Untitled23.ipynb

import os

from sklearn.metrics import r2\_score

import pandas as pd

import glob

import plotly.express as px

import sqlite3

import numpy as np # linear algebra

import pandas as pd # data processing, CSV file I/O (e.g. pd.read\_csv)

import matplotlib.pyplot as plt

from pandas import Timestamp as pd\_datetime

import math, time

import itertools

import datetime

# from operator import itemgetter

from math import sqrt

import requests

name=glob.glob(r"/content/air2/\*")

a=pd.read\_csv(name[0],encoding='big5')

for i in name[1:]:

  temp=pd.read\_csv(i,encoding='big5')

  a=pd.concat([a,temp])

def Pearson\_correlation(X,Y):

  if len(X)==len(Y):

    Sum\_xy = sum((X-X.mean())\*(Y-Y.mean()))

    Sum\_x\_squared = sum((X-X.mean())\*\*2)

    Sum\_y\_squared = sum((Y-Y.mean())\*\*2)

    corr = Sum\_xy / np.sqrt(Sum\_x\_squared \* Sum\_y\_squared)

    return corr

  else:

    return 0

a=a.dropna(subset=[ 'so2','co','o3','o3\_8hr', 'pm10', 'pm2.5', 'no2', 'nox', 'no'])

# nox  pm2.5 pm10

a['datetime']=a['publishtime'].apply(pd.to\_datetime)

temp\_sitename=a[a['sitename']=='左營']

temp\_sitename=temp\_sitename.sort\_values(by=['datetime'])

R=[]

pollutant=[]

date\_list=[]

start=temp\_sitename.iloc[0,24]

while(start<temp\_sitename.iloc[-1,24]):

  temp\_r=temp\_sitename[(temp\_sitename['datetime']>start)&(temp\_sitename['datetime']<start+datetime.timedelta(days=6))]

  if not temp\_r.empty:

    R.append(Pearson\_correlation(temp\_r['so2'],temp\_r['aqi']))

    pollutant.append('so2')

    date\_list.append(temp\_r.iloc[0,16])

    start=start+datetime.timedelta(days=1)

  else:

    start=start+datetime.timedelta(days=1)

start=temp\_sitename.iloc[0,24]

while(start<temp\_sitename.iloc[-1,24]):

  temp\_r=temp\_sitename[(temp\_sitename['datetime']>start)&(temp\_sitename['datetime']<start+datetime.timedelta(days=6))]

  if not temp\_r.empty:

    R.append(Pearson\_correlation(temp\_r['co'],temp\_r['aqi']))

    # site.append(temp\_sitename.iloc[0,0])

    pollutant.append('co')

    date\_list.append(temp\_r.iloc[0,16])

    start=start+datetime.timedelta(days=1)

  else:

    start=start+datetime.timedelta(days=1)

start=temp\_sitename.iloc[0,24]

while(start<temp\_sitename.iloc[-1,24]):

  temp\_r=temp\_sitename[(temp\_sitename['datetime']>start)&(temp\_sitename['datetime']<start+datetime.timedelta(days=6))]

  if not temp\_r.empty:

    R.append(Pearson\_correlation(temp\_r['o3'],temp\_r['aqi']))

    # site.append(temp\_sitename.iloc[0,0])

    pollutant.append('o3')

    date\_list.append(temp\_r.iloc[0,16])

    start=start+datetime.timedelta(days=1)

  else:

    start=start+datetime.timedelta(days=1)

start=temp\_sitename.iloc[0,24]

while(start<temp\_sitename.iloc[-1,24]):

  temp\_r=temp\_sitename[(temp\_sitename['datetime']>start)&(temp\_sitename['datetime']<start+datetime.timedelta(days=6))]

  if not temp\_r.empty:

    R.append(Pearson\_correlation(temp\_r['pm10'],temp\_r['aqi']))

    # site.append(temp\_sitename.iloc[0,0])

    pollutant.append('pm10')

    date\_list.append(temp\_r.iloc[0,16])

    start=start+datetime.timedelta(days=1)

  else:

    start=start+datetime.timedelta(days=1)

start=temp\_sitename.iloc[0,24]

while(start<temp\_sitename.iloc[-1,24]):

  temp\_r=temp\_sitename[(temp\_sitename['datetime']>start)&(temp\_sitename['datetime']<start+datetime.timedelta(days=6))]

  if not temp\_r.empty:

    R.append(Pearson\_correlation(temp\_r['pm2.5'],temp\_r['aqi']))

    # site.append(temp\_sitename.iloc[0,0])

    pollutant.append('pm2.5')

    date\_list.append(temp\_r.iloc[0,16])

    start=start+datetime.timedelta(days=1)

  else:

    start=start+datetime.timedelta(days=1)

start=temp\_sitename.iloc[0,24]

while(start<temp\_sitename.iloc[-1,24]):

  temp\_r=temp\_sitename[(temp\_sitename['datetime']>start)&(temp\_sitename['datetime']<start+datetime.timedelta(days=6))]

  if not temp\_r.empty:

    R.append(Pearson\_correlation(temp\_r['nox'],temp\_r['aqi']))

    # site.append(temp\_sitename.iloc[0,0])

    pollutant.append('nox')

    date\_list.append(temp\_r.iloc[0,16])

    start=start+datetime.timedelta(days=1)

  else:

    start=start+datetime.timedelta(days=1)

dataframe=pd.DataFrame({'p':pollutant,"date\_list":date\_list,'R':R})

# print(dataframe)

fig = px.line(dataframe, x="date\_list", y="R", color='p')

fig.show()

untitled0

!pip install --upgrade descartes

!apt install libspatialindex-dev

!pip install rtree

!pip install geopandas

!pip install adjacency

import os

from google.colab import drive

drive.mount('/content/drive')

import numpy as np

import matplotlib.pyplot as plt

import matplotlib as mpl

import geopandas as gpd

import pandas as pd

import shapely

from descartes import PolygonPatch

cancer\_death = pd.read\_csv('/content/drive/MyDrive/data/cancer105.csv')

county\_code = pd.read\_csv('/content/drive/MyDrive/data/County\_Code\_100.csv')

cancer\_death\_new = pd.merge(left=cancer\_death, right=county\_code, left\_on='county',right\_on='county')

cancer\_code = pd.read\_csv('/content/drive/MyDrive/data/CancerCode\_97.csv')

cancer\_code=cancer\_code.drop(['ICD-10'], axis=1)

cancer\_death\_new1=pd.merge(left=cancer\_death\_new, right=cancer\_code, left\_on='cause',right\_on='97年以後cause')

cancer\_death\_new1['city'] = cancer\_death\_new1['100年~鄉鎮市區'].copy(deep=True)

cancer\_death\_new1['city'] = cancer\_death\_new1['city'].apply(lambda t: t[:3])

cancer\_death\_new1['district'] = cancer\_death\_new1['100年~鄉鎮市區'].copy(deep=True)

cancer\_death\_new1['district'] = cancer\_death\_new1['district'].apply(lambda t: t[3:])

Kaohsiung\_cancer\_death=cancer\_death\_new1[(cancer\_death\_new1['city']=='高雄市')]

town\_shp = gpd.read\_file('/content/drive/MyDrive/TW97 shp/TOWN\_MOI\_1081121.shp',encoding='utf-8') #呼叫台灣圖資

town\_shp = town\_shp.to\_crs(epsg=3826)

North\_town\_shp=town\_shp[(town\_shp['COUNTYNAME']=='高雄市')]

North\_town\_shp.head()

Kaohsiung\_cancer\_death\_M = Kaohsiung\_cancer\_death[Kaohsiung\_cancer\_death['sex']==1]

Kaohsiung\_cancer\_death\_M=Kaohsiung\_cancer\_death\_M.drop(['cause','county','sex','97年以後cause','city'],axis=1)

Kaohsiung\_Colorectal\_cancer\_M=Kaohsiung\_cancer\_death\_M[Kaohsiung\_cancer\_death\_M['死因分類']=='氣管、支氣管和肺癌']

Kaohsiung\_Colorectal\_cancer\_M=Kaohsiung\_Colorectal\_cancer\_M.drop(Kaohsiung\_Colorectal\_cancer\_M.columns[[0,1,5]], axis=1)

Kaohsiung\_Colorectal\_cancer\_M\_sum=Kaohsiung\_Colorectal\_cancer\_M.groupby('100年~鄉鎮市區').sum()

Kaohsiung\_Colorectal\_cancer\_M\_sum['District']=Kaohsiung\_Colorectal\_cancer\_M\_sum.index

Kaohsiung\_Colorectal\_cancer\_M\_sum['District']=Kaohsiung\_Colorectal\_cancer\_M\_sum['District'].apply(lambda t: t[3:])

Kaohsiung\_Colorectal\_cancer\_M\_sum.head()

Kaohsiung\_cancer\_death\_M[Kaohsiung\_cancer\_death\_M['死因分類'].str.contains("肺")]

Kaohsiung\_Colorectal\_cancer\_M\_sum.sort\_values(by='N',ascending=False)

left = North\_town\_shp.set\_index('TOWNNAME') #這個應該是 geopandas 的 GeoDataFrame 物件

right = Kaohsiung\_Colorectal\_cancer\_M\_sum.set\_index('District') #這應該是 pandas 的 DataFrame 物件

Kaohsiung\_ColorectalCancerDistrict = left.join(right)

Kaohsiung\_ColorectalCancerDistrict['N'] = Kaohsiung\_ColorectalCancerDistrict['N'].replace(np.nan, 0)#用取代的功能將nan補0

fig, ax = plt.subplots(1,1, figsize=(10,10)) # 定義畫布大小，其中(1,1)代表一行一列，即為一組座標

Kaohsiung\_ColorectalCancerDistrict.plot(column='N',cmap='Reds', ax=ax)

ax.set\_xlim([140000, 270000]) # 這行設定 X 範圍

ax.set\_ylim([ 2480000, 2600000]) # 這行設定 Y 範圍

plt.tight\_layout() # 加這行去把圖盡可能推展到 10,15 的版面

TCN

!pip install wget

!pip install keras-tcn

import wget

import matplotlib.pyplot as plt

import numpy as np

import pandas as pd

from tensorflow.keras import Input, Model

from tensorflow.keras.layers import Dense

from tqdm.notebook import tqdm

import plotly.express as px

from tcn import TCN

from glob import glob

from keras import backend as K

import keras

from tensorflow.keras.layers import Input, LSTM, Dense

import os

from google.colab import drive

drive.mount('/content/drive')

air = pd.read\_csv('/content/drive/MyDrive/ppt/air\_total.csv')

air=air.drop(['pollutant','status','nox'],axis=1)

air=air[air['sitename']=='臺東'].dropna()

air=air.sort\_values(by=['publishtime'])

air=air.reset\_index(inplace=False,drop=True)

lookback\_window = 12

air\_v=air.iloc[:,2:11]

air\_v.iloc[:14,:]

air = pd.read\_csv('/content/drive/MyDrive/ppt/air\_total.csv')

air=air.drop(['pollutant','status'],axis=1)

air=air[air['sitename']=='臺東'].dropna()

air=air.sort\_values(by=['publishtime'])

air=air.reset\_index(inplace=False,drop=True)

lookback\_window = 12

air\_v=air.iloc[:,2:11]

air\_v1=air\_v.values

x = []

y = []

for i in tqdm(range(lookback\_window, len(air\_v1))):

  x.append(air\_v1[i - lookback\_window:i,1:])

  y.append(air\_v1[i,0])

def root\_mean\_squared\_error(y\_true, y\_pred):

  return K.sqrt(K.mean(K.square(y\_pred - y\_true), axis=-1))

x = np.array(x)

x = np.asarray(x).astype('float32')

y = np.array(y)

y = np.asarray(y).astype('float32')

print(x.shape)

print(y.shape)

lookback\_window=12

i = Input(shape=(lookback\_window, 8))

m = LSTM(units=100)(i)

m = Dense(1, activation='linear')(m)

model = Model(inputs=[i], outputs=[m])

# model.summary()

model.compile('adam',loss = root\_mean\_squared\_error)

h=model.fit(x[:round(len(x)\*0.7)], y[:round(len(x)\*0.7)], epochs=100,batch\_size=30,validation\_split = 0.1)

keras.utils.plot\_model(model, to\_file='/content/1.jpg', show\_shapes=True)

plt.plot(h.history['loss'])

plt.plot(h.history['val\_loss'])

plt.title('model accuracy')

plt.ylabel('loss')

plt.xlabel('epoch')

plt.legend(['train', 'val'], loc='upper left')

plt.show()

# temp=pd.DataFrame({'time':list(air['publishtime'][:-lookback\_window]),'pre':predict,'real':y})

# fig = px.line(temp, x="time", y="aqi",color='c')

# LSTM

predict = model.predict(x)

import plotly.graph\_objects as go

fig = go.Figure()

fig.add\_trace(go.Scatter(x=list(air['publishtime'][:-lookback\_window]), y=predict.reshape(predict.shape[0],).tolist()[:round(len(x)\*0.7)],

                    mode='lines',

                    name='訓練'))

fig.add\_trace(go.Scatter(x=list(air['publishtime'][:-lookback\_window]), y=y,

                    mode='lines+markers',

                    name='實際'))

fig.add\_trace(go.Scatter(x=list(air['publishtime'][:-lookback\_window])[round(len(x)\*0.7):], y=predict.reshape(predict.shape[0],).tolist()[round(len(x)\*0.7):],

                    mode='lines',

                    name='預測'))

fig.update\_layout(

            title={

            'text' : '',

            'x':0.5,

            'xanchor': 'center'

        })

fig.show()

from sklearn.metrics import mean\_squared\_error

import math

MSE = mean\_squared\_error(predict, y)

RMSE = math.sqrt(MSE)

print("LSTM Root Mean Square Error:", RMSE)

import seaborn as sns

import warnings

warnings.filterwarnings('ignore')

df=pd.read\_csv('/content/drive/MyDrive/ppt/air\_total.csv')

st1='花蓮'

df=df[df['sitename']==st1]

binary\_columns = df[['so2','co', 'o3', 'o3\_8hr', 'pm10','pm2.5', 'no2', 'no']]

correlation\_matrix = binary\_columns.corr()

# plt.figure(figsize=(18, 18))

sns.heatmap(correlation\_matrix, annot=True, cmap='Reds', linewidths=0.5)

cancer

import os

from google.colab import drive

drive.mount('/content/drive')

mport pandas as pd

import plotly.express as px

import glob

from sklearn.linear\_model import LinearRegression

import numpy as np

import matplotlib.pyplot as plt

from plotly.subplots import make\_subplots

add=glob.glob("/content/drive/MyDrive/ppt/去年資料/\*.csv")

a1=pd.read\_csv(add[0],encoding='big5')

for i in add[1:]:

   temp=pd.read\_csv(i,encoding='big5')

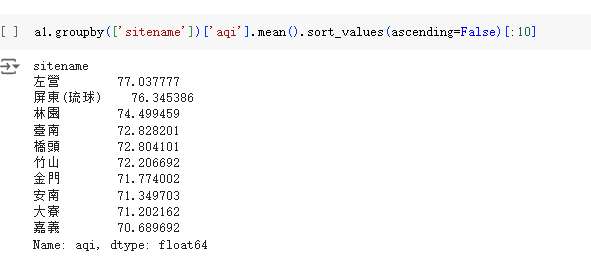
   a1=pd.concat([a1,temp])

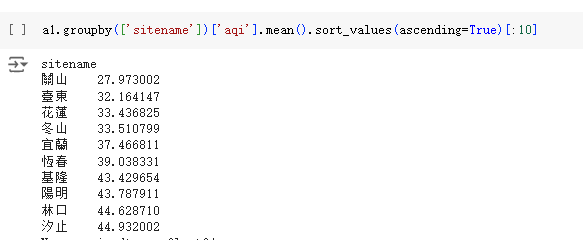
  #  a1=pd.read\_csv("/content/drive/MyDrive/data/total.csv",encoding='big5')

a1.groupby(['sitename'])['aqi'].sum().sort\_values(ascending=False)

a1.groupby(['sitename'])['aqi'].sum().sort\_values(ascending=False).to\_csv("aqi\_sum.csv",encoding='big5')

a1.groupby(['sitename'])['aqi'].mean().sort\_values(ascending=False).to\_csv("aqi\_mean.csv",encoding='big5')





這兩部份要介紹用什麼時間資料跑的結果，日期:2023/9/12到2024/4/12

import pandas as pd

import plotly.express as px

import glob

from sklearn.linear\_model import LinearRegression

import numpy as np

import matplotlib.pyplot as plt

from plotly.subplots import make\_subplots

a=glob.glob("/content/drive/MyDrive/data/癌症死因統計/can\*.csv")

df=pd.read\_csv(a[0])

for i in a[1:]:

    temp=pd.read\_csv(i)

    df=pd.concat([df,temp])

df=df.sort\_values(by='year')

df=df.reset\_index(inplace=False,drop=True)

total=pd.read\_csv("/content/drive/MyDrive/data/total.csv",encoding='big5')

def re1(a):

  a=a.replace(',','').replace('\u3000','')

  return a

total['number']=total['number'].apply(re1)

total['area']=total['area'].apply(re1)

total['number']=total['number'].astype('int')

total['area']=total['area'].replace("臺北縣",'新北市')

import plotly.graph\_objects as go

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error, mean\_absolute\_error

import numpy as np

import pandas as pd

data2 = pd.read\_csv("/content/drive/MyDrive/data/KH.csv", encoding='big5')

fig = go.Figure()

rmse\_list = []

mse\_list = []

mae\_list = []

r\_squared\_list = []

adjusted\_r\_squared\_list = []

# 定义颜色列表

colors = ['#880000', '#000088', '#888800', '#008888', '#880088']

def lighten\_color(color, factor=0.7):

    """

    Generate a lighter version of the given color.

    :param color: The color in hexadecimal format.

    :param factor: The factor by which to lighten the color (default is 0.7).

    :return: The lighter version of the color in hexadecimal format.

    """

    if color.startswith('#'):

        color = color[1:]

    r = int(color[0:2], 16)

    g = int(color[2:4], 16)

    b = int(color[4:6], 16)

    r = int(r + (255 - r) \* factor)

    g = int(g + (255 - g) \* factor)

    b = int(b + (255 - b) \* factor)

    return "#{:02x}{:02x}{:02x}".format(r, g, b)

for index, i in enumerate(data2['area'].unique()):

    model = LinearRegression()

    temp = data2[data2['area'] == i]

    X = temp.iloc[:, [0, 4]]

    y = temp.iloc[:, 1]

    model.fit(X, y)

    print(i)

    print(model.coef\_)

    print(model.intercept\_)

    y\_pred = model.predict(X)

    temp['pred'] = y\_pred

    rmse = np.sqrt(mean\_squared\_error(y, y\_pred))

    rmse\_list.append(rmse)

    print("RMSE for each area:", rmse\_list)

    # Calculating MSE

    mse = mean\_squared\_error(y, y\_pred)

    mse\_list.append(mse)

    print("MSE for", i, ":", mse)

    # Calculating MAE

    mae = mean\_absolute\_error(y, y\_pred)

    mae\_list.append(mae)

    print("MAE for", i, ":", mae)

    # Calculating R-squared

    ssr = np.sum((y\_pred - np.mean(y)) \*\* 2)

    sst = np.sum((y - np.mean(y)) \*\* 2)

    r\_squared = ssr / sst

    r\_squared\_list.append(r\_squared)

    print("R-squared for", i, ":", r\_squared)

    # Calculating Adjusted R-squared

    n = len(y)

    p = X.shape[1]

    adjusted\_r\_squared = 1 - (1 - r\_squared) \* (n - 1) / (n - p - 1)

    adjusted\_r\_squared\_list.append(adjusted\_r\_squared)

    print("Adjusted R-squared for", i, ":", adjusted\_r\_squared)

    color = colors[index % len(colors)]  # 使用模运算确保循环颜色选择

    fig.add\_trace(go.Scatter(x=temp.year, y=temp['N'],

                         mode='lines+markers',

                         name=i + '測站' + '(\_real)',

                         line=dict(color=color)))

    fig.add\_trace(go.Scatter(x=temp.year, y=temp['pred'],

                         mode='lines+markers',

                         name=i + '測站' + '(\_predict)',

                         line=dict(color=lighten\_color(color, 0.7))))

# 使用update\_layout更新圖例文字大小

fig.update\_layout(legend=dict(

    font=dict(

        family="Courier",

        size=24,  # 將圖例文字大小設為18

    )

))

fig.update\_layout(

    title={

        'text': "民國87年到民國111年五個空氣汙染嚴重地區肺癌死亡人口數趨勢訓練圖",

        'x': 0.5,

        'y': 0.95,

        'xanchor': 'center',

        'yanchor': 'top',

        'font': dict(

            size=30

        )

    },

   xaxis\_title={

        'text': "年份",

        'font': dict(

            size=30  # 設定X軸標題文字大小為 18

        )

    },

    yaxis\_title={

        'text': "肺癌死亡人口數",

        'font': dict(

            size=30  # 設定Y軸標題文字大小為 18

        )

    }

)

fig.show()

import plotly.graph\_objects as go

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error, mean\_absolute\_error

import numpy as np

import pandas as pd

data2 = pd.read\_csv("/content/drive/MyDrive/data/Hualien.csv", encoding='big5')

fig = go.Figure()

rmse\_list = []

mse\_list = []

mae\_list = []

r\_squared\_list = []

adjusted\_r\_squared\_list = []

# 定义颜色列表

colors = ['#880000', '#000088', '#888800', '#008888', '#880088']

def lighten\_color(color, factor=0.7):

    """

    Generate a lighter version of the given color.

    :param color: The color in hexadecimal format.

    :param factor: The factor by which to lighten the color (default is 0.7).

    :return: The lighter version of the color in hexadecimal format.

    """

    if color.startswith('#'):

        color = color[1:]

    r = int(color[0:2], 16)

    g = int(color[2:4], 16)

    b = int(color[4:6], 16)

    r = int(r + (255 - r) \* factor)

    g = int(g + (255 - g) \* factor)

    b = int(b + (255 - b) \* factor)

    return "#{:02x}{:02x}{:02x}".format(r, g, b)

for index, i in enumerate(data2['area'].unique()):

    model = LinearRegression()

    temp = data2[data2['area'] == i]

    X = temp.iloc[:, [0, 4]]

    y = temp.iloc[:, 1]

    model.fit(X, y)

    print(i)

    print(model.coef\_)

    print(model.intercept\_)

    y\_pred = model.predict(X)

    temp['pred'] = y\_pred

    rmse = np.sqrt(mean\_squared\_error(y, y\_pred))

    rmse\_list.append(rmse)

    print("RMSE for each area:", rmse\_list)

    # Calculating MSE

    mse = mean\_squared\_error(y, y\_pred)

    mse\_list.append(mse)

    print("MSE for", i, ":", mse)

    # Calculating MAE

    mae = mean\_absolute\_error(y, y\_pred)

    mae\_list.append(mae)

    print("MAE for", i, ":", mae)

    # Calculating R-squared

    ssr = np.sum((y\_pred - np.mean(y)) \*\* 2)

    sst = np.sum((y - np.mean(y)) \*\* 2)

    r\_squared = ssr / sst

    r\_squared\_list.append(r\_squared)

    print("R-squared for", i, ":", r\_squared)

    # Calculating Adjusted R-squared

    n = len(y)

    p = X.shape[1]

    adjusted\_r\_squared = 1 - (1 - r\_squared) \* (n - 1) / (n - p - 1)

    adjusted\_r\_squared\_list.append(adjusted\_r\_squared)

    print("Adjusted R-squared for", i, ":", adjusted\_r\_squared)

    color = colors[index % len(colors)]  # 使用模运算确保循环颜色选择

    fig.add\_trace(go.Scatter(x=temp.year, y=temp['N'],

                         mode='lines+markers',

                         name=i + '測站' + '(\_real)',

                         line=dict(color=color)))

    fig.add\_trace(go.Scatter(x=temp.year, y=temp['pred'],

                         mode='lines+markers',

                         name=i + '測站' + '(\_predict)',

                         line=dict(color=lighten\_color(color, 0.7))))

# 使用update\_layout更新圖例文字大小

fig.update\_layout(legend=dict(

    font=dict(

        family="Courier",

        size=24,  # 將圖例文字大小設為18

    )

))

fig.update\_layout(

    title={

        'text': "民國87年到民國111年五個空氣汙染不嚴重地區死亡人口數趨勢訓練圖",

        'x': 0.5,

        'y': 0.95,

        'xanchor': 'center',

        'yanchor': 'top',

        'font': dict(

            size=30

        )

    },

   xaxis\_title={

        'text': "年份",

        'font': dict(

            size=30  # 設定X軸標題文字大小為 18

        )

    },

    yaxis\_title={

        'text': "肺癌死亡人口數",

        'font': dict(

            size=30  # 設定Y軸標題文字大小為 18

        )

    }

)

fig.show()