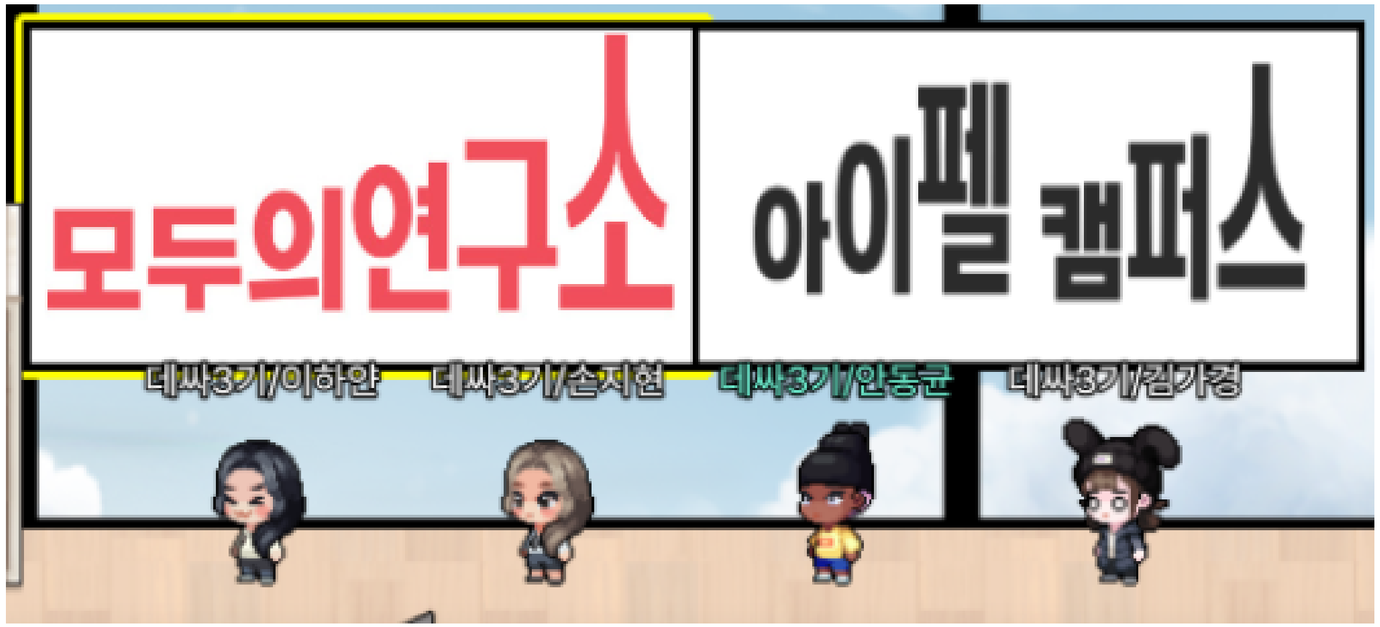
ICU 내 항생제 치료 전략 및 다제내성균(MDR) 감염의 치료 성공률 분석

TEAM MeMI(Medi+MIMIC)

목차



Data Set

데이터 정보

Death Pattern

MIMIC III Dataset

- MIMIC-III 데이터셋은 ICU에 입원한 4만 명 이상 환자들의 임상 기록, 생리학적 측정, 처방 정보, 검사 결과, 약물 처방 기록 등 다양한 의료 정보를 포함함
- 데이터 수집 기간 : 2001년 ~ 2012년
- 25개의 CSV 파일 존재

```
ADMISSIONS.csv
CALLOUT.csv
CAREGIVERS.csv
CHARTEVENTS.csv
CPTEVENTS.csv
DATETIMEEVENTS.csv
DIAGNOSES_ICD.csv
DRGCODES.csv
D_CPT.csv
D_ICD_DIAGNOSES.csv
D_ICD_PROCEDURES.csv
D_ITEMS.csv
D_LABITEMS.csv
ICUSTAYS.csv
INPUTEVENTS_CV.csv
INPUTEVENTS_MV.csv
LABEVENTS.csv
LICENSE.txt
MICROBIOLOGYEVENTS.csv
NOTEEVENTS.csv
OUTPUTEVENTS.csv
PATIENTS.csv
PRESCRIPTIONS.csv
PROCEDUREEVENTS_MV.csv
PROCEDURES_ICD.csv
README.md
SERVICES.csv
SHA256SUMS.txt
TRANSFERS.csv
Untitled.ipynb
checksum_md5_unzipped.txt
checksum_md5_zipped.txt
```

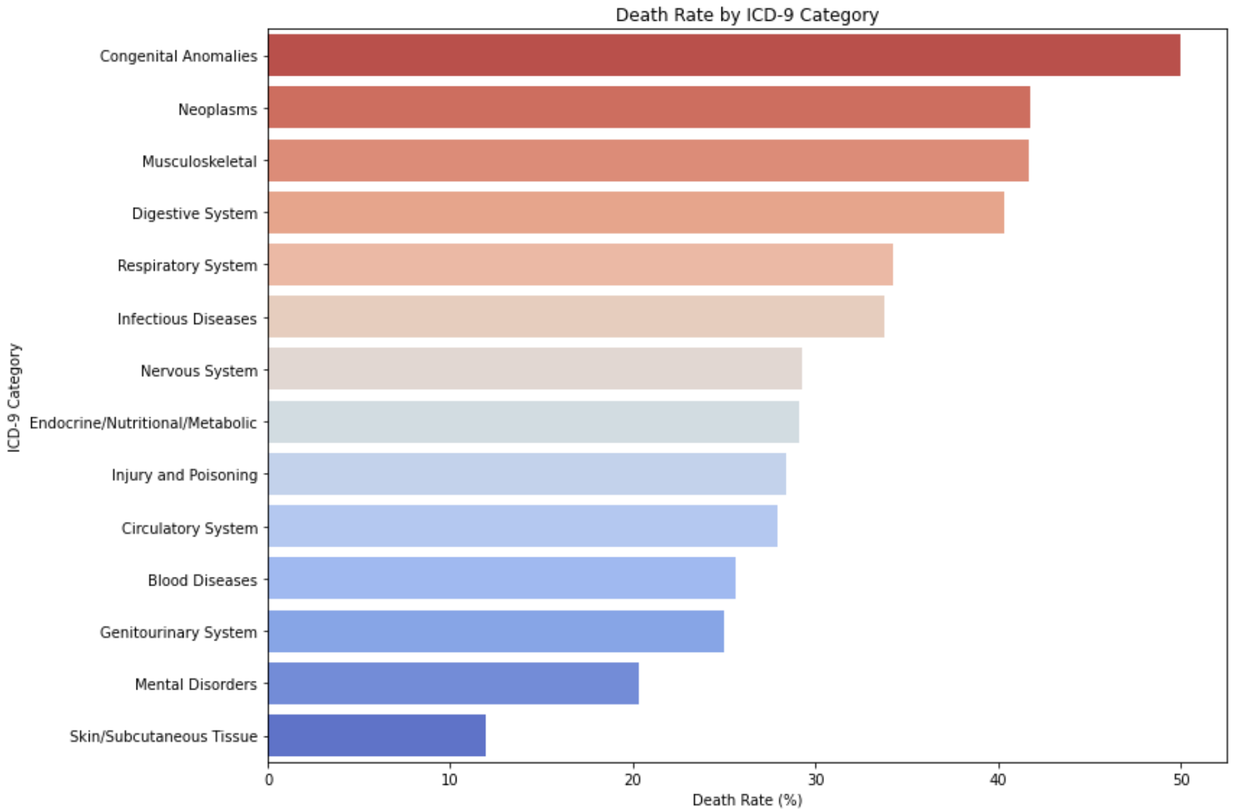
🔗 MIMIC-III Clinical Database

Data Set

- 데이터 정보
- Death Pattern

EDA

MIMIC III 에서 사망 원인



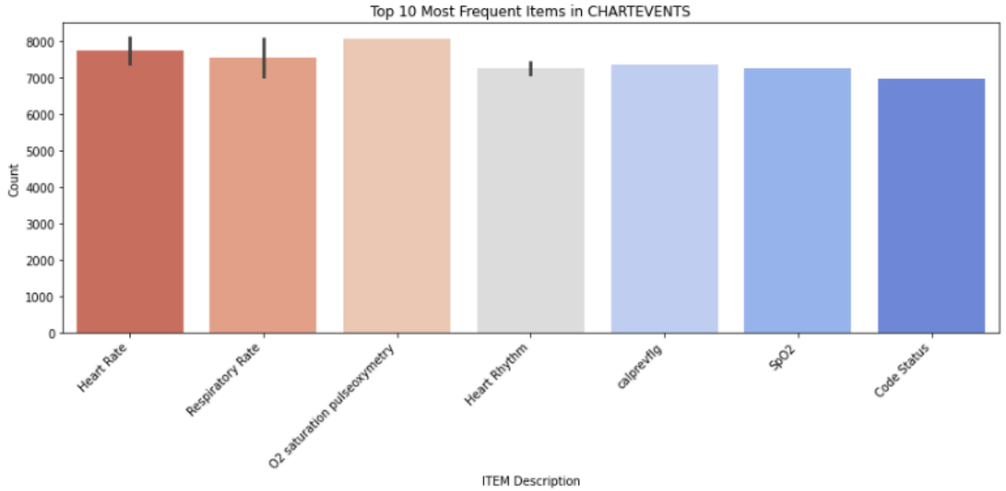
Congenital anormalies > Neoplasms > Musculoskeletal

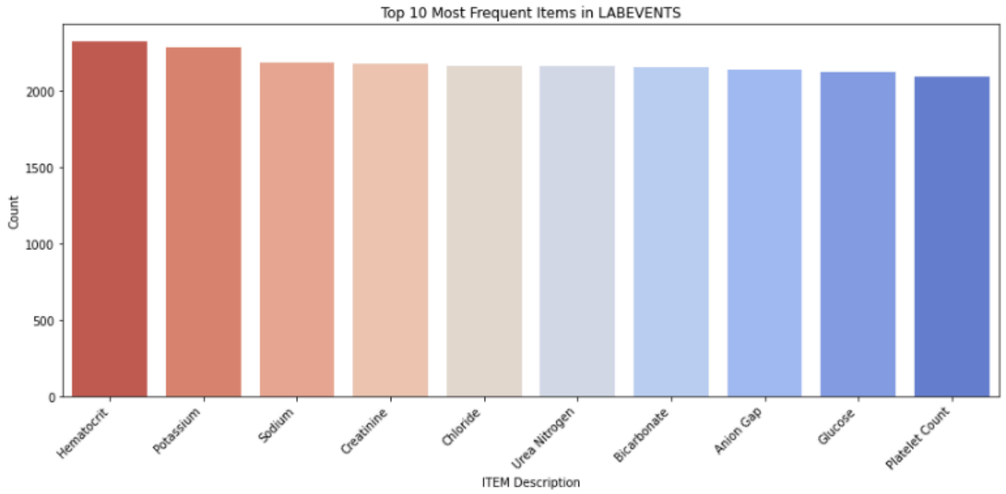
EDA

- 데이터 정보
- Death Pattern

MIMIC 데이터 살펴보기

MIMIC III 에서 사망 원인





Analysis Topic

1번째 주제

2번째 주제

ICU 환자의 항생제 사용과 치료 성공률 분석

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

Data PreProcessing

- Using Data Set
- Column Filtering
- Feature Engineering

ICUSTAYS.CSV

- 중환자실 체류 정보
- 컬럼 개수 : 12개

	subject_id	hadm_id	icustay_id	dbsource	first_careunit	last_careunit	first_wardid	last_wardid	intime	outtime	los	hospital_expire_flag
0	268	110404	280836	carevue	MICU	MICU	52	52	2198-02-14 23:27:38	2198-02-18 05:26:11	3.2490	1
1	269	106296	206613	carevue	MICU	MICU	52	52	2170-11-05 11:05:29	2170-11-08 17:46:57	3.2788	0

PRESCRIPTIONS.CSV

- 약물 사전
- 컬럼 개수 : 19개

	ROW_ID	SUBJECT_ID	HADM_ID	ICUSTAY_ID	STARTDATE	ENDDATE	DRUG_TYPE	DRUG	DRUG_NAME_POE	DRUG_NAME_GENERIC	FORMULARY_DRUG_CD	GSN	NDC	PROD_STRENGTH
0	2214776	6	107064	NaN	2175-06-11 00:00:00	2175-06-12 00:00:00	MAIN	Tacrolimus	Tacrolimus	Tacrolimus	TACR1	021796	469061711.0	1mg Capsule
1	2214775	6	107064	NaN	2175-06-11 00:00:00	2175-06-12 00:00:00	MAIN	Warfarin	Warfarin	Warfarin	WARF5	006562	56017275.0	5mg Tablet

LABEVENTS.CSV

- 실험실 검사 결과
- 컬럼 개수 : 9개

	ROW_ID	SUBJECT_ID	HADM_ID	ITEMID	CHARTTIME	VALUE	VALUENUM	VALUEUOM	FLAG
0	281	3	NaN	50820	2101-10-12 16:07:00	7.39	7.39	units	NaN
1	282	3	NaN	50800	2101-10-12 18:17:00	ART	NaN	NaN	NaN

ADMISSIONS.CSV

- 환자 입원 정보
- 컬럼 개수 : 19개

	ROW_ID	SUBJECT_ID	HADM_ID	ADMITTIME	DISCHTIME	DEATHTIME	ADMISSION_TYPE	ADMISSION_LOCATION	DISCHARGE_LOCATION	INSURANCE	LANGUAGE	RELIGION	MARITAL_STATUS
0	21	22	165315	2196-04-09 12:26:00	2196-04-10 15:54:00	NaN	EMERGENCY	EMERGENCY ROOM ADMIT	DISC-TRAN CANCER/CHLDRN H	Private	NaN	UNOBTAINABLE	MARRIED
1	22	23	152223	2153-09-03 07:15:00	2153-09-08 19:10:00	NaN	ELECTIVE	PHYS REFERRAL/NORMAL DELI	HOME HEALTH CARE	Medicare	NaN	CATHOLIC	MARRIED

Data PreProcessing

- Using Data Set
- Column Filtering
- Feature Engineering

ICU Data Filtering

```
# ICU내 필요한 Column 필터링
icustays = icustays[['subject_id', 'hadm_id', 'icustay_id', 'los', 'hospital_expire_flag']]
```

- **ICUSTAYS 테이블의 필요 컬럼만 유지**
  - **subject_id** : 환자 ID
  - **hadm_id** : 입원 ID
  - **icustay_id** : ICU 체류 ID
  - **los** : ICU 체류 기간(일 단위)
  - **hospital_expire_flag** : 환자의 사망 여부 플래그

Data PreProcessing

- Using Data Set
- Column Filtering
- Feature Engineering

ICU Data Filtering

```
# LABEVENTS 검사 지표 필터링
infection_markers = [51300, 51301, 51200, 51000]
infection_tests = labevents[labevents['itemid'].isin(infection_markers)]

# 기존의 Column 유지
infection_tests = infection_tests[['subject_id', 'hadm_id', 'itemid', 'valuenum', 'charttime']]
infection_tests.head()
```

- **ICU 환자의 감염 여부 파악(LABEVENTS)**
  - **item_id** : 검사 항목 ID
  - **valuenum**: 차트에 기록된 시간
  - **charttime**: 환자의 사망 여부 플래그

Data PreProcessing

- Using Data Set
- Column Filtering
- Feature Engineering

Drug Data Filtering(1차)

```
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'
]


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

```
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
```

- 공식 문서에서 참고한 사용 항생제 리스트로 Filtering 진행
- 총 항생제의 개수는 141개

Data PreProcessing

- Using Data Set
- Column Filtering
- Feature Engineering

Drug Data Filtering(1차)

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

'NEO*IV*Gentamicin', 'Gentamicin', 'Gentamicin Sulfate',
'Gentamicin Sulfate Ophth.', 'Gentamicin Sulf. Ophth. Soln',
'Gentamicin ', 'NF*GENTAMICIN SOP', 'Gentamicin ophth oint 0.3%',
'Gentamicin Intraventricular', 'Gentamicin 0.1% Cream',
'Gentamicin 0.3% Ophth. Ointment', 'Gentamicin 0.3% Ophth. Soln'
```

```
Penicillin G K Desensitization 의 개수 : 195
Meropenem Desensitization 의 개수 : 208
CefTAZidime Desensitization 의 개수 : 7
CefazoLIN Desensitization 의 개수 : 21
Cefepime Desensitization 의 개수 : 41
Ceftriaxone Desensitization 의 개수 : 13
```

- 항생제의 사용 방법에 관한 데이터 존재(추가 설명)
- But 항생제 사용 여부에 중점을 두기에 해당 데이터 사용

Data PreProcessing

- Using Data Set
- Column Filtering
- Feature Engineering

Drug Data Filtering(2차)

```python
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
```

```python
for name in antibiotics:
    antibiotics_prescriptions = data_rename(antibiotics_prescriptions, name)
```

```
약물 고유 개수 :  26
해당 약물 사용 데이터 개수 :  193795
```

- 총 26개의 항생제에 관하여 분석 진행

Data PreProcessing

- Using Data Set
- Column Filtering
- Feature Engineering

Feature Engineering

```
data['antibiotic_duration'] = (data['enddate'] - data['startdate']).dt.days
```

- 항생제 투여 기간 계산

```
# 최종 데이터 셋
data = data.groupby(['subject_id', 'hadm_id']).first().reset_index()
data = data.merge(lab_features, on=['subject_id', 'hadm_id'], how='left')
```

- 최종 데이터

Modeling

- 데이터 및 모델 관련
- 1st Modeling
- 2nd Modeling
- Optuna Tuning

학습 데이터 정리

```
# Prepare features and labels
X = data.drop(columns=['hospital_expire_flag', 'subject_id', 'hadm_id', 'icustay_id_y', 'drug'])
y = data['hospital_expire_flag']
```

Feature(X)

- **Drop**
  - **hospital_expire_flag** : Target
  - **subject_id, hadm_id , icustay_id_y**: 고유 ID
  - **drug** : 약물 이름에 관한 정보는 불필요

Label(y)

- **Drop**
  - **hospital_expire_flag** : Target(이진 형태)
    - 0 : 생존, 1 : 사망

Modeling

- 데이터 및 모델 관련
- 1st Modeling
- 2nd Modeling
- Optuna Tuning

모델 선정

이진 분류의 특성을 가진 데이터

- 클래스 불균형으로 인해 다수의 생존 데이터에 과적합될 가능성이 있음
  - 생존과 사망 이진 분류로 구성

Random Forest

- 과적합 방지에 유리
- 클래스 불균형 문제를 해결하는 “클래스 가중치 설정" 가능

XGBoost

- 대규모 데이터에 유리
  - MIMIC III의 경우, 대규모 데이터에 해당
- 빠른 속도

Modeling

- 데이터 및 모델 관련
- 1st Modeling
- 2nd Modeling
- Optuna Tuning

모델 적용

Random Forest

- random_state 고정
- 모델 초기화, 학습 및 예측, 성능 평가 순으로 진행
  - class 0과 1의 Precision, Recall, F1-Score 계산
  - 2개 class의 Macro Avg, Weighted Avg 계산(추가 설명)
  - 전체 평가 지표 : **ROC-AUC Score(지표 사용 이유)**

```
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))
```

Modeling

- 데이터 및 모델 관련
- 1st Modeling
- 2nd Modeling
- Optuna Tuning

모델 적용

XGBoost

- random_state 고정, 라벨 인코더 비활성화(Warning 방지), 평가 지표를 logloss로 설정
- 모델 초기화, 학습 및 예측, 성능 평가 순으로 진행
- 평가 항목은 이전과 동일

```
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))
```

Modeling

- 데이터 및 모델 관련
- 1st Modeling
- 2nd Modeling
- Optuna Tuning

모델 적용

Hyperparameter Tuning

- GridSearch 방식
  - 조합할 하이퍼파라미터 값 지정
  - 최적 파라미터 출력
  - 최적 파라미터로 학습한 결과 출력

```
# 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]
}
```

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

...

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

```
# 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]
}
```

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

...

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

Modeling

- 데이터 및 모델 관련
- 1st Modeling
- 2nd Modeling
- Optuna Tuning

항생제 키워드 필터링_1차

튜닝 이전

- **Random Forest**

  ROC-AUC Score : 0.7436463242399466

```
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
```

- **XGBoost**

  ROC-AUC Score : 0.7470484650263881

```
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
```

Modeling

- 데이터 및 모델 관련
- 1st Modeling
- 2nd Modeling
- Optuna Tuning

항생제 키워드 필터링_1차

튜닝 이후

- **Random Forest**

  ROC-AUC Score : 0.751056381476250**6** -> 약 **0.008** ⬆️

```
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 : 0.764**0**216680294359 -> 약 **0.017** ⬆️

```
Best parameters:
    {'colsample_bytree': 1.0,
    'learning_rate': 0.01,
    'max_depth': 6,
    'n_estimators': 300,
    'subsample': 0.7
    }
```

```
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
```

Modeling

- 데이터 및 모델 관련
- 1st Modeling
- 2nd Modeling
- Optuna Tuning

항생제 키워드 필터링_2차

튜닝 이전

- **Random Forest**

  ROC-AUC Score : 0.7590781981714116

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

           0       0.88      0.99      0.94      9784
           1       0.64      0.08      0.14      1375

    accuracy                           0.88     11159
   macro avg       0.76      0.54      0.54     11159
weighted avg       0.86      0.88      0.84     11159

ROC-AUC Score: 0.7590781981714116
```

- **XGBoost**

  ROC-AUC Score : 0.7686489630565674

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

           0       0.89      0.98      0.93      9784
           1       0.54      0.16      0.24      1375

    accuracy                           0.88     11159
   macro avg       0.72      0.57      0.59     11159
weighted avg       0.85      0.88      0.85     11159

ROC-AUC Score: 0.7686489630565674
```

Modeling

- 데이터 및 모델 관련
- 1st Modeling
- 2nd Modeling
- Optuna Tuning

항생제 키워드 필터링_2차

튜닝 이후

- **Random Forest**

  ROC-AUC Score : 0.770252806065561**6** -> 약 **0.011** ⬆️

```
Best parameters:
    {'bootstrap': False,
    'max_depth': 20,
    'min_samples_leaf': 4,
    'min_samples_split': 10,
    'n_estimators': 200
    }
```

```
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**

  ROC-AUC Score : 0.789598416710027**5** -> 약 **0.021** ⬆️

```
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
```

Modeling

- 데이터 및 모델 관련
- 1st Modeling
- 2nd Modeling
- Optuna Tuning

Hyperparameter Tuning 추가

Optuna

- **사용 방법**
  - xgboost의 optuna 라이브러리 설치 및 import
    - **Study** : 목적 함수에 기반한 최적화
    - **Trial** : 목적함수 시행(다양한 조합으로 시행)

- **사용 이유**
  - GridSearch의 소요 시간 개선

- **추가 조건**
  - 클래스 불균형 문제 해결을 위한 class_weight를 balanced로 설정

🔗 velog.io/하이퍼파라미터 튜닝

Modeling

- 데이터 및 모델 관련
- 1st Modeling
- 2nd Modeling
- Optuna Tuning

Hyperparameter Tuning 추가

Optuna *RF 기준(XGB도 코드 유사)

- 파라미터 값 설정

```
# 파라미터 값 조합
    n_estimators = trial.suggest_int('n_estimators', 50, 300)
    max_depth = trial.suggest_int('max_depth', 5, 20)
    min_samples_split = trial.suggest_int('min_samples_split', 2, 10)
    min_samples_leaf = trial.suggest_int('min_samples_leaf', 1, 5)
    bootstrap = trial.suggest_categorical('bootstrap', [True, False])
    class_weight = 'balanced'
```

```
rf_model = RandomForestClassifier(
        n_estimators=n_estimators,
        max_depth=max_depth,
        min_samples_split=min_samples_split,
        min_samples_leaf=min_samples_leaf,
        bootstrap=bootstrap,
        class_weight=class_weight,
        random_state=42
    )
```

- Study 설정 및 Trial 시도 횟수 지정

```
rf_study = optuna.create_study(direction='maximize')
rf_study.optimize(objective_rf, n_trials=20)
print("Best Random Forest Parameters:", rf_study.best_params)
```

🔗 velog.io/하이퍼파라미터 튜닝

Modeling

- 데이터 및 모델 관련
- 1st Modeling
- 2nd Modeling
- Optuna Tuning

Hyperparameter Tuning (Optuna)

Random Forest

- ROC-AUC Score : 0.777209931797268**9** -> 약 **0.007** ⬆️

```
Best Random Forest Parameters:
  {'n_estimators': 255,
   'max_depth': 20,
   'min_samples_split': 2,
   'min_samples_leaf': 3,
   'bootstap': True
  }
```

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

           0       0.88      1.00      0.93      6502
           1       0.70      0.07      0.13       938

    accuracy                           0.88      7440
   macro avg       0.79      0.53      0.53      7440
weighted avg       0.86      0.88      0.83      7440

ROC-AUC Score: 0.7772099317972689
```

XGBoost

- ROC-AUC Score : 0.803**3**568808416501 -> 약 **0.014** ⬆️

```
Best XGBoost Parameters:
    {'n_estimators': 271,
    'max_depth': 12,
    'learning_rate': 0.05530542777551273,
    'min_child_weight': 7,
    'subsample': 0.9241201201834832,
    'colsample_bytree': 0.5273685647420798,
    'gamma': 1.515287778996012,
    'class_weight': 4
    }
```

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

           0       0.89      0.99      0.94      6502
           1       0.67      0.13      0.21       938

    accuracy                           0.88      7440
   macro avg       0.78      0.56      0.57      7440
weighted avg       0.86      0.88      0.85      7440

ROC-AUC Score: 0.8033568808416501
```

회고

항생제 사용을 통한 치료 성공률

- 최대 ROC-AUC Score : **0.803**3568808416501 (XGB)

항생제 지표 외 원인 분석

- **데이터 전처리**
  - 도메인 지식에 대한 아쉬움 : 데이터 전처리(정규화, 인코딩, 리샘플링) 등이 더욱 체계적으로 이루어질 수 있다면 성능 향상에 도움이 될 것으로 예상

- **항생제 사용 실패 원인**
  - 항생제의 사용 유무를 제외한 다른 요소를 고려하지 않음
  - 항생제 부작용 : 감염 치료 효과를 분석하여 부작용 요소를 제거하게 되면, 치료 성공률이 더욱 향상될 것으로 추측

Analysis Topic

1번째 주제

2번째 주제

중환자실 내 다제내성균(MDR) 감염 치료 효과 분석

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

*다제 내성균(MDR) : 항생제에 대해 내성을 가진 병원체

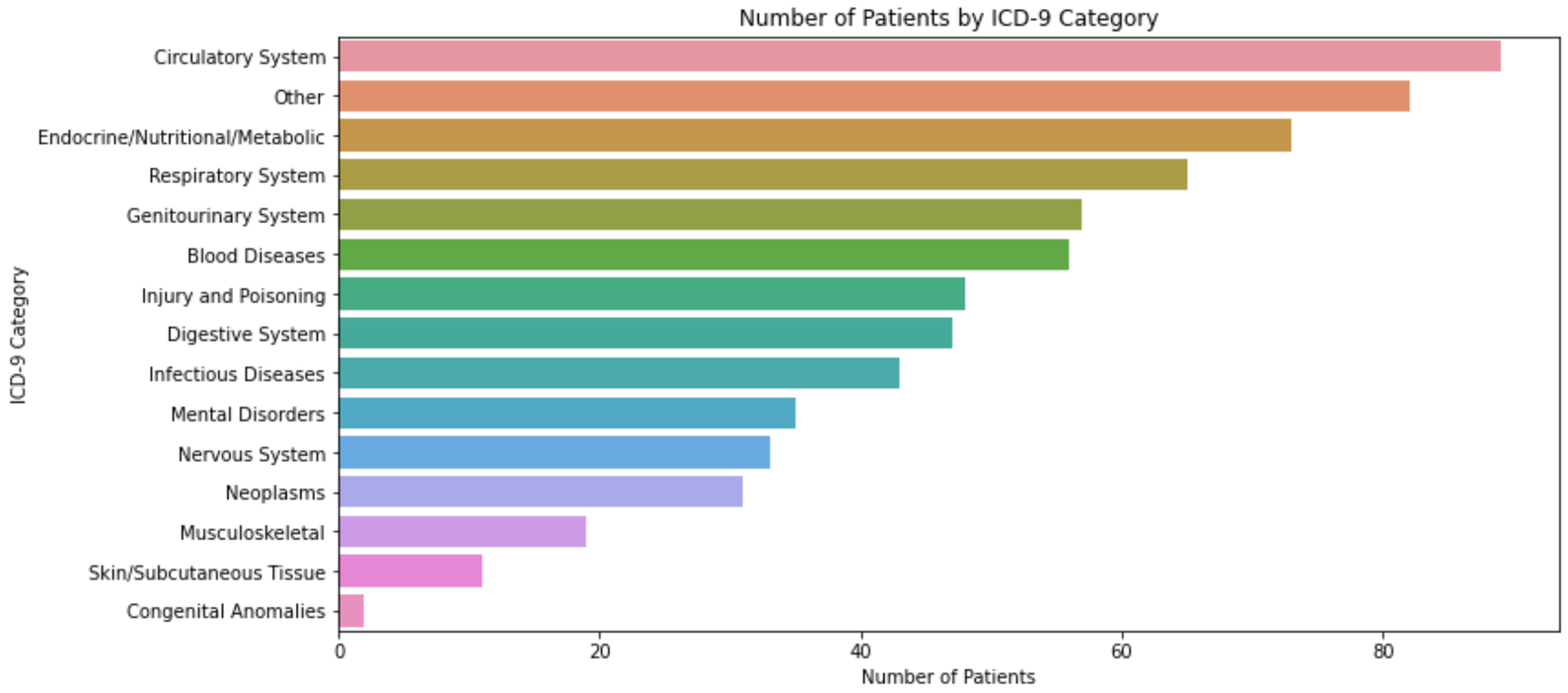
EDA

DISEASE PATTERN

ICD-9 코드(국제질병분류 9차 개정판): 전 세계적으로 질병, 상태 및 절차를 분류하는 표준화된 코드 집합

- 질병과 의료 절차를 정확하게 식별하고 기록하는 데 사용
- 의료 기관 간의 진단 일관성을 유지
- 건강 통계 추적, 의료 서비스 청구 및 보험 처리, 연구 데이터 수집에 필수적인 도구
- 특정 질병이나 상태를 나타내는 코드를 통해 의료 제공자들은 환자의 건강 상태를 일관되게 기록하고 분석 가능
  https://ko.wikipedia.org/wiki/ICD-9_%EC%BD%94%EB%93%9C_%EB%AA%A9%EB%A1%9D

MIMIC III 에서 ICD-9 범주에 따른 환자 수

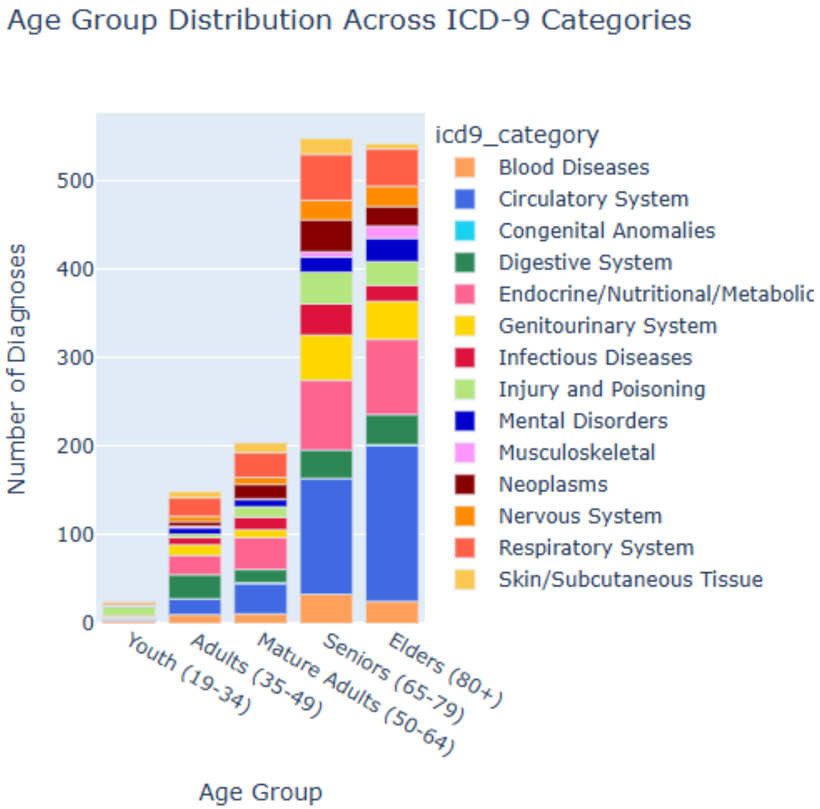


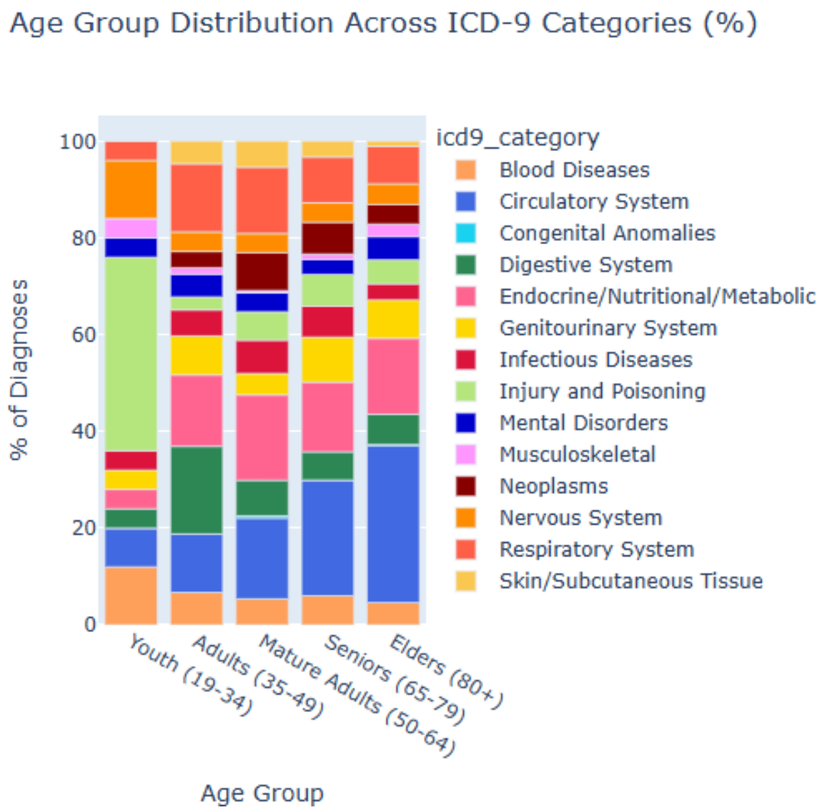
Circulatory System > Endocrine/Nutritional/Metabolic > Respiratory System

EDA

DISEASE PATTERN

MIMIC III 에서 ICD-9 범주에 따른 연령 별 환자 수





- *Senior* 이상에서 환자가 다수를 차지
- *Youth*에서는 상대적으로 *Injury*로 인한 입원 환자가 가장 많은 비율
- *Adult*에서 *Digestive System* 입원 환자가 많음
- *Adult* 이상에서는 *Circulatory System*이 점차 늘어남

Data PreProcessing

- Using Data Set
- Column Filtering
- Feature Engineering
- Result Indicator

ADMISSIONS.CSV

- 환자 입원 정보
- 컬럼 개수 : 19개

	ROW_ID	SUBJECT_ID	HADM_ID	ADMITTIME	DISCHTIME	DEATHTIME	ADMISSION_TYPE	ADMISSION_LOCATION	DISCHARGE_LOCATION	INSURANCE	LANGUAGE	RELIGION	MARITAL_STATUS
0	21	22	165315	2196-04-09 12:26:00	2196-04-10 15:54:00	NaN	EMERGENCY	EMERGENCY ROOM ADMIT	DISC-TRAN CANCER/CHLDRN H	Private	NaN	UNOBTAINABLE	MARRIED
1	22	23	152223	2153-09-03 07:15:00	2153-09-08 19:10:00	NaN	ELECTIVE	PHYS REFERRAL/NORMAL DELI	HOME HEALTH CARE	Medicare	NaN	CATHOLIC	MARRIED

LABEVENTS.CSV

- 실험실 검사 결과
- 컬럼 개수 : 9개

	ROW_ID	SUBJECT_ID	HADM_ID	ITEMID	CHARTTIME	VALUE	VALUENUM	VALUEUOM	FLAG
0	281	3	NaN	50820	2101-10-12 16:07:00	7.39	7.39	units	NaN
1	282	3	NaN	50800	2101-10-12 18:17:00	ART	NaN	NaN	NaN

PRESCRIPTIONS.CSV

- 약물 사전
- 컬럼 개수 : 19개

	ROW_ID	SUBJECT_ID	HADM_ID	ICUSTAY_ID	STARTDATE	ENDDATE	DRUG_TYPE	DRUG	DRUG_NAME_POE	DRUG_NAME_GENERIC	FORMULARY_DRUG_CD	GSN	NDC	PROD_STRENGTH
0	2214776	6	107064	NaN	2175-06-11 00:00:00	2175-06-12 00:00:00	MAIN	Tacrolimus	Tacrolimus	Tacrolimus	TACR1	021796	469061711.0	1mg Capsule
1	2214775	6	107064	NaN	2175-06-11 00:00:00	2175-06-12 00:00:00	MAIN	Warfarin	Warfarin	Warfarin	WARF5	006562	56017275.0	5mg Tablet

DIAGNOSES_ICD.CSV

- 환자의 상태
- 컬럼 개수 : 5개

	ROW_ID	SUBJECT_ID	HADM_ID	SEQ_NUM	ICD9_CODE
0	1297	109	172335	1.0	40301
1	1298	109	172335	2.0	486

D_LABITEMS.CSV

- 실험실 검사 항목
- 컬럼 개수 : 5개

	ROW_ID	ITEMID	LABEL	FLUID	CATEGORY	LOINC_CODE
0	546	51346	Blasts	Cerebrospinal Fluid (CSF)	Hematology	26447-3
1	547	51347	Eosinophils	Cerebrospinal Fluid (CSF)	Hematology	26451-5

Data PreProcessing

- Using Data Set
- Column Filtering
- Feature Engineering
- Result Indicator

MDR Patients Filtering

```
mdr_icd_codes = ['04104', '04112', '04119', '04184', '04103','0416', '0417']

# MDR 감염 환자 필터링
mdr_patients = diagnoses_icd[diagnoses_icd['ICD9_CODE'].isin(mdr_icd_codes)]

# 필요한 환자 정보 추출
mdr_subject_ids = mdr_patients['SUBJECT_ID'].unique()
print(f"MDR 감염 환자 수: {len(mdr_subject_ids)}")

MDR 감염 환자 수: 1567
```

- MDR 병원체의 ICD-9 필터링
- MDR 감염 환자 필터링

```
#prescriptions 테이블에서 mdr_subject_ids와 일치하는 항생제 데이터를 추출
mdr_prescriptions = prescriptions[prescriptions['SUBJECT_ID'].isin(mdr_subject_ids)]
print(f"MDR 감염 환자의 약물 처방 데이터 수: {mdr_prescriptions.shape}")

MDR 감염 환자의 항생제 처방 데이터 수: (410015, 19)
```

- MDR 감염환자에게 처방한 약물 데이터 필터링

Data PreProcessing

- Using Data Set
- Column Filtering
- Feature Engineering
- Result Indicator

DRUG Data Filtering

```python
# 항생제 목록
target_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'
]

# PRESCRIPTIONS 테이블에서 항생제 필터링
mdr_antibiotics = mdr_prescriptions[
    mdr_prescriptions['DRUG_NAME_GENERIC'].str.upper().isin(target_antibiotics)
]

# 필터링된 데이터 확인
print(f"필터링된 항생제 데이터 수: {mdr_antibiotics.shape[0]}")

필터링된 항생제 데이터 수: 1318
```

- MDR 환자에게 처방한 약물 데이터에서 항생제 데이터 필터링

Data PreProcessing

- Using Data Set
- Column Filtering
- Feature Engineering
- Result Indicator

LABEVENTS Data Filtering

```
# MDR 감염 환자의 SUBJECT_ID를 기반으로 LABEVENTS 필터링
mdr_labevents = labevents[labevents['SUBJECT_ID'].isin(mdr_prescriptions['SUBJECT_ID'].unique())]

# 필터링된 LABEVENTS 데이터에서 ITEMID 추출
mdr_item_ids = mdr_labevents['ITEMID'].unique()

# D_LABITEMS 테이블에서 해당 ITEMID에 매핑된 LABEL 확인
mdr_labels = d_labitems[d_labitems['ITEMID'].isin(mdr_item_ids)]
print(f"총 라벨 개수: {mdr_labels.shape[0]}")

총 라벨 개수: 637
```

- MDR 감염 환자 환자 ID 필터링
- MDR 감염 환자의 검사 항목 필터링

```
# 감염 지표로 유효한 항목의 ITEMID 필터링
infection_related_itemids = [
    50889,  # C-Reactive Protein
    51300,  # WBC Count(백혈구 수)
    51144,  # Bands(미성숙 백혈구)
    51256,  # Neutrophils(중성구)
    51265,  # Platelet Count(혈소판 수치)
    50954   # Lactate(조직 저산소증 및 패혈증 평가)
]

filtered_labevents = labevents[labevents['ITEMID'].isin(infection_related_itemids)]
print(f"감염 관련 라벨 수: {filtered_labevents.shape[0]}")

감염 관련 라벨 수: 9
```

- 검사 항목 중 감염 지표 관련 데이터 필터링

Data PreProcessing

- Using Data Set
- Column Filtering
- Feature Engineering
- Result Indicator

LABEVENTS Data Filtering

```
# 혈소판 수치(ITEMID=51265)의 값 데이터 필터링
platelet_data = labevents[labevents['ITEMID'] == 51265]
platelet_data = platelet_data.sort_values(by=['SUBJECT_ID', 'CHARTTIME'])

print(platelet_data[['SUBJECT_ID', 'CHARTTIME', 'VALUENUM', 'FLAG']])
```

```
        SUBJECT_ID            CHARTTIME  VALUENUM      FLAG
609              2  2138-07-17 20:48:00       5.0  abnormal
632              2  2138-07-17 21:10:00     302.0       NaN
```

- 혈소판 수치 관련 데이터 필터링

```
# FLAG가 abnormal(이상) 데이터 필터링
abnormal_data = labevents[labevents['FLAG'] == 'abnormal']
print(abnormal_data[['SUBJECT_ID', 'ITEMID', 'CHARTTIME', 'VALUENUM', 'VALUEUOM', 'FLAG']])
```

```
        SUBJECT_ID  ITEMID            CHARTTIME  VALUENUM VALUEUOM      FLAG
4                3   50808  2101-10-12 18:17:00      0.93   mmol/L  abnormal
15               3   50912  2101-10-13 03:00:00      1.70    mg/dL  abnormal
```

- 검사 결과가 이상으로 나온 데이터 필터링

Data PreProcessing

- Using Data Set
- Column Filtering
- Feature Engineering
- Result Indicator

LABEVENTS Data Filtering

```
# ITEMID가 필요한 데이터로 필터링
# 치료 효과 분석: 검사 결과 초기 값과 이후 값 비교
def calculate_change(group):
    initial = group.iloc[0]['VALUENUM']
    final = group.iloc[-1]['VALUENUM']

    return pd.Series({'INITIAL_VALUE': initial, 'FINAL_VALUE': final, 'CHANGE': final - initial})

filtered_labevents = labevents[labevents['ITEMID'].isin(infection_related_itemids)]
changes = filtered_labevents.groupby(['SUBJECT_ID', 'ITEMID']).apply(calculate_change).reset_index()
print(changes.head())
```

```
   SUBJECT_ID  ITEMID  INITIAL_VALUE  FINAL_VALUE  CHANGE
0           2   51144            0.0          1.0     1.0
1           2   51256          100.0         70.0   -30.0
```

- 항생제 사용 전, 후 치료 효과 분석
- 항생제 사용 전, 후 감염 지표에 대한 차이점

Data PreProcessing

- Using Data Set
- Column Filtering
- Feature Engineering
- Result Indicator

LABEVENTS Data Filtering

```python
# 성공 기준 정의
success_criteria = {
    50889: lambda x: x < 10,         # C-Reactive Protein (CRP) ; < 10 mg/L
    51300: lambda x: 4 <= x <= 11,   # WBC Count : 4–11 × 10^9/L
    51144: lambda x: x < 5,          # Bands : < 5%
    51256: lambda x: 40 <= x <= 70, # Neutrophils : 40–70%
    51265: lambda x: 150 <= x <= 450, # Platelet Count : 150–450 K/uL
    50954: lambda x: x < 2           # Lactate : < 2 mmol/L (중증 감염 시 중요)
}

# 성공 여부 추가
changes['SUCCESS'] = changes.apply(
    lambda row: success_criteria[row['ITEMID']](row['FINAL_VALUE']) if row['ITEMID'] in
success_criteria else False,
    axis=1
)

print(changes[['SUBJECT_ID', 'ITEMID', 'INITIAL_VALUE', 'FINAL_VALUE', 'SUCCESS']])
```

```
        SUBJECT_ID  ITEMID  INITIAL_VALUE  FINAL_VALUE  SUCCESS
0                2   51144            0.0          1.0     True
1                2   51256          100.0         70.0     True
```

- 항생제 사용 전, 후의 검사 결과가 조금이라도 변경 되었다면(True)

Data PreProcessing

- Using Data Set
- Column Filtering
- Feature Engineering
- Result Indicator

카이 제곱 검정

```python
from scipy.stats import chi2_contingency

# SUCCESS 여부와 항생제 사용 빈도
contingency_table = pd.crosstab(features['SUCCESS'], features['DRUG_NAME_GENERIC'])
chi2, p, dof, expected = chi2_contingency(contingency_table)

# 결과 출력
print(f"카이제곱 검정 결과: chi2={chi2}, p={p}")
if p < 0.05:
    print("항생제 사용과 치료 성공 간의 관계가 유의미합니다.")
else:
    print("항생제 사용과 치료 성공 간의 관계가 유의미하지 않습니다.")
```

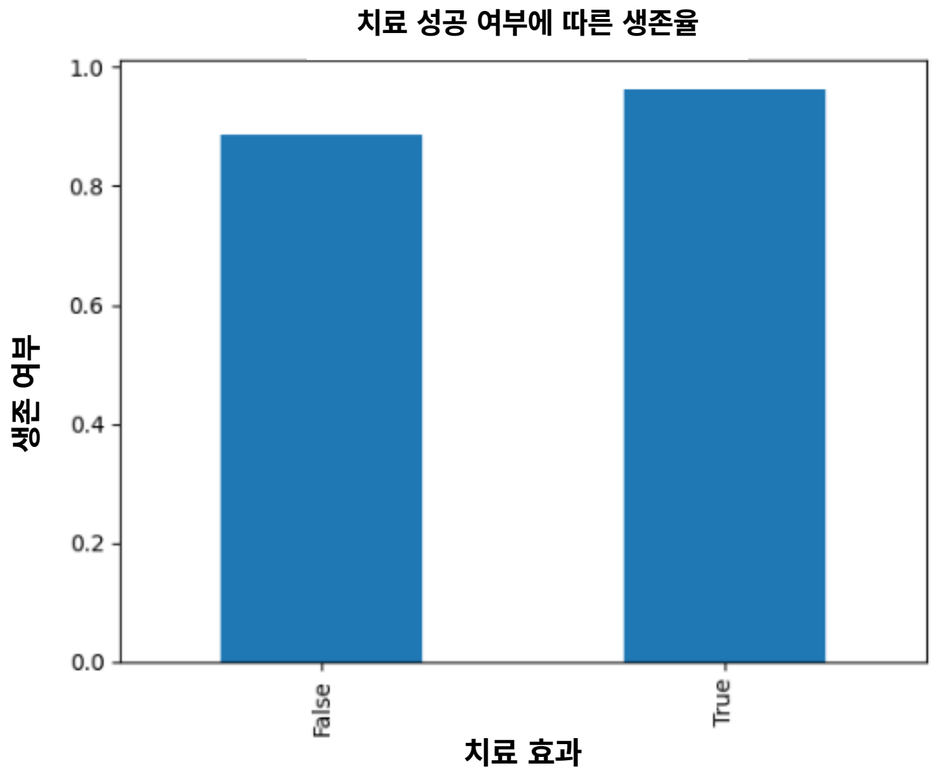
```
카이제곱 검정 결과: chi2=435.4865866342416, p=1.762831705349424e-86
항생제 사용과 치료 성공 간의 관계가 유의미합니다.
```

- 가설 : 항생제의 사용 여부가 감염 지표 결과 변화(SUCCESS)의 영향을 끼치는 지

Data PreProcessing

- Using Data Set
- Column Filtering
- Feature Engineering
- Result Indicator

결과 지표



그래프에서 관찰된 점

1) 치료효과 = True에서 높은 생존율
   - 치료가 성공한 환자의 생존율이 매우 높음

2) 치료효과 = False에서도 높은 생존율
   - 치료가 성공하지 못했음에도 불구하고 생존율이 비교적 높음

Data PreProcessing

- Using Data Set
- Column Filtering
- Feature Engineering
- Result Indicator

항생제와 SUCCESS 변수에 대한 관계

```python
from scipy.stats import ttest_ind

# 항생제 목록
target_antibiotics = features['DRUG_NAME_GENERIC'].unique()

# 항생제별 t-검정 결과 저장
t_test_results = []

for drug in target_antibiotics:
    # 해당 항생제를 사용한 데이터 필터링
    drug_data = features[features['DRUG_NAME_GENERIC'] == drug]

    # SUCCESS=True와 SUCCESS=False의 CHANGE 값 추출
    success_group = drug_data[drug_data['SUCCESS'] == True]['CHANGE']
    failure_group = drug_data[drug_data['SUCCESS'] == False]['CHANGE']

    # t-검정 수행
    t_stat, p_value = ttest_ind(success_group, failure_group, nan_policy='omit')

    # 결과 저장
    t_test_results.append({'DRUG_NAME_GENERIC': drug, 't_stat': t_stat, 'p_value': p_value})

# 결과를 데이터프레임으로 정리
t_test_df = pd.DataFrame(t_test_results)
```

	DRUG_NAME_GENERIC	t_stat	p_value
3	ERYTHROMYCIN	-17.875945	4.004132e-67
4	RIFAMPIN	3.785209	1.692247e-04
6	CLINDAMYCIN	-2.868488	4.518490e-03
7	CEFTRIAXONE	-2.453442	1.644046e-02
12	VANCOMYCIN	-1.822925	9.830602e-02
1	LEVOFLOXACIN	-1.272601	2.031765e-01
9	CEFEPIME	-1.192733	2.462768e-01
10	TETRACYCLINE	-1.085581	2.894173e-01
8	CEFUROXIME	-0.937226	3.572671e-01
2	LINEZOLID	-0.882907	3.773263e-01
11	CEFAZOLIN	-0.594558	5.667831e-01
5	AMPICILLIN	-0.053829	9.570868e-01
0	NaN	NaN	NaN

항생제 이름	t-값	p-값	해석
ERYTHROMYCIN	-17.88	4.00e-67	두 그룹 간 차이가 매우 유의미.
RIFAMPIN	3.79	1.69e-04	두 그룹 간 차이가 유의미.
CLINDAMYCIN	-2.87	4.52e-03	두 그룹 간 차이가 유의미.
CEFRIAXONE	-2.45	1.64e-02	두 그룹 간 차이가 유의미.

- **총 4개의 항생제가 감염 지표 변화에 유의미한 사실 확인**

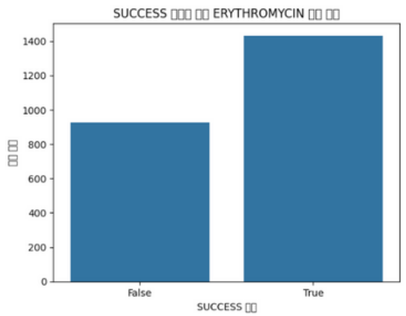
Correlation

- Using Data Set
- Column Filtering
- Feature Engineering
- Result Indicator

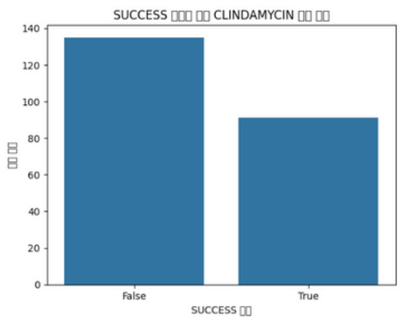
항생제 사용과 치료 성공 여부의 관계

특정 항생제의 사용이 치료 성공에 미치는 영향을 확인

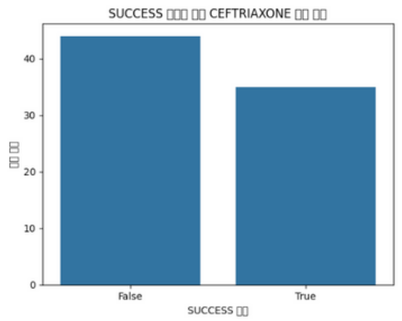
- 새롭게 선정된 항생제 목록에 따른 사용 빈도 시각화



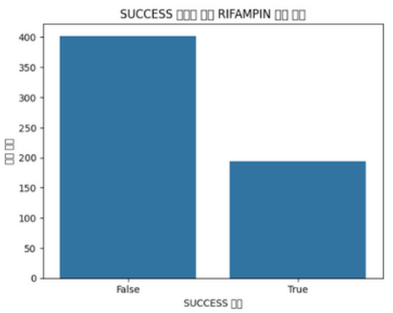
ERYTHROMYCIN



CLINDAMYCIN



CEFTRIAXONE



RIFAMPIN

회고

1. 임상지식, 공식문서 부족으로 개념 정의 모호함

- MDR 감염 환자의 정의
- 감염 지표에 대한 정의
- 감염 지표 단위 변환 어려움
- 치료 성공 기준(감염 지표의 정상화)에 대한 정의

2. 시간부족으로 인한 통제변인 조절 어려움

감사합니다!

지금까지 MEMI였습니다.👊