**Project Code Description**

## Camera Testing Code Description

**Pre-Processing Techniques:**

* Conversion: Converts video frames from BGR to RGB for MediaPipe compatibility.
* Normalisation: Normalises pose landmarks relative to frame size.

**AI Techniques and Algorithm:**

* Algorithm: SVM trained to differentiate between "fall" and "no fall" states.
* Features: Utilises thirteen critical pose landmarks from MediaPipe.

**Key Features:**

* Real-time Detection: Analyses streaming video for immediate fall detection.
* Notification System: Sends alerts via Pushover API upon detecting a fall, enabling prompt response.

**Parameters and Hyperparameters:**

* MediaPipe Settings: Configured for optimal balance between detection confidence and tracking.
* SVM Configuration: Tuned for precision in fall identification.

**Experimentation and Testing:**

* Evaluation: Performed in real-time with focus on accuracy and system responsiveness.
* Metrics: Uses accuracy, speed, and notification delivery time.

**Technical Requirements:**

* Software: Python, MediaPipe, OpenCV, scikit-learn.
* Hardware**:** Requires a camera and processing capability for real-time video analysis.

**Future Work:**

* Enhancements: Explore advanced ML models for accuracy, broaden notification options, and improve environmental adaptability.

## MP4 Testing Code Description

**Pre-Processing Techniques:**

* Conversion and Normalisation: Adapts MP4 video frames to a consistent format and scale suitable for pose analysis.

**AI Techniques and Algorithm:**

* Algorithm: Uses a pre-trained SVM to classify video frames as "fall" or "no fall" based on pose landmarks.
* Features: Extracts and analyses key pose landmarks using MediaPipe for accurate fall detection.

**Key Features:**

* Offline Video Analysis: Tests the model on pre-recorded MP4 video files to assess detection efficacy.
* Batch Processing: Handles multiple files sequentially, demonstrating the system’s scalability and robustness.

**Parameters and Hyperparameters:**

* MediaPipe and SVM Settings: Optimised for accurate and efficient video frame analysis.

**Experimentation and Testing:**

* Evaluation Method: Performance tested on diverse video datasets to validate detection accuracy.
* Metrics: Focuses on precision, recall, and computational efficiency.

**Technical Requirements:**

* Software: Requires Python, MediaPipe, OpenCV, scikit-learn, and dependencies for video processing.
* Hardware: Compatible with standard computational setups capable of handling video data processing.

**Future Work:**

* System Enhancements: Plans to integrate real-time streaming capabilities and improve detection algorithms for better accuracy and lower false positives.

## 

## 

## 

## Final Pose Detection Tool Code Description

**Pre-Processing Techniques:**

* Frame Handling: Frames from an MP4 video are read and processed to normalise colours and enhance landmark visibility for pose detection.
* Sampling: Processes every 5th frame to optimise computational efficiency without sacrificing accuracy.

**AI Techniques and Algorithm:**

* Algorithm: MediaPipe Pose, a machine learning framework that provides real-time pose detection.
* Features: Utilises specific body landmarks like shoulders, elbows, hips, and knees to determine human poses accurately.

**Key Features:**

* Landmark Extraction: Identifies and extracts significant body landmarks from video frames, enhancing detailed pose analysis.
* Visualisation: Marks extracted landmarks and draws connections between them on video frames for visual verification.

**Parameters and Hyperparameters:**

* MediaPipe Configuration: Configured with standard pose detection parameters to ensure robust and accurate landmark tracking across various frames.

**Experimentation and Testing:**

* Testing Mechanism: Evaluated by processing a series of frames from pre-recorded videos, verifying landmark accuracy visually.
* Metrics: Focuses on the consistency and precision of landmark detection across processed frames.

**Technical Requirements:**

* Software: Python, OpenCV for video processing, MediaPipe for pose detection.
* Hardware: Compatible with standard PCs, preferably with GPU support for faster processing.

**Data Management:**

* CSV Data Handling: Extracts and records landmark data into a CSV file for further analysis or machine learning training.
* Data Transformation: Processes raw CSV data to pivot tables, aligning landmarks across frames for easier analysis.

**Future Work:**

* Enhancements: Plans to integrate more complex analysis tools, improve real-time processing capabilities, and expand landmark accuracy under varied conditions.

## CVAT Annotation and CSV File Merging Tool Code Description

**Data Integration Techniques:**

* XML Parsing: Utilises ElementTree to load and parse XML files containing frame annotations (Fall/No Fall) exported from CVAT.
* Data Frame Creation: Converts annotation data into a structured Pandas DataFrame for ease of manipulation.

**AI Techniques and Algorithm:**

* Data Merging: Employs Pandas to merge pose landmark data (from a previously generated CSV file) with frame annotations based on frame numbers.
* Feature Integration: Aligns and combines multiple data sources to prepare a comprehensive dataset for training machine learning models.

**Key Features:**

* Annotation Extraction: Extracts and organises annotation labels from XML, assigning them to corresponding video frame numbers.
* Robust Data Merging: Ensures accurate alignment of pose data with annotation labels, preserving data integrity across frames.

**Parameters and Hyperparameters:**

* Merge Strategy: Configured to perform a left join on frame numbers, ensuring no loss of pose data while integrating annotations.

**Experimentation and Testing:**

* Validation: Displays initial rows of merged data to verify correct integration of pose landmarks with annotations.
* Metrics: Focuses on data completeness and correctness in the merged dataset.

**Technical Requirements:**

* Software: Python, Pandas for data manipulation, and XML handling libraries.
* Hardware: Requires standard computational resources capable of handling data processing tasks.

**Data Management:**

* CSV Operations: Outputs the final merged dataset to a CSV file, ready for use in model training.
* File Management: Carefully manages file paths to avoid overwriting existing data, ensuring traceability and version control.

**Future Work:**

* Enhancements: Plans to automate the merging process further, incorporate error handling for missing or mismatched data, and optimise data processing for larger datasets.

## Fall Detection Support Vector Machine Training/Testing Code Description

**Pre-Processing Techniques:**

* Feature Order: Correct the order of features to be in the correct format ready for training.
* Standardise Annotations: Replace lowercase versions of the annotations with upper case strings to be consistent.

**AI Techniques and Algorithm:**

* Support Vector Machine: Implement the already trained SVM into the GUI to make predictions on the video stream.

**Key Features:**

* Training: Main feature of the code is to train a SVM to predict when someone falls based on the landmark coordinates. Final model had an accuracy of 93% in comparison to the first model which had accuracy of 65%.
* Dataset: Can load all data from dataset, check if all data is standardised and split the data up into training and testing portions. Prepares data for SVM model.
* Loading/Saving: Can load and save a model for future work. Doesn’t require going through the training process again.

**Experimentation and Testing:**

* Metrics: Gauge performance of the model using various metrics such as accuracy, confusion matrix, F1 score, recall and precision score.
* Testing: Check the performance of the model by making predictions on the testing portion of the dataset.
* Predictions: Model can make predictions on testing data to check if the model is making accurate predictions or not.

**Technical Requirements:**

* Software: Python. Libraries include Pandas, Numpy, JobLib, SKLearn and OS.
* Hardware: Requires any computer with enough processing power to run Jupyter Notebooks. Can be run in standard Python or Google Colab.

**Future Work:**

* Enhancements: Improve the performance and quality of the model by increasing the amount of data for both training and testing.
* Experiments: Experiment with other models similar to SVM to check performance and find a more accurate model.

## Testing Camera GUI Code Description

**Pre-Processing Techniques:**

* Colour Correction: Correct colours of the video stream being transmitted from a phone camera using DroidCam.

**AI Techniques and Algorithm:**

* Support Vector Machine: The algorithm of choice for this project was a support vector machine. Implemented using SKLearn libraries.

**Key Features:**

* Predictions: Can make predictions in real time by using frames from the video stream. Make a prediction 30 times a second.
* Landmark Coordinates: Displays the coordinates of all the landmarks each video frame. Useful for checking if pose detection model is working correctly.
* Skeleton Model: Displays a skeleton model on the video stream when a person is detected. Skeleton is green when there is no fall and red when there is a fall.

**Parameters and Hyperparameters:**

* Video Type: Can switch between MP4 videos or live video stream from cameras to check performance of the model.

**Experimentation and Testing:**

* Testing: Check the performance of the model by making predictions on the video stream.
* Predictions: Model can make predictions on the video stream to check if the model is making accurate predictions or not.

**Technical Requirements:**

* Software: Python. Libraries include OpenCV, MediaPipe, Numpy, JobLib, SKLearn and TKinter.
* Hardware: Requires any computer with enough processing power to run Jupyter Notebooks. Can be run in standard Python or Google Colab.

**Future Work:**

* UI: Improve the look and feel of the UI to make it more presentable for an average user.
* Options: Include the options to switch between videos and video streams without having to modify code.
* Notifications: Include notification system to work with the UI similar to the regular testing code.