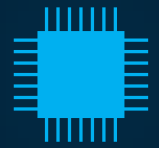


살균기 데이터를 활용한 식품 살균 공정에서의 품질 예측

2019100890 이승재

2019100895 이윤서

2019100899 임동현

목차

- 중간보고서 요약
- 의사결정나무 개선
- 딥러닝 실습
- 모델비교
- 결과 해석
- 모델 활용 가능성

분석배경 및 목적

공정개요

- ✓ 가열살균공정: 열처리를 통해 액상 식품의 미생물을 사멸시켜 식품의 안전성과 보존성을 향상시키는 공정
- ✓ 분석에서 다루는 공정: 식품제조업 저온살균공정
  (용해공정에서 용해·혼합된 내용물을 50~70℃에서 30분 이상 살균 및 교반)
- ✓ 살균 공정: HACCP의 CCP → 식품안전관리인증기준 이행 측면에서 매우 중요한 공정

문제상황

- ✓ 가열방식의 특성상 살균공정 동안 내용물의 온도는 지속적으로 변동
- ✓ 살균기 내부 센서가 설정 온도에 맞춰 가열 온도 조정하지만 조정 과정에서 시차로 인해 온도변화는 불가피
  → 내용물 전체에 영향
- ✓ 노후화된 설비 & 작업자의 주관적 판단에 의한 설정 온도 조정 → 일관적인 설정온도 조정에 한계 존재

해결방안

- ✓ 살균여부의 주요 지표인 살균온도 & 제품 불량 데이터의 머신러닝 → 실시간 품질 예측 및 공정 제어 이용

분석목적

- ✓ CCP 이행여부 결정, 제품품질에 영향 주는 살균온도 데이터, 최종품질검사 수행을 통해 획득한 제품 불량여부 데이터
  → 머신러닝 학습 → 공정 중의 설비 운영 데이터를 학습모델에 적용 → 완제품 품질 예측

데이터셋 소개

✓ 데이터 속성 정의

변수명	설명	데이터 타입
STD_DT	날짜, 시간(YYYY-MM-DD HH:MM:SS)	object
MIXA_PASTEUR_STATE	살균기A 가동상태	float64
MIXA_PASTEUR_STATE	살균기B 가동상태	float64
MIXA_PASTEUR_TEMP	살균기A 살균온도	float64
MIXA_PASTEUR_TEMP	살균기B 살균온도	float64
INSP	불량여부	object

:독립변수 :종속변수 object = 문자, float64 = 실수

✓ 데이터 크기 및 수량 : 6개 칼럼, 210,794개의 관측치

✓ 데이터 수집장비: PLC(설비데이터) 및 DBMS(품질데이터)

✓ 데이터 수집기간: 2020년 3월 4일 ~ 2020년 11월 11일 (약 8.5개월)

✓ 데이터 수집주기: 불규칙 (전처리 후 사이클타임 약 30분)

※ MIXA & MIXB → 분석대상 기업에서는 살균공정을 설비 2대에 나누어 병렬적으로 진행

※ 살균온도 데이터(_PASTEUR_TEMP) → 소수점 1자리가 생략, 값 nnn은 실제로 nn.n을 의미(예: 501 → 50.1℃)

※ 살균기 가동상태(_PASTEUR_STATE), 불량여부(INSP) → (0: 정지, 1: 운전), (OK: '양품', NG: '불량')

데이터셋 소개

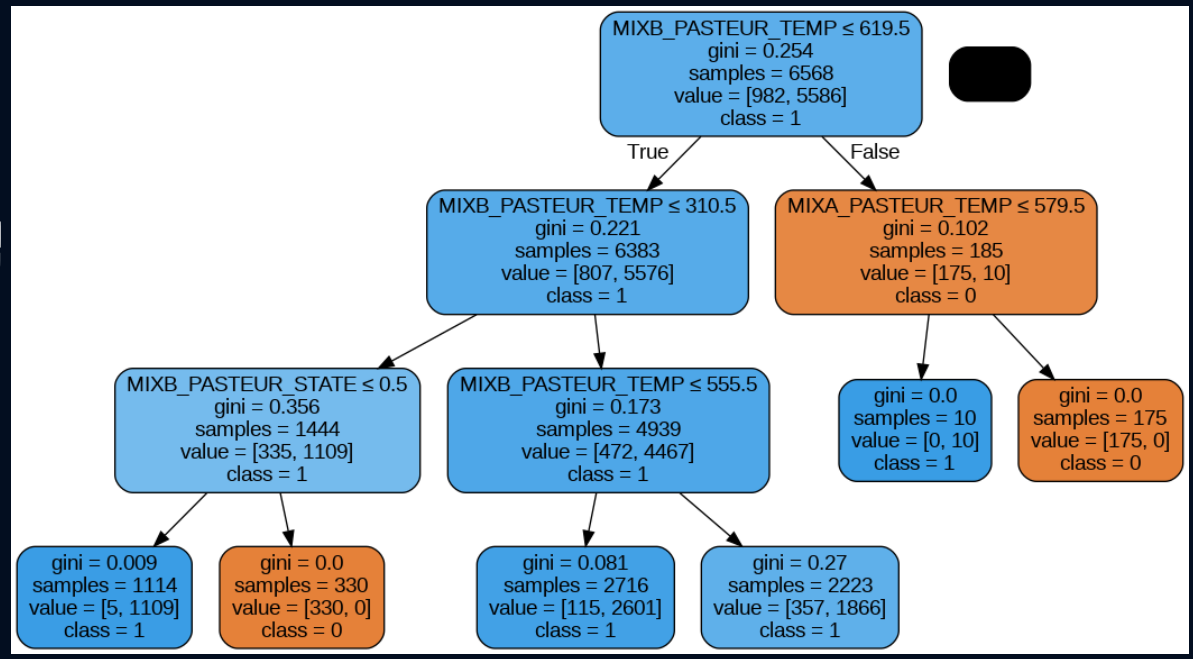
의사결정 나무(Decision Tree)

의사결정나무실습

정의

✓ 분류와 회귀 문제에 널리 사용되는 모델로,
데이터를 분석하여 이들 사이에 존재하는 패턴을
예측 가능한 규칙들의 조합으로 나타내는데 사용

✓ 규칙들을 트리 구조로 표현, 각 분기점(node)은
특정 질문에 대한 답을, 각 잎사귀(leaf)는 결정 결과를 나타냄



특징

✓ 이해하기 쉽고 해석이 용이

✓ 데이터의 사전 처리가 거의 필요하지 않음

✓ 숫자형과 범주형 입력 변수를 모두 다룰 수 있음

한계

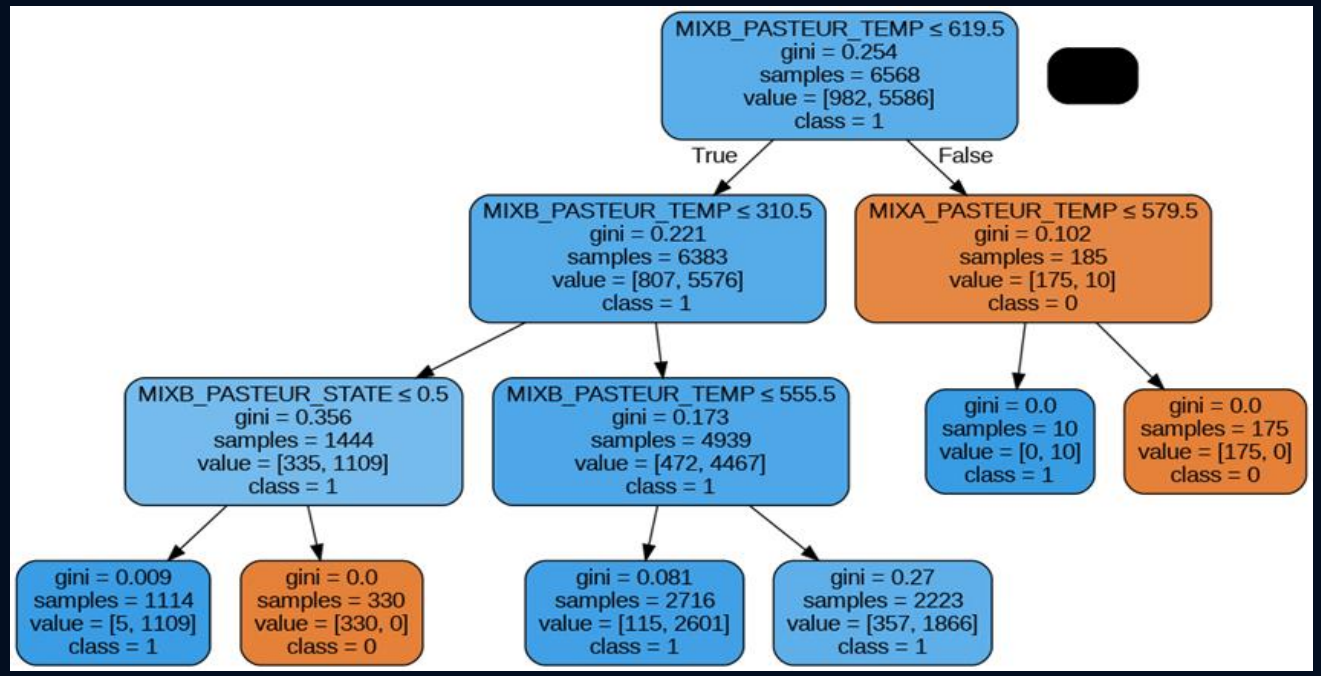
✓ 과적합(Overfitting) 문제가 발생 가능

✓ 안정성이 떨어짐 → 데이터의 작은 변화에도 결과가 크게 변동 가능

중간보고서 요약

중간보고서 실습 요약

- ✓ 라이브러리 & 데이터 불러오기
- ✓ 데이터 종류 및 개수 확인
- ✓ 데이터 정제 후 데이터 특성파악
- ✓ 학습 / 평가 데이터 분리
- ✓ 모델 훈련 – 의사결정나무 실행
- ✓ 결과분석 및 해석



분류성능:

오차 행렬 [[237 196]
[1 2381]]

정확도 : 0.9300 정밀도: 0.9239, 재현율: 0.9996, F1: 0.9603, AUC: 0.7735

의사결정나무개선

모델 훈련 – 의사결정나무 실행

```
# 이전 코드 동일
dt_clf = DecisionTreeClassifier(max_depth=5, min_samples_leaf=2) #의사결정나무 알고리즘 불러오기
dt_clf = dt_clf.fit(X_train, y_train) # 알고리즘에 데이터 학습시키기
dt_prediction = dt_clf.predict(X_test) # 훈련된 모델 테스트
```

의사결정나무 시각화

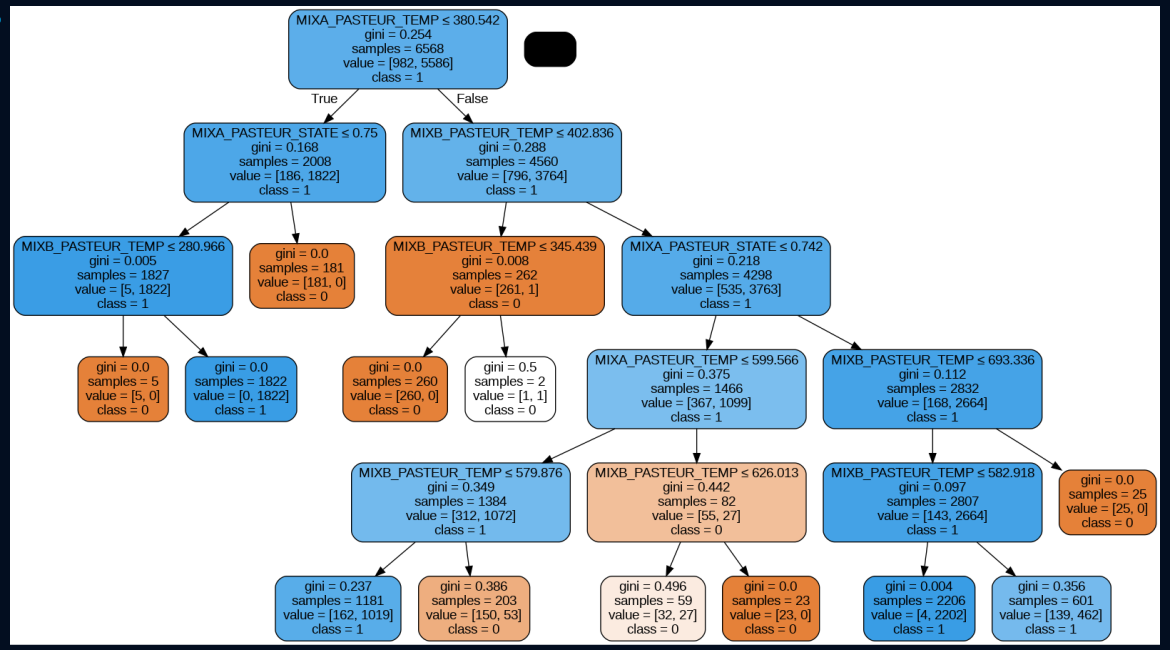
```
feature_names = df.columns.tolist()
feature_names = feature_names[1:5] #'STD_DT'는 날짜라서 제거, 'INSP'는 타겟이라 제거
target_name = np.array(['0', '1']) # 원본 데이터 변수이름 추출 및 타겟변수 이름 설정

# Graphviz로 의사결정나무 시각화
dt_dot_data = tree.export_graphviz(dt_clf, feature_names=feature_names, class_names=target_name,
                                   filled=True, rounded = True, special_characters = True)
dt_graph = pydotplus.graph_from_dot_data(dt_dot_data)

Image(dt_graph.create_png())
```

결과분석 및 해석

의사결정나무 개선

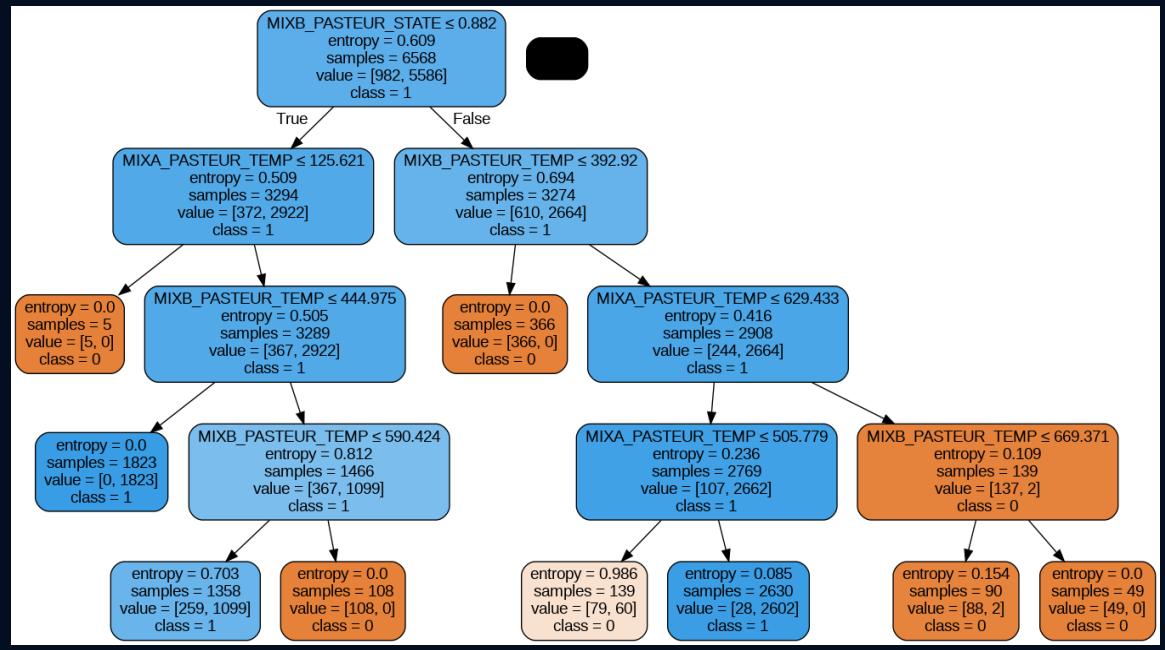
```
# Confusion Matrix로 분류성능 확인
from sklearn.metrics import accuracy_score, precision_score, recall_score, roc_auc_score
from sklearn.metrics import f1_score, confusion_matrix, precision_recall_curve, roc_curve

def get_clf_eval(y_test=None, pred=None):
    confusion = confusion_matrix(y_test, pred)
    accuracy = accuracy_score(y_test, pred)
    precision = precision_score(y_test,pred)
    recall = recall_score(y_test, pred)
    f1 = f1_score(y_test, pred)
    roc_auc = roc_auc_score(y_test, pred)
    print('오차 행렬')
    print(confusion)
    print('정확도 :{0: .4f}, 정밀도: {1: .4f}, 재현율: {2: .4f}, F1 : {3: .4f}, ₩
    AUC: {4: .4f}'.format(accuracy, precision, recall, f1, roc_auc))

get_clf_eval(y_test, dt_prediction)
```

의사결정나무 개선



하이퍼파라미터

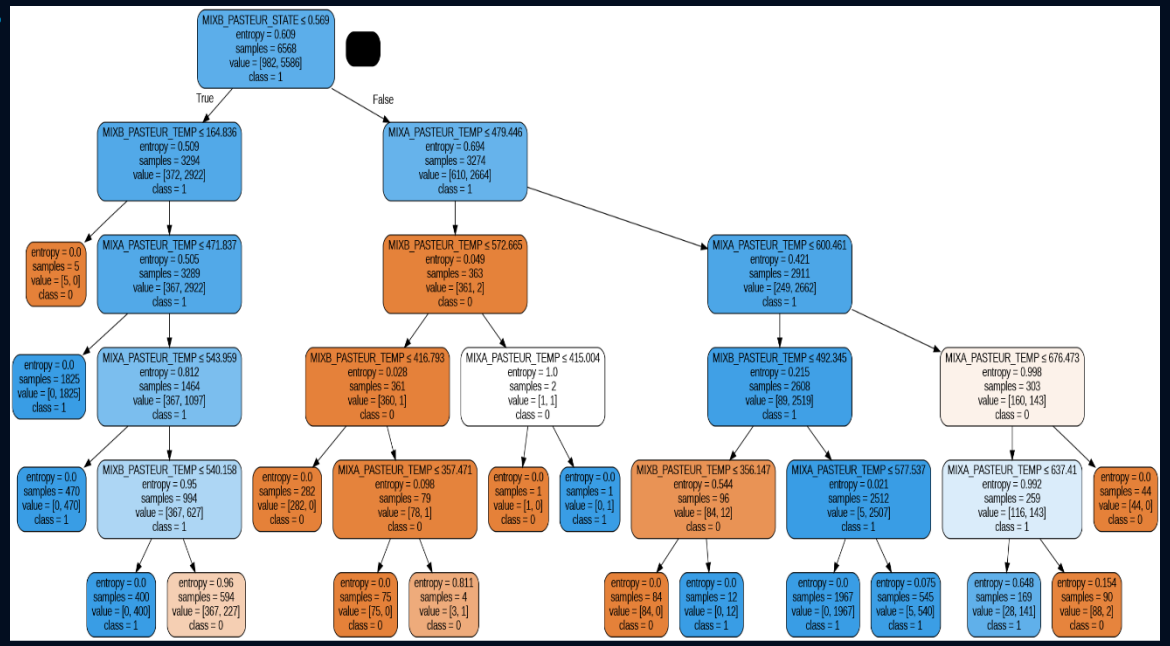
- ✓ Criterion = 'gini'
- ✓ Splitter = 'random'
- ✓ Max_depth = 5
- ✓ Min_sample_leaf = 2
- ✓ 나머지 하이퍼파라미터는 default 값



하이퍼파라미터

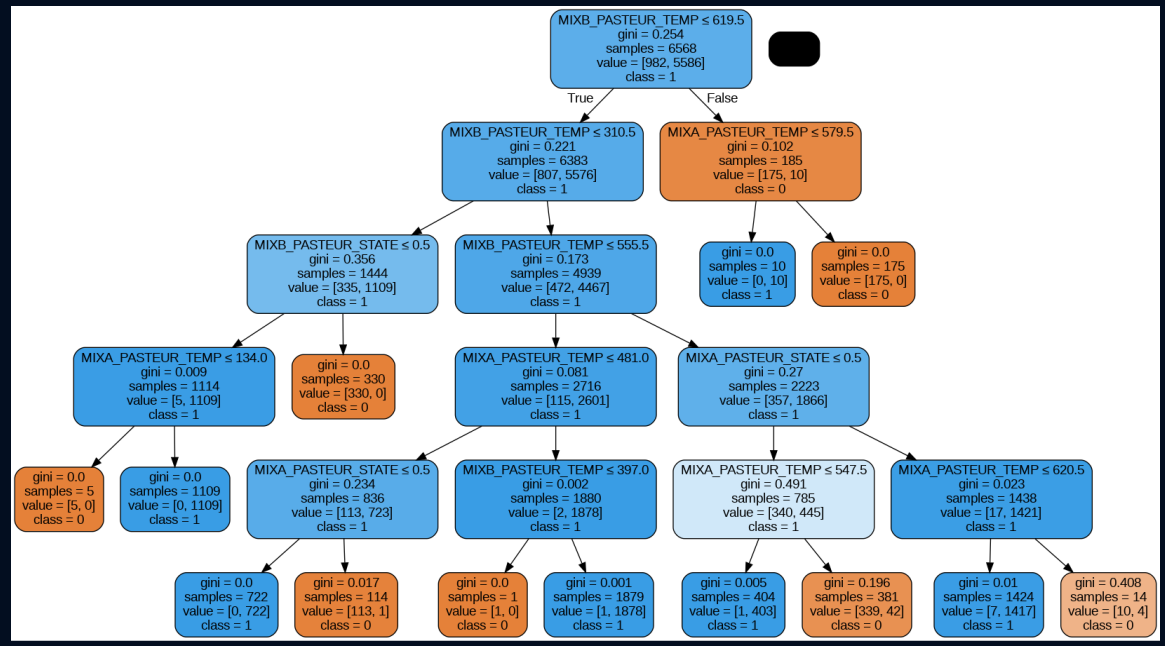
- ✓ Criterion = 'entropy'
- ✓ Splitter = 'random'
- ✓ Max_depth = 4
- ✓ Min_sample_leaf = 1
- ✓ 나머지 하이퍼파라미터는 default 값

의사결정나무 개선



하이퍼파라미터

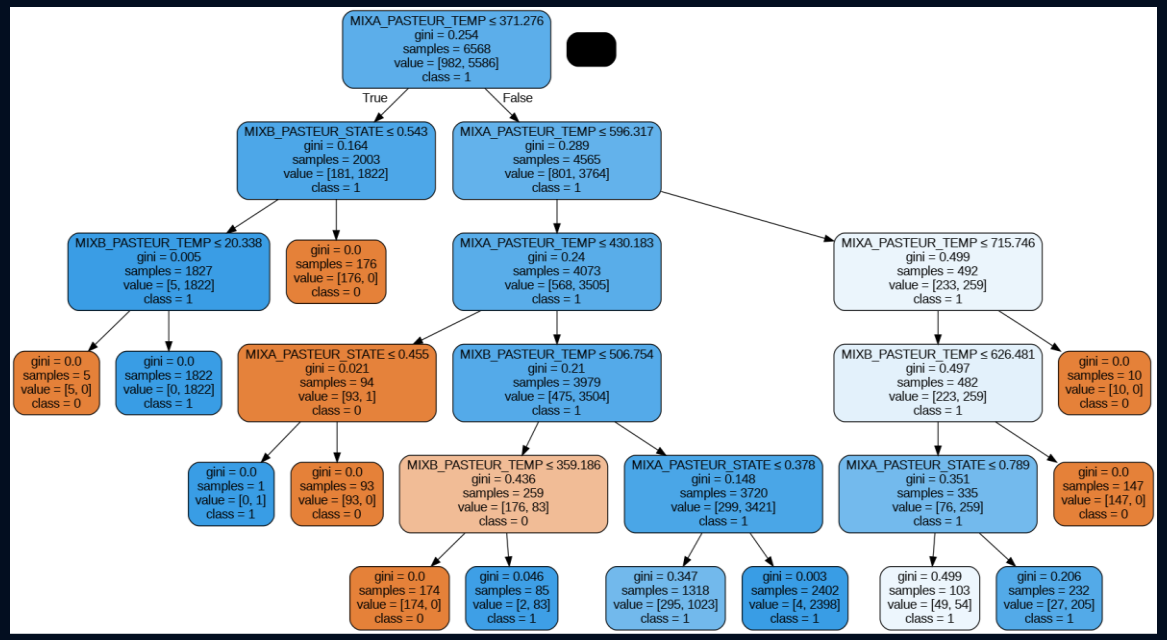
- ✓ Criterion = 'entropy'
- ✓ Splitter = 'random'
- ✓ Max_depth = 5
- ✓ Min_sample_leaf = 1
- ✓ 나머지 하이퍼파라미터는 default 값



하이퍼파라미터

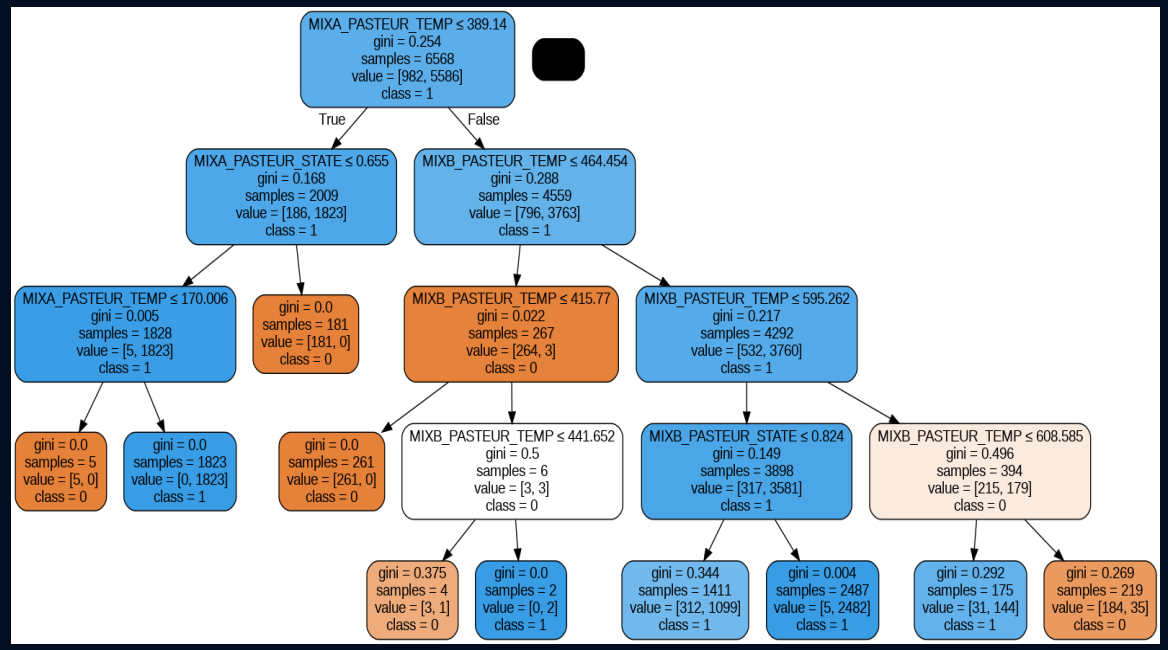
- ✓ Criterion = 'gini'
- ✓ Splitter = 'best'
- ✓ Max_depth = 5
- ✓ Min_sample_leaf = 1
- ✓ 나머지 하이퍼파라미터는 default 값

의사결정나무 개선



하이퍼파라미터

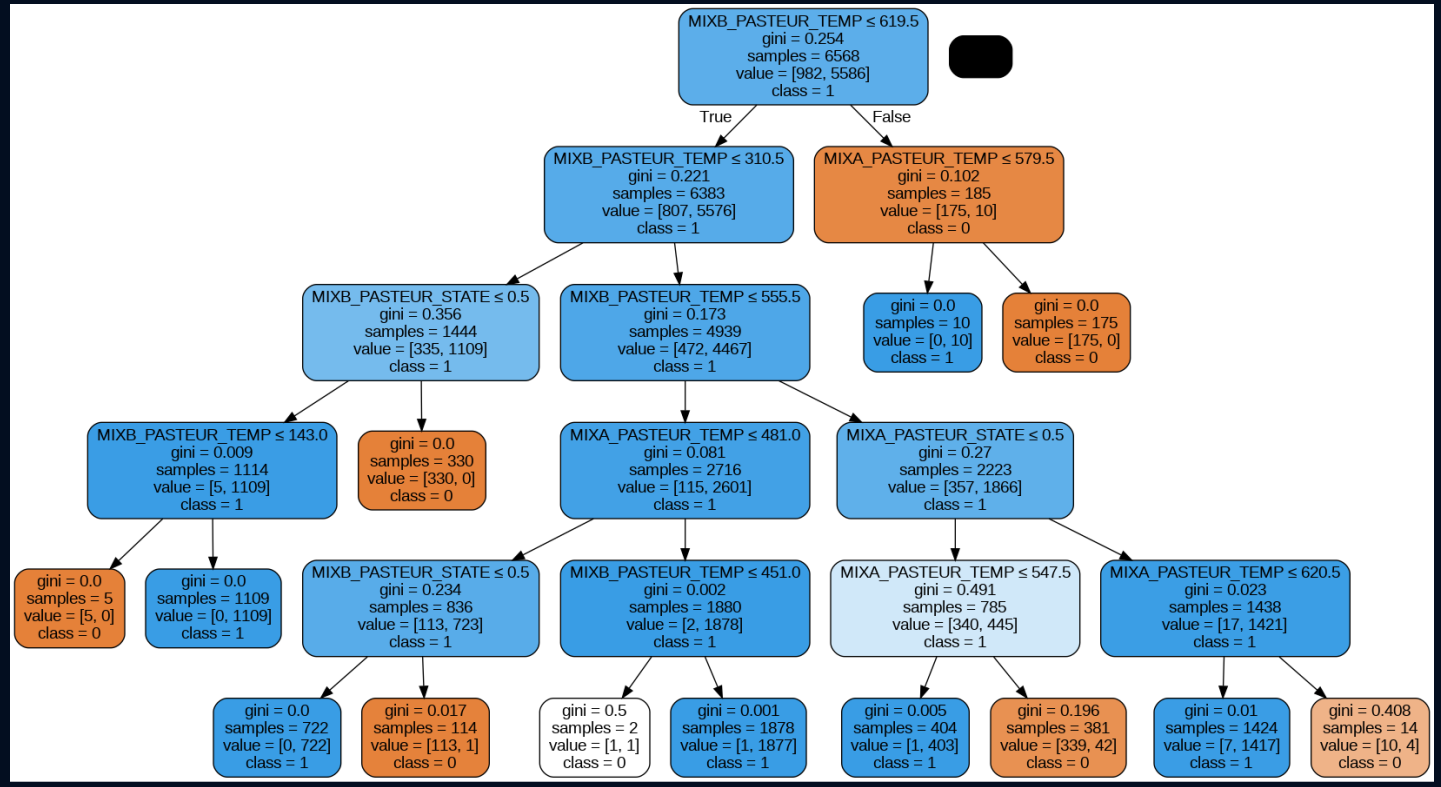
- ✓ Criterion = 'gini'
- ✓ Splitter = 'random'
- ✓ Max_depth = 5
- ✓ Min_sample_leaf = 1
- ✓ 나머지 하이퍼파라미터는 default 값

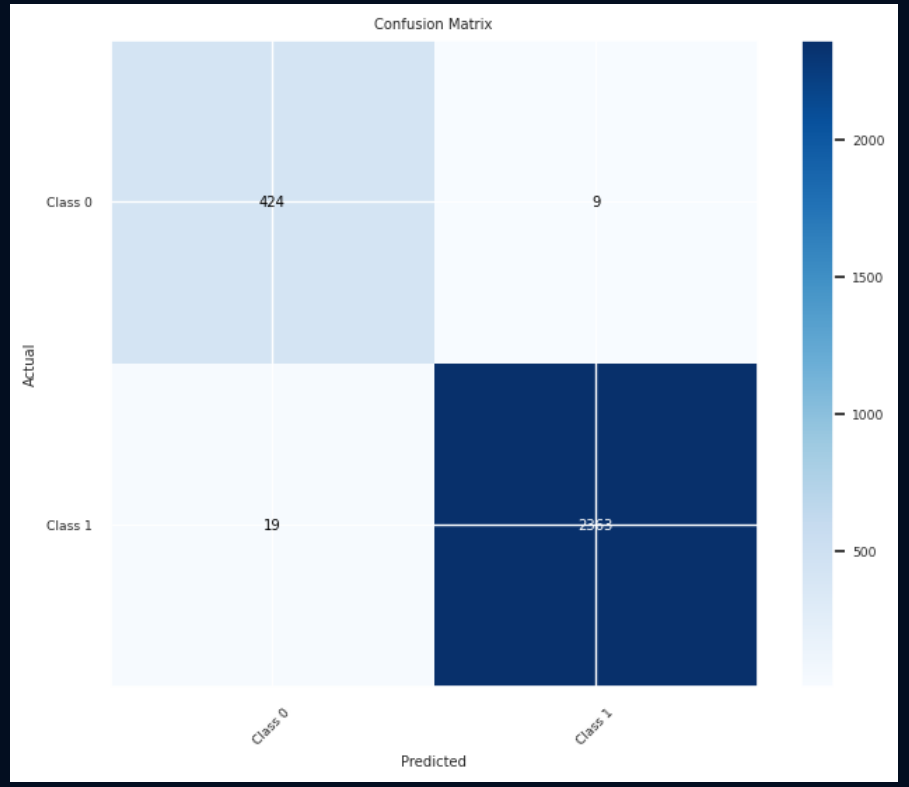


하이퍼파라미터

- ✓ Criterion = 'gini'
- ✓ Splitter = 'random'
- ✓ Max_depth = 4
- ✓ Min_sample_leaf = 1
- ✓ 나머지 하이퍼파라미터는 default 값

의사결정나무개선 - BEST





하이퍼파라미터

- ✓ Criterion = 'gini'
- ✓ Splitter = 'best'
- ✓ Max_depth = 5
- ✓ Min_sample_leaf = 2
- ✓ 나머지 하이퍼파라미터는 default 값

분류성능:

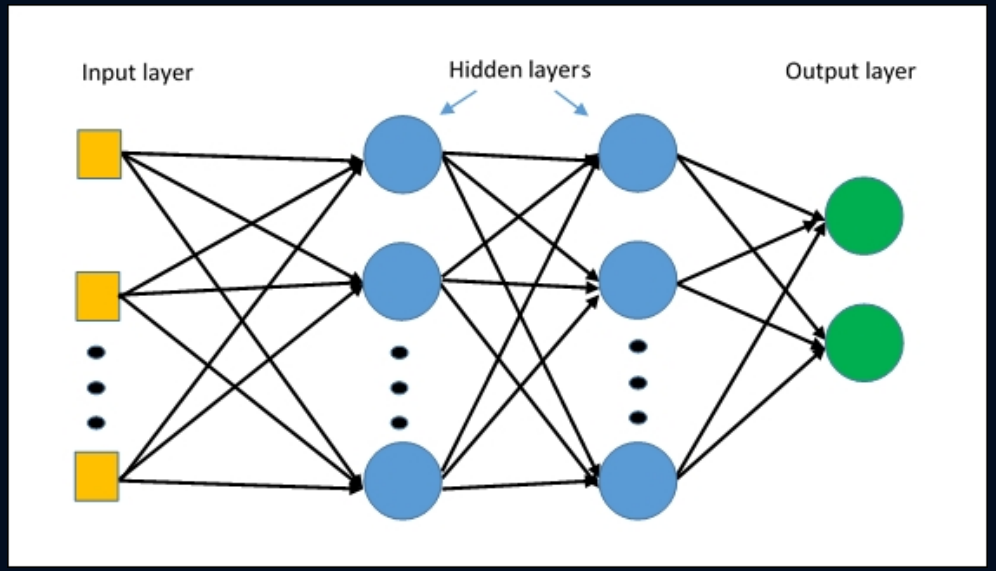
오차 행렬 [[424 9]
[19 2363]]

정확도 : 0.9901 정밀도: 0.9962, 재현율: 0.9920, F1: 0.9941, AUC: 0.9856

딥러닝실습 - DNN

DNN(Deep Neural Network)

- 특징: 다수의 은닉층으로 구성
- 목적: 복잡한 데이터 패턴 학습
- 구조: 입력층-여러 은닉층-출력층
- 학습 방식: 역전파와 그래디언트 하강법을 사용하여 가중치 조절
- 활성화 함수: 비선형 함수 (ex-ReLU)



딥러닝 실습 - 코드

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import
StandardScaler, OneHotEncoder

# 데이터셋 컬럼 구조
columns = ['MIXA_PASTEUR_STATE','MIXB_PASTEUR_STATE',
'MIXA_PASTEUR_TEMP','MIXB_PASTEUR_TEMP','INSP']

# 범주형 데이터 (One-hot encoding)
categorical_cols =
['MIXA_PASTEUR_STATE','MIXB_PASTEUR_STATE']
encoder = OneHotEncoder()
categorical_data =
encoder.fit_transform(data[categorical_cols]).toarray()

# 연속형 데이터 (표준화)
continuous_cols =
['MIXA_PASTEUR_TEMP','MIXB_PASTEUR_TEMP']
scaler = StandardScaler()
continuous_data = scaler.fit_transform(data[continuous_cols])

# 데이터 합치기
X = np.concatenate([categorical_data, continuous_data], axis=1)
# 타겟 변수 (One-hot encoding)
encoder2 = OneHotEncoder()
Y = encoder2.fit_transform(data[['INSP']]).toarray()

# 데이터 분할
X_train, X_test, Y_train,
Y_test = train_test_split(X, Y, test_size=0.3, random_state=1)
```

딥러닝 실습 - 코드

```
model = Sequential()
model.add(Dense(256, input_shape=(X_train.shape[1],),
activation='elu', kernel_initializer='he_normal'))
model.add(BatchNormalization())
model.add(Dropout(0.25))
model.add(Dense(128, activation='elu',
kernel_initializer='he_normal'))
model.add(BatchNormalization())
model.add(Dropout(0.25))
model.add(Dense(64, activation='elu',
kernel_initializer='he_normal'))
model.add(BatchNormalization())
model.add(Dropout(0.25))
model.add(Dense(Y_train.shape[1], activation='softmax'))

ReduceLR = ReduceLROnPlateau(monitor='val_accuracy',
mode ='max', factor=0.5,  min_lr=1e-7, verbose = 2, patience=5)
EarlyStop = EarlyStopping(monitor='val_loss', mode='min',
verbose = 2, patience=10, restore_best_weights=True)
```

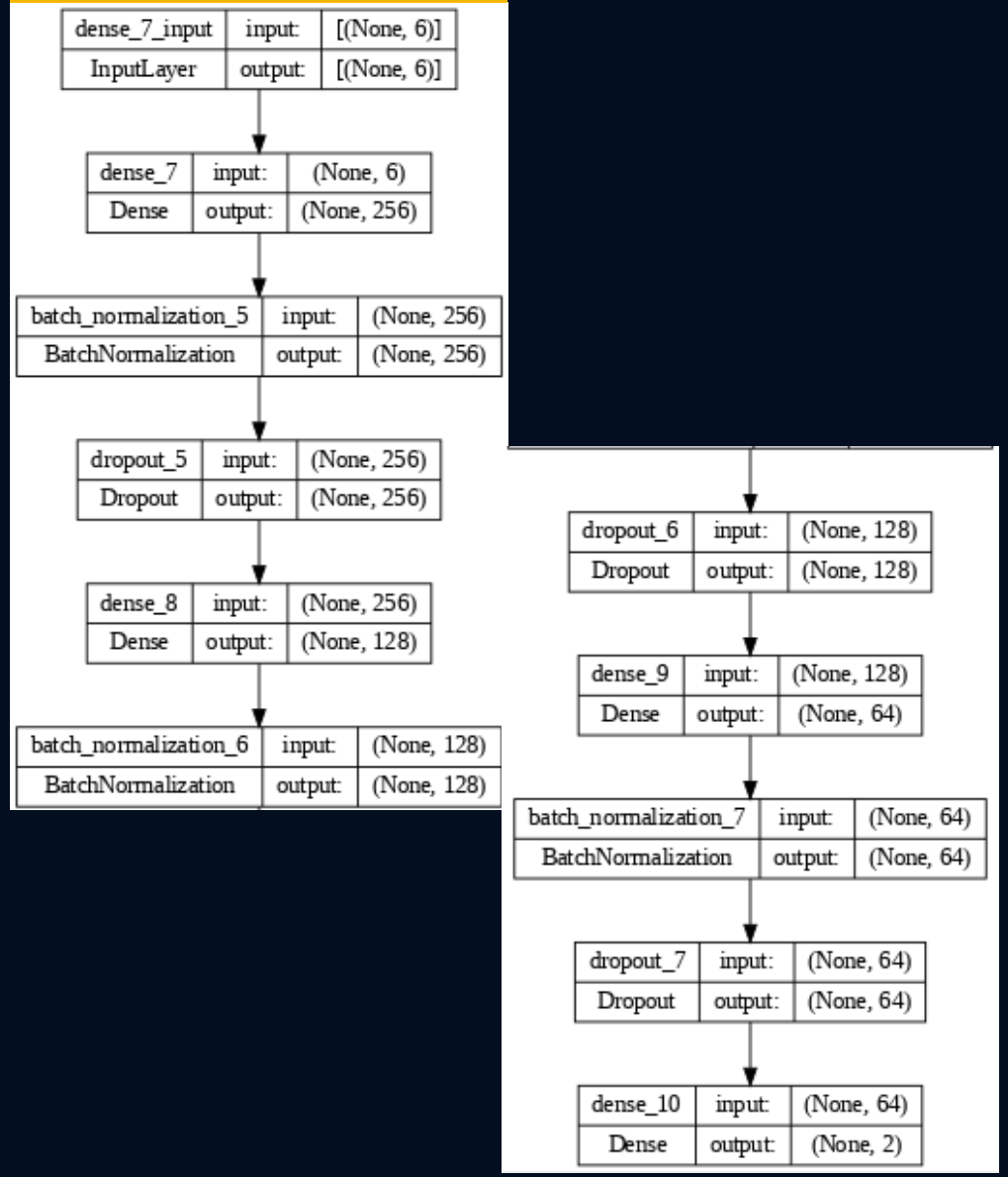
```
from tensorflow.keras import layers, models, datasets, utils
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout,
BatchNormalization, Activation
from tensorflow.keras.optimizers import Nadam
from tensorflow.keras.callbacks import
EarlyStopping, ReduceLROnPlateau

model.compile(loss='categorical_crossentropy',
optimizer=Nadam(0.001),metrics=['accuracy'])
hist=model.fit(X_train,Y_train, batch_size=64, epochs=100,
validation_split=0.2, callbacks=[EarlyStop,ReduceLR],
shuffle = True, verbose=1)

# 하이퍼파라미터
# 활용한 방법
```

딥러닝 실습 - 코드

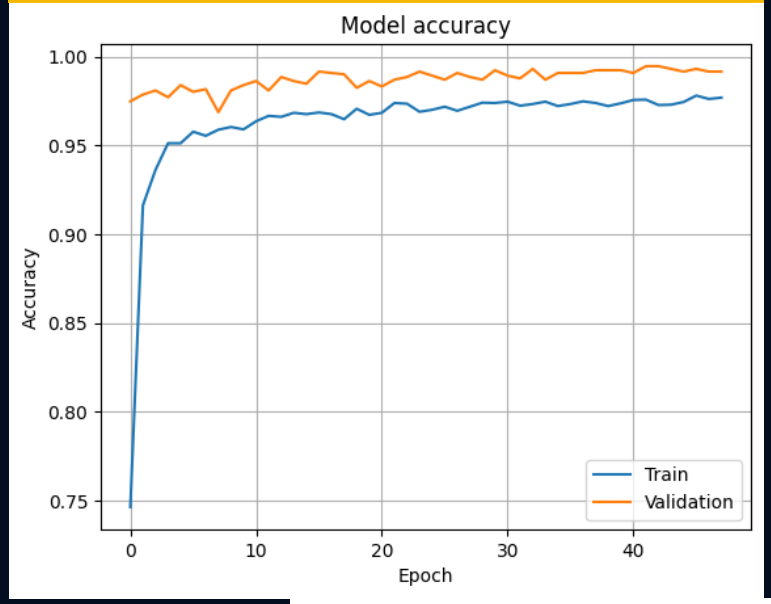
```
from sklearn.metrics import accuracy_score, precision_score,
recall_score, roc_auc_score, confusion_matrix, precision_recall_curve,
roc_curve

#저장된 모델 불러오기
from tensorflow.python.keras.models import load_model
loaded_model = tf.keras.models.load_model(dir+"dnn_pasteurizer.keras")
Y_pred = loaded_model.predict(X_test)
Y_pred_classes = np.argmax(Y_pred, axis=1)
Y_test_classes = np.argmax(Y_test, axis=1)

#혼돈 행렬
conf_matrix = confusion_matrix(Y_test_classes, Y_pred_classes)

# 정확도
accuracy = accuracy_score(Y_test_classes, Y_pred_classes)
# 정밀도
precision = precision_score(Y_test_classes, Y_pred_classes,
average='weighted')
# 재현율
recall = recall_score(Y_test_classes, Y_pred_classes,
average='weighted')
# F1 점수
f1 = f1_score(Y_test_classes, Y_pred_classes,
average='weighted')
# ROC AUC 점수
roc_auc = roc_auc_score(Y_test, Y_pred, multi_class='ovr')
# 정밀도-재현율 곡선
precision_vals, recall_vals, thresholds_pr =
precision_recall_curve(Y_test.ravel(), Y_pred.ravel())
# ROC 곡선
fpr, tpr, thresholds_roc = roc_curve(Y_test.ravel(), Y_pred.ravel())
```

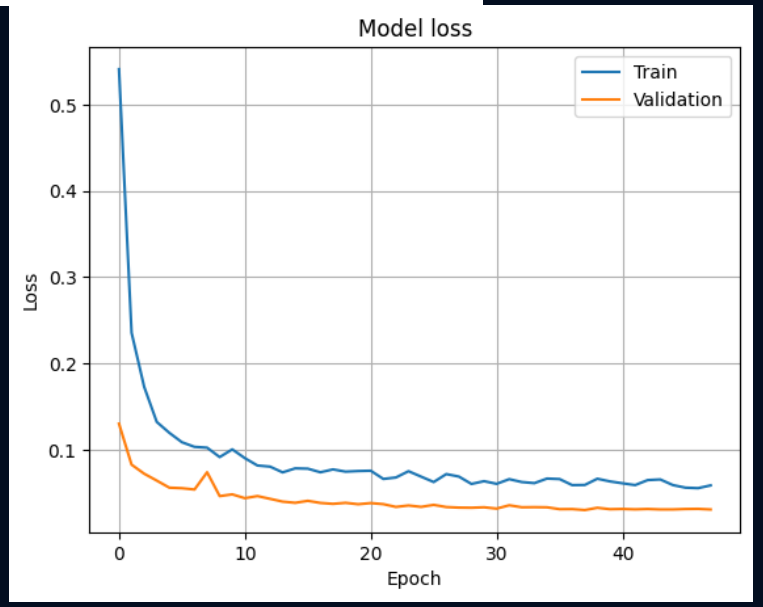
딥러닝실습 - 구조 및 학습

모델 구조

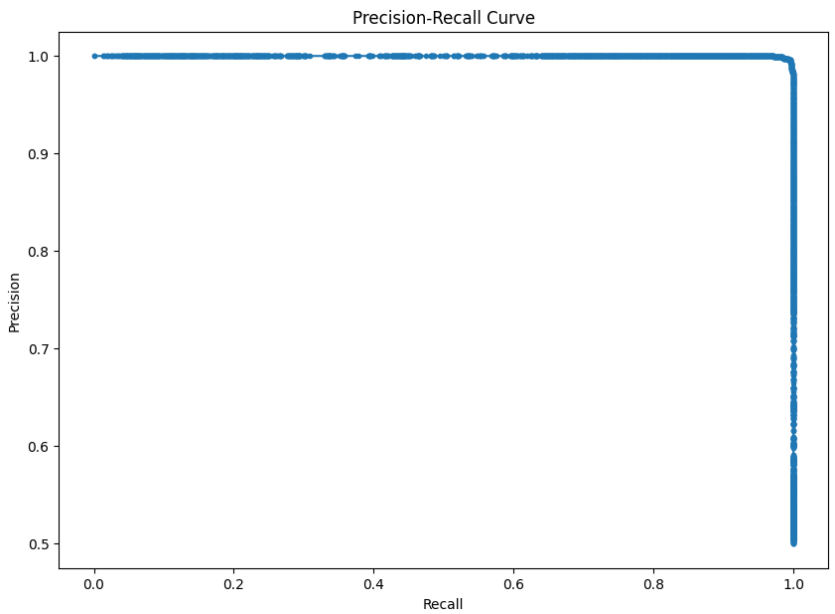


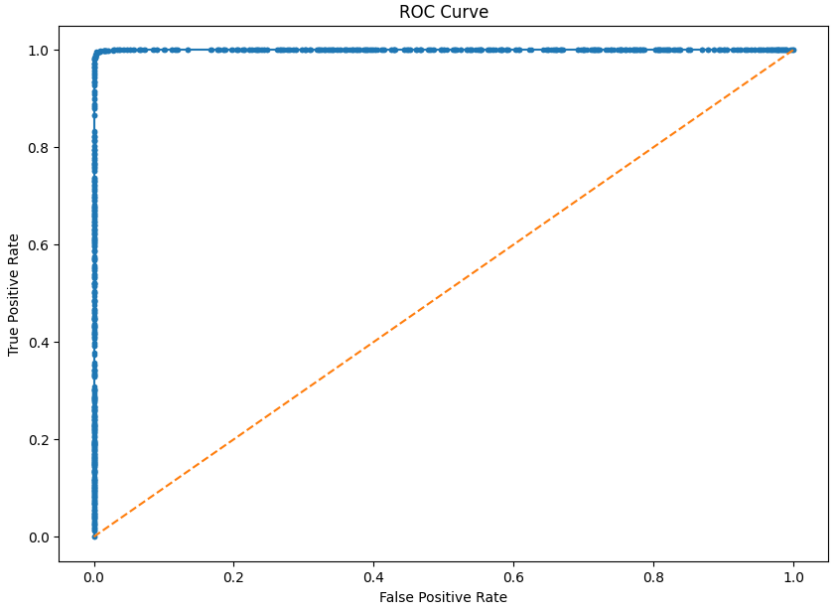
학습 그래프



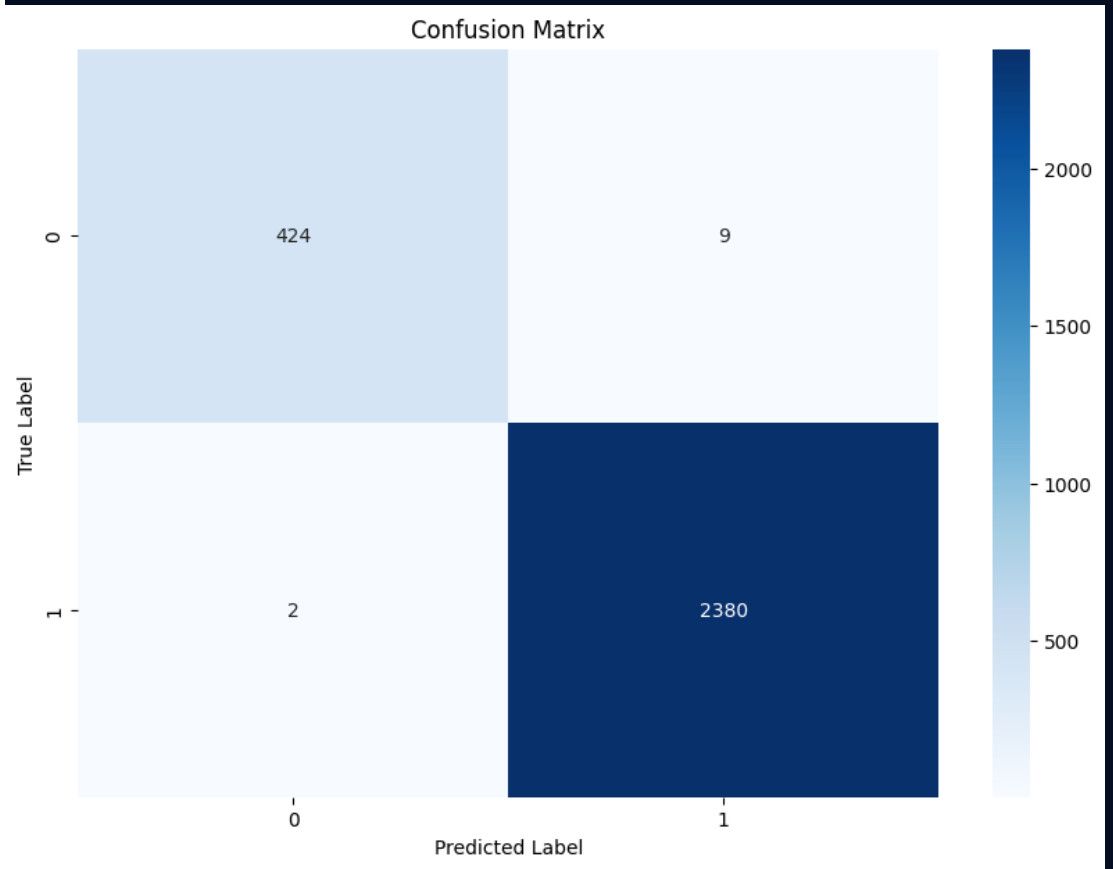


딥러닝 실습 - 학습 결과





Accuracy: 0.9960923623445825
Precision: 0.9960900557785811
Recall: 0.9960923623445825 F1
F1 Score: 0.9960793072928823
ROC AUC Score: 0.999809968140577



모델 비교

Decision Tree

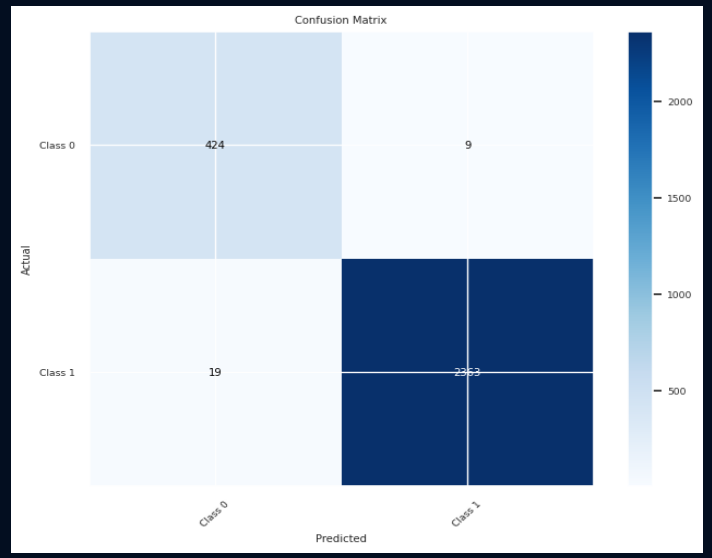
Accuracy: 0.9901

Precision: 0.9962

Recall: 0.9920

F1 Score: 0.9941

ROC AUC Score: 0.9856



Deep Neural Network

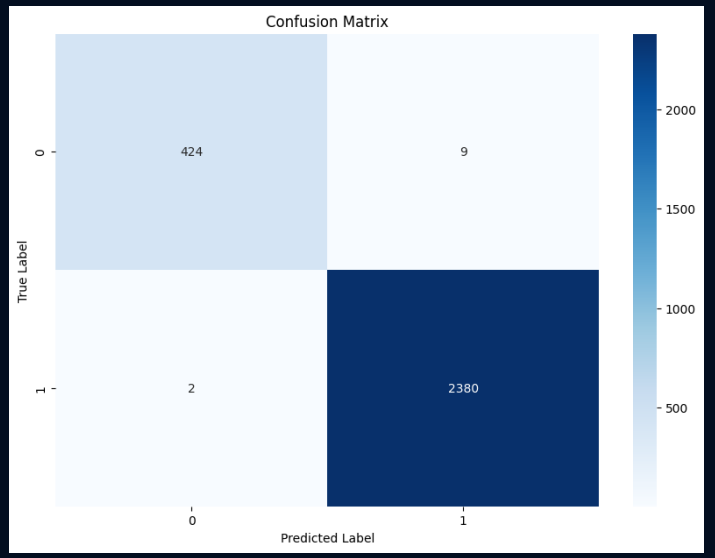
Accuracy: 0.9960

Precision: 0.9960

Recall: 0.9960

F1 Score: 0.9960

ROC AUC Score: 0.9998



모델 비교

Decision Tree

장점 :

결과가 명확하고 해석하기 쉬움

정규화/표준화 필요 X → 데이터 전처리 최소

상대적으로 적은 양의 데이터로도 효과적

단점 :

데이터 변화에 민감해 안정성 ↓

복잡한 패턴에서 인식 어려울 수 있음

과적합 대응 :

가지치기

Deep Neural Network

장점 :

중요한 특성을 자동으로 인식하고 학습

대량의 데이터 처리 적합

데이터가 많을수록 성능 개선

단점 :

해석하기 어려움

많은 계산 자원과 데이터, 시간 필요

과적합 대응 :

매개변수 조정 / 정규화 기술

결과해석

의사결정나무 해석

- ✓ MIXB_PASTEUR_TEMP가 310.5 이하, MIXA_PASTEUR_STATE=0, MIXA_PASTEUR_TEMP가 134 초과시 양품
- ✓ MIXA_PASTEUR_TEMP가 481 이하, MIXB_PASTEUR_STATE=0, MIXB_PASTEUR_TEMP가 310.5 초과, 5555 이하시 양품
- ✓ MIXA_PASTEUR_TEMP가 481 초과, MIXB_PASTEUR_TEMP가 451 초과, 5555 이하시 양품
- ✓ MIXA_PASTEUR_TEMP가 547.5 이하, MIXB_PASTEUR_STATE=0, MIXB_PASTEUR_TEMP가 5555 초과, 619.5 이하시 양품
- ✓ MIXA_PASTEUR_TEMP가 620.5 이하, MIXB_PASTEUR_STATE=1, MIXB_PASTEUR_TEMP가 5555 초과, 619.5 이하시 양품
- ✓ MIXA_PASTEUR_TEMP 579.5 이하, MIXB_PASTEUR_TEMP 619.5 초과시 양품

제조현장 관점에서 분석결과 해석

- ✓ 살균기 B의 살균온도가 31.05°C 이하, 살균기 A가 가동이 중지된 상태에서 살균온도가 13.4 °C 초과로 운영하는 경우 양품 생산 예측됨
- ✓ 살균기 A의 살균온도가 48.1°C 이하, 살균기 B가 가동이 중지된 상태에서 살균온도가 31.05°C 초과, 55.55°C 이하로 운영하는 경우 양품 생산 예측됨
- ✓ 살균기 A의 살균온도가 48.1°C 초과, 살균기 B의 살균온도가 45.1°C 초과, 55.55°C 이하로 운영하는 경우 양품 생산 예측됨
- ✓ 살균기 A의 살균온도가 54.75°C 이하, 살균기 B가 가동이 중지된 상태에서 살균온도가 55.55°C 초과, 61.95°C 이하로 운영하는 경우 양품 생산 예측됨
- ✓ 살균기 A의 살균온도가 62.05°C 이하, 살균기 B가 가동된 상태에서 살균온도가 55.55°C 초과, 61.95°C 이하로 운영하는 경우 양품 생산 예측됨
- ✓ 살균기 A의 살균온도가 57.95°C 이하, 살균기 B의 살균온도가 61.95°C 초과로 운영하는 경우 양품 생산 예측됨

활용가능성

유사 타 현장에 분석 결과 적용

분석 결과 적용 가능한 제조 현장 소개

- ✓ 최첨단 살균기 사용 불가
- ✓ 살균기의 일관적인 설정온도 조정에 한계가 존재
- ✓ 공정운영 데이터를 설비 PLC 또는 센서를 통해 확인 및 수집 가능
- ✓ 품질검사 결과를 전산화하여 관리

위 조건을 모두 만족하는 식품 제조업체

분석 결과 적용시 주요 고려사항

- ✓ 생산제품 및 투입원료에 따라 분석 결과가 변화 → 해당 제품의 데이터로 분석모델 재학습 필요
- ✓ 살균공정에서 측정되는 다른 변수(공정, 환경, 품질 데이터 등)들도 포함하여 모델 학습 진행 → 더 정확한 모델 생성 가능
- ✓ 본 분석에서 활용한 특성값은 타 제조현장 상황과 다를 수 있음 → 적용시 현장 전문가의 의견을 반영하여 적용 여부 결정