# 데이터 마이닝 최종발표

## 고객 보험 가입 예측

[4조]김용현 권락원 정의연 윤유정 박서연

# INDEX

## 주제 선정

**목표** 효율적인 판촉을 위해 자동차 보험에 가입 가능성이 있는 **고객 선별**

**활용 데이터** 고객의 특성, 과거 보험 내역, 사고 여부, 판매 채널 경로 등등

**Data Source :** https://www.kaggle.com/anmolkumar/health-insurance-cross-sell-prediction

**Data Mining을 활용하여 보험 가입여부 분류, 예측**

## 데이터 파악

```
data_A = pd.read_csv("train.csv")
data_A
```

| | id | Gender | Age | Driving_License | Region_Code | Previously_Insured | Vehicle_Age | Vehicle_Damage | Annual_Premium | Policy_Sales_Channel |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | Male | 44 | 1 | 28.0 | 0 | > 2 Years | Yes | 40454.0 | 26.0 |
| **1** | 2 | Male | 76 | 1 | 3.0 | 0 | 1-2 Year | No | 33536.0 | 26.0 |
| **2** | 3 | Male | 47 | 1 | 28.0 | 0 | > 2 Years | Yes | 38294.0 | 26.0 |
| **3** | 4 | Male | 21 | 1 | 11.0 | 1 | < 1 Year | No | 28619.0 | 152.0 |
| **4** | 5 | Female | 29 | 1 | 41.0 | 1 | < 1 Year | No | 27496.0 | 152.0 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **381104** | 381105 | Male | 74 | 1 | 26.0 | 1 | 1-2 Year | No | 30170.0 | 26.0 |
| **381105** | 381106 | Male | 30 | 1 | 37.0 | 1 | < 1 Year | No | 40016.0 | 152.0 |
| **381106** | 381107 | Male | 21 | 1 | 30.0 | 1 | < 1 Year | No | 35118.0 | 160.0 |
| **381107** | 381108 | Female | 68 | 1 | 14.0 | 0 | > 2 Years | Yes | 44617.0 | 124.0 |
| **381108** | 381109 | Male | 46 | 1 | 29.0 | 0 | 1-2 Year | No | 41777.0 | 26.0 |

381109 rows × 12 columns

**Data Set : 12 columns, 381,109 records**

## 변수 파악

| | |
|---|---|
| Age | Age of Customer |
| Annual_Premium | The total amount of annual premiums paid |
| Vintage | Number of Days, Customer has been associated with the company(보험 가입 기간) |
| Gender | Male or Female |
| Driving_License | 0: 면허 미보유, 1: 면허 보유 |
| Region_Code | Unique code for the region of the customer |
| Previously_Insured | 0: 가입한 차량 보험이 없음, 1: 가입한 차량 보험이 있음 |
| Vehicle_Age | Age of the Vehicle |
| Vehicle_Damage | 0: 차량 사고가 발생한 적이 없음, 1: 차량 사고가 발생한 적이 있음 |
| Policy_Sales_Channel | Anonymized Code for the channel of outreaching to the customer ie. Different Agents, Over Mail, Over Phone, In Person, etc. |
| Response | 0: 가입가능성 없음, 1: 가입가능성 있음 |

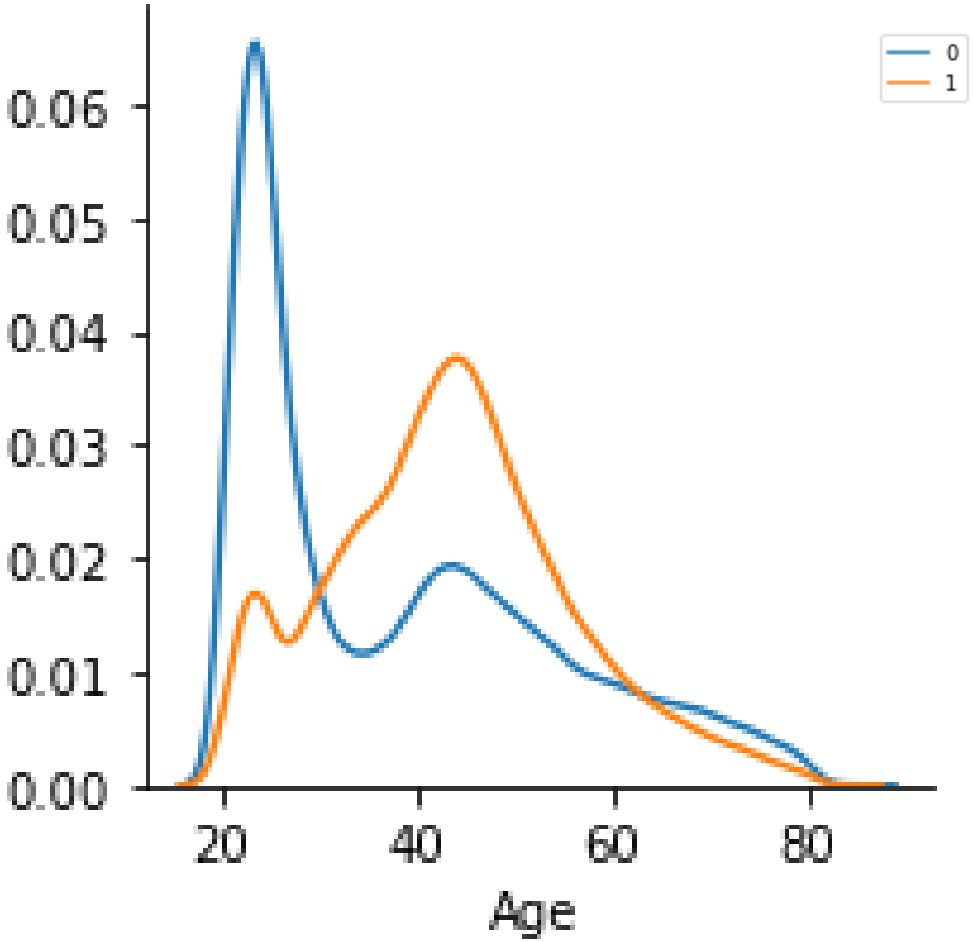
3개의 Numerical Data

7개의 Categorical Data

→ Target Data

## 수치형 데이터 파악

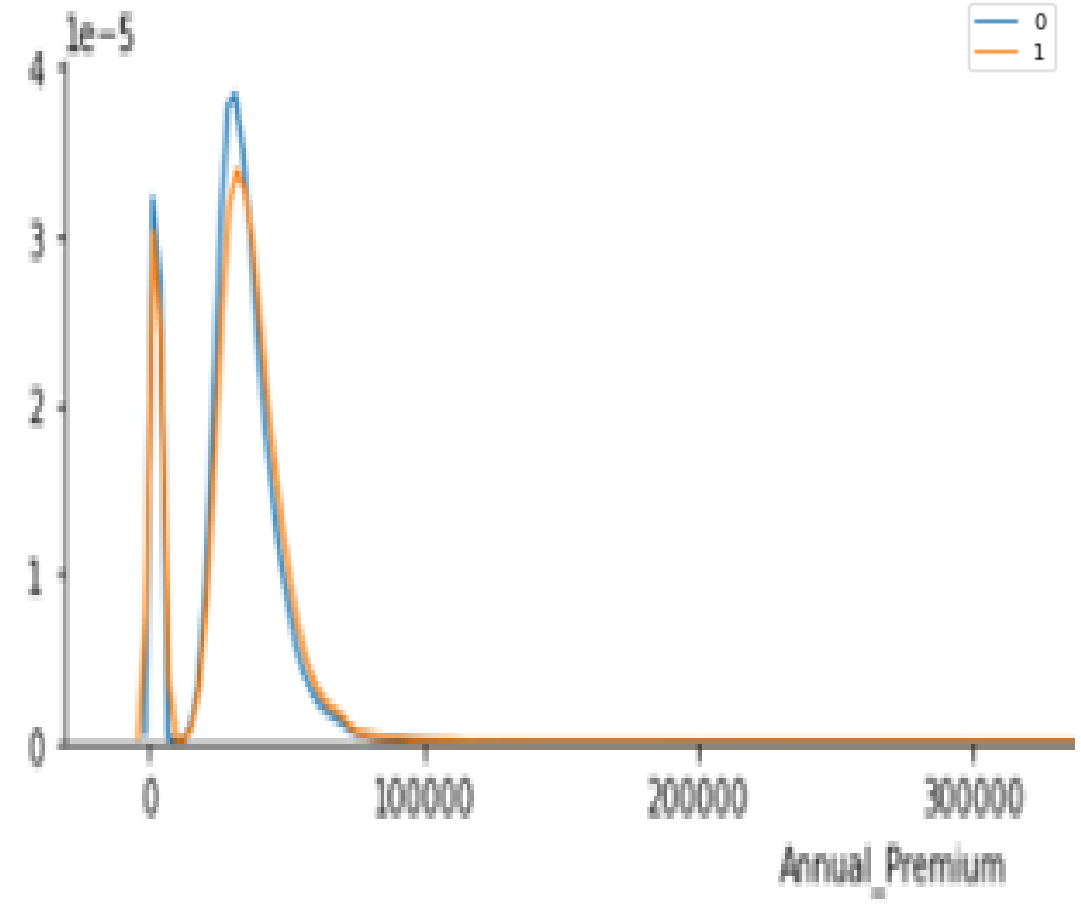
# Age




| | Age | Response | count |
|---|---|---|---|
| 0 | 20 | 0 | 6061 |
| 1 | 20 | 1 | 171 |
| 2 | 21 | 0 | 15883 |
| 3 | 21 | 1 | 574 |
| 4 | 22 | 0 | 20201 |
| ... | ... | ... | ... |
| 125 | 82 | 1 | 1 |
| 126 | 83 | 0 | 21 |
| 127 | 83 | 1 | 1 |
| 128 | 84 | 0 | 11 |
| 129 | 85 | 0 | 11 |

## 수치형 데이터 파악

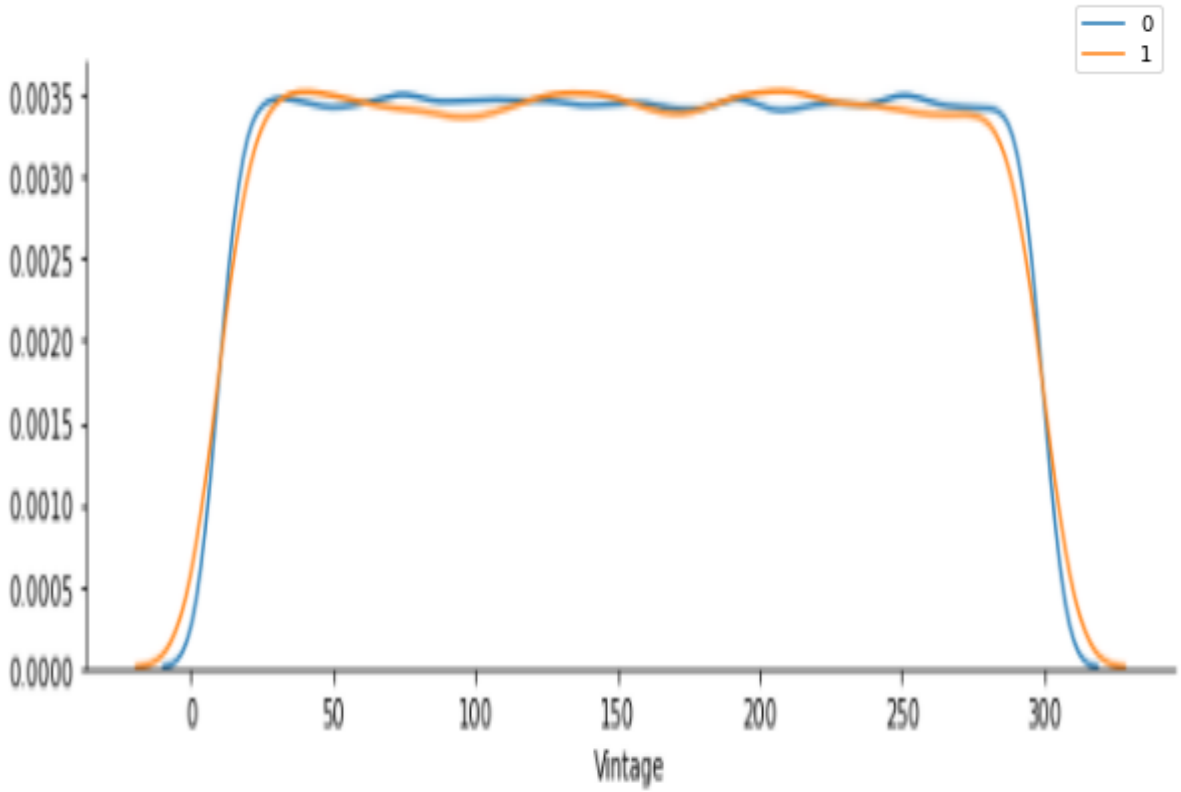
# Annual_Premium




| | Annual_Premium | Response | count |
|---|---|---|---|
| 0 | 2630.0 | 0 | 56372 |
| 1 | 2630.0 | 1 | 8505 |
| 2 | 6098.0 | 0 | 1 |
| 3 | 7670.0 | 0 | 1 |
| 4 | 8739.0 | 0 | 1 |
| ... | ... | ... | ... |
| 70289 | 489663.0 | 1 | 1 |
| 70290 | 495106.0 | 0 | 1 |
| 70291 | 508073.0 | 0 | 1 |
| 70292 | 540165.0 | 0 | 3 |
| 70293 | 540165.0 | 1 | 1 |

## 수치형 데이터 파악

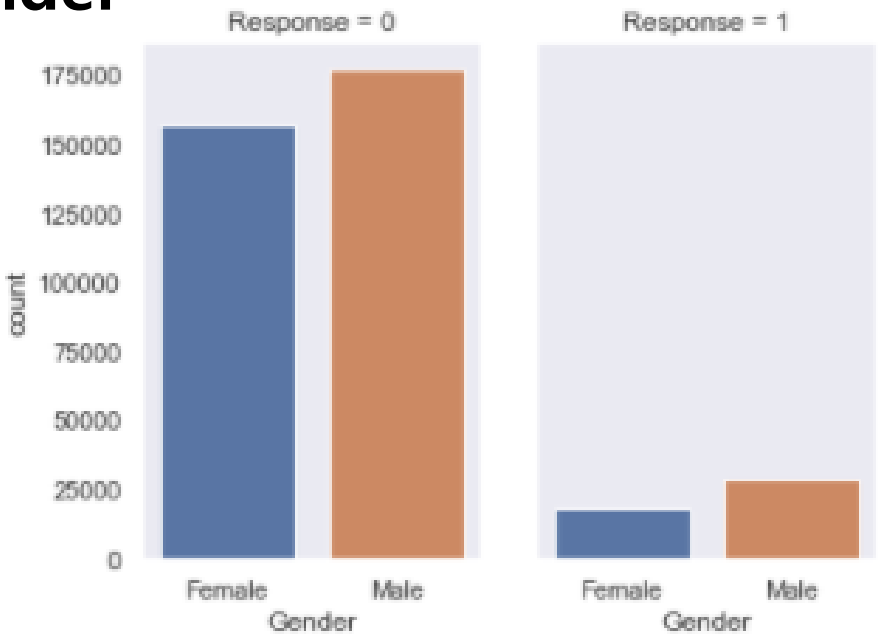
# Vintage




| | Vintage | Response | count |
|---|---|---|---|
| 0 | 10 | 0 | 1164 |
| 1 | 10 | 1 | 147 |
| 2 | 11 | 0 | 1153 |
| 3 | 11 | 1 | 191 |
| 4 | 12 | 0 | 1092 |
| ... | ... | ... | ... |
| 575 | 297 | 1 | 142 |
| 576 | 298 | 0 | 1198 |
| 577 | 298 | 1 | 186 |
| 578 | 299 | 0 | 1114 |
| 579 | 299 | 1 | 169 |

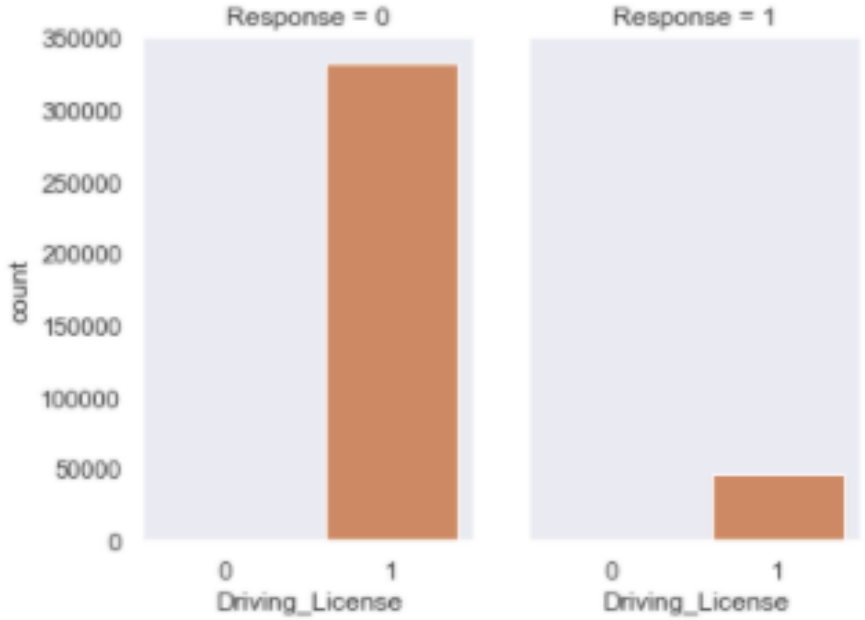
## 범주형 데이터 파악

### Gender




| | Gender | Response | count |
|---|---|---|---|
| 0 | Female | 0 | 156835 |
| 1 | Female | 1 | 18185 |
| 2 | Male | 0 | 177564 |
| 3 | Male | 1 | 28525 |

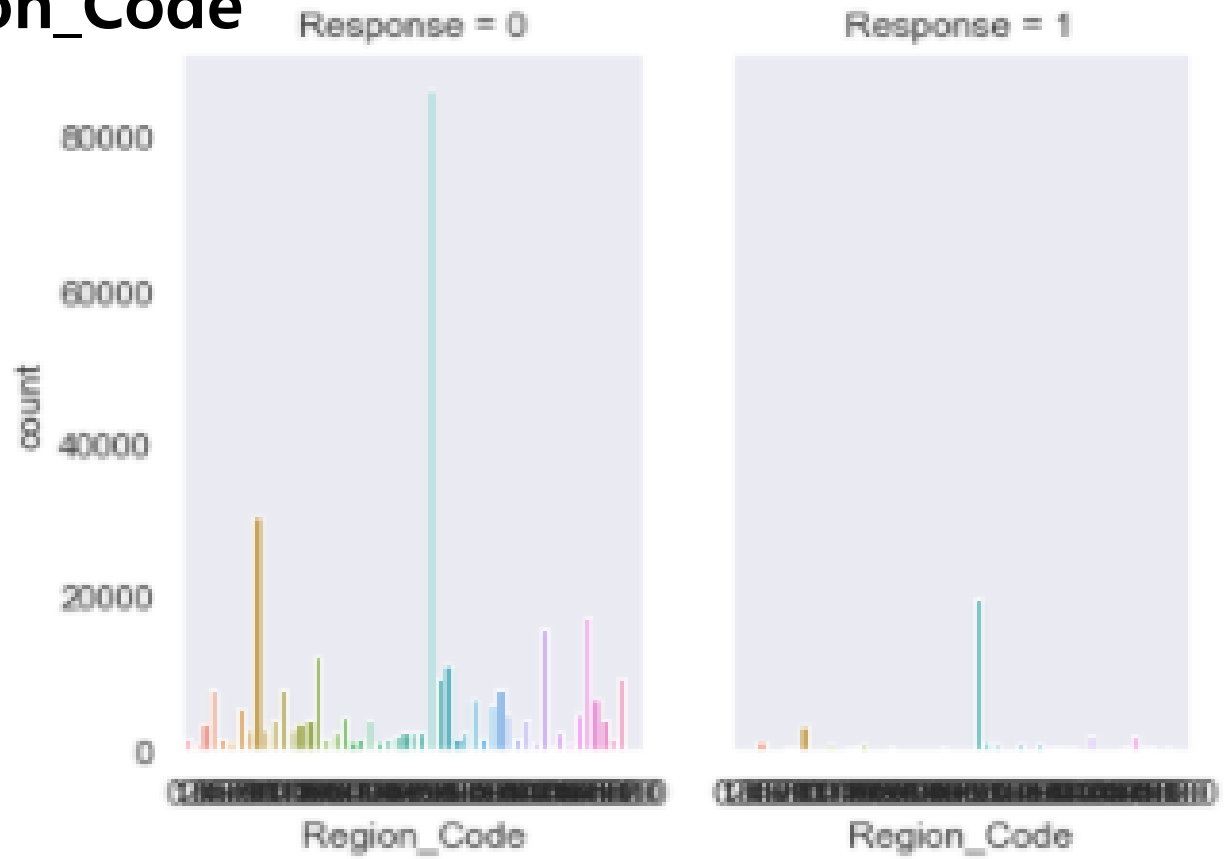
### Driving_License




| | Driving_License | Response | count |
|---|---|---|---|
| 0 | 0 | 0 | 771 |
| 1 | 0 | 1 | 41 |
| 2 | 1 | 0 | 333628 |
| 3 | 1 | 1 | 46669 |

## 범주형 데이터 파악

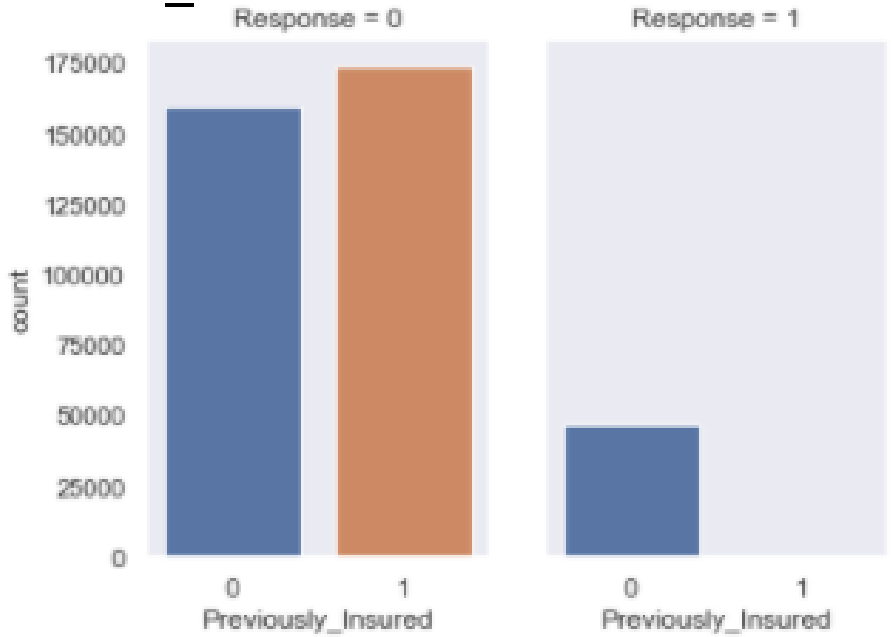
# Region_Code




| | Region_Code | Response | count |
|---|---|---|---|
| 0 | 0.0 | 0 | 1847 |
| 1 | 0.0 | 1 | 174 |
| 2 | 1.0 | 0 | 899 |
| 3 | 1.0 | 1 | 109 |
| 4 | 2.0 | 0 | 3751 |
| ... | ... | ... | ... |
| 101 | 50.0 | 1 | 642 |
| 102 | 51.0 | 0 | 155 |
| 103 | 51.0 | 1 | 28 |
| 104 | 52.0 | 0 | 234 |
| 105 | 52.0 | 1 | 33 |

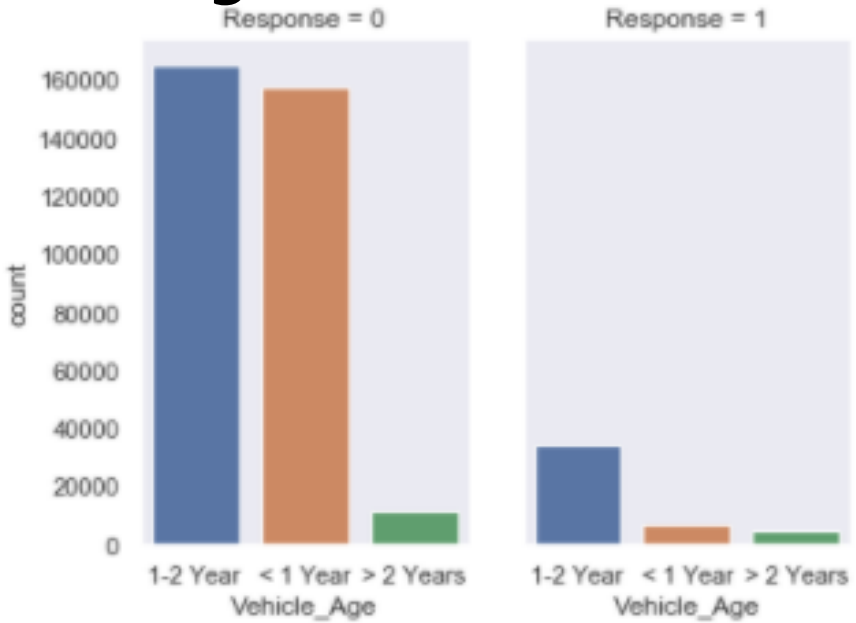
## 범주형 데이터 파악

# Previous_Insured




| | Previously_Insured | Response | count |
|---|---|---|---|
| 0 | 0 | 0 | 159929 |
| 1 | 0 | 1 | 46552 |
| 2 | 1 | 0 | 174470 |
| 3 | 1 | 1 | 158 |

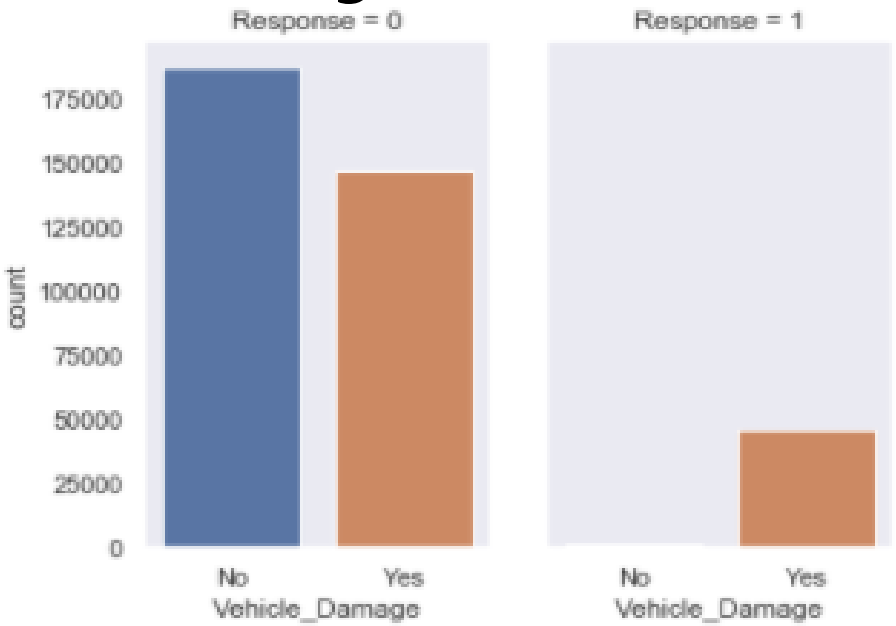
# Vehicle_Age




| | Vehicle_Age | Response | count |
|---|---|---|---|
| 0 | 1-2 Year | 0 | 165510 |
| 1 | 1-2 Year | 1 | 34806 |
| 2 | < 1 Year | 0 | 157584 |
| 3 | < 1 Year | 1 | 7202 |
| 4 | > 2 Years | 0 | 11305 |
| 5 | > 2 Years | 1 | 4702 |

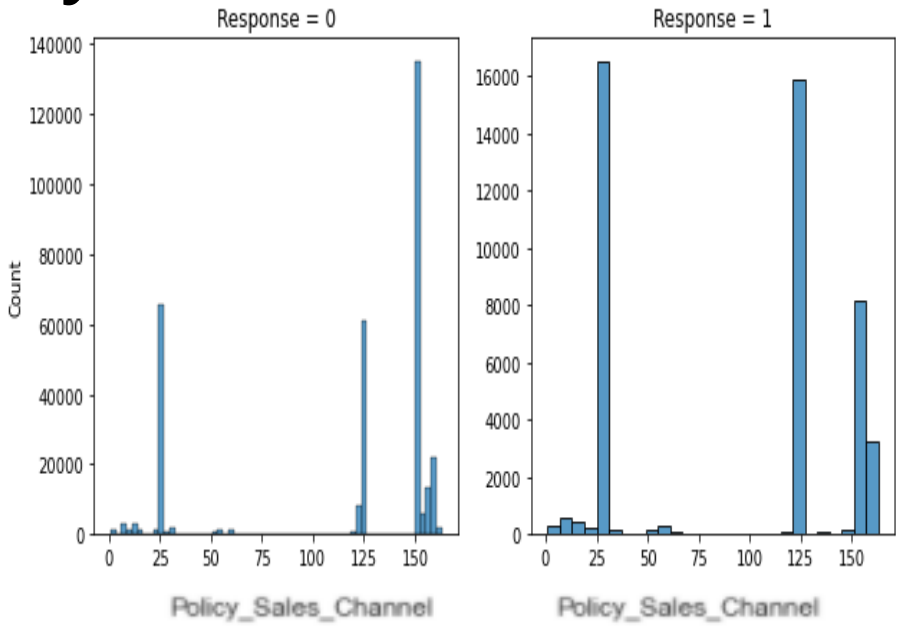
## 범주형 데이터 파악

# Vehicle_Damage



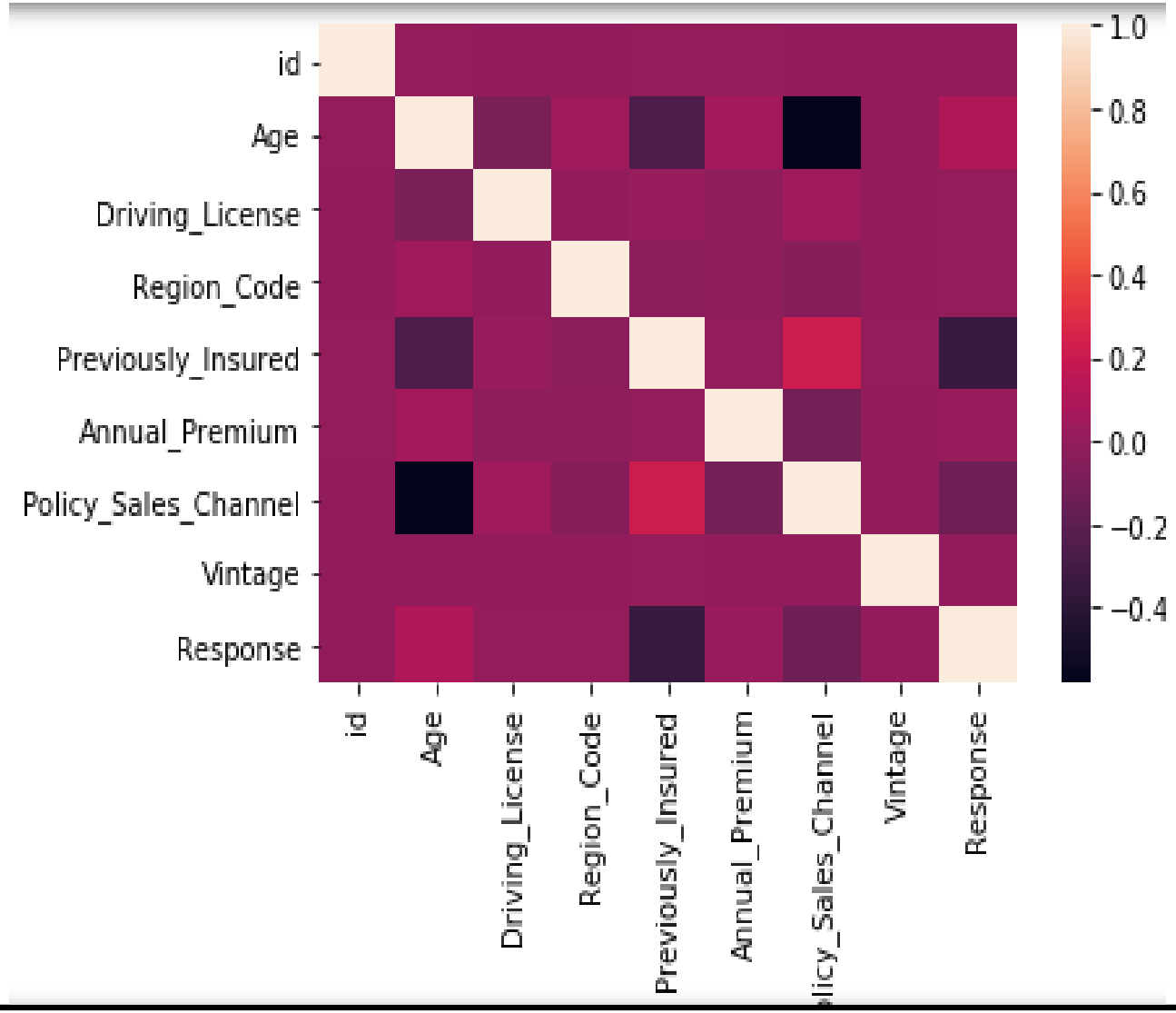

| | Vehicle_Damage | Response | count |
|---|---|---|---|
| 0 | No | 0 | 187714 |
| 1 | No | 1 | 982 |
| 2 | Yes | 0 | 146685 |
| 3 | Yes | 1 | 45728 |

# Policy_Sales_Channel




| | Policy_Sales_Channel | Response | count |
|---|---|---|---|
| 0 | 1.0 | 0 | 1039 |
| 1 | 1.0 | 1 | 35 |
| 2 | 2.0 | 0 | 3 |
| 3 | 2.0 | 1 | 1 |
| 4 | 3.0 | 0 | 364 |
| ... | ... | ... | ... |
| 269 | 159.0 | 1 | 1 |
| 270 | 160.0 | 0 | 21304 |
| 271 | 160.0 | 1 | 475 |
| 272 | 163.0 | 0 | 2013 |
| 273 | 163.0 | 1 | 880 |

## 상관관계 분석




각 변수들의 상관관계 분석

특별한 상관관계가 없어

모든 변수로 모델링 진행

## 변수 변환

```
# 범주형 변수 중 str형식으로 되어있는 변수 0,1,2로 변경
# 범주형 변수: Gender, Driving_License, Region_Code, Previously_Insured, V
# Gender: Male-1, Female-0
data_A.loc[data_A["Gender"] == "Female","Gender"] = 0
data_A.loc[data_A["Gender"] == "Male","Gender"] = 1
# Vehicle_Age: <1-0, 1-2:1, >2:2
data_A.loc[data_A["Vehicle_Age"] == "< 1 Year","Vehicle_Age"] = 0
data_A.loc[data_A["Vehicle_Age"] == "1-2 Year","Vehicle_Age"] = 1
data_A.loc[data_A["Vehicle_Age"] == "> 2 Years","Vehicle_Age"] = 2
# Vehicle_Damage: Yes-1, No-1
data_A.loc[data_A["Vehicle_Damage"] == "No","Vehicle_Damage"] = 0
data_A.loc[data_A["Vehicle_Damage"] == "Yes","Vehicle_Damage"] = 1
```

**범주형 변수들에 숫자 값 부여**

## MinMax scaler

```
from sklearn.preprocessing import MinMaxScaler
data_B = data_A
data_B['Age'] = (data_B['Age']-data_B['Age'].min())/(data_B['Age'].max()-data_B['Age'].min())
data_B['Annual_Premium'] = (data_B['Annual_Premium']-data_B['Annual_Premium'].min())/(data_B['Annual_Premium'].max()-data_B['Annual_P
data_B['Vintage'] = (data_B['Vintage']-data_B['Vintage'].min())/(data_B['Vintage'].max()-data_B['Vintage'].min())
```

**변수들의 단위가 달라 영향이 존재**
**MinMax Scaler를 이용하여 해결**

## 전처리 전·후 비교

| | id | Gender | Age | Driving_License | Region_Code | Previously_Insured | Vehicle_Age | Vehicle_Damage | Annual_Premium | Policy_Sales_Channel | Vintage |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | Male | 44 | 1 | 28.0 | 0 | > 2 Years | Yes | 40454.0 | 26.0 | 217 |
| 1 | 2 | Male | 76 | 1 | 3.0 | 0 | 1-2 Year | No | 33536.0 | 26.0 | 183 |
| 2 | 3 | Male | 47 | 1 | 28.0 | 0 | > 2 Years | Yes | 38294.0 | 26.0 | 27 |
| 3 | 4 | Male | 21 | 1 | 11.0 | 1 | < 1 Year | No | 28619.0 | 152.0 | 203 |
| 4 | 5 | Female | 29 | 1 | 41.0 | 1 | < 1 Year | No | 27496.0 | 152.0 | 39 |

| | Gender | Age | Driving_License | Region_Code | Previously_Insured | Vehicle_Age | Vehicle_Damage | Annual_Premium | Policy_Sales_Channel | Vintage |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0.369231 | 1 | 28.0 | 0 | 2 | 1 | 0.070366 | 26.0 | 0.716263 |
| 1 | 1 | 0.861538 | 1 | 3.0 | 0 | 1 | 0 | 0.057496 | 26.0 | 0.598616 |
| 2 | 1 | 0.415385 | 1 | 28.0 | 0 | 2 | 1 | 0.066347 | 26.0 | 0.058824 |
| 3 | 1 | 0.015385 | 1 | 11.0 | 1 | 0 | 0 | 0.048348 | 152.0 | 0.667820 |
| 4 | 0 | 0.138462 | 1 | 41.0 | 1 | 0 | 0 | 0.046259 | 152.0 | 0.100346 |

## 데이터 분리 및 Over Sampling

```
# 트레이닝, 테스트 데이터 분리
X_train, X_test, y_train, y_test = train_test_split(data_B, res, test_size=0.3, random_state=0)
X_train, X_val, y_train, y_val = train_test_split(data_B, res, test_size=0.2/0.7, random_state=0)
```

```
# Smote 샘플링 // 모델설정(ratio 비율설정, kind 분류선 설정)
smote_nc = SMOTENC([0, 2, 3, 4, 5, 6, 8], random_state=0)
X_resampled, y_resampled = smote_nc.fit_resample(X_train,list(y_train))
```

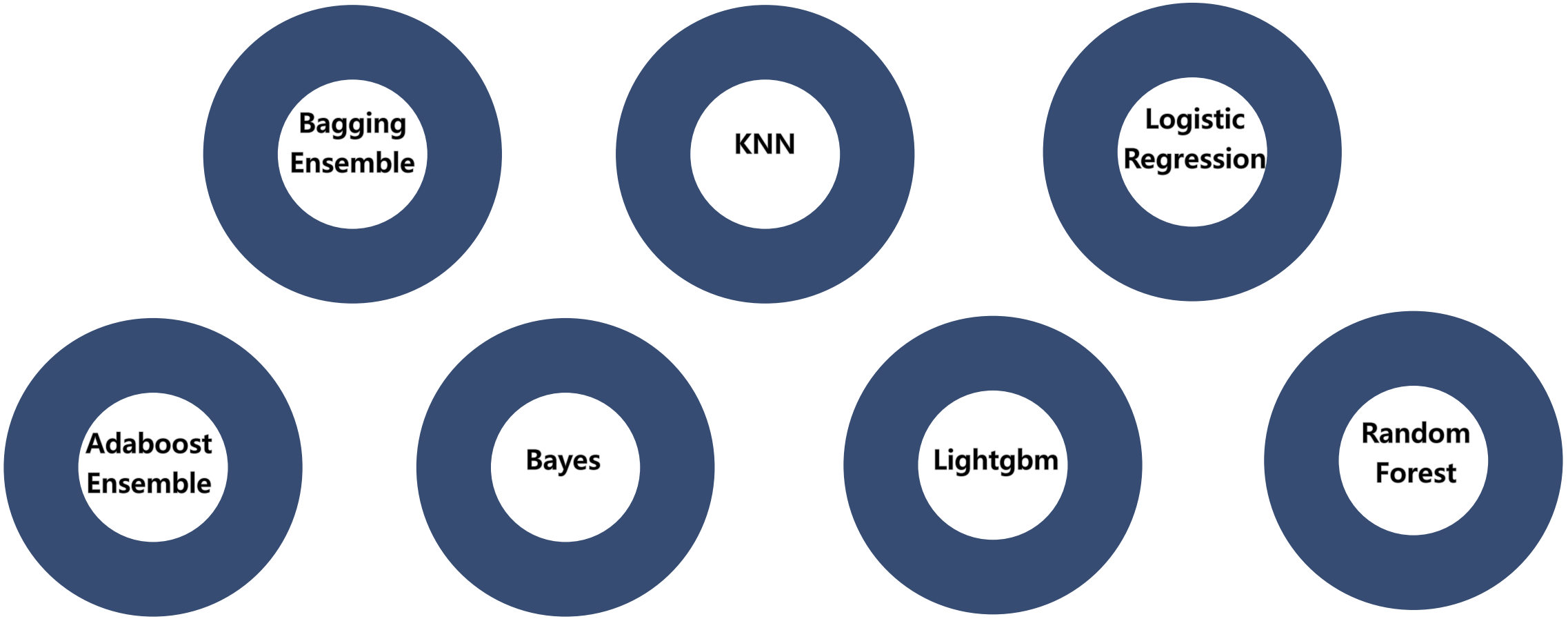
```
X_resampled
```

| | Gender | Age | Driving_License | Region_Code | Previously_Insured | Vehicle_Age | Vehicle_Damage | Annual_Premium | Policy_Sales_Channel |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0.076923 | 1 | 50.0 | 1 | 0 | 0 | 0.071106 | 152.0 |
| 1 | 0 | 0.600000 | 1 | 28.0 | 0 | 2 | 1 | 0.000000 | 158.0 |
| 2 | 1 | 0.784615 | 1 | 46.0 | 0 | 1 | 1 | 0.069060 | 11.0 |
| 3 | 0 | 0.046154 | 1 | 11.0 | 1 | 0 | 0 | 0.058925 | 152.0 |
| 4 | 0 | 0.046154 | 1 | 2.0 | 1 | 0 | 0 | 0.053913 | 152.0 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 477767 | 1 | 0.509545 | 1 | 46.0 | 0 | 1 | 1 | 0.070452 | 26.0 |
| 477768 | 1 | 0.239825 | 1 | 28.0 | 0 | 1 | 1 | 0.076413 | 26.0 |
| 477769 | 1 | 0.430769 | 1 | 28.0 | 0 | 1 | 1 | 0.072406 | 26.0 |
| 477770 | 0 | 0.225870 | 1 | 28.0 | 0 | 1 | 1 | 0.065035 | 26.0 |
| 477771 | 1 | 0.422980 | 1 | 28.0 | 0 | 1 | 1 | 0.054870 | 26.0 |

**정확한 학습을 위해 SMOTENC 기법 적용**

**Target 비율을 1:1로 조정**

## 7가지 모델선정


Bagging Ensemble
KNN
Logistic Regression
Adaboost Ensemble
Bayes
Lightgbm
Random Forest

## Parameter

```python
from sklearn.model_selection import GridSearchCV

params = { 'n_estimators' : [10, 30, 50, 70, 100],
           'max_depth' : [10, 30, 50, 70],
           'min_samples_leaf' : [8, 12, 14, 18],
           'min_samples_split' : [20, 30, 40]
         }

# RandomForestClassifier 객체 생성 후 GridSearchCV 수행
rf_clf = RandomForestClassifier(random_state = 0, n_jobs = -1)
grid_cv = GridSearchCV(rf_clf, param_grid = params, cv = 3, n_jobs = -1)
grid_cv.fit(X_train, y_train)

print('최적 하이퍼 파라미터: ', grid_cv.best_params_)
print('최고 예측 정확도: {:.4f}'.format(grid_cv.best_score_))
```

```
최적 하이퍼 파라미터:  {'max_depth': 70, 'min_samples_leaf': 8, 'min_samples_split': 40, 'n_estimators': 100}
최고 예측 정확도: 0.7777
```

**Grid search 기법을 통해 최적의 Parameter 설정**

**max_depth : 70**

**min_samples_leaf : 8**

**min_samples_split : 40**

**n_estimators : 100**

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.98 | 0.70 | 0.82 | 95513 |
| 1 | 0.29 | 0.88 | 0.44 | 13376 |
| accuracy | | | 0.73 | 108889 |
| macro avg | 0.63 | 0.79 | 0.63 | 108889 |
| weighted avg | 0.89 | 0.73 | 0.77 | 108889 |

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.96 | 0.77 | 0.85 | 95513 |
| 1 | 0.31 | 0.76 | 0.45 | 13376 |
| accuracy | | | 0.77 | 108889 |
| macro avg | 0.64 | 0.77 | 0.65 | 108889 |
| weighted avg | 0.88 | 0.77 | 0.80 | 108889 |

## Adaboost Ensemble

### Confusion Matrix

```
[[82632 12881]
[ 7450  5926]]
```

### Classification Report

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.92 | 0.87 | 0.89 | 95513 |
| 1 | 0.32 | 0.44 | 0.37 | 13376 |
| accuracy | | | 0.81 | 108889 |
| macro avg | 0.62 | 0.65 | 0.63 | 108889 |
| weighted avg | 0.84 | 0.81 | 0.83 | 108889 |

**Adaboost**

**adaptive + boosting**

**간단한 약분류기들이 상호보완 하도록**

**단계적(순차적)으로 학습, 가중치 할당**

**예측의 정확도에 따라 가중치 변화**

## Bagging Ensemble

### Confusion Matrix

```
[[64522 30991]
 [ 1259 12117]]
```

### Classification Report

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.98 | 0.68 | 0.80 | 95513 |
| 1 | 0.28 | 0.91 | 0.43 | 13376 |
| accuracy | | | 0.70 | 108889 |
| macro avg | 0.63 | 0.79 | 0.61 | 108889 |
| weighted avg | 0.89 | 0.70 | 0.75 | 108889 |

**Bagging**

**bootstrap + aggregating**

**학습용 데이터셋을 변화시켜 각각의 베이스 모델들을 만드는 기법**

**불안정한 모델들의 안정성 개선가능**

## Bayes

### Confusion Matrix

```
[[69293 26220]
[ 5147  8229]]
```

### Classification Report

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.93 | 0.73 | 0.82 | 95513 |
| 1 | 0.24 | 0.62 | 0.34 | 13376 |
| accuracy | | | 0.71 | 108889 |
| macro avg | 0.58 | 0.67 | 0.58 | 108889 |
| weighted avg | 0.85 | 0.71 | 0.76 | 108889 |

**Bayes**
**베이즈 정리에 기반한 통계적 분류**
**가장 단순한 지도 중 하나.**

**Feature끼리 서로 독립이라는**
**조건필요**

## KNN

**Confusion Matrix**

```
[[79665 15848]
 [ 6797  6579]]
```

**Classification Report**

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.92 | 0.83 | 0.86 | 95513 |
| 1 | 0.29 | 0.49 | 0.37 | 13376 |
| accuracy | | | 0.79 | 108889 |
| macro avg | 0.61 | 0.66 | 0.62 | 108889 |
| weighted avg | 0.84 | 0.79 | 0.81 | 108889 |

**kNN**

새로운 데이터와
기존 데이터들간 거리를 측정
가까운 데이터들의 종류가 무엇인지
확인하여 새로운 데이터의 종류를 판
별하는 알고리즘

모든 데이터를 비교하므로 데이터가
많으면 처리시간 상당

## Lightgbm

### Confusion Matrix

```
[[74377 21136]
[ 3507  9869]]
```

### Classification Report

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.95 | 0.78 | 0.86 | 95513 |
| 1 | 0.32 | 0.74 | 0.44 | 13376 |
| accuracy | | | 0.77 | 108889 |
| macro avg | 0.64 | 0.76 | 0.65 | 108889 |
| weighted avg | 0.88 | 0.77 | 0.81 | 108889 |

**Light GBM**

Light GBM은 다른 알고리즘과 다르게 Tree가 수직적으로 확장

즉, Light GBM은 leaf-wise 인 반면 다른 알고리즘은 level-wise

확장하기 위해서 max delta loss를 가진 leaf를 선택, 동일한 leaf를 확장할 때, leaf-wise 알고리즘은 level-wise 알고리즘보다 더 많은 loss, 손실을 줄일 수 있음

## Logistic Regression

### Confusion Matrix

```
[[56307 39206]
[  295 13081]]
```

### Classification Report

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.99 | 0.59 | 0.74 | 95513 |
| 1 | 0.25 | 0.98 | 0.40 | 13376 |
| accuracy | | | 0.64 | 108889 |
| macro avg | 0.62 | 0.78 | 0.57 | 108889 |
| weighted avg | 0.90 | 0.64 | 0.70 | 108889 |

Logistic Regression
Linear regression을 dependent variable Y가 categorical variable인 경우로 확장한 것

가중치의 해석이 어려움

## Random Forest

### Confusion Matrix

```
[[67238 28275]
 [ 1636 11740]]
```

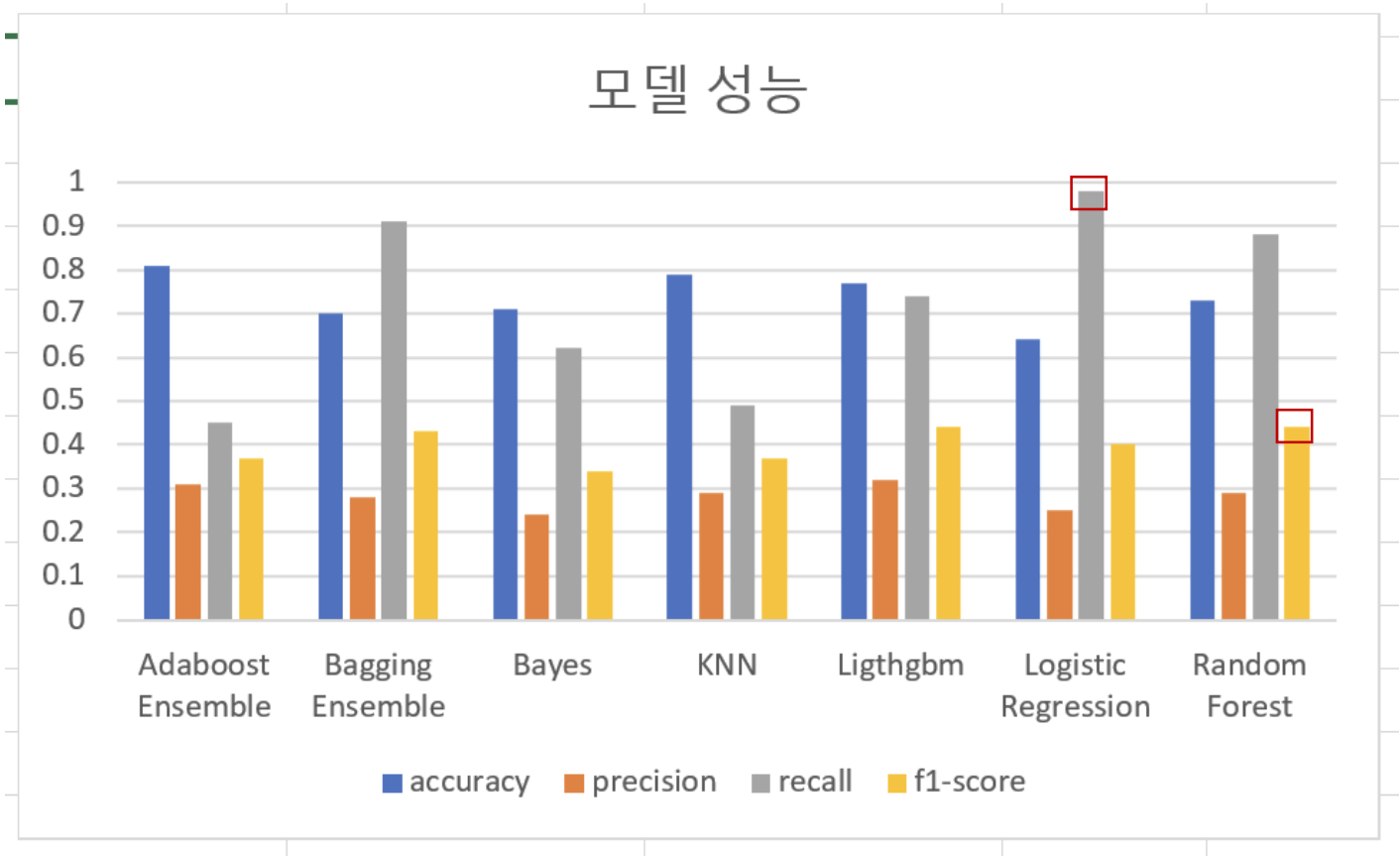
### Classification Report

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.98 | 0.70 | 0.82 | 95513 |
| 1 | 0.29 | 0.88 | 0.44 | 13376 |
| accuracy | | | 0.73 | 108889 |
| macro avg | 0.63 | 0.79 | 0.63 | 108889 |
| weighted avg | 0.89 | 0.73 | 0.77 | 108889 |

**Random Forest**
**결정 트리를 기반으로**
**여러 개의 결정트리 classifier를 생성**
**각자의 방식으로 sampling 하여**
**개별적으로 학습**

**훈련데이터에 과적합되는 경향이 있음**

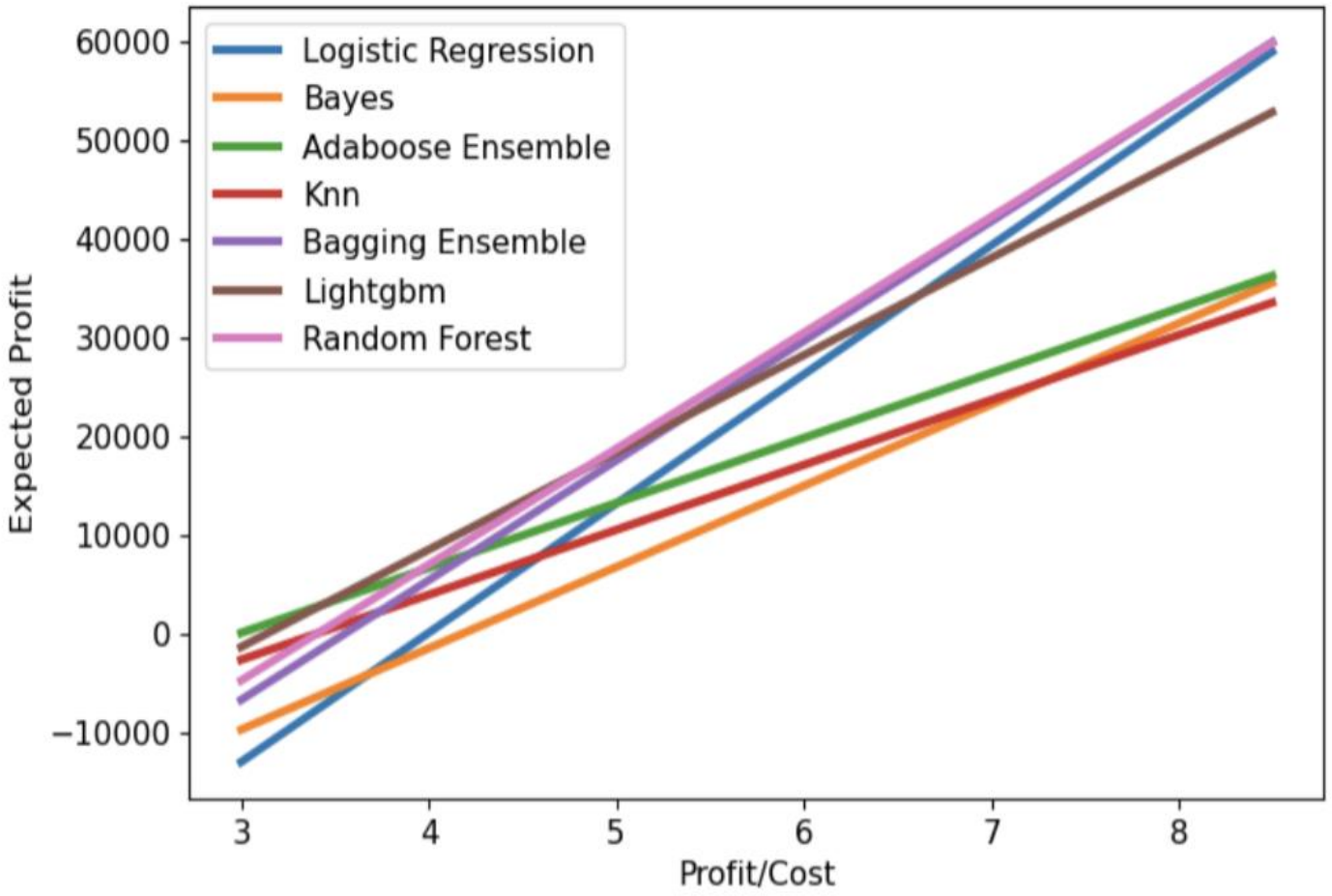
## 모델 성능 비교




모델들의 성능이
큰 차이를 보이지 않음

기대수익 고려

## 모델 성능 비교




기대수익 = $(E * A) * x$- $Ec$

예측값 : $E$

실제값 : $A$

수익 : $x$

비용 : $c$

(비용은 1로 가정)

## 최종 분류 결과

```
# 랜덤포레스트 모델링(criterion:분류기준,n_estimators:나무 생성 수,min_samples_split:,max_features:최대 고려 특성 수,oob_score:정확도)
RF = RandomForestClassifier(criterion = 'entropy',n_estimators=100,min_samples_split=40,min_samples_leaf=8,max_features=4,max_depth=70,ran
RF.fit(X_resampled, y_resampled)
```

```
RandomForestClassifier(criterion='entropy', max_depth=70, max_features=4,
                       min_samples_leaf=8, min_samples_split=40,
                       random_state=0)
```

```
y_pred_RF = RF.predict(X_test)
```

```
Test data
 [[79392 20849]
 [ 3960 10132]]
```

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.95 | 0.79 | 0.86 | 100241 |
| 1 | 0.33 | 0.72 | 0.45 | 14092 |
| accuracy | | | 0.78 | 114333 |
| macro avg | 0.64 | 0.76 | 0.66 | 114333 |
| weighted avg | 0.88 | 0.78 | 0.81 | 114333 |

**Grid search를 이용하여 찾은 parameter를 설정한 분류 결과, 모델중 가장 높은 성능을 보임**

## 최종결론

최종목표 : 기대수익 및 Recall 최대화

Parameter 수정을 통한 최적화

‘Random Forest’ 모델을 채택

# THANK YOU