Acne and External Environment

여드름과 외부환경의 상관관계 분석과 예측

파이썬과 R을 활용한 빅데이터 머신러닝 전문가 양성과정
11반 김예찬

CONTENTS

01

프로젝트 목표

‘외부환경에서 발생 가능성이 있는지 확인 및 예측하고
활용방안에 대해서 고민해보자’

여드름

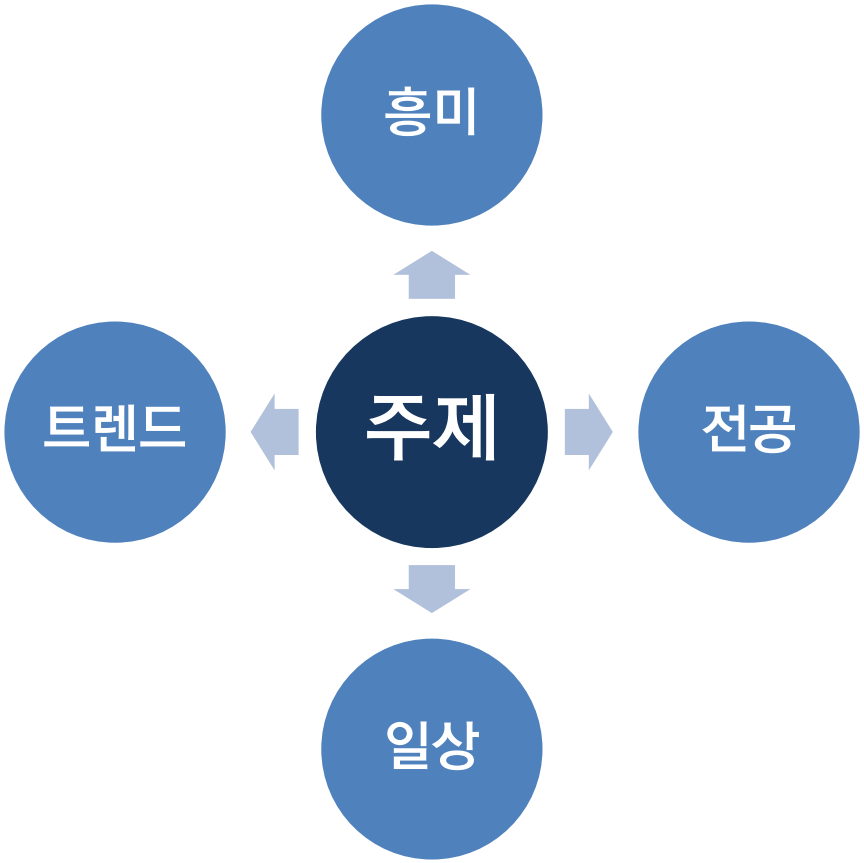
- 건강보험심사평가원(심평원) 제공
- 질병코드 등 다양한 코드
- 질병 종류별 많은 코드 존재
- 복잡한 데이터 구조 및 명칭

외부환경

- 기상청 오픈 API 활용
- 2016년 ~ 2020년 2월 데이터
- 평균기온, 강수량, 자외선, 미세먼지
- 네이버, 검색 트렌드 및 지식in API
- 제한적인 데이터 수집

01

“ WHY 여드름?”



나와 관련 있는 주제

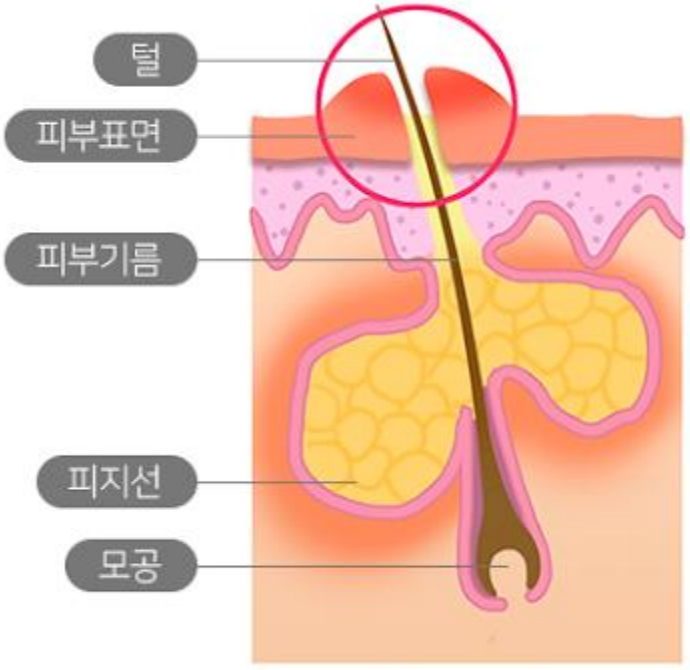
공감할 수 있는 주제

관심있는 주제

01

“ 공공의 적 ”

여드름이란?



여드름(Acne)

- 주로 상체에 발생하는 염증성 피부질환
- 남성호르몬이 발생 요인
- 피지와 Acne균이 만나서 발생
- 다양한 증상 존재
- 재발 가능성이 매우 높은 질환

01

“ 공공의 적 ”

청소년의 상징?

사랑하면 생긴다?

젊다는 증거?

01

“ 공공의 적 ”

육체적인 문제

정신적인 문제

01

“ 공공의 적 ”

육체적인 문제

흉터

모공 확대

붉은 기

고통

01

“ 공공의 적 ”

정신적인 문제

외모 자신감 하락

스트레스

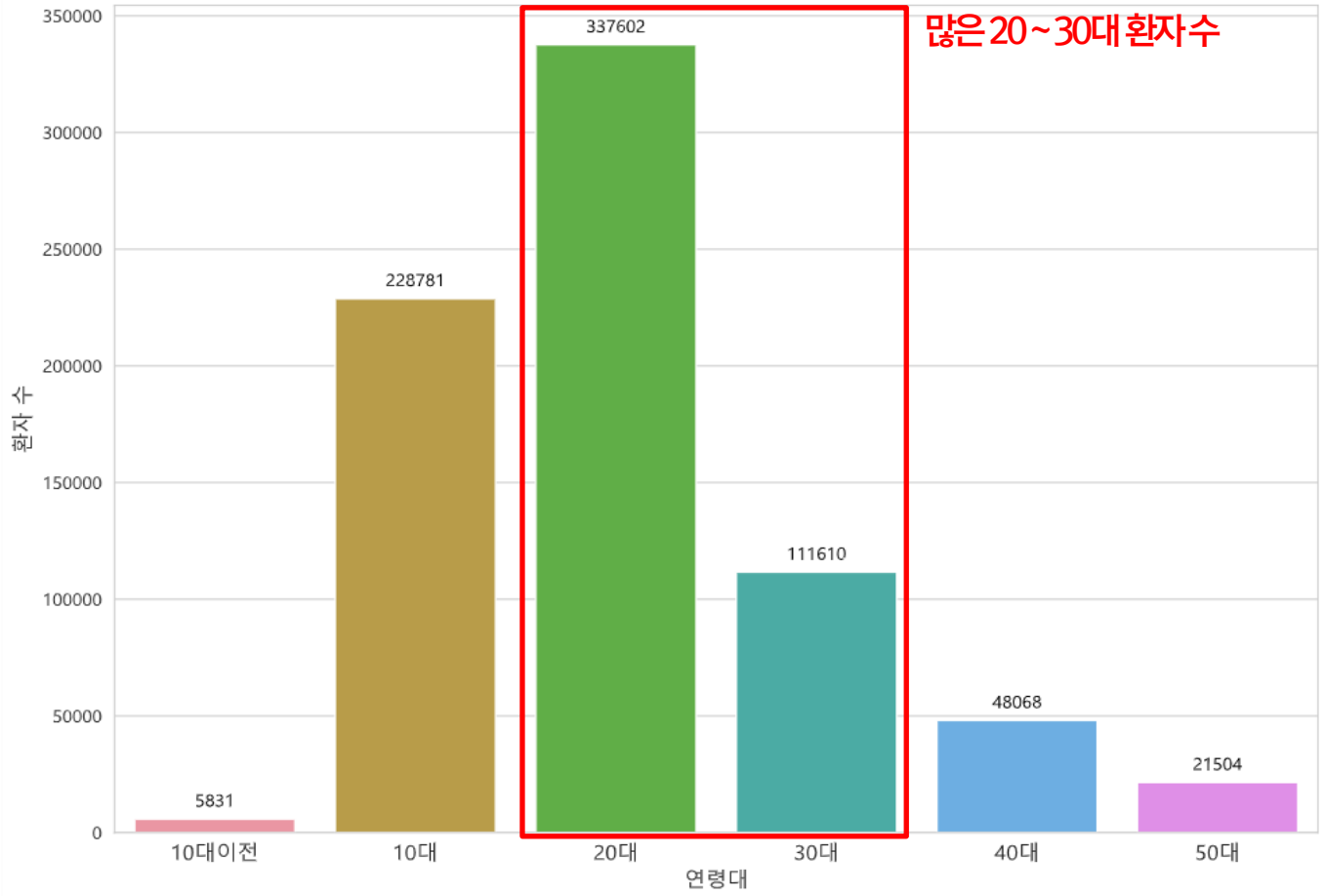
신체이형 장애

비싼 치료비

01

“청소년의 상징?”

2016년 1월~2020년 2월

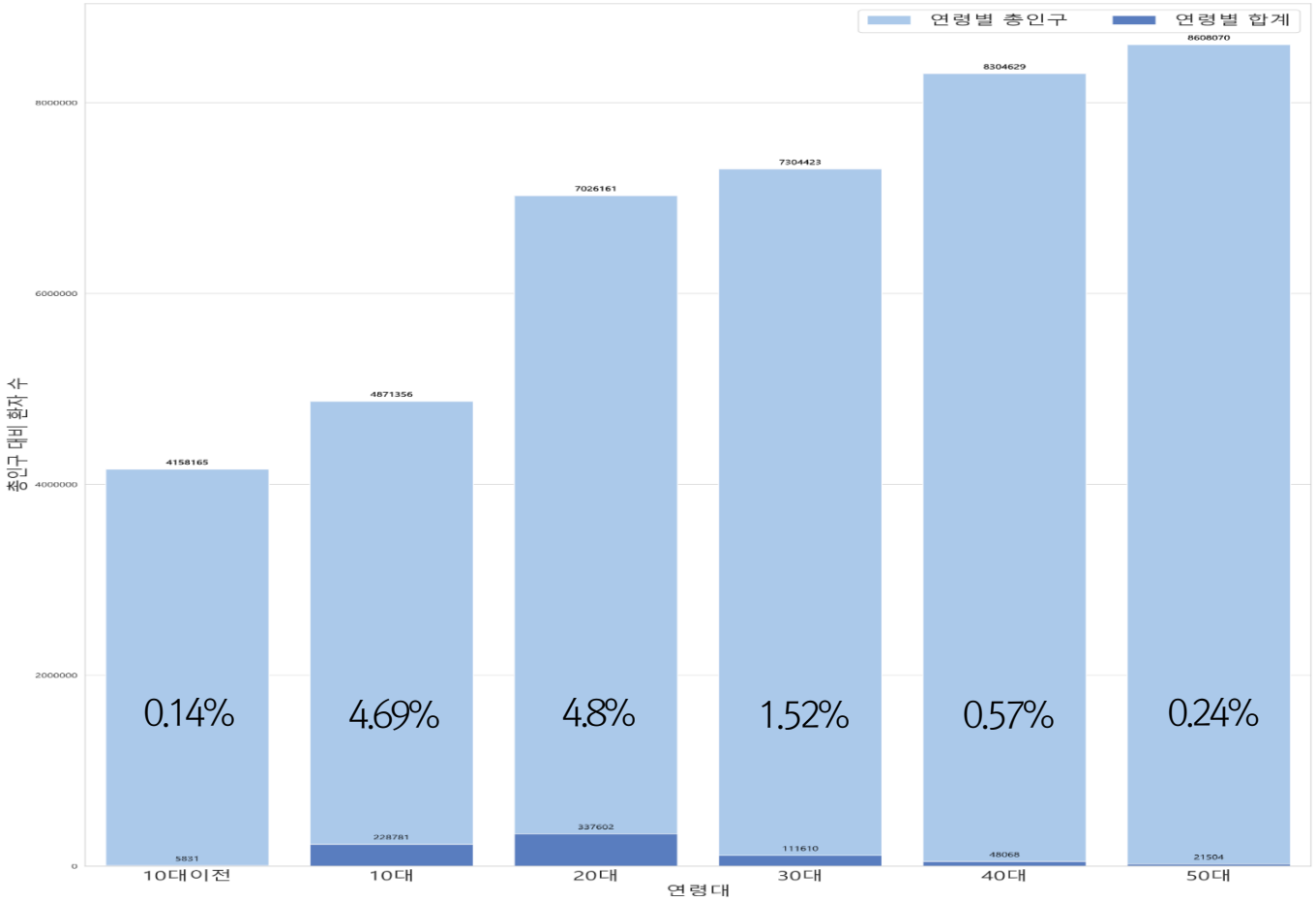


건강보험심사평가원, 여드름 관련 질병코드 종합

01

“청소년의 상징?”

2016년 1월~2020년 2월

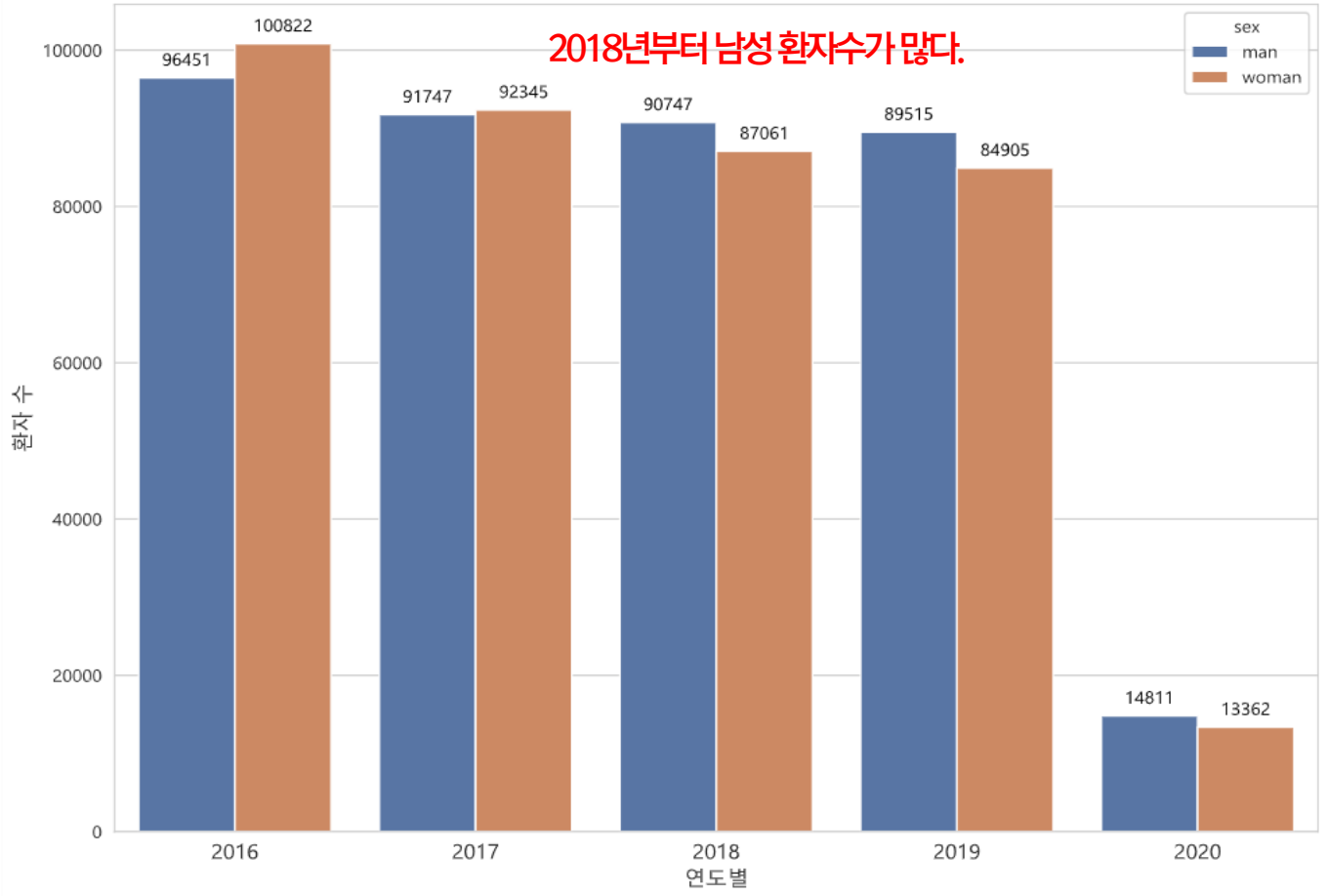


건강보험심사평가원, 여드름 관련 질병코드 종합

01

“ 성별의 차이? ”

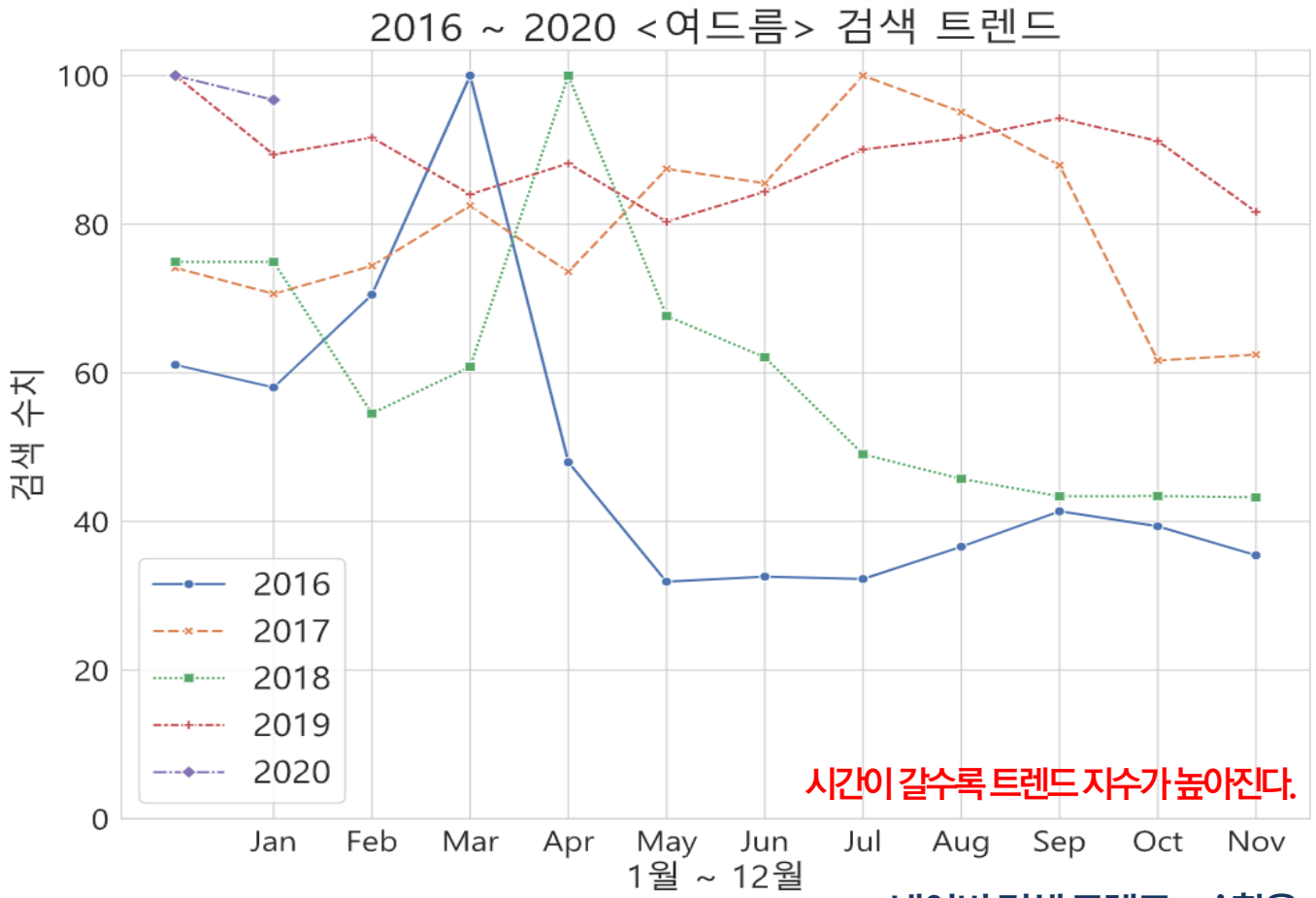
2016년 1월~2020년 2월



건강보험심사평가원, 여드름 관련 질병코드 종합

01

“ 공공의적 ”

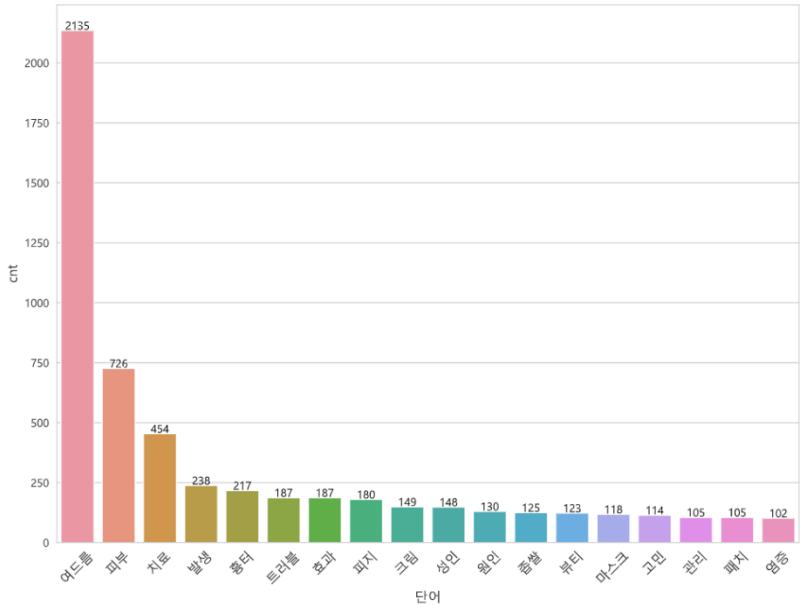


네이버 검색 트렌드 api 활용

01

“ 공공의 적 ”

네이버 뉴스기사 1000개 대상 명사 추출
네이버 API 활용

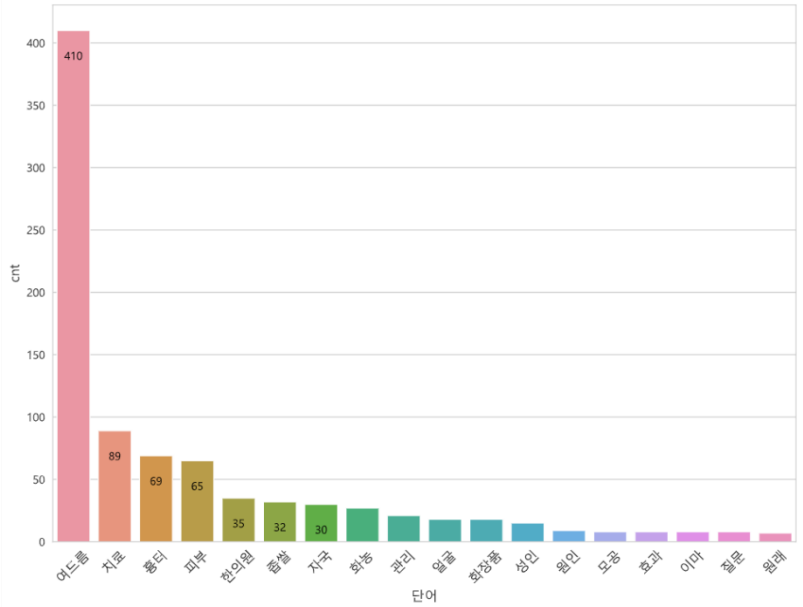


상위 빈도수 18개 추출

01

“ 공공의적 ”

네이버 지식in1000개 글 대상 명사 추출
네이버 API 활용



상위 빈도수 18개 추출

02

“ 데이터 소개 ”

독립변수

- temp : 월 평균기온 데이터
- pm2.5 : 월 평균 미세먼지 데이터
- rain : 월 평균 강수량 데이터
- uv : 월 평균 자외선 데이터

종속변수

- 월별 환자 수 데이터
- 성별, 연령별 개별 데이터

02

“ 데이터 소개 ”

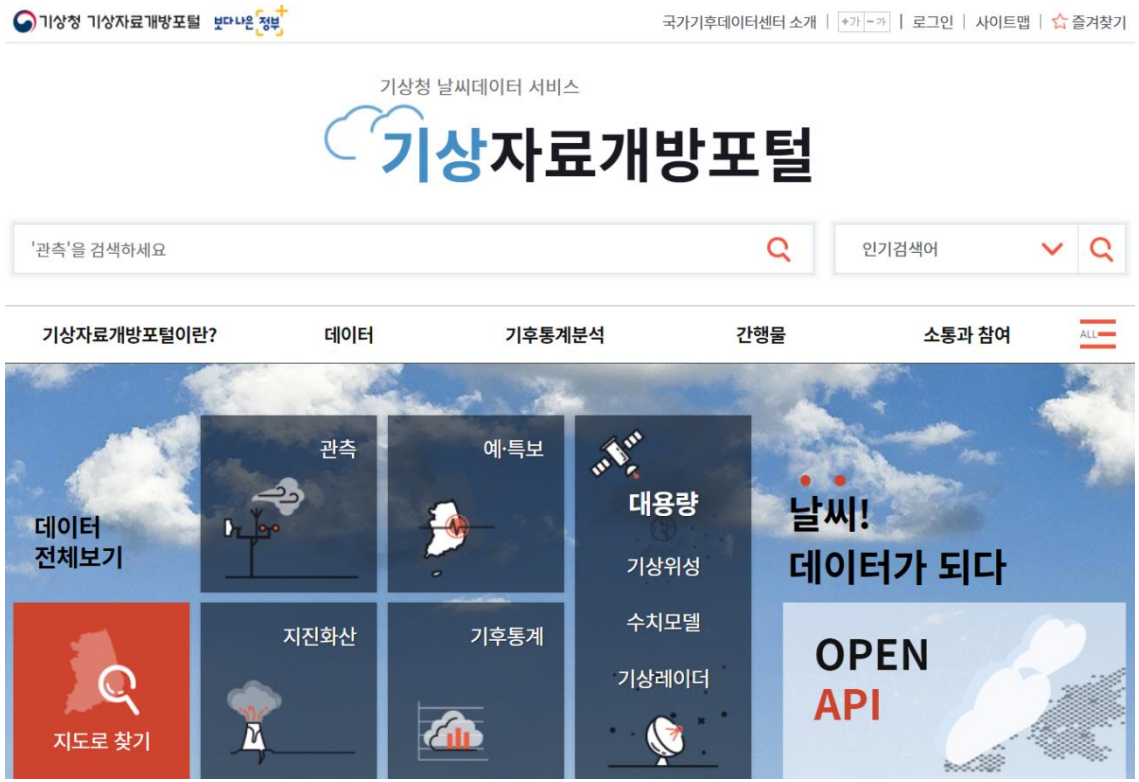
- 종속변수

보건의료빅데이터개방시스템 : https://opendata.hira.or.kr/home.do

02

“ 데이터 소개 ”

- 독립변수

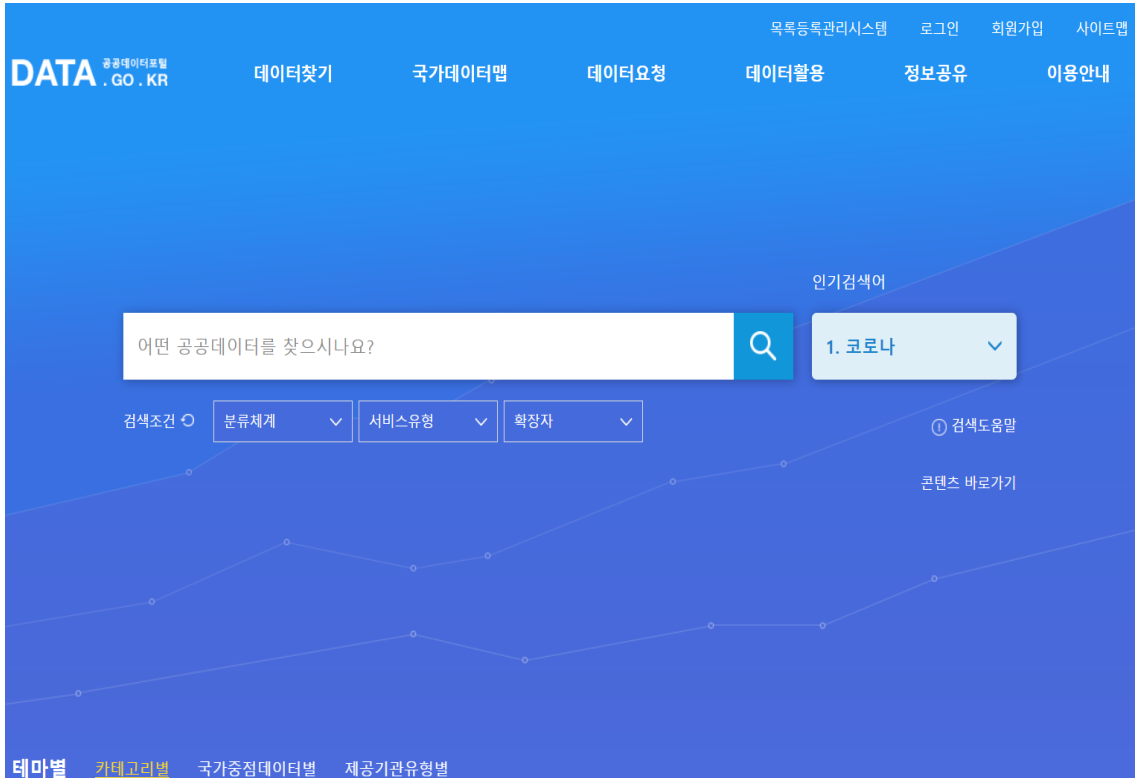


기상자료개방포털 : https://data.kma.go.kr/cmmn/main.do

02

“ 데이터 소개 ”

– 독립변수



공공데이터포털 : https://www.data.go.kr/index.do

03

“ 분석 방법 ”

03

“상관관계 분석”

– 피어슨, 스피어맨 상관관계 분석

1에 가까울수록 강한양의상관관계
-1에 가까울수록 강한음의상관관계
0에 가까우면 선형적인 상관관계 없음을 의미

1.피어슨

```
corr = data.corr(method='pearson')
corr
```

1.피어슨 결과

	total	temp	pm2_5	rain	uv
total	1.000000	0.230657	-0.375881	0.243334	0.085707
temp	0.230657	1.000000	-0.710151	0.645603	0.904398
pm2_5	-0.375881	-0.710151	1.000000	-0.554194	-0.580772
rain	0.243334	0.645603	-0.554194	1.000000	0.551017
uv	0.085707	0.904398	-0.580772	0.551017	1.000000

2.스피어맨

```
corr = data.corr(method='spearman')
corr
```

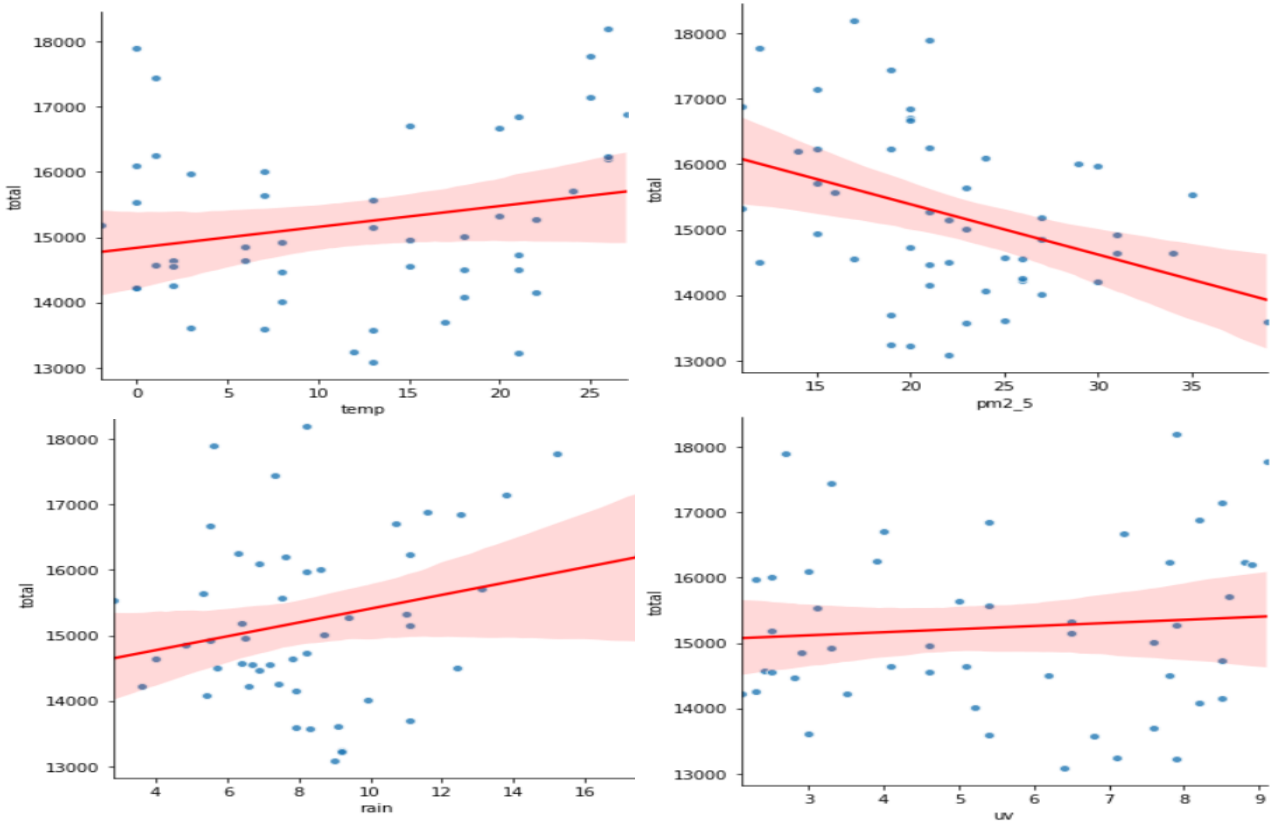
2.스피어맨 결과

	total	temp	pm2_5	rain	uv
total	1.000000	0.220622	-0.386648	0.134790	0.111210
temp	0.220622	1.000000	-0.730235	0.644907	0.881629
pm2_5	-0.386648	-0.730235	1.000000	-0.546422	-0.628217
rain	0.134790	0.644907	-0.546422	1.000000	0.528825
uv	0.111210	0.881629	-0.628217	0.528825	1.000000

03

“상관관계 분석”

- 종속변수와 독립변수의 상관관계 분석을 위한 산점표

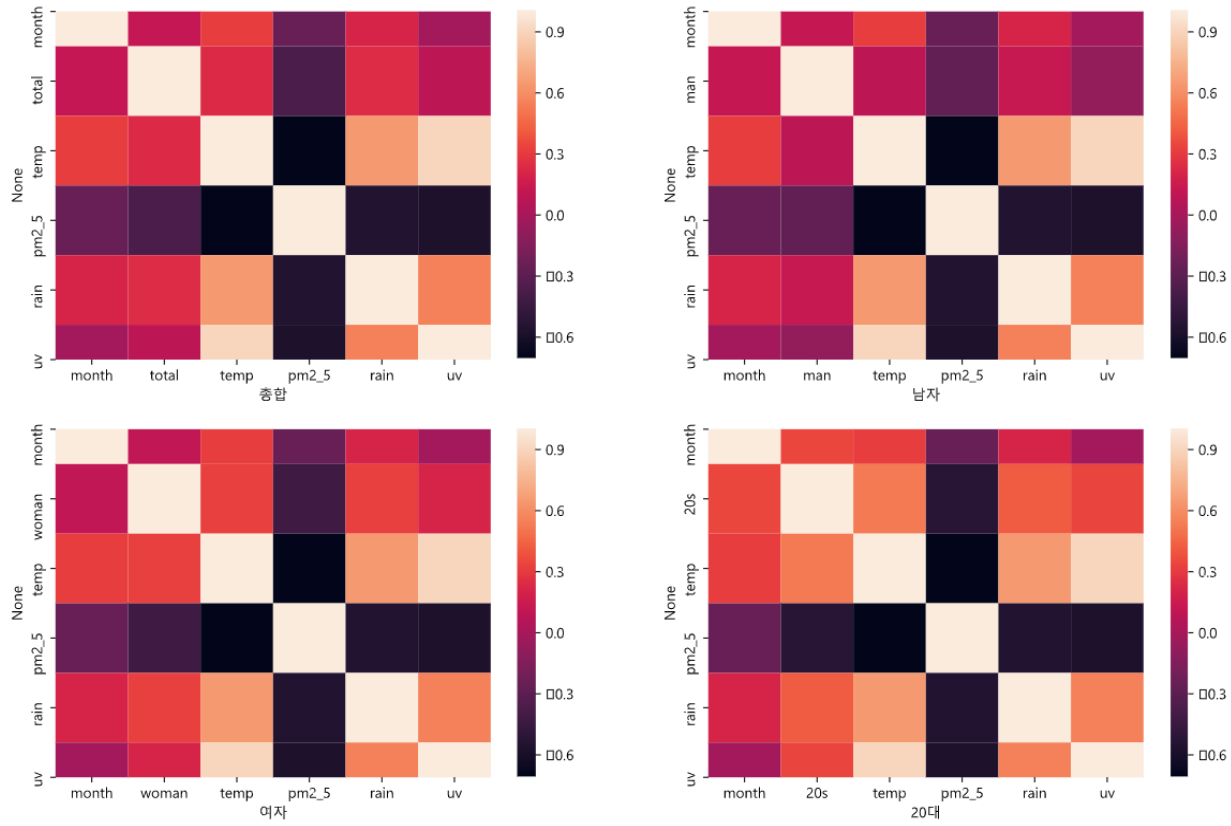


예시) 총 환자 수와 종속변수 산점표

03

“상관관계 분석”

- 종속변수와 상관관계 분석을 위해 총합, 성별(남, 녀), 20대 분류



03

“ 회귀 분석 ”

1) 회귀 분석 in R

```
# 독립변수 : temp, pm2.5, rain, uv
# 종속변수 : total
fit <- lm(total ~ temp + pm2_5 + rain + uv, data= ad)
summary(fit)

# 독립변수 : temp, pm2.5, rain, uv
# 종속변수 : man
fit1 <- lm(man ~ temp + pm2_5 + rain + uv, data= ad1)
summary(fit1)

# 독립변수 : temp, pm2.5, rain, uv
# 종속변수 : woman
fit2 <- lm(woman ~ temp + pm2_5 + rain + uv, data= ad2)
summary(fit2)

# 독립변수 : temp, pm2.5, rain, uv
# 종속변수 : 10대
fit3 <- lm(X10s ~ temp + pm2.5 + rain + uv, data= ad3)
summary(fit3)

# 독립변수 : temp, pm2.5, rain, uv
# 종속변수 : 20대
fit4 <- lm(X20s ~ temp + pm2_5 + rain + uv, data= ad4)
summary(fit4)

# 독립변수 : temp, pm2.5, rain, uv
# 종속변수 : 30대
fit5 <- lm(X30s ~ temp + pm2_5 + rain + uv, data= ad5)
summary(fit5)
```

```
Call:
lm(formula = total ~ temp + pm2_5 + rain + uv, data = ad)

Residuals:
    Min      1Q  Median      3Q     Max
-1970.3  -796.1    42.2   706.0  2567.3

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 17446.47    1327.15  13.146   <2e-16 ***
temp           59.59      53.47   1.115   0.2710
pm2_5         -70.83      40.13  -1.765   0.0844 .
rain           32.29      77.11   0.419   0.6774
uv           -301.91     177.26  -1.703   0.0954 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’

Residual standard error: 1203 on 45 degrees of freedom
Multiple R-squared:  0.2005,    Adjusted R-squared:  0.1295
F-statistic: 2.822 on 4 and 45 DF,  p-value: 0.03583
```

값이 작을수록 의미가있다.

값이클수록 의미가있다.

값이 작을수록 의미가있다.

03

“ REGRESSION 결과 ”

1) 단순회귀 결과 분석 및 의미 해석

	기온	미세먼지	강수량	자외선
총 환자 수	無	有	無	無
남성	無	有	無	無
여성	有	有	無	無
20대	有	有	有	有
30대	有	有	有	有

※ 결정계수 확인 → 모형의 적합도 확인 → 회귀계수 확인 → t값의 유의확률 확인

03

“ DECISION TREE ”

1) DECISION TREE

```
# 의사결정 나무
from sklearn.metrics import accuracy_score
from sklearn.tree import export_graphviz
from IPython.core.display import Image

X = acne_df3_total.iloc[:,1:]
y = acne_df3_total.iloc[:,0]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=0)
from collections import Counter
Counter(y_train)
Counter(y_test)

acne_model = DecisionTreeClassifier(criterion='entropy', max_depth = 2)
acne_model.fit(X_train, y_train)
acne_pred = acne_model.predict(X_test)
acne_model.score(X_train, y_train)
#정확도 계산
acne_model.score(X_test, y_test)
accuracy_score(y_test, acne_pred)
acne_model.feature_importances_
# 중요도 확인
pd.DataFrame({'feature' : acne_df3_total.iloc[:,1:],
              'importance' : acne_model.feature_importances_})
acne_model.classes_
acne_model.predict([[15234,23,12, 6.8, 4.4]])
```

정확도

```
acne_model.score(X_test, y_test)
```

```
0.2
```

입력변수 별 중요도

	feature	importance
0	(total,)	0.000000
1	(temp,)	0.703093
2	(pm2_5,)	0.000000
3	(rain,)	0.000000
4	(uv,)	0.296907

새로운 입력에 대한 예측값

```
acne_model.predict([[15234,23,12
```

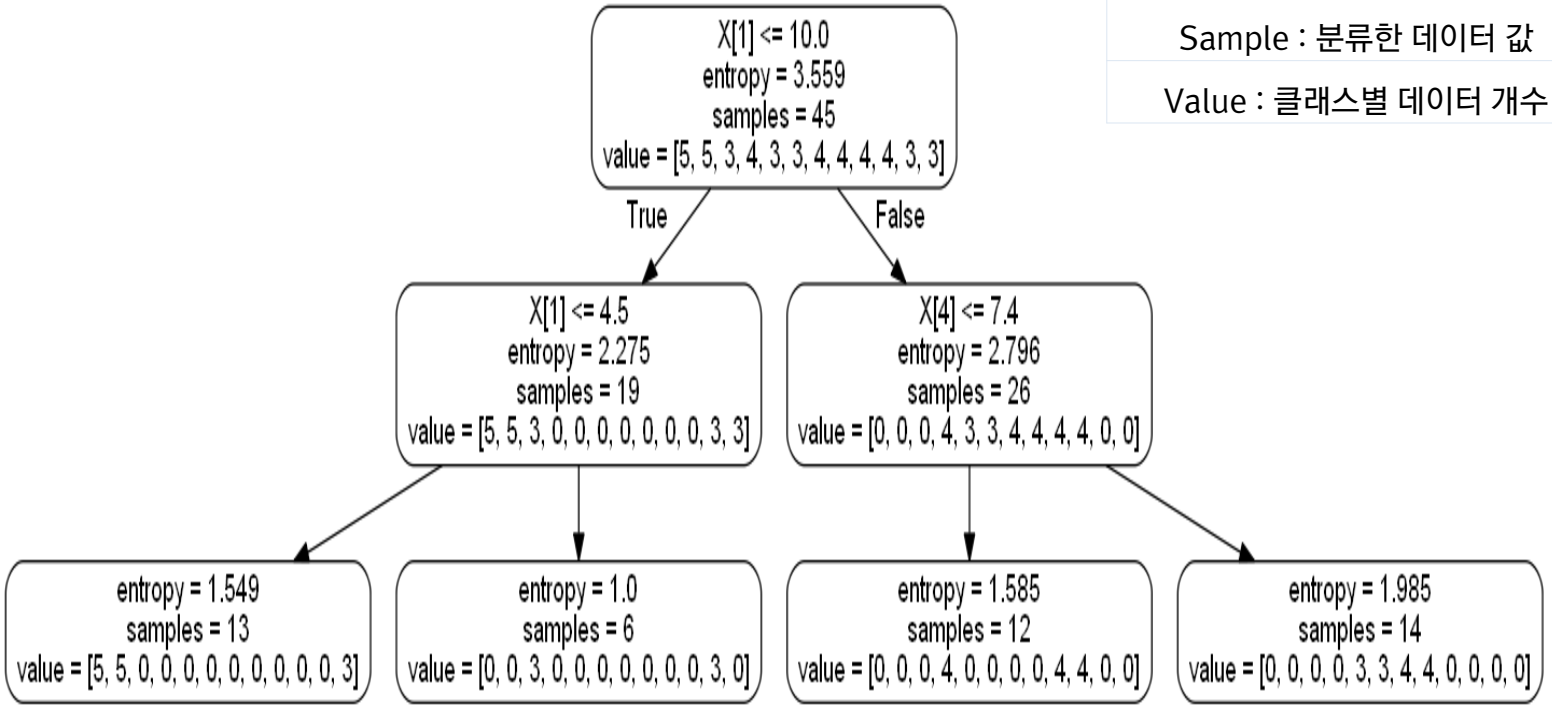
```
array([4], dtype=int64)
```

03

“ DECISION TREE ”

1) DECISION TREE

그래프 해석
첫번째 줄 : 분류 기준
Entropy : 엔트로피 값
Sample : 분류한 데이터 값
Value : 클래스별 데이터 개수



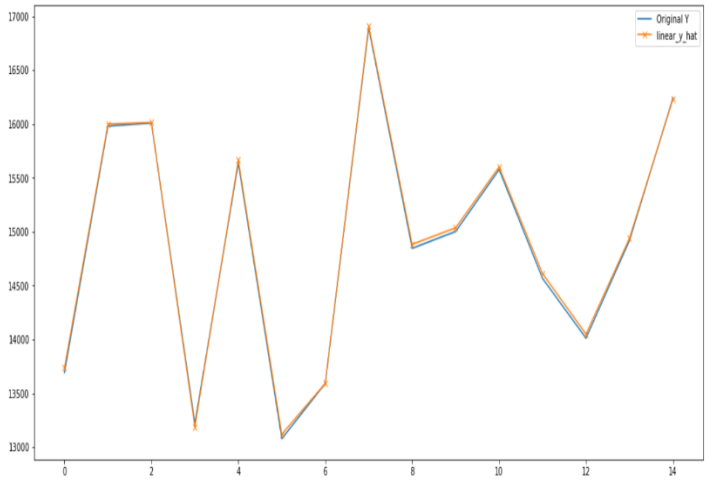
03

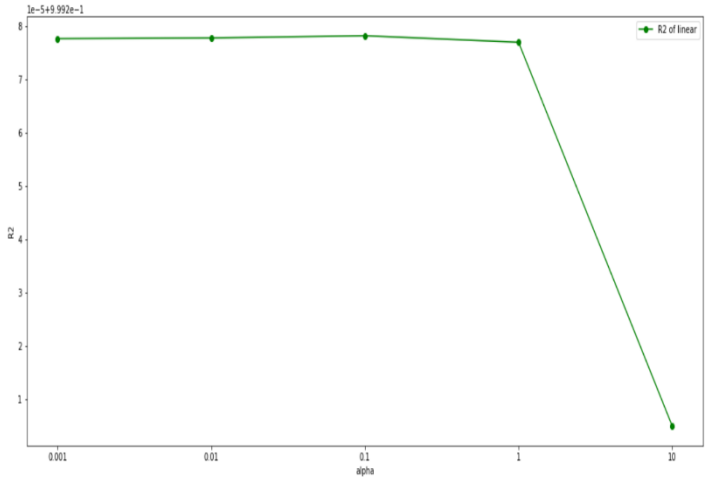
“ LINEAR REGRESSION ”

1) LINEAR REGRESSION

```
X = data.iloc[:,1:]
y = data.iloc[:,0]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
linear = LinearRegression()
linear.fit(X_train,y_train)
linear_y_hat = linear.predict(X_test)
linear_MAE =  mean_absolute_error(y_test,linear_y_hat)

linear_r2 = r2_score(y_test,linear_y_hat)
linear_MSE = mean_squared_error(y_test,linear_y_hat)
linear_MAE = mean_absolute_error(y_test,linear_y_hat)

print('R2 score - Linear: %.2f' %(linear_r2))
print('MSE - Linear: %.2f' %(linear_MSE))
print('MAE - Linear: %.2f' %(linear_MAE))
```

R2 : 1.00 MSE : 931.96 MAE : 27.29





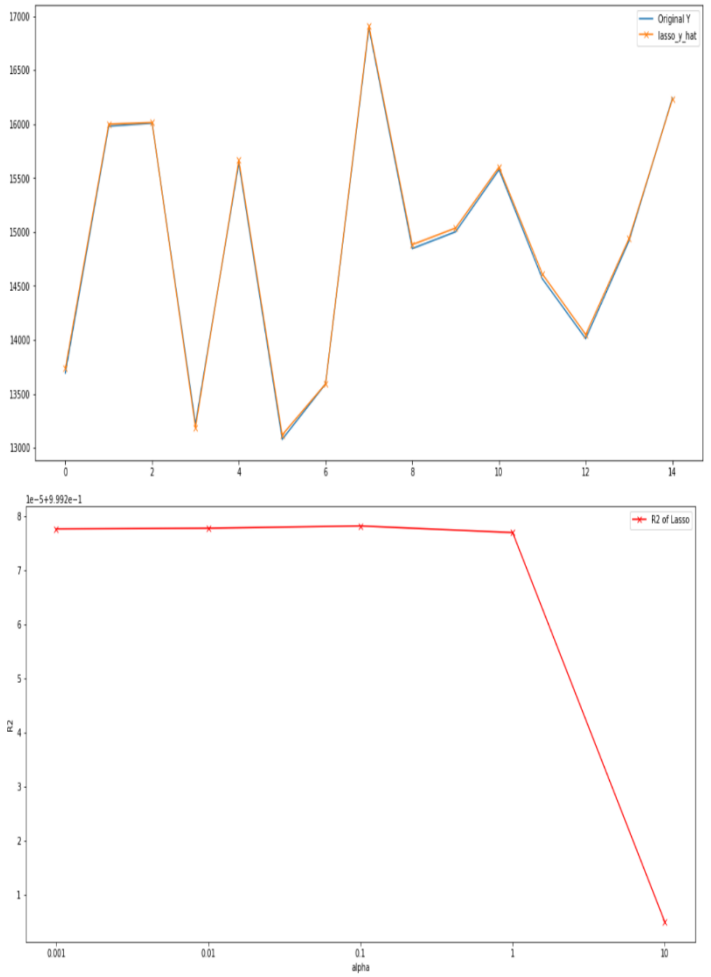
03

“ LASSO REGRESSION ”

1) LASSO REGRESSION

```
#LASSO==================================
from sklearn.metrics import r2_score,mean_absolute_error, mean
X = acne_df3_total.iloc[:,2:]
y = acne_df3_total.iloc[:,1]
X_train, X_test, y_train, y_test = train_test_split(X, y, test
lasso_alpha = 0.1
lasso = Lasso(alpha = lasso_alpha)
lasso.fit(X_train,y_train)
lasso_y_hat = lasso.predict(X_test)
linear_MAE, ridge_MAE, lasso_MAE = mean_absolute_error(y_test,

print('R2 score - Lasso: %.2f' %(lasso_r2))
print('MSE - Lasso: %.2f' %(lasso_MSE))
print('MAE - Lasso: %.2f' %(lasso_MAE))
```

R2 : 1.00 MSE : 931.25 MAE : 27.32

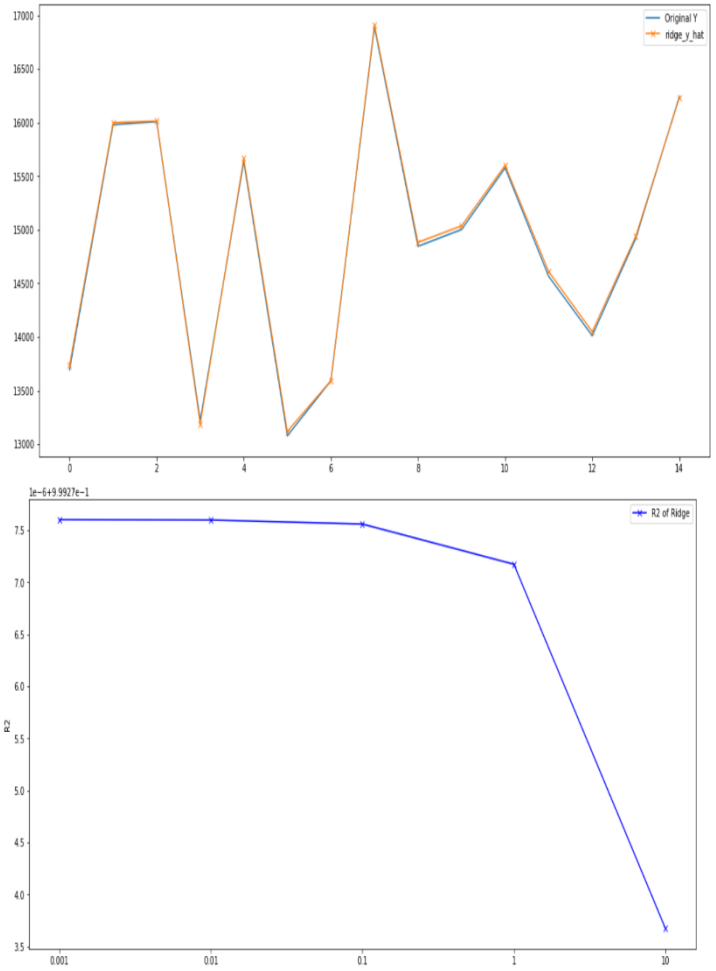


01

“ RIDGE REGRESSION ”

1) RIDGE REGRESSION

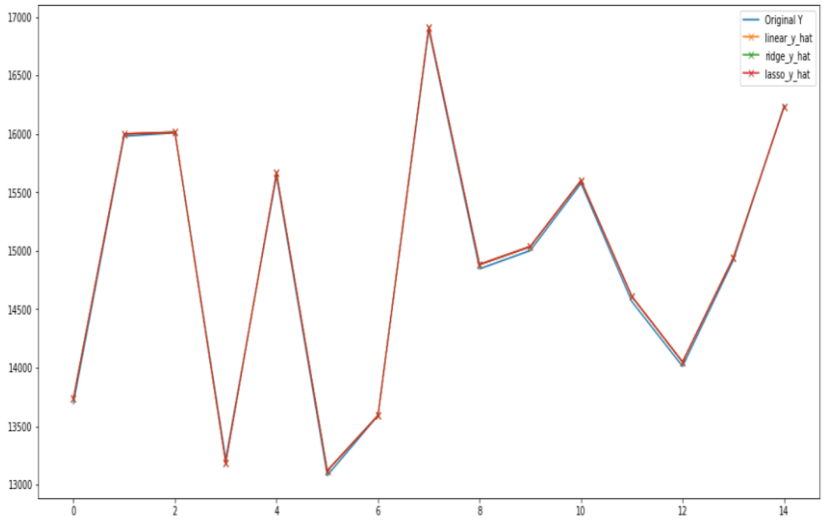
```
#RIDGE==================================
from sklearn.metrics import r2_score,mean_absolute_error, mean_squared_er
X = acne_df3_total.iloc[:,2:]
y = acne_df3_total.iloc[:,1]


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
ridge_alpha = 1
ridge = Ridge(alpha = ridge_alpha)
ridge.fit(X_train,y_train)
ridge_y_hat = ridge.predict(X_test)

ridge_r2 = r2_score(y_test,ridge_y_hat)
ridge_MSE = mean_squared_error(y_test,ridge_y_hat)
ridge_MAE = mean_absolute_error(y_test,ridge_y_hat)

print('R2 score -  Ridge: %.2f' %(ridge_r2))
print('MSE - Ridge: %.2f ' %(ridge_MSE))
print('MAE - Ridge: %.2f' %(ridge_MAE))
```

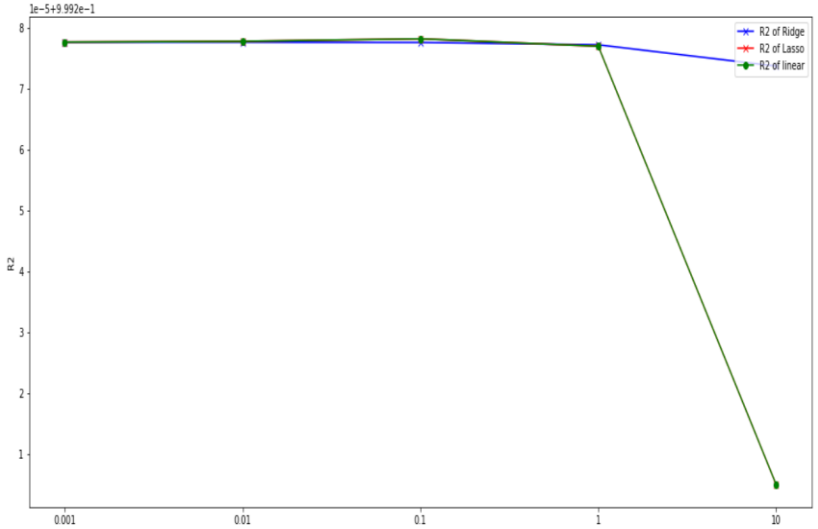


R2 : 1.00 MSE : 932.52 MAE : 27.32

03

“ REGRESSION 비교 ”





LASSO가 가장 적합한 분석기법이라는 것을 의미

03

“ REGRESSION 비교 ”

항목	R2 Score (결정계수)	MSE (손실함수)	MAE (평균 에러 지표)
설명	0에 가까우면 설명력 ↓ 1에 가까울수록 설명력 ↑	오답일수록 큰 값 정답일수록 작은 값	값이 작을수록 좋음
Linear Regression	1.00	931.96	27.29
Ridge Regression	1.00	932.52	27.32
Lasso Regression	1.00	931.25	27.32

R2는 동일, MSE와 MAE로 선택하는 것이 좋음

03

"statsmodels 알고리즘"

- Python내에서 다양한 통계분석을 가능하게 하는 모듈

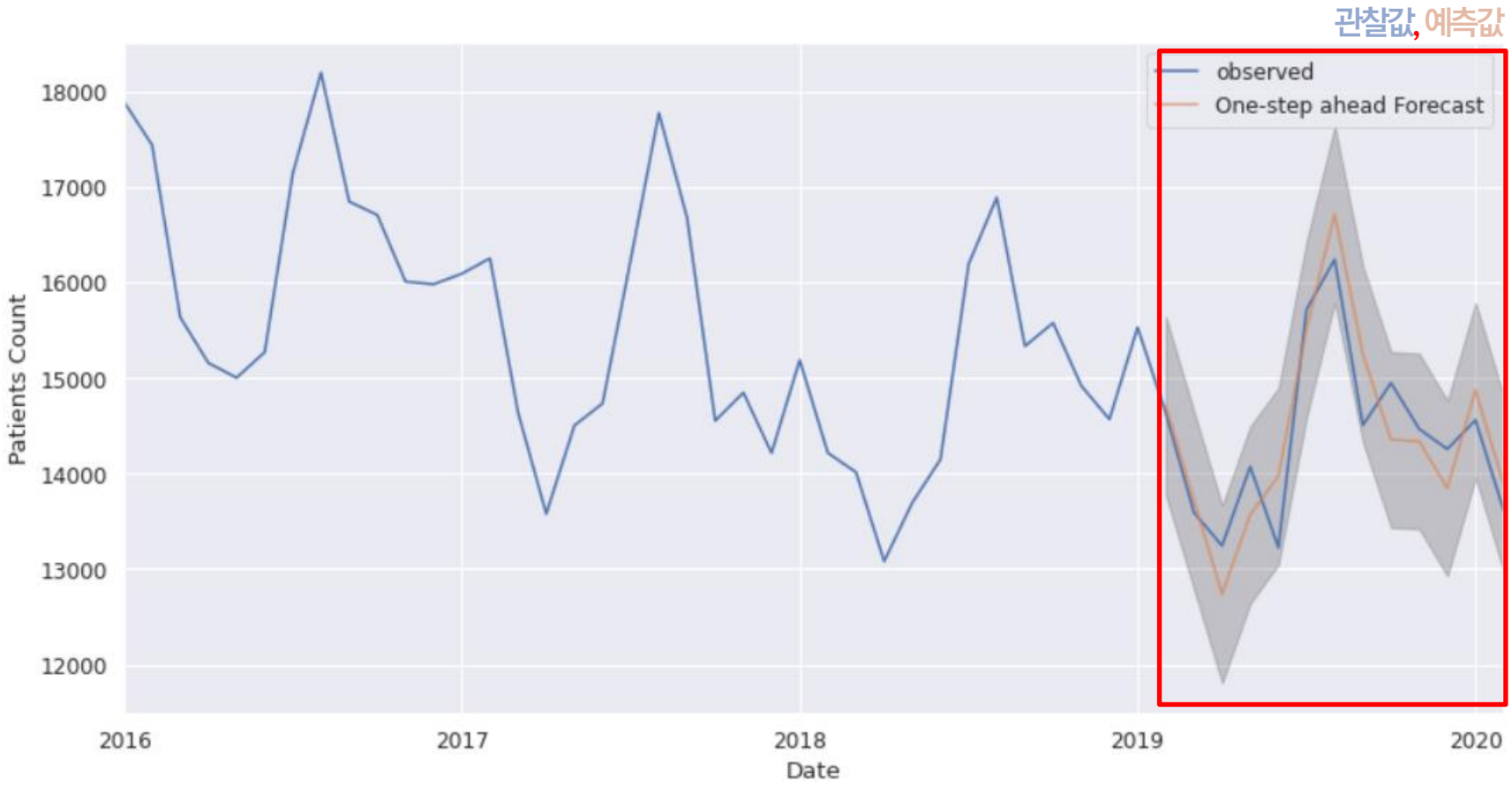
```
acne = files.upload()
data = pd.read_csv(io.BytesIO(acne['acne_df3.csv']))
data.head()
data['date'] = ['2016-01-01', '2016-02-01', '2016-03-01', '2016-04-01', '2016-05-01
                '2017-01-01', '2017-02-01', '2017-03-01', '2017-04-01', '2017-0
                '2018-01-01', '2018-02-01', '2018-03-01', '2018-04-01', '2018-0
                '2019-01-01', '2019-02-01', '2019-03-01', '2019-04-01', '2019-0
                '2020-01-01', '2020-02-01']
data['date'] = pd.to_datetime(data['date'])
data.set_index('date', inplace=True)
y = data['total'].resample('MS').mean()
y.plot(figsize = (15,6))
plt.show()
decomposition = sm.tsa.seasonal_decompose(y, model='additive') # tsa.seasonal_decom
fig = decomposition.plot()
plt.show()



# ARIMA : ARIMA는 자기진행적 통합 이동 평균, ARIMA 모델은 표기법 ARIMA(p, d, q)로 표
p = d = q = range(0, 2)
pdq = list(itertools.product(p, d, q)) # itertools : 자신만의 반복자를 만드는 모듈
seasonal_pdq = [(x[0], x[1], x[2], 12) for x in list(itertools.product(p, d, q))]

print('Examples of parameter combinations for Seasonal ARIMA...')
print('SARIMAX: {} x {}'.format(pdq[1], seasonal_pdq[1]))
print('SARIMAX: {} x {}'.format(pdq[1], seasonal_pdq[2]))
print('SARIMAX: {} x {}'.format(pdq[2], seasonal_pdq[3]))
print('SARIMAX: {} x {}'.format(pdq[2], seasonal_pdq[4]))
```

```
pred = results.get_prediction(start=pd.to_datetime('2019-02-01'), dynamic=False) # 예측할 시작 날짜 입력
pred_ci = pred.conf_int() #추정된 계수의 신뢰구간 계산
ax = y['2016':].plot(label='observed')
pred.predicted_mean.plot(ax=ax, label='One-step ahead Forecast', alpha=.7, figsize=(14, 7))
ax.fill_between(pred_ci.index,
                pred_ci.iloc[:, 0],
                pred_ci.iloc[:, 1], color='k', alpha=.2)
ax.set_xlabel('Date')
ax.set_ylabel('Patients Count')
plt.legend()
plt.show()

y_forecasted = pred.predicted_mean
y_truth = y['2019-02-01':]
mse = ((y_forecasted - y_truth) ** 2).mean()
print('The Mean Squared Error of our forecasts is {}'.format(round(mse, 2)))

print('The Root Mean Squared Error of our forecasts is {}'.format(round(np.sqrt(mse), 2)))

pred_uc = results.get_forecast(steps=100)
pred_ci = pred_uc.conf_int() #추정된 계수의 신뢰구간 계산
ax = y.plot(label='observed', figsize=(14, 7)) # observed : 관찰값
pred_uc.predicted_mean.plot(ax=ax, label='Forecast') # Forecast : 예측값
ax.fill_between(pred_ci.index,
                pred_ci.iloc[:, 0],
                pred_ci.iloc[:, 1], color='k', alpha=.25)
ax.set_xlabel('Date')
ax.set_ylabel('Patients Count')
plt.legend()
plt.show()
```

03

"statsmodels 알고리즘"

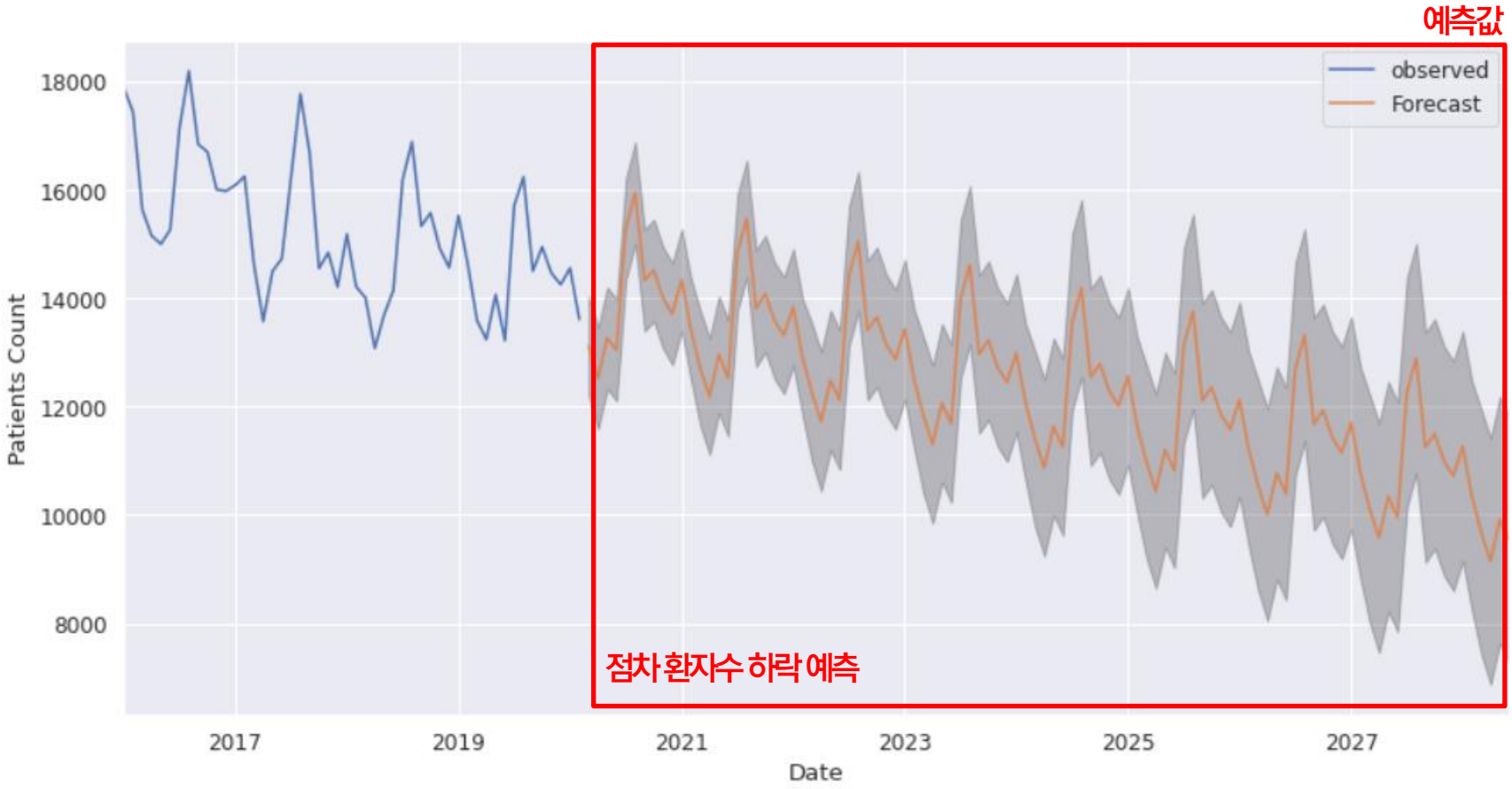
- Python내에서 다양한 통계분석을 가능하게 하는 모듈



03

"statsmodels 알고리즘"

- Python내에서 다양한 통계분석을 가능하게 하는 모듈



03

“ Prophet 알고리즘 ”

- 페이스북에서 만든 시계열 알고리즘으로 압도적으로 쉽다는 장점

```
data['date'] = ['2016-01-01', '2016-02-01', '2016-03-01', '2016-04-01', '2016-05-01', '2016-06-01', '2016-07-01', '2016-08-01', '20
                '2017-01-01', '2017-02-01', '2017-03-01', '2017-04-01', '2017-05-01', '2017-06-01', '2017-07-01', '2017-08-01',
                '2018-01-01', '2018-02-01', '2018-03-01', '2018-04-01', '2018-05-01', '2018-06-01', '2018-07-01', '2018-08-01',
                '2019-01-01', '2019-02-01', '2019-03-01', '2019-04-01', '2019-05-01', '2019-06-01', '2019-07-01', '2019-08-01',
                '2020-01-01', '2020-02-01']
data['date'] = pd.to_datetime(data['date'])

acne_prophet = data[['date', 'total']]
acne_prophet = acne_prophet.rename(columns = {'date' : 'ds', 'total' : 'y'})
acne_prophet = acne_prophet[['ds', 'y']].reset_index(drop=True)
model = Prophet(growth='linear',
                interval_width=0.97, # 정확도 조정
                seasonality_mode='multiplicative',
                yearly_seasonality=True,
                changepoint_range=0.7, # 데이터의 70% 정도에서 changepoint
                changepoint_prior_scale=0.3)
model.fit(acne_prophet)
future = model.make_future_dataframe(periods=365)
forcast = model.predict(future)
forcast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']]
fig = model.plot(forcast)
a = add_changepoints_to_plot(fig.gca(), model, forcast)
accne = model.plot(forcast)
sns.set(font_scale=1.1)
fig1 = model.plot_components(forcast)
plt.tight_layout()
```

03

“ Prophet 알고리즘 ”

- 페이스북에서 만든 시계열 알고리즘으로 압도적으로 쉽다는 장점

03

“ Prophet 알고리즘 ”

- 페이스북에서 만든 시계열 알고리즘으로 압도적으로 쉽다는 장점

04

“ 상관관계 분석과 예측 결과 ”

1) 상관관계 분석

1. 여드름은 기온 〉 강수량 〉 자외선 순으로 유의미한 관계가 있음
   (단, 약한 양의 상관관계로 큰 의미 X)

2) 회귀분석

1. R2 결정 계수가 1로 높지만, MSE의 점수가 높아 데이터의 적합성에 의문
   (단, 3가지의 회귀분석 기법의 눈에 띄는 차이는 없음)

3) 예측 결과

1. 20년 2월 이후 환자 수가 줄어들 것으로 예측
   (단, 코로나 19 변수에 적용 불가로 실제 20년 2월 이후 관측값과 차이가 있을 것)

05

“ 결론 및 발전방향 ”

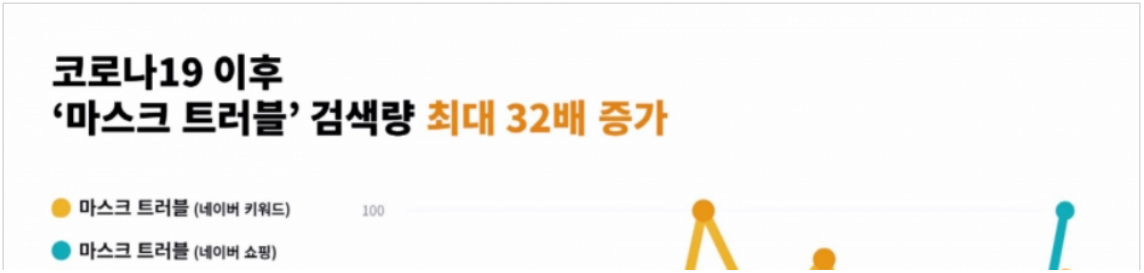
코로나 마스크로 지친 피부 달래주오~검색어 '마스크 트러블' 급증

위클리포스트 | 승인 2020.11.04 00:24 | 댓글 0

가가

[2020년 11월 03일] - 초겨울을 앞두고 코로나19 마스크 착용에 환절기까지 겹치면서 피부 트러블로 화난 모공을 진정시키기 위한 '저자극 스킨케어템'을 찾는 이들이 늘고 있다. 마스크 속 고온다습한 환경에 피지 분비가 활성화되면서 피지와 땀, 노폐물이 엉켜 모공을 막는 데다, 지속적인 마찰로 피부가 예민해져 '마스크 트러블'이 생기는 것.

실제로 지난 2월부터 약 8개월간 네이버 검색량을 분석한 결과 키워드 '피부 트러블'은 18% 증가했지만, 코로나 연관 키워드인 '마스크 트러블' 네이버 검색량은 161%, 네이버 쇼핑 화장품-미용 분야 검색량은 3,233% 급증했다. 최근엔 마스크(mask)와 여드름(acne)의 합성어인 '마스크네'(maskne)라는 말까지 등장하는 등 관심도가 많이 늘어나고 있다.



출처 : 위클리포스트 : http://www.weeklypost.kr/news/articleView.html?idxno=1598

05

“ 결론 및 발전방향 ”

M 메이크업 픽서 (makeup fixer)

A 트러블 케어 (anti-trouble)

S 방구석 셀프 뷰티족 (self beauty)

K 이너 뷰티 건강기능식품의 약진 (keep healthy)

장기간 마스크 착용으로 여드름 환자 증가 예측
분석 결과와 오차가 클 것으로 예상

05

“ 결론 및 발전방향 ”

성별
맞춤 서비스

연령별
맞춤 서비스

상품
세분화

시기별
맞춤 마케팅

05

“ 결론 및 발전방향 ”

1) 시사점

1. 헬스케어 분야 데이터 습득과 정제, 분석의 어려움이 있다.
2. 추가적인 데이터 확장과 알고리즘 사용으로 발전 가능성이 있다.
3. 도메인 지식이 매우 중요하다.
4. 공공 데이터의 중요성을 다시 알게 되었다.

05

“ 결론 및 발전방향 ”

1) 한계점

1. 미래 관측 데이터와 실제 데이터의 차이가 있을 것으로 예측
2. 질병과 관련된 데이터가 매우 제한적이다. (2020년 자료 無)
3. 외부환경에 대한 추가적인 데이터 수집이 제한적이다.
4. 환자 수와 외부환경의 유의미한 상관관계를 분석하지 못했다.
5. 실질적인 환자들의 데이터를 수집하지 못했다.

THANK YOU

발표자 김 예 찬