[TIL] 24/12/04_ICU 환자의 항생제 사용과 치료 성공률 분석 + 다제내성균(MDR) 감염 치료 효과 분석

후보 주제

▼ ICU 환자의 항생제 사용과 치료 성공률 분석

- 주제 설명
  - ICU에서 항생제를 사용한 환자들의 치료 성공 여부(퇴원 여부 또는 생존 여부)를 예측
    항생제 처방 패턴과 환자의 생리적 상태(LABEVENTS 데이터)를 기반으로 분석

  - 활용 테이블
    PRESCRIPTIONS: 항생제 이름, 용량, 투여 기간.
    LABEVENTS: 감염 관련 실험실 검사 값(예: WBC, CRP, 프로칼시토닌).
    ICUSTAYS: 환자의 체류 기간, 사망 여부 등 상태 정보.

  - 모델링 방향:
    입력: 항생제 처방 데이터 + 실험실 검사 결과.
    출력: 생존 여부(0=사망, 1=생존) 또는 퇴원 여부.
    알고리즘: 로지스틱 회귀, 랜덤 포레스트, XGBoost

▼ 중환자실에서의 다제내성균(MDR) 감염 치료 효과 분석

- 주제 설명: 다제내성균 감염 환자의 항생제 사용 실태와 치료 성공률 평가
  특정 항생제의 사용이 치료 성공률에 미치는 영향을 분석

- 활용 테이블
  - DIAGNOSES_ICD: MDR 감염 관련 진단 코드
  - PRESCRIPTIONS: 항생제 처방 데이터
  - LABEVENTS: 감염 지표(예: CRP, WBC) 변화

- 모델링 방향
  - 입력: 항생제 사용량 + 실험실 검사 데이터
  - 출력: 감염 치료 성공 여부(0=실패, 1=성공)
  - 알고리즘: 로지스틱 회귀, XGBoost

- 위의 주제로 진행하기 위해 필요한 전제 조건
  - 항생제가 1개일 경우, 진행 불가 → 감염지표, 항생제의 종류, 약물 사용 빈도수까지 파악하여 분석이 가능할지 여부에 대해서 파악해야 함.

- 항생제 리스트업(30개의 키워드(공식문서 발췌)를 통해 필터링 진행)
- 항생제 관련 단어 141개

```python
antibiotics = [
    'GENTAMICIN', 'OXACILLIN', 'ERYTHROMYCIN', 'PENICILLIN',
      'LEVOFLOXACIN', 'NITROFURANTOIN', 'PIPERACILLIN/TAZO', 'MEROPENEM',
      'CEFTAZIDIME', 'CEFAZOLIN', 'CEFEPIME', 'TRIMETHOPRIM/SULFA',
      'TOBRAMYCIN', 'IMIPENEM', 'CEFTRIAXONE', 'CIPROFLOXACIN',
      'VANCOMYCIN', 'CLINDAMYCIN', 'TETRACYCLINE', 'RIFAMPIN',
      'CHLORAMPHENICOL', 'AMPICILLIN', 'LINEZOLID', 'PIPERACILLIN',
      'AMPICILLIN/SULBACTAM', 'CEFUROXIME', 'PENICILLIN G', 'DAPTOMYCIN',
      'AMIKACIN', 'CEFPODOXIME'
]

pattern = '|'.join(antibiotics)

antibiotics_prescriptions = data[data['DRUG'].str.contains(pattern, case=False, na=False)]
```

```python
[269] print('약물 고유 개수 : ', len(antibiotics_prescriptions['DRUG'].unique()))
      print('해당 약물 사용 데이터 개수 : ', len(antibiotics_prescriptions['DRUG']))
```

```
약물 고유 개수 :  141
해당 약물 사용 데이터 개수 :  193795
```

성능 지표 개선 방안

- 기존 샘플 데이터 → 전체 데이터로 성능 재확인
- 튜닝 값 변경을 통해 향상
- 기존 이진 분류 → 다중 분류로 변경하여 확인
- target 값 조정
- 항생제 관련(동균님, 지현님 진행)

  ▼ 약물 대조 과정 진행

  https://github.com/tanlab/MIMIC-III-Clinical-Drug-Representations/blob/main/01-MIMIC-III-Drugs-Names-To-Pubchem-ID.ipynb

  https://github.com/ssshddd/DrugRec

V1 : antibiotics_keywords 수정 이전

- 기존 샘플 데이터 → 전체 데이터로 성능 재확인

  💡 기준 정리
  - 기존 : MeMl_Analysis_master.ipynb
  - 현재 : [updating] MeMl_Analysis_master.ipynb

  ◦ 기존
  - 랜덤 포레스트
    - ROC-AUC Score: 0.6710526315789475
  - XGBoost
    - ROC-AUC Score: 0.5986842105263158

  ◦ 전체 데이터 적용
  - 랜덤 포레스트
    - ROC-AUC Score: 0.7503478004481398
  - XGBoost
    - ROC-AUC Score: 0.7519268245266666

- Random Forest

```
rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(X_train, y_train)

y_pred_rf = rf_model.predict(X_test)
y_pred_rf_proba = rf_model.predict_proba(X_test)[:, 1]

print("Classification Report:\n", classification_report(y_test, y_pred_rf))
print("ROC-AUC Score:", roc_auc_score(y_test, y_pred_rf_proba))
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.99      0.93      7718
           1       0.61      0.10      0.18      1161

    accuracy                           0.87      8879
   macro avg       0.75      0.55      0.55      8879
weighted avg       0.85      0.87      0.83      8879

ROC-AUC Score: 0.7503478004481398
```

```
[57]  xgb_model = XGBClassifier(use_label_encoder=False, eval_metric='logloss', rand
      xgb_model.fit(X_train, y_train)

      y_pred_xgb = xgb_model.predict(X_test)
      y_pred_xgb_proba = xgb_model.predict_proba(X_test)[:, 1]

      print("Classification Report:\n", classification_report(y_test, y_pred_xgb))
      print("ROC-AUC Score:", roc_auc_score(y_test, y_pred_xgb_proba))
```

```
/usr/local/lib/python3.10/dist-packages/xgboost/core.py:158: UserWarning: [02:35:
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.97      0.93      7718
           1       0.49      0.17      0.25      1161

    accuracy                           0.87      8879
   macro avg       0.69      0.57      0.59      8879
weighted avg       0.83      0.87      0.84      8879

ROC-AUC Score: 0.7519268245266666
```

    - 원인 분석
        - 차이점 정리
            - 기존 노트북 : MIMIC III 데모 데이터, 전체 데이터테이블 로드
            - 현재 노트북 : MIMIC III 원본 데이터, 사용하는 4개의 데이터테이블 로드
        - ~~ROC 커브 확인~~
            - ~~ROC-AUC Score: 0.7503478004481398~~
            - ~~ROC-AUC Score:~~

- 튜닝 값 변경을 통해 향상
    - 랜덤 포레스트
        - 기존값으로 1차 튜닝 진행(소요 시간 : 2시간 12분 30초)

```
# Define hyperparameters to tune
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'bootstrap': [True, False]
}
```

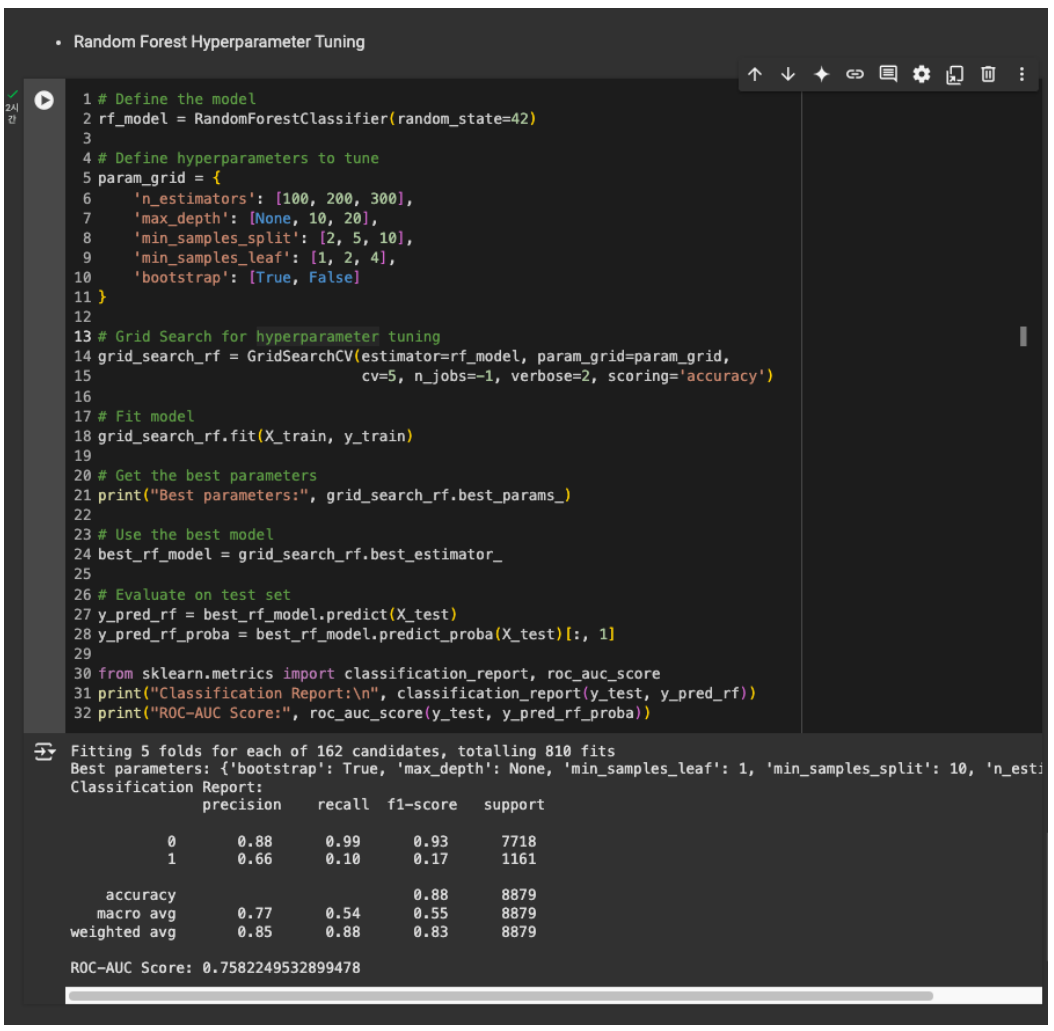
        - 결과
            - ROC-AUC Score: 0.7582249532899478(0.007증가)

```
Fitting 5 folds for each of 162 candidates, totalling 810 fits
Best parameters: {'bootstrap': True, 'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 10, 'n_esti
Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.99      0.93      7718
           1       0.66      0.10      0.17      1161
```

```
    accuracy                           0.88      8879
   macro avg       0.77      0.54      0.55      8879
weighted avg       0.85      0.88      0.83      8879

ROC-AUC Score: 0.7582249532899478
```



- XGBoost
  - 기존값으로 1차 튜닝 진행(소요 시간 : 30분 30초)

```
# Define hyperparameters to tune
param_grid_xgb = {
    'n_estimators': [100, 200, 300],
    'max_depth': [3, 6, 10],
    'learning_rate': [0.01, 0.05, 0.1],
    'subsample': [0.7, 0.8, 1.0],
    'colsample_bytree': [0.7, 0.8, 1.0]
}
```

  - 결과
    - ROC-AUC Score: 0.7740517429751899

```
Fitting 5 folds for each of 243 candidates, totalling 1215 fits
/usr/local/lib/python3.10/dist-packages/xgboost/core.py:158: UserWarning: [11:47:32] WARNING: /workspace/src/le
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
Best parameters: {'colsample_bytree': 0.7, 'learning_rate': 0.1, 'max_depth': 3, 'n_estimators': 200, 'subsampl
Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.99      0.93      7718
           1       0.59      0.11      0.19      1161

    accuracy                           0.87      8879
   macro avg       0.74      0.55      0.56      8879
weighted avg       0.84      0.87      0.83      8879

ROC-AUC Score: 0.7740517429751899
```

```python
# Define the model
xgb_model = XGBClassifier(use_label_encoder=False, eval_metric='logloss', random_state=42)

# Define hyperparameters to tune
param_grid_xgb = {
    'n_estimators': [100, 200, 300],
    'max_depth': [3, 6, 10],
    'learning_rate': [0.01, 0.05, 0.1],
    'subsample': [0.7, 0.8, 1.0],
    'colsample_bytree': [0.7, 0.8, 1.0]
}

# Grid Search for hyperparameter tuning
grid_search_xgb = GridSearchCV(estimator=xgb_model, param_grid=param_grid_xgb,
                               cv=5, n_jobs=-1, verbose=2, scoring='accuracy')

# Fit model
grid_search_xgb.fit(X_train, y_train)

# Get the best parameters
print("Best parameters:", grid_search_xgb.best_params_)

# Use the best model
best_xgb_model = grid_search_xgb.best_estimator_

# Evaluate on test set
y_pred_xgb = best_xgb_model.predict(X_test)
y_pred_xgb_proba = best_xgb_model.predict_proba(X_test)[:, 1]

print("Classification Report:\n", classification_report(y_test, y_pred_xgb))
print("ROC-AUC Score:", roc_auc_score(y_test, y_pred_xgb_proba))
```

```
Fitting 5 folds for each of 243 candidates, totalling 1215 fits
/usr/local/lib/python3.10/dist-packages/xgboost/core.py:158: UserWarning: [11:47:32] WARNING: /workspace/src/learner.cc:740:
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
Best parameters: {'colsample_bytree': 0.7, 'learning_rate': 0.1, 'max_depth': 3, 'n_estimators': 200, 'subsample': 0.8}
Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.99      0.93      7718
           1       0.59      0.11      0.19      1161

    accuracy                           0.87      8879
   macro avg       0.74      0.55      0.56      8879
weighted avg       0.84      0.87      0.83      8879

ROC-AUC Score: 0.7740517429751899
```

V2 : 1차 키워드 변형

▼ 변경점

- 기존 코드

```
# v1
# Filter prescriptions for antibiotics
antibiotics_keywords = ['cef', 'penicillin', 'amoxicillin', 'tetracycline', 'vancomycin', 'meropenem']
antibiotics = prescriptions[prescriptions['drug'].str.contains('|'.join(antibiotics_keywords), case=False, na=Fals

# Keep relevant columns
antibiotics = antibiotics[['subject_id', 'hadm_id', 'startdate', 'enddate', 'drug', 'dose_val_rx']]
```

- 변경 코드

```
# Filter prescriptions for antibiotics
antibiotics_keywords = ['GENTAMICIN', 'OXACILLIN', 'ERYTHROMYCIN', 'PENICILLIN',
       'LEVOFLOXACIN', 'NITROFURANTOIN', 'PIPERACILLIN/TAZO', 'MEROPENEM',
       'CEFTAZIDIME', 'CEFAZOLIN', 'CEFEPIME', 'TRIMETHOPRIM/SULFA',
       'TOBRAMYCIN', 'IMIPENEM', 'CEFTRIAXONE', 'CIPROFLOXACIN',
       'VANCOMYCIN', 'CLINDAMYCIN', 'TETRACYCLINE', 'RIFAMPIN',
       'CHLORAMPHENICOL', 'AMPICILLIN', 'LINEZOLID', 'PIPERACILLIN',
       'AMPICILLIN/SULBACTAM', 'CEFUROXIME', 'PENICILLIN G', 'DAPTOMYCIN',
       'AMIKACIN', 'CEFPODOXIME'
]


antibiotics = prescriptions[prescriptions['drug'].str.contains('|'.join(antibiotics_keywords), case=False, na=False)]

# Keep relevant columns
antibiotics = antibiotics[['subject_id', 'hadm_id', 'startdate', 'enddate', 'drug', 'dose_val_rx']]
```

- 1차 모델링
  - 랜덤 포레스트
    - ROC-AUC Score: 0.7436463242399466
  - XGBoost
    - ROC-AUC Score: 0.7470484650263881

- Random Forest

```
rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(X_train, y_train)

y_pred_rf = rf_model.predict(X_test)
y_pred_rf_proba = rf_model.predict_proba(X_test)[:, 1]

print("Classification Report:\n", classification_report(y_test, y_pred_rf))
print("ROC-AUC Score:", roc_auc_score(y_test, y_pred_rf_proba))
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.99      0.94      9784
           1       0.61      0.07      0.13      1375

    accuracy                           0.88     11159
   macro avg       0.75      0.53      0.53     11159
weighted avg       0.85      0.88      0.84     11159

ROC-AUC Score: 0.7436463242399466
```

```
xgb_model = XGBClassifier(use_label_encoder=False, eval_metric='logloss', random_state=42)
xgb_model.fit(X_train, y_train)

y_pred_xgb = xgb_model.predict(X_test)
y_pred_xgb_proba = xgb_model.predict_proba(X_test)[:, 1]

print("Classification Report:\n", classification_report(y_test, y_pred_xgb))
print("ROC-AUC Score:", roc_auc_score(y_test, y_pred_xgb_proba))
```

```
/usr/local/lib/python3.10/dist-packages/xgboost/core.py:158: UserWarning: [05:09:52] WARNING: /workspa
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.98      0.93      9784
           1       0.50      0.13      0.20      1375

    accuracy                           0.88     11159
   macro avg       0.69      0.56      0.57     11159
weighted avg       0.84      0.88      0.84     11159

ROC-AUC Score: 0.7470484650263881
```

- 튜닝 값 변경을 통해 향상
  - 랜덤 포레스트
    - ROC-AUC Score: 0.7510563814762506

```
Fitting 5 folds for each of 162 candidates, totalling 810 fits
Best parameters: {'bootstrap': True, 'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 5, 'n_estim
Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.99      0.94      9784
           1       0.62      0.07      0.13      1375

    accuracy                           0.88     11159
   macro avg       0.75      0.53      0.53     11159
weighted avg       0.85      0.88      0.84     11159

ROC-AUC Score: 0.7510563814762506
```

```
# Define the model
rf_model = RandomForestClassifier(random_state=42)

# Define hyperparameters to tune
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'bootstrap': [True, False]
}

# Grid Search for hyperparameter tuning
grid_search_rf = GridSearchCV(estimator=rf_model, param_grid=param_grid,
                              cv=5, n_jobs=-1, verbose=2, scoring='accuracy')

# Fit model
grid_search_rf.fit(X_train, y_train)

# Get the best parameters
print("Best parameters:", grid_search_rf.best_params_)

# Use the best model
best_rf_model = grid_search_rf.best_estimator_

# Evaluate on test set
y_pred_rf = best_rf_model.predict(X_test)
y_pred_rf_proba = best_rf_model.predict_proba(X_test)[:, 1]

from sklearn.metrics import classification_report, roc_auc_score
print("Classification Report:₩n", classification_report(y_test, y_pred_rf))
print("ROC-AUC Score:", roc_auc_score(y_test, y_pred_rf_proba))
```

```
Fitting 5 folds for each of 162 candidates, totalling 810 fits
Best parameters: {'bootstrap': True, 'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 5, 'n_estimators': 300}
Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.99      0.94      9784
           1       0.62      0.07      0.13      1375

    accuracy                           0.88     11159
   macro avg       0.75      0.53      0.53     11159
weighted avg       0.85      0.88      0.84     11159

ROC-AUC Score: 0.7510563814762506
```

- XGBoost
  - ROC-AUC Score: 0.7640216680294359

```
Fitting 5 folds for each of 243 candidates, totalling 1215 fits
/usr/local/lib/python3.10/dist-packages/xgboost/core.py:158: UserWarning: [05:45:36] WARNING: /workspace/src/le
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
Best parameters: {'colsample_bytree': 1.0, 'learning_rate': 0.01, 'max_depth': 6, 'n_estimators': 300, 'subsamp
Classification Report:
              precision    recall  f1-score   support

           0       0.88      1.00      0.94      9784
           1       0.63      0.06      0.10      1375

    accuracy                           0.88     11159
   macro avg       0.76      0.53      0.52     11159
weighted avg       0.85      0.88      0.83     11159

ROC-AUC Score: 0.7640216680294359
```

- XGBoost Hyperparameter Tuning

```python
# Define the model
xgb_model = XGBClassifier(use_label_encoder=False, eval_metric='logloss', random_state=42)

# Define hyperparameters to tune
param_grid_xgb = {
    'n_estimators': [100, 200, 300],
    'max_depth': [3, 6, 10],
    'learning_rate': [0.01, 0.05, 0.1],
    'subsample': [0.7, 0.8, 1.0],
    'colsample_bytree': [0.7, 0.8, 1.0]
}

# Grid Search for hyperparameter tuning
grid_search_xgb = GridSearchCV(estimator=xgb_model, param_grid=param_grid_xgb,
                               cv=5, n_jobs=-1, verbose=2, scoring='accuracy')

# Fit model
grid_search_xgb.fit(X_train, y_train)

# Get the best parameters
print("Best parameters:", grid_search_xgb.best_params_)

# Use the best model
best_xgb_model = grid_search_xgb.best_estimator_

# Evaluate on test set
y_pred_xgb = best_xgb_model.predict(X_test)
y_pred_xgb_proba = best_xgb_model.predict_proba(X_test)[:, 1]

print("Classification Report:\n", classification_report(y_test, y_pred_xgb))
print("ROC-AUC Score:", roc_auc_score(y_test, y_pred_xgb_proba))
```

```
Fitting 5 folds for each of 243 candidates, totalling 1215 fits
/usr/local/lib/python3.10/dist-packages/xgboost/core.py:158: UserWarning: [05:45:36] WARNING: /workspace/src/le
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
Best parameters: {'colsample_bytree': 1.0, 'learning_rate': 0.01, 'max_depth': 6, 'n_estimators': 300, 'subsamp
Classification Report:
              precision    recall  f1-score   support

           0       0.88      1.00      0.94      9784
           1       0.63      0.06      0.10      1375

    accuracy                           0.88     11159
   macro avg       0.76      0.53      0.52     11159
weighted avg       0.85      0.88      0.83     11159

ROC-AUC Score: 0.7640216680294359
```

문제점

- 항생제 관련
  - 항생제 리스트에 따르면, unique 값이 30이 나와야 함.
  - 하위 분류가 더 있는 것으로 추측 → 필터링 진행

```python
# List of common antibiotics
antibiotics = [
    'GENTAMICIN', 'OXACILLIN', 'ERYTHROMYCIN', 'PENICILLIN',
       'LEVOFLOXACIN', 'NITROFURANTOIN', 'PIPERACILLIN/TAZO', 'MEROPENEM',
       'CEFTAZIDIME', 'CEFAZOLIN', 'CEFEPIME', 'TRIMETHOPRIM/SULFA',
       'TOBRAMYCIN', 'IMIPENEM', 'CEFTRIAXONE', 'CIPROFLOXACIN',
       'VANCOMYCIN', 'CLINDAMYCIN', 'TETRACYCLINE', 'RIFAMPIN',
       'CHLORAMPHENICOL', 'AMPICILLIN', 'LINEZOLID', 'PIPERACILLIN',
       'AMPICILLIN/SULBACTAM', 'CEFUROXIME', 'PENICILLIN G', 'DAPTOMYCIN',
       'AMIKACIN', 'CEFPODOXIME'
]

# Combine all antibiotics into a regex pattern
pattern = '|'.join(antibiotics)

# Filter the prescriptions for antibiotics using a case-insensitive search
antibiotics_prescriptions = data[data['DRUG'].str.contains(pattern, case=False, na=False)]
```

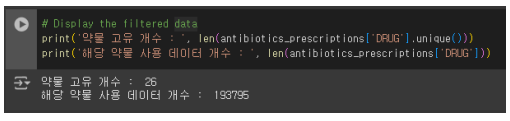
```
약물 고유 개수 :  141
해당 약물 사용 데이터 개수 :  193795
```

- 같은 성분을 쓰나 대상, 목적에 맞게 다른 이름으로 구성 되어 있음
- 같은 성분을 사용하는 약물끼리 통합

```
def data_rename(df, renames):
    renames_df = df[df['DRUG'].str.contains(renames, case=False, na=False)]
    unique_list = renames_df['DRUG'].unique()
    df['DRUG'] = df['DRUG'].replace(unique_list, renames)

    return df
```

```
for name in antibiotics:
    antibiotics_prescriptions = data_rename(antibiotics_prescriptions, name)항생제 141개에서 26개로 통합
```



- 전체 데이터
  - ANTIBIOTICS.csv, INFECTION_TEST.csv 파일 없음
  - MICROBIOLOGYEVENTS 테이블에서 interpretation을 통해 인사이트 도출

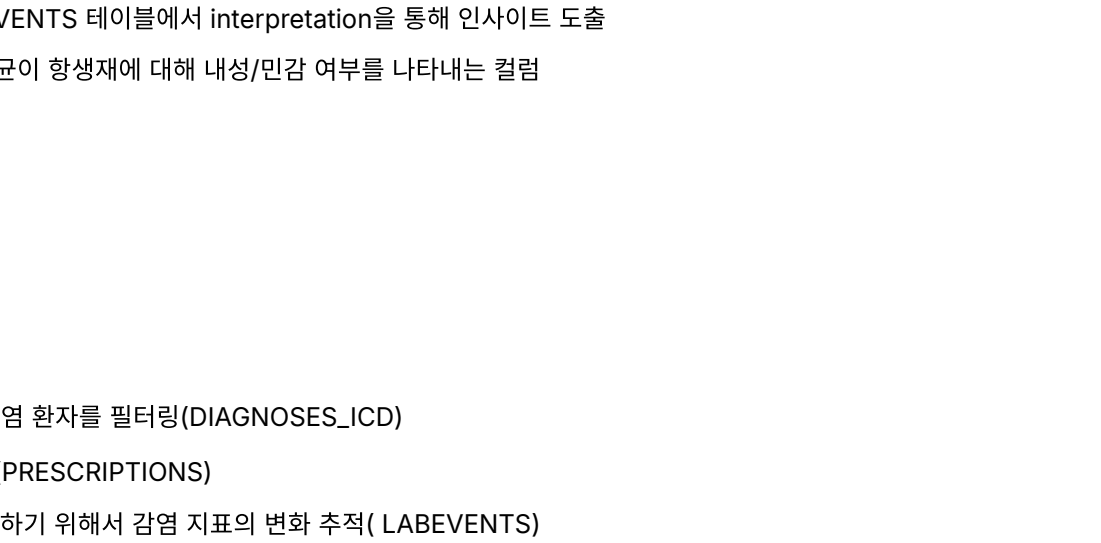
    *interpretation: 세균이 항생재에 대해 내성/민감 여부를 나타내는 컬럼

V3 : 지표 정리

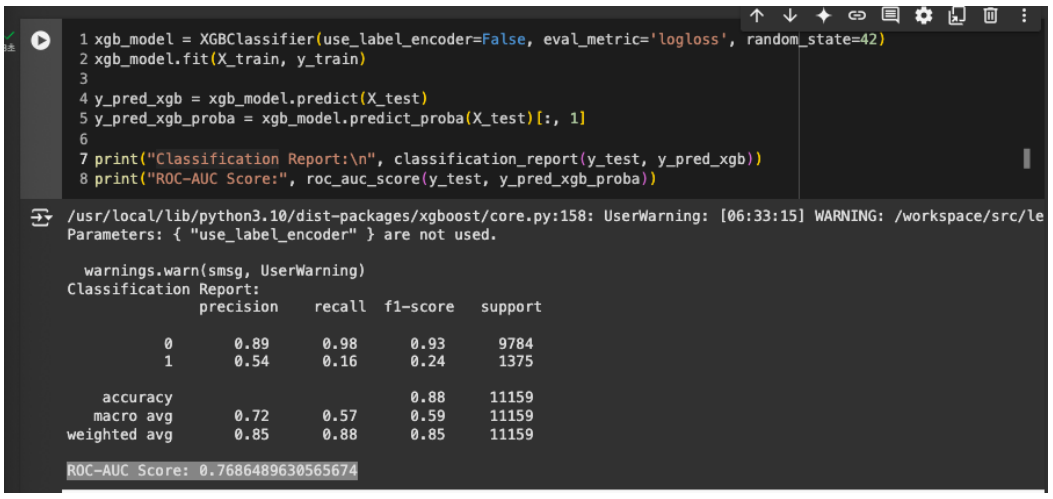
- 항생제
  - ICD-9 진단 코드 기반으로 감염 환자를 필터링(DIAGNOSES_ICD)
  - 환자에게 처방된 항생제 추출(PRESCRIPTIONS)
  - 치료 전후 환자의 상태를 확인하기 위해서 감염 지표의 변화 추적(LABEVENTS)
    - 치료 성공 여부 확인 : LABEVENTS 기준 지표 정상화 →생존, 지표 악화 → 사망

- 노트북
  - [updating-keyword2차필터링] MeMI_Analysis_master.ipynb

- 랜덤포레스트
  - ROC-AUC Score: 0.7590781981714116



- XGBoost
  - ROC-AUC Score: 0.7686489630565674



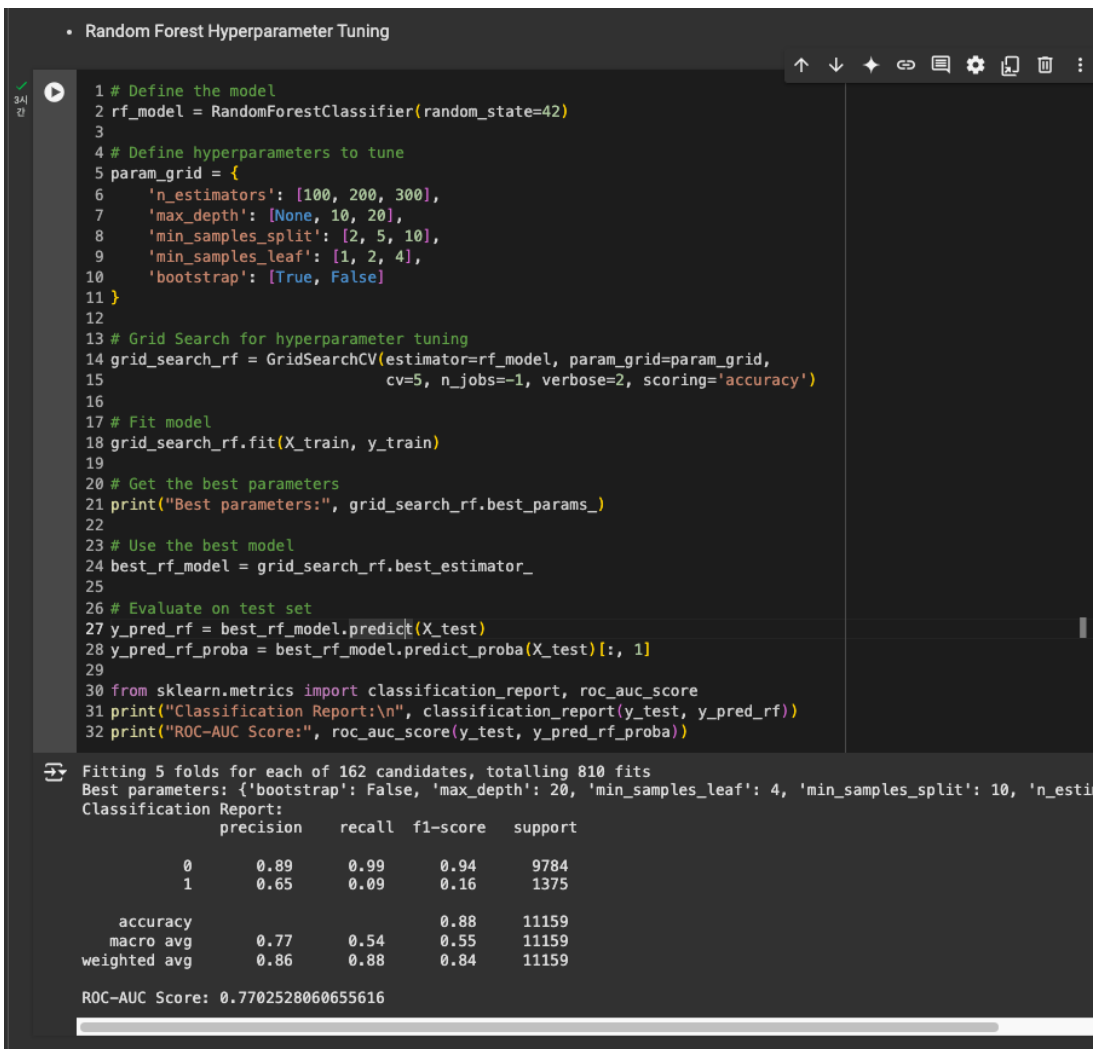
- 랜덤 포레스트 튜닝

```
Fitting 5 folds for each of 162 candidates, totalling 810 fits
Best parameters: {'bootstrap': False, 'max_depth': 20, 'min_samples_leaf': 4, 'min_samples_split': 10, 'n_estimators'
Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.99      0.94      9784
           1       0.65      0.09      0.16      1375

    accuracy                           0.88     11159
   macro avg       0.77      0.54      0.55     11159
weighted avg       0.86      0.88      0.84     11159

ROC-AUC Score: 0.7702528060655616
```



- XGBoost 튜닝

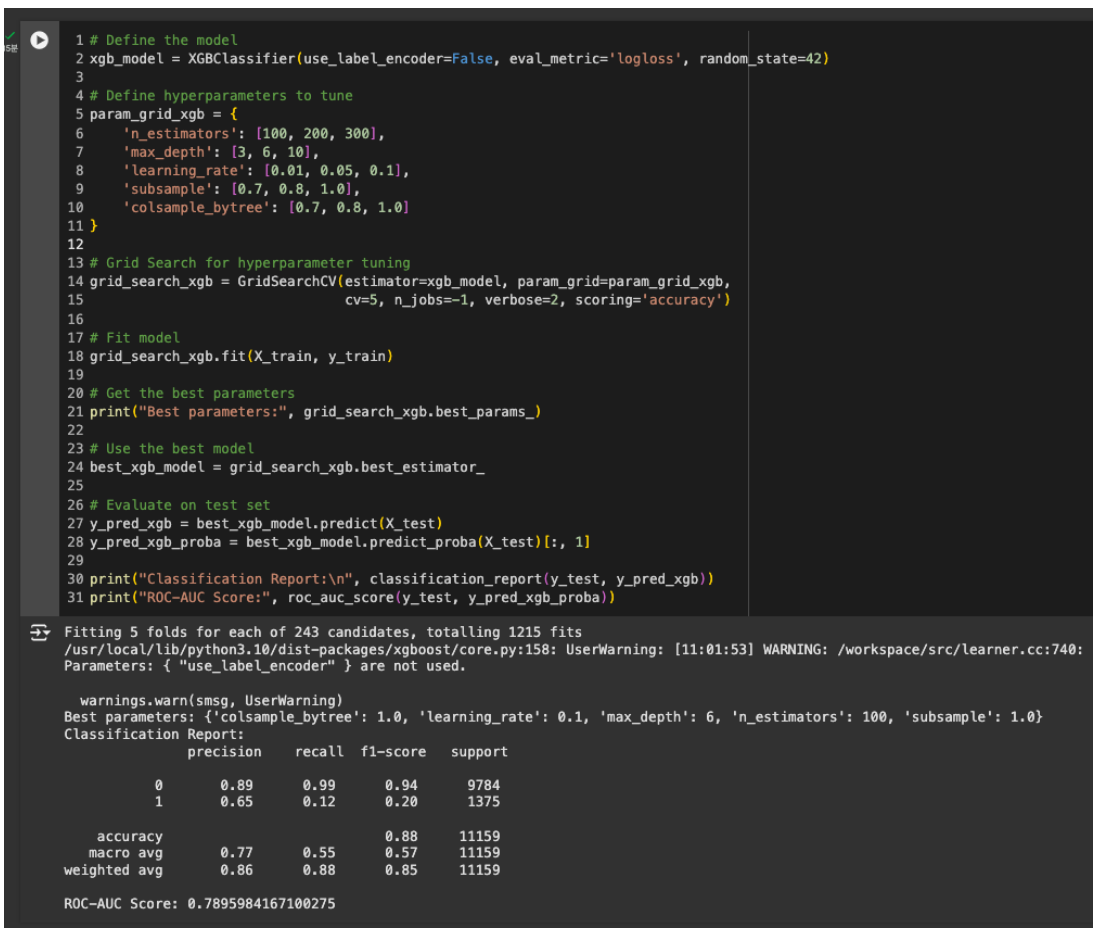
```
Fitting 5 folds for each of 243 candidates, totalling 1215 fits
/usr/local/lib/python3.10/dist-packages/xgboost/core.py:158: UserWarning: [11:01:53] WARNING: /workspace/src/learner.
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
Best parameters: {'colsample_bytree': 1.0, 'learning_rate': 0.1, 'max_depth': 6, 'n_estimators': 100, 'subsample': 1.
Classification Report:
              precision    recall  f1-score   support
```

```
              0       0.89      0.99      0.94      9784
              1       0.65      0.12      0.20      1375

       accuracy                           0.88     11159
      macro avg       0.77      0.55      0.57     11159
   weighted avg       0.86      0.88      0.85     11159

ROC-AUC Score: 0.7895984167100275
```



```python
# Define the model
xgb_model = XGBClassifier(use_label_encoder=False, eval_metric='logloss', random_state=42)

# Define hyperparameters to tune
param_grid_xgb = {
    'n_estimators': [100, 200, 300],
    'max_depth': [3, 6, 10],
    'learning_rate': [0.01, 0.05, 0.1],
    'subsample': [0.7, 0.8, 1.0],
    'colsample_bytree': [0.7, 0.8, 1.0]
}

# Grid Search for hyperparameter tuning
grid_search_xgb = GridSearchCV(estimator=xgb_model, param_grid=param_grid_xgb,
                               cv=5, n_jobs=-1, verbose=2, scoring='accuracy')

# Fit model
grid_search_xgb.fit(X_train, y_train)

# Get the best parameters
print("Best parameters:", grid_search_xgb.best_params_)

# Use the best model
best_xgb_model = grid_search_xgb.best_estimator_

# Evaluate on test set
y_pred_xgb = best_xgb_model.predict(X_test)
y_pred_xgb_proba = best_xgb_model.predict_proba(X_test)[:, 1]

print("Classification Report:\n", classification_report(y_test, y_pred_xgb))
print("ROC-AUC Score:", roc_auc_score(y_test, y_pred_xgb_proba))
```

```
Fitting 5 folds for each of 243 candidates, totalling 1215 fits
/usr/local/lib/python3.10/dist-packages/xgboost/core.py:158: UserWarning: [11:01:53] WARNING: /workspace/src/learner.cc:740:
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
Best parameters: {'colsample_bytree': 1.0, 'learning_rate': 0.1, 'max_depth': 6, 'n_estimators': 100, 'subsample': 1.0}
Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.99      0.94      9784
           1       0.65      0.12      0.20      1375

    accuracy                           0.88     11159
   macro avg       0.77      0.55      0.57     11159
weighted avg       0.86      0.88      0.85     11159

ROC-AUC Score: 0.7895984167100275
```

ROC-AUC Score 정리

- 랜덤포레스트
  - 원본
    - 튜닝 이전 : 0.7503478004481398
    - 튜닝 이후 : 0.7582249532899478
  - 1차 필터링
    - 튜닝 이전 : 0.7436463242399466
    - 튜닝 이후 : 0.7510563814762506
  - 2차 필터링
    - 튜닝 이전 : 0.7590781981714116
    - 튜닝 이후 : 0.7702528060655616

- XGboost
  - 원본
    - 튜닝 이전 : 0.7519268245266666
    - 튜닝 이후 : 0.7740517429751899
  - 1차 필터링
    - 튜닝 이전 : 0. 7470484650263881
    - 튜닝 이후 : 0.7640216680294359
  - 2차 필터링
    - 튜닝 이전 : 0.7686489630565674
    - 튜닝 이후 : 0.7895984167100275