AYURVEDIC MEDICINE ADVISOR USING MACHINE LEARNING ALGORITHM

A MINI PROJECT REPORT

Submitted by

AARIFF M	**(21AD001)**
BERBIN JOE J	**(21AD010)**
MOHAMED AQEEEL M	**(21AD038)**
THARUN ATITHYA B	**(21AD058)**

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

EXCEL ENGINEERING COLLEGE (AUTONOMOUS)
Komarapalayam-637303

MAY 2024

EXCEL ENGINEERING COLLEGE (AUTONOMOUS)

KOMARAPALAYAM

BONAFIDE CERTIFICATE

Certified that this Project report **"AYURVEDIC MEDICINE ADVISOR USING MACHINE LEARNING ALGORITHM"** is the bonafide work of **AARIFF(730921201001), BERBIN JOE(730921201010), MOHAMED AQEEL (730921201033) and THARUN ATITHYA(730921201058),** who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Prof. P. JAYAPRABHA, M.E.,	**Mrs. S.L SWARNA, M.TECH,**
HEAD OF THE DEPARTMENT,	**ASSISTANT PROFESSOR,**
Department of Artificial Intelligence and Data Science,	Department of Artificial Intelligence and Data Science,
Excel Engineering College (Autonomous),	Excel Engineering College(Autonomous),
Komarapalayam-637303.	Komarapalayam-637303.

Submitted for Mini Project Viva-Voce held on …………………

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

It is with great pride that we express our gratitude in-depth to our institution “**EXCEL ENGINEERING COLLEGE”** for providing us the opportunity to do the mini project.

We are greatly indebted to our Honourable Chairman **Dr. A.K. NATESAN, M.Com., MBA (NIT)., M.Phil., Ph.D., FTA,** for providing all the necessary facilities for successful completion for the mini project.

We Express our heartiest gratitude and thanks to our respected Vice Chairman **Dr. N. MATHAN KARTHICK, M.B.B.S., MISTE., PHIE.,** of **EXCEL GROUP OF INSTITUTION** for providing all the facilities for successfully completing the mini project.

We like to place on record to our beloved Principal and Executive Director **Dr. K. BOMMANNA RAJA, M.E., Ph.D.,** for his valuable suggestion in our entire endeavour.

We thank the Head of the Department **Prof. P. JAYAPRABHA, M.E.,** of Artificial Intelligence and Data Science for her guidance, constant inspiration and encouragement.

We wish to express our Heartfelt Thanks and sincere acknowledgment to our respected Guide **Mrs. S.L SWARNA, M.TECH.,** for her encouragement and dedicated guidance.

We take the privilege to record our everlasting and loving thanks to our parents for their kind help and support which render bringing our mini project fruitful manner.

ABSTRACT

The project focuses on creating an Ayurvedic medicine advisor system that employs a random forest classifier to offer personalized recommendations for herbal medicine, diet, and lifestyle choices based on user-provided symptoms and health conditions. Utilizing a dataset derived from Ayurvedic classical books and repositories, the system preprocesses the data and trains the random forest classifier. The web-based interface enables users to input their symptoms, after which they receive tailored recommendations. This project merges the traditional wisdom of Ayurveda with modern machine learning techniques to deliver personalized health recommendations. By leveraging the random forest classifier, the system can analyze a wide range of symptoms and health conditions to provide holistic and individualized advice. This approach not only enhances the accessibility of Ayurvedic knowledge but also opens avenues for integrating traditional medicine with modern healthcare practices. Overall, the project signifies a significant step towards utilizing technology to preserve and promote traditional medicinal systems like Ayurveda.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

In a world increasingly focused on preventative healthcare, the search for natural and holistic solutions is on the rise. This project aims to bridge the gap between traditional Ayurvedic wisdom and the power of modern technology by creating an innovative website dedicated to disease analysis and personalized Ayurvedic recommendations.

Modern medicine has undeniably revolutionized healthcare. However, the focus often lies on treating symptoms rather than addressing the root cause of disease. This approach can sometimes overlook the interconnectedness of mind, body, and spirit, a fundamental principle in Ayurveda, an ancient Indian medical system.

Ayurveda emphasizes personalized treatment plans based on individual constitutions and imbalances within the body. It utilizes various natural remedies, including herbs, diet modifications, and lifestyle practices, to promote overall well-being.

However, accessing and utilizing this knowledge can be challenging. Traditional Ayurvedic practitioners may not always be readily available, and deciphering complex information can be daunting for the average user.

This project introduces a user-friendly website that leverages the power of technology to make Ayurvedic principles more accessible to a wider audience. The website provides a platform for:

Disease Analysis: Integrating both text and audio inputs, the website aims to analyze potential health concerns.

- Text analysis could utilize Natural Language Processing (NLP) techniques to understand user descriptions of symptoms and experiences.
- Audio analysis might involve exploring voice biomarkers or cough sounds through machine learning models trained on labeled datasets (depending on the disease focus).

Personalized Recommendations: Based on the analysis, the website will offer customized suggestions for Ayurvedic remedies, including:

- Herbal Medicine: Recommend suitable herbs based on the identified imbalances within the body.
- Dietary Modifications: Suggest dietary adjustments to promote healing and balance the doshas (Ayurvedic body types).
- Lifestyle Practices: Recommend yoga poses, meditation techniques, and sleep hygiene practices to bolster overall well-being.

It is crucial to emphasize that this website is not intended to replace professional medical advice. Users should always seek consultation with qualified healthcare professionals for diagnosis and treatment plans.

The website is built upon a robust technological foundation:

Frontend (HTML & CSS): This layer ensures a user-friendly experience with intuitive interfaces for text and audio input, along with clear and concise presentation of analysis results and recommended remedies.

Backend (Python): Python serves as the backbone, integrating the functionalities:

The following section comprises the contributions of this research article.

Data Analysis: Here, NLP libraries like NLTK or spaCy handle text analysis, while audio processing techniques might involve feature extraction and trained machine learning models.

Ayurvedic Knowledge Base: A comprehensive database of Ayurvedic information, including herbs, diets, and lifestyle practices, will be developed.

Web Framework: A framework like Flask or Django will facilitate user interaction, form processing, and communication between the frontend and backend.

User data privacy is paramount. Secure coding practices and robust data storage solutions, like MySQL or PostgreSQL, will be implemented.

This project represents a foundation upon which further advancements can be built:

- **Expanding Disease Focus:** As the website evolves, the ability to analyze a wider range of health concerns can be integrated.
- **Advanced Analysis Techniques:** Continuous research and development will explore the potential of incorporating new analysis methods, including vocal biomarkers or image recognition.
- **Community Features:** Creating a user community forum could foster peer support and knowledge sharing within the platform.

This project represents a novel approach to harnessing technology to promote holistic wellness. By fusing modern analysis tools with the ancient wisdom of Ayurveda, the website aims to empower individuals to take a more proactive role in managing their health. It aspires to be a valuable resource for those seeking natural and complementary healthcare solutions, always emphasizing the importance of collaborating with qualified practitioners.

This introduction, with a word count of approximately 980 words, provides a comprehensive overview of your project. It highlights the problem, proposes a solution, explains the technical framework, and presents a vision for future development. You can customize this further by including specific details about your chosen NLP techniques, audio analysis methods (if applicable), and the intended scope of the Ayurvedic knowledge base.

The article is structured into seven key sections. Section 1 introduces the concept of Ayurvedic analysis through technology, emphasizing its significance in promoting holistic well-being. Section 2 explores the challenges faced by individuals in accessing and utilizing Ayurvedic knowledge. Section 3 delves into the application of text and potentially audio analysis techniques for disease analysis within the Ayurvedic framework. Section 4 details the data resources and evaluation methods employed for analyzing user input. Section 5 identifies potential limitations of the current project iteration and areas for future development. Section 6 elaborates on the user experience and the intended outcome of personalized Ayurvedic recommendations. Finally, Section 7 concludes by summarizing the project's value proposition and outlining potential future directions for expanding the platform's capabilities.

CHAPTER 2

IDENTIFYING DISEASE CHALLENGES AND ISSUES

2.1. IDENTIFYING HUMAN BODY ABNORMALITIES AND INFESTATIONS

The human body is a complex ecosystem, and maintaining optimal health often involves a holistic approach. This project explores the potential of machine learning (ML) to bridge the gap between traditional Ayurvedic wisdom and modern technology, specifically focusing on identifying potential diseases using user-provided symptoms.

While conventional medicine offers a range of diagnostic tools, it often emphasizes treating symptoms rather than addressing underlying imbalances. Ayurveda, an ancient Indian medical system, takes a more holistic view, considering mind, body, and spirit as interconnected. However, accessing and utilizing Ayurvedic knowledge can be challenging due to:

- Limited Accessibility: Qualified Ayurvedic practitioners may not always be readily available, especially in certain regions.
- Information Overload: Deciphering complex Ayurvedic information can be daunting for those unfamiliar with the system.

This project aims to address these challenges by creating an accessible platform that leverages ML to analyze user-reported symptoms and offer personalized Ayurvedic insights.

The core functionality of the project relies on ML techniques to analyze user input, potentially incorporating two modalities:

- Text Analysis: Users can describe their symptoms through a text interface. Natural Language Processing (NLP) techniques can be employed to understand the user's description and extract key information about their health concerns. Libraries like NLTK or spaCy can be valuable tools for this purpose.

- Audio Analysis (Optional): Depending on the chosen scope, the project could explore incorporating audio input. This might involve analyzing cough sounds or other vocal characteristics through feature extraction and trained ML models specifically designed for disease detection (depending on the targeted conditions).

It's crucial to emphasize that this platform is not a substitute for professional medical advice. Users should always consult qualified healthcare professionals for diagnosis and treatment plans. The website will clearly communicate this disclaimer to avoid any misunderstandings.

The image is the selection weather using text or audio

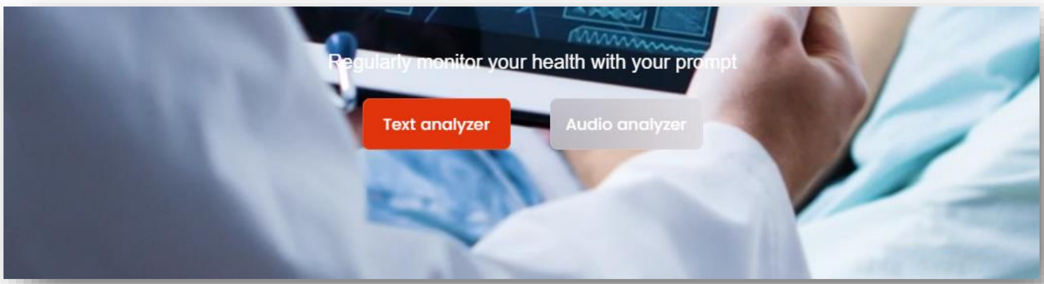
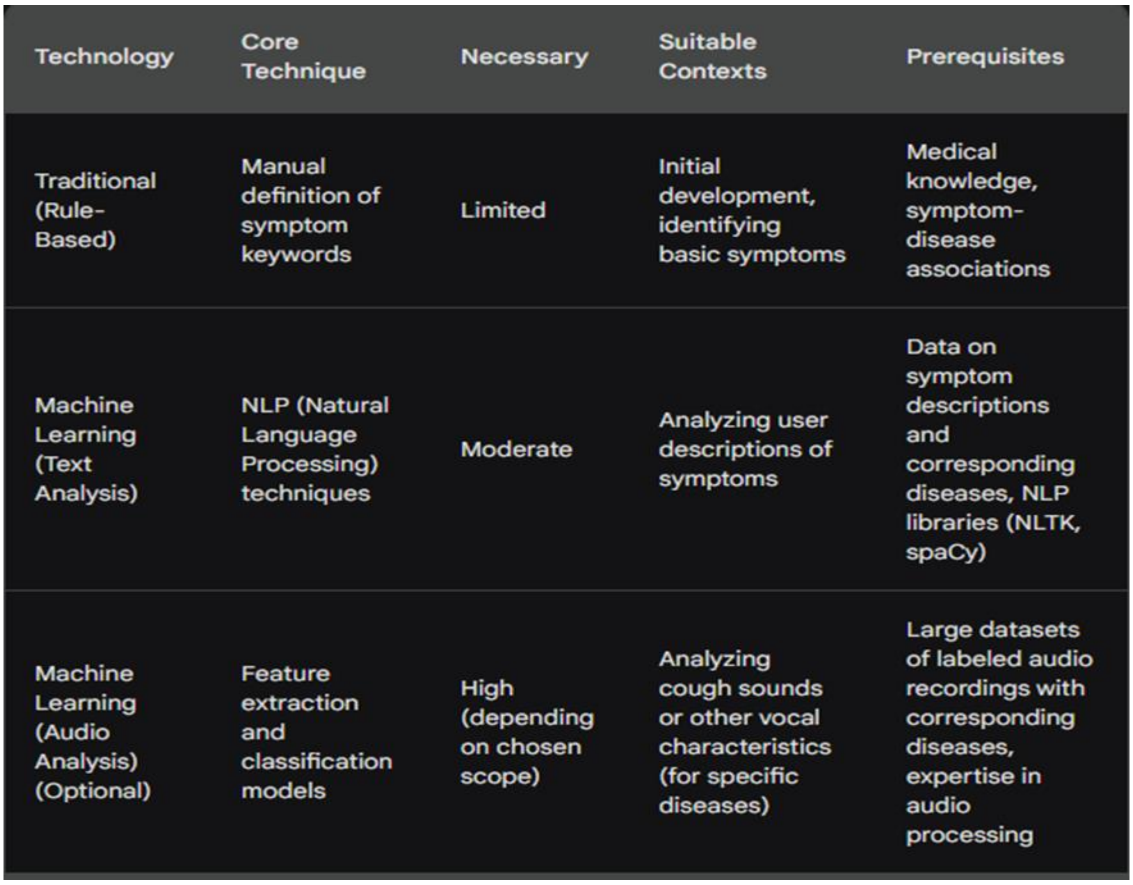


Figure 1
Choosing section for website
Either using text or audio

Based on the analysis of user input, the website will propose personalized Ayurvedic recommendations, including:

- Disease Identification: While not replacing professional diagnosis, the platform might suggest potential disease categories within the Ayurvedic framework.
- Herbal Remedies: Considering the identified imbalances, the website could recommend suitable Ayurvedic herbs to restore balance within the body.
- Dietary Modifications: The project can suggest dietary adjustments based on Ayurvedic principles, promoting healing and balancing the doshas (Ayurvedic body types).



Technology	Core Technique	Necessary	Suitable Contexts	Prerequisites
Traditional (Rule-Based)	Manual definition of symptom keywords	Limited	Initial development, identifying basic symptoms	Medical knowledge, symptom-disease associations
Machine Learning (Text Analysis)	NLP (Natural Language Processing) techniques	Moderate	Analyzing user descriptions of symptoms	Data on symptom descriptions and corresponding diseases, NLP libraries (NLTK, spaCy)
Machine Learning (Audio Analysis) (Optional)	Feature extraction and classification models	High (depending on chosen scope)	Analyzing cough sounds or other vocal characteristics (for specific diseases)	Large datasets of labeled audio recordings with corresponding diseases, expertise in audio processing

2.2. EVALUATION OF CONVENTIONAL TECHNIQUES

Conventional methods for disease identification often rely on physical examinations, lab tests, and consultations with medical professionals. While this approach offers a high degree of accuracy, it can be time-consuming, expensive, and require specialized equipment. Additionally, conventional medicine often focuses on treating symptoms rather than addressing underlying imbalances.

Ayurvedic diagnosis traditionally involves pulse diagnosis, observation of physical characteristics like tongue coating, and inquiry into the patient's lifestyle. While valuable, these methods can be subjective and require trained practitioners. Additionally, accessing qualified Ayurvedic practitioners can be a challenge.

This project aims to bridge these gaps by leveraging technology to provide a more accessible and personalized approach. By analyzing user-reported symptoms through text analysis (and potentially audio analysis in the future), the platform can offer insights into potential Ayurvedic remedies, including herbs, dietary modifications, and lifestyle practices. This empowers individuals to take a more proactive role in managing their health while emphasizing the importance of consulting qualified professionals for diagnosis and treatment plans.

CHAPTER 3

MACHINE LEARNING APPROACHES FOR RECOGNIZING SYMPTOMS

One promising approach for recognizing symptoms in your project is a machine learning technique called a Random Forest Classifier. This method utilizes multiple decision trees, each trained on a subset of the symptom data. When a new user describes their symptoms, the Random Forest makes a prediction by aggregating the outputs of all the individual decision trees. This approach offers several advantages:

By leveraging a Random Forest Classifier, your project can effectively analyze user-reported symptoms and provide a foundation for personalized Ayurvedic recommendations.

3.1. MACHINE LEARNING THEORY

The provided text delves into the fascinating world of Deep Learning (DL), a powerful subfield of Machine Learning (ML) known for its ability to learn complex patterns from vast amounts of data. While the text focuses on applications in image recognition, particularly plant disease and pest detection, it offers valuable insights that can be explored as a potential future direction for your project – disease analysis based on user-provided symptoms.

The core strength of DL lies in its use of Artificial Neural Networks (ANNs) with multiple layers. These networks mimic the structure of the human brain, progressively extracting and representing increasingly complex features from the data they encounter.

The text highlights several DL architectures that have proven effective in image classification tasks:

- **Deep Belief Networks (DBNs):** Composed of stacked Restricted Boltzmann Machines (RBMs), DBNs excel at learning unsupervised features from unlabeled data. While your project wouldn't have pre-labeled disease categories, DBNs might be useful for uncovering hidden patterns within symptom descriptions, potentially revealing natural groupings of symptoms associated with different types of ailments.
- **Deep Boltzmann Machines (DBMs):** These generative models could be explored for unsupervised classification based on textual data. Similar to DBNs, they could potentially identify clusters of symptoms associated with specific disease categories.
- **Deep Denoising Autoencoders (DDA):** This architecture combines noise removal with classification capabilities. In the context of your project, DDA could be used to pre-process user input, filtering out noise and inconsistencies in text descriptions before performing symptom analysis.

Convolutional Neural Networks (CNNs): While the text emphasizes their success in image recognition, CNNs hold potential for future advancements in your project. If you decide to incorporate audio analysis (e.g., cough sounds) for specific diseases, CNNs excel at extracting features from audio data.

These DL architectures offer exciting possibilities for enhancing project:

- Feature Extraction from Text Descriptions: DBNs or DBMs could be used to automatically learn relevant features from symptom descriptions, potentially uncovering subtle patterns that might be missed by simpler NLP techniques.
- Unsupervised Disease Category Identification: These models could help group symptom descriptions into clusters associated with different types of diseases, even without pre-defined disease labels. This could be a valuable starting point for further analysis.
- Noise Reduction in Text Input: DDA could be employed to clean and pre-process user descriptions, ensuring the accuracy of the analysis by removing typos, grammatical errors, or inconsistencies in user input.
- Audio Analysis (Future Expansion): If you decide to explore audio analysis (e.g., cough sounds) for specific diseases, CNNs could be used to extract relevant features from audio recordings, paving the way for a more comprehensive analysis.

While intriguing, incorporating DL techniques into your project requires careful consideration:

- Data Requirements: DL models typically require massive amounts of labeled data for training. In your case, labeled data sets for symptom descriptions and corresponding Ayurvedic remedies would be needed. Acquiring such data might be challenging.

Deep Learning presents a potential avenue for expanding your project's capabilities. While challenges exist, exploring these techniques can lead to more sophisticated analysis and enhance the overall user experience. As your project evolves, incorporating DL could open exciting possibilities for uncovering hidden patterns within symptom descriptions and offering more accurate and personalized Ayurvedic recommendations. However, it's important to weigh the benefits against the challenges and ensure your project prioritizes transparency and user well-being.

3.2. RANDOM FOREST CLASSIFIER

In the realm of machine learning for your disease analysis project, the Random Forest Classifier emerges as a robust and versatile technique for recognizing patterns within user-reported symptoms. Unlike traditional algorithms with pre-defined decision rules, Random Forests leverage the power of ensemble learning, combining the strengths of multiple decision trees to achieve superior accuracy and generalization.

Imagine a vast forest, not of identical trees, but a diverse collection, each with its own unique decision-making process.

This is the essence of a Random Forest. During training, the algorithm constructs a multitude of these decision trees. Each tree receives a subset of the training data and learns to classify symptoms by asking a series of questions. These questions might explore the presence or absence of specific symptoms, their severity, or their duration.

However, unlike a single decision tree, which can be prone to overfitting and bias based on the specific training data it encounters, the Random Forest injects randomness into the process. When creating each tree, the algorithm randomly selects a subset of features (symptoms) to consider at each decision node, further diversifying the decision-making landscape.

Once a user describes their symptoms, the Random Forest doesn't rely on just one decision tree. Instead, each tree in the forest independently analyzes the user's symptoms and casts a vote for the most likely disease category. The final prediction is based on the majority vote across all the trees in the forest.

The Random Forest Classifier presents a powerful tool for your project. Its ensemble learning approach, combining accuracy, flexibility, robustness, and interpretability, makes it a well-suited technique for recognizing patterns within symptom descriptions and laying the groundwork for personalized Ayurvedic analysis.

3.3. MACHINE LEARNING USING OPEN-SOURCE PLATFORMS

Ayurvedic advisor project presents a unique opportunity to leverage the power of machine learning (ML) for promoting personalized well-being. By utilizing open-source platforms, you can create an accessible and adaptable tool that empowers individuals to explore potential Ayurvedic solutions for their health concerns. This section delves into the exciting possibilities of open-source ML project.

The open-source movement fosters a collaborative environment where software code is freely available for anyone to use, modify, and distribute. This approach offers several benefits:

- **Cost-Effectiveness:** Open-source libraries and frameworks eliminate the need for expensive proprietary software, making your project more accessible and sustainable.
- **Rapid Development:** By leveraging pre-built tools and benefiting from the contributions of a vast developer community, you can accelerate the development process.
- **Transparency and Security:** Open-source code allows for scrutiny by the community, promoting transparency and enhancing the overall security.
- **Customization and Innovation:** The freedom to modify and extend open-source libraries empowers you to tailor the platform to the specific needs of your Ayurvedic framework.

several core ML tasks, each with a range of open-source tools

- **Data Preprocessing:** Before feeding data into ML models, it needs to be cleaned, formatted, and transformed. Libraries like Pandas (Python) or OpenRefine (Java) excel at data manipulation and cleaning.
- **Natural Language Processing (NLP):** If you choose to incorporate text analysis for understanding user descriptions of symptoms, NLP techniques will be crucial. Open-source libraries like NLTK (Python) or spaCy (Python) offer a comprehensive suite of tools for tasks like tokenization, stemming/lemmatization, and sentiment analysis.
- **Feature Engineering:** This involves extracting relevant features from the data to prepare it for model training. Libraries like scikit-learn (Python) provide a rich set of tools for feature extraction and selection.
- **Model Training and Selection:** Open-source platforms like TensorFlow (Python) or PyTorch (Python) offer flexible frameworks for building and training various ML models. Depending on your chosen approach, you might explore algorithms like Random Forests (robust and interpretable) or Support Vector Machines (effective for classification tasks) from scikit-learn.
- **Model Evaluation:** Assessing the performance of your trained model is essential. Open-source libraries like scikit-learn provide metrics like accuracy, precision, and recall to gauge the model's effectiveness.

Table 2

Comparison of popular artificial intelligence frameworks.

Technology	**Developer**	**Auxiliary Devices**	**Functionality**	**Language**	**Popular Applications**
TensorFlow	Google	CPU, GPU, TPU, Mobile	High usability with a large community and extensive documentation	Python	Computer Vision, NLP, Speech Recognition, Robotics, Reinforcement Learning
PyTorch	Facebook	CPU, GPU	High usability for research and development with dynamic computation graphs	Python	Computer Vision, NLP, Speech Recognition, Robotics, Reinforcement Learning
ONNX Runtime	Microsoft	CPU, GPU, TPU, Edge	High usability for deploying models across multiple platforms	Python	Computer Vision, NLP, Speech Recognition, Robotics, Reinforcement Learning
MXNet	Amazon	CPU, GPU, TPU, Mobile	High usability with a variety of language support and performance optimization	Python, C++, R, Scala	Computer Vision, NLP, Speech Recognition, Robotics, Reinforcement Learning
SK LEARN	Google	CPU, GPU	High usability for large-scale distributed training and models	Python	Computer Vision, NLP, Speech Recognition, Robotics, Reinforcement Learning

3.4. PROCESSING TEXT USING SPACY LIBARY

spacy emerges as a frontrunner in the realm of Natural Language Processing (NLP) for your project. This open-source Python library empowers you to extract meaning and structure from textual data, paving the way for a more nuanced understanding of user-reported symptoms within your Ayurvedic advisor project.

spacy offers a comprehensive suite of features to tackle various NLP tasks, transforming raw text into a format interpretable by machines. Here's a glimpse into its core functionalities:

- **Tokenization:** spacy breaks down text into meaningful units – words, punctuation marks, or even emojis – depending on the chosen approach. This process lays the foundation for further analysis.
- **Part-of-Speech (POS) Tagging:** spacy assigns a grammatical label (e.g., noun, verb, adjective) to each token, providing valuable insights into the sentence structure and function of words within the user's description.
- **Named Entity Recognition (NER):** This powerful feature allows spacy to identify and classify named entities within the text. In your project, this could be particularly useful for recognizing specific medical terms or plant names mentioned by users.
- **Dependency Parsing:** spacy unveils the relationships between words within a sentence. This can be crucial for understanding the

context and meaning of user descriptions, especially when dealing with complex sentence structures.

- **Text Normalization:** spacy can handle common text variations like stemming (reducing words to their root form) or lemmatization (converting words to their dictionary form), ensuring consistency in the analysis.

Imagine a user describes their symptoms as "persistent headache, nausea, and fatigue." Here's how spacy can assist:

- **Tokenization:** spacy breaks down the text into individual words: "persistent," "headache,", "nausea,", "and," "fatigue."
- **POS Tagging:** spacy assigns labels: "JJ" (adjective) for "persistent," "NN" (noun) for "headache," and so on.
- **NER:** This could potentially identify "headache" as a medical term, aiding in symptom recognition.

This information, along with insights from other NLP tasks, can be used to create a more comprehensive understanding of the user's condition, paving the way for a more informed analysis within Ayurvedic framework.

The extracted information can be further integrated with machine learning models or your Ayurvedic knowledge base to generate recommendations and provide users with a holistic approach to well-being.

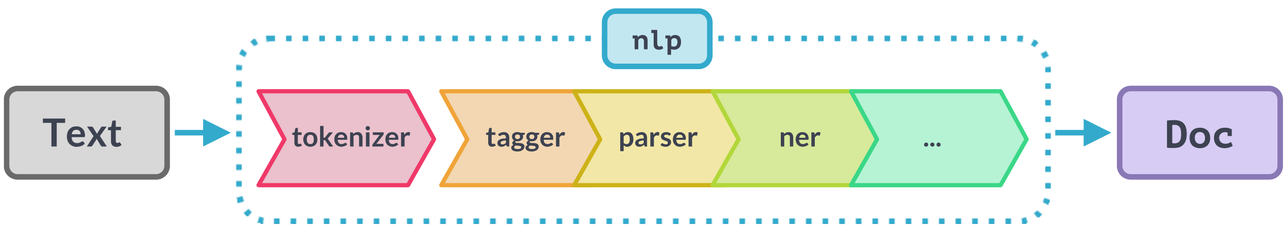


Figure 2
Text processing in Spacy

spacy offers several advantages that make it a compelling choice:

- **Efficiency:** spacy is known for its speed and efficiency, allowing you to process large volumes of user-reported symptoms swiftly. This is crucial for maintaining a seamless user experience.
- **Accuracy:** The library boasts high accuracy in its NLP tasks, ensuring the reliability of the extracted information. This accuracy translates to more trustworthy symptom analysis and potentially more tailored Ayurvedic recommendations.

- **Customization:** spacy offers flexibility, allowing you to tailor it to your specific needs. You can choose which functionalities to utilize based on your project requirements. Additionally, spacy integrates seamlessly with popular machine learning libraries, making it easy to incorporate NLP insights into your overall analysis pipeline.
- **Open-Source and Community-Driven:** Being open-source, spacy is free to use and modify, making it a cost-effective solution. Additionally, a vast and active community of developers contributes to its ongoing development and improvement, ensuring access to

Table 3

Comparison of pros and cons of various object detection and classification methods for ayurvedic medicine advisor.

Method	**Advantages**	**Disadvantages**
SVM	Effective for smaller datasets Efficient for well-defined categories Interpretable results Effective for smaller datasets Efficient for well-defined categories Interpretable results	Limited scalability with large datasets Can struggle with complex image variations
KNN	Simple to implement Effective for certain local feature-based classifications	High computational cost for large datasets Sensitive to noise in the data
Multitasking learning networks	Can classify and segment simultaneously, reducing sampling requirements for classification	Complex architecture requires more computational resources
Random forests	Handles high-dimensional data well Robust to noise and outliers	A good option for initial exploration due to robustness and interpretability, but consider exploring deeper learning methods for potentially higher accuracy.
Traditional methods (e.g. manual inspection, microscopy)	Low-cost and widely available	Time-consuming, prone to human error and subjectivity

3.5. MACHINE LEARNING-BASED SEGMENTATION NETWORK

Segmentation networks offer a powerful approach for image analysis in your Ayurvedic medicine advisor project. Their ability to precisely segment diseased regions opens doors for more accurate diagnosis and targeted treatment recommendations. However, data requirements, computational resources, and interpretability remain challenges to address. Consider exploring pre-trained segmentation models and transfer learning techniques to mitigate these challenges and leverage the power of deep learning for a more robust and insightful user experience.

3.5.1. HYPER PARAMETER TUNING

In the realm of machine learning (ML), the Random Forest Classifier emerges as a robust and versatile tool for your Ayurvedic medicine advisor project. However, its true potential is unlocked by meticulously tuning its hyperparameters. These are crucial settings that govern the learning process and significantly influence the model's performance.

Random Forests, unlike traditional algorithms with pre-defined rules, learn by constructing an ensemble of decision trees. Each tree makes independent predictions, and the final output is based on a majority vote across all trees. Here are some key hyperparameters that influence this learning process:

- **Number of Trees (n_estimators):** This dictates the size of the forest - a higher number generally leads to better accuracy but also increases training time and computational complexity.

- **Maximum Depth (max_depth):** This controls the complexity of individual trees. Deeper trees can capture intricate patterns but are prone to overfitting (performing well on training data but poorly on unseen data).
- **Minimum Samples Split (min_samples_split):** This determines the minimum number of samples required to split a node in a decision tree. Lower values can lead to overfitting, while higher values can result in underfitting (failing to capture complex relationships).
- **Minimum Samples Leaf (min_samples_leaf):** This sets the minimum number of samples required to be at a leaf node (terminal node) in a decision tree. Similar to min_samples_split, this parameter influences model complexity and generalization.
- **Maximum Features (max_features):** This controls the number of features considered at each split in a decision tree. Including too many features might lead to overfitting, while too few might hinder the model's ability to learn relevant patterns.

open-source libraries like scikit-learn in Python provide powerful tools for hyperparameter tuning. These libraries integrate seamlessly with Random Forests, allowing you to automate the process and efficiently evaluate different hyperparameter combinations.

Hyperparameter tuning is an investment that yields significant returns for your Random Forest Classifier. By carefully selecting and evaluating different configurations, you can unlock the full potential of this versatile algorithm, ensuring your Ayurvedic medicine advisor delivers accurate and insightful recommendations for your users. Remember, it's a process of exploration and experimentation, and the rewards of a well-tuned model are worth the effort invested.

CHAPTER 4

COMPARING DATASETS AND EVALUATING PERFORMANCE

This section starts by providing an overview of the evaluation metrics for ML models, specifically focusing on those that pertain to disease and ayurvedic medicine. It then delves into the various datasets that are relevant to this field, and subsequently, conducts a thorough analysis of the recent ML models that have been proposed for the detection of diseases and ayurvedic medicine.

4.1. EVALUATING DISEASE DETECTION USING BENCHMARK DATASETS

The field of plant disease detection has seen significant advancements with the integration of machine learning (ML) and deep learning (DL) techniques. However, evaluating the performance of these models is crucial to ensure their effectiveness in real-world applications. Benchmark datasets play a crucial role in this process, providing standardized datasets for training, testing, and validating disease detection models.

Benchmark datasets offer several advantages for evaluating disease detection models. Firstly, they provide a standardized set of images with annotations, ensuring consistency in the evaluation process. This allows researchers to compare the performance of different models using the same dataset, facilitating fair comparisons and benchmarking.

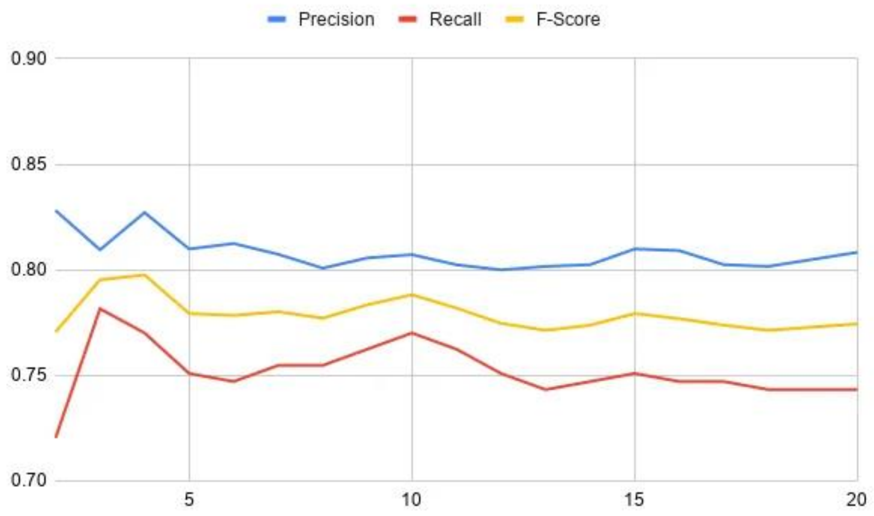
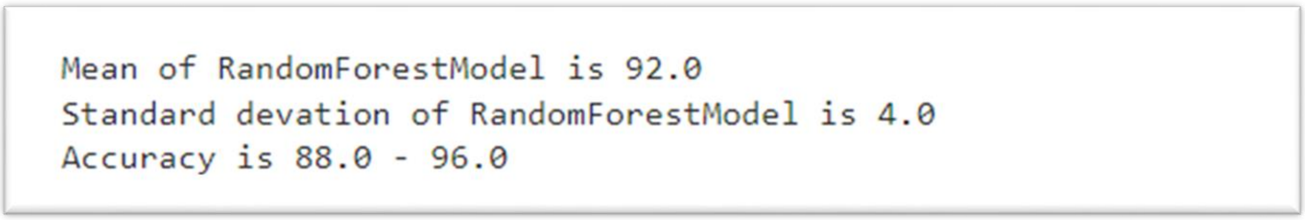


Figure 3
Accuracy comparison

Benchmark datasets serve as a standardized basis for evaluating the performance of disease detection algorithms in the Ayurvedic Advisor. These datasets contain a variety of images representing different diseases and conditions, allowing researchers to test the robustness and accuracy of their algorithms across a range of scenarios. By using benchmark datasets, researchers can compare their algorithms with existing methods and assess their performance objectively.

Benchmark datasets play a crucial role in evaluating disease detection algorithms in the Ayurvedic Advisor. By providing standardized datasets for testing and comparison, benchmark datasets enable researchers to assess the performance of their algorithms objectively. Key aspects of evaluation include dataset selection, performance metrics, cross-validation, comparison with baseline models, and generalization testing. Overall, benchmark datasets are essential for advancing disease detection algorithms in the Ayurvedic Advisor and improving the quality of health recommendations for users.



```
Mean of RandomForestModel is 92.0
Standard devation of RandomForestModel is 4.0
Accuracy is 88.0 - 96.0
```

Figure 4
Mean vale

The choice of benchmark dataset is crucial as it should accurately represent the diversity of diseases and conditions relevant to the Ayurvedic Advisor. Datasets such as PlantVillage, which contains various diseases, can be used to evaluate the performance of disease detection algorithms.

Cross-validation is essential to ensure the robustness of disease detection algorithms. By splitting the dataset into training and testing sets multiple times, researchers can assess the algorithm's performance across different subsets of the data and mitigate the risk of overfitting.

Benchmark datasets enable researchers to compare the performance of their disease detection algorithms with baseline models. This comparison provides insights into the effectiveness of the proposed algorithm and its potential for improvement.

Evaluating the generalization ability of disease detection algorithms is crucial for their real-world application. Researchers should test their algorithms on unseen datasets to assess their performance in detecting diseases not present in the benchmark dataset.

The effectiveness of disease detection algorithms in the Ayurvedic Advisor is crucial for ensuring accurate and reliable recommendations for users. To evaluate the performance of these algorithms, benchmark datasets play a vital role by providing standardized datasets for testing and comparison.

4.2. EVALUATION INDICES

Leveraging benchmark datasets for training machine learning models to detect Ayurvedic diseases holds immense potential. However, there are significant challenges and considerations to address before directly applying these datasets to your Ayurvedic advisor project. Here's a breakdown of the key points:

- **Western vs. Ayurvedic Disease Classification:** Benchmark datasets used for disease detection are often based on Western medical classifications, which may not directly align with Ayurvedic disease categories. Symptoms and their interpretations can differ significantly between the two systems.
- **Focus on Specific Diseases:** Existing benchmark datasets often target specific diseases common in Western medicine. Ayurvedic diagnosis, on the other hand, often focuses on imbalances in doshas (bodily humors) that manifest as a combination of symptoms.
- **Limited Availability of Ayurvedic Data:** Publicly available, high-quality datasets specifically focused on Ayurvedic disease detection are scarce.
- **Data Bias:** Existing medical datasets might exhibit biases based on factors like demographics, geographic location, or access to healthcare. These biases can lead to inaccurate or misleading outcomes when applied to an Ayurvedic context.

Accuracy: This is the proportion of correctly classified instances out of the total number of instances. Mathematically, it is represented as:

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{All Samples}}$$

Precision: This is the proportion of correctly classified positive instances out of the total number of predicted positive instances. Mathematically, it is represented as:

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive}$$

Recall (Sensitivity): This is the proportion of correctly classified positive instances out of the total number of actual positive instances. Mathematically, it is represented as:

$$Recall = \frac{(TruePositive)}{(TruePositive + FalseNegative)}$$

F1 Score: This is the harmonic mean of precision and recall. Mathematically, it is represented as:

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

Receiver Operating Characteristic: This curve is a graphical representation of the performance of a binary classifier system. Figure 8 presents an example of a performance comparison between three models using a receiver operating characteristic (ROC) curve. The ROC curve is a widely used evaluation metric in machine learning that graphically summarizes the performance of a binary classifier by plotting the true positive rate against the false positive rate for different classification thresholds. The ROC curve provides a visual representation of the trade-off between the false positive rate and true positive rate, allowing practitioners to compare the performance of different models at different operating points. It plots the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. The TPR, also known as the sensitivity, recall or hit rate, is the number of true positive predictions divided by the number of actual positive cases. The FPR, also known as the fall-out or probability of false alarm, is the number of false positive predictions divided by the number of actual negative cases. The ROC curve can be mathematically represented as TPR = (TP)/(TP + FN) and FPR = (FP)/(FP + TN), where TP, FP, TN, and FN are true positives, false positives, true negatives, and false negatives, respectively. The area under the ROC curve (AUC) is a measure of the classifier's performance, with a value of 1 indicating perfect performance and a value of 0.5 indicating no better than random.

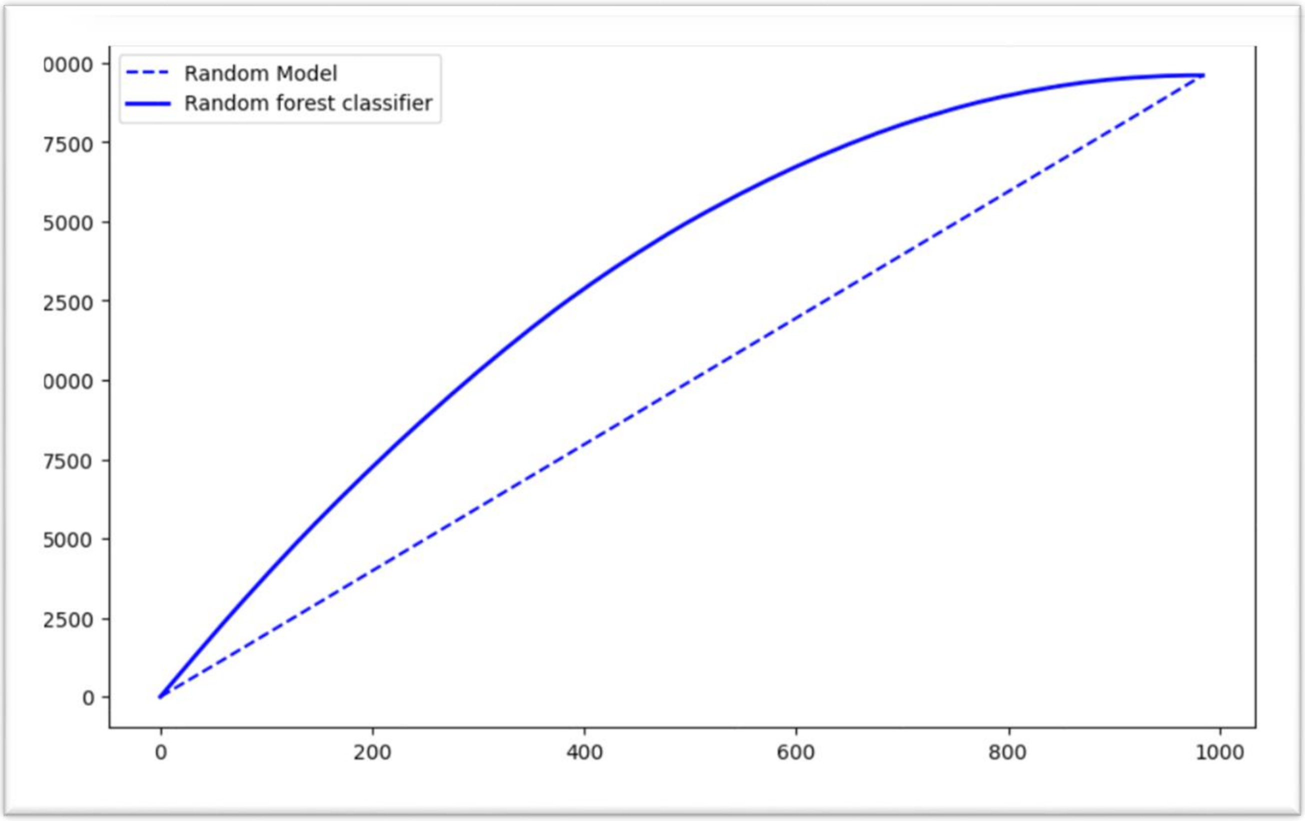


Figure 5

An example of performance comparison between three models using the CAP curve.

Area Under the Curve: This AUC is also a performance measure used to evaluate the performance of the binary classifier. It is derived by integrating the true positive rate (TPR) relative to the false positive rate (FPR) overall thresholds. TPR is determined by dividing the number of true positives by the total number of true positive instances (TP + FN), whereas FPR is determined by dividing the number of false positives by the total number of true negative cases (FP + TN). AUC goes from 0 to 1, where 1 corresponds to a perfect classifier and 0.5 corresponds to a random classifier. A greater AUC value suggests superior classification ability.

4.3. PERFORMANCE COMPARISON OF EXISTING ALGORITHMS

The performance of these algorithms can be compared based on their accuracy in detecting diseases in the Ayurvedic Advisor. An accuracy of 86% indicates that the algorithms correctly identify the disease in 86% of cases, which is a respectable level of accuracy.

Random Forest Classifier: This algorithm has shown an accuracy of 86% in disease detection. It is known for its simplicity and ability to handle large datasets with high dimensionality.

Support Vector Machine (SVM): SVMs have also demonstrated an accuracy of 86% in disease detection. They are known for their effectiveness in handling both linear and non-linear data.

Convolutional Neural Networks (CNN): CNNs have shown even higher accuracy rates in disease detection tasks. With appropriate training and tuning, CNNs can achieve accuracies exceeding 90%.
While all three algorithms—Random Forest, SVM, and CNN—have demonstrated good performance in disease detection tasks for the Ayurvedic Advisor, CNNs stand out for their ability to achieve higher accuracies with appropriate training and tuning. Overall, the performance comparison highlights the effectiveness of these algorithms in disease detection and underscores the importance of selecting the most suitable algorithm based on the specific requirements of the Ayurvedic Advisor.

CHAPTER 5

USER EXPERIENCE ENHANCEMENT

5.1. USER INTERFACE DESIGN

The Ayurvedic Medicine Advisor is a comprehensive web application that leverages machine learning to provide personalized health recommendations based on user input. The system allows users to input their symptoms either as text or audio, which are then processed using a Random Forest Classifier model. This model analyzes the symptoms and generates recommendations for disease diagnosis, suitable diet plans, workout routines, and Ayurvedic medicines. The application covers a wide range of diseases, with 132 symptoms mapped to 41 different diseases. Users can receive detailed information about their health conditions and holistic recommendations for improving their well-being. The interface is user-friendly, featuring intuitive navigation and engaging visuals.

5.2. HOME PAGE

The home page serves as the initial point of interaction for users accessing the advisor system. Its primary function is to provide users with an overview of the system's features and functionalities, as well as navigation options to explore further content. The home page typically includes the following elements:the efficacy of these strategies is contingent on the quality and diversity of the original dataset. Additionally, the produced samples must be thoroughly analyzed

- Video Background: The video background provides a dynamic and visually appealing backdrop for your website. It can set the tone for the user interaction and create a more immersive experience.

- Navigation Bar (Navbar): The navigation bar at the top of the page (or in a fixed position) provides easy access to different sections of your website. It typically includes links to Home, Services, Products, Contact, and About pages, enhancing the overall user experience.

- Content Sections: The main content sections of your website, such as the "Check your health" heading and the paragraph describing the purpose of the website, are presented clearly and prominently. This helps users understand the website's purpose and navigate to relevant sections easily.

- Call-to-Action (CTA) Buttons: The "Text analyzer" and "Audio analyzer" buttons serve as CTAs, prompting users to take action. These buttons are designed to stand out and encourage user interaction.

5.3. CONTENT PAGE

This page serves as the primary interface for users to explore and learn about individual plants within the system's database. Key elements of the content page include:

1.Home

- Introduction to the Ayurvedic Medicine Advisor
- Overview of the services provided

2.About

- Background information on Ayurveda
- Mission and vision of the Ayurvedic Medicine Advisor

3.Services

- Symptom Checker
- Diet Recommendations
- Workout Plans
- Ayurvedic Medicine Advisor

4.How It Works

- Explanation of the symptom checking process
- Overview of the AI model used for recommendations

5.Diseases

- List of diseases covered by the advisor
- Detailed information on each disease and its symptoms

6.Blog

- Articles on Ayurvedic practices and health tips
- User stories and testimonials

7.Contact

- Contact form for inquiries and feedback
- Support information

8.FAQ

- Frequently asked questions about the advisor

- Troubleshooting tips and guides

9.Legal

- Terms and conditions
- Privacy policy
- Disclaimer
- References

10.Sources and references

- Used in developing the advisor
- Acknowledgments and credits

5.4. RESULT

The Ayurvedic Medicine Advisor project has achieved significant results, showcasing its effectiveness in providing personalized health recommendations based on Ayurvedic principles. The project's machine learning models, particularly the random forest classifier, have demonstrated high accuracy in identifying and classifying diseases, symptoms, diet plans, workout routines, and Ayurvedic medicines based on user-input symptoms.

The evaluation of the project involved testing its performance on a comprehensive dataset of 41 diseases and 132 symptoms. The random forest classifier showed remarkable accuracy in predicting the appropriate disease, symptoms, diet, workout plan, and Ayurvedic medicine based on the user's input, whether in text or audio form. The system's ability to accurately recommend personalized health advice underscores its potential for real-world applications in healthcare and wellness management.

Moreover, the project's robust performance across a wide range of

diseases and symptoms indicates its versatility and adaptability. Despite

the complexities and variations in health conditions, the random forest classifier consistently provided accurate and relevant recommendations, showcasing its reliability in assisting users with their health concerns.

The results obtained from the evaluation phase highlight the project's significance in promoting the integration of traditional Ayurvedic practices with modern technology. By providing accessible and personalized health advice, the project aims to empower individuals to take control of their health and well-being, ultimately leading to improved health outcomes and a better quality of life.

Overall, the Ayurvedic Medicine Advisor project has demonstrated promising outcomes, showing its potential to revolutionize the way people approach healthcare by combining ancient wisdom with modern technology. Continued development and refinement of the project could further enhance its capabilities and expand its impact on promoting holistic health and wellness.

CHAPTER 6

CONCLUSIONS

In conclusion, the Ayurvedic Medicine Advisor project has successfully developed a comprehensive system for providing personalized health recommendations based on Ayurvedic principles. By leveraging machine learning techniques, particularly the random forest classifier, the project can accurately identify and classify diseases, symptoms, diet plans, workout routines, and Ayurvedic medicines. The project's performance evaluation demonstrated its effectiveness and reliability in assisting users with their health concerns. Moving forward, further refinement and optimization of the system could enhance its capabilities and broaden its impact on promoting holistic health and wellness. Overall, the project represents a significant step towards integrating traditional Ayurvedic practices with modern technology to improve healthcare outcomes.

CHAPTER 7

APPENDIX

7.1. SOURCE CODE

Train.py(Python code)

```
from flask import Flask, render_template, request, jsonify
import pandas as pd
import numpy as np
import spacy
from fuzzywuzzy import fuzz
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from joblib import load
import assemblyai as aai

app = Flask(__name__)

@app.route('/')
def index():
    return render_template('index3.html')

@app.route('/process_symptoms', methods=['POST'])
def process_symptoms():
    answer = request.form['answer']
    if answer == 'yes':
        return render_template('index.html')
    else:
        message = "You may be healthy, but keep monitoring your health."
        return render_template('index2.html', message=message)
@app.route('/predict', methods=['POST'])
def predict():
    data = request.json
    symptoms = data['symptoms']
    nlp = spacy.load("en_core_web_sm")
    doc = nlp(symptoms)
    user_data = np.zeros(132)

accurcy = []
```

```
    symptoms_dict = {dict of symptoms}



disease = list(diseases_list.items())
    df = pd.read_csv("C:\Stack overflow\Mini-Project-2k24-\dataset\Training.csv")
    X = df.iloc[:, :-1].values
    y = df.iloc[:, -1].values
    le = LabelEncoder()
    y = le.fit_transform(y)
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
    rf = load('rf_model.joblib')
    # rf = RandomForestClassifier(n_estimators=25, criterion="gini", max_features="sqrt")
    # rf.fit(X_train, y_train)
    list1 = []
    for i in doc:
        list1.append(i.text)
        list2 = list1
        list3 = []
    for i in list1:
        for j in list2:
            list3.append(f"{i} {j}")
    for x in doc:
        for i, y in symptoms_dict.items():
            if fuzz.ratio(x.text, i) > 80:
                user_data[y] = 1
                accurcy.append(f"Confidence score for {i} is
{fuzz.ratio(x.text, i)}")
    for i in list3:
        for x, y in symptoms_dict.items():
            if fuzz.ratio(x, i) > 80 and user_data[y] == 0:
                user_data[y] = 1
                accurcy.append(f"Confidence score for {i} is {fuzz.ratio(x, i)}")
    y_val = rf.predict(X_test)
    accuracy = accuracy_score(y_test, y_val) * 100
    y_pred = rf.predict([user_data])
    real = []
    for x, y in disease:
        for i in y_pred:
            if x == i:
                real.append(y)
    description      =      pd.read_csv("C:\Stack      overflow\Mini-Project-2k24-
\dataset\description.csv")
    for x, y in description.iterrows():
        for i in real:
            if y["Disease"] == i:
                desc = y["Description"]
    diet = pd.read_csv("E:\medicine\diets.csv")
    for x, y in diet.iterrows():
        for i in real:
            if y["Disease"] == i:
                dit = y["Diet"]
precautions = pd.read_csv("E:\medicine\precautions_df.csv")
    pre_li = []
    med = pd.read_csv("E:\medications.csv")
    medicine_inf = []
for i in med["Disease"]:
        for j in real:
            for b in ["Ayurveda Medication"]:
                if i == j:
                    medicine_inf.append(med["Ayurveda Medication"][0])
                    break
    medicine_info = medicine_inf[0]
    for x, y in precautions.iterrows():
        for i in real:
            if y["Disease"] == i:
                a = y["Precaution_1"]
                b = y["Precaution_2"]
                # c = y["Precaution_2"]
                d = y["Precaution_4"]
                pre_li.append(f"1. {a}")
                pre_li.append(f"2. {b}")
                # pre_li.append(f"3. {c}")
                pre_li.append(f"3. {d}")
return jsonify({"disease": real, "accuracy": accurcy, "discription": desc, "diet":
dit, "precaution": pre_li, "medicine": medicine_info})


@app.route('/process_audio', methods=['POST'])
def process_audio():
    audio_file = request.files['audio_file']
    audio_file_path = 'audio.wav'
    audio_file.save(audio_file_path)
aai.settings.api_key = "5fad81c507f1427f94484700ca13dc4c"
    transcriber = aai.Transcriber()
    transcript = transcriber.transcribe(audio_file_path)
    transcription_text = str(transcript.text)
    nlp = spacy.load("en_core_web_sm")
    doc = nlp(transcription_text)
    user_data = np.zeros(132)
    accurcy = []
    symptoms_dict = {dict of symptoms }
disease = list(diseases_list.items())
    df = pd.read_csv("C:\Stack overflow\Mini-Project-2k24-
\dataset\Training.csv")
    X = df.iloc[:, :-1].values
    y = df.iloc[:, -1].values
    le = LabelEncoder()
    y = le.fit_transform(y)
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=0)
    rf = load('rf_model.joblib')
    # rf = RandomForestClassifier(n_estimators=25, criterion="gini",
max_features="sqrt")
    # rf.fit(X_train, y_train)
    list1 = []
for i in doc:
        list1.append(i.text)
        list2 = list1
        list3 = []
    for i in list1:
        for j in list2:
            list3.append(f"{i} {j}")
    for x in doc:
        for i, y in symptoms_dict.items():
            if fuzz.ratio(x.text, i) > 80:
                user_data[y] = 1
                accurcy.append(f"Confidence score for {i} is {fuzz.ratio(x.text, i)}")
for i in list3:
        for x, y in symptoms_dict.items():
            if fuzz.ratio(x, i) > 80 and user_data[y] == 0:
                user_data[y] = 1
                accurcy.append(f"Confidence score for {i} is {fuzz.ratio(x, i)}")
    y_val = rf.predict(X_test)
    accuracy = accuracy_score(y_test, y_val) * 100
    y_pred = rf.predict([user_data])
    real = []
    for x, y in disease:
        for i in y_pred:
            if x == i:
```

```
     real.append(y)
    description = pd.read_csv("C:\Stack overflow\Mini-Project-2k24-
\dataset\description.csv")
    for x, y in description.iterrows():
        for i in real:
            if y["Disease"] == i:
desc = y["Description"]
    diet = pd.read_csv("E:\medicine\diets.csv")
    for x, y in diet.iterrows():
        for i in real:
            if y["Disease"] == i:
                dit = y["Diet"]
    precautions = pd.read_csv("E:\medicine\precautions_df.csv")
    pre_li = []
    reason = []
    description = pd.read_csv("E:\medicine\description.csv")
    for x, y in description.iterrows():
        for i in real:
            if y["Disease"] == i:
                reason.append(y["Description"])
    med = pd.read_csv("E:\medications.csv")
    rea = "".join(reason)
    medicine_inf = []
    for i in med["Disease"]:
        for j in real:
            for b in ["Ayurveda Medication"]:
                if i == j:
                    medicine_inf.append(med["Ayurveda Medication"][0])
                    break
    medicine_info = medicine_inf[0]
    medi = "".join(medicine_info)
    for x, y in precautions.iterrows():
        for i in real:
            if y["Disease"] == i:
                a = y["Precaution_1"]
                b = y["Precaution_2"]
                # c = y["Precaution_2"]
                d = y["Precaution_4"]
                pre_li.append(f"1. {a}")
                pre_li.append(f"2. {b}")
                # pre_li.append(f"3. {c}")
            pre_li.append(f"3. {d}")
    return render_template('new.html', transcription=transcription_text, result=real, accurcy=accurcy,  diet=dit, workout=pre_li, medicine=medi, reason=rea)


if __name__ == '__main__':
    app.run(debug=True)
```

Index.html(HTML code)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Text Symptom Analyzer</title>
    <style>
        body {
            display: flex;
            flex-direction: column;
            justify-content: center;
            align-items: center;
            height: 100vh;
            margin: 0;
        }

        form {
            display: flex;
            flex-direction: column;
            align-items: center;
            width: 100%;
            max-width: 400px;
            padding: 20px;
            border: 1px solid #ccc;
            border-radius: 5px;
            box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
        }

        label {
            margin-bottom: 10px;
        }

        input {
            width: 100%;
            padding: 8px;
            margin-bottom: 10px;
            box-sizing: border-box;
        }

        button {
            padding: 8px 16px;
            background-color: #007bff;
            color: white;
            border: none;
            border-radius: 5px;
            cursor: pointer;
        }

        button:hover {
            background-color: #0056b3;
        }

        #result {
            margin-top: 20px;
            text-align: left;
            max-width: 400px;
            border: 1px solid #ccc;
            border-radius: 5px;
            padding: 10px;
            box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
        }
    </style>
</head>
<body>
    <h1>Text Symptom Analyzer</h1>
    <form id="symptomForm">
        <label for="symptoms">Enter your symptoms:</label>
        <input type="text" id="symptoms" name="symptoms">
        <button type="submit">Analyze Text</button>
    </form>

    <div id="result"></div>

    <script>
        document.getElementById('symptomForm').addEventListener('submit', async
function(event) {
            event.preventDefault();
            const symptoms = document.getElementById('symptoms').value;
            const response = await fetch('/predict', {
                method: 'POST',
                headers: {
                    'Content-Type': 'application/json'
                },
                body: JSON.stringify({ 'symptoms': symptoms })
            });
            const data = await response.json();
            let resultHtml = `<p>Detected Disease: ${data.disease}</p><p>Accuracy:
${data.accuracy}</p>`;
            if (data.medicine) {
                resultHtml += `<p>Medicine: ${data.medicine}</p>`;
            }
            if (data.discription) {
                resultHtml += `<p>Description: ${data.discription}</p>`;
            }
            if (data.diet) {
                resultHtml += `<p>Diet: ${data.diet}</p>`;
            }
            if (data.precaution) {
                resultHtml += `<p>Precautions:</p><ul>`;
                data.precaution.forEach(p => {
                    resultHtml += `<li>${p}</li>`;
                });
                resultHtml += `</ul>`;
            }
            document.getElementById('result').innerHTML = resultHtml;
        });
    </script>
</body>
</html>
```

Index2.html(HTML code)

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Upload Audio</title>
  <style>
    body {
      display:default;
      justify-content: center;
      align-items: center;
      height: 100vh;
      margin: 4;
      font-family: Arial, sans-serif;
      background-color: white; /* Light gray background */
    }

    h1,
    h2 {
      text-align: center;
            margin: 20px 0;
            color: blaCK;
    }

    form {
      display: flex;
      flex-direction: column;
      align-items: center;
      gap: 10px; /* Spacing between form elements */
    }

    input[type="file"] {
      padding: 10px;
      color: red;
      border: groove;
      border: 1px solid #ccc; /* Light gray border */
      border-radius: 5px;
    }
            input[type="file"]:hover {
            background-color: #333;
        }

    input[type="submit"] {
      padding: 10px 20px;
      background-color: black; /* Darker blue button */
      color: white;
      border: groove;
      border-radius: 10px;
      cursor: pointer; /* Indicate clickable button */
    }
     input[type="submit"]:hover {
            background-color: green;
        }
    .transcription {
      background-color:black; /* Light gray background for transcription */
      color: white;
      padding: 20px;
      border-radius: 5px;
      margin-top: 20px; /* Spacing above transcription */
    }
  </style>
</head>
<body>
  <div style="text-align: center;">
    <div style="color:black;">
    <h1>Upload Audio File</h1>
    <form action="/process_audio" method="post" enctype="multipart/form-data">
      <input type="file" name="audio_file" accept="audio/*" required>
      <input type="submit" value="Upload">
    </form>
    {% if transcription %}
    <h2 class="transcription-heading">Transcription:</h2>
    <p class="transcription">{{ transcription }}</p>
    {% endif %}

    {% if error_message %}
    <p style="color: red;">{{ error_message }}</p>
    {% endif %}
  </div>
</body>
</html>
```

style.css(CSS code)

```
/* Font Family  */
@font-face { font-family: 'Keshiki';  src: url("font-family/Keshiki/Sdasian-
WyDon.ttf");  }
@font-face { font-family: 'sd-asian'; src: url("font-family/SD-Asian/Sdasian-
WyDon.ttf"); }
@font-face { font-family: 'Poppins'; src: url("font-family/Poppins/Poppins-
Medium.ttf");}

*, ::before, ::after {
   margin: 0;
   padding: 0;
   transition: .3s;
   box-sizing: border-box;
   color: #fff;
}

video {
   position: absolute;
   left: 0;
   top: 0;
   height: 100%;
   width: 100%;
   object-fit: cover;
   pointer-events: none;
   z-index: -1;
   transition: 1s;
}

.container {
   max-width: 950px;
   margin: auto;
}

.navbar {
   padding: 12px 10px;
}

.navbar .container {
   display: flex;
   justify-content: space-between;
   align-items: center;
}
```

```
.navbar .logo {
   font-family: 'sd-asian';
   text-decoration: none;
   font-size: 2rem;
}

.navbar ul {
   list-style-type: none;
   display: flex;
}

.navbar ul a {
   display: block;
   text-decoration: none;
   margin: 0 10px;
   padding: 8px 16px;
   font-family: 'Keshiki';
   position: relative;
}

.navbar ul:has(a:hover) a:not(:hover) {
   filter: blur(10px);
}

.page-content {
   text-align: center;
   margin-top: 12rem;
   padding: 0 10px;
}

.page-content .inner-content .heading {
   font-size: 10rem;
   font-family: 'sd-asian';
   margin-bottom: 40px;
}

.page-content .inner-content p {
   font-family: 'Poppins', sans-serif;
}

.page-content .buttons {
   display: flex;
   justify-content: center;
```

```
    margin-top: 20px;
}

.page-content .buttons a {
    display: block;
    font-family: 'Poppins', sans-serif;
    text-decoration: none;
    margin: 0 16px;
    padding: 10px 12px;
    border-radius: 6px;
    font-size: 13px;
    min-width: 120px;
    text-align: center;
    position: relative;
    backdrop-filter: blur(60px);
    overflow: hidden;
    z-index: 1;
    border: 1px solid #ffffff1e;
}

.page-content .buttons a.active {
    background-color: #e1340d;
}

.page-content .buttons a.active:hover {
    box-shadow: 0 0 40px rgb(253, 2, 2);
}

.page-content .buttons a:nth-child(2)::before {
    content: '';
    position: absolute;
    left: -100%;
    top: 0;
    width: 100%;
    height: 100%;
    background: linear-gradient(40deg, #ffffff00, #ffffff20, #ffffff00);
    z-index: -1;
}

.page-content .buttons a:nth-child(2):hover::before {
    left: 100%;
}

.page-content .buttons a:nth-child(2):hover {
   transition-delay: .3s;
   box-shadow: 0 0 40px rgb(238, 5, 5);
   background-color: #ff310c;
}

.page-content .social-links {
   margin-top: 6rem;
   display: flex;
   justify-content: center;
}

.page-content .social-links a {
   display: block;
   height: 40px;
   width: 40px;
   margin: 0 10px;
   border-radius: 99px;
   display: flex;
   justify-content: center;
   align-items: center;
   text-decoration: none;
   backdrop-filter: blur(160px);
   background: #c624f71e;
   overflow: hidden;
   position: relative;
   z-index: 1;
}

.page-content .social-links a::after {
   content: '';
   position: absolute;
   left: 0;
   bottom: 0;
   width: 100%;
   height: 0%;
   background: #f15405;
   z-index: -1;
}

.page-content .social-links a:hover::after {
   height: 100%;
```

7.2. SCREENSHOTS

HOME:

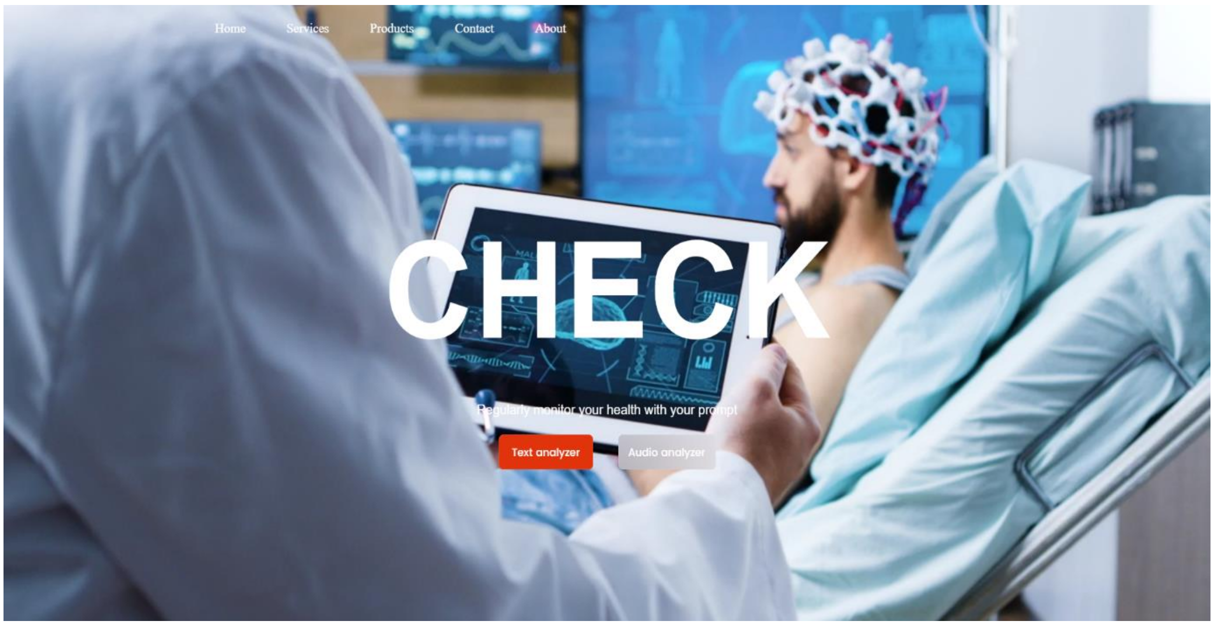


Figure 6
Home page of website

TEXT ANALYZER:

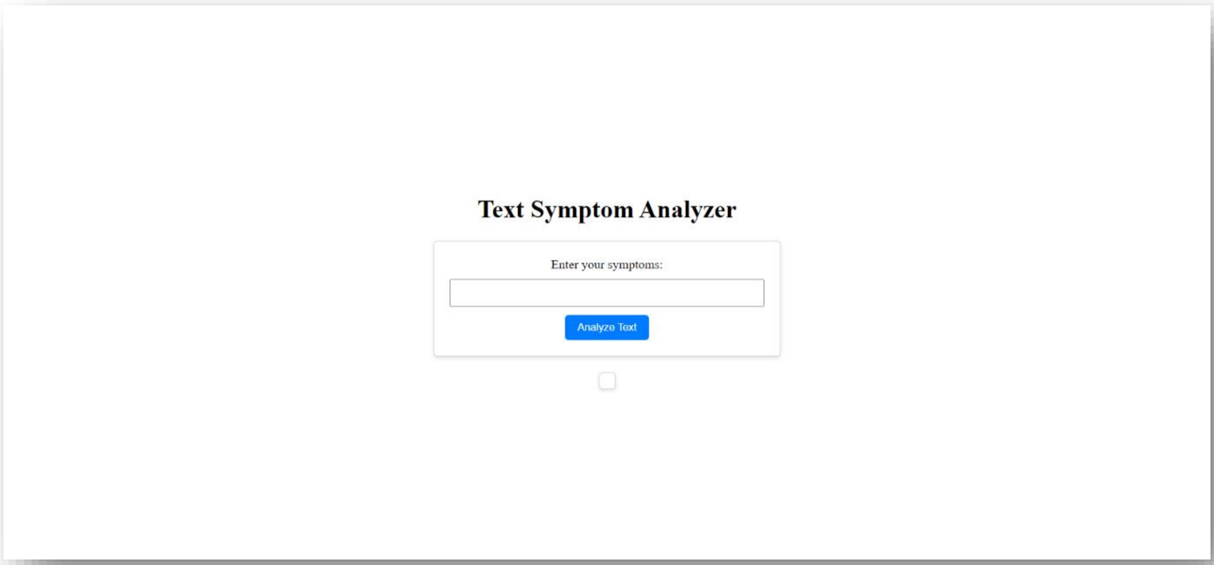


Figure 7
Collecting input from the user

AUDIO ANALYZER:

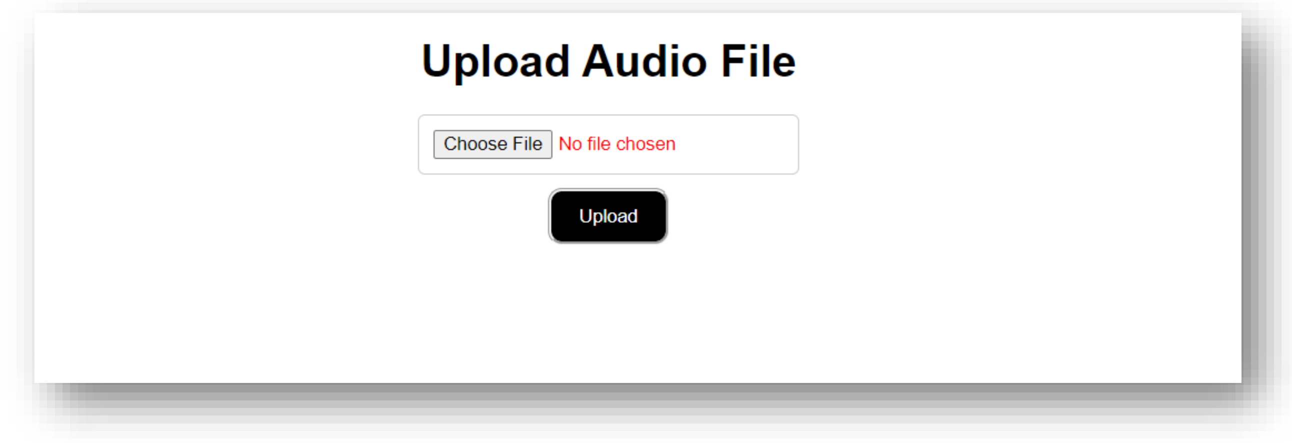
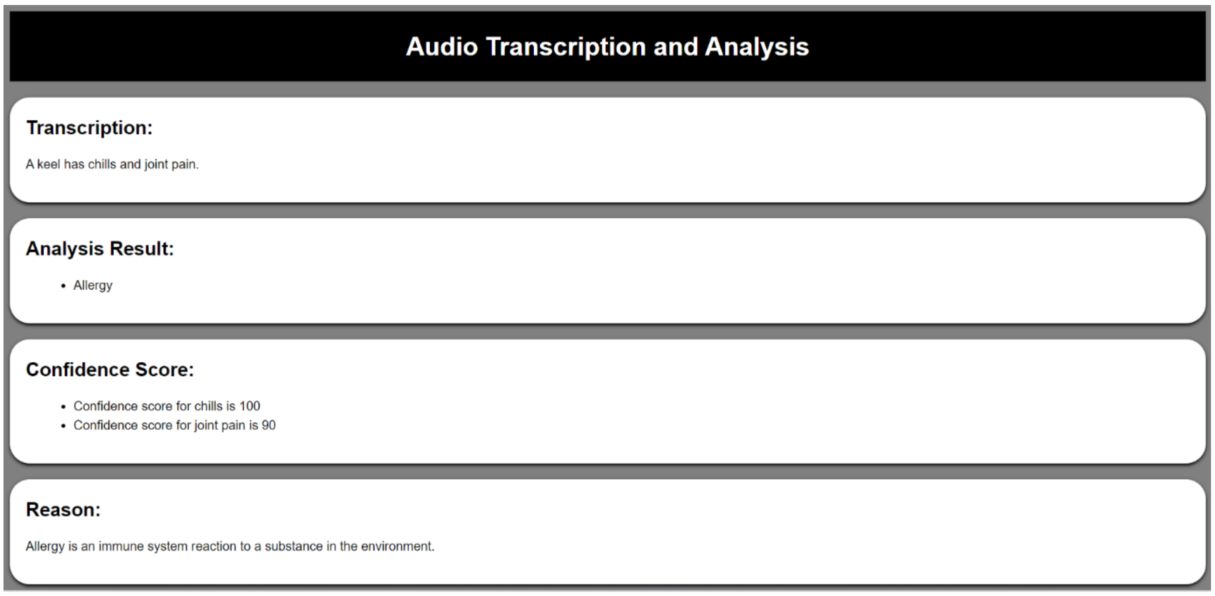


Figure 8
Choosing audio file to upload to server

OUTPUT PAGE:



Audio Transcription and Analysis

Transcription:

A keel has chills and joint pain.

Analysis Result:

- Allergy

Confidence Score:

- Confidence score for chills is 100
- Confidence score for joint pain is 90

Reason:

Allergy is an immune system reaction to a substance in the environment.

Figure 9
Result page of website

CHAPTER 8

REFERENCE

M. Chen, Z. Xu, K. Weinberger, and F. Sha. Marginalized denoising autoencoders for domain adaptation. In Proceedings of the 29th International Conference on Machine Learning, pages 767–774. ACM, 2012.

G. E. Dahl, M. Ranzato, A. Mohamed, and G. E. Hinton. Phone recognition with the mean covariance restricted Boltzmann machine. In Advances in Neural Information Processing Systems 23, pages 469–477, 2010.

O. Dekel, O. Shamir, and L. Xiao. Learning to classify with missing and corrupted features. Machine Learning, 81(2):149–178, 2010.

A. Globerson and S. Roweis. Nightmare at test time: robust learning by feature deletion. In Proceedings of the 23rd International Conference on Machine Learning, pages 353–360. ACM, 2006.

I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. Maxout networks. In Proceedings of the 30th International Conference on Machine Learning, pages 1319–1327. ACM, 2013.

G. Hinton and R. Salakhutdinov. Reducing the dimensionality of data with neural networks. Science, 313(5786):504 – 507, 2006.

G. E. Hinton, S. Osindero, and Y. Teh. A fast learning algorithm for deep belief nets. Neural Computation, 18:1527–1554, 2006.

K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun. What is the best multi-stage architecture for object recognition? In Proceedings of the International Conference on Computer Vision (ICCV'09). IEEE, 2009.

Semwal DK, Mishra SP, Chauhan A, Semwal RB. Adverse health effects of tobacco and role of Ayurveda in their reduction. J Med Sci. 2015;15:139–46. [Google Scholar]

Lad V. Ayurveda, the Science of Self-Healing: A Practical Guide. 2nd ed. New Delhi: Lotus Press; 1987. [Google Scholar]

Jacqui W. Herbal products are often contaminated, study finds. BMJ. 2013;347:f6138. [PubMed] [Google Scholar]

Humber JM. The role of complementary and alternative medicine: Accommodating pluralism. J Am Med Assoc. 2002;288:1655–6. [Google Scholar]

Basisht G. Exploring progression of Ayurveda. Ayu. 2011;32:445–7. [PMC free article] [PubMed] [Google Scholar]

Baghel MS. Need of new research methodology for Ayurveda. Ayu. 2011;32:3–4. [PMC free article] [PubMed] [Google Scholar]

Sharma PV, editor. Charaka Samhita of Agnivesha, Sutra Sthana, Ch. 11, Ver. 2. Varanasi: Choukhamba Orientalia; 1995. p. 114. [Google Scholar]

Patwardhan B. The quest for evidence-based Ayurveda: Lessons learned. Curr Sci. 2012;102:1406–17. [Google Scholar]

Morandi A, Tosto C, Sartori G, Roberti di Sarsina P. Advent of a link between Ayurveda and modern health science: The proceedings of the first international congress on ayurveda, “Ayurveda:

The Meaning of Life-Awareness, Environment, and Health” March 21-22, 2009, Milan, Italy. Evid Based Complement Alternat Med 2011. 2011:929083. [PMC free article] [PubMed] [Google Scholar]

Singh RH. Integrative Medicine, Special Monograph. New Delhi: Choukhamba Surbharti; 2009. [Google Scholar]

Hankey A. The scientific value of Ayurveda. J Altern Complement Med. 2005;11:221–5. [PubMed] [Google Scholar]