A

PROJECT REPORT

ON

Wellness Prediction Suite

Submitted in partial fulfilment of the requirements

For the award of a Degree of

BACHELOR OF ENGINEERING

IN

CSE(AIML)

Submitted By

Kashetty Aneesh **245320748020**

Under the guidance of

Ms. R. Koteswaramma

Assistant Professor

Department of CSE(AIML)

NEIL GOGTE INSTITUTE OF TECHNOLOGY

Kachavanisingaram Village, Hyderabad, Telangana 500058.

JUNE 2024

NEIL GOGTE INSTITUTE OF TECHNOLOGY

A Unit of Keshav Memorial Technical Education (KMTES)

Approved by AICTE, New Delhi & Affiliated to Osmania University, Hyderabad

CERTIFICATE

This is to certify that the Major project work entitled "**Wellness Prediction Suite"** *is a bonafide work carried out by* **Kashetty Aneesh (245320748020)** of **IV** year **VIII** semester **Bachelor of Engineering** *in CSE(AIML) by Osmania University, Hyderabad during the academic year* **2023-2024** *is a record of bonafide work carried out by him. The results embodied in this report have not been submitted to any other University or Institution for the award of any degree.*

Internal Guide	**Head of Department**
Ms. R. Koteswaramma	Dr. T. Prem Chander

External

NEIL GOGTE INSTITUTE OF TECHNOLOGY
A Unit of Keshav Memorial Technical Education (KMTES)
Approved by AICTE, New Delhi & Affiliated to Osmania University

DECLARATION

I hereby declare that the Major Project Report entitled, "**Wellness Prediction Suite**" submitted for the B.E degree is entirely our work and all ideas and references have been duly acknowledged.

It does not contain any work for the award of any other degree.

Date:

Kashetty Aneesh **245320748020**

ACKNOWLEDGEMENT

We are happy to express our deep sense of gratitude to the **Principal of the college, Prof. R. Shyam Sunder**, Neil Gogte Institute of Technology, for having provided us with adequate facilities to pursue our project.

We would like to thank **Dr. T. Prem Chander**, **Head of the Department,** CSE(AIML) , Neil Gogte Institute of Technology, for having provided the freedom to use all the facilities available in the department, especially the laboratories and the library.

We would also like to thank my internal guide **MS. R. Koteswaramma**, **Assistant Professor** for her technical guidance & constant encouragement.

We sincerely thank our seniors and all the teaching and non-teaching staff of the Department of CSE(AIML) for their timely suggestions, healthy criticism, and motivation during this work.

Finally, we express our immense gratitude with pleasure to the other individuals who have either directly or indirectly contributed to our need at the right time for the development and success of this work.

ABSTRACT

The Wellness Predictor Suite represents a significant advancement in healthcare technology, offering acomprehensive platform for disease prediction and risk assessment. Developed using Flask, a Python- based web framework, this machine learning web application harnesses the power of predictive analytics to evaluate an individual's susceptibility to various medical conditions. With a focus on disease prevention and early intervention, the suite encompasses predictive models for a diverse array of ailments, including diabetes, heart disease, Pneumonia, Indian liver disease, kidney disease and brain stroke.At its core, the suite utilizes sophisticated machine learning algorithms trained on extensive datasets toanalyze input data provided by users. By inputting relevant medical information and health indicators, users can obtain personalized predictions regarding their likelihood of developing specific diseases. This predictive capability empowers individuals to make informed decisions about their health and take proactive measures to mitigate risks and improve outcomes.

TABLE OF CONTENTS

List of Figures

List of Tables

CHAPTER – 1

INTRODUCTION

1.1 PROBLEM STATEMENT

Optimal health maintenance is vital, yet predicting potential health issues remains challenging due to the human body's complexity. From chronic diseases like diabetes to acute conditions like stroke, early detection is crucial for effective management. The Wellness Predictor Suite addresses this by using machine learning to provide personalized predictions for various diseases. By analyzing diverse health indicators, users can take proactive measures to mitigate risks. With its user-friendly interface, the suite enables easy input of health data, empowering individuals to make informed decisions and prioritize preventive care, ultimately enhancing overall well-being.

1.2 MOTIVATION

The Wellness Predictor Suite is driven by a fundamental desire to revolutionize healthcare by providing individuals with personalized predictive insights into their health status. By harnessing the power of machine learning and predictive analytics, the suite aims to empowerindividuals to take proactive measures to safeguard their well-being. With the ability to predict various diseases and health conditions, the suite not only enhances individual health outcomes but also has the potential to transform healthcare on a broader scale. By promoting preventive care and early intervention, the suite seeks to improve overall health outcomes, reduce healthcare costs, and ultimately enhance quality of life for individuals.

1.3 SCOPE

The scope of the Wellness Predictor Suite encompasses leveraging machine learning algorithms to predict various diseases and health conditions, including diabetes, heart disease, cancer, and more. By integrating user-friendly interfaces and predictive analytics, the suite enables individuals to input their health data and receive personalized predictions in real-time. The project aims to streamline preventive care measures, enhance health outcomes, and promote a culture of proactive health management. Through its scalable and efficient approach, the suite has the potential to revolutionize healthcare delivery and improve overall well-being.

Pre-processing Phase:

- **Image Data:** The dataset undergoes normalization using statistical techniques such as Mean, Median, and Mode to standardize the input data. This ensures that the neural network receives consistent and reliable data for training.
- **Patient Health Data:** Normalize patient health data using statistical techniques such as Mean, Median, and Mode to standardize input data. This ensures consistency and reliability for training the predictive models.

Training Phase:

- **Training Multiple Models:**
    - Utilize pre-processed data to train various machine learning models employing diverse algorithms such as Decision Trees, Random Forest, Support Vector Machines (SVM), and Many more.
    - Each model is trained to capture different aspects and nuances of the data, ensuring a comprehensive exploration of the predictive space.

- **Performance Evaluation:**
    - Evaluate the performance of each trained model using appropriate metrics such as accuracy, precision, recall, and F1-score.Conduct cross-validation techniques to assess model generalization and robustness across different subsets of the data.

Classification Phase:

- **Disease Prediction:**
    - Utilize the selected model to predict disease likelihood based on input health data.
    - Present users with personalized insights into potential health risks, leveraging the accuracy of the chosen model.

- **Anomaly Detection:**
    - Employ anomaly detection techniques to identify any irregularities or outliers in the prediction results.
    - Provide users with alerts or notifications for any unusual health predictions, ensuring proactive health management.

1.4 OUTLINE

The Wellness Predictor Suite project is structured into three key phases:

- Data Collection and Processing: Acquire and preprocess diverse health data, including medicalrecords and vital signs.
- Model Development and Training: Utilize machine learning algorithms to develop predictivemodels for various diseases and health conditions.
- Deployment and User Interface: Integrate predictive models into a user-friendly web interface, enabling individuals to input their health data and receive personalized predictions in real-time.

This structured approach ensures efficient development, deployment, and usability of the suite, facilitating proactive health management and improved well-being.

CHAPTER – 2

LITERATURE SURVEY

EXISTING SYSTEM:

Existing systems in conventional healthcare practices often rely on subjective assessments and manual data analysis, leading to inefficiencies and potential errors. With the increasing volume of health data, there is a needfor automated systems to assist in predictive analysis and risk assessment. Currently, individuals have limited access to personalized predictive insights into their health status, hindering proactive health management. The absence of user-friendly interfaces further complicates the utilization of predictive analytics in healthcare. There's a clear need for a comprehensive wellness prediction suite that integrates machine learning algorithms and user-friendly interfaces to empower individuals with personalized health predictions and promote proactive health management

LIMITATIONS OF EXISTING SYSTEM:

- **Subjective Assessments and Manual Analysis:** Traditional healthcare relies heavily on subjective assessments and manual data analysis, leading to inefficiencies and potential errors in diagnosis and treatment.

- **Growing Volume of Health Data:** The exponential increase in health data volume necessitates automated systems for predictive analysis and risk assessment, as manual methods struggle to keep pace with the influx of information.

- **Limited Access to Personalized Insights:** Individuals currently face barriers in accessing personalized predictive insights into their health status, impeding their ability to proactively manage their well-being.

- **Complexity of Utilizing Predictive Analytics:** The absence of user-friendly interfaces complicates the integration and utilization of predictive analytics in healthcare, hindering its adoption and effectiveness.

- **Need for Comprehensive Wellness Prediction Suite:** There's a pressing need for a comprehensive wellness prediction suite that seamlessly integrates machine learning algorithms with user-friendly interfaces. Such a suite would empower individuals with personalized health predictions, facilitating proactive health management and improving overall well-being.

PROPOSED SYSTEM:

The envisioned wellness prediction suite will leverage advanced machine learning techniques to provide personalized health predictions to users. Data collection will involve gathering diverse health data sources, including medical records and lifestyle information. Machine learning algorithms will be employed to develop predictive models for various diseases and health conditions, ensuring accuracy and reliability. The suite will feature user-friendly interfaces for individuals to input their health data and receive personalized predictions in real-time. Continuous monitoring and updates will ensure the accuracy and relevance of predictions over time, empowering individuals to proactively manage their health.

ADVANTAGES OF PROPOSED SYSTEM:

• **Personalized Predictions:** Machine learning algorithms offer tailored health predictions based onindividual data.

• **Proactive Health Management:** Users can take preventive measures based on predictive insights,improving overall well-being.

• **User-Friendly Interface:** Intuitive design facilitates easy input of health data and interpretation ofpredictions.

• **Continuous Updates:** Regular model updates ensure predictions remain accurate and relevant overtime.

• **Efficient Resource Utilization:** Maximizes computational resources, optimizing performance andscalability.

• **Real-Time Monitoring:** Enables users to monitor their health status and receive timelyrecommendations.

• **Enhanced Decision-Making:** Empowers individuals to make informed decisions about their healthand lifestyle choices.

CHAPTER – 3

SOFTWARE REQUIREMENTS SPECIFICATION

3.1 Overall Description:

- **Disease Prediction:** Machine learning models, including decision trees, random forests, and support vector machines (SVM), are utilized to predict the likelihood of various diseases based on input health data. The selected model identifies anomalies by comparing predicted disease likelihoods with expected values, flagging deviations from normal patterns as potential health concerns.

- **Dietary Recommendation:** Leveraging nutritional data and user preferences, personalized dietary recommendations are provided based on predicted disease likelihoods and dietary guidelines. The recommendation system identifies anomalies by detecting significant deviations from expected dietary recommendations, signalling potential dietary risks or deficiencies.

3.2. Operating Environment:

HARDWARE REQUIREMENTS

• **Processor**	:	Intel Core™ i3 Processor (Min)
• **Speed**	:	2.9 GHz (Min)
• **RAM**	:	4 GB (Min)
• **Hard Disk**	:	64 GB (Min)

SOFTWARE REQUIREMENTS

• **Operating System**	:	Windows 7 (Min)
• **Front End**	:	HTML, CSS, JS (Min)
• **Back End**	:	Python
• **Database**	:	Spreadsheets (Min)

3.3 Functional Requirements:

The system must accurately predict the likelihood of various diseases using machine learning techniques. For disease prediction, the system utilizes machine learning algorithms such as decision trees, random forests, and support vector machines (SVM) to analyse input health data and determine the probability of different diseases. The system should provide users with clear visualizations and insights into the predicted disease likelihoods, facilitating informed decision-making and proactive health management. It must efficiently handle large datasets to support real-time processing and analysis of health data. Additionally, the system should incorporate user feedback mechanisms to enhance prediction accuracy and user experience over time. The interface should be intuitive, offering clear instructions and visual representations of the predicted disease likelihoods to support user understanding and engagement. Additionally, the system may include static information such as a healthy diet and hospital recommendations, provided through external links for reference.

User Characteristics:

The primary users of this Wellness Prediction Suite are healthcare professionals, wellness enthusiasts, and individuals seeking personalized health insights. These users encompass medical practitioners seeking diagnostic support, researchers exploring health trends, and individuals striving for proactive well-being management. Users possess varying levels of expertise in healthcare and wellness, necessitating a system that offers accurate predictions and an intuitive, user-friendly interface. Additionally, users prioritize accuracy, efficiency, and the ability to handle diverse health datasets, highlighting the importance of robust preprocessing, model training, and prediction functionalities. The suite is tailored to meet these diverse user needs by providing comprehensive documentation, streamlined workflows, and scalable, high-performance processing to support informed decision-making and proactive health management.

Admin Functionality:

- Admin plays a major role in this project.
- Adding new training data to improve the system's accuracy and reliability.
- Monitoring system performance and making necessary adjustments to the prediction algorithms.
- Verifying the predictions and ensuring the system's output is accurate and reliable.
- Updating the system with new algorithms or improvements to enhance its prediction capabilities.
- Providing support and guidance to users, ensuring they can effectively utilize the system for their disease prediction needs.

3.4 Non-Functional Requirements:

3.4.1 Performance Requirements:

Performance requirements refer to static numerical requirements placed on the interaction between the users and the software.

***Response Time*:**

Average response time shall be less than 10 sec.

Recovery Time:

In case of system failure, the redundant system shall resume operations within 30 secs. Average repair time shall be less than 45 minutes.

Start-Up/Shutdown Time:

The system shall be operational within 1 minute of starting up.

Utilization of Resources:

The system shall store in the database no more than 200 different colleges with room for improvement.

3.4.2 Safety Requirements:

-NA-

3.4.3 Security Requirements:

The model will be running on a secure website i.e., an HTTPS website and also on a secure browser such as Google Chrome, Brave, etc.

3.4.4 Software Quality Attributes: ***Reliability:***

The system shall be reliable i.e., in case the webpage crashes, progress will be saved.

Availability:

The website will be available to all its users round the clock i.e., they can access the website at any time.

Security:

The model will be running on a secure website i.e., an HTTPS website, and on a secure browser such as Google Chrome, Brave, etc.

Maintainability:

The model shall be designed in such a way that it will be very easy to maintain in the future. Our model is a web-based system and will depend much on the web server. However, the web application will be designed using proper database modeling along with extensive documentation which will make it easy to develop, troubleshoot, and maintain in the future.

Usability:

The interfaces of the system will be user friendly enough that every user will be able to use it easily.

Scalability:

The system will be designed in such a way that it will be extendable. If more species or algorithms are going to be added in the system, then it would easily be done.

The same system can also be developed to become a mobile application rather than just a website.

CHAPTER – 4

SYSTEM DESIGN

Use case Diagram:

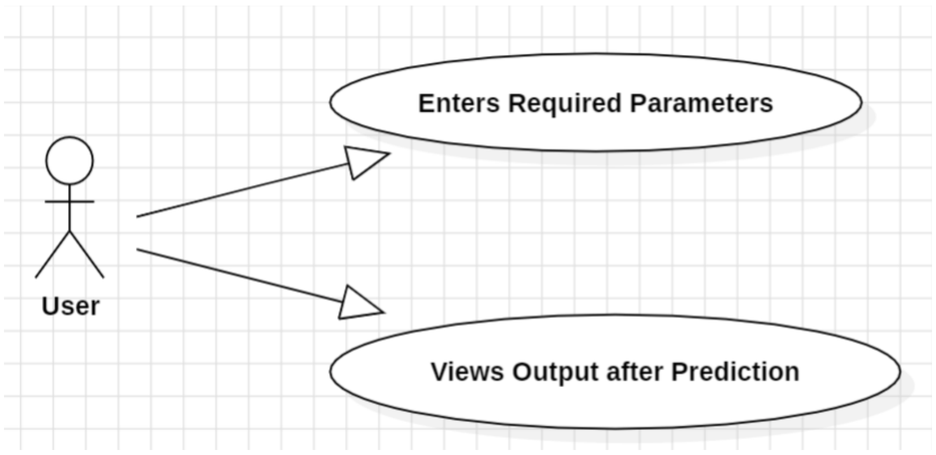


fig 4.1: Use case diagram for User

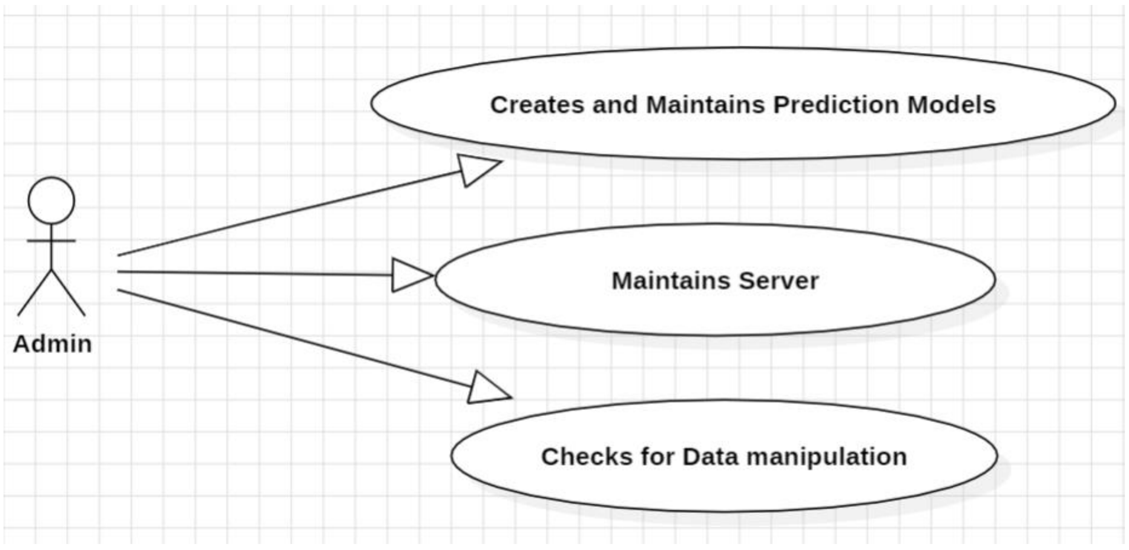

Creates and Maintains Prediction Models
Admin
Maintains Server
Checks for Data manipulation

fig 4.2: Use case diagram for Admin

Class Diagram:

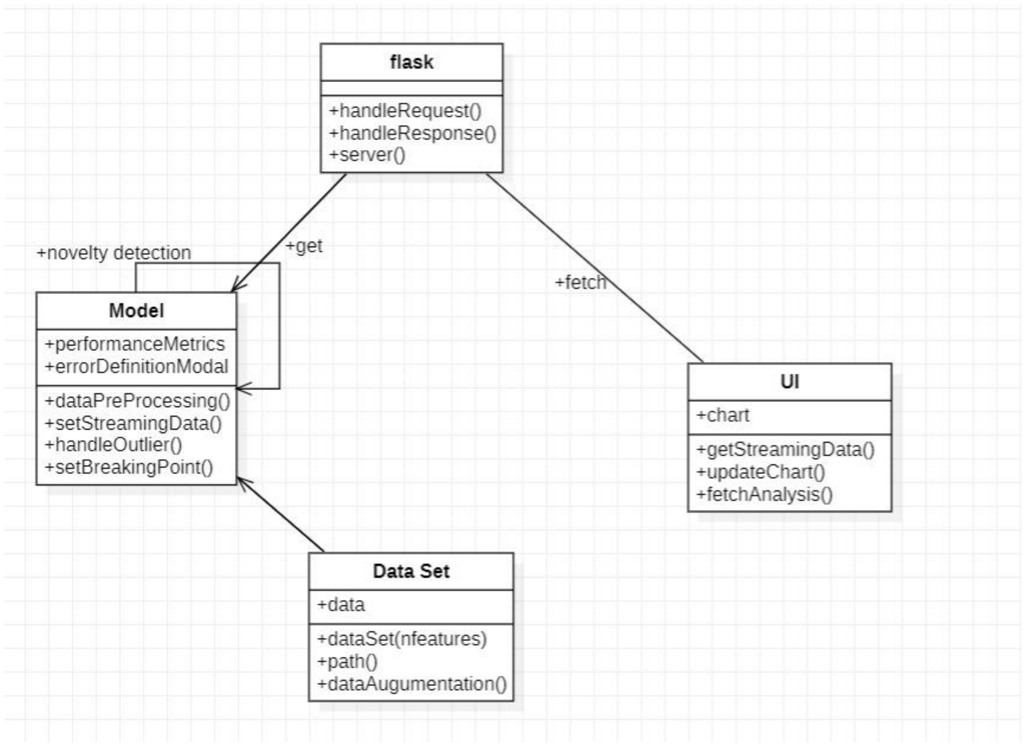


Fig.4.3: Class diagram

Sequential Diagram:

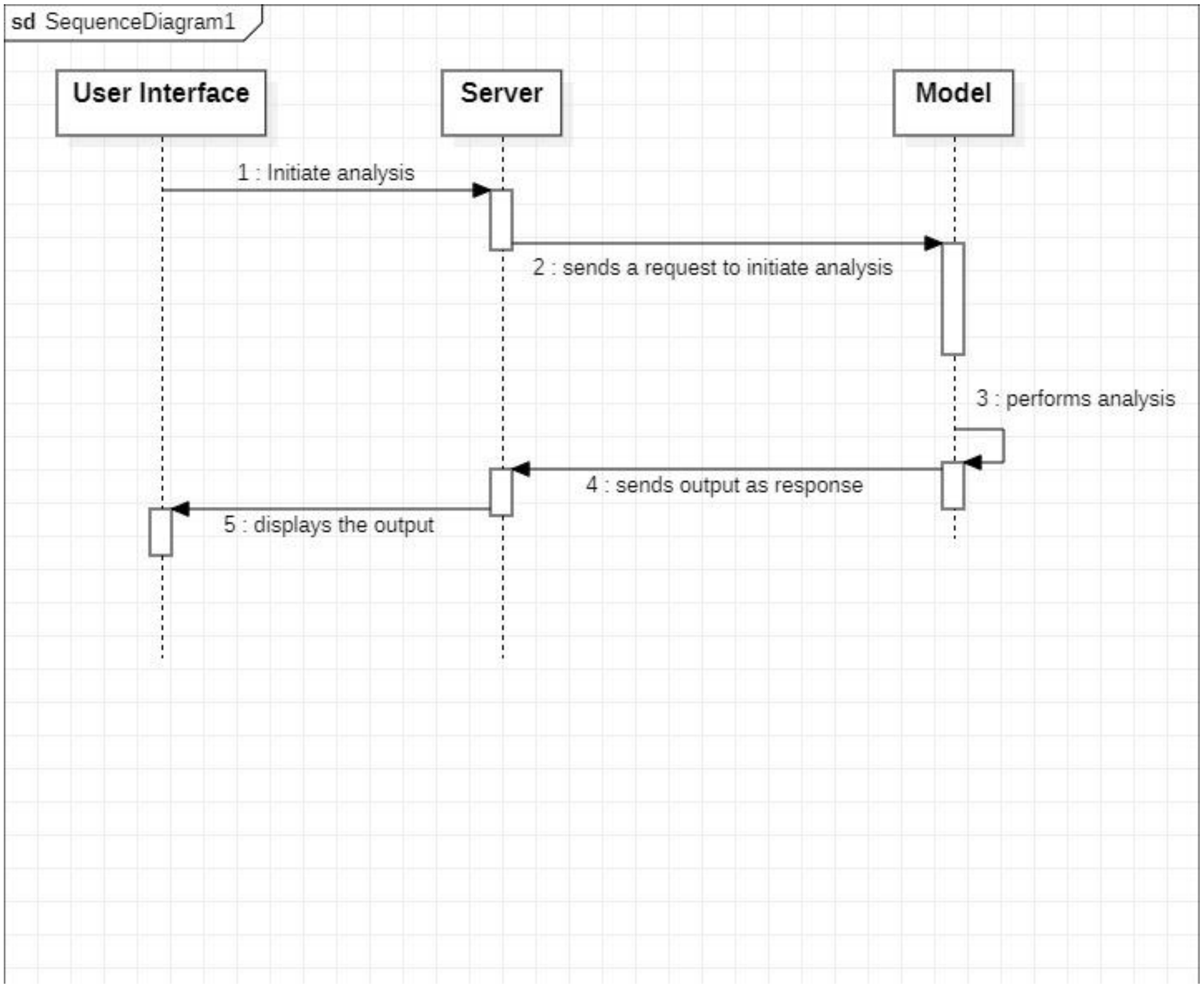


Fig4.4 : Sequence Diagram

CHAPTER – 5

IMPLEMENTATION

5.1 COMPONENTS

1. Disease Prediction::

- **Data Preparation and Preprocessing**: Health datasets are meticulously pre-processed to normalize values and handle missing data, ensuring a clean and standardized input for model training.

- **Model Architecture Definition:** A Various machine learning models, such as Decision Trees, Random Forests, and Support Vector Machines (SVM), are defined and evaluated to determine the best architecture for each disease prediction.

- **Model Training and Fine-Tuning:** Selected models undergo rigorous training and fine-tuning, leveraging cross-validation and hyperparameter tuning to optimize performance and extract crucial features for accurate disease prediction.

- **Performance Evaluation and Selection:** Model performance is evaluated using metrics such as accuracy, precision, recall, and F1-score. The model with the highest accuracy and robust performance across different metrics is selected for deployment.

- **Prediction and Thresholding:** The selected model calculates disease likelihoods based on input health data, with specific thresholds established to categorize the risk levels (e.g., low, moderate, high).

- **Visualization and Storage:** Predicted disease risks are visually represented in a user-friendly interface, providing clear insights and actionable information. Results are systematically stored for subsequent analysis and tracking.

2. **Recommendation System:**

- **Healthy Diet Information:** Provide users with a static, evidence-based healthy diet plan to support general well-being. This information is easily accessible through the user interface.

- **Hospital Recommendation:** Utilize external links, such as JustDial, to offer users recommendations for nearby hospitals and healthcare facilities, ensuring they have access to relevant healthcare services when needed.

3. **User Feedback Mechanisms:**

- **Feedback Collection:** Implement user feedback forms and mechanisms to gather user input on prediction accuracy and overall user experience.

- **Continuous Improvement:** Use collected feedback to continuously refine and improve the prediction models and user interface, enhancing the system's accuracy and user satisfaction over time.

5.2 CODE

Predictions.py

```
from flask import Blueprint, render_template, request, send_from_directory
from .app_functions import ValuePredictor, pred
import os
from werkzeug.utils import secure_filename

prediction = Blueprint('prediction', __name__)

UPLOAD_FOLDER = 'uploads'
STATIC_FOLDER = 'static'

dir_path = os.path.dirname(os.path.realpath(__file__))



@prediction.route('/predict', methods=["POST", 'GET'])
def predict():

    if request.method == "POST":
        to_predict_list = request.form.to_dict()
        to_predict_list = list(to_predict_list.values())
        to_predict_list = list(map(float, to_predict_list))
        result, page = ValuePredictor(to_predict_list)
        return render_template("result.html", prediction=result, page=page)
    else:
        return render_template( 'base.html')
```

```
@prediction.route('/upload', methods=['POST','GET'])
def upload_file():
    if request.method=="GET":
        return render_template('pneumonia.html', title='Pneumonia Disease')
    else:
        file = request.files["file"]
        basepath = os.path.dirname(__file__)
        file_path = os.path.join(basepath,'uploads',  secure_filename(file.filename))
        file.save(file_path)
        indices = {0: 'Pneumonia', 1:'Normal' }
        result = pred(file_path)


        if result>0.5:
            label = indices[1]
            accuracy = result * 100
        else:
            label = indices[0]
            accuracy = 100 - result
        return render_template('deep_pred.html', image_file_name=file.filename, label = label,
accuracy = accuracy)

@prediction.route('/uploads/<filename>')
def send_file(filename):
    return send_from_directory(UPLOAD_FOLDER, filename)
```

Views.py

```
from flask import Blueprint, render_template


views = Blueprint('views', __name__)


@views.route("/")
def home():
    return render_template(r'base.html')
@views.route("/kidney")
def kidney():
    return render_template(r'kidney_index.html')


@views.route("/kidney_form")
def kidney_form():
    return render_template(r'kidney.html')
@views.route("/liver")
def liver():
    return render_template(r'liver_index.html')


@views.route("/liver_form")
def liver_form():
    return render_template(r'liver.html')


@views.route("/heart")
def heart():
    return render_template(r'heart_index.html')
```

```
@views.route("/heart_form")
def heart_form():
    return render_template(r'heart.html')


@views.route("/stroke")
def stroke():
    return render_template(r'stroke_index.html')


@views.route("/stroke_form")
def stroke_form():
    return render_template(r'stroke.html')
@views.route("/diabete")
def diabete():
    return render_template(r'diabete_index.html')


@views.route("/diabete_form")
def diabete_form():
    return render_template(r'diabete.html')
@views.route("/pneumonia")
def pneumonia():
    return render_template(r'pneumonia_index.html')


@views.route("/pneumonia_form")
def pneumonia_form():
    return render_template(r'pneumonia.html')
```

Pneumonia.ipynb

```
import os
os.environ['KAGGLE_CONFIG_DIR'] = '/content'
!kaggle datasets download -d paultimothymooney/chest-xray-pneumonia
import zipfile

z= zipfile.ZipFile('chest-xray-pneumonia.zip')
z.extractall()
import os, shutil
import random
import numpy as np
from PIL import Image
import pandas as pd
import itertools
from tqdm import tqdm, tqdm_notebook
import cv2
from scipy import stats
from sklearn.metrics import confusion_matrix, roc_curve,auc, classification_report,
precision_score, recall_score
from sklearn.linear_model import LinearRegression

import skimage
import skimage.segmentation
import copy
import matplotlib.pyplot as plt
%matplotlib inline
plt.style.use('ggplot')
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.image as mpimg
```

```
%tensorflow_version 2.x
import tensorflow as tf
device_name = tf.test.gpu_device_name()
if device_name != '/device:GPU:0':
  raise SystemError('GPU device not found')
print('Found GPU at: {}'.format(device_name))

print(tf.__version__)
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Model
from tensorflow.keras.models import load_model
from tensorflow.keras.layers import Input, Flatten, Dense, MaxPooling2D, Conv2D, Dropout
from tensorflow.keras.applications.vgg19 import VGG19
from tensorflow.keras.optimizers import SGD, RMSprop, Adam
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping, ReduceLROnPlateau
from tensorflow.keras.applications.imagenet_utils import decode_predictions
labels = ['PNEUMONIA', 'NORMAL']
img_size = 150
def get_data(data_dir):
    data = []
    max_length = 0  # Keep track of the maximum length of inner lists

    for label in labels:
        path = os.path.join(data_dir, label)
        class_num = labels.index(label)
        for img in os.listdir(path):
            try:
                img_path = os.path.join(path, img)
                img_arr = Image.open(img_path).convert('L')
                resized_arr = img_arr.resize((img_size, img_size))

                # Append the resized array and class number to the data list
                data.append([resized_arr.tobytes(), class_num])
            # Update the maximum length of inner lists
            max_length = max(max_length, len(data[-1][0]))
        except Exception as e:
            print(f"Error processing image {os.path.join(path, img)}: {e}")

    # Pad the inner lists with zeros to make them all the same length
    for i in range(len(data)):
        data[i][0] = data[i][0].ljust(max_length, b'\x00')

    return np.array(data)

train = get_data('chest_xray/train')
test = get_data('chest_xray/test')
val = get_data('chest_xray/val')

pneumonia = os.listdir("chest_xray/train/PNEUMONIA")
pneumonia_dir = "chest_xray/train/PNEUMONIA"

plt.figure(figsize=(20, 10))

for i in range(9):
    plt.subplot(3, 3, i + 1)
    img = plt.imread(os.path.join(pneumonia_dir, pneumonia[i]))
    plt.imshow(img, cmap='gray')
    plt.axis('off')
    plt.title("Pneumonia X-Ray")
plt.tight_layout()

train_datagen = ImageDataGenerator(rescale=1./255.,
                                   horizontal_flip=0.4,
                                   vertical_flip=0.4,
                                   rotation_range=40,
                              shear_range=0.2,
                              width_shift_range=0.4,
                              height_shift_range=0.4,
                              fill_mode='nearest')
valid_datagen = ImageDataGenerator(rescale=1.0/255.)
test_datagen = ImageDataGenerator(rescale=1.0/255.)

train_generator = train_datagen.flow_from_directory(train_dir,
                                                    batch_size=32,
                                                    target_size=(128,128),
                                                    class_mode='categorical',
                                                    shuffle=True,
                                                    seed=42,
                                                    color_mode='rgb')

valid_generator = valid_datagen.flow_from_directory(valid_dir,
                                                    batch_size=32,
                                                    target_size=(128, 128),
                                                    class_mode='categorical',
                                                    shuffle=True,
                                                    seed=42,
                                                    color_mode='rgb')

class_labels = train_generator.class_indices
class_names = {value:key for (key, value) in class_labels.items()}



# Build the network
base_model = VGG19(include_top=False, input_shape=(128,128,3))
x = base_model.output
flat=Flatten()(x)
class_1 = Dense(4608, activation='relu')(flat)
```

```
drop_out = Dropout(0.2)(class_1)
class_2 = Dense(1152, activation='relu')(drop_out)
output = Dense(2, activation='softmax')(class_2)
model_03 = Model(base_model.inputs, output)

# Load weights
model_03.load_weights('model_weights/vgg19_model_02.h5')

sgd = SGD(learning_rate=0.0001, momentum=0.9, nesterov=True)

# Legacy Keras Optimizer with decay
legacy_sgd = LegacySGD(learning_rate=0.0001, decay=1e-6, momentum=0.9, nesterov=True)
# Compile the model
model_03.compile(optimizer=sgd, loss='categorical_crossentropy', metrics=['accuracy'])
history_03 = model_03.fit(train_generator,
                          steps_per_epoch=100,
                          epochs=35,
                          callbacks = [es, cp, lrr],
                          validation_data = valid_generator)



# save model
if not os.path.isdir('model_weights/'):
    os.mkdir('model_weights/')
model_03.save_weights(filepath='model_weights/vgg_unfrozen.h5', overwrite=True)
# Load the saved model
model_03.load_weights('model_weights/vgg_unfrozen.h5')
# Evaluate the model on the hold out validation and test datasets

vgg_val_eval_03 = model_03.evaluate(valid_generator)
vgg_test_eval_03 = model_03.evaluate(test_generator)

print('Validation loss     :{0:.4f}'.format(vgg_val_eval_03[0]))
print('Validation accuracy :{0:.4f}'.format(vgg_val_eval_03[1]))
print('Test loss           :{0:.4f}'.format(vgg_test_eval_03[0]))
print('Test accuracy       :{0:.4f}'.format(vgg_test_eval_03[1]))


filenames = test_generator.filenames
nb_samples = len(filenames)
vgg_predictions_03 = model_03.predict(test_generator,
                                      steps = nb_samples,
                                      verbose=1)
vgg_pred_labels_03 = np.argmax(vgg_predictions_03, axis=1)


# Classification Report
print(classification_report(test_generator.classes, vgg_pred_labels_03,
                            target_names=['healthy', 'infected']))
vgg_conf_mat_03 = pd.DataFrame(confusion_matrix(test_generator.classes, vgg_pred_labels_03),
                        index=['NORMAL', 'PNEUMONIA'],
                        columns=['NORMAL', 'PNEUMONIA'])


fig, ax = plt.subplots(figsize=(5,5))

sns.heatmap(vgg_conf_mat_03, annot=True, fmt=".1f", linewidths=0.5, square=True,
cmap='Purples')
ax.set_ylabel("Actual Label", fontsize=14)
ax.set_xlabel("Predicted Label", fontsize=14)
all_sample_title="Accuracy Score: {0:.2f}".format(vgg_test_eval_03[1])
ax.set_title(all_sample_title, size=15)
ax.set_ylim(len(vgg_conf_mat_03)-0.05, -0.05)
plt.tight_layout()
```

CHAPTER – 6

TESTING

6.1 TEST CASES

Test Case to check whether the required Software is installed on the systems

Test Case ID:	1
Test Case Name:	Required Software Testing
Purpose:	To check whether the required Software is installed on the systems
Input:	Enter python command
Expected Result:	Should Display the version number for the python
Actual Result:	Displays python version
Failure	If the python environment is not installed, then the Deployment fails

Table 6.1.1 python Installation verification

Test Case to check Program Integration Testing

Test Case ID:	2
Test Case Name:	Programs Integration Testing
Purpose:	To ensure that all the modules work together
Input:	All the modules should be accessed.
Expected Result:	All the modules should be functioning properly.
Actual Result:	All the modules should be functioning properly.
Failure	If any module fails to function properly, the implementation fails.

Table 6.1.2 python Programs Integration Testing

Case to Collect Dataset and Load the Dataset

Test Case ID:	3
Test Case Name:	Can upload image for pneumonia detection
Purpose:	To check if image can be uploaded
Input:	Upload relevant x-ray image
Expected Result:	Displays the uploaded image
Actual Result:	Uploaded image displayed
Failure	If the image is not displayed, wrong image is uploaded.

Table 6.1.3 Collect image and Load the image

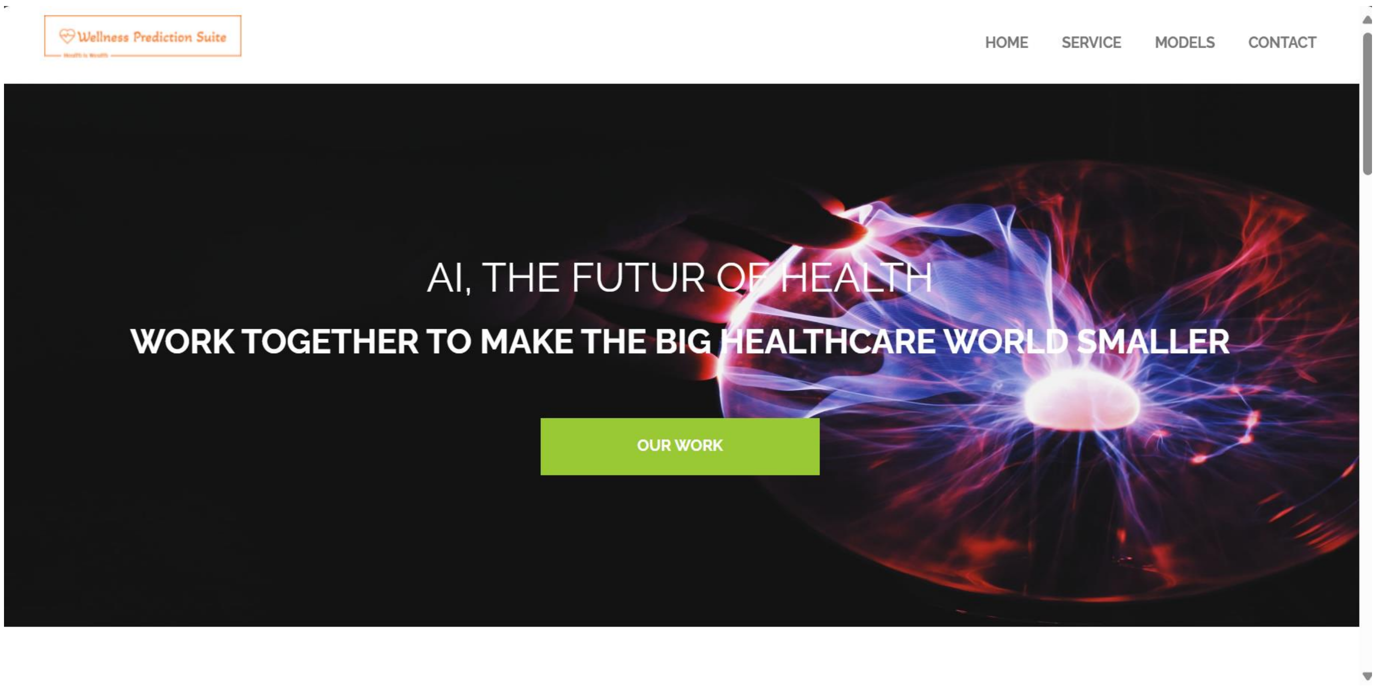
Test Case to check whether the response is appropriate

Test Case ID:	4
Test Case Name:	Redirecting to external links
Purpose:	Verifying the function of the buttons
Input:	Select the recommendations button
Expected Result:	Redirect to other page
Actual Result:	Response related to the query is displayed
Failure	If error 404 occurs, the page redirected to is not present

Table 6.1.4 Functionality test

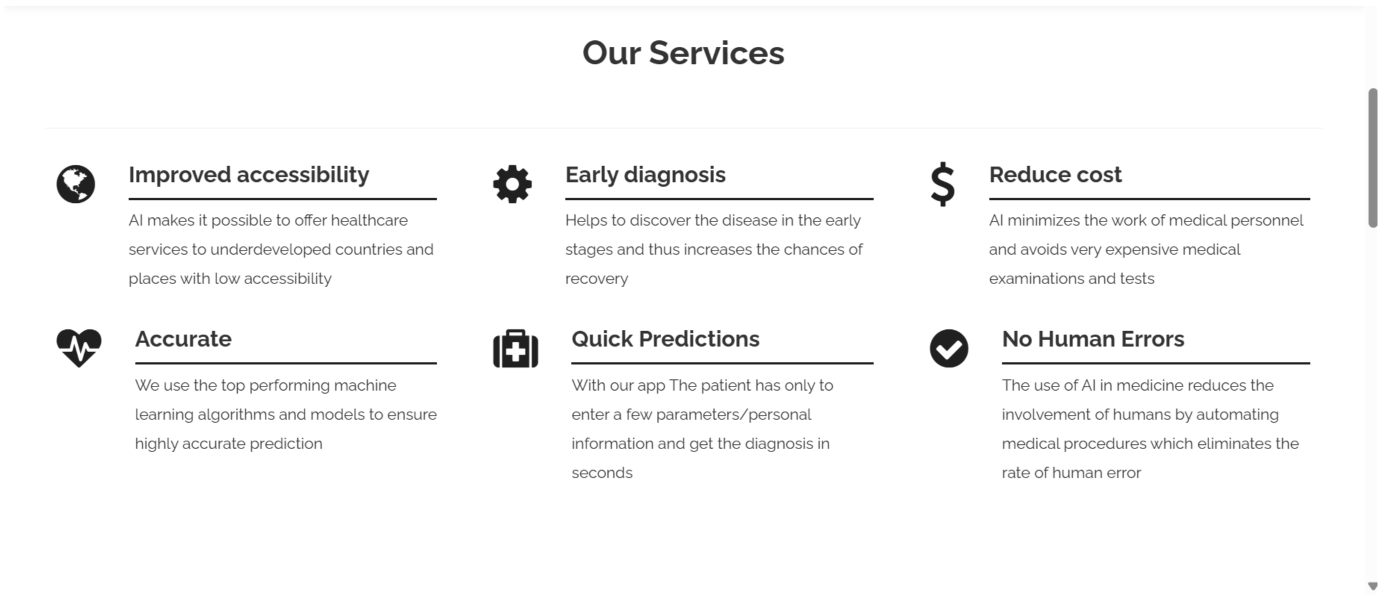
CHAPTER – 7

SCREENSHOTS



AI, THE FUTUR OF HEALTH

WORK TOGETHER TO MAKE THE BIG HEALTHCARE WORLD SMALLER



Our Services

Improved accessibility

AI makes it possible to offer healthcare services to underdeveloped countries and places with low accessibility

Early diagnosis

Helps to discover the disease in the early stages and thus increases the chances of recovery

Reduce cost

AI minimizes the work of medical personnel and avoids very expensive medical examinations and tests

Accurate

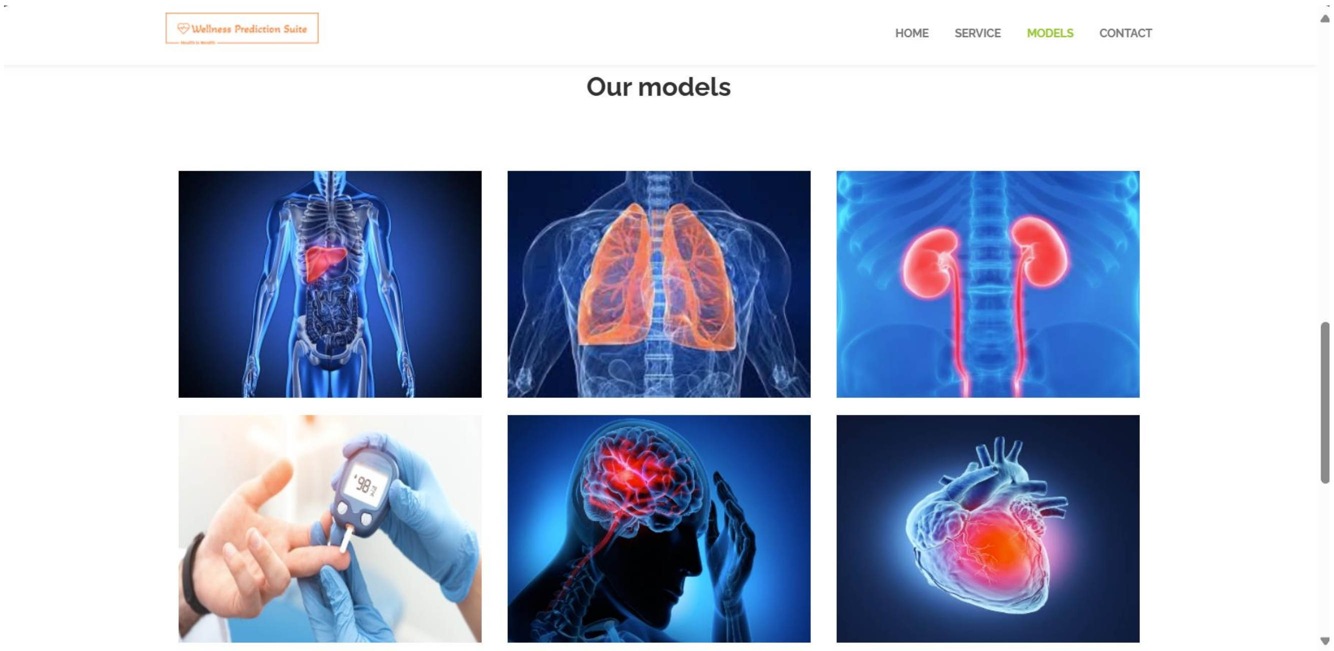
We use the top performing machine learning algorithms and models to ensure highly accurate prediction

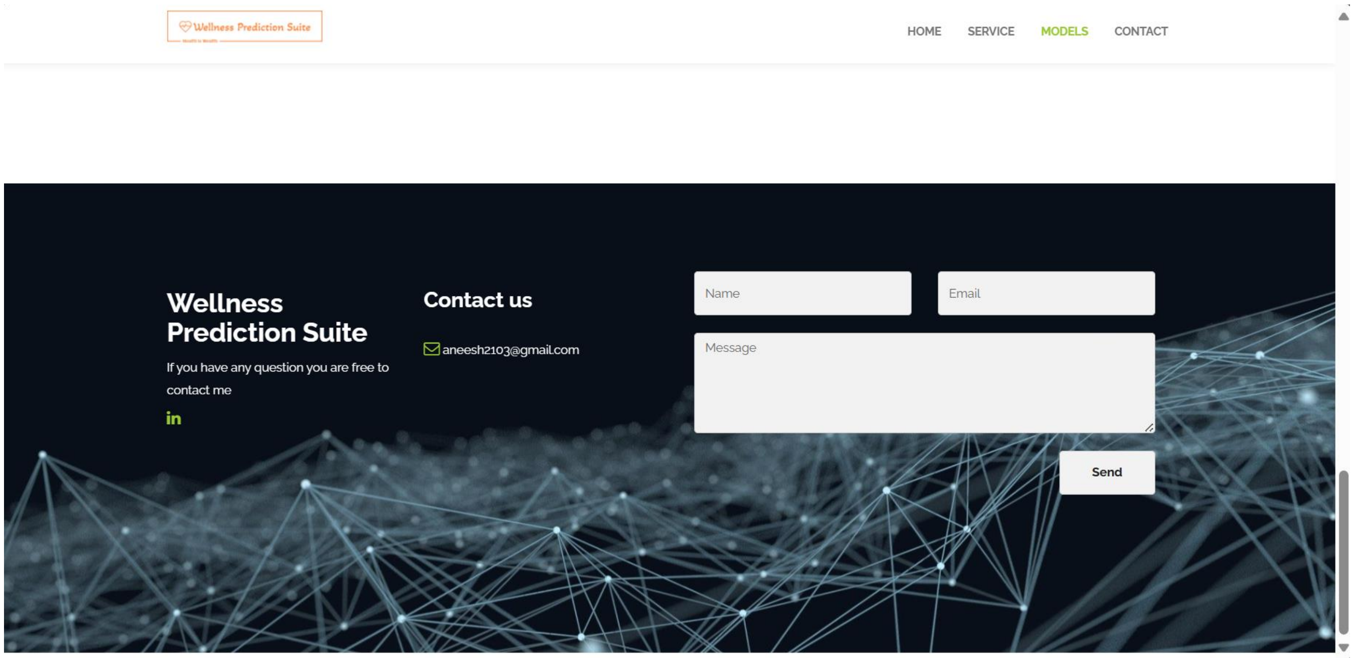
Quick Predictions

With our app The patient has only to enter a few parameters/personal information and get the diagnosis in seconds

No Human Errors

The use of AI in medicine reduces the involvement of humans by automating medical procedures which eliminates the rate of human error



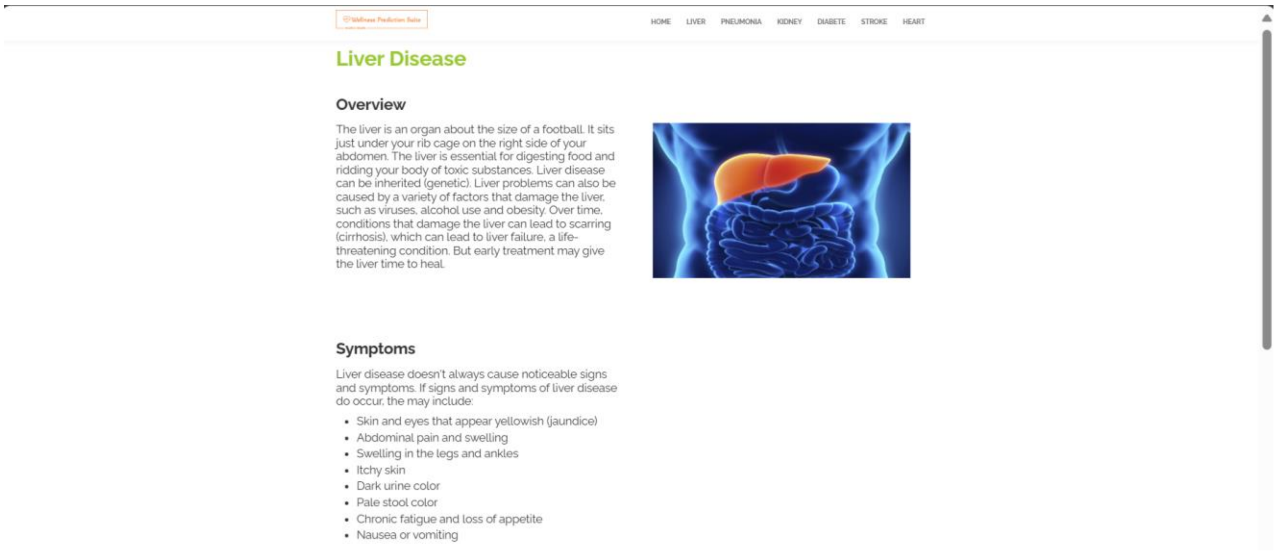


Wellness Prediction Suite

If you have any question you are free to contact me

Contact us

aneesh2103@gmail.com



Liver Disease

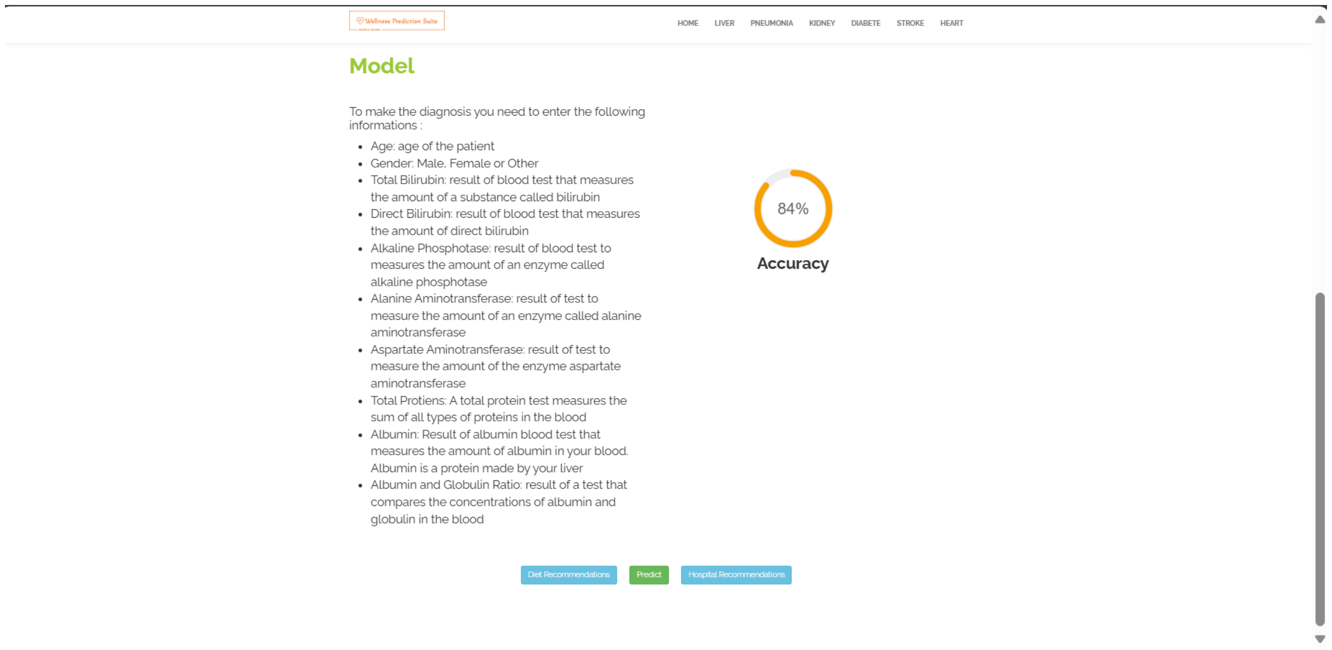
Overview

The liver is an organ about the size of a football. It sits just under your rib cage on the right side of your abdomen. The liver is essential for digesting food and ridding your body of toxic substances. Liver disease can be inherited (genetic). Liver problems can also be caused by a variety of factors that damage the liver, such as viruses, alcohol use and obesity. Over time, conditions that damage the liver can lead to scarring (cirrhosis), which can lead to liver failure, a life-threatening condition. But early treatment may give the liver time to heal.

Symptoms

Liver disease doesn't always cause noticeable signs and symptoms. If signs and symptoms of liver disease do occur, the may include:

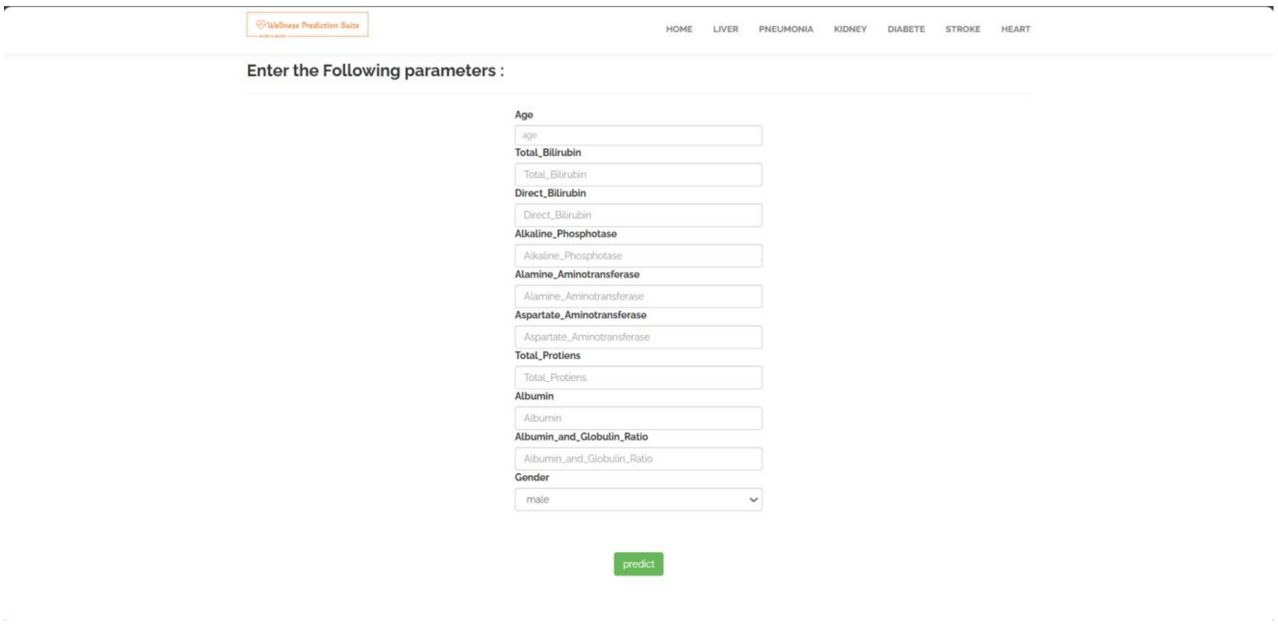
- Skin and eyes that appear yellowish (jaundice)
- Abdominal pain and swelling
- Swelling in the legs and ankles
- Itchy skin
- Dark urine color
- Pale stool color
- Chronic fatigue and loss of appetite
- Nausea or vomiting



Model

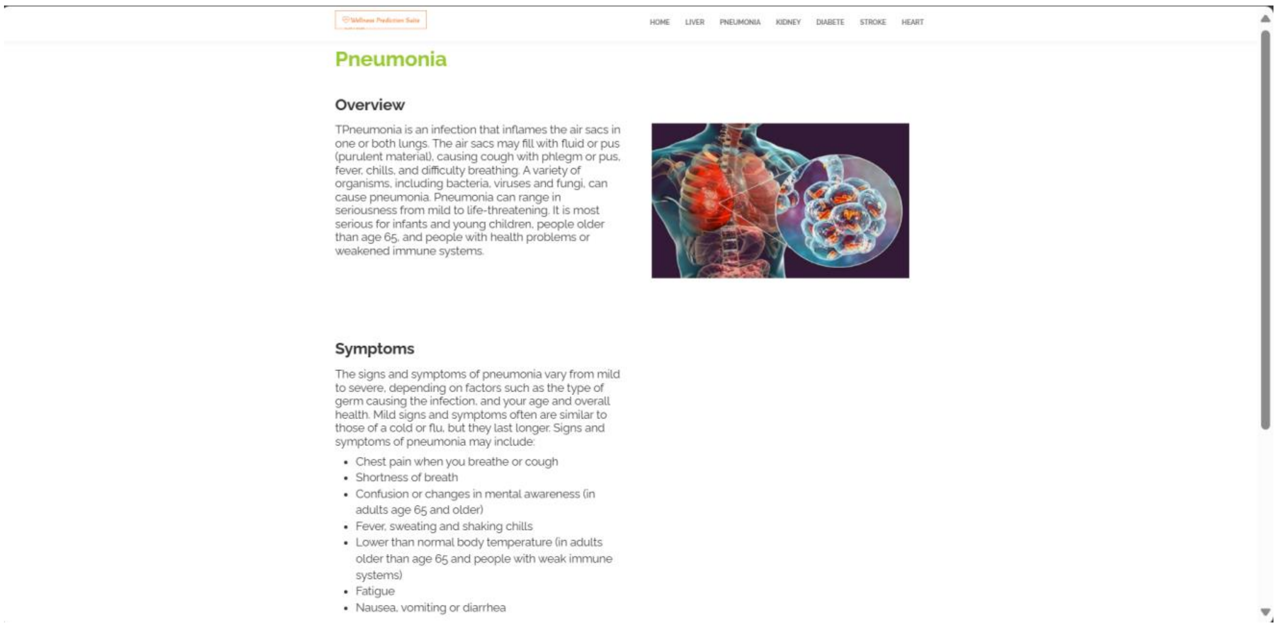
To make the diagnosis you need to enter the following informations :

- Age: age of the patient
- Gender: Male, Female or Other
- Total Bilirubin: result of blood test that measures the amount of a substance called bilirubin
- Direct Bilirubin: result of blood test that measures the amount of direct bilirubin
- Alkaline Phosphotase: result of blood test to measures the amount of an enzyme called alkaline phosphotase
- Alanine Aminotransferase: result of test to measure the amount of an enzyme called alanine aminotransferase
- Aspartate Aminotransferase: result of test to measure the amount of the enzyme aspartate aminotransferase
- Total Protiens: A total protein test measures the sum of all types of proteins in the blood
- Albumin: Result of albumin blood test that measures the amount of albumin in your blood. Albumin is a protein made by your liver
- Albumin and Globulin Ratio: result of a test that compares the concentrations of albumin and globulin in the blood



Enter the Following parameters :

Age	age
Total_Bilirubin	Total_Bilirubin
Direct_Bilirubin	Direct_Bilirubin
Alkaline_Phosphotase	Alkaline_Phosphotase
Alamine_Aminotransferase	Alamine_Aminotransferase
Aspartate_Aminotransferase	Aspartate_Aminotransferase
Total_Protiens	Total_Protiens
Albumin	Albumin
Albumin_and_Globulin_Ratio	Albumin_and_Globulin_Ratio
Gender	male



Pneumonia

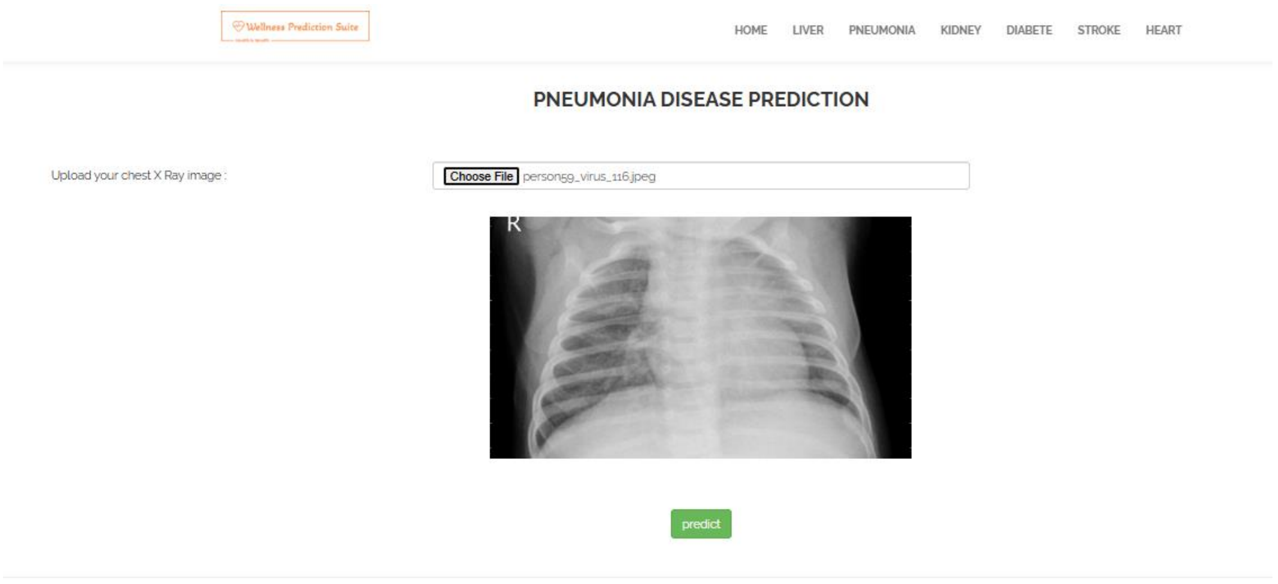
Overview

TPneumonia is an infection that inflames the air sacs in one or both lungs. The air sacs may fill with fluid or pus (purulent material), causing cough with phlegm or pus, fever, chills, and difficulty breathing. A variety of organisms, including bacteria, viruses and fungi, can cause pneumonia. Pneumonia can range in seriousness from mild to life-threatening. It is most serious for infants and young children, people older than age 65, and people with health problems or weakened immune systems.

Symptoms

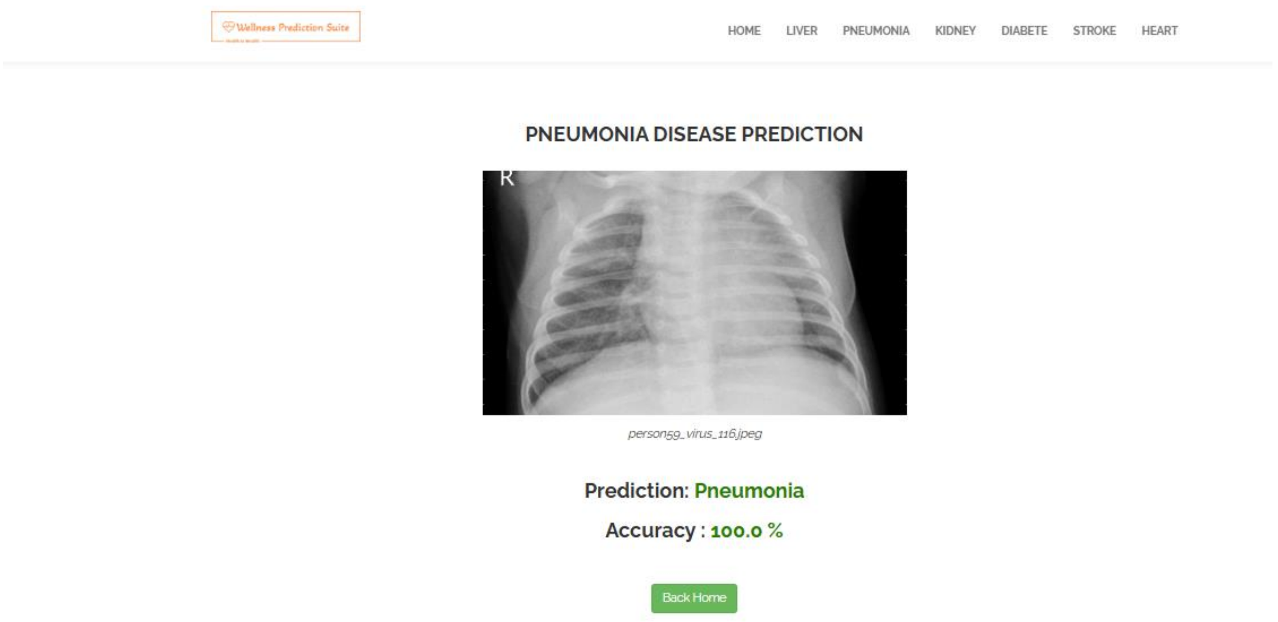
The signs and symptoms of pneumonia vary from mild to severe, depending on factors such as the type of germ causing the infection, and your age and overall health. Mild signs and symptoms often are similar to those of a cold or flu, but they last longer. Signs and symptoms of pneumonia may include:

- Chest pain when you breathe or cough
- Shortness of breath
- Confusion or changes in mental awareness (in adults age 65 and older)
- Fever, sweating and shaking chills
- Lower than normal body temperature (in adults older than age 65 and people with weak immune systems)
- Fatigue
- Nausea, vomiting or diarrhea



PNEUMONIA DISEASE PREDICTION

Upload your chest X Ray image : person59_virus_116.jpeg

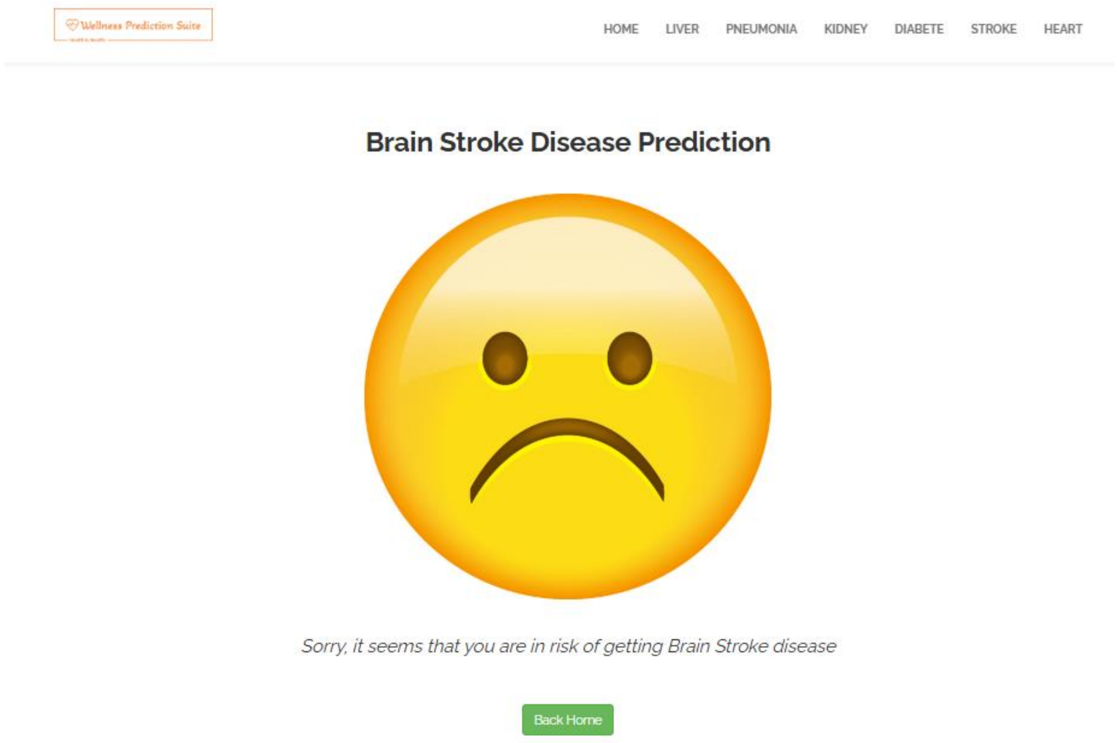


PNEUMONIA DISEASE PREDICTION

person59_virus_116.jpeg

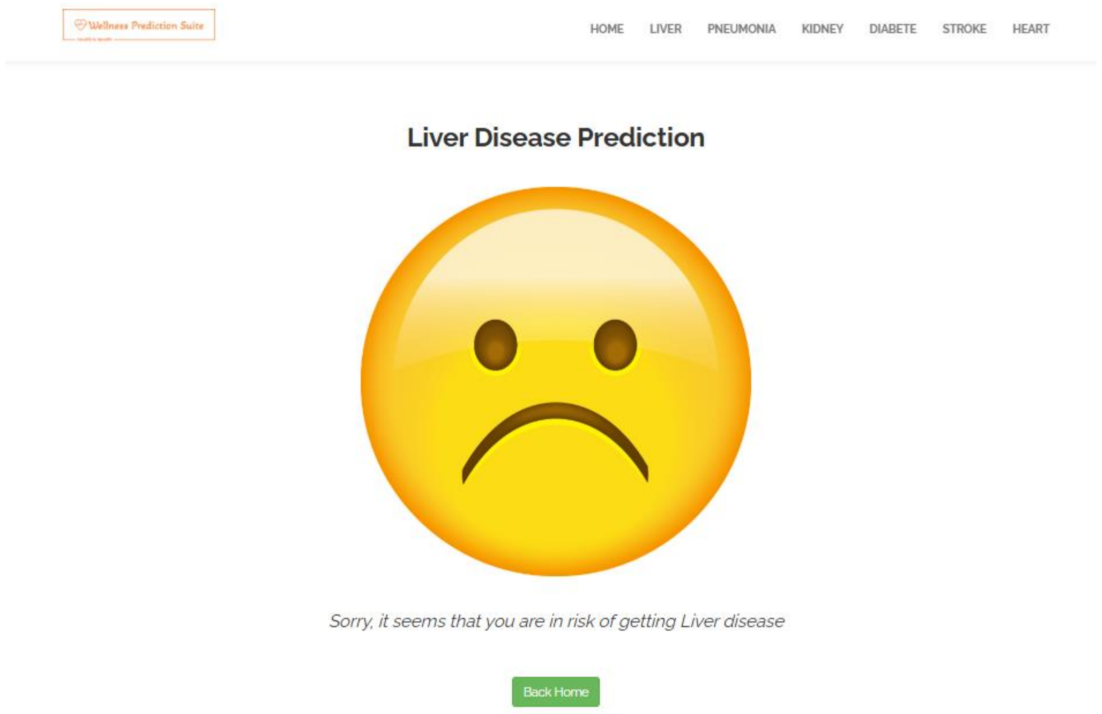
Prediction: Pneumonia

Accuracy : 100.0 %



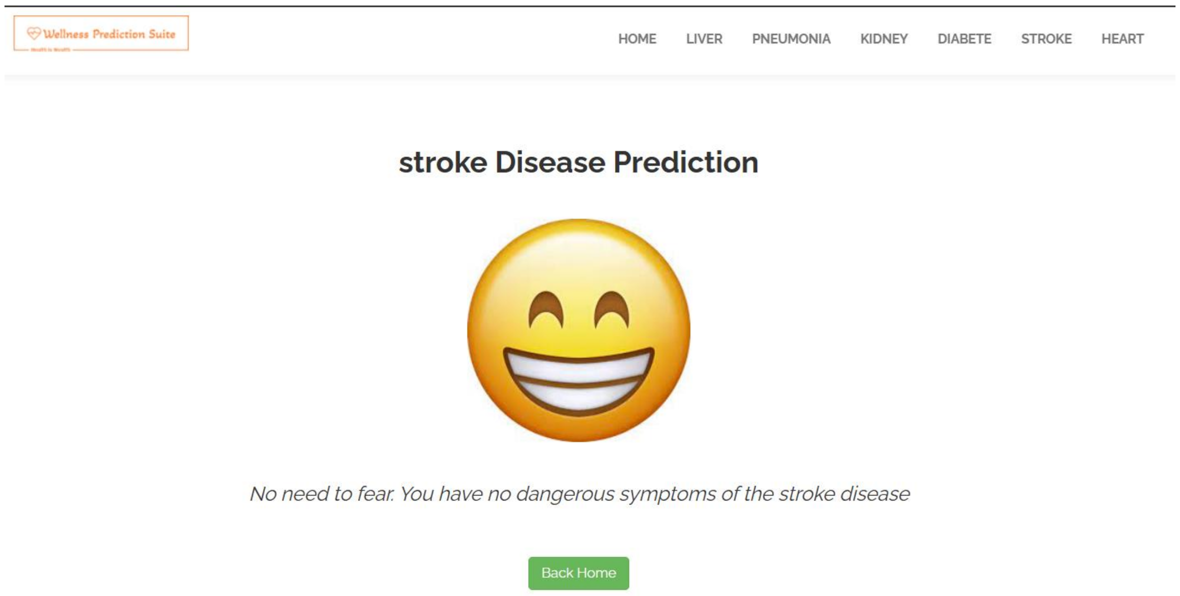
Brain Stroke Disease Prediction

Sorry, it seems that you are in risk of getting Brain Stroke disease



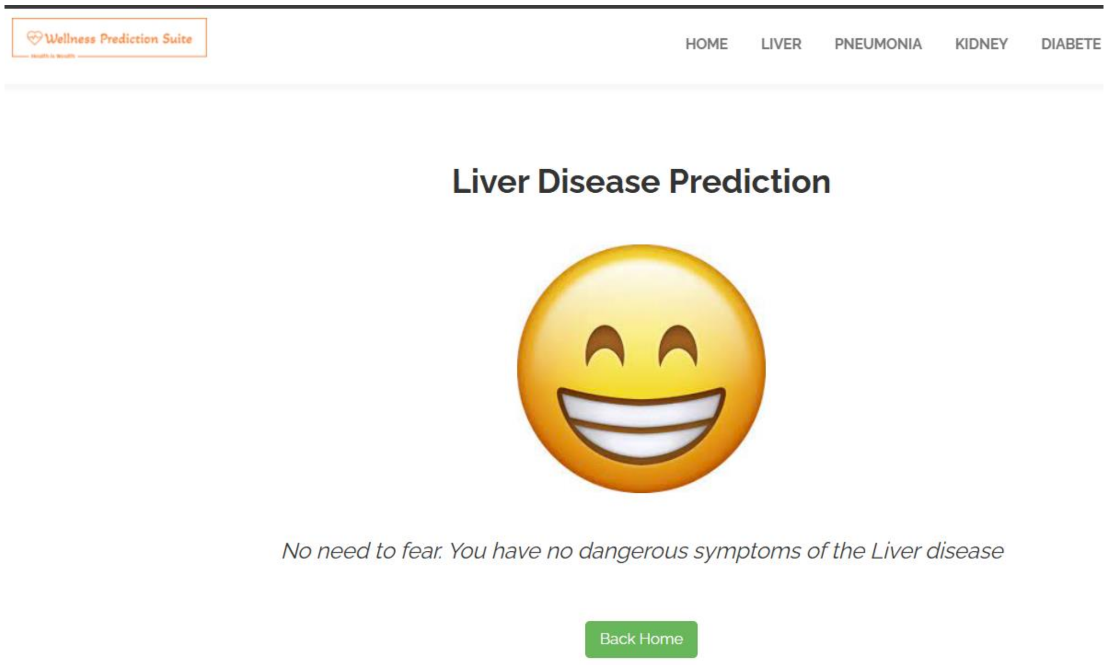
Liver Disease Prediction

Sorry, it seems that you are in risk of getting Liver disease



stroke Disease Prediction

No need to fear. You have no dangerous symptoms of the stroke disease



Liver Disease Prediction

No need to fear. You have no dangerous symptoms of the Liver disease

CHAPTER – 8

CONCLUSION AND FUTURE SCOPE

The development of the Wellness Prediction Suite marks a significant advancement in the field of predictive healthcare technology. By integrating machine learning algorithms and user-friendly interfaces, the suite empowers individuals to proactively manage their health and well-being. The project has demonstrated promising results in providing personalized health predictions, therebyenhancing preventive care measures & improving overall health outcomes.

Looking ahead, the future scope of the Wellness Prediction Suite is extensive and dynamic. Continuous refinement of machine learning models is essential to improve prediction accuracy and accommodate evolving health trends. Leveraging advancements in artificial intelligence and data analytics can further enhance the suite's predictive capabilities, enabling more accurate and timely health predictions.

Furthermore, integrating real-time data streams and implementing adaptive learning mechanisms can enhance the suite's ability to adapt to individual health needs and changing environmental factors. Collaboration with healthcare professionals and institutions is crucial to validate the effectiveness of the suite in clinical settings and ensure its seamless integration into existing healthcare infrastructure. Moreover, the deployment pipeline utilizing modern software development practices enables scalability and reproducibility, facilitating the suite's expansion to address a broader range of health conditions and predictive analytics.

Addressing challenges such as data privacy, model interpretability, and regulatory compliance will be paramount to ensure the widespread adoption of the Wellness Prediction Suite in healthcare settings. By addressing these challenges and embracing continuous research, innovation, and collaboration, the suite has the potential to revolutionize healthcare delivery, ultimately leading to improved health outcomes and enhanced quality of life for individuals worldwide.

BIBLIOGRAPHY

Code References:

- OpenCV Documentation on Background Subtraction: https://docs.opencv.org/3.4/db/d5c/tutorial_py_bg_subtraction.html
- OpenCV Documentation on cv2.findContours: https://docs.opencv.org/3.4/df/d0d/tutorial_find_contours.html
- OpenCV Documentation on cv2.morphologyEx: https://docs.opencv.org/4.x/d9/d61/tutorial_py_morphological_ops.html

Advanced Video Anomaly Detection Techniques:

- **Prediction and Classification:**
  - Scikit-learn: https://scikit-learn.org/ offers tools for building anomaly scoring systems (e.g., Support Vector Machines, Random Forests).
  - TensorFlow / PyTorch: https://www.tensorflow.org/, https://pytorch.org/ enable development of deep learning models for prediction & classification.

Additional Resources:

- OpenCV Tutorials: https://docs.opencv.org/4.x/d9/df8/tutorial_root.html provide in-depth guides on advanced background subtraction and optical flow analysis.

APPENDIX A: TOOLS AND TECHNOLOGIES

This appendix provides an overview of the various tools and technologies used in the development of our intelligent chatbot system. Each tool and technology played a crucial role in ensuring the functionality, efficiency, and user experience of the chatbot.

- **Python:**
  - **Programming Language:** Python serves as the foundation for code development, enabling the integration of OpenCV's functionalities and potential future libraries.

  - **Flask:** Flask provides the foundation for web development, facilitating the creation of robust web applications and APIs, with potential for integration with various extensions and libraries.
- **NumPy:**
  - **Efficient Array Manipulation:** While not used in the current implementation, NumPy offers efficient array manipulation capabilities. It can significantly accelerate computations during image processing tasks like background subtraction and feature extraction (for future enhancements).
- **TensorFlow/Keras:**
  - **Machine Learning Integration:** TensorFlow and Keras are powerful libraries for building and training machine learning models.

These tools and technologies collectively contributed to the successful development and deployment of our intelligent chatbot system, providing a robust, secure, and user-friendly solution for handling anomaly detection in images and videos.