

**CSE1901 - Technical Answers to Real World Problems  
(TARP)**

**Project Report**

**Title**

*By*

Reg. No: 19BAI1153

Name: Sagnik Chatterjee

B. Tech Computer Science and Engineering

*Submitted to*

**Premalatha M**

**School of Computer Science and Engineering**

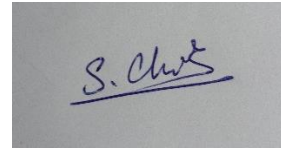


**VIT<sup>®</sup>**  
**Vellore Institute of Technology**  
(Deemed to be University under section 3 of UGC Act, 1956)

*April 2022*

## **DECLARATION**

I hereby declare that the report titled “**Analysis of Body Gestures and Detecting Psychological Distress**” submitted by me to VIT Chennai is a record of bona-fide work undertaken by me under the supervision of **Premalatha M**, School of Computer Science and Engineering, Vellore Institute of Technology, Chennai.

A rectangular box containing a handwritten signature in blue ink. The signature appears to be 'S. Chatterjee' with a stylized flourish at the end.

Signature of the Candidate

**Sagnik Chatterjee**

**19BAI1153**

## **CERTIFICATE**

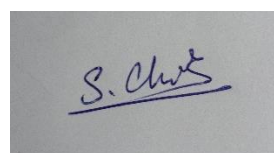
Certified that this project report entitled “**Analysis of Body Gestures and Detecting Psychological Distress**” is a bonafide work of **Nishant Chaturvedi (19BAI1109)**, **Sagnik Chatterjee (19BAI1153)** and they carried out the Project work under my supervision and guidance for CSE1901 - Technical Answers to Real World Problems (TARP).

**Premalatha M**  
SCOPE, VIT Chennai

## **ACKNOWLEDGEMENT**

I would like to thank my “Technical Answers to Real World Problems (TARP)” teacher, Premalatha M who gave me a golden opportunity to work on this project. I’d also like to express my gratitude to my school dean wholeheartedly.

I must also thank my parents and project partner for the immense support and help during this project. Without their help, completing this project would have been very difficult.

A rectangular box containing a handwritten signature in blue ink. The signature appears to be 'S. Chatterjee' written in a cursive style.

**Sagnik Chatterjee**

**Reg. No. 19BAI1153**

## **ABSTRACT**

Psychological distress is a significant and growing issue in society. Automatic detection, assessment, and analysis of such distress is an active area of research. Compared to modalities such as face, head, and vocal, research investigating the use of the body modality for these tasks is relatively bare. This is, in part, due to the limited available datasets and difficulty in automatically extracting useful body features. Recent advances in pose estimation and deep learning have enabled new approaches to this modality and domain. For this project, we will collect and analyze a new dataset containing full body videos for example short interviews and self-reported distress labels. This project will take a step forward to investigate body gesture analyzing methods and develop a program that bridges the daily human life habits and mental health issues.

## **CONTENTS**

	Declaration	i
	Certificate	ii
	Acknowledgement	iii
	Abstract	iv
1	Introduction	1
1.1	Objective and goal of the project .....	1
1.2	Problem Statement. ....	2
1.3	Motivation .....	3
1.4	Challenges .....	5
2	Literature Survey	6
3	Requirements Specification	7
3.1	Hardware Requirements .....	8
3.2	Software Requirements .....	9
4	System Design	10
5	Implementation of System	
6	Results & Discussion	
7	Conclusion and Future Work	
8	References	
	Appendix <Sample code, snapshot etc.>	

# **1. Introduction**

## **1.1 Objective and goal of the project**

Our objectives for this project is to evaluate the different object detection techniques and filtering algorithms to perform body gesture analysis. Upon successful detection, we would develop software solution that will take the detection results and convert them to psychological distress labels.

## **1.2 Problem Statement**

Early detection of distress is consistently noted as a key factor in treatment and positive outcomes. Early detection requires an ongoing assessment to identify distress when it begins. This project will take a step forward to investigate body gesture analyzing methods and develop a program that bridges the daily human life habits and mental health issues.

## **1.3 Motivation**

Psychological distress and mental disorders are significant threats to global health. According to the World Health Organization (WHO), an estimated 450 million people around the world suffer from neuropsychiatric conditions, with depression and anxiety being the most common mental disorders. Despite existing strategies for the treatment of distress, such as depression, it is estimated that nearly two-thirds of people suffering distress have never received help from a health professional. Early detection of distress is consistently noted as a key factor in treatment and positive outcomes. Early detection requires an ongoing assessment to identify distress when it begins. Currently, the most effective automated distress detection approaches utilize multi-modal machine learning.

#### **1.4 Challenges**

Compared to modalities such as face, head, and vocal, research investigating the use of the body modality for these tasks is relatively bare. This is, in part, due to the limited available datasets and difficulty in automatically extracting useful body features.

Currently, the most effective automated distress detection approaches utilize multi-modal machine learning.

## **2. Literature Survey**

### **2.1 Robo-Nanny: ConvNets for Intelligent Baby Monitoring**

In this paper they have applied supervised learning techniques on state-of-the-art deep convolutional neural networks (CNN) to infer a baby's status inside a crib using a baby monitor video feed.

A dataset of 2500 images assigned to 5 predefined classes was used for training and hyper parameter tuning. After comparing different model architectures, the best class-weighted accuracy of 96.7% on test set was achieved on pre-trained ResNet18.

A Nest IP Camera was placed over a baby crib and a continuous stream of frames were captured at a rate of 1 frame per 10 seconds. Two techniques were briefly tested in order to efficiently extract salient images from the video feed: image subtractions, Optical Flow.

### **2.2 A Study of Vision based Human Motion Recognition and Analysis**

This paper discusses applications, general framework of human motion recognition, and the details of each of its components. The paper emphasizes on human motion representation and the recognition methods along with their advantages and disadvantages.

Human motion can be either directly recognized from image sequences, or it can be done in a multiple layer process. Generally, for simple actions, motion is recognized directly from image sequences and they can be viewed as single layer approaches.



For baby monitoring system we will use computer vision to detect activity or motion of baby. So basically, detecting baby motion is similar to human motion detection. By using this research, we will get an idea about different data sets and different methods to detect human activity and detection.

### **2.3 Looking at The Body: Automatic Analysis of Body Gestures and Self-Adaptors in Psychological Distress**

In today's society, psychological anguish is a big and developing problem. The automatic identification, appraisal, and analysis of such distress is a hot topic in the scientific community. In comparison to modalities like face, head, and voice, research on the use of the body modality for these activities is limited. This is due in part to the scarcity of datasets and the difficulty of collecting useful body information mechanically. New methods to this modality and area have been enabled by recent advancements in posture estimation and deep learning. We collected and analysed a novel dataset combining full body footage for short interviews and self-reported distress labels to enable this research. We provide a new method for automatically detecting self-adaptors and fidgeting, a subset of self-adaptors that has been linked to psychological distress. We examine how participants' activities are affected by distress levels by analysing statistical body movements and fidgeting traits. Then, utilising Multi-modal Deep Denoising Auto-Encoders and Improved Fisher Vector Encoding, we offer a multi-modal strategy that integrates diverse feature representations. In a dataset labelled with self-reported anxiety and depression levels, we show that our proposed model, which combines audio-visual elements with automatically detected fidgeting behavioural indicators, can reliably predict distress levels.

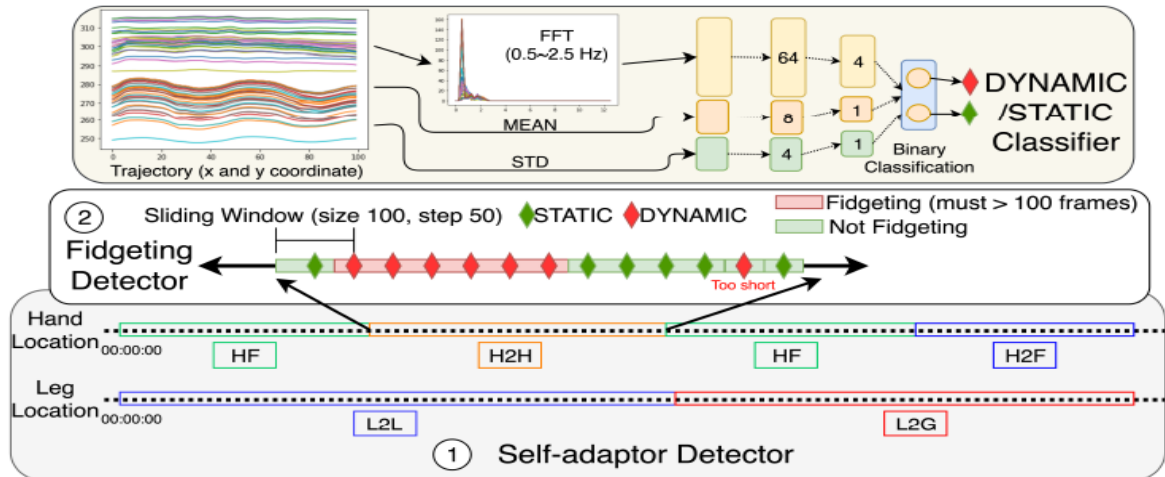


Fig. 1. Hierarchical self-adaptor detection workflow. (1) First, detect hand/leg location; (2) Classify motion using *DYNAMIC/STATIC Classifier* and then finally combine location and motion to give high-level fidgeting event. The figure shows the detection of H<sub>2</sub>H (Hand to hand) fidget. The same principle applies to other fidgets.

### 3 Requirements Specification

#### 3.1 Hardware Requirements

- i) Minimum 4 Gb of RAM
- ii) Decent GPU (minimum GTX 1550 2GB)
- iii) Integrated/Non Integrated Web Cam

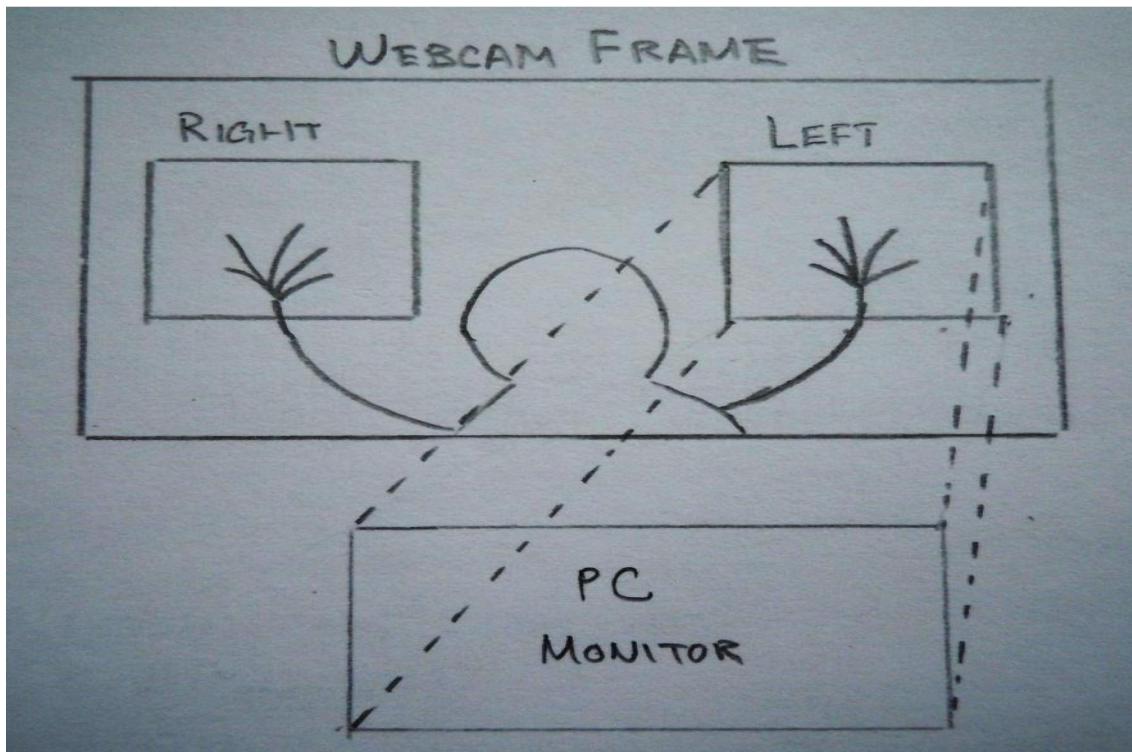
#### 3.2 Software Requirements

- i) OpenCV
- ii) Python 3.0
- iii) Pandas
- iv) numpy
- v) pickle
- vi) csv
- vii) Scikit-learn
- viii) MediaPipe Holistic

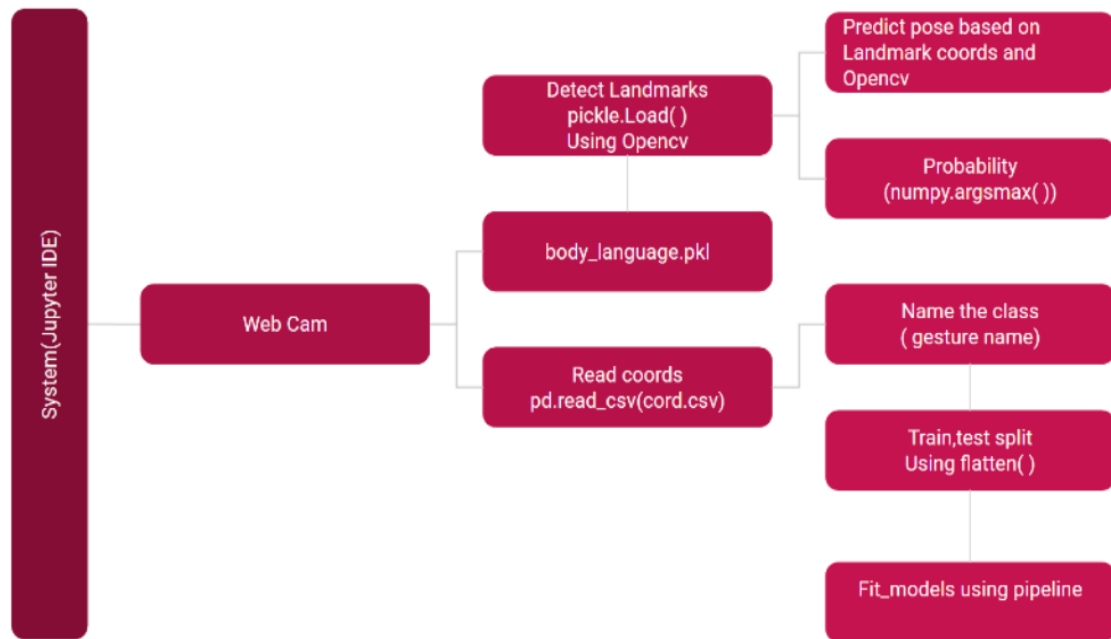
## 4 System Design

The Basic functional design of this project follows:

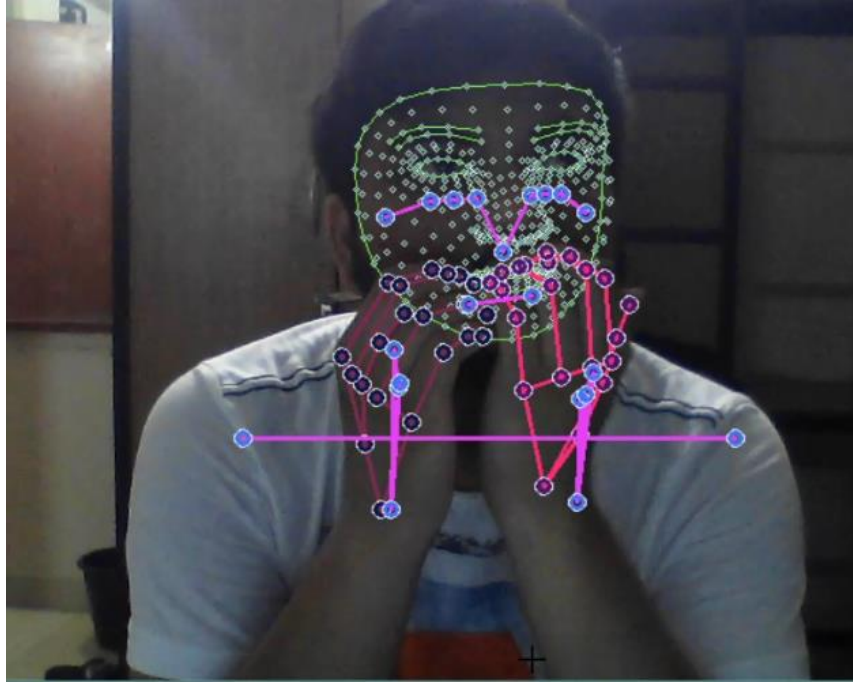
- (a) **Import dependencies:** In this step mediapipe, opencv, pandas and scikit-learn are installed and imported.
- (b) **Perform some detections:** Using cv2, a webcam window is opened, and specifications such as the colour and thickness of our face, as well as hand and pose landmarks, are mentioned/modified.
- (c) **Capture landmarks and export to CSV:** Various facial expressions and body gestures are captured and exported to a csv file called coords.csv.
- (d) **Training using Scikit-learn:** Here, the collected data is read and processed in order to train our machine learning classification model.  
After that, the model is evaluated and serialised.
- (e) **Make detections with the model:** When the code is run, the trained model can now predict the user's gestures.



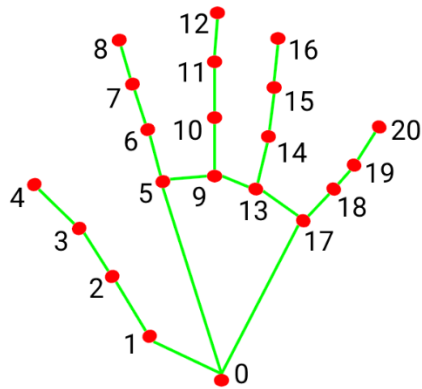
## 5 Implementation of System



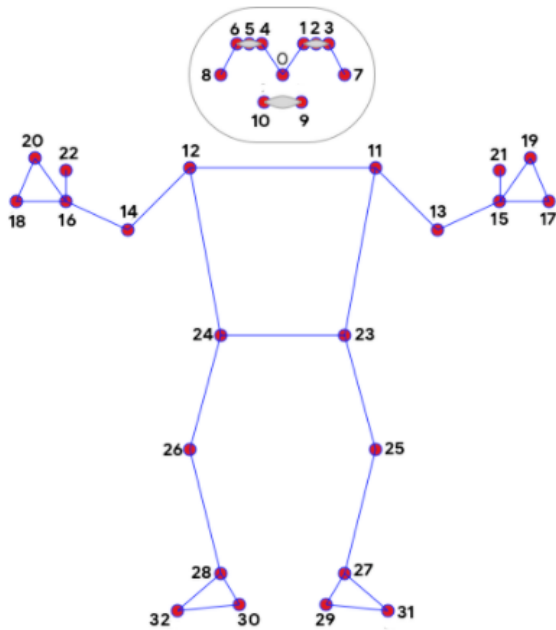
Initially, we used the Jupyter IDE to run the code. When the code is run, the camera begins to operate, we use the MediaPipe holistic to capture our landmarks namely pose, hand and facial which are 501 in total. It reads the coordinates using holistic MEDIAPIPE and writes them in csv file for each instance (frame) for different psychological distress labels such as angry, anxiety, fatigue etc. For each label we train approximately 200 rows. Then finally we train the data after spilling by using scikit-learn and made models by making pipeline of various models like ridge classifier, logistic regression, random forest etc., out of these we use random forest to create our model and save it using pickle module. (Other algorithms can also be used to train the final model)



For the next part we first load our previously created model. It then reads the coordinates using holistic MEDIAPIPE and compares them to the previously trained body language.pkl file. The gesture is then predicted. It also forecasts pose correctness by comparing it to a pre-trained gesture using the `numpy-argmax()` function from the Numpy library. It also displays the max probability for which it is assigning the label. When we run the code to train the model, the web cam begins to run. It uses the holistic function from the MediaPipe library to read the coords from our body and writes them to the coords.csv file. The class is named after the gesture we are learning. The `flatten()` function from the scikit-learning ML library is then used to cluster these coordinates.



- 0. WRIST
- 1. THUMB\_CMC
- 2. THUMB\_MCP
- 3. THUMB\_IP
- 4. THUMB\_TIP
- 5. INDEX\_FINGER\_MCP
- 6. INDEX\_FINGER\_PIP
- 7. INDEX\_FINGER\_DIP
- 8. INDEX\_FINGER\_TIP
- 9. MIDDLE\_FINGER\_MCP
- 10. MIDDLE\_FINGER\_PIP
- 11. MIDDLE\_FINGER\_DIP
- 12. MIDDLE\_FINGER\_TIP
- 13. RING\_FINGER\_MCP
- 14. RING\_FINGER\_PIP
- 15. RING\_FINGER\_DIP
- 16. RING\_FINGER\_TIP
- 17. PINKY\_MCP
- 18. PINKY\_PIP
- 19. PINKY\_DIP
- 20. PINKY\_TIP



- 0. nose
- 1. left\_eye\_inner
- 2. left\_eye
- 3. left\_eye\_outer
- 4. right\_eye\_inner
- 5. right\_eye
- 6. right\_eye\_outer
- 7. left\_ear
- 8. right\_ear
- 9. mouth\_left
- 10. mouth\_right
- 11. left\_shoulder
- 12. right\_shoulder
- 13. left\_elbow
- 14. right\_elbow
- 15. left\_wrist
- 16. right\_wrist
- 17. left\_pinky
- 18. right\_pinky
- 19. left\_index
- 20. right\_index
- 21. left\_thumb
- 22. right\_thumb
- 23. left\_hip
- 24. right\_hip
- 25. left\_knee
- 26. right\_knee
- 27. left\_ankle
- 28. right\_ankle
- 29. left\_heel
- 30. right\_heel
- 31. left\_foot\_index
- 32. right\_foot\_index

## 6. Results and Discussion

Now we will see some of the results that we obtained from training our model and then making detections by using that model.

```
fit_models['rf'].predict(X_test)

array(['Anxiety', 'Fatigue', 'Happy', 'Angry', 'Normal', 'Normal',
       'Happy', 'Angry', 'Anxiety', 'Happy', 'Fatigue', 'Anxiety',
       'Fatigue', 'Normal', 'Fatigue', 'Anxiety', 'Angry', 'Anxiety',
       'Normal', 'Angry', 'Fatigue', 'Anxiety', 'Normal', 'Angry',
       'Normal', 'Fatigue', 'Anxiety', 'Anxiety', 'Fatigue', 'Happy',
       'Normal', 'Anxiety', 'Normal', 'Anxiety', 'Fatigue', 'Fatigue',
       'Happy', 'Fatigue', 'Anxiety', 'Anxiety', 'Happy', 'Normal',
       'Angry', 'Fatigue', 'Happy', 'Happy', 'Normal', 'Normal',
       'Fatigue', 'Normal', 'Fatigue', 'Anxiety', 'Normal', 'Happy',
       'Normal', 'Happy', 'Normal', 'Happy', 'Normal', 'Anxiety',
       'Normal', 'Normal', 'Happy', 'Anxiety', 'Anxiety', 'Anxiety',
       'Angry', 'Happy', 'Angry', 'Normal', 'Fatigue', 'Angry', 'Happy',
       'Anxiety', 'Anxiety', 'Anxiety', 'Happy', 'Normal', 'Happy',
       'Angry', 'Normal', 'Normal', 'Angry', 'Happy', 'Normal', 'Happy',
       'Fatigue', 'Normal', 'Normal', 'Normal', 'Normal', 'Anxiety',
       'Angry', 'Fatigue', 'Fatigue', 'Anxiety', 'Anxiety', 'Normal',
       'Anxiety', 'Anxiety', 'Anxiety', 'Normal', 'Anxiety', 'Normal',
       'Angry', 'Happy', 'Anxiety', 'Fatigue', 'Anxiety', 'Anxiety',
       'Angry', 'Fatigue', 'Normal', 'Anxiety', 'Anxiety', 'Anxiety',
       'Anxiety', 'Happy', 'Angry', 'Happy', 'Anxiety', 'Angry',
```

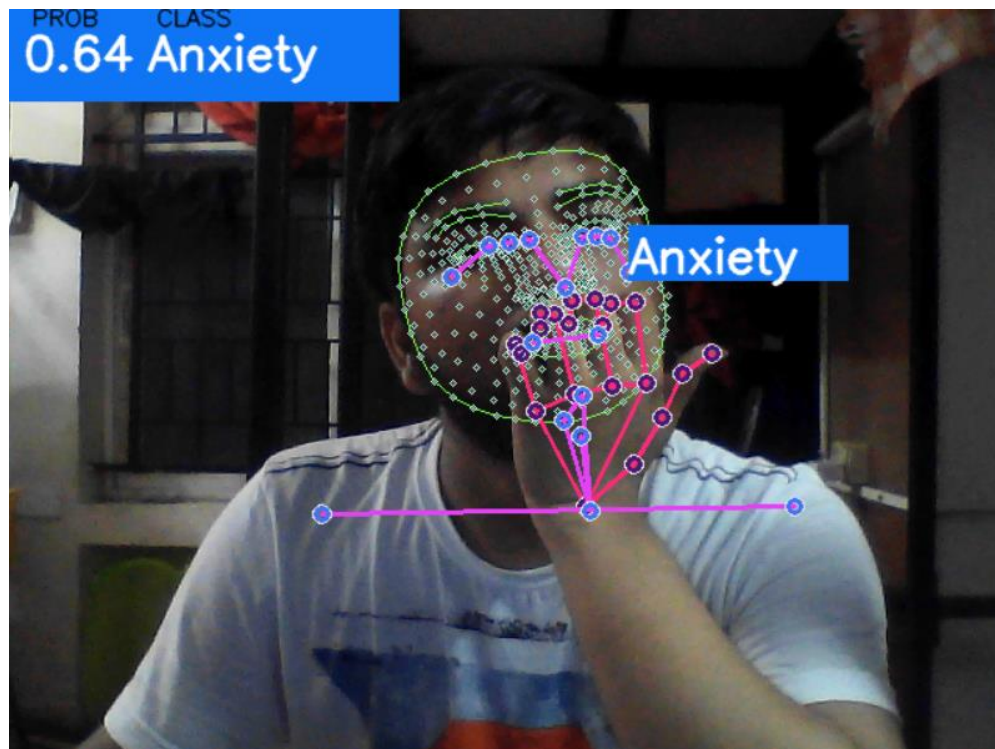
As we can see our model is predicting labels for the coordinates that are in test set (0.2).

```
for algo, model in fit_models.items():
    yhat = model.predict(X_test)
    print(algo, accuracy_score(y_test, yhat))

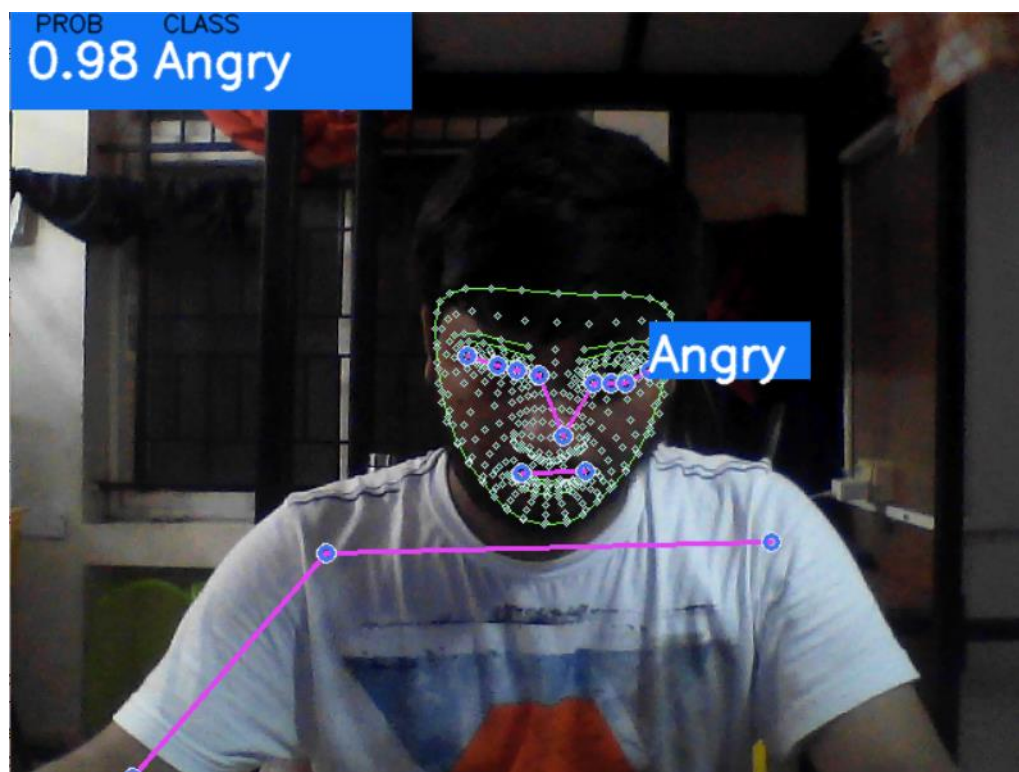
lr 1.0
rc 0.9967532467532467
rf 0.9902597402597403
gb 0.9967532467532467
```

Here we can see the accuracy results for the various models in the pipeline that we made.





As we can see here our model is predicting the labels for some of the acting of psychological distress here.





## **7. Conclusion and Future Work**

Body language detection and analysis has recently received a lot of attention. The ability to recognize and analyze client/customer facial expressions aids organizations and marketing teams in obtaining honest assessments and feedback. However, face expression is only one aspect of body language. Other characteristics of body language include hand gestures and body positions. And when it comes to communicating, body language is crucial. In interviews, for example, interviewers analyze the candidate's body language. By improving this project, interviewers will have access to a tool that will help them understand how the candidate responds when presented questions from other domains or placed in various scenarios during HR rounds. The dataset created here was by us so it was obviously not that accurate. But if the dataset is created by professionals then it will give great results. Hand sign language detection is possible with this project because it offers real-time hand landmark detection. Not only that, but by utilizing this project, existing projects such as driver sleepiness detection, action detection, and others may be made much easier to execute with much better results.

## 8. REFERENCES

- [1] A Framework for Whole-Body Gesture Recognition from Video Feeds - C. N. Joseph, S. Kokulakumaran, K. Srijevanth, A. Thusyanth, C. Gunasekara, and C. D. Gamage Department of Computer Science and Engineering University of Moratuwa, Sri Lanka
- [2] Mental stress detection using bioradar respiratory signals - José Raúl Machado Fernández, Lesya Anishchenko
- [3] Looking At The Body: Automatic Analysis of Body Gestures and Self-Adaptors in Psychological Distress - Weizhe Lin, Member, IEEE, Indigo Orton, Qingbiao Li, Gabriela Pavarini, and Marwa Mahmoud, Member, IEEE
- [4] Review on psychological stress detection using biosignals - Giorgos Giannakakis, *Member, IEEE*, Dimitris Grigoriadis, Katerina Giannakaki, Olympia Simantiraki, Alexandros Roniotis and Manolis Tsiknakis, *Member, IEEE*
- [5] A survey of machine learning techniques in physiology based mental stress detection systems - Suja Sreeith Panicker, Prakasam Gayathri
- [6] <https://arxiv.org/pdf/1608.06761.pdf>
- [7] <https://www.ijariit.com/manuscript/ai-body-language-decoder-using-mediapipe-and-python/>
- [8] <http://personal.ee.surrey.ac.uk/Personal/S.Hadfield/papers/Stoll19.pdf>
- [9] <https://arxiv.org/pdf/1608.06761.pdf>

# APPENDIX

## 0. Install and Import Dependencies

```
In [1]: !pip install mediapipe opencv-python pandas scikit-learn
import mediapipe as mp #
import cv2 # Import opencv
from sklearn.metrics import accuracy_score # Accuracy metrics
import pickle
import pandas as pd
from sklearn.model_selection import train_test_split
import csv
import os
import numpy as np
mp_drawing = mp.solutions.drawing_utils # Drawing helpers
mp_holistic = mp.solutions.holistic # Mediapipe Solutions
import mediapipe as mp # Import mediapipe
import cv2 # Import opencv
```

Requirement already satisfied: mediapipe in c:\users\tnahs\anaconda3\lib\site-packages (0.8.9.1)  
Requirement already satisfied: opencv-python in c:\users\tnahs\anaconda3\lib\site-packages (4.5.4.58)

## 1. Make Some Detections

```
In [6]: cap = cv2.VideoCapture(0)
# Initiate holistic model
with mp_holistic.Holistic(min_detection_confidence=0.5, min_tracking_confidence=0.5) as holistic:

    while cap.isOpened():
        ret, frame = cap.read()

        # Recolor Feed
        image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        image.flags.writeable = False

        # Make Detections
        results = holistic.process(image)
        # print(results.face_landmarks)

        # face_landmarks, pose_landmarks, left_hand_landmarks, right_hand_landmarks

        # Recolor image back to BGR for rendering
        image.flags.writeable = True
        image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)

        # 1. Draw face Landmarks
        mp_drawing.draw_landmarks(image, results.face_landmarks, mp_holistic.FACEMESH_CONTOURS,
                                  mp_drawing.DrawingSpec(color=(80,110,10), thickness=1, circle_radius=1),
                                  mp_drawing.DrawingSpec(color=(80,256,121), thickness=1, circle_radius=1)
                                  )

        # 2. Right hand
        mp_drawing.draw_landmarks(image, results.right_hand_landmarks, mp_holistic.HAND_CONNECTIONS,
                                  mp_drawing.DrawingSpec(color=(80,22,10), thickness=2, circle_radius=4),
                                  mp_drawing.DrawingSpec(color=(80,44,121), thickness=2, circle_radius=2)
                                  )

        # 3. Left Hand
        mp_drawing.draw_landmarks(image, results.left_hand_landmarks, mp_holistic.HAND_CONNECTIONS,
                                  mp_drawing.DrawingSpec(color=(121,22,76), thickness=2, circle_radius=4),
                                  mp_drawing.DrawingSpec(color=(121,44,250), thickness=2, circle_radius=2)
                                  )
```

```
        mp_drawing.DrawingSpec(color=(121,44,250), thickness=2, circle_radius=2)
        )

    # 4. Pose Detections
    mp_drawing.draw_landmarks(image, results.pose_landmarks, mp_holistic.POSE_CONNECTIONS,
                              mp_drawing.DrawingSpec(color=(245,117,66), thickness=2, circle_radius=4),
                              mp_drawing.DrawingSpec(color=(245,66,230), thickness=2, circle_radius=2)
                              )

    cv2.imshow('Raw Webcam Feed', image)

    if cv2.waitKey(10) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

```
results.face_landmarks.landmark[0].visibility
```

```
In [3]: import csv
import os
import numpy as np
```

```
In [30]: num_coords = len(results.pose_landmarks.landmark)+len(results.face_landmarks.landmark)
num_coords
```

Out[30]: 501

```
In [31]: landmarks = ['class']
         for val in range(1, num_coords+1):
             landmarks += ['x{}'.format(val), 'y{}'.format(val), 'z{}'.format(val), 'v{}'.format(val)]
```

```
In [11]: landmarks
```

```
Out[11]: ['class',
          'x1',
          'y1',
          'z1',
          'v1',
          'x2',
          'y2',
          'z2',
          'v2',
          'x3',
          'y3',
          'z3',
          'v3',
          'x4',
          'y4',
          'z4',
          'v4',
          'x5',
          'y5',
          'z5',
          'v5']
```

```
In [31]: class_name = "Anxiety"
```

```
In [36]: cap = cv2.VideoCapture(0)
# Initiate holistic model
with mp_holistic.Holistic(min_detection_confidence=0.5, min_tracking_confidence=0.5) as holistic:

    while cap.isOpened():
        ret, frame = cap.read()

        # Recolor Feed
        image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        image.flags.writeable = False

        # Make Detections
        results = holistic.process(image)
        # print(results.face_Landmarks)

        # face_Landmarks, pose_Landmarks, Left_hand_Landmarks, right_hand_Landmarks

        # Recolor image back to BGR for rendering
        image.flags.writeable = True
        image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)

        # 1. Draw face Landmarks
        mp_drawing.draw_landmarks(image, results.face_landmarks, mp_holistic.FACEMESH_CONTOURS,
                                  mp_drawing.DrawingSpec(color=(80,110,10), thickness=1, circle_radius=1),
                                  mp_drawing.DrawingSpec(color=(80,256,121), thickness=1, circle_radius=1)
                                  )

        # 2. Right hand
        mp_drawing.draw_landmarks(image, results.right_hand_landmarks, mp_holistic.HAND_CONNECTIONS,
                                  mp_drawing.DrawingSpec(color=(80,22,10), thickness=2, circle_radius=4),
                                  mp_drawing.DrawingSpec(color=(80,44,121), thickness=2, circle_radius=2)
                                  )
```

```

# 3. Left Hand
mp_drawing.draw_landmarks(image, results.left_hand_landmarks, mp_holistic.HAND_CONNECTIONS,
                           mp_drawing.DrawingSpec(color=(121,22,76), thickness=2, circle_radius=4),
                           mp_drawing.DrawingSpec(color=(121,44,250), thickness=2, circle_radius=2)
                           )

# 4. Pose Detections
mp_drawing.draw_landmarks(image, results.pose_landmarks, mp_holistic.POSE_CONNECTIONS,
                           mp_drawing.DrawingSpec(color=(245,117,66), thickness=2, circle_radius=4),
                           mp_drawing.DrawingSpec(color=(245,66,230), thickness=2, circle_radius=2)
                           )

# Export coordinates
try:
    # Extract Pose Landmarks
    pose = results.pose_landmarks.landmark
    pose_row = list(np.array([[landmark.x, landmark.y, landmark.z, landmark.visibility] for landmark in pose]).flatten())

    # Extract Face Landmarks
    face = results.face_landmarks.landmark
    face_row = list(np.array([[landmark.x, landmark.y, landmark.z, landmark.visibility] for landmark in face]).flatten())

    # Concat rows
    row = pose_row+face_row

    # Append class name
    row.insert(0, class_name)

    # Export to CSV
    with open('coords.csv', mode='a', newline='') as f:
        csv_writer = csv.writer(f, delimiter=',', quotechar='"', quoting=csv.QUOTE_MINIMAL)
        csv_writer.writerow(row)

except:
    pass

cv2.imshow('Raw Webcam Feed', image)

if cv2.waitKey(10) & 0xFF == ord('q'):
    break

cap.release()
cv2.destroyAllWindows()

```

### 3.1 Read in Collected Data and Process

```
In [4]: import pandas as pd
        from sklearn.model_selection import train_test_split
```

```
In [33]: df = pd.read_csv('coords.csv')
```

```
In [34]: df.head()
```

```
Out[34]:
```

	class	x1	y1	z1	v1	x2	y2	z2	v2	x3	...	z499	v499	x500	y500	z500	v500
0	Happy	0.519015	0.709711	-0.873736	0.999543	0.562548	0.628961	-0.875275	0.998818	0.587117	...	-0.042088	0	0.616321	0.633365	-0.017780	0
1	Happy	0.518136	0.673005	-0.897379	0.999582	0.560578	0.597451	-0.888712	0.998919	0.584675	...	-0.024727	0	0.611566	0.577311	0.001688	0
2	Happy	0.514587	0.616093	-0.842586	0.999618	0.555625	0.543377	-0.821337	0.999009	0.578807	...	-0.014664	0	0.601421	0.528039	0.011426	0
3	Happy	0.514153	0.588563	-0.847342	0.999652	0.554745	0.518093	-0.826340	0.999098	0.577614	...	-0.012216	0	0.604919	0.514460	0.013796	0
4	Happy	0.514170	0.576236	-0.814643	0.999682	0.554615	0.505182	-0.784966	0.999177	0.577188	...	-0.008579	0	0.607744	0.506996	0.017951	0

5 rows x 2005 columns

```
In [35]: df.tail()
```

```
Out[35]:
```

	class	x1	y1	z1	v1	x2	y2	z2	v2	x3	...	z499	v499	x500	y500	z500	v500
1021	Angry	0.520437	0.465284	-0.367676	0.999500	0.547221	0.427252	-0.299362	0.999077	0.560491	...	0.005844	0	0.571175	0.448963	0.021185	0
1022	Angry	0.520338	0.465969	-0.418594	0.999495	0.546949	0.427647	-0.344232	0.999051	0.560239	...	0.004595	0	0.569815	0.450470	0.020036	0
1023	Angry	0.520085	0.466898	-0.473887	0.999462	0.545111	0.428245	-0.404624	0.998930	0.558175	...	0.004736	0	0.567147	0.446311	0.019050	0
1024	Angry	0.520053	0.467071	-0.441177	0.999456	0.544153	0.428593	-0.375526	0.998860	0.557023	...	0.005451	0	0.570375	0.446093	0.019240	0
1025	Angry	0.521677	0.459457	-0.342312	0.999502	0.544469	0.424694	-0.279833	0.998943	0.557050	...	0.006753	0	0.576301	0.441015	0.020348	0

5 rows x 2005 columns

```

In [41]: pipelines = {
        'lr': make_pipeline(StandardScaler(), LogisticRegression()),
        'rc': make_pipeline(StandardScaler(), RidgeClassifier()),
        'rf': make_pipeline(StandardScaler(), RandomForestClassifier()),
        'gb': make_pipeline(StandardScaler(), GradientBoostingClassifier()),
        }

In [42]: fit_models = {}
        for algo, pipeline in pipelines.items():
            model = pipeline.fit(X_train, y_train)
            fit_models[algo] = model

C:\Users\tnahs\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:818: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,

In [43]: fit_models
Out[43]: {'lr': Pipeline(steps=[('standardscaler', StandardScaler()),
                                ('logisticregression', LogisticRegression())]),
          'rc': Pipeline(steps=[('standardscaler', StandardScaler()),
                                ('ridgeclassifier', RidgeClassifier())]),
          'rf': Pipeline(steps=[('standardscaler', StandardScaler()),
                                ('randomforestclassifier', RandomForestClassifier())]),
          'gb': Pipeline(steps=[('standardscaler', StandardScaler()),
                                ('gradientboostingclassifier', GradientBoostingClassifier())])}

# Make Detections
X = pd.DataFrame([row])
body_language_class = model.predict(X)[0]
body_language_prob = model.predict_proba(X)[0]
print(body_language_class, body_language_prob)

# Grab ear coords
coords = tuple(np.multiply(
    np.array(
        (results.pose_landmarks.landmark[mp_holistic.PoseLandmark.LEFT_EAR].x,
         results.pose_landmarks.landmark[mp_holistic.PoseLandmark.LEFT_EAR].y))
    , [640,480]).astype(int))

cv2.rectangle(image,
              (coords[0], coords[1]+5),
              (coords[0]+len(body_language_class)*20, coords[1]-30),
              (245, 117, 16), -1)
cv2.putText(image, body_language_class, coords,
            cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2, cv2.LINE_AA)

# Get status box
cv2.rectangle(image, (0,0), (250, 60), (245, 117, 16), -1)

# Display Class
cv2.putText(image, 'CLASS'
            , (95,12), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 0), 1, cv2.LINE_AA)
cv2.putText(image, body_language_class.split(' ')[0]
            , (90,40), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2, cv2.LINE_AA)

# Display Probability
cv2.putText(image, 'PROB'
            , (15,12), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 0), 1, cv2.LINE_AA)
cv2.putText(image, str(round(body_language_prob[np.argmax(body_language_prob)],2))
            , (10,40), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2, cv2.LINE_AA)

# except:
# pass

cv2.imshow('Raw Webcam Feed', image)

if cv2.waitKey(10) & 0xFF == ord('q'):
    break

cap.release()
cv2.destroyAllWindows()

```