Medical Referral Urgency Classifier

Product View –

In clinical workflows, the timely triaging of patient referrals is critical. Manual review of referrals, especially in large hospital systems, can delay care for urgent cases and overburden staff. Automating urgency classification using referral documents in PDF form can:

- Ensure faster intervention for high-risk patients.
- Reduce operational load on clinical coordinators.
- Create a scalable pre-screening mechanism for downstream AI decision systems.

This project classifies the **urgency** level of medical referrals using **PDF form data**. The pipeline involves:

- Generating structured data as a CSV file,
- Converting each CSV record into a **synthetic PDF form**,
- Extracting key fields from PDFs,
- Creating a structured dataset,
- Generating text embeddings (TF-IDF or SBERT),
- Applying multiple machine learning models,
- Comparing model performance to identify the best classifier.

Technologies Used -

Python, pandas (structured data handling), reportlab (PDF generation). PyPDF2 (PDF parsing), scikit-learn (ML models, vectorization), sentence-transformers (SBERT embeddings), imblearn (SMOTE for class balancing), XGBoost (advanced ensemble classifier), DistilBERT

1)Synthetic Data Generation

This section outlines how synthetic patient referral data is generated to simulate real-world scenarios across Canadian provinces and medical departments. The data mimics realistic variability in symptoms, diagnostic workflows, urgency determination, and provincial ID formatting.

1. Geographic Representation

Province	Health Card Format	Postal Code Pattern
Ontario	10-digit number	Starts with 'M'
British Columbia	10-digit PHN starting with '9'	Starts with 'V'
Alberta	9-digit Unique Lifetime Identifier (ULI)	Starts with 'T'

2. Department-Specific Configuration
   Each department (e.g., Cardiology, Psychiatry, Ultrasound) defines:
   - A list of **possible symptoms**.
   - A subset of **routine symptoms** to distinguish lower-risk cases.
   - Allowed **test types** (ECG, MRI, etc.).
   - A department-specific **referral summary template** with placeholders for symptoms.

3. Symptom Assignment

A synthetic patient is randomly assigned 1–3 symptoms from a department's list. This introduces:

- Intra-departmental variation (e.g., Cardiology can involve "chest pain" or "fatigue").
- Routine vs. high-risk signals, influencing downstream test/urgency logic.

4. Test Enforcement Logic

- If a patient has **high-risk symptoms** (e.g., "chest pain" in Cardiology), we ensure that a relevant diagnostic **test** is assigned

5. Urgency Classification only for *Challenge 1a*

Urgency is determined using a **symptom-test-department triad**, with tunable randomness to avoid overfitting. These decision-making mimics the **triage complexity** seen in real hospitals, where urgency is not always binary.

- If a case includes **high-risk symptoms** or an **urgent test** (CT, MRI, Biopsy), it's probabilistically labeled Urgent.
- If only **routine symptoms**, it's mostly labeled Routine.
- A balance (urgency_ratio) ensures that both Routine and Urgent cases are adequately represented.

2)Patient Referral PDF Generation

The create_referral_pdf function is used to generate a structured PDF referral form for a patient, incorporating both clinical and demographic information. It appends metadata such as a unique Referral ID and Referral Date, and formats the data in a clean, printable layout. The output consists of a professionally styled PDF saved at the specified path, containing:

- Referral metadata (ID, date)
- Patient demographics
- Medical history and test information
- Referral reasons and urgency labels

3a) Urgency Classification Pipeline

The pipeline classifies referral urgency by leveraging structured data extracted from PDFs. Key steps include:

1) **Feature Construction:** Combine critical clinical fields into a unified text feature -
    - Reason for Referral
    - Medical Condition
    - Symptoms
2) **Text Vectorization:** Transform text into numerical features using –
    - TfidfVectorizer (or optionally, Sentence Transformers for embeddings)
3) **Model Training:** Train multiple classifiers –
    - Logistic Regression
    - Support Vector Machine (SVM)
    - Random Forest
    - XGBoost - (with **SMOTE** applied to handle class imbalance)
4) **Evaluation:**
   Predict urgency labels on test data and evaluate performance using classification_report (precision, recall, F1-score).

3b) Urgency Classification Pipeline

The pipeline classifies referral urgency (routine vs. urgent) using unstructured text extracted from PDFs. Key steps include:

1) **Dataset Preparation**
   - Referral text data was extracted from PDF files, with **each referral classified as either urgent or routine** based on presence of predefined clinical keywords (e.g., "chest pain", "stroke").
   - The text samples (texts) and labels (labels) are shuffled for randomness.
   - A HuggingFace Dataset object is constructed with a predefined schema (ClassLabel for binary classification).
   - The dataset is split into three parts: training set, validation set, and a true holdout test set for final evaluation.

2) **Text Tokenization**
   - Uses distilbert-base-uncased tokenizer to convert raw referral text into model-compatible input (token IDs, attention masks).
   - Padding and truncation are applied to ensure consistent input length.

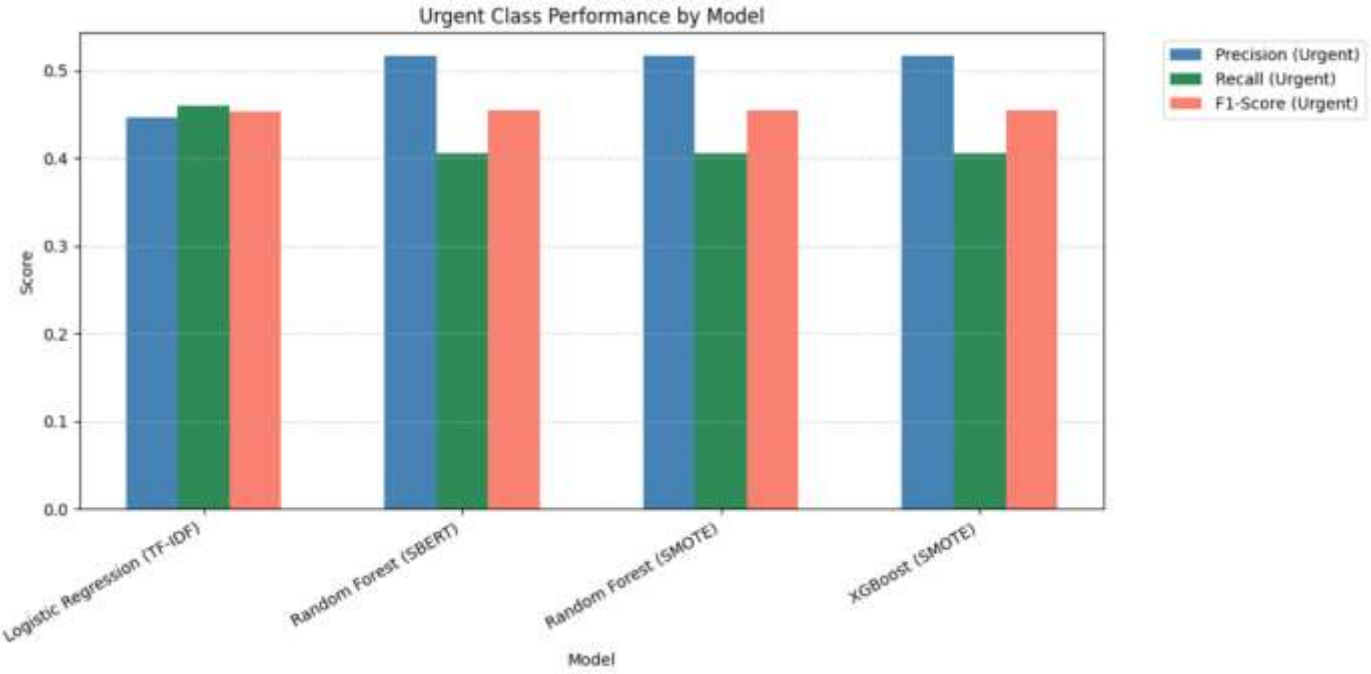
3) **Model Fine-tuning**
   - Loads a pre-trained DistilBERT model (AutoModelForSequenceClassification) with 2 output labels.
   - The model is fine-tuned using the Trainer API with specified training parameters (e.g., epochs, batch size, fp16 support).

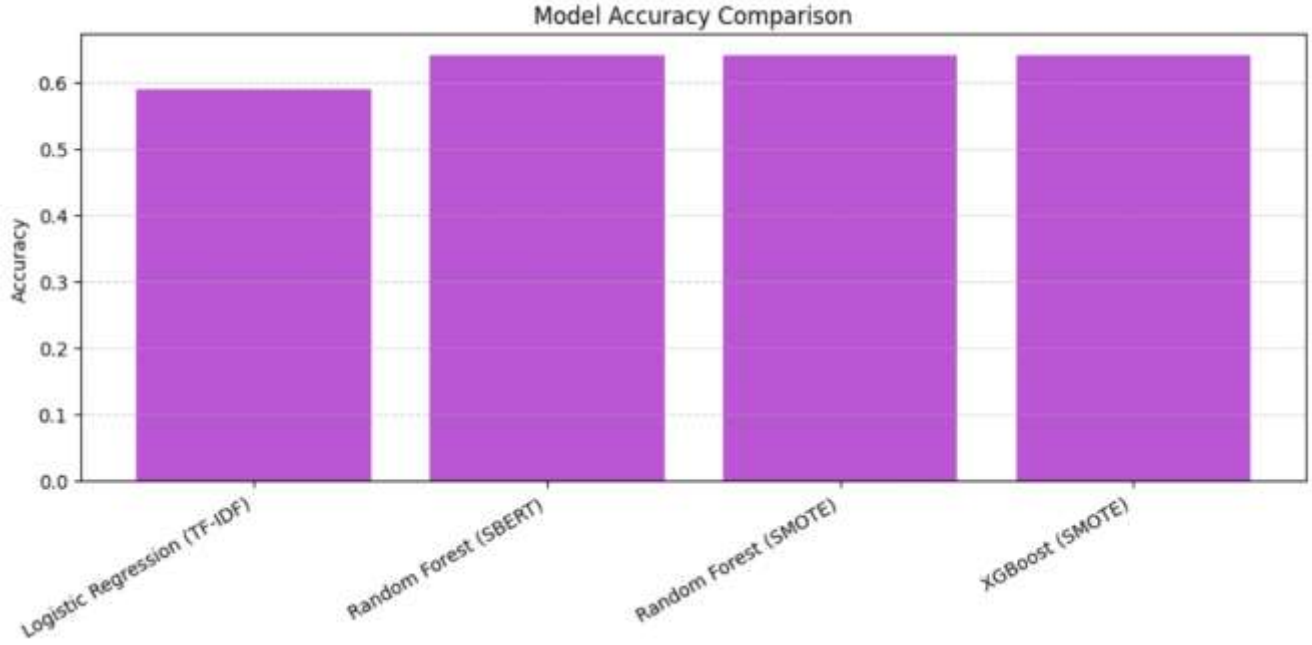
4) **Internal Evaluation (Validation Set)**
   - After training, the model is evaluated on a validation set.
   - The goal was to assess how well the model distinguishes **routine vs urgent** cases before final testing.
   - Generates classification metrics: **accuracy**, **F1 score**, detailed **classification report**, and a **confusion matrix**.

5) **External Evaluation (Holdout Set)**
   - The trained model is further tested on a previously unseen holdout set.
   - This simulates real-world performance and ensures no data leakage.
   - Results are visualized using a heatmap confusion matrix (seaborn + matplotlib).

4a) Results

Model	Precision (Urgent)	Recall (Urgent)	F1-Score (Urgent)	Accuracy
Logistic Regression (TF-IDF)	0.45	0.46	0.45	0.59
Random Forest (SBERT)	0.52	0.41	0.45	0.64
Random Forest (SMOTE)	0.49	0.49	0.49	0.62
XGBoost (SMOTE)	0.49	0.49	0.49	0.62


Urgent Class Performance by Model
Precision (Urgent)
Recall (Urgent)
F1-Score (Urgent)
Score
0.0
0.1
0.2
0.3
0.4
0.5
Logistic Regression (TF-IDF)
Random Forest (SBERT)
Random Forest (SMOTE)
XGBoost (SMOTE)
Model


Model Accuracy Comparison
Accuracy
0.0
0.1
0.2
0.3
0.4
0.5
0.6
Logistic Regression (TF-IDF)
Random Forest (SBERT)
Random Forest (SMOTE)
XGBoost (SMOTE)

4b) Results

```
Evaluation on validation set...

Validation Classification Report:
              precision    recall  f1-score   support

     routine     0.9630    1.0000    0.9811        26
      urgent     1.0000    0.9583    0.9787        24

    accuracy                         0.9800        50
   macro avg     0.9815    0.9792    0.9799        50
weighted avg     0.9807    0.9800    0.9800        50


Validation Confusion Matrix:
                 Predicted Routine  Predicted Urgent
Actual Routine                  26                 0
Actual Urgent                    1                23
```

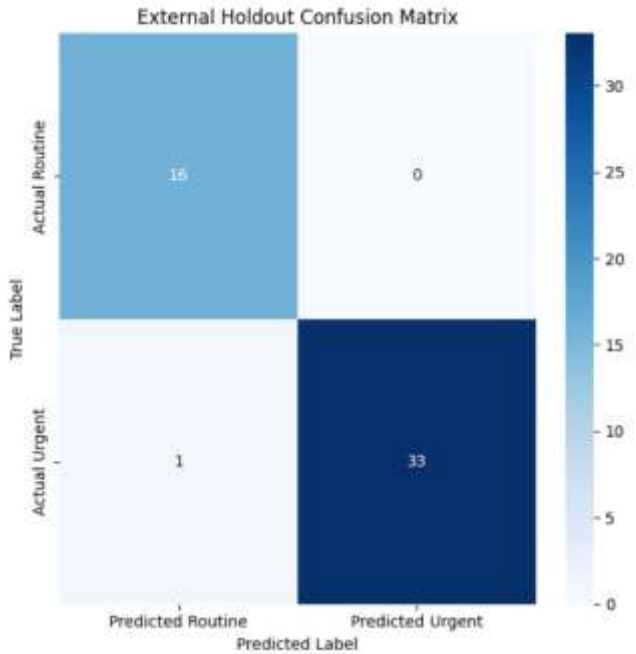
```
Evaluation on external holdout set (never seen during training)...

Holdout Classification Report:
              precision    recall  f1-score   support

     routine     0.9412    1.0000    0.9697        16
      urgent     1.0000    0.9706    0.9851        34

    accuracy                         0.9800        50
   macro avg     0.9706    0.9853    0.9774        50
weighted avg     0.9812    0.9800    0.9802        50


Holdout Confusion Matrix:
                 Predicted Routine  Predicted Urgent
Actual Routine                  16                 0
Actual Urgent                    1                33
```



5a) Conclusion

- Multiple classification models—**Logistic Regression**, **Random Forest**, and **XGBoost**—were evaluated using different feature extraction techniques: **TF-IDF** and **Transformer embeddings**.
- Class imbalance was addressed using **SMOTE**, which improved model performance, especially for the minority "Urgent" class.
- **Tree-based ensemble models** (Random Forest and XGBoost) combined with SBERT embeddings and SMOTE achieved the **best results**, with accuracy around 62% and balanced precision, recall, and F1-scores across classes.
- **Logistic Regression** with **TF-IDF** features served as a **strong baseline**, achieving moderate performance (around 59% accuracy) with good interpretability.
- Models using Transformer embeddings without SMOTE generally underperformed compared to TF-IDF and SBERT with SMOTE, highlighting the importance of both **embedding choice** and **data balancing**.
- Overall, **combining** semantic text embeddings (like **SBERT**) with **ensemble classifiers** and **oversampling** techniques like SMOTE is **effective** for imbalanced text classification tasks.
- Future improvements could include further hyperparameter tuning, exploring more advanced embeddings, or hybrid model architectures to boost predictive accuracy.

5b) Conclusion

- The **DistilBERT**-based classification model demonstrated excellent performance in distinguishing between "routine" and "urgent" text samples. After fine-tuning for just two epochs, the model achieved **98% accuracy** on both the internal validation and true external holdout sets.
- The success underscores the effectiveness of using pretrained Transformer models with minimal tuning for binary text classification, especially when supported by proper data preprocessing, balanced splits, and robust evaluation on an unseen holdout set.
- Future improvements could explore longer training schedules, hyperparameter tuning, and evaluation on larger, more diverse datasets to ensure scalability and robustness in real-world settings.

Top Challenges & Limitations

- **Synthetic Data Validity:** While synthetic data allowed safe experimentation, it may not capture the full complexity and subtlety of real referral language and clinical context, possibly limiting generalizability.
- **Class Imbalance:** Urgent cases are naturally less frequent, requiring SMOTE oversampling to train robust models and prevent bias toward non-urgent predictions.
- **Limited Clinical NLP:** Current approach does not deeply leverage medical ontologies or terminology normalization, which could improve semantic understanding.
- **Memory Constraints**: Training large transformer models (e.g., BERT) with long input sequences (max length 512) and full tokenization of all referral PDFs led to RAM/session memory issues.
- **Urgency Label Noise**: Using keyword heuristics to assign urgent vs routine labels introduces potential noise or mislabeling in borderline cases.

Reflection on Synthetic Data Utility & Risks

- Synthetic data enables rapid prototyping without privacy concerns, but models trained on such data may miss rare real-world patterns or emergent clinical terms.

- There's risk of bias if synthetic data distributions don't fully reflect true referral populations, leading to potential under- or over-classification of urgency.
- Real clinical text often contains ambiguous phrasing, abbreviations, and misspellings, which the synthetic dataset may not fully simulate.

Potential Applications & Next Steps

- Training on real referral data, if available, with robust de-identification and ethics compliance
- Incorporating clinical NLP tools (e.g - UMLS, MedCAT) for better terminology understanding
- Expanding model architectures, including transformer fine-tuning and hybrid approaches
- Implementing human-in-the-loop verification to reduce misclassification risks
- Monitoring model drift and updating with new data continuously