Symptom-Based Disease Prediction and Specialty Recommendation System

Jingyi Sun, Yuli Yang, Jing Du, Xingrui Huang

Abstract

Identifying possible health conditions based on symptoms can be confusing for the general public, often leading to unnecessary anxiety or delayed care. This project introduces a machine learning-powered system that predicts potential diseases from user-reported symptoms and suggests relevant medical specialties. We trained on an augmented dataset of 247,000 anonymized patient cases, mapping 377 symptoms to 773 diseases. To manage dimensionality, we selected the top 300 most frequent symptoms as features. Among several classifiers—Decision Tree, K-Nearest Neighbors (KNN), Stochastic Gradient Descent (SGD), and Multinomial Naive Bayes (MNB)—the MNB model achieved the best performance, with an 85.55% test accuracy. To support real-world use, we developed an interactive web application using Streamlit. Users can select symptoms and receive ranked disease predictions with associated probabilities. Each prediction is mapped to the appropriate medical specialty, and users can optionally locate nearby healthcare providers via the NPPES (National Plan and Provider Enumeration System) registry API. While the tool offers meaningful insights, it is designed for informational and educational purposes only and is not a substitute for professional medical advice. Future work includes incorporating more nuanced symptom data, testing advanced models, and enabling user feedback for continuous system improvement.

1.Introduction：

1.1. Motivation and Problem Statement

In an age of readily available online information, individuals experiencing health symptoms often turn to the internet for initial guidance. However, navigating the vast amount of often conflicting or unreliable information can be overwhelming and potentially lead to incorrect self-assessments or undue anxiety. There exists a need for more structured, data-driven tools that can provide preliminary insights into potential conditions based on reported symptoms and guide users toward appropriate levels of medical care by suggesting relevant specialties. While not replacing professional diagnosis, such a tool can empower users with better information for discussing concerns with healthcare providers.

This project aims to bridge this gap by developing a machine learning system capable of predicting disease likelihood based on a user's symptoms. The goal is to provide an accessible, initial orientation rather than a definitive diagnosis.

1.2. Project Objectives

The primary objectives of this project were:

- To process and analyze a large-scale dataset linking patient symptoms to diagnosed diseases.
- To perform exploratory data analysis (EDA) to understand data characteristics and identify patterns.
- To implement and evaluate several machine learning classification models suitable for predicting diseases based on binary symptom inputs.
- To compare model performance based on relevant metrics (accuracy, efficiency) and select the optimal model.
- To develop a user-friendly web application allowing users to input symptoms and receive ranked disease predictions.
- To enhance the application by mapping predicted diseases to appropriate medical specialties.

1.3. Scope and Limitations

This project focuses on building a predictive model and an accompanying application based on the provided

https://www.kaggle.com/datasets/dhivyeshrk/diseases-and-symptoms-dataset?resource=download (`Final_Augmented_dataset_Diseases_and_Symptoms.csv`) dataset. The system predicts the likelihood of various diseases given a set of binary symptom inputs (present/absent).

Key Limitations:

- **Educational Tool:** The system is designed for informational and educational purposes only and is not a diagnostic tool. It does not replace consultation with qualified healthcare professionals.
- **Data Dependency:** Model performance is inherently tied to the quality, completeness, and potential biases of the training dataset. The binary nature of symptoms (present/absent) lacks nuance regarding severity, duration, or context.
- **Model Assumptions:** The chosen models operate under specific statistical assumptions (e.g., feature independence for Naive Bayes) that may not perfectly hold true for medical data.

2.Data Description and Problem Formulation

2.1. Dataset Overview

Our disease-symptom dataset is extensive, containing 246,945 entries with 378 columns. Of these columns, 377 represent different symptoms (binary features), and one column identifies the disease. The dataset has 773 unique diseases, with the most common diseases like cystitis, vulvodynia, and nose disorder appearing approximately 1,219 times each. This rich dataset provides a strong foundation for training accurate disease prediction models.

Structure: Each row represents a case, with the first column indicating the diagnosed disease and subsequent columns representing individual symptoms.

Size: The dataset contains 246,945 records and 378 columns.

2.2. Target Variable

The target variable for prediction is the `diseases` column.

- **Type:** Categorical (Disease Name).
- **Number of Classes:** There are 773 unique disease classes represented in the dataset.
- **Distribution:** The classes exhibit significant imbalance. The most frequent disease ("cystitis") appears 1219 times, while several diseases appear only once. This imbalance poses a challenge for model training and evaluation, requiring techniques or model choices robust to such distributions (e.g., using `class_weight='balanced'` where applicable or evaluating with weighted metrics).

2.3. Features

The input features consist of 377 distinct symptoms.

- **Type:** Binary indicators (0 or 1), representing the absence or presence of a specific symptom for a given case.
- **Nature:** The high number of symptom features results in a high-dimensional input space. The binary nature simplifies input but lacks information about symptom severity, duration, or co-occurrence context beyond simple presence.

2.4. Problem Formulation

The task is formulated as a multi-class classification problem. The goal is to train a model that takes a binary vector representing a patient's reported symptoms as input and outputs a prediction of the most likely disease from the 773 possible classes. Additionally, providing probability estimates for the top predictions adds significant value for the user.

3.Exploratory Data Analysis (EDA)

3.1. Selected EDA Findings

- **Dataset Dimensions & Types:** Confirmed dataset size (~247k x 378), the target variable (`diseases`, 773 unique values), and the binary nature of the 377 symptom features. No missing values were immediately apparent in the core data based on the initial checks.
- **Disease Distribution:** Analysis revealed a significant class imbalance, with diseases like Cystitis, Vulvodynia, and Nose disorder being the most frequent, while many

others were rare. This highlights the need for evaluation metrics beyond simple accuracy (like weighted F1-score) or models robust to imbalance.

- **Symptom Prevalence:** The frequency of individual symptoms varies greatly.

```
Top 10 most common symptoms:
sharp abdominal pain    32307
vomiting                27874
headache                24719
cough                   24296
sharp chest pain        24016
nausea                  23687
back pain               21809
shortness of breath     21346
fever                   20394
dizziness               17272
dtype: int64
```

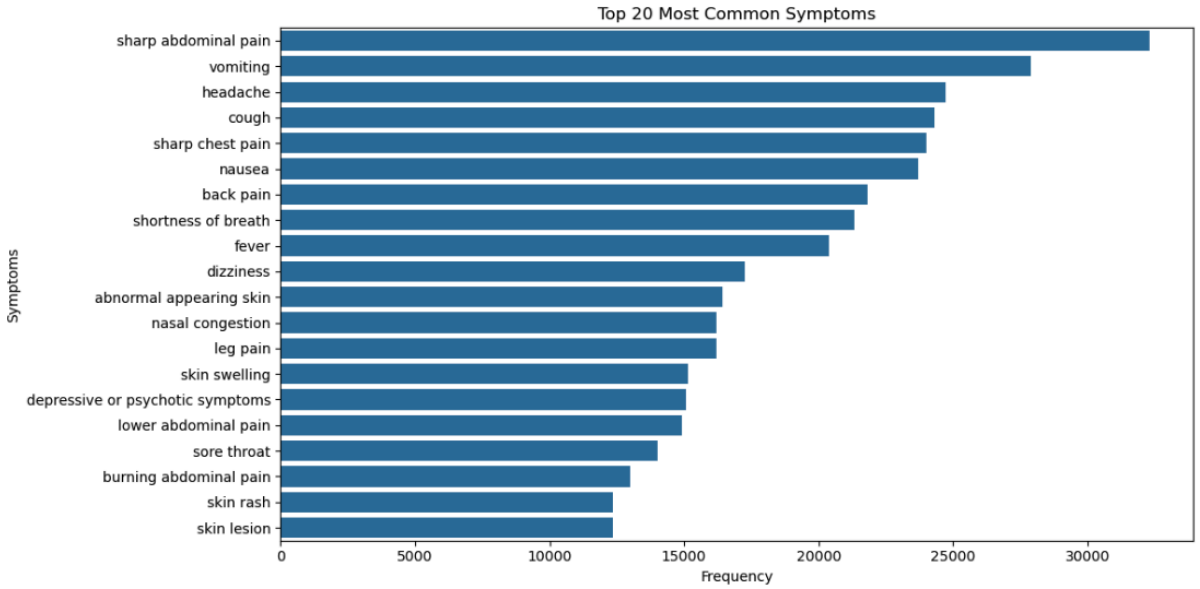

Figure 1 : Bar Chart of Top 20 Most Common Symptoms

  - As shown in Figure 1, symptoms like "sharp abdominal pain", "vomiting", "headache", "cough", and "sharp chest pain" are among the most commonly reported across all cases. This suggests these symptoms might be less specific indicators compared to rarer symptoms.
- **Symptoms per Case:** On average, each case in the dataset reports approximately 5.3 symptoms. This indicates that disease diagnosis often relies on a combination of symptoms rather than a single indicator.

- **Symptom-Disease Relationships:**

  - Analysis of top symptoms associated with specific common diseases (e.g., 'cystitis' strongly associated with 'involuntary urination', 'side pain') and top diseases associated with specific symptoms (e.g., 'abnormal appearing skin' frequently linked to 'eczema') provided insights into typical clinical presentations within the dataset.

```
Most associated symptoms for 'eczema':
itching of skin                 633
skin lesion                     619
acne or pimples                 617
skin swelling                   615
skin rash                       614
warts                           612
irregular appearing scalp       609
skin irritation                 603
allergic reaction               601
abnormal appearing skin         600
```

- A heatmap visualizing the mean presence of top symptoms across top diseases, helping to confirm expected medical relationships and potentially revealing less obvious patterns in the data.

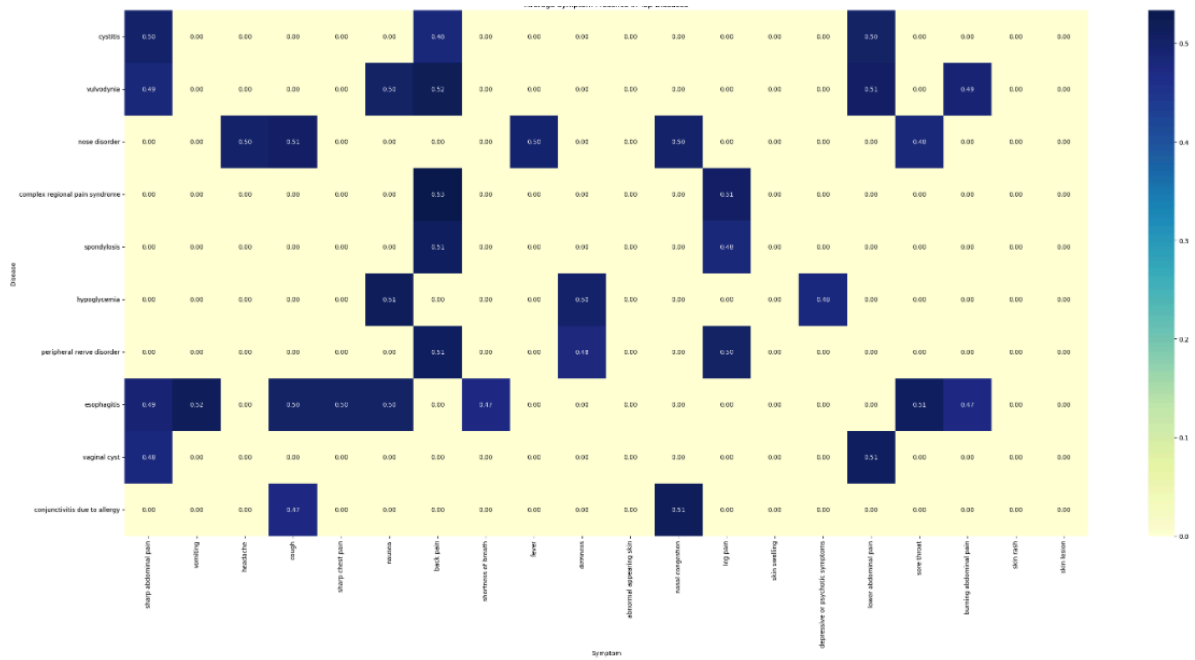

Figure 2: Symptom-Disease Heatmap Snippet

The EDA phase confirmed the high-dimensional, multi-class, and imbalanced nature of the problem, guiding the subsequent feature selection and modeling approach.

4.Methodology

4.1. Data Preprocessing

The dataset primarily consists of binary features (symptom presence/absence) and a categorical target (disease). Given this structure, minimal preprocessing was required beyond separating features and the target variable.

- **Encoding:** The features were already in a suitable binary format (0/1).
- **Missing Values:** Initial checks in the EDA notebook indicated no significant missing values requiring imputation for the features used.
- **Scaling:** Not applicable for the binary features and models like Multinomial Naive Bayes or Decision Trees used here.

4.2. Feature Selection

Due to the high dimensionality (377 symptoms), a feature selection strategy was implemented to improve model efficiency and potentially performance by focusing on the most relevant symptoms.

- **Method:** Frequency-based selection. The total count for each symptom across all records was calculated.
- **Selection Criterion:** The top 300 most frequently occurring symptoms were selected as the final feature set for model training and evaluation.
- **Rationale:** This approach assumes that more frequent symptoms are more likely to be informative for distinguishing between common diseases, while also reducing computational load and the risk of overfitting on very rare symptoms.

4.3. Model Selection Approach

Several standard classification algorithms suitable for this type of task were selected for comparative evaluation:

- **Decision Tree Classifier:** A non-linear model capable of capturing feature interactions.
- **Multinomial Naive Bayes (MNB):** A probabilistic classifier often effective for text classification and suitable for count-based or binary features, assuming feature independence.
- **K-Nearest Neighbors (KNN):** An instance-based learner that classifies based on the majority class among nearest neighbors in the feature space.
- **Stochastic Gradient Descent (SGD) Classifier:** A linear classifier (used here with 'log_loss' to approximate Logistic Regression) efficient for large datasets.

4.4. Training and Evaluation

- **Data Split:** The dataset (using the selected 300 features) was split into training (80%) and testing (20%) sets using `sklearn.model_selection.train_test_split` with `random_state=42` for reproducibility. This resulted in 197,556 training samples and 49,389 test samples.
- **Training:** Each selected model was trained on the `X_train`, `y_train` data. Hyperparameters were set based on common practices or simple tuning
- **Evaluation Metrics:** Model performance was primarily assessed using:

- **Accuracy:** Overall percentage of correct predictions on the test set (`X_test`, `y_test`).
- **Classification Report:** Provided weighted average Precision, Recall, and F1-score to account for class imbalance.
- **Training Time:** Time taken to fit the model.
- **Prediction Time:** Time taken to predict on the test set.

5.Model Comparison and Selection

5.1. Models Tested and Optimizations

The following models were trained and evaluated using the top 300 symptom features:

- **Decision Tree Classifier:**
  - Parameters: `max_depth=25`, `min_samples_split=5`, `min_samples_leaf=2`, `class_weight='balanced'`, `random_state=42`. Optimization focused on limiting depth and setting minimum sample sizes to mitigate overfitting, while using balanced class weights to address data imbalance.
- **Multinomial Naive Bayes (MNB):**
  - Parameters: `alpha=0.1`. A small Laplace smoothing parameter (`alpha`) was chosen, suitable for binary feature data.
- **K-Nearest Neighbors (KNN):**
  - Parameters: `n_neighbors=5`, `weights='distance'`, `algorithm='auto'`, `n_jobs=-1`. K=5 is a common starting point, distance weighting gives more influence to closer neighbors, and `n_jobs=-1` utilizes available CPU cores for efficiency.
- **SGD Classifier (with log_loss):**
  - Parameters: `loss='log_loss'`, `penalty='l2'`, `alpha=0.0001`, `max_iter=1000`, `tol=1e-3`, `random_state=42`, `n_jobs=-1`. Configured to perform logistic regression with L2 regularization, using sufficient iterations for convergence.

5.2. Performance Comparison

The performance of each model on the test set is summarized below.

Models sorted by accuracy:

name	Accuracy	Training Time	Prediction Time
Multinomial Naive Bayes	85.55%	167.49 sec	0.2012 sec
SGD Classifier	83.43%	89.42 sec	0.2195 sec
K-Nearest Neighbors	81.38%	0.21 sec	126.4341 sec
Decision Tree	5.87%	2.82 sec	0.1355 sec

Table 1 : Model Performance Comparison

- **Accuracy:** Multinomial Naive Bayes (MNB) achieved the highest accuracy at 85.55%. SGD Classifier followed closely at 83.43%, and KNN achieved 81.38%. The Decision Tree performed poorly with only 5.87% accuracy, likely due to the high dimensionality and complexity of the multi-class problem, even with parameter tuning.
- **Efficiency:**
    - *Training Time:* KNN was the fastest to "train" (as it's instance-based), followed by the Decision Tree. MNB and SGD required significantly longer training times (167s and 89s, respectively) due to iterating over the large dataset.
    - *Prediction Time:* MNB and SGD were very fast at prediction (~0.2s). KNN was notably slow (126s) due to the need to compare test instances against the large training set. The Decision Tree was also fast at prediction (0.14s).
- **Weighted Metrics:** MNB also demonstrated strong performance in weighted Precision (0.86), Recall (0.86), and F1-Score (0.85), indicating robust performance across the imbalanced classes.

5.3. Final Model Selection

- **Model Chosen: Multinomial Naive Bayes (MNB)**
- **Rationale:**
    1. **Suitability:** MNB is often well-suited for high-dimensional, sparse datasets with discrete/binary features, aligning with the nature of symptom data.
    2. **Highest Accuracy:** Among all tested models, MNB achieved the highest test accuracy at **85.55%**, outperforming SGD, Decision Tree, and KNN.
    3. **Efficient for Sparse High-Dimensional Data:** Given that our dataset represents binary symptom presence across 300 features, MNB's probabilistic nature is a strong match for such sparse, high-dimensional input spaces.
    4. **Robustness to Imbalance:** Despite the extreme class imbalance (some diseases with hundreds of examples, others with only a few), MNB maintained high **micro-averaged ROC AUC = 1.00**, as shown in Figure 1. This suggests it performs well across all classes when aggregated.
    5. **Low Inference Latency:** In contrast to KNN, which was computationally intensive at prediction time, MNB provides **instantaneous predictions**, supporting real-time web deployment.

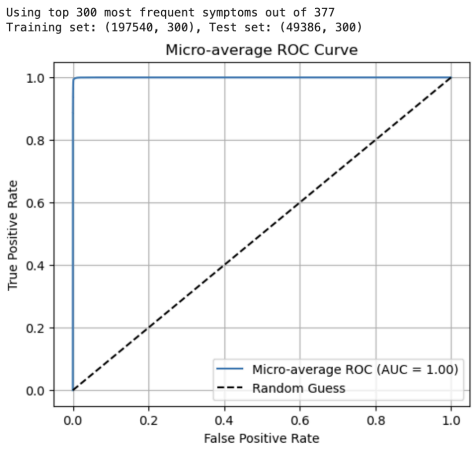

Figure 3: Micro-averaged ROC Curve showing near-perfect class-level separability for the test set

To further validate performance beyond a single metric, we generated a **confusion matrix** (Figure 4) for the top-20 most supported classes in the test set. The model demonstrates strong class separability for high-frequency diseases such as **arthritis of the hip**, **knee ligament tear**, and **epilepsy**, where most predictions fall on the diagonal.

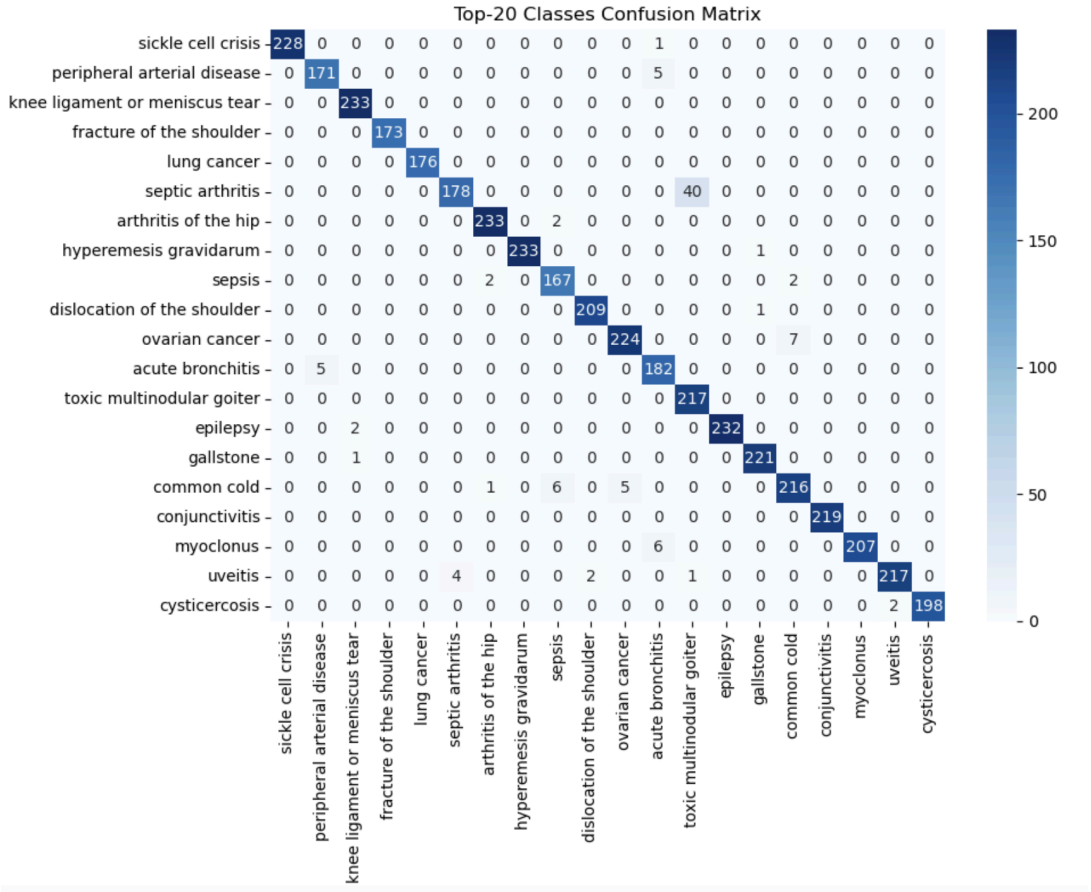

Figure 4: Top-20 Confusion Matrix reveals strong diagonal dominance in major disease categories

Additionally, we analyzed **precision, recall, and F1-score** for the most common conditions (Figure 5). Most top-20 classes achieve balanced precision and recall, often exceeding **0.85 F1**, indicating the model's consistent reliability across frequently reported diseases.

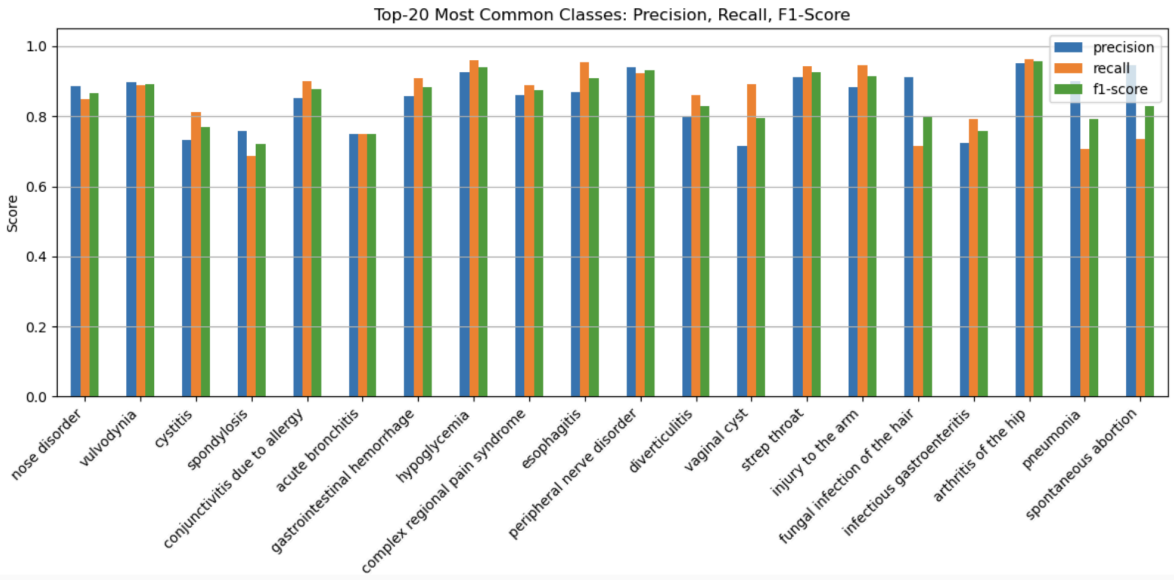

Figure 5: Performance metrics (precision, recall, F1) for top-20 most common diseases

Although SGD provided competitive accuracy and speed, MNB edged it out slightly on the primary metric (accuracy). KNN's prohibitive prediction time made it unsuitable for the application despite reasonable accuracy. The Decision Tree's performance was insufficient. Overall, MNB not only achieves high performance but also aligns with the application's requirements for **scalability, interpretability, and responsiveness**, making it the best choice for initial deployment.

5.4. Model Deployment:

The final model was saved as

`disease_prediction_multinomial_naive_bayes.joblib`, with the corresponding feature vocabulary saved in `model_features.csv`. These files are integrated into a Streamlit web app where users can select symptoms and receive probabilistic disease predictions.

6.Application Design and Implementation

Based on the selected Multinomial Naive Bayes model, a web application was developed to provide users with an interactive interface for disease prediction and specialty guidance.

6.1. System Overview

The application is built using Python and the Streamlit framework, creating a simple yet effective web-based front-end. The backend logic handles user input, model prediction, specialty mapping, and external API calls for provider lookup.

6.2. User Interface (UI) and Workflow

1. **Symptom Selection:** Users are presented with a multi-select dropdown widget populated with the 300 symptoms the model was trained on (`model_features.csv`). They can search and select multiple relevant symptoms.
2. **Prediction Trigger:** Upon clicking the "Predict Disease" button, the selected symptoms are processed.
3. **Results Display:** The application displays:
    - The single most likely predicted disease.
    - A list of the top 3 potential diseases ranked by the model's predicted probability (confidence score).
    - The recommended medical specialty associated with the top predicted disease.
4. **Provider Search:** If a specialty is recommended, a "Find Providers" button appears. Clicking this reveals input fields for location (City, State, ZIP) and triggers a search for relevant healthcare providers.
5. **Provider Display:** Search results from the NPPES API are displayed, including provider names, addresses, contact information (if available), and taxonomy descriptions.

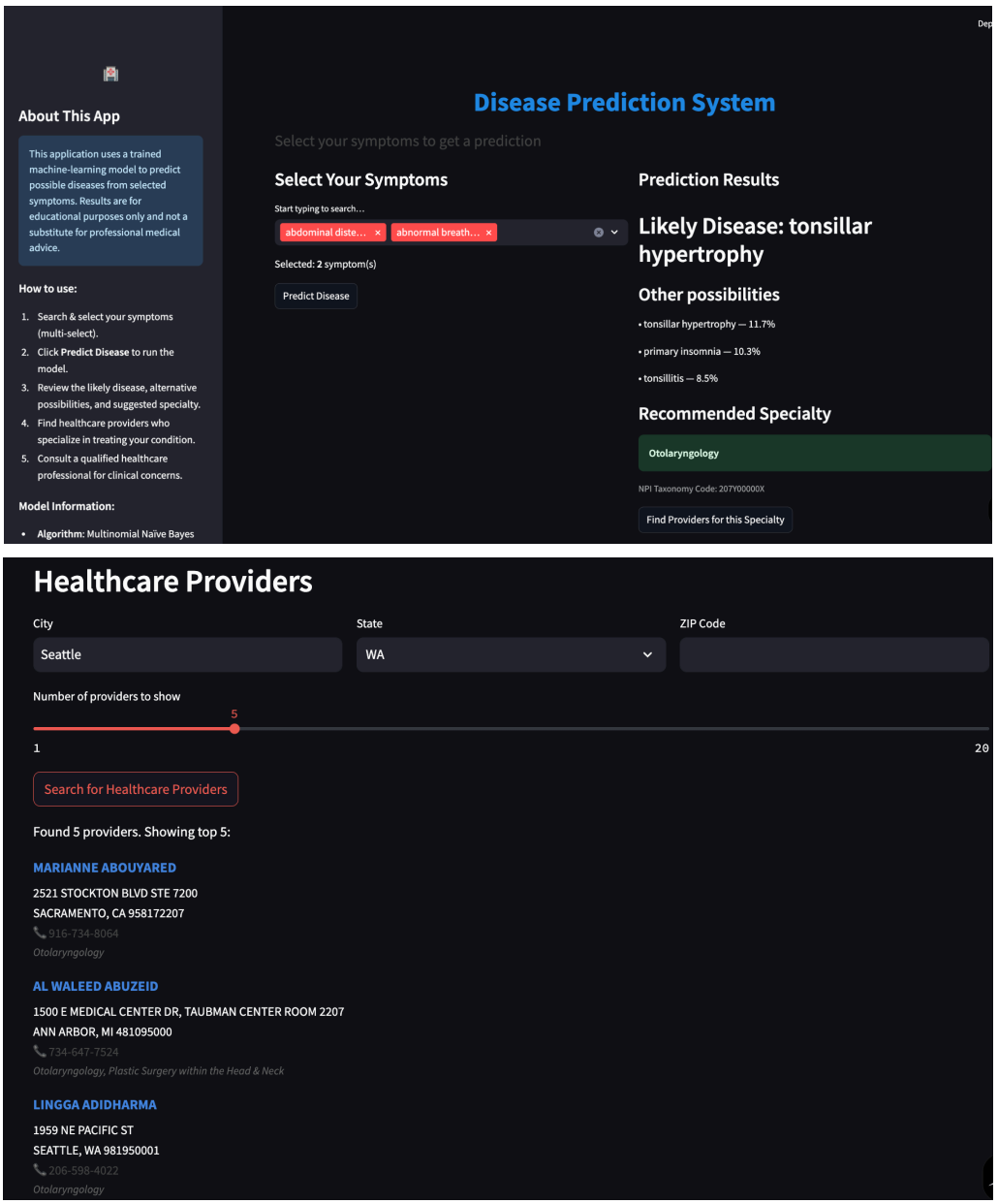

Figure 3 : Screenshot of the Streamlit Application Interface

6.3. Key Implementation Details

- **Model Integration:** The pre-trained MNB model (`disease_prediction_multinomial_naive_bayes.joblib`) is loaded at application startup using `joblib.load()`.
- **Input Encoding:** A helper function (`one_hot` in `app.py`) converts the list of selected symptoms into a binary feature vector (DataFrame row) matching the model's input requirements (300 features), setting selected symptoms to 1 and others to 0.
- **Prediction & Probabilities:** The loaded model's `.predict()` method is used for the top prediction, and `.predict_proba()` provides the confidence scores for ranking alternatives.

- **Specialty Mapping:** The `disease_specialty_mapping.tsv` file is loaded into a dictionary during startup. After prediction, the application looks up the predicted disease (case-insensitive matching might be needed) to retrieve the corresponding Specialty, Classification, and Taxonomy Code.
- **NPPES API Integration:** The `Workspace_providers_by_specialty` function constructs a query URL using the recommended specialty (and optional location parameters) and uses the `requests` library to call the NPPES NPI Registry API. The JSON response is parsed by the `render_provider_results` function for display.
- **Caching:** Streamlit's `@st.cache_resource` is used for loading the model, features, and mapping file efficiently, preventing reloads on every interaction.

6.4. Technology Stack The core technologies and libraries used include:

- **Python:** Programming Language
- **Streamlit:** Web Application Framework
- **Scikit-learn:** Machine Learning (Model training, evaluation, MNB implementation)
- **Pandas:** Data manipulation and loading CSV/TSV files
- **Joblib:** Saving and loading the trained model
- **Requests:** Making HTTP requests to the NPPES API

7.Conclusion and Future Work

7.1. Conclusion

This project successfully developed a Symptom-Based Disease Prediction and Specialty Recommendation System. By leveraging a large dataset and comparing several machine learning models, we identified Multinomial Naive Bayes as the most effective classifier for this task, achieving a promising accuracy of 85.55% on the test set.

The implemented Streamlit web application provides an intuitive interface for users to input symptoms and receive ranked predictions of potential diseases along with associated confidence levels. Crucially, the system bridges the gap between potential conditions and appropriate care by recommending relevant medical specialties based on the predictions. The optional integration with the NPPES API further enhances practical value by enabling users to search for local providers within that specialty.

The system demonstrates the potential of machine learning as an accessible, preliminary tool for health information navigation. However, it is crucial to reiterate that this application is intended for educational and informational purposes only and should never replace consultation with qualified healthcare professionals.

7.2. Limitations

Several limitations should be acknowledged:

- **Data Quality and Scope:** The model's predictions are entirely dependent on the underlying dataset

(`Final_Augmented_dataset_Diseases_and_Symptoms.csv`).Potential biases, inaccuracies, or outdated information in this dataset directly impact performance. The binary representation of symptoms lacks crucial clinical context like severity, duration, onset, and patient history. The data augmentation process (if any) is not detailed.

- **Model Limitations:** MNB assumes feature independence, which may not hold true for symptoms. While achieving good accuracy, 85.6% is not perfect, and misclassifications will occur, especially for rarer diseases or those with overlapping symptoms.
- **Input Method:** Relying on users selecting from a predefined list may miss nuances or unlisted symptoms.
- **Diagnostic Capability:** The system predicts likelihood based on patterns in the data; it **cannot** perform medical diagnosis. The "confidence" scores are model probabilities, not clinical certainties.
- **Provider Matching:** The NPPES API search is based on specialty and location but doesn't account for insurance, availability, or specific sub-specializations.

7.3. Future Work

Several avenues exist for future improvement and expansion:

- Several promising directions exist to further enhance the system's usability, accuracy, and clinical relevance:
- **Natural Language Symptom Input:**
  Enable users to describe their symptoms in free text. Leveraging advanced natural language processing (NLP) techniques (e.g., named entity recognition, keyword expansion, synonym resolution), the system can automatically extract relevant symptoms and contextual features for downstream prediction. This will bridge the gap between layperson language and clinical terminology.
- **Interactive Symptom-to-Diagnosis Pipeline:**
  Upon user input, the model should not only classify the likely disease but also visualize the decision path using model interpretability tools (e.g., SHAP or LIME), offering transparency into why certain symptoms led to specific diagnoses.
- **Enhanced Symptom Knowledge Base:**
  Incorporate structured medical ontologies (e.g., SNOMED CT) and expand the symptom list to include severity, duration, and onset time. Synonym matching and fuzzy symptom extraction should reduce the reliance on exact phrasing.
- **Physician Mapping and Geolocation Recommendation:**
  Integrate provider databases with geolocation APIs (e.g., Google Maps, Leaflet) to display nearby specialists visually on an interactive map. The map can include filters for insurance compatibility, availability, and verified reviews (where legally and ethically feasible).
- **Personalization and Demographics-Aware Prediction:**
  Account for patient-specific features such as age, gender, medical history (where ethically permissible), and comorbidities to improve differential diagnosis accuracy.
- **Robust Modeling and Validation:**
  Explore alternative classification methods like Random Forests, XGBoost, LightGBM, or transformer-based models (e.g., BERT variants fine-tuned for clinical text).

Implement cross-validation with stratification to ensure performance consistency across rare classes.

- **User Feedback and Continuous Learning:**
  Allow users (e.g., clinicians) to provide feedback on prediction correctness, enabling active learning and periodic model refinement.
- **Longitudinal Symptom Tracking:**
  For users returning over time, build temporal models to monitor symptom progression and improve diagnosis by learning from historical patterns.

References

References

- National Plan and Provider Enumeration System (NPPES) NPI Registry API.
- Kaggle Medical Specialty Classification Dataset
- Doctor Specialist Recommendation System Dataset
- Streamlit Documentation: https://docs.streamlit.io/
- Scikit-learn Documentation: https://scikit-learn.org/stable/
- Pandas Documentation: https://pandas.pydata.org/docs/
- Joblib Documentation: https://joblib.readthedocs.io/