항치배 데이터 정리

```
import pandas as pd

# 첫 번째 CSV 파일 읽어오기
input_path_d3 = "C:/yelin/sp/합친파일/데이터/항생제.csv"
df_d3 = pd.read_csv(input_path_d3)

# 두 번째 CSV 파일 읽어오기
input_path_result = "C:/yelin/sp/합친파일/데이터/치료결과.csv"
df_result = pd.read_csv(input_path_result)


# VTHOS_NO와 RGNO가 같은 경우를 기준으로 합치기
merged_df = pd.merge(df_d3, df_result, on=['VTHOS_NO', 'RGNO'], how='left')

# 결과를 새로운 파일로 저장
output_path_merged = "C:/yelin/sp/합친파일/데이터/항치1.csv"
merged_df.to_csv(output_path_merged, index=False)
```

```
import pandas as pd

# CSV 파일을 읽어옴
file_path = "C:/yelin/sp/합친파일/데이터/항치1.csv"
data = pd.read_csv(file_path)

# null값 있는 행 제거
data = data.dropna()

# 결과를 CSV 파일로 저장
output_file_path = "C:/yelin/sp/합친파일/데이터/항치.csv"
data.to_csv(output_file_path, index=False)
```

```
import pandas as pd

# 첫 번째 CSV 파일 읽어오기
a = "C:/yelin/sp/합친파일/데이터/항치.csv"
a1 = pd.read_csv(a)

# 두 번째 CSV 파일 읽어오기
b = "C:/yelin/sp/합친파일/데이터/배양.csv"
b1 = pd.read_csv(b)


# VTHOS_NO와 RGNO가 같은 경우를 기준으로 합치기
c = pd.merge(a1, b1, on=['VTHOS_NO', 'RGNO'], how='left')

# 결과를 새로운 파일로 저장
output_path_merged = "C:/yelin/sp/합친파일/데이터/항치배1.csv"
c.to_csv(output_path_merged, index=False)
```

```
import pandas as pd

# CSV 파일을 읽어옴
file_path = "C:/yelin/sp/합친파일/데이터/항치배1.csv"
data = pd.read_csv(file_path)

# null값 있는 행 제거
data = data.dropna()

# 결과를 CSV 파일로 저장
output_file_path = "C:/yelin/sp/합친파일/데이터/항치배2.csv"
data.to_csv(output_file_path, index=False)
```

```
import pandas as pd

# CSV 파일을 읽어옴
file_path = "C:/yelin/sp/합친파일/데이터/항치배2.csv"
data = pd.read_csv(file_path)

# 'VTHOS_NO', 'RGNO' 열 제거
columns_to_drop = ['VTHOS_NO', 'RGNO']
data.drop(columns=columns_to_drop, inplace=True)

# 열 이름 변경
data.rename(columns={'RSLT_CONT_x': 'antibiotics', 'RSLT_CONT_y': 'RSLT_CONT'}, inplace=True)
```

```
# 결과를 CSV 파일로 저장
output_file_path = "C:/yelin/sp/합친파일/데이터/항치배.csv"
data.to_csv(output_file_path, index=False)
```

```
##########################################################################################################
```

```
import pandas as pd
from sklearn.model_selection import train_test_split

# CSV 파일에서 데이터 불러오기
data = pd.read_csv("C:/yelin/sp/합친파일/데이터/항치배.csv")

# 'DAY'는 수치형 데이터이므로 따로 선택
numeric_feature = ['TPRSC_CAPA', 'DAY']
X_numeric = data[numeric_feature]

# 'INGR_NM'을 원핫인코딩하여 선택
categorical_features = ['INGR_NM', 'ANTBT_NM','antibiotics']
X_categorical = pd.get_dummies(data[categorical_features], columns=categorical_features)

# 두 데이터프레임을 합쳐 입력 데이터 생성
X = pd.concat([X_categorical, X_numeric], axis=1)

# "RSLT_CONT" 값을 숫자로 변환
y = data['RSLT_CONT'].map({'R': 1, 'S': 0})


# 학습 데이터와 테스트 데이터 분할
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# 나눈 데이터의 크기 출력
print("학습 데이터:", X_train.shape)
print("테스트 데이터:", X_test.shape)
print("학습 레이블:", y_train.shape)
print("테스트 레이블:", y_test.shape)

# 학습 데이터와 테스트 데이터를 CSV 파일로 저장
X_train.to_csv('C:/yelin/sp/합친파일/항치배/X_train.csv', index=False)
X_test.to_csv('C:/yelin/sp/합친파일/항치배/X_test.csv', index=False)
y_train.to_csv('C:/yelin/sp/합친파일/항치배/y_train.csv', index=False, header=['RSLT_CONT'])
y_test.to_csv('C:/yelin/sp/합친파일/항치배/y_test.csv', index=False, header=['RSLT_CONT'])
```

```
학습 데이터: (465891, 133)
테스트 데이터: (199668, 133)
학습 레이블: (465891,)
테스트 레이블: (199668,)
```

```
##########################################################################################################
```

```
import pandas as pd

# CSV 파일로부터 데이터 불러오기
X_train = pd.read_csv('C:/yelin/sp/합친파일/항치배/X_train.csv')
X_test = pd.read_csv('C:/yelin/sp/합친파일/항치배/X_test.csv')
y_train = pd.read_csv('C:/yelin/sp/합친파일/항치배/y_train.csv')['RSLT_CONT']
y_test = pd.read_csv('C:/yelin/sp/합친파일/항치배/y_test.csv')['RSLT_CONT']


from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score
from sklearn.model_selection import train_test_split

# 랜덤 포레스트 모델 생성
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

# 예측 결과 얻기
y_pred = rf_model.predict(X_test)

# AUROC 계산
auroc = roc_auc_score(y_test, y_pred)

# 정확도 계산
accuracy = accuracy_score(y_test, y_pred)

# Precision 계산
precision = precision_score(y_test, y_pred, average='weighted')

# Recall 계산
```

```
recall = recall_score(y_test, y_pred, average='weighted')

# F1 Score 계산
f1 = f1_score(y_test, y_pred, average='weighted')

# 혼동 행렬 계산
conf_matrix = confusion_matrix(y_test, y_pred)
true_positive = conf_matrix[1][1]
false_positive = conf_matrix[0][1]
true_negative = conf_matrix[0][0]
false_negative = conf_matrix[1][0]

# 민감도와 특이도 계산
sensitivity = true_positive / (true_positive + false_negative)
specificity = true_negative / (true_negative + false_positive)

# 결과 출력
print(f"AUROC: {auroc:.3f}")
print(f"accuracy: {accuracy:.3f}")
print(f"Precision: {precision:.3f}")
print(f"Recall: {recall:.3f}")
print(f"F1 Score: {f1:.3f}")
print("Sensitivity:", sensitivity)
print("Specificity:", specificity)
```

```
AUROC: 0.732
accuracy: 0.784
Precision: 0.778
Recall: 0.784
F1 Score: 0.779
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score

# 로지스틱 회귀 모델 초기화
logistic_model = LogisticRegression(max_iter=1000)

# 모델 학습
logistic_model.fit(X_train, y_train)

# 예측
y_pred = logistic_model.predict(X_test)

# 평가 지표 계산
roc_auc = roc_auc_score(y_test, y_pred, average='weighted')
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')


# 결과 출력
print("AUROC:", roc_auc)
print("accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
```

```
AUROC: 0.6250765492777215
accuracy: 0.7282038183384418
Precision: 0.7137506878126055
Recall: 0.7282038183384418
F1 Score: 0.6987803478770321
```

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score

# 그레디언트 부스트 모델 모델 초기화
boosting_model = GradientBoostingClassifier(n_estimators=100, random_state=42)

# 모델 학습
boosting_model.fit(X_train, y_train)

# 예측
y_pred = boosting_model.predict(X_test)

# 평가 지표 계산
roc_auc = roc_auc_score(y_test, y_pred, average='weighted')
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')
```

```
# 결과 출력
print("AUROC:", roc_auc)
print("accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
```

```
AUROC: 0.6296007617598417
accuracy: 0.738485886571709
Precision: 0.7313899730907748
Recall: 0.738485886571709
F1 Score: 0.7050132928817835
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score


# AdaBoost 모델 초기화
adaboost_model = AdaBoostClassifier(random_state=42)

# 모델 학습
adaboost_model.fit(X_train, y_train)

# 예측
y_pred = adaboost_model.predict(X_test)

# 평가 지표 계산
roc_auc = roc_auc_score(y_test, y_pred, average='weighted')
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')

# 결과 출력
print("AUROC:", roc_auc)
print("accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
```

```
AUROC: 0.654463521298606
accuracy: 0.7411152513171865
Precision: 0.728324693650628
Recall: 0.7411152513171865
F1 Score: 0.7219480953899972
```