SECOM 반도체 데이터 수율 예측

2020170850 김형진

목차

1. 프로젝트 개요

최근 제조 산업에서는 공정 자동화 및 품질 개선의 중요성이 증가하고 있으며, 특히 불량률을 낮추는 것은 생산 효율성과 고객 만족도 향상에 핵심적인 역할을 한다. 이에 따라 본 프로젝트에서는 제조 공정 데이터를 기반으로 제품의 불량 여부를 예측하는 이진 분류 모델을 개발하고, 이를 통해 공정 내 품질 이상 신호를 조기에 감지하는 데 목적을 둔다.

사용한 데이터는 다양한 센서 및 품질 측정 지표로 구성된 반도체 유사 공정 데이터로, 각 제품에 대한 수백 개의 공정 변수(feature)와 최종 불량 여부(label)가 포함되어 있다. 해당 데이터는 고차원의 특성을 가지며, 클래스 불균형 문제가 존재하여 불량 샘플이 전체에서 매우 적은 비율을 차지한다는 점이 모델 개발 과정의 주요 고려 사항이었다.

본 프로젝트의 주요 목표는 다음과 같다:

- 다양한 전처리 및 정규화 기법을 적용하여 모델 성능 향상 도모
- 단일 및 앙상블 분류 모델을 설계 및 하이퍼파라미터 튜닝 수행
- 모델별 성능 비교를 통해 최적의 예측 모델 선정
- 최종 선정된 모델 기반으로 변수 중요도 분석 및 공정 개선 인사이트 도출

이러한 접근을 통해 단순한 모델 성능 비교에 그치지 않고, 실질적인 **공정 품질 개선의 방향성**을 제시하고자 한다.

2. 데이터 설명 및 분할

본 프로젝트에서는 SECOM 공정 데이터를 활용하였다. 해당 데이터는 반도체 공정에서 수집된 센서 측정값으로 구성되며, 각 샘플은 제품 한 개에 해당한다.

- secom.data: 총 1,567개의 샘플, 590개의 센서 기반 특성(feature)
- secom_labels.data: 각 샘플의 품질 라벨 (-1: 양품, 1: 불량)

데이터는 학습과 성능 평가의 명확한 구분을 위해 사전에 분할하였으며, 이는 **데이터 누수(Data Leakage)를 방지**하고, 실질적인 일반화 성능을 확보하기 위함이다. 분할 비율은 다음과 같다:

- **Train : Validation : Test = 6 : 2 : 2**
- **Stratified Split** 기법을 사용하여 클래스 불균형이 각 세트에 동일하게 반영되도록 하였다.

또한, **결측값이 포함된 원본 상태에서 먼저 데이터를 분할**하고, 이후 전처리 과정은 오직 Train 데이터에 대해서만 수행하였다. 이를 통해 전처리 단계에서의 정보가 Validation/Test 데이터에 유입되는 것을 철저히 차단하였다.

각 데이터셋의 규모와 클래스 분포는 [표 1]과 같다.

데이터셋	샘플 수	변수 수	양품 비율	불량 비율
Train	940	590	93.4%	6.6%
Validation	313	590	93.3%	6.7%
Test	314	590	93.3%	6.7%

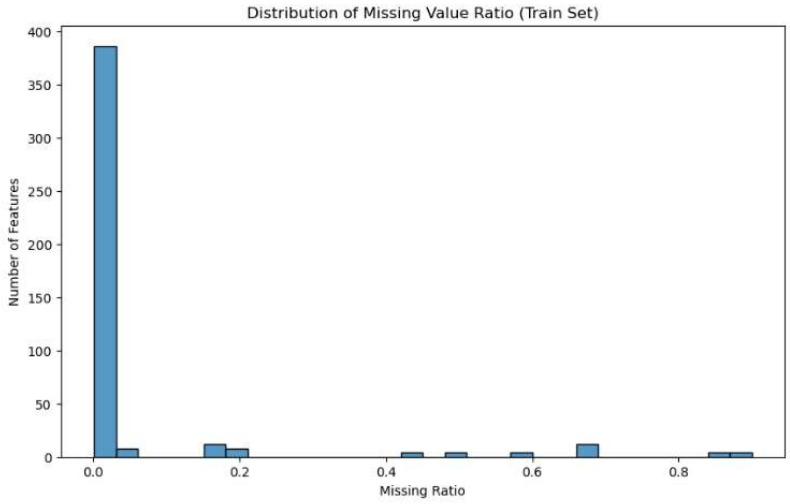
[표 1]

이와 같이 클래스 간 불균형(Class Imbalance)이 존재하므로, 이후의 학습 및 평가 과정에서는 **불균형 데이터를 고려한 평가 지표 및 모델링 전략**이 요구된다.

3. EDA (Exploratory Data Analysis)

탐색적 데이터 분석은 **오직 Train 데이터 기준**으로만 수행되었으며, 이는 Validation 및 Test 세트와의 정보 유입을 방지하기 위한 조치이다. EDA의 주요 목적은 데이터의 전반적 분포를 파악하고, 불량(Label = 1)과 양품(Label = -1)의 차이를 확인하여 모델링에 유의미한 통찰을 제공하는 데 있다.

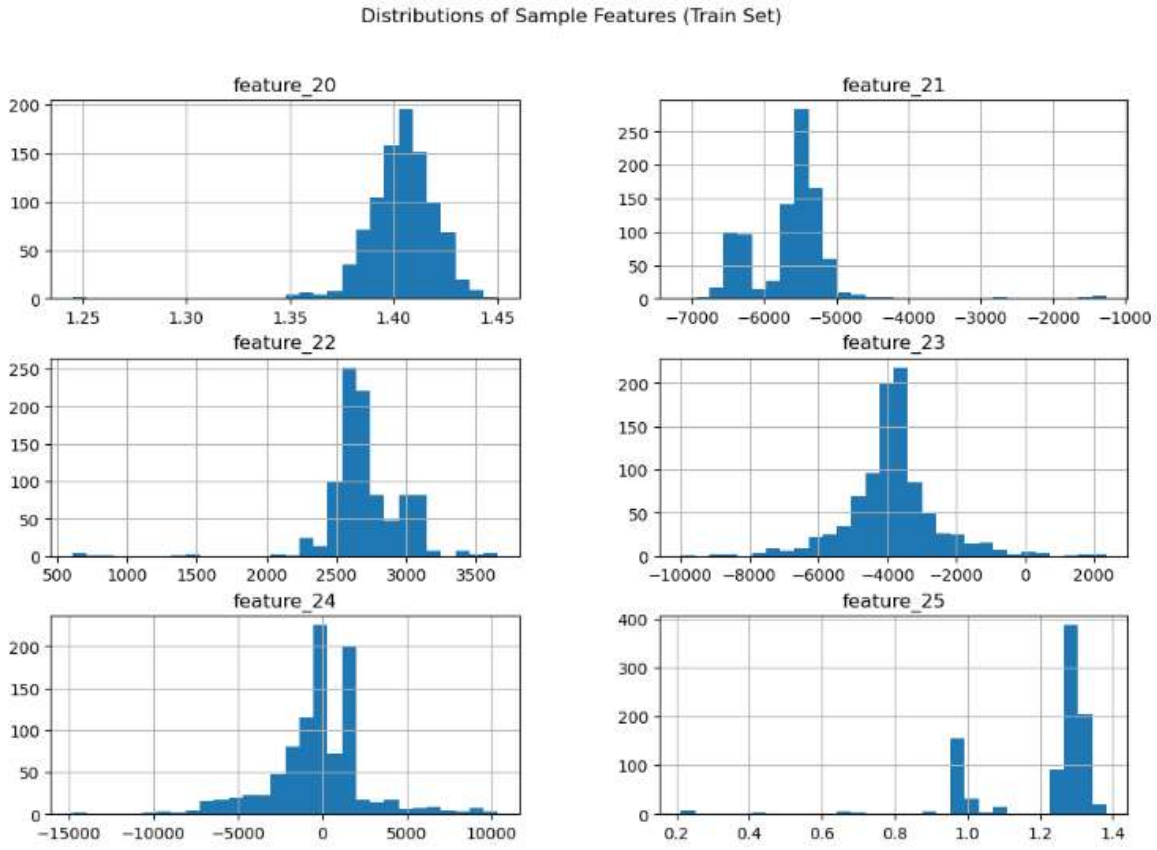
결측치 분석



[그림 1]

[그림 1]과 같이 약 380개 변수는 결측치가 거의 없다. 그러나 일부 변수는 결측률이 20%~90% 이상으로 매우 높다. 이를 통해 결측률이 50% 이상인 변수는 제거하는 것이 바람직하다고 판단된다

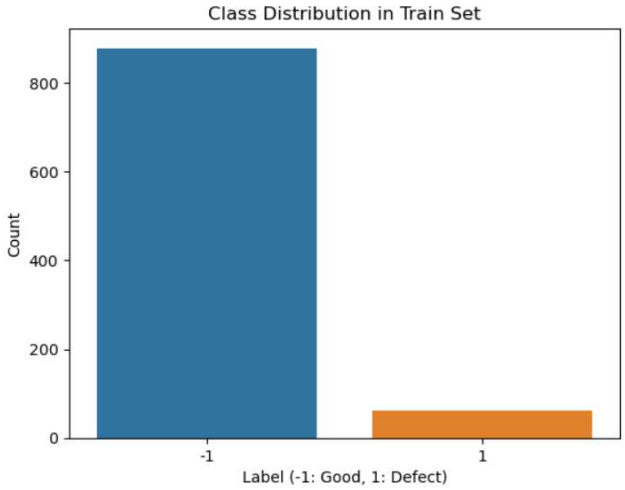
변수 분포 및 이상치



[그림 2]

[그림 2]는 결측치가 없는 컬럼 중 일부를 선택하여 시각화한 것으로 23번 변수를 빼고 대부분의 변수가 비대칭 분포(Skewed)를 띠며, 극단값(outlier)가 존재한다. 이를 통해 모델링 단계에서 정규화(Scaling)가 필요하다고 판단된다.

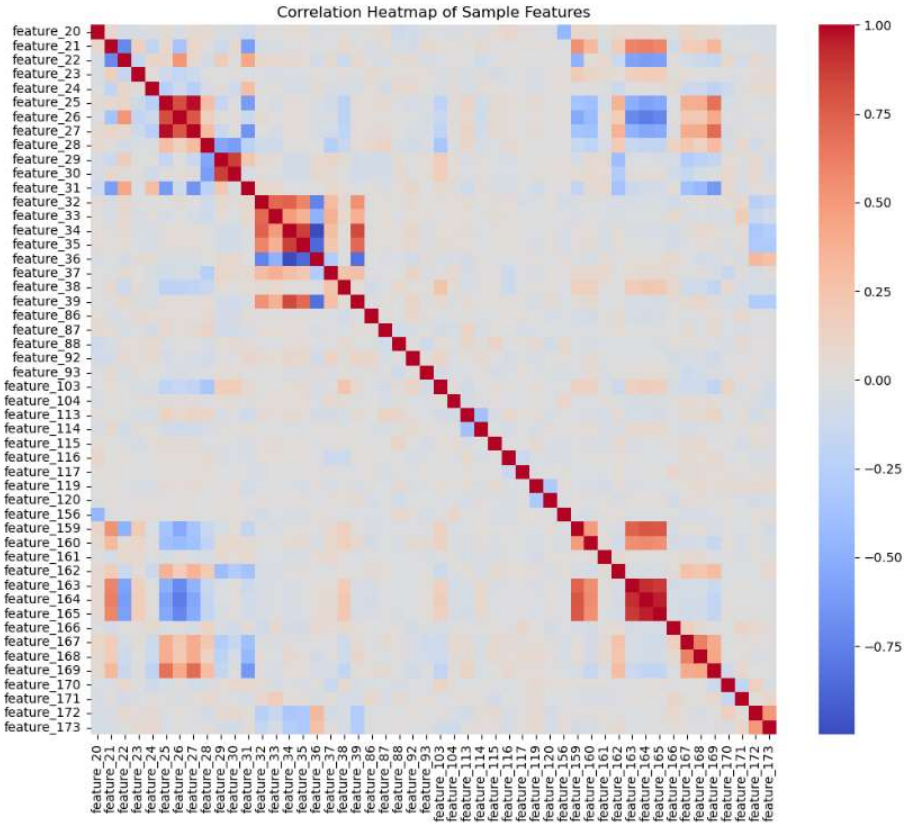
클래스 불균형



[그림 3]

[그림 3]과 같이 Train 데이터 기준으로 불량 비율은 약 **6.6% 수준으로** 극심한 클래스 불균형으로 인해 정확도(accuracy)만으로는 성능을 판단하기 어려우며, **F1-score, ROC AUC 등 평가 지표의 다면적 고려**가 요구된다. 또한 향후 모델 학습 시 **클래스 불균형 처리 전략(SMOTE 등)** 적용이 필요하다.

변수 간 상관관계



[그림 4]

[그림 4]를 보면 일부 feature 간의 상관계수가 **0.9 이상**으로 매우 높은 것을 알 수 있다. 이는 **다중공선성(multicollinearity)** 문제가 발생할 수 있어 Feature Selection나 PCA 같은 **차원 축소** 기법의 적용이 고려된다.

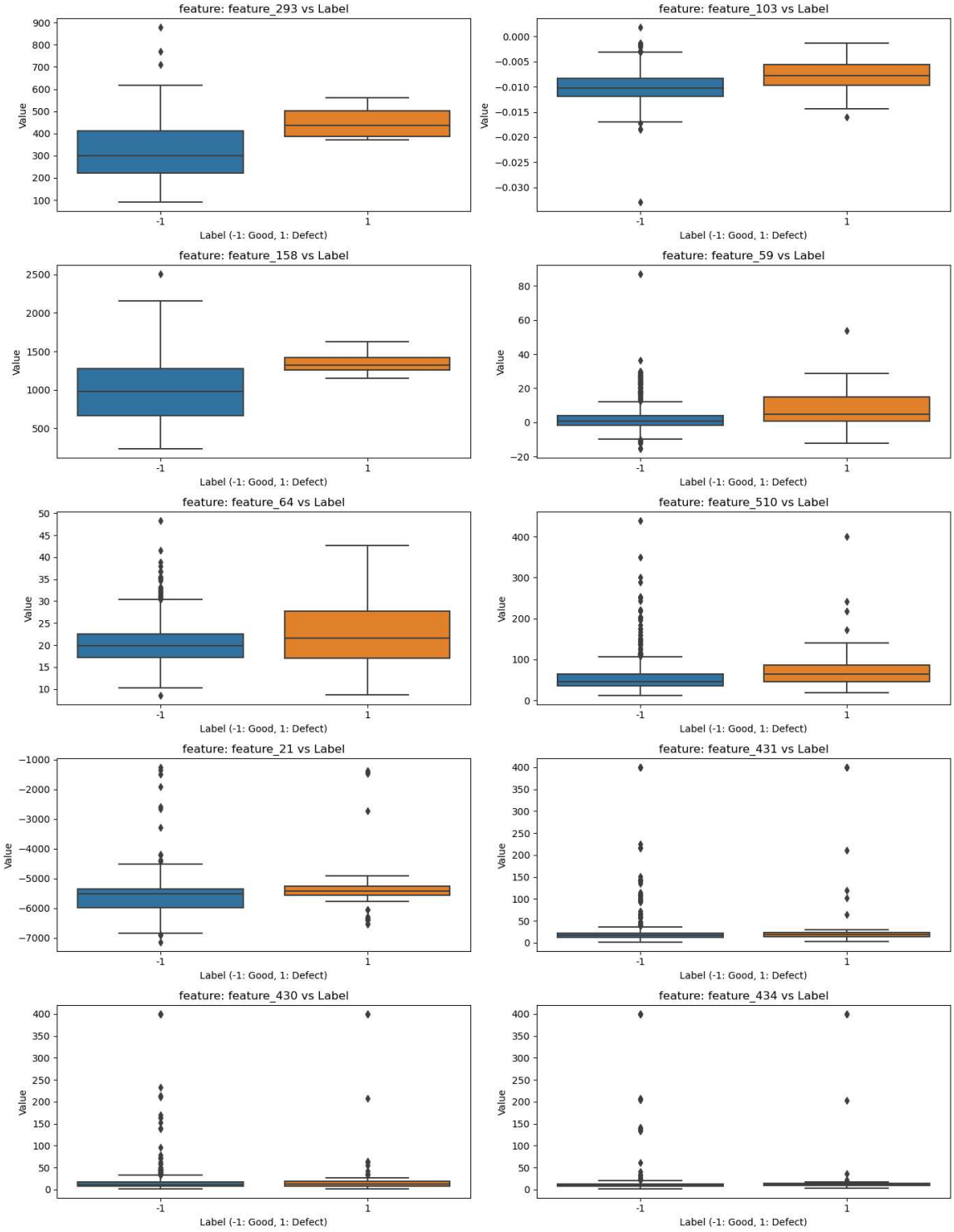
y(Label)과의 상관관계 분석

각 변수와 타겟 변수(y) 간의 상관계수를 계산한 결과, 절댓값 기준으로 가장 높은 상위 10개 변수는 다음과 같다:

['feature_293', 'feature_103', 'feature_158', 'feature_59', 'feature_64',

'feature_510', 'feature_21', 'feature_431', 'feature_430', 'feature_434']

[그림 5]는 이 변수들과 양품/불량 간 분포를 박스플롯으로 시각화한 것이다.


feature: feature_293 vs Label
feature: feature_103 vs Label
feature: feature_158 vs Label
feature: feature_59 vs Label
feature: feature_64 vs Label
feature: feature_510 vs Label
feature: feature_21 vs Label
feature: feature_431 vs Label
feature: feature_430 vs Label
feature: feature_434 vs Label
Value
Label (-1: Good, 1: Defect)

[그림 5]

라벨 변수와 각 변수별 특징을 요약하자면 [표 2]와 같다.

Feature	주요 해석 포인트
feature_293	불량 제품에서 중앙값이 확연히 높음 → 핵심 지표 가능성
feature_103	불량 클래스가 약간 더 높은 분포에 위치
feature_158	양품 vs 불량 간 중앙값 및 분산 차이 뚜렷
feature_59	불량에서 분포 넓음 → 공정의 불안정성 시사
feature_64	불량일 때 변동성이 큼
feature_510	차이는 있으나 이상치 영향 강함
feature_21	불량이 평균적으로 약간 높음
feature_431/430/434	차이를 알기 힘듦

[표 2]

EDA를 요약하자면 결측치 처리 및 이상치 대응이 필수적이며, 상관관계 기반 Feature Selection도 병행되어야 한다. 특히, feature_293, feature_158, feature_59 등은 불량 여부와의 관계가 뚜렷하여 중요 변수로 활용 가치가 높다. 본 결과는 이후 Feature Importance 및 변수 해석 단계와도 유기적으로 연결된다.

4. 전처리 (Preprocessing)

모델의 성능을 극대화하고 일반화 능력을 확보하기 위해 결측치 처리, 정규화, Feature selection, 클래스 불균형 처리와 같은 전처리 과정을 수행하였다. 특히 본 프로젝트는 불균형 데이터(Class Imbalance)를 다루므로, 정확도(accuracy)보다 **F1-score**를 성능 지표로 사용하여 각 전처리 단계의 효과를 평가하였다.

결측치 처리

[그림 1]을 봤을 때처럼 전체 590개 변수 중 390개에는 결측치가 거의 없지만, 일부 변수는 50% 이상의 결측률을 나타낸다. 이에 따라 결측치 처리는 다음의 두 단계로 구성되었다:

1. **50% 이상 결측치가 있는 변수 제거**
2. **남은 변수에 대해 다양한 대체 방법을 비교**

비교한 결측치 대체 방식은 다음과 같다:

- 평균 대체 (Mean Imputation)
- 중앙값 대체 (Median Imputation)
- KNN 기반 대체

각 방법의 성능은 **Decision Tree Classifier**를 기준으로 **5-Fold Cross Validation**을 통해 평가하였으며, 결과는 [표 3]과 같다.

결측치 처리 방법	F1-score
Mean	0.1660
Median	0.1594
KNN	0.1093

[표 3]

평균 대체(Mean Imputation)가 가장 우수한 성능을 보여 선택되었다. 따라서 Train 데이터 기준으로 평균값을 계산하여 fit하고, 이를 Validation 및 Test 데이터에 일관되게 적용하였다.

정규화 (Scaling)

센서 측정값은 변수마다 단위와 분포가 상이하므로, 학습 안정성을 위해 정규화를 수행하였고 다음 두 가지 방식을 비교하였다:

•**StandardScaler**: 평균 0, 표준편차 1로 정규화 (Z-score)

•**MinMaxScaler**: 최대 최소값을 기준으로 모든 값을 0~1 범위로 정규화

동일하게 **Decision Tree Classifier + 5-Fold CV + F1-score**로 평가한 결과는 [표 4]와 같다.

정규화 방법	F1-score
StandardScaler	0.1548
MinMaxScaler	0.1548

[표 4]

두 방식이 동일한 성능을 보였지만, 일반적으로 **StandardScaler**는 선형 모델 및 거리 기반 모델에서 보다 안정적인 학습을 가능하게 하므로 본 프로젝트에서는 이를 채택하였다. 또한 Validation 및 Test 데이터에도 일관되게 적용하였다.

Feature Selection (F-test 기반)

전체 변수 중 타겟 변수(y)와 유의미한 상관관계를 갖는 feature만 선별하여 과적합을 방지하고 해석 가능성을 높이고자 했다. 사용 기법은 변수와 y간의 F-test로 SelectKBest with f_classif (F-test)를 사용하였고 유의수준은 0.05로 설정하여 p-value가 0.05보다 작은 변수를 선택하였다. 그리고 마찬가지로 Validation 및 Test 데이터에 동일한 변수만 적용하였다.

상위 주요 변수 10개는 ['feature_14', 'feature_21', 'feature_22', 'feature_26', 'feature_28',

'feature_59', 'feature_63', 'feature_64', 'feature_65', 'feature_67']로 feature_59, feature_64, feature_21 등은 EDA 단계에서도 중요성이 관찰된 바 있으며, 이는 **해석과 성능 측면에서 모두 타당한 선택**임을 시사한다.

클래스 불균형 처리 (SMOTE)

Train 데이터에서 불량 제품(클래스 1)의 비율은 약 6.6%로 매우 낮은 수준이다. 이로 인해 모델은 양품 예측에만 치우칠 우려가 있으며, 이를 보완하기 위해 SMOTE(Synthetic Minority Over-sampling Technique)를 적용하였다. 데이터 누수 방지를 위해 **Train 데이터에만 적용**하였고 Validation/Test 데이터에는 사용하지 않았다.

5. 모델 성능 비교

본 프로젝트는 불균형한 제조 데이터에서 불량품을 효과적으로 예측하기 위해 다양한 분류 모델을 실험하고, 가장 성능이 우수한 모델을 선정하는 것을 목표로 하였다.

5.1 기본 파라미터로 모델 성능 비교 (Validation 기준)

먼저 기본 설정(default parameter)으로 다양한 모델을 학습한 후, 유의미한 성능을 보인 모델에 대해 하이퍼파라미터 튜닝을 수행하였다.

클래스 불균형을 고려하여 F1-score를 주요 기준으로 사용하였고 각각의 모델들의 성능은 [표 5]와 같다.

모델	F1-score (Validation)
Logistic Regression	0.1748
MLP	0.1702
Naive Bayes	0.0526
SVM	0.2456
Decision Tree	0.1455
KNN	0.0432

[표 5]

SVM, Logistic Regression, MLP에서 높은 성능을 보였고 Decision Tree에서 조금 낮은 성능을 보였다. 그러나 Naïve Bayes는 데이터의 특성상 변수가 많아 변수간 독립성을 보장하기 힘들어 성능이 낮게 나온 것으로 보인다. 또한, KNN은 y 데이터를 사용하지 않은 비지도학습으로 성능이 떨어진 것으로 보인다.

따라서 **단일 모델 중에서 유의미한 성능**을 보인 SVM, Logistic Regression, MLP을 중심으로 **하이퍼파라미터 튜닝**을 진행하였다. 이후 단일 모델 중 성능이 가장 우수한 것들을 기반으로 앙상블 전략(Voting / Stacking 등)을 구성하였다.

그리고 RandomForest, AdaBoost, BaggingClassifier, XGBoost는 별도로 튜닝 및 성능 향상을 시도하여 비교 분석하였고 이때 Base model로 Decision Tree가 사용되었다.

5.2 단일 모델 하이퍼파라미터 튜닝 및 성능 비교

본 절에서는 SVM, Logistic Regression, MLP를 중심으로 하이퍼파라미터 튜닝을 통해 성능을 개선하고, 테스트 데이터에서 불량품 탐지 성능을 비교하였다. 각 모델은 클래스 불균형을 고려하여

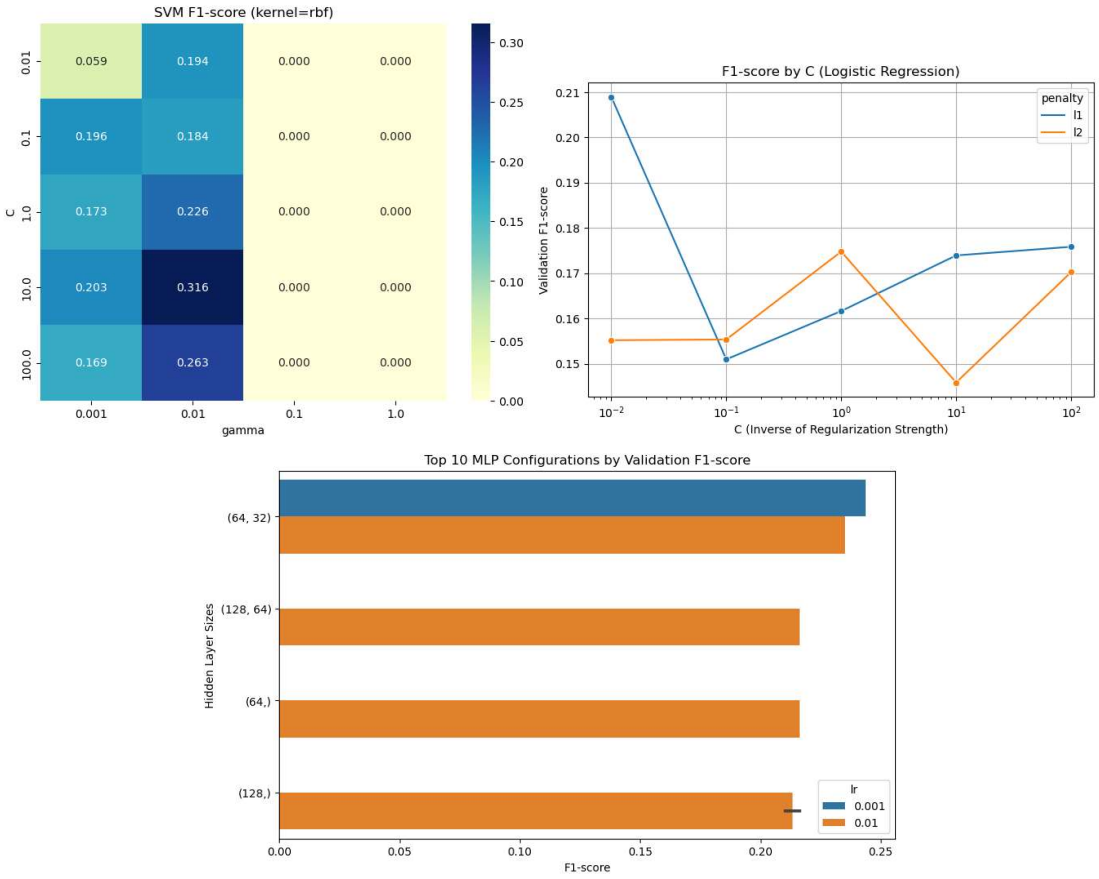
F1-score (특히 불량 클래스 기준)를 중심으로 평가되었으며, Precision, Recall, ROC AUC 등의 지표를 함께 분석하였다.

튜닝 범위 개요:

Model	주요 튜닝 파라미터
SVM	C: [5, 8, 10, 12, 15] gamma: [0.005, 0.0075, 0.01, 0.0125, 0.015] kernel='rbf'
Logistic Regression	C: [0.01, 0.1, 1, 10, 100] penalty: ['l1', 'l2'] solver='liblinear'
MLP	hidden_layer_sizes: [(64,), (128,), (64,32), (128,64)] alpha: [0.0001, 0.001, 0.01] learning_rate_init: [0.001, 0.01] activation: ['relu']

[표 6]

각각의 모델에 대해서 f1-score는 [그림 6]과 같다.



[그림 6]

최적 파라미터:

Model	최적 파라미터
SVM	C: [10] gamma: [0.01] kernel='rbf'
Logistic Regression	C: [0.01] penalty: ['l1'] solver='liblinear'
MLP	hidden_layer_sizes: [(64,32)] alpha: [0.001] learning_rate_init: [0.001] activation: ['relu']

[표 7]

Confusion Matrix:

SVM Confusion Matrix:

Actual / Pred	Good (0)	Defect (1)
Good (0)	284	9
Defect (1)	18	3

[표 8]

Logistic Regression Confusion Matrix:

Actual / Pred	Good (0)	Defect (1)
Good (0)	241	52
Defect (1)	10	11

[표 9]

MLP Confusion Matrix:

Actual / Pred	Good (0)	Defect (1)
Good (0)	285	8
Defect (1)	18	3

[표 10]

Test 성능 모음 (Defect class 기준):

Model	Accuracy	Precision	Recall	F1-score	ROC AUC
SVM	0.91	0.25	0.14	0.18	0.556
Logistic Regression	0.80	0.17	0.52	0.26	0.673
MLP	0.92	0.27	0.14	0.19	0.558

[표 11]

Logistic Regression은 불량 제품(Defect)의 Recall이 0.52로 가장 높아 실제 불량을 가장 많이 탐지할 수 있었으며, 전체 Accuracy는 0.80으로 양호한 수준이다. 반면 MLP는 Precision이 0.27로 가

장 높아 예측한 불량 중 실제 불량 비율이 가장 높았고, Accuracy는 0.92로 가장 뛰어났다. SVM은 보수적인 예측으로 인해 전체 Accuracy는 높았지만 Recall이 낮아 불량 탐지 성능은 떨어졌다. 각 모델은 서로 다른 강점을 가지고 있어 향후 Voting이나 Stacking과 같은 앙상블 전략을 통해 성능을 보완할 수 있는 가능성을 보여준다.

5.3 단일 모델 분석 기반 앙상블 모델 분석

SVM, Logistic Regression, MLP 세 모델을 기반으로 최적의 파라미터로 셋팅한 후 Hard Voting, Soft Voting, Stacking Classifier를 구성하여 성능을 비교하였다.

Confusion Matrix:

Hard Voting:

Actual / Pred	Good (0)	Defect (1)
Good (0)	282	11
Defect (1)	18	3

[표 12]

Soft Voting:

Actual / Pred	Good (0)	Defect (1)
Good (0)	286	7
Defect (1)	18	3

[표 13]

Stacking:

Actual / Pred	Good (0)	Defect (1)
Good (0)	290	3
Defect (1)	18	3

[표 14]

Test 성능 모음 (Defect class 기준):

Model	Accuracy	Precision	Recall	F1-score	ROC AUC
SVM	0.91	0.25	0.14	0.18	0.556
Logistic Regression	0.80	0.17	0.52	0.26	0.673
MLP	0.92	0.27	0.14	0.19	0.558
Hard Voting	0.91	0.21	0.14	0.17	0.553
Soft Voting	0.92	0.30	0.14	0.19	0.559
Stacking	0.93	0.50	0.14	0.22	0.566

[표 15]

해석 및 결론:

Hard Voting은 보수적인 예측 모델들의 성향이 반영되어 Recall이 낮고 Precision도 낮았다.
Soft Voting은 각 모델의 장점을 평균적으로 반영하며 Precision이 개선되었고, 안정적인 Accuracy를 보였다. Stacking은 Precision이 가장 높았으며, 전체 Accuracy도 가장 높아 성능 면에서 가장 우수한 결과를 보였다. Recall은 여전히 낮았지만 Logistic Regression를 제외한 다른 앙상블 방법이나 단일 모델과 동일한 수준이었다.
특히 Stacking은 False Positive 수가 가장 적었고, Precision 또한 가장 높아 실제 적용 가능성 측면에서 유리하다.

5.4 Decision Tree 기반 앙상블 모델 분석

튜닝 범위 개요:

Model	주요 튜닝 파라미터
Random Forest	n_estimators: [100, 200] max_depth: [3, 5, 7] min_samples_split: [2, 5] max_features: ['sqrt', 'log2'] class_weight='balanced'
AdaBoost	base_estimator: DecisionTreeClassifier(max_depth=1~3) n_estimators: [50, 100, 200] learning_rate: [0.001, 0.01, 0.1, 1.0]
Bagging	base_estimator: DecisionTreeClassifier(max_depth=5) n_estimators: 100 max_samples: 1.0
XGBoost	n_estimators: [50, 100, 200] max_depth: [3, 5, 7] learning_rate: [0.01, 0.05, 0.1] subsample: [0.8, 1.0] colsample_bytree: [0.8, 1.0]

[표 16]

최적 파라미터 요약:

Model	최적 파라미터
Random Forest	n_estimators=200, max_depth=5, min_samples_split=2, max_features='sqrt', class_weight='balanced'
AdaBoost	base_estimator=DecisionTree(max_depth=2), learning_rate=0.01, n_estimators=200
Bagging	base_estimator=DecisionTree(max_depth=5), max_samples=1.0, n_estimators=100
XGBoost	n_estimators=100, max_depth=3, learning_rate=0.1, subsample=0.8, colsample_bytree=0.8

[표 17]

Confusion Matrix:

Random Forest:

Actual / Pred	Good (0)	Defect (1)
Good (0)	270	23
Defect (1)	15	6

[표 18]

AdaBoost:

Actual / Pred	Good (0)	Defect (1)
Good (0)	270	23
Defect (1)	16	5

[표 19]

Bagging:

Actual / Pred	Good (0)	Defect (1)
Good (0)	278	15
Defect (1)	15	6

[표 20]

XGBoost

Actual / Pred	Good (0)	Defect (1)
Good (0)	282	11
Defect (1)	18	3

[표 21]

Test 성능 모음 (Defect class 기준):

Model	Accuracy	Precision	Recall	F1-score	ROC AUC
SVM	0.91	0.25	0.14	0.18	0.556
Logistic Regression	0.80	0.17	0.52	0.26	0.673
MLP	0.92	0.27	0.14	0.19	0.558
Hard Voting	0.91	0.21	0.14	0.17	0.553
Soft Voting	0.92	0.30	0.14	0.19	0.559
Stacking	0.93	0.50	0.14	0.22	0.566
Random Forest	0.88	0.21	0.29	0.24	0.748
AdaBoost	0.88	0.18	0.24	0.20	0.671

Bagging	0.90	0.29	0.29	0.29	0.757
XGBoost	0.91	0.21	0.14	0.17	0.553

[표 22]

Bagging Classifier는 Defect 클래스에서 Precision, Recall, F1-score, ROC AUC 모두 가장 우수한 성능을 보여주었으며, 전체 Accuracy도 90%로 안정적인 결과를 나타냈다. Random Forest는 다소 불안정하지만 높은 ROC AUC(0.75)를 보여주었고, AdaBoost는 보수적인 예측 성향을 보여주며 Precision이 낮은 편이었다. XGBoost는 전체 정확도는 높았으나 Defect 탐지 능력에서는 다소 약한 성능을 보였다. Stacking은 Precision이 높고 Accuracy도 93%로 최고 수준이지만 Recall은 개선되지 않았다. 이러한 분석을 기반으로 최종 모델 선정에 들어간다.

6. 최종 모델 선정

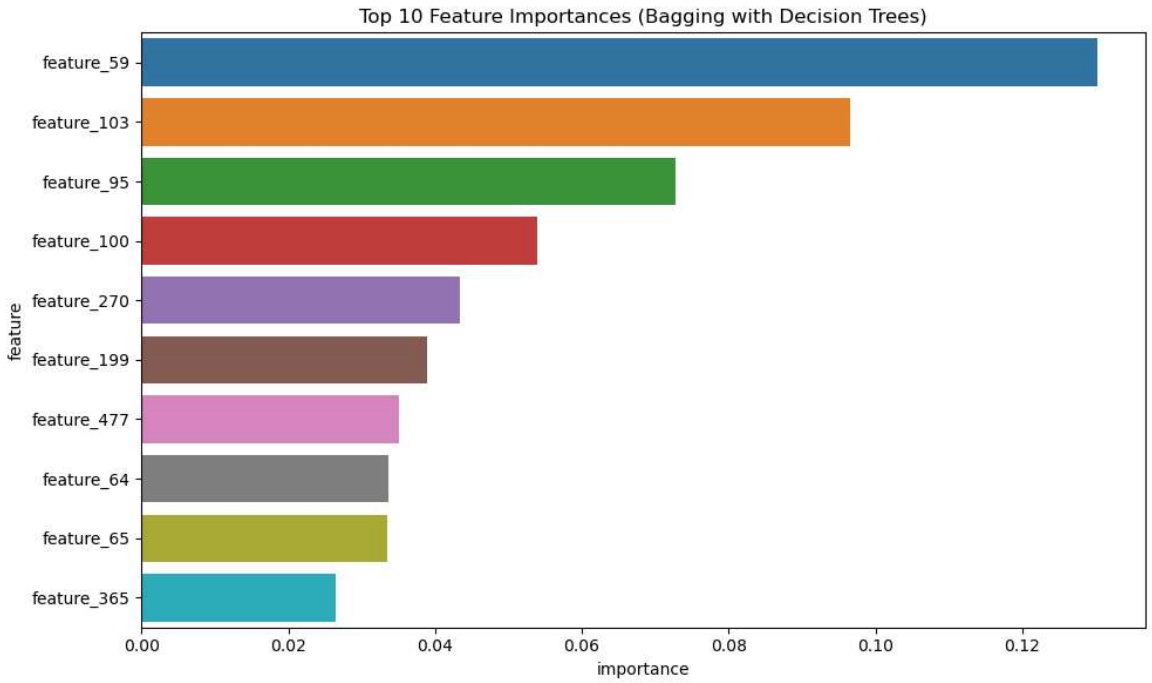
본 프로젝트에서는 다양한 단일 및 앙상블 모델에 대해 하이퍼파라미터 튜닝과 성능 평가를 진행하였으며, 그 결과를 바탕으로 최종 모델을 선정하였다. 모델 선정의 핵심 기준은 불량(Defect) 클래스에 대한 **F1-score, Precision, Recall의 균형성**, **ROC AUC**, 그리고 **Accuracy**를 종합적으로 고려하는 것이었다.

그 결과, Bagging Classifier with Decision Tree(max_depth=5)는 Accuracy 0.90으로 높은 전체 정확도를 보이는 동시에, Precision(0.29), Recall(0.29), F1-score(0.29)에서 모두 높은 성능을 보였다. 특히, ROC AUC가 0.76으로 모든 모델 중 가장 높게 나타나 불량 여부를 가장 잘 구분하는 모델임을 확인할 수 있었다. 이는 Decision Tree 기반이라는 단순성에도 불구하고 Bagging 기법을 통해 모델의 분산을 효과적으로 줄여 안정성과 일반화 성능을 확보한 결과이다.

다른 후보 모델들과 비교해도 Bagging의 우수성이 분명하다. 예를 들어, Stacking Classifier는 Accuracy가 0.93으로 가장 높고 Precision(0.50)도 뛰어났으나, Recall은 0.14로 매우 낮아 불량 탐지 목적에는 적합하지 않았다. 불량이라고 예측한 경우에 맞을 확률은 높지만, 실제 불량 샘플의 대부분을 놓친다는 점에서 한계가 있었다. 반면, Random Forest와 AdaBoost는 Recall이 각각 0.29, 0.24로 상대적으로 나았지만, F1-score 및 ROC AUC 측면에서 Bagging보다 낮은 성능을 기록하였다. 특히 XGBoost나 SVM과 같은 모델은 Recall이 0.14로 낮고, 불균형한 데이터 구조에서 과적합 가능성이 존재하는 점이 확인되었다.

결론적으로, 본 연구의 목적이 전체 정확도보다는 **불량 클래스(label=1)의 정밀한 탐지**에 있는 점을 고려하면, **F1-score와 ROC AUC 중심의 평가 지표를 기준으로 Bagging Classifier를 최종 모델로 선정**하는 것이 가장 타당하다. 이 모델은 불량 샘플을 놓치지 않으면서도 과도한 False Positive 없이 안정적으로 분류할 수 있어, 실제 공정에 적용하기에도 적합한 선택으로 판단된다.

7. 변수 해석 및 공정 인사이트 도출



[그림 7]

Bagging 모델의 feature importance 분석 결과에 따르면, 공정 변수 중 feature_59, feature_103, feature_95가 불량 예측에 가장 큰 영향을 미치는 것으로 나타났다.

이 중 feature_59는 중요도가 가장 높았을 뿐 아니라, 상관관계 분석에서도 불량 샘플에서 분포가 넓고 변동성이 큰 특징을 보여 공정의 불안정성과 직결되는 지표일 가능성이 높다. 또한 이 변수는 F-test 상위 10개 변수에도 포함되어 있어, 통계적으로 클래스 간 차이도 유의미한 것으로 확인되었다.

feature_103 역시 중요도 2위이며, 상관관계 분석 결과 불량 클래스에서 값이 더 높게 분포하는 경향을 보였고, feature_64, feature_65도 importance 상위권에 위치하며 각각 F-test 유의성과 불량 시 높은 변동성을 함께 나타낸다.

이는 모델 기반 중요도와 통계 기반 중요도가 동시에 높게 나타나는 변수들이 실제 공정에서 불량을 유발할 수 있는 핵심 인자일 가능성을 시사한다. 특히 feature_64는 F-test, 상관관계, 모델 중요도 모두에서 상위권을 차지하고 있어, 실시간 모니터링 대상 변수로 고려될 수 있다.

결론적으로, feature_59, feature_103, feature_64는 머신러닝 기반 예측과 통계 기반 검정을 모두 고려했을 때, 불량을 사전에 탐지하고 공정을 제어하기 위한 주요 관리 지표로 설정할 수 있으며, 이들을 중심으로 한 원인 분석 및 임계값 설정이 실질적인 공정 개선으로 이어질 수 있다.