XGBoost

In [378]:

```
import xgboost as xgb
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split

X = df[numerical_columns]# 예측 변수 데이터
y = df['MAI']# 타겟 변수 데이터

# 데이터셋 분할
X_train, X_valid, y_train, y_valid = train_test_split(X, y, test_size=0.2, random_state=1)

# XGBoost 데이터셋 생성
dtrain = xgb.DMatrix(X_train, label=y_train)
dvalid = xgb.DMatrix(X_valid, label=y_valid)

# XGBoost 회귀 모델 설정
xgbr = xgb.XGBRegressor(objective='reg:squarederror', random_state=42)

# 모델 훈련
xgbr.fit(X_train, y_train)

# 테스트 세트로 예측
y_pred = xgbr.predict(X_test)

# 평가 메트릭 계산
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

print("Mean Squared Error:", mse)
print("Root Mean Squared Error:", rmse)
print("R-squared:", r2)
```

```
Mean Squared Error: 2346.1095248421466
Root Mean Squared Error: 48.43665476518942
R-squared: 0.9486661951726639
```

In [404]:

```
X = df[numerical_columns]# 예측 변수 데이터
y = df['MAI']# 타겟 변수 데이터

# 데이터셋 분할
X_train, X_valid, y_train, y_valid = train_test_split(X, y, test_size=0.2, random_state=1)

# XGBoost 데이터셋 생성
dtrain = xgb.DMatrix(X_train, label=y_train)
dvalid = xgb.DMatrix(X_valid, label=y_valid)

# XGBoost 회귀 모델 설정
xgbr = xgb.XGBRegressor(objective='reg:squarederror', random_state=42)

# 탐색할 매개변수 그리드 생성
param_grid = {
    'eta': [0.01, 0.1 , 0.5],
    'subsample': [0.3, 0.5, 0.8],
    'n_estimators': [1000, 1500, 2000]
}

# GridSearchCV를 사용하여 교차 검증 및 매개변수 그리드 탐색
grid_search = GridSearchCV(estimator=xgbr, param_grid=param_grid, cv=5, scoring='neg_mean_squared_error')
grid_search.fit(X_train, y_train)

# 최적의 매개변수 조합 출력
print("Best parameters:", grid_search.best_params_)

# 최적의 모델로 예측 수행
best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_valid)

# 예측 결과 평가 (RMSE 계산)
rmse = mean_squared_error(y_valid, y_pred, squared=False)
print("Root Mean Squared Error (RMSE):", rmse)
```

```
Best parameters: {'eta': 0.01, 'n_estimators': 2000, 'subsample': 0.8}
Root Mean Squared Error (RMSE): 97.52622618628264
```

In [416]:

```
X = df[numerical_columns]# 예측 변수 데이터
y = df['MAI']# 타겟 변수 데이터

X = df[numerical_columns]# 예측 변수 데이터
y = df['MAI']# 타겟 변수 데이터

# 데이터셋 분할
X_train, X_valid, y_train, y_valid = train_test_split(X, y, test_size=0.2, random_state=1)

# XGBoost 데이터셋 생성
dtrain = xgb.DMatrix(X_train, label=y_train)
dvalid = xgb.DMatrix(X_valid, label=y_valid)

# XGBoost 회귀 모델 설정
xgbr = xgb.XGBRegressor(objective='reg:squarederror', random_state=42,
                        eta = 0.01,
                        n_estimators = 2000,
                        subsample = 0.8,
                        colsample_bytree = 0.8)

# 모델 훈련
xgbr.fit(X_train, y_train)

# 테스트 세트로 예측
y_pred = xgbr.predict(X_test)

# training set
pred_train = xgbr.predict(X_train)

### R square
print('train R square:', xgbr.score(X_train, y_train))  # training set

# 평가 메트릭 계산
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

print("Mean Squared Error:", mse)
print("Root Mean Squared Error:", rmse)
print("R-squared:", r2)
```

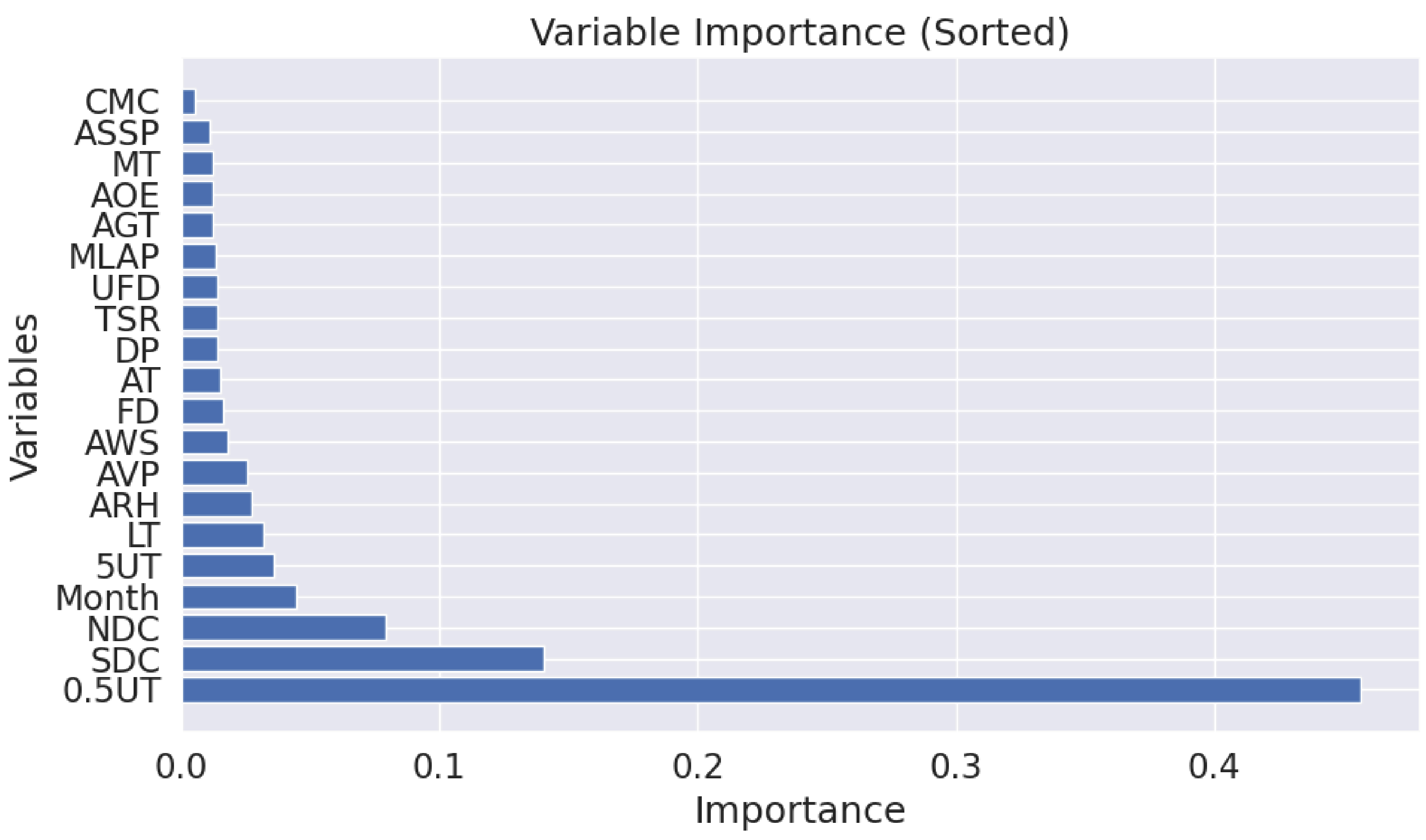
```
train R square: 0.9985410922282981
Mean Squared Error: 2261.9178567887393
Root Mean Squared Error: 47.559624228842885
R-squared: 0.9505083421867647
```

In [502]:
```python
# 변수 중요도 추출
importances = xgbr.feature_importances_


# 변수 중요도 출력
for features, importance in zip(X.columns, importances):
    print(f"{features}: {importance}")


# 변수 중요도를 크기 순으로 나열한 그래프 그리기
plt.figure(figsize=(10, 6))
plt.barh(df_importance_sorted['Variable'], df_importance_sorted['Importance'], align='center')
plt.xlabel('Importance')
plt.ylabel('Variables')
plt.title('Variable Importance (Sorted)')
plt.tight_layout()
plt.show()
```

```
AT: 0.008874556049704552
AWS: 0.02077140286564827
AOE: 0.013655860908329487
AGT: 0.01338735781610012
DP: 0.013184743002057076
ARH: 0.013143985532224178
AVP: 0.07718594372272491
MLAP: 0.23050560057163239
TSR: 0.10461051017045975
FD: 0.026102451607584953
UFD: 0.02857055887579918
NDC: 0.02412785217165947
CMC: 0.018557654693722725
SDC: 0.014801444485783577
```


Variable Importance (Sorted)
CMC
ASSP
MT
AOE
AGT
MLAP
UFD
TSR
DP
AT
FD
AWS
AVP
ARH
LT
5UT
Month
NDC
SDC
0.5UT
Variables
Importance