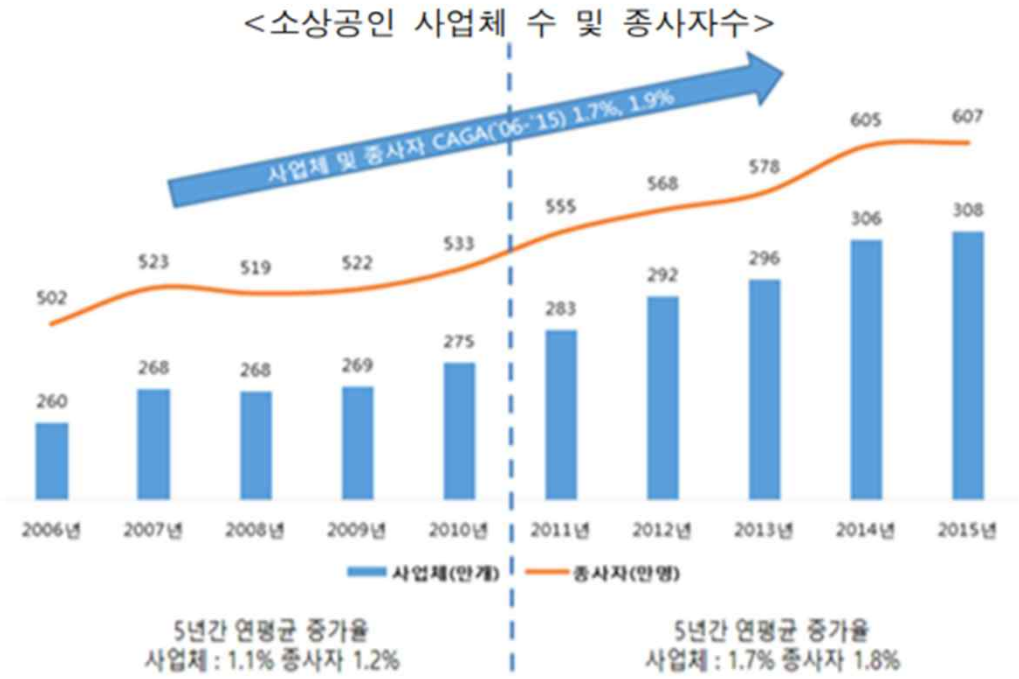
# 소상공인 신용평가 알고리즘

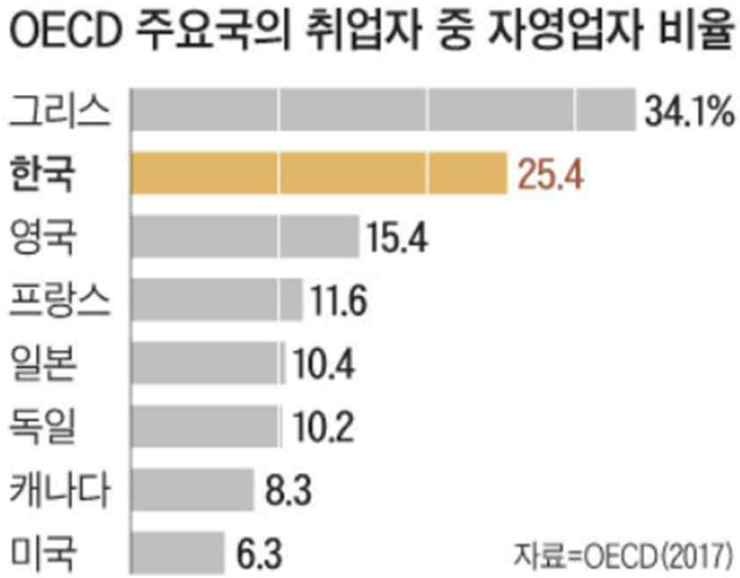
부산시 고객만족 상권분석을 활용하여

# 목차

- 문제 인식
- 프로젝트의 목표
- 대상 선정
- 벤치마킹
- 상권분석
- 변수선정
- 모델링
- 서비스 제공 – Web Page

# 1. 문제 인식



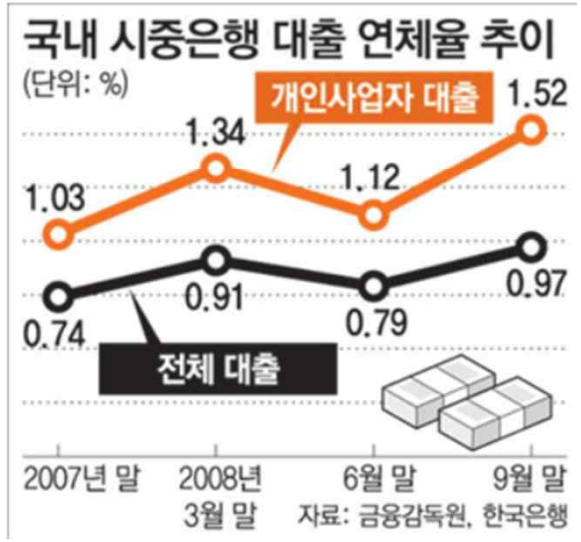

- 우리나라의 자영업자는 증가 추세

# 1. 문제 인식

뉴스 > 사회 > 복지

## [단독]高신용 2등급도 대출 퇴짜... 벼랑끝 소상공인에 여전한 은행 문턱

김동혁 기자, 장윤정 기자, 세종=송충현 기자 입력 2020-04-08 03:00 수정 2020-04-08 10:09




- 신용등급이 높아도 대출이 어려운 현실
- 개인사업자 대출의 경우,

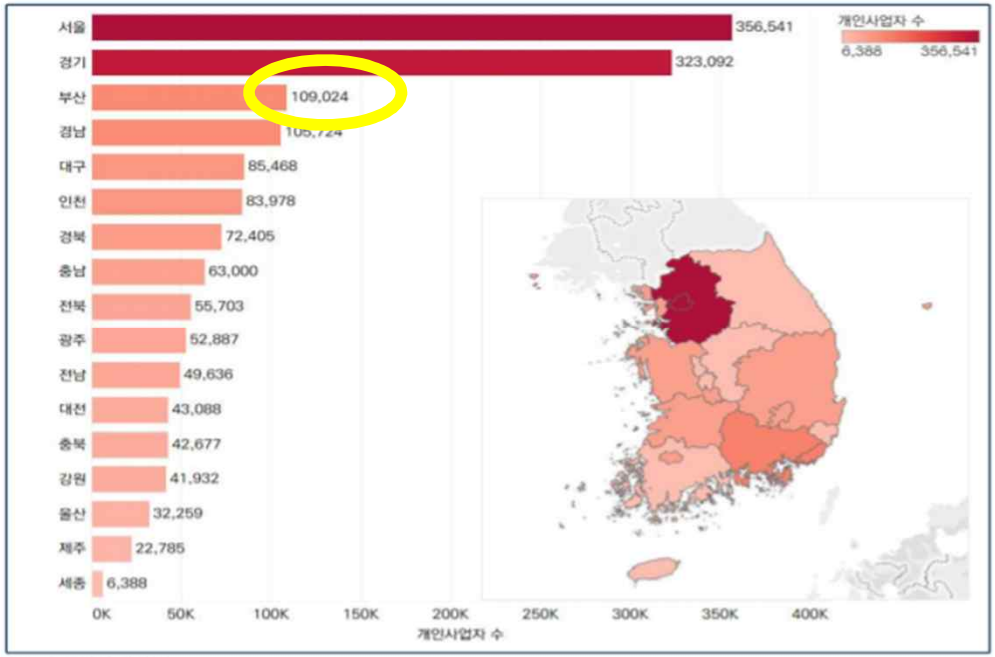
  일반 대출보다 연체율이 높음

# 2. 프로젝트의 목표

- **재무정보, 비재무정보**를 통해 **ML로 신용평가를 수행**할 수 있다.
- **고객경험(UX)**을 활용해 **상권분석**을 수행한다.
- 사용자에게 신용평가요소에 영형을 끼친 **변수를 설명**해주는 서비스 개발

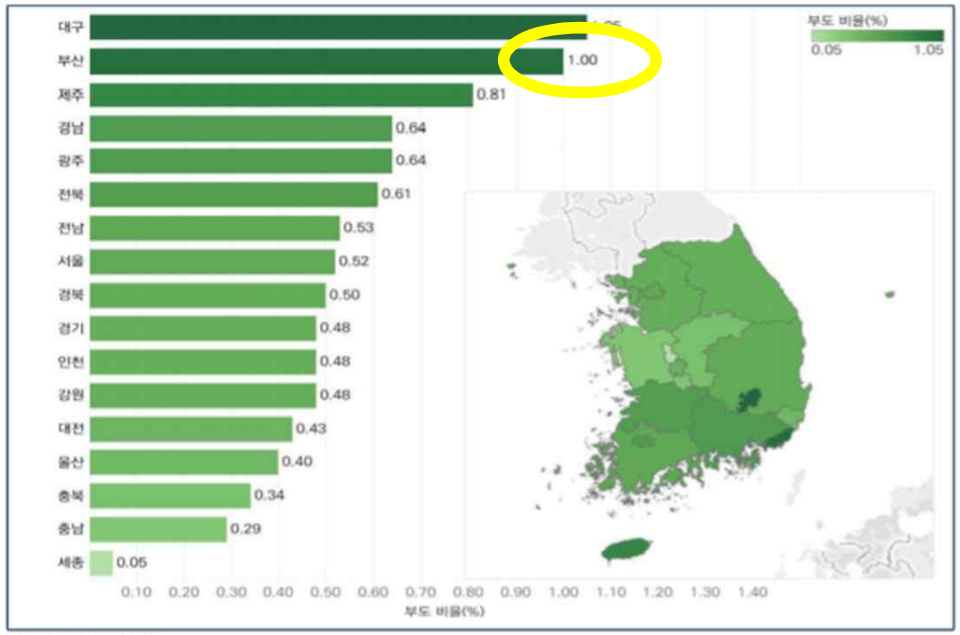
# 3. 대상 선정

[전국 17개 시도별 개인사업자 수]



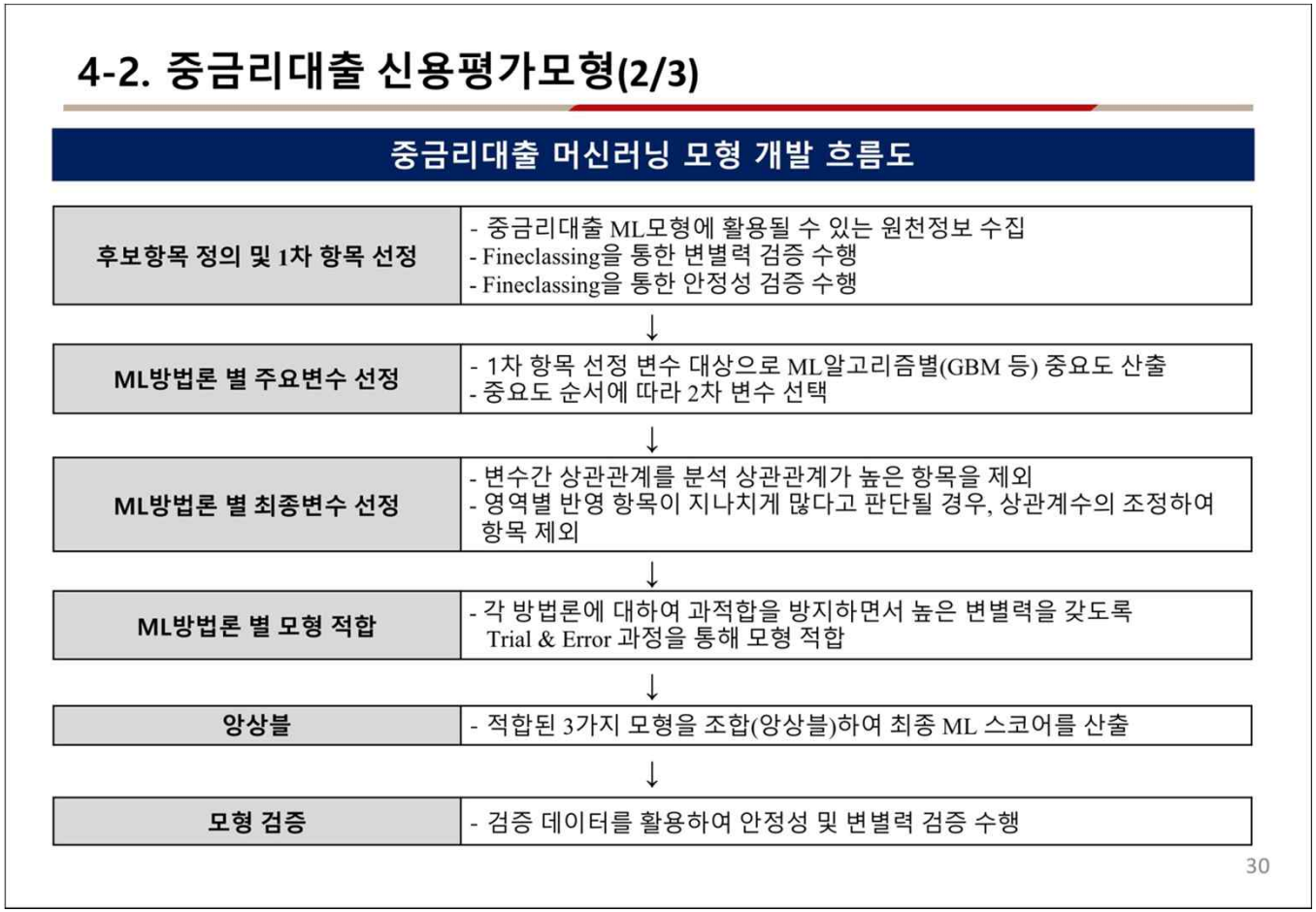

자료: 한국신용정보원
주: 사업자등록번호 기준 개인사업자 수

[전국 17개 시도별 개인사업자의 부도 비율 분포]




자료: 한국신용정보원

# 4. 벤치마킹



## 4-2. 중금리대출 신용평가모형(2/3)

**중금리대출 머신러닝 모형 개발 흐름도**

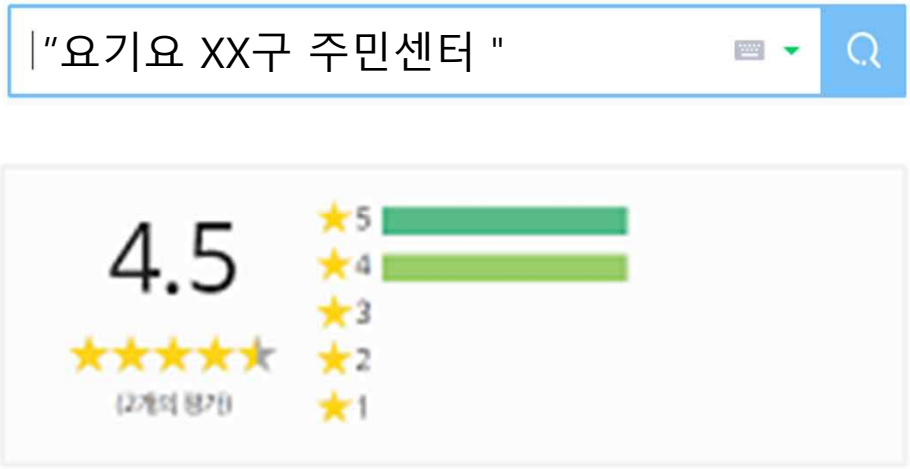
| | |
|---|---|
| **후보항목 정의 및 1차 항목 선정** | - 중금리대출 ML모형에 활용될 수 있는 원천정보 수집<br>- Fineclassing을 통한 변별력 검증 수행<br>- Fineclassing을 통한 안정성 검증 수행 |
| ↓ | |
| **ML방법론 별 주요변수 선정** | - 1차 항목 선정 변수 대상으로 ML알고리즘별(GBM 등) 중요도 산출<br>- 중요도 순서에 따라 2차 변수 선택 |
| ↓ | |
| **ML방법론 별 최종변수 선정** | - 변수간 상관관계를 분석 상관관계가 높은 항목을 제외<br>- 영역별 반영 항목이 지나치게 많다고 판단될 경우, 상관계수의 조정하여 항목 제외 |
| ↓ | |
| **ML방법론 별 모형 적합** | - 각 방법론에 대하여 과적합을 방지하면서 높은 변별력을 갖도록 Trial & Error 과정을 통해 모형 적합 |
| ↓ | |
| **앙상블** | - 적합된 3가지 모형을 조합(앙상블)하여 최종 ML 스코어를 산출 |
| ↓ | |
| **모형 검증** | - 검증 데이터를 활용하여 안정성 및 변별력 검증 수행 |

30

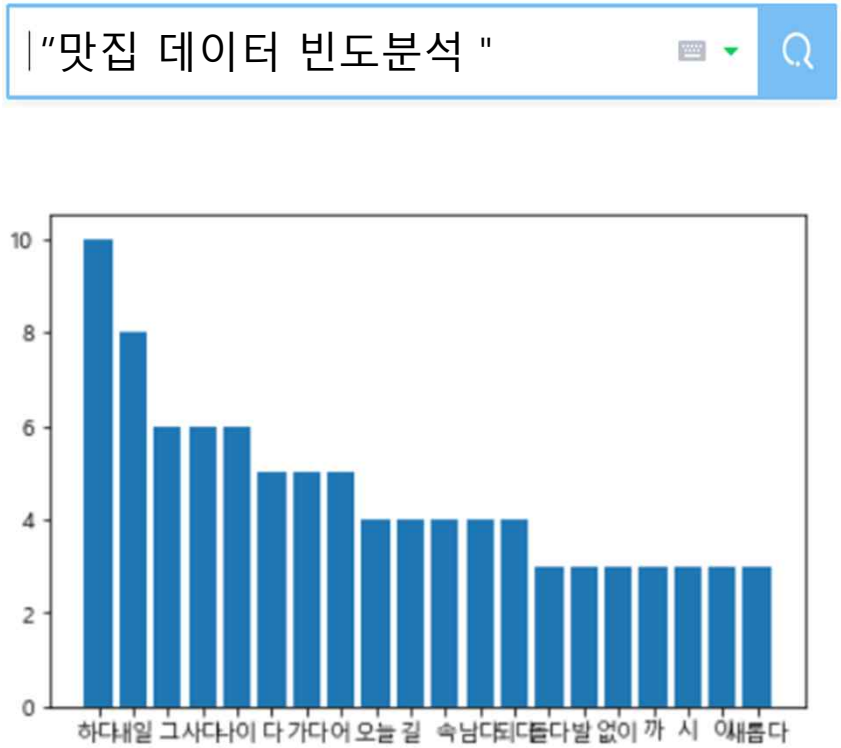
## 앙상블

- ✓ Logistic Regression
- ✓ DNN
- ✓ XGBoost

# 5. 상권분석 – 요기요 별점 크롤링



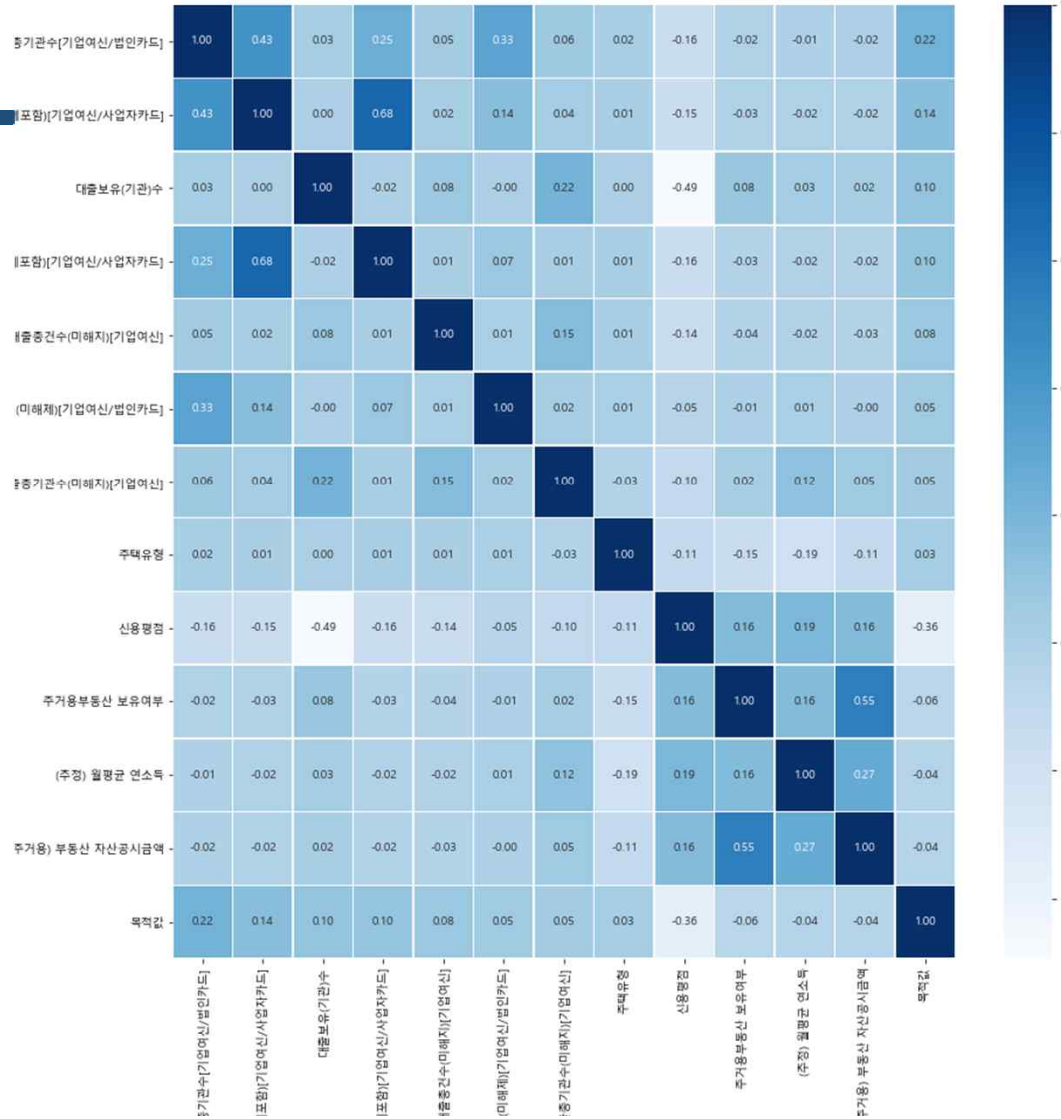
# 5. 상권분석 – 빈도분석



# 6. 변수선정

'미해제연체총기관수[기업여신/법인카드]'
'총연체일수(1년내유지)(해제포함)[기업여신/사업자카드]'
'대출보유(기관)수'
'최장연체일수(해제포함)[기업여신/사업자카드]'
'저축은행업종대출총건수(미해지)[기업여신]'
'총연체금액(미해제)[기업여신/법인카드]'
'대출총기관수(미해지)[기업여신]'
'주택유형'
'신용평점'
'주거용부동산 보유여부'
'(추정) 월평균 연소득'
'(주거용) 부동산 자산공시금액'
'목적값'

12 개 독립변수 투입
1개 종속변수 투입 (목적값)



# 6. 변수선정

```
목적값                                          1.000000
미해제연체총기관수[기업여신/법인카드]                          0.195679
총연체일수(1년내유지)(해제포함)[기업여신/사업자카드]      0.124319
대출보유(기관)수                                0.097102
최장연체일수(해제포함)[기업여신/사업자카드]              0.085217
저축은행업종대출총건수(미해지)[기업여신]                  0.072511
총연체금액(미해제)[기업여신/법인카드]                    0.046244
대출총기관수(미해지)[기업여신]                        0.042266
주택유형                                        0.031773
(주거용) 부동산 자산공시금액                        -0.038346
(추정) 월평균 연소득                             -0.041093
주거용부동산 보유여부                              -0.052591
신용평점                                        -0.331231
Name: 목적값, dtype: float64
```

# 7. 모델링 – Linear Regression

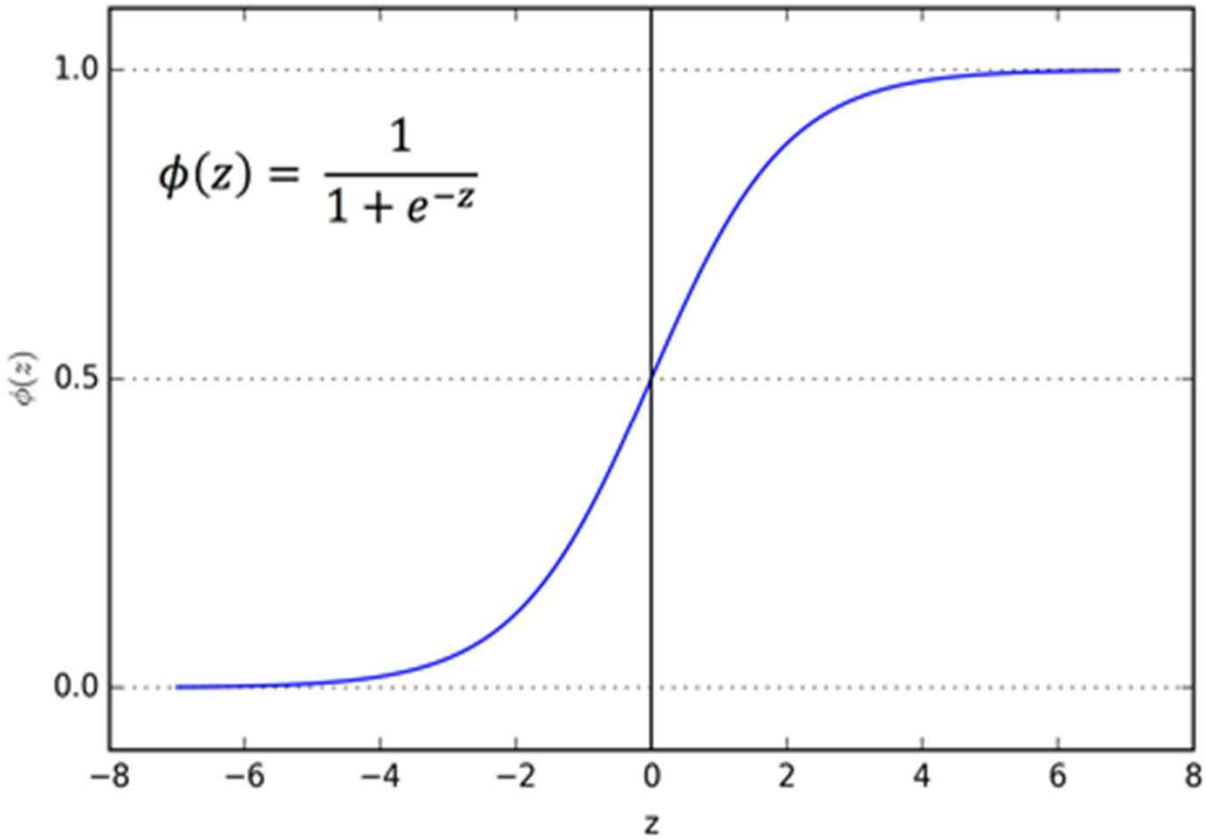
In [44]:
```
from sklearn.linear_model import LinearRegression

mlr = LinearRegression()
mlr.fit(X_train, y_train)
y_predict = mlr.predict(X_test)
```

In [45]:
```

print("훈련 세트 점수: {:.2f}".format(mlr.score(X_train, y_train)))
print("테스트 세트 점수: {:.2f}".format(mlr.score(X_test, y_test)))
```

```
훈련 세트 점수: 0.16
테스트 세트 점수: 0.15
```

선형회귀 모델은 적당하지 않음.
하지만 이 경험으로 모델 선택의 중요성에 대해서 알게 됨 .

# 7. 모델링 – Logistic Regression




‘분류’에 적합한 모델

‘로지스틱 회귀’

Binary step

# 7. 모델링 – Logistic Regression

```
# 데이터 변경란

# 열 변경
# data_1 = data_.drop('', '', axis=1)
# data_1.head(2)

# 행 변경
# data_ = data[data['도시'] == '부산광역시']

# 1 과 2를 합침
data['목적값'] = data['목적값'].replace(2,1)

# 데이터 추출

data_1 = data[['미해제연체총기관수[기업여신/법인카드]','총연체일수(1년내유지)(해제포함)[기업여신/사업자카드]',
               '대출보유(기관)수','최장연체일수(해제포함)[기업여신/사업자카드]','저축은행업종대출총건수(미해지)[기업여신]',
                '총연체금액(미해제)[기업여신/법인카드]','대출총기관수(미해지)[기업여신]','주택유형',
               '신용평점','주거용부동산 보유여부','(추정) 월평균 연소득','(주거용) 부동산 자산공시금액','목적값']]
```

보다 정확한 분류를 위하여 종속변수 '목적값'의 '2'와 '1' 을 합침

In [122]:

```
# data_1 에서 특성 표준화 _ 안할거면 취소하기

from sklearn.preprocessing import StandardScaler
std_scaler = StandardScaler()
data_1.head()

fitted = std_scaler.fit(X)
output = std_scaler.transform(X)
data_a = pd.DataFrame(output, columns=X.columns, index=list(X.index.values))
data_a.describe()

```

Out[122]:

| | 미해제연체총기관수[기업여신/법인카드] | 총연체일수(1년내유지)(해제포함)[기업여신/사업자카드] | 대출보유(기관)수 | 최장연체일수(해제포함)[기업여신/사업자카드] | 저축은행업종대출총건수(미해지)[기업여신] | 총연체금액(미해제)[기업여신/법인카드] | 대출총기관수(미해지)[기업여신] | 주택유형 | 신용평점 | 주거용부동산보유여부 | ( |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **count** | 6.482500e+04 | 6.482500e+04 | 6.482500e+04 | 6.482500e+04 | 6.482500e+04 | 6.482500e+04 | 6.482500e+04 | 6.482500e+04 | 6.482500e+04 | 6.482500e+04 | 6. |
| **mean** | -7.015000e-18 | -2.630625e-18 | -1.622219e-17 | -5.918906e-18 | -1.600297e-17 | -1.534531e-18 | -2.334680e-17 | 9.448328e-17 | 8.659140e-18 | -4.395336e-17 | -2 |
| **std** | 1.000008e+00 | 1.000008e+00 | 1.000008e+00 | 1.000008e+00 | 1.000008e+00 | 1.000008e+00 | 1.000008e+00 | 1.000008e+00 | 1.000008e+00 | 1.000008e+00 | 1. |
| **min** | -4.673901e-02 | -5.053492e-02 | -1.020249e+00 | -6.027057e-02 | -9.720668e-02 | -1.207952e-02 | -6.895977e-01 | -1.077119e+00 | -6.012434e+00 | -8.901572e-01 | -1. |
| **25%** | -4.673901e-02 | -5.053492e-02 | -1.020249e+00 | -6.027057e-02 | -9.720668e-02 | -1.207952e-02 | -6.895977e-01 | -1.077119e+00 | -5.597641e-01 | -8.901572e-01 | -4 |
| **50%** | -4.673901e-02 | -5.053492e-02 | -2.744228e-01 | -6.027057e-02 | -9.720668e-02 | -1.207952e-02 | -6.895977e-01 | 9.284026e-01 | 2.908525e-01 | -8.901572e-01 | -3 |
| **75%** | -4.673901e-02 | -5.053492e-02 | 4.714038e-01 | -6.027057e-02 | -9.720668e-02 | -1.207952e-02 | 8.284915e-01 | 9.284026e-01 | 7.488768e-01 | 1.123397e+00 | 7 |
| **max** | 6.617963e+01 | 6.941727e+01 | 5.692190e+00 | 6.192185e+01 | 3.900159e+01 | 1.901550e+02 | 8.418937e+00 | 9.284026e-01 | 1.257793e+00 | 1.123397e+00 | 5. |

In [123]:

```
# 정규화하기

import numpy as np
from sklearn.preprocessing import Normalizer

# 변환기 객체를 만듭니다
normalizer = Normalizer(norm="l2")

normalizer.transform(data_a)
data_x = pd.DataFrame(data_a, columns=X.columns, index=list(X.index.values))
data_x.describe()

```

| | 기관수[기업여신/법인카드] | 제포함)[기업여신/사업자카드] | 수 | 업여신/사업자카드] | 해지)[기업여신] | 해제)[기업여신/법인카드] | (미해지)[기업여신] | 주택유형 | 신용평점 | 보유여부 |
|---|---|---|---|---|---|---|---|---|---|---|
| **count** | 6.482500e+04 | 6.482500e+04 | 6.482500e+04 | 6.482500e+04 | 6.482500e+04 | 6.482500e+04 | 6.482500e+04 | 6.482500e+04 | 6.482500e+04 | 6.482500e+04 |
| **mean** | -7.015000e-18 | -2.630625e-18 | -1.622219e-17 | -5.918906e-18 | -1.600297e-17 | -1.534531e-18 | -2.334680e-17 | 9.448328e-17 | 8.659140e-18 | -4.395336e-17 |
| **std** | 1.000008e+00 | 1.000008e+00 | 1.000008e+00 | 1.000008e+00 | 1.000008e+00 | 1.000008e+00 | 1.000008e+00 | 1.000008e+00 | 1.000008e+00 | 1.000008e+00 |
| **min** | -4.673901e-02 | -5.053492e-02 | -1.020249e+00 | -6.027057e-02 | -9.720668e-02 | -1.207952e-02 | -6.895977e-01 | -1.077119e+00 | -6.012434e+00 | -8.901572e-01 |
| **25%** | -4.673901e-02 | -5.053492e-02 | -1.020249e+00 | -6.027057e-02 | -9.720668e-02 | -1.207952e-02 | -6.895977e-01 | -1.077119e+00 | -5.597641e-01 | -8.901572e-01 |
| **50%** | -4.673901e-02 | -5.053492e-02 | -2.744228e-01 | -6.027057e-02 | -9.720668e-02 | -1.207952e-02 | -6.895977e-01 | 9.284026e-01 | 2.908525e-01 | -8.901572e-01 |
| **75%** | -4.673901e-02 | -5.053492e-02 | 4.714038e-01 | -6.027057e-02 | -9.720668e-02 | -1.207952e-02 | 8.284915e-01 | 9.284026e-01 | 7.488768e-01 | 1.123397e+00 |
| **max** | 6.617963e+01 | 6.941727e+01 | 5.692190e+00 | 6.192185e+01 | 3.900159e+01 | 1.901550e+02 | 8.418937e+00 | 9.284026e-01 | 1.257793e+00 | 1.123397e+00 |

# 7. 모델링 – Logistic Regression

In [36]:

```
from sklearn.linear_model import LogisticRegression
import mglearn
logreg = LogisticRegression().fit(X_train, y_train)
print("훈련 세트 점수: {:.3f}".format(logreg.score(X_train, y_train)))
print("테스트 세트 점수: {:.3f}".format(logreg.score(X_test, y_test)))

from sklearn.metrics import confusion_matrix


prediction =  logreg.predict(X_test)
confusion = confusion_matrix(y_true=y_test, y_pred=prediction)
print("로지스틱 회귀")
print(confusion)

```

```
C:₩Users₩a₩anaconda3₩lib₩site-packages₩sklearn₩utils₩validation.py:73: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  return f(**kwargs)
훈련 세트 점수: 0.978
테스트 세트 점수: 0.979
로지스틱 회귀
[[15860     7]
 [  329    11]]
```

[전국 단위 로지스틱 회귀 분석결과]

정규화/표본화 하기 전

In [44]:

```
from sklearn.linear_model import LogisticRegression
import mglearn
logreg = LogisticRegression().fit(X_train, y_train)
print("훈련 세트 점수: {:.3f}".format(logreg.score(X_train, y_train)))
print("테스트 세트 점수: {:.3f}".format(logreg.score(X_test, y_test)))

from sklearn.metrics import confusion_matrix


prediction =  logreg.predict(X_test)
confusion = confusion_matrix(y_true=y_test, y_pred=prediction)
print("로지스틱 회귀")
print(confusion)
```

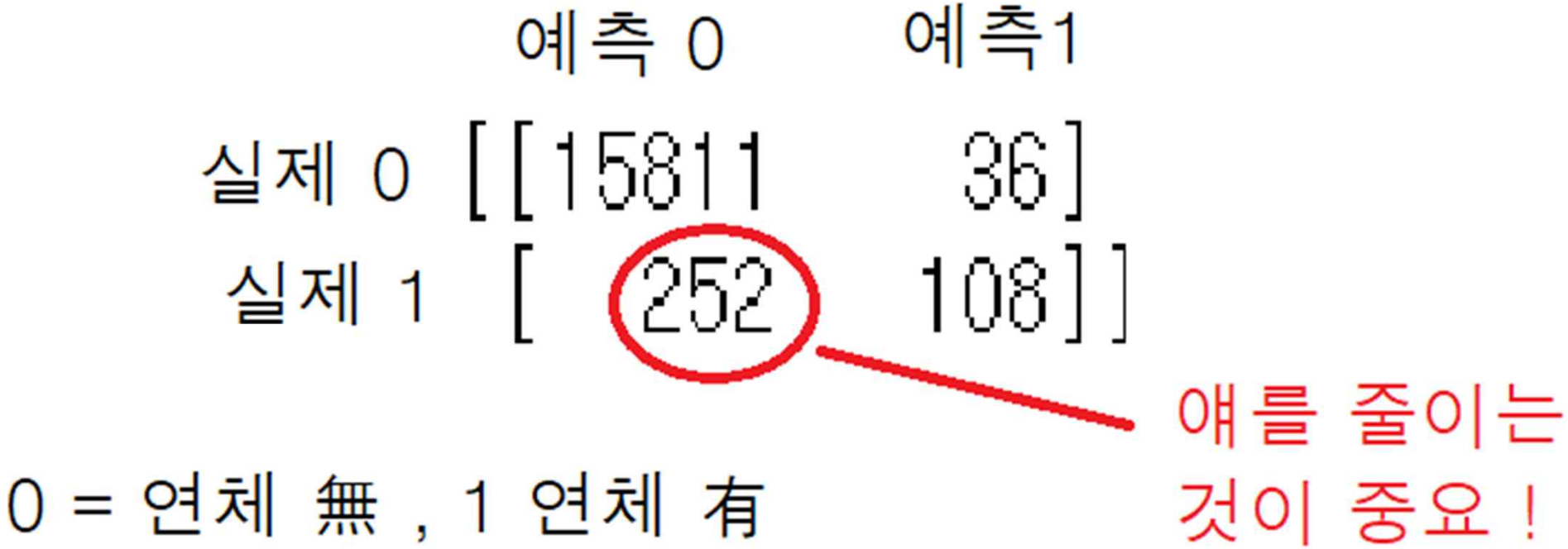
```
훈련 세트 점수: 0.981
테스트 세트 점수: 0.982
로지스틱 회귀
[[15811    36]
 [  252   108]]
C:₩Users₩a₩anaconda3₩lib₩site-packages₩sklearn₩utils₩validation.py:73: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  return f(**kwargs)
```

정규화/표본화 한 후

정규화/표본화의 중요성에 대해 알게됨

# 7. 모델링 – Logistic Regression

아동 휴대폰
유해사이트 점검

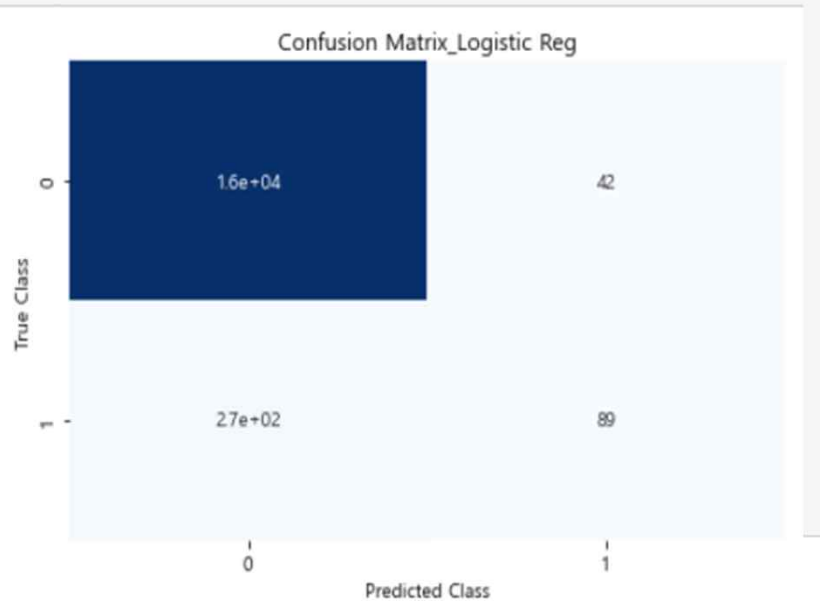


# 7. 모델링 – Logistic Regression 1

In [133]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix
import mglearn
logreg = LogisticRegression(random_state=42).fit(X_train, y_train)
print("훈련 세트 점수: {:.3f}".format(logreg.score(X_train, y_train)))
print("테스트 세트 점수: {:.3f}".format(logreg.score(X_test, y_test)))

from sklearn.metrics import confusion_matrix


prediction =  logreg.predict(X_test)
confusion = confusion_matrix(y_true=y_test, y_pred=prediction)
print("로지스틱 회귀 1")
print(confusion)


```

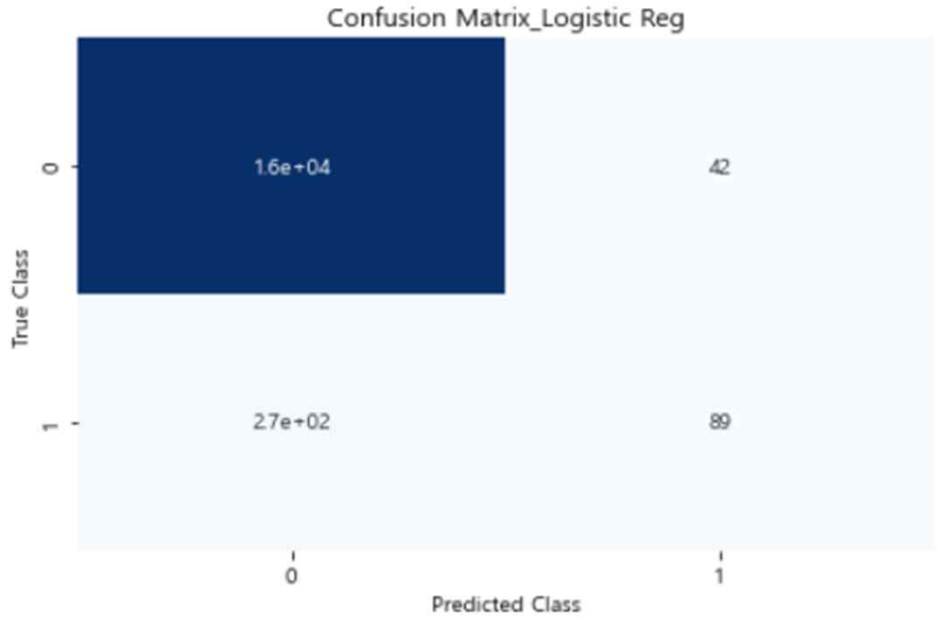
```
훈련 세트 점수: 0.982
테스트 세트 점수: 0.981
로지스틱 회귀 1
[[15805    42]
 [  271    89]]
```



# 7. 모델링 – Logistic Regression 2

```
훈련 세트 점수: 0.982
테스트 세트 점수: 0.981
로지스틱 회귀
[[15805    42]
 [  271    89]]
```

Solver : 최적화에 사용할 알고리즘 결정




큰 차이가 없음을 깨닳음 -> 알고리즘의 문제가 아님을 알 수 있음.

# 7. 모델링 – Logistic Regression 3

In [131]:

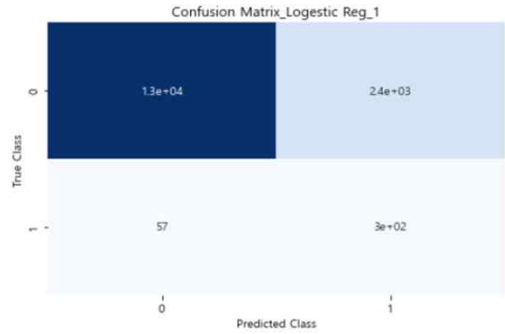
```
from sklearn.linear_model import LogisticRegression
import mglearn
logreg = LogisticRegression(random_state=0, class_weight="balanced").fit(X_trai
print("훈련 세트 점수: {:.3f}".format(logreg.score(X_train, y_train)))
print("테스트 세트 점수: {:.3f}".format(logreg.score(X_test, y_test)))

from sklearn.metrics import confusion_matrix


prediction =  logreg.predict(X_test)
confusion = confusion_matrix(y_true=y_test, y_pred=prediction)
print("로지스틱 회귀")
print(confusion)
```

```
훈련 세트 점수: 0.849
테스트 세트 점수: 0.846
로지스틱 회귀
[[13403  2444]
 [   57   303]]
```

```
C:₩Users₩a₩anaconda3₩lib₩site-packages₩sklearn₩utils₩validation.py:73: DataConversi
onWarning: A column-vector y was passed when a 1d array was expected. Please change
the shape of y to (n_samples, ), for example using ravel().
  return f(**kwargs)
```

전체적인 점수는
떨어졌지만
1을 더 잘 찾아냄

1 : 연체 有




What's Next?

- 보다 정확한 데이터 정제화
- 다양한 모델과 알고리즘을 사용하여 정확성 높이기
- Label engineering
- 크롤링과 감성분석을 이용하여 상권분석 (+ 식신 데이터)
- 카드사 데이터(특히 '재방문율') 用
- 웹  페이지 구현

모델링 후보 : DNN, Random Forest, XG Boost etc…

# 8. 서비스 제공 – Web Page 구현


- 사용자에게 매출액,

  부동산 보유여부,

  대출 보유여부,

  업장 정보 등

  필요한 정보를 입력 받는다.

# 8. 서비스 제공 – Web Page 구현

소상공인 신용평가 결과

| | | | |
|---|---|---|---|
| 신청번호 | 20051226215531970 | | |
| 업체명 | ABC | 주민등록번호(법인등록번호) | 7104221691814 |
| 대표자 | 임준우 | 사업자등록번호 | 3148207010 |
| 신청일 | 2006년 12월 20일 | | |
| 보증가능여부 | 보증가능 | | |

상기의 신용평가결과는 신청인의 입력자료에 따라 신용도 및 자금상태를 판단한 결과이며,
신용보증재단이 신청인에 대한 신용조사를 통해 입력자료 변경으로 신용보증가능여부가 변동될 수 있습니다.
계속 진행하시겠습니까? (보증불가의 경우 진행 불가)

확인 취소 인쇄

- 개인 사업자의

  대출 가능 여부를 출력하고

- 만약 보증 불가 하다면,

  critical한 변수를 보여준다.