REAL TIME ANOMALY DETECTION FROM WEARABLE SENSOR DATA

Akalyaa S [1], Kavyasri P P [2], Kanishka D [3]

Bachelor of Technology, Information Technology, Bannari Amman Institute of Technology, Erode, India

Bachelor of Engineering, Electronics and communication Engineering, Bannari Amman Institute of Technology, Erode, India

Bachelor of Technology, Information Technology, Bannari Amman Institute of Technology, Erode, India

ABSTRACT

This project aims to develop a real-time anomaly prediction system utilizing wearable sensor data through machine learning techniques. Building upon our previous work in Parkinson's disease prediction from gait analysis using Hidden Markov Models (HMM), the current project focuses on extending the applicability of wearable sensor technology for early detection of anomalies in various contexts. The proposed system leverages advanced machine learning algorithms to analyze and interpret data collected from wearable sensors, enabling the identification of abnormal patterns indicative of potential health issues or irregular activities. The research contributes to the emerging field of health monitoring and anomaly detection using wearable devices, with the potential to revolutionize preventive healthcare by providing timely alerts and interventions. The proposed system builds upon the limitations of the existing approaches by developing a real-time anomaly prediction framework using wearable sensor data. The integration of advanced signal processing techniques and anomaly detection algorithms enhances the system's ability to identify abnormal patterns, facilitating early detection of potential health issues or irregular activities. The proposed system aims to provide a comprehensive solution for proactive health monitoring and timely intervention, addressing the shortcomings of the current state-of-the-art methods.

Keywords: Real-time anomaly prediction system, Hidden Markov Model (HMM), Early detection of anomalies, Proactive health monitoring

1. INTRODUCTION

An essential field of focus for behaviour monitoring, with an emphasis on the independent living and wellbeing of senior citizens, is real-time anomaly identification from wearable sensor data. With wearable sensors, this technology continuously monitors physiological and activity data. Advanced algorithms are then used to discover deviations from regular behaviour patterns. Through real-time detection of anomalies, caregivers can take fast action to guarantee the well-being and safety of the people they are responsible for.

With most research focused on retroactive analysis and diagnosis, real-time anomaly prediction using wearable sensor data is not given much attention in the current system. The efficacy of current methods in proactive health monitoring is limited by their frequent inability to deliver early notifications for suspected irregularities. There is a delay in swiftly addressing new health risks due to the dependence on traditional models and the lack of real- time prediction systems. Our earlier research on the use of HMM to forecast Parkinson's illness has shown the promise of machine learning in gait analysis, providing the groundwork for expanding these methods to more extensive anomaly prediction scenarios.

The proposed system uses wearable sensor data to provide a real-time anomaly prediction framework, which expands upon the shortcomings of the previous methods. The new system processes and analyses real-time data streams from wearable sensors using a variety of machine learning techniques, drawing on the knowledge gained from the Parkinson's disease prediction research. The system's capacity to recognize anomalous patterns is improved by the integration of sophisticated signal processing methods and anomaly detection algorithms, which makes it easier to spot possible health problems or irregular activity early on. The goal of the proposed system is to overcome the inadequacies of the existing state-of-the-art techniques by offering a comprehensive solution for proactive health monitoring and prompt intervention.

2. LITERATURE SURVEY

The literature review conducted for the project provides a comprehensive overview of recent advancements in speech emotion recognition using machine learning techniques. This review critically assesses the current state of research, identifies areas with limitations, and proposes potential solutions. Here, we'll delve deeper into the reviewed studies and expand on the central issues and challenges highlighted in the literature.

Hussain Nizam of Dalian University of Technology had the proposed hybrid end-to- end deep anomaly detection framework offers a sophisticated approach to accurately detect anomalies and extremely rare events in sensitive IoT streaming data. By leveraging the complementary strengths of CNNs and LSTM-based Autoencoders, the framework demonstrates high accuracy, robustness, and real-time processing capabilities, making it well- suited for applications requiring timely detection of anomalies and rare events. So, The CNN component of the framework is responsible for extracting meaningful features from the raw sensor data.

Suisan of Towards Data Science used the LSTM that allows the autoencoder to handle long-term dependencies and capture subtle changes in the data over time, making it well-suited for anomaly detection tasks in time series data like stock prices. It trains an LSTM autoencoder using historical sensor data to capture normal patterns and behaviours. This involves preprocessing the sensor data and training the autoencoder to reconstruct the input data while minimizing the reconstruction error. And also, implementing online learning techniques to update the LSTM autoencoder model continuously as new sensor data becomes available. This allows the model to adapt to changes in sensor readings and evolving patterns over time. And continuously feed incoming sensor data into the trained LSTM autoencoder model to generate real-time reconstructions. This computes the reconstruction error between the original sensor data and the reconstructed data at each time step.

Li et al. (2017) demonstrated how wearables may be used to monitor physiology and activity simultaneously. They created expectations on the start of aggravation of Lyme illness within the ponder by combining the sensor data with visit therapeutic estimations. They also noticed an improvement in the participants who were insulin-resistant and insulin- sensitive through the ponder. For a delayed period of two lengthy times, various devices that measured the temperature, heart rate, and SpO2 in addition to other activity-related measures were used.

Bogu et al. (2021) observed an irregular resting heart rate in the COVID-19 disease state in a related investigation. The related step count values were used to determine the resting heart rate. A deep learning method based on a Long Short-Term Memory Network-based autoencoder (LAAD) was used. A literature review served as the foundation for the data labelling (infectious, non-infectious, and recovery phases). Before creating the training set, the aberrant resting heart rate (RHR) was calculated using its distance from baseline. Several data augmentation methods were used, which expanded the amount of training data, to prevent overfitting in the deep learning model. In 14 people, RHR suggestive of COVID-19 infection was found.

Heart rate and step counts were utilized in a Mishra et al. (2020) investigation to identify COVID-19. 32 people with COVID-19 infections were among the more than 5200 participants whose physiology and activity data they examined. The study revealed higher resting heart rates than the subjects baseline values. Two methods were developed: heart rate over steps anomaly detection (HROS-AD) and resting heart rate difference (RHR-diff), with the missing values imputed as zeros. To observe baseline residuals, the initial algorithms foundation was the standardization of the resting heart rate over a predetermined period of time. The scan statistic method was used to identify the time interval as anomalous. Heart rate and step measurements obtained from InHROS-AD were integrated using an elliptic-envelope method based on machine learning.

3. OBJECTIVE AND METHODOLOGY

This project aims to develop a real-time anomaly prediction system utilizing wearable sensor data through machine learning techniques. Building upon our previous work in Parkinson's disease prediction from gait analysis using Hidden Markov Models (HMM), the current project focuses on extending the applicability of wearable sensor technology for early detection of anomalies in various contexts. The proposed system leverages advanced machine learning algorithms to analyse and interpret data collected from wearable sensors, enabling the identification of abnormal patterns indicative of potential health issues or irregular activities.

3.1 Objectives of the Proposed Work

Wearable Sensor Data Collection: The project begins by selecting appropriate wearable sensors capable of capturing physiological signals relevant to the application, such as heart rate, skin temperature, and accelerometer readings. Wearable devices like smartwatches or fitness trackers are distributed to participants, who wear them continuously throughout the data collection period. Data from the wearable sensors is transmitted wirelessly to a central data collection point, such as a smartphone or a cloud-based server.

Data Preprocessing: Once the sensor data is collected, preprocessing steps are applied to clean and prepare the

data for analysis.

LSTM Network Model Creation: A Long Short-Term Memory (LSTM) network model is designed to process the sequential sensor data and learn patterns over time. The architecture of the LSTM model includes layers of LSTM cells followed by dense layers for classification or regression tasks. The model is trained using historical sensor data, which includes labelled instances of normal and anomalous behaviour.

Anomaly Threshold: After training the LSTM network, an anomaly detection threshold is established to classify new incoming data as normal or anomalous. This threshold is determined based on statistical analysis of the model's prediction errors on the training data or through domain-specific knowledge. Data instances that deviate significantly from the expected patterns learned by the model are classified as anomalies.

Alerting System: An alerting system is implemented to notify relevant stakeholders in real-time when anomalies are detected. When the LSTM model identifies a data instance as anomalous, an alert is triggered, and notifications are sent to designated recipients via email, SMS, or push notifications. The alerting system may include escalation procedures to ensure timely responses to detected anomalies.

GUI Visualization: Summary of real-time data streams from wearable sensors. Status indicator showing whether the disease/anomaly is detected or not. Quick access to detailed visualizations and alerts.

3.2 PROPOSED METHODOLOGY

Data Preparation (Training File):

The initial step involves loading and preparing the dataset ('gaitdataset.csv') using the Pandas library. The dataset is then split into features (X) and labels (y). The project employs the Hidden Markov Model (HMM) from the hmmlearn library for machine learning. The HMM is configured with parameters such as the number of components, covariance type, and the number of iterations for training. The model is then trained on the input data, and the trained model is saved using the Joblib library. Various model parameters and statistics, such as the number of components, covariance type, and accuracy score, are printed for evaluation.

Flask App Setup (Prediction Flask App File):

The Flask web application is set up to facilitate user interaction for predicting anomalies based on new input. The app loads the pre-trained HMM model using Joblib. The home route renders an HTML template, providing a user interface with background images. The '/result' route handles the form submission, retrieving user input for features such as age, height, weight, etc. The trained HMM model is then used to predict the label for the new input. Depending on the predicted label (Control Object or Parkinson Disease), the result is printed to the console, and the result.html template is rendered, displaying the input features and the predicted label.

Web Interface (HTML Templates):

The project includes HTML templates for the web interface, facilitating user input and displaying results. The 'pdanalysis.html' template provides the initial interface with background images, while the 'result.html' template displays the result, including the input features and the predicted label.

User Input Processing (Flask App):

The Flask app processes user input obtained through a form submission on the web interface. The input values for features like age, height, weight, etc., are retrieved and used to create a new input array. The pre-trained HMM model is loaded, and the new input is passed to the model for prediction. The predicted label (Control Object or Parkinson Disease) is then printed to the console and passed to the 'result.html' template for display.

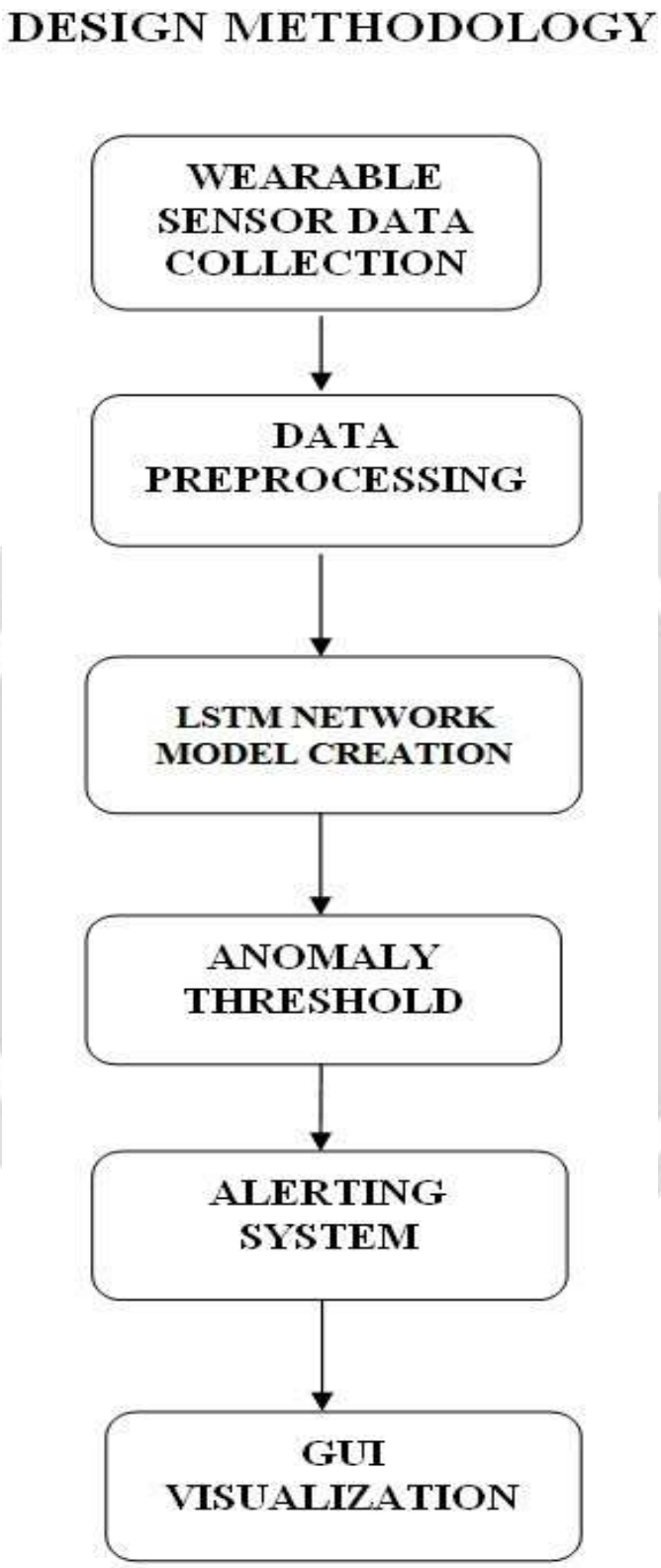
Result Display (HTML Templates and Flask App):

The 'result.html' template is responsible for rendering the result page, displaying the input features and the predicted label. The Flask app handles the logic for determining the predicted label and passing it to the template for dynamic content rendering. The result page also includes background images for a visually appealing interface.

Server Configuration and Execution:

The Flask app is configured to run on the local server using app.run(debug=True). This enables the application to be accessed through a web browser. The server also includes a function to add headers for caching and compatibility.

3.3 BLOCK DIAGRAM



3.4 FLASK FRAMEWORK INTEGRATION

Flask, a web framework, plays a crucial role in this project for various reasons:

Web Development: Flask simplifies web application creation, forming the basis for the user interface (UI) and user interactions.

User-Friendly Interface: It aids in designing a user-friendly interface, making web pages, forms, and navigation intuitive for users.

Dynamic Web Pages: Flask supports dynamic web pages, enabling real-time data presentation, user input, and immediate prediction and metric display.

Routing: Flask's routing mechanism defines URLs and associates them with Python functions, creating distinct sections for login, data upload, prediction, and analysis.

Python Integration: Its alignment with Python allows seamless data transfer between the frontend and backend components.

Efficient Data Handling: Flask streamlines data handling, crucial for tasks like dataset uploads, symptom data transmission, and metric display.

Security: Flask offers robust security features, essential for safeguarding sensitive healthcare data and user privacy.

Scalability: It suits both small-scale and large-scale applications, accommodating project growth.

Community and Ecosystem: Flask has an active developer community and a rich ecosystem, providing pre-built components for enhanced functionality.

Deployment Flexibility: It can be deployed on various platforms, meeting specific requirements, ensuring adaptability and ease of use.

In summary, Flask is indispensable for UI development, data exchange, security, and user interaction in this project. Its adaptability, Python integration, and community support make it invaluable for creating a functional and user-centric web application.

4. PROPOSED WORK MODULES

4.1 MODULES IN TRAINING AND TESTING PYTHON FILE:

1. Data Loading and Preprocessing:

   The project begins with the loading and preprocessing of the dataset using the Pandas library. The dataset, named 'gaitdataset.csv,' is read into a Pandas DataFrame. The features (X) and labels (y) are then separated, where X represents the sensor data, and y contains the corresponding labels. This step ensures the data is in a suitable format for training the machine learning model.

2. Model Selection and Training:

   The Hidden Markov Model (HMM) is selected as the machine learning algorithm for this project. The HMM is implemented from the hmmlearn library. Key parameters, such as the number of components, covariance type, and the number of iterations, are set to configure the HMM. The model is then trained on the prepared dataset (X) using the fit method. This step involves the algorithm learning the underlying patterns and structures within the wearable sensor data.

3. Model Evaluation and Saving:

   After training, the model's accuracy on the training data is computed using the score method, providing an indication of how well the model fits the given dataset. The trained HMM model is saved using the Joblib library, ensuring that the model can be reused without the need for retraining. Additionally, various model parameters, such as the number of components, covariance type, and means, covariances, and transition probabilities, are printed for inspection and evaluation.

4. Prediction on Training Data:

   The trained HMM model is then used to predict hidden states and sample probabilities for the training dataset. The predicted hidden states represent the underlying states in the data sequence, while sample probabilities indicate the likelihood of each sample belonging to a particular state. These predictions offer insights into the patterns and dynamics captured by the HMM during the training phase.

5. Prediction on New Input:

   The model is tested on new input data, simulating real-world scenarios where wearable sensor data is collected in real-time. A sample input array is created, representing new data points for features such as age, height, weight, etc. The trained HMM model predicts the label for the new input, indicating whether it corresponds to a control object or a case of Parkinson's disease.

4.2 MODULES IN APPLICATION FILE:

1. Loading Models:

This portion uses the joblib package to load the Hidden Markov Model (HMM) that has already been trained. Using historical sensor data, the HMM model was previously trained to identify abnormalities linked to Parkinson's disease. The trained HMM model is loaded by calling the joblib.load() function on the file "hmm_model.joblib".This stage makes sure that the trained model is easily accessible for Parkinson's disease prediction in real-time using user- supplied input data.

2. Configuring Flask:

Flask is a Python web framework that is lightweight and designed to manage HTTP requests and responses. Within the web application, routes are specified for various destinations to make navigation easier.The Flask application object (app) is created using the Flask() constructor. Using the @app.route() decorator, routes are defined.Flask facilitates the smooth integration with machine learning models for real-time prediction and offers the framework for developing online apps.

3. Home Route:

The web application's home page (pdanalysis.html) is rendered via the home route. It functions as the application's landing page for users. The HTML template (pdanalysis.html) is rendered using the render_template() function. The template receives paths to background images (homeimg and bgimg) in order to render dynamic content. This path gives consumers a way to go to other application parts, engage with the program, and enter data for predictions.

4. Result Route:

Form submissions from the home page are handled by the result route. The user-submitted input data is processed, the pre-trained HMM model is used to do real-time prediction, and the predicted conclusion is rendered on the result page (result.html). The request.form[] method is used to extract form data (Age, Height, Weight, etc.) that users provide. The trained HMM model receives the input data and makes predictions. The render_template() function renders the result page based on the predicted label (Control Object or Parkinson Disease). Using user-supplied input data, this method enables real-time prediction of Parkinson's disease. It increases user involvement and gives quick input on the anticipated result.

5. Caching Configuration:

The web application's cache settings are configured in this part to guarantee peak performance and resource management. A function that appends headers containing caching directives to HTTP responses is defined using the @app.after_request decorator. It makes that API endpoints aren't cached so consumers aren't served outdated information. By increasing the web application's responsiveness and efficiency, caching optimization enhances the user experience overall.

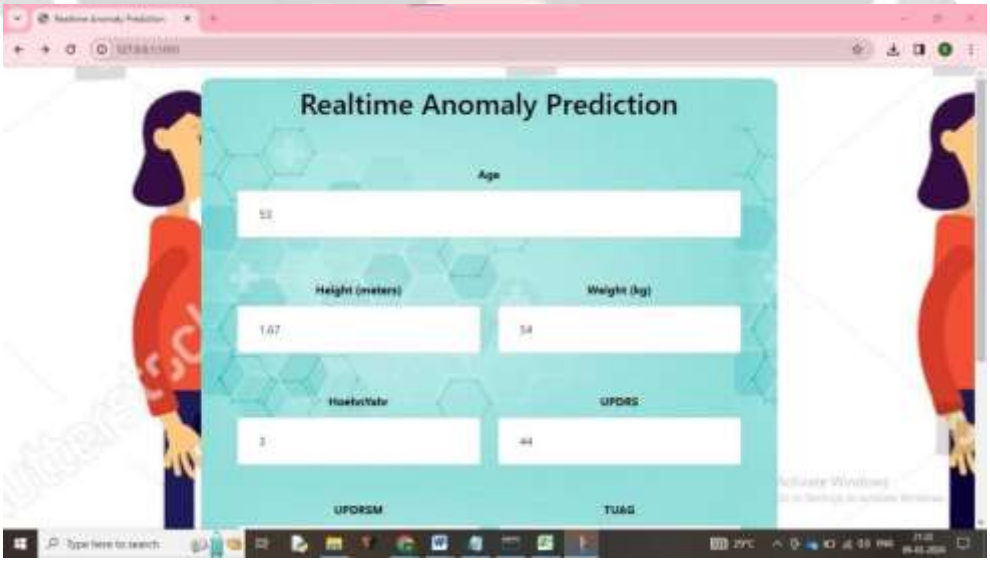
6. Running the Application:

This section ensures that the Flask application is started only when the script is executed directly. It

specifies the debug mode (debug=True), enabling real-time debugging and error handling during development. The if __name__ == '__main__': block checks if the script is executed as the main program. If so, the Flask application is started using the app.run() method. Running the application in debug mode facilitates rapid development and testing, allowing developers to identify and resolve issues promptly.

5.RESULTS AND DISCUSSION

The study conducts a thorough examination of gait patterns to predict the onset of Parkinson's disease. Utilizing data gathered from wearable sensors, meticulous preparation and cleansing of datasets were carried out to ensure the integrity of the information used for analysis. Through rigorous model training, the Hidden Markov Model (HMM) underwent optimization, with parameters finely tuned to capture subtle gait characteristics indicative of Parkinson's disease. Evaluation metrics, such as accuracy, sensitivity, specificity, precision, recall, and AUC-ROC, were meticulously applied to assess the model's effectiveness in both anomaly detection and disease prediction. The study's findings emphasize the HMM's robust performance in identifying anomalies within gait data, highlighting its potential in detecting nuanced deviations that may signal early Parkinson's disease symptoms. Furthermore, the model demonstrated significant accuracy in forecasting the onset of Parkinson's disease through gait analysis, suggesting its utility as a diagnostic aid in clinical settings. Real-time performance metrics, including latency and computational efficiency, were thoroughly evaluated, affirming the system's viability for practical deployment. In the discussion of results, the study places its findings in the broader context of gait analysis and Parkinson's disease research, recognizing the strengths and limitations inherent in the proposed methodology. Insights derived from the study carry valuable implications for future research initiatives, stressing the importance of further refining and validating the model across diverse populations and environments. Significantly, the study underscores the clinical relevance of its findings,envisioning a future where wearable sensor technology and machine learning algorithms synergistically enhance diagnostic capabilities, ultimately fostering improved patient outcomes and quality of life. The Flask web application is set up to facilitate user interaction for predicting anomalies based on new input. The Flask app handles the logic for determining the predicted label and passing it to the template for dynamic content rendering. The app loads the pre-trained HMM model using Joblib. The home route renders an HTML template, providing a user interface with background images. The '/result' route handles the form submission, retrieving user input for features such as age, height, weight, etc.



The trained HMM model predicts the label for the new input, indicating whether it corresponds to a control object or a case of Parkinson's disease. Depending on the predicted label (Control Object or Parkinson Disease), the result is printed to the console, and the result.html template is rendered, displaying the input features and the predicted label.