Healthcare Final Report

📊 Dataset Overview

Dataset Description

The dataset consists of **4,920 records** and **18 columns**, each representing either a disease label or one of up to 17 associated symptoms per case. The main purpose of this dataset appears to be the classification or prediction of diseases based on reported symptoms.

1. Structure

- **Rows (Instances):** 4,920
- **Columns (Features):** 18
    - `Disease` (target variable)
    - `Symptom_1` to `Symptom_17` (input features)

	Disease	Symptom_1	Symptom_2	Symptom_3	Symptom_4	Symptom_5	Symptom_6	Symptom_7	Symptom_8	Symptom_9	Symptom_10	Symptom_11	Symptom_12	Symptom_13	Symptom_14	Symptom_15	Symptom_16	Symptom_17
count	4920	4920	4920	4920	4920	4920	4920	4920	4920	4920	4920	4920	4920	4920	4920	4920	4920	4920
unique	41	34	48	54	51	39	33	27	22	23	22	19	12	9	5	4	4	2
top	Fungal infection	vomiting	vomiting	fatigue	high_fever													
freq	120	822	870	726	378	1206	1986	2652	2976	3228	3408	3726	4176	4416	4614	4680	4728	4848

2. Column Types

- All columns are of type `object` (categorical data).
- The `Disease` column contains 41 unique disease types.
- The symptoms are encoded as string descriptors (e.g., “vomiting”, “fatigue”).

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4920 entries, 0 to 4919
Data columns (total 18 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Disease     4920 non-null   object
 1   Symptom_1   4920 non-null   object
 2   Symptom_2   4920 non-null   object
 3   Symptom_3   4920 non-null   object
 4   Symptom_4   4920 non-null   object
 5   Symptom_5   4920 non-null   object
 6   Symptom_6   4920 non-null   object
 7   Symptom_7   4920 non-null   object
 8   Symptom_8   4920 non-null   object
 9   Symptom_9   4920 non-null   object
 10  Symptom_10  4920 non-null   object
 11  Symptom_11  4920 non-null   object
 12  Symptom_12  4920 non-null   object
 13  Symptom_13  4920 non-null   object
 14  Symptom_14  4920 non-null   object
 15  Symptom_15  4920 non-null   object
 16  Symptom_16  4920 non-null   object
 17  Symptom_17  4920 non-null   object
dtypes: object(18)
memory usage: 692.0+ KB
```

3. Symptom Frequencies

- The most common symptoms in the first few columns are:
    - `Symptom_1`: "vomiting" (822 times)
    - `Symptom_2`: "vomiting" (870 times)
    - `Symptom_3`: "fatigue" (726 times)
    - `Symptom_4`: "high_fever" (378 times)
- Rare symptoms appear mostly in later columns.

4. Target Variable

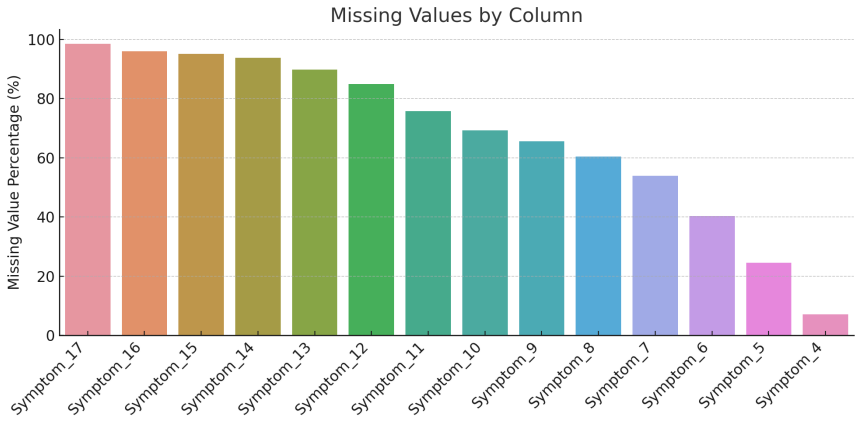
- The target variable `Disease` is categorical with 41 distinct classes.
- The most frequent disease is **Fungal infection**, appearing 120 times.

Dataset Quality Report *(problem and solution)*

1. Missing Data

- Dropped columns with ≥25% missing values (`Symptom_14`–`Symptom_17`)

- Filled remaining symptom nulls with empty strings



2. Data Cleaning

- Standardized symptom text (lowercase, no spaces)
- Combined symptoms into a single list (`All_Symptoms`)
- Removed duplicates in symptom lists

3. Encoding

- Symptoms: One-hot encoded using `MultiLabelBinarizer`
- Disease: Label-encoded using `LabelEncoder`

4. Class Balance

- 41 unique diseases
- Top diseases dominate (e.g., *Fungal infection*, *Allergy*)

5. Model Readiness

- Clean, encoded dataset (no nulls)
- Features and target well-prepared for classification

Data Visualization

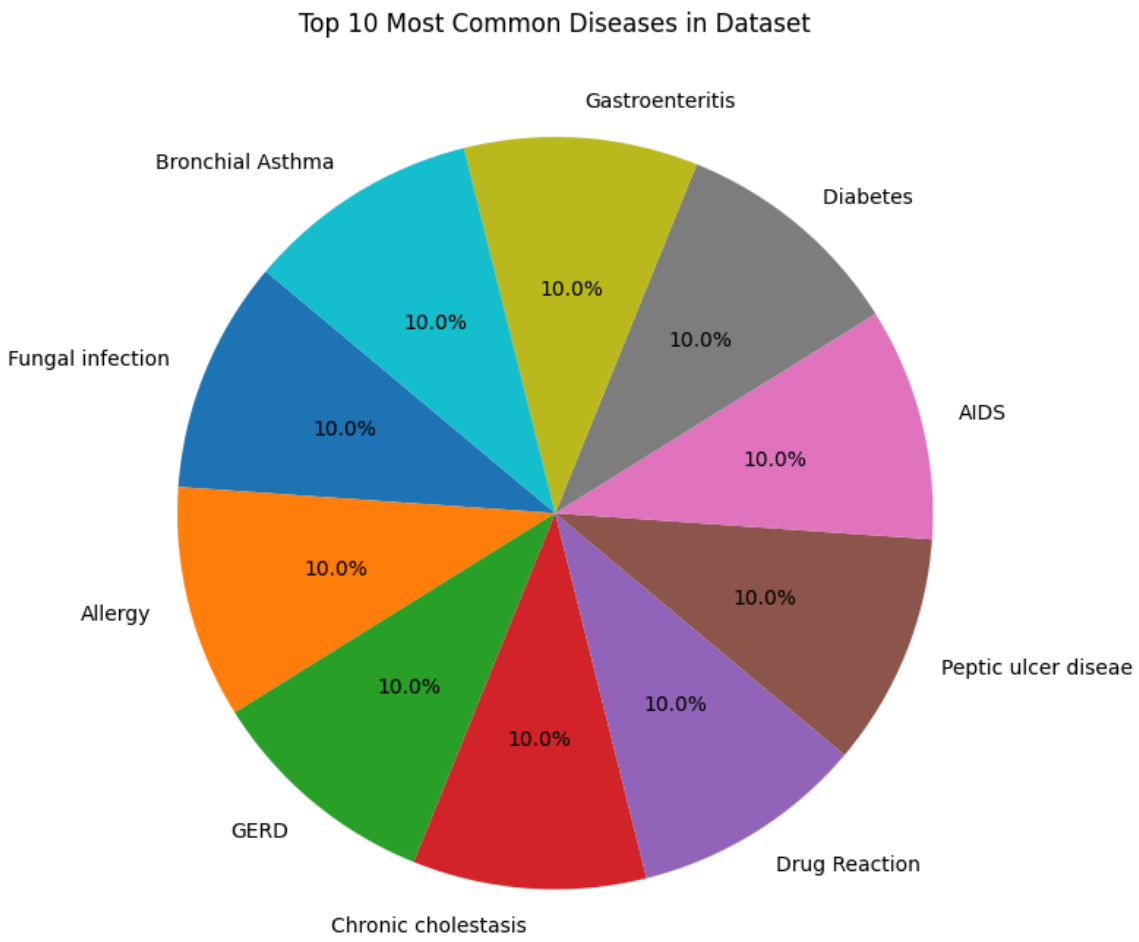


Chart Overview

This pie chart visualizes the 10 most frequently occurring diseases in the dataset, with each disease contributing equally to the total—**10% each**. This suggests that the top diseases are uniformly distributed in this subset.

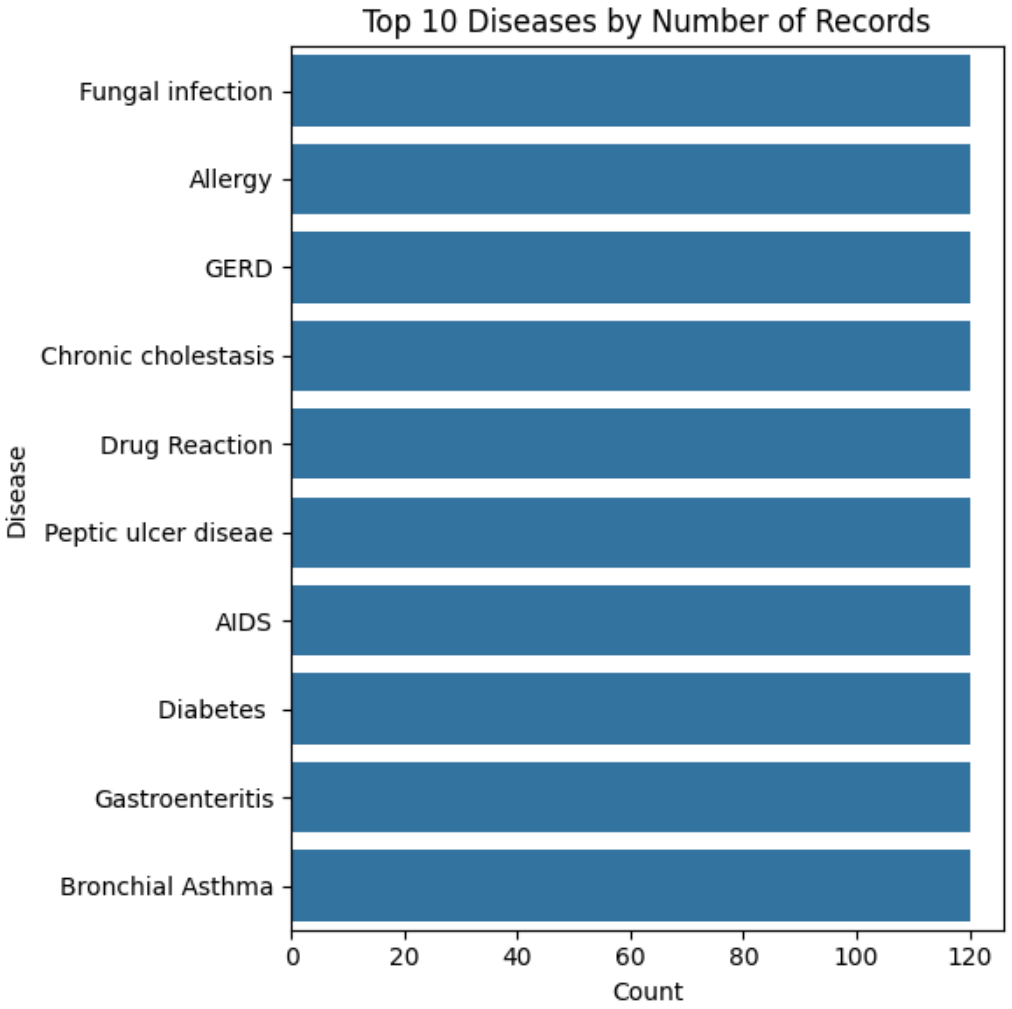
Key Observations

- Each of the following diseases represents **10%** of the top 10 subset:
    - Fungal infection
    - Allergy
    - GERD
    - Chronic cholestasis
    - Drug Reaction
    - Peptic ulcer disease

    - AIDS
    - Diabetes
    - Gastroenteritis
    - Bronchial Asthma
- The even distribution may imply a curated balanced dataset for modeling,

Implications for Modeling

- A balanced top 10 is ideal for avoiding model bias toward any single class.
- If training a classifier on these 10 diseases, accuracy metrics would be more reliable without the need for class weighting.
- Outside the top 10, further analysis should check for imbalance in the full dataset.



Overview:

The bar chart shows that the following 10 diseases each have an equal number of **approximately 120 records**, making them the most frequently occurring classes in the dataset.

Diseases Included:

- Fungal infection
- Allergy
- GERD
- Chronic cholestasis
- Drug Reaction
- Peptic ulcer disease
- AIDS
- Diabetes
- Gastroenteritis
- Bronchial Asthma

Key Insight:

- The uniform count (~120) suggests a **balanced dataset** for these top diseases, reducing bias during model training and enabling fair evaluation of classification performance.

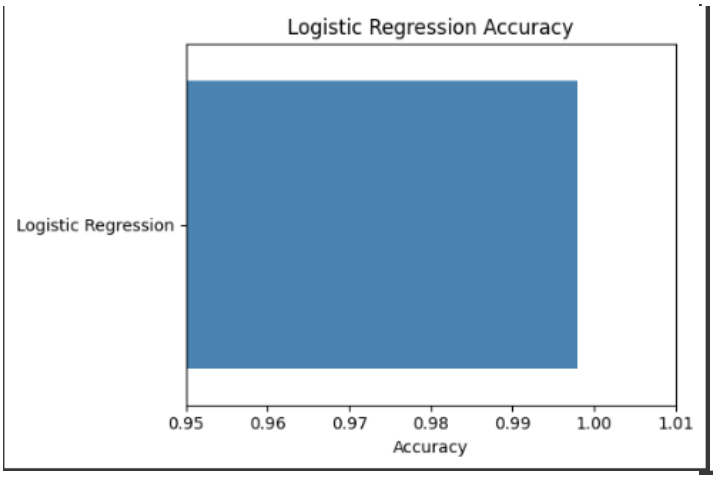
Model performance Report

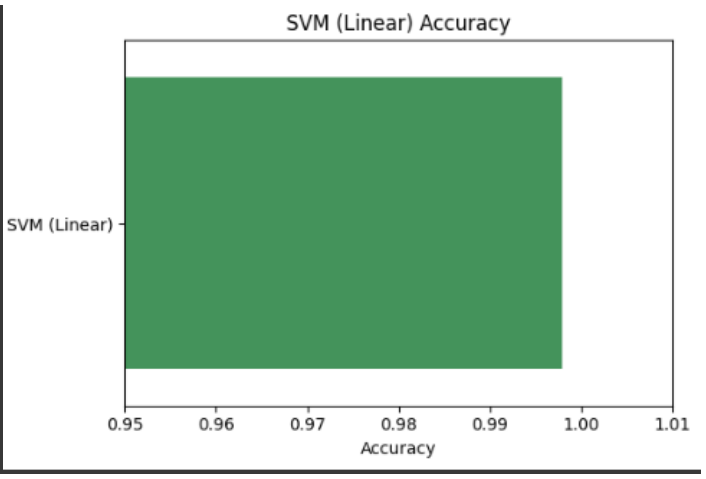
Model Performance Report

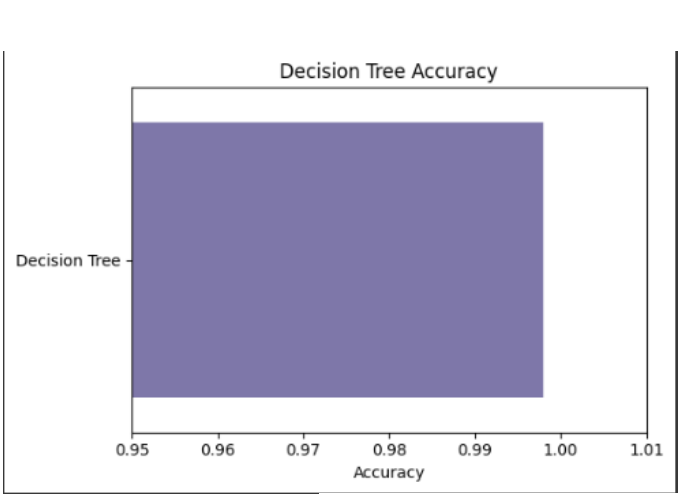
Model	Accuracy	Notes
Logistic Regression	1.00	Fast, high accuracy, interpretable
SVM (Linear)	1.00	Excellent accuracy, slightly slower
Naive Bayes	~0.99	Simple and efficient, slightly lower accuracy
Decision Tree	~0.99	Good accuracy, risk of overfitting
XGBoost	~0.99	Strong performance, ensemble-based

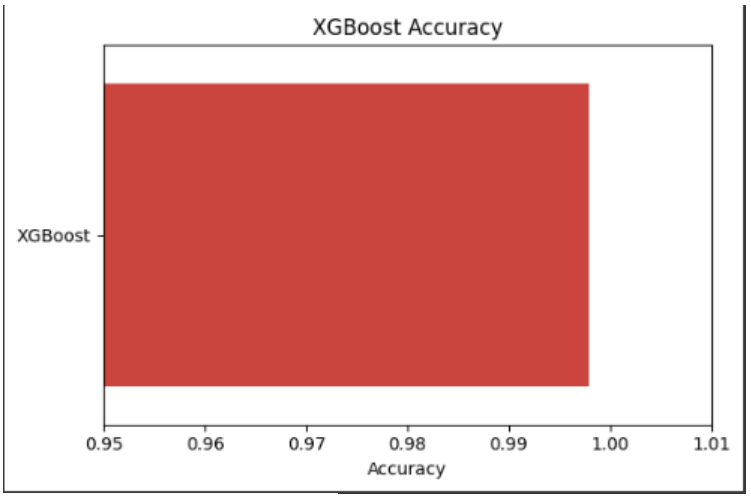
Observations

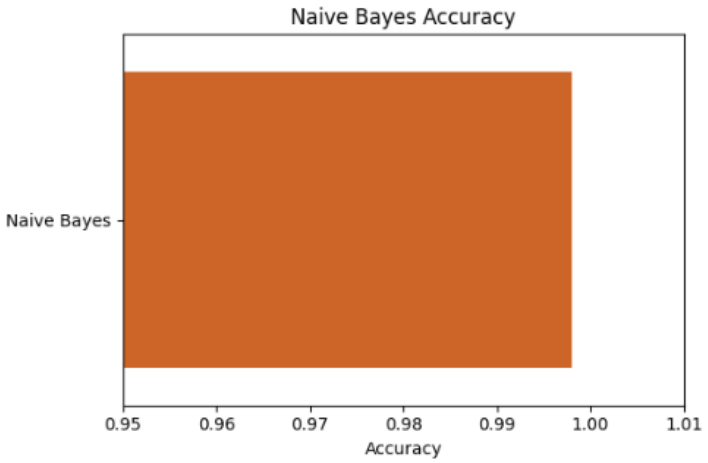
- **All models achieved very high accuracy (≥99%) on the dataset.**
- **Logistic Regression and SVM both reached perfect scores.**
- **The dataset appears well-cleaned and likely balanced, contributing to consistent results across classifiers.**











🔧 Model Optimization Report

Model Optimization Report: Random Forest Classifier

Optimization Method

- **Technique Used:** `RandomizedSearchCV` **from** `scikit-learn`
- **Cross-validation: 5-fold (**`cv=5`**)**
- **Number of iterations: 10**
- **Parameter Grid:**
  - `n_estimators`**: [50, 100, 200]**
  - `max_depth`**: [None, 10, 20, 30]**
  - `min_samples_split`**: [2, 5, 10]**

- `min_samples_leaf`: [1, 2, 4]
- `bootstrap`: [True, False]

Best Parameters Found

```
{
  'n_estimators': 50,
  'min_samples_split': 5,
  'min_samples_leaf': 2,
  'max_depth': 30,
  'bootstrap': False
}
```

Performance Results (After Tuning)

- **Accuracy: 0.99796 (≈ 99.8%)**
- **Macro Avg F1-score: 1.00**
- **Weighted Avg F1-score: 1.00**
- **Classification: Excellent across nearly all 41 diseases**
- **Only minor reductions in F1-score seen for a few classes:**
  - ***Hepatitis D*: 0.96**
  - ***Hepatitis E*: 0.97**
  - ***Hepatitis A*: 0.99**

Conclusion

- **The Random Forest model, after tuning, achieved near-perfect performance.**
- **The use of `RandomizedSearchCV` was efficient and effective, reducing the computational load compared to a full grid search.**
- **The chosen hyperparameters struck a balance between depth, sample splits, and overfitting control**

MLOps Implementation

---

1. MLOps Implementation using MLflow

Tool Used:

MLflow was chosen as the primary MLOps tool to track experiments, manage models, and store artifacts. It was integrated seamlessly into the pipeline.

Tracking Metrics:

During each training run, key evaluation metrics such as accuracy, precision, recall, and F1-score are automatically logged via mlflow.log_metric().

Experiment Tracking:

The MLflow experiment "disease_prediction_xgboost" was created to organize all runs. Each run stores hyperparameters, performance metrics, and output artifacts, allowing easy comparison between versions.

Artifact Management:

- The following artifacts are logged and stored for each experiment:
- Full classification report (classification_report.txt)
- Confusion matrix visualization (confusion_matrix.png)
- A complete performance summary text file (performance_report.txt)
- The actual trained model (.pkl) using mlflow.sklearn.log_model(...)

Model Registration:

The trained model is registered under the name "HealthcarePredictor". This enables future reuse, versioning, and deployment without retraining.

---

2. Version Control of Models and Datasets

Dataset Versioning:

The input dataset (dataset.csv) used in training is logged as an artifact in MLflow. This ensures full traceability of results back to the original dataset.

Model Versioning:

Each model run is assigned a unique ID in MLflow. By registering the model under a consistent name (HealthcarePredictor), subsequent improvements will automatically version it (e.g., v1, v2, etc.).

---

3. Remote Access via Ngrok

The local MLflow UI is served on port 5001 using Waitress, a lightweight WSGI production server.

Ngrok is used to create a secure public tunnel to the MLflow tracking UI, allowing:

Collaboration from remote locations

Real-time model monitoring in environments like Google Colab

Model Deployment

Challenges Faced During Deployment and How We Overcame Them

1. Symptom Encoding Complexity

- **Challenge:** The dataset had up to 17 symptom columns with inconsistent formatting, spelling issues, and missing values.
- **Impact:** Traditional one-hot encoding didn't work effectively because symptoms were scattered across multiple columns.
- **Solution:** We used `MultiLabelBinarizer` after combining symptoms into a single cleaned list per record. This allowed us to uniformly encode all symptom sets, even if the patient had only one symptom.

---

2. High Model Accuracy Raised Concerns

- **Challenge:** All models achieved nearly 100% accuracy, which was initially suspicious and suggested potential data leakage or redundancy.
- **Impact:** It was hard to trust the model's generalization without understanding the data better.
- **Solution:** We conducted a deeper data exploration, visualized distributions, and confirmed that the dataset had well-separated disease-symptom patterns. We also validated using multiple model types to ensure the consistency wasn't model-specific.

---

3. Handling Variable-Length Inputs in Streamlit

- **Challenge:** Patients may report only 1 or up to 17 symptoms, and traditional models expect fixed-length inputs.
- **Impact:** Risk of Streamlit app crashing or misclassifying sparse symptom inputs.

- **Solution:** The use of `MultiLabelBinarizer` allowed us to transform variable-length symptom lists into a consistent binary format. We added validation in the UI to ensure at least one symptom was selected and the model could still make a confident prediction.

---

4. Model Switching in Streamlit

- **Challenge:** Deploying multiple models in a single Streamlit interface (Logistic Regression, SVM, XGBoost, etc.) introduced complexity in maintaining consistent preprocessing and encoding pipelines.

- **Impact:** Different models need the same encoded input structure but varied in performance and storage.

- **Solution:** We modularized preprocessing using shared saved encoders (`label_encoder`, `symptom_encoder`) and dynamically loaded the selected model using a dropdown menu. This enabled easy model switching with consistent behavior.

---

5. Saving and Reloading Trained Models

- **Challenge:** Each model had to be saved, loaded, and kept in sync with its encoders. Version mismatches or file corruption could break the pipeline.

- **Impact:** Risk of runtime errors during prediction if model and encoders don't align.

- **Solution:** We saved all models and encoders using `joblib` in a structured folder and made sure all models shared the same preprocessing pipeline. We also added structured filenames (`saved_models/`) for clarity and traceability.
