# Context Aware Email Reply Generator

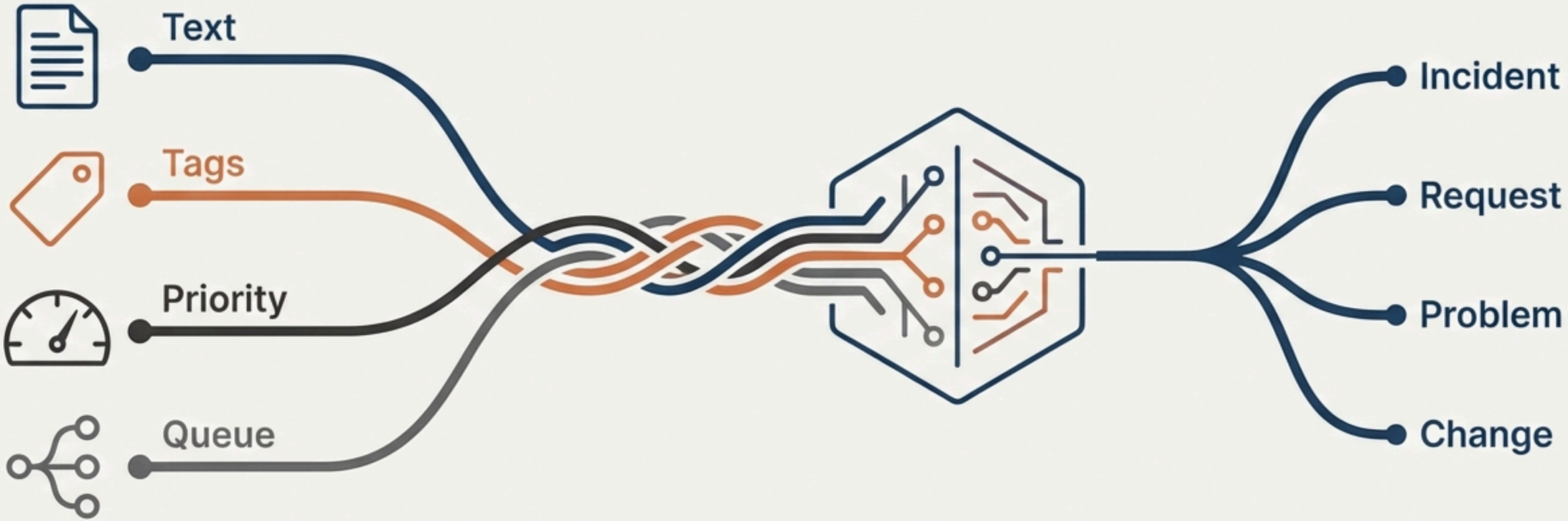
## Group - 8

# GROUP MEMBERS

TARANPREET
MANDAR
PRASHANT
SAHIL
SUMIT
C.SAI ANAND

# Building a Smarter Support System: From Raw Text to Contextual Understanding

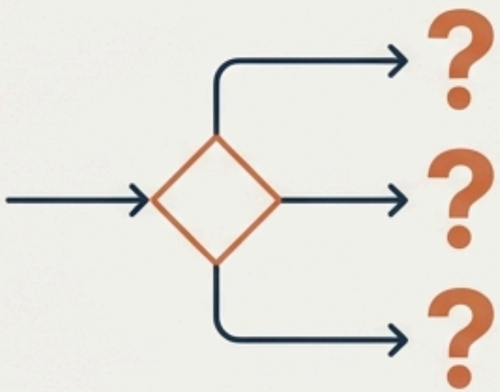
A technical walkthrough of a multi-input deep learning model for automated support ticket classification.



# The Challenge: Manual Ticket Sorting is a Bottleneck to Efficient Support

Every day, support teams face a high volume of incoming tickets. Manually reading, categorizing, and routing each one is slow, prone to human error, and delays resolution for customers. Inconsistent categorization leads to poor data quality, making it difficult to identify trends and systemic issues. Our goal was to build an automated system to classify incoming support tickets accurately and instantly, freeing up agent time for high-value problem-solving.

Time Consuming



Error-Prone

Poor Data Quality

# The First Step: Can We Classify Tickets Based on Text Alone?

The most prominent feature of any ticket is its text content. Our initial hypothesis was to leverage the `subject` and `body` fields to predict the ticket `type`.

We began by consolidating the `subject` and `body` into a single text field and applying standard text cleaning procedures.

```
# Consolidate and clean text fields
df["text"] = (
    df["subject"].fillna("") + " " +
    df["body"].fillna("")
).apply(clean_text)

# clean_text function:
# 1. Convert to lowercase
# 2. Remove URLs
# 3. Remove non-alphabetic characters
# 4. Standardize whitespace
```

> So What? This preprocessing is essential to create a standardized, noise-free input for a language model.

# Unlocking Context by Integrating Rich Metadata

We identified three key metadata fields that provide crucial contextual clues for classification:

## Queue

- Description: The support team the ticket is assigned to (e.g., 'Technical Support').
- Signal: Indicates the domain of the problem.
- Encoding: `LabelEncoder` (Assigns a unique integer to each queue name).

## Priority

- Description: The urgency of the ticket ('Low', 'Medium', 'High').
- Signal: Directly correlates with certain ticket types like 'Incident'.
- Encoding: Custom Ordinal Mapping (`{"Low": 0, "Medium": 1, "High": 2}`").

## Tags

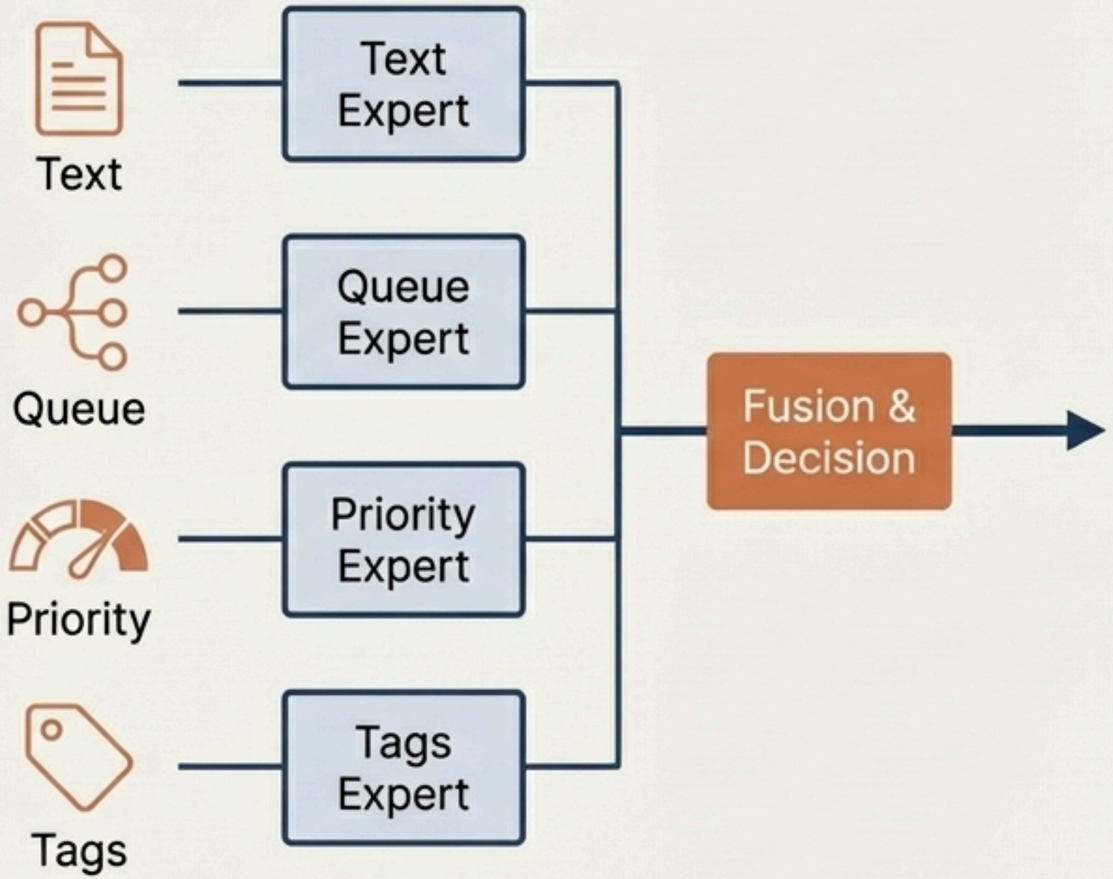
- Description: User- or system-applied labels (e.g., 'Outage', 'Account Disruption').
- Signal: Provides specific, granular details about the issue.
- Encoding: `MultiLabelBinarizer` (Creates a binary vector, or 'multi-hot' encoding, for all possible tags).

# The Solution: A Fusion Model that Reads Text and Understands Context

Instead of a single model for text, we designed a multi-input neural network with specialized ‘branches’ to process each data type independently.

Each branch acts as an expert, extracting patterns from its specific input (text, queue, priority, or tags).

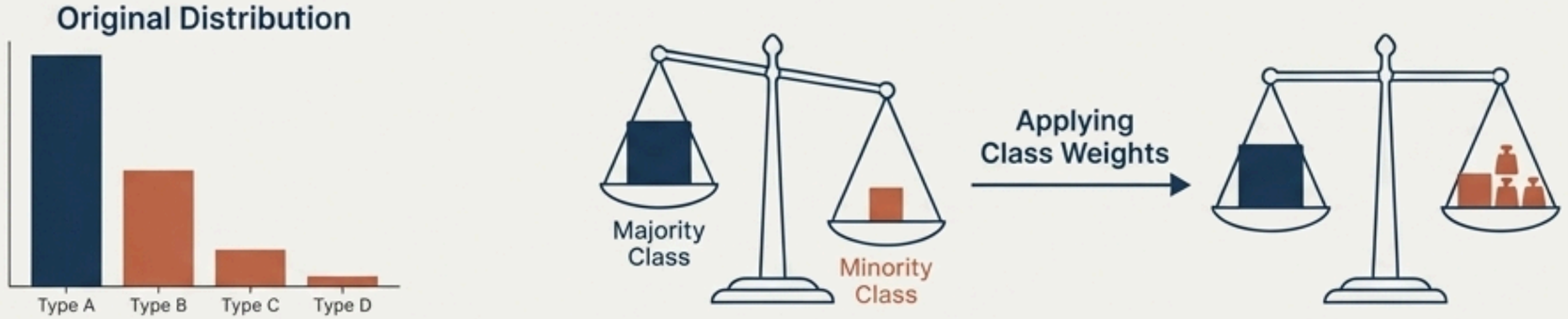
The insights from all branches are then ‘fused’ together in a final step, allowing the model to make a holistic and context-aware classification.



# Training a Fair Model: Addressing Class Imbalance with Weighted Loss

The training data is not perfectly balanced; some ticket types appear far more frequently than others. Without intervention, the model would become biased towards the majority classes and perform poorly on rare but critical ticket types.

To counteract this, we applied class weights during training. This technique effectively penalizes the model more for misclassifying examples from underrepresented classes.




**By using balanced class weights, we ensure the model learns to identify all ticket types effectively, not just the most common ones.**

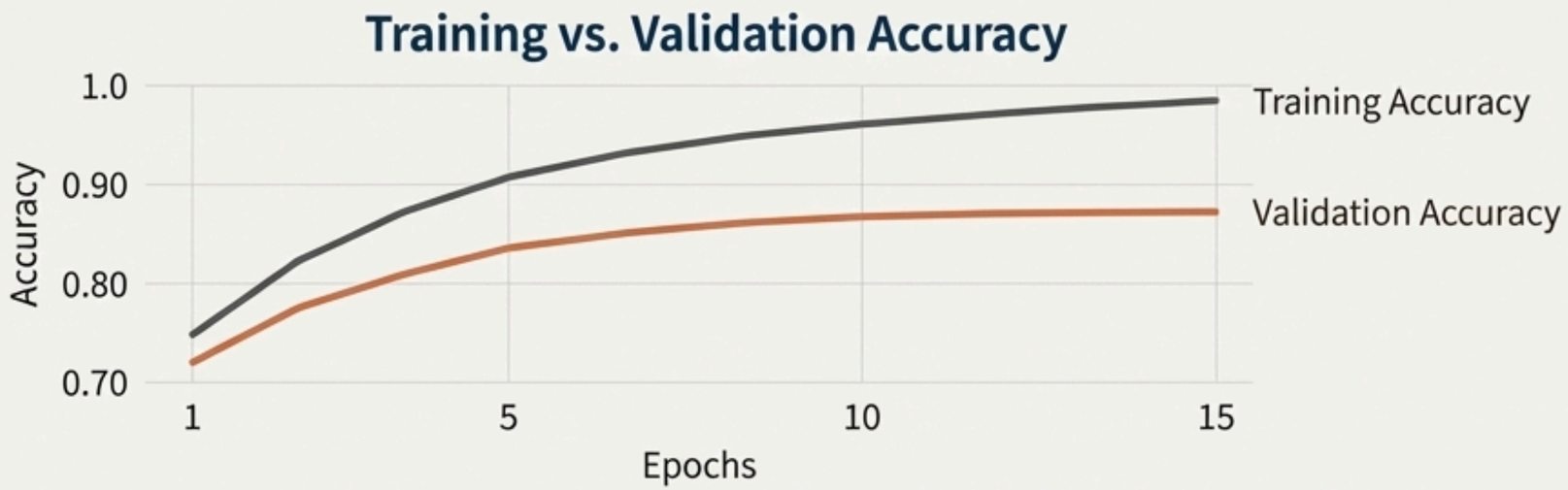
```
# Calculate weights to penalize errors on minority classes more heavily
class_weights = compute_class_weight(
    class_weight="balanced",
    classes=np.unique(y_tr),
    y=y_tr
)
```

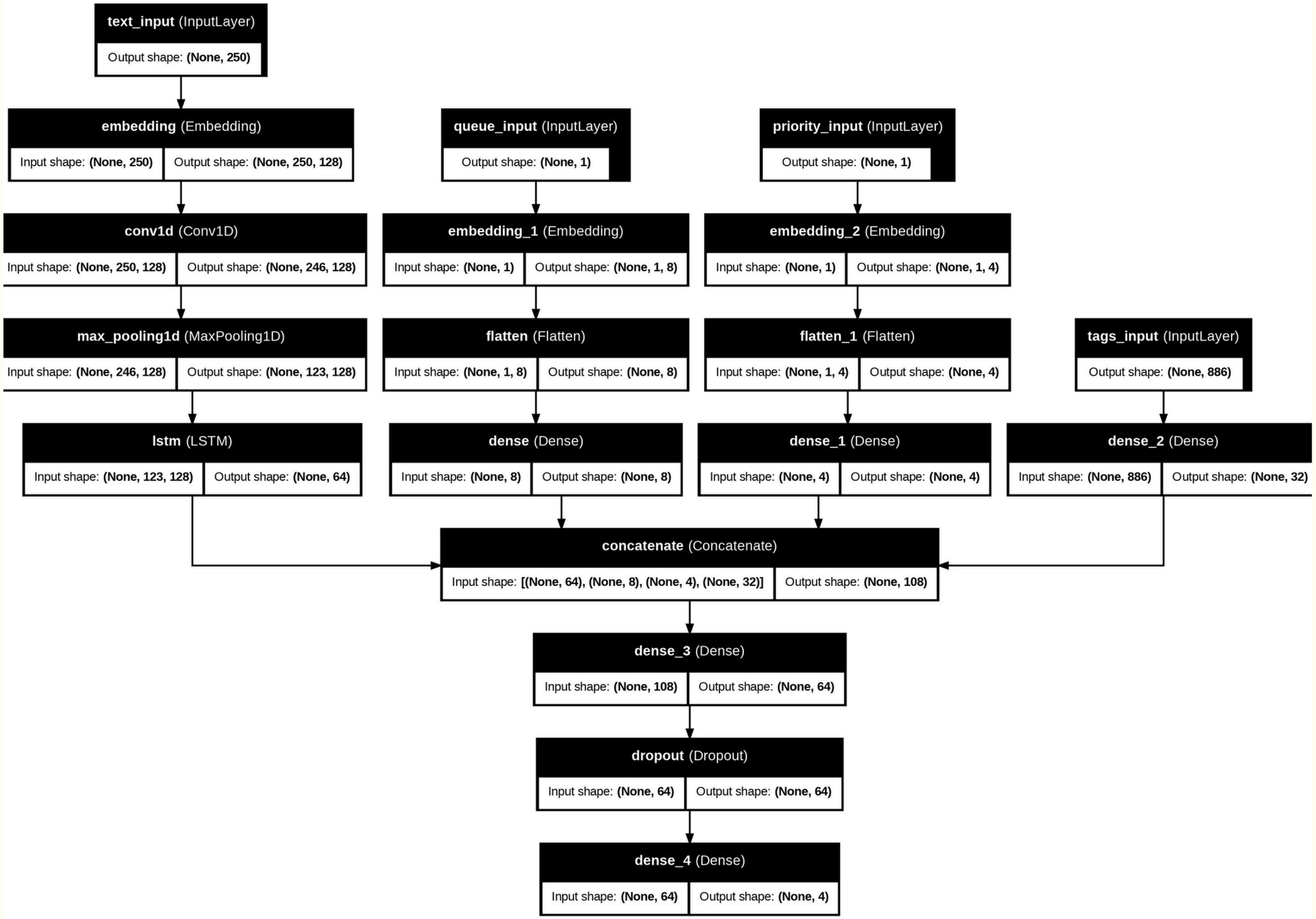
# The Verdict: An 87% Accurate Classification Engine

After training for 15 epochs, the model was evaluated on a held-out test set it had never seen before. The model successfully automates ticket classification with high accuracy.

# 87%

`weighted avg accuracy` on 3,268 test samples.




text_input (InputLayer)
Output shape: (None, 250)
embedding (Embedding)
Input shape: (None, 250)
Output shape: (None, 250, 128)
queue_input (InputLayer)
Output shape: (None, 1)
priority_input (InputLayer)
Output shape: (None, 1)
conv1d (Conv1D)
Input shape: (None, 250, 128)
Output shape: (None, 246, 128)
embedding_1 (Embedding)
Input shape: (None, 1)
Output shape: (None, 1, 8)
embedding_2 (Embedding)
Input shape: (None, 1)
Output shape: (None, 1, 4)
max_pooling1d (MaxPooling1D)
Input shape: (None, 246, 128)
Output shape: (None, 123, 128)
flatten (Flatten)
Input shape: (None, 1, 8)
Output shape: (None, 8)
flatten_1 (Flatten)
Input shape: (None, 1, 4)
Output shape: (None, 4)
tags_input (InputLayer)
Output shape: (None, 886)
lstm (LSTM)
Input shape: (None, 123, 128)
Output shape: (None, 64)
dense (Dense)
Input shape: (None, 8)
Output shape: (None, 8)
dense_1 (Dense)
Input shape: (None, 4)
Output shape: (None, 4)
dense_2 (Dense)
Input shape: (None, 886)
Output shape: (None, 32)
concatenate (Concatenate)
Input shape: [(None, 64), (None, 8), (None, 4), (None, 32)]
Output shape: (None, 108)
dense_3 (Dense)
Input shape: (None, 108)
Output shape: (None, 64)
dropout (Dropout)
Input shape: (None, 64)
Output shape: (None, 64)
dense_4 (Dense)
Input shape: (None, 64)
Output shape: (None, 4)

# Performance Under the Microscope: A Class-by-Class Breakdown

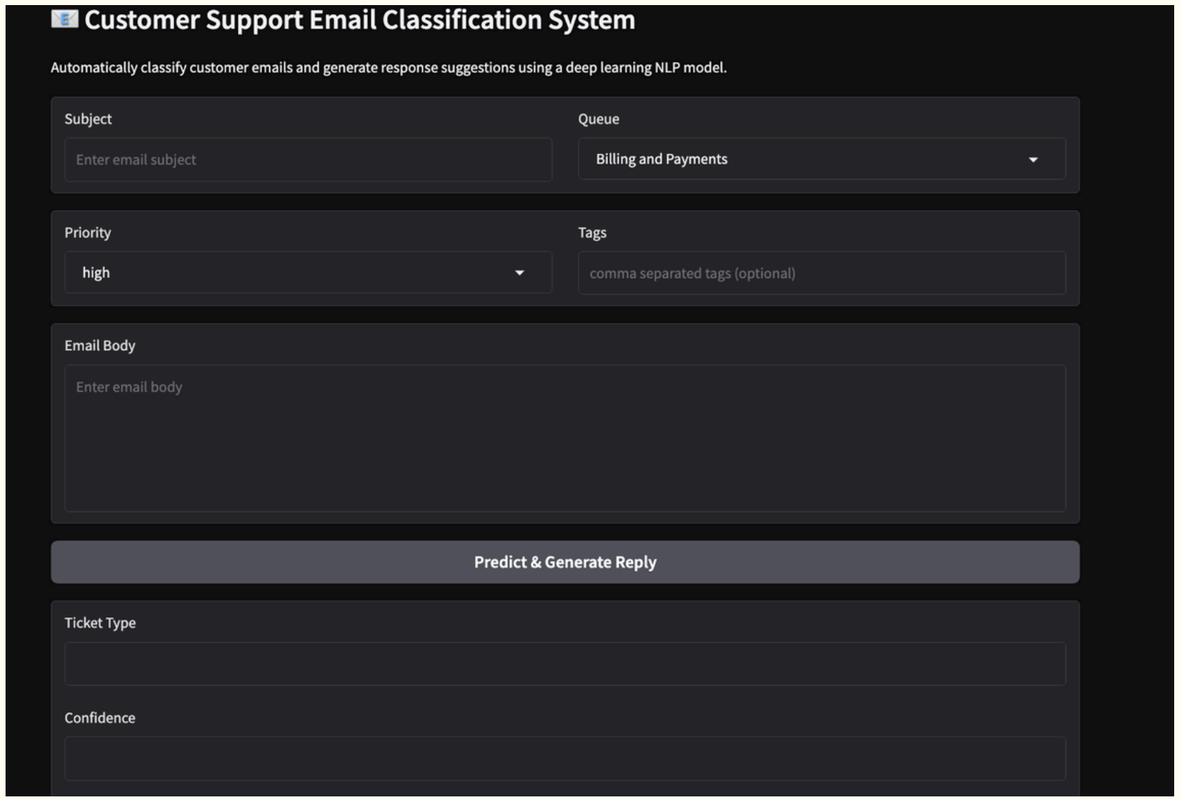
The overall accuracy is strong, but a detailed report reveals where the model truly excels and where it has room for improvement.

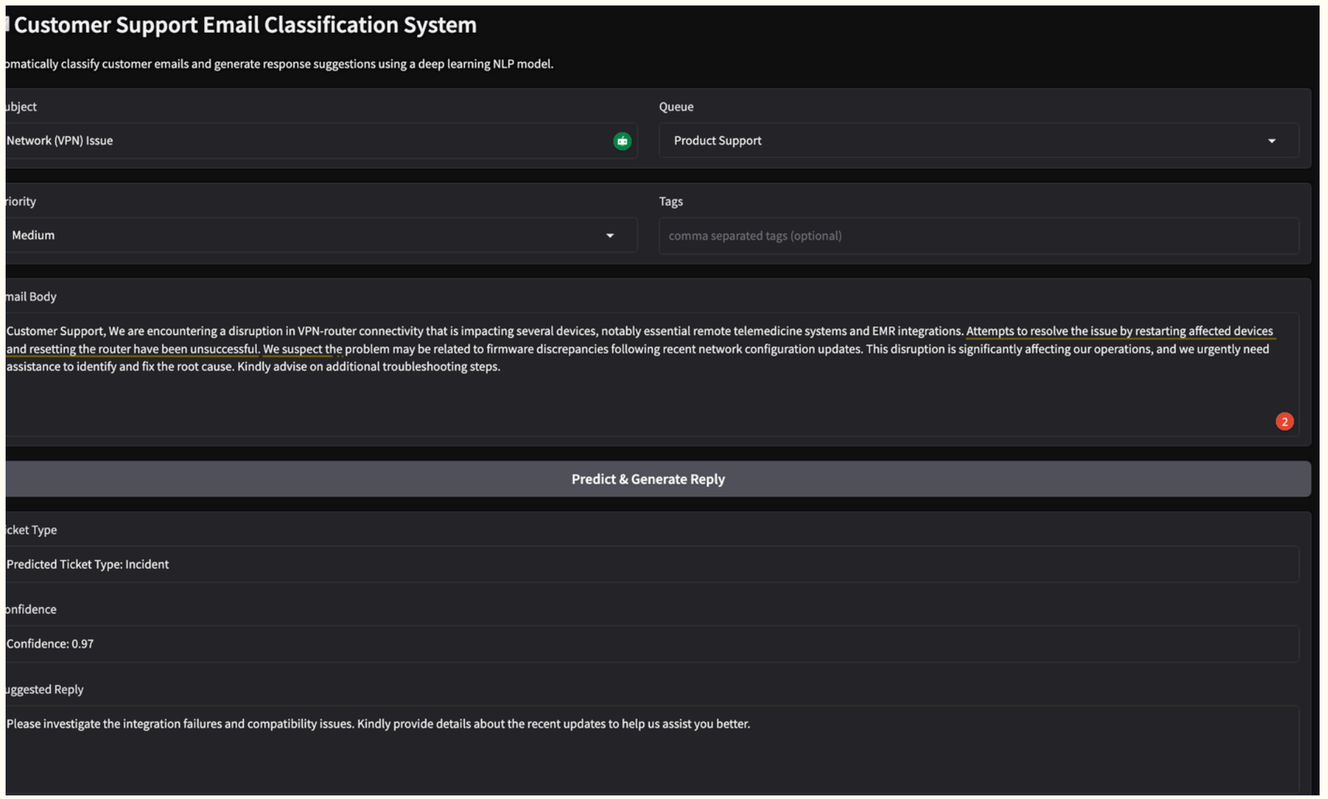
| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Change | 0.99 | 0.97 | 0.98 | 341 |
| Incident | 0.84 | 0.84 | 0.84 | 1314 |
| Problem | 0.70 | 0.69 | 0.69 | 680 |
| Request | 0.99 | 1.00 | 0.99 | 933 |
| accuracy | | | 0.87 | 3268 |
| macro avg | 0.88 | 0.88 | 0.88 | 3268 |
| weighted avg | 0.87 | 0.87 | 0.87 | 3268 |

**Exceptional Performance:** The model classifies 'Request' and 'Change' tickets with near-perfect precision and recall. These are likely well-defined categories with distinct language and metadata.

**Area for Improvement:** The model's performance on 'Problem' tickets is lower (70% precision). This suggests a significant overlap in language and context with other categories, particularly 'Incident'.

# USER INTERFACE



# FUTURE SCOPE

- **More accurate replies:** The system can be improved to give replies that better match the exact problem.
- **Faster handling:** It can learn which issues are more serious and need quick attention.
- **Use in more places:** It can be trained to understand emails in different languages and improve over time.


Thank
You