Disease Prediction Project

Project Overview

The Disease Prediction Project is a machine learning-based web application designed to predict diseases from user-input symptoms, addressing the need for accessible early diagnosis in healthcare. Built for a hackathon, it leverages a dataset of 42 diseases (~120 samples per class, 4920 training samples, 42 test samples) with 132 binary symptom features.

The pipeline includes preprocessing with Variance Inflation Factor (VIF) to reduce multicollinearity, model tuning for high accuracy, and a Flask web app with a modern, user-friendly interface featuring a real-time symptom search bar. Deployed on Hugging Face Spaces, the app provides a practical tool for users to input symptoms and receive accurate disease predictions with confidence scores, demonstrating innovation in healthcare technology.

Methodology

The project follows an end-to-end ML pipeline, implemented in Python with Jupyter notebooks and scripts.

Data Preprocessing

Dataset: train.csv (4920 samples, 132 symptoms, 42 diseases) and test.csv (42 samples).

Processing (src/data/preprocess.py, Notebooks/data_preparation.ipynb):

- Encoded prognosis (disease labels) using LabelEncoder, saved as models/label_encoder.pkl
- Removed unnecessary and empty columns
- Applied VIF (threshold=5) to reduce 132 features to ~40-50, addressing multicollinearity
- **Outputs:** data/processed/train_processed.csv (4920 rows, ~40-50 features + prognosis), data/processed/test_processed.csv (42 rows, same structure)

Exploratory Data Analysis

(Notebooks/exploratory_data_analysis.ipynb):

- Visualized symptom distributions and correlations
- Confirmed balanced classes (~120 samples per disease)

Model Training

Implementation (src/models/train.py, Notebooks/model_training.ipynb):

- Evaluated four models: LogisticRegression, RandomForest, GradientBoosting, XGBoost

- Used GridSearchCV to tune hyperparameters (e.g., max_depth, n_estimators for tree-based models, C for LogisticRegression)
- Trained on train_processed.csv, validated with 5-fold cross-validation
- The best model (LogisticRegression) achieved >97% accuracy
- Saved as models/trained_model.pkl
- **Metrics:** Accuracy, precision, recall, F1-score evaluated on validation set

Web Application

Implementation (src/app.py, templates/, static/):

Built a Flask app to load trained_model.pkl and label_encoder.pkl.

Features:

- **Symptom Form:** ~40-50 checkboxes from train_processed.csv columns
- **Search Bar:** JavaScript-powered real-time symptom filtering (e.g., type "itch" to show "Itching")
- **UI:** Modern, healthcare-themed design (blue/white palette, rounded elements, responsive grid)
- **Prediction:** Converts user inputs to a DataFrame, predicts disease with confidence score (e.g., "Fungal infection, 95.23%")

Fixed sklearn warning by using DataFrame input for predictions, ensuring compatibility with LogisticRegression.

Templates:

- index.html: Symptom selection with search bar
- result.html: Displays predicted disease, confidence, and selected symptoms

Deployment

Hosted on Hugging Face Spaces with:

- **Framework:** Docker
- **Dependencies:** Managed via requirements.txt
- **Live URL:** https://huggingface.co/spaces/shubzz13/disease-prediction
- **Advantages:** Free hosting, easy deployment, integration with ML community, automatic scaling

Results

Data Preprocessing

- Reduced features from 132 to ~40-50 via VIF, maintaining predictive power while addressing multicollinearity
- **Shapes:** Training (4920 rows, ~40-50 features), Test (42 rows, same)

Model Performance

- Best model (LogisticRegression) achieved >97% accuracy on 5-fold cross-validation and test set
- **Metrics (validation):** Precision, recall, F1-score >97% across 42 classes (balanced dataset)

Web App

- Responsive UI with ~40-50 symptom checkboxes, real-time search bar, and healthcare-themed design
- Predictions match test set results, e.g., itching + skin_rash will give "Fungal infection" with ~95% confidence
- **Output:** predictions/predictions.csv: 42 test predictions with prognosis_encoded and prognosis, validated for accuracy

Discussion/Challenges

Redundant Columns

Challenge: The dataset (train.csv, test.csv) contained redundant columns, such as empty columns or features with no predictive value (e.g., constant or near-constant values), which could inflate model complexity and reduce interpretability.

Solution: Implemented a custom function in src/data/preprocess.py to identify and remove redundant columns. The function checked for columns with zero variance (constant values) or missing values exceeding 90%, dropping them before VIF analysis. This reduced the initial 132 features to a cleaner set, improving preprocessing efficiency and model performance.

Multicollinearity

Challenge: High correlation among 132 symptoms risked model instability.

Solution: Applied VIF (threshold=5) to reduce to ~40-50 features, preserving predictive power.

Sklearn Warning

Challenge: UserWarning: X does not have valid feature names during Flask predictions with LogisticRegression.

Solution: Used pandas DataFrame with feature names (symptom_cols) in app.py, ensuring consistency with training.

UI Scaling

Challenge: Displaying ~40-50 symptoms in a user-friendly way.

Solution: Implemented a responsive grid layout and JavaScript search bar for real-time filtering, enhancing usability.

Deployment

Challenge: Ensuring consistent file paths (templates/, static/, models/) and compatibility with Hugging Face Spaces environment.

Solution: Configured app.py for Hugging Face Spaces deployment, ensuring proper file structure and dependencies. Leveraged HF Spaces' seamless integration with ML models and automatic environment setup.

Impact: The app enables early diagnosis, accessible via Hugging Face Spaces, with a scalable pipeline integrated into the ML community ecosystem for future enhancements and collaboration.

Conclusion

The Disease Prediction Project delivers a robust, end-to-end ML pipeline for healthcare, from data preprocessing to web deployment on Hugging Face Spaces. By addressing redundant columns, multicollinearity with VIF, achieving >97% accuracy with tuned models, and deploying a user-friendly Flask app with a modern UI and search bar, it demonstrates technical excellence and innovation.

Challenges like redundant columns, warnings, and UI scaling were resolved through thoughtful solutions, ensuring a production-ready application accessible to the broader ML community. The deployment on Hugging Face Spaces provides additional benefits including community visibility, easy sharing, and integration with the open-source ML ecosystem.

Future improvements could include symptom categorization, confidence visualizations, or integration with medical databases. This project showcases the potential of ML in healthcare, supporting early diagnosis and accessibility while contributing to the open-source healthcare AI community.