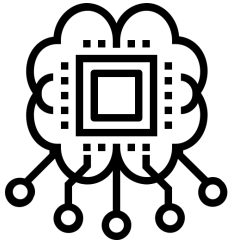
데이터마이닝 프로젝트

20191612 윤기웅

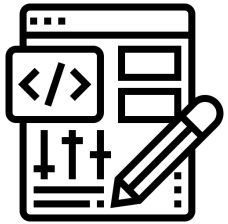
20191588 박성진

데이터마이닝



윤기웅

보고서 작성
AI Modeling



박성진

보고서 작성
데이터 분석

CONTENTS

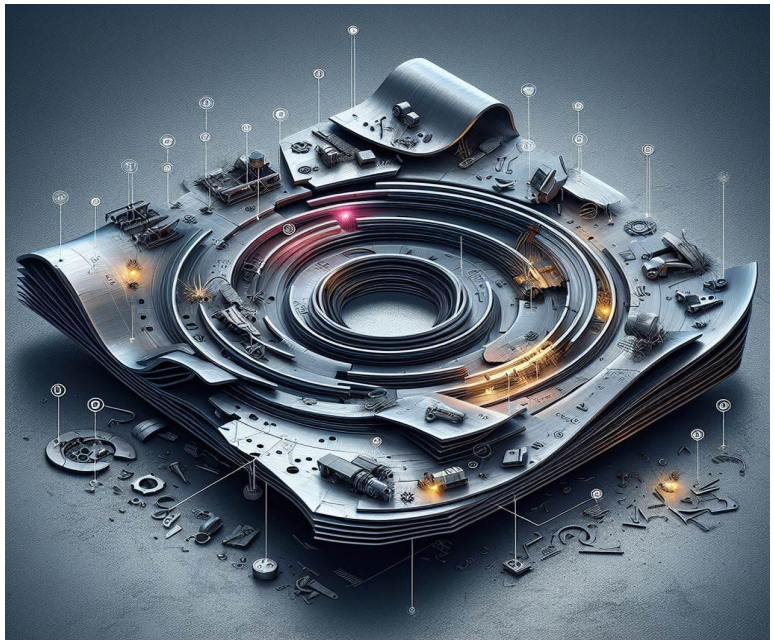
01 프로젝트 소개

kaggle

Steel Plate Defect Prediction

Playground Series - Season 4, Episode 3

Overview Data Code Models Discussion Leaderboard Rules Team Submissions



01 프로젝트 소개

Competition Host: Kaggle

Timeline : March 1, 2024 ~ March 31, 2024

Evaluation: ROC_AUC_score

Participation: 2199 Teams

Rank : 74th (Top 4%)

02 프로젝트 목적 및 설명

kaggle

[르포] 포스코 미래 책임질 고부가가치 전기강판 공장… 최고 품질 위해 '구슬땀'

광양=김성현 기자
입력 2023-07-06 11:07 | 수정 2023-07-06 15:32

포스코 고효율 전기강판 'Hyper NO'로 전기차 시장 주도

전기에너지→회전 에너지 변화과정서 발생하는 에너지 손실
일반 전기강판 대비 30% 이상 낮아 모터 효율 상승시켜

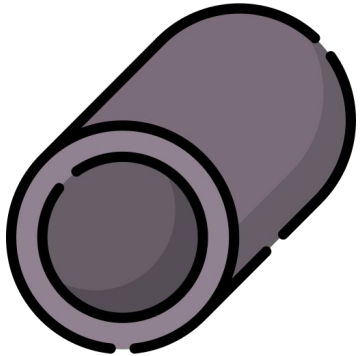
스페셜 > 특집섹션

'꿈의 강판' 기가스틸 생산 늘려 전기차용 소재시장 선점한다

포스코

02 프로젝트 목적 및 설명

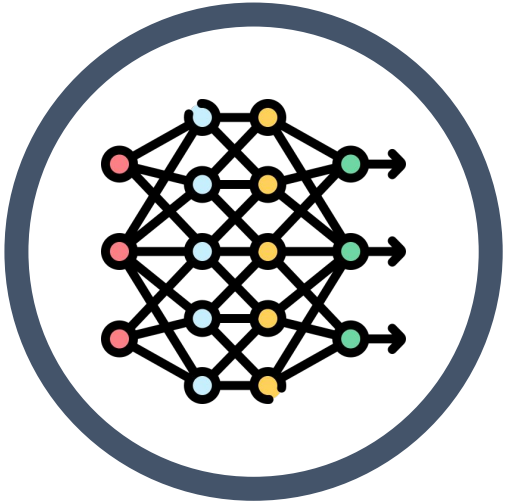
- 강판은 다양한 곳에 사용됨
- 전기차의 성장과 전기 강판의 수요 증가
- 품질의 중요성
- 품질 검증에 있어 많은 시간 소요



02 프로젝트 목적 및 설명

kaggle

강판 결함 분류



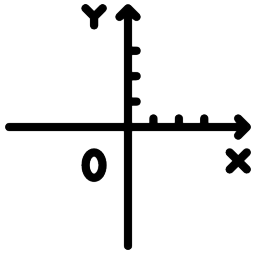
머신러닝을 통한 자동화

03 데이터 이해와 분석

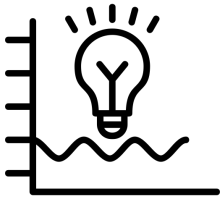
kaggle

- Features

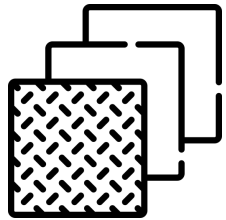
- Coordinate: X/Y 좌표 값



- Size : 결함의 크기



- Luminosity : 결함의 밝기



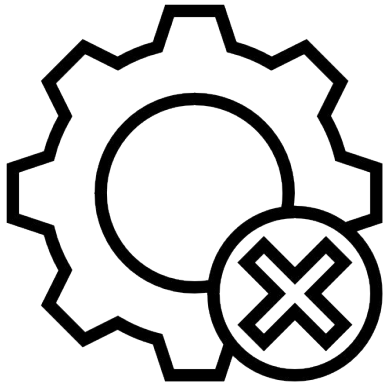
- Material & Index : 강판에 사용된 재료, 두께, 인덱스

- Etc: 크기와 인덱스에 Log 또는 Sigmoid 적용

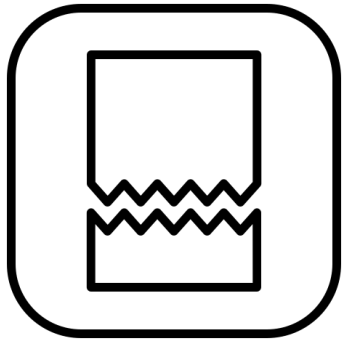
03 데이터 이해와 분석

kaggle

- **Targets:** 7개의 결함에 속할 확률을 출력하는 Multi-Label 분류

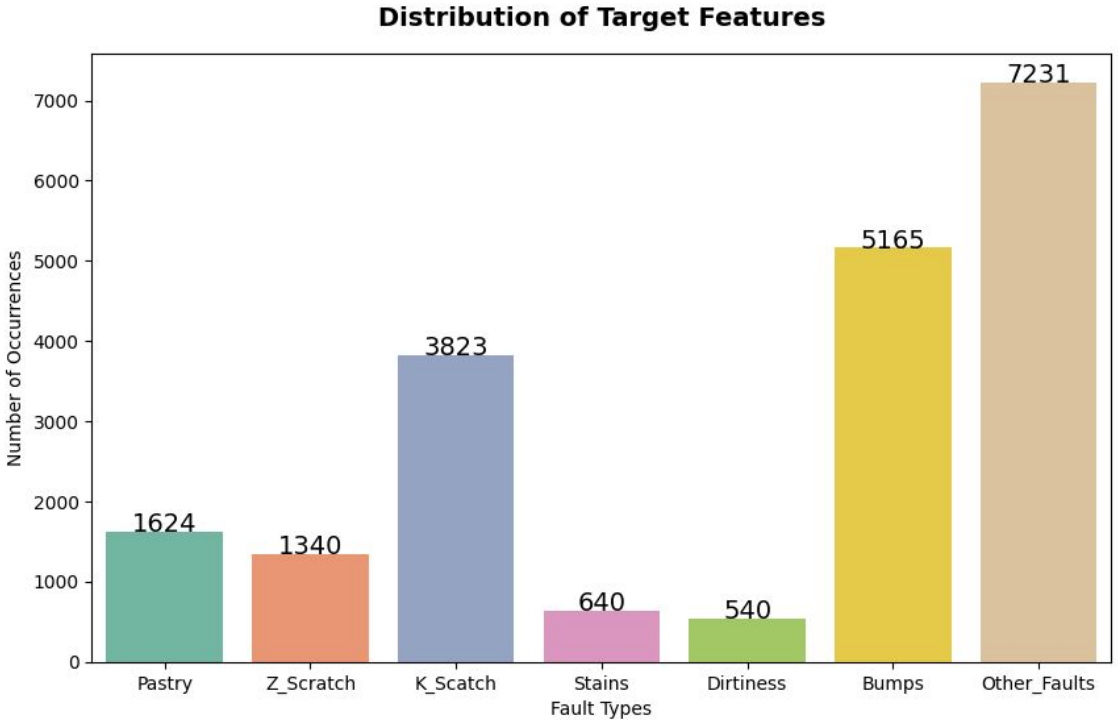


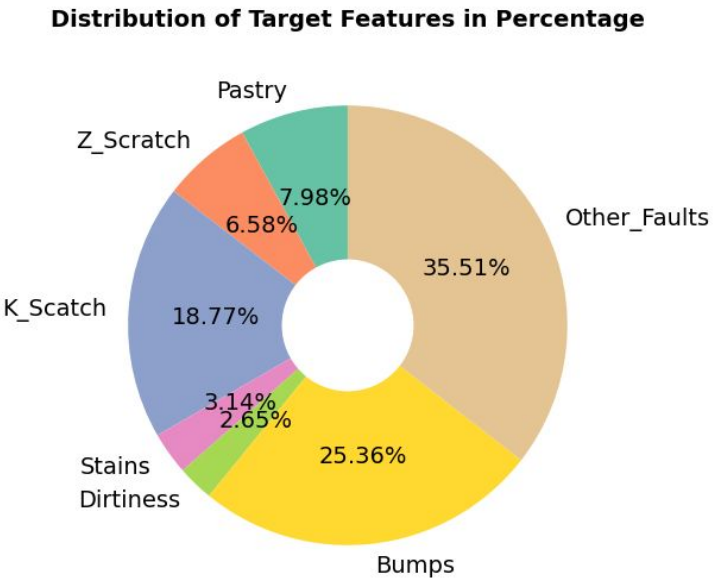
- Pastry: 표면에 발생한 작은 불규칙성
- Z_Scratch: 회전 방향과 평행하게 생긴 마모
- K_Scratch: 회전 방향과 수직으로 생긴 마모
- Stains: 변색되거나 녹슨 영역
- Dirtiness: 먼지나 오염된 영역
- Bumps: 표면에 솟아오른 부분 또는 돌출된 영역
- Other faults: 그 외 기타 결함



03 데이터 이해와 분석

- Train set 타겟 분포

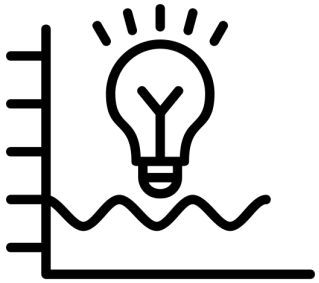




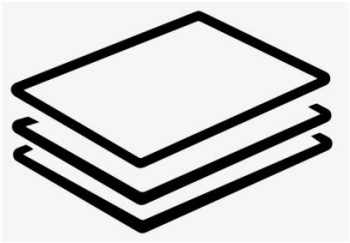
04 전처리 및 특성공학

kaggle

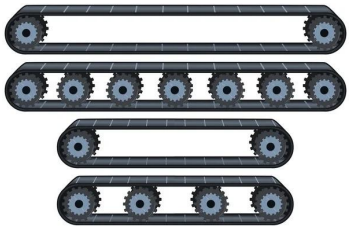
- 파생 변수



- Average_Luminosity: 평균 밝기 값 (밝기 총합 / 결함 면적)



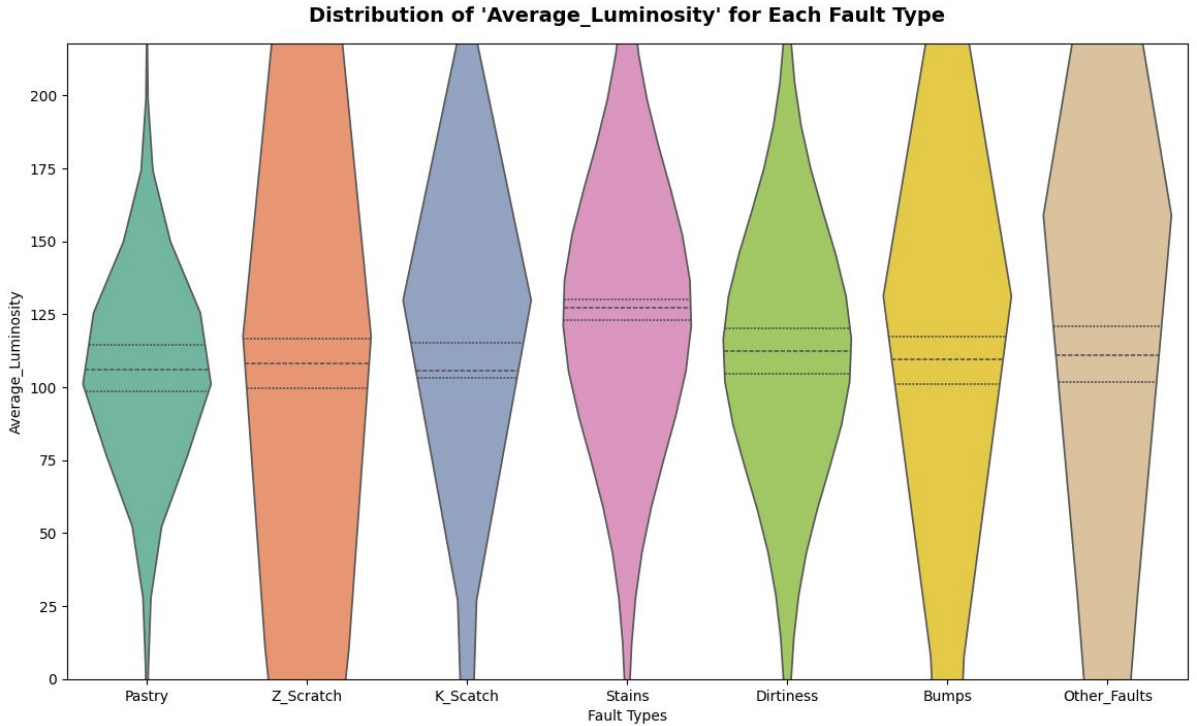
- X_Range*Pixels_Areas: 결함의 X축 길이와 면적의 곱



- Ratio_Length_Thickness: 컨베이어 길이 / 강판 두께

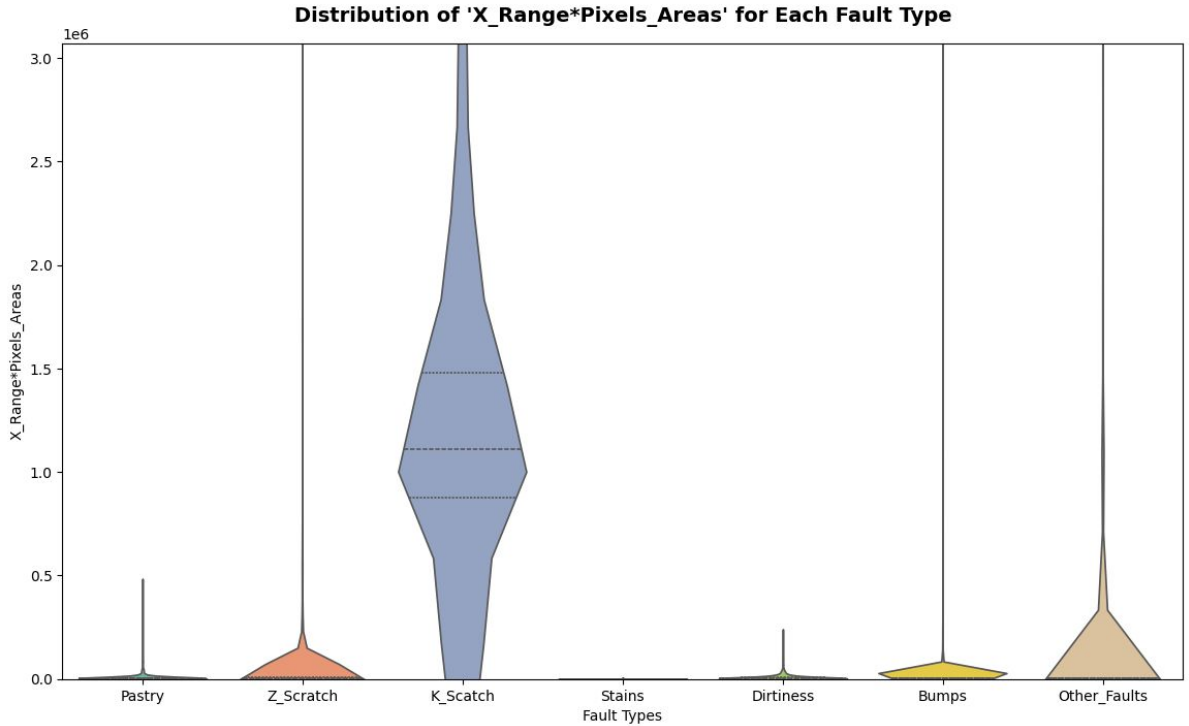
04 전처리 및 특성공학

- 파생 변수: Average_Luminosity (밝기의 총합 / 면적)



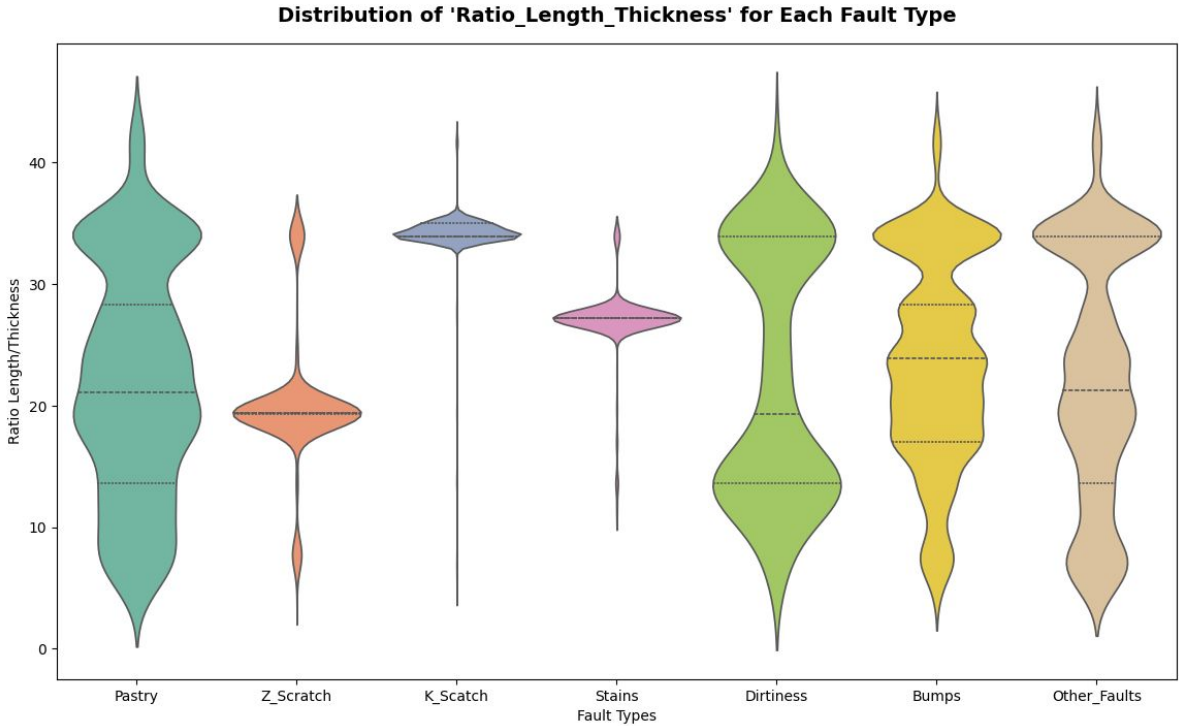
04 전처리 및 특성공학

- 파생 변수: X_Range*Pixels_Areas (X축 길이 x 면적)



04 전처리 및 특성공학

- 파생 변수: Ratio_Length_Thickness (컨베이어 길이 / 강판 두께)



04 전처리 및 특성공학

kaggle

- Feature Engineering

```
# new features
def feature_engineering(data):
    data['Ratio_Length_Thickness'] = data['Length_of_Conveyer'] / data['Steel_Plate_Thickness']
    data['Average_Luminosity'] = data['Sum_of_Luminosity'] / data['Pixels_Areas']
    data['X_Range*Pixels_Areas'] = (data['X_Maximum'] - data['X_Minimum']) * data['Pixels_Areas']
    return data
train_data = feature_engineering(train_data)
test_data = feature_engineering(test_data)
```

상호 연관성이 존재하는 특성들을 통한 새로운 특성 생성

기대효과

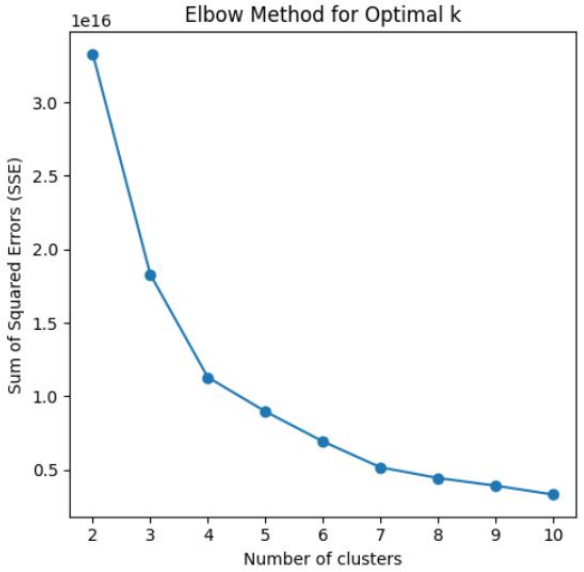
- 변수간의 복잡한 관계 학습
- 모델 분류 성능의 향상

04 전처리 및 특성공학

- Clustering

```
# 클러스터링에 사용할 특성 선택
features = ['X_Minimum', 'Y_Minim                                    minosity', 'Steel_Plate_Thickness']
# 클러스터링 모델 생성 및 학습
kmeans = KMeans(n_clusters=4)

kmeans.fit(train_data[features])
# train 데이터에 클러스터링 결과 추가
train_data['Cluster'] = kmeans.la
# test 데이터에 클러스터링 결과 추가
test_data['Cluster'] = kmeans.pre
```



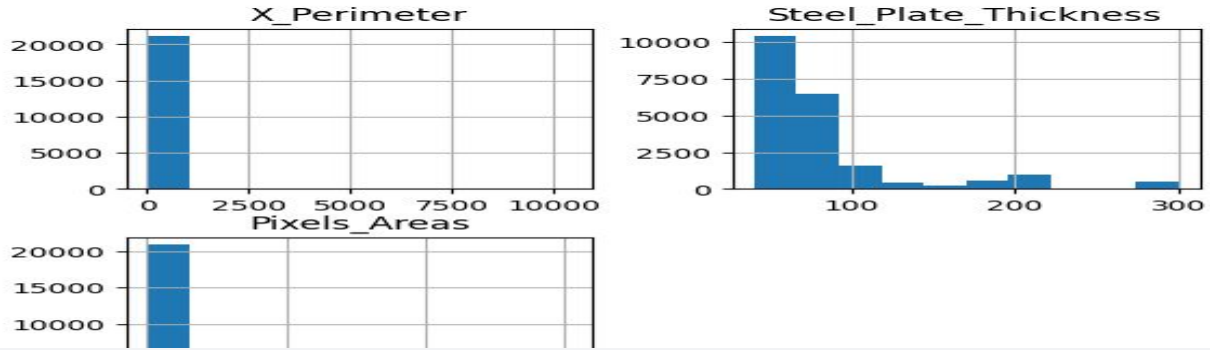
강판의 주요 특성들을 이용한 클러스터링 수행

기대효과

- 다양한 모델링 접근

04 전처리 및 특성공학

- Log Scaling



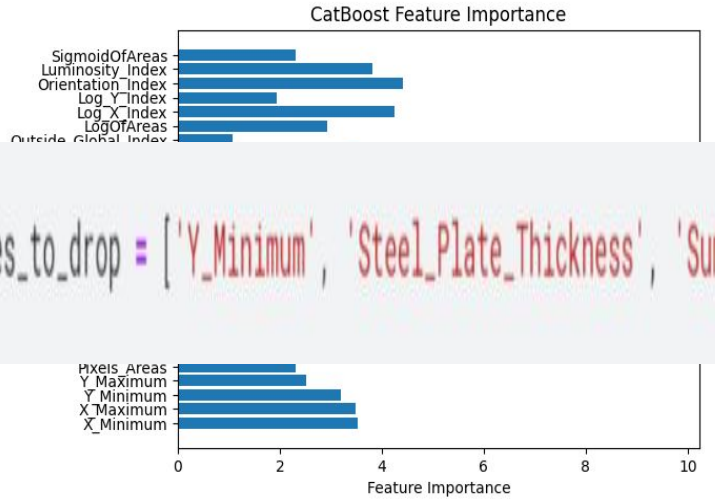

```
# log scaling
for col in ['X_Perimeter','Steel_Plate_Thickness','Pixels_Areas']:
    train_data[col] = np.log1p(train_data[col])
    test_data[col] = np.log1p(test_data[col])
```

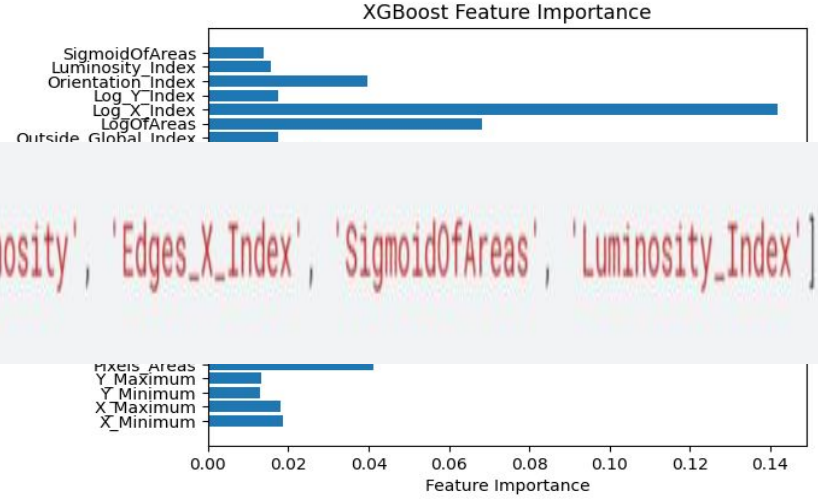
기대효과

- 데이터 분포 정규화 및 스케일 조정
- 이상치의 영향 감소

04 전처리 및 특성공학

- Drop Features






```
features_to_drop = ['Y_Minimum', 'Steel_Plate_Thickness', 'Sum_of_Luminosity', 'Edges_X_Index', 'SigmoidOfAreas', 'Luminosity_Index']
```

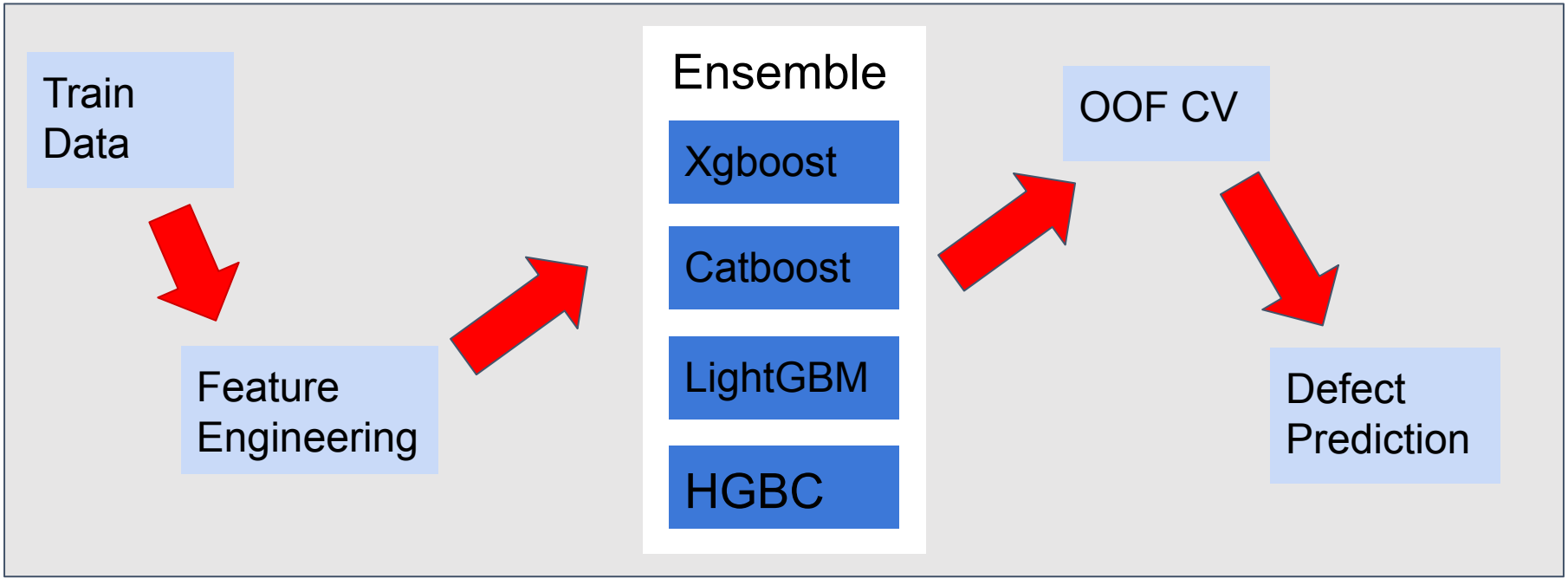
Feature importance를 고려한 feature 제거

기대효과

- 모델 복잡도 감소
- 다중공선성 감소

05 모델링

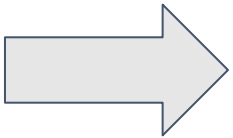
kaggle



05 모델링

- Models We Used

- Decision Tree
- Random Forest
- LightGBM
- XGBoost
- CatBoost
- HistGradientBoosting
- KNN



분류 성능 확인

- LightGBM
- XGBoost
- CatBoost
- HistGradientBoosting

05 모델링

kaggle

- Hyperparameter optimization


```
RETRAIN_LGBM_MODEL = True
def objective(trial):
    # Define parameters to be optimized for the LGBMClassifier
    param = {
    'objective': 'multiclass',  # Equivalent to multi:softmax but needs num_class as well
    'num_class': 8,  # Specify the number of classes if your task is multi-class classification
    'learning_rate': trial.suggest_float('learning_rate', 0.005, 0.1),
    'n_estimators': 3000,
    'lambda_l1': trial.suggest_float('lambda_l1', 1e-8, 10.0, log=True),
    'lambda_l2': trial.suggest_float('lambda_l2', 1e-8, 10.0, log=True),
    'max_depth': trial.suggest_int('max_depth', 3, 15),
    'colsample_bytree': trial.suggest_float('colsample_bytree', 0.3, 1.0),
    'subsample': trial.suggest_float('subsample', 0.5, 1.0),
    'min_child_weight': trial.suggest_int('min_child_weight', 1, 8),
    'device_type': 'cpu',    'num_leaves': trial.suggest_int('num_leaves', 4, 2048),
    "min_child_samples": trial.suggest_int("min_child_samples", 5, 100),"verbosity": -1,"early_stopping_rounds": 50,
    }
    auc_scores = []
    for train_idx, valid_idx in cv.split(X, y):
        X_train_fold, X_valid_fold = X.iloc[train_idx], X.iloc[valid_idx]
        y_train_fold, y_valid_fold = y.iloc[train_idx], y.iloc[valid_idx]
        model = LGBMClassifier(**param)
        model.fit(X_train_fold, y_train_fold, eval_set=[(X_valid_fold, y_valid_fold)], verbose=False)
        y_prob = model.predict_proba(X_valid_fold)
        average_auc = roc_auc_score(targets_bin.iloc[valid_idx], y_prob[:, 1:], multi_class="ovr", average="macro")
        auc_scores.append(average_auc)
    return np.mean(auc_scores)
```

05 모델링

- K-Fold Cross Validation

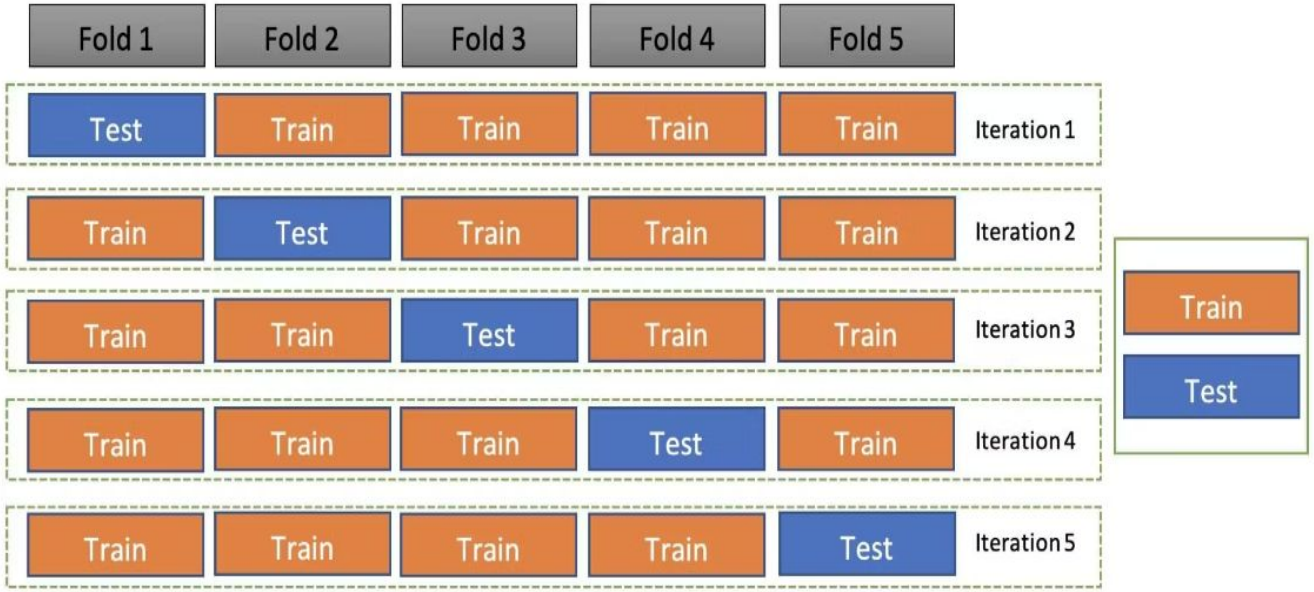


Image source: sqlrelease.com

05 모델링

- Stratified K-Fold Cross Validation

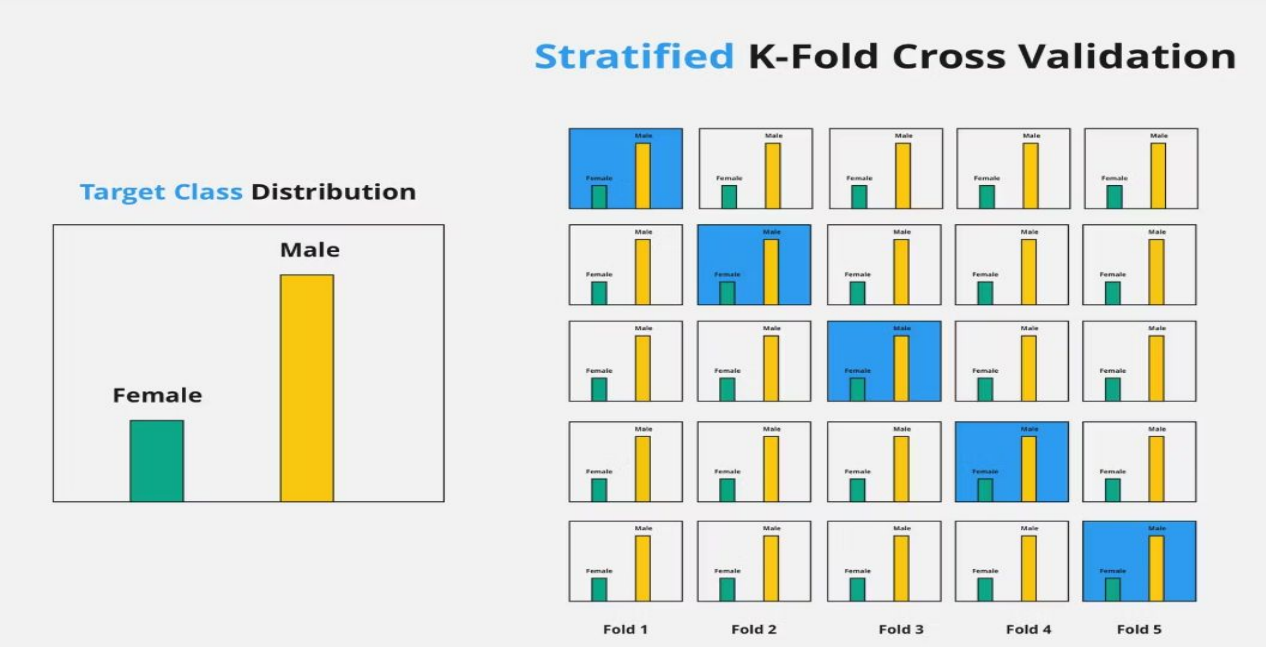


Image source: dataaspirant.com

05 모델링

- ROC-AUC Scores

Model	OOF ROC-AUC score	
DecisionTree	0.82987	
RandomForest	0.85229	
KNN	0.86540	
XGBoost	0.89889	✓
LGBM	0.89902	✓
CAT	0.89868	✓
HGBC	0.89589	✓

05 모델링

kaggle

05 모델링

- Different Model weights for each Labels

SciPy

05 모델링

- Different Model weights for each Labels

Class	XGB	LGBM	CAT	HGBC
Pastry	0.126	0.229	0.621	0.024
Z_Scratch	0.302	0.454	0.049	0.196
K_Scatch	0.091	0.761	0.102	0.045
Stains	0.070	0.384	0.452	0.093
Dirtiness	0.018	0.633	0.349	0.000
Bumps	0.135	0.385	0.454	0.026
Other_Faults	0.360	0.396	0.094	0.150

```
from functools import
from scipy.optimize i
blend = np.zeros((xgb
preds = np.zeros((xgb
initial_weights = np.

def calculate_roc_auc
    # Normalize weigh
    weights /= np.sum
    weighted_sum = oo
    # Calculate ROC A
    score = roc_auc_s
    return -score
bounds = [(0, None),

for k in range(len(TA
    result = minimize

    optimal_weights =
    # Update print st
    blend[:, k] = (xg
                   ca
    preds[:, k] = (xg
                   ca
```

```
+ oof_4 * weights[3]

of[:, k],
rgets_bin.iloc[:, k]),

_weights[1] +
_weights[3])
imal_weights[1] +
imal_weights[3])
```

06 결과

Private Score: 0.88905

KeyPoint 1 : 데이터 전처리, Feature Engineering

KeyPoint 2 : OOF를 통한 과적합 방지 및 성능 평가

KeyPoint 3 : Target Class 마다 적합한 모델 가중치 적용

Result : 일반화 된 강건한 모델

Model	OOF ROC-AUC Score
XGB	0.89889
LGBM	0.89902
CAT	0.89868
HGBC	0.89589
Blend of Models	0.90067

kaggle

Thank You for listening!