

# 2023 설 특송기간 승객 수요예측 프로젝트 보고서



2023. 1. 4 - 2023. 1. 19

데이터분석 수련생

박찬성 이재은

# 목 차

## 1. 서론

1.1 분석 개요	...	p.3
1.2 분석 프로세스	...	p.3

## 2. 본론

2.1 데이터 수집	...	p.3
2.2 데이터 전처리	...	p.4
2.3 분석 모델링	...	p.8

## 3. 결론

3.1 분석 결과	...	p.10
3.2 향후 과제	...	p.11

## 4. 부록

4.1 데이터 명세	...	p.12
4.2 코드 명세	...	p.13

# 1. 서론

## 1.1 분석 개요

### 1.1.1 분석 배경 및 필요성

매년 돌아오는 대명절 설과 추석에는 특히 공항에 대한 수요가 단기간에 집중되는 경향이 있다. 이에 따라 공항 이용객에게 더 나은 서비스를 제공함과 동시에 혹시 모를 사고의 예방을 위해, 일자별 그리고 시간대별로 정확한 승객 수요량을 예측할 필요가 있다. 이러한 작업을 매년 수행하고 있으나, 모델을 활용한 데이터 분석적인 접근보다 사람의 감을 중심으로 예측해왔기에 그 필요성이 더욱 강조되었다.

### 1.1.2 분석 목표

이에 본 프로젝트는 2023년 설을 앞두고 해당 특송기간에 얼마나 많은 승객이 몰릴 것인지를 예측하고자 하였고, 가장 많이 붐비는 일자와 시간대를 도출함으로써 공항 서비스 및 운영과 관련된 여러 의사 결정에 도움을 주는 것이 최종 목표이다.

## 1.2 분석 프로세스



[그림 1] 프로젝트 분석 프로세스

본 프로젝트의 분석 프로세스는 상단의 그림 1처럼, 데이터 수집, 데이터 전처리, 분석 모델링, 결과 도출 순으로 진행되었다. 즉, 필요한 원천 데이터를 수집하고 전처리 과정을 진행한 후, 해당 데이터를 바탕으로 모델링 작업을 수행해 최종 예측값을 얻었다. 이후 분석 결과를 집계함으로써 최종 완성하였다.

# 2. 본론

## 2.1 데이터 수집

수집한 데이터는 총 두 가지로, 항공 운항 데이터와 특송기간 데이터이다. 두 개의 데이터를 결합하고 전처리 과정을 거쳐 학습 데이터를 완성하였다.

### 2.1.1 항공 운항 데이터

한국공항공사의 내부 데이터로, 2010년 1월 1일부터 2022년 9월 30일까지 김포공항

내의 일일 항공 운항 데이터이다. 해당 데이터는 '운항 일자', '출발도착구분코드', '공항코드', '좌석수', '화물' 등 항공편별 운항 정보를 담고 있다. 그중 필요한 열인 '실제운항일자', '출발도착구분코드', 'IATA공항코드', '실제운항시분', '좌석수', 'IATA항공사코드', '운항편명', '상대IATA공항코드', '총유임승객수', '총무임승객수', '노선구분코드'만 추출하였고, 해당 데이터를 바탕으로 항공편별 '총승객수'와 '운항시', '주야여부' 그리고 '탑승률'이라는 파생 변수를 생성하였다.

### 2.1.2 특송기간 데이터

동일하게 한국공항공사의 내부 데이터로, 2003년 추석 연휴부터 2023년 설 연휴 동안의 특송기간을 명시하는 데이터이다. 해당 데이터는 특송기간의 '종류'와 '시작일자' 그리고 '종료일자'를 담고 있다. 이를 통해 일자별로 '운항일자'와, 해당 일자가 특송기간의 몇 번째 날인지를 의미하는 'n번째날', 그리고 해당 특송 기간의 길이를 뜻하는 '특송기간'이라는 파생 변수를 추가로 생성하였다.

## 2.2 데이터 전처리

### 2.2.1 특송기간 데이터

특송기간 데이터의 경우 앞서 2.1.2에서 언급했듯이 시작일자와 종료일자로 표현돼 있기에, 이를 일 단위로 표현하는 형태로 수정하고자 하였다.

#### ◆ 특송기간 추출 및 파생 변수 생성

우선 시작일자와 종료일자를 int에서 datetime으로 형 변환 시켜준 다음, 각각의 행별로 즉 특송기간별로 실제 날짜들을 추출하는 작업을 수행하였다. 이를 '실제운항일자'로 설정한 후, 일자별로 그날이 해당 특송기간의 몇 번째 날인지를 의미하는 'n번째날'과 특송기간의 길이를 뜻하는 '특송기간'을 추출하였다.

특송기간코드	시작일자	종료일자	종류	시작일자	종료일자	실제운항일자	n번째날	특송기간	종류
1	CH	20030909	20030915	1	CH	2003-09-09	2003-09-15	7	CH
3	CH	20040924	20040930	3	CH	2004-09-24	2004-09-30	7	CH
4	SL	20050207	20050211	4	SL	2005-02-07	2005-02-11	5	SL
6	CH	20050916	20050920	6	CH	2005-09-16	2005-09-20	5	CH
7	SL	20060127	20060131	7	SL	2006-01-27	2006-01-31	5	SL
...	...	...	...	...	...	...	...	...	...
75	SL	20210208	20210213	75	SL	2021-02-08	2021-02-13	6	SL
77	CH	20210917	20210922	77	CH	2021-09-17	2021-09-22	6	CH
78	SL	20220128	20220202	78	SL	2022-01-28	2022-02-02	6	SL
80	CH	20220908	20220912	80	CH	2022-09-08	2022-09-12	5	CH
82	SL	20230120	20230124	82	SL	2023-01-20	2023-01-24	5	SL

[그림2] 특송기간 날짜 데이터 및 파생 변수 생성

## 2.2.2 운항 데이터

학습 데이터의 기준이 되는 데이터로, 특송기간을 추출하면서 각종 파생 변수를 다양하게 생성하여 학습 데이터를 완성하였다.

### ◆ 실제운항일자 전처리

특송기간을 추출하기 위해 먼저 '실제운항일자'를 기준으로 null 값과 데이터 형태 등을 정리하였다. 이때 날짜가 '20200101'처럼 여덟 자로 되어있지 않고 '200101'처럼 앞의 '20'이 없는 데이터가 존재하여 이를 처리해주었다.

실제운항일자	실제운항일자
0	7368
20190915	1542
20190914	1539
20190807	1533
20190923	1532
...	...
120928	1
100923	1
100806	1
130728	1
141228	1

Name: 실제운항일자, Length: 4672, dtype: int64

20190915.0	1542
20190914.0	1539
20190807.0	1533
20190803.0	1532
20190923.0	1532
...	...
20210107.0	200
20210108.0	180
20120828.0	177
20200902.0	82
20200826.0	56

Name: 실제운항일자, Length: 4656, dtype: int64

[그림3] 실제운항일자 전처리

### ◆ 특송기간 추출

이후 정리된 특송기간 데이터와 운항 데이터를 join 시킴으로써 특송기간에 해당하는 데이터만을 추출하였다. 그리고 최종 목표는 2023년도 설 특송기간의 수요를 예측하는 것이기에, 지난 5년간 데이터를 학습시키고자 하면서 코로나 기간은 배제하였다. 따라서 2017년부터 2020년을 제외한 2022년까지를 추출하였다.

	실제운항일자	출발도착구분코드	IATA공항코드	실제운항시분	좌석수	IATA항공사코드	운항편명	상대IATA공항코드	총유임승객수	총무임승객수	노선구분코드
95077	2017-01-26	A	KWJ	842.0	189.0	TW	TW902	CJU	175	3	D
95078	2017-01-26	D	KWJ	1530.0	189.0	TW	TW907	CJU	181	4	D
95079	2017-01-26	A	CJU	1556.0	189.0	TW	TW907	KWJ	181	4	D
95080	2017-01-26	D	TAE	2129.0	189.0	7C	7C2985	DAD	186	0	I
95081	2017-01-26	D	CJU	813.0	189.0	TW	TW902	KWJ	175	3	D
...	...	...	...	...	...	...	...	...	...	...	...
206957	2022-09-12	D	GMP	943.0	189.0	LJ	LJ437	HIN	77	0	D
206958	2022-09-12	A	GMP	1209.0	189.0	LJ	LJ438	HIN	189	6	D
206959	2022-09-12	A	GMP	1907.0	189.0	LJ	LJ440	HIN	188	3	D
206960	2022-09-12	D	GMP	1651.0	189.0	LJ	LJ439	HIN	83	0	D
206961	2022-09-12	A	GMP	914.0	189.0	LJ	LJ302	CJU	187	0	D

[그림3] 운항 데이터 최근 5년 치 특송기간 추출

#### ◆ 파생 변수 생성 및 컬럼 정리

이어서 학습에 필요한 여러 파생 변수를 도출해내었는데, 이때 초반의 파생 변수를 몇 개 생성하고 모델링을 거쳐 그 성능을 파악한 후 성능 보완을 위해 계속해서 새로운 파생 변수를 추가해가는 과정을 거쳤다.


우선 '총유임승객수'와 '총무임승객수'를 더하여 '총승객수'를 도출하였다. 시간대별 탑승률을 예측해야 하므로 '실제운항시분'에서 시 단위의 '운항시'를 추출하였고, 항공편이 주간엔 운행했는지 야간에 운행했는지를 담은 '주야여부' 또한 생성하였다. 그리고 '총승객수'와 '좌석수'를 이용하여 '탑승률'을 도출하였다. 그 결과 완성된 컬럼은 다음과 같다.

	실제운항일자	출발도착구분코드	IATA공항코드	실제운항시분	좌석수	총유임승객수	총무임승객수	노선구분코드	시즌ID	n번째날	특송기간	종류	총승객수	운항시	주야여부	탑승률
95077	2017-01-26	A	KWJ	842.0	189.0	175	3	D	123281.0	1	5	SL	178	8시	주간	94.2
95078	2017-01-26	D	KWJ	1530.0	189.0	181	4	D	123279.0	1	5	SL	185	15시	주간	97.9
95079	2017-01-26	A	CJU	1556.0	189.0	181	4	D	123279.0	1	5	SL	185	15시	주간	97.9
95080	2017-01-26	D	TAE	2129.0	189.0	186	0	I	123988.0	1	5	SL	186	21시	주간	98.4
95081	2017-01-26	D	CJU	813.0	189.0	175	3	D	123281.0	1	5	SL	178	8시	주간	94.2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
206957	2022-09-12	D	GMP	943.0	189.0	77	0	D	279095.0	5	5	CH	77	9시	주간	40.7
206958	2022-09-12	A	GMP	1209.0	189.0	189	6	D	279097.0	5	5	CH	195	12시	주간	103.2
206959	2022-09-12	A	GMP	1907.0	189.0	188	3	D	279098.0	5	5	CH	191	19시	주간	101.1
206960	2022-09-12	D	GMP	1651.0	189.0	83	0	D	279096.0	5	5	CH	83	16시	주간	43.9
206961	2022-09-12	A	GMP	914.0	189.0	187	0	D	279090.0	5	5	CH	187	9시	주간	98.9

[그림4] 파생 변수 생성

그리고 중복되는 노선의 유무 역시 항공편의 탑승률에 영향을 미칠 것으로 판단하여 '실제운항일자'와 '운항편명'을 기준으로 '탑승률'의 평균을 도출함으로써 노선별 평균 탑승률을 새로운 변수로 생성하였다.

실제운항일자	운항편명	탑승률
2010-02-12	7C100	64.0
	7C101	92.6
	7C104	90.5
	7C105	102.2
	7C106	95.8
...	...	...
2022-09-12	VJ991	43.0
	VJ992	99.1
	VJ993	55.9
	VN426	74.3
	VN427	57.2



n번째날	특송기간	특송종류	총승객수	운항시	주야여부	탑승률	평균탑승률
1	5	SL	178	8시	주간	94.2	87.9
1	5	SL	185	15시	주간	97.9	90.5
1	5	SL	185	15시	주간	97.9	90.5
1	5	SL	186	21시	주간	98.4	83.7
1	5	SL	178	8시	주간	94.2	87.9
...	...	...	...	...	...	...	...
5	5	CH	77	9시	주간	40.7	78.4
5	5	CH	195	12시	주간	103.2	76.9
5	5	CH	191	19시	주간	101.1	76.9

[그림5] 파생 변수 생성

마지막으로 특송기간 중에서도 어떤 요일이었는지, 주말이었는지 아닌지의 여부를

따지는 '요일', '주말' 변수를 추가로 생성함으로써 학습 데이터를 완성하였다.

	실제운항 일자	출발도착 구분코드	IATA공항 코드	실제운항 시분	좌석 수	IATA항공 사코드	운항편 명	상대IATA공 항코드	총유원 승객수	총무원 승객수	...	특송 종류	총승 객수	운항 시	주야 여부	탑승 률	평균탑 승률	요 일	주 말
0	2017-01-26	A	KWJ	842.0	189.0	TW	TW902	CJU	175	3	...	SL	178	8시	주간	94.2	87.9	3	0
1	2017-01-26	D	KWJ	1530.0	189.0	TW	TW907	CJU	181	4	...	SL	185	15시	주간	97.9	90.5	3	0
2	2017-01-26	A	CJU	1556.0	189.0	TW	TW907	KWJ	181	4	...	SL	185	15시	주간	97.9	90.5	3	0
3	2017-01-26	D	TAE	2129.0	189.0	7C	7C2985	DAD	186	0	...	SL	186	21시	주간	98.4	83.7	3	0
4	2017-01-26	D	CJU	813.0	189.0	TW	TW902	KWJ	175	3	...	SL	178	8시	주간	94.2	87.9	3	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
96743	2022-09-12	D	GMP	943.0	189.0	LJ	LJ437	HIN	77	0	...	CH	77	9시	주간	40.7	78.4	0	0
96744	2022-09-12	A	GMP	1209.0	189.0	LJ	LJ438	HIN	189	6	...	CH	195	12시	주간	103.2	76.9	0	0
96745	2022-09-12	A	GMP	1907.0	189.0	LJ	LJ440	HIN	188	3	...	CH	191	19시	주간	101.1	76.9	0	0
96746	2022-09-12	D	GMP	1651.0	189.0	LJ	LJ439	HIN	83	0	...	CH	83	16시	주간	43.9	78.4	0	0
96747	2022-09-12	A	GMP	914.0	189.0	LJ	LJ302	CJU	187	0	...	CH	187	9시	주간	98.9	86.0	0	0

[그림6] 완성된 학습 데이터

## 2.2.3 예측 데이터

예측 데이터의 경우 학습 데이터와 그 형태가 똑같아야 하므로, 역시 전처리 과정을 수행하였다.

### ◆ 예측 데이터 전처리

	예정운항일자	운항편	공항	상대공항	국내외	국제	출발	도착	항공사코드	기종	부정기여부	임시승편여부	예정운항시분	기준탑승률	공급석	예상승객수
0	20230120	LJ038	PUS	CEB	I	A	A	LJ	738A	N	N	N	45	88.4	189	178
1	20230121	LJ038	PUS	CEB	I	A	A	LJ	738A	N	N	N	45	88.4	189	167
2	20230122	LJ038	PUS	CEB	I	A	A	LJ	738A	N	N	N	45	88.4	189	170
3	20230123	LJ038	PUS	CEB	I	A	A	LJ	738A	N	N	N	45	88.4	189	185
4	20230124	LJ038	PUS	CEB	I	A	A	LJ	738A	N	N	N	45	88.4	189	185
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
6608	20230120	LJL582	CJJ	CJU	D	A	A	LJ	B739	N	N	N	2320	77.4	188	156
6609	20230121	LJL582	CJJ	CJU	D	A	A	LJ	B739	N	N	N	2320	77.4	188	145
6610	20230122	LJL582	CJJ	CJU	D	A	A	LJ	B739	N	N	N	2320	77.4	188	149
6611	20230123	LJL582	CJJ	CJU	D	A	A	LJ	B739	N	N	N	2320	77.4	188	172
6612	20230124	LJL582	CJJ	CJU	D	A	A	LJ	B739	N	N	N	2320	77.4	188	172

[그림7] 전처리 전

우선 필요한 컬럼을 추출한 후, 학습 데이터를 완성하였던 과정과 동일한 작업을 수행함으로써 학습 데이터와 똑같은 틀로 예측 데이터를 전처리하였다(그림8). 추후 학습 데이터를 활용하여 모델링을 수행한 후 예측 데이터의 탑승률을 도출하는 것으로 진행된다.

	실제운항일 자	출발도착구분 코드	IATA공항 코드	좌석 수	IATA항공사 코드	운항편 명	상대IATA공항 코드	노선구분 코드	n번째 날	특송기 간	특송종 류	운항 시	주야여 부	평균탑승 률	요일	주말
0	2023-01-20	A	PUS	189	LJ	LJ038	CEB	I	1	5	SL	0시	야간	78.7	4	0
1	2023-01-21	A	PUS	189	LJ	LJ038	CEB	I	2	5	SL	0시	야간	78.7	5	1
2	2023-01-22	A	PUS	189	LJ	LJ038	CEB	I	3	5	SL	0시	야간	78.7	6	1
3	2023-01-23	A	PUS	189	LJ	LJ038	CEB	I	4	5	SL	0시	야간	78.7	0	0
4	2023-01-24	A	PUS	189	LJ	LJ038	CEB	I	5	5	SL	0시	야간	78.7	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
6608	2023-01-20	A	CJJ	188	LJ	LJLJ582	CJU	D	1	5	SL	23시	야간	82.6	4	0
6609	2023-01-21	A	CJJ	188	LJ	LJLJ582	CJU	D	2	5	SL	23시	야간	82.6	5	1
6610	2023-01-22	A	CJJ	188	LJ	LJLJ582	CJU	D	3	5	SL	23시	야간	82.6	6	1
6611	2023-01-23	A	CJJ	188	LJ	LJLJ582	CJU	D	4	5	SL	23시	야간	82.6	0	0
6612	2023-01-24	A	CJJ	188	LJ	LJLJ582	CJU	D	5	5	SL	23시	야간	82.6	1	0

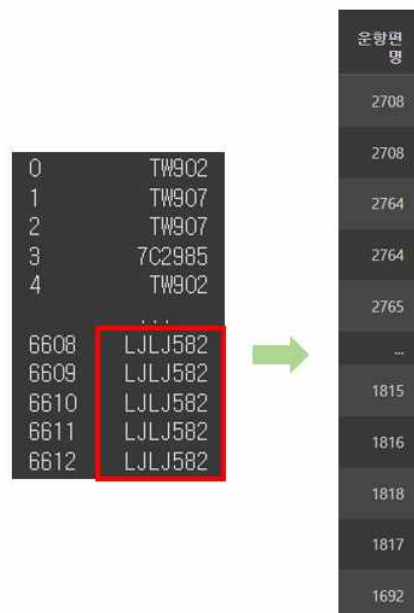
[그림8] 전처리 후

## 2.3 분석 모델링

모델의 경우, 탑승률을 예측값으로 하고 변수 대부분이 Categorical 변수인 점을 고려하여 CatboostRegressor로 선정하였다.

### 2.3.1 라벨 인코딩

변수 중 '운항편명'의 경우 기존의 '시즌ID'를 대체하여 선정한 변수로, 데이터값 중 동일 값임에도 앞의 문자가 반복되어 다른 값으로 인식하는 문제가 있어(ex: 'LJLJ582'  $\neq$  'LJ582') 라벨 인코딩을 따로 수행하였다.

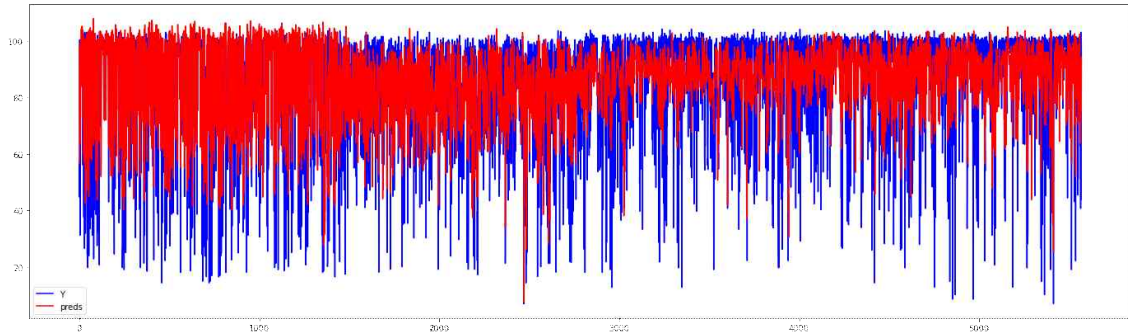


[그림9] '운항편명' 라벨 인코딩



## 2.3.2 학습 후 예측

이후 2017년 설부터 2022년 설까지를 학습 데이터로, 22년 추석을 테스트 데이터로 설정한 후 모델을 학습시켰다. 그 결과, MAPE가 학습 데이터는 8.58, 테스트 데이터는 15.65였고, RMSE는 각각 9.9, 19.63이었다.



[그림10] 모델 학습 결과 ‘실제값’-‘예측값’

이렇게 학습된 모델을 활용하여 예측 데이터의 최종 예측값(예상승객수)을 도출해내었다.

출발노선구분코드	IATA공항코드	좌석수	IATA항공사코드	운항편명	강대IATA공항코드	노선구분코드	n번째날	특송기간	특송종류	운행시	주여여부	평균탑승률	요일	주말	예측값	일자	예상승객수	
0	A	PUS	189	LJ	LJ038	CEB	I	1	5	SL	0시	야간	78.7	4	0	82.9	2023-01-20	157
1	A	PUS	189	LJ	LJ038	CEB	I	2	5	SL	0시	야간	78.7	5	1	83.2	2023-01-21	157
2	A	PUS	189	LJ	LJ038	CEB	I	3	5	SL	0시	야간	78.7	6	1	79.6	2023-01-22	150
3	A	PUS	189	LJ	LJ038	CEB	I	4	5	SL	0시	야간	78.7	0	0	91.7	2023-01-23	173
4	A	PUS	189	LJ	LJ038	CEB	I	5	5	SL	0시	야간	78.7	1	0	92.4	2023-01-24	175
--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	
6608	A	CJJ	188	LJ	LJUS82	CJU	D	1	5	SL	23시	야간	82.6	4	0	79.8	2023-01-20	150
6609	A	CJJ	188	LJ	LJUS82	CJU	D	2	5	SL	23시	야간	82.6	5	1	82.2	2023-01-21	155
6610	A	CJJ	188	LJ	LJUS82	CJU	D	3	5	SL	23시	야간	82.6	6	1	79.9	2023-01-22	150
6611	A	CJJ	188	LJ	LJUS82	CJU	D	4	5	SL	23시	야간	82.6	0	0	89.3	2023-01-23	168
6612	A	CJJ	188	LJ	LJUS82	CJU	D	5	5	SL	23시	야간	82.6	1	0	95.8	2023-01-24	180

[그림11] 최종 예측값 도출

### 3. 결론

#### 3.1 분석 결과

도출된 최종 예측값을 토대로 2023년도 설 특송기간 중 가장 승객수가 많은 노선 및 시간대를 확인하기 위해 결과값을 집계하면서 확인하였다.

	출발도착구분코드	노선구분코드	IATA공항코드	일자	운행시	예상승객수
0	A	D	CJJ	2023-01-20	10시	477
1	A	D	CJJ	2023-01-20	12시	160
2	A	D	CJJ	2023-01-20	13시	168
3	A	D	CJJ	2023-01-20	14시	162
4	A	D	CJJ	2023-01-20	15시	272
...	...	...	...	...	...	...
1554	D	I	YNY	2023-01-23	19시	80
1555	D	I	YNY	2023-01-23	21시	73
1556	D	I	YNY	2023-01-23	8시	81
1557	D	I	YNY	2023-01-24	20시	128
1558	D	I	YNY	2023-01-24	8시	103

[그림 12] 공항별 노선, 일자-시간대 집계

주요 공항인 김포, 김해, 제주 공항의 경우, '예상승객수'를 기준으로 따져봤을 때 각각 24일 21시, 22일 8시, 24일 15시에 탑승객이 가장 많은 것으로 추정하였다.

	A	B	C	D	E	F	G
1		출발도착구분코드	노선구분코드	IATA공항코드	일자	운행시	예상승객수
17	15	A	D	GMP	2023-01-24	21시	3320
47	45	D	D	GMP	2023-01-21	7시	2913
49	47	A	D	GMP	2023-01-24	14시	2900
52	50	D	D	GMP	2023-01-23	7시	2862
53	51	D	D	GMP	2023-01-24	13시	2862
54	52	A	D	GMP	2023-01-20	21시	2857
62	60	A	D	GMP	2023-01-23	21시	2816
66	64	D	D	GMP	2023-01-24	18시	2775
71	69	A	D	GMP	2023-01-21	14시	2747
74	72	D	D	GMP	2023-01-24	15시	2731

[그림 13] 김포공항 '예상승객수' TOP 10

	A	B	C	D	E	F	G
1		출발도착구분코드	노선구분코드	IATA공항코드	일자	운항시	예상승객수
240	238	D	I	PUS	2023-01-22	8시	1719
248	246	A	I	PUS	2023-01-24	6시	1651
251	249	A	I	PUS	2023-01-23	7시	1634
260	258	D	I	PUS	2023-01-23	21시	1552
261	259	D	I	PUS	2023-01-24	8시	1536
262	260	D	I	PUS	2023-01-21	8시	1523
265	263	A	I	PUS	2023-01-20	7시	1502
266	264	D	I	PUS	2023-01-23	8시	1486
272	270	D	I	PUS	2023-01-20	8시	1436
273	271	A	D	PUS	2023-01-24	20시	1427

[그림14] 김해공항 ‘예상승객수’ TOP 10

	A	B	C	D	E	F	G
1		출발도착구분코드	노선구분코드	IATA공항코드	일자	운항시	예상승객수
2	0	D	D	CJU	2023-01-24	15시	3954
3	1	A	D	CJU	2023-01-24	16시	3883
4	2	D	D	CJU	2023-01-23	15시	3781
5	3	A	D	CJU	2023-01-24	19시	3670
6	4	A	D	CJU	2023-01-23	14시	3641
7	5	A	D	CJU	2023-01-20	19시	3578
8	6	A	D	CJU	2023-01-21	16시	3576
9	7	A	D	CJU	2023-01-24	14시	3541
10	8	A	D	CJU	2023-01-23	19시	3498
11	9	A	D	CJU	2023-01-22	16시	3447

[그림15] 제주공항 ‘예상승객수’ TOP 10

## 3.2 기대 효과 및 향후 과제

기존에는 해당 데이터와 친숙한 담당자의 감으로 설정한 대푯값으로 계산하여 예측 값을 얻어왔고 또 그 결과가 다소 좋았으나, 이번 프로젝트의 경우 데이터를 기반으로 예측하였다는 것에 의의가 있다. 예측 성능 또한 낮지 않았기에 앞으로도 해당 모델을 활용하여 특정 기간의 예측에 도움이 될 것으로 보인다.

다만, 이번 프로젝트는 단기에 수행되었기에 충분히 다른 요인들을 고려하지 못한 점이 아쉽다. 하여 추후 모델의 성능을 올리기 위해 설과 추석 등 특정 기간에 영향을 주는 요인을 추가로 고려하여 변수로 설정해준다면 더욱 정확한 예측도를 얻어낼 것으로 예상된다.

## 4. 부록

### 4.1 데이터 명세

본 프로젝트에서 활용한 데이터의 목록과 데이터별 컬럼 명세는 다음과 같다.

데이터	구분	중요도	제공기관	비고
항공 운항	정형/분석	필수	한국공항공사	항공편별 운항 일자, 탑승률 등
특송기간	정형/분석	필수	한국공항공사	연도별 특송기간

[표1] 활용 데이터 목록

#### 4.1.1 항공 운항

컬럼명	데이터 타입	예시
실제운항일자	Categorical	2017-01-26
출발도착구분코드	Categorical	A
IATA공항코드	Categorical	KWJ
실제운항시분	Numerical	842
좌석수	Numerical	189
IATA항공사코드	Categorical	TW
상대IATA항공사코드	Categorical	CJU
운항편명	Categorical	TW902
총유임승객수	Numerical	175
총무임승객수	Numerical	3
노선구분코드	Categorical	SL

[표2] 항공 운항 데이터 컬럼 명세

#### 4.1.2 특송기간 데이터

컬럼명	데이터 타입	예시
특별기간코드	Categorical	CH
시작일자	Numerical	20030909
종료일자	Numerical	20030915

[표3] 특송기간 데이터 컬럼 명세

### 4.1.3 전처리 후 학습 데이터

컬럼명	데이터 타입	예시
출발도착구분코드	Categorical	A
IATA공항코드	Categorical	KWJ
좌석수	Numerical	189
IATA항공사코드	Categorical	TW
운항편명	Categorical	2818
상대IATA공항코드	Categorical	CJU
노선구분코드	Categorical	D
n번째날	Categorical	1
특송기간	Numerical	5
특송종류	Categorical	SL
운항시	Categorical	8시
주야여부	Categorical	주간
평균탑승률	Numerical	87.9
요일	Categorical	3
주말	Categorical	0

[표4] 학습 데이터 컬럼 명세

## 4.2 코드 명세

### ◆ 구글 드라이브 연결 및 경로 설정

원천 데이터의 경우, 보안상 클라우드 환경에 업로드가 불가하므로 1차 전처리는 로컬에서 수행하였고, 그 후 전처리가 완료된 데이터를 구글 드라이브에 업로드하여 Colab으로 작업을 진행하였다. 이를 위해 구글 드라이브에 연결하고, 파일을 저장하고 읽어올 경로를 설정하였다.

```
from google.colab import drive
drive.mount('/content/drive')
# 데이터 접근/저장경로(로컬, 구글 드라이브)
# path = 'C:/Users/user/Documents/intern_project/jan_project/'
path = '/content/drive/MyDrive/공항공사_인턴/2023설날예측/2023_특송기간예측/'
```

### ◆ 필요한 라이브러리 설치 및 import

전처리와 모델링 과정에서 필요한 라이브러리를 pip install 명령어로 설치한 후 import하여 사용하였다. CatboostRegressor를 활용할 예정이므로, catboost를 설치하였다. 그리고 설치한 패키지가 제대로 import 되는지 확인해야 한다.

```

!pip install catboost      # catboost 설치

# 필요 라이브러리 불러오기
import pandas as pd
import numpy as np

import warnings
warnings.filterwarnings('ignore')

# 시간 관련
from datetime import datetime, date, timedelta
from dateutil import relativedelta

# 모델링
from catboost import CatBoostRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_percentage_error, mean_squared_error,
mean_absolute_error, r2_score
from sklearn.preprocessing import LabelEncoder
import matplotlib.pyplot as plt

```

#### ◆ 항공 운항 원천 데이터 가공(로컬에서만 실행할 것)

pd.read\_csv로 원천 데이터(하단의 'data' 파일)를 불러서 컬럼명을 확인하고, 영문으로 돼 있는 컬럼을 한글로 바꿔주기 위해 해당 정보를 담고 있는 다른 데이터(하단의 'test' 파일)를 함께 불러와서 작업을 수행하였다.

```

# 원본 데이터 로드
data = pd.read_csv(path + '항공통계(2010.01-2022.09).csv', encoding = 'cp949')
# data.columns          # 데이터 컬럼명(영문)

# 영문명별 한글명이 매칭돼있는 데이터 로드
test = pd.read_excel('path + 'iFIS통계마스터(컬럼명세 및 코드명세)_220919.xlsx')

en_col = test['COLUMN_NAME']          # 영문명 컬럼 리스트 추출
kor_col = test['COMMENTS 2 (수정)']    # 한글명 컬럼 리스트 추출

real_data = data[en_col]              # 원본 데이터의 컬럼을 ABC 순으로 정렬
real_data.columns = kor_col           # 컬럼명을 영문->한글로 변경
real_data = real_data.reset_index(drop = True)      # 인덱스 초기화
real_data.to_csv(path + '컬럼명정리_2010.01_2022.09.csv') # 데이터 저장

```

#### ◆ 특송기간 원천 데이터 가공(로컬에서만 실행할 것)

마찬가지로 특송기간 데이터도 pd.read\_csv로 읽어온 후, '추석'과 '설날' 부분만 추출한 후 적절한 전처리를 수행하였다.

```
# 특송기간 데이터 로드 후 '추석', '설날' 데이터 추출
spec = pd.read_excel(path + '특송일정.xls')
spec = spec[(spec['특별기간코드'] == 'CH') | (spec['특별기간코드'] == 'SL')]
spec = spec.iloc[:, 2:4] # 필요 컬럼 추출(특별기간코드, 시작일자, 종료일자)
spec.isnull().sum()      # null값 확인

spec = spec.rename(columns = {'특별기간코드' : '종류'}) # 컬럼명 수정
# 데이터 형 변환 int -> datetime# ex) '20030909' -> '2003-09-09'
spec['시작일자'] = spec['시작일자'].apply(
    lambda x: pd.to_datetime(str(x), format = '%Y-%m-%d'))
spec['종료일자'] = spec['종료일자'].apply(
    lambda x: pd.to_datetime(str(x), format = '%Y-%m-%d'))
```

현재 특송기간이 시작, 종료일자로만 명시되어있기에, 특송기간 내 모든 날짜를 출력함과 동시에 이와 관련된 파생 변수를 만들었다. 먼저 'n번째날'은 앞서 2.1.2에서 언급했듯 특송기간에서 특정 날짜가 몇 번째 날인지를 뜻하고, '특송기간'은 해당 특송기간의 길이를 의미한다.

```
# 특송기간 날짜 데이터프레임(기준) 생성
total_date = pd.DataFrame(columns = ['실제운항일자', 'n번째날', '특송기간'])

for i in range(len(spec)):
    holi = spec.iloc[i,0] # i번째 행 특송기간의 종류
    start = spec.iloc[i,1] # i번째 행 특송기간의 시작 날
    end = spec.iloc[i,1] # i번째 행 특송기간의 마지막 날
    periods = (end-start).days + 1 # i번째 행 특송기간의 길이
    # i번째 행 특송기간만큼 날짜 생성 후 데이터프레임으로 변환
    add_date = pd.date_range(start, periods=periods, freq='D')
    add_df = pd.DataFrame(add_date, columns = ['실제운항일자'])
    add_df['종류'] = holi
    add_df['n번째날'] = [i for i in range(1, periods+1)] # 'n번째날' 컬럼 생성
    add_df['특송기간'] = periods # '특송기간' 컬럼 생성
    # 해당 데이터프레임을 기준 데이터프레임에 붙이기
    total_date = pd.concat([total_date, add_df])

total_date = total_date.reset_index(drop=True) # 인덱스 초기화
total_date.to_csv(path+'10년치_특송기간.csv', encoding = 'cp949') # 데이터 저장
```

## ◆ 학습 데이터 생성

다시 운항 데이터로 돌아와서, 컬럼이 정리된 데이터를 로드한 후 null 값 유무를 확인하고 처리하였다. 또한, null값 뿐만 아니라 '2000xxxx'가 아닌 '00xxxx'처럼 형태가 다른 데이터가 존재하였기에 함수(del\_date)를 선언하고 이를 수정하였다.

```
# 한글명 컬럼으로 된 데이터 로드
data = pd.read_csv(path + '컬럼명정리_2010.01_2022.09.csv', index_col = 0)
data['실제운항일자'].isnull().sum() # null값 확인 # 존재함
data2 = data.dropna(subset = ['실제운항일자']) # '운항일자' null값 제거
data2['실제운항일자'] = data2['실제운항일자'].astype('int') # 형 변환 float -> int

# '운항일자' 전처리
def del_date(x):
    if x <= 100000: # '0'으로 기입돼있는 데이터
        x = np.nan # NaN으로 대체
    elif x <= 1000000: # ex) '120928'
        x += 20000000 # ex) '20120928'
    else:
        return x # 그 외에는 그대로 유지
    return x

data2['실제운항일자'] = data2['실제운항일자'].apply(
    lambda x: del_date(x)) # 운항일자 데이터 전처리
data3 = data2.dropna(subset=['실제운항일자']) # NaN값 삭제
data3['실제운항일자'] = data3['실제운항일자'].astype('int') # 형 변환 str -> int
data3['실제운항일자'] = data3['실제운항일자'].apply(
    lambda x: pd.to_datetime(str(x), format='%Y-%m-%d')) # 형 변환 int -> datetime

data3['실제운항일자'].value_counts() # 운항일자 전처리 결과 확인
data3['실제운항일자'].isnull().sum() # null값 확인
data3.to_csv(path + '날짜추출전_2010_2022.csv', encoding = 'cp949')
```

이후 전처리 된 데이터에서 특송기간에 해당되는 날짜만 추출하기 위해 두 데이터를 불러온 후 join 하였다.

```
# 전처리 된 데이터와 특송기간 데이터 로드
ex_df = pd.read_csv(path + '날짜추출전_2010_2022.csv', encoding='cp949', index_col=0)
total_date = pd.read_csv(path + '10년치_특송기간.csv', encoding='cp949', index_col=0)

# 운항일자 초 단위 출력 삭제 # ex) 2003-09-09 00:00:00 -> 2003-09-09
total_date['실제운항일자'] = total_date['실제운항일자'].apply(lambda x: x[:10])
```



```
# 특송기간에 해당되는 데이터만 추출하기 위해 기존 데이터와 특송기간 데이터 join
res_df = pd.merge(ex_df, total_date,
                  left_on = '실제운항일자', right_on = '실제운항일자', how = 'inner')
# res_df.to_csv(path + '특송기간_2010_2022.csv', encoding = 'cp949')
```

또한, 전체 데이터의 경우 2010년부터 2022년까지의 기간으로, 중간에 코로나로 인해 항공편의 수가 급격히 줄었던 시기도 함께 포함하고 있다. 이때 2023년도를 예측하는 데에 10년도 더 된 데이터는 그 영향력이 희미하다고 판단되어, 최종 학습 데이터의 기간은 코로나 기간인 2020년을 뺀 최근 5년, 즉 2017년 설부터 2022년 설까지로 설정하였다.

```
# 특송기간 데이터 로드
df = pd.read_csv(path + '특송기간_2010_2022.csv', encoding = 'cp949', index_col = 0)
# 필요한 데이터 컬럼 및 기간 추출
df.info() # 데이터 확인
df2 = df.iloc[:, [0, 1, 2, 3, 6, 29, 54, 60, 61, 78, -3, -2, -1]]
df2 = df2[((df2.실제운항일자 >= '2021-01-01')) | ((df2.실제운항일자 <= '2019-12-31')
          & (df2.실제운항일자 > '2017-01-01'))] # 2020년 제외(코로나)
```

그리고 학습용 데이터를 완성하기 위해 여러 파생 변수를 생성하였다. 먼저 모델링에서 '탑승률'을 예측값으로 설정할 것이므로, '총유임승객수'와 '총무임승객수'를 더한 '총승객수'를 구하고 이를 '좌석수'로 나눠 '탑승률'을 도출하였다. 또한, '운항시'의 경우, 추후 시간대별 탑승률을 구해야 하므로 '실제운항시분'으로부터 따로 추출하였다. 그리고 이를 통해 주간인지 야간 비행인지를 판단한 '주야여부' 변수도 함께 생성하였다.

```
# 파생 변수 생성
df2["총승객수"] = df2["총유임승객수"] + df2["총무임승객수"]
df2["탑승률"] = round(df2["총승객수"] / df2["좌석수"] * 100, 1)
df2["운항시"] = (df2["실제운항시분"] // 100).astype("str").apply(lambda x : x[:-2] + "시")
df2["주야여부"] = ["주간"
                    if ((df2["실제운항시분"][i] // 100) < 22) & ((df2["실제운항시분"][i] // 100) > 6)
                    else '야간' for i in range(df2.shape[0])]
```

또한, 운항일자와 운항편명을 기준으로 노선별 평균 탑승률을 계산하여 '평균탑승률'을 생성하였다.

```
# 노선별 평균 탑승률 계산 후 '평균탑승률' 컬럼 생성
ex_rate = pd.DataFrame(df2.groupby(['실제운항일자', '운항편명']).mean()['탑승률'])
total_df = pd.merge(df2, res, how = 'left',
                    on = ['출발도착구분코드', 'IATA공항코드', '상대IATA공항코드'])
```

```
total_df.rename(columns = {'종류': '특송종류'}, inplace = True)    # 컬럼명 수정
# total_df.to_csv(path + '전처리완료.csv', encoding = 'cp949')
```

그리고 주말 여부와 요일 변수를 추가로 생성하였다. 먼저 해당 날짜가 무슨 요일인지를 추출한 뒤, 주말 여부를 판단하고 해당 컬럼을 붙여줌으로써 학습 데이터를 완성하였다.

```
# 주말(토, 일요일) 여부 확인 함수
def check_day(date_today):
    year = round(date_today / 10000)
    mon = round((date_today % 10000) / 100)
    day = date_today % 100
    # 일자 입력 -> 요일(숫자) 반환
    today_w = datetime(year, mon, day).weekday()
    return today_w

t_df = pd.read_csv(path + '전처리완료.csv', encoding = 'cp949', index_col = 0)
# 데이터 형 변환
t_df['실제운항일자'] = pd.to_datetime(t_df['실제운항일자'])
t_df2 = t_df['실제운항일자'].apply(lambda x: x.strftime('%Y%m%d'))
t_df2 = pd.DataFrame(t_df2.astype("int"))
t_df2["weekday"] = t_df2["실제운항일자"].apply(lambda x: check_day(x))    # 요일 추출

# 요일별 인코딩
day_t_df = pd.get_dummies(t_df2['weekday'])
day_t_df.columns = ['월요일', '화요일', '수요일', '목요일', '금요일', '토요일', '일요일']
tot_t_df = pd.concat([t_df2, day_t_df], axis=1)

# 토요일 or 일요일 -> '주말': 1
tot_t_df['주말'] = day_t_df[['토요일', '일요일']].sum(axis=1)

train_data_weekday = tot_t_df[["실제운항일자", "weekday", "주말"]]    # 필요 컬럼 추출
# 기존 데이터에 '요일', '주말' 컬럼 생성
t_df["요일"] = train_data_weekday["weekday"]
t_df["주말"] = train_data_weekday["주말"]
# t_df.to_csv(path + "전처리완료2.csv", encoding = "cp949")
```

#### ◆ 데이터 모델링 수행(구글 드라이브)

완성된 학습 데이터를 바탕으로 모델링 구축을 시작하였다. 우선 '운항편명'의 경우 2.3.1에서 설명했듯 라벨 인코딩이 필요하여 이를 처리하였다. 그리고 학습 및 테스트 데이터는 필요한 컬럼을 설정한 후 기간도 함께 설정하여 나누었다.

```
# 전처리 완료된 학습용 데이터 로드
df = pd.read_csv(path + '전처리완료2.csv', encoding = 'cp949', index_col = 0)
df = df[df.총승객수 != 0]

df2 = df.copy()          # 데이터 복사
# '운항편명' 라벨 인코딩
encoder = LabelEncoder()
fitted = encoder.fit(df2['운항편명'])

tt = df2['운항편명']
ex_df = pd.read_csv(path + '2023특송_(일자포함)2.csv', encoding = 'cp949', index_col = 0)
ex_df['운항편명'] = ex_df['운항편명'].astype('str')
tt = tt.append(ex_df['운항편명']) # 학습 데이터의 운항편명 로드

fitted = encoder.fit(tt)
encoder.transform(ex_df['운항편명'])
df2['운항편명'] = encoder.transform(df2['운항편명'])

# 학습, 테스트 데이터 분리
train = df2.loc[:'2022-01-31', :] # 22년 설까지 학습
test = df2.loc['2022-09-09':, :] # 22년 추석 테스트

# 필요 컬럼만 추출
x_train = train.iloc[:, [1, 2, 4, 5, 6, 7, 10, 12, 13, 14, 16, 17, 19, 20, 21]]
y_train = train['탑승률']
x_test = test.iloc[:, [1, 2, 4, 5, 6, 7, 10, 12, 13, 14, 16, 17, 19, 20, 21]]
y_test = test['탑승률']
```

이후, CatBoostRegressor 모델을 생성하고 학습을 진행하였다. 이때, 회귀모델이므로 Categorical 변수를 명확히 알려줄 필요가 있어 이를 명시하였다. 학습을 마친 후 rmse와 mape로 해당 모델의 성능을 확인하였고, 모델과 그 예측 결과값을 확인하였다(해당 과정은 최고 성능의 모델이 도출될 때까지 파생 변수를 새로 도출하고 테스트하면서 계속 반복되었다).

그리고 예측값 중 100을 넘는 경우가 있어, 해당 부분은 기관 담당자의 요청에 따라 99.9로 수정하였다.

```

# catboostRegressor 모델 생성 및 학습
model = CatBoostRegressor(cat_features = ['출발도착구분코드', 'IATA공항코드',
                                          'IATA항공사코드', '운항편명', '상대IATA공항코드',
                                          '노선구분코드', 'n번째날', '특송종류', '운항시',
                                          '주야여부', '요일', '주말'])

model.fit(x_train, y_train)

# 학습 결과
train_pred = model.predict(x_train)
test_pred = model.predict(x_test)

# mape, rmse
print(round(mean_absolute_percentage_error(train_pred, y_train) * 100, 2),
        np.sqrt(mean_squared_error(train_pred, y_train)))
print(round(mean_absolute_percentage_error(test_pred, y_test) * 100, 2),
        np.sqrt(mean_squared_error(test_pred, y_test)))

# 결과값 저장
preds = np.concatenate((train_pred, test_pred), axis=0)
df2['예측값'] = preds # 예측값 컬럼 생성
model.save_model(path + 'cb_all.model') # 모델 저장

df2['예측값'].describe() # 예측값 확인
df2.loc[df2['예측값'] >= 100, '예측값'] = 99.9 # 100 이상의 예측값은 99.9로 처리
df2['예측값'].describe() # 수정 후 확인
# df2.to_csv(path + '학습후_예측.csv', encoding = 'cp949')

```

#### ◆ 예측 데이터 전처리(구글 드라이브)

최종 2023년도 데이터를 예측하기에 앞서, 모델링을 돌리기 위해 데이터의 형태를 학습 데이터와 같이 전처리하는 과정을 거쳤다. 해당 과정은 앞서 학습 데이터 전처리 과정과 동일하므로 설명은 생략하였다.

```

# 출도착, 공항별 평균탑승률 데이터 로드
res = pd.read_csv(path + '2010_2022_평균탑승률.csv', encoding = 'cp949')

# 예측데이터 로드(2023 설연휴)
test = pd.read_excel(path + '(230112기준)특송스케줄(230120-230124).xls', header = 1)
test_df = test.iloc[:, [0, 1, 2, 3, 4, 5, 6, 10, 11, 12]] # 필요 컬럼 추출

```

```

# 학습 데이터와 맞게 틀 맞추기
# 운항시, 주야여부, 특송기간, n번째날 변수 생성 및 정리

test_df['특송종류'] = 'SL'
test_df['예정운항시분'] = (test_df['예정운항시분'] // 100).astype('str')
test_df['예정운항시분'] = test_df['예정운항시분'].apply(lambda x : x + '시')
test_df['예정운항일자'] = test_df['예정운항일자'].apply(
    lambda x: pd.to_datetime(str(x), format = '%Y-%m-%d'))

# 컬럼명 수정
test_df.rename(columns={'예정운항일자': '실제운항일자',
                        '국내D 국제I': '노선구분코드',
                        '출발D 도착A': '출발도착구분코드',
                        '예정운항시분': '운항시',
                        '항공사코드': 'IATA항공사코드',
                        '공항': 'IATA공항코드',
                        '공급석': '좌석수',
                        '상대공항': '상대IATA공항코드',
                        '운항편': '운항편명'}, inplace = True)
test_df['주야여부'] = test_df['운항시'].apply(
    lambda x: '주간' if ((int(x[:-1]) < 22) & (int(x[:-1]) > 6)) else '야간')

start = pd.Timestamp(2023, 1, 20)
end = pd.Timestamp(2023, 1, 24)
periods = (end-start).days + 1
test_df['특송기간'] = periods
test_df['n번째날'] = test_df['실제운항일자'].apply(lambda x: (x-start).days + 1)
test_df = test_df.drop(['기준탑승률'], axis = 1)

# '평균탑승률' 추가
total = pd.merge(test_df, res, how = 'left',
                  on = ['출발도착구분코드', 'IATA공항코드', '상대IATA공항코드'])
# 컬럼 순서 정렬
total = total.loc[:, ['실제운항일자', '출발도착구분코드', 'IATA공항코드', '좌석수',
                     'IATA항공사코드', '운항편명', '상대IATA공항코드', '노선구분코드',
                     'n번째날', '특송기간', '특송종류', '운항시', '주야여부', '평균탑승률']]
total.info()      # 데이터 확인
# total.to_csv(path + '2023_특송(일자포함).csv', encoding = 'cp949')

```

```

## 요일, 주말, 일자/시간대별 항공편수 파생 변수 생성
# 요일, 주말
p_df = pd.read_csv(path+'2023_특송(일자포함).csv', encoding = 'cp949', index_col = 0)
p_df['실제운항일자'] = pd.to_datetime(p_df['실제운항일자'])
p_df2 = p_df['실제운항일자'].apply(lambda x: x.strftime('%Y%m%d'))
p_df2 = pd.DataFrame(p_df2.astype("int"))
p_df2["weekday"] = p_df2["실제운항일자"].apply(lambda x: check_day(x))    # 요일 추출

# 요일 기준 원핫 인코딩 데이터프레임 생성
day_p_df = pd.get_dummies(p_df2['weekday'])
day_p_df.columns = ['월요일', '화요일', '금요일', '토요일', '일요일']    # 컬럼명 설정
tot_p_df = pd.concat([p_df2, day_p_df], axis=1)    # 기존 데이터와 병합
tot_p_df['주말'] = day_p_df[['토요일', '일요일']].sum(axis = 1)    #토/일요일 - '주말': 1

pred_data_weekday = tot_p_df[["실제운항일자", "weekday", "주말"]] # 필요 컬럼 추출

# 기존 데이터에 '요일', '주말' 컬럼 생성
p_df["요일"] = pred_data_weekday["weekday"]
p_df["주말"] = pred_data_weekday["주말"]
p_df.to_csv(path + "2023_특송(일자포함)2.csv", encoding = "cp949") # 데이터 저장

# 일자/시간대별 항공편수 집계
p_count_hour = p_df.groupby(
    ["실제운항일자", "출발도착구분코드", "IATA공항코드",
     "상대IATA공항코드", "운항시"]).size().reset_index().rename(
    columns = {0: "노선_시간별운항횟수"})
p_count_day = p_df.groupby(
    ["실제운항일자", "출발도착구분코드", "IATA공항코드",
     "상대IATA공항코드"]).size().reset_index().rename(
    columns = {0: "노선_일별운항횟수"})

# 기존 데이터와 병합
p_df2 = pd.merge(p_df, p_count_hour, how = 'left',
    on = ["실제운항일자", "출발도착구분코드", "IATA공항코드",
          "상대IATA공항코드", "운항시"])
p_df2 = pd.merge(p_df2, p_count_day, how = 'left',
    on = ["실제운항일자", "출발도착구분코드", "IATA공항코드",
          "상대IATA공항코드"])
# p_df2.to_csv(path + "2023_특송(일자포함)2.csv", encoding = "cp949")

```

### ◆ 최종 데이터 예측(구글 드라이브)

완성된 모델과 예측 데이터를 통해 최종 예측값을 도출하기 위해, 이전에 학습 데이터를 준비할 때 활용했던 encoder로 운항편명을 라벨 인코딩 수행하고 필요한 컬럼만을 추출하였다.

```
# 전처리된 예측 데이터 로드
ex_df = pd.read_csv(path + '2023특송_(일자포함)2.csv', encoding = 'cp949', index_col = 0)
ex_df['운항편명'] = encoder.transform(ex_df['운항편명']) # 운항편명 인코딩
ex_df = ex_df.fillna(0)

# 필요 컬럼 추출
test_df = ex_df.loc[:, ['출발도착구분코드', 'IATA공항코드', '좌석수', 'IATA항공사코드',
                        '운항편명', '상대IATA공항코드', '노선구분코드', 'n번째날',
                        '특송기간', '특송종류', '운항시', '주야여부', '평균탑승률',
                        '요일', '주말']]
```

이후 저장해둔 모델을 활용하기 위해 똑같은 틀의 CatBoostRegressor를 만들고 모델을 불러왔다. 그리고 최종 예측값을 도출해내고, 운항편명은 다시 한글로 디코딩한 후 예측값의 상한값도 99.9로 조정하였다. 또한, 현재 예측값의 경우 '탑승률'이므로, 해당 값에 좌석수를 곱하고 100으로 나눔으로써 '예상승객수'를 최종으로 얻어내었다.

```
# 저장해둔 모델 로드
test = CatBoostRegressor(cat_features = ['출발도착구분코드', 'IATA공항코드',
                                         'IATA항공사코드', '운항편명', '상대IATA공항코드',
                                         '노선구분코드', 'n번째날', '특송종류', '운항시',
                                         '주야여부', '요일', '주말'])

test.load_model(path+'모델/cb_all.model')

test_df['예측값'] = test.predict(test_df) # 최종 데이터 예측값
test_df['운항편명'] = encoder.inverse_transform(test_df['운항편명']) # 운항편명 디코딩
test_df['일자'] = ex_df['실제운항일자']
test_df.loc[test_df["예측값"] >= 100, "예측값"] = 99.9 # 100 이상이면 99.9로 변환

# 예상승객수 = (좌석수 * 예측값 / 100)
test_df['예상승객수'] = round((test_df.좌석수 * test_df.예측값 / 100),0)
test_df['예상승객수'] = test_df.예상승객수.apply(lambda x: int(x)) # 형변환: float -> int
test_df['예측값'] = test_df.예측값.apply(lambda x: round(x, 1)) # 소수 첫째자리까지
# test_df.to_csv(path + '최종예측결과_전체.csv', encoding = 'cp949')
```

#### ◆ 최종 예측값 집계(구글 드라이브)

최종 목표는 가장 승객 탑승률이 높은 일자와 시간대를 도출해내는 것이므로, 집계를 수행하였다. 해당 결과는 엑셀로 저장한 후, 시각적으로 확인하였다.

```
final_res = pd.read_csv(path + "최종예측결과_전체.csv", encoding = "cp949", index_col = 0)
a = final_res.groupby(
    ["출발도착구분코드", '노선구분코드', "IATA공항코드", "일자", "운항시"])
    ["예상승객수"].sum().reset_index()
a.to_excel(path + "공항_시간별_승객예측값.xlsx") # 집계 결과 저장
```