ARTIFICIAL INTELLIGENCE & EXPERT SYSTEMS (CT-361)

INSTRUCTOR: Sir. Abdullah

COMPLEX COMPUTING PROBLEM REPORT (CCP)

GROUP MEMBERS:

FAIZAN TAHIR	CT-22074
SHAHZAIN ADIL	CT-22076
MUHAMMAD AHMED RAZA	CT-22077
NOFIL AHMED KHAN	CT-22301

DEPARTMENT: CSIT-B

SUBMISSION DATE: 18 May 2025

ACKNOWLEDGEMENT

First and foremost, **all praise and gratitude are due to Allah Almighty**, whose blessings and guidance enabled us to successfully complete this project. Without His mercy, this achievement would not have been possible.

We would also like to express our sincere appreciation to our **team members** for their dedication, collaboration, and consistent efforts throughout the development of this project. Your support and teamwork made this journey both productive and enjoyable.

Special thanks to our **respected instructor, Sir. Abdullah**, for his valuable guidance, constructive feedback, and continuous encouragement which helped shape this project into its final form.

We are also grateful for the assistance provided by advanced AI systems, particularly **ChatGPT and DeepSeek**, whose support in structuring content, debugging code, and refining our ideas played a significant role in enhancing the quality and professionalism of this work.

Finally, we extend our heartfelt thanks to everyone who contributed directly or indirectly to the successful completion of the **College Doctor (Disease Detection System)** project.

TABLE OF CONTENTS:

Introduction

The Disease Detection System is an AI-based web application that accepts symptoms from users and predicts potential diseases using three different machine learning models: Decision Tree, Random Forest, and Naive Bayes. Additionally, it provides relevant doctor suggestions by scraping search engine results.

Terminology and Description

a. Symptoms

Input features selected by the user. They represent signs of disease that the models use for diagnosis.

b. Disease Mapping (nmap)

A dictionary mapping disease names to unique numeric indices. This is essential for machine learning models, which work with numerical labels.

c. Models Used

- **Decision Tree Classifier**: A model that splits data into branches to arrive at decisions based on feature values.
- **Random Forest Classifier**: An ensemble of decision trees that increases accuracy and reduces overfitting.
- **Naive Bayes Classifier**: A probabilistic model based on Bayes' theorem, assuming independence among features.

d. train_models()

This function trains the three models using a dataset containing symptom vectors and associated diseases. Accuracy is printed to validate model performance.

e. get_doctors_scrape()

A utility function that uses web scraping to find top specialist doctors for a predicted disease by querying a search engine.

f. index() Route

Handles the front-end interaction. It:

- Receives symptom input from users.
- Convert symptoms into feature vectors.
- Use trained models to predict diseases.
- Calculates prediction confidence.
- Scrapes and displays doctor suggestions.

Code with Justification

- ```
  # Mapping diseases to unique IDs
  nmap = { ... }
  disease = sorted(nmap, key=lambda x: nmap[x])
  ```

  Justification: Machine learning models require numerical output labels. Sorting maintains consistent disease order.

- ```
  dt_model = rf_model = nb_model = None
  ```

  Justification: Placeholders for the models so they can be accessed globally.

- ```
  def train_models():
   ...
  dt_model = DecisionTreeClassifier().fit(X, y)
  rf_model = RandomForestClassifier().fit(X, y)
  nb_model = GaussianNB().fit(X, y)
  ```

  Justification: Trains all three classifiers on the same dataset. Using multiple algorithms allows comparison and robustness.

- ```
  def get_doctors_scrape(disease_name, num_results=5):
  ...
  ```

  Justification*:* Dynamically fetching doctor suggestions for real-time relevance using simple search queries.

- ```
  @app.route('/', methods=['GET', 'POST'])
  def index():
   ...
  ```
  Justification: Main route handling form submission, predictions, and rendering results. It enables complete interaction with the AI system.

Justification for Algorithm Use

- **Decision Tree**: Provides an easily interpretable structure. Good for understanding feature importance.
- **Random Forest**: Improves prediction accuracy and prevents overfitting compared to a single tree.
- **Naive Bayes**: Fast, efficient, and effective for categorical input like symptoms.

Each model contributes uniquely, allowing a broader perspective and user trust when results from multiple classifiers converge.

Technical Overview

- **Frontend**: HTML form to collect user input.
- **Backend**: Flask application handling routing and logic.
- **Machine Learning**: Trained using scikit-learn on symptom-disease datasets.
- **Web Scraping**: googlesearch module fetches top specialist links.

Dataset Source and Preprocessing

- The dataset used includes symptoms and their corresponding diagnosed diseases.
- Preprocessing involved converting symptoms into binary feature vectors (0 or 1) indicating presence/absence.
- Label encoding was used for disease classes.
  ```

- Data was split into training and testing for model validation.

Benefits of the System

- **Quick Diagnosis**: Provides fast predictions based on symptoms.
- **Accessibility**: Easy to use through a web interface.
- **Multiple Opinions**: Uses three algorithms for better reliability.
- **Actionable Output**: Suggests real-world doctors based on disease prediction.

Future Enhancements

- **Voice Input**: Add support for voice-based symptom entry.
- **Severity Detection**: Suggest emergency services if severe conditions are detected.
- **Doctor Ratings**: Integrate ratings/reviews from Google or health platforms.
- **Mobile App**: Develop a mobile version for wider accessibility.
- **Multilingual Support**: Support multiple languages for broader usage.

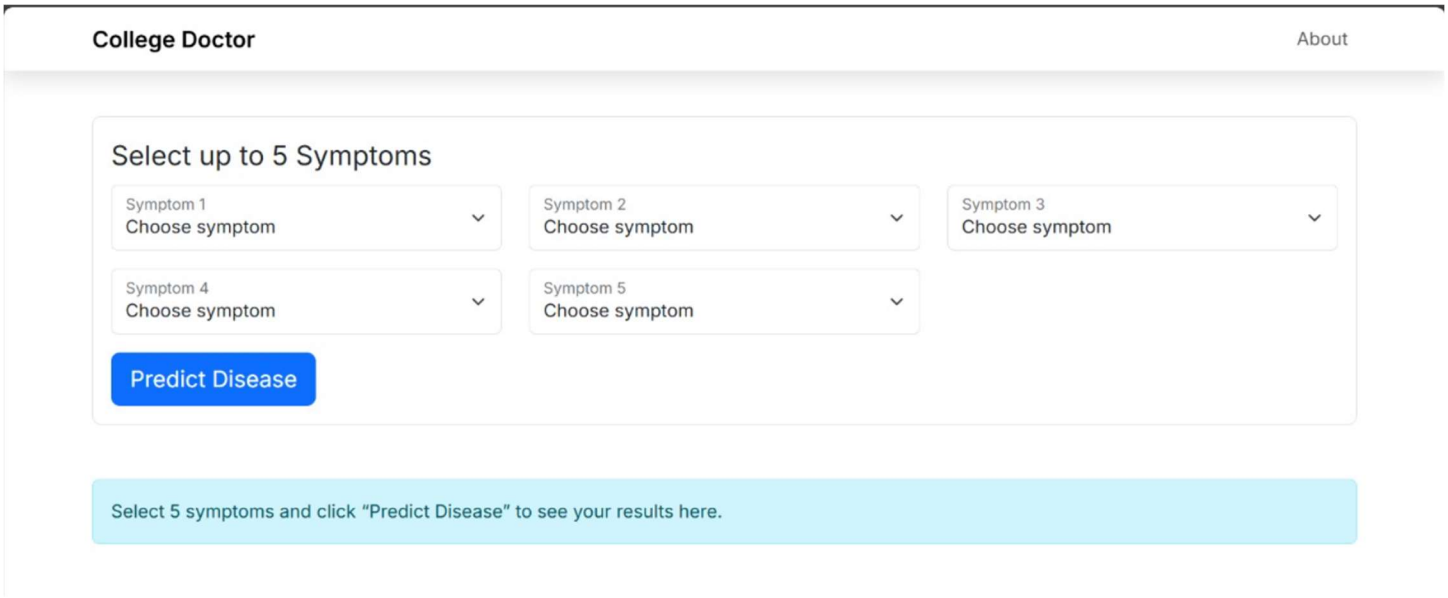
Conclusion

This Disease Detection System demonstrates the practical use of AI in healthcare. By combining machine learning with web scraping, the system not only predicts diseases but also offers actionable suggestions by directing users to relevant medical professionals.

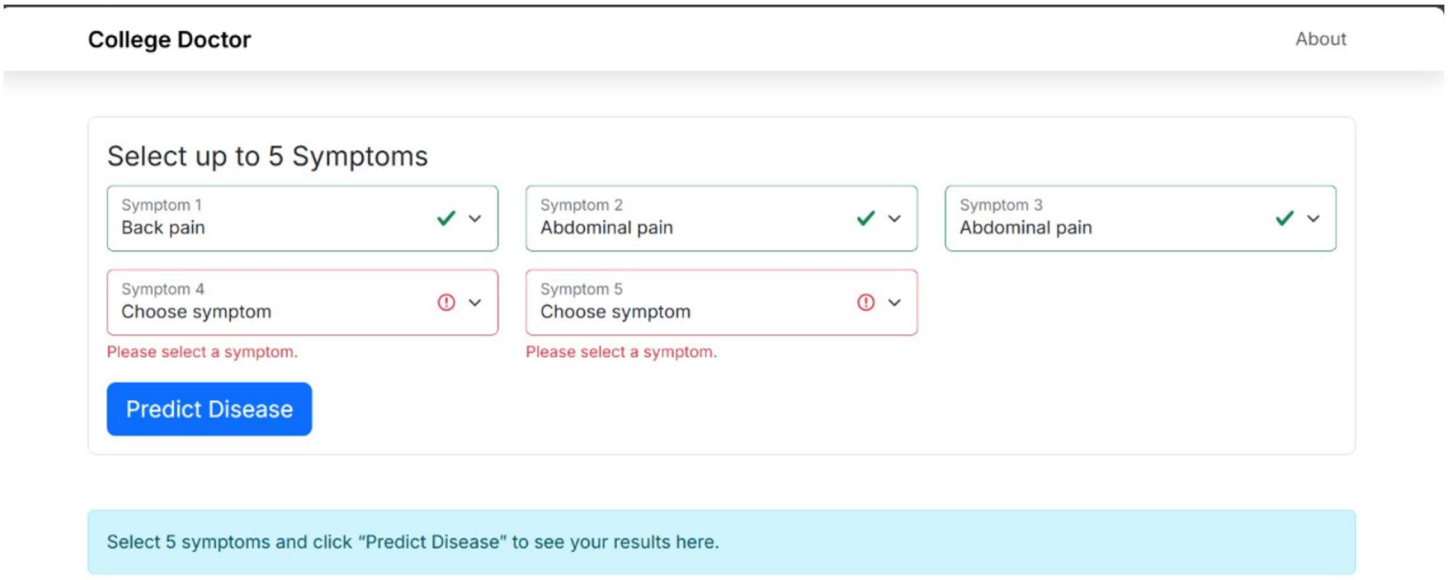
Note: Accuracy and predictions depend on the quality of input data and model training.

Screenshots

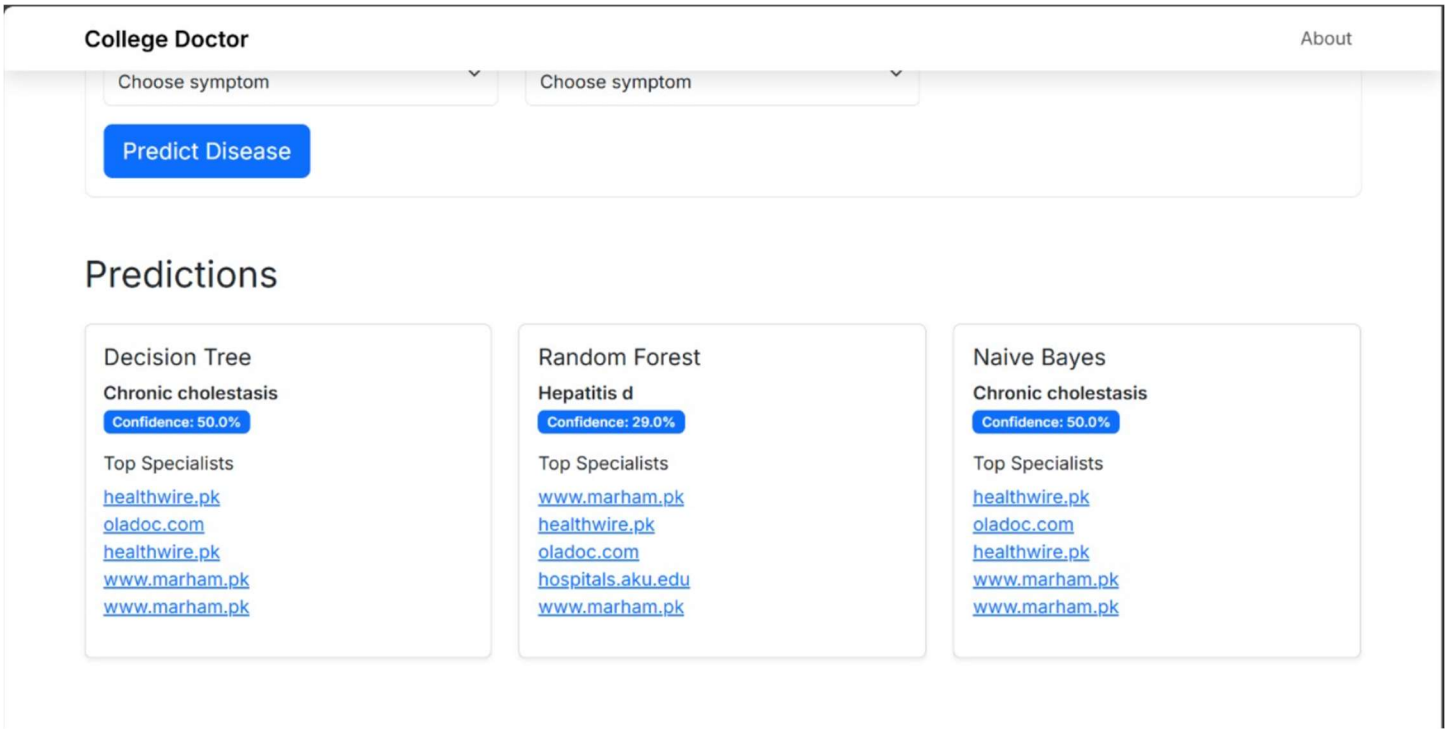
- **Dashboard**



- **Input Validation**



- **Predictions**



- **Redirection**