# **Sensitivity Analysis and Testing**

## **1. Objective**

The objective of this sensitivity analysis is to evaluate the robustness of the NLP classification model against human-like typographical errors in transaction descriptions. Given that real-world data may contain typos due to manual entry, it is essential to assess how well the model performs when exposed to such perturbations. This analysis ensures that the model remains reliable in practical scenarios where minor textual variations occur.

## **2. Overview of Methodology**

To conduct this sensitivity analysis, we introduced controlled levels of typographical errors into the input text column of the dataset. Specifically, random character modifications such as swaps, deletions, insertions, and replacements were applied at predefined typo rates: **5%, 10%, and 13%**. These rates were chosen based on:

* Studies indicating that typical human typists exhibit a typo rate between **1% and 5%**.
* The nature of our dataset, which consists of wire transaction descriptions **manually entered by bank personnel**, rather than casual or mobile device inputs.
* The structured nature of long transaction descriptions, which often follow a **template format**, inherently reduces the likelihood of frequent typographical errors, whereas shorter descriptions are manually created and may be more prone to typos.

Given these factors, **5% was selected as a conservative baseline**, **10% as a moderate stress test**, and **13% as a high-stress scenario** to observe performance degradation beyond normal human error levels.

### **Keyboard Adjacency Mapping and Typo Injection**

A **keyboard adjacency mapping** was created to simulate realistic typographical errors based on common QWERTY keyboard mistakes. This mapping associates each letter with its adjacent keys, simulating likely mispresses. Below is an example of the adjacency mapping used:

| **Letter** | **Adjacent Keys** |
| --- | --- |
| q | wa |
| w | qse |
| e | wsdr |
| r | etdf |
| t | ryfg |
| y | tugh |
| u | yihj |
| i | uojk |
| o | ipkl |
| p | ol |
| a | qwsz |
| s | awedxz |
| d | serfcx |
| f | drtgvc |
| g | ftyhbv |
| h | gyujnb |
| j | huikm |
| k | jiolm |
| l | kop |
| z | asx |
| x | zsdc |
| c | xdfv |
| v | cfgb |
| b | vghn |
| n | bhjm |
| m | njk |

We applied four types of typo modifications:

1. **Swap:** Two adjacent characters are swapped (e.g., fraud → fardu).
2. **Replace:** A character is replaced with a randomly selected adjacent key (e.g., wire → wjre).
3. **Delete:** A character is randomly deleted (e.g., money → moey).
4. **Insert:** A random adjacent character is inserted next to an existing character (e.g., bank → baank).

### **Application of Typo Rate**

The typo rate was applied by selecting a percentage of characters within each transaction description for modification. The process involved:

* Calculating the number of characters to be altered based on the total length of the string and the specified typo rate.
* Rounding up to ensure at least one modification is applied even for short words.
* Randomly choosing positions within the string to apply modifications.
* Ensuring that numeric values and specific words such as 'miss' remained unaltered.
* Applying a mix of swap, replace, delete, and insert operations proportionally across the selected characters.

For example, given the input transaction complete, applying a **10% typo rate** (rounding up to affect 2 characters) could yield:

* Swap (affects 2 characters at once): trnasaction complete (swapping 'r' and 'n')
* Replace (affects 2 characters through separate replacements): tarnsactuon complete (replacing 'r' with 'a' and 'i' with 'u')
* Delete (affects 2 characters by removing them): trasactin complete (removing 'n' and 'o')
* Insert (affects 2 characters by adding new ones): trannasaction complete (inserting an extra 'n' and 'a')

## **3. Sensitivity Analysis and Adversarial Testing**

The experiment was conducted by evaluating the model’s accuracy across different levels of induced typographical noise. The results were as follows:

| **Typo Rate** | **Model Accuracy** |
| --- | --- |
| **0% (Original Data)** | **79.00%** |
| **5% Typos** | **79.34%** (+0.34%) |
| **10% Typos** | **78.21%** (-0.79%) |
| **13% Typos** | **71.99%** (-7.01%) |

### **Key Observations:**

* **5% Typo Rate:** Minimal impact on model accuracy (+0.34%), suggesting the model is robust to minor typographical variations.
* **10% Typo Rate:** Slight decrease in accuracy (-0.79%), indicating some sensitivity to more frequent typos.
* **13% Typo Rate:** Significant accuracy drop (-7.01%), highlighting that extensive noise can degrade model performance.

These findings suggest that while the model remains stable under realistic typo scenarios (≤10%), **extensive typographical errors can meaningfully impact classification accuracy**.

## **4. Conclusion**

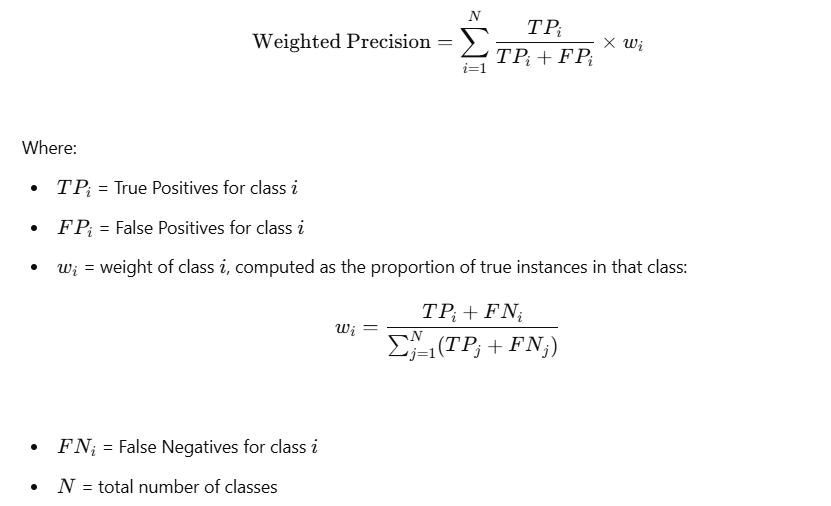
The sensitivity analysis demonstrates that the model is **robust to minor typographical errors**, especially at a realistic 5% typo rate, where no meaningful accuracy loss was observed. However, as the typo rate increases beyond 10%, performance degradation becomes noticeable, with a sharp decline at 13%.

These results validate that the model is well-suited for real-world applications where transaction descriptions are manually entered by bank personnel, as long as typo rates remain within reasonable human error thresholds. Future improvements could explore additional robustness techniques, such as **character-level embeddings** or **spell correction mechanisms**, to further mitigate classification errors under high-stress conditions.

### **Model Evaluation Metrics (PySpark MulticlassMetrics)**

#### **1. Weighted Precision**

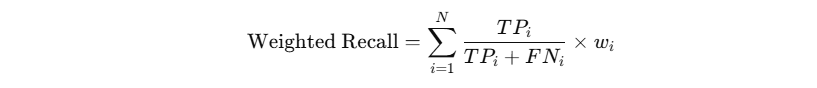
**Precision** measures the proportion of correctly classified positive instances among all instances predicted as positive for a given class. However, when dealing with **multiclass classification**, we compute the **weighted precision**, which is the sum of the **precision of each class weighted by the number of true instances in that class**.



**Interpretation**: A high weighted precision indicates that, on average, when the model predicts a class, it is correct more often than not.

### **2. Weighted Recall**

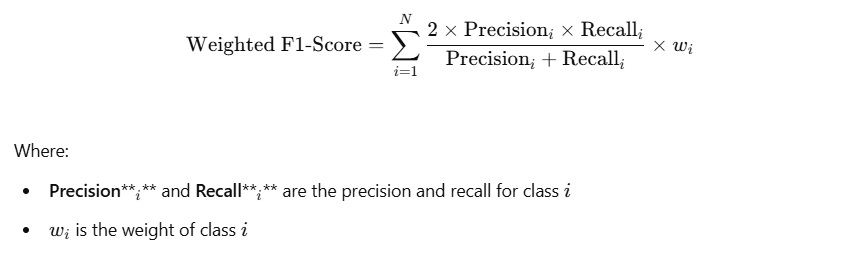
Recall (also called **sensitivity**) measures the proportion of correctly classified instances for a given class among all actual instances of that class. The **weighted recall** is calculated similarly, by taking the recall for each class and weighting it by the actual proportion of instances in that class.



**Interpretation**: A high weighted recall means that the model correctly identifies a high percentage of actual instances across all classes.

### **3. Weighted F1-Score**

The **F1-score** is the harmonic mean of precision and recall. The **weighted F1-score** considers the F1-score of each class and weights it by the actual class distribution.



**Interpretation**: The weighted F1-score provides a balanced measure of precision and recall across all classes, taking class imbalance into account.

### **Why Use Weighted Metrics?**

Since real-world datasets often have **class imbalances**, using weighted metrics ensures that performance is **not dominated by majority classes**. Instead of computing a simple average, weighting by class frequencies ensures that each class contributes proportionally to the final score.