LG Aimers Hackathon
SMART FACTORY

팀원

정우섭
김유민
김유진
장동언
황정묵

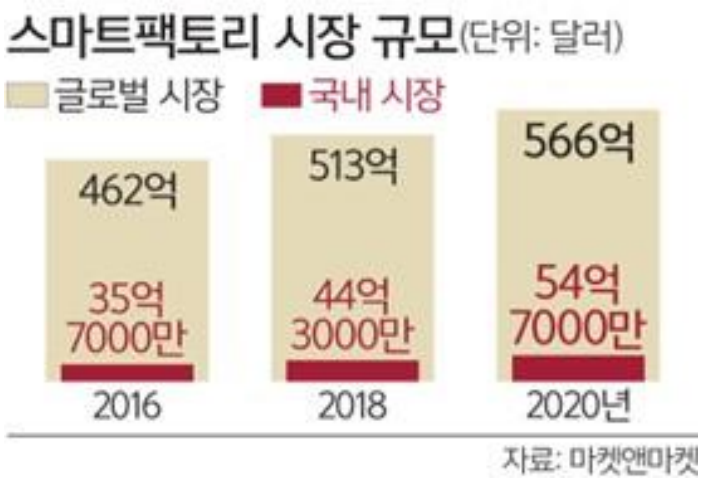
CONTENTS

01
Smart Factory

1.1 Importance of Smart Factory

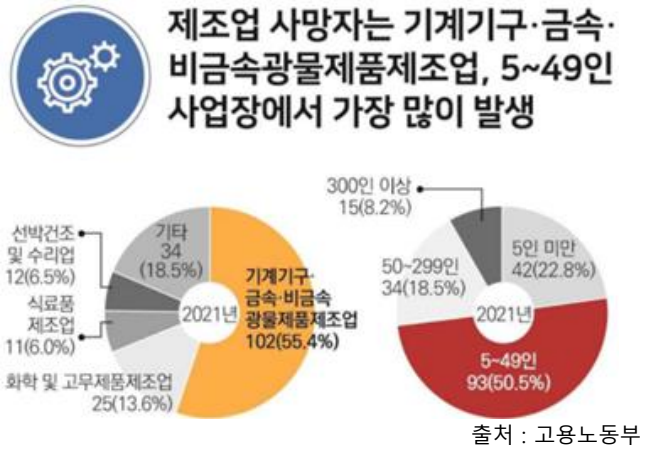
스마트 팩토리 시장의 성장에 따라 제조 생산현장의 **작업자의 안전**과
불량품 발생으로 인한 **품질 비용을 절감**할 수 있는 대안 필요

<스마트팩토리 시장의 성장>



IoT, 센서, 머신러닝 등 기술 발전과 함께
제조 지능화의 가속화를 통해 높은 생산성
및 품질관리 수준 확보.

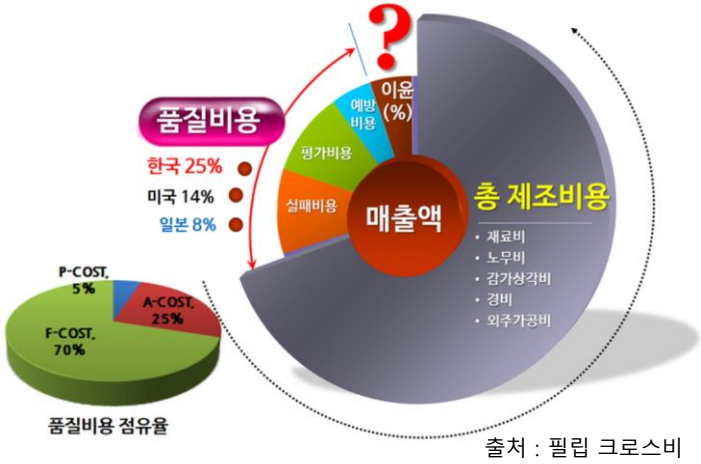
<제조 생산현장의 안전사고 현황>



출처 : 고용노동부

지속적으로 발생하는 생산현장의 안전사고.
최신 기술을 활용해 작업자의 안전을
확보하기 위한 노력 필요.

<총 제조비용 중 품질비용의 비율>



출처 : 필립 크로스비

제조업체의 일반적인 품질 유지비용은
총 매출액의 20~30% 점유.
불량을 만들지 않는 프로세스 구축 필요.

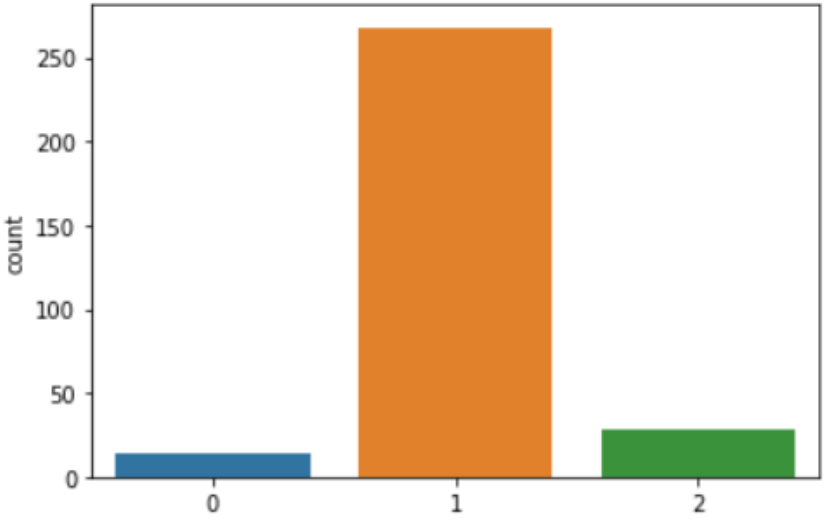
"품질 편차를 최소화해 생산성과 안정성을 극대화할 수 있는
스마트 공장의 제어시스템 구축 필요"

02
Data & Preprocessing

2.1 Data

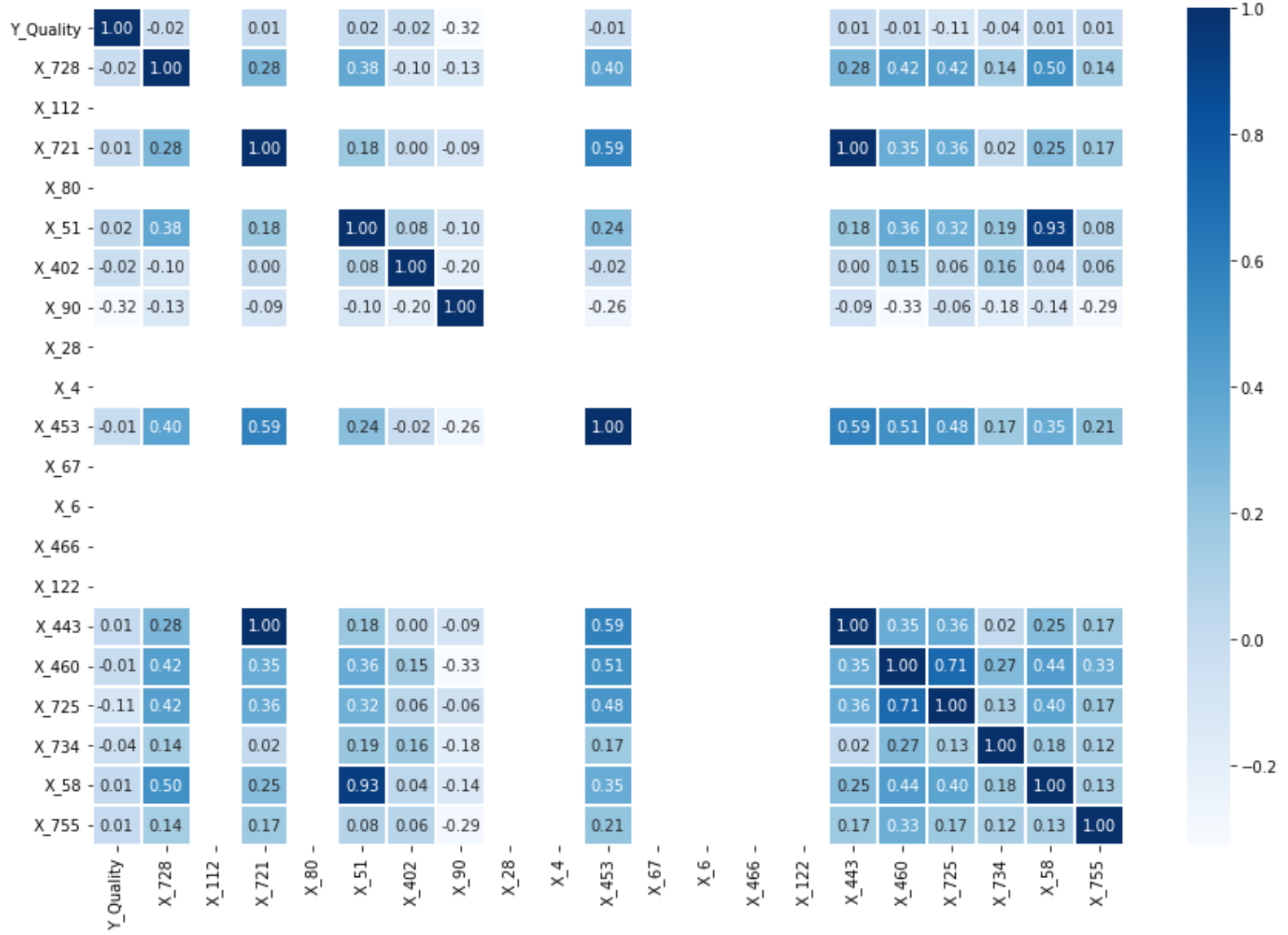
- Columns
  - ['LINE'] 제품의 공정 LINE 6종류 ('T050304', 'T050307', 'T100304', 'T100306', 'T010306', 'T010305')
  - ['PRODUCT_CODE'] : 제품의 CODE 3종류 ('A_31', 'T_31', 'O_31')
  - ['PRODUCT_ID'], ['TIMESTAMP']
  - ['Y_Quality'] : 제품 품질

- Target features
  - ['Y_Class']: 제품 품질의 상태
  - 0: 부적합
  - 1: 적합
  - 2: 부적합



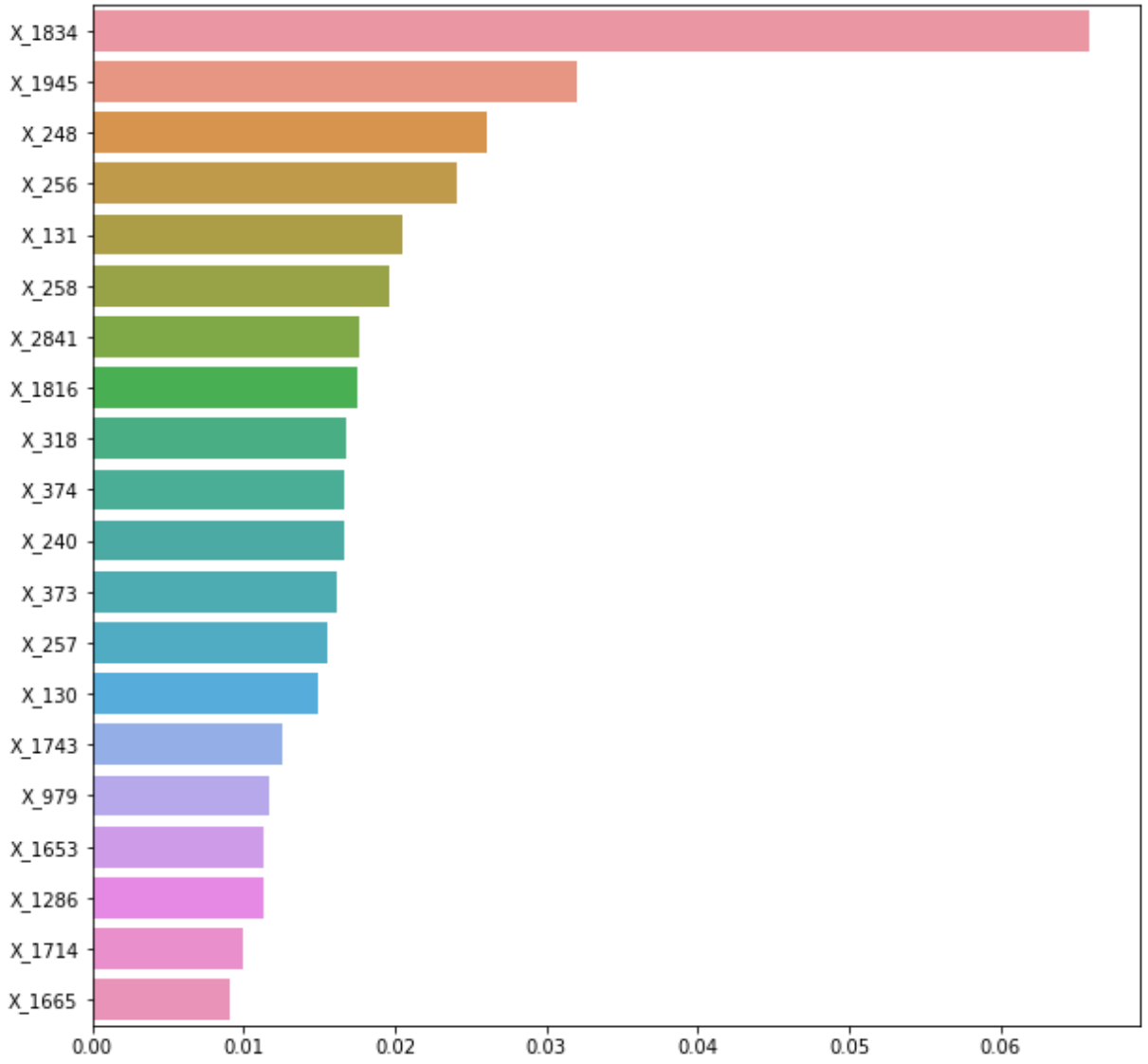
2.2 EDA

- Feature Correlation
  - 높은 상관관계 보인 변수
  - ['X_58' & 'X_51']
  - ['X_721' & 'X_443']



2.2 EDA

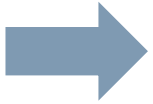
- Feature Importance
  - ['X_1834']: 영향력이 가장 큰 변수



2.3 Preprocessing

1. Drop Features
   - ['PRODUCT_ID', 'TIMESTAMP', 'Y_Quality'] 변수 제거
2. Label Encoding
   - ['LINE', 'PRODUCT_CODE'] 모델 적합을 위해 범주형에서 수치형 변수로 인코딩
3. Missing Values
   - ['LINE', 'PRODUCT_CODE’] <-- fillna(0)
   - 여러 데이터가 합쳐진 형태 (mean, median의 신뢰도가 떨어짐)

LINE	PRODUCT_CODE
T050304	A_31
T050307	A_31
T050304	A_31
T050307	A_31
T050304	A_31



LINE	PRODUCT_CODE
2	0
3	0
2	0
3	0
2	0

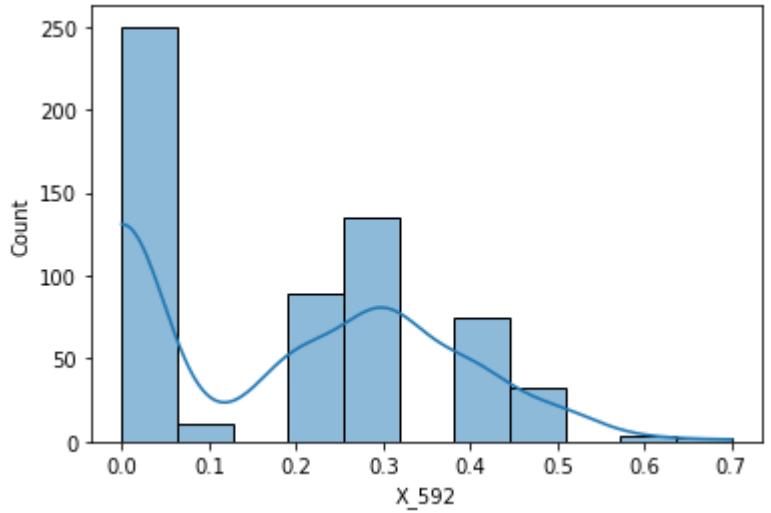
2.3 Preprocessing

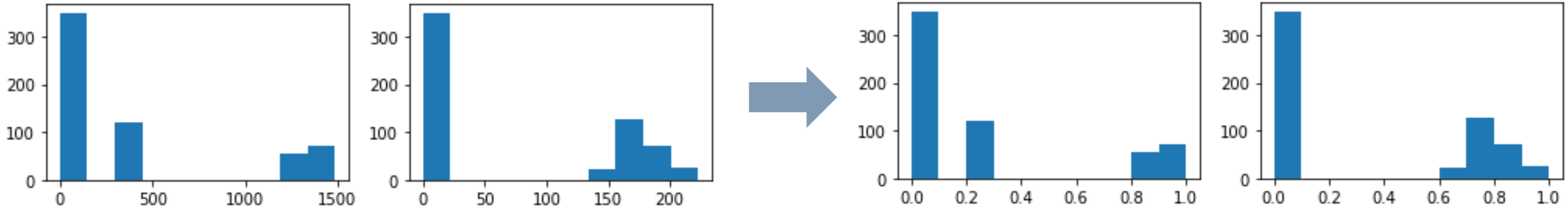
4. Feature Scaling

- 변수 값 범위를 일정한 범위 내로 조정

4.1 MinMax Normalization

- ['MinMaxScaler']
  - -0.4부터 40000까지 다양한 변수의 값 변위
  - 모든 변수가 최소 0, 최대 1의 값을 따르도록 정규화





03
Modeling

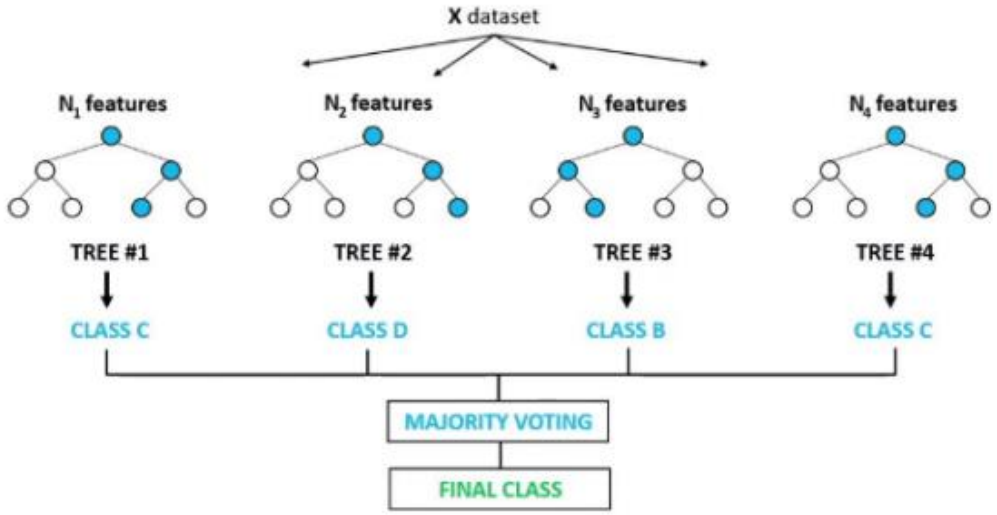
3.1 Machine Learning Models

1. Random Forest Classifier
2. Gradient Boosting Classifier
3. XGB Classifier
4. LGBM Classifier
5. CatBoost Classifier
6. Ridge Classifier
7. Bagging Classifier

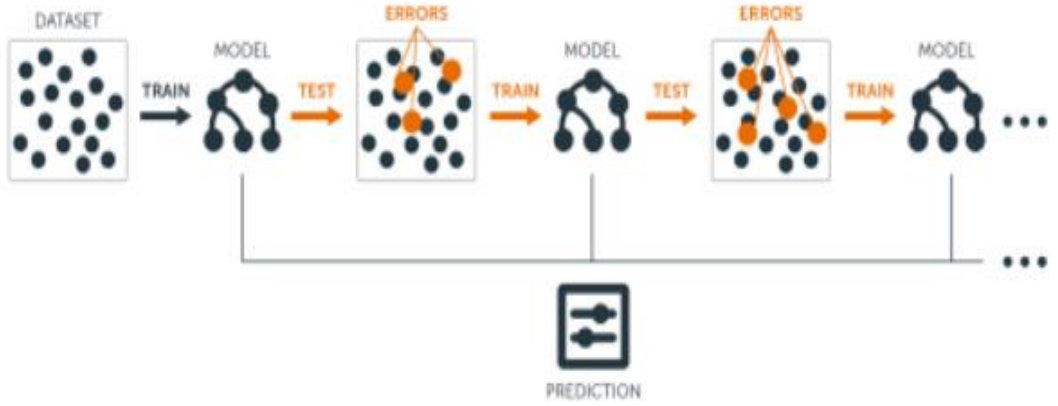
- 위 모델들을 사용하여 각각의 test set accuracy score 도출
- 그 결과를 ensemble 과정의 가중치 설정의 근거로 활용 (5장 참고)

3.2.1 Tree Models

- Random Forest Classifier
  - 의사결정나무(Decision Tree)를 여러 개 합쳐서 만든 앙상블 모델
  - 장점: Overfitting 예방 & 일반화 성능 향상

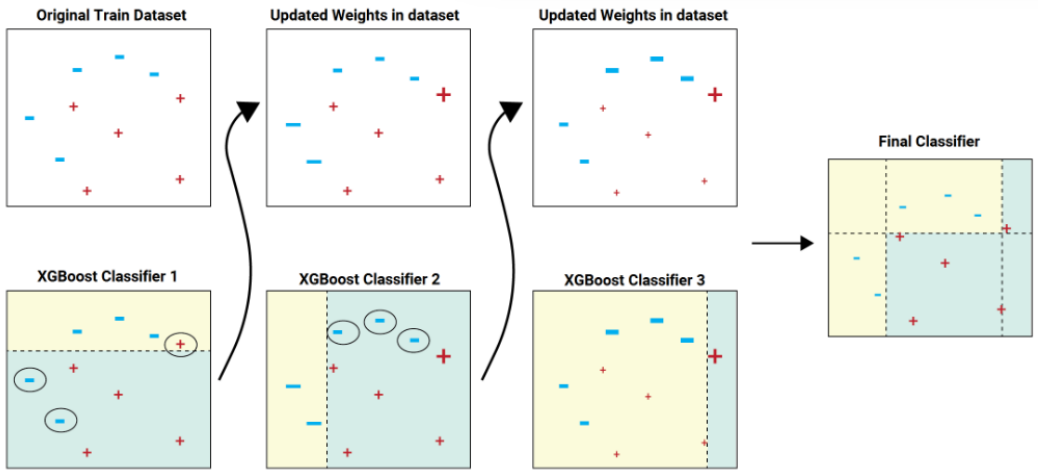


- Gradient Boosting Classifier
  - 여러 개의 의사결정나무(Decision Tree)를 연속적으로 만들어 예측 모델을 개선
  - 첫번째 Decision Tree 학습 후, 첫번째에 학습하지 않은 부분을 학습하여 예측

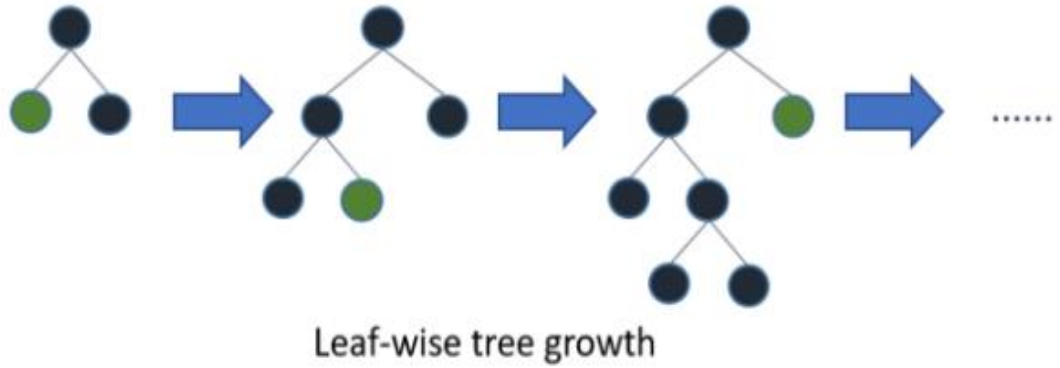


3.2.2 Gradient Boosting

- XGB Classifier
  - Gradient Boosted Tree 기반 앙상블 모델
  - 장점
    - 대용량 데이터셋에서 높은 예측 성능
    - 다양한 hyperparameter 제공해 모델 성능 조정 가능

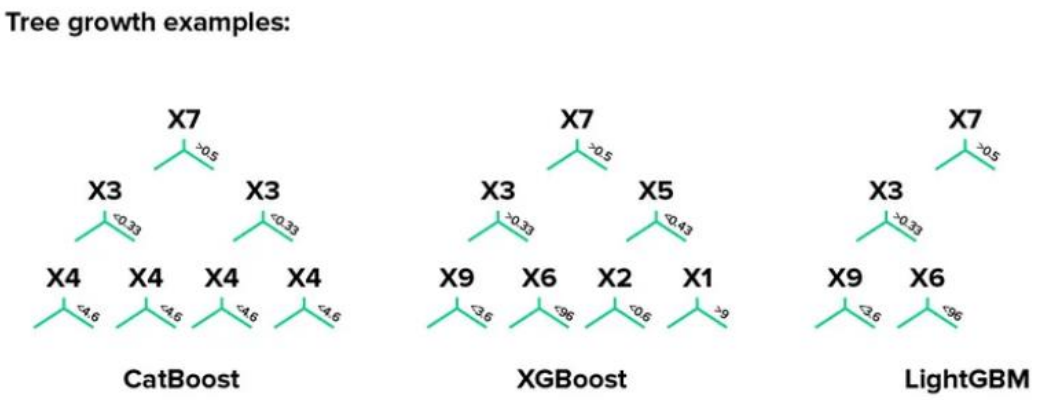


- LGBM Classifier
  - 경량화 된 Gradient Boosting 알고리즘
  - 장점:
    - 대용량 데이터셋을 빠르게 처리 가능
    - 범주형 변수의 처리, early stopping, 분산 학습 기능 제공

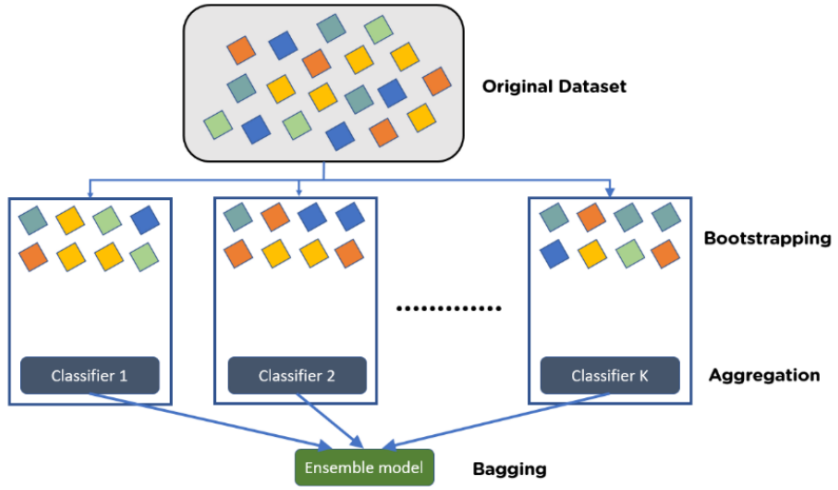


3.2.3 Boosting & Bagging

- Catboost Classifier
  - 범주형 변수 처리에 특화된 알고리즘
  - 장점
    - 빠른 속도, 높은 예측 성능
    - 범주형 변수가 많은 데이터셋에서 좋은 성능

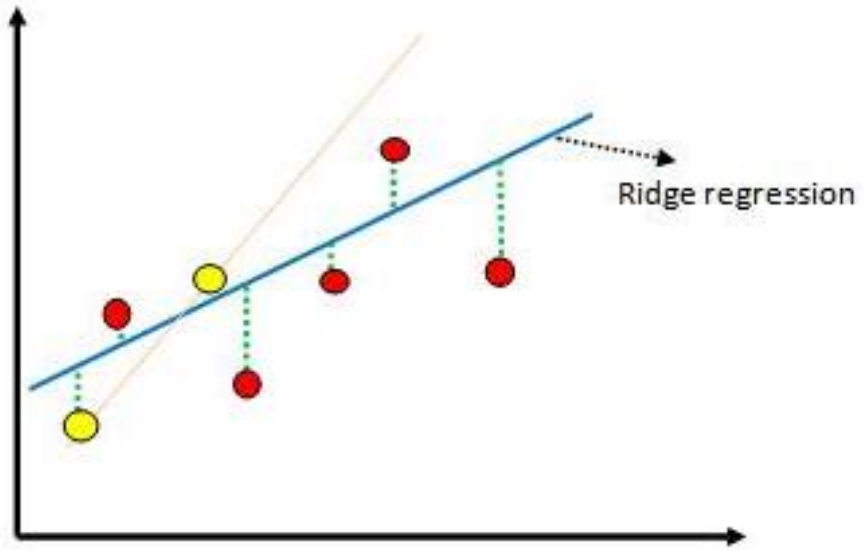


- Bagging Classifier
  - Random Forest와 유사한 앙상블 모델
  - 샘플링을 통해 여러 개의 분류 모델 만들어 해당 예측 결과를 결합해 최종 예측 도출
  - 장점:
    - 대용량 데이터셋에서 높은 예측 성능
    - 모델의 일반화 성능 향상



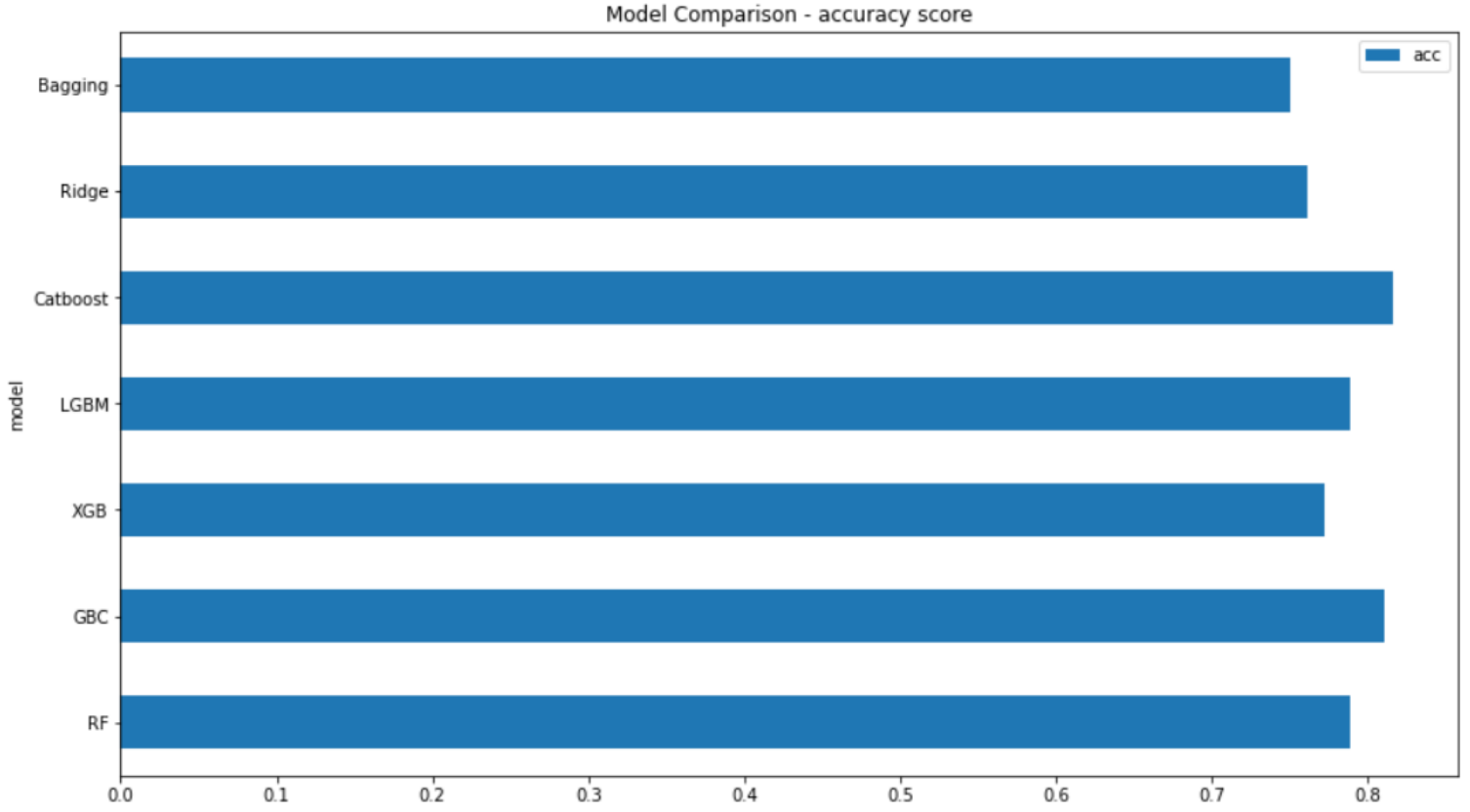
3.2.4 Ridge Classifier

- Ridge Classifier
  - Ridge 회귀를 분류 문제에 적용한 모델
  - L2 정규화 적용해 모델 일반화
  - 장점
    - 작은 데이터 셋에도 잘 작동



3.3 Used Models

- ML models' test set accuracy
  - 각 Machine Learning classification 모델 별 test set accuracy score 도출
  - Models' accuracy
    - RF: 0.78
    - GBC: 0.81
    - XGB: 0.77
    - LGBM: 0.78
    - CatBoost: 0.81
    - Ridge: 0.76
    - Bagging: 0.75



3.4 Sequential MLP

1. Multilayer Perceptron class 정의
   - 2개의 hidden layer 사용하여 MLP 구현
   - nn.Linear 클래스 사용해 선형 층 정의
   - Hidden layer에 ReLU activation function 적용
   - Output layer에 softmax activation function 적용 --> 결과를 확률 값 형태로 반환
2. Forward function
   - MLP의 정방향 계산으로, 층을 차례로 통과하며 결과 출력
   - apply_softmax = True: 출력층에서 softmax 함수 사용
3. Modeling
   - 256개의 데이터 (batch)를 입력으로 받음
   - 입력 벡터 크기 = 학습 데이터 크기 (train_x.shape[-1])
   - torch.rand 함수 사용해 입력 데이터 (x_in) 생성하여 모델 적용 후 결과 출력

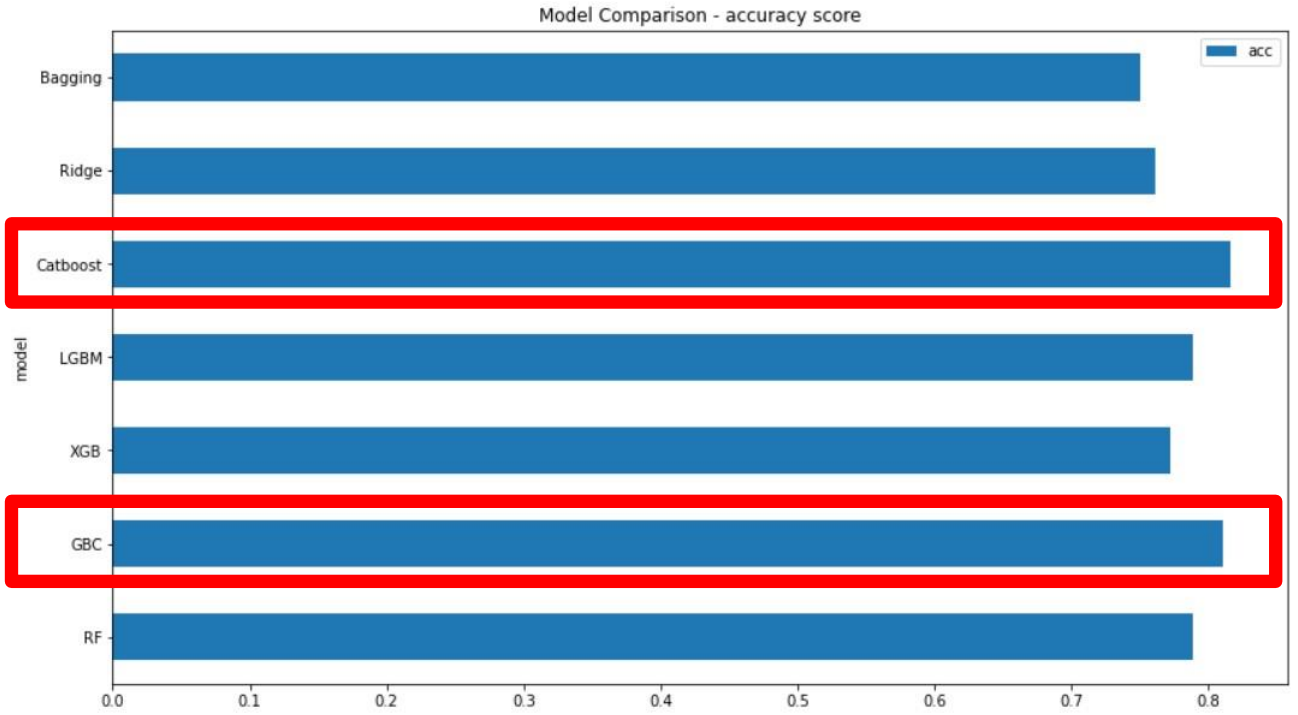
04
Parameters Optimization

4.1 Optimization

1. 2-Models Parameters
   - Test set accuracy 상위 2개 모델:
     Catboost Classifier,
     Gradient Boosting Classifier 선정

2. Optimization
   - hyperparameters optimization
     GridSearch & Optuna

3. Find best parameters



4.2.1 Gradient Boosting Classifier Optimization

Params

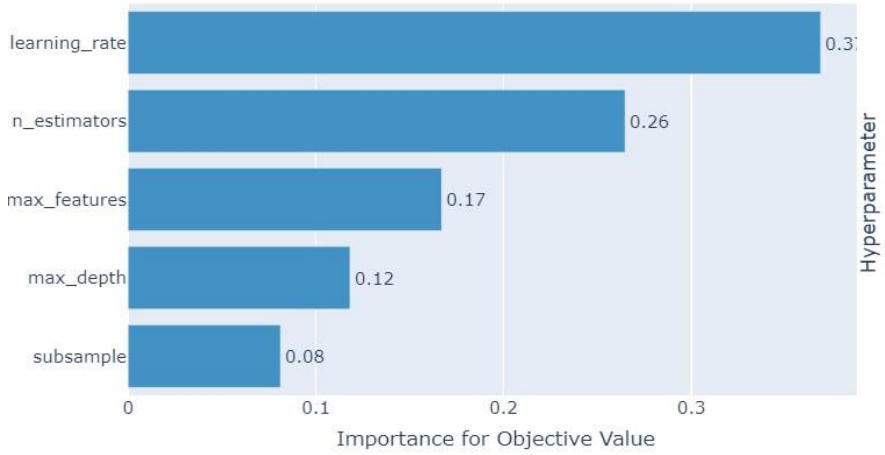
n_estimators	(100~5000, step=100)
learning_rate	(1e-4 ~ 0.3)
max_depth	(3 ~ 9)
subsample	(0.5 ~ 0.9, step=0.1)
max_features	(“auto”, “sqrt”, “log2”)
random_state	(42)

Best Params

n_estimators	: 1200
learning_rate	: 0.0091
max_depth	: 9
subsample	: 0.8
max_features	: ‘auto’

	precision	recall	f1-score	support
0	0.69	0.43	0.53	21
1	0.81	0.97	0.89	131
2	1	0.39	0.56	28
accuracy			0.82	180
macro avg	0.84	0.6	0.66	180
weighted avg	0.83	0.82	0.79	180

Optuna Accuracy Score : 81.7 %



< Hyperparameter 중요도 >

4.2.1 CatBoost Classifier Optimization

Params

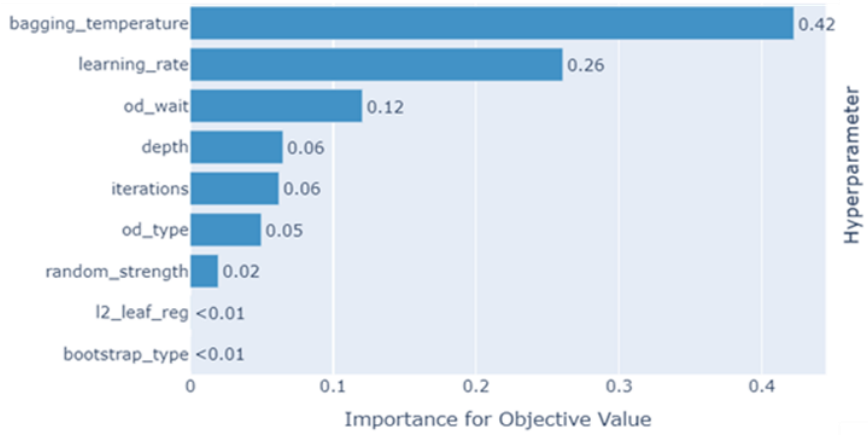
Iterations	(100~1000)
learning_rate	(1e-3 ~ 1e-1)
depth	(4 ~ 10)
12_leaf_reg	(1e-8 ~ 1e-1)
Booststrap_type	("Bayesian")
Random_strength	(1e-8 ~ 10.0)
Bagging_temp	(0.0 ~ 10.0)
Od_type	('IncToDec', 'Iter')
Od_wait	(10 ~ 50)

Best Params

Iterations	: 857
learning_rate	: 0.0709
depth	: 8
12_leaf_reg	: 0.00012
Booststrap_type	: 'Bayesian'
Random_strength	: 5.359
Bagging_temp	: 0.714
Od_type	: 'IncToDec'
Od_wait	: 'od_wait'

	precision	recall	f1-score	support
0	0.67	0.38	0.48	21
1	0.8	0.97	0.88	131
2	1	0.32	0.49	28
accuracy			0.8	180
macro avg	0.82	0.56	0.62	180
weighted avg	0.81	0.8	0.77	180

Optuna Accuracy Score : 80.0 %



< Hyperparameter 중요도 >

05
Ensemble Model

5.1 Ensemble techniques

- Ensemble techniques
  - 다수의 학습 모델을 조합하여 단일 모델의 성능 보다 더 발전된 성능을 얻는 기법
  - 1) Bagging: 다수결 투표 기반 학습
  - 2) Boosting: bootstrap 샘플 기반 분류 모델
  - 3) Stacking: 다수의 학습 모델 결합
- Voting Classifier: majority voting
  - M-Ensemble learning: 다수결 투표로 성능 향상
  - 3-Ensemble model의 우수한 성능:
    - 1) Multilayer Perceptron-based classification,
    - 2) 4-Ensemble model 보다 높은 accuracy

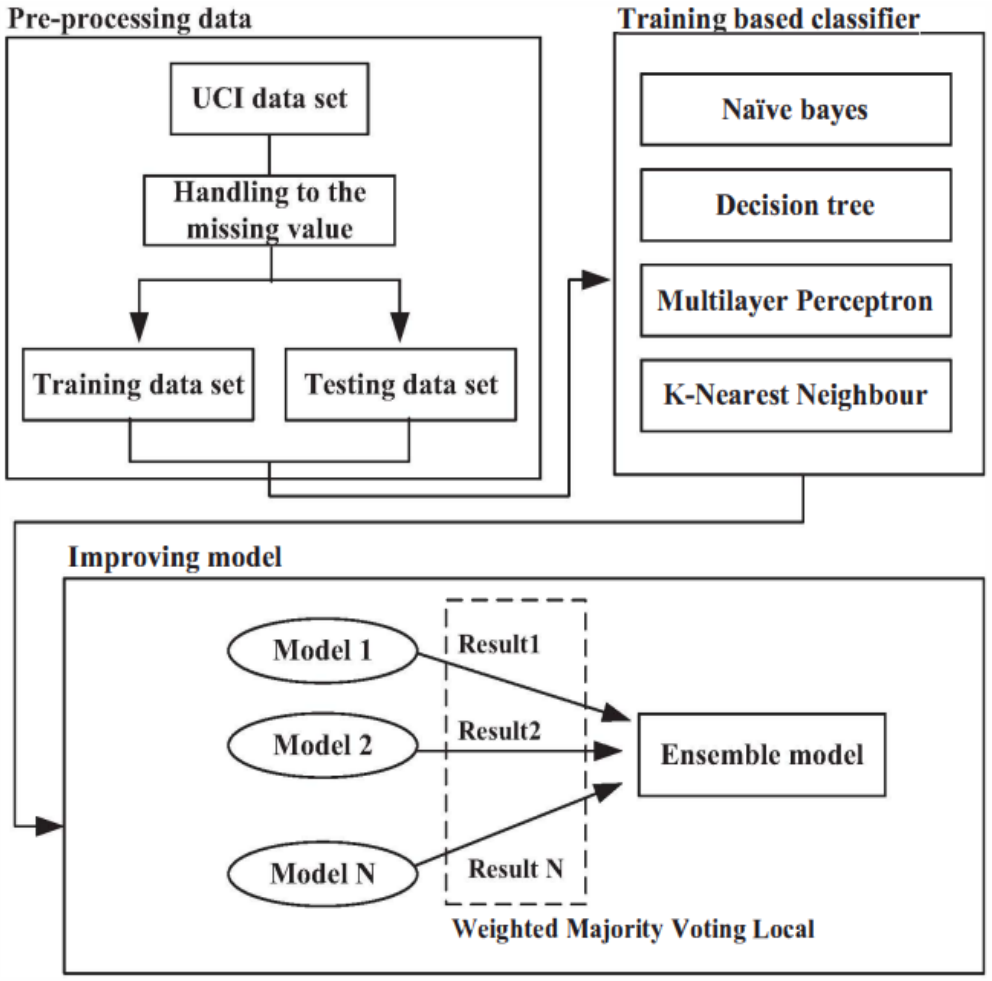


Figure 1. Framework of Majority Voting with the M-Ensemble Model.

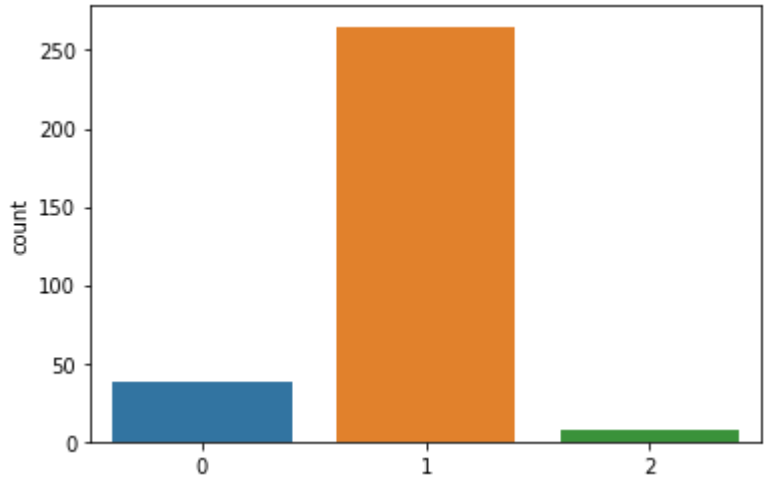
5.2 3-Ensemble Voting Classifier

- 3 Models Ensemble
  - Accuracy 상위 2개 + Overfitting 방지 목적 하위 1개 모델 앙상블
  - models = [CatBoostClassifier, GradientBoostingClassifier, XGBClassifier]
  - weight = [2, 2, 1]
  - voting = ‘hard’

weight	MODEL	ACCURACY	ENSEMBLE
2	CatBoost	0.816667	high training accuracy
2	GradientBoosting	0.811111	high training accuracy
X	RandomForest	0.788889	inferior to Gradient-boosted trees on compelx problem
X	LGBM	0.788889	lower performance than XGBoost
1	XGBoost	0.772222	prevent overfitting
X	Ridge	0.761111	low training accuracy
X	Bagging	0.75	low training accuracy

5.3 Final Submission Model

- 3-Ensemble Model's validation accuracy: 100%
  - scores = sklearn.metrics.accuracy_score(y_test, preds)
- Modeling Assumption:
  - train.csv test set 특성 및 분포가 미래 데이터인 test.csv 에도 보편적으로 적용될 것
  - train.csv 에서 좋은 성능을 보인 모델이 test.csv 에서도 좋은 성능을 보일 것
- Final Model:
  - test set validation accuracy 100% 성능을 보인 3-ensemble model 제출



References

References

- **Label Encoding:** https://paperswithcode.com/paper/label-encoding-for-regression-networks-1
- **MinMax:** Patro, S. G. O. P. A. L., and Kishore Kumar Sahu. "Normalization: A preprocessing stage." arXiv preprint arXiv:1503.06462 (2015).
- **Random Forest Classifier:** https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
- **Gradient Boosting Classifier:**
  - https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html
  - Lusa, Lara. "Gradient boosting for high-dimensional prediction of rare events." Computational Statistics & Data Analysis 113 (2017): 19-37.
- **XGBoost Classifier:**
  - https://xgboost.readthedocs.io/en/stable/
  - Chen, Tianqi, and Carlos Guestrin. "Xgboost: A scalable tree boosting system." Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining. 2016.
- **LGBM Classifier:** https://lightgbm.readthedocs.io/en/latest/pythonapi/lightgbm.LGBMClassifier.html
- **CatBoost Classifier:**
  - https://catboost.ai/en/docs/concepts/python-reference_catboostclassifier
  - Hancock, John T., and Taghi M. Khoshgoftaar. "CatBoost for big data: an interdisciplinary review." Journal of big data 7.1 (2020): 1-45.
- **Ridge Classifier:** https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.RidgeClassifier.html
- **Bagging Classifier:** https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingClassifier.html
- **Ensemble voting classifier:**
  - Rojarath, Artittayapron, Wararat Songpan, and Chakrit Pong-inwong. "Improved ensemble learning for classification techniques based on majority voting." 2016 7th IEEE international conference on software engineering and service science (ICSESS). IEEE, 2016.
  - Ruta, Dymitr, and Bogdan Gabrys. "Classifier selection for majority voting." Information fusion 6.1 (2005): 63-81.

THANK YOU