Face ID

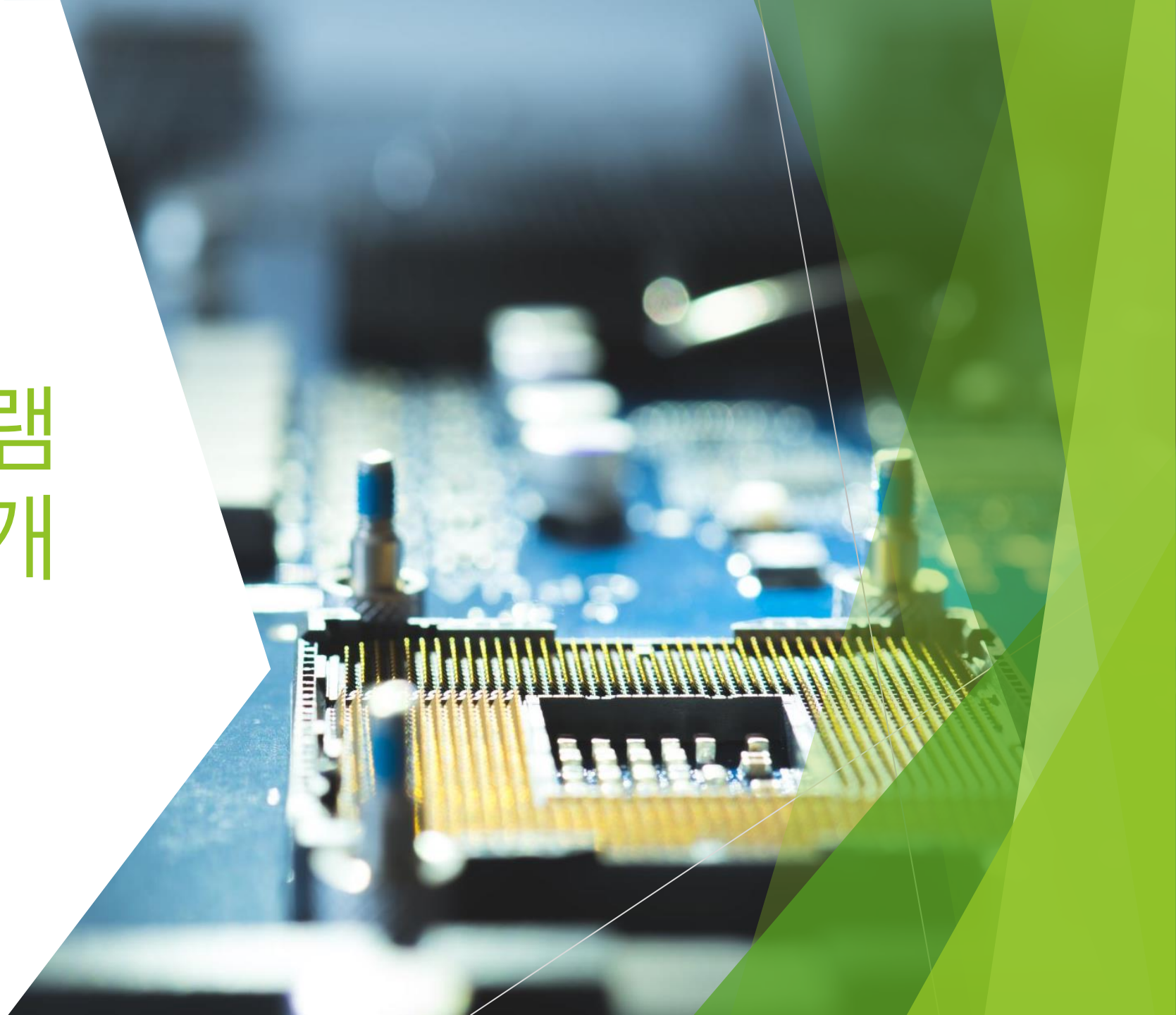
202104033 제갈륜

202104022 신성우

201904020 윤건용

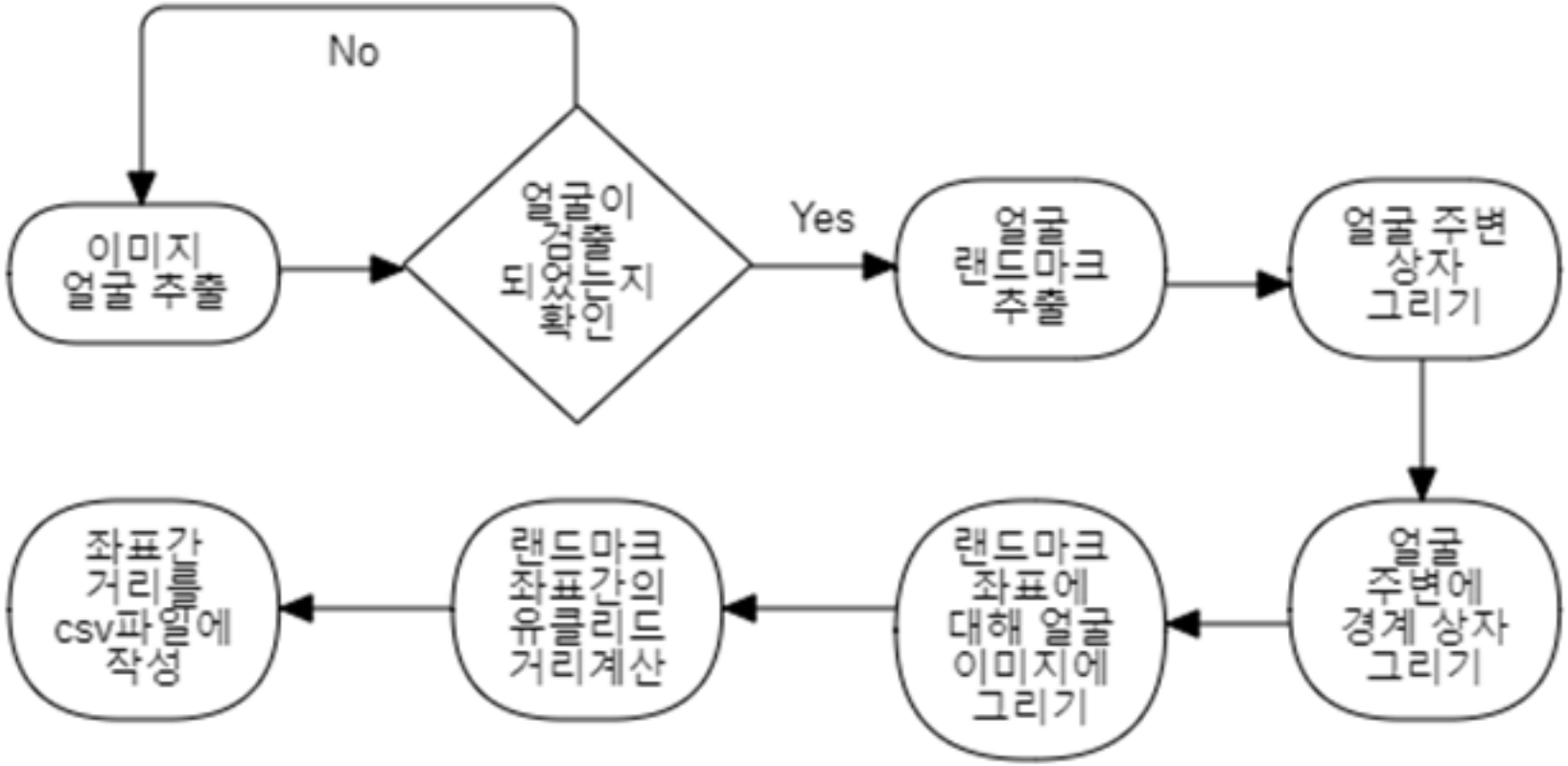
목차

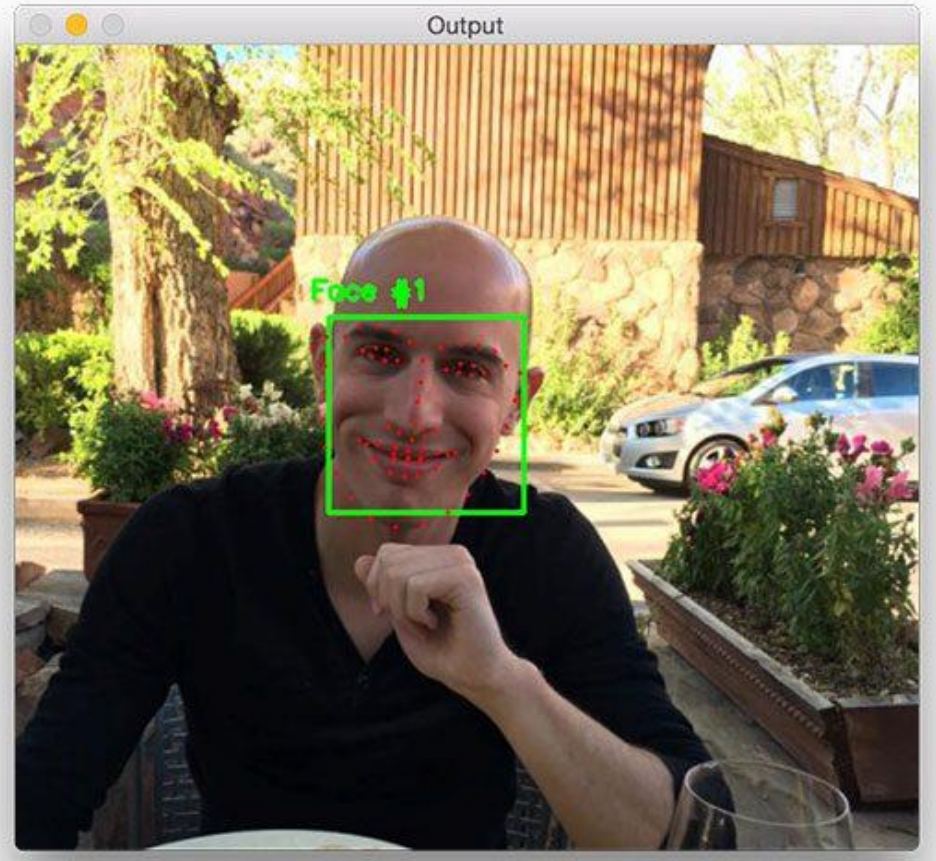
1. 프로그램 소개



2. 프로그램 필요성

3. 데이터 분석


No
이미지
얼굴 추출
얼굴이
검출
되었는지
확인
Yes
얼굴
랜드마크
추출
얼굴 주변
상자
그리기
얼굴
주변에
경계 상자
그리기
랜드마크
좌표에
대해 얼굴
이미지에
그리기
랜드마크
좌표간의
유클리드
거리계산
좌표간
거리를
csv파일에
작성


Output
Face #1

3. 데이터 분석

인덱스	이름	데이터 타입	값	설명
0	landmark	연속형	정수형	Landmark의 번호
1	1	연속형	정수형	Landmark 1번과의 거리
n	1	연속형	정수형	Landmark n번과의 거리

Landmark	1	2	3	4	5	6	7	8	...
1	0	32	64.19502	96.74709	128.7245	160.854	188.5099	211.4048	...
2	32	0	32.38827	65.11528	97.29337	129.8075	158.2403	182.4829	...
3	64.19502	32.38827	0	32.75668	65	97.71898	126.7162	152.1217	...
4	96.74709	65.11528	32.75668	0	32.28002	65.19202	94.82616	121.6059	...
5	128.7245	97.29337	65	32.28002	0	33.10589	63.51378	91.96739	...
6	160.854	129.8075	97.71898	65.19202	33.10589	0	31.14482	61.40033	...
...	...	...	...	...	...	...	...	...	...

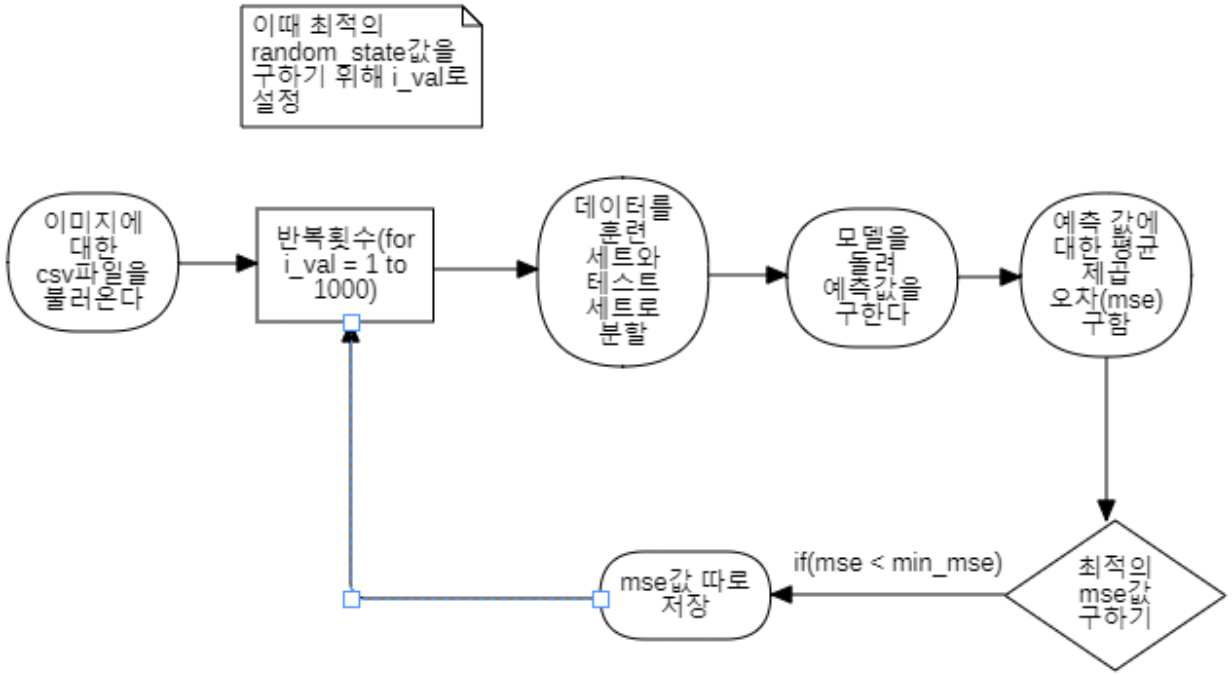
4. 사용 알고리즘

선현회귀 모델을 통해 학습시키는 random state값을 변화시켜
얼굴에 맞는 최적의 random state값을 찾는다.

가장 최적의 random state값을 가지고 mse, mae, R-squared 값을 구한다.

Radom state값을 가지고 판별할 데이터를 학습시켜
성능평가 값을 구하고 이 성능평가 값을 가지고 판별한다.

5. 순서도 및 구현


이때 최적의 random_state값을 구하기 위해 i_val로 설정
이미지에 대한 csv파일을 불러온다
반복횟수(for i_val = 1 to 1000)
데이터를 훈련 세트와 테스트 세트로 분할
모델을 돌려 예측값을 구한다
예측 값에 대한 평균 제곱 오차(mse) 구함
최적의 mse값 구하기
if(mse < min_mse)
mse값 따로 저장

CSV파일 생성 코드

```
for i in range(image_start, image_end+1):
    image_path = f"IMAGE/{i}.jpg"


    if os.path.exists(image_path):
        image = cv2.imread(image_path)
        gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    else:
        print(f"Error: No face detected in the image face_{i}.")
        continue

    faces = detector(gray)

    for j, face in enumerate(faces):
        landmarks = predictor(gray, face)
        landmarks = np.array([(landmarks.part(i).x, landmarks.part(i).y) for i in range(68)])

        x, y, w, h = face.left(), face.top(), face.width(), face.height()
        cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0), 2)

        for (x, y) in landmarks:
            cv2.circle(image, (x, y), 1, (0, 0, 255), -1)

        distances = pdist(landmarks)
        distance_matrix = squareform(distances)

        with open(f"landmark_distances/landmark_distances({i}).csv", "w", newline="") as csvfile2:
            writer2 = csv.writer(csvfile2)
            header = ["Landmark"] + [str(i) for i in range(1, 69)]
            writer2.writerow(header)
            writer2.writerows((i+1, *[f"{d:.8f}" for d in row]) for i, row in enumerate(distance_matrix))
```

최적의 random state 구하기

```
file_numbers = range(image_start, image_end+1)
i_values = range(1, 100)
for i_val in i_values:
    for file_number in range(image_start, image_end + 1):
        distances_file = f"landmark_distances/landmark_distances({file_number}).csv"

        if not os.path.exists(distances_file):
            continue
        data = pd.read_csv(distances_file)

        # 특징과 라벨 분리
        X = data.drop(columns=["Landmark"])
        y = data["Landmark"]

        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=i_val)

        # 선형 회귀 모델 생성 및 훈련
        model = LinearRegression()
        model.fit(X_train, y_train)

        # 검증 데이터를 사용하여 성능평가: 예측
        test_predictions = model.predict(X_test)

        # R-squared 계산
        r2 = r2_score(y_test, test_predictions)
        # print(f"R-squared: {r2:.4f}")

        # 평균 제곱 오차(MSE) 계산
        mse = mean_squared_error(y_test, test_predictions)
        # print(f"Mean Squared Error: {mse:.4f}")

        # 평균 절대 오차(MAE) 계산
        mae = mean_absolute_error(y_test, test_predictions)
        # print(f"Mean Absolute Error: {mae:.4f}")

        if mse < min_mse and mae < min_mae and r2 > min_r2:
            min_mse = mse
            min_mae = mae
            min_r2 = r2
            min_i = i_val
            min_file_number = file_number
```

```
# 결과 출력
print(f"가장 작은 MSE: {min_mse}")
print(f"가장 작은 MAE: {min_mae}")
print(f"가장 큰 R-squared: {max_r2}")
print(f"최소 MSE에 해당하는 i 값: {min_i}")
print(f"가장 작은 MSE에 해당하는 파일 번호: {min_file_number}")
```

✓ 0.0s

```
가장 작은 MSE: 0.04259587124337109
가장 작은 MAE: 0.17443148525575122
가장 큰 R-squared: 0.9999205603375609
최소 MSE에 해당하는 i 값: 75
가장 작은 MSE에 해당하는 파일 번호: 241
```

6. 결론