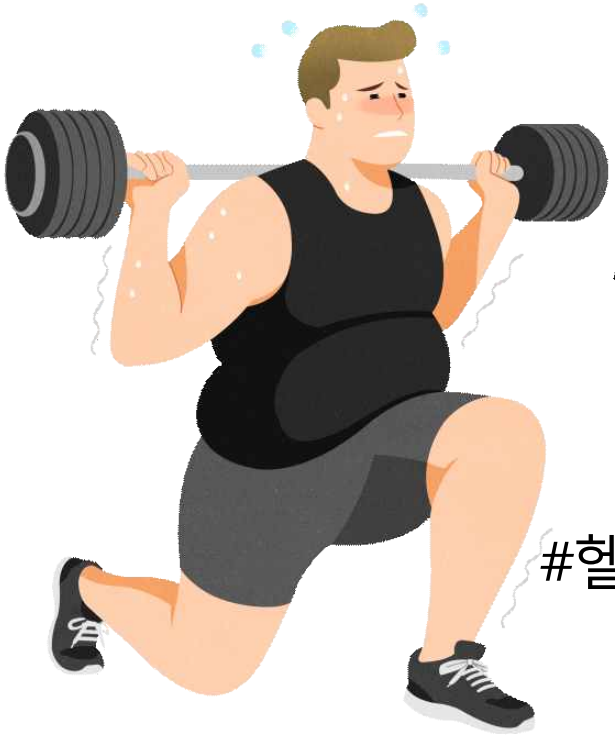
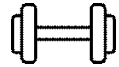
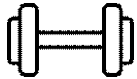


완전초보를위한 헬스헬퍼!

#헬스 #홈트 #몸짱 #웨이트 #유산소

CONTENTS



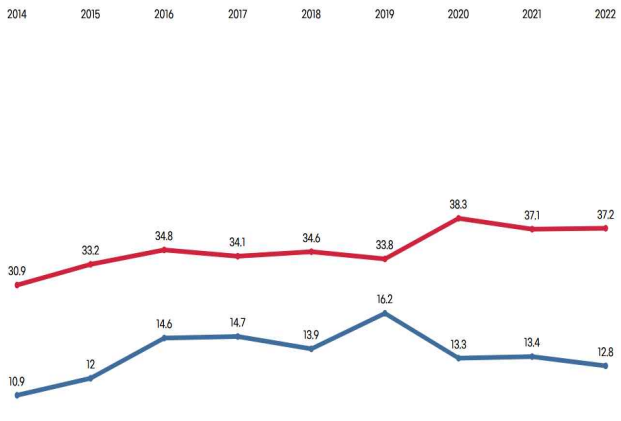


CHAPTER 01

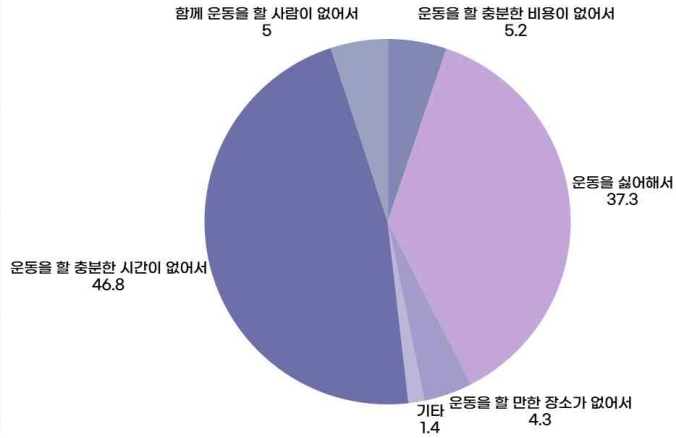
주제 선정

주제 선정

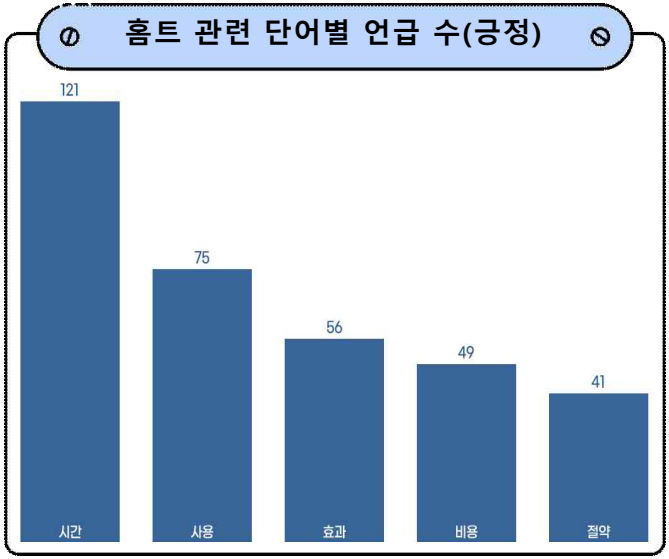
비만율 / 헬스 이용객 비율

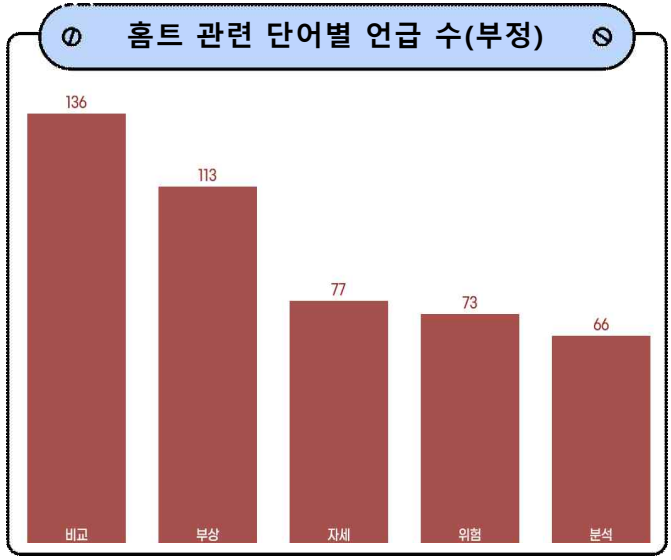


운동을 하지 못하는 이유

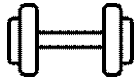


주제 선정





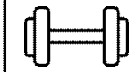
긍정에 시간이 있는 것을 보아 시간이 부족한 사람의 경우 홈트를 하는
경우가 많은 반면 부상에 대한 두려움이 있는것으로 보임



CHAPTER 02

프로젝트 수행 개요

프로젝트 수행 개요



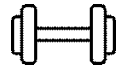
데이터 수집 → 모델 생성 → 모델 구현 → 모델 시연

!

본 프로젝트를 효율적으로 수행하기 위한 개요를 기획하여 진행

이를 보다 계획적으로 진행하기 위해 **WBS**제작

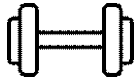
WBS(Work Breakdown Structure)



Health Helper

서동현,이원석,노승욱,경진우

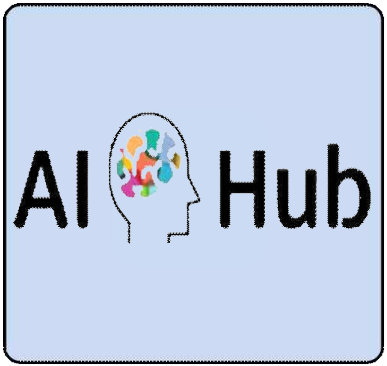
WBS	작업*	비고	시작일*	완료일*	총 작업량	계획 작업량	총 기간	계획 기간	실제 총기간	담당	산출물	계획	실적*
1	**Health Helper**		**2024-10-23**	**2024-11-18**	**460.0**	**421.0**	**153.0**	**124.0**	**0**			**100.00%**	100%
1.1	**프로젝트 준비**		**2024-10-23**	**2024-10-24**	**23**	**23**	**5**	**5**	**5**	**서동현**		**100.00%**	100%
1.1.2	프로젝트 팀원모집		2024-10-23	2024-10-24	8.0	8.0	2.0	2.0	2	서동현		100.00%	100%
1.1.3	프로젝트 장비구비		2024-10-23	2024-10-24	14.0	14.0	2.0	2.0	2	서동현		100.00%	100%
1.1.4	프로젝트 필요사항 준비		2024-10-23	2024-10-24	1.0	1.0	1.0	1.0	1	서동현		100.00%	100%
1.1.3	**프로젝트 진행**		**2024-10-25**	**2024-11-17**	**173**	**173**	**127**	**27**	**25**	**서동현,이원석,노승욱,경진우**		**100.00%**	100%
1.1.3.1	공유 노션 설정		2024-10-25	2024-10-26	4.0	4.0	2.0	2.0	1	서동현,이원석,노승욱,경진우		100.00%	100%
1.1.3.2	회의 장소 마련 및 첫 미팅		2024-10-27	2024-10-27	1.0	1.0	1.0	1.0	1	서동현,이원석,노승욱,경진우		100.00%	100%
	프로젝스 주간회의		2024-10-25	2024-11-18	56.0	56.0	8.0	8.0	8.0	서동현,이원석,노승욱,경진우		100.00%	100%
1.1.3.2	프로젝스 일일업무		2024-10-25	2024-11-18	112.0	112.0	116.0	16.0	16	서동현,이원석,노승욱,경진우		100.00%	100%
1.1.5	**중간결과 확인**		**2024-11-10**	**2024-11-10**	**5.0**	**5.0**	**15.0**	**2.0**	**2**	**서동현,이원석,노승욱,경진우**		**100.00%**	100%
1.1.5.1	중간 결과 확인		2024-11-10	2024-11-10	1.0	1.0	14.0	1.0	1	서동현,이원석,노승욱,경진우		100.00%	100%
1.1.5.2	중간 결과 피드백		2024-11-10	2023-11-09	4.0	4.0	1.0	1.0	1	서동현,이원석,노승욱,경진우		100.00%	100%
1.1.9	**프로젝트 종료**		**2024-11-13**	**2024-11-18**	**35.0**	**35.0**	**11.0**	**11.0**	**10**	**서동현,이원석,노승욱,경진우**		**100.00%**	100%
1.1.9.1	발표 자료 제작		2024-11-13	2024-11-17	15.0	15.0	5.0	5.0	5	노승욱, 이원석		100.00%	100%
1.1.9.3	시연 영상 제작		2024-11-13	2024-11-17	20.0	20.0	5.0	5.0	5	노승욱, 이원석		100.00%	100%
1.1.9.4	최종 발표		2024-11-18	2024-11-18	1.0	1.0	1.0	1.0	1	서동현		100.00%	100%
1.2.1.1	**운동 데이터셋 수집**		**2024-10-27**	**2024-10-28**	**8.0**	**8.0**	**4.0**	**4.0**	**4.0**	**서동현,이원석,노승욱,경진우**		**100.00%**	100%
1.2.1.1.1	맨몸 운동 라벨링 데이터셋		2024-10-27	2024-10-28	4.0	4.0	2.0	2.0	2.0	서동현,이원석,노승욱,경진우		100.00%	100%
1.2.1.1.2	바벨/덤벨 라벨링 데이터셋		2024-10-27	2024-10-28	4.0	4.0	2.0	2.0	2.0	서동현,이원석,노승욱,경진우		100.00%	100%
1.3.1	**AI(딥러닝) 모델 설계**		**2024-10-29**	**2024-11-02**	**50.0**	**50.0**	**10.0**	**10.0**	**10.0**	**서동현,이원석,노승욱,경진우**		**100.00%**	100%
1.2.1.1.1	맨몸 운동 라벨링 데이터셋		2024-10-29	2024-11-02	25.0	25.0	5.0	5.0	5.0	서동현,이원석,노승욱,경진우		100.00%	100%
1.2.1.1.2	바벨/덤벨 라벨링 데이터셋		2024-10-29	2024-11-02	25.0	25.0	5.0	5.0	5.0	서동현,이원석,노승욱,경진우		100.00%	100%
1.3.2	**AI(딥러닝) 모델 구현**		**2024-11-02**	**2023-11-15**	**224.0**	**224.0**	**28.0**	**28.0**	**28.0**	**서동현,경진우**		**100.00%**	100%
1.2.1.1.1	맨몸 운동 라벨링 데이터셋		2024-11-02	2023-11-15	112.0	112.0	14.0	14.0	14.0	서동현,경진우		100.00%	100%
1.2.1.1.2	바벨/덤벨 라벨링 데이터셋		2024-11-02	2023-11-15	112.0	112.0	14.0	14.0	14.0	서동현,경진우		100.00%	100%
1.3.3	**AI(딥러닝) 모델 시연**		**2024-11-13**	**2023-07-17**	**30.0**	**30.0**	**10.0**	**10.0**	**10.0**	**노승욱**		**100.00%**	100%
1.2.1.1.1	맨몸 운동 라벨링 데이터셋		2024-11-13	2023-07-17	15.0	15.0	5.0	5.0	5.0	노승욱		100.00%	100%
1.2.1.1.2	바벨/덤벨 라벨링 데이터셋		2024-11-13	2023-07-17	15.0	15.0	5.0	5.0	5.0	노승욱		100.00%	100%



CHAPTER 03

데이터 수집

데이터 수집



AI Hub

피트니스 자세 이미지 데이터

KOSIS 국가통계포털

비만율 통계 자료

헬스 이용객 통계 자료

네이버 블로그

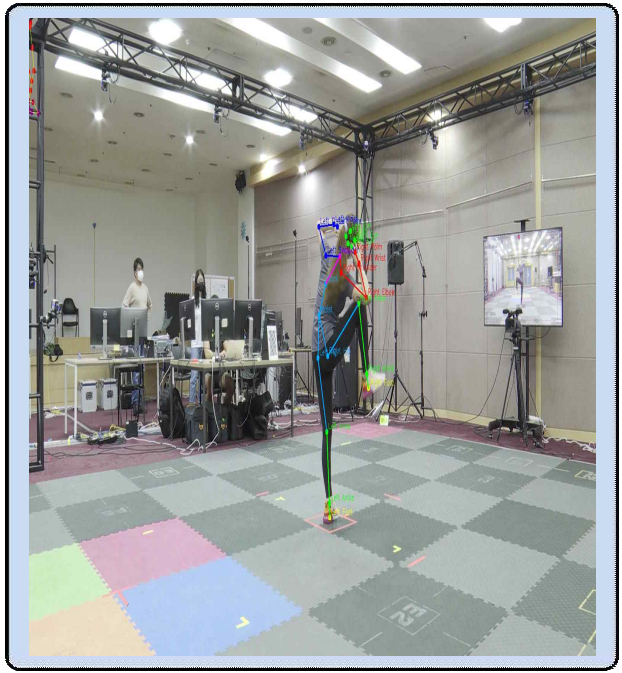
크롤링 데이터

데이터 설명

이미지

JSON 파일

2D 좌표 - 인체 2차원 좌표
3D 좌표 - 인체 3차원 좌표
운동명 - 이미지의 운동명
운동상태 - 3 ~ 5개의 정자세 측정 및 현재 상태 설명



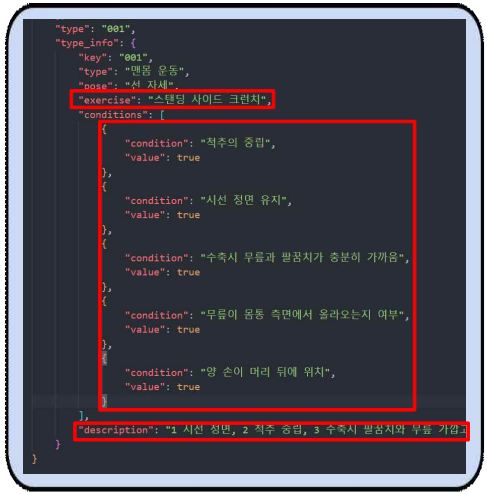
데이터 설명 및 전처리

1

2D 데이터
관절, 카메라 각도 라벨링

2

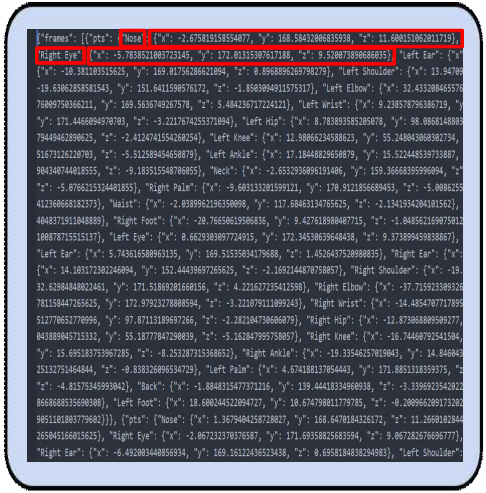
2D 데이터
운동상태 라벨링



데이터 설명 및 전처리

3

3D 데이터 라벨링



3D 데이터의 경우 2D 데이터와 다르게 **하나의 VIEW , 3개의 좌표**로 구성 되어 2D 데이터와 결합하기 위해서 전처리가 필요

데이터 설명 및 전처리

4

2D, 3D 데이터 결합

```
# 프레임별 데이터 처리
for frame_idx, frame_data in enumerate(data_2d["frames"]):
    frame_coords = []

    # 2D 관절 좌표
    for joint in joint_names:
        view_coords = [frame_data[view]["pts"].get(joint, {"x": 0, "y": 0}) for view in frame_data if "view" in view]
        avg_x = np.mean([coord["x"] for coord in view_coords])
        avg_y = np.mean([coord["y"] for coord in view_coords])
        frame_coords.extend([avg_x, avg_y])

    # 3D 데이터 처리 (존재하는 경우)
    if data_3d and frame_idx < len(data_3d["frames"]):
        for joint in joint_names:
            coord_3d = data_3d["frames"][frame_idx]["pts"].get(joint, {"x": 0, "y": 0, "z": 0})
            frame_coords.extend([coord_3d["x"], coord_3d["y"], coord_3d["z"]])

    X_data.append(frame_coords)
    y_labels.append(data_2d['type_info'].get('exercise', 'Unknown'))

# 레이블 인코딩
label_encoder = LabelEncoder()
y_labels_encoded = label_encoder.fit_transform(y_labels)

# 데이터 패딩 및 정규화
max_coords = max(len(frame) for frame in X_data)
X_data_padded = np.array([np.pad(frame, (0, max_coords - len(frame)), 'constant') for frame in X_data])
X_data_padded = X_data_padded / np.max(X_data_padded)

# 데이터 분할
X_train, X_val, y_train, y_val = train_test_split(X_data_padded, y_labels_encoded, test_size=0.2, random_state=42)

# DataLoader 준비
train_dataset = TensorDataset(torch.tensor(X_train, dtype=torch.float32), torch.tensor(y_train, dtype=torch.long))
val_dataset = TensorDataset(torch.tensor(X_val, dtype=torch.float32), torch.tensor(y_val, dtype=torch.long))

train_loader = DataLoader(train_dataset, batch_size=32, shuffle=True)
val_loader = DataLoader(val_dataset, batch_size=32)
```

결합 과정

2D 데이터:

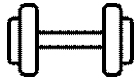
- 관절 이름별로 각 view에서 해당 관절의 (x, y) 좌표를 로드
- 여러 view가 존재할 경우 평균값을 계산하여 frame_coords에 추가

3D 데이터:

- 동일한 관절 이름별로, 프레임 인덱스에 해당하는 3D 좌표 (x, y, z)를 로드
- 3D 데이터는 frame_coords에 이어서 추가

최종 결합 결과:

- frame_coords는 [x1, y1, x2, y2, ..., x1_3d, y1_3d, z1_3d, x2_3d, y2_3d, z2_3d, ...] 형태로 구성됩니다.



CHAPTER 04

모델 생성

모델 설명

1

TSM 레이어

Temporal Shift Module

•**시계열 데이터**에서 시간적 특성을 학습하기 위한 매커니즘

• 입력 데이터의 일부 채널을 **시간축**으로 이동시키는 방식으로서 프레임 간의 상호작용을 학습.

•이는 주로 **시간적 정보가 중요한 동작 분석이나 행동 인식 작업**에서 사용

2

분류모델 구조

TSM-Based Model Structure

• **TSM** 레이어를 포함한 전체 모델의 설계

• 데이터의 공간적-시간적 특징을 학습하고 최종적으로 분류나 예측 작업을 수행

• **TSM** 레이어는 이 전체 구조에서 시간적 정보를 학습하는 데 도움을 주는 구성 요소로 사용

전체 레이어와 모델

```python
# TSM 레이어 정의
class TSMLayer(nn.Module):
    def __init__(self, in_channels, shift_size=1):
        super(TSMLayer, self).__init__()
        self.in_channels = in_channels
        self.shift_size = shift_size

    def forward(self, x):
        N, C, T = x.size()


        shift_left = x[:, :, :-self.shift_size]
        shift_right = x[:, :, self.shift_size:]
        shifted = torch.cat([shift_left, shift_right], dim=2)

        return shifted

# TSM 기반 모델 정의
class TSMModel(nn.Module):
    def __init__(self, num_classes, in_channels=120, shift_size=1):
        super(TSMModel, self).__init__()
        self.tsm = TSMLayer(in_channels, shift_size)
        self.conv1 = nn.Conv1d(in_channels, 64, kernel_size=3, padding=1)
        self.pool = nn.MaxPool1d(2)
        self.bn1 = nn.BatchNorm1d(64)
        self.lstm1 = nn.LSTM(64, 128, batch_first=True, bidirectional=True)
        self.dropout = nn.Dropout(0.2)
        self.fc1 = nn.Linear(128 * 2, 64)
        self.fc2 = nn.Linear(64, num_classes)

    def forward(self, x):
        x = self.tsm(x)
        x = self.conv1(x)
        x = self.pool(x)
        x = self.bn1(x)


        x = x.permute(0, 2, 1)

        x, _ = self.lstm1(x)
        x = self.dropout(x[:, -1, :])
        x = torch.relu(self.fc1(x))
        x = self.fc2(x)
        return x
```

모델 구조

Layer	Input Shape	Output Shape	Description
Input Data	[32, 120, 100]	[32, 120, 100]	모델에 입력되는 시계열 데이터 (관절 좌표)
TSM Layer	[32, 120, 100]	[32, 120, 100]	Temporal Shift Module. 채널 데이터를 시간축으로 이동하여 프레임 간 상호작용을 학습
Conv1D	[32, 120, 100]	[32, 64, 100]	1D 합성곱 레이어로 공간적 특징 추출.
MaxPooling + BatchNorm	[32, 64, 100]	[32, 64, 50]	MaxPooling으로 시간축 크기 절반 감소, BatchNorm으로 학습 안정성 강화.
Bi-LSTM	[32, 50, 64]	[32, 256]	양방향 LSTM으로 시계열 데이터의 시간적 특성 학습.
Dropout	[32, 256]	[32, 256]	과적합 방지를 위한 뉴런 드롭.
Fully Connected	[32, 256]	[32, 5]	최종 분류를 위한 완전 연결 레이어.

모델 결과 및 평가 지표

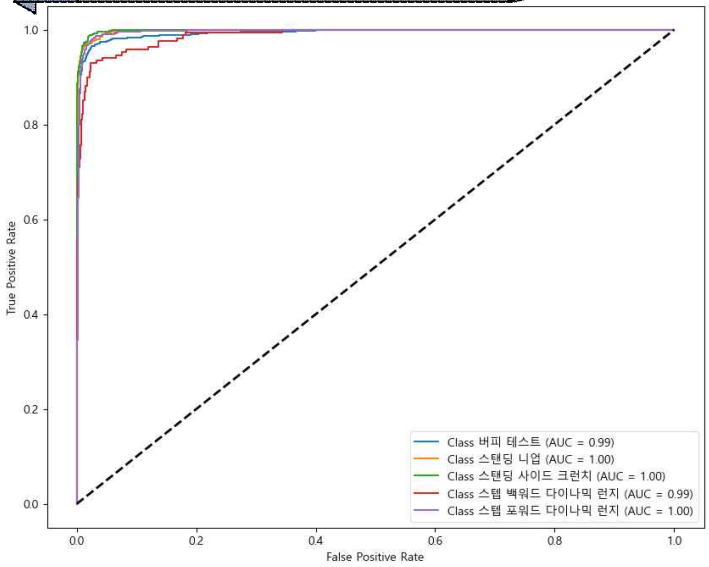
평가 지표

Class	Precision	Recall	F1-Score	샘플 수
버피 테스트	0.95	0.95	0.95	824
스탠딩 니업	0.94	0.96	0.95	413
스탠딩 사이드 크런치	0.97	0.97	0.97	849
스텝 백워드 다이나믹 런지	0.82	0.85	0.83	169
스텝 포워드 다이나믹 런지	0.97	0.94	0.96	805
Overall Accuracy	0.95	0.95	0.95	3060
Macro Avg	0.93	0.93	0.93	3060
Weighted Avg	0.95	0.95	0.95	3060

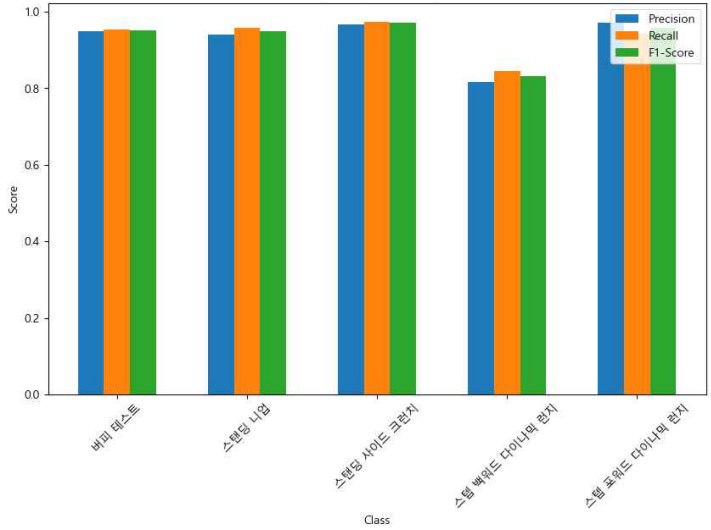
1 Confusion matrix

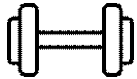
모델 결과 및 평가 지표

2 ROC Curve for Each Class



3 Precision, Recall, F1-Score per Class





CHAPTER 05

모델 및 홈페이지 구현

모델 시연

```python
# 관절 좌표 추출 함수
def extract_keypoints(results):
    keypoints = []
    landmarks = results.pose_landmarks.landmark
    for joint, landmark_enum in joint_mapping.items():
        if landmark_enum is None:
            keypoints.extend([0, 0, 0])  # (0, 0, 0)으로 처리
        else:
            landmark = landmarks[landmark_enum]
            keypoints.extend([landmark.x, landmark.y, landmark.z])

    # Neck, Pelvis, Spine 계산 (수식)
    left_shoulder = landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER]
    right_shoulder = landmarks[mp_pose.PoseLandmark.RIGHT_SHOULDER
    neck_x = (left_shoulder.x + right_shoulder.x) / 2
    neck_y = (left_shoulder.y + right_shoulder.y) / 2
    neck_z = (left_shoulder.z + right_shoulder.z) / 2
    keypoints.extend([neck_x, neck_y, neck_z])

    left_hip = landmarks[mp_pose.PoseLandmark.LEFT_HIP]
    right_hip = landmarks[mp_pose.PoseLandmark.RIGHT_HIP]
    pelvis_x = (left_hip.x + right_hip.x) / 2
    pelvis_y = (left_hip.y + right_hip.y) / 2
    pelvis_z = (left_hip.z + right_hip.z) / 2
    keypoints.extend([pelvis_x, pelvis_y, pelvis_z])

    spine_x = (pelvis_x + neck_x) / 2
    spine_y = (pelvis_y + neck_y) / 2
    spine_z = (pelvis_z + neck_z) / 2
    keypoints.extend([spine_x, spine_y, spine_z])

    if len(keypoints) < 120:
        keypoints.extend([0] * (120 - len(keypoints)))
    elif len(keypoints) > 120:
        keypoints = keypoints[:120]

    return keypoints
```

Mediapipe

model_complexity=2 옵션을 통해 모델의 복잡도를 높여 정확도를 향상시키고, 자세의 감지와 추적을 수행할 수 있도록 설정

Mediapipe가 추출한 관절 좌표 데이터는 운동 예측 모델에 입력되어 운동을 분류하고, 자세의 정확도를 평가

Mediapipe에서 지원하지 않는 **Neck(목), Pelvis(골반), Spine(척추)** 좌표는 다음과 같은 방식으로 계산
Neck: 양쪽 어깨 좌표의 중앙을 계산해 추정.
Pelvis: 양쪽 골반 좌표의 중앙을 계산해 추정.
Spine: Neck과 Pelvis 좌표의 중앙을 사용해 위치 계산.

2D, 3D 좌표를 결합하여 120개의 좌표를 구성.

모델 시연

1 관절간의 상호작용을 함수화

```python
import numpy as np
import mediapipe as mp

mp_pose = mp.solutions.pose

# 각도 계산 함수 정의
def calculate_angle(a, b, c):
    ba = (a.x - b.x, a.y - b.y)
    bc = (c.x - b.x, c.y - b.y)


    cosine_angle = (ba[0] * bc[0] + ba[1] * bc[1]) / (np.sqrt(ba[0]**2 + ba[1]**2) * np.sqrt(bc[0]**2 + bc[1]**2))
    angle = np.degrees(np.arccos(cosine_angle))
    return angle
```

2 운동 동작을 함수화

```python
# 개별 조건 평가 함수들
def evaluate_척추의_중립(landmarks):
    neck = landmarks[mp_pose.PoseLandmark.NOSE]
    waist = landmarks[mp_pose.PoseLandmark.LEFT_HIP]
    back = landmarks[mp_pose.PoseLandmark.RIGHT_HIP]
    spine_alignment = abs(neck.x - waist.x) < 0.05
    if not spine_alignment:
        print("척추가 중립 상태가 아닙니다. 척추를 곧게 펴세요.")
    return spine_alignment

def evaluate_시선_정면_유지(landmarks):
    left_eye = landmarks[mp_pose.PoseLandmark.LEFT_EYE]
    right_eye = landmarks[mp_pose.PoseLandmark.RIGHT_EYE]
    eye_alignment = abs(left_eye.y - right_eye.y) < 0.02
    if not eye_alignment:
        print("시선이 정면을 향하고 있지 않습니다. 시선을 정면으로 고정하세요.")
    return eye_alignment
```

자세평가

```python
# 조건 평가 함수
def evaluate_condition(landmarks, condition_text):
    if condition_text == "척추의 중립":
        return evaluate_척추의_중립(landmarks)
    elif condition_text == "시선 정면 유지":
        return evaluate_시선_정면_유지(landmarks)
    elif condition_text == "수축시 무릎과 팔꿈치가 충분히 가까움":
        return evaluate_수축시_무릎과_팔꿈치가_충분히_가까움(landmarks)
    elif condition_text == "무릎이 몸통 측면에서 올라오는지 여부":
        return evaluate_무릎이_몸통_측면에서_올라오는지_여부(landmarks)
    elif condition_text == "양 손이 머리 뒤에 위치":
        return evaluate_양_손이_머리_뒤에_위치(landmarks)
    elif condition_text == "팔 펴고 엎드렸을때 허리 휨 없음":
        return evaluate_팔_펴고_엎드렸을때_허리_휨_없음(landmarks)
    elif condition_text == "이완시 팔꿈치 90도":
        return evaluate_이완시_팔꿈치_90도(landmarks)
    elif condition_text == "가슴의 충분한 이동":
        return evaluate_가슴의_충분한_이동(landmarks)
    elif condition_text == "손의 위치 가슴 중앙 여부":
        return evaluate_손의_위치_가슴_중앙_여부(landmarks)
    elif condition_text == "고개 젖힘/숙임 여부":
        return evaluate_고개_젖힘_숙임_여부(landmarks)
    else:
        return False
```

모델 구현 영상 설명

1 **운동 종류 분류**

TSM 모델로 영상의
운동 분류

1 **운동 자세**

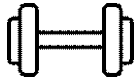
운동동작을 구분하여
정확한 자세를 파악

1 운동 상태

bad : 정확한 자세 2개이하

good : 정확한 자세 3개 이상

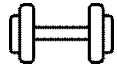
perfect : 모든 자세 정확



CHAPTER 06

한계점

한계점



대용량 데이터

1TB가 넘는 데이터로 인해
데이터 분석 및 사용의 어려움

서버 구축

컴퓨터의 메모리 부족으로 인해
모든 데이터를 한번에 학습시킬 수
없음

시간의 부족

데이터 분석 및 서버 구축을 위한
시간의 부족

위 세가지의 조건을 만족했다면 성능이 더 좋은 헬퍼가 완성되었을 것으로 기대됨

THANK YOU!
감사합니다!