

# Contactless Employee Authentication Using Gait Analysis

A Real-World Sensor-Based Machine Learning System  
with Explainable AI

**Presented by:** Ravi pal

**Institute:** VIT Bhopal University

**Domain:** Machine Learning & Data Science



# PROBLEM STATEMENT

- Traditional employee authentication systems used in offices and organizations face several practical limitations:
- Access cards and ID badges can be lost, stolen, or shared
- Password-based systems are inconvenient and insecure
- Contact-based biometrics such as fingerprints require physical interaction
- Face recognition systems raise privacy and surveillance concerns
- Most existing methods require active effort from the user
- Objective: To design a system that can authenticate employees automatically using their natural walking behavior, without requiring any explicit action.

## SOLUTION: GAIT ANALYSIS:

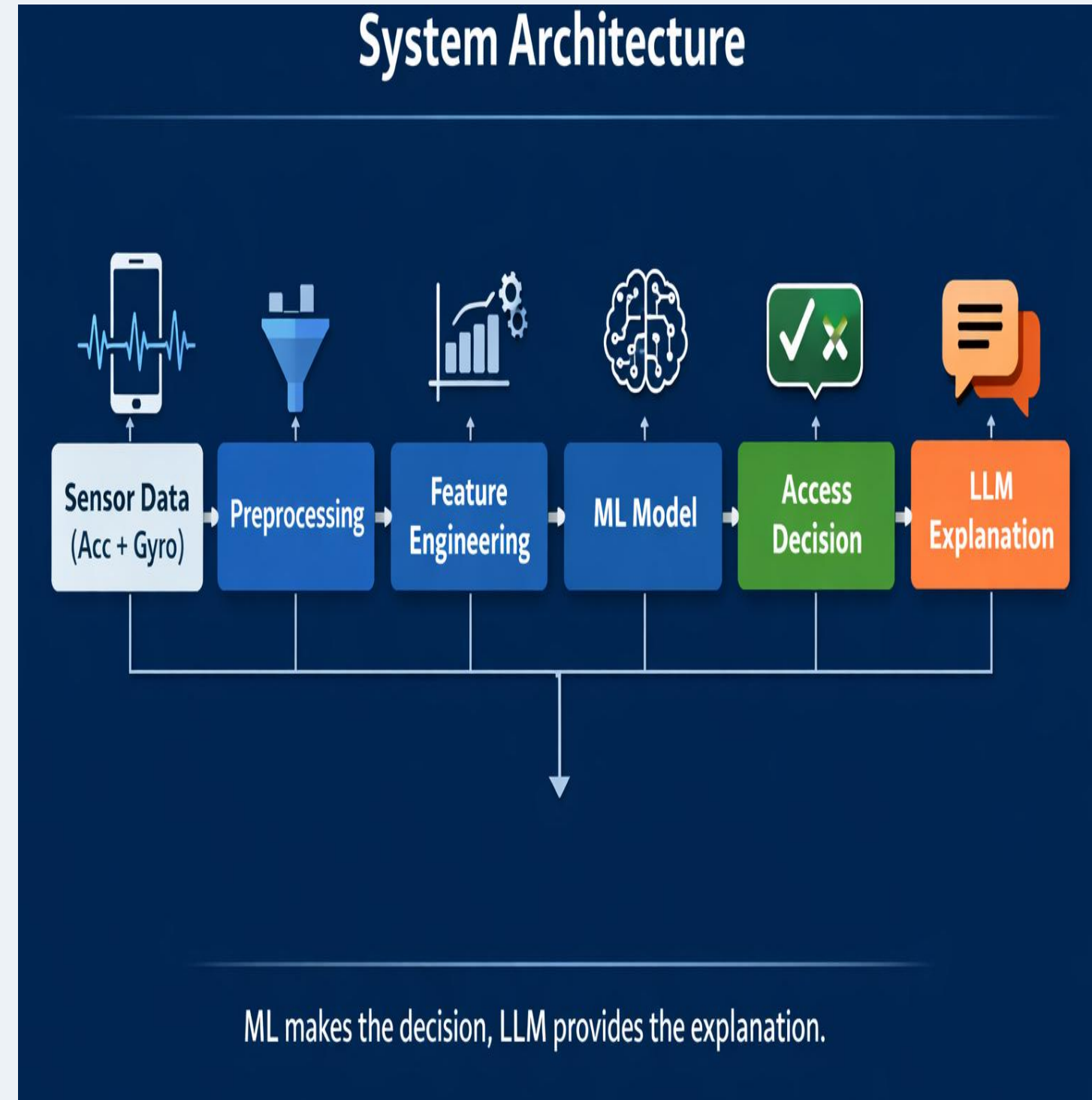
- Gait refers to the way a person walks. It is a **behavioral biometric** that is naturally produced during daily movement.
- Gait analysis is suitable for employee authentication because:
- Every individual has a **unique walking pattern**
- Gait is **difficult to consciously imitate**
- It can be captured **passively** while the user walks
- No physical contact or user interaction is required
- Smartphone sensors can record gait data **without extra hardware**
- Unlike passwords or cards, gait-based authentication works **in the background**, making it both secure and user-friendly.

# SYSTEM ARCHITECTURE

- The proposed system follows a structured pipeline that converts raw sensor data into an access control decision.

## Overall Flow

- Smartphone collects **accelerometer and gyroscope data**
- Raw sensor signals are **preprocessed** to remove noise and gravity
- Signals are segmented into windows and **features are extracted**
- A machine learning model **identifies the employee**
- A confidence-based rule decides **access grant or denial**
- An LLM generates a **human-readable explanation** of the decision

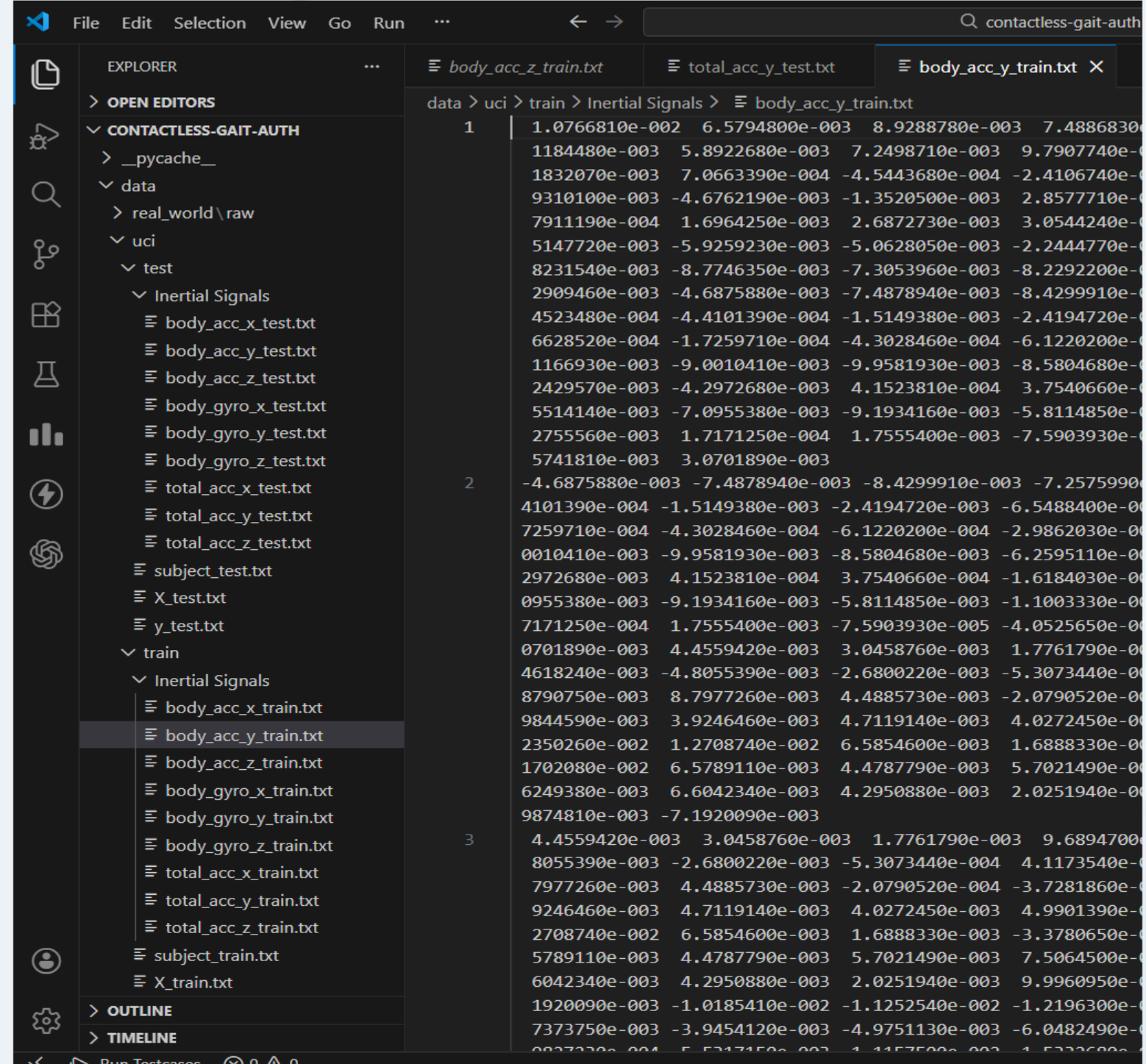




# Data Collection & Preprocessing

## 1. Dataset-Based Training (UCI HAR)

1. Data collected from **30 subjects**
2. Sensors used:
  - **Accelerometer (3-axis)**
  - **Gyroscope (3-axis)**
3. Sampling frequency: **50 Hz**
4. Activities used for gait:
  - Walking
  - Walking Upstairs
  - Walking Downstairs
5. Preprocessed using:
  - Noise filtering
  - Gravity separation (Butterworth filter)
  - Sliding window segmentation (2.56 sec, 128 samples)



The screenshot displays a code editor interface with a dark theme. The Explorer panel on the left shows the project structure for 'CONTACTLESS-GAIT-AUTH', including a 'data' directory with 'real\_world\raw' and 'uci' subdirectories. The 'uci' directory contains 'test' and 'train' subdirectories, each with 'Inertial Signals' and various sensor data files. The 'body\_acc\_y\_train.txt' file is selected and its content is visible in the main editor area. The file contains a large table of numerical data, organized into three columns (1, 2, 3) and multiple rows, representing acceleration data for different subjects and activities.

```
data > uci > train > Inertial Signals > body_acc_y_train.txt
1 | 1.0766810e-002 6.5794800e-003 8.9288780e-003 7.4886830e-003
  | 1184480e-003 5.8922680e-003 7.2498710e-003 9.7907740e-003
  | 1832070e-003 7.0663390e-004 -4.5443680e-004 -2.4106740e-004
  | 9310100e-003 -4.6762190e-003 -1.3520500e-003 2.8577710e-003
  | 7911190e-004 1.6964250e-003 2.6872730e-003 3.0544240e-003
  | 5147720e-003 -5.9259230e-003 -5.0628050e-003 -2.2444770e-003
  | 8231540e-003 -8.7746350e-003 -7.3053960e-003 -8.2292200e-003
  | 2909460e-003 -4.6875880e-003 -7.4878940e-003 -8.4299910e-003
  | 4523480e-004 -4.4101390e-004 -1.5149380e-003 -2.4194720e-003
  | 6628520e-004 -1.7259710e-004 -4.3028460e-004 -6.1220200e-004
  | 1166930e-003 -9.0010410e-003 -9.9581930e-003 -8.5804680e-003
  | 2429570e-003 -4.2972680e-003 4.1523810e-004 3.7540660e-004
  | 5514140e-003 -7.0955380e-003 -9.1934160e-003 -5.8114850e-003
  | 2755560e-003 1.7171250e-004 1.7555400e-003 -7.5903930e-003
  | 5741810e-003 3.0701890e-003
2 | -4.6875880e-003 -7.4878940e-003 -8.4299910e-003 -7.2575990e-003
  | 4101390e-004 -1.5149380e-003 -2.4194720e-003 -6.5488400e-003
  | 7259710e-004 -4.3028460e-004 -6.1220200e-004 -2.9862030e-004
  | 0010410e-003 -9.9581930e-003 -8.5804680e-003 -6.2595110e-003
  | 2972680e-003 4.1523810e-004 3.7540660e-004 -1.6184030e-004
  | 0955380e-003 -9.1934160e-003 -5.8114850e-003 -1.1003330e-003
  | 7171250e-004 1.7555400e-003 -7.5903930e-003 -4.0525650e-003
  | 0701890e-003 4.4559420e-003 3.0458760e-003 1.7761790e-003
  | 4618240e-003 -4.8055390e-003 -2.6800220e-003 -5.3073440e-004
  | 8790750e-003 8.7977260e-003 4.4885730e-003 -2.0790520e-004
  | 9844590e-003 3.9246460e-003 4.7119140e-003 4.0272450e-003
  | 2350260e-002 1.2708740e-002 6.5854600e-003 1.6888330e-003
  | 1702080e-002 6.5789110e-003 4.4787790e-003 5.7021490e-003
  | 6249380e-003 6.6042340e-003 4.2950880e-003 2.0251940e-003
  | 9874810e-003 -7.1920090e-003
3 | 4.4559420e-003 3.0458760e-003 1.7761790e-003 9.6894700e-003
  | 8055390e-003 -2.6800220e-003 -5.3073440e-004 4.1173540e-003
  | 7977260e-003 4.4885730e-003 -2.0790520e-004 -3.7281860e-004
  | 9246460e-003 4.7119140e-003 4.0272450e-003 4.9901390e-003
  | 2708740e-002 6.5854600e-003 1.6888330e-003 -3.3780650e-003
  | 5789110e-003 4.4787790e-003 5.7021490e-003 7.5064500e-003
  | 6042340e-003 4.2950880e-003 2.0251940e-003 9.9960950e-003
  | 1920090e-003 -1.0185410e-002 -1.1252540e-002 -1.2196300e-002
  | 7373750e-003 -3.9454120e-003 -4.9751130e-003 -6.0482490e-003
  | 0027330e-004 5.5317150e-003 1.1157500e-003 1.5333600e-003
```

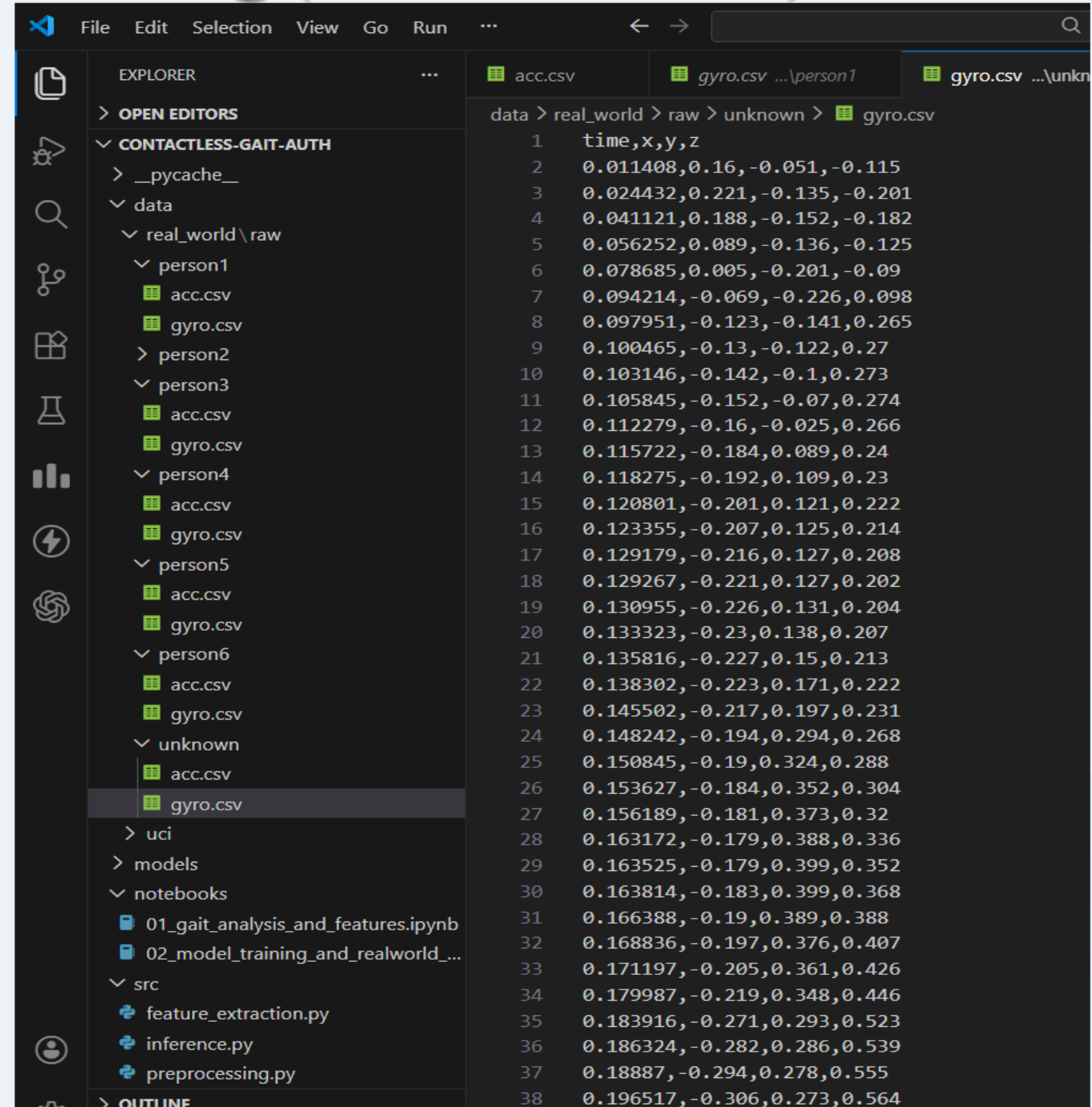
# Data Collection & Preprocessing(Continue...)

## 2. Real-World Data Collection

- Collected using **Physics Toolbox Sensor Suite (Android app)**
- Data collected from **5+ real users**
- Raw data format:
  - time, x, y, z
- Sensors:
  - Accelerometer
  - Gyroscope
- No preprocessing applied at collection time

## 3. Preprocessing Pipeline (Applied by Us)

- Resampling to match **50 Hz**
- Noise filtering (low-pass filter)
- Gravity removal
- Windowing to fixed-size segments
- Feature extraction to match training format



The screenshot shows a code editor interface with a dark theme. On the left, the 'EXPLORER' sidebar displays a file tree structure. The tree is expanded to show a folder named 'real\_world' containing a subfolder 'raw', which in turn contains a folder 'unknown'. Inside 'unknown', there are two files: 'acc.csv' and 'gyro.csv'. The 'gyro.csv' file is selected and its content is displayed in the main editor area on the right. The content of 'gyro.csv' is a list of 38 rows, each containing four numerical values representing time, x, y, and z coordinates. The first row is a header: 'time,x,y,z'. The subsequent rows contain numerical data points.

```
data > real_world > raw > unknown > gyro.csv
1 time,x,y,z
2 0.011408,0.16,-0.051,-0.115
3 0.024432,0.221,-0.135,-0.201
4 0.041121,0.188,-0.152,-0.182
5 0.056252,0.089,-0.136,-0.125
6 0.078685,0.005,-0.201,-0.09
7 0.094214,-0.069,-0.226,0.098
8 0.097951,-0.123,-0.141,0.265
9 0.100465,-0.13,-0.122,0.27
10 0.103146,-0.142,-0.1,0.273
11 0.105845,-0.152,-0.07,0.274
12 0.112279,-0.16,-0.025,0.266
13 0.115722,-0.184,0.089,0.24
14 0.118275,-0.192,0.109,0.23
15 0.120801,-0.201,0.121,0.222
16 0.123355,-0.207,0.125,0.214
17 0.129179,-0.216,0.127,0.208
18 0.129267,-0.221,0.127,0.202
19 0.130955,-0.226,0.131,0.204
20 0.133323,-0.23,0.138,0.207
21 0.135816,-0.227,0.15,0.213
22 0.138302,-0.223,0.171,0.222
23 0.145502,-0.217,0.197,0.231
24 0.148242,-0.194,0.294,0.268
25 0.150845,-0.19,0.324,0.288
26 0.153627,-0.184,0.352,0.304
27 0.156189,-0.181,0.373,0.32
28 0.163172,-0.179,0.388,0.336
29 0.163525,-0.179,0.399,0.352
30 0.163814,-0.183,0.399,0.368
31 0.166388,-0.19,0.389,0.388
32 0.168836,-0.197,0.376,0.407
33 0.171197,-0.205,0.361,0.426
34 0.179987,-0.219,0.348,0.446
35 0.183916,-0.271,0.293,0.523
36 0.186324,-0.282,0.286,0.539
37 0.18887,-0.294,0.278,0.555
38 0.196517,-0.306,0.273,0.564
```

# MODEL TRAINING, EVALUATION & RESULTS

## 1. Problem Formulation

- Multi-class classification problem
- Each employee is treated as **one class**
- Goal: **Identify the employee from gait patterns**

## 2. Model Selection

- **Random Forest Classifier**
- Chosen because:
  - Handles non-linear gait patterns
  - Robust to noise in real-world sensor data
  - Performs well with limited samples
  - No strict feature scaling required

## 3. Training Strategy

- Trained on **walking-related activities only**
  - Walking
  - Walking Upstairs
  - Walking Downstairs
- Feature vectors extracted from sliding windows
- Cross-validation used for performance estimation

## 4. Evaluation Metrics

- Cross-validation accuracy
- Confidence scores per prediction
- Threshold-based decision for access control

## Results (Short & Honest)

- Model shows **high confidence** for enrolled employees
- **Low confidence** for unknown users
- Clear separation between known and unknown gait patterns



FileEditSelectionViewGoRun

contactless-gait-auth

1

EXPLORER

> OPEN EDITORS

CONTACTLESS-GAIT-AUTH

> \_\_pycache\_\_

> data

> real\_world\raw

> person1

acc.csv

gyro.csv

> person2

> person3

acc.csv

gyro.csv

> person4

acc.csv

gyro.csv

> person5

acc.csv

gyro.csv

> person6

acc.csv

gyro.csv

> unknown

acc.csv

gyro.csv

> uci

> models

> notebooks

01\_gait\_analysis\_and\_features.ipynb

02\_model\_training\_and\_realworld\_...

> src

feature\_extraction.py

inference.py

preprocessing.py

> OUTLINE

> TIMELINE

02\_model\_training\_and\_realworld\_test.ipynb

acc.csv

gyro.csv ...\person1

gyro.csv ...\unknown

notebooks > 02\_model\_training\_and\_realworld\_test.ipynb > UNKNOWN\_DIR = Path("../data/real\_world/raw/unknown")

+ Code + Markdown | Run All Restart Clear All Outputs Jupyter Variables Outline

Python 3.12.8

```
# ----- confidence-based filtering -----
valid_preds = [
    (p, c) for p, c in zip(preds, confs)
    if c >= CONFIDENCE_THRESHOLD
]

if len(valid_preds) == 0:
    return "ACCESS DENIED", None, np.mean(confs)

# majority vote
final_label = Counter([p for p, _ in valid_preds]).most_common(1)[0][0]
avg_conf = np.mean([c for _, c in valid_preds])

return "ACCESS GRANTED", reverse_label_map[final_label], avg_conf
```

[36] Python

```
UNKNOWN_DIR = Path("../data/real_world/raw/unknown")

status, person, confidence = predict_unknown_person(UNKNOWN_DIR)

print("Decision  :", status)
print("Person    :", person)
print("Confidence :", round(confidence, 3))
```

[42] Python

```
... Decision  : ACCESS GRANTED
Person    : person1
Confidence : 0.862
```

```
def get_confidences(person_dir):
    acc_df = pd.read_csv(person_dir / "acc.csv")
    gyro_df = pd.read_csv(person_dir / "gyro.csv")
```

[43] Python

# LLM-BASED EXPLAINABILITY & DECISION TRANSPARENCY

## ➤ Why Explainability Is Required

- The machine learning model produces **numerical outputs** such as probabilities and confidence scores
- These outputs are **not easily understandable** by security staff or non-technical users
- For access control systems, it is important to clearly explain **why access was granted or denied**

## ➤ How the LLM Is Used

- After the ML model predicts the employee and confidence score:
  - A short decision summary is created
  - This summary is passed to the LLM
- The LLM converts the decision into a **simple, human-readable explanation**

## ➤ Key Design Principle

- The **ML model makes the decision**
- The **LLM only explains the decision**