Data밥상팀 1조 3차 프로젝트 (ML 예측 모델 구현) 보고서

주제 : 항공사 승객 만족도 분석

💡 참여자 : 김선우, 백준원, 조혜진

1. 서론 및 데이터

1-1 .분석목적

승객의 연령대, 비행 거리, 여행 목적(개인/비즈니스) 및 클래스(이코노미/비즈니스)에 따른 만족도 패턴을 분석하여, 장거리 승객을 위한 주요 서비스 항목을 도출하고, 만족도에 가장 큰 영향을 미치는 요인을 파악한다.

이를 기반으로 만족도를 예측하는 모델을 설계하여, 맞춤형 서비스 제공 및 마케팅 전략을 제시하는 것을 목표로 한다.

1-2. 데이터 컬럼 설명

분석 데이터 요약

1. 승객 관련 데이터(성별, 나이, 유형 등등)
2. 비행기 관련 데이터(비행거리, 시간 지연, 클래스 등등)
3. 만족도 관련 데이터(전반적인 만족도, 세부 만족도)

데이터 개요

데이터명	Airline Passenger Satisfaction
데이터 출처	kaggle
데이터 url	https://www.kaggle.com/datasets/teejmahal20/airline-passenger-satisfaction/data
데이터 설명	항공사의 만족도 설문조사가 포함된 고객들의 데이터셋
데이터 포맷	csv
컬럼 수	25
레코드 수	train : 103904, test : 25976

컬럼 정의서

Column Name	Data Type	Description
Unnamed: 0	int64	인덱스 열
id	int64	승객 ID
Gender	object	승객의 성별
Customer Type	object	고객 유형
Age	int64	승객의 실제 나이
Type of Travel	object	승객의 비행 목적
Class	object	비행기 내 여행 클래스
Flight Distance	int64	여정의 비행 거리
Inflight wifi service	int64	기내 와이파이 서비스 만족도 (0 : 해당 없음)
Departure/Arrival time convenient	int64	출발/도착 시간 편의성 만족도
Ease of Online booking	int64	온라인 예약 편리성 만족도
Gate location	int64	게이트 위치 만족도
Food and drink	int64	음식 및 음료 만족도
Online boarding	int64	온라인 체크인(탑승 수속) 만족도
Seat comfort	int64	좌석 편안함 만족도
Inflight entertainment	int64	기내 오락 시설에 대한 만족도
On-board service	int64	기내 승무원 서비스에 대한 만족도

Column Name	Data Type	Description
Leg room service	int64	다리 공간 서비스에 대한 만족도
Baggage handling	int64	수하물 처리 만족도
Checkin service	int64	체크인 서비스에 대한 만족도
Inflight service	int64	전반적인 항공기 서비스에 대한 만족도
Cleanliness	int64	기내 청결에 대한 만족도
Departure Delay in Minutes	float64	출발 지연 시간 (분 단위)
Arrival Delay in Minutes	float64	도착 지연 시간 (분 단위)
satisfaction	object	항공사 만족도 (만족, 중립 또는 불만족)

2. 데이터 전처리

2-1. 결측치 제거

데이터의 맨 처음 열인 index는 id가 그 역할을 대신 할 수 있으므로 지울 수 있다.

```
# 불필요한 열 삭제 (예: 'Unnamed: 0' 제거)
airplane_data_cleaned = airplane_data.drop(columns=['Unnamed: 0'])

# 결측치 확인
missing_values_summary = airplane_data_cleaned.isnull().sum()

missing_values_summary
```

- 프린트 결과

```
id                                  0
Gender                              0
Customer Type                       0
Age                                 0
Type of Travel                      0

...

Cleanliness                         0
Departure Delay in Minutes          0
Arrival Delay in Minutes          310
satisfaction                        0
dtype: int64
```

Arrival Delay in Minutes는 값의 범위는 0~1584로 순서형 데이터가 아닌 연속형 데이터 이기에, 이를 평균 값의 형태로 넣기에는 부적절한 것으로 보여 dropna 로 제거하는 방식을 사용했다.

```
# 결측치가 있는 행 삭제 ('Arrival Delay in Minutes' 열의 결측치가 있는 행 제거)
airplane_data_cleaned = airplane_data_cleaned.dropna(subset=['Arrival Delay in Minutes'])
```

2-2. 이상치 처리

```
# 수치형 변수 리스트
numerical_columns = ['Age', 'Flight Distance', 'Inflight wifi service',
                     'Departure/Arrival time convenient', 'Ease of Online booking',
                     'Gate location', 'Food and drink', 'Online boarding',
                     'Seat comfort', 'Inflight entertainment', 'On-board service',
                     'Leg room service', 'Baggage handling', 'Checkin service',
                     'Inflight service', 'Cleanliness', 'Departure Delay in Minutes',
                     'Arrival Delay in Minutes']


# IQR(Interquartile Range) 방법을 사용하여 이상치를 확인하는 함수 정의
def detect_outliers_iqr(data, column):
    Q1 = data[column].quantile(0.25)
    Q3 = data[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    outliers = data[(data[column] < lower_bound) | (data[column] > upper_bound)]
    return outliers

# 수치형 변수들에 대해 이상치 탐지
outlier_summary = {}

for column in numerical_columns:
    outliers = detect_outliers_iqr(airplane_data_cleaned, column)
    outlier_summary[column] = len(outliers)

# 이상치가 있는 변수와 그 개수를 출력
outlier_summary
```

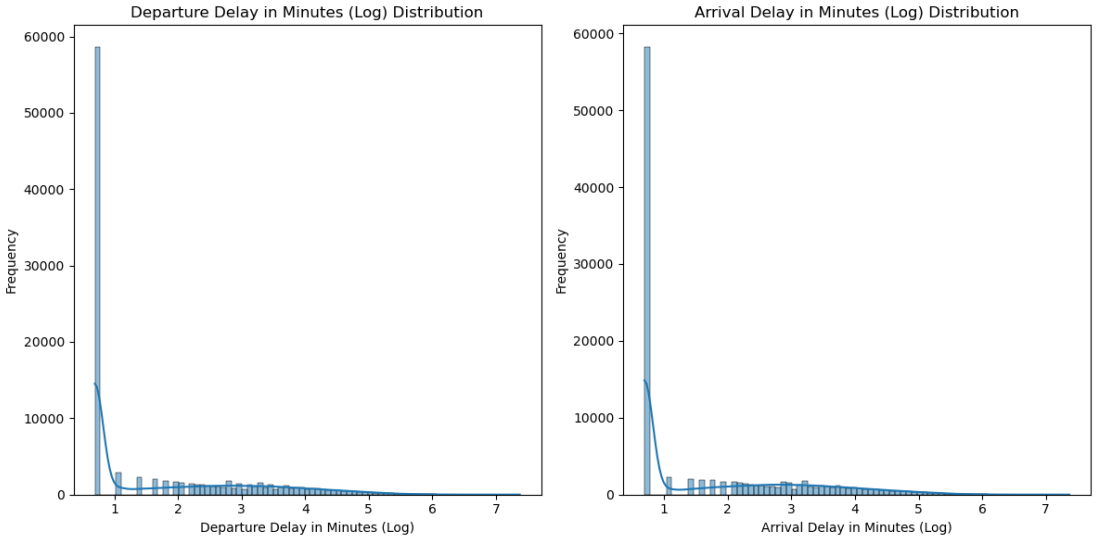
Columns	Count
Age	0
Flight Distance	**2291**
Inflight wifi service	0
Departure/Arrival time convenient	0
Ease of Online booking	0
Gate lcoation	0
Food and drink	0
Online boarding	0
Seat comfort	0
Inflight entertainment	0
On-board service	0
Leg room service	0
Baggage handling	0
Checkin service	**12891**
Inflight service	0
Cleanliness	0
Departure Delay in Minutes	**14529**
Arrival Delay in Minutes	**13954**

- **Flight Distance**: 2,291개의 이상치
- **Checkin service**: 12,891개의 이상치
- **Departure Delay in Minutes**: 14,529개의 이상치
- **Arrival Delay in Minutes**: 13,954개의 이상치

Flight Distance 와 Checkin service 는 0~5 의 순서형 데이터이므로 문제가 되지않지만, Departure Delay in Minutes 와 Arrival Delay in Minutes의 경우 두개의 항목의 경우 항공이 천재지변이나 공항의 문제등으로 **일반적인 서비스수준의 지연 대비 많은 시간이 딜레이 될 수 있음을 생각해야한다.**

이를 그대로 모델링에 적용할 경우 **이상치**들이 큰 영향을 줄 수 있으므로 로그변환을 통해 큰 값의 영향을 줄이고 작은 값들을 더 분포가 고르게 하여 이상치의 영향을 줄이도록 만들어야한다.

```
import numpy as np

# 로그 변환 전에 0 또는 음수 값이 있는지 확인하고, 1을 더해 양수로 변환
airplane_data_cleaned['Departure Delay in Minutes'] = airplane_data_cleaned['Departure Del
ay in Minutes'] + 1
airplane_data_cleaned['Arrival Delay in Minutes'] = airplane_data_cleaned['Arrival Delay i
n Minutes'] + 1

# 로그 변환 적용
airplane_data_cleaned['Departure Delay in Minutes (Log)'] = np.log(airplane_data_cleaned
['Departure Delay in Minutes'])
airplane_data_cleaned['Arrival Delay in Minutes (Log)'] = np.log(airplane_data_cleaned['Ar
rival Delay in Minutes'])

# 변환 결과 확인 (분포 시각화)
plt.figure(figsize=(12, 6))

# Departure Delay in Minutes (Log)
plt.subplot(1, 2, 1)
sns.histplot(airplane_data_cleaned['Departure Delay in Minutes (Log)'], kde=True)
plt.title('Departure Delay in Minutes (Log) Distribution')
plt.xlabel('Departure Delay in Minutes (Log)')
plt.ylabel('Frequency')

# Arrival Delay in Minutes (Log)
plt.subplot(1, 2, 2)
sns.histplot(airplane_data_cleaned['Arrival Delay in Minutes (Log)'], kde=True)
plt.title('Arrival Delay in Minutes (Log) Distribution')
plt.xlabel('Arrival Delay in Minutes (Log)')
plt.ylabel('Frequency')

plt.tight_layout()
plt.show()
```



다른 컬럼의 만족도의 범주가 0~5이므로 출발/도착 딜레이 대한 부분 역시 minmax_scaler 를 활용해서 0~5안에 값으로 가두는 일을 진행한다.

또한, 딜레이의 값은 0에 수렴할 수록 좋기때문에 , 분류모델에서 더 좋은 정확도를 위해 스케일을 반전 시킬 필요가 있다.

```
# Min-Max 스케일링 적용: 딜레이 변수들을 0~5 사이로 정규화

columns_to_scale = ['Departure Delay in Minutes (Log)', 'Arrival Delay in Minutes (Log)']
minmax_scaler_0_to_5 = MinMaxScaler(feature_range=(0, 5))
airplane_data_cleaned[columns_to_scale] = minmax_scaler_0_to_5.fit_transform(airplane_data
_cleaned[columns_to_scale])

# 반전된 스케일링 적용: 값이 높을수록 좋은 평가를 나타내도록 변경 (분류 모델에 적합하도록)
# 반전된 스케일: 값이 높을수록 긍정적인 상태를 나타내도록 변경 (0에서 5 사이의 값을 반전)
airplane_data_cleaned[columns_to_scale] = 5 - airplane_data_cleaned[columns_to_scale]


# Min-Max 스케일링을 0~5 범위로 적용
minmax_scaler_flight = MinMaxScaler(feature_range=(0, 5))

# 정규화된 'Flight Distance'를 기존 'Flight Distance' 컬럼 옆에 추가
airplane_data_cleaned['Flight Distance (Scaled)'] = minmax_scaler_flight.fit_transform(air
plane_data_cleaned[['Flight Distance']])
```

2-3. 범주형 변수 원 핫 인코딩

선형 모델이나 트리기반 모델 들은 입력으로 수치형 데이터가 필요하며, 범주형 데이터를 이해하지 못하기 때문에, 이를 인코딩을 통해 수치형으로 변환하는 과정이 필요하다

클래스의 경우 3가지 부류가 있기때문에 boolean 형태로 만들었다

```
# Class 변수를 원-핫 인코딩으로 변환 (Business, Eco, Eco Plus 각각의 컬럼으로 분리)
airplane_data_encoded = pd.get_dummies(airplane_data_cleaned, columns=['Class'], prefix=
['Class'], drop_first=False)
```

```
# Class 변수가 Business, Eco, Eco Plus로 바뀌도록 명칭 변경
airplane_data_encoded = airplane_data_encoded.rename(columns={
    'Class_Business': 'Class_Business',
    'Class_Eco': 'Class_Eco',
    'Class_Eco Plus': 'Class_Eco_Plus'
})
```

	id	Gender	Customer Type	Age	Type of Travel	Class	Flight Distance	Inflight wifi service	Departure/Arrival time convenient	Ease of Online booking	...	Leg room service	Baggage handling	Checkin service	Inflight service	Cleanliness	Departure Delay in Minutes
0	70172	1	0	13	1	2	460	3	4	3	...	3	4	4	5	5	26
1	5047	1	1	25	0	0	235	3	2	3	...	5	3	1	4	1	2
2	110028	0	0	26	0	0	1142	2	2	2	...	3	4	4	4	5	1
3	24026	0	0	25	0	0	562	2	5	5	...	5	3	1	4	2	12
4	119299	1	0	61	0	0	214	3	3	3	...	4	4	3	3	3	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
103899	94171	0	1	23	0	1	192	2	1	2	...	1	4	2	3	2	4
103900	73097	1	0	49	0	0	2347	4	4	4	...	5	5	5	5	4	1
103901	68825	1	1	30	0	0	1995	1	1	1	...	2	4	5	5	4	8
103902	54173	0	1	22	0	1	1000	1	1	1	...	5	1	5	4	1	1
103903	62567	1	0	27	0	0	1723	1	3	3	...	1	4	4	3	1	1

103594 rows × 26 columns

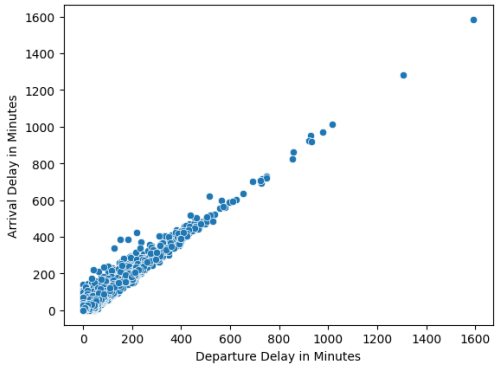
컬럼명 (변수)	값	설명 (범주)
Gender (성별)	0	Female (여성)
	1	Male (남성)
Customer Type (고객 유형)	0	Loyal Customer
	1	disloyal Customer
Type of Travel (여행 유형)	0	Business travel (비즈니스 여행)
	1	Personal Travel (개인 여행)
Class Business	0,1	
Class Eco	0,1	
Class Eco Plus	0,1	
satisfaction (만족도)	0	neutral or dissatisfied (중립 또는 불만족)
	1	satisfied (만족)

2-4. 다중공산성 문제 예측

상관관계를 통해 모델에 들어갈 컬럼에 대해 가감을 할 수 있는데, 이는 모델의 성능을 향상시키고, 분석의 효율성을 높히며, 직면하는 다양한문제에 해결에 있어서 반드시 필요하다.

중복된 내용을 제거하거나, 상관관계가 높은 컬럼들이 다수 입력되어 모델이 불안정해지는 다중공산성의 문제 또한 해결할 수 있으며, 이를 통해 모델이 과적합되는 것 또한 막을수 있다.
또한 y(종합만족도)와의 상관관계가 낮은 변수를 제거하여 노이즈를 제거하는데에 기여할 수 있다.

예를 들면 Departure Delay in Minutes와 **Arrival Delay in Minutes 와 상관계수는 0.97로 두 컬럼 모두 적용하게 된다면 편향된 모델을 만들어낼 수도 있다. 이의 경우 하나의 컬럼만 적용하는 식으로 편향을 막을 수 있다.**



컬럼테마 분류 (14개)

예약관련: Checkin service, Ease of Online booking, Online boarding, Departure/Arrival time convenient(예약시간에 대한 만족도? 예약한 시간이 얼마나 편리했는지)

음식만족도: Food and drink

좌석만족도: Seat comfort, Leg room service

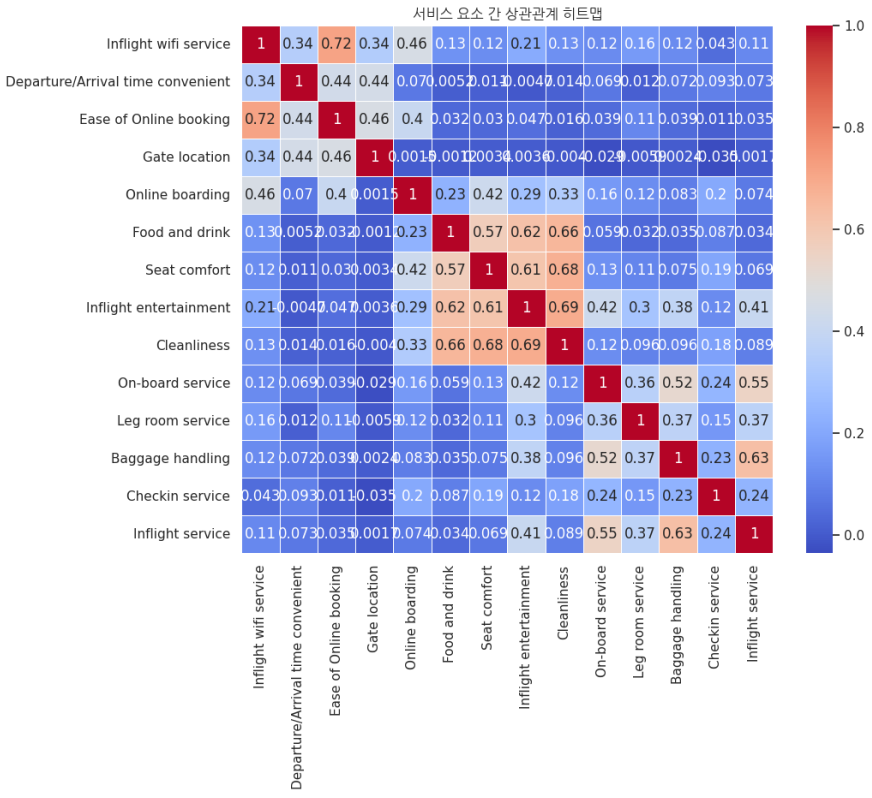
청결 관련 만족도: Cleanliness

승무원 서비스 만족도: On-board service

인터넷 관련 서비스 만족도: Inflight wifi service
기내 만족도 ?: Inflight service, Inflight entertainment

수하물 , 게이트 관련: Baggage handling, Gate location

위의 분류를 활용해 상관관계 분석을 하여, 가감 할 컬럼이 무엇인지 탐색할 것이다.



- 와이파이서비스 : 온라인예약의 편리함과 0.72의 상관관계, 온라인탑승 0.46의 상관관계

온라인예약의 편리함은 가정에서 항공편온라인예약에 대한 편리함(홈페이지 혹은 어플 이용의 편리함)으로 이해했는데 기내 와이파이서비스와 관련이 높은 걸 보니 비행기 내에서 여러 서비스를 예약하는 것을 의미할 수도 있다.

Food and drink, Seat comfort, Inflight entertainment, Cleaniness 네가지 요소가 서로 높은 상관계수를 가지기 때문에 (0.5~0.7 사이) 네가지 요소를 기내서비스로 묶을 수 있는지 확인해 볼 것이다.

비슷한 기내 서비스 요소의 그룹화 가능성

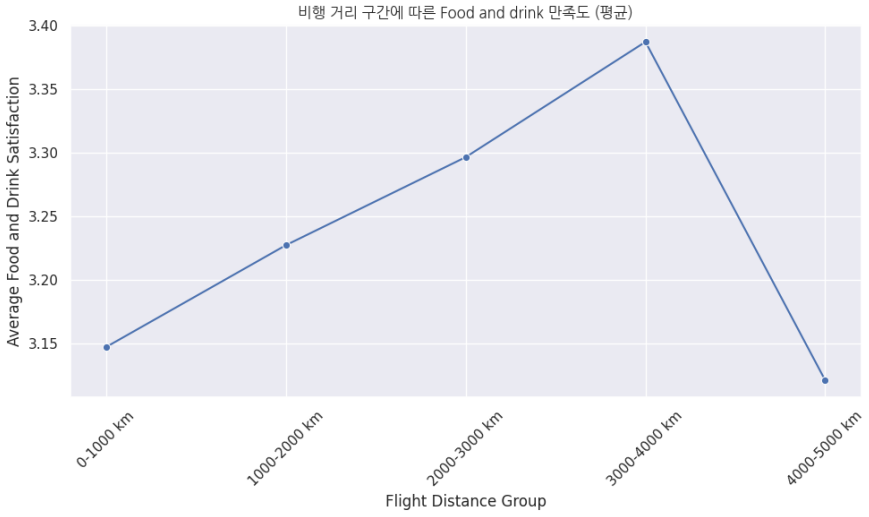
네가지를 기내서비스로 묶은 후 종합만족도와 어떤 상관관계를 보이는지 확인

```
# 1. 네 가지 요소를 기내 서비스 점수로 묶음 (평균 계산)
df['Inflight_service_score'] = df[['Food and drink', 'Seat comfort', 'Inflight entertainme
nt', 'Cleanliness']].mean(axis=1)

# 2. 종합 만족도(satisfaction)를 숫자로 변환 (만족: 1, 불만족: 0)
df['satisfaction_numeric']

# 3. 기내 서비스 점수와 종합 만족도의 상관관계 계산
correlation = df[['Inflight_service_score', 'satisfaction_numeric']].corr()

# 4. 상관관계 출력
print("기내 서비스와 종합 만족도의 상관관계:", correlation)
print(correlation)
```

기내 서비스와 종합 만족도의 상관관계: **0.369475**

Food and drink는 종합 만족도에 대한 상관계수는 낮지만, 다른 요소들과의 상관관계가 높음

- 추가) 비행 거리에 따른 food and drink 만족도



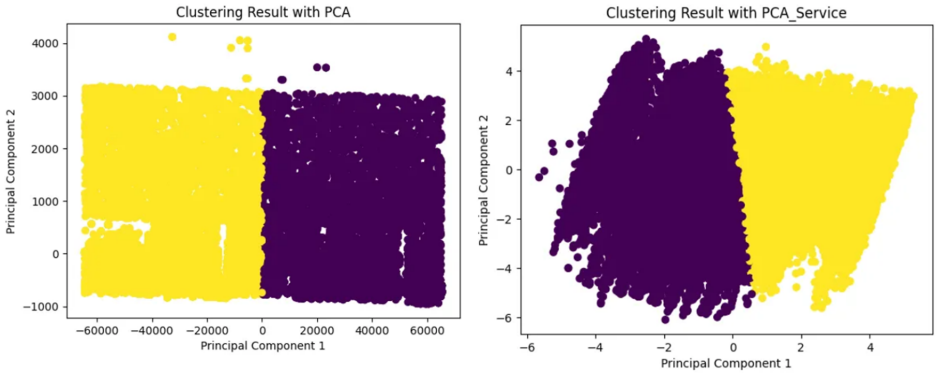
2-5. neutral or dissatisfied 분리 가능

- 다음은 satisfaction의 value_counts값이다.

satisfaction	count
neutral or dissatisfied	58879
satisfied	45025

- 크게 봤을 때 satisfied의 값이 더 적은 것을 알 수 있는데, 그 요인으로 neutral과 dissatisfied를 서로 묶어놓은 이유가 크다고 보고 있다. 하지만 정확한 값을 우리가 알 수 없기에 이 둘을 임의로 분리를 시킬 수 있을지 확인해보려고 한다.
- 그렇기 위해 우리는 PCA를 통해 전체적인 데이터로 차원축소를 한번, 만족도 조사와 관련된 데이터로만 차원축소를 한번, 이렇게 총 2번의 차원축소를 통해 분류 확인을 해보았다.

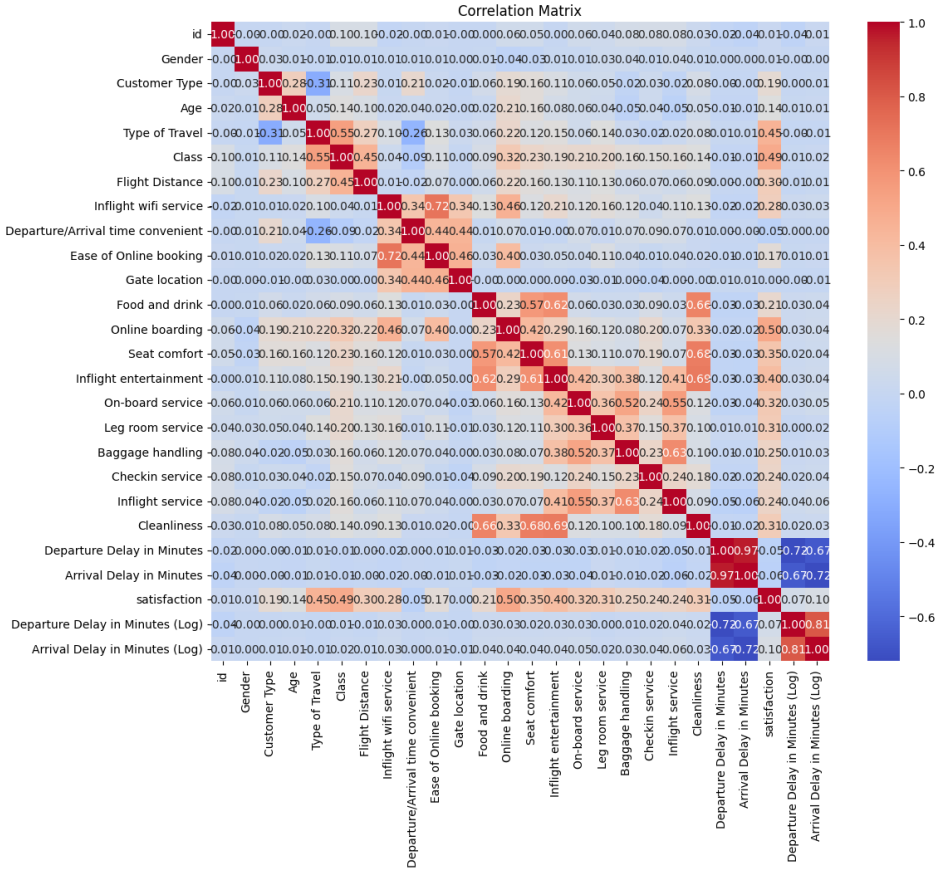
- 차원축소 결과 (전체 데이터, 만족도 조사 데이터)



- 다음과 같이 군데군데 데이터가 밀집이 되어있는 걸 볼 수 있으며, 겉으로만 보아도 깔끔한 군집이라고 하기에는 어려운 모습을 하고 있다. 이런 그래프는 어떤 식으로 선을 그어도 크게 의미가 없을 거라고 판단이 들었다.
- 그래서 neutral과 dissatisfied와의 분류를 그만두고, 오로지 satisfied를 판단하는거에 초점을 두기로 하였다.

3. EDA(탐색적 데이터 분석)

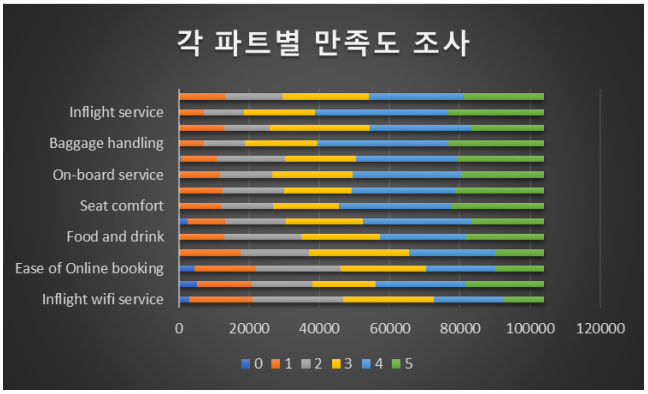
3-1. 전체 상관관계 분석



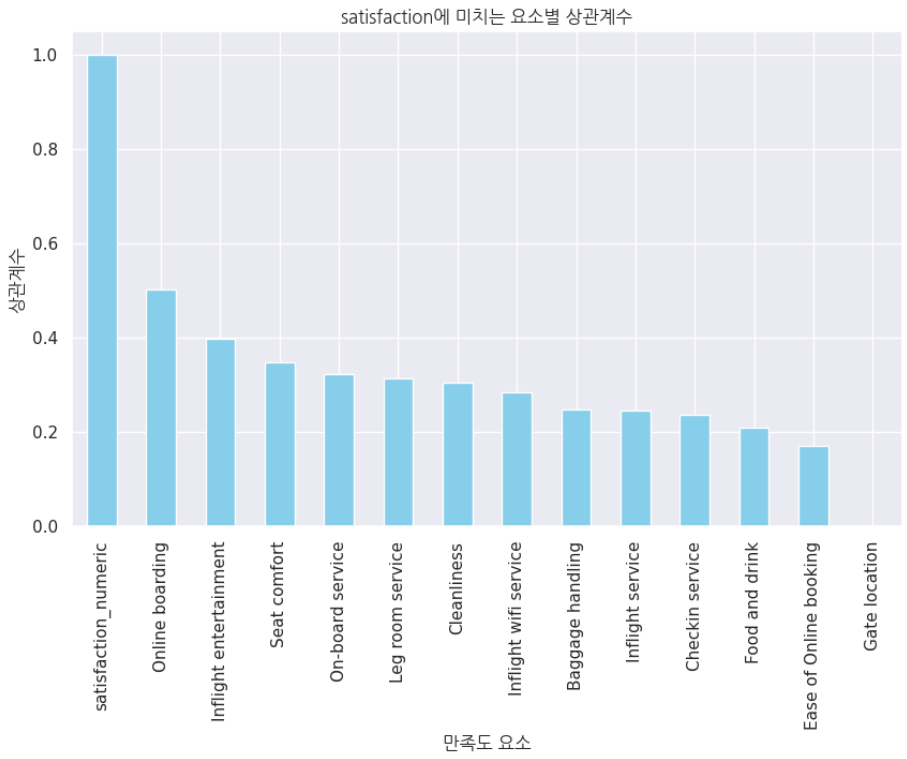
- **온라인 예약의 편리함과 비행 중 와이파이 서비스 사이에도 높은 상관관계가 있음.(0.72)**
- **청결도와 (음식 및 음료(0.66), 좌석 편안함(0.68), 기내 엔터테이먼트(0.69)) 사이에도 높은 상관관계가 있음**
- Type of Travel, Class, Online boarding 3가지 타입이 satisfaction과 가장 직접적인 관계가 보임
- 기내 전반 서비스 - 기내 엔터테이먼트, 탑승 서비스, 수하물 처리 서비스

각 서비스 만족도와 종합 만족도의 연관성

- 각 파트별 만족도 조사



```
# satisfaction을 0과 1로 변환 (불만족/만족)
df['satisfaction_numeric'] = df['satisfaction'].apply(lambda x: 1 if x == 'satisfied' else
0)

# 1. satisfaction과 서비스 요소 간 상관계수 계산
correlation_4 = df[service_columns + ['satisfaction_numeric']].corr()['satisfaction_numeri
c'].sort_values(ascending=False)
print("satisfaction에 대한 각 요소의 상관계수:")
print(correlation_4)

# 2. 막대 그래프 시각화
plt.figure(figsize=(10,6))
correlation_4[:-1].plot(kind='bar', color='skyblue')
plt.title('satisfaction에 미치는 요소별 상관계수',fontproperties=fontprop)
plt.ylabel('상관계수',fontproperties=fontprop)
plt.xlabel('만족도 요소',fontproperties=fontprop)

plt.show()
```



- satisfaction에 대한 각 요소의 상관계수

satisfaction에 대한 각 요소	상관계수
satisfaction_numeric	1.000000
Online boarding	**0.503447**
Inflight entertainment	0.398203
Seat comfort	0.349112
On-board service	0.322450
Leg room service	0.313182

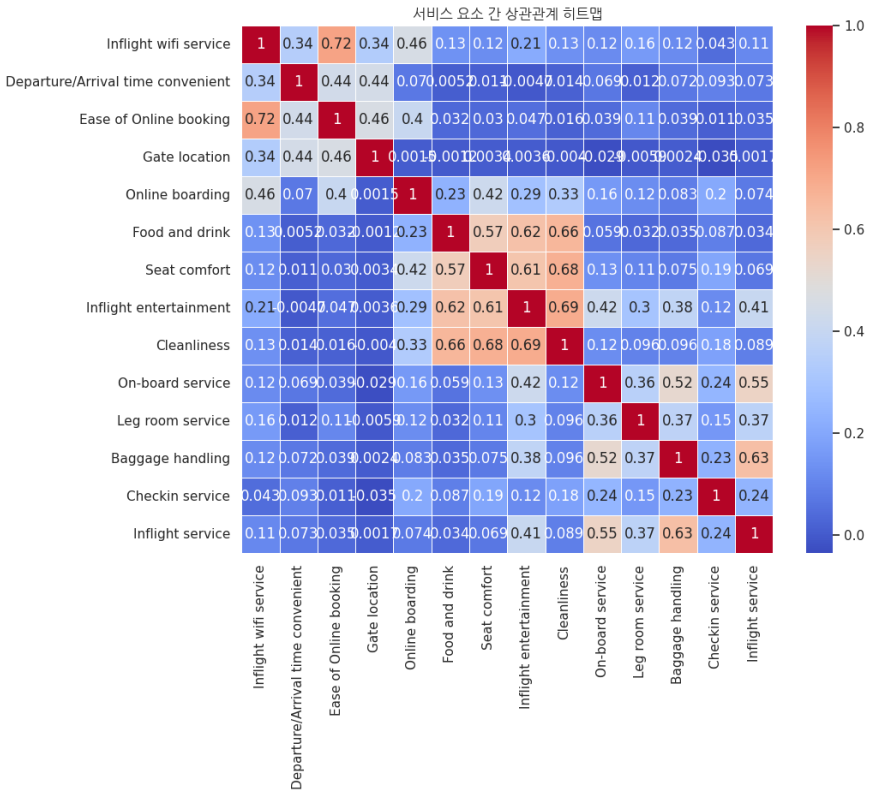
satisfaction에 대한 각 요소	상관계수
Cleanliness	0.305050
Inflight wifi service	0.284163
Baggage handling	0.247819
Inflight service	0.244852
Checkin Service	0.235914
Food and drink	0.209659
Ease of Online Booking	0.171507
Gate location	**0.000449**
Departure/Arrival time convenient	**0.051718**

- **Online boarding (온라인 체크인)**이 0.5로 가장 높은 상관계수를 보이며, **Inflight entertainment (기내오락시설)**, **Seat comfort (좌석 편안함)**, **On-board service (기내 승무원 서비스 만족도)가 그 뒤를 이었다.**

  인터넷 서비스, 수화물,기내식은 어느 정도 영향을 미치지만, 상대적으로 덜 중요한 요소로 분석되었다.

- **Gate location** (게이트 위치)와 **Departure/Arrival time convenient** (출발/도착 시간의 편리성)은 거의 종합 만족도에 영향을 미치지 않았다.
  항공사의 통제 밖에 있는 외부적인 요인으로 인식했기 때문에, 모델에 적용할 때 제외 할 요소로 정할 것이다.

3-2. 서비스 요소 간 상관관계 히트맵



<서비스요소간 히트맵 해석>

- leg room service와 Seat comfort 는 상관관계가 0.1 로 낮으며, 이는 승객들이 좌석의 편안함과 다리 공간을 별개의 요소로 평가할 가능성이 있음을 의미함.
  - 두가지요소를 각각 별개로 인식해야하므로 컬럼을 합치기엔 어려움이 있음

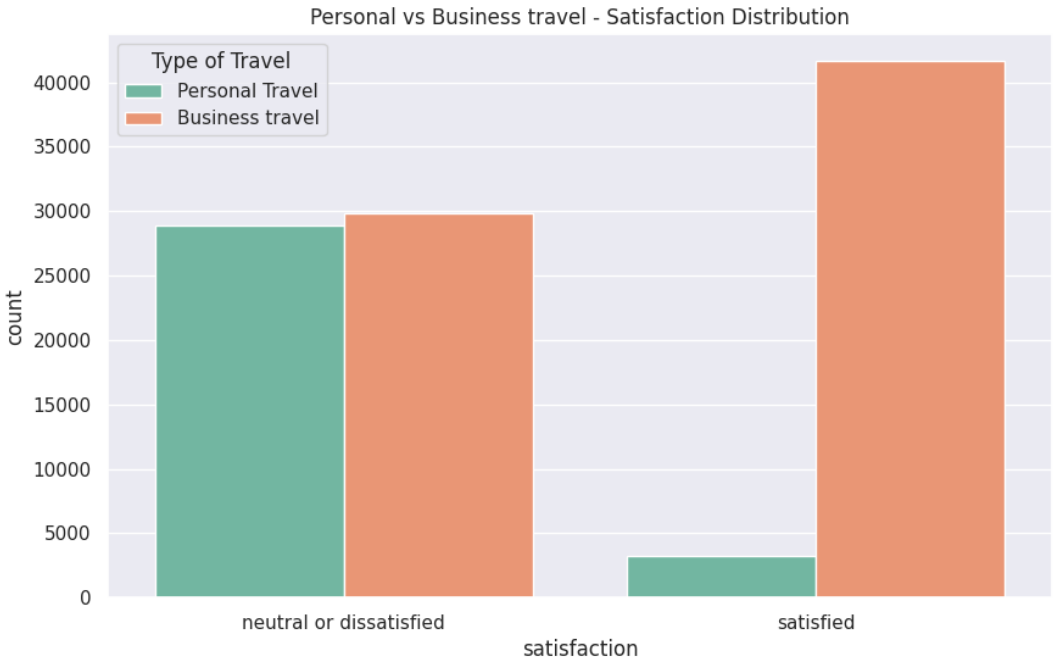
- On-board service (승무원만족도)을 기준으로 높은 상관관계를 가지는 항목은 다음과 같다.
  - 전반적인 비행서비스 (0.5 이상)
  - 수화물 운반 (0.5 이상)
  - 기내 오락시설 (0.4 이상)

- 각 항목별 만족도 조사

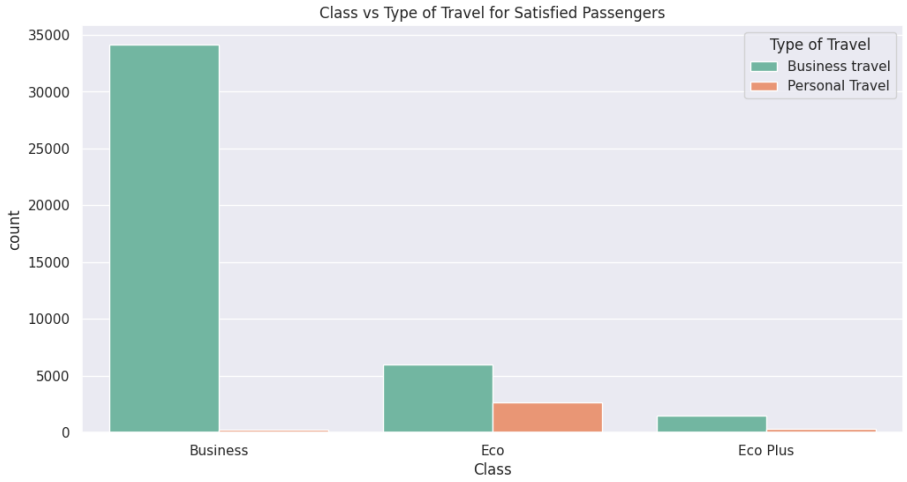
	Inflight wifi service	Departure/Arrival time convenient	Ease of Online booking	Gate location	Food and drink	Online boarding	Se
0 - satisfied	3095	2520	2980	1	50	1351	0
0 - dissatisfied	8	2780	1507	0	57	1077	1
1 - satisfied	5806	7541	6590	8703	2568	1473	26
1 - dissatisfied	12034	7957	10935	8859	10269	9219	93
2 - satisfied	6423	7657	7301	8965	8530	2019	33
2 - dissatisfied	19407	9534	16720	10494	13458	15486	115
3 - satisfied	6482	7873	7537	9922	8839	2959	39
3 - dissatisfied	19386	10093	16912	18655	13461	18845	147
4 - satisfied	11856	9906	10391	9490	12788	19166	178
4 - dissatisfied	7938	15640	9180	14936	11571	11596	139
5 - satisfied	11363	9528	10226	7944	12250	18057	172
5 - dissatisfied	106	12875	3625	5935	10063	2656	92

3-3. 항공 클래스 & 여행타입 기준 분석

- 여행유형 Personal travel vs Business travel를 구분해봄.



- 불만족 분포는 비슷하나 비즈니스 여행이 만족 카운트가 훨씬 높음
  - 비즈니스 목적의 사람들은 비즈니스 좌석일 확률이 높음.
  - personal , business 또는 eco , eco plus , business으로 구분할지
  - 또는 2x3 으로 구분할지 class 로만 구분할지

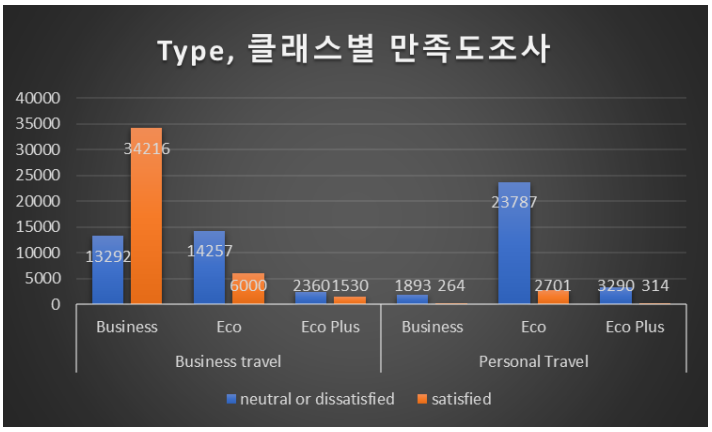


- **personal travel** 승객은 주로 **Eco**나 **Eco Plus** 클래스를 이용하고 business travel 인경우는 Business class 인경우가 압도적이므로 2*3으로 복잡하게 구분하기보다는, **Class**별로 간단하게 구분하는 것이 더 효과적일 수 있음
    - 그러나 Eco class에서 여행유형타입 차이가 드러나므로 타입에 따른 만족도 차이가 있는지 파악해본다.

- 같은 eco클래스여도 여행타입별 만족도 차이가 존재함.
  비즈니스 여행일 경우 만족비율이 높게 나타난다.
    - **Business travel** 승객은 **시간**이나 **편리함**을 중요하게 생각할 가능성이 크다.
      이들에게는
      탑승 절차, **시간 준수**, **좌석 배정** 등에서 우선적인 서비스가 제공될 수 있다.
    - 반면, **Personal travel** 승객은 **편안함**과 **즐거움**을 중시할 가능성이 있음.

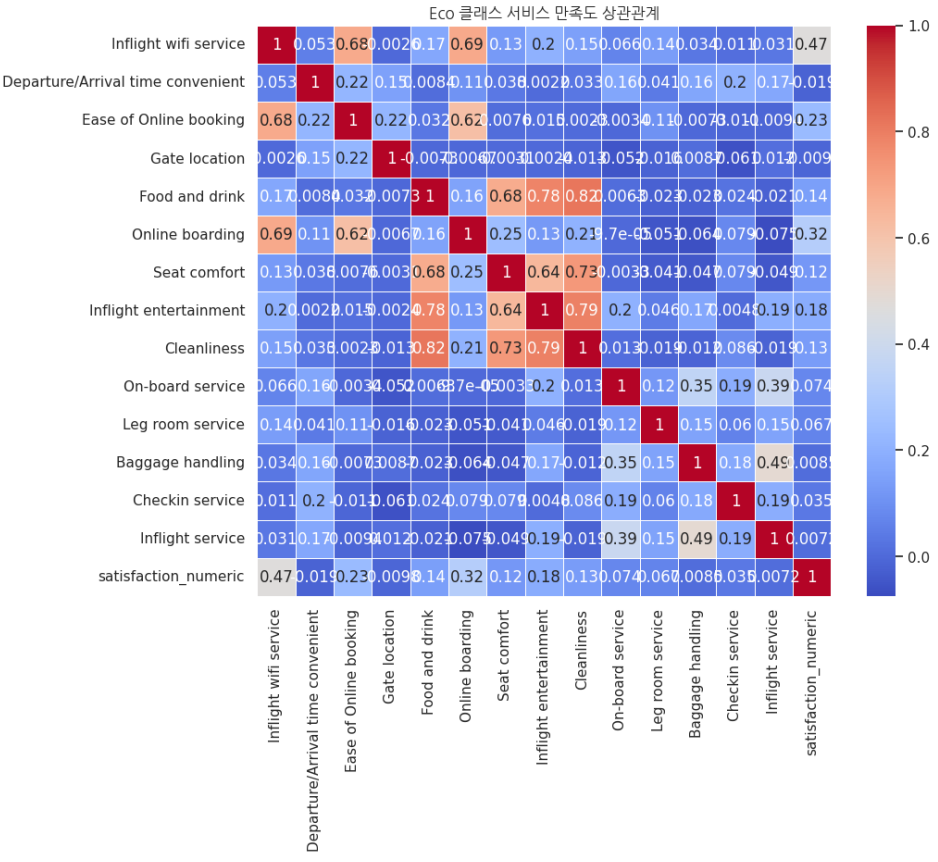
      기내 오락 시설이나 **음식**, **승무원 서비스**가 더 중요하게 평가될 수 있다.
    - 즉, 고객의 특성에 따른 만족도 차이가 있을 수 있음
- 비즈니스 경우에는 여행타입을 구분하지 않고, eco인 경우에만 구분하는 것도 방법이 될 수 있음

- Type, 클래스별 만족도 조사



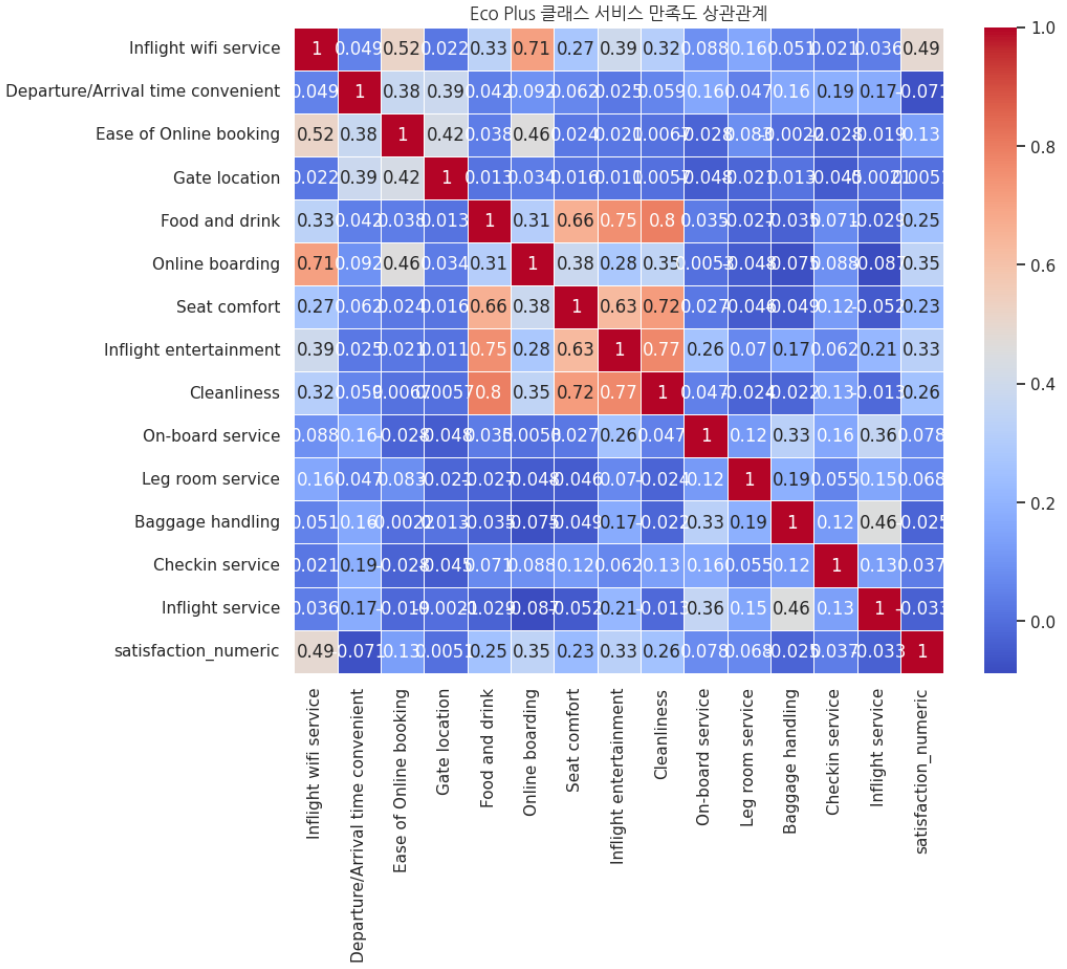
클래스별 종합만족도에 영향을 미치는 요인 분석

- 지금까지 분석한 것을 토대로,종합 만족도는 class에 따른 차이가 크다는 것을 발견함.
  따라서 각 서비스별 만족도 및 종합만족도(satisfaction_numeric) 를 class별로 나누어서 상관분석을 진행함.

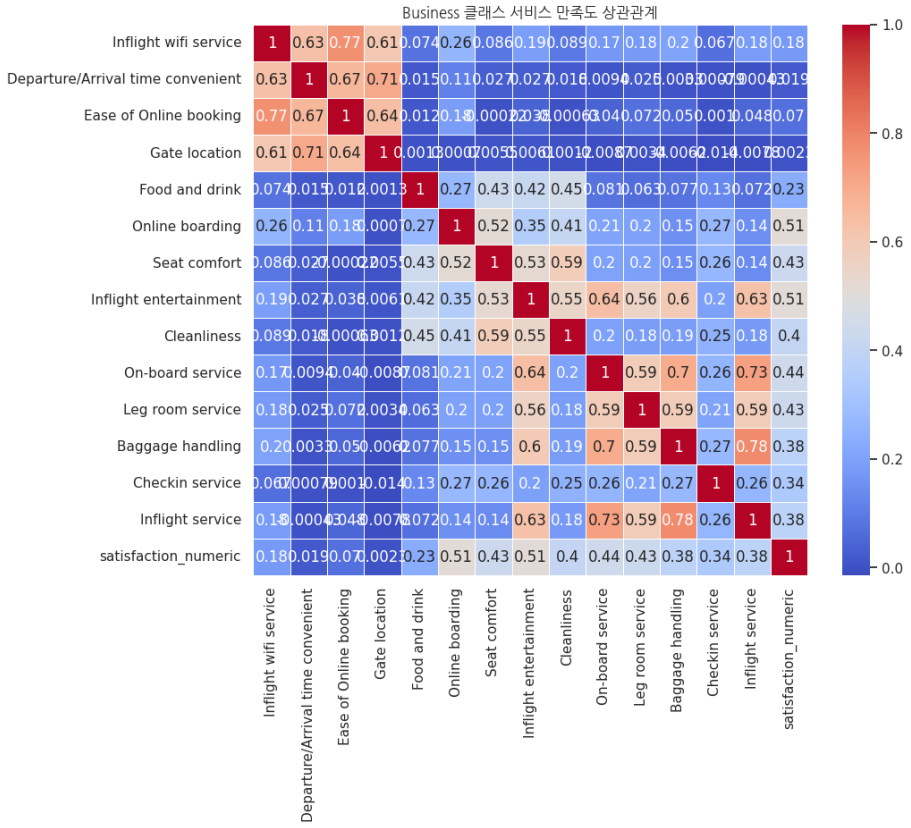


1. 앞서 기내서비스로 묶었던 4개의 카테고리간 상관관계가 매우 강하게 나타남
2. 와이파이서비스- 온라인boarding 간의 관계가 전체에서는 0.46이었는데 0.69로 상승함
   - eco에서는 와이파이-온라인boarding의 만족도가 비슷한 방향으로 흘러간다.

3. 종합만족도와의 관계를 보면, 와이파이와 0.47의 관계를 보이고, 온라인예약편리성, 온라인 boarding 세가지 요소와 약한 상관관계가 있다. (온라인관련 3개카테고리의 영향 높음)
4. 기내서비스(4개 카테고리)와 평균 0.13쯤의 관계가 있으며 나머지는 관계가 없다고 해도 무방하다.
5. 그 외에는 전체 상관분석 했을 때와 수치적인 차이 외에 전체적인 경향은 크게 다르지 않음.



- eco와 eco plus는 매우 비슷한 양상을 보임. 그렇기에 ***이 2개의 요소를 결합할 가능성을 야기할 수 있음.***



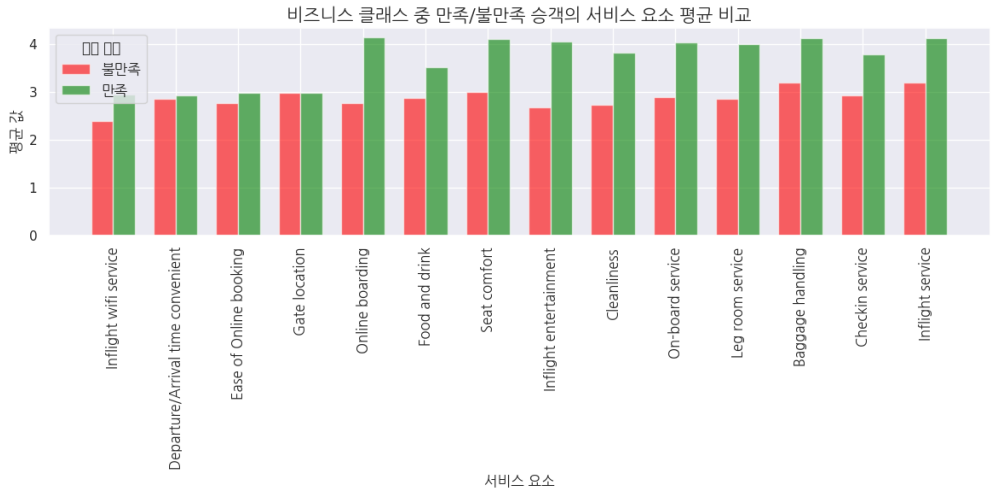
- **<u>반면에 비즈니스 클래스의 상관분석 양상은 매우 다르다.</u>**
  - 비즈니스에서는 와이파이, 출발시간의 편리성, 온라인예약의 편리함, 게이트위치 4개의 상관관계가 높게 나왔으며 **기내서비스관련해서는 상관관계가 다소 낮게 나옴**
  - 또한 비행서비스(inflight service)와 기내오락시설, on_board service, lag room service, 수화물서비스간 관계가 높게나옴 (0.6~0.7)

- 전체 만족도 관계에서 online boarding , 기내 오락시설과 0.51로 가장 큰 상관관계를 보이며
  전반적으로 0.3이상의 상관관계를 보임
  .
  와이파이서비스는 0.18로 매우 낮음
  - 이는 eco class와 큰 차이점이며, 비즈니스는 와이파이, 출발시간의 편리성, 온라인예약의 편리함, 게이트위치 4가지 요소(4개간의 상관관계가 높은것을 볼때 4개의 만족도가 비슷하지만, 종합만족도에 영향을 미치지 않을것) 를 제외한 나머지 모든 요소가 종합점수에 영향을 미친다는 것을 알 수 있다.

비즈니스 class에 대한 추가분석

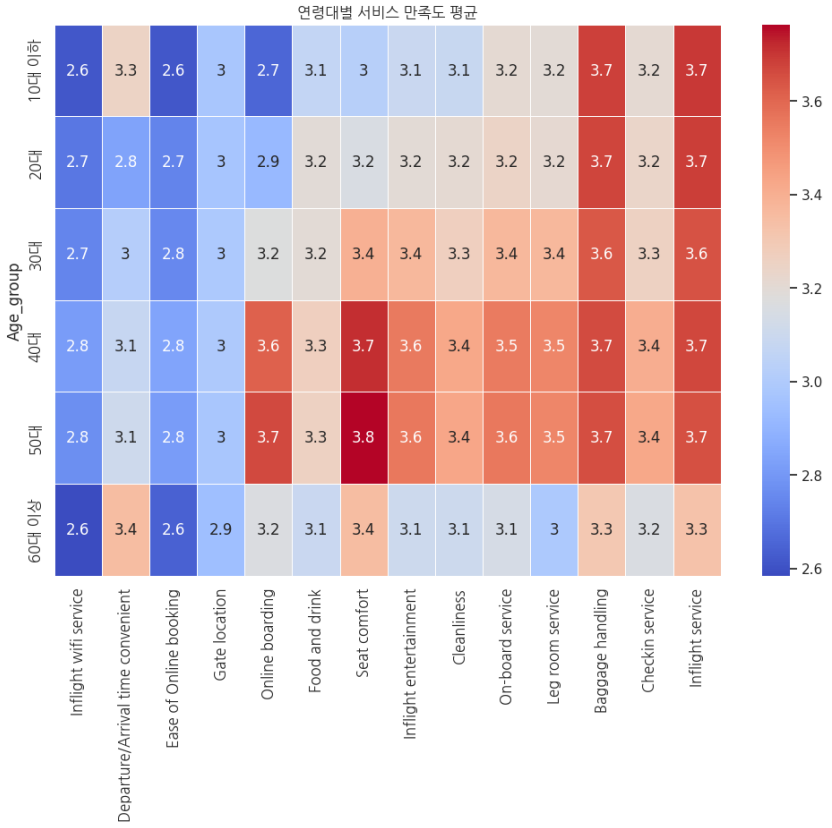
따라서 비즈니스 클래스의 만족도 상관분석이 전체 만족도 분석과 차이가 있으므로 비즈니스클래스에 대한 만족/ 불만족인 사람의 특성을 추가적으로 파악해본다.

Business Class 서비스 평균값	불만족한 승객들	만족한 승객들
Inflight wifi service	2.390874	2.944461
Departure/Arrival time convenient	2.863501	2.924455
Ease of Online booking	2.759691	2.982059
Gate location	2.978340	2.985112
Online boarding	2.774285	4.131259
Food and drink	2.878822	3.518348
Seat comfort	2.989566	4.100378
Inflight entertainment	2.681569	4.055394
Cleanliness	2.723701	3.809334
On-board service	2.892954	4.025996
Leg room service	2.853860	3.992876
Baggage handling	3.199432	4.126112
Checkin service	2.920029	3.782728
Inflight service	3.201677	4.127479



- **와이파이, 출발편리함, 온라인예약의 편리함, 게이트위치에 대한 만족도**는 전반적으로 낮으며 만족/불만족 결과에 영향을 미치지 않음
  - 이는 상관분석결과와 일치한다.
  - 비즈니스 좌석의 고객만족도를 예측할 경우 이 열을 제외시키는 방향을고려해볼 수 있다.
- 그외의 모든 열이 전반적으로 3.5 이상을 기록했을때 승객이 만족했음을 알 수 있고, 반대로 모든열이 3점 이하일때 승객이 불만족이라고 답했다.

3-4. 연령대별 분석

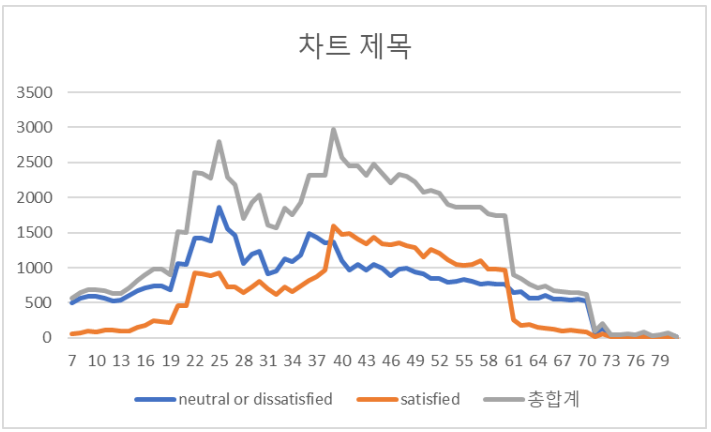


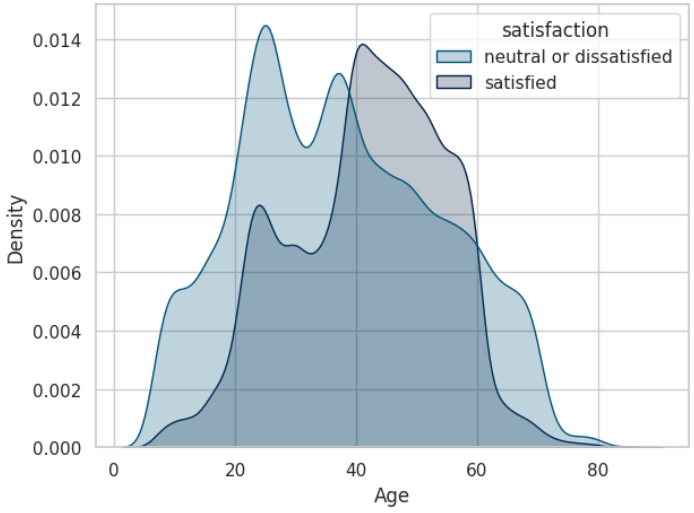
- 40~50대 승객의 만족도가 전반적으로 높음
  - 이에 대한 요인으로 비즈니스를 탈 확률이 높다고 생각할 수 있다. 따라서
    비즈니스 클래스
    이용률이 높은지 확인하고, **클래스별 만족도 차이**를,
    더 나아가 연령대별 나타나는 만족도 평점 차이가 유의미한지 파악한다.
- 나이별 상관관계 상위 3개

	1	2	3
10대 미만	Inflight wifi service (0.27)	Online boarding (0.26)	Ease of Online booking (0.21)
10대	Online boarding (0.41)	Inflight wifi service (0.35)	Class (0.31)
20대	Online boarding (0.47)	Class (0.42)	Inflight wifi service (0.32)
30대	Online boarding (0.54)	Inflight entertainment (0.42)	Class (0.39)
40대	Class (0.51)	Type of Travel (0.50)	Online boarding (0.49)
50대	Class (0.56)	Type of Travel (0.56)	Leg room service (0.50)
60대	Type of Travel (0.48)	Class (0.41)	Leg room service (0.37)
70대 이상	Inflight entertainment (0.37)	Leg room service (0.35)	Inflight wifi service (0.33)

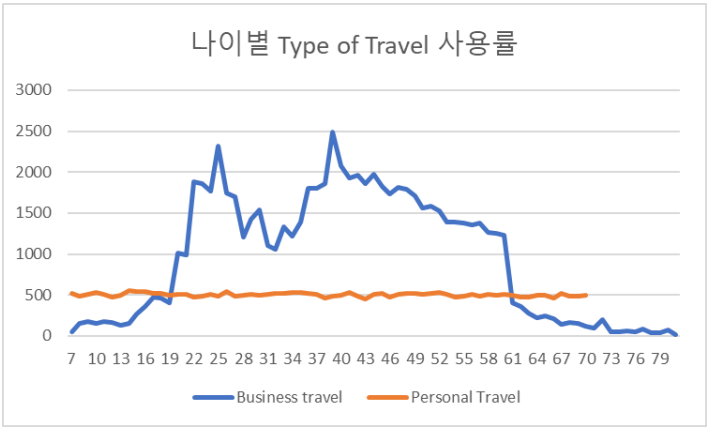
- 연령과 상관없이 와이파이서비스, **출발시간편리성, 온라인예약편리성, 게이트위치**에 대한 만족도는 낮게 나옴
  - 그러나 와이파이서비스를 제외한 3가지 요소는 위의 상관관계분석에 따르면 종합적인 만족도에 큰 영향을 미치지 않기 때문에 이것이 전체 평점에 영향을 주는지는 추가적인 분석이 필요함
- 수화물 운반, 기내서비스에 대한 만족도는 전 연령대가 높다.
  - 60대만 유일하게 수화물서비스 만족도가 다소 낮은데 이는 노인에 대한 서비스가 부족한 것을으로 볼 수 있다.
- 앞서 Online boarding이 0.5로 종합 만족도와 가장 상관관계가 높았는데 40,50대를 제외한 연령대에서 낮게 나옴. 특히 10대, 20대가 2점대로 낮음
  - **10~20대의 종합만족도가 불만족일 확률이 높다.**

- 연령대별 만족도 조사



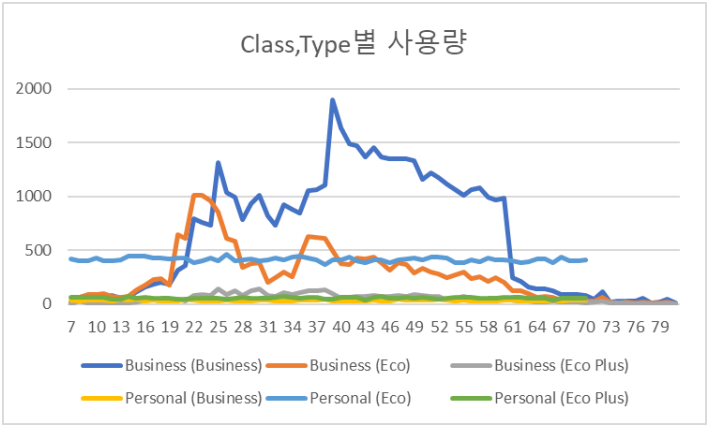


- 크게 20대와 40대 중반 이후로 군집이 형성이 되어있다. 20대 구간에서는 상대적으로 불만족 의사가 높게 띄어있으며, 40대 중반 구간에서는 만족도가 더 높은 것을 볼 수 있다.
    - 이걸 통해 처음으로 추측한 내용이, '20대는 여행 목적으로 비행기를 주로 타고, 중년 이후는 사업과 비즈니스 목적으로 비행기를 많이 탈 것이다. 그래서 이 항공사는 비즈니스에 특화되어 Eco를 주로 쓰는 젊은 층한테 안 좋은 것이다' 이였다.
- 나이별 Type of Travel 이용률



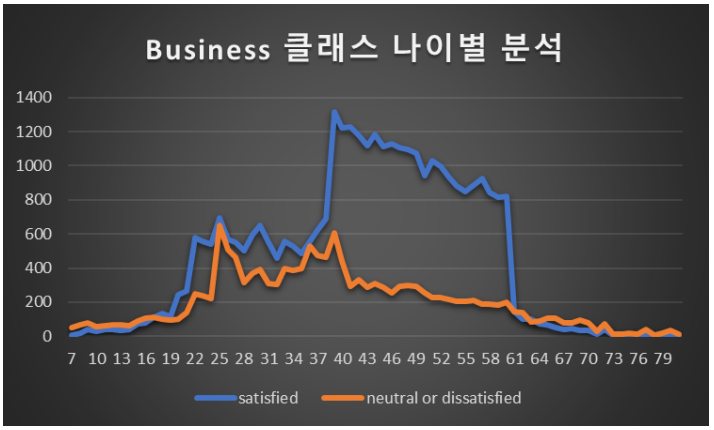
    - 하지만 실제로 보면 그렇지 않다. 사람들은 이 항공사를 비즈니스 목적으로 이용한다. 그렇다는 건 나이대별로 같은 비즈니스라도 느끼는게 다를 수 있다는 얘기다.

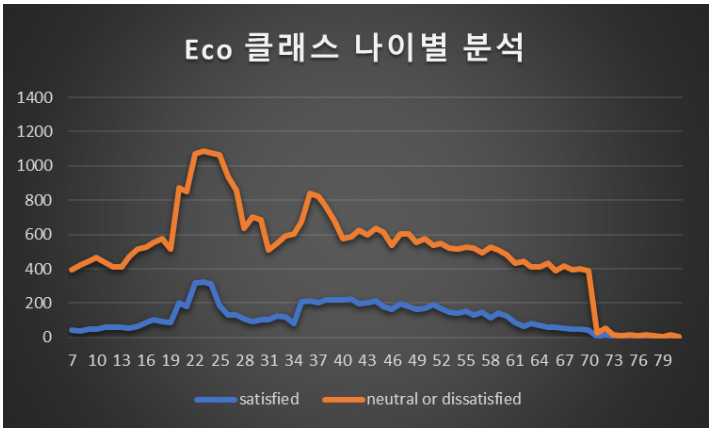
- Type,클래스 별 사용량



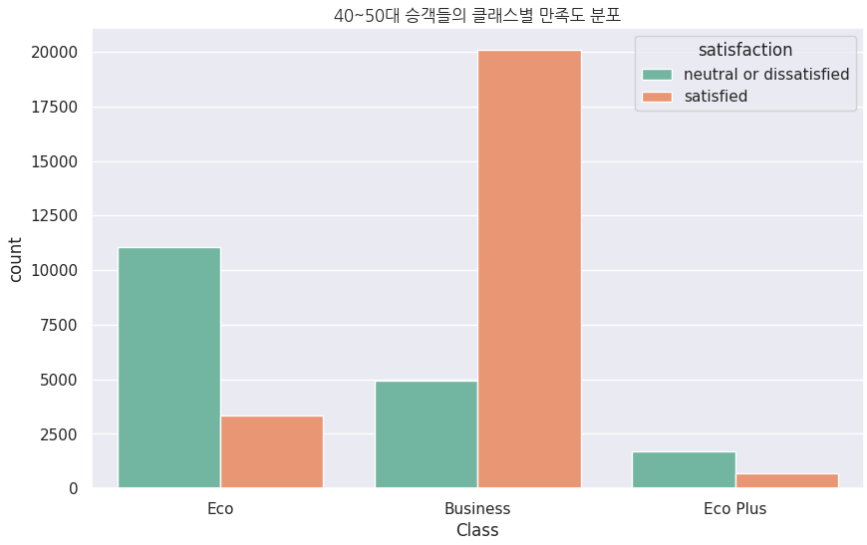
  - 다만 비즈니스 목적으로 항공사를 이용한다고 해도 경제적 여유에 따라 타는 비행기가 다를 순 있다. 그걸 입증해 주는게 다음 표인데, **젊은 사람들은 비즈니스 목적이라도 경제적 여유를 받쳐주지 못하는지 Eco석을 주로 탄다**. 물론 비즈니스석을 타는 젊은 층도 보이지만, 뒤로 갈수록 Eco석 비즈니스맨은 줄어들고 비즈니스석이 호황을 이루는 걸 볼 수 있다.
  - type 보다는 class가 만족도에 큰 영향을 미친다고 볼 수 있음

각 클래스(Eco, Business, EcoPlus(X)), 나이별 분석



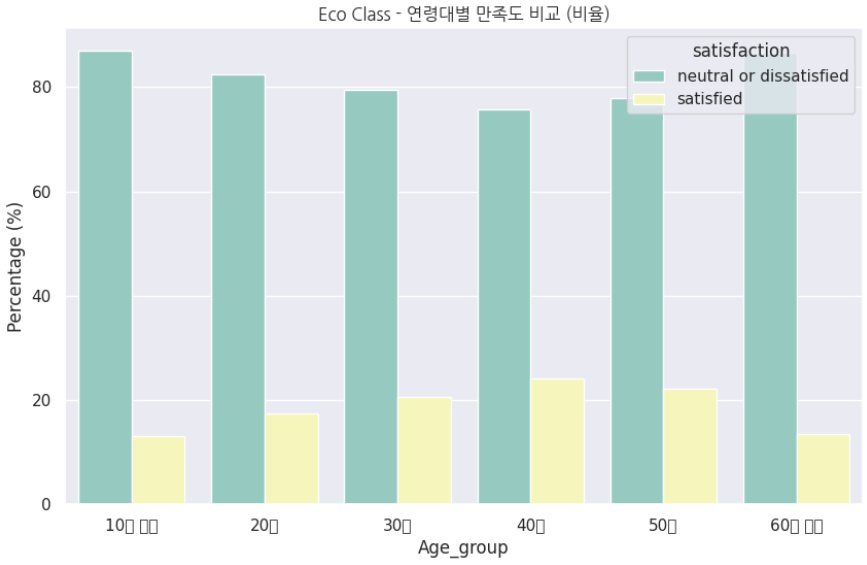


- 문제점은 **비즈니스 목적으로 비행기를 타려는 사람이 Eco석 끊었을 때 발생한다.** 전체적으로 봤을 때 Eco석을 끊은 사람들은 평가가 매우 야박했다.
- 위의 분석결과 중 40~50대의 클래스별 만족도 분포를 통하여 만족도가 높은 원인이 클래스의 차이 때문인지 확인한다.



- 40~50 대 승객은 비즈니스 이용률이 높으며 비즈니스에 대한 만족도가 높다.
- eco좌석인 경우에는 불만족인 경우가 높다.

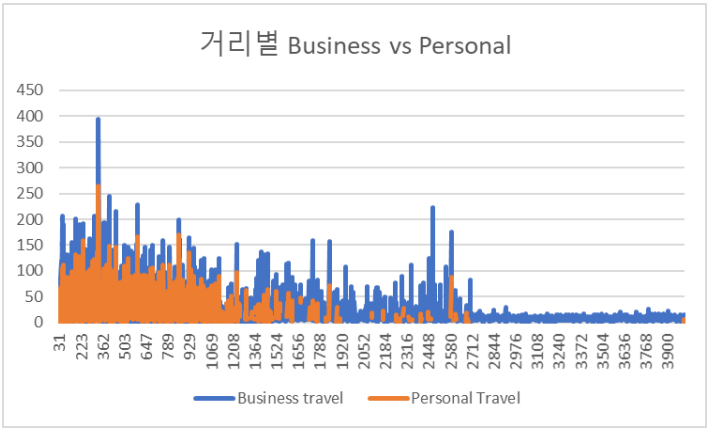
연령대별 eco class 만족도 차이가 있는지 확인한다.

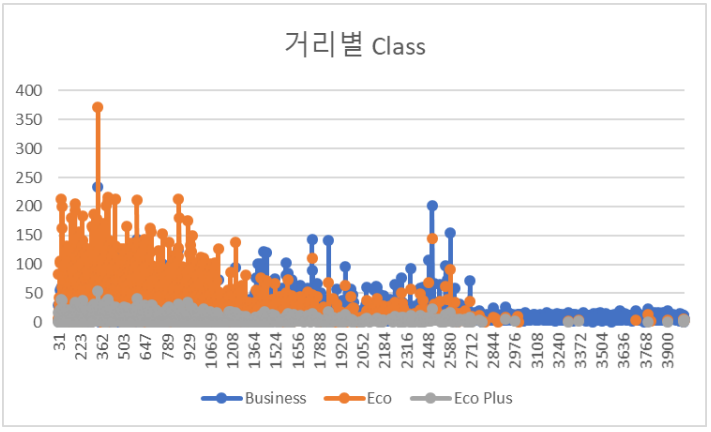


- 40-50대 연령의 eco class 만족도 분포가 타 연령과 크게 차이가 없는 것으로 보아 40~50대의 높은 만족도는 비즈니스클래스 탑승 비율이 높음으로 인한 결과로 볼 수 있다.
  - 다만, 40~50대라고 특별히 eco의 만족도가 높지는 않음
    - 연령보단 좌석의 차이가 만족도의 차이를 불러온다는 결론을 내림

3-5. 이동거리별 분석

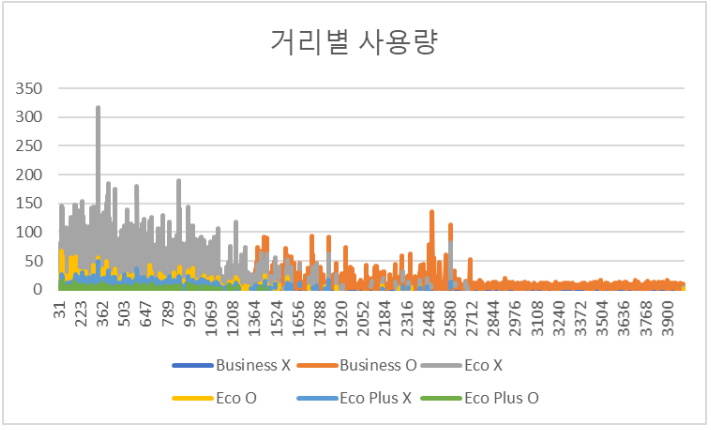
- 비행거리 별 차이





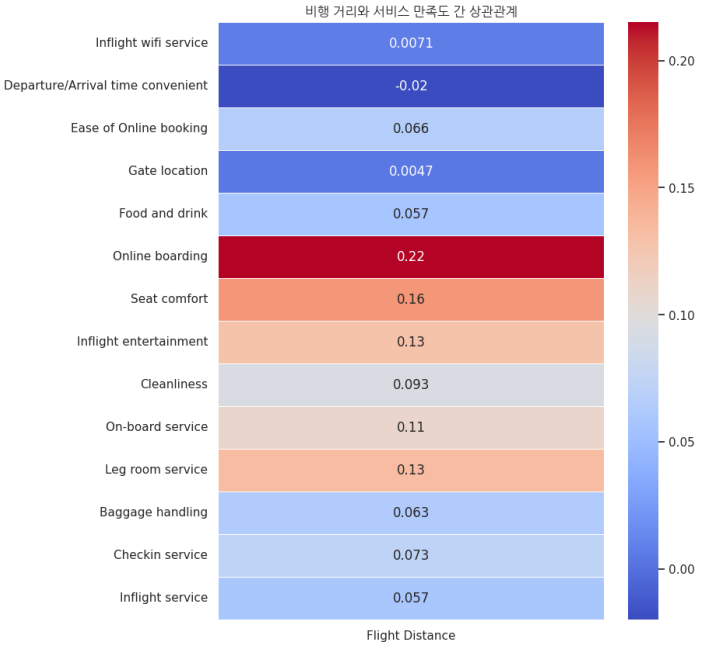
  - 거리별 사용량을 봤을 때, 두 그래프를 보고 알 수 있는 점은, **단거리를 이동하는 경우에는 비즈니스 목적으로 이용한다 해도 Eco 좌석을 사용하는 경우가 많다.** 자세한걸 분류를 해보자면

- 거리별 클래스 사용량



  - 장거리를 위해 비즈니스석을 이용하는 사람들은 거의 평가가 매우 좋다. 그래프상으로 봤을 때 그 반대의 경우는 보이지도 않는다. **다만 단거리로 갈수록 Eco 사용량이 늘어나는데 평가가 말이 안된다.**

- 이외 비행거리와 서비스 만족도 간 상관관계

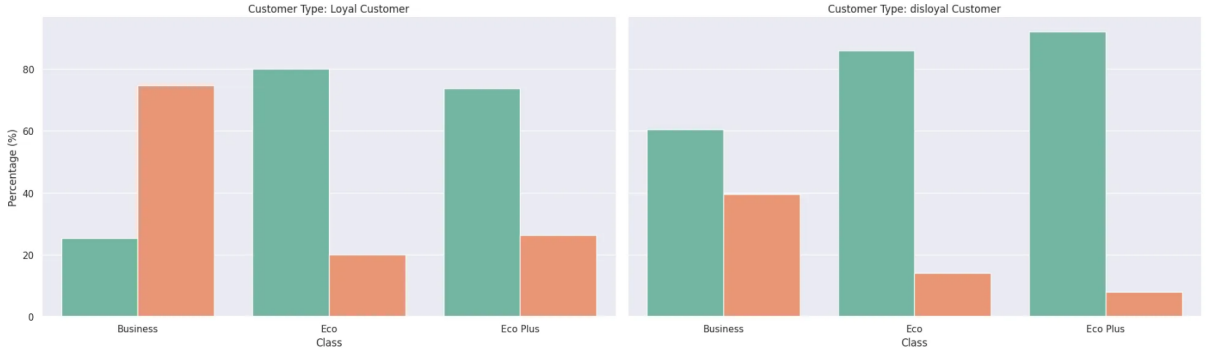


- 거리와 서비스요소 상관분석을 했는데 수치가 높게 나오진 않아서 따로 해석하진 않았다.

3-6. 로얄/비로얄 고객의 클래스별 서비스 만족도 분석

성별, 비행거리, 소비자 유형정도의컬럼만 남았는데, 그 중 소비자유형에 따른 만족도 차이가 있을 것이라고 판단되어 추가 분석을 해봤다.

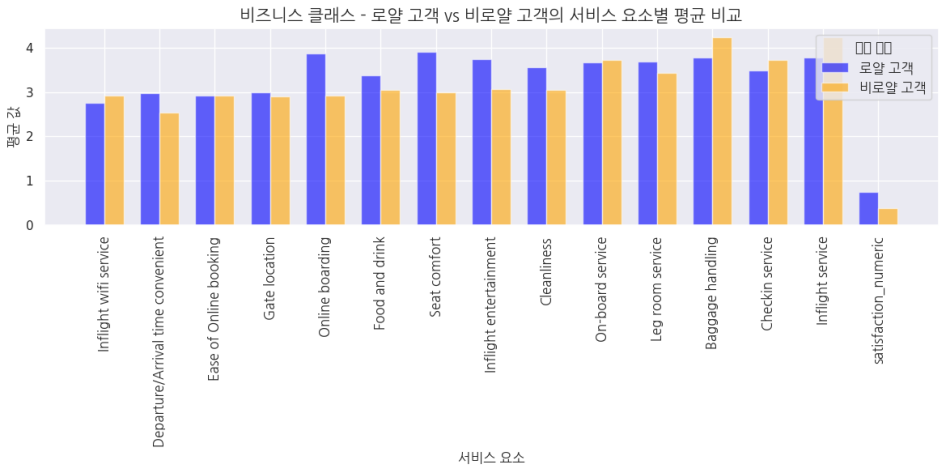
- customer 타입에 따른 만족도



- 소비자 유형이 로얄인경우(왼쪽) 비즈니스 class에서 만족한 경우가 높고 **소비자유형이 비로얄인경우에는 비즈니스class임에도 불구하고 불만족인 경우가 높다.**
  - 참고로 주황색이 만족, 초록색이 불만족이다.
  - 고객유형에 따른 비즈니스 클래스 만족도가 매우 다름을 알 수 있음

Business Class 만족도 평균	Loyal Customer	Disloyal Customer
Inflight wifi service	2.748762	2.927268
Departure/Arrival time convenient	2.971724	2.527104

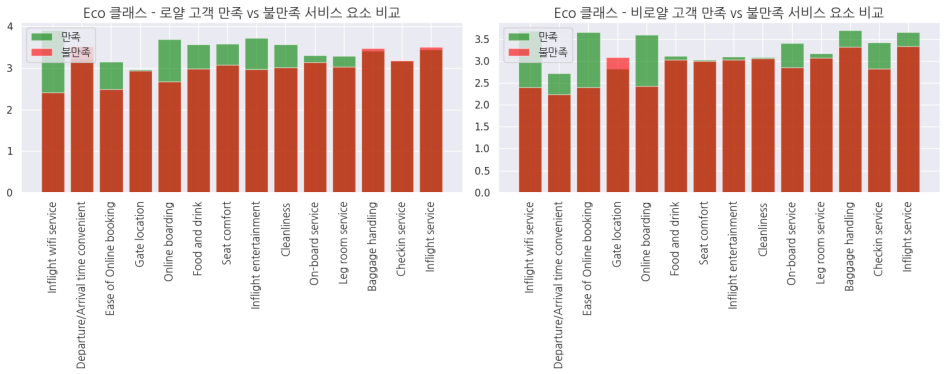
Business Class 만족도 평균	Loyal Customer	Disloyal Customer
Ease of Online booking	2.914674	2.910651
Gate location	2.998270	2.895533
Online Boarding	**3.856652**	2.910515
Food and drink	**3.372473**	3.037592
Seat comfort	**3.895381**	2.987333
Inflight entertainment	**3.735157**	3.062108
Cleanliness	**3.552487**	3.046173
On-board Service	3.673532	3.714519
Leg room service	3.681306	3.434078
Baggage handling	3.776990	**4.221057**
Checkin service	3.484938	**3.714655**
Inflight service	3.778294	**4.224598**
satisfaction numeric	0.746273	0.395533



- 로얄고객이 online boarding, food and drink, seat comfort, 기내 오락시설, cleanlines 요소의 평균 만족도가 더 높음 (기내서비스 요소의 만족도가 더 높음)
    - 같은 비즈니스여도 고객유형에 따라 만족도 차이가 있다.

class별, 고객별 만족 vs불만족 서비스 고객의 요소 비교

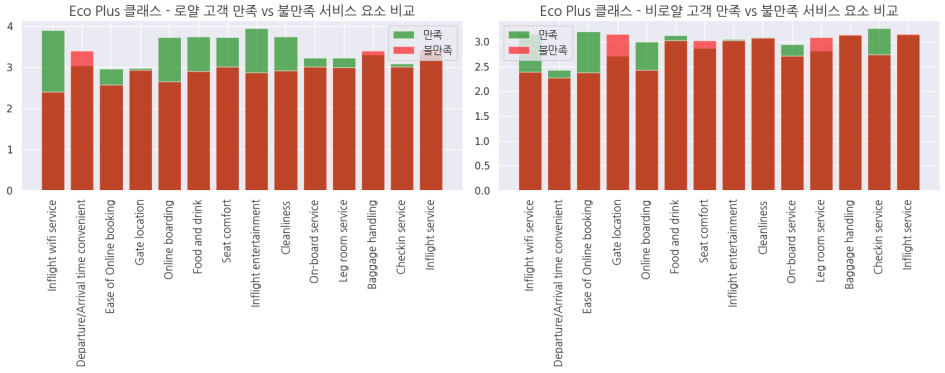
(위의 흐름을 이어가면 eco는 여행유형별로 한번 더 나누는게 좋을 것 같음)



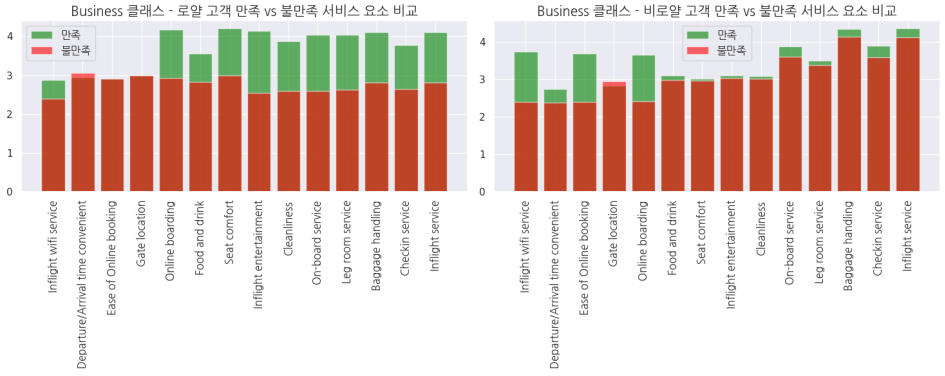
- 로얄고객은 와이파이서비스에 대한 만족도 차이영향이 큼

- 더 정확히는 와이파이 ,기내서비스(Food and drink, Seat comfort, Inflight entertainment, Cleaniness), 온라인 예약의 편리성, 온라인 탑승(온라인관련) 요소의 점수가 평균 3이상일때 종합서비스를 만족하는 경향이 있다.
- **즉, 인터넷관련서비스 , 기내서비스 두가지 요소가 크게 작용함**

- 비로얄고객은 로얄고객에 비해 전체적인 평점이 낮으며 **기내서비스요소**가 만족도를 결정하는데 차이가 없다.
  - **와이파이, 온라인예약의 용이성, 온라인탑승 이 세가지 요소가 크게 작용**하며,
    평점이 2.5~3 이상이면 만족함



- 전반적인 그래프모양이 eco class와 거의 일치하며 비로얄의 경우 eco보다 전체 평점이 낮다.



- 비즈니스의 로얄고객의 경우 온라인 예약의 편리성은 차이가 없으며 거의 모든 요소의 평점이 높아야지만 만족하는 경향이 있음.
  반대로 비로얄은 다른 class와 비슷한 그림이나, 전체 평점이 높은 편임

로얄 고객의 특징(eco vs 비즈니스)

<eco, eco plus>

- 와이파이 편리성이 만족과 불만족을 결정짓는데 크게 작용한다.
- 와이파이 ,기내서비스(Food and drink, Seat comfort, Inflight entertainment, Cleaniness), 온라인 예약의 편리성, 온라인 탑승(온라인관련) 요소의 점수가 평균 3이상일때 종합서비스를 만족하는 경향이 있다.

<비즈니스>

- 로얄 고객은 대부분의 서비스 요소에서 **평균 3점대 중후반**의 점수를 기록하면 만족한다고 답하는 경향이 있음 (와이파이, 출발시간편리함, 온라인예약의 편리성, 게이트위치 제외)
- **3점대 이하**인 경우 불만족을 느끼는 경향이 강하며, 모든 서비스 요소에서 고르게 높은 점수를 기록한 경우에만 만족을 표시함
  - 로얄 고객의 만족도를 예측할 때는 **모든 서비스 요소**가 일정 기준 이상(3점 중후반)이 되어야만 만족한다고 평가하는 경향이 있기 때문에, 각 요소의 **개별 평균 값이 높을수록** 만족할 가능성이 크다는 점을 반영한 모델을 설계한다.

종합적인 비로얄 고객의 특징:

- 비로얄 고객은 **와이파이, 온라인 예약의 편리함, 온라인 탑승** 세 가지 서비스 요소에 대한 점수가 높은 경우 만족하는 경향이 있음 → 가중치를 둔다
- 이 세 가지 요소에서 **3점대 후반**의 점수를 기록한 고객은 만족하는 경향이 강하지만, **2점대를 기록한 경우**에는 다른 요소에서 아무리 높은 점수를 기록해도 전반적으로 **불만족**을 표한다.
- **기내서비스(음식, 좌석 편리함, 청결, 엔터테인먼트)** 요소는 **모든 class가 공통적으로** 만족 고객과 불만족 고객 간에 큰 차이가 없으며, 이 부분은 중요도가 낮은 서비스 요소로 평가됨.

결론

고객 타입을 기준으로 데이터셋을 로얄 고객과 비로얄 고객으로 나눈 후, 각 집단에 대해 별도로 모델을 설계할 수 있음

- 로얄 고객의 경우, 다양한 서비스 요소의 고른 만족도를 반영하는 모델 설계.
- 비로얄 고객의 경우, 특정 서비스 요소에 대한 민감도를 반영하는 모델 설계

4.인사이트

지금까지 한걸 모두 종합해서 인사이트를 도출해보자

4-1. 비슷한 기내서비스를 하나로 묶을 수 있는가?

- 전체 상관분석을 해봤을때 Food and drink, Seat comfort, Inflight entertainment, Cleaniness 네가지 요소가 서로 높은 상관계수를 가지기 때문에 (0.5~0.7사이) 네가지 요소를 기내서비스로 묶었음
  - 전체 만족도는 온라인탑승이 가장 높은 상관관계, 그다음 기내서비스(4가지요소) 가 높은 관계를보임

4-2. 항공 클래스와 여행타입 간의 관계 분석

- class를 여행타입별로 나누는 것이 의미가 있는지 분석한 결과,
  비즈니스class는 대부분 비즈니스 목적 이기때문에 구분할 필요가 없지만, eco class는 여행타입별 만족도 차이가 있으므로 구분하는 것이 좋다고 할 수 있다.

→ eco , eco plus 를 하나로 묶고, business 와 별개로 나누어 0,1 형태로 분류

```
# Create the 'Class Type_Business' column
df['Class Type_Business'] = df['Class (Business)'].apply(lambda x: 1 if x == 'Business' else

# Drop the original 'Class (Business)' column if desired
df.drop(columns=['Class (Business)'], inplace=True)
```

4-3. 연령대와 클래스에 따른 만족도 차이

- 전체적으로 Personal 목적의 사람들은 연령대와 관계없이 일정한 수준의 사람 수를 보였는데, 그래프상으로 보면 일직선으로 되어있는, 말 그대로 부자연스러운 모습을 보인다.
  - 이로 인한 요인으로 Personal 목적 승객들은 나이와 상관없이 항공편을 이용하는 경우가 많다던가, 개인 여행객은 다양한 연령층에 분포할 수 있다던가, Personal 목적의 데이터가 상대적으로 적거나 특정 연령대에 데이터가 제대로 수집되지 않았을 수 있습니다.
- 반면 Business 목적의 사람들은 몇 가지 특징이 보이는데, 크게 20대와 40~50대로 밀집되어있는 것을 볼 수 있음.
  - 20대 같은 경우에는 Business 목적이라도 상대적으로 높은 수치의 Eco, Eco Plus석을 예매하는 것을 볼 수 있는데, 이로 인한 요인으로 사회초년기이기에 경제적 여유가 없어 더 싼 Eco석을 골랐을걸로 추정이 됨.
  - 그리고 Eco석 젊은 층들의 만족도 조사 결과가 매우 안 좋은데, 비즈니스 목적과 맞지 않는 환경이라 안 좋을 거라 예상이 됨.

- 40-50대 같은 경우, 전체적으로 비즈니스석에 대한 만족도가 높고, 승객수 또한 매우 많음.
  - 이러한 비율때문에 전체적인 만족도가 높게 나오는 경향이 있음.
  - 연령대별 eco좌석의 만족도 차이는 크게 없으므로, 나이에 따른 만족도 차이 또한 크게 나타나지 않는 것으로 결론을 내림
  - 이런 결과로 이 항공사는 경제적 여유를 가지고 있으며, 어느정도 지위가 있는 중년층들을 위해 비즈니스에 특화된 항공사라고 생각을 할 수 있음.
- 결론적으로 연령대별로 확인을 했을 때, 2개의 밀집구역을 확인할 수 있었으며, 만약 계산을 포함한 데이터의 단순화를 해야한다고 가정했을 때, 데이터상 골짜기를 띄고 있는 32살을 기점으로 32살 이하 & 32살 초과로 나누어 0과 1로 구분을 지을 수 있음
- 다음은 25~35살 사이의 value 값임.

```
age_counts = train[(train['Age'] >= 25) & (train['Age'] <= 35)]['Age'].value_counts().s
ort_index()
age_counts_df = pd.DataFrame(age_counts).transpose()
```

Age	25	26	27	28	29	30	31	32	33	34	35
count	2798	2289	2186	1707	1932	2030	1607	1570	1851	1753	1923

4-4. 클래스별 만족도와 서비스 상관관계 비교

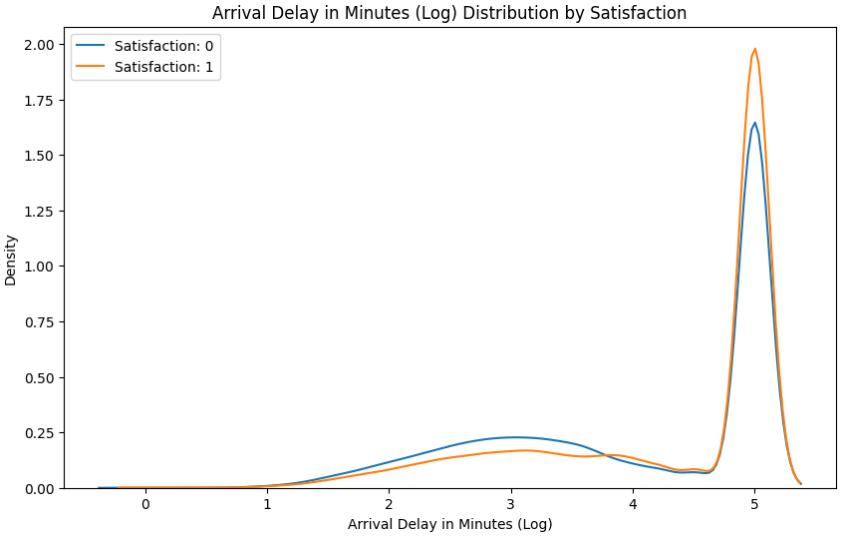
- 비즈니스 class이면 만족도가 높다 ⇒ class별로 만족도 차이가 있다. 따라서 class별 만족도 간의 상관분석을 하였음 → eco와 eco plus는 비슷한 결과. 비즈니스는 다른 결과를 보임
  1. Eco에서는 와이파이 편리성, Ease of Online booking, Online boarding 세가지가 상관관계가 높으나, 비즈니스에서는 와이파이편리성,Ease of Online booking 두요소만 상관관계가 높고**Online boarding은 낮게 나옴(** Ease of Online booking와 Online boarding는 다르게 작용함)-
  2. 비즈니스에서만 유독 와이파이, 출발시간의 편리성, 온라인예약의 편리함, 게이트위치 네가지 요소의 상관관계가 0.6~0.7 높게 나옴 → 결론적으로 eco와 다른 상관관계를 보인다.

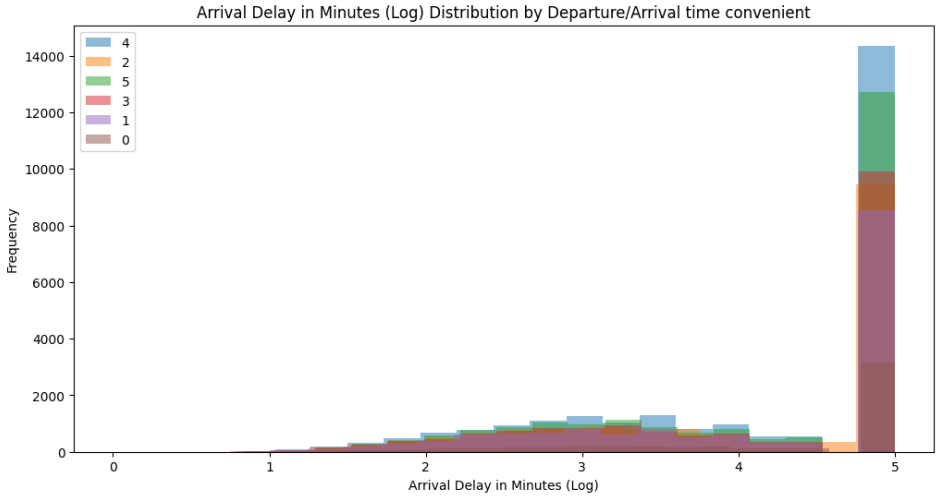
- 전체만족도와의 서비스 관계도 class별로 다른 결과를 보여준다.
  1. 비즈니스는 전체 만족도와 online boarding , 기내 오락시설과 0.51로 가장 큰 상관관계를 보이며 전반적으로 0.3이상의 상관관계를 보임.(와이파이서비스는 0.18로 매우 낮음)
  2. eco, eco plus → 전체만족도와 와이파이서비스가 약 0.5로 가장 큰 상관관계를 보임
  - 제일 큰 차이점은 '와이파이'만족도가 eco에서는 중요한데 비즈니스에서는 안중요함
  - **결론적으로는 eco와 비즈니스의 종합만족도를 결정짓는 요소가 다르며, 후에 만들 모델에서 어떤 컬럼을 적용할 지 충분히 고려해야 될 사항임**

4-5. 로얄/비로얄 고객의 클래스별 서비스 만족도 분석

- 로얄고객과 비로얄 고객의 비즈니스 class만족도 분포차이가 크다는 것을 발견함
  - 따라서 class와 고객타입 별 만족/불만족 승객의 서비스 평점 비교 진행
    1. 비로얄고객의 만족도 분포는 class에 따른 차이가 없음
    2. 로얄고객은 eco 와 비즈니스별로 다른 양상을 보이며, 높은 상관관계를 가지는 컬럼이 서로 다름.
- 결론적으로, 로얄 여부는 모델을 설계함에 있어 고려될 수 있는 사항이지만, 분류 자체가 기준점이 대비되지않으므로 , 2개의 데이터셋으로 나누어 모델을 2개로 만드는식으로 진행한다.

4-6. 지연시간에 따른 만족도 분석





Departure/Arrival time convenient 와 Arrival Delay in Minutes (Log) 높은 결합성을 보이고 있다.

Arrival Delay in Minutes (Log)의 경우 지연시간이 길어질수록 0에 수렴하도록 만들었을 때 의 결과이다

즉 1에 해당하는 지연시간이 높은경우 에도 1~5의 만족도 분포가 동일한 수준을 보이고 있으므로 ,

이는 만족도에 영향을 끼쳤다고 보기 힘들다고 할 수 있다 .

4-7. 소결(모델 최종 설계 구성)

데이터셋 정제

1. Food and drink, Seat comfort, Inflight entertainment, Cleaniness 를 하나로 묶음
    - In-flight Service Quality으로 통합 개명

```
import pandas as pd
import numpy as np

# 데이터셋 로드
file_path = 'encoded_airplane.csv'
data = pd.read_csv(file_path)

# 해당 컬럼을 In-flight Service Quality으로 통합
```

```
data['In-flight Service Quality'] = data[['Food and drink', 'Seat comfort', 'Inflight e
ntertainment', 'Cleanliness']].mean(axis=1)

# 기존의 컬럼 삭제
data.drop(columns=['Food and drink', 'Seat comfort', 'Inflight entertainment', 'Cleanli
ness'], inplace=True)

# 변경된 데이터프레임 확인
data.head()
```

2. 원핫인코딩에 의해 의미가 희미한 컬럼 이름 수정
   - Customer Type → disloyal Customer

```
#Loyal Customer 열 이름을 disloyal Customer로 변경

# 열이름 변경
data.rename(columns={'Customer Type': 'disloyal Customer'}, inplace=True)

# 변경된 데이터프레임 확인
data.head()
```

3. 4-3 결과에 따라로 연속형 데이터를 명목형 데이터로 나눔
   - age(연속형) → age_over_32

```
# Age 연속형 데이터를 age_over_32 명목형 데이터로 나누기


# age를 기준으로 age_over_32 열 추가
data['age_over_32'] = np.where(data['Age'] > 32, 1, 0)

# 변경된 데이터프레임 확인
data.head()
```

5. Gate location 은 종합 만족도와의 관계에서 **0.000449 이라는 낮은 수치를 보였으므로 데이터에서 제외한다.**
6. 4-6에 결과 근거로 Departure/Arrival time convenient 는 제거

```
# 'Gate location'과 'Departure/Arrival time convenient' 컬럼은 분석에서 제외하기로 결정했으므로
제거합니다.
data.drop(columns=['Gate location', 'Departure/Arrival time convenient'], inplace=True)

# 변경 사항 확인을 위해 수정된 데이터프레임의 첫 몇 행을 출력
print(data.head())
```

6. 로열 데이터

- 결론적으로, 로얄 여부는 모델을 설계함에 있어 고려될 수 있는 사항이지만, 분류 자체가 기준점이 대비되지않으므로 , 2개의 데이터셋으로 나누어 모델을 2개로 만드는식으로 진행한다.

```
# 로열 데이터에 따라 두 개의 데이터셋으로 나누기
loyal_data = data[data['disloyal Customer'] == 0]
disloyal_data = data[data['disloyal Customer'] == 1]
```

7. 그 외 상관관계 및

```
columns_to_exclude = [
    'Age',
    'Flight Distance',
```

```
    'Departure Delay in Minutes',
    'Arrival Delay in Minutes',
    'Departure Delay in Minutes (Log)',
    'Arrival Delay in Minutes (Log)',
    'Flight Distance (Scaled)',
    'Gender'
]

# 지정된 컬럼 제거
data.drop(columns=columns_to_exclude, inplace=True)

data['Class Type'] = data.apply(lambda row: 1 if row['Class Business'] == 1 else 0, axis=1
data.drop(columns=['Class Business', 'Class Eco', 'Class Eco Plus'], inplace=True)
```

- 이와 같은 방법으로 테스트 데이터 셋도 전처리를 진행하였다.

```
test = pd.read_csv('test.csv')

# 필요없는 것부터 삭제
test.drop(columns=['Unnamed: 0', 'Gender', 'Flight Distance', 'Gate location', 'Departure/
Arrival time convenient', 'Departure Delay in Minutes','Arrival Delay in Minutes'], inplac
e=True)

# 해당 컬럼을 In-flight Service Quality으로 통합
test.insert(17, 'In-flight Service Quality', test[['Food and drink', 'Seat comfort', 'Infl
ight entertainment', 'Cleanliness']].mean(axis=1))
# 기존의 컬럼 삭제
test.drop(columns=['Food and drink', 'Seat comfort', 'Inflight entertainment', 'Cleanlines
s'], inplace=True)

# 열이름 변경
test.rename(columns={'Customer Type': 'disloyal Customer'}, inplace=True)

# 비즈니스 : 1, 그외 : 0
class_mapping = {'Business': 1, 'Eco Plus': 0, 'Eco': 0}
test['Class'] = test['Class'].map(class_mapping)
test.rename(columns={'Class': 'Class Type'}, inplace=True)

# 나이 32 기준 위아래로 분류
test.insert(2, 'age_over_32', np.where(test['Age'] > 32, 1, 0))
test.drop(columns=['Age'], inplace=True)

# 남은 데이터 수치화
test['disloyal Customer'] = np.where(test['disloyal Customer'] == 'disloyal Customer', 1,
0)
test['Type of Travel'] = np.where(test['Type of Travel'] == 'Business travel', 1, 0)
test['satisfaction'] = np.where(test['satisfaction'] == 'satisfied', 1, 0)
```

5. 모델 적용

로지스틱 회귀

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

from google.colab import files
uploaded = files.upload()  # 파일 업로드

# 업로드된 파일을 데이터프레임으로 읽기
df_train = pd.read_csv('modified_airplane.csv')
df_test = pd.read_csv('modified_test.csv')

# 공백이 있는 열 이름을 언더스코어로 변경
df_train.columns = df_train.columns.str.replace(' ', '_')
df_test.columns = df_test.columns.str.replace(' ', '_')

# 로얄 고객과 비로얄 고객으로 나누기
train_loyal = df_train[df_train['disloyal_Customer'] == 0]
train_disloyal = df_train[df_train['disloyal_Customer'] == 1]

test_loyal = df_test[df_test['disloyal_Customer'] == 0]
test_disloyal = df_test[df_test['disloyal_Customer'] == 1]

# 독립 변수(X)와 종속 변수(y) 설정
# 로얄 고객 데이터셋 (train/test)
X_loyal = train_loyal.drop(columns=['satisfaction', 'disloyal_Customer'])
y_loyal = train_loyal['satisfaction']

# 비로얄 고객 데이터셋 (train/test)
X_disloyal = train_disloyal.drop(columns=['satisfaction', 'disloyal_Customer'])
y_disloyal = train_disloyal['satisfaction']

# 로얄 고객 검증 세트 나누기 (0.2 비율)
X_train_loyal, X_val_loyal, y_train_loyal, y_val_loyal = train_test_split(X_loyal, y_loyal, t

# 비로얄 고객 검증 세트 나누기 (0.2 비율)
X_train_disloyal, X_val_disloyal, y_train_disloyal, y_val_disloyal = train_test_split(X_dislo

# 로지스틱 회귀 모델 학습 (로얄 고객)
model_loyal = LogisticRegression(max_iter=1000)
model_loyal.fit(X_train_loyal, y_train_loyal)

# 로지스틱 회귀 모델 학습 (비로얄 고객)
model_disloyal = LogisticRegression(max_iter=1000)
model_disloyal.fit(X_train_disloyal, y_train_disloyal)
```

```
# 검증 세트에 대한 평가 (로얄 고객)
y_pred_val_loyal = model_loyal.predict(X_val_loyal)
print("로얄 고객 모델 - 검증 세트 평가")
```

```
print("Accuracy:", accuracy_score(y_val_loyal, y_pred_val_loyal))
print("Confusion Matrix:\n", confusion_matrix(y_val_loyal, y_pred_val_loyal))
print("Classification Report:\n", classification_report(y_val_loyal, y_pred_val_loyal))

# 검증 세트에 대한 평가 (비로얄 고객)
y_pred_val_disloyal = model_disloyal.predict(X_val_disloyal)
print("\n비로얄 고객 모델 - 검증 세트 평가")
print("Accuracy:", accuracy_score(y_val_disloyal, y_pred_val_disloyal))
print("Confusion Matrix:\n", confusion_matrix(y_val_disloyal, y_pred_val_disloyal))
print("Classification Report:\n", classification_report(y_val_disloyal, y_pred_val_disloyal))

# 테스트 세트에 대한 평가 (로얄 고객)
y_pred_test_loyal = model_loyal.predict(test_loyal.drop(columns=['satisfaction', 'disloyal_Cu
print("\n로얄 고객 모델 - 테스트 세트 평가")
print("Accuracy:", accuracy_score(test_loyal['satisfaction'], y_pred_test_loyal))
print("Confusion Matrix:\n", confusion_matrix(test_loyal['satisfaction'], y_pred_test_loyal))
print("Classification Report:\n", classification_report(test_loyal['satisfaction'], y_pred_te

# 테스트 세트에 대한 평가 (비로얄 고객)
y_pred_test_disloyal = model_disloyal.predict(test_disloyal.drop(columns=['satisfaction', 'di
print("\n비로얄 고객 모델 - 테스트 세트 평가")
print("Accuracy:", accuracy_score(test_disloyal['satisfaction'], y_pred_test_disloyal))
print("Confusion Matrix:\n", confusion_matrix(test_disloyal['satisfaction'], y_pred_test_disl
print("Classification Report:\n", classification_report(test_disloyal['satisfaction'], y_pred
```

```
로얄 고객 모델 - 검증 세트 평가
Accuracy: 0.8755093604204807
Confusion Matrix:
 [[7787 1102]
 [1006 7038]]
Classification Report:
               precision    recall  f1-score   support

           0       0.89      0.88      0.88      8889
           1       0.86      0.87      0.87      8044

    accuracy                           0.88     16933
   macro avg       0.88      0.88      0.88     16933
weighted avg       0.88      0.88      0.88     16933


비로얄 고객 모델 - 검증 세트 평가
Accuracy: 0.860575653551624
Confusion Matrix:
 [[2744  145]
 [ 383  515]]
Classification Report:
               precision    recall  f1-score   support

           0       0.88      0.95      0.91      2889
           1       0.78      0.57      0.66       898

    accuracy                           0.86      3787
   macro avg       0.83      0.76      0.79      3787
weighted avg       0.85      0.86      0.85      3787
```

```
로얄 고객 모델 - 테스트 세트 평가
Accuracy: 0.3916985408698116
Confusion Matrix:
 [[6360 4622]
 [8260 1935]]
Classification Report:
              precision    recall  f1-score   support

           0       0.44      0.58      0.50     10982
           1       0.30      0.19      0.23     10195

    accuracy                           0.39     21177
   macro avg       0.37      0.38      0.36     21177
weighted avg       0.37      0.39      0.37     21177


비로얄 고객 모델 - 테스트 세트 평가
Accuracy: 0.8499687434882267
Confusion Matrix:
 [[3417  174]
 [ 546  662]]
Classification Report:
              precision    recall  f1-score   support

           0       0.86      0.95      0.90      3591
           1       0.79      0.55      0.65      1208

    accuracy                           0.85      4799
   macro avg       0.83      0.75      0.78      4799
weighted avg       0.84      0.85      0.84      4799
```

- **로얄 고객**:

검증 세트에서 높은 정확도(0.8755)와 F1 Score(0.87)를 기록했지만, 테스트 세트에서 정확도가 0.3917로 크게 감소함. 이는 모델이 검증 세트에 **과적합(overfitting)** 되었음을 시사함. F1 Score 또한 0.23으로 낮아, 모델의 신뢰성이 떨어짐.

- **비로얄 고객**:

검증 세트와 테스트 세트에서 비교적 일관된 성능을 보임. 검증 세트의 정확도는 0.8606, F1 Score는 0.79였으며, 테스트 세트에서도 0.8499와 0.65로 좋은 성능을 유지하고 있음.

로얄 고객 모델의 성능이 검증 세트와 테스트 세트 간에 큰 차이를 보이는 반면, 비로얄 고객 모델은 안정적인 성능을 보임. 이는 로얄 고객 데이터에서 과적합이 발생했음을 나타내며, 모델의 일반화 능력을 개선하기 위한 추가적인 조치가 필요함.

하이퍼파라미터를 적용하여 모델의 개선이 있는지 확인한다.

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

#  검증 세트로 분할
X_train_loy, X_val_loy, y_train_loy, y_val_loy = train_test_split(X_train_loyal, y_train_loya
X_train_dis, X_val_dis, y_train_dis, y_val_dis = train_test_split(X_train_disloyal, y_train_d

# 하이퍼파라미터 그리드 서치 설정
param_grid = {'C': [0.1, 1, 10, 100], 'penalty': ['l1', 'l2']}
```

```
#  로얄 고객 모델 하이퍼파라미터 조정
grid_loyal = GridSearchCV(LogisticRegression(max_iter=1000), param_grid, cv=5, scoring='accur
grid_loyal.fit(X_train_loy, y_train_loy)

#  최적의 하이퍼파라미터 적용
best_params_loyal = grid_loyal.best_params_
print("로얄 고객 최적 하이퍼파라미터: ", best_params_loyal)

# 로얄 고객 모델 학습
model_loyal = LogisticRegression(max_iter=1000, **best_params_loyal)
model_loyal.fit(X_train_loy, y_train_loy)

#  로얄 고객 모델 평가 (검증 세트)
y_pred_val_loyal = model_loyal.predict(X_val_loy)
print("로얄 고객 모델 - 검증 세트 평가")
print("Accuracy:", accuracy_score(y_val_loy, y_pred_val_loyal))
print("Confusion Matrix:\n", confusion_matrix(y_val_loy, y_pred_val_loyal))
print("Classification Report:\n", classification_report(y_val_loy, y_pred_val_loyal))

# 비로얄 고객 모델 하이퍼파라미터 조정
grid_disloyal = GridSearchCV(LogisticRegression(max_iter=1000), param_grid, cv=5, scoring='ac
grid_disloyal.fit(X_train_dis, y_train_dis)

# 최적의 하이퍼파라미터 적용
best_params_disloyal = grid_disloyal.best_params_
print("비로얄 고객 최적 하이퍼파라미터: ", best_params_disloyal)

# 비로얄 고객 모델 학습
model_disloyal = LogisticRegression(max_iter=1000, **best_params_disloyal)
model_disloyal.fit(X_train_dis, y_train_dis)

#  비로얄 고객 모델 평가 (검증 세트)
y_pred_val_disloyal = model_disloyal.predict(X_val_dis)
print("\n비로얄 고객 모델 - 검증 세트 평가")
print("Accuracy:", accuracy_score(y_val_dis, y_pred_val_disloyal))
print("Confusion Matrix:\n", confusion_matrix(y_val_dis, y_pred_val_disloyal))
print("Classification Report:\n", classification_report(y_val_dis, y_pred_val_disloyal))

# 최종 모델 평가를 위한 테스트 세트 평가 (로얄 고객)
y_test_loyal = test_loyal['satisfaction']
X_test_loyal = test_loyal.drop(columns=['satisfaction', 'disloyal_Customer'])
y_pred_test_loyal = model_loyal.predict(X_test_loyal)

print("\n로얄 고객 모델 - 테스트 세트 평가")
print("Accuracy:", accuracy_score(y_test_loyal, y_pred_test_loyal))
print("Confusion Matrix:\n", confusion_matrix(y_test_loyal, y_pred_test_loyal))
print("Classification Report:\n", classification_report(y_test_loyal, y_pred_test_loyal))

# 최종 모델 평가를 위한 테스트 세트 평가 (비로얄 고객)
y_test_disloyal = test_disloyal['satisfaction']
X_test_disloyal = test_disloyal.drop(columns=['satisfaction', 'disloyal_Customer'])
y_pred_test_disloyal = model_disloyal.predict(X_test_disloyal)

print("\n비로얄 고객 모델 - 테스트 세트 평가")
print("Accuracy:", accuracy_score(y_test_disloyal, y_pred_test_disloyal))
print("Confusion Matrix:\n", confusion_matrix(y_test_disloyal, y_pred_test_disloyal))
print("Classification Report:\n", classification_report(y_test_disloyal, y_pred_test_disloyal
```

```
로얄 고객 모델 - 검증 세트 평가
Accuracy: 0.8752399232245681
Confusion Matrix:
 [[6217  946]
 [ 744 5639]]
Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.87      0.88      7163
           1       0.86      0.88      0.87      6383

    accuracy                           0.88     13546
   macro avg       0.87      0.88      0.88     13546
weighted avg       0.88      0.88      0.88     13546

비로얄 고객 모델 - 검증 세트 평가
Accuracy: 0.8613403763618356
Confusion Matrix:
 [[2214   92]
 [ 328  395]]
Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.96      0.91      2306
           1       0.81      0.55      0.65       723

    accuracy                           0.86      3029
   macro avg       0.84      0.75      0.78      3029
weighted avg       0.86      0.86      0.85      3029


로얄 고객 모델 - 테스트 세트 평가
Accuracy: 0.41304245171648485
Confusion Matrix:
 [[6590 4392]
 [8038 2157]]
Classification Report:
              precision    recall  f1-score   support

           0       0.45      0.60      0.51     10982
           1       0.33      0.21      0.26     10195

    accuracy                           0.41     21177
   macro avg       0.39      0.41      0.39     21177
weighted avg       0.39      0.41      0.39     21177


비로얄 고객 모델 - 테스트 세트 평가
Accuracy: 0.8497603667430714
Confusion Matrix:
 [[3428  163]
 [ 558  650]]
Classification Report:
              precision    recall  f1-score   support

           0       0.86      0.95      0.90      3591
           1       0.80      0.54      0.64      1208
```

```
    accuracy                           0.85      4799
   macro avg       0.83      0.75      0.77      4799
weighted avg       0.84      0.85      0.84      4799
```

<하이퍼파라미터 적용 후>

로얄고객: test 에서 정확도가 0.39애서 0.41로 소폭 증가했으나 여전히 낮은 수치를 보인다.

- class별 가중치 적용

인사이트에 따르면 로얄고객은 class type별로 다른 만족도 결과를 보여주기 때문에 이를 활용하여 로얄고객 모델 성능을 높이고자 한다.

실행 계획

1. 클래스별 가중치를 설정하여 모델을 학습시킨다.
2. 하이퍼파라미터 튜닝을 통해 최적의 모델을 찾는다.
3. 검증 세트를 통해 모델 성능을 평가하고, 테스트 세트를 통해 최종 성능을 확인다.

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.preprocessing import StandardScaler

# 로얄 고객과 비로얄 고객으로 나누기
train_loyal = df_train[df_train['disloyal_Customer'] == 0]
train_disloyal = df_train[df_train['disloyal_Customer'] == 1]

test_loyal = df_test[df_test['disloyal_Customer'] == 0]
test_disloyal = df_test[df_test['disloyal_Customer'] == 1]

#  특성별 가중치 설정
eco_weights = {
    'Inflight_wifi_service': 1.5,
    'In-flight_Service_Quality': 1.2,
    'Ease_of_Online_booking': 1.3,
    'Online_boarding': 1.4
}

business_weights = {
    'In-flight_Service_Quality': 1.2,
    'Online_boarding': 1.2,
    'Ease_of_Online_booking': 0
}
```

1. 비즈니스는 전체 만족도와 online boarding , 기내 오락시설과 0.51로 가장 큰 상관관계를 보이며 전반적으로 0.3이상의 상관관계를 보임.(와이파이서비스는 0.18로 매우 낮음)
2. eco, eco plus → 전체만족도와 와이파이서비스가 약 0.5로 가장 큰 상관관계를 보임

- 제일 큰 차이점은 '와이파이'만족도가 eco에서는 중요한데 비즈니스에서는 안중요함
- **결론적으로는 eco와 비즈니스의 종합만족도를 결정짓는 요소가 다르며, 후에 만들 모델에서 어떤 컬럼을 적용할 지 충분히 고려해야 될 사항임**

<eco>

- 와이파이 편리성이 만족과 불만족을 결정짓는데 크게 작용한다.

- 와이파이 ,기내서비스(In-flight_Service_Quality), 온라인 예약의 편리성, 온라인 탑승(온라인관련) 요소의 점수가 평균 3이상일때 종합서비스를 만족하는 경향이 있다.

<비즈니스>

- 로얄 고객은 대부분의 서비스 요소에서 **평균 3점대 중후반**의 점수를 기록하면 만족한다고 답하는 경향이 있음 (와이파이, 출발시간편리함, 온라인예약의 편리성, 게이트위치 제외)
- **3점대 이하**인 경우 불만족을 느끼는 경향이 강하며, 모든 서비스 요소에서 고르게 높은 점수를 기록한 경우에만 만족을 표시함
    - 로얄 고객의 만족도를 예측할 때는 **모든 서비스 요소**가 일정 기준 이상(3점 중후반)이 되어야만 만족한다고 평가하는 경향이 있기 때문에, 각 요소의 **개별 평균 값이 높을수록** 만족할 가능성이 크다는 점을 반영한 모델을 설계한다.

종합적인 비로얄 고객의 특징:

- 비로얄 고객은 **와이파이, 온라인 예약의 편리함, 온라인 탑승** 세 가지 서비스 요소에 대한 점수가 높은 경우 만족하는 경향이 있음 → 가중치를 둔다
- 이 세 가지 요소에서 **3점대 후반**의 점수를 기록한 고객은 만족하는 경향이 강하지만, **2점대를 기록한 경우**에는 다른 요소에서 아무리 높은 점수를 기록해도 전반적으로 **불만족**을 표한다.
- **기내서비스(음식, 좌석 편리함, 청결, 엔터테인먼트)** 요소는 **모든 class가 공통적으로** 만족 고객과 불만족 고객 간에 큰 차이가 없으며, 이 부분은 중요도가 낮은 서비스 요소로 평가됨. → 제외시킬 것

인사이트에 기반하여 가중치를 설정함

```
#. 가중치를 반영한 특성 값 계산
def apply_weights(row, class_type):
    if class_type == 0 :
        return sum(row[col] * eco_weights[col] for col in eco_weights)
    elif class_type == 1 :
        return sum(row[col] * business_weights[col] for col in business_weights)
    return 0

# 가중치를 적용하여 새로운 특성 추가
df_train['Weighted_Feature'] = df_train.apply(lambda row: apply_weights(row, row['Class_Type'
df_test['Weighted_Feature'] = df_test.apply(lambda row: apply_weights(row, row['Class_Type'])

# 독립 변수(X)와 종속 변수(y) 설정 (로얄 고객)
X_train_loyal = train_loyal.drop(columns=['satisfaction', 'disloyal_Customer'])
y_train_loyal = train_loyal['satisfaction']

X_test_loyal = test_loyal.drop(columns=['satisfaction', 'disloyal_Customer'])
y_test_loyal = test_loyal['satisfaction']

# 독립 변수(X)와 종속 변수(y) 설정 (비로얄 고객)
X_train_disloyal = train_disloyal.drop(columns=['satisfaction', 'disloyal_Customer', 'In-flig
y_train_disloyal = train_disloyal['satisfaction']

X_test_disloyal = test_disloyal.drop(columns=['satisfaction', 'disloyal_Customer', 'In-flight
y_test_disloyal = test_disloyal['satisfaction']

# 5. 표준화 (스케일링)
scaler = StandardScaler()
X_train_loyal_scaled = scaler.fit_transform(X_train_loyal)
X_test_loyal_scaled = scaler.transform(X_test_loyal)

X_train_disloyal_scaled = scaler.fit_transform(X_train_disloyal)
X_test_disloyal_scaled = scaler.transform(X_test_disloyal)
```

인사이트에 기반하여 disloyal에는 'In-flight_Service_Quality'을 x에서 제외함

```
# 로지스틱 회귀 모델 학습 (하이퍼파라미터 튜닝을 위한 GridSearchCV)
param_grid = {'C': [0.1, 1, 10, 100], 'penalty': ['l1', 'l2']}
grid_loyal = GridSearchCV(LogisticRegression(max_iter=1000), param_grid, cv=5, scoring='accur
grid_loyal.fit(X_train_loyal_scaled, y_train_loyal)

print("로얄 고객 Best Parameters: ", grid_loyal.best_params_)
print("로얄 고객 Best Score: ", grid_loyal.best_score_)

# 최적의 하이퍼파라미터를 사용하여 로얄 고객 모델 학습
best_params_loyal = grid_loyal.best_params_
model_loyal = LogisticRegression(max_iter=1000, C=best_params_loyal['C'], penalty=best_params
model_loyal.fit(X_train_loyal_scaled, y_train_loyal)

# 7. 로얄 고객 모델 평가
y_pred_loyal = model_loyal.predict(X_test_loyal_scaled)
print("로얄 고객 모델 평가")
print("Accuracy:", accuracy_score(y_test_loyal, y_pred_loyal))
print("Confusion Matrix:\n", confusion_matrix(y_test_loyal, y_pred_loyal))
print("Classification Report:\n", classification_report(y_test_loyal, y_pred_loyal))

# 6. 비로얄 고객 모델 학습 (하이퍼파라미터 튜닝을 위한 GridSearchCV)
grid_disloyal = GridSearchCV(LogisticRegression(max_iter=1000), param_grid, cv=5, scoring='ac
grid_disloyal.fit(X_train_disloyal_scaled, y_train_disloyal)

print("비로얄 고객 Best Parameters: ", grid_disloyal.best_params_)
print("비로얄 고객 Best Score: ", grid_disloyal.best_score_)

# 최적의 하이퍼파라미터를 사용하여 비로얄 고객 모델 학습
best_params_disloyal = grid_disloyal.best_params_
model_disloyal = LogisticRegression(max_iter=1000, C=best_params_disloyal['C'], penalty=best_
model_disloyal.fit(X_train_disloyal_scaled, y_train_disloyal)

# 7. 비로얄 고객 모델 평가
y_pred_disloyal = model_disloyal.predict(X_test_disloyal_scaled)
print("\n비로얄 고객 모델 평가")
print("Accuracy:", accuracy_score(y_test_disloyal, y_pred_disloyal))
print("Confusion Matrix:\n", confusion_matrix(y_test_disloyal, y_pred_disloyal))
print("Classification Report:\n", classification_report(y_test_disloyal, y_pred_disloyal))
```

```
로얄 고객 Best Parameters:  {'C': 10, 'penalty': 'l2'}
로얄 고객 Best Score:  0.8873520602152514
로얄 고객 모델 평가
Accuracy: 0.6114180478821363
Confusion Matrix:
 [[8207 2775]
 [5454 4741]]
Classification Report:
               precision    recall  f1-score   support

           0       0.60      0.75      0.67     10982
           1       0.63      0.47      0.54     10195

    accuracy                           0.61     21177
   macro avg       0.62      0.61      0.60     21177
weighted avg       0.62      0.61      0.60     21177
```

```
비로얄 고객 Best Parameters:  {'C': 1, 'penalty': 'l2'}
비로얄 고객 Best Score:  0.8626667592903742

비로얄 고객 모델 평가
Accuracy: 0.854553031881642
Confusion Matrix:
 [[3362  229]
 [ 469  739]]
Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.94      0.91      3591
           1       0.76      0.61      0.68      1208

    accuracy                           0.85      4799
   macro avg       0.82      0.77      0.79      4799
weighted avg       0.85      0.85      0.85      4799
```

Accuracy: 0.39
Precision (클래스 1): 0.30
Recall (클래스 1): 0.19
f1-score (클래스 1): 0.23

가중치 적용 후:

Accuracy: 0.61
Precision (클래스 1): 0.63
Recall (클래스 1): 0.47

f1-score (클래스 1): 0.54

→ 정확도가 0.42에서 0.61로 상승.

- **로얄 고객 모델:**
  - 하이퍼파라미터 적용 후 정확도가 0.3917에서 0.6114로 증가하였으나 여전히 낮은 수준이다.
  - Precision, Recall, F1-score 모두 개선되었지만, 여전히 불만족 클래스에 대한 예측 성능이 부족하다.
  - 기본 모델에 비해 F1-score가 0.27에서 0.54로 상승하며 다소 개선된 결과를 보인다.
- **비로얄 고객 모델:**
  - 하이퍼파라미터 적용 후 정확도가 0.8499에서 0.8546으로 소폭 증가하였다.
  - Precision과 Recall 모두 높은 수치를 유지하고 있어 비로얄 고객에 대한 모델 성능이 안정적임을 보여준다.

→ 로얄 고객 모델은 하이퍼파라미터 조정을 통해 성능이 개선되었지만, 여전히 정확도가 낮고 F1-score가 미흡하다. 이는 로얄 고객의 데이터가 불균형하거나, 특정 특성에 대한 반응이 더 복잡할 수 있음을 시사한다.

- **가중치 적용 배경**

  로얄 고객의 경우, 여러 서비스 요소에서 고르게 높은 만족도를 기대하는 경향이 있다. 특히 **와이파이 서비스**, **기내 서비스 품질**, **온라인 예약의 편리성**과 같은 특성들이 만족도를 결정짓는 주요 요소로 나타났다. 이러한 요소에 더 큰 가중치를 부여함으로써 모델이 더 중요한 특성에 집중할 수 있게 했다.
- **성능 향상의 원인**
  - **특성별 가중치 부여**: 로지스틱 회귀에서 **와이파이 서비스**와 **기내 서비스 품질** 같은 요소에 더 큰 가중치를 부여함으로써, 고객의 만족도에 큰 영향을 미치는 특성들이 모델에서 더 중요한 역할을 하게 됐다. 이를 통해 **정확도**가 0.61로, **f1-score**는 0.54로 향상되었다.

- **복잡한 상호작용 반영**: 로얄 고객은 다양한 서비스 요소에서 **고른 만족도**를 기대하는 경향이 있기 때문에, 가중치를 부여함으로써 이 상호작용을 반영하는 데 성공했다. 여러 요소가 결합되어 만족도를 결정하는 패턴을 더 잘 학습할 수 있었다.

**

로얄 고객과 비로얄 고객 모델 성능 차이의 원인

1. **로얄 고객의 다양한 서비스 기대치**

   로얄 고객은 여러 서비스 요소에서 고르게 높은 만족도를 기대하기 때문에, 모델이 이러한 복잡한 패턴을 예측하기 어려운 경향이 있다.

2. **비로얄 고객의 단순한 기대치**

   비로얄 고객은 몇 가지 핵심 요소(와이파이, 온라인 예약, 탑승)에 집중하기 때문에, 모델이 더 단순한 패턴을 학습하여 예측이 용이하다.

3. **로얄 고객의 서비스 요소 간 상호작용**

   로얄 고객은 다양한 서비스 요소들이 상호작용하여 만족도를 결정하기 때문에, 단순한 모델로는 이러한 상호작용을 충분히 반영하지 못한다.

4. **클래스 간 차이**

   로얄 고객은 Eco와 Business 클래스에서 다른 서비스 기대치와 만족도를 보이므로, 이 차이를 단순한 모델이 효과적으로 반영하지 못한다. 반면 비로얄 고객은 클래스 간 차이가 뚜렷하지 않다.

   - **로얄 고객**은 **Eco**와 **Business**에서 서비스 기대치와 만족도가 다름. **비즈니스 고객**은 **전반적으로 고른 서비스**에 만족하고, **에코 고객**은 **특정 서비스**에 만족하는 경향이 있는데 이런 차이를 **로지스틱 회귀 같은 단순 모델로 반영하기 어려운** 측면이 있다.

5. **데이터 불균형 또는 표본 크기 차이**

   로얄 고객의 클래스 간 표본 크기가 불균형할 가능성이 있어, 모델이 특정 클래스를 학습하는 데 어려움을 겪을 수 있다. 반면 비로얄 고객은 더 균일한 분포로 인해 예측이 더 잘 이루어진다.

이와 같은 이유로, 비로얄 고객의 모델이 더 높은 정확도를 보이며, 로얄 고객의 경우 더 정교한 모델 설계가 필요하다.

→ 따라서 로얄고객 모델의 경우
로지스틱 회귀 외에
랜덤 포레스트, **XGBoost와** 같은 **비선형 모델**을 사용하면, 로얄 고객의 복잡한 서비스 기대치 간의 상호작용을 더 잘 반영할 수 있다. 이러한 모델은 **비선형 관계**와 **다양한 변수 간의 상호작용**을 더 잘 학습할 수 있어 성능 향상에 기여할 수 있다.

XGBoost

기본 학습

본 XGBoost 모델을 학습하여 성능을 평가했다. 이 단계에서는 모델이 데이터를 어떻게 처리하고 예측하는지 기본적인 성능을 파악하는 것이 목적이다.

기본 모델을 학습한 후, **검증 세트**와 **테스트 세트**에 대한 성능을 평가했다. 검증 세트는 모델 성능을 측정하고 튜닝하는 데 사용되었으며, 테스트 세트는 모델의 **일반화 성능**을 확인하는 데 사용했다.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
import xgboost as xgb

# Load datasets
train_data = pd.read_csv('modified_airplane.csv')
test_data = pd.read_csv('modified_test.csv')

# Separate features and target
X = train_data.drop(columns=['satisfaction'])
y = train_data['satisfaction']

# Split based on Disloyal Customer
X_disloyal = X[X['disloyal Customer'] == 1]
y_disloyal = y[X['disloyal Customer'] == 1]
```

```
X_loyal = X[X['disloyal Customer'] == 0]
y_loyal = y[X['disloyal Customer'] == 0]


# 특성과 목표 변수 분리 (테스트 세트)
X_test = test_data.drop(columns=['satisfaction'])  # 테스트 세트에서 특성 데이터
y_test = test_data['satisfaction']  # 테스트 세트에서 목표 변수

# Split into train and validation sets
X_train_dis, X_val_dis, y_train_dis, y_val_dis = train_test_split(X_disloyal, y_disloyal,
test_size=0.2, random_state=42)
X_train_loy, X_val_loy, y_train_loy, y_val_loy = train_test_split(X_loyal, y_loyal, test_s
ize=0.2, random_state=42)

# Train model for disloyal customers
model_disloyal = xgb.XGBClassifier()
model_disloyal.fit(X_train_dis, y_train_dis)

# Train model for loyal customers
model_loyal = xgb.XGBClassifier()
model_loyal.fit(X_train_loy, y_train_loy)

# Predict and evaluate disloyal model
preds_dis = model_disloyal.predict(X_val_dis)
print("Disloyal Customer Model Report:")
print(classification_report(y_val_dis, preds_dis))

# Predict and evaluate loyal model
preds_loy = model_loyal.predict(X_val_loy)
print("Loyal Customer Model Report:")
print(classification_report(y_val_loy, preds_loy))
```

```
Disloyal Customer Model Report:
              precision    recall  f1-score   support

           0       0.95      0.96      0.96      2889
           1       0.87      0.84      0.85       898

    accuracy                           0.93      3787
   macro avg       0.91      0.90      0.91      3787
weighted avg       0.93      0.93      0.93      3787

Loyal Customer Model Report:
              precision    recall  f1-score   support

           0       0.96      0.98      0.97      8889
           1       0.98      0.96      0.97      8044

    accuracy                           0.97     16933
   macro avg       0.97      0.97      0.97     16933
weighted avg       0.97      0.97      0.97     16933
```

1. *비로얄고객의 만족도 분포는 class에 따른 차이가 없음*

2. *로얄고객은 eco 와 비즈니스별로 다른 양상을 보이며, 높은 상관관계를 가지는 컬럼이 서로 다름.*

위의 인사이트를 바탕으로 모델을 loyal 기준으로 분류해서 돌린결과

비로얄의 y= 1 즉, 비로얄 중 만족하는사람을 예측하는 부분에서 점수가 떨어지는 것을 볼 수 있었다. 상관관계 분석 및 인사이트의 경우에서도 로얄고객 대비 뚜렷한 경향이 드러나지 않았기때문에, 위와 같이 모델 결과에서도 예측 가능한 결과가 드러났다고 볼 수 있다.

```
# 1. 기본 XGBoost 모델 학습
model = xgb.XGBClassifier(scale_pos_weight=3.22, use_label_encoder=False, eval_metric='log
loss')
model.fit(X_train_dis_smote, y_train_dis_smote)

# 2. 검증 세트 예측 및 평가
y_pred_val = model.predict(X_val_dis)
print("Initial XGBoost Model Report (Validation Set):")
print(classification_report(y_val_dis, y_pred_val))

# 3. 테스트 세트 예측 및 평가
y_pred_test = model.predict(X_test)
print("Initial XGBoost Model Report (Test Set):")
print(classification_report(y_test, y_pred_test))

# 4. 테스트 세트 정확도 확인
test_accuracy = model.score(X_test, y_test)
print(f"Test Accuracy: {test_accuracy:.4f}")
```

```
Initial XGBoost Model Report (Validation Set):
              precision    recall  f1-score   support

           0       0.98      0.89      0.94      2890
           1       0.74      0.94      0.83       897

    accuracy                           0.91      3787
   macro avg       0.86      0.92      0.88      3787
weighted avg       0.92      0.91      0.91      3787

Initial XGBoost Model Report (Test Set):
              precision    recall  f1-score   support

           0       0.70      0.92      0.80     14573
           1       0.83      0.51      0.63     11403

    accuracy                           0.74     25976
   macro avg       0.77      0.71      0.71     25976
weighted avg       0.76      0.74      0.72     25976

Test Accuracy: 0.7393
```

- **검증 세트**: 모델이 학습되지 않은 데이터에서 얼마나 잘 예측하는지 평가하기 위해 사용.
- **테스트 세트**: 학습 후 실제로 모델이 새 데이터를 어떻게 예측하는지 평가하여 **일반화 성능**을 측정하기위해 사용.

임계값 조정

```
# 필요한 평가 함수들 임포트
from sklearn.metrics import precision_score, recall_score, f1_score
# 예측 확률 계산
```

```
y_proba = model.predict_proba(X_val_dis)[:, 1]

# 다양한 임계값을 사용하여 성능 평가
thresholds = np.arange(0.1, 0.9, 0.05)
best_threshold = 0.5
best_f1 = 0

print("Threshold 조정 결과:")
for threshold in thresholds:
    y_pred_adjusted = (y_proba >= threshold).astype(int)
    precision = precision_score(y_val_dis, y_pred_adjusted)
    recall = recall_score(y_val_dis, y_pred_adjusted)
    f1 = f1_score(y_val_dis, y_pred_adjusted)

    print(f"Threshold: {threshold:.2f}, Precision: {precision:.3f}, Recall: {recall:.3f},
F1-Score: {f1:.3f}")

    if f1 > best_f1:
        best_f1 = f1
        best_threshold = threshold

print(f"Best Threshold: {best_threshold:.2f}, Best F1-Score: {best_f1:.3f}")

# 최적 임계값으로 예측
y_pred_best = (y_proba >= best_threshold).astype(int)
print(f"Classification report with threshold {best_threshold}:")
print(classification_report(y_val_dis, y_pred_best))
```

```
Threshold 조정 결과:
Threshold: 0.10, Precision: 0.459, Recall: 0.980, F1-Score: 0.625
Threshold: 0.15, Precision: 0.512, Recall: 0.971, F1-Score: 0.670
Threshold: 0.20, Precision: 0.560, Recall: 0.970, F1-Score: 0.710
Threshold: 0.25, Precision: 0.598, Recall: 0.965, F1-Score: 0.739
Threshold: 0.30, Precision: 0.631, Recall: 0.957, F1-Score: 0.761
Threshold: 0.35, Precision: 0.659, Recall: 0.953, F1-Score: 0.779
Threshold: 0.40, Precision: 0.689, Recall: 0.950, F1-Score: 0.799
Threshold: 0.45, Precision: 0.715, Recall: 0.950, F1-Score: 0.816
Threshold: 0.50, Precision: 0.735, Recall: 0.942, F1-Score: 0.826
Threshold: 0.55, Precision: 0.757, Recall: 0.933, F1-Score: 0.836
Threshold: 0.60, Precision: 0.772, Recall: 0.926, F1-Score: 0.842
Threshold: 0.65, Precision: 0.787, Recall: 0.909, F1-Score: 0.844
Threshold: 0.70, Precision: 0.802, Recall: 0.893, F1-Score: 0.845
Threshold: 0.75, Precision: 0.816, Recall: 0.877, F1-Score: 0.846
Threshold: 0.80, Precision: 0.840, Recall: 0.842, F1-Score: 0.841
Threshold: 0.85, Precision: 0.869, Recall: 0.807, F1-Score: 0.837
Best Threshold: 0.75, Best F1-Score: 0.846
Classification report with threshold 0.7500000000000002:
              precision    recall  f1-score   support

           0       0.96      0.94      0.95      2890
           1       0.82      0.88      0.85       897

    accuracy                           0.92      3787
   macro avg       0.89      0.91      0.90      3787
weighted avg       0.93      0.92      0.93      3787
```

목적: **Recall**을 지나치게 높게 설정한 초기 모델에서는 **Precision**이 낮게 나타났기 때문에, 임계값 조정을 통해 **Precision**과 **Recall**의 균형을 맞추고 **F1-Score**를 최적화하려 했다.

임계값 0.75에서 최적의 **F1-Score**(0.846)를 달성했으며, **Precision**과 **Recall** 간의 균형이 잘 맞춰짐.

결과: 검증 세트에서의 **F1-Score**가 임계값 조정으로 **0.83에서 0.85**로 증가, 이는 만족 고객 예측 성능이 향상되었음을 보여준다.

하이퍼 파라미터 튜닝

모델의 성능을 최적화하여 **Precision**, **Recall**, **F1-Score**와 같은 주요 성능 지표를 개선하기 위함

```
from sklearn.model_selection import GridSearchCV

# 하이퍼파라미터 그리드 설정
param_grid = {
    'gamma': [0.1, 0.3],
    'learning_rate': [0.05, 0.1, 0.3],
    'max_depth': [5, 7, 9],
    'min_child_weight': [1, 3],
    'n_estimators': [100, 200, 300],
    'scale_pos_weight': [1, 2, 3],
    'subsample': [0.7, 0.8, 1.0]
}

# GridSearchCV 설정
grid_search = GridSearchCV(
    estimator=xgb.XGBClassifier(use_label_encoder=False, eval_metric='logloss'),
    param_grid=param_grid,
    scoring='f1',
    cv=5,
    verbose=1,
    n_jobs=-1
)

# Grid Search 수행
grid_search.fit(X_train_dis_smote, y_train_dis_smote)

# 최적의 파라미터와 성능 출력
print("Best parameters after advanced tuning:", grid_search.best_params_)
print("Best score:", grid_search.best_score_)
```

```
Best parameters after advanced tuning: {'gamma': 0.1, 'learning_rate': 0.1, 'max_depth':
7, 'min_child_weight': 1, 'n_estimators': 300, 'scale_pos_weight': 2, 'subsample': 0.8}
Best score: 0.9319005743219089
```

```
import xgboost as xgb
from sklearn.metrics import classification_report

# 최적의 하이퍼파라미터로 XGBoost 모델 정의
best_model_advanced = xgb.XGBClassifier(
    gamma=0.1,
    learning_rate=0.1,
    max_depth=7,
    min_child_weight=1,
```

```
    n_estimators=300,
    scale_pos_weight=2,
    subsample=0.8,
    eval_metric='logloss'
)

# 최적 하이퍼파라미터로 모델 학습
best_model_advanced.fit(X_train_dis_smote, y_train_dis_smote)

# 검증 세트에서 예측
y_pred_val = best_model_advanced.predict(X_val_dis)

# 검증 세트 성능 평가
print("Validation Set Model Report:")
print(classification_report(y_val_dis, y_pred_val))
```

```
Validation Set Model Report:
              precision    recall  f1-score   support

           0       0.97      0.92      0.95      2890
           1       0.79      0.91      0.84       897

    accuracy                           0.92      3787
   macro avg       0.88      0.92      0.89      3787
weighted avg       0.93      0.92      0.92      3787
```

결과적으로 임계값을 사용하여 검증세트에 적용 한 모델을 가지고 test 세트에서의 성능평가를 해볼 수 있다.

```
# ------------------------------------------------------------
# 최적 임계값을 테스트 세트에 적용하여 성능 평가
# ------------------------------------------------------------

# 테스트 세트에서 예측 확률 계산
y_proba_test = model.predict_proba(X_test)[:, 1]

# 최적 임계값을 사용하여 테스트 세트에서 예측
y_pred_best_test = (y_proba_test >= best_threshold).astype(int)

# 테스트 세트 성능 평가
print(f"Classification report with threshold {best_threshold} on Test Set:")
print(classification_report(y_test, y_pred_best_test))

# 최종 정확도 출력
test_accuracy = precision_score(y_test, y_pred_best_test)
print(f"Test Set Accuracy with threshold {best_threshold}: {test_accuracy:.4f}")
```

```
              precision    recall  f1-score   support

           0       0.70      0.94      0.80     14573
           1       0.86      0.48      0.62     11403

    accuracy                           0.74     25976
```

```
   macro avg       0.78      0.71      0.71     25976
weighted avg       0.77      0.74      0.72     25976
```

Random Forest

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

# train_clean 데이터프레임에서 'satisfaction' 열을 제외한 나머지 열을 X로, 'satisfaction' 열을 y로 설정
X_train = train_clean.drop('satisfaction', axis=1)
y_train = train_clean['satisfaction']

# test_clean 데이터프레임에서 'satisfaction' 열을 제외한 나머지 열을 X로, 'satisfaction' 열을 y로 설정
X_test = test_clean.drop('satisfaction', axis=1)
y_test = test_clean['satisfaction']

# Random Forest 모델 생성 및 훈련
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)

# 테스트 세트를 사용하여 모델 예측
y_pred = model.predict(X_test)

# 모델 정확도 평가
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

# ---> Accuracy: 0.6233061287342162
```

```
from sklearn.model_selection import cross_val_score

# 교차 검증 (5-fold cross-validation)
scores = cross_val_score(model, X_train, y_train, cv=5)

print("Cross-Validation Scores:", scores)
print("Average Accuracy:", scores.mean())

# ---> Cross-Validation Scores: [0.95854047 0.95858873 0.9568512  0.95892659 0.95950381]
# ---> Average Accuracy: 0.958482161338029
```

```
from sklearn.model_selection import GridSearchCV

# Define the parameter grid to search
param_grid = {
    'n_estimators': [50, 100, 200],  # Number of trees in the forest
    'max_depth': [None, 10, 20],     # Maximum depth of the trees
    'min_samples_split': [2, 5, 10], # Minimum number of samples required to split an inte
rnal node
    'min_samples_leaf': [1, 2, 4]    # Minimum number of samples required to be at a leaf
node
}
```

```
# ---> Accuracy with best hyperparameters: 0.6127579303972898
Best Hyperparameters: {'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 5,
'n_estimators': 200}
Best Hyperparameters: 0.9605286154464354
```

```
param_grid_1 = {
    'n_estimators': [50, 100, 200],
    'max_depth': [3,5, 7],
    'min_samples_leaf': [1, 2, 4]
}

# ---> Accuracy with best hyperparameters: 0.6579534955343393
Best Hyperparameters: {'max_depth': 7, 'min_samples_leaf': 2, 'n_estimators': 200}
Best Hyperparameters: 0.9362801101091999
```

```
X_train_loyal = train_clean[train_clean['disloyal Customer'] == 0].drop('satisfaction', ax
is=1)
y_train_loyal = train_clean[train_clean['disloyal Customer'] == 0]['satisfaction']

X_train_disloyal = train_clean[train_clean['disloyal Customer'] == 1].drop('satisfaction',
axis=1)
y_train_disloyal = train_clean[train_clean['disloyal Customer'] == 1]['satisfaction']

X_test_loyal = test_clean[test_clean['disloyal Customer'] == 0].drop('satisfaction', axis=
1)
y_test_loyal = test_clean[test_clean['disloyal Customer'] == 0]['satisfaction']

X_test_disloyal = test_clean[test_clean['disloyal Customer'] == 1].drop('satisfaction', ax
is=1)
y_test_disloyal = test_clean[test_clean['disloyal Customer'] == 1]['satisfaction']

# ---> loyal Accuracy: 0.4905794021816121
# ---> disloyal Accuracy: 0.923734111273182
```

```
# loyal customer
Best Score: 0.9440363165612967
Accuracy with best hyperparameters: 0.562780374935071

# disloyal customer
Best Score: 0.9334460504357555
Accuracy with best hyperparameters: 0.932485934569702
```

- 일단 교차검증으로는 이 모델이 잘 만들어진 모델이라고 확인할 수는 없음.
  - 기본적인 모델도 교차검증은 95%언저리가 나오는데 막상 테스트셋으로 확인해보면 정확도가 62%으로 떨어짐.
  - 다른 하이퍼파라미터튜닝으로 비교분석을 해본 경우에도 마찬가지로, 조금씩 차이가 있을 뿐 전부 60~65% 언저리에서 머물고 있었음.
- 다만, 로얄고객과 비로얄고객을 나눠 Random Forest를 돌린 결과는 달랐음.
  - 로얄고객의 정확도는 최대 56%가 나오면서 모델의 성능을 개선할 가능성이 **매우 낮음.**
  - 대신 비로얄고객의 경우 최대 93%의 수치로 매우 높은 값을 가져가는것을 확인함. 이는 EDA분석결과 로얄고객의 특징이 더 많이 보인다라는 1차적인 결과와 매우 다른 양상을 보여주고 있음.
  - 전체적인 데이터셋에 대한 모델 성능이 높지 않았던 이유는 **다름 아닌 로얄 고객**이라는 판단을 하게 됨.
- 하지만 재현율을 살펴보면 낮은 수치를 띄고 있음. 사실 비로얄 사람들의 불만족 비율이 높다보니, 어지간히 불만족으로 예측을 해버리면 정확도를 높일 수 있게 됨. 사실상 큰 의미가 있는 정확도인지도 확인을 해봐야된다.

앙상블

이 코드는 **Stacking 앙상블 모델**을 사용하여, **Disloyal 고객**의 만족도를 예측하는 모델을 구성한다. 스태킹 앙상블 모델은 여러 모델을 결합하여 더 나은 성능을 얻기 위한 기법이다. 모델 설명을 다음과 같이 요약할 수 있다.

Stacking 앙상블 모델 개념

Stacking은 여러 개의 머신러닝 모델(기본 학습기)을 학습한 후, 그 결과를 기반으로 다시 메타 모델을 학습하는 방법이다. 기본 모델이 각각 다른 성능을 보이는 경우, 이를 결합하여 더 나은 예측 성능을 기대할 수 있다. 스태킹은 개별 모델의 약점을 보완하면서도 다양한 예측을 통합하여 성능을 향상시키는 데 도움을 준다.

기본 학습기 (Base Models):

- **XGBoost**: 트리 기반 부스팅 앙상블 모델로, 비선형 데이터와 복잡한 상호작용을 잘 학습하는 특성이 있다. 빠르고 강력한 성능을 제공하며, 과적합 방지 메커니즘이 강력하다.
- **Random Forest**: 여러 개의 결정 트리를 훈련시키고 그 결과를 평균내어 최종 예측을 수행하는 앙상블 모델이다. 다양한 트리 기반의 예측을 결합하여 과적합을 줄이고 안정적인 성능을 제공한다.

메타 모델 (Meta Model):

- **Logistic Regression**: 기본 학습기들의 예측 결과를 조합하여 최종 예측을 수행하는 메타 모델이다. 스태킹 앙상블에서 메타 모델은 기본 모델들의 출력을 입력으로 받아 학습하며, 최종 출력 결과를 생성한다. 이 경우 **로지스틱 회귀**는 확률 기반의 선형 모델로, 성능을 안정적으로 통합할 수 있다.

```
# XGBoost, Random Forest, Logistic Regression 모델 정의
xgb_model = XGBClassifier(n_estimators=200, random_state=42, n_jobs=-1)
rf_model = RandomForestClassifier(n_estimators=200, random_state=42, n_jobs=-1)
lr_model = LogisticRegression(max_iter=1000)

# 스태킹 앙상블 모델 정의 (XGB, RF, Logistic Regression)
stacking_model = StackingClassifier(
    estimators=[
        ('xgb', xgb_model),
        ('rf', rf_model)
    ],
    final_estimator=lr_model,  # 메타 모델로 Logistic Regression 사용
    n_jobs=-1  # 병렬 처리 사용
)

# 모델 학습 (Disloyal 고객)
stacking_model.fit(X_train_dis, y_train_dis)

# 검증 세트 성능 평가
y_pred_val_dis = stacking_model.predict(X_val_dis)
print("Disloyal Customer Model Report (Validation Set):")
print(classification_report(y_val_dis, y_pred_val_dis))

# 테스트 세트 성능 평가
y_pred_test_dis = stacking_model.predict(X_test_dis)
print("Disloyal Customer Model Report (Test Set):")
print(classification_report(y_test_dis, y_pred_test_dis))
```

```
Disloyal Customer Model Report (Validation Set):
              precision    recall  f1-score   support

           0       0.95      0.96      0.96      2889
```

```
           1       0.88      0.83      0.85       898

    accuracy                           0.93      3787
   macro avg       0.91      0.90      0.91      3787
weighted avg       0.93      0.93      0.93      3787

Disloyal Customer Model Report (Test Set):
              precision    recall  f1-score   support

           0       0.93      0.96      0.95      3591
           1       0.88      0.79      0.83      1208

    accuracy                           0.92      4799
   macro avg       0.90      0.87      0.89      4799
weighted avg       0.92      0.92      0.92      4799
```

5.결론

분석결론

- **기내 서비스 개선 및 통합:** 기내 서비스 항목인 음식, 좌석 편안함, 오락 시설, 청결도는 높은 상관관계를 가지며 승객들은 이를 종합적으로 경험하고 평가하는 경향을 보인다. 따라서 기내 서비스를 통합적으로 관리하고 개선하는 것이 필요하다. 이를 위해 **기내 서비스 패키지**를 운영하여 특정 서비스를 개선할 때 다른 서비스에도 긍정적인 영향을 주도록 한다.
- **항공 클래스 및 여행 타입별 맞춤형 서비스 제공**: 이코노미 클래스는 여행 목적에 따라 만족도 차이가 발생하는 반면, 비즈니스 클래스는 일관되게 높은 만족도를 보인다. 비즈니스 목적의 승객이 이코노미 좌석을 이용할 경우 만족도가 낮아지는 경향이 있으므로, 이코노미 클래스에서도 **개인 여행객**과 **비즈니스 여행객**을 구분해 맞춤형 서비스를 제공하도록 한다. 이를 통해 **우선 탑승**, **좌석 업그레이드 제안** 등 혜택을 제공하여 승객의 만족도를 높인다.
- **연령대에 따른 만족도 차이 및 서비스 차별화:** 40-50대 승객은 비즈니스 클래스에서 매우 높은 만족도를 보이는 반면, 20대 승객은 이코노미 클래스에서 상대적으로 낮은 만족도를 보인다. 특히, 20대 승객들은 **온라인 예약 시스템**과 **와이파이 서비스**에 불만이 크므로, 디지털 서비스 강화를 통해 만족도를 높인다. **중년층 전용 서비스 패키지**를 통해 비즈니스 클래스의 맞춤형 서비스를 제공함으로써 이들의 높은 만족도를 유지할 수 있다.
- **로얄 고객과 비로얄 고객에 대한 차별화 전략**: 로얄 고객은 모든 서비스에서 고른 만족을 원하지만, 비로얄 고객은 **와이파이 서비스**, **온라인 예약**, **탑승 편리성** 같은 특정 서비스에 더 민감하다. **로얄 고객**에게는 **고품질 VIP 서비스**를 제공하고, **비로얄 고객**에게는 온라인 서비스의 편리성을 높이는 방향으로 서비스 전략을 차별화해야 한다.
- **지연 시간과 만족도**: 출발 및 도착 지연은 승객의 만족도에 큰 영향을 미치지 않으며, 고객들은 이를 외부 요인으로 인식하는 경향이 있다. 따라서 **지연 자체를 최소화하기보다는** 지연 시 고객에게 미리 안내하거나 **대체 서비스**를 제공하는 것이 고객 만족도를 높이는 데 효과적이다.
- **데이터 기반 모델 적용 제안**: 클래스, 여행 목적, 로얄 여부 등을 기준으로 데이터를 세분화한 후, 각각의 집단에 맞는 **다중 분류 모델**을 적용하는 것이 효과적이다. 특히, **로얄 고객과 비로얄 고객**을 별도로 구분하여 두 개의 모델을 개발해 각 집단에 맞는 예측을 최적화하도록 한다.

모델 해석

본 분석에서는 세 가지 주요 머신러닝 모델인
XGBoost, **Random Forest**, **로지스틱 회귀**를 사용하여 **로얄 고객**과 **비로얄 고객**의 만족도를 예측하는 모델을 구축하고, 각각의 성능을 비교하였다. 또한, 이 세 모델을 기반으로 **앙상블(스태킹 모델)**을 적용하여 성능을 향상시키는 접근도 시도하였다. 이를 바탕으로 최종적으로 가장 적합한 모델을 선택하고자 한다.

모델 해석 및 비교

본 분석에서는 두 가지 주요 머신러닝 모델인 **XGBoost**와 **Random Forest**를 사용하여 **로얄고객**과 **비로얄고객**의 만족도를 예측하는 모델을 구축하고, 각각의 성능을 비교하였다. 이를 바탕으로 최종적으로 가장 적합한 모델을 선택하고자 한다.

- 결과 분석

	로얄 고객	비로얄고객
로지스틱 회귀	미탐지가 많아 실제 만족한 고객을 불만족으로 예측하는 문제가 컸다. 이는 모델이 로얄 고객의 복잡한 특성을 제대로 반영하지 못한 결과이다.	미탐지와 오탐지가 적었고, 특히 불만족 고객을 잘 예측하는 재현율(Recall)이 높았다.
XGBoost	오탐지와 미탐지가 거의 없어서 매우 높은 성능을 보였다. 이는 XGBoost가 로얄 고객의 복잡한 상호작용을 잘 학습했기 때문이다.	미탐지가 적었고, 임계값 조정을 통해 오탐지를 줄여 정밀도(Precision)와 **재현율(Recall)** 간의 균형을 맞추었다.
Random Forest	오탐지와 미탐지가 많아, 실제 고객 만족을 예측하는 데 어려움이 있었다. 모델이 로얄 고객의 복잡한 요구 사항을 잘 반영하지 못했다.	미탐지와 오탐지가 모두 적었으며, 매우 높은 정확도를 기록했다. 재현율(Recall)이 높아 불만족 고객을 잘 예측했다.
앙상블 모델	오탐지와 미탐지가 가장 적었으며, 정밀도(Precision)와 **재현율(Recall)** 간의 균형이 뛰어나 최종 성능이 향상되었다.	**XGBoost**와 **랜덤 포레스트**의 강점을 결합하여 높은 정확도와 균형 잡힌 성능을 보여준다.

결론 및 솔루션

1. **로얄 고객**:
   - 로얄 고객의 복잡한 요구를 반영하기 위해 **XGBoost** 모델을 사용하고, 추가적으로 **하이퍼파라미터 튜닝**과 **임계값 조정**을 통해 예측 성능을 향상시키는 것이 적합하다. 또한 **앙상블 모델**을 통해 여러 모델의 강점을 결합할 수 있다.
2. **비로얄 고객**:
   - **솔루션**: 비로얄 고객의 예측에는 **XGBoost**나 **랜덤 포레스트**를 사용하는 것이 적합하며, 특히 **불만족 고객**을 잘 예측하는 **재현율(Recall)**을 높게 유지해야 한다.

- 모델 선택:
  - **로얄 고객**에 대해서는 **XGBoost**와 **앙상블 모델**을 사용하는 것이 가장 적합하다. **XGBoost**는 복잡한 상호작용을 반영하며, 정밀도(Precision)와 재현율(Recall)의 균형을 맞출 수 있다. 또한, **앙상블 모델**을 통해 여러 모델의 성능을 결합하여 성능을 더욱 향상시킬 수 있다.
  - **비로얄 고객**에 대해서는 **랜덤 포레스트**나 **XGBoost** 모두 적합하지만, **XGBoost**가 **임계값 조정**을 통해 정밀도(Precision)와 재현율(Recall)의 균형을 더 잘 맞출 수 있다. 특히, 비로얄 고객은 불만족 고객을 더 잘 예측해야 하므로 재현율(Recall)을 최적화하는 것이 중요하다.
- **고객 세분화 모델**:
  - 로얄 고객과 비로얄 고객을 구분하는 방식처럼, 추가적인 고객 세분화를 고려할 수 있다. 예를 들어, **여행 목적**에 따라 고객을 나누어 각각 다른 모델을 설계함으로써, 더 세밀한 예측 성능을 달성할 수 있다. 이는 특정 목적에 맞춘 고객군을 구분함으로써 예측을 더 정확하게 할 수 있게 한다.
- **실시간 모델 업데이트**:
  - 고객 만족도를 실시간으로 반영할 수 있는 **온라인 러닝 기법**(Online Learning)을 적용하여, 실시간 데이터를 바탕으로 모델을 지속적으로 업데이트할 수 있다.
  - 이를 통해 항공사는 고객 요구에 즉각적으로 대응하고, 모델 예측 성능을 유지할 수 있다. 특히, 고객의 실시간 피드백을 반영하여 만족도 예측을 빠르게 개선할 수 있다.