Report

On Multi-class Medical Severity Classification
from Patient–Doctor Conversations Using TF-IDF, GloVe, and BERT

Submitted by

Mr.	Perawis	Buranasing	6688012
Miss	Supitsara	Tanasarnsukstid	6688121
Mr.	Atichat	Kangsamut	6688193

Submitted to

Asst. Prof. Charnyote Pluempitiwiriyawej, Ph.D.

This report is part of
ITCS348 Introduction to Natural Language Processing, Semester 2 2025
The Faculty of Information and Communication Technology, Mahidol University

1. Introduction

Medical services often need to decide **how urgent a case is** before a patient meets a doctor. In online health platforms, patients write short descriptions of symptoms and concerns. If a system can read these texts and **estimate severity**, it can help with early triage. This does not replace clinicians. It is a decision-support tool that can help organize cases and reduce delays for more serious cases.

This project studies **multi-class text classification** for medical severity. The goal is to classify each patient case into **three severity levels: low, medium, or high**. The input is written English text from patient-doctor conversations. The output is a severity label. Because the labels are discrete categories, this is a supervised learning problem.

To solve this task, classical machine learning models and neural models are compared. The classical baselines use **TF-IDF with linear classifiers**. The deep learning baseline is an FNN/MLP model that uses mean-pooled GloVe embeddings as input. **BERT fine-tuning** is also included. These models represent different levels of complexity.

The report is organized as follows. First, the classification problem and evaluation approach are defined. Next, this report describes the dataset and exploratory data analysis (EDA), including **label imbalance**. Then the preprocessing experiments and the final chosen text pipeline are explained. After that, the feature representations used by different model families are presented. Finally, model implementations, hyperparameter tuning, and **evaluation results on validation and test sets** are described.

2. Problem Definition

2.1 Task definition

The task is multi-class severity classification with three classes:

- low severity --> label 0
- medium severity --> label 1
- high severity --> label 2

Each example is a short medical text created from the dataset fields. The main text used for modeling is text_raw, which is built by combining the Description and Patient fields into one input string (the Doctor field is not used as model input in this pipeline). The goal is to predict the correct severity label from this text.

Formally:

- Input: a text sequence x (patient case text)
- Output: a label $y \in \{0,1,2\}$

2.2 Why macro-averaged metrics matter

The dataset is imbalanced: medium severity is the largest class, while high severity is the smallest. In such cases, a model can achieve reasonable accuracy by focusing on the majority class. For this reason, the following metrics are reported:

- Accuracy
- Macro-Precision
- Macro-Recall
- Macro-F1

Macro-averaging treats each class equally by averaging the metric across classes. This is important here because correct detection of high severity cases is meaningful even if they are fewer.

2.3 Experimental protocol

The dataset is split into Train / Validation / Test using a stratified 70/15/15 split, so each split keeps a similar label distribution. Models are trained on the training set. Hyperparameters are chosen using the validation set (and cross-validation inside GridSearchCV for some models). Final reporting includes both validation performance (for tuning comparisons) and test performance (for generalization evaluation).

3. Dataset and Exploratory Data Analysis (EDA)

3.1 Dataset source and structure

The dataset is loaded from:

- hf://datasets/mahfoos/Patient-Doctor-Conversation/pred_status.csv

The raw file has 4 columns:

- Description
- Doctor
- Patient
- Status

In the modeling pipeline, the label is taken from Status, and the text input uses Description + Patient as the main text signal. The data is filtered to keep only the three target classes (low/medium/high). After filtering, the total number of samples used in this experiment is:

- Total samples used (3-class): 3320
- Original columns: Description, Doctor, Patient, Status
- Input text: text_raw = Description + Patient
- Classes: low/medium/high (3 classes), mapped to 0/1/2

3.2 Train/Validation/Test split and label distribution

The stratified split sizes are:

- Train: 2323
- Validation: 498
- Test: 499

Label distribution is consistent across splits (by design of stratified sampling). The overall class proportions are also stable:

- medium severity: 53.58% (1779 / 3320)
- low severity: 32.17% (1068 / 3320)
- high severity: 14.25% (473 / 3320)

For training data, the imbalance ratio (largest class / smallest class) is:

- 1245/331≈3.76

Class	Total (n, %)	Train (n, %)	Val (n, %)	Test (n, %)
low severity	1068 (32.17%)	747 (32.16%)	160 (32.13%)	161 (32.26%)
medium severity	1779 (53.58%)	1245 (53.59%)	267 (53.61%)	267 (53.51%)
high severity	473 (14.25%)	331 (14.25%)	71 (14.26%)	71 (14.23%)

Table 3.2: Label distribution (counts + %; total and split) | Source: exported split CSVs (tr_df.csv, va_df.csv, te_df.csv)

EDA insight 1 (imbalance): Medium severity is the majority class. High severity is the minority class, which can make it harder for models to learn reliable patterns for high severity without overfitting.

3.3 Text length analysis

Text length is measured as number of tokens using len(str(text).split()). The overall token-length statistics are:

- Train mean: 122.67 tokens (median 101)
- Validation mean: 116.58 tokens (median 98)
- Test mean: 128.32 tokens (median 104)

This suggests the test set has slightly longer texts on average.

Split	count	min	p25	median	mean	p75	max	std
Train	2323	13	65.0	101.0	122.67	153.0	852	88.83
Val	498	17	63.25	98.0	116.58	151.0	619	77.34
Test	499	17	66.0	104.0	128.32	159.0	1003	97.80

Table 3.3: Overall length stats (Tokens) | Source: experiment.ipynb Section 3

EDA insight 2 (length variation): Text length varies widely. The maximum length is large (up to 1003 tokens in test). This motivates truncation for transformer models and suggests that short and long cases may behave differently in classification.

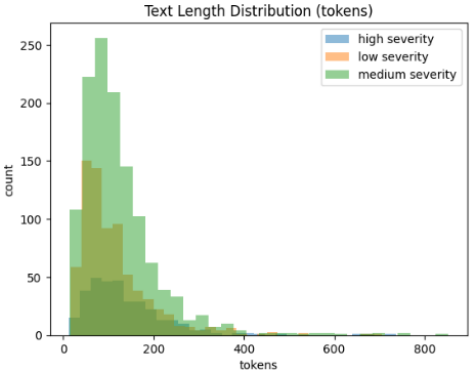


Figure 3.3: Text length distribution plot (per class)
Source: experiment.ipynb Section 3 (length histogram)

3.4 Vocabulary statistics and lexical diversity

Vocabulary statistics are computed per class (training set). Two useful values are:

- vocab_size: number of unique tokens observed in that class
- TTR (type-token ratio): lexical diversity indicator (higher TTR means more unique words relative to total words)

Class	num_docs	vocab_size	TTR
high severity	331	4852	0.105667
low severity	747	6143	0.072722
medium severity	1245	8385	0.054245

Table 3.4: Vocabulary stats per class (vocab_size, TTR) | Source: eda_df.csv (vocab_size/TTR per class)

EDA insight 3 (vocab and TTR): Medium severity has the largest vocabulary size, likely because it has the most documents. However, its TTR is the lowest. High severity has fewer documents, smaller vocabulary, but the highest TTR, suggesting higher lexical variety relative to its size.

3.5 Frequent words and class-specific keywords

To understand what the model might learn, word frequency by class is analyzed after removing stopwords. The most frequent content words are still general conversational words (for example “doctor”, “hello”, “please”), but symptom-related terms also appear (for example “pain”, “blood”, “test”, “back”).

Class	words
high severity	“pain”, “blood”, “test”, “months”, “back” (among top words)
low severity	“pain”, “last”, “time”, “day”, “back” (among top words)
medium severity	“pain”, “days”, “normal”, “back”, “last” (among top words)

Table 3.5.1: Example Top words per class (after stopword removal) | Source: experiment.ipynb Section 3 EDA “Example Top 20 words per class (remove stopwords)”

Because frequency alone can be dominated by general words, a PMI-like scoring method is also used to find more class-specific keywords. This method compares the probability of a word inside one class to its overall probability, using a log ratio score.

Class	Word	PMI-like Score	Freq (Class)	Freq (All)
High severity	suicidal	1.729001	11	12
High severity	platelet	1.698229	8	9
High severity	radiculopathy	1.682481	7	8
Low severity	pigmentation	1.225294	15	15
Low severity	blister	1.225294	8	8
Low severity	retainer	1.225294	14	14
Medium severity	coronary	0.609439	10	10
Medium severity	appendicitis	0.609439	5	5
Medium severity	choking	0.609439	5	5

Table 3.5.2: Top 3 PMI-like keywords per class | Source: experiment.ipynb Section 3 EDA “PMI-like word score”

EDA insight 4 (keyword patterns): PMI-like keywords tend to pick words that are rare overall but appear more strongly in one class. These words may represent specific conditions, medications, or clinical terms that are more common in a severity category.

EDA insight 5 (overlap risk): Many high-frequency words overlap across classes (for example “pain”, “days”, “help”), which suggests that severity prediction depends on combinations of symptoms, context, and intensity rather than single keywords only.

4. Text Preprocessing

4.1 Baseline text used for modeling

The pipeline creates a unified text field (text_raw) and then tests several preprocessing options. The baseline TF-IDF experiments use raw casing control via lowercase=False in the vectorizer, because preprocessing decisions are handled explicitly in preprocessing functions during ablation.

4.2 Preprocessing operations considered

The ablation experiment tests four variants:

- A_raw: no lowercasing, no punctuation removal, no stopword removal, no lemmatization
- B_clean: lowercasing + punctuation removal (regex), whitespace normalization
- C_stop: B_clean + stopword removal
- D_stop_lemma: C_stop + lemmatization (spaCy)

These steps are implemented in a single preprocessing function with boolean flags:

- lowercasing
- punctuation removal with regex
- stopword removal
- lemmatization

4.3 Ablation results and choice

The ablation uses the same classifier for comparison (Logistic Regression) and the same TF-IDF setting (ngram 1–2, min_df=2). The metrics reported are macro-F1 and vocabulary size, plus vocabulary reduction percentage compared to the raw baseline.

Experiment	macro-F1	vocab_size	vocab_reduction_%
A_raw	0.479090	31660	0.0000
B_clean	0.479090	31660	0.0000
D_stop_lemma	0.468989	18699	40.9381
C_stop	0.462282	18137	42.7132

Table 4.3: Preprocessing ablation (experiment, macro-F1, vocab size, vocab reduction %) | Source: ablation_df.csv + experiment.ipynb Section 4 (preprocessing ablation code/output)

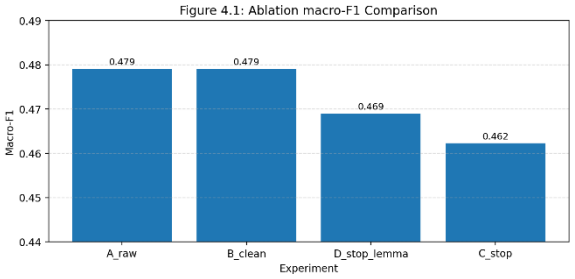


Figure 4.3: Ablation macro-F1 comparison plot | Source: Table 4.3

Interpretation:

- The best macro-F1 is achieved by A_raw and B_clean, which are tied.
- Stopword removal and lemmatization reduce vocabulary size strongly (about 41–43%), but they slightly reduce macro-F1 in this setup.

Final preprocessing decision:

For the main model comparisons, preprocessing is kept minimal (raw or clean), because it gives the best validation macro-F1 in the ablation. However, cleaned text is still useful for EDA and keyword inspection because it normalizes casing and punctuation.

5. Feature Extraction

5.1 TF-IDF (sparse n-gram features)

TF-IDF converts text into a sparse vector where each dimension is a word or n-gram. Two baseline configurations are tested on the train split:

- TF-IDF (1–2 grams), min_df=2
  - Shape: (2323, 31660)
  - Sparsity: 99.5559%
- TF-IDF (1–3 grams), min_df=2
  - Shape: (2323, 50113)
  - Sparsity: 99.6591%

In tuning, the best Logistic Regression configuration uses a larger n-gram range and includes rarer terms, whereas the best Naive Bayes configuration relies on a smaller vocabulary.

- LR tuned TF-IDF (1–3 grams), min_df=1
  - Shape: (2323, 318081)
  - Sparsity: 99.9100%
- NB tuned TF-IDF (1–1 grams), min_df=2
  - Shape: (2323, 6079)
  - Sparsity: 98.8366%

These results show that TF-IDF matrices are extremely sparse, especially when the vocabulary grows.

5.2 GloVe mean embeddings (dense 100-dimensional vectors)

For the deep learning baseline (FNN/MLP), each text is converted into a dense vector:

- Pre-trained embedding: glove-wiki-gigaword-100 (Pennington et al., 2014)
- Dimension: 100
- Representation: mean pooling over token embeddings
- If no known tokens are found, the vector is set to zeros.

This produces a dense matrix: Shape: (2323, 100)

5.3 BERT tokenization (contextual input)

For BERT fine-tuning, the input is tokenized with:

- Tokenizer: bert-base-uncased
- Truncation: True
- Maximum length: 512
- Average token length (train): 135.39
- Maximum token length (train): 512

This confirms that some samples reach the limit, so truncation is necessary for training stability.

Representation	Used by	Dimensionality	Sparsity	Notes on memory/size
TF-IDF(1–2), min_df=2	LR/SVM/NB (baseline)	31660	~99.56%	Sparse, moderate vocab
TF-IDF(1–3), min_df=2	feature comparison	50113	~99.66%	Sparse, larger vocab
TF-IDF(1–3), min_df=1	LR tuned	318081	~99.91%	Very large sparse vocab
TF-IDF(1–1), min_df=2	NB tuned	6079	~98.84%	Smaller vocab than LR tuned
GloVe mean (100d)	FNN/MLP	100	0% (dense)	Compact dense vectors
BERT tokens	BERT	≤512 tokens	n/a	Contextual, truncation at 512

Table 5.3: Feature representation summary (dimensionality, sparsity, memory cost)
Source: experiment.ipynb Section 5 (TF-IDF/GloVe/BERT stats) + user-run code cell "TF-IDF tuned stats"

6. Model Implementation

6.1 Classical ML baselines (TF-IDF + linear models)

All classical baselines use TF-IDF features and are trained on the training set, then evaluated on the validation set.

Logistic Regression (LR)

- Model: LogisticRegression
- Key settings: class_weight="balanced", max_iter=500 (baseline)
- Motivation: strong baseline for sparse text features; handles linear separation.

Linear SVM

- Model: LinearSVC
- Key settings: class_weight="balanced"
- Motivation: often strong for high-dimensional sparse vectors.

Multinomial Naive Bayes (NB)

- Model: MultinomialNB()
- Key settings: default in baseline
- Motivation: simple probabilistic baseline; fast and commonly used for text.

6.2 Deep learning baseline: FNN/MLP (GloVe mean embeddings)

The deep learning baseline is a small feed-forward neural network (MLP) trained on mean-pooled GloVe embeddings.
Input: 100-dimensional dense vector per text

Network (baseline):

- Linear(100 --> 128)
- ReLU
- Dropout(0.3)
- Linear(128 --> 3)

Training setup (baseline):

- Loss: CrossEntropyLoss
- Optimizer: Adam (lr = 1e-3)
- Epochs: 10
- Batch size: 32
- Device: GPU if available

This model is a simple neural baseline that can learn non-linear patterns beyond linear classifiers, but it relies on static embeddings and does not model word order deeply.

6.3 Transformer model: BERT fine-tuning

The BERT model (Devlin et al., 2019) is fine-tuned as a sequence classification model.

Tokenizer and model:

- Tokenizer: bert-base-uncased
- Model: AutoModelForSequenceClassification("bert-base-uncased", num_labels=3)

Training configuration:

- Epochs: 10
- Learning rate: 3e-5
- Evaluation: per epoch
- Save: per epoch
- Batch size: 16 (train and eval)
- Load best model at end: True, based on macro_f1
- Early stopping: patience = 2
- Metrics computed: accuracy, macro-F1, precision, recall

This setup fine-tunes all model parameters on the severity task. Compared to TF-IDF models, BERT can use context and capture richer language patterns.

7. Hyperparameter Tuning and Evaluation Results

7.1 Validation results (baseline models)

Baseline validation results compare the five model families under a consistent evaluation approach.

Model	Accuracy	Macro-Precision	Macro-Recall	Macro-F1
Logistic Regression	0.540161	0.477200	0.481675	0.479090
Linear SVM	0.534137	0.424307	0.408701	0.405891
Multinomial NB	0.536145	0.178715	0.333333	0.232680
FNN (GloVe mean)	0.564257	0.385527	0.379198	0.337035
BERT	0.578313	0.528295	0.494542	0.505512

Table 7.1: Validation results (all models): Accuracy, Macro-Precision, Macro-Recall, Macro-F1
Source: results_df.csv + experiment.ipynb Section 6 (validation evaluation outputs)

Validation interpretation:

- BERT achieves the best macro-F1 on validation (0.5055).
- Logistic Regression is the strongest classical baseline by macro-F1 (0.4791).
- Naive Bayes performs poorly on macro-F1, mainly because macro metrics penalize weak minority-class performance.
- The FNN/MLP baseline improves accuracy but has lower macro-F1 than LR and BERT, suggesting it may still struggle with minority-class balance.

7.2 Hyperparameter tuning for LR, SVM, and NB

Using GridSearchCV with macro-F1 scoring (3-fold CV training data) to tune classical models. The best parameter sets are:

- LR (tuned): {'clf__C': 0.1, 'tfidf__min_df': 1, 'tfidf__ngram_range': (1, 3)}
- SVM (tuned): {'clf__C': 0.1, 'tfidf__min_df': 2, 'tfidf__ngram_range': (1, 2)}
- NB (tuned): {'clf__alpha': 0.1, 'tfidf__min_df': 2, 'tfidf__ngram_range': (1, 1)}

Model	Best params (summary)	Accuracy	Macro-Precision	Macro-Recall	Macro-F1
LR (tuned)	C=0.1, ngram=(1,3), min_df=1	0.522088	0.477261	0.487885	0.476687
Linear SVM (tuned)	C=0.1, ngram=(1,2), min_df=2	0.572289	0.497061	0.435140	0.436589
Multinomial NB (tuned)	alpha=0.1, ngram=(1,1), min_df=2	0.600402	0.575954	0.431212	0.416140

Table 7.2: Tuned ML results + best params | Source: tuned_ml_df.csv + experiment.ipynb Section 7 (GridSearchCV outputs)

Tuning interpretation:

- Tuning improves SVM and NB compared to their baselines, especially in accuracy and precision.
- LR tuned is close to the LR baseline macro-F1 but not higher on validation.
- A key reason is that LR baseline is already strong and macro-F1 is sensitive to minority-class behavior.

7.3 FNN/MLP tuning (GloVe mean)

FNN tuning tests combinations of:

- hidden_dim ∈ {64, 128, 256}
- dropout ∈ {0.2, 0.3, 0.5}
- learning rate ∈ {0.001, 0.0005}
- batch size ∈ {32, 64}

Best configuration (by validation macro-F1):

- hidden_dim=256, dropout=0.2, lr=0.001, batch_size=32
- val_macro_f1 = 0.410495

This tuned macro-F1 is higher than the baseline FNN macro-F1 (0.3370), showing that model capacity and regularization matter even for simple neural baselines.

7.4 BERT tuning grid

BERT tuning tests learning rate and batch size (epochs fixed at 10). Results are:

lr	batch_size	epochs	val_accuracy	val_macro_f1	val_loss
3e-5	16	10	0.580321	0.518154	1.484953
2e-5	8	10	0.616466	0.506041	0.880834
2e-5	16	10	0.590361	0.498468	1.075962
3e-5	8	10	0.604418	0.472621	0.894638

Table 7.4: BERT tuning grid + best config | Source: bert_tune_df.csv

Best by macro-F1: lr = 3e-5, batch size = 16, epochs = 10

7.5 Test results (generalization)

After selecting configurations, the models are evaluated on the held-out test set.

Model	Accuracy	Macro-Precision	Macro-Recall	Macro-F1
LR	0.571142	0.521302	0.533836	0.522610
SVM	0.593186	0.506236	0.462202	0.467086
NB	0.589178	0.587267	0.427837	0.419425
FNN	0.589178	0.717408	0.452683	0.422811
BERT	0.631263	0.592289	0.528923	0.545308

Table 7.5: Comparative Performance of All Classification Models (Test Set) | Source: results_df.csv + macro_df.csv

Key result:

- BERT achieves the best test macro-F1 (0.5453) and the best test accuracy (0.6313).
- LR is the strongest classical model by test macro-F1 (0.5226), but still behind BERT.

7.6 Confusion matrix analysis (best model)

The best test macro-F1 model is BERT. The label order is:

- 0 = low, 1 = medium, 2 = high

BERT test confusion matrix (rows=true, cols=pred):

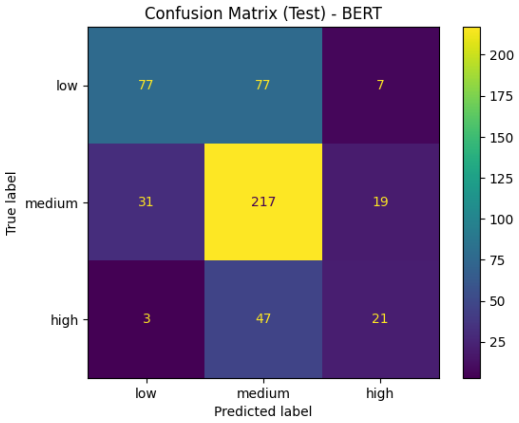


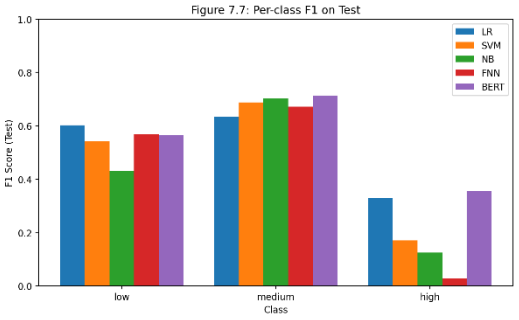
Figure 7.6: Confusion matrix (Test) for best model (BERT)
Source: experiment.ipynb Section 11.3 (BERT_pred vs y_true)

Confusion interpretation (BERT):

- The model predicts medium severity well (217 correct out of 267).
- The hardest confusion is between low and medium (77 low --> medium, and 31 medium --> low).
- High severity has 21 correct out of 71, with many high cases predicted as medium (47). This indicates that distinguishing high from medium remains difficult.

7.7 Per-class performance on test

Per-class F1 helps explain macro-F1 differences. Because high severity is the smallest class, weak high-severity recall can reduce macro-F1 strongly.



A clear pattern across models is that:

- Medium is usually the easiest class.
- High severity is usually the hardest class, especially for linear models and NB.
- BERT improves high-severity performance relative to most baselines, but high vs medium confusion still exists.

Figure 7.7: Per-class F1 on Test (grouped bar) | Source: per_class_df.csv

7.8 Qualitative Error Analysis (with Examples)

7.8.1 Main confusion patterns (from the test confusion matrix): From the test confusion matrix of the best model (BERT), the most frequent mistakes happen between adjacent classes. Low and medium are often confused with each other, and many high cases are predicted as medium. This pattern is expected in severity classification because the boundary between "medium" and "high" can be subtle in short medical text. A case may contain serious keywords but still lack enough context (duration, intensity, vital signs, or risk factors). This makes the model uncertain and pushes predictions toward the middle class.

7.8.2 Misclassified examples (qualitative): To make the error patterns clear, real misclassified test examples are reviewed and explanations are provided for why they are difficult. The examples are grouped by confusion type.

true_name	pred_name	conf	snippet	explanation
low	medium	0.999423	on taking medications my postnatal drip has become thick why hello doctor I am having postnatal...	Class overlap
high	medium	0.999421	I have headache and increased heart rate with increased get and alt what is my problem hi doctor I...	Class overlap

Table 7.8: Misclassified test examples | Source: experiment.ipynb Section 9.7 (examples) / Section 9.6 (errors all table) + examples_df.csv

Explanation column labels:

- Class overlap: the true label is adjacent to the predicted label (low ↔ medium or medium ↔ high).
- Linguistic ambiguity: unclear wording, missing context, or mixed signals (negation/uncertainty).
- Symptom severity confusion: the text contains severe cues but the overall case is not clearly high (or the opposite).
- Other/uncategorized: unusual or rare cases.

From Example snippets that show "high-confidence wrong" behavior: Some errors are especially risky because the model is confident but wrong (for example, predicting high severity for a low case, or predicting low severity for a high case). These cases should be highlighted because they can matter more in a real triage setting.

7.9 Deeper Comparative Analysis (Strengths and Weaknesses Across Models)

7.9.1 Effect of class imbalance and macro metrics: The dataset is imbalanced, so macro-averaged metrics are important. A model can look good in accuracy by focusing on the majority class (medium), but still fail on high severity. For this reason, macro-precision, macro-recall, and macro-F1 are reported in addition to accuracy.

7.9.2 High severity detection is the main difficulty: Across models, medium severity is usually the easiest class. High severity is usually the hardest class, and performance on high strongly affects macro-F1. From the per-class results on the test set, BERT improves high-severity F1 compared to most baselines, but high recall is still limited. The confusion matrix also shows that many true high cases are predicted as medium, which means the model tends to "under-triage" severe cases into the middle category. This suggests that a key limitation is the boundary between medium and high. In text-only data, missing clinical detail can make the boundary hard even for BERT.

7.9.3 Why BERT performs best (but not perfect): BERT achieves the best test macro-F1 and accuracy among the compared models. A likely reason is that BERT can use context and word interactions better than TF-IDF and static embeddings, which helps when severity depends on combinations of symptoms and wording. However, BERT is not perfect. The remaining errors are concentrated in adjacent-class overlap and linguistic ambiguity (unclear or incomplete symptoms), which are hard problems for any model without extra clinical features.

7.9.4 Cost vs performance trade-off (practical view): A practical comparison should also consider training and inference cost. BERT has much higher training time than classical baselines (seconds for LR/SVM/NB versus hundreds of seconds for BERT in this run). This means BERT is the best choice when accuracy and macro-F1 are the main priority, but classical models are attractive when compute and latency are strict constraints.

model_key	train_time_sec	num_params	model_size_mb	inf_time_sec	samples_per_sec
LR	4.462953	1114437	19.11052	0.127847	3903.101
SVM	1.009071	110970	1.68222	0.073881	6754.062
NB	0.327901	20058	0.440515	0.034484	14470.52
FNN	1.52928	26627	0.101574	0.000464	1076339
BERT	490.7962	1.09E+08	417.6504	8.30108	60.11266

Table 7.9: Cost comparison (train time, parameters/model size, inference speed)
Source: experiment.ipynb Section 12.4–12.5 (cost table build) / cost_df.csv

7.10 Interpretability

7.10.1 Interpreting NB with top indicative words: For Multinomial Naive Bayes, top indicative tokens were inspected to interpret class-specific evidence. This gives a simple explanation of what words strongly push predictions. For example, some high-severity indicative tokens include words such as "suicidal" and other severe-condition terms, while low-severity tokens include more minor-condition or everyday health terms. This supports the idea that NB relies heavily on keyword evidence, which can fail when severity depends on context rather than single words.

Class	Token	Score	Interpretation
Low	retainer	1.8323	Dental/orthodontic term, usually routine care.
Medium	shivering	1.2617	Symptom that may indicate infection or systemic issue but not always critical.
High	suicidal	2.2262	Strong indicator of severe psychiatric emergency (high-risk case).

Table 7.10.1: Example of Top indicative tokens per class (NB) | Source: experiment.ipynb Section "NB: Top indicative features" (interpretability output)

7.10.2 Interpreting LR/SVM with feature weights (overview): For LR and Linear SVM, the most positive and negative TF-IDF features can also be extracted from the learned coefficients. This is helpful to understand decision boundaries, especially between low and medium.

Model	Class	Top Positive Feature	Weight
LR	Low	skin	0.2121
LR	Medium	pain	0.2302
LR	High	cancer	0.2777

Model	Class	Top Positive Feature	Weight
SVM	Low	teeth	0.5744
SVM	Medium	pain	0.5619
SVM	High	cancer	0.9393

Table 7.10.2: Example of Top weighted TF-IDF features for LR/SVM (per class) | Source: experiment.ipynb SECTION 10.1 (Linear models: top features)

7.10.3 Limits of interpretability for BERT: BERT provides the best performance but is harder to interpret directly because it uses deep contextual representations. Therefore, this experiment uses interpretability mainly to understand baseline behavior and error patterns, while BERT is assessed primarily through robust evaluation, confusion matrix, and qualitative error analysis.

8. Ethical Considerations

8.1 Risks of misclassification: The most important risk is under-triage, where a true high severity case is predicted as medium or low. In the confusion matrix analysis, many high cases are predicted as medium. If used in real settings, this could delay urgent care. There is also over-triage, where low cases are predicted as high. This can increase workload for doctors and reduce trust in the system.

8.2 Bias and fairness concerns: The dataset is imbalanced, with high severity as the minority class. This can bias models toward the majority class (medium). As a result, minority-class recall (high severity) may be lower, which is ethically important because high severity errors can be more harmful. Another risk is language variation. Patient-written text may include incomplete information, spelling issues, and vague symptoms. This can affect different patient groups differently if some groups tend to write shorter or less detailed descriptions.

8.3 Mitigation strategies (how to use the model safely): To reduce risk, the model should be used as decision support, not as an automatic final decision. A safe deployment approach includes:

- Human-in-the-loop: clinicians review model outputs before action.
- Risk-aware thresholds: treat "high" predictions conservatively and review borderline cases.
- Abstain option: if confidence is low or the input is too short/ambiguous, the system should say "needs review" instead of forcing a class.
- Ongoing monitoring: track errors over time, especially high-severity false negatives, and update the model when data changes.

These steps align the model with real-world safety needs and reduce harm from misclassification.

9. Conclusion

This experiment shows that transformer fine-tuning is the most effective approach for 3-class medical severity classification, with BERT achieving the best test performance (Accuracy = 0.6313, Macro-F1 = 0.5453) among all compared models. However, the main remaining difficulty is adjacent-class separation. Especially under-triage where many true high cases are predicted as medium (e.g., high: 21/71 correct, 47 predicted as medium), alongside frequent low ↔ medium confusion. Therefore, the key limitation is missing or ambiguous clinical context in text-only inputs, which pushes predictions toward the middle class; in practical use, high-vs-medium borderline cases should be handled conservatively (e.g., human-in-the-loop review) to reduce harmful under-triage risk.