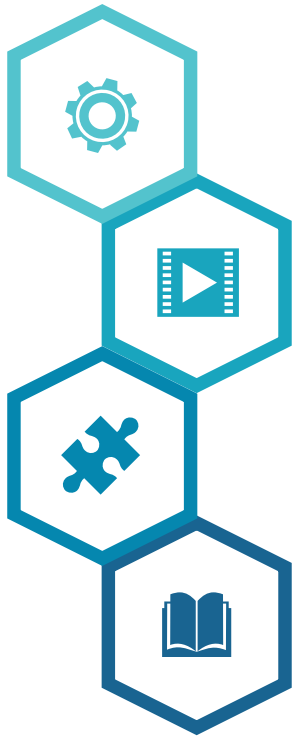
SCDC 2020

데이터 이미지화를 이용한 예측 모델

전체 개요

01 데이터 해석 및 시각화

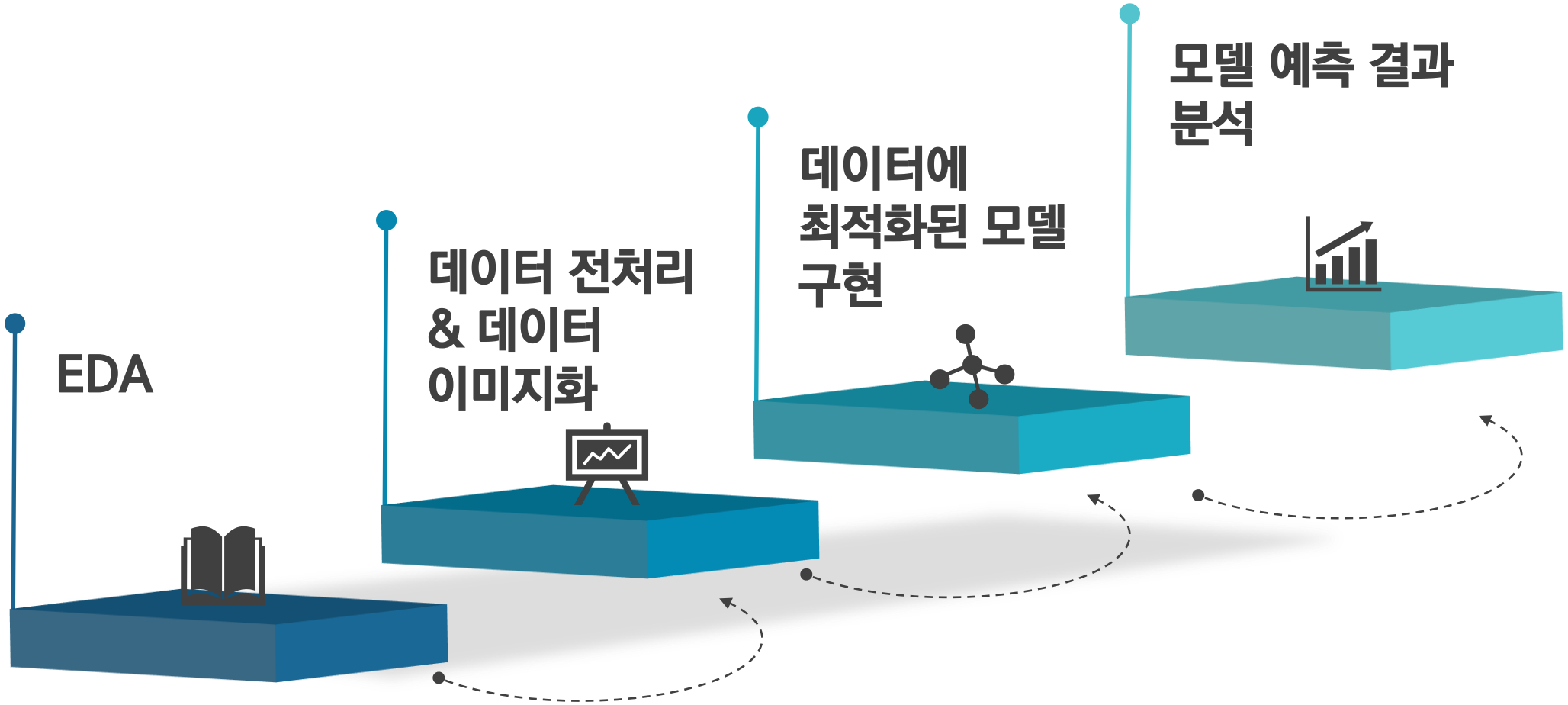


데이터 전처리 02

03 데이터 모델링 과정

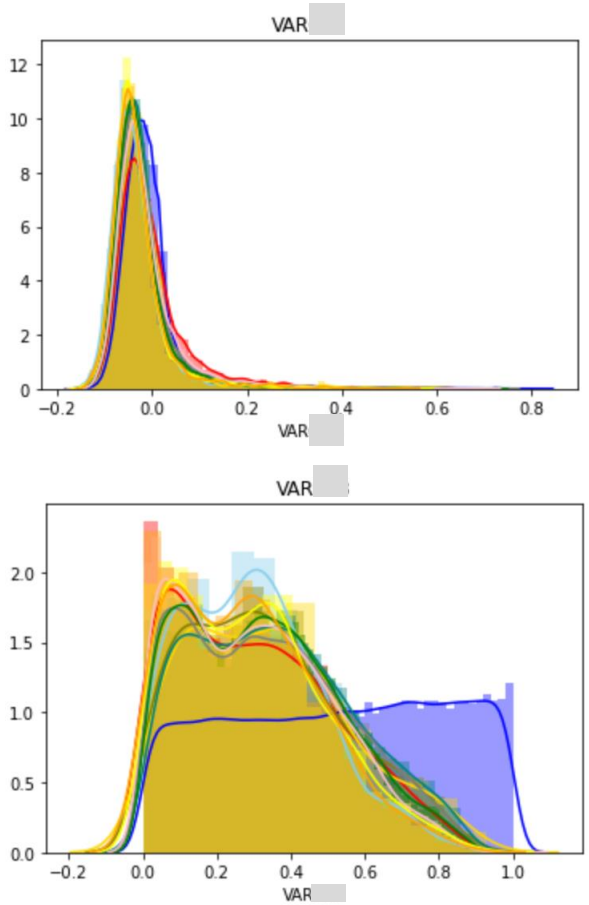
데이터 분석결과 04

데이터 모델링 전반 프로세스

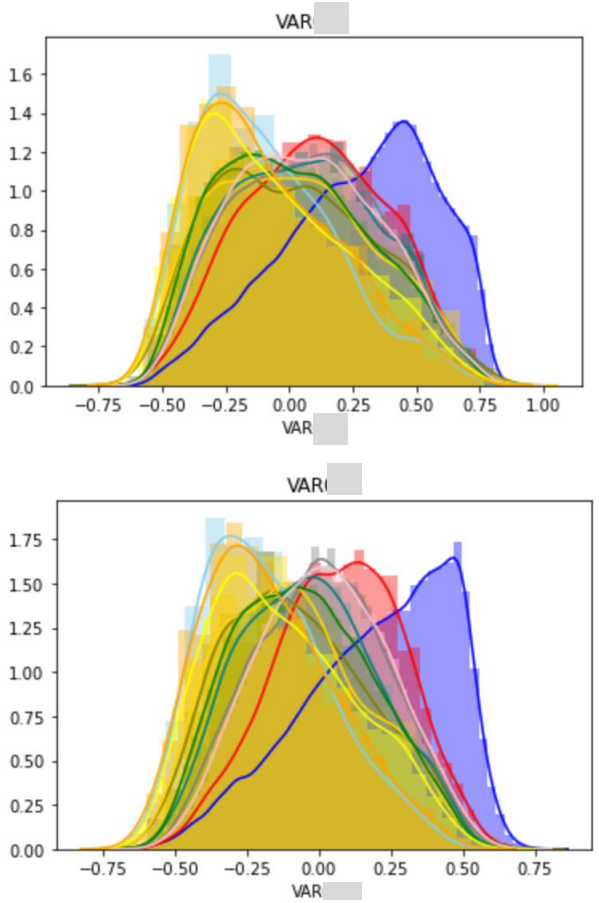


01. 데이터 해석 및 시각화

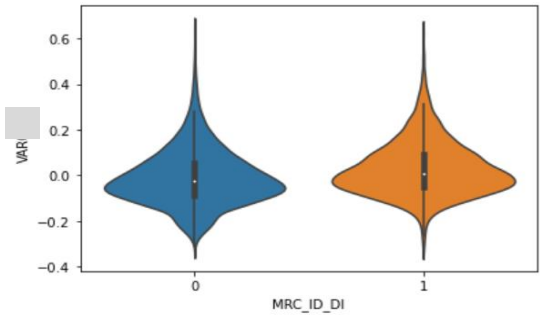
EDA를 통해, 변수들의 분포도 및 MRC_ID_DI와의 관련성 파악.



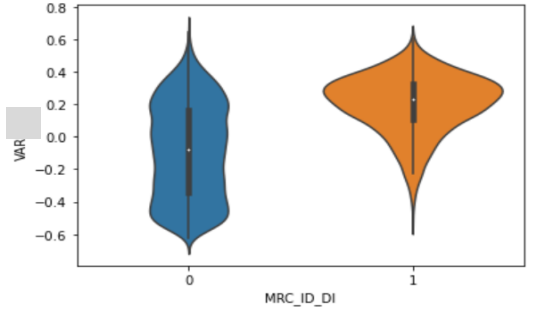
MRC_ID_DI 선정에 영향을 주지 않는 변수



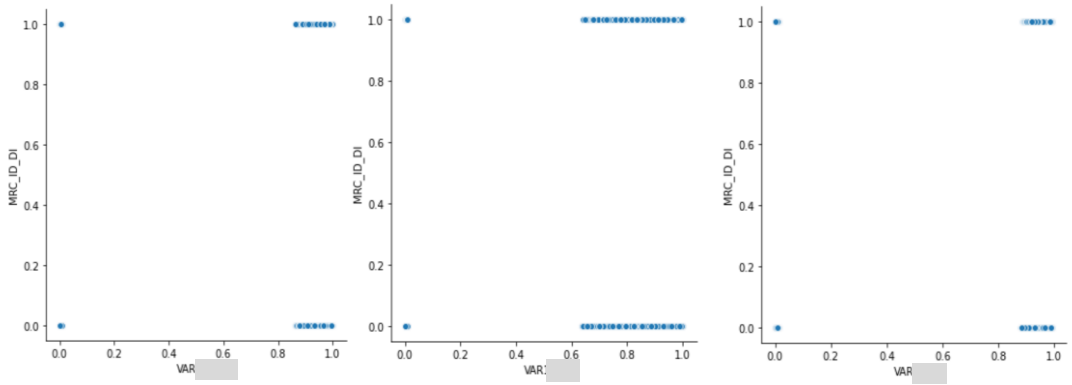
MRC_ID_DI 선정에 영향을 주는 변수



MRC_ID_DI 선정에 영향을 주지 않는 변수



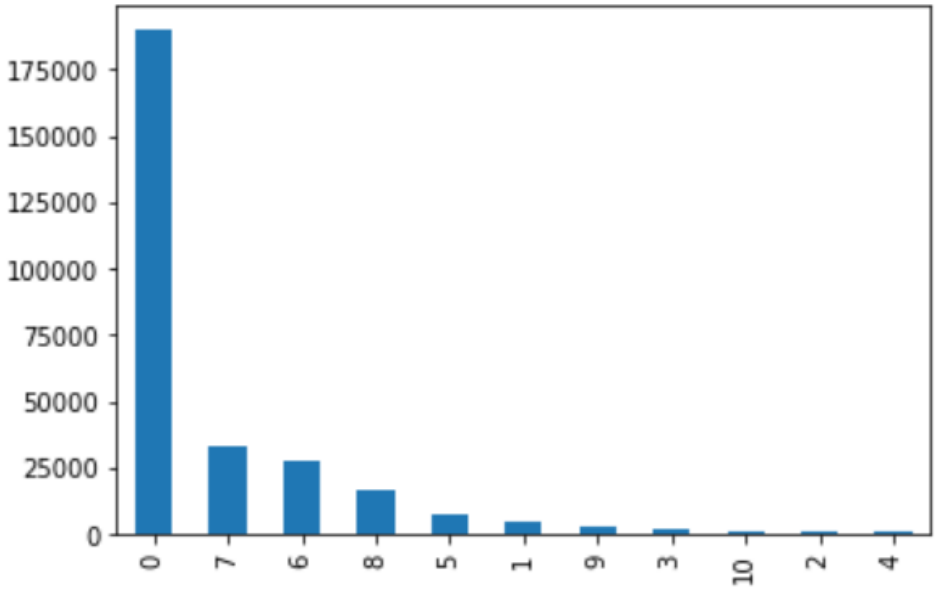
MRC_ID_DI 선정에 영향을 주는 변수



MRC_ID_DI 선정에 영향을 주지 않는 변수

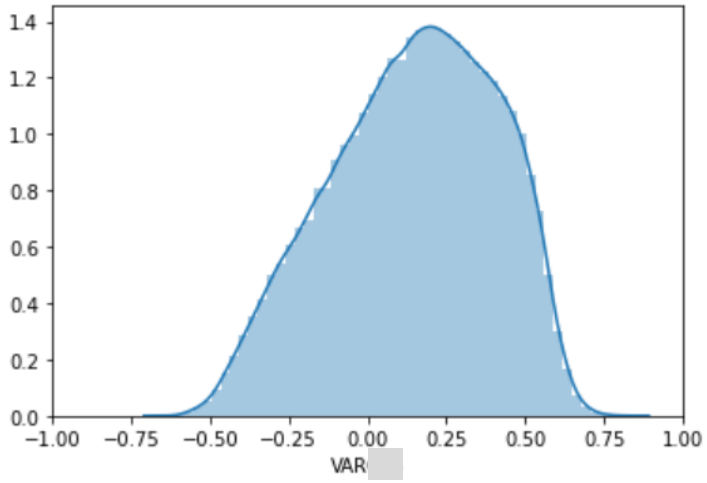
01. 데이터 해석 및 시각화

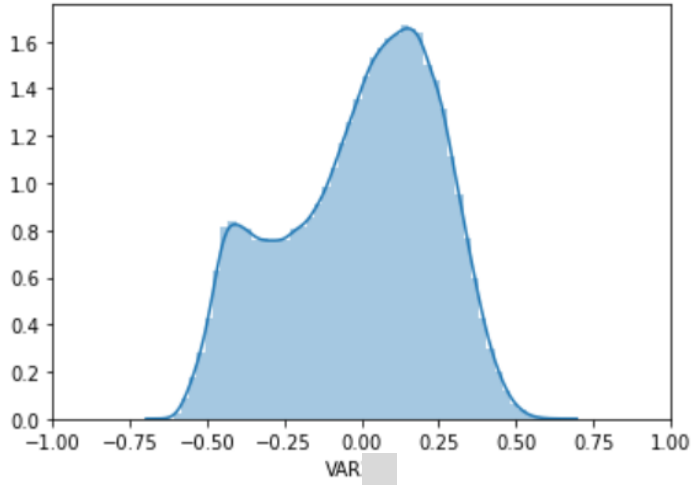
MRC_ID_DI별 데이터 수의 극심한 불균형



균형적인 학습을 위해
MRC_ID_DI별 데이터 수를 균등하게 조절

모든 변수가 -1 ~ 1 범위에서 벗어나지 않음





데이터(-1, 1)를 이미지화(0, 255) 할 수 있겠다는
아이디어 도출

02. 데이터 전처리

Outlier 처리

데이터 type이 numerical인 피처에서, 전체 분포의 99%보다 크거나 1%보다 작은 값을 가질 경우 50% 값으로 변경합니다.

```
for i in df.columns[0:-1]:
    if nc.loc[i, 'dType'] == 'numerical':
        d_90 = df[i].quantile(0.99)
        d_10 = df[i].quantile(0.01)
        d_50 = df[i].quantile(0.50)
        df[i] = np.where(df[i] > d_90, d_90, df[i])
        df[i] = np.where(df[i] < d_10, d_10, df[i])
```

02. 데이터 전처리

데이터 불균형 완화

Train 데이터의 각 MRC_ID_DI의 수를 동일하게 만들어줍니다.
Stratify를 이용해 Test 데이터 MRC_ID_DI 분포를 전체 데이터 MRC_ID_DI 분포와 일치하게 합니다.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.01, random_state=1, stratify = y)
```

```
train = X_train
train['cst_id_di'] = y_train.index
train = train.set_index('cst_id_di')
train['MRC_ID_DI'] = y_train

train_0 = train[train['MRC_ID_DI'] == 0].sample(frac=1)
train_1 = train[train['MRC_ID_DI'] == 1].sample(frac=1)
train_2 = train[train['MRC_ID_DI'] == 2].sample(frac=1)
train_3 = train[train['MRC_ID_DI'] == 3].sample(frac=1)
train_4 = train[train['MRC_ID_DI'] == 4].sample(frac=1)
train_5 = train[train['MRC_ID_DI'] == 5].sample(frac=1)
train_6 = train[train['MRC_ID_DI'] == 6].sample(frac=1)
train_7 = train[train['MRC_ID_DI'] == 7].sample(frac=1)
train_8 = train[train['MRC_ID_DI'] == 8].sample(frac=1)
train_9 = train[train['MRC_ID_DI'] == 9].sample(frac=1)
train_10 = train[train['MRC_ID_DI'] == 10].sample(frac=1)
```

해당 과정을 거침으로써
데이터 불균형 완화를 통한 acc 및 성능지표가 증가함을 확인

```
sample_size = min(len(train_0),len(train_1), len(train_2), len(train_3), len(train_4), len(train_5), len(train_6), len(train_7),
                  len(train_8), len(train_9), len(train_10))

train_f = pd.concat([train_0.head(sample_size), train_1.head(sample_size), train_2.head(sample_size), train_3.head(sample_size),
                     train_4.head(sample_size), train_5.head(sample_size), train_6.head(sample_size), train_7.head(sample_size),
                     train_8.head(sample_size), train_9.head(sample_size), train_10.head(sample_size)]).sample(frac=1)
```

03. 모델링 과정

1. 유효 변수 선택

Column수의 변화: 226 -> 90 -> 4005 -> 380

Step1. SelectKBest로 피처 선택 k=90
Step2. combinations, PolynomialFeatures로 피처 간의 조합 생성
Step3. SelectKBest로 피처 선택 k=380

```
import sklearn.feature_selection
select = sklearn.feature_selection.SelectKBest(k=90)
selected_features = select.fit(X_train, y_train)
indices_selected = selected_features.get_support(indices=True)
colnames_selected = [X_train.columns[i] for i in indices_selected]
```

```
from itertools import combinations
X_train = X_train.astype(np.float16)
combos = list(combinations(list(X_train.columns), 2))
colnames = list(X_train.columns) + ['_'.join(x) for x in combos]
```

```
from sklearn.preprocessing import PolynomialFeatures
poly = PolynomialFeatures(interaction_only=True, include_bias=False)
X_train = poly.fit_transform(X_train)
X_train = pd.DataFrame(X_train)
X_train.columns = colnames
```

03. 모델링 과정

2. 이미지 생성

데이터의 형태 (p,)를 이미지 (1,p,p,3)으로 변환

Step1. DataFrame의 한 row를
row_array = (p,1), col_array = (1,p)로 resize

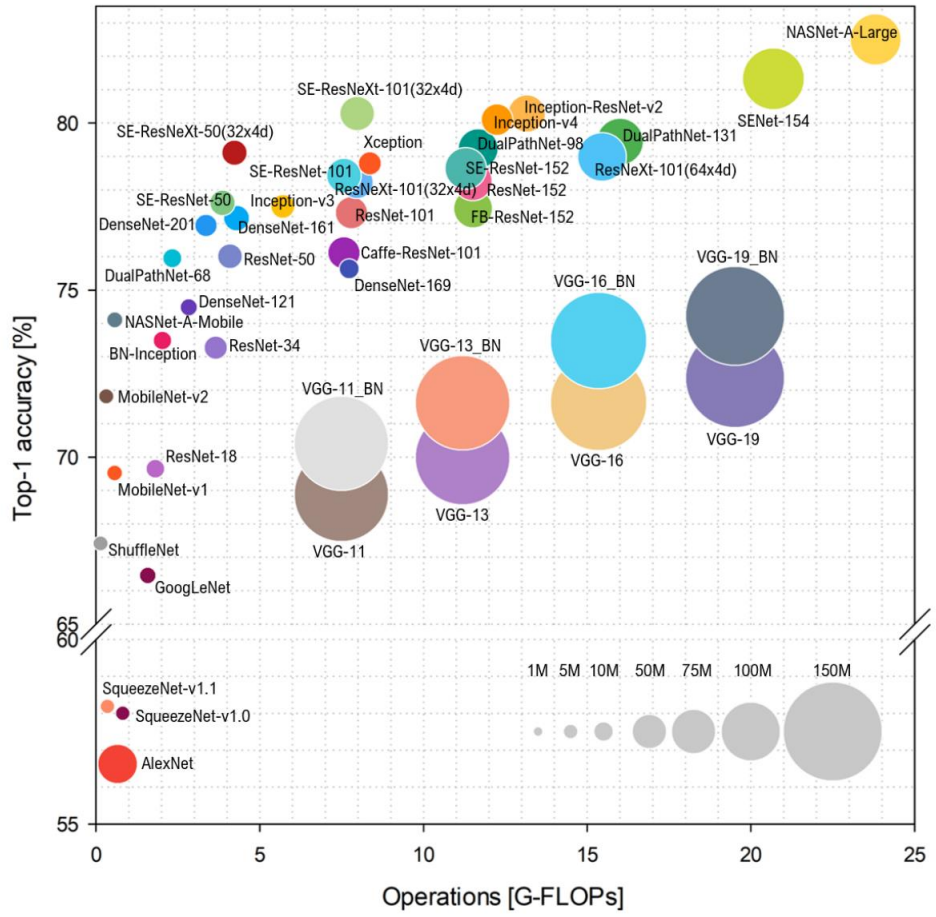
Step2. np.dot(row_array,col_array) 하여
(p,p)의 행렬을 생성
기존의 데이터 값이 -1 ~ 1 인 것을 고려하여,
np.dot 의 결과가 정규화된 이미지 값 0 ~ 1이
되기 위해 전체에 (+1) /2

Step3. np.repeat을 한 뒤 reshape,
np.expaned_dims 을 통해
(p,p) => (1,p,p,3)의 이미지 데이터 생성

```
X = X.replace("[", "").replace("]", "")
X = np.fromstring(X, sep = " ")
an_array = np.resize(X, (p, 1))
t_array = np.resize(X,(1, p))
final_array = (np.dot(an_array, t_array) +1) / 2
final_array = final_array.astype(np.float16)
final_array = np.repeat(final_array.flatten(), 3)
data = final_array.reshape((p,p, 3))
data = np.expand_dims(data, axis=0)
```

03. 모델링 과정

3. Transfer Learning 모델 후보군



효과적인 image classification을 위해 Transfer Learning을 결정하였고, 그러기 위해서 Transfer Learning할 모델 선정이 필요했습니다.
왼쪽 그림표는 여러 모델들을 하나의 표로 나타낸 것으로, 원의 크기는 학습 데이터 셋의 크기, x축은 연산의 복잡도 y축은 정확도를 나타냅니다. 저희는 해당 그림표에서 데이터의 크기, 컴퓨팅 파워 등을 고려하여 몇 개의 모델 후보 군을 추려냈습니다.

1) Xception

2) InceptionV3

3) MobileNetV2

4) NASNetLarge

03. 모델링 과정

4. Transfer Learning을 위한 Base 모델 선정

Model	Size	Top-1 Acc	Top-5 Acc	Parameters	Depth	ACC 비교 (%)
Xception	88MB	0.790	0.945	22,910,480	126	4.73 % ↑
InceptionV3	92MB	0.779	0.937	23,851,784	159	1.2 % ↑
MobileNetV2	14MB	0.713	0.901	3,538,984	88	3.34 % ↑
NASNetLarge	23MB	0.825	0.960	88,949,818	-	-

각 모델의 특징을 표로 작성한 것입니다.
ACC비교는, 동일한 환경에서 SCDC2020 데이터로 각 모델을 학습시켰을 때의 accuracy를 비교한 것입니다.
제일 마지막 열에 있는 정보가 SCDC2020 데이터 셋으로 실행을 했을 때의 ACC 변화입니다.(NASNetLarge 기준)
이와 같은 결과와 모델의 경량화를 고려하여 Xception을 선택하였습니다.

03. 모델링 과정

5. 일반 데이터로 DNN vs 이미지화 데이터로 CNN

	LIFT 비교 (%)	AUROC 비교 (%)	ACC 비교 (%)
이미지화 데이터 CNN	14.54% ↑	2.96% ↑	8.01% ↑
일반 데이터 DNN	-	-	-

비교 결과 이미지화 데이터로 CNN모델을 구현한 경우가
일반 데이터로 DNN모델을 구현한 경우보다 효과적임을 확인했습니다.

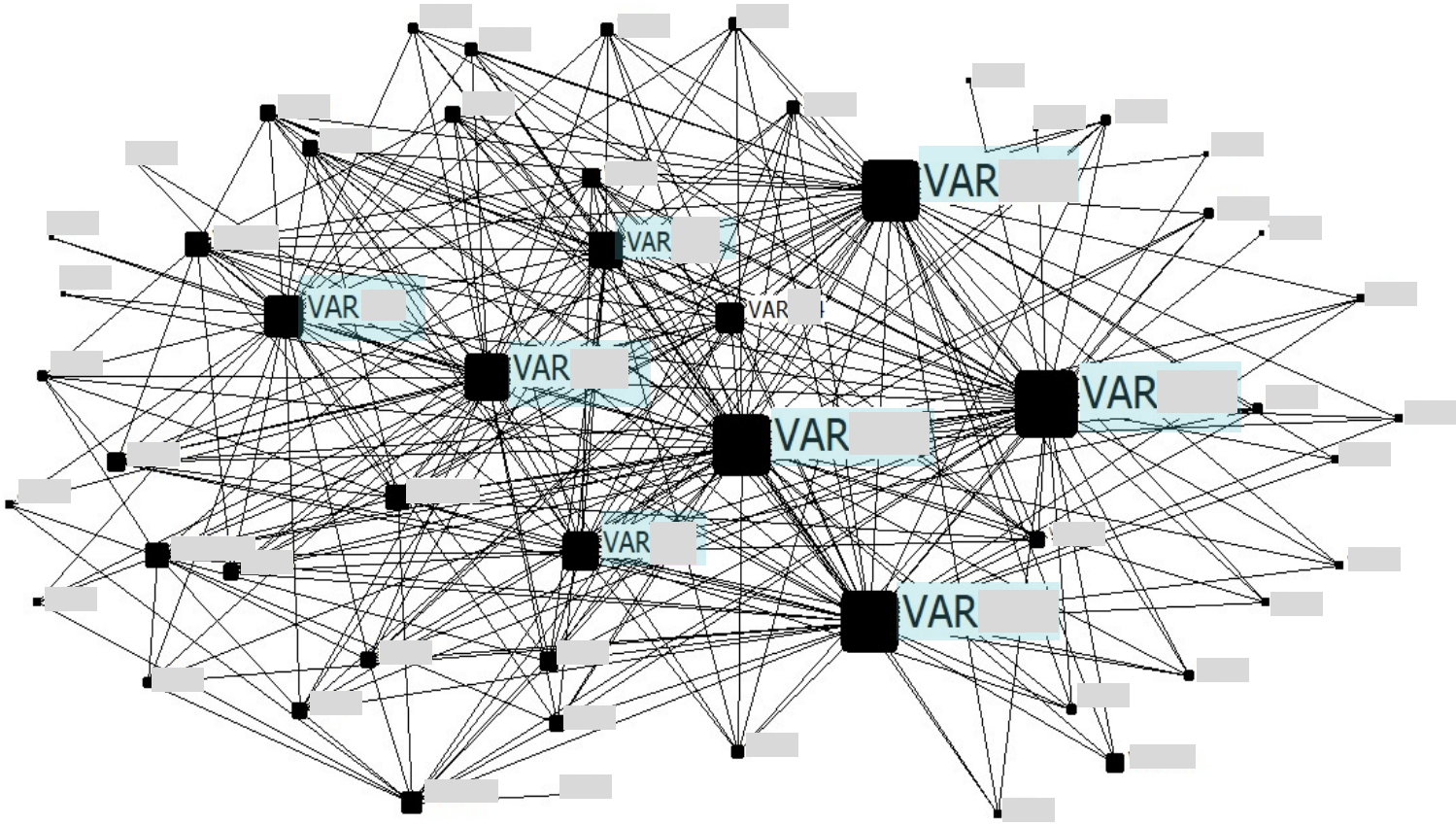
04. 데이터 분석결과

조합변수 피처 간의 관계와 빈도수를 따져봤을 때

VAR , VAR , VAR , VAR ,
VAR , VAR , VAR , VAR

에 해당하는 피처가 예측에 영향력이 큼을 확인했습니다.

이 피처들을 마케팅에 활용한다면 온라인 가맹점의 이용률을 높일 수 있을 것입니다.



감사합니다.

김지은, 조유미, 추유진