A HIGH LEVEL DESIGN ON

# Predicting Drug Review Polarity Using ML&NLP Approch

Submitted in the partial fulfillment of requirements to

CS -454– Project Lab

By

**Batch No.07**

Onteru Sandeep Kumar(Y22CS138)

Nettem Vamsi Krishna(Y22CS134)

Nune Jaswanth(Y22CS136)


**R.V.R.&J.C.COLLEGE OF ENGINEERING(Autonomous)**

**(NAAC 'A+' Grade)**

**Approved by AICTE::Affiliated to Acharya Nagarjuna University**

**Chandramoulipuram :: Chowdavaram**

**Guntur–522019, AndhraPradesh, India.**

| **Dr.M.Srikanth** | **Dr.SJRK PadminivalliV** | **Dr.M.Sreelatha** |
|---|---|---|
| Project Guide | Project In-charge | Prof.&Head, CSE |

# Contents

# 1.Problem Statement

The increasing volume of textual drug reviews, especially those provided by medical professionals, contains critical insights into the performance, safety, and real-world effectiveness of medications. However, extracting meaningful information such as patient sentiment, adverse drug reactions (ADRs), and the associated medical conditions from this unstructured text remains a significant challenge. Manual analysis is not only inefficient but also prone to inconsistency and oversight. There is a pressing need for an intelligent system that can automatically process and classify these reviews to support data-driven healthcare decisions. This project addresses this gap by developing a machine learning-based model capable of classifying drug review sentiments, predicting the underlying medical conditions, and extracting potential side effects using Natural Language Processing (NLP) techniques. The system leverages the UCI ML Drug Review Dataset to train and evaluate various models, using techniques such as Bag of Words, TF-IDF, and advanced classifiers including Naive Bayes, Passive Aggressive, SVM, and Neural Networks. Additionally, deep learning models like BERT are explored to improve contextual understanding and enable the extraction of deeper clinical insights from the text data.

## 2.Functional Requirements

1. **User Input**: Enable users (e.g., healthcare analysts or medical staff) to input drug reviews or queries through a simple web-based interface ( Frontend UI is implemented).

2. **Data Processing**:Implementing robust preprocessing pipelines to clean and normalize textual input. This includes removing noise such as stopwords, punctuation, numbers, and special characters. Apply advanced NLP techniques like tokenization, stemming, and lemmatization to standardize the vocabulary and prepare the data for feature extraction.

3. **Feature Extraction**: Convert processed text into numerical form using feature engineering techniques such as **Bag of Words (BoW)** and **Term Frequency-Inverse Document Frequency (TF-IDF)**. These vectorized representations serve as inputs to the classification models, capturing term importance and contextual frequency.

4. **Sentiment and Condition Classification**:Utilize machine learning algorithms like **Naive Bayes**, **Support Vector Machines (SVM)**, **Passive Aggressive Classifier**, and **Neural Networks** to classify reviews into three sentiment categories:**positive (8–10 ratings)**, **neutral (5–7 ratings)**, and **negative (1–4 ratings)**. Additionally, predict the **underlying medical condition** based on contextual cues in the review. These models are trained and validated using labeled review data from the UCI ML Drug Review Dataset.

5. **Side Effect Extraction(ADRs):**Implement Named Entity Recognition (NER) and rule-based or model-based extraction techniques to identify **adverse drug reactions (ADRs)** mentioned in the review text. This helps in recognizing side effects and safety signals associated with the medication.

6. **Deep Learning Integration**:Integrate transformer-based models such as **BERT** or **BioBERT** for enhanced text understanding and improved prediction accuracy. These models can provide deeper insights by capturing context, clinical semantics, and complex linguistic structures.

7. **Result Display**:Present classification results—including sentiment, predicted condition, and extracted side effects—in a user-friendly output format.

# 3.Basic Architecture

## 3.1 Major Components

1. **Frontend:** A user-friendly interface developed using **React.js** (or any modern frontend framework) allows users to input single or multiple drug reviews via a form or file upload. It also displays the output, including predicted **sentiment categories**, **medical conditions**, and **side effects**. Visualizations like sentiment pie charts or word clouds of extracted ADRs can be integrated to enhance interpretability..

2. **Backend**:The backend is powered by **FastAPI**,ensuring a fast, asynchronous, and scalable API framework. It handles incoming review data, routes it through preprocessing pipelines, and interfaces with trained machine learning models for prediction. It exposes RESTful endpoints for tasks such as sentiment classification, condition prediction, and side effect extraction.

3. **Data Processing Module**:This module leverages **NLTK**, **SpaCy**, and custom scripts to clean the raw input data. Core preprocessing steps include:

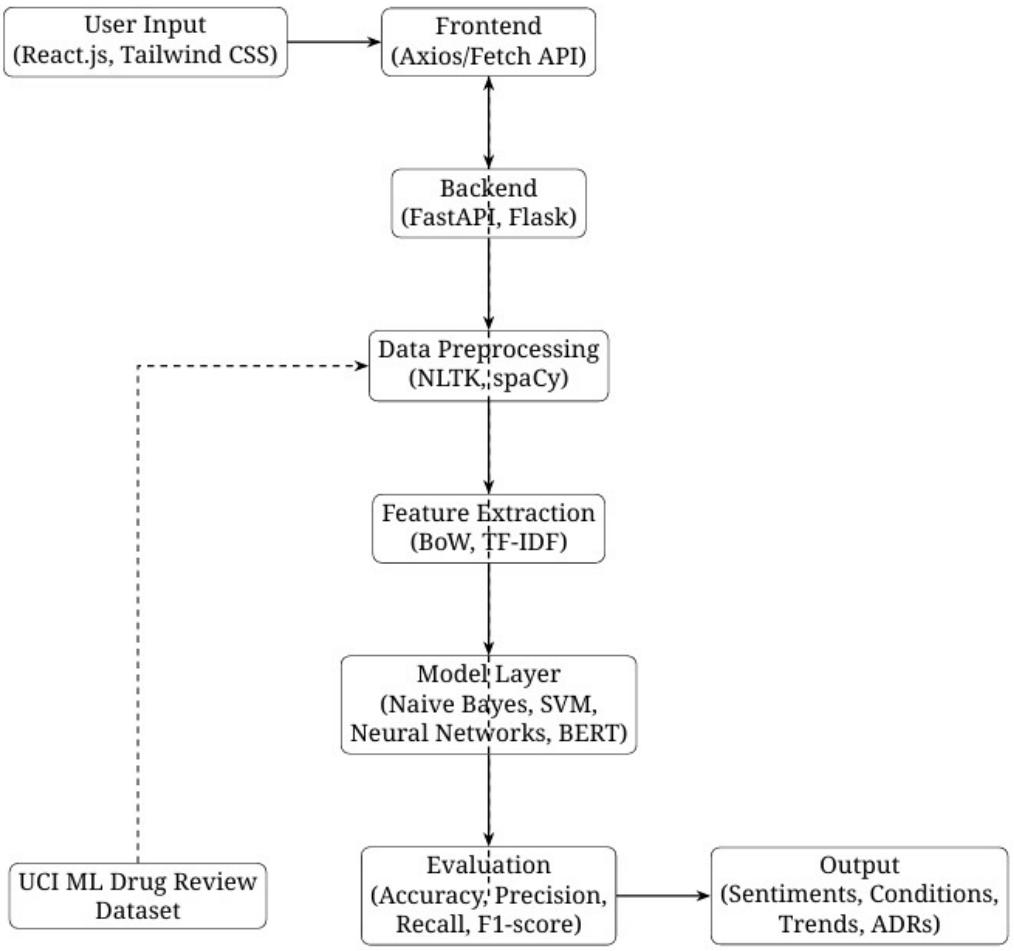
    - Removing stopwords, special characters, and punctuation
    - Tokenization
    - Lowecasing Text
    - Lemmatization or stemming
      After preprocessing, the text is transformed into numerical vectors using **BoW** or **TF-IDF** for use in classical ML models.

**4. Model Layer:**

- **Naive Bayes**: Lightweight and effective for text classification tasks.
- **Passive Aggressive Classifier**: Suitable for large-scale and online learning.
- **Support Vector Machine (SVM)**: High-accuracy classifier for separating sentiments.
- **BERT / BioBERT**: Transformer-based models fine-tuned on drug-related corpora for deeper contextual understanding and more accurate predictions.
  These models are responsible for both sentiment classification and condition prediction.

## 3.2 Diagram

The following is a textual representation of the Enhanced System Architecture Diagram, describing the flow from user input to result visualization.For rendering, use a Mermaid-compatible viewer




figureEnhanced System Architecture Diagram

## 3.3 Workflow

1. **Data Collection:**
   Begin by collecting comprehensive drug review data from the UCI ML Drug Review Dataset. This includes structured fields such as drug names, associated medical conditions, patient ratings, and textual reviews. Though the dataset is originally composed of patient inputs, the system is adapted for use with doctor-generated reviews to maintain clinical relevance.

2. **Text Preprocessing**:Apply NLP-based cleaning steps to prepare the raw textual data for analysis. This includes:

   - **Tokenization**: Splitting text into individual words or tokens
   - **Stemming**: Reducing words to their root form (e.g., "running" → "run")
   - **Lemmatization**: Converting words to their base dictionary form (e.g., "was" → "be")
   - Stopword removal (e.g., "is", "the", "and") to reduce noise

3. **Feature Extraction**: Convert the cleaned text into numerical feature vectors using:

   - **Bag of Words (BoW)**: A simple yet effective representation based on word counts
   - **TF-IDF (Term Frequency-Inverse Document Frequency)**: Captures word importance by scaling frequent terms down and rare but informative terms up
     These features are used as inputs for classical machine learning models.

4. **Model Training**:Split the dataset into training, validation, and testing subsets using a 70/15/15 split. Train multiple models (e.g., **Naive Bayes**, **SVM**, **Passive Aggressive**, **Neural Networks**) on the training set. Use the validation set to tune hyperparameters and prevent overfitting, and evaluate final model performance on the test set.

5. **Classification**:Use the trained models to classify:

   - **Sentiment**: Categorizing reviews into **positive (8–10)**, **neutral (5–7)**, and **negative (1–4)**
   - **Condition Prediction**: Predict the medical condition being treated based on text and review metadata
     Advanced models like **BERT** are optionally used to enhance contextual predictions.

6. **Result Analysis**:Analyze classification outputs to extract insights such as:

    - Common **adverse drug reactions (ADRs)** from side effect mentions.
    - Shifts in sentiment across different drugs or conditions.

### 3.4 Pattern

The system follows a modular pipeline pattern, separating data preprocessing, feature extraction, model training, and deployment for scalability and maintainability.

## 4.Non-Functional Requirements

1. **Performance**:The system should be optimized to classify and analyze a review within a few seconds, ensuring near real-time response for single queries and efficient batch processing for large datasets.

2. **Scalability**:The architecture must support horizontal scaling to handle increased loads, such as high-volume API requests or large-scale datasets.

3. **Security**:Input data must be sanitized to prevent common attacks such as SQL injection or cross-site scripting (XSS). All API endpoints should be protected using **HTTPS**, and optionally authenticated using tokens (e.g., JWT) if the service is extended to multiple users.

4. **Usability**:The UI must be clean, responsive, and intuitive, ensuring accessibility for non-technical users such as healthcare professionals or analysts. Include validation, tooltips, and clear error messages.

5. **Maintainability**:Code should follow modular principles, with clearly separated components (e.g., preprocessing, classification, result visualization). Each module should be well-documented, use version control (e.g., Git), and include unit tests to facilitate easy updates and debugging.

6. **Reliability**:The system must handle exceptions gracefully and continue operating even if a non-critical component fails. Implement logging and monitoring (e.g., using tools like Loguru or Prometheus) to ensure error detection and recovery.

7. **Portability**:The system should be platform-independent and easily deployable on local machines, virtual servers, or cloud platforms such as **Render**, **Heroku**,**Vercel**. Use configuration files and environment variables to ensure smooth deployment across different environments.

## 5.Technology Stack

### 1.Frontend:

- React.js: Used to develop a dynamic and responsive user interface for entering reviews and viewing prediction results.
- Axios / Fetch API: For sending asynchronous HTTP requests to the FastAPI backend and handling responses.
- Tailwind CSS: For designing a clean, consistent, and mobile-responsive interface with utility-first CSS classes.

### 2.Backend(Python):

- **FastAPI**: A high-performance web framework for building APIs quickly and asynchronously, ideal for ML model serving.
- **NLTK / SpaCy**: For natural language preprocessing, such as tokenization, lemmatization, and Named Entity Recognition.
- **Scikit-learn**: To implement traditional ML models like Naive Bayes, Passive Aggressive, Support Vector Machines, and basic feedforward neural networks.
- **Transformers (Hugging Face)**: To integrate pretrained transformer models like **BERT**, **DistilBERT**, or **BioBERT** for contextual classification and ADR extraction.

### 3. Dataset:

- **UCI ML Drug Review Dataset (from Kaggle)**: Contains structured fields including drug names, conditions, ratings, and free-text reviews. It serves as the foundation for training and evaluating the model.

### 4. Evaluation Metrics:

- **Accuracy**: Measures the overall correct predictions.
- **Precision**: Evaluates how many predicted positives are actually correct.
- **Recall**: Measures the ability to find all relevant instances.
- **F1-Score**: Harmonic mean of precision and recall, useful for imbalanced data.

### 5.Development Tools *:*

- **Jupyter Notebook / Google Colab**: For experimentation and initial model training.

- **VS Code:**As IDEs for backend development.
- **Git + GitHub**: For version control and collaboration.
- **Vercel + Render**: For packaging the application and simplifying deployment across environme

# 6.Summary

This project presents a comprehensive system for analyzing doctor-provided drug reviews using Natural Language Processing (NLP) and Machine Learning (ML) techniques to classify sentiment, predict medical conditions, and extract potential side effects. Utilizing the UCI ML Drug Review Dataset from Kaggle, the system is designed to categorize reviews into positive, neutral, or negative sentiments based on their ratings, while also identifying the medical conditions being treated and any adverse drug reactions (ADRs) mentioned in the text. The architecture follows a modular pipeline that includes text preprocessing (tokenization, stopword removal, lemmatization), feature extraction using Bag of Words and TF-IDF, and classification using models such as Naive Bayes, Passive Aggressive, SVM, and Neural Networks. To enhance semantic understanding, deep learning models like BERT, accessed via Hugging Face Transformers, are incorporated for improved prediction accuracy and context-aware side effect extraction. An optional React.js frontend and FastAPI backend enable user interaction, making the system suitable for real-time analytics and user-friendly deployment. With a focus on performance, scalability, and reliability, this system provides a clinically meaningful solution to assess drug efficacy, monitor satisfaction, detect adverse reactions, and support data-driven healthcare decisions. Ultimately, it bridges the gap between unstructured clinical text and actionable insights in the medical domain, with potential applications in pharmacovigilance, clinical research, and intelligent drug recommendation systems.