* **분석 개요**

1. **사용 언어 :** Python
2. **분석 환경 :** [CPU] Apple M2, [RAM] 8GB, [GPU] Colab T4
3. **분석 데이터 정보**
   1. 데이터 수집 장소 : 자동차 부품사 CNC 가공 공정
   2. 데이터 내용 : CNC 가공 시 설비에서 자동 저장 되는 데이터 및 설비 설정 조건 등 총 60개의 Features
      * X, Y, Z 축 부품의 위치, 속도, 전류, 전압 등 (자동 저장)
      * 스핀들 위치, 속도, 전류, 전압, 토크, 현재 가공 단계 등 (자동 저장)
      * 가공 소재, Tool 이동 속도, 소재 Clamping 압력 등 (설비 설정 조건)
      * Tool condition, 공정 완료 여부, 외관 검사 판정 결과 등 (수동 작성)
   3. 데이터 파일 : 총 26개
      * experiment\_01.csv ~ experiment\_25.csv (25개의 자동 저장 csv 파일)
      * train.csv (1개의 결과요약 csv 파일)
4. **분석 데이터 구조**
   1. Features : 60 columns
   2. Samples : 32048 raw (이중 30%를 Teste 데이터셋으로 할당)
   3. Labels : 1 or 0 (Pass or Fail)
5. **학습 배경 및 목적**
   1. 학습 배경
      * 현 상황 : CNC 가공이 완료 된 이후 외관검사를 통해서 불량을 판정 하고 설비 조건에 대한 조치가 가능함
      * 현 상황 문제점 : 설비 조건에 문제가 있을 경우, 마지막 외관검사 공정까지 진행 되는 동안 계속 불량품을 생산하게 됨 -> Loss 발생
   2. 학습 목적
      * 각 공정에서의 조건 및 상태에 따라 이 제품이 불량품이 될 것인지 양품이 될 것인지 예측 (성능 : 99.9% 이상 정확도)
      * 마지막 공정까지 가기 전 미리 양불을 판단하여 설비 조건을 조정 가능 -> Loss 최소화
6. **데이터 학습에 사용된 모델**
   1. XGBClassifier
   2. LGBMClassifier
   3. CatBoostClassifier
   4. RandomForestClassifier
   5. LogisticRegression
   6. **DNN (Deep Neural Network) -> Best Model**
   7. **CNN (Convolutional Neural Network)**
7. **데이터 학습 및 예측 결론**
   1. Best 모델 : DNN, CNN
   2. DNN 모델은 Fully Connected Layers 로 구성되어 모든Feature값들에 대한 가중치 (파라미터) 를 찾아 최적의 모델을 찾는 알고리즘
   3. CNN모델은 주로 이미지나 지역적 패턴이 있는 데이터를 학습 할 때 사용되는 모델. 분석 대상인 데이터셋은 X, Y, Z 축의 Featrure들이 연속적으로 배치되어 있기 때문에 지역적인 패턴이 있어 CNN 모델도 학습 되는 것으로 보임

[모델별 예측 정확도] : 전체 Features 학습 시

[모델별 예측 정확도] : 2개 Featrue 삭제 후 학습 (수동 작성 Feature 삭제)

'tool\_condition' : 일정시간 사용한 Tool인지, 새로운 Tool인지 여부

'machining\_finalized' : 공정 완료 여부

텍스트, 스크린샷, 번호, 폰트이(가) 표시된 사진

자동 생성된 설명

* **데이터 분석 : 원본데이터 구조 확인**

1. 설비에서 자동 저장 된 25개의 csv 파일 요약본 확인
   * 25개의 각 csv 파일마다 공정별 데이터가 저장 되어 있음

**텍스트, 스크린샷, 폰트, 흑백이(가) 표시된 사진

자동 생성된 설명텍스트, 폰트, 스크린샷, 화이트이(가) 표시된 사진

자동 생성된 설명**

* + 아래는 25개 파일에 대한 설정값 및 결과 요약 데이터프레임

(1행 = 1개 csv 파일 정보)

* + 결과 라벨(yes or no), 공정 완료여부(yes or no) 등 6개의 정보를 담고 있음

**텍스트, 스크린샷, 번호이(가) 표시된 사진

자동 생성된 설명**

* + - Machining\_finalized (공정 완료여부) 항목이 ‘no’ 일 경우, pssed\_visual\_inspecion(외관검사) 항목은 ‘NaN’ 으로 되어있음 (외관검사를 진행하지 않았기 때문)

1. 25개의 csv 파일 크기 확인 및 라벨 클래스의 빈도 균형 확인
   * 먼저, **csv** 파일명 25개를 리스트로 저장 (각 파일명을 for 문으로 불러오기 위해)

텍스트, 스크린샷, 디자인이(가) 표시된 사진

자동 생성된 설명

* + - cnc\_data 폴더에는 25개의 원본 csv 파일과 1개의 결과 요약 파일(train.csv)이 있음
    - 결과 요약 파일을 제외하고 25개의 csv 파일명을 리스트로 저장
  + 25개 원본 csv 파일을 데이터프레임 형태로 변환 후 리스트에 저장

텍스트, 스크린샷, 디자인이(가) 표시된 사진

자동 생성된 설명

* + - summary\_df : 결과 요약 데이터프레임
    - 각 csv파일(각 행)의 크기 정보를 추가

-> 라벨 열(paased\_visual\_inspection)의 빈도 분포를 확인 하기 위함

* + 라벨 클래스별 빈도 확인

텍스트, 스크린샷, 소프트웨어, 웹 페이지이(가) 표시된 사진

자동 생성된 설명

* + - 공정 완료 여부, 외관검사 합격 여부 feature를 기준으로 groupby() 하여 클래스 빈도 확인
      * 공정 미완료 fail : 3228 (10.07%)
      * 공정 완료 fail : 6175(19.27%)
      * 공정 완료 pass : 22645(70.66%)
    - pass, fail 의 비중이 약 7:3 정도로 나타남 -> 학습 하는데는 문제 없을 것
* **데이터 전처리 : 25개 원본파일을 1개로 합치기, 원핫인코딩 처리**

1. 25개의 csv파일에 결과 요약 데이터를 합친 뒤 25개 파일을 1개의 파일로 만들기
   * 결과 요약 feature를 각 데이터프레임에 합치기

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

* + - concat() 처리 후 25개 데이터프레임의 열은 56개
  + 25개 데이터 프레임을 하나로 합치기

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

* + - concat\_df : 총 32048개의 행(데이터)를 가진 데이터 프레임

1. 불필요한 feature 제거, 이진 데이터 1과0으로 변환, 원핫인코딩

텍스트, 스크린샷, 폰트, 웹 페이지이(가) 표시된 사진

자동 생성된 설명

* + - 모든 값이 같거나 의미 없는 feature 제거 (7개 열)
    - 이진 데이터 값은 1과 0으로 변환 (3개 열 : 라벨, 공정완료여부, tool\_condition)
    - 원핫인코딩 처리 (2개 열 : 프로그램 번호, 현재 공정명)
    - 전처리 완료 후 데이터 프레임 : df

1. 데이터 전처리 완료 후 데이터프레임 정보 (df.info())

RangeIndex: 32048 entries, 0 to 32047

Data columns (total 60 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 X\_ActualPosition 32048 non-null float64

1 X\_ActualVelocity 32048 non-null float64

2 X\_ActualAcceleration 32048 non-null float64

3 X\_SetPosition 32048 non-null float64

4 X\_SetVelocity 32048 non-null float64

5 X\_SetAcceleration 32048 non-null float64

6 X\_CurrentFeedback 32048 non-null float64

7 X\_DCBusVoltage 32048 non-null float64

8 X\_OutputCurrent 32048 non-null int64

9 X\_OutputVoltage 32048 non-null float64

10 X\_OutputPower 32048 non-null float64

11 Y\_ActualPosition 32048 non-null float64

12 Y\_ActualVelocity 32048 non-null float64

13 Y\_ActualAcceleration 32048 non-null float64

14 Y\_SetPosition 32048 non-null float64

15 Y\_SetVelocity 32048 non-null float64

16 Y\_SetAcceleration 32048 non-null float64

17 Y\_CurrentFeedback 32048 non-null float64

18 Y\_DCBusVoltage 32048 non-null float64

19 Y\_OutputCurrent 32048 non-null int64

20 Y\_OutputVoltage 32048 non-null float64

21 Y\_OutputPower 32048 non-null float64

22 Z\_ActualPosition 32048 non-null float64

23 Z\_ActualVelocity 32048 non-null float64

24 Z\_ActualAcceleration 32048 non-null float64

25 Z\_SetPosition 32048 non-null float64

26 Z\_SetVelocity 32048 non-null float64

27 Z\_SetAcceleration 32048 non-null float64

28 S\_ActualPosition 32048 non-null float64

29 S\_ActualVelocity 32048 non-null float64

30 S\_ActualAcceleration 32048 non-null float64

31 S\_SetPosition 32048 non-null float64

32 S\_SetVelocity 32048 non-null float64

33 S\_SetAcceleration 32048 non-null float64

34 S\_CurrentFeedback 32048 non-null float64

35 S\_DCBusVoltage 32048 non-null float64

36 S\_OutputCurrent 32048 non-null int64

37 S\_OutputVoltage 32048 non-null float64

38 S\_OutputPower 32048 non-null float64

39 S\_SystemInertia 32048 non-null float64

40 M\_sequence\_number 32048 non-null int64

41 M\_CURRENT\_FEEDRATE 32048 non-null int64

42 feedrate 32048 non-null int64

43 clamp\_pressure 32048 non-null float64

44 tool\_condition 32048 non-null int64

45 machining\_finalized 32048 non-null int64

46 target 32048 non-null int64

47 process\_End 32048 non-null uint8

48 process\_Layer 1 Down 32048 non-null uint8

49 process\_Layer 1 Up 32048 non-null uint8

50 process\_Layer 2 Down 32048 non-null uint8

51 process\_Layer 2 Up 32048 non-null uint8

52 process\_Layer 3 Down 32048 non-null uint8

53 process\_Layer 3 Up 32048 non-null uint8

54 process\_Prep 32048 non-null uint8

55 process\_Repositioning 32048 non-null uint8

56 process\_Starting 32048 non-null uint8

57 p\_num\_0 32048 non-null uint8

58 p\_num\_1 32048 non-null uint8

59 p\_num\_4 32048 non-null uint8

dtypes: float64(38), int64(9), uint8(13)

* + - 모든 Feature가 숫자형태로 변환 되어 학습 준비 완료
* **데이터 분석 : 공정 미완료 데이터 분석, 결과에 영향을 주는 feature 확인**

1. 공정 미완료 데이터에 대한 분석 (학습에 포함 시켜야 할지 확인)
   * 공정 미완료 데이터는 외관 검사가 진행되지 않았기 때문에 ‘fail’로 일괄 처리 했음. 불량의 공정 조건을 찾는데 학습 노이즈가 될 수 있음
   * 각 Feature 별로 공정 완료/미완료 클래스를 구분하여 평균값 분포 확인

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

* + - 공정 완료/미완료 클래스의 feature별 평균값에 차이가 있음
    - 단순히 시간적으로 공정이 미완료 된 것이 아니라, 조건의 차이로 인해 문제가 발생되어 공정이 완료되지 않은 것으로 보임
    - 따라서, 미완료 항목을 포함하여 학습 시키는 것이 맞음. (fail 클래스의 feature 특징을 가지고 있으므로)

1. 공정 완료/미완료 클래스의 현재 진행 공정 분포 확인
   * 공정 미완료 클래스의 공정 분포가 일부 앞 공정에만 쏠려 있을 경우 학습에 영향을 줄 수도 있어 확인이 필요
     + 만약 미완료 클래스의 공정이 모두 1,2번 공정에서 끝난다면 학습 결과 1,2번 공정의 데이터는 모두 fail로 판정할 확률이 높음

텍스트, 스크린샷, 다채로움, 원이(가) 표시된 사진

자동 생성된 설명

* + - 공정 완료 클래스는 현재 공정의 빈도가 균일함
    - 공정 미완료 클래스는 현재 공정의 빈도가 균일 하지는 않으나, 학습에는 문제 되지 않을 것으로 보임
    - 전체 클래스는 현재 공정의 빈도가 균일함

1. 전체 데이터의 feature 별 평균 값 분포 확인

**텍스트, 소프트웨어, 라인, 웹 페이지이(가) 표시된 사진

자동 생성된 설명**

* + - 라벨 클래스별로 feature값에 차이가 나타남 --> 모델 학습이 가능함
  + 라벨 클래스별 평균 값의 차이가 큰 feature 확인

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

클래스별 평균 값의 차이가 큰 Top3 feature

- machining\_finaliszed : 공정 완료 여부

- clamp\_pressure : 가공 소재를 Clamping 하는 압력

- feedrate : 가공 Tool의 이동 속도

1. 라벨(target) 과 상관계수 분포 확인

텍스트, 스크린샷, 라인, 디자인이(가) 표시된 사진

자동 생성된 설명

클래스별 평균 값의 차이가 큰 Top3 feature

- machining\_finaliszed : 공정 완료 여부

- clamp\_pressure : 가공 소재를 Clamping 하는 압력

- feedrate : 가공 Tool의 이동 속도

* + - target과 상관계수 높은 feature top10 항목과 target 클래스 별 평균값 차이가 큰 feature top10 은 일치함

-> 라벨 클래스별 평균 값 차이가 큰 feature가 클래스 결정에 영향을 준다.

1. 결과에 가장 영향을 많이 주는 feature 값이 클래스별로 어떻게 차이가 나는지 확인

**텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명**

**텍스트, 도표, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명**

**[clamp\_pressure]**

0 클래스 83%가 4.0

-> 클래스별 압력 값의 분포가 다르기 때문에 결과 예측 시 영향을 줄 수 있음

**[feedrate]**

0 클래스만 12, 15 값을 가짐

-> 결과를 예측할 때 직접적인 영향을 줄 수 있음

**텍스트, 도표, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명**

* **모델 학습 및 결과 예측 : 전체 feature 학습**

1. 학습 전 train, test 데이터셋 나누기, 라벨 열 나누기

텍스트, 스크린샷, 폰트, 문서이(가) 표시된 사진

자동 생성된 설명

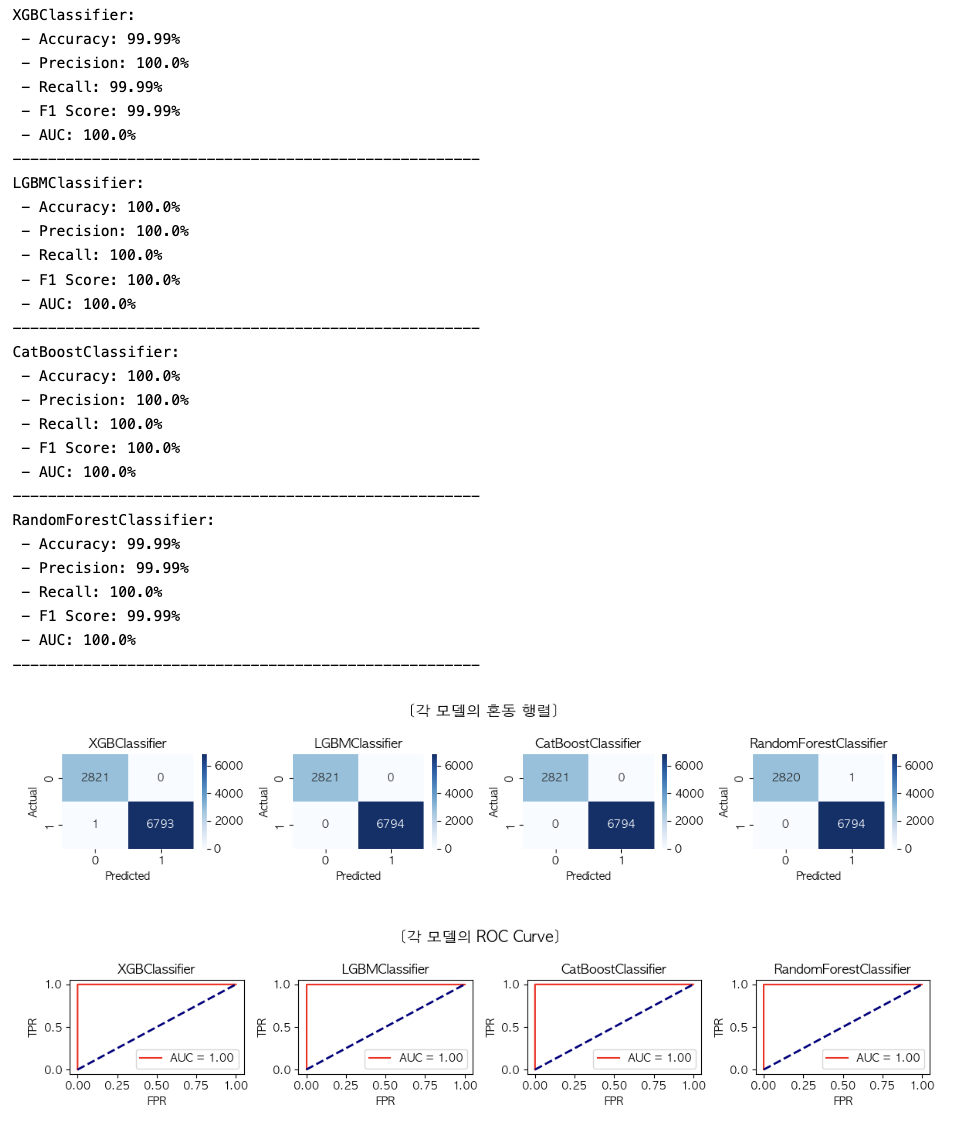
* + - test\_size = 0.3 (전체 데이터 중 30%를 test 데이터 셋으로)
    - stratify = df[‘target’] : train, test 데이터셋을 분리한 이후에도 지정한 column의 클래스 비율을 일정하게 유지함
    - 학습 시킬 train 데이터의 feature는 59개 - x\_train shape : (22433, 59)

1. 머신러닝 4가지 모델 학습 및 결과 예측

텍스트, 스크린샷, 문서, 폰트이(가) 표시된 사진

자동 생성된 설명

1. 머신러닝 4가지 모델 결과 예측 및 평가



* + - 머신러닝 모델로 학습 시 평가 정확도는 100% (오답개수 0 또는 1)
    - 설비에서 자동저장 되는 데이터 또는 설비 설정 값 이외의 데이터는 삭제 후 학습 시켜볼 필요 있음 (아래 2개의 feature)
      * machining\_finalized : 공정 완료 여부 - (관리자가 별도로 기록)
      * tool\_condition : 일정시간 사용한 Tool인지 새로운 Tool 인지 – (관리자가 별도로 기록)
* **설비에서 자동 저장 되는 데이터와 설비 설정 값만 사용하여 재학습 (2개 feature 삭제)**

1. 수동으로 기록하는 2개의 feature는 삭제
   * 2개 feature 삭제 : 'tool\_condition' , 'machining\_finalized'

텍스트, 스크린샷, 문서, 폰트이(가) 표시된 사진

자동 생성된 설명

* + - 학습데이터의 feature 개수 : 57개 - x\_train shape : (22433, 57)

1. 4가지 머신러닝 모델 학습 및 결과 예측
   * 학습 및 평가 코드 동일함 생략

텍스트, 도표, 스크린샷, 평행이(가) 표시된 사진

자동 생성된 설명

* + - 정확도 : 99.9 ~ 100%
    - 수동으로 얻어지는 feature 제거 후에도 CatBoost 모델은 정확도 100% 기록

1. Logistic regression 모델 학습 및 결과 예측

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

* + - Logistic regression 모델 정확도 82.7%로 성능 좋지 못함
    - 로지스틱 회귀 모델이 성능이 떨어지는 원인 추정 :
      * 학습 데이터가 비선형 데이터, 불균형 데이터 (1과0 클래스 비율 7:3)

1. DNN (Deep Neural Network) 모델 학습 및 결과 예측

텍스트, 스크린샷, 메뉴, 편지이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트, 문서이(가) 표시된 사진

자동 생성된 설명

* 활성 함수 : 최적 함수 : LeakyReLU
  + Relu, swish, softplus에 비해 과적합이 덜 일어나고, 정확도도 가장 좋게 나타남
* Optimizer : 최적 optimizer : adam
  + SGD 사용 시 정확도 떨어짐
* Batch\_size : 최적 배치 수 : 128
  + 배치 수가 많을 수록 학습 속도는 빨라지나, 정확도가 떨어지는 경향이 있음
  + 실험결과 학습 속도와 정확도를 고려했을 때 128이 적합
* 최적모델 저장 기준(monitor) : 최적 기준 : ‘val\_sparse\_categorical\_accuracy’
  + ’val\_loss’ 를 기준으로 할 경우 정확도 떨어짐
* Epoch : 최적 Epoch : 400회
  + Epoch 300회 초과 되어도 성능이 개선 되지 않음

(단, SDG optimizer 사용했을 경우는 예외)

* Drop 비율 : drop 생략
  + 데이터 구조가 복잡하지 않아서 Drop 시키지 않아도 과적합 일어나지 않음
  + Drop 시킬 경우 정확도 0.2% 이상 떨어짐
* DNN 모델 학습 결과

텍스트, 스크린샷, 라인, 폰트이(가) 표시된 사진

자동 생성된 설명

* + Drop을 생략해서 초반 과적합 발생 하지만 후반으로 갈수록 정확도 높아짐
  + DNN 혼동행렬, ROC Curve

텍스트, 도표, 그래프, 라인이(가) 표시된 사진

자동 생성된 설명

* DNN 모델 평가
  + Accuracy: 99.74%
  + Precision: 99.88%
  + Recall: 99.75%
  + F1 Score: 99.82%
  + AUC: 99.99%

1. CNN (Convolutional Neural Network) 모델 학습 및 결과 예측

텍스트, 스크린샷, 문서, 폰트이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 문서, 폰트이(가) 표시된 사진

자동 생성된 설명

* + CNN 모델 학습 결과

텍스트, 스크린샷, 도표, 폰트이(가) 표시된 사진

자동 생성된 설명

* + CNN 혼동행렬, ROC Curve

텍스트, 스크린샷, 라인, 그래프이(가) 표시된 사진

자동 생성된 설명

* DNN 모델 평가
  + Accuracy: 99.74%
  + Precision: 99.87%
  + Recall: 99.76%
  + F1 Score: 99.82%
  + AUC: 99.99%
* **데이터 분석 및 학습 실험 결과 =====================================================**

1. 최적의 학습 모델
   * CNN (Convolutional Neural Network)
   * 정확도 : 97.2 ~ 97.5 %
2. 데이터 전처리 후 학습 결과 (3가지)
   * 데이터 전처리 효과 없음

(1) 평균 센서 값 높은 샘플 제거 (5개, 10개, 20개 행 제거)

(2) 403번 이후 센서만 학습 (98개 열로만 학습)

(3) target 상관관계 하위 센서 제거 (10개, 30개 열 제거)

1. 모델별 학습 결과 요약

텍스트, 스크린샷, 번호, 폰트이(가) 표시된 사진

자동 생성된 설명