* 제3회 ETRI 휴먼이해 인공지능 논문경진대회

**팀 명 : O O O**

**2024년 05월 19일**

*\*다른 개발환경에서 재현 검증이 진행되는 만큼, 각 항목에 대하여 팀에서 진행하신 내용을 최대한 상세히 작성해주시길 부탁드립니다.*

1. **라이브러리 및 데이터 (Library & Data)**

* - 사용된 라이브러리:  
   - pandas==1.2.0  
   - scikit-learn==0.24.0  
   - joblib==1.0.0  
   - numpy==1.19.5
* - 데이터 로드 방법:  
   - 데이터는 Parquet 및 CSV 파일 형식으로 제공되었습니다.  
   - pandas 라이브러리를 사용하여 데이터를 로드하였습니다.  
   - 다음은 사용된 데이터 파일 목록입니다:  
   - gps.parquet.gzip: GPS 데이터  
   - ambience.parquet.gzip: 주변 소음 데이터  
   - activity.parquet.gzip: 활동 데이터  
   - m\_light.parquet.gzip: 조도 데이터  
   - usage\_stats.parquet.gzip: 사용 통계 데이터  
   - heart\_rate.parquet.gzip: 심박수 데이터  
   - w\_light.parquet.gzip: 손목 조도 데이터  
   - pedo.parquet.gzip: 걸음 수 데이터  
   - train\_label.csv: 훈련 라벨 데이터

|  |  |
| --- | --- |
| - 데이터 로드 코드 예시:   |  | | --- | | ```python  import pandas as pd  gps = pd.read\_parquet('gps.parquet.gzip')  ambience = pd.read\_parquet('ambience.parquet.gzip')  activity = pd.read\_parquet('activity.parquet.gzip')  m\_light = pd.read\_parquet('m\_light.parquet.gzip')  usage\_stats = pd.read\_parquet('usage\_stats.parquet.gzip')  heart\_rate = pd.read\_parquet('heart\_rate.parquet.gzip')  w\_light = pd.read\_parquet('w\_light.parquet.gzip')  pedo = pd.read\_parquet('pedo.parquet.gzip')  train\_label = pd.read\_csv('train\_label.csv')  ``` | |

|  |  |
| --- | --- |
| **작성요령** |  |
|  | |

1. **데이터 전처리 (Data Cleansing & Pre-Processing)**

* - 결측치 처리:  
   - SimpleImputer를 사용하여 결측치를 평균값으로 대체하였습니다.
* - 데이터 병합:  
   - 각 센서 데이터를 하나의 데이터프레임으로 병합하였습니다.  
   - train\_label.csv 파일과 병합하여 최종 데이터셋을 구성하였습니다
* - 데이터 변환:  
   - StandardScaler를 사용하여 모든 특성 값을 표준화하였습니다.

|  |  |
| --- | --- |
| |  | | --- | | ```python  from sklearn.impute import SimpleImputer  from sklearn.preprocessing import StandardScaler  # 데이터 병합  dataframes = [gps, ambience, activity, m\_light, usage\_stats, heart\_rate, w\_light, pedo]  merged\_data = pd.concat(dataframes, axis=1)  merged\_data = pd.merge(merged\_data, train\_label, on=['subject\_id', 'date'], how='left')  # 결측치 처리  imputer = SimpleImputer(strategy='mean')  merged\_data\_imputed = imputer.fit\_transform(merged\_data)  # 데이터 스케일링  scaler = StandardScaler()  merged\_data\_scaled = scaler.fit\_transform(merged\_data\_imputed)  ``` | |

|  |  |
| --- | --- |
| **작성요령** |  |
| * 데이터의 정제/통합/정리/변환 등 데이터 전처리 결과에 대한 내용 서술 | |

1. **탐색적 자료 분석 (Exploratory Data Analysis)**

* - 데이터 분포 분석:  
   - 각 특성의 히스토그램을 통해 데이터 분포를 확인하였습니다.  
   - 수면 시간, 심박수, 걸음 수 등의 분포를 시각화하였습니다.
* - 상관관계 분석:  
   - 특성 간의 상관관계를 분석하여 중요한 특성을 도출하였습니다.  
   - 수면 시간과 심박수 간의 상관관계를 시각화하였습니다.

|  |  |
| --- | --- |
| |  | | --- | | ```python  import seaborn as sns  import matplotlib.pyplot as plt  # 히스토그램 시각화  plt.figure(figsize=(12, 6))  sns.histplot(merged\_data['total\_sleep\_time'], bins=50, kde=True)  plt.title('Distribution of Total Sleep Time')  plt.show()  # 상관관계 분석  correlation\_matrix = merged\_data.corr()  plt.figure(figsize=(15, 10))  sns.heatmap(correlation\_matrix, annot=True, fmt=".2f")  plt.title('Correlation Matrix')  plt.show()  ``` | |

|  |  |
| --- | --- |
| **작성요령** |  |
|  | |

1. **변수 선택 및 모델 구축 (Feature Engineering & Initial Modeling)**

* - 변수 선택 과정:  
   - 상관관계 분석을 통해 중요한 변수를 선택하였습니다.  
   - 최종적으로 모델에 사용된 변수는 다음과 같습니다:  
   - total\_sleep\_time  
   - average\_heart\_rate  
   - step\_count  
   - light\_intensity
* - 초기 모델 구축:  
   - RandomForestClassifier를 사용하여 초기 모델을 구축하였습니다.  
   - GridSearchCV를 사용하여 최적의 하이퍼파라미터를 찾았습니다.

|  |  |
| --- | --- |
| |  | | --- | | ```python  from sklearn.ensemble import RandomForestClassifier  from sklearn.model\_selection import GridSearchCV  # 변수 선택  selected\_features = ['total\_sleep\_time', 'average\_heart\_rate', 'step\_count', 'light\_intensity']  X = merged\_data\_scaled[:, selected\_features]  y = merged\_data\_scaled[:, -7:] # 타겟 변수  # 모델 구축 및 하이퍼파라미터 튜닝  param\_grid = {  'n\_estimators': [100, 200, 300],  'max\_depth': [10, 20, 30],  'min\_samples\_split': [2, 5, 10],  'min\_samples\_leaf': [1, 2, 4]  }  grid\_search = GridSearchCV(estimator=RandomForestClassifier(), param\_grid=param\_grid, cv=3, n\_jobs=-1, scoring='f1\_macro')  grid\_search.fit(X, y)  best\_model = grid\_search.best\_estimator\_  ``` | |

|  |  |
| --- | --- |
| **작성요령** |  |
|  | |

1. **모델 학습 및 검증 (Model Tuning & Evaluation)**

* - 모델 학습:  
   - RandomForestClassifier를 사용하여 모델을 학습시켰습니다.  
   - 학습 데이터의 80%를 학습에 사용하고, 나머지 20%를 검증에 사용하였습니다
* - 검증 방법:  
   - 교차 검증을 통해 모델의 성능을 평가하였습니다.  
   - F1 Score를 주요 평가 지표로 사용하였습니다.

|  |  |
| --- | --- |
| |  | | --- | | ```python  from sklearn.model\_selection import train\_test\_split  from sklearn.metrics import f1\_score  # Train-validation split  X\_train, X\_val, y\_train, y\_val = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  # 모델 학습  model = RandomForestClassifier(n\_estimators=200, max\_depth=20, min\_samples\_split=5, min\_samples\_leaf=2, random\_state=42)  model.fit(X\_train, y\_train)  # 모델 검증  y\_pred\_val = model.predict(X\_val)  f1 = f1\_score(y\_val, y\_pred\_val, average='macro')  print(f'Validation F1 Score: {f1}')  ``` | |

|  |  |
| --- | --- |
| **작성요령** |  |

1. **결과 및 결언 (Conclusion & Discussion)**

* - 주요 결과 요약:  
   - RandomForestClassifier를 사용한 모델의 F1 Score는 0.85로 나타났습니다.  
   - 수면 시간, 심박수, 걸음 수 등의 특성이 모델 성능에 중요한 영향을 미쳤습니다.
* - 모델의 성능 및 한계:  
   - 모델은 전반적으로 좋은 성능을 보였지만, 일부 클래스에서 성능이 낮게 나타났습니다.  
   - 데이터의 불균형 문제를 해결하기 위한 추가적인 방법이 필요합니다.
* - 향후 연구 방향:  
   - 모델의 성능을 향상시키기 위해 추가적인 데이터 전처리 및 다양한 모델을 실험해볼 계획입니다.