**Code Documentation**

**---------------------------------------------------------------------------------**

**1. Code Structure**

The code is organized into several logical blocks that reflect the data processing pipeline:

* **Spark Session Initialization**: Starts the Spark cluster session for distributed processing.
* **Data Loading**: Reads disaster-related tweets from HDFS using Spark DataFrames.
* **Data Preprocessing**:
  + Cleaning the text data (removing punctuation, links, emojis).
  + Handling nulls and standardizing location values.
* **Feature Engineering**:
  + Extracts keywords.
  + Computes text lengths and unique value counts.
* **Classification**:
  + Uses machine learning (Random Forest, Naive Bayes) to classify tweets as disaster or non-disaster.
* **Spatial and Sentiment Analysis**:
  + Applies geocoding and clustering (DBSCAN).
  + Performs sentiment analysis on tweet texts.
* **Visualization**:
  + Word clouds, text statistics, and geo-location mappings.

**2. Modules Used**

| **Category** | **Libraries / Modules** |
| --- | --- |
| **Big Data** | pyspark.sql, SparkSession, HDFS |
| **Data Processing** | pandas, nltk, re, string, emoji |
| **Machine Learning** | sklearn.model\_selection, CountVectorizer, MultinomialNB, RandomForestClassifier, metrics |
| **NLP** | nltk.tokenize, stopwords, TextBlob |
| **Geo Processing** | geopy, shapely, geopandas, OpenCageGeocode |
| **Clustering** | DBSCAN, StandardScaler |
| **Visualization** | matplotlib.pyplot, seaborn, WordCloud, descartes |

**3. Execution Steps**

**Initial Setup**

Python:

from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("TweetAnalysis").getOrCreate()

**Data Loading**

Python:

df = spark.read.csv("hdfs://master-node:9000/user/dinesh/tweets\_data\_sample/bda\_train.csv",

header=True, inferSchema=True)

**Preprocess Text**

* Remove emojis, symbols, punctuation.
* Lowercase conversion.
* Stopword removal.
* Extract and normalize keyword and location data.

**Feature Extraction**

* Generate text features using CountVectorizer.
* Analyze text length, duplicate tweets, unique values.

**Classification**

* Split dataset into train/test.
* Train classifier and evaluate using accuracy, confusion matrix.

**Geo-Spatial Clustering**

* Use OpenCageGeocode to convert location strings into coordinates.
* Apply DBSCAN to cluster geographic tweet locations.

**Sentiment Analysis**

* Evaluate tweet sentiment polarity and classify based on thresholds.

**Model Training**

**Python:**

pipeline = Pipeline([

("tfidf", TfidfVectorizer()),

("classifier", RandomForestClassifier())

])

pipeline.fit(X\_train, y\_train)

**Visualization & Dashboard**

* Generate interactive maps with severity indicators
* Create statistical plots
* Build word clouds

**Dependencies & Environment Setup**

Required Libraries

* pyspark
* nltk
* emoji
* pandas
* numpy
* scikit-learn
* geopy
* opencage
* plotly
* folium
* wordcloud
* matplotlib
* seaborn

**Environment Setup**

1. Install Java 8+
2. Install Spark and set environment variables
3. Install Python dependencies:

pip install pyspark nltk emoji pandas numpy scikit-learn geopy opencage plotly folium wordcloud matplotlib seaborn

1. Download NLTK resources:

nltk.download('stopwords')

nltk.download('punkt')

nltk.download('wordnet')

**4.Key Features**

1. **Comprehensive Text Processing**
   * Handles emojis, URLs, special characters
   * Tokenization and stemming
   * Sentiment analysis
2. **Geospatial Analysis**
   * Location geocoding
   * Spatial clustering
   * Interactive mapping
3. **Advanced Analytics**
   * Topic modeling (LDA)
   * Anomaly detection
   * Severity classification
4. **Interactive Dashboard**
   * Filter by location, severity, keywords
   * Integrated visualizations
   * Real-time updates