

원숭이 두창 시각화 및 예측

2018108254 김태경

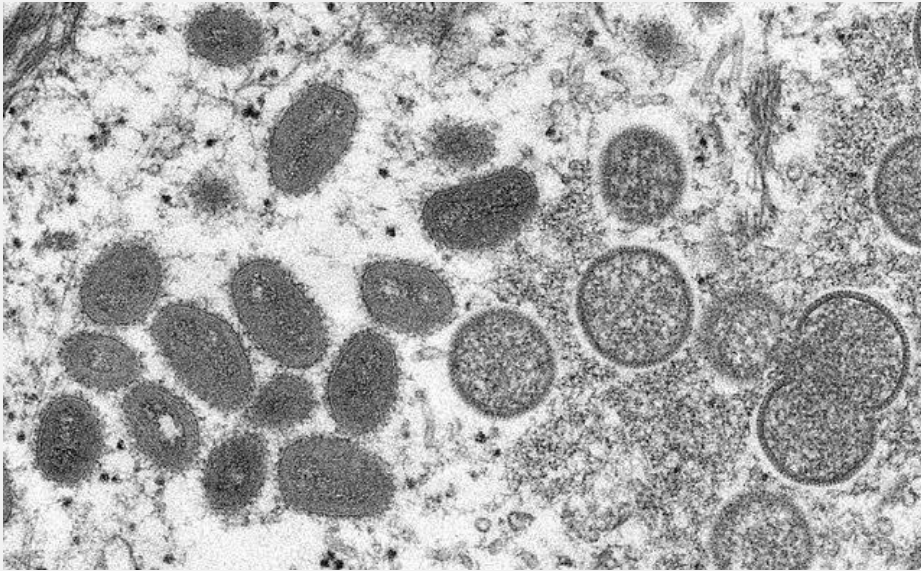
Index

1. 개요
2. 사용 라이브러리의 종류
3. 파일로드
4. 데이터 시각화와 분석
5. 특징 상관관계 분석
6. 데이터 전처리 및 데이터 분할

7. 머신러닝 알고리즘
8. 모델 평가
9. 맺는 말
10. 참고 문헌



MONKEYPOX



- 원숭이두창(Monkeypox)은 원숭이두창 바이러스(Monkeypox virus)에 감염되어 발생하는 급성 발진성 감염병
- 2022년 유행 전까지는 중앙 아프리카 및 서부 아프리카의 농촌 열대우림지역에서 주로 발생하는 풍토병

MONKEYPOX

- 2022년 5월 이후 유럽과 북미를 중심으로 다수국가에서 풍토병지역과 연관성이 없는 감염사례가 이례적으로 유행하여 환자가 증가하고 발생지역이 확대



세계보건기구(WHO)는 2022년 7월 23일 원숭이두창에 대해
'국제적 공중보건 비상사태(PHEIC)'를 선언

질병 억제를 위한 연구와 자금 지원, 국제적 보건 조치 등을
강력하게 추진

INFECTION ROUTE

- 원숭이두창 감염은 원인 바이러스가 사람의 피부, 호흡기, 점막을 통해 체내로 들어온다.
- 바이러스에 감염된 동물이나 바이러스에 오염된 물건을 통한 전파가 가능하며, 사람 간에는 병변 · 체액 · 호흡기 비말 및 침구와 같은 오염된 물질과의 접촉을 통해 감염

SYMPTOMS

- 치명률은 3~6% 내외, COVID-19는 치명률 1% 내외
- 증상은 천연두와 비슷하게 발열, 두통, 근육통, 요통, 림프절 비대, 오한, 허약감 등을 시작으로 1~3일 후에 얼굴을 중심으로 발진 증상이 나타나며 점차 몸의 다른 부위로 발진이 확산
- 발진의 경우 수포나 농포 등으로 진행되는데, 특히 손에는 수포성 발진과 함께 심한 가려움증이 나타난다. 잠복기는 보통 6~13일이며, 발현된 증상은 약 2~4주간 지속
- 원숭이두창의 진단은 항원검사, PCR(유전자 검출검사), 바이러스 배양 등
- 치료의 경우 전용 치료제는 없고 항바이러스제가 사용

NECESSITY

- 원숭이두창의 사례는 꾸준히 증가하고 있으며 유행병이 되기 전에 가능한 한 빨리 패턴, 증상 및 치료법을 예측할 수 있어야 함.
- 패턴을 예측해 볼 수 있고 사용 가능한 데이터를 사용하여 해결책을 찾을 수 있어야 함.



2. 사용 라이브러리의 종류

```
#Data Loading
import numpy as np
import pandas as pd

#Data Visualization
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px

#Feature Engineering
from sklearn.model_selection import train_test_split #학습용 & 테스트용 분리
from sklearn import preprocessing #전처리

#Modelling
from sklearn.linear_model import LinearRegression #선형 회귀
from sklearn.tree import DecisionTreeRegressor #결정트리
from sklearn.ensemble import RandomForestRegressor #랜덤포레스트
from sklearn.linear_model import ElasticNet #penalized regression

#Evaluation
from sklearn.metrics import mean_absolute_error, mean_squared_error, mean_absolute_percentage_error, r2_score

import os
```


DATA LOADING



- NumPy는 다양한 수치연산, 변환 기능 등을 갖는 모듈
- 빠른 연산 속도



- Pandas는 데이터프레임을 읽어들이고 유지하고 관리할 수 있는 모듈
- 유연한 데이터 구조 제공

2. 사용 라이브러리의 종류

DATA VISUALIZATION



seaborn

matplotlib

plotly | Graphing Libraries

분석한 데이터를
시각화하여 가독성을
높이는 역할

MODELLING







- LinearRegression
- DecisionTreeRegressor
- RandomForestRegressor
- ElasticNet

3. 파일 로드

DATA LOADING

```
# CSV 파일 읽어오기
df = pd.read_csv('../input/monkeypox-dataset-daily-updated/Monkey_Pox_Cases_Worldwide.csv')
# 불러온 데이터의 기본적으로 상위 5개의 행을 출력
df.head()
```

Input

- ▼  monkeypox-dataset-daily-updated
 -  Daily_Country_Wise_Confirmed_Cases.csv
 -  Monkey_Pox_Cases_Worldwide.csv
 -  Worldwide_Case_Detection_Timeline.csv

| | Country | Confirmed_Cases | Suspected_Cases | Hospitalized | Travel_History_Yes | Travel_History_No |
|---|---------------|-----------------|-----------------|--------------|--------------------|-------------------|
| 0 | England | 3412.0 | 0.0 | 5.0 | 2.0 | 7.0 |
| 1 | Portugal | 908.0 | 0.0 | 0.0 | 0.0 | 34.0 |
| 2 | Spain | 7083.0 | 0.0 | 13.0 | 2.0 | 0.0 |
| 3 | United States | 24403.0 | 0.0 | 4.0 | 41.0 | 11.0 |
| 4 | Canada | 1388.0 | 12.0 | 1.0 | 5.0 | 0.0 |

Country

Confirmed_Cases

Suspected_Cases

Hospitalized

Travel_History_Yes

Travel_History_No

나라

확진

의심

입원

여행 이력 있음

여행 이력 없음

3. 파일 로드

PRE-PROCESSING

데이터에 대한 전반적인 정보
`df.info()`

df를 구성하는 행과 열의 크기, 컬럼명,
컬럼을 구성하는 값의 자료형 출력

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 129 entries, 0 to 128
Data columns (total 6 columns):
#   Column                Non-Null Count  Dtype  
---  -
0   Country                129 non-null   object 
1   Confirmed_Cases        129 non-null   float64
2   Suspected_Cases        129 non-null   float64
3   Hospitalized           129 non-null   float64
4   Travel_History_Yes     129 non-null   float64
5   Travel_History_No      129 non-null   float64
dtypes: float64(5), object(1)
memory usage: 6.2+ KB
```

#column 별 결측값 개수의 합
`df.isnull().sum()`

결측값의 개수를 측정

```
Country                0
Confirmed_Cases        0
Suspected_Cases        0
Hospitalized           0
Travel_History_Yes     0
Travel_History_No      0
dtype: int64
```

4. 탐색적 데이터 분석

EXPLORATORY DATA ANALYSIS

- 데이터를 분석하기 전에 그래프나 통계적인 방법으로 자료를 직관적으로 바라보는 과정

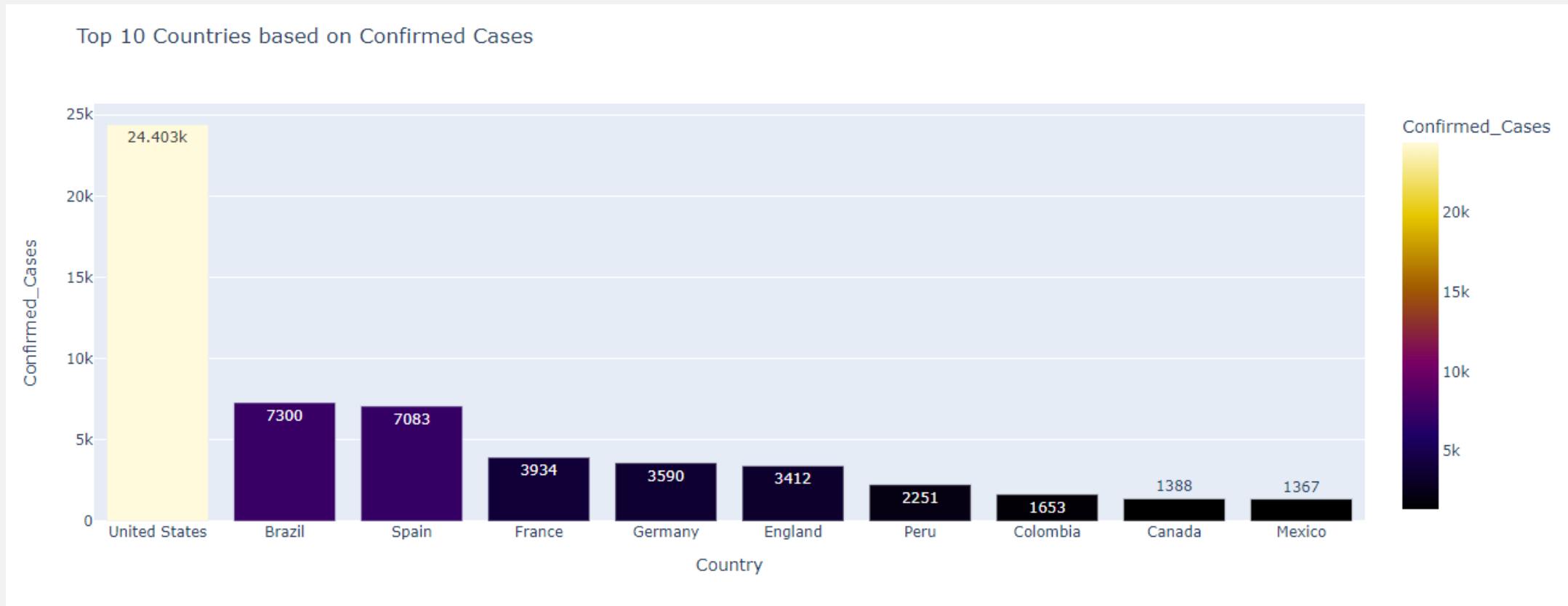
```
def top10plots(col=None):  
    #Sorting the Dataset  
    df_sorted = df.sort_values(by=col,ascending=False).reset_index()  
    #Getting the Top10  
    top10 = df_sorted[:10]  
    # Plotting the Top10  
    label_text = ' '.join(col.split('_'))  
    labeldict = {'size':'15','weight':'3'}  
    titledict = {'size':'20','weight':'3'}  
    fig = px.bar(x='Country',  
                 y=col,  
                 data_frame=top10,  
                 labels=['Country',label_text],  
                 color=col,  
                 color_continuous_scale='electric',  
                 text_auto=True,  
                 title=f'Top 10 Countries based on {label_text}')  
  
    fig.show()
```

데이터셋을 정렬하여 상위 10개
데이터를 col에 따라 색을 구별하여
bar로 그림

4. 탐색적 데이터 분석

- 시각화

```
top10plots(col='Confirmed_Cases')
```

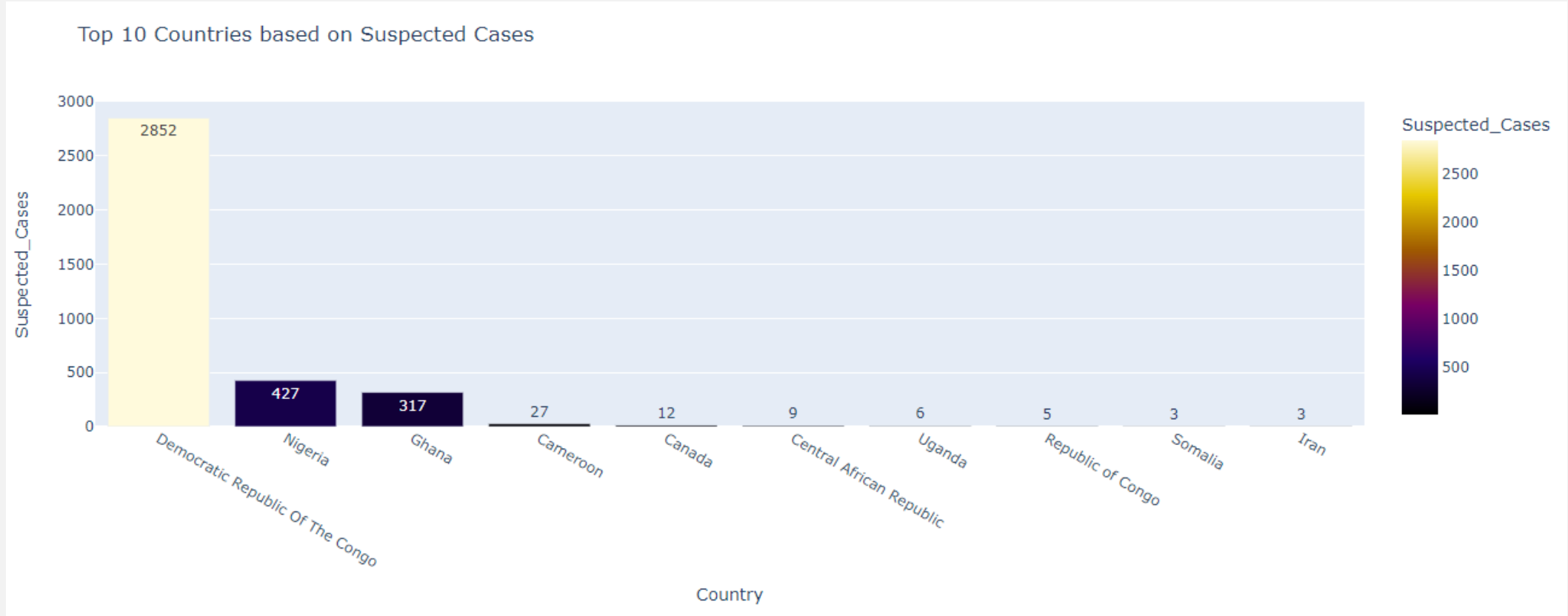


➡ 북미와 남미, 유럽 위주로 확진

4. 탐색적 데이터 분석

- 시각화

```
top10plots(col='Suspected_Cases')
```

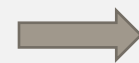
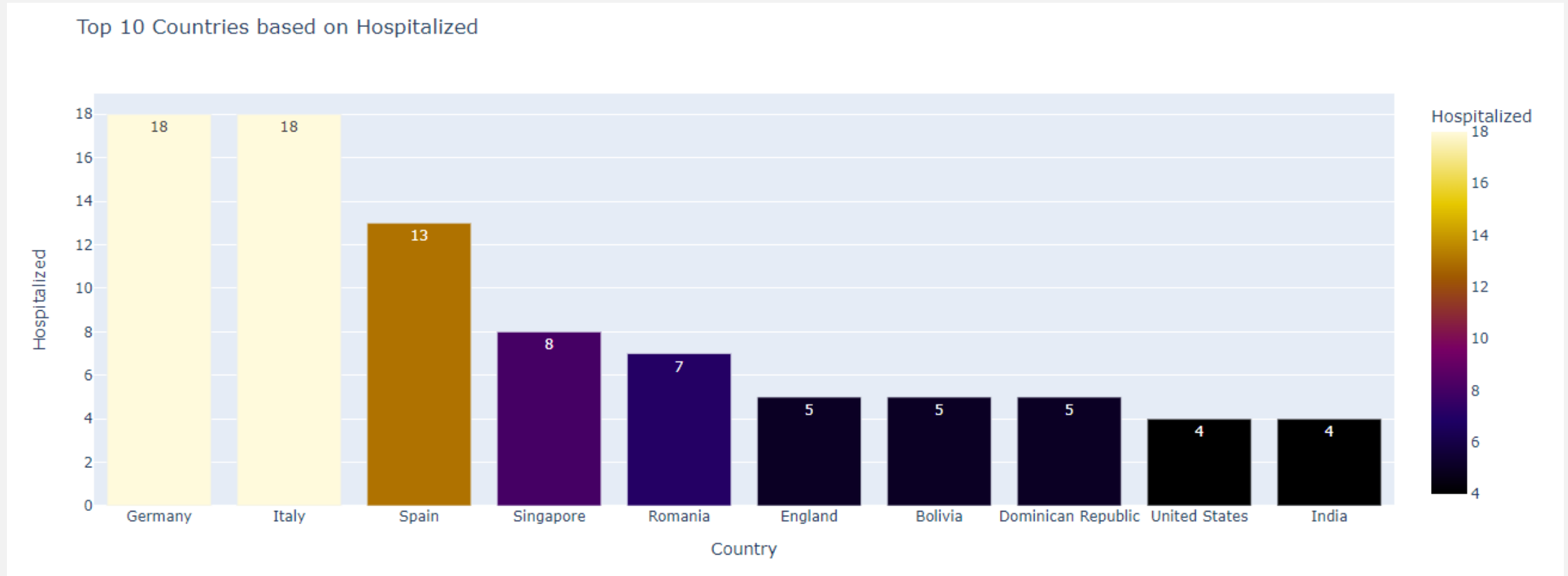


아프리카 외 주로 의심 환자 발생

4. 탐색적 데이터 분석

- 시각화

```
top10plots(col='Hospitalized')
```

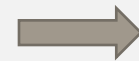
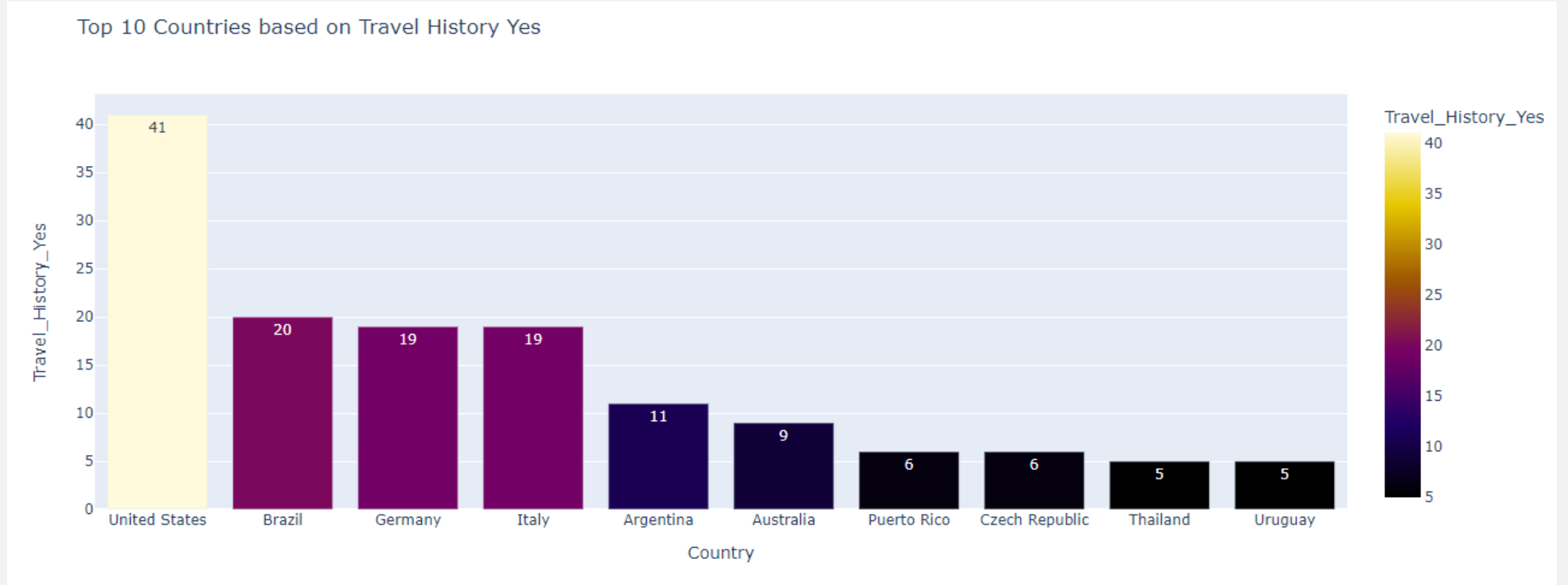


유럽 위주로 입원 중인 상황

4. 탐색적 데이터 분석

- 시각화

```
top10plots(col='Travel_History_Yes')
```

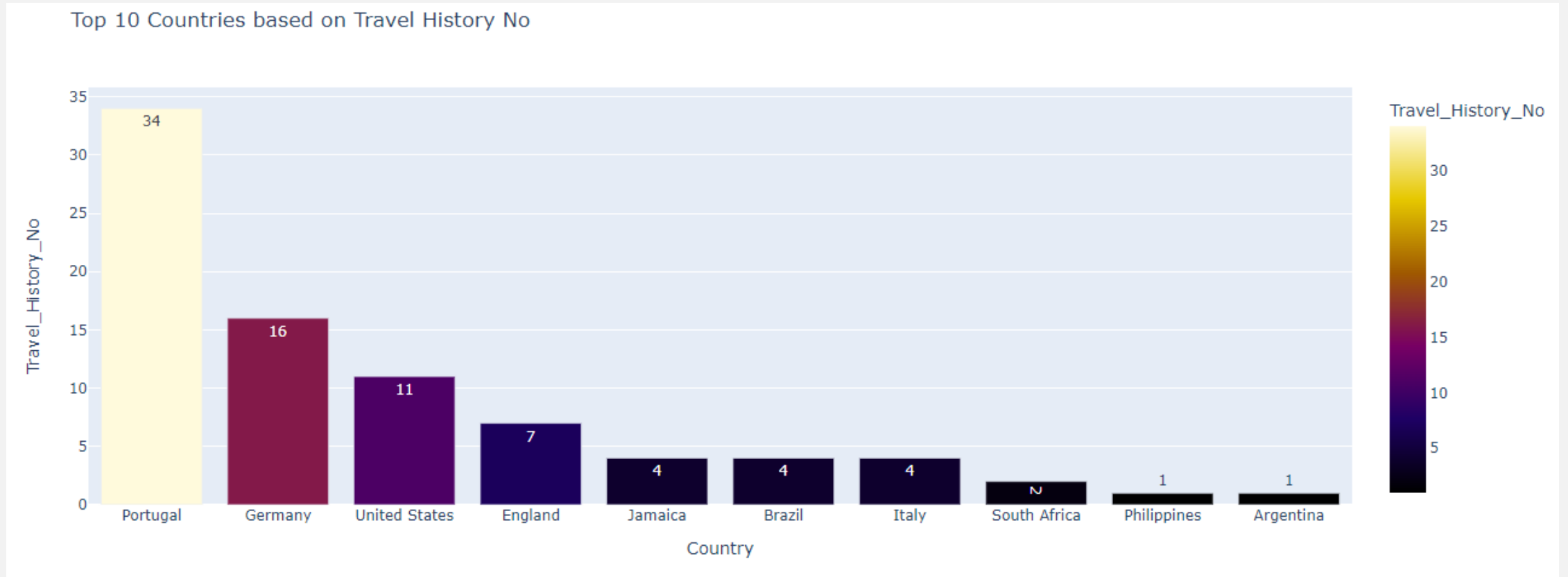


미국이 여행 이력이 있는 환자가
가장 많이 발생하는 지역

4. 탐색적 데이터 분석

- 시각화

```
top10plots(col='Travel_History_No')
```



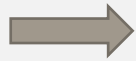
포르투갈이 여행 이력이 없는
환자가 가장 많이 발생하는 지역

4. 탐색적 데이터 분석

```
df[df['Country']=='Portugal']
```

| | Country | Confirmed_Cases | Suspected_Cases | Hospitalized | Travel_History_Yes | Travel_History_No |
|---|----------|-----------------|-----------------|--------------|--------------------|-------------------|
| 1 | Portugal | 908.0 | 0.0 | 0.0 | 0.0 | 34.0 |

확진자 중 여행 이력이 있다고 응답한 사람은 없고
여행 이력 없는 사람이 가장 많음



포르투갈에서 시작되었을
가능성

4. 탐색적 데이터 분석

```
def world_map(col=None, title=None):  
    ...  
    Function to plot a choropleth world Map.  
    Arguments required:  
    1. Column Name for which distribution is to be plotted.  
    2. The Title of the Graph.  
    ...  
    fig = px.choropleth(df,  
                        locations='Country',  
                        locationmode='country names',  
                        hover_name='Country',  
                        color=col,  
                        color_continuous_scale='electric')  
  
    fig.update_layout(title_text=title)  
    fig.show()
```

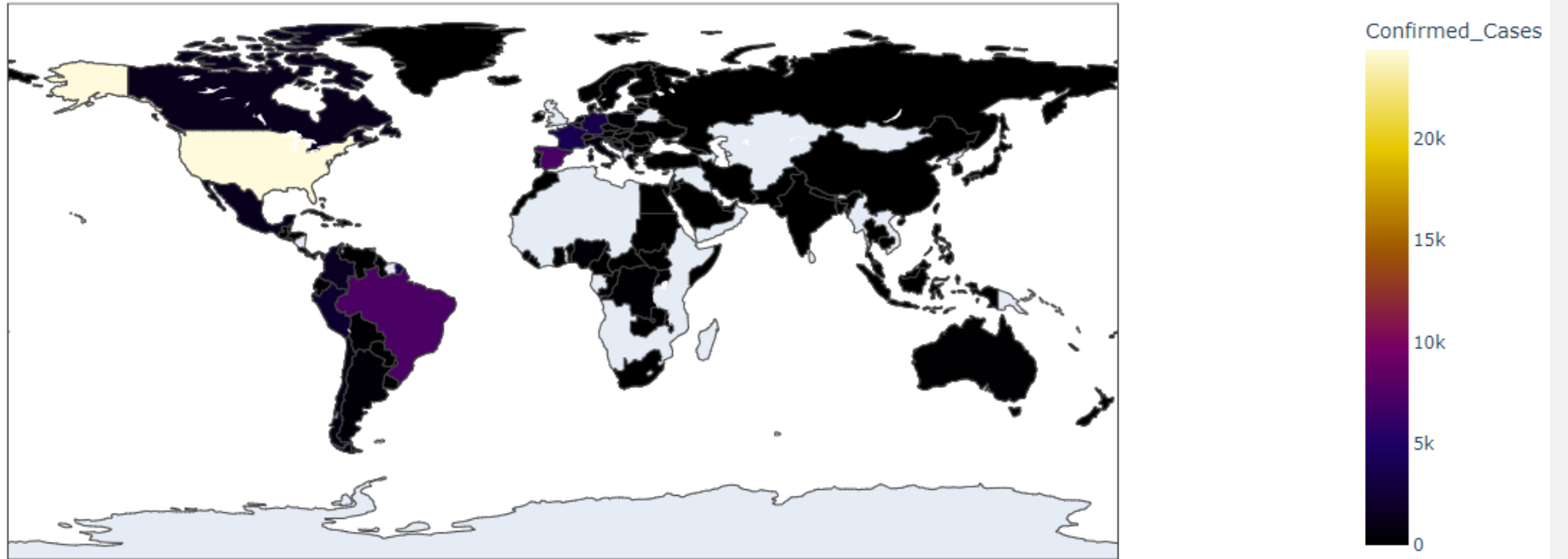
데이터 df를 읽어와서 col 값에
따라 국가 별로 색깔로 구분

4. 탐색적 데이터 분석

- 시각화

```
world_map(col='Confirmed_Cases',title='Confirmed MonkeyPox Cases Across The Globe')
```

Confirmed MonkeyPox Cases Across The Globe



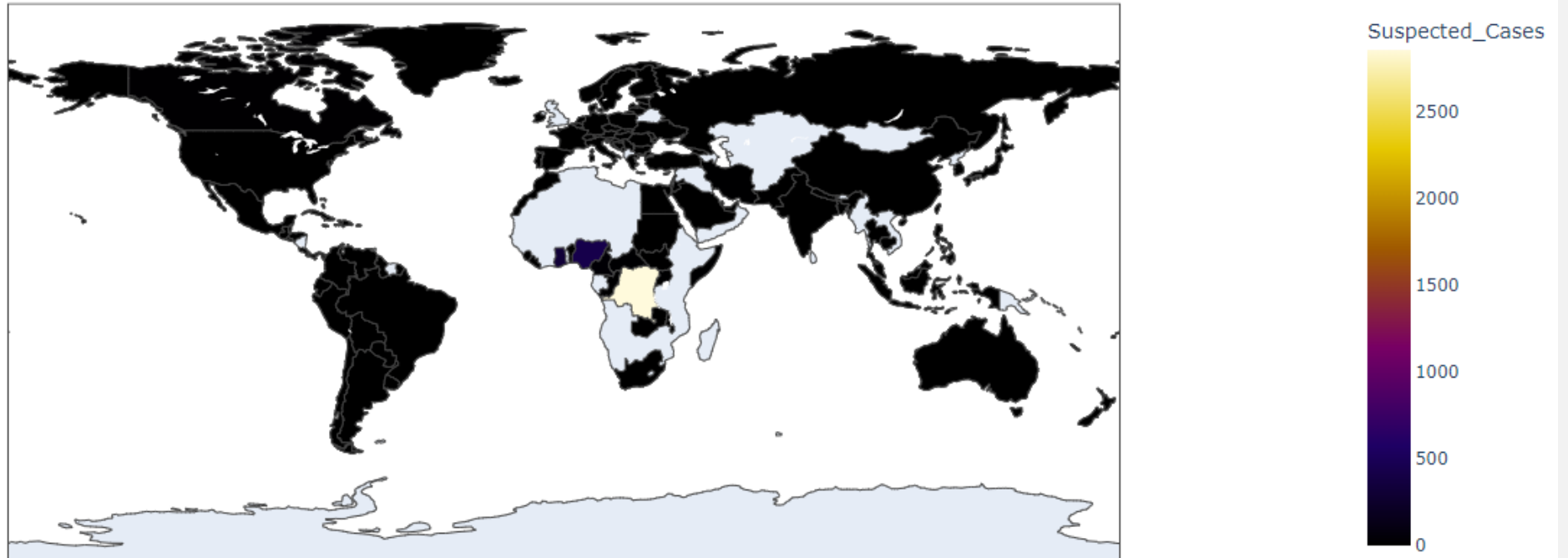
확진 현황을 한 눈에 볼 수 있는 지도

4 탐색적 데이터 분석

- 시각화

```
world_map(col='Suspected_Cases',title='Suspected MonkeyPox Cases Across The Globe')
```

Suspected MonkeyPox Cases Across The Globe



의심 케이스를 한 눈에 볼 수 있는 지도

FEATURE ENGINEERING

- Feature Engineering은 머신러닝 알고리즘을 작동하기 위해 데이터에 대한 도메인 지식을 활용하여 특징(Feature)를 만들어내는 과정
- 가지고 있는 데이터가 방대할 지라도 모든 데이터를 결과로 도출하는데 쓰면 정확하게 나타날 듯하지만 오히려 결과를 잘못 도출하는 경우가 많다.
- 머신 러닝의 성능은 어떠한 데이터를 입력하는 것에 따라 굉장히 의존적

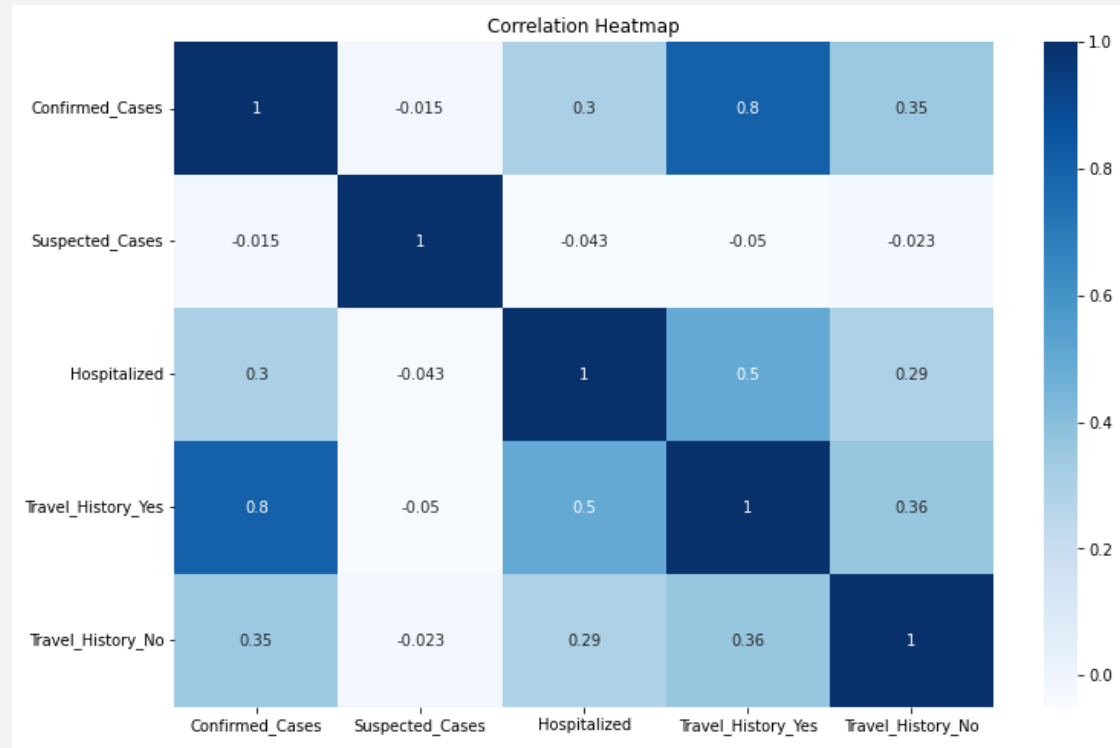


5. 특징 상관관계 분석

CORRELATION HEATMAP

```
plt.figure(figsize=(12,8));  
sns.heatmap(df.corr(),annot=True,cmap='Blues');  
plt.title('Correlation Heatmap');
```

- 여행 이력과 확진은 긍정적인 상관관계
- 여행 이력과 입원은 긍정적인 상관관계



5. 특징 상관관계 분석

SCATTERPLOT DISTRIBUTION

```
def corr_scatter(col1=None, col2=None, dataframe=None):  
    plt.figure(figsize=(12,8));  
    sns.scatterplot(x=col1,y=col2,data=dataframe);  
    xlabel = ' '.join(col1.split('_'))  
    ylabel = ' '.join(col2.split('_'))  
    plt.xlabel(xlabel);  
    plt.ylabel(ylabel);  
    plt.title(f'{xlabel} vs {ylabel}');  
    plt.show();
```

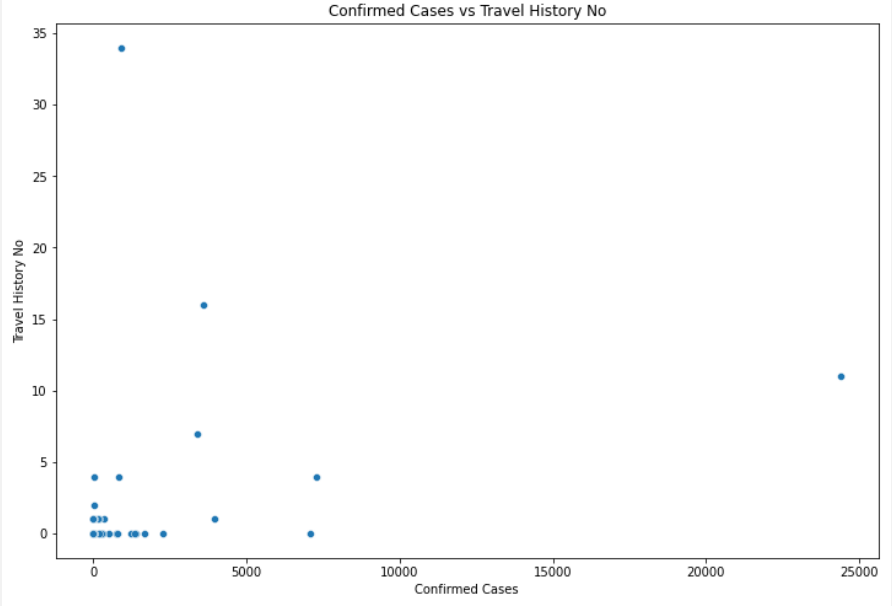
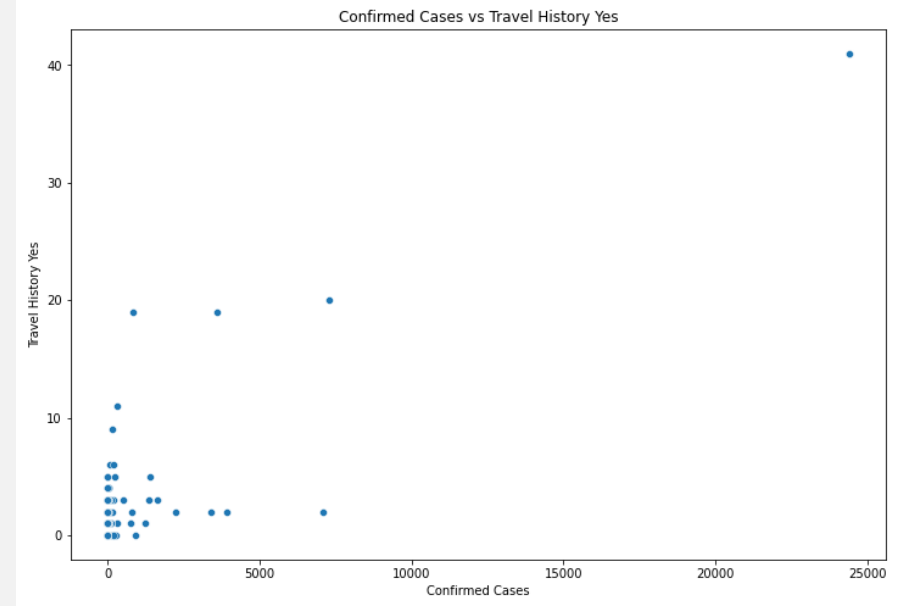
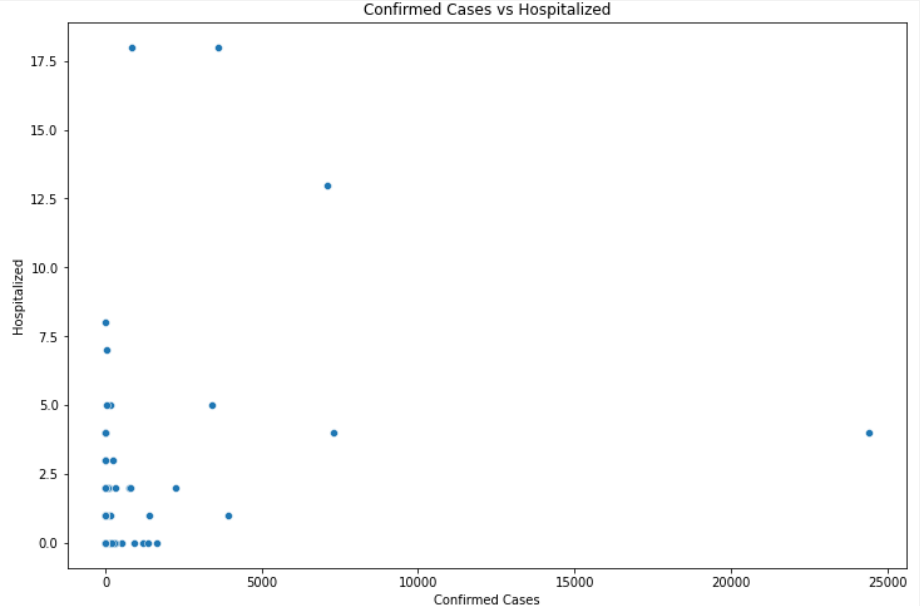
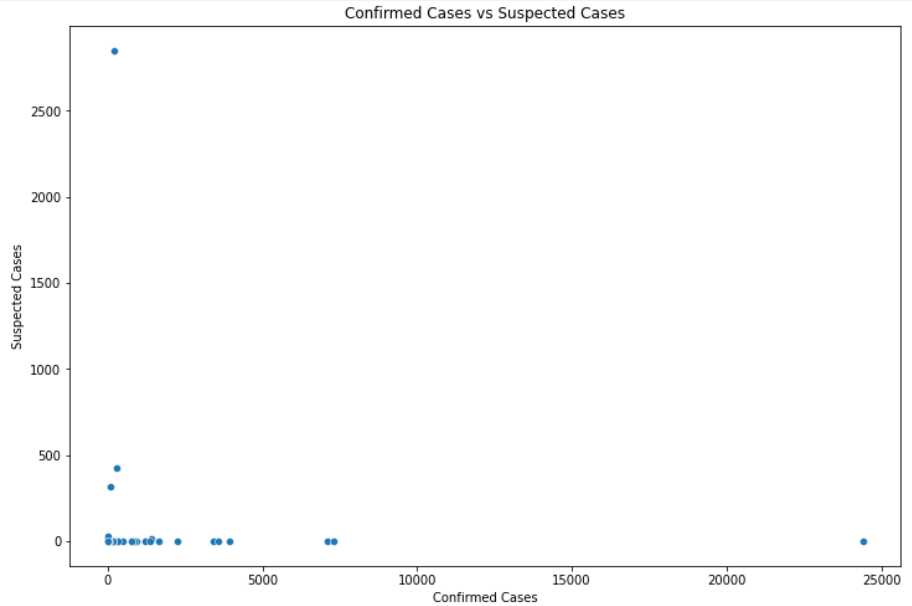
산점도

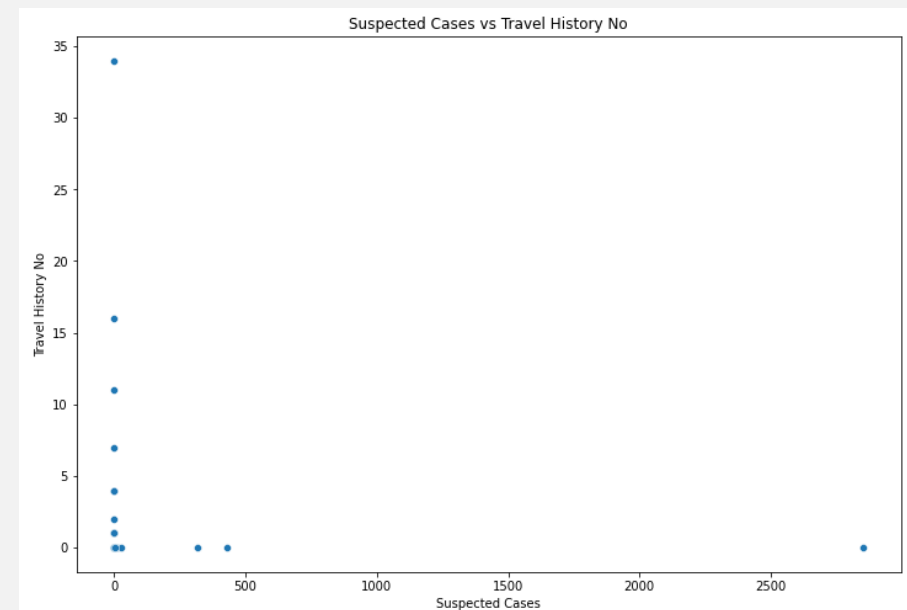
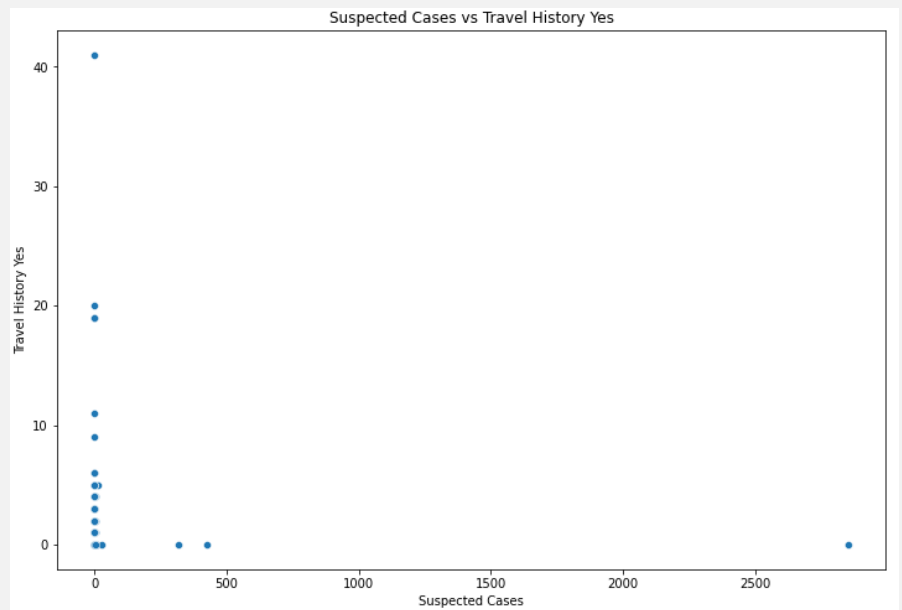
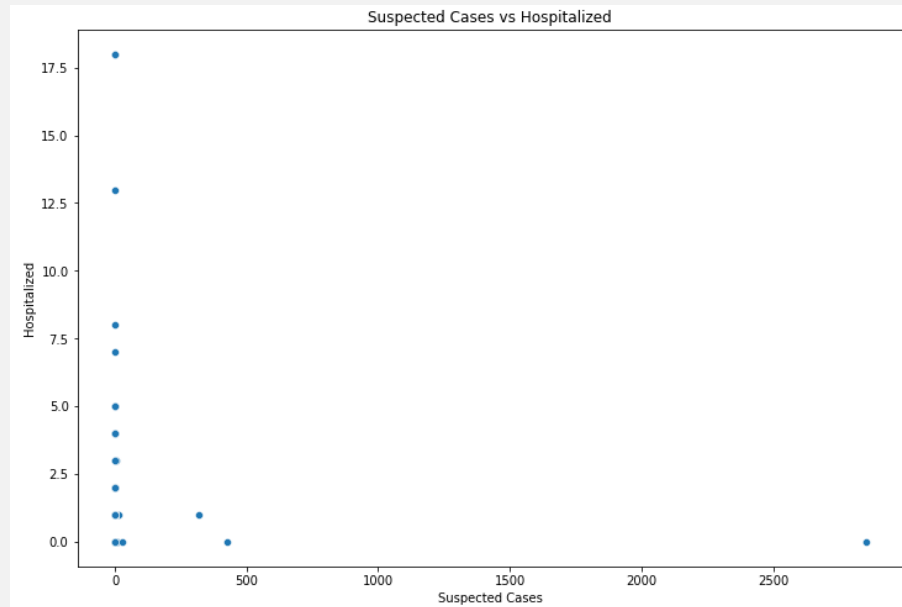
x와 y 간 상관관계가 연관
정도를 판단하는데 사용

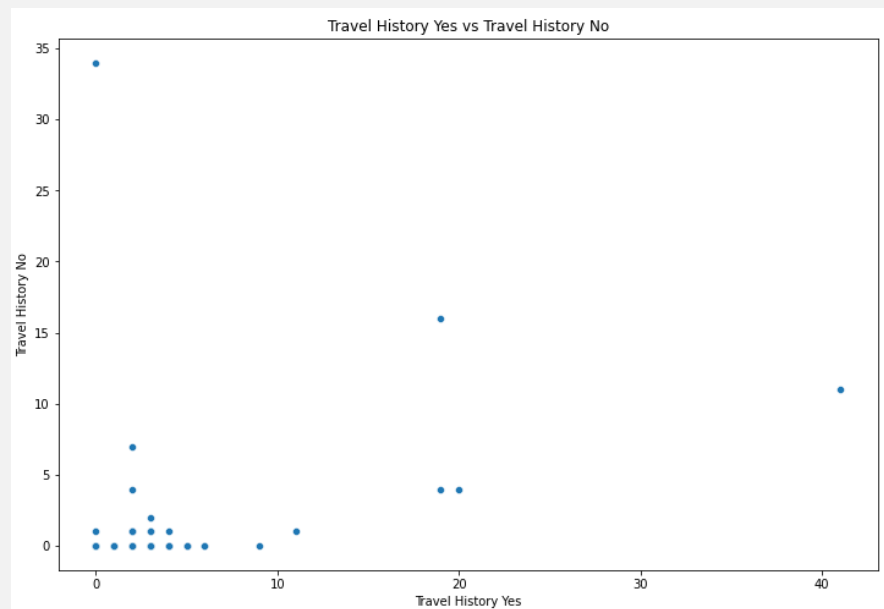
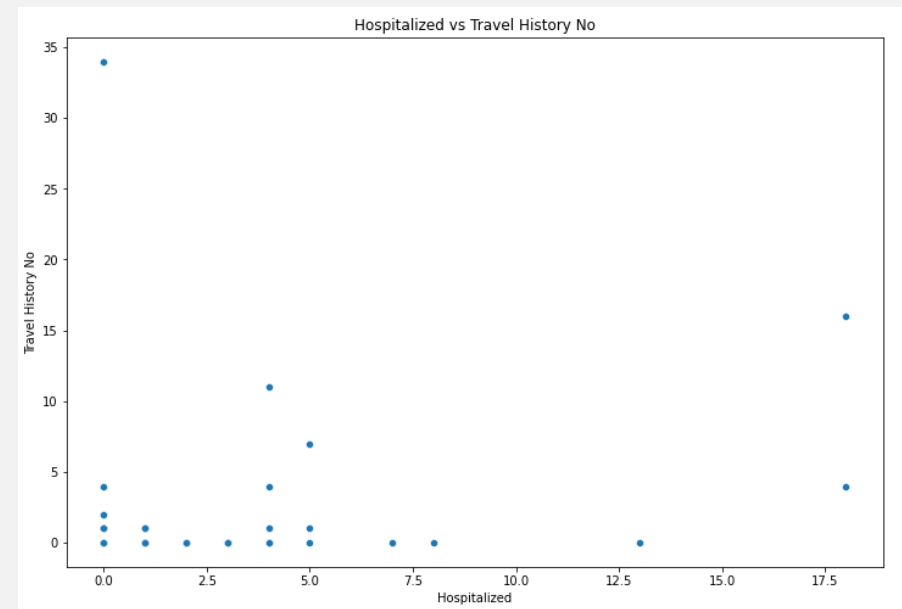
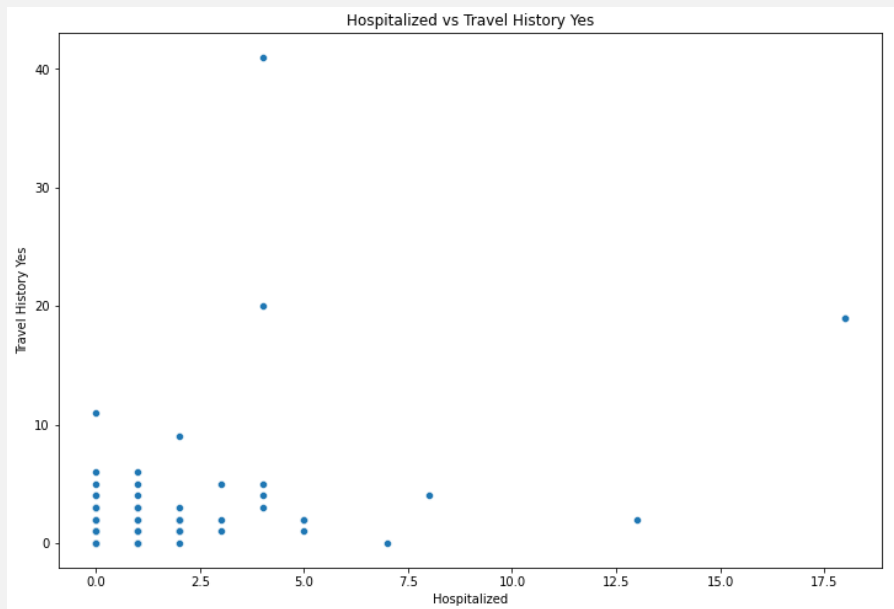
```
columns = df.columns.drop(['Country'])
```

```
for col1 in columns:  
    for col2 in columns.drop([col1]):  
        corr_scatter(col1=col1,col2=col2,dataframe=df);  
    columns = columns.drop([col1])
```

모든 열(columns)의
상관관계를 확인







5. 특징 상관관계 분석

PAIR PLOT

```
pair_df = df.drop(['Country'],axis=1)
```

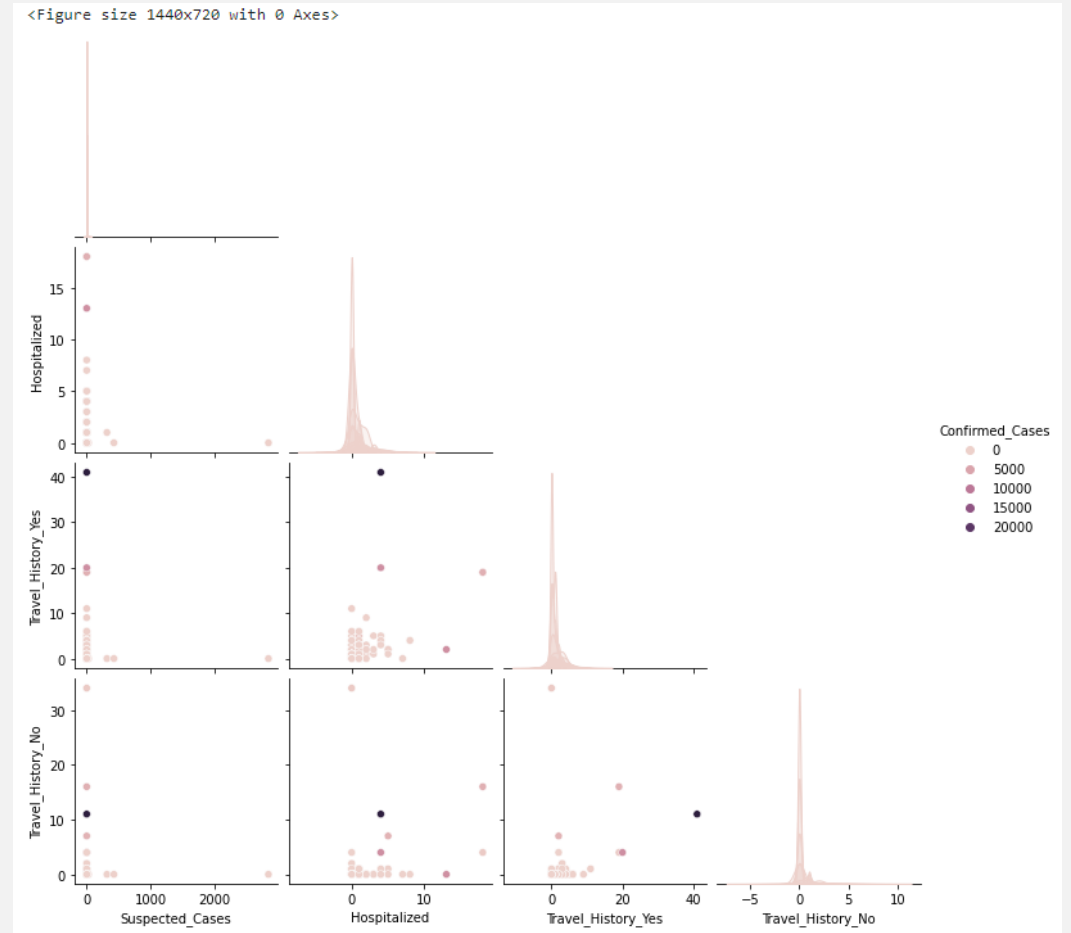
columns

```
plt.figure(figsize=(20,10));  
sns.pairplot(pair_df,corner=True,hue='Confirmed_Cases');
```

서로 다른 변수는 산점도를 그림
같은 변수는 히스토그램으로 표현

corner = True
대각선 기준으로 한쪽만 남김

hue 옵션
기존 pairplot에 hue 기준으로
나눠 표현



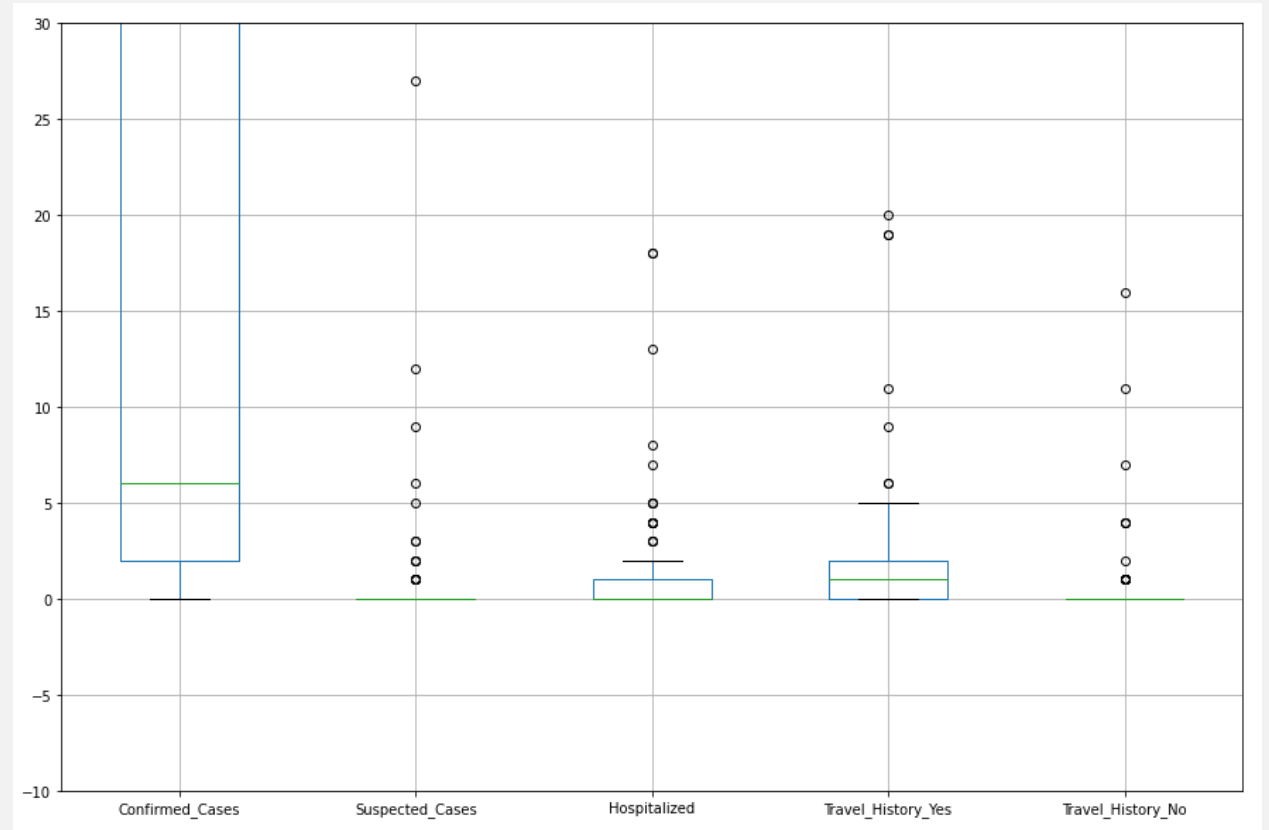
5. 특징 상관관계 분석

OUTLIER DETECTION

```
pair_df.boxplot(figsize=(15,10));  
plt.ylim(-10,30);
```

데이터에는 다른 것과는 거리가 먼
관측값으로 정의되는 이상값이 포함

- 대부분의 데이터가 0에 가까울 때
문에 특이치를 많이 볼 수 있음
- 이러한 특이치는 실제로 특이치가
아니지만 작은 데이터 집합으로 인
해 그렇게 보임.



5. 특징 상관관계 분석

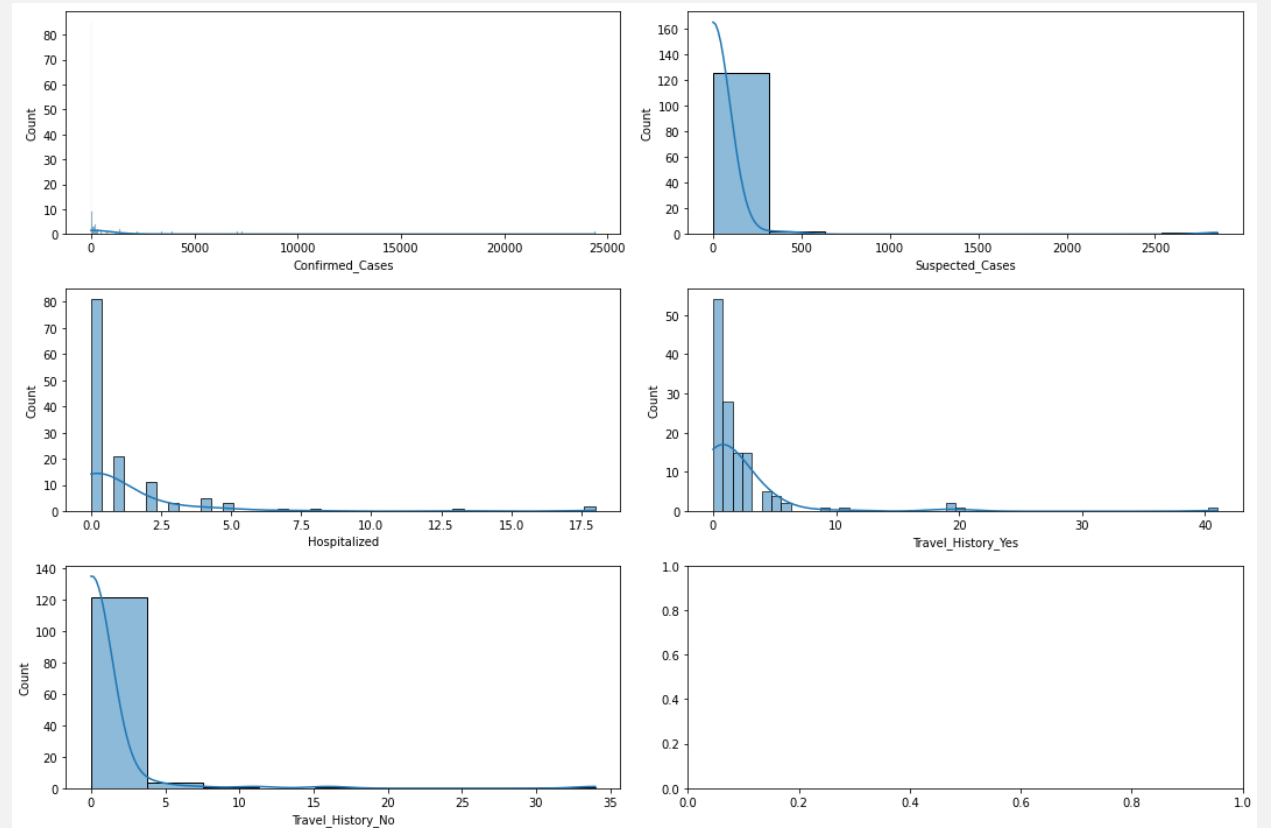
DISTRIBUTION CURVE

```
columns = df.columns.drop(['Country'])
```

```
fig, axes = plt.subplots(3,2,figsize=(15,10))  
for name, ax in zip(columns, axes.flatten()):  
    sns.histplot(x=name,kde=True, data=df, ax=ax)  
  
plt.tight_layout()
```

정규분포는 수집된 자료의 분포를
근사하는 데에 자주 사용

- 모든 열 대부분 값이 0이기 때문
에 거의 치우쳐 있음



6. 데이터 전처리 및 데이터 분할

NORMALIZATION

```
X = df.drop(['Country', 'Confirmed_Cases'], axis=1)
y = df['Confirmed_Cases']
```

int

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
scaler = preprocessing.StandardScaler()
```

→ 평균이 0이고 분산이 1인 정규 분포로 만들기

```
normalized_x_train = pd.DataFrame(scaler.fit_transform(X_train), columns = X_train.columns)
normalized_x_train.head()
```

| | Suspected_Cases | Hospitalized | Travel_History_Yes | Travel_History_No |
|---|-----------------|--------------|--------------------|-------------------|
| 0 | -0.124958 | 2.711188 | 0.371794 | -0.189668 |
| 1 | -0.124958 | -0.435317 | -0.434411 | -0.189668 |
| 2 | -0.124958 | -0.435317 | -0.232860 | -0.189668 |
| 3 | -0.124958 | 1.531248 | -0.031309 | 1.763910 |
| 4 | -0.124958 | 0.744622 | 0.573345 | -0.189668 |

6. 데이터 전처리 및 데이터 분할

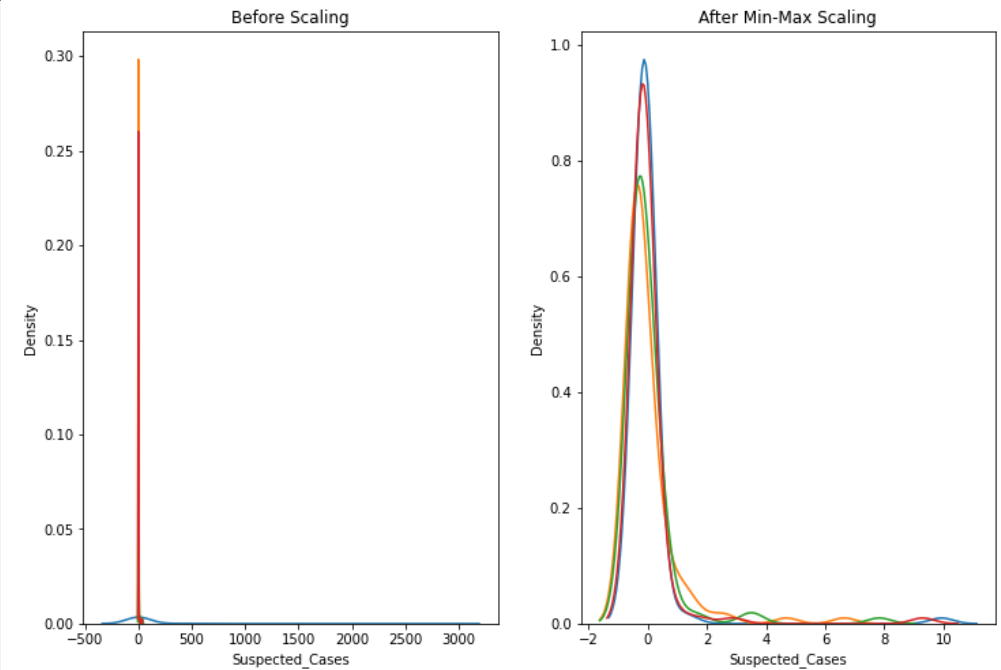
BEFORE VS AFTER SCALING

```
fig, (ob1, ob2) = plt.subplots(ncols=2, figsize=(12,8))
ob1.set_title('Before Scaling')
sns.kdeplot(X_train['Suspected_Cases'], ax=ob1)
sns.kdeplot(X_train['Hospitalized'], ax=ob1)
sns.kdeplot(X_train['Travel_History_Yes'], ax=ob1)
sns.kdeplot(X_train['Travel_History_No'], ax=ob1)

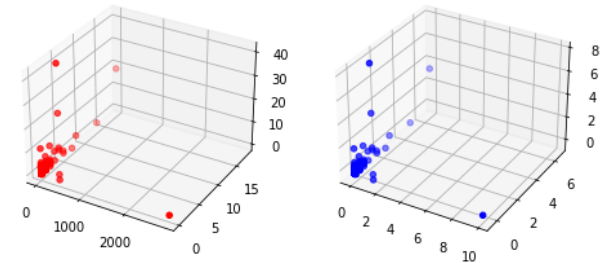
ob2.set_title('After Min-Max Scaling')
sns.kdeplot(normalized_x_train['Suspected_Cases'], ax=ob2)
sns.kdeplot(normalized_x_train['Hospitalized'], ax=ob2)
sns.kdeplot(normalized_x_train['Travel_History_Yes'], ax=ob2)
sns.kdeplot(normalized_x_train['Travel_History_No'], ax=ob2)
plt.show()

fig = plt.figure(figsize=(8,6))
ob3 = fig.add_subplot(121, projection='3d')
ob4 = fig.add_subplot(122, projection='3d')
ob3.scatter(X_train['Suspected_Cases'], X_train['Hospitalized'], X_train['Travel_History_Yes'], color='red')
ob4.scatter(normalized_x_train['Suspected_Cases'], normalized_x_train['Hospitalized'], normalized_x_train['Travel_History_Yes'], color='blue')
plt.show()
```

```
normalized_x_test = pd.DataFrame(scaler.transform(X_test), columns = X_test.columns)
```



- 0이 많아서 분포를 변경하는 데 스케일링이 데이터에 큰 변화 X



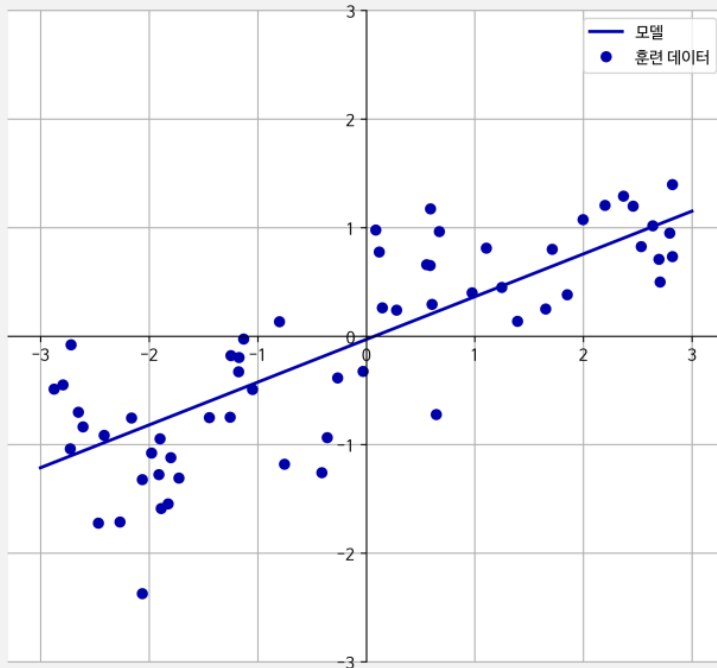
MODELLING

둘 이상의 변수 간의 관계를
보여주는 통계적 방법



- LinearRegression
- DecisionTreeRegressor
- RandomForestRegressor
- ElasticNet

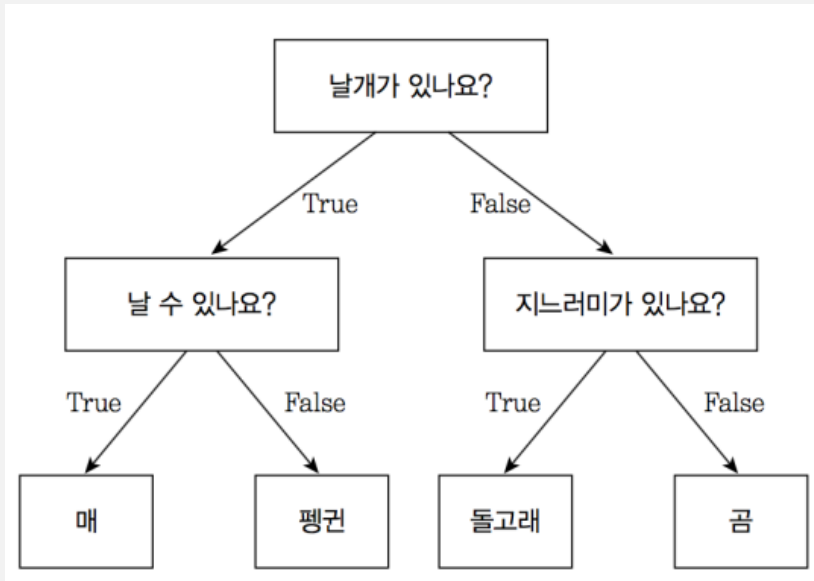
LINEAR REGRESSION



종속 변수 y 와 한 개 이상의 독립 변수 X 와의 선형 상관 관계를 모델링하는 회귀분석 기법

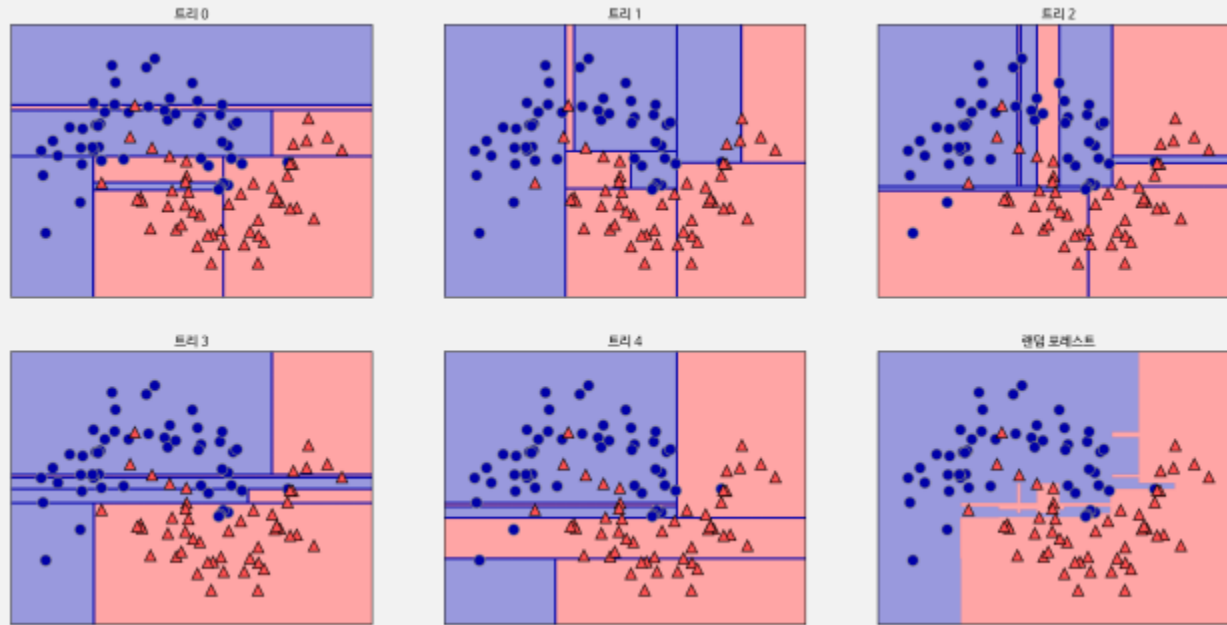
일차함수 $y = ax + b$ 를 기준으로 예측하는 모델

DECISION TREE REGRESSOR



- 결정 트리(decision tree)는 의사 결정 규칙과 그 결과를 트리 구조로 도식화
- 결정 트리를 학습한다는 것은 정답에 가장 빨리 도달하는 예/아니오 질문 목록을 학습한다는 뜻

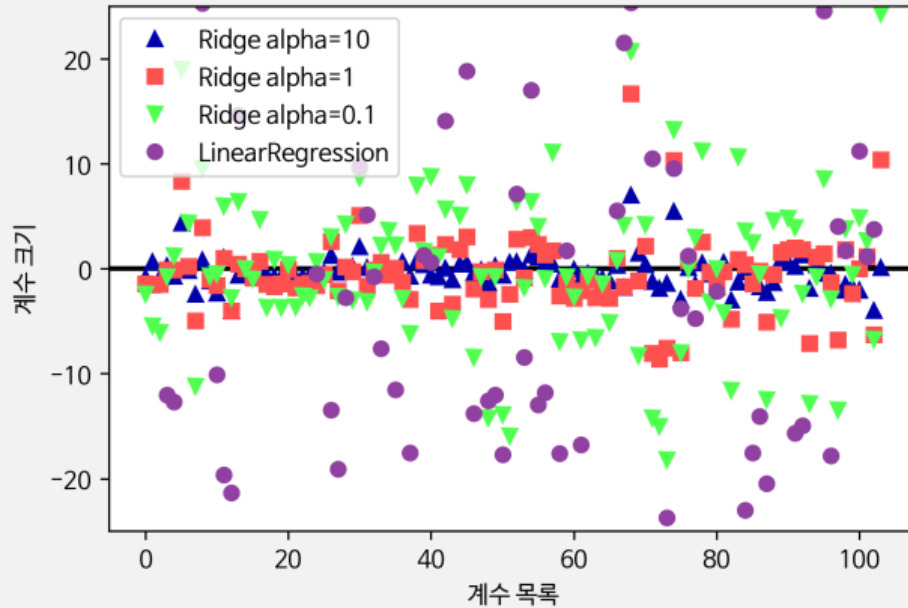
RANDOM FOREST REGRESSOR



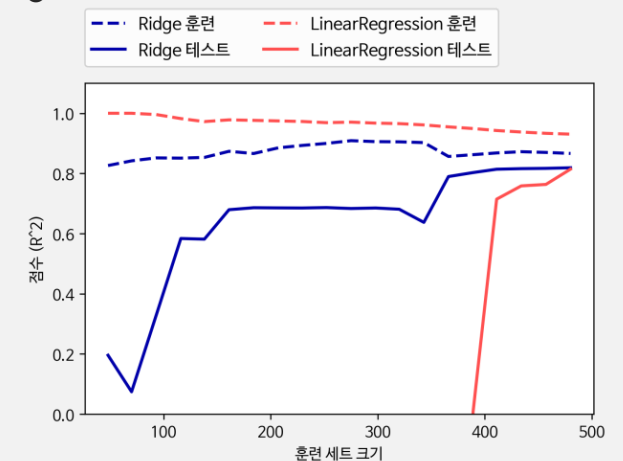
- 앙상블(ensemble)은 여러 머신러닝 모델을 연결하여 더 강력한 모델을 만드는 기법
- 결정트리의 경우 각 트리는 비교적 예측을 잘 할 수 있지만 데이터의 일부에 과도하게 적합하는 경향
- 잘 작동하되 서로 다른 방향으로 과대적합된 트리들 많이 만들어 그 결과를 평균냄

7. 머신러닝 알고리즘

RIDGE

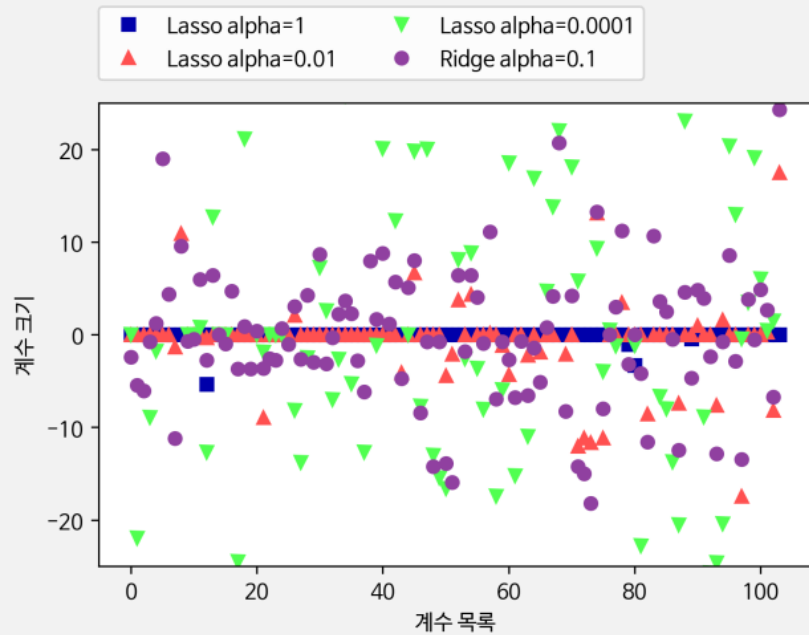


- Ridge 회귀에서의 가중치(w) 선택 (기울기)
- 가중치의 절댓값은 가능한 한 작게
- Regularization : 과대적합이 되지 않도록 모델을 강제로 제한
- L2 규제라 함



7. 머신러닝 알고리즘

LASSO



- 라쏘lasso도 계수를 0에 가깝게 만들려고 함 (L1 규제)
- 어떤 계수는 0이 되어 모델에서 완전히 제외되는 특성이 발생
- 입력 특성 중 일부만 사용하므로 쉽게 해석할 수 있는 모델 만들 수 있음

ELASTIC NET

- Lasso와 Ridge의 페널티를 결합
- 즉 L1 규제와 L2 규제를 적용

$$l_1 \times \sum_{j=1}^m w_j + \frac{1}{2} \times l_2 \times \sum_{j=1}^m w_j^2$$

ElasticNet 규제식

EVALUATING THE MODEL

```
d={}

def metric_scores(actual,predicted,name):

    """
    Function To evaluate and store the evaluation metric scores .
    Arguments to be passed:
    1. y_test
    2. y_pred
    3. name of the model
    """

    mae = mean_absolute_error(actual,predicted)
    mse = mean_squared_error(actual,predicted)
    mape = mean_absolute_percentage_error(actual,predicted)
    r2 = r2_score(actual,predicted)

    d[name]=[mae,mse,mape,r2]

    print('Mean Absolute Error is {:.3f}'.format(mae))
    print()
    print('Mean Squared Error is {:.3f}'.format(mse))
    print()
    print('Mean Absolute Percentage Error is {:.3f}'.format(mape))
    print()
    print('R Squared Error is {:.3f}'.format(r2))
    print()
```

모델 평가

- **평균 절대 오차** (MAE: mean_absolute_error:)
 - 실제 값과 예측 값의 차이 (Error) 를 절대 값으로 변환해 평균화
- **평균 제곱 오차** (MSE: mean_squared_error)
 - 실제 값과 예측 값의 차이 (Error) 를 제곱해 평균화
- **평균 절대비 오차** (MAPE: mean_absolute_percentage_error)
 - MAE 를 퍼센트로 변환

8. 모델 평가

#예측을 계산하고 모델을 평가하기 위해 만들어진 함수

```
def evaluate_model(x_test=None, y_test=None, model=None, name='Linear Regression'):
```

```
    """
```

```
    A function to make predictions, evaluate the model and plot the regression line.  
    The function requires the following values:
```

1. x_test
 2. y_test
 3. model object
 4. Name of the model
- ```
 """
```

```
 predicted = model.predict(x_test)
 actual=y_test
```

```
 metric_scores(actual, predicted, name)
 plt.figure(figsize=(12,8))
 plt.scatter(actual, predicted, c='hotpink')
```

```
 p1 = max(max(predicted), max(actual))
 p2 = min(min(predicted), min(actual))
 plt.plot([p1, p2], [p1, p2], '#000066')
 plt.xlabel('True Values', fontsize=15)
 plt.ylabel('Predictions', fontsize=15)
 plt.title(name, fontsize=30)
 plt.axis('equal')
 plt.show()
```

1. 분할한 데이터와 모델을 받아와
2. 예측을 하고
3. 평가를 하여
4. 결과를 시각화하는 함수

## 8. 모델 평가

# LINEAR REGRESSION

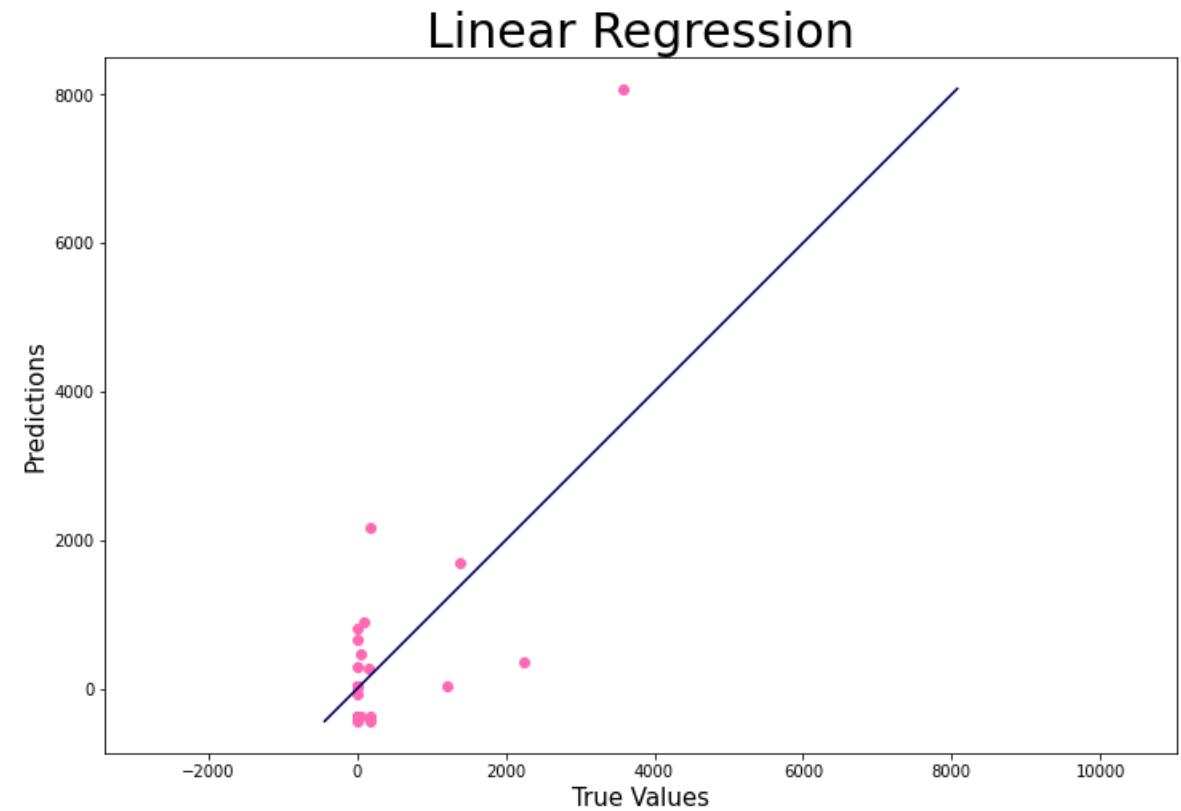
```
lr = LinearRegression()
lr.fit(normalized_x_train,y_train)
evaluate_model(x_test=normalized_x_test,y_test=y_test,model=lr,name='Linear Regression')
```

Mean Absolute Error is 675.932

Mean Squared Error is 1277093.813

Mean Absolute Percentage Error is 198342613389775488.000

R Squared Error is -0.833



## 8. 모델 평가

# DECISION TREE REGRESSOR

```
DR = DecisionTreeRegressor(random_state=42)

DR.fit(normalized_x_train, y_train)

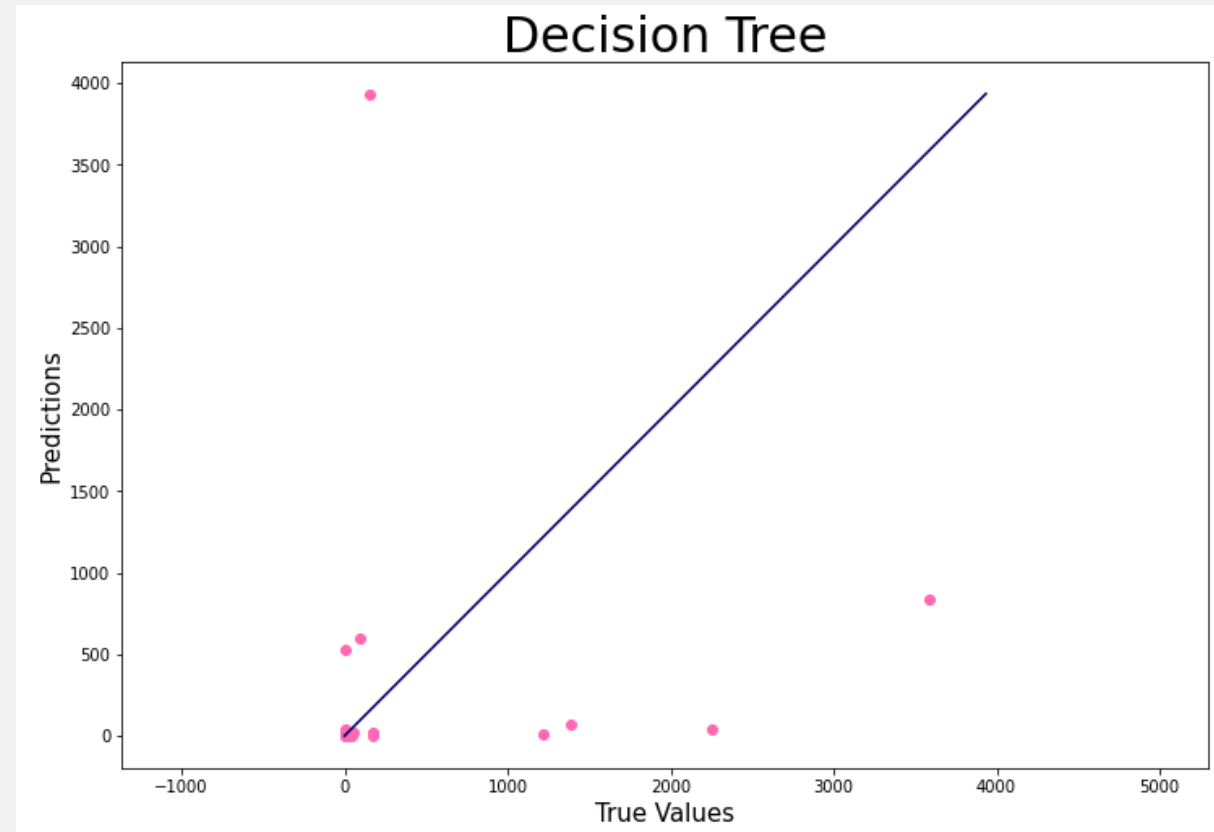
evaluate_model(x_test=normalized_x_test, y_test=y_test, model=DR, name='Decision Tree')
```

Mean Absolute Error is 501.984

Mean Squared Error is 1176941.613

Mean Absolute Percentage Error is 8634119995668894.000

R Squared Error is -0.689



## 8. 모델 평가

# RANDOM FOREST

```
model=RandomForestRegressor(n_estimators=100, verbose=4)

model.fit(normalized_x_train, y_train)

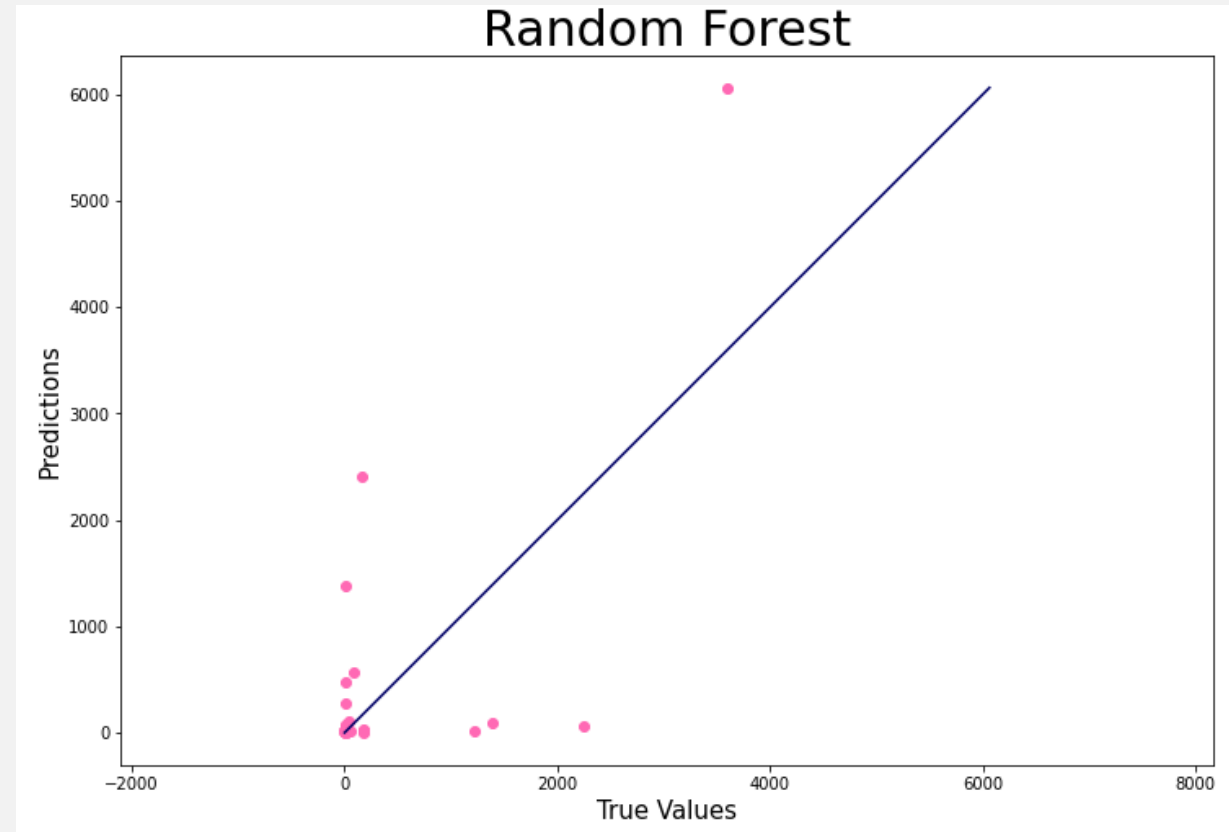
evaluate_model(x_test=normalized_x_test,y_test=y_test,model=model,name='Random Forest')
```

Mean Absolute Error is 490.289

Mean Squared Error is 830426.919

Mean Absolute Percentage Error is 8681993951803089.000

R Squared Error is -0.192



## 8. 모델 평가

# ELASTIC NET

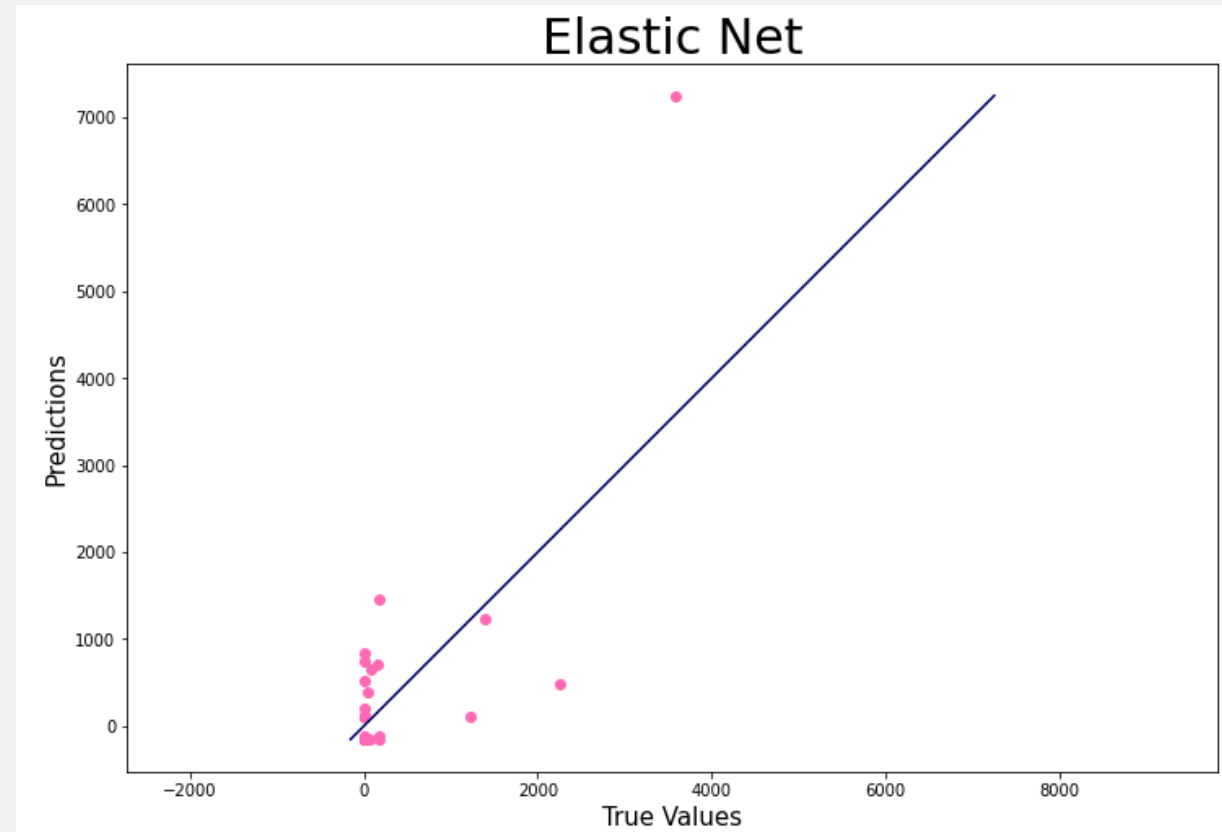
```
regr = ElasticNet()
regr.fit(normalized_x_train.values, y_train.values)
evaluate_model(x_test=normalized_x_test.values, y_test=y_test, model=regr, name='Elastic Net')
```

Mean Absolute Error is 539.828

Mean Squared Error is 852887.653

Mean Absolute Percentage Error is 79925416157109744.000

R Squared Error is -0.224



## 8. 모델 평가

# RESULT

```
results = pd.DataFrame(d,index = ['MAE','MSE','MAPE','R2'])

results
```

|      | Linear Regression | Decision Tree | Random Forest | Elastic Net   |
|------|-------------------|---------------|---------------|---------------|
| MAE  | 6.759316e+02      | 5.019842e+02  | 4.902889e+02  | 5.398278e+02  |
| MSE  | 1.277094e+06      | 1.176942e+06  | 8.304269e+05  | 8.528877e+05  |
| MAPE | 1.983426e+17      | 8.634120e+15  | 8.681994e+15  | 7.992542e+16  |
| R2   | -8.327027e-01     | -6.889785e-01 | -1.917101e-01 | -2.239426e-01 |

모델 결과가 좋지 않음

이는 확진 및 설문 응답의 경우가 부족하다는 의미

따라서 확진 케이스가 늘어날 수록, 데이터가  
늘어날 수록 정확성은 개선



## CONCLUDING REMARKS

- 머신러닝 알고리즘에 대하여 찾아보면서 수학적 기반으로 구현하는 것을 보았습니다. 컴퓨터 과학자들이 고민하고 또 고민하여 수학적식을 도출한 것일 것인데 이를 구현하여 패키지에 손쉽게 사용할 수 있도록 만든 것에 놀랐습니다.
- 이 사례에 대한 좋은 결과를 얻을 수 없었지만 이 과정에서 머신러닝 알고리즘 같이 많은 것을 배울 수 있었음에 감사드립니다.

# CITATION

- <https://numpy.org/>
- <https://pandas.pydata.org/>
- <https://plotly.com/python-api-reference/index.html>
- <https://scikit-learn.org/stable/modules/classes.html>
  
- <https://aws.amazon.com/ko/what-is/linear-regression/>
- <https://tensorflow.blog/>
  
- **이미지** Freepik/drobotdean, Freepik/jcomp, Freepik/starline
- <https://www.who.int/>

Thank you