항공 운항 데이터 활용

“항공 지연 예측”

퓨처스리그

TEAM NAME : NA랑 NULL자

INDEX

01 Project Purpose

01 Project Purpose

프로젝트 목표

- 2017년 1월 1일 ~ 2019년 6월 30일의 항공 데이터를 활용하여
- 2019년 9월의 항공 지연을 예측하는 모델들을 세우고
- 정확도와 f1-score(정밀도, 재현도) 를 기준으로 모델을 선택한다.

02 Data & EDA

02 Data & EDA

(1) 데이터 정제

주어진 AFSNT.csv 데이터 활용 -> 2017년 1월~ 2019년 6월 30일 간의 국내항공운항데이터

■ - 제거 ■ - 추가

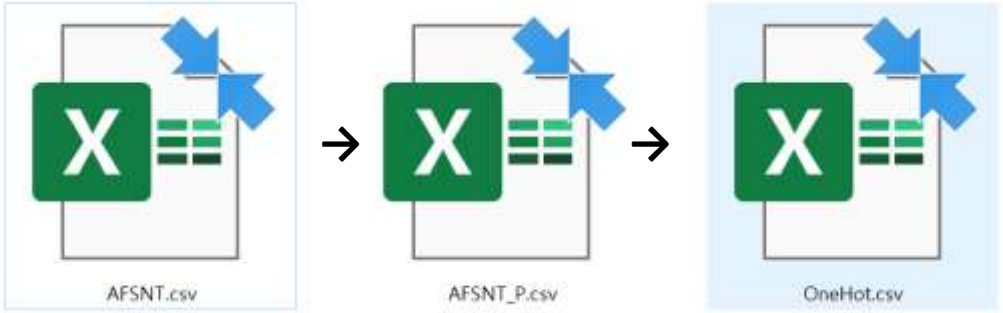
AFSNT.csv	→	AFSNT_P.CSV
SDT_YY	년	
SDT_MM	월	
SDT_DD	일	
SDT_DY	요일	
ARP	해당공항	
ODP	상대공항	
FLO	항공사	
AOD	출도착 A,D	
DLY	지연 Y,N	
STT	계획시각	(정수) 시 , 분 -> CNT변수에 활용 (정수) 시 -> OneHot.csv 에 (운항)시간대 변수로 활용
CNT	정수값	한 운항의 (STT - 1시간) ~ STT인 같은 ARP에서의 운항 수
FLT	편명	변수로 사용하기에 정보 부족
REG	등록기호	FLT와 동일
IRR	부정기편	표본 수가 작아 구분 X
ATT	실제시각	예측할 데이터에 적용 어렵다고 판단, 제거
DRR	지연사유	특정 사유가 큰 비중 차지하여 구분 필요 없다고 판단
CNL	결항여부	
CNR	결항사유	

02 Data & EDA

(1) 데이터 정제

* 범주형 변수들을 0과 1인 가변수로 변형

OnehotCSV			
DLY	지연여부	ARP_ARP1 ~ ARP15	해당공항
MM_1 ~ MM_12	월	ODP_ARP1 ~ ARP15	상대공항
DD_1 ~ DD_31	일	FLO_A ~ FLO_L	항공사
DY_월 ~ DY_일	요일	AOD_A , AOD_D	도착 , 출발
HR_0 ~ HR_23	시각	CNT	1시간 이전부터의 운항 수
SDT_YY	년	STT	계획시각

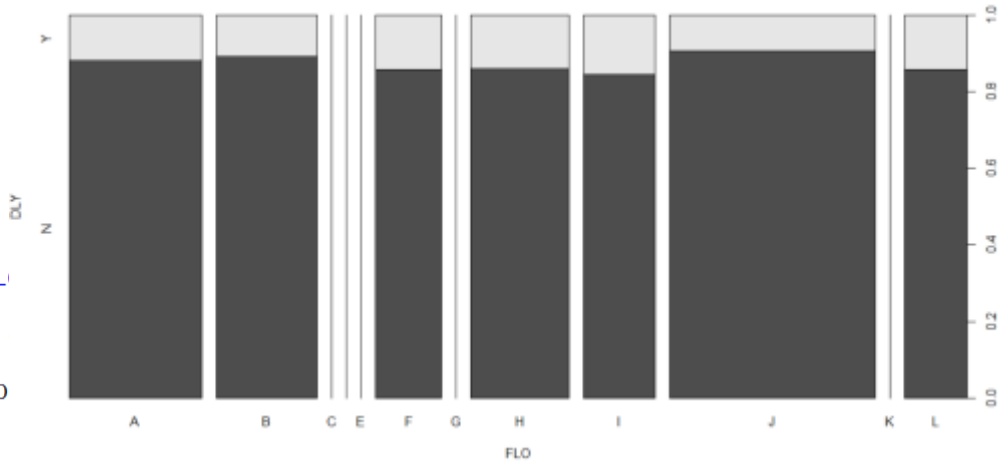


이후 OneHot.csv 를 이용해 train , test 데이터로 나누어 모델 학습

02 Data & EDA

(2) EDA

각 FLO 에 대한 지연 비교




```
> prop.test(x = c(sum(J) , sum(ALL)) , n = c(sum(SDT_FL

        2-sample test for equality of proportions with

data:  c(sum(J), sum(ALL)) out of c(sum(SDT_FLOTEST$FLO
X-squared = 1514.2, df = 1, p-value < 2.2e-16
alternative hypothesis: two.sided
95 percent confidence interval:
 -0.02793568 -0.02540716
sample estimates:
    prop 1     prop 2
0.09374672 0.12041814
```

```
> prop.test(x = c(sum(B) , sum(ALL)) , n = c(sum(SDT_FLOTEST$FLO == 'B') , length(SDT_FLOTEST$FLO)))

        2-sample test for equality of proportions with continuity correction

data:  c(sum(B), sum(ALL)) out of c(sum(SDT_FLOTEST$FLO == "B"), length(SDT_FLOTEST$FLO))
X-squared = 178.61, df = 1, p-value < 2.2e-16
alternative hypothesis: two.sided
95 percent confidence interval:
 -0.01432444 -0.01076888
sample estimates:
   prop 1    prop 2
0.1078715 0.1204181
```

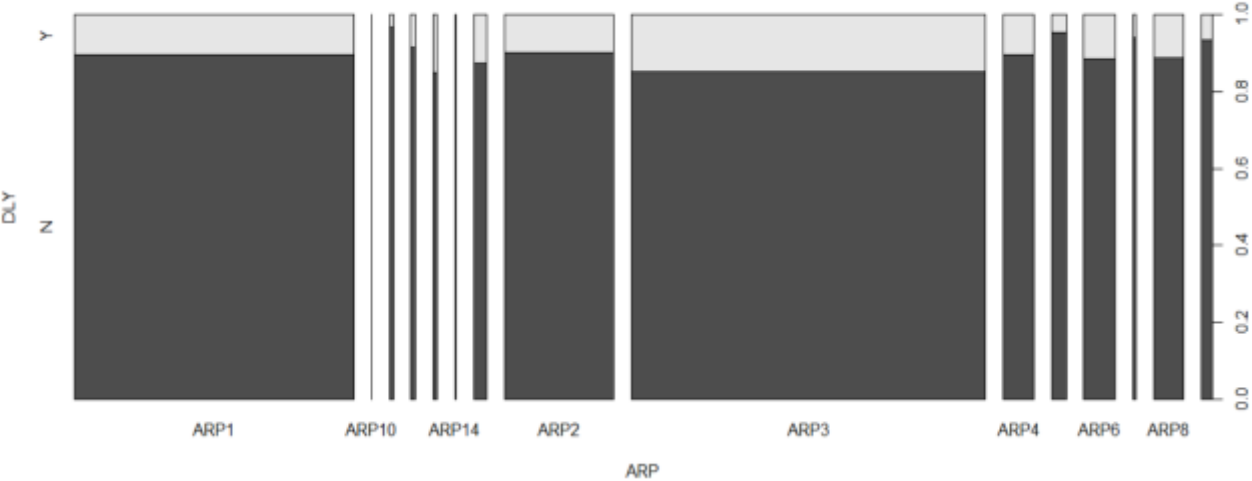
→ FLO별 지연이 어느정도 차이가 있다.

02 Data & EDA

(2) EDA

각 ARP에 대한 지연 비교

```
> sort(table(SDT$ARP)) # ARP1 ARP3 의 이용이 가장 많음
```




```
ARP4   ARP6   ARP2   ARP1   ARP3
33623  34472 121513 310665 393607
```

```
> prop.test(x = c(sum(ARP3), sum(AP_ALL)) , n

        2-sample test for equality of proporti

data:  c(sum(ARP3), sum(AP_ALL)) out of c(sum(
X-squared = 1901.6, df = 1, p-value < 2.2e-16
alternative hypothesis: two.sided
95 percent confidence interval:
 0.02671090 0.02932671
sample estimates:
   prop 1    prop 2
0.1495298 0.1215110
```

```
> prop.test(x = c(sum(ARP1), sum(AP_ALL)) , n

        2-sample test for equality of proport

data:  c(sum(ARP1), sum(AP_ALL)) out of c(sum
X-squared = 627.4, df = 1, p-value < 2.2e-16
alternative hypothesis: two.sided
95 percent confidence interval:
 -0.01808997 -0.01554179
sample estimates:
   prop 1    prop 2
0.1046951 0.1215110
```

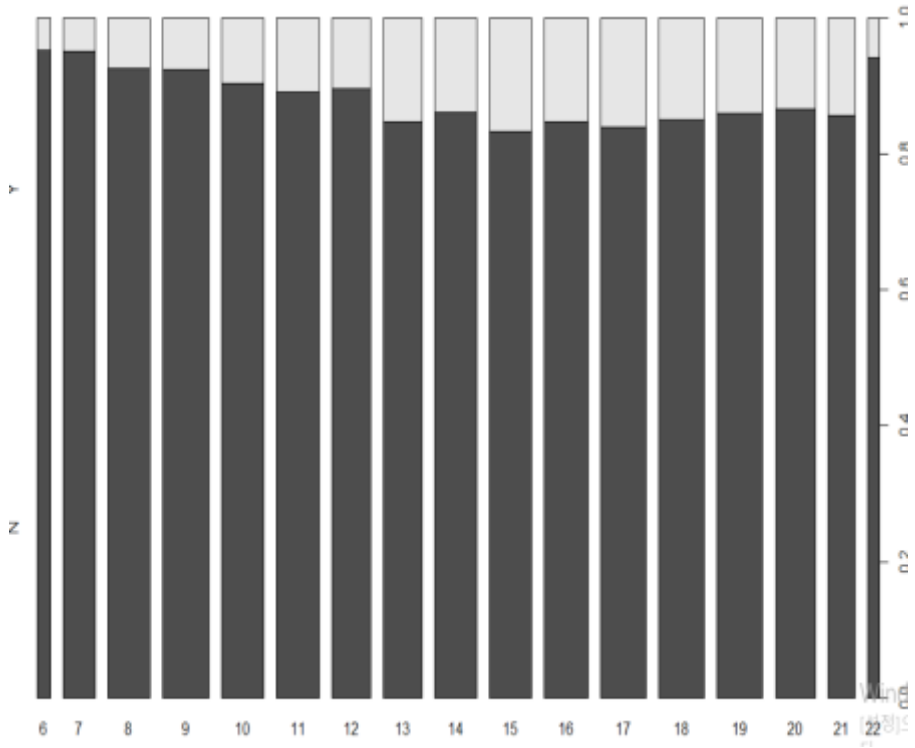
```
> prop.test(x = c(sum(ARP2), sum(AP_ALL)) , n

        2-sample test for equality of proport

data:  c(sum(ARP2), sum(AP_ALL)) out of c(sum
X-squared = 502.21, df = 1, p-value < 2.2e-16
alternative hypothesis: two.sided
95 percent confidence interval:
 -0.02392055 -0.02029968
sample estimates:
    prop 1     prop 2
0.09940086 0.12151098
```

→ ARP별 지연이 어느정도 차이가 있다.

02 Data & EDA

(2) EDA

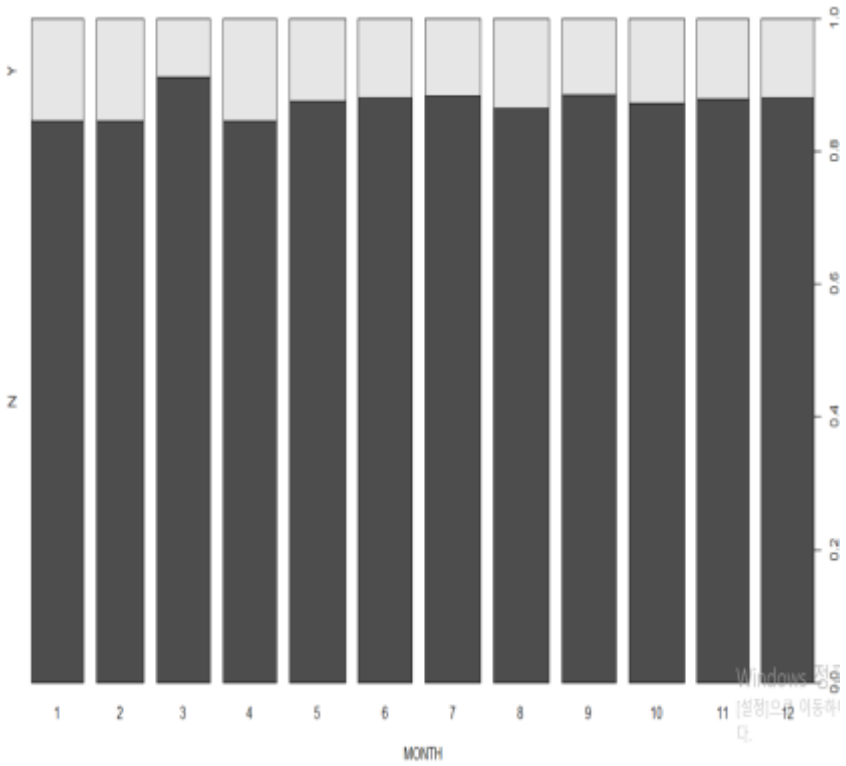
시각 별 지연 비교



6시~22시 각각의 지연횟수/총 운항 수

→ 시간별로 지연에 차이를 보이며
특히 오후 시간대에 지연율이 높음

월 별 지연 비교



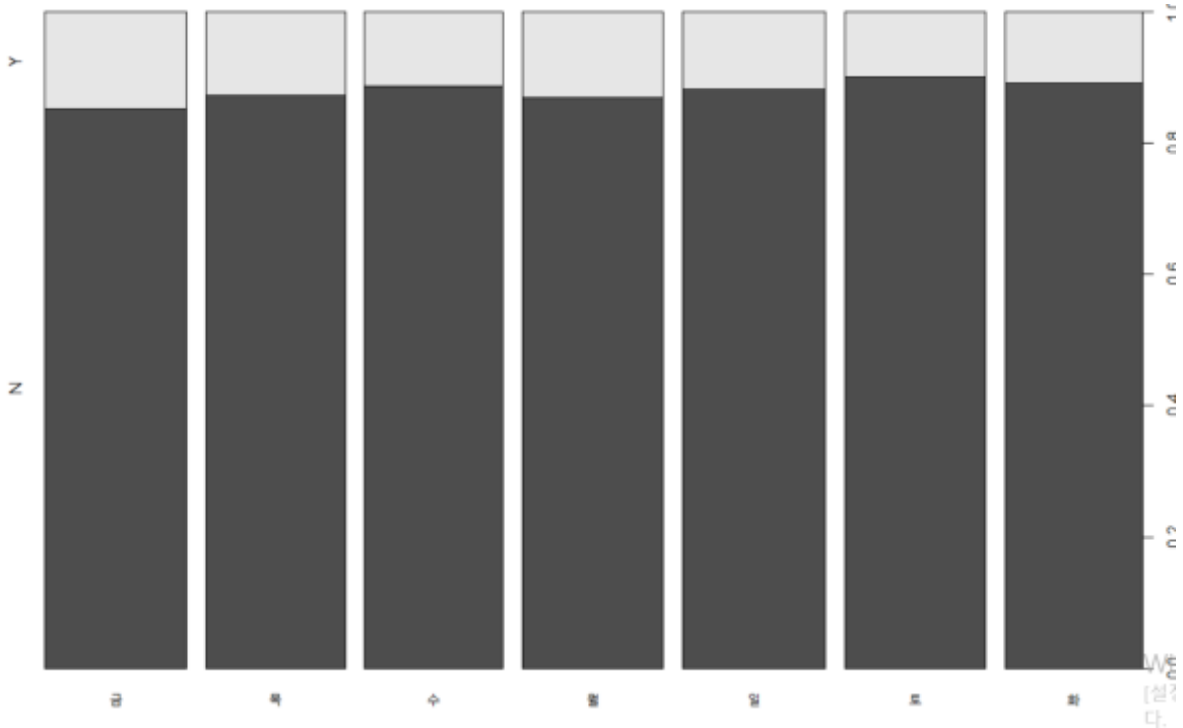
1~12월 각각의 지연횟수/총 운항 수

→ 3월에 특히 지연율이 낮음

02 Data & EDA

(2) EDA

요일 별 지연 비교



월~일요일 각각의 지연횟수/총 운항 수

→ 금요일에 지연율이 높은 편임을 확인

(2) EDA

요일 별 지연 비교

```
> prop.test(x = c(sum(FRI), sum(DY_ALL)) , n = c(sum(SDT$SDT_DY =

        2-sample test for equality of proportions with continuit

data:  c(sum(FRI), sum(DY_ALL)) out of c(sum(SDT$SDT_DY == "금"),
X-squared = 810.85, df = 1, p-value < 2.2e-16
alternative hypothesis: two.sided
95 percent confidence interval:
 0.02460042 0.02849865
sample estimates:
   prop 1    prop 2
0.1469666 0.1204170
```

```
> prop.test(x = c(sum(SAT), sum(DY_ALL)) , n = c(sum(SDT$SDT_DY

        2-sample test for equality of proportions with continuit

data:  c(sum(SAT), sum(DY_ALL)) out of c(sum(SDT$SDT_DY == "토"),
X-squared = 588.32, df = 1, p-value < 2.2e-16
alternative hypothesis: two.sided
95 percent confidence interval:
 -0.02391794 -0.02055117
sample estimates:
    prop 1     prop 2
0.09818249 0.12041705
```

```
> prop.test(x = c(sum(SUN), sum(DY_ALL)) , n = c(sum(SDT$SDT_DY

        2-sample test for equality of proportions with continuit

data:  c(sum(SUN), sum(DY_ALL)) out of c(sum(SDT$SDT_DY == "일")
X-squared = 5.9952, df = 1, p-value = 0.01434
alternative hypothesis: two.sided
95 percent confidence interval:
 -0.0040415444 -0.0004591145
sample estimates:
   prop 1    prop 2
0.1181667 0.1204170
```

```
> prop.test(x = c(sum(MON), sum(DY_ALL)) , n = c(sum(SDT$SDT_DY

        2-sample test for equality of proportions with continuit

data:  c(sum(MON), sum(DY_ALL)) out of c(sum(SDT$SDT_DY == "월")
X-squared = 130.89, df = 1, p-value < 2.2e-16
alternative hypothesis: two.sided
95 percent confidence interval:
 0.008763155 0.012515204
sample estimates:
   prop 1    prop 2
0.1310562 0.1204170
```

금, 토, 일, 월 모두 전체 지연 비율과 비교했을 때 차이를 보였다.

토요일은 예상과 다르게 지연 비율이 평균보다 더 낮게 나왔다.

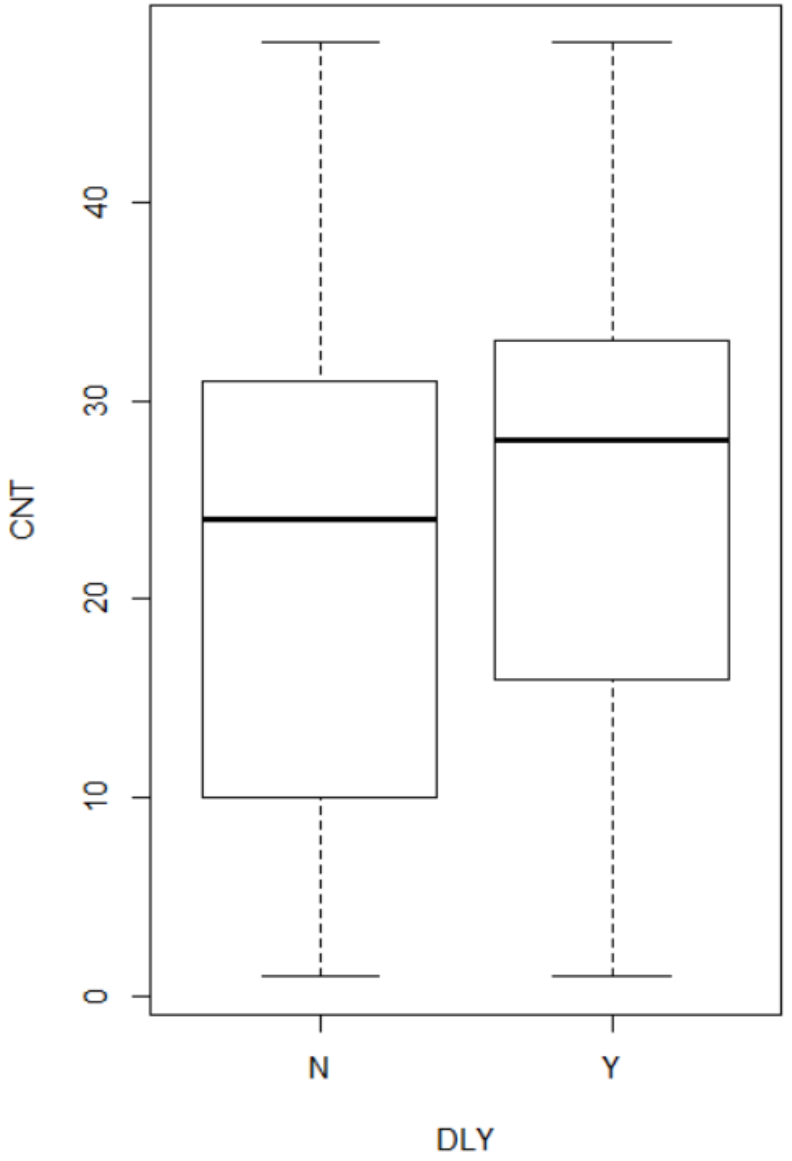
02 Data & EDA

(2) EDA

추가변수 CNT 확인

```
# welch
#만든 변수 확인
ONE <- oneway.test(CNT~DLY , data = SDT_TIMETEST , var.equal = F)
ONE
boxplot(CNT~DLY , data = SDT_TIMETEST)
```

```
        One-way analysis of means (not assuming equal variances)

data:  CNT and DLY
F = 8630.8, num df = 1, denom df = 154270, p-value < 2.2e-16
```



DLY 가 Y일 때 CNT가 크다(차이가 있다)고 볼 수 있다.

추가변수 CNT는 어느정도 유의하다고 볼 수 있다.

03 Modeling

(1) 모델링 선정

기본적 선형 분류 모델인 로지스틱 회귀 적용

```
# 로지스틱
from sklearn.linear_model import LogisticRegression

logreg = LogisticRegression(C = 1).fit(X_train , y_train)
pred_log = logreg.predict(X_test)
print("훈련 : " , logreg.score(X_train , y_train))
print("테스트 : " , logreg.score(X_test , y_test))
print(classification_report(y_test , pred_log))
```

```
훈련 :  0.8796351418300415
테스트 :  0.8795478844035508
              precision    recall  f1-score   support

           0       0.88      1.00      0.94    217173
           1       0.83      0.00      0.00     29755

    accuracy                           0.88    246928
   macro avg       0.86      0.50      0.47    246928
weighted avg       0.87      0.88      0.82    246928
```

DLY 가 1(Y)에서의 Recall(재현율) 이 0 이다.

F1-score : 0

DLY 가 1일 때의 분류가
잘 이루어지지 않을 수 있음

03 Modeling

(1) 모델링 선정

랜덤 포레스트

기본적 선형 분류 모델인 로지스틱 회귀 적용

```
# 랜덤 포레스트1

from sklearn.ensemble import RandomForestClassifier

forest = RandomForestClassifier(n_estimators = 20 , random_state = 0)
forest.fit(X_train , y_train)
pred_forest = forest.predict(X_test)
print("훈련 : " , forest.score(X_train , y_train))
print("테스트 : " , forest.score(X_test , y_test))
print(classification_report(y_test , pred_forest))
```

```
훈련 :  0.991251125501329
테스트 :  0.8794466403162056
              precision    recall  f1-score   support

           0       0.89      0.98      0.93    217173
           1       0.50      0.13      0.21     29755

    accuracy                           0.88    246928
   macro avg       0.70      0.56      0.57    246928
weighted avg       0.84      0.88      0.85    246928
```

```
행렬 :
 [[215036   2137]
 [ 27304   2451]]
```

여전히 분류에 어려움이 있음

03 Modeling

(1) 모델링 선정

```
행렬 :
 [[215036   2137]
 [ 27304   2451]]
```

→

TN	FP
FN	TP

TN : 음성클래스를 음성으로 예측

FP : 음성클래스를 양성으로 예측

FN : 양성클래스를 음성으로 예측

TP : 양성클래스를 양성으로 예측

03 Modeling

(2) Over Sampling

```
from imblearn.over_sampling import *
X_smo_t , y_smo_t = SMOTE(random_state = 0  ).fit_sample(X_train , y_train)

print(X_smo_t.shape)
print(y_smo_t.shape)
```

```
(1303198, 117)
(1303198,)
```

DLY가 1인 소수 데이터를 증가시키는 오버 샘플링을 통해 정밀도 F1-SCORE 향상 시도

03 Modeling

(3) 모델링 선정

나이브 베이즈 BernoulliNB

```
# 나이브 베이즈 이진분류
from sklearn.naive_bayes import BernoulliNB
nb = BernoulliNB(alpha = 100 , class_prior = None,fit_prior=True)
nb.fit(X_smo_t , y_smo_t)
pred4 = nb.predict(X_test)
print("훈련 : " , nb.score(X_smo_t , y_smo_t))
print("테스트 : " , nb.score(X_test , y_test))
print(classification_report(y_test , pred4))
print("₩nf1 스코어 : " , f1_score(y_test , pred4))
```

```
훈련 :  0.7063278181826553
테스트 :  0.6797366033823625
              precision    recall  f1-score   support

           0       0.93      0.69      0.79    217173
           1       0.21      0.62      0.32     29755

    accuracy                           0.68    246928
   macro avg       0.57      0.65      0.55    246928
weighted avg       0.84      0.68      0.73    246928


f1 스코어 :  0.31817633162623077
```

F1-score는 개선되었으나 정확도가 다소 떨어짐

03 Modeling

(3) 모델링 선정

랜덤 포레스트

In [16]:
```
# 랜덤 포레스트2


forest = RandomForestClassifier(n_estimators = 20 , random_state = 0)
forest.fit(X_smo_t , y_smo_t)
pred5 = forest.predict(X_test)
print("훈련 : " , forest.score(X_smo_t , y_smo_t))
print("테스트 : " , forest.score(X_test , y_test))
print(classification_report(y_test , pred5))
print("₩nf1 스코어 : " , f1_score(y_test , pred5))
```

```
훈련 :  0.9972705605748321
테스트 :  0.8630734465107238
              precision    recall  f1-score   support

           0       0.91      0.94      0.92    217173
           1       0.41      0.31      0.36     29755

    accuracy                           0.86    246928
   macro avg       0.66      0.63      0.64    246928
weighted avg       0.85      0.86      0.85    246928


f1 스코어 :  0.3556741305383516
```

Over sampling 이전보다
테스트 정확도가 약간 감소하였으나 f1-score가 향상됨

03 Modeling

(3) 모델 확정

랜덤 포레스트 2

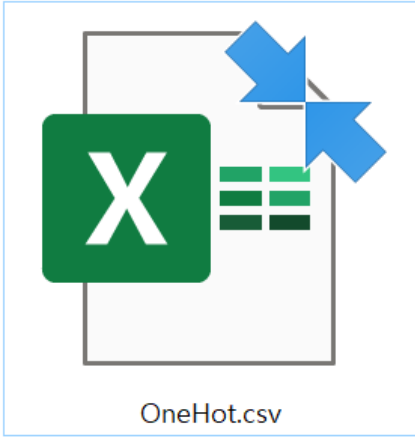
In [21]:
```
#랜덤포레스트 - class_weight = balanced

forest = RandomForestClassifier(n_estimators = 100 , random_state = 0 , max_features = 3 , max_depth=45 , class_weight = 'balanced')
forest.fit(X_smo_t, y_smo_t)
pred1 = forest.predict(X_test)
print("훈련 : " , forest.score(X_smo_t, y_smo_t))
print("테스트 : " , forest.score(X_test , y_test))
print(classification_report(y_test , pred1))
print("₩nf1 스코어 : " , f1_score(y_test , pred1))
```

```
훈련 :  0.9929933901064919
테스트 :  0.8520094926456295
              precision    recall  f1-score   support

           0       0.92      0.91      0.92    217173
           1       0.40      0.44      0.42     29755

    accuracy                           0.85    246928
   macro avg       0.66      0.67      0.67    246928
weighted avg       0.86      0.85      0.86    246928


f1 스코어 :  0.4152090767975164
```

매개변수 조정으로 테스트 정확도가 다소 떨어지나
F1-score가 큰 폭 상승 , precision , recall 균형

03 Modeling

(3) 모델 적용시키기




```
# AFSNT_DLY 원핫인코딩한 X_AFSNT_DLY 읽기

DATA2 = pd.read_pickle(r"C:/Users/dong/Desktop/R/data/X_AFSNT_DLY.pkl")

# AFASNT_DLY 읽기
DATA3 = pd.read_csv(r"C:/Users/dong/Desktop/R/data/AFSNT_DLY.csv" , encoding = "CP949" , sep=",")
```

Onehot.csv 를 통해 X_AFSNT_DLY.csv의 데이터를 이용
AFSNT_DLY.csv에 적용시킴

04 Reference

- 서적 -

안드레아스 뮐러, 세라 가이도 - 파이썬 라이브러리를 활용한 머신러닝 , 한빛미디어

- 인터넷 -

https://scikit-learn.org/