

최종 발표

정신최리자-몸으로말해요

목차

1. HW 제작 완료
2. 데이터 수집 과정
3. 데이터 전처리
 1. 결측치 처리
 2. 표준화
4. 모델링

데이터 수집 과정

- Pyserial 라이브러리
- 아두이노 시리얼 포트 연결
- 문자열 데이터 , 기준으로 나눠 저장
- csv파일로 저장

"Timestamp", "No.", "roll1", "pitch1", "roll2", "pitch2",
"roll3", "pitch3", "roll4", "pitch4", "roll5", "pitch5",
"roll6", "pitch6", "FSR", "gesture"

```
import serial
import csv
import time
import msvcrt
import os # 파일 및 디렉토리 관리를 위한 os 모듈

data = []
t = 1

# 데이터 폴더 생성
data_folder = 'data'
if not os.path.exists(data_folder):
    os.makedirs(data_folder)

# 시리얼 포트 연결 (포트 이름과 baud rate는 환경에 맞게 수정)
try:
    ser = serial.Serial('COM20', 115200, timeout=1) # 예시: 'COM9' 포트, 115200 baud
except Exception as e:
    print("시리얼 포트 연결 실패:", e)
    exit()

# 시리얼 포트 안정화를 위해 잠시 대기 (장치 초기화 시간)
time.sleep(2)

print("시리얼 데이터 수신 시작 (ctrl+c로 종료)")

try:
    while True:
        # 한 줄씩 읽어 들임 (바이트 형태)
        line, buttonState = ser.readline().decode('utf-8').strip().split('|')

        # 문자열이 있으면 변수에 저장 후 출력
        if line:
            values = line.split(',')
            if len(values) == 13:
                timestamp = time.strftime('%Y-%m-%d %H:%M:%S')

                roll1, pitch1, roll2, pitch2, roll3, pitch3, roll4, pitch4, roll5, pitch5, roll6, pitch6, FSR = map(float, line.split(','))
                FSR = int(FSR)

                print(t, roll1, pitch1, roll2, pitch2, roll3, pitch3, roll4, pitch4, roll5, pitch5, roll6, pitch6, FSR, buttonState)
                data.append([timestamp, t, roll1, pitch1, roll2, pitch2, roll3, pitch3, roll4, pitch4, roll5, pitch5, roll6, pitch6, FSR, buttonState])
                t += 1

except KeyboardInterrupt:
    print("\n프로그램 종료합니다.")
finally:
    ser.close()

# 날짜 및 현재 시각을 파일 이름에 추가 (형식: YYYYMMDD_HHMMSS)
current_datetime = time.strftime("%Y%m%d_%H%M%S")
filename = f"{data_folder}/recorded_data_{current_datetime}.csv"
```

데이터 준비 완료

```
# data load-----
file1_path = '/content/recorded_data_20250121_235908.csv'
file2_path = '/content/recorded_data_20250122_002233-fsrSquarePlz.csv'

data = pd.concat([df1, df2], ignore_index=True)
print(data.head())
```

| | Timestamp | No. | roll1 | pitch1 | roll2 | pitch2 | roll3 | pitch3 | roll4 | pitch4 | roll5 | pitch5 | roll6 | pitch6 | FSR | gesture |
|---|---------------------|-----|-------|--------|-------|--------|-------|--------|-------|--------|-------|--------|--------|--------|-----|---------|
| 0 | 2025-01-21 23:59:10 | 1 | -1.58 | 1.09 | 3.06 | -0.77 | 1.63 | 1.4 | 1.49 | 1.2 | 14.13 | nan | -25.78 | 89.49 | 0 | A |
| 1 | 2025-01-21 23:59:10 | 2 | -1.67 | 1.23 | 3.1 | -0.87 | 1.94 | 1.46 | 1.51 | 1.27 | 25.95 | -88.33 | -41.8 | nan | 0 | n |
| 2 | 2025-01-21 23:59:11 | 3 | -1.42 | 0.84 | 2.89 | -0.6 | 1.31 | 1.43 | 1.23 | 1.22 | 19.73 | nan | -32.94 | nan | 0 | n |
| 3 | 2025-01-21 23:59:11 | 4 | -1.52 | 1.08 | 3.01 | -0.77 | 1.89 | 1.36 | 1.57 | 1.2 | 16.73 | -86.69 | -30.89 | nan | 0 | n |
| 4 | 2025-01-21 23:59:12 | 5 | -1.79 | 1.13 | -3.1 | -0.79 | 2.2 | 1.07 | 2 | 0.88 | 13.32 | -88.57 | -31.73 | 89.12 | 0 | n |
| | 2025-01-21 23:59:12 | 6 | -2.72 | 0.98 | -2.46 | -0.65 | 2.62 | 0.66 | 2.65 | 0.69 | 9.59 | nan | -37.02 | nan | 0 | n |
| | 2025-01-22 0:22 | 1 | -1.68 | 1 | -3.09 | -0.29 | 3.12 | 0.53 | 2.99 | 0.43 | 37.98 | nan | -60.33 | 89.75 | 0 | n |
| | 2025-01-22 0:22 | 2 | -2.27 | 0.98 | -2.72 | -0.26 | 3.11 | 0.2 | 3.03 | 0.21 | 33.52 | nan | -55.15 | 89.32 | 0 | n |
| | 2025-01-22 0:22 | 3 | -2.64 | 0.68 | -2.47 | -0.29 | 3.13 | 0.52 | 3 | 0.57 | 21.87 | -89.48 | -45.89 | 89.58 | 0 | n |
| | 2025-01-22 0:22 | 4 | -2.48 | 0.54 | -2.55 | -0.16 | -2.86 | 0.67 | -3.03 | 0.79 | 31.74 | -89.84 | -60.91 | 89.56 | 0 | n |
| | 2025-01-22 0:22 | 5 | -2.45 | 0.47 | -2.53 | -0.09 | -2.93 | 0.54 | -3.03 | 0.64 | 27.02 | -89.42 | -59.36 | 89.6 | 0 | C |
| | 2025-01-22 0:22 | 6 | -2.56 | 0.45 | -2.53 | -0.06 | -2.93 | 0.47 | -3.02 | 0.58 | 30.25 | nan | -59.02 | 89.58 | 0 | n |
| | 2025-01-22 0:22 | 7 | -2.61 | 0.46 | -2.26 | -0.26 | -2.61 | 0.44 | -2.88 | 0.52 | 32.31 | -89.77 | -51.32 | 89.05 | 0 | n |
| | 2025-01-22 0:22 | 8 | -2.67 | 0.46 | -2.42 | -0.22 | 3.08 | 0.29 | 3.04 | 0.31 | 27.17 | -89.32 | -45.3 | nan | 0 | n |
| | 2025-01-22 0:22 | 9 | -2.76 | 0.53 | -2.4 | -0.26 | 3.02 | 0.33 | 2.97 | 0.32 | 28.92 | -89.28 | -48.08 | 88.92 | 0 | n |
| | 2025-01-22 0:22 | 10 | -2.55 | 0.52 | -2.5 | -0.17 | -3.1 | 0.37 | 3.12 | 0.4 | 25.25 | nan | -47.92 | nan | 0 | n |
| | 2025-01-22 0:22 | 11 | -2.78 | 0.44 | -2.43 | -0.24 | 3.1 | 0.67 | 3.06 | 0.73 | 22.82 | nan | -48 | 89.31 | 0 | n |

데이터 전처리 - 결측치

- pitch5와 pitch6 column 제거

```
# pitch5와 pitch6 column에서 결측치 비율
missing_pitch5_ratio = filtered_C['pitch5'].isna().mean()
missing_pitch6_ratio = filtered_C['pitch6'].isna().mean()
missing_pitch5and6_ratio = filtered_C[filtered_C['pitch5'].isna() & filtered_C['pitch6'].isna()].shape[0] / filtered_C.shape[0]

# Display the results
print(f"Missing ratio in 'pitch5': {missing_pitch5_ratio:.2%}")
print(f"Missing ratio in 'pitch6': {missing_pitch6_ratio:.2%}")
print(f"Missing ratio in 'pitch5 and pitch6': {missing_pitch5and6_ratio:.2%}")
```

Missing ratio in 'pitch5': 50.27%
Missing ratio in 'pitch6': 48.52%
Missing ratio in 'pitch5 and pitch6': 24.71%

```
[27] # 'pitch5' and 'pitch6' column 제거
data = filtered_C.drop(columns=['pitch5', 'pitch6'])
print(data)
```

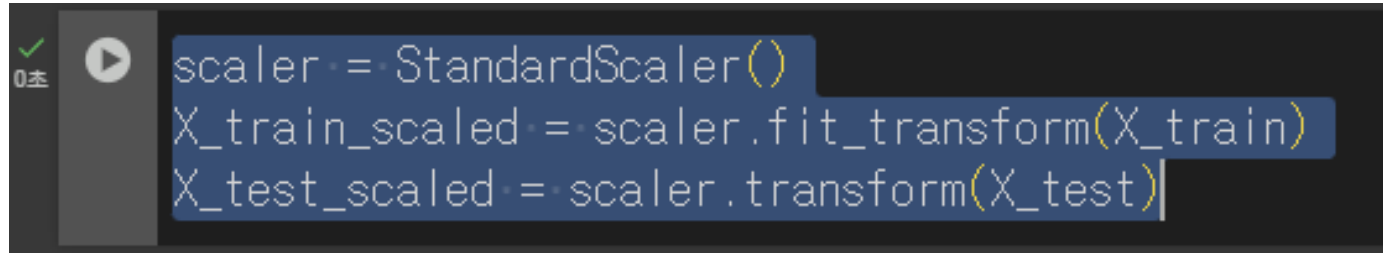
| | Timestamp | No. | roll1 | pitch1 | roll2 | pitch2 | roll3 | pitch3 | # |
|------|---------------------|------|-------|--------|-------|--------|-------|--------|---|
| 0 | 2025-01-21 23:59:10 | 1 | -1.58 | 1.09 | 3.06 | -0.77 | 1.63 | 1.40 | |
| 1 | 2025-01-21 23:59:10 | 2 | -1.67 | 1.23 | 3.10 | -0.87 | 1.94 | 1.46 | |
| 2 | 2025-01-21 23:59:11 | 3 | -1.42 | 0.84 | 2.89 | -0.60 | 1.31 | 1.43 | |
| 3 | 2025-01-21 23:59:11 | 4 | -1.52 | 1.08 | 3.01 | -0.77 | 1.89 | 1.36 | |
| 4 | 2025-01-21 23:59:12 | 5 | -1.79 | 1.13 | -3.10 | -0.79 | 2.20 | 1.07 | |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| 2624 | 2025-01-22 00:35:49 | 1546 | 2.70 | 0.89 | -1.91 | -0.62 | 2.40 | -0.25 | |
| 2625 | 2025-01-22 00:35:49 | 1547 | 2.35 | 0.47 | -1.53 | -0.71 | 2.41 | -0.08 | |
| 2626 | 2025-01-22 00:35:50 | 1548 | 2.41 | 0.59 | -1.57 | -0.65 | 2.07 | 0.06 | |
| 2627 | 2025-01-22 00:35:50 | 1549 | 2.50 | 0.76 | -1.45 | -0.77 | 1.98 | 0.01 | |
| 2628 | 2025-01-22 00:35:51 | 1550 | 2.80 | 0.96 | -1.45 | -0.77 | 2.02 | -0.03 | |

Train/test data 나누기

```
sensor_columns = ['roll1', 'pitch1', 'roll2', 'pitch2', 'roll3', 'pitch3', 'roll4', 'pitch4', 'roll5', 'roll6', 'FSR']  
X = data[sensor_columns]  
y = data['gesture']  
  
# Split data into training and testing sets  
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)  
X_train = X_train[:1000]  
y_train = y_train[:1000]
```

- 과적합을 방지하기 위해 랜덤으로 1000개 데이터만 학습

데이터 표준화

A screenshot of a code editor with a dark background. On the left, there is a vertical sidebar with a green checkmark and the text '0초' (0 seconds) next to a play button icon. The main area of the editor contains three lines of Python code: `scaler = StandardScaler()`, `X_train_scaled = scaler.fit_transform(X_train)`, and `X_test_scaled = scaler.transform(X_test)`. The code is written in a light blue font with syntax highlighting: parentheses and function names are yellow, and variable names and attributes are light blue.

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

- 모든 특성이 동일한 스케일(평균 0, 표준편차 1)을 가지도록 만든다
- 이로서 모델이 특정 특성에 대해 과도한 영향을 받는 문제를 줄일 수 있다.

모델 학습 및 정확도 산출

- 랜덤포레스트 모델을 사용했다.
- 과적합을 방지하기 위해 하이퍼파라미터 조정한다.
- 학습 후 정확도를 평가한다.

```
# 하이퍼파라미터 조정
model = RandomForestClassifier(
    n_estimators=100,      # 트리의 개수
    max_depth=10,         # 트리의 최대 깊이 (과적합 방지)
    min_samples_split=10, # 분할을 위한 최소 샘플 수 (과적합 방지)
    min_samples_leaf=4,   # 리프 노드의 최소 샘플 수 (과적합 방지)
    max_features='sqrt',  # 트리가 분할을 위한 최대 특징 수 (과적합 방지)
    random_state=42       # 재현성을 위한 랜덤 시드
)

# 훈련 데이터로 모델 학습
model.fit(X_train, y_train)

# 예측
y_pred = model.predict(X_test)

# 정확도 평가
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test, y_pred)
print(f"Test Accuracy: {accuracy:.4f}")
```

Test Accuracy: 0.8596

모델 저장 및 다운로드

```
[44] import joblib

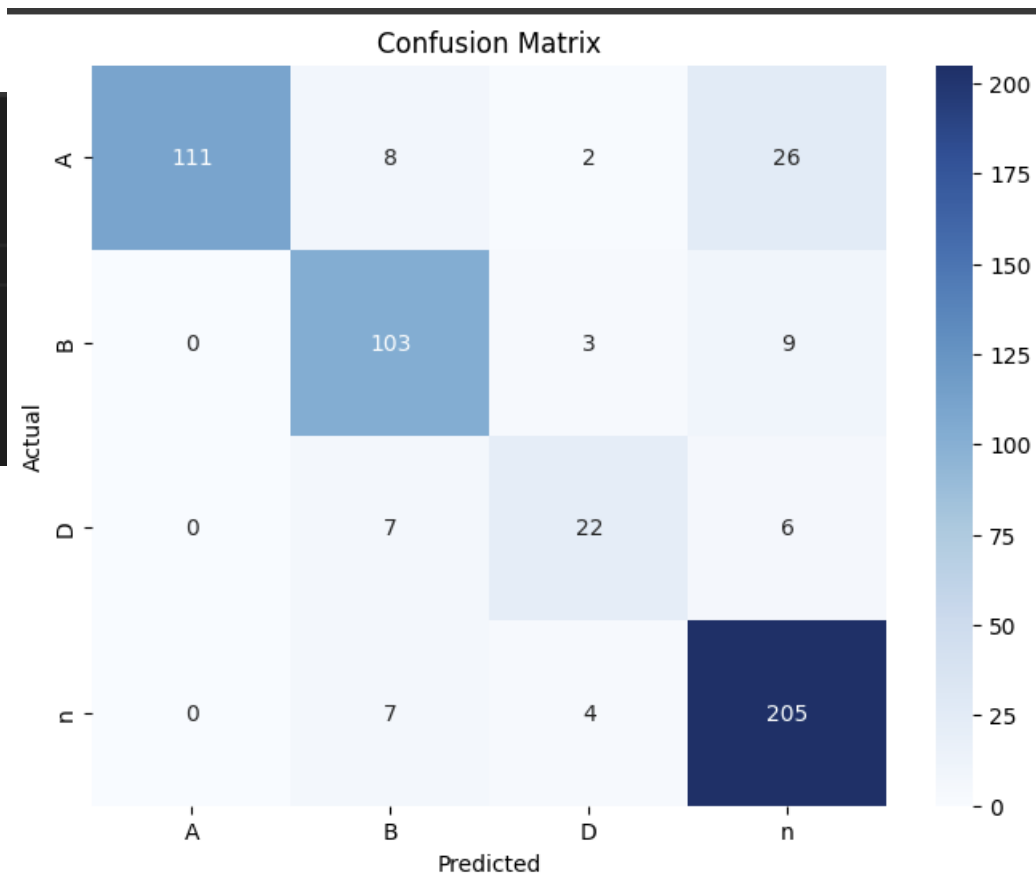
# 학습된 모델 (예: RandomForestClassifier, SVM 등)
model_path = "/content/model.pkl"

# 모델 저장
joblib.dump(model, model_path)

# Colab에서 다운로드 링크 생성
from google.colab import files
files.download(model_path)
```

히트맵

```
conf_matrix = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',
             xticklabels=model.classes_, yticklabels=model.classes_)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```



러닝커브

```
[42] # 학습 곡선 데이터 생성
train_sizes, train_scores, test_scores = learning_curve(
    model, X_train_scaled, y_train, cv=5, n_jobs=-1, train_sizes=np.linspace(0.1, 1.0, 10)
)
```

평균과 표준편차 계산

```
train_mean = train_scores.mean(axis=1)
train_std = train_scores.std(axis=1)
test_mean = test_scores.mean(axis=1)
test_std = test_scores.std(axis=1)
```

학습 곡선 시각화

```
plt.figure(figsize=(10, 6))
plt.plot(train_sizes, train_mean, label="Training accuracy")
plt.plot(train_sizes, test_mean, label="Cross-validation accuracy")
```

표준편차 영역 표시

```
plt.fill_between(train_sizes, train_mean - train_std, train_mean + train_std, label="Training std")
plt.fill_between(train_sizes, test_mean - test_std, test_mean + test_std, label="Cross-validation std")
```

그래프 제목과 레이블 설정

```
plt.title("Learning Curve (RandomForestClassifier)")
plt.xlabel("Training Size")
plt.ylabel("Accuracy")
plt.legend(loc="best")
plt.grid(True)
plt.show()
```

