



# Software Requirements Specification

## Disaster Tweets Analyzer: Enhanced by NLP and LLMs

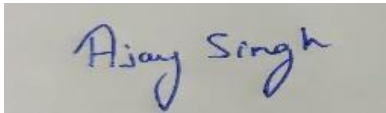


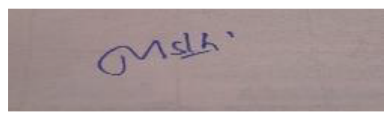

Submitted in partial fulfilment of the requirements for the Post Graduate Diploma in Big Data Analytics, C-DAC Bengaluru.

By

Ajay Singh – 240350125006  
Damini Choudhary -240350125021  
Dhananjay Singh – 240350125023  
Mansi – 240350125038  
Ritik Verma - 240350125059

# Acknowledgement

In today's competitive world, success is achieved through the collaboration and support of many. The completion of this project would not have been possible without the participation and assistance of numerous individuals and communities. First and foremost, we acknowledge the contributions of the Kaggle and Papers with Code community, whose work on disaster tweet classification inspired this project. Their innovative approaches and shared knowledge have been a cornerstone for our research. We also extend our deepest gratitude to our project guide, Mr. B Reddy Ravi Kumar, for his unwavering support, guidance, and understanding throughout the project. His inspiration and valuable instructions were instrumental in accomplishing this project within the prescribed time. We are immensely thankful to the developers of the open-source libraries used in this project, including pandas, scikit-learn, and NLP. Their invaluable tools have made our work not only possible but efficient and effective. Additionally, we would like to express our gratitude to our parents and friends for their kind cooperation and encouragement, which greatly helped us in completing this project. Their support was a constant source of motivation.

S.NO	Name	Roll No	Signature
1.	Ajay Singh	240350125006	
2.	Damini Choudhary	240350125021	
3.	Dhananjay Singh	240350125023	
4.	Mansi	240350125038	
5.	Ritik Verma	240350125059	

# Abstract

The objective of this project is to develop a Natural Language Processing (NLP) application that classifies tweets related to disasters. By processing tweets the system can potentially alert authorities and individuals about disaster events, enabling timely interventions and reducing harm. The project utilizes both traditional machine learning methods and advanced techniques involving Large Language Models (LLMs) for feature extraction and classification.

## Table of Contents

<b>Table Of Contents .....</b>	<b>iv</b>
<b>Revision History</b>	
<b>1. Introduction .....</b>	<b>5</b>
1.1 Purpose .....	5
1.2 Scope.....	5
1.3 Definition, Acronyms & Abbreviations .....	5
1.4 References .....	5
1.5 Project Objective .....	6
1.6 Evaluation Metrics.....	6
<b>2. Overall Description .....</b>	<b>7</b>
2.1 Product Perspective .....	7
2.2 Product Function.....	7
2.3 User Classes and Characteristics.....	7
2.4 Operating Environment.....	7
2.5 Design And Implementation Constraints.....	8
2.6 User Documentation.....	8
2.7 System Model Description .....	8
2.8 System Model Diagram.....	9
2.9 Components .....	10
<b>3. External Interface Requirements .....</b>	<b>11</b>
3.1 User Interfaces.....	11
3.2 Hardware Requirements.....	11
3.3 Software Requirements .....	11
3.4 Communication Interface.....	11
<b>4. System Features.....</b>	<b>12</b>
4.1 Data Loading.....	12
4.2 Data Pre-Processing.....	12
4.3 Feature Extraction .....	12
4.4 Model Training .....	12
4.5 Model Evaluation.....	12
<b>5. Non-functional Requirement .....</b>	<b>13</b>
5.1 Performance.....	13
5.2 Usability .....	13
5.3 Maintainability.....	13
5.4 Reliability .....	13
5.5 Availability .....	13
5.6 Security .....	13
<b>6. Acceptance Criteria.....</b>	<b>14</b>
<b>7. Deliverables .....</b>	<b>15</b>

# 1.0. Introduction

## 1.1. Purpose

The purpose of this document is to outline the requirements for developing a Natural Language Processing (NLP) application aimed at classifying tweets related to disasters. This document serves as a guide for developers, stakeholders, and end-users by detailing the application's functionalities, requirements, and design.

## 1.2. Scope

The scope of the NLP application includes data loading, preprocessing, feature extraction using advanced techniques (LLM embeddings), model training with various algorithms, and evaluation of the models' performance. The application will be deployed in a Jupyter Notebook environment using Python.

## 1.3. Definitions, Acronyms, and Abbreviations

- **NLP:** Natural Language Processing
- **ML:** Machine Learning
- **LLM:** Large Language Model
- **CSV:** Comma-Separated Values

## 1.4. References

- **Datasets:** Utilize Kaggle datasets like Twitter US Airline Sentiment Dataset for disaster-related tweets.
- **Frameworks:** Employ NLP tasks, and TensorFlow for deep learning implementations.
- **Models:** Focus on state-of-the-art models for tweet analysis and sentiment classification.
- **Applications:** Explore case studies demonstrating sentiment analysis and entity recognition in disaster tweets for practical insights.

## 1.5. Project Objective

The objective of the project is to predict whether a particular tweet, based on its text (and occasionally the keyword and location), indicates a real disaster or not. Twitter is a major social media platform where many share occurrences of incidents, including disasters. For example, during a fire, nearby individuals are likely to tweet about it, sending early alerts to people to evacuate and to authorities to act. Detecting disaster-indicating tweets can significantly enhance emergency disaster management.

## 1.6. Evaluation Metric

In this problem, the positive class (tweets indicating actual disasters) is more critical than the negative class (tweets not indicating any disaster). Therefore, the goal is to minimize false positives (FP) and false negatives (FN), balancing both precision and recall.

Definitions:

- **TP (True Positives):** Tweets correctly classified as disasters.
- **TN (True Negatives):** Tweets correctly classified as non-disasters.
- **FP (False Positives):** Tweets incorrectly classified as disasters.
- **FN (False Negatives):** Tweets incorrectly classified as non-disasters.

$$\text{Precision} = \frac{TP}{TP + FP}.$$

$$\text{Recall} = \frac{TP}{TP + FN}.$$

$$F_1\text{-Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}.$$

F1-score balances precision and recall, making it suitable for evaluating the models in this project.

## 2.0. Overall Description

### 2.1. Product Perspective

The application is a standalone Python-based solution deployed in a Jupyter Notebook. It integrates various NLP techniques and machine learning models, including LLMs, to classify tweets, making it a powerful tool for identifying disaster-related content in social media data.

### 2.2. Product Functions

- **Data Loading:** Import tweet data from CSV files using pandas.
- **Data Preprocessing:** Clean and preprocess text data by removing noise and tokenizing text.
- **Feature Extraction:** Using (BERT) LLMs.
- **Model Training:** Implement and train machine learning models, including Random Tree Forest, and LLM-based classifiers.
- **Model Evaluation:** Assess model performance using metrics like accuracy, precision, recall, and F1 score.

### 2.3. User Classes and Characteristics

- **Data Scientists:** Experts in data analysis and machine learning.
- **ML Engineers:** Professionals with experience in implementing and deploying machine learning models.
- **Researchers:** Individuals interested in disaster management and social media data analysis.

### 2.4. Operating Environment

The application runs in a Jupyter Notebook environment using Python. Required libraries include pandas, scikit-learn, NLP. Internet access is necessary for library installations and dataset download.

## **2.5. Design and Implementation Constraints**

- Internet access required for downloading datasets and libraries.
- Limited to the capabilities of the Jupyter Notebook environment.
- Dependent on hardware performance for processing and training.

## **2.6. User Documentation**

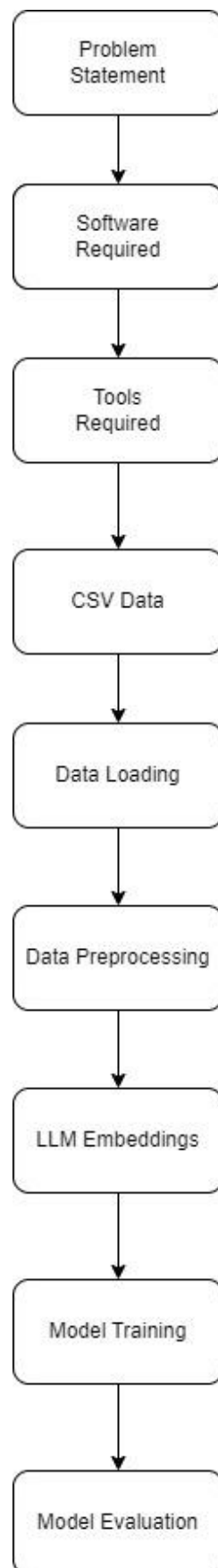
- Detailed explanations and code comments in the Jupyter Notebook.
- Step-by-step guidance for data loading, preprocessing, feature extraction, model training, and evaluation.

## **2.7. System Model Description**

The system model provides a high-level overview of the data flow and interactions between different components of the NLP application. The major components include data loading, preprocessing, feature extraction, model training, and evaluation. The system integrates traditional ML approaches with advanced LLM techniques like (BERT) to enhance classification accuracy and performance.



## 2.8. System Model Diagram



## 2.9. Components:

1. **Data Loading:** Loads the CSV file containing tweet data.
2. **Data Preprocessing:** Cleans and tokenizes the text data.
3. **LLM Embeddings:** Generates text embeddings using a pre-trained LLM (BERT).
4. **Model Training:** Trains machine learning models on the combined features.
5. **Model Evaluation:** Evaluates the trained models using various performance metrics.

## **3.0. External Interface Requirements**

### **3.1. User Interface**

- The application is presented in a Jupyter Notebook with markdown explanations and code cells.
- Visualizations for data analysis and model performance are included.

### **3.2. Hardware Interface**

- The application should run on a standard laptop with a minimum of 8GB RAM and a modern CPU.

### **3.3. Software Interface**

- The application requires Python and Jupyter Notebook.
- Necessary Python libraries include pandas, scikit-learn, nltk, and transformers.

### **3.4. Communication Interface**

- Internet access for downloading datasets and libraries.

# 4.0. System Features

## 4.1. Data Loading

- Load tweet data from a CSV file using pandas.
- Handle missing values and inconsistent data formats during loading.

## 4.2. Data Preprocessing

- Clean text data by removing punctuation, special characters, and stop words.
- Tokenize text data and convert it to lowercase.
- Handle missing values by removing or imputing them appropriately.

## 4.3. Feature Extraction

- Use embeddings from a pre-trained LLM to enhance feature extraction.

## 4.4. Model Training

- Implement Random Tree Forest, and LLMs (BERT)-based classifiers.
- Split the dataset into training and testing sets.

## 4.5. Model Evaluation

- Evaluate models using accuracy, precision, recall, and F1 score.
- Present evaluation metrics clearly.

# 5.0. Non-Functional Requirements

## 5.1. Performance

- Process and train on a dataset of 10,000 tweets within 5 minutes on a standard laptop.
- Efficiently handle large datasets without significant slowdowns.

## 5.2. Usability

- Provide clear documentation and inline comments for each step.
- Organize the Jupyter Notebook into well-defined sections.

## 5.3. Maintainability

- Modular code with functions and classes for easy updates.
- Follow standard coding practices and conventions.

## 5.4. Reliability

- Robust preprocessing to handle noisy and incomplete tweet data.
- Ensure data consistency during preprocessing.

## 5.5. Availability

- Accessible through the Jupyter Notebook environment.

## 5.6. Security

- Follow standard security practices for data input and output.

## 6.0. Acceptance Criteria

- The application correctly loads and preprocesses tweet data from CSV files.
- LLM embeddings like (BERT).
- It implements and trains machine learning models, achieving an acceptable level of accuracy, precision, recall, and F1 score.
- The Jupyter Notebook is well-documented and organized, with clear explanations and visualizations.

## 7.0. Deliverables

- Jupyter Notebook containing the complete implementation, including data loading, preprocessing, feature extraction, model training, and evaluation.
- Documentation within the notebook providing detailed explanations and code comments.
- Sample dataset for testing and validation.
- Evaluation metrics and visualizations of model performance.