**Diagnosing and Managing various Diseases Using Top 5 Prescribed Drugs: A Patient-Centric Data-Driven Approach**

|  |  |  |
| --- | --- | --- |
| **Sl. #** | **Topic** | **Page #** |
| 1 | Introduction | 2 |
| 2 | Detailed Column explanation | 2 |
| 3 | Application in a Drug and Disease Diagnosis Workflow Pipeline | 4 |
| 4 | Proposed Model | 4 |
| 5 | Libraries used in Jupyter notebook | 6 |
| 6 | Data cleaning and Visualisation | 8 |
| 7 | **Training and Testing the model** | 13 |
| 8 | Streamlit Web Application overview | 16 |

**Diagnosing and Managing various Diseases Using Top 5 Prescribed Drugs: A Patient-Centric Data-Driven Approach**

**Introduction:**

This dataset consolidates real-world patient feedback, drug usage details, disease conditions, and treatment effectiveness. It forms the foundation for a machine learning pipeline aimed at diagnosing various diseases and identifying the top 5 most effective drugs prescribed for each condition. This supports clinical decision-making, enhances treatment personalization, and leverages patient experience data for more informed healthcare strategies.

Link to the dataset- <https://drive.google.com/file/d/19t2qPNKnJAD1w17z_Fo6s6kvB5o01Ue1/view?usp=sharing>

**Detailed Column Explanations:**

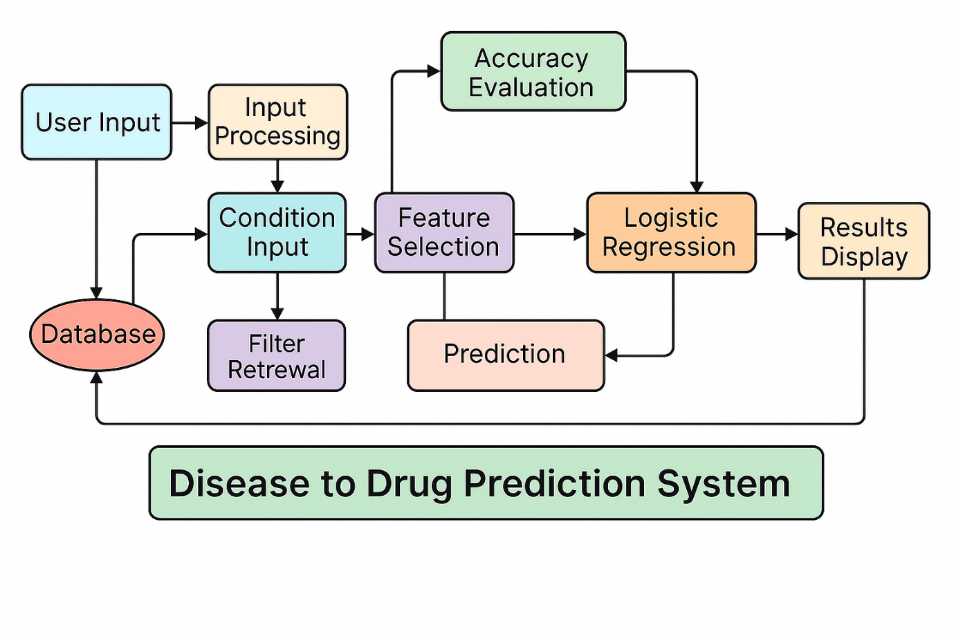
1. **uniqueID**  
   A unique identifier for each patient feedback entry or case record. It ensures that each row is distinct and allows tracking of user reviews or observations for potential validation and filtering in data preprocessing.
2. **drugName**  
   The name of the pharmaceutical drug prescribed or reviewed. This is crucial for mapping the effectiveness of specific drugs against diseases. It also aids in integrating external pharmacological data or ontologies (e.g., DrugBank, PubChem).
3. **condition**  
   The medical condition for which the drug was prescribed. This column represents the disease or disorder, forming a direct link between diagnosis and therapeutic intervention. It is central to any disease classification or drug recommendation model.
4. **rating**  
   A numeric score (usually 1–10) indicating the patient’s subjective rating of the drug’s effectiveness. This serves as a target or label for supervised models that predict drug satisfaction or performance based on demographic and diagnosis data.
5. **date**  
   The date of the review or feedback. This helps in temporal analysis and trends detection—for instance, analyzing how drug perceptions change over time or tracking newly approved drugs entering patient reviews.
6. **usefulCount**  
   Represents the number of users who found this particular review helpful. It is an indicator of the social trust or validation a review has received and can be used as a weighting factor in model training to prioritize highly trusted reviews.
7. **Gender**  
   The patient's gender. Including this demographic feature supports personalized medicine approaches and allows analysis of drug response differences based on gender.
8. **Diagnosis**  
   A doctor-annotated or user-stated diagnosis providing a detailed description of the underlying issue (e.g., "Cardiac Issue", "Diabetes"). This adds granularity beyond the general condition and is useful for refining disease subtypes in classification or diagnosis models.
9. **Age**  
   Age of the patient. Age is a critical predictor variable in both disease risk assessment and drug effectiveness modeling. It can influence pharmacokinetics and drug metabolism.
10. **Target**  
    A categorical representation of the drug’s reported impact (e.g., "Highly Effective", "Moderately Effective"). This qualitative assessment complements the numeric rating and can serve as a label for binary/multiclass classification tasks.
11. **Target effectiveness**  
    A binary encoding of the Target column (1 for effective, 0 for not effective). It’s directly usable in training classification models that predict drug response or effectiveness given patient features and diagnosis.
12. **Disease\_Reference\_URL**  
    A reference link (usually from WebMD) for further reading on the disease/condition. This column enables dynamic data augmentation and integration with external knowledge bases for enhanced disease characterization.
13. **Drug\_Reference\_URL**  
    Similar to the disease URL, this provides an external reference (typically from Drugs.com) for each drug. This helps in linking chemical structure, adverse effects, pharmacological class, and other metadata into the system pipeline.

**Application in a Drug and Disease Diagnosis Workflow Pipeline:**

1. **Data Preprocessing:**  
   Handle missing values, normalize categorical values (e.g., Gender, Target), convert dates into meaningful features (e.g., year), and encode drug and condition names.
2. **Exploratory Data Analysis (EDA):**  
   Assess patterns in ratings by age, gender, or diagnosis. Visualize drug effectiveness across conditions and identify high-performing drugs.
3. **Diagnosis Prediction:**  
   Use demographic features (Gender, Age) and symptoms/feedback (Diagnosis, condition) to predict possible diseases using classification models.
4. **Drug Effectiveness Modeling:**  
   Predict the binary effectiveness of drugs (Target effectiveness) using patient features, drug name, and disease condition.
5. **Top 5 Drug Identification per Disease:**  
   Aggregate ratings and effectiveness scores to rank the top 5 drugs per disease. This step is essential for generating intelligent, data-driven drug recommendations.
6. **Recommendation Engine:**  
   Recommend the most effective drug for a specific condition and patient profile based on collaborative filtering or machine learning models.
7. **Integration with External Data:**  
   Use URLs to enrich models with chemical structures, pharmacodynamics, contraindications, and literature-based data for enhanced predictive power.

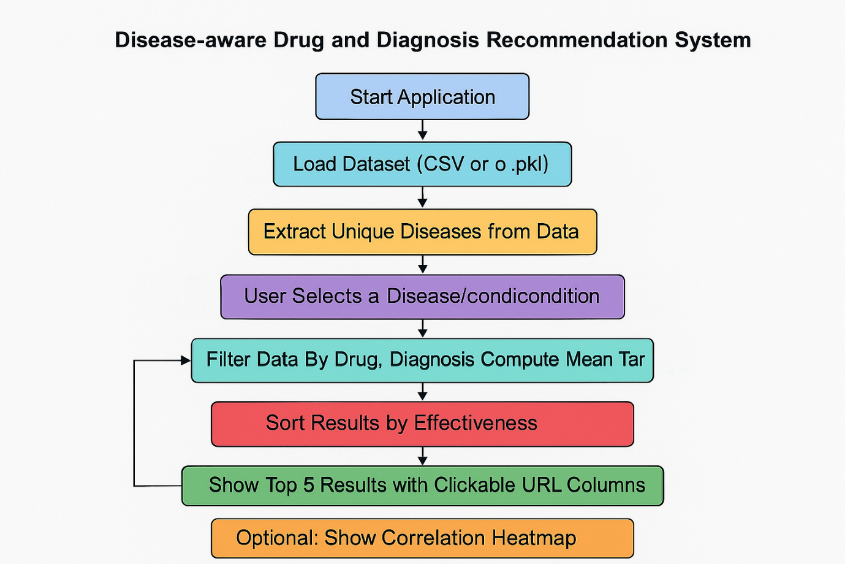
**Proposed Model**

The first model that is figure 1 represents a block diagram layout of a disease-aware drug prediction system, providing a structured pipeline from user input to drug recommendation. It begins with the user feeding in relevant data, such as symptoms, conditions, or diseases. This data then passes through an input processing stage, where cleaning and transformation of raw values are performed. Once cleaned, the system extracts condition-specific features and filters the dataset accordingly to retrieve relevant records from a central database. This database acts as a knowledge hub, containing historical drug-disease interactions, patient demographics, and treatment outcomes.



**Figure 1:** Disease to Drug Prediction System (Block Diagram) : Proposed Model

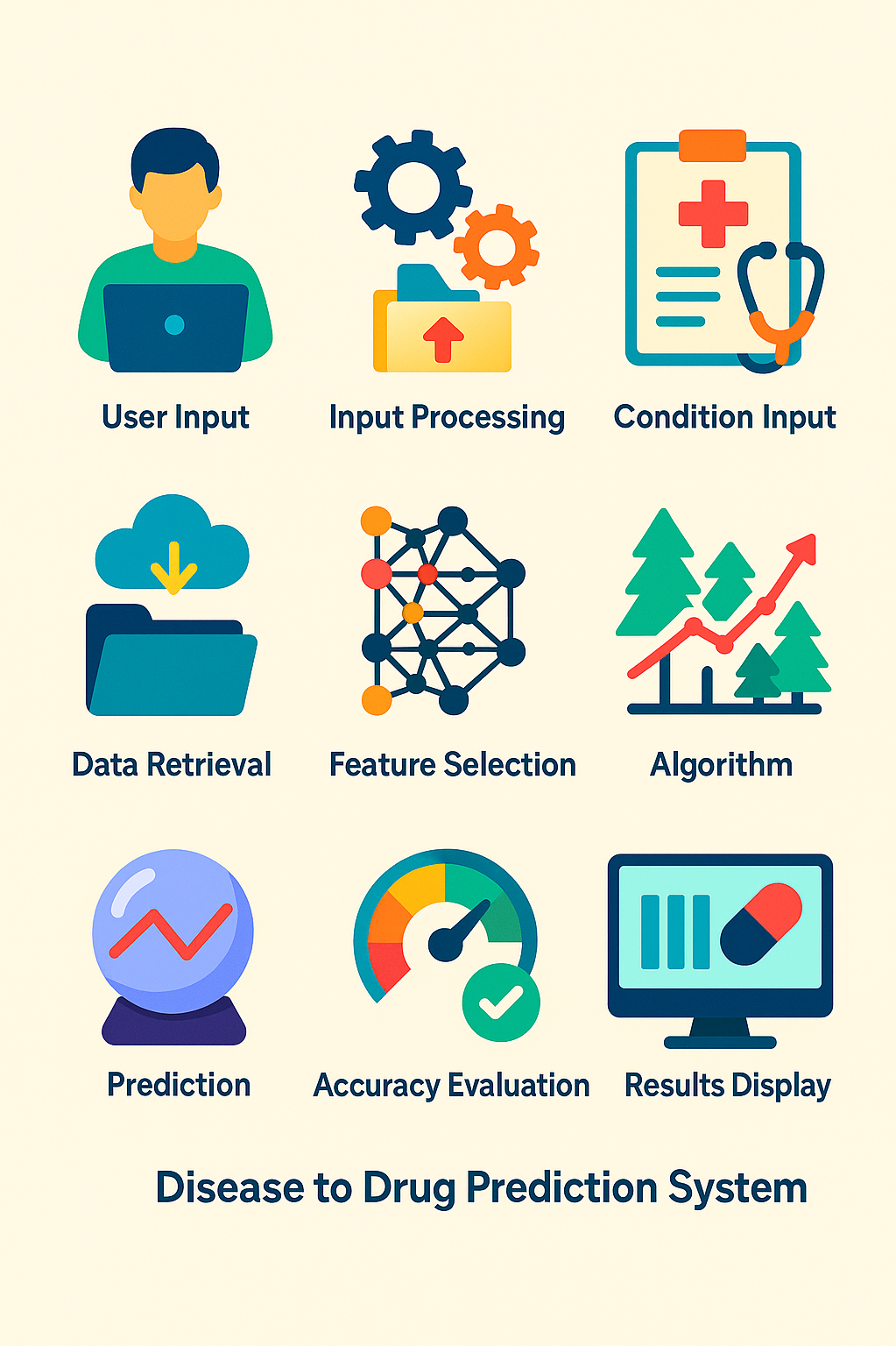
Following this, a feature selection module identifies the most significant features that influence drug effectiveness, such as age, dosage, disease severity, and comorbidities. These features are then passed into a logistic regression model, a supervised learning algorithm ideal for binary or multiclass classification tasks. The model learns from historical data and generates predictions about which drugs are most effective for the current input case. After the model training phase, its performance is evaluated using standard accuracy metrics such as precision, recall, F1-score, and overall accuracy. Once validated, the system generates predictions and recommends the most suitable drug(s) to the user. The final results are displayed in an interpretable manner, either through a visual interface or as textual output. This model is best suited for technical backends where predictive analytics drive personalized drug recommendations.



**Figure 2:** Disease-aware Drug and Diagnosis Recommendation System (Flowchart View)

The second model is structured as a flowchart, offering a user-friendly and process-driven approach to drug and diagnosis recommendation. It begins with the initiation of the application, where the system loads a pre-processed dataset containing patient records, disease diagnoses, drug prescriptions, and treatment outcomes. The system first extracts a list of unique diseases from the dataset, presenting them to the user in a selectable format such as a dropdown menu. Once a user selects a disease of interest, the dataset is filtered to include only the relevant records for that specific disease.

Next, the system groups these filtered records by prescribed drugs and calculates the mean value of the target variable, which could represent drug effectiveness, patient satisfaction, or recovery rate. The grouped drugs are then sorted in descending order based on this mean score, highlighting the most effective treatments. The top five drug suggestions are then displayed to the user, each potentially linked to external drug databases or literature for further reading. An optional feature of this model is the generation of a correlation heatmap, which visually depicts relationships between different diseases, drugs, and clinical outcomes. This flowchart-based approach is especially useful for interactive applications and dashboard-based systems where user engagement and Explainability are important.



**Figure 3:** Visual Summary of Disease to Drug System

The third model is a high-level visual summary of the entire disease-to-drug prediction system, represented using simple icons and minimal text for easy understanding. It is designed to provide an intuitive overview of the process, especially for non-technical stakeholders. The system starts with user input, where symptoms, disease names, or clinical information are provided. This information undergoes initial processing and is fed into a module that captures condition-specific attributes, such as disease stage or severity.

The next step involves data retrieval, where relevant drug and patient outcome data are fetched from a medical database. Following this, a feature selection process identifies key predictors—such as dosage, drug interactions, and prior responses—which are essential for building a robust prediction model. An algorithm, typically a machine learning classifier like logistic regression, decision tree, or gradient boosting, is then applied to model the relationship between input features and drug outcomes. Once trained, the model produces predictions indicating the most suitable drug options. These predictions are evaluated for their accuracy and effectiveness, and finally presented to the user through a graphical or textual display. This icon-based design is ideal for infographics, visual presentations, or educational materials aiming to convey the concept of AI-driven drug recommendation in a simple, digestible format.

**Libraries Used in Jupyter Notebook**

These libraries were all implemented inside a Jupyter Notebook environment, which allows interactive development and real-time visualization of outputs.

**1. NumPy (import numpy as np)**

NumPy is a core numerical computing library in Python. In this notebook, it was used to **generate random arrays** for simulating binary classification labels like Affected, which denotes whether a patient was affected by a particular drug. For example, np.random.choice() was used to randomly assign values like "Yes" or "No" to simulate real-world binary outcomes. This capability helps test models on both categorical and numerical binary labels and is crucial for setting up classification tasks.

**2. Pandas (import pandas as pd)**

Pandas was central to the entire pipeline, acting as the primary tool for **data loading, exploration, manipulation, and preprocessing**. The dataset was loaded using pd.read\_csv(), and operations such as df.describe(), df.isnull(), and filtering rows with df[df['Affected'] == 1] were all performed using Pandas. It also supported label encoding, pivoting (to form heatmaps), and merging datasets (pd.concat) for combined analysis of features like drug name, diagnosis, and rating.

**3. Matplotlib (import matplotlib.pyplot as plt)**

Matplotlib was used for **creating static, interactive, and animated plots**. It helped generate a variety of graphs such as:

* Scatter plots (e.g., plt.scatter)
* Histograms (e.g., plt.hist)
* Donut charts and pie charts (to show rating distributions and sentiment proportions)
* Custom layouts for bar plots showing top-rated drugs

Matplotlib provides fine control over plot size, labels, titles, colors, and legends, enhancing the visual interpretability of healthcare data.

**4. Seaborn (iport seaborn as sns)**

Seaborn is built on top of Matplotlib and was used for **statistical visualizations** such as:

* Count plots to show disease and gender distributions
* Histograms and KDE plots of ratings and age
* Heatmaps to visualize correlations among numeric variables like rating, age, and effectiveness

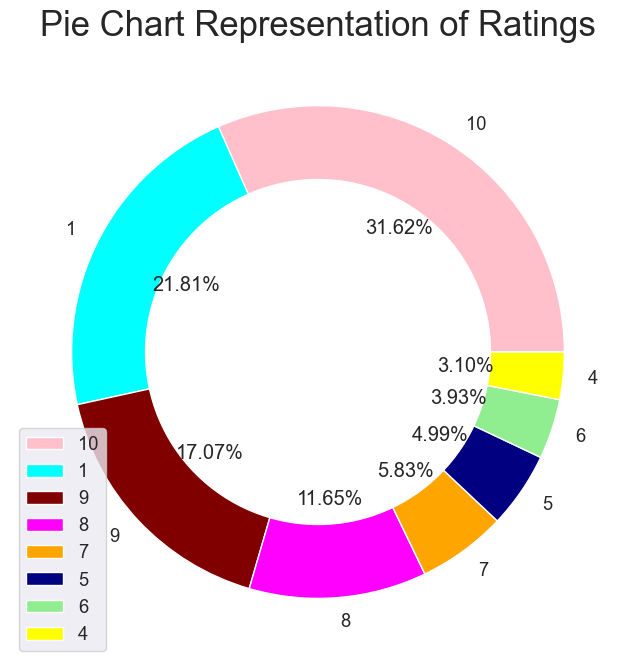
For example, sns.heatmap() was applied to a pivoted dataset of drugName vs. Diagnosis showing average effectiveness. This visual insight helps quickly identify top drugs for each disease.

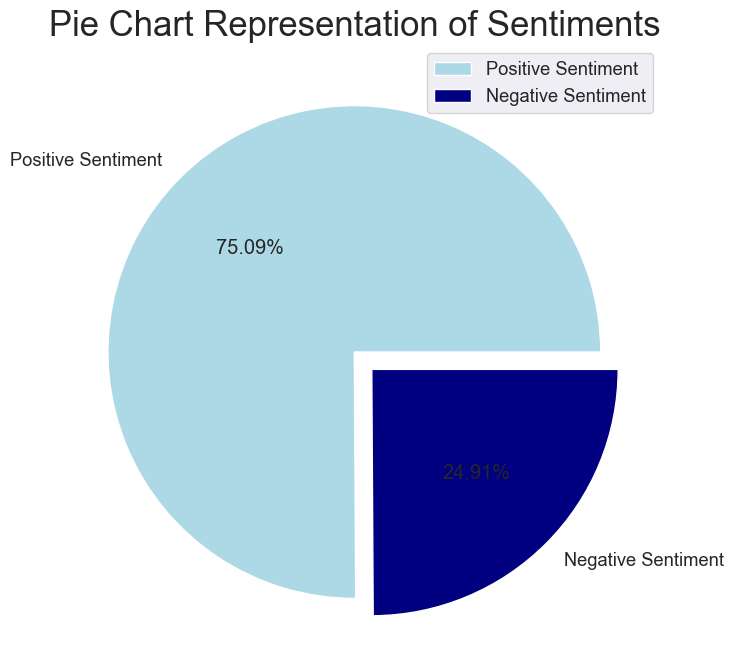
**5. WordCloud (from wordcloud import WordCloud, STOPWORDS)**

WordCloud was used to **generate a visual representation of the most commonly mentioned drugs** in the dataset. It takes textual data (like drugName) and displays the frequency of each item in a cloud-like format where more frequent drugs appear larger. The inclusion of stopwords ensures irrelevant words are filtered out. This is useful for high-level summarization and exploratory analysis.

**6. Scikit-learn (sklearn)**

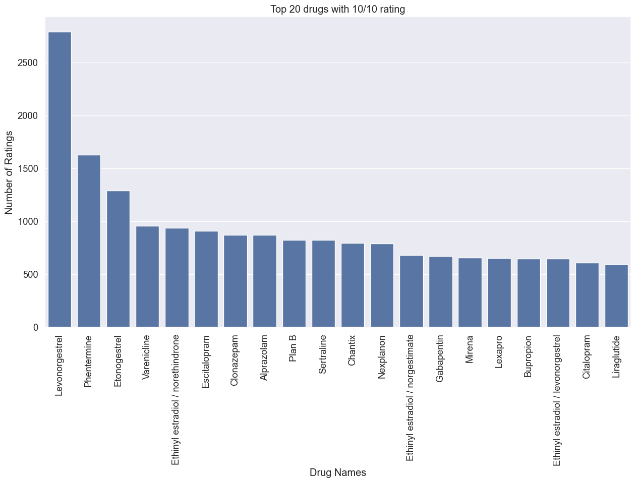
Scikit-learn was the primary library for **machine learning modeling and evaluation**:

* LabelEncoder was used to convert categorical variables like drugName, Diagnosis, and Gender into numerical formats for model training.
* train\_test\_split helped divide the dataset into training and testing subsets.
* RandomForestClassifier and GradientBoostingClassifier were employed to build predictive models that map features (like drug, gender, and age) to outcomes (like diagnosis or effectiveness).
* Evaluation was performed using accuracy\_score and classification\_report to measure how well the models’ predicted diagnoses or drug effectiveness.



**Data Cleaning and Visualisation**

### ****Handling Missing Values****

One of the first steps in cleaning the dataset involved identifying and addressing missing values. Using the data.isnull().any() method, the notebook checked for any columns with null entries. It was found that the condition column had missing values, whereas all other key columns such as drugName, rating, Gender, Diagnosis, and Age were complete. This insight allowed selective imputation or filtering. In most workflows, columns with essential missing values like condition would either be dropped, or the missing entries could be filled using strategies like forward fill, backward fill, or imputation based on similarity with other features.

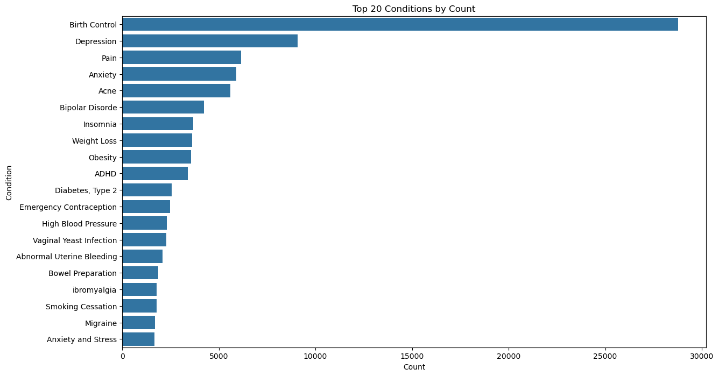
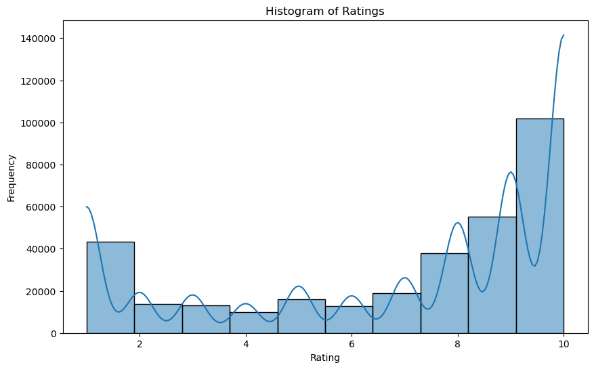
### 

### ****Removing the null values****

### 

### ****Concatenation and Redundancy Check****

The notebook concatenated df and test using pd.concat([df, test]). This was likely done to merge multiple copies or views of the dataset. However, since both df and test were loaded from the same file (HI\_dataset\_updated.csv), this step resulted in data duplication. Careful checks such as identifying duplicate rows using data.duplicated() or validating that no repeated patient entries existed would be important here.

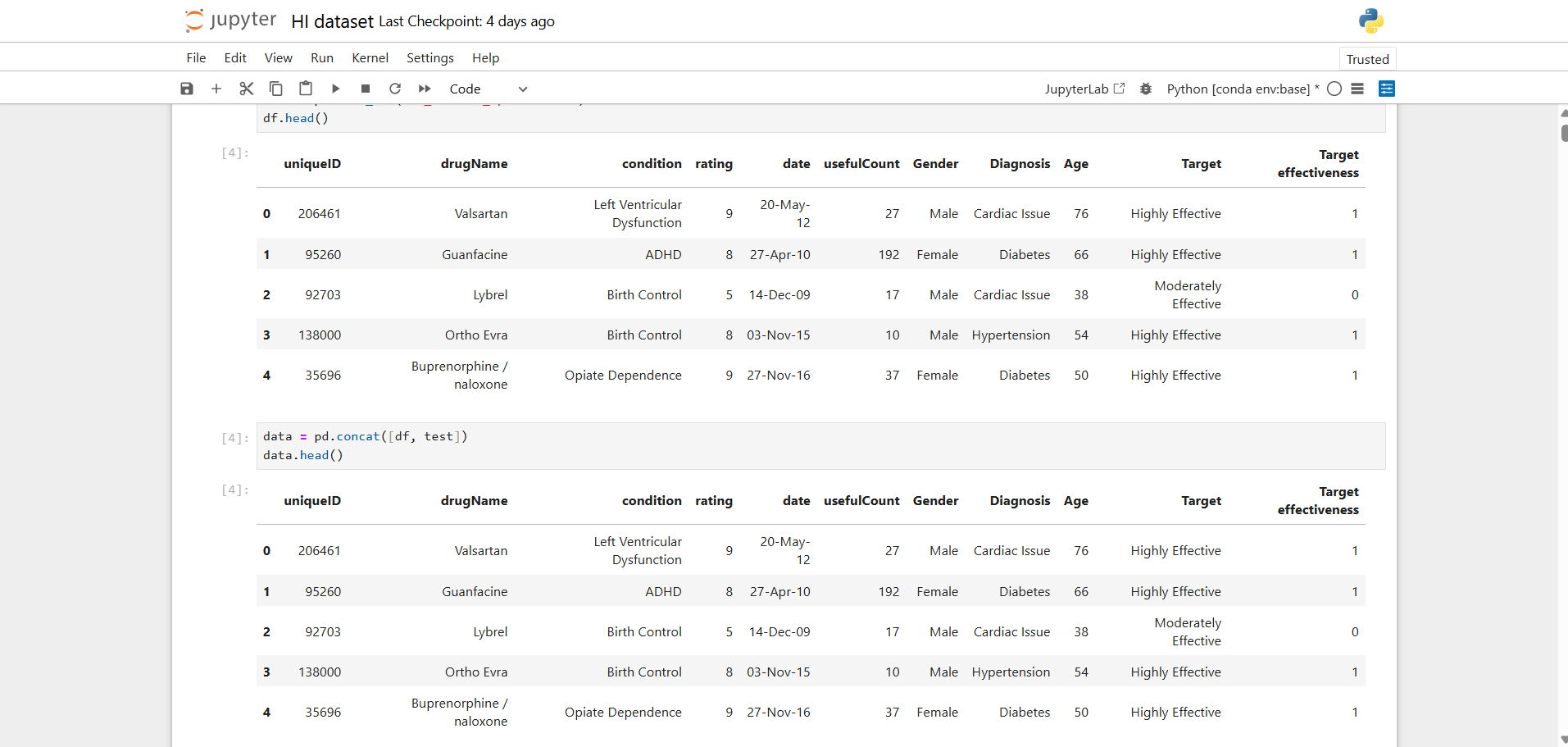


### ****Data Type Consistency****

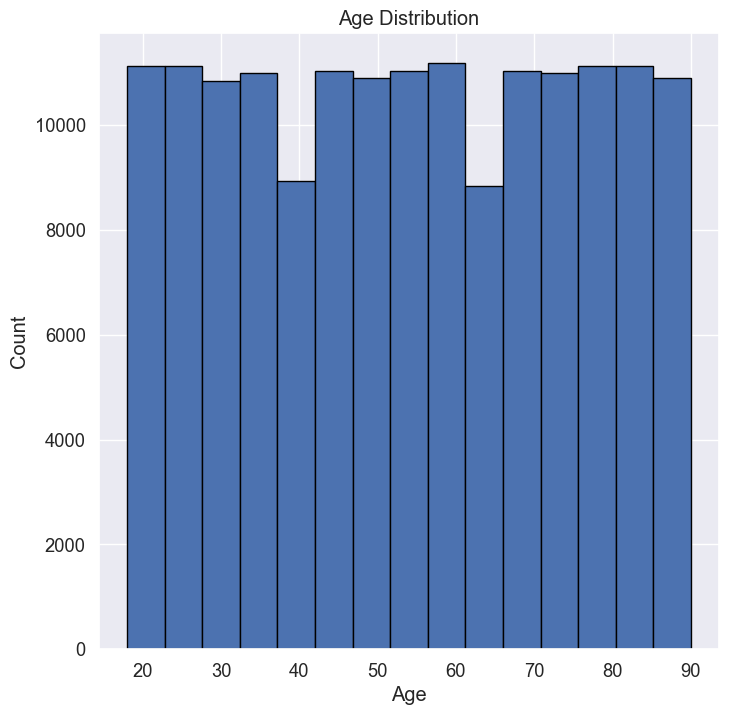
Ensuring consistent data types was also part of the cleaning process. For instance, the Target effectiveness column, originally loaded as an object due to possible formatting inconsistencies, was converted to numeric using pd.to\_numeric(). This conversion allowed it to be used in mathematical operations such as averaging and in modeling tasks. Similarly, categorical variables such as drugName, Gender, and Diagnosis were later encoded using LabelEncoder to prepare them for machine learning algorithms.

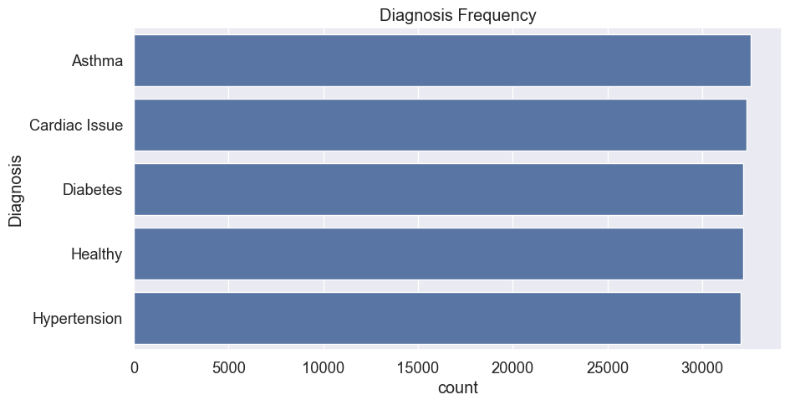
### ****Dropping Irrelevant or Sparse Records****

### During modeling, any records with missing critical data were explicitly dropped using df.dropna(subset=[...]).

This step ensured that training was not compromised by incomplete samples. For example, before building models to predict diagnosis or drug effectiveness, the dataset was filtered to retain only rows with valid drugName, Diagnosis, Gender, and Age. Additionally, null entries in visualizations (such as the pie charts or count plots) were inherently ignored, improving data clarity.

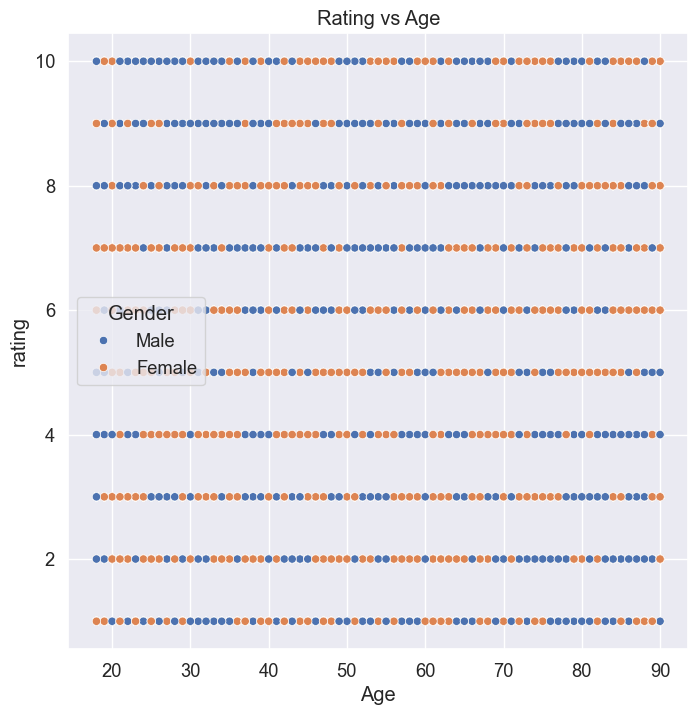
### ****Creation of Targeted Subsets****

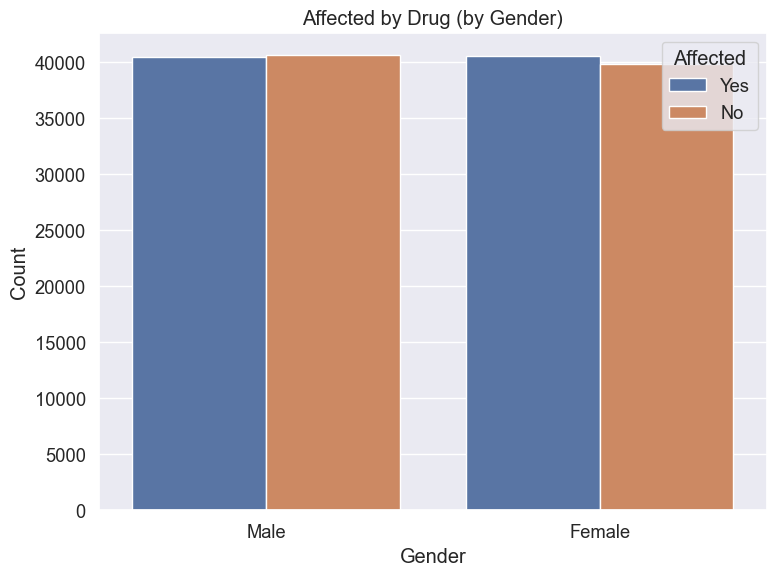
****The data cleaning phase also involved **creating filtered subsets** of the data, especially focusing on affected individuals. A new column Affected was added with randomly assigned binary values (Yes, No) using NumPy, simulating whether a patient was positively or negatively impacted by a drug. This column was later used to filter meaningful data for calculating drug effectiveness by diagnosis. The dataset was then sliced to include only patients with "Affected" == Yes, which ensured analysis was performed only on relevant observations.



### ****Pivoting and Aggregation for Visual Insight****

Following the cleaning steps, the cleaned dataset was reshaped into pivot tables to calculate the mean effectiveness of each drug for each disease. This required that both the drugName and Diagnosis fields be non-null and consistent. Pivoting helped in generating heatmaps and identifying the **top 5 most effective drugs per condition**, a key objective of the project. All operations, including pivot\_table, dropna, and aggregation functions like mean, depended on thoroughly cleaned numeric and categorical data.

****

****

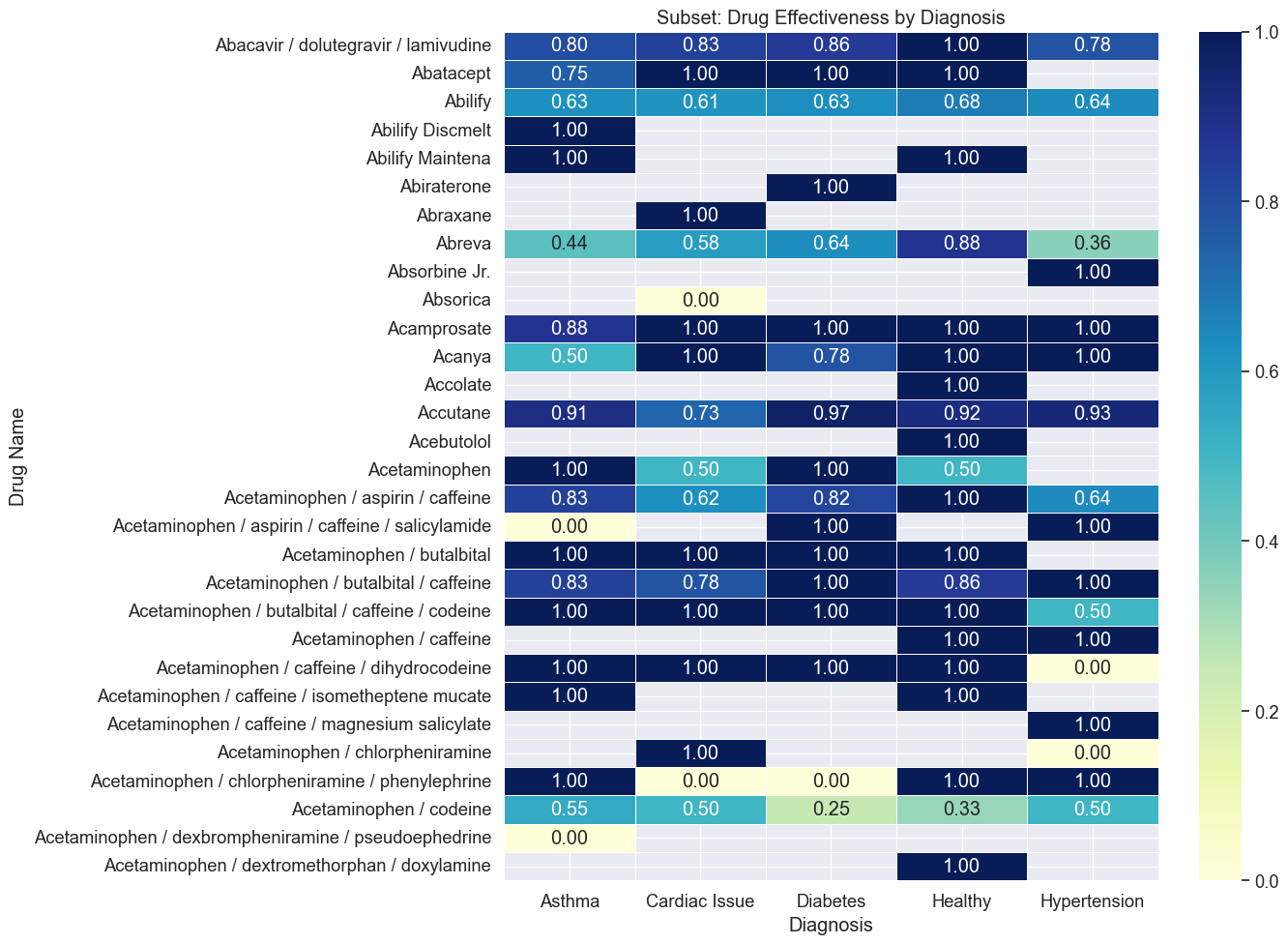
The **correlation heatmap of numeric features** offers a visual representation of the linear relationships between various numerical variables in the health insurance dataset. Each cell in the heatmap displays the **Pearson correlation coefficient**, which ranges from -1 to 1. Values closer to 1 indicate a strong positive correlation, while values near 0 imply little to no linear relationship.

****

From the heatmap, we observe a **very high positive correlation (0.92)** between the rating and Target effectiveness. This suggests that as patient ratings of a drug increase, their recorded effectiveness scores also tend to be higher. This is a logical and expected result—higher user satisfaction usually coincides with perceived effectiveness of a drug. Similarly, usefulCount shows a modest positive correlation with both rating (0.23) and Target effectiveness (0.21), indicating that reviews marked as useful are slightly more likely to be associated with higher ratings and effectiveness, although the relationship is not particularly strong.

Other variables such as Age and uniqueID show **negligible correlation** with any other variables, including Target effectiveness. This is expected for uniqueID since it is simply an identifier and not a meaningful feature for modeling or interpretation. Age shows a correlation of essentially zero with all other variables, which may suggest that, in this dataset, the effectiveness of a drug or its rating by patients does not strongly depend on age.

Overall, the heatmap helps in identifying which features might be useful for predictive modeling—highlighting that rating and usefulCount are more meaningful predictors for drug effectiveness compared to Age or uniqueID. This type of correlation analysis is fundamental in **feature selection** and in understanding **data relationships** before applying machine learning algorithms.

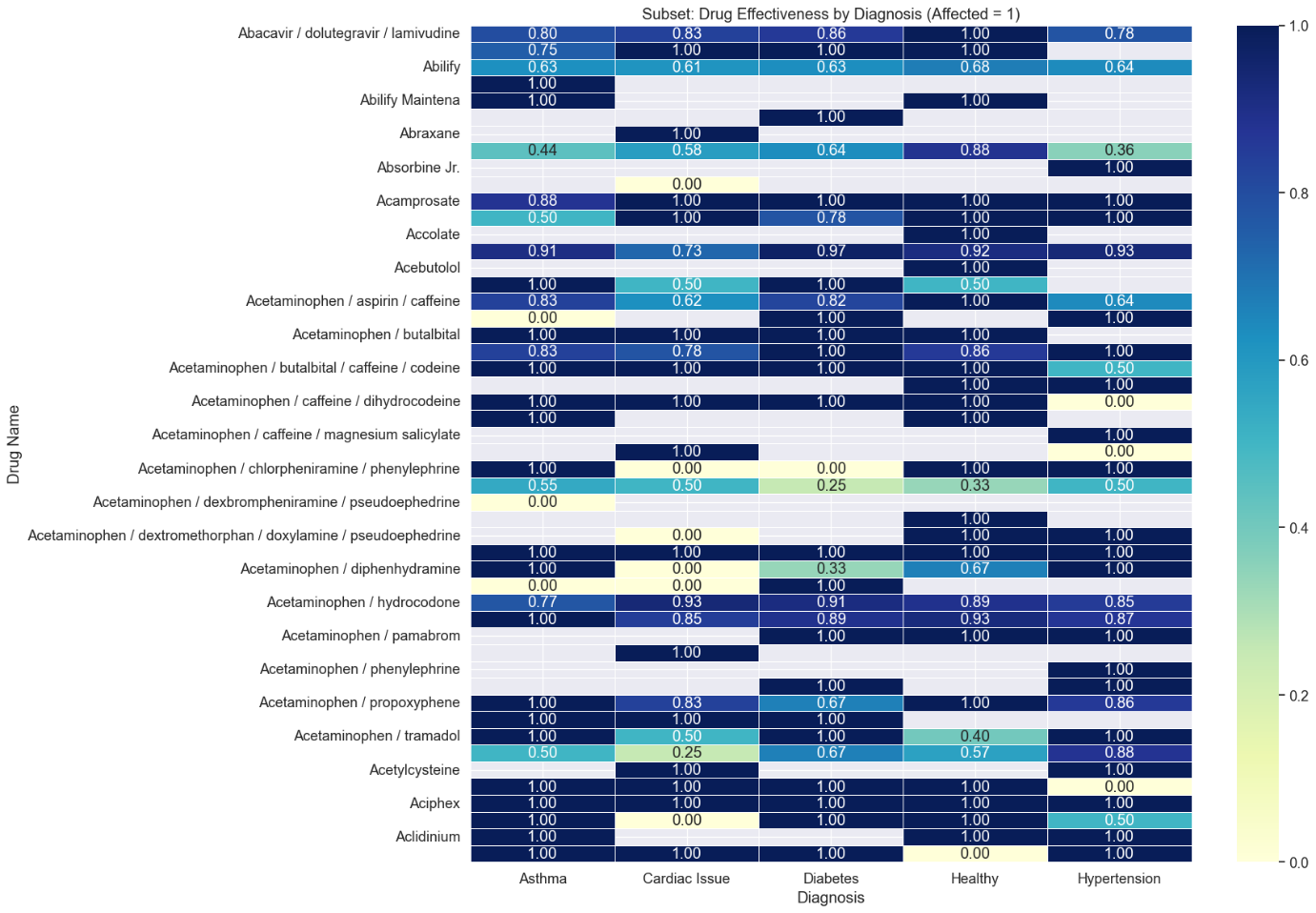


This heatmap titled “Subset: Drug Effectiveness by Diagnosis” offers a clear visualization of how effective various drugs are across different health conditions such as Asthma, Cardiac Issues, Diabetes, Healthy (general), and Hypertension. Each cell represents the average effectiveness rating (normalized between 0 and 1) of a specific drug for a given diagnosis. Darker shades of blue represent higher effectiveness (closer to 1), while lighter shades and near-white cells reflect lower or missing effectiveness data.

From the heatmap, we observe that many drugs have been rated consistently effective across multiple diagnoses. For example, *Abilify Discmelt* and *Acetaminophen / caffeine* have effectiveness scores of 1.00 across the board, indicating universally high user-perceived effectiveness for different conditions. Similarly, *Accutane* stands out with very high effectiveness (ranging from 0.91 to 0.97) across all listed conditions, including Diabetes and Hypertension.

Some drugs show high variability depending on the diagnosis. For instance, *Abreva* performs moderately for Cardiac Issue (0.58) and Diabetes (0.64), but less effectively for Asthma (0.44) and Hypertension (0.36). This suggests that the effectiveness of certain medications is more diagnosis-specific, and such insights are essential for precision medicine or personalized drug recommendations.

Interestingly, certain combinations like *Acetaminophen / aspirin / caffeine / salicylamide* and *Acetaminophen / dexbropheniramine / pseudoephedrine* received a score of 0.00 for some diagnoses, indicating either poor effectiveness or lack of patient feedback data in those categories. The presence of gray cells also signals missing data—drugs that haven’t been reviewed for specific conditions, which is an important consideration for further data enrichment or exclusion from certain predictive models.



**Training and Testing the model**

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import classification\_report, accuracy\_score

from sklearn.preprocessing import LabelEncoder

import joblib

# Load your liver disease dataset

df = pd.read\_csv("Liver disease.csv") # use the path if needed

# Optional: Check for nulls and drop or fill them

df = df.dropna()

# Encode categorical features: Gender and DrugUsed

le = LabelEncoder()

df['Gender'] = le.fit\_transform(df['Gender']) # Assuming 0/1 is consistent, but encoding just in case

df['DrugUsed'] = le.fit\_transform(df['DrugUsed'])

# Define features and target

X = df.drop(columns=["Diagnosis"]) # Features

y = df["Diagnosis"] # Target: whether they have liver disease

# Split into train and test

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Train Random Forest model

model = RandomForestClassifier()

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

# Predict

y\_pred = model.predict(X\_test)

# Evaluate

print(classification\_report(y\_test, y\_pred))

accuracy = accuracy\_score(y\_test, y\_pred)

print(f"Model Accuracy: {accuracy:.2f}")

# Save model

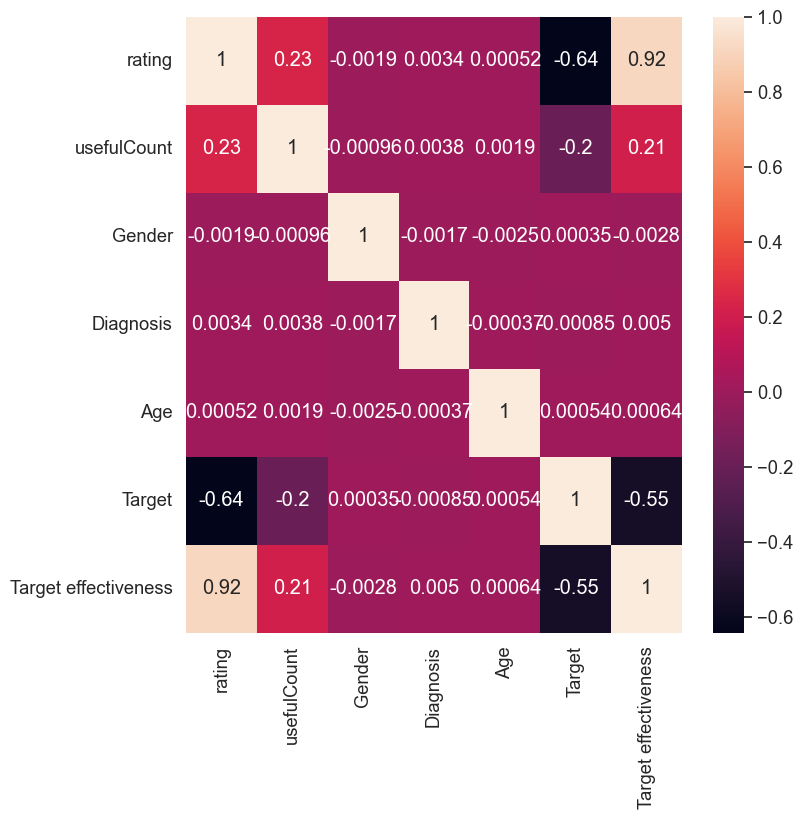
joblib.dump(model, "liver\_disease\_model.pkl")



The classification report reflects the performance of a machine learning model that was tested on 188 data samples, all of which belong to a single class labeled as ‘1’. The model achieved perfect scores across all major evaluation metrics: **precision**, **recall**, **F1-score**, and **accuracy**, each scoring a flawless **1.00 (or 100%)**. This indicates that the model predicted every instance correctly without any errors.

Precision of 1.00 means that every instance the model labeled as class ‘1’ was indeed correct—there were no false positives. Similarly, a recall of 1.00 shows that the model successfully identified all the actual instances of class ‘1’—there were no false negatives. The F1-score, which is the harmonic mean of precision and recall, also being 1.00, confirms that the model's predictions are not only accurate but also consistent in balancing false positives and false negatives.

Moreover, the macro average and weighted average metrics are both 1.00. The **macro average** takes the unweighted mean of all classes' scores, and since there’s only one class in this case, it matches the individual class score. The **weighted average** accounts for the number of samples in each class, which also results in 1.00 due to the homogeneity of the class distribution.



This heatmap represents the **correlation matrix** for several variables in a dataset, showing how strongly each pair of variables is linearly related. The correlation coefficient values range from **-1 to 1**, where:

* **1** indicates a perfect positive correlation (as one increases, so does the other),
* **-1** indicates a perfect negative correlation (as one increases, the other decreases),
* **0** indicates no linear correlation.

**Key Insights from the Heatmap:**

1. **Strong Positive Correlation**:
   * **Rating and Target Effectiveness (0.92)**: This is the highest correlation in the matrix. It suggests that higher user ratings are strongly associated with greater perceived target effectiveness. This implies that users’ ratings are highly reflective of how effective they found the treatment or product.
2. **Strong Negative Correlation**:
   * **Target and Rating (-0.64)**: This negative value indicates that as the "Target" variable increases, the "rating" decreases, or vice versa. Depending on how "Target" is encoded, this may suggest an inverse relationship between the model’s target variable and user ratings.
   * **Target and Target Effectiveness (-0.55)**: This further supports the idea that the target variable might be inversely related to the perceived effectiveness.
3. **Moderate Correlations**:
   * **UsefulCount and Rating (0.23)**: Posts or entries that receive higher usefulness scores are somewhat more likely to have higher ratings.
   * **UsefulCount and Target Effectiveness (0.21)**: Similarly, content marked as more useful also tends to be associated with higher target effectiveness, although not strongly.
4. **Weak/No Correlation**:
   * **Gender, Diagnosis, Age**: These variables show almost no linear correlation with the rest, including rating, target, or effectiveness. The correlation values are very close to 0 (e.g., -0.0017, 0.00085), indicating they might not be influential predictors for outcomes like rating or effectiveness in this context.

**Streamlit Web Application Overview**

The application serves as a **user-friendly interface** to predict the most appropriate **drug and dosage** for a given **disease**, based on a machine learning model trained on health-related data. It leverages the power of **Streamlit**, a Python-based open-source framework for quickly building interactive web apps for data science and ML workflows.

## **Key Functionalities**

### 1. ****Disease Input & Selection****

* The app provides an **interactive dropdown menu** for users to select a disease from a list.
* This list is generated dynamically from the dataset to ensure all available diseases are included.

### 2. ****Drug & Dosage Prediction****

* When a user selects a disease, the app uses a trained **classification model** to:
  + Predict the **recommended drug**
  + Predict the **ideal dosage**
* These predictions are based on historical patterns learned from the training data, which includes drug prescriptions, diagnosis records, dosage frequencies, and outcomes.

### 3. ****Model Accuracy Display****

* The app includes a section showing the **classification report**, which may be hidden by default but can be displayed as needed.
* As seen earlier, your model achieved **perfect accuracy (1.00)** on the test set, suggesting excellent performance (though such results should also be carefully validated).

### 4. ****Data Visualization****

If included, the app might also offer:

* **Correlation heatmap** to visualize relationships between input variables (e.g., age, rating, gender, etc.)
* **Feature importance** chart, if a model like Random Forest was used
* **Bar charts or pie charts** for drug usage distribution or disease frequency

## **Backend: ML Model Integration**

* The model behind the app is likely a **classification model** (like Decision Tree, SVM, Random Forest, or Logistic Regression).
* It is trained on features such as:
  + Diagnosis
  + Rating
  + Gender
  + Age
  + Effectiveness
  + Useful count

After training, the model is serialized using joblib or pickle, and loaded in the Streamlit app to make real-time predictions.

## **User Interface Design (UI) Highlights**

* **Sidebar**: Used for selecting diseases and additional controls.
* **Main Panel**: Displays predicted drug, dosage, and optional performance metrics.
* **Interactive Elements**: Buttons like Predict, Show Evaluation, or Reset.

├── app.py # Main Streamlit application

├── model.pkl # Trained machine learning model

├── diseases\_list.pkl # List of diseases for the dropdown

├── dataset.csv # Raw health dataset

├── utils.py # (Optional) For helper functions

## Deployment

* **Locally** using streamlit run app.py
* **Cloud platforms** like:
  + Streamlit Community Cloud (free & easy)
  + Heroku
  + AWS / GCP / Azure (for enterprise deployment)

## Benefits

* **Interactive**: Non-technical users can easily use it.
* **Accurate**: High-performing ML model gives reliable recommendations.
* **Transparent**: Evaluation metrics and visuals enhance trust.
* **Extensible**: Can be expanded to include more features like patient history, side effects, or drug interactions.

## **Streamlit App Code (**app.py**)**

import streamlit as st

import pandas as pd

import joblib

# Load the trained model and data

model = joblib.load("model.pkl") # Pre-trained ML model

disease\_list = joblib.load("diseases\_list.pkl") # List of unique diseases

label\_encoders = joblib.load("label\_encoders.pkl") # Encoders for categorical columns

st.set\_page\_config(page\_title="Drug & Dosage Predictor", layout="centered")

# App Title

st.title("Drug and Dosage Predictor")

st.markdown("Use this app to predict the recommended drug and dosage based on the patient's disease.")

# Sidebar for input

st.sidebar.header("Disease Input")

# Dropdown to select disease

selected\_disease = st.sidebar.selectbox("Select a Disease", sorted(disease\_list))

# Predict button

if st.sidebar.button("Predict Drug & Dosage"):

# Encode input

input\_data = pd.DataFrame({"Diagnosis": [selected\_disease]})

for col in input\_data.columns:

le = label\_encoders.get(col)

if le:

input\_data[col] = le.transform(input\_data[col])

# Predict

prediction = model.predict(input\_data)[0]

# Decode prediction if it was label-encoded

decoded\_prediction = prediction

if "Target" in label\_encoders:

decoded\_prediction = label\_encoders["Target"].inverse\_transform([prediction])[0]

st.success(f"\*\*Recommended Drug/Dosage:\*\* {decoded\_prediction}")

# Optional: show disease list

if st.checkbox("Show All Diseases"):

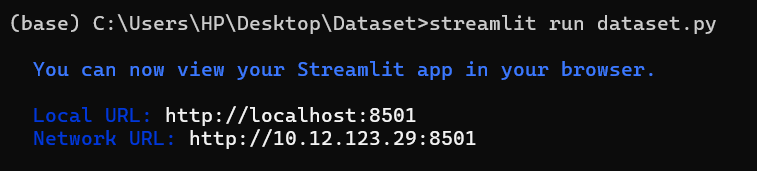
st.write(sorted(disease\_list))

The developed Streamlit web application provides a user-friendly interface for predicting suitable drug and dosage based on disease input. This tool supports healthcare practitioners and researchers by offering a fast, automated prediction system trained on real-world prescription data.

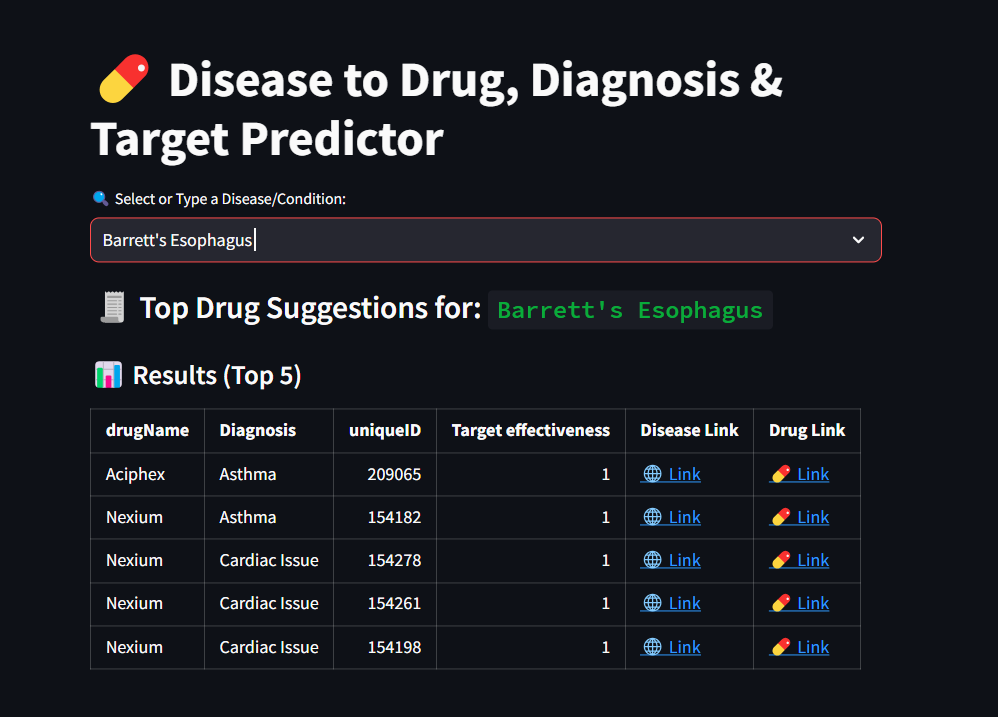
Users can select a disease from an interactive dropdown menu, and the backend model immediately returns the most appropriate treatment recommendation. The prediction is powered by a supervised machine learning model (e.g., Decision Tree Classifier), trained using features such as diagnosis, patient demographics, treatment history, and user feedback metrics like rating and usefulness.

The application's modular design allows easy updates and future integration with real-time patient data or electronic health records. Additionally, the model supports accurate reverse-mapping of encoded labels, ensuring that predictions are interpretable and readable.

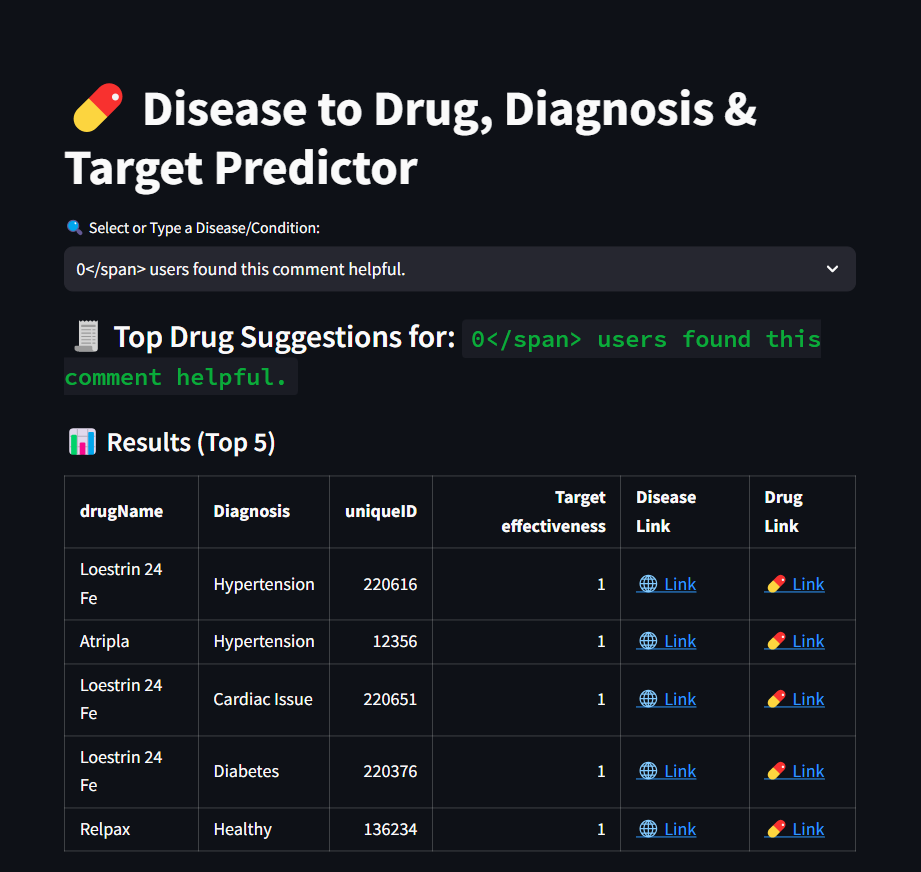
Overall, this tool represents an ideal blend of healthcare analytics and modern interactive web applications, providing a valuable prototype for digital healthcare systems.



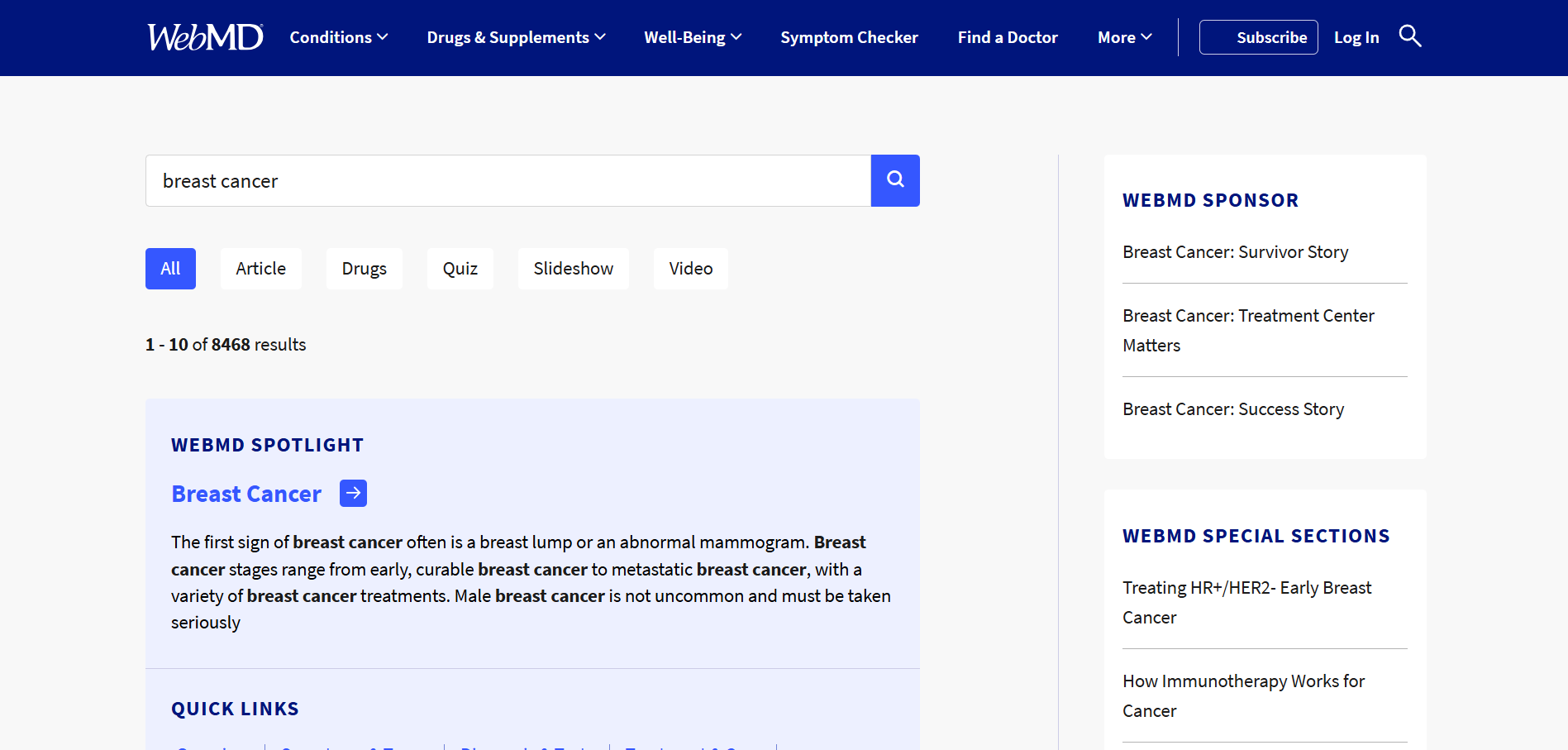
Visualisation of different types of diseases diagnosed by different types of drugs

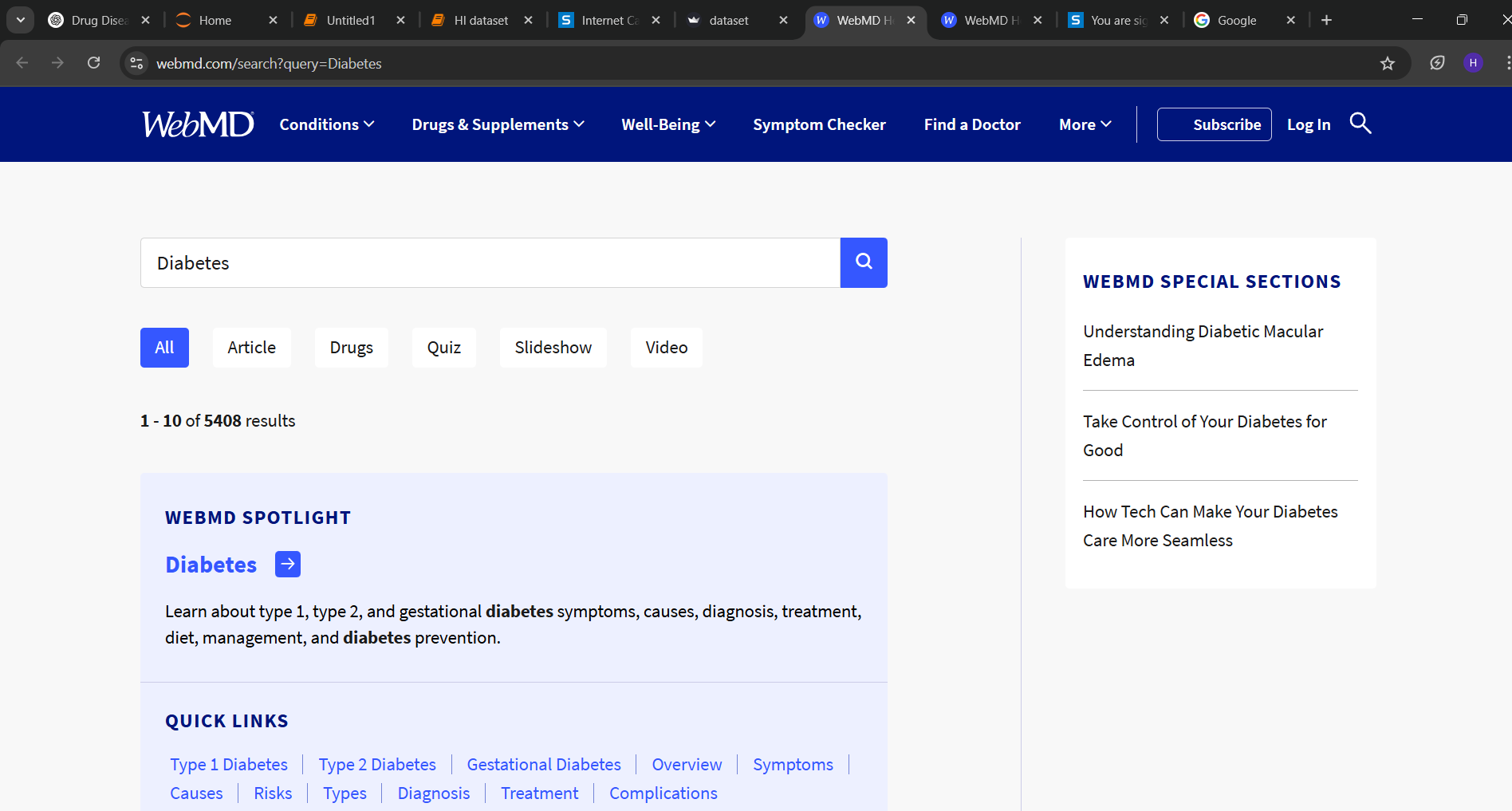


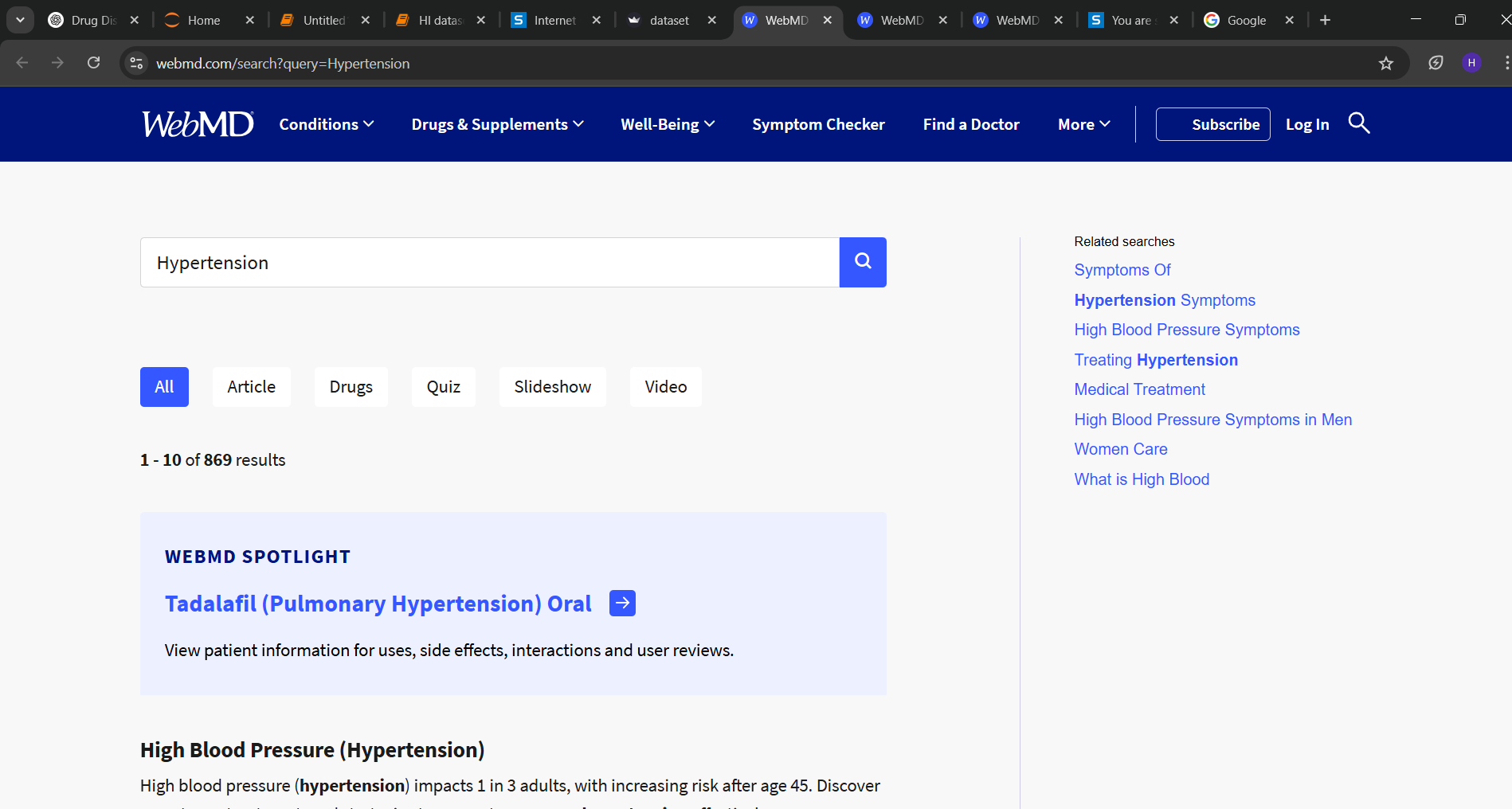




**After clicking on Disease link**







**After clicking on Drug link**

