# 기계학습론

## 기계학습을 활용한 반도체 불량 검출 모델링

2017010055 박현일
2017012415 이정인
2018098977 최영훈
2019054193 최현지

CONTENTS


01 서론

02 데이터 분석

03 모델

04 분석 결과

05 결론


서론
데이터 분석
모델
분석 결과
결론

# 문제 정의

반도체 공정 내에서 측정된 여러 값을 분석하여 어떤 표본이 불량인지 아닌지 식별하는 모델을 만들어 불량 반도체를 식별한다. 반도체 제조사가 불량 제품을 직접 확인하지 않고도 어떤 제품이 불량인지 알 수 있게 하여 쉽게 불량 제품과 정상 제품을 분류하도록 한다.

서론
데이터 분석
모델
분석 결과
결론

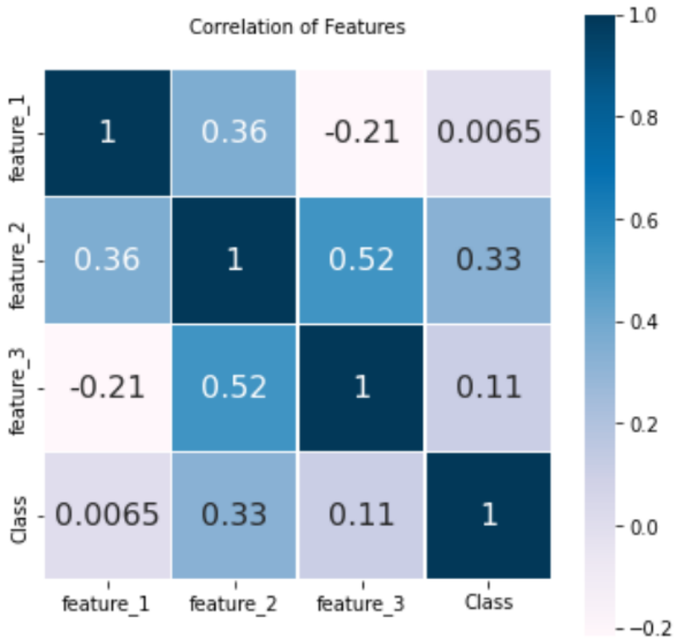
# 데이터 설명

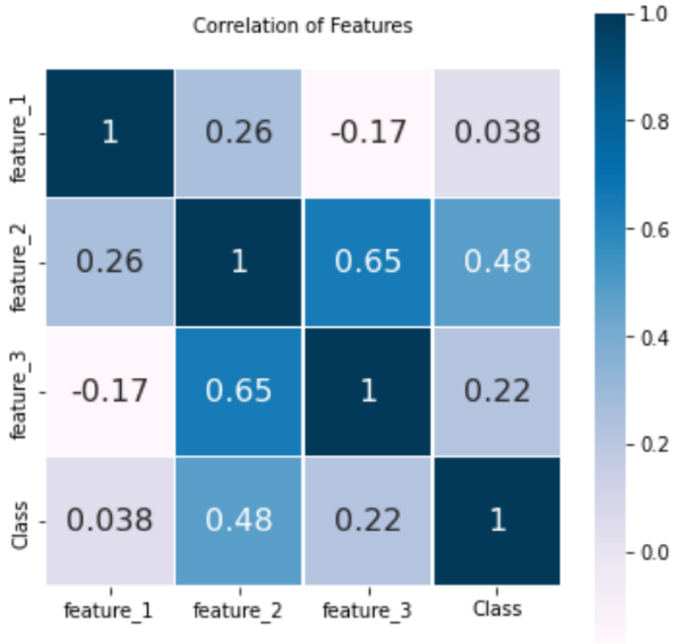
| Feature | 속성 |
|---|---|
| Feature_1 | 수치형 |
| Feature_2 | 수치형 |
| Feature_3 | 수치형 |
| Feature_4 | 이진 범주형 |
| … | … |
| Feature_1558 | 이진 범주형 |
| Class | 이진 범주형 |

| | feature_1 | feature_2 | feature_3 | feature_4 | feature_5 | feature_6 | Class |
|---|---|---|---|---|---|---|---|
| 0 | 100 | 160 | 1.6000 | 0 | 0 | 0 | 0 |
| 1 | 20 | 83 | 4.1500 | 1 | 0 | 0 | 0 |
| 2 | 99 | 150 | 1.5151 | 1 | 0 | 0 | 0 |
| 3 | 40 | 40 | 1.0000 | 0 | 0 | 0 | 0 |
| 4 | 12 | 234 | 19.5000 | 1 | 0 | 0 | 0 |
| 5 | 90 | 90 | 1.0000 | 0 | 0 | 0 | 0 |
| 6 | 1 | 1 | 2.0000 | 1 | 0 | 0 | 0 |
| 7 | 15 | 80 | 5.3333 | 0 | 0 | 0 | 0 |

서론
데이터 분석
모델
분석 결과
결론

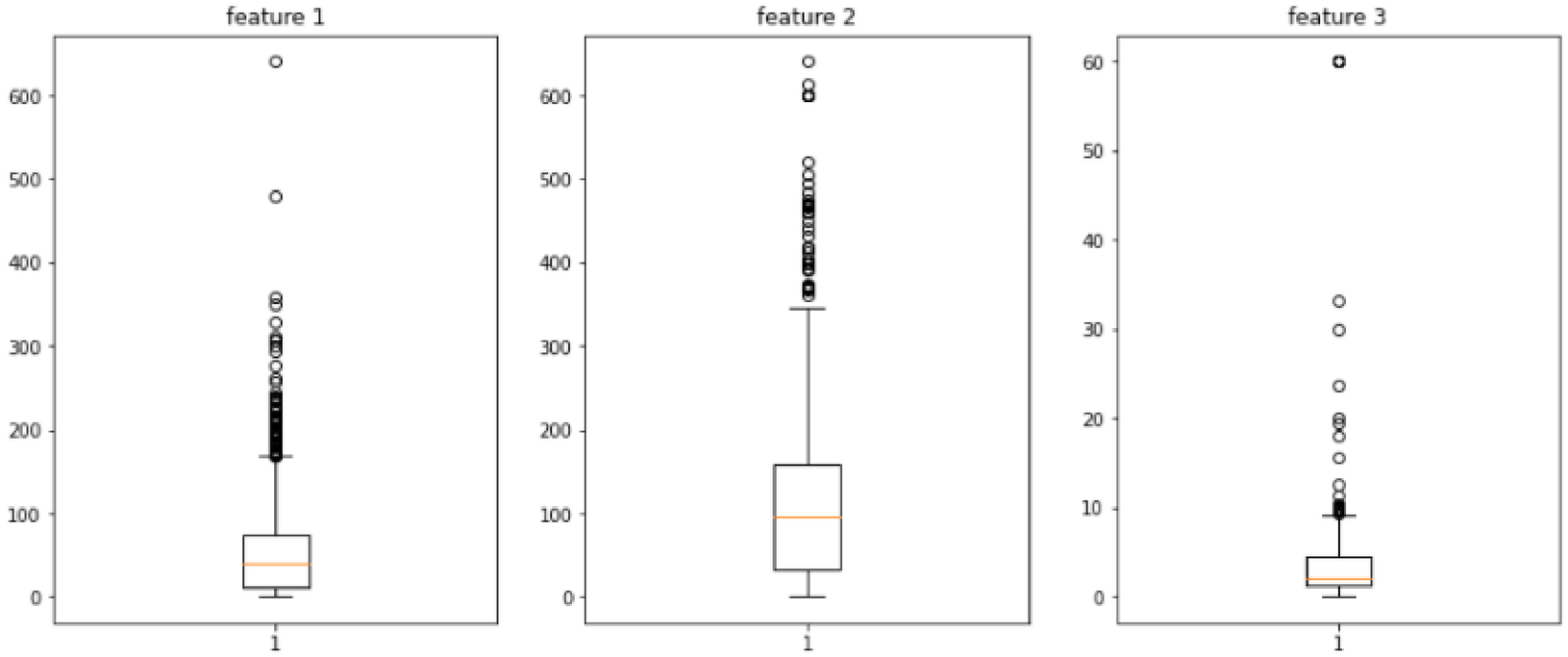
# 시각화 – Feature 간 상관관계



서론
데이터 분석
모델
분석 결과
결론

# 시각화 – 수치형 데이터 Boxplot

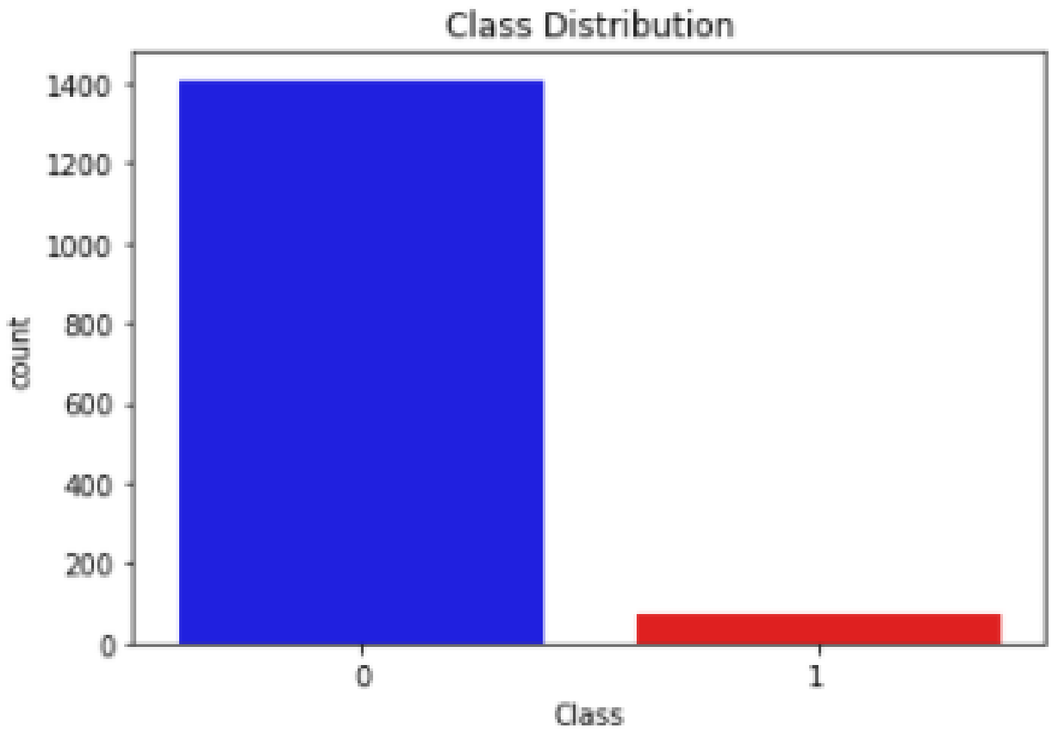


서론
데이터 분석
모델
분석 결과
결론

# 시각화 – 클래스 분포

양품 – 91.89%

불량 – 8.11%




# 전처리 – Train set, Test set 분리

서론
—
데이터 분석
모델
분석 결과
결론

## Train set, Test set 분리

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=777, stratify=Y)
X_train.reset_index(drop=True, inplace=True)
X_test.reset_index(drop=True, inplace=True)
Y_train.reset_index(drop=True, inplace=True)
Y_test.reset_index(drop=True, inplace=True)
```

```
# 분리 확인
print(X_train.shape, X_test.shape, Y_train.shape, Y_test.shape)
```

```
(1186, 1522) (297, 1522) (1186,) (297,)
```


# 전처리 – 중복 레코드 처리

서론
데이터 분석
모델
분석 결과
결론

## 중복 레코드 제거

```
temp = pd.concat([X_train, Y_train], axis=1)
```

```
temp.drop_duplicates(keep='first', inplace=True)
```

```
Y_train = temp['Class']
X_train = temp.drop(['Class'], axis=1)
```


# 전처리 – 단일 값을 갖는 feature 제거

서론
데이터 분석
모델
분석 결과
결론

## 단일 클래스를 가진 피쳐 제거

```
for i in X_train.columns:
    if len(X_train[i].unique()) == 1:
        X_train.drop(i, axis=1, inplace=True)
        X_test.drop(i, axis=1, inplace=True)
```

```
# 감소한 피쳐 수 확인
print(X_train.shape, X_test.shape)
```

```
(1378, 1515) (353, 1515)
```

서론
데이터 분석
모델
분석 결과
결론

# 전처리 – 이상치 제거

## Outlier 처리 #1 iqr 기준 삭제

```
toScale = ["feature_1", "feature_2" , "feature_3"]
```

```
trimmer = OutlierTrimmer(capping_method='iqr', tail='both', fold=3, variables=toScale)
X_train = trimmer.fit_transform(X_train)
```


# 전처리 – Feature selection

서론
데이터 분석
모델
분석 결과
결론

## Feature간 상관계수 계산 후 제거 (코드 일부)

```
for i in overCorr.index:
    try:
        if corrX_Y[corrX_Y['features'] == overCorr.loc[i]['INDEX']]['corr'].values[0] <= \
        corrX_Y[corrX_Y['features'] == overCorr.loc[i]['COLUMN']]['corr'].values[0]:
            X_train.drop([overCorr.loc[i]['INDEX']], axis=1, inplace=True)
            X_test.drop([overCorr.loc[i]['INDEX']], axis=1, inplace=True)
        else:
            X_train.drop([overCorr.loc[i]['COLUMN']], axis=1, inplace=True)
            X_test.drop([overCorr.loc[i]['COLUMN']], axis=1, inplace=True)
    except:
        continue
```

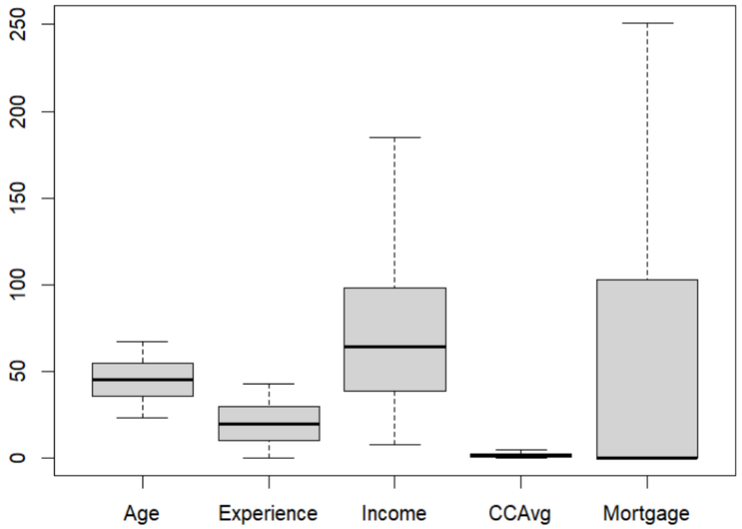
```
X_train.shape
```

```
iO]:  (1360, 412)
```

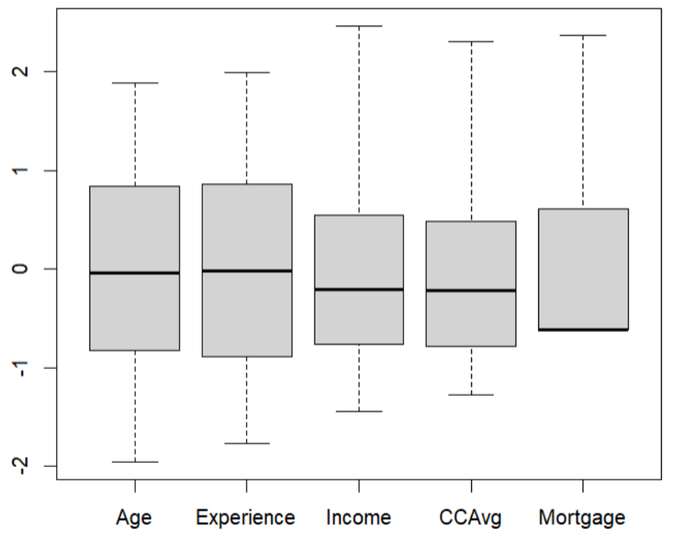

# 전처리 – Scaling

서론
데이터 분석
모델
분석 결과
결론

Original DATA




Standardized DATA




# 전처리 – Scaling

서론
데이터 분석
모델
분석 결과
결론

## Feature Scaling #1 StandardScaler

```
scaler = StandardScaler()
for feature in toScale:
    X_train.loc[:, feature] = scaler.fit_transform(X_train[feature].to_numpy().reshape(-1, 1))
    X_test.loc[:, feature] = scaler.transform(X_test[feature].to_numpy().reshape(-1, 1))
```

## Feature Scaling #2 MinMaxScaler

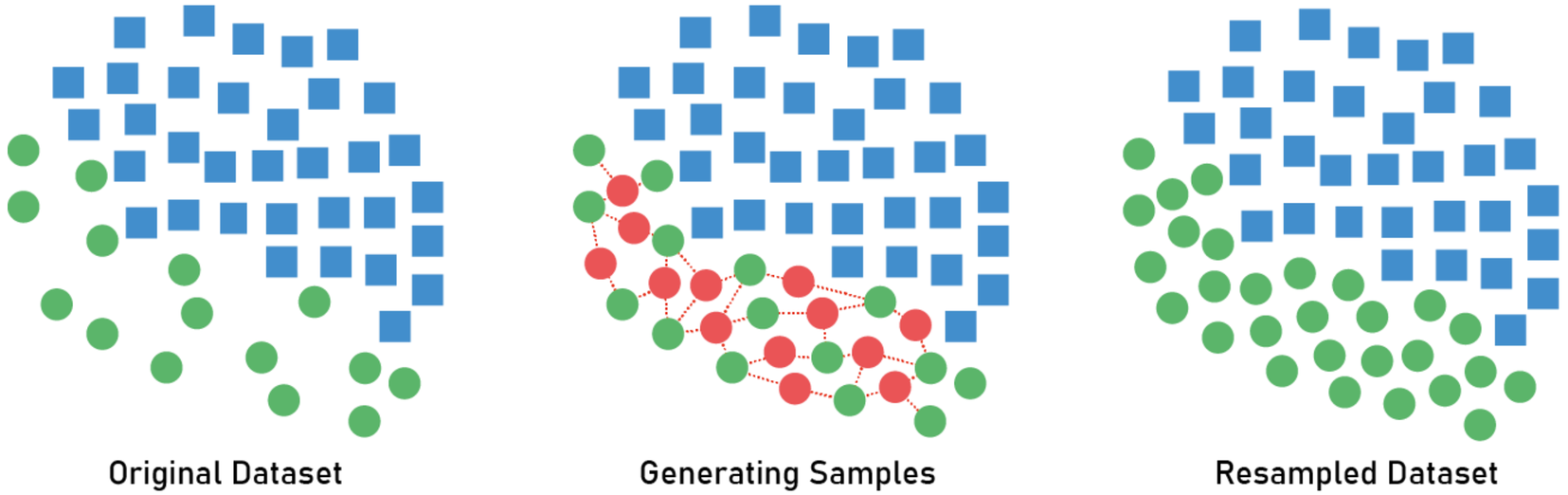
```
mmScaler = MinMaxScaler()
for feature in toScale:
    X_train.loc[:, feature] = mmScaler.fit_transform(X_train[feature].to_numpy().reshape(-1, 1))
    X_test.loc[:, feature] = mmScaler.transform(X_test[feature].to_numpy().reshape(-1, 1))
```

서론
데이터 분석
모델
분석 결과
결론

# 전처리 – 클래스 불균형

임의의 소수 클래스 데이터로부터 인근 소수 클래스 사이에 새로운 데이터를 생성하여 클래스 불균형을 해소하는 기법

## Synthetic Minority Oversampling Technique




서론
데이터 분석
모델
분석 결과
결론

# 전처리 – Oversampling

## 오버샘플링 #1 SMOTE

```
smote = SMOTE(random_state=2, k_neighbors=3)
X_train, Y_train = smote.fit_resample(X_train, Y_train)
```

## 오버샘플링 #2 RamdomOverSampling

```
randomOver = RandomOverSampler(random_state=0)
X_train, Y_train = randomOver.fit_resample(X_train, Y_train)
```

```
# Y 밸런스 확인
Y_train.value_counts()
```

```
: 0    1127
  1    1114
  Name: Class, dtype: int64
```


서론
데이터 분석
모델
분석 결과
결론

# 전처리 – Class weight

모델을 학습할 때 소수 클래스에 가중치를 주어

학습함으로써 클래스 불균형을 해소

```
weight_0 = (1 / Y_train.value_counts()[0]) * (Y_train.value_counts().sum() / 2.0)
weight_1 = (1 / Y_train.value_counts()[1]) * (Y_train.value_counts().sum() / 2.0)
class_weight = {0 : weight_0, 1 : weight_1}
print(class_weight)
```

```
{0: 0.545308740978348, 1: 6.017699115044247}
```

서론

데이터 분석

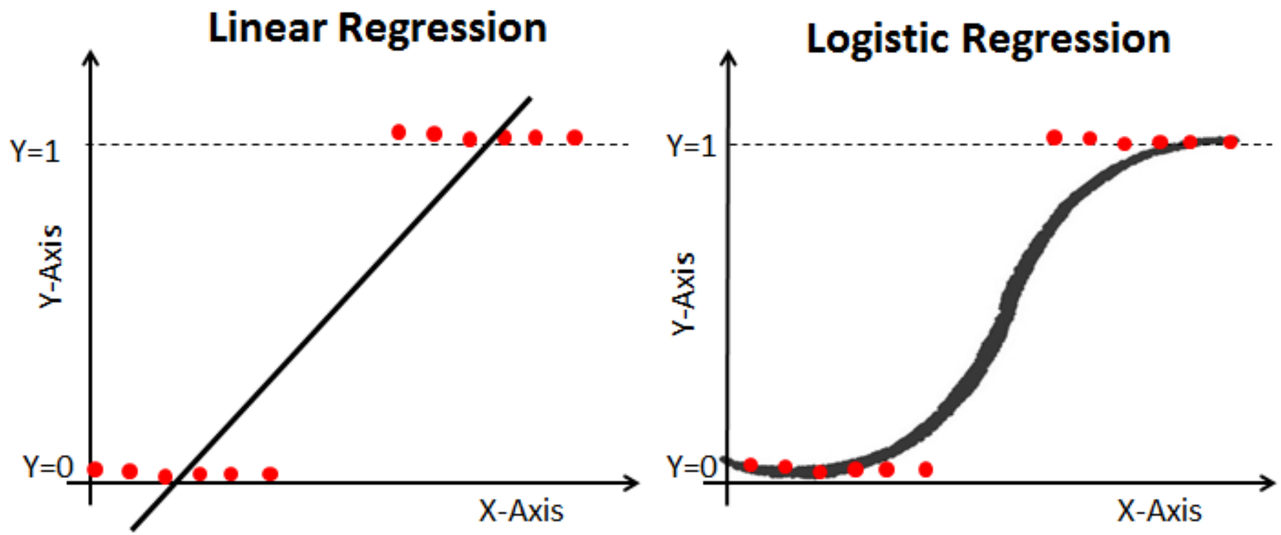
모델

분석 결과

결론

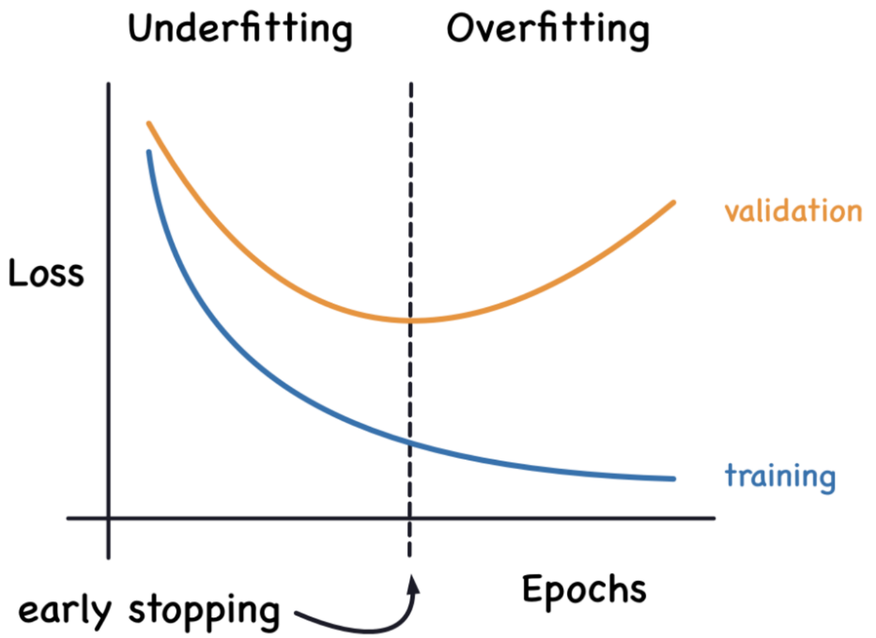
# 모델 소개 – Keras Logistic regression

회귀를 사용하여 데이터가 어떤 클래스에 속할 확률을 0 과 1 사이의 값으로 예측하고 그 확률에 따라 분류하는 모델



서론
데이터 분석
모델
분석 결과
결론

# 과적합 방지




$$L(x,y) \equiv \sum_{i=1}^{n} (y_i - h_\theta(x_i))^2 + \lambda \sum_{i=1}^{n} \theta_i^2$$


서론

데이터 분석

모델

분석 결과

결론

# 성능 지표

| | Accuracy | Precision | Recall | F1-score | ROC-AUC |
|---|---|---|---|---|---|
| Keras logistic regression | 0.917847 | 0.500000 | 0.655172 | 0.567164 | 0.932024 |
| Keras MLP | 0.900850 | 0.384615 | 0.344828 | 0.363636 | 0.863144 |
| Keras DNN | 0.906516 | 0.400000 | 0.275862 | 0.326531 | 0.824021 |
| Sklearn logistic regression | 0.909348 | 0.463415 | 0.655172 | 0.542857 | 0.793636 |
| Sklearn logistic regression - GridSearchCV | 0.909348 | 0.463415 | 0.655172 | 0.542857 | 0.793636 |
| LightGBM - GridSearchCV | 0.917847 | 0.500000 | 0.551724 | 0.524590 | 0.751171 |
| LightGBM | 0.917847 | 0.500000 | 0.551724 | 0.524590 | 0.751171 |
| Sklearn Decision Tree | 0.889518 | 0.375000 | 0.517241 | 0.434783 | 0.722201 |
| Sklearn Decision Tree - GridSearchCV | 0.892351 | 0.384615 | 0.517241 | 0.441176 | 0.721584 |
| Sklearn RandomForest | 0.929178 | 0.600000 | 0.413793 | 0.489796 | 0.648334 |
| Sklearn RandomForest - GridSearchCV | 0.917847 | 0.500000 | 0.275862 | 0.355556 | 0.625585 |
| XGBoost | 0.917847 | 0.000000 | 0.000000 | 0.000000 | 0.500000 |
| XGBoost - GridSearchCV | 0.917847 | 0.000000 | 0.000000 | 0.000000 | 0.500000 |


서론
데이터 분석
모델
분석 결과
결론

# 결론

IQR 3

MaxAbs

0.93202427

Weight

Logistic

# 감사합니다