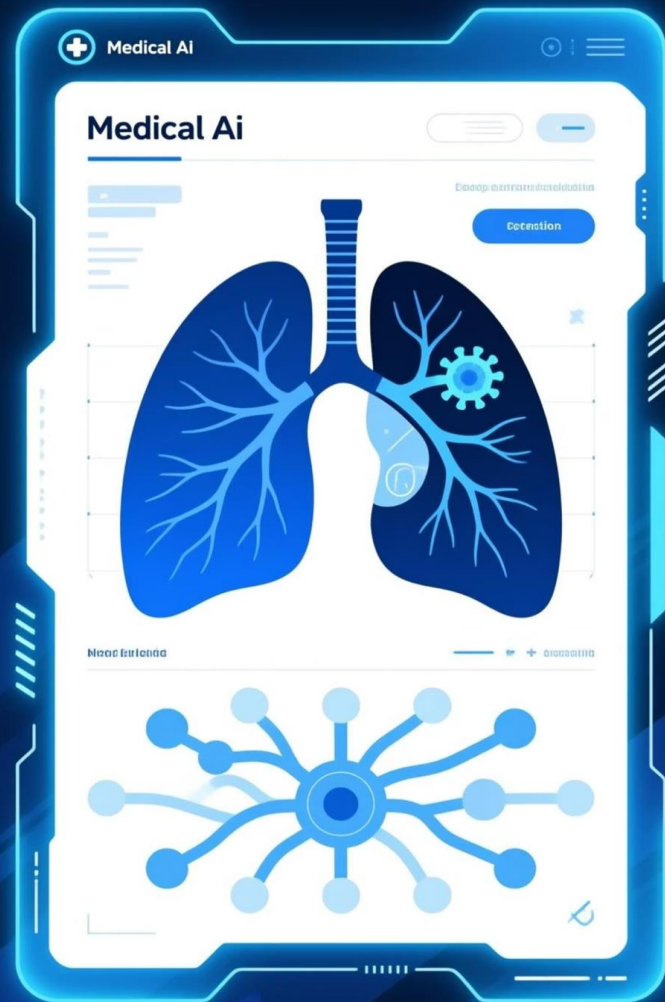


COVID-19와 관련된 AI모델 제작 팀 프로젝트

팀명 : COVID-19

팀원 : 신대현, 전홍균, 정승환



목차

1. 데이터 선정 및 소개
2. 데이터 전처리
3. 머신러닝 분석
4. 딥러닝 분석
5. 모델 구현 결과 및 후기



데이터 선정 및 소개

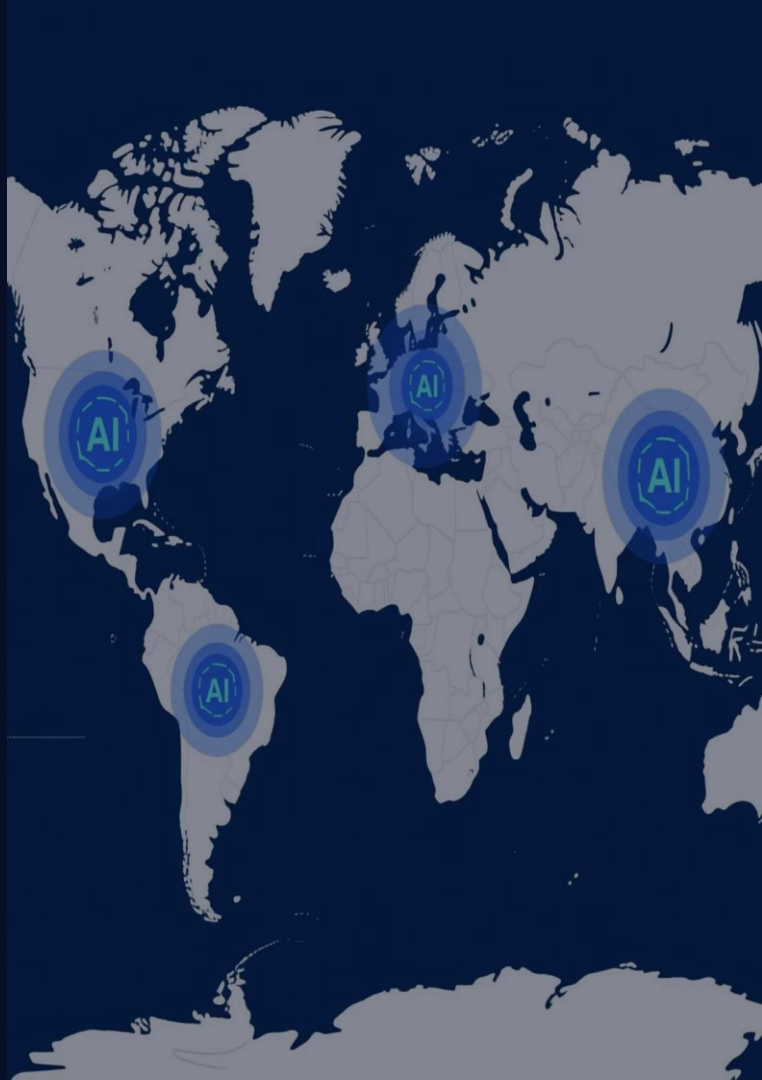
COVID-19 팬데믹과 AI의 역할

코로나19 팬데믹이 전 세계적으로 남긴 영향이 크기 때문이며, 데이터 기반 접근을 통해 향후 대응 전략 마련에 기여하고자 하였다.

2019년 말 발생한 전 세계적 보건 위기는 2025년 기준으로 2억 명 이상의 확진자와 400만 명 이상의 사망자를 초래하였다.

머신러닝 분석에서 멕시코 정부가 제공한 COVID-19 공공 데이터셋을 활용하여, 심장질환, 만성질환, 폐렴 등의 기저 질환이 사망과 어떠한 연관성을 갖는지 예측,조사 하였다.

또한, 딥러닝에서 COVID-19 환자의 흉부 X-ray 사진을 이용하여 양성 및 음성 환자를 분류하는 진단 보조 모델 구축을 목표로 하였다.



목표

머신러닝

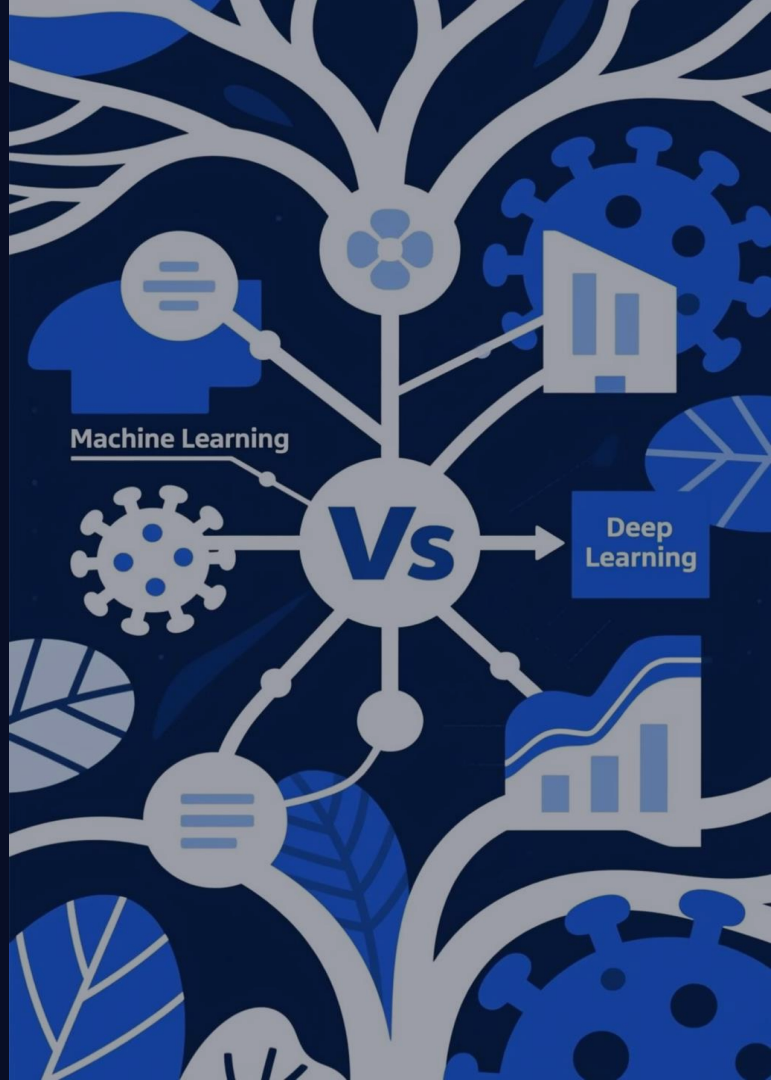
데이터에서 패턴을 발견하고 학습하여 모델 생성, COVID-19에 관한 여러 시점으로 데이터를 분석하였다.

- 다양한 모델을 사용하여 각 데이터에 맞는 결과 산출
- 약 1,000,000명의 데이터셋을 이용하여 사망과의 연관성 및 위험도 분석에 활용

딥러닝

데이터의 복잡한 특징을 추출하고 학습 후 분류, 예측과 같은 작업을 수행하며, 이미지를 이용한 비정형 데이터를 효과적으로 처리하였다.

- COVID-19 양성, 음성 환자의 흉부 X-ray 분석
- 약 1400장의 X-ray 사진을 이용하여 COVID-19 발생 예측
- 이진분류를 시각적으로 보여주는 GRAD-CAM기법 사용



머신러닝 공동작업 : 데이터전처리

데이터셋 : <https://www.kaggle.com/datasets/meirizri/covid19-dataset>

이 데이터 세트(1,048,576명의 고유 환자로 구성)는 멕시코 정부에서 제공했습니다

데이터 전처리

데이터 전처리 과정 설명

상관관계 히트맵

```
import numpy as np
import pandas as pd

# 데이터 처리
# 파일 경로 불러오기
df = pd.read_csv('Covid Data.csv')

# 'DATE_DIED' 열의 값이 '9999-99-99'이면 0, 아니면 1으로 변경
df['DEATH'] = np.where(df['DATE_DIED'] == '9999-99-99', 0, 1)

# 필요없는 열 삭제
df = df.drop(['MEDICAL_UNIT', 'DATE_DIED', 'PATIENT_TYPE', 'USER', 'INMISUR', 'OTHER_DISEASE', 'PREGNANT'], axis=1)

# 'AGE' 열의 값이 91 이상인 모든 행을 삭제
df = df[df['AGE'] < 91]

# 'IQU', 'INTUBED' 97/99/99 값 -(정보 없음)으로 변경
df['IQU'] = df['IQU'].replace(97: -1, 99: -1)
df['INTUBED'] = df['INTUBED'].replace(97: -1, 99: -1)

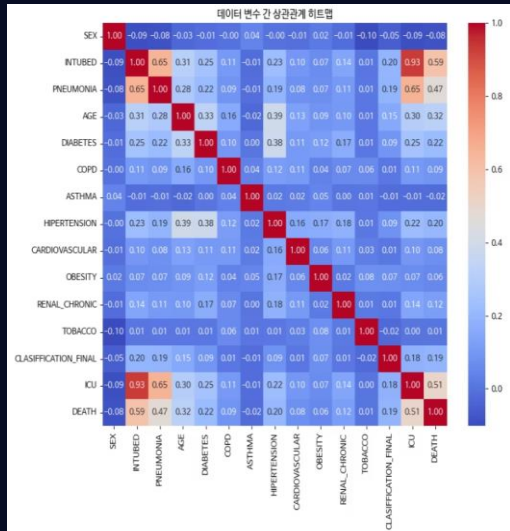
# 모든 열의 97, 99, 99 값을 pd.NA로 변경
df = df.replace(97: pd.NA, 99: pd.NA, 99: pd.NA)

# 'CLASSIFICATION_FINAL' 열의 값을 변경
df['CLASSIFICATION_FINAL'] = df['CLASSIFICATION_FINAL'].replace(list(range(1, 4)), 1)
df['CLASSIFICATION_FINAL'] = df['CLASSIFICATION_FINAL'].replace(4, 0)

# 'AGE' 열을 제외한 모든 열에서 2를 0으로 변경
columns_to_replace = [col for col in df.columns if col != 'AGE']
df[columns_to_replace] = df[columns_to_replace].replace(2: 0)

# NaN 값 제거
df = df.dropna()
```

단계	작업 내용	목적
1	데이터 확인	데이터의 구조, 결측치, 이상치 파악 (완료)
2	결측치 처리	NaN → 평균/중앙값/삭제/예측 등 (완료)
3	이상치 처리	값이 너무 크거나 작은 경우 → 제거/대체 (완료)
4	데이터 타입 변환	숫자형, 범주형, 날짜형으로 변환 (완료)
5	범주형 인코딩	문자형 → 숫자형으로 변환 (Label, OneHot 등) (완료)
6	스케일링	값 범위 정규화 (표준화, 정규화 등)(완)
7	데이터 분할	훈련/검증/테스트 세트 나누기(진행 중)



상세내용 : 주석참고

공동작업 : 범주형 인코딩 단계

상관관계 확인

데이터 전처리 상세내역

- 사망 날짜 카테고리 내 '9999-99-99' 생존자, 날짜 작성되어 있는 항목은 사망자로 사망 카테고리를 생성하여 0(생존자), 1(사망자)로 구분
- 기존 97/98/99로 작성되어 있는 Missing data(자료없음)를 NaN값으로 변경
- ICU(응급실)과 INTUBED(삽관) 카테고리는 97/98/99 항목을 -1 값으로 변경 (-1은 대중적으로 해당없음을 뜻하며 상황에 따라 삭제 가능)
- CLASIFFICATION_FINAL(코로나 확진) 카테고리 내 1~7까지 있는 항목을 0과 1로 변경
- 기존 1(긍정)과 2(부정)으로 되어있던 데이터를 0(부정)과 1(긍정)으로 변경
- NaN이 포함된 항목 제거
- NaN이 포함되어 있던 카테고리의 경우 Object타입으로 변경되어 숫자형(Int(64))형으로 변경
- 타겟에 따라 Standard 및 MinMax 스케일링 진행
- 각 열별 결측치 비율 확인(결측치가 모두 없는 것 확인)

COVID-19 데이터 전처리 후기

신대현

kaggle을 통해서 전처리가 완료된 데이터일 지라도 사용자에 따라 다시 전처리과정을 해야한다는 것을 알게 되었다.

데이터셋에 담긴 의미를 찾고, 데이터를 처리 하는 과정 중 데이터 제거,추가,보류 등을 의논하며 혼자만의 생각이아닌 팀원들과 회의를 통한 여러가지 다른 방향성을 알 수 있는 좋은 계기가 되었다.

데이터 전처리에 따라 결과가 달라지는 것, 그리고 데이터 전처리의 중요성을 알게되었다.

전홍균

전처리가 완료된 데이터라도 필요한 주제에 대해 모델을 만들려면 다시 전처리를 해야 한다는 것을 알게 되었고 의료업계에서 사용되는 표기법인 임신에서 남자성별의 표기법, '9999-99-99' 등 의미에 대해서도 찾아보면서 어떤 내용을 포함하고 있는지 알아가는 과정이 흥미로웠다.

정승환

데이터 전처리 과정에 대해 어떤 방식으로 접근하는가에 따라 정제 과정이 달라질 수 있다는걸 알았고, 팀원들과 의견을 공유하여 공통된 방향으로 결과를 도출할 수 있었음.



머신러닝 : 공동 타겟 질환별 사망에 관한 연관성 조사

1

데이터 선정

기저질환, 심장질환, 만성질환에 따른 사망 연관성 조사

2

모델 개발

로지스틱 회귀, Random Forest Classifier 등 다양한 모델을 구현
모델 테스트 결과값 비교

3

성능 평가

recall 0.80 이상 목표

※ Accuracy보다 recall을 중요하게 한 이유 :

최우선으로 환자 안전 및 생명을 윤리적으로 고려하여

생존자와 사망자의 데이터불균형으로 인하여 recall(재현율)을 우선순위로 선정

4

결과

팀원별로 구현한 모델을 통해 각 질환 별 사망과의 연관성 도출

머신러닝 공동작업 : 타겟 선정 및 모델(1) 구현

신대현	전홍균	정승환
모델 : LogisticRegression	모델 : LogisticRegression	모델 : LogisticRegression
타겟 : 감염자 중 기저질환 환자의 사망 연관성	타겟 : 만성질환 환자와 사망의 연관성	타겟 : 심장 질환이 있는 환자들의 사망 확률 예측
<pre> from sklearn.model_selection import train_test_split from sklearn.preprocessing import StandardScaler from sklearn.linear_model import LogisticRegression from sklearn.metrics import accuracy_score, classification_report, roc_auc_score #0. 위의 확인자로 df.confirmed 범위를 좁혀놓은 상태 #1. 필요한 변수 설정 df_lungdisease = df.confirmed[["PNEUMONIA", "COVID"]] # 2. X(특징), y(타겟) 설정 # X = df["df_lungdisease"] #0인 열을 만들지않았으니 아래처럼 변수로 사용해야한다. X = df_lungdisease y = df.confirmed["DEATH"] # 타겟 (0=생존, 1=사망) # 3. 학습/테스트 데이터 분리 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42) # 4. 스케일링 (표준화) scaler = StandardScaler() X_train_scaled = scaler.fit_transform(X_train) X_test_scaled = scaler.transform(X_test) # 5. 모델 선택 및 학습 (Logistic Regression) # class_weight='balanced' 옵션을 주면 자동으로 데이터 불균형 반영 model = LogisticRegression() model.fit(X_train_scaled, y_train) # 6. 예측 y_pred = model.predict(X_test_scaled) y_proba = model.predict_proba(X_test_scaled)[:, 1] # 사망 확률 # 7. 평가 print("Accuracy:", accuracy_score(y_test, y_pred)) print("ROC-AUC:", roc_auc_score(y_test, y_proba)) print(classification_report(y_test, y_pred)) Accuracy: 0.8651207058430117 ROC-AUC: 0.8165908818946627 precision recall f1-score support 0 0.87 1.00 0.93 67115 1 0.62 0.04 0.07 10636 accuracy 0.75 0.52 0.87 77751 macro avg 0.75 0.50 0.7751 weighted avg 0.63 0.87 0.81 </pre>	<pre> # 2. 로지스틱 회귀 모델 학습 print("--- 2. 모델 학습 시작 ---") # 로지스틱 회귀 모델 생성 및 학습 log_model = LogisticRegression(random_state=42, class_weight='balanced') log_model.fit(X_train_scaled, y_train) print("모델 학습 완료!\n\n") --- 2. 모델 학습 시작 --- 모델 학습 완료! 성능 평가 # 3. 모델 성능 평가 print("--- 3. 모델 평가 시작 ---") y_pred = log_model.predict(X_test_scaled) # 정확도 accuracy = accuracy_score(y_test, y_pred) print(f"✅ 정확도 (Accuracy): {accuracy:.4f}\n\n") # 혼동 행렬 print(f"📊 혼동 행렬 (Confusion Matrix):") conf_matrix = confusion_matrix(y_test, y_pred) print(conf_matrix) print("\n") # 분류 리포트 print(f"📋 분류 리포트 (Classification Report):") class_report = classification_report(y_test, y_pred, target_names=['생존(0)', '사망(1)']) print(class_report) --- 3. 모델 평가 시작 --- ✅ 정확도 (Accuracy): 0.8049 📊 혼동 행렬 (Confusion Matrix): [[58220 8788] [6346 4228]] 📋 분류 리포트 (Classification Report): precision recall f1-score support 생존(0) 0.90 0.87 0.88 67008 사망(1) 0.32 0.40 0.36 10574 accuracy 0.80 0.80 0.80 77582 macro avg 0.61 0.63 0.62 77582 weighted avg 0.62 0.90 0.61 77582 </pre>	<pre> from sklearn.model_selection import train_test_split import pandas as pd # 독립 변수(features) 지정 X = df_undersampled[["SEX", "AGE", "CARDIOVASCULAR", "HIPERTENSION", "IQ"]] # 종속 변수(target) 지정 y = df_undersampled["DEATH"] # 훈련 및 테스트 세트 분할 (예: 8:2 비율) X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42) # 훈련 세트를 다시 훈련 및 검증 세트로 분할 X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.25, random_state=42) print(f"훈련 세트 크기: {len(X_train)}") print(f"검증 세트 크기: {len(X_val)}") print(f"테스트 세트 크기: {len(X_test)}") from sklearn.model_selection import train_test_split from sklearn.linear_model import LogisticRegression from sklearn.metrics import accuracy_score import pandas as pd # 모델 객체 생성 model = LogisticRegression() # 훈련 데이터로 모델 학습 model.fit(X_train, y_train) print("로지스틱 회귀 모델 학습 완료") # 테스트 데이터로 예측 수행 predictions = model.predict(X_test) # 정확도(Accuracy) 계산 accuracy = accuracy_score(y_test, predictions) print(f"📊 모델의 예측 정확도: {accuracy:.4f}") </pre>
특이사항 precision(정확성)은 높으나 recall(낮은 재현율) 확인 데이터 불균형 확인	특이사항 LogisticRegeression 생존에 대한 정확도는 높으나 사망에 대한 정확도가 낮고, 재현율도 비슷한 결과가 남음.	특이사항 정확도 및 재현율이 낮은 결과가 나왔고 그에 대해 데이터 불균형이 원인으로 추정

신대현	전홍균	정승환
모델 : RandomForestClassifier	모델 : LogisticRegression	모델 : LogisticRegression
수정 : pipeline구축, class_weight='balanced' 적용	수정 : 사망과 연관이 높은 폐렴과 나이 특성 추가	수정 : 데이터 언더샘플링 및 독립 변수 추가
<pre> # 0. 위의 확진자로 df_confirmed 범위를 좁혀놓은 상태 # 1. 폐질환 변수 설정 df_lungdisease = df_confirmed[["PNEUMONIA", "COPOD", "ASTHMA", "TOBACCO"]] # 2. X(특징), y(타겟) 설정 X = df_lungdisease y = df_confirmed["DEATH"] # 타겟 (0=생존, 1=사망) # 3. 학습/테스트 데이터 분리 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42) # 4. 파이프라인 구축 (스케일링 + 랜덤 포레스트) pipeline = Pipeline([('scaler', StandardScaler()), # 첫 번째 단계: 스케일링 ('classifier', RandomForestClassifier(n_estimators=100, # 트리 개수 (기본값은 100) max_depth=None, # 트리의 최대 깊이 (None은 완전히 성장할 때까지) min_samples_split=2, # 노드 분할에 필요한 최소 샘플 수 min_samples_leaf=1, # 리프 노드에 필요한 최소 샘플 수 class_weight='balanced', # 클래스스 가중치 (불균형 데이터 처리) random_state=42 # 재현성을 위한 랜덤 시드))]) # 5. 파이프라인 학습 pipeline.fit(X_train, y_train) # 6. 예측 y_pred = pipeline.predict(X_test) y_proba = pipeline.predict_proba(X_test)[:, 1] # 사망 확률 # 7. 평가 print("Accuracy:", accuracy_score(y_test, y_pred)) print("ROC-AUC:", roc_auc_score(y_test, y_proba)) print(classification_report(y_test, y_pred)) Accuracy: 0.8475646615477614 ROC-AUC: 0.8183876013724962 precision recall f1-score support 0 0.96 0.86 0.91 67115 1 0.47 0.77 0.58 10636 accuracy 0.71 0.81 0.85 77751 macro avg 0.71 0.74 0.75 77751 weighted avg 0.89 0.85 0.86 77751 </pre>	<pre> # 3. 모델 성능 평가 print("=== 3. 모델 평가 시작 ===") y_pred = log_model.predict(X_test_scaled) # 정확도 accuracy = accuracy_score(y_test, y_pred) print(f"✅ 정확도 (Accuracy): {accuracy:.4f}\n") # 혼동 행렬 print(f"✅ 혼동 행렬 (Confusion Matrix):") conf_matrix = confusion_matrix(y_test, y_pred) print(conf_matrix) print("\n") # 분류 리포트 print(f"✅ 분류 리포트 (Classification Report):") class_report = classification_report(y_test, y_pred, target_names=['생존(0)', '사망(1)']) print(class_report) --- 3. 모델 평가 시작 --- ✅ 정확도 (Accuracy): 0.8376 ✅ 혼동 행렬 (Confusion Matrix): [[56212 10796] [1807 8767]] ✅ 분류 리포트 (Classification Report): precision recall f1-score support 생존(0) 0.97 0.84 0.90 67008 사망(1) 0.45 0.83 0.58 10574 accuracy 0.84 0.84 0.84 77582 macro avg 0.71 0.83 0.74 77582 weighted avg 0.90 0.84 0.86 77582 </pre>	<pre> from sklearn.utils import resample import pandas as pd df_confirmed = df[df["CLASSIFICATION_FINAL"] == 1] # 1. 클래스 분리 df_majority = df_confirmed[df_confirmed['DEATH'] == 0] # 생존자 그룹 df_minority = df_confirmed[df_confirmed['DEATH'] == 1] # 사망자 그룹 # 2. 생존자 데이터 샘플링 df_majority_undersampled = resample(df_majority, replace=False, # 독립 추출(False) n_samples=len(df_minority), # 사망자 수만큼 샘플링 random_state=42) # 재현성을 위한 시드값 # 3. 데이터 병합 df_undersampled = pd.concat([df_majority_undersampled, df_minority]) # 결과 확인 print("언더샘플링 후 데이터 크기:") print(df_undersampled['DEATH'].value_counts()) 언더샘플링 후 데이터 크기: DEATH 0 63046 1 63046 Name: count, dtype: int64 from sklearn.metrics import classification_report predictions = model.predict(X_test) print(classification_report(y_test, predictions)) precision recall f1-score support 0 0.87 0.84 0.85 10712 1 0.84 0.87 0.85 10507 accuracy 0.85 0.85 0.85 21219 macro avg 0.85 0.85 0.85 21219 weighted avg 0.85 0.85 0.85 21219 </pre>
특이사항 recall(낮은 재현율)을 개선, precision(정확성) 부족 데이터 불균형 부분 해결	특이사항 낮은 재현율을 개선하기 위해 사망과 연관성이 높은 폐렴과 나이 특성을 추가하여 재현율을 목표치까지 높임.	특이사항 언더샘플링을 통해 데이터 불균형 해결 및 상관관계가 더 높 은 독립 변수 추가하여 정확도 및 재현율 상승

프로젝트 시각화 및 결과

의료 데이터는 생존자 및 사망자와 같은 데이터 불균형이 상당히 심한 경우가 흔합니다.

사람의 생명과 직결될 수 있기 때문에 일반적인 데이터와는 매우 다른, 훨씬 엄격한 기준과 철학을 가지고 접근하게 됩니다.

최우선으로 환자 안전 및 생명을 윤리적으로 고려 하며, 의료 분야에서는 '위음성(False Negative)'을 줄이는 것이 '위양성(False Positive)'을 줄이는 것보다 훨씬 중요할 수 있습니다.

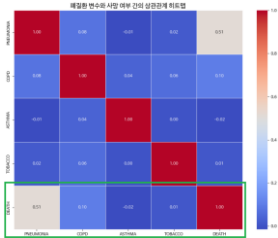
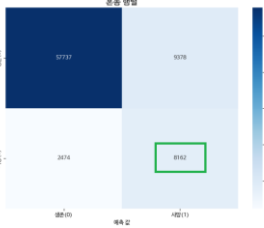

다시말해, 실제 환자인데 아니라고 예측하는 것이 실제 환자가 아닌데 환자라고 예측하는 것보다 더 중요합니다.

따라서 재현율(Recall)을 매우 중요하게 여기며, 경우에 따라 정밀도(Precision)를 다소 희생하기도 합니다.

위와같은 내용을 바탕으로 결과물을 산출하였습니다.



공동작업 : 타겟 선정 및 모델(1) 결과 및 시각화 자료

	타겟	시각화 자료	시각화 자료	최종결과
신대현	코로나 환자 중 기저질환 환자의 사망 확률 예측			recall 0.77 폐 관련한 사망자 중 COPR(기존 폐질환 환자), PNEUMONIA(폐렴) 환자의 높은 사망률 확인
전홍균	만성질환을 가진 코로나 환자의 사망률 예측			recall 0.83 만성질환을 가진 확진자는 사망에 대한 연관성이 매우 적음으로 폐렴을 가진 코로나 확진자 만큼의 연관성은 없는 것으로 확인.
정승환	심장 질환이 있는 환자들 의 사망 확률 예측			recall 0.87 심장 질환 변수만으로는 사망 예측이 힘들었고 다른 변수(나이, 응급실 등)가 사망과 더 상관관계가 있다는 것을 확인

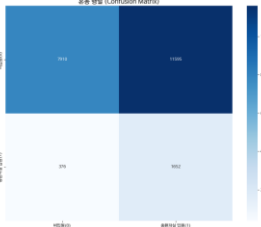
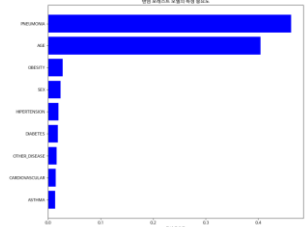

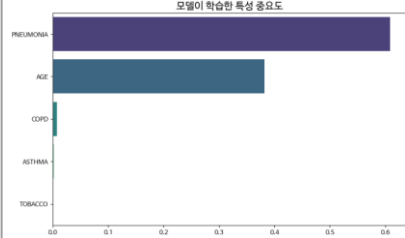
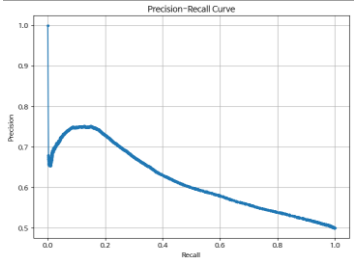
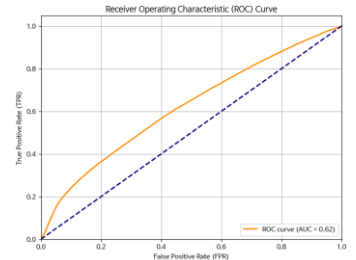
머신러닝 : 개인 타겟

머신러닝 개인작업 : 타겟 선정 및 모델(2) 구현

신대현	전홍균	정승환
모델 : LogisticRegression	모델 : RandomForestClassifier	모델 : RandomForestClassifier
타겟 : 확진자 중 중환자실 입실 여부 예측,특징 파악	타겟 : 폐 질환 환자의 사망률 예측	타겟 : 합병증이 있는 사람들의 코로나 확진 확률 예측
<pre>from sklearn.model_selection import train_test_split from sklearn.preprocessing import StandardScaler from sklearn.linear_model import LogisticRegression from sklearn.metrics import accuracy_score, classification_report, roc_auc_score #0. 위와 확진자로 df.confirmed 범위를 중환자실 상태 #1. 불확한 변수 설정 df_lungdisease = df.confirmed[['PNEUMONIA',"COPO"]] # 2. X(특징), y(타겟) 설정 # X = df[['df_lungdisease']] #이런 열을 만들지않았으니 아래처럼 변수로 사용해야한다. X = df_lungdisease y = df.confirmed["DEATH"] # 타겟 (0=생존, 1=사망) # 3. 학습/테스트 데이터 분리 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42) # 4. 스케일링 (표준화) scaler = StandardScaler() X_train_scaled = scaler.fit_transform(X_train) X_test_scaled = scaler.transform(X_test) # 5. 모델 선택 및 학습 (Logistic Regression) # class_weight='balanced' 옵션을 주면 자동으로 데이터 불균형 반영 model = LogisticRegression() model.fit(X_train_scaled, y_train) # 6. 예측 y_pred = model.predict(X_test_scaled) y_proba = model.predict_proba(X_test_scaled)[: , 1] # 사망 확률 # 7. 평가 print("Accuracy:", accuracy_score(y_test, y_pred)) print("ROC-AUC:", roc_auc_score(y_test, y_proba)) print(classification_report(y_test, y_pred)) Accuracy: 0.8651207058430117 ROC-AUC: 0.81659308818946627 precision recall f1-score support 0 0.87 1.00 0.93 67115 1 0.62 0.04 0.07 10636 accuracy 0.87 77751 macro avg 0.75 0.52 0.50 77751 weighted avg 0.83 0.87 0.81 77751</pre>	<pre># ===== # 3. 모델 학습 및 평가 # ===== print("--- 모델 학습 및 평가 시작 ---") # 확진자 데이터 필터링 및 특성(X)과 타겟(y) 정의 df_analysis = df[df['CLASSIFICATION_FINAL'] == 1].copy() features = ['PNEUMONIA', 'COPO', 'TOBACCO', 'ASTHMA', 'AGE'] X = df_analysis[features] y = df_analysis['DEATH'] # 데이터 분리 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y) # SMOTE로 학습 데이터 증강 smote = SMOTE(random_state=42) X_train_over, y_train_over = smote.fit_resample(X_train, y_train) # RandomForest 모델 생성 및 학습 rf_model = RandomForestClassifier(n_estimators=100, random_state=42, n_jobs=-1) rf_model.fit(X_train_over, y_train_over) # 예측 및 성능 평가 y_pred = rf_model.predict(X_test) print("✅ 최종 모델 분류 리포트:") print(classification_report(y_test, y_pred)) --- 모델 학습 및 평가 시작 --- ✅ 최종 모델 분류 리포트: precision recall f1-score support 0 0.91 0.76 0.83 12643 1 0.61 0.83 0.70 5665 accuracy 0.78 18308 macro avg 0.76 0.80 0.77 18308 weighted avg 0.62 0.78 0.79 18308</pre>	<pre>from imblearn.over_sampling import SMOTE import pandas as pd from collections import Counter # 독립 변수(features) 지정 X = df[['DIABETES', 'COPO', 'ASTHMA', 'HYPERTENSION', 'CARDIOVASCULAR', 'OBESITY', 'RENAL_CHRONIC', 'TOBACCO', 'PNEUMONIA']] # X의 데이터 타입을 float로 변환 (SMOTE 오류 방지) X = X.astype(float) # 종속 변수(target) 지정 y = df['CLASSIFICATION_FINAL'] print("SMOTE 적용 전 데이터 크기:") print(Counter(y)) # SMOTE 객체 생성 및 적용 smote = SMOTE(random_state=42) X_resampled, y_resampled = smote.fit_resample(X, y) print("SMOTE 적용 후 데이터 크기:") print(Counter(y_resampled)) SMOTE 적용 전 데이터 크기: Counter({0: 635966, 1: 388751}) 랜덤 포레스트 모델 성능: precision recall f1-score support 0 0.56 0.72 0.63 77638 1 0.62 0.44 0.51 77863 accuracy 0.58 155501 macro avg 0.59 0.58 0.57 155501 weighted avg 0.59 0.58 0.57 155501 모델의 예측 정확도: 0.5927</pre>
특이사항 높은 precision(정확성) 낮은 recall(재현율) 확인 데이터 불균형 확인	특이사항 LogisticRegression 모델보다 RandomForest 모델이 더 적합하다고 판단하여 모델을 교체함.	특이사항 데이터 불균형을 오버 샘플링으로 시도하였으나 결과가 매우 낮음

신대현	전흥균	정승환
모델 : RandomForestClassifier	모델 : RandomForestClassifier	모델 : LogisticRegression
수정 : pipeline구축, class_weight='balanced' 적용	수정 : 없음	수정 : 언더샘플링 및 모델 변경
<pre> import numpy as np import pandas as pd from sklearn.model_selection import train_test_split from sklearn.preprocessing import StandardScaler from sklearn.ensemble import RandomForestClassifier from sklearn.metrics import accuracy_score, classification_report, roc_auc_score, recall_score, precision_score from sklearn.pipeline import Pipeline # 보르나 확진자의 데이터 (CLASSIFICATION_FINAL) 1인 경우가 확진자라고 가정) df_train_final = df[df['CLASSIFICATION_FINAL'] == 1].copy() # 2. X(특성), y(결과) 분리 y = df_train_final['DEATH'] features = ['AGE', 'SEX', 'PNEUMONIA', 'DIABETES', 'ASTHMA', 'HYPERTENSION', 'CHRONIC_KIDNEY_DISEASE', 'CARDIOVASCULAR', 'OBESITY'] X = df_train_final[features] # 3. 학습/테스트 데이터 분리 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42) # y_train과 y_test의 데이터 인덱스를 메모 명확히 저장 y_train = y_train.astype(int) y_test = y_test.astype(int) # 4. 데이터의 구조 (스케일링 + 모델 포리스트) pipeline = Pipeline([('scaler', StandardScaler()), ('classifier', RandomForestClassifier(n_estimators=100, max_depth=12, min_samples_leaf=5, class_weight='balanced', # 불균형 데이터 처리를 위해 class_weight는 무지 random_state=42))]) # 5. 데이터에 대한 학습 pipeline.fit(X_train, y_train) # 6. 예측 (입력한 조건을 위한 예측 결과) y_proba = pipeline.predict_proba(X_test)[1, :] # 7. 임계값 조정 : 0.5에서 0.45로 약간 낮춰 재현율을 조금 더 끌어올렸나보다. threshold = 0.45 y_pred = y_proba > threshold).astype(int) # 8. 결과 출력 print("""보르나 확진자의 대상으로 한 중환자실 입원 예측 모델 결과 """) print("Accuracy: ", accuracy_score(y_test, y_pred)) print("Precision (중환자실 Class 1): ", precision_score(y_test, y_pred)) print("Recall (중환자실 Class 1): ", recall_score(y_test, y_pred)) print("F1-Score: ", f1_score(y_test, y_pred)) print("Classification Report(y_test, y_pred)") # 9. 특성 중요도 확인 feature_importances = pipeline.named_steps['classifier'].feature_importances_ for feature, importance in zip(features, feature_importances): print(f"feature: {importance:.4f}") --- 보르나 확진자의 대상으로 한 중환자실 입원 예측 모델 결과 --- Accuracy: 0.44462507405062506 Precision (중환자실 Class 1): 0.5247974900000004 Recall (중환자실 Class 1): 0.4656667456567 F1-Score: 0.49599742421897 precision recall f1-score support accuracy 0.76 0.80 0.78 18308 macro avg 0.54 0.59 0.56 18308 weighted avg 0.60 0.64 0.62 18308 </pre>	<pre> ##### # 3. 모델 학습 및 평가 ##### print("""--- 모델 학습 및 평가 시작 ---""") # 확진자 데이터 필터링 및 특성(X)과 타겟(y) 정의 df_analysis = df[df['CLASSIFICATION_FINAL'] == 1].copy() features = ['PNEUMONIA', 'COPD', 'TOBACCO', 'ASTHMA', 'AGE'] X = df_analysis[features] y = df_analysis['DEATH'] # 데이터 분리 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y) # SMOTE로 학습 데이터 증강 smote = SMOTE(random_state=42) X_train_over, y_train_over = smote.fit_resample(X_train, y_train) # RandomForest 모델 생성 및 학습 rf_model = RandomForestClassifier(n_estimators=100, random_state=42, n_jobs=-1) rf_model.fit(X_train_over, y_train_over) # 예측 및 성능 평가 y_pred = rf_model.predict(X_test) print("""최종 모델 분류 리포트:"") print(classification_report(y_test, y_pred)) --- 모델 학습 및 평가 시작 --- ✓ 최종 모델 분류 리포트: precision recall f1-score support accuracy 0.91 0.76 0.83 12643 macro avg 0.61 0.61 0.61 12643 weighted avg 0.82 0.78 0.79 18308 </pre>	<pre> from imblearn.under_sampling import RandomUnderSampler import pandas as pd from collections import Counter # 독립 변수(features) 지정 X = df[['DIABETES', 'COPD', 'ASTHMA', 'HYPERTENSION', 'CARDIOVASCULAR', 'OBESITY', 'RENAL_CHRONIC', 'TOBACCO', 'PNEUMONIA', 'AGE', 'SEX']] # 종속 변수(target) 지정 y = df['CLASSIFICATION_FINAL'] print("언더샘플링 적용 전 데이터 크기:") print(Counter(y)) # RandomUnderSampler 객체 생성 및 적용 rus = RandomUnderSampler(random_state=42) X_resampled, y_resampled = rus.fit_resample(X, y) print("\n언더샘플링 적용 후 데이터 크기:") print(Counter(y_resampled)) 언더샘플링 적용 전 데이터 크기: Counter({0: 63956, 1: 38875}) 언더샘플링 적용 후 데이터 크기: Counter({0: 38875, 1: 38875}) precision recall f1-score support accuracy 0.57 0.71 0.63 77638 macro avg 0.62 0.46 0.53 77663 weighted avg 0.59 0.59 0.59 155501 </pre>
특이사항 recall(낮은 재현율)을 개선, precision(정확성) 부족 데이터 불균형 부분 해결	특이사항 크게 3분류로 나누었던 질환 특성을 모두 적용시켜 어떤 질환이 가장 사망과 연관이 있는지 확인 함.	특이사항 언더샘플링 및 모델 변경하였으나 이전 결과와 크게 다르지 않았고 코로나 확진과 질병 카테고리와의 상관관계가 크게 없었음을 확인

개인작업 : 타겟 선정 및 모델(2) 결과 및 시각화 자료

	타겟	시각화 자료	시각화 자료	최종결과
신대현	확진자 중 중환자실 입실 여부 예측, 특징 파악			recall 0.81 폐렴(PNEUMONIA)과 나 이(AGE)가 중환자실 입원 예측에 가장 중요한 특징 확인
전홍균	폐질환 환자의 사망률 연 관성 예측			recall 0.83 폐렴과 나이를 제외한 나 머지 항목은 연관성이 거 의 없다는 것을 확인 함.
정승환	합병증이 있는 사람들의 코로나 확진 확률 예측			recall 0.46 코로나 확진과 질병간의 상관관계가 크지 않은 것 을 재현율 및 ROC 그래프 수치로 확인

머신러닝 개인작업 : 타겟 선정 및 모델(3) 구현

신대현	전홍균	정승환
모델 : LogisticRegression	모델 : RandomForestClassifier	모델 : RandomForestClassifier
타겟 : 코로나 감염 여부 예측,감염 위험군 특정	타겟 : 다양한 질환과 사망의 연관성 예측	타겟 : 코로나 확진자 중 인공호흡기 사용 예측
<pre>import numpy as np from sklearn.model_selection import train_test_split from sklearn.preprocessing import StandardScaler from sklearn.linear_model import LogisticRegression from sklearn.metrics import accuracy_score, classification_report, roc_auc_score # 2. X(특징), y(타겟) 설정 y = df['CLASSIFICATION_FINAL'] == 1 # 코로나 확진 여부 (1=확진, 0=비확진) # CLASSIFICATION_FINAL을 제외한 특징들만 사용 features = ['AGE', 'SEX', 'PNEUMONIA', 'DIABETES', 'ASTHMA', 'HIPERTENSION', 'OTHER_DISEASE', 'CARDIOVASCULAR', 'OBESITY', 'COPD', 'RENAL_CHRONIC', 'TOBACCO'] X = df[features].values # 3. 학습,테스트 분리 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=50) # 4. 스케일링(표준화) scaler = StandardScaler() X_train_scaled = scaler.fit_transform(X_train) X_test_scaled = scaler.transform(X_test) # 5. 모델 model = LogisticRegression() model.fit(X_train_scaled, y_train) # 6. 예측 y_pred = model.predict(X_test_scaled) y_proba = model.predict_proba(X_test_scaled)[:, 1] # 7. 평가 print("Accuracy:", accuracy_score(y_test, y_pred)) print("ROC-AUC:", roc_auc_score(y_test, y_proba)) print(classification_report(y_test, y_pred)) Accuracy: 0.6293776903863527 ROC-AUC: 0.6444143734538135 precision recall f1-score support False 0.61 0.39 0.47 16239 True 0.64 0.81 0.71 21395 accuracy 0.62 macro avg 0.60 weighted avg 0.63 0.61 0.63 37634</pre>	<pre>##### # 3. 모델 학습 및 평가 ##### # 확진자 데이터 필터링 및 특성(이과 타겟(y) 제외) df_analysis = df[df['CLASSIFICATION_FINAL'] == 1].copy() features = ['OBESITY', 'RENAL_CHRONIC', 'DIABETES', 'AGE', 'PNEUMONIA', 'COPD', 'ASTHMA', 'LUNG_DISEASE', 'HIPERTENSION', 'CARDIOVASCULAR', 'OTHER_DISEASE', 'TOBACCO'] X = df_analysis[features] y = df_analysis['DEATH'] # 데이터 분리 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y) # SMOTE를 학습 데이터 공간 smote = SMOTE(random_state=42) X_train_lower, y_train_lower = smote.fit_resample(X_train, y_train) # RandomForest 모델 생성 및 학습 rf_model = RandomForestClassifier(n_estimators=100, random_state=42, n_jobs=-1) rf_model.fit(X_train_lower, y_train_lower) # 예측 및 성능 평가 y_pred = rf_model.predict(X_test) print("전통 기법으로") print(classification_report(y_test, y_pred)) ##### 참종 지표도: precision recall f1-score support 0 0.97 0.82 0.89 67008 1 0.41 0.82 0.55 10574 accuracy 0.69 macro avg 0.69 0.82 0.72 77592 weighted avg 0.69 0.82 0.84 77592</pre>	<pre>import pandas as pd from sklearn.model_selection import train_test_split from imblearn.over_sampling import SMOTE from collections import Counter from sklearn.ensemble import RandomForestClassifier from sklearn.metrics import classification_report # 'CLASSIFICATION_FINAL'이 1인(코로나 확진자) 데이터만 필터링 df_filtered = df[(df['CLASSIFICATION_FINAL'] == 1) & (df['INTUBED'] != -1)] # 독립 변수(features) 지정 # INTUBED와 상관관계가 높은 변수를 선택 X = df_filtered[['PNEUMONIA', 'AGE', 'HIPERTENSION', 'DIABETES', 'DEATH']] # 종속 변수(target) 지정 y = df_filtered['INTUBED'] ##### print("SMOTE 적용 전 데이터 크기:") print(Counter(y)) # SMOTE 객체 생성 및 적용 # INTUBED 변수도 불균형이 심할 수 있으므로 SMOTE 적용 smote = SMOTE(random_state=42) X_resampled, y_resampled = smote.fit_resample(X, y) print("\nSMOTE 적용 후 데이터 크기:") print(Counter(y_resampled)) SMOTE 적용 전 데이터 크기: Counter({0: 84794, 1: 23300}) SMOTE 적용 후 데이터 크기: Counter({1: 84794, 0: 84794})</pre>
특이사항 precision(정확성), recall(재현율) 확인 데이터 불균형 확인	특이사항 크게 3분류로 나누었던 질환 특성을 모두 적용시켜 어떤 질환이 가장 사망과 연관이 있는지 확인 함.	특이사항 데이터 불균형을 오버 샘플링으로 수정

신대현	전홍균	정승환
모델 : RandomForestClassifier	모델 : RandomForestClassifier	모델 : RandomForestClassifier
수정 : pipeline구축, class_weight='balanced' 적용	수정 : 없음	수정 : 오버샘플링 수정
<pre> import numpy as np import pandas as pd from sklearn.model_selection import train_test_split from sklearn.preprocessing import StandardScaler from sklearn.ensemble import RandomForestClassifier from sklearn.metrics import accuracy_score, classification_report, roc_auc_score from sklearn.pipeline import Pipeline # 2. X(특징), y(타겟) 설정 y = df["CLASSIFICATION_FINAL"] == 1 # 코로나 확진 여부 (True=확진, False=비확진) # 가져올항까지 모두 포함한 특징 리스트 features = ['AGE', 'SEX', 'PNEUMONIA', 'DIABETES', 'ASTHMA', 'HYPERTENSION', 'OTHERDISEASE', 'CARDIOVASCULAR', 'OBESITY', 'COPD', 'RENALCHRONIC', 'TOBACCO'] X = df[features] # 3. 학습/테스트 데이터 분리 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=50) # 4. 파이프라인 구축 (스케일링 + 랜덤 포레스트) pipeline = Pipeline([('scaler', StandardScaler()), # 첫 번째 단계: 표준화 ('classifier', RandomForestClassifier(n_estimators=100, # 트리 개수 criterion='entropy', # 불순도 측정 max_depth=10, # 트리의 최대 깊이 max_features='sqrt', # 매 분기마다 고려할 특성 수 min_samples_leaf=4, # 리프 노드의 최소 샘플 수 class_weight='balanced', # 클래스 불균형 처리 oob_score=True, # Out-of-bag 샘플을 이용한 성능 평가 random_state=50 # 재현성을 위한 랜덤 시드))]) # 5. 파이프라인 학습 pipeline.fit(X_train, y_train) # 6. 예측 y_pred = pipeline.predict(X_test) y_proba = pipeline.predict_proba(X_test)[:, 1] # 코로나 양성 확률 # 7. 평가 print("Accuracy:", accuracy_score(y_test, y_pred)) print("ROC-AUC:", roc_auc_score(y_test, y_proba)) print("OOB Score:", pipeline.named_steps['classifier'].oob_score_) # Out-of-bag 값수 print(classification_report(y_test, y_pred)) --- 랜덤 포레스트 파이프라인 모델 결과 --- Accuracy: 0.6210873146622795 ROC-AUC: 0.6277707159623689 OOB Score: 0.6294554779050302 precision recall f1-score support False 0.56 0.56 0.56 16239 True 0.67 0.67 0.67 17395 accuracy 0.61 0.61 0.61 37634 macro avg 0.61 0.61 0.61 37634 weighted avg 0.62 0.62 0.62 37634 </pre>	<pre> # 3. 모델 학습 및 평가 # ===== # 특징자 데이터 불균형 및 특성(1)과 타겟(y) 평가 dt_analysis = dt[df["CLASSIFICATION_FINAL"] == 1].copy() features = ['OBSERV', 'RENALCHRONIC', 'DIABETES', 'AGE', 'PNEUMONIA', 'COPD', 'ASTHMA', 'HYPERTENSION', 'CARDIOVASCULAR', 'OTHERDISEASE', 'TOBACCO'] X = dt_analysis[features] y = dt_analysis['DEATH'] # 데이터 분리 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y) # SMOTE 학습 데이터 분할 smote = SMOTE(random_state=42) X_train_sm, y_train_sm, y_test = smote.fit_resample(X_train, y_train) # RandomForest 모델 생성 및 학습 rf_model = RandomForestClassifier(n_estimators=100, random_state=42, n_jobs=-1) rf_model.fit(X_train_sm, y_train_sm) # 예측 및 성능 평가 y_pred = rf_model.predict(X_test) print("전통 리포트") print(classification_report(y_test, y_pred)) 전통 리포트: precision recall f1-score support 0 0.97 0.92 0.96 6709 1 0.41 0.92 0.55 1074 accuracy 0.69 0.92 0.82 7782 macro avg 0.69 0.92 0.72 7782 weighted avg 0.69 0.92 0.84 7782 </pre>	<pre> # 훈련 및 테스트 세트 분할 X_train, X_test, y_train, y_test = train_test_split(X_resampled, y_resampled, test_size=0.2, random_state=42) # RandomForestClassifier 모델 객체 생성 및 학습 model = RandomForestClassifier(n_estimators=100, random_state=42) model.fit(X_train, y_train) # 테스트 데이터로 예측 및 성능 평가 predictions = model.predict(X_test) print("\n랜덤 포레스트 모델 성능:") print(classification_report(y_test, predictions)) 랜덤 포레스트 모델 성능: precision recall f1-score support 0 0.81 0.69 0.75 16951 1 0.73 0.84 0.78 16967 accuracy 0.77 0.77 0.77 33918 macro avg 0.77 0.77 0.77 33918 weighted avg 0.77 0.77 0.77 33918 </pre>
특이사항 precision(정확성), recall(재현율) 개선 데이터 불균형 부분해결	특이사항 크게 3분류로 나누었던 질환 특성을 모두 적용시켜 어떤 질환이 가장 사망과 연관이 있는지 확인 함.	특이사항 코로나 확진자와 인공호흡기 간 상관관계가 매우 높았고 오버샘플링을 통해 데이터 불균형을 해결했지만 실제 의료 데이터에서는 자주 사용하지 않는다고 확인

개인작업 : 타겟 선정 및 모델(3) 결과 및 시각화 자료

	타겟	시각화 자료	시각화 자료	최종결과
신대현	환자들의 임상적 특징을 바탕으로 코로나19 감염 여부를 예측			recall 0.67 나이(AGE)와 폐렴(PNEUMONIA)이 코로나19 감염 여부를 예측하는데 가장 큰 영향력을 보이는 것을 확인
전홍균	다른 질환을 가진 코로나 확진자의 사망 연관성 예측			recall 0.82 코로나 환자 중 (폐렴)이 가장 연관성이 높은 것으로 나타났으며, 다른 질환에 대한 연관성은 매우 낮다는 것을 확인.
정승환	코로나 확진자 중 인공호흡기 사용 예측			recall 0.84 특성 중요도 그래프를 통해 죽음이 매우 높은 상관관계를 보이는 것을 확인

COVID-19 머신러닝 후기

신대현

의료데이터 특징상 데이터 불균형이 심해 이런부분을 어떻게 해결하는지 머신러닝을 구현하며 이런 데이터불균형에 관해서 깊게 공부할 수 있는점이 좋았다.

원했던 결과보다 이상한 결과가 나와 정확성을 중요시했던 기존의 편견이 의료적으로는 재현율을 중요하게 봐야한다는 시점으로 봐야한다는 것도 알게 되었다.

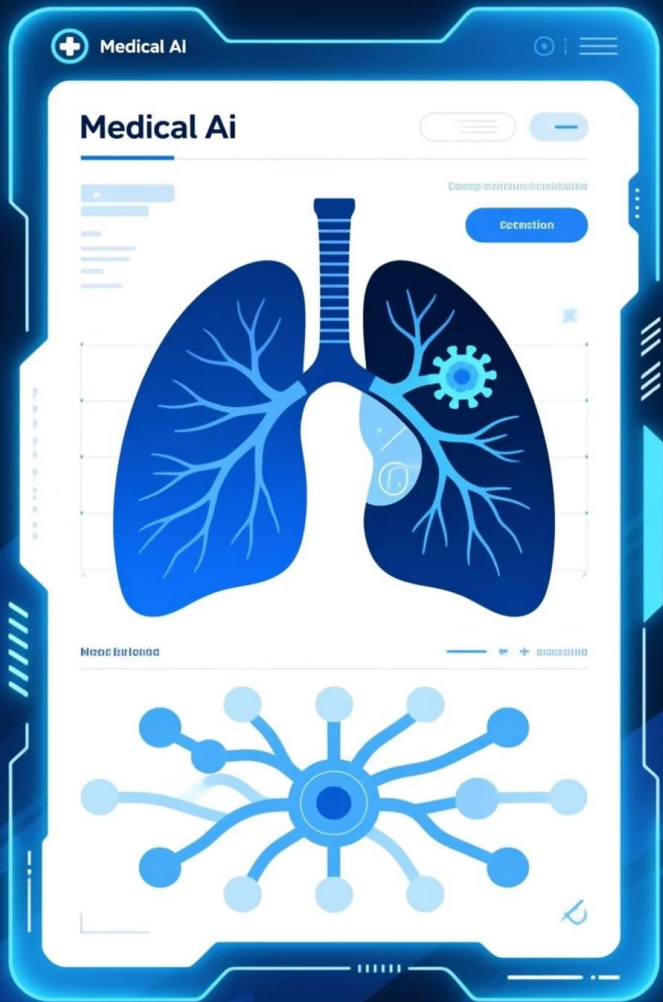
내가 예측하거나 조사하는 것이 단순 정확성을 추출하는 것이 아닌 어떤 시점으로 타겟에 어떤 어떻게 접근해야 할지를 알 수 있게 된 계기였다.

전홍균

처음에는 만성질환과 관련된 3개의 특성만으로 시작했지만 만족스러운 결과를 얻지 못했다. 하지만 'AGE', 'PNEUMONIA'와 같이 영향력 있는 특성을 추가하고, 모델을 바꾸는 등 반복적인 실험을 통해 재현율을 4%에서 86%까지 끌어올릴 수 있었다. 이 과정은 데이터 분석이 정해진 답을 찾는 것이 아니라, 가설을 세우고, 실험하고, 결과를 분석하며 점진적으로 최적의 해답을 찾아가는 과학적인 탐구 과정과 같다는 것을 깨닫게 해주었다.

정승환

데이터 간 상관관계를 확인하는 부분이 흥미로웠고 독립 변수를 변경하거나 데이터 불균형을 해결하는 방법 등으로 모델의 정확도나 재현율을 변경해가는 과정에서 한층 더 머신러닝에 대해 이해할 수 있었음



딥러닝 : COVID-19 환자의 흉부X-ray 분석



X-ray 사진 진단

COVID-19 양성(positive)과 음성(negative) 이진분류
90% 이상 정확도 목표(accuracy)

Densenet121, GRAD-CAM 를 활용한 모델 개발

딥러닝 - 예측모델(Densenet121)

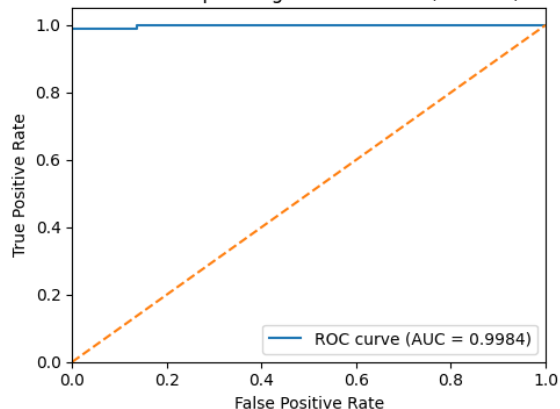
코로나 환자의 흉부 X-ray를 분석,조사하여 양성,음성을 분류하는 모델 구현

Classification Report (Test Set)

	precision	recall	f1-score	support
Covid-19	0.9865	1.0000	0.9932	73
Normal	1.0000	0.9884	0.9942	86
accuracy			0.9937	159
macro avg	0.9932	0.9942	0.9937	159
weighted avg	0.9938	0.9937	0.9937	159

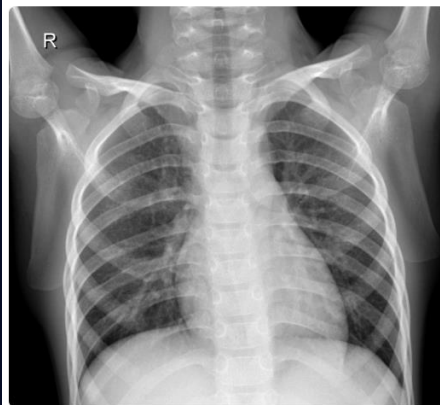
ROC AUC (Test Set): 0.9984

Receiver Operating Characteristic (Test Set)



분석 결과

Negative일 확률이 99.53%입니다.



원본 이미지



Grad-CAM 분석 결과

Grad-CAM 해석:

- 붉은색 영역은 모델이 'Positive'라고 판단하는 데 가장 큰 영향을 미친 부분입니다.
- 푸른색 영역은 판단에 거의 영향을 미치지 않은 부분입니다.
- 이 시각화는 모델의 판단을 해석하는 데 도움을 주지만, 100% 정확한 의학적 진단을 의미하지는 않습니다.

음성(negative)

(1)



(2)



(3)

분석 결과

Negative일 확률이 99.82% 입니다.



원본 이미지



Grad-CAM 분석 결과

(5)

(4)

분석 결과

Negative일 확률이 98.80% 입니다.



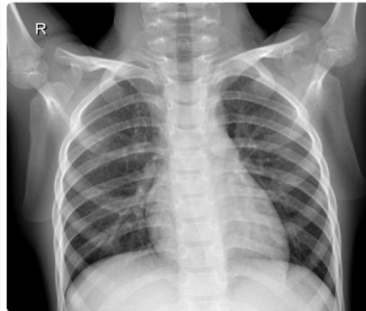
원본 이미지



Grad-CAM 분석 결과

분석 결과

Negative일 확률이 99.53% 입니다.



원본 이미지



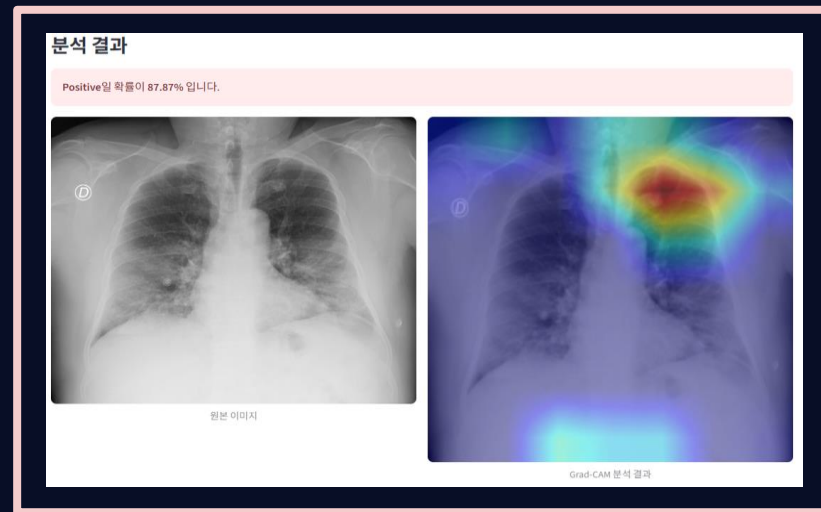
Grad-CAM 분석 결과

양성(positive)

(1)



(2)



(3)

분석 결과

Positive일 확률이 94.05% 입니다.



원본 이미지

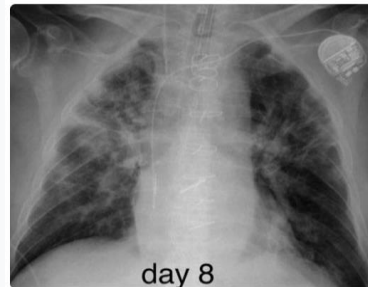


(5)

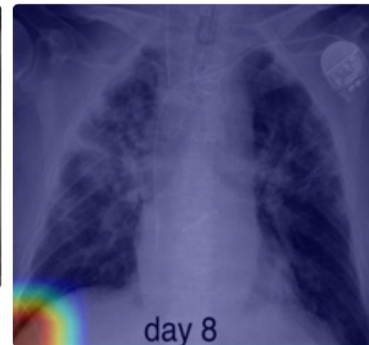
(4)

분석 결과

Positive일 확률이 99.66% 입니다.



원본 이미지



Grad-CAM 분석 결과

분석 결과

Negative일 확률이 92.01% 입니다.



원본 이미지



Grad-CAM 분석 결과

(오진)

COVID-19 딥러닝 후기

신대현

딥러닝으로 여러가지 모델을 검색하며 Densenet121을 선택해 결과물을 만드는 과정에 있어 같은 모델과 베이스로 시작을 했지만 결과값이 다르게 나오는 점이 신기하게 느껴졌다.

좋은 결과값을 만들기 위해 하이퍼파라미터를 수정하고 파인 튜닝을 하면서 여러 결과값을 도달하면서 회의하는 과정이 재밌었다.

전홍균

"테스트 셋에서 98%라는 높은 정확도를 얻었을 때, 프로젝트가 거의 성공했다고 생각했다. 하지만 학습 과정에서 보지 못한 새로운 이미지를 Streamlit 앱에 넣었을 때 예측이 계속 빗나가는 것을 보고 충격을 받았다. 이를 통해 **지표상의 성능과 실제 현장에서의 성능 사이에는 큰 차이가 있을 수 있다**는 것을 깨달았다. 모델의 진짜 성능은 통제된 데이터셋이 아닌, 예측 불가능한 실제 데이터로 검증해야 한다는 것을 배운 값진 경험이었다."

정승환

딥러닝 모델 학습에 대한 구조를 이해할 수 있었고 같은 코드로 시작하였어도 각자 다른 결과값이 도출되는 것을 알 수 있었음.

감사합니다!

