✈ 항공 지연 예측 모델링

Team: Gotica
(공해인, 권나현, 임승연, 신오석)

INDEX

1. 개요

늘어나는 항공기 지연... 얼마나 비용 초래할까 [뉴스+]

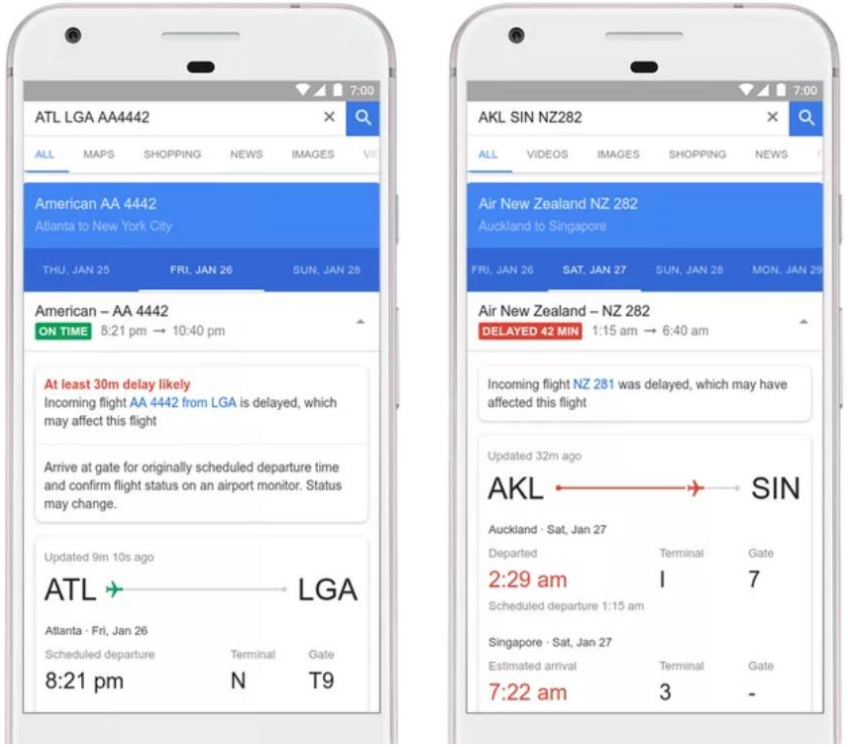
공중 지연으로 인한 피해는 항공사와 승객 모두에게 돌아간다.

항공사 입장에서는 유류비, 연결편 항공기 지연 등 피해가 발생하며, 승객의 입장에서는 지연에 따라 일정 차질 등이 일어날 수 있기 때문이다.

이에 연구팀은 2016년 국내 항공사 항공유 구입비를 통해 공중 지연에 따른 항공사의 유류비도 분석했는데, 2018년 1~6월 공중 지연에 따른 유류비는 약 5억5600만원으로 추정됐다.

이를 연간으로 환산하면 약 11억1300만원이다.

Google is now using machine learning to predict flight delays



2. 데이터 탐색

데이터 소개

- 훈련 데이터: AFSNT.csv (행: 987709, 열:17)

  2017.1.1 – 2019.6.30 운항한 항공편의 실적 데이터

- 테스트 데이터: AFSNT_DLY.csv (행: 283835, 열:11)

  2019.9.16 – 2019.9.30 운항하는 국내선 항공편에 대한 데이터

  훈련 데이터에 있는 REG, IRR, CNR, CNL, DRR, IRR, ATT이 없음

- 외부 데이터: 한국공항공사 공항 별 통계

  국내선, 정기운항, 전체 공항 대상으로 2017-2018년도 운항(편수)와 여객(명) 데이터

이상치 제거: 결항 데이터 제거

- 대회 문제는 항공 지연을 예측하는 것으로 결항 데이터를 이상치로 간주하여 제거
- 전체 데이터 987,709개 중 결항 데이터 8,259개 제거
- 제거 후 데이터 수 : 979,450개

SDT_YY	SDT_MM	SDT_DD	SDT_DY	ARP	ODP	FLO	FLT	REG	AOD	IRR	STT	ATT	DLY	DRR	CNL	CNR
2017	1	10	화	ARP13	ARP3	J	J1921	NaN	A	N	14:00	0:00	N	NaN	Y	A02
2017	1	10	화	ARP13	ARP3	J	J1920	NaN	D	N	14:35	0:00	N	NaN	Y	A02
2017	1	10	화	ARP3	ARP13	J	J1921	NaN	D	N	13:10	0:00	N	NaN	Y	A02
2017	1	10	화	ARP3	ARP13	J	J1920	NaN	A	N	15:25	0:00	N	NaN	Y	A02
2017	1	13	금	ARP13	ARP3	J	J1920	NaN	D	N	14:35	0:00	N	NaN	Y	A02

8259 rows × 17 columns

이상치 제거: 부정기편 제거

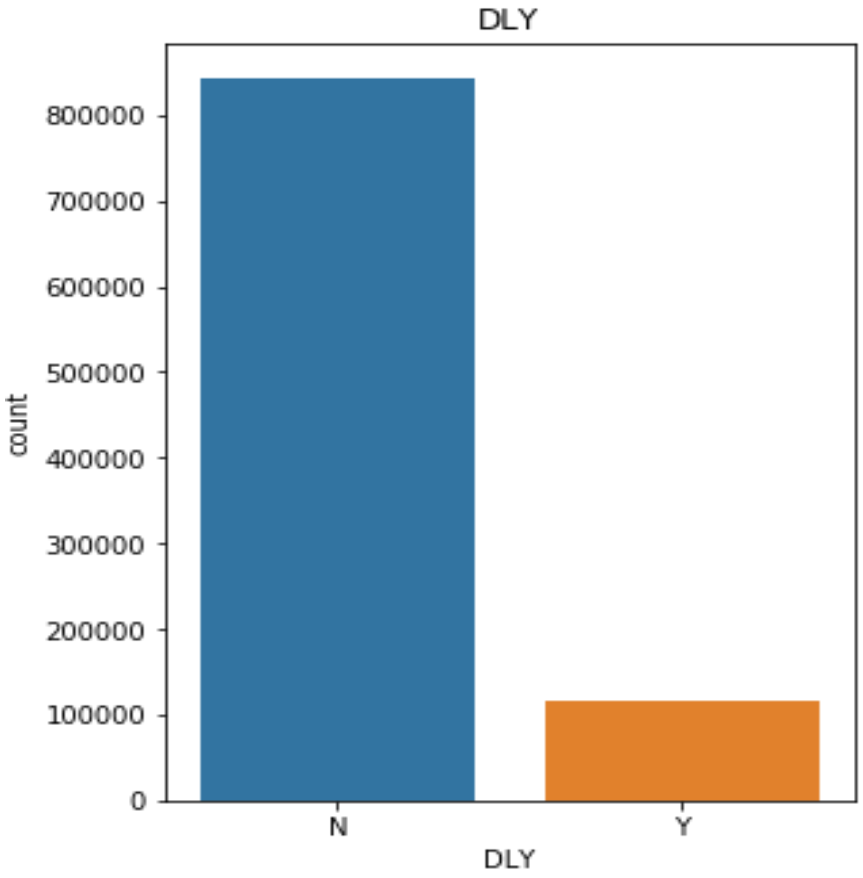
- 부정기편은 정기편과 달리 일정한 주기없이 특별한 목적에 따라 부정기적으로 운항하는 항공편이기 때문에 예측 대상이 아님
- 전체 데이터 979,450개 중 부정기편 21,300개 제거
- 제거 후 데이터 수 : 958,150개

SDT_YY	SDT_MM	SDT_DD	SDT_DY	ARP	ODP	FLO	FLT	REG	AOD	IRR	STT	ATT	DLY	DRR	CNL	CNR
2017	1	1	일	ARP2	ARP15	J	J1465	SEw3NTI2	D	Y	10:40	10:22	N	NaN	N	NaN
2017	1	2	월	ARP1	ARP15	J	J1464	SEw3NDYw	D	Y	6:20	6:48	N	NaN	N	NaN
2017	1	2	월	ARP1	ARP15	J	J1463	SEw3NzY1	A	Y	8:15	8:17	N	NaN	N	NaN
2017	1	6	금	ARP1	ARP3	J	J1256	SEw3NzI2	D	Y	10:50	11:05	N	NaN	N	NaN
2017	1	6	금	ARP1	ARP3	J	J1259	SEw3NzI2	A	Y	13:50	13:52	N	NaN	N	NaN

21300 rows × 17 columns

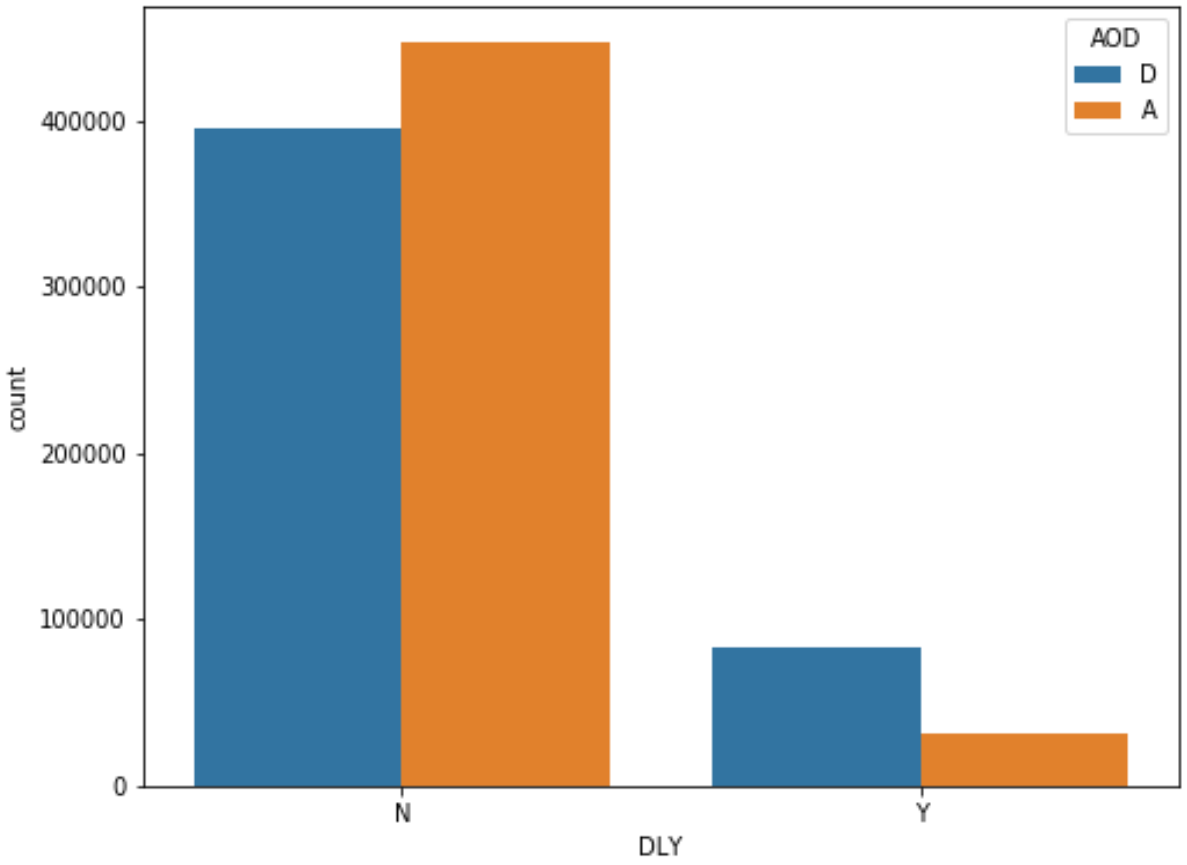
3. 데이터 시각화

지연 여부



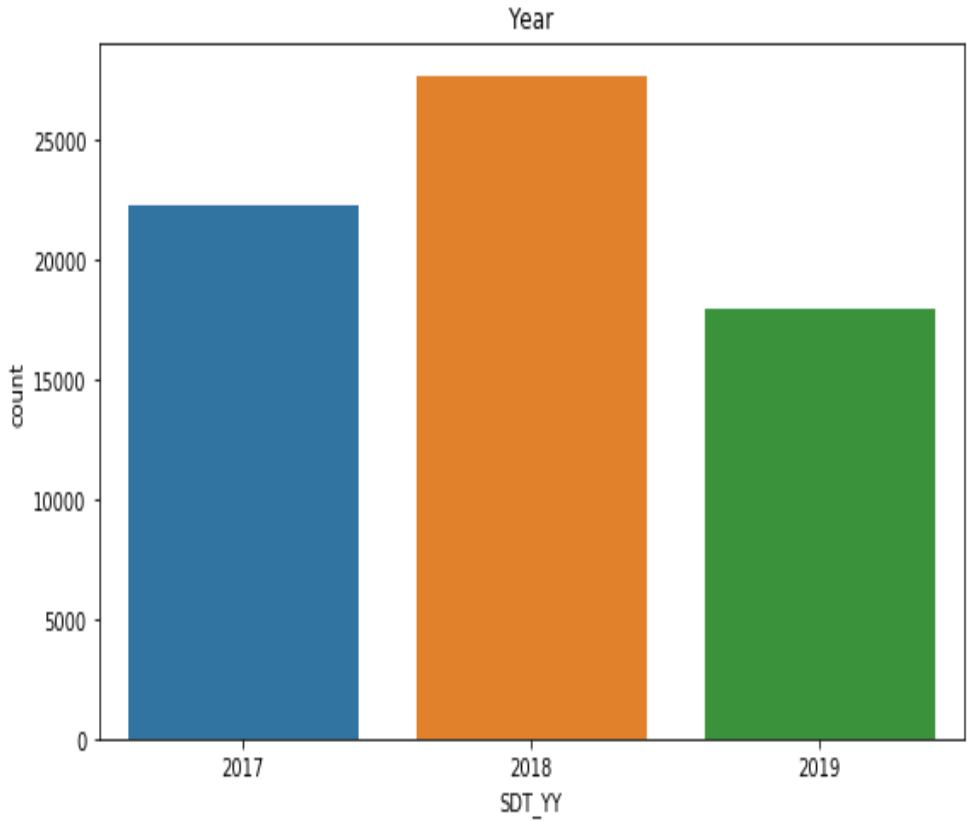
종속 변수인 'DLY' 는 상당히 불균형함

출도착에 따른 지연 여부



출발 지연이 도착 지연보다 더 많음

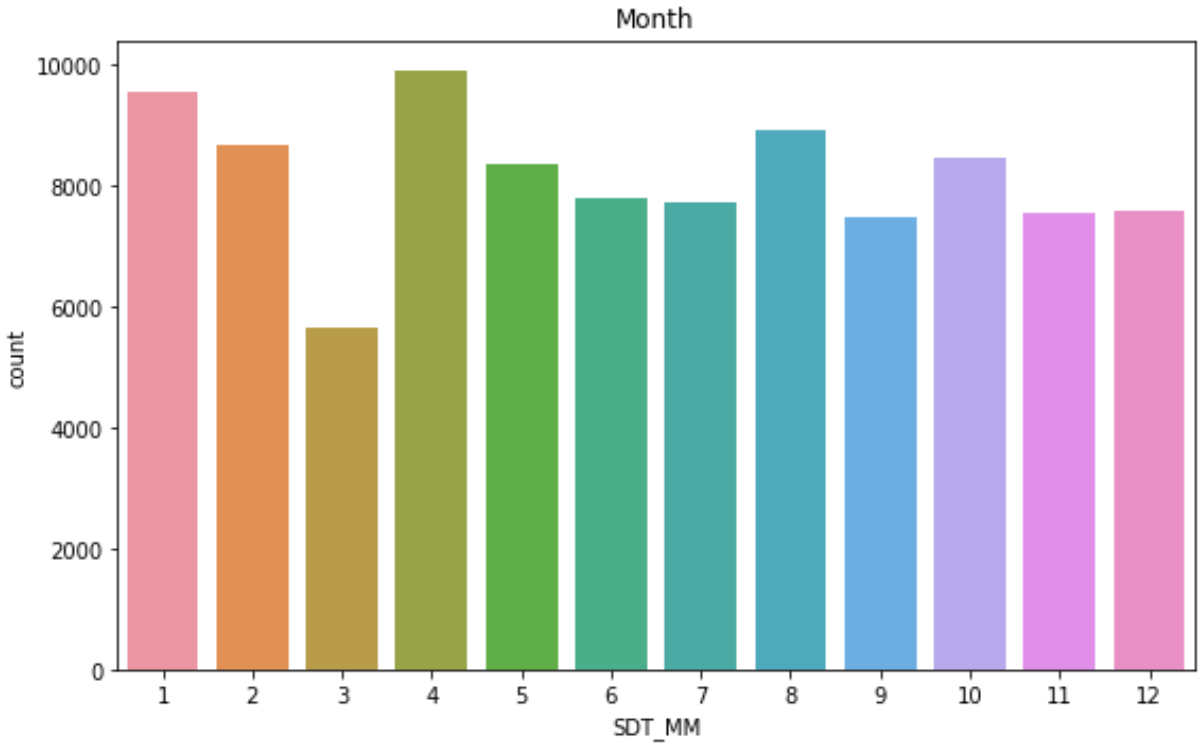
연도



19년은 상반기 밖에 없어서 정확한 비교를 위해 상반기만 비교

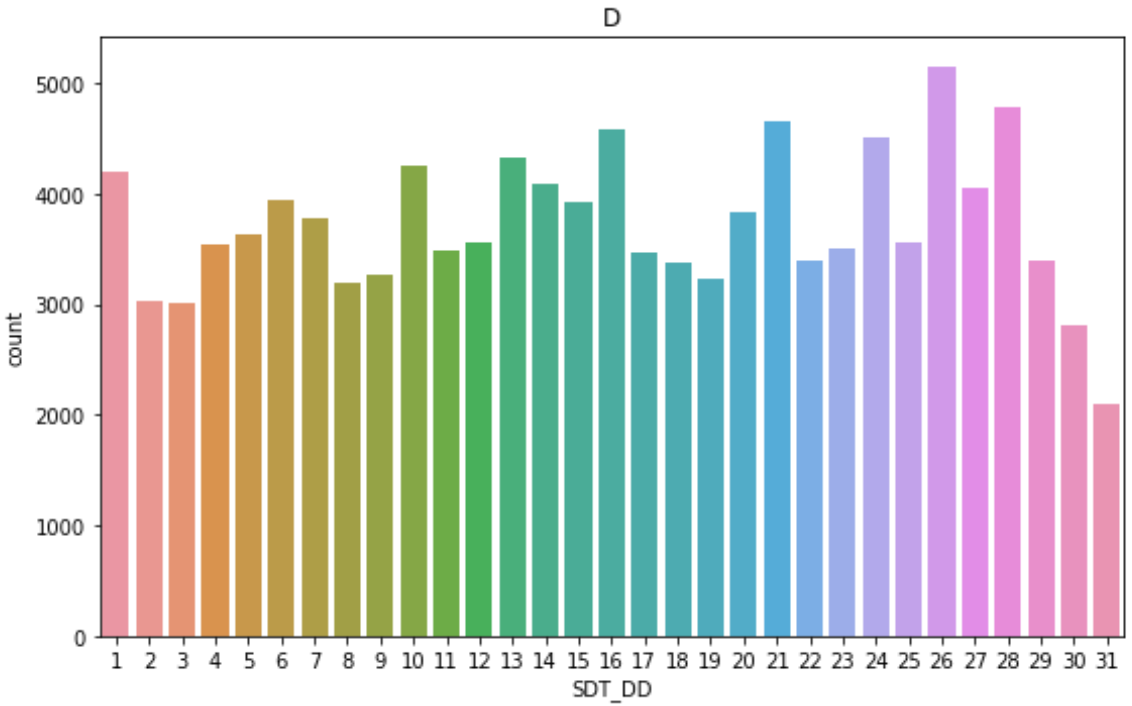
17년과 18년에 비해 19년은 확연히 지연이 줄어듬

월



3월이 가장 지연이 적음

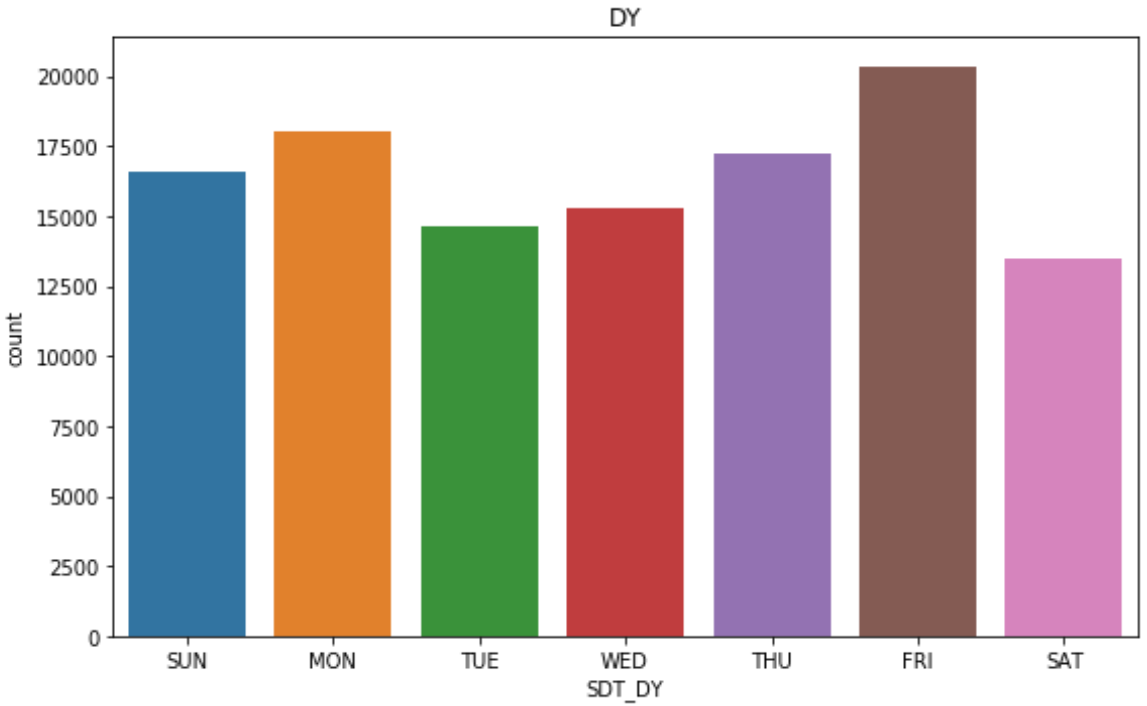
일



상당히 상이하게 지연이 분포되어 있어서 뚜렷한 특징이 없음

하지만 말일로 갈수록 지연이 적어짐

요일

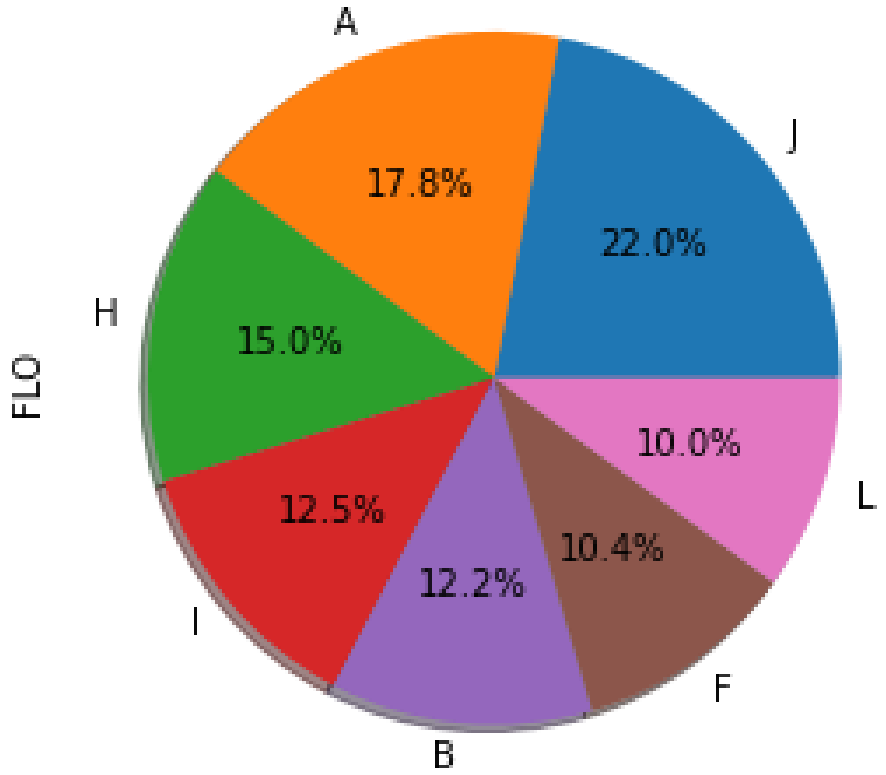


금요일이 가장 많이 지연되고

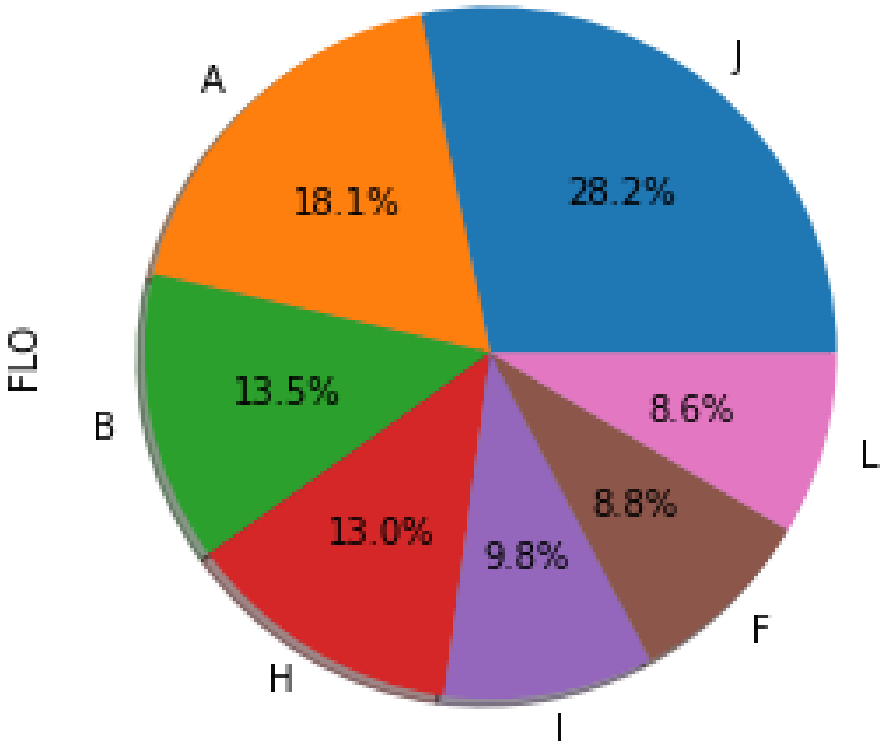
이에 반해 토요일이 가장 적음

항공사

원 데이터



지연인 데이터



지연된 것의 그래프와 원 데이터의 그래프를 비교해보면

지연율이 미미하게 조금 더 높은 항공사가 있음을 확인할 수 있음

	원 데이터	Value_count	지연인 데이터	Value_count	원 데이터 Value_count
FLT	J1959	1820	B1021	677	1814
	H1115	1820	F1217	658	1814
	L1805	1820	F1220	590	1800
	A1234	1820	I1566	574	1776
	H1117	1820	I1324	549	1604

각각 상위 5개 항목을 보았을 때 겹치는 것이 아예 없을 정도로 상이한 결과

항공사와 노선의 정보를 포함하고 있는 FLT가 빈도 수에 따라서 지연이 높게 나오는 것이 아님

4. 데이터 전처리

ARP(공항), ODP(상대 공항)

Test 데이터에 있는 오타 난 데이터
ARP1 로 변환 (FAQ 참고)

ARP
ARP1
ARP3
`
ARP3
ARP4

ARP
ARP1
ARP3
ARP1
ARP3
ARP4

ARP 데이터의 기존 형태에서
숫자만 남겨서 라벨링

ARP
ARP3
ARP3
ARP3
ARP3
ARP3

ARP
3
3
3
3
3

STT(계획 시각)

STT
10:05
9:30
12:45
13:25
16:10

STT_hour	STT_min
10	5
9	30
12	45
13	25
16	10

STT를 시, 분으로 나누어 int 형식으로 변환

공항 유추

```
route 1 --> 2   :        51006
route 1 --> 3   :        217671
route 1 --> 4   :        790
route 1 --> 5   :        12025
route 1 --> 8   :        3823
route 1 --> 9   :        7469
route 1 --> 11   :       3413
route 1 --> 12   :       3534
route 2 --> 1   :        50973
route 2 --> 3   :        57236
route 2 --> 15   :       9817
route 3 --> 1   :        217647
route 3 --> 2   :        57254
route 3 --> 4   :        27814
route 3 --> 5   :        3319
route 3 --> 6   :        33416
route 3 --> 7   :        3014
route 3 --> 8   :        28967
route 3 --> 9   :        5030
route 3 --> 12   :       1206
route 3 --> 13   :       4100
route 3 --> 14   :       1820
```

시작 연도 2017 | 시작 월 1 | 종료 연도 2019 | 종료 월 6

노선 구분 대구 ~ 전체 | 운항 구분 전체

여객화물 구분 전체 | 여객 구분 전체 | 화물 구분 전체

확 인

엑셀파일

구분			운항(편수)			여객(명)			화물(ton)		
노선	항공사코드	항공사명	도착	출발	계	도착	출발	계	도착	출발	계
대구 - 제주	AAR	아시아나항공	2,724	2,722	5,446	405,388	422,656	828,044	5,035.9	6,002.1	11,038
	ABL	에어부산	1,871	1,870	3,741	339,708	333,861	673,569	1,849.4	1,582.1	3,431.5
	JJA	제주항공	2,176	2,173	4,349	397,245	396,443	793,688	2,080	1,713.8	3,793.8
	KAL	대한항공	2,896	2,890	5,786	410,928	410,567	821,495	5,686.3	4,474.1	10,160.4
	TWB	티웨이항공	4,527	4,529	9,056	825,704	822,540	1,648,244	4,181.8	3,591.7	7,773.6
	소 계		14,194	14,184	28,378	2,378,973	2,386,067	4,765,040	18,833.4	17,363.8	36,197.2
대구 - 김포	ABL	에어부산	432	429	861	39,065	56,382	95,447	66.4	101.5	167.8
	KAL	대한항공	5	2	7	336	13	349	1.3	0.1	1.3
	소 계		437	431	868	39,401	56,395	95,796	67.6	101.5	169.2
대구 - 인천	KAL	대한항공	1,750	1,748	3,498	122,251	127,146	249,397	2,236.8	2,216.6	4,453.4
합 계			16,381	16,363	32,744	2,540,625	2,569,608	5,110,233	21,137.9	19,682	40,819.8

< 결과 >

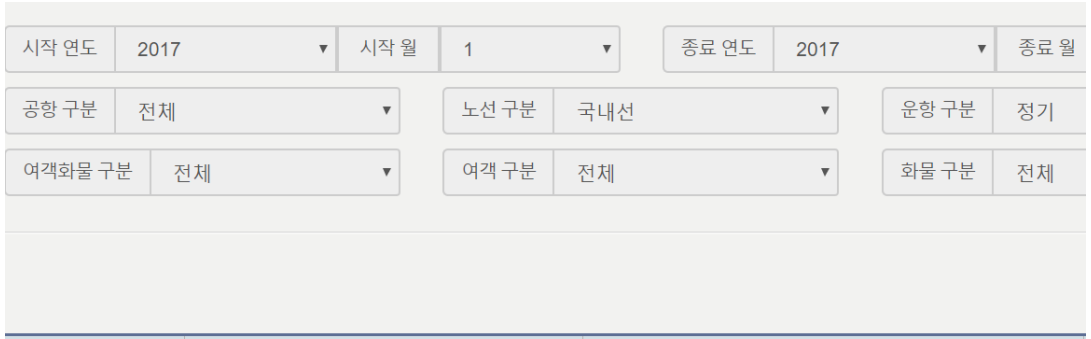
ARP1 = 김포
ARP2 = 김해
ARP3 = 제주
ARP4 = 대구
ARP5 = 울산
ARP6 = 청주
ARP7 = 무안
ARP8 = 광주
ARP9 = 여수
ARP10 = 양양
ARP11 = 포항
ARP12 = 사천
ARP13 = 군산
ARP14 = 원주
ARP15 = 인천공항

공항 별로 노선을 확인하고

한국공항공사에서 제공하는 국내선 노선별 통계를 이용하여 노선과 이용 빈도수를 고려하여 공항을 유추

공항 별 평균 운항(편수), 여객(명)

- 공항 별 운항(편수), 여객(명)은 **공항의 혼잡도**를 나타낼 수 있는 지표
- 2017-18년도 운항(편수)와 여객(명) 데이터를 토대로 공항 별 평균 운항과 여객 변수 생성
- 추정한 ARP 에 맞춰 훈련과 테스트 데이터에 평균 운항, 평균 여객 변수 추가

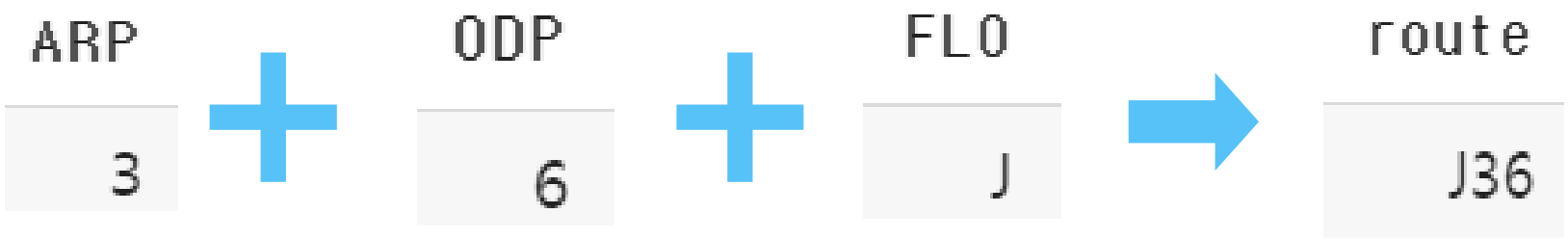


공항명	운항(편수)			여객(명)		
	도착	출발	계	도착	출발	계
김포	61,465	61,475	122,940	10,383,018	10,316,588	20,699,606
김해	24,076	24,119	48,195	3,732,992	3,754,680	7,487,672
제주	77,302	77,327	154,629	13,896,660	13,933,868	27,830,528
대구	6,342	6,297	12,639	1,005,473	1,006,291	2,011,764

공항명	2017 여객	2018 여객	여객 평균	2017 운항	2018 운항	운항 평균
김포	20,699,606	19,950,151	20324878.5	122,940	118,620	120780
김해	7,487,672	7,109,005	7298338.5	48,195	46,852	47523.5
제주	27,830,528	27,091,000	27460764	154,629	153,036	153832.5
대구	2,011,764	1,974,269	1993016.5	12,639	13,048	12843.5
울산	553,333	759,987	656660	5,227	6,784	6005.5

Route(항공사별 경로)

- ARP, ODP, FLO(공항, 상대공항, 항공사) 변수를 이용하여 생성
- 출,도착으로 분리되어 있는 하나의 항공 데이터를 동일하게 취급
  - ARP와 ODP를 비교하여 작은 공항 번호를 앞으로 배치
- 항공사별 경로 표현 위해 항공사 알파벳 추가



5. 모델링

최종 모델에 사용된 변수

- SDT_YY
- SDT_MM
- SDT_DD
- SDT_DY
- ARP
- ODP
- AOD
- STT_hour
- STT_min
- FLT
- route
- pas (평균 여객)
- num (평균 운항)

제외된 변수

- REG
- ATT
- STT
- CNL
- CNR
- IRR
- DRR
- STT
- FLO

∴ 총 13개의 변수 사용

분석에 사용된 후보 모델

▪ Catboost	범주형 변수에 특화된 모델
▪ Catboost under-sampling	클래스 불균형을 맞추는 모델
▪ Random forest	트리 기반 모델
▪ Light GBM	그래디언트 부스팅 기반 모델

☞ 모델 간 AUC 비교를 통해 최종 모델 결정

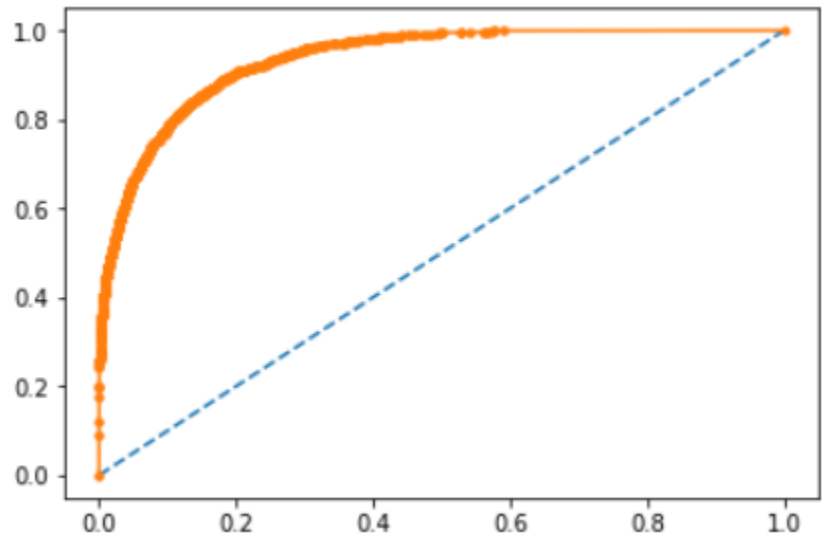
CatBoost

Gradient Boosting Algorithm을 강화하는 범주형 기능 지원 라이브러리

- **Category 변수에 대한 전처리 문제를 해결**
  - One-hot-encoding 처리를 훈련 도중에 진행하는 것이 더 빠르다는 점을 착안
  - 범주형 변수를 자동으로 처리해 타입 변환 오류를 피할 수 있음
- **One-hot-max-encode를 이용하여 범주형과 수치형 변수들의 combination을 처리**
  - 변수 간의 상관관계를 계산함과 동시에 속도 개선
  - Multiple-category 데이터를 다룰 때 유용

Catboost 코드

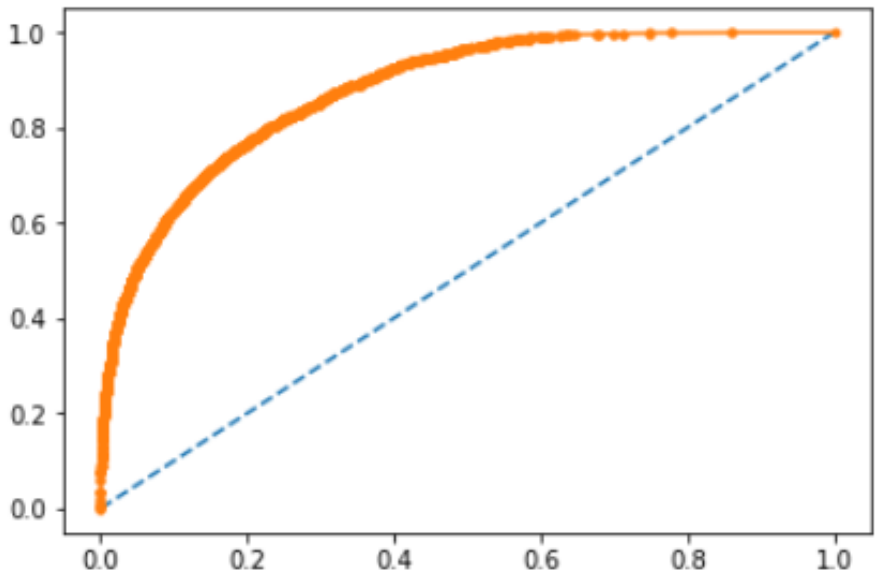
```
# data, target 지정
x = data.drop(['DLY'], axis = 1)
y = data['DLY']
y = y.replace(['N','Y'],[0,1])
np.random.seed(1)

# category_features 지정
cat_features = ['SDT_YY','SDT_MM','SDT_DD','SDT_DY','ARP','ODP','AOD',#
                'STT_hour','route','STT_min','FLT']

x = data.drop(['DLY'], axis = 1)
y = data['DLY']
y = y.replace(['N','Y'],[0,1])

train_x,eval_x,train_y,eval_y = train_test_split(x,y,test_size=0.016)

#모델 지정
ml_cb =  CatBoostClassifier(random_seed=1234,iterations=3000,depth=8,#
          loss_function='CrossEntropy',eval_metric=['AUC'])
#모델 학습
ml_cb_output = ml_cb.fit(train_x,train_y,cat_features = cat_features,#
         early_stopping_rounds=150,eval_set=(eval_x,eval_y),use_best_model=True)
```



Catboost(under-sampling)코드

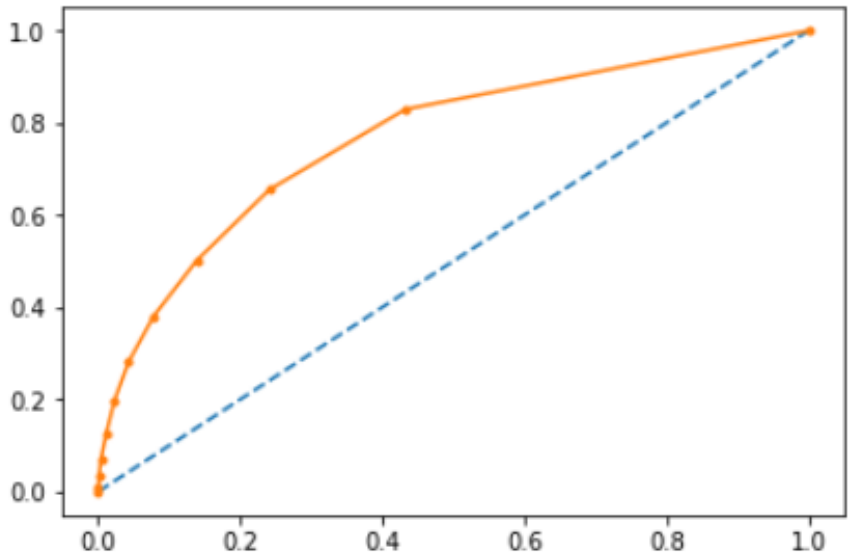
Imblearn의 RandomUnderSampler 사용

```
from imblearn.under_sampling import *

under_data = data
under_x = under_data.drop(['DLY'], axis = 1)
under_y = under_data['DLY']
under_y = under_y.replace(['N','Y'],[0,1])
train_x_ru,test_x_ru,train_y_ru,test_y_ru = train_test_split(under_x,under_y,₩
                                                             test_size=0.016)

train_x_r,train_y_r=RandomUnderSampler(random_state=0).fit_sample(train_x_ru,₩
                                                                  train_y_ru)
cat_features = [0,1,2,3,4,5,6,7,8,9,12]


ml_cb =  CatBoostClassifier(random_seed=1234,iterations=3000,depth=8,₩
                            loss_function='CrossEntropy')

ml_cb_output = ml_cb.fit(train_x_r,train_y_r,cat_features = cat_features)
pred_ru = ml_cb_output.predict_proba(test_x_ru)
results_ru = ml_cb_output.predict(test_x_ru)
```



Random forest 코드

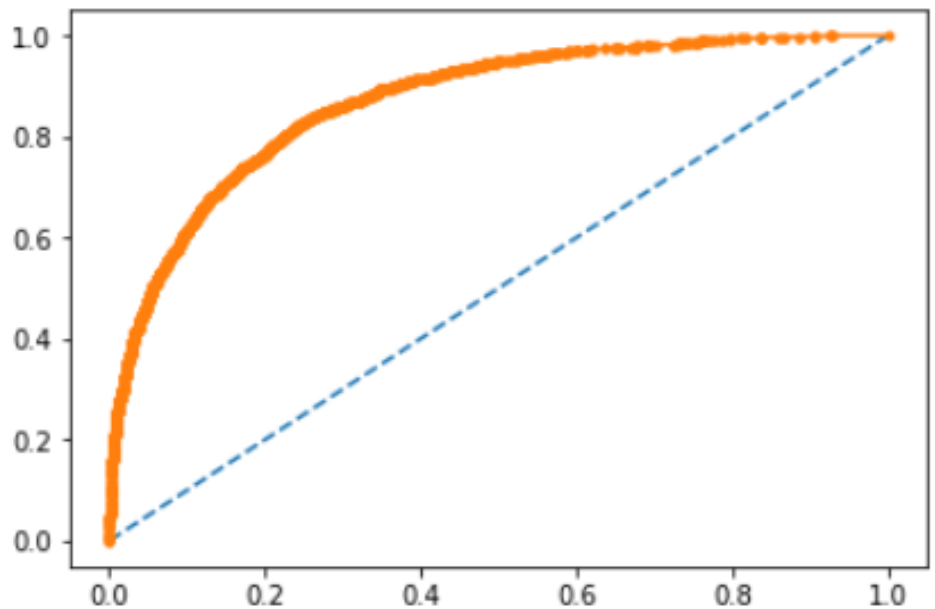
```
import category_encoders as ce
from sklearn.preprocessing import LabelEncoder

random_data = data

oe = ce.OrdinalEncoder(cols=['SDT_DY','FLT','AOD','route'])

encoded_data = oe.fit_transform(random_data)
random_x = encoded_data.drop(['DLY'], axis = 1)
random_y = encoded_data['DLY']
random_y = random_y.replace(['N','Y'],[0,1])

np.random.seed(1)
train_random_x,test_random_x,train_random_y,test_random_y =
train_test_split(random_x,random_y,test_size=0.016)

from sklearn.ensemble import RandomForestClassifier
ml = RandomForestClassifier(criterion='entropy',n_estimators=10)
ml.fit(train_random_x,train_random_y)

pred_rd = ml.predict_proba(test_random_x)
results_rd = ml.predict(test_random_x)
```



Light GBM 코드

Gradient Boosting Algorithm

```
import lightgbm as lgb

train = data.drop(['DLY'], axis = 1)
label = data['DLY']
label = label.replace(['N','Y'], [0,1])

from sklearn.metrics import roc_auc_score
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn import svm

params = {"learning_rate" : 0.01,
          "max_depth" : 12,
          "bagging_fraction" : 0.75,
          'feature_fraction' : 0.4,
          'objective': 'cross_entropy',
          'boosting_type': 'gbdt',
          'metric' : {'auc'},
          'device' : 'cpu',
          'reg_alpha':0.1,
          'verbose' : 1}

cat_features =['SDT_YY','SDT_MM','SDT_DD','SDT_DY','ARP','ODP','AOD',#
               'STT_hour','route','STT_min','FLT']

for i in enumerate (cat_features) :
  ca = i[1]
  train[ca] = train[ca].astype('category')

Xtrain, Xtest, ytrain,ytest = train_test_split(train,label,test_size=0.016)

lgb_train = lgb.Dataset(Xtrain, ytrain)
lgb_eval = lgb.Dataset(Xtest, ytest)

clf = lgb.train(params, lgb_train, num_boost_round=100000,#
                valid_sets=lgb_eval,early_stopping_rounds=100,verbose_eval=200)
```



모델 간 성능 비교

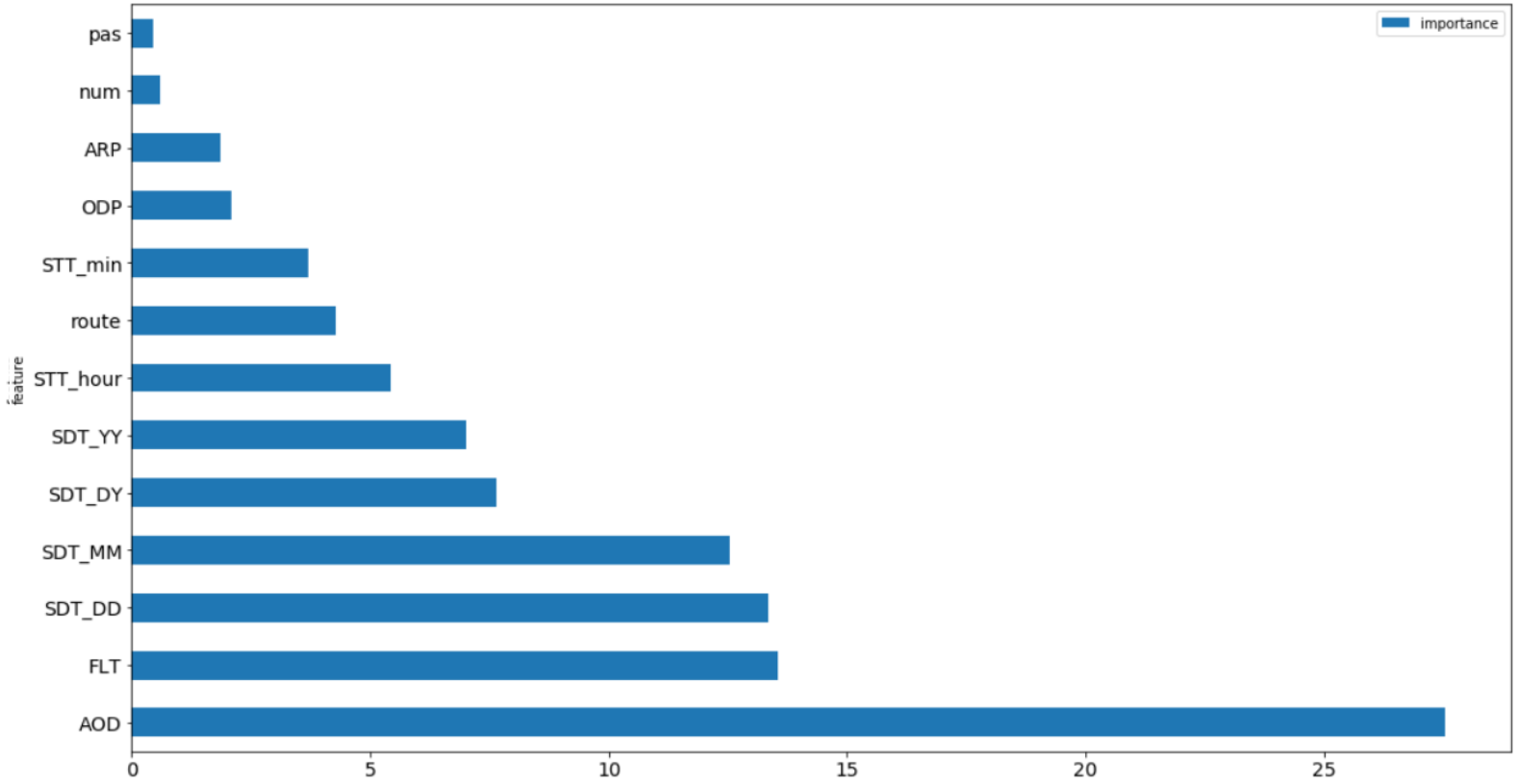
	Catboost	Catboost (under-sampling)	Light GBM	Random Forest
AUC Score	0.932	0.881	0.858	0.766
Recall	0.475	0.612	0.218	0.197
Precision	0.790	0.469	0.709	0.536

AUC 성능이 가장 좋은 Catboost 를 최종 모델로 선정

6. 결론

- Catboost 모델의 변수 중요성

	feature	importance
0	AOD	27.538480
1	FLT	13.536948
2	SDT_DD	13.335065
3	SDT_MM	12.535257
4	SDT_DY	7.636493
5	SDT_YY	7.012797
6	STT_hour	5.419226
7	route	4.279253
8	STT_min	3.706356
9	ODP	2.096000
10	ARP	1.850883
11	num	0.603600
12	pas	0.449644



- 항공 지연을 예측하는 데 중요한 상위 5개 변수

1. 출도착
2. 편명
3. 일
4. 월
5. 요일

참 고 자 료

- 나기천, 늘어나는 항공기 지연… 얼마나 비용 초래할까 [뉴스+], 세계일보, 2019.1.12
- 한국공항공사 항공 통계
- 항공위키 부정기편
- James Vincent, Google is now using machine learning to predict flight delays, The verge, 2018.1.31
- Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., & Gulin, A. (2018). CatBoost: unbiased boosting with categorical features. In *Advances in Neural Information Processing Systems* (pp. 6638-6648).
- Dorogush, A. V., Ershov, V., & Gulin, A. (2018). CatBoost: gradient boosting with categorical features support. *arXiv preprint arXiv:1810.11363*.