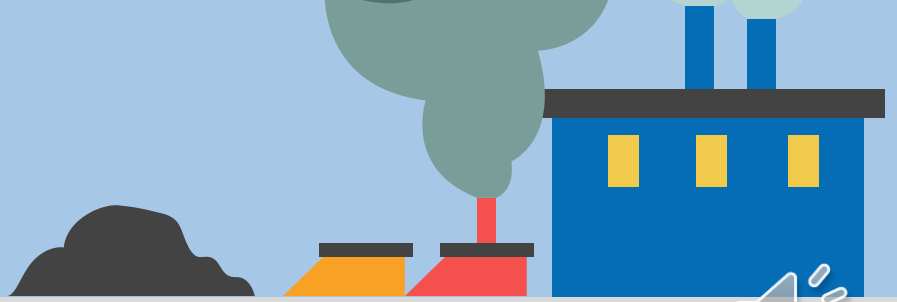
Lab Group SC3
Tutor: Liu Di Kai
Team 6

Estimating Frequency of Natural Disasters with Machine Learning



Foo Jen Sean, Jeffrey Lim Yi Ren,
Karishein Chandran

Data Analytics Pipeline

01 Introduction

- Problem Definition
- Problem Analysis and Breakdown
- Dataset Sourcing

02 Exploratory Data Analysis

- Analysis of Data
- Visualisation of Data
- Data Cleaning
- Merging of Datasets

03 Machine Learning

- Machine Learning Process
- Parameters Tuning
- Prediction and Modelling
- Model Validation

04 Conclusion

- Data-driven Insights and Recommendations
- Interesting Observations
- Challenges Faced and Learning Outcomes
- Conclusion

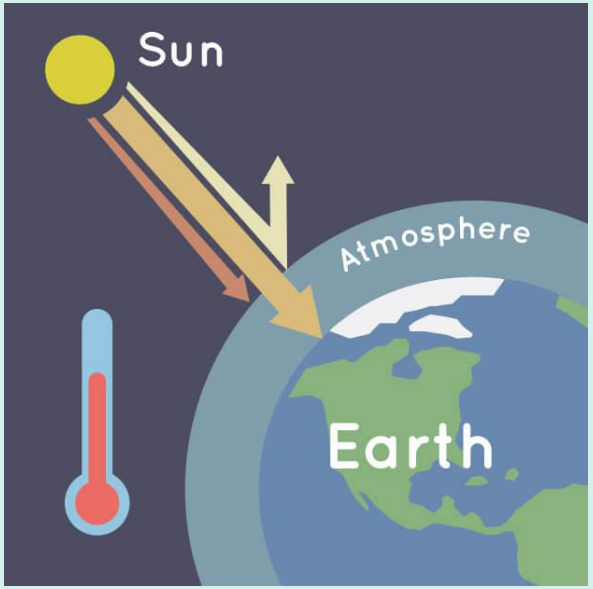
Problem Definition

Does the increase of greenhouse gases in our climate contribute to a more immediate and imminent problem such as Natural Disasters?

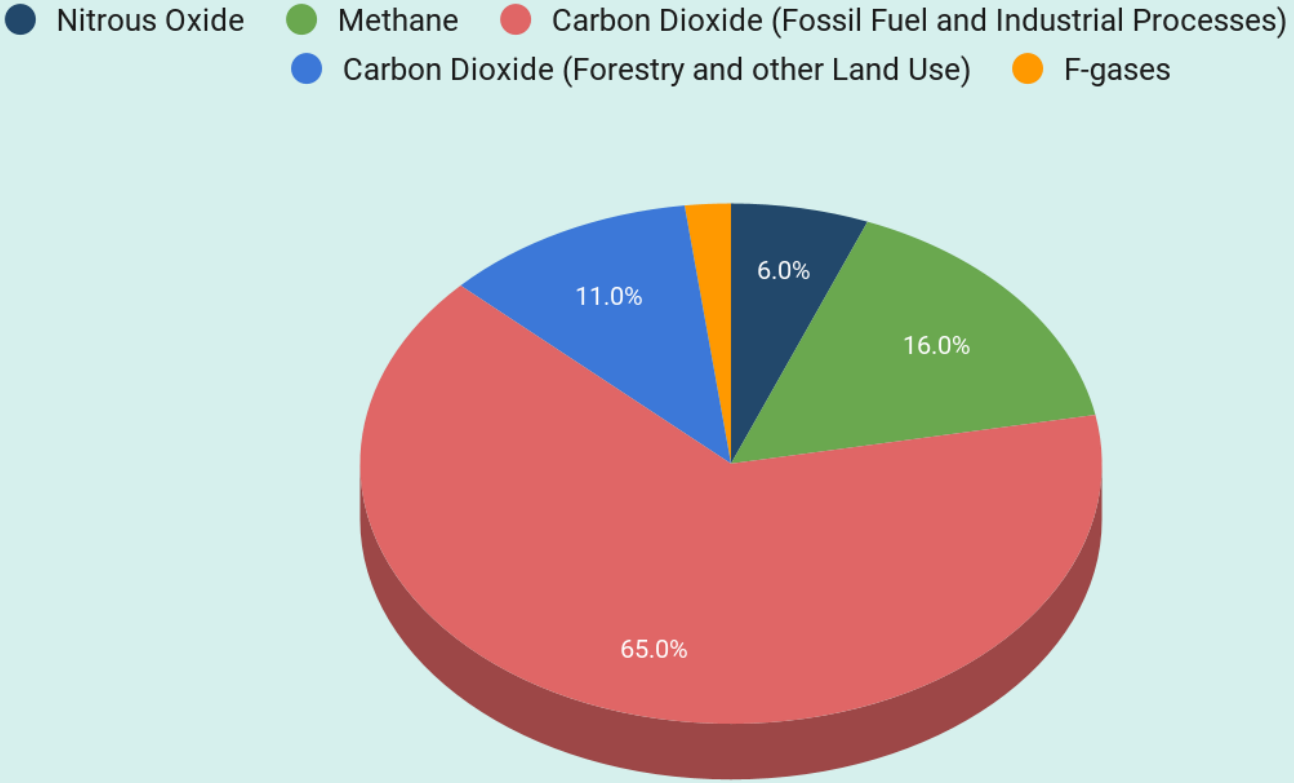
Greenhouse Gases

Gases that trap heat in the atmosphere

- **Advantages:** Greenhouse effect
- **Disadvantage:** Global warming



Greenhouse Gases



https://www.epa.gov/ghgemissions/global-greenhouse-gas-emissions-data

Why is Global Warming an issue ?

Loss of Agricultural Productivity

Change in Precipitation Patterns

Rising Sea Level

Threats to Biodiversity

Natural Disasters

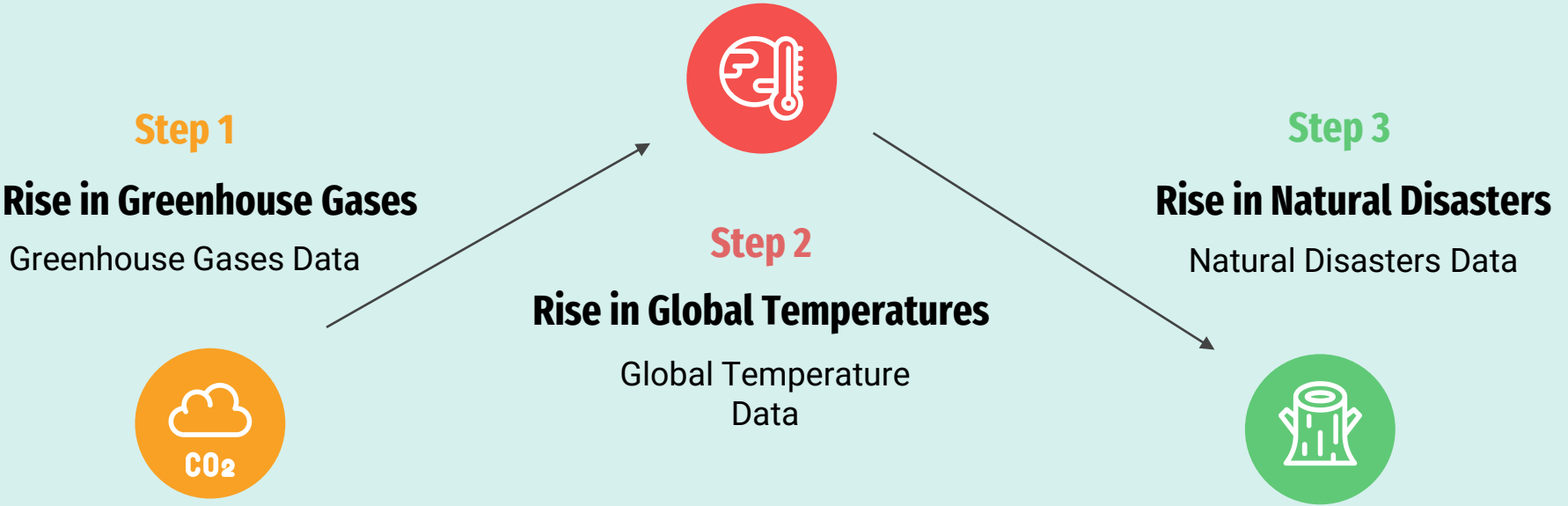
Since 1970, on average, for each disaster:

- Kills 367 people,
- Affects 740,209 lives,
- Causes USD $757,386.15 in damages

```
[71] averageDeaths = disasterData['Total Deaths'].mean()
     averageAffected = disasterData["Total Affected"].mean()
     averageDamages = disasterData["Total Damages ('000 US$)"].mean()
     avgDeathAffectedDamages = pd.DataFrame([[int(averageDeaths), \
     int(averageAffected), "USD $ " + str(round(averageDamages, 2))]], \
     columns=["Average Deaths", "Average Affected", "Average Damages"]);
     avgDeathAffectedDamages = avgDeathAffectedDamages.rename(index={0: 'Value'})
     avgDeathAffectedDamages
```

	Average Deaths	Average Affected	Average Damages
Value	367	740209	USD $ 767386.15

Problem Analysis and Breakdown



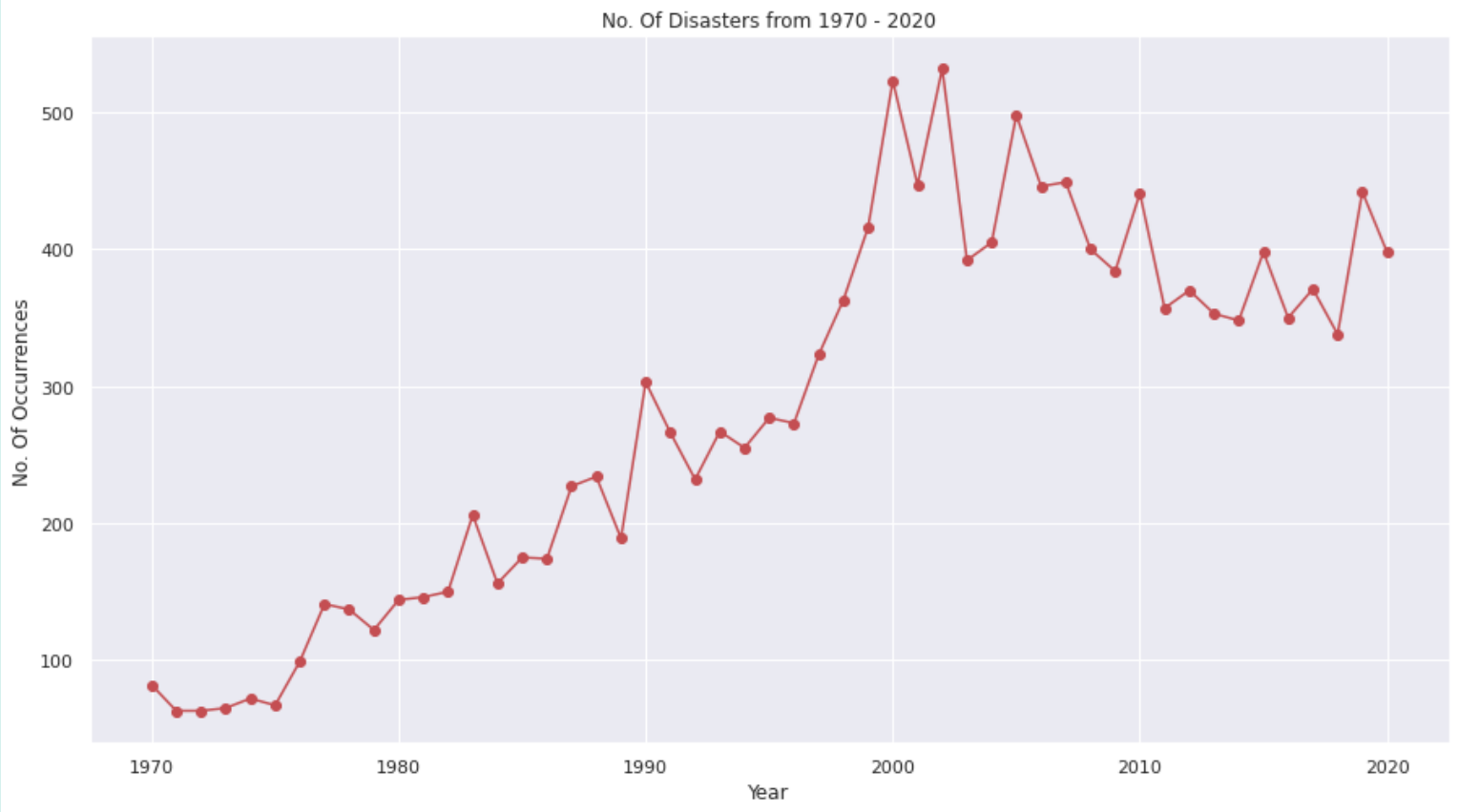
Datasets Sourcing

1. **Natural Disasters Dataset:** Kaggle - Baris Dincer
   - ALL NATURAL DISASTERS 1900-2021 / EOSDIS [1]
2. **Temperature Changes Dataset:** NASA
   - GISS Surface Temperature Analysis (GISTEMP v4) [2]
3. **Greenhouse Gases Datasets:** Global Monitoring Laboratory
   - Trends in Atmospheric Carbon Dioxide [3]
   - Trends in Atmospheric Methane [4]
   - Trends in Atmospheric Nitrous Oxide [5]

[1]: https://www.kaggle.com/datasets/brsdincer/all-natural-disasters-19002021-eosdis
[2]: https://data.giss.nasa.gov/gistemp/
[3]: https://gml.noaa.gov/ccgg/trends/data.html
[4]: https://gml.noaa.gov/ccgg/trends_ch4/
[5]: https://gml.noaa.gov/ccgg/trends_n2o/

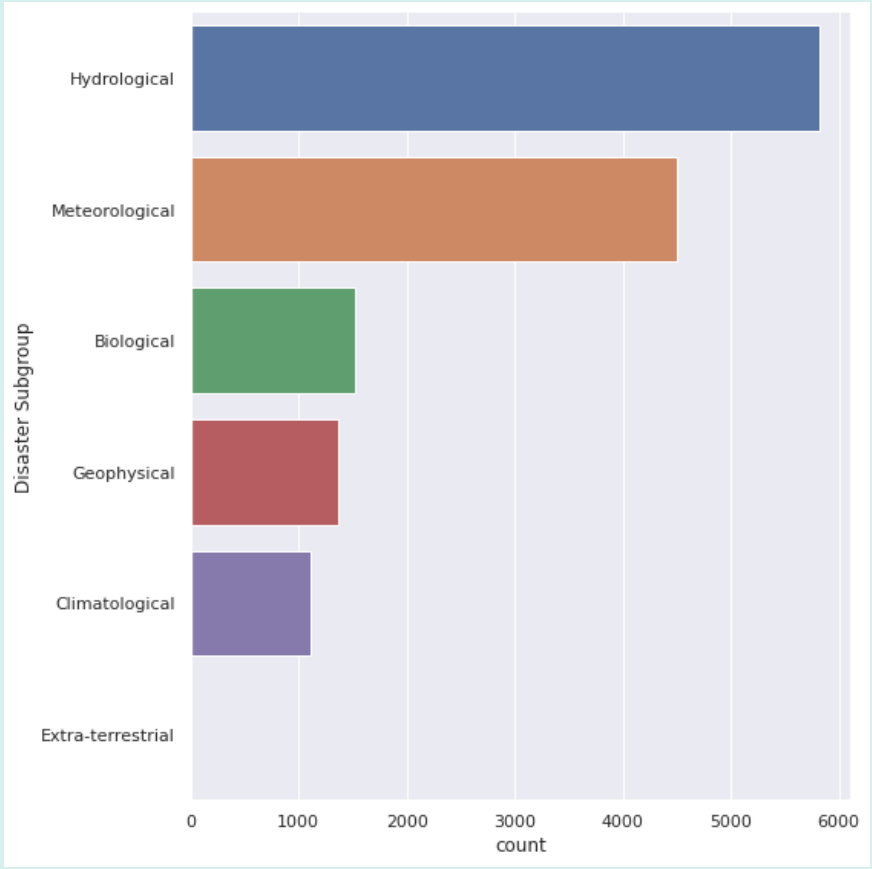
EDA

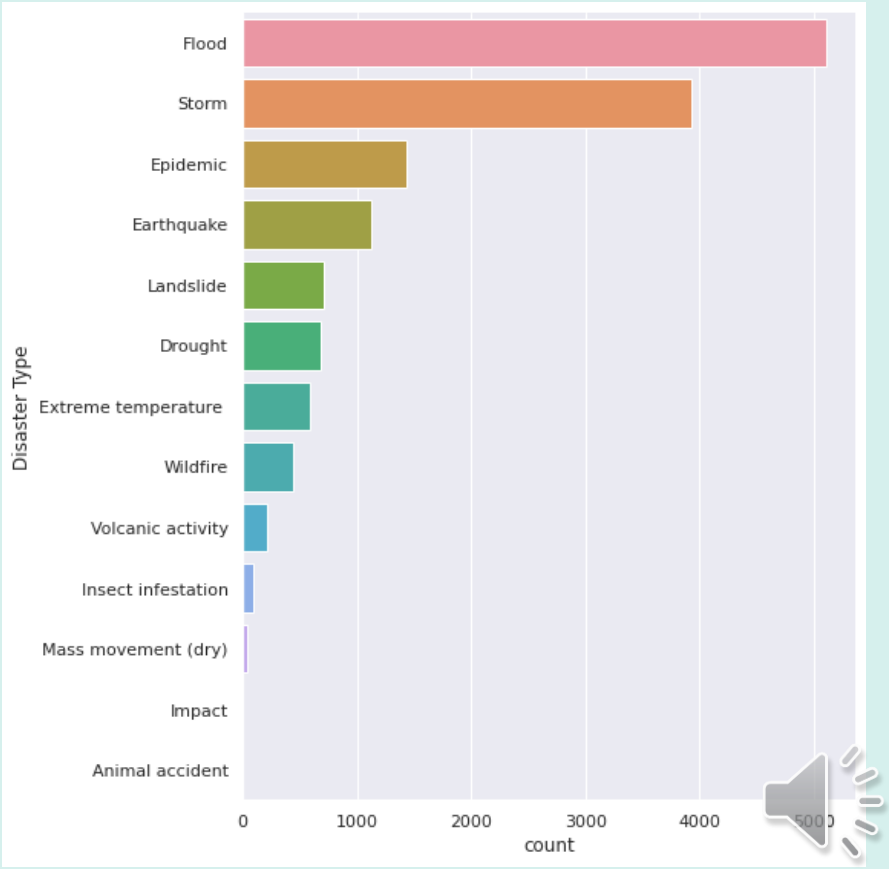
No. of Disasters from 1970 to 2020



EDA

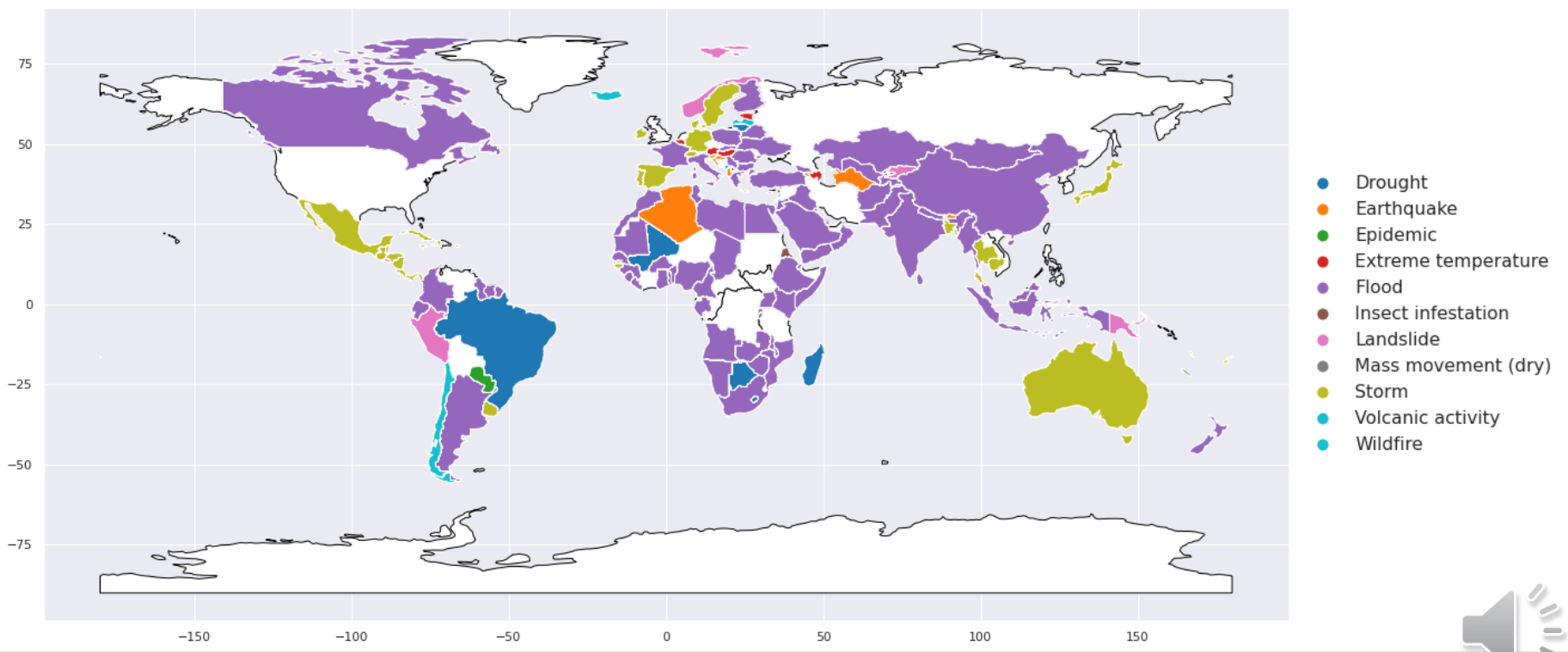
Disaster Subgroups and Types





EDA

World Map of Natural Disasters



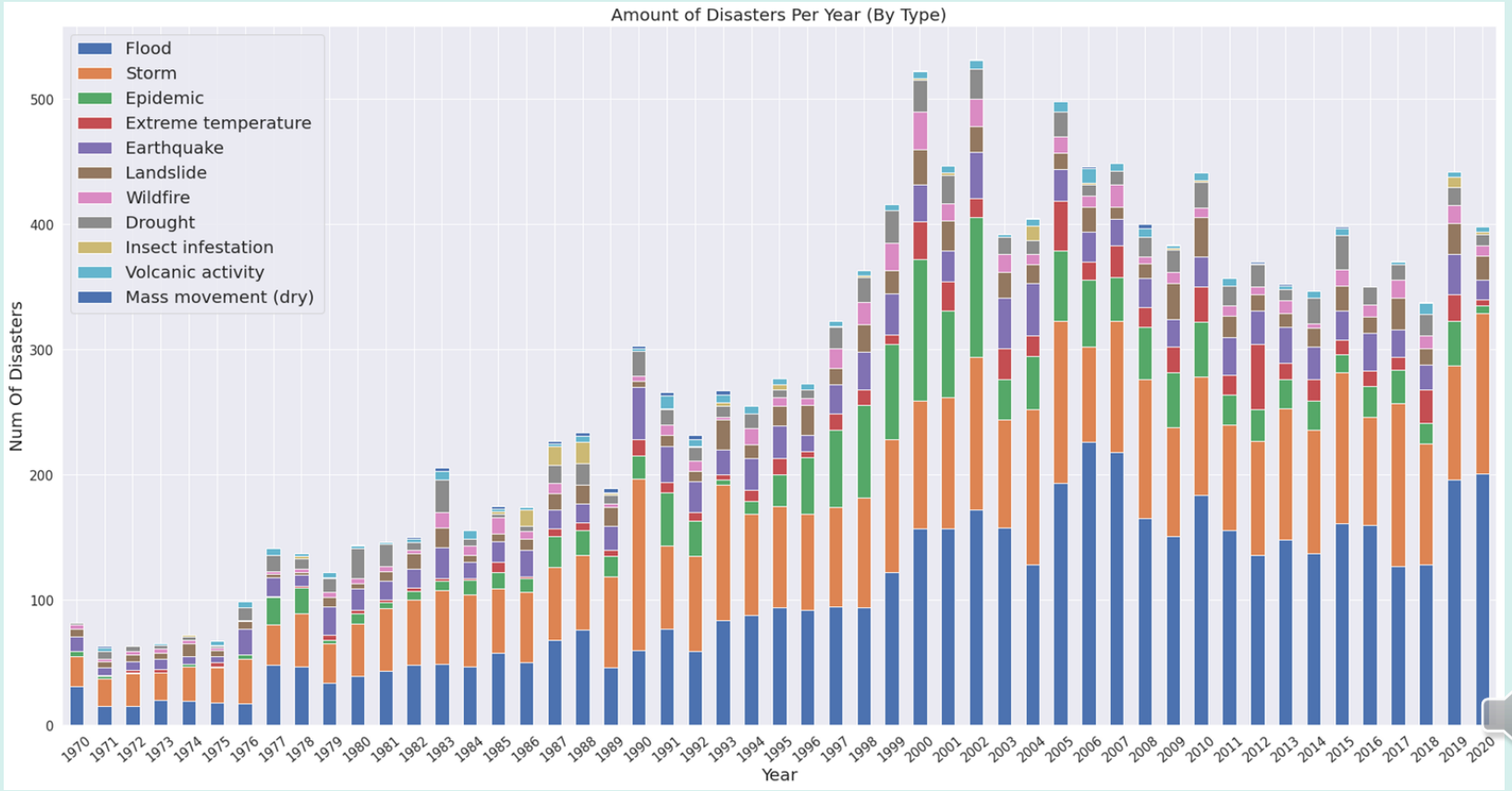
Year and Disaster Type Matrix

Year	Flood	Storm	Epidemic	Extreme temperature	Earthquake	Landslide	Wildfire	Drought	Insect infestation	Volcanic activity	Mass movement (dry)	Animal accident	Impact
1970	31	24	4	0	12	6	3	2	0	0	0	0	0
1971	15	22	2	1	6	5	2	6	0	3	1	0	0
1972	15	26	1	2	7	5	3	4	0	0	0	0	0
1973	20	22	0	3	8	5	3	3	0	1	0	0	0
1974	19	28	2	0	6	10	3	3	1	0	0	0	0
1975	18	28	1	3	5	5	2	1	1	3	0	0	0
1976	17	36	3	0	21	6	1	10	0	5	0	0	0
1977	48	32	22	1	15	3	2	13	0	5	0	0	0
1978	47	42	21	1	9	2	3	8	2	2	0	0	0
1979	34	31	3	4	23	7	4	11	1	4	0	0	0
1980	39	42	8	3	17	4	4	24	0	2	1	0	0
1981	43	50	5	2	15	8	4	18	0	1	0	0	0

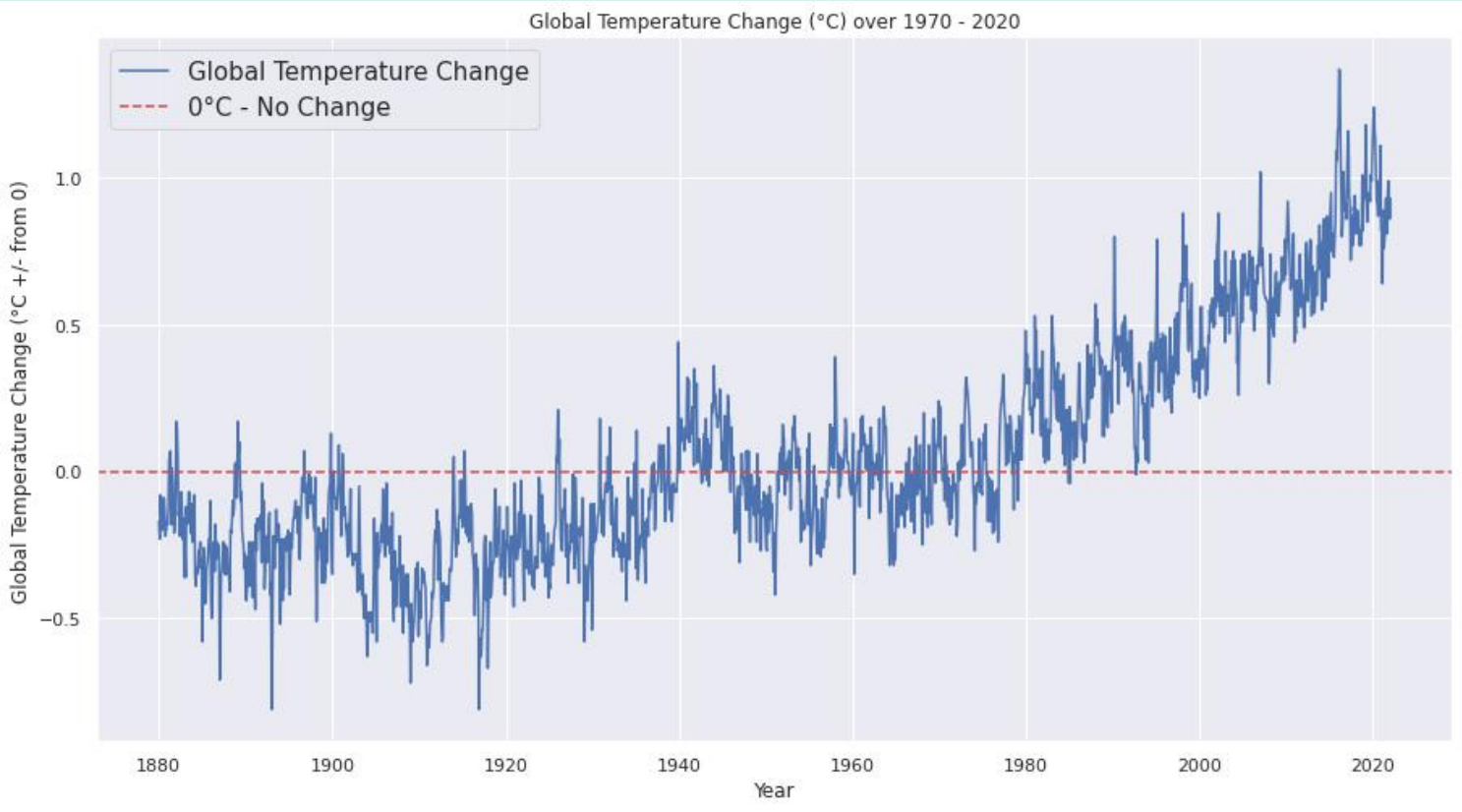
```
columnHeaders = disasterTypesPerYear["Disaster Type"].unique().tolist()
columnHeaders.insert(0, "Year")
rowHeaders = disasterTypesPerYear["Year"].unique().tolist()
yearDisasterMatrix = pd.DataFrame(columns=columnHeaders)

for i in range(len(rowHeaders)):
  yearDisasterMatrix.loc[i] = [rowHeaders[i], 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] # set all values to 0 first

yearDisasterMatrix.set_index("Year", inplace = True)
yearDisasterMatrix.sort_index(inplace = True)

for row in disasterTypesPerYear.iterrows():
  yearDisasterMatrix.at[row[1][0], row[1][1]] = row[1][2] # add data from previous dataframe

yearDisasterMatrix
```

EDA

Amount of Different Disasters Per Year

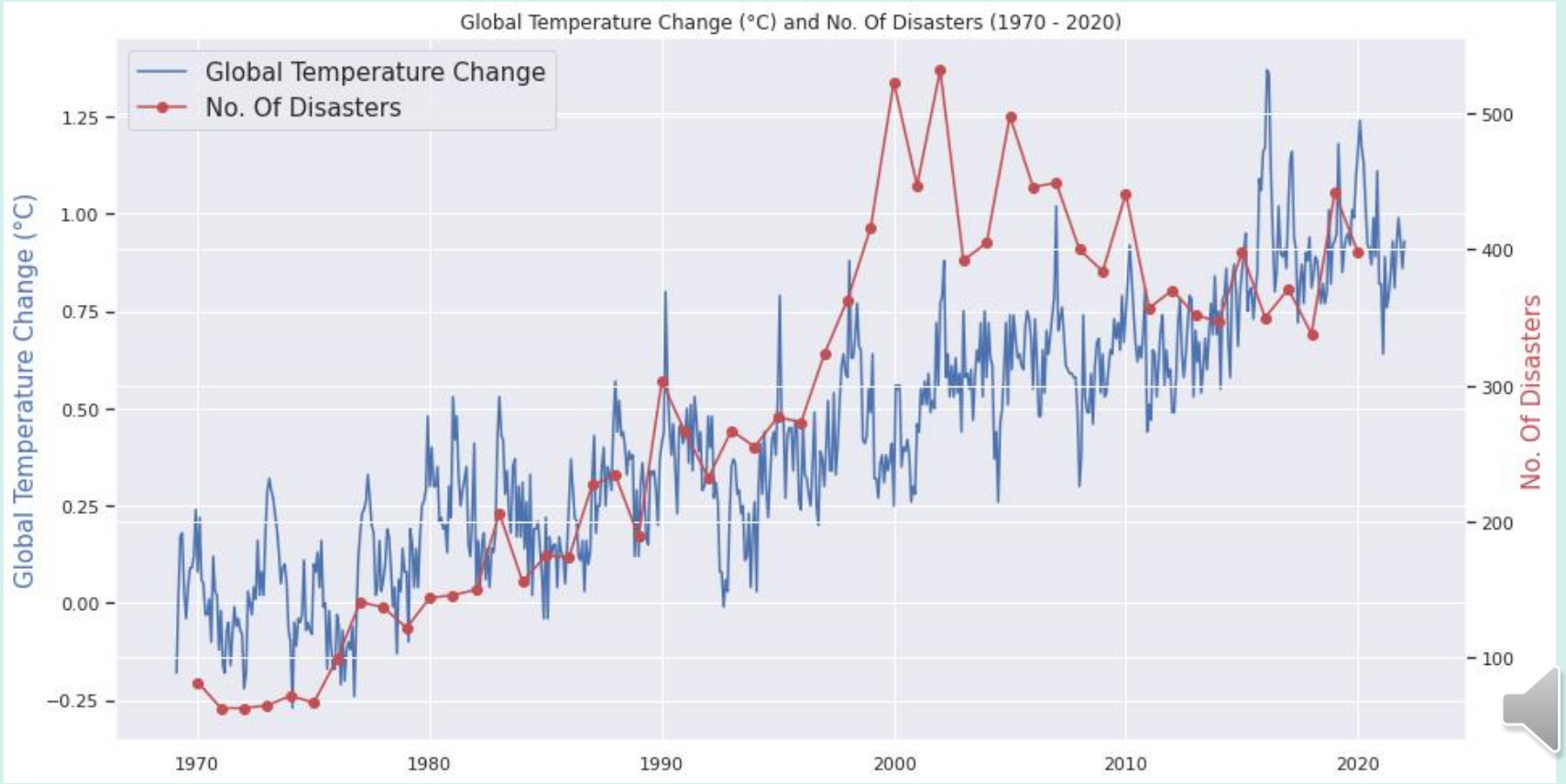


Temperature Change Dataset



EDA

Global Temperature Change and No. of Disasters from 1970 to 2020



Greenhouse Gases

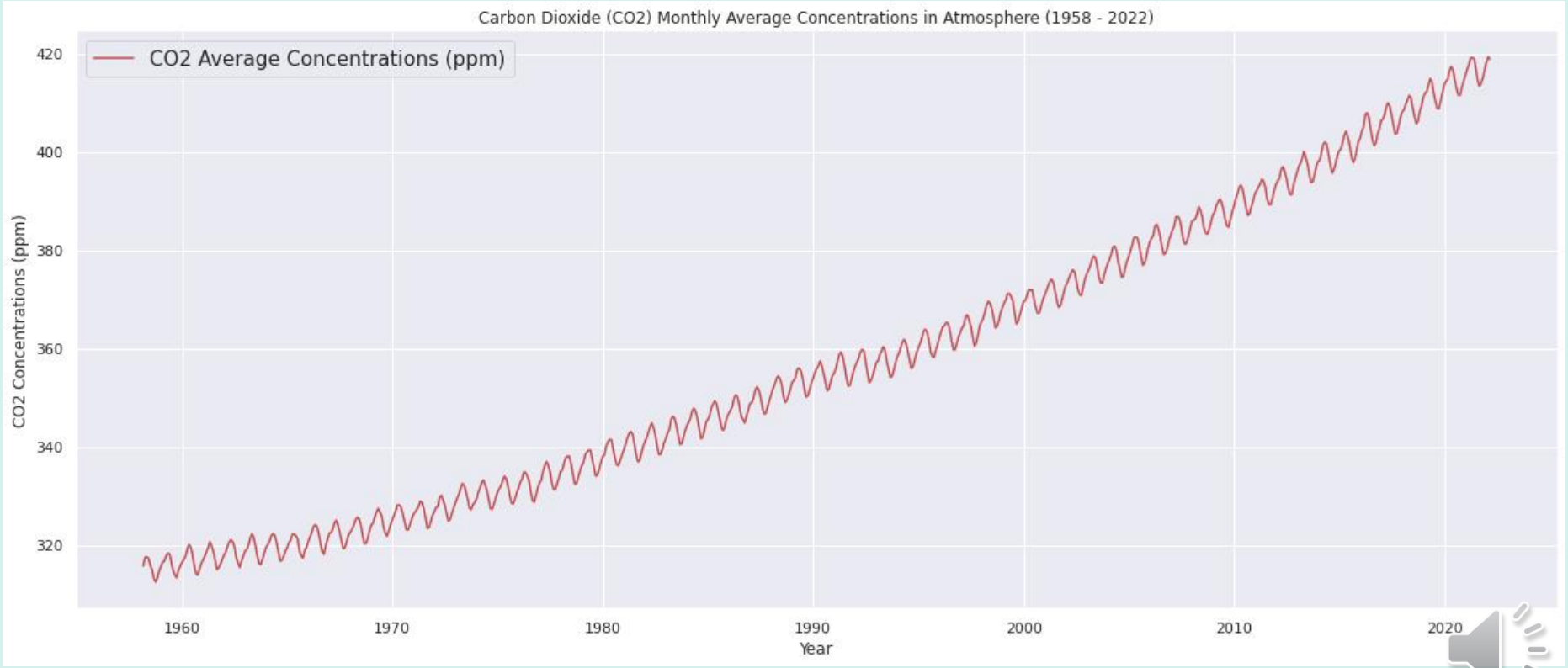
```
co2Data_model = co2Data.drop(columns=["decimal date", "interpolated", "trend", "ndays"])
co2Data_model = co2Data_model.loc[:, ~co2Data_model.columns.str.match('Unnamed')]
co2Data_model["period"] = co2Data_model["month"].astype(str) + "/" + co2Data_model["year"].astype(str)
co2Data_model.set_index("period", inplace=True)
co2Data_model.index = pd.DatetimeIndex(co2Data_model.index).to_period("M")
co2Data_model.drop(columns=["year", "month"], inplace=True)
co2Data_model.info()
co2Data_model
```

1. Combined months and years to one column to get the monthly period
2. Drop other columns and keep only the average columns
3. Set the period as the index for the datasets
4. This process was repeated for all the greenhouse gases dataset

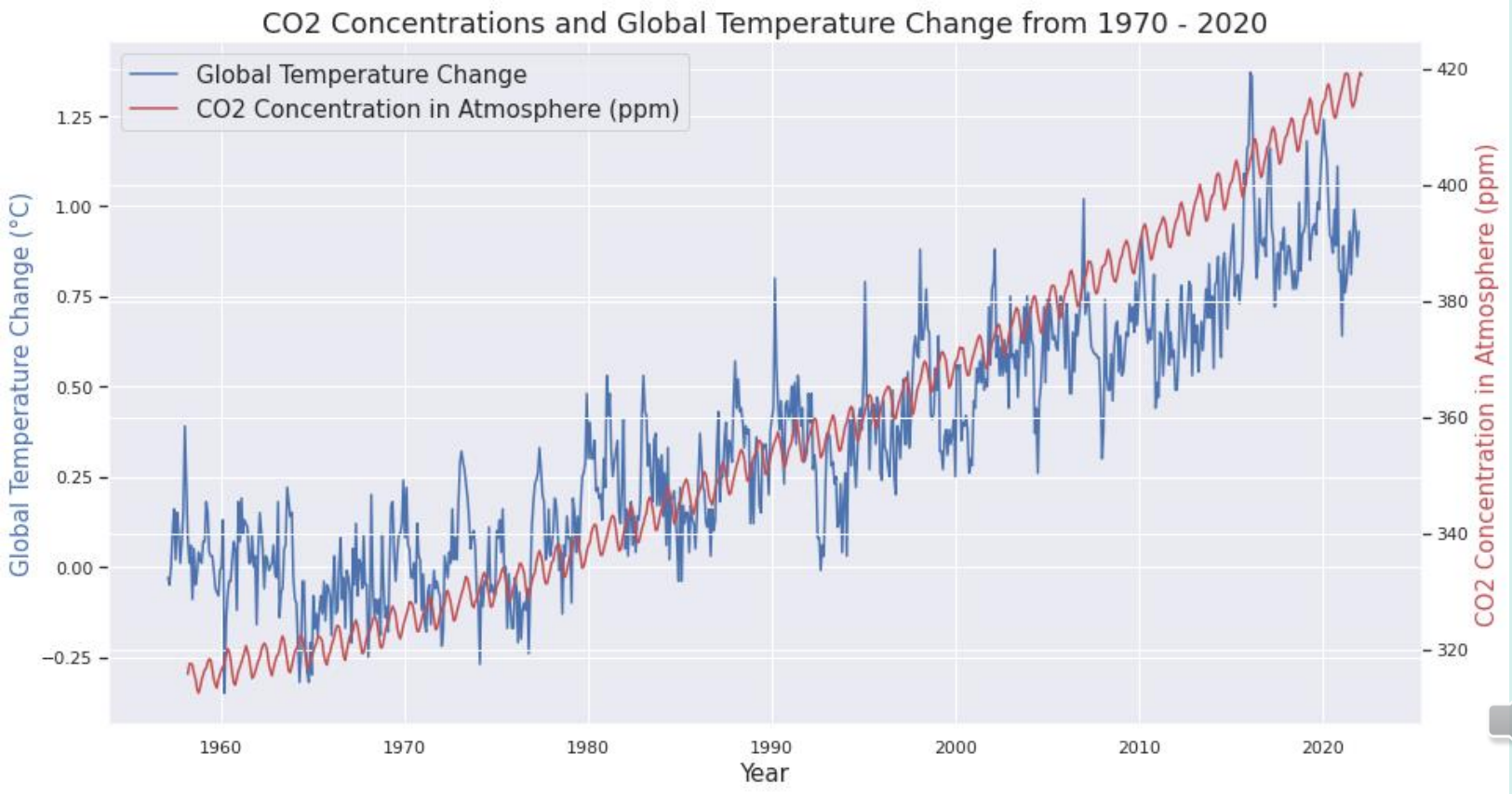
period	average
1958-03	315.70
1958-04	317.45
1958-05	317.51
1958-06	317.24
1958-07	315.86
...	...

EDA

CO2 Concentrations

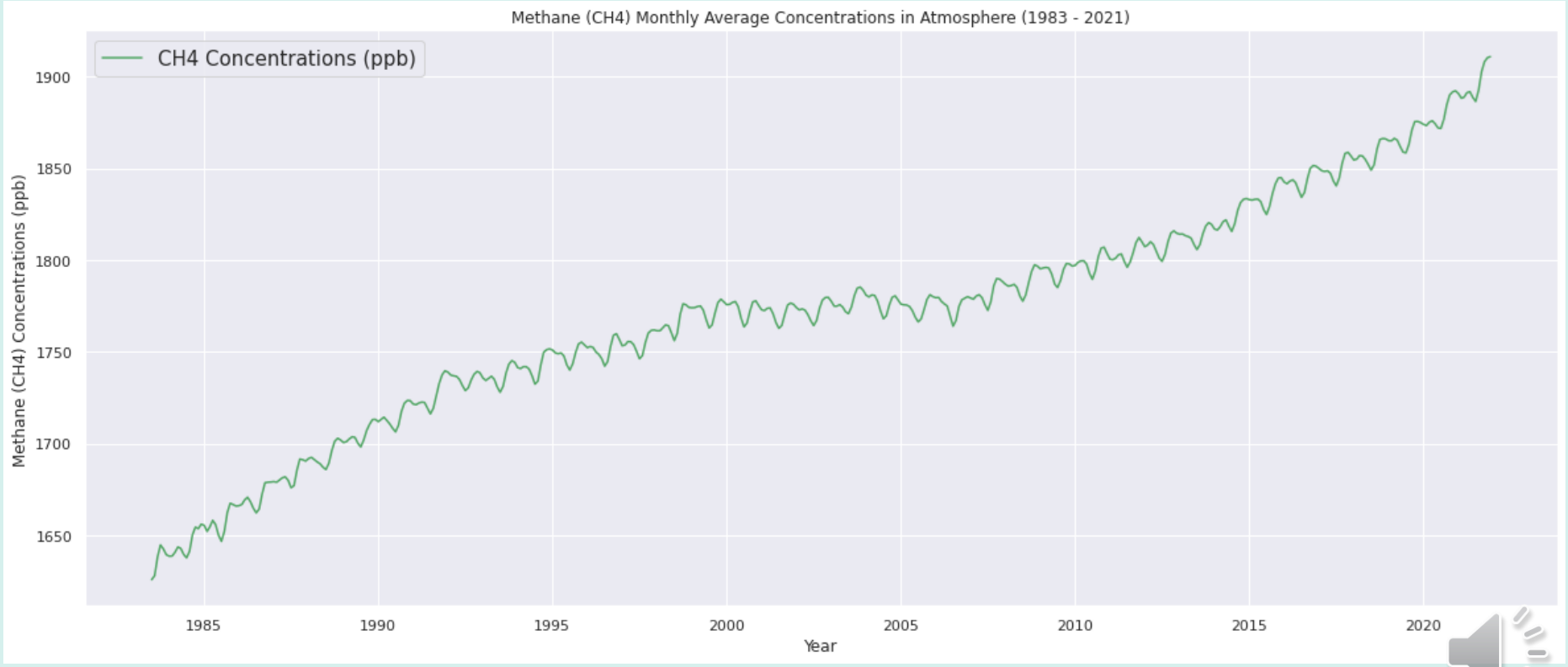


CO2 Concentrations and Temperature Change

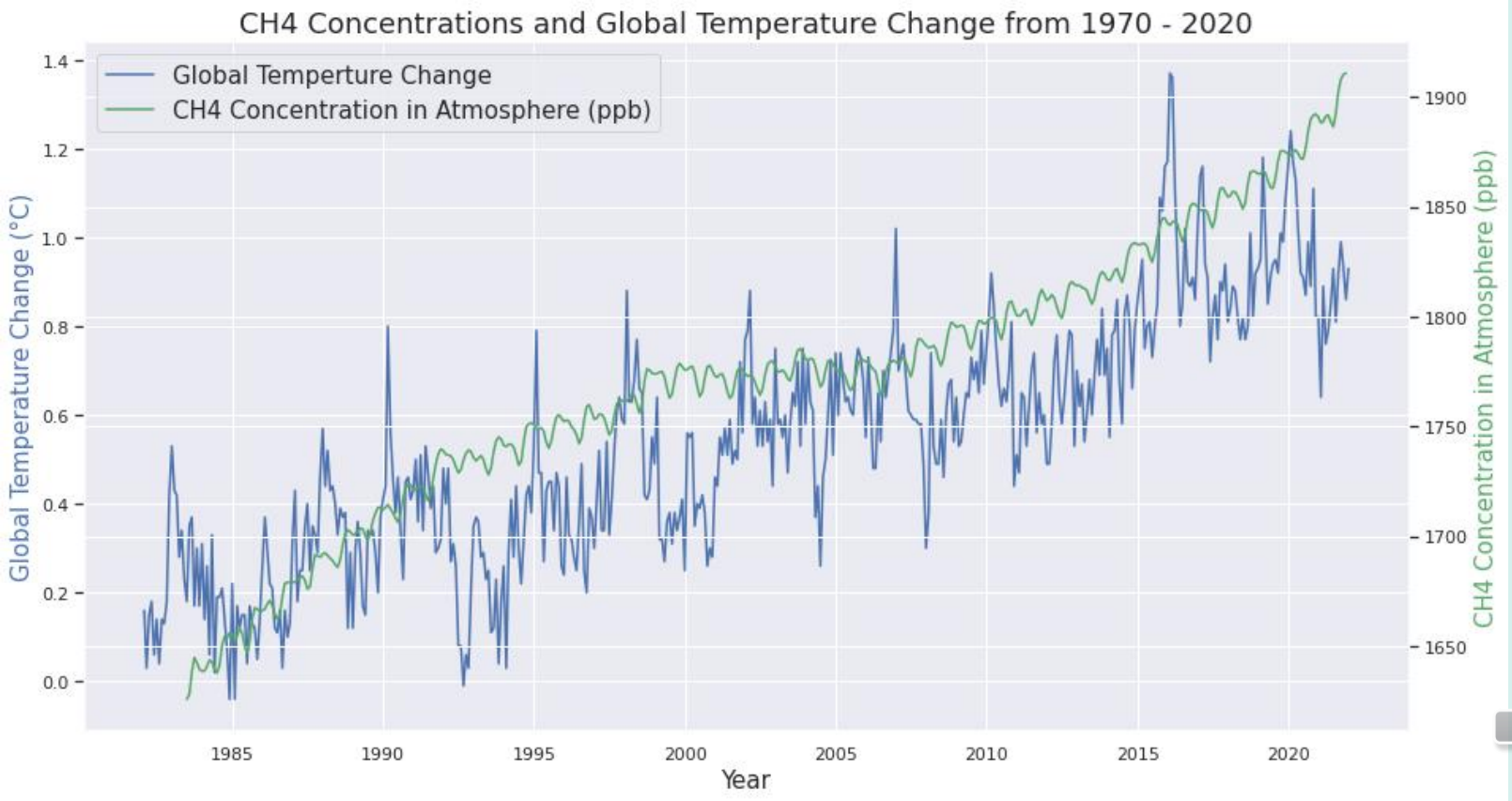


EDA

CH4 Concentrations

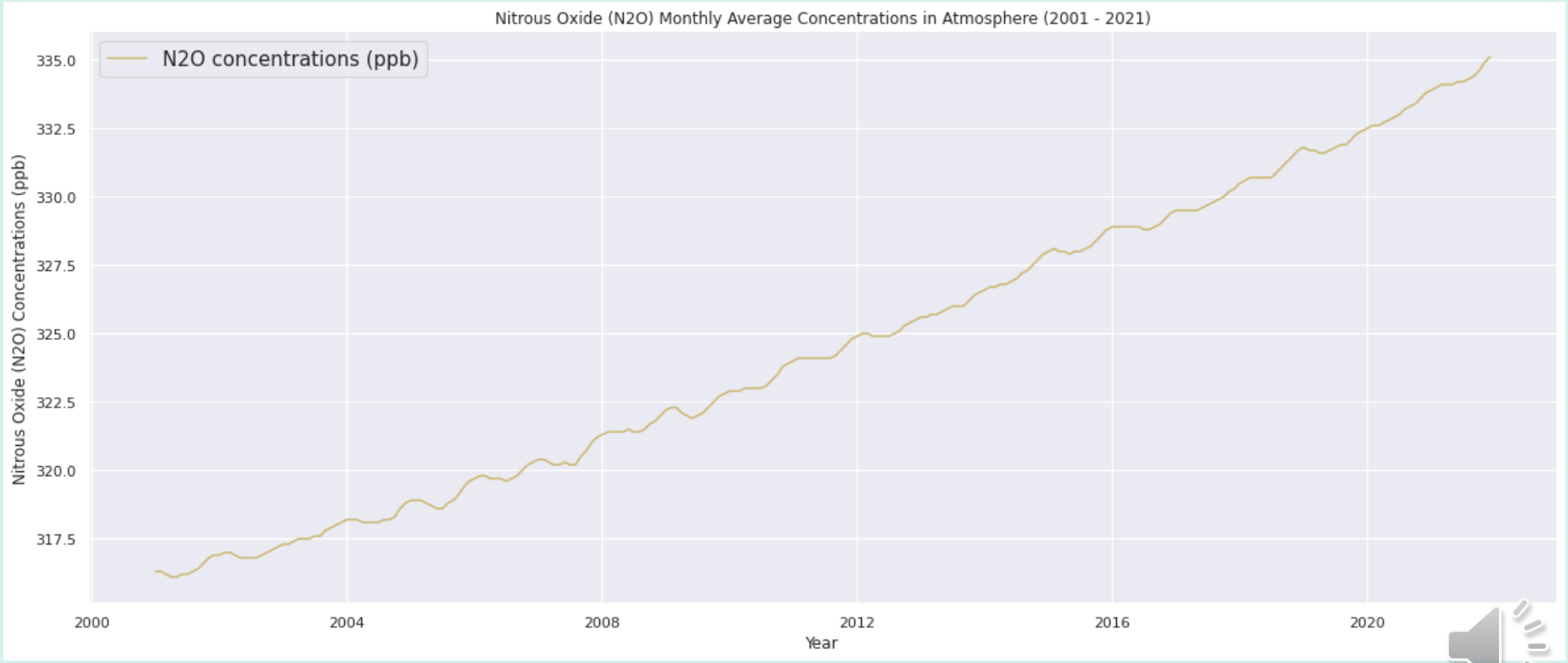


CH4 Concentrations and Temperature Change

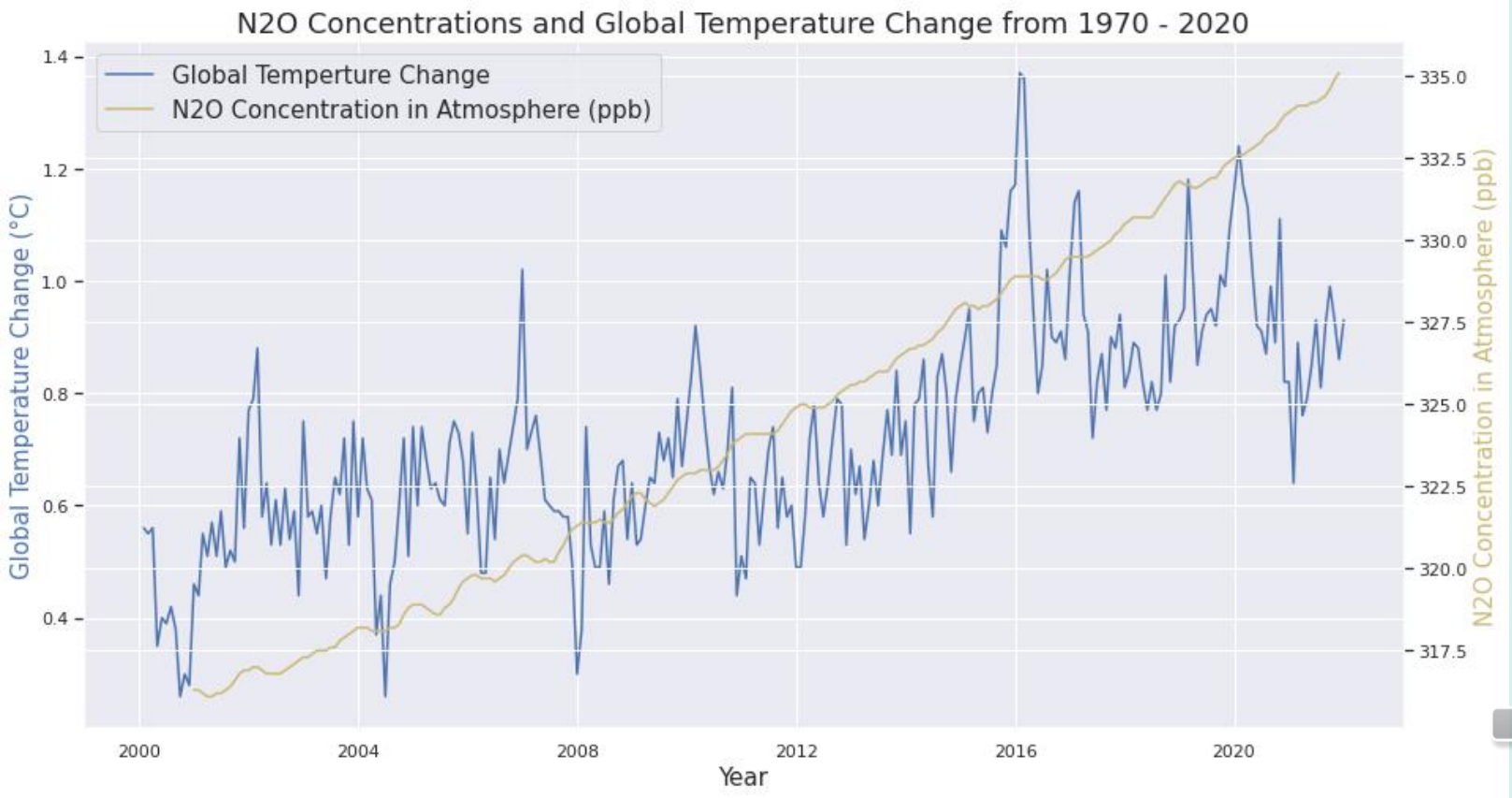


EDA

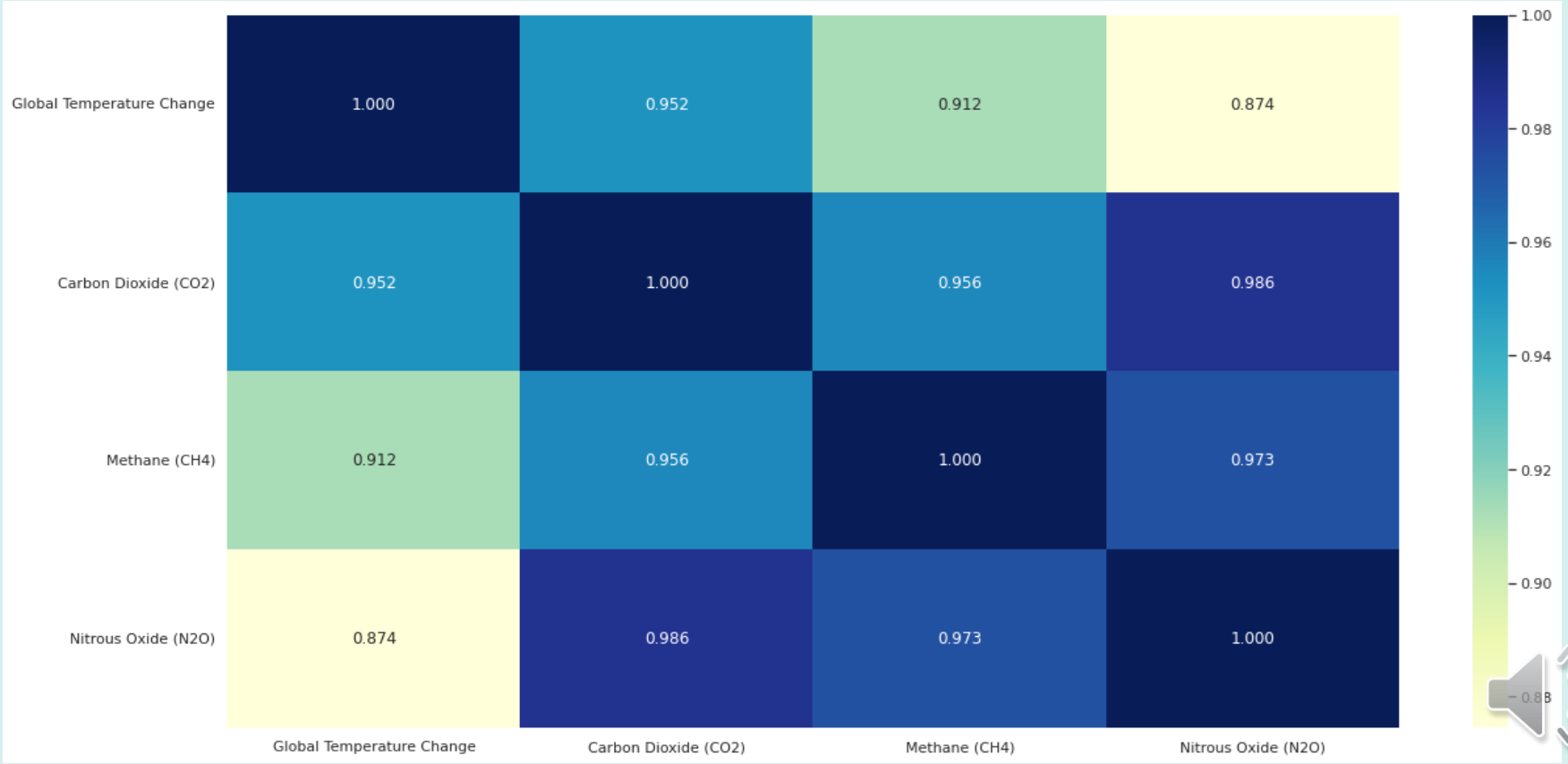
N2O Concentrations



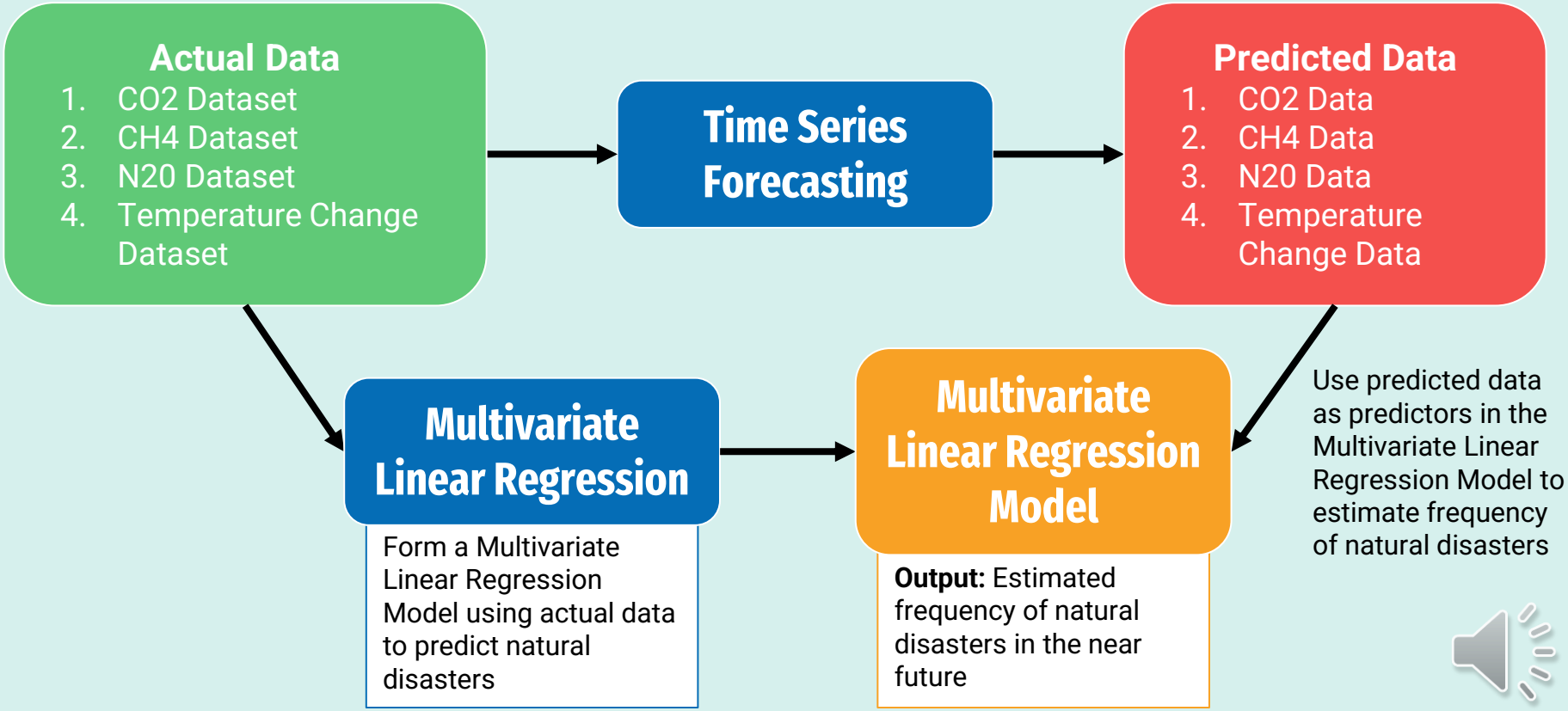
N2O Concentrations and Temperature Change



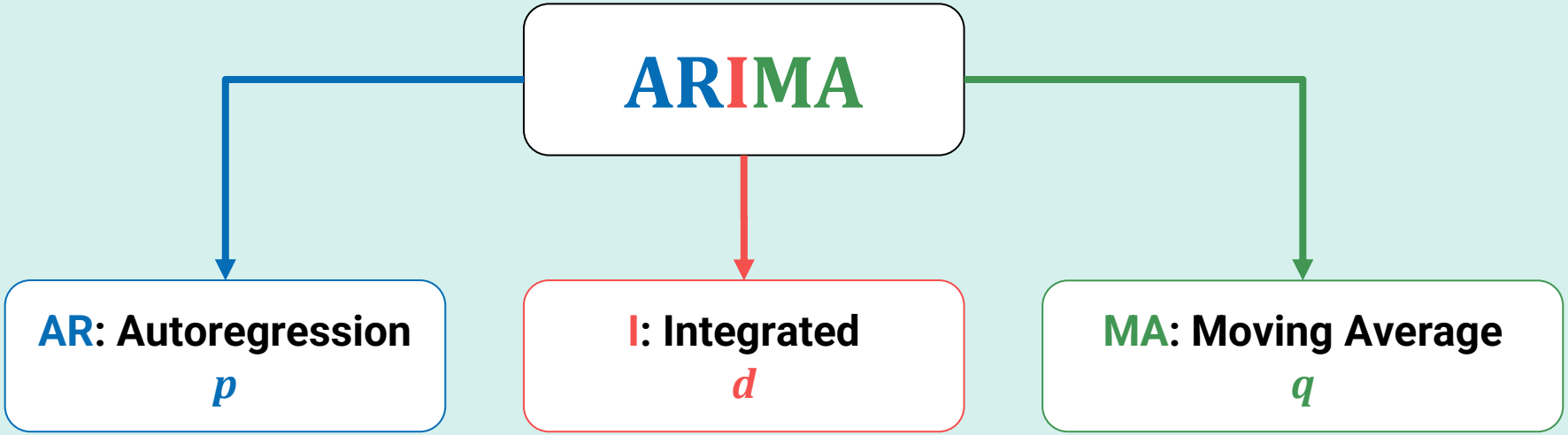
Correlation Matrix



Machine Learning Process



ARIMA Forecasting Model



Generalised Equation:

$$y'_t = c + \phi_1 y'_{t-1} + \cdots + \phi_p y'_{t-p} + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q} + \varepsilon_t,$$

ARIMA: Autoregression (AR, p)

Autoregression:

Forecasting variable of interest using a **linear combination of past values** of variable

If p = 0:
Equivalent to white noise; **no dependance on any past values**

If p = 1:
Autoregressive process based on **immediate preceding value**

If p = 2:
Autoregressive process is one in which the current value is based on the **previous two values**

If p = 3:
Autoregressive process is one in which the current value is based on the **previous three values**

Autoregressive model of order p:

$$Y_t = \alpha + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + .. + \beta_p Y_{t-p} + \epsilon_1$$

ARIMA: Integration (I, d)

Differencing:

Data values are replaced by the difference between the data values and the previous values

- **Differencing** is to make the data *stationary*; roams around a defined mean
- A stationary time series is one whose properties do not depend on the time at which the series is observed
  - **Crucial in ARIMA Model**

period	average	diff
1958-03	315.70	0.00
1958-04	317.45	1.75
1958-05	317.51	0.06
1958-06	317.24	-0.27
1958-07	315.86	-1.38
...	...	...
2021-11	415.01	1.08
2021-12	416.71	1.70
2022-01	418.19	1.48
2022-02	419.28	1.09
2022-03	418.81	-0.47

Differencing with different orders:

If d = 0: $y_t = Y_t$

If d = 1: $y_t = Y_t - Y_{t-1}$

If d = 2: $y_t = (Y_t - Y_{t-1}) - (Y_{t-1} - Y_{t-2}) = Y_t - 2Y_{t-1} + Y_{t-2}$

ARIMA: Moving Average (MA, q)

Moving Average:

A calculation used to analyze data points by creating a series of averages of different subsets of the full data set

The reason for calculating the moving average of data values is to help smooth out the value of the data by creating a constantly updated average value.

q: Size of moving average window

Moving Average model of order q:

$$Y_t = \alpha + \epsilon_t + \phi_1 \epsilon_{t-1} + \phi_2 \epsilon_{t-2} + .. + \phi_q \epsilon_{t-q}$$

ARIMA: Tuning the Parameters (p, d, q)

Grid Search

p: 0, 1 … 14, 15

d: 1, 2

q: 0, 1 … 14, 15

Best Fit:

1. Root Mean Squared Error (RMSE)
2. Akaike Information Criteria (AIC)

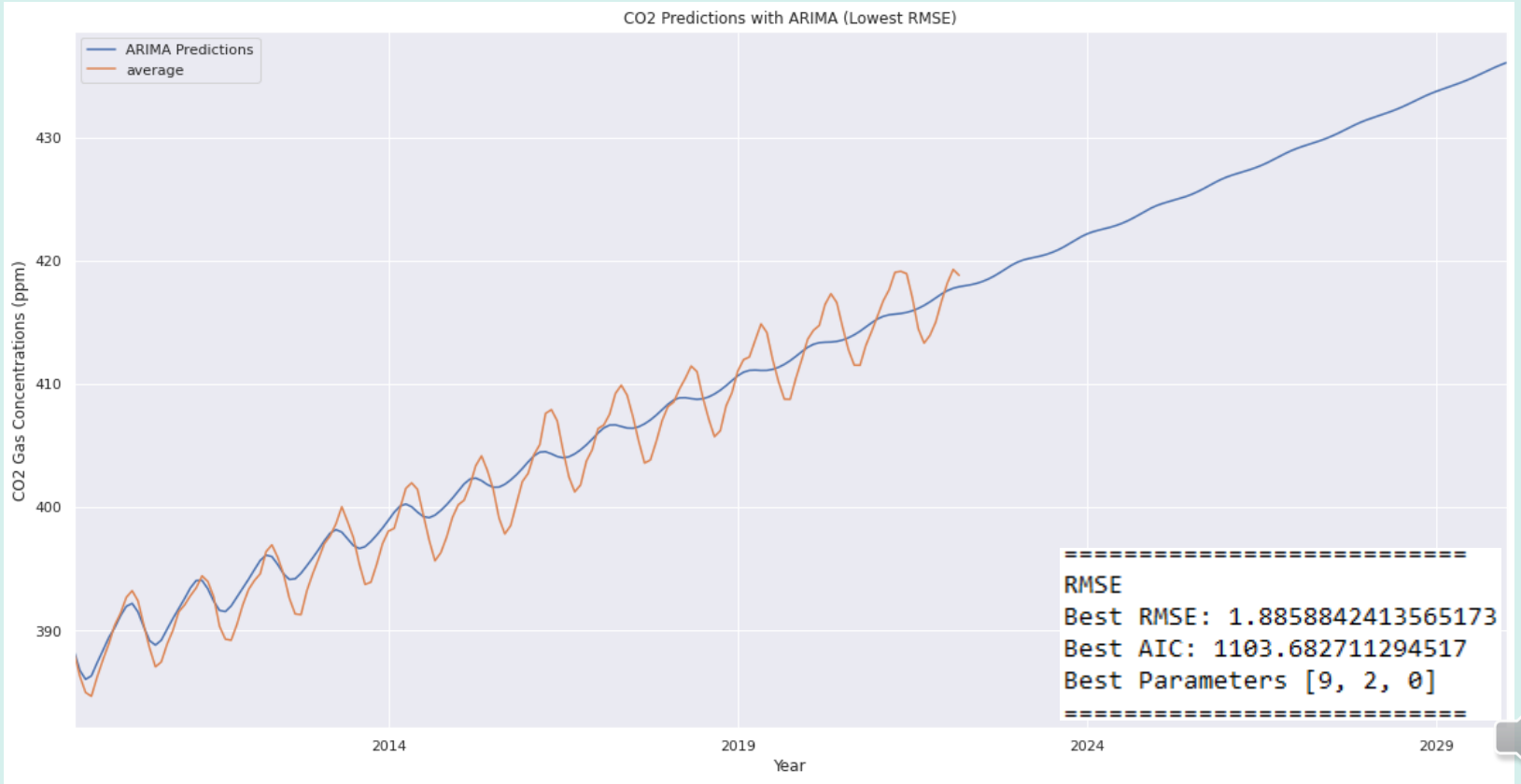
$$\mathrm{AIC} = -2\log(L) + 2(p + q + k + 1),$$

L: Likelihood of the data

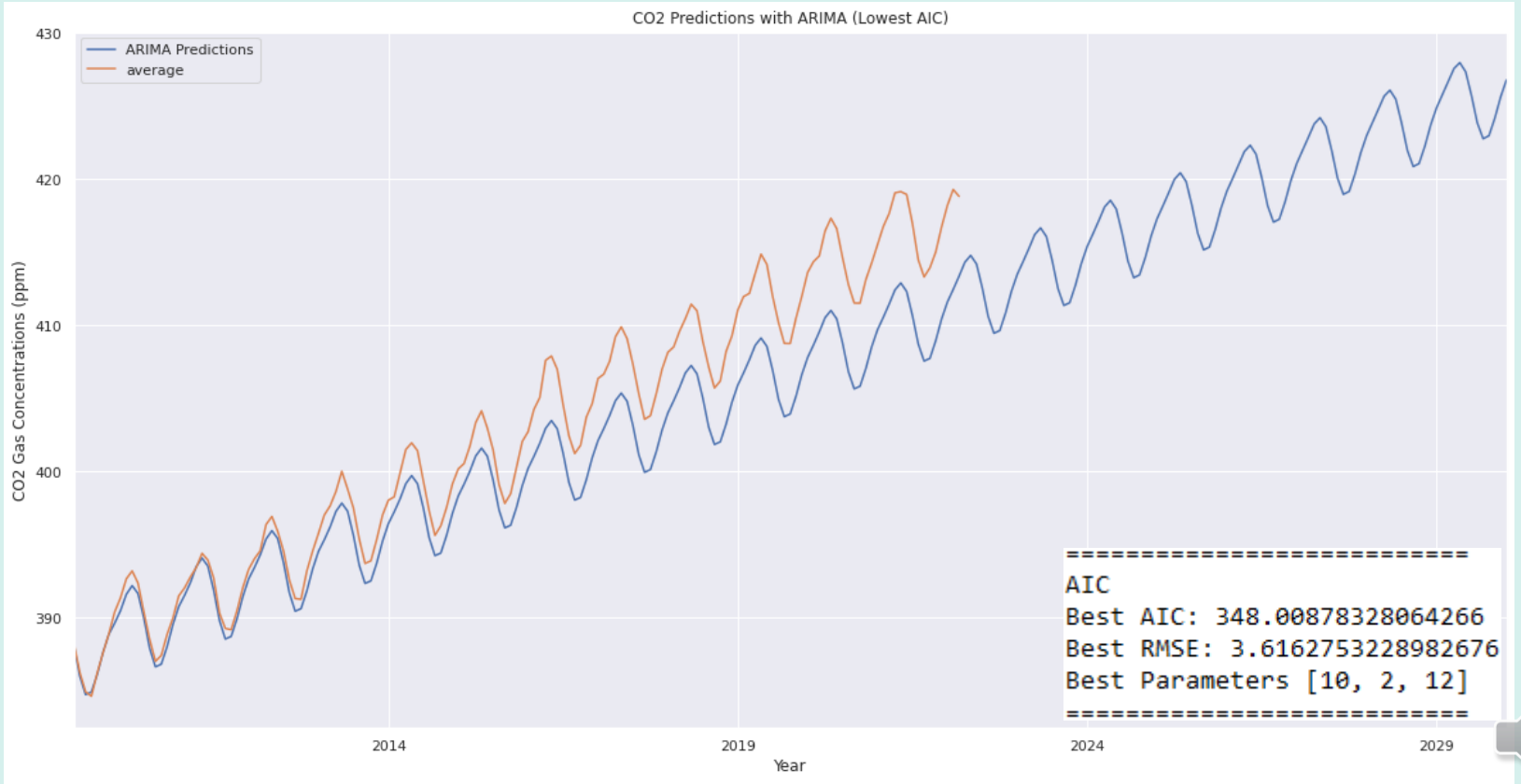
```
# Grid Search for Hyperparameters (p,q)
RMSE_best_parameters = [None, None, None]
RMSE_best_rmse = None
RMSE_best_aic = None

AIC_best_parameters = [None, None, None]
AIC_best_aic = None
AIC_best_rmse = None

train=co2Data_model.iloc[:-153]
test=co2Data_model.iloc[-153:]

for p in range(0,16):
    for d in range (1,3):
        for q in range (0,16):
            model = ARIMA(train['average'],order=(p, d, q))
            model = model.fit()

            start = len(train)
            end = len(train) + len(test) - 1
            pred=model.predict(start=start,end=end,typ='levels').rename('ARIMA Predictions, p:' + str(p) + ' q: ' + str(q))
            print("Order:", p, d, q)
            aic = model.aic
            print("AIC:", aic)
            rmse = np.sqrt(mean_squared_error(pred,test['average']))
            print("Test RMSE:", rmse)

            if RMSE_best_rmse is None or rmse < RMSE_best_rmse:
                RMSE_best_rmse = rmse
                RMSE_best_aic = aic
                RMSE_best_parameters[0] = p
                RMSE_best_parameters[1] = d
                RMSE_best_parameters[2] = q

            if AIC_best_aic is None or aic < AIC_best_aic:
                AIC_best_rmse = rmse
                AIC_best_aic = aic
                AIC_best_parameters[0] = p
                AIC_best_parameters[1] = d
                AIC_best_parameters[2] = q

print("============================")
print("RMSE")
print("Best RMSE:", RMSE_best_rmse)
print("Best AIC:", RMSE_best_aic)
print("Best Parameters", RMSE_best_parameters)
print("============================")
print("============================")
print("AIC")
print("Best AIC:", AIC_best_aic)
print("Best RMSE:", AIC_best_rmse)
print("Best Parameters", AIC_best_parameters)
print("============================")
```

```
============================
RMSE
Best RMSE: 1.8858842413565173
Best AIC: 1103.682711294517
Best Parameters [9, 2, 0]
============================
============================
AIC
Best AIC: 348.00878328064266
Best RMSE: 3.6162753238932676
Best Parameters [10, 2, 12]
============================
```

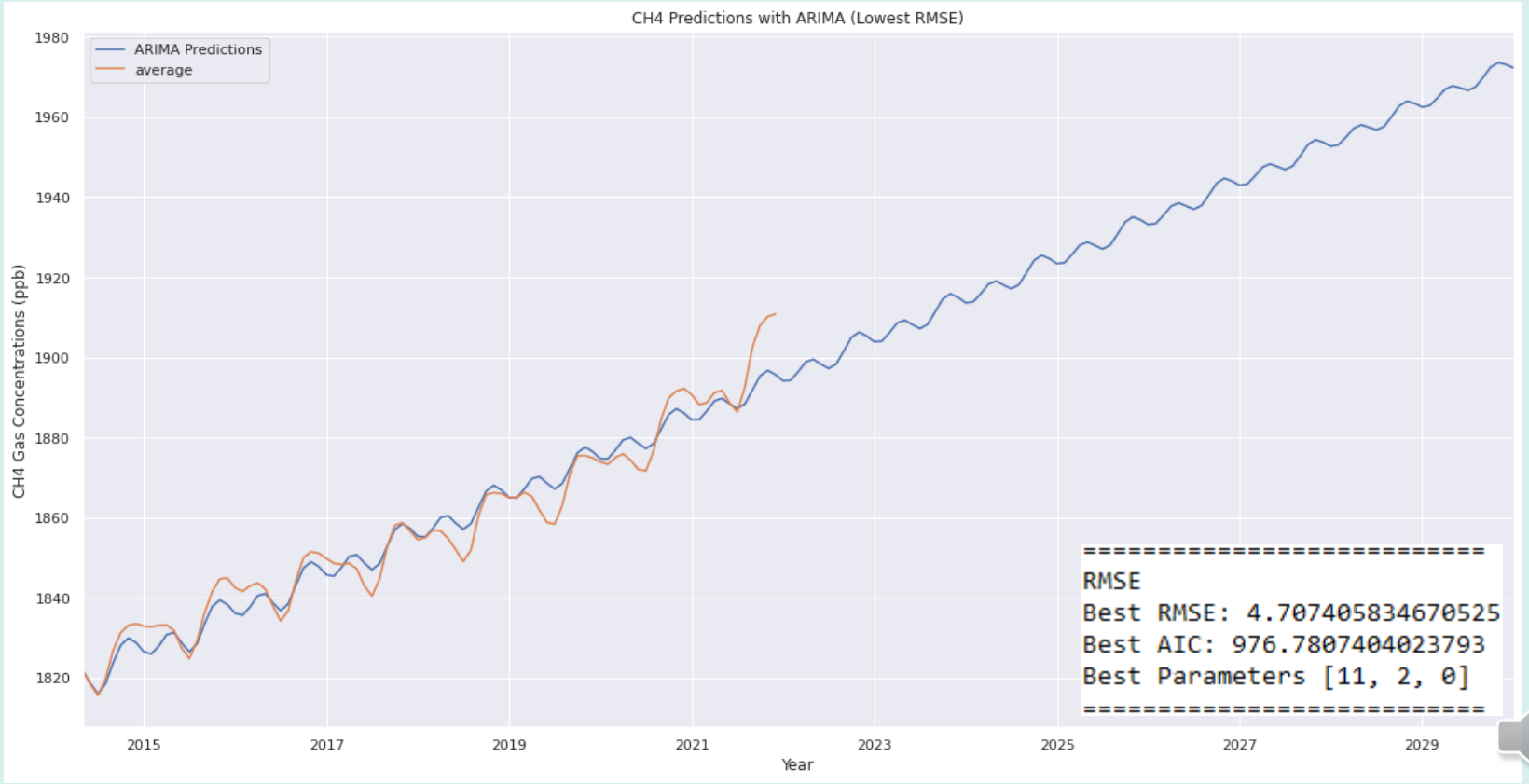
ARIMA - CO2 Predictions (Lowest RMSE)



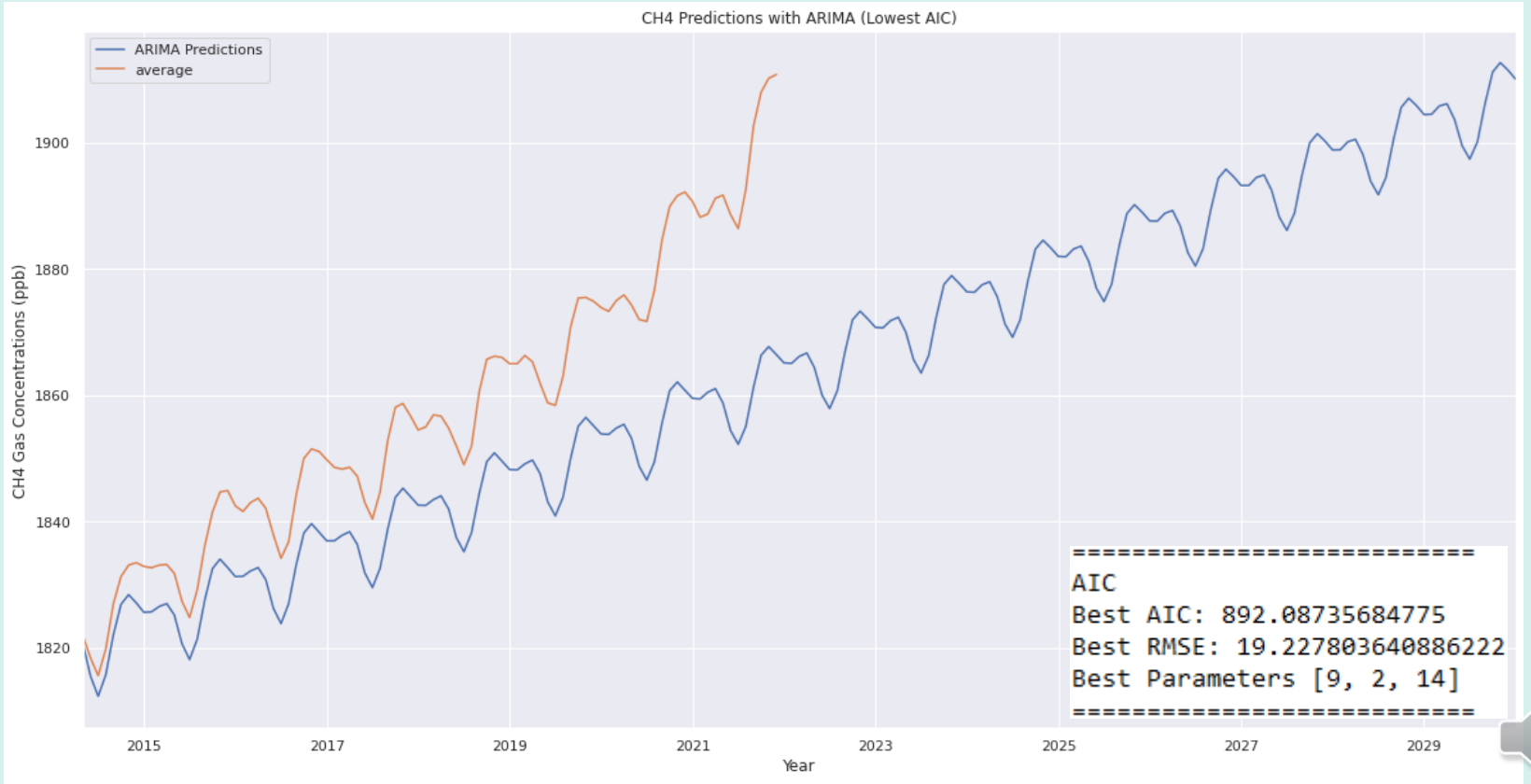
ARIMA - CO2 Predictions (Lowest AIC)



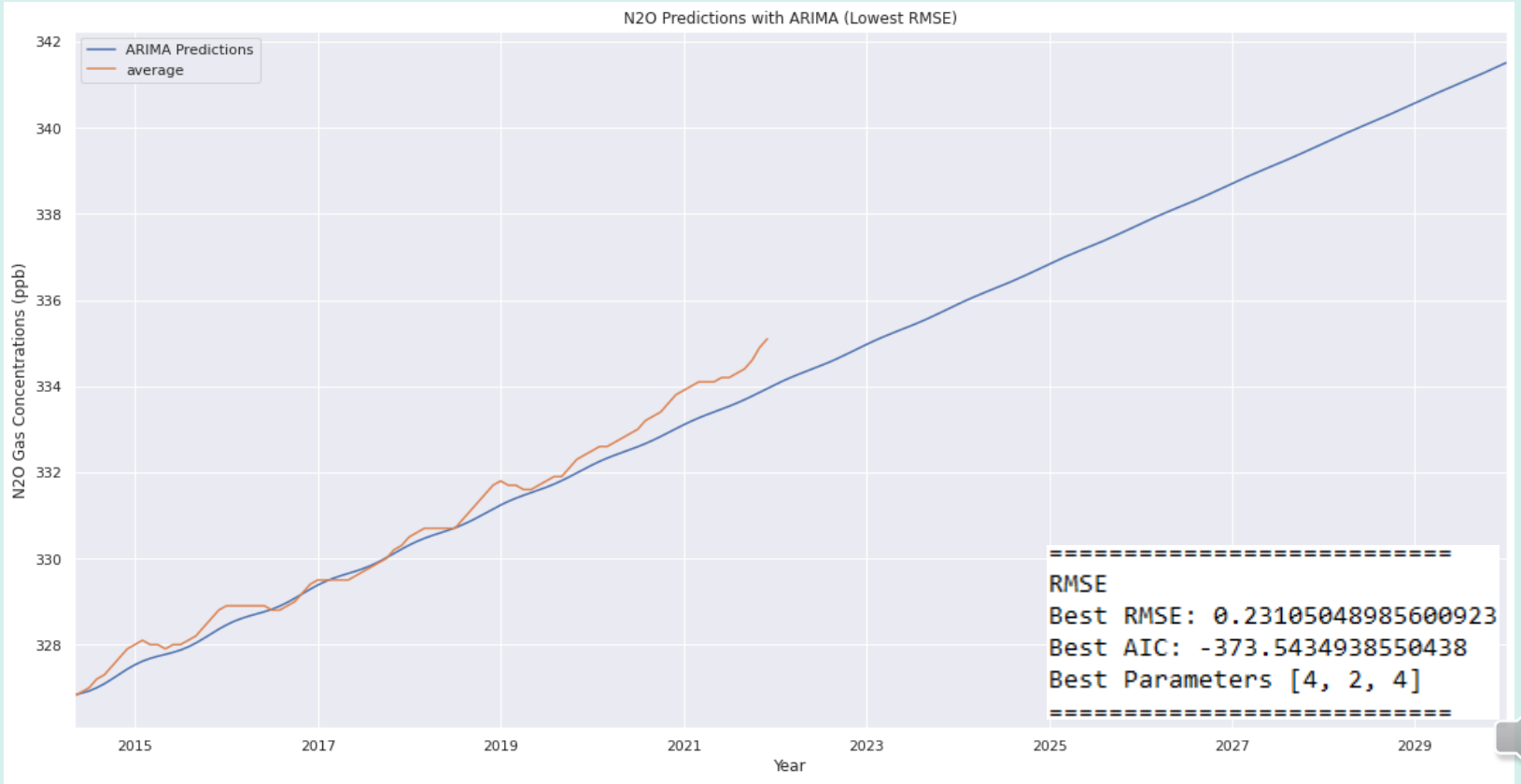
ARIMA - CH4 Predictions (Lowest RMSE)



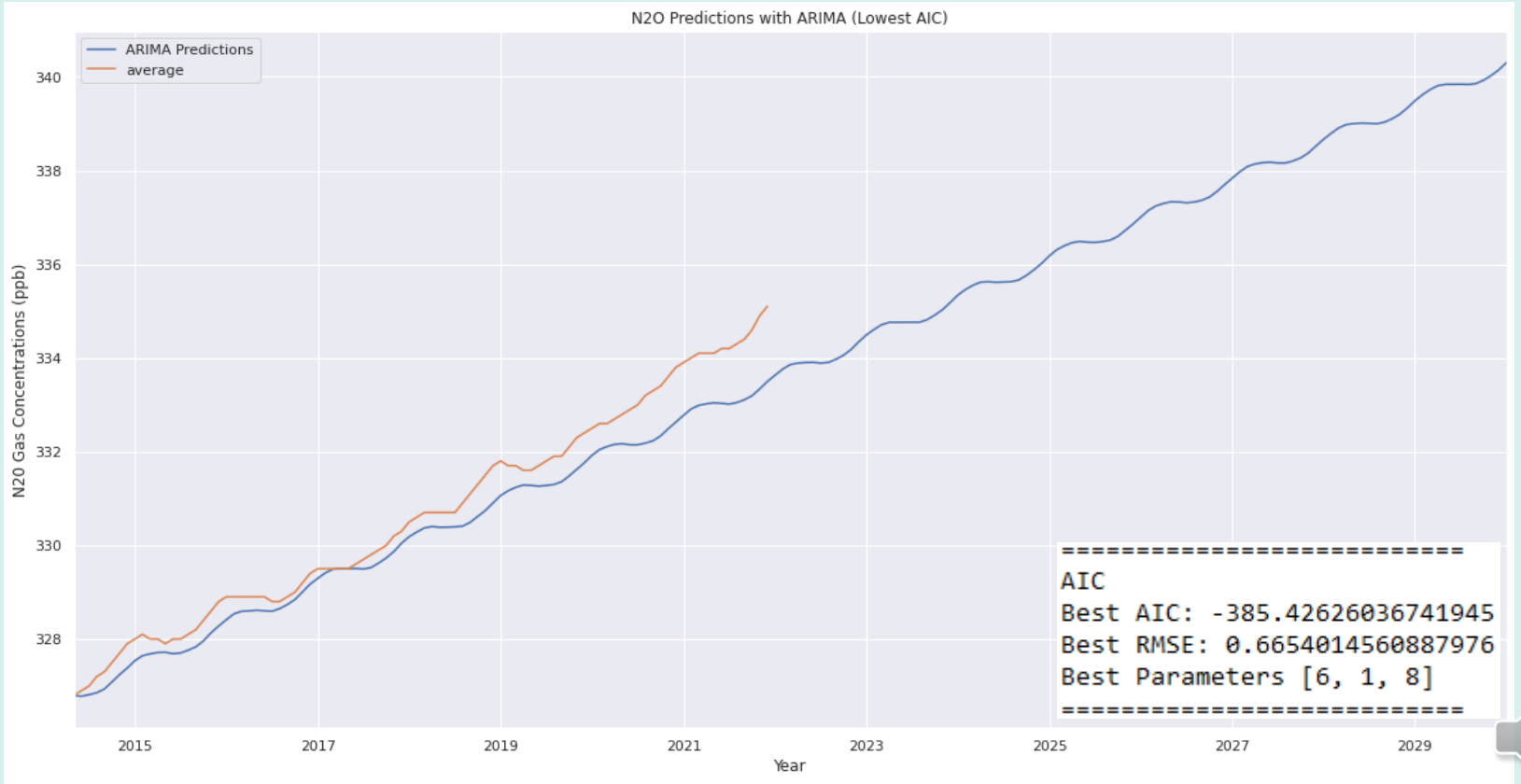
ARIMA - CH4 Predictions (Lowest AIC)



ARIMA - N2O Predictions (Lowest RMSE)



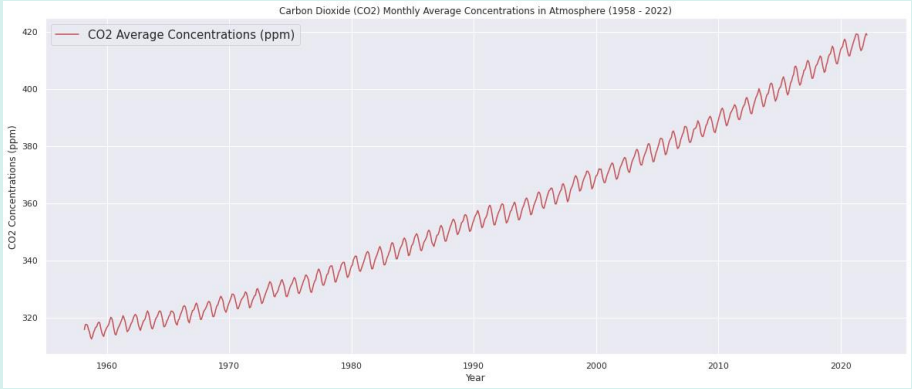
ARIMA - N2O Predictions (Lowest AIC)

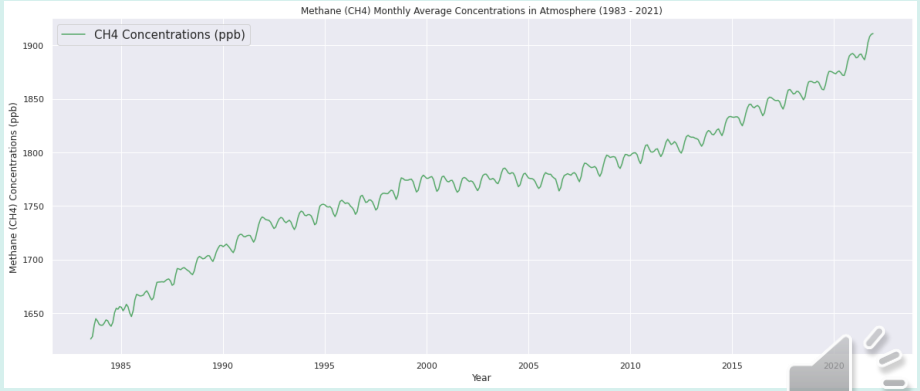


ARIMA Summary

ARIMA is **not accurate** enough for our use:

- It does not take into account the **seasonality in data**
- In certain months of the year, there is a **repeated fluctuation** of concentration of greenhouse gases in the atmosphere
- We decided to use **Seasonal-ARIMA (SARIMA)** instead





SARIMA: Equation

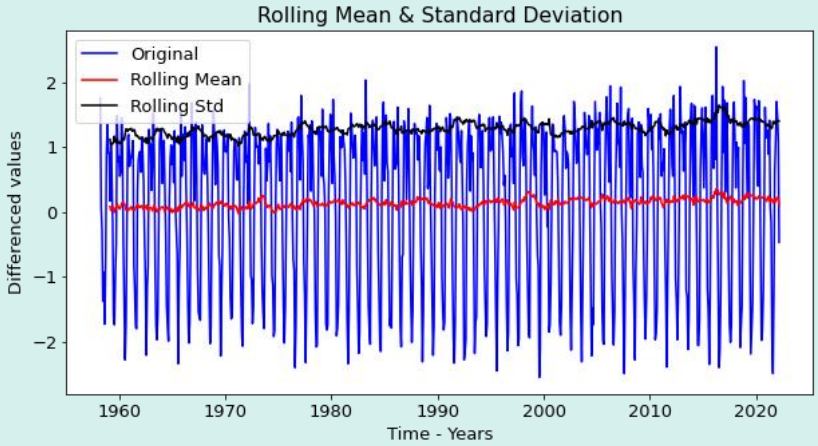
$$SARIMA \underbrace{(p,d,q)}_{non-seasonal} \underbrace{(P,D,Q)_m}_{seasonal}$$

SARIMA: Stationarity & Differencing

```
def test_stationarity(timeseries, rolling=12):

    #Determing rolling statistics
    rolmean = timeseries.rolling(rolling).mean()
    rolstd = timeseries.rolling(rolling).std()

    #Plot rolling statistics:
    plt.figure(figsize=(10, 5))
    orig = plt.plot(timeseries, color='blue',label='Original')
    mean = plt.plot(rolmean, color='red', label='Rolling Mean')
    std = plt.plot(rolstd, color='black', label = 'Rolling Std')
    plt.title('Power consumption Old data')
    plt.xlabel('Time - Years')
    plt.ylabel('Differenced values')
    plt.legend(loc='best')
    plt.title('Rolling Mean & Standard Deviation')
    plt.show()

    #Perform Dickey-Fuller test:
    print ('Results of Dickey-Fuller Test:')
    dftest = adfuller(timeseries, autolag='AIC')
    dfoutput = pd.Series(dftest[0:4], index=['Test Statistic','p-value','#Lags Used',
                                             'Number of Observations Used'])
    for key,value in dftest[4].items():
        dfoutput['Critical Value (%s)'%key] = value
    print (dfoutput)
    if dfoutput['p-value'] < 0.05:
        print('The time series is stationary at 95% level of confidence')
    else:
        print('The time series is not stationary at 95% level of confidence')


co2_diff = co2['average'] - co2['average'].shift(1) # perform differencing operation
co2_diff = co2_diff.dropna()
test_stationarity(co2_diff, rolling=12)
```

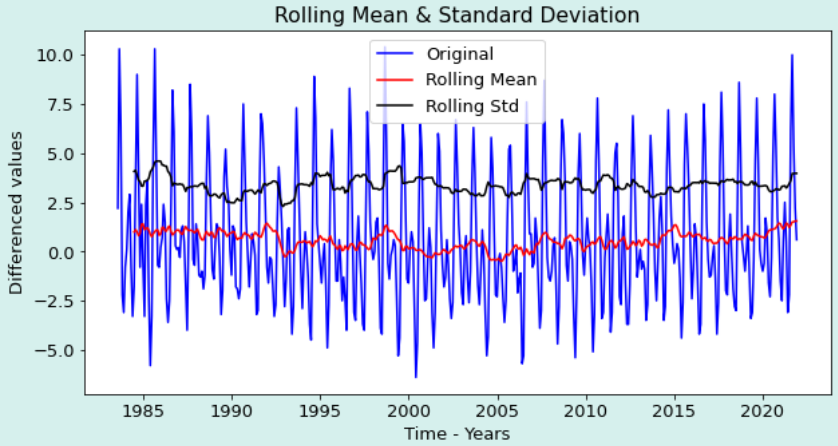
SARIMA: Stationarity & Differencing

CO_2




```
Results of Dickey-Fuller Test:
Test Statistic                  -5.245204
p-value                          0.000007
#Lags Used                      20.000000
Number of Observations Used    747.000000
Critical Value (1%)             -3.439134
Critical Value (5%)             -2.865417
Critical Value (10%)            -2.568834
dtype: float64
The time series is stationary at 95% level of confidence
```

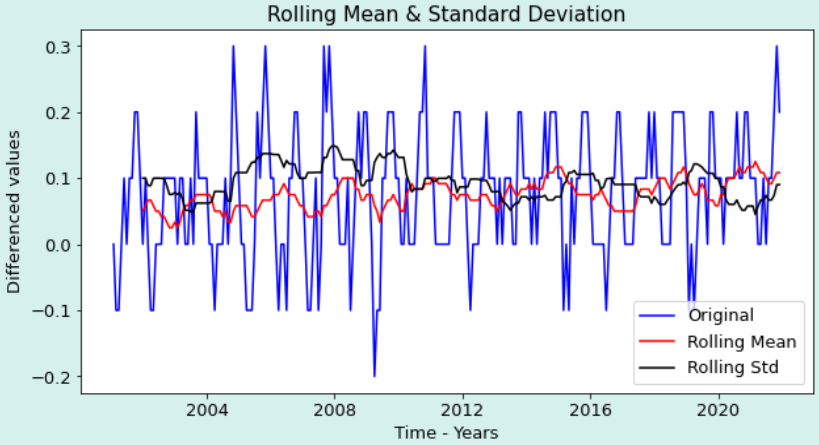
CH_4




```
Results of Dickey-Fuller Test:
Test Statistic                  -3.089793
p-value                          0.027318
#Lags Used                      12.000000
Number of Observations Used    448.000000
Critical Value (1%)             -3.445031
Critical Value (5%)             -2.868013
Critical Value (10%)            -2.570218
dtype: float64
The time series is stationary at 95% level of confidence
```

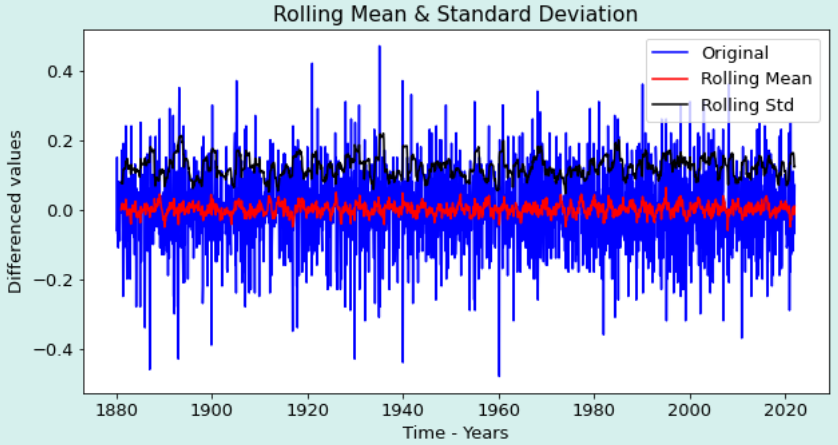
SARIMA: Stationarity & Differencing

N_2O




```
Results of Dickey-Fuller Test:
Test Statistic                  -3.017139
p-value                          0.033325
#Lags Used                      12.000000
Number of Observations Used    238.000000
Critical Value (1%)             -3.458128
Critical Value (5%)             -2.873762
Critical Value (10%)            -2.573283
dtype: float64
The time series is stationary at 95% level of confidence
```

Temperature Change




```
Results of Dickey-Fuller Test:
Test Statistic                 -1.236645e+01
p-value                         5.400854e-23
#Lags Used                      2.300000e+01
Number of Observations Used     1.680000e+03
Critical Value (1%)            -3.434248e+00
Critical Value (5%)            -2.863262e+00
Critical Value (10%)           -2.567687e+00
dtype: float64
The time series is stationary at 95% level of confidence
```

SARIMA: Parameters Tuning

```python
# Grid-search function for finding the p, q and seasonal parameters P, Q for the SARIMA model

def optimize_SARIMA(parameters_list, d, D, s, exog):
    """
        Return dataframe with parameters, corresponding AIC and SSE

        parameters_list - list with (p, q, P, Q) tuples
        d - integration order
        D - seasonal integration order
        s - length of season
        exog - the exogenous variable
    """

    results = []

    for param in tqdm_notebook(parameters_list):
        try:
            model = SARIMAX(exog, order=(param[0], d, param[1]),
                            seasonal_order=(param[2], D, param[3], s)).fit(disp=-1)
        except:
            continue

        aic = model.aic
        results.append([param, aic])

    result_df = pd.DataFrame(results)
    result_df.columns = ['(p,q)x(P,Q)','AIC']

    #Sort in ascending order, lower AIC is better
    result_df = result_df.sort_values(by='AIC', ascending=True).reset_index(drop=True)

    return result_df
```

```python
p = range(0,4,1)
d = 1
q = range(0,4,1)

P = range(0,4,1)
D = 1
Q = range(0,4,1)

s = 12

parameters = product(p, q, P, Q)
parameters_list = list(parameters)
```

SARIMA: Forecasting

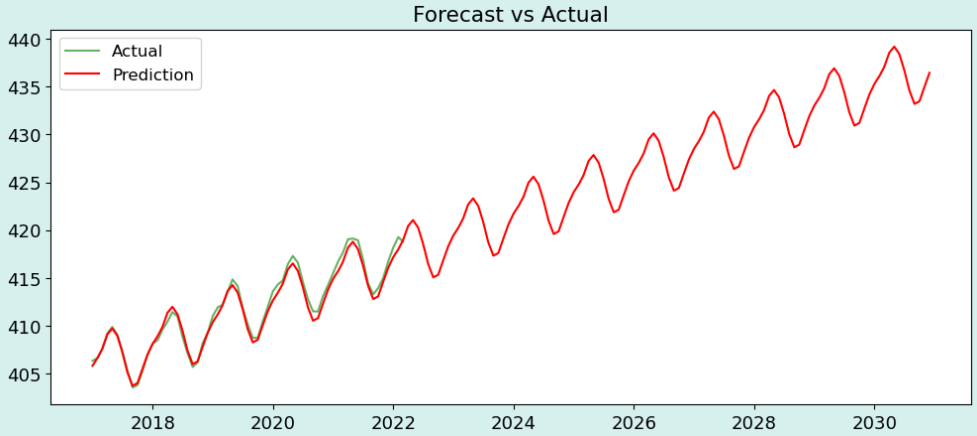
Datasets	Non-Seasonal (p,d,q)	Seasonal (P,D,Q) m
CO_2	(1,1,1)	(0,1,1) 12
CH_4	(3,1,3)	(0,1,1) 12
N_2O	(1,1,0)	(0,1,1) 12
Temperature change	(1,1,2)	(2,1,2) 12

```python
# Function for splitting the train and test
def train_test_split(timeseries, lags_for_prediction):
    split=len(timeseries)-lags_for_prediction
    train=timeseries[:split]
    test=timeseries[split:]
    return train, test

train_co2, test_co2 = train_test_split(co2, 63)
```

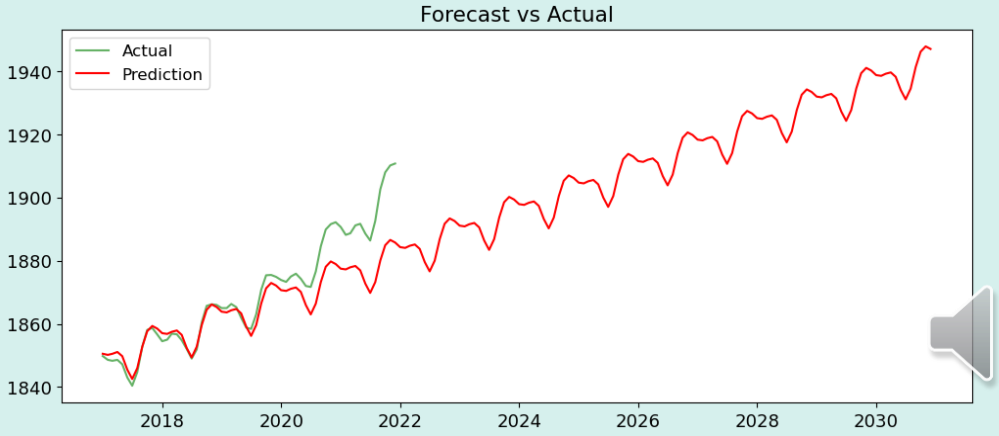
```python
def forecasting (p,d,q,P,D,Q,season, lags_for_forecast, train, test):

    model = SARIMAX(train, order=(p,d,q), seasonal_order=(P,D,Q,season),
                                          simple_differencing=0, #if True time series provided
                                          enforce_stationarity=True,
                                          enforce_invertibility=False)
    fitted = model.fit(disp=-1)

    start = '2017-01-01'
    end = '2030-12-01'

    #start = len(train)
    #end = len(train) + len(test)-1

    # Forecast
    forecast = fitted.predict(start=start,end=end,typ='levels')

    # Plot
    plt.figure(figsize=(12,5), dpi=100)
    #plt.plot(train, color='blue', label='train')
    plt.plot(test, color='green', label='Actual', alpha=0.6)
    plt.plot(forecast, color='red', label = 'Prediction')
    plt.title('Forecast vs Actual')
    plt.legend(loc='upper left', fontsize=12)
    plt.show()

    from sklearn import metrics
    from math import sqrt

    #print('\033[1m' + 'MAE: ' + '\033[0m', metrics.mean_absolute_error(test, forecast))
    #print('\033[1m' + 'MSE: ' + '\033[0m', metrics.mean_squared_error(test, forecast))
    #print('\033[1m' + 'RMSE:' + '\033[0m', np.sqrt(metrics.mean_squared_error(test, forecast)))
    #print('\033[1m' + 'MEAN:' + '\033[0m', test.mean())

    #RSS=np.sqrt(sum(forecast.values-test['average'].values.reshape(-1))**2)/lags_for_forecast

    print(fitted.summary())

    return forecast

forecast_co2 = forecasting (1,1,1, 0,1,1, 12, 63, train_co2['average'], test_co2['average'])
```

SARIMA: Model Plots

CO_2 Plot

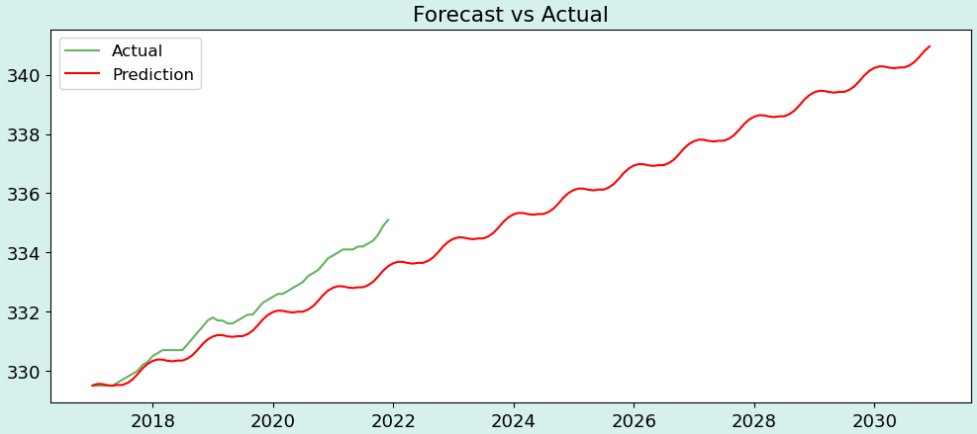


CH_4 Plot

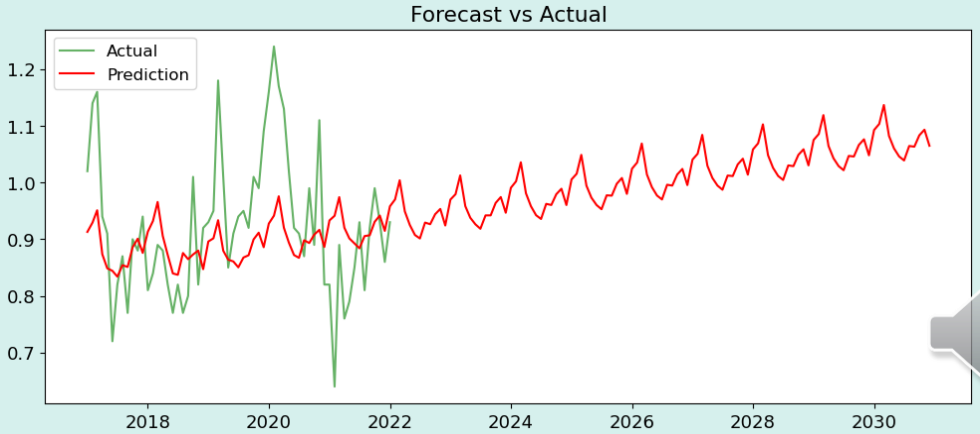


SARIMA: Model Plots

N_2O Plot



Temperature Change Plot

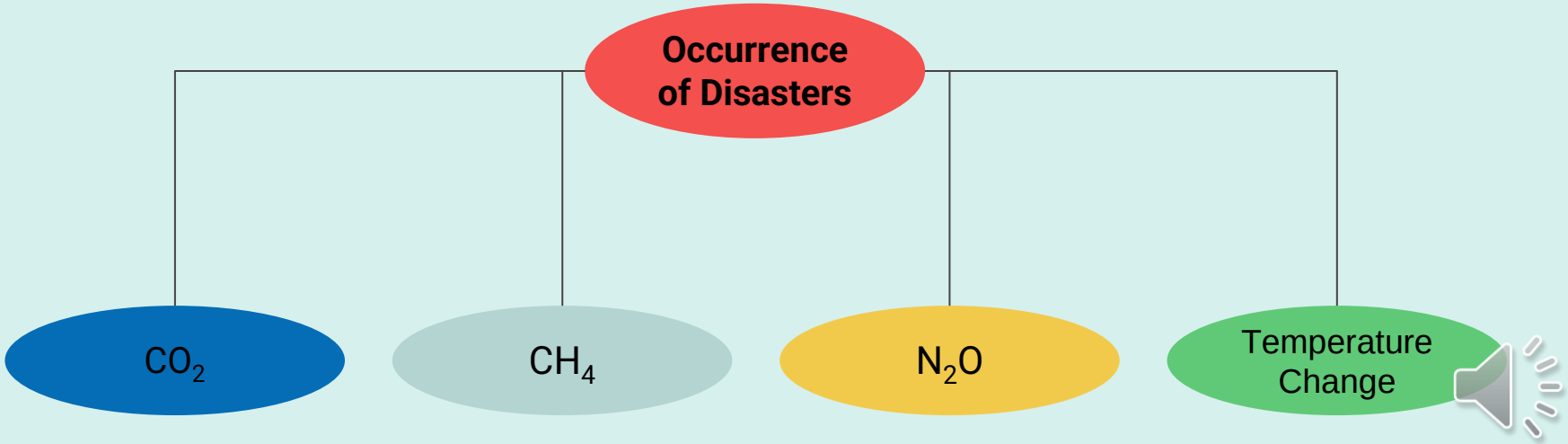


Multivariate Linear Regression

Generalised Equation:

$$y = \beta 0 + \beta 1.x1 + \beta 2.x2 + \ldots\ldots + \beta n.xn$$

Dependent Variable: y | **Independent Variables:** x1 ~ xn | **Coefficients:** β0 ~ βn



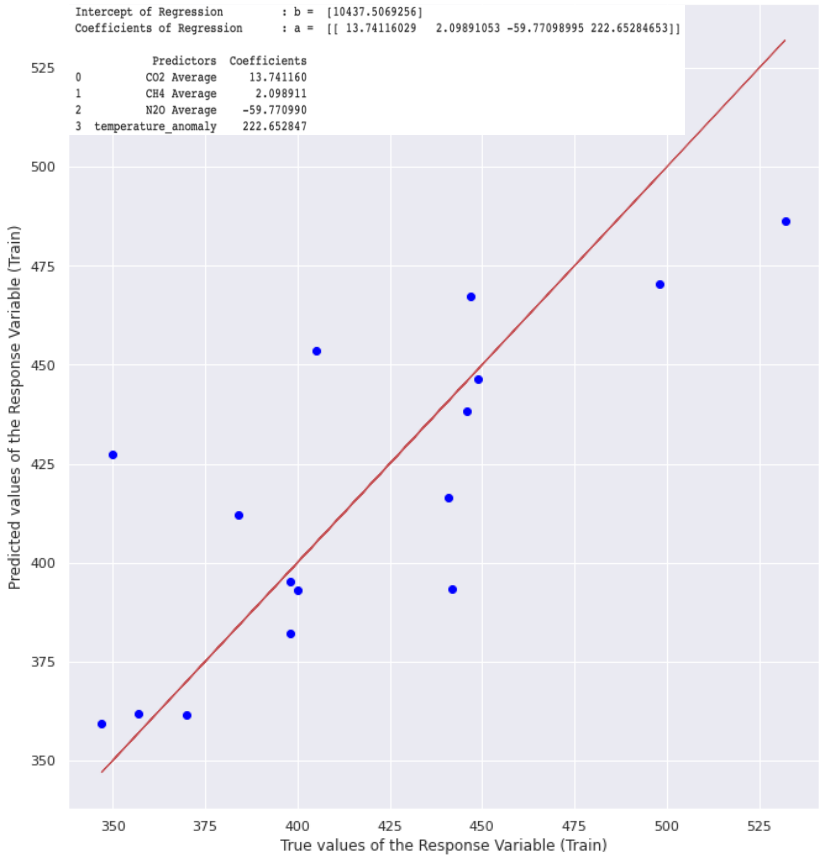
Multivariate Linear Regression

Combined Dataset:

	CO2 Average	CH4 Average	N2O Average	temperature_anomaly	No. Of Disasters
2001	371.319167	1771.275000	316.366667	0.535000	447
2002	373.452500	1772.741667	316.933333	0.627500	532
2003	375.983333	1777.400000	317.625000	0.615833	392
2004	377.698333	1777.008333	318.258333	0.533333	405
2005	379.983333	1774.191667	318.933333	0.675833	498
2006	382.090833	1775.033333	319.833333	0.635000	446
2007	384.025000	1781.525000	320.466667	0.661667	449
2008	385.830833	1787.091667	321.516667	0.540000	400
2009	387.643333	1793.600000	322.266667	0.653333	384
2010	390.101667	1799.008333	323.191667	0.718333	441
2011	391.850833	1803.166667	324.225000	0.604167	357
2012	394.055833	1808.116667	325.066667	0.645000	370
2013	396.737500	1813.450000	325.950000	0.674167	352
2014	398.870833	1822.741667	327.091667	0.744167	347
2015	401.012500	1834.366667	328.175000	0.895833	398
2016	404.412500	1843.225000	328.958333	1.015833	350
2017	406.758333	1849.741667	329.750000	0.922500	371
2018	408.715000	1857.433333	330.925000	0.845833	338
2019	411.654167	1866.691667	331.875000	0.977500	442
2020	414.238333	1879.250000	333.033333	1.019167	398

Multivariate Linear Regression (Model 1)

```
Intercept of Regression        : b =  [10437.5069256]
Coefficients of Regression     : a =  [[ 13.74116029   2.09891053 -59.77098995 222.65284653]]

             Predictors  Coefficients
0          CO2 Average     13.741160
1          CH4 Average      2.098911
2          N2O Average    -59.770990
3  temperature_anomaly    222.652847
```



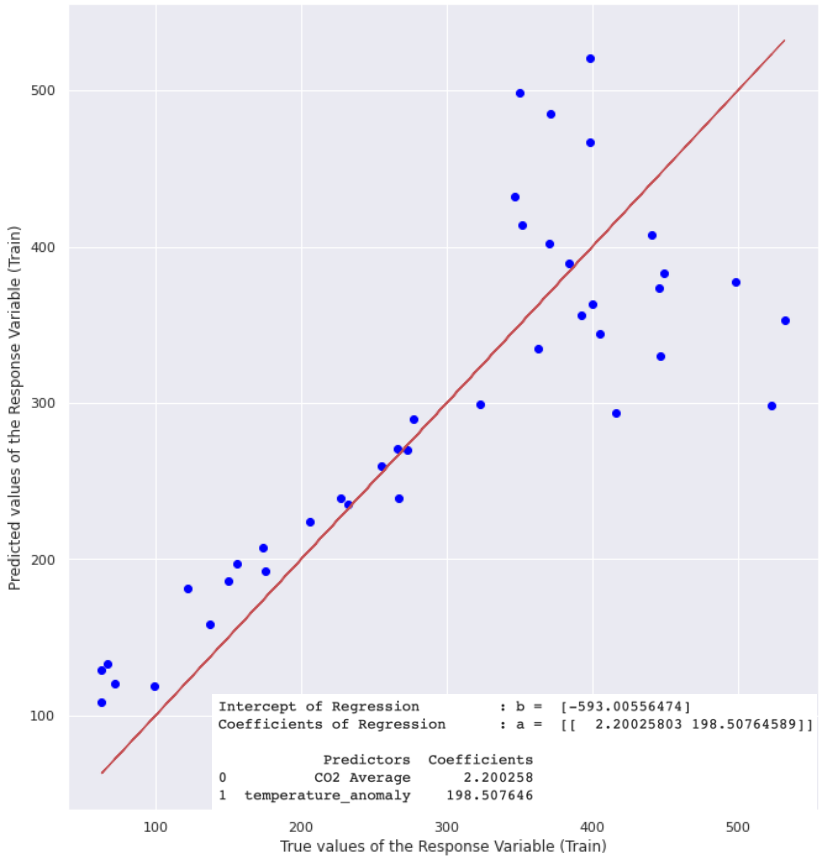

```
Goodness of Fit of Model                    Train Dataset
Explained Variance (R^2)                    : 0.6118267149600654
Mean Squared Error (MSE)                    : 999.3036006746815
Root Mean Squared Error (RMSE)   : 31.61176364385071

Goodness of Fit of Model                    Test Dataset
Explained Variance (R^2)                    : -5.559433554388047
Mean Squared Error (MSE)                    : 2706.996234976517
Root Mean Squared Error (RMSE)   : 52.028801975218656
```

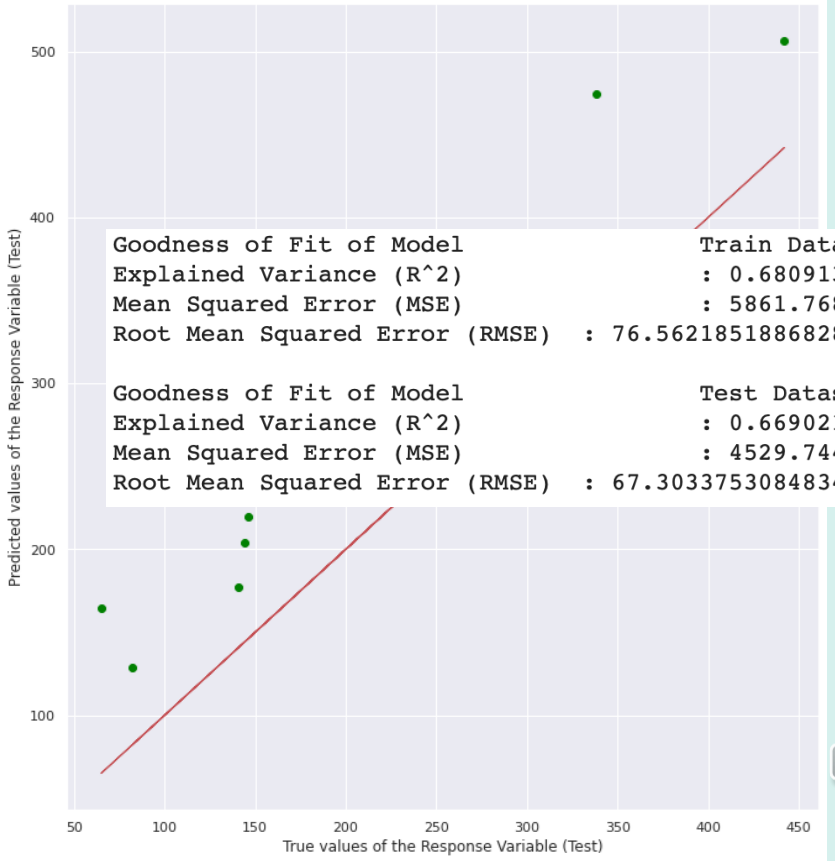
Multivariate Linear Regression (Model 2)

Updated Dataset after removal of CH_4 and N_20:

	CO2 Average	temperature_anomaly	No. Of Disasters
1970	325.681667	0.025833	82
1971	326.319167	-0.081667	63
1972	327.456667	0.009167	63
1973	329.677500	0.161667	65
1974	330.191667	-0.067500	72
1975	331.115833	-0.011667	67
1976	332.026667	-0.095833	99
1977	333.843333	0.178333	141
1978	335.415000	0.068333	137
1979	336.835833	0.165833	122
1980	338.762500	0.260000	144
1981	340.119167	0.322500	146
1982	341.479167	0.140000	150
1983	343.152500	0.314167	206
1984	344.848333	0.156667	156
1985	346.352500	0.118333	175
1986	347.608333	0.180833	174
1987	349.312500	0.321667	227
1988	351.690833	0.390000	234
1989	353.205000	0.271667	189
1990	354.453333	0.450000	303
1991	355.704167	0.406667	266
1992	356.544167	0.220833	232

Multivariate Linear Regression (Model 2)

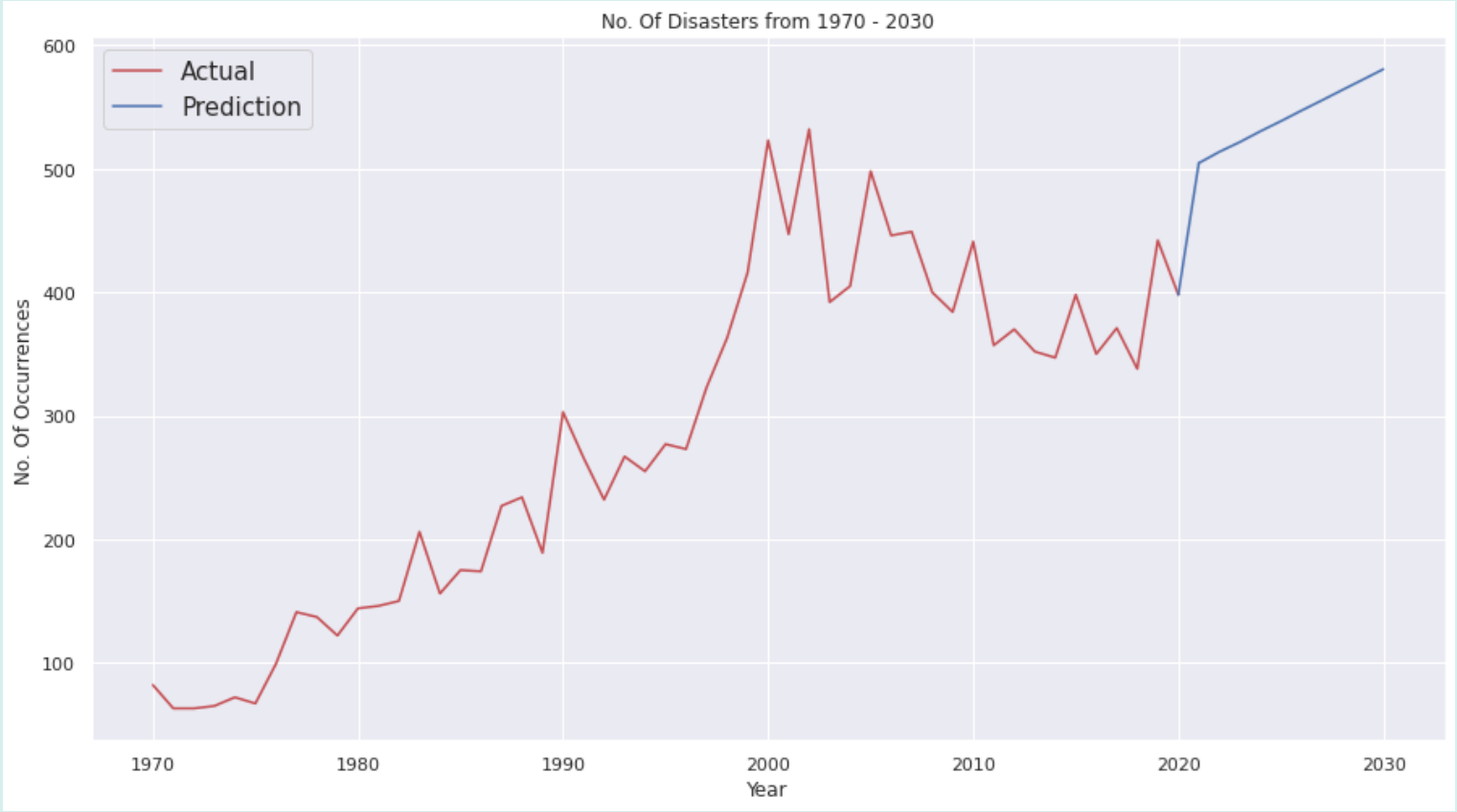




Multivariate Linear Regression (Model 2) Predictions

	CO2 Average	temperature_anomaly	Predicted Amount of Natural Disasters
2021	415.783164	0.920804	504.611409
2022	418.047540	0.941394	513.680797
2023	420.311916	0.956269	521.615845
2024	422.576292	0.975124	530.340922
2025	424.840668	0.991229	538.520125
2026	427.105044	1.009238	547.077279
2027	429.369421	1.025930	555.372887
2028	431.633797	1.043533	563.849512
2029	433.898173	1.060506	572.200858
2030	436.162549	1.077915	580.638908

Multivariate Linear Regression (Model 2)



Multivariate Linear Regression Models

Model 1

Predictors: CO2, CH4, N2O, Temperature Change

```
Goodness of Fit of Model                    Train Dataset
Explained Variance (R^2)                    : 0.6118267149600654
Mean Squared Error (MSE)                    : 999.3036006746815
Root Mean Squared Error (RMSE)   : 31.61176364385071

Goodness of Fit of Model                    Test Dataset
Explained Variance (R^2)                    : -5.559433554388047
Mean Squared Error (MSE)                    : 2706.996234976517
Root Mean Squared Error (RMSE)   : 52.028801975218656
```

Model 2

Predictors: CO2, Temperature Change

```
Goodness of Fit of Model                    Train Dataset
Explained Variance (R^2)                    : 0.6809136724472099
Mean Squared Error (MSE)                    : 5861.768200866161
Root Mean Squared Error (RMSE)   : 76.5621851886828

Goodness of Fit of Model                    Test Dataset
Explained Variance (R^2)                    : 0.6690211053435802
Mean Squared Error (MSE)                    : 4529.744327914573
Root Mean Squared Error (RMSE)   : 67.3033753084834
```

Comparison of Multivariate Linear Regression Models 1 and 2

Summary:

Model 2 is the favoured model

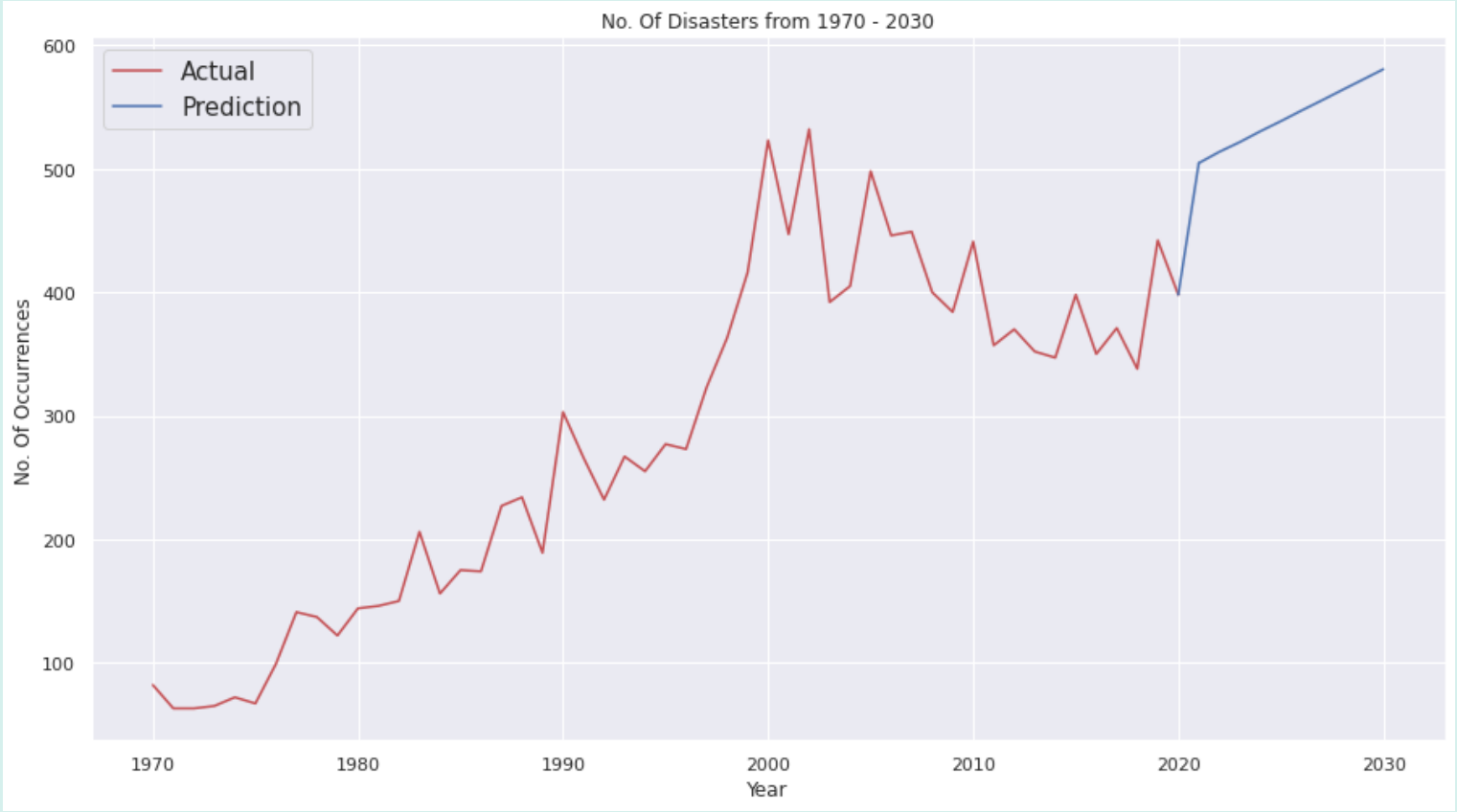
Model 1 (Predictors: CO2, CH4, N2O, Temperature Change):

- Very **poor and fluctuating R^2 value** when model was ran multiple times, due to low amount of data points
    - Only used data from 2001, due to N20 dataset

Model 2 (Predictors: CO2, Temperature Change):

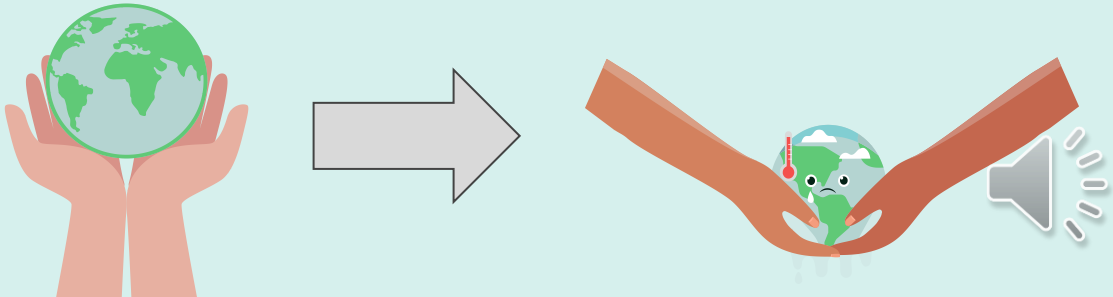
- **Stable R^2 value** even when model was ran multiple times, **decent goodness of fit** (RMSE)
    - Used data since 1970

Finalised Multivariate Linear Regression Model (Model 2)

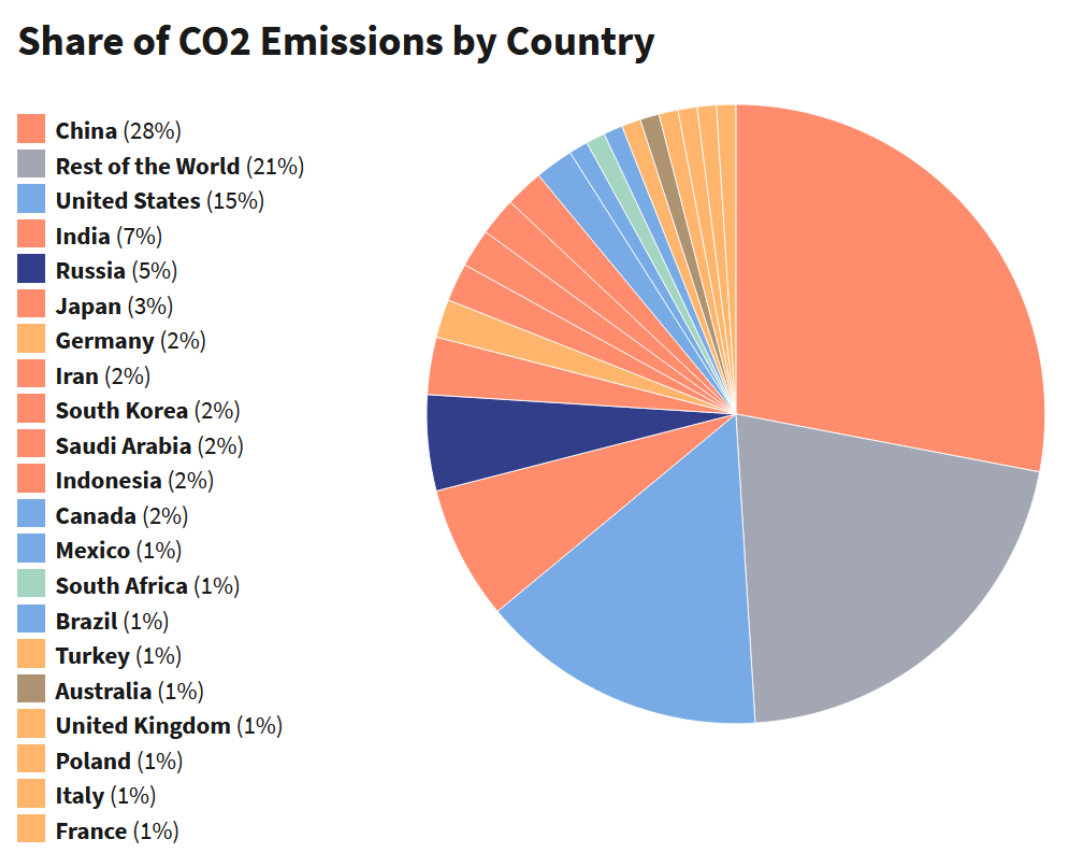


Data-Driven Insights and Recommendations

1. **Increased greenhouse gases will lead to global warming**, which in turn/concurrently, leads to the increase in occurrence of natural disasters
2. From our forecasted data, this **trend will continue to increase throughout from 2021 to 2030** if no actions are taken to reduce the emission of greenhouse gases
3. This climate change will cause an **increase in natural disasters such as droughts, floods and forest fires** for already affected countries based on our regression model



Data-Driven Insights and Recommendations



Data-Driven Insights and Recommendations

United Nations ActNow:

To preserve a livable climate, greenhouse-gas emissions must be reduced to net zero by 2050 - **stabilise CO2 gas concentrations in the atmosphere**

- In accordance with our model, this would **reduce frequency of natural disasters** and at the same time, slow down global warming

Challenges Faced and Learning Outcomes

We found that…

1. ARIMA forecasting model **does not account for seasonality** in the time series data
2. The best hyperparameters for the model was obtained by using **grid search** and selected based on the **lowest AIC** value
3. SARIMA model was used as it **accounts for seasonality** in the time series data
4. The accuracy of the prediction of greenhouse gases and temperature change by using SARIMA forecasting model is determined by the **RMSE** value
5. Multivariate Linear Regression is then used to **estimate the natural disaster occurrences from the predicted values** given by the SARIMA forecasting model
6. Best estimation of the regression model is determined by the metrics (**RMSE, R^2**) values

Conclusion

We learnt that…

- Data Science and Machine Learning is all about **experimenting with data and different tools**
- Data Science and Machine Learning is about making **data-driven decisions**
- There is **no one path to your goal**