1 => 2
2 => 4
Policy Impact & Feasibility
1 => 1
2 => 4
Social & National Importance
1 => 0
2 => 4
Urgency & National Significance
1 => 0
2 =>3
Total Score
1 : 5 / 25
2 : 20 / 25
>= 3
1 = not_important
2 = important

Figure 13

5c. Model building and evaluation.

Given the small dataset and the need for transparency, we avoided high-complexity models as they invite the *"Difficult to Understand"* data hazard label. Instead, XGBoost was opted to balances interpretability with the ability to uncover meaningful patterns. It allowed us to preserve nulls in deviation_across_regions as meaningful signals.

Feature Selection

Different feature combinations were evaluated with ethical integrity (see Table 9). petition_status_encoded was excluded due to high risk of label leakage and institutional bias — and its removal had no adverse effect on performance. It was also considered to drop deviation_across_regions due to urban bias concerns; however, it proved useful when combined with text and topic features. Ultimately, it was retained with caution, ensuring it did not dominate model predictions. The final feature set included petition_text_encoded, petition_topic_encoded, and deviation_across_regions. We excluded relevant_department due to its high risk of encoding structural bias, potentially leading the model to favour certain policy areas. Similarly, has_named_entity was omitted as its signal is already captured by the full petition text. Deviation_across_regions for task 2 was left unscaled to preserve the meaning of its values. Nulls indicate no signature count, while zero would incorrectly imply no deviation, and -1 would introduce an artificial out-of-range value.

Features	Model Configuration	Precision Class 0	Recall Class 0	F1 Score Class 0
Deaviation_across_regions, topic_encoded, status_encoded, text_encoded	n_estimators=100, learning_rate=0.1	0.70	0.70	0.70
Topic_encoded, status_encoded, text_encoded	n_estimators=100, learning_rate=0.1	0.71	1.00	0.83
Deaviation_across_regions, topic_encoded, text_encoded	n_estimators=100, learning_rate=0.1	0.70	0.70	0.70

Table 9

Model

Initially a grid search with 5-fold stratified cross-validation was used to explore hyperparameters (Table 10). Class imbalance (Class 0: 54, Class 1: 31) was handled using scale_pos_weight ≈ 0.57. It caused major overfitting (F1 class 0 score Gap = 0.33), likely due to deep trees, low learning rate, and excessive parameter tuning on a small dataset.

hyperparameter	Range	Best Value	Description
N_estimator	[50,100,200]	100	Controls model capacity
Learning_rate	[0.05 , 0.1 , 0.2]	0.05	Balances training speed and stability
Max_depth	[2 , 4 , 6]	4	Deeper trees capture more structure but increase complexity
Min_child_weight	[1 , 3]	1	Reduce overfitting by limiting tree splits
Gamma	[0 , 0.1]	0	Encourages pruning
Subsample	[0.7 , 0.8]	0.8	Adds randomness for regularisation
Colsample_bytree	[0.7 , 0.8]	0.7	Helps prevent reliance on dominant features
Reg_lambda	[1 , 5]	1	Prevents large weights
Reg_alpha	[0 , 1]	0	Encourages sparsity

Table 10

Therefore, an ultra-simple XGBoost model was used to reduce overfitting through shallow trees (max_depth=1), fewer estimators (30), and class weighting. As shown in Tables 11 and 12 there is no overfitting (Overfit Gap (Train - Test F1) = 0.0104).

Train	precision	recall	f1 score	support
important	0.73	1.00	0.84	54
not_important	1.00	0.35	0.52	31

Table 11

Test	precision	recall	f1 score	support
important	0.71	1.00	0.83	10
not_important	1.00	0.20	0.33	5

Table 12

Evaluation

As shown Figure 14, the final model identified all important petitions (recall = 1.00), while misclassifying 4 out of 5 "not important" examples. The model's ROC-AUC score on the test set was 0.79, and log loss = 0.5377, indicating reasonably confident probabilistic outputs (average confidence = 0.69).

Compared to a majority class baseline—which would achieve 66.7% accuracy but fail to distinguish class boundaries—our model performs substantially better on the task that matters most. Given the small, imbalanced dataset and the client's priority of not missing high-value petitions, this performance is a strong result. The trade-off is acceptable in a system where human review follows initial filtering.

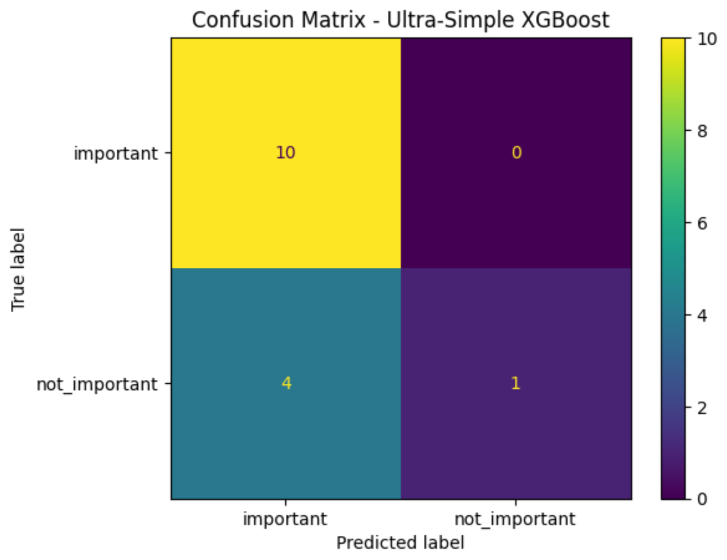


Figure 14

5d. Task 2 Conclusions.

- The model meets the client's success criteria by achieving recall = 1.00 and not missing any important texts, even at the cost of some false positives.
- While it suits exploring applied ML, several limitations remain; Subjectivity of importance, oversimplifies binary framing, Real-world use unclear (whether it uses ranking, filtering ...).
- Expand labelled dataset for semi-supervised learning, enabling use of explainability tools such as SHAP or LIME or use fairness-aware classifiers (like Fairlearn, AIF360).

6. Self-reflection.

It was challenging to align the model architecture with the way the task was framed. In hindsight, a multi-label approach would have been more appropriate, given the overlap between petition topics. Ethical considerations were also difficult to apply meaningfully, especially when deciding how fairness and bias should influence modelling choices.

7. References.

Analytics Educator. (n.d.). *Python Advanced*. [online] Available at: https://www.analyticseducator.com/Courses/PythonAdvance.html [Accessed 17 Mar. 2025].

Data Hazards. (n.d.). *Data Hazards Labels*. [online] Available at: https://datahazards.com/ [Accessed 16 Mar. 2025].

Data Overload. (2023). *Comparing XGBoost and LightGBM: A Comprehensive Analysis*. [online] Medium. Available at: https://medium.com/@data-overload/comparing-xgboost-and-lightgbm-a-comprehensive-analysis-9b80b7b0079b [Accessed 16 Mar. 2025].

Gov.uk. (n.d.). *Petition the UK Government and Parliament*. [online] Available at: https://www.gov.uk/petition-government [Accessed 4 Mar. 2025].

Guide to Procedure. (n.d.). *Petitions*. [online] UK Parliament. Available at: https://guidetoprocedure.parliament.uk/collections/cDdOWeLu/petitions [Accessed 4 Mar. 2025].

Kaggle. (n.d.). *Classification with NLP, XGBoost and Pipelines*. [online] Available at: https://www.kaggle.com/code/diveki/classification-with-nlp-xgboost-and-pipelines [Accessed 16 Mar. 2025].

Kaggle. (n.d.). *NLP: XGBoost, BERT, GloVe, RNNs, Vectorizers*. [online] Available at: https://www.kaggle.com/code/namansood/nlp-xgboost-bert-glove-rnns-vectorizers [Accessed 15 Mar. 2025].

Kaggle. (n.d.). *BiLSTM for Text Classification*. [online] Available at: https://www.kaggle.com/code/lapidshay/bilstm-for-text-classification [Accessed 7 Mar. 2025].

LeanWisdom. (n.d.). *The Ethical Challenges in Data Science*. [online] Available at: https://www.leanwisdom.com/blog/the-ethical-challenges-in-data-science/ [Accessed 17 Mar. 2025].

NeurIPS. (2017). *LightGBM: A Highly Efficient Gradient Boosting Decision Tree*. [online] Proceedings of the 31st Conference on Neural Information Processing Systems. Available at: https://proceedings.neurips.cc/paper_files/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf [Accessed 19 Mar. 2025].

Petitions – UK Parliament. (n.d.). *Petition the UK Government and Parliament*. [online] Available at: https://petition.parliament.uk/ [Accessed 4 Mar. 2025].

RSS (Royal Statistical Society). (2019). *A Guide for Ethical Data Science*. [pdf] Available at: https://rss.org.uk/RSS/media/News-and-publications/Publications/Reports%20and%20guides/A-Guide-for-Ethical-Data-Science-Final-Oct-2019.pdf [Accessed 16 Mar. 2025].

Schneppat, D. (n.d.). *Bidirectional Long Short-Term Memory (BiLSTM)*. [online] Available at: https://schneppat.com/bidirectional-long-short-term-memory_bilstm.html [Accessed 7 Mar. 2025].

ScienceDirect. (n.d.). *Bidirectional Long Short-Term Memory Network*. [online] Available at: https://www.sciencedirect.com/topics/computer-science/bidirectional-long-short-term-memory-network [Accessed 7 Mar. 2025].

DagsHub. (n.d.). *RNN, LSTM and BiLSTM Explained*. [online] Available at: https://dagshub.com/blog/rnn-lstm-bidirectional-lstm/ [Accessed 7 Mar. 2025].

Restack. (n.d.). *Natural Language Processing with XGBoost*. [online] Available at: https://www.restack.io/p/natural-language-processing-knowledge-xgboost-nlp-cat-ai [Accessed 17 Mar. 2025].