스마트 공장 제품 품질 상태 분류
AI 오프라인 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

2.2. 데이터 파악

```
-- LINE --
{'T100304': 404, 'T100306': 434, 'T010305': 66, 'T050307': 66, 'T050304': 86, 'T010306': 76}

-- PRODUCT_CODE --
{'T_31': 830, 'A_31': 294, 'O_31': 8}
```

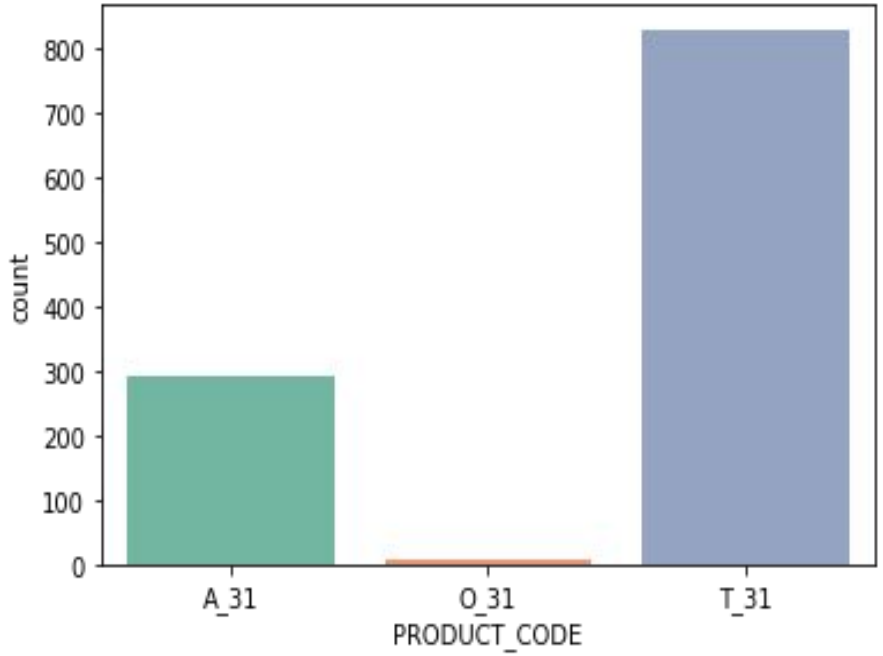
LINE	T010305	T010306	T050304	T050307	T100304	T100306
PRODUCT_CODE						
A_31	66.0	76.0	86.0	66.0	NaN	NaN
O_31	NaN	NaN	NaN	NaN	4.0	4.0
T_31	NaN	NaN	NaN	NaN	400.0	430.0

➡ PIVOT을 통해 PRODUCT_CODE 마다 LINE이 다르게 존재하는 것을 확인

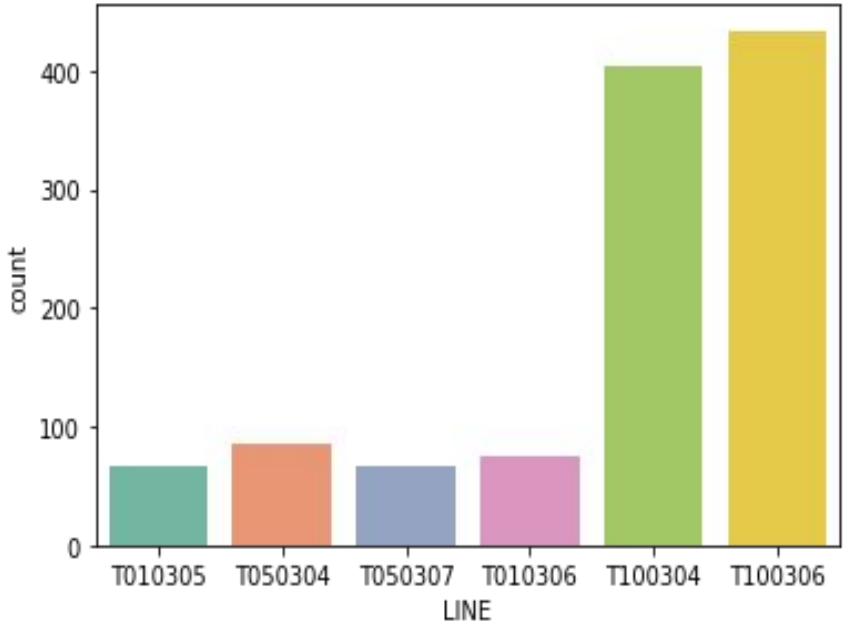
2. EDA

2.3 데이터 구성 탐색

PRODUCT_CODE 분포 확인



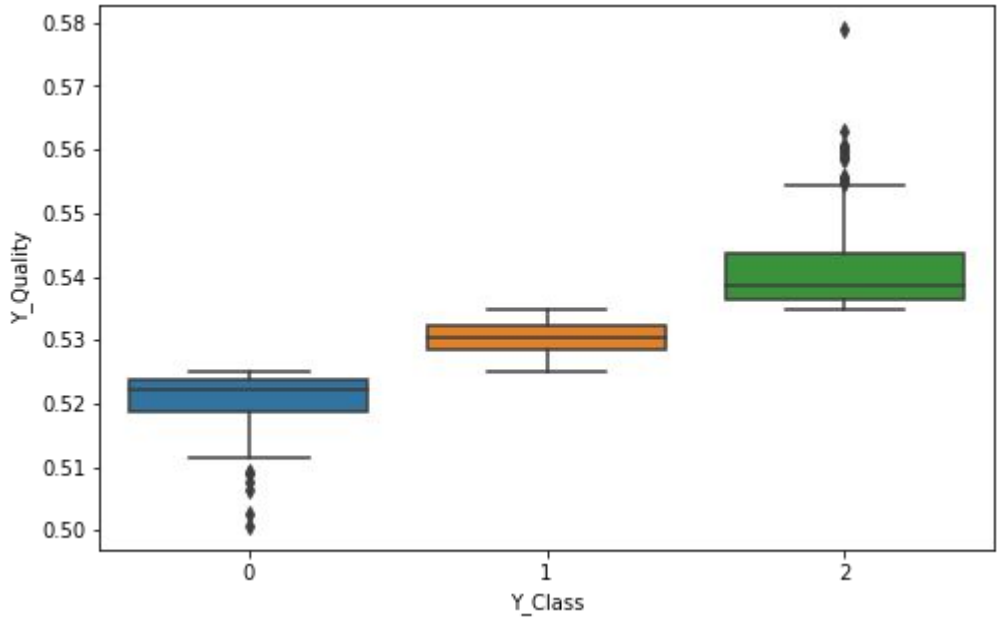
LINE 분포 확인



2. EDA

2.3 데이터 구성 탐색

Y_Class에 따른 Y_Quality 확인



→ 0 < 1 < 2 구간 차이 존재

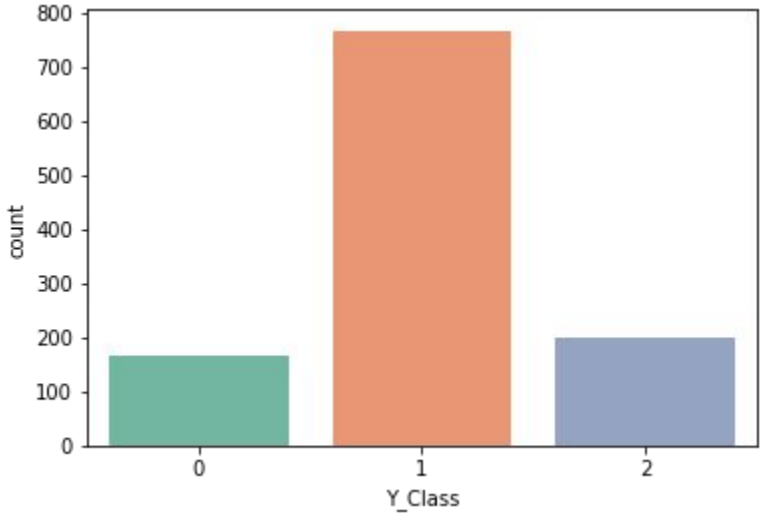
	min			max		
PRODUCT_CODE	A_31	O_31	T_31	A_31	O_31	T_31
Y_Class						
0	0.500856	NaN	0.502517	0.525046	NaN	0.525067
1	0.525086	0.525916	0.525114	0.534843	0.533702	0.534837
2	0.535114	0.534951	0.534951	0.578841	0.535205	0.560454

PRODUCT_CODE 별로 Y_Quality의 min, max 값이 다름

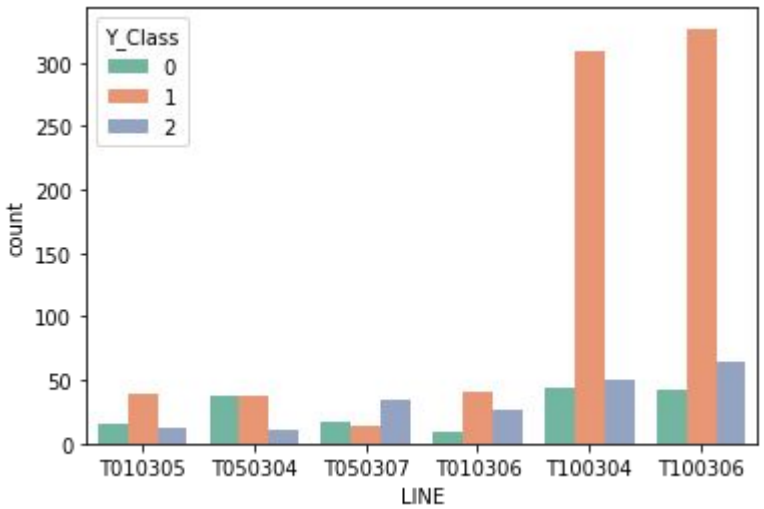
2. EDA

2.3 데이터 구성 탐색

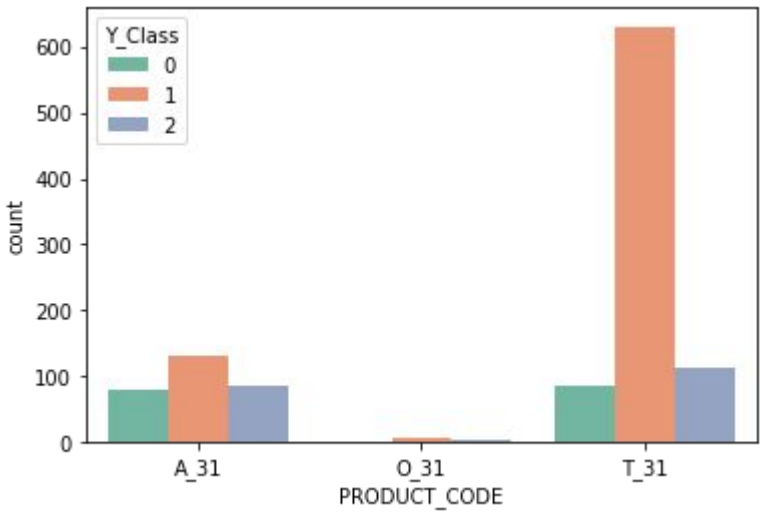
Y_Class 분포 확인



전체 train data



LINE 별



PRODUCT_CODE 별

2. EDA

2.3 Data pipeline 구축

결측치를 채우기 위해 다양한 방법을 시도
ex) 최빈값, 평균값, imputer 활용 예측값

▼

test data를 사전에 알 수 없음
→ test data 사용 시 Data Leakage 발생 위험
비식별화된 변수 (X_1 ~ X_3326)
→ data 변수 별 특성을 제한하기 어려움

▼

최대한 원본 data의 특성을 유지하고자
train 열 평균으로 결측치 보간 하고자 함

3. 가설 설정

스마트 팩토리 공장 특성 상, 설계 프로세스가 정형화 되어 있음

- 내부 문제 발생 가능성 보다 외부 문제 발생 가능성이 높음
- X변수 전체보다 일부 X변수가 분류에 영향을 미칠 것으로 예상

▼

비식별화된 X 변수 전처리 과정 중 중복 값으로 채워진 경우
분류에 큰 영향을 미치지 않는 것으로 판단하여 삭제

	LINE	PRODUCT_CODE	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	...	X_3317	X_3318	X_3319
0	4	2	0.006536	0.533333	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.654045	0.002252	0.531399
1	5	2	0.006536	0.600000	0.0	0.0	0.0	0.0	0.363636	0.0	...	0.604131	0.144144	0.027793
2	5	2	0.006536	0.533333	0.0	0.0	0.0	0.0	0.681818	0.0	...	NaN	NaN	NaN
3	5	2	0.006536	0.000000	0.0	0.0	0.0	0.0	0.363636	0.0	...	0.495697	0.191441	0.117571
4	5	2	0.006536	0.533333	0.0	0.0	0.0	0.0	0.272727	0.0	...	NaN	NaN	NaN

5 rows × 3328 columns

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.loc[10:13, 'X_1':'X_8']
```

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8
10	1.0	94.0	0.0	45.0	10.0	0.0	45.0	10.0
11	2.0	97.0	0.0	45.0	11.0	0.0	45.0	10.0
12	3.0	91.0	0.0	45.0	10.0	0.0	51.0	10.0
13	2.0	89.0	0.0	45.0	10.0	0.0	51.0	10.0

정규화 후

```
train_x.loc[10:13, 'X_1':'X_8']
```

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8
10	0.000000	0.466667	0.0	0.0	0.0	0.0	0.000000	0.0
11	0.006536	0.666667	0.0	0.0	1.0	0.0	0.000000	0.0
12	0.013072	0.266667	0.0	0.0	0.0	0.0	0.272727	0.0
13	0.006536	0.133333	0.0	0.0	0.0	0.0	0.272727	0.0

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[1127:1130, 'X_1':'X_7']
```

	X_1	X_2	X_3	X_4	X_5	X_6	X_7
1127	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1128	0.006536	1.000000	0.0	0.0	1.0	0.0	0.000000
1129	0.000000	0.066667	0.0	0.0	0.0	0.0	0.272727
1130	NaN	NaN	NaN	NaN	NaN	NaN	NaN

결측치 채운 후

```
train_x.loc[1127:1130, 'X_1':'X_7']
```

	X_1	X_2	X_3	X_4	X_5	X_6	X_7
1127	0.011473	0.530231	0.0	0.005967	0.335322	0.0	0.163051
1128	0.006536	1.000000	0.0	0.000000	1.000000	0.0	0.000000
1129	0.000000	0.066667	0.0	0.000000	0.000000	0.0	0.272727
1130	0.011473	0.530231	0.0	0.005967	0.335322	0.0	0.163051

3. Feature Engineering

중복 값인 열 삭제하기

001. train_x 행, 열 바꾸기

```
train_xt = train_x.transpose()
train_xt
```

	0	1	2	3	4	5	6	7	8	9
LINE	4.000000	5.000000	5.000000	5.000000	5.000000	5.000000	5.000000	4.000000	5.000000	4.000000
PRODUCT_CODE	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000
X_1	0.006536	0.006536	0.006536	0.006536	0.006536	0.006536	0.006536	0.000000	0.013072	0.307190
X_2	0.533333	0.600000	0.533333	0.000000	0.533333	0.533333	0.533333	0.666667	0.400000	0.666667
X_3	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
...	...	...	...	...	...	...	...	...	...	...
X_3322	0.068966	0.758621	0.463106	0.844828	0.463106	0.720690	0.463106	0.463106	0.693103	0.310345
X_3323	0.374749	0.174349	0.330111	0.468938	0.330111	0.571142	0.330111	0.330111	0.232465	0.995992
X_3324	0.382716	0.703036	0.583531	0.759760	0.583531	0.843177	0.583531	0.583531	0.723056	0.646313
X_3325	0.500000	0.000000	0.448529	0.375000	0.448529	0.375000	0.448529	0.448529	0.500000	0.750000
X_3326	0.072881	0.877966	0.470894	0.949153	0.470894	0.898305	0.470894	0.470894	0.884746	0.057288

3328 rows × 1132 columns

3. Feature Engineering

중복 값인 열 삭제하기

002. train_x 중복 행 삭제하기

```
train_xtd = train_xt.drop_duplicates(keep = False)
train_xtd
```

	0	1	2	3	4	5	6	7	8	9
LINE	4.000000	5.000000	5.000000	5.000000	5.000000	5.000000	5.000000	4.000000	5.000000	4.000000
PRODUCT_CODE	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000
X_1	0.006536	0.006536	0.006536	0.006536	0.006536	0.006536	0.006536	0.000000	0.013072	0.307190
X_2	0.533333	0.600000	0.533333	0.000000	0.533333	0.533333	0.533333	0.666667	0.400000	0.666667
X_4	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
...	...	...	...	...	...	...	...	...	...	...
X_3322	0.068966	0.758621	0.463106	0.844828	0.463106	0.720690	0.463106	0.463106	0.693103	0.310345
X_3323	0.374749	0.174349	0.330111	0.468938	0.330111	0.571142	0.330111	0.330111	0.232465	0.995992
X_3324	0.382716	0.703036	0.583531	0.759760	0.583531	0.843177	0.583531	0.583531	0.723056	0.646313
X_3325	0.500000	0.000000	0.448529	0.375000	0.448529	0.375000	0.448529	0.448529	0.500000	0.750000
X_3326	0.072881	0.877966	0.470894	0.949153	0.470894	0.898305	0.470894	0.470894	0.884746	0.057288

1381 rows × 1132 columns

3. Feature Engineering

중복 값인 열 삭제하기

003. 다시 train_x 행, 열 바꾸기

```
train_xtt = train_xtd.transpose()
train_xtt
```

	LINE	PRODUCT_CODE	X_1	X_2	X_4	X_5	X_7	X_8	X_9	X_11
0	4.0	2.0	0.006536	0.533333	0.000000	0.000000	0.000000	0.000000	0.0000	0.530547
1	5.0	2.0	0.006536	0.600000	0.000000	0.000000	0.363636	0.000000	1.0000	0.915327
2	5.0	2.0	0.006536	0.533333	0.000000	0.000000	0.681818	0.000000	1.0000	0.735263
3	5.0	2.0	0.006536	0.000000	0.000000	0.000000	0.363636	0.000000	1.0000	0.482315
4	5.0	2.0	0.006536	0.533333	0.000000	0.000000	0.272727	0.000000	1.0000	0.798499
...	...	...	...	...	...	...	...	...	...	...
1127	2.0	0.0	0.011473	0.530231	0.005967	0.335322	0.163051	0.027446	0.5179	0.659740
1128	4.0	2.0	0.006536	1.000000	0.000000	1.000000	0.000000	0.000000	0.0000	0.918542
1129	5.0	2.0	0.000000	0.066667	0.000000	0.000000	0.272727	0.000000	1.0000	0.336549
1130	1.0	0.0	0.011473	0.530231	0.005967	0.335322	0.163051	0.027446	0.5179	0.659740
1131	4.0	2.0	0.006536	0.933333	0.000000	1.000000	0.000000	0.000000	0.0000	0.554126

1132 rows × 1381 columns

3. Feature Engineering

중복 값인 열 삭제하기

004. train columns 기준으로 test column 추출

```
## train의 기존 columns을 리스트 변수로 설정
train_col_origin = train.columns[train.columns.str.contains('X')].tolist()

## 중복 열이 제거된 train의 남은 columns을 리스트 변수로 설정
train_col = train_xtt.columns[train_xtt.columns.str.contains('X')].tolist()

## train 열 중 삭제 된 columns 구하기
train_col_main = [x for x in train_col_origin if x not in train_col]
```

```
## 삭제된 columns을 test에서도 삭제하기
for col in train_col_main:
  test_x = test_x.drop(columns = col)
```

4. Modeling

```
train_xtt.shape, train_y.shape, test_x.shape
```

```
((1132, 1381), (1132,), (535, 1381))
```

```
et_cls = ExtraTreesClassifier(n_estimators = 500, min_samples_leaf = 4,
                              min_samples_split= 7, max_features=1381, random_state = 2045)
et_cls.fit(train_xtt, train_y)
```

```
ExtraTreesClassifier
ExtraTreesClassifier(max_features=1381, min_samples_leaf=4, min_samples_split=7,
                     n_estimators=500, random_state=2045)
```

5. 결과 도출

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

예측값 확인

```
pred
```

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


감사합니다