스마트 공장 제품 품질 상태 분류
AI 온라인 해커톤

Team : NTJ 공장

이영인 annmismatch@gmail.com
이형민 bm199564@naver.com
이찬미 ddrt44al@naver.com

Index

1. 개발환경

패키지 및 Version 정보

Google Colab

Python 3.8.10

pandas 1.3.5

numpy 1.22.4

2. EDA

2.1 결측치 확인

Train 결측치

X_1	X_2	X_3	X_4	...	X_2866	X_2867	X_2868	X_2869	X_2870	X_2871	X_2872	X_2873	X_2874	X_2875
NaN	NaN	NaN	NaN	...	39.34	40.89	32.56	34.09	77.77	NaN	NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN	...	38.89	42.82	43.92	35.34	72.55	NaN	NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN	...	39.19	36.65	42.47	36.53	78.35	NaN	NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN	...	37.74	39.17	52.17	30.58	71.78	NaN	NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN	...	38.70	41.89	46.93	33.09	76.97	NaN	NaN	NaN	NaN	NaN

Test 결측치

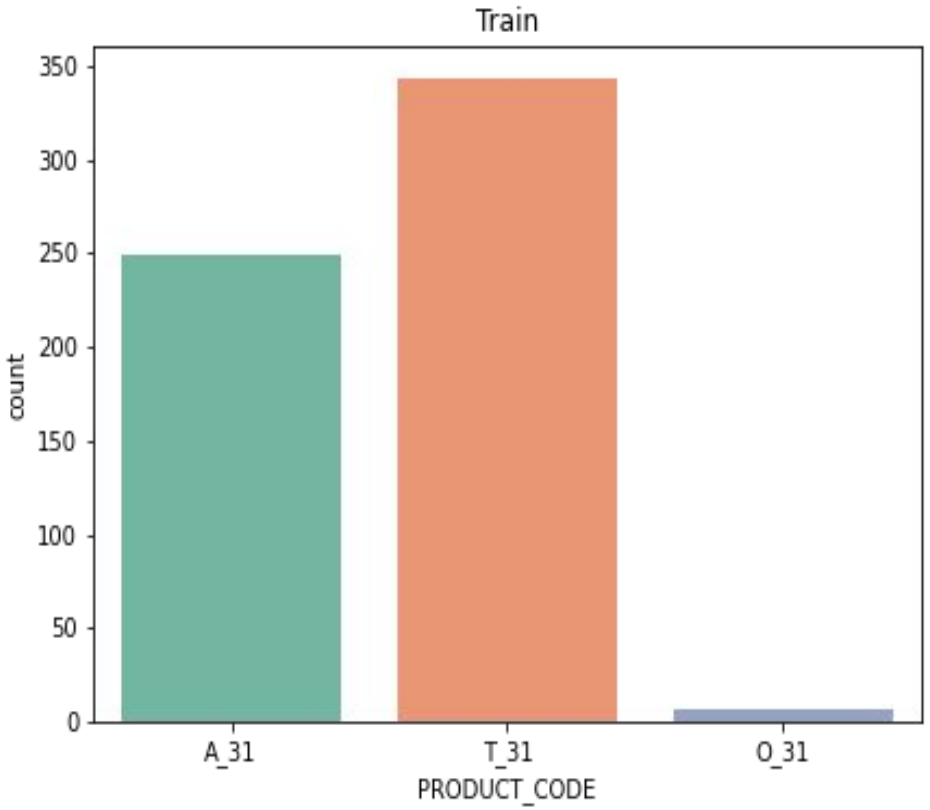
X_1	X_2	X_3	X_4	X_5	X_6	...	X_2866	X_2867	X_2868	X_2869	X_2870	X_2871	X_2872	X_2873	X_2874	X_2875
2.0	94.0	0.0	45.0	10.0	0.0	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2.0	93.0	0.0	45.0	11.0	0.0	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2.0	95.0	0.0	45.0	11.0	0.0	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

➡ PRODUCT_CODE, LINE에 따라 특정 열에 다수의 결측치가 존재함

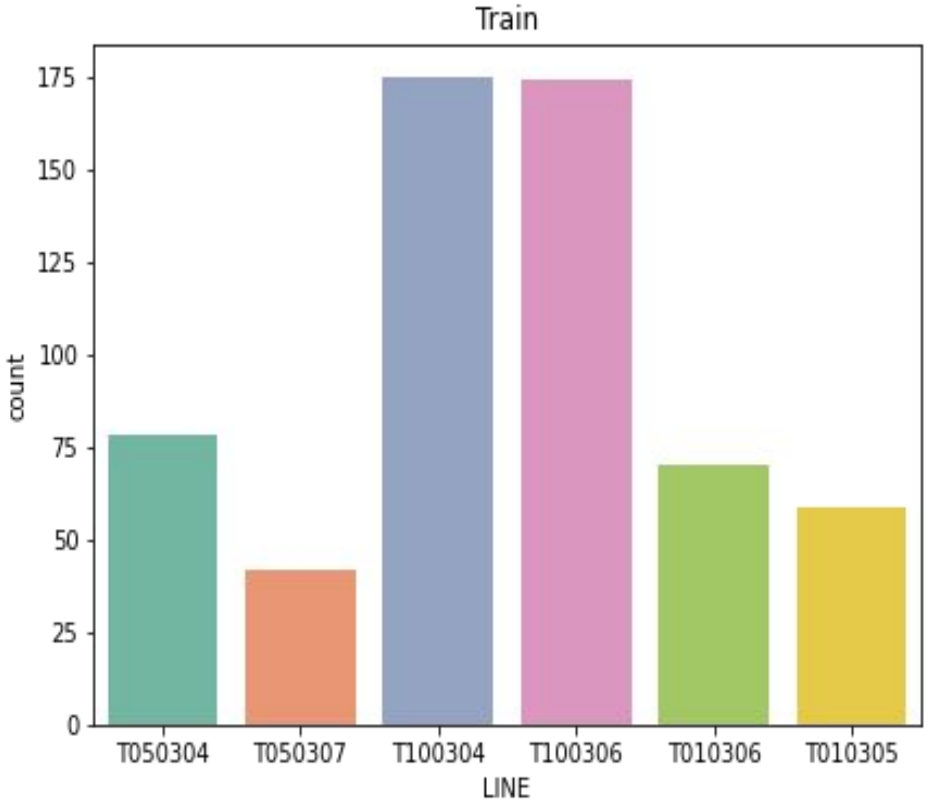
2. EDA

2.2 데이터 구성 탐색

PRODUCT_CODE 분포 확인



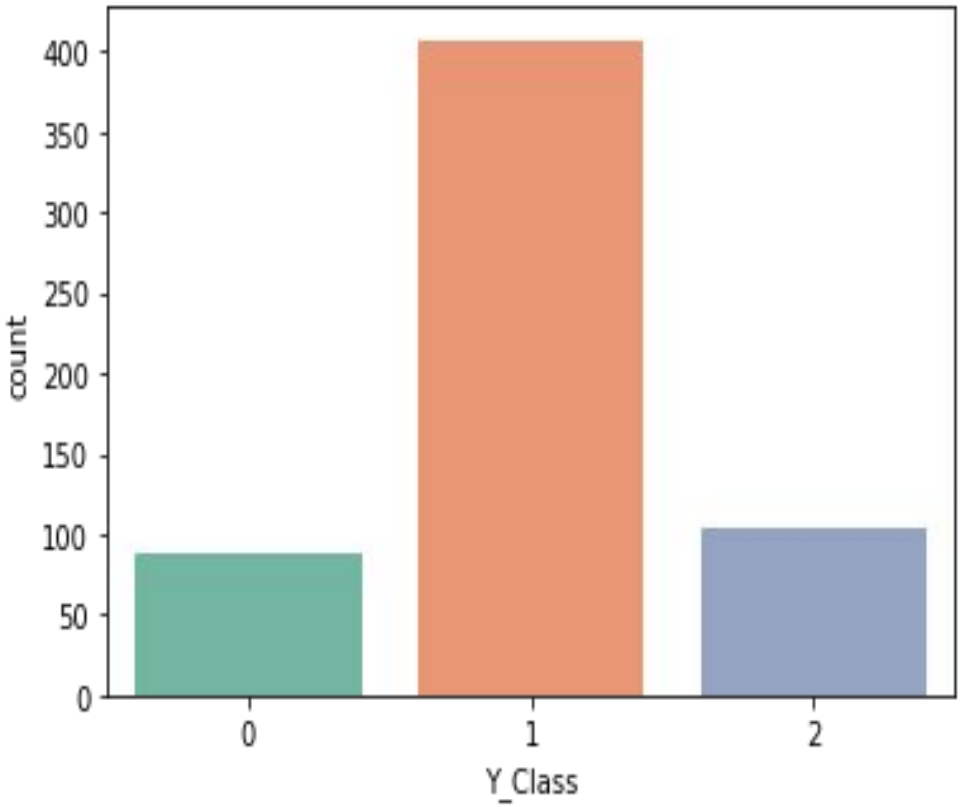
LINE 분포 확인



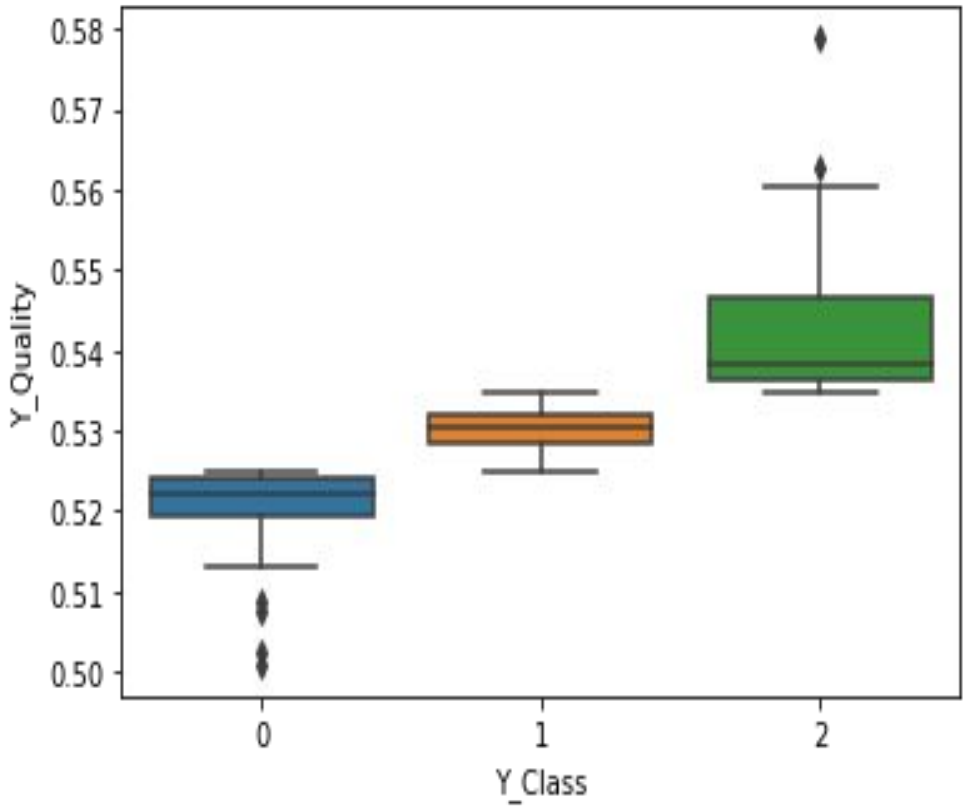
2. EDA

2.2 데이터 구성 탐색

전체 train data 중 Y_Class 분포 확인



Y_Class에 따른 Y_Quality 확인

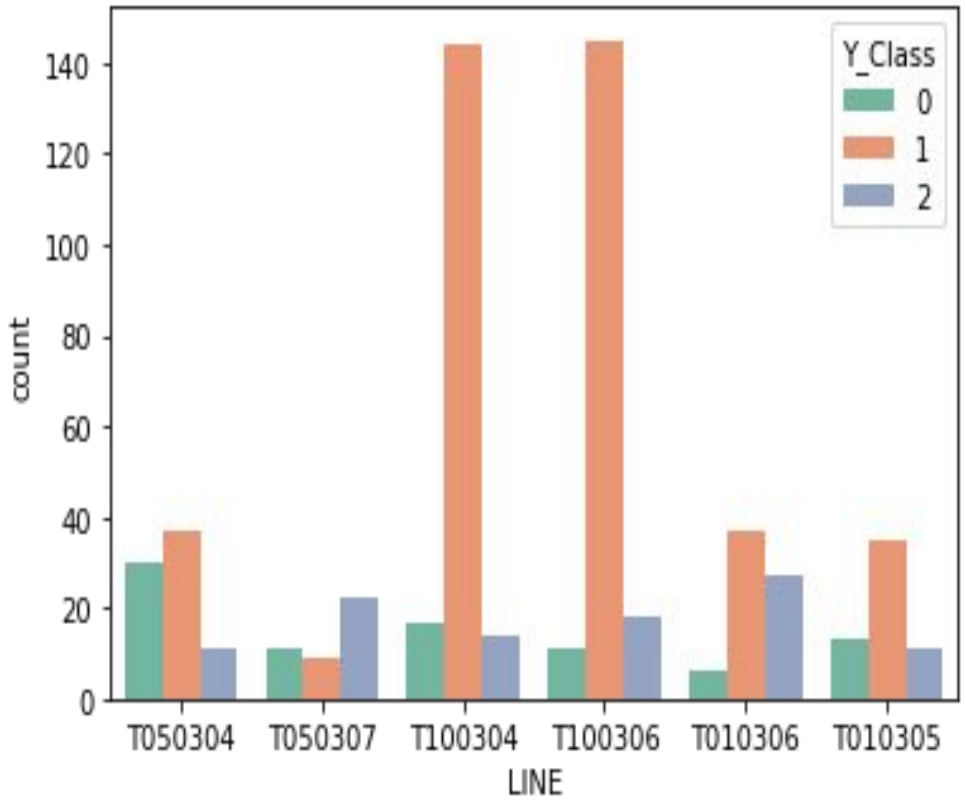


→ 0 < 1 < 2 구간차이가 뚜렷한 편

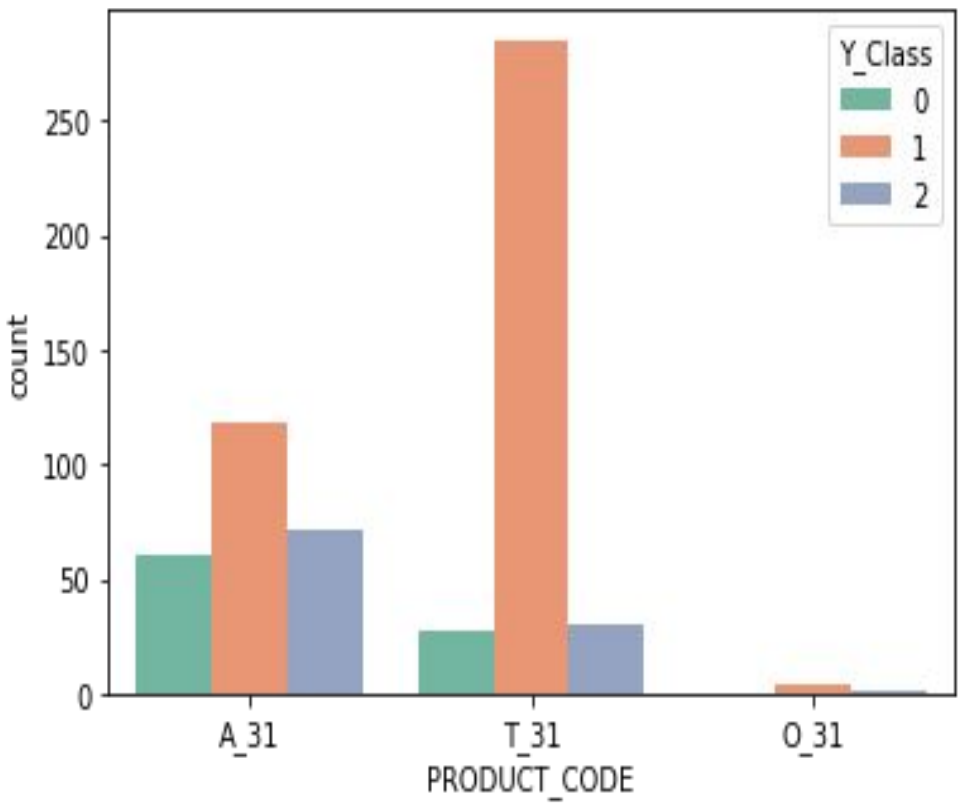
2. EDA

2.2 데이터 구성 탐색

LINE 별 Y_Class 분포 확인



PRODUCT_CODE 별 Y_Class 분포 확인



2. EDA

2.3 Data pipeline 구축

결측치를 채우기 위해 다양한 방법을 시도

▼

test data를 사전에 알 수 없음
→ **test data** 사용 시 **Data Leakage** 발생 위험
비식별화된 변수 **(X_1 ~ X_2875)**가 많음
→ **data** 변수 별 특성을 제한하기 어려움

▼

최대한 원본 **data**의 특성을 유지하고자 함

3. Feature Engineering

라벨 인코딩

```
# qualitative to quantitative
qual_col = ['PRODUCT_CODE', 'LINE']

for i in qual_col:
    le = LabelEncoder()
    le = le.fit(train[i])
    train[i] = le.transform(train[i])

    for label in np.unique(test[i]):
        if label not in le.classes_:
            le.classes_ = np.append(le.classes_, label)
    test[i] = le.transform(test[i])
print('Done.')
```

```
Done.
```

라벨 인코딩 전

```
train['PRODUCT_CODE'].unique()
```

```
array(['A_31', 'T_31', 'O_31'], dtype=object)
```

```
train['LINE'].unique()
```

```
array(['T050304', 'T050307', 'T100304', 'T100306', 'T010306', 'T010305'],
      dtype=object)
```

라벨 인코딩 후

```
train['PRODUCT_CODE'].unique()
```

```
array([0, 2, 1])
```

```
train['LINE'].unique()
```

```
array([2, 3, 4, 5, 1, 0])
```

3. Feature Engineering

정규화

```
from sklearn.preprocessing import MinMaxScaler

x_col = train.columns[train.columns.str.contains('X')].tolist()
scaler = MinMaxScaler()

scaler.fit(train_x[x_col])

train_x[x_col] = scaler.transform(train_x[x_col])
test_x[x_col] = scaler.transform(test_x[x_col])
```

정규화 전

```
train_x.loc[:3,'X_2866':'X_2870']
```

	X_2866	X_2867	X_2868	X_2869	X_2870
0	39.34	40.89	32.56	34.09	77.77
1	38.89	42.82	43.92	35.34	72.55
2	39.19	36.65	42.47	36.53	78.35
3	37.74	39.17	52.17	30.58	71.78

정규화 후

```
train_x.loc[:3,'X_2866':'X_2870']
```

	X_2866	X_2867	X_2868	X_2869	X_2870
0	0.256757	0.248647	0.000000	0.122283	0.890487
1	0.240754	0.300866	0.407899	0.164742	0.601770
2	0.251422	0.133929	0.355835	0.205163	0.922566
3	0.199858	0.202110	0.704129	0.003057	0.559181

3. Feature Engineering

전체 평균으로 결측값 채우기

```
train_x = train_x.fillna(train_x.mean())
test_x = test_x.fillna(train_x.mean())
```

결측치 채우기 전

```
train_x.loc[592:,:'X_8']
```

	LINE	PRODUCT_CODE	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8
592	4	2	0.009804	0.733333	0.0	0.0	0.0	0.0	0.000000	0.0
593	5	2	0.009804	0.533333	0.0	0.0	0.0	0.0	0.294118	0.0
594	2	0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
595	2	0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
596	4	1	0.382353	0.466667	0.0	0.0	1.0	0.0	0.000000	0.0
597	5	1	0.196078	0.000000	0.0	0.0	0.0	0.0	0.941176	0.0

결측치 채운 후

```
train_x.loc[592:,:'X_8']
```

	LINE	PRODUCT_CODE	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8
592	4	2	0.009804	0.733333	0.0	0.0	0.00000	0.0	0.000000	0.000000
593	5	2	0.009804	0.533333	0.0	0.0	0.00000	0.0	0.294118	0.000000
594	2	0	0.013821	0.541547	0.0	0.0	0.39255	0.0	0.223664	0.048711
595	2	0	0.013821	0.541547	0.0	0.0	0.39255	0.0	0.223664	0.048711
596	4	1	0.382353	0.466667	0.0	0.0	1.00000	0.0	0.000000	0.000000
597	5	1	0.196078	0.000000	0.0	0.0	0.00000	0.0	0.941176	0.000000

3. Feature Engineering

결측값 제거

```
train_x = train_x.dropna(axis =1)
test_x = test_x.dropna(axis=1)
```

결측치 제거 전 데이터 프레임 **shape** 확인

```
train_x.shape, train_y.shape, test_x.shape
```

```
((598, 2877), (598,), (310, 2877))
```

결측치 제거 후 데이터 프레임 **shape** 확인

```
train_x.shape, train_y.shape, test_x.shape
```

```
((598, 2795), (598,), (310, 2795))
```

4. Modeling

Tree model 중 성능 비교를 위해 voting 사용

```
#lightGBM, ExtraTreesClassifier, RandomForestClassifier 선택
model = lgbm.LGBMClassifier()

et_cls = ExtraTreesClassifier(n_estimators=500, min_samples_leaf=5, min_samples_split=7, max_features=2795)
rf_cls = RandomForestClassifier(n_estimators=500, min_samples_leaf=5, min_samples_split=7, max_features=2795)
lg_cls = model.fit(train_x, train_y)

# 모델 voting
voting = VotingClassifier(
    estimators=[
        ('et', et_cls),
        ('rf', rf_cls),
        ('lg', lg_cls)
    ]
)
voting.fit(train_x, train_y)
```

5. 결과도출

```
pred = voting.predict(test_x)
sub['Y_Class'] = pred
```

예측값 확인

```
pred
```

```
array([1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 2, 1, 1, 0, 0, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0,
       0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 0, 1,
       0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0,
       0, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 2, 2, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 2, 1, 1, 0,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0,
       0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1])
```


감사합니다