AI-DRIVEN REAL-TIME THREAT RESPONSE AND ALERT SYSTEM FOR WOMEN AND VULNERABLE GROUPS USING KNN, VOICE RECOGNITION AND IOT

Irene Lu [1], Richard Guo [2]

[1] Arnold O. Beckman High School, 3588 Bryan Ave, Irvine, CA 92602
[2] Computer Science Department, California State Polytechnic University, Pomona, CA 91768

ABSTRACT

This paper addresses the pressing issue of female safety through the design of a wearable device capable of detecting abnormal movements and recognizing voice commands [1]. Our approach integrates KNN-based motion detection and real-time speech recognition, offering a more reliable and accurate system for emergency detection [2]. Unlike previous methods that rely on computationally intensive models, such as ANNs and SVMs, our solution is optimized for low-power devices, ensuring real-time response without compromising efficiency [3]. The system was tested on various activities and in noisy environments, demonstrating its ability to detect emergency situations reliably. The results highlighted the importance of addressing edge cases, such as slower movements in elderly users and fast actions in athletes, which we plan to refine in future iterations. The combination of motion detection and voice recognition provides a flexible and dynamic safety mechanism, enhancing personal security [4]. Our proposed solution offers a significant step toward affordable and accessible safety technology, with the potential to empower women globally by improving their security in everyday situations.

KEYWORDS

KNN Classifier, Voice Recognition, Datasets, Real-Time

1. INTRODUCTION

1.1. Problem Statement: Female Safety Under Threat

The problem at hand is the ongoing threat to female safety worldwide, specifically concerning physical and sexual violence, harassment, and human trafficking. Despite technological advancements, affordable and accessible safety devices remain scarce, particularly in low- and middle-income countries. Many women face barriers in seeking formal assistance, with less than 40% of those experiencing violence reaching out for help. This highlights the pressing need for easily accessible, user-friendly safety tools. Our goal is to design wearable electronic devices with movement detection and voice recognition capabilities to provide immediate assistance and enhance personal security for women [5].

David C. Wyld et al. (Eds): ADCO, CMCA, NeCoM, NLPI, CSITEC, BDBS, DMML, SP – 2025
pp. 117-128, 2025. CS & IT - CSCP 2025 DOI: 10.5121/csit.2025.150510

1.2. Background/History of the Problem

Globally, nearly one in three women (736 million) have experienced physical and/or sexual violence from partners or non-partners. Human trafficking compounds the issue, with women and girls making up 91% of those trafficked for sexual exploitation [6]. School-related gender-based violence also poses significant challenges to girls' education. While technological advancements have improved safety options, access to affordable devices remains limited in many parts of the world. According to a GSMA survey, 68% to 94% of women in 11 low- and middle-income countries reported feeling safer with mobile phones, yet many still lack access to affordable safety tools. This disparity in access emphasizes the urgent need for cost-effective solutions tailored to women's safety requirements.

1.3. Importance of the Problem

Female safety is crucial not only for individual security but also for societal progress [7]. Violence against women results in physical, emotional, and economic consequences, affecting families, communities, and nations at large. Addressing this issue is pivotal for achieving gender equality and reducing the societal costs of violence. The development of effective, affordable safety technologies can empower women, promote their confidence in public spaces, and enable their full participation in social and economic activities

.

1.4. Long-Term Impact

This issue affects women across all demographics, particularly in underserved regions, with broad implications for societal development and gender equity.

In Methodology A, ANN models offer high accuracy in activity recognition but struggle with high computational demands and data transfer latency, which can delay real-time responses. Our KNN-based approach mitigates these issues, offering a lighter, more efficient solution without compromising accuracy.

Methodology B explores the use of SVMs, which classify movement patterns effectively but require extensive feature engineering and tuning, making them computationally expensive. SVMs also suffer from poor performance on imbalanced datasets. Our KNN model addresses these limitations with a dynamically trained, computationally efficient approach, combining motion and voice recognition to reduce false positives.

Methodology C utilizes HMMs to predict abnormal movement based on predefined states. While effective in structured scenarios, HMMs lack real-time adaptability and struggle with diverse, unpredictable threats. Our system improves upon this by incorporating sensor fusion, ensuring dynamic adjustment to various conditions, and providing a more responsive solution with integrated KNN and speech recognition.

To address the issues identified, we propose a novel portable electronic device that combines acceleration detection with voice recognition to minimize the potential for misdetection. Our solution uses an improved KNN algorithm to differentiate between similar movement data, such as jogging and running in a rush, when abnormal behaviors are detected by the accelerometer [8]. To further ensure accuracy, we integrated Python's speech-recognition library to verify potential danger scenarios. This dual-detection process ensures a higher success rate in identifying threats. Compared with previous work,our approach offers a significant improvement by incorporating both movement and voice data for more reliable detection. Previous solutions focus on movement analysis, which carries a higher risk of false positives due to the similarity in patterns. By

combining acceleration and voice recognition, we address this limitation and provide a more robust safety mechanism. We believe this method is effective because it enhances the reliability and precision of detection in real-world scenarios. The integration of advanced algorithms and dual-detection processes not only minimizes false alarms but also ensures timely and accurate responses to potential threats. This makes it a far more comprehensive solution compared to existing approaches.

In the experiments, we aimed to test the accuracy and reliability of our system in detecting anomalies in different scenarios. The first experiment focused on simulating edge cases, such as elderly users and athletes, to evaluate the robustness of the KNN model. We tested our system on various activities, including normal walking, running, and activities involving sudden movements, ensuring the system could reliably detect emergency situations. The results revealed that while the system performed well in most cases, elderly users' slower movements resulted in false negatives, while athletes' dynamic activities sometimes caused false positives. These findings highlighted the need to enhance the algorithm's ability to distinguish between deliberate and emergency movements. The second experiment tested the effectiveness of our speech recognition system in noisy environments. The integration of noise cancellation techniques improved the system's performance, ensuring reliable voice detection even in challenging conditions.

2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

2.1. KNN Algorithm

A significant challenge is the potential operation failure of a standard KNN algorithm when implemented on a lightweight system such as the Raspberry Pi Zero. To address this, I could propose using a lightweight pre-trained KNN model optimized for the device's limited processing power. This model would reduce computational demands while maintaining performance, ensuring reliable operation even in constrained environments.

2.2. Detecting Trigger Words

Another challenge involves detecting trigger words in environments with significant background noise. Background noise could interfere with the microphone's ability to pick up trigger words accurately. To resolve this, I could implement noise cancellation techniques, either internally within the device or externally through compatible software or hardware. This approach would enhance the clarity of audio inputs and improve the reliability of voice detection.

2.3. Movement Misdetection

Movement misdetection presents a critical challenge, particularly in distinguishing between similar activities like jogging and rushing in a panic. Such misdetections could lead to false alarms, causing unnecessary distress for users and their families. To address this, I could apply an enhanced algorithm that combines an improved KNN approach with parallel speech recognition. This dual-layered detection process would increase accuracy and reduce the likelihood of false positives, ensuring a better user experience.

3. SOLUTION

The program integrates three core components: a motion detection system, a speech recognition system, and a data management backend. The flow begins with powering on the device, initializing its sensors, and continuously monitoring for irregular movements using an accelerometer. Upon detecting uncommon movements, the program triggers the speech recognition system to capture verbal commands or alert words. This process loops back to monitor further movement or proceed with predefined actions depending on user input.

Data from the motion detection and speech recognition systems are serialized into JSON format and transmitted to the Firebase cloud database for storage and analysis. The cloud infrastructure manages user data and facilitates communication between the frontend and backend. The user interface, developed using Flutterflow, acts as the bridge between the user and the backend, allowing real-time updates and seamless data retrieval.

To achieve efficient functionality, we employed lightweight ML models like KNN for motion classification, considering the limited computational capacity of the Raspberry Pi Zero. Similarly, speech recognition leverages PyAudio and Google's speech-to-text API for robust and adaptable recognition [9]. These components interconnect seamlessly through structured APIs and real-time data handling via Firebase. The program thus ensures a streamlined workflow, from detecting inputs to executing user-centric actions.

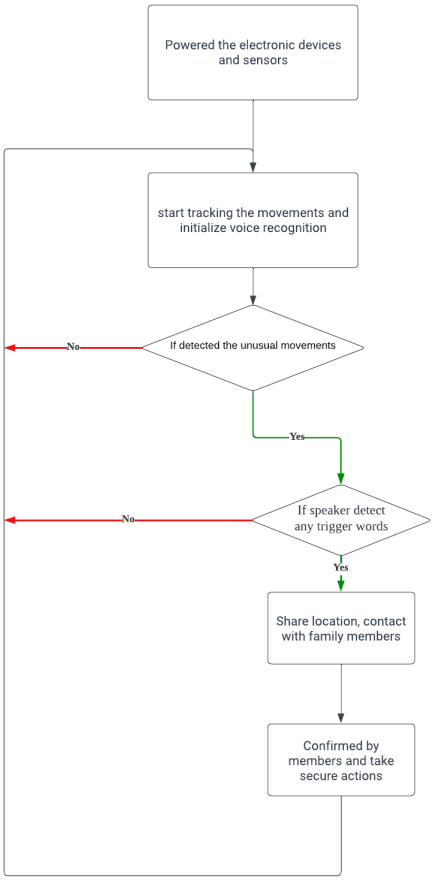

Figure 1. Overview of the solution

The motion detection system ensures accurate identification of unusual movements using accelerometer and gyroscope data. This system relies on the K-Nearest Neighbors (KNN) algorithm for classifying motion patterns. For KNN algorithm implementation for the classification of our acceleration, $||x^{(a)} - x^{(b)}||_2 = \sqrt{\sum_{j=1}^{d}(x_j^{(a)} - x_j^{(b)})^2}$, we applied the algorithm in the method of calculating the Euclidean distance between new data points and pre-labeled samples, determining the most likely activity category. We will test on the platform and decide on the best-suited K values for our acceleration classification.

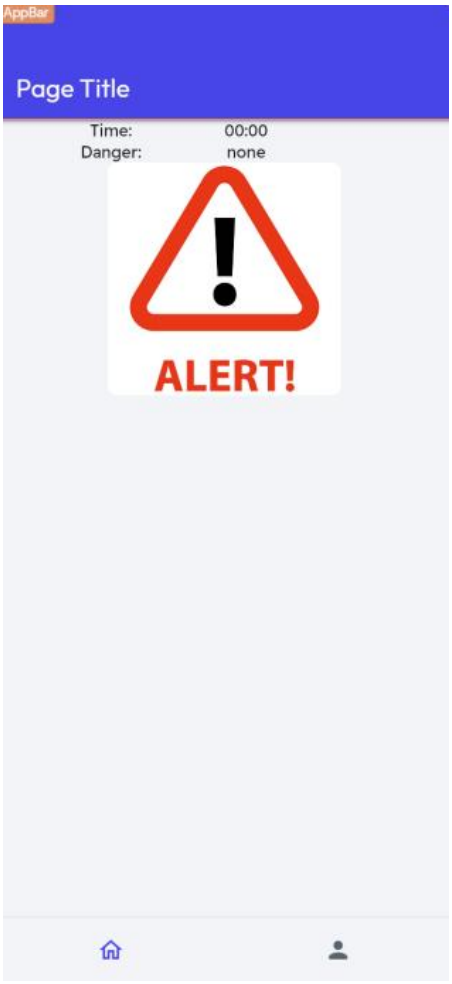

Figure 2. Screenshot of the page title

```
from google.colab import files
uploaded = files.upload()
#Libs import
import pandas as pd
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

file_name = list(uploaded.keys())[0]
dataset = pd.read_csv(file_name)

print(dataset.head())

#Finding the most suitable k value
```

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score

# Feature and target selection
X = dataset[['acceleration_x', 'acceleration_y', 'acceleration_z',
'gyro_x', 'gyro_y', 'gyro_z']]
y = dataset['activity']

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Feature scaling
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Hyperparameter tuning using GridSearchCV
param_grid = {'n_neighbors': range(1, 21)}
grid_search = GridSearchCV(KNeighborsClassifier(), param_grid, cv=5,
scoring='accuracy')
grid_search.fit(X_train_scaled, y_train)

# Best model and parameter
best_knn = grid_search.best_estimator_
print(f"Best k: {grid_search.best_params_['n_neighbors']}")

# Fit the best model
best_knn.fit(X_train_scaled, y_train)

# Predict and evaluate
pred_knn = best_knn.predict(X_test_scaled)
knn_acc = accuracy_score(y_test, pred_knn)
print("Optimized Accuracy for KNN: {:.2f}".format(knn_acc))
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
X = dataset[['acceleration_x', 'acceleration_y', 'acceleration_z',
'gyro_x', 'gyro_y', 'gyro_z']]
y = dataset['activity']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

knn = KNeighborsClassifier(n_neighbors=3)

knn.fit(X_train, y_train)

pred_knn = knn.predict(X_test)

knn_acc = accuracy_score(y_test, pred_knn)
print("Accuracy for KNN: {:.2f}".format(knn_acc))
import joblib

model_path = "knn_model.joblib"
joblib.dump(knn, model_path)
print(f"Model saved as {model_path}")

from google.colab import files
files.download(model_path)
```

Figure 3. Screenshot of code 1

The provided code implements a KNN model for classifying accelerometer and gyroscope data. This code runs during the model pre-training phase to prepare a classification model for deployment on a Raspberry Pi.

1. Libraries like pandas and sklearn handle data manipulation and modeling. The dataset containing motion data is loaded and inspected.
2. The features include accelerometer and gyroscope values, while the target represents activity labels. Data is split into training and testing sets and scaled using StandardScaler for uniformity.
3. GridSearchCV tests KNN with different k values (from 1 to 21) to identify the best one based on accuracy, and we found the best value of 3 with an accuracy of 0.99.
4. Apply k=3 as our KNN's k value and save that as the parameter of pretraining settings by training the model of KNN this class
5. The trained model is saved as a .joblib file for deployment, where the Raspberry Pi uses it for real-time classification.

For the speech recognition component, we employed the speech_recognition library, which integrates various speech-to-text engines and APIs. Its primary purpose is to detect specific trigger words from the user's speech, activating the next phase of the program. The system relies on offline and online engines, allowing flexibility depending on the operational context.

The implementation also uses PyAudio to access microphone input, enabling real-time processing of audio data. We configured the microphone index to ensure compatibility with the Raspberry Pi's audio hardware, specifically identifying the WM8960 microphone. The recognizer.adjust_for_ambient_noise method is used to cancel background noise, ensuring accurate speech detection even in non-ideal environments.

This component is crucial because it enables hands-free operation, making the system accessible to users with physical limitations. Speech recognition involves analyzing the audio signal, converting it into a text format, and checking for specific trigger words [10]. Once detected, the system transitions to the movement classification or other specified actions.

The microphone listens for user input and uses Google's speech recognition engine for transcription. Errors, such as no speech detected or API issues, are handled gracefully to ensure robust performance. This integration connects real-world inputs (speech) with the program's digital functionalities.

```
import speech_recognition as sr
import os

recognizer = sr.Recognizer()

mic_device_index = 0
recognizer_instance = sr.Mircophone()

for index, name in enumerate(sr.Microphone.list_microphone_names()):
  if "WM8960" in name:
    mic_device_index = index
    break
    print(f"Microphone with name \"{name}\" found")
    print(f"Microphone index: {index}")

if mic_device_index is None:
  raise RuntimeError("No microphone found")

def listen_for_trigger():
  try:
    with sr.Microphone(device_index=mic_device_index) as sourse:
      recognizer.adjust_for_ambient_noise(source)

      print("Waiting for speech(Press ctrl C to stop)...")
      audio_data = recognizer.listen(source, timeout = 2)
      text = recognizer.recognize_google(audio_data, language="en-US")
      print(f"Recognized text: {text}")

      except sr.UnknownValueError:
        print("No speech detected. Waiting for speech...")

      except sr.RequestError as e:
        print(f"Could not request results: {e}")


except KeyboardInterrupt:
  print("Stopping...")
```

Figure 4. Screenshot of code 2

This code sets up a speech recognition system using the speech_recognition library to capture audio from a microphone and convert it into text. The script first initializes a Recognizer object, which handles speech recognition tasks. It then searches through the list of available microphones and selects the one with the name "WM8960" for recording. If no such microphone is found, it raises an error. The main function, listen_for_trigger(), continuously listens for audio input. It adjusts for ambient noise, listens for speech, and then sends the recorded audio to Google's

speech recognition API for transcription. If successful, the recognized text is printed; if no speech is detected or if there's a network issue, corresponding error messages are displayed.

The function is designed to handle real-time audio input, which means it will keep running and waiting for speech until manually stopped (e.g., by pressing Ctrl + C). The recognizer object plays a key role in processing the audio input, adjusting for noise, and performing speech-to-text conversion. The microphone is set using the specified device index, and the recognized text is output to the console. In case of any interruptions or errors, the program gracefully handles them using blocks. This setup can be useful in applications where voice commands or continuous speech recognition are required.

The third major component of our program is the database system, which ensures secure data storage and management. This system primarily relies on Firebase Realtime Database, selected for its ability to handle dynamic, real-time data updates, and compatibility with our program's backend and frontend systems.

The purpose of the database is to store key data elements such as user inputs, device activity logs, and processed results from the machine learning models and the speech recognition module. Firebase's cloud-based system ensures that all data can be accessed and synchronized across devices seamlessly.

Firebase integrates smoothly with our program through its SDKs, allowing efficient read and write operations directly from both the Raspberry Pi and the Flutterflow frontend. The database is secured using Firebase Authentication, ensuring that only authorized devices and users can access sensitive data. Additionally, Firebase's structure provides real-time event-driven updates, ensuring that any changes made by the program are immediately reflected in the cloud and visible to the user interface.

This component is critical for the overall functionality of the system. For example, after the KNN model classifies movement patterns or the speech recognizer detects trigger words, the corresponding data is written to Firebase. The frontend retrieves this data to update the user dashboard, keeping all system components connected and functional.

```
import firebase_admin
from firebase_admin import credentials, storage, firestore


class Database:
    def __init__(self):
        # Replace the projectID eg.
        self.bucket_name = 'testone-b55de.appspot.com'
        # self.bucket_name = '<projectID>.appspot.com'

        # You need to download the serviceaccount.json
        self.fb_cred = 'service_account.json'
        cred = credentials.Certificate(self.fb_cred)
        firebase_admin.initialize_app(cred,
                                      {'storageBucket':
self.bucket_name})
        self.db = firestore.client()  # this connects to our
Firestore database

    def set_events(self, collection: str, device: str, data: dict):
        collection = self.db.collection(collection)  # opens
collection
        doc = collection.document(device)  # specifies the  document
        doc.set(data, merge=True)

    def exists_on_cloud(self, filename):
        bucket = storage.bucket()
        blob = bucket.blob(filename)
        if blob.exists():
            return blob.public_url
        else:
            return False

    def upload_file(self, firebase_path, local_path):
        bucket = storage.bucket()
        blob = bucket.blob(firebase_path)
        blob.upload_from_filename(local_path)
        print('This file is uploaded to cloud.')
        blob.make_public()
        url = blob.public_url
        return url
```

Figure 5. Screenshot of code 3

The code defines a Database class that connects to Firebase services—specifically Firestore and Firebase Storage—using the firebase_admin library. The __init__ method initializes the Firebase application by loading credentials from a service account JSON file (service_account.json) and specifying the storage bucket. Once initialized, it sets up the Firestore client to interact with the Firestore database, which allows the class to perform database operations such as adding or retrieving documents. Firebase Storage is also connected through the storage bucket, enabling file uploads and management in cloud storage.

The class has three key methods: set_events, exists_on_cloud, and upload_file. The set_events method uploads data to Firestore by setting a document in a specified collection, where merge=True ensures that the data is merged with existing content instead of replacing it. The exists_on_cloud method checks if a file exists in Firebase Storage by verifying if the blob (file) is present, returning a public URL if found. The upload_file method uploads a file from a local path to Firebase Storage, making the file publicly accessible and returning its public URL. These methods provide essential functionality for managing both database and storage operations in Firebase.

4. EXPERIMENT

The blind spot in our program is detecting anomalies in behavior for edge cases like elderly women or athletes. Reliable detection is crucial to avoid false positives or missed emergencies.

To address this, we will design an experiment simulating extreme conditions, such as intense physical activity (athletes) or slower movements (elderly individuals), and record them into a controlled dataset. These simulations will mimic real-life conditions to evaluate the robustness of our algorithm in detecting emergencies accurately across diverse groups.

Practical tests will include elderly volunteers performing daily activities and athletes engaging in high-intensity exercises. For control data, we'll use a subset of 50 women's datasets from Figshare to provide baseline statistics for comparison. This setup helps isolate edge cases and identify gaps in accuracy.

Dataset: Raw data from accelerometer and MET values
(we only take 50 women's dataset statistics)

Accuracy： 99 Percent correct

The data showed unexpected results in extreme cases: false negatives were more frequent in elderly users due to slower movement patterns, while athletes' dynamic activities sometimes triggered false positives. This deviation highlights that edge cases significantly affect detection accuracy.

The surprising observation was that the algorithm struggled to differentiate between abrupt, deliberate actions and emergencies. This likely results from insufficient diversity in the original dataset during training, which primarily focused on average user behaviors.

The largest impact on results was the variability of accelerometer data, emphasizing the need to refine the algorithm and train it on a more representative dataset.

5. RELATED WORK

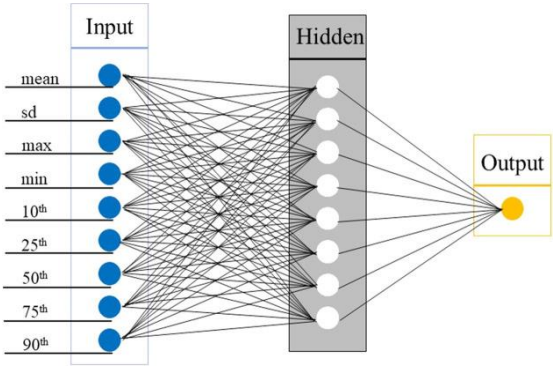

Figure 6. Figure of input and output

One widely used approach for detecting abnormal movement in safety applications is Artificial Neural Networks (ANNs) [11]. ANN models learn complex patterns in accelerometer and gyroscope data by training on large datasets. A study on human activity recognition using ANN models showed that they could effectively classify movement types with high accuracy. However, ANNs require significant computational resources, making them unsuitable for real-time applications on low-power wearable devices. Additionally, data transfer between edge devices and cloud servers can introduce latency, preventing timely detection and response. This method also struggles with transparency, as the black-box nature of ANNs makes it difficult to interpret decisions. Our approach improves upon this by employing a lightweight K-Nearest Neighbors (KNN) model optimized for low-power processors. Unlike ANNs, KNN requires less computational power and provides interpretable results while maintaining high accuracy. By integrating real-time speech recognition with motion analysis, our system enhances safety detection without excessive latency or resource demands.

Support Vector Machines (SVMs) are commonly used for movement classification, distinguishing between activities such as walking, running, and falling by establishing decision boundaries in high-dimensional spaces [12]. While effective, SVMs require extensive feature engineering and hyperparameter tuning, which can be computationally expensive. Additionally, they struggle with imbalanced datasets, often leading to misclassification in rare emergency scenarios. Our approach improves upon SVMs by employing a dynamically trained KNN model, which is computationally efficient and adapts to diverse activity patterns. By integrating motion and voice recognition, we reduce false positives, making the system more reliable for real-time emergency detection in safety applications.

Hidden Markov Models (HMMs) are often used for fall detection and gait analysis, modeling movement as a sequence of states and transitions [13]. However, they rely on predefined models, limiting their adaptability to unpredictable real-world scenarios, such as sudden threats. HMMs also struggle with complex, unstructured environments where movement is highly variable. Our approach improves upon HMMs by incorporating real-time sensor fusion with KNN and speech recognition, allowing for dynamic adjustments to varying conditions. This ensures more accurate detection without the need for predefined states, providing a flexible and responsive safety mechanism suitable for real-world emergency situations.

6. CONCLUSIONS

While the proposed system is a significant advancement in wearable safety technology, it still faces some limitations. One primary limitation is the reliance on real-time data processing, which may be affected by hardware limitations, especially in low-cost or resource-constrained devices. Although we optimized the KNN algorithm for performance, the accuracy and responsiveness of the system could be improved by implementing more advanced models that can handle complex patterns more effectively [14]. Additionally, the dual-detection system (motion and voice recognition) could be enhanced to further reduce the risk of false positives in diverse environments. Another area for improvement is the integration of more diverse datasets to better handle edge cases, such as elderly users or individuals with limited mobility, who may exhibit atypical movement patterns. If given more time, I would work on optimizing the system's processing speed and reducing its energy consumption for extended wearability. Incorporating machine learning models that allow for more adaptive learning could also improve the system's accuracy over time.

In conclusion, our system presents an innovative solution to enhancing female safety through wearable technology [15]. By integrating motion detection and voice recognition, we offer a more reliable and real-time response to emergencies. Despite some limitations, the system shows promising potential in addressing safety concerns and improving personal security for women.

REFERENCES

[1] Congdon, Venetia. "The'lone female researcher': isolation and safety upon arrival in the field." Journal of the Anthropological Society of Oxford Online 7.1 (2015).

[2] Paramasivam, Kalaivani, Mohamed Mansoor Roomi Sindha, and Sathya Bama Balakrishnan. "KNN-based machine learning classifier used on deep learned spatial motion features for human action recognition." Entropy 25.6 (2023): 844.

[3] Kurani, Akshit, et al. "A comprehensive comparative study of artificial neural network (ANN) and support vector machines (SVM) on stock forecasting." Annals of Data Science 10.1 (2023): 183-208.

[4] Gasper, Des, and Oscar A. Gómez. "Human security thinking in practice:'personal security','citizen security'and comprehensive mappings." Contemporary Politics 21.1 (2015): 100-116.

[5] Perrachione, Tyler K., Stephanie N. Del Tufo, and John DE Gabrieli. "Human voice recognition depends on language ability." Science 333.6042 (2011): 595-595.
[6] Weitzer, Ronald. "New directions in research on human trafficking." The ANNALS of the American Academy of Political and Social Science 653.1 (2014): 6-24.
[7] Jennings, Wesley G., Angela R. Gover, and Dagmar Pudrzynska. "Are institutions of higher learning safe? A descriptive study of campus safety issues and self-reported campus victimization among male and female college students." Journal of criminal justice education 18.2 (2007): 191-208.
[8] Zhang, Shichao, et al. "A novel kNN algorithm with data-driven k parameter computation." Pattern Recognition Letters 109 (2018): 44-54.
[9] Shakhovska, Nataliya, Oleh Basystiuk, and Khrystyna Shakhovska. "Development of the Speech-to-Text Chatbot Interface Based on Google API." MoMLeT. 2019.
[10] Gaikwad, Santosh K., Bharti W. Gawali, and Pravin Yannawar. "A review on speech recognition technique." International Journal of Computer Applications 10.3 (2010): 16-24.
[11] Zou, Jinming, Yi Han, and Sung-Sau So. "Overview of artificial neural networks." Artificial neural networks: methods and applications (2009): 14-22.
[12] Hearst, Marti A., et al. "Support vector machines." IEEE Intelligent Systems and their applications 13.4 (1998): 18-28.
[13] Rabiner, Lawrence, and Biinghwang Juang. "An introduction to hidden Markov models." ieee assp magazine 3.1 (1986): 4-16.
[14] Pandey, Amit, and Achin Jain. "Comparative analysis of KNN algorithm using various normalization techniques." International Journal of Computer Network and Information Security 10.11 (2017): 36.
[15] Iqbal, Mohammed H., et al. "A review of wearable technology in medicine." Journal of the Royal Society of Medicine 109.10 (2016): 372-380.