2021-07-19

MEDICAL APPOINTMENT NO-SHOW IN BRAZIL

TEAM-4

CONTENTS

01. 전처리 과정

```
data.describe() # Age값 -1 발견
```

	PatientId	AppointmentID	Age	Scholarship	Hipertension	Diabetes
count	1.105270e+05	1.105270e+05	110527.000000	110527.000000	110527.000000	110527.000000
mean	1.474963e+14	5.675305e+06	37.088874	0.098266	0.197246	0.071865
std	2.560949e+14	7.129575e+04	23.110205	0.297675	0.397921	0.258265
min	3.921784e+04	5.030230e+06	-1.000000	0.000000	0.000000	0.000000
25%	4.172614e+12	5.640286e+06	18.000000	0.000000	0.000000	0.000000
50%	3.173184e+13	5.680573e+06	37.000000	0.000000	0.000000	0.000000
75%	9.439172e+13	5.725524e+06	55.000000	0.000000	0.000000	0.000000
max	9.999816e+14	5.790484e+06	115.000000	1.000000	1.000000	1.000000

```
# Age값 -1 제거
data = data.loc[data['Age'] != -1]
```

```
# PatientID, AppointmentID 제거
data = data.drop(["PatientId","AppointmentID"], axis = 1)
```

01. 전처리 과정

```
# Schedule Date & Appointment Date 년월일시간 분리
data = data.rename(columns = {'ScheduledDay' : "ScheduledDate" , "AppointmentDay" : "AppointmentDate"})
```

```
data['ScheduledYear'] = data['ScheduledDate'].apply(lambda x: np.int(x[0:4]))
data['ScheduledMonth'] = data['ScheduledDate'].apply(lambda x: np.int(x[5:7]))
data['ScheduledDay'] = data['ScheduledDate'].apply(lambda x: np.int(x[8:10]))
data['ScheduledHour'] = data['ScheduledDate'].apply(lambda x: np.int(x[11:13]))
data['ScheduledMinute'] = data['ScheduledDate'].apply(lambda x: np.int(x[14:16]))
```

```
data['AppointmentYear'] = data['AppointmentDate'].apply(lambda x: np.int(x[0:4]))
data['AppointmentMonth'] = data['AppointmentDate'].apply(lambda x: np.int(x[5:7]))
data['AppointmentDay'] = data['AppointmentDate'].apply(lambda x: np.int(x[8:10]))
```

```
# Schedule YMD 생성 (ScheduleDate & AppointmentDate에서 년월일시간만 추출 )
data['ScheduledYMD'] = data['ScheduledDate'].apply(lambda x: np.str(x[0:10]))
data['AppointmentYMD'] = data['AppointmentDate'].apply(lambda x: np.str(x[0:10]))
```

AppointmentYear	AppointmentMonth	AppointmentDay	ScheduledYMD	AppointmentYMD
2016	4	29	2016-04-29	2016-04-29
2016	4	29	2016-04-29	2016-04-29
2016	4	29	2016-04-29	2016-04-29
2016	4	29	2016-04-29	2016-04-29
2016	4	29	2016-04-29	2016-04-29
...	...	...	...	...
2016	6	7	2016-05-03	2016-06-07
2016	6	7	2016-05-03	2016-06-07
2016	6	7	2016-04-27	2016-06-07
2016	6	7	2016-04-27	2016-06-07
2016	6	7	2016-04-27	2016-06-07

02. 연령대별 No-Show

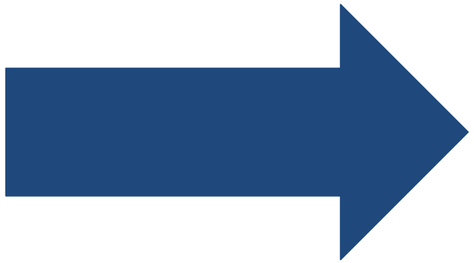
```
age_rng = list(range(0, 120, 10))
```

```
age_rng_label = [str(x) + '대' for x in age_rng]
```

```
age_rng_label
```

```
['0대',
 '10대',
 '20대',
 '30대',
 '40대',
 '50대',
 '60대',
 '70대',
 '80대',
 '90대',
 '100대',
 '110대']
```

```
df['AgeGroup'] = pd.cut(df['Age'], age_rng, right = False, labels = age_rng_label[:-1])
```

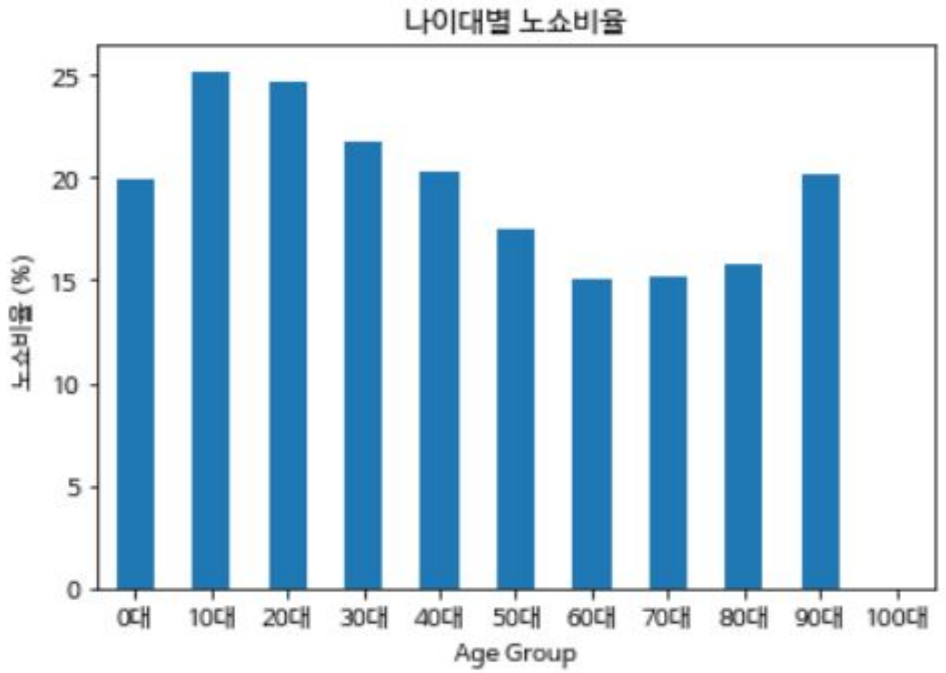


AgeGroup
60대
50대
60대
0대
50대
...
50대
50대
20대
30대
50대

02. 연령대별 No-Show

```
age_analysis = df['No-show'].eq('Yes').groupby(df.AgeGroup).mean().reset_index(name = 'proportion_by_age_group')
age_analysis['proportion_by_age_group'] = age_analysis['proportion_by_age_group']*100
```

```
age_analysis.plot.bar(x = 'AgeGroup', y = 'proportion_by_age_group', rot = 0,
           xlabel = 'Age Group', ylabel = '노쇼비율 (%)',
           title = '나이대별 노쇼비율', legend = 0)
plt.show()
```



03. 지역별 No-Show (Top10)

```
region_analysis = df['No-show'].eq('Yes').groupby(df.Neighbourhood).mean().reset_index(name = 'proportion_by_Neighbourhood')
```

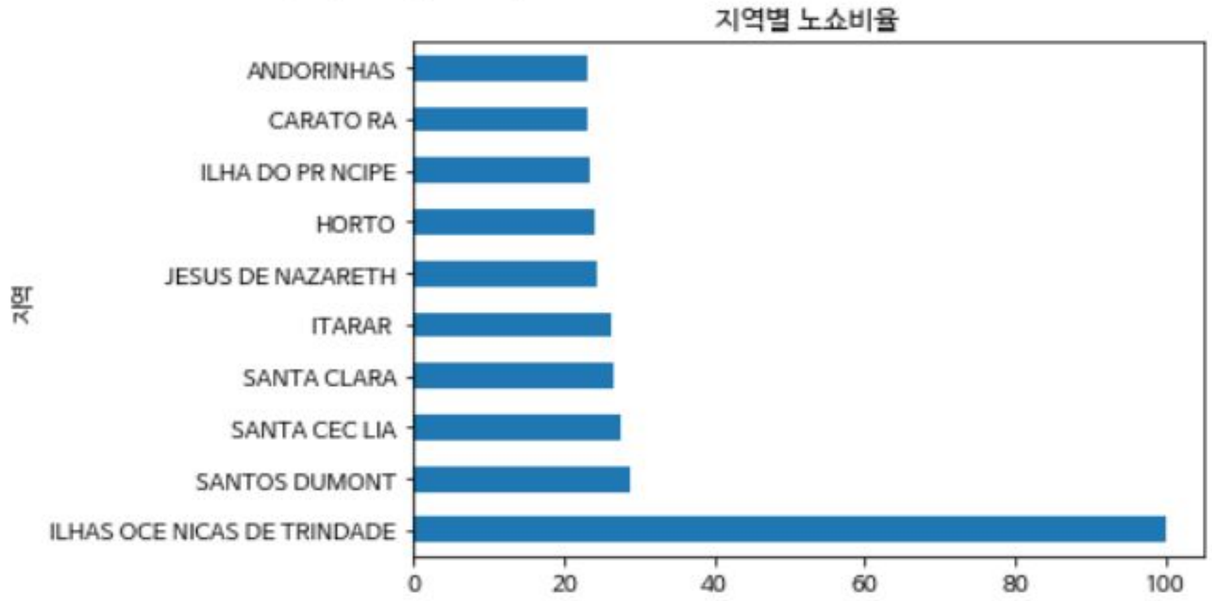
```
region_analysis['proportion_by_Neighbourhood'] = region_analysis['proportion_by_Neighbourhood']*100
```

```
region_analysis = region_analysis.sort_values('proportion_by_Neighbourhood', ascending = False)
```

```
#노쇼 지역별 top10
region_analysis_top10.plot.barh(x = 'Neighbourhood', y = 'proportion_by_Neighbourhood', rot = 0,
           xlabel = '지역', ylabel = '노쇼비율 (%)',
           title = '지역별 노쇼비율', legend = 0)
plt.show()
```

```
region_analysis_top10
```

	Neighbourhood	proportion_by_Neighbourhood
34	ILHAS OCEÂNICAS DE TRINDADE	100.000000
70	SANTOS DUMONT	28.918495
61	SANTA CECÍLIA	27.455357
62	SANTA CLARA	26.482213
36	ITARARÉ	26.266363
40	JESUS DE NAZARETH	24.395373
28	HORTO	24.000000
33	ILHA DO PRÍNCIPE	23.477493
9	CARATOÍRA	23.040936
1	ANDORINHAS	23.032714



03. 지역별 No-Show (Bottom 10)

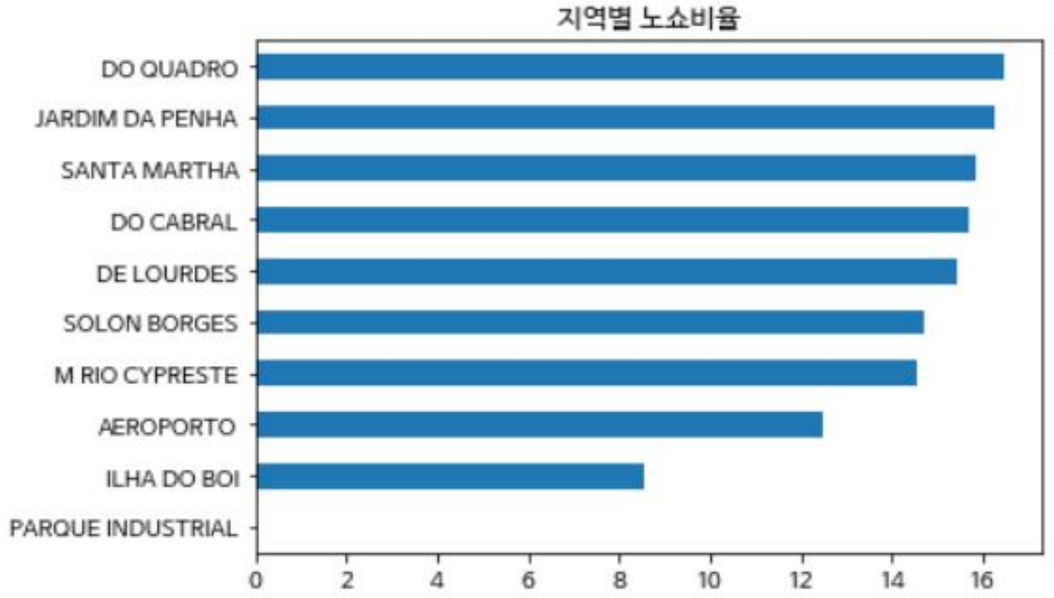
```
region_analysis = region_analysis.sort_values('proportion_by_Neighbourhood', ascending = True)
```

```
region_analysis_bottom10 = region_analysis.head(10)
```

```
#노쇼 지역별 bottom10
region_analysis_bottom10.plot.barh(x = 'Neighbourhood', y = 'proportion_by_Neighbourhood', rot = 0,
           xlabel = '지역', ylabel = '노쇼비율 (%)',
           title = '지역별 노쇼비율', legend = 0)
plt.show()
```

```
region_analysis_bottom10
```

	Neighbourhood	proportion_by_Neighbourhood
51	PARQUE INDUSTRIAL	0.000000
31	ILHA DO BOI	8.571429
0	AEROPORTO	12.500000
48	MÁRIO CYPRESTE	14.555256
73	SOLON BORGES	14.712154
16	DE LOURDES	15.409836
17	DO CABRAL	15.714286
66	SANTA MARTHA	15.841584
39	JARDIM DA PENHA	16.275471
19	DO QUADRO	16.489988

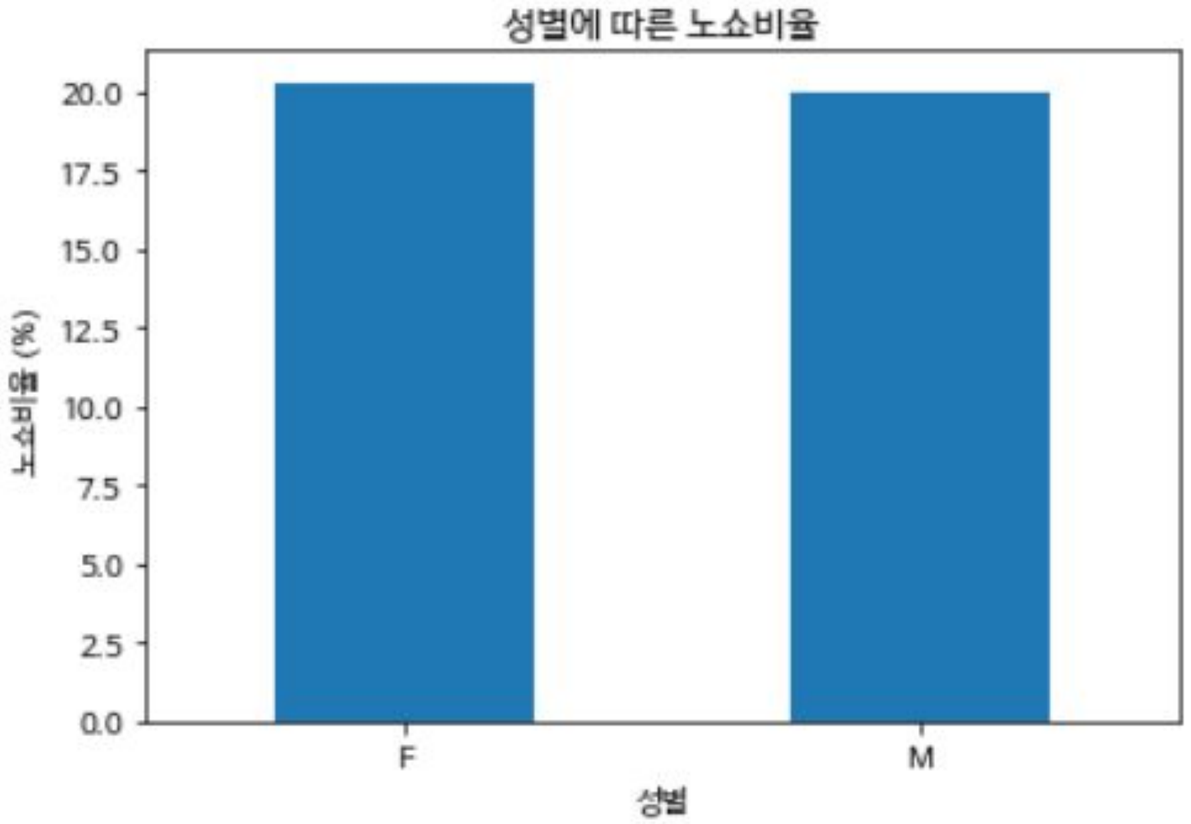


04. 성별 No-Show

```
[ ] gender_analysis = df['No-show'].eq('Yes').groupby(df.Gender).mean().reset_index(name = 'proportion_by_gender')
```

```
gender_analysis['proportion_by_gender'] = gender_analysis['proportion_by_gender']*100
```

	Gender	proportion_by_gender
0	F	20.314871
1	M	19.967948



05. 기다린 시간에 따른 No-Show

```
df['AppointmentYMD'] = pd.to_datetime(df['AppointmentYMD'])
df['ScheduledYMD'] = pd.to_datetime(df['ScheduledYMD'])
```

```
df['날짜차이'] = df['AppointmentYMD'] - df['ScheduledYMD']
```

```
type(df['날짜차이'])
```

```
pandas.core.series.Series
```

```
df["날짜차이"] = (df["날짜차이"]).dt.days
```

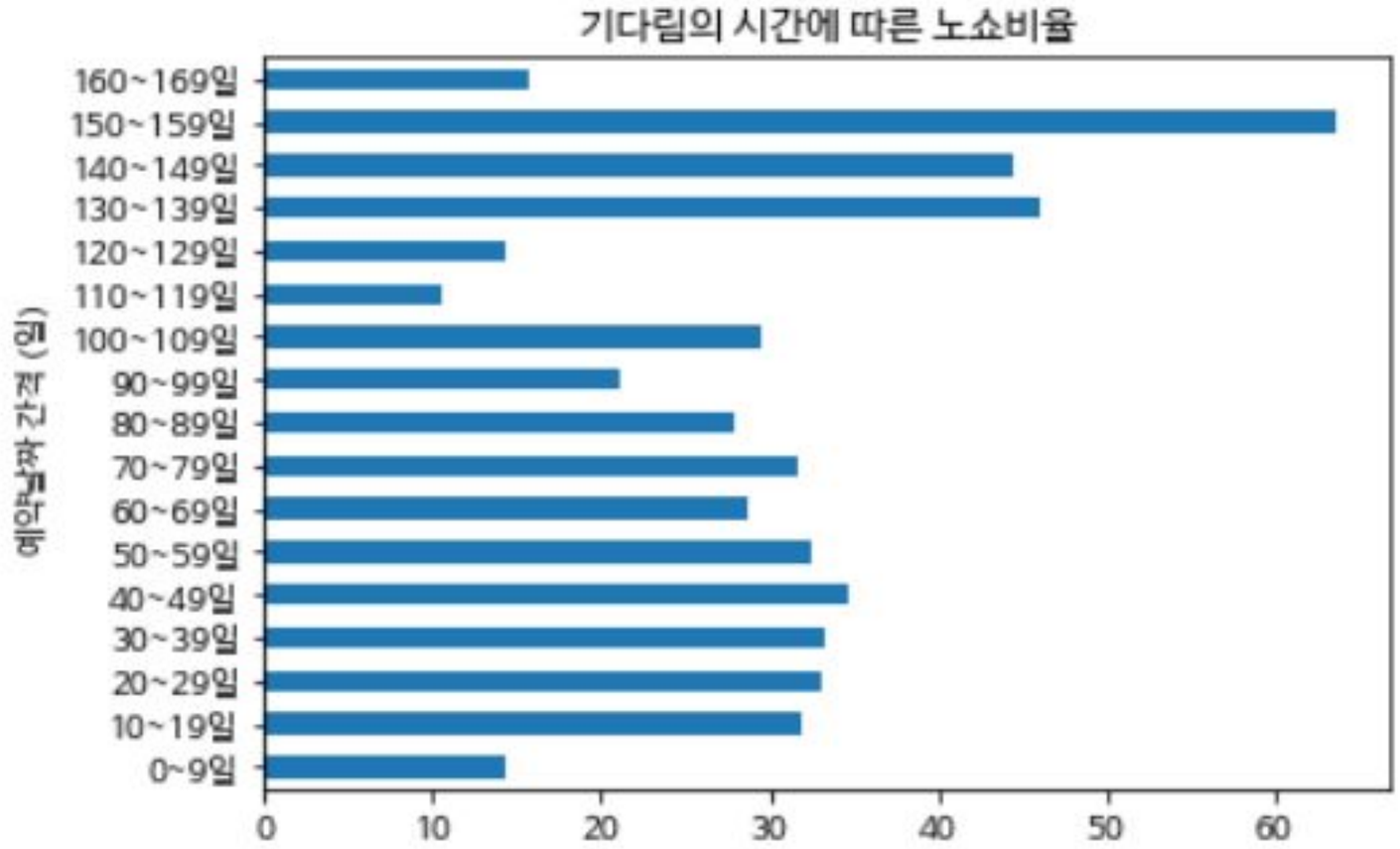
날짜차이
110526.000000
10.183794
15.255034
-6.000000
0.000000
4.000000
15.000000
179.000000

05. 기다린 시간에 따른 No-Show

```
['0~9일',
 '10~19일',
 '20~29일',
 '30~39일',
 '40~49일',
 '50~59일',
 '60~69일',
 '70~79일',
 '80~89일',
 '90~99일',
 '100~109일',
 '110~119일',
 '120~129일',
 '130~139일',
 '140~149일',
 '150~159일',
 '160~169일',
 '170~179일']
```

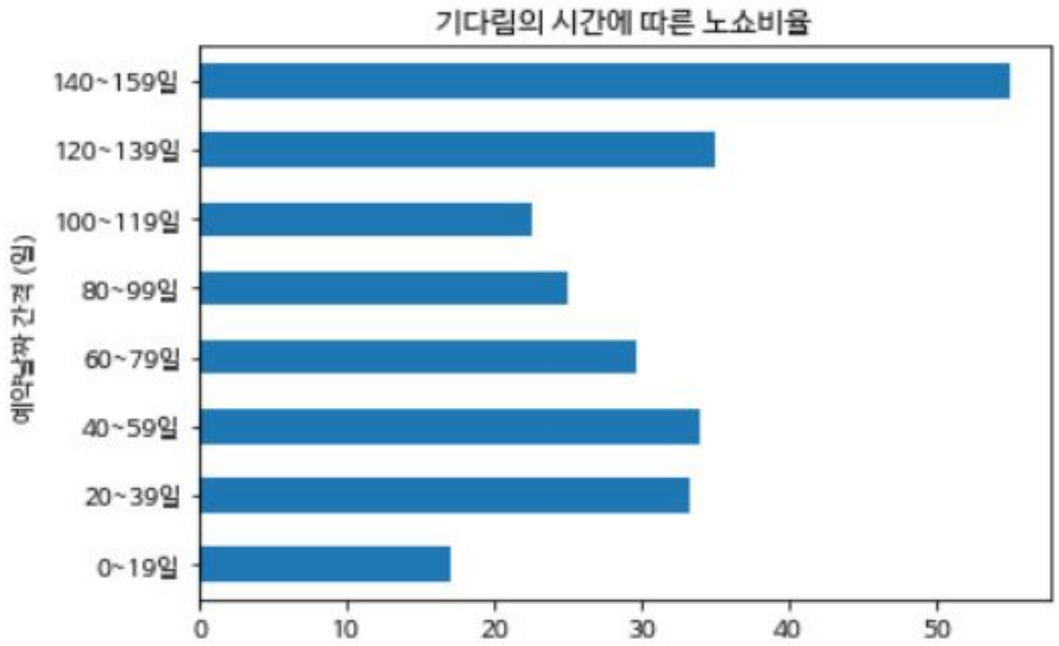
```
diff_rng_label = [str(x) + '~' + str(x+9) + '일'for x in diff_rng]
```

```
df['날짜차이그룹'] = pd.cut(df['날짜차이'], diff_rng, right = False, labels = diff_rng_label[:-1])
```



05. 기다린 시간에 따른 No-Show (20일단위)

```
# (추가) 20일 단위
diff_rng2 = list(range(0, 180, 20))
diff_rng_label2 = [str(x) + '~' + str(x+19) + '일'for x in diff_rng2]
df['날짜차이그룹2'] = pd.cut(df['날짜차이'], diff_rng2, right = False, labels = diff_rng_label2[:-1])
```



05. 기다린 시간에 따른 No-Show – 당일예약

```
p = df['No-show'].eq('Yes').groupby(df['날짜차이그룹']).mean().reset_index(name = 'proportion_by_date')
```

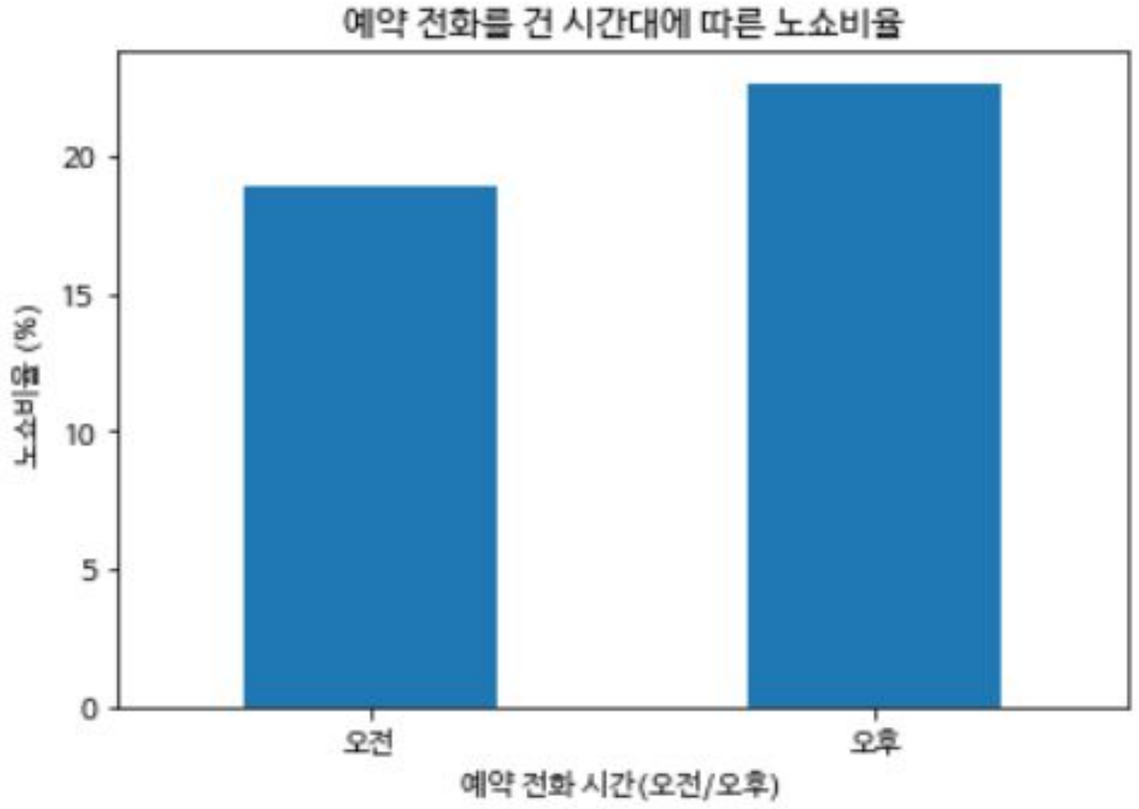
```
p['proportion_by_date'] = p['proportion_by_date']*100
```

```
p.plot.bar(x = '날짜차이그룹', y = 'proportion_by_date', rot = 0,
           xlabel = '예약날짜 간격 (일)', ylabel = '노쇼비율 (%)',
           title = '기다림의 시간에 따른 노쇼비율', legend = 0)
plt.show()
```

06. 예약시간에 따른No-Show

```
df['오전/오후'] = np.where(df['ScheduledHour'] > 12 ,'오후','오전')
```

```
call_time_analysis = df['No-show'].eq('Yes').groupby(df['오전/오후']).mean().reset_index(name = 'proportion_by_timeset')
```

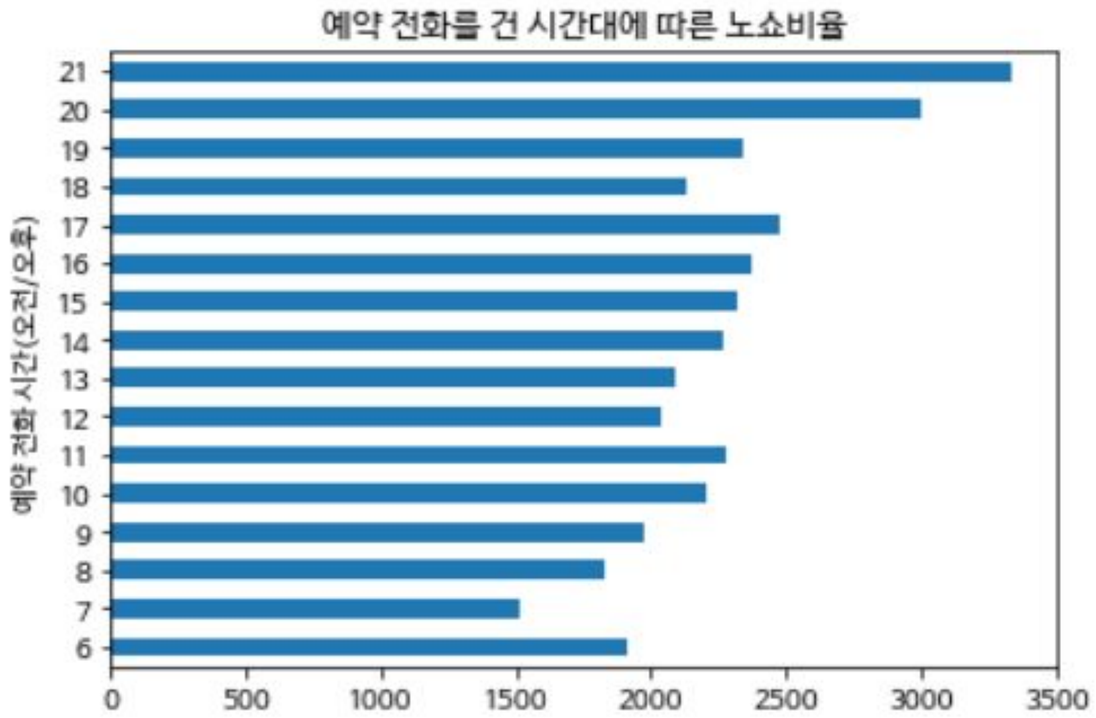


날짜 차이 그룹	날짜 차이 그룹2	오전/오후
0~9일	0~19일	오후
0~9일	0~19일	오후
0~9일	0~19일	오후
0~9일	0~19일	오후
0~9일	0~19일	오후
...	...	...
30~39일	20~39일	오전
30~39일	20~39일	오전
40~49일	40~59일	오후
40~49일	40~59일	오후
40~49일	40~59일	오후

06. 예약시간에 따른No-Show

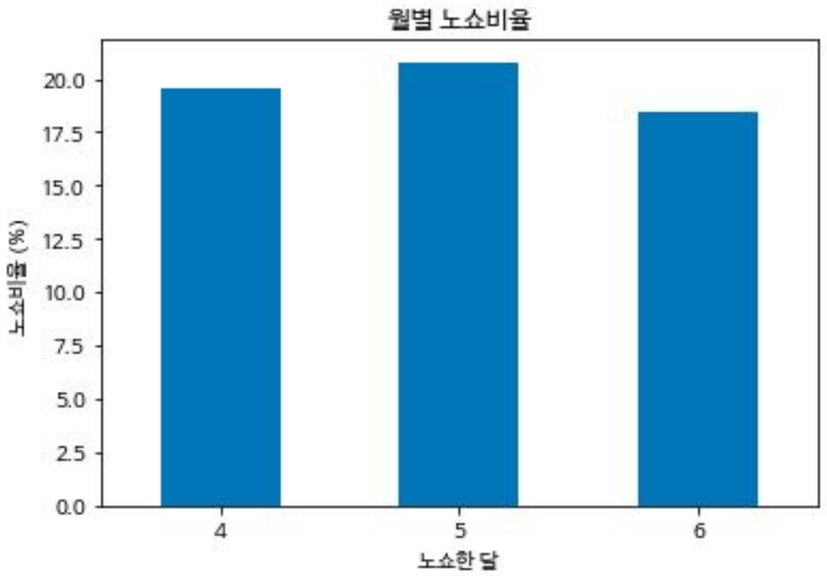
```
call_time_analysis2 = df['No-show'].eq('Yes').groupby(df['ScheduledHour']).mean().reset_index(name = 'proportion_by_calltime')
call_time_analysis2['proportion_by_calltime'] = call_time_analysis2['proportion_by_calltime']*100
call_time_analysis2 = call_time_analysis2.sort_values('ScheduledHour', ascending = True)
call_time_analysis2
```

	ScheduledHour	proportion_by_calltime
0	6	19.150285
1	7	15.151200
2	8	18.269481
3	9	19.698978
4	10	22.062415
5	11	22.784212
6	12	20.361490
7	13	20.909896
8	14	22.671488
9	15	23.183562
10	16	23.763984
11	17	24.819526
12	18	21.268657
13	19	23.360656
14	20	30.000000
15	21	33.333333



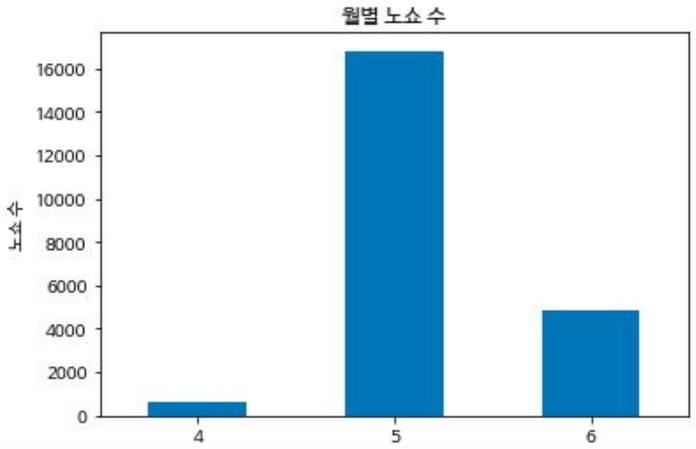
07. 월별No-Show (비율)

```
month_analysis.plot.bar(x = 'AppointmentMonth', y = 'proportion_by_month', rot = 0,
            xlabel = '노쇼한 달', ylabel = '노쇼비율 (%)',
            title = '월별 노쇼비율', legend = 0)
plt.show()
```



07. 월별No-Show (합계)

```
month_analysis = df['No-show'].eq('Yes').groupby(df['AppointmentMonth']).sum().reset_index(name = 'proportion_by_month')
month_analysis.plot.bar(x = 'AppointmentMonth', y = 'proportion_by_month', rot = 0,
            xlabel = '노쇼한 달', ylabel = '노쇼 수',
            title = '월별 노쇼 수', legend = 0)
plt.show()
```



모든 언어 ▾ 2016년 5월 1일 – 2016년 5월 31일 ▾ 모든 결과 ▾ 초기화

브라질 축구 국가대표팀
월드컵 예선 1위

경기 | 뉴스 | 순위 | 선수

코파 아메리카 · 조별 리그 · 경기일(2/5)

		풀타임 6. 18.
브라질	4 ◂	▶ 3:14
페루	0	

코파 아메리카 · 조별 리그 · 경기일(4/5)

		풀타임 6. 24.
브라질	2 ◂	▶ 3:18
콜롬비아	1	

코파 아메리카 · 조별 리그 · 경기일(5/5)

		풀타임 6. 28.
브라질	1	▶ 3:16
에콰도르	1	

코파 아메리카 · 8강전

		풀타임 7. 3.
브라질	1 ◂	▶ 3:10
칠레	0	

코파 아메리카 · 4강전

		풀타임 7. 6. (화)
브라질	1 ◂	▶ 3:13
페루	0	

코파 아메리카 · 결승전

		풀타임 7. 11. (일)
아르헨티나	1 ◂	▶ 3:28
브라질	0	

08. 요일별No-Show (Categorical 사용 정렬)

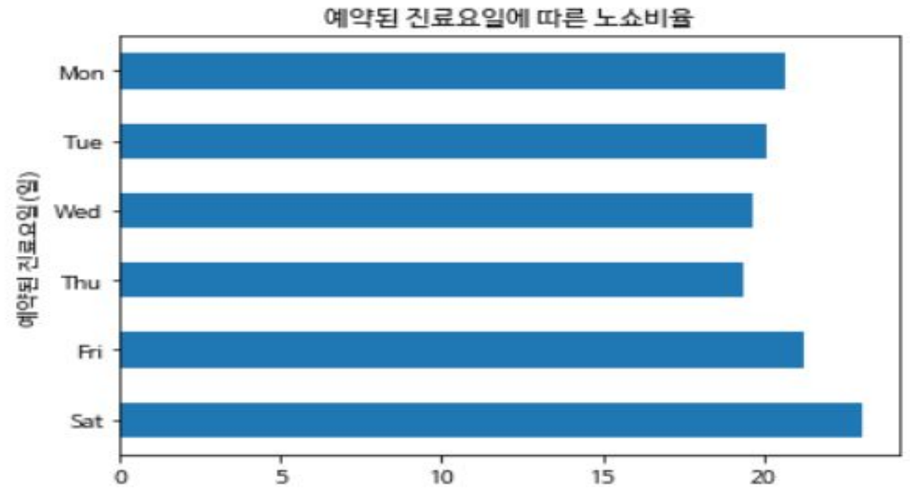
```
df['진료요일'] = df['AppointmentYMD'].dt.dayofweek #0: monday/ 6:sunday
```

```
df['진료요일'].replace(0, "Mon", inplace = True)
df['진료요일'].replace(1, "Tue", inplace = True)
df['진료요일'].replace(2, "Wed", inplace = True)
df['진료요일'].replace(3, "Thu", inplace = True)
df['진료요일'].replace(4, "Fri", inplace = True)
df['진료요일'].replace(5, "Sat", inplace = True)
```

	진료요일	proportion_by_weekday
2	Sat	0.230769
0	Fri	0.212261
3	Thu	0.193494
5	Wed	0.196861
4	Tue	0.200874
1	Mon	0.206446

```
weekday_analysis = df['No-show'].eq('Yes').groupby(df['진료요일']).mean().reset_index(name = 'proportion_by_weekday')
```

```
weekday_analysis['진료요일'] = pd.Categorical(weekday_analysis['진료요일'], ["Mon","Tue","Wed","Thu","Fri","Sat"])
weekday_analysis = weekday_analysis.sort_values("진료요일", ascending = False )
```



09. 상습No-Show 고객 (5번 이상 No-show)

```
df_ss.info() #110521 PatientId
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 110521 entries, 0 to 110520
Data columns (total 27 columns):
 #   Column           Non-Null Count   Dtype  
---  ------           --------------   -----  
 0   PatientId        110521 non-null  float64
 1   Gender           110521 non-null  object 
 2   Age              110521 non-null  int64  
 3   Neighbourhood    110521 non-null  object 
 4   Scholarship      110521 non-null  int64  
 5   Hipertension     110521 non-null  int64  
 6   Diabetes         110521 non-null  int64  
 7   Alcoholism       110521 non-null  int64  
 8   Handcap          110521 non-null  int64  
 9   SMS_received     110521 non-null  int64  
 10  No-show          110521 non-null  object 
 11  ScheduledYear    110521 non-null  int64  
 12  ScheduledMonth   110521 non-null  int64  
 13  ScheduledDay     110521 non-null  int64  
 14  ScheduledHour    110521 non-null  int64  
 15  ScheduledMinute  110521 non-null  int64  
 16  AppointmentYear  110521 non-null  int64  
 17  AppointmentMonth 110521 non-null  int64  
 18  AppointmentDay   110521 non-null  int64  
```

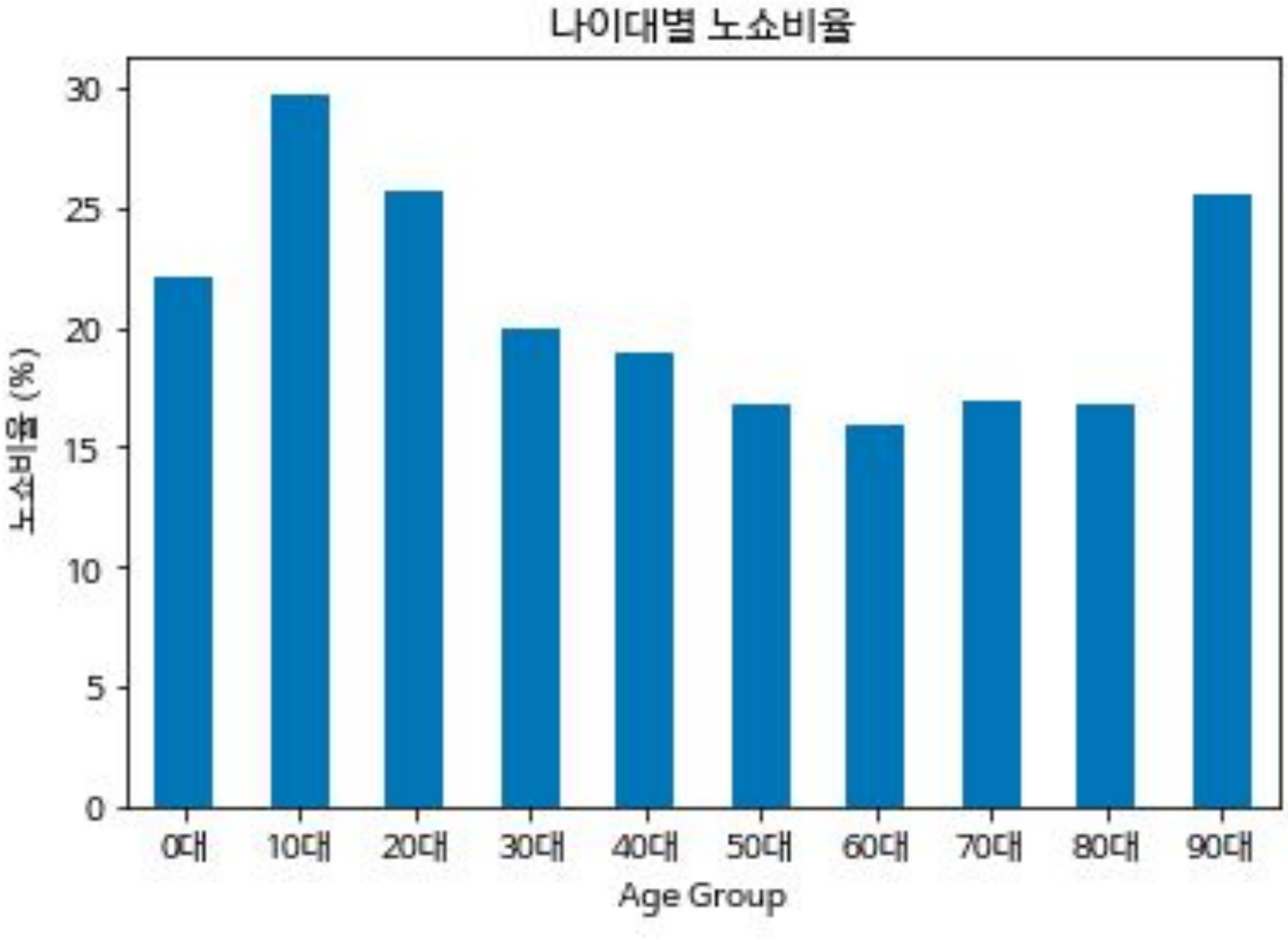
```
len(df_ss['PatientId'].unique()) #62298 unique, meaning about 50000 did not show up more than one time.
```

```
PatientId_counts = df_ss['PatientId'].value_counts()

# Select the values where the count is less than 5 (or 5 if you like)
to_remove = PatientId_counts[PatientId_counts < 2].index

# Keep rows where the city column is not in to_remove
df_ss = df_ss[~df_ss.PatientId.isin(to_remove)]
len(df_ss['PatientId'].unique())
#방법2: df_ss = df_ss.groupby('PatientId').filter(lambda x : len(x)>=5)
```

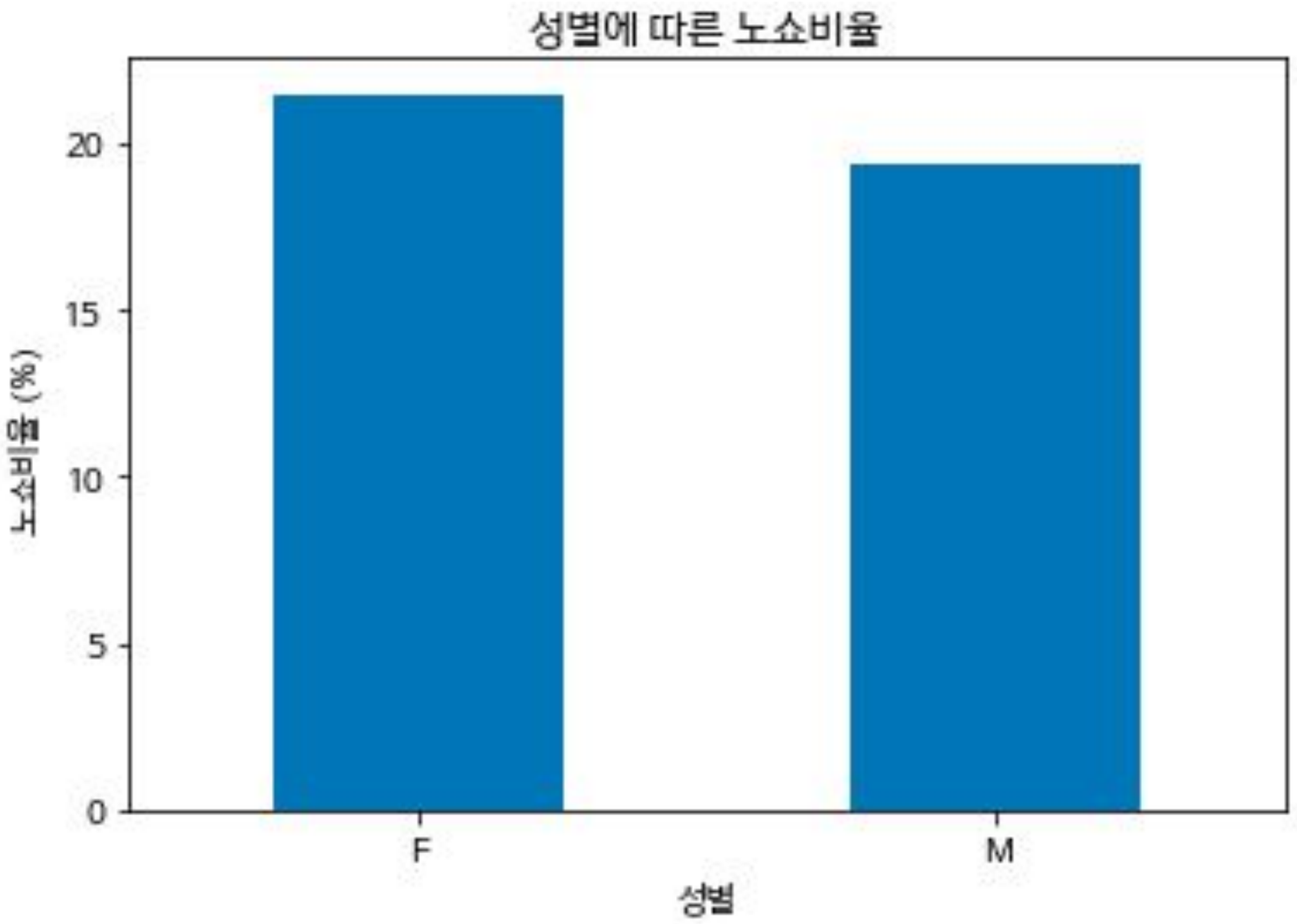
```
len(df_ss['PatientId'].unique())
```

```
2615
```

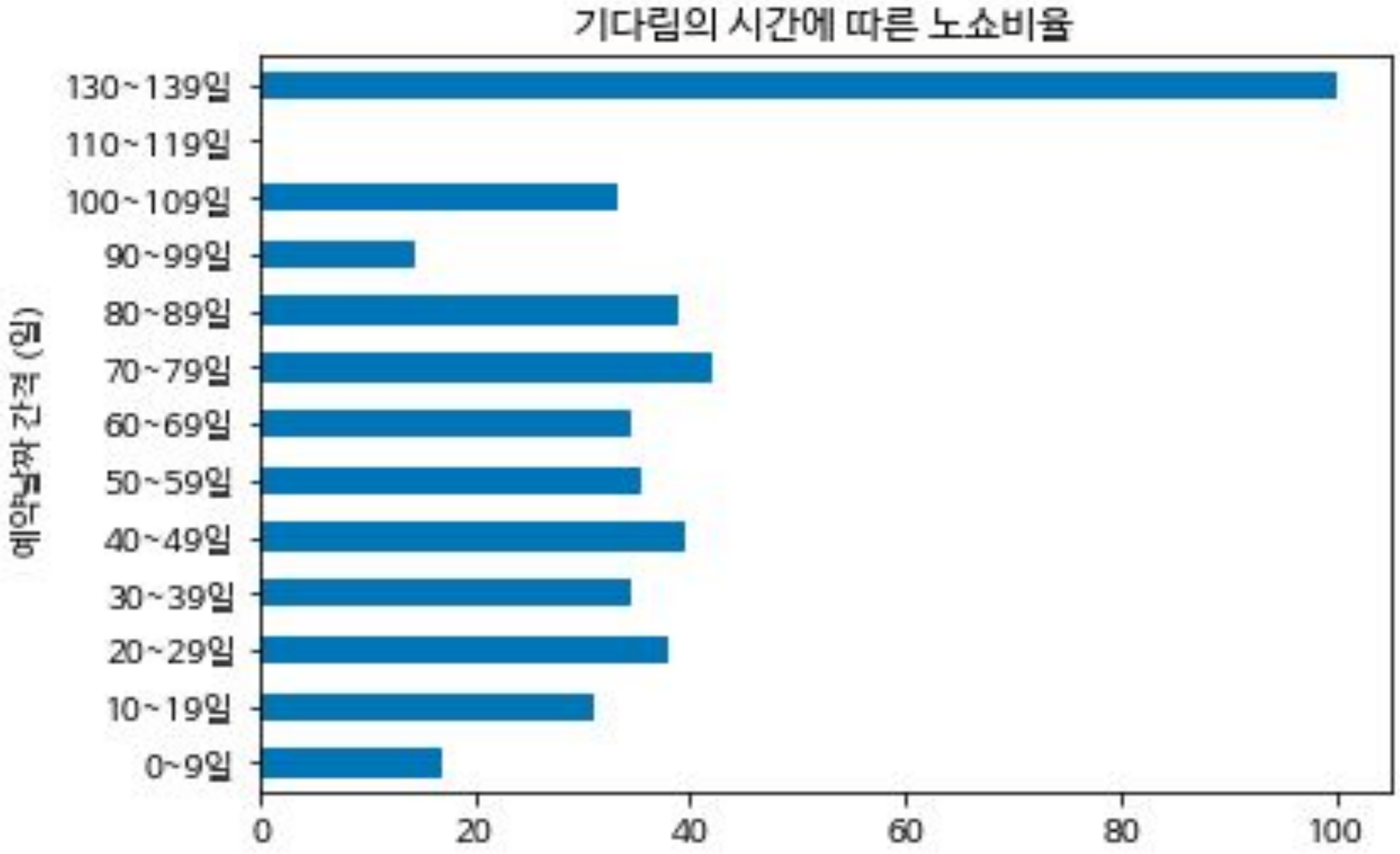
09. 상습No-Show 고객 *



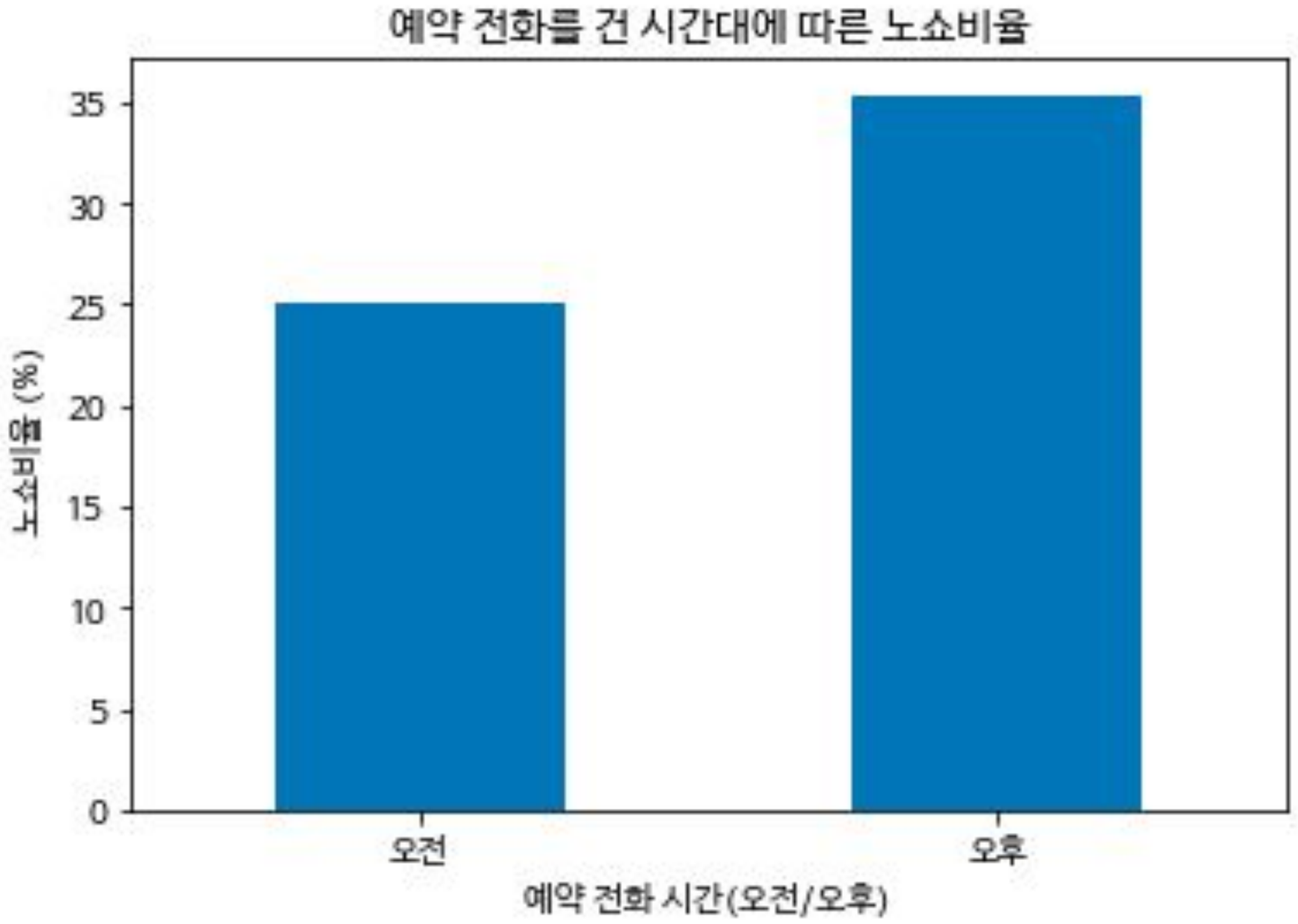
09. 상습No-Show 고객



09. 상습No-Show 고객

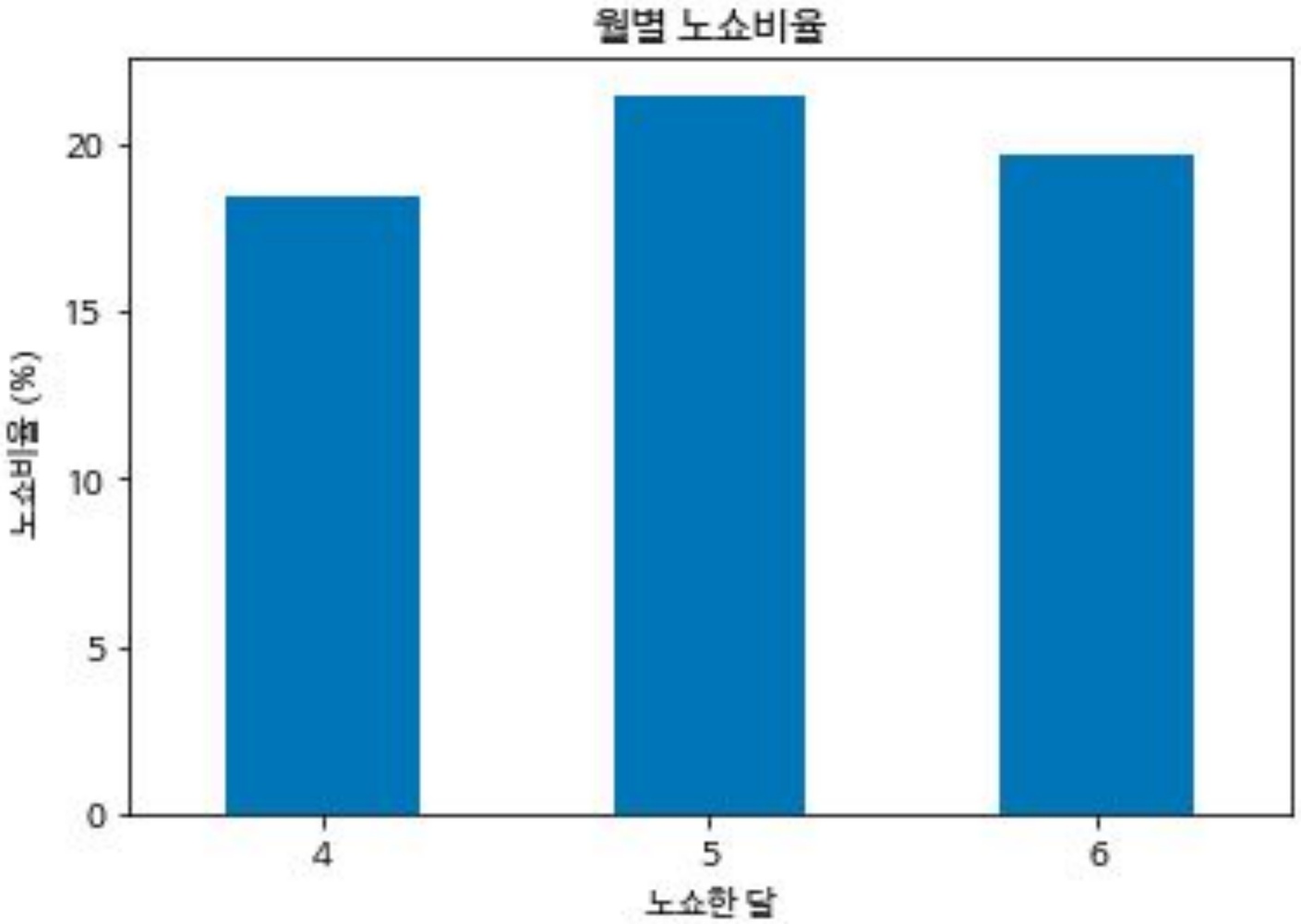


09. 상습No-Show 고객

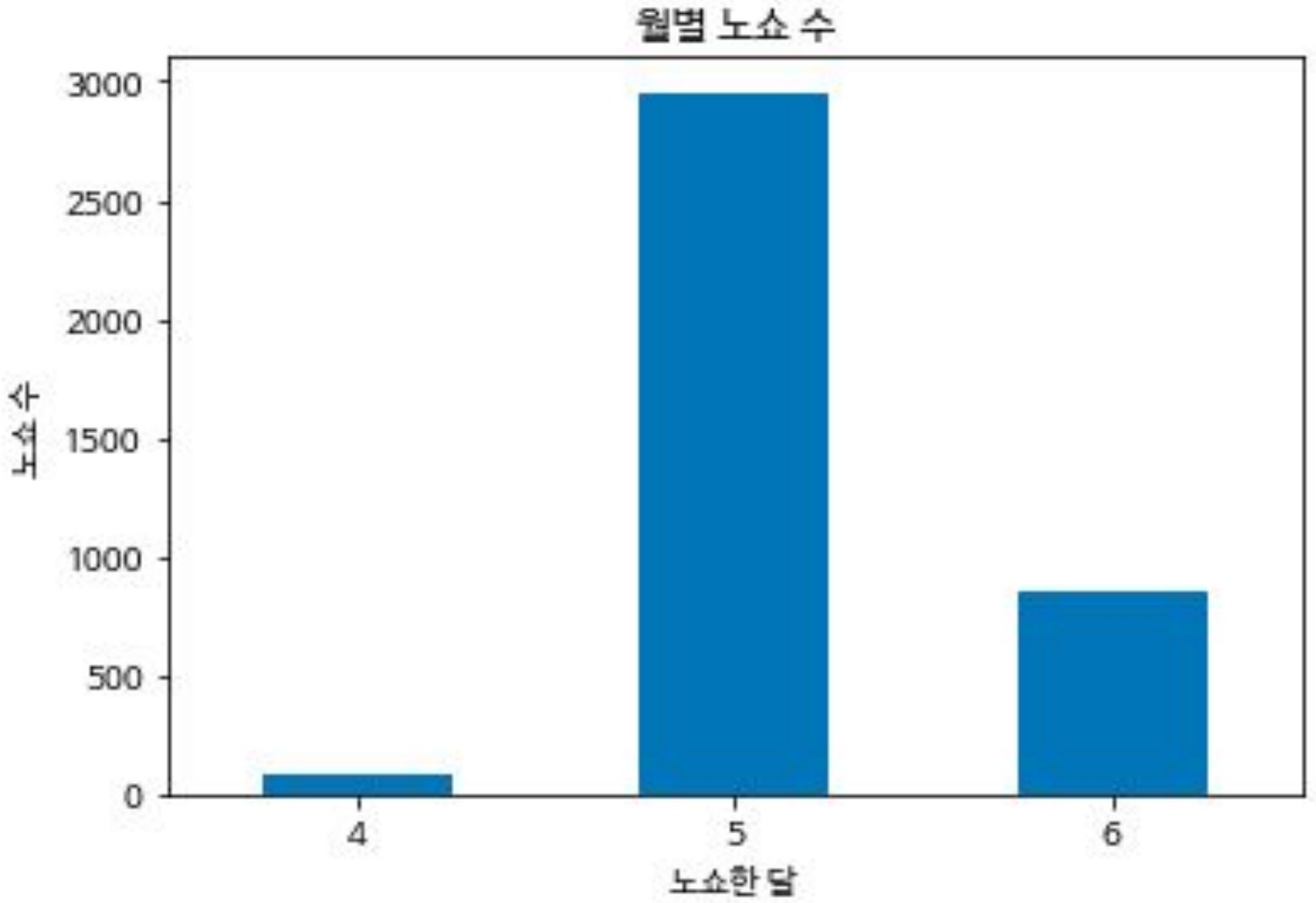


09. 상습No-Show 고객

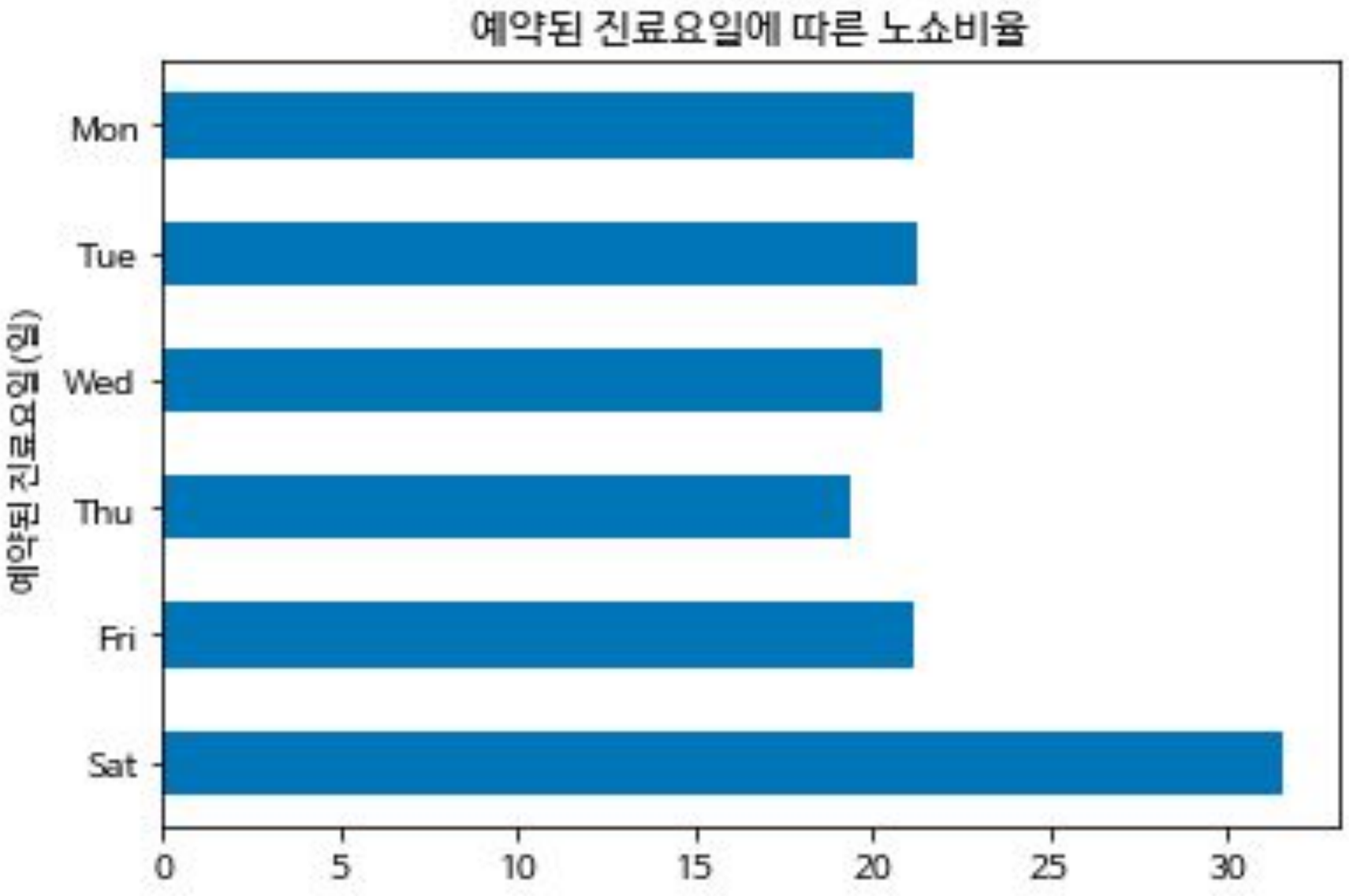
09. 상습No-Show 고객



09. 상습No-Show 고객



09. 상습No-Show 고객 *



01~09. 결론

- 나이가 젊을수록
- 예약 전화 건 날짜와 실제 진료 날짜 사이가 길수록
- 예약 전화 건 시간대가 늦을수록 (오후>오전)
- 인기행사 유무 (예. 5월 스포츠 행사)
- 토요일

10. Machine Learning 을 이용한 회귀분석: 데이터

	Age	No-show	M	1	오후	Mon	Sat	Thu	Tue	Wed
0	62	0	0	0	1	0	0	0	0	0
1	56	0	1	0	1	0	0	0	0	0
2	62	0	0	0	1	0	0	0	0	0
3	8	0	0	0	1	0	0	0	0	0
4	56	0	0	0	1	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...
110521	56	0	0	1	0	0	0	0	1	0
110522	51	0	0	1	0	0	0	0	1	0
110523	21	0	0	1	1	0	0	0	1	0
110524	38	0	0	1	1	0	0	0	1	0
110525	54	0	0	1	1	0	0	0	1	0

110526 rows × 10 columns

독립변수:
Age
Gender: Dummy Variable (0:F, 1:M)
SMS-received: Dummy Variable (0, 1)
오후: Dummy Variable (0:오전, 1:오후)
요일: Dummy Variable (0, 1:Friday)

종속변수:
No-show: 0, 1

10. Machine Learning 을 이용한 회귀분석: 모델선택

머신러닝에서 2진 분류(Binary Classification) 모델로 사용되는 로지스틱 회귀 알고리즘

로지스틱 회귀(Logistic Regression): 회귀를 사용하여 데이터가 어떤 범주에 속할 확률을 0에서 1 사이의 값으로 예측하고 그 확률에 따라 가능성이 더 높은 범주에 속하는 것으로 분류해주는 지도 학습 알고리즘

ex. 스팸 메일 분류기 어떤 메일을 받았을 때 그것이 스팸일 확률이 0.5 이상이면 spam으로 분류하고, 확률이 0.5보다 작은 경우 ham으로 분류. 이렇게 데이터가 2개의 범주 중 하나에 속하도록 결정하는 것을 2진 분류(binary classification)라고 한다.

출처: http://hleecaster.com/ml-logistic-regression-concept/

10. Machine Learning 을 이용한 회귀분석

[176]
```
#test train split

from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(train1.drop('No-show',axis=1),
                                                    train1['No-show'], test_size=0.30,
                                                    random_state=100)
```

```
# training and predicting

from sklearn.linear_model import LogisticRegression
logmodel = LogisticRegression()
logmodel.fit(x_train,y_train)
predictions = logmodel.predict(x_test)
```

10. Machine Learning 을 이용한 회귀분석: Evaluation

```
[180] y_pred = logreg.predict(X_test)
      print('Accuracy of logistic regression classifier on test set: {:.2f}'.format(logreg.score(X_test, y_test)))
```

```
Accuracy of logistic regression classifier on test set: 0.80
```

```
[181] from sklearn.metrics import confusion_matrix
      confusion_matrix = confusion_matrix(y_test, y_pred)
      print(confusion_matrix)
```

```
[[26565     0]
 [ 6593     0]]
```

```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.80      1.00      0.89     26565
           1       0.00      0.00      0.00      6593

    accuracy                           0.80     33158
   macro avg       0.40      0.50      0.44     33158
weighted avg       0.64      0.80      0.71     33158
```

- **Accuracy**는 올바르게 예측된 데이터의 수를 전체 데이터의 수로 나눈 값
- **Recall**은 실제로 **True**인 데이터를 모델이 **True**라고 인식한 데이터의 수
- **Precision**은 모델이 **True**로 예측한 데이터 중 실제로 **True**인 데이터의 수
- **F1 score**는 **precision** 과 **recall**의 조화평균

10. Machine Learning 을 이용한 회귀분석

```
import statsmodels.api as sm
logSm = sm.Logit(train1['No-show'], train1.drop('No-show',axis=1))
result = logSm.fit()
result.summary2()
```

```
Optimization terminated successfully.
         Current function value: 0.506984
         Iterations 6
```

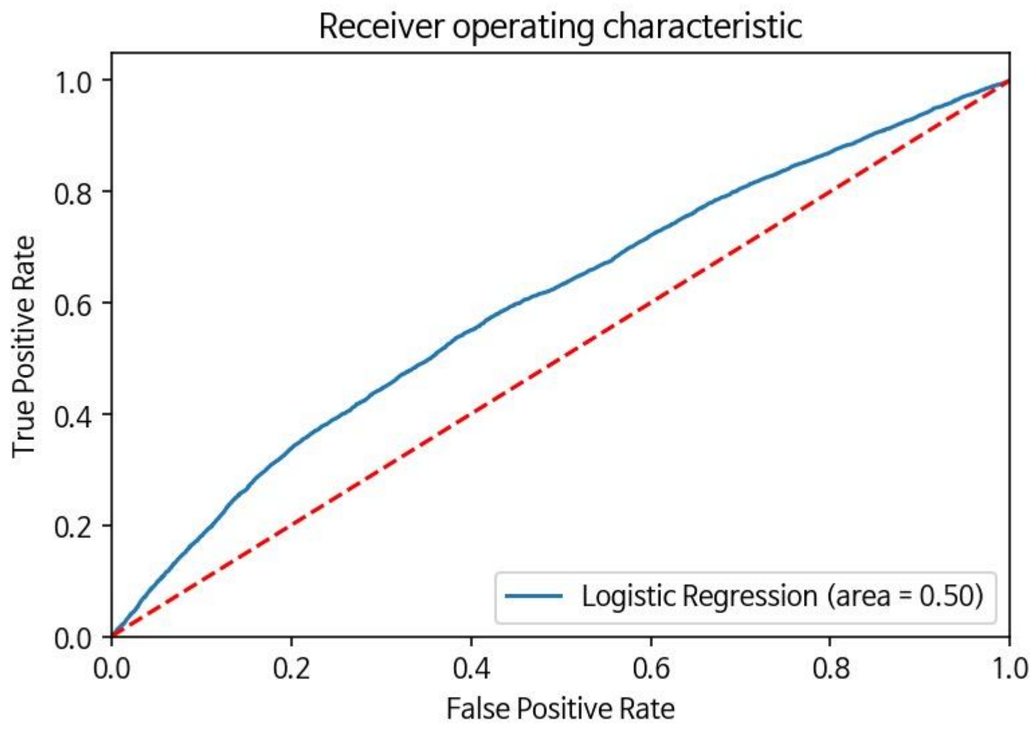
Model:	Logit	Pseudo R-squared:	-0.008
Dependent Variable:	No-show	AIC:	112087.8994
Date:	2021-07-19 06:09	BIC:	112174.4164
No. Observations:	110526	Log-Likelihood:	-56035.
Df Model:	8	LL-Null:	-55603.
Df Residuals:	110517	LLR p-value:	1.0000
Converged:	1.0000	Scale:	1.0000
No. Iterations:	6.0000		

	Coef.	Std.Err.	z	P>|z|	[0.025	0.975]
Age	-0.0175	0.0003	-63.0135	0.0000	-0.0181	-0.0170
M	-0.3232	0.0151	-21.3981	0.0000	-0.3528	-0.2936
1	0.5102	0.0154	33.0485	0.0000	0.4800	0.5405
오후	-0.0143	0.0153	-0.9362	0.3492	-0.0443	0.0157
Mon	-0.8083	0.0198	-40.7745	0.0000	-0.8471	-0.7694
Sat	-0.4060	0.3881	-1.0462	0.2955	-1.1666	0.3546
Thu	-0.8689	0.0223	-38.9799	0.0000	-0.9126	-0.8252
Tue	-0.8868	0.0195	-45.4277	0.0000	-0.9251	-0.8486
Wed	-0.8839	0.0193	-45.8017	0.0000	-0.9217	-0.8461

p-value < 0.05 를 살펴보면,

나이가 많을수록 노쇼할 확률이 **0.0175**만큼 낮아진다.
SMS를 받은 사람은 안받은 사람에 비해서 **0.5102** 만큼 노쇼할 확률이 높아진다.
남자는 여자에 비해서 **0.3232**만큼 노쇼할 확률이 낮아진다.
월요일예약은 금요일예약에 비해서 **0.8083**만큼 노쇼할 확률이 낮아진다.
화요일예약은 금요일예약에 비해서 **0.8868**만큼 노쇼할 확률이 낮아진다.
수요일예약은 금요일예약에 비해서 **0.8839**만큼 노쇼할 확률이 낮아진다.
목요일예약은 금요일예약에 비해서 **0.8689**만큼 노쇼할 확률이 낮아진다.

10. Machine Learning 을 이용한 회귀분석: ROC Curve



ROC Curve의 X축

$$False\ Positive\ Rate = \frac{\sum False\ Positive\ 수}{\sum 정답의\ Negative\ 수}$$

Total Population	모델의 예측 Positive	모델의 예측 Negative
정답이 Positive	True Positive	False Negative
정답이 Negative	False Positive	True Negative

SW

ROC Curve의 Y축

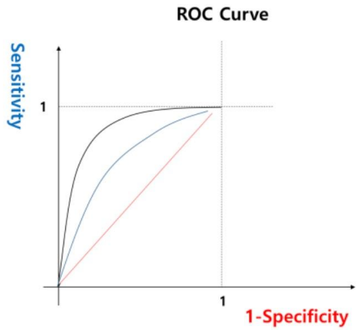
$$True\ Positive\ Rate = \frac{\sum True\ Positive\ 수}{\sum 정답의\ Positive\ 수}$$

Total Population	모델의 예측 Positive	모델의 예측 Negative
정답이 Positive	True Positive	False Negative
정답이 Negative	False Positive	True Negative

SW

그러면 ROC curve에 이 개념을 넣어보자.

x축(1 - specificity = FPR(False Positive Rate))은 가짜 중에 진짜를 찾은 비율(가짜 중에 잘못 예측한 비율)이고, y축(Sensitivity)이 의미하는 바는 진짜 중에 진짜를 찾은 비율(진짜 중에 진짜를 잘 찾은 비율)이 된다. 진짜로 예측한 값들 중에서 실제로도 진짜일 경우가 실제로 가짜일 경우보다 높아야 AUC가 높게 나온다.



출처: https://m.blog.naver.com/PostView.naver?isHttpsRedirect=true&blogId=sw4r&logNo=221015817276

감사합니다.