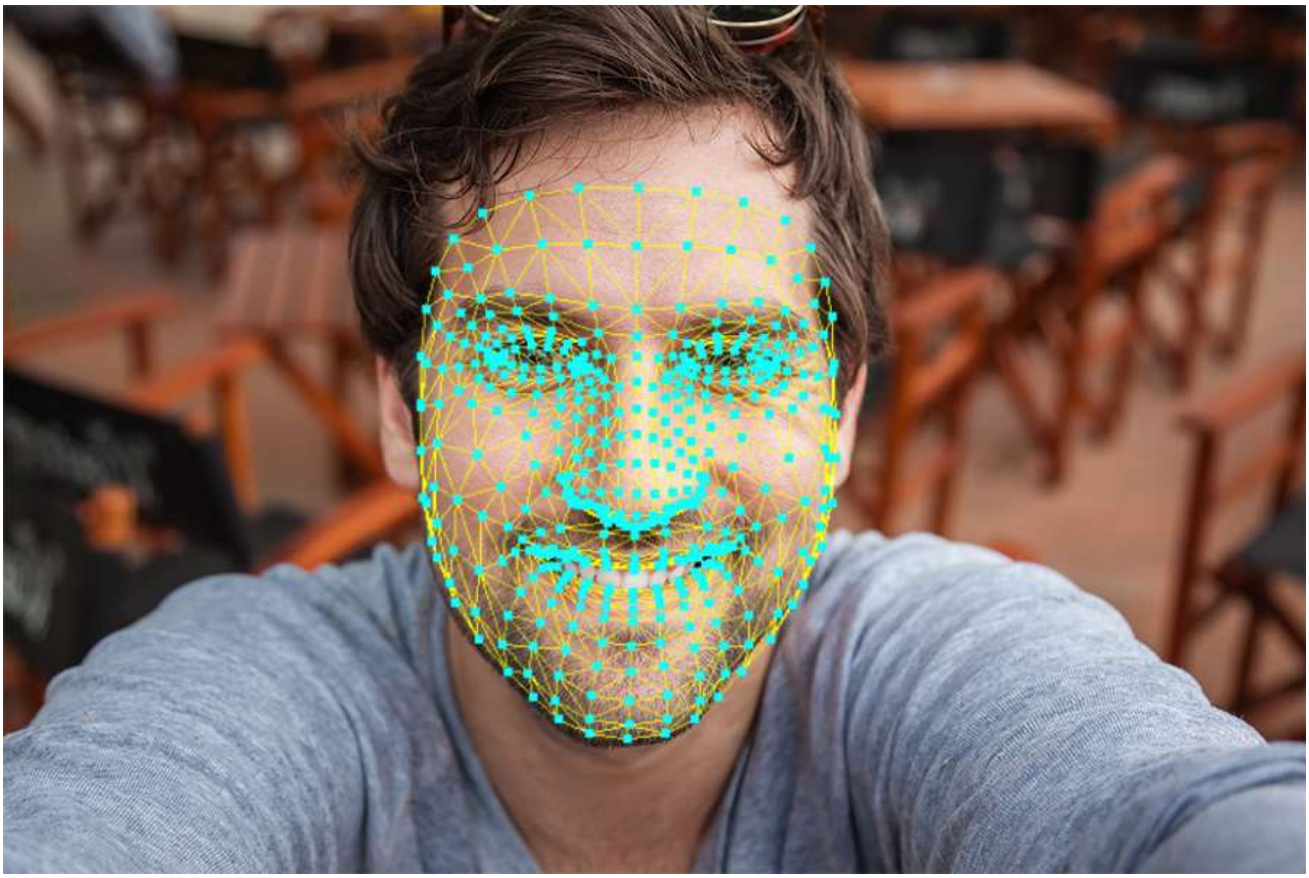
Dataintelliage

Body language decoder

Real time action recognition using Mediapipe for single person

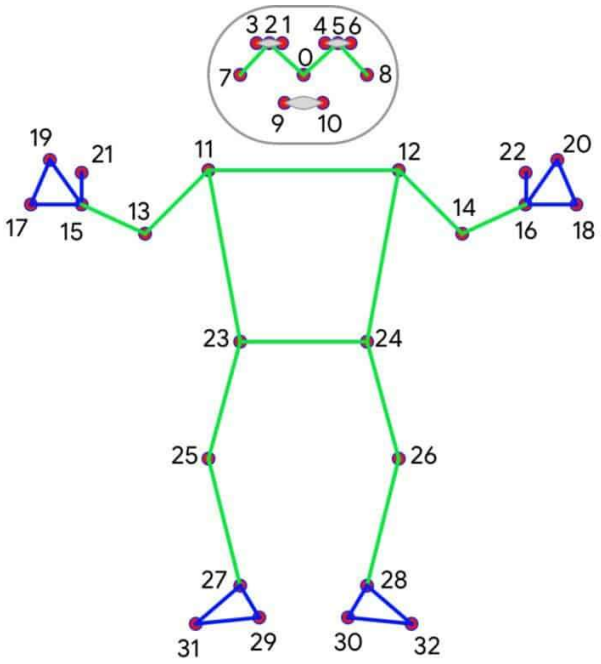
1. What is Mediapipe?

MediaPipe is an open-source framework for building pipelines to perform computer vision inference over arbitrary sensory data such as video or audio. Using MediaPipe, such a perception pipeline can be built as a graph of modular components. MediaPipe is currently in alpha at v0.7, and there may still be breaking API changes. Stable APIs are expected by v1.0.



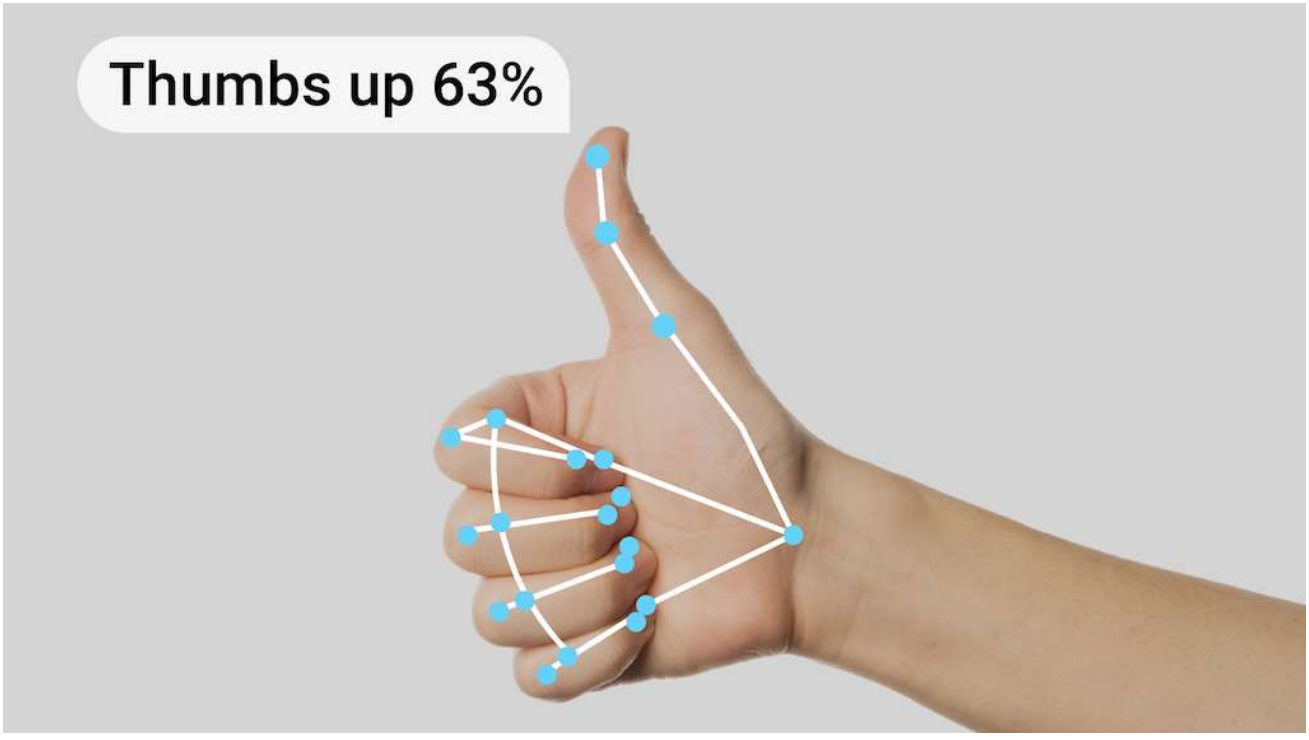
2.Pose landmarker

The pose landmarker model tracks 33 body landmark locations, representing the approximate location of the following body parts:



0. nose
1. right eye inner
2. right eye
3. right eye outer
4. left eye inner
5. left eye
6. left eye outer
7. right ear
8. left ear
9. mouth right
10. mouth left
11. right shoulder
12. left shoulder
13. right elbow
14. left elbow
15. right wrist
16. left wrist
17. right pinky knuckle #1
18. left pinky knuckle #1
19. right index knuclke #1
20. left index knuckle #1
21. right thumb knuckle #2
22. left thumb knuckle #2
23. right hip
24. left hip
25. right knee
26. left knee
27. right ankle
28. left ankle
29. right heel
30. left heel
31. right foot index
32. left foot index

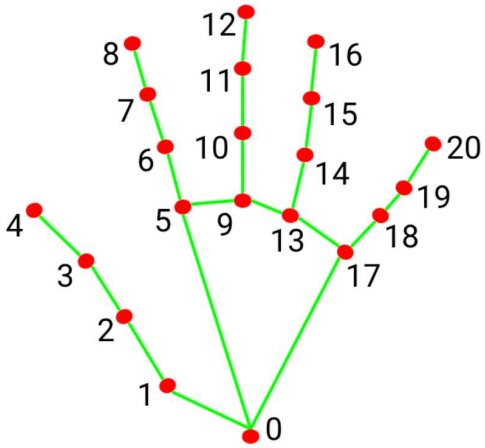
3.Application of mediapipe



-Object detection

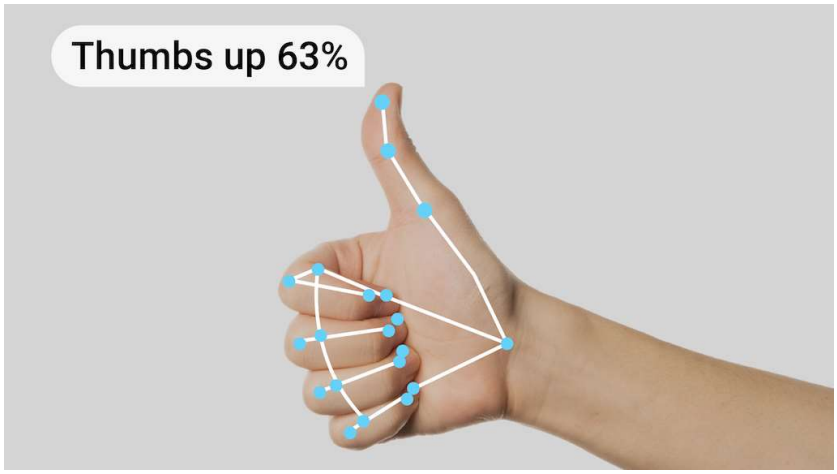
-Face detection

-Hand landmark dtection



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

-Gesture control



-Emotion recognition

-Single body action recognition

4.Procedure

-creating separate environment

-installing dependencies

-extracting landmarks (coordinates)

-saving all coordinates in csv file with different class (standing ,seating , happy , etc)

-training our models on the coordinates

-then predicting actions , emotions

Install and import depencencies

In [1]:
```
!pip install mediapipe opencv-python pandas scikit-learn
```

```
Requirement already satisfied: mediapipe in c:\users\asus\appdata\roaming\python\pyth
on311\site-packages (0.10.7)
Requirement already satisfied: opencv-python in c:\users\asus\anaconda3\lib\site-pack
ages (4.8.1.78)
Requirement already satisfied: pandas in c:\users\asus\anaconda3\lib\site-packages
(2.0.3)
Requirement already satisfied: scikit-learn in c:\users\asus\anaconda3\lib\site-packa
ges (1.3.0)
Requirement already satisfied: absl-py in c:\users\asus\anaconda3\lib\site-packages
(from mediapipe) (2.0.0)
Requirement already satisfied: attrs>=19.1.0 in c:\users\asus\anaconda3\lib\site-pack
ages (from mediapipe) (22.1.0)
Requirement already satisfied: flatbuffers>=2.0 in c:\users\asus\anaconda3\lib\site-p
ackages (from mediapipe) (23.5.26)
Requirement already satisfied: matplotlib in c:\users\asus\anaconda3\lib\site-package
s (from mediapipe) (3.7.2)
Requirement already satisfied: numpy in c:\users\asus\anaconda3\lib\site-packages (fr
om mediapipe) (1.24.3)
Requirement already satisfied: opencv-contrib-python in c:\users\asus\appdata\roaming
\python\python311\site-packages (from mediapipe) (4.8.1.78)
Requirement already satisfied: protobuf<4,>=3.11 in c:\users\asus\anaconda3\lib\site-
packages (from mediapipe) (3.20.3)
Requirement already satisfied: sounddevice>=0.4.4 in c:\users\asus\appdata\roaming\py
thon\python311\site-packages (from mediapipe) (0.4.6)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\asus\anaconda3\lib
\site-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\asus\anaconda3\lib\site-packa
ges (from pandas) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in c:\users\asus\anaconda3\lib\site-pac
kages (from pandas) (2023.3)
Requirement already satisfied: scipy>=1.5.0 in c:\users\asus\anaconda3\lib\site-packa
ges (from scikit-learn) (1.11.1)
Requirement already satisfied: joblib>=1.1.1 in c:\users\asus\anaconda3\lib\site-pack
ages (from scikit-learn) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\asus\anaconda3\lib\si
te-packages (from scikit-learn) (2.2.0)
Requirement already satisfied: six>=1.5 in c:\users\asus\anaconda3\lib\site-packages
(from python-dateutil>=2.8.2->pandas) (1.16.0)
Requirement already satisfied: CFFI>=1.0 in c:\users\asus\anaconda3\lib\site-packages
(from sounddevice>=0.4.4->mediapipe) (1.15.1)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\asus\anaconda3\lib\site-p
ackages (from matplotlib->mediapipe) (1.0.5)
Requirement already satisfied: cycler>=0.10 in c:\users\asus\anaconda3\lib\site-packa
ges (from matplotlib->mediapipe) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\asus\anaconda3\lib\site-
packages (from matplotlib->mediapipe) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\asus\anaconda3\lib\site-
packages (from matplotlib->mediapipe) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\users\asus\anaconda3\lib\site-pa
ckages (from matplotlib->mediapipe) (23.1)
Requirement already satisfied: pillow>=6.2.0 in c:\users\asus\anaconda3\lib\site-pack
ages (from matplotlib->mediapipe) (9.4.0)
Requirement already satisfied: pyparsing<3.1,>=2.3.1 in c:\users\asus\anaconda3\lib\s
ite-packages (from matplotlib->mediapipe) (3.0.9)
Requirement already satisfied: pycparser in c:\users\asus\anaconda3\lib\site-packages
(from CFFI>=1.0->sounddevice>=0.4.4->mediapipe) (2.21)
```

In [1]:
```
import mediapipe as mp # Import mediapipe
import cv2
```

In [2]:
```
mp_drawing = mp.solutions.drawing_utils # Drawing helpers
mp_holistic = mp.solutions.holistic
```

make detection

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

    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_
                                  mp_drawing.DrawingSpec(color=(80,110,10), thickness=1
                                  mp_drawing.DrawingSpec(color=(80,256,121), thickness=
                                  )

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

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

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

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

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

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

In [4]:
```
results.face_landmarks.landmark[0].visibility
```

Out[4]:
```
0.0
```

capture ladmarks

In [17]:
```
import csv
import os
import numpy as np
```

In [18]:
```
num_coords = len(results.pose_landmarks.landmark)+len(results.face_landmarks.landmark)
num_coords
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[18], line 1
----> 1 num_coords = len(results.pose_landmarks.landmark)+len(results.face_landmarks.
landmark)
      2 num_coords

NameError: name 'results' is not defined
```

In [17]:
```
landmarks = ['class']
for val in range(1, num_coords+1):
    landmarks += ['x{}'.format(val), 'y{}'.format(val), 'z{}'.format(val), 'v{}'.forma
```

In [18]:
```
landmarks
```

Out[18]:
```
['class',
 'x1',
 'y1',
 'z1',
 'v1',
 'x2',
 'y2',
 'z2',
 'v2',
 'x3',
 'y3',
 'z3',
 'v3',
 'x4',
 'y4',
 'z4',
 'v4',
 'x5',
 'y5',
 'z5',
 'v5',
 'x6',
 'y6',
 'z6',
 'v6',
 'x7',
 'y7',
 'z7',
 'v7',
 'x8',
 'y8',
 'z8',
 'v8',
 'x9',
 'y9',
 'z9',
 'v9',
 'x10',
 'y10',
 'z10',
 'v10',
 'x11',
 'y11',
 'z11',
 'v11',
 'x12',
 'y12',
 'z12',
 'v12',
 'x13',
 'y13',
 'z13',
 'v13',
 'x14',
 'y14',
 'z14',
 'v14',
 'x15',
 'y15',
 'z15',
```

```
'v15',
'x16',
'y16',
'z16',
'v16',
'x17',
'y17',
'z17',
'v17',
'x18',
'y18',
'z18',
'v18',
'x19',
'y19',
'z19',
'v19',
'x20',
'y20',
'z20',
'v20',
'x21',
'y21',
'z21',
'v21',
'x22',
'y22',
'z22',
'v22',
'x23',
'y23',
'z23',
'v23',
'x24',
'y24',
'z24',
'v24',
'x25',
'y25',
'z25',
'v25',
'x26',
'y26',
'z26',
'v26',
'x27',
'y27',
'z27',
'v27',
'x28',
'y28',
'z28',
'v28',
'x29',
'y29',
'z29',
'v29',
'x30',
'y30',
'z30',
```

```
'v30',
'x31',
'y31',
'z31',
'v31',
'x32',
'y32',
'z32',
'v32',
'x33',
'y33',
'z33',
'v33',
'x34',
'y34',
'z34',
'v34',
'x35',
'y35',
'z35',
'v35',
'x36',
'y36',
'z36',
'v36',
'x37',
'y37',
'z37',
'v37',
'x38',
'y38',
'z38',
'v38',
'x39',
'y39',
'z39',
'v39',
'x40',
'y40',
'z40',
'v40',
'x41',
'y41',
'z41',
'v41',
'x42',
'y42',
'z42',
'v42',
'x43',
'y43',
'z43',
'v43',
'x44',
'y44',
'z44',
'v44',
'x45',
'y45',
'z45',
```

```
'v45',
'x46',
'y46',
'z46',
'v46',
'x47',
'y47',
'z47',
'v47',
'x48',
'y48',
'z48',
'v48',
'x49',
'y49',
'z49',
'v49',
'x50',
'y50',
'z50',
'v50',
'x51',
'y51',
'z51',
'v51',
'x52',
'y52',
'z52',
'v52',
'x53',
'y53',
'z53',
'v53',
'x54',
'y54',
'z54',
'v54',
'x55',
'y55',
'z55',
'v55',
'x56',
'y56',
'z56',
'v56',
'x57',
'y57',
'z57',
'v57',
'x58',
'y58',
'z58',
'v58',
'x59',
'y59',
'z59',
'v59',
'x60',
'y60',
'z60',
```

```
'v60',
'x61',
'y61',
'z61',
'v61',
'x62',
'y62',
'z62',
'v62',
'x63',
'y63',
'z63',
'v63',
'x64',
'y64',
'z64',
'v64',
'x65',
'y65',
'z65',
'v65',
'x66',
'y66',
'z66',
'v66',
'x67',
'y67',
'z67',
'v67',
'x68',
'y68',
'z68',
'v68',
'x69',
'y69',
'z69',
'v69',
'x70',
'y70',
'z70',
'v70',
'x71',
'y71',
'z71',
'v71',
'x72',
'y72',
'z72',
'v72',
'x73',
'y73',
'z73',
'v73',
'x74',
'y74',
'z74',
'v74',
'x75',
'y75',
'z75',
```

```
'v75',
'x76',
'y76',
'z76',
'v76',
'x77',
'y77',
'z77',
'v77',
'x78',
'y78',
'z78',
'v78',
'x79',
'y79',
'z79',
'v79',
'x80',
'y80',
'z80',
'v80',
'x81',
'y81',
'z81',
'v81',
'x82',
'y82',
'z82',
'v82',
'x83',
'y83',
'z83',
'v83',
'x84',
'y84',
'z84',
'v84',
'x85',
'y85',
'z85',
'v85',
'x86',
'y86',
'z86',
'v86',
'x87',
'y87',
'z87',
'v87',
'x88',
'y88',
'z88',
'v88',
'x89',
'y89',
'z89',
'v89',
'x90',
'y90',
'z90',
```

```
'v90',
'x91',
'y91',
'z91',
'v91',
'x92',
'y92',
'z92',
'v92',
'x93',
'y93',
'z93',
'v93',
'x94',
'y94',
'z94',
'v94',
'x95',
'y95',
'z95',
'v95',
'x96',
'y96',
'z96',
'v96',
'x97',
'y97',
'z97',
'v97',
'x98',
'y98',
'z98',
'v98',
'x99',
'y99',
'z99',
'v99',
'x100',
'y100',
'z100',
'v100',
'x101',
'y101',
'z101',
'v101',
'x102',
'y102',
'z102',
'v102',
'x103',
'y103',
'z103',
'v103',
'x104',
'y104',
'z104',
'v104',
'x105',
'y105',
'z105',
```

```
'v105',
'x106',
'y106',
'z106',
'v106',
'x107',
'y107',
'z107',
'v107',
'x108',
'y108',
'z108',
'v108',
'x109',
'y109',
'z109',
'v109',
'x110',
'y110',
'z110',
'v110',
'x111',
'y111',
'z111',
'v111',
'x112',
'y112',
'z112',
'v112',
'x113',
'y113',
'z113',
'v113',
'x114',
'y114',
'z114',
'v114',
'x115',
'y115',
'z115',
'v115',
'x116',
'y116',
'z116',
'v116',
'x117',
'y117',
'z117',
'v117',
'x118',
'y118',
'z118',
'v118',
'x119',
'y119',
'z119',
'v119',
'x120',
'y120',
'z120',
```

```
'v120',
'x121',
'y121',
'z121',
'v121',
'x122',
'y122',
'z122',
'v122',
'x123',
'y123',
'z123',
'v123',
'x124',
'y124',
'z124',
'v124',
'x125',
'y125',
'z125',
'v125',
'x126',
'y126',
'z126',
'v126',
'x127',
'y127',
'z127',
'v127',
'x128',
'y128',
'z128',
'v128',
'x129',
'y129',
'z129',
'v129',
'x130',
'y130',
'z130',
'v130',
'x131',
'y131',
'z131',
'v131',
'x132',
'y132',
'z132',
'v132',
'x133',
'y133',
'z133',
'v133',
'x134',
'y134',
'z134',
'v134',
'x135',
'y135',
'z135',
```

```
'v135',
'x136',
'y136',
'z136',
'v136',
'x137',
'y137',
'z137',
'v137',
'x138',
'y138',
'z138',
'v138',
'x139',
'y139',
'z139',
'v139',
'x140',
'y140',
'z140',
'v140',
'x141',
'y141',
'z141',
'v141',
'x142',
'y142',
'z142',
'v142',
'x143',
'y143',
'z143',
'v143',
'x144',
'y144',
'z144',
'v144',
'x145',
'y145',
'z145',
'v145',
'x146',
'y146',
'z146',
'v146',
'x147',
'y147',
'z147',
'v147',
'x148',
'y148',
'z148',
'v148',
'x149',
'y149',
'z149',
'v149',
'x150',
'y150',
'z150',
```

```
'v150',
'x151',
'y151',
'z151',
'v151',
'x152',
'y152',
'z152',
'v152',
'x153',
'y153',
'z153',
'v153',
'x154',
'y154',
'z154',
'v154',
'x155',
'y155',
'z155',
'v155',
'x156',
'y156',
'z156',
'v156',
'x157',
'y157',
'z157',
'v157',
'x158',
'y158',
'z158',
'v158',
'x159',
'y159',
'z159',
'v159',
'x160',
'y160',
'z160',
'v160',
'x161',
'y161',
'z161',
'v161',
'x162',
'y162',
'z162',
'v162',
'x163',
'y163',
'z163',
'v163',
'x164',
'y164',
'z164',
'v164',
'x165',
'y165',
'z165',
```

```
'v165',
'x166',
'y166',
'z166',
'v166',
'x167',
'y167',
'z167',
'v167',
'x168',
'y168',
'z168',
'v168',
'x169',
'y169',
'z169',
'v169',
'x170',
'y170',
'z170',
'v170',
'x171',
'y171',
'z171',
'v171',
'x172',
'y172',
'z172',
'v172',
'x173',
'y173',
'z173',
'v173',
'x174',
'y174',
'z174',
'v174',
'x175',
'y175',
'z175',
'v175',
'x176',
'y176',
'z176',
'v176',
'x177',
'y177',
'z177',
'v177',
'x178',
'y178',
'z178',
'v178',
'x179',
'y179',
'z179',
'v179',
'x180',
'y180',
'z180',
```

```
'v180',
'x181',
'y181',
'z181',
'v181',
'x182',
'y182',
'z182',
'v182',
'x183',
'y183',
'z183',
'v183',
'x184',
'y184',
'z184',
'v184',
'x185',
'y185',
'z185',
'v185',
'x186',
'y186',
'z186',
'v186',
'x187',
'y187',
'z187',
'v187',
'x188',
'y188',
'z188',
'v188',
'x189',
'y189',
'z189',
'v189',
'x190',
'y190',
'z190',
'v190',
'x191',
'y191',
'z191',
'v191',
'x192',
'y192',
'z192',
'v192',
'x193',
'y193',
'z193',
'v193',
'x194',
'y194',
'z194',
'v194',
'x195',
'y195',
'z195',
```

```
'v195',
'x196',
'y196',
'z196',
'v196',
'x197',
'y197',
'z197',
'v197',
'x198',
'y198',
'z198',
'v198',
'x199',
'y199',
'z199',
'v199',
'x200',
'y200',
'z200',
'v200',
'x201',
'y201',
'z201',
'v201',
'x202',
'y202',
'z202',
'v202',
'x203',
'y203',
'z203',
'v203',
'x204',
'y204',
'z204',
'v204',
'x205',
'y205',
'z205',
'v205',
'x206',
'y206',
'z206',
'v206',
'x207',
'y207',
'z207',
'v207',
'x208',
'y208',
'z208',
'v208',
'x209',
'y209',
'z209',
'v209',
'x210',
'y210',
'z210',
```

```
'v210',
'x211',
'y211',
'z211',
'v211',
'x212',
'y212',
'z212',
'v212',
'x213',
'y213',
'z213',
'v213',
'x214',
'y214',
'z214',
'v214',
'x215',
'y215',
'z215',
'v215',
'x216',
'y216',
'z216',
'v216',
'x217',
'y217',
'z217',
'v217',
'x218',
'y218',
'z218',
'v218',
'x219',
'y219',
'z219',
'v219',
'x220',
'y220',
'z220',
'v220',
'x221',
'y221',
'z221',
'v221',
'x222',
'y222',
'z222',
'v222',
'x223',
'y223',
'z223',
'v223',
'x224',
'y224',
'z224',
'v224',
'x225',
'y225',
'z225',
```

```
'v225',
'x226',
'y226',
'z226',
'v226',
'x227',
'y227',
'z227',
'v227',
'x228',
'y228',
'z228',
'v228',
'x229',
'y229',
'z229',
'v229',
'x230',
'y230',
'z230',
'v230',
'x231',
'y231',
'z231',
'v231',
'x232',
'y232',
'z232',
'v232',
'x233',
'y233',
'z233',
'v233',
'x234',
'y234',
'z234',
'v234',
'x235',
'y235',
'z235',
'v235',
'x236',
'y236',
'z236',
'v236',
'x237',
'y237',
'z237',
'v237',
'x238',
'y238',
'z238',
'v238',
'x239',
'y239',
'z239',
'v239',
'x240',
'y240',
'z240',
```

```
'v240',
'x241',
'y241',
'z241',
'v241',
'x242',
'y242',
'z242',
'v242',
'x243',
'y243',
'z243',
'v243',
'x244',
'y244',
'z244',
'v244',
'x245',
'y245',
'z245',
'v245',
'x246',
'y246',
'z246',
'v246',
'x247',
'y247',
'z247',
'v247',
'x248',
'y248',
'z248',
'v248',
'x249',
'y249',
'z249',
'v249',
'x250',
'y250',
'z250',
...]
```

In [28]:
```
with open('coord.csv', mode='w', newline='') as f:
    csv_writer = csv.writer(f, delimiter=',', quotechar='"', quoting=csv.QUOTE_MINIMAL
    csv_writer.writerow(landmarks)
```

In [59]:
```
class_name = "standing"
```

In [60]:
```
cap = cv2.VideoCapture(0)
# Initiate holistic model
with mp_holistic.Holistic(min_detection_confidence=0.5, min_tracking_confidence=0.5) a

    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_
                                  mp_drawing.DrawingSpec(color=(80,110,10), thickness=1
                                  mp_drawing.DrawingSpec(color=(80,256,121), thickness=
                                  )

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

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

        # 4. Pose Detections
        mp_drawing.draw_landmarks(image, results.pose_landmarks, mp_holistic.POSE_CONN
                                  mp_drawing.DrawingSpec(color=(245,117,66), thickness=
                                  mp_drawing.DrawingSpec(color=(245,66,230), thickness=
                                  )
        # Export coordinates
        try:
            # Extract Pose landmarks
            pose = results.pose_landmarks.landmark
            pose_row = list(np.array([[landmark.x, landmark.y, landmark.z, landmark.vi

            # Extract Face landmarks
            face = results.face_landmarks.landmark
            face_row = list(np.array([[landmark.x, landmark.y, landmark.z, landmark.vi

            # Concate rows
            row = pose_row+face_row

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

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

        except:
            pass

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

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

```
            break

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

In [1]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
```

In [2]:
```
df = pd.read_csv('coord.csv')
```

In [3]:
```
df.head()
```

Out[3]:

	class	x1	y1	z1	v1	x2	y2	z2	v2	x3
0	working	0.452353	0.278074	-0.483196	0.999997	0.470504	0.234207	-0.449846	0.999994	0.481901
1	working	0.454130	0.276581	-0.543517	0.999996	0.470552	0.233398	-0.492786	0.999993	0.481845
2	working	0.454202	0.275508	-0.542326	0.999995	0.470423	0.232785	-0.493952	0.999993	0.481429
3	working	0.458049	0.270778	-0.489343	0.999994	0.471127	0.229890	-0.432510	0.999992	0.481477
4	working	0.457954	0.270314	-0.481695	0.999994	0.470534	0.229904	-0.423149	0.999992	0.479913

5 rows × 2005 columns

In [4]:
```
df.tail()
```

Out[4]:

	class	x1	y1	z1	v1	x2	y2	z2	v2	
627	standing	0.442585	0.288592	-0.405592	0.999865	0.449690	0.273587	-0.382563	0.999829	0.4530
628	standing	0.504412	0.291029	-0.243900	0.999705	0.500397	0.272937	-0.213305	0.999541	0.5010
629	standing	0.382215	0.317442	-0.340185	0.999013	0.389706	0.299102	-0.337546	0.998961	0.3935
630	standing	0.384418	0.312225	-0.317954	0.999196	0.392302	0.292905	-0.303286	0.999143	0.3966
631	standing	0.379089	0.292984	-0.433839	0.999272	0.388350	0.273902	-0.414130	0.999213	0.3938

5 rows × 2005 columns

In [5]:
```
df[df['class']=='working']
```

Out[5]:

	class	x1	y1	z1	v1	x2	y2	z2	v2	x
0	working	0.452353	0.278074	-0.483196	0.999997	0.470504	0.234207	-0.449846	0.999994	0.4819
1	working	0.454130	0.276581	-0.543517	0.999996	0.470552	0.233398	-0.492786	0.999993	0.4818
2	working	0.454202	0.275508	-0.542326	0.999995	0.470423	0.232785	-0.493952	0.999993	0.4814
3	working	0.458049	0.270778	-0.489343	0.999994	0.471127	0.229890	-0.432510	0.999992	0.4814
4	working	0.457954	0.270314	-0.481695	0.999994	0.470534	0.229904	-0.423149	0.999992	0.4799
...	...	...	...	...	...	...	...	...	...	...
115	working	0.395265	0.248851	-0.508753	0.999328	0.413086	0.197402	-0.484638	0.998845	0.4273
116	working	0.404025	0.235304	-0.520985	0.999140	0.422906	0.182646	-0.500296	0.998510	0.4367
117	working	0.426504	0.189636	-0.527140	0.999161	0.447941	0.138167	-0.505648	0.998507	0.4624
118	working	0.484437	0.113024	-0.565920	0.998896	0.511246	0.062458	-0.538478	0.997820	0.5268
119	working	0.513042	0.110527	-0.575804	0.998516	0.539975	0.058484	-0.557212	0.996654	0.5543

120 rows × 2005 columns

In [6]:
```
X = df.drop('class', axis=1) # features
y = df['class'] # target value
```

In [7]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=
```

In [8]:
```
y_test
```

Out[8]:
```
558           standing
234    talking on phone
430           sleeping
326            seating
389            walking
             ...
555           standing
454           sleeping
511           sleeping
285            seating
390            walking
Name: class, Length: 190, dtype: object
```

In [ ]:
```
```

In [9]:
```
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler

from sklearn.linear_model import  RidgeClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
```

In [10]:
```
pipelines = {
    #'lr':make_pipeline(StandardScaler(), LogisticRegression()),
    'rc':make_pipeline(StandardScaler(), RidgeClassifier()),
    'rf':make_pipeline(StandardScaler(), RandomForestClassifier()),
```

```
    'gb':make_pipeline(StandardScaler(), GradientBoostingClassifier()),
}
```

In [14]:
```
fit_models = {}
for algo, pipeline in pipelines.items():
    model = pipeline.fit(X_train, y_train)
    fit_models[algo] = model
```

In [15]:
```
fit_models
```

Out[15]:
```
{'rc': Pipeline(steps=[('standardscaler', StandardScaler()),
                       ('ridgeclassifier', RidgeClassifier())]),
 'rf': Pipeline(steps=[('standardscaler', StandardScaler()),
                       ('randomforestclassifier', RandomForestClassifier())]),
 'gb': Pipeline(steps=[('standardscaler', StandardScaler()),
                       ('gradientboostingclassifier', GradientBoostingClassifier())])}
```

In [31]:
```
fit_models['rc'].predict(X_test)
```

Out[31]:
```
array(['standing', 'talking on phone', 'sleeping', 'seating', 'walking',
       'talking on phone', 'sleeping', 'standing', 'talking on phone',
       'talking on phone', 'seating', 'walking', 'working', 'sleeping',
       'talking on phone', 'working', 'standing', 'standing', 'standing',
       'talking on phone', 'talking on phone', 'standing', 'walking',
       'sleeping', 'sleeping', 'standing', 'seating', 'standing',
       'seating', 'sleeping', 'standing', 'sleeping', 'standing',
       'standing', 'talking on phone', 'seating', 'sleeping', 'seating',
       'sleeping', 'talking on phone', 'standing', 'standing', 'sleeping',
       'seating', 'working', 'sleeping', 'seating', 'seating', 'working',
       'standing', 'seating', 'walking', 'seating', 'talking on phone',
       'sleeping', 'working', 'talking on phone', 'seating',
       'talking on phone', 'talking on phone', 'talking on phone',
       'talking on phone', 'sleeping', 'talking on phone', 'working',
       'walking', 'standing', 'standing', 'standing', 'talking on phone',
       'walking', 'talking on phone', 'talking on phone',
       'talking on phone', 'seating', 'walking', 'talking on phone',
       'seating', 'talking on phone', 'sleeping', 'standing', 'standing',
       'working', 'seating', 'sleeping', 'working', 'seating',
       'talking on phone', 'talking on phone', 'talking on phone',
       'working', 'walking', 'walking', 'talking on phone', 'walking',
       'standing', 'working', 'walking', 'walking', 'sleeping', 'working',
       'sleeping', 'working', 'seating', 'talking on phone',
       'talking on phone', 'standing', 'talking on phone', 'seating',
       'seating', 'sleeping', 'walking', 'sleeping', 'working', 'working',
       'working', 'talking on phone', 'talking on phone', 'sleeping',
       'talking on phone', 'sleeping', 'sleeping', 'talking on phone',
       'sleeping', 'seating', 'standing', 'walking', 'sleeping',
       'talking on phone', 'seating', 'standing', 'seating',
       'talking on phone', 'working', 'working', 'working', 'sleeping',
       'seating', 'seating', 'standing', 'working', 'working', 'working',
       'talking on phone', 'seating', 'seating', 'working', 'sleeping',
       'talking on phone', 'walking', 'talking on phone', 'working',
       'working', 'sleeping', 'sleeping', 'working', 'seating',
       'talking on phone', 'sleeping', 'seating', 'sleeping', 'walking',
       'standing', 'seating', 'working', 'sleeping', 'seating',
       'talking on phone', 'working', 'seating', 'sleeping', 'sleeping',
       'talking on phone', 'working', 'walking', 'working', 'walking',
       'sleeping', 'standing', 'seating', 'talking on phone', 'sleeping',
       'sleeping', 'standing', 'working', 'standing', 'sleeping',
       'sleeping', 'seating', 'walking'], dtype='<U16')
```

In [17]:
```
from sklearn.metrics import accuracy_score # Accuracy metrics
import pickle
```

In [18]:
```
for algo, model in fit_models.items():
    yhat = model.predict(X_test)
    print(algo, accuracy_score(y_test, yhat))
```

```
rc 1.0
rf 1.0
gb 0.9947368421052631
```

In [32]:
```
fit_models['rc'].predict(X_test)
```

Out[32]:
```
array(['standing', 'talking on phone', 'sleeping', 'seating', 'walking',
       'talking on phone', 'sleeping', 'standing', 'talking on phone',
       'talking on phone', 'seating', 'walking', 'working', 'sleeping',
       'talking on phone', 'working', 'standing', 'standing', 'standing',
       'talking on phone', 'talking on phone', 'standing', 'walking',
       'sleeping', 'sleeping', 'standing', 'seating', 'standing',
       'seating', 'sleeping', 'standing', 'sleeping', 'standing',
       'standing', 'talking on phone', 'seating', 'sleeping', 'seating',
       'sleeping', 'talking on phone', 'standing', 'standing', 'sleeping',
       'seating', 'working', 'sleeping', 'seating', 'seating', 'working',
       'standing', 'seating', 'walking', 'seating', 'talking on phone',
       'sleeping', 'working', 'talking on phone', 'seating',
       'talking on phone', 'talking on phone', 'talking on phone',
       'talking on phone', 'sleeping', 'talking on phone', 'working',
       'walking', 'standing', 'standing', 'standing', 'talking on phone',
       'walking', 'talking on phone', 'talking on phone',
       'talking on phone', 'seating', 'walking', 'talking on phone',
       'seating', 'talking on phone', 'sleeping', 'standing', 'standing',
       'working', 'seating', 'sleeping', 'working', 'seating',
       'talking on phone', 'talking on phone', 'talking on phone',
       'working', 'walking', 'walking', 'talking on phone', 'walking',
       'standing', 'working', 'walking', 'walking', 'sleeping', 'working',
       'sleeping', 'working', 'seating', 'talking on phone',
       'talking on phone', 'standing', 'talking on phone', 'seating',
       'seating', 'sleeping', 'walking', 'sleeping', 'working', 'working',
       'working', 'talking on phone', 'talking on phone', 'sleeping',
       'talking on phone', 'sleeping', 'sleeping', 'talking on phone',
       'sleeping', 'seating', 'standing', 'walking', 'sleeping',
       'talking on phone', 'seating', 'standing', 'seating',
       'talking on phone', 'working', 'working', 'working', 'sleeping',
       'seating', 'seating', 'standing', 'working', 'working', 'working',
       'talking on phone', 'seating', 'seating', 'working', 'sleeping',
       'talking on phone', 'walking', 'talking on phone', 'working',
       'working', 'sleeping', 'sleeping', 'working', 'seating',
       'talking on phone', 'sleeping', 'seating', 'sleeping', 'walking',
       'standing', 'seating', 'working', 'sleeping', 'seating',
       'talking on phone', 'working', 'seating', 'sleeping', 'sleeping',
       'talking on phone', 'working', 'walking', 'working', 'walking',
       'sleeping', 'standing', 'seating', 'talking on phone', 'sleeping',
       'sleeping', 'standing', 'working', 'standing', 'sleeping',
       'sleeping', 'seating', 'walking'], dtype='<U16')
```

In [33]:
```
y_test
```

Out[33]:
```
558            standing
234    talking on phone
430            sleeping
326             seating
389             walking
             ...
555            standing
454            sleeping
511            sleeping
285             seating
390             walking
Name: class, Length: 190, dtype: object
```

In [34]:
```
with open('body_language_pose_rf.pkl', 'wb') as f:
    pickle.dump(fit_models['rc'], f)
```

4. Make Detections with Model

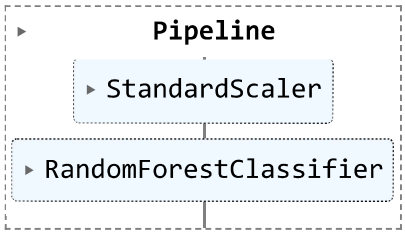
In [25]:
```
with open('body_language_pose.pkl', 'rb') as f:
    model = pickle.load(f)
```

In [26]:
```
with open('body_language_pose.pkl1', 'rb') as f:
    model1 = pickle.load(f)
```

In [35]:
```
with open('body_language_pose_rf.pkl', 'rb') as f:
    model2 = pickle.load(f)
```

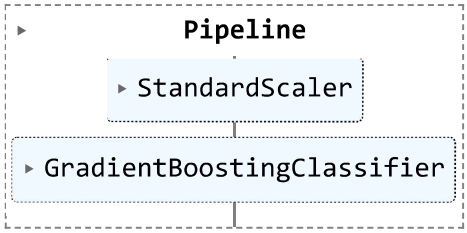
In [28]:
```
model
```

Out[28]:



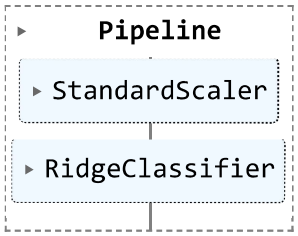
In [29]:
```
model1
```

Out[29]:



In [36]:
```
model2
```

Out[36]:



In [ ]:
```
```

In [1]:
```
cap = cv2.VideoCapture(0)
# Initiate holistic model
with mp_holistic.Holistic(min_detection_confidence=0.5, min_tracking_confidence=0.5) a

    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_
                                  mp_drawing.DrawingSpec(color=(80,110,10), thickness=1
                                  mp_drawing.DrawingSpec(color=(80,256,121), thickness=
                                  )

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

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

        # 4. Pose Detections
        mp_drawing.draw_landmarks(image, results.pose_landmarks, mp_holistic.POSE_CONN
                                  mp_drawing.DrawingSpec(color=(245,117,66), thickness=
                                  mp_drawing.DrawingSpec(color=(245,66,230), thickness=
                                  )
        # Export coordinates
        try:
            # Extract Pose landmarks
            pose = results.pose_landmarks.landmark
            pose_row = list(np.array([[landmark.x, landmark.y, landmark.z, landmark.vi

            # Extract Face landmarks
            face = results.face_landmarks.landmark
            face_row = list(np.array([[landmark.x, landmark.y, landmark.z, landmark.vi

            # Concate rows
            row = pose_row+face_row

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

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

            # 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.PoseLandm
                                 results.pose_landmarks.landmark[mp_holistic.PoseLandm
                        , [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.LI
            cv2.putText(image, body_language_class.split(' ')[0]
                        , (90,40), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2, cv

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



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

        except:
            pass

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

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

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

```
  Cell In[1], line 59
    row.insert(0, class_name)
    ^
IndentationError: unexpected indent
```

In [26]:
```
tuple(np.multiply(np.array((results.pose_landmarks.landmark[mp_holistic.PoseLandmark.L
results.pose_landmarks.landmark[mp_holistic.PoseLandmark.LEFT_EAR].y)), [640,480]).ast
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[26], line 1
----> 1 tuple(np.multiply(np.array((results.pose_landmarks.landmark[mp_holistic.PoseL
andmark.LEFT_EAR].x,
      2 results.pose_landmarks.landmark[mp_holistic.PoseLandmark.LEFT_EAR].y)), [640,
480]).astype(int))

NameError: name 'np' is not defined
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [1]:
```
import csv
import os
import numpy as np
import pandas as pd
```

In [2]:
```
import pickle
```

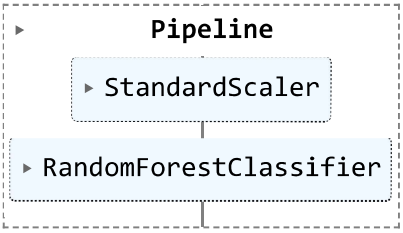
In [3]:
```
import mediapipe as mp # Import mediapipe
import cv2
```

In [4]:
```
mp_drawing = mp.solutions.drawing_utils # Drawing helpers
mp_holistic = mp.solutions.holistic
```

In [5]:
```
with open('body_language_pose.pkl', 'rb') as f:
    model = pickle.load(f)
```

In [6]:
```
model
```

Out[6]:



In [10]:
```
cap = cv2.VideoCapture(0)
# Initiate holistic model
with mp_holistic.Holistic(min_detection_confidence=0.5, min_tracking_confidence=0.5) a

    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_
                                  mp_drawing.DrawingSpec(color=(80,110,10), thickness=1
                                  mp_drawing.DrawingSpec(color=(80,256,121), thickness=
                                  )

        # 2. Right hand
        mp_drawing.draw_landmarks(image, results.right_hand_landmarks, mp_holistic.HAN
                                  mp_drawing.DrawingSpec(color=(80,22,10), thickness=2,
```

```
                                mp_drawing.DrawingSpec(color=(80,44,121), thickness=2
                                )

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

        # 4. Pose Detections
        mp_drawing.draw_landmarks(image, results.pose_landmarks, mp_holistic.POSE_CONN
                                mp_drawing.DrawingSpec(color=(245,117,66), thickness=
                                mp_drawing.DrawingSpec(color=(245,66,230), thickness=
                                )
        # Export coordinates
        try:
            # Extract Pose landmarks
            pose = results.pose_landmarks.landmark
            pose_row = list(np.array([[landmark.x, landmark.y, landmark.z, landmark.vi

            # Extract Face landmarks
            face = results.face_landmarks.landmark
            face_row = list(np.array([[landmark.x, landmark.y, landmark.z, landmark.vi

            # Concate rows
            row = pose_row+face_row

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

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

            # 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
            coord = tuple(np.multiply(
                            np.array(
                                (results.pose_landmarks.landmark[mp_holistic.PoseLandm
                                 results.pose_landmarks.landmark[mp_holistic.PoseLandm
                        , [640,480]).astype(int))

            cv2.rectangle(image,
                          (coord[0], coord[1]+5),
                          (coord[0]+len(body_language_class)*20, coord[1]-30),
                          (245, 117, 16), -1)
            cv2.putText(image, body_language_class, coord,
                        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.LI
        cv2.putText(image, body_language_class.split(' ')[0]
                    , (90,40), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2, cv

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

    except:
        pass

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

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

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

```
C:\Users\Asus\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning: X does no
t have valid feature names, but StandardScaler was fitted with feature names
  warnings.warn(
C:\Users\Asus\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning: X does no
t have valid feature names, but StandardScaler was fitted with feature names
  warnings.warn(
C:\Users\Asus\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning: X does no
t have valid feature names, but StandardScaler was fitted with feature names
  warnings.warn(
C:\Users\Asus\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning: X does no
t have valid feature names, but StandardScaler was fitted with feature names
  warnings.warn(
talking on phone [0.   0.43 0.   0.5  0.   0.07]
sleeping [0.   0.45 0.   0.4  0.   0.15]
C:\Users\Asus\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning: X does no
t have valid feature names, but StandardScaler was fitted with feature names
  warnings.warn(
C:\Users\Asus\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning: X does no
t have valid feature names, but StandardScaler was fitted with feature names
  warnings.warn(
C:\Users\Asus\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning: X does no
t have valid feature names, but StandardScaler was fitted with feature names
  warnings.warn(
C:\Users\Asus\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning: X does no
t have valid feature names, but StandardScaler was fitted with feature names
  warnings.warn(
sleeping [0.   0.67 0.   0.07 0.   0.26]
sleeping [0.   0.74 0.   0.04 0.   0.22]
```

```
C:\Users\Asus\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning: X does no
t have valid feature names, but StandardScaler was fitted with feature names
  warnings.warn(
C:\Users\Asus\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning: X does no
t have valid feature names, but StandardScaler was fitted with feature names
  warnings.warn(
C:\Users\Asus\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning: X does no
t have valid feature names, but StandardScaler was fitted with feature names
  warnings.warn(
C:\Users\Asus\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning: X does no
t have valid feature names, but StandardScaler was fitted with feature names
  warnings.warn(
sleeping [0.   0.74 0.   0.04 0.   0.22]
sleeping [0.   0.73 0.   0.04 0.   0.23]
C:\Users\Asus\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning: X does no
t have valid feature names, but StandardScaler was fitted with feature names
  warnings.warn(
C:\Users\Asus\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning: X does no
t have valid feature names, but StandardScaler was fitted with feature names
  warnings.warn(
C:\Users\Asus\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning: X does no
t have valid feature names, but StandardScaler was fitted with feature names
  warnings.warn(
C:\Users\Asus\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning: X does no
t have valid feature names, but StandardScaler was fitted with feature names
  warnings.warn(
sleeping [0.   0.68 0.   0.04 0.   0.28]
sleeping [0.   0.69 0.   0.04 0.   0.27]
C:\Users\Asus\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning: X does no
t have valid feature names, but StandardScaler was fitted with feature names
  warnings.warn(
C:\Users\Asus\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning: X does no
t have valid feature names, but StandardScaler was fitted with feature names
  warnings.warn(
C:\Users\Asus\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning: X does no
t have valid feature names, but StandardScaler was fitted with feature names
  warnings.warn(
C:\Users\Asus\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning: X does no
t have valid feature names, but StandardScaler was fitted with feature names
  warnings.warn(
sleeping [0.   0.71 0.   0.04 0.   0.25]
sleeping [0.   0.69 0.   0.04 0.   0.27]
C:\Users\Asus\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning: X does no
t have valid feature names, but StandardScaler was fitted with feature names
  warnings.warn(
C:\Users\Asus\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning: X does no
t have valid feature names, but StandardScaler was fitted with feature names
  warnings.warn(
C:\Users\Asus\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning: X does no
t have valid feature names, but StandardScaler was fitted with feature names
  warnings.warn(
C:\Users\Asus\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning: X does no
t have valid feature names, but StandardScaler was fitted with feature names
  warnings.warn(
sleeping [0.   0.69 0.   0.04 0.   0.27]
sleeping [0.   0.68 0.   0.03 0.   0.29]
sleeping [0.   0.69 0.   0.03 0.   0.28]
```