Steering Wheel State Warning

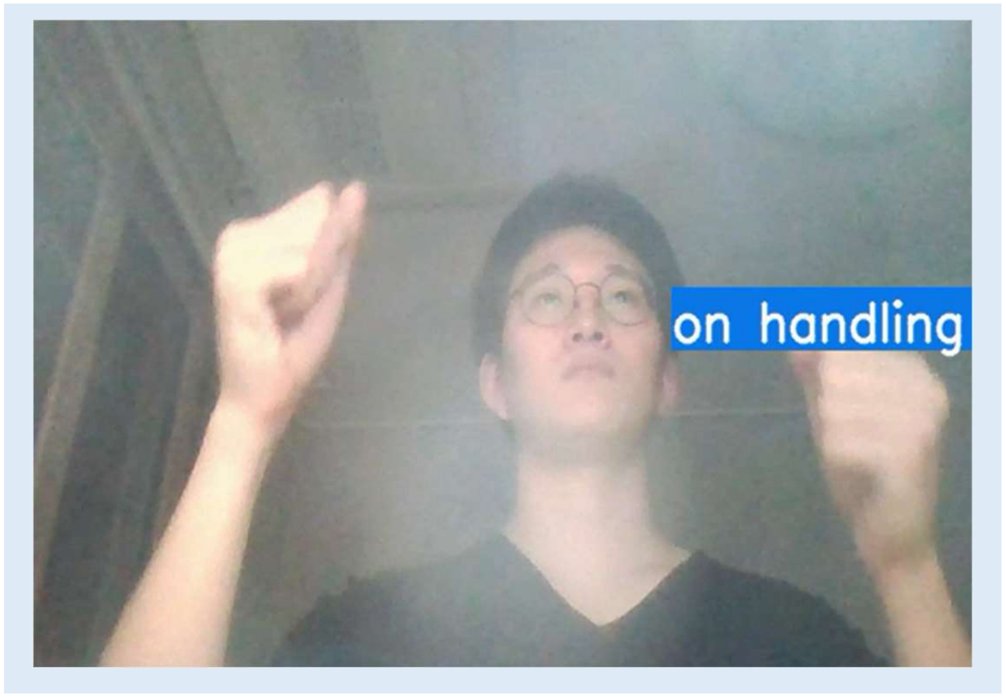
국민대학교 자동차공학과
20163323 이인재

Problem

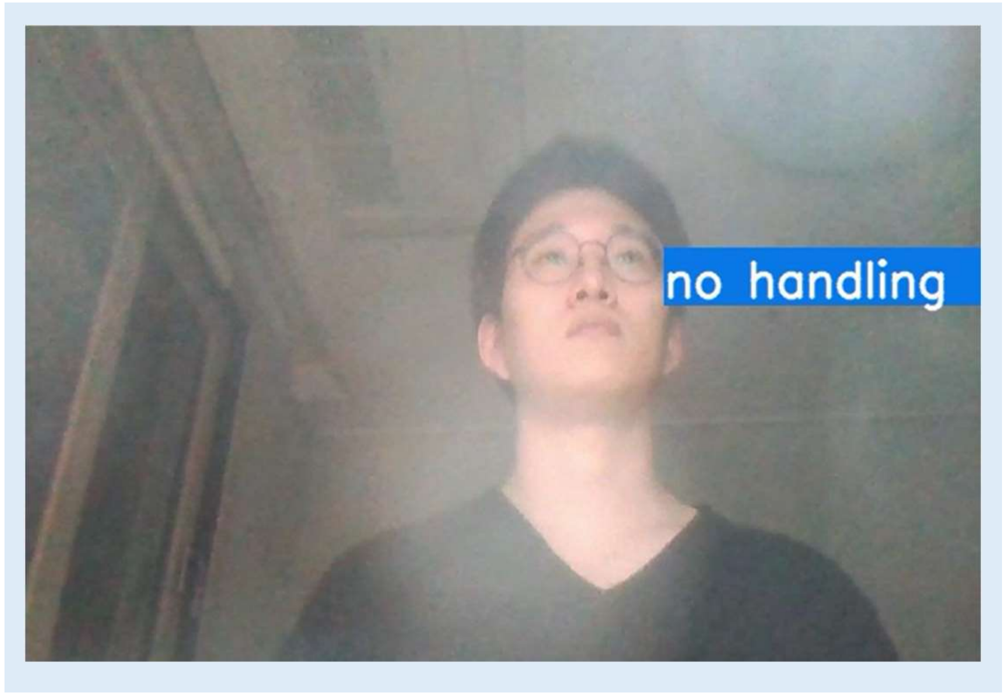
Place weighty objects on the steering wheel, which self driving car misunderstand users is grabbing the steering wheel. This can cause terrible accident when driver faces corner case.

Proposal

What about using cameras to detect whether driver is grabbing the steering wheel or not?



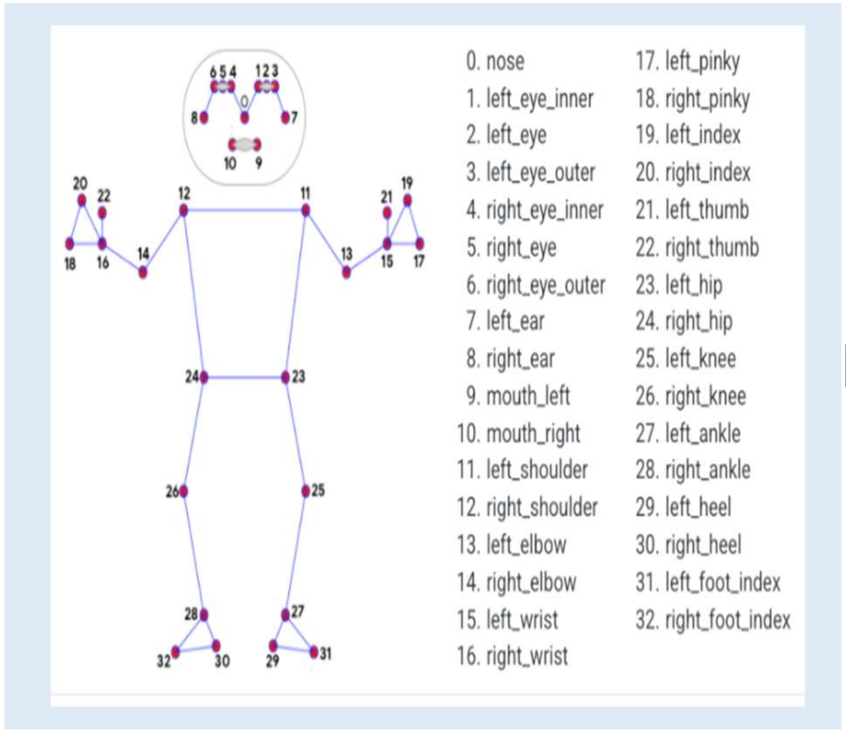
Driver grabbing the steering wheel



Driver not grabbing the steering wheel

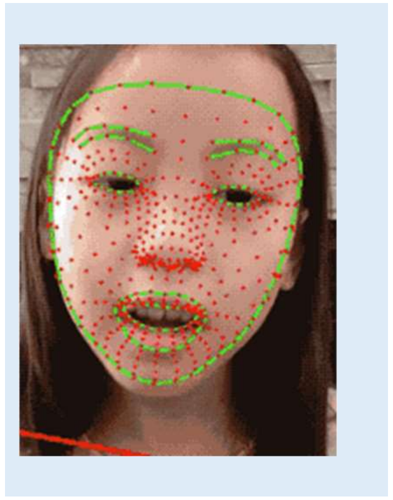
Mediapipe Solution

Utilize Mediapipe Pose Solution to use landmarks as datasets.

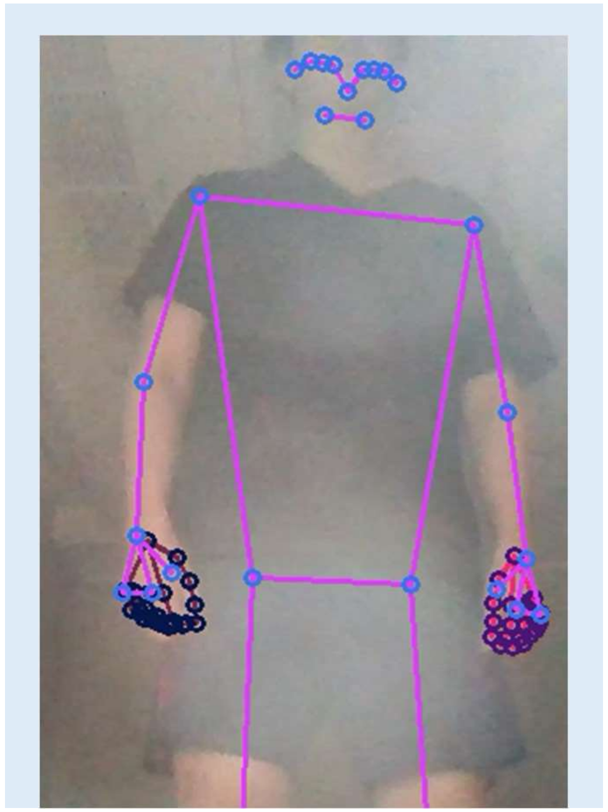


0. nose
1. left_eye_inner
2. left_eye
3. left_eye_outer
4. right_eye_inner
5. right_eye
6. right_eye_outer
7. left_ear
8. right_ear
9. mouth_left
10. mouth_right
11. left_shoulder
12. right_shoulder
13. left_elbow
14. right_elbow
15. left_wrist
16. right_wrist
17. left_pinky
18. right_pinky
19. left_index
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

Pose landmarks (# 33)



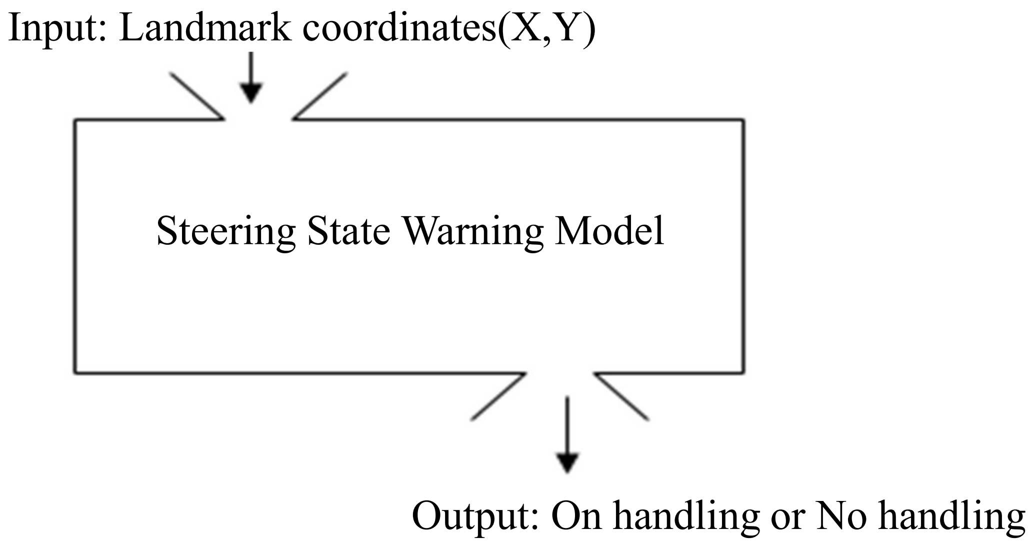
Face landmarks(#468)



Pose+Face landmarks(#501)

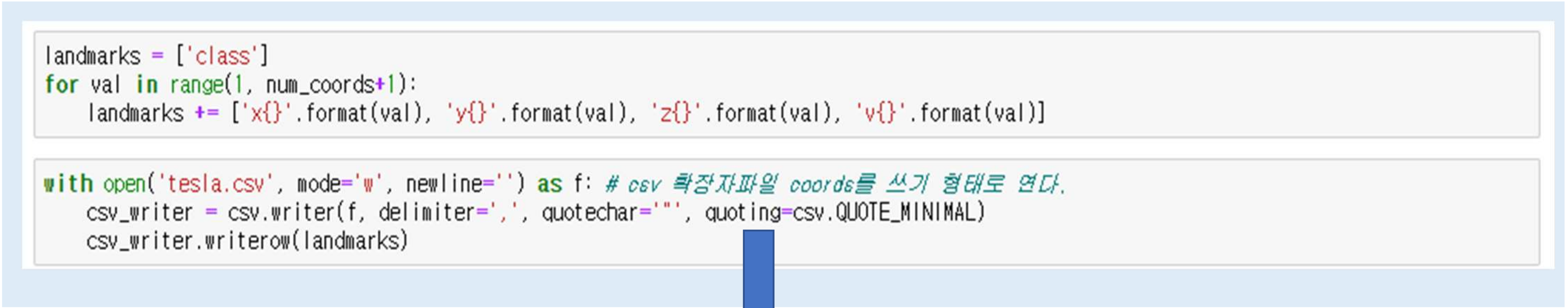
Define Machine-learning Model

Define classifier machine-learning model which gets real time coordinates of driver's landmark as input and class name as output whether driver is grabbing the steering wheel or not.


Input: Landmark coordinates(X,Y)
Steering State Warning Model
Output: On handling or No handling

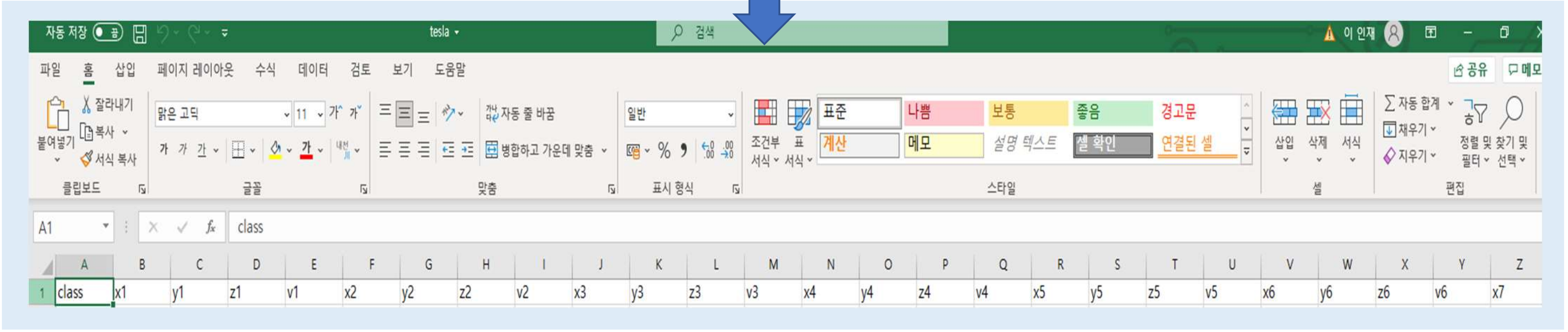
Collecting Dataset(Make CSV File)

Define coordinates of landmark as x,y,z and open tesla.csv file as write mode.



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

```
with open('tesla.csv', mode='w', newline='') as f: # csv 확장자파일 coords를 쓰기 형태로 연다.
    csv_writer = csv.writer(f, delimiter=',', quotechar='"', quoting=csv.QUOTE_MINIMAL)
    csv_writer.writerow(landmarks)
```



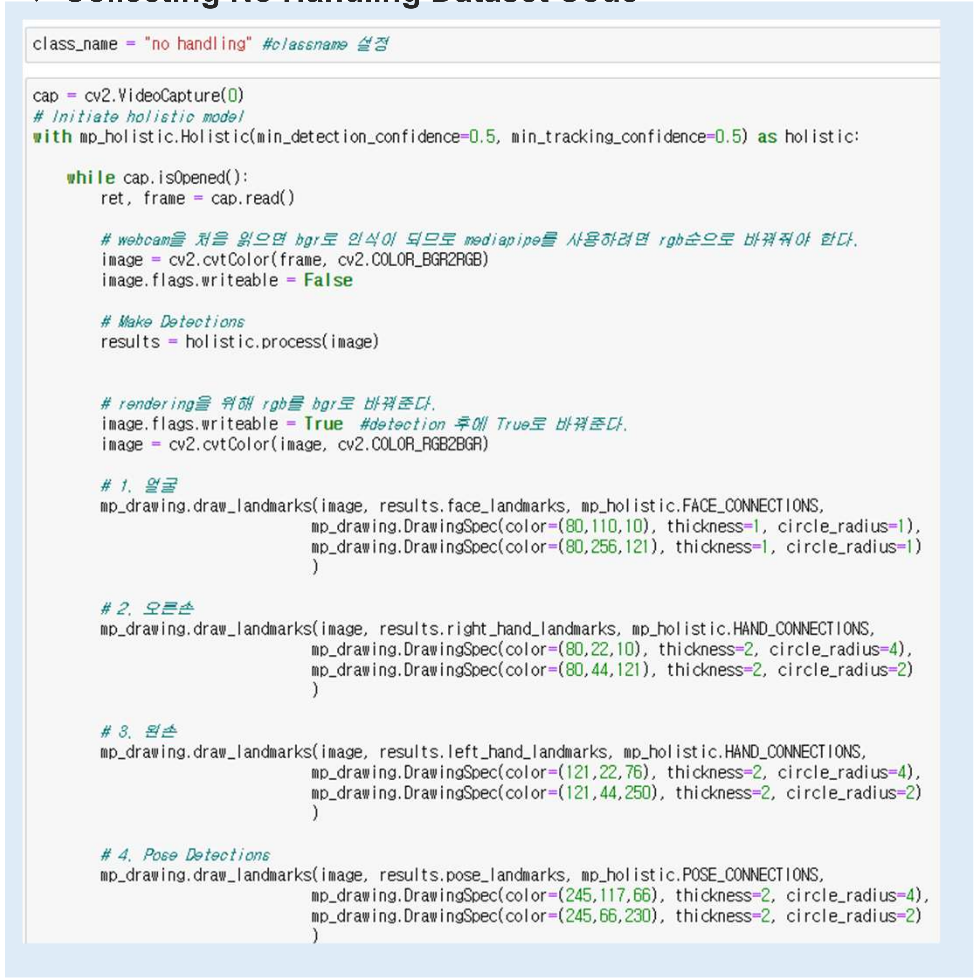
A1 | class

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
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

When open the made tesla.csv file you can See x1,y1 to x501 y501 which means landmarks coordinates

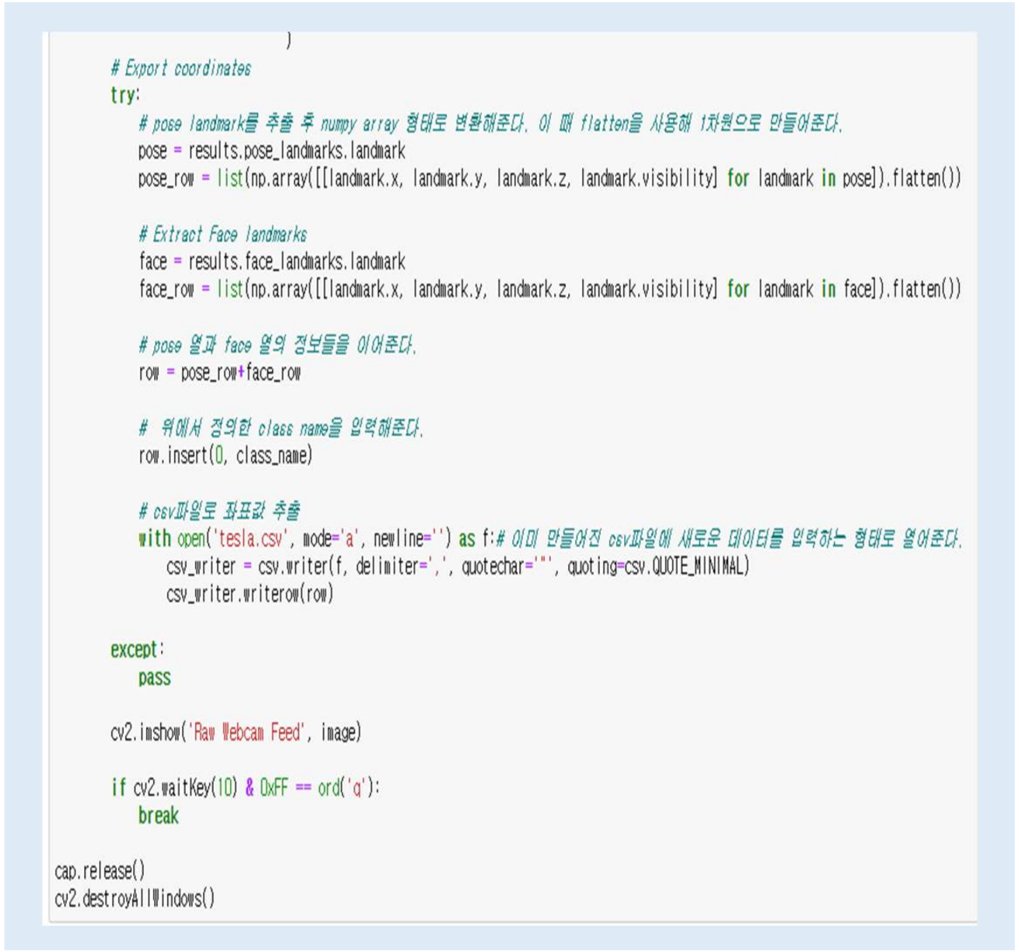
Collect Dataset(No Handling Code)

❖ Collecting No Handling Dataset Code



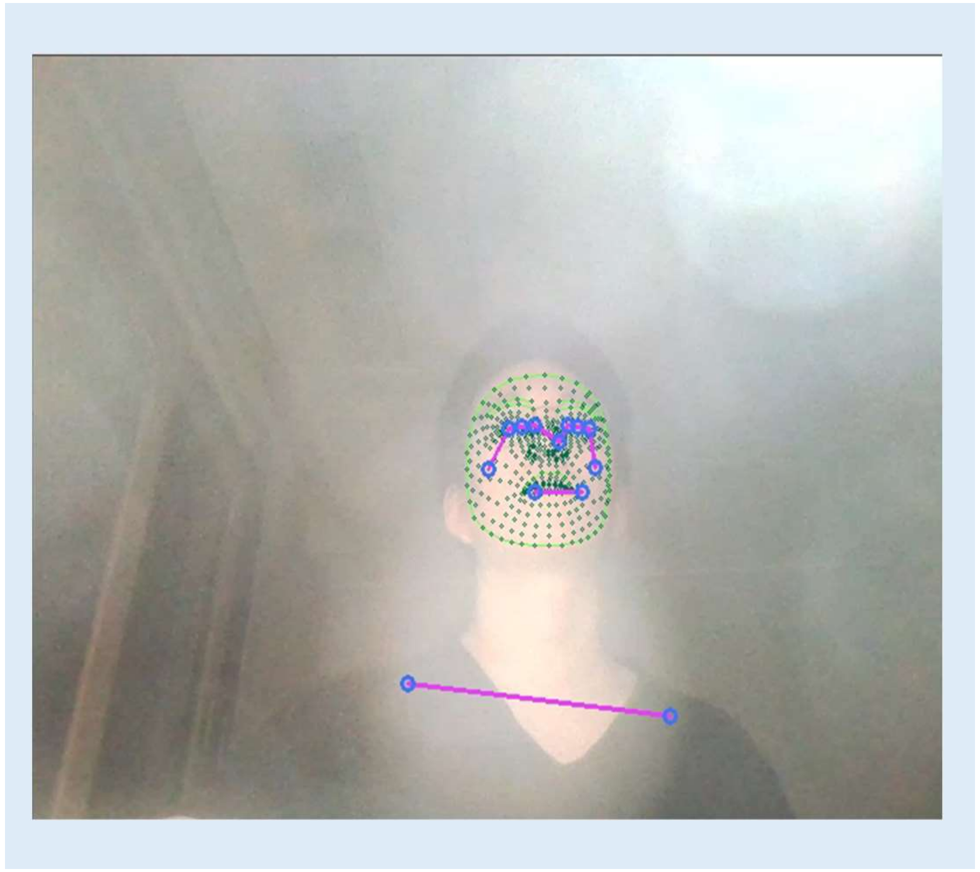
```
class_name = "no handling" #classname 설정
```

```
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()

        # webcam을 처음 읽으면 bgr로 인식이 되므로 mediapipe를 사용하려면 rgb순으로 바꿔줘야 한다.
        image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        image.flags.writeable = False

        # Make Detections
        results = holistic.process(image)


        # rendering을 위해 rgb를 bgr로 바꿔준다.
        image.flags.writeable = True  #detection 후에 True로 바꿔준다.
        image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)

        # 1. 얼굴
        mp_drawing.draw_landmarks(image, results.face_landmarks, mp_holistic.FACE_CONNECTIONS,
                                 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. 오른손
        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. 왼손
        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:
            # pose landmark를 추출 후 numpy array 형태로 변환해준다. 이 때 flatten을 사용해 1차원으로 만들어준다.
            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())

            # pose 열과 face 열의 정보들을 이어준다.
            row = pose_row+face_row

            #  위에서 정의한 class name을 입력해준다.
            row.insert(0, class_name)

            # csv파일로 좌표값 추출
            with open('tesla.csv', mode='a', newline='') as f:# 이미 만들어진 csv파일에 새로운 데이터를 입력하는 형태로 열어준다.
                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()
```

Collecting Dataset(No handling)

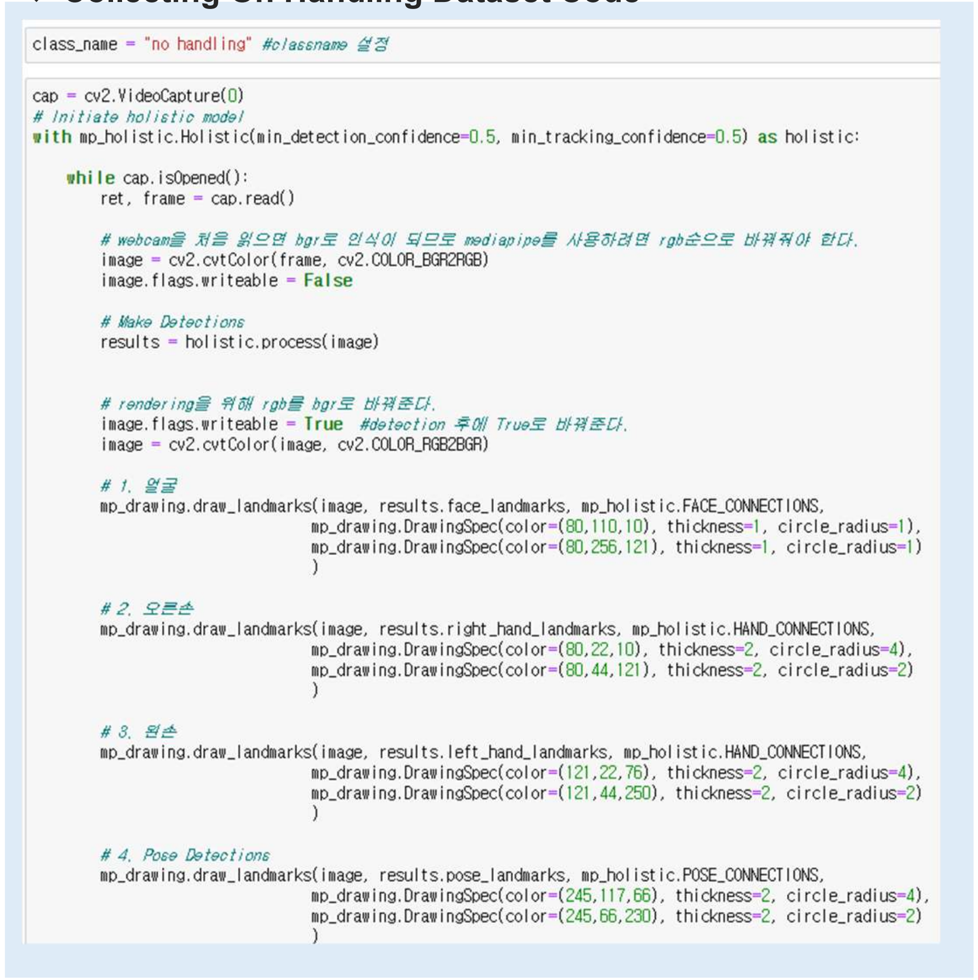


1. Run the code then collect no grabbing steering wheel datasets using web cam.

2. You should change the angle of face and upper body to get accurate machine-learning model.

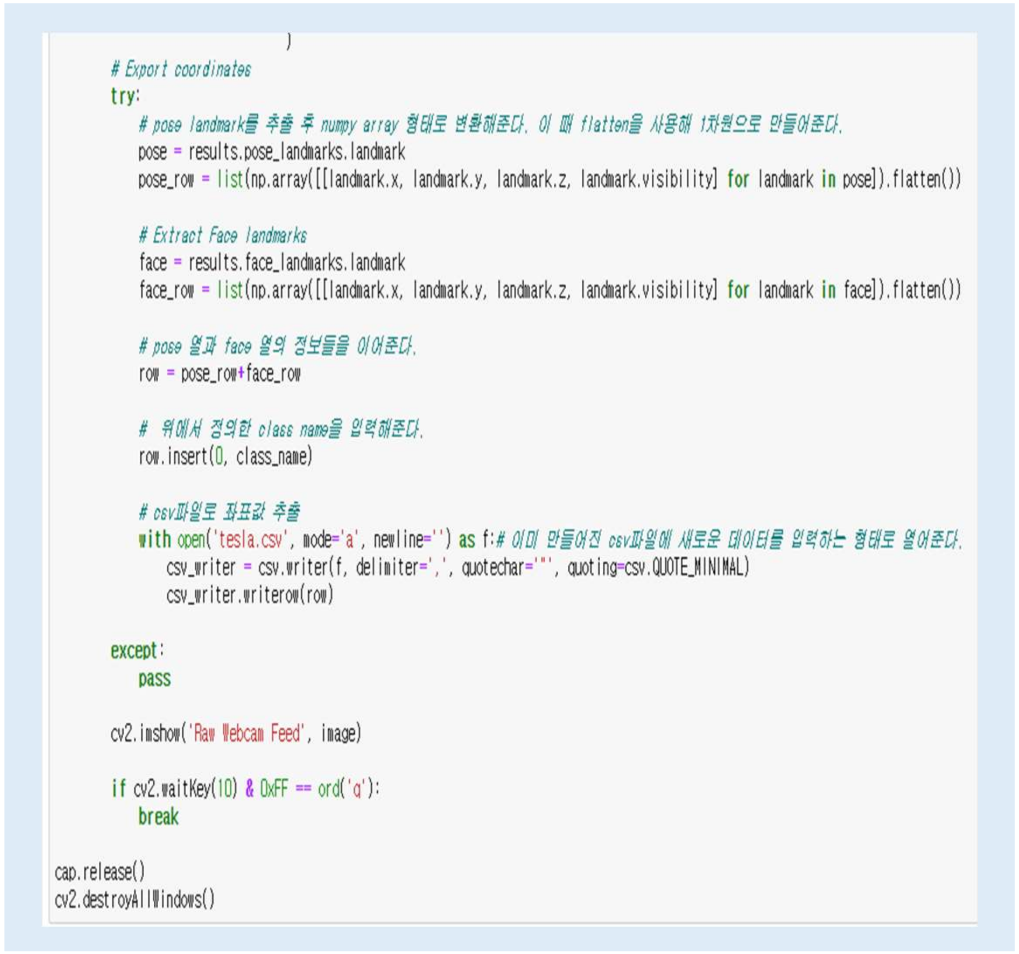
Collect Dataset(On Handling Code)

❖ **Collecting On Handling Dataset Code**



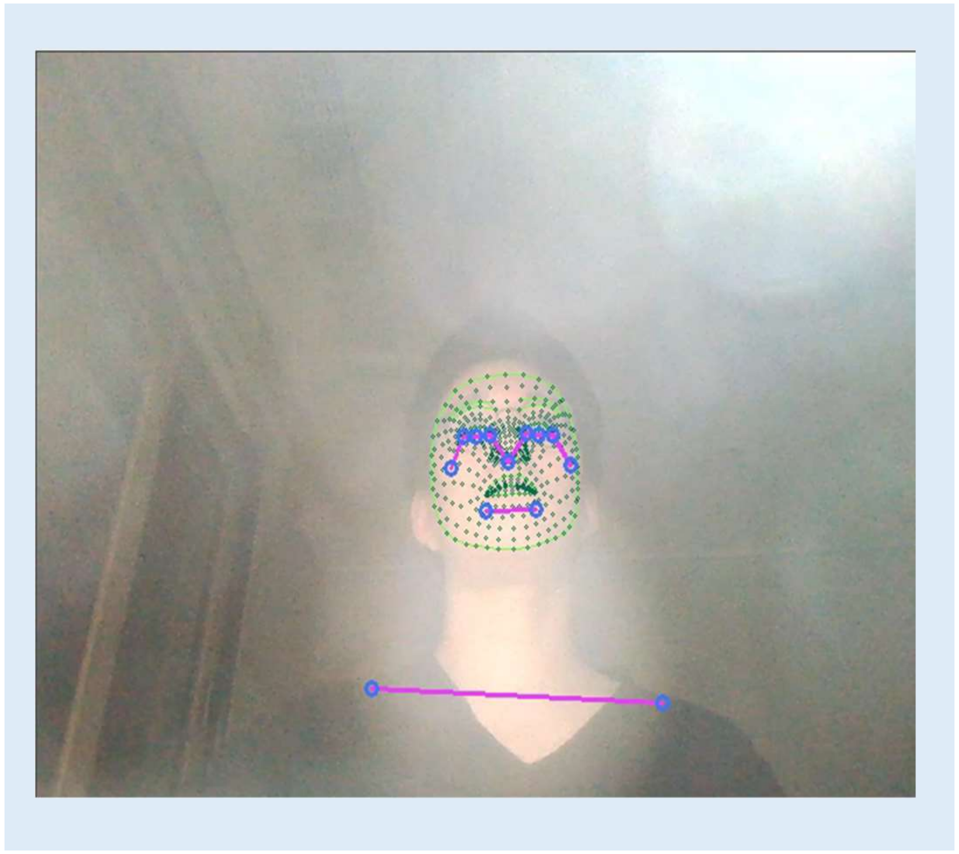
```
class_name = "no handling" #classname 설정
```

```
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()

        # webcam을 처음 읽으면 bgr로 인식이 되므로 mediapipe를 사용하려면 rgb순으로 바꿔줘야 한다.
        image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        image.flags.writeable = False

        # Make Detections
        results = holistic.process(image)


        # rendering을 위해 rgb를 bgr로 바꿔준다.
        image.flags.writeable = True  #detection 후에 True로 바꿔준다.
        image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)

        # 1. 얼굴
        mp_drawing.draw_landmarks(image, results.face_landmarks, mp_holistic.FACE_CONNECTIONS,
                                  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. 오른손
        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. 왼손
        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:
            # pose landmark를 추출 후 numpy array 형태로 변환해준다. 이 때 flatten을 사용해 1차원으로 만들어준다.
            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())

            # pose 열과 face 열의 정보들을 이어준다.
            row = pose_row+face_row

            #  위에서 정의한 class name을 입력해준다.
            row.insert(0, class_name)

            # csv파일로 좌표값 추출
            with open('tesla.csv', mode='a', newline='') as f:# 이미 만들어진 csv파일에 새로운 데이터를 입력하는 형태로 열어준다.
                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()
```

Collecting Dataset(On Handling)

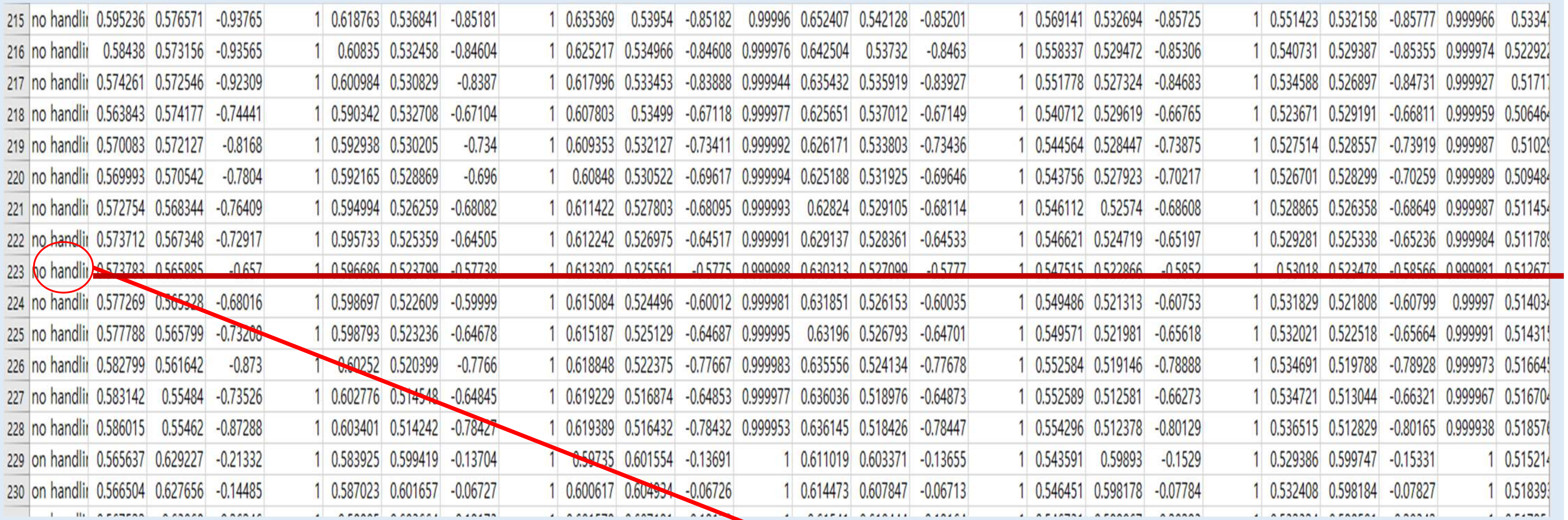


1. Run the code then collect grabbing steering wheel datasets using web cam.

2. You should change the angle of face and location of arms(one arm grabbing steering wheel, two arm grabbing steering wheel) to get accurate machine-learning model.

Collected **Dataset**

❖ **Dataset in telsa.csv files**

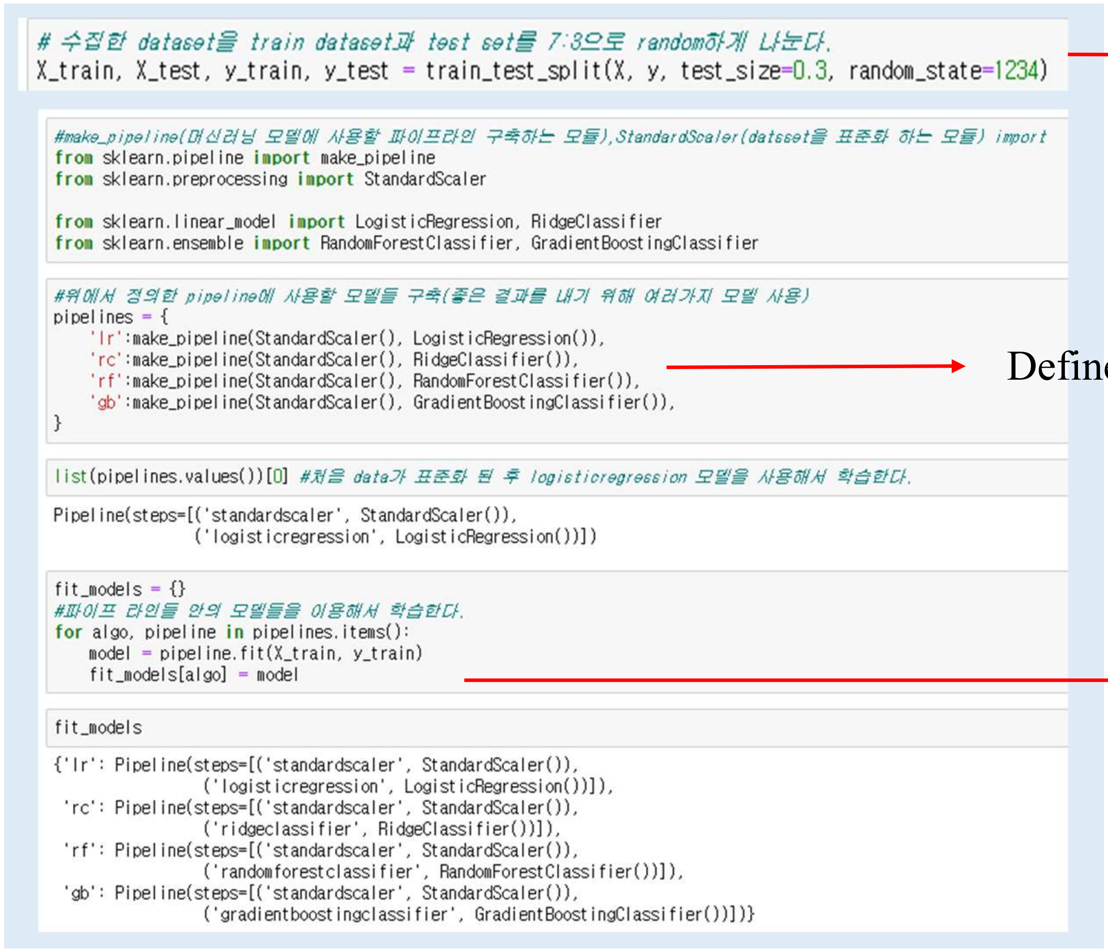


215	no handli	0.595236	0.576571	-0.93765	1	0.618763	0.536841	-0.85181	1	0.635369	0.53954	-0.85182	0.99996	0.652407	0.542128	-0.85201	1	0.569141	0.532694	-0.85725	1	0.551423	0.532158	-0.85777	0.999966	0.53347
216	no handli	0.58438	0.573156	-0.93565	1	0.60835	0.532458	-0.84604	1	0.625217	0.534966	-0.84608	0.999976	0.642504	0.53732	-0.8463	1	0.558337	0.529472	-0.85306	1	0.540731	0.529387	-0.85355	0.999974	0.52292
217	no handli	0.574261	0.572546	-0.92309	1	0.600984	0.530829	-0.8387	1	0.617996	0.533453	-0.83888	0.999944	0.635432	0.535919	-0.83927	1	0.551778	0.527324	-0.84683	1	0.534588	0.526897	-0.84731	0.999927	0.51717
218	no handli	0.563843	0.574177	-0.74441	1	0.590342	0.532708	-0.67104	1	0.607803	0.53499	-0.67118	0.999977	0.625651	0.537012	-0.67149	1	0.540712	0.529619	-0.66765	1	0.523671	0.529191	-0.66811	0.999959	0.50646
219	no handli	0.570083	0.572127	-0.8168	1	0.592938	0.530205	-0.734	1	0.609353	0.532127	-0.73411	0.999992	0.626171	0.533803	-0.73436	1	0.544564	0.528447	-0.73875	1	0.527514	0.528557	-0.73919	0.999987	0.51029
220	no handli	0.569993	0.570542	-0.7804	1	0.592165	0.528869	-0.696	1	0.60848	0.530522	-0.69617	0.999994	0.625188	0.531925	-0.69646	1	0.543756	0.527923	-0.70217	1	0.526701	0.528299	-0.70259	0.999989	0.50948
221	no handli	0.572754	0.568344	-0.76409	1	0.594994	0.526259	-0.68082	1	0.611422	0.527803	-0.68095	0.999993	0.62824	0.529105	-0.68114	1	0.546112	0.52574	-0.68608	1	0.528865	0.526358	-0.68649	0.999987	0.51145
222	no handli	0.573712	0.567348	-0.72917	1	0.595733	0.525359	-0.64505	1	0.612242	0.526975	-0.64517	0.999991	0.629137	0.528361	-0.64533	1	0.546621	0.524719	-0.65197	1	0.529281	0.525338	-0.65236	0.999984	0.51178
223	no handli	0.573783	0.565885	-0.657	1	0.596686	0.523799	-0.57738	1	0.613302	0.525561	-0.5775	0.999988	0.630313	0.527099	-0.5777	1	0.547515	0.522866	-0.5852	1	0.53018	0.523478	-0.58566	0.999981	0.51267
224	no handli	0.577269	0.565328	-0.68016	1	0.598697	0.522609	-0.59999	1	0.615084	0.524496	-0.60012	0.999981	0.631851	0.526153	-0.60035	1	0.549486	0.521313	-0.60753	1	0.531829	0.521808	-0.60799	0.99997	0.51403
225	no handli	0.577788	0.565799	-0.73208	1	0.598793	0.523236	-0.64678	1	0.615187	0.525129	-0.64687	0.999995	0.63196	0.526793	-0.64701	1	0.549571	0.521981	-0.65618	1	0.532021	0.522518	-0.65664	0.999991	0.51431
226	no handli	0.582799	0.561642	-0.873	1	0.60252	0.520399	-0.7766	1	0.618848	0.522375	-0.77667	0.999983	0.635556	0.524134	-0.77678	1	0.552584	0.519146	-0.78888	1	0.534691	0.519788	-0.78928	0.999973	0.51664
227	no handli	0.583142	0.55484	-0.73526	1	0.602776	0.514548	-0.64845	1	0.619229	0.516874	-0.64853	0.999977	0.636036	0.518976	-0.64873	1	0.552589	0.512581	-0.66273	1	0.534721	0.513044	-0.66321	0.999967	0.51670
228	no handli	0.586015	0.55462	-0.87288	1	0.603401	0.514242	-0.78427	1	0.619389	0.516432	-0.78432	0.999953	0.636145	0.518426	-0.78447	1	0.554296	0.512378	-0.80129	1	0.536515	0.512829	-0.80165	0.999938	0.51857
229	on handli	0.565637	0.629227	-0.21332	1	0.583925	0.599419	-0.13704	1	0.59735	0.601554	-0.13691	1	0.611019	0.603371	-0.13655	1	0.543591	0.59893	-0.1529	1	0.529386	0.599747	-0.15331	1	0.51521
230	on handli	0.566504	0.627656	-0.14485	1	0.587023	0.601657	-0.06727	1	0.600617	0.604934	-0.06726	1	0.614473	0.607847	-0.06713	1	0.546451	0.598178	-0.07784	1	0.532408	0.598184	-0.07827	1	0.51839

After collecting dataset, You can see each row's first column contains class names(on handling or no handling), and rests of column contains coordinates of driver's landmarks.

Train

❖ Train Code



```
# 수집한 dataset을 train dataset과 test set를 7:3으로 random하게 나눈다.
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1234)
```

```
#make_pipeline(머신러닝 모델에 사용할 파이프라인 구축하는 모듈),StandardScaler(datsset을 표준화 하는 모듈) import
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler

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

```
#위에서 정의한 pipeline에 사용할 모델들 구축(좋은 결과를 내기 위해 여러가지 모델 사용)
pipelines = {
    'lr':make_pipeline(StandardScaler(), LogisticRegression()),
    'rc':make_pipeline(StandardScaler(), RidgeClassifier()),
    'rf':make_pipeline(StandardScaler(), RandomForestClassifier()),
    'gb':make_pipeline(StandardScaler(), GradientBoostingClassifier()),
}
```

```
list(pipelines.values())[0] #처음 data가 표준화 된 후 logisticregression 모델을 사용해서 학습한다.
```

```
Pipeline(steps=[('standardscaler', StandardScaler()),
                ('logisticregression', LogisticRegression())])
```

```
fit_models = {}
#파이프 라인들 안의 모델들을 이용해서 학습한다.
for algo, pipeline in pipelines.items():
    model = pipeline.fit(X_train, y_train)
    fit_models[algo] = model
```

```
fit_models
```

```
{'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())])}
```

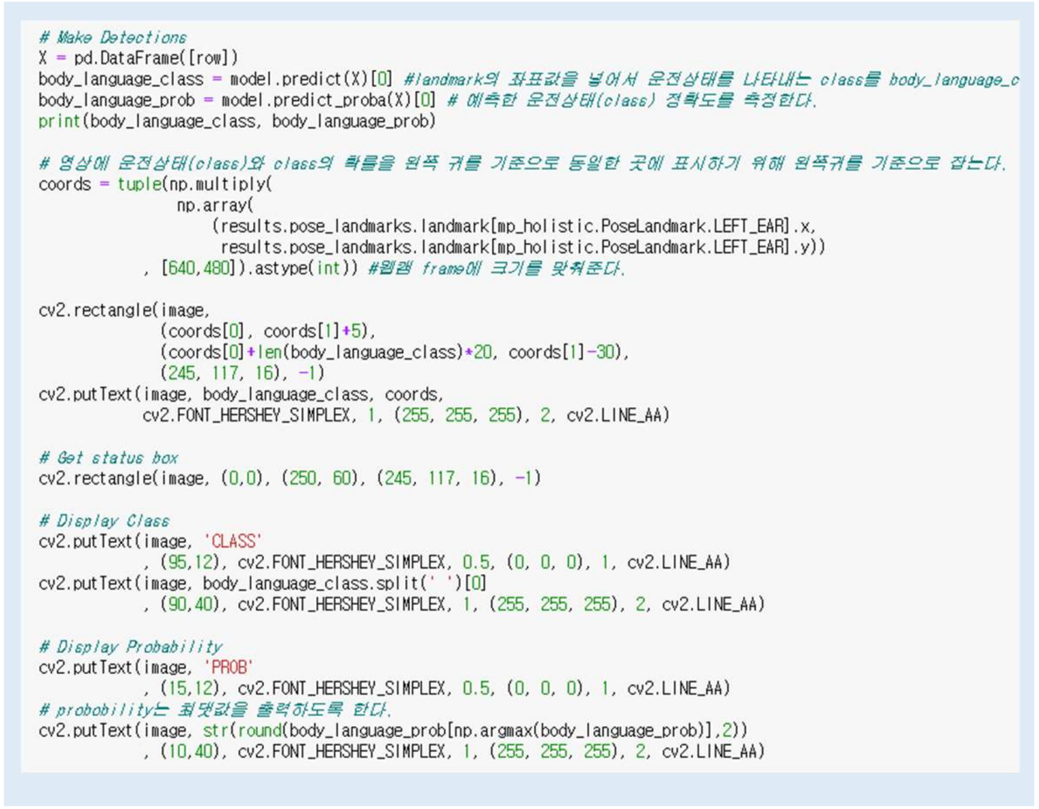
Split dataset(train: test=7:3)

Define pipeline containing 4 machine-learning model

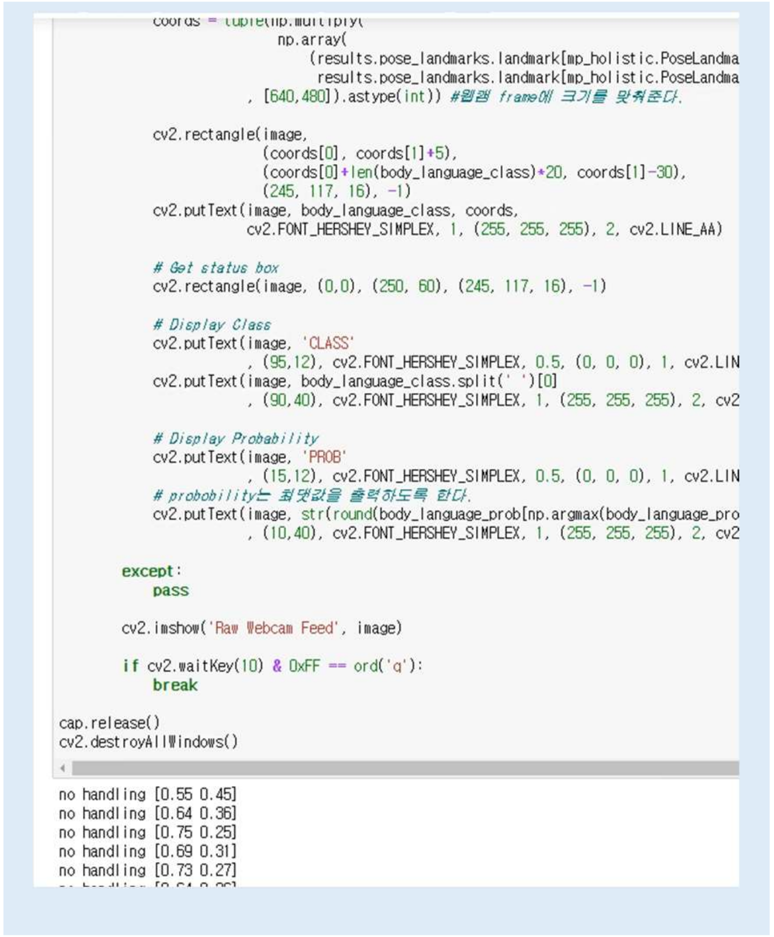
Train model

Test

❖ Test Code



```python
# Make Detections
X = pd.DataFrame([row])
body_language_class = model.predict(X)[0] #landmark의 좌표값을 넣어서 운전상태를 나타내는 class를 body_language_c
body_language_prob = model.predict_proba(X)[0] # 예측한 운전상태(class) 정확도를 측정한다.
print(body_language_class, body_language_prob)

# 영상에 운전상태(class)와 class의 확률을 왼쪽 귀를 기준으로 동일한 곳에 표시하기 위해 왼쪽귀를 기준으로 잡는다.
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)) #웹캠 frame에 크기를 맞춰준다.

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)
# probobility는 최댓값을 출력하도록 한다.
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)
```



```python
            coords = tuple(np.multiply(
                            np.array(
                                (results.pose_landmarks.landmark[mp_holistic.PoseLandma
                                 results.pose_landmarks.landmark[mp_holistic.PoseLandma
                        , [640,480]).astype(int)) #웹캠 frame에 크기를 맞춰준다.

            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.LIN
            cv2.putText(image, body_language_class.split(' ')[0]
                        , (90,40), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2, cv2

            # Display Probability
            cv2.putText(image, 'PROB'
                        , (15,12), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 0), 1, cv2.LIN
            # probobility는 최댓값을 출력하도록 한다.
            cv2.putText(image, str(round(body_language_prob[np.argmax(body_language_pro
                        , (10,40), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2, cv2

        except:
            pass

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

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

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

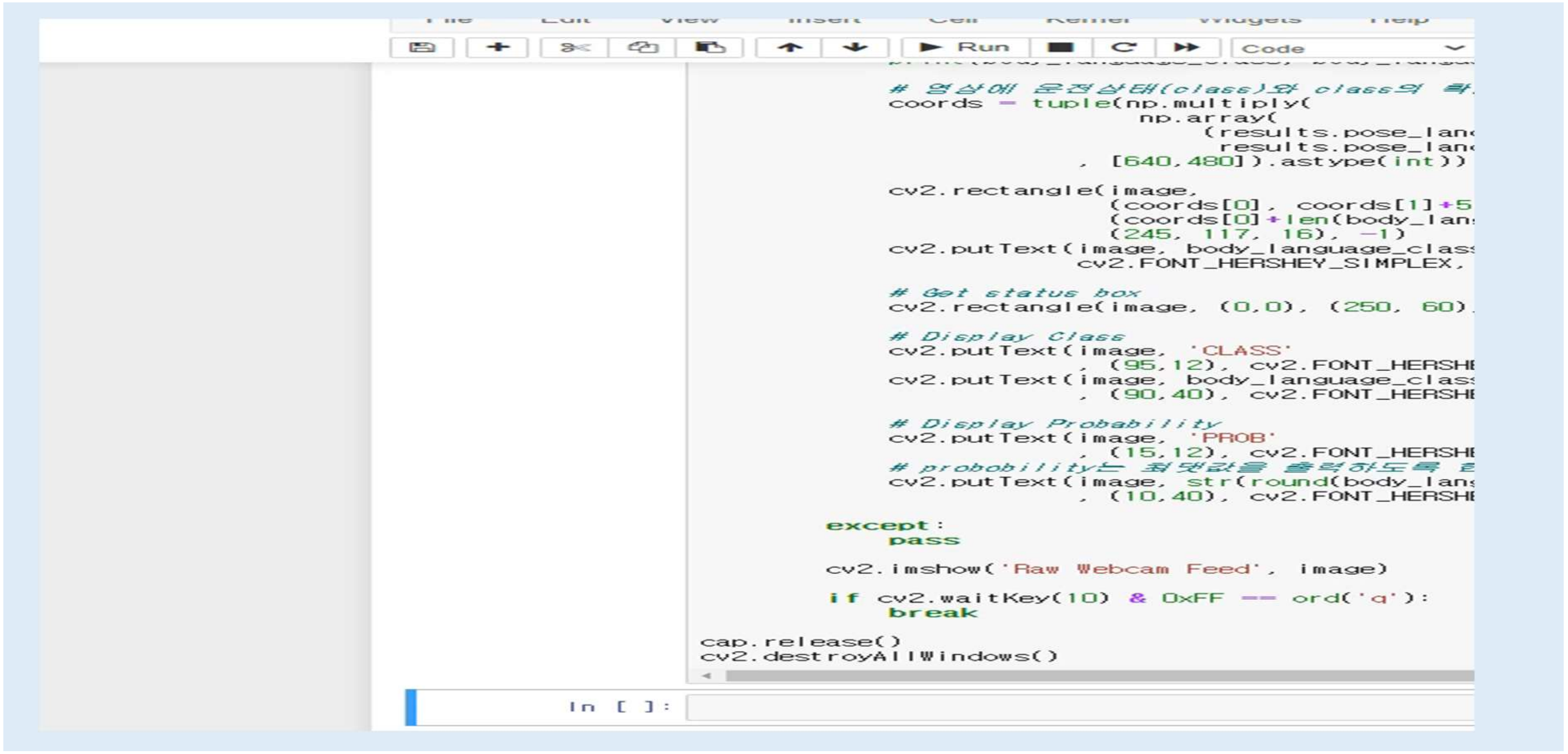
```
no handling [0.55 0.45]
no handling [0.64 0.36]
no handling [0.75 0.25]
no handling [0.69 0.31]
no handling [0.73 0.27]
```

Expected Performance

- ✓ The automobile **can notice that driver is not grabbing the steering wheel** although driver place weighty objects on the handle.
- ✓ The steering wheel state warning system **can be used with Tesla's technology.**
- ✓ Develop systems which **alarm or slow down the velocity** of automobile when driver is not grabbing the steering wheel.
- ✓ The combined systems(steering wheel state warning system + Tesla's autopilot) can **reduce the ratio of traffic accident largely.**

Driver State Warning(Additional)

❖**Video about driver state warning**



```
            # 영상에 운전상태(class)와 class의 확
            coords = tuple(np.multiply(
                            np.array(
                                (results.pose_lan
                                 results.pose_lan
                        , [640,480]).astype(int))

            cv2.rectangle(image,
                          (coords[0], coords[1]+5
                          (coords[0]+len(body_lan
                          (245, 117, 16), -1)
            cv2.putText(image, body_language_clas
                        cv2.FONT_HERSHEY_SIMPLEX,

            # Get status box
            cv2.rectangle(image, (0,0), (250, 60),

            # Display Class
            cv2.putText(image, 'CLASS'
                        , (95,12), cv2.FONT_HERSHE
            cv2.putText(image, body_language_clas
                        , (90,40), cv2.FONT_HERSHE

            # Display Probability
            cv2.putText(image, 'PROB'
                        , (15,12), cv2.FONT_HERSHE
            # probobility는 최댓값을 출력하도록 
            cv2.putText(image, str(round(body_lan
                        , (10,40), cv2.FONT_HERSHE

        except:
            pass

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

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

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

This system percepts whether driver is on drowsy driving or safe driving

Thank You!