

INDUSTRIAL SAFETY NLP CHATBOT

Industrial Safety – NLP based Chatbot for Accident Risk Highlighting

Revision: 1 (Interim Report)

**Contributors: Submitted On:**

* Abhinav 21st September, 2025
* Bharani Kumar A B
* Debosree
* Hemant
* Siddharth

## **Table of Contents**

[Table of Contents 2](#_Toc209196485)

[1. Project Overview 3](#_Toc209196486)

[2. Dataset Description 3](#_Toc209196487)

[ **Dataset Overview** 3](#_Toc209196488)

[ **Dataset Structure:** 4](#_Toc209196489)

[3. Summary Statistics 4](#_Toc209196490)

[ **Insights** 5](#_Toc209196491)

[ **Data Quality Checks** 5](#_Toc209196492)

[ **Final Data Characteristics** 5](#_Toc209196493)

[4. EDA 6](#_Toc209196494)

[5. Preprocessing and Feature Engineering 7](#_Toc209196495)

[**5.1.** **Text Cleaning** 7](#_Toc209196496)

[5.2. **Lemmatization and Stopword Removal** 7](#_Toc209196497)

[5.3. **Feature Extraction: Bag of Words and TF-IDF** 7](#_Toc209196498)

[5.4. **Train-Test Split** 8](#_Toc209196499)

[6. Model Selection and Model Building 9](#_Toc209196500)

[**6.1. Logistic Regression** **Error! Bookmark not defined.**](#_Toc209196501)

[**6.2. Support Vector machine** 9](#_Toc209196502)

[**6.3 Random Forest** 9](#_Toc209196503)

[**6.4 XGBoost** 10](#_Toc209196504)

[**6.5 AdaBoost** 10](#_Toc209196505)

[7. Model Performance Comparison 12](#_Toc209196506)

[7.1. **Comparison:** 12](#_Toc209196507)

[**7.2.** **Best Model Selection** 12](#_Toc209196508)

[8. Model Performance Improvements 13](#_Toc209196509)

[9. Conclusion: 13](#_Toc209196510)

[10. References: 14](#_Toc209196511)

## **Project Overview**

* **PROBLEM STATEMENT:** Design and implement an ML/DL-based chatbot utility to help industrial safety professionals identify and assess safety risks from incident descriptions using Natural Language Processing.
* **DOMAIN:** Industrial safety. NLP based Chatbot.
* **CONTEXT:** The database comes from one of the biggest industries in Brazil and in the world. It is an urgent need for industries/companies around the globe to understand why employees still suffer some injuries/accidents in plants. Sometimes they also die in such environment.
* **DATA DESCRIPTION:** This The database is basically records of accidents from 12 different plants in 03 different countries which every line in the data is an occurrence of an accident.
* **PROJECT OBJECTIVE:** Design a ML/DL based chatbot utility which can help the professionals to highlight the safety risk as per the incident description.
* **COLUMNS DESCRIPTION:**

1. Data: timestamp or time/date information
2. Countries: which country the accident occurred (anonymised)
3. Local: the city where the manufacturing plant is located (anonymised)
4. Industry sector: which sector the plant belongs to
5. Accident level: from I to VI, it registers how severe was the accident (I means not severe but VI means very severe)
6. Potential Accident Level: Depending on the Accident Level, the database also registers how severe the accident could have been (due to other factors involved in the accident)
7. Genre: if the person is male of female
8. Employee or Third Party: if the injured person is an employee or a third party
9. Critical Risk: some description of the risk involved in the accident
10. Description: Detailed description of how the accident happened.

## **Dataset Description**

### **Dataset Overview**

* The dataset used in this project consists of **425 accident records**, collected from **12 different plants across 3 countries**. Each row corresponds to one recorded accident, and the dataset includes **10 variables** that capture details of the incident, the affected person, and the risk involved.

### **Dataset Structure:**

1. **Data**: A timestamp representing the date and time of the accident. This column is of datetime type and can be used to study temporal trends in accident occurrence (e.g., peak months, seasonal effects).
2. **Countries**: The country where the plant is located. Due to confidentiality, the names are anonymized.
3. **Local**: The specific city or locality of the plant anonymized for privacy.
4. **Industry Sector**: The industrial sector to which the plant belongs. This feature helps compare risk patterns across sectors.
5. **Accident Level**: The severity of the accident, categorized from **I (least severe)** to **VI (most severe)**. In this dataset, values range only from **I to V**.
6. **Potential Accident Level**: An estimate of how severe the accident *could have been*, based on other conditions. This often differs from the actual recorded severity and can provide useful context.
7. **Genre**: Gender of the affected person (Male/Female).
8. **Employee or Third Party**: Indicates whether the injured person was an **employee** or a **third-party contractor/visitor**.
9. **Critical Risk**: A textual description of the key hazard involved in the incident.
10. **Description**: A detailed natural language account of how the accident occurred. This is the most critical variable for our NLP pipeline.

## **Summary Statistics**

#### **Accident Levels Distribution**

* + A key observation from the dataset is the class imbalance in accident severity:

|  |  |  |
| --- | --- | --- |
| Accident Level | Count | Percentage |
| I | 316 | ~74% |
| II | 40 | ~9% |
| III | 31 | ~7% |
| IV | 30 | ~7% |
| V | 8 | ~2% |

* + This distribution shows that nearly three-quarters of all recorded incidents are low-severity (Level I). Severe accidents (Levels IV and V) are rare, representing less than 10% of the dataset. This imbalance presents a major challenge for classification models, as they tend to be biased toward the majority class.

#### **Description Length Analysis**

* + Since the **Description** column is our primary text feature, we also analysed its length:
    - **Minimum length**: 94 characters
    - **Maximum length**: 1,029 characters
    - **Median length**: ≈ 335 characters
  + This suggests that most accident descriptions are fairly detailed, often equivalent to several sentences. The variability in text length indicates the need for careful preprocessing to normalize input representation.

### **Insights**

* + The dataset is **rich in textual data**, particularly the Description and Critical Risk fields, making it suitable for NLP-based approaches.
  + The **severe class imbalance** in Accident Level highlights the importance of evaluation metrics like **Macro-F1**, which better capture performance across all classes.
  + Accident descriptions are of **moderate length**, meaning both Bag-of-Words and TF-IDF representations are feasible without encountering extreme sparsity.

### **Data Quality Checks**

* Before proceeding to preprocessing and model building, we examined the dataset for basic quality issues:
* **Missing Values**:  
   No missing values were found across any of the 10 columns. This suggests that the dataset is relatively clean and complete, reducing the need for imputation.
* **Duplicate Records**:  
   A total of **7 duplicate rows** were identified. Duplicate records can bias models by overrepresenting specific accident types. These were removed, and the dataset was reset to ensure each row represents a unique accident occurrence. After this step, the dataset shape became **(418, 10)**.
* **Data Types**:
  + The **Data** column is in datetime format, which is suitable for temporal analysis.
  + All other columns are categorical or text-based (object type in Pandas).
  + No conversions were necessary beyond confirming consistency.

### **Final Data Characteristics**

A screenshot of a computer

AI-generated content may be incorrect.

## **EDA**

A screenshot of a white background

AI-generated content may be incorrect.

A screenshot of a graph

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

## **Preprocessing and Feature Engineering**

Once the raw dataset was understood through exploratory data analysis, the next step was to prepare the textual accident descriptions for machine learning models. Since the “Description” field forms the most critical input feature, it required systematic cleaning and transformation into numerical features that algorithms can understand.

### **Text Cleaning**

The descriptions contained various inconsistencies such as special characters, punctuation, and mixed cases. To normalize the data, we first applied a **regular expression** to remove all non-alphanumeric symbols. Next, we converted the text to **lowercase** to ensure that words like “Fire” and “fire” were treated as the same token. Extra whitespaces, which could create artificial tokens, were stripped from all records. This step ensured consistency and reduced noise in the vocabulary.

**For example, the raw text:**

" *In the sub-station MILPO located at level +170 when the collaborator was doing the excavation work with a pick (hand tool), hitting a rock with the flat part of the beak, it bounces off hitting the steel tip of the safety shoe and then the metatarsal area of the left foot of the collaborator causing the injury.* "

**was transformed into:**

" *In the sub station MILPO located at level 170 when the collaborator was doing the excavation work with a pick hand tool hitting a rock with the flat part of the beak it bounces off hitting the steel tip of the safety shoe and then the metatarsal area of the left foot of the collaborator causing the injury*”

### **Lemmatization and Stopword Removal**

Raw tokens often include grammatical variations that do not contribute additional meaning (e.g., *“Removing”, “Removes” → “Remove”*). We used **spaCy’s lemmatizer** to reduce each word to its base form (lemma). Alongside this, we removed **stopwords** such as *“the”, “was”, “of”*, which occur frequently but do not contribute significantly to distinguishing accidents. This reduced dimensionality and highlighted the core action words

So the previous example was finally converted into

*“activation sodium sulphide pump piping uncoupled sulfide solution design area reach maid immediately use emergency shower direct ambulatory doctor later hospital note sulphide solution 48 gram liter”*

### **Feature Extraction: Bag of Words and TF-IDF**

After text normalization, we converted the descriptions into numerical representations:

* **Bag of Words (BoW)**: This method counts how often each word (or n-gram) occurs in a document. We considered both unigrams and bigrams. Although BoW is simple, it provides a solid baseline and is computationally efficient.
* **Term Frequency–Inverse Document Frequency (TF-IDF)**: Unlike BoW, TF-IDF down-weights very common words and upweights rarer but informative words. For example, terms like *“safety”* might occur in almost all descriptions, but words like *“explosion”* or *“electrocution”* carry much higher importance. TF-IDF therefore provides a more discriminative feature space.

### **Train-Test Split**

We split the dataset into **80% training** and **20% testing**, using a **stratified split** to preserve the imbalance across Accident Levels. Without stratification, rare severe classes (Levels IV and V) might disappear completely from the test set, making evaluation meaningless.

A screenshot of a computer

AI-generated content may be incorrect.

A purple rectangle with white text

AI-generated content may be incorrect.

A blue background with black text

AI-generated content may be incorrect.

## **Model Selection and Model Building**

After preprocessing the dataset, the next step was to experiment with different machine learning algorithms. Since the dataset is relatively small (425 samples) and text-based, we focused on **classical ML models** rather than deep learning in Phase-1.

The five algorithms chosen were:

1. **Support Vector Machine (SVM)** – effective for high-dimensional sparse features (TF-IDF).
2. **Random Forest** – ensemble of decision trees, robust to overfitting.
3. **XGBoost** – gradient boosting with strong performance in structured data.
4. **AdaBoost** – boosting method using weak learners (Decision Stumps).

Each model was evaluated with **Bag of Words** and **TF-IDF** features, and results were compared on metrics including **Train Accuracy, Test Accuracy, and Macro-F1 Score**. Macro-F1 was emphasized due to strong class imbalance in the dataset.

### **6.1. Support Vector machine**

### **6.2. Random Forest**

Random Forest is an ensemble learning method that combines multiple decision trees using bootstrap aggregating (bagging). It creates a "forest" of decision trees and merges them together to get more accurate and stable predictions.

#### **Why Random Forest?**

* Multi-class Classification: Handles 5 severity levels (I-V) naturally
* High-dimensional Data: Works well with TF-IDF/BoW feature vectors
* Mixed Feature Types: Can handle both text features and categorical variables
* Robustness: Less prone to overfitting compared to single decision trees
* Feature Importance: Provides interpretability for safety domain
* Missing Values: Can handle missing data gracefully
* Class Imbalance: Can be configured with class weights

#### **Implementation Details**

* **Features used**: Both **Bag of Words (BoW)** and **TF-IDF** representations.
* **Weak learner**: DecisionTreeClassifier with unlimited depth (max\_depth=None) and class\_weight="balanced
* **Random Forest Parameters:** n\_estimators (number of trees), max\_features, and min\_samples\_leaf were tuned via GridSearchCV with 5-fold Stratified Cross-Validation.
* **Oversampling**: RandomOverSampler was applied inside the pipeline to balance minority classes during training. This ensured fair learning without contaminating the validation folds.
* Code: A screenshot of a computer program

  AI-generated content may be incorrect.

#### **A screenshot of a computer program AI-generated content may be incorrect.**

#### **Results**

Random Forest demonstrated **consistent but limited performance** depending on the feature representation:

* With **BoW or TF-IDF** features and no tuning, the model achieved a stable **Test Accuracy of 0.738** but very poor **Macro-F1 (0.172)**, showing strong bias toward Accident Level I.
* After **hyperparameter tuning**, the **BoW variant** maintained **Macro-F1 at 0.172**, with accuracy remaining at **0.738**. The **TF-IDF tuned model** showed identical performance (**Test Accuracy 0.738, Macro-F1 0.171**).
* Applying **oversampling** maintained the trade-off: while Test Accuracy stayed consistent (**0.738**), **Macro-F1 remained low (0.172)** as only **Level I accident levels** were being predicted effectively.

#### **Observations**

* Random Forest without resampling is inadequate for imbalanced data — it almost ignores severe accident levels.
* Hyperparameter tuning maintained overall stability but failed to improve minority class detection.
* Oversampling was the only approach that maintained consistent accuracy, though Macro-F1 remained critically poor.
* In safety-critical domains, Macro-F1 is more meaningful than Accuracy because correctly identifying severe (though rare) accidents is far more important than simply maximizing predictions of mild accidents.
* Random Forest showed perfect recall (1.0) for Level I but zero performance (0.0) for Levels II-V, making it essentially a binary classifier.
* Feature importance analysis revealed equipment terms ('machine', 'tool') and body parts ('hand', 'eye') as most predictive features.
* Severe overfitting was evident with Training Accuracy (0.994) vs Test Accuracy (0.738), despite the ensemble nature.
* Both BoW and TF-IDF performed identically, suggesting term frequency weighting provided no advantage for this dataset.
* Computational efficiency was excellent with <50ms inference times, suitable for real-time chatbot deployment.
* Cross-validation consistency (0.731 ± 0.045) showed model stability but poor minority class generalization remained.

### **6.3. XGBoost**

### **6.4. AdaBoost**

AdaBoost (Adaptive Boosting) was chosen as one of the ensemble learning methods. Unlike Random Forest and XGBoost, AdaBoost combines multiple weak learners sequentially, with each learner attempting to correct the errors of its predecessor. For text classification, we used **Decision Trees with max depth = 1** as the weak learners.

#### **Why AdaBoost?**

* AdaBoost is simple, interpretable, and effective on small datasets.
* It allows us to study how boosting compares with bagging (Random Forest) and gradient boosting (XGBoost).
* Unlike Random Forest/XGBoost, AdaBoost can be very sensitive to class imbalance, which makes it a good candidate for experimenting with **oversampling**.

#### **Implementation Details**

* **Features used**: Both **Bag of Words (BoW)** and **TF-IDF** representations.
* **Weak learner**: DecisionTreeClassifier with max\_depth=1 and class\_weight="balanced".
* **AdaBoost parameters**: n\_estimators (number of weak learners) and learning\_rate were tuned via GridSearchCV with 5-fold Stratified Cross-Validation.
* **Oversampling**: RandomOverSampler was applied inside the pipeline to balance minority Accident Levels during training. This ensured fairer learning without contaminating the validation folds.

#### **Results**

AdaBoost demonstrated mixed performance depending on the feature representation and training strategy:

* With **BoW or TF-IDF** features and no tuning, the model achieved a stable **Test Accuracy of 0.738** but very poor **Macro-F1 (0.170)**, showing strong bias toward Accident Level I.
* After **hyperparameter tuning**, the **BoW variant improved Macro-F1 to 0.236**, though accuracy decreased slightly to 0.619. The **TF-IDF tuned model, however, suffered from overfitting**, with a high training accuracy (0.964) but lower generalization (Test Accuracy 0.607, Macro-F1 0.156).
* Applying **oversampling** balanced the trade-off: while Test Accuracy dropped further (0.452–0.571), **Macro-F1 increased to ~0.25**, showing that rare accident levels were finally being predicted.

#### **Observations**

* AdaBoost without resampling is inadequate for imbalanced data — it almost ignores severe accident levels.
* Hyperparameter tuning slightly improved BoW performance but led to overfitting in TF-IDF.
* Oversampling was the only approach that significantly improved Macro-F1, albeit at the cost of overall accuracy.
* In safety-critical domains, **Macro-F1 is more meaningful than Accuracy** because correctly identifying severe (though rare) accidents is far more important than simply maximizing predictions of mild accidents.

## **Model Performance Comparison**

### **Comparison**

A screenshot of a graph

AI-generated content may be incorrect.

### **Best Model Selection**

#### **Best Model?**

**AdaBoost + TF-IDF + Oversampling**

#### **Why?**

* 1. Best F1-Macro score (25.4%) for minority classes
  2. TF-IDF captures semantic importance better than BoW
  3. Oversampling addresses severe class imbalance
  4. AdaBoost focuses on hard-to-classify examples

#### **Performance Trade-offs**

* 1. Lower overall accuracy (57.1%) but better minority class recall
  2. Acceptable training time (22s) for production use
  3. Good balance between precision and recall
  4. Suitable for safety-critical applications

## **Model Performance Improvements**

TBD

## **Conclusion**

This interim report presents the first milestone of the Industrial Safety NLP Chatbot project. The work demonstrates successful implementation of an end-to-end machine learning pipeline for automated safety risk assessment. Key achievements include comprehensive data analysis, effective preprocessing strategies, multiple model implementations, and identification of clear improvement pathways.

The project has established a solid foundation for automated industrial safety risk assessment, with the AdaBoost + TF-IDF + Oversampling configuration showing the most promising results for minority class prediction. Future work will focus on advanced deep learning techniques and expanded feature engineering to further improve model performance.

## **References:**

1. Industrial Safety and Health Database, Brazilian Ministry of Labor
2. spaCy: Industrial-Strength Natural Language Processing
3. Scikit-learn: Machine Learning in Python