Patient Condition Classification and Drugs Recommendation

- The project aims to classify health conditions based on user-submitted symptoms or medication reviews and recommend the top drugs for the predicted condition.

- Used the Drug Review Dataset from the UCI Machine Learning Repository. This dataset contains:
  Drug Name: Name of the drug.
  Condition: The medical condition treated.
  Review: User's textual feedback.
  Rating: Patient's rating of the drug on a scale of 1-10.
  Useful Count: Count of users who found the review helpful.
  https://archive.ics.uci.edu/dataset/462/drug+review+dataset+drugs+com

1. Text Cleaning

Purpose:

To remove unwanted elements and simplify the input text for downstream tasks.

- Lowercasing:
  All text is converted to lowercase to ensure uniformity and avoid case-sensitive mismatches.
  Example: "The Drug Works Well" → "the drug works well"
- Removing Special Characters and Numbers:
  Special characters (@, #, !) and numbers are removed, as they do not add semantic meaning for condition classification.

2. Tokenization

Purpose:

Break down text into smaller units (tokens), such as words or phrases, for individual analysis.

- Definition: Tokenization splits text into meaningful elements.
  Example: "This drug is effective." → ["this", "drug", "is", "effective"]
- Why Important:
  Models and vectorizers like TF-IDF work at the token level, making tokenization a foundational step in NLP.

3. Stop Word Removal

Purpose:

To filter out common words that do not carry significant meaning (e.g., "the," "and," "is").

- How It Works:
  Stop words are removed from the tokenized text using a predefined list (e.g., NLTK stop words).
- Effect:
  Reduces noise and focuses on the words most relevant to classification.
  Example: "This drug is very effective and useful." → ["drug", "effective", "useful"]

4. Stemming and Lemmatization

Purpose:

To normalize words by reducing them to their base or root forms.

- Stemming:
  Cuts off suffixes from words to get the root form, often leading to non-dictionary words.
  Example: "running" → "run", "happier" → "happi"
- Lemmatization:
  Converts words to their base dictionary form by considering the context (e.g., part of speech).
  Example: "better" → "good", "studying" → "study"
- Choice in Project:
  Lemmatization was preferred for more accurate grammatical normalization.

5. Vectorization with TF-IDF

Purpose:

To convert the cleaned text into numerical format, which is required for machine learning models.

- What is TF-IDF?
  - Term Frequency (TF): Measures how frequently a term appears in a document.
    Formula: TF = (Number of occurrences of term t in a document) / (Total number of terms in the document)
  - Inverse Document Frequency (IDF): Assigns higher weights to terms unique to a document while penalizing common terms across all documents.
    Formula: IDF = log(Total number of documents / Number of documents with term t)
- Why TF-IDF?
  - Helps identify the most relevant words for each review.
  - Reduces the impact of commonly occurring terms (e.g., "drug," "take").

6. Passive Aggressive Classifier

The Passive Aggressive Classifier is an efficient machine learning model, particularly suited for online learning scenarios and text classification problems like yours.

- Passive: If the model makes a correct prediction, it does nothing (passive).
- Aggressive: If the model misclassifies a sample, it aggressively updates its decision boundary to correctly classify the sample.

Step 1: Input - TF-IDF Features

- After text preprocessing (cleaning, tokenization, stop word removal, and lemmatization), the reviews are converted into numerical vectors using TF-IDF.
- Example:
  Review: *"I feel restless and cannot sleep at night."*
  TF-IDF Vector: [0.2, 0.6, 0.4, 0.1, 0.0, ...]
  Features represent the importance of words like *"sleep"* or *"restless."*

Step 2: Condition Prediction - PAC

- The TF-IDF vector is passed to the Passive Aggressive Classifier, which classifies the review into a specific medical condition.
- The PAC optimizes its decision boundary based on whether it correctly or incorrectly classifies the review.
    - If correct → No update (passive).
    - If incorrect → Updates aggressively to minimize classification error.

7. Condition Prediction Output

- A user-submitted symptom (e.g., *"I feel sad, low energy, and tired all day."*) is classified into one of the predefined conditions:
    - Depression
    - Anxiety
    - Insomnia
    - Migraine
    - High Blood Pressure
    - Diabetes Type 2

Predicted Condition: *"Depression"*

Drug Recommendation System

Once the condition was predicted, the system recommended the best drugs based on user ratings and usefulness.

1. Filter Data:
    - From the dataset, drugs were filtered based on two criteria:
        - High Ratings (≥ 9): Ensured the drugs were effective for their conditions.
        - High Usefulness Count (≥ 100): Ensured recommendations were based on well-reviewed drugs.
2. Sort and Select:
    - Drugs were sorted by their rating and usefulness scores in descending order.
    - The top 3 unique drugs for the predicted condition were selected to avoid duplicates.

Output:

- Once the condition was predicted, the system displayed:
    1. Predicted Condition: Based on the user's input.
    2. Top Drug Recommendations: A list of up to 3 drugs highly rated for that condition.

Example

1. **Input Review**:
   "I have trouble sleeping and feel restless all night."
2. **Preprocessing**:
   Cleaned and tokenized: *["trouble", "sleeping", "restless", "night"]*.
3. **Vectorization (TF-IDF)**:
   TF-IDF vector generated: [0.2, 0.5, 0.7, 0.3, 0.0, ...].
4. **Condition Classification (PAC)**:
   PAC predicts: **"Insomnia"**.
5. **Drug Recommendation**:
   Retrieve drugs for Insomnia → Filter by rating and usefulness → Top 3:
    - *"Zolpidem"*
    - *"Eszopiclone"*
    - *"Trazodone"*