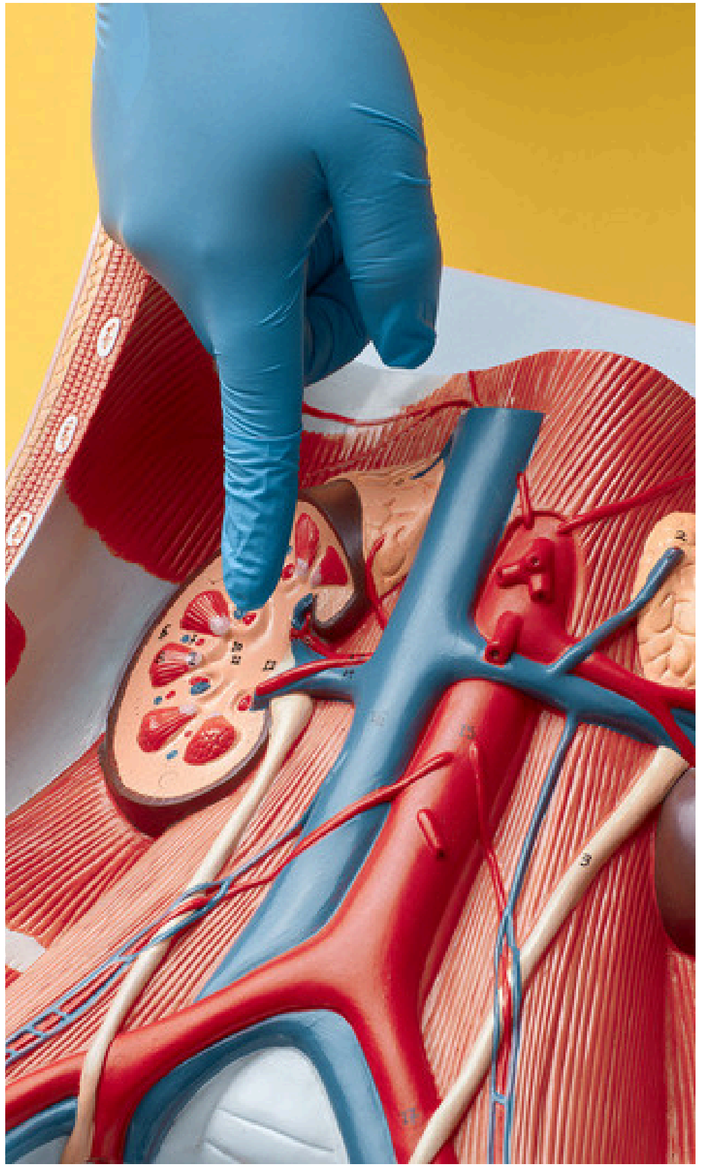

## 2025 여름 디지털 헬스케어 부트캠프 팀 프로젝트 발표

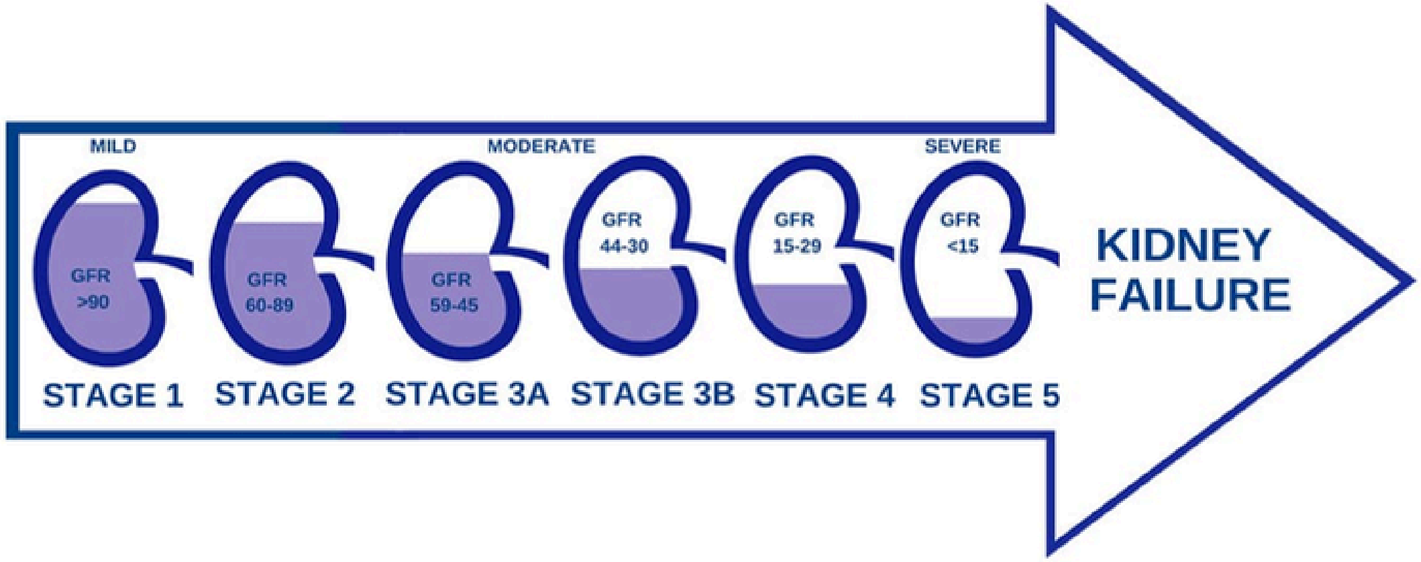
# 만성신장질환 예측

2팀 백혜연, 양호헌, 윤혜원, 이정규, 전준서, 최지인



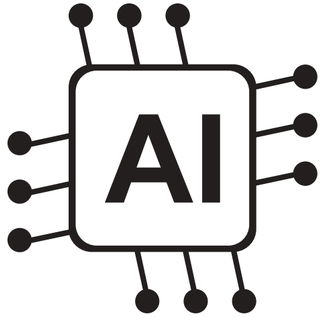
# 문제정의

- 만성 신장 질환은 증상이 늦게 나타나 조기 진단이 어려움
- 치료 지연 시 투석 또는 신장이식 등 고위험 치료 필요
- 혈액·소변 검사 등으로 조기 발견 가능하나, 전문가의 해석이 필요함
- AI 기반 예측 모델로 조기 선별 가능성 확인이 필요



# 프로젝트 필요성

01 조기 진단 시 치료비 절감,
합병증 예방 가능




02 AI 예측 모델 개발로
고위험 환자 스크리닝 가능

03 디지털 헬스케어와의
접목 가능성

# 프로젝트 목표

01 Kaggle 기반 공개 데이터셋을 활용한 CKD 예측 모델 개발

02 주요 생체징후 및 검사 수치를 바탕으로 CKD 여부를 분류

03 고위험군 조기 선별 및 예측 가능한 모델 확보

# 데이터셋 소개

-활용 데이터셋: Chronic Kidney Disease Prediction

-출처: Kaggle

-URL: https://www.kaggle.com/code/niteshyadav3103/chronic-kidney-disease-prediction-98-accuracy/input

-인도에서 2개월간 수집된 데이터

-적혈구, 부종, 당 수치 등 24개의 변수, 400명의 환자 데이터

# 주요변수

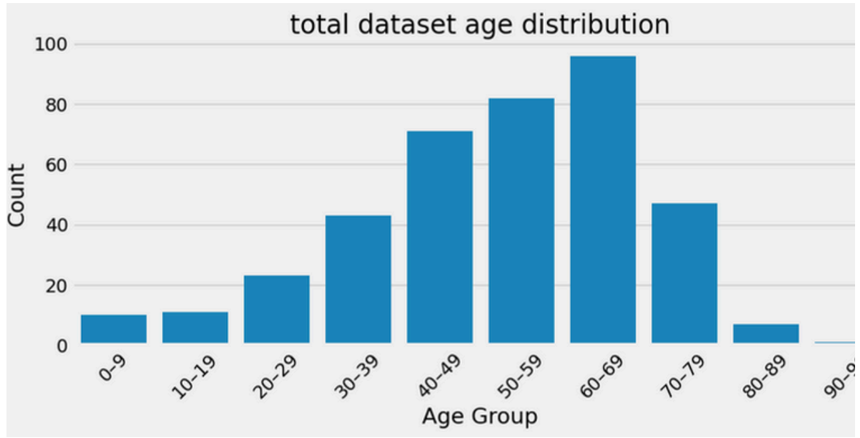
## 입력 변수 (Features)

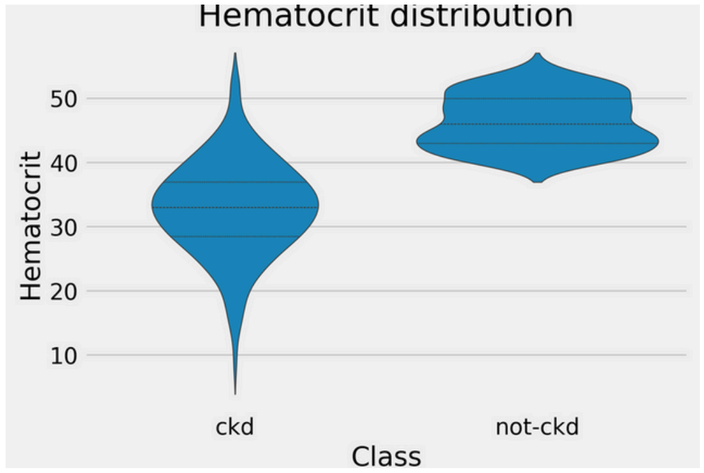
| | | | | |
|---|---|---|---|---|
| Age | Blood Pressure (bp) | Specific Gravity (sg) | Albumin | Sugar |
| Red Blood Cells (rbc) | Pus Cell (pc) | Pus Cell Clumps (pcc) | Bacteria (ba) | Blood Glucose Random (bgr) |
| Blood Urea | Serum Creatinine | Sodium | Potassium | Hemoglobin |
| Packed Cell Volume | WBC | RBC | Hypertension | Diabetes Mellitus |
| Coronary Artery Disease | Appetite | Pedal Edema | Anemia | |

## 타겟 변수 (Features)

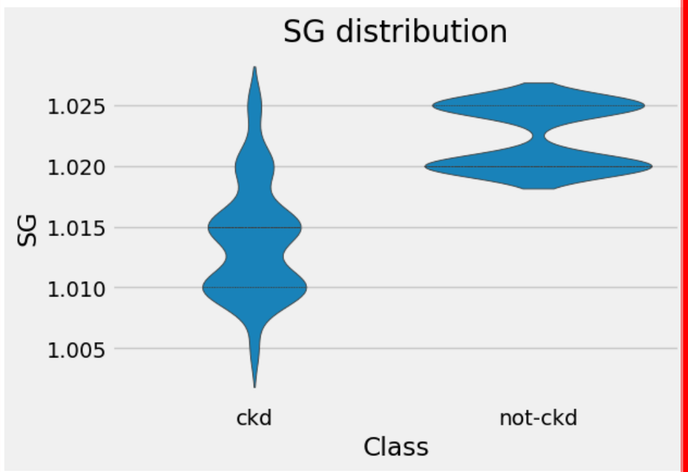
classification (ckd or not ckd)

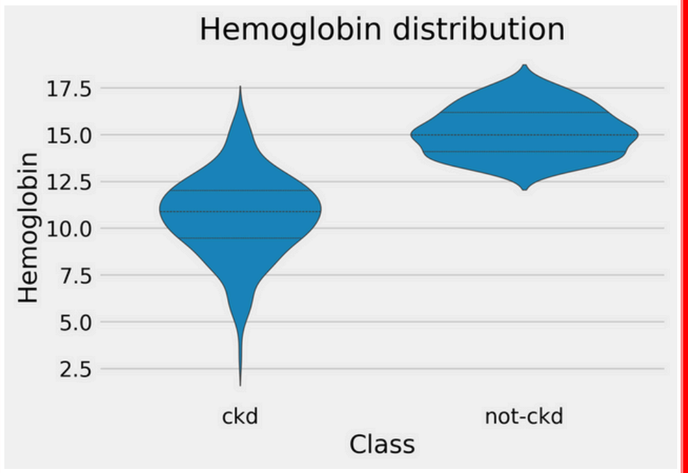
# EDA



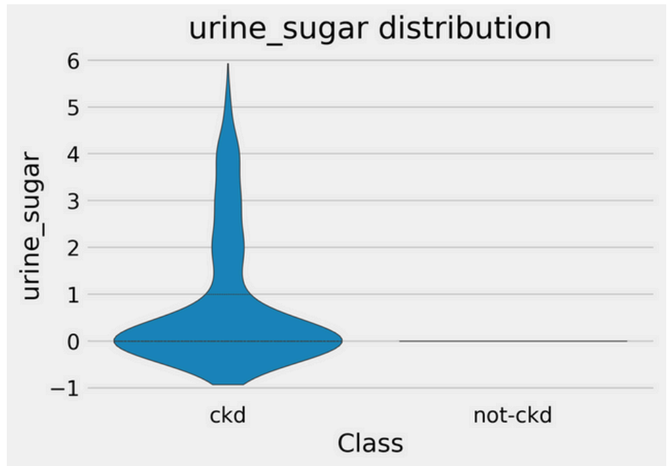



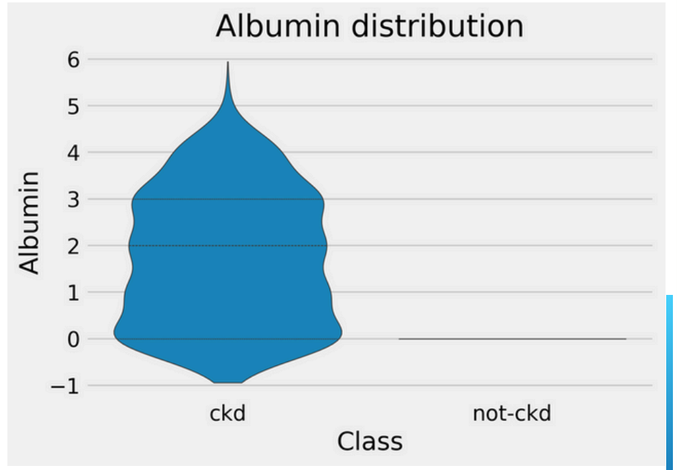

*ckd, not-ckd 그룹의 분포가 확연히 다른 feature*

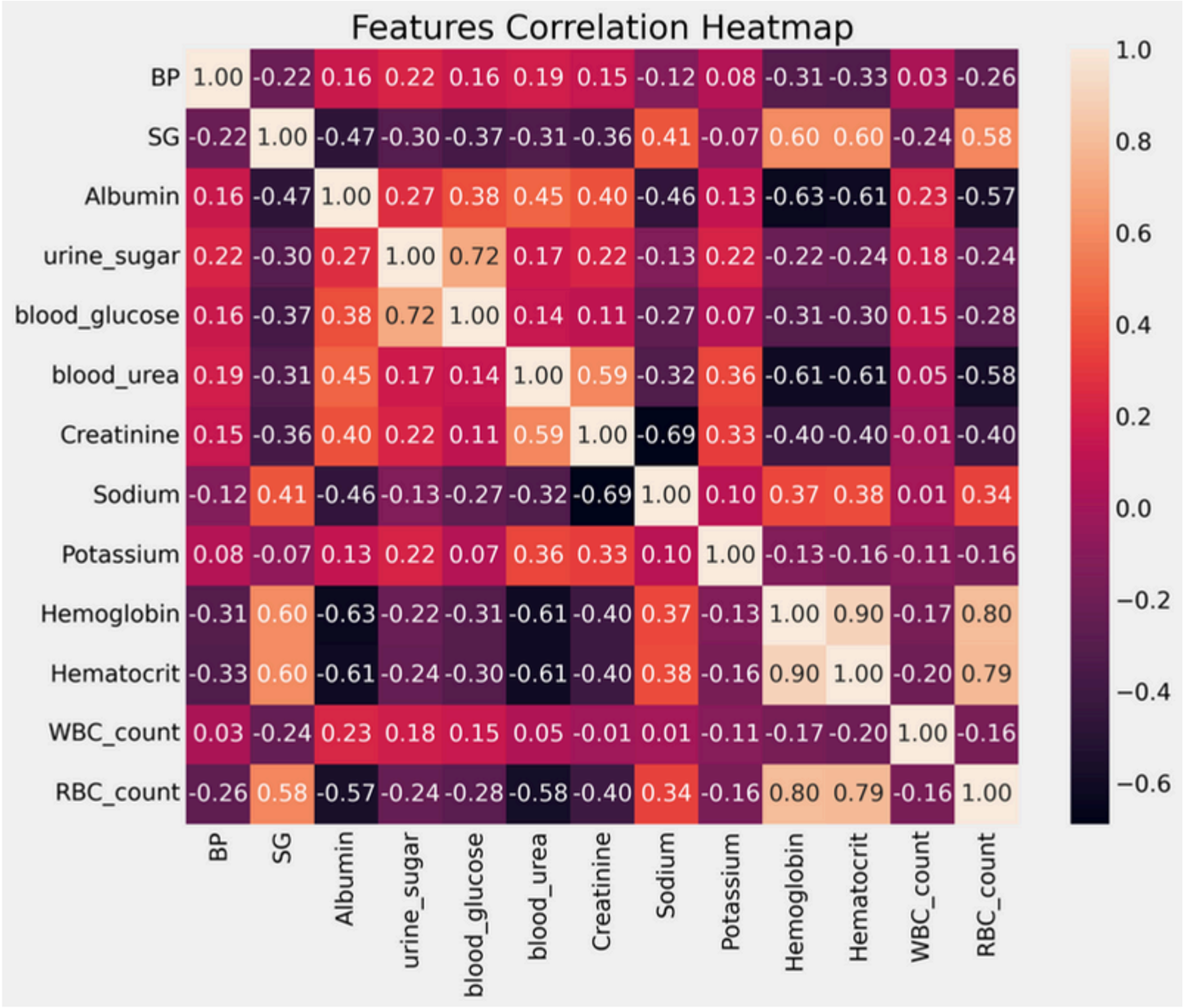



*not-ckd 그룹에서는 분산이 0인 feature*




Features Correlation Heatmap
BP 1.00 -0.22 0.16 0.22 0.16 0.19 0.15 -0.12 0.08 -0.31 -0.33 0.03 -0.26
SG -0.22 1.00 -0.47 -0.30 -0.37 -0.31 -0.36 0.41 -0.07 0.60 0.60 -0.24 0.58
Albumin 0.16 -0.47 1.00 0.27 0.38 0.45 0.40 -0.46 0.13 -0.63 -0.61 0.23 -0.57
urine_sugar 0.22 -0.30 0.27 1.00 0.72 0.17 0.22 -0.13 0.22 -0.22 -0.24 0.18 -0.24
blood_glucose 0.16 -0.37 0.38 0.72 1.00 0.14 0.11 -0.27 0.07 -0.31 -0.30 0.15 -0.28
blood_urea 0.19 -0.31 0.45 0.17 0.14 1.00 0.59 -0.32 0.36 -0.61 -0.61 0.05 -0.58
Creatinine 0.15 -0.36 0.40 0.22 0.11 0.59 1.00 -0.69 0.33 -0.40 -0.40 -0.01 -0.40
Sodium -0.12 0.41 -0.46 -0.13 -0.27 -0.32 -0.69 1.00 0.10 0.37 0.38 0.01 0.34
Potassium 0.08 -0.07 0.13 0.22 0.07 0.36 0.33 0.10 1.00 -0.13 -0.16 -0.11 -0.16
Hemoglobin -0.31 0.60 -0.63 -0.22 -0.31 -0.61 -0.40 0.37 -0.13 1.00 0.90 -0.17 0.80
Hematocrit -0.33 0.60 -0.61 -0.24 -0.30 -0.61 -0.40 0.38 -0.16 0.90 1.00 -0.20 0.79
WBC_count 0.03 -0.24 0.23 0.18 0.15 0.05 -0.01 0.01 -0.11 -0.17 -0.20 1.00 -0.16
RBC_count -0.26 0.58 -0.57 -0.24 -0.28 -0.58 -0.40 0.34 -0.16 0.80 0.79 -0.16 1.00
BP
SG
Albumin
urine_sugar
blood_glucose
blood_urea
Creatinine
Sodium
Potassium
Hemoglobin
Hematocrit
WBC_count
RBC_count
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6

# 데이터 전처리 과정

## 01 범주형 변수 결측치 처리

- 각 feature의 최빈 비결측값으로 결측값을 대체한다

```
def impute_mode(feature):
    mode = df[feature].mode()[0]
    df[feature] = df[feature].fillna(mode)
```

```
for col in cat_cols:
    impute_mode(col)
```

## 02 수치형 변수 결측치 처리

- 각 feature의 비결측값 중 무작위로 중복없이 선별하여, 결측값을 대체한다
- 평균/중앙값을 활용하면 분산이 줄어들 수 있으므로, 비결측값 내 무작위 선별을 통해 분산에 큰 영향이 가지 않도록 한다

```
def random_value_imputation(feature):
    random_sample = df[feature].dropna().sample(df[feature].isna().sum())
    random_sample.index = df[df[feature].isnull()].index
    df.loc[df[feature].isnull(), feature] = random_sample
```

```
for col in num_cols:
    random_value_imputation(col)
```

```
random_value_imputation('urine_RBC')
random_value_imputation('urine_pc')
```

## 03 범주형 변수 0, 1로 인코딩

- 모든 범주형 변수가 2가지 category를 지닌다
- 따라서 LabelEncoder를 사용하여 0, 1 로 변환한다

```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

for col in cat_cols:
    df[col] = le.fit_transform(df[col])
```

## 04 Scaling

```
preprocessor = ColumnTransformer([
    ('num', StandardScaler(), numeric_features),
    ('bin', 'passthrough', binary_features),
])
```

# 모델링

**목표** 여러 머신러닝 모델을 비교하여 최적의 모델을 탐색한다

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score
# AutoGluon
from autogluon.tabular import TabularPredictor
```

```
# 5. AutoGluon을 활용한 모델 학습
train_data = X_train.copy()
train_data['class'] = y_train
predictor = TabularPredictor(
    label='class',
    eval_metric='f1'
).fit(
    train_data,
    time_limit=300,
    presets='best_quality'
)
```

## AutoGluon의 특징

- **자동 모델 탐색 *(AutoML)***
  - 다양한 알고리즘(KNN, RF, XGBoost, CatBoost, NeuralNet 등)을 시도하여
  - 각 모델 성능을 비교하고 최적 모델을 선택한다

- ***Stacking* 기반 앙상블 지원**
  - 모델들을 단순 비교만 하지 않고, 앙상블하여 성능 향상

**결과** 여러 머신러닝 모델을 비교하여 최적의 모델을 찾는다

```
# 7. 학습 결과 요약 (리더보드, 최적 모델 및 하이퍼파라미터)
leaderboard = predictor.leaderboard(df_test, silent=True)
print('=== Leaderboard ===')
print(leaderboard)

# 최적 모델 이름 (deprecation 해결: model_best 속성 사용)
best_model = predictor.model_best
print(f'Best model: {best_model}')
```

- Validation set에서의 f1-score를 기준으로 best model을 선정함 (결과: WeightedEnsemble_L2 )
- Precision과 **recall** 모두 높아야 하기 때문에 f1-score를 사용함

**=== Leaderboard ===**

| | model | score_test | score_val | eval_metric |
|---|---|---|---|---|
| 0 | CatBoost_r137_BAG_L1 | 0.983051 | 0.96748 | f1 |
| 1 | CatBoost_BAG_L1 | 0.983051 | 0.97541 | f1 |
| 2 | CatBoost_r177_BAG_L1 | 0.983051 | 0.963563 | f1 |
| 3 | RandomForestEntr_BAG_L1 | 0.983051 | 0.974576 | f1 |
| 4 | CatBoost_r13_BAG_L1 | 0.983051 | 0.971429 | f1 |
| 5 | XGBoost_r89_BAG_L1 | 0.983051 | 0.983193 | f1 |
| 6 | ExtraTreesEntr_BAG_L1 | 0.983051 | 0.987448 | f1 |
| 7 | ExtraTrees_r42_BAG_L1 | 0.983051 | 0.987448 | f1 |
| 8 | ExtraTreesGini_BAG_L1 | 0.983051 | 0.991597 | f1 |

# 모델 평가

## 01 평가 지표

- AutoGluon을 통해 얻은 대부분의 머신러닝 모델이 높은 정확도와 정밀도, 재현율을 보여주었다
- 예시로 몇 개 모델의 accuracy, precision, recall, f1-score, roc-auc를 첨부하였다

| Metrics for Model: CatBoost_r13_BAG_L1 | |
|---|---|
| **Accuracy** | 0.9875 |
| **Precision** | 1.0000 |
| **Recall** | 0.9667 |
| **F1 Score** | 0.9831 |
| **ROC-AUC** | 1.0000 |
| **Metrics for Model: RandomForest_r195_BAG_L1** | |
| **Accuracy** | 0.9875 |
| **Precision** | 1.0000 |
| **Recall** | 0.9667 |
| **F1 Score** | 0.9831 |
| **ROC-AUC** | 1.0000 |

| Metrics for Model: XGBoost_r89_BAG_L1 | |
|---|---|
| **Accuracy** | 0.9875 |
| **Precision** | 1.0000 |
| **Recall** | 0.9667 |
| **F1 Score** | 0.9831 |
| **ROC-AUC** | 1.0000 |
| **Metrics for Model: WeightedEnsemble_L2** | |
| **Accuracy** | 0.9625 |
| **Precision** | 0.9335 |
| **Recall** | 0.9667 |
| **F1 Score** | 0.9508 |
| **ROC-AUC** | 0.9987 |

# 모델 평가

## 01 모델 선정

- 의료분야: 단순 성능지표뿐 아니라 **설명 가능한** 모델을 찾음
  - leaderboard 모델 중 **Xgboost classifier** 선정(Feature importance 등)
  - **Logistic regression**
  - **Decision tree**

## 02 모델 평가

- **Logistic Regression**

**Logistic Regression** CV **ROC-AUC**: 0.998 ± 0.002

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.91 | 0.97 | 0.94 | 30 |
| 1 | 0.98 | 0.94 | 0.96 | 50 |
| | | | | |
| **accuracy** | | | 0.95 | 80 |
| **macro avg** | 0.94 | 0.95 | 0.95 | 80 |
| **weighted avg** | 0.95 | 0.95 | 0.95 | 80 |

| | accuracy | f1- score | roc-auc |
|---|---|---|---|
| **Logistic Regression** | 0.95 | 0.9592 | 0.9967 |
| **Decision Tree** | 0.95 | 0.9592 | 0.9533 |
| **XGBoost** | 0.9625 | 0.9697 | 0.9967 |

| Feature | Coefficients | Odds Ratios |
|---|---|---|
| SG | -1.926873 | 0.145603 |
| appetite | 1.754349 | 5.779686 |
| DM | 1.609054 | 4.998083 |
| HTN | 1.267148 | 3.55071 |
| Hematocrit | -1.237264 | 0.290177 |
| Hemoglobin | -1.21877 | 0.295594 |
| Albumin | 1.00697 | 2.737295 |
| BP | 0.9206 | 2.510797 |
| pedal_edema | 0.830566 | 2.294618 |
| Creatinine | 0.659127 | 1.933104 |
| anemia | 0.654691 | 1.924548 |
| blood_glucose | 0.52042 | 1.682734 |

| Feature | Coefficients | Odds Ratios |
|---|---|---|
| Sodium | -0.367152 | 0.692705 |
| urine_sugar | 0.363455 | 1.43829 |
| RBC_count | -0.355607 | 0.700748 |
| WBC_count | 0.321659 | 1.379414 |
| urine_pcc | 0.262579 | 1.300279 |
| blood_urea | -0.195366 | 0.822534 |
| Potassium | -0.186803 | 0.829607 |
| Age | -0.166124 | 0.846941 |
| urine_pc | -0.123676 | 0.883666 |
| urine_bacteria | 0.077683 | 1.08078 |
| CAD | 0.039044 | 1.039816 |
| urine_RBC | 1.034229 | 2.812936 |

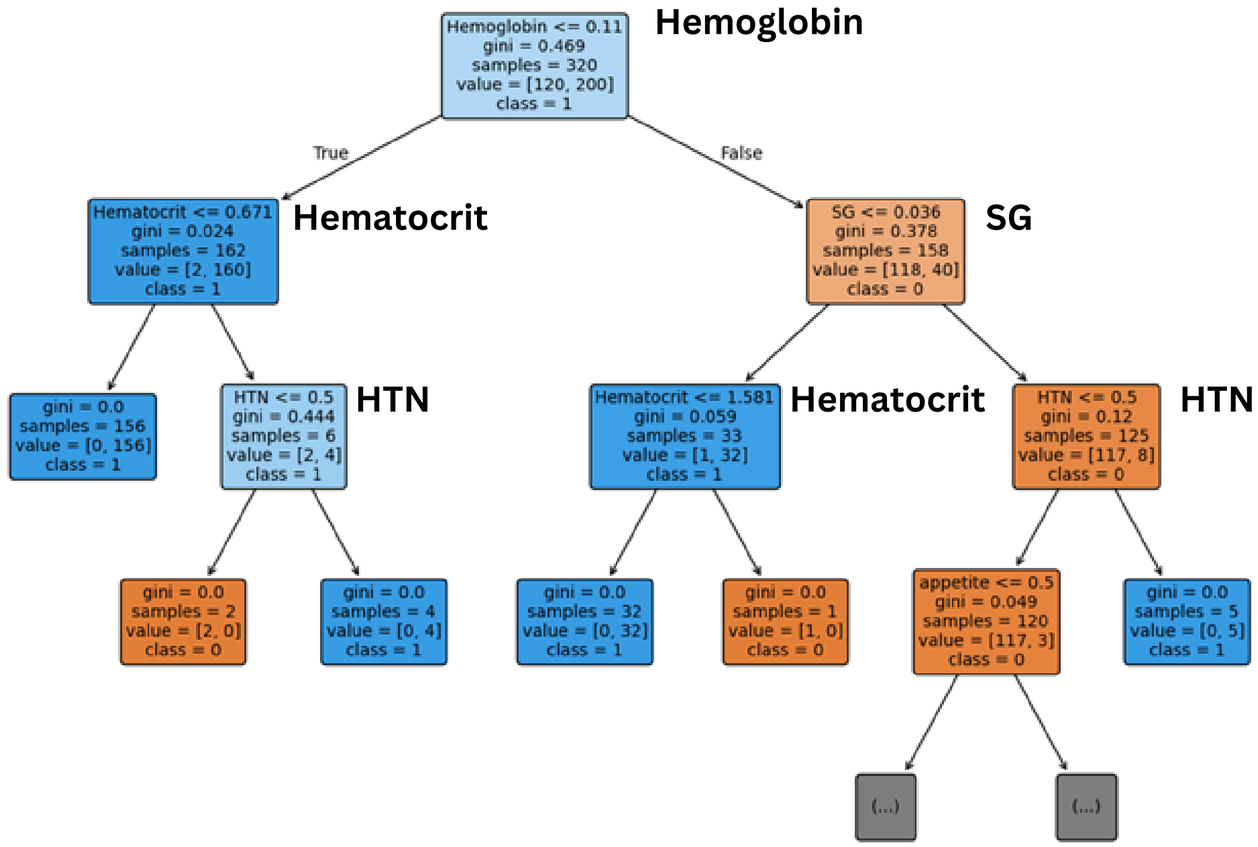
# 모델 평가

## 03 모델 평가

- **Decision Tree (depth=4)**

**DecisionTree** CV **ROC-AUC**: 0.952 ± 0.008

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.91 | 0.97 | 0.94 | 30 |
| 1 | 0.98 | 0.94 | 0.96 | 30 |
| | | | | |
| **accuracy** | | | 0.96 | 80 |
| **macro avg** | 0.94 | 0.95 | 0.96 | 80 |
| **weighted avg** | 0.95 | 0.95 | 0.96 | 80 |



# 모델 평가

## 03 모델 평가

- **XGBoost Classifier**

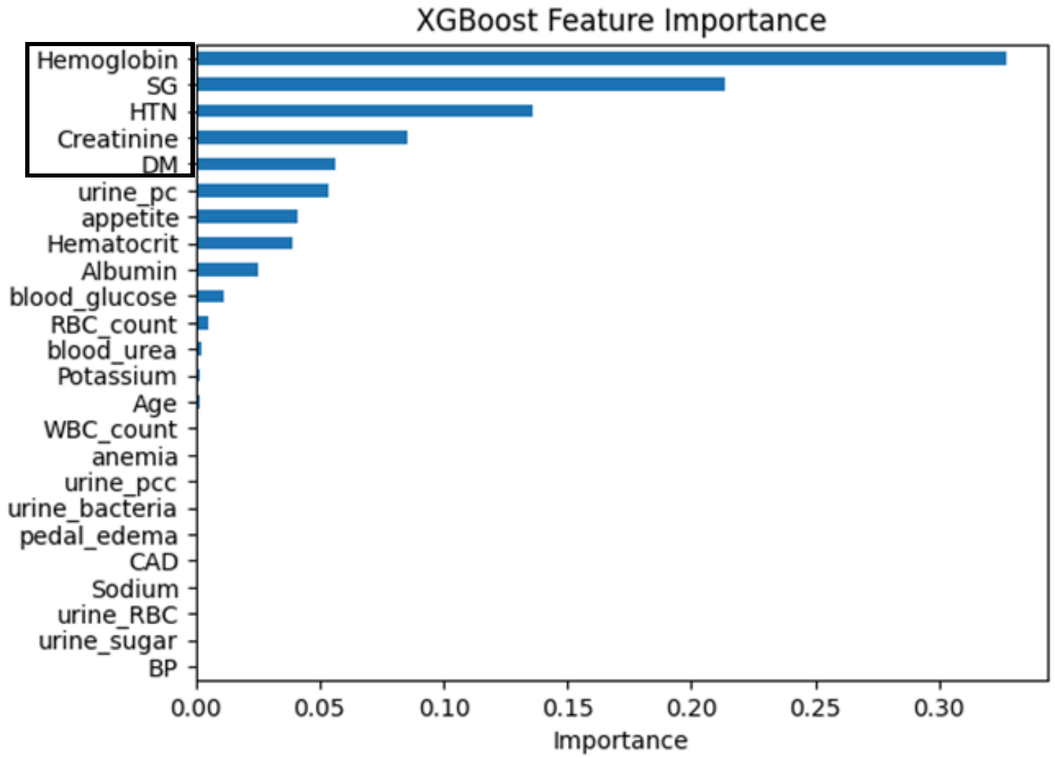
- **XGBoost** CV **ROC-AUC**: 0.998 ± 0.002

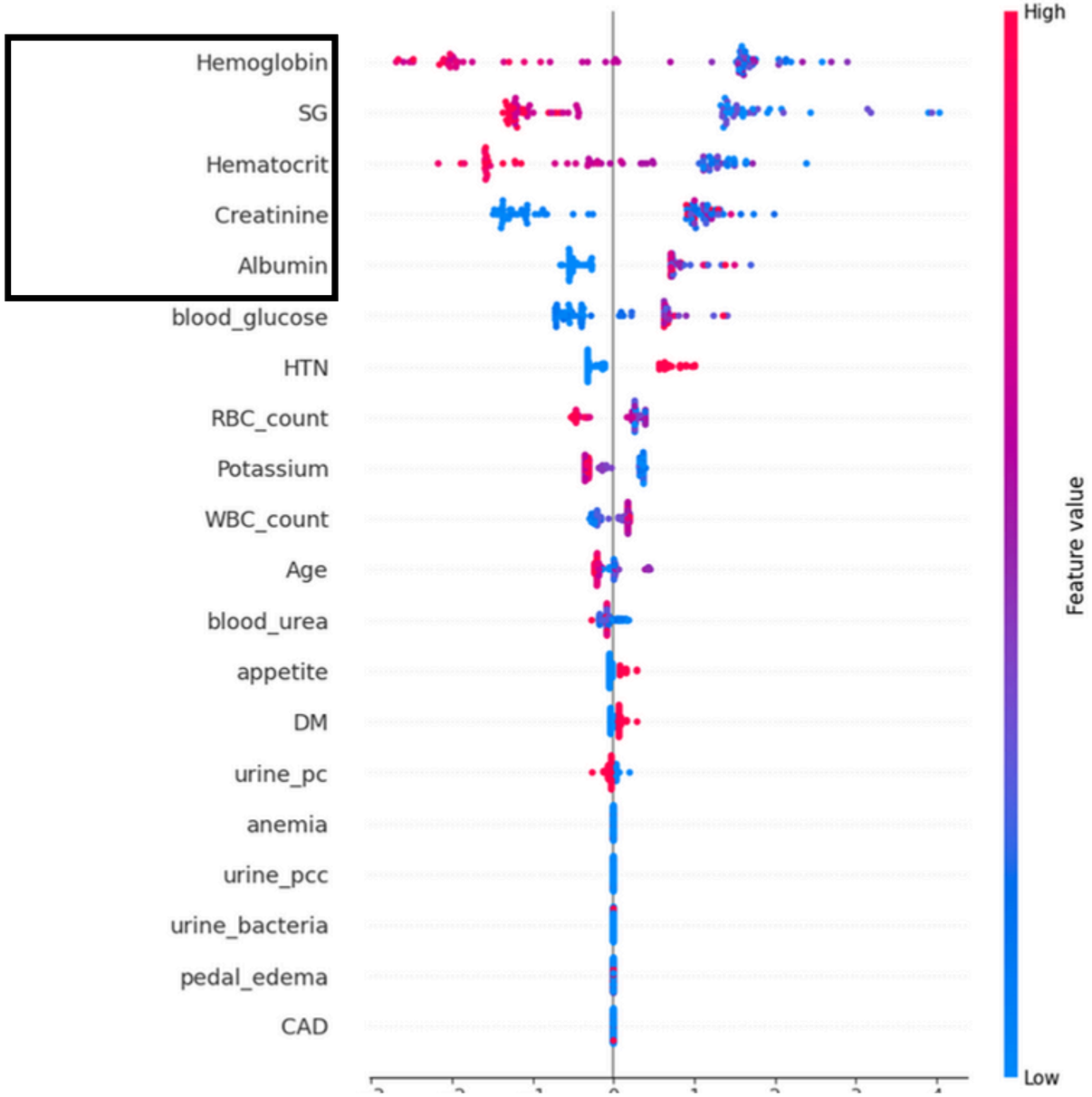
| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.94 | 0.97 | 0.95 | 30 |
| 1 | 0.98 | 0.96 | 0.97 | 30 |
| | | | | |
| **accuracy** | | | 0.96 | 80 |
| **macro avg** | 0.96 | 0.96 | 0.96 | 80 |
| **weighted avg** | 0.96 | 0.96 | 0.96 | 80 |

# 모델 평가

## 03 모델 평가

- **XGBoost Classifier**



# 임상 시나리오

## 농어촌 지역의 CKD 조기 선별

- **배경: 농어촌 지역**
  - 전문 인력이 부족, 정기적 신장 기능 검사 어려움
  - 고령 인구 수, 고혈압·당뇨병 유병률 높은 편임

- **시나리오**

: 농촌 지역 보건소와 협력하는 병원의 시스템에 CKD 선별 모델을 통합하여 주민 정기 건강검진 결과를 해석, CKD 선별

- **실제 상황**

1. 65세 농부 A모씨가 보건소에서 혈액검사와 소변검사를 받음
2. 검체는 분석 기기가 있는 주변 병원으로 24시간 내 이송
3. 검사 결과를 입력하여 CKD 여부를 0(정상), 1(CKD)으로 출력
4. 판정 결과는 보건소에 전달
5. 보건소는 A모씨에게 검사 결과 통지

→ 1(CKD)인 경우: 진료 및 정밀검사를 병원에 의뢰

## 임팩트

고위험 환자 조기 발견 → 신속한 2차 진료 연결

자원 부족 지역에서도 CKD 1~2기 환자 조기 탐지 가능

# 한계점

## 단순 분류 모델

01

CKD 단계나 예후는 반영하지 않음

→ 병기(staging), 진행 위험도, 악화 속도 예측에는 한계

## 시계열 정보 미반영

02

단일 시점의 데이터만 사용함

→ 진행성 질환인 CKD는 연속적인 시계열 정보 반영이 중요
(Cr 변화, 혈압 추이 등)

## 설명력 부족

03

임상의가 CKD 분류 이유의 직관적 파악 어려움

## 일반화 어려움

04

샘플 수 400명

단일 기관 데이터

# Future Work

## 다중 분류 모델 - 2가지

**(1) 정상 / AKI / CKD 분류**

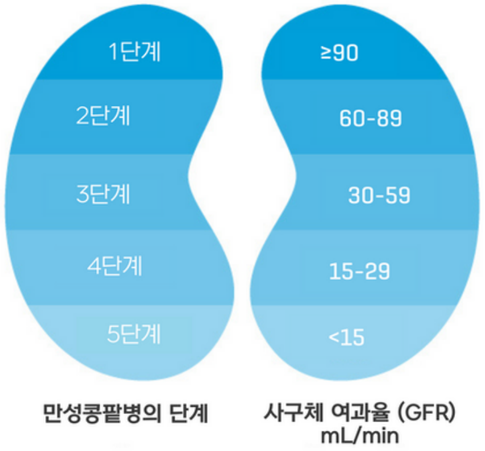
0 = 정상, 1 = CKD, 2 = AKI

- AKI 환자 데이터셋 이용

**(2) eGFR → CKD병기(stage 1~5) 분류**

- 성별, 나이, 혈청 크레아티닌 이용하여 계산

## CKD 위험도 산출 모델




**고위험/중위험/저위험**

- predict_proba()를 활용하여 위험도 산출 (0~1)
- 위험 단계 분류 후 데이터셋을 참고하여 등급화

## EMR 연동

환자의 신기능 실시간 분석 및 CKD 조기선별

- 입원 중 신기능 저하(약물, 수술, 장기간 부동 등)로 인한 CKD 감지
- 과거 기록을 고려하여 판단 가능
- 선별 시 EMR 팝업 메세지 설정

[참고문헌]


1. Trends in the Number of Chronic Kidney Disease Patients and Medical Expenses, During 2012–2022. Public Health Weekly Report 2024; 17(9): 381-382. https://doi.org/10.56786/PHWR.2024.17.9.3
2. Chan, M. R., Dall, A. T., Fletcher, K. E., Lu, N., & Trivedi, H. (2007). Outcomes in patients with chronic kidney disease referred late to nephrologists: A meta-analysis. The American Journal of Medicine, 120(12), 1063–1070. https://doi.org/10.1016/j.amjmed.2007.04.024
3. Tangri, N., Peach, E. J., Franzén, S., Barone, S., & Kushner, P. R. (2023). Patient management and clinical outcomes associated with a recorded diagnosis of stage 3 chronic kidney disease: The REVEAL-CKD study. Advances in Therapy, 40(6), 2869–2885. https://doi.org/10.1007/s12325-023-02482-5
4. Ominext. (2024, November 26). 전자 의무 기록 시스템 (EMR): 개요 및 특징. Ominext Blog. https://www.ominext.com/kr/our-new/what-is-emr-features-examples-and-use-cases
5. MyKidneyJourney. (n.d.). 만성 신장 질환(Chronic Kidney Disease). MyKidneyJourney. https://apac.mykidneyjourney.com/ko/chronic-kidney-disease

2025 여름 디지털 헬스케어 부트캠프 팀 프로젝트 발표

# 감사합니다

2팀 만성신장질환 예측

# 코딩: 양호헌, 이정규

# ppt: 백혜연, 전준서, 최지인

# 발표: 양호헌, 윤혜원