# Medicine Review Categorization using Machine Learning

## Project Description:

The **Medicine Review Categorization** project aims to classify reviews of medicines into categories such as **Excellent**, **Average**, or **Poor** based on the provided information. The model is trained using features like **Uses**, **Side Effects**, and **Manufacturer** of the medicine. This project leverages **Natural Language Processing (NLP)** techniques and **Machine Learning** to automatically categorize medicine reviews and predict their quality, making it easier for healthcare professionals, researchers, and consumers to assess the effectiveness of a medication.

## Objectives:

1. **Classify Medicine Reviews**: The goal of this project is to build a model that can classify the review of a medicine into three categories — Excellent, Average, or Poor — based on textual data and manufacturer information.
2. **Use Textual Data**: The model utilizes textual data provided in fields like **Uses** and **Side Effects** to understand the context of the medication and apply it to the classification task.
3. **Incorporate Manufacturer Information**: The manufacturer of the medicine is also a feature that helps in enhancing the prediction model, as certain manufacturers might have specific trends in their product reviews.
4. **Provide API for Prediction**: The project includes a **Flask-based API** that allows users to input new data (medicine use, side effects, and manufacturer) and receive the predicted review category.

## Key Components:

1. **Dataset**: The project uses a dataset of medicine details, including columns such as **Uses**, **Side Effects**, **Manufacturer**, and various review percentage columns like **Excellent Review %**, **Average Review %**, and **Poor Review %**.
2. **Target Variable**: The target variable, **Review_Category**, is created by categorizing the review based on which percentage is the highest — "Excellent", "Average", or "Poor".
3. **Textual Features**:
    - **Uses**: Describes the purpose of the medicine (e.g., "Pain relief", "Treatment of Bacterial infections").
    - **Side Effects**: Lists potential side effects of the medicine (e.g., "Vomiting", "Nausea", "Diarrhea").
    - **Manufacturer**: The company or organization that manufactures the medicine.
4. **Modeling Process**:
    - **Text Vectorization**: The text in the **Uses** and **Side Effects** columns is transformed into numerical representations using **TF-IDF** (Term Frequency-Inverse Document Frequency) vectorization.
    - **Label Encoding**: The **Manufacturer** information is encoded using **Label Encoding**.

    - **Random Forest Classifier**: A machine learning model is trained using a **Random Forest Classifier**, which is suitable for both numerical and categorical data.
5. **Flask API**: A simple Flask app exposes an API endpoint (`/predict`) where users can send POST requests with the required fields (**Uses**, **Side Effects**, **Manufacturer**) in JSON format. The model predicts the review category based on the provided data.

## Workflow:

1. **Data Preprocessing**:
    - Load and clean the dataset.
    - Create the target variable **Review_Category** based on review percentages.
    - Split the data into features (X) and the target variable (y).
2. **Feature Engineering**:
    - Use **TF-IDF** to convert textual features like **Uses** and **Side Effects** into numerical vectors.
    - Encode categorical features like **Manufacturer** using **LabelEncoder**.
    - Combine all features into a single feature matrix.
3. **Model Training**:
    - Split the dataset into training and testing sets.
    - Train the model using a **Random Forest Classifier** to predict the **Review_Category**.
4. **Model Evaluation**:
    - Evaluate the model using classification metrics like accuracy, confusion matrix, and classification report.
    - Fine-tune the model to optimize performance.
5. **Deployment**:
    - Deploy the trained model into a Flask web application.
    - Expose the `/predict` endpoint to allow users to send data and receive predictions.
6. **Testing**:
    - Test the deployed Flask app using tools like `curl`, `Postman`, or Python's `requests` library to ensure that the prediction API works as expected.

## Tools and Technologies:

- **Python**: The primary programming language used for data analysis, modeling, and API development.
- **Flask**: A lightweight web framework used to build and deploy the prediction API.
- **Pandas**: For data manipulation and preprocessing.
- **Scikit-learn**: For machine learning tasks such as data splitting, vectorization (TF-IDF), and model training (Random Forest Classifier).
- **Joblib**: Used for saving the trained machine learning model, vectorizers, and encoders.
- **curl/Postman**: Tools used to send API requests to test the deployed Flask app.

## Expected Outcome:

The application will be able to predict the review category (Excellent, Average, or Poor) for a given medicine based on its **Uses**, **Side Effects**, and **Manufacturer**. This can be useful for businesses in the pharmaceutical industry to analyze and categorize customer feedback, or for users who want quick insights into the overall review category of a particular medicine.

## Potential Improvements:

- **Multi-class Classification**: Although the model predicts one of three categories, it could be expanded to predict more detailed levels (e.g., Very Poor, Poor, Average, Good, Excellent).
- **Text Preprocessing**: Improvements can be made by cleaning and normalizing text data, such as removing stopwords or using more advanced text representation techniques like Word2Vec or BERT.
- **Web Interface**: Instead of using `curl` or `Postman`, the Flask app could be extended to include a frontend where users can input data and view predictions via a graphical interface.

This project demonstrates the application of machine learning in the healthcare domain, particularly in automating the process of categorizing medicine reviews, ultimately helping users and businesses make informed decisions about pharmaceutical products.